LEARNING RESOURCE
CODING

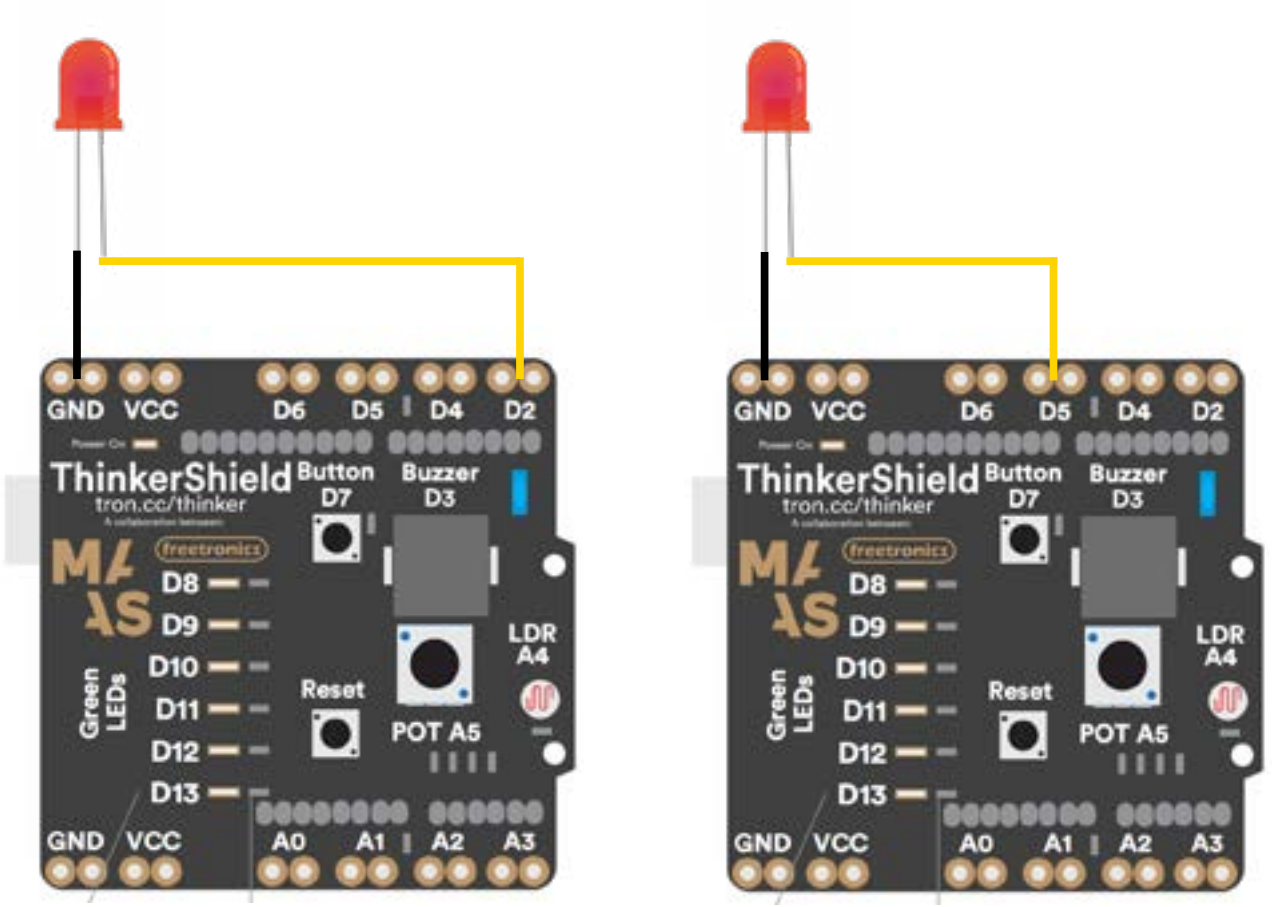
Looking to extend your ThinkerShield project with extra components not included in the kit? This handy guide will show you how to connect all the most common components in minutes and the code required to make it work. These components are available at most electronics retailers.

POWERHOUSE MUSEUM

CONNECTING LEDS

Add some light to your project with an LED (Light Emitting Diode). Who doesn't like having lights in their project?
Extra component required: LED (Light Emitting Diode)

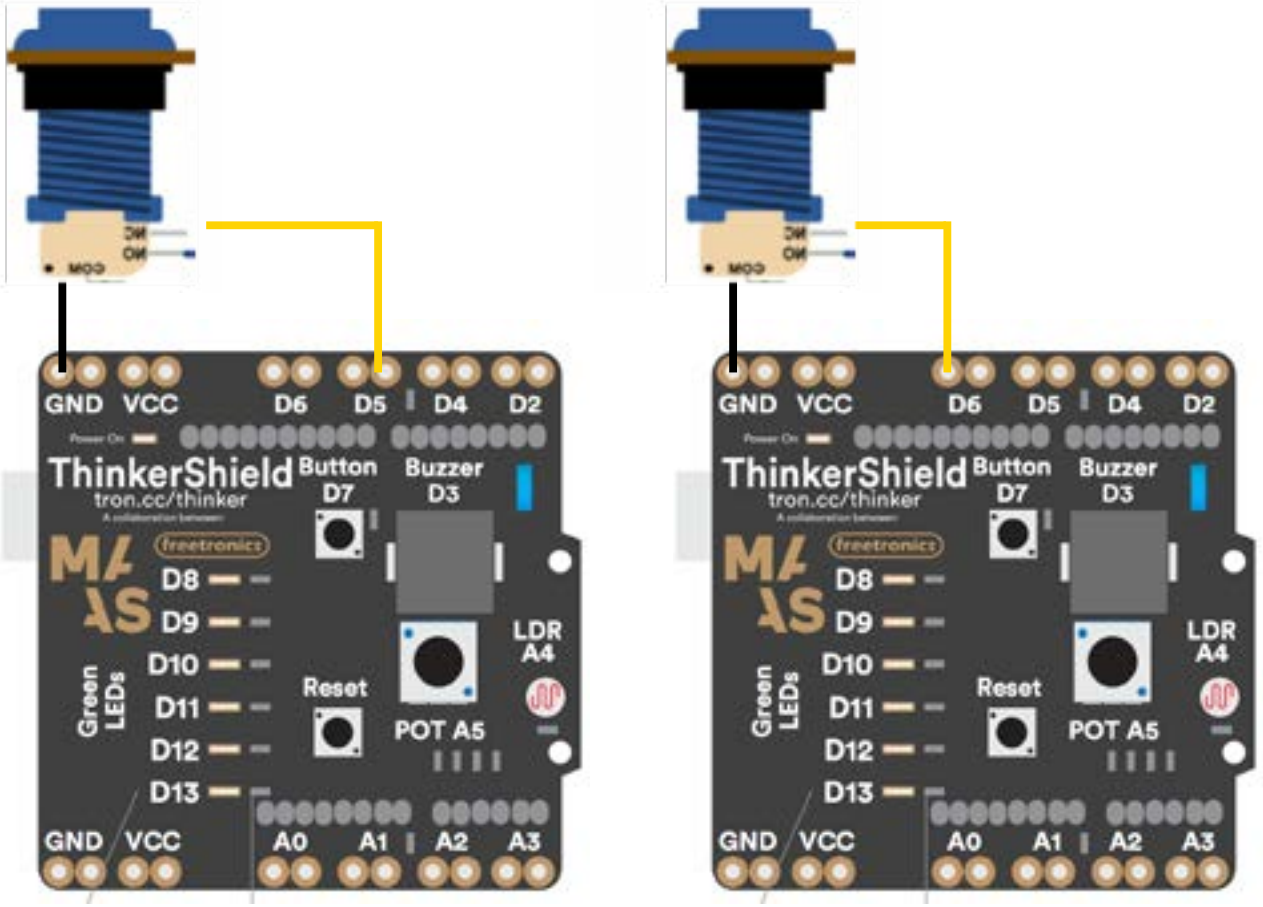
```
// declare your LED
int led = 2;

void setup() {
  // set your LED as an outpu
  pinMode (led, OUTPUT);
}

void loop() {
  // turn your LED on
  digitalWrite (led, HIGH);

  // wait 1 second
  delay(1000);

  // turn your LED off
  digitalWrite (led, LOW);

  // wait 1 second
  delay(1000);

}
```



CONNECTING ARCADE BUTTONS

Give your users a way to interact with your project with an arcade button. This is great for making games.
Tip: You can use the other off pin to reverse the way the button works. Give it a go!
Extra component required: Arcade button

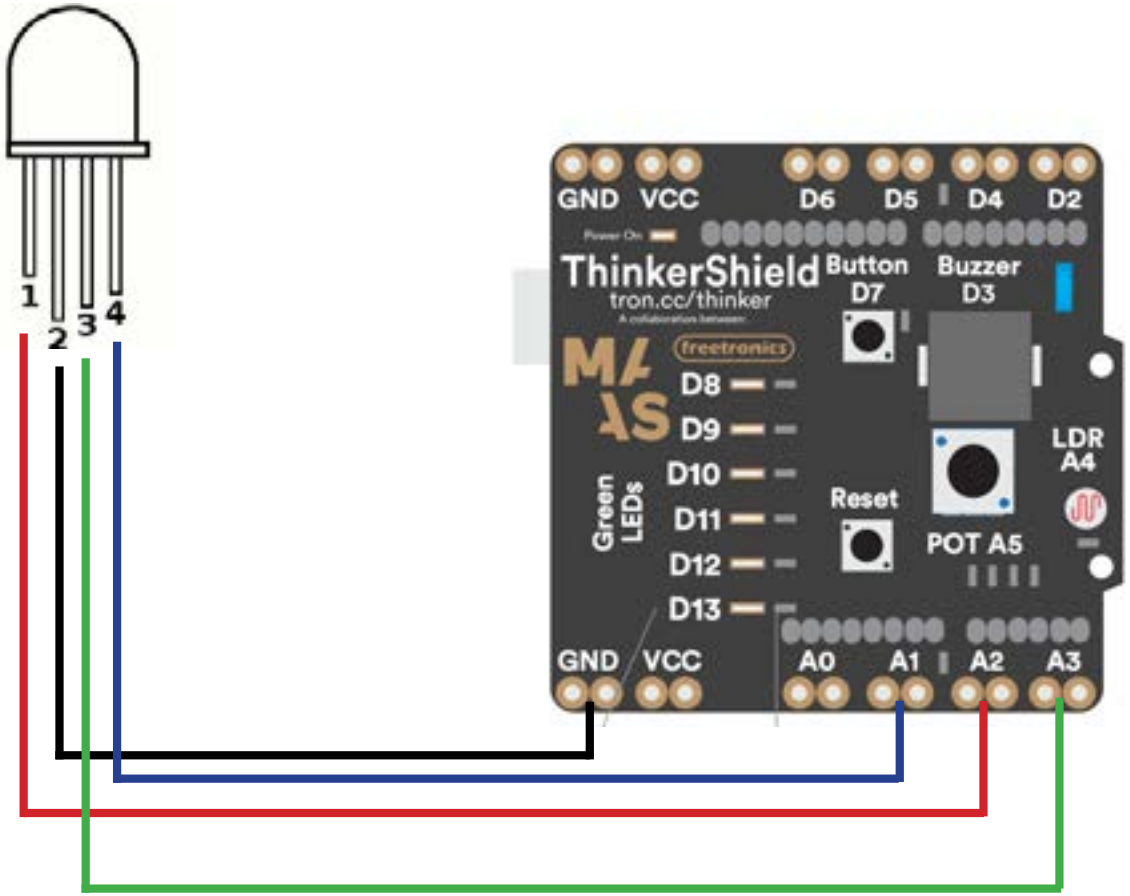
```
// declare the variables
int led = 2;
int button = 5;

void setup() {
  // set your led as an output and button as input_pullup
  pinMode (led, OUTPUT);
  pinMode (button, INPUT_PULLUP);
}

void loop() {

  // check if your button has been pressed
  if (digitalRead(button) == HIGH)
  {
  // turn the LED on
  }

  else
  {
  // turn the LED off

  }

}
```



CONNECTING RGB LEDS

An RGB (Red/Green/Blue) can be used to add any colour and even changing colours to your project.
The model in example is a “Common Cathode RGB LED” and other RGB LEDs might be wired differently.
Extra component required: Common Cathode RGB LED

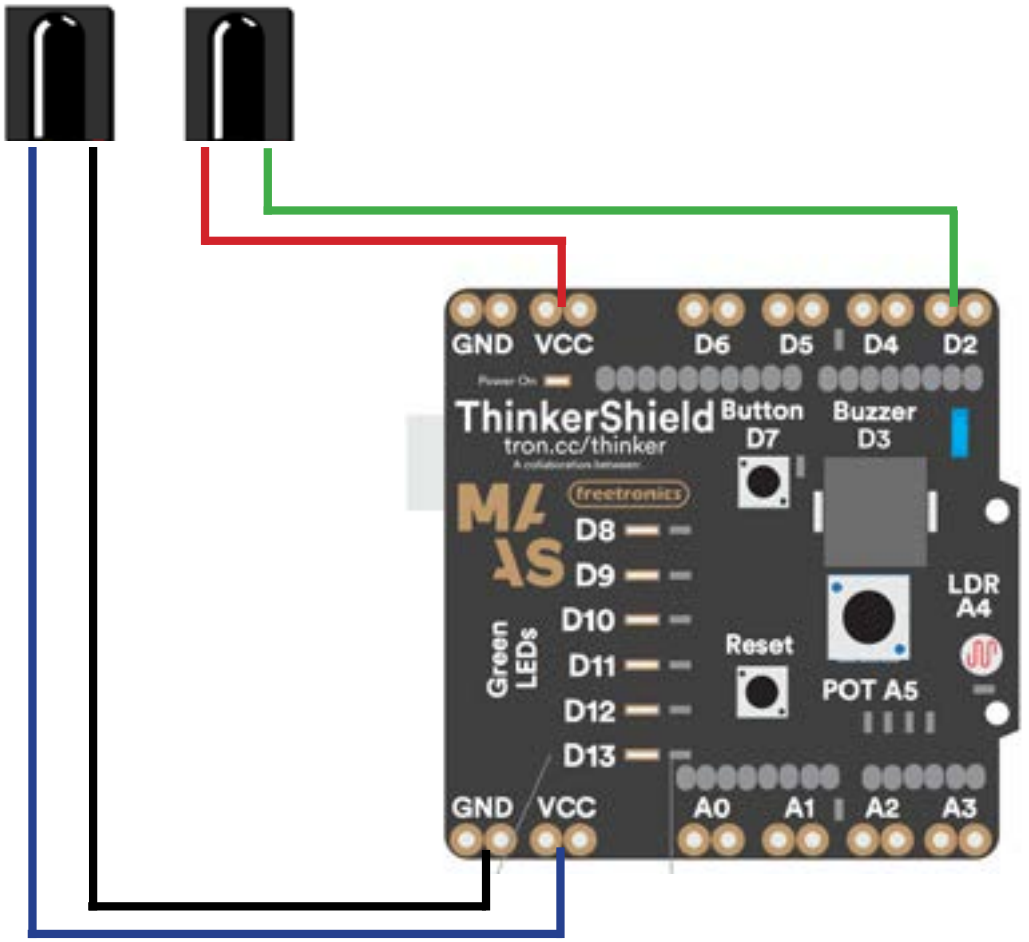
```
// declare the pins
int red = A2;
int green = A3;
int blue = A1;

void setup() {
 // set each of the pins as outputs
 pinMode (red, OUTPUT);
 pinMode (green, OUTPUT);
 pinMode (blue, OUTPUT);
}

void loop() {
 // set the colour of the LED by making
 //combinations of the primary colours
 //purple
 digitalWrite(red, HIGH);
 digitalWrite(green, LOW);
 digitalWrite(blue, HIGH);
}
```



CONNECTING BREAK BEAM SENSORS

A break beam sensor can determine if an object passes through the beam.
This is great for adding inputs that work without the user needing to touch.
Extra component required: Break beam sensor

```
// declare sensor and LED variables
int sensor = 2;
int led = 8;
void setup() {
  // setup the sensor as an input
  pinMode (sensor, INPUT);
  // write HIGH to the sensor to get it started
  digitalWrite (sensor, HIGH);
  //setup the led as an output)
  pinMode (led, OUTPUT);
}
void loop() {
  // check if the beam has been broken (read LOW)
  if (digitalRead (sensor) == LOW)
  {
    // turn on the LED
    digitalWrite (led, HIGH);
  }
  // else if the beam is unbroken (read HIGH)
  else
  {
    // turn off the LED
    digitalWrite (led, LOW);
  }
}
```

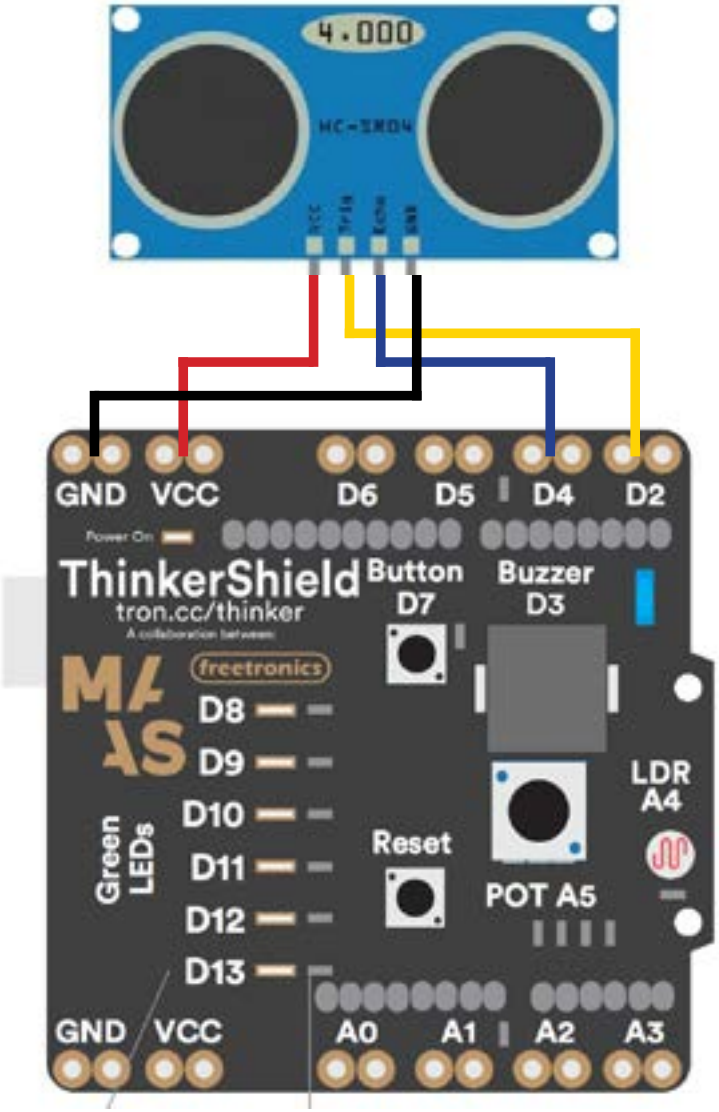


CONNECTING SONAR DISTANCE SENSORS

A distance sensor can determine how far away an object is from the sensor using sonar. To use this you will also need to go to the library menu in Arduino and add the “newping” library. This great for giving users warnings if they get too close to objects.

Extra component required: Sonar distance sensor

```
// include the sonar library
#include <NewPing.h>

// declare the sonar sensor pins
int trigger = 2;
int echo = 4;
// then declare the sonar sensor
NewPing sonar(trigger, echo);

void setup()
{

}

void loop()
{
    // get the distance in centimetres
    int distance = sonar.ping_cm();
}
```

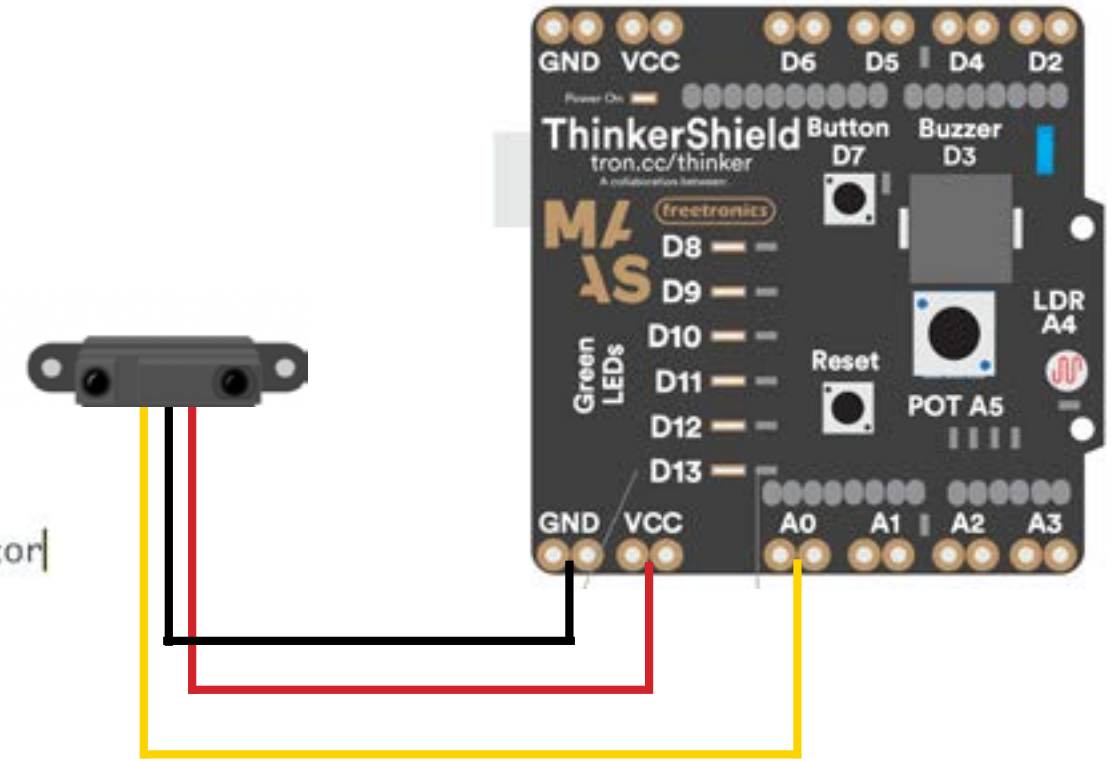


CONNECTING INFARED DISTANCE SENSORS

A distance sensor can determine how far away an object is from the sensor using sonar. To use this you will also need to go to the library menu in Arduino and add the “newping” library. This great for giving users warnings if they get too close to objects.

Extra component required: Infrared distance sensor

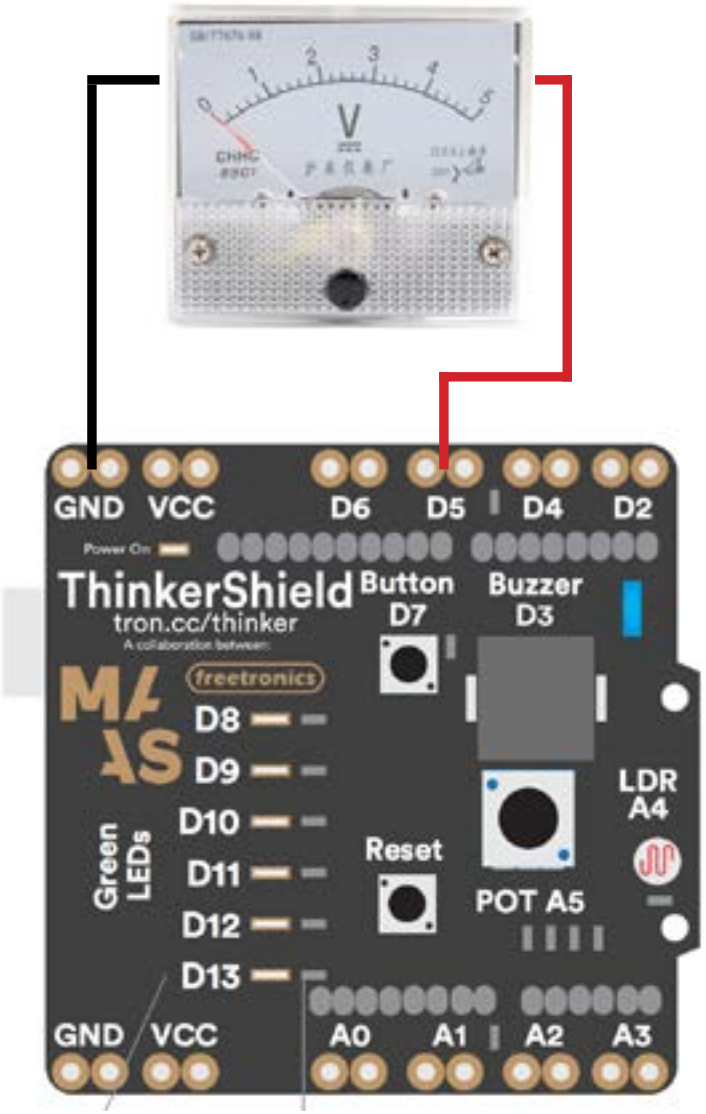
```
// declare the sensor variable
int sensor = A0;

void setup() {
  // begin a serial connection
  Serial.begin (9600);

  // set the sensor pin as an input
  pinMode (sensor, INPUT);
}

void loop() {
  // store the reading from the sensor
  // into the variable distance
  int distance = analogRead (sensor);

  // print the distance to the serial monitor
  Serial.println (distance);
}
```



CONNECTING A VOLTMETER

A voltmeter measures the voltage from an output but is also great to create dials and meters.
Make sure you get a voltmeter that measures from zero volts to five volts for this to work best with ThinkerShield.
Extra component required: 5V voltmeter

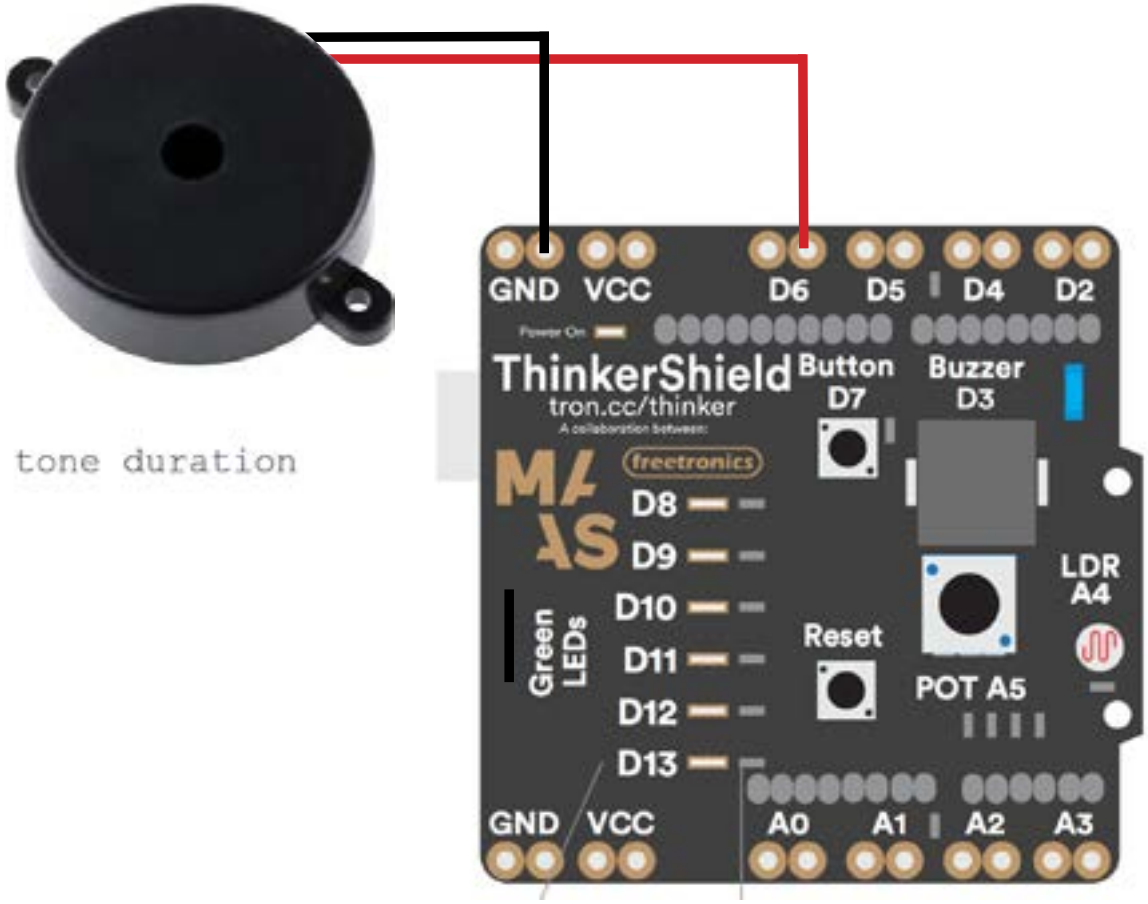
```
// declare the voltmeter pin
int voltmeter = 5;

void setup()
{
    // set the voltmeter as an output
    pinMode(voltmeter, OUTPUT);
}

void loop()
{
    // any number between 0 and 255
    int amount = 255;
    // set the voltage on the voltmeter pin
    analogWrite(voltmeter, amount);
}
```



CONNECTING PIEZO BUZZERS

These buzzers are great for adding electronic sounds to your project.
For more detail on to make specific notes with a buzzer have a look at the ThinkerShield Get on with it guide.
Extra component required: Piezo buzzer

```
//declare the buzzer
int buzzer = 6;

void setup() {
    //setup the buzzer as an output
    pinMode (buzzer, OUTPUT);
}

void loop() {
    //play a tone on the buzzer
    //262 = Middle C
    //100 = Duration
    tone(buzzer, 262, 100);
    //Delay should be longer than the tone duration
    delay(200);
}
```



CONNECTING SERVO MOTOR

Looking to add some motion to your project, a servo motor is perfect for you. Remember you can only directly connect 5 volt servo motors. If you have a more powerful one be sure to read the manufacturer specificiations.

Extra component required: 5V servo motor

```
// include the servo library
#include <Servo.h>

// declare the servo
Servo servo;

void setup()
{
    // attach the servo to pin A0
    servo.attach(A0);
}

void loop()
{
    // set the speed of the servo
    //   0 = maximum counter-clockwise
    //  90 = stopped
    // 180 = maximum clockwise
    servo.write(180);
}
```