Fourth Lecture

Analog Input & Output

By:

Ayub Othman Abdulrahman

University Of Halabja

Introduction

- The **Arduino** can **input and output analog** signals as well as digital signals. An **analog** signal is one that can take on any number of values, unlike a digital signal which has only two values: HIGH and LOW.

Analog Input

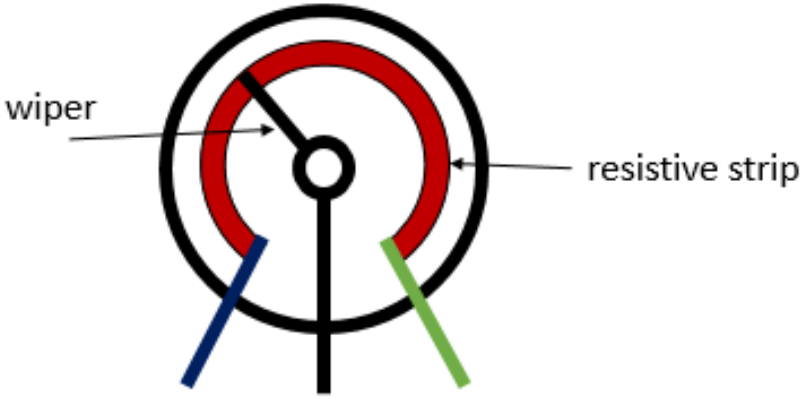
- Variable resistor (a potentiometer) is used as analog input, which is read its value using one analog input of an Arduino or Genuino board. The resistor's analog value is read as a voltage because this is how the analog inputs work.

Potentiometer

- A potentiometer, may come in a wide variety of shapes and are used in many applications in your daily life, for example to control the audio volume of the radio.

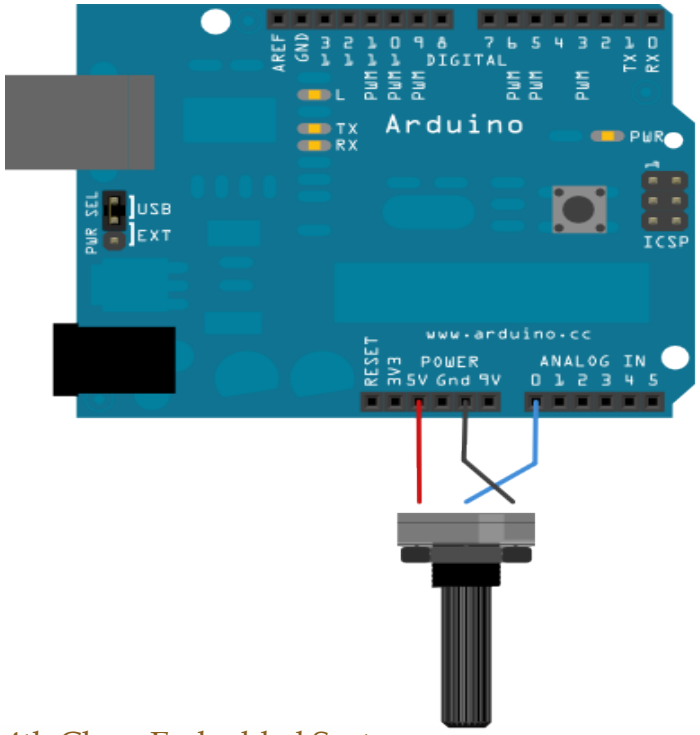
Potentiometer

- A potentiometer is a manually adjustable variable resistor with three terminals.
- Two terminals are connected to a resistive element and the third terminal is connected to an adjustable wiper.



Potentiometer (Circuit Diagram)

- The following circuit diagram is sample diagram potentiometer



Potentiometer (Code Example)

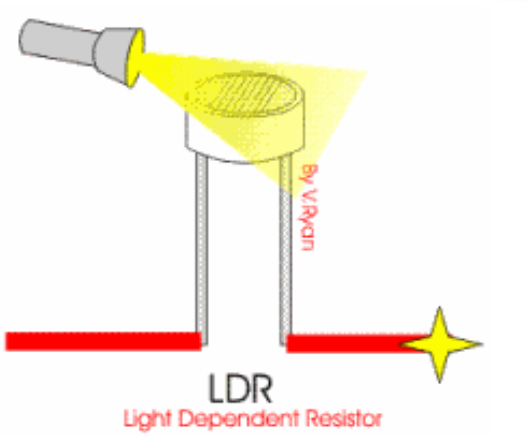
```
1.  int sensorPin = A0;    // select the input pin for the potentiometer
2.  int sensorValue = 0;  // variable to store the value coming from the sensor
3.  void setup() {
4.  Serial.begin(9600);
5.  }

6.  void loop() {
7.   // read the value from the sensor:
8.    sensorValue = analogRead(sensorPin);
9.   Serial.println(sensorValue);
10. delay(200);

11.  }
```

Photoresistor (LDR)

- In order to detect the intensity of light or darkness, a sensor is used that called an LDR (Light Dependent Resistor). The LDR is a special type of resistor that allows higher voltages to pass through it (low resistance) whenever there is a high intensity of light, and passes a low voltage (high resistance) whenever it is dark.

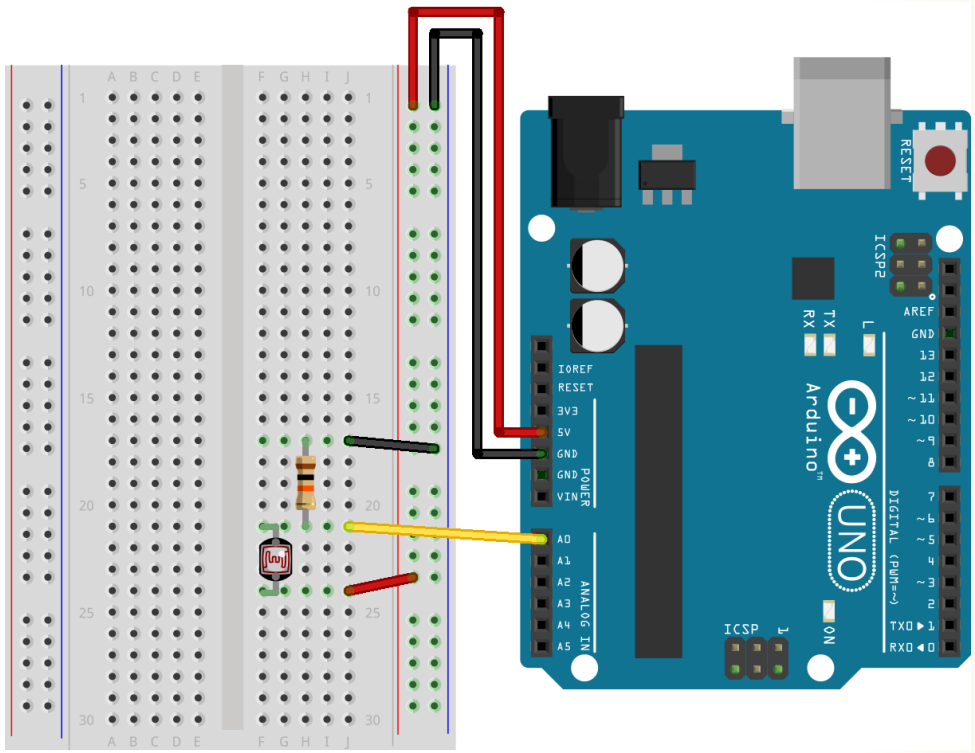
How Does LDR Work?

- This system works by sensing the intensity of light in its environment. The sensor that can be used to detect light is an LDR. The LDR gives out an analog voltage when connected to VCC (5V)



Photoresistor (Circuit Diagram)

- The following circuit diagram is sample diagram Photoresistor using 10K ohm photoresistor and 10K ohm resistor



Photoresistor (Code Example)

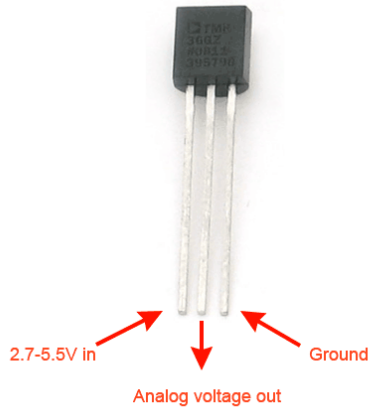
```
1.  int sensorPin = A0;    // select the input pin for the potentiometer
2.  int sensorValue = 0;  // variable to store the value coming from the sensor
3.  void setup() {
4.  Serial.begin(9600);
5.  }

6.  void loop() {
7.   // read the value from the sensor:
8.    sensorValue = analogRead(sensorPin);
9.   Serial.println(sensorValue);
10. delay(200);

11.  }
```

LM35 Temperature Sensor (Thermometer)

- LM35 is a precision temperature sensor with its output proportional to the temperature (in ºC). With LM35, temperature can be measured more accurately than with a thermistor. It also possess low self heating and does not cause more than 0.1 ºC temperature rise in still air.

LM35 Temperature Sensor (Thermometer)

- LM35 is an integrated analog temperature sensor whose electrical output is proportional to Degree Centigrade. LM35 Sensor does not require any external calibration or trimming to provide typical accuracies.

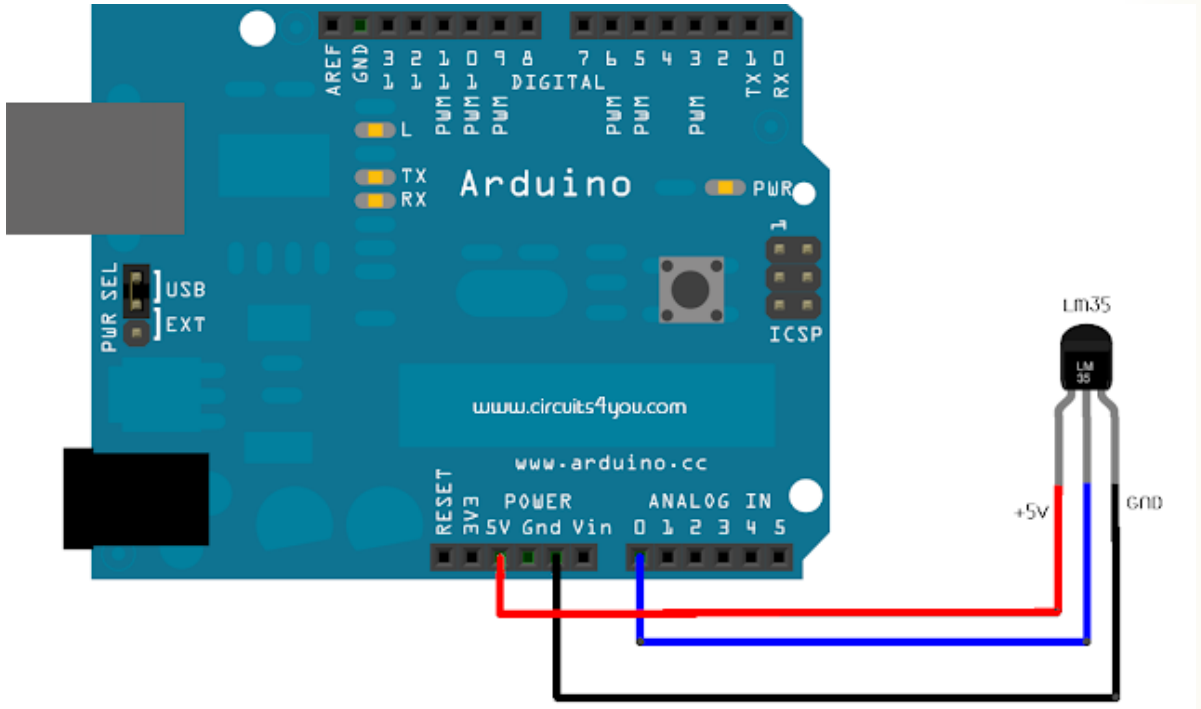


How Does LM35 Sensor Work?

- Main advantage of LM35 is that it is linear i.e. 10mv/°C which means for every degree rise in temperature the output of LM35 will rise by 10mv. So if the output of LM35 is 220mv/ 0.22V the temperature will be 22°C. So if room temperature is 32°C then the output of LM35 will be 320mv i.e. 0.32V.

LM35 Sensor (Circuit Diagram)

- The following circuit diagram is sample diagram LM35 Sensor



Thermometer (Code Example)

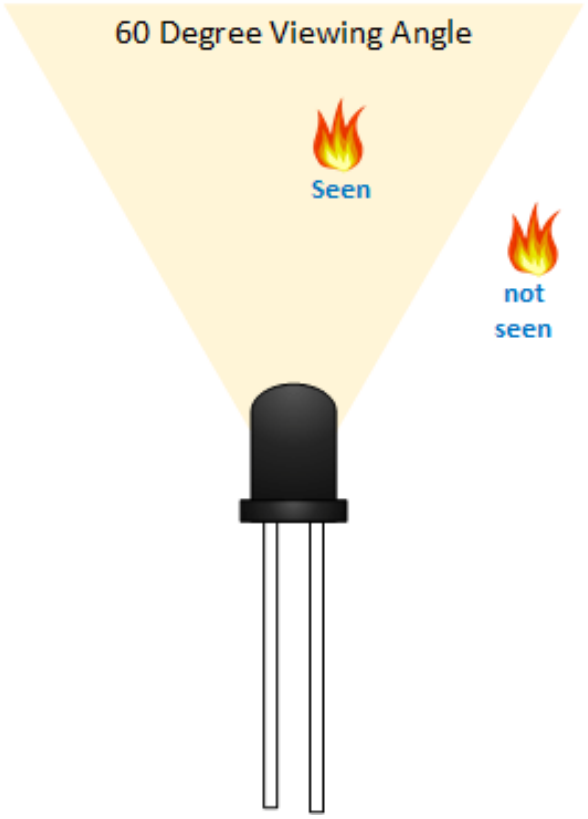
```
1.  int sensorPin = A0;    // select the input pin for the potentiometer
2.  int sensorValue = 0;  // variable to store the value coming from the sensor
3.  void setup() {
4.  Serial.begin(9600);
5.  }
6.  void loop() {
7.  int readValue = analogRead(sensorPin);  //To read analog pin (Thermistor)
8.  // calibration analod read to get temperature value
9.  float temperature = (readValue * 0.0049)*100; // Temperature in C
10. float temperatureF = (temperature * 1.8) + 32;      // Temperature in F
11. Serial.println(temperature);
12. delay(100);
13.  }
```

Flame Sensor

- You will interface a flame sensor to the Arduino. This sensor is specially designed to isolate specific bands in the IR (infrared) wavelengths and looks for unique patterns in flames for an extremely accurate reading.

Viewing Angle

- The viewing angle is at sixty degrees. Thus the sensor view is incredibly important as you design your projects.



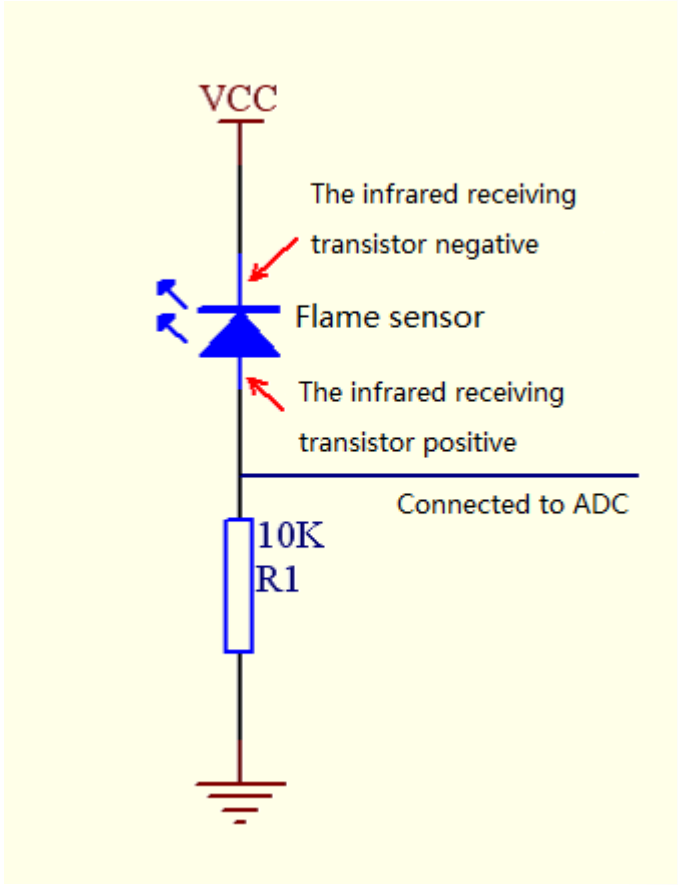
Flame Sensor Module Wavelength

- The flame sensor module detects wavelengths from 760nm-1100nm. There are other sources of heat that will also detect this wavelength. It is therefore important that you ensure that the only source of this particular range will be the flame that you want to detect.

Flame Sensor Diagram

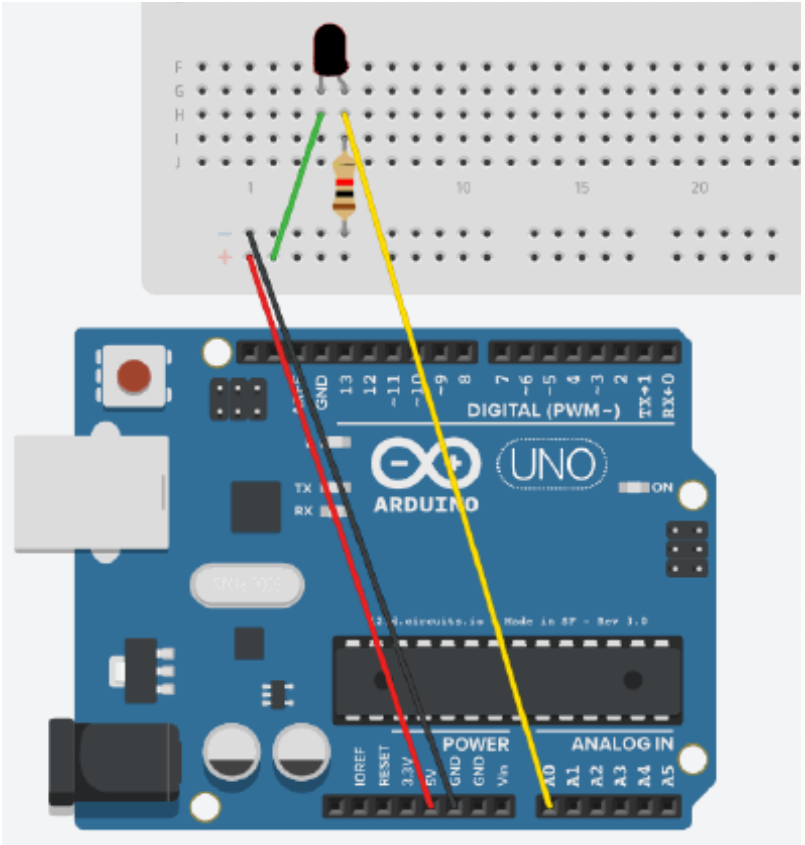
- Hardware Required:

1. Uno R3 board *1
2. USB Cable *1
3. Flame sensor *1
4. 10K resistor *1 (from long leg)
5. Breadboard*1
6. jumper wires* several



Flame Sensor (Circuit Diagram)

- The following circuit diagram is sample flame Sensor



Flame Sensor (Code_Example)

```
1.  int flame=0;// select analog pin 0 for the sensor
2.   int val=0;// initialize variable
3.   void setup() {
4.   Serial.begin(9600);// set baud rate at "9600"
5.   }
6.  void loop()
7.  {
8.    val=analogRead(flame);// read the analog value of the sensor
9.    Serial.println(val);// output and display the analog value
10.   delay(500);
11. }
```

Analog Write

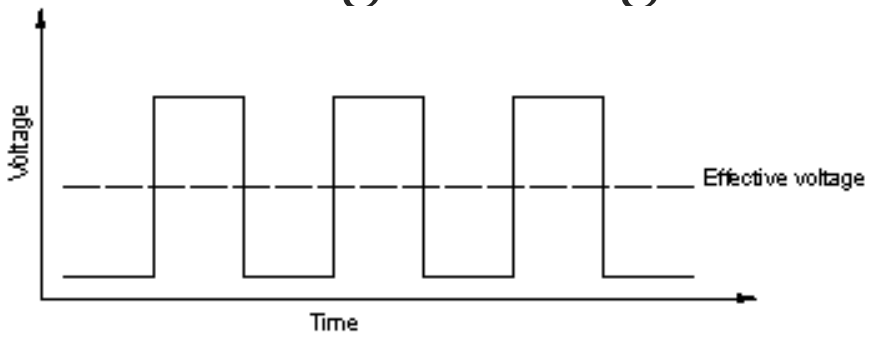
- The Arduino, and other digital microcontrollers can't produce a varying voltage, they can only produce a high voltage (5V) or low (0V). So instead, we "fake" an analog voltage by producing a series of voltage pulses at regular intervals, and varying the width of the pulses. This is called pulse width modulation (PWM).

Pulse Width Modulation

- The Arduino's programming language makes PWM easy to use; simply call analogWrite(pin, dutyCycle), where dutyCycle is a value from 0 to 255, and pin is one of the PWM pins (3, 5, 6, 9, 10, or 11). The analogWrite function provides a simple interface to the hardware PWM.

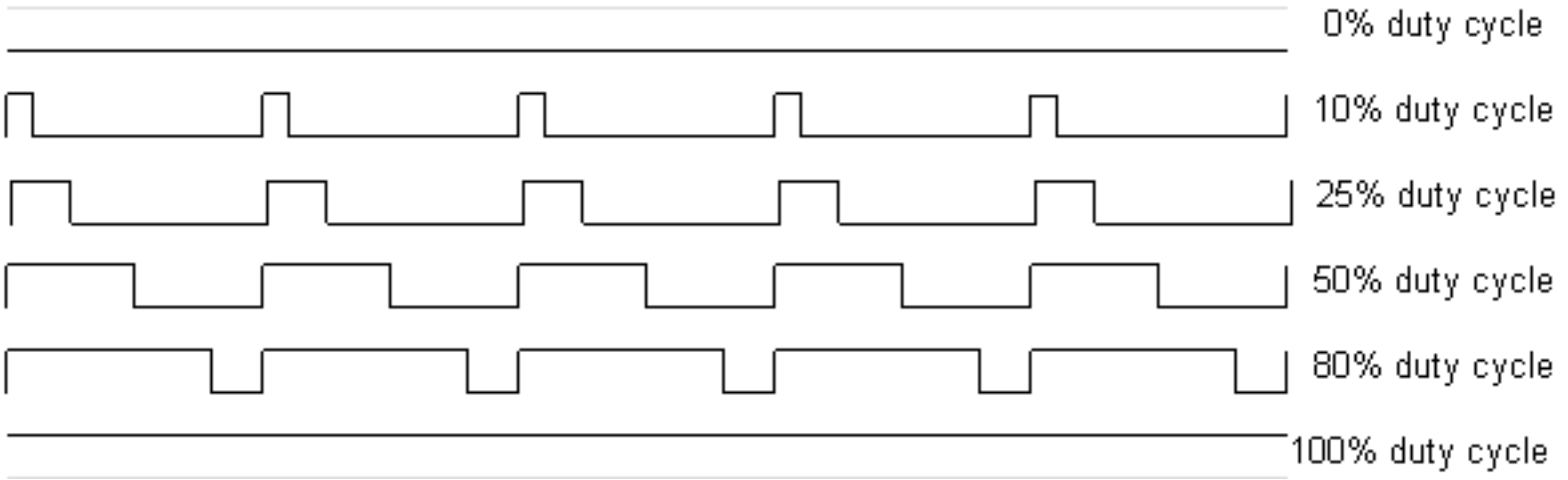
Duty cycle

- In the graph below, we pulse our pin high for the same length of time we pulse it low. The time the pin is high (called the pulsewidth). This ratio is called the duty cycle. The duty cycle is 50%,and the average voltage is about half the total volt



Duty cycle

- A PWM signal is a digital square wave, where the frequency is constant, but that fraction of the time the signal is on (the duty cycle) can be varied between 0 and 100%.



analogWrite()

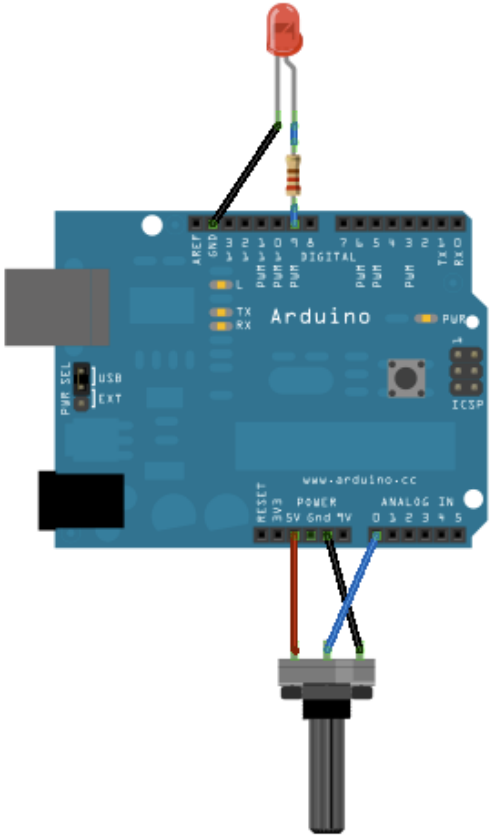
- Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite()) on the same pin.

analogWrite()

- You do not need to call pinMode() to set the pin as an output before calling analogWrite().
- Syntax
- ✓ analogWrite(pin, value)
- Parameters
- ✓ pin: the pin to write to. Allowed data types: int.
- ✓ value: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: int

analogWrite (Circuit Diagram)

- The following circuit diagram is sample diagram potentiometer and LED as analog output



analogWrite (Code_Example)

```
1.  int ledPin = 9;          // LED connected to digital pin
2.  int analogPin = 0;       // potentiometer connected to analog pin 0
3.  int val = 0;             // variable to store the read value
4.  void setup() {
5.  }
6.  void loop() {
7.  val = analogRead(analogPin);  // read the input pin
8.  analogWrite(ledPin, val / 4);   //analogWrite values from 0 to 255
9.  }
```

map() Function

- Re-maps a number from one range to another. That is, a value of **fromLow** would get mapped to **toLow**, a value of **fromHigh** to **toHigh**, values in-between to values in-between, etc.
- The map() function uses integer math so will not generate fractions, Fractional remainders are truncated, and are not rounded or averaged.

map() Function

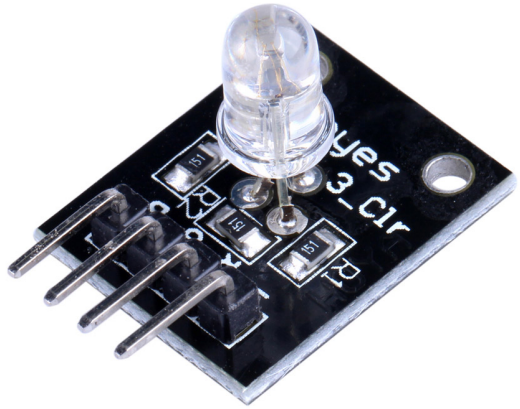
- **Syntax**
- ✓ map(value, fromLow, fromHigh, toLow, toHigh)
- **Parameters**
- ✓ value: the number to map
- ✓ fromLow: the lower bound of the value's current range
- ✓ fromHigh: the upper bound of the value's current range
- ✓ toLow: the lower bound of the value's target range
- ✓ toHigh: the upper bound of the value's target range

map function (Code_Example)

```
1.  int ledPin = 9;          // LED connected to digital pin
2.  int analogPin = 0; // potentiometer connected to analog pin 0
3.  int val = 0;             // variable to store the read value
4.  void setup() {
5.  }
6.  void loop() {
7.  val = analogRead(analogPin );
8.  int value = map(val, 0, 1023, 0, 255);
9.  analogWrite(ledPin , value);
10. }
```

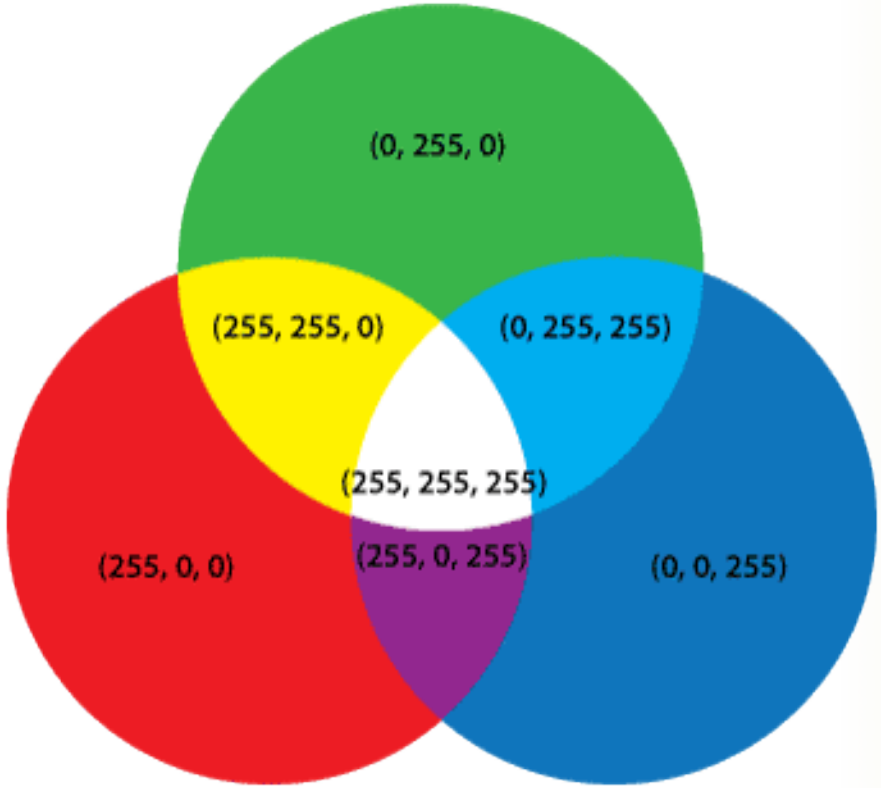
RGB color

- RGB color LED module can let LED emitting different color lights as RED,GREEN,BLUE signal input. This module will change of color assortment and make the LED flashing like rainbow.



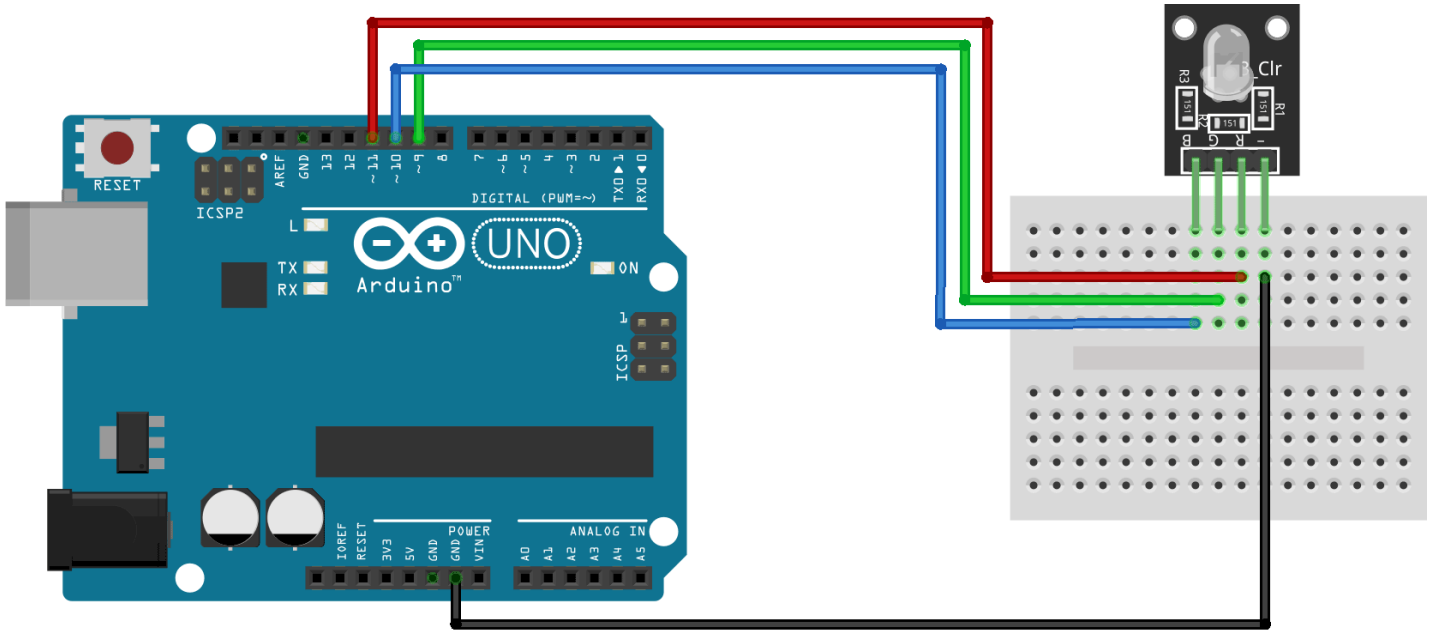
RGB color

- Combination of different colour will generate new colour



RGB module (Circuit Diagram)

- The following circuit diagram is sample diagram RGB Module



RGB module (Code Example)

```
1.  int redPin= 11;
2.  int greenPin = 10;
3.  int bluePin = 9;
4.  void setup() {
5.  }
6.  void loop() {
7.    setColor(255, 0, 0); // Red Color
8.    delay(1000);
9.     setColor(255, 255, 0); // Yello Color
10.   delay(1000);
11.   setColor(0, 0, 255); // Blue Color
12.   delay(1000);
13.   setColor(0, 255, 0); // Green Color
14.   delay(1000);
15.   setColor(255, 255, 255); // White Color
16.   delay(1000);
17.   setColor(170, 0, 255); // Purple Color
18.   delay(1000);

19.  }
20.  void setColor(int redValue, int greenValue, int blueValue) {
21.   analogWrite(redPin, redValue);
22.   analogWrite(greenPin, greenValue);
23.   analogWrite(bluePin, blueValue);
24.  }
```

Bit-banging Pulse Width Modulation

- You can "manually" implement PWM on any pin by repeatedly turning the pin on and off for the desired times.
- This technique has the advantage that it can use any digital output pin. In addition, you have full control the duty cycle and frequency.

Manually Implement PWM

```
1.  void setup()
2.  {
3.    pinMode(13, OUTPUT);
4.  }

5.  void loop()
6.  {
7.  for (int i=0; i<10000; i++){
8.    digitalWrite(13, HIGH);
9.    delayMicroseconds(i);
10.   digitalWrite(13, LOW);
11.   delayMicroseconds(10000 - i);
12. }
```

End

Any Questions?