9 Blockly - Handshaking -VEX to VEX

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 can be just as simple as having two robots communicate. It is a very simple form of communication and 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.

The limit switch attached to each microcontroller will control the on and off function of the other controllers motor.

KEY VOCABULARY

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

EQUIPMENT & SUPPLIES

- 2 VEX Cortex
- 2 Limit Switches
- 2 VEX Cables
- Servo Extension Cables
- Handshake Modules
- 2 Motors
- 2 VEX Batteries
-

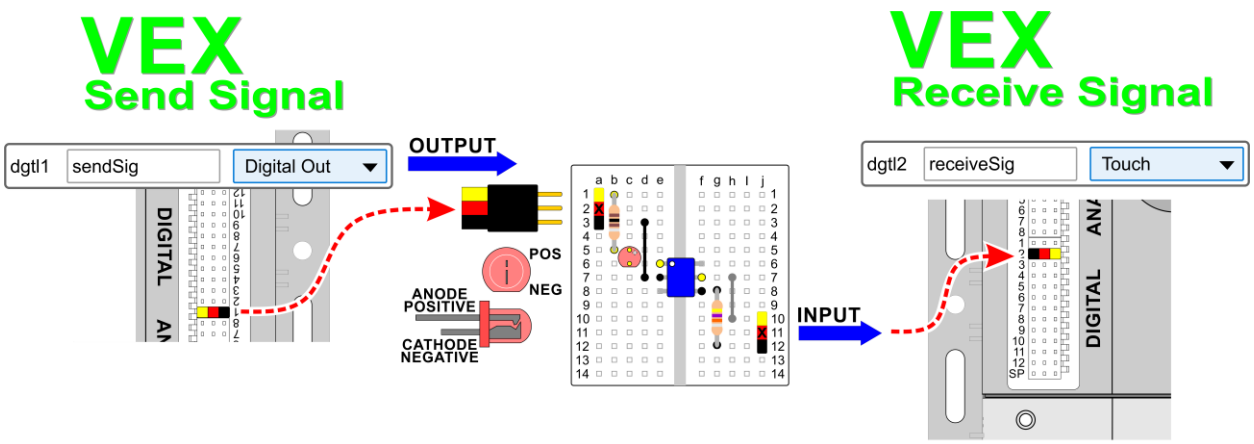
ESSENTIAL QUESTIONS

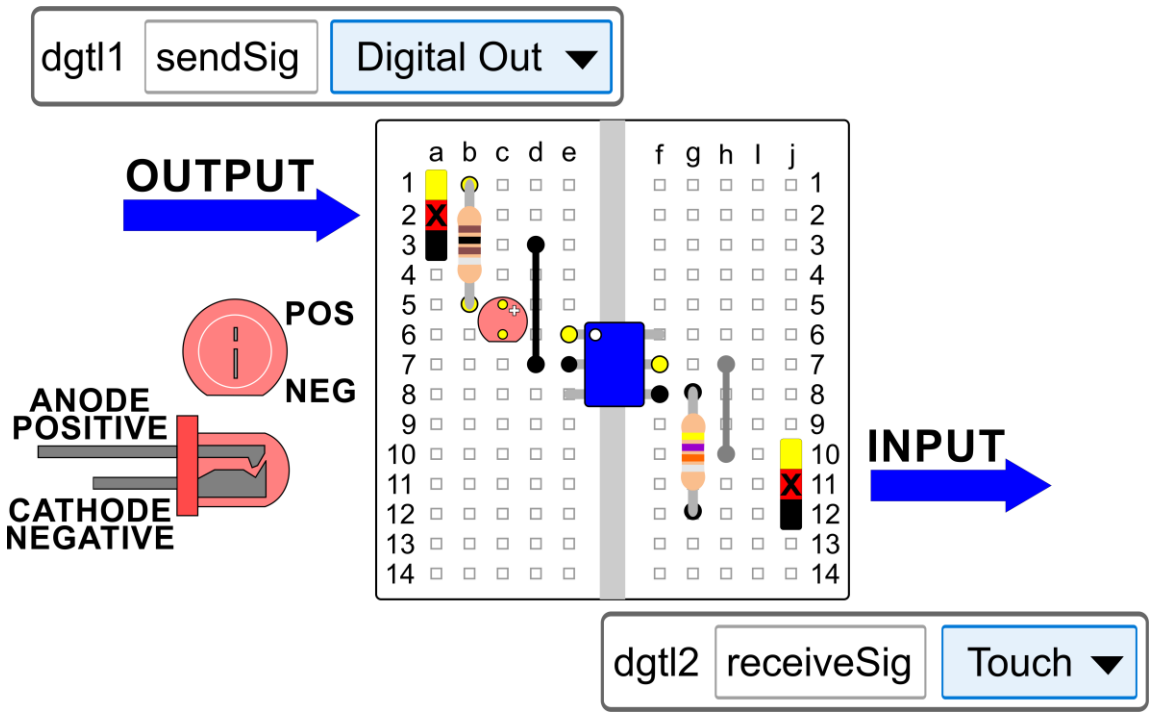
Essential questions answered in this activity include:

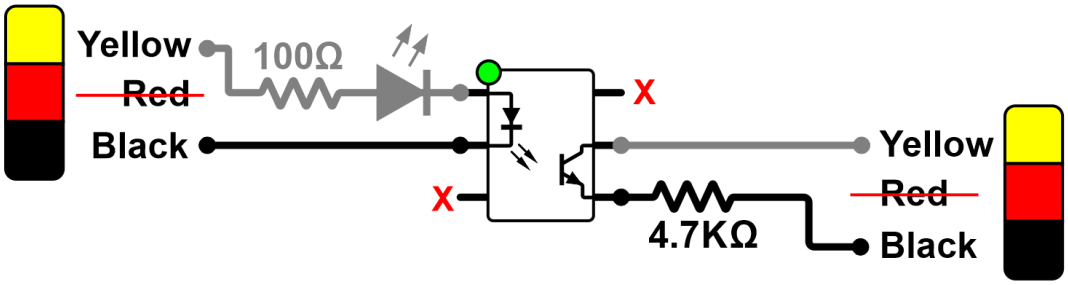
- How do I make my VEX Cortex send and receive signals?
- What kind of software is necessary to communicate?
- How do I wire the hardware to communicate?
- How is RobotC similar to Arduino software? How is it different?

PROCEDURE

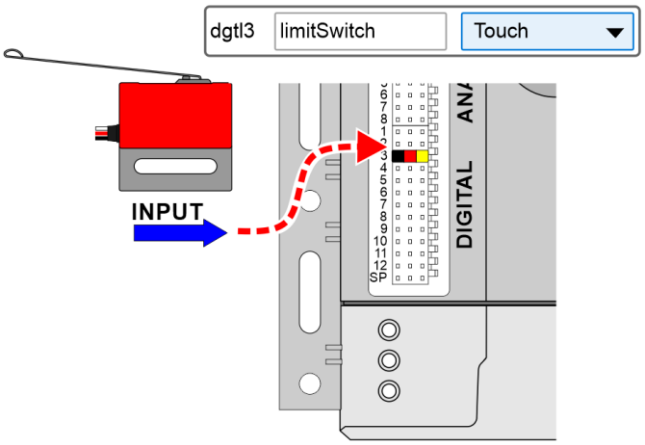
1. The *digital output* of each Cortex (dgtl1) will be attached to a touch input (dgtl2) of the other Cortex
2. Build and wire 2 handshake modules. One as seen below and one mirrored.





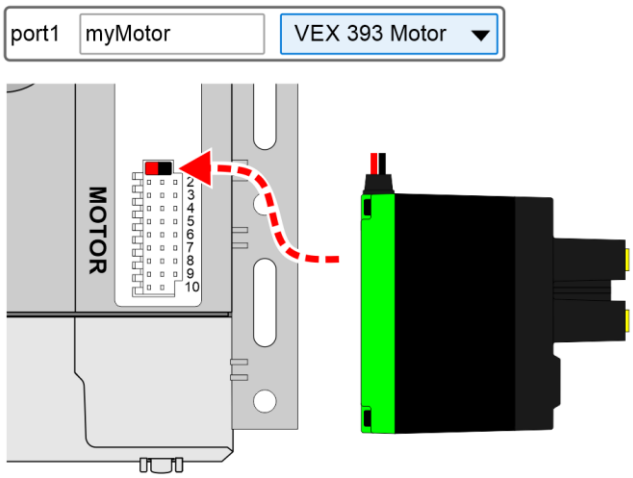


Wire a *limit switch* to each cortex as a Touch on dgtl3.

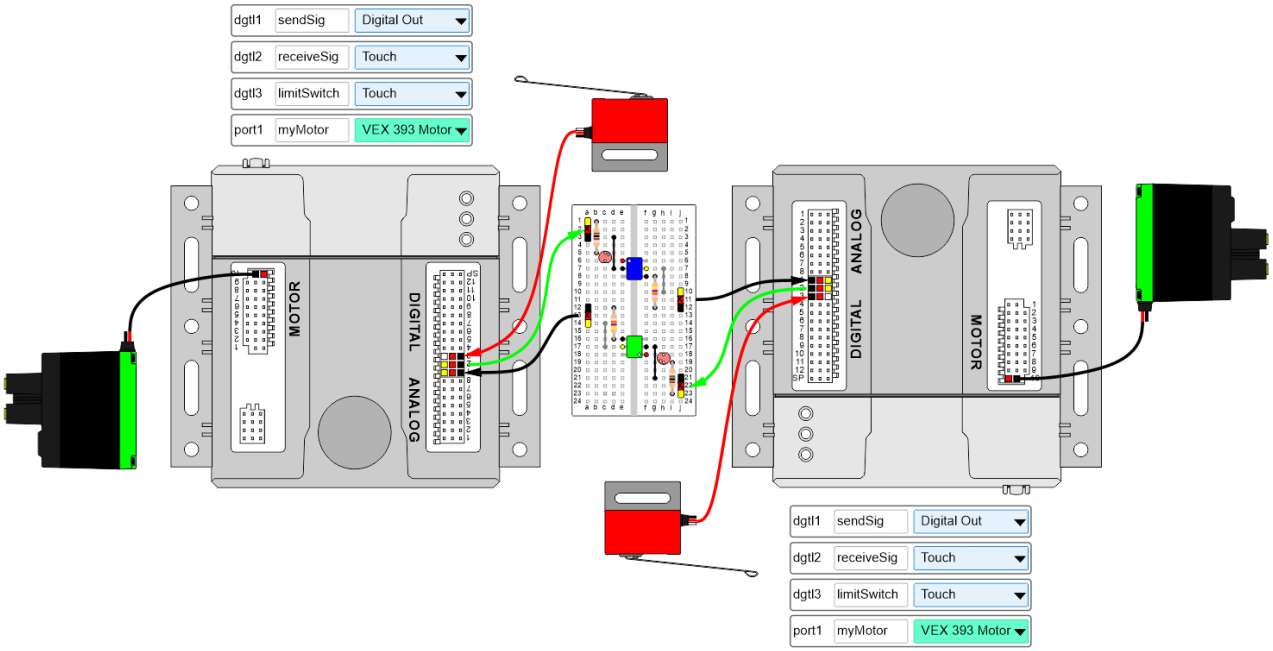


Repeat the process for the other Cortex

Wire a motor to either motor port 1 or port 2.



Repeat the process for the other Cortex



Once all of the wiring is complete develop the syntax programming 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)

PRESS A LIMIT SWITCH	→	SEND SIGNAL
RECEIVE A SIGNAL	→	TURN ON MOTOR

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

STOP PRESSING SWITCH	→	STOP SENDING SIGNAL
STOP RECEIVING A SIGNAL	→	TURN OFF MOTOR

Now start grouping our individual actions into like groups

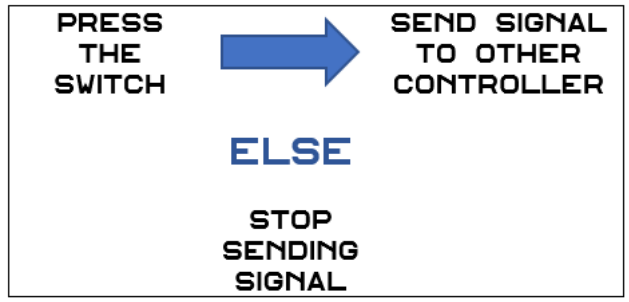
```
if (condition)
{
  body
}
if (condition)
{
  body
}
```

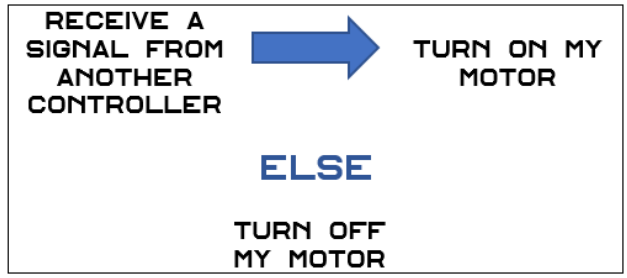
PRESS MY LIMIT SWITCH	→	SEND SIGNAL TO OTHER CONTROLLER
DO NOT PRESS MY LIMIT SWITCH	→	STOP SENDING SIGNAL TO OTHER CONTROLLER

RECEIVE A SIGNAL FROM OTHER CONTROLLER	→	TURN MY MOTOR ON
DO NOT RECEIVE A SIGNAL	→	TURN MY MOTOR OFF

We should 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.

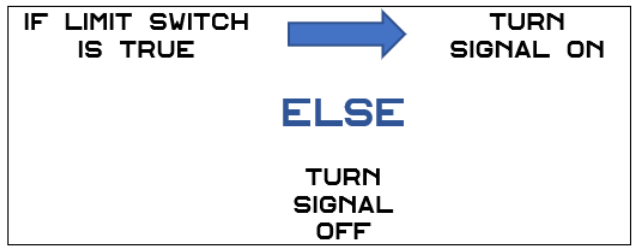
```
if (condition)
{
  body
}
else
{
  body
}
```

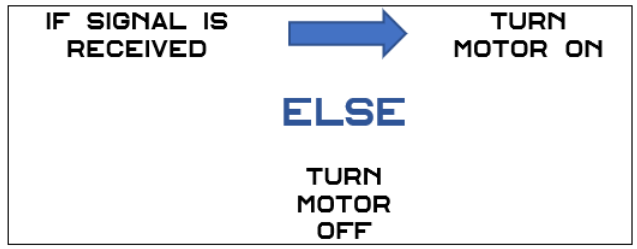




Now simplify the code down to only the essentials

```
if (SensorValue[limitSwitch]==1)
  {
    body
  }
```





In order to get the program to constantly evaluate either or both situations (multiple If statements can be true at one time) we will put our program in a While True Loop

```
while (true)
{
  body
}
```

We will use the SensorValue command each time we need to evaluate or a sensor value or turn on and off the *digital output*.

```
SensorValue[sendSig]=1; // Turn On dgtl1
SensorValue[sendSig]=0; // Turn Off dgtl1
if (SensorValue[limitSwitch]==1)//Is Touch ON?

if (SensorValue[receiveSig]==1)//Is Touch ON?
```

Reminder

1 (=) sets a value

2 (==) evaluates a value

An example of the entire program in RobotC is seen below.

Program Only

```
task main()
{
  while (true)
  {
    if (condition)
    {
      body
    }
    else
    {
      body
    }
    if (condition)
    {
      body
    }
    else
    {
      body
    }

  }
}
```

Program with Comments

```
task main()
{
  while (true) //Forever Loop
  {
    if (condition)//Limit Switch is ON
    {
      body // Turn ON dgtl1
    }
    else
    {
      body// Turn OFF dgtl1
    }
    if (condition)//Signal Received
    {
      body //Motor ON
    }
    else
    {
      body //Motor OFF
    }
  }
}
```

```
#pragma config(Sensor, dgtl1,  sendSig,        sensorDigitalOut)
#pragma config(Sensor, dgtl2,  receiveSig,     sensorTouch)
#pragma config(Sensor, dgtl3,  limitSwitch,    sensorTouch)
#pragma config(Motor,  port1,           myMotor,       tmotorVex393_HBridge, openLoop)
```

```
task main()
{
  while (true) //Forever Loop
  {
    if (SensorValue[limitSwitch]==1)//Limit Switch is ON
    {
      SensorValue[sendSig]=1; // Turn ON dgtll
    }
    else // No Limit Switch
    {
      SensorValue[sendSig]=0;// Turn OFF dgtll
    }
    if (SensorValue[receiveSig]==1)//Signal Received
    {
      startMotor(myMotor, 127); //Motor ON
    }
    else //No Signal
    {
      stopMotor(myMotor); //Motor OFF
    }
  }
}
```

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

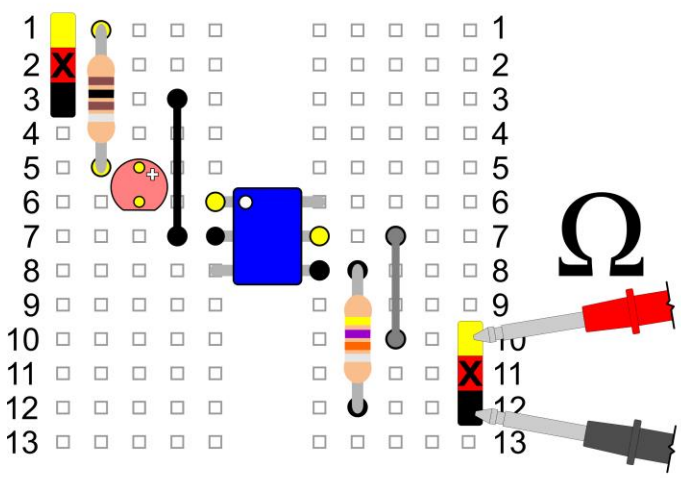
Use the Debugger window to troubleshoot your program. When the Limit Switch is pressed, both the dgtl1 (Send Signal) and dgtl3 (Limit Switch) should read true

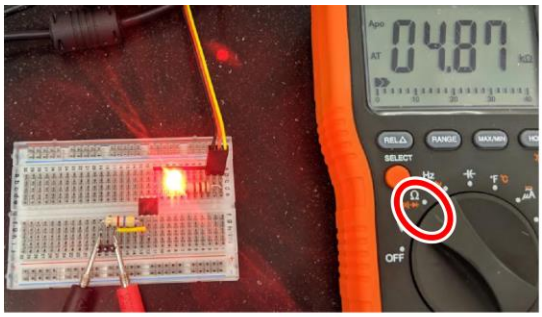
Sensors

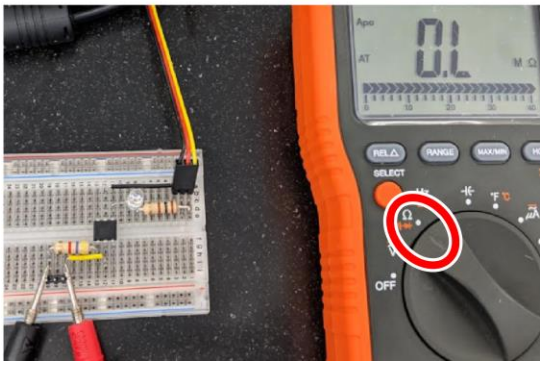
Index	Sensor	Type	Value
dgtl1	sendSig	Digital Out	1
dgtl2	receiveSig	Touch	0
dgtl3	limitSwitch	Touch	1

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

If the Debugger window reads correctly, the LED turns on, but no signal is being seen by the other cortex, 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.







Optical Isolator OFF = 0Ω
Optical Isolator ON = 4.9 kΩ

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 VEX Cortex to start this process? Use the Dobot Input/Output 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 VEX Cortex? Be sure to check with your instructor first!*

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, and be sure to use the correct inputs and outputs in RobotC.