IEEE Arduino AnalogRead Workshop

Introduction

analogRead() is an function that allows you to read information from an analog input. These pins read in a voltage between 0 and 5V, then converts it to a number between 0-1023. Refer to Table 1 for a conversion chart.

Input Voltage (V)	Stored Number
5	1023
4	818
3	613
2	409
1	204

Table 1: Conversions from voltage to binary numbers

On the Arduino UNO boards there are 6 analog input pins (A0-A5) that can be found in the bottom left of figure 1.

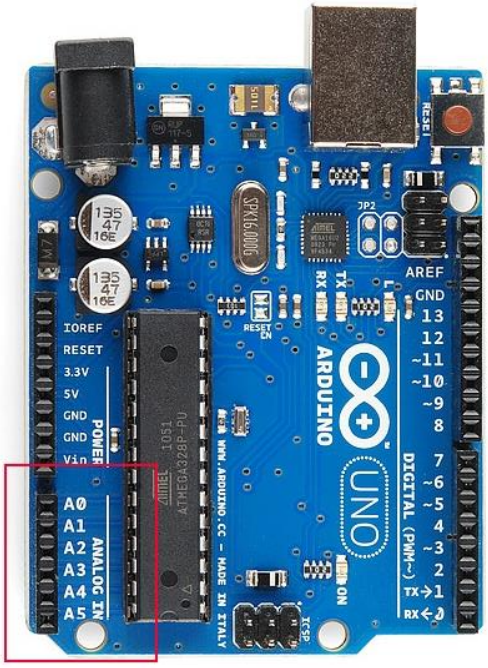

Figure 1: Arduino UNO Board

Another component we will be using is a potentiometer. This is a variable resistor that uses a wiper on a resistive track that alters the resistance as the wiper goes around. An example of this can be seen in Figure 2.

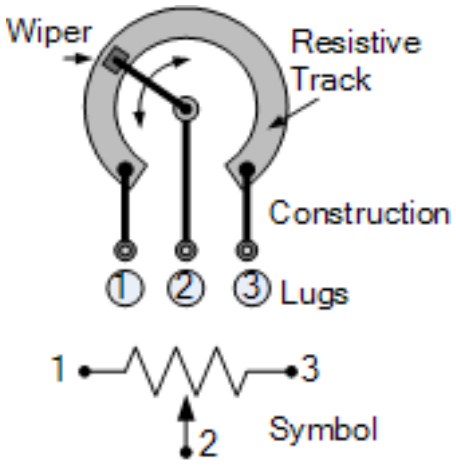

Figure 2: Potentiometer Outline

Part 1: Change how fast LED flashed with potentiometer

First, hook up a potentiometer and change the delay on a blinking LED. Then, hook up the potentiometer and attach the “output” pin (center pin) of that to the input pin of the analog input. This analog input will give a number from 0-1023, depending on the output voltage. After that, use this value to set the delay of the blinking LED. This is the same code we used when we did the blinking LED in the first workshop just adding in the potentiometer.

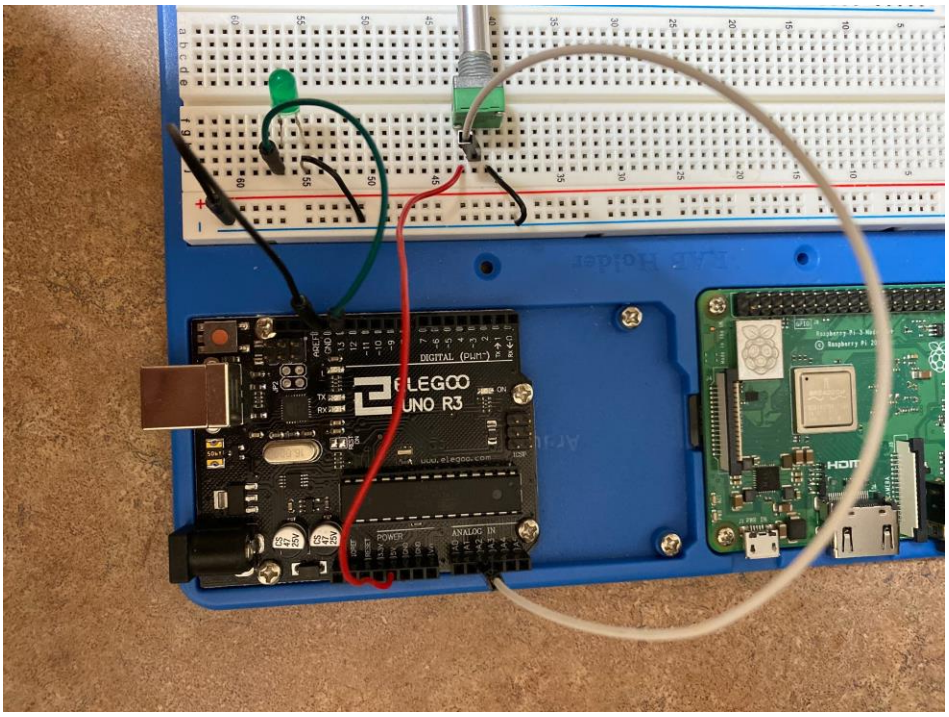

Figure 3: Hardware set for part 1. Connect the positive end of the LED to pin 13 for the LED and other end to GND (Do not forget LEDs has polarity!). Then connect the outside pins on the potentiometer to 5V and GND, then the center pin (output pin) to A2. You can combine the grounds to the side nodes that run the length of the breadboard. Refer to Figure 11 to see how breadboard work.

```
int potPin = 2;      // input pin for potentiometer
int ledPin = 13;     // output pin for LED
int delayTime;       // Variable to set for delay time

void setup() {
  pinMode(ledPin, OUTPUT);  // declare LED as an output
}


void loop() {
  delayTime = analogRead(potPin);    // Read voltage value from potentiometer
  digitalWrite(ledPin, HIGH);  // Turn LED on
  delay(delayTime);                  // delay for the amount of time in which the pot voltage is
  digitalWrite(ledPin, LOW);   // turn the ledPin off
  delay(delayTime);                  // delay same amount as before
}
```

Figure 4: Arduino code for part 1

Part 2: Using photoresistor to turn LED on and off

Next, we will be using a photoresistor to turn an LED on and off. A photoresistor is a sensor that responds to the intensity of light and has a certain resistance based on that. The more light it has, the lower resistance it has. This is like a potentiometer, just with light. The circuit is setup so the LED will be on for the lighting in the room, but if you add more light, like a flashlight, it will turn off. So, after you get this setup try using a flashlight to turn the LED off.

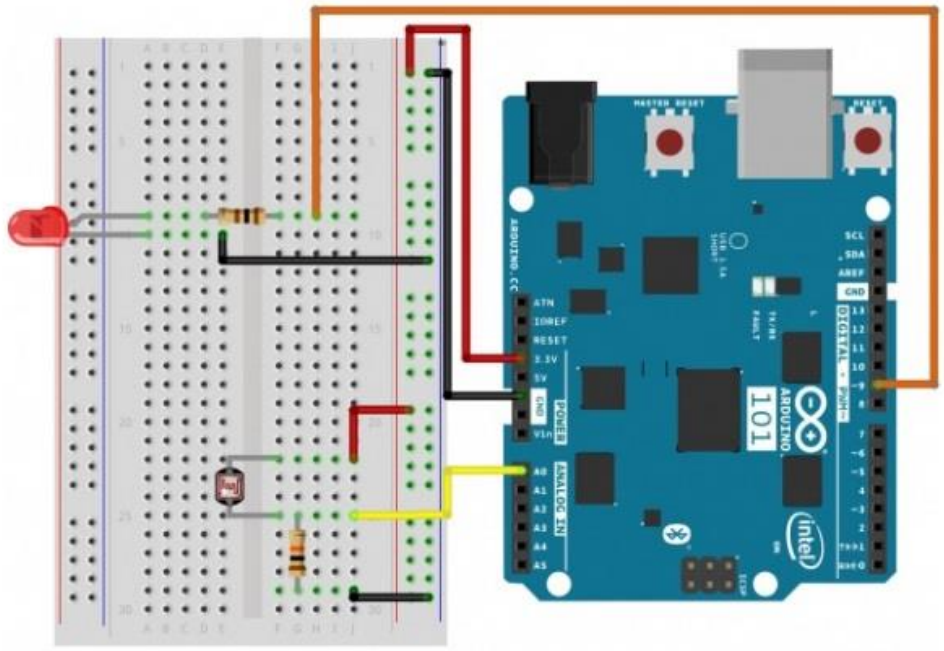

Figure 5: Hardware setup for Part 2

```
int sensorPin = 0;
int ledPin = 9;
int measuredInput;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {

  //Set the input of the voltage over the resistor
  measuredInput = analogRead(sensorPin);

  //If the voltage is lower than 200 (~1V), turn the LED on (no light)
  if (measuredInput < 200)
  {
    digitalWrite(9, HIGH);
  }

  //If the voltage is lower than 200 (~1V), turn the LED off (lit up)
  else
  {
    digitalWrite(9, LOW);
  }

}
```

Figure 6: Arduino code for Part 2

Part 3: Using temp sensor to make thermometer

In this circuit, we will be using a temperature sensor. This sensor has 3 pins: power, GND and signal. The output signal is a ratio of the input voltage, which is dependent on the temperature of the sensor. We will then convert the output voltage into temperature and print it to the serial monitor. This can be opened by pressing CTRL+SHIFT+M or by looking in the "Tools" drop down menu at the top right of the Arduino IDE, which can be seen in Figure 8 circled in red.

After you have set this up, squeeze the temperature sensor to heat it up. What happens?

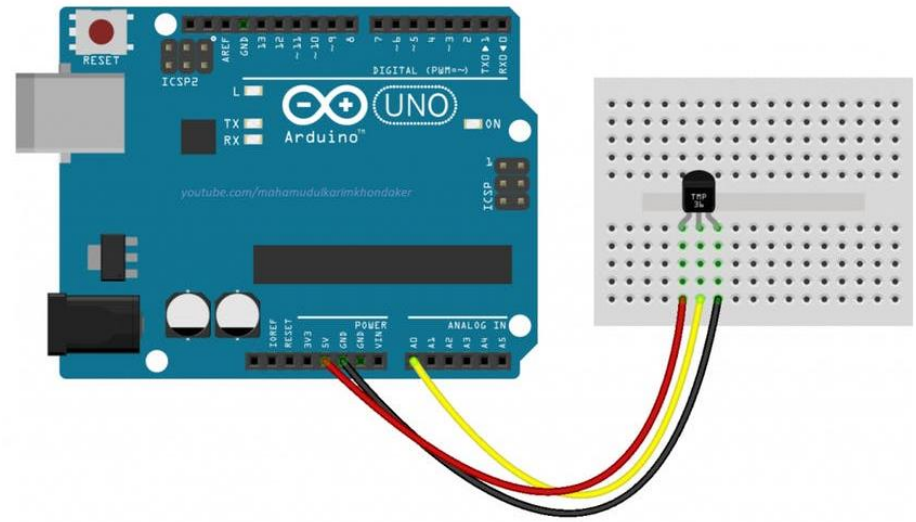

Figure 7: Hardware setup for Part 3

Figure 8: Arduino code for Part 3 and where to open serial monitor window

Part 4: Using ultrasonic sensor to measure distance

An ultrasonic sensor is a sensor that sends and received ultrasonic waves to determine the distance it is from another object. The range of the ultrasonic sensor is between 2 cm and 4 m. We will not be using the analogRead() command for this part of the workshop. Some sensors use a digital input such as pins 0-13, like we normally use for our digital outputs. When you get this set up, open the serial monitor to see what distance is detected.

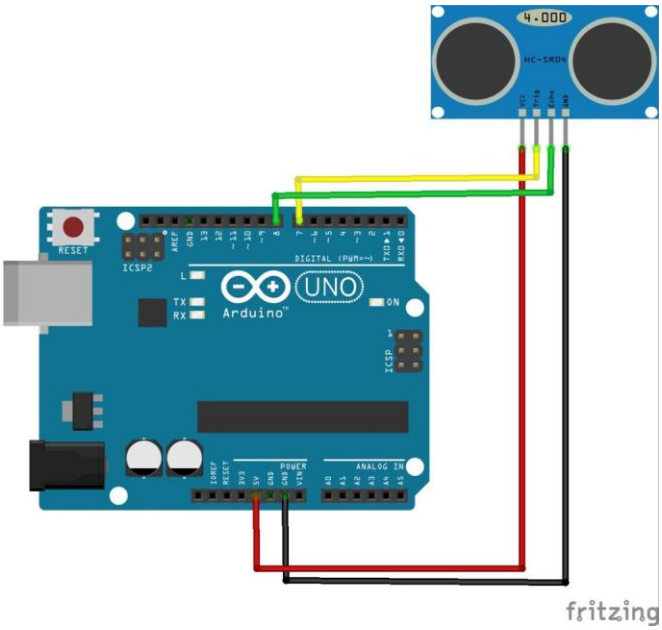

Figure 9: Hardware setup for Part 4

```
int trigPin = 7;                    //Set each pin we will be using
int echoPin = 8;
int duration;                       //Initialize all variables needed
int distance;

void setup() {
pinMode(trigPin, OUTPUT);            // Sets the trigPin as an Output
pinMode(echoPin, INPUT);             // Sets the echoPin as an Input
Serial.begin(9600);                  // Starts the serial communication
}

void loop() {

digitalWrite(trigPin, LOW);          //Set low to start to ensure it
delayMicroseconds(2);                //signal only at proper time

digitalWrite(trigPin, HIGH);         //This will sent out the ultrasonic signal
delayMicroseconds(10);               //for 10 microseconds, long enough for it to
digitalWrite(trigPin, LOW);          //reach on object and come back for detection

duration = pulseIn(echoPin, HIGH);   //read the input signal and store how long
                                     //it took for the signal to return

distance= duration*0.034/2;          //Convert time into distace
Serial.print("Distance: ");          //Print the distance
Serial.println(distance);

delay(500);                          //Only run every half second
}
```

Figure 10: Arduino code for Part 4

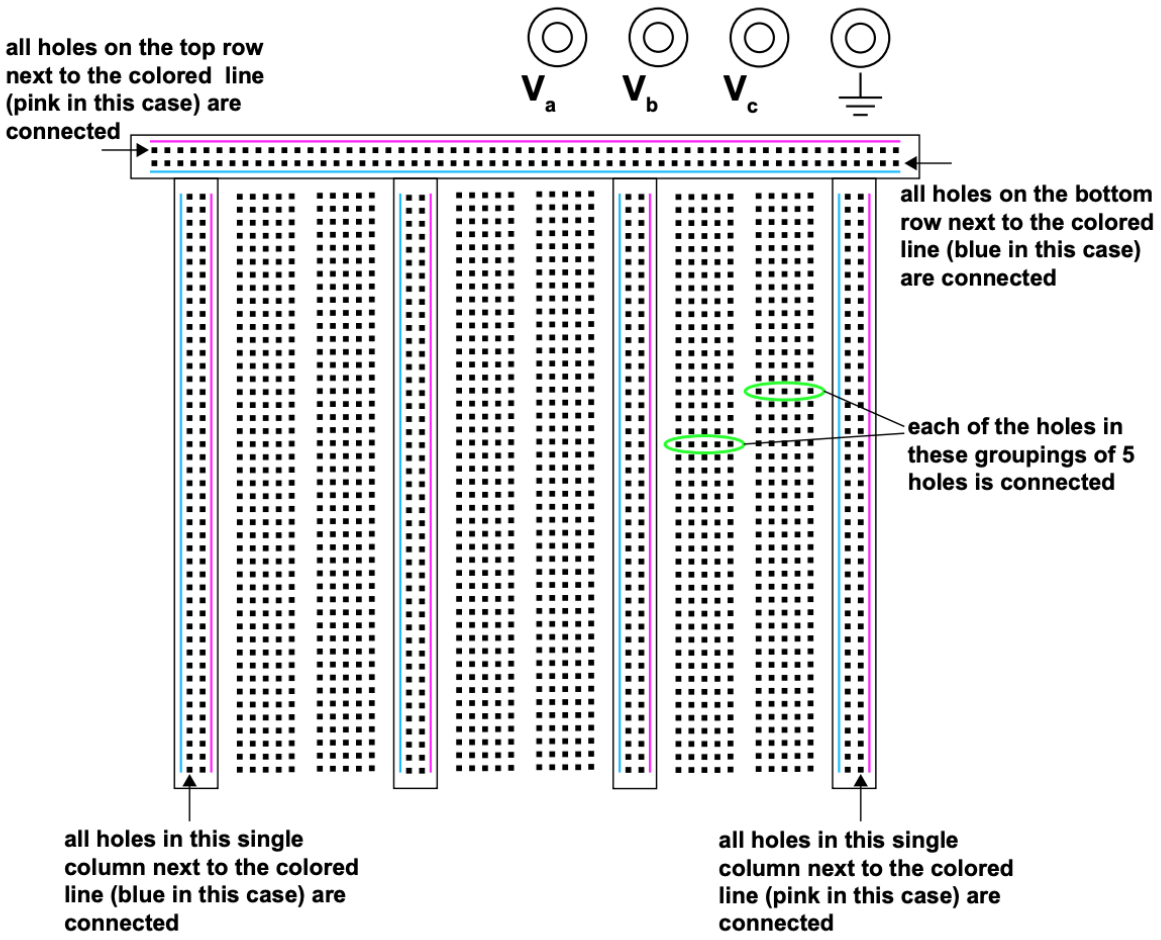

Figure 11: How breadboards work