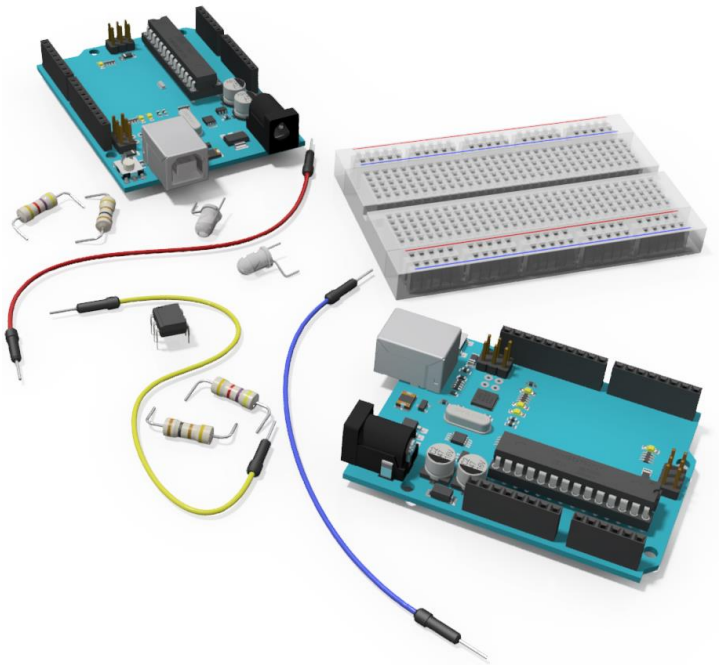
8 Blockly - Handshaking - Arduino to Arduino

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 Arduino microcontrollers.

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

HELPFUL TIPS

This activity was written for arduino uno microcontrollers, but the principles taught in this activity could be applied to any other microcontroller as well. The principles of communication between devices are the same, but the hardware and software setup may be different. Even when using different "brands" of arduinos, the programming may be a bit different. If you try something as presented here, and it does not work with your microcontroller, you may have to do some troubleshooting to get it to work. It is also advisable to check the documentation for microcontrollers, as well as any user forums on line as well.

KEY VOCABULARY

- Limit Switch
- If/Else Statement
- Loop
- Void/Function
- Handshaking

EQUIPMENT & SUPPLIES

- 2 Arduino Uno Microcontrollers
- 2 Limit Switches
- 2 USB Cables
- Jumper Wires
- Handshake Modules
- 4 LEDs
- 4 100 Ohm Resistors
- 2 4.7K Ohm Resistors

ESSENTIAL QUESTIONS

Essential questions answered in this activity include:

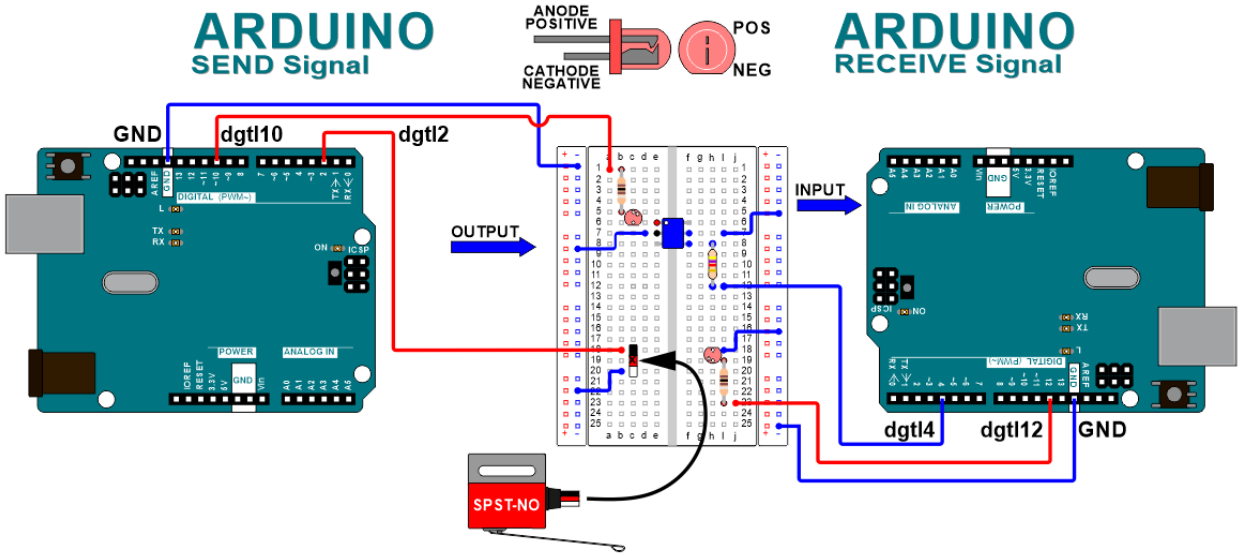
- How do I make my microcontroller send and receive signals?
- What kind of software is necessary to communicate?
- How do I wire the hardware to communicate?
- How do I troubleshoot a complex system?

PROCEDURE

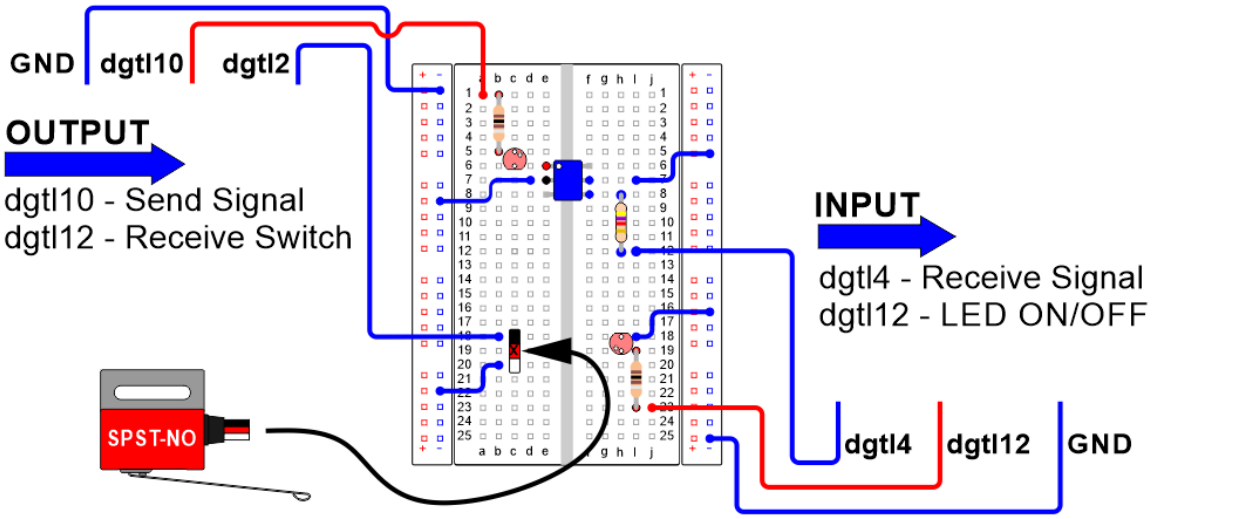
Each Arduino will be able to control the ON/OFF of the LED on the other Arduino.

1. The *digital output* of each Arduino will be attached to a *digital input* of the other Arduino
2. A SPST Normally Open (N.O.) *limit switch* will be connected to each Arduino
3. An LED will be used as a controllable output on each LED
4. Build and wire 2 handshake modules. One as seen below and one mirrored.

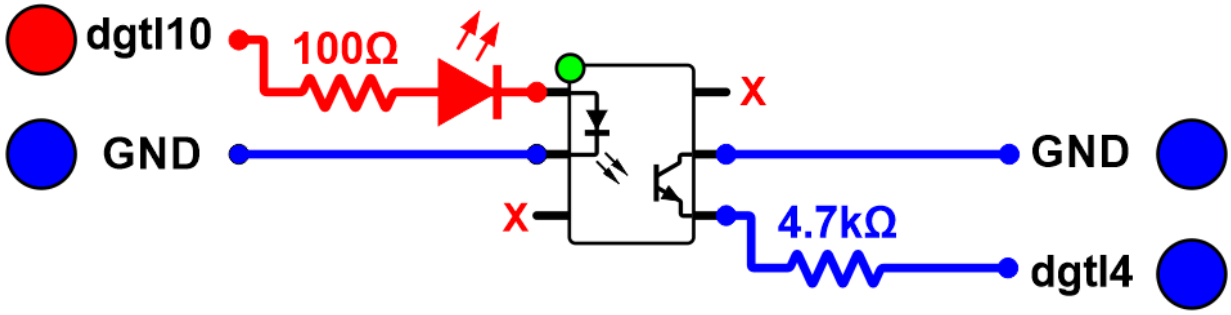
The wiring diagram to send a signal one way is shown below:



```
void setup() {
pinMode (2, INPUT_PULLUP); // Set Port 2 to Digital Pullup (Limit Switch)
pinMode (10, OUTPUT); // Set Port 10 to Output (Send Signal)
}
```

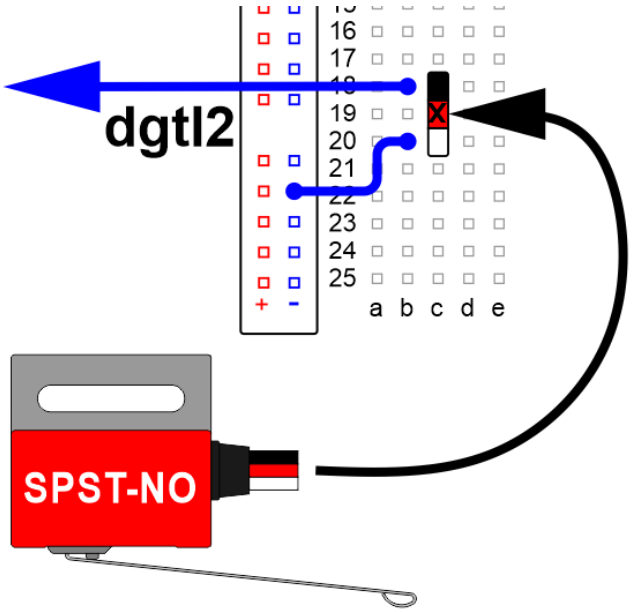



```
void setup() {
pinMode (4, INPUT_PULLUP); // Set Port 4 to Digital Pullup (Receive Signal)
pinMode (12, OUTPUT); // Set Port 12 to Output (Turn On LED)
}
```



Wire a *limit switch* to each Arduino on dgtl2.

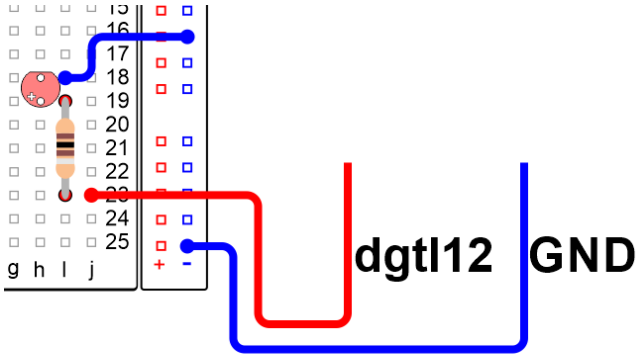
Repeat the process for the other Arduino




```
pinMode (2, INPUT_PULLUP);
```

Wire an LED to dgtl12.

Repeat the process for the other Arduino




```
pinMode (12, OUTPUT);
```

```
void setup() {
// ****************************************
// Send Setup
pinMode (2, INPUT_PULLUP); // Set Port 2 to Digital Pullup (Limit Switch)
pinMode (10, OUTPUT); // Set Port 10 to Output (Send Signal)
// ****************************************
// Receive Setup
pinMode (4, INPUT_PULLUP); // Set Port 4 to Digital Pullup (Receive Signal)
pinMode (12, OUTPUT); // Set Port 12 to Output (Turn On LED)
}
```

Once all of the wiring is complete start developing the syntax programming for this activity. Start by defining the individual steps 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 LED

Now take care of the other conditions….

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

What has to be done when we are not sending or receiving signals?

Start by grouping individual steps into like groups

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

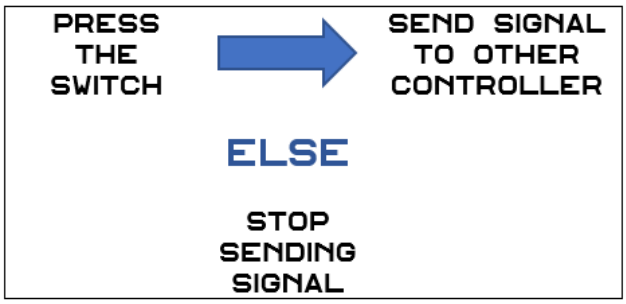
AND

RECEIVE A SIGNAL FROM OTHER CONTROLLER	→	TURN ON MY LED
DO NOT PRESS MY LIMIT SWITCH	→	STOP SENDING SIGNAL TO OTHER CONTROLLER

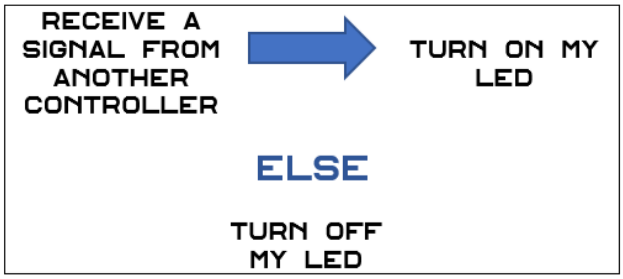
```
if(Condition) {
   // Body
}
if(Condition) {
   // Body
}
```

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
}
```



OR



We will use a digitalRead() command to evaluate if an input signal is high or low.

```
digitalRead (2)==LOW
```

Reminder

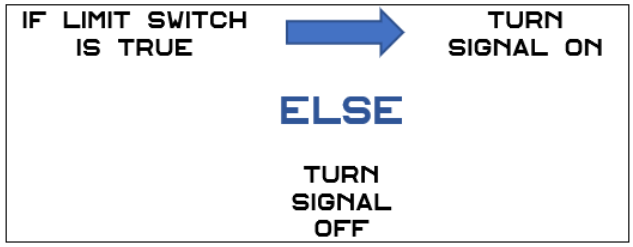
1 (=) sets a value

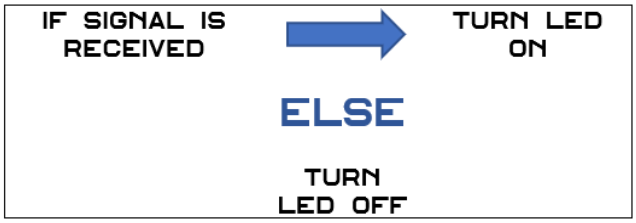
2 (==) evaluates a value

Because we are using dgtl2 as a pullup input, all of its values will read backwards. On is Off and Off is on… HIGH is OFF and LOW is ON

Now simplify the code down to only the essentials

```
if(digitalRead (2) ==LOW) {
   // Body
}
else{
   // Body
}
```





Instead of having to read the condition in our IF statement as ***digitalRead(2)***, we can set the port value to a variable. This will make reading and troubleshooting our programs easier.

An example is shown to the right.

```
int limitSwitch =  digitalRead (2);
```

We can now use limitSwitch in place if digitalRead(2).

```
if(limitSwitch ==LOW) {
   // Body
}
else{
   // Body
}
```

In order to get the program to constantly evaluate either or both situations (multiple If statements can be true at one time) put the program inside the function LOOP.

```
void loop() {
// Body
}
```

Use the digitalWrite command each time we need to turn on and off the digital output.

```
digitalWrite(10, HIGH); // Set Port10 to ON
```

The entire program is shown in the example here:

```
void loop() {
//************************************
//Send Signal Conditions
int Variable =  Port;
   if(Condition) {
     //Body
   }
   else{
     //Body
   }
//************************************
//Receive Signal Conditions
int Variable =  Port;
   if(Condition) {
     //Body
   }
   else{
     //Body
   }
}
```

```
void loop() {
//************************************
//Send Signal Conditions
int limitSwitch =  digitalRead (2);
   if(limitSwitch==LOW) {
     digitalWrite(10, HIGH);
   }
   else{
     digitalWrite(10, LOW);
   }
//************************************
//Receive Signal Conditions
int receiveSignal =  digitalRead (4);
   if(limitSwitch==LOW) {
     digitalWrite(12, HIGH);
   }
   else{
     digitalWrite(12, LOW);
   }
}
```

```
void setup() {
//************************************
//Send Signal Setup
pinMode (2, INPUT_PULLUP); // Set Port 2 to Digital Pullup
pinMode (10, OUTPUT); // Set Port 2 to Output
//************************************
//Receive Signal Setup
pinMode (4, INPUT_PULLUP); // Set Port 2 to Digital Pullup
pinMode (12, OUTPUT); // Set Port 2 to Output
}
```

```
void loop() {
//***********************************
//Send Signal Conditions
int limitSwitch =  digitalRead (2); // Set Variable limitSwitch to Port2 Value
   if(limitSwitch==LOW) { // When the Limit Switch is pressed, Send Signal
      digitalWrite(10, HIGH); // Set Port10 to ON
   }
   else{ // While the Limit Switch is not Pressed
      digitalWrite(10, LOW); // Set Port10 to OFF
   }
//***********************************
//Receive Signal Conditions
int receiveSignal =  digitalRead (4); // Set Variable receiveSignal to Port4 Value
   if(limitSwitch==LOW) { // When the Signal is Received, LED ON
      digitalWrite(12, HIGH); // Set Port12 to ON
   }
   else{ // While the Limit Switch is not Pressed
      digitalWrite(12, LOW); // Set Port12 to OFF (LED 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.

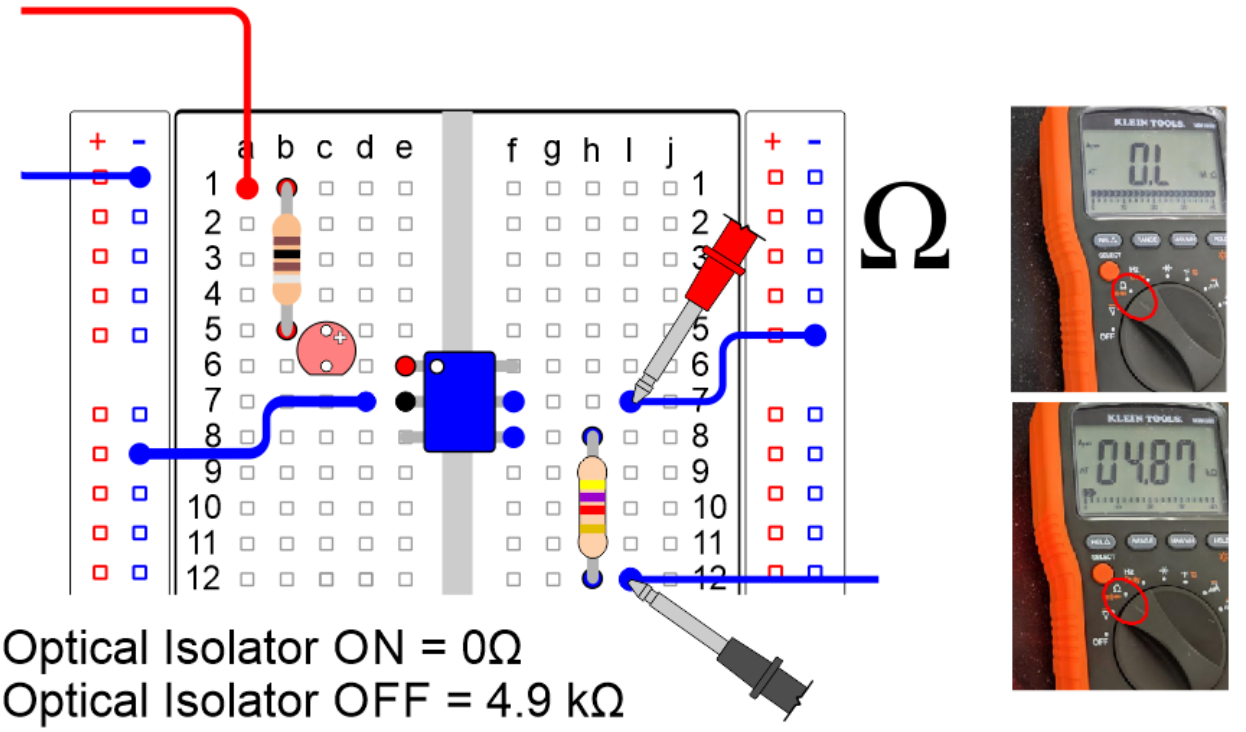
If your set up did not work correctly the first time, what did you have to do to make it work?

TROUBLESHOOTING THE OPTICAL ISOLATOR

HELPFUL TIPS

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 Arduino, 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 are some similarities of microcontroller programming and blockly? List at least two and explain how they are similar.*

2. *What are some differences between microcontroller programming and blockly? List at least two and explain how they are different.*

3. *Why is it important to use the optical isolator when communicating? Explain in your own words.*

GOING BEYOND

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

_____ 1. Use a motor instead of an LED.

Be sure to consider what type of motor you are using, and be sure to check with your teacher before attempting this; especially if you switch outputs on the microcontroller.