Pumpkin Project

Day #3: Using Sensors

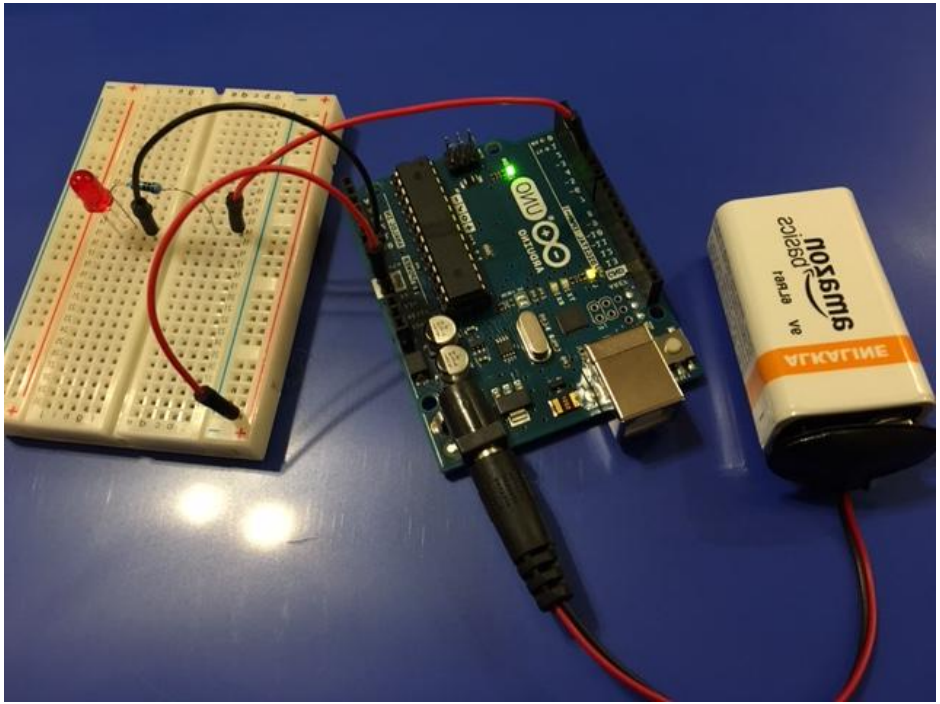
Where we left off

All the groups should have a blinking LED! (or two)

For this step of the project we used a digital pin as output

> We set the voltage to HIGH to light the LED and LOW to turn it off

Today we will see how we can use a digital pin for input to gather information about the world



STEP 7: Add a Sensor

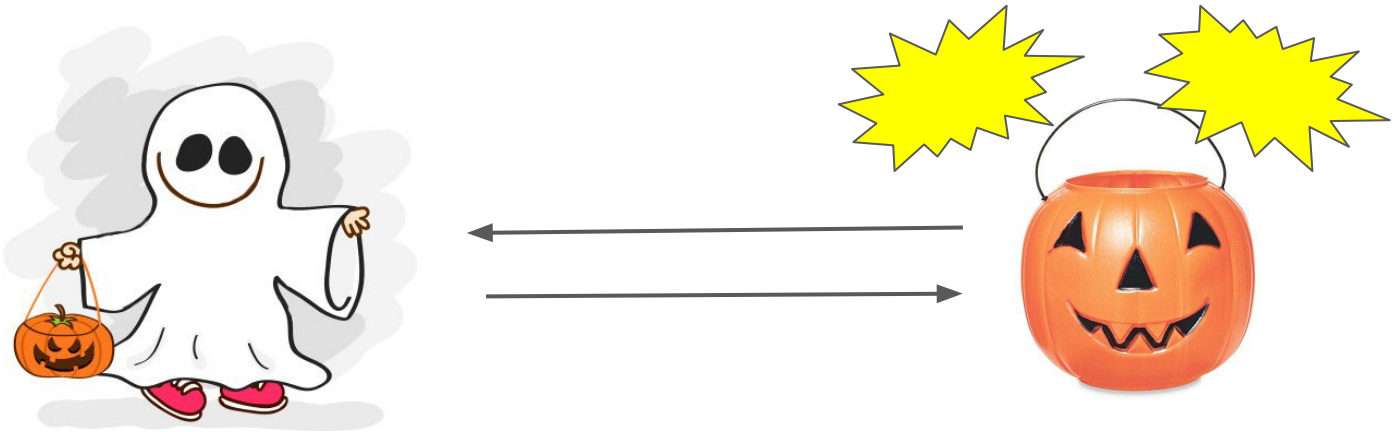
Today we are going to learn about how sensors work and how we can use one in our projects to detect when something gets near our pumpkin.

Adding a Sensor

A **sensor** is a device that detects or measures a physical quantity

Our program can take the value reported from a sensor and use it to decide something about the state of the world and then act accordingly

We want our pumpkin to detect that a trick or treater is approaching and to flash the LEDs in response

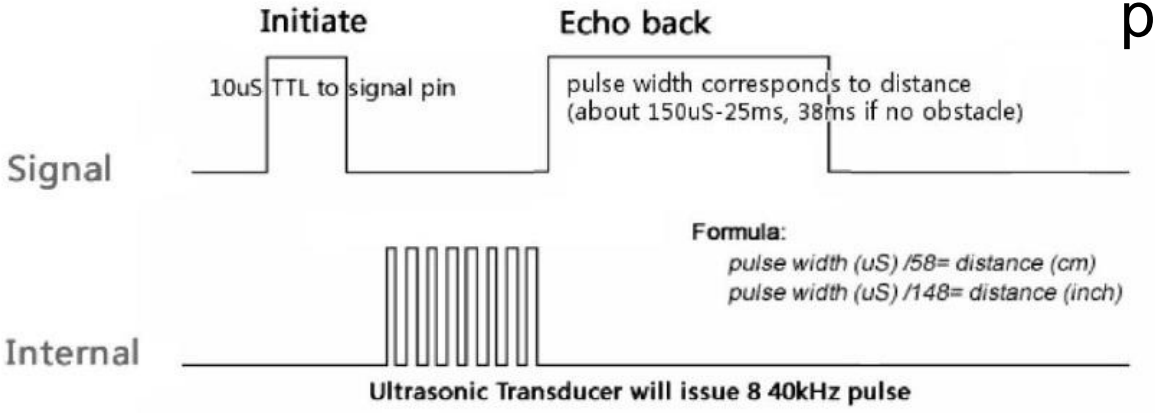


What sensor will we use?

Uses sonar to measure distance

What sensor will we use?

Uses sonar to measure distance



pulse

echo

The time it takes for the echo to return can be used to estimate the distance to an object

What are the steps to using the sensor

1. Wire the sensor into our circuit
2. Write code that reads a value from the sensor

Sensor Data Sheets

The data sheet for a sensor will give you an overview of how the sensor works and how to program it - this is a good place to start

PARALLAX
599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003
General: info@parallax.com
Technical: support@parallax.com
Web Site: www.parallax.com
Educational: www.stampsinclass.com

PING)))™ Ultrasonic Distance Sensor (#28015)

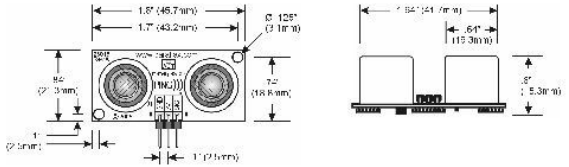
The Parallax PING))) ultrasonic distance sensor provides precise, non-contact distance measurements from about 2 cm (0.8 inches) to 3 meters (3.3 yards). It is very easy to connect to BASIC Stamp® or Javelin Stamp microcontrollers, requiring only one I/O pin.

The PING))) sensor works by transmitting an ultrasonic (well above human hearing range) burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width the distance to target can easily be calculated.

Features

- Supply Voltage – 5 VDC
- Supply Current – 30 mA typ; 35 mA max
- Range – 2 cm to 3 m (0.8 in to 3.3 yrds)
- Input Trigger – positive TTL pulse, 2 uS min, 5 µs typ.
- Echo Pulse – positive TTL pulse, 115 uS to 18.5 ms
- Echo Hold-off – 750 µs from fall of Trigger pulse
- Burst Frequency – 40 kHz for 200 µs
- Burst Indicator LED shows sensor activity
- Delay before next measurement – 200 µs
- Size – 22 mm H x 46 mm W x 16 mm D (0.84 in x 1.8 in x 0.6 in)

Dimensions



Wiring the sensor

VCC = power (5V)

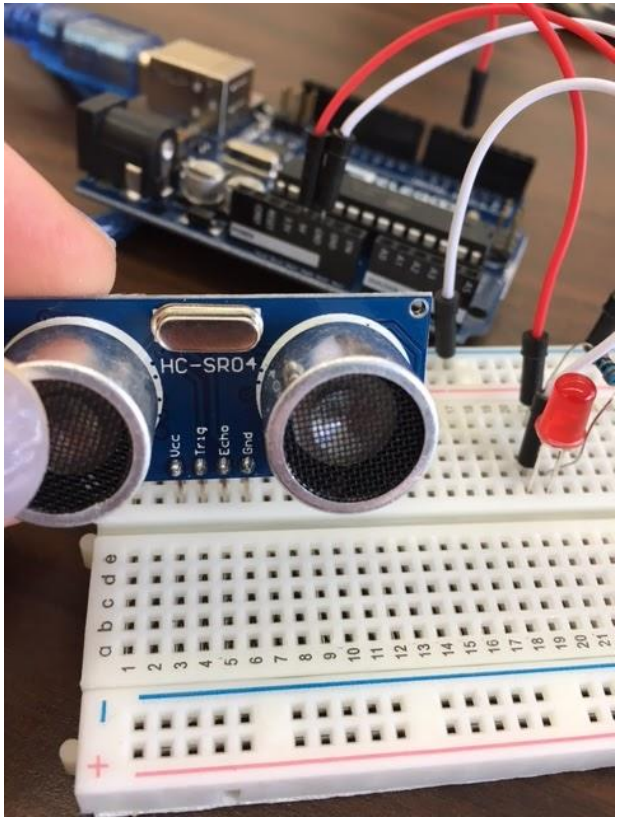
GND = ground

TRIG = trigger pin - use a digital pin on the Arduino

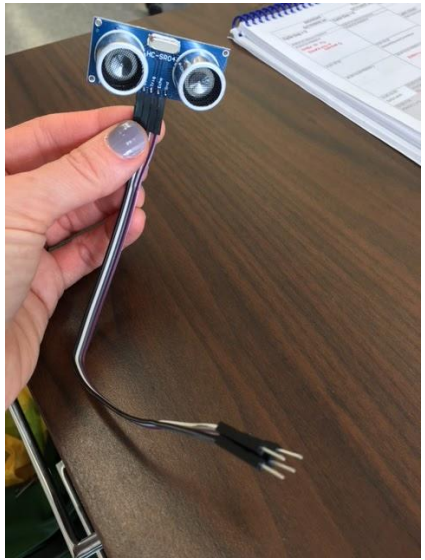
ECHO = echo pin - use another digital pin on the Arduino

Pro tip: color code your wires!

Wiring the Sensor

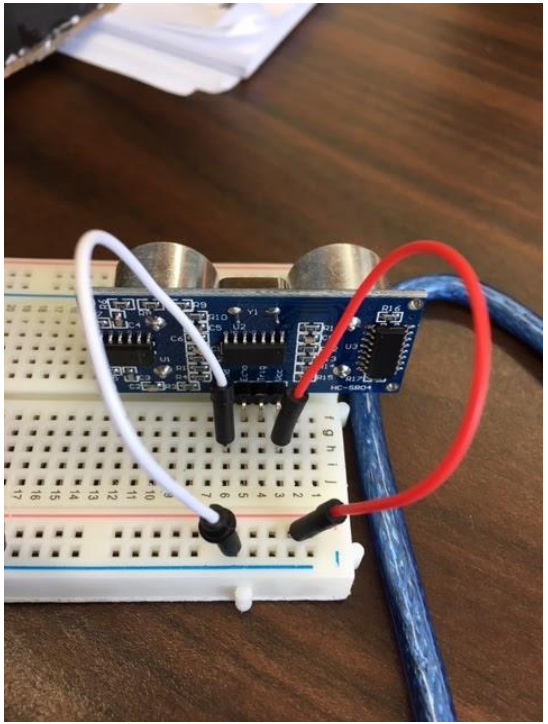


Plug the sensor into four consecutive **rows** in the breadboard



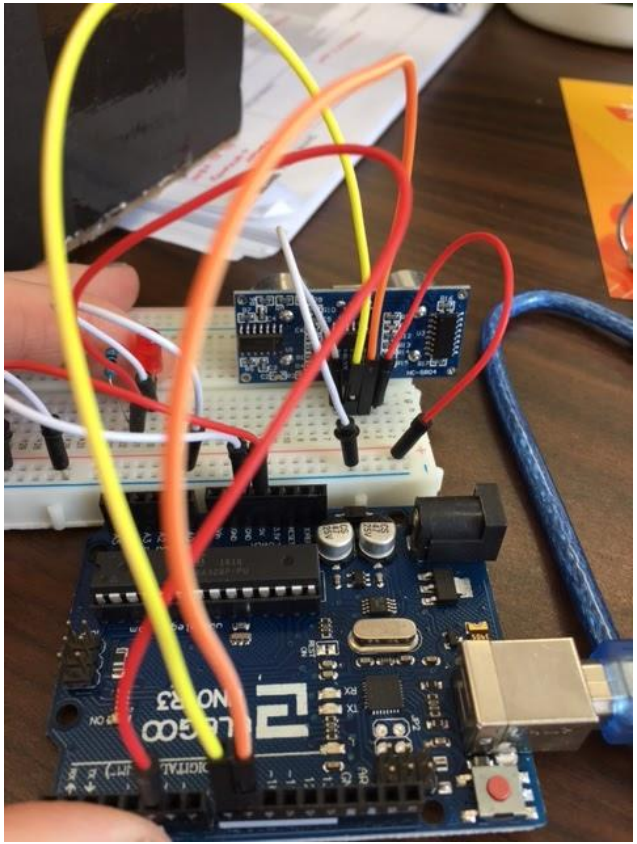
When you wire your pumpkin you will be able to use jumper wires to help with sensor placement

Wire the Sensor



First connect ground and power

Then connect the trigger and echo pins to two digital pins on the Arduino (these are the orange and yellow jumper wires in the picture)

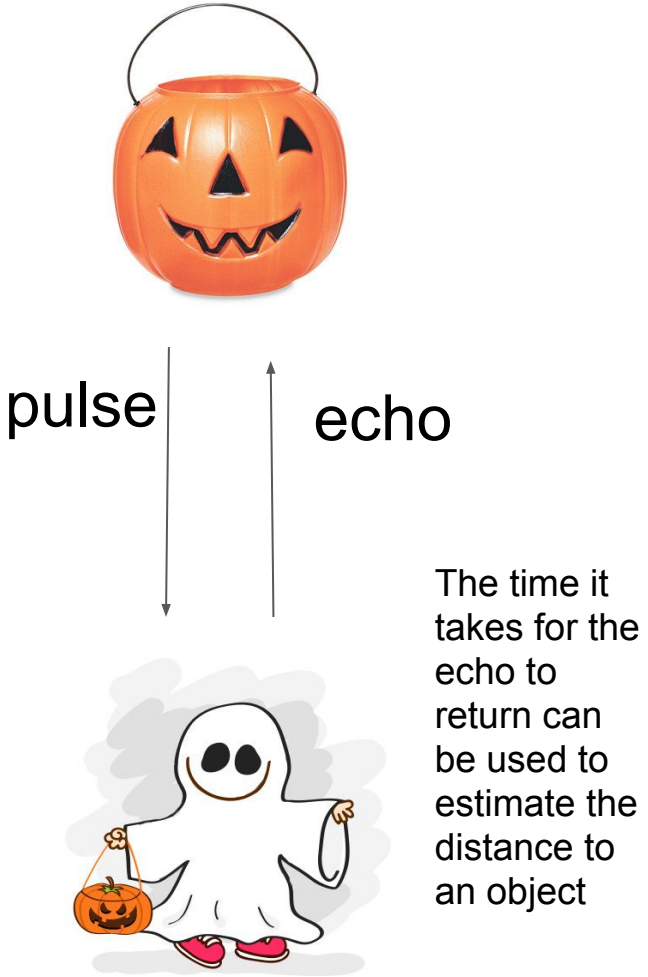


Reading sensor data

Now that we've wired our sensor we want to write the code that will help us read distance data

In Snap! we could import blocks written by other people to use in our code

For this project, we will use a **library** that makes using our distance sensor easier



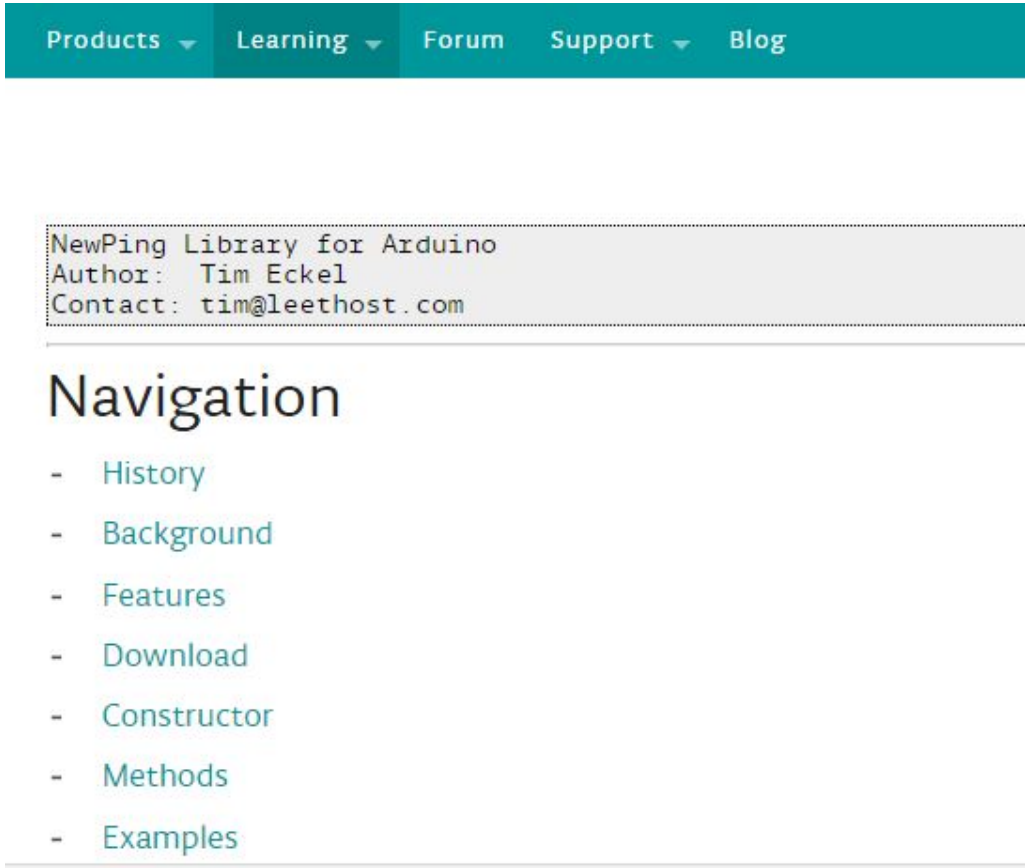
The time it takes for the echo to return can be used to estimate the distance to an object

Sensor Libraries

To find a library we can Google or search on the Arduino Playground site

The NewPing library

Libraries provide an API - a way to access their functions



Methods

- **sonar.ping();** - Send a ping, returns the echo time in microseconds or 0 (zero) if no ping echo within set distance limit
- **sonar.ping_in();** - Send a ping, returns the distance in inches or 0 (zero) if no ping echo within set distance limit
- **sonar.ping_cm();** - Send a ping, returns the distance in centimeters or 0 (zero) if no ping echo within set distance limit
- **sonar.ping_median(iterations);** - Do multiple pings (default=5), discard out of range pings and return median in microseconds
- **sonar.convert_in(echoTime);** - Converts microseconds to distance in inches
- **sonar.convert_cm(echoTime);** - Converts microseconds to distance in centimeters
- **sonar.ping_timer(function);** - Send a ping and call function to test if ping is complete.
- **sonar.check_timer();** - Check if ping has returned within the set distance limit.
- **timer_us(frequency, function);** - Call function every frequency microseconds.
- **timer_ms(frequency, function);** - Call function every frequency milliseconds.
- **timer_stop();** - Stop the timer.

Instead of sending out the pulse, capturing the echo and converting time to distance … we can just use these functions that do all of that for us!

Download the NewPing library

To use the new library we will need to install it.

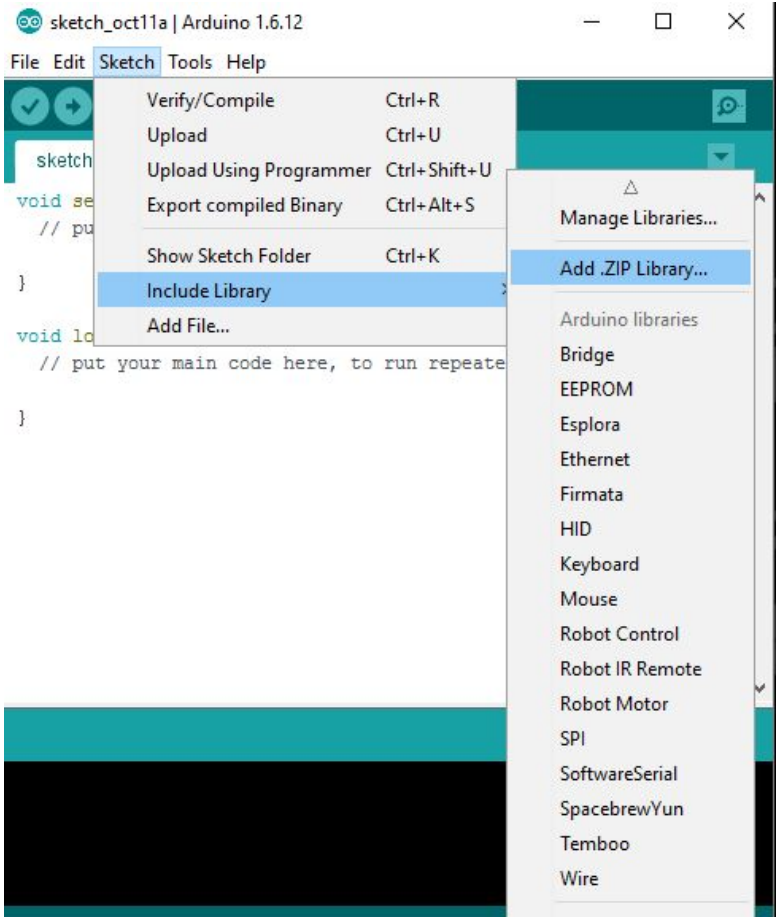
A .zip file will be stored in your downloads folder

You don’t need to unzip it!

Add Library

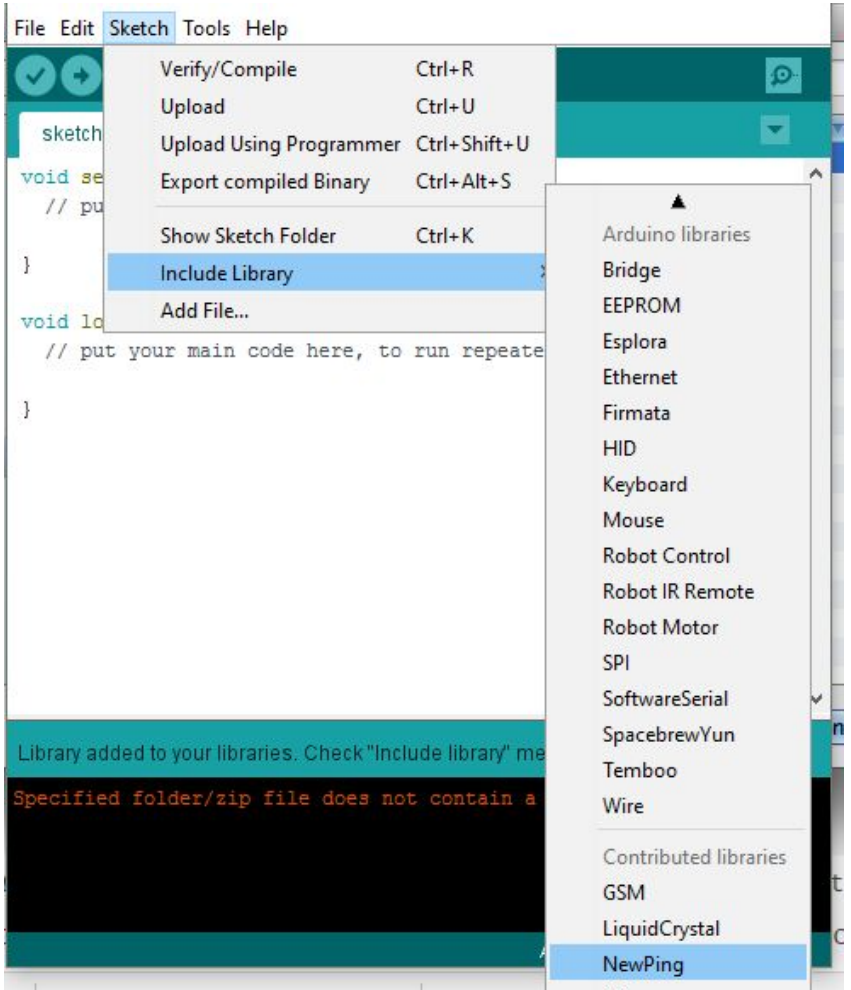
You can Add a .ZIP library to Arduino in a few steps *do not unzip the folder before you do this*

First go to the **sketch** menu, choose **Include Library** and then **Add .ZIP library**



Add Library

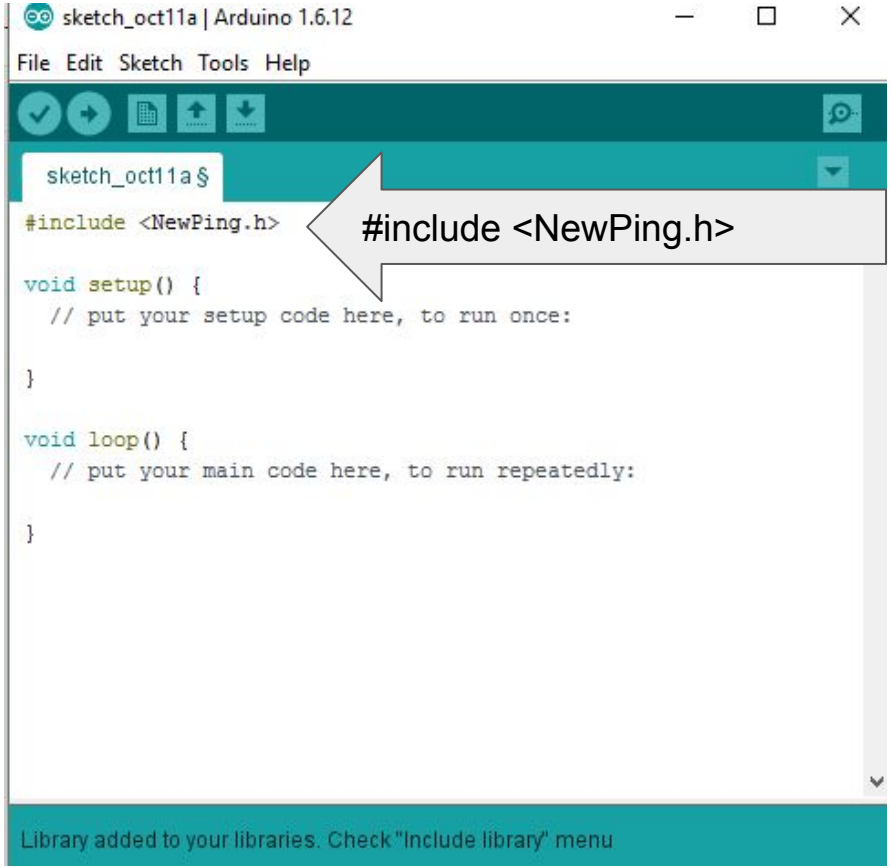
Restart the Arduino IDE and then go to **Sketch -> Include Library** again, now you should see the **NewPing** library as one of the options. Select it.



NewPing

Now NewPing is included in your sketch. The #include statement lets us include library files written by other programmers

This library will make it easier to read the data from the distance sensor

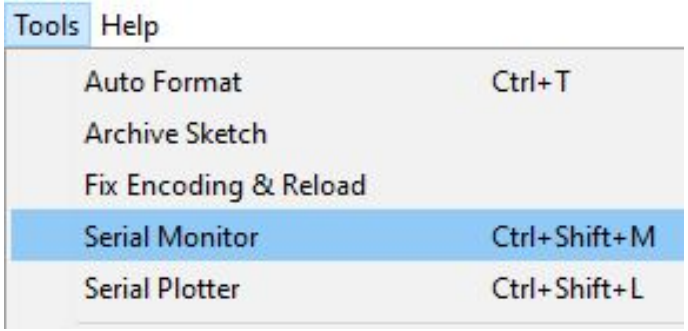


Code

Before we use our sensor in our pumpkin, we will want to make sure it is working.

The first program we write will just print the distance readings from the sensor to the **Serial Monitor**

There's something 10 cm away!

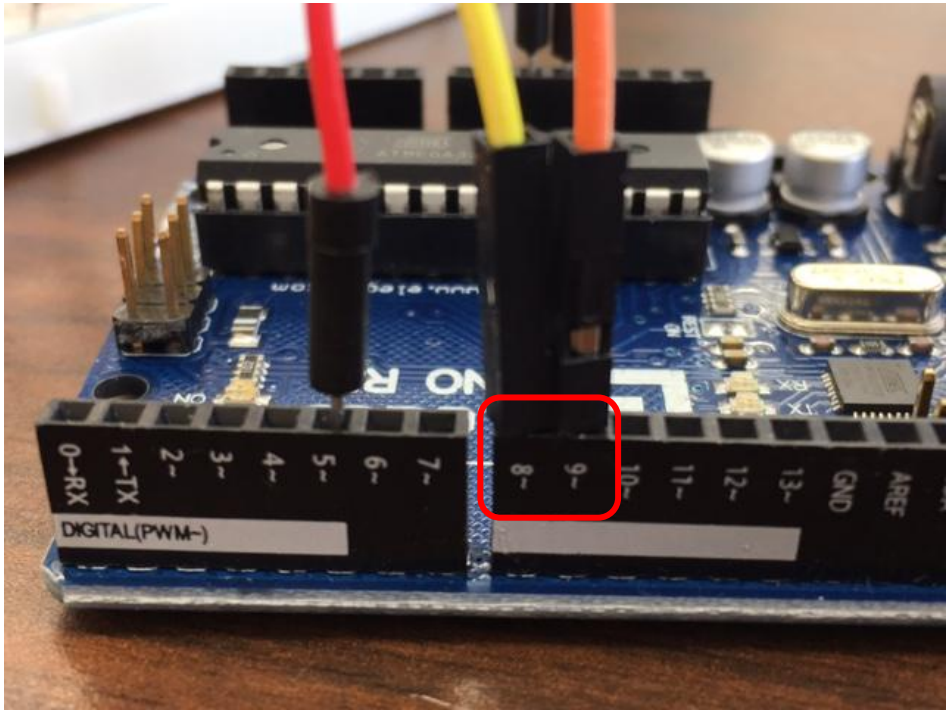


Declare Constants for Sensor Pins

sketch_oct21a §

```
#include <NewPing.h>

#define TRIGGER_PIN 9
#define ECHO_PIN 8


void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedl

}
```

Double check pin numbers



Make a NewPing object

NewPing sonar(TRIGGER_PIN, ECHO_PIN);

This allows us to access the sensor in our code using the **sonar** variable

sketch_oct21a §

```
#include <NewPing.h>

#define TRIGGER_PIN 9
#define ECHO_PIN 8

NewPing sonar(TRIGGER_PIN, ECHO_PIN);
```

setup()

In our setup() function, all we need to do is to set up the baud rate for serial communication (how many bits per second). We will want to do this in any program that uses the serial monitor

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
}
```

loop()

Inside the loop, we will read the sensor every half a second and output the value to the serial monitor

```
void loop() {
  // put your main code here, to run
  int uS = sonar.ping_in();
  Serial.print("Ping: ");
  Serial.print(uS);
  Serial.println(" inches");
  delay(500);

}
```

What does that code do?

```
int uS = sonar.ping_in();
```

`int us:` this part declares a variable (just like in Snap!) the variable is named uS

the int stands for integer the **type** of the variable. In Arduino you must give the type of a variable when you declare it.

`sonar.ping_in()` is a call to the ping_in() function (the one that will use the sensor to give us distance measurement

= in Arduino is called the **assignment operator** it takes the value returned by ping_in() and assigns it to the variable uS

What does that code do?

```
Serial.print(“Ping: ”)

Serial.print(uS)
```

These lines print to the serial monitor

Anything inside double quotes is a **literal** and will be printed to the screen exactly as written

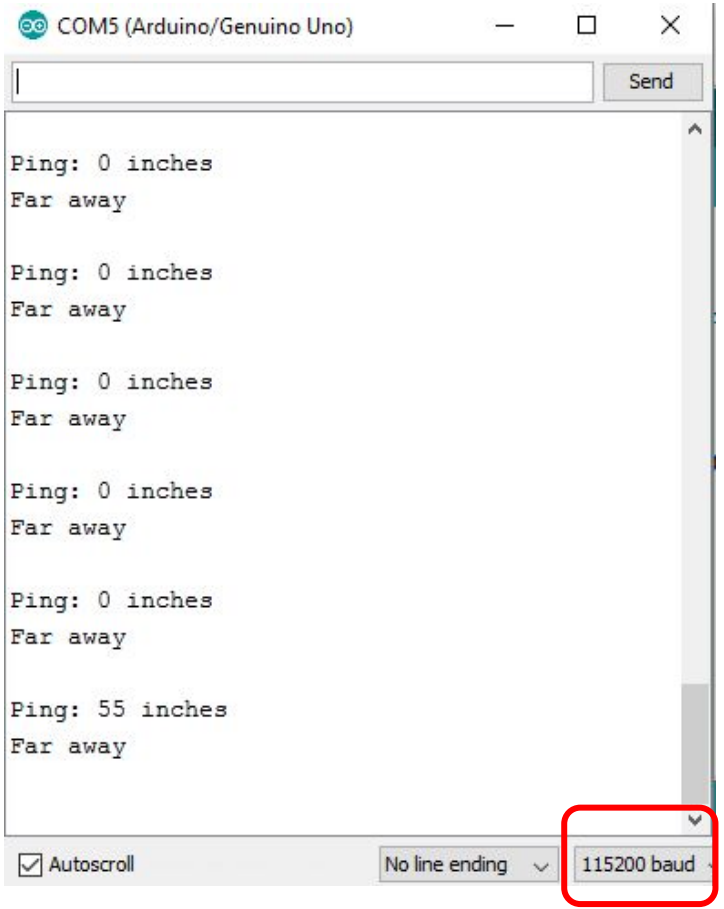
The second line prints the value of the uS variable to the Serial monitor

Uploading the code

Remember the steps to upload your code:

(1) Plug the arduino into the USB port on your laptop
(2) Check that the correct board and port are selected (under the tools menu)
(3) Click the arrow to compile and upload your code

After your code is uploaded, go to the tools menu and open the **Serial Monitor** to see the output from your sensor - is it what you expected? Why or why not?



Code

Now that we can read the distance from the sensor, we want to figure out how to use the raw distance to figure out if something is “near”

How could we do this?

Conditions in Arduino


```
void loop() {
  // put your main code here, to run repeatedly:
  int uS = sonar.ping_in();
  Serial.print("Ping: ");
  Serial.print(uS);
  Serial.println(" inches");
  if(uS > 0 && uS < 5)
  {
    Serial.println("REALLY CLOSE!\n");
  }
  else
  {
    Serial.println("Far away\n");
  }
  delay(500);
}
```

Conditions in Arduino

We can use if to check conditions like in Snap!

The conditions go inside parentheses

We use the same inequality operators

For logical operators

&& -> AND

|| -> OR

```
int uS = sonar.ping_in();
Serial.print("Ping: ");
Serial.print(uS);
Serial.println(" inches");
if(uS > 0 && uS < 5)
{
  Serial.println("REALLY CLOSE!\n");
}
else if(uS >=5 && uS < 10)
{
  Serial.println("Kinda close\n");
}
else
{
  Serial.println("Far away\n");
}
```

Use curly braces instead of blocks

Conditions in Arduino

Remember that everything inside the loop function will repeatedly loop

You will check your condition *when you get to it* and then not again until you come back to that line in the loop

```
void loop() {
  // put your main code here, to run repeatedly:
  int uS = sonar.ping_in();
  Serial.print("Ping: ");
  Serial.print(uS);
  Serial.println(" inches");
  if(uS > 0 && uS < 5)
  {
    Serial.println("REALLY CLOSE!\n");
  }
  else
  {
    Serial.println("Far away\n");
  }
  delay(500);
}
```

STEP 8: Put it all together

We have code that blinks an LED at a regular interval and code that reads the sonar sensor

How can you use the input from the sonar sensor to change how the LED blinks?

Work on this with your partner

STEP 9: Plan your Pumpkin!

Use the page provided in your lab report document to plan your pumpkin design. Be sure to leave room for your LEDs and your sensor.

You will have two days to work in class and then we will have our trick-or-treat gallery walk