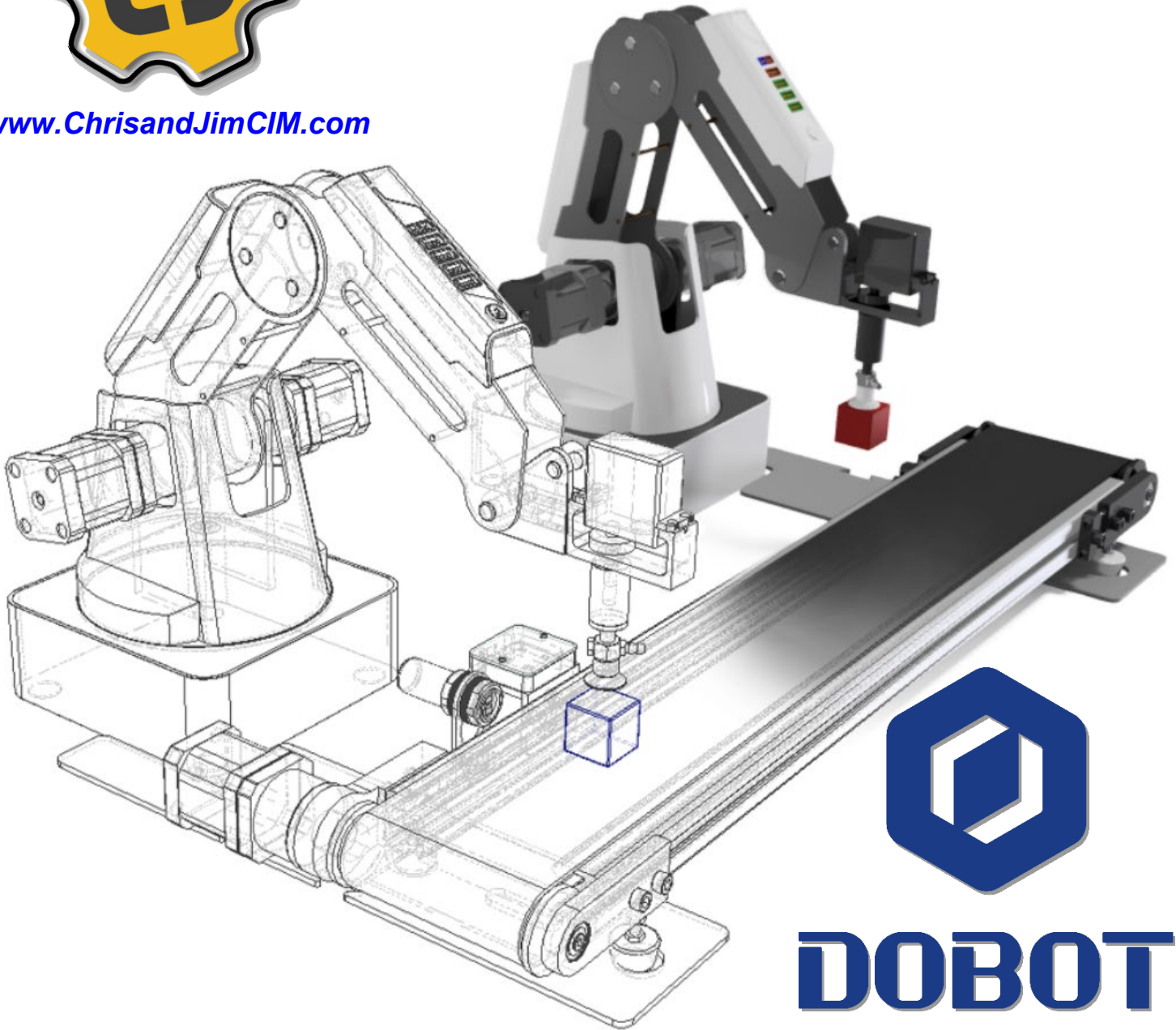


Intro to Robotics V3



Dobot Magician

www.ChrisandJimCIM.com



DOBOT

www.dobot-robots.com

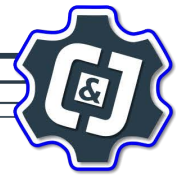


TABLE OF CONTENTS

How to use this curriculum

This curriculum was designed to teach middle school, high school, and college level students the basics of robotics, as used in industry. All concepts taught within can be done most easily with the Dobot Magician, Magician Light, Sensor Kits, & Peripherals.



Chapters with this icon denote that they are designed to be used with **DobotBlock** programming in the **DobotLab** software.



Chapters with this icon denote that they are designed to be used with **Teaching & Playback** programming in the **DobotLab** software.



Chapters with this icon denote that they are designed to be used with **DobotBlock** programming with the in the **DobotLab** software with the **Magician Sensor Kit**.

Table of Contents			
1	<i>Robot Movement</i>		6
	 <i>1.1 T&P - Robot Axis and Motion</i>		7
	 <i>1.2 T&P - Pick & Place Routines</i>		21
	 <i>1.3 Block - Pick & Place Routines</i>		29
	 <i>1.4 T&P - Using Jumps & Loops</i>		42
	 <i>1.5 Block - Pick & Place with Jumps & Loops</i>		50
	 <i>1.6 T&P - Palletization Routines</i>		58
	 <i>1.7 Block - Creating a Cube Matrix</i>		64



2	<i>Robot Communication</i>	<i>82</i>
	 <i>T&P - Dobot to Dobot Handshaking</i>	<i>83</i>
	 <i>Block - Dobot to Dobot Handshaking</i>	<i>90</i>
	 <i>Block - Dobot to Dobot: Internal Communication</i>	<i>105</i>
	 <i>T&P - Dobot to Microcontroller Handshaking</i>	<i>114</i>
	 <i>Block - Dobot to Microcontroller Handshaking: DobotBlock</i>	<i>124</i>
	 <i>VEXCode V5 - Microcontroller to Microcontroller Communication</i>	<i>135</i>
	 <i>Block - Magic Box to Magic Box Wireless Communication</i>	<i>144</i>

3	<i>Robotics & The Use of Sensors</i>	<i>152</i>
	 <i>T&P - Digital Inputs: The Microswitch</i>	<i>153</i>
	 <i>Block - Pick and Place with Inputs</i>	<i>161</i>
	 <i>Block - Introduction to Inputs and Outputs (Sensor Kit)</i>	<i>177</i>
	 <i>Block - Pick and Place with the Magic Box (Sensor Kit)</i>	<i>188</i>
	 <i>Block - Sensor Exploration (Sensor Kit)</i>	<i>196</i>
	 <i>Block - Introduction to the Color Sensor (Sensor Kit)</i>	<i>218</i>
	 <i>Block - Sorting with the Color Sensor (Mini Conveyor Kit)</i>	<i>226</i>



4	<i>Robotic Scenarios & the Use of Peripherals</i>		<i>237</i>
		<i>Block - Conveyor Systems Basics: Motion (Mini Conveyor Kit)</i>	<i>238</i>
		<i>Block - Pick and Place with the Linear Slidebase (Sliding Rail Kit)..... ..</i>	<i>247</i>
		<i>Block - Stack Light with the Magic Box (Sensor Kit)</i>	<i>256</i>
		<i>Block - AI Camera Vision System (Sensor Kit)</i>	<i>261</i>
	ALL	<i>T&P / Block - Work Cell Design Without a Microcontroller</i>	<i>271</i>
	ALL	<i>T&P / Block - Work Cell Design With a Microcontroller</i>	<i>275</i>
	ALL	<i>T&P / Block - Work Cell Design Scenario</i>	<i>281</i>

5	<i>Robotics Resources and Printable Help Files</i>		<i>285</i>
	ALL	<i>Vertical Field Diagram</i>	<i>286</i>
	ALL	<i>Horizontal Field Diagram</i>	<i>287</i>
		<i>Color Test Template</i>	<i>288</i>
	ALL	<i>Guide to Connecting Inputs and Outputs to the Dobot</i>	<i>289</i>
		<i>Guide to Connecting Peripherals to the Dobot</i>	<i>291</i>
		<i>Guide to Connecting Sensors & the Magic Box to the Dobot</i>	<i>292</i>



	ALL	<i>Wiring Guide for Work Cell Set Up</i>	294
	ALL	<i>Glossary: Industrial Robotics Terms</i>	295
		<i>Glossary: DobotBlock Terminology</i>	299
	ALL	<i>Educational Standards Addressed in This Curriculum</i>	306



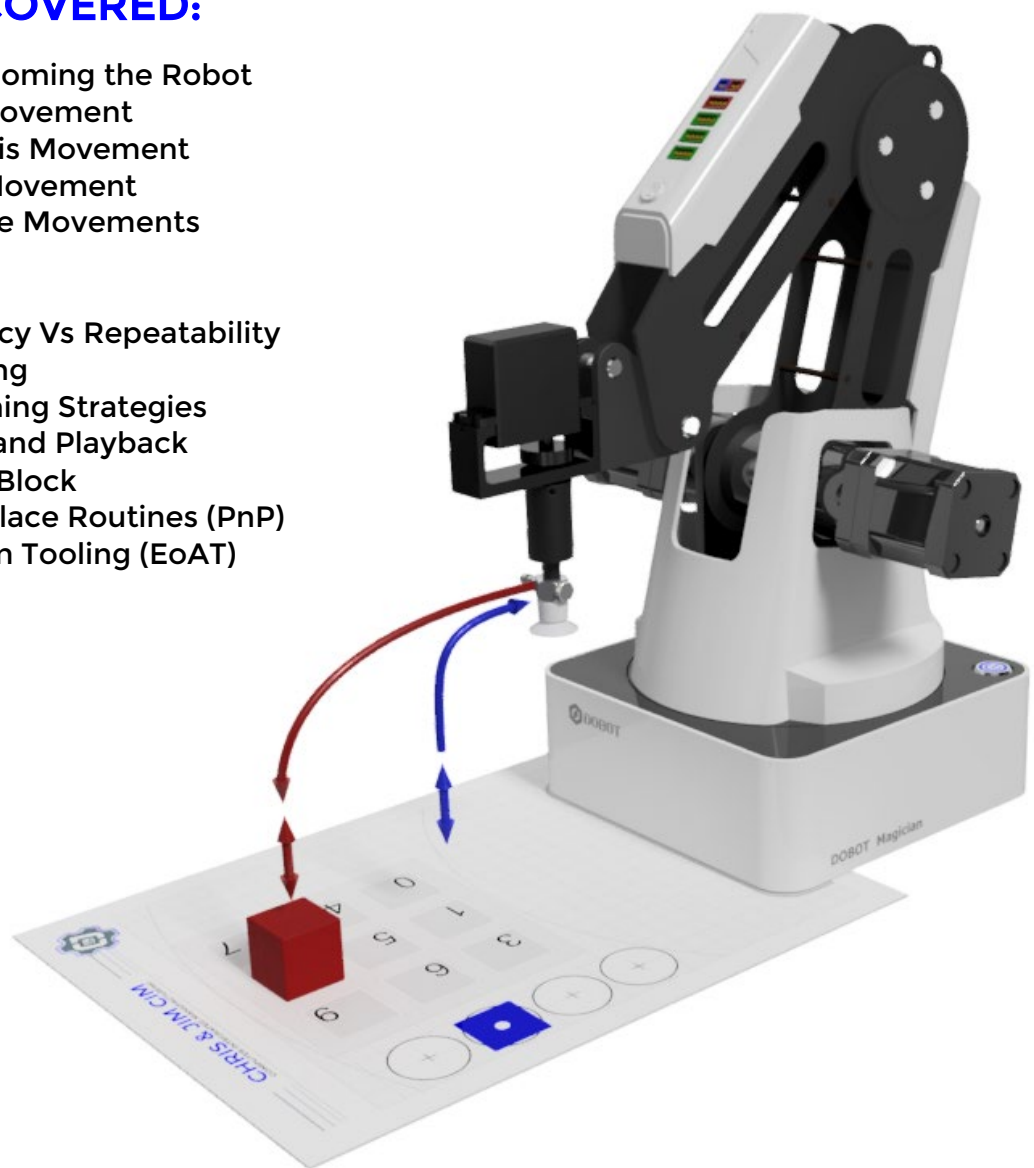
- CHAPTER 1 - ROBOT MOVEMENT

CHAPTER INTRODUCTION:

Jointed arm robots are useful for many different tasks due to their range of motion and degrees of freedom. This chapter will illustrate how to both move and program a robotic arm in multiple ways.

TOPICS COVERED:

- Wiring & Homing the Robot
- Robotic Movement
 - XYZ Axis Movement
 - Joint Movement
 - Relative Movements
 - Jumps
 - Loops
 - Accuracy Vs Repeatability
 - Ramping
- Programming Strategies
 - Teach and Playback
 - Dobot Block
- Pick and Place Routines (PnP)
- End of Arm Tooling (EoAT)



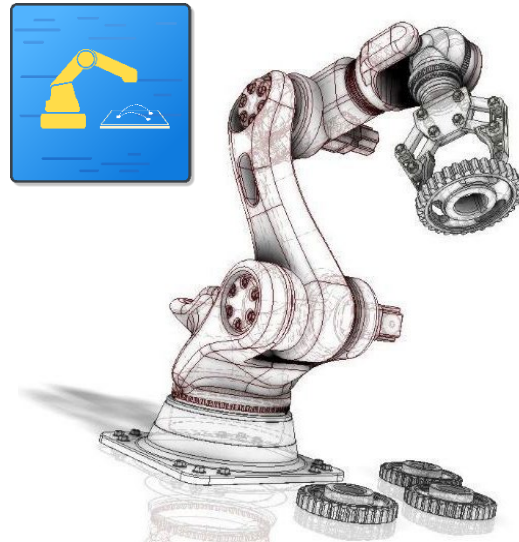
1.1 TP - Robot Axis and Movement

NAME: _____ Date: _____ Section: _____

INTRODUCTION

Jointed arm robots are useful for many different tasks because of its range of motion and degrees of freedom. In this activity, you will learn how to move a robotic arm in many different ways and write a program to make the robot write the word "CIM" with **ACCURACY** and **REPEATABILITY**. CIM stands for Computer Integrated Manufacturing.

The method of measurement and positioning we will use to do this is called **RELATIVE COORDINATES**. We will have the robot move the pen "relative" to where it was last. We will also use a method of saving points called **TEACHING**. This is where we type in coordinates without having to move the robot arm.



KEY CONCEPTS

- Differentiate ways that robot arms can move: **Move Linear** and **Move Joint**.
- Differentiate between **absolute** and **relative** coordinates.
- Differentiate between **teaching** and **recording** points.
- Starting up and connecting the Dobot Magician.
- How to utilize a robot arm to move through a group of points by using the pen **end effector**.
- How to use DobotLab Teach and Playback Module.

KEY VOCABULARY

- | | |
|------------------------|-----------------------------|
| • Relative Coordinates | • Teach |
| • Joint movement | • Linear move |
| • Axis movement | • Home |
| • Work envelope | • Loop |
| • Ramping | • End Effector |
| • Accuracy | • End of Arm Tooling (EoAT) |

EQUIPMENT & SUPPLIES

- Dobot Magician
- Dobot Field Diagram
- Pen end effector bracket
- DobotStudio software
- Pen
- Masking tape

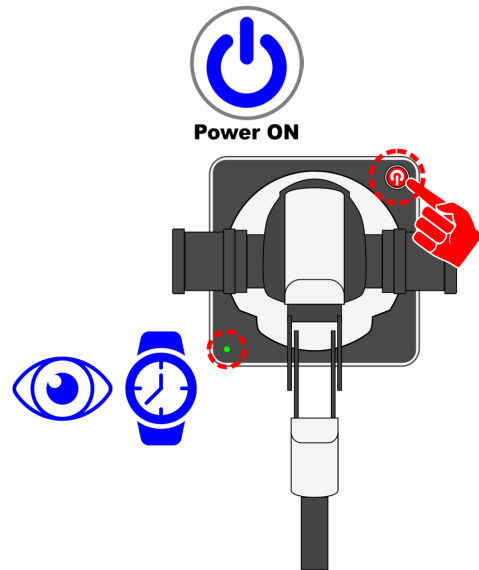
PROCEDURE



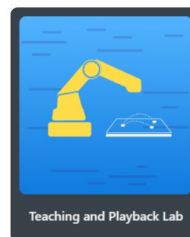
Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur.

1. Typical Start Up Procedure

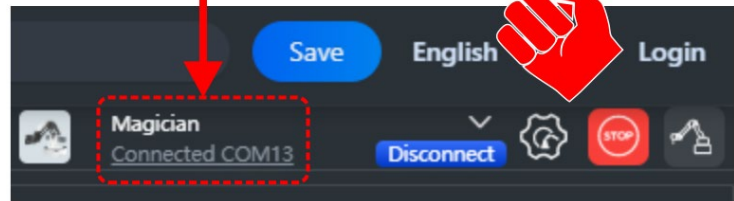
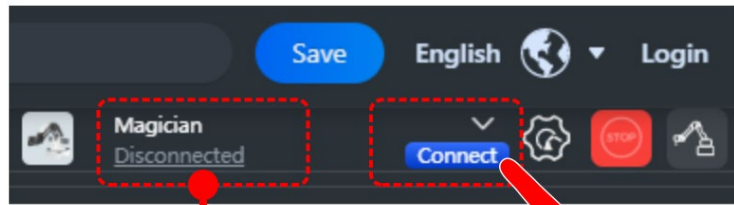
- Attach the Pen tool as the **END EFFECTOR** or **END of ARM TOOLING** on the Dobot.
- Plug the 120-240VAC power into an outlet.
- Attach the 12VDC 7A barrel plug of the power supply and USB to the Dobot. Plug the USB into the computer and wait for a connection. **WATCH** and **WAIT** for the robot to completely power on (GREEN INDICATOR)
- Turn on the power to the Dobot (Robot will beep when ready).
- Open DobotLab software.



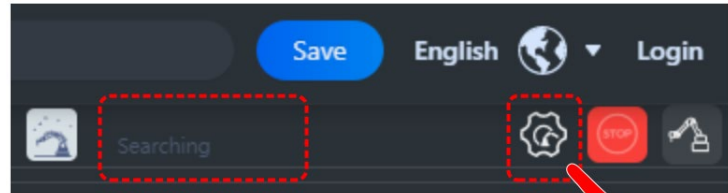
2. Once the Robot's USB is connected to the computer and the robot is powered on, open the DobotLab software
3. Once the software is opened select the Teaching and Playback Lab.



4. Select connect to establish control from the software to the robot. Disconnected should change to Connected and show the COM being used (COM is the USB port being used by the robot)



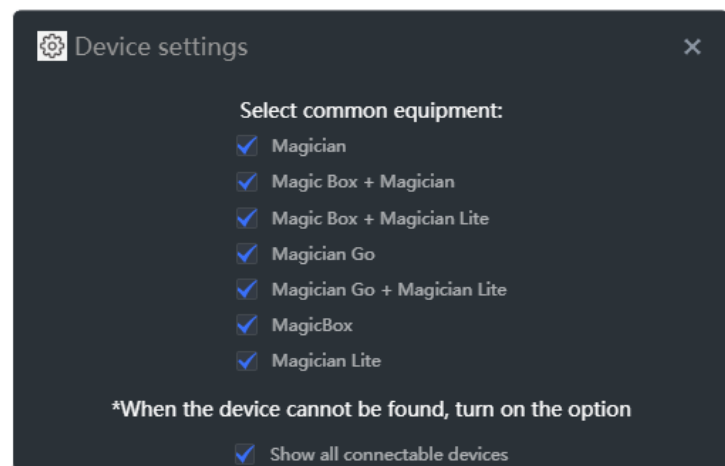
5. NOTE: If no robot is found, the prompt will show as Searching



SEARCHING... Complete the steps below if no robot is found.

Select the Gear Icon next to the Force Stop Icon .

Make sure the hardware you are connected is checked .

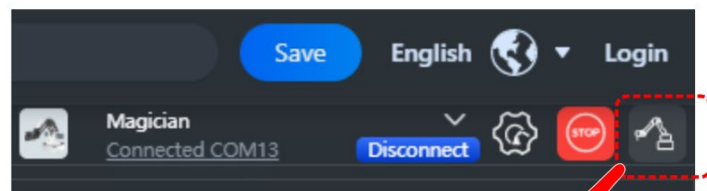


6. Select the **ARM CONTROL** Icon to open the arm control panel.

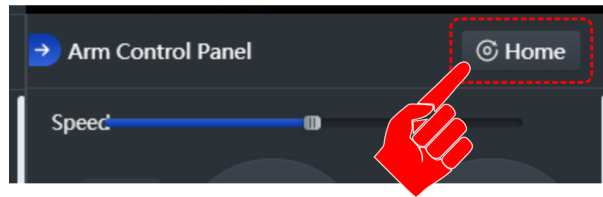


Before **HOMING** the robot, make sure the robot's **WORK ENVELOPE**, the area in which the robot can reach is clear.

7. Select the **HOME** icon to start the robots homing process.



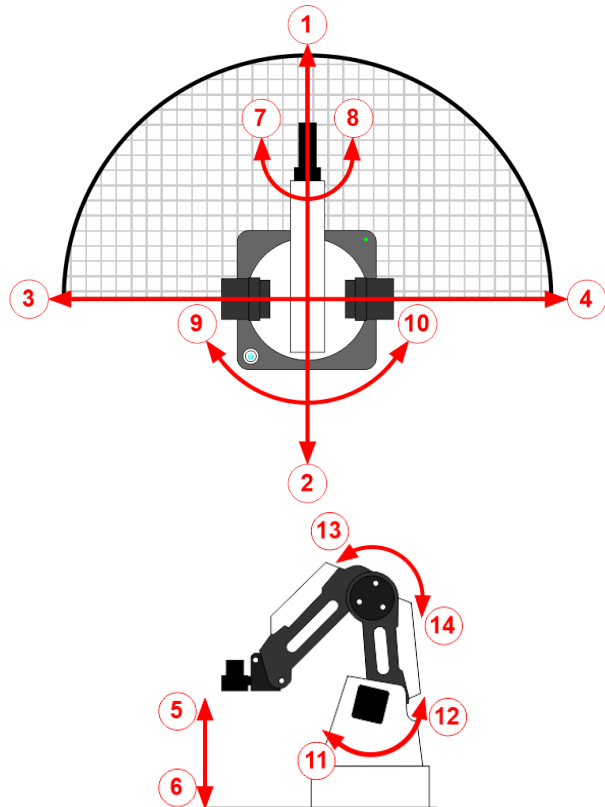
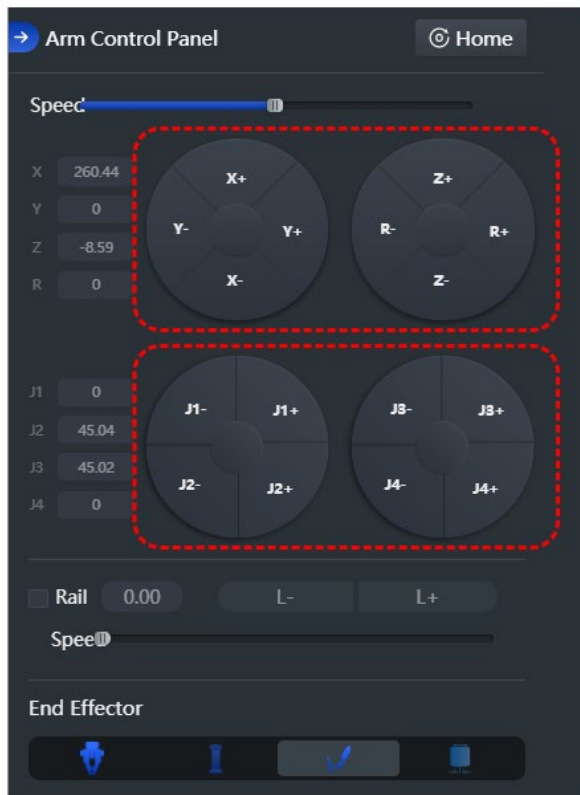
HOMING the robot will return the robot to its initial **HOME** position.



8. Be sure the **Pen** is chosen as the Dobot's **END EFFECTOR** or **EoAT** (End of Arm Tooling).



9. Open the manual control panel and use the Axis and Joint buttons to move the robot around. Using the chart diagram provided, identify the **AXIS MOVEMENT** and **JOINT MOVEMENTS** for the robot (1-14). Be sure to label them as +/-.



Using the information you documented on the previous diagram, write down the correct axis/joint button in the first empty column, and a description of what it does in the second.

		AXIS / JOINT + / -	DESCRIPTION IN / OUT / UP / DN / LEFT / RIGHT
1	Axis		
2	Axis		
3	Axis		
4	Axis		
5	Axis		
6	Axis		
7	Wrist		
8	Wrist		
9	Waist		
10	Waist		
11	Shoulder		
12	Shoulder		
13	Elbow		
14	Elbow		

10. Are the axis defined as you expected? Explain:

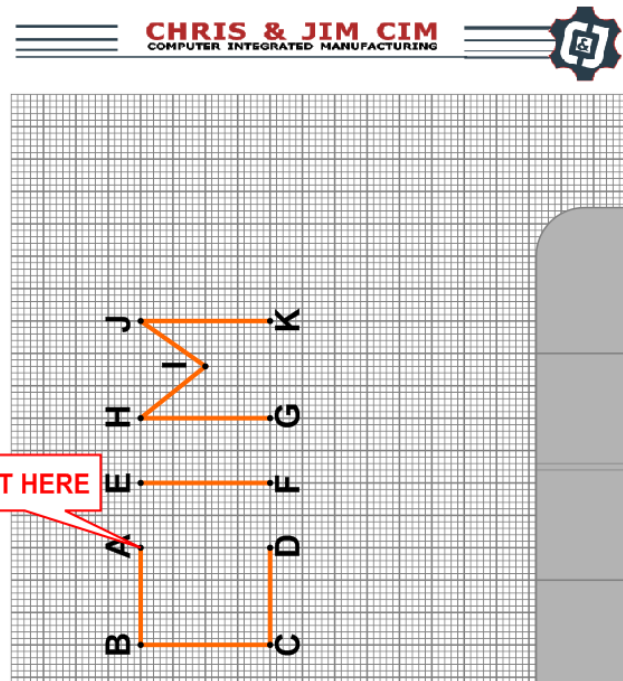
11. How do the XYZ movements differ from the J movements?



12. MAIN OBJECTIVE:

First, handwrite the word "CIM" in pencil on the Dobot Template as shown (see step 10 for segment dimensions). Second, using the Dobot with the pen end effector, start at point A on your diagram and move the robot from point to point to re-write the word using straight lines. Be sure to pick up the pen between letters and send the robot to a position away from the paper when finished.

START HERE



13. What do you think is this the best way to write the letters CIM (order of movement)? Did you take time into consideration? How?

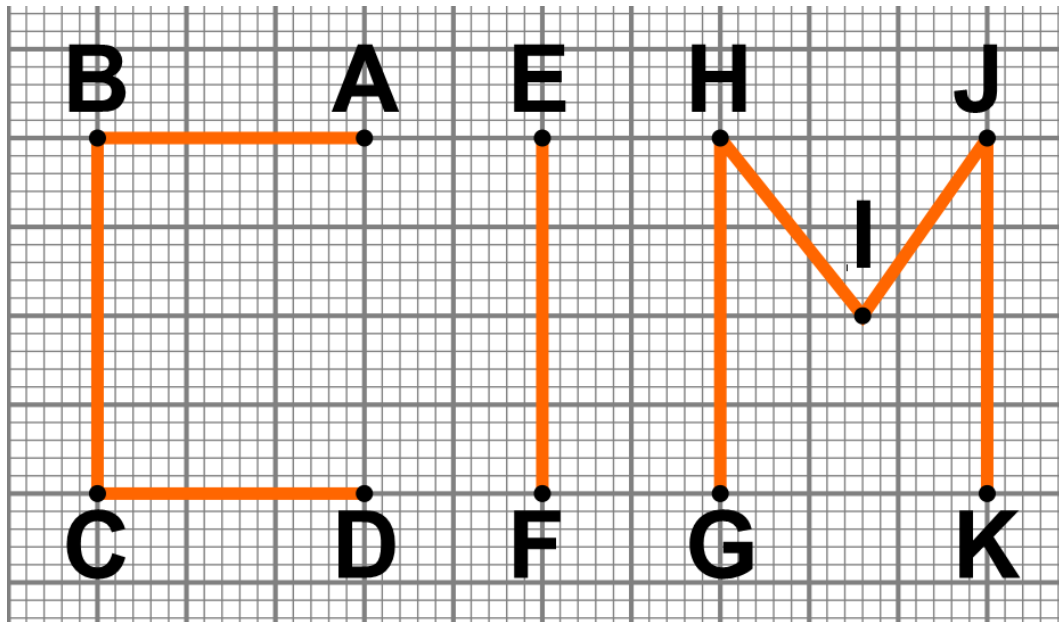


When finding the initial points for the pen, keep the cap on the pen.

Notice that the pen is spring loaded; this gives the pen a softer touch to the paper and allows for the same points to work on an uneven or irregular surface. This also allows the pen not to smash into the table and break it. When moving in the z-axis though, try to put just light pressure (about half of the spring position) on it when you write, or it may tear the paper or break the tip.

14. For this activity, we will use a combination of recorded and taught positions. We will teach the robot the points shown using actual coordinates. The big squares on the paper are equal to about 10mm, this will help us plan our letters.





15. Line Segment lengths are as follows:

AB = 30mm

BC = 40mm

CD = 30mm

EF = 40mm

GH = 40mm

*HI = ???mm

*IJ = ???mm

JK = 40mm

**what is the length of these two line segments? How do you know? Explain below.*

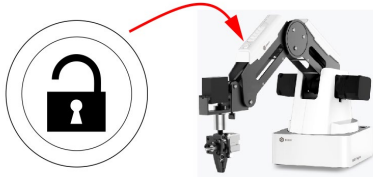
16. Make sure you are in the **Teach and Playback** lab.

We will use this lab to **RECORD** and **TEACH** the robot the points we want it to go to write the word CIM.



There are two methods to RECORDING a point.

Method 1: RECORDED points can be found using the lock button on the robot's arm. When the lock button is pressed, the arm becomes free to manually move to a new location. The location is automatically **RECORDED** and added as a step in the program when the lock is released. No manual movements in between the steps (while the button is held) will be recorded.



Method 2: RECORDED points can also be found by manually driving the robot to a position from the *Arm Control Panel* and selecting Save Point when the **EoAT** is at the desired location



TEACHING points is the process of manually typing in the XYZ coordinates if they are known. We often use this method to **TOUCH UP** points (round them to whole numbers, change them to a known number, or duplicate positions from previous steps that should not have changed).

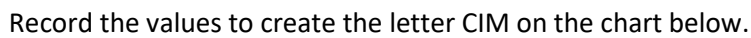


Always start a program with a safe/clear HOME position. Home is a position that is up and away or clear of any obstacles.

17. Using the XYZ buttons on the *Arm Control Panel*, move the robot to a safe **HOME** position. We can refer to this as "HOME" (Above Point A).
-Record these values on the chart located on the next page.

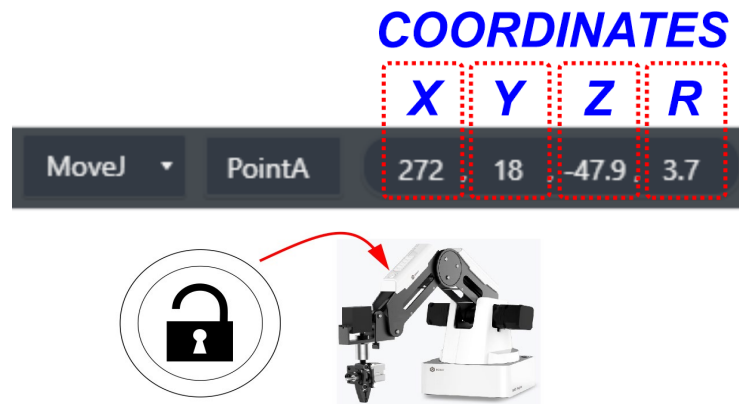
Using the XYZ buttons on the *Arm Control Panel*, move the robot to a point above **POINT A**. We can refer to this as AB-A (Above Point A). Do not touch the paper yet with the pen! - Record these values on the chart located on the next page.



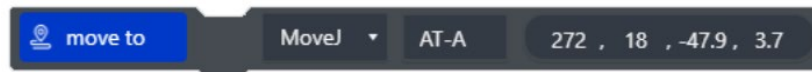


18. You can also press the “Lock” button on the arm and it will **RECORD** the point where the pen is when the button is released. If the point has moved a bit, you can touch up the points by **TEACHING** them in the XYZ coordinates window. Ignore the last number in the coordinate’s window, as this stands for “rotation”. It is not used with the pen.





Click in the cell with the "P" for each point to create a descriptive name for each position. See Example Below "AT-A" (EoAT is At Point A).



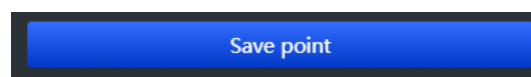
Movements can either be *MoveJ*, *MoveL*, *Jump*, or *Arc*. We will get to *Jump* and *Arc* in later activities.

MoveJ movements always have an arc to them. We use them when traveling from home to an above position or when traveling from an above to another above position.

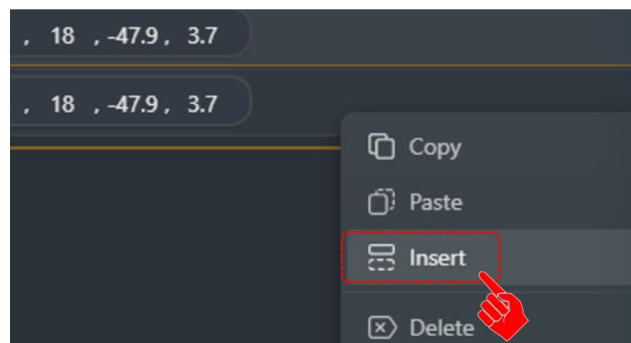
MoveL movements are linear. We use them when moving down to a point or object, or when moving away from a point or object to ensure movement accuracy. **All of the lines to create the letters for CIM should use *MoveL* movements**

If TEACHING the positions, select the Save Point tool to create 19 steps. Using the chart you created, fill in the names of the steps and the XYZ values.

Note: The coordinates automatically used for the Save Point is the current location of the **EoAT**



You may also right click on an existing step and select insert to create an additional step



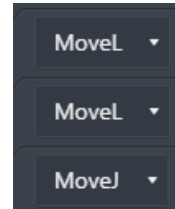
If RECORDING the positions, the positions are automatically created as well as filled in. These positions should be touched up.

Touch Up Example 1: AB (Above) positions should only have a different Z value. The X and Y should be the same.

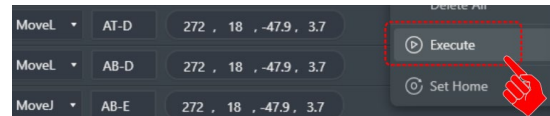
Touch Up Example 2: All AB (Above) positions should have the same Z value. All AT (At) positions should have the same Z value.

Touch Up Example 3: The positions for AT-A and AT-B should only change on one axis.

19. Change the Motion Styles to *MOVEJ* and *MOVL* where appropriate.



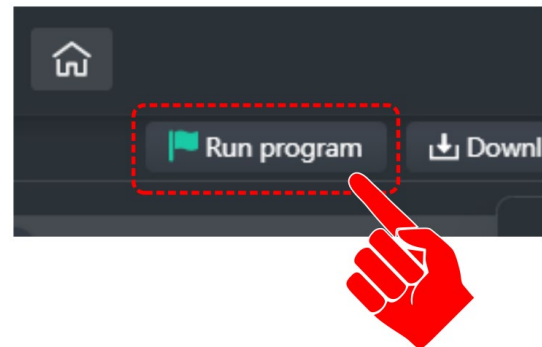
20. It is beneficial to step through (run one line of code at a time) a program to find any errors. Right click on each step one at a time (from the top down) and select EXECUTE. This will run just that step.



If there are errors in a step, update the XYZ coordinate values until they are correct.

Repeat this process for each step.

21. Once each individual step is verified, we can try to run the entire program. Select the first line of code, ensure there are no obstructions in the robots path, and select the RUN PROGRAM option at the top of the screen.



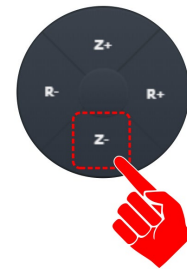
22. Now that we know all the points are correct, we must change the Z value so that it writes on the paper. To do this, follow these steps:
- A. Select the first AT line, right click & use *EXECUTE* to move the robot to point A.
 - B. Change the Z by increasing its value -2mm at a time and hitting *RunSelected* until the pen



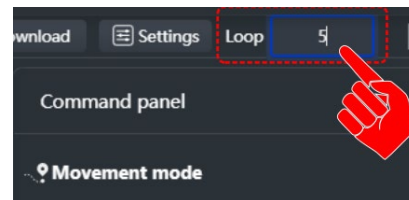
just touches the paper. You may also use the Z- from the Arm Control Panel

- C. Now change all the Z values where the pen must actually write a line. Leave the AB (Above) value alone when it moves from letter to letter.

COORDINATES			
X	Y	Z	R
272	18	-47.9	3.7



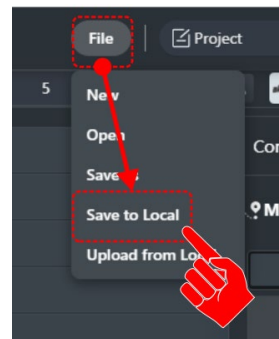
23. Using the *Loop* button, change it to 5 and watch your program **LOOP** five times when you hit the start button.



Save your work.

Select **File** from the top tool bar.

Select **Save to Local** to save your project on to the computer.



Let's check the **ACCURACY & REPEATABILITY** of our robot. Get another piece of graph paper and replace the old one. Be sure to tape it to your work surface. Now run your program again. Check all the lines closely. *How accurate was your robot at reproducing the word CIM five times? Describe it below.*

How accurate and repeatable was your robot and program? Be sure to describe both terms. Response:

What happens to the **ACCURACY & REPEATABILITY** if you increase speed to 75? Try it and see.

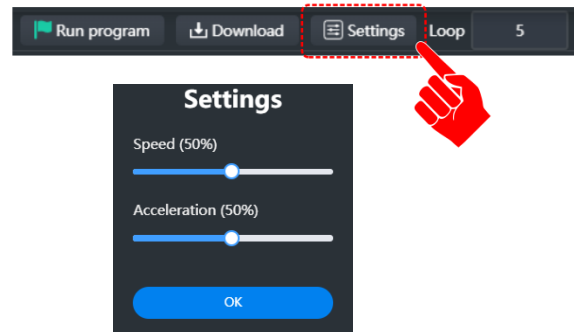


Please do not increase speed above what is specified by your instructor.





To change the speed of your program, select the settings tab at the top of the screen.



Speed Adjustment: What happened when you changed the speed? Be specific.
Response:

CONCLUSION

1. What does it mean to touch up a program?
2. Explain completely the difference between MOVJ and MOVL Motion Styles.
3. Does speed influence **ACCURACY** or **REPEATIBILITY** of your robot? Describe how.
4. What would be the effect on the robot's accuracy at higher speeds if you greatly increased the mass of the pen?
5. After completing this activity, how would you define the difference between a robot's accuracy versus its repeatability?



6. Why do we use AB (Above) positions before going to AT (At) Positions? Explain fully.

7. What is the purpose of a HOME position?

GOING BEYOND

Finished early? Try drawing one of the options below. When finished, show your instructor and have them initial on the line.

Teacher
Signature

- _____ 1. Teach the robot to make a barcode with the pen.
- _____ 2. Teach the robot to write your name.
- _____ 3. Teach the robot to write arcs (can you figure Arc movements out?).



1.2 TP - Pick & Place Routines

NAME: _____

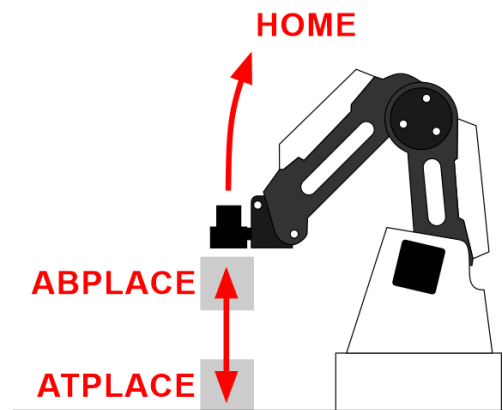
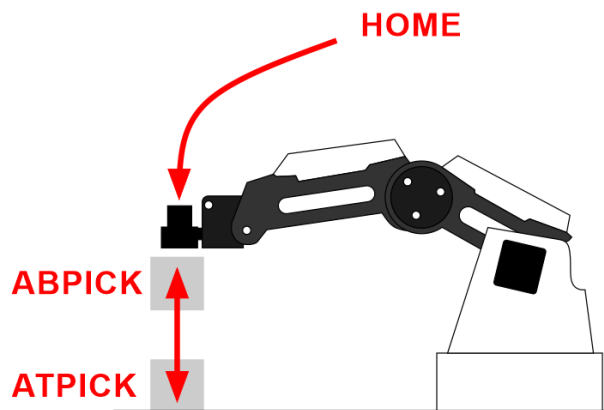
Date: _____

Section: _____

INTRODUCTION

Robotic arms are excellent for performing **PICK AND PLACE** operations such as placing small electronic components on circuit boards, as well as large boxes on pallets. A pick and place operation will require at least 5 points:

- Home
- Above the pick point
- At the pick point
- Above the pick point
- Above the place point
- At the drop point
- Above the place point
- Home



As a rule, always go to a position above the pick or place point first so that the robot can accurately and repeatedly place the object straight down in a linear motion, with no friction or interference with the work surface or other objects in the work envelope.

KEY CONCEPTS

- How does a robot perform a **pick and place** operation?
- What **END EFFECTOR** or **END OF ARM TOOLING (EoAT)** works best?
- What are Pick and Place **conventions** in industry?
- How do I attach the **Mechanical gripper** to the Dobot?
- How do I **record** positions with the Dobot?
- How do I easily edit a program in DobotLab and Playback Module?



KEY VOCABULARY

- Relative positions
- Palletizing
- Pick and Place
- Home
- Linear movements
- Joint movements
- End effector
- End of Arm Tooling (EoAT)

EQUIPMENT & SUPPLIES

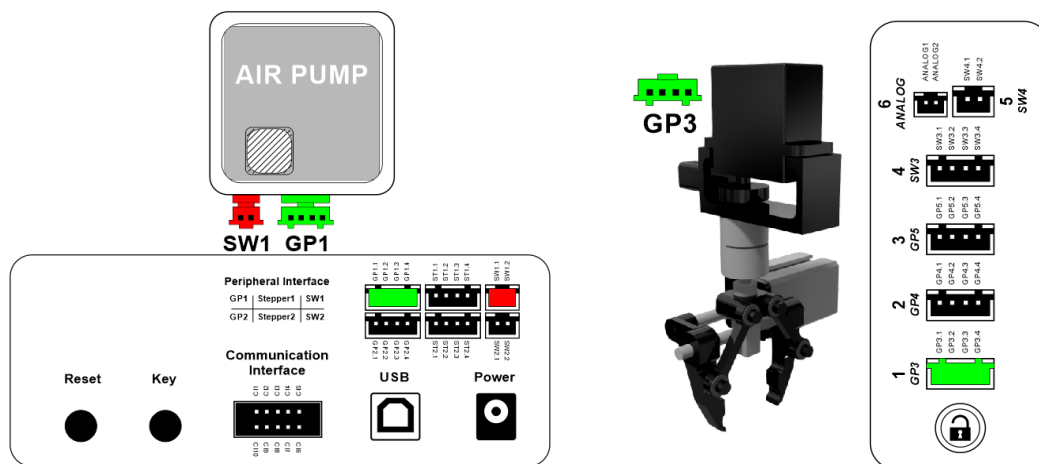
- Dobot Magician
- Dobot Field Diagram
- 1 inch cylinders or 1 inch cubes
- Pneumatic Gripper
- DobotLab software
- Accompanying video: Pick & Place with the Dobot

PROCEDURE

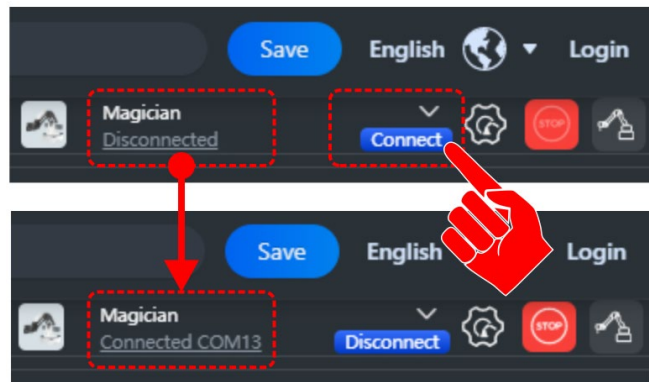


Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur.

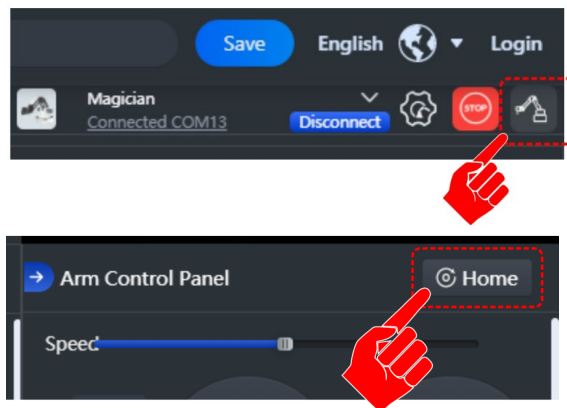
1. Attach the *Gripper* to the *rotational servo* and then attach it to the robot and plug in the wires as shown below.
2. See instructor for demo if the gripper is not already installed on the servo.



3. Open DobotLab software, enter the Teach and Playback Lab, and connect the robot.



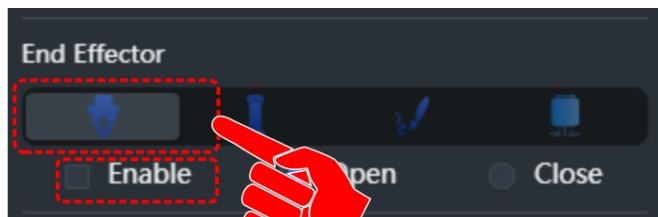
4. Open the Arm Control window and Home the robot making sure the robot's work envelope is clear.



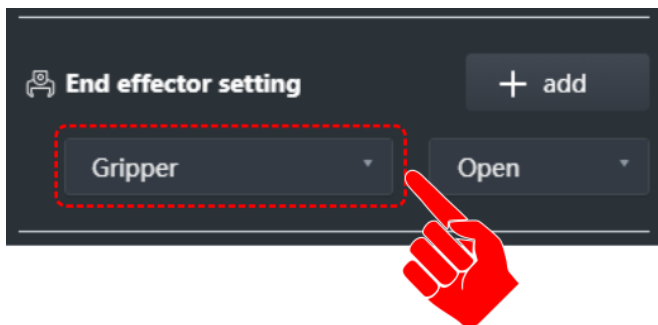
5. Be sure the Gripper is chosen as the **END EFFECTOR or EoAT**



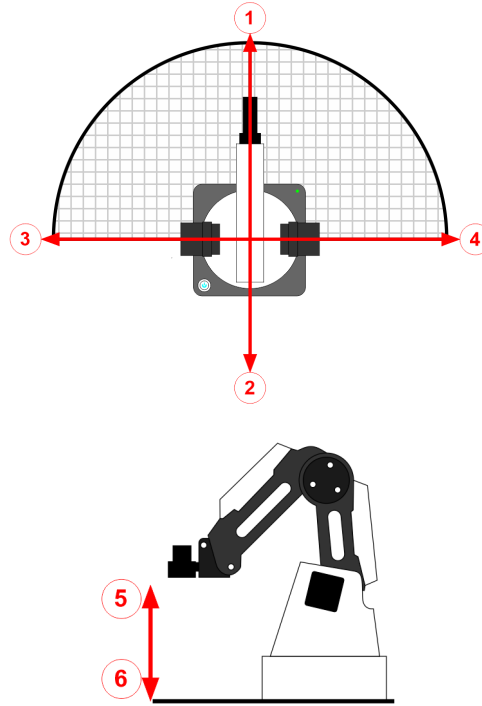
The **END EFFECTOR** only needs to be ENABLED when the gripper need to be active. We will activate it later in the activity.



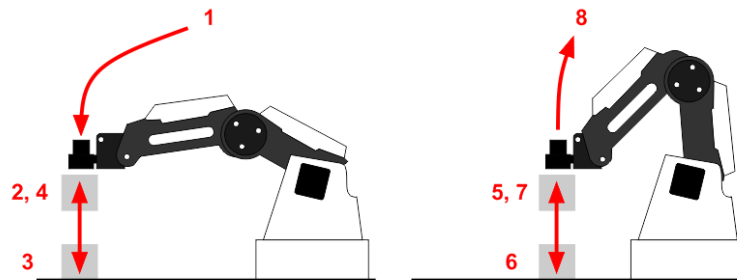
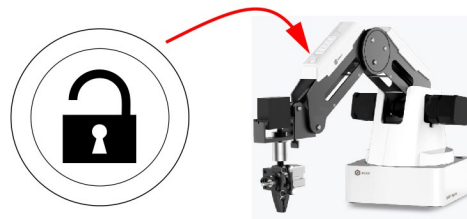
6. The Gripper ALSO needs to be selected as the **END EFFECTOR** setting in the Control Panel in order to record when it is open and closed.



7. In the *Arm Control Panel*, and move the robot around. Using the diagram below, identify the X and Y axis on the robot. Be sure to identify which direction is positive and which is negative +/-.



8. Place one of your cylinders or cubes on one of the squares of the field diagram.
9. Use the *“Lock”* button on the arm to record all the positions necessary to do a pick and place operation in this order:
- 1 - Home
 - 2 - Above Pick – (ABPICK)
 - 3 - At Pick – (ATPICK)
 - 4 - Above Pick – (ABPICK)
 - 5 - Above Place – (ABPLACE)
 - 6 - At Place – (ATPLACE)
 - 7 - Above Place – (ABPLACE)
 - 8 - Home

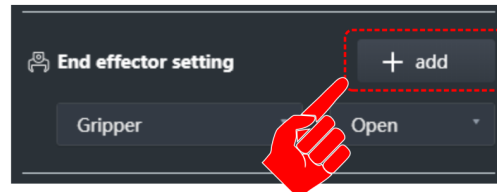


10. Name the positions in the *name column* of the program.



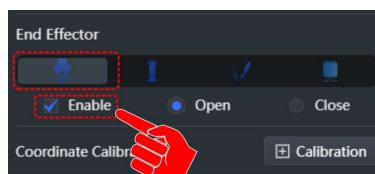
Be sure to name the positions something relevant so that others will be able to tell what the positions are. Example: A point named ABPICK means the point above the place where the object is picked up.

11. Next, add the commands to open and close the gripper. Select the add button in the **END EFFECTOR** Settings window. This will add the step to the end of your program. Click and drag the new step into the correct position of your program sequence. Repeat this step to open the gripper when the object needs to be released.

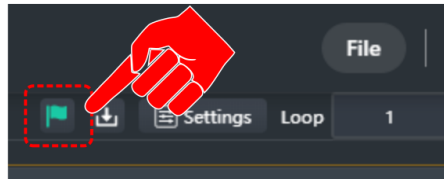


Right click on each step (one at a time), and select Execute. This will allow you to step through the program to find any errors before trying to run the entire program. Make any edits to the sequence as necessary as you go.

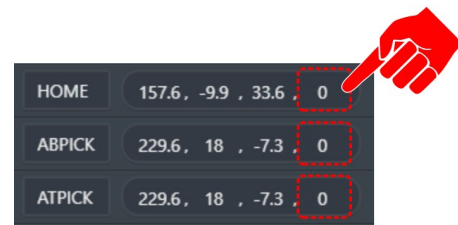
NOTE: The gripper will **AUTOMATICALLY** become active (pump will turn on) when an open/close step is executed. It can also be manually activated and deactivated by selecting or unselecting the Enable options for the **END EFFECTOR** in the Arm Control Window. The pump will switch modes while opening and closing the gripper (*the sound coming from the pump will alternate when the mode is switched*). One mode produces pressure to open the gripper while the other mode produces a vacuum to pull the gripper closed.



12. Select the “Run” button to run your entire sequence and see what happens. *Did it work the first time? If not, what did you have to change to make it work?*



13. If you are using cubes, we will want the gripper to remain parallel to the robot. To do this, change all of the Roll Angles to ZERO. This will keep the gripper in the same orientation throughout the program. The gripper’s roll angle DOES NOT matter if the objects are cylinders.

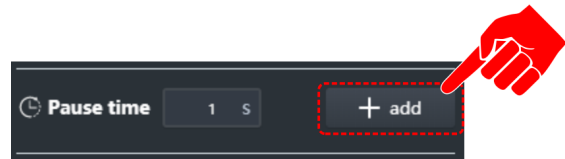


HOME	157.6 , -9.9 , 33.6	0
ABPICK	229.6 , 18 , -7.3	0
ATPICK	229.6 , 18 , -7.3	0

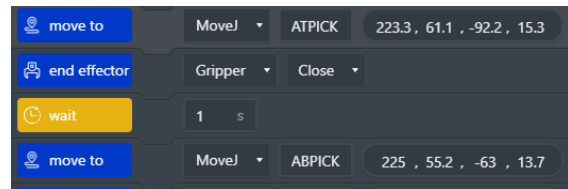
14. Change the position type of step #8 (At Pick) to *Move Linear (MOVL)*. *Run the program. What changed? What positions should be Linear Movements, and which should/could be Joint Movements? Edit the program to have appropriate movement types for each position in the sequence.*



15. Next we will add Pauses for time in-between step. This will allow the robots EoAT to completely close, or open, before it moves onto the next step in the sequence.

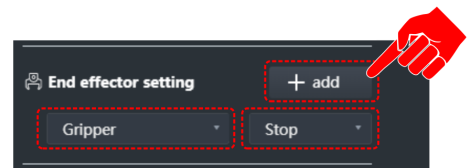


When the Add button is pressed, the pause step will be added to the end of the sequence. Grab the newly created step and drag it below the close gripper step. For this activity, 1 second should be enough.

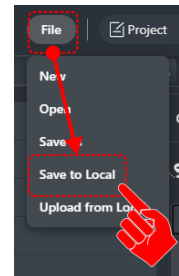


Add additional Pauses to the program sequence where appropriate. Pauses should be used when trying to ensure the gripper or suction cup has built up full pressure or suction before picking up or letting go of an object.

16. Finally, we will add an additional step at the end of the program to stop or turn off the pump.



17. Save your work.



CONCLUSION

1. How can you get the suction to turn on in time to pick up the part, or get it to shut off in time to drop it off correctly? Explain below after you have tried it in the program.
2. What happens if you replace the $\frac{3}{4}$ " cylinder with a $\frac{3}{4}$ " wooden cube? Run it and see. Describe below what happens.

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- | | |
|-------|---|
| _____ | 1. Have the robot reverse the process and put the cube back in its original place. |
| _____ | 2. Make the process happen multiple times. |
| _____ | 3. Adjust the speeds and acceleration to increase efficiency but not lose accuracy. |



1.3 Block- Pick and Place

NAME: _____

Date: _____

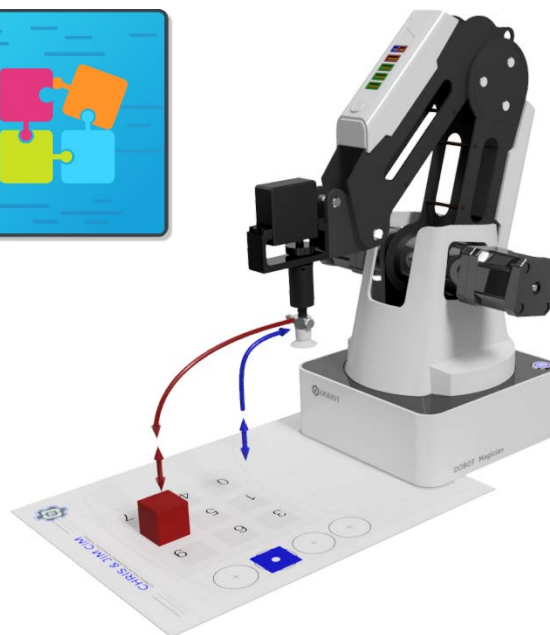
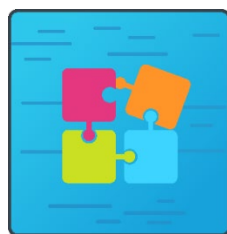
Section: _____

INTRODUCTION

Robotic arms are excellent for performing pick and place operations such as placing small electronic components on circuit boards, as well as large boxes on pallets. A pick and place operation will require at least 5 points:

1. A home or safe location
2. A position above the object
3. A position at the object
4. A position above the drop off
5. A position at the drop off

In this activity you will learn how to make a basic **Pick and Place** operation in blockly. Through this activity you will learn how to program the robot to move and turn on its suction cup in blockly



WARNING



Power OFF

Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS shutdown the Dobot before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

KEY VOCABULARY

- MoveTo
- Placeholder
- Delay time
- Suction Cup
- Blockly Programming
- Pick and Place



All blockly commands and robotics terms have been put into separate *Appendices* and can be referred to at any time throughout these activities.

EQUIPMENT & SUPPLIES

- Robot Magician
- Dobot Blockly - Pick and Place Field Diagram
- 1" cubes or cylinders
- DobotLab Software or DobotStudio
- Suction Cup Gripper

KEY CONCEPTS

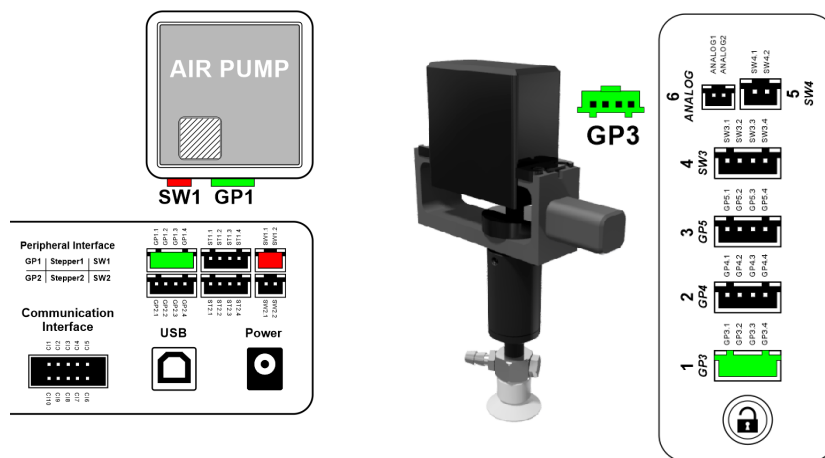
- Differentiate ways that robot arms can move using blockly: **Straight Line** and **Joint**.
- Differentiate between **absolute** and **relative** coordinates.
- What **Delays** are used for in robotics and how to create them.
- . What are some of the basic Dobot configurations?
- How **suction** is used to move objects and how to turn it on and off using blockly.
- Starting up and connecting the Dobot Magician.
- How to utilize a robot arm to move through a group of points by using an **end effector**.

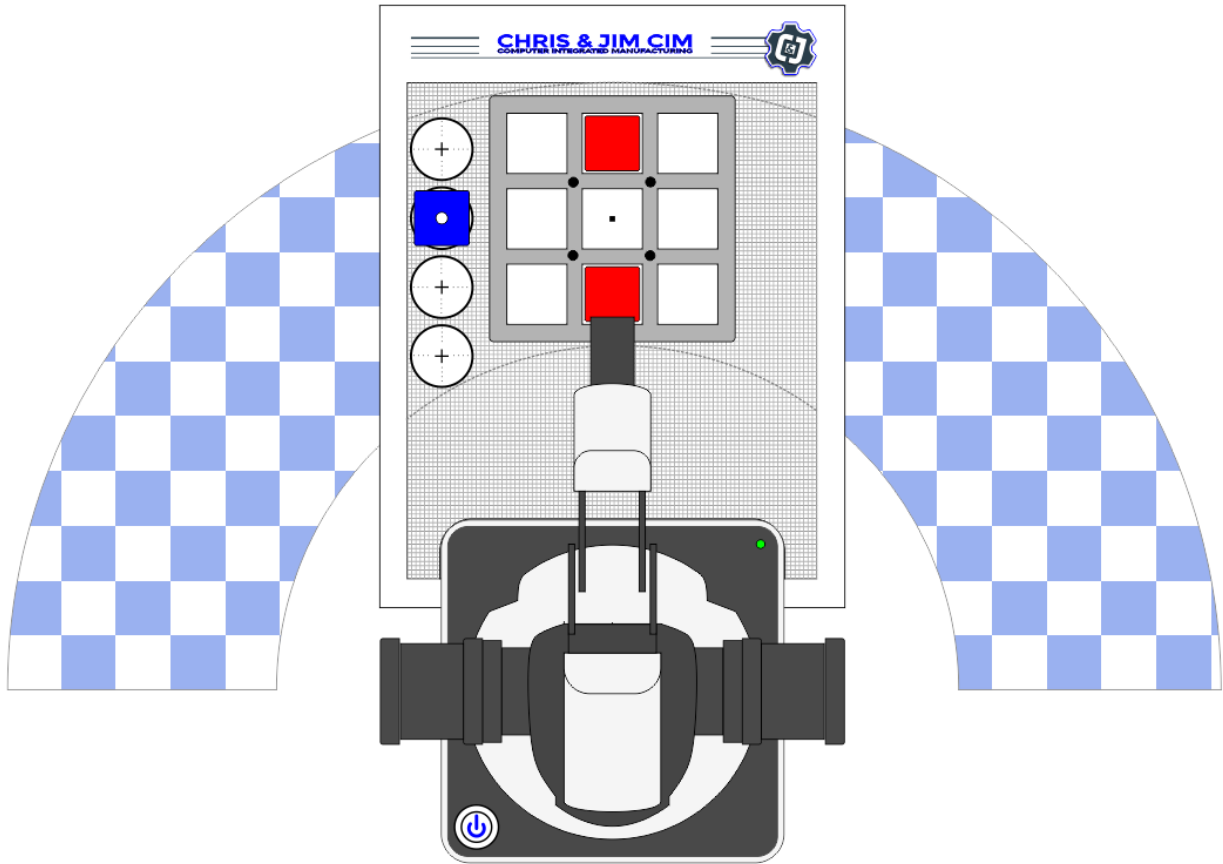
PROCEDURE



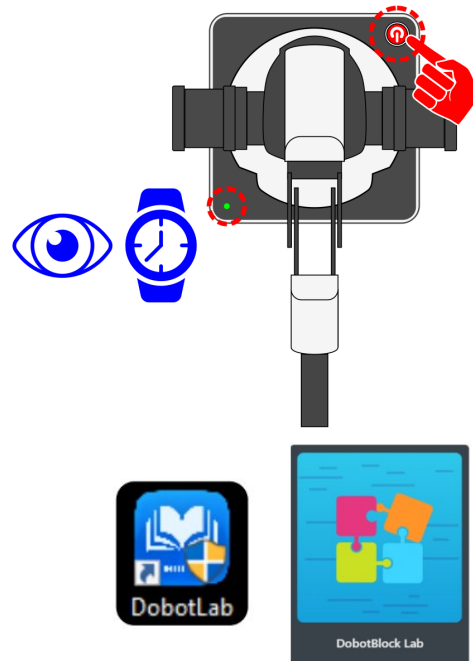
Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Print the Blockly - Pick and Place Field Diagram
2. Set up the robot with a **SUCTION CUP** and Air pump and place a cube in one of the red squares on the field diagram provided.





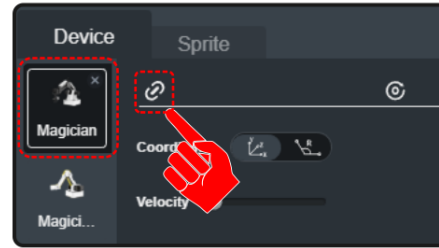
3. Once **ALL** of the wiring is done (sensors connected and robot arm connected), power ON the Magic Box and Robot. **WATCH** and **WAIT** for the robot to completely power on (GREEN INDICATOR)



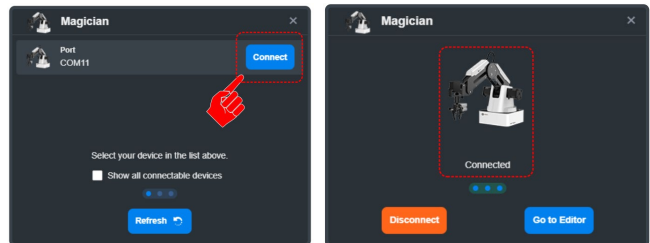
4. Open up DobotBlock Lab in the software and connect the robot.



5. Select the connect icon to establish control from the software to the robot. A connection window should pop up.

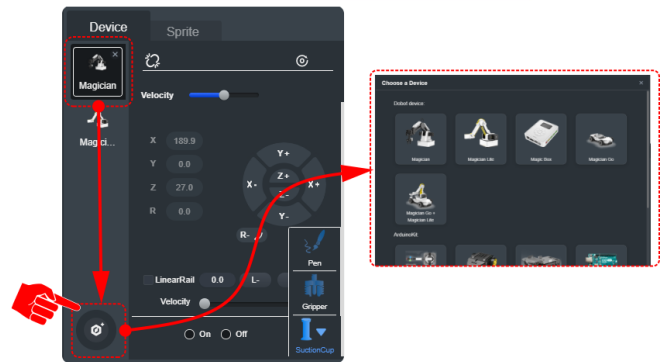
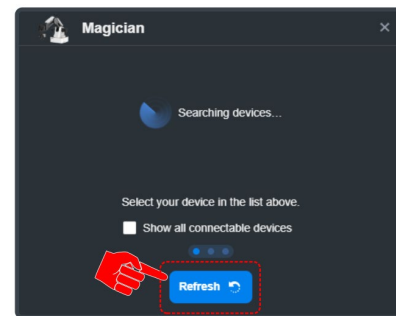


6. Select Connect from the pop up window. A new window should appear and show that the robot is Connected.



NOTE: if the Magician is not on the Device list, scroll down to select the “Choose a Device” option

NOTE: If no robot is found, the prompt will show as “Searching Devices”. Reconnect the robot and select Refresh

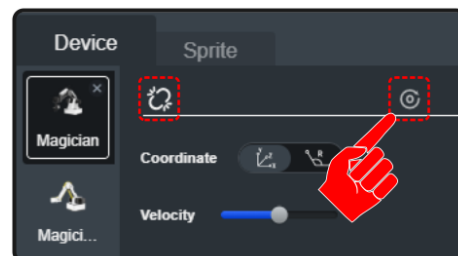


7. Once the robot is connected we need to home the robot to set its home position.



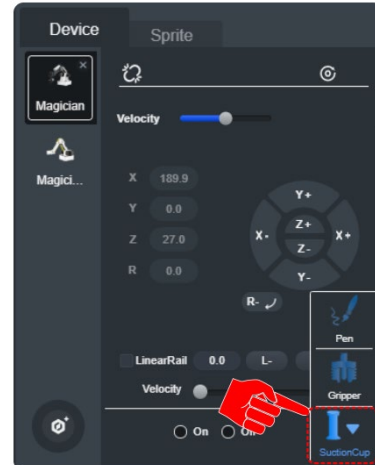
Before **HOMING** the robot, make sure the robot’s **WORK ENVELOPE**, the area in which the robot can reach, is clear.

Select the **HOME** icon to start the robots homing process. **HOMING** the robot will return and set the robot to its initial **HOME** position.



- Next, choose the SuctionCup as the Dobot's **END EFFECTOR** or **EoAT** (End of Arm Tooling).

NOTE: You may need to scroll down to see the **EoAT** selections



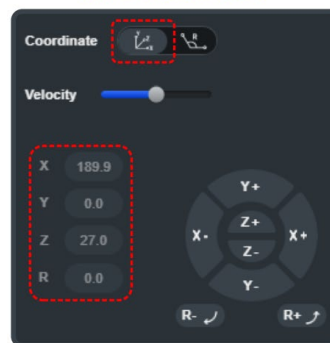
- As with industrial robots, DobotBlock Lab provides the user with two methods for driving the robot manually. These options are XYZ MOTION and JOINT MOTION.

JOINT MOTION - This method of movement allows you to configure one joint at a time by **DEGREES**. All of the other joints stay in the same relationship to the joint being driven. Practice with this type of movement.

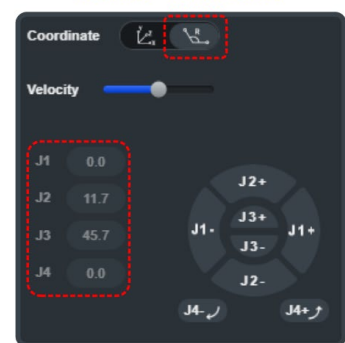
XYZ MOTION – XYZ Motion will move on multiple axis at one time by **ABSOLUTE COORDINATES**. The robots motion is based on the keeping the EoAT in line with the axis chosen. Practice with this type of movement.

The position data can be collected from either method depending on the coordinate motion style used when developing a program.

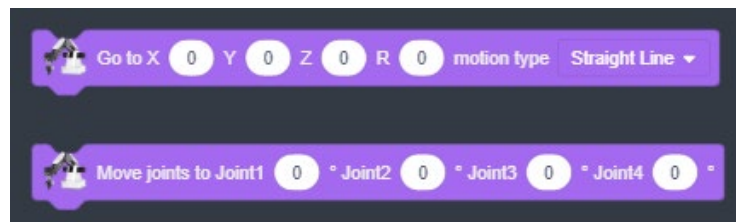
XYZ MOTION



JOINT MOTION



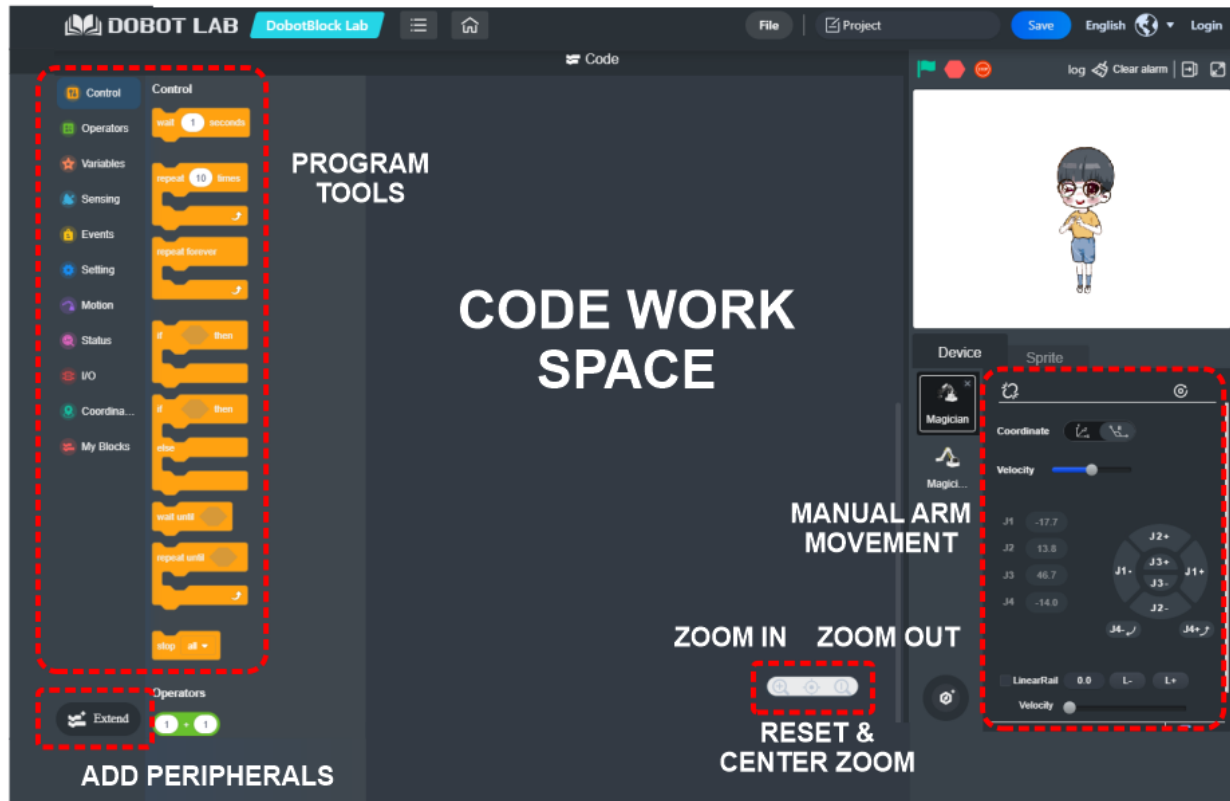
LINEAR MOTION	JOINT MOTION
X – In and Out	J1- Waist
Y – Side to Side	J2 - Shoulder
Z – Up and Down	J3 - Elbow
	J4 – Wrist (<i>Only available when the gripper or suction cup servo is attached</i>)



Shown below is the main menu for the Blockly programming section of DobotBlock Lab. On the left are categories and tools that each block of code is sorted into to. On the right are controls for zooming around the program as you develop it. The scroll wheel on the mouse can also be used to scroll up and down. The scroll wheel or the left mouse button can also be pressed and held to pan throughout the program workspace.



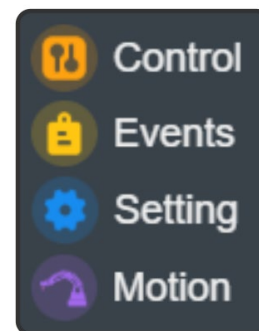
The Crosshair Icon can be used to reset the zoom level back to default and center your code in the middle of the field.



For this activity, we will be using tools from multiple categories. The categories we will be using are:

- Control
- Events
- Settings
- Motion

And can be found in the left menu as shown above.

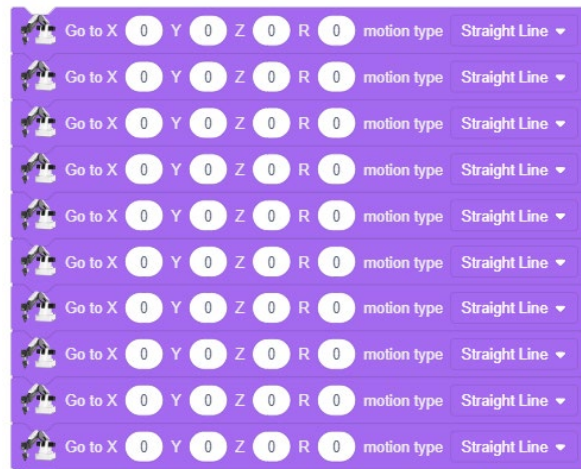
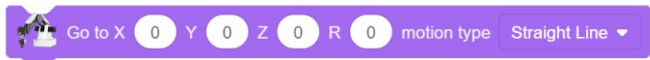


10. Each program will typically start with some setup code (EoAT, Motion Velocity, Inputs, Outputs....)

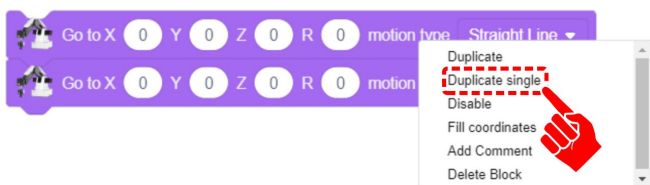
For this program we will start by assigning the EoAT. Drag over the Set end effector tool from the **SETTINGS** category. Change the EoAT to the Suction Cup



11. Next, drag over multiple “Go to XYZR” from the **MOTION** category (8 steps, one for each movement of a pick and place routine).



NOTE: you can also right click on an existing step in the program workspace and select DUPLICATE or DUPLICATE SINGLE.

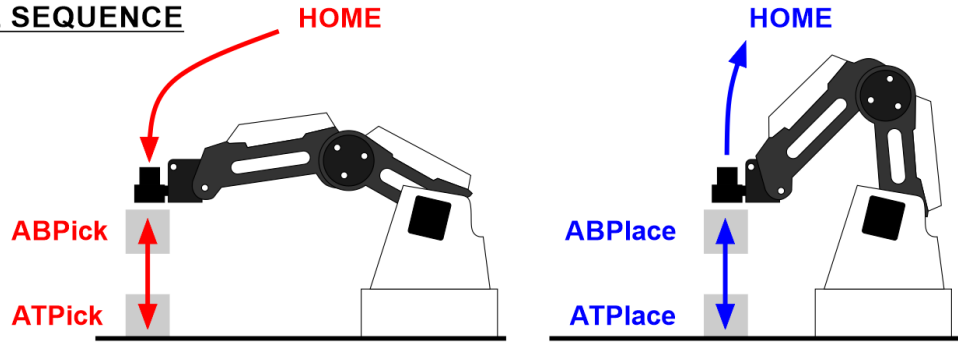


DUPLICATE will create a copy of the current step and all the steps that are below and attached to the current step.

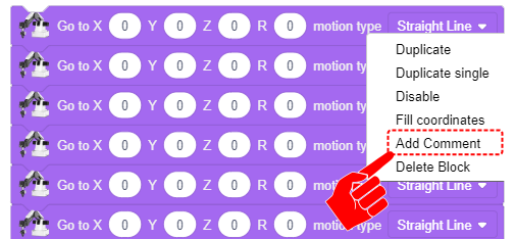


PICK & PLACE SEQUENCE

1. HOME
2. ABPick
3. ATPick
4. Vacuum On
5. ABPick
6. ABPlace
7. ATPPlace
8. Vacuum Off
9. ABPlace
10. HOME

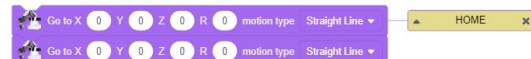


12. We will now add names/comments to each step. Right click on a step and select **ADD COMMENT**.

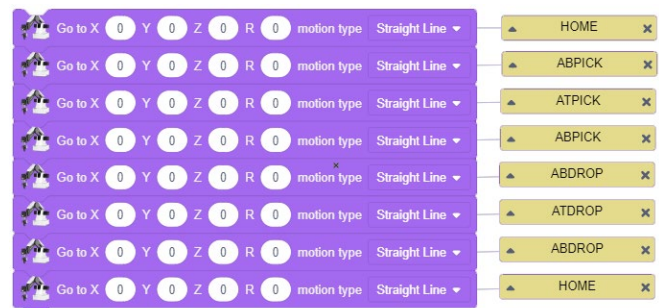


Type in the name of the step using the diagram above as a guide.

When the comment is complete, click the down triangle to collapse the comment



Repeat this process for all 8 movement steps



13. We now need to find the XYZR coordinates for each of the positions

Press and hold the lock button on the arm of the robot. This will release control of the arm to the operator and allow you to move the arm to a new location. Once at the new location, release the lock button to have the arm regain control and hold that position.

We will find the HOME position first. Using the lock button, move the arm to a new HOME position and release the button.

Ensure the COORDINATES shown are the XYZR

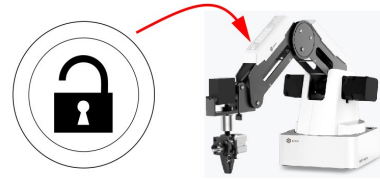
Right click on the 1st step (HOME) and select Fill Coordinates. This will use the robots current position to fill in the values for XYZR

It may be necessary to touch up the steps. For example, the X value of 189.9 can be set to 190.

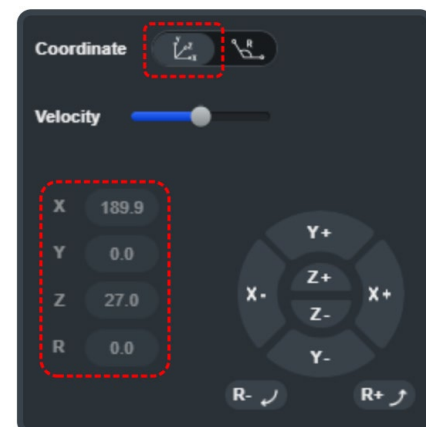
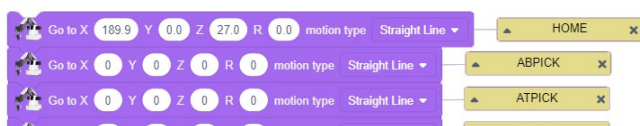
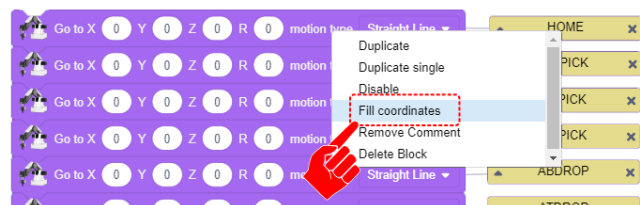
Repeat this process for the remaining steps.



Once the lock button is released, use the manual controls to fine tune the position or type in the desired values (round the numbers).



XYZ MOTION



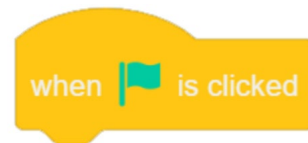


When dragging around blocks of code, if you select the top block, it will allow you to drag around that block and all of the blocks connected below it. If you selected a block of code from the middle of a string of blocks, it will disconnect that block and all of the blocks below it.

Complete the table below with all of the XYZ coordinates needed.

	X	Y	Z	R
1. Home				
2. Above Pick				
3. At Pick				
5. Above Pick				
6. Above Place				
7. At Place				
9. Above Place				
10. Home				

14. At this point we should test the positions we gathered. We will now perform a dry run (only perform movements) to see if our positions will work. To do this, the program needs a line of code that instructs it to run when the go button is selected.

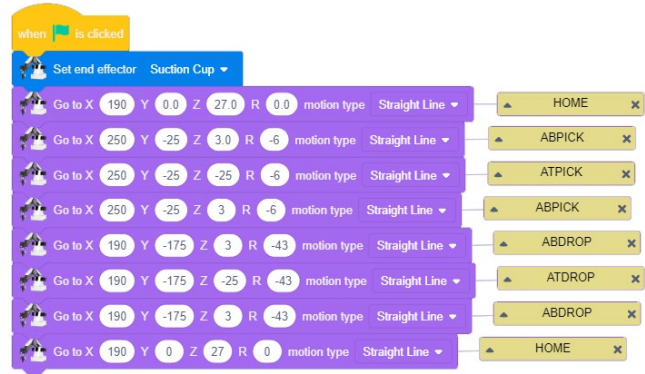


15. Select the Go button (the GREEN flag) to run the program.

From the EVENTS category, drag over the WHEN GO IS CLICKED tool and attach it to the top of your program.

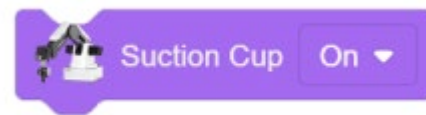


Make any adjustments to program as needed.

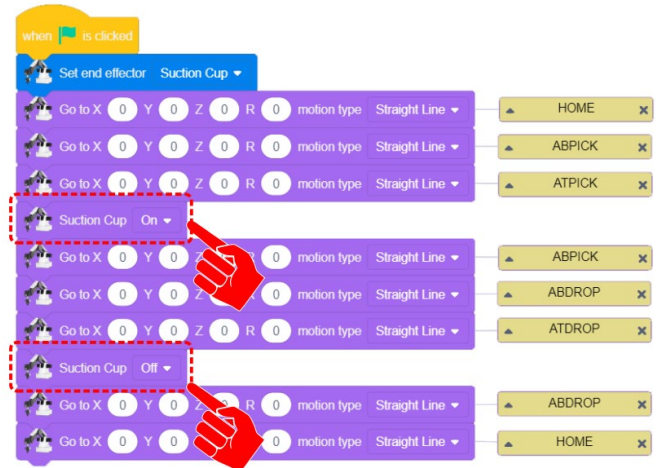


16. We will now add the steps needed to turn on and off the suction cup.

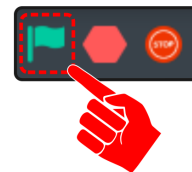
From the MOTION category, drag over the GRIPPER tool.



We will insert this step into the program. As you drag the step over to your program, the program will automatically separate where you drag the tool. We need the GRIPPER to close after we are at the position ATPICK and open after position ATDROP

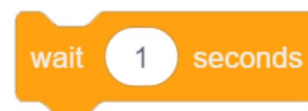


Test the program and see if it works as expected.

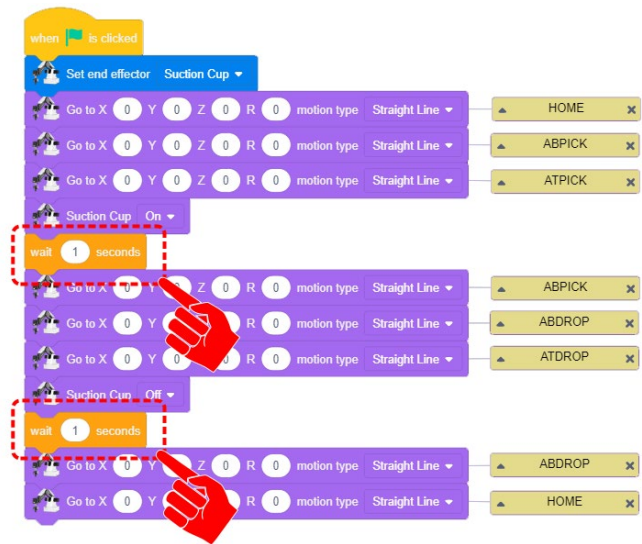


17. Finally, we will add a short pause after we turn on the suction cup and after we turn off the suction cup. This will ensure we have full pressure and have completely released the part before moving to the next step.

From the **CONTROL** category, drag over the wait/seconds tool



Place a 1 second **WAIT** after each EoAT operation



Test the program and see if it works as expected.



When dragging around blocks of code, if you select the top block, it will allow you to drag around that block and all of the blocks connected below it. If you selected a block of code from the middle of a string of blocks, it will disconnect that block and all of the blocks below it.

If your set up did not work correctly the first time, what did you have to do to make it work?



CONCLUSION

1. *What are the five needed positions for a pick and place operation?*
2. *Explain in your own words why it was necessary to add delay times into the program in the space below.*
3. *What is the purpose of the safe positions that are programmed above the object before it is picked up?*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- | | |
|-------|---|
| _____ | 1. Reverse the process so that at the end the robot puts the cube/cylinder back in its original position |
| _____ | 2. Try picking and placing the object on locations that are not directly in front of the robot. What does this change? Can this be corrected in Blockly as it was in Teach and Playback? How? |
| _____ | 3. For the sake of accuracy, change the speed of the arm as it moves from the ABPICK to the ATPICK, and ABDROP and ATDROP to slow it down by half (The codes needed to control speed are in the SETTINGS toolbox). |



1.4 TP - PnP with Jumps and Loops

NAME: _____

Date: _____ Section: _____

INTRODUCTION

A **PICK AND PLACE** style of moving objects around are staples of industrial robots. Another reason to use robots in industry and automation is because of danger to humans. Robots can work in adverse environments that are dangerous to humans; especially when dealing with chemicals and other toxic substances.

Sometimes the best way to pick up an object is with a suction cup or **VACUUM GRIPPER**. This works especially well with very small objects and provides a less expensive alternative to a **MECHANICAL GRIPPER**.

In this activity, you will perform an anodizing operation (which can use harmful chemicals) with a robot arm using the air pump kit and a vacuum gripper.

Do not let go of the object in the tanks... Just wait for the 2 second dip to complete and move on.



Dip Tank Set Up using 3D printed dip tanks and a Dobot field diagram. In this diagram four tanks are used.



Remember: We always go to a position above the pick or place point first so that the robot places the object straight down, with no friction or interference. This will increase accuracy!

KEY VOCABULARY

- TCP
- Loop
- Touch up
- Teach
- Suction
- Vacuum
- Linear movement
- Joint movement

EQUIPMENT & SUPPLIES

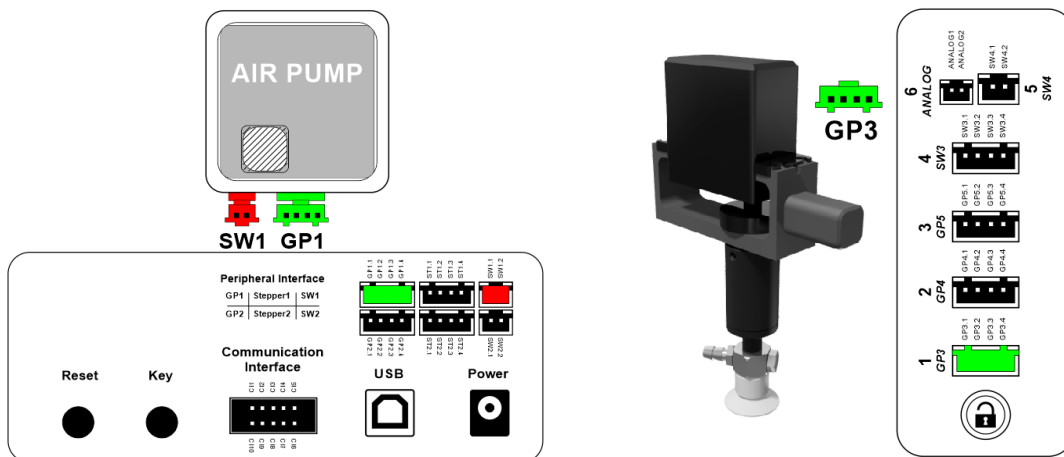
- Dobot Magician
- Dobot Dipping Field Diagram
- ¾ inch wooden cylinders
- End of Arm Tooling (EoAT)
- Suction Cup Gripper
- DobotLab software
- 4 simulated dip tanks
- (1" PVC Caps or small plastic cups work well)

PROCEDURE



Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Attach the *Suction Cup* to the *rotational servo* and then attach it to the robot, and plug in the wires as shown below.

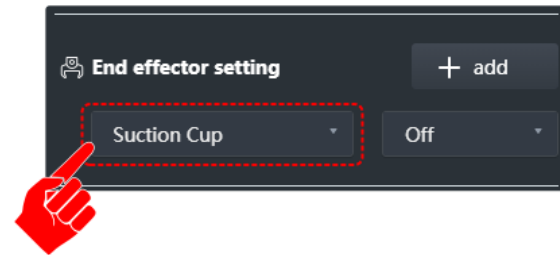


1. Open DobotLab software, open the Teach and Playback Lab, connect and home the robot, and be sure the *Suction Cup* is chosen as the **END OF ARM TOOLING (EoAT)** in both the **ARM CONTROL PANEL** and the **COMMAND CONTROL PANEL**

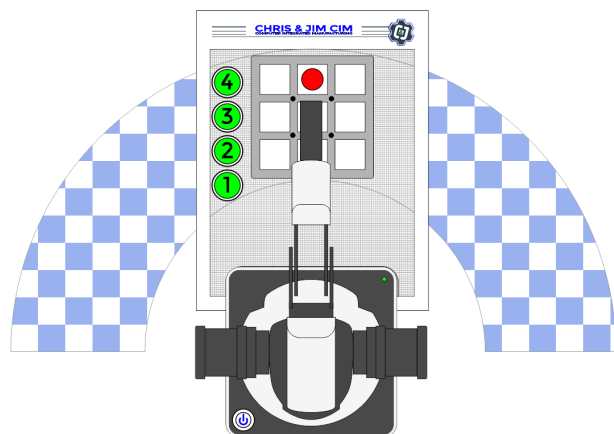
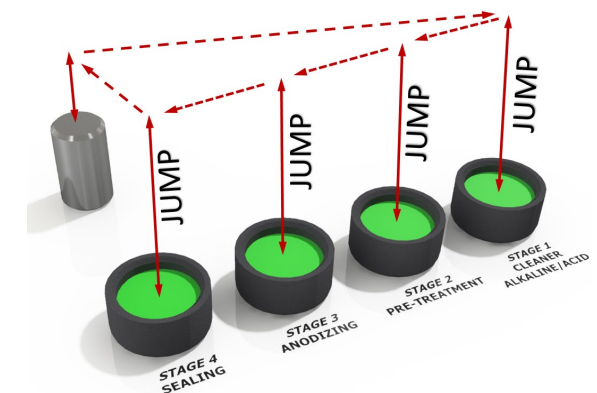
Arm Control Panel



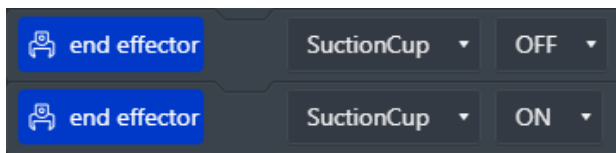
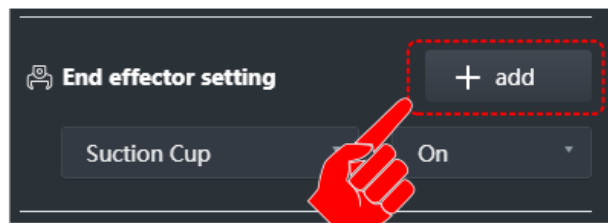
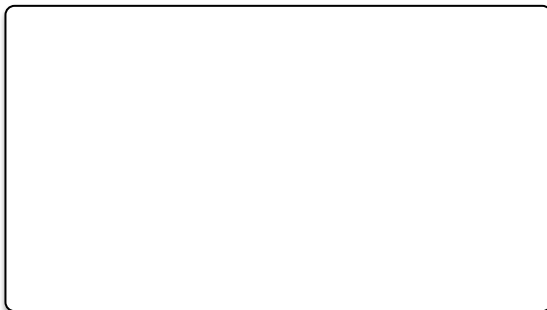
Command Control Panel



2. Once the robot is setup, create the rough sequence to perform the following actions:
 - a. Go home
 - b. Pick up the cylinder from the pallet
 - c. Dip it in Tank 1 for 2 seconds
 - d. Move the object and dip it in Tank 2 for 2 seconds
 - e. Move the object and dip it in Tank 3 for 2 seconds
 - f. Move the object and dip it in Tank 4 for 2 seconds
 - g. Move the part back to the pallet.
 - h. Go home

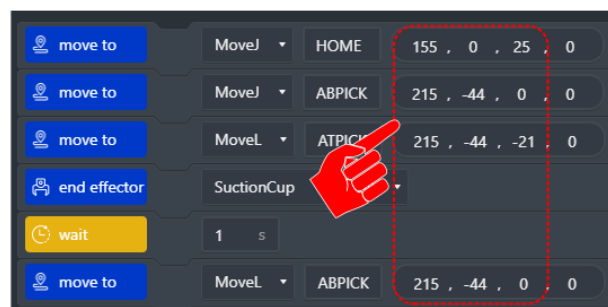


3. Be sure to turn on/off the *Suction Cup* when necessary.
4. Hit the *Play* button to run your program and see what happens. *Did it work the first time? If not, what did you have to change to make it work?*



5. When recording points using the *lock button*, it is not always very accurate. When you look at all the Z values of all the above tank positions, they are probably very different.

We can make them all the same by clicking on the values and typing in a value that works well; this is called **TEACHING**. When fixing points like this it is also called **TOUCHING UP** points.

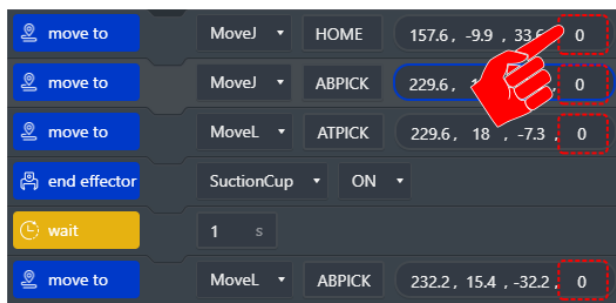


Go and **TOUCH UP** all the z values to make the dipping operation happen at the same heights above the tank, and in the tank and have your instructor check it.



You can even use this method to touch up the x and y values of the tanks to make sure they are in a straight line.

If you are using cubes, change all of the Roll Angles to zero as we did in Activity 2.



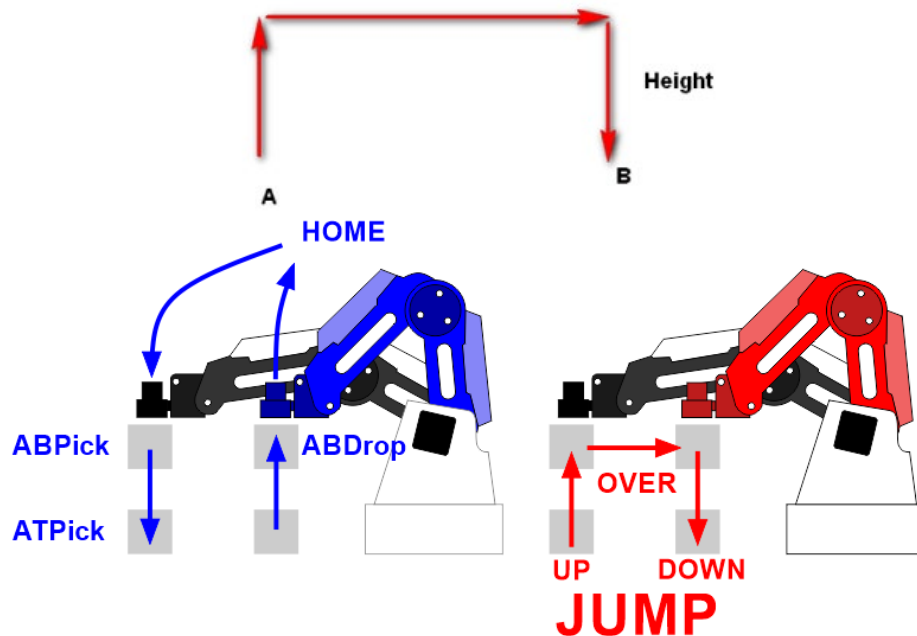
6. How many lines of code did it take to write the program using what you have learned before? How many points did you have to teach it?

7. Now we are going to use the **JUMP** command to make the program even easier, and much more accurate. Open a new program.



What is a **JUMP**? A JUMP movement combines three steps into one. It combines the raise up, over, and back down to the same Z level at new location. This type of movement simplifies repetitive movements such as a dipping operation or soldering operation. The Z Height is defined in the JUMP parameters in the settings menu.

A JUMP movement does not replace the initial ABOVE position, AT the initial position or ABOVE the final position. It is only used to RASIE UP / MOVE TO A DIFFERENT POSITION / RETURN TO Z FROM LAST POSITION.



8. This time, only record the following points:
 Home,
 Above Pick and At Pick,
 At Tank 1,
 At Tank 2,
 At Tank 3,
 Above Tank 4 and At Tank 4
 Home

Do not record any above points except for Above Pick and Above Tank 4. Jumps will replace these points

9. Now change the **MotionStyle** of the Tank 1, Tank 2, Tank 3, and Tank 4 to *JUMP*. Double click on the **MotionStyle** box for each point and change it from *MOVJ* to *JUMP*

Now play the program.

What does this change do to the program?

move to	MoveJ	HOME	0	0	0	0
move to	MoveJ	ABPICK	0	0	0	0
move to	MoveJ	ATPICK	0	0	0	0
move to	MoveJ	AT-T1	0	0	0	0
move to	MoveJ	AT-T2	0	0	0	0
move to	MoveJ	AT-T3	0	0	0	0
move to	MoveJ	AT-T4	0	0	0	0
move to	MoveJ	AB-T4	0	0	0	0
move to	MoveJ	HOME	0	0	0	0

move to	MoveJ	HOME	0	0	0	0
move to	MoveJ	ABPICK	0	0	0	0
move to	MoveJ	ATPICK	0	0	0	0
move to	MoveJ	AT-T1	0	0	0	0
move to	MoveJ	AT-T2	0	0	0	0
move to	MoveJ	AT-T3	0	0	0	0
move to	Jump	AT-T4	0	0	0	0
move to	MoveJ	AB-T4	0	0	0	0
move to	MoveJ	HOME	0	0	0	0

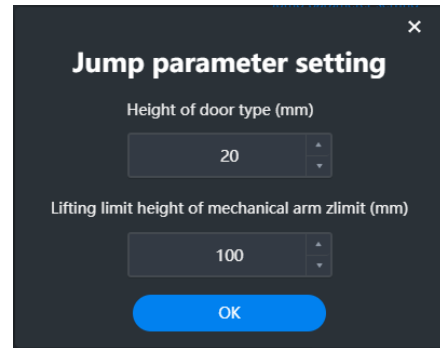
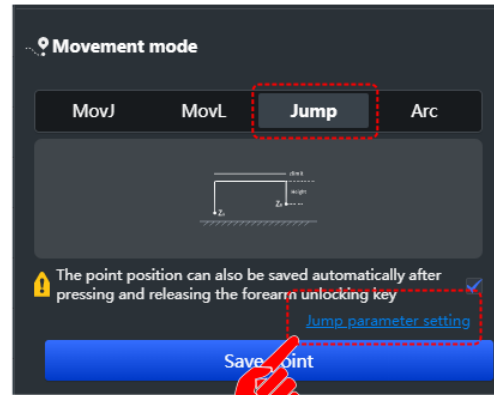
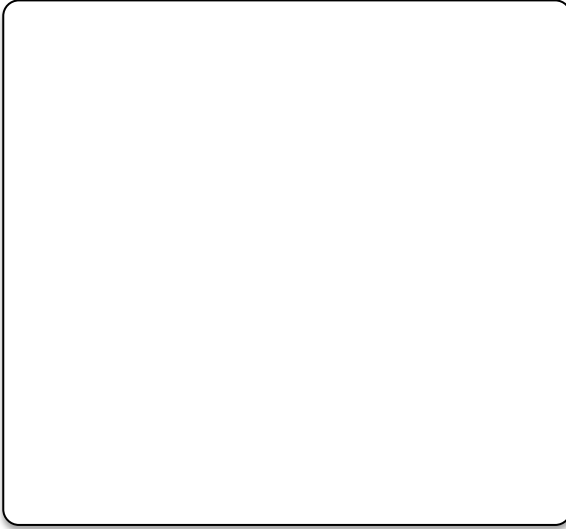


move to	MoveJ	HOME	0	0	0	0
move to	MoveJ	ABPICK	0	0	0	0
move to	MoveJ	ATPICK	0	0	0	0
move to	Jump	AT-T1	0	0	0	0
move to	Jump	AT-T2	0	0	0	0
move to	Jump	AT-T3	0	0	0	0
move to	Jump	AT-T4	0	0	0	0
move to	MoveJ	AB-T4	0	0	0	0
move to	MoveJ	HOME	0	0	0	0



10. If you ever want to change the height in Z that the *JUMP* actually jumps to, open the *Jump Parameter Settings*. You can edit the *Height of Door* to change the *Jump Height* to whatever you need it to be to clear the tank.

Compare the first dipping program to the second one with JUMPS and explain how it is more efficient.

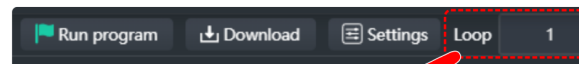


11. Now add the missing steps to finish the sequence.

- Move Linear Movements (MoveL)
- Pause for Time
- Suction Cup On and Off

12. Next, add the ability to make the robot repeat the operation five times. Do this with a process called **LOOP**. **LOOPS** will make something happen multiple times so that our program is shorter and more efficient. In the TEACH AND PLAYBACK LAB, the loop function will loop the entire program from start to finish. We do not have the ability to control how much of the program will loop or even if it should loop (a conditional loop).

13. To make your program **LOOP** multiple times, just add the number of times you want it to repeat in the loop window.



14. Be sure to save your work when done.



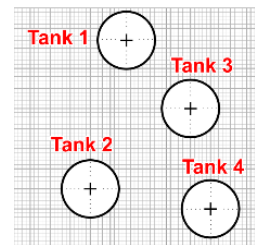
CONCLUSION

1. Describe a MOVL move using the Dobot.
2. Describe a MOVJ move when using a Dobot.
3. What makes a Jump command more efficient to code than a MOVJ or MOVL?
4. What kind of manufacturing process is dipping? Explain your answer.
5. Anodizing is a manufacturing process. Explain how it works and why we do it.

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- | | |
|-------|--|
| _____ | 1. Take away the first tank and change the program to fix the operation using the software. |
| _____ | 2. Add another tank and change the program to fix the operation using the software. |
| _____ | 3. Raise the tanks using a block of wood provided by your instructor and fix the operation using the software. <i>What is the most efficient way to do this?</i> |
| _____ | 4. Randomize the tank patterns for the dipping process |



1.5 Block - PnP with Jumps and Loops

NAME: _____

Date: _____

Section: _____

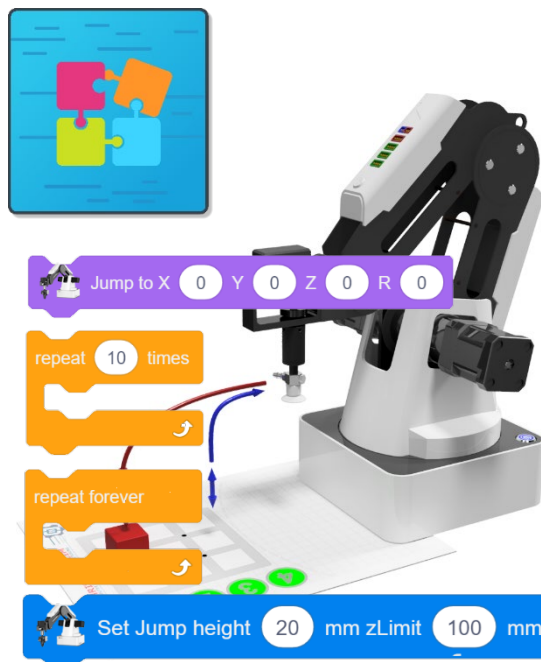
INTRODUCTION

When programming a robotic arm, it often becomes necessary to repeat operations a set number of times or indefinitely. This can be accomplished by adding different styles of loops to our program. It is also a good programming habit to optimize or reduce your lines of code when appropriate.

In this activity you will learn one way to simplify your code as well as add different styles of loops to a basic Pick and Place operation in DobotLab.

The two main types of loops are:

- Forever loops
- While loops
- Repeat



Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS shutdown the Dobot before making connections or damage to the robot could occur.

KEY VOCABULARY

- | | |
|----------------|---------------|
| • Forever Loop | • Jump |
| • While Loop | • Placeholder |
| • Repeat | • Condition |
| • True | |



All blockly commands have been put into a separate document called *Blockly Vocabulary*, and can be referred to at any time throughout all activities.

EQUIPMENT & SUPPLIES

- Robot Magician
- Dobot Field Diagram
- 1" cubes or cylinders.
- DobotLab software
- Suction Cup Gripper

KEY CONCEPTS

Essential questions answered in this activity include:

- What's a *Jump*, and why is it important?
- How to make the robot repeat a motion or a task?
- How do I use a *Delay* statement? What are they used for?
- How do I use a *Forever* statement and what are they used for?
- How do I set jump height?

PROCEDURE



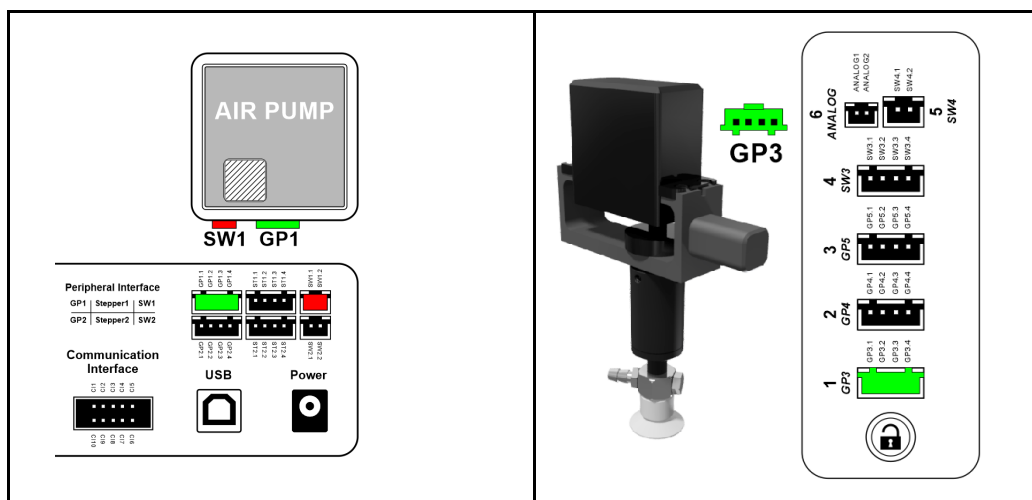
WARNING

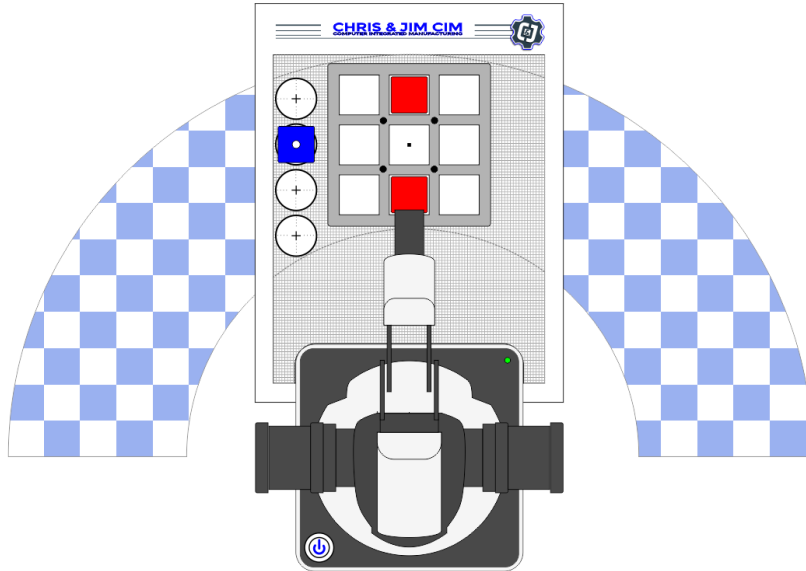


Power OFF

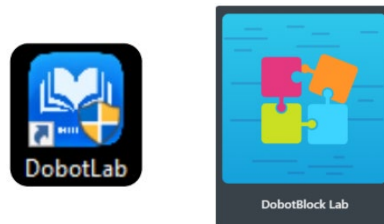
Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Print the Blockly - Pick and Place Field Diagram
2. Set up the robot with a suction cup and place a cube in one of the red squares on the field diagram provided





3. Open up the DobotBlock lab in the DobotLab software .

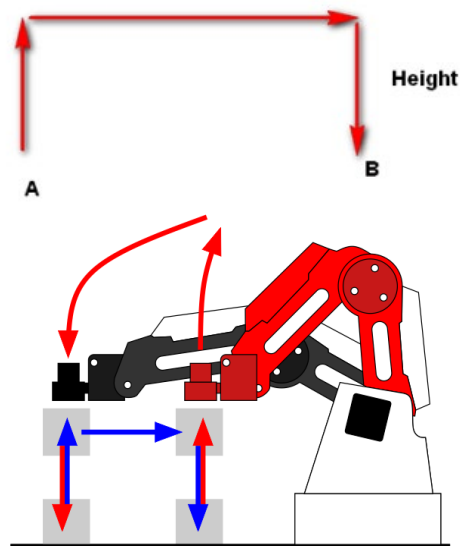


4. Open your Pick and Place Blockly Activity.

In this activity, we will add a **JUMP** movement as well as a **LOOP**. The JUMP command in blockly works the same as it does in Teach and Playback. This will allow us to remove a few lines of code, creating the same operation in less steps.

JUMP Movement - A **JUMP** movement combines three steps into one. It combines the retract, move over, and back down movements into one step. This type of movement simplifies repetitive movements such as a dipping operations or soldering operations. The Z Height for the JUMP movement is defined with a **SETTING** block.

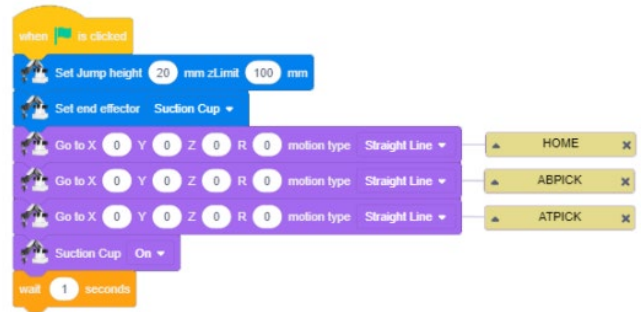
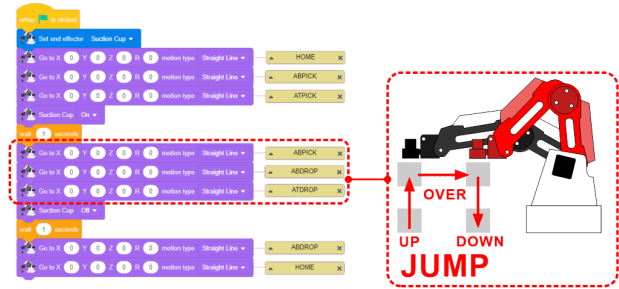
A **JUMP** movement does not replace the initial ABPick to ATPick as well as the final ABDrop.



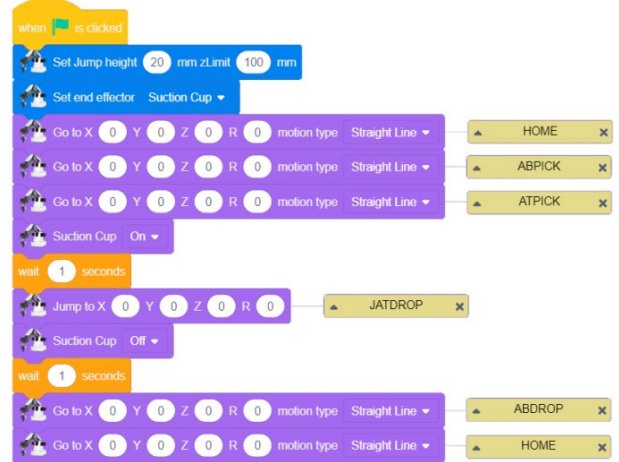
We first need to identify what portions of our last program, if any, can be replaced with a **JumpTo** command.

After we have turned ON the suction cup, we can see a group of code that is an UP-OVER-DOWN. We will combine these three steps into one **JUMP**.

- Isolate the section of code we are going to replace and pull it off to the side (we still need the ATDROP coordinates).



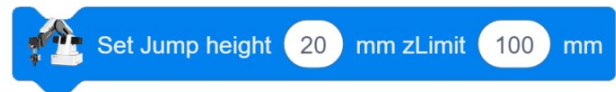
- From the MOTION category, drag over a **JUMP TO XYZR**. Fill in the coordinates with the values for **ATDROP** (the final point we would like to **JUMP** to) and add a comment the block. Reassemble the code with the new **JUMP** line.



- Run your code to ensure proper operation.



- Next, we want to add a command that will allow us to control the **JUMP Z** Height. Drag over a **SET JUMP HEIGHT** from the **SETTING** category.



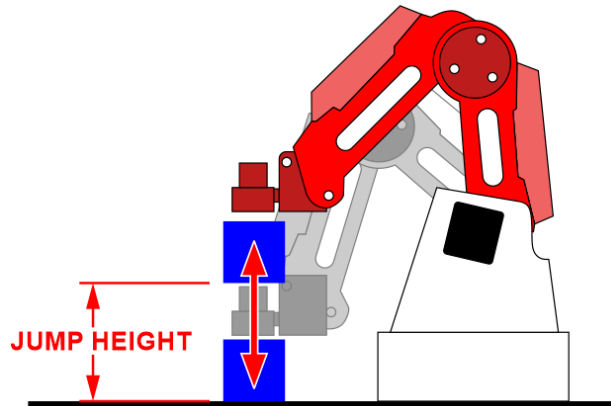
The JUMP height is a relative coordinate, not an absolute coordinate. We want the jump height to be the distance we want the JUMP to travel up for its current location

Example:

ABPICK Z = 2.0 ABSOLUTE

ATPICK Z = -33 ABSOLUTE

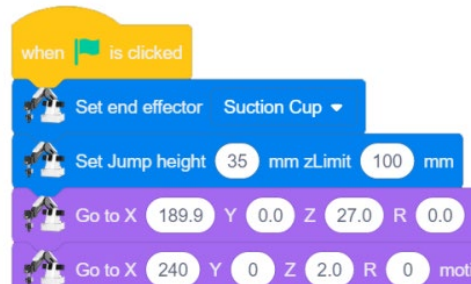
$$\text{JUMP HEIGHT} = (33+2) = 35$$



Since this is a setting command line, we will add it to the top of our program where we assigned the EoAT.



This step can be added anywhere in the program, as long as it is placed before the jump command is used.



If your set up did not work correctly the first time, what did you have to do to make it work?



9. Next, we are now going to make the program loop, or repeat, for a certain number of times. To do this we will be using the **REPEAT** block from the CONTROL category. and the *while* loop, both can be found in the *Loops* section on the left.

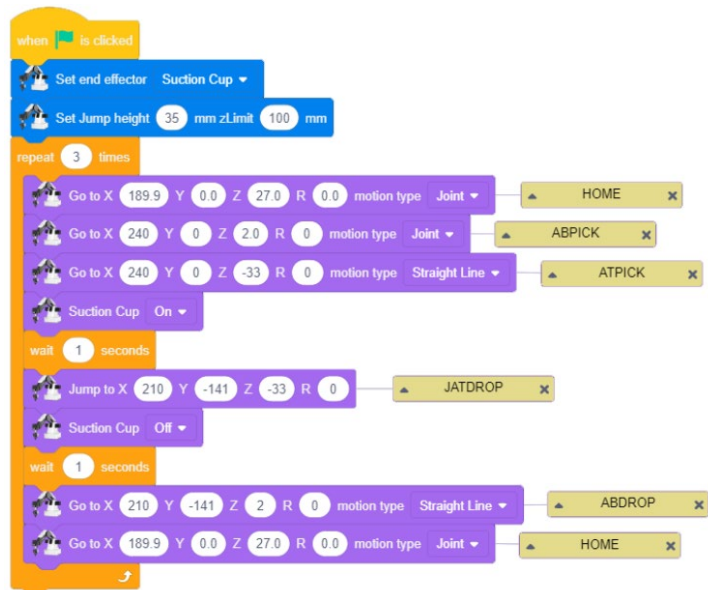
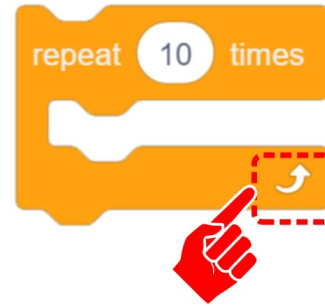


The white arrow at the bottom of a C Block (Referred to as a “C” block because of its shape) indicates that the original

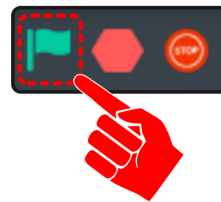
condition will be rechecked to see if it is still true at the end of each loop. If it is, the loop will run again. In this example, our condition is the number of times to REPEAT. The REPEAT will loop until the condition is no longer true.

Drag only the operations we wish to repeat into this “C” block. We will leave the rest of the program outside of the loop.

Edit the REPEAT block to make this section run 3 times

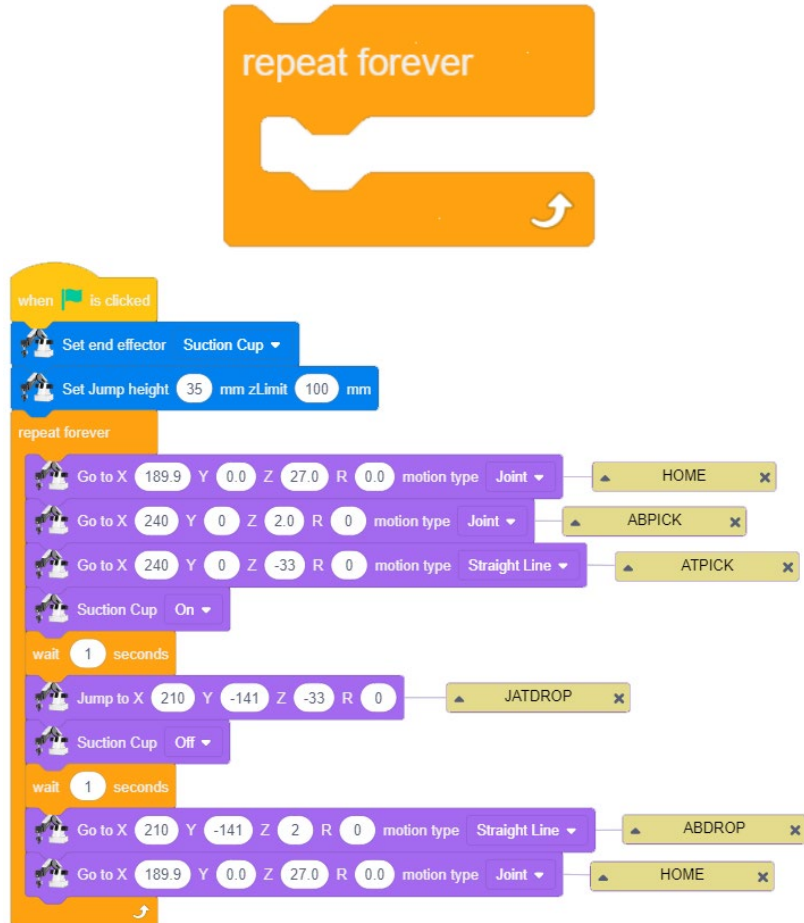


10. Run your code to ensure proper operation.



Often, we need a section of code to run **FOREVER** instead of a set number of times. TO do this, we will use a **FOREVER** block. Drag over a **FOREVER** block from the **CONTROL** category. Replace the **REPEAT** block with the **FOREVER** block and test your program.

Run your code to ensure proper operation.



*If you select the **Repeat** Loop to be deleted before the program section is pulled out of the loop, it will delete the loop and everything it contains.*

Notice that once the program drops off the block and returns to its safe position, it goes back to the start of the program to try to pick up another block and that this will continue repeating indefinitely until you hit the stop button or the robot is turned off.

If your set up did not work correctly the first time, what did you have to do to make it work?



CONCLUSION

1. *What is the difference between a forever loop and a repeat loop?*
2. *Can a loop be placed inside another loop? Give an example of how you might use this when programming in Blockly.*
3. *Describe in your own words how a JumpTo command works. Why is it good programming practice to use JumpTo's?*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- _____ 1. Nesting Loops
Produce 2 different pick and place routines
Routine 1 will repeat three times
Then routine 2 will repeat twice
This process will loop routine 1 and then 2 forever
- _____ 2. Produce a repeat loop for a dipping operation. Use the JumpTo when appropriate



1.6 TP - Palletizing and Roll Angles

NAME: _____

Date: _____

Section: _____

INTRODUCTION

Have you ever wondered why most consumer goods are delivered to your doorstep in rectangular boxes? Think about the shape of the truck they were delivered in. Or the shape of the boxcar or shipping container that they left the factory in. Most consumer goods are moved around on pallets that are rectangular as well.



In this activity you will learn how to palletize rectangular containers. Think about it... every time you move a rectangle from one point to another, there is a good chance that the edges will not line up. In this activity, you will learn how to calculate the amount you need to turn the boxes (**ROLL ANGLE**) to place them on a pallet properly. You will also learn how to make a robot perform the process called **PALLETIZATION**.

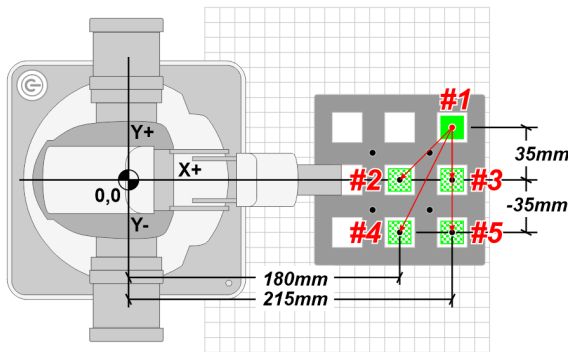
KEY VOCABULARY

- Pallet
- Roll Angle
- Palletization
- Pitch
- Yaw

EQUIPMENT & SUPPLIES

- Robot Magician
- Dobot Field Diagram
- Four or more $\frac{3}{4}$ " cubes
- DobotLab software
- Suction Cup Gripper
- Pneumatic Gripper

PROCEDURE



In this activity, you are going to make the Dobot complete the following task:

- A person places a cube at the START Position #1.
- The robot will wait for five seconds and then pick up the block from Position #1 and place it at Position #5.

Repeat the process of

- Manually replacing a cube at Position #1
- Make the robot wait
- Taking the cube from Position #1 and placing the cube at Position 4, 3, and 2.

- Using teach, record, and the *options* menu in the *teaching and playback* lab of DobotLab make the robot place the two parts on the pallet.

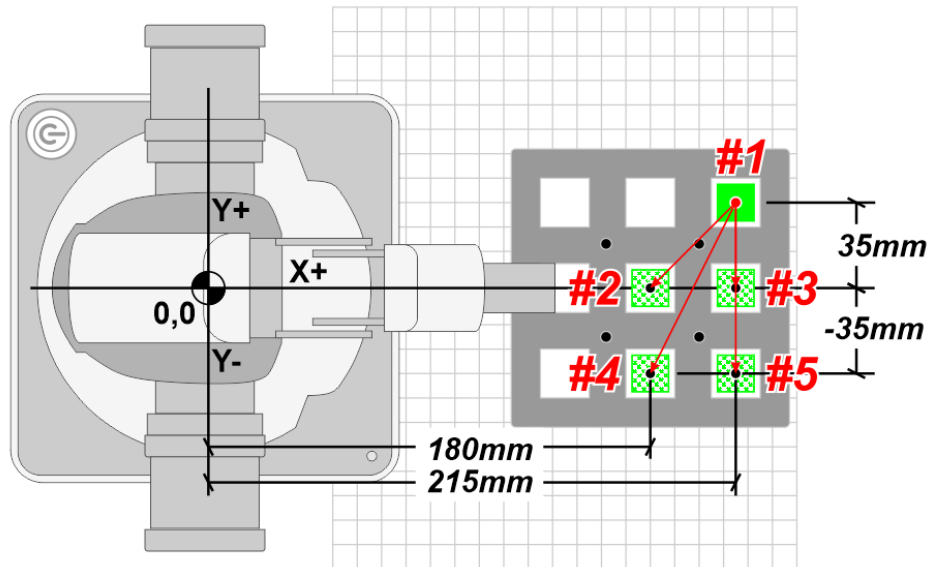
_____ Have your instructor check your work when finished.

- What is wrong with the placement of the objects? Why is this so?

- DobotLab will calculate the **WRIST ROLL ANGLE** for you. In the following steps, we are going to use basic Trigonometry to calculate how the software came up with these positions. After a demonstration by your instructor on how to do this, go ahead and calculate the angles in the problems below.



Move the blocks from the common unload position (Position #1), to the finished pallet positions (Positions #2, 3, 4 & 5) as shown in the diagram below.
Manually replace the block in Position #1 for each pallet position.

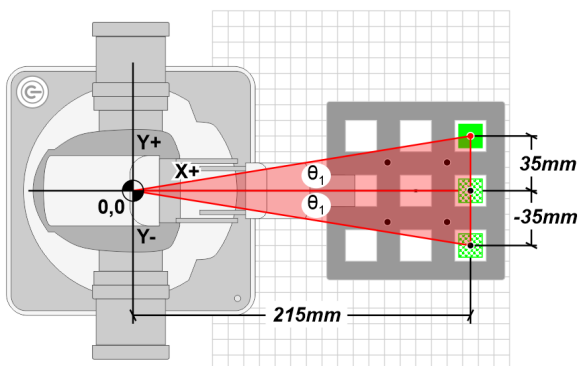


5. In the space provided below, calculate the correct roll angle (θ_1) for positions #1 & #5 relative to the X axis. Show your work even though you are using a calculator.

Note that: **Tan = Opp/Adj** and **= Tan⁻¹(Opp/Adj)**

$$\theta_1^\circ = \tan^{-1} (\text{Opp/Adj})$$

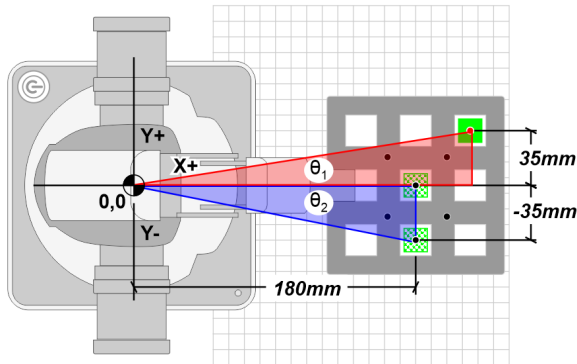
Calculation:



7. In the space provided below, calculate the correct roll angle (θ_2) for position #4. Show your work.

$$\theta_2^\circ = \tan^{-1} (\text{Opp}/\text{Adj})$$

Calculation:



8. For this activity, we would like to keep the WRIST ROLL ANGLE parallel to the robots Y-Axis. To do this we can have the software calculate the roll angle by changing the value in the last column for **WRIST ROLL ANGLE** to zero "0". This will allow the software to calculate the angle and keep this axis the same for each position.

COORDINATES

	MoveJ	P	190 , -0.4 , 27 , 0
move to	MoveJ	P	248.5 , -39.1 , -17 , 0
end effector	Gripper	Open	
wait	1	s	
move to	MoveJ	P	210.3 , -146 , -53.8 , 0
move to	MoveJ	P	210.3 , -146 , -53.8 , 0

R

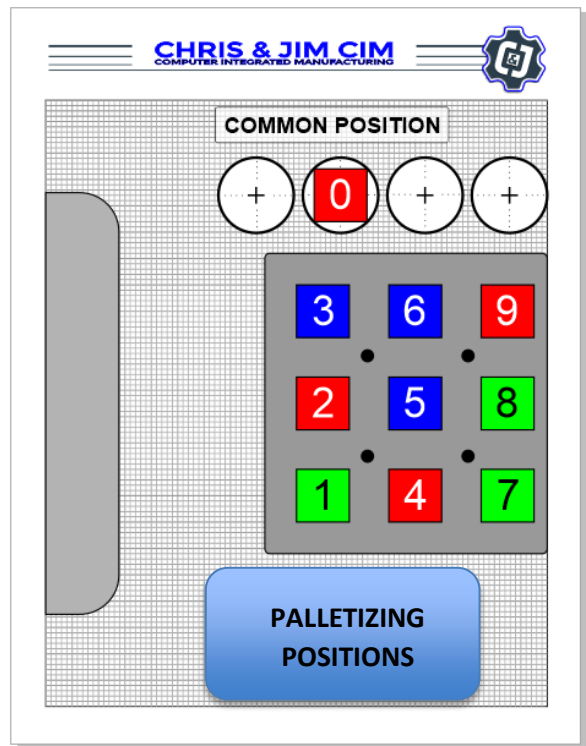


9. Now that we know how to adjust the **WRIST ROLL ANGLE** to keep a consistent angle of a product we pick up, we will use this to **PALLETIZE** a group of cubes for shipping.

NOTE: For this step, you will need at least 9 cubes. See previous activities or ask your instructor if you have any questions on how to do this.

PALLETIZING SCENARIO: (see the diagram on the next page)

- The robot will start at a HOME position
- Make the robot wait for five seconds, while you place a block at position #1.
- Make the robot take a block from this position and JUMP it onto the pallet in the order show below.
- Once the cube is placed onto the pallet, the robot will return home
- The robot will wait until another cube has manually been placed in the common position for palletizing.
- This process will repeat for all 9 steps.



NOTE: The complexity of the programming available in TEACH and PLAYBACK LAB is limited. We will be able to simplify and add to the programming in the DobotBlock Lab.

CONCLUSION

1. Is there a better way to make the palletization process start rather than to use “wait for five seconds? Did this work every single time?
2. Pick another position off of the pallet, sketch the diagram in your notebook or on the field diagram and calculate the roll angle if the robot were to move a part there from position #1. be sure to show your work.
3. If you stacked another layer on top of the layer that is already on the pallet, would the roll angles change? Why or why not? Justify your answer.
4. If you swapped to the mechanical gripper, and touched up the points, would the roll angles still be correct? Why or why not? Justify your answer.
5. What is an advantage to a vacuum or suction gripper, when palletizing, over a mechanical gripper? Justify your answer.



GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- _____ 1. *Change the gripper to the mechanical gripper and try to recreate this.*
- _____ 2. *Stack another set of blocks on top of the first set.*
- _____ 3. *Start with the pick spot off to the side of the pallet and fill a pallet with one layer.*
- _____ 4. *Complete a second layer after completing #3.*



1.7 Block - Developing a Cube Matrix

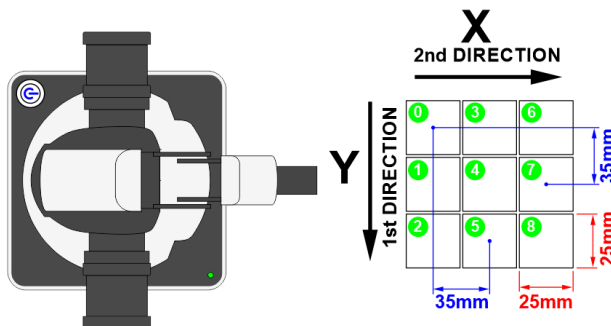
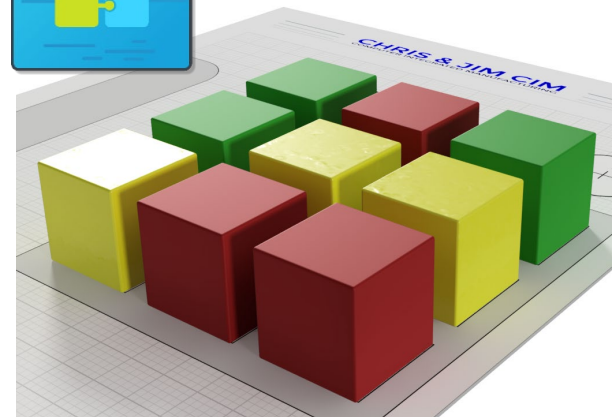
NAME: _____

Date: _____

Section: _____

INTRODUCTION

Matrices come in many different sizes throughout industry and can be used in other areas besides robotics. Many companies will use a **Matrix** in order to efficiently store products and materials, so it is important for you to know how they work and how to program one yourself. It is especially helpful for **palletizing** routines when placing boxes on a pallet efficiently.



In this activity, you will simulate removing boxes from a pallet.

This process is called **De-palletization**.



Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS shutdown the Dobot before making connections or damage to the robot could occur.

KEY VOCABULARY

- Forever Loop
- Repeat Loop
- Matrix
- SuctionCup
- Delaytime
- Function
- User-Defined Function
- Inputs
- Variable

EQUIPMENT & SUPPLIES

- Robot Magician
- Dobot Field Diagram
- 1" cylinders or cubes
- DobotLab software
- Suction Cup Gripper
- Dobot Input/Output Guide

KEY CONCEPTS

Essential questions answered in this activity include:

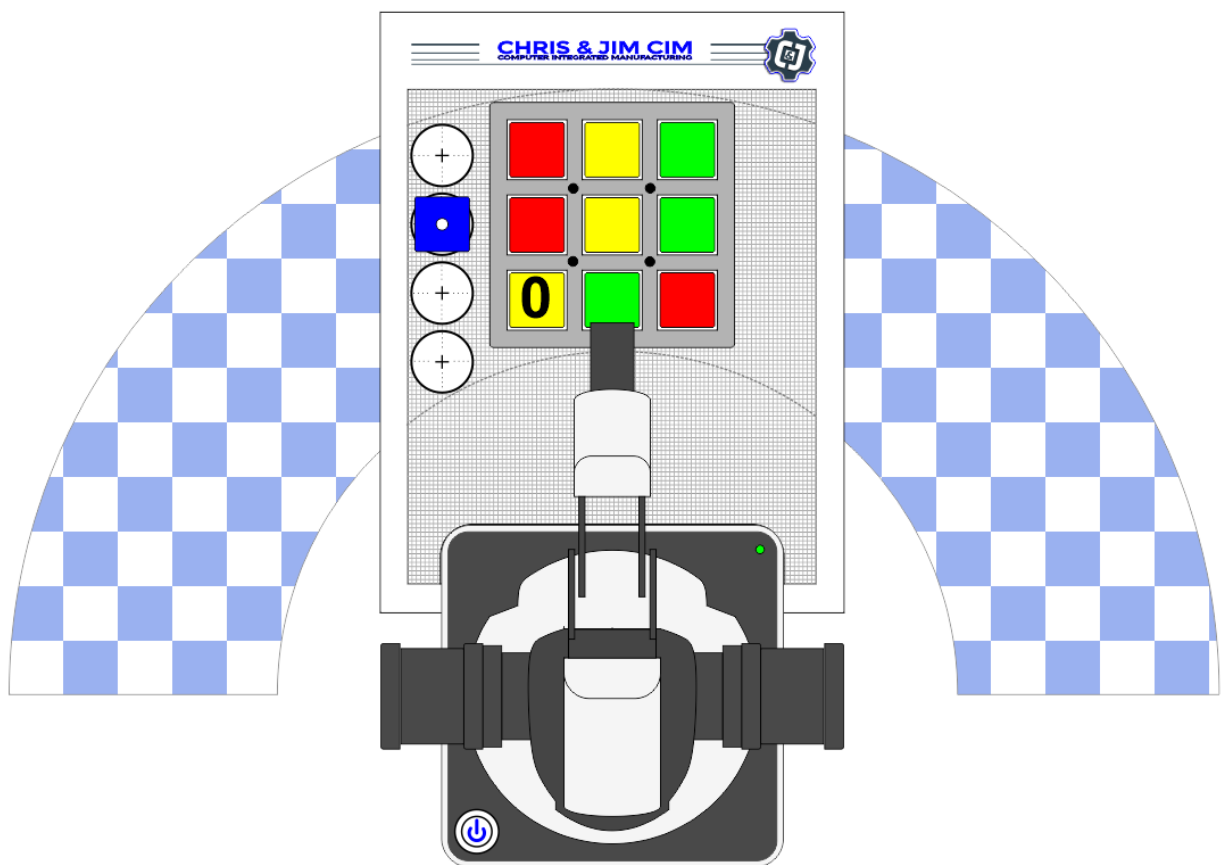
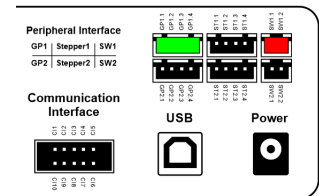
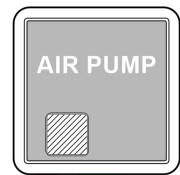
- Why are **variables** used?
- How do I use **variables** in blockly?
- How can I use math to palletize?
- What are **MyBlocks** and how do they simplify programming?
- What are some of the blockly commands needed to do this?
- Why is **palletization** and **de-palletization** important in industry?

PROCEDURE

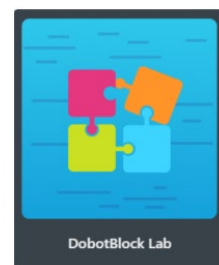


Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Print the Dip Tank Field Diagram
2. Set up the robot with a suction cup and place a cube on all of the empty squares on the field diagram provided. (Cube color does not matter for this activity)



1. Open up DobotBlock Lab in the software and connect the robot

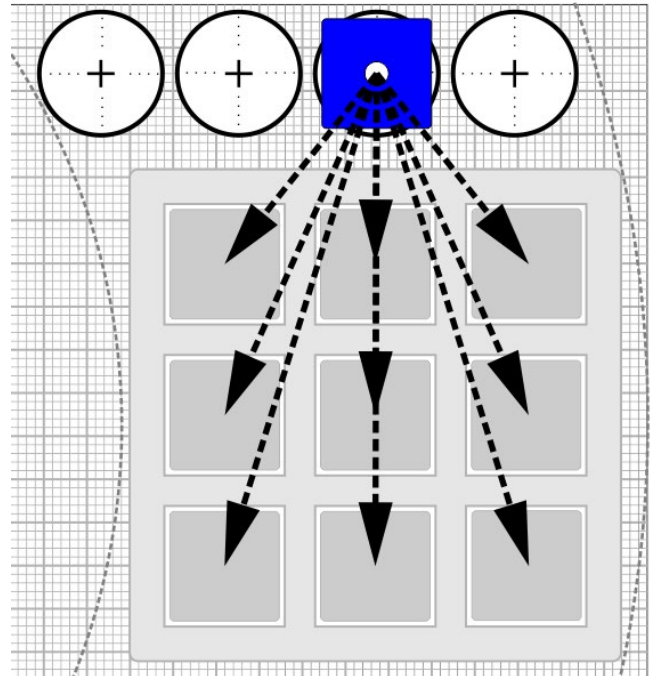


Programming the robot to pick up each cube from a matrix of cubes and drop them all off at a common location is not necessarily hard to do, just tedious, time consuming, and numerous lines of code that are repetitive motions. This same process can be done in reverse, such as when you move parts from a feeder and place them in a matrix. This is called *palletization*.

From this activity we can learn how to condense and simplify our program when we have things in common or repeated.

In this exercise we have the following constants:

- Above Z locations are the same
- At Z locations are the same
- Place X and Y locations are the same
- They are all EVENLY spaced in a regular matrix.



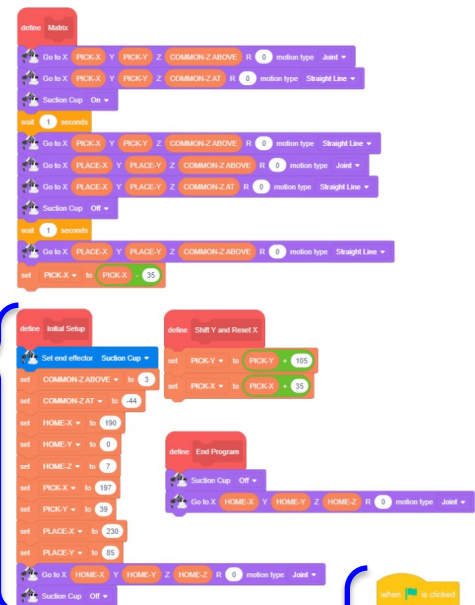
The only thing that keeps changing, the variable, is the X and Y coordinates of the cubes in the matrix field.

We can use these common factors to simplify our operations and our code. We will do this with user defined *variables* and *MyBlocks (Functions/Voids)*.

MyBlocks (Functions/Voids) are a named section of a program that performs a task that will be repeated over and over again in our program. This is a procedure or a routine that will greatly simplify our otherwise complicated program.

MYBLOCK (FUNCTIONS)

SETUP & DEFINED VARIABLES



ACTUAL PROGRAM

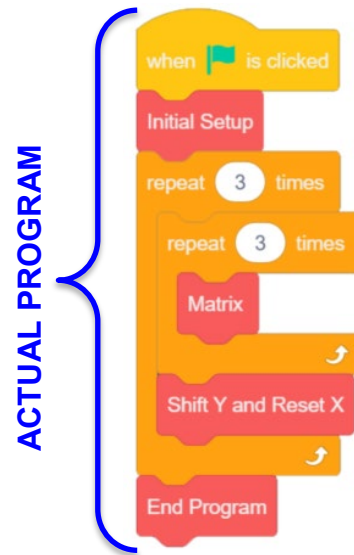
An example of a program that uses MyBlocks/ Functions



Functions/Voids are several lines of code that are needed to complete a single task or operation. They allow a programmer to group thoughts or actions.

We use Functions/Voids when:

- **Used Multiple Times:** Functions can be called to run multiple times throughout a program.
- **Simplifying Programs:** Allows repeated code to be condensed into a single line of code throughout the main program.
- **Simplifying Editing:** Code that repeats itself multiple times throughout a program now only needs to be edited once.

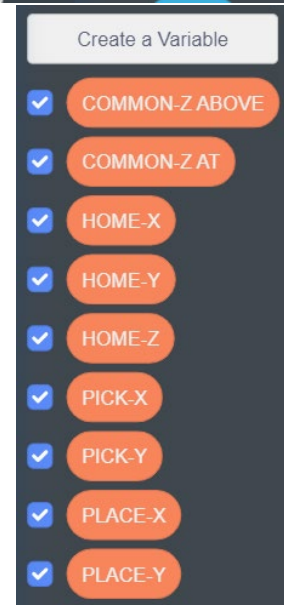
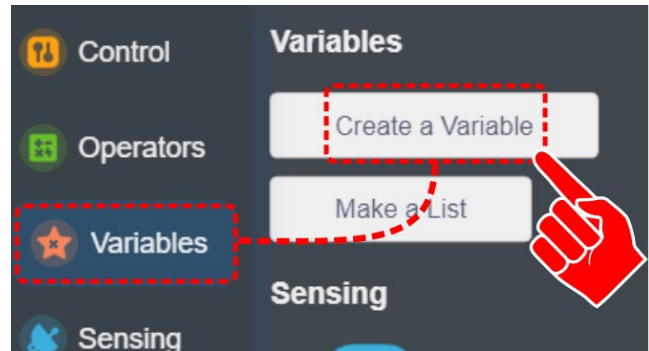


The actual program after using MyBlocks/Functions!

We will start our program by defining a set of variables. Think of *Variables* as a group of lockers where information can be stored. The names of the lockers are user defined. Each time the locker's name is called, referenced, or used in the program, the name is replaced in the program by the value stored inside it. Variables can also be updated in a variety of ways.

Start by creating the list of variables below

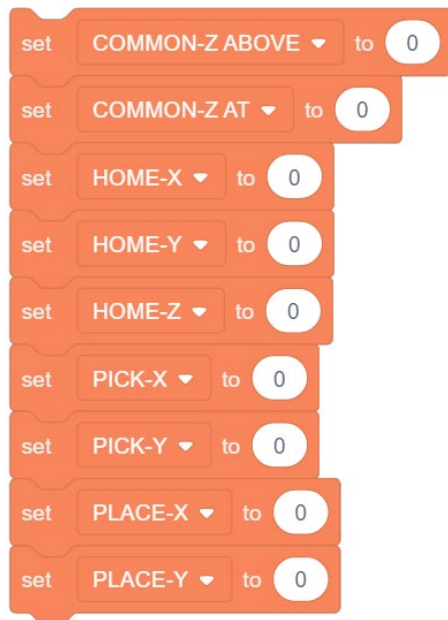
- COMMON-Z ABOVE
- COMMON-Z AT
- HOME-X
- HOME-Y
- HOME-Z
- PICK-X
- PICK-Y
- PLACE-X
- PLACE-Y



Next, from the *Variables* section, drag over a *Set Variable* block into the programming field.

Either duplicate this block or drag over a total of 9 Set Variables.

Change the variable names by clicking the down arrow on the Set Variable block



The next step is to start finding each of our starting values.

Using either the Jog Controls, or the Lock Button find the values for the table below.



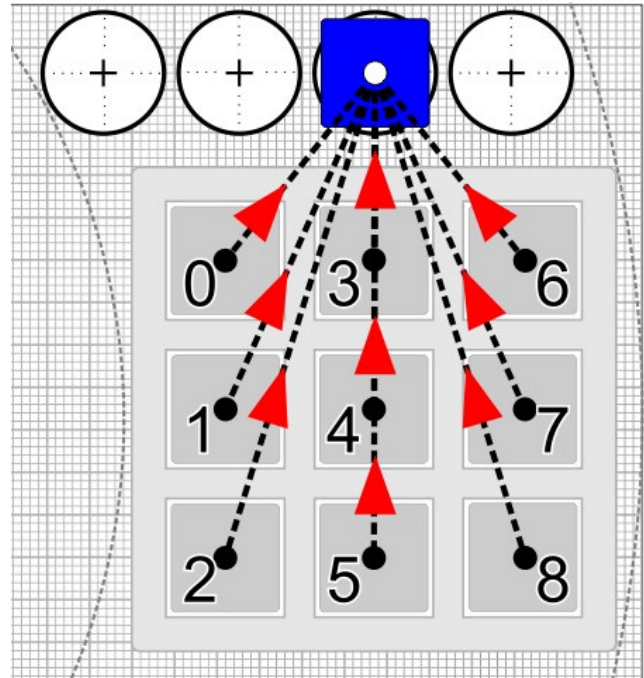
It makes it easier to use the Lock Button to get close to the desired position and then use the Jog Controls for smaller movements.



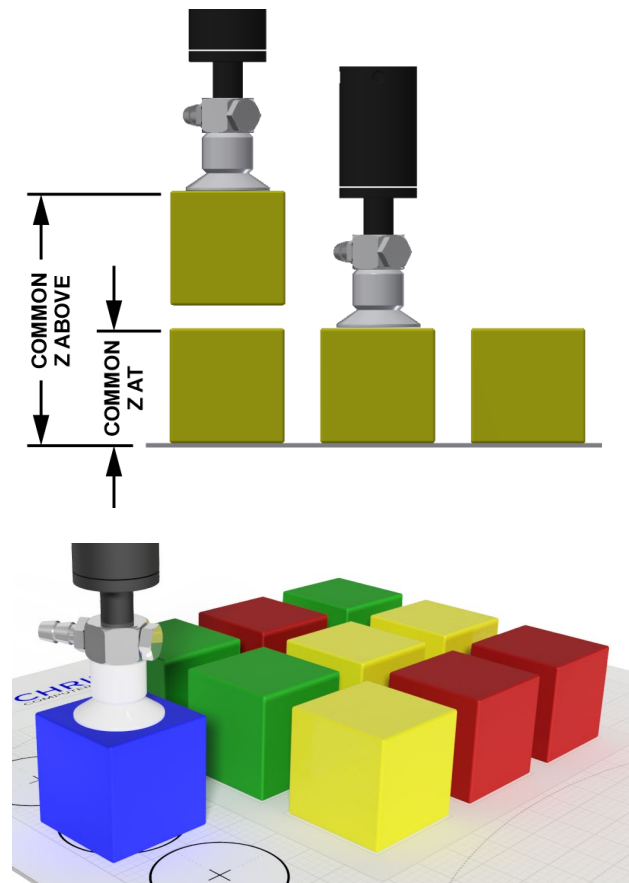
DE-PALLETIZING SEQUENCE

- Round all values to the nearest whole number
- Align the center of the vacuum gripper to the center of the cube
- Write the values in the table below or in your notebook for future reference.
- **HOME-X, HOME-Y, & HOME-Z:** Find a comfortable Home position that can be used for the project.
- **PICK-Y & PICK-X:** Use the center of Cube '0', from the diagram on the right, as the X and Y location for the Pick Location.
- **PLACE-X & PLACE-Y:** Use the location of BLUE cube for the common Drop-Off point.
- **COMMON-Z AT:** Find a common Z value that can be used to PICK UP all of the blocks.
- **COMMON-Z ABOVE:** Find a common Z value that can be used to raise up and carry all of the cubes. Ensure this position is high enough to clear the other blocks

COMMON-Z AT	
COMMON-Z ABOVE	
HOME-X	
HOME-Y	
HOME-Z	
PICK-X	
PICK-Y	
PLACE-X	
PLACE-Y	



DE-PALLETIZING SEQUENCE



Part 1: Setting Variables

Start by recording the **COMMON-Z ABOVE** position in the number window of the block we created



*If you select **Rename Variable**, it will **Rename** that variable and every where it has been used in the program*

The order in which we set the **variables** does not matter in this program.

It is good practice to start each program by sending the robot to a HOME position, turning off or opening the EoAT (End of Arm Tooling) and turning off any DIGITAL OUTPUTS.

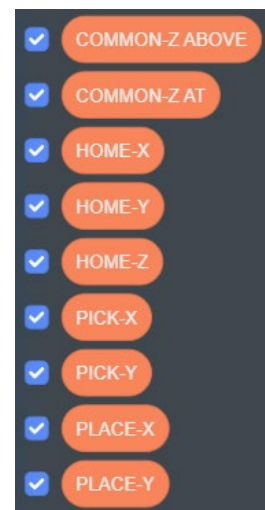
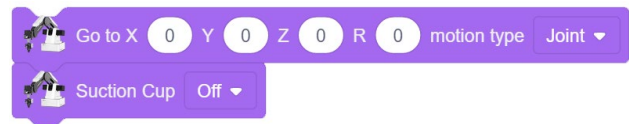
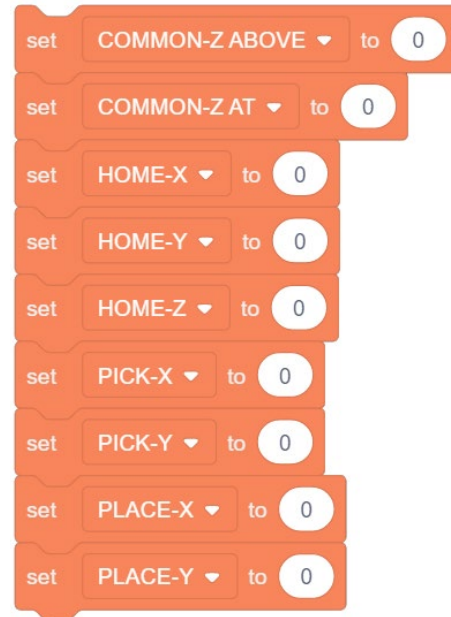
Drag over a GO TO (JOINT) and a SUCTION CUP (OFF)



*Remember that these tools come from the **Motion** section.*

We will now use some of the user defined **variables** that we just created.

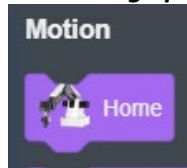
From the Variable section, you should see that all of the variables that we have created are now options to use.



Drag over your **HOME Variables** (not the **set HOME Variables**) and drag it into their corresponding positions for the **GO TO** step. Leave the Roll Value as "0".

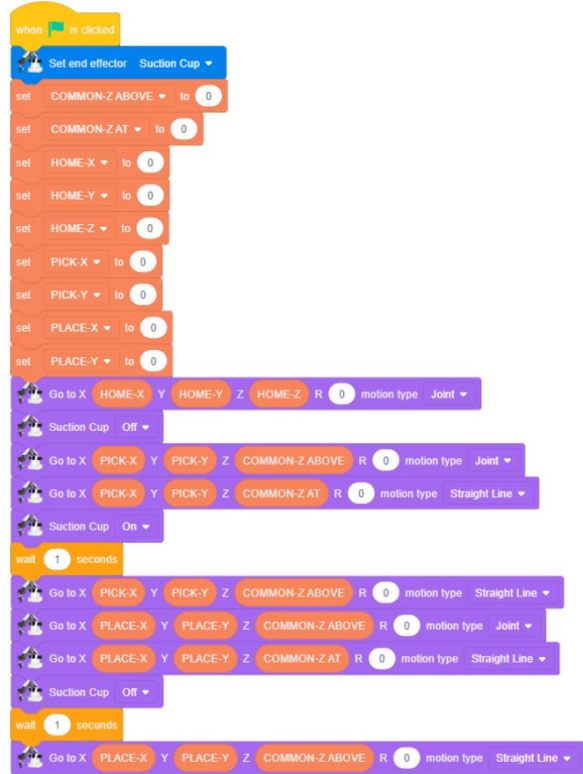
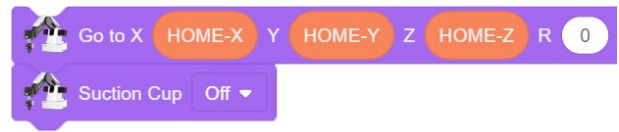


Do not use the HOME Block. This block runs the homing operation.



Assigning variables helps when coordinates in a program need to be changed multiple times or is going to be used as a template for multiple programs. They can be altered once at the beginning of the program and that alteration will update everywhere they were used throughout the program. It also makes reading the program easier as the user now sees words in place of number values.

Next, add all the steps needed to go get the first block '0' from the first assigned pick location and move it to the place (BLUE) location.

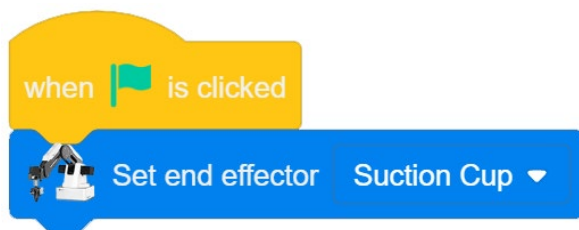


Link everything we now have as one program and add to the top of the program:

EVENTS: WHEN STARTED

&

SETTINGS: SET END EFFECTOR (EoAT)



Once the program is linked, run it and see if it works correctly. If it does not work, troubleshoot it until it does.



Remember: if you hit a limit with the robot at any time, or you hear a clicking or a grinding noise, it's always a good idea to home the robot again. Also, if the robot does not return to the same position, just re-home it.

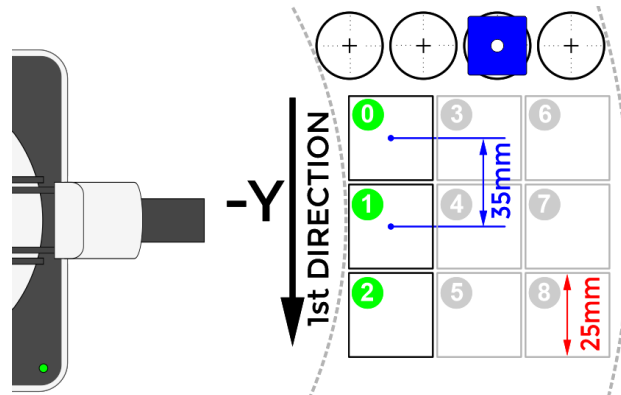
If your set up did not work correctly the first time, what did you have to do to make it work?

ALL NEW - THE TRICKY PART!!!

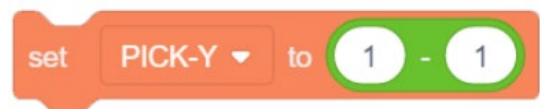
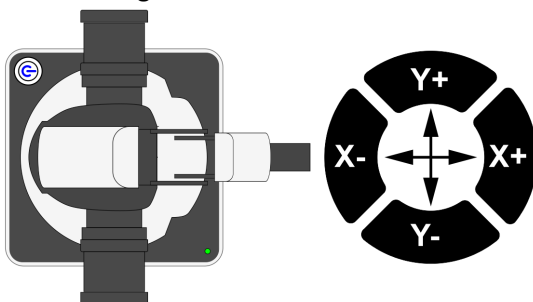
Now it's time for the second cube.

The only difference in the programming for this cube should be the X Position.

From the diagram shown, the blocks are 35mm apart and that the block is moving *toward* the robot. Moving toward the robot is shown mathematically by subtracting 35mm from the current X position value for block number 2 and again for block number 3.



In order to adjust the X position, set up a block (**Return the inverse (subtraction) of two numbers**) from the OPERATORS section, and attach it to the **SetVariable** block. Attach the two blocks together.



In order to subtract 35mm from the current PICK-Y value and assign that value to the new PICK-Y value, we will create a computation block that replaces the first value with PICK-Y and the second value with 35.

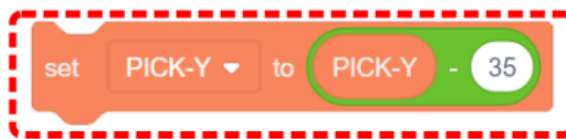
Current Y value equals itself minus a constant of 35 (Negative due to the direction of motion)

Mathematically: $PICK-Y = PICK-Y - 35$



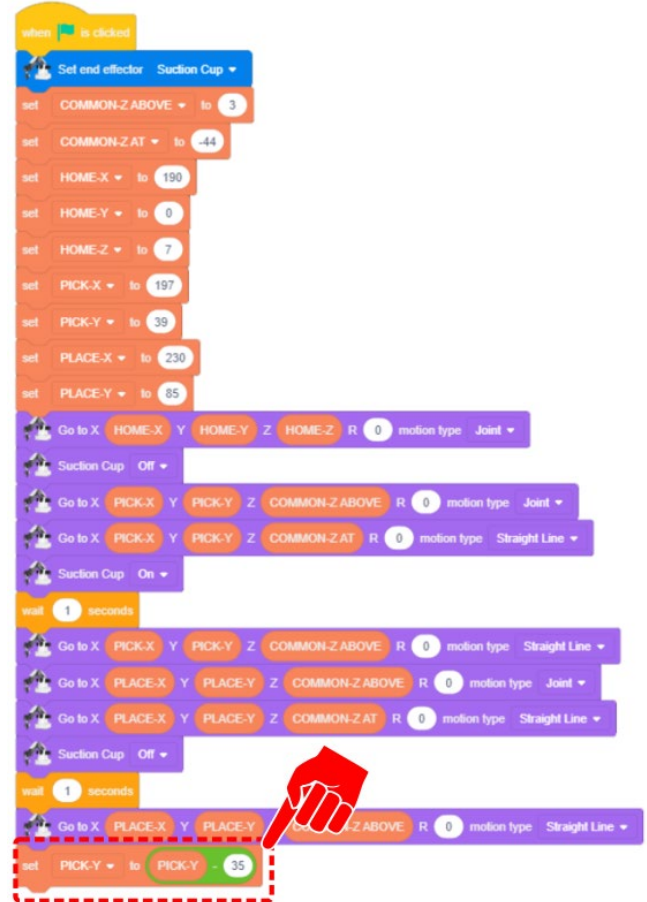
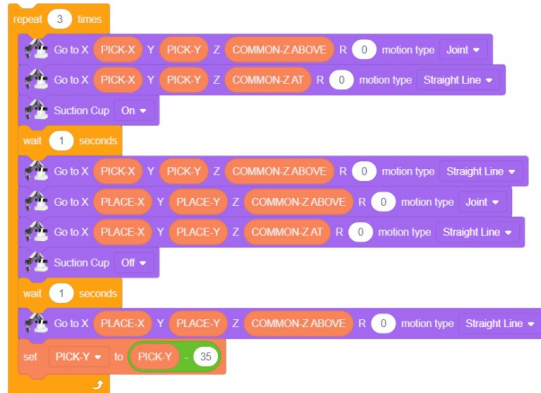
Grab the PICK-Y Variable from the Variables section or duplicate it from the main program

Add this block line to the bottom of your program



Now that the **PICK-Y** value has been adjusted, repeat the entire process of picking up the second and third cubes from row 1 and dropping them off at the place location.

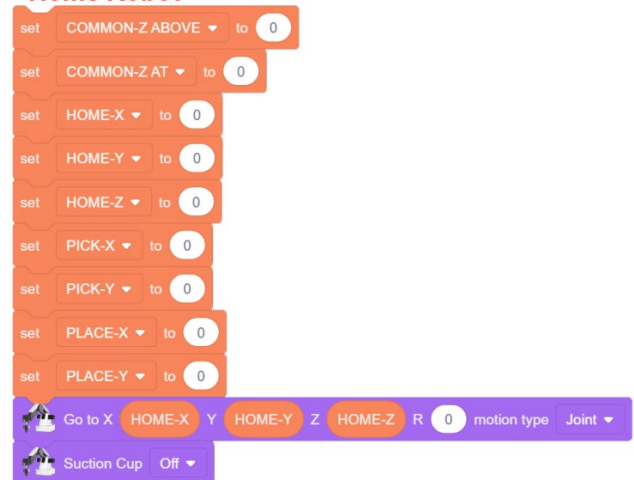
Instead of re-creating all of STATE 2 two more times, we will place STATE 2 inside a REPEAT LOOP. Drag over a REPEAT LOOP from the Control section. Grab the 1st *Go To X* and drop it inside the REPEAT LOOP.



STATE 1

SETUP STATE

- Assign Variables
- Home Robot



STATE 2

PALLETIZE STATE

- Pick Block
- Place Block
- Reassign Variables



We now have the ability to empty one full row (Block 0, 1, & 3).

Now is the fun part.

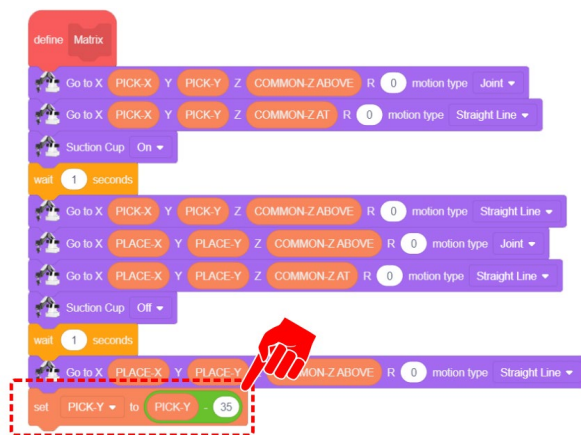
At the end of the 3 times loop, all we need to do is shift over to the next column and reset the PICK-X to its initial value

STATE 3

SHIFT STATE

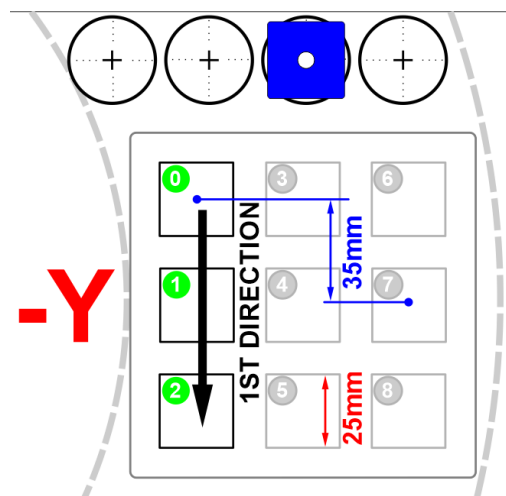
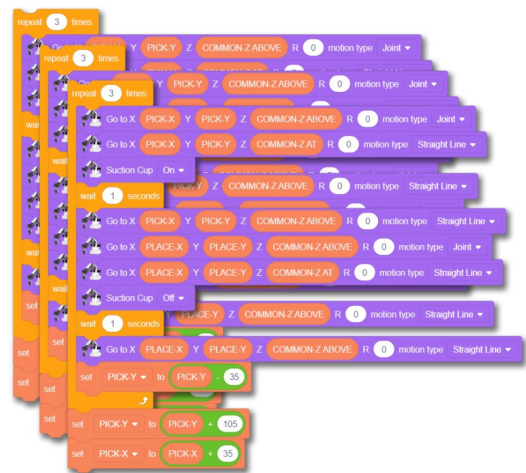
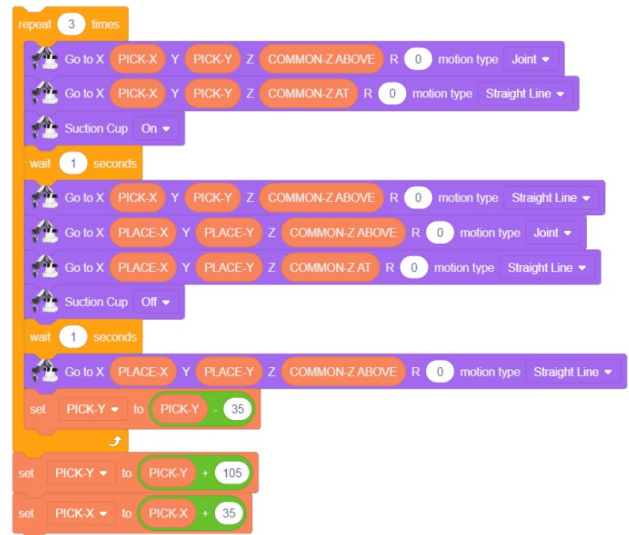
- Shift Column Variable
- Reassign X Variable

... Lets see... we could go back and change the initial value for PICK-X... or... we could reverse the math... we took off -35mm twice... so... $(35+35=70)$... $PICK-X = PICK-X + 70$?... Oh, Do not forget... the -35 was taken off at the end of each loop... so we accidentally took off "3" 35 increments... so $(35+35+35=105)$ $PICK-X + 105$... This just sets it back to what it was.

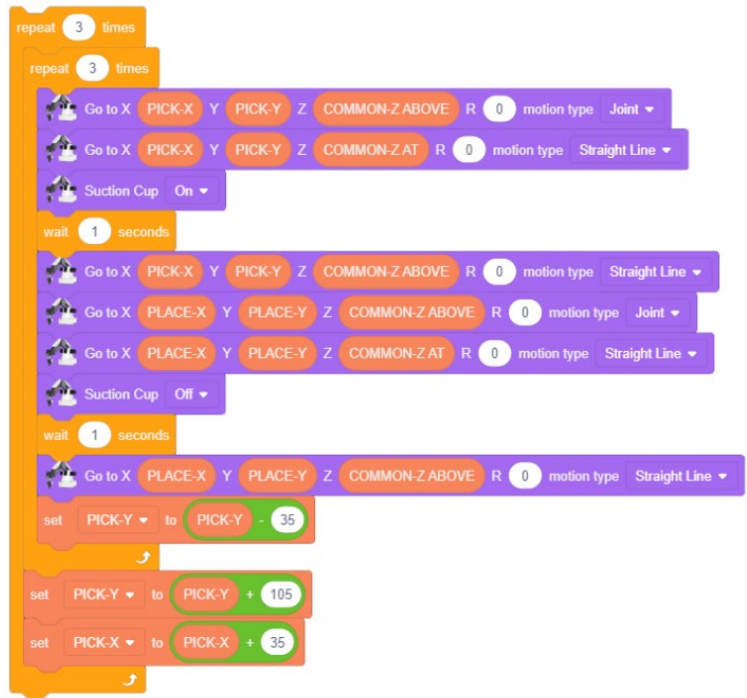


We could then duplicate the entire process two more times...

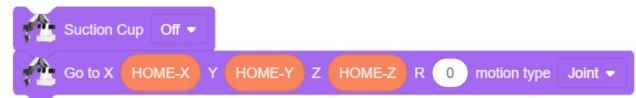
However, using this method, the program becomes very long, and we would have to do a lot of scrolling to move around in our program. Editing or updating the program can also be tedious and leaving a lot of room for errors.



There must be a better way to keep the program simpler, shorter, and easier to edit... if we notice that we are duplicating the exact same program three times... can we just REPEAT the REPEAT LOOP 3 times? Drag over a second REPEAT LOOP and drag all of the entire original REPEAT LOOP inside of it (a nested loop, or a loop inside of another loop).



We can then duplicate and add one final SuctionCup Off and a final Go To Home to end the code.



Add this to the program, then try to run it.

The program still seems very long even though variables and loops were used to make it shorter and simpler.

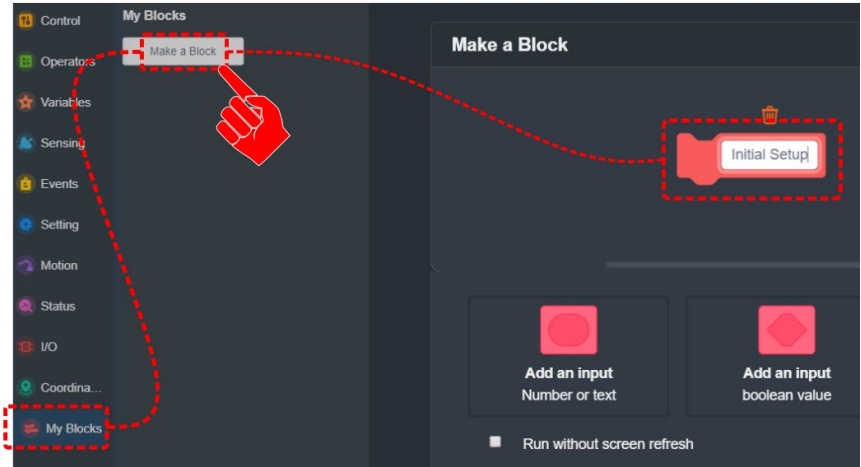
The next step is to clean up the program even further with a block of code called a function or void. Blockly programs refer to this code as creating MY BLOCKS. A *function* is a named section of a program that performs a task or a group of tasks. It can also be considered a procedure or a routine and can greatly simplify otherwise complicated programs.

When a program calls MY BLOCKS to run, the program pauses, runs the MY BLOCK, and then returns to the program where it left off.



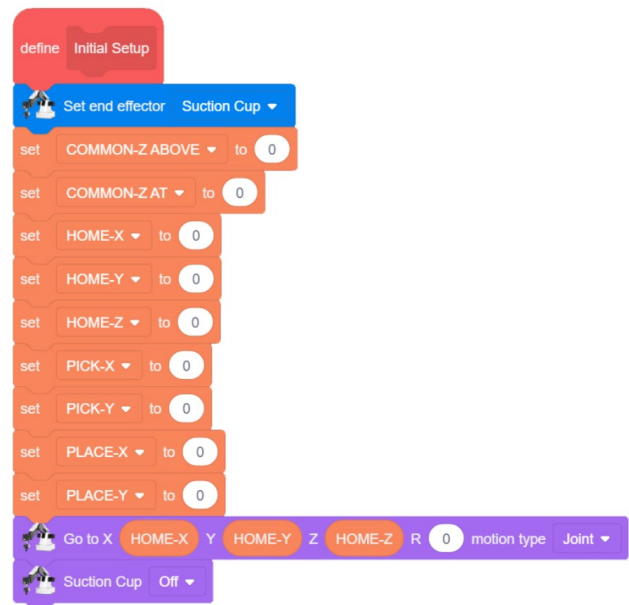
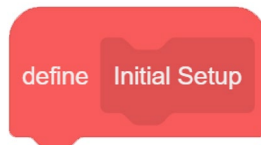
Click on “Make a Block” from the My Blocks toolbar.

Name the first MY BLOCK
Initial Setup



Once the new MY BLOCK is named, a new header will appear in the programming area.

Drag all of the setup code over and attach it to our new header.

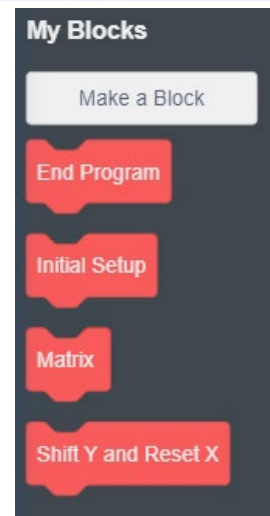
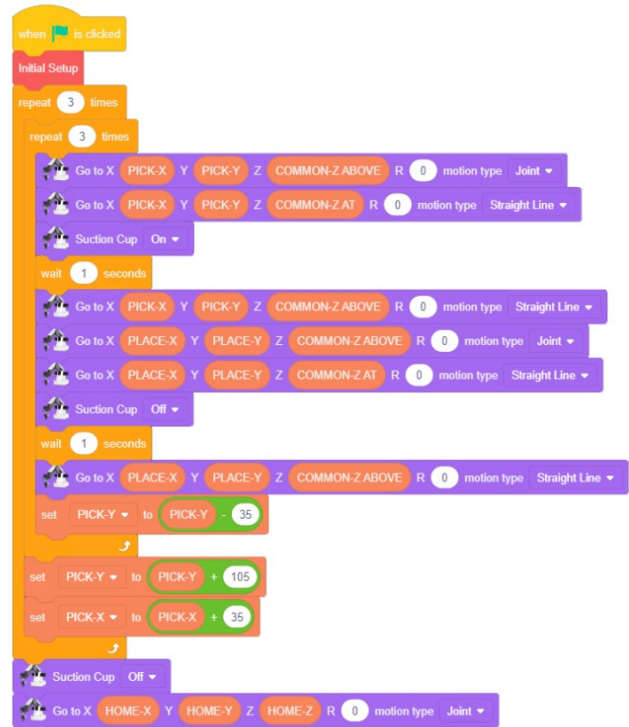
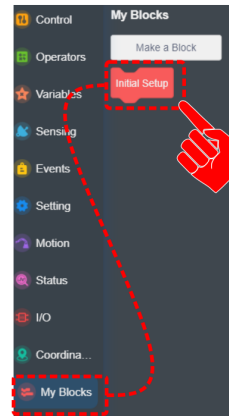


Now that the MY BLOCK is defined and the tasks are added to it, we need to add the ability for the main program to call the MY BLOCK to run. Drag the new block from the “My Blocks” tool bar over to the program area and attach it to the WHEN STARTED

Link the new header back on to the remaining portion of the program and run it... just to make sure everything is still running correctly.

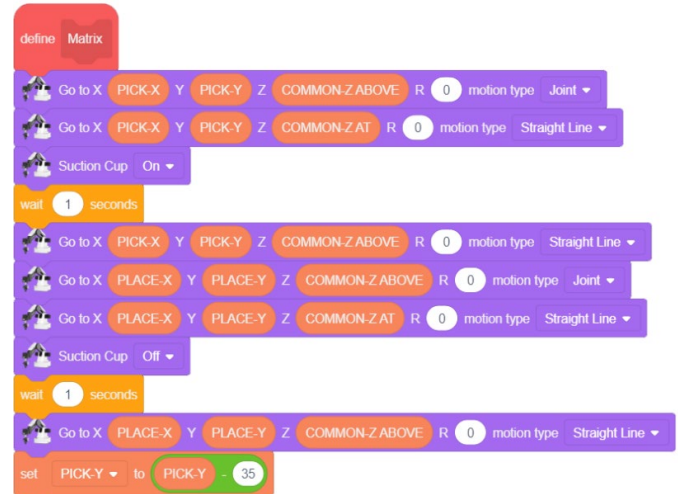
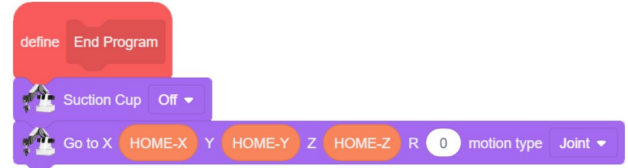
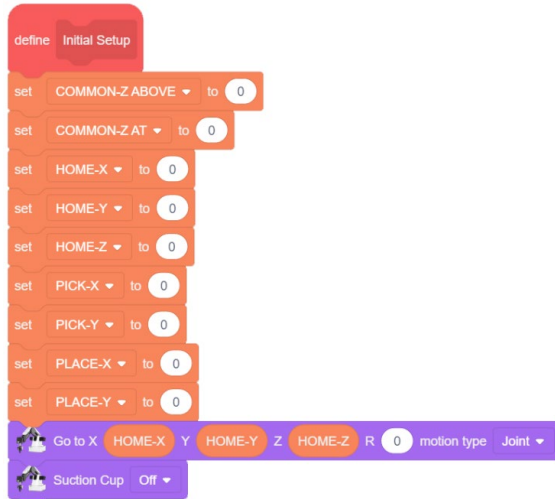
To clean-up the program even more, lets break it up into a few more groups and create additional my blocks

- Initial Setup – Any setup coding, Initial Variable values, sending the robot to our defined home and turning off the EoAT and digital outputs.
- Matrix – Matrix program for one full row
- Shift Y and Reset X – Move over to the next column and reset the X to the initial value
- End Program – ensure the EoAT is off and send the robot back to home



Drag each section of the program over to their new headers

MY BLOCKS just float in the programming environment unattached to the main program. They are essentially separate mini programs to be called by the main program.



Install the new MY BLOCKS into the program and test it.

Developing MY BLOCKS (Functions/Voids) not only produce groups of code that can be used multiple times but can also help a programmer organize their program into mini programs or separate thoughts.



Once the program is written, run it and see if it works correctly. If it does not work, troubleshoot it until it does.



If your set up did not work correctly the first time, what did you have to do to make it work?

CONCLUSION

1. *In your own words, define a variable.*
2. *In your own words, define a MY BLOCK (Function/VOID).*
3. *Explain what would have to be done to palletize two layers using bullet points or a step-by-step list below.*

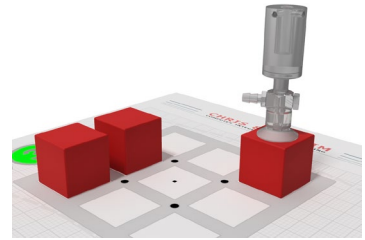


GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

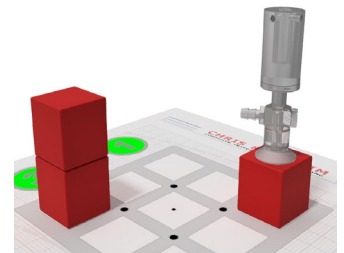
1. Use a switch as an input to make the robot wait until you move the block from the Place position and hit the switch.

2. Move a row of three cubes to another row (Matrix to Matrix)



3. Develop a program that will run the entire process in reverse. Take a block from a common location and distribute them into a 3x3 matrix. This process is referred to as palletizing.

4. Take the cubes from a common location and stack them in a column vertically using variables for the height.





- CHAPTER 2 - ROBOTIC COMMUNICATIONS

CHAPTER INTRODUCTION:

Robotic arms in industry need to communicate with other robots, as well as with other machines, within a work cell, or factory. This form of communication is called handshaking and can be done safely between different machines, devices, and robots. Handshaking is a very simple form of communication and is done with simple ones and zeros; or “ons” and “offs”.

Example:

- Robot places a part in a machine to be processed.
- The robot tells the machine the part is ready and the machine can safely start by sending a signal.
- The machine returns a signal to the robot when that the part is done and the robot can retrieve the part and send it on its way to the next process.

TOPICS COVERED:

- Optical Isolators
- Handshaking with a Rely Module
 - Robot to Robot
 - Microcontroller to Microcontroller (VEX V5)
 - Robot to Microcontroller
- Internal Handshaking
- Wireless Bluetooth Handshaking



2.1 TP - Handshaking: Dobot to Dobot

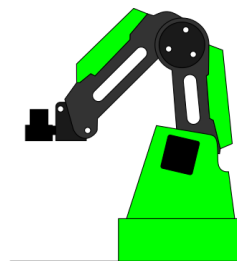
NAME: _____

Date: _____

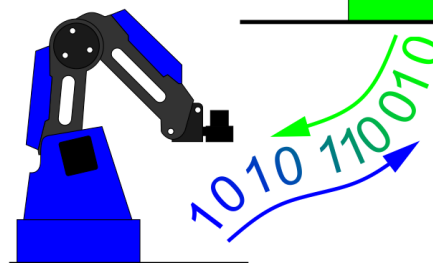
Section: _____

INTRODUCTION

Robotic arms need to communicate with other robots in a work cell, or factory. This is called **HANDSHAKING** and can be done between different machines, devices, and robots. It is a very simple form of communication and is done with simple ones and zeros; or “ons” and “offs”.



In this activity you will learn how to make a robot handshake with another robot. Robot 1 will pass a part into another robot's work envelope and then send a signal and go to a safe position. The other robot will then receive the signal, get the part, and place it somewhere else.



WARNING



Power OFF

Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS shutdown the Dobot before making connections or damage to the robot could occur.

KEY VOCABULARY

- Handshake
- Signal
- Pull down resistor
- Optical Isolator
- Text
- Current limiting resistor

EQUIPMENT & SUPPLIES

- 2 Dobot Magicians
- Dobot Field Diagram
- 1" x 3/4" cylinders or 3/4" cubes
- Servo extension cables
- Dobot Input/Output Manual
- Breadboard/wire/4N25 Optical Isolator and 100 ohm & 4.7K ohm resistors
- DobotLab software
- RobotC or other VEX control software
- Pneumatic Gripper or Suction Cup Gripper
- Handshake module
- VEX bump switch or limit switch
- VEX, PIC, Arduino may all be used in this activity, but wiring may vary.

ESSENTIAL QUESTIONS

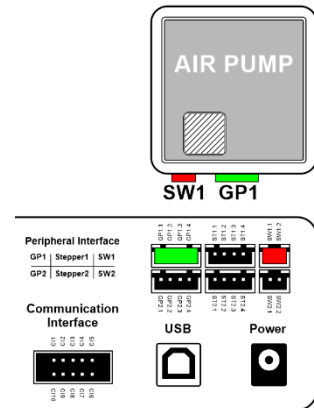
Essential questions answered in this activity include:

- How do I make a robot send a signal?
- How do I get a robot to receive a signal?
- How is this done in Dobot Lab Software?
- How do I make two robots talk to one another?

PROCEDURE

1. Set up both **Robots** with a suction cup - **GP1 & SW1** and place Dobot field diagrams, taped to the work surface, between the two robots.
2. Wire **Robot1** with an **OUTPUT** signal **GP2 - EIO13**.
3. Wire **Robot2** with an **INPUT** signal **GP5 - EIO5**.
4. Wire both Robots to the Handshaking module as shown in the following pages.

Be sure that wires are not going to be pulled out by the motion of the robots.



Open Loop System Block Diagram: Robot #1 acts as the input for robot #2. Robot #2 does not communicate back to robot #1 so there is no feedback. The Handshake Module acts as the device that helps make the handshake happen safely.



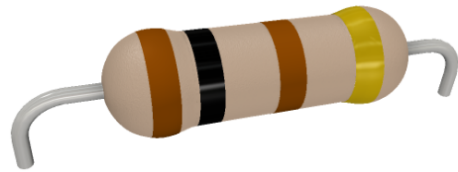
WARNING



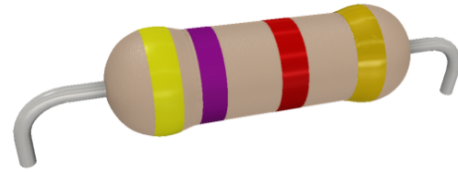
Power OFF

Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.



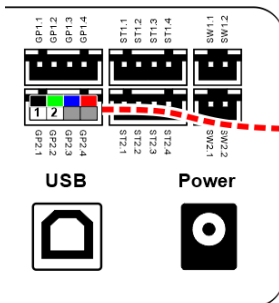


BROWN	BLACK	BROWN	GOLD
1	0	X10	5%

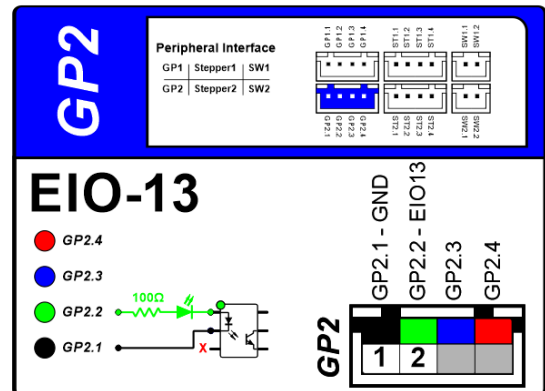
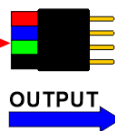


YELLOW	VIOLET	RED	GOLD
4	7	X100	5%

EIO-13

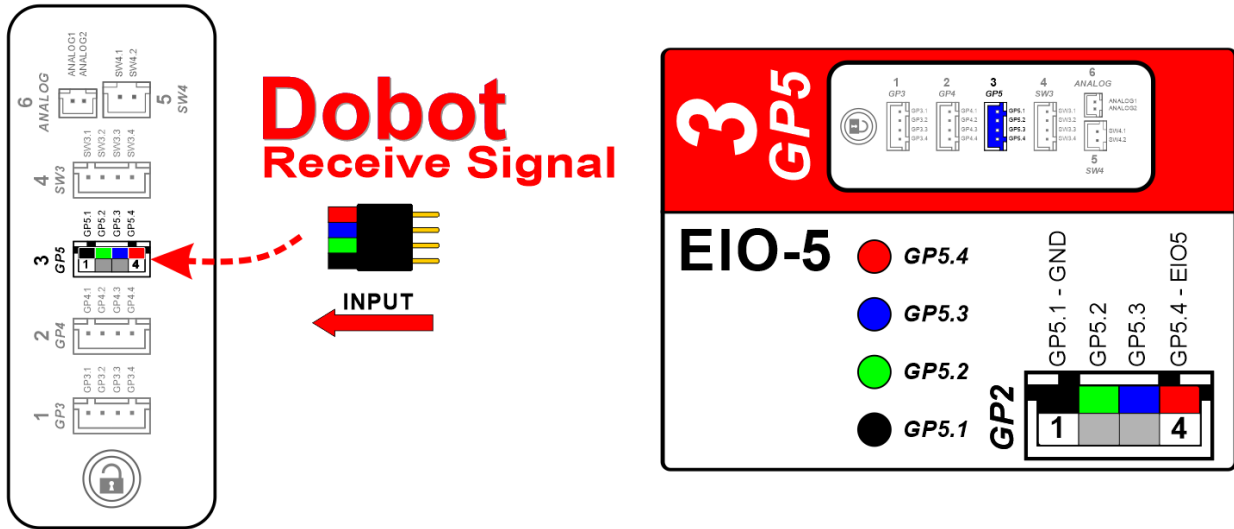


Dobot Send Signal

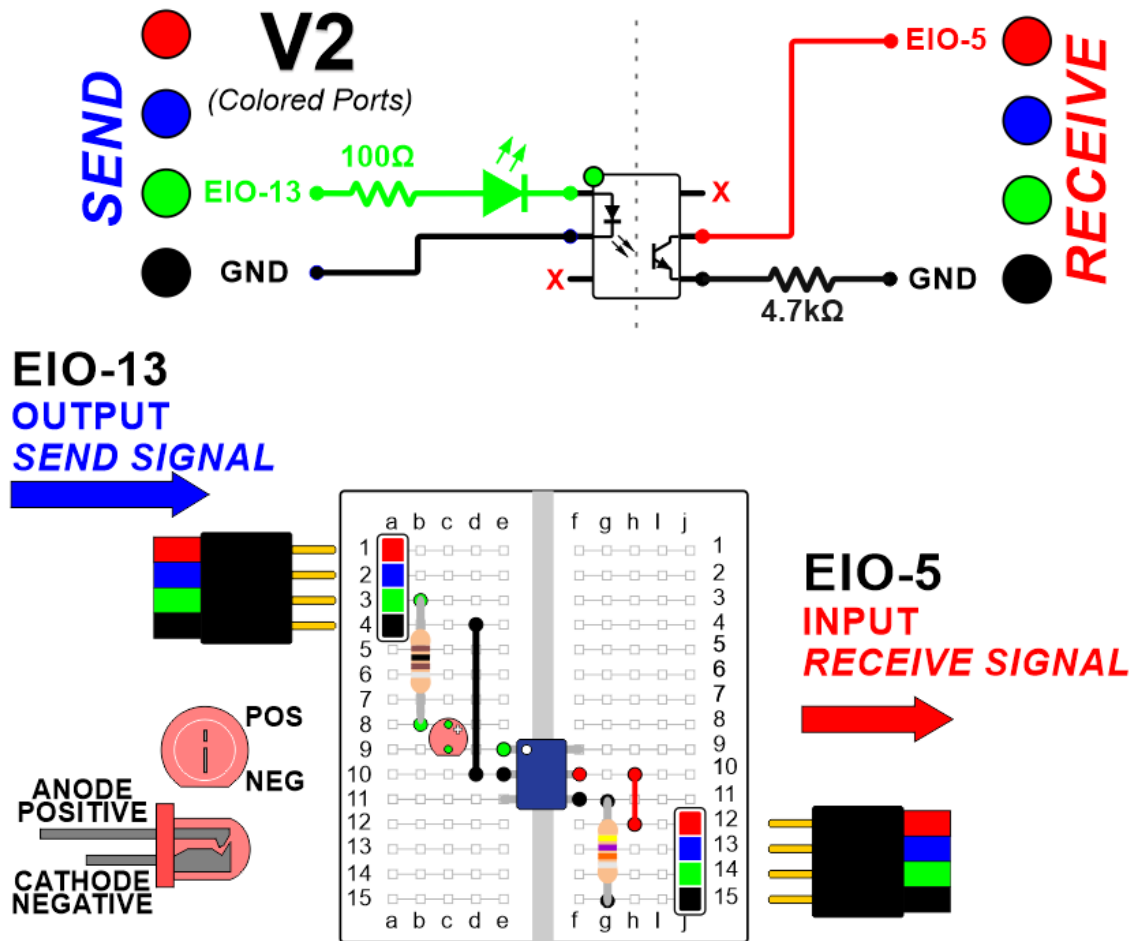


INPUT SIGNAL

Dobot Magician V2 (Colored IO Ports)



Dobot Magician V2 (Colored IO Ports)



Wire Robot #1 so that GP2 is wired to the input of the handshake module. Then wire the output of the handshake module to the Input of robot #2. This is labelled #3 on the arm. Please refer to the wiring diagram above. You can use servo extension cables to do the wiring.

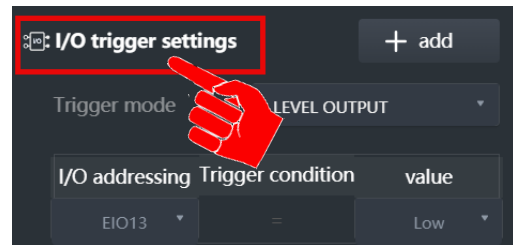


Notice that on the **INPUT** to the handshake module you will use the RED and GREEN wires to trigger the Optical Isolator. On the **OUTPUT** side you will use the RED and BLACK wires to trigger the input on the other robot. **These colors match the diagram above but may vary depending on the wires you use.**

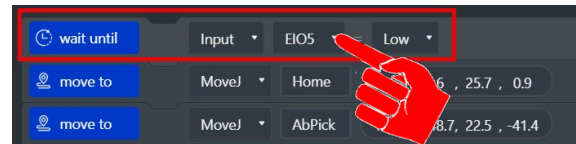
1. Set up the DobotLab program for robot # 1 so that it moves the part from its position into the work envelope of the second robot. It needs to send a signal to EIO13 for 1 second (high), then turn the signal off (low)



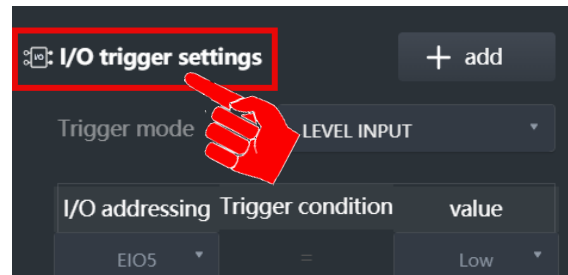
2. To do this, use the I/O Trigger Settings in the Command Panel as shown.
 - a. Set Trigger mode to **Level Output**
 - b. Set Address to **EIO13**
 - c. Set Value to **High or Low** as needed
 - d. Click on the **+add** button to insert into program



3. Set up the DobotLab program for robot # 2 so that it waits until robot #1 sends a signal to pick it up, then place it somewhere else. Be sure to use EIO5 as the trigger input, and set it at Low for 1 second, then turn it off in the next step.



1. To do this, use the I/O Trigger Settings in the Command Panel as shown.
 - a. Set Trigger mode to **Level Input**
 - b. Set Address to **EIO5**
 - c. Set **Value** to **Low** as needed
2. Click on the **+add** button to insert into program



Be sure to consult the Dobot Input/Output manual if you want to use other inputs and outputs, as damage to your robot or other equipment may result.

3. Write down any notes that you think you may need to make this work:



4. Once both programs are written, run the programs. If written correctly, robot #2 will wait for the signal from the first robot. If it does not work correctly, troubleshoot until it does.
5. *If your set up did not work correctly the first time, what did you have to do to make it work?*

CONCLUSION

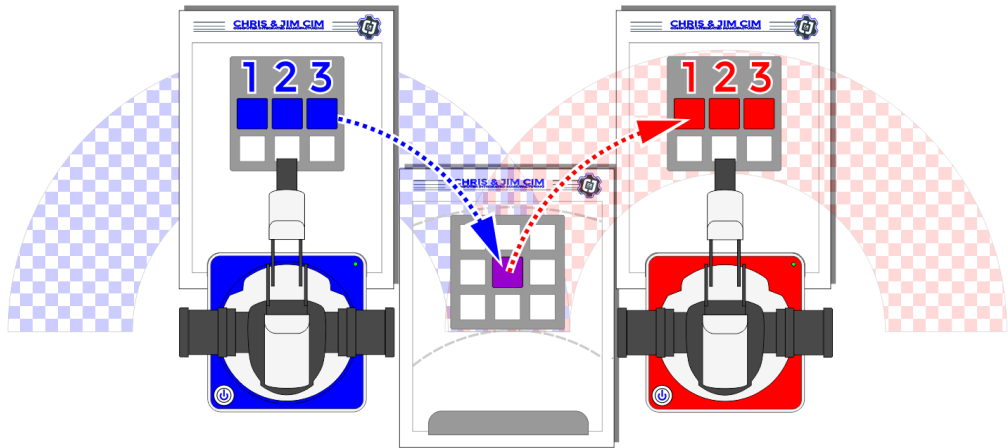
1. *What would you have to do to make this program run five times without any human intervention? Explain fully below.*
2. *What other inputs could you use on your robot to start this process? Use the Input/Output manual to answer this question, and do not attempt to try it without your instructor's permission.*
3. *What other outputs could you use on your robot to start this process? Use the Input/Output manual to answer this question, and do not attempt to try it without your instructor's permission.*



GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- _____ 1. Add an input switch to the first robot to start the whole process.
- _____ 2. Make the robots **palletize** three **cubes** as the diagram below shows. Start with three cubes on the first pallet, use the second pallet as a buffer to do the hand off, and then have the second robot place them on pallet 3. Be sure to adjust the **roll angle** accordingly.



2.2 Block - Handshaking: Dobot to Dobot

NAME: _____

Date: _____

Section: _____

INTRODUCTION

Robotic arms need to communicate with other robots in a work cell, or factory. This is called **HANDSHAKING** and can be done between different machines, devices, and robots. It is a very simple form of communication and is done with simple ones and zeros; or “ons” and “offs”.

In this activity you will learn how to make a robot handshake with another robot.

Robot 1 will pass a part into another robot’s work envelope, go to a safe position and then send a signal to other robot. The signal received by the other robot will initiate a sequence to get the part, and place it somewhere else.



Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS shutdown the Dobot before making connections or damage to the robot could occur.

KEY VOCABULARY

- Input
- Output
- Function / Voids
- Signal
- Handshaking

EQUIPMENT & SUPPLIES

- 2 Robot Magicians
- 1” cylinders or cubes
- Suction Cup Gripper
- DobotLab software
- [Dobot Input/Output Guide](#)
- Handshake Modules

ESSENTIAL QUESTIONS

Essential questions answered in this activity include:

- How do I make a robot send a signal?
- How do I get a robot to receive a signal?
- How is this done in Dobot Lab Software?
- How do I make two robots talk to one another?

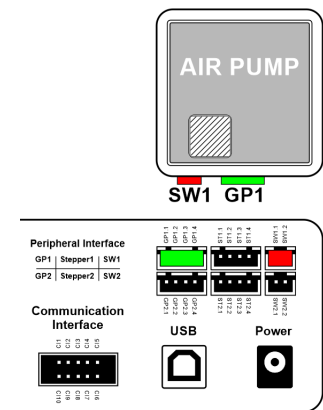
PROCEDURE



Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Set up both **Robots** with a suction cups - **GP1 & SW1** and place Dobot field diagrams, taped to the work surface, between the two robots.
2. Wire **Robot1** with an **OUTPUT** signal **GP2 - EIO13**.
3. Wire **Robot2** with an **INPUT** signal **GP5 - EIO5**.
4. Wire both Robots to the Handshaking module as shown in the following pages. Be sure that wires are not going to be pulled out by the motion of the robots.

Be sure that wires are not going to be pulled out by the motion of the robots.



Open Loop System Block Diagram: Robot #1 acts as the input for robot #2. Robot #2 does not communicate back to robot #1 so there is no feedback. The Handshake Module acts as the device that helps make the handshake happen safely.



Order of operations

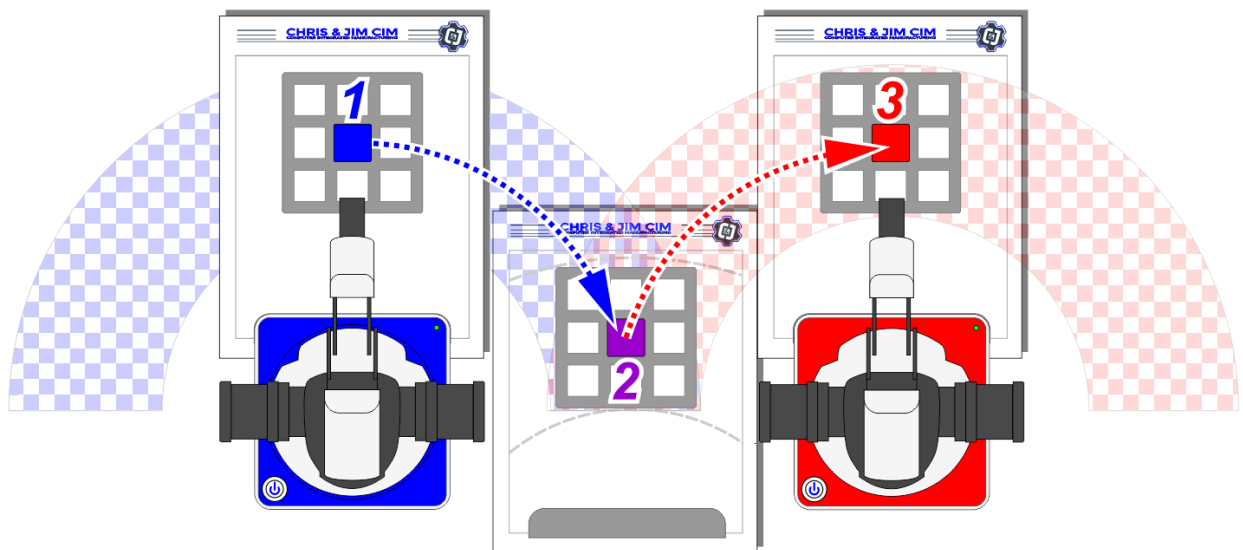
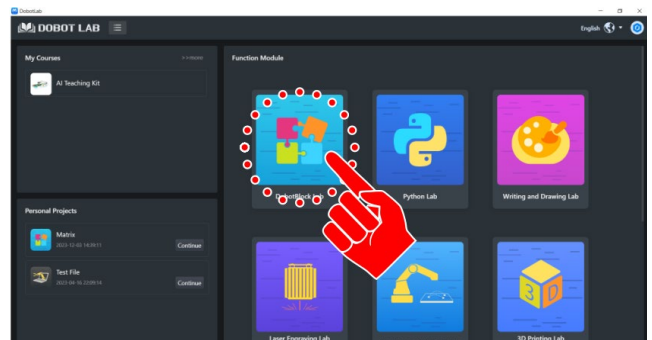
ROBOT 1 - OUTPUT SIGNAL

- Move - Home
- Move - ABPick
- Move - ATPick
- Jump - ATPlace
- Move - ABPlace
- Move - Home
- Send Output Signal

ROBOT 2 - INPUT SIGNAL

- Move - Home
- Wait for INPUT Signal
- Move - ABPick
- Move - ATPick
- Jump - ATPlace
- Move - ABPlace
- Move - Home

Open a new file for DobotBlock Lab



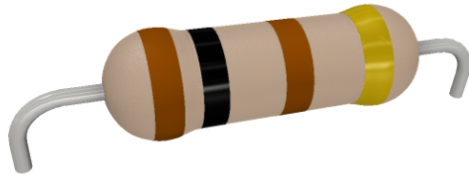
WARNING



Power OFF

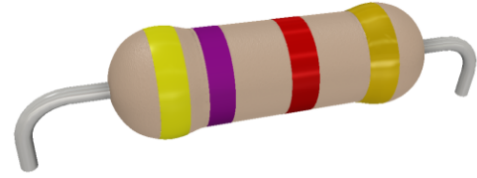
Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.





100 Ω - SEND SIDE

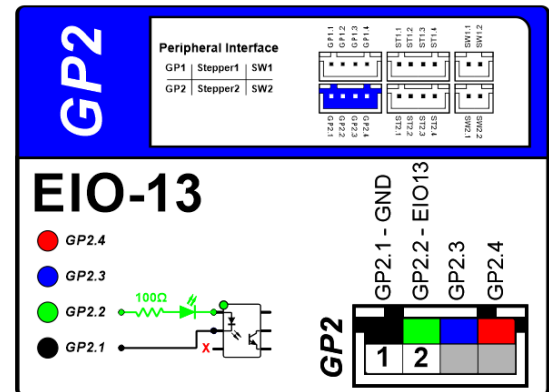
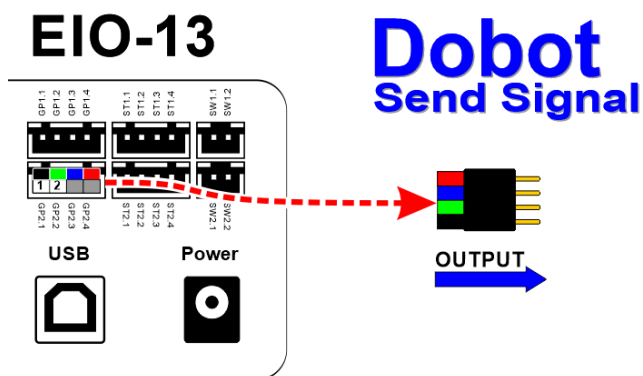
BROWN	BLACK	BROWN	GOLD
1	0	X10	5%



4.7k Ω - RECEIVE SIDE

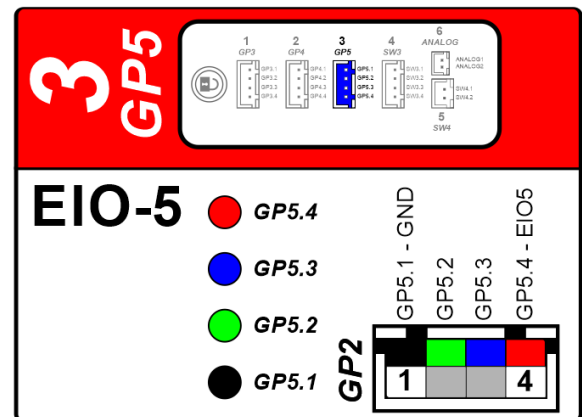
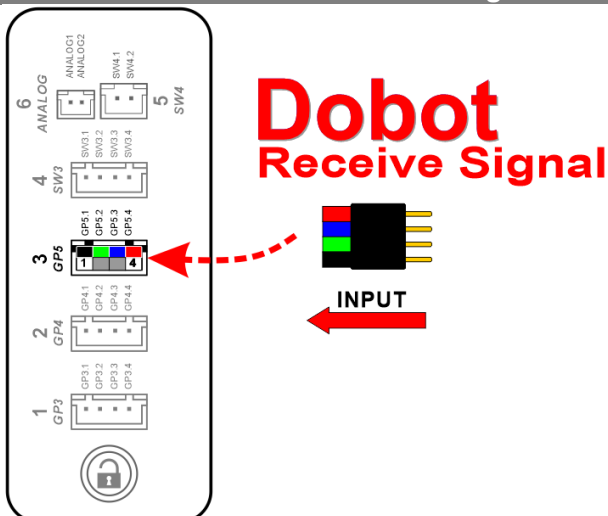
YELLOW	VIOLET	RED	GOLD
4	7	X100	5%

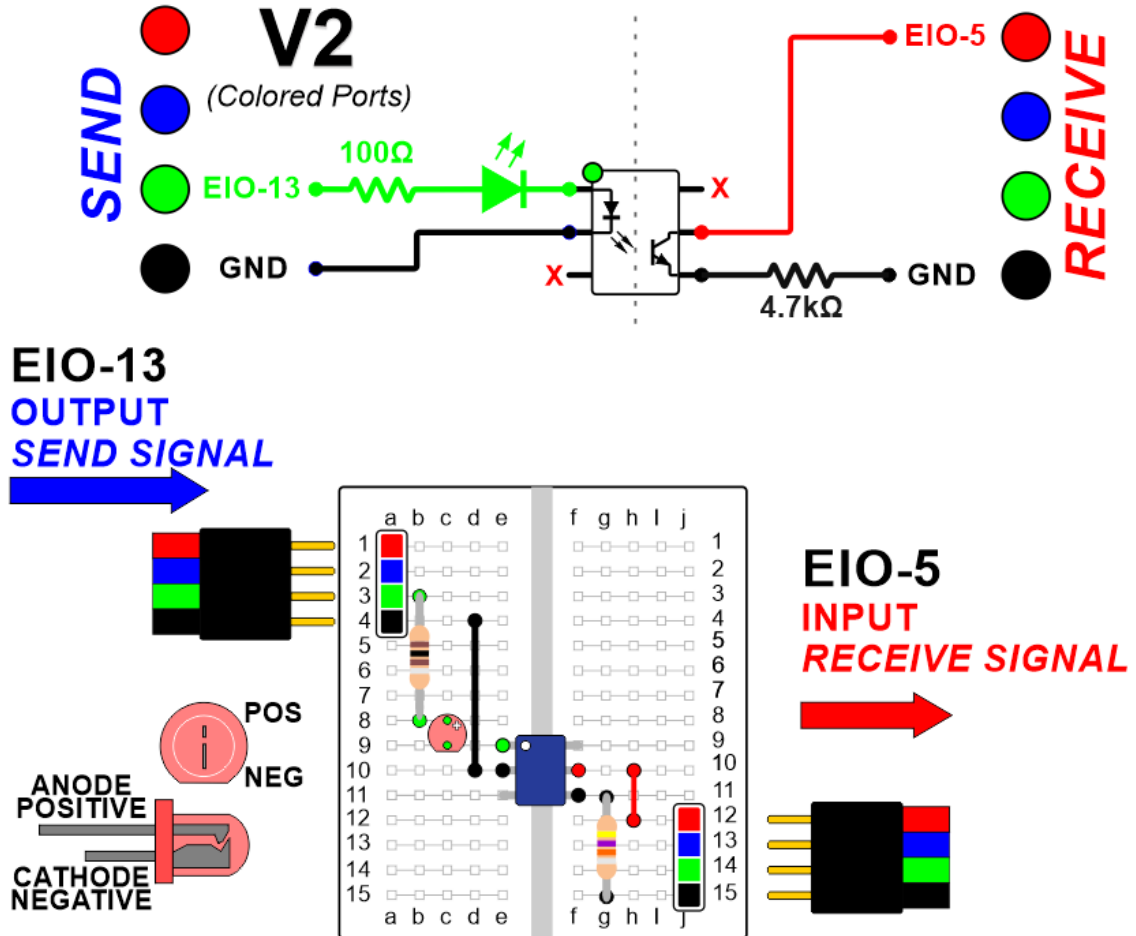
OUTPUT SIGNAL



INPUT SIGNAL

Dobot Magician V2 (Colored IO Ports)





Wire Robot #1 so that GP2 is wired to the input of the handshake module. Then wire the output of the handshake module to the Input of robot #2. This is labelled #3 on the arm. Please refer to the wiring diagram above. You can use servo extension cables to do the wiring.



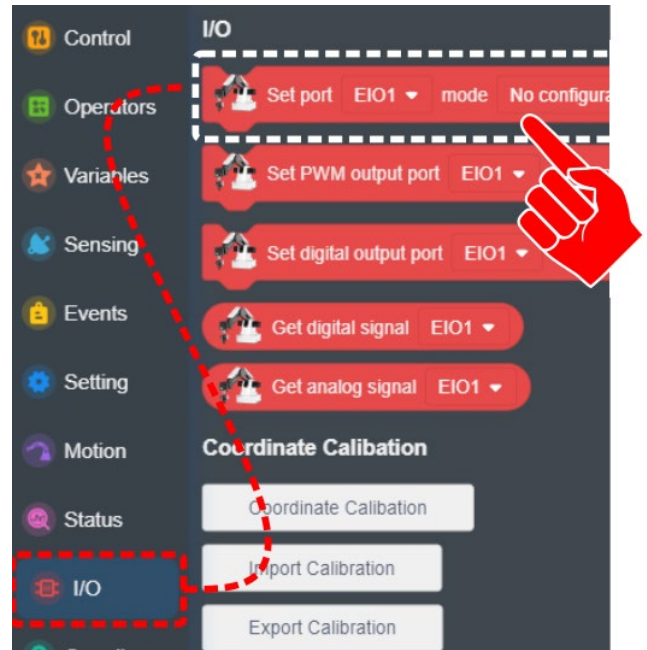
Notice that on the **INPUT** to the handshake module you will use the RED and GREEN wires to trigger the Optical Isolator. On the **OUTPUT** side you will use the RED and BLACK wires to trigger the input on the other robot. These colors match the diagram above but may vary depending on the wires you use.



LEARNING TO SEND A SIGNAL

Before we start this activity we need to learn how to send a signal (SEND - DIGITAL OUTPUT) using BLOCKLY. We will be adding this to the skills learned from the INPUT (RECEIVE _ DIGITAL SIGNAL) activity.

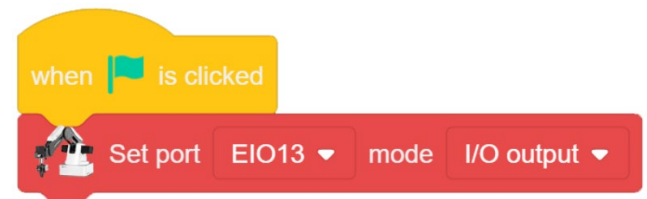
Drag over a SET PORT from the I/O Toolbar and attach it to a when started



Change the settings:

PORT: EIO13

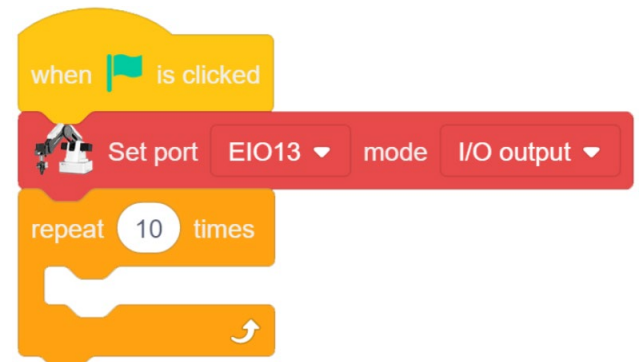
MODE: I/O OUTPUT



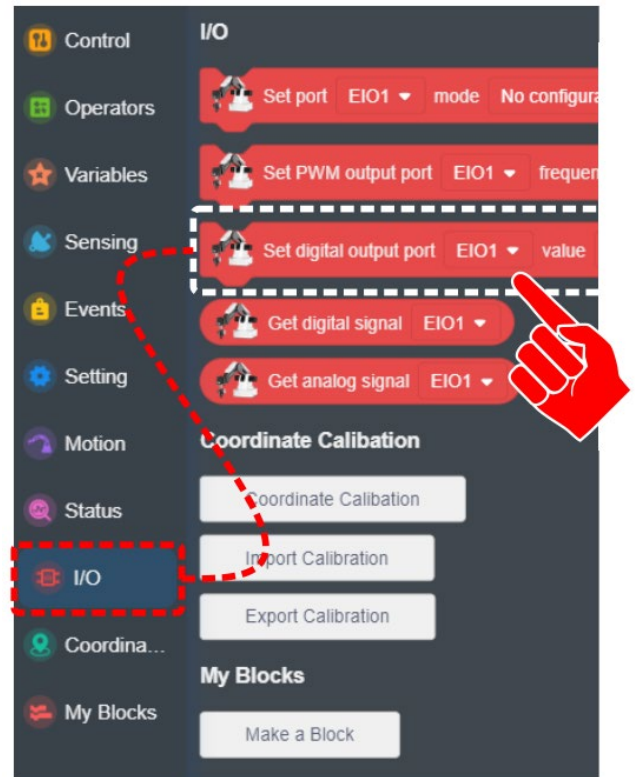
In order to test the OUTPUT signal, we will create a simple program that loops the process of turning on and off the output.

Drag over the **REPEAT LOOP** block

10 times should be enough to ensure our setup is working correctly so make it repeat this many times.



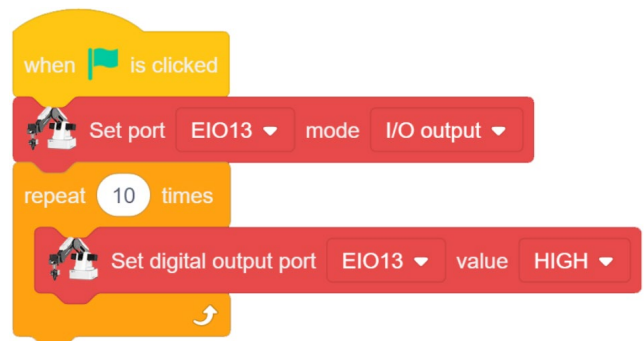
Drag over the **SET DIGITAL OUTPUT** block from the I/O Toolbox.



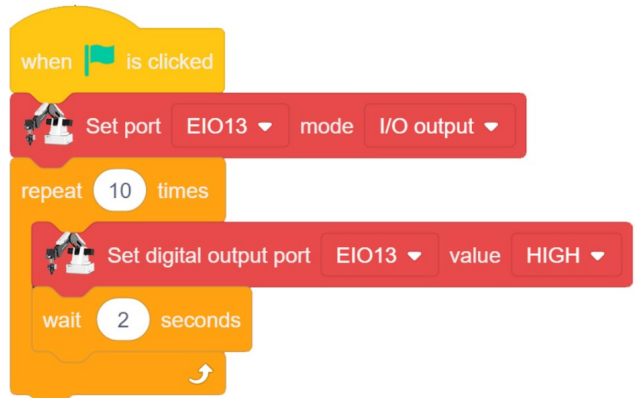
Add the block inside the REPEAT LOOP

Set the **PORT** to **EIO13**

Set **VALUE** to **HIGH = ON**



Insert a **WAIT** under the **SET DIGITAL OUTPUT** block. This will allow us to send the signal for two seconds.



Duplicate both steps

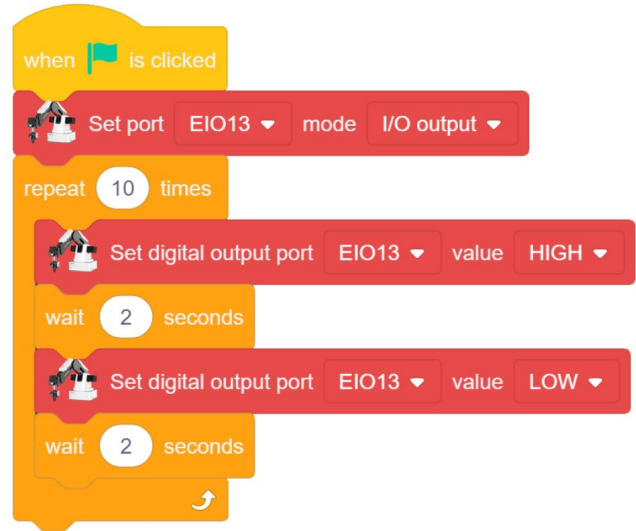
Set **VALUE** to **LOW (OFF)**

This mini program should allow us to send a signal to the Handshake module 10 times (2 seconds on, 2 seconds off x 10)



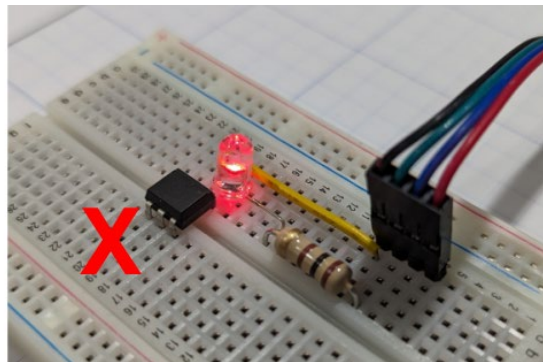
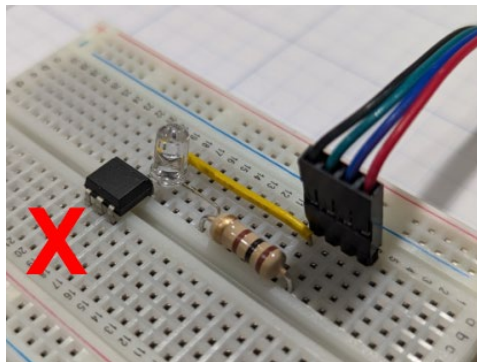
WHY IS IT NOT WORKING?

- LED Backwards?
- Not using the correct pins of the Optical Isolator?
- Plugged into the incorrect PORT (EIO13)?
- 4-Wire plugged in incorrectly (Breadboard)?
- 4-Wire plugged in incorrectly (Robot)?
- Jumper wire incorrect?
- Blockly program – did not change to EIO13 Throughout?
- Bad LED?
- Bad Optical Isolator? (there is an LED inside it as well)



TROUBLESHOOTING THE OPTICAL ISOLATOR

Once this portion of the program is completed, run it and see if it works correctly. The LED on the handshake module should light up whenever EIO13 is enabled ON. If it does not work, troubleshoot it until it does. If the LED is coming ON, the SEND side of the Isolator is working PERFECT!



If your set up did not work correctly the first time, what did you have to do to make it work?



Programming for ROBOT 1 – OUTPUT – SEND SIGNAL

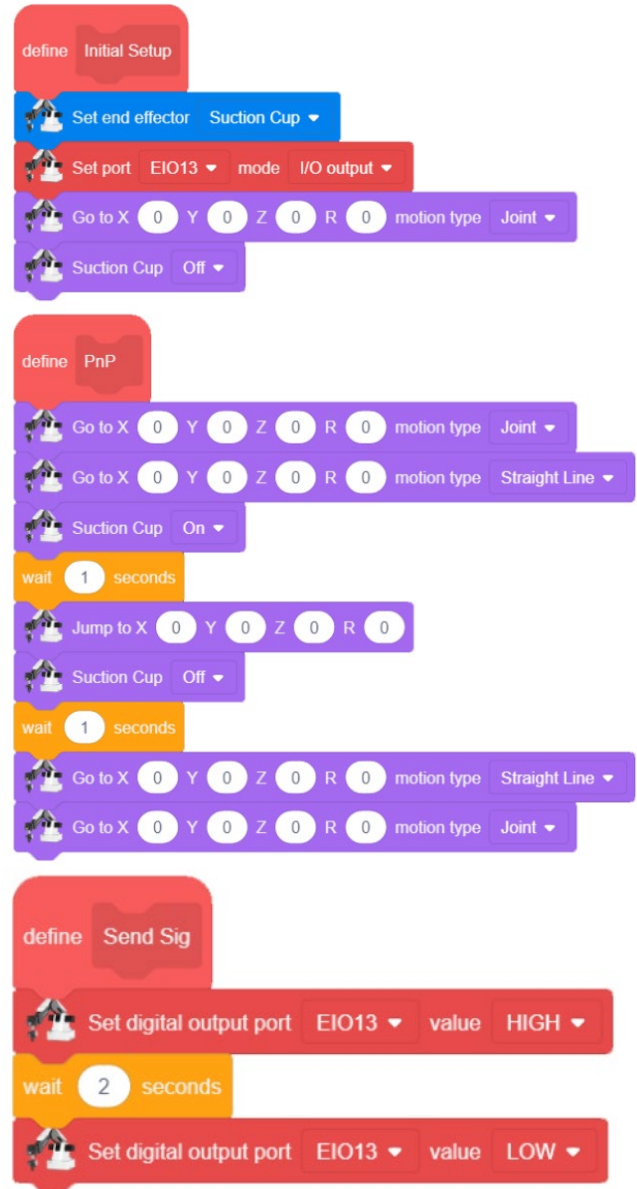
Throughout the remainder of this activity, we will use many of the skills that you have developed in previous activities. If you need clarification at any point, you can go back to the other activities for more information.

As we have done in the past:

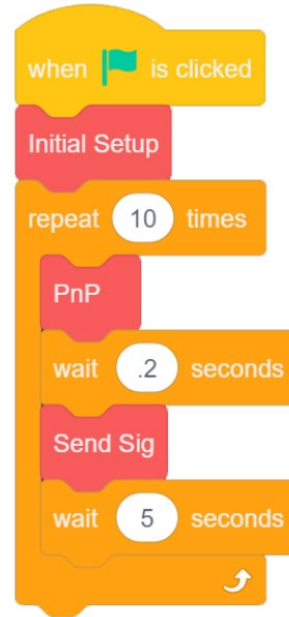
1. Create two **MY BLOCKS**
2. Fill in Positions (Right Click FILL COORDINATES)

- Initial Setup
 - Set EoAT
 - Set Digital Inputs and Outputs
 - Send Robot to Home Position
 - Open or turn off EoAT
- PnP
 - Go to Above Pick JOINT
 - Go to At Pick Linear
 - Turn ON EoAT
 - Wait 1 second (allow EoAT to Charge)
 - Jump to Place
 - Turn OFF EoAT
 - Wait 1 second (Allow EoAT to Release)
 - Go to Above Place
 - Return Home

3. Create a new **MY BLOCK** – Send Sig



4. Put the program together.



Programming for ROBOT 2 – INPUT – RECEIVE SIGNAL



Two robots can be plugged into the same computer at the same time!

Select choose a device and add a second Magician.

Connect to this device and home it just as we have done with previous activities.

As you click back and forth between the magician devices (icons), you will see a different programming page, one for each. When you save the program, you will save both programs at the same time.



If TWO separate programs were already written, TWO versions of DobotLab can be opened and run at the same time as well.



Throughout the remainder of this activity, we will make assumptions that you have already learned several concepts from previous activities. If you have any questions, check your work from the other activities before you ask your instructor.

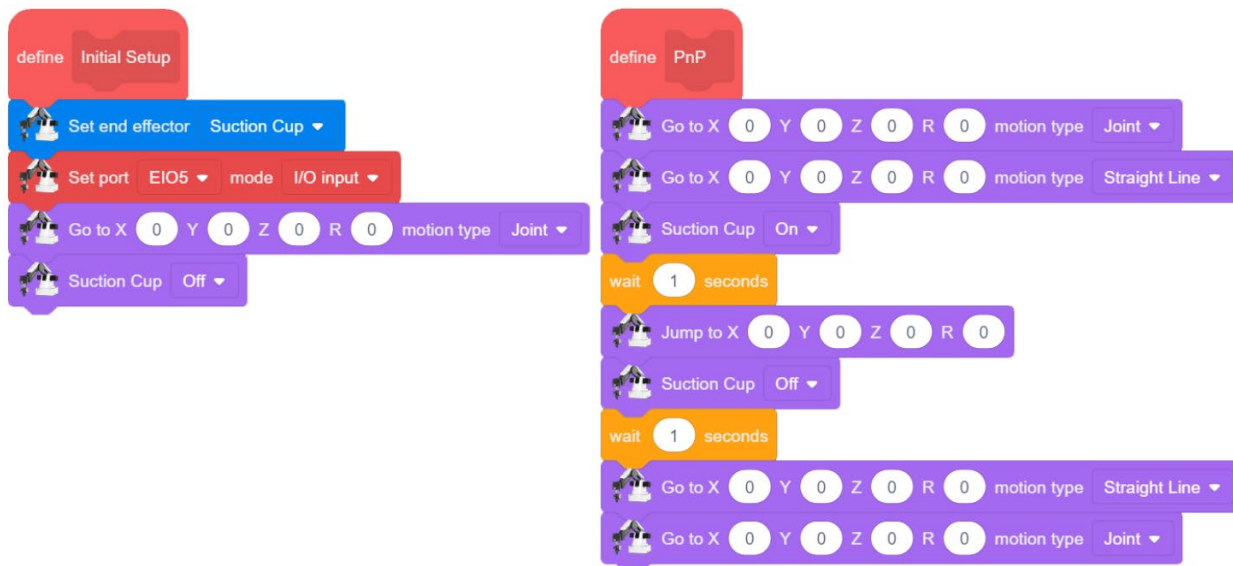




WARNING

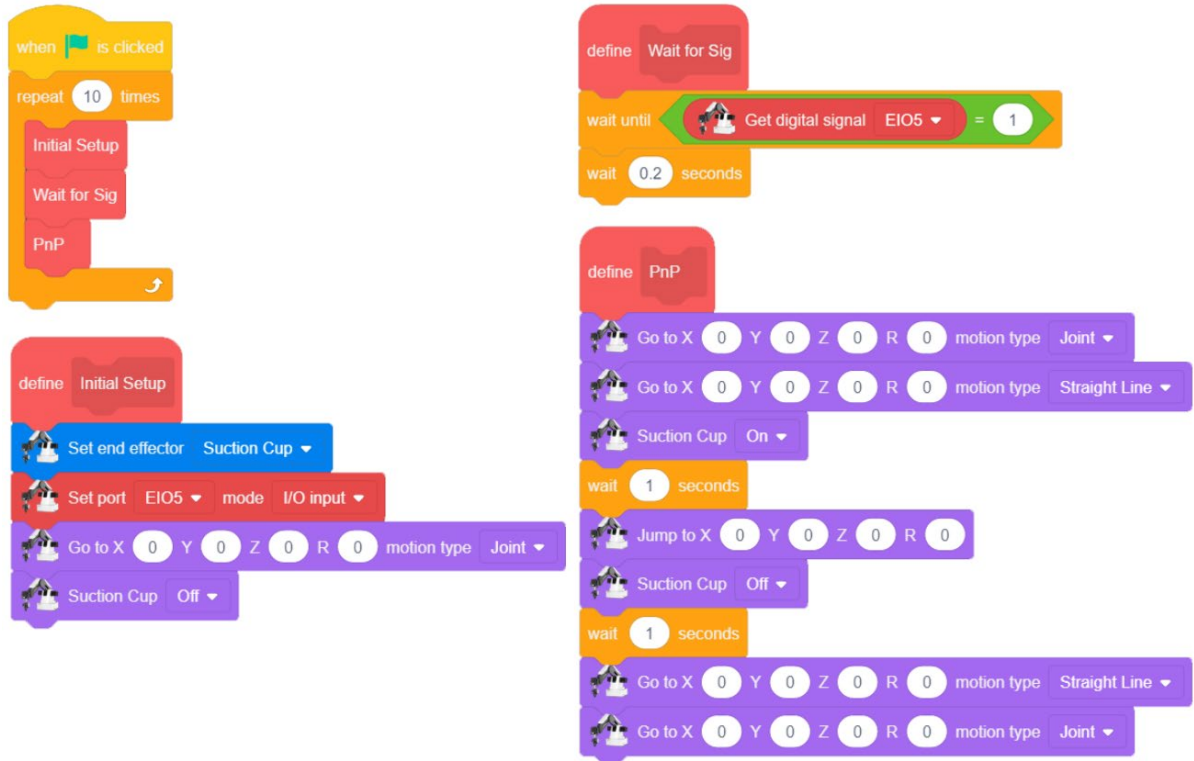
Until everything is working and tested, do not plug this robot into the handshake module. Use a manual input switch for testing the signal instead.

Create an Initial Setup and PnP Block as we have done in the past



Put the different parts of the program together.

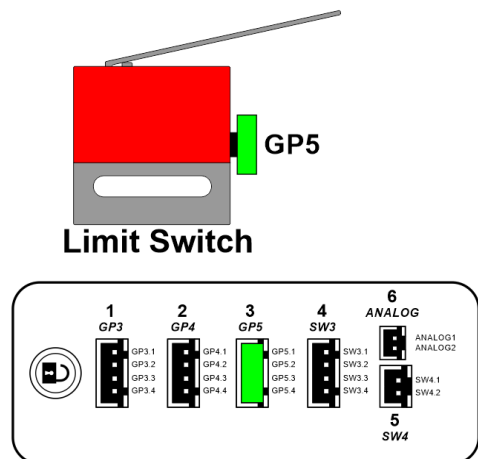




USE A LIMIT SWITCH HARDWIRED TO THE ARM FOR TESTING THIS PROGRAM

Once this portion of the program is completed, run it and see if it works correctly. If it does not work, troubleshoot it until it does.

When you press the switch, what happens?



If your set up did not work correctly the first time, what did you have to do to make it work?



Final Project Wiring



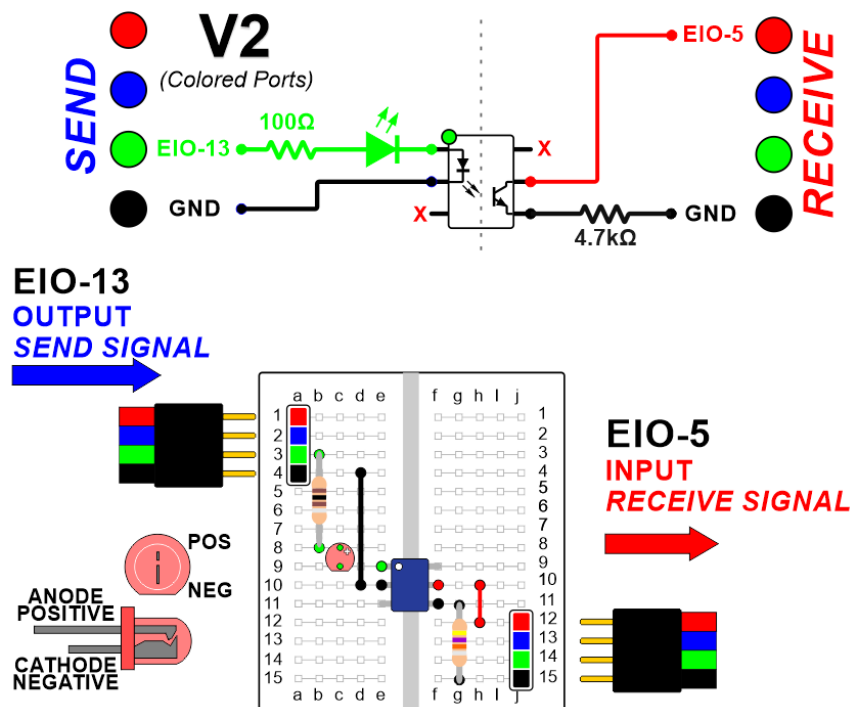
WARNING



Power OFF

Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

We can now plug in the wire from the breadboard to the Input port of the second robot.



Once the programs are completed, run them together and see if it works correctly. If it does not work, troubleshoot it until it does.

If your set up did not work correctly the first time, what did you have to do to make it work?

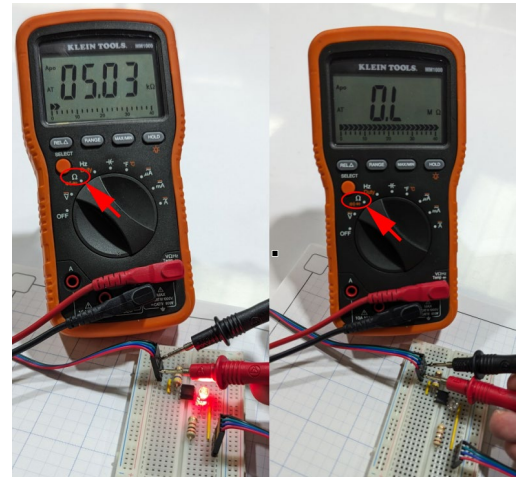
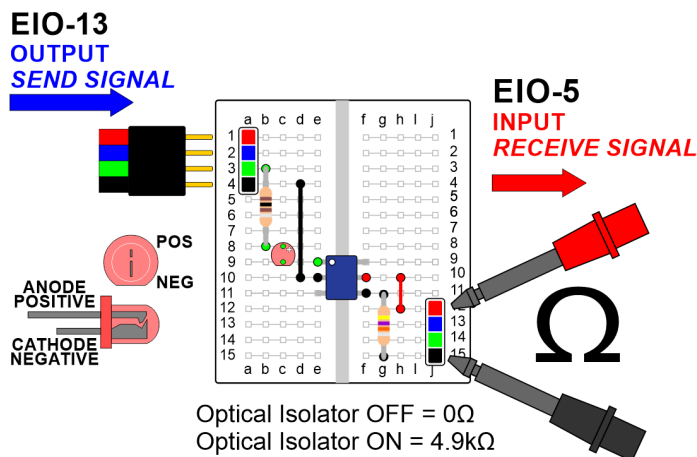


TROUBLESHOOTING THE OPTICAL ISOLATOR

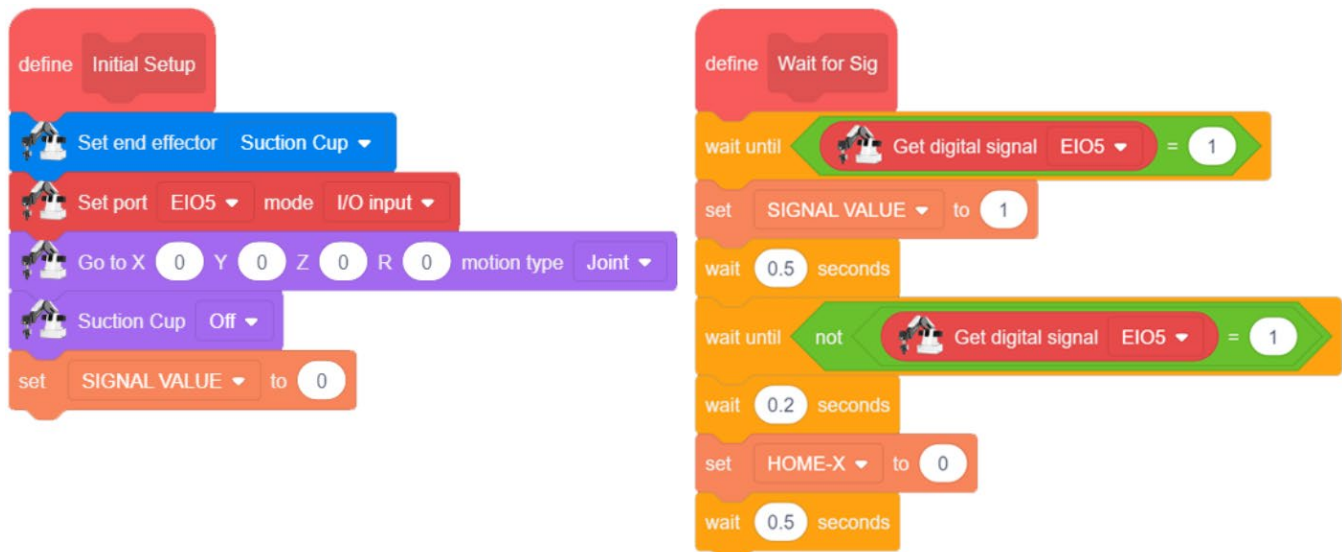


The LED helps us determine if the first robot is actually sending the signal. We can write a simple Blockly program to see if the second robot is getting the signal. The issue is if the second robot is not getting the signal. WHY is it not getting the signal? Check the previous helpful tip.

An easy way to check is to use a voltage meter set to check RESISTANCE. When no signal is present, the voltmeter should read zero. When a signal is present, it should read a little over the resistance value of the resistor used.



Consider adding Variable "SIGNAL VALUE" to see what the inputs and outputs should be reading at any given time.



CONCLUSION

1. What would you have to do to make this program run five times without any human intervention? Explain fully below.
2. What other inputs could you use on your robot to start this process? Use the [Dobot I/O Guide](#) to answer this question, and do not attempt to try it without your instructor's permission.
3. What other outputs could you use on your robot to start this process? Use the [Dobot I/O Guide](#) to answer this question, and do not attempt to try it without your instructor's permissions

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

1. Turn your program into a CLOSED LOOP. Add the ability to return the signal.
 - Robot 1 retrieves a block but waits if a signal is being sent from robot 2 (robot BUSY) before sending the pickup signal.
 - When no signal is present (WAITING), Robot 1 sends a signal for Robot 2 to receive.
 - Robot 2 waits for a signal before picking up the block. When the pickup signal is received, Robot 2 sends a busy signal to Robot 1. Robot 2 keeps sending the signal until it is ready for another block. ***Be sure to consult the [Dobot Input/Output Guide](#) if you want to use other inputs and outputs, as damage to your robot or your other equipment may result.***



2.3 Block - Handshaking: Dobot to Dobot “NO” Relay Module

NAME: _____

Date: _____ Section: _____

INTRODUCTION

Robotic arms need to communicate with other robots in a work cell, or factory. This is called **HANDSHAKING** and can be done between different machines, devices, and robots. It is a very simple form of communication and is done with simple ones and zeros; or “ons” and “offs”.

This activity has already been completed using CIRCUITRY and HARDWARE. The 4N25 optical isolator (relay) and is an excellent way to have two robots safely handshake. With DobotBlock Lab, we can safely have the robots communicate with no wiring, or other hardware (internal communication).

Two or more robots will still be communicating with YES and NO signals called **messages**, this time it will be internally controlled by the computer as if they were networked using software. This too is an industry standard for equipment to be able to communicate at the lowest level (1s and 0s).



No more handshaking with hardware that has to be built and wired. DobotBlock Lab software can create internal robot to robot communication/handshaking.

KEY VOCABULARY

- Input
- Output
- Signal
- Network Handshaking

EQUIPMENT & SUPPLIES

- 2 Robot Magicians
- 1” cylinders or cubes
- Suction Cup Grippers
- DobotLab software
-
-

ESSENTIAL QUESTIONS

Essential questions answered in this activity include:

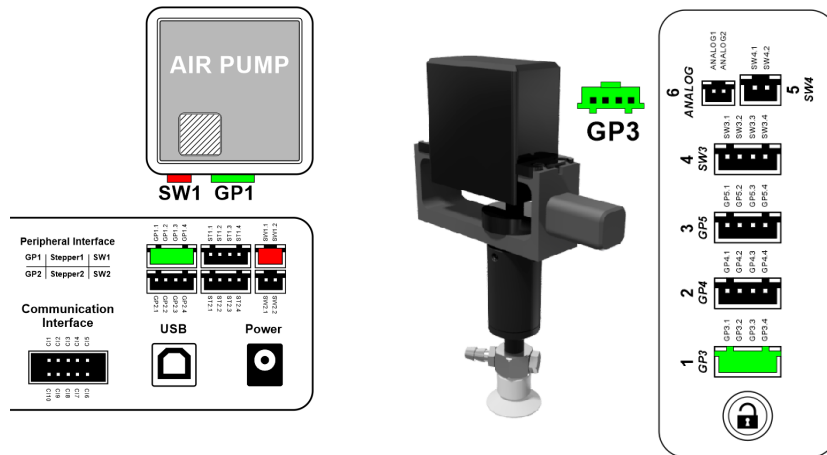
- How do I make a robot send an internal signal?
- How do I get a robot to receive an internal signal?

SECTION 1 – SETUP BOTH ROBOTS



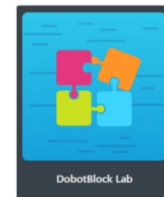
Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Set up the robots with a **SUCTION CUPS** and Air Pumps.

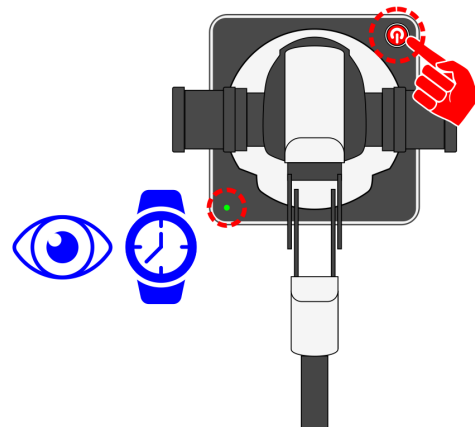


SECTION 2 – CONNECTING ROBOTS to the COMPUTER & DOBOTLAB

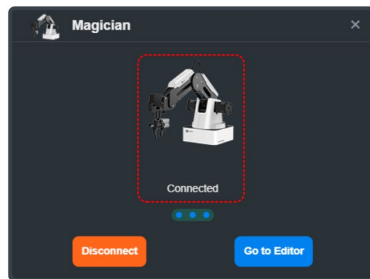
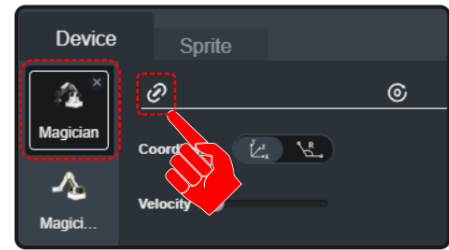
1. Connect both robots to the computer using their included USB cables
2. Open up DobotBlock Lab in the software.



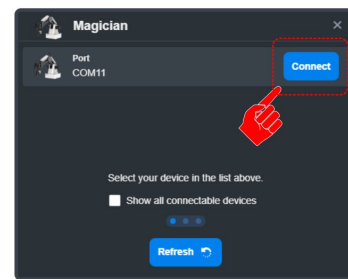
3. Power on BOTH robots and wait for them to completely boot up.(green lights)



4. Select the connect icon to establish control from the software to ROBOT 1. A connection window should pop up.
5. Select Connect from the pop up window. A new window should appear and show that the robot is Connected.



6. HOME ROBOT 1 to ensure communication.



Two robots can be plugged into the same computer at the same time!

7. Select choose a device and add a second Magician.

Connect to this device and home it just as we have done with previous activities.

As you click back and forth between the magician devices (icons), you will see a different programming page, one for each. When you save the program, you will save both programs at the same time.

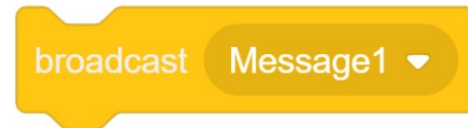
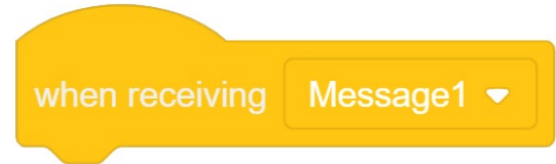


SKILL BUILDER 3 – BROADCAST - INTERNAL HANDSHAKING BETWEEN ROBOTS

The process of sending a signal from one robot to another robot can be done with messages.

1. Drag over WHEN RECEIVING and BROADCAST black from the EVENTS toolbox.

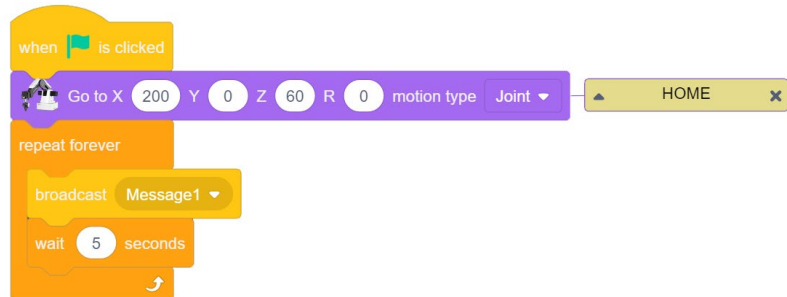
The WHEN RECEIVING works the same as the WHEN STARTED, but instead of selecting the START FLAG to be clicked, the program waits for a message to be received.



The BROADCAST block is the tool that initiates the message. Both blocks also have the option to create new messages in their drop downs.

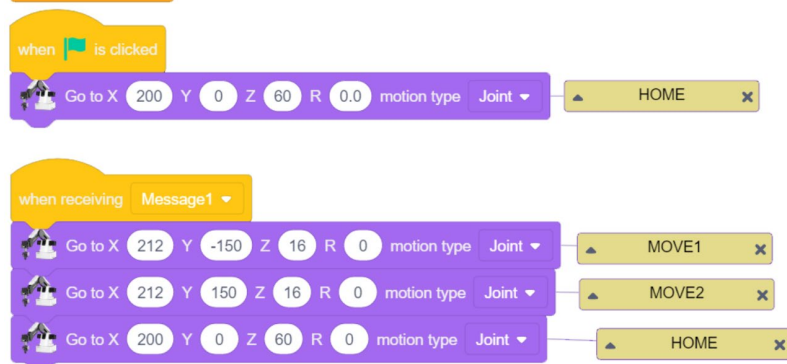
1. Create a program similar to the one on the right for ROBOT 1

- a) WHEN STARTED – send ROBOT1 HOME
- b) BROADCAST MESSAGE
- c) WAIT 5 seconds and repeat



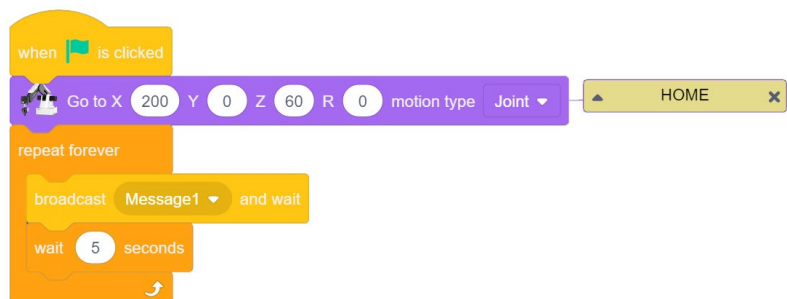
2. Create a program similar to the one on the right for ROBOT 2

- a) WHEN STARTED – send ROBOT2 HOME
- b) WAIT a message
- c) Once RECEIVED, move ROBOT2



The program created is an OPEN LOOP system. ROBOT1 does not know ROBOT2 has received the signal.

3. Change the BROADCAST MESSAGE to BROADCAST and WAIT. ROBOT1 will now send the signal and wait until the broadcast is received by ROBOT2



Recreate the CLOSED LOOP process using VARIABLES

ROBOT 1

ROBOT 2



Create a VARIABLE called SIG RECEIVED ROBOT2 and alter the two programs

Run the program



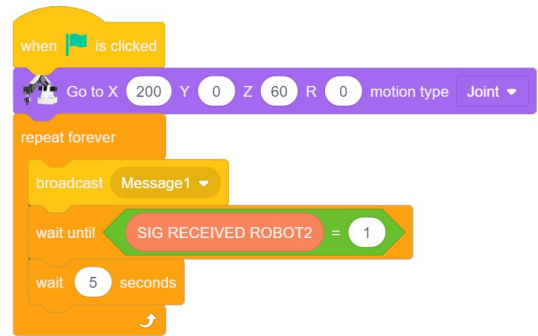
Is the program really working correctly? It may look like it is working, but once the variable is set to TRUE... it never changes. The program is now skipping the WAIT UNTIL.

4. Alter ROBOT1's program to change the variable back to 0 once it is received.
5. Change the time to 10 seconds to make sure it is working correctly.

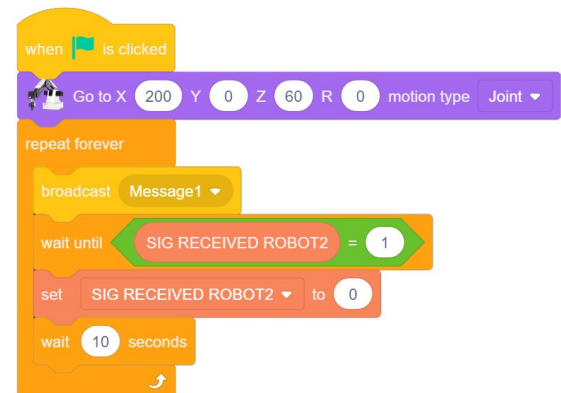
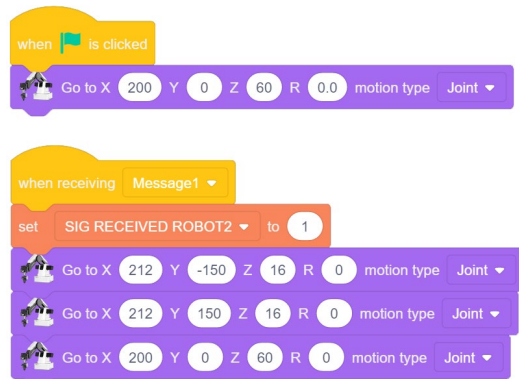
Run the program.



ROBOT 1



ROBOT 2



These tools are great for one way communication, but the trick is how to wait for a message in the middle of a program rather than just using it to start the ENTIRE PROCESS... Is there a better way to do this?



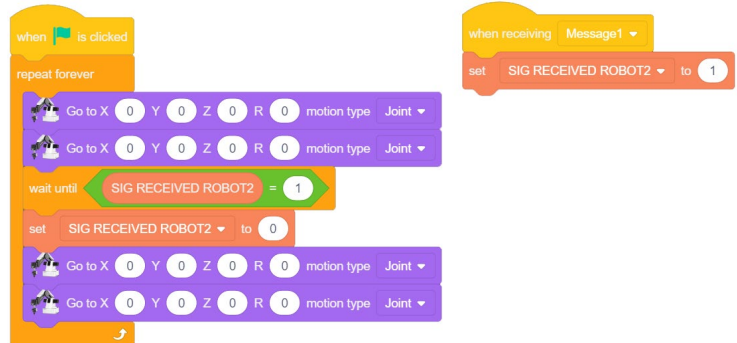


...Notice that there is only one list of variables for the SPRITE

...NOTICE that variables are not only GLOBAL for one robot's program but for all of them!!
(each robot can SET AND USE the same variables)

If the program can use WHEN MESSAGE RECEIVED blocks to control variables... we can then handshake in the middle of a program.

The program can use the variable in the middle of the program to tell the robot when to proceed. *(each time the message is received, set a variable to tell the program the message has been received)*



The WAIT UNTIL with the SIG RECEIVED block is a great way to make this happen.



If this is the case... can the program JUST USE VARIABLES instead of broadcasting messages?

YES!!!! BROADCAST MESSAGE has its place in programing, especially when is can be used to replace a WHEN STARTED for a second program.

SKILL BUILDER 4 – VARIABLES ONLY - INTERNAL HANDSHAKING BETWEEN ROBOTS

For the remainder of this activity, the BROADCAST MESSAGES will NOT be used.

6. Create an additional variable
 - a) SEND SIG **ROBOT1**
 - b) SIG RECEIVED **ROBOT2**

ROBOT1 will use SET variable SEND SIG to TRUE(1) to start **ROBOT2**.

ROBOT2 will use SET variable SIG RECEIVED to TRUE (1) to let **ROBOT1** know the signal has been processed.

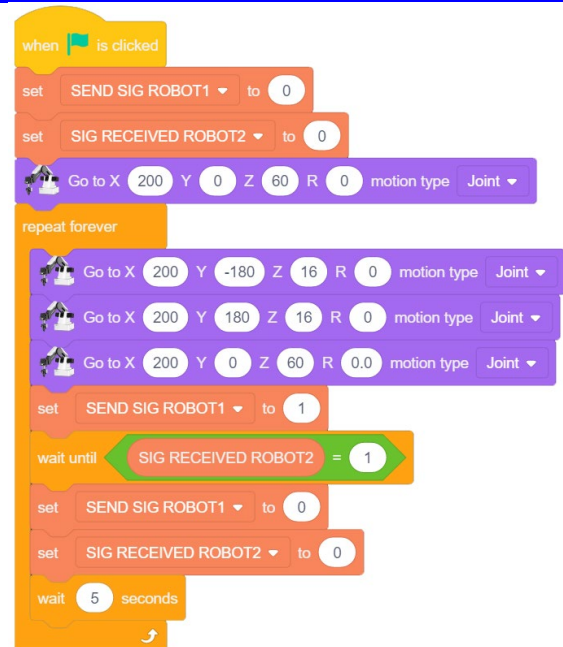
The variables will be SET to FALSE(0) to reset them for the next loop

For this activity, integers will be used rather than strings (words), however, the blockly program can use both.



OPERATION – ROBOT 1 (CONTROL ROBOT)

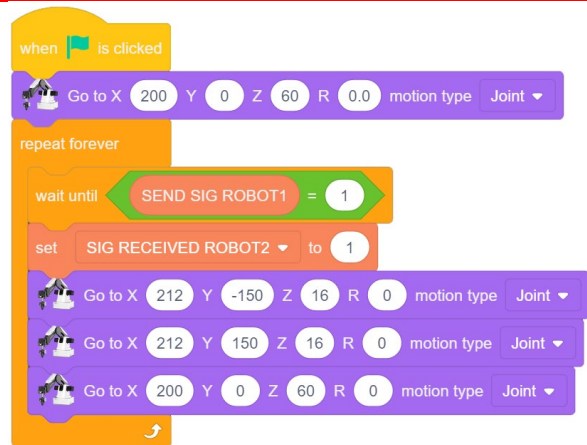
- Initialize variables to ZERO (only one robot needs to do this, they share the same variables).
- GO TO HOME
START THE FOREVER LOOP
- Move the robot to location “A” any location
- Move the robot to location “B” any location
- GO TO HOME
- SEND the signal to start **ROBOT2**
- WAIT until **ROBOT2** has received the signal
- Turn off the SEND signal and the RECEIVE signal
- Wait 5 seconds
- Loop the process



OPERATION – ROBOT 2

- GO TO HOME
START THE FOREVER LOOP
- SEND a signal to **ROBOT1** that the sequence has been started (**ROBOT1** will reset the variables)
- Move the robot to location “A” any location
- Move the robot to location “B” any location
- GO TO HOME
- Loop the Process

Both programs will start with the same
WHEN CLICKED



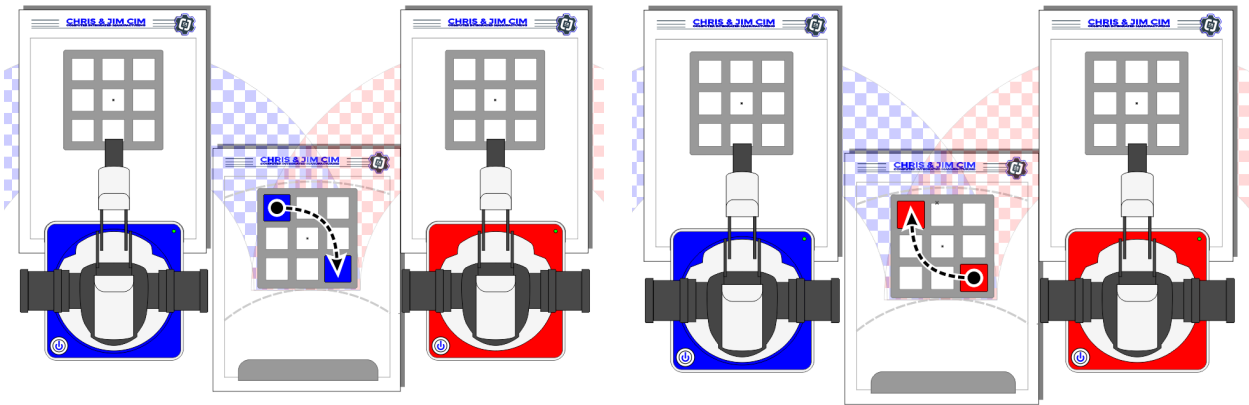
CHALLENGE

SECTION 5 – PASS the BLOCK (HOT POTATO)



HOME both robots before placing them on the templates. NOTICE there is an overlap in their work envelopes. The robots can run into each other during the homing operation.

Order of operations



ROBOT 1 -

- Move - Home
- Move - ABPick
- Move - ATPick
- Jump - ATPlace
- Move - ABPlace
- Move - HOME
- Send START SIGNAL to **ROBOT2**
- WAIT for START SIGNAL from **ROBOT2**
- Turn OFF Signal from **ROBOT2**
- **LOOP

ROBOT 2 -

- Move - HOME
- WAIT for START SIGNAL from **ROBOT1**
- Turn OFF Signal from **ROBOT1**
- Put the cube back - Move - ABPick
- Move - ATPick
- Jump - ATPlace
- Move - ABPlace
- Move - HOME
- Send START SIGNAL to **ROBOT2**
- **LOOP

If your set up did not work correctly the first time, what did you have to do to make it work?



CONCLUSION

1. *Explain the similarities in your own words between handshaking with the optical isolator and using internal signals. Use at least three bullet points.*
2. *Explain the differences in your own words between handshaking with the optical isolator and using internal signals. Use at least three bullet points.*
3. *Which way do you prefer? Why? Use at least three bullet points.*
4. *If you had to incorporate sensors into this activity, which ones would you use, and what would you use them for?*

GOING BEYOND

- _____
1. Use the two robots to perform a stacking operation to stack two or more blocks.
 - a. Robot 1 picks and places a block from the first pallet to the second.
 - b. Robot 1 goes home and sends a signal to robot 2.
 - c. Robot 2 waits for the signal, then picks up a different block on the pallet.
 - d. Robot 2 stacks the block on the other one.
 - e. Robot 2 goes home.
 - f. Repeat this multiple times.
 - g. How many can you get them to stack successfully?



2.4 TP - Handshaking: Dobot to Microcontroller

NAME: _____

Date: _____

Section: _____

INTRODUCTION

Robotic arms sometimes need to communicate with other machines in a workcell, or factory. This is called **HANDSHAKING** and can be done between different machines, devices, and robots. It is a very simple form of communication and is done with simple ones and zeros; or “ons” and “offs”.

In this activity you will learn how to make a robot handshake with a simulated machine. We will use a simple example: a grinder. You will make the robot talk to the grinder to turn it on during an automated grinding operation. Remember, this is CIM, we do not want to keep the grinder running all the time; we need to conserve energy, time, and resources, so the grinder needs to stop, and the robot will then replace the part.



Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur.

KEY VOCABULARY

- Handshake
- Open Loop
- Microwait
- Optical Isolator
- Closed Loop

EQUIPMENT & SUPPLIES

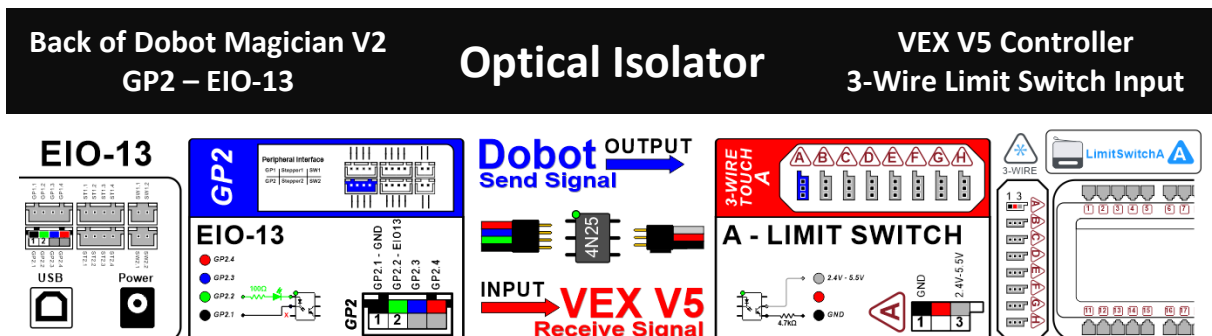
- Robot Magician
- Dobot Field Diagram
- 1" x ¾" cylinders or ¾" cubes
- Servo extension cables
- Dobot Input/Output Manual
- Breadboard/wire/4N25 Optical Isolator and 100 ohm & 4.7K ohm resistors
- VEX V5, PIC, Arduino may all be used in this activity, but wiring may vary.
- DobotLab software
- VEXcode V5 control software
- Pneumatic Gripper or Suction Cup Gripper
- Handshake module
- Normally open digital switch
- VEX parts: various for building the grinder including a baseplate, motor & controller.

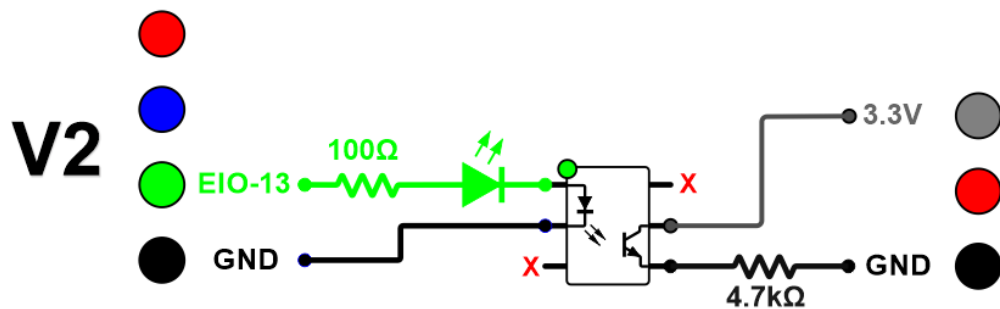
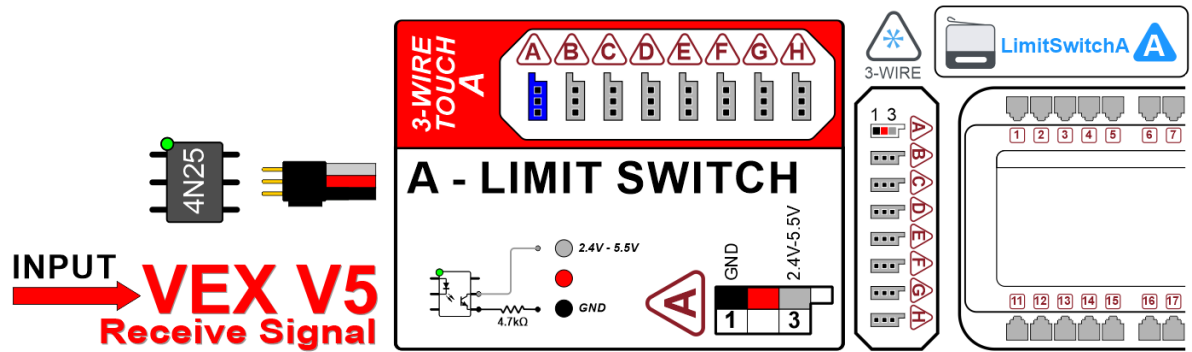
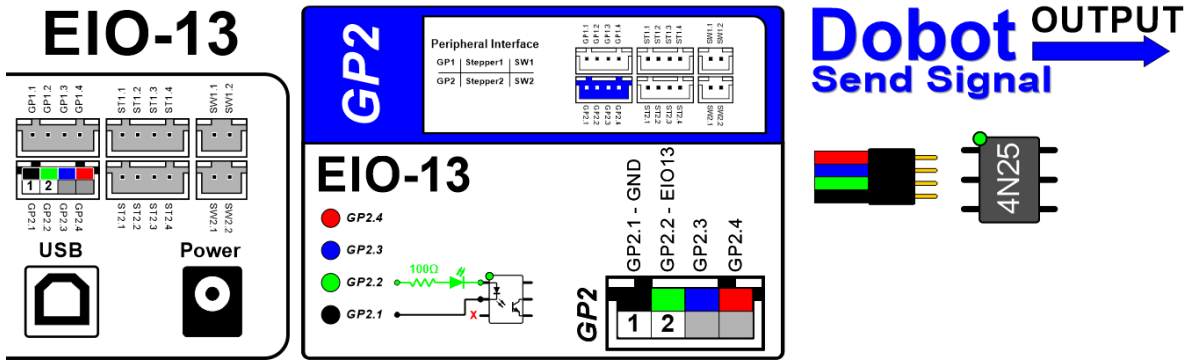
PROCEDURE



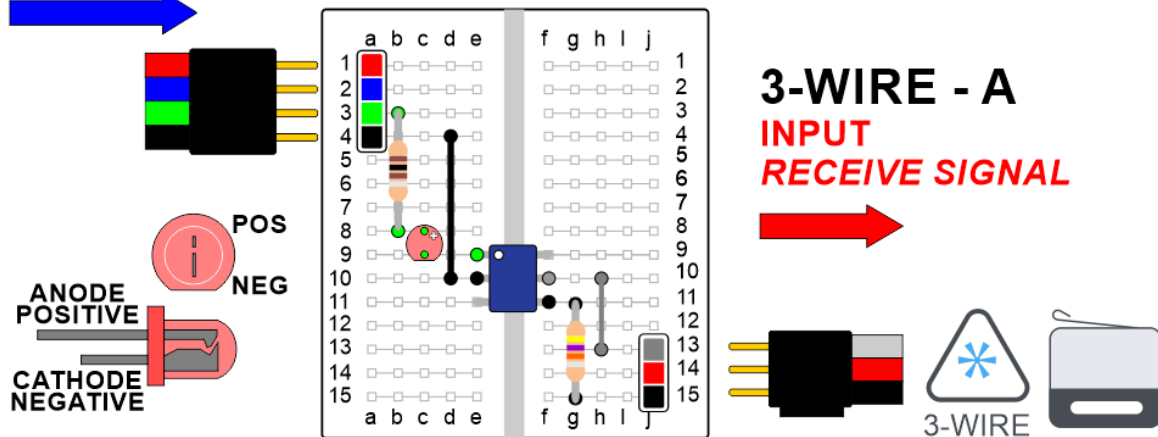
Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Wire the robot to the VEX V5 Robot Brain through the handshake controller as shown below. Any other microcontroller, such as an Arduino a PIC or a Raspberry Pi may be used as well.





EIO-13
OUTPUT
SEND SIGNAL



Connect EIO13 on the back of the robot to the input of the handshake controller, and then wire the output of the handshake controller to a 3-Wire Input (Limit Switch) Input A on the VEX V5 Robot Brain. Use a 3-wire extension cable to do this.



VEX V5 - On the **OUTPUT** side you will use the WHITE and BLACK wires to trigger the input on the VEX V5 Robot Brain (the RED wire is not active).

DOBOT - If you are not using the handshake module as shown above, but some other form of optical isolator, understand that the signal from the back of the robot is coming from the Black (Negative) and Green (Positive) wires from the 4-Wire extension cable.

These correspond to the signal being sent from pins GP2.1 and GP2.2 on the back of the robot.

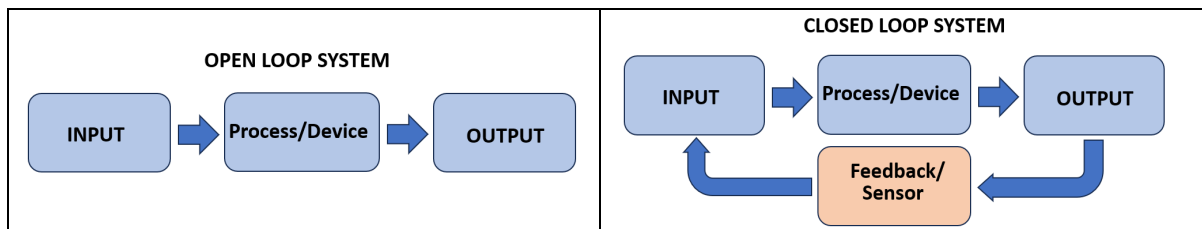


Be sure to consult the Dobot Input/Output manual if you want to use other inputs and outputs, as damage to your robot or your other equipment may result.

2. Attach the Air Pump/Vacuum Gripper kit or the Pneumatic Gripper to the robot as done in previous activities.

PART 1 - OPEN LOOP (LOOP WITH NO FEEDBACK)

For part one, we will develop a loop that is considered an OPEN loop. An open loop is a looping process that does not provide feedback. An open loop is often a system that is run on a timed loop rather than a sensor-based system that is aware that the process is complete (no feedback).

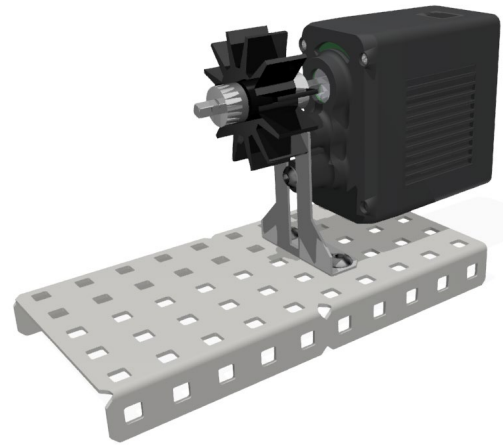
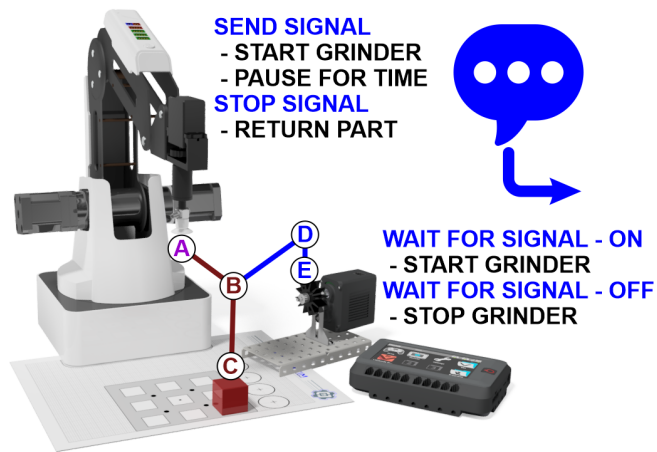


Examples:

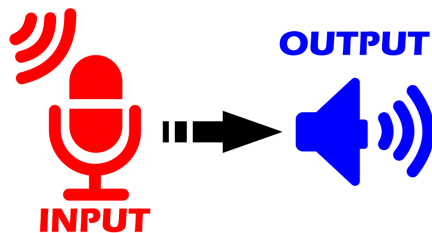
OPEN LOOP	VS.	CLOSED LOOP
Fire Pit		Electric Oven
No ability to control fire temperature		Temperature sensor used to provide feedback and control temperature
Gravity Pet Feeder		Electronic Pet Feeder
No ability to know if the feeder is empty		Sensors to provide feedback of when to distribute food, how much to distribute, and know when the feeder is empty



OPEN LOOP - 1WAY COMMUNICATION



1-WAY COMMUNICATION



4. Build a simple grinder to simulate a manufacturing process. In this case VEX parts are used to simulate a simple grinder.

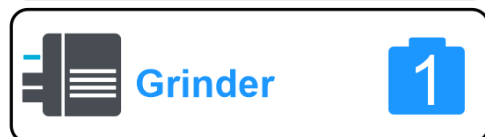
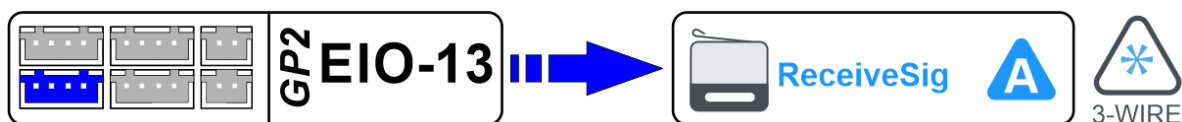
3. Set up the DobotLab program so that it moves the part through the motions of a grinding operation.
5. Write the microcontroller code that will make the robot complete the following tasks:
 - a. Send the robot home.
 - b. LOOP:
 - c. Wait for the limit switch to be pressed to start the sequence.
 - d. Get the part from the pallet.
 - e. Bring the part to the grinder.
 - f. Send signal to turn on the grinder.
 - g. Complete the grinding operation.
 - h. Turn off the signal to stop the grinder
 - i. Return the part to the pallet.
 - j. Send the robot home.

1	A HOME
2	WAIT FOR LIMIT (START)
3	B ABPART
4	C ATPART
5	B ABPART
6	D ABGRIND
7	DOBOT SEND SIG
8	MICRO RECEIVE SIG
9	E ATGRIND
10	DOBOT STOP SIG
11	D ABGRIND
12	B ABPART
13	C ATPART
15	B ABPART
14	A HOME

Pseudocode for grinding operation.



6. Microcontroller Setup and Programming



0.2 seconds (200 ms) waits are used to slow the program down (*called microwaits*). Often programs run through tasks faster than we can think and will skip through events. Example: *Looking for*

a signal to come on (HIGH) and then off (LOW) - Once the program sees the signal to start the grinder, the time it takes to start the grinder is almost immediate (spin Grinder Forward). We are then looking for the signal to drop low to stop the process and we we to make sure we are not looking at the instance of when the signal became true (unstable portion of the signal signal). The 0.2 second wait slows this process down to make sure the program doesn't skip through the process of waiting for the signal to stop (go LOW).



VEX Users: Be sure to set up the input to the VEX V5 Robot Brain as a Touch Sensor, not a Digital IN! If you use a Digital IN, "off" will be "on", and "on" will be "off".



7. *Write down any notes that you think you may need to make this work:*

8. Once both programs are written, Run the DobotLab program , and the VEXcode or microcontroller program. If written correctly, the robot will wait for the press of the switch, and the VEXcode program will wait for an input from the grinder.
If it does not work correctly, troubleshoot until it does.

9. *If your setup did not work correctly the first time, what did you have to do to make it work?*

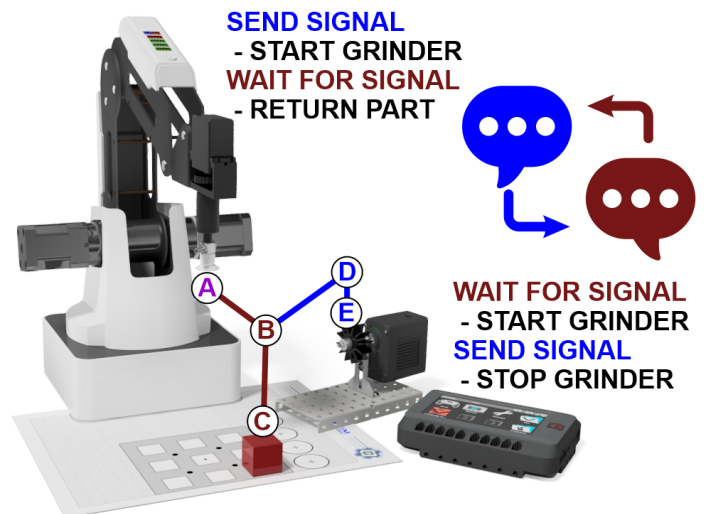


PART 2 - NEXT STEPS: CLOSED LOOP (LOOP WITH FEEDBACK)

10. If time permits create a second handshake module to establish two-way communication between the Dobot and the microcontroller, turning an open loop system (no feedback in the loop) into a closed loop system (a complete loop with feedback).

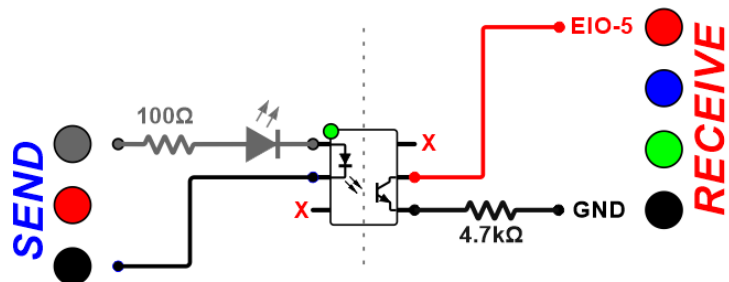
1	A HOME
2	WAIT FOR LIMIT (START)
3	B ABPART
4	C ATPART
5	B ABPART
6	D ABGRIND
7	DOBOT SEND SIG
8	MICRO RECEIVE SIG
9	E ATGRIND
10	DOBOT STOP SIG
11	DOBOT WAITS FOR SIG
12	DOBOT RECEIVES SIG
13	B ABPART
14	C ATPART
15	B ABPART
16	A HOME

CLOSED LOOP - 2WAY COMMUNICATION

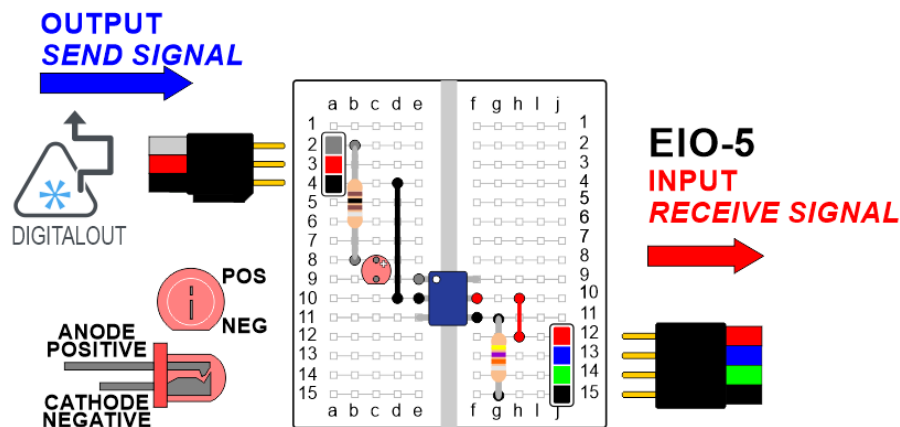


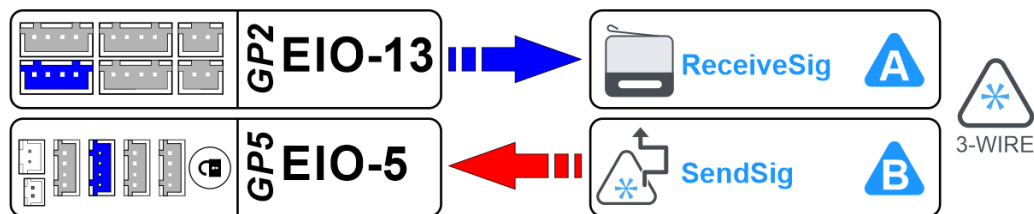
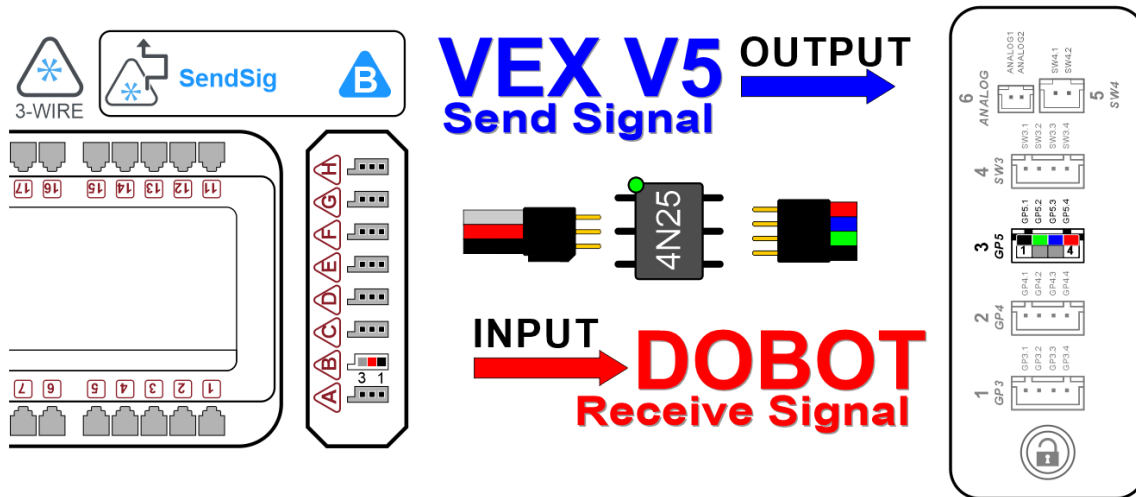
Pseudocode for closed loop operation.

11. 2nd Breadboard – Microcontroller to Dobot

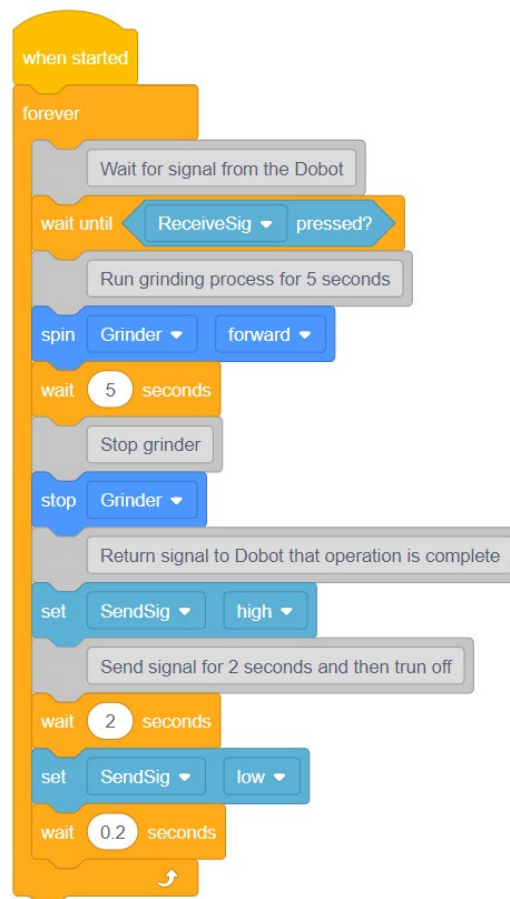
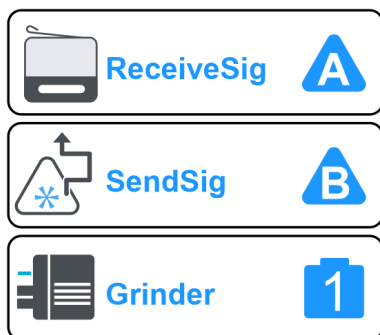


DIGITAL OUT - B





12. VEXcode Setup and Programming



CONCLUSION

1. *What would you have to do to make this program run five times without any human intervention? Explain fully.*
2. *What is the order of operations for successfully starting this setup? Which program has to start first? Explain why.*
3. *Which gripper did you choose to complete this activity? Justify your choice.*
4. *In VEXcode, if you use a Digital IN, Off will be On, and On will be Off. Explain why this is.*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- | | |
|-------|---|
| _____ | 1. Use a sensor to make this loop multiple times. Make it so the robot goes home after the first cycle and waits for a part to be placed before starting again. |
| _____ | 2. Complete Going Beyond #1 above and make it palletize the product when finished. |



2.5 Block - Handshaking: Dobot to Microcontroller

NAME: _____

Date: _____ Section: _____

INTRODUCTION

Robotic arms sometimes need to communicate with other machines in a workcell, or factory. This is called **HANDSHAKING** and can be done between different machines, devices, and robots. It is a very simple form of communication and is done with simple ones and zeros; or “ons” and “offs”.

In this activity you will learn how to make a robot handshake with a simulated machine. We will use a simple example: a grinder. You will make the robot talk to the grinder to turn it on during an automated grinding operation. Remember, this is CIM, we do not want to keep the grinder running all the time; we need to conserve energy, time, and resources, so the grinder needs to stop, and the robot will then replace the part.



In this example we will use VEX equipment to build our simulation. Your instructor may change this.



WARNING



Power OFF

Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur.

KEY VOCABULARY

- Inputs and Outputs
- Variables
- Open vs Closed Loop Systems
- Micro Wait
- My Blocks / Function / Voids
- Handshaking
- Optical Isolator

9v ! t a 9b¢ ¢ { ! t t l 9¢

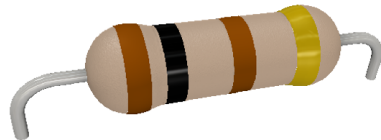
- Robot Magician
- Dobot Field Diagram
- 1" x ¾" cylinders or ¾" cubes
- Servo extension cables
- Dobot Input/Output Manual
- Breadboard/wire/4N25 Optical Isolator and 100 ohm & 4.7K ohm resistors
- VEX V5, PIC, Arduino may all be used in this activity, but wiring may vary.
- DobotLab software
- VEXcode V5 control software
- Pneumatic Gripper or Suction Cup Gripper
- Handshake module
- Normally open digital switch
- VEX parts: various for building the grinder including a baseplate, motor & controller or smart motor.

twh /95 ! w9



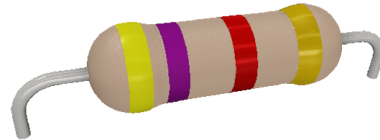
/I-dzú!2y! b9±9w 0!NŠ I-yēúK!y3 ú2 úKŠ 5202ú a I-3!0!-y 0K!tŠ lú KI-ā L!20ŠN 2y0 ! [² ! , { údzN!y lú 2FF 0ŠT2NŠ YI-!y3 02yYŠ0ú!2yā 2N! R!-YI-3Š ú2 úKŠ N!202ú 02dz!R 200dzN! . Š ādzNŠ ú2 I-ā! è2dzN! yāúN!dz0ú2N! 1F è2dz KI-0Š I-yē l!dzŠāú!2yā0

1. Wire the robot to the VEX V5 Robot Brain through the handshake controller as shown below. Any other microcontroller, such as an Arduino a PIC or a Raspberry Pi may be used as well.



100 Ω - **SEND SIDE**

BROWN	BLACK	BROWN	GOLD
1	0	X10	5%



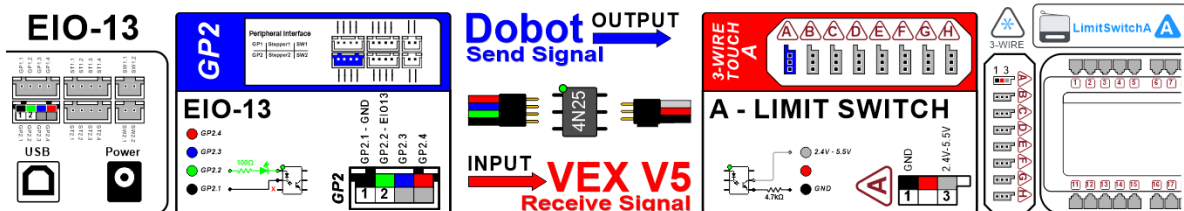
4.7k Ω - **RECEIVE SIDE**

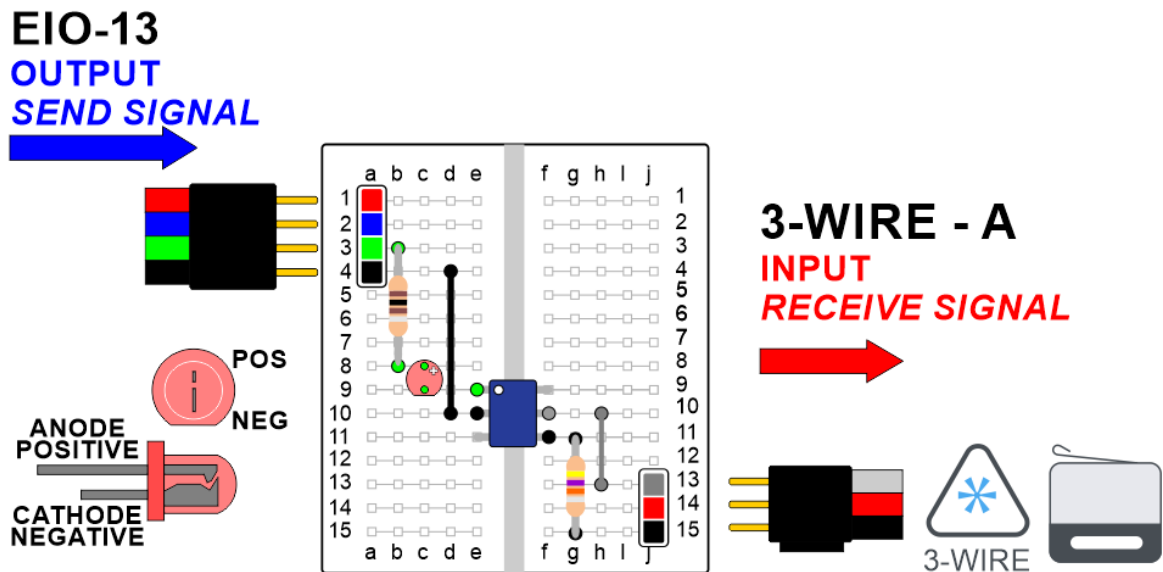
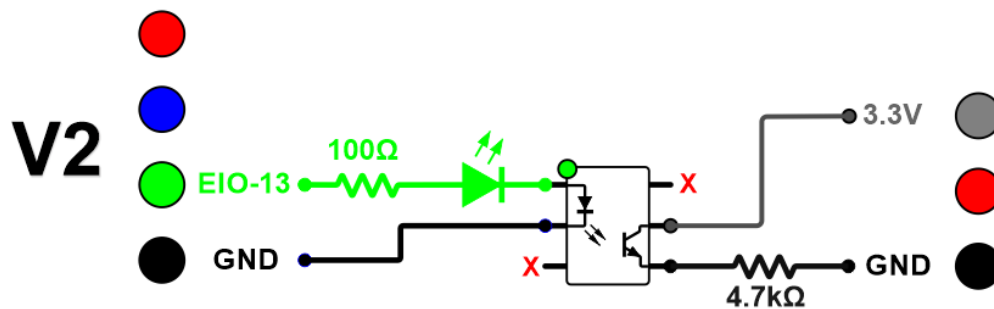
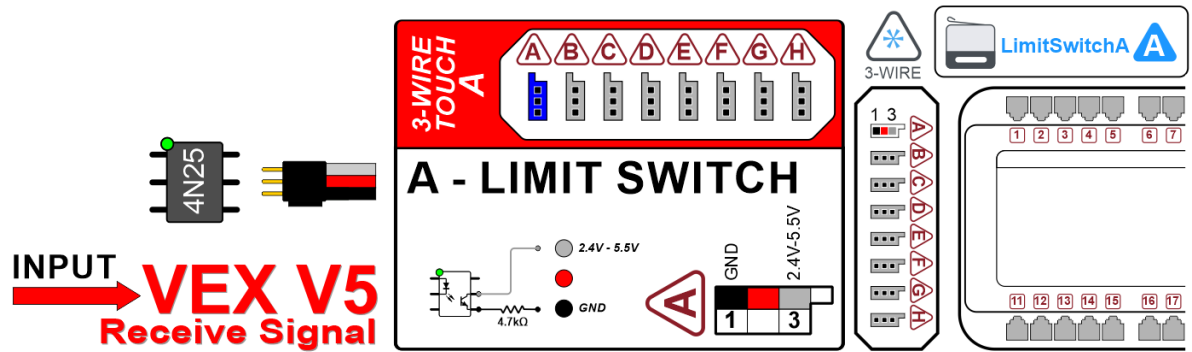
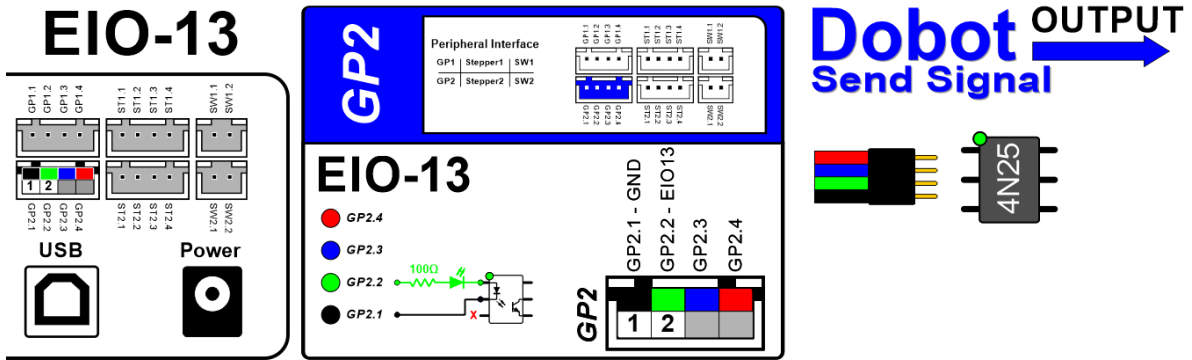
YELLOW	VIOLET	RED	GOLD
4	7	X100	5%

Back of Dobot Magician V2
GP2 – EIO-13

Optical Isolator

VEX V5 Controller
3-Wire Limit Switch Input





Connect EIO13 on the back of the robot to the input of the handshake controller, and then wire the output of the handshake controller to a 3-Wire Input (Limit Switch) Input A on the VEX V5 Robot Brain. Use a 3-wire extension cable to do this.



VEX V5 - On the **OUTPUT** side you will use the **WHITE** and **BLACK** wires to trigger the input on the VEX V5 Robot Brain (the **RED** wire is not active).

DOBOT - If you are not using the handshake module as shown above, but some other form of optical isolator, understand that the signal from the back of the robot is coming from the Black (Negative) and Green (Positive) wires from the 4-Wire extension cable.

These correspond to the signal being sent from pins GP2.1 and GP2.2 on the back of the robot.

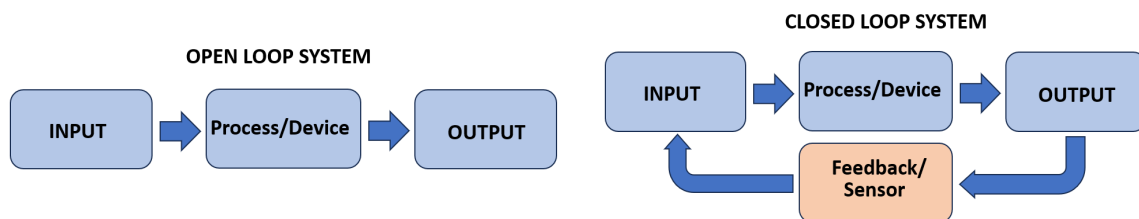


Be sure to consult the Dobot Input/Output manual if you want to use other inputs and outputs, as damage to your robot or your other equipment may result.

2. Attach the Air Pump/Vacuum Gripper kit or the Pneumatic Gripper to the robot as done in previous activities.

PART 1 - OPEN LOOP (LOOP WITH NO FEEDBACK)

For part one, we will develop a loop that is considered an OPEN loop. An open loop is a looping process that does not provide feedback. An open loop is often a system that is run on a timed loop rather than a sensor-based system that is aware that the process is complete (no feedback).

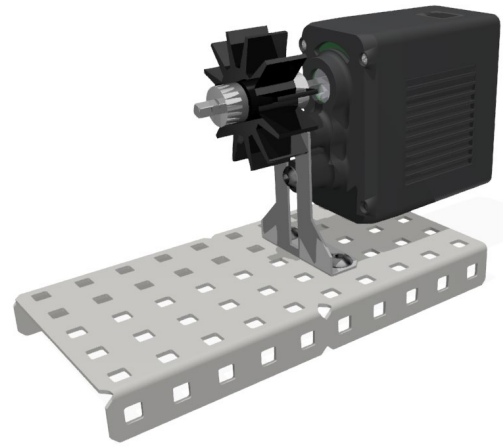
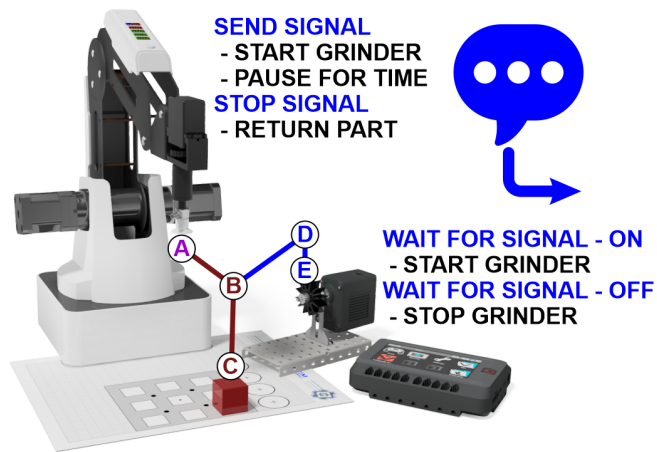


Examples:

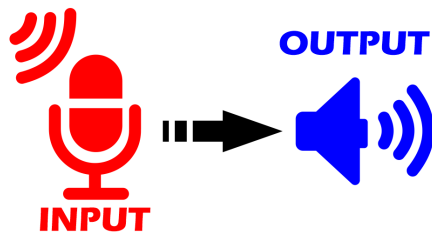
OPEN LOOP	VS.	CLOSED LOOP
Fire Pit <i>No ability to control fire temperature</i>		Electric Oven <i>Temperature sensor used to provide feedback and control temperature</i>
Gravity Pet Feeder <i>No ability to know if the feeder is empty</i>		Electronic Pet Feeder <i>Sensors to provide feedback of when to distribute food, how much to distribute, and know when the feeder is empty</i>



OPEN LOOP - 1WAY COMMUNICATION



1-WAY COMMUNICATION



4. Build a simple grinder to simulate a manufacturing process. In this case VEX parts are used to simulate a simple grinder.

3. Set up the DobotLab program so that it moves the part through the motions of a grinding operation.
5. Write the microcontroller code that will make the robot complete the following tasks:
 - a. Send the robot home.
 - b. LOOP:
 - c. Wait for the limit switch to be pressed to start the sequence.
 - d. Get the part from the pallet.
 - e. Bring the part to the grinder.
 - f. Send signal to turn on the grinder.
 - g. Complete the grinding operation.
 - h. Turn off the signal to stop the grinder
 - i. Return the part to the pallet.
 - j. Send the robot home.

1	A HOME
2	WAIT FOR LIMIT (START)
3	B ABPART
4	C ATPART
5	B ABPART
6	D ABGRIND
7	DOBOT SEND SIG
8	MICRO RECEIVE SIG
9	E ATGRIND
10	DOBOT STOP SIG
11	D ABGRIND
12	B ABPART
13	C ATPART
15	B ABPART
14	A HOME

Pseudocode for grinding operation.



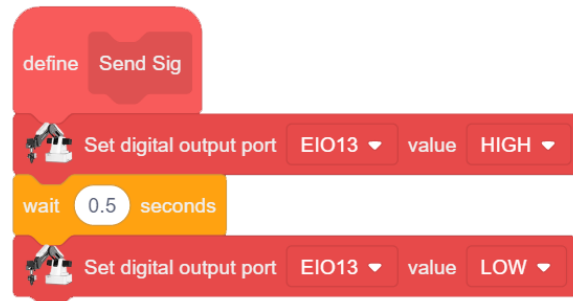
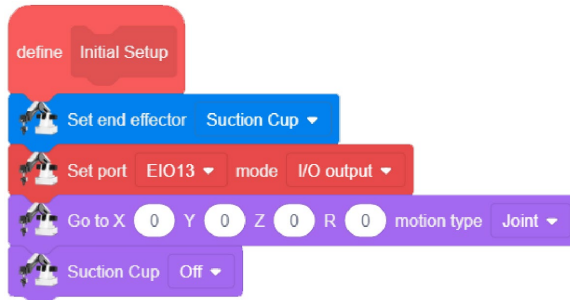
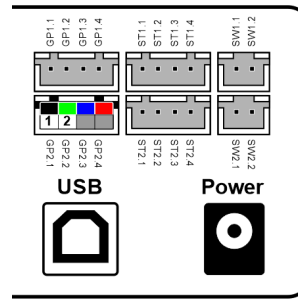
Dobot Setup and Programming

6. A low-voltage signal will be sent from the Dobot to the VEX V5 Controller

Create a similar program to the one that was created for the Dobot to Dobot Activity.

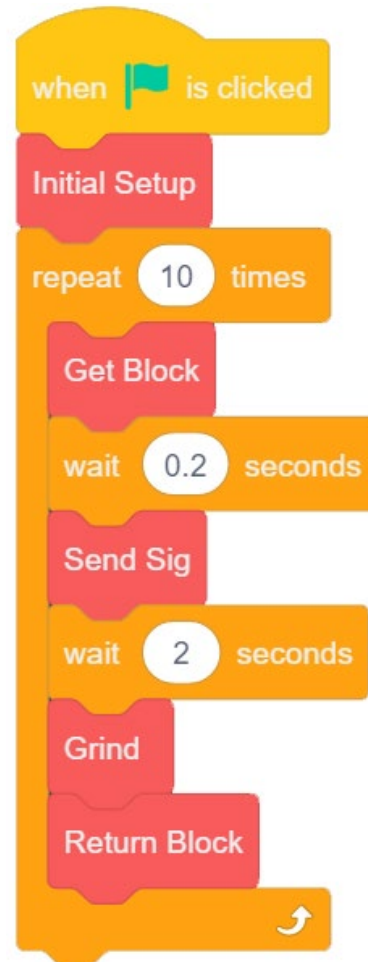
We will again use EIO-13 to send a signal

EIO-13



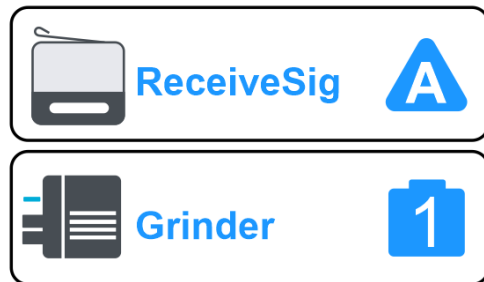
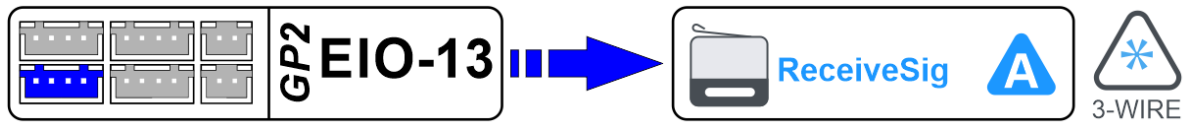
Break your program up into 5 parts

1. MY BLOCK – Initial Setup
 - Setup EoAT
 - Setup I/O Ports
2. MY BLOCK – Get Block
 - Pick the block from a common position
 - Take the block over to the grinder
3. MY BLOCK – SEND SIGNAL
 - Send a signal to the V5 to start the grinder
 - Turn off the signal (the V5 should start as soon as the signal is received).
4. MY BLOCK – GRIND
 - Grind the block for a set amount of time
5. MY BLOCK – Return Block
 - Return the block and go home and prepare for another loop



Microcontroller Setup and Programming

7. A signal will be Received by the V5 Controller as a 3-Wire Limit Switch



0.2 seconds (200 ms) waits are used to slow the program down (*called microwaits*). Often programs run through tasks faster than we can think and will skip through events. Example: *Looking for a signal to come on (HIGH) and then off (LOW)* - Once the program sees the signal to start the grinder, the time it takes to start the grinder is almost immediate (spin Grinder Forward). We are then looking for the signal to drop low to stop the process and we we to make sure we are not looking at the instance of when the signal became true (unstable portion of the signal signal). The 0.2 second wait slows this process down to make sure the program doesn't skip through the process of waiting for the signal to stop (go LOW).



VEX Users: Be sure to set up the input to the VEX V5 Robot Brain as a Touch Sensor, not a Digital IN! If you use a Digital IN, "off" will be "on", and "on" will be "off".



8. *Write down any notes that you think you may need to make this work:*

9. Once both programs are written, Run the DobotLab program , and the VEXcode or microcontroller program. If written correctly, the robot will wait for the press of the switch, and the VEXcode program will wait for an input from the grinder.
If it does not work correctly, troubleshoot until it does.

10. *If your setup did not work correctly the first time, what did you have to do to make it work?*

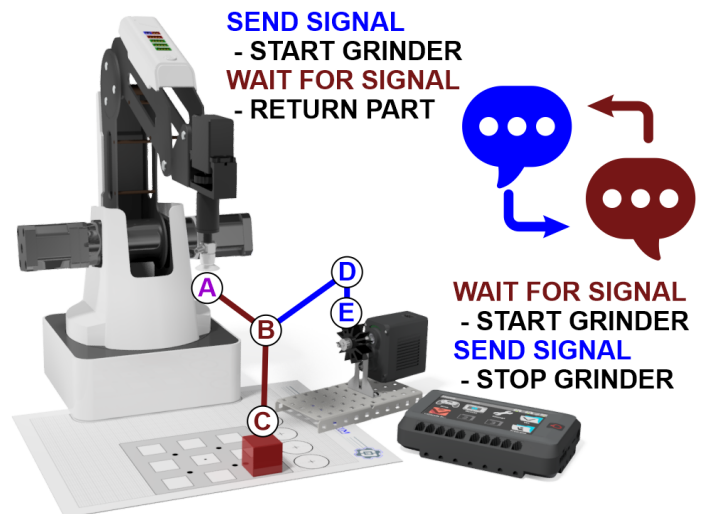


PART 2 - NEXT STEPS: CLOSED LOOP (LOOP WITH FEEDBACK)

11. If time permits create a second handshake module to establish two-way communication between the Dobot and the microcontroller, turning an open loop system (no feedback in the loop) into a closed loop system (a complete loop with feedback).

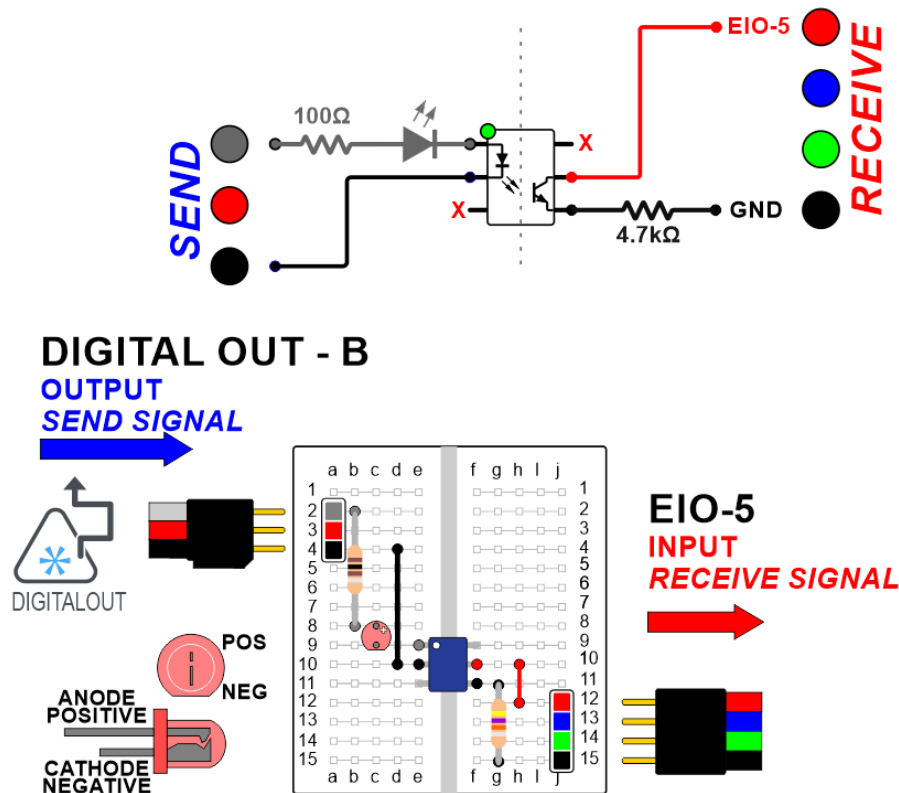
1	A HOME
2	WAIT FOR LIMIT (START)
3	B ABPART
4	C ATPART
5	B ABPART
6	D ABGRIND
7	DOBOT SEND SIG
8	MICRO RECEIVE SIG
9	E ATGRIND
10	DOBOT STOP SIG
11	DOBOT WAITS FOR SIG
12	DOBOT RECEIVES SIG
13	B ABPART
14	C ATPART
15	B ABPART
16	A HOME

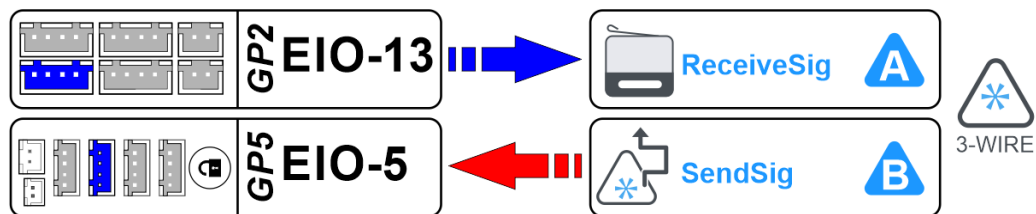
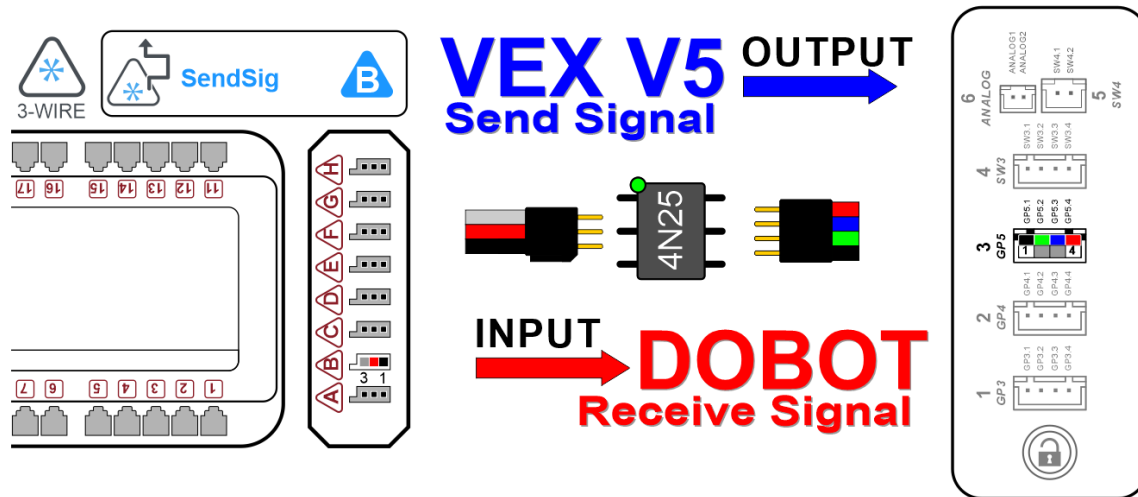
CLOSED LOOP - 2WAY COMMUNICATION



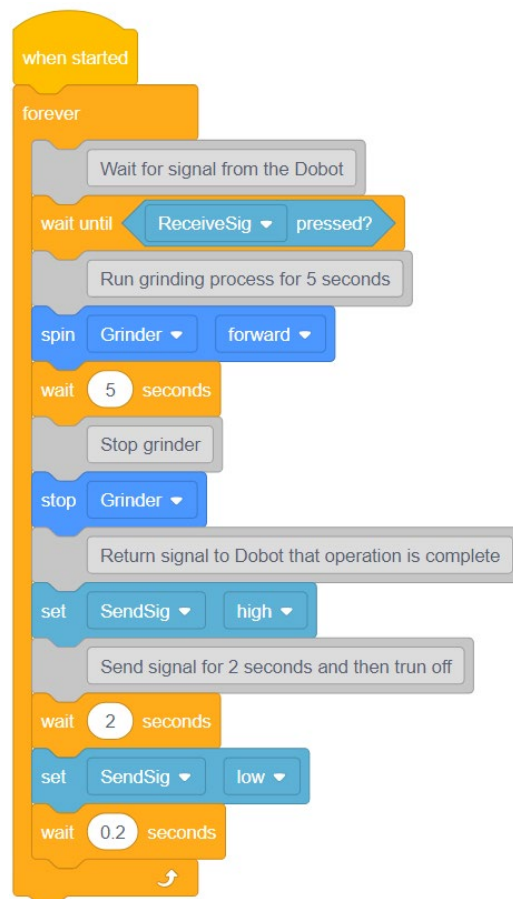
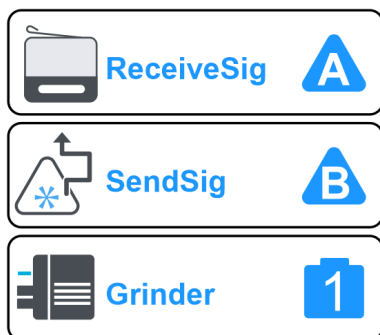
Pseudocode for closed loop operation.

12. 2nd Breadboard – Microcontroller to Dobot





13. VEXcode Setup and Programming



CONCLUSION

1. *What would you have to do to make this program run five times without any human intervention? Explain fully.*
2. *What is the order of operations for successfully starting this setup? Which program has to start first? Explain why.*
3. *Which gripper did you choose to complete this activity? Justify your choice.*
4. *In VEXcode, if you use a Digital IN, Off will be On, and On will be Off. Explain why this is.*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- | | |
|-------|---|
| _____ | 1. Use a sensor to make this loop multiple times. Make it so the robot goes home after the first cycle and waits for a part to be placed before starting again. |
| _____ | 2. Complete Going Beyond #1 above and make it palletize the product when finished. |



2.6 Block - Handshaking VEX V5 to VEX V5

NAME: _____

Date: _____ Section: _____

INTRODUCTION

Often the process of handshaking goes beyond a robot's need to communicate with another robot. In industry the process of have **Programmable Logic Controllers (PLCs)** communicate with each other, or robotic arms is very simple. It is done with simple ones and zeros; or "ons" and "offs".

In this activity we will reuse the theory of communicating with simple ones and zeros; or "ons" and "offs" from the previous activity. We will replace the robots with two microcontrollers. For this activity, we will focus on the wiring and syntax programming for two VEX microcontrollers.

In this activity, the limit switch attached to each microcontroller will control the on and off function of the other controller's motor.



In this activity we will be using VEX V5 equipment, but the concepts taught can be used with other microcontrollers as well.

KEY VOCABULARY

- Limit Switch
- If/Else Statement
- While True Loop (Forever Loop)
- Handshaking

EQUIPMENT & SUPPLIES

- 2 VEX V5 Brains
- 2 3-Wire Limit Switches
- 2 VEX USB Micro Cables
- 4 3-Wire Extension Cables
- Handshake Modules
- 2 Smart Motors with Smart Cables
- 2 VEX V5 Batteries with Connection Cables
- 4 3-Pin Male Headers

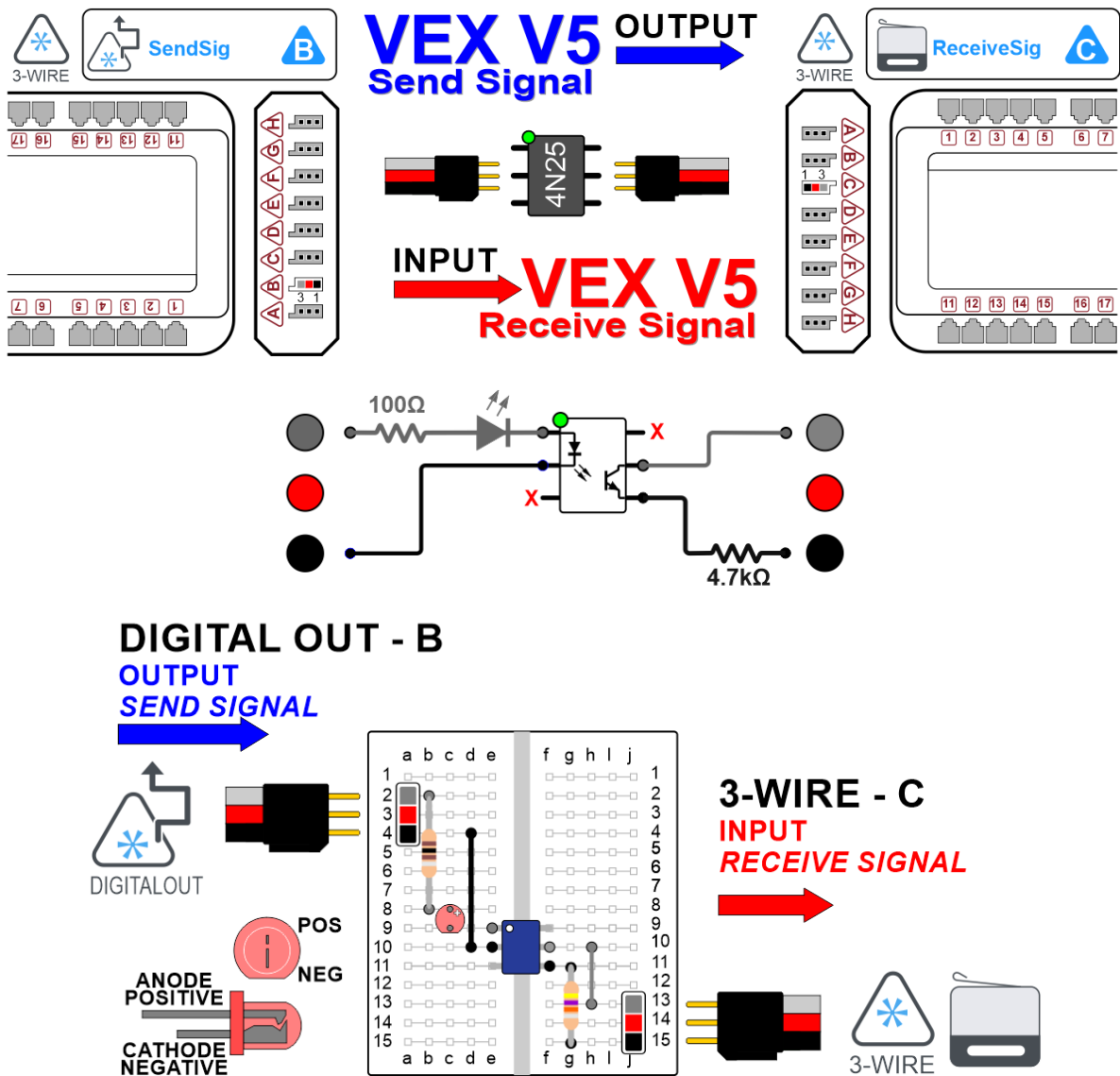
ESSENTIAL QUESTIONS

Essential questions answered in this activity include:

- How do I make my VEX V5 Brain send and receive signals?
- What kind of software is necessary to communicate?
- How do I wire the hardware to communicate?
- How is VEXCode V5 similar to Arduino software? How is it different?

PROCEDURE

1. The **digital output** of each V5 Brain (3-Wire B) will be attached to a Limit Switch (3-Wire A) of the other V5 Brain
2. Build and wire TWO handshake modules. One as seen below and one mirrored. If you have a large enough breadboard it may be easier to wire them both on the same one.

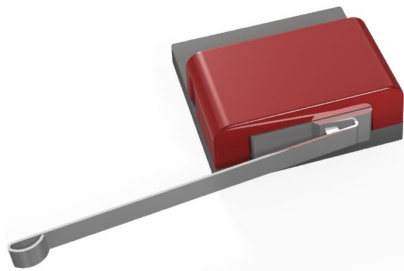


Wire a **Limit Switch** to each V5 Brain as a 3-Wire LimitSwitch on PORT A.

Notes:

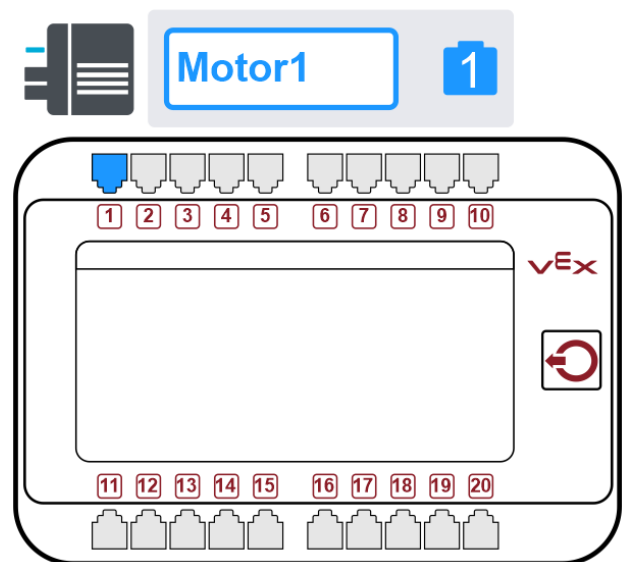
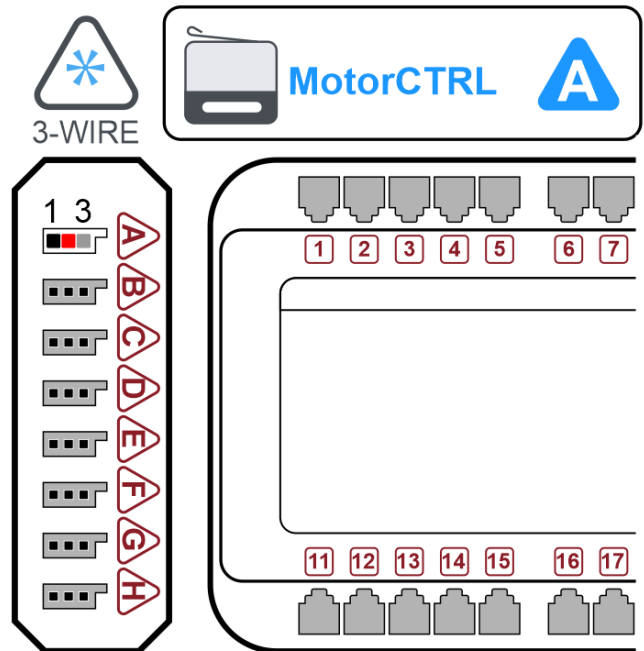
- The VEX **Limit Switch** will be used as a digital input.
- The VEX Limit Switch is a momentary switch. When it is pressed, it will be seen as a high (true/ON) signal. When it is released, it will be seen as a low (false/OFF) signal.

Repeat the process for the other V5 Brain

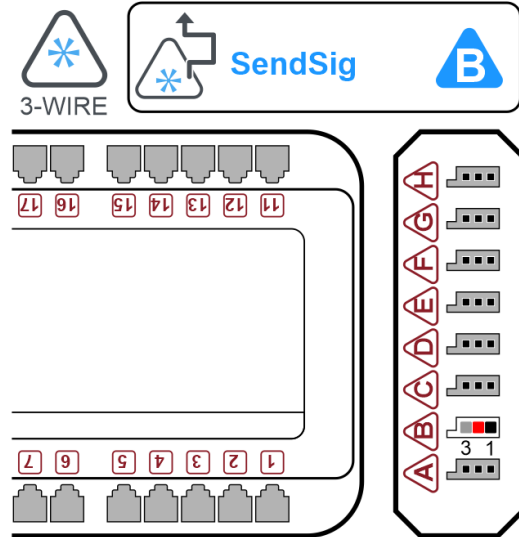


Wire a VEX Smart Motor to PORT 1.

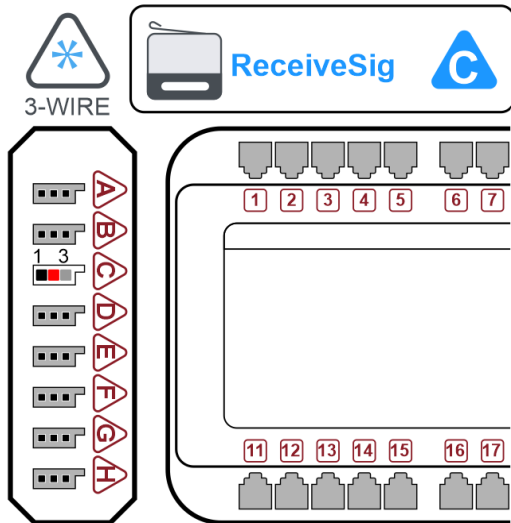
Repeat the process for the other V5 Brain



Attach a 3-Wire Extension Cable to PORT B as a **DIGITAL OUT** (SendSig). This wire will be attached to the **OUTPUT** side of handshake module to send a signal over to the other V5 Brain



Attach a 3-Wire Extension Cable to PORT C as a LIMIT SWITCH (ReceiveSig). This wire will be attached to the **INPUT** side of handshake module to receive a signal over to the other V5 Brain.

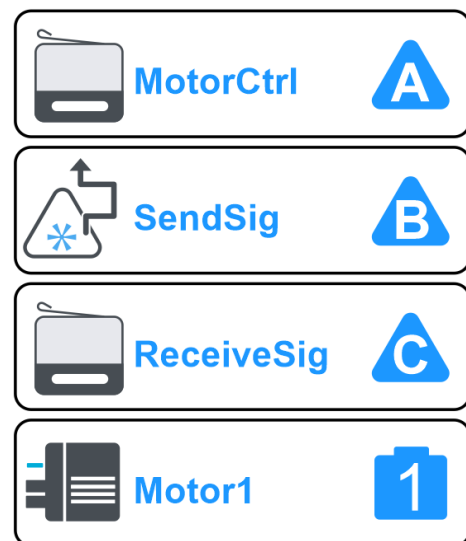


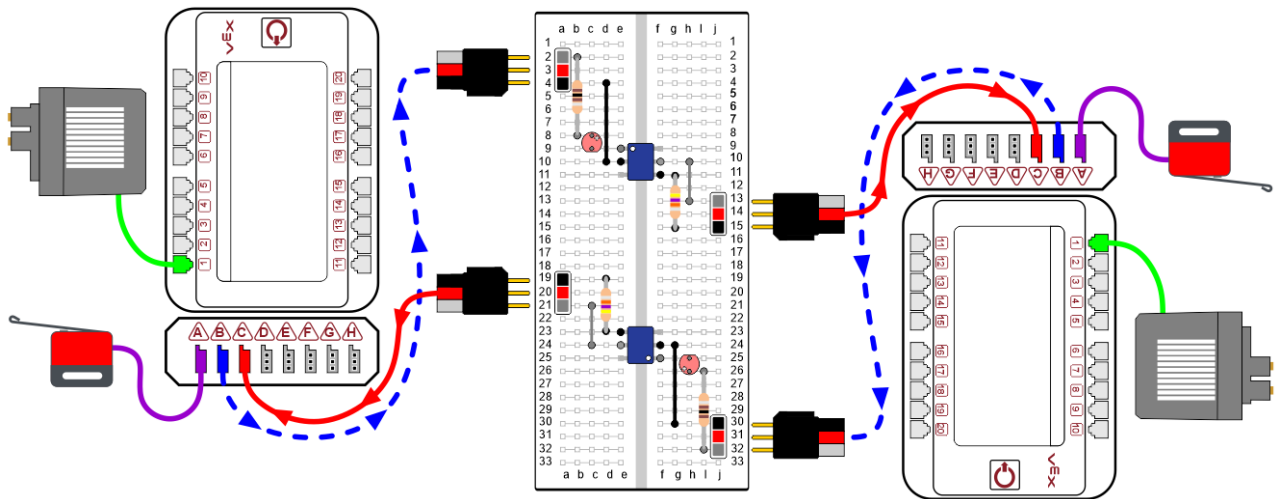
MotorCtrl – User controlled INPUT that when received, will call for a signal to be sent to the other V5 Brain to turn on its Motor.

SendSig – A digital signal that will be turned on when the MotorCtrl input is received to send a signal over to the other V5 Brain.

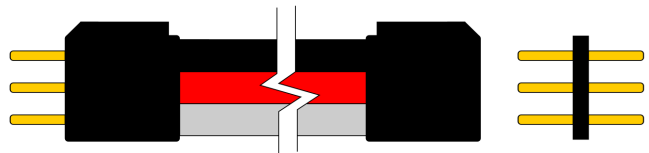
ReceiveSig – A digital signal received from the other V5 Brain that when received will turn on the attached Motor.

Motor1 – Motor that's operation is controlled from the other V5 Brain.





A 3-Pin Male Header will need to be added to the end of each 3-Wire extension cable in order to attach them to the breadboard.



WARNING

At the conclusion of the activity, remove the header pin extensions. If they are accidentally inserted into Ports A-H, the pins may be stuck inside the V5 Brain rendering the ports unusable.

Programming

Now that all of the wiring is complete, develop the programming theory for this activity.

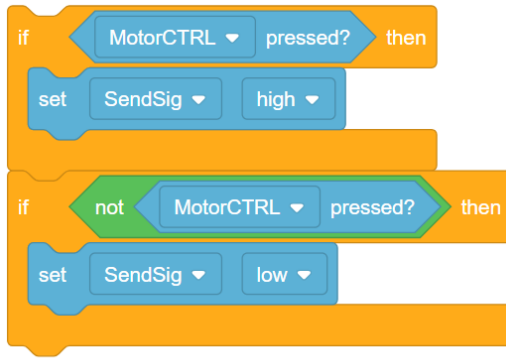
Start by defining the actions that will control each individual microcontroller. (the program for one controller should be identical to the other)

Define the other conditions.... What to do when no signal is sent or received.

Limit Switch MotorCtrl - Port A	➡	Send Signal SendSig - Port B
Receive Signal ReceiveSig - Port C	➡	Motor ON Motor1 - Port 1

NO Switch MotorCtrl - Port A	➡	Stop Signal SendSig - Port B
NO Signal ReceiveSig - Port C	➡	Motor OFF Motor1 - Port 1

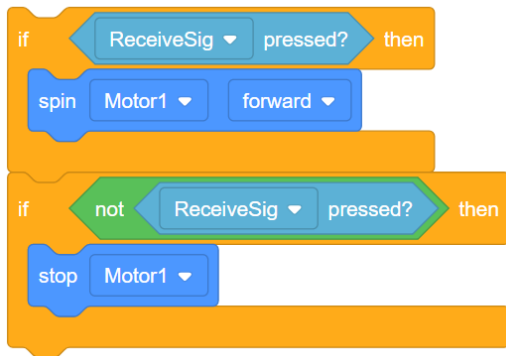




Now start grouping our individual actions into like groups.

Limit Switch MotorCtrl - Port A	==>	Send Signal SendSig - Port B
------------------------------------	-----	---------------------------------

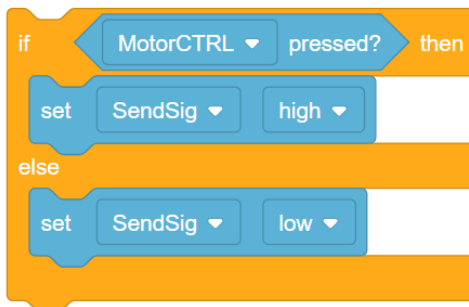
Receive Signal ReceiveSig - Port C	==>	Motor ON Motor1 - Port 1
---------------------------------------	-----	-----------------------------



NO Switch MotorCtrl - Port A	==>	Stop Signal SendSig - Port B
---------------------------------	-----	---------------------------------

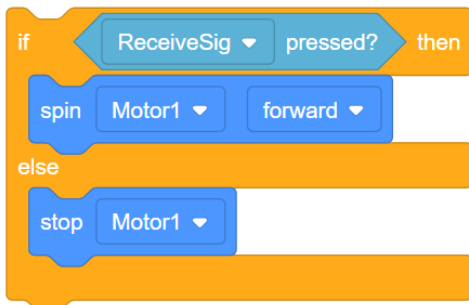
NO Signal ReceiveSig - Port C	==>	Motor OFF Motor1 - Port 1
----------------------------------	-----	------------------------------

Notice that there are only two possible situations for each grouping. It is, or is not. We can simplify our code to If a specific condition is met, do operation 1 else, do operation 2.



IF Limit Switch MotorCtrl-Port A	==>	Send Signal SendSig-Port B
--	-----	-------------------------------

ELSE	==>	Stop Signal SendSig -Port B
-------------	-----	--------------------------------



IF Receive Signal ReceiveSig-Port C	==>	Motor ON Motor1-Port 1
---	-----	---------------------------

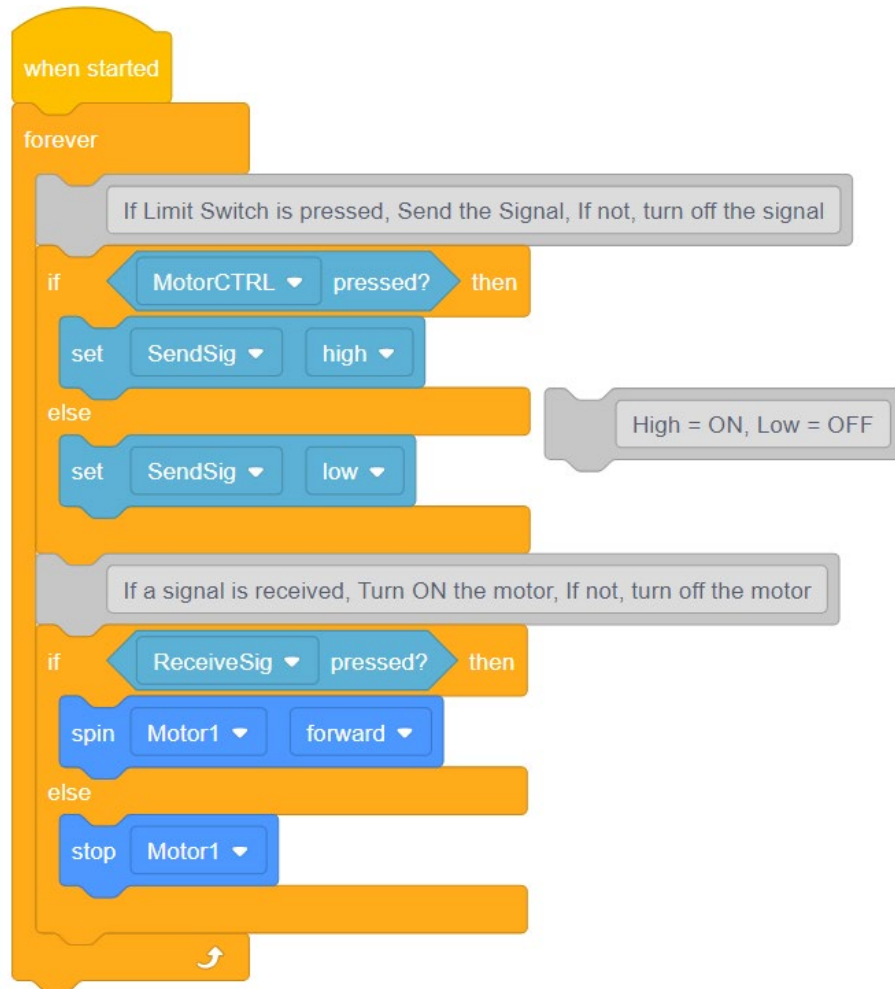
ELSE	==>	Motor OFF Motor1-Port 1
-------------	-----	----------------------------



In order to get the program to constantly evaluate either or both situations we will put our program in a FOREVER Loop



An example of the entire program in VEXCode V5 is shown below.



Once this portion of the program is completed, run it, and see if it works correctly. If it does not work, troubleshoot it until it does.

If your set up did not work correctly the first time, what did you have to do to make it work?



TROUBLESHOOTING THE OPTICAL ISOLATOR

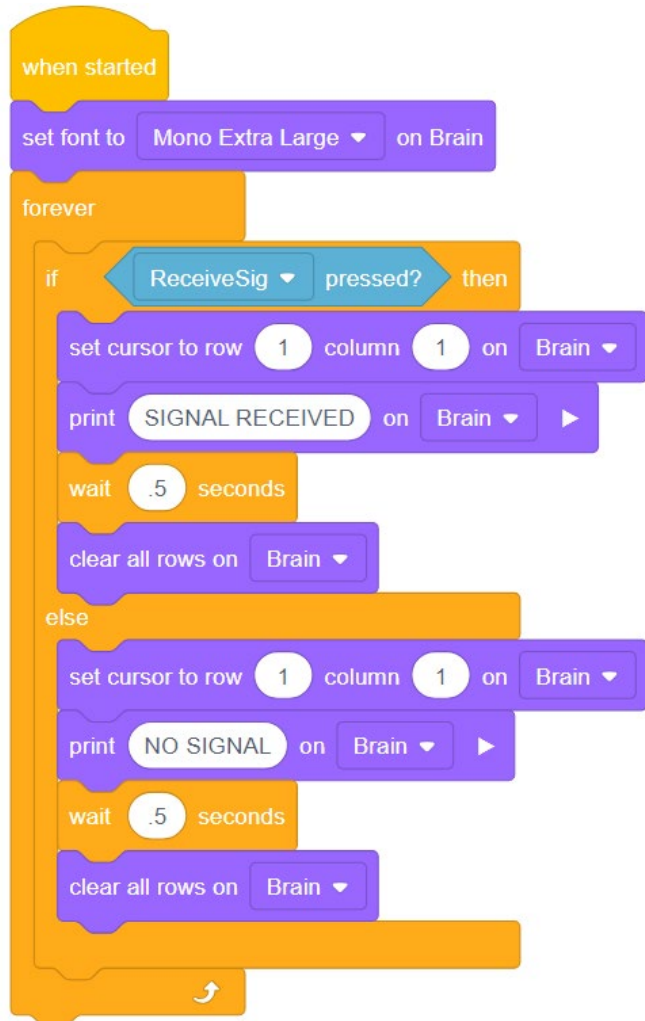
Programming not working? It could be one of many different issues.



Try creating a simple program to show the value (ON/OFF, High/Low) of the signal being received on the V5 Brain. When the Limit Switch is pressed on the other V5 Brain, it should report to the screen "Signal Received". If no signal is received (else), it should read "NO signal".

WHY IS IT NOT WORKING? Check the following things below:

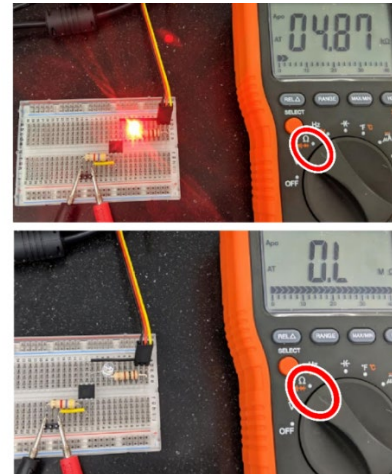
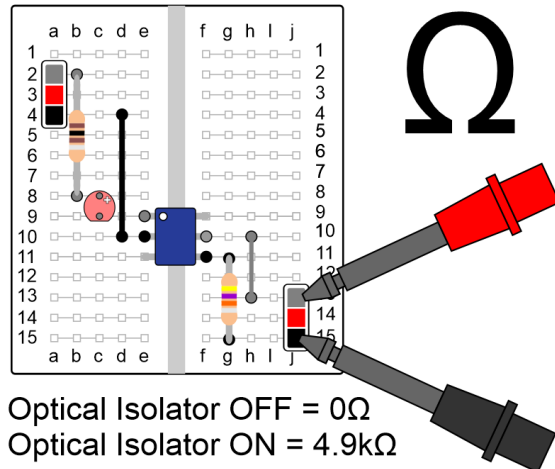
- LED Backwards?
- Not using the correct pins of the Optical Isolator?
- Plugged into the incorrect port?
- 3-Wire plugged in incorrectly (Breadboard)?
- 4-Wire plugged in incorrectly (VEX Brain)?
- Jumper wire incorrect?
- VEX Program – written correctly?
- Bad LED?
- Bad Optical Isolator? (there is an LED inside it as well)



When the digital Output is turned on / high, the LED on the handshake module should turn on.



If the V5 Brain reads correctly, the LED turns on, but no signal is being seen by the other V5 Brain, check your wiring. As a last resort, the Optical Isolator could be damaged. Ask your instructor to help you evaluate the signal side of the Isolator using a voltmeter and the images below. Voltmeter should be set to OHMS. The meter should read zero when no signal is present and approximately the value of the resistor used on the signal side when a signal is present.



CONCLUSION

1. What would you have to do to make this program run five times without any human intervention? Explain fully below.
2. What other ways could you use on your V5 Brain to start this process?

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- _____ 1. Change the Motors to servos. Be sure to get your instructor's permission.
- _____ 2. Replace the Limit Switch with a Potentiometer or a distance sensor



2.7 - Magic Box to Magic Box Handshaking “NO” Relay Module

NAME: _____

Date: _____

Section: _____

INTRODUCTION



These activities have shown you that you can **handshake** without an **optical isolator** or relay for hardware or you can do it through software with **variables** and **broadcasting**. There is still yet one more way to make two Dobot's communicate with one another. This is through **Bluetooth**. Bluetooth is a short-range wireless communication protocol that will allow two devices, such as robots to communicate wirelessly across a room.

In this activity, signals will be sent wirelessly between two **magic boxes**. The **2-Button Input** module will be the input and the **LED module** will be the output.



KEY VOCABULARY

- Input
- Output
- Optical Isolator
- Broadcast
- Magic Box
- LED Module
- Signal
- Handshake
- Variables
- Bluetooth
- 2-Button input

EQUIPMENT & SUPPLIES

- 2 Dobot Magic Boxes
- Digital - 2-Button Input Module
- LED Module
- 2 Separate Computers
- DobotLab software
- 2x 12V Power Cable and Supply
- 4x Common Sensor Cables
- 2x USB C to A Cable

ESSENTIAL QUESTIONS

Essential questions answered in this activity include:

- How do I get a Magic Box to send a Bluetooth signal?
- How do I get a Magic Box to receive a Bluetooth signal?
- What hardware is required to send and receive signals?
- How are signals programmed in Blockly?
- What are the advantages and disadvantages for sending signals wirelessly?

PROCEDURE

SECTION 1 – SETUP BOTH ROBOTS



WARNING



Power OFF

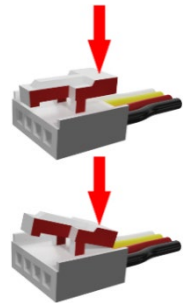
Caution: NEVER wire anything to the Magic Box while it has power on.

ALWAYS shutdown the BOX before making connections or damage to the controller could occur. Be sure to ask your instructor if you have any questions.



WARNING

Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.



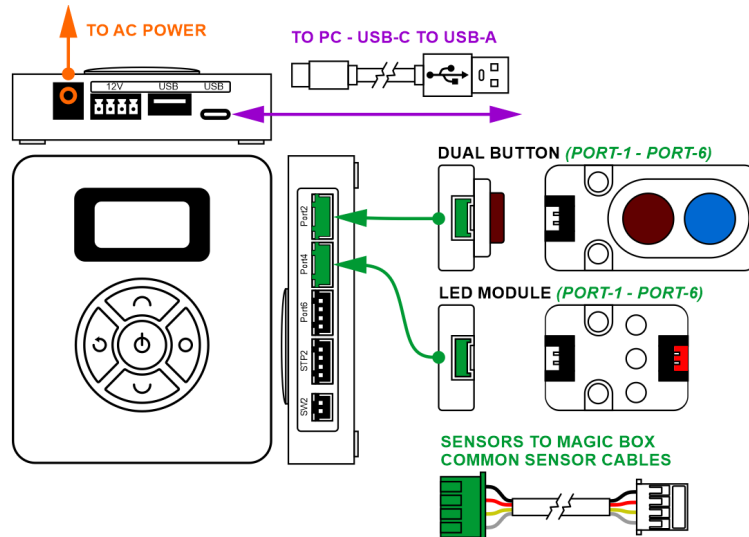
SECTION 2 – WIRING the MAGIC BOX AND SENSORS – 1st SETUP

SETUP #1

1. For this activity, you will need:

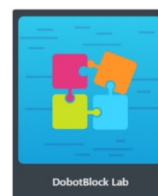
- 2x Magic Box
- 2x USB-A to USB-C Cable
- 2x AC/DC Power Adapter (12V)
- 4x Common Sensor Cables
- 1x Dual Button – **PORT 2**
- 1x LED Module – **PORT 4**

Wire the Magic Box as shown below



SECTION 3 – CONNECTING MAGIC BOXES to the COMPUTER & DOBOTLAB

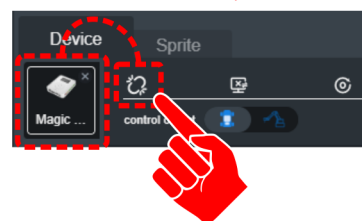
1. Connect both Magic Boxes to separate computers using their included USB cables
2. Open up DobotBlock Lab in the software.



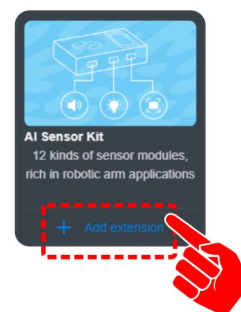
3. Once all of the wiring is done (sensors connected), power ON the Magic Box



4. Follow the same process from previous activities to add the Magic Box as a device and connect the software to it (establish communication).



5. Click on "+ Add extension" for the AI SENSOR KIT.



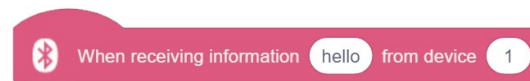
SKILL BUILDER 1 – SEND A SIGNAL THROUGH BLUETOOTH

The setup for sensing a signal between Magic Boxes is done with a few tools.

SET UP BLUETOOTH GROUP is the block that allows the program to send signals on a unique Bluetooth network. Each device must run this block at the beginning of the program.

A unique group (Street Name) should be chosen for each group of Magic Boxes that are to communicate (6 digit number. (000001 – 999999))

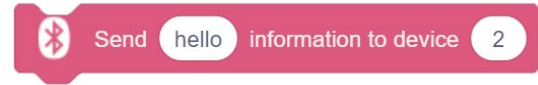
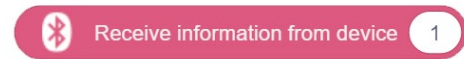
Each device is then given a specific device ID (House Number). The number can be any number from 0-255



WHEN RECEIVING INFORMATION – Work similar to BROADCASTING MESSAGES. This block can replace the WHEN STARTED FLAG event header to start an operation each time a signal is received.

RECEIVE INFO FROM DEVICE – Can be set to a variable. Both numerical and string values can be received.

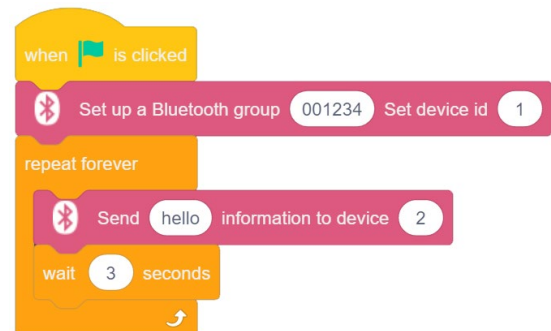
SEND INFO TO DEVICE – Block used to send numeric and string values to a specific device in the group.



The Bluetooth group # must be 6 digits long. For example: if you want to use the Bluetooth group # 1234, as in this activity, you must type in the leading two 0's: 001234!

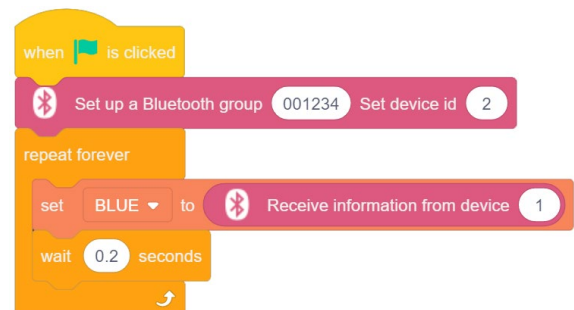
DEVICE 1

1. Drag over a SETUP BLUETOOTH and a SEND BLUETOOTH from the WIRELESS category in the block toolbox.
2. Create the program to the right on one computer.
 - a) Set up a unique group number for both devices.
 - b) This device will be DEVICE ID 1
 - c) Send the STRING HELLO to DEVICE 2 in a loop every three seconds.



DEVICE 2

3. On the second computer, create a new variable (BLUE)
4. Drag over a SETUP BLUETOOTH and a RECEIVE INFO from the WIRELESS category in the block toolbox.
5. Create the program to the right on one computer.
 - a) Use the same group number from the 1st device.
 - b) This device will be DEVICE ID 2.
 - c) Set the VARIABLE to the information RECEIVED from device 1.



Run both programs. The STRING hello should briefly appear and disappear in the BLUE variables box.

BLUE hello

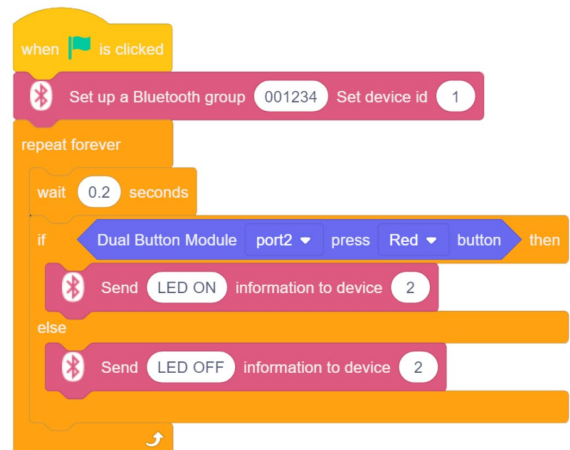
If your set up did not work correctly the first time, what did you have to do to make it work?

SKILL BUILDER 2 – CONTROL A DEVICE THROUGH BLUETOOTH

Now that a signal can be both sent and received through Bluetooth, use this signal to control one device from another. The following program will use a button press (DUAL BUTTON) from Magic Box 1 to turn ON and OFF the LED MODULE from Magic Box 2.

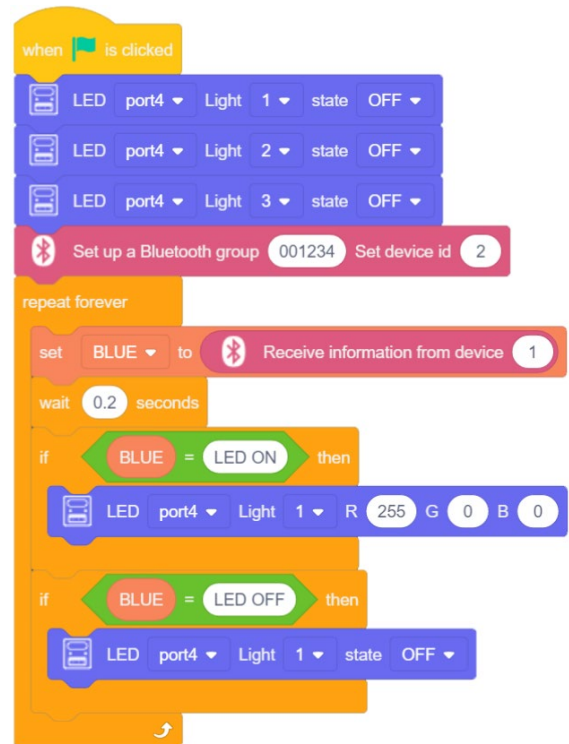
DEVICE 1

1. Drag over a SETUP BLUETOOTH and a SEND BLUETOOTH from the WIRELESS category in the block toolbox.
2. Create the program to the right on one computer.
 - a) Keep the same GROUP and ID numbers
 - b) Drag over an IF/ELSE Statements from the Control toolbox
 - c) IF BUTTON RED is pressed, Send the STRING LED ON else LED OFF



DEVICE 2

3. Drag over a SETUP BLUETOOTH and a RECEIVE BLUETOOTH from the WIRELESS category in the block toolbox.
4. Create the program to the right on computer 2.
 - a) Start the program by turning off all of the LEDs
 - b) Keep the same GROUP and ID numbers
 - c) Set the variable BLUE to the info received from device 1
 - d) Drag over two IF Statements from the Control toolbox
 - e) IF variable BLUE = LED ON, turn ON LED 1 (RED)
 - f) IF variable BLUE = LED OFF, turn OFF LED 1



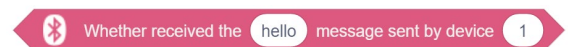
Run both programs.



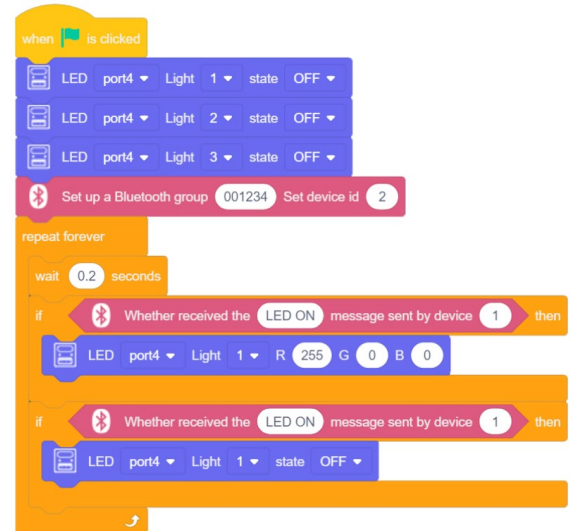
NOTE: There will be a small delay in the LED turning on and OFF. Be patient, the delay may be 1 to 2 seconds.

If your set up did not work correctly the first time, what did you have to do to make it work?

The program can be simplified further by using the WHETHER RECEIVED instead of SETTING a variable to the value received



5. Create the program to the right on computer 2.
 - a) Keep the same GROUP and ID numbers
 - b) The Variable (BLUE) can be removed completely from the program
 - c) Drag over a WHETHER RECEIVED and place one in each IF Statement
 - d) IF variable BLUE = LED ON, turn ON LED 1 (RED)
 - e) IF variable BLUE = LED OFF, turn OFF LED 1
 - f) Run both Programs.



If your set up did not work correctly the first time, what did you have to do to make it work?

CHALLENGE

SECTION 4 – PASS the LED SIGNALS (HOT POTATO)

Create a program that will allow BOTH Magic Boxes to cross communicate

Program both Magic Boxes with the same program

1. The BLUE and RED DUAL BUTTONS control LED 1 and LED 2 from the opposite Magic Boxes
2. The RED BUTTON turns LED 1 RED
3. The BLUE BUTTON turns LED 2 BLUE

If your set up did not work correctly the first time, what did you have to do to make it work?



CONCLUSION

1. *List two advantages to wireless communication between devices.*
 - A.
 - B.
2. *List two disadvantages to wireless communication between devices. Did you experience any?*
 - A.
 - B.
3. *In the challenge above, did you use any My Blocks? List them with bullet points.*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

1. Use My Blocks to make the challenge above simpler.
2. Have the Joystick attached to one Magic Box and the Micro Servo attached to the other. Use the Joystick to control the positions of the Micro Servo





- CHAPTER 3 - ROBOTICS & SENSORS (I/Os)

CHAPTER INTRODUCTION:

Imagine a robot who could sense the temperature of a part that it is working with or one that could tell the color of different objects. Imagine if a robot could tell that the object it was picking up was fragile, so it would pick it up gently. This is not science fiction and happens in factories around the world everyday by equipping robots and machinery in industry with sensors.

There is a wide array of sensors used for many different things and you will get a taste of how to use some sensors with your robotic arm. Some of the sensors used in this chapter include:

- PIR Sensor
- Sound Sensor
- Gesture Sensor
- Humiture sensor
- Potentiometer Sensor
- Color Sensor
- Light Sensor
- Photoelectric switch
- Joystick
- Microservo
- Limit Switch
- LED Module

TOPICS COVERED:

- Inputs Vs Outputs
- Analog Vs Digital Signals
- Sensor Applications



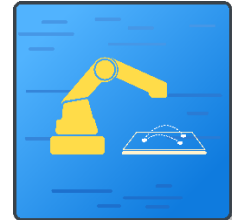
3.1 TP Using Digital Inputs

NAME: _____

Date: _____ Section: _____

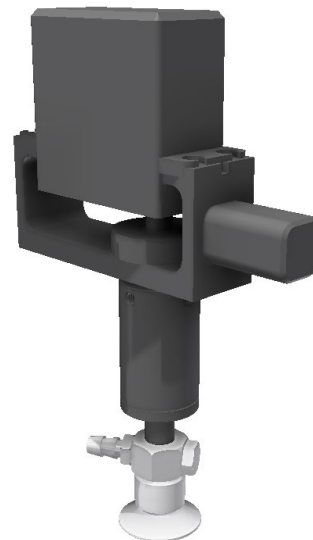
INTRODUCTION

Robotic arms need to be able to communicate with an operator so that you can tell it when to begin or end an operation. We can do this very easily with an input connected to the arm. This input can be something as simple as digital on/off switch.



In this activity, you will perform a pick and place with a robot arm using the air pump kit and a vacuum gripper or mechanical gripper. You will make the robot wait for an input on a switch before it starts, and eventually you will make the robot repeat this five times. Here is what you want your robot to do:

- Wait for the operator to hit the start switch
- Go home
- Pick up the object
- Place the object in another spot
- Return home
- Operator moves the part back to the start
- Repeat



Suction cup gripper attached to the rotational servo.



Remember: Always go to a position above the pick or place point first so that the robot places the object straight down, with no friction or interference. This will increase accuracy!

KEY VOCABULARY

- TCP
- Loop
- Pull Down Resistor
- Normally Open
- Suction
- Vacuum
- Digital
- Normally Closed

EQUIPMENT & SUPPLIES

- Dobot Magician
- Dobot Field Diagram
- ¾ inch wooden cylinders
- Servo extension cables
- Electrical tape
- Suction cup gripper
- Suction Cup or Mechanical Gripper
- DobotStudio software
- Digital Switch
- Jumper wire
- 3 prong male header pin
- Resistor 4.7KΩ to 15KΩ

PROCEDURE



Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS shutdown the Dobot before making connections or damage to the robot could occur.

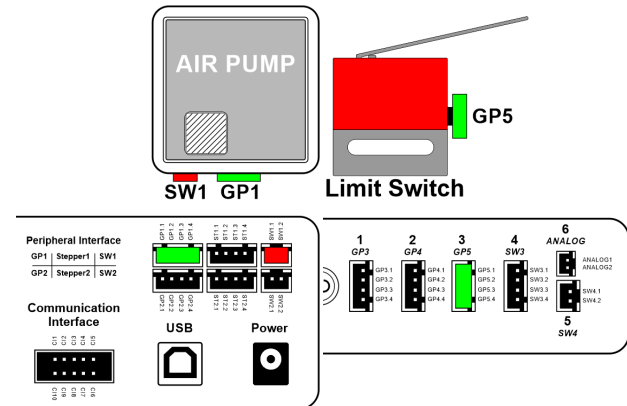
Caution: Please Refer to the [Dobot Input/Output Guide](#) for wiring your input. The setup for wiring inputs is not the same between the Magician V1 (White arm I/O port) and the Magician V2

1. Attach the Suction Cup or Mechanical Gripper as done in previous activities to the rotational servo and then attach it to the robot and plug in the wires as shown below.
2. Add the switch as an input.

Be sure to turn off power to the Dobot before continuing!



- Caution: Please Refer to the Dobot Input/Output Guide for wiring your input. The setup for wiring inputs is not the same between the Magician V1 and the Magician V2**



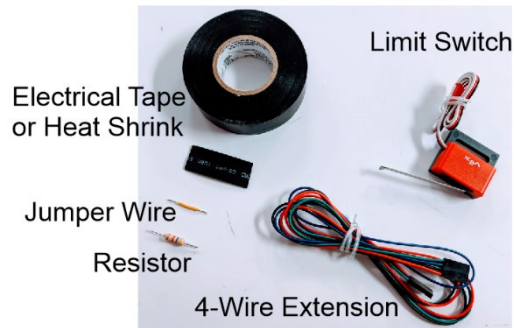
V2 = Colored plastic housings for inputs and outputs

This resistor allows the robot to communicate correctly with the input.

It will not work correctly without it.



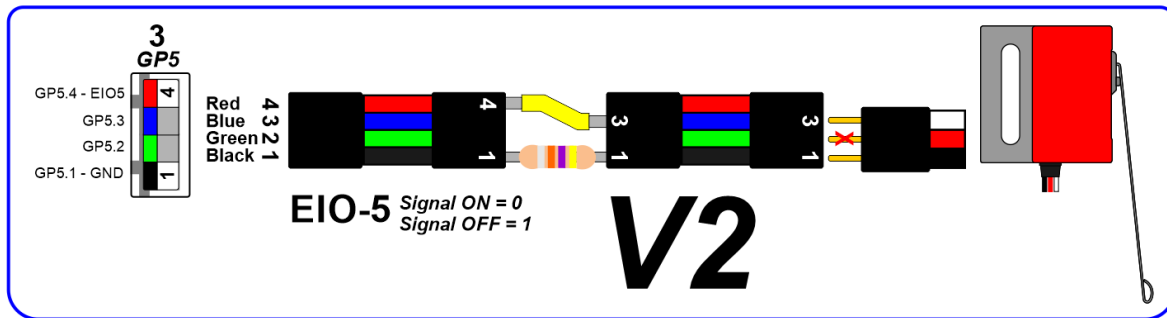
INPUT OPTION 1 - Direct Wire



INPUT WIRE PARTS LIST

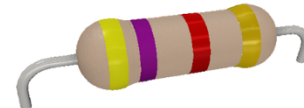
QTY	PARTS
2	4 - Wire Extension Cable
1	Short Solid Jumper Wire
1	Resistor 4.7kΩ to 15kΩ
1	Digital Limit Switch (SPDT or SPST)
	Electrical Tape or Heat Shrink

Example 4-Wire Extension Cable
*Color Sequence Can Vary



Step 1 - Install Jumper Wire and Resistor

*The resistor needs to be on the negative side to work as a pull down resistor



4.7k Ω - GROUND SIDE

YELLOW	VIOLET	RED	GOLD
4	7	X100	5%

Dobot Magician V2 (Colored IO Ports)

Step 2 - Secure the Components

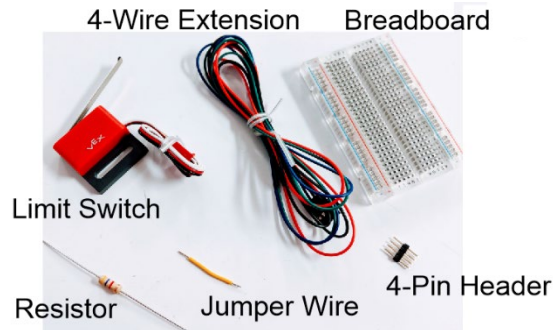
- Wrap the resistor and jumper wire with electrical tape or heat shrink to keep them from separating.
- Lightly tug on the assembly to ensure connection is stable



Step 3 - Install Digital Switch *Red wire is NOT active when using a VEX limit switch



INPUT OPTION 2 - Breadboard



INPUT WIRE PARTS LIST

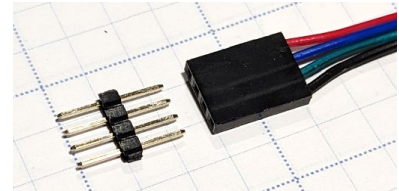
QTY	PARTS
2	4 - Wire Extension Cable
1	Short Solid Jumper Wire
1	Resistor 4.7k Ω to 15k Ω
1	Digital Limit Switch (SPDT or SPST)
1	Breadboard
1	4 Pin Male Header Pins

Example 4-Wire Extension Cable
*Color Sequence Can Vary



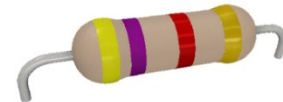
Step 1 – Install the 4-Pin Male Header

-Insert the Header pins into one end of the 4 wire extension cable. This will allow the 4-wire cable to be attached to the bread board.



Step 2 - Install Jumper Wire, Resistor, and Extension Cable

-Insert the resistor and jumper wire into the breadboard as shown in the illustrations below.



4.7k Ω - GROUND SIDE

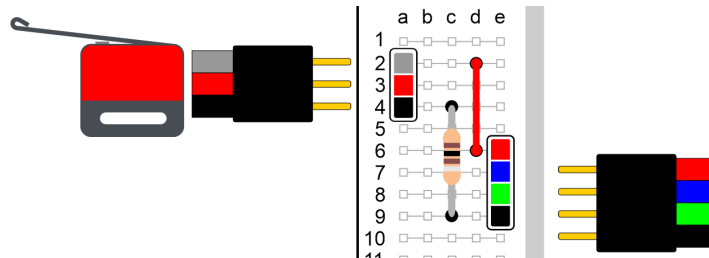
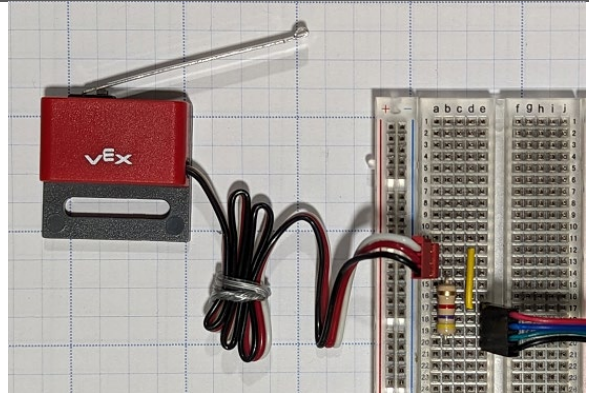
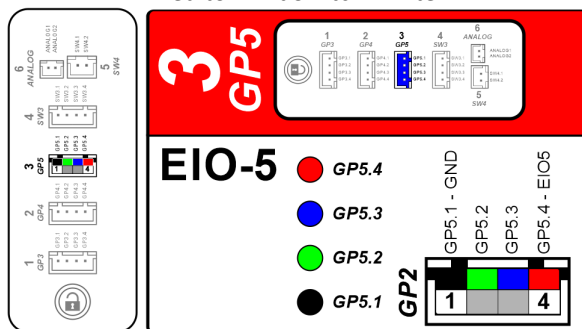
YELLOW	VIOLET	RED	GOLD
4	7	X100	5%

**Note that the 4-wire cable is not in the same orientation for both setups.*

Dobot Magician V2 (Colored IO Ports)

-Black to Limit Switch Black

-Red to Limit Switch White



www.chrisandjimcim.com V24

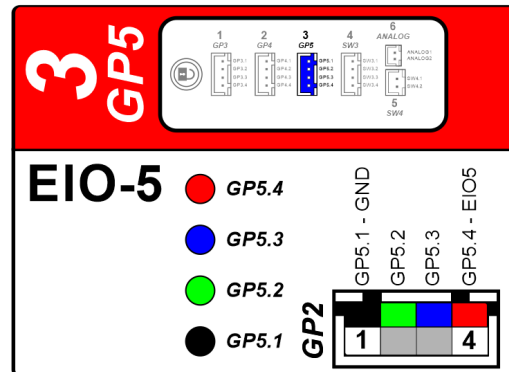
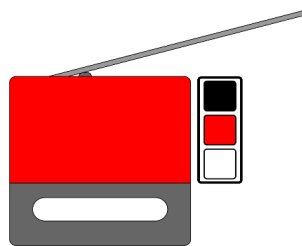
3.1 Block - Digital Inputs: Microswitch



Attach the 4-Wire Extension Cable to the Robot Arm

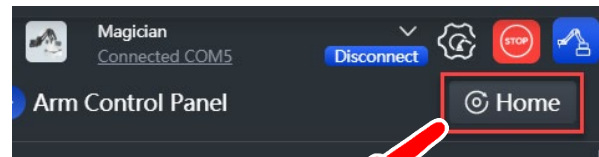
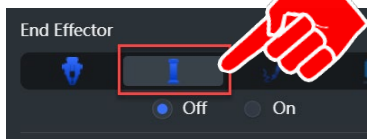


Example 4-Wire Extension Cable
*Color Sequence Can Vary



Be sure to consult the Dobot Input/Output Guide if you want to use other inputs and outputs, as damage to your robot or your other equipment may result.

- Open DobotLab software, open the Teach and Playback module, connect and home the robot, and be sure the *Suction Cup* is chosen.



- Next perform a pick and place with a wooden cylinder as you did in a previous activity. The only difference is that this time you will use the Suction Cup to pick and place the object. Be sure to use "above" positions.



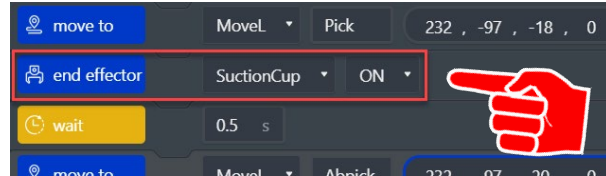


- Try to use Jump commands where appropriate.
- Try using the program you wrote for activity 2, and just edit it.
- If you do edit it be sure to use “Save As” when you are done and save with a different name.

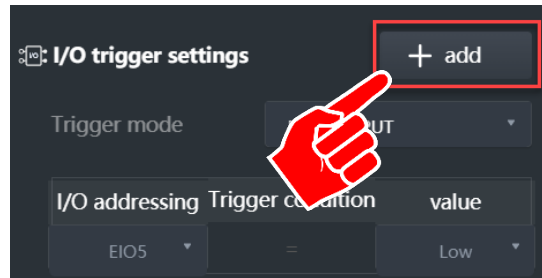
5. Save your program when finished. You may even be able to use a program you have already written and just touch up the points.

Be sure to have your instructor check when completed.

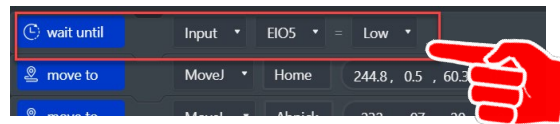
6. Be sure to turn on/off the suction when necessary.



8. Be sure to add EIO5 as an Input and be sure the value is set to **Low**. This will ensure that when you hit the switch (Turn it on) the robot will start the rest of the program.



9. Be sure to drag this line of code to the top of the program.



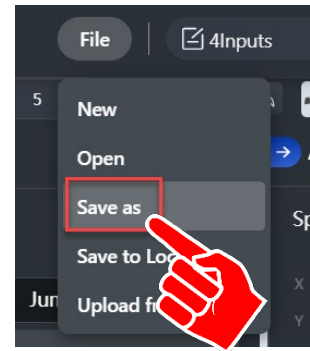
10. Hit the *Play* button to run your program and see what happens.

11. *Did it work the first time? If not, what did you have to change to make it work?*

12. Now let's make the robot do this five times, as you did in the last activity.

Be sure to have your instructor check when completed.

13. Be sure to save your work when done.



CONCLUSION

1. *What is the purpose of the resistor in this activity?*
2. *What might happen if you did not use this resistor?*
3. *What does it mean that the switch is DIGITAL?*
4. *What type of operation could this be simulating?*
5. *Is the switch you used Normally open, or Normally closed? How do you know?*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- | | |
|-------|--|
| _____ | 1. Look in the inputs and outputs manual and find another open input you might be able to use. Check with your instructor to be sure you choose an appropriate one, and then make the robot move the object to another predefined position when you hit that switch. |
| _____ | 2. Can you use something else as an input to the robot to make it do something? Check with your instructor and try something else. |



3.2 Block - Pick and Place with Inputs

NAME: _____

Date: _____

Section: _____

INTRODUCTION

Often it is necessary for a robotic arm to wait for another machine or process to finish before moving with its program. This can be done by adding the ability for a robot to read **INPUTS** values.

In this activity you will learn how to add an input to a basic **Pick and Place** operation in Blockly. Through this you will learn how to program the robot to move, turn on its suction cup, and set up **Inputs** in DobotBlock.



WARNING



Power OFF

Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS shutdown the Dobot before making connections or damage to the robot could occur.

KEY VOCABULARY

- Forever Loop
- Wait Until
- Set Port Mode – I/O
- Placeholder
- My Blocks
- Inputs
- Micro Wait
- Disable Block

EQUIPMENT & SUPPLIES

- Robot Magician
- Dobot Field Diagram
- 1" cubes or cylinders
- Digital Switch
- DobotLab software
- Suction Cup Gripper
- [Dobot Input/Output Guide](#)
- Various components for wiring the switch

ESSENTIAL QUESTIONS

- What's the difference between an *input* and an *output* in robotics?
- What are some examples of *digital inputs*?
- How do I code a digital switch as an *input* in DobotLab?

PROCEDURE



WARNING



Power OFF

Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

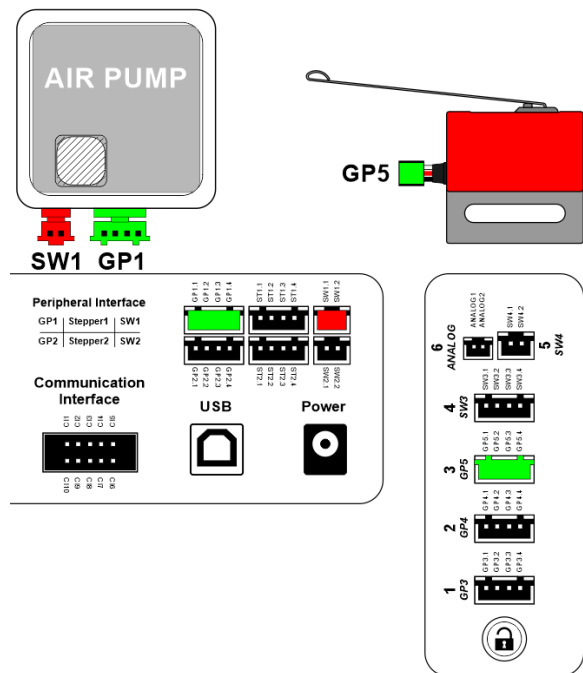
1. Print the - Pick and Place Field Diagram
2. Set up the robot with a suction cup and place a cube in one of the red squares on the field diagram provided
3. Wire the robot such that the LIMIT SWITCH is plugged into port GP5.

Caution: Please Refer to the Dobot Input/Output Guide for wiring your input.



WARNING

The setup for wiring inputs is not the same between the Magician V1 and the Magician V2



Before we get to the programming for this activity, we want to create an input wire with a limit switch and test the signal to ensure it is working.



Be sure to consult the Dobot Input/Output Guide if you want to use other inputs and outputs, as damage to your robot or your other equipment may result.

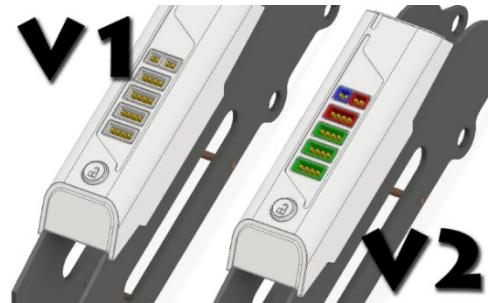


Be sure to turn off power to the Dobot before continuing!

In order to properly wire your Dobot Magician we will first need to identify which Dobot you have.

V1 = All white plastic housings for inputs and outputs

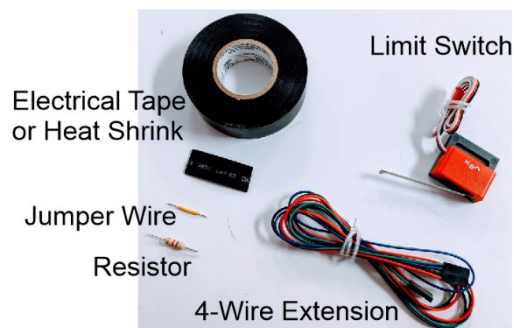
V2 = Colored plastic housings for inputs and outputs



The Dobot Magician V1 needs a **PULL DOWN RESISTOR** installed on the ground side of the servo extension cable to work properly. The pull-down resistor will insure a well-defined logical level in our logic circuit. This resistor will help eliminate possible float values and will pull the input pin to a logical low state in all conditions. The pull-down resistor can have any value from 4.7KΩ to 15KΩ. The larger the resistance, the slower the input pins response will be to voltage in milliseconds.

This resistor allows the robot to communicate correctly with the input. It will not work without it.

INPUT OPTION 1 - Direct Wire

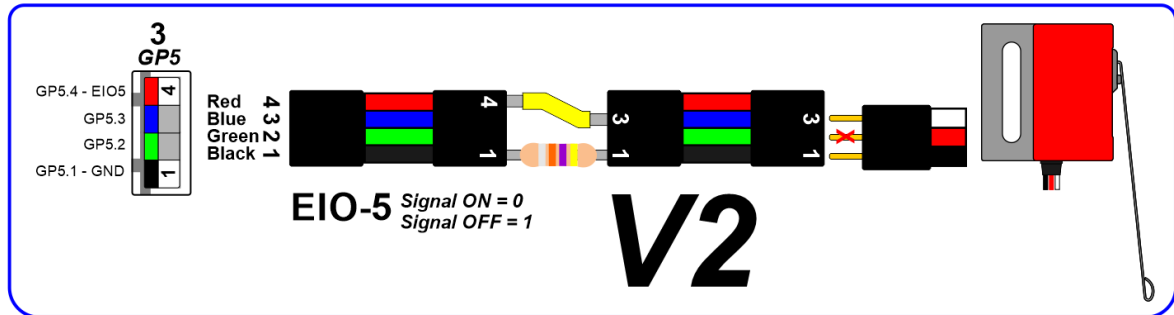


INPUT WIRE PARTS LIST

QTY	PARTS
2	4 - Wire Extension Cable
1	Short Solid Jumper Wire
1	Resistor 4.7kΩ to 15kΩ
1	Digital Limit Switch (SPDT or SPST)
	Electrical Tape or Heat Shrink

Example 4-Wire Extension Cable
*Color Sequence Can Vary





Step 1 - Install Jumper Wire and Resistor

*The resistor needs to be on the negative side to work as a pull down resistor



4.7k Ω - GROUND SIDE

YELLOW	VIOLET	RED	GOLD
4	7	X100	5%

Dobot Magician V2 (Colored IO Ports)

Step 2 - Secure the Components

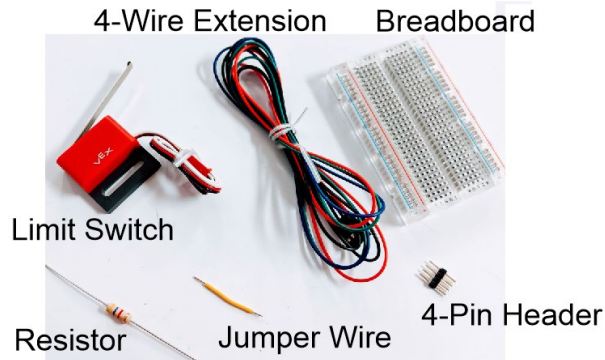
- Wrap the resistor and jumper wire with electrical tape or heat shrink to keep them from separating.
- Lightly tug on the assembly to ensure connection is stable



Step 3 - Install Digital Switch *Red wire is NOT active when using a VEX limit switch



INPUT OPTION 2 - Breadboard



INPUT WIRE PARTS LIST

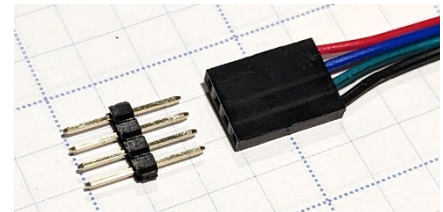
QTY	PARTS
2	4 - Wire Extension Cable
1	Short Solid Jumper Wire
1	Resistor 4.7kΩ to 15kΩ
1	Digital Limit Switch (SPDT or SPST)
1	Breadboard
1	4 Pin Male Header Pins

Example 4-Wire Extension Cable
*Color Sequence Can Vary



Step 1 – Install the 4-Pin Male Header

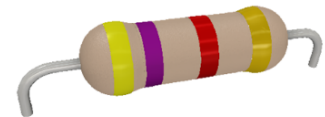
-Insert the Header pins into one end of the 4 wire extension cable.
This will allow the 4-wire cable to be attached to the bread board.



Step 2 - Install Jumper Wire, Resistor, and Extension Cable

-Insert the resistor and jumper wire into the breadboard as shown in the illustrations below.

**Note that the 4-wire cable is not in the same orientation for both setups.*

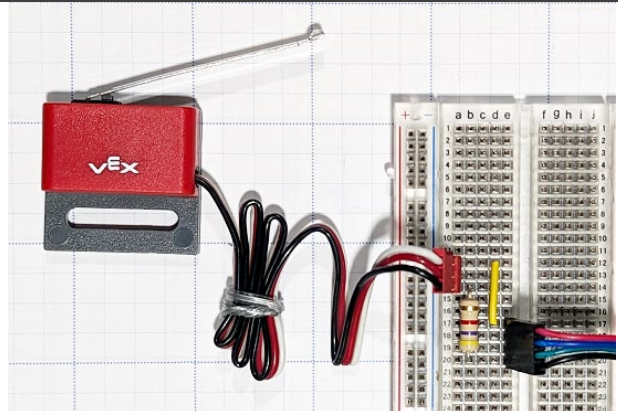
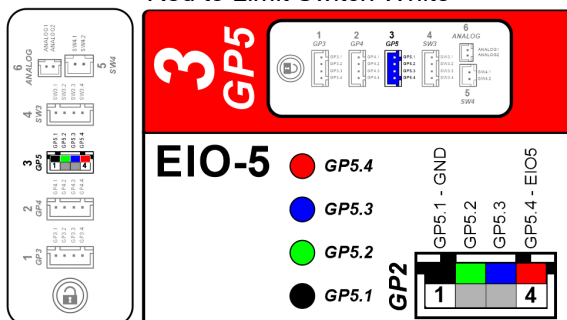


4.7k Ω - GROUND SIDE

YELLOW	VIOLET	RED	GOLD
4	7	X100	5%

Dobot Magician V2 (Colored IO Ports)

-Black to Limit Switch Black
-Red to Limit Switch White

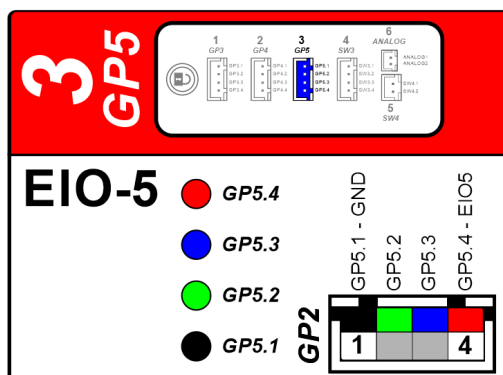
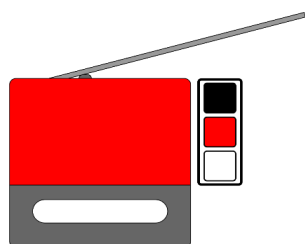




Attach the 4-Wire Extension Cable to the Robot Arm

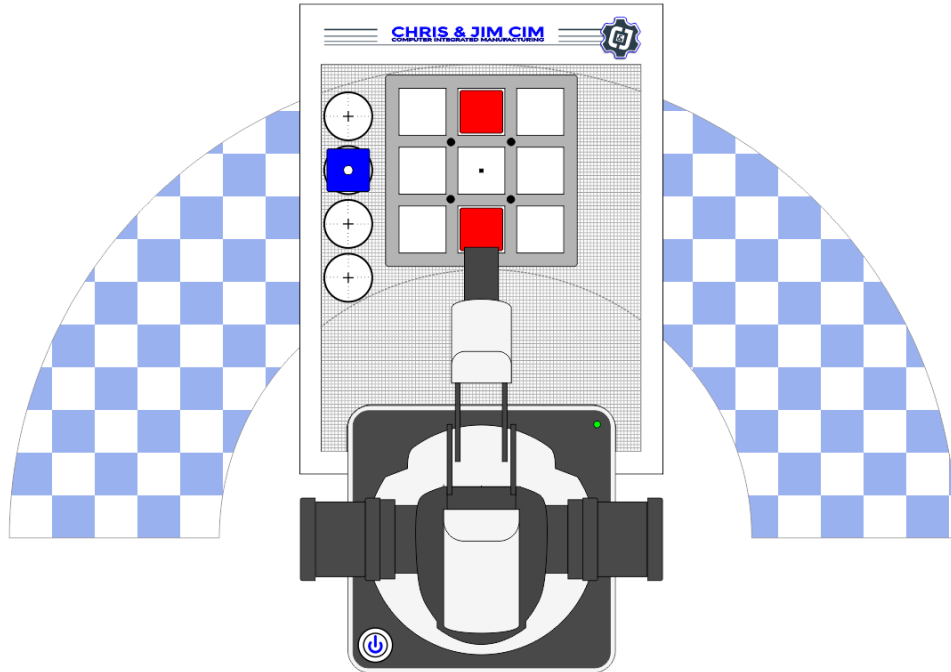


Example 4-Wire Extension Cable
*Color Sequence Can Vary

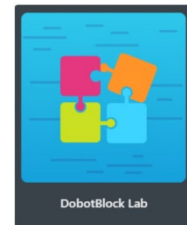


WARNING Be sure to consult the Dobot Input/Output Guide if you want to use other inputs and outputs, as damage to your robot or your other equipment may result.





4. Open up DobotBlock in the DobotLab software and Open your DobotBlock - Pick and Place with Jumps and Loops file from the previous activity.



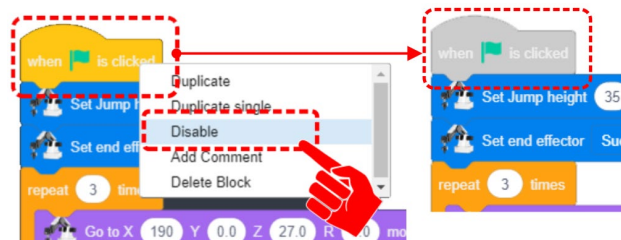
In this activity we are going to add the ability for the program to wait for an input and then have the program run whenever a limit switch is pressed. To do this, you must prepare a limit switch and plug it into GP5 (PORT3) (see above for instructions).



Power OFF

Make sure the robot is off before you plug or unplug anything from the robot.

Right click on the WHEN CLICKED CONTROL BLOCK and select Disable. This will grey out the selected block making the program ignore this portion of the code. When we want it re-enabled, right click it again and select **CANCEL**.



We disabled the initial program so that we could create a simple program just to test the limit switch signal.

Before we develop the program, we need to verify the switch is working as a digital input on EIO5. To do this, we will create a simple program that reports the state of the limit switch as a variable (on, off or 1, 0).

5. Drag over a new **WHEN STARTED**

6. Create a new **VARIABLE** called **INPUT STATE**

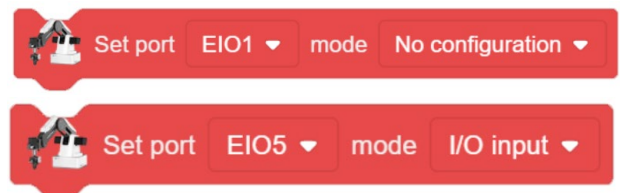
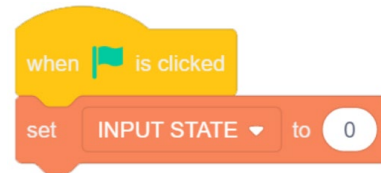
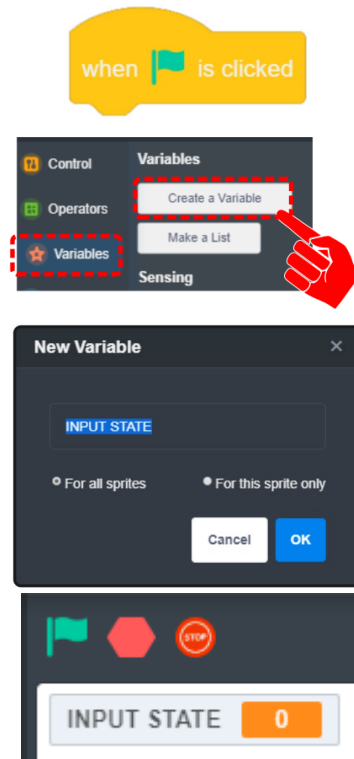
Make the **VARIABLE** “for all sprites”

The variable should now show in the top left corner of the Sprite window

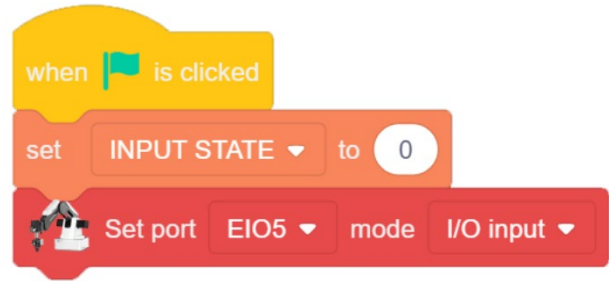
7. We will **SET** the initial value of the **INPUT STATE** to **ZERO**. Drag over a **SET VARIABLE** to step and attach it to our new **WHEN CLICKED**.

8. Next we need to teach the software that we will be using an input on **EIO5**

Change the port to **EIO5** and set the mode to **I/O input**



Connect the step to our new program



- Next we want to attach the input value to our variable value. There are several ways to program this. For our example, we will use REPEAT CONTROLS.

Drag over all of the parts you see to the right

- 1x REPEAT FOREVER
- 2x WAIT UNTIL
- 2x SET VARIABLE TO
- 2x GET DIGITAL SIGNAL
- 2x "=" From the OPERATOR Category



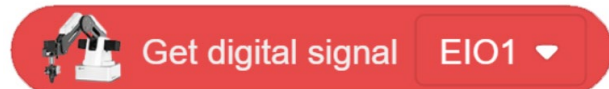
To test our signals, we want to create two separate conditions

- When the input is ON, set the variable to "1"
- When the input is OFF, set the variable to "0"

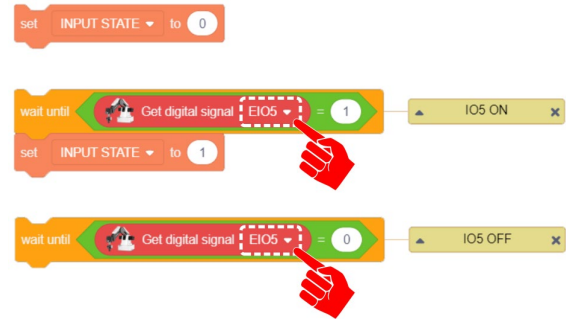
To do this, we will start combining the program pieces we just pulled over.

NOTE: The GET DIGITAL SIGNAL Blocks will look at the current state of the input. CHANGE the GET DIGITAL SIGNAL to EIO5.

We will start by setting the INPUT STATE VARIABLE to 0.



We will then wait for the STATE to change from a “0” to a “1”. Once it does we will reassign the variable to a.



We want this process to repeat forever (or until we stop the program).

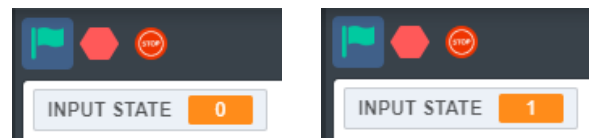
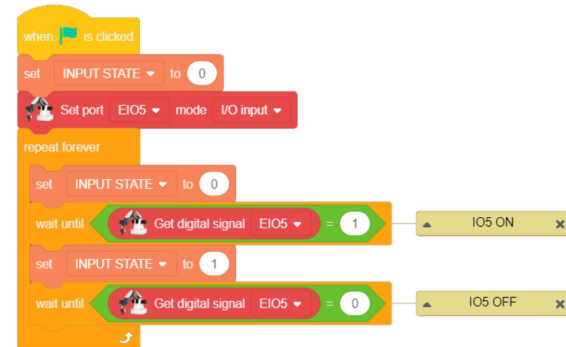


10. Attach our new **WHEN CLICKED**, our setup code, and our **INPUT MONITORING LOOP**.

Watch the state of the variable in the SPRITE window. It should be changing to match the value of your limit switch.

When the switch is pressed the signal should be a “1” and when released it should be a “0”

NOTE: If you have a V2 robot (Colored ports), these value will be backwards. We can adjust for this in our programming.



TYPICALLY:

NO (NORMALY OPEN) - If you see 0's that change to 1's, your switch or the Dobot is wired NO - Normally Open. The input will read Low or False until the switch is pressed.

NC (NORMALY CLOSED) - If you see 1's that change to 0's, your switch or the Dobot is wired NC - Normally Closed. The input will read high or true until the switch is pressed.



-We have found that some of the robots input ports read backwards (reads NC when it is correctly wired as NO) (0 when OFF, 1 when ON)? DON'T WORRY, just remember this when using the signals in your program.

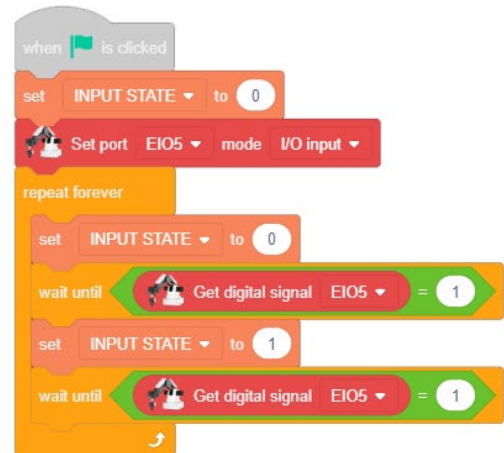
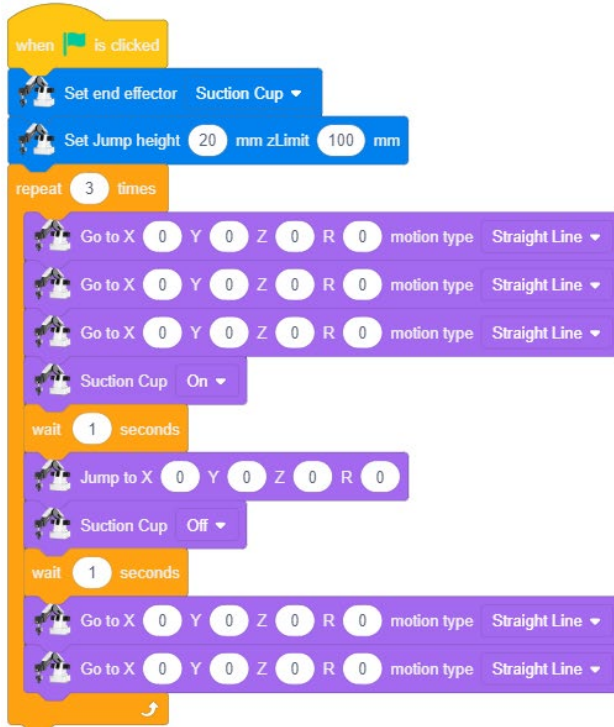
-If your variables are NOT changing at all, you may be plugged into the wrong port, have the wrong port selected for the IO setup, or have the wiring for the switch incorrect. Please refer to the Dobot Input/Output Guide. for your setup and try again.

Remember to turn the power off to the robot if you switch any wires.



If your setup did not work correctly the first time, what did you have to do to make it work?

Now that we are able to see the switch, we can disable our test program and re-enable the main program.



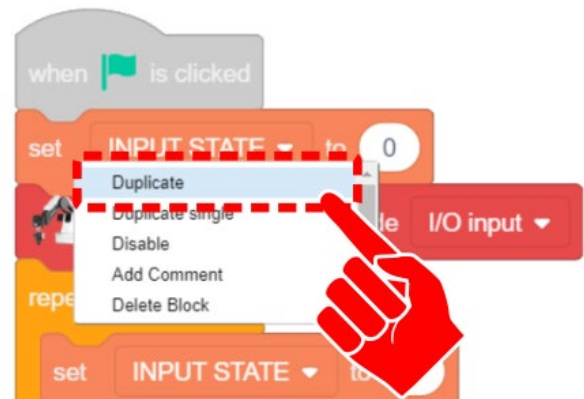
Our next step is to add the input switch into our PnP program.



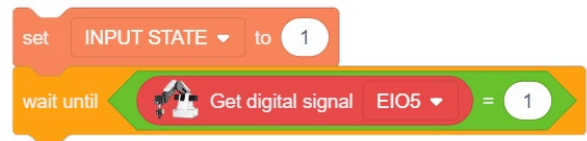
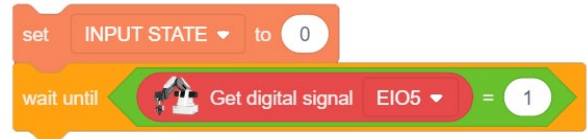
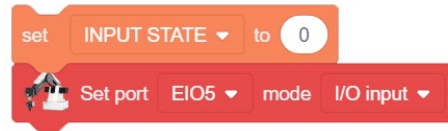
We suggest keeping the input testing program to debug your program if needed.... Just in case

11. Right Click on first line of code from the test program (Set INPUT STATE) and select DUPLICATE.

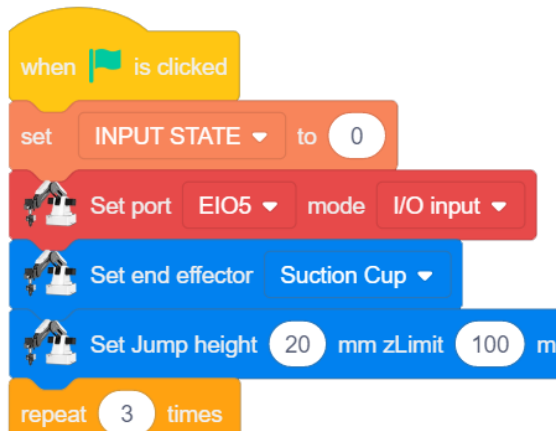
We will drag pieces of this program into our PNP program.



12. Pull apart the duplicated portion of the program and delete the FOREVER LOOP block



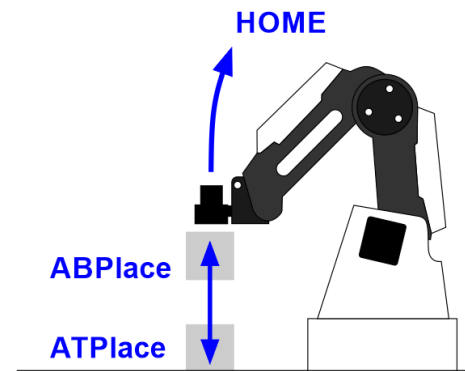
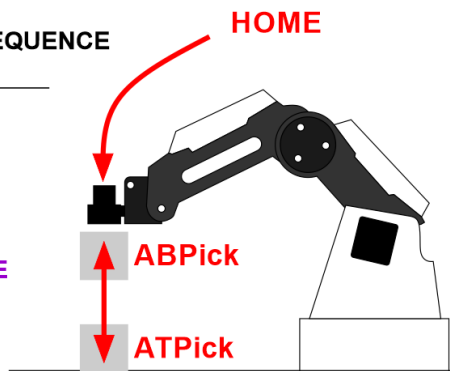
13. Drag and Drop the first section just below the when started. This will set the initial state of our variable to ZERO and add the port configuration to our main program.



Our next step is to decide when and where we would like the robot to look for the input of the limit switch. We will add the first limit switch control to go from our HOME position to the ABPICK. The robot will then Wait for an additional signal to get the part and move it to a new location.

PICK AND PLACE SEQUENCE WITH INPUTS

1. HOME
2. WAIT FOR SIGNAL
3. ABPICK
4. WAIT FOR SIGNAL
5. ATPICK
6. VACUUM ON
7. JUMP TO ATPLACE
8. VACUUM OFF
9. ABPLACE
10. HOME



Because we are waiting for the same signal twice, we will have better luck waiting until the input signal switch from a high (ON) signal to a low (OFF) signal (It has been pressed and released before we continue to the next step in the code. **What happens if the switch is stuck or never released?

For this process, we will need a few lines of code.



Remember that your values may read backwards from the input signal. Just swap the 1 and 0 for the WAIT UNTILS.

14. Create the block of code to the right using part of the code we duplicated from our test program.

Once we receive a signal (the limit switch is pressed), we will change our Variable to a 1 (ON).

We will then add a small micro wait to ensure the signal is stable as ON

Next we will wait for the signal to FALL (the limit switch is released). Once the signal falls, we will change the variable back to zero and move to the next line of code.

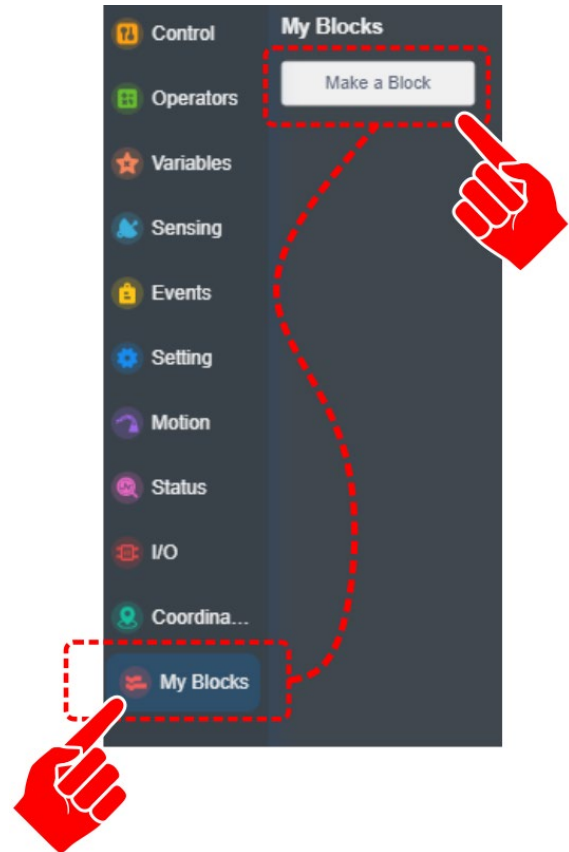


Since we will be repeating this exact same code twice, this is a perfect time to create our own block!

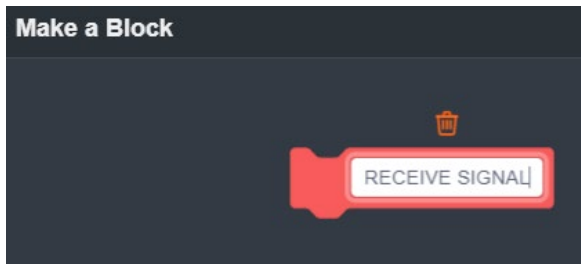
We are going to group all five lines of code into one code. Think of it as a mini program or a subprogram



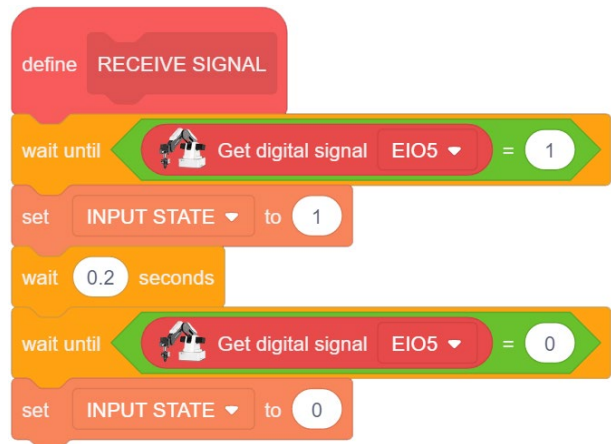
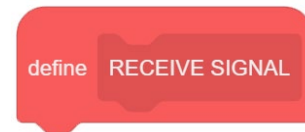
15. Select MY BLOCKS from the Toolbox and select MAKE A BLOCK



16. Give the BLOCK a name: "RECEIVE SIGNAL"



17. Once you select OK, a new header will appear on your screen. Drag the new header to the top of you WAIT FOR process.



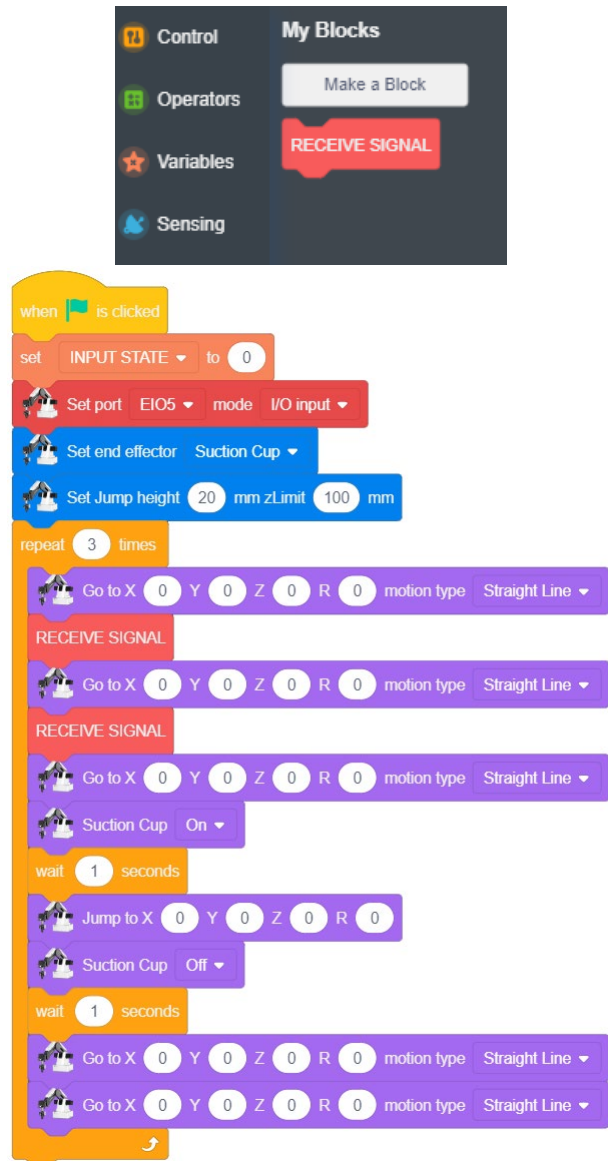
You should notice that there is a new block code available in your MY BLOCKS.

We can now add this one line of code anywhere in our program that we would like to look for our signal.

18. Add the new MY BLOCK to the program where needed

PICK AND PLACE SEQUENCE WITH INPUTS

1. HOME
2. WAIT FOR SIGNAL
3. ABPICK
4. WAIT FOR SIGNAL
5. ATPICK
6. VACUUM ON
7. JUMP TO ATPLACE
8. VACUUM OFF
9. ABPLACE
10. HOME



Once the program is written, run it and see if it works correctly. If it does not work, troubleshoot it until it does.

If your set up did not work correctly the first time, what did you have to do to make it work?



CONCLUSION

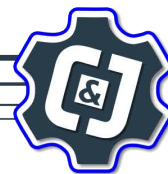
1. *What would happen if the loop detecting the switch wasn't in a forever loop? Try it and describe what happens. (Hint: Make sure to be holding the switch when you start the program)*
2. *What happens to the Pick and Place process when the switch is released (The loop condition becomes false)?*
3. *In your own words, define a nested loop.*
4. *In your own words, define a condition.*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- | | |
|-------|---|
| _____ | 1. Break the pick and place activity into multiple sections that will wait for the limit switch before performing each step. |
| _____ | 2. Make your own block (My Blocks) that tests and displays the Input state of a digital switch. Figure out how to change the Sprite to look like the robot you are using. |
| _____ | 3. Perform a three-tank dipping operation that dips it in a different tank each time the switch is pressed. |





3.3 - Magic Box - Intro to I/O Digital vs Analog

NAME: _____

Date: _____

Section: _____

INTRODUCTION –

Sensors are all around us helping control various forms of automation in our daily lives. They range from simple proximity sensors that activate automatic doors to the light sensors in your car that automatically turn on and off your headlights. These sensors provide the needed feedback that help control various devices.

Sensors in an industrial environment work in the same manner. They can inform an operator when a machine is low on coolant, or help add intelligence to a robotic arm that allows workers to safely work side by side with it in the same environment. This activity will explore how inputs and outputs work and how they may be programmed.



KEY VOCABULARY

- Sensors
- Inputs
- Outputs
- Magic Box
- Digital
- Analog
- Normally Open (NO)
- Normally Closed (NC)

EQUIPMENT & SUPPLIES

- Dobot Magic Box
- Digital - 2-Button Input Module
- Analog – Potentiometer Module
- LED Module
- Common Sensor Cables
- 12V Power Cable and Supply
- USB C to A Cable

KEY CONCEPTS

Essential questions answered in this activity include:

- What is the difference between an input and an output?
- What is the difference between a digital signal and an analog signal?
- How can a robot use input signals to provide feedback?
- What can robots use outputs for?
- Can a robot use multiple inputs and outputs at the same time? How?

PROCEDURE

SECTION 1 – WIRING the MAGIC BOX AND SENSORS



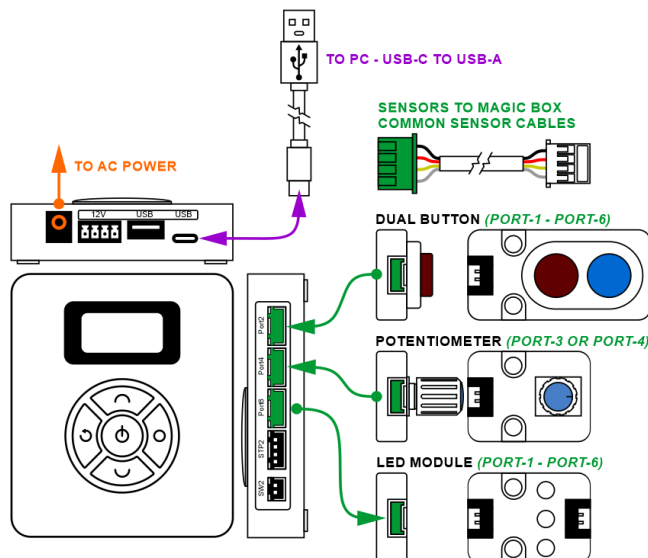
WARNING



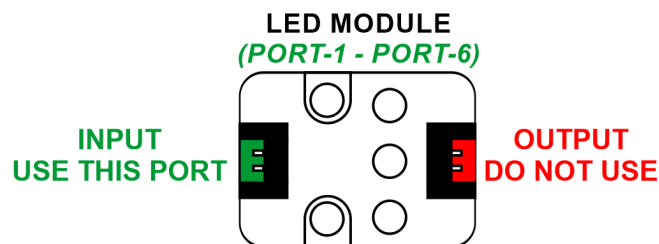
Power OFF

Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS shutdown the BOX before making connections or damage to the controller could occur. Be sure to ask your instructor if you have any questions.

1. For this activity, you will need:
 - 1x Magic Box
 - 1x USB-A to USB-C Cable
 - 1x AC/DC Power Adapter (12V)
 - 3x Common Sensor Cables
 - 1x Dual Button Sensor – **PORT 2**
 - 1x Potentiometer Sensor – **PORT 4**
 - 1x LED Module – **PORT 6**
2. Wire the Magic Box with sensors as shown.

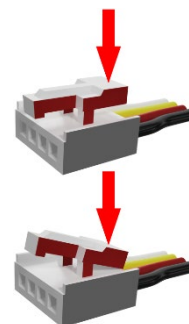


3. The LED has two ports, one is an input, and one is an output used to connect multiple devices together. *Be sure to plug it into the Magic box using the port that is farthest from the LED's.*



WARNING

Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.



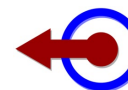
SECTION 2 – DEFINING INPUTS AND OUTPUTS

INPUTS – Defined as any data or signals that a robot receives that provides feedback about its environment. Information that is brought “into” the controller.

Examples: *Sensor Values, Signals from another Device or Robot, User Commands.*

OUTPUTS – Defined as an outgoing device or signal that is controlled by the robot. An output is often something that the robot can turn on or off light a light. An output can also be a value that is produced by the robot’s programming.

Examples: *LED, Sound, Motor ON/OFF, Motor Speed and Direction Control, Electromagnet, Vacuum Gripper.*



SECTION 3 – DEFINING DIGITAL & ANALOG INPUTS

DIGITAL INPUT/SIGNAL – Defined as a signal that can only be read in one of two states: YES or NO, ON or OFF, TRUE or FALSE, HIGH or LOW.

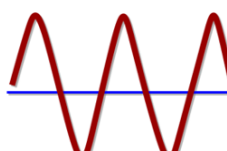
Digital sensors are often wired as either Normally Open (NO), or Normally Closed (NC). When a digital sensor is wired NO, no signal is received until the sensor is triggered. Once the sensor is active, it reports a high or true signal to the controller. Most often this signal is controlled by electric current or voltage (voltage present or not present). Sensors wired NC work in the opposite state.

Examples: Limit Switch, IR Photoelectric Sensor, Inductive Sensor

ANALOG INPUT/SIGNAL – Defined as a type of signal that has a range of values. This value is often affected by the intensity or magnitude of a measured external quantity (something found in the robot's environment like light or sound).

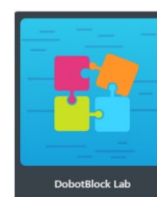
Examples: Potentiometer, Color Sensor, Humidity Sensor, Light Sensor

011011001
Digital


Analog

SECTION 4 – CONNECTING the MAGIC BOX to DOBOTLAB

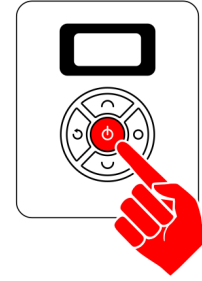
1. Open up DobotBlock Lab in the software.



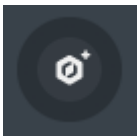
Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS turn it off before making connections or damage to the Controller could occur. Be sure to ask your instructor if you have any questions.



- Once all of the wiring is done (sensors connected), power ON the Magic Box
Dual Button Sensor – **PORT 2**
Potentiometer Sensor – **PORT 4**
LED Module – **PORT 6**



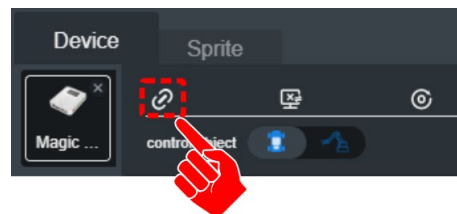
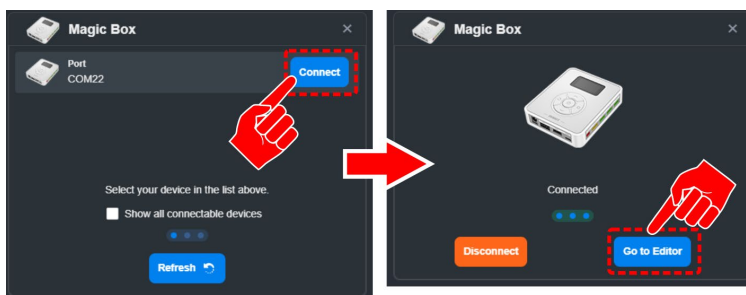
- You do not need a robot for this activity.** Delete both the Magician and Magician Lite Devices. Note: This will then default to the Sprite tab once there are no devices loaded.
- Click back on the devices tab and click on Choose a Device



- Select the Magic Box



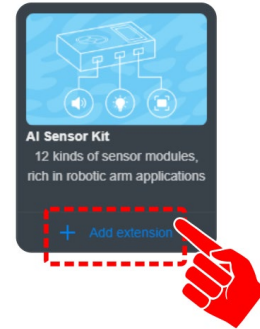
- Connect to the Device



- For this activity, we will need to add an EXTENSION to use the MAGIC BOX. Click the EXTEND icon in the bottom left corner.



- Click on "+ Add extension" for the AI SENSOR KIT.



SECTION 5 – EXPLORATION - DIGITAL INPUTS & with OUTPUTS

SKILL BUILDER 1 – LED MODULE – DIGITAL OUTPUT

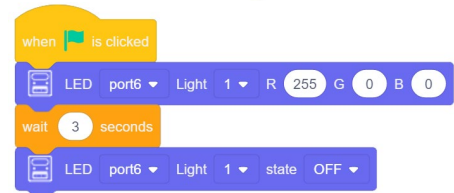
- The first device used in this activity is the LED OUTPUT MODULE.



- From the AI SENSOR KIT toolbox, LIGHT CATEGORY, drag over the LED RGB and LED STATE blocks.

Settings: PORT 6, LIGHT 1

Now turn on the FIRST LED as RED for 3 seconds, then turn off the LED STATE and end the program.



- The values to the right are examples of the values used to establish a specific color for the LED Module. The combination of R255 G0 B0 sets the color RED

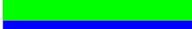
WHITE		rgb (255,255,255)
RED		rgb (255,0,0)
LIME		rgb (0,255,0)
BLUE		rgb (0,0,255)
YELLOW		rgb (255,255,0)

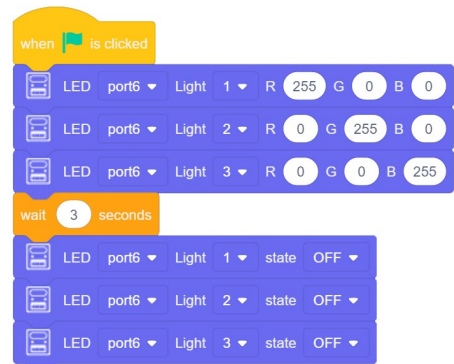


TEST your code, troubleshoot if necessary.

Be sure the LED module is plugged in correctly and the port is called out correctly.

- Next turn on all three LEDS, 1=RED, 2=GREEN, 3=BLUE for 3 seconds

WHITE		rgb (255,255,255)
RED		rgb (255,0,0)
LIME		rgb (0,255,0)
BLUE		rgb (0,0,255)
YELLOW		rgb (255,255,0)



If the program does not call for the LED Outputs to turn OFF at the end of the program, they will stay on even after the program has ended.

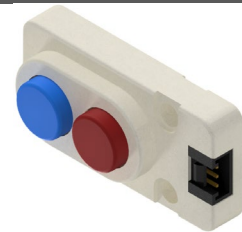


Once the program is completed, run it and see if it works correctly. If it does not work, troubleshoot it until it does.

If your set up did not work correctly the first time, what did you have to do to make it work?

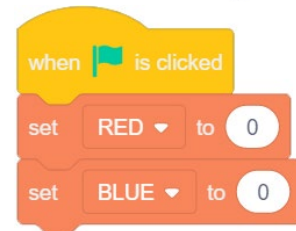
SKILL BUILDER 2 – DUAL BUTTON – DIGITAL INPUT

5. The second device in this activity is the DIGITAL DUAL BUTTON INPUT MODULE.

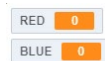
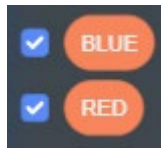


6. Since the DUAL BOTTON MODULE is an **Input** signal, we need to create a way to “SEE” the values that the sensor is reporting to the **Magic Box** (ON/OFF, 1/0)

Create TWO new variables (RED and BLUE) and set their initial values to ZERO



As long as there are check marks next to the variables you created, they should appear in the SPRITE window.



7. Next, setup two separate IF/ELSE Statements.

CONDITIONS: If the Button is pressed, set the Variable to ONE, else set it to ZERO.

8. From the AI SENSOR KIT toolbox, SENSOR CATEGORY, drag over the DUAL BUTTON MODULE blocks and place it inside one of the IF/ELSE Statements.

Settings: PORT 2, RED

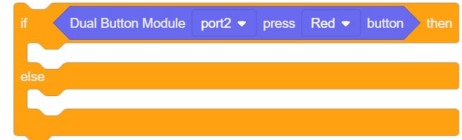
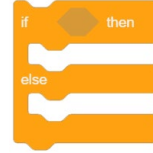
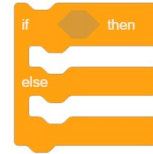
This will serve as the condition for this statement.
These sensors are wired NORMALLY OPEN (NO).
BUTTON PRESSED = TRUE (ON, 1)
BUTTON NOT PRESSED = FALSE (OFF, 0)

9. Next, setup what we want the IF STATEMENT to DO

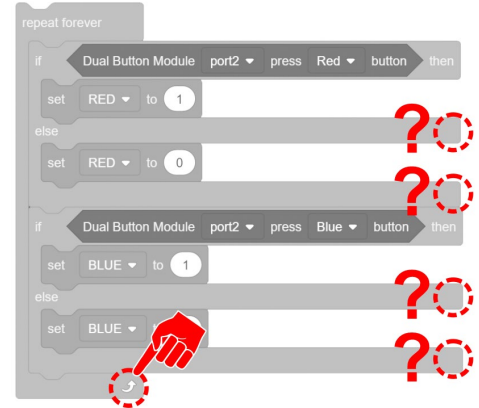
Drag the SET Variable block into both parts of the IF STATEMENT

STATEMENT IS TRUE = VARIABLE is 1
STATEMENT IS FALSE (ELSE) = VARIABLE is 0

10. Repeat this process for the other IF/ELSE Statement (or duplicate the first one and change the values)



11. You need to continuously evaluate these sensors to see when they are true and false. To do this, we will put them inside a forever loop



Notice that the IF/ELSE STATEMENTS do not have repeat arrows at the bottom of them. IF STATEMENTS will only be evaluated at the start of each loop. If they are true, they will run the entire process inside the IF portion, FALSE, everything inside the ELSE portion. IF STATEMENTS then move onto the next section of code (NO REPEAT). Both of our Statements do not have a perceivable time needed to run the code and move on, this will allow both buttons to be pressed at the same time and both return true conditions.

12. Add on the header code to finish the program

13. Add a **Micro Wait** inside the Forever Loop

Once the program is completed, run it and see if it works correctly. If it does not work, troubleshoot it until it does.



If your set up did not work correctly the first time, what did you have to do to make it work?



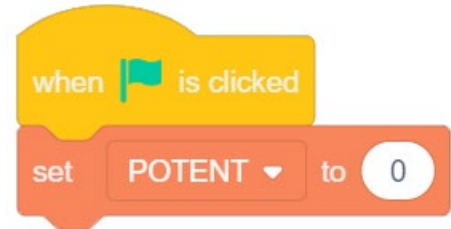
SKILL BUILDER 3 – POTENTIOMETER KNOB – ANALOG INPUT

14. The Third device used in this activity is the POTENTIOMETER KNOB INPUT MODULE. This device is an **analog input**.



15. Since the POTENTIOMETER MODULE is an Input signal, create a VARIABLE as a way to “SEE” the values the sensor is reporting to the magic box.

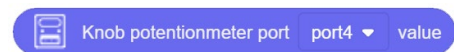
Create a new variables (POTENT) and set its initial values to ZERO. This will clear out any initial values for this variable.



16. Next inside a FOREVER LOOP block, we will assign the variables value to the sensors value.

From the AI SENSOR KIT toolbox, SENSOR CATEGORY, drag over the KNOB POTENT VALUE block.

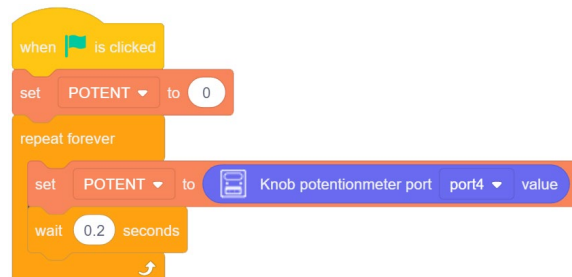
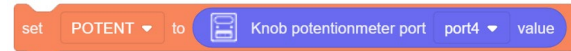
Settings: PORT 4



17. Drag the sensors value into the SET VARIABLE block.

This will allow the variable to update to whatever reading the sensor is currently reading... but only if it is in a FOREVER LOOP block.

Place a SMALL MICRO WAIT (0.2) after setting the variable to the sensors value before it repeats the loop. This will slow down the cycle and not pull the value from the sensor so quickly.



18. Once the program is completed, run it and see if it works correctly. If it does not work, troubleshoot it until it does.

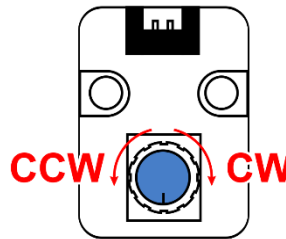
If your set up did not work correctly the first time, what did you have to do to make it



19. What was the Highest Value the sensor reported?

20. What was the Lowest Value the sensor reported?

21. Which direction **increases** the value (CW / CCW)



CHALLENGE

SECTION 6 – CONTROL OUTPUTS WITH INPUTS

Challenge Scenario:

- ALL of the conditions below are in the same program
 - When the BLUE button is pressed, LIGHT 1 will turn BLUE
 - When the RED button is pressed, LIGHT 3 will turn RED
 - The corresponding lights will be OFF when the corresponding buttons are **NOT** pressed
 - Divide the POTENTIOMETER High value into thirds –



***NOTE – Use GREATER THAN, LESS THAN, and possibly the AND OPERATORS for the next portion to create the needed conditions (Hexagon Shapes) for the analog sensor?*

- When the POTENTIOMETER value is in the bottom third or the top third, LIGHT 2 will be WHITE
- When the POTENTIOMETER value is in the middle third, LIGHT 2 will be YELLOW

Once the program is completed, run it and see if it works correctly. If it does not work, troubleshoot it until it does. SHOW YOUR INSTRUCTOR WHEN THE CHALLENGE IS COMPLETE

If your set up did not work correctly the first time, what did you have to do to make it work?



CONCLUSION

1. What type of values are reported from analog sensors?
2. What type of sensors would be found in an automated vehicle? List as many as you can think and identify them as analog or digital.
3. What type of sensors might a robot use/need to allow people to work near the robot or inside its work envelope?
4. What are the RGB values for

MAGENTA:



ORANGE:



GREY:

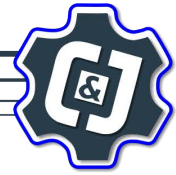


GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- | | |
|-------|---|
| _____ | 1. RED BUTTON is pressed, LIGHT TURNS RED, but only for 3 seconds (even if the button is still held) |
| _____ | 2. ALL three lights are off unless, the red button and the blue button are pressed and held as well as the potentiometer is at its lowest setting. If all three conditions are met, make all three lights turn magenta... until the three conditions are not met again. |
| _____ | 3. When the RED BUTTON IS PRESSED make the 3 LED's "chase one another to the right as RED. When the BLUE BUTTON IS PRESSED, make the LED's "chase" one another to the left 5 times as BLUE. |
| _____ | 4. When the POTENTIOMETER is in the FIRST THIRD OF ITS TRAVEL Turn on the LED TO RED. When the POTENTIOMETER is in the MIDDLE THIRD OF ITS TRAVEL Turn on the LED TO YELLOW. When the POTENTIOMETER is in the LAST THIRD OF ITS TRAVEL Turn on the LED TO GREEN. |





3.4 Magic Box - PnP

NAME: _____ Date: _____ Section: _____

INTRODUCTION

Adding sensors to a robot's programming increases its ability to be aware of various conditions that are in its environment. It also allows the programmer to build in the functions that can allow a user to manually take control of when operations should happen or should stop happening.

This activity has two main goals. The first goal is to teach the programmer how to safely wire the Magic Box to communicate data between the robot and the magic box. The second goal is to provide examples on how to program this communication to help provide sensor base feedback to the robot's functionality.



Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS shutdown the BOX before making connections or damage to the controller could occur. Be sure to ask your instructor if you have any questions.

KEY VOCABULARY

- Magic Box
- 2-Button Input Module

EQUIPMENT & SUPPLIES

- Dobot Magic Box
- Digital - 2-Button Input Module
- Additional Sensor (Your Choice)
- Pick and Place Template
- Magician Robot Arm w/ Suction Cup
- 12V Power Cable and Supply
- Common Sensor Cables
- USB C to A Cable

ESSENTIAL QUESTIONS

Essential questions answered in this activity include:

- How do I safely connect the Magic box to the Robot?
- Where do I connect the 2-Button input module on the magic box?
- How do I program in blockly to give the robot some sensor-based feedback?
- Which sensors would make a good "Start" in a blockly program? What makes them good for this?

PROCEDURE

SECTION 1 – WIRING the MAGIC BOX, SENSORS, & MAGICIAN ROBOT

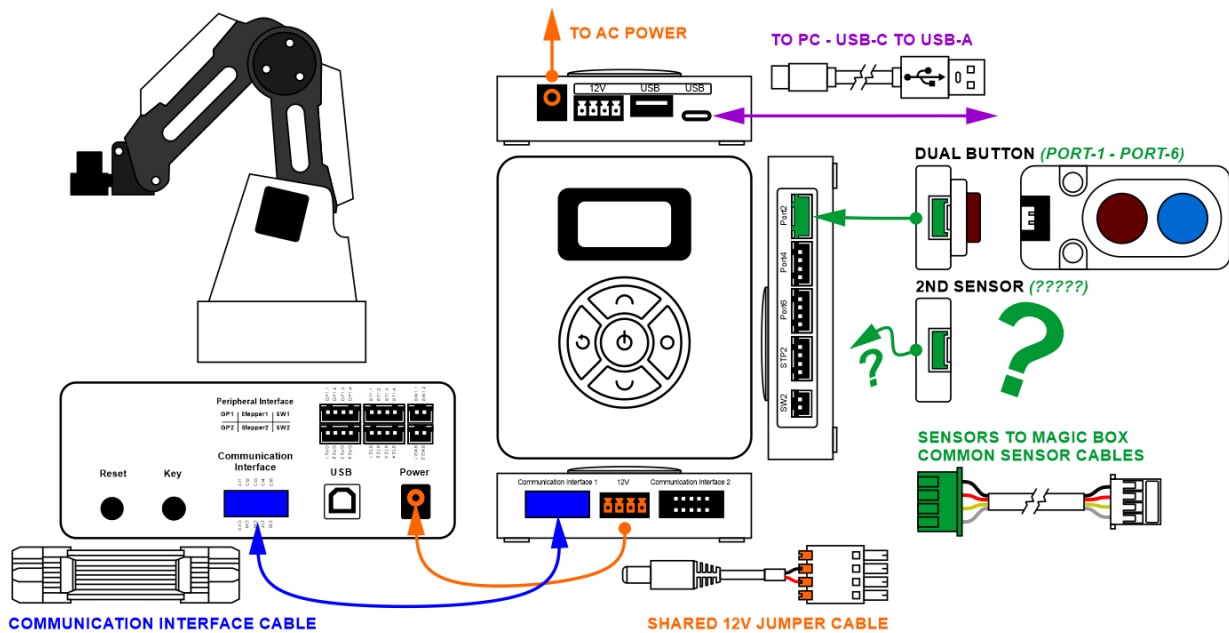


Caution: NEVER wire anything to the Magic Box or Robot while it has power on. ALWAYS turn them off before making connections or damage to the Controllers could occur. Be sure to ask your instructor if you have any questions.

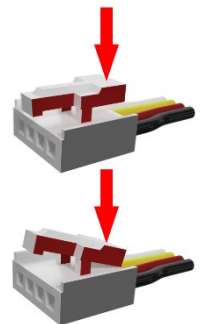
1. For this activity, you will need:

- 1x Magic Box
- 1x USB-A to USB-C Cable
- 1x AC/DC Power Adapter (12V)
- 2x Common Sensor Cables
- 1x Dual Button Sensor – **PORT 2**
- 1x Additional Sensor – **ANY AVAILABLE CORRESPONDING PORT**

2. Wire the Magic Box as shown below

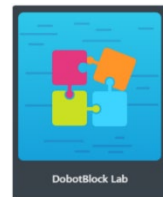


Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.

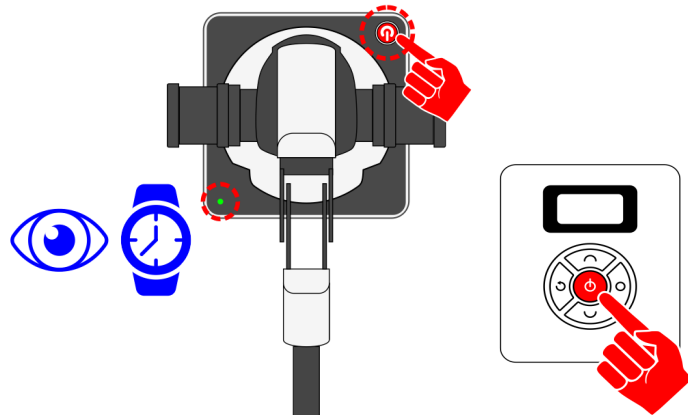


SECTION 2 – CONNECTING the MAGIC BOX & ROBOT to DOBOTLAB

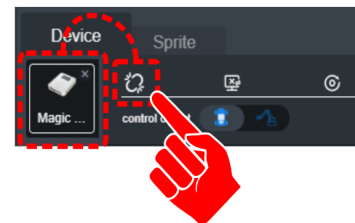
1. Open up DobotBlock Lab in the software.



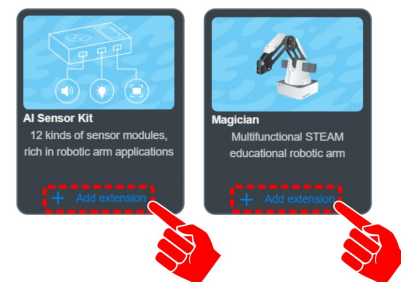
2. Once **ALL** of the wiring is done (sensors connected and robot arm connected), power ON the Magic Box and Robot. **WATCH** and **WAIT** for the robot to completely power on (GREEN INDICATOR)



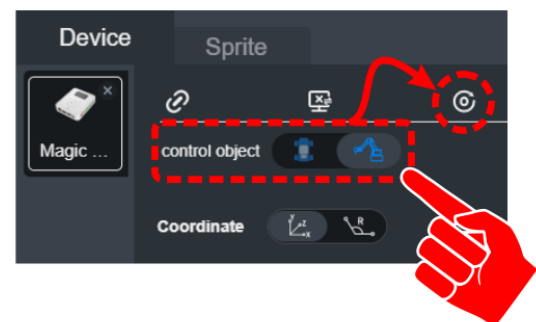
3. Follow the same process from previous activities to add the Magic Box as a device and connect the software to it (establish communication).



4. Click on "+ Add extension" for the AI SENSOR KIT & MAGICIAN.



5. Once the EXTENSIONS are added, slide the control object from the Magician GO to the Magician Robot.



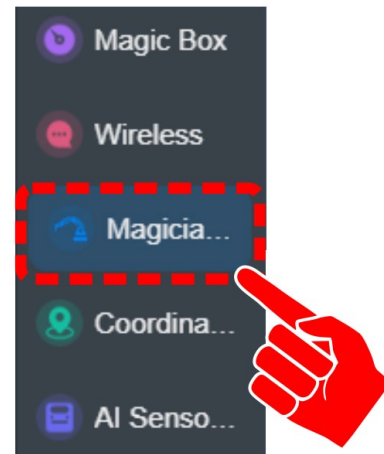
Ensure the robot's work envelop is clear!

6. Home to robot to check if communication is established between the magic box and the robot arm.





Notice that after adding the Magician Extension, a new category has been added to the toolbox (MAGICIAN PLUGIN)

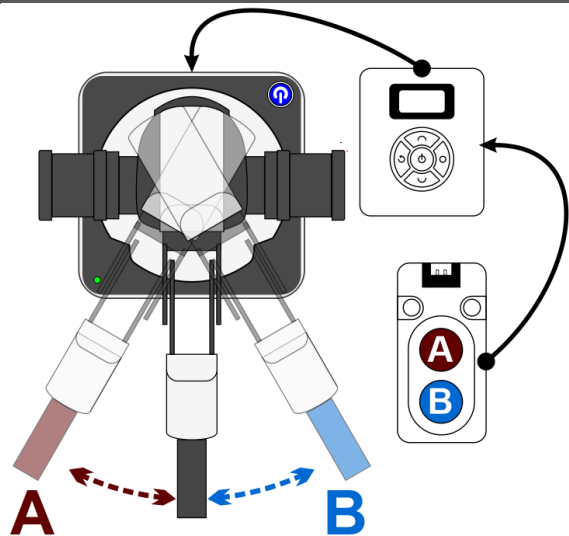


SECTION 3 – BASIC PROGRAMMING BETWEEN THE MAGIC BOX & ROBOT ARM

SKILL BUILDER – DUAL BUTTON AND MAGICIAN ARM

This activity will use the DUAL BUTTON input sensor to send the robot to different positions.

- When no button is pressed, the robot will go to or remain at the home position.
- When Button “A” is pressed, the robot will go to a position in the negative direction.
- When Button “B” is pressed, the robot will go to a position in the positive direction.
- The robot will stay at a position if the button is held



- Drag over three GO TO POSITIONS from the NEW category.

- Find the coordinates and fill them in.



Either drive the robot to the desired positions or use the Unlock Arm Button. Either type in the position numbers or use the fill coordinates option.

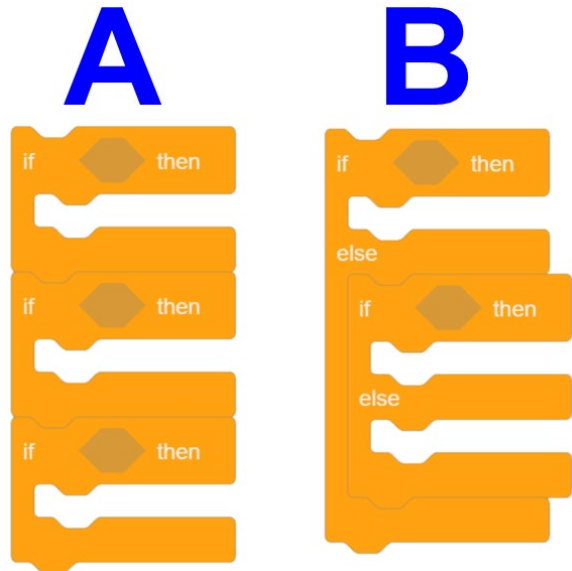


The robot needs to be able to make a choice of which position it should be at.

There will be three possible locations for the robot to choose from: HOME, NAGATIVE, or POSITIVE.

While there are several ways to program this process, this activity will look at two of them.

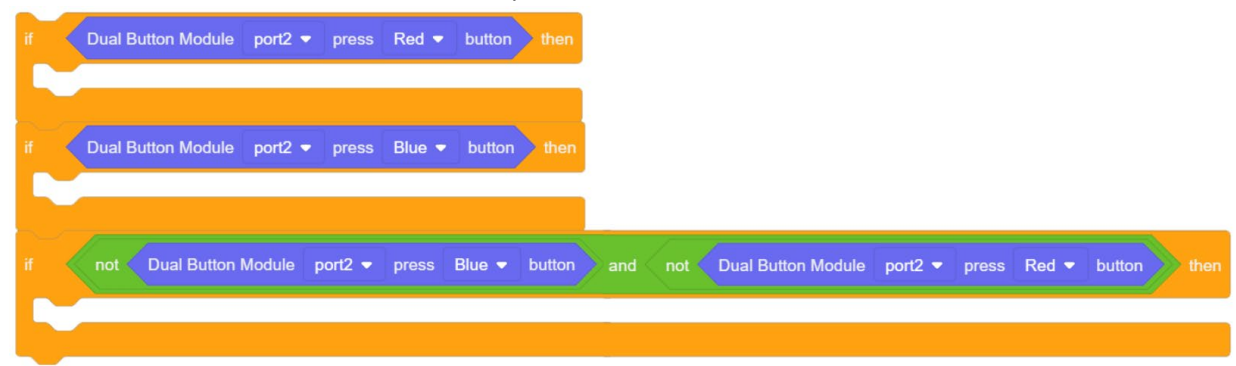
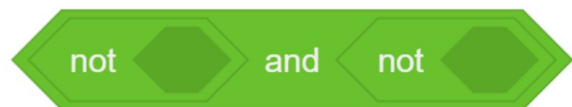
- a) A group of individual IF Statements
OR
- b) A nested set of IF Statements for this process.



OPTION A - A group of individual IF Statements

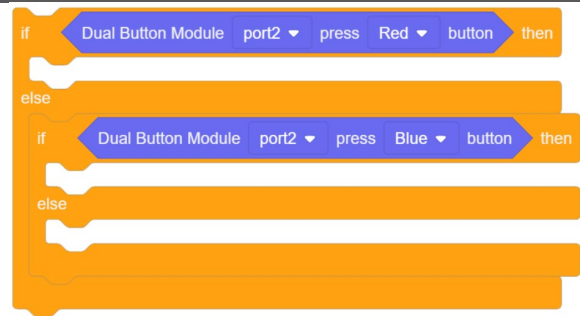
The condition of the DUAL BUTTON will be placed inside the first two IF Statement. The third IF Statement will instruct the robot on what to do IF the buttons are not pressed

Option A provides the programmer the opportunity to build a complex computation for the third condition. The condition should only be met if the RED button and the BLUE button are NOT pressed.



OPTION B - A NESTED SET OF IF STATEMENTS

- The first condition of the DUAL BUTTON (RED) will be placed inside the IF Statement.
- The second condition of the DUAL BUTTON (BLUE) will be placed inside the ELSE IF Statement.
- The third condition false into the final ELSE condition (IF neither condition is TRUE, do this).

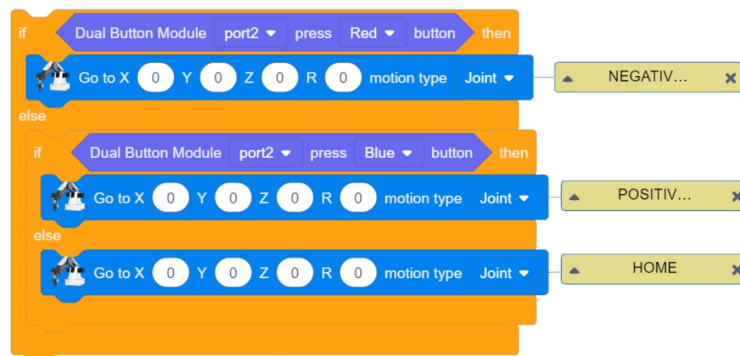


This option simplifies the code and places the choices in a hierarchy (order of importance) that also takes care of the fourth possible condition... BOTH the RED button and BLUE button are pressed at the SAME time. The RED condition will override the BLUE condition in this format.

ADD IN THE MOVEMENT COMMANDS

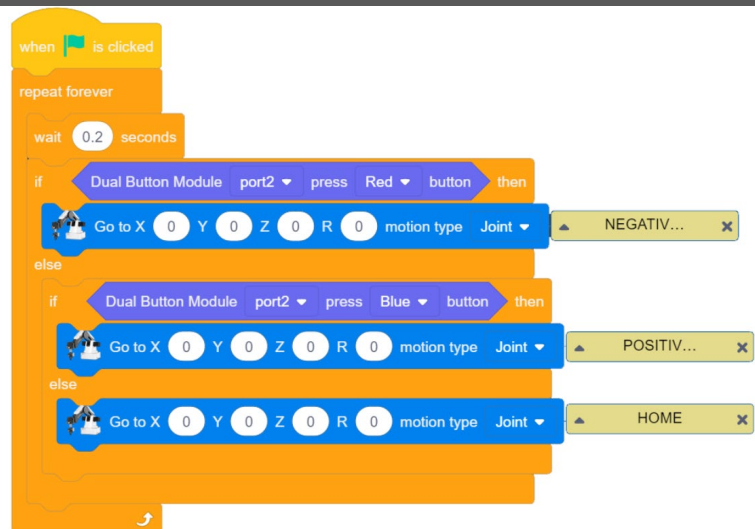
- Regardless of which option is used, the GO TO POSITION blocks can now be added to each of the Statements.

EXAMPLE:



ADD ADDITIONAL CODE

- Add additional code to the program to complete it for testing.
 - When Started – Start the process.
 - Forever Loop – Loop/Rewind and repeat the process.
 - Small Micro Wait – Small wait to slow the process of looking for the sensor values.
 - If Nest - Look for conditions, Run conditions if TRUE or Run ELSE.



- Once the program is completed, run it and see if it works correctly.



If your set up did not work correctly the first time, what did you have to do to make it work?

CHALLENGE

SECTION 4 – USE A DIFFERENT SENSOR(S) TO CONTROL THE ROBOT'S MOVEMENT THROUGH A PICK AND PLACE ROUTINE

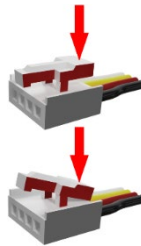
For this challenge, choose a different sensor from the kit that you have practiced with that will fit this challenge.



REMINDER: If the sensor is not already attached to the Magic BOX or a sensor is being removed, make sure it is powered off before

installing the new sensor to keep from damaging the controller.

If a sensor is being uninstalled or removed, make sure the tab is completely pressed to keep from damaging the sensor.

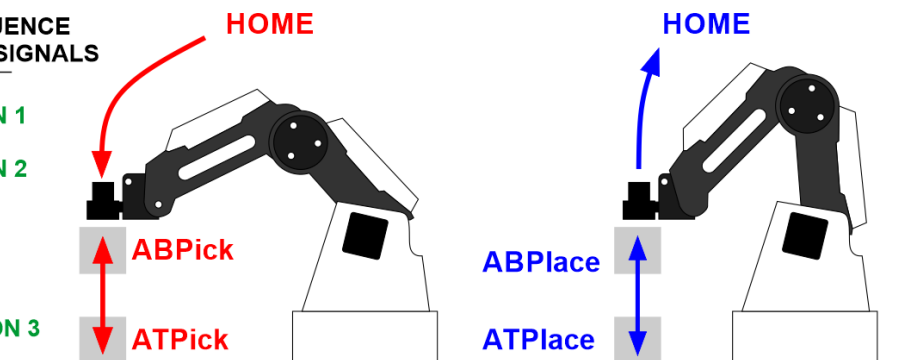


Pick and Place Challenge.

- The robot will start at a home position and wait for an input signal before moving to the ABOVE PICK location.
- The robot will wait at the ABOVE PICK location until a signal (different signal?) is received to pick up the block and jump it over to the AT PLACE location.
- The robot will sit at the ABOVE PLACE location until another signal (different signal?) tells it to move home.
- The process should then be looped and ready for another loop.

PICK AND PLACE SEQUENCE WITH MAICBOX INPUT SIGNALS

1. HOME
2. WAIT FOR CONDITION 1
3. ABPICK
4. WAIT FOR CONDITION 2
5. ATPICK
6. VACUUM ON
7. JUMP TO ATPLACE
8. VACUUM OFF
9. ABPLACE
10. WAIT FOR CONDITION 3
11. HOME



If your set up did not work correctly the first time, what did you have to do to make it work?

CONCLUSION

1. *Give a reason why the Dual Button module would be useful in industry and explain your answer.*
2. *What sensor did you choose to start the challenge activity? Justify your choice with at least three bullet points.*
3. *What other sensor might be a good choice? Why?*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- | | |
|-------|--|
| _____ | 1. Use the JOYSTICK to control the robot's movements through a pick and place operation. Use a digital input to turn on and off the vacuum gripper. |
| _____ | 2. Use the Dual-button module to complete the following task: <ol style="list-style-type: none">a. Place a block to be picked up.b. If the block is red, have the operator press the red button and drop it into a bin for red blocks.c. If the block is blue, have the operator press the blue button and drop it in the blue block bin.d. If the block is any other color, drop it in a garbage bin by using a different input.e. Repeat this forever. |



3.5 Magic Box - Exploring I/Os

NAME: _____

Date: _____

Section: _____

INTRODUCTION

In this activity you will get a chance to use many other types of sensors, inputs, and outputs with the magic box. These include:

- PIR Sensor
- Sound Sensor
- Gesture Sensor
- Humiture sensor
- Light Sensor
- Photoelectric switch
- Joystick
- Microservo



KEY VOCABULARY

- Input / Output (I/O)
- Pulse width modulation (PWM)
- UART
- Analog/Digital
- Analog to digital conversion (ADC)
-

EQUIPMENT & SUPPLIES

- Dobot Magic Box
- Micro Servo
- Photoelectric Sensor
- Joystick
- Light Sensor
- Sound Sensor
- PIR Sensor
- Gesture Sensor
- Humiture Sensor
- Common Sensor Cables
- 12V Power Cable and Supply
- USB C to A Cable

KEY CONCEPTS

Essential questions answered in this activity include:

- What does a PIR sensor do, and how is it used?
- What does a Sound sensor do, and how is it used?
- What does a Gesture sensor do, and how is it used?
- What does a Humiture sensor do, and how is it used?
- What does a Light sensor do, and how is it used?

- What does a PIR sensor do, and how is it used?
- What does a Photoelectric switch do, and how is it used?
- What can a joystick be used for in robotics?
- What is a micro servo and how is it used in robotics?

PROCEDURE

SECTION 1 – WIRING the MAGIC BOX AND SENSORS – 1st SETUP



WARNING



Power OFF

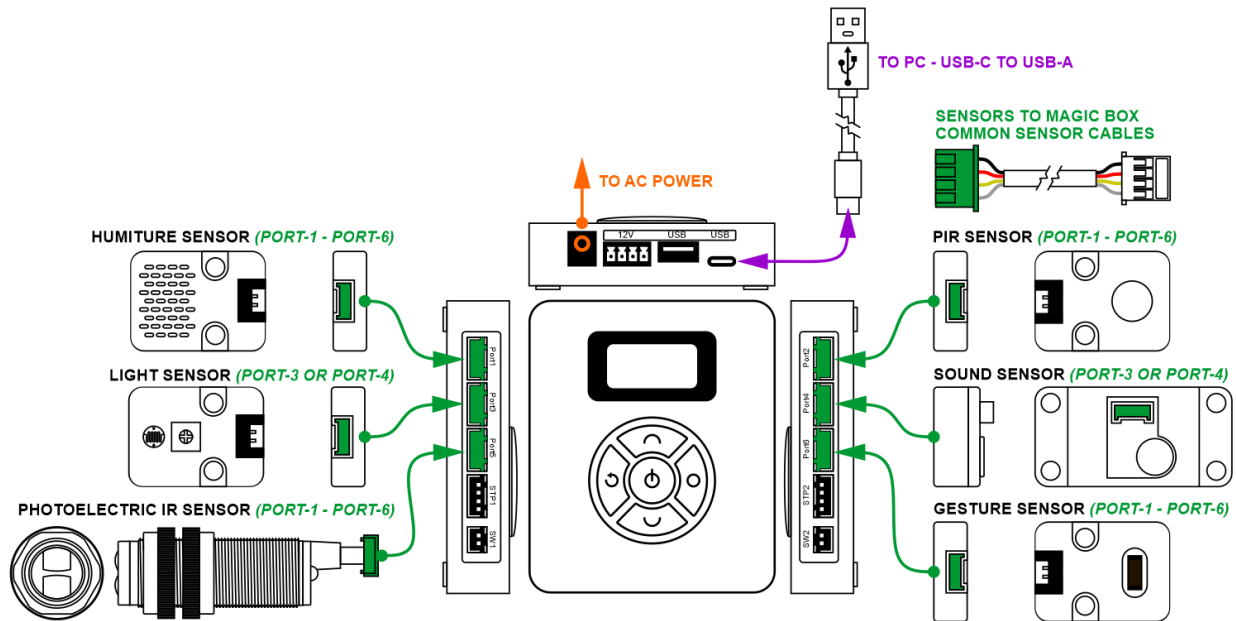
Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS turn it off before making connections or damage to the Controller could occur. Be sure to ask your instructor if you have any questions.

SETUP #1

1. For this activity, you will need:

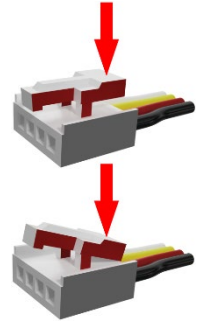
- 1x Magic Box
- 1x USB-A to USB-C Cable
- 1x AC/DC Power Adapter (12V)
- 5x Common Sensor Cables (Photoelectric Sensor does not need a common sensor cable)
- 1x PIR Sensor – **PORT 2**
- 1x Sound Sensor – **PORT 4**
- 1x Gesture Sensor – **PORT 6**
- 1x Humiture Sensor – **PORT 1**
- 1x Light Sensor – **PORT 3**
- 1x Photoelectric IR Sensor – **PORT 5**

2. Wire the Magic Box as shown below:



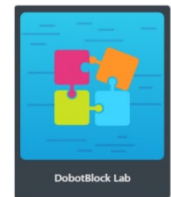


Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.

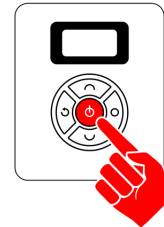


SECTION 2 – CONNECTING the MAGIC BOX to DOBOTLAB

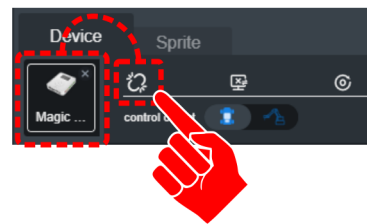
1. Open up DobotBlock Lab in the software.



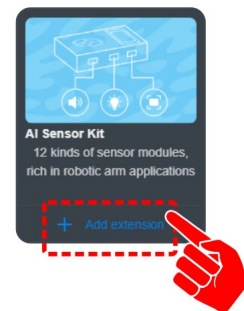
2. Once all of the wiring is done and all of the sensors connected, power ON the Magic Box



3. Follow the same process from previous activities to add the Magic Box as a device and connect the software to it (establish communication).



4. Click on “+ Add extension” for the AI SENSOR KIT.



SECTION 3 – SETUP #1 - EXPLORING ADVANCED SENSORS

- **PIR, SOUND, GESTURE, HUMITURE, LIGHT, PHOTOELECTRIC**

SKILL BUILDER 1 – PIR SENSOR – DIGITAL INPUT

1. The first sensor explored in this activity is the PIR SENSOR. In the software it is referred to as a **human** sensor.

The PIR sensor is a body infrared sensor. It works as a passive pyroelectric infrared detector and detects the infrared radiation emitted or reflected by the human body or other objects. When the sensor detects infrared it outputs a **TRUE** signal. It then carries out a time delay (during which the true signal is maintained and repeated triggering is allowed) until the triggering signal disappears (restoring **FALSE** signal). Note: There is a two-second delay after each detection is triggered.

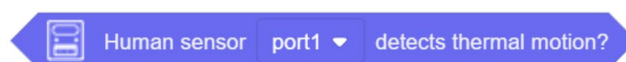
ANY GREEN
PORT

PORT 1-6



2. Drag over a Human Sensor Detects block from the AI SENSOR KIT toolbox

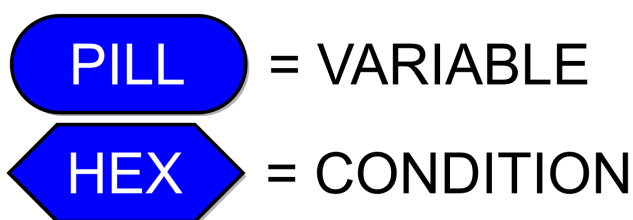
This block will produce a TRUE result if thermal motion is detected



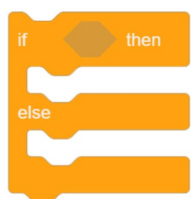
3. Change the PORT value to the correct port (see page 2).

4. A few simple lines of code are needed to retrieve values from the sensors used throughout this activity.

Some of the sensors can be tied to a **variable** (PILL shaped) and some will need to be used as a **condition** in a loop (HEX shaped)



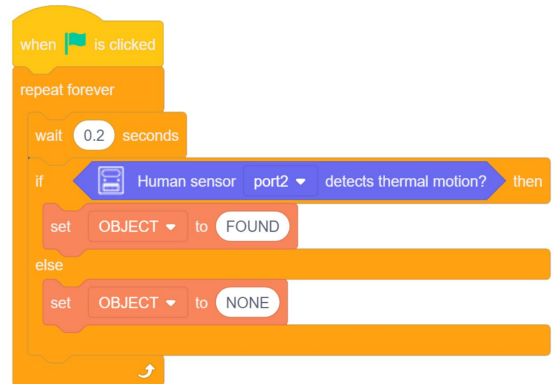
5. This sensor is a digital sensor. It will only report a TRUE/HIGH or FALSE/LOW signal. The values for this sensor are reported by a HEX shape. Create a variable (OBJECT) and an IF/ELSE Statement. A TRUE condition will change the variable OBJECT to FOUND ELSE (not true) the variable will show as NONE



DIGITAL SIGNALS	
TRUE	FALSE
YES	NO
1	0
ON	OFF
HIGH	LOW



6. Place everything inside a forever loop with a small micro wait.
7. Download the code to the Magic Box. Practice with this group of code and record your results below.



A. What type of range does the sensor have? (How far can the sensor detect motion?)

B. What type of vision does the sensor have? (Different angles, all the way around?)

C. What type of motion does the sensor detect? (side to side, forward backward, rotation?)

D. After the object is still (no motion) for more than two seconds, what happens?



E. Can the sensor detect motion of an object?

-Ball rolling, does it have to be large (marble?)

-Something tossed across its field of view (did it detect the object or you tossing the object)

F. What happens if the motion is smooth and continuous (rotating fan or oscillating fan, back & forth)

G. Can you or an object move slow enough to keep the sensor from reading TRUE?

SKILL BUILDER 2 – SOUND SENSOR – ANALOG INPUT

1. The second sensor this activity will explore is the SOUND SENSOR

The sound sensor is used to detect the sound intensity of the surroundings. The greater the sound intensity it receives, the stronger the output signal and the greater the return value is.

PORT 3 or 4



2. Drag over a Sound Intensity Value from the AI SENSOR KIT toolbox



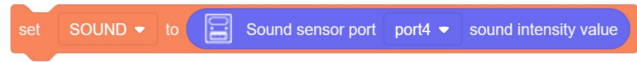
This sensor is an analog sensor. The values produced by this sensor will increase or decrease depending on the level/intensity of the sound it receives.



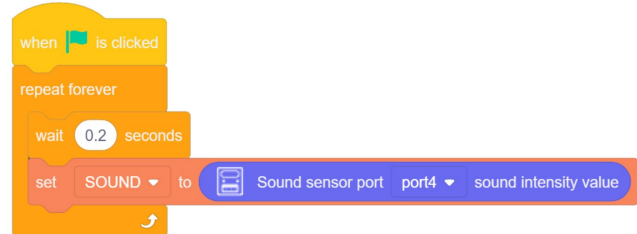
3. Change the PORT value to the correct port (see page 2).

4. The values for this sensor are reported to a PILL shape. Create a variable (SOUND).

1. Add this block to a SET VARIABLE TO "SOUND".



5. Place everything inside a forever loop with a small micro wait.



6. Download the code to the Magic Box. Practice with this group of code and record your results on the next page.

A. Can the sensor detect a whisper?

B. What analog value is reported when no sound is present?

C. Does the length of time the sound is heard by the sensor change the value?

D. What is the highest value you can get the sensor to report?

E. Document additional observations below.



SKILL BUILDER 3 – GESTURE SENSOR – DIGITAL INPUT

1. The Third sensor this activity will explore is the GESTURE SENSOR

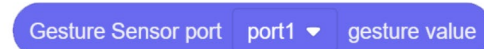
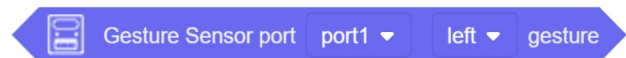
ANY GREEN
PORT

PORT 1-6



The gesture sensor is a 3D gesture recognition sensor using a I2C communication interface. It supports eight types of gestures by default with a maximum gesture update frequency that can read 240Hz. It also has anti-ambient light interference ability. With strong stability, fast recognition speed, high accuracy, and low power consumption, it is suitable for a variety of applications, including non-contact remote control, robot interaction, human-machine interaction games, and gesture lighting control.

2. Drag over the two main blocks related to the gesture sensor from the AI SENSOR KIT toolbox. *NOTE: We will not use the additional block for GESTURES in these activities.*

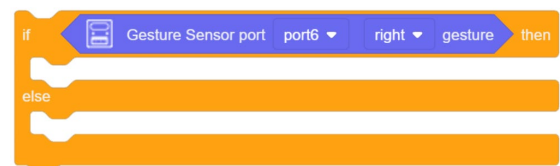
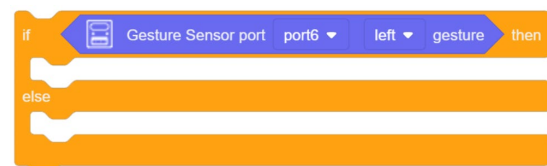


Notice that this sensor has both PILL and HEX Blocks. The GESTURE SENSOR is an intelligent sensor that can report a range of responses.

It can be used as a condition for a loop as we did with the PIR Sensor.
Example: Is left motion detected (Set variable to TRUE / FALSE)

It can also be paired with a variable to report which of the 8 motion is being detected in words (string values)

3. Change the PORT value to the correct port (see page 2).
4. Create TWO IF/ELSE Statements
5. Place the a GESTURE TRUE block inside each (one of the set to LEFT is TRUE and one the RIGHT is TRUE)



6. The next step will be a new use of the IF/ELSE statement.

Drag the second IF/ELSE into the ELSE condition of the first IF/ELSE.

This creates a condition known as an ELSE IF. This type of formatting creates a hierarchy or order of importance to the IF statement.

If two conditions could be true at the same time, nesting them in this fashion lets the program make a decision on which operation to do (which is more important). It also allows the program to still contain an ELSE (what to do if no conditions are TRUE).



HELPFUL TIPS

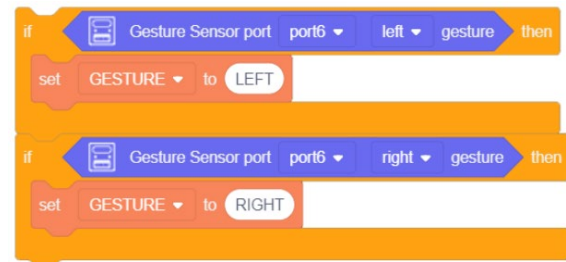
Two IF/ELSE loops will not work for this example. When either condition is not true, the ELSE will always want to set the Variable to NONE.



HELPFUL TIPS



Two single IF statement loops will partially work for this example. Each will clear out the variable when the other becomes true, but nothing will clear out the variable when neither is true. It will start blank (since we did not initialize it to a value), but once either condition is true, it will keep repeating that until told to do something different.

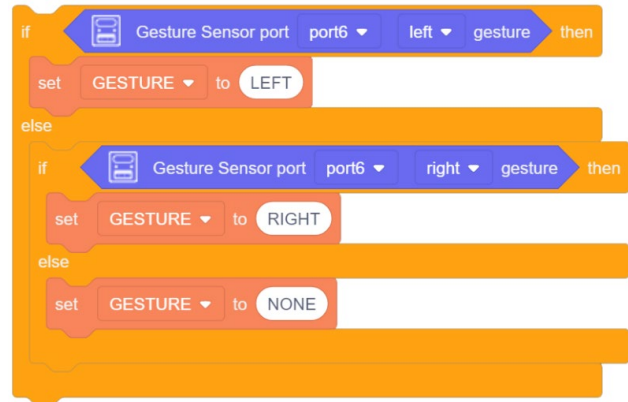


7. Create a new variable (GESTURE) and drag over (or duplicate) three of them (one for each condition).

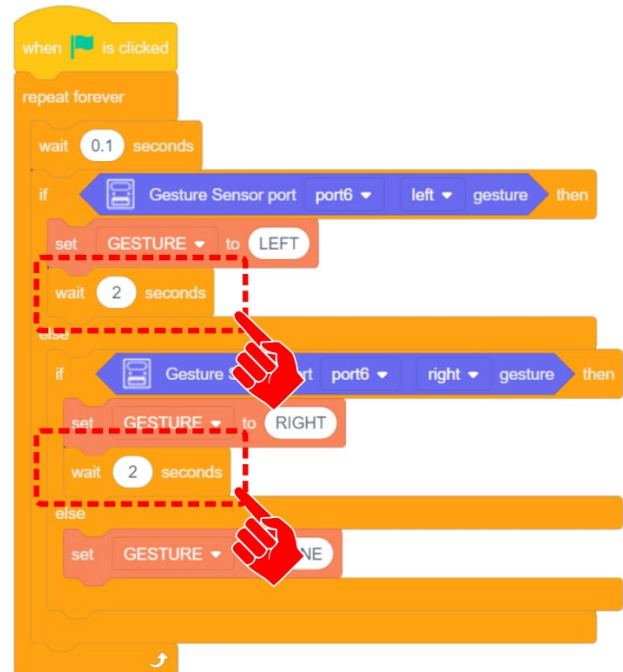
GESTURE LEFT (Most important, *do this if left is true... even if right is also true*)

GESTURE RIGHT (2nd Most important, *do this if right is true... as long as left is not true*)

GESTURE NONE (What to do if LEFT or RIGHT are not TRUE)

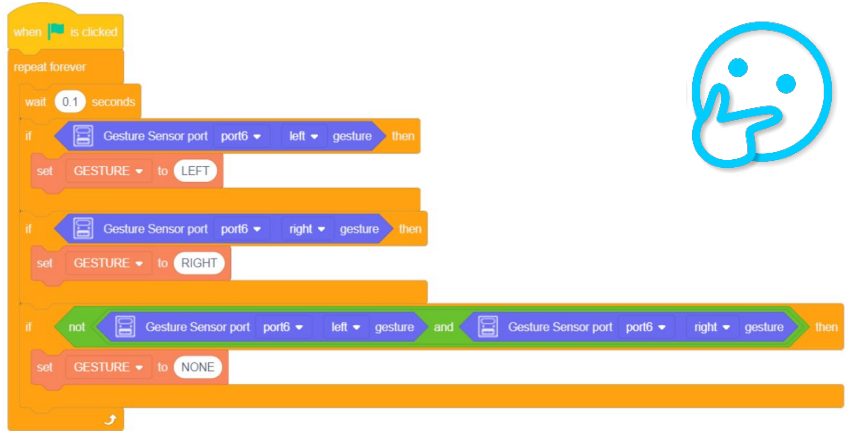


8. Add a WAIT inside the LEFT and RIGHT conditions so the string of text will remain up for a set time before clearing
9. Place everything inside a forever loop with a small micro wait.





An alternative to the ELSE IF would be to create three IF Statements with a NOT condition (BOTH LEFT and RIGHT are NOT TRUE)

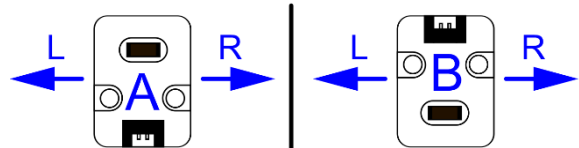


10. Download the code to the Magic Box. Practice with this group of code and record your results below.

A. Does SPEED matter (how fast the movement goes across the sensor)?

B. Does DISTANCE matter (how far the movement is from the sensor)?

C. Which orientation reads correctly "A" or "B"?

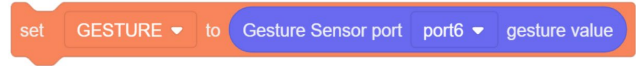


D. Does one direction or the other (left or right) seem to read more often (more consistently)?

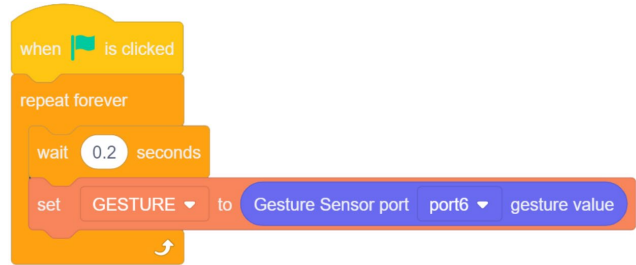
E. Document additional observations below



1. This time, add the GESTURE SENSOR VALUE to a SET VARIABLE TO "GESTURE".



2. Place everything inside a forever loop with a small micro wait.



3. Download the code to the Magic Box.
Practice with this group of code and record your results below.

F. List all 8 gestures the sensor is capable of reading and indicate if they can be read consistently (majority of the time), inconsistently (sometimes), or rarely.

1.	Consistent / Inconsistent / Rarely	5.	Consistent / Inconsistent / Rarely
2.	Consistent / Inconsistent / Rarely	6.	Consistent / Inconsistent / Rarely
3.	Consistent / Inconsistent / Rarely	7.	Consistent / Inconsistent / Rarely
4.	Consistent / Inconsistent / Rarely	8.	Consistent / Inconsistent / Rarely

G. Does the orientation of the sensor matter?

H. Can the sensor see the gesture if it is a moving object? Move the sensor across the object instead of moving an object across the sensor... rotate the sensor... move the sensor toward an object...

*I. Is this block **MORE** consistent, **LESS** consistent, or about the **SAME** as the other block with LEFT and RIGHT gestures?*

J. Document additional observations below.



SKILL BUILDER 4 – HUMITURE SENSOR – ANALOG INPUT

1. The forth sensor this activity will explore is the HUMITURE SENSOR

The humiture sensor is used to detect either temperature or humidity (amount of water vapor in the atmosphere) found in the current environment. It reports analog signals to the controller

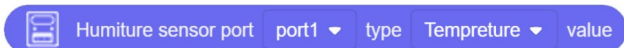
ANY GREEN
PORT

PORT 1-6



2. Drag over a Humiture Intensity Value from the AI SENSOR KIT toolbox

This one block is capable of reading either Temperature or Humidity.



3. Change the PORT value to the correct port (see page 2).

4. Create a new variable (HUMITURE) and place it inside a Set Variable loop as done in previous activities



5. Start with the Temperature value

6. Download the code to the Magic Box. Practice with this group of code and record your results below.

A. What value do you get a room temperature (a default reading)?

B. Looking at the reading you receive at room temperature, does the value make sense (F vs C... maybe...)?

C. Try to increase and decrease the temperature. Think of creative ways to increase or decrease the temperature reading of the sensor without risk of damaging the sensor. What is Lowest and Highest you can get the sensor to read? What method do you use?

Lowest Value:
Method:

Highest Value:
Method:

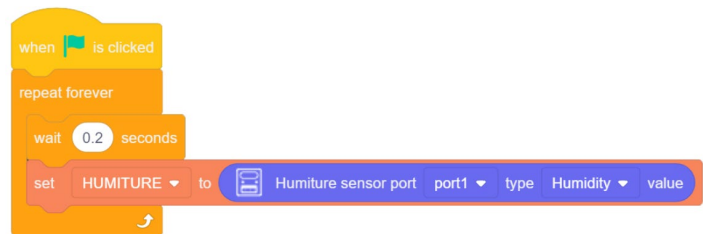


D. If the readings do not directly relate to something known (F or C). Can the data still be used? If so, how?

YES / NO HOW:

E. Document additional observations below.

7. Change the Sensor Reading to HUMIDITY
8. Download the code to the Magic Box. Practice with this group of code and record your results below.



F. What default value do you get (current amount of water vapor in the air)?

G. Try to increase and decrease the sensor reading. Think of creative ways to increase or decrease the humidity reading of the sensor without risk of damaging the sensor. What is Lowest and Highest you can get the sensor to read? What method do you use?

Lowest Value:
Method:

Highest Value:
Method:

H. Document additional observations below.



SKILL BUILDER 5 – LIGHT SENSOR – ANALOG INPUT

1. The fifth sensor this activity will explore is the LIGHT SENSOR

The light sensor contains a photosensitive resistor. The resistance value decreases with the increase of light intensity. Based on this, the change of its voltage, is detected and the light intensity data is obtained through AD conversion.

PORT 3 or 4



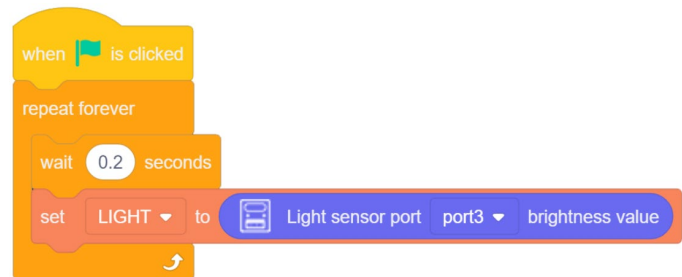
2. Drag over a Light Brightness Value from the AI SENSOR KIT toolbox
This one block is capable of reading the brightness or intensity of the light seen in an environment.



3. Change the PORT value to the correct port (see page 2).

4. Create a new variable (LIGHT) and place it inside a Set Variable loop as done in previous activities

5. Download the code to the Magic Box. Practice with this group of code and record your results below.



A. *What default value do you get (Amount of ambient light, no direct alteration to the amount of light in an environment)?*

B. *Try to increase and decrease the sensor reading. Think of creative ways to increase or decrease the light reading of the sensor without risk of damaging the sensor. What is Lowest and Highest you can get the sensor to read? What method did you use?*

Lowest Value:
Method:

Highest Value:
Method:

C. *Do shadows have effect on the sensor readings (stand over the sensor to create a shadow)?*

D. *Document additional observations below.*



SKILL BUILDER 6 – PHOTOELECTRIC IR SENSOR –DIGITAL INPUT

1. The sixth sensor this activity will explore is the PHOTOELECTRIC IR SENSOR

The photoelectric sensor, often called a photoelectric proximity switch, or IR sensor is a digital input. It detects the presence of an object through the connected circuit if there is an object shielding or reflecting the IR beam. The photoelectric sensor converts the input current into an optical signal on the transmitter side. The receiver side detects a target object if the transmitter signal is received by the receiver.

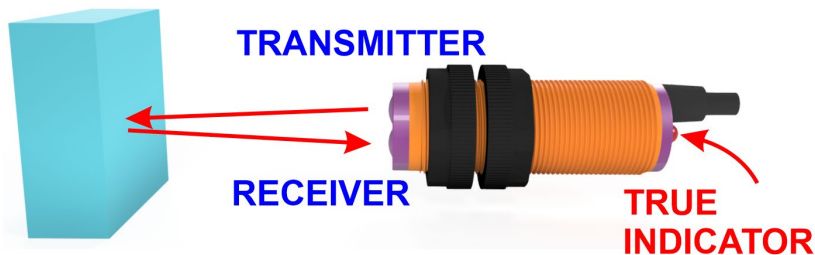
The detection distance can be adjusted by rotating the screw on the backside of the sensor. Rotating the screw clockwise will increase the distance.

ANY GREEN PORT

PORT 1-6

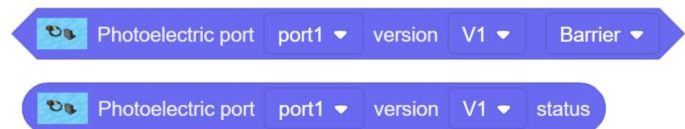


OBJECT



1. Drag over both Photoelectric tools from the AI SENSOR KIT toolbox

These blocks both report the status of the photoelectric sensor.



2. Change the PORT values to the correct ports (see page 2) and set the VERSION to V2.
3. Create a new variable (PHOTOIR) and place it inside a Set Variable loop as done in previous activities
4. Download the code to the Magic Box. Practice with this group of code and record your results below.



What values does the variable report? If the program does not work, try to change the version number from V2 to V1.

What value for ON - Object Found:

What value for OFF – NO Object Found:



B. Document additional observations below.



Note that there is a small red LED on the back of the sensor the light when a signal is received.

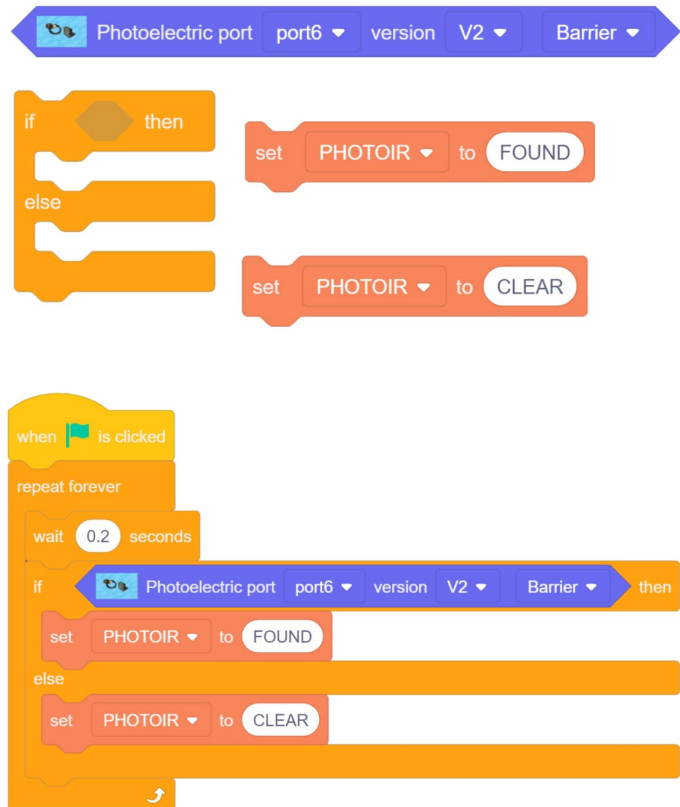
5. Notice from the previous test that photoelectric sensor is a digital sensor. The last block can only report either a 1 or a 0. This new block will allow us to set the variable to report the same numbers values **OR** a string value of our choosing.

Create an IF/ELSE Statement.

A TRUE condition will change the variable PHOTOIR to FOUND

ELSE (not true) the variable will show as CLEAR

6. Place everything inside a forever loop with a small micro wait.
7. Download the code to the Magic Box. Practice with this group of code and record your results below.



C. Using the setting BARRIER, what values does the variable report?

What value for ON:

What value for OFF:

D. Using the setting NO BARRIER, what values does the variable report?

What value for ON:

What value for OFF:

E. What happens when you switch between the two settings?



F. What is the terminology used for when a switch reads False by default when no signal is present? (Term was identified in a previous activity when digital inputs were introduced)

G. What is the terminology used for when a switch reads True by default when no signal is present? (Term was identified in a previous activity when digital inputs were introduced)

H. Document additional observations below.

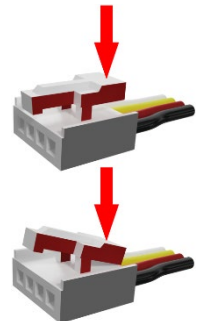
SECTION 4 – WIRING the MAGIC BOX AND SENSORS – #2 SETUP



Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS turn it off before making connections or damage to the Controller could occur. Be sure to ask your instructor if you have any questions.



Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.



CARFEULLY DISCONNECT ALL OF THE CURRENT INPUTS FROM THE MAGIC BOX

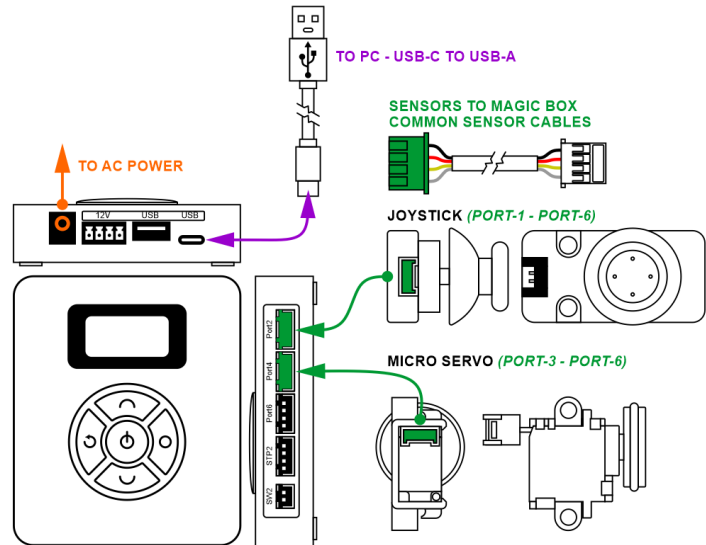
SETUP #2

3. For this part of the activity, you will need:

- 1x Magic Box
- 1x USB-A to USB-C Cable
- 1x AC/DC Power Adapter (12V)
- 6x Common Sensor Cables
- 1x Joystick – **PORT 4**
- 1x Micro Servo – **PORT 6**



4. Wire the Magic Box as shown.



SECTION 5 – SETUP #2 - EXPLORING ADVANCED SENSORS & OUTPUTS

• JOYSTICK & MICRO SERVO

SKILL BUILDER 7 – JOYSTICK – ANALOG/DIGITAL INPUT

1. The sixth sensor this activity will explore is the Joystick

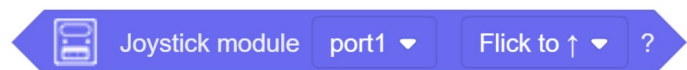
ANY GREEN
PORT

The working principle of the joystick is similar to that of a general gamepad joystick. The X and Y axis correspond to two 10K potentiometers respectively. When the joystick moves, it generates a corresponding analog signal and outputs the offset value. The Z-axis acts as a button (momentary digital signal)

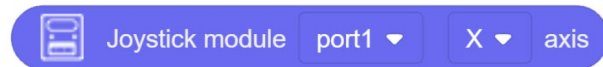
PORT 1-6



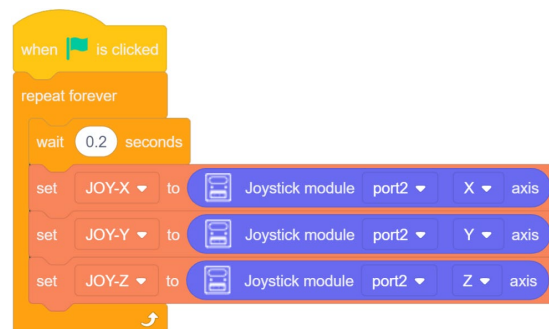
2. Drag over the two blocks for the joystick Value from the AI SENSOR KIT toolbox



3. Change the PORT values to the correct ports (see page 18).



4. Create three new variables (JOY-X), (JOY-Y), (JOY-Z) and place them inside a Set Variable loops as done in previous activities



5. Download the code to the Magic Box. Practice with this group of code and record your results below.

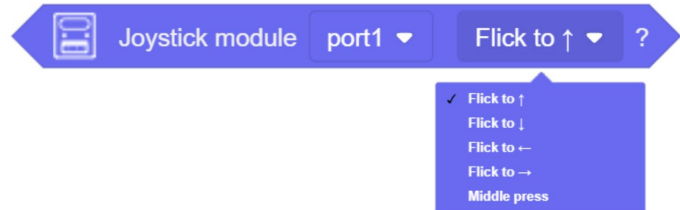


A. Fill in the chart below.

- Which axis is the Y and which is the Z.
- Push the joystick to its four limits & document the extreme values (lowest or highest value reported)
- Press down on the joystick, what values are reported when it is pressed and not pressed

LABEL THE AXIS	LABEL THE EXTREME VALUES	PRESS DOWN Z AXIS

8. The conditional block allows a programmer to create statements that can perform various operations when each motion is detected (disregarding the amount of motion).



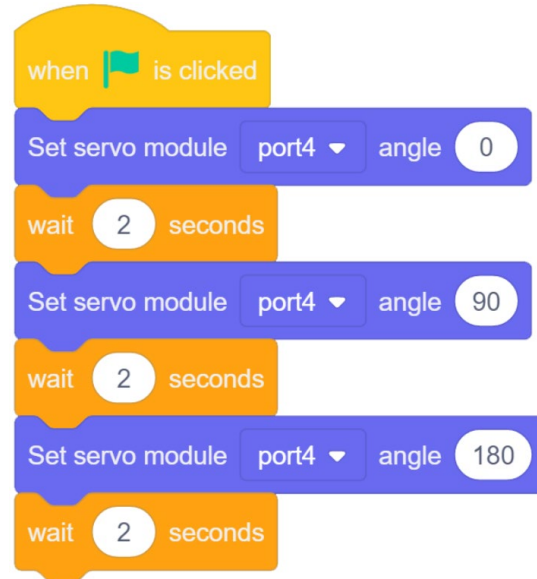
SKILL BUILDER 8 – MICRO SERVO - OUTPUT

- This device is the only **output** used in these experiments.
The servo acts as a motor that can be accurately positioned between 0 and 180 degrees. The position can be held indefinitely and is controlled by an internal potentiometer. The servo runs at a constant speed.
- Drag over the Set servo angle block from the AI SENSOR KIT toolbox. This block allows the programmer to drive the servo to a specific location and hold that position.
- Change the PORT values to the correct ports (see page 18).

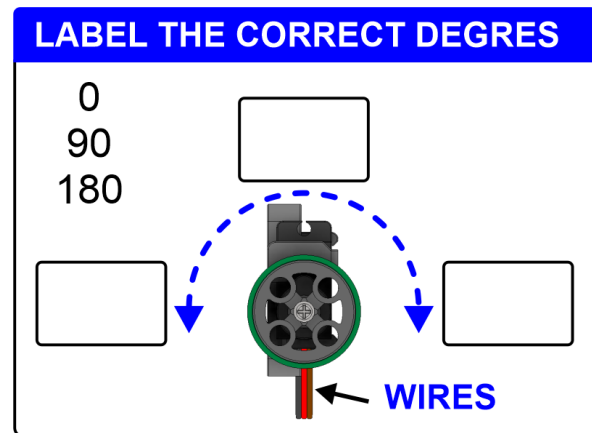
PORT 3-6



4. Create the simple program to the right
5. Download the code to the Magic Box. Practice with this group of code and record your results below.



- A. Hold the servo in the orientation shown to the right. After running the program, fill in the servo positions (0, 90, 180)



CHALLENGE

SECTION 6 – USE the JOYSTICK TO CONTROL THE SERVO

1. Use the joystick to control the position of the servo.
 - If the joystick is pushed left, the servo will turn CCW (to the left) to its extreme degree and stay there.
 - If the joystick is pushed right, the servo will turn CW (to the right) to its extreme degree and stay there.
 - If the joystick is pressed down, the servo will center itself and stay there.
 - This process will repeat forever.



CONCLUSION

Each sensor has its own conclusion questions built in. Answer the questions above for each section and get your instructor's approval.

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- _____ 1. Use the joystick's analog values to control the position of the servo
- _____ 2. Scenario: Wave your hand to the right and the robot looks right. Wave your hand to the left and the robot looks left.
Use the gesture sensor to control the position of the servo
- _____ 3. Scenario: Nighttime, the windows close. Daytime the windows open.
Use the light sensor to control the servo. If the light is low, the servo will turn to on extreme. If the light is high, the servo will turn to the opposite extreme.
- _____ 4. Scenario: Swipe your hand right, and the robot picks up a block from a common position, and places it on a pallet to the right. If you swipe your hand left, it will pick up the block from the common position and place it on the left pallet. The robot will wait for a hand swipe.



3.6 Magic Box - Exploring the Color Sensor

NAME: _____

Date: _____

Section: _____

INTRODUCTION

A **Color Sensor** is an **analog input** device that allows technologists, engineers, and others, to add an additional level of intelligence to their programs written for a robot. These types of sensors can be used for processes that require the need to distinguish between different products, detect data coding, or even determine if a product may have a defect that is distinguished by varying color.

Color sensors can often be found in automation processes that involve textiles, paints, food, printing, and even pharmaceuticals.



WARNING



Power OFF

Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS shutdown the BOX before making connections or damage to the controller could occur. Be sure to ask your instructor if you have any questions.

KEY VOCABULARY

- Color Sensor
- Input
- Magic Box
- Analog

EQUIPMENT & SUPPLIES

- Dobot Magic Box
- Color Sensor
- LED Module
- Common Sensor Cables
- 12V Power Cable and Supply
- USB C to A Cable
- Printed – Color Test Template

KEY CONCEPTS

Essential questions answered in this activity include:

- What can a color sensor be used for in industry? How does it work?
- How do you code a color sensor in Blockly?

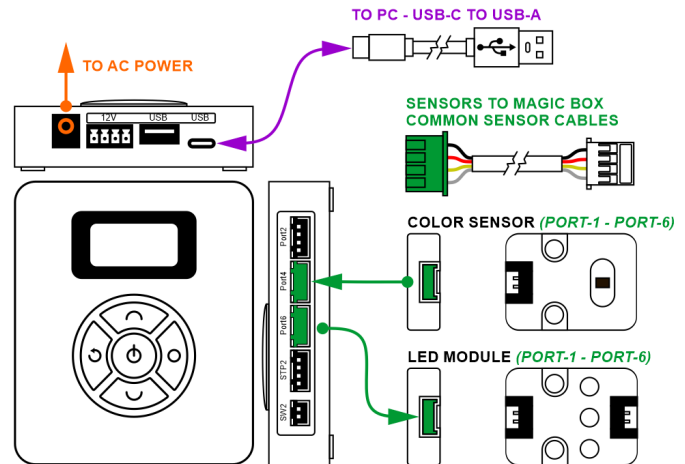
PROCEDURE

SECTION 1 – WIRING the MAGIC BOX AND SENSORS

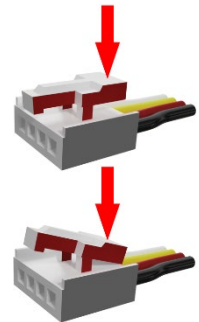


Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS turn it off before making connections or damage to the Controller could occur. Be sure to ask your instructor if you have any questions.

1. For this activity, you will need:
 - 1x Magic Box
 - 1x USB-A to USB-C Cable
 - 1x AC/DC Power Adapter (12V)
 - 2x Common Sensor Cables
 - 1x Color Sensor – **PORT 4**
 - 1x LED Module – **PORT 6**
2. Wire the Magic Box as shown to the right



Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.

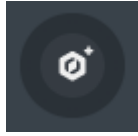


SECTION 2 – CONNECTING the MAGIC BOX to DOBOTLAB

1. Open up DobotBlock Lab in the software.
2. Once all of the wiring is done (sensors connected), power ON the Magic Box



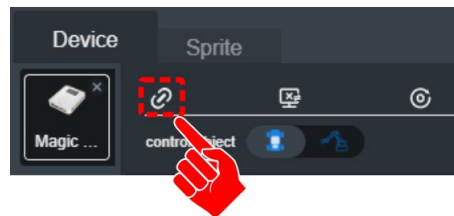
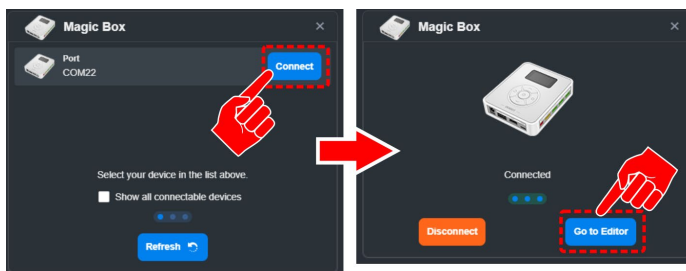
3. **No robot is necessary for this activity.** Delete both the Magician and Magician Lite Devices.
Note: This will send you over to the Sprite tab once there are no devices loaded.
4. Click back on the devices tab and click on Choose a Device



5. Select the Magic Box

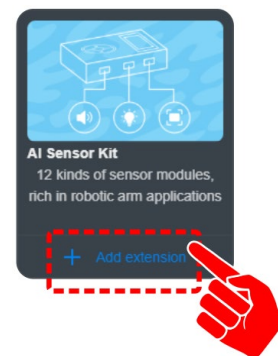


6. Connect to the Device



For this activity, you will need to add an EXTENSION to use the MAGIC BOX. Click the EXTEND icon in the bottom left corner.

Click on "+ Add extension" for the AI SENSOR KIT.



SECTION 3 – EXPLORE COLOR SENSOR

The color sensor can produce two different sets of values.

1. The sensor can identify the color of the object and return a set of RGB values
2. The sensor can identify six preset colors and return a true value if one of these colors are detected.
BLACK, WHITE, RED, GREEN, BLUE, YELLOW

ANY GREEN
PORT

PORT 1-6



SKILL BUILDER 1 – FIND INDIVIDUAL VALUES FOR RED / GREEN / BLUE

1. Drag over a Color Sensor Color Value block from the AI SENSOR KIT toolbox

This block will report a separate numeric value for the amount of R, G, B the sensor sees.

2. Change the PORT value to the port you are using

3. Next, create three separate VARIABLES

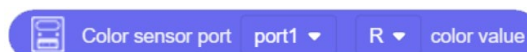
4. Create the program to the right.

The 0.2 MICROWAIT will slow down the program. It only allows a new sensor values to be pulled (looked at) every 0.2 seconds

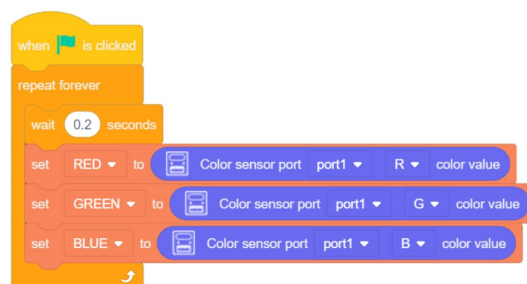
5. Print a copy of the Color Test Template

Run the program and document the values reported to the variables by the sensor.

Are the values reported to the variables the same or close to the typical values used in industry for each color?



RED / GREEN / BLUE



CHRIS & JIM CIM		
COLOR TEST TEMPLATE		
COLOR	RGB VALUE REPORTED	
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,255,255) r - g - b -	
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,0,0) r - g - b -	Red
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (0,255,0) r - g - b -	Green
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (0,0,255) r - g - b -	Blue
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,255,0) r - g - b -	Yellow
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,100,0) r - g - b -	Orange
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,0,255) r - g - b -	Magenta
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (0,0,0) r - g - b -	Black



Is there a pattern to the numbers reported that can be used to alter the variables (a computation / formula) to show a value that better represents the values expected? If so, what formula computation could be used for the variables?

Using the grey scale colors, what type of values are reported when no red, green, or blue values are present?

SKILL BUILDER 2 – DETECT KNOWN COLORS

1. Drag over a Color Sensor Detected Color block from the AI SENSOR KIT toolbox

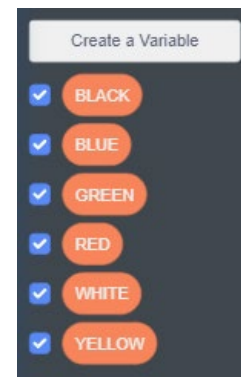


This command will report if a specific color is present.
BLACK, WHITE, RED, GREEN, BLUE, YELLOW

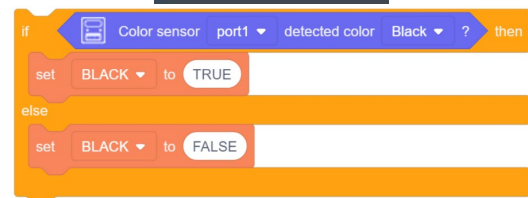
2. Change the PORT value to the port you are using

One way to use this block is to create sets of IF/ELSE statements. If the color is TRUE/PRESENT set the variable to 1, else set it to 0

3. Using the same program from TEST 1, create additional variables for the remaining colors (six total).



4. Create an IF/ELSE statement for EACH color

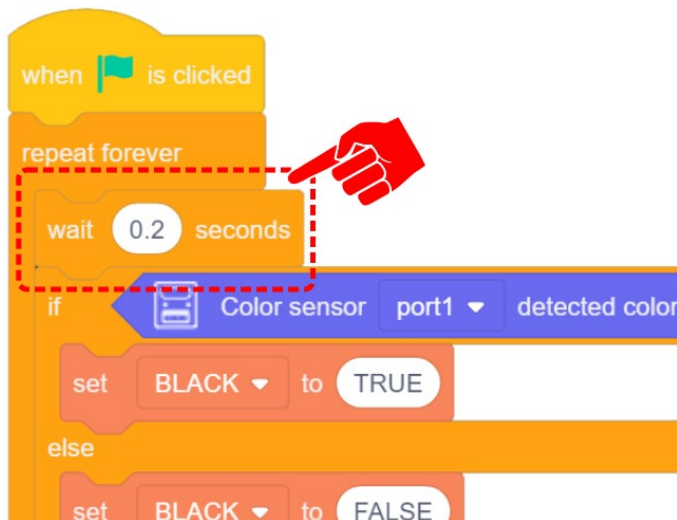




Ensure the PORT number matches the port you are using and that the color is different for each Statement

In previous activities, the SET VARIABLES have been set to **INTERGERS (whole numbers)** to represent YES (1) and NO (0). Notice in the example to the right, **STRING** values (**text**) can also be used.

- Link all of the IF/ELSE statements together inside a FOREVER Block
- Insert a small 0.2 second MICROWAIT at the begging of the program to slow the cycle down

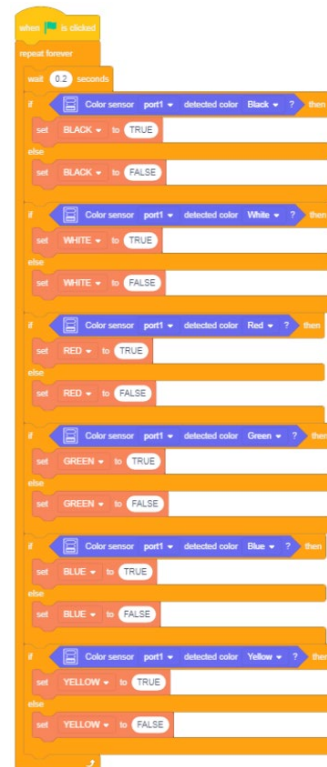


- Run the program and document if the colors are seen correctly (circle YES or NO).

Check the “Consistently Box” if the correct color is reported by the sensor (all the time or almost all the time)

or

Check the “Inconsistently Box” (sometimes) if the value reported by the sensor is often incorrect or sometimes does not read and report the color at all.



CHRIS & JIM CIM COLOR TEST TEMPLATE			
COLOR FOUND	RGB VALUE REPORTED	Consistently Found	Inconsistently Found
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,255,255) r: _____ g: _____ b: _____		
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,0,0) r: _____ g: _____ b: _____		
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (0,255,0) r: _____ g: _____ b: _____		
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (0,0,255) r: _____ g: _____ b: _____		
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,255,0) r: _____ g: _____ b: _____		
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (0,0,0) r: _____ g: _____ b: _____		
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,100,0) r: _____ g: _____ b: _____		
YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,0,255) r: _____ g: _____ b: _____		



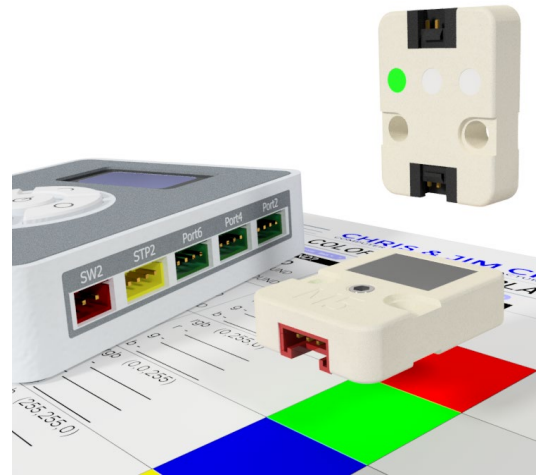
CHALLENGE

SECTION 4 – USE the COLOR SENSOR TO CONTROL THE LED

1. Use the values reported by the color sensor to control the color of LED1

As the color sensor is held over the Color Test Template, the color of the LED 1 will change color to match the color seen (sensor reports RED, LED becomes RED).

2. As an added level of intelligence to your program, have the variables show the value received as well.



- **The LED will show no color when black is received.**
- **Do not use the grey scale for this activity.**
- **Do not use ORANGE and MAGENTA for this activity.**

Continue to test your program until it works as needed.

If your set up did not work correctly the first time, what did you have to do to make it work?

CONCLUSION

1. *How accurate was the color sensor at displaying RGB values?*
2. *How accurate was the color sensor at determining colors from the color test template?*
3. *Would you be able to determine the color of the cubes provided by your instructor using this color sensor? Defend your answer, and then try it.*
4. *Explain how you might be able to use the robot with the color sensor. What tasks will you be able to do with it?*



GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

1. Get some colored cubes from your instructor. Write a program to determine the color of the blocks (hint: this is already done) Test it. How accurate is it at determining block color?

2. Program the LED to mimic the color of at least 4 different color blocks.
Example: (Red Block? Red LED turns on, ect.)



3.7 Block - Using the Color Sensor

NAME: _____

Date: _____

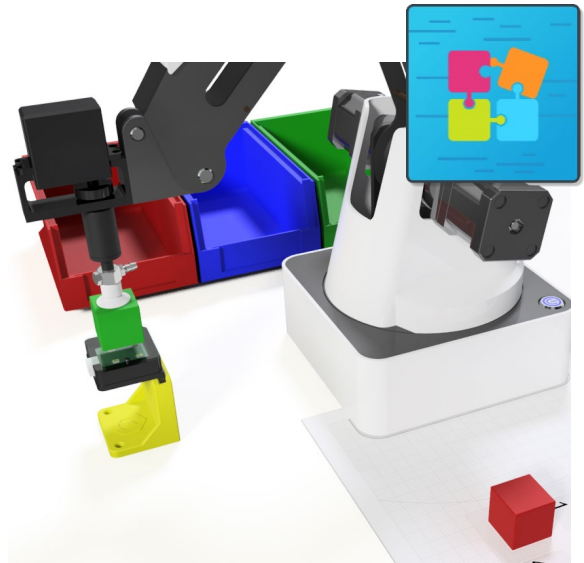
Section: _____

INTRODUCTION

Sensors are often added to industrial robots in order for them to perform specific tasks. These sensors can be as simple as a color detecting sensor, or as complex as a full vision system that will allow a robot to be aware of its surroundings or find a part and determine its location and orientation.

In this activity you will learn how to use and program the color sensor in blocky.

The Dobot will pick up a part and move it above the color sensor. The Dobot will then check the part's color and place it in a specific location for that specific color part. The robot will repeat the process each time a limit switch is pressed.

**WARNING****Power OFF**

Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS shutdown the Dobot before making connections or damage to the robot could occur.

KEY VOCABULARY

- Color Sensor
- If / Else If / Else Statement
- List
- Return True
- MY BLOCKS (Function / Voids)
- Identify Color
- Sum of List

EQUIPMENT & SUPPLIES

- Robot Magician
- Dobot Field Diagram
- 1" cylinders or 1" cubes (RED, BLUE, GREEN)
- Dobot Color Sensor
- DobotLab software
- Suction Cup Gripper
- [Dobot Input/Output Guide](#)
- 3 Small Containers

Y9_ /hb/9tç{

Essential questions answered in this activity include:

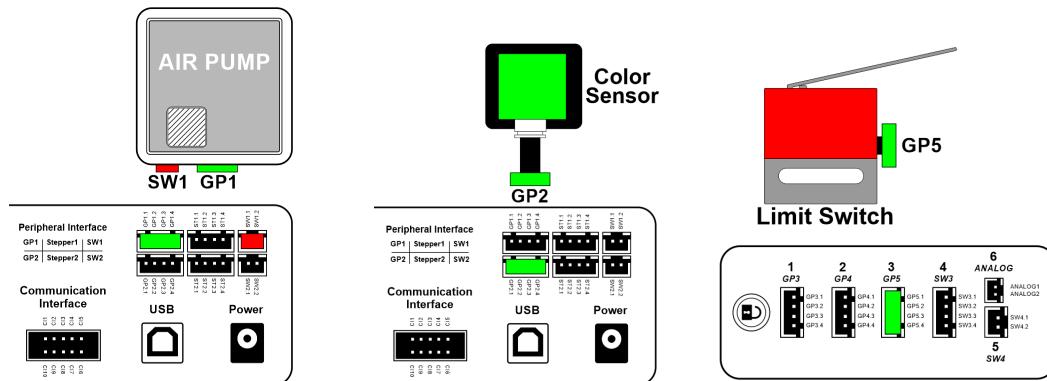
- What's the difference between digital and analog?
- What colors can I sense with the color sensor?
- How does the color sensor work?
- How do I wire the components together?
- How do I get a value from the color sensor?
- How do I print to a log?
- How do I sort items by color with a robotic arm?

twh/95 !w9

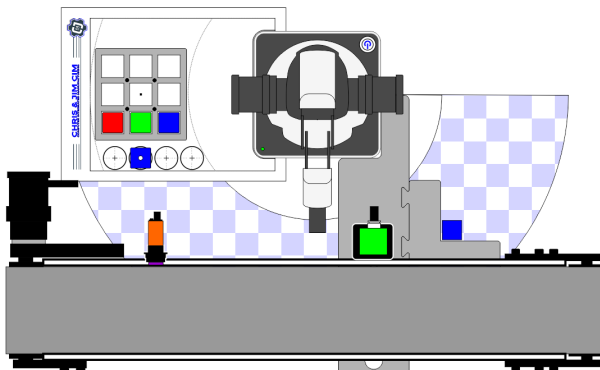


WARNING Power OFF

1. Set up the robot with a suction cup - **Dtm g { ² M**
2. Wire the robot with the Color Sensor plugged into port **Dth**
3. Wire the robot with the Limit Switch plugged into port **Dtp π 9Lnp**



Set up the robot, conveyor, and color sensor as shown in the diagram below:



The only part of the conveyor that you will be using for this activity is the color sensor.

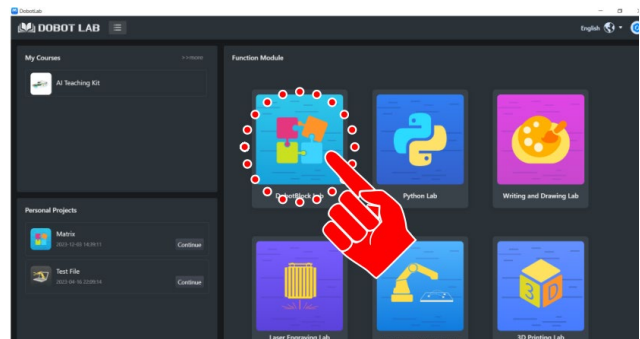
You do not have to connect anything else to the robot at this time. Just the color sensor.



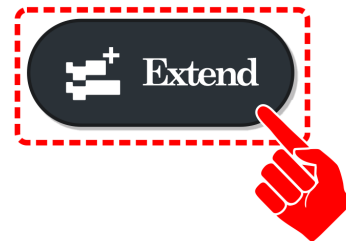
Order of operations

- Start with the robot in a safe/home position.
- Place a single cube, one at a time for the robot to pick up from a common location.
- A limit switch will be used to call for the robot to come and get the block for evaluation.
- After determining its color, drop the cube off at a specific location for that color.
- Manually remove the cube once it has been placed and send the robot to its home position.

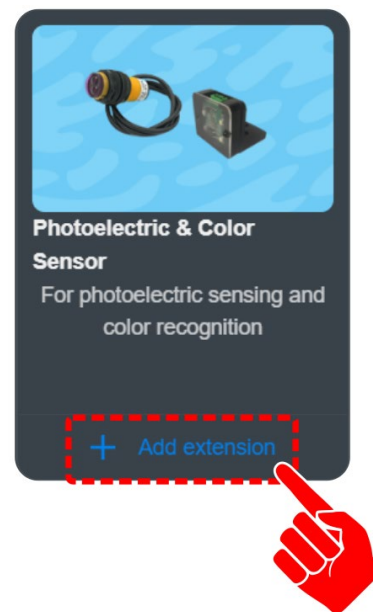
Open a new file for DobotBlock Lab



For this activity, you need to add an EXTENSION to use the COLOR SENSOR. Click the EXTEND icon in the bottom left corner.



Click on "+ Add extension" for the Photoelectric and Color Sensor



The first step in programming this activity is to set up the inputs. You need to teach the program which sensors we are adding and which port they are plugged into.

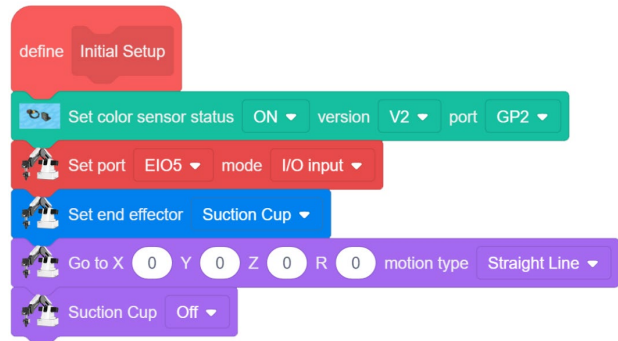
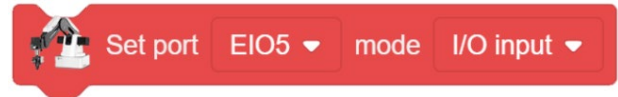
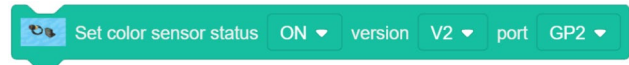
Next, set up the limit switch as we have done in past activities.

SET PORT GP5 EI05 I/O INPUT

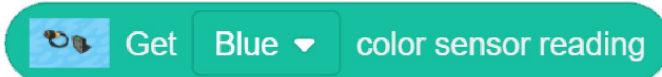
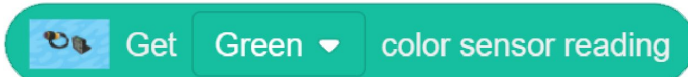
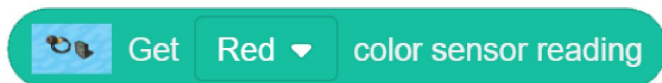
Add these setup blocks to our typical *Initial Setup* MY BLOCK

Color Sensor Extended Toolbox:

- SET COLOR SENSOR STATUS
Settings: ON, V2, GP2



Now that the INPUTS are setup, the COLOR SENSOR needs to be tested to make sure that three colors are read correctly. The color sensor should report what color it is currently reading. This sensor is only setup to read RED, GREEN, and BLUE (The values for the different colors are preset in the blockly codes so they cannot be seen or changed).



Create a loop that will show the value currently being reported

IF the color is RED

Or IF the color is GREEN

Or IF the color is BLUE

This can be done by creating an **IF/ IF/ IF Loop**



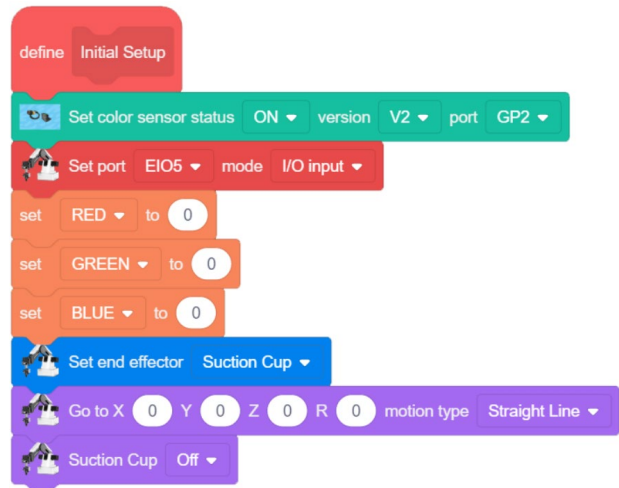
In order to allow us to see which color is “TRUE”, we will create three different variables; one for each color. We will be able to watch the values change in the SPRITE window.



Can you change the sprite to a Magician as shown in the diagram?



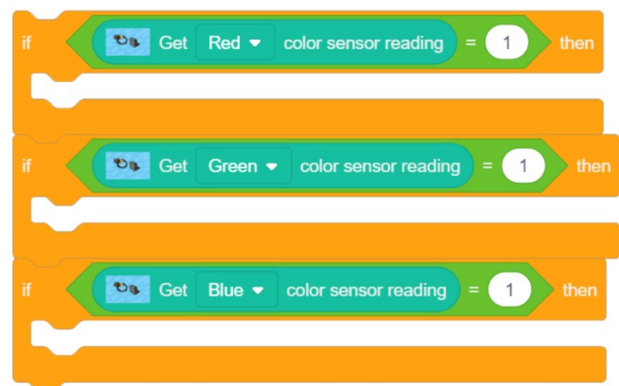
Add these variables to the INITIAL SETUP and set their initial values to zero



Next create three separate true statements that will go into each IF STATEMENT.

Use the EQUALS condition as has been done in the past and set the SENSOR READING equal to one (1).

Place the conditions into the IF STATEMENTS.

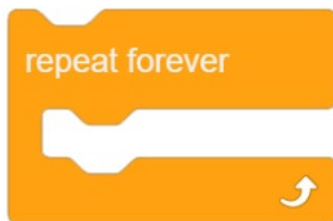
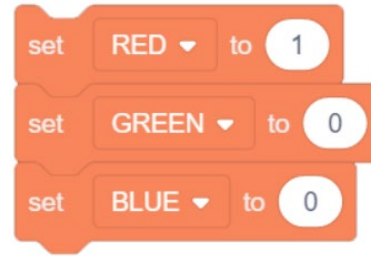


Place code inside each IF STATEMENT that will show us which color the sensor sees. Use the variables we just created to do this.

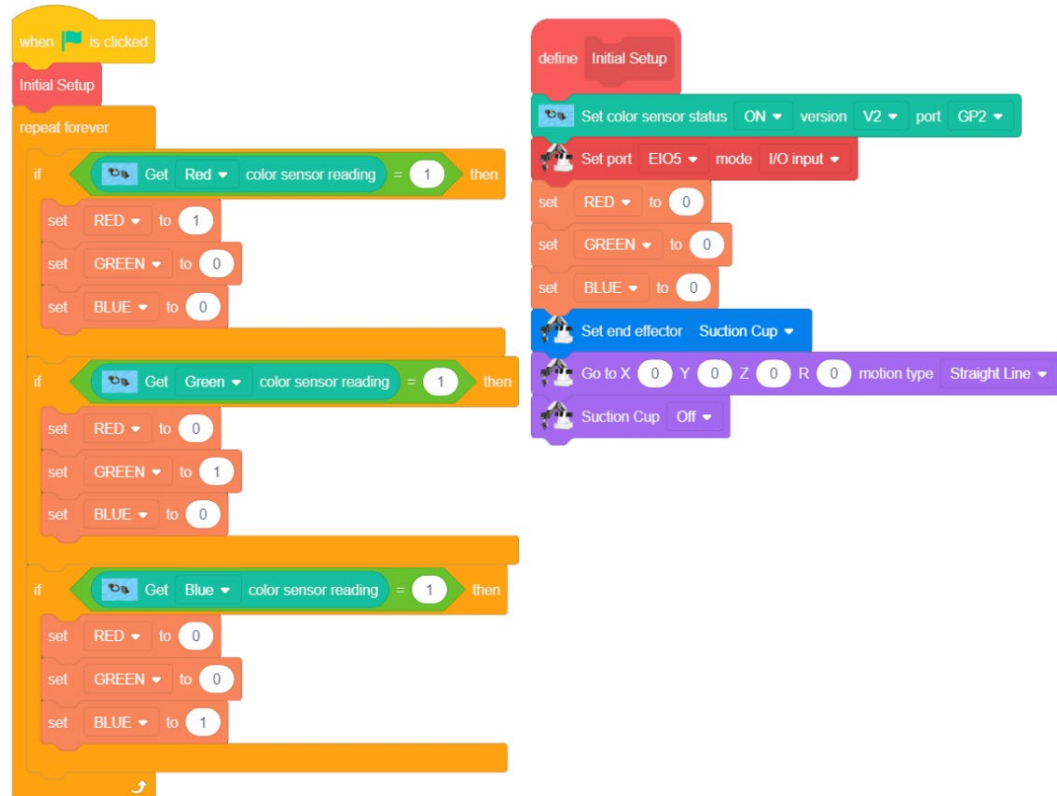
In order for this to work correctly, turn on one variable and turn off the other two.

Remember: in the digital world, 1 = ON, and 0 = OFF

Put the IF STATEMENTS in a FOREVER LOOP, otherwise the program will only be checking for color once.



Put together everything and this is what it looks like



Once the program is completed, run it and see if it works correctly. Every time you put a colored block in front of the sensor, you should see the correct color reported in the running log. If it does not work, troubleshoot it until it does.

Points for discussion:

What does the sensor read when there is no cube placed on the sensor?

What happens if you put the yellow cube on the sensor?

What type of reading do you get as you raise the cube up and away from the sensor?

Sometimes you will get a random incorrect reading as it changes between colors. Why?



You may notice that the sensor seems to report a certain color, even when there is no cube present.

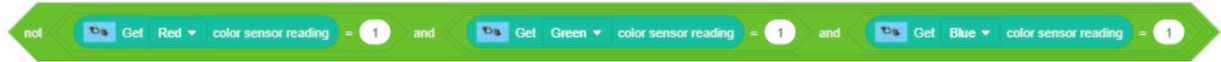
Is this because of the order of the If statements, and which color comes first? Try changing the order of the IF STATEMENTS

Do you get the same results? Did changing the order effect the readings?

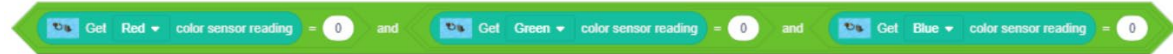


If you noticed there was never any option in the program telling it what to do if none of the values were seen. Try adding one additional IF STATEMENT that turns off the VARIABLES if none of the colors are reported from the sensors. Something like the codes below:

Sensor Readings are NOT TRUE (RED and GREEN and BLUE)

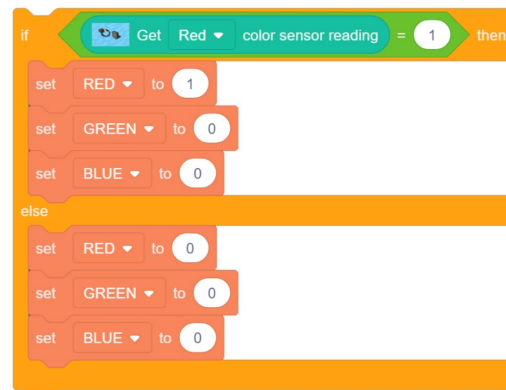


Sensor Readings are FALSE (ALL ZERO) (RED and GREEN and BLUE)

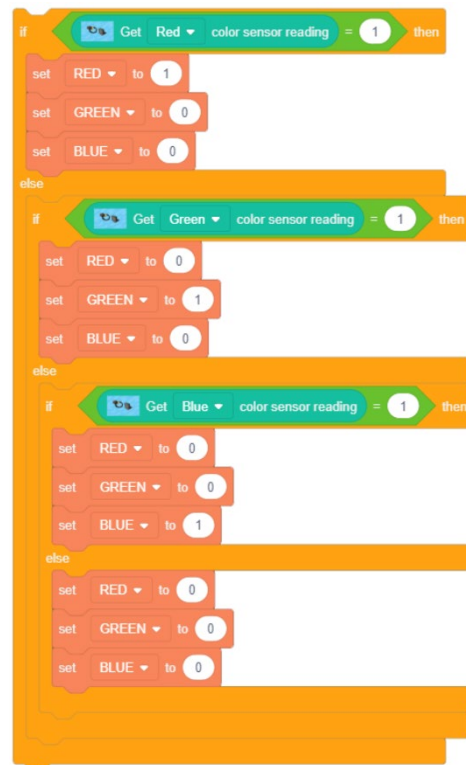


But there is an alternate solution.

If you use three separate IF/ELSE STATEMENTS (Variable on if TRUE, all off if NOT TRUE)... this may not make the program any shorter... if anything, it may even make it longer.



Instead, try nesting a few IF/ELSE statements. The final ELSE will be ALL OFF.



If your setup is like ours... it did not help. It seems to fluctuate a little less often. You could even try to put some [micro waits](#) in the program to slow down each cycle and look for the sensor value at a slower pace. The [IF/ELSE Statements](#) are better code theory, but since you cannot see the raw data from the sensor you cannot change how it works. As long as you know how the sensor is going to act, this will work as long as the robot does not drop the cube (no cube present when testing the color).



*The color sensor works best if the object is held at a consistent distance from the sensor
Dobot suggested distance = (0mm to 10mm) or (0 to 3/8 inch)*



The different versions of the conveyor kits can come with different shades of colored cubes. The V2 sensor (GREEN COLORED PORT) tends to be more forgiving when it comes to shades of colors.

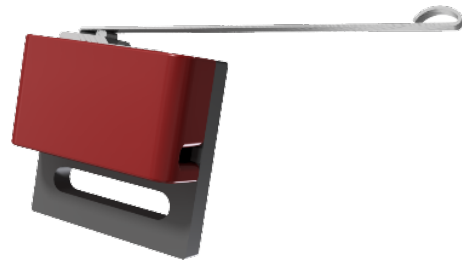
Now that we have the color sensor reading correctly, you can develop the rest of the program.

In order to keep the program short and organized it will be easiest to use several **MY BLOCKS** (*functions/voids*).

Use a manual input to start each loop

Process Flow

- Start at HOME
- Get a block from a common location
- Take the block to the color sensor station
- Evaluate the color
- Drop the block off in one of three bins (one for each color)
- Return HOME and wait for a new signal (a new block is in place a ready for evaluation)



MY BLOCK – INITIAL SETUP

Use what we have already developed. Add any additional Variables as needed.



MY BLOCK – PNP

- Create a Pick and Place routine.

- Wait for INPUT SIGNAL
- Start with the robot at a **Home** position
- **Pick and Place** - Get the block and put it on (near) the color sensor.
- **ADD** a 1 second wait at the end of the PnP routine before looking for the color (allow the reading to become a solid reading)

MY BLOCK – EVALUATE

- Evaluate the COLOR

Use the IF / IF / IF Statements we used earlier



There are so many different creative ways you can proceed with dropping the cube off in their correct colored bins once the color is known. Using what you have already learned, implement one that works for you.



Be sure to consult the Dobot Input/Output Guide if you want to use other inputs and outputs, as damage to your robot or your other equipment may result.

Once the program is completed, run it and see if it works correctly. If it does not work, troubleshoot it until it does. Issues with the colors?

If your set up did not work correctly the first time, wait did you have to do to make it work?



CONCLUSION

1. What happens if a block isn't there when the color sensor is told to get a color with the current program? Give a reason why.
2. How might you keep track of how many blocks there are of each color?

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- _____ 1. Randomly arrange cubes in a 3x3 palletized matrix for the robot to pick from. After determining its color, drop the cube off at a specific color location. Manually remove the cube once it has been placed.
- _____ 2. Have the robot stack the cubes in columns by color.
- _____ 3. Color the 9 squares on the Dobot field diagram either red, blue, or green. You will then write a program that takes cubes from a cube matrix and puts the respective color cubes on the colored squares.





- CHAPTER 4 -

ROBOTICS SCENARIOS & PERIPHERALS

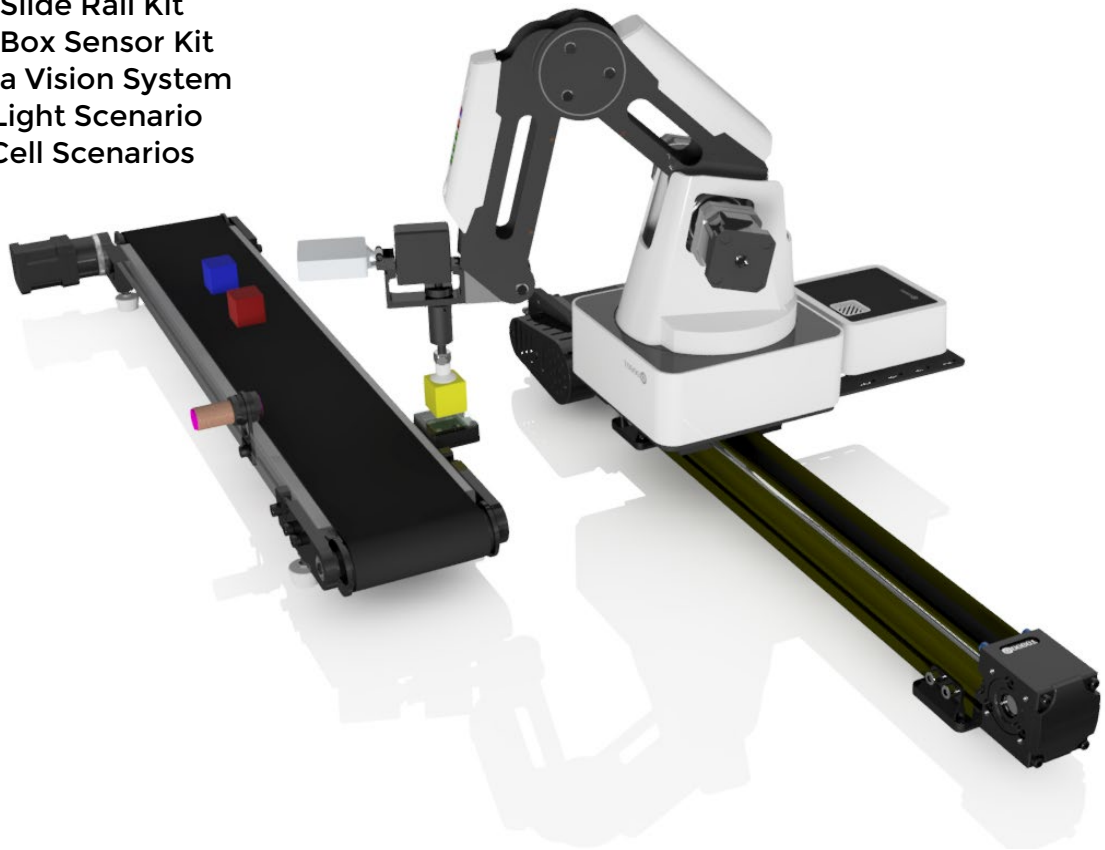
CHAPTER INTRODUCTION:

In this chapter, you will explore how to extend the usefulness of a robotic arm in an industrial setting by adding different peripherals. Imagine if a robot could move further, its parts could be automatically brought to it, or it had a camera to be able to identify different parts. This can be easy to do, and you will get the opportunity to use these capabilities in these activities.

This chapter will introduce you to the use, wiring, and programming of these peripherals and provide you the chance to model a wider variety of industrial operations and scenarios.

TOPICS COVERED:

- Mini Conveyor Kit
- Linear Slide Rail Kit
- Magic Box Sensor Kit
- Camera Vision System
- Stack Light Scenario
- Work Cell Scenarios



4.1 Blockly - Conveyor System Basics: Motion

NAME: _____

Date: _____

Section: _____

INTRODUCTION

Robotic arms need to communicate with each other as well as other **peripherals** such as **conveyor belts** or **linear rails** in order to move materials or products through stages of a work cell. The robot is used to control all such peripherals, making it the “Boss” of the work cell.



In this activity you will learn how to program a robotic arm to control a conveyor belt and use an Infrared Sensor to create a **closed loop system**. The sensor can be used to start or stop the conveyor in the same spot every time, increasing accuracy of parts placement.



Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS shutdown the Dobot before making connections or damage to the robot could occur.

KEY VOCABULARY

- Stepper Motor
- Conveyor Belt
- Linear Rail
- Output
- IR - Infrared Sensor
- Closed Loop System

EQUIPMENT & SUPPLIES

- Robot Magician
- Dobot Field Diagram
- 1" cylinders or cubes
- Dobot Conveyor
- DobotLab software
- Suction Cup Gripper
- Dobot I/O Guide
- IR Sensor

KEY CONCEPTS

Essential questions answered in this activity include:

- Is the conveyor an input or an output to the robot?
- Is the Infrared sensor an input or an output?
- How do I wire the components together?
- What can I do with the infrared sensor?
- How do I code the conveyor in blockly?
- How do I code the infrared sensor in blockly?

PROCEDURE

Order of operations

- The Robot will pick up a cube from a known location and place it on the conveyor belt.
- The Robot will return home and then start the conveyor belt.
- The conveyor belt will run until the block arrives at the IR Sensor for inspection.
- The block will be manually removed from the belt, inspected and then returned to the belt.
- Once the block is returned to the belt, the belt will run again until the parts runs off the belt and into storage.
- The process will loop forever.



WARNING

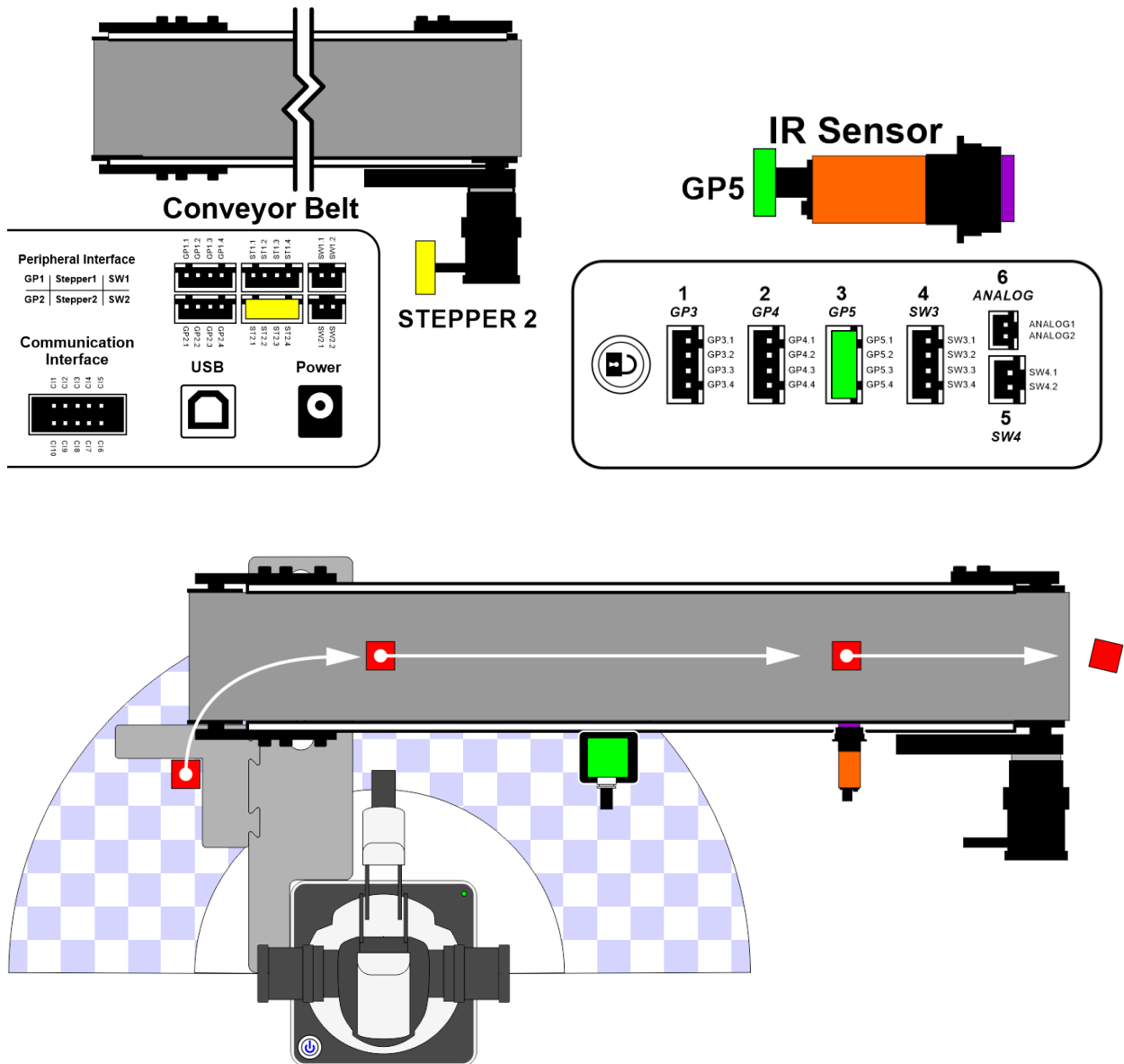


Power OFF

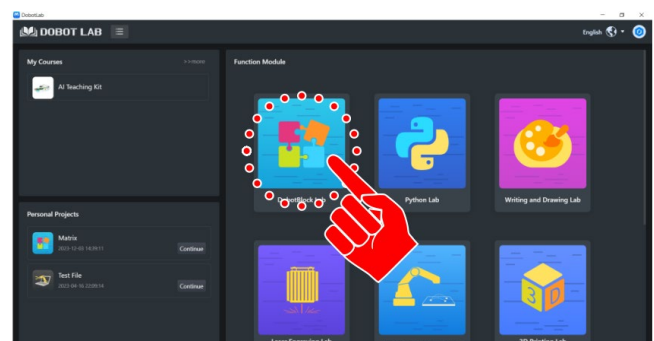
Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Set up the robot with the suction cup gripper.
2. Plug the Conveyor Belt into **STEPPER2** of Robot and plug the IR sensor into **GP5** of the Robot

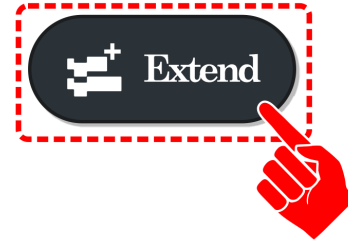




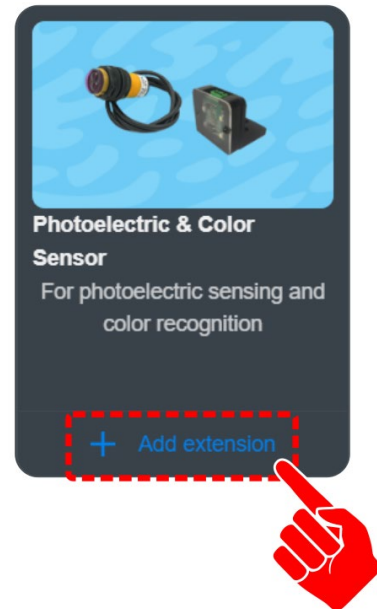
Open a new file for DobotBlock Lab



For this activity, we will need to add an EXTENSION to use the PHOTOELECTRIC SENSOR (IR Sensor). Click the EXTEND icon in the bottom left corner.



Click on "+ Add extension" for the Photoelectric and Color Sensor

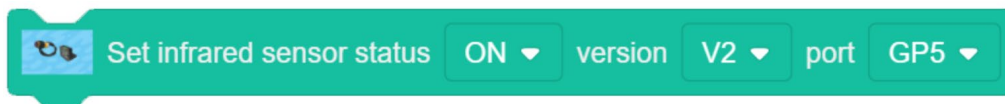


We will also need additional new blocks for this activity:

Photoelectric Sensor Extended Toolbox:

- **SET INFRARED SENSOR STATUS**

Settings: ON, V2, GP5



Settings Toolbox:

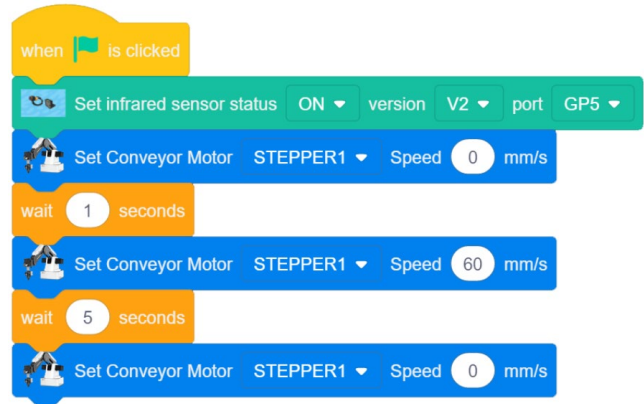
- **SET CONVEYOR MOTOR**

Settings: STEPPER1, 60 mm/s



Write a quick code that will turn on the conveyor for 5 seconds

In this example, 60 mm/s is used for the speed of the conveyor. Play with this value until you have a controllable speed for this activity.

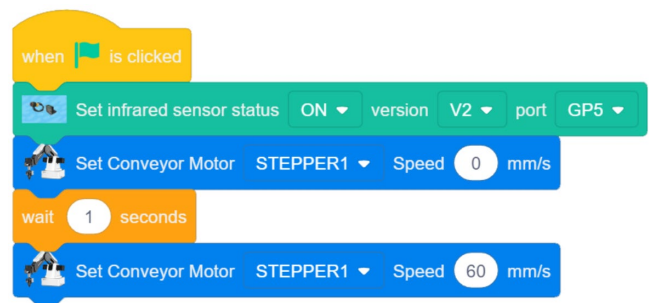


Once the program is completed, run it and see if it works correctly. If it does not work, troubleshoot it until it does.

If your set up did not work correctly the first time, what did you have to do to make it work?

Now edit the current code to get the conveyor belt to stop when an object is detected by the IR Sensor.

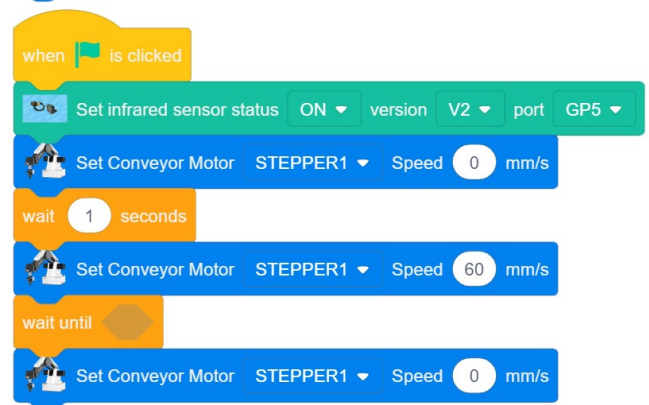
Remove the **DelayTime** from the program.



Change this OPEN loop system to a CLOSED loop system by using the sensor to add feedback.

To create a loop that will keep checking the IR Sensor value until an object is detected, we are going to replace the *wait for time* with a **WAIT UNTIL**.

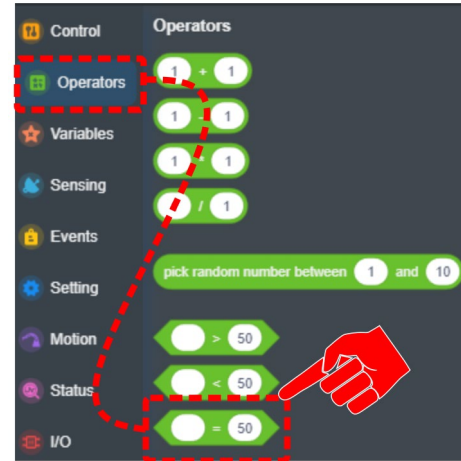
Insert a **WAIT UNTIL** loop and insert it where the wait for time block was.



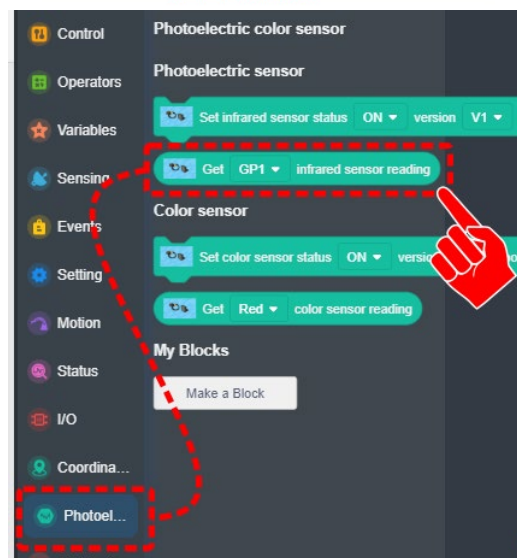
Create a **condition** that evaluates the IR sensors value. We are looking for the sensors value to change from 0 to 1.

The IR Sensor reads true/high/1 when an object is present. The small LED on the back of the IR Sensor will also light up.

Insert the **EQUALS** condition from the Operators Toolbox.

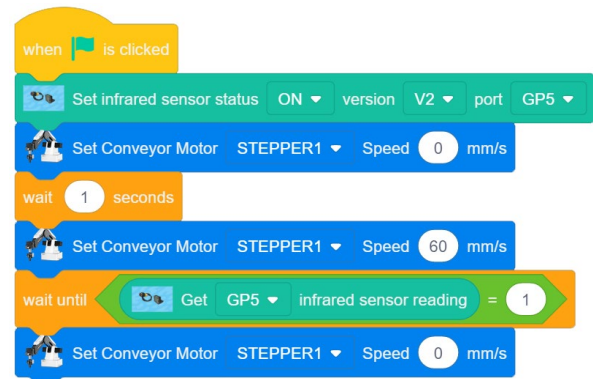


Insert the **GET INFRARED SENSOR READING** from the Photoelectric Color Sensor Extended Toolbox.



Add both blocks together to complete the statement.

Make sure to change the **GET INFRARED SENSOR READING** value from GP1 to GP5 and the number value to 1



Run the program. Let the conveyor run for a few seconds, then manual place a cube in front of the sensor. The conveyor should stop.

If it does not work, troubleshoot it until it does.



If your set up did not work correctly the first time, what did you have to do to make it work?

Now that the conveyor starts and stops using the IR Sensor, you can find positions, set any needed VARIABLES, and create MY BLOCKS and set variables for this activity.

Create a Pick and Place routine.

- Start with the **robot at a Home** position
- **Pick and Place** - Get the block and put it on the belt and return to the home position.

Next, start the conveyor until the block reaches the other end.

- **Start the belt** and run it **until the cube reaches the sensor**
- **Stop** the belt

Create a variable named “waiting”. Set this variable to one (1) when the robot is waiting and zero(0) when it is not.

Once the program is completed, run it and see if it works correctly. If it does not work, troubleshoot it until it does.

If your set up did not work correctly the first time, what did you have to do to make it work?

Reminder:

This activity requires the following operations:

- The Robot will pick up a cube from a known location and place it on the conveyor belt.
- The Robot will return home and then start the conveyor belt.
- The conveyor belt will run until the block arrives at the IR Sensor for inspection.
- The block will be manually removed from the belt, inspected, and then returned to the belt.
- Once the block is returned to the belt, the belt will run again until the part runs off the belt and into storage.
- The process will loop forever.



Now that the conveyor starts and stops, the next step is to create a loop that will allow the block to be removed and then wait for it to be returned. The issue is that we have no idea when the block has been removed, how long it will take for the inspection, or when it will be returned.

We need to create a closed loop system that will look for the following conditions without respect to time.

The Block has been REMOVED (Sensor Feedback)

The Block has been RETURNED (Sensor Feedback)



*Some of this program can be condensed into separate **MY BLOCKS** (functions) in order to simplify the main program. This makes the program much more efficient and easier to troubleshoot.*

The final task is to start the conveyor again and run the part off the end and into a container. Use the same code you started with to accomplish this task. The time needed to run the part off the conveyor will be different for each project, depending on where the sensor is mounted.



SAFETY: Place a small **WAIT** in front of this section of code to allow time for the user to get their hand away from the belt before it starts running.

Once the program is written, try it and make sure that it works as expected. If it does not work correctly, troubleshoot until it does.

If your set up did not work correctly the first time, what did you have to do to make it work?



CONCLUSION

1. Why is it better to have the infrared sensor stopping the conveyor belt rather than just running it for time?
2. How would the program be different if the conveyor belt could not be run as a linear rail?
3. What's one way to determine where the robot is at any given time in the program?

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- _____ 1. Use functions to make your program as short as possible.
- _____ 2. Add the AI extension as done in previous activities and use the color sensor and make the robot report what color block is being sent down the conveyor using voice.



4.2 Block - PnP with Linear Slide Rail

NAME: _____

Date: _____

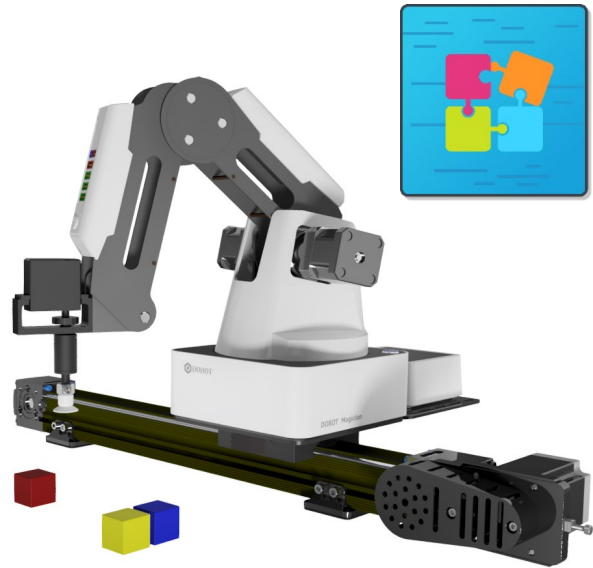
Section: _____

INTRODUCTION

When the workspace of Dobot Magician is not enough, you can extend it with a sliding rail. This peripheral allows you to mount the robot on a rail, and precisely control its position along the rail. Through the robot, you can change the position of the robot through the use of a stepper motor.

This device can be useful, especially in an industrial setting. It allows you to do long distance pick and place and palletization routines for one, sometimes making it possible for one robot to do the work of two or more, saving money and time.

It is also very useful when **Machine tending**. A robot can put raw materials into a machine and remove finished product allowing for **Lights Out Manufacturing**.



WARNING



Power OFF

Caution: NEVER wire anything to the Magician or the Magic Box while it has power on. ALWAYS shutdown the BOX before making connections or damage to the controller could occur. Be sure to ask your instructor if you have any questions.

KEY VOCABULARY

- Sliding Rail
- Lights Out Manufacturing
- Machine tending
- Stepper Motor

EQUIPMENT & SUPPLIES

- Dobot Magic Box
- Digital - 2-Button Input Module
- Analog – Potentiometer Module
- 12V Power Cable and Supply
- Magician Robot Arm
- Suction Cup Gripper
- Module Connecting Wires
- USB C to A Cable

KEY CONCEPTS

Essential questions answered in this activity include:

- How can I use a robot to move a sliding base?
- What can a sliding base be used for?

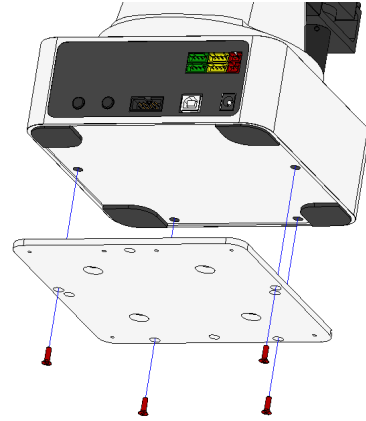
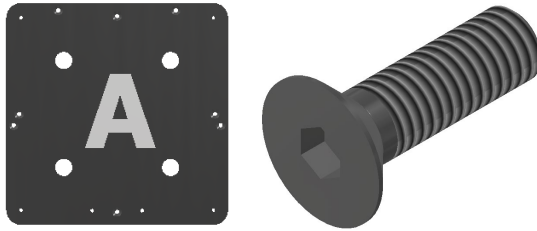
PROCEDURE

SECTION 1 – INSTALLING THE ROBOT ON THE LINEAR SLIDE RAIL



If this section is already complete, skip to SECTION 2 – WIRING.

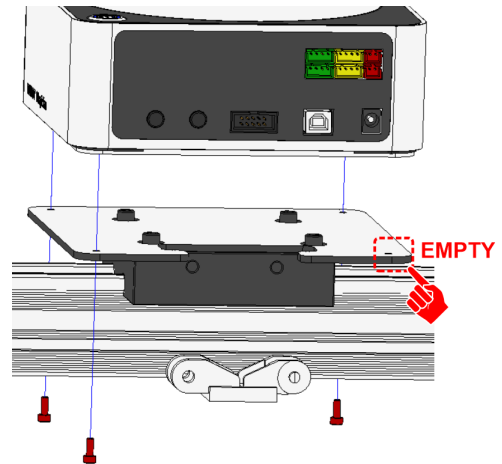
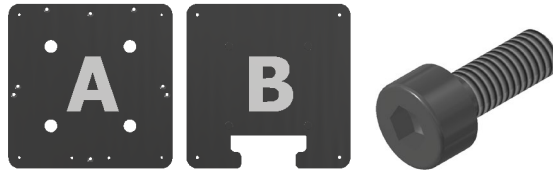
Step 1 – Mount the Dobot Magician on **PLATE A** with **FOUR** M3x10 countersink cap screws. Ensure the countersinks of **PLATE A** point outward. Orient the plate as shown in the image to the right.



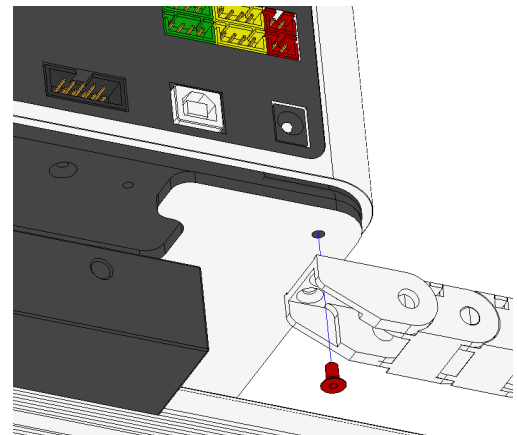
STEP 2 – Mount **PLATE A** (with the Dobot Magician attached) onto **PLATE B** with **THREE** M3x8 Hex-Socket Head Cap Screws. Make sure the back of the robot faces the cut out on the back of **PLATE B**.



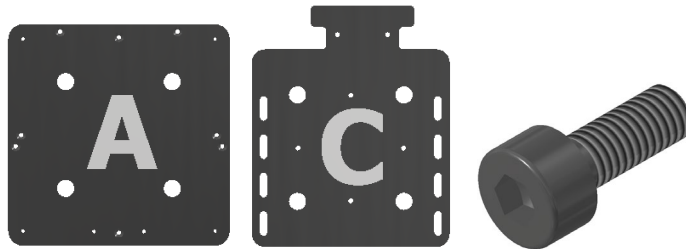
Do not put a screw in the back right corner (**EMPTY**). We need this hole for the next step



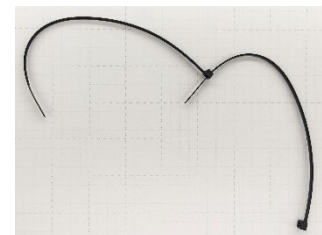
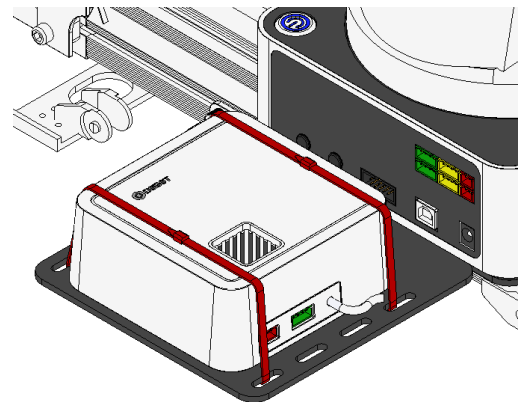
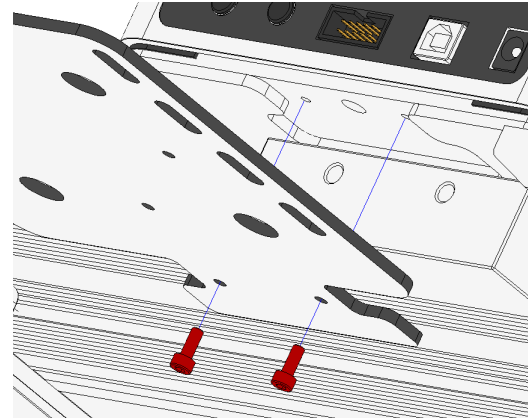
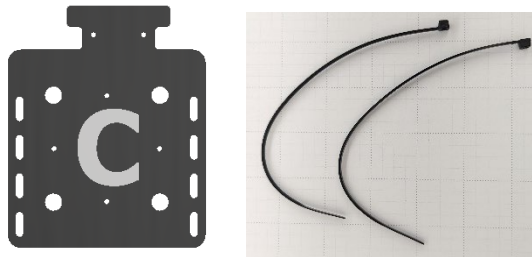
STEP 3 – Connect the free end of the **WIRE DUCT** to **PLATE B** with **ONE** M3x6 Countersink screw.



STEP 4 – For this activity, in order for use to use the vacuum gripper, we will need **PLATE C** to carry the vacuum pump. Attach **PLATE C** onto **PLATE A** with **TWO M3x8** Hex-Socket Head Cap Screws.



STEP 5 – Use the included **ZIP-TIES** to attach the **VACUUM PUMP** to **PLATE C**. It may be necessary to link/chain more than one zip tie in order to make a complete loop around the pump and bracket.



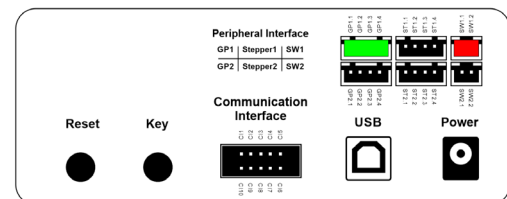
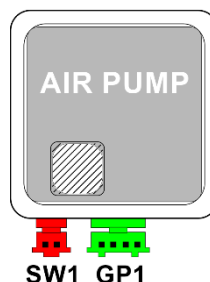
SECTION 2 – WIRING THE ROBOT & LINEAR SLIDE RAIL

If this section is already complete, skip to SECTION 3 – SETUP & PROGRAMMING.



STEP 1 – Wire the AIR PUMP to the Robot as we have done in the past.

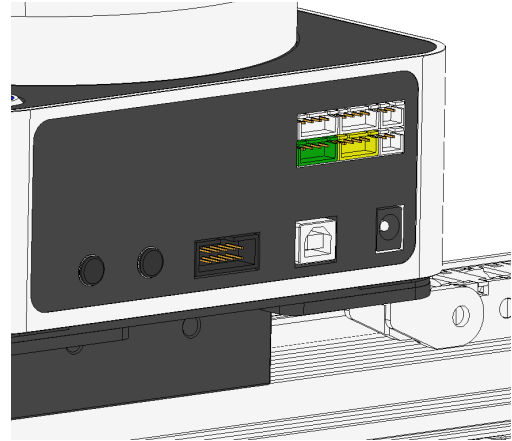
GREEN – GP1
RED – SW1



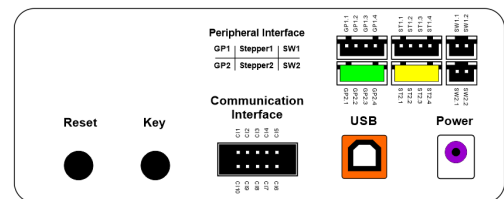
STEP 2 – Connect the four wires from the linear slide rail to the back of the robot. The wires should reach from the wire duct to the robot.

GREEN – GP1

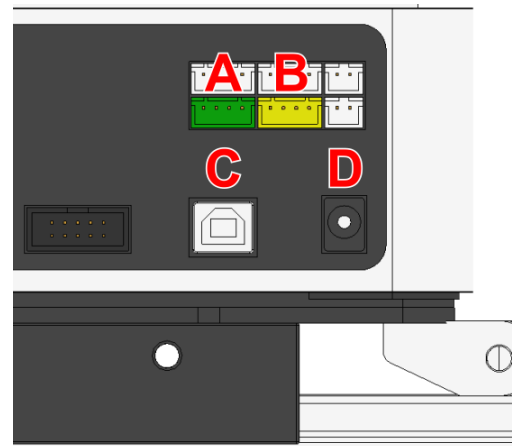
RED – SW1



A		MOTOR INTERFACE
B		HOMING INTERFACE
C		USB CONNECTION
D		12V DC POWER



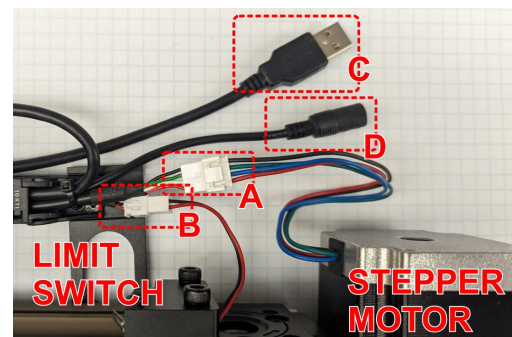
- (A) – The motor interface controls the direction and location of the stepper motor
- (B) The homing cable is attached to the limit switch at the end of the conveyor and reports when the slide base has been homed to a zero location.
- (C) - The USB connection is an extension cable routed through the wire duct and allows the robot to communicate directly with the computer.
- (D) - The 12V DC cable powers both the robot and the linear slide base



STEP 3 – Ensure the stepper motor is connected to the Stepper Motor Extension cable (Four-Wire Connector - Opposite end of cable “A”) and the Limit Switch is plugged into the Homing Extension cable (Two-Wire Connector - Opposite end of cable “B”)

Cable C will go directly to the computer

Cable D will get its power from the robots original DC power connector.



LINEAR SLIDING RAIL SPECIFICATIONS & PAREMETERS

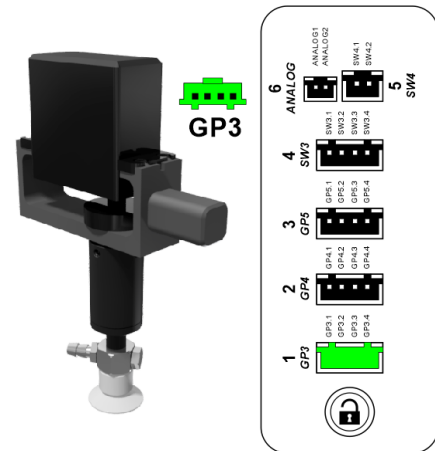
Parameters	Limitations
Maximum Payload	5kg or 11 lbs
Maximum Distance	1000 mm or 39.7 in
Maximum Speed	150 mm/s or 5.9 in/s
Maximum Acceleration	150 mm/s or 5.9 in/s
Repeated Positioning Accuracy	0,01 mm or 0.0004 in
Absolute Positioning Accuracy	0,25 mm or 0.01

SECTION 3 – SETTING UP & PROGRAMMING THE ROBOT & LINEAR SLIDE RAIL

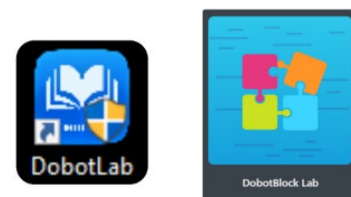


Caution: NEVER wire anything to the Dobot Magician or the Magic Box while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

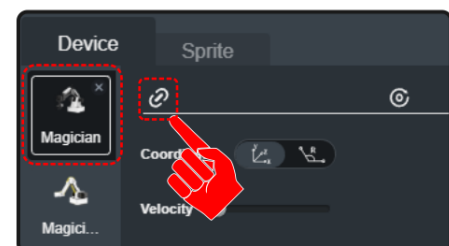
1. Print TWO copies of the Blockly - Pick and Place Field Diagram
2. Set up the robot with a **SUCTION CUP** and Air pump and place a cube in one of the red squares on the field diagram provided.



3. Open up DobotBlock Lab in the software and connect the robot.

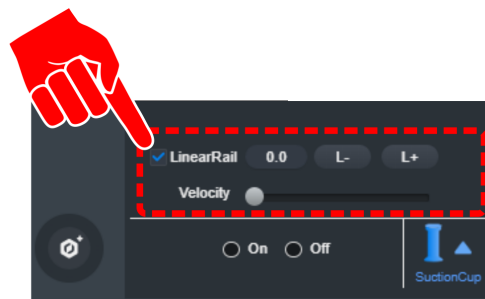


4. Select the connect icon to establish control from the software to the robot. A connection window should pop up.

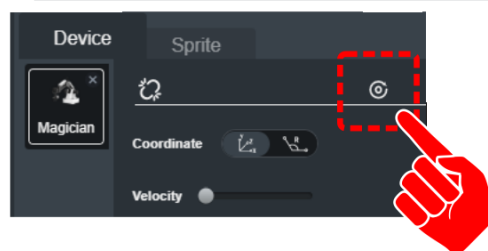




5. Select the CHECK box for the LinearRail BEFORE homing the robot.



6. ENSURE THE AREA IS CLEAR OF OBSTRUCTIONS & LIMIT SWITCH CABLE IS CONNECTED ON BOTH ENDS. When the home icon is selected, the Linear Slide Base will slide all the way to the limit switch (End with the stepper motor). The robot will home after the slide base has homed.

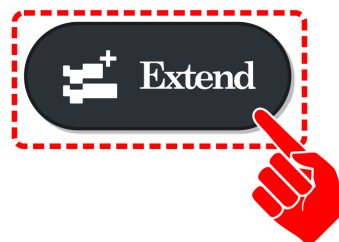


7. Once the Linear Slide Base and Robot are homed, select the L+ icon. The robot should slide away from the stepper motor. The value shown is not actually the distance traveled. Since the motor is a stepper motor, the number shown is a position that the slide base has traveled too.

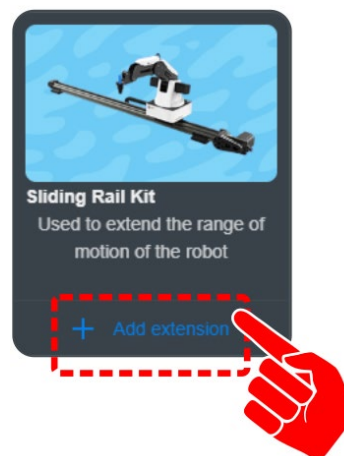


DO NOT ADJUST THE VELOCITY SLIDER

8. For this activity, we will need to add an EXTENSION to use the COLOR SENSOR. Click the EXTEND icon in the bottom left corner.



9. Click on "+ Add extension" for the Sliding Rail Kit.



10. The first step in programming this activity is to set the status of the rail.

V1 – RED Bumpers at the end of each rail
V2 – BLUE Bumpers at the end of each rail



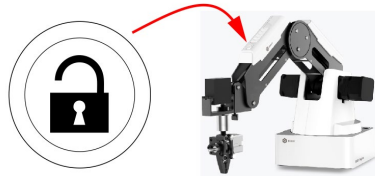
At this time do not use the Set Velocity Block

11. Next, create a simple test program that will send the robot out to location 80mm, wait for 2 seconds, travel back to location 20mm, wait for 2 seconds, and then repeat the loop.

12. Next, add a movement after the robot has reached the 80mm and a different movement at 20mm.

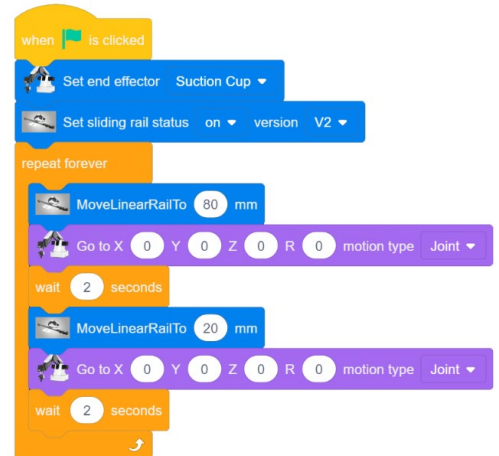
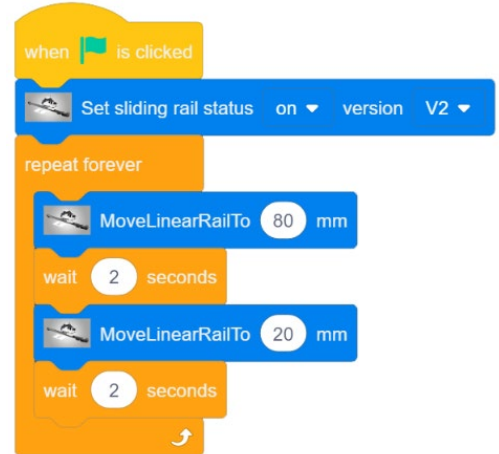
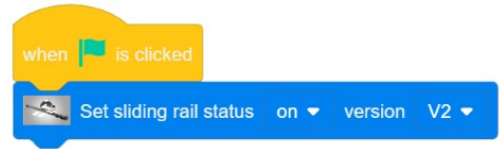


Use the Unlock Arm Control to Grab new locations (any new positions will do for this test) and Fill those Coordinates into the Go To Commands



Sliding Rail Extended Toolbox:

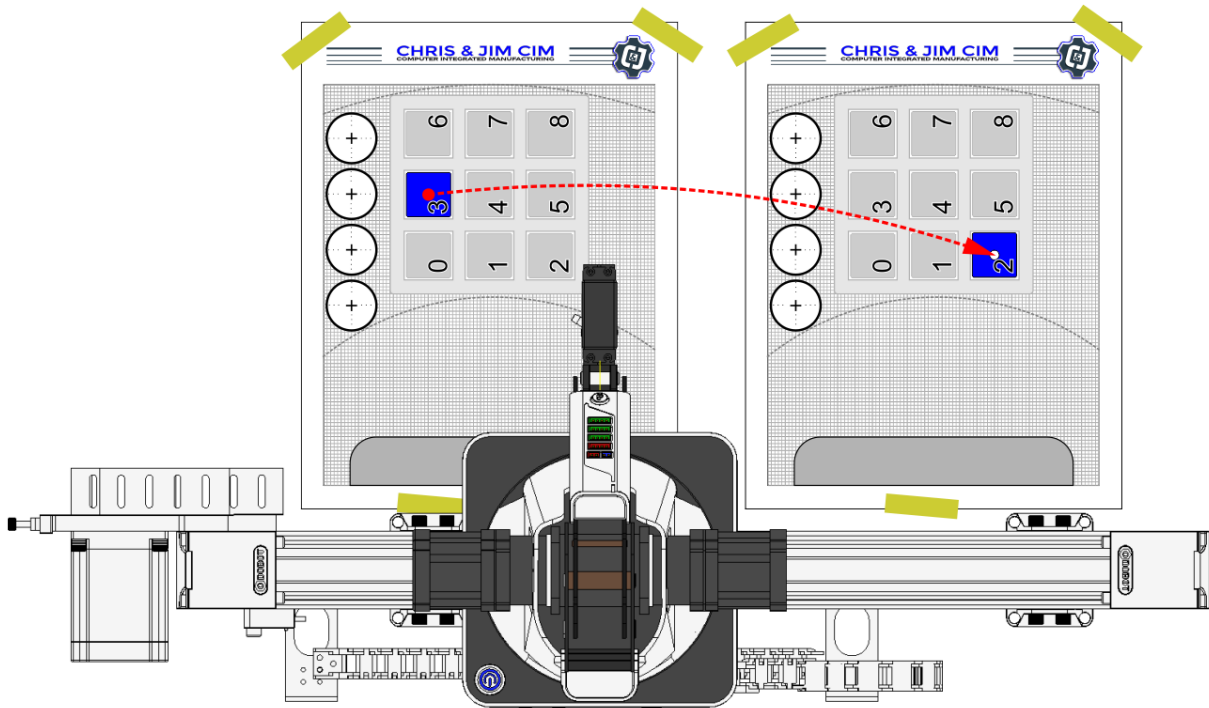
- Set Sliding Rail Status “ON” & “V2”



CHALLENGE

SECTION 4 – PICK AND PLACE USING THE LINEAR SLIDE BASE

1. Tape both field diagrams to the table as shown below (one towards each end of the slide base. See your instructor for specific locations) **NOTE: the slide base shown has been reduced in length*



2. Tape both field diagrams to the table as shown.
3. Create a pick and place activity that will move a block from one field diagram to the other. Get the starting and ending locations from your instructor.

If your set up did not work correctly the first time, what did you have to do to make it work?



CONCLUSION

1. Find a video example of a working sliding base being used in industry. Explain what operation it is doing.
2. How is the slidebase in question #1 similar to the one you are using?
3. How is the slidebase in question #1 different from the one you are using?
4. What was the most difficult part of this activity for you? Why? Explain fully.

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- _____ 1. Move a pallet of 4 blocks from one pallet to the other.

- _____ 2. Set up the robot to “tend a machine”. Use a cardboard box for the machine at one end of the slide, a switch for the machine on/off, a field diagram for the pallet at the other end of the slide, and a cube or cylinder for the part.
 - a. Pick a part off the Pallet
 - b. Place a part in the machine (Box)
 - c. Move the robot out of the machine
 - d. When the machine is done (press the switch)
 - e. Have the robot get the part out of the machine and return it to the pallet at the other end of the slide



4.3 Magic Box - Stack Light Scenario

NAME: _____

Date: _____

Section: _____

INTRODUCTION

What is a stack light?

Stack lights are a crucial part of a manufacturing cell. These devices provide visual feedback to the status of a work cell to operators and personnel. The number of lights and colors vary depending on the application.

Common stack lights consist of three colors:

- **GREEN** – Normal Working conditions
- **YELLOW** – Attention is needed, request for help, or a warning that material may be low.
- **RED** – Indicates that a condition has been met that requires immediate attention. Often comes with an alarm.

Additional Colors

BLUE, CLEAR/WHITE – These colors may indicate various conditions such as the machine is running, machine service required, or a user defined signal.



Humans can trigger **inputs** in a manufacturing cell to start or stop different actions. An output like a **stack light** can provide valuable feedback.



Caution: NEVER wire anything to the Magic Box while it has power on. ALWAYS shutdown the BOX before making connections or damage to the controller could occur. Be sure to ask your instructor if you have any questions.

KEY VOCABULARY

- | | |
|-------------------------|----------------|
| • Stack Light | • PIR sensor |
| • 2-Button Input Module | • Human Sensor |
| • Magic Box | • Input |
| • Output | |

EQUIPMENT & SUPPLIES

- | | |
|-----------------------------------|-------------------------------------|
| • Dobot Magic Box | • Magician Robot Arm w/ Suction Cup |
| • Digital - 2-Button Input Module | • 12V Power Cable and Supply |
| • Digital – PIR Sensor | • Common Sensor Cables |
| • LED Module | • USB C to A Cable |

KEY CONCEPTS

Essential questions answered in this activity include:

- What is a stack light and how is it used in an industrial setting?
- How does the magic box connect with the robot and inputs and outputs?
- How can the PIR sensor be used for in a manufacturing cell?
- What other sensors can be used for safety in a manufacturing cell?

PROCEDURE

SECTION 1 – WIRING the MAGIC BOX, SENSORS, & MAGICIAN ROBOT

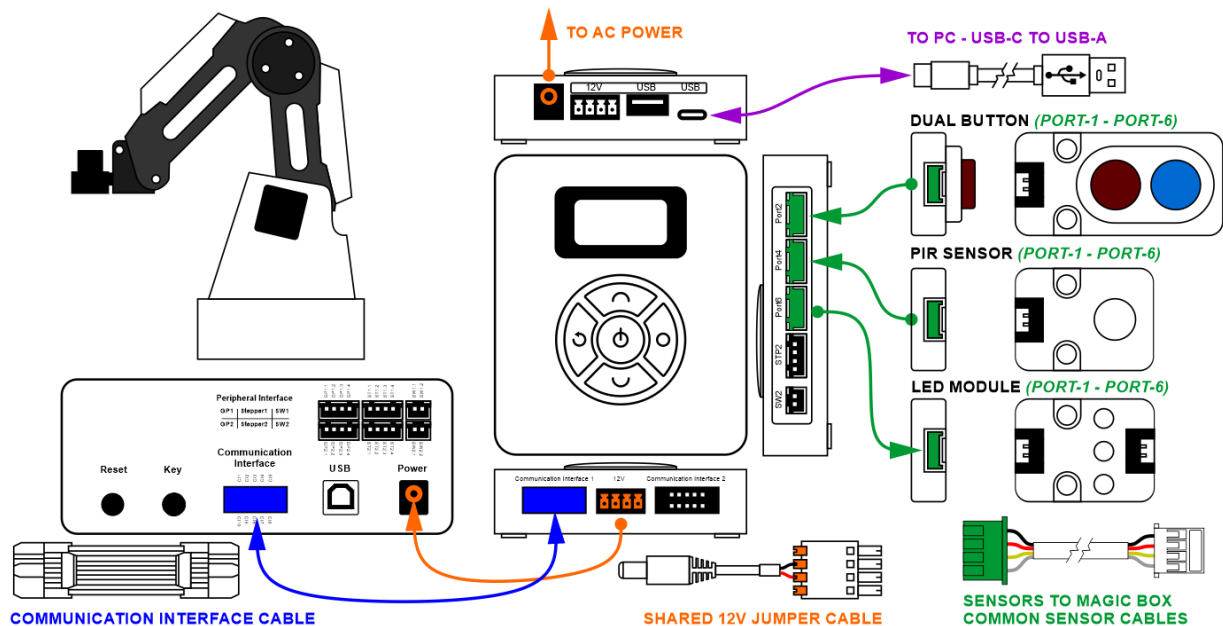


Caution: NEVER wire anything to the Magic Box or Robot while it has power on. ALWAYS turn them off before making connections or damage to the Controllers could occur. Be sure to ask your instructor if you have any questions.

1. For this activity, you will need:

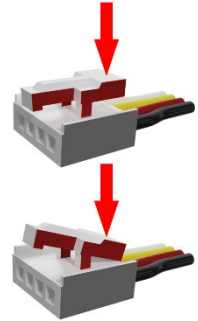
- 1x Magic Box
- 1x USB-A to USB-C Cable
- 1x AC/DC Power Adapter (12V)
- 3x Common Sensor Cables
- 1x DUAL BUTTON – **PORT 2**
- 1x PIR SENSOR – **PORT 4**
- 1x LED MODULE – **PORT 6**

2. Wire the Magic Box as shown below



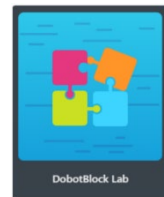


Caution: MAKE SURE THE TAB IS PRESSED DOWN ON THE WHITE CONNECTOR WHEN DISCONNECTING ALL SENSORS!!! It is very easy to damage the sensors if the white connector is pulled or tugged on without pressing down the small white tab to release the cable from the sensor housing.

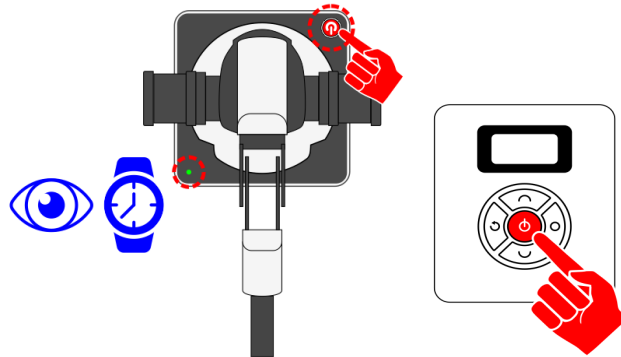


SECTION 2 – CONNECTING the MAGIC BOX & ROBOT to DOBOTLAB

1. Open up DobotBlock Lab in the software.



2. Once **ALL** the wiring is done (sensors connected and robot arm connected), power ON the Magic Box and Robot. **WATCH** and **WAIT** for the robot to completely power on (GREEN INDICATOR)



3. Repeat the connection process completed in previous activities.
 - a) Connect to the Magic Box in DobotLab.
 - b) Add the AI Sensor Kit and Magician Extensions.
 - c) Ensure the robot's envelope is clear and home the robot.

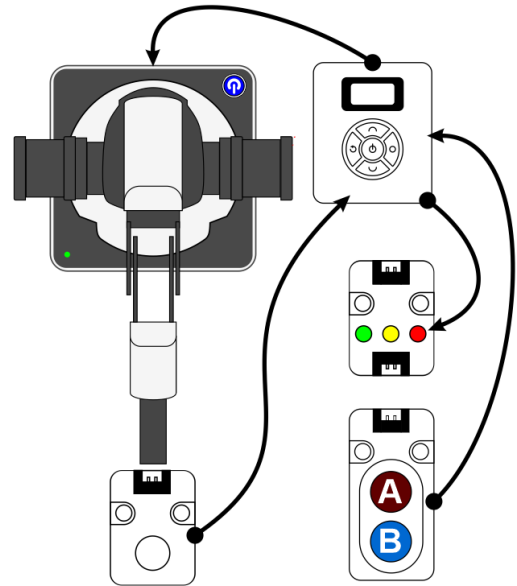


CHALLENGE

SECTION 3 – CREATE A STACK LIGHT PROGRAM

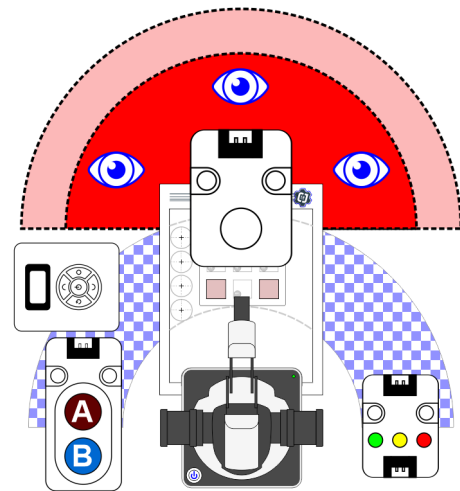
For this challenge, develop a pick and place routine that adds:

- A push button (B-BLUE) to start each complete loop (block in place)
- A stack light for visual feedback
 - GREEN ONLY - Normal Operation - Moving
 - GREEN & YELLOW Waiting for Block (B-BLUE Push Button)
 - RED ONLY – Motion detected while in operation
- A PIR sensor that detects movement around the work cell environment that stops the primary operation (only while in operation)
- A push button (A-RED) to restart the process from warning



At this time, using Blockly, a code does not exist that will stop motion once it has been started (GO TO or JUMP). For this simulation, the program will only look for

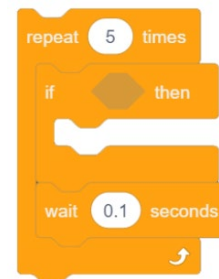
motion (from the PIR) inside the work envelope in-between each motion. This is an excellent opportunity to develop a MY BLOCK that can be placed throughout the program.



THINGS TO THINK ABOUT:

- What variables would be helpful? (both to simplify the program but also to debug it if it is not working as planned)
- What MY BLOCKS would be helpful (break up the program into sections... waiting?)
- Use of JUMPS to cut down on motion blocks... but... does that take away the places to insert "look for motion"?

- Possibly try something like the code below to try to catch motion?
 $5 \times 0.1 = 0.5$... half a second?



If your set up did not work correctly the first time, what did you have to do to make it work?

CONCLUSION

1. *Why was the program unable to stop the robot while it was in motion?*
2. *What other sensors/inputs from the kit could be used to add safety in a manufacturing cell, and how would they be used?*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- | | |
|-------|--|
| _____ | 1. Add the servo as an output in the work envelope to simulate dropping a safety shield while the robot is in motion. |
| _____ | 2. Replace the 2-Button Input module with the gesture sensor. Make “swipe left” do blue button, and “swipe right” do red button. |



4.4 Block - Camera Vision System Item Detection

NAME: _____

Date: _____

Section: _____

INTRODUCTION

Adding a camera to the Magician robot adds vision to the robot. **Vision** systems allow robots to “see” and can be used to help an industrial robot identify, navigate, inspect, handle parts, or complete tasks. This adds a new level of intelligence to the robot allowing it to make more complex choices.

In this activity, the robot will be taught to identify some objects after **calibration**, then see how accurate it is at finding them. It will also use voice to tell us what it has found.



WARNING



Power OFF

Caution: NEVER wire anything to the Magician or the Magic Box while it has power on. ALWAYS shutdown the ROBOT before making connections or damage could occur. Be sure to ask your instructor if you have any questions.

KEY VOCABULARY

- Calibration
- Picture/Image Recognition
- Training Model
- Audio Feedback

EQUIPMENT & SUPPLIES

- Dobot AI USB Camera
(Any USB color camera)
- Analog – Potentiometer Module
- 12V Power Cable and Supply
- Magician Robot Arm
- Suction Cup Gripper
- 4 Colored Cubes: Red, Green, Blue, Yellow

KEY CONCEPTS

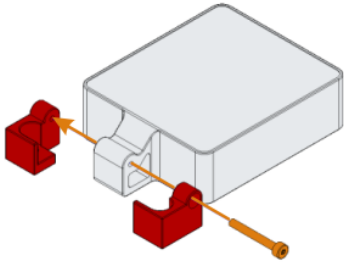
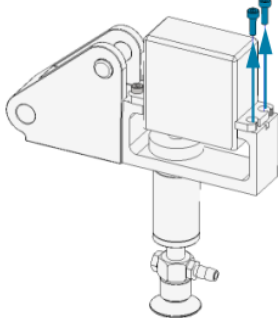
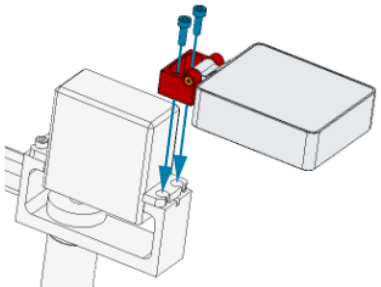
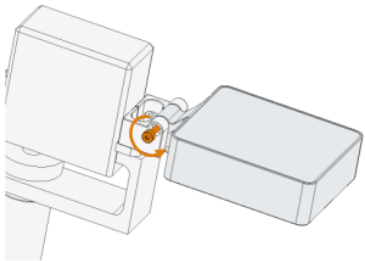
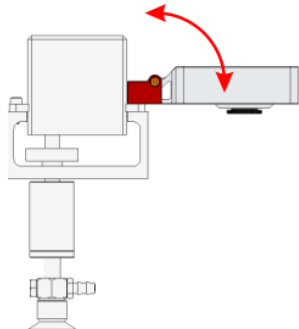
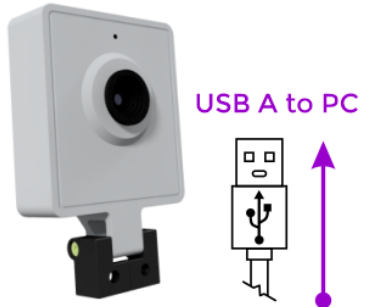
Essential questions answered in this activity include:

- What are the advantages of adding vision to a robot?
- Why does the robot need to be taught (shown) multiple images of the same object?
- Is the current vision system seeing color only?

PROCEDURE

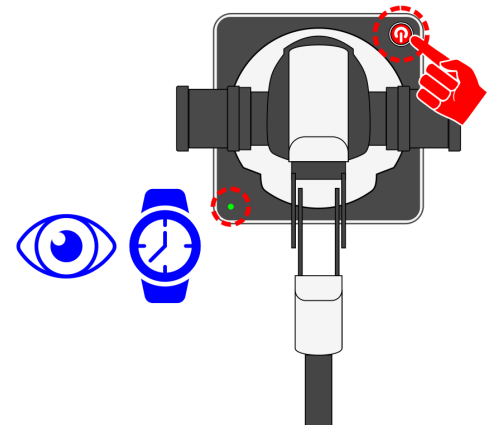
SECTION 1 – CONNECTING the CAMERA to the MAGICIAN

1. Install the camera on the Dobot Magician (*Any USB camera can be used, be creative in mounting it.*)

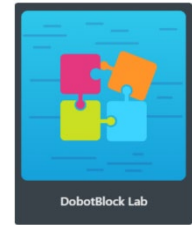
① Attach the Mounting brackets to the camera	② Remove the mounting screws from the servo bracket	③ Reuse the screws to attach the camera bracket to the servo
		
④ Loosen the screw before adjusting the camera angle	⑤ Adjust the camera angle and tighten the screw	⑥ Connect the camera to the computer using the attached USB cable
		

SECTION 2 – CREATING THE CLASSIFICATIONS and TRAINING the MODELS

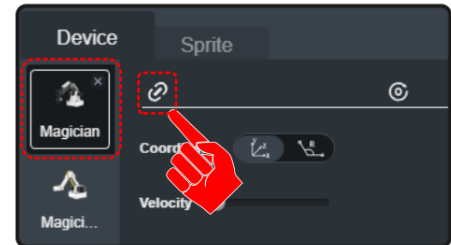
1. Once the USB camera is mounted to the robot and the USB cable is plugged into the computer, power ON Robot. **WATCH** and **WAIT** for the robot to completely power on (GREEN INDICATOR)



1. Open up DobotBlock Lab in the software and connect to the robot.



2. Follow the same process from previous activities to add the Magician as a device and connect the software to it (establish communication).

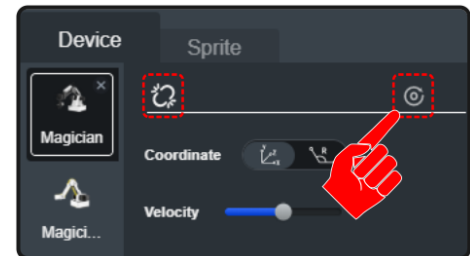


3. Once the robot is connected, we need to home the robot to set its home position.

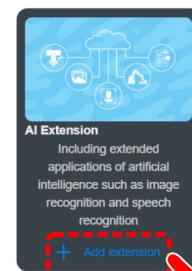


Before **HOMING** the robot, make sure the robot's **WORK ENVELOPE**, the area in which the robot can reach, is clear.

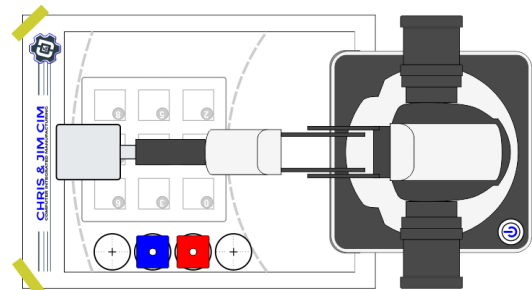
Select the **HOME** icon to start the robots homing process. **HOMING** the robot will return and set the robot to its initial **HOME** position.



4. Click on "+ Add extension" for the AI EXTENSION.

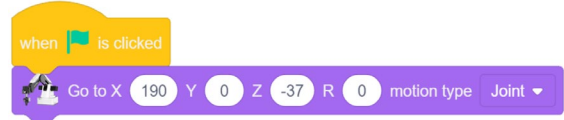


5. Tape a Field Template down to the table.

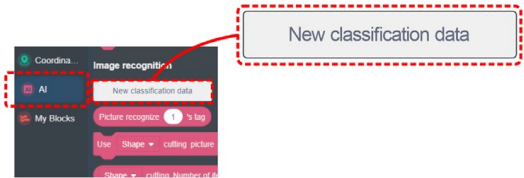


- Send the Robot to a known position to use the camera.

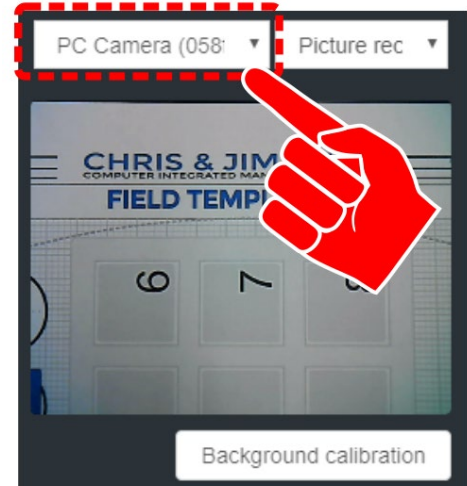
For this activity use coordinates similar to:
X190 Y0 Z-37 R0.



- Click on NEW CLASSIFICATION DATA from the AI category in the block toolbox.



- Click the drop down for the camera device selection. Choose the Camera attached to the arm.



- The background used in this activity is not a plain white field. Choose Background calibration. This will allow the software to ignore the background as it learns the images during training.



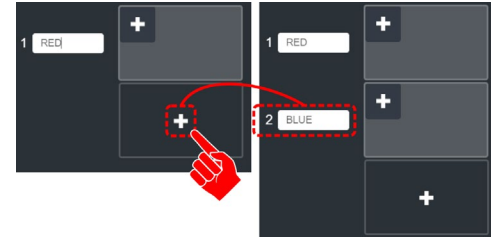
When the background of the camera is not pure white, click this button to calibrate the background to improve the automatic recognition rate. Note: pure white background is always the best choice. Before calibration, please make sure that there is no debris in the camera's field of view. Do not place the objects to be trained in the field until after calibration.



- Label the first classification as RED.

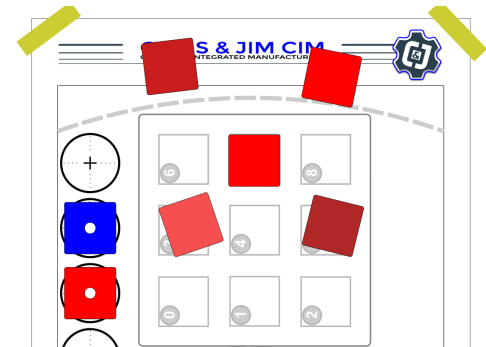
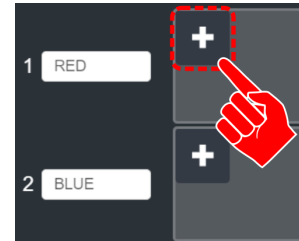


11. Click the Plus in the CENTER of the next window to create a second classification. Label the second class as BLUE.

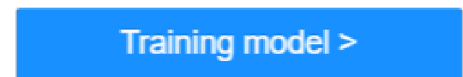


The software now needs to be taught how to distinguish the difference between the two classifications.

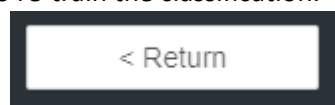
12. Place the red cube within the camera's field of view. Click the PLUS in the top left of the RED class window. A picture will be taken and stored. Move the block to a new location in the field and click the plus again. This will create a second image for the software to use for comparison. Take at least three different images. Try to vary angle, location, different faces, different lighting, and different shadows (any condition that could be present during evaluation).



13. Repeat the process for the BLUE class.
14. Now that the robot has resources to compare against, click the TRAINING MODEL to move to the next phase (testing).
15. Place RED or BLUE cube in the camera's vision. The goal is to have model matching that is close to 100%. Click FINISH to complete the model training.



There can be a delay in switching between classes. If the model matching fails or is not near 100% for each class, click <Return, recollect images, change the lighting, or add additional pictures to re-train the classification.

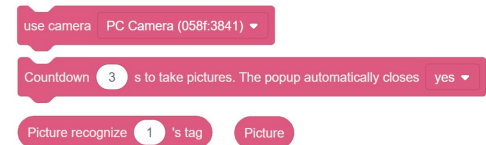


SECTION 2 – TESTING the MODELS with BLOCK CODE

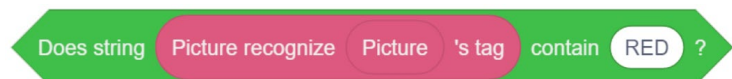
SKILL BUILDER 1 – COMPARING NEW IMAGE TO STORED DATA (MODELS)

Now that the classes have been created and the models have been trained, create a block program to test the accuracy of the setup.

1. Create two variables (RED & BLUE)
2. This program will use a few new blocks from the AI Extension category.
 - A. Camera Category – USE CAMERA
 - B. Image Acquisition – COUNTDOWN to TAKE PICTURE
 - C. Image Acquisition – PICTURE
 - D. Image Recognition – PICTURE RECOGNIZE

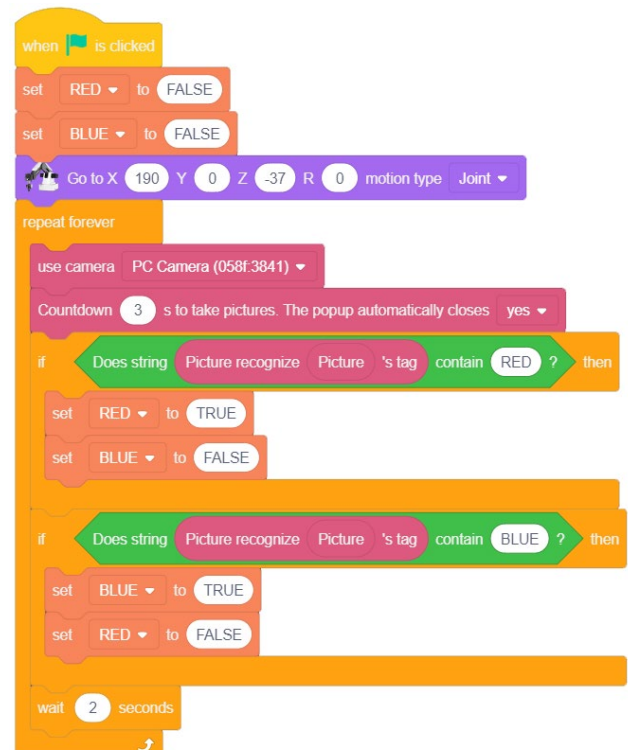


3. The program will also require one new block from the OPERATORS category
4. Link the blocks together as seen to the right.



Do not use a variable, be sure to just type “RED” or “BLUE”.

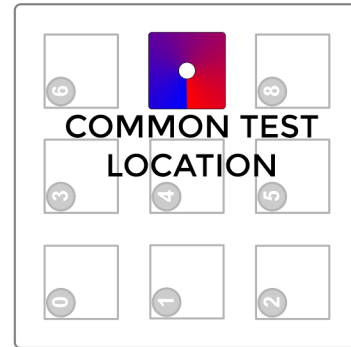
- This series of blocks will create a condition that decides if the picture it takes fits into the stored model data for the RED classification.
5. Using the conditions created, the variables, and a few if statements, create the program to the right that will test the models saved.
 - A. Initialize the Variables as False.
 - B. Send the robot to or near its HOME location.
 - C. Choose which camera to use.
 - D. Pause for three seconds and take a picture (the camera popup window will automatically pop up and take the picture. The new picture is stored as “PICTURE”.
 - E. IF Statement – Ask the question if the new “PICTURE” recognizes/matches the ones stored in the RED class or BLUE class. When they match, set the matching variable to TRUE and display that value for 2 seconds before looping and retesting the model.



Run the program and test the stored models for the RED and BLUE cube.



What if there were no block present?
Could you make a new classification called
“none” and teach it what no block looks
like? Try it!

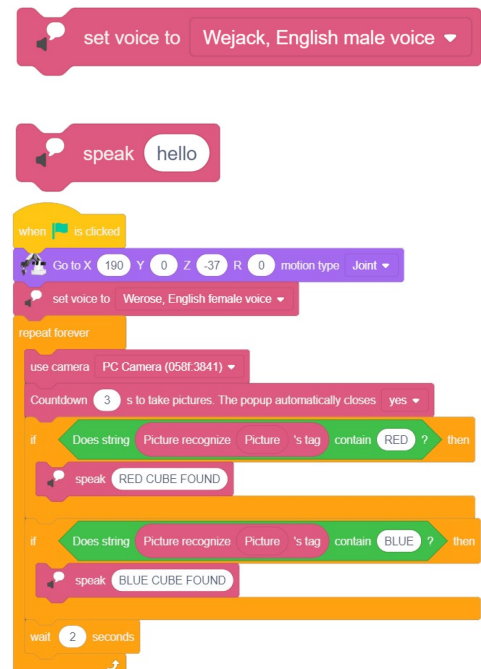


If your set up did not work correctly the first time, what did you have to do to make it work?

SKILL BUILDER 2 – ADDING AUDIO FEEDBACK

Providing user feedback can be a valuable part of programming. Up to this point, the programs have used variables and or LEDs to display visual feedback. This skill builder will show how to add **AUDIO FEEDBACK** to the program. To do this, the program will new a few blocks from the AI Extension category.

1. Drag over the SET VOICE TO and the SPEAK blocks from the Text to Speech section of the AI Extension category.
2. In order to keep the program small, remove the variables.
3. Add the Set voice to the header of the program.
4. Replace the variables with the SPEAK Block.



Run the program and test the stored models for the RED and BLUE cube with audio feedback.

If your set up did not work correctly the first time, what did you have to do to make it work?

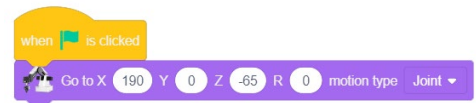
SKILL BUILDER 3 – IMAGE RECOGNITION with NON-CUBE OBJECTS

It may appear that this program is only comparing the color of the cubes. In reality, the images are also comparing things such as size, shape, form, objects with multiple colors and details. They are full color photos that the program is using to build a 3D picture of what it is looking for. This skill builder will test this capability.

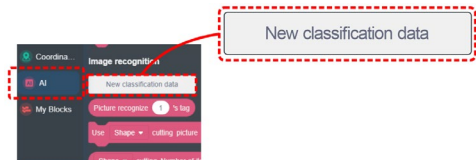
1. For this portion of the activity, lower the Camera's Z to take larger sampling image of the objects & replace the template with a piece of white paper.

Use coordinates similar to:

X190 Y0 Z-65 R0.



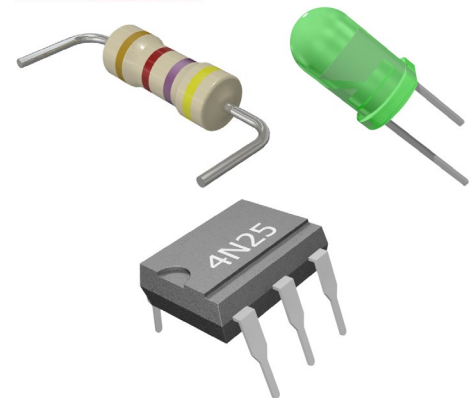
2. Save the current file and start a new one. Create a NEW CLASSIFICATION DATA model.



3. Create THREE classifications
A. Resistor B. LED C. Isolator

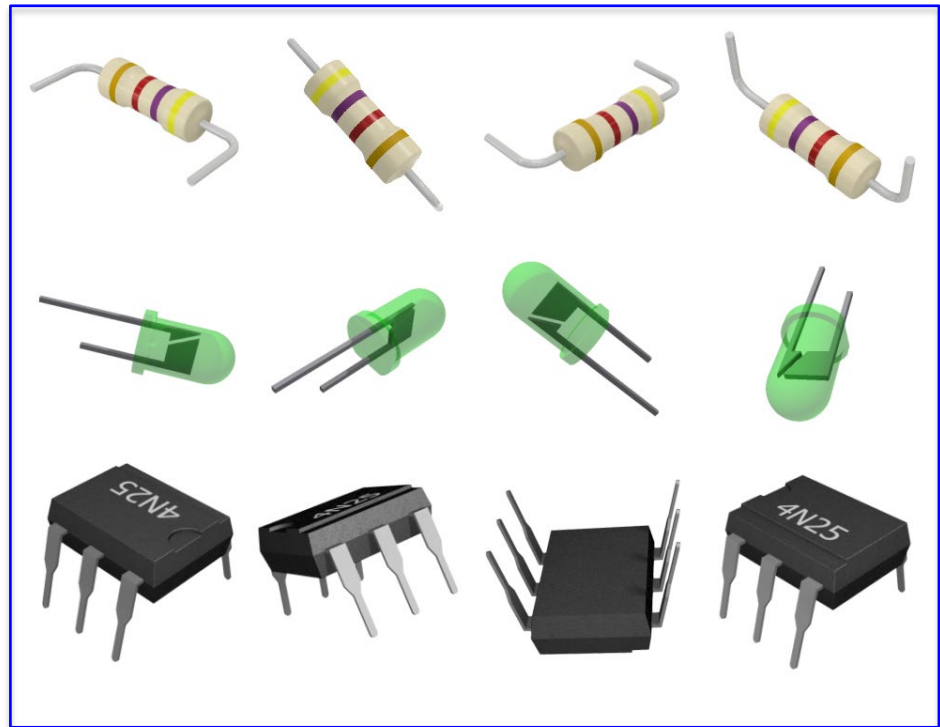
4. Repeat the process used for the cubes with the electrical components from the handshaking activities. Again, use different angles, orientations, and locations (the more photos taken, the more data the software has to use for comparison).

5. Use either Variables (visual feedback) or Speech (audio feedback) to test the program's image recognition capabilities.

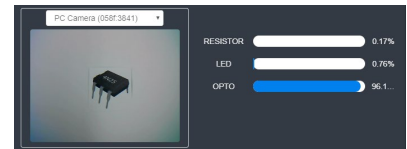
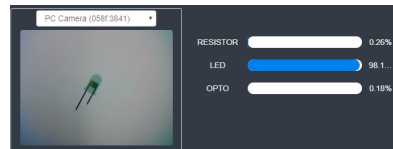




Print and cut out the images to the right and use them as the reference images (remove the blue border). The images can be used more than once if additional data is needed, just rotate or move the images.



Continue reteaching the models as needed to get consistent results in the 90% or better range.



Run the program and test the stored models for the electrical components.

If your set up did not work correctly the first time, what did you have to do to make it work?

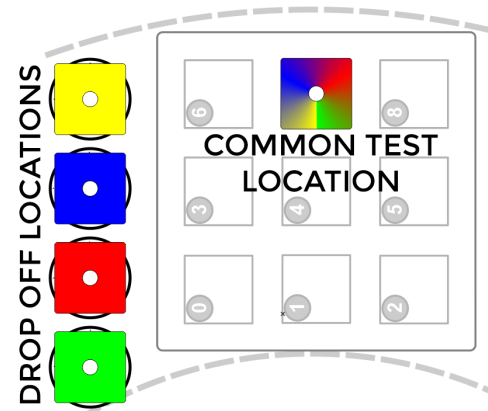


CHALLENGE

SECTION 3 – USE the CAMERA to SORT 4 DIFFERENT COLOR CUBES

Develop a program that will test four different cube colors using the camera.

- A. Start at a HOME location.
- B. Send the robot to a common test location.
- C. Test the color of the cube with the camera.
- D. Move to a pick-up location.
- E. Pick up the cube and place it in a designated drop off location.
- F. Loop the program for another cube.



CONCLUSION

1. *Is the camera sensing color or shape only? One or the other? How do you know?*
2. *If you printed the electrical components in black and white, would it still work?*
3. *What percentage was the accuracy for your robot in identifying objects? If it was lower than 100%, what could you do to increase its accuracy?*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor, and have them initial on the line.

- | | |
|-------|---|
| _____ | 1. Complete the skill builder above and have your instructor check your work. |
| _____ | 2. Complete the Challenge above and have your instructor check your work. |
| _____ | 3. What happens when there is nothing in the camera's field of view? Can you make the robot tell you that there is nothing there? How? |
| _____ | 4. Get three or four different electronic components from your instructor. The actual objects, not photographs. Teach the classifications and see if the robot can identify the actual objects. |
| _____ | 5. If you have the sensor kit, teach it three different sensors. (like the Potentiometer, the light sensor, and the PIR sensor) Can it accurately see the difference? |



4.5 - TP/Block

Work Cell Design – Without Microcontroller

NAME: _____

Date: _____

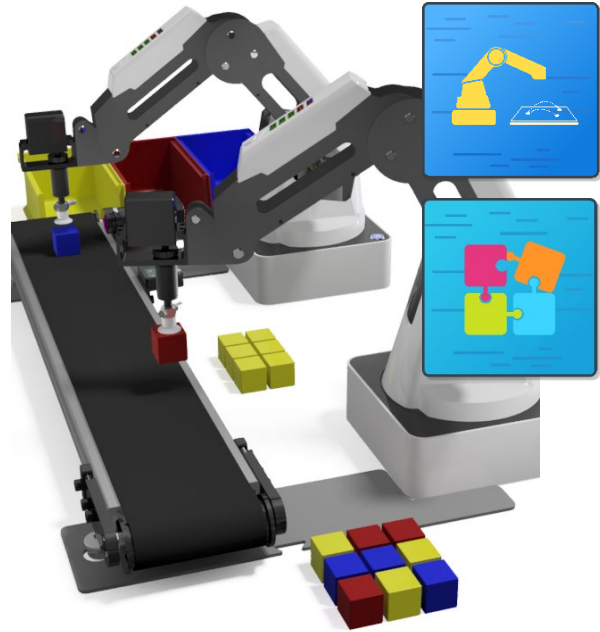
Section: _____

INTRODUCTION

A robotic workcell is defined as the complete environment around a robot. This environment may include tools, machines and/or other robots. This environment may include tools, machines and/or other robots and perform a variety of different manufacturing processes as well as moving parts around.

In this activity you will use a robot and a microcontroller system to recreate a workcell. Your workcell will incorporate all of the devices and skills that you have learned about in previous activities including:

- Inputs & outputs
- Sensors
- Machines
- Robots



KEY VOCABULARY

- Handshaking
- Workcell
- Output
- Nesting
- Optical Isolator
- Sensor
- Input
- Palletize

EQUIPMENT & SUPPLIES

- 2 Robot Magicians
- Dobot Pneumatic Gripper or Suction Cup
- x1 Dobot Conveyor System
- x1 Dobot Color Sensor
- x1 Dobot IR Sensor
- Dobot Colored 1 Inch Cubes
- x3 Small Bins for Sorting
- Input Limit Switch
- DobotLab software
- Dobot Input/Output Manual
- 4-Wire Extension Cables or [3-Wire Servo Extensions Cables](#)
- Handshaking Modules or [Breadboard/wire/4N25 Optical Isolator and 100 ohm & 4.7K ohm resistors](#)

PROCEDURE



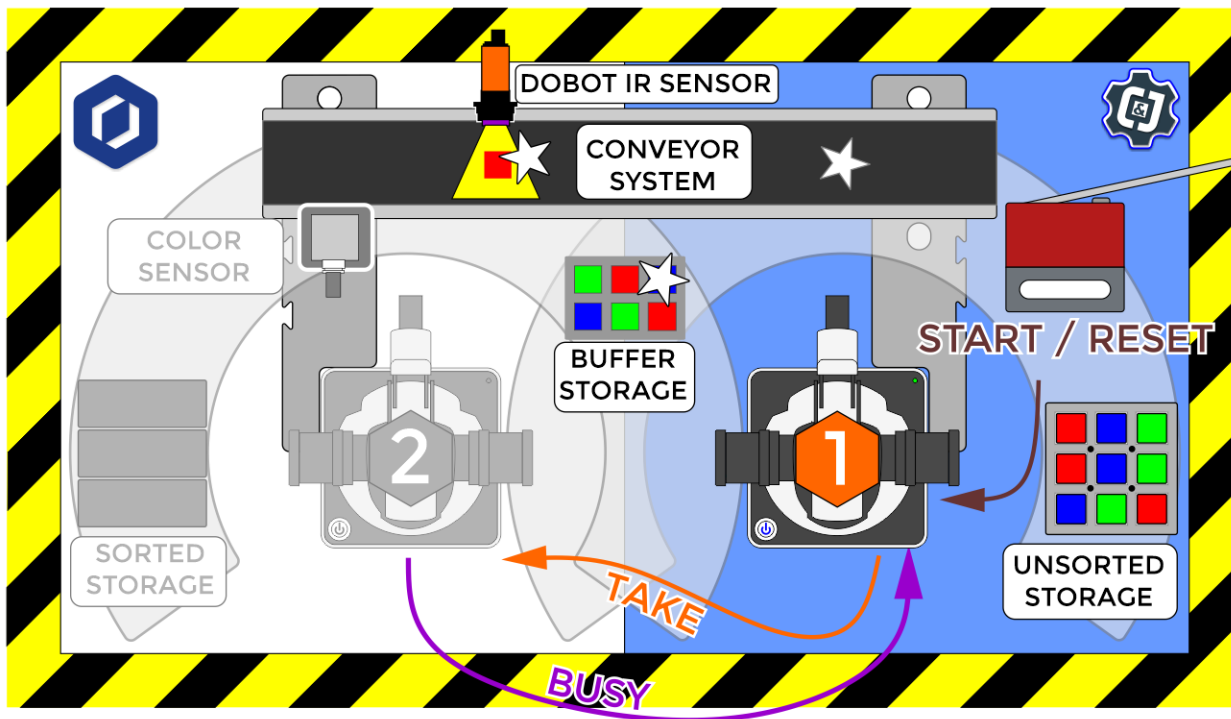
WARNING Power OFF

Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Create a fully integrated **WORKCELL** using the following scenario.

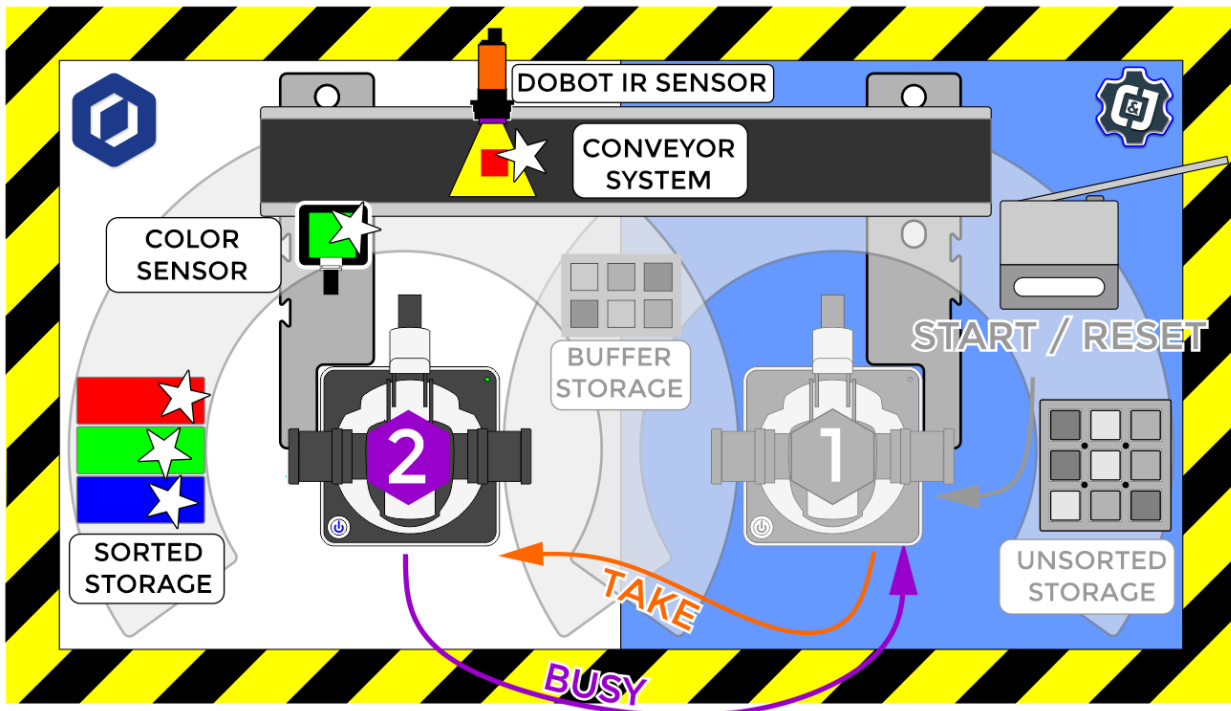
DOBOT MAGICIAN - 1

- Primary sequence is initiated by a **LIMIT SWITCH** that sends a signal to **ROBOT 1**
- Once the sequence is started, **ROBOT 1** will look to see if the **UNSORTED MATRIX** of cubes is empty.
 - If **UNSORTED STORAGE MATRIX** is **NOT** empty, **ROBOT 1** will take one cube at a time and place it onto the conveyor. This process will repeat until the matrix is empty.
 - If **UNSORTED STORAGE MATRIX** is empty (all nine cubes have been sent to sorting), **ROBOT 1** will take parts from the matrix of cube parts in the **BUFFER STORAGE**. **ROBOT 1** will check to see if the **UNSORTED STORAGE MATRIX** has been refilled (**START/RESET** switch has been pressed again) after each part is taken from the **BUFFER STORAGE**.
- After a part is placed on the **CONVEYOR** system, **ROBOT 1** will return home
- **ROBOT 1** checks if **ROBOT 2** is sending a busy signal
 - If **ROBOT 2** is sending a **BUSY** signal, **ROBOT 1** will wait
 - If **ROBOT 2** is **NOT** sending a **BUSY** signal, **ROBOT 1** will start the **CONVEYOR** system.
- **ROBOT 1** will stop the **CONVEYOR** system when an object is detected by the **IR SENSOR**
- **ROBOT 1** will send a **PICK UP** signal to **ROBOT 2**. Continue sending the signal until **ROBOT 2** sends back a **BUSY** signal
- Once the busy signal is received, **ROBOT 1** will loop back to pick another cube
- **ROBOT 1** will take a cube from the **UNSORTED STORAGE MATRIX** and place it in the **CONVEYOR** system.



ROBOT 2 MAGICIAN - 2

- **ROBOT 2** waits for signal
 - If NO **PICK UP** signal is received from **ROBOT 1**, **ROBOT 2** waits
 - If a **PICK UP** signal is received from **ROBOT 1**, **ROBOT 2** will start sending a **BUSY** to **ROBOT 1**. **ROBOT 2** will then pick up the cube sitting on the **CONVEYOR** system at the **IR SENSOR**.
- **ROBOT 2** will evaluate the cubes color by hovering it about 3mm above the **COLOR** sensor
 - If the color is **GREEN**, the robot will place the cube in the **GREEN STORAGE**
 - If the color is **BLUE**, the robot will place the cube in the **BLUE STORAGE**
 - If the color is **RED**, the robot will place the cube in the **RED STORAGE**
- **ROBOT 2** will return home and stop sending the **BUSY** signal to **ROBOT 1**.
- **ROBOT 2** will then loop back to waiting for the **PICK UP** signal from **ROBOT 1**.



Be sure to consult the Dobot Input/Output manual if you want to use other inputs and outputs, as damage to your robot or your other equipment may result.

Note any additional parameters that are provided by your instructor. Take notes in the space below.



CONCLUSION

Answer the questions below and hand-in electronically to your instructor.

1. *What are the inputs you used in your workcell?*
2. *What are the outputs you used in your workcell?*
3. *What was the hardest part of this activity for you? Explain how you made it work.*
4. *Describe the process that you and your team went through to complete this project by inserting at least 5 pictures, with captions, that you took throughout the project. Be sure to include programming, designing, wiring, building, and testing.*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.



Be sure to consult the Dobot Input/Output manual if you want to use other inputs and outputs and consult with your instructor before turning the robot on, as damage to your robot or your other equipment may result.

- _____ 1. *Replace the human component of starting the system with a sensor. What sensor did you use and why?*
- _____ 2. *Could you use other sensors in this project? Which ones, and how would you use them? Check with your instructor once you come up with an idea, and be sure you get approval before you try it.*
- _____ 3. *Integrate a stack light with your robot so that you know when the signals are being sent.*
- _____ 4. *Incorporate the Camera from the Sensor Kit and have robot two announce what is going into the bins... (example: red block = case of wrenches, yellow block = case of screwdrivers etc.) nt.*



4.6 - TP/Block

Work Cell Design – With Microcontroller

NAME: _____

Date: _____

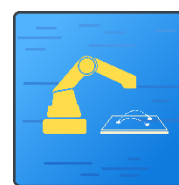
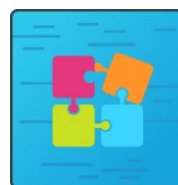
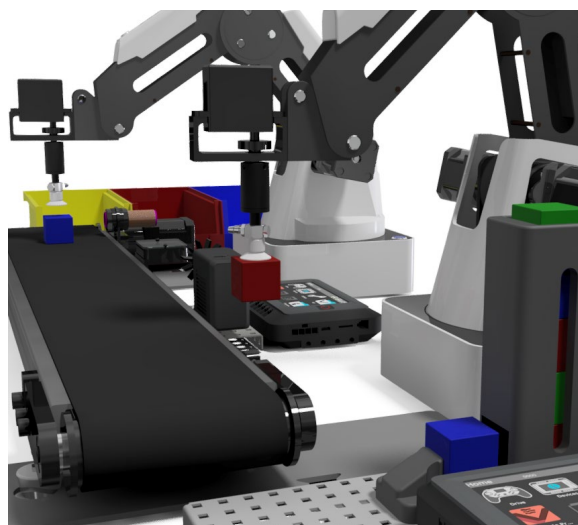
Section: _____

INTRODUCTION

A robotic **workcell** is defined as the complete environment around a robot. This environment may include tools, machines and/or other robots. This environment may include tools, machines and/or other robots and perform a variety of different manufacturing processes as well as moving parts around.

In this activity you will use two robots and a part feeder using a **microcontroller** system, a conveyor, and sensors to create a workcell. Your **workcell** will incorporate all the devices and skills that you have learned about in previous activities including:

- Inputs & outputs
- Sensors
- Machines
- Robots



Due Date: _____

KEY VOCABULARY

- | | |
|---------------|--------------------|
| • Handshaking | • Optical Isolator |
| • Workcell | • Sensor |
| • Output | • Input |
| • Nesting | • Palletize |
| • Conveyor | • Microcontroller |

EQUIPMENT & SUPPLIES

- 2 Robot Magicians
- Dobot Pneumatic Gripper or Suction Cup
- x1 Dobot Conveyor System
- x1 Dobot Color Sensor
- x1 Dobot IR Sensor
- Dobot Colored 1 Inch Cubes
- x3 Small Bins for Sorting
- Microcontroller System & Components
- Input & Output devices
- DobotLab software
- Microcontroller Software
- Dobot Input/Output Manual
- 4-Wire Extension Cables
(Option: 3-Wire Servo Extensions Cables)
- Handshaking Modules
(Option: Breadboard/wire/4N25 Optical Isolator and 100 ohm & 4.7K ohm resistors)

KEY CONCEPTS

Essential questions answered in this activity include:

- How do you integrate robots and other parts of a workcell to complete a given task?
- How do you safely communicate between a microcontroller and a robot?
- What are the different types and styles of inputs and outputs needed to complete your given tasks?
- Which end of arm tooling is most appropriate for your workcell?
- Where is it appropriate to use the jump command within the workcell?
- What other software & hardware will I need to complete this task?



PROCEDURE



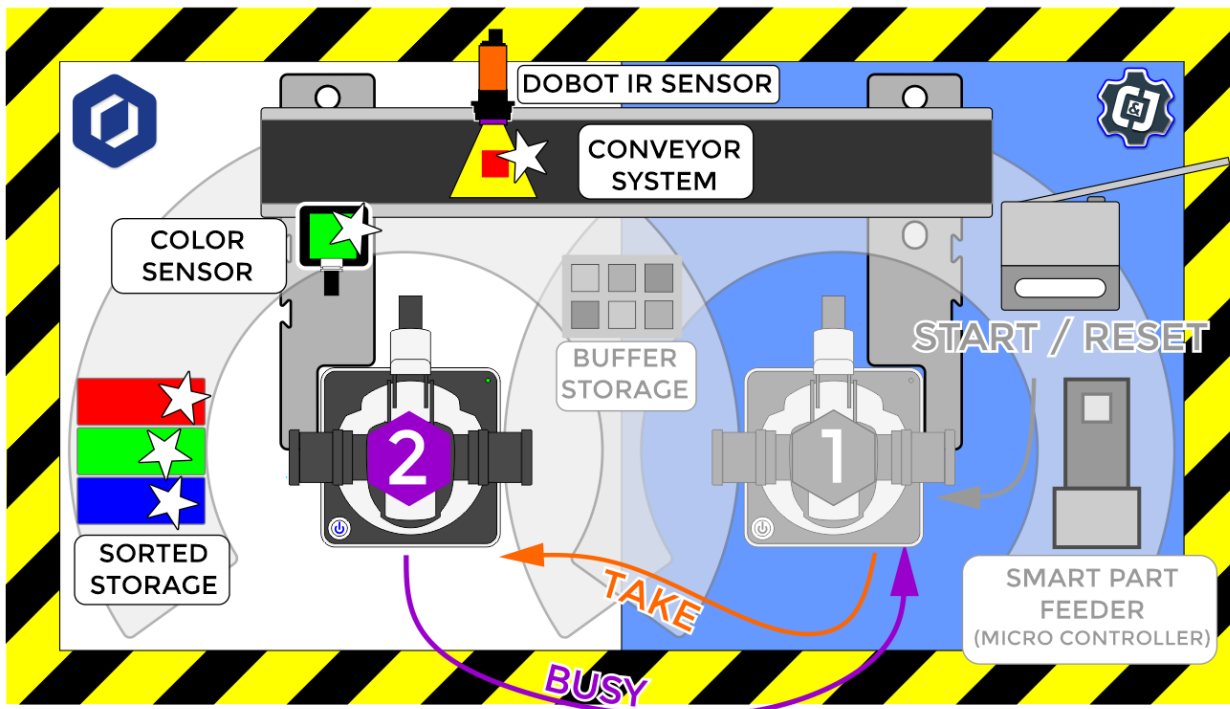
WARNING Power OFF

Caution: NEVER wire anything to the Dobot Magician or the Magic Box while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. Create a fully integrated **WORKCELL** using the following scenario.

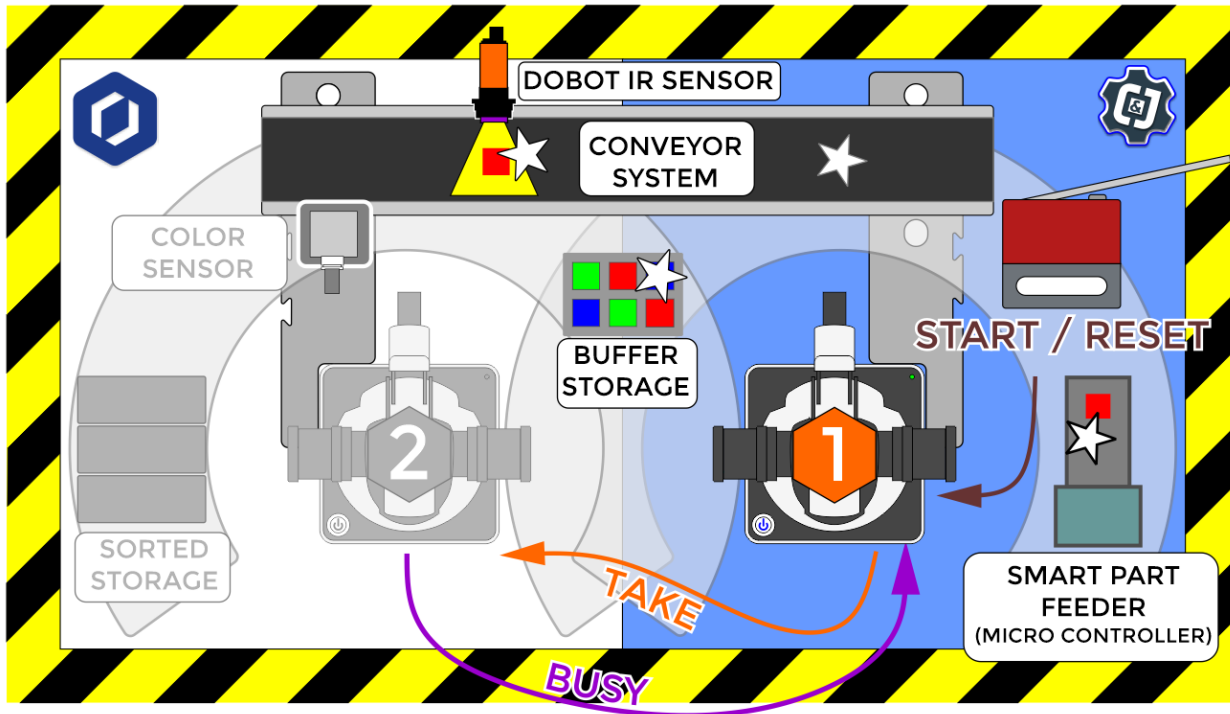
DOBOT MAGICIAN - 1

- Primary sequence is initiated by a **LIMIT SWITCH** that sends a signal to **ROBOT 1**
- The smart part **FEEDER** sends an input signal to **ROBOT 1** if the feeder is empty
 - If **NO** signal is received from the **FEEDER**, **ROBOT 1** will take a part from the feeder and place it on the **CONVEYOR** system
 - If a signal is received from the **FEEDER** (empty), **ROBOT 1** will take parts from the matrix of cube parts in the **BUFFER STORAGE** until the **FEEDER** is no longer empty (check for **FEEDER** status at the beginning of each **BUFFER STORAGE** loop).
- After a part is placed on the **CONVEYOR** system, **ROBOT 1** will return home
- **ROBOT 1** checks if **ROBOT 2** is sending a busy signal
 - If **ROBOT 2** is sending a **BUSY** signal, **ROBOT 1** will wait
 - If **ROBOT 2** is **NOT** sending a **BUSY** signal, **ROBOT 1** will start the **CONVEYOR** system.
- **ROBOT 1** will stop the **CONVEYOR** system when an object is detected by the **IR SENSOR**
- **ROBOT 1** will send a **PICK UP** signal to **ROBOT 2**. Continue sending the signal until **ROBOT 2** sends back a **BUSY** signal
- Once the busy signal is received, **ROBOT 1** will loop back to pick another cube



DOBOT MAGICIAN - 2

- **ROBOT 2** waits for signal
 - If NO **PICK UP** signal is received from **ROBOT 1**, **ROBOT 2** waits
 - If a **PICK UP** signal is received from **ROBOT 1**, **ROBOT 2** will start sending a **BUSY** to **ROBOT 1**. **ROBOT 2** will then pick up the cube sitting on the **CONVEYOR** system at the **IR SENSOR**.
- **ROBOT 2** will evaluate the cubes color by hovering it about 3mm above the **COLOR** sensor
 - If the color is **GREEN**, the robot will place the cube in the **GREEN STORAGE**
 - If the color is **BLUE**, the robot will place the cube in the **BLUE STORAGE**
 - If the color is **RED**, the robot will place the cube in the **RED STORAGE**
- **ROBOT 2** will return home and stop sending the **BUSY** signal to **ROBOT 1**.
- **ROBOT 1** will then loop back to waiting for the **PICK UP** signal from **ROBOT 1**.

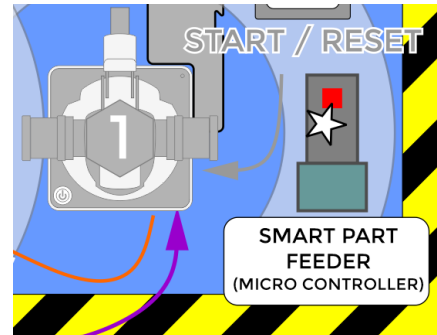


MICROCONTROLLER SMART PART FEEDER

- Design, build, program, and test a **FEEDER** system that can present one cube at a time to **ROBOT 1**.
- Once the cube is removed by **ROBOT 1**, the **FEEDER** should replace the cube at a common pick up position (same position each time) until the **FEEDER** is empty.
 - If the **FEEDER** is empty, it should send an EMPTY signal to **ROBOT 1** and should turn **ON** an indication that the feeder needs to be refilled (sound or light).
 - If the **FEEDER** has a cube present at the common pick up position, it should stop sending the **EMPTY** signal to **ROBOT 1**.



Be sure to consult the Dobot Input/Output manual if you want to use other inputs and outputs, as damage to your robot or your other equipment may result.



Note any additional parameters that are provided by your instructor. Take notes in the space below.

CONCLUSION

Answer the questions below and hand-in electronically to your instructor.

- 1. What are the inputs you used in your workcell? Make a complete chart of inputs below.*
- 2. What are the outputs you used in your workcell? Make a complete chart of outputs below.*
- 3. Describe how your parts feeder works. Explain fully.*
- 4. Attach photos of the parts feeder that you have designed.*



5. *Attach photos of your alternate idea sketches for the parts feeder that you have designed.*
6. *What was the hardest part of this activity for you? Explain how you made it work*
7. *Describe the process that you and your team went through to complete this project by inserting at least six pictures, with captions, that you took throughout the project. Be sure to include programming, designing, wiring, building, and testing.*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

1. Replace the human component of starting the system with a sensor. What sensor did you use and why?

2. Using materials provided by your instructor, design a simulated manufacturing process to add to your workcell. (Examples: drilling, cutting, sanding, spraying, assembly, etc.)



Be sure to consult the Dobot Input/Output manual if you want to use other inputs and outputs, and consult with your instructor before turning the robot on, as damage to your robot or your other equipment may result.

3. *If time permits, remove one of the robots and color sensor from your cell, and add the machine you designed in #1 above, and any necessary sensors, inputs, and outputs. Use the following order of operations:*
 - a. *Make the conveyor deliver the raw material to the robot.*
 - b. *Have the robot deliver the part to the machine.*
 - c. *When finished, have the robot place the part in the storage*
 - d. *Document the process in your engineering notebook, and take pictures along the way.*



4.7 - TP/Block Work Cell Scenario

NAME(s): _____

Date: _____

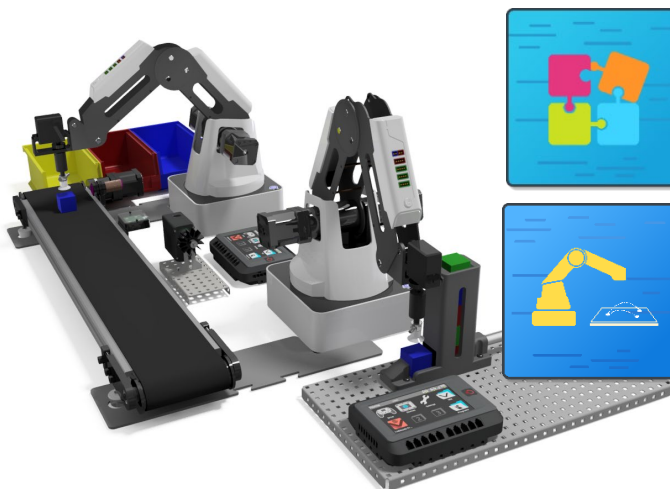
Section: _____

INTRODUCTION

A robotic **workcell** is defined as the complete environment around a robot. This environment may include tools, machines and/or other robots.

In this activity you will use a robot and a **microcontroller** system to recreate a workcell. Your **workcell** will incorporate all the devices that you have learned about in previous activities including:

- Inputs & outputs
- Sensors
- Conveyor belt
- Machines
- Robots



An example of a workcell using robots, prototyped parts and VEX components

KEY VOCABULARY

- | | |
|---------------|--------------------|
| • Handshaking | • Optical Isolator |
| • Workcell | • Sensor |
| • Output | • Input |
| • Nesting | • Palletize |
| • Conveyor | • Microcontroller |

EQUIPMENT & SUPPLIES

- | | |
|--|--|
| • Robot Magician(s) | • DobotLab software |
| • Microcontroller System & Components | • VEXCode V5 control software or other microcontroller software |
| • 1" cylinders or cubes | • Pneumatic Gripper or Suction Cup Gripper |
| • Servo extension cables | • Handshake device |
| • <u>Dobot Input/Output Guide</u> | • Input & Output devices |
| • Breadboard/wire/4N25 Optical Isolator and 100 ohm & 4.7K ohm resistors | • VEX, PIC, Arduino may all be used in this activity, but wiring may vary. |
| • Conveyor System | |

KEY CONCEPTS

Essential questions answered in this activity include:

- How do you integrate robots and other parts of a workcell to complete a given task?
- How do you safely communicate between a microcontroller and a robot?
- What are the different types and styles of inputs and outputs needed to complete your given tasks?
- Which end of arm tooling is most appropriate for your workcell?
- Where is it appropriate to use the jump command within the workcell?
- Where would it be appropriate in your programming to use either variables or functions while programming?
- What components of blockly did you need to complete this task?
- What other software & hardware will I need to complete this task?

PROCEDURE



Caution: NEVER wire anything to the Dobot Magician while it has power on. ALWAYS turn it off before making connections or damage to the robot could occur. Be sure to ask your instructor if you have any questions.

1. You and your team will design, organize, create, program and test a full work cell. Your instructor will have you pick from the list below the items that must be included in your work cell.

☐ Dobot Magician - # of _____	☐ Microcontroller - # of _____
☐ Dobot Conveyor System	☐ Student Designed Feeder System
☐ Dobot IR Sensor	☐ Student Designed Machine - # of _____
☐ Dobot Color Sensor	☐ Student Designed Storage System
☐ Handshakes - # of _____	☐ _____
☐ _____	☐ _____

Machine Examples	
• Pneumatic Stamping Press	• Painting Station
• Rotary or Reciprocating Saw	• Drying Station
• Welder	• Drilling Station
• Quality Control Station	• ???



2. Be sure to note any additional parameters that are given to you by your instructor (due date, size or storage requirements, and additional items from home. Take notes in the space below.

3. In the time allotted for this project design a workcell that includes the following:
 - a. Accurate Robot **PICK AND PLACE** routines.
 - b. **JUMPS** where appropriate.
 - c. Proper usage of **FUNCTIONS** and **VARIABLES**.
 - d. A simulation of a manufacturing process.
 - e. A **PALLETIZE** or **STACKING** routine.
4. Create a video of your workcell.



Be sure to consult the [Dobot Input/Output Guide](#) if you want to use other inputs and outputs, as damage to your robot or your other equipment may result.

Notes:



CONCLUSION

1. *Make a flowchart/Process flowchart of your workcell as indicated by your instructor in the space below.*
2. *What's the pseudocode that you used for your microcontroller program? Copy and paste it here.*
3. *What are the inputs you used in your work cell?*
4. *What are the outputs you used in your work cell?*

GOING BEYOND

Finished early? Try some of the actions below. When finished, show your instructor and have them initial on the line.

- | | |
|-------|---|
| _____ | 1. Make your work cell communicate with someone else's work cell in your class. When your process ends, theirs begins. |
| _____ | 2. Same as number one, but use the same part in both cells. In other words, perform 2 operations in two separate cells, on the same part. |





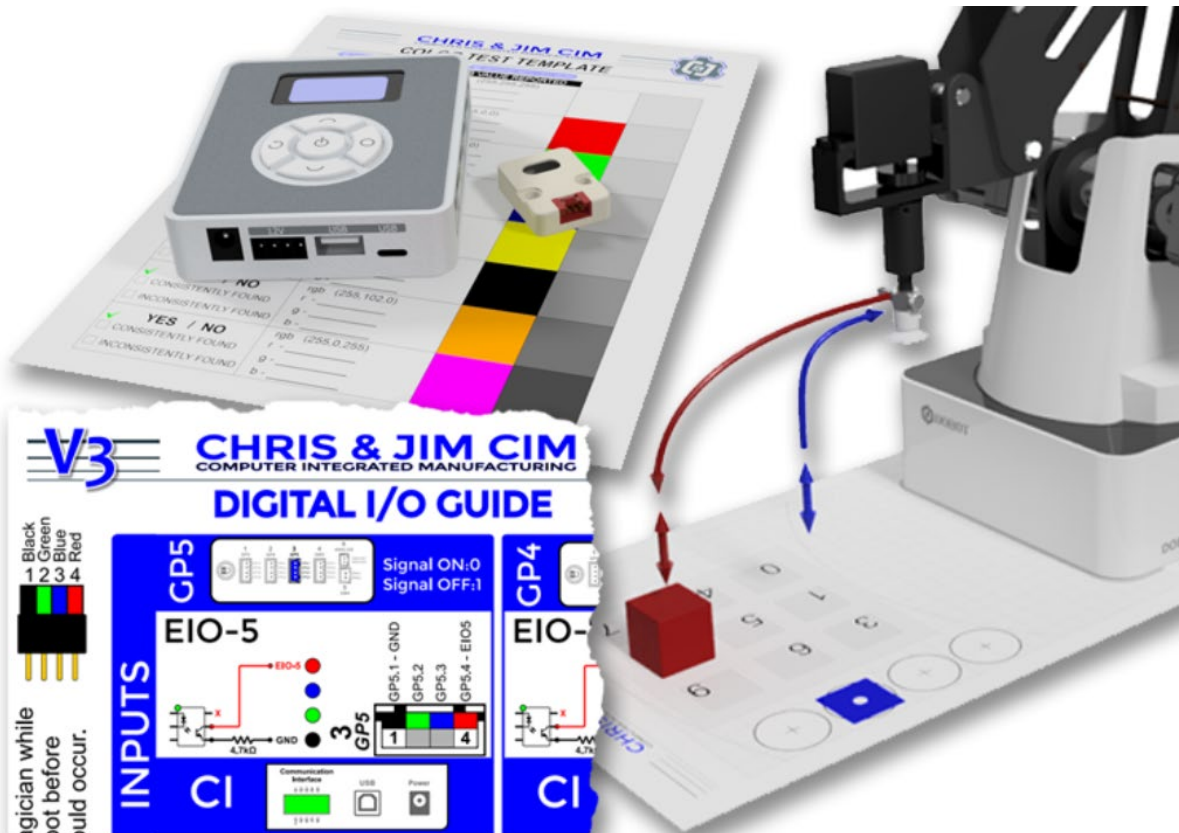
- CHAPTER 5 - ROBOTICS RESOURCES

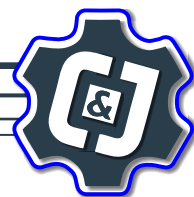
CHAPTER INTRODUCTION:

This chapter contains additional items that may need to be printed such as field templates and wiring diagrams. It also contains a glossary for industrial robotics and one for blockly programming, as well as a list of academic standards that can be found throughout the curriculum.

CHAPTER RESOURCES:

- Field Templates
- Color Test Template
- Digital Input and Output Guide
- Peripheral Wiring Guide
- Magic Box IO Guide
- Work Cell Wiring Guide
- Robotics Glossary
- Blockly Glossary
- Academic Standards
 - Standards for Technology and Literacy
 - Next Generation Science Standards
 - ISTE Standards





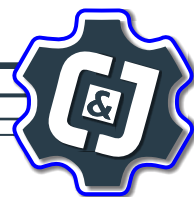
FIELD TEMPLATE

The diagram is set on a fine grid background. A central gray rounded rectangle contains a 3x3 keypad with the following numbers:

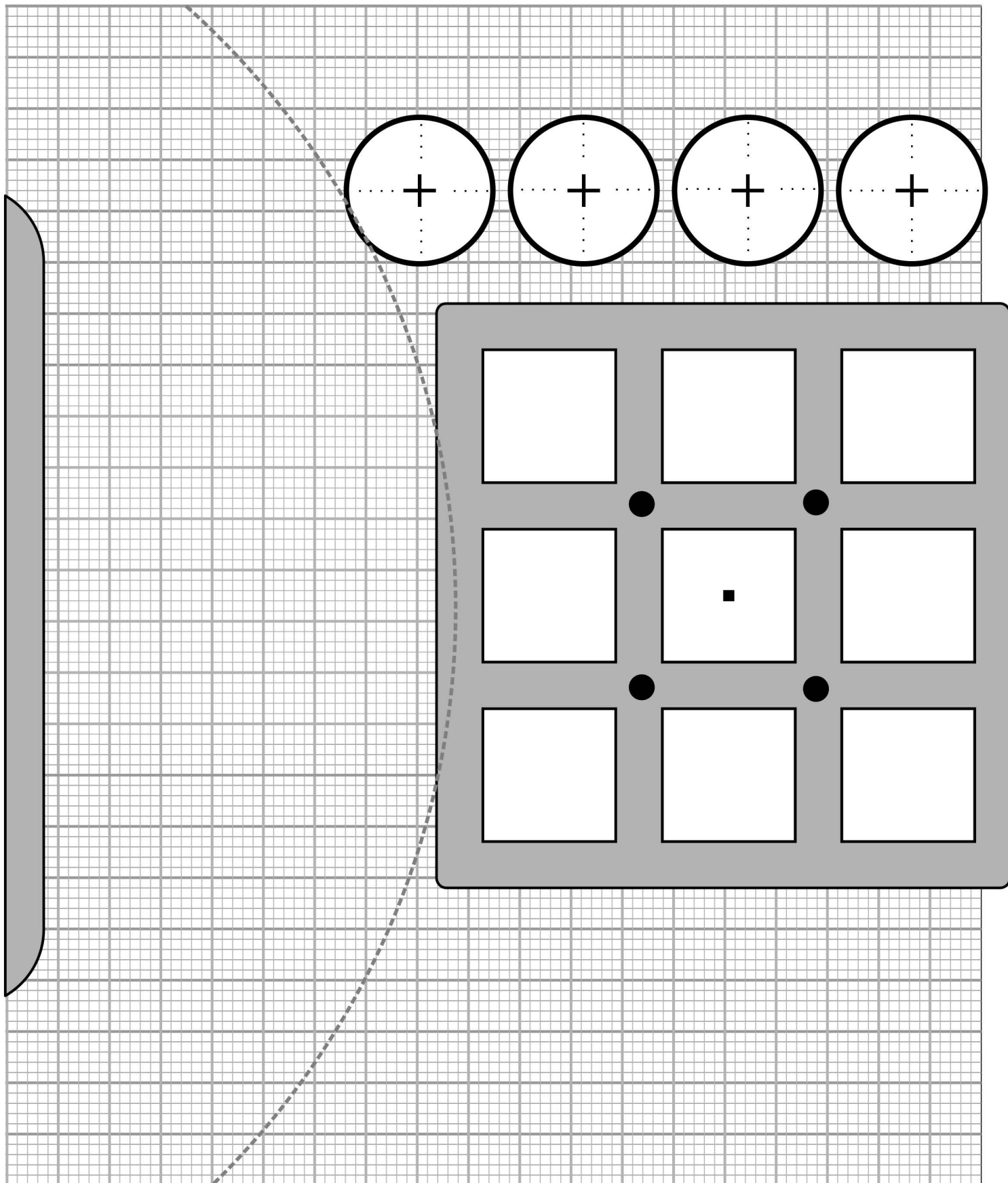
6	7	8
3	4	5
0	1	2

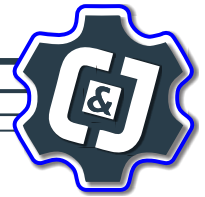
To the left of the keypad is a vertical column of four circles. The top and bottom circles contain a crosshair. The middle circle contains a blue square with a white dot in the center.

At the bottom of the grid is a large gray rounded rectangle.



FIELD TEMPLATE





COLOR TEST TEMPLATE

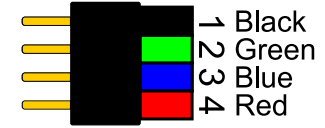
Color sensor port1 detected color Black ?

Color sensor port port1 R color value

COLOR FOUND?		RGB VALUE REPORTED	
☒ YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,255,255) r - _____ g - _____ b - _____		
☒ YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,0,0) r - _____ g - _____ b - _____		
☒ YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (0,255,0) r - _____ g - _____ b - _____		
☒ YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (0,0,255) r - _____ g - _____ b - _____		
☒ YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,255,0) r - _____ g - _____ b - _____		
☒ YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,102,0) r - _____ g - _____ b - _____		
☒ YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (255,0,255) r - _____ g - _____ b - _____		
☒ YES / NO ☐ CONSISTENTLY FOUND ☐ INCONSISTENTLY FOUND	rgb (0,0,0) r - _____ g - _____ b - _____		



Caution: NEVER wire anything to the Dobot Magician while it has power on. **ALWAYS** shutdown the Dobot before making connections or damage to the robot could occur.



V3

CHRIS & JIM CIM
COMPUTER INTEGRATED MANUFACTURING

DIGITAL I/O GUIDE



OUTPUTS

GP1

Peripheral Interface
GP1 Stopper | SW1
GP2 Stopper | SW2

EIO-10

1000Ω

GP1

GP1.1 - GND
GP1.2 - EIO10
GP1.3
GP1.4

EIO-11

1000Ω

GP1

GP1.1 - GND
GP1.2
GP1.3 - EIO11
GP1.4

GP2

Peripheral Interface
GP1 Stopper | SW1
GP2 Stopper | SW2

EIO-13

1000Ω

GP2

GP2.1 - GND
GP2.2 - EIO13
GP2.3
GP2.4

EIO-14

1000Ω

GP2

GP2.1 - GND
GP2.2
GP2.3 - EIO14
GP2.4

EIO-15

1000Ω

GP2

GP2.1 - GND
GP2.2
GP2.3
GP2.4 - EIO15

INPUTS

GP5

Signal ON:0
Signal OFF:1

EIO-5

4.7kΩ

GP5

GP5.1 - GND
GP5.2
GP5.3
GP5.4 - EIO5

C1

Communication Interface
USB
Power

EIO-19

4.7kΩ

C110

C19 - GND
C18
C17 - EIO19
C16

GP4

Signal ON:0
Signal OFF:1

EIO-7

4.7kΩ

GP4

GP4.1 - GND
GP4.2
GP4.3
GP4.4 - EIO7

C1

Communication Interface
USB
Power

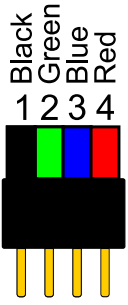
EIO-20

4.7kΩ

C18

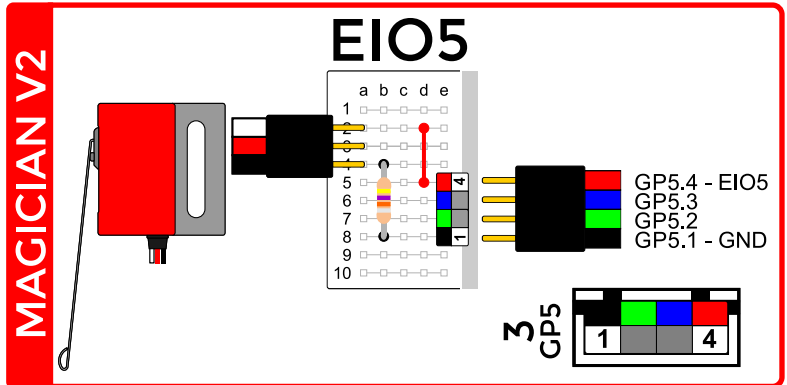
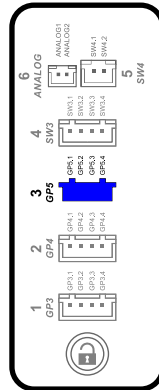
C110 - GND
C19
C18 - EIO20
C17
C16

DIGITAL I/O TESTING



EXAMPLE
How to
Test Inputs

**3/GP5
- EIO5**

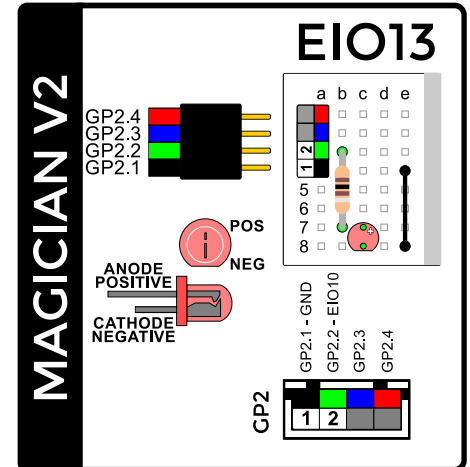
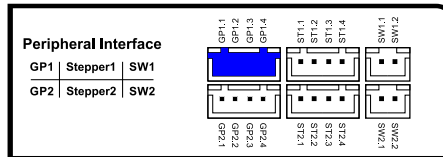


Caution: NEVER wire anything to the Dobot Magician while it has power on. **ALWAYS** shutdown the Dobot before making connections or damage to the robot could occur.

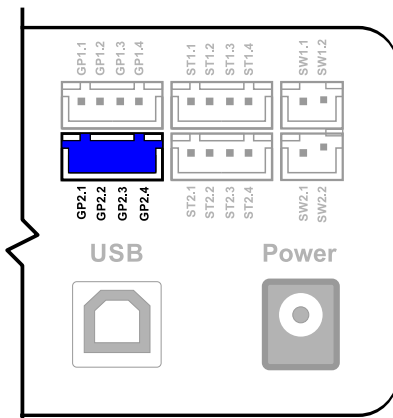


EXAMPLE
How to
Test Outputs

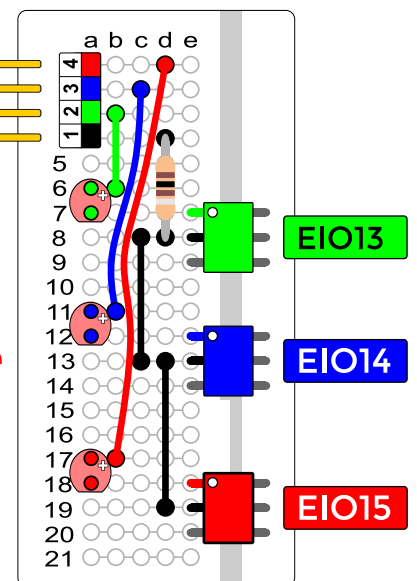
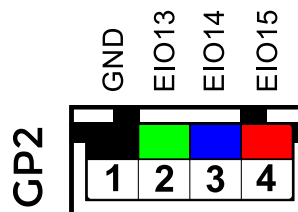
GP2 - EIO13



EXAMPLE
How to Use a
Common GND



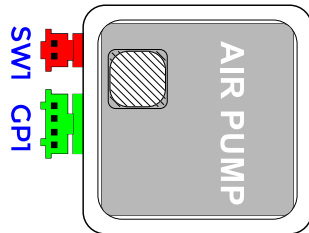
**Multiple
Outputs
from One Cable**



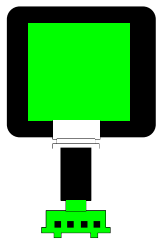
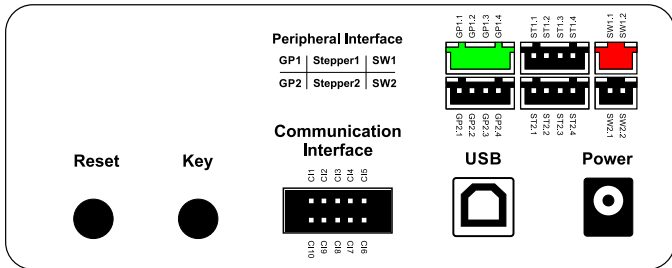
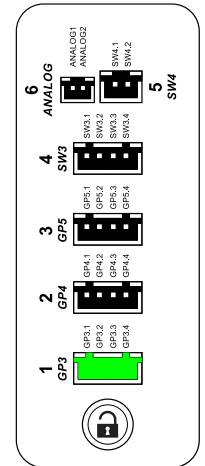
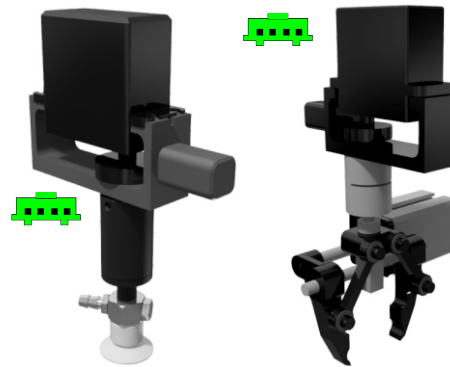
GP2 - EIO13, 14, 15

PERIPHERAL GUIDE

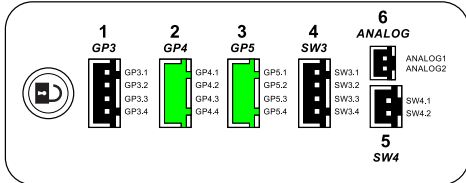
Air Pump SW1 GP1



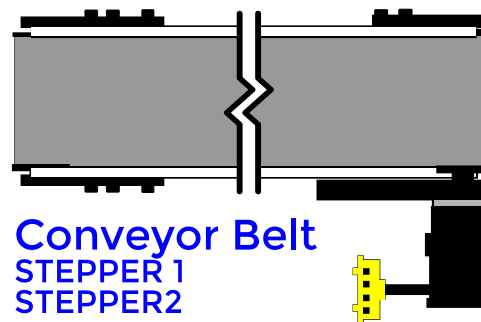
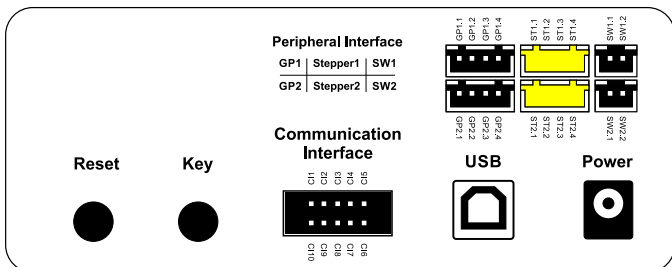
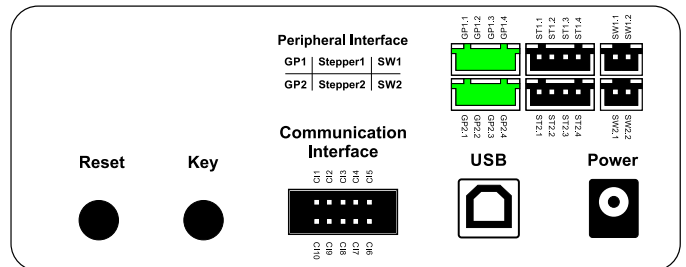
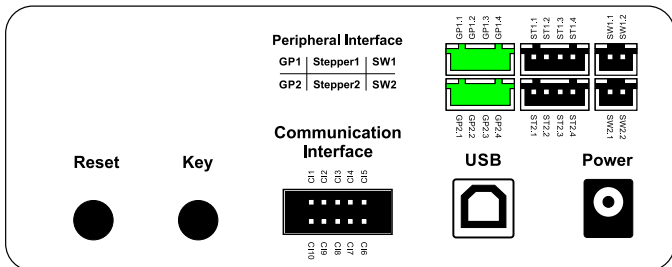
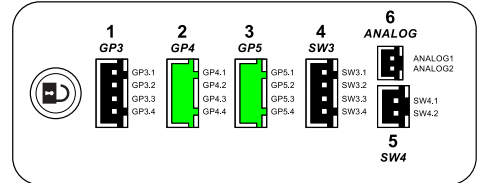
Wrist Rotate Micro Servo GP3



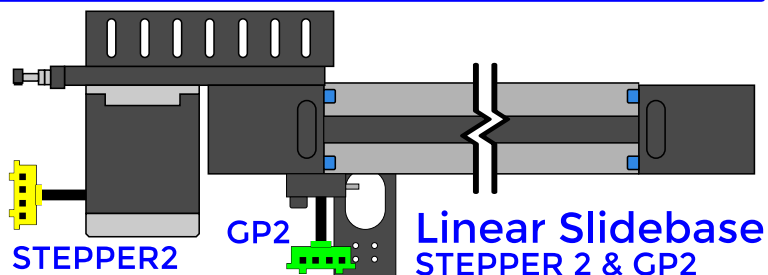
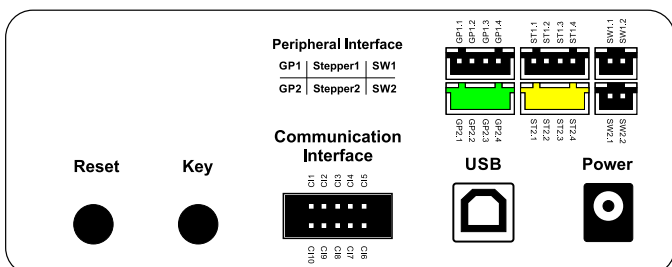
Color Sensor GP1, GP2, GP4, GP5



Photoelectric IR Sensor GP1, GP2, GP4, GP5



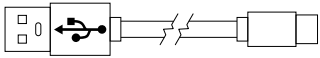
Conveyor Belt STEPPER 1 STEPPER2



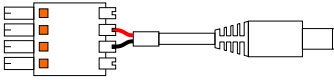
Linear Slidebase STEPPER 2 & GP2

SENSOR GUIDE

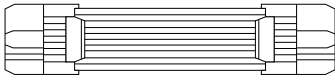
USB-A TO USB-C CABLE



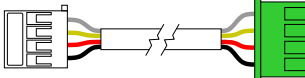
4 PIN POWER CABLE - 12V



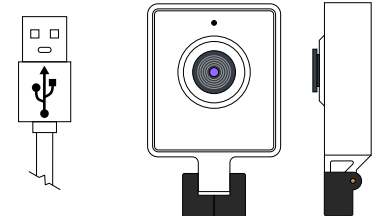
10 PIN COMMUNICATION CABLE

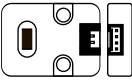
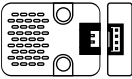
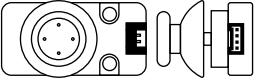
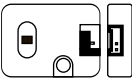
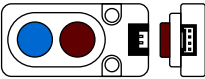
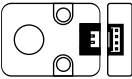
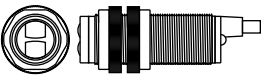
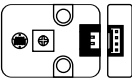
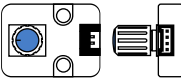
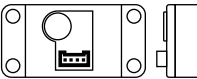
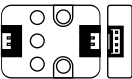
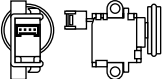


4 PIN COMMON SENSOR WIRE



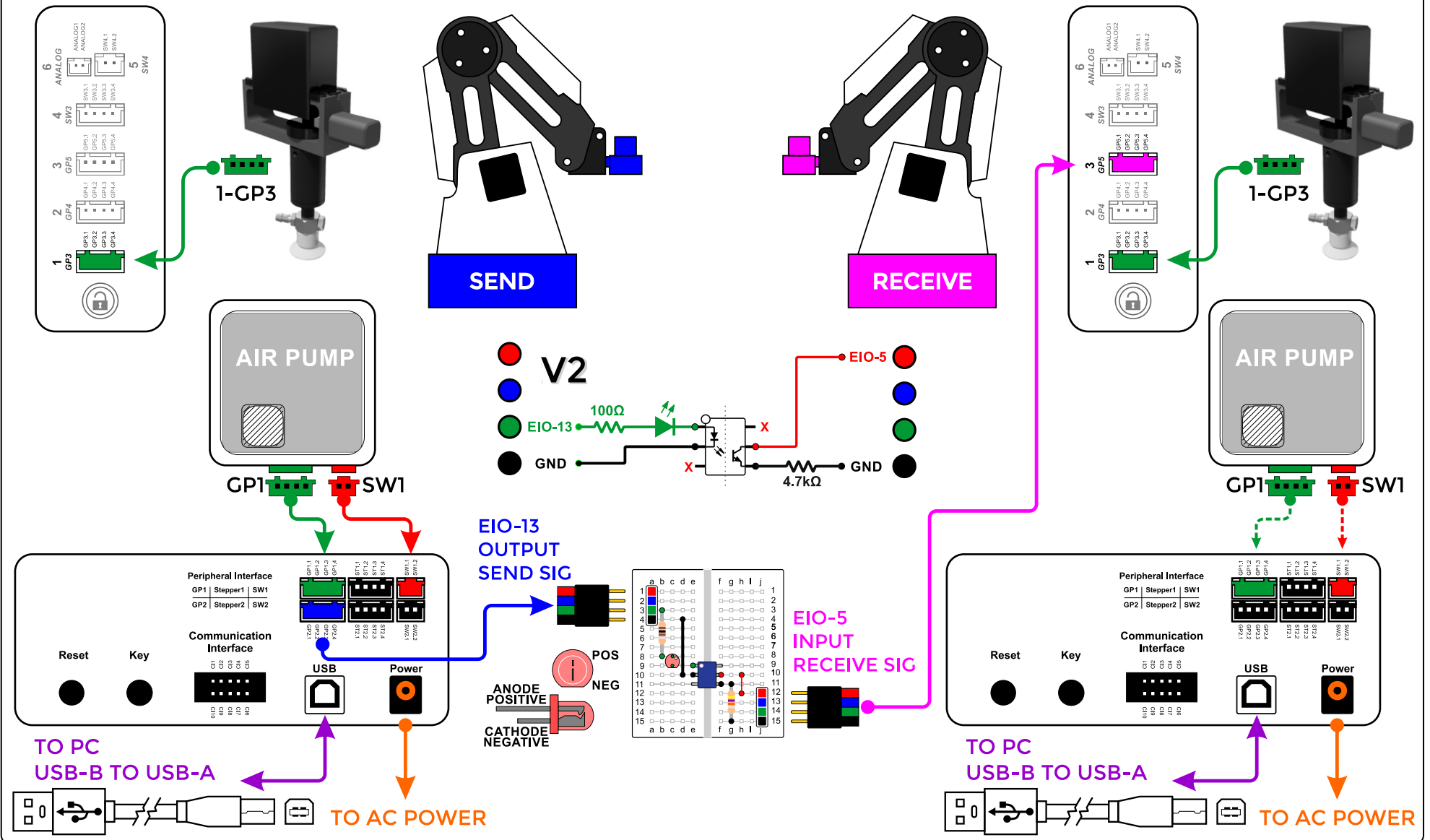
CAMERA



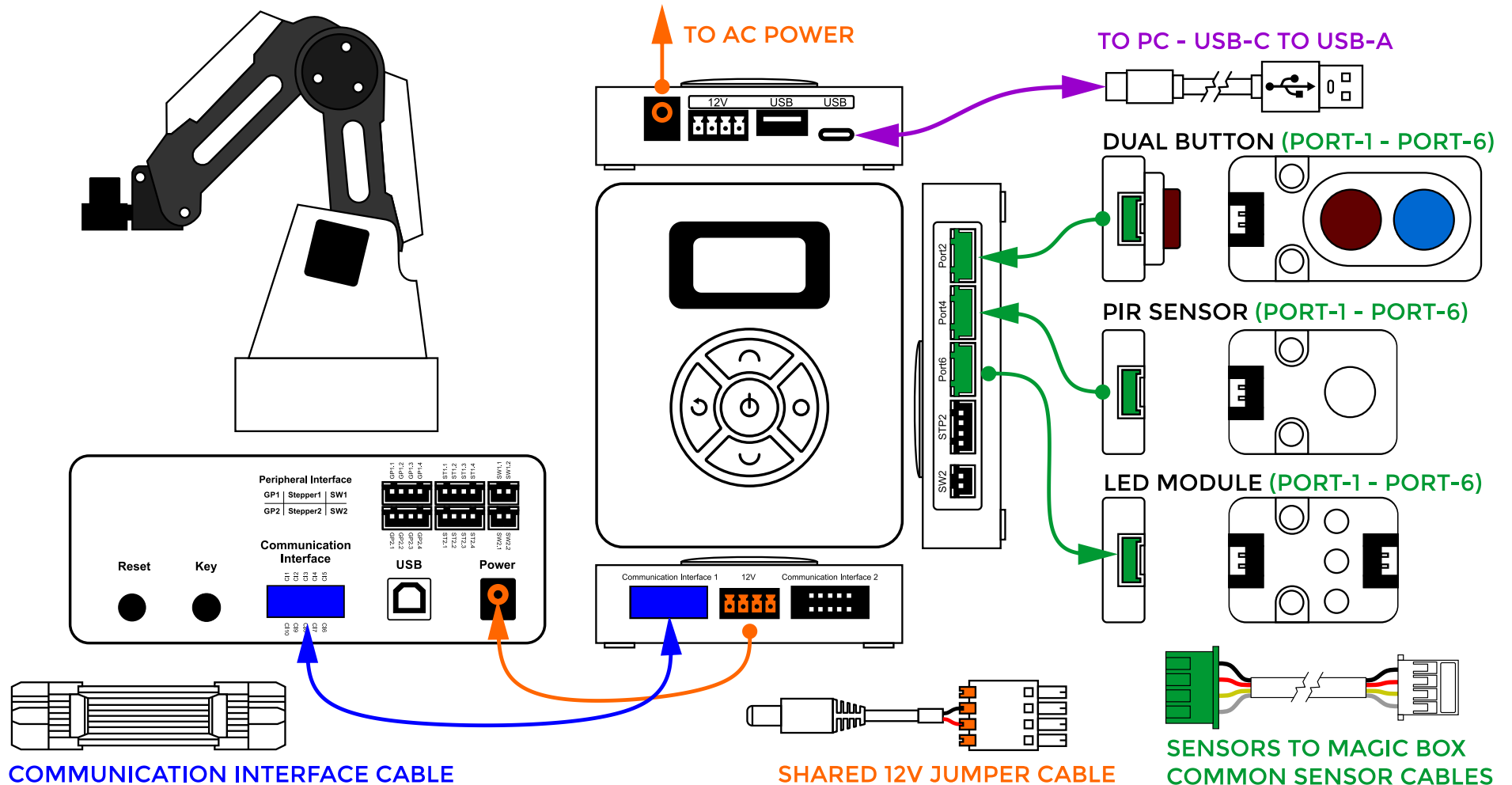
MODULE	COMM MODE	PORTS	SIGNAL	NOTES
GESTURE SENSOR 	I2C	1 - 6	DIGITAL	EFFECTIVE DISTANCE 2in - 6in, UP DOWN RIGHT LEFT FORWARD, BACKWARD, CW, CCW
HUMITURE SENSOR 	I2C	1 - 6	ANALOG	TEMP RANGE 0 - 140 F HUMIDITY RANGE, 10-90 RH
JOYSTICK 	I2C	1 - 6	ANALOG DIGITAL	X/Y VALUE 0-250 Z 0:RELEASED 1:PRESSED
COLOR SENSOR 	I2C	1 - 6	ANALOG	LARGER VALUE = DARKER COLOR, 0:NO COLOR, 1:RED, 2:GREEN, 3:BLUE, 4:YELLOW, 5:BLACK, 6:WHITE
DUAL BUTTON 	IO	1 - 6	DIGITAL	0:RELEASED 1:PRESSED
PIR SENSOR 	IO	1 - 6	DIGITAL	EFFECTIVE DISTANCE 59in,
PHOTOELECTRIC SENSOR 	IO	1 - 6	DIGITAL	VALUE RANGE 0:NONE 1:DETECTED RESPONSE TIME <2ms
LIGHT SENSOR 	ADC	3 & 4	ANALOG	BRIGHTNESS RETURN VALUE 140-600 LARGER VALUE = MORE LIGHT
POTENTIOMETER KNOB 	ADC	3 & 4	ANALOG	RETURN VALUE 0-407
SOUND SENSOR 	ADC	3 & 4	ANALOG	RANGE 0-1023 SENSITIVITY 48db-52db
LED MODULE 	SINGLE BUS	1 - 6	DIGITAL OUT	RGB RANGE 0-255, BRIGHTNESS RANGE 0-100%
MICRO SERVO 	PWM	3 - 6	PWM	DEGREE RANGE 0-180

WORK CELL EXAMPLE GUIDE 1

MAGICIAN TO MAGICIAN WIRING



WORK CELL EXAMPLE GUIDE 2



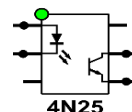
Vocabulary

The vocabulary below is used throughout the curriculum and contains general definitions for robotics in industry.

Key Term	Definition
Active sensors	Sensors that require an external power supply to operate.
Analog	A type of signal that has a range of values, not just 0 and 1. A good example is an 8-bit light sensor with at least 256 different values.
Audio Feedback	Provides the robot with the ability to use sounds or speech as an output.
Axis Movement	Movement of a robot's wrist in a straight line on the X, Y or Z axis.
Calibration	Calibration is a tool used when creating visual models that allows the user to define a background other than solid white when using a camera with the robot (attempts to ignore background/constant visual information and focus on new elements).
Current limiting resistor	A resistor used in a circuit to limit the current going to a component such as a light emitting diode. (LED)
Conveyor System	A system used in industrial settings to move items around efficiently. Commonly seen in the form of a belt but may also be sets of motorized rubber wheels that can change direction. Can be controlled by stepper motors for positional accuracy.
Degrees of Freedom	The range and flexibility of motion. Each added degree of freedom adds greater range of motion to the robot's tooling within its work area. <i>Also see Roll, Pitch and Yaw.</i>
Digital	A signal that can only be read in one of two states: YES or NO, ON or OFF, TRUE or FALSE, HIGH or LOW. There are no values in-between.
Drive system	The power or motors that are used to control the positioning of the robotic arm. Usually electric but may be hydraulic or pneumatic as well.
Electric drive system	A drive system used for high accuracy, repeatability, and speed. Used to control motors, servos, or stepper motors.
End of arm tooling (EoAT)	The tooling added to the end of a robot's arm that allows it to perform specific operations. Also known as an <i>end effector</i> .
End Effector	The tooling added to the end of a robot's arm that allows it to perform specific operations. Also known as an <i>End of Arm Tooling</i> .
Function	A named section of a program that performs a task. It can also be considered a procedure or a routine and greatly simplifies otherwise complicated programs.

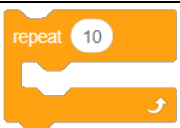
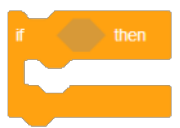
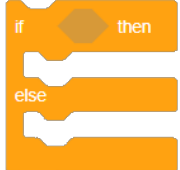
Vocabulary	Definition
Gesture Sensor	A 3D sensor that recognizes hand gestures that can be used for human-machine interaction.
Handshaking	Handshaking is defined as safe communication between two or more, like or unlike, pieces of technology.
Home	A position within the robot's work envelope designated as a starting point for the operation.
Humiture Sensor	A sensor used to detect relative humidity and temperature of the environment in which it is placed.
Hydraulic drive systems	A drive system used for larger robots that require large amounts of power. Linear movements are produced by hydraulic pistons while rotary is produced by rotary vanes.
Input	Any data or signals that a robot receives that provides feedback about its environment. Information that is brought "into" the controller. Examples: sensor values, signals from another device or robot, or user commands.
Joint movement	The movement of a robotic arms end effector defined by the movement of its individual joints.
Light Sensor	An analog sensor that detects light using resistance. As light intensity increases, resistance decreases. This results in a change in voltage that can be used to determine light intensity in the environment.
Lights Out Manufacturing	The practice of fully automating a production system in industry that requires no human intervention. Refers to the ability for the system to run overnight, without humans, with the "lights out".
Linear movement	The same as an axis movement; Movement of a robot's end effector in a straight line on the X, Y or Z axis.
Loop	The ability to repeat a given set of instructions or code a specific number of times
Main controller	The brain of the robotic system. Used to control the motion of and programming of the robot as well as control various inputs and outputs and communicate with other elements in a work cell.
Machine Tending	Using a robot to load and/or unload a machine with parts or raw material. An example would be using a robot to put raw materials into a machine, turning it on, and removing finished product for palletization. This is done using a closed loop system.
Magic box	This is Dobot's robotic input/output interface. It allows you to plug in sensors, and outputs that can be used with the Dobot Magician and Macian Lite.
Matrix	A rectangular array of parts, made of a finite number of rows and columns. a good example is an array of boxes on a pallet.
Microservo	A small electric actuator with accurate positioning through at least 180 degrees.
Microwait	A very short delay written into a program so that some event isn't missed. For example: when a switch moves from high to low quickly.
Microcontroller	A microcontroller is a small computer, usually on a single chip. They are used to control many things in industry. Examples of microcontrollers are Arduino, VEX, and PLC's

Vocabulary	Definition
Nesting	Making parts fit most efficiently on a pallet or in a given space.
Normally closed (NC)	Digital sensor, like a switch where a signal is received until the sensor is triggered. Switch not pressed = ON (1), Pressed = OFF (0). Useful in handshaking when you want something to happen when an input signal turns off.
Normally open (NO)	Digital sensor, like a switch where a signal is received until the sensor is triggered. Switch not pressed = OFF (0), Pressed = ON (1). Useful in handshaking when you want something to happen when an input signal turns on.
Optical isolator	An electronic device that separates two unlike voltages or currents in a robotic communication system. It accomplishes this using an infrared LED and a light sensor embedded in a device.
Output	An outgoing device or signal that is controlled by the robot. An output is often something that the robot can turn on or off. An output can also be a value that is produced by the robot's programming. Some examples are motors, lights, and displays.
Palletizing	An automated process that uses a robot to load and unload parts, containers, or boxes onto pallets for storage and shipping. Removing boxes or parts from a pallet is known as de-palletization.
Passive sensor	Sensors that detect inputs from the physical environment without external power.
Payload	The size and weight of the material that a robotic arm can safely lift.
Photoelectric Sensor	A digital sensor that can detect the presence or absence of an object. Also known as a <i>proximity switch</i> .
Pick and place	When a robot retrieves parts from one location and consistently places them in a new location.
Picture Recognition	The process of identifying a feature or an object in a picture/image. Also known as image recognition.
PIR Sensor	The PIR sensor is a body infrared sensor. When the sensor detects infrared, it outputs a 1 (ON) signal until the triggering signal disappears and outputs a 0 (OFF).
Pitch	The degree of freedom of a robotic arms end effector around its Y axis.
Pneumatic drive system	A drive system used in smaller robots. Used to control rotary actuators or sliding joints. Typically used for high-speed operations and limited movements.
Programable Logic Controller	It is a simple electronic device that allows very precise control of machinery and allows multiple devices to communicate. Also known as a <i>PLC</i> .
Pull-down resistor	A resistor used in an electrical circuit to force a signal to be low when not activated (0v). This prevents a signal from floating between a 1(5v) and 0(0v) in a digital circuit.
Pull-up resistor	A resistor used in an electrical circuit to force a signal to be High (5v) when not activated. This prevents a signal from floating in between a 1(5v) and 0(0v) in a digital circuit.
Ramping	To accelerate or decelerate the speed with which a robot moves over a short period of time.

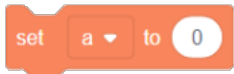
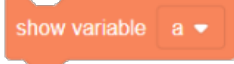


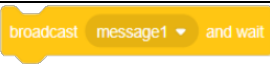
Vocabulary	Definition
Record	To save a position with a robot arm by moving the robot to that position and then storing the values of all the robot's joints.
Relative coordinates	A cartesian coordinate system where each successive point is relative to the point before it. Also known as <i>incremental</i> .
Relative positions	A system of recording positions where each point is relative in X, Y, and Z coordinates to all the other points in a program.
Robot accuracy	How close a robot's actual movement is to a given position.
Robot repeatability	A robot's ability to return to a given position multiple times.
Robotic arm	The parts of a robot that are used to locate and position the end of arm tooling. Usually includes multiple joints, and axis of motion.
Roll	The degree of freedom of a robotic arms end effector around its Z axis
Roll angle	Adjusting this angle allows a robot's end of arm tooling to line up and nest parts on a pallet.
Sensor	Inputs that provide feedback and bring in a variety of data into the robot. Commonly used for automation of tasks.
Stack Light	A set of lights used in industry to indicate the status of a machine or the operation it is performing. Provides visual, and sometimes audible indicators for operators and workers.
Suction	The gripper or end of arm tooling that uses vacuum and a flexible suction cup to pick up parts easily.
Teach	To save a position with a robot arm by typing in an X, Y and Z value and storing the position.
Teach pendant	A handheld device used to manually control, program, and troubleshoot a robotic arm without the need for a full control terminal.
Tool center point	The X, Y and Z value recorded or taught in robotics that represents where the work is being done in space by the robot's end of arm tooling. (TCP)
Touch up	To easily correct or round saved points in a robotics program by manually adjusting X, Y, and Z values in the control software.
Training Model	Training model is a process of testing camera/photo data that has been stored in user-defined classifications.
Vacuum gripper	An end effector that creates negative pressure to allow a robot to pick up an object easily with a suction cup type device.
Variable	A changeable quantity in a program that can be represented by a word or a letter. Variables can be assigned, changed, or referenced throughout a program and makes programming more efficient.
Workcell	The complete environment around a robot. May include tools, machines and/or other robots. Also known as a <i>manufacturing cell</i> .
Work envelope	Defined as the range and area each robot may work within.
Yaw	The degree of freedom of a robotic arms end effector around its Y axis

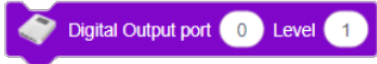
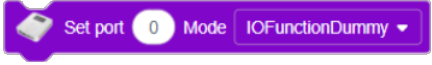
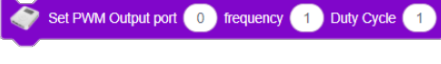
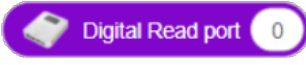
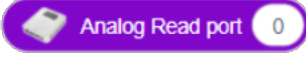
The vocabulary below is used throughout the Blockly activities, lessons, and presentations for DobotLab software. Please note that not all commands are defined here, just the ones most important to completing the activities. See the Help Section in DobotLab for other definitions.

Control Menu		
Function	Block	Description
Wait		Wait for a specific amount of time. If it is less than one second, it can be considered a micro-wait.
Repeat # of times		Repeat a block of code a specific number of times.
Repeat Forever		Repeat a block of code forever. Also known as a forever loop.
If-Then		Execute a block of code only if the specified condition is true.
If-Then-Else		Executes the first block of code as long as the condition is true, otherwise execute the second block of code.
Wait Until		Execute a block of code if the specified condition is true.
Repeat Until		Execute s a block of code repeatedly until the specified condition is true.
Stop		Stops execution of a block of code.

Operators Menu		
Function	Block	Description
Addition		Execute an addition operation.
Subtraction		Execute a subtraction operation.
Multiplication		Execute a multiplication operation.
Division		Execute a division operation.
Get Random Number		Generate a random number within a specified range.
A Greater Than B		If the value of the first parameter A exceeds the specified value B then the result is True.
A Less Than B		If the value of the first parameter A is less than the specified value B then the result is True.
A Equals B		If the value of the first parameter A is equal to the specified value B then the result is True.
A And B		If A and B are both True at the same time, then the result is True.
A Or B		If either A or B are True then the result is True.
Not A		If the condition is False then return a True.
String Contains		If the condition of a string value (input) contains any of the words or characters return a True value
Camera Classification Development		Creates and saves visual data (models) in defined classifications.

Variables Menu		
Function	Block	Description
Make a Variable		Make a variable to get, store, or display data.
Set Variable		Set a variable to a specified number.
Increase Variable		Increase the value of a variable.
Show Variable		Display the value of a variable.

Events Menu		
Function	Block	Description
When Green Flag Clicked		When the green flag is clicked, operate the blocks below. This is used to start a program.
When Key is Pressed		When the specified key is clicked, operate the blocks below.
When Message Received		When a specified message is received, operate the blocks below. Used for communications between different parts of a program or even different devices.
Broadcast Message		Send a message to another part of the program, or another attached device. Used for communications between different parts of a program or even different devices.
Broadcast Message & Wait		Send a message to another part of the program, or another attached device and wait until the program that it executes is finished.

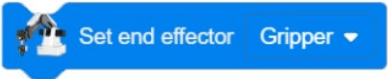
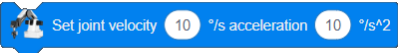
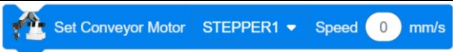
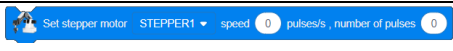
Magic Box Menu		
Function	Block	Description
Output Digital Signal		Send a digital signal, a 1 or a 0, on the specified port.
Set Port Status		Allows you to set the port type. Examples are: PWM, ADC, and others.
Set PWM Output		Allows you to change the output of the PWM port by changing the frequency and the duty cycle.
Get Digital Signal		Read the digital signal on the given port.
Get Analog Signal		Read the analog signal on the given port.
Set Stepper Speed		Set the speed, in pulses per second, of the given stepper motor.
Set Conveyor Speed		Set Speed of the conveyor system in millimeters per second.

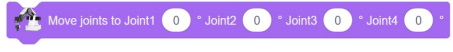
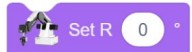
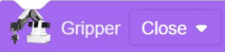
Wireless Menu

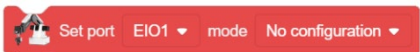
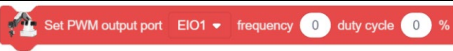
Function	Block	Description
Set Up Bluetooth		Allows you to set up a six-digit Bluetooth group.
Receive Message		Receive information from a device in a Bluetooth group.
Send Message		Send information to a device in a Bluetooth group.

AI Sensor Menu - Added Extension

Function	Block	Description
Set LED		Sets the RGB Value of the LED.
Set LED Brightness		Sets the RGB Value of the LED and controls the brightness.
Set LED Status		Sets the LED to an ON or OFF state. Allows for choice of port, and LED number 1, 2, or 3.
Get Potentiometer Value		Returns the analog value of the knob potentiometer.
Get Light Value		Gets the analog value of the light sensor.
Get Sound Value		Returns the analog value of the sound sensor.
Get Temp & Humidity		Returns the analog value of the temperature (C) or the relative humidity. (%)
Get Color Value		Returns the color value of the color sensor.
Detect What Color		Determines what color the color sensor has detected.
Detect Specific Color		Look for a specified color with the color sensor.
Photoelectric Sensor Status		Determines if there is an object (1) or not (0) within its range.
Gesture Sensor Status		Looks for a specific Gesture: left, right, forward, backward, clockwise, counterclockwise, up, or down.
Dual Button Status		Look for a specific button to be pressed, either red or blue. Digital sensor.
Set Servo Angle		Sets the servo at a specific angle between 0 and 180. Analog Sensor.

Setting Menu		
Function	Block	Description
Set End Effector		Tells the program which end effector is being used: Gripper, Suction cup or Pen.
Set Motion Ratio		Sets both acceleration and velocity of the robot's moves. Uses %
Set Joint Speed		Sets the velocity of the robot's joint motion in degrees per second ² .
Set Jump Height		Sets the height of all jump moves in millimeters.
Set Conveyor		Set's the speed of the conveyor's stepper motor in pulses per second.
Set Stepper Speed		Sets the stepper motor's speed in pulses per second.
Set Stepper Speed & Pulse		Sets the stepper motor's speed in pulses per second, and number of pulses.

Motion		
Function	Block	Description
Jump To Position		Jump to a given position. XYZ coordinate with rotation in degrees.
Go to Position		Go to an XYZ position, and move either in a straight line, or a joint move.
Move Relative		Makes the robot arm move relative to the position of where it is on the XYZ axis.
Move Joints		Makes the robot arm move each joint a certain number of degrees.
Set Rotate		Set wrist rotation angle in degrees.
Set Gripper		Turn the gripper on or off.
Set Suction Cup		Turn the suction cup on or off.

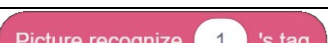
I/O		
Function	Block	Description
Set Port		Set Digital I/O ports 1-20 to High (1) or Low (0).
Set PWM (Servo)		Sets the frequency and the duty cycle for a pulse width (PWM) modulation port. Used to set an output for a servo.
Set Output		Sets an I/O port's output to high (1) or Low (0)
Get Digital Signal		Get an incoming digital signal on a given I/O port 1-20.
Get Analog Signal		Get an incoming analog signal on a given I/O port 1-20.

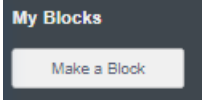
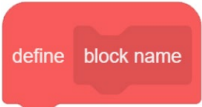
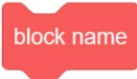
Photoelectric Color Sensor – Added Extension		
Function	Block	Description
Set IR Sensor	 Set infrared sensor status ON version V1 port GP1	Sets the state of the infrared sensor to ON (1) or Off (0).
Get IR Reading	 Get GP1 infrared sensor reading	Get the digital signal from the infrared sensor. ON (1) or Off (0).
Set Color Sensor	 Set color sensor status ON version V1 port GP1	Sets the status of the color sensor to On or Off.
Get Color Sensor Reading	 Get Red color sensor reading	Get a reading from the color sensor when it is Red, Blue, or Green.

Sliding Rail – Added Extension		
Function	Block	Description
Sliding Rail Status	 Set sliding rail status on version V1	Sets the status of the sliding rail to off or on.
Set Sliding Rail Speed	 Set sliding rail velocity 0 % in point-to-point mode	Allows sliding rail speed to be set using a percentage of full speed.
Move Linear Rail	 MoveLinearRailTo 0 mm	Allows linear rail to move to a given position. Measured in millimeters from home.

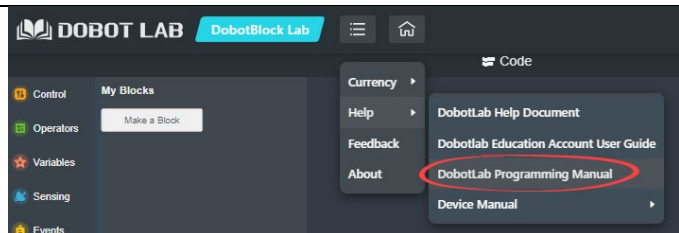
AI Menu: Text to Speech – Added Extension		
Function	Block	Description
Speak	 speak hello	Allows user input of text the robot will convert to speech.
Set Voice	 set voice to Wejack, English male voice	Allows user input of language and voice characteristics in strings that the robot speaks.

AI Menu: Camera – Added Extension		
Function	Block	Description
Use Camera	 use camera Integrated Camera (13d3:56b2)	Allows user input as to which camera the robot uses.

AI Menu: Image Recognition – Added Extension		
Function	Block	Description
Picture Input	 Picture	Allows the camera to take and store a temporary photo that can be compared to stored models.
Picture Recognition	 Picture recognize 1 's tag	Allows the user to compare data to stored models.

My Blocks Menu		
Function	Block	Description
Make a Block		Allows you to make your own block of code by making one block out of many.
Define block		The code attached to this determines what the block does. It is user defined.
Block Name		Gets the My Block and runs its code within the main program. Can be used multiple times.

*Please remember that all programs are dynamic and change continuously. Therefore, some blocks may go away, and more may be added. To get the latest block definitions go to the **Help Menu** in **DobotLab** and look for the **DobotLab Programming Manual** and check out the **DobotBlock** section.*



Curriculum Standards

The standards defined below are far reaching and very large in scope. They include the following standards:

- Standards for Technological Literacy
<https://www.iteea.org/File.aspx?id=67767>
- Next Generation Science Standards
<http://www.nextgenscience.org/> pg 102
- ISTE (International Society for Technology in Education)
https://cdn.iste.org/www-root/PDF/ISTE%20Standards-One-Sheet_Combined_09-2021_vF3.pdf

All the activities in this curriculum, once completed, can cover all the standards outlined below, and most of them multiple times. Having students working with Google docs and sharing, or another local high school, will cover even more and give students a more global approach to learning.

Standards for Technological Literacy	
The Standards for Technological Literacy (STL) were developed by the International Technology and Engineering Educators Association (ITEEA) and are available as a complete download for free here: https://www.iteea.org/File.aspx?id=67767	
2-W	<i>Standard:</i> Students will develop an understanding of the core concepts of technology. <i>Benchmark:</i> Systems thinking applies logic and creativity with appropriate compromises in complex real-life problems.
2-Z	<i>Standard:</i> Students will develop an understanding of the core concepts of technology. <i>Benchmark:</i> Selecting resources involves trade-offs between competing values, such as availability, cost, desirability, and waste.
2-AA	<i>Standard:</i> Students will develop an understanding of the core concepts of technology. <i>Benchmark:</i> Requirements involve the identification of the criteria and constraints of a product or system and the determination of how they affect the final design and development.
2-BB	<i>Standard:</i> Students will develop an understanding of the core concepts of technology. <i>Benchmark:</i> Optimization is an ongoing process or methodology of designing or making a product and is dependent on criteria and constraints.
8-H	<i>Standard:</i> Students will develop an understanding of the attributes of design.

	<i>Benchmark:</i> The design process includes defining a problem, brainstorming, researching and generating ideas, identifying criteria and specifying constraints, exploring possibilities, selecting an approach, developing a design proposal, making a model or prototype.
8-I	<i>Standard:</i> Students will develop an understanding of the attributes of design. <i>Benchmark:</i> Design problems are seldom presented in a clearly defined form.
8-J	<i>Standard:</i> Students will develop an understanding of the attributes of design. <i>Benchmark:</i> The design needs to be continually checked and critiqued, and the ideas of the design must be redefined and improved.
8-K	<i>Standard:</i> Students will develop an understanding of the attributes of design. <i>Benchmark:</i> Requirements of a design, such as criteria, constraints, and efficiency, sometimes compete with each other.
9-I	<i>Standard:</i> Students will develop an understanding of engineering design. <i>Benchmark:</i> Established design principles are used to evaluate existing designs, to collect data, and to guide the design process.
9-J	<i>Standard:</i> Students will develop an understanding of engineering design. <i>Benchmark:</i> Engineering design is influenced by personal characteristics, such as creativity, resourcefulness, and the ability to visualize and think abstractly.
9-K	<i>Standard:</i> Students will develop an understanding of engineering design. <i>Benchmark:</i> A prototype is a working model used to test a design concept by making actual observations and necessary adjustments.
9-L	<i>Standard:</i> Students will develop an understanding of engineering design. <i>Benchmark:</i> The process of engineering design takes into account a number of factors.
11-N	<i>Standard:</i> Students will develop the abilities to apply the design process. <i>Benchmark:</i> Identify criteria and constraints and determine how these will affect the design process.
11-O	<i>Standard:</i> Students will develop the abilities to apply the design process. <i>Benchmark:</i> Refine a design by using prototypes and modeling to ensure quality, efficiency, and productivity of the final product.
11-P	<i>Standard:</i> Students will develop the abilities to apply the design process. <i>Benchmark:</i> Evaluate the design solution using conceptual, physical, and mathematical models at various intervals of the design process in order to check for proper design and to note areas where improvements are needed.
11-Q	<i>Standard:</i> Students will develop the abilities to apply the design process. <i>Benchmark:</i> Develop and produce a product or system using a design process.

11-R	<i>Standard:</i> Students will develop the abilities to apply the design process. <i>Benchmark:</i> R. Evaluate final solutions and communicate observation, processes, and results of the entire design process, using verbal, graphic, quantitative, virtual, and written means, in addition to three-dimensional models.
12-P	<i>Standard:</i> Students will develop the abilities to use and maintain technological products and systems. <i>Benchmark:</i> Use computers and calculators to access, retrieve, organize, process, maintain, interpret, and evaluate data and information in order to communicate.
Next Generation Science Standards	
The Next Generation Science Standards is a multi-state effort to create new education standards that are "rich in content and practice, arranged in a coherent manner across disciplines and grades to provide all students an internationally benchmarked science education." More information can be found here: http://www.nextgenscience.org/ pg 102	
HS.ETS1.2	Engineering Design
	Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.
HS.ETS1.3	Engineering Design
	Evaluate a solution to a complex real-world problem based on prioritized criteria and trade-offs that account for a range of constraints, including cost, safety, reliability, and aesthetics, as well as possible social, cultural, and environmental impacts.
DCI - ETS1.B	Engineering Design - Developing Possible Solutions
	When evaluating solutions, it is important to take into account a range of constraints, including cost, safety, reliability, and aesthetics, and to consider social, cultural, and environmental impacts. (HS-ETS1-3)
DCI - ETS1.C	Engineering Design - Optimizing the Design Solution
	Criteria may need to be broken down into simpler ones that can be approached systematically, and decisions about the priority of certain criteria over others (tradeoffs) may be needed. (secondary to HS-PS1-6)
	<i>Science and Engineering Practice</i> - Planning and Carrying Out Investigations
	Plan and conduct an investigation or test a design solution in a safe and ethical manner including considerations of environmental, social, and personal impacts.
	<i>Science and Engineering Practice</i> - Using Mathematics and Computational Thinking.
	Create and/or revise a computational model or simulation of a phenomenon, designed device, process, or system.

	<i>Crosscutting Concepts - Systems and System Models</i>
	• A system is an organized group of related objects or components; models can be used for understanding and predicting the behavior of systems. • Systems can be designed to do specific tasks. • When investigating or describing a system, the boundaries and initial conditions of the system need to be defined and their inputs and outputs analyzed and described using models. • Models (e.g., physical, mathematical, computer models) can be used to simulate systems and interactions—including energy, matter, and information flows—within and between systems at different scales. • Models can be used to predict the behavior of a system, but these predictions have limited precision and reliability due to the assumptions and approximations inherent in models.
	<i>Connections to Engineering, Technology, and Applications of Science</i>
	Influence of Science, Engineering, and Technology on Society and the Natural World New technologies can have deep impacts on society and the environment, including some that were not anticipated. Analysis of costs and benefits is a critical aspect of decisions about technology. (HS-ETS1-1) (HSET1-3)
ISTE (The International Society for Technology in Education) Standards	
The ISTE Standards serve as a framework for innovation and excellence in learning, teaching and leading. As a body of work, the suite of standards has guided educator practice, school improvement planning, professional growth and advances in curriculum. More information can be found here: https://cdn.iste.org/www-root/PDF/ISTE%20Standards-One-Sheet_Combined_09-2021_vF3.pdf	
1.1	Empowered Learner Students leverage technology to take an active role in choosing, achieving, and demonstrating competency in their learning goals, informed by the learning sciences.
	1.1a Build networks and customize their learning environments in ways that support the learning process.
	1.1b Use technology to seek feedback that informs and improves their practice and to demonstrate their learning in a variety of ways.
	1.1.d. Understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.

1.3	Knowledge Constructor Students critically curate a variety of resources using digital tools to construct knowledge, produce creative artifacts and make meaningful learning experiences for themselves and others. Students:
	1.3.c. curate information from digital resources using a variety of tools and methods to create collections of artifacts that demonstrate meaningful connections or conclusions
	1.3.d. build knowledge by actively exploring real-world issues and problems, developing ideas and theories and pursuing answers and solutions
1.4	Innovative Designer Students use a variety of technologies within a design process to identify and solve problems by creating new, useful or imaginative solutions. Students:
	1.4.a. know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems
	1.4.b. select and use digital tools to plan and manage a design process that considers design constraints and calculated risks.
	1.4.c. develop, test and refine prototypes as part of a cyclical design process.
	1.4.d. exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems
1.5	Computational Thinker Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions. Students:
	1.5.a. formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.
	1.5.b. collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.
	1.5.c. break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving
	1.5.d. understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

1.6	Creative Communicator Students communicate clearly and express themselves creatively for a variety of purposes using the platforms, tools, styles, formats and digital media appropriate to their goals. Students:
	1.6.a. choose the appropriate platforms and tools for meeting the desired objectives of their creation or communication
	1.6.c. communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models or simulations
1.7	Global Collaborator Students use digital tools to broaden their perspectives and enrich their learning by collaborating with others and working effectively in teams locally and globally. Students:
	1.7.b. use collaborative technologies to work with others, including peers, experts or community members, to examine issues and problems from multiple viewpoints.
	1.7.c. contribute constructively to project teams, assuming various roles and responsibilities to work effectively toward a common goal
	1.7.d. explore local and global issues and use collaborative technologies to work with others to investigate solutions.