# 5 Blockly - Using the Color Sensor

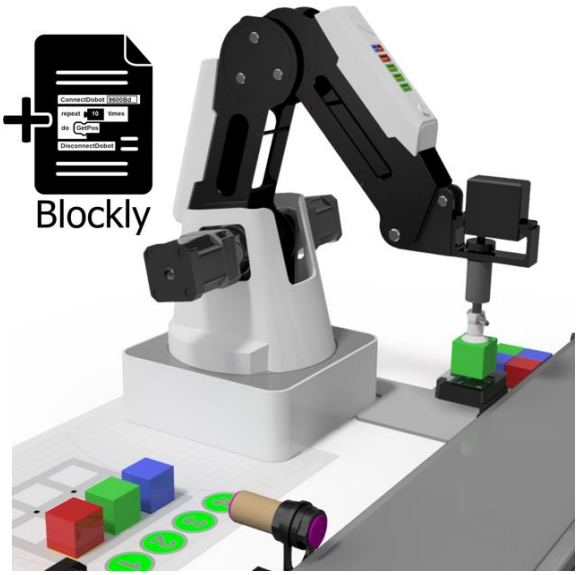
*NAME:* _____ *Date:* _____ *Section:* _____

## *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.





***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
- 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
- DobotStudio software
- Suction Cup Gripper
- *Dobot Input/Output Guide*
- 3 Small Containers

## *ESSENTIAL QUESTIONS*

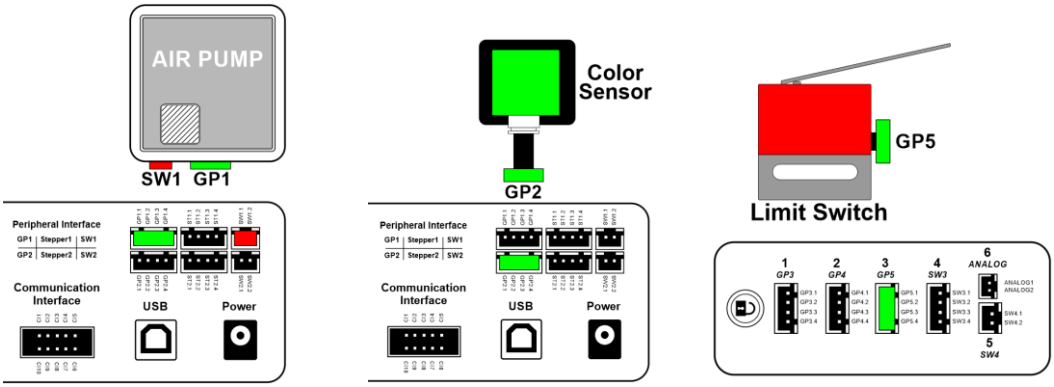
Essential questions answered in this activity include:

- 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?

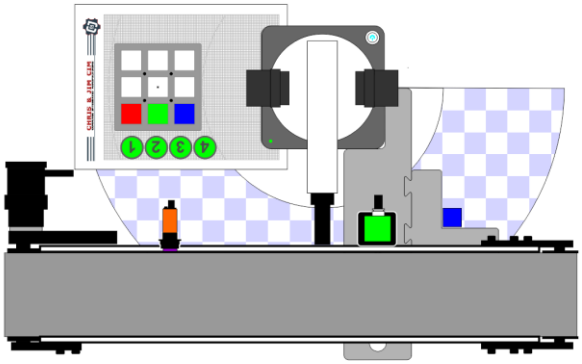
## *PROCEDURE*


***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 the robot with a suction cup - ***GP1 & SW1***
2. Wire the robot with the Color Sensor plugged into port ***GP2***
3. Wire the robot with the Limit Switch plugged into port ***GP5 - EI05***




Set up the robot, conveyor, and color sensor as shown in the diagram below:

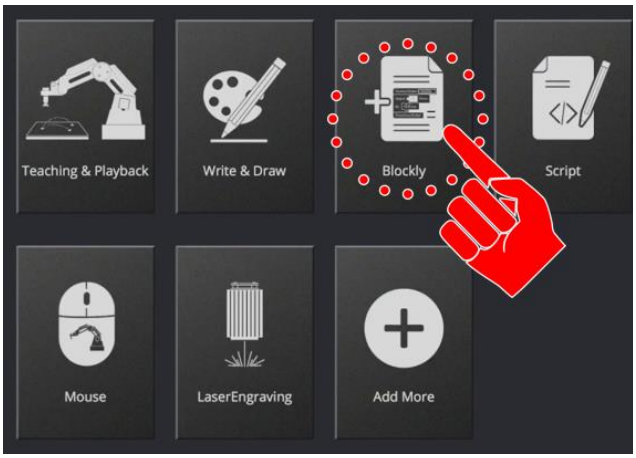


## ***Order of operations***

- Start with the robot 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 up Blockly in the software

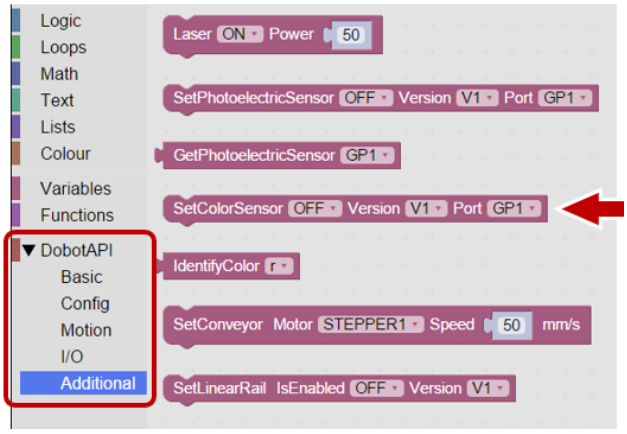

*When you re-open this program check that the name of the file on top matches the code in the file, if it does not, you may end up overwriting another program.*




The first step in programming this activity is to set up the inputs.

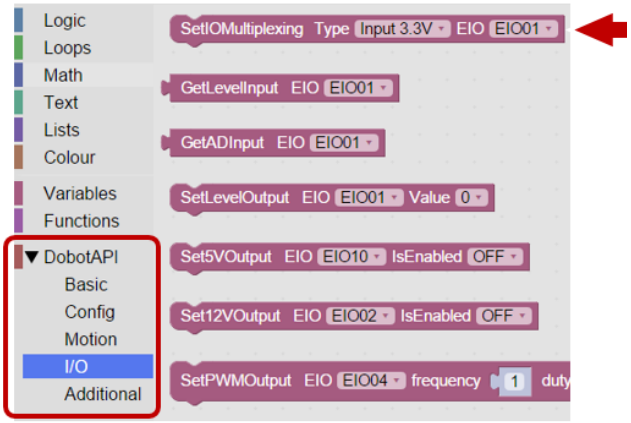
Drag over the ***SetColorSensor*** from the *DobotAPI/Additional* tool box

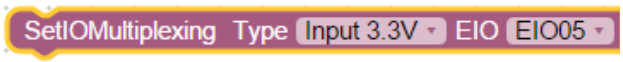
**Set the sensor to ON / V2 / GP2**




Next, set up the limit switch as we have done in past activities.

**Input 3.3V GP5  EI05**



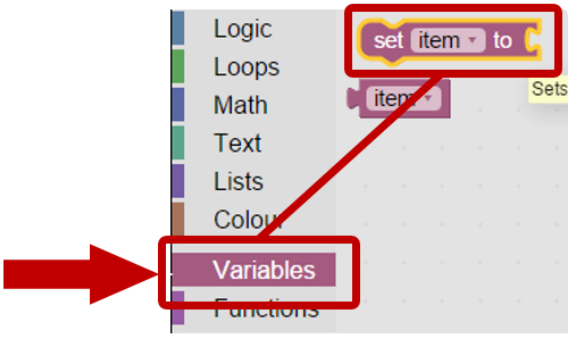



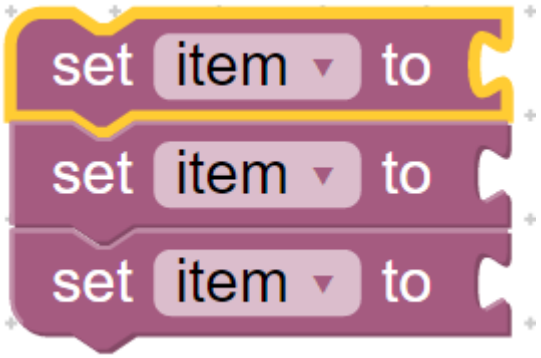

## COLOR SENSOR SETUP

The next set of commands are needed to get the values from the color sensor to report the correct values to our program/

### Step 1

Bring over and link together three ***Set Variables to Input*** blocks

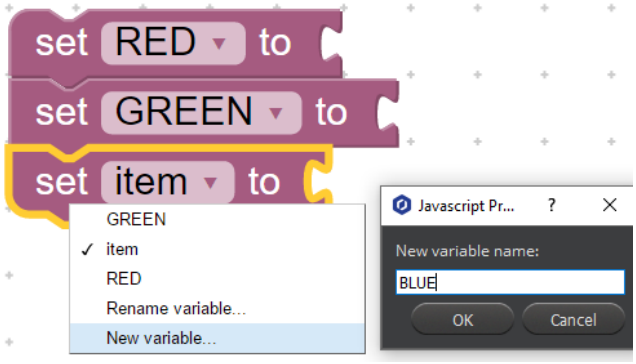


Select *item* for each block and create a New Variable.


We need to create a *variable* for each color that we will use in our program

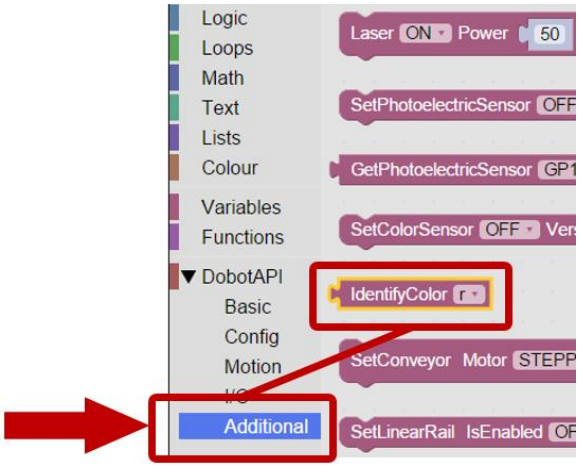
Create *variables* named:
RED
GREEN
BLUE



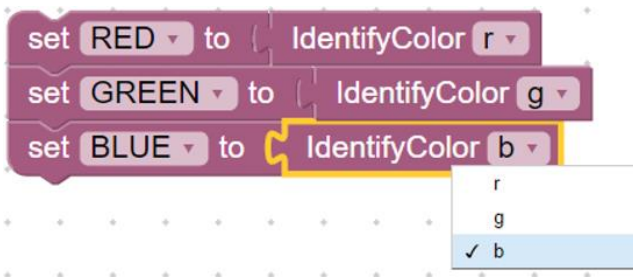

## Step 2

The next step is to set each individual variable to a pre-set range collected from the Color Sensor

The ***IdentifyColor*** command is used to identify three basic colors: Blue, Green, & Red. This block returns a true or 1 value when the specific color is identified and a false or 0 when it is not



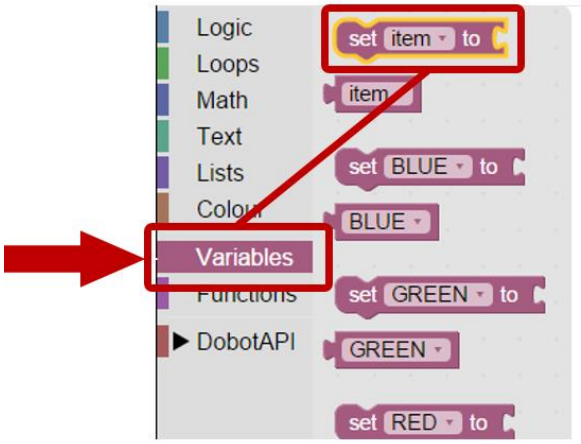

Attach an ***IdentifyColor*** block to each of your ***Variables*** and select the corresponding color letter for each block.

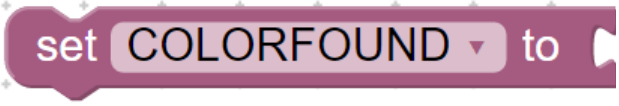


## Step 3

Group or consolidate all three color possibilities into one variable that can be used in the program.

Create one last variable called COLORFOUND.



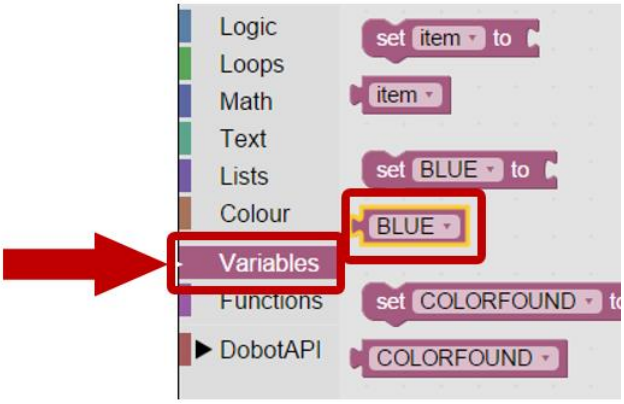




## Step 4

Attach each variable (RED, GREEN, and BLUE) to a *LIST*. The *LIST* will allow us to verify that a color is being read. From this command we will not capture which color is being read, we will just verify that a valid color from the list is being read.

Select the ***Create List With*** block

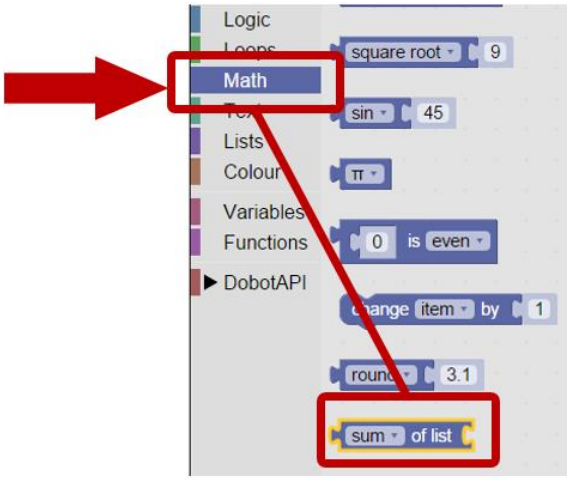
In order to put each variable's value into the list, we need to attach each variable to the list.




Attach the variable to all three empty links on the ***create list with*** block.


Drag over the ***Sum of List*** block. This block is a *RETURN DATA TYPE* command. It will allow us to look statistically at the data from the list.

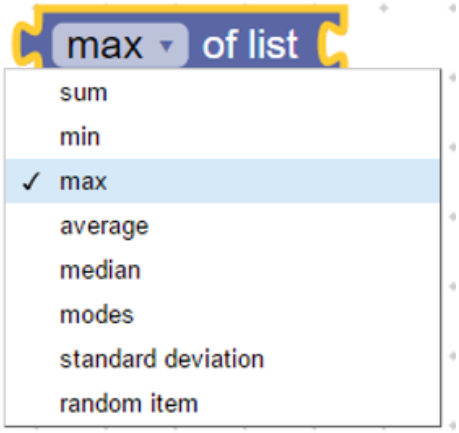
Using the blocks drop down, change *SUM* to *MAX*. Each time a color from the list is read, they will report a 1 to the *RETURN DATA TYPE* block (for this activity, the color variables will only return a 1 or 0). This will allow it to detect if any color is read and write either a 1 or 0 to our COLORFOUND variable.




**Any Color Read, MAX value = 1**

**No Color Read, MAX value = 0**

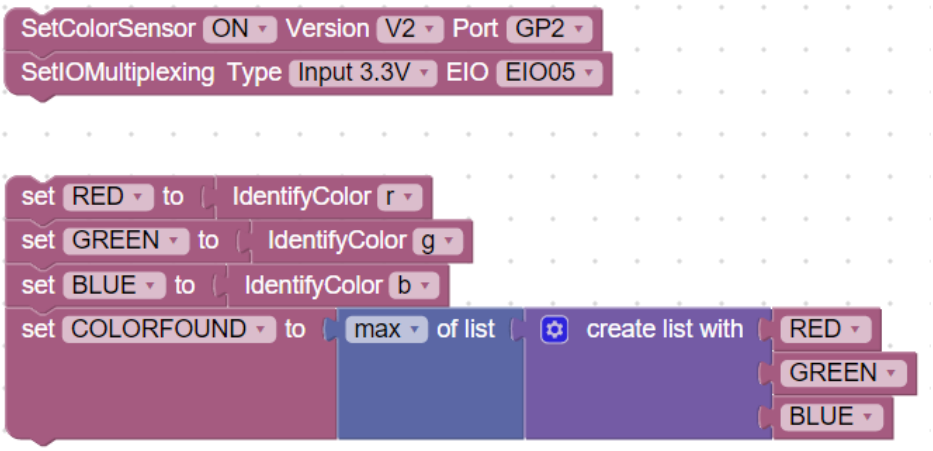
The last step is to attach the *LIST* and the *RETURN DATA* block to the variable COLORFOUND.




Connect all three elements together as seen below

Final Variable - Max Value from the List - Individual Color Variables


We can now link all of our variables together in one grouping as seen below



Now that the sensor is setup, it needs to be tested to make sure everything is reading correctly. The color sensor needs to report what color it is currently reading and is setup in Blockly to only read RED, BLUE, and GREEN.

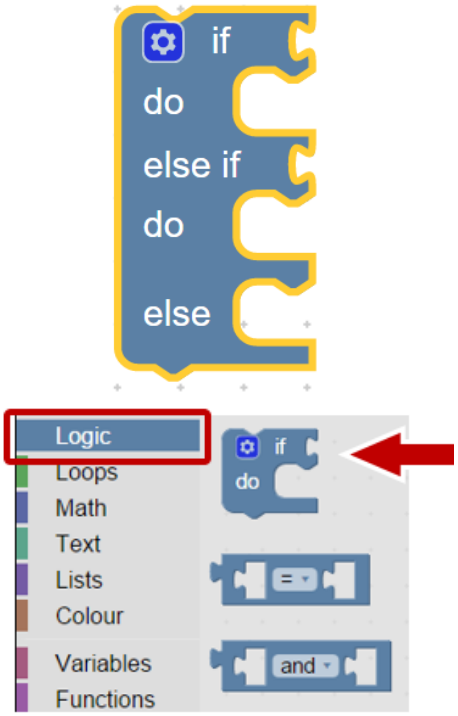
A loop needs to be created that will show in the Running Log the following information:

**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/ELSE IF/ELSE Loop***




There are two new commands that will be needed to check the cube's color and decide which location to take the cube to.

1. ***If/Else If/Else Statement***
2. ***IdentifyColor*** - Uses the current value from the color sensor

"If" statements are used when 2 or more conditions need to be evaluated. A three part "If" statement needs to be set up.

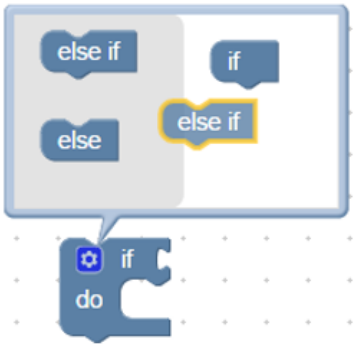
**IF** the value is Identified as RED, do this

**ELSEIF** the value is GREEN, do this

**ELSE** do this (Else will be read as anything other than the RED or GREEN cube which will include our BLUE cube.

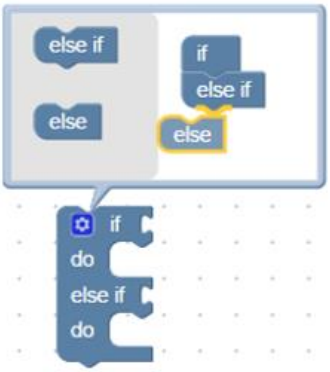

*The **If** structure used in this example will allow a NO CUBE or INVALID COLOR to be read as BLUE since BLUE is setup as the **else** condition. If your program is only reading BLUE, it may not be reading a color at all.*

In order to get the Logic If statement to expand to the three conditions we need, click on the gear next to the word if.




Drag the *Else If* over to the bottom pin connection of the *If* in the expansion window.

Do the same for the *else*.




This will build the three part IF statement that we need.

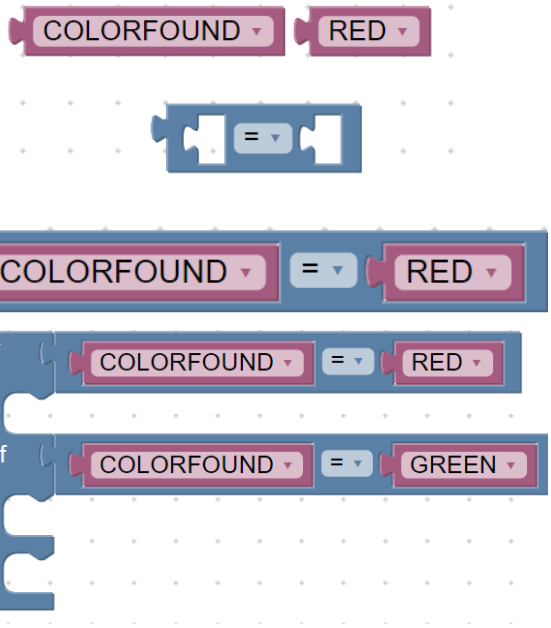
Conditions now must be made for when each level of the IF structure runs.


Drag over the ***Return True*** block from the logic section.

This block will allow us to compare the variable value *COLORFOUND* to each individual color variable.

When these two match (1 is equal to 1), that level of the IF structure will be ran.

Place the variable *COLORFOUND* and the variable *RED* inside the ***Return True*** block.




We now need to fill in our ***If/Else If/Else*** Statements.

If it is RED, stop evaluating and DO the statements included with the IF

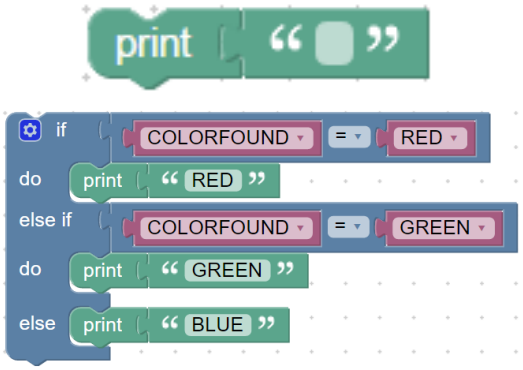
If it is not RED, it must be GREEN or BLUE

If it is GREEN, stop evaluating and DO the statements included with the ELSE IF

If it is not GREEN, it must be BLUE. Stop evaluating and DO the statements included with the ELSE

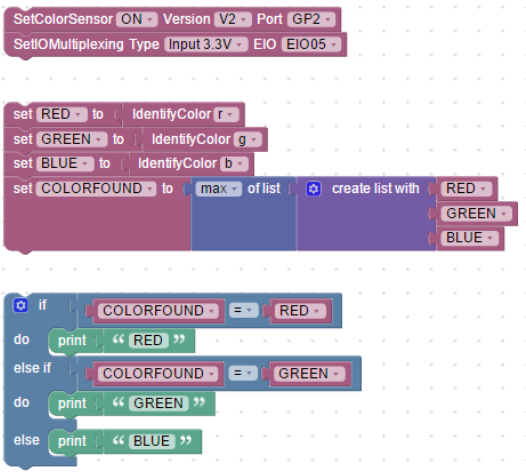
As our programs start getting more complex, it may help to start adding print commands to the code in order to make it easier to troubleshoot.

Add the ***Print*** block into each statement. These values will report to the Running Log so that you can see what is going on in the program.

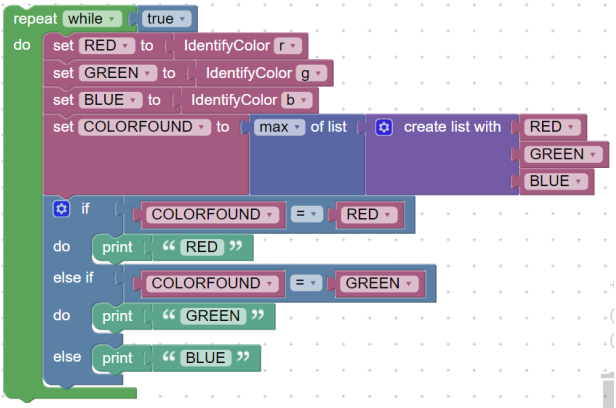


We now have three separate groups of code

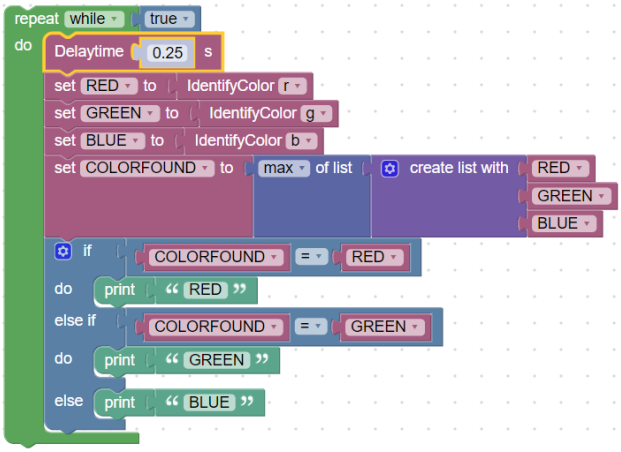
1. Setting Up Inputs and Outputs
2. Setting Up Variables
3. If Structure



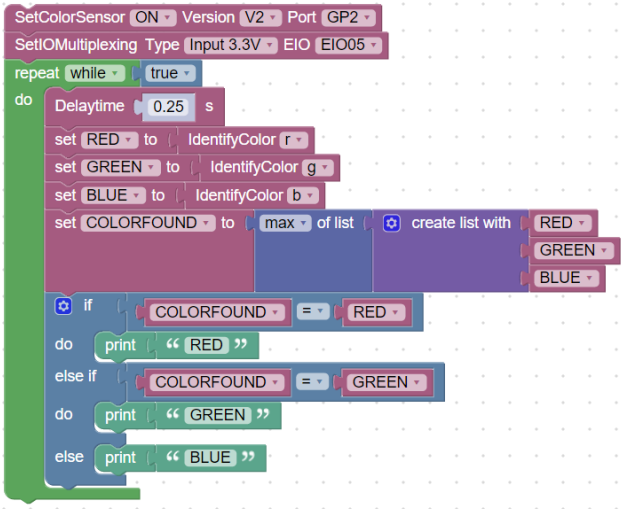

In order to get the If statements to constantly look for new values, both the *variables* and the If structure need to be put into a Forever Loop




We will also add a small delay *(from the DobotAPI/Basic section)* to slow down the looping process at the beginning of each loop.

Add together our header code and the color checking code so that it looks like the diagram here.




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 reads 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 single missed, or incorrect reading as it changes between colors. Why?

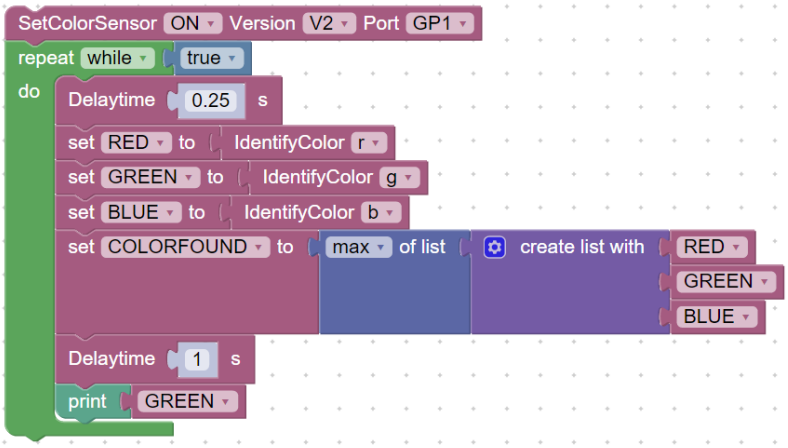
*If your set up did not work correctly the first time, what did you have to do to make it work?*

# *TROUBLESHOOTING*

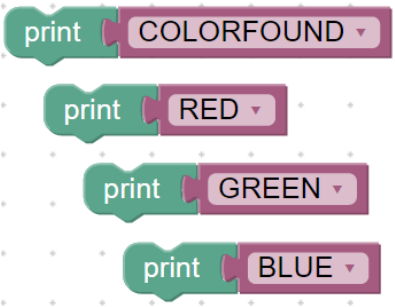
*Checking Individual Variables*

Develop a code similar to the one on the right.

The print can be replaced with each variable to isolate which ones are ready correctly, and which ones are not.




Each variable should read as 1 when they are true and 0 when they are false. The COLORFOUND will read as a 1 if it detects RED, GREEN, or BLUE (YELLOW reads as GREEN).




HELPFUL TIPS

*The color sensor works best if the object is held at a consistent distance from the sensor Dobot suggested distance = (5mm to 10mm) or (¼ to ⅜)*

HELPFUL TIPS

*The different versions of the color sensors can come with different shades of colored cubes. The V1 sensors tend to come with a lighter green and blue than the V2 sensors. This can cause an issue with the sensors reading correctly. The V2 sensor tend to be more forgiving when it comes to shades of colors.*

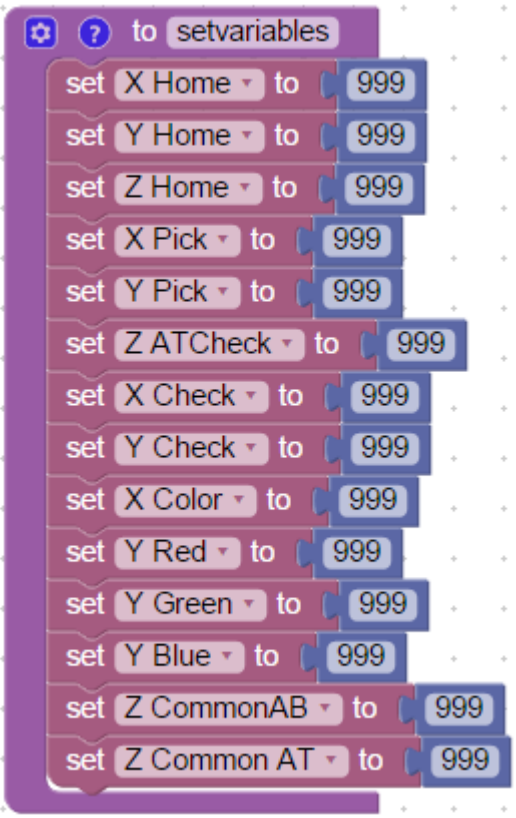
*Remember that the color sensor is an analog sensor that reports a range of variable values as it detects colors to the program. Since we cannot see or adjust this number, you may need to play around with different colors and distances .*

## Now that we have the color sensor reading correctly, we can start developing the rest of the program.

In order to keep our program short and organized we will use several *functions/voids*.

Create the first *function* which will house all of our variables for the entire activity.

Call this function **setvariables.**




HELPFUL TIPS

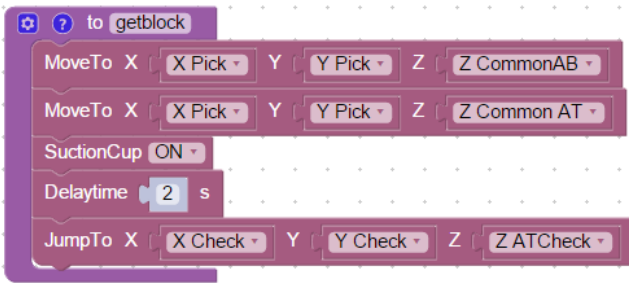
***The Z values can be shared for both the pick and place positions.***

***The X Value can be shared for all of the place positions***

***The Lower Z Value for the Color Sensor needs to be about 10mm above the sensor face. This height allows the sensor to catch the reflection off the block to determine its color.***

The second *Function* will go and get the block from a common location and JUMP to the Z ATCheck for the color sensor

Call this function **getblock**



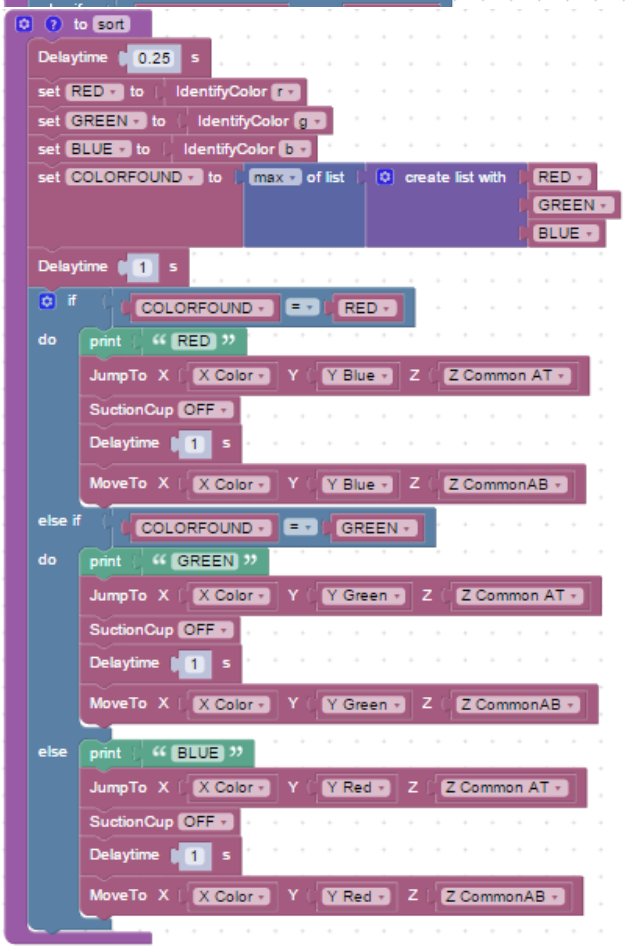
Add all the necessary JumpTo and MoveTo commands for each individual color.

The blocks are in a row. They should all share the same X value


Once the If Statements are complete, we will place it into its own function labeled **sort**.

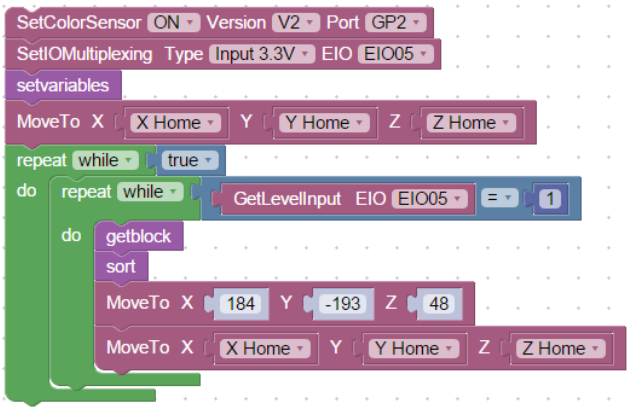
Each Statement will include:

1. A short ***DelayTime*** before the IF Statement to ensure the cube is in place and the color has been evaluated correctly (Remember some of the error readings you may have received earlier during your testing)
2. *Jump* to correct color sort location
3. Turn the Vacuum off
4. Delay - 1 Second
5. Move up

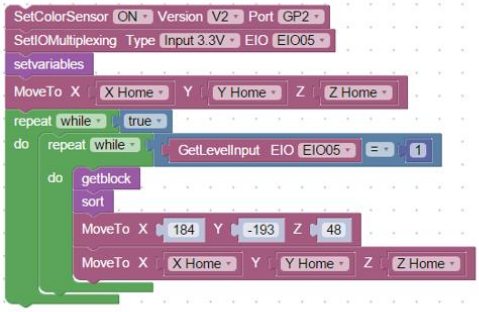



Finish developing the main program by creating two loops

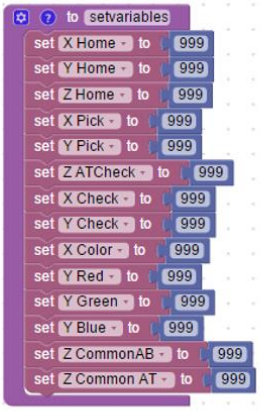
1. Forever Loop
2. Conditional Input Loop
    a. Get the block and move it to the sensor
    b. Identify the color and move it to the correct location
    c. Move to home



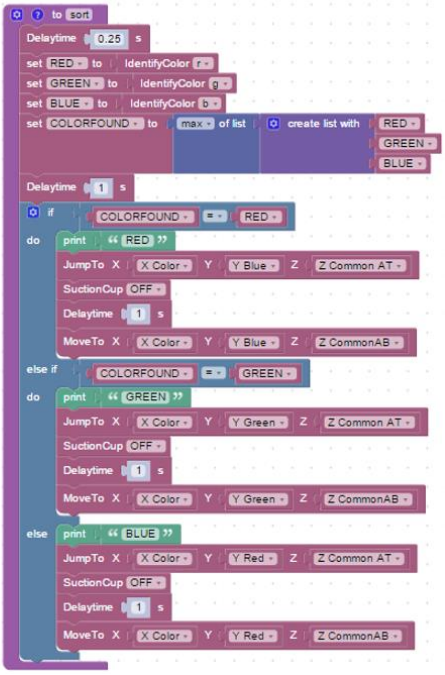
## Main Program




## Set Variables




## Color Sort




## Get Block & Move to Sensor


*All of the separate sections of the program should look like this.*


***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? See a simple color checking program below the "Going Beyond Section"

*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.