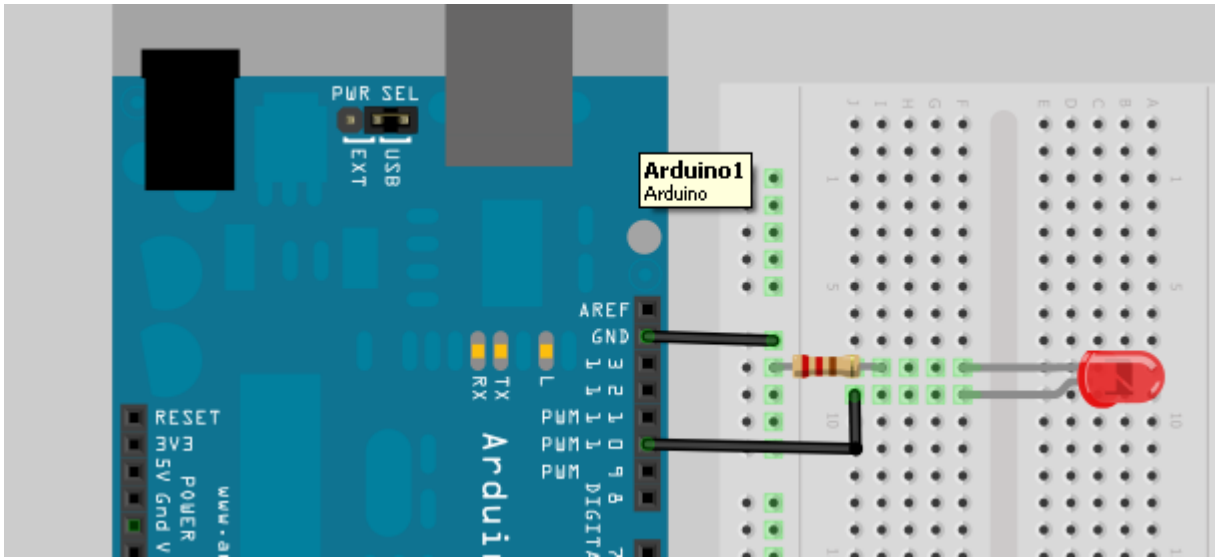
Aulas de Arduino uno

Book: Beginning Arduino-

Aula 1 LED Flasher

Hardware:



Code for “LED flasher”

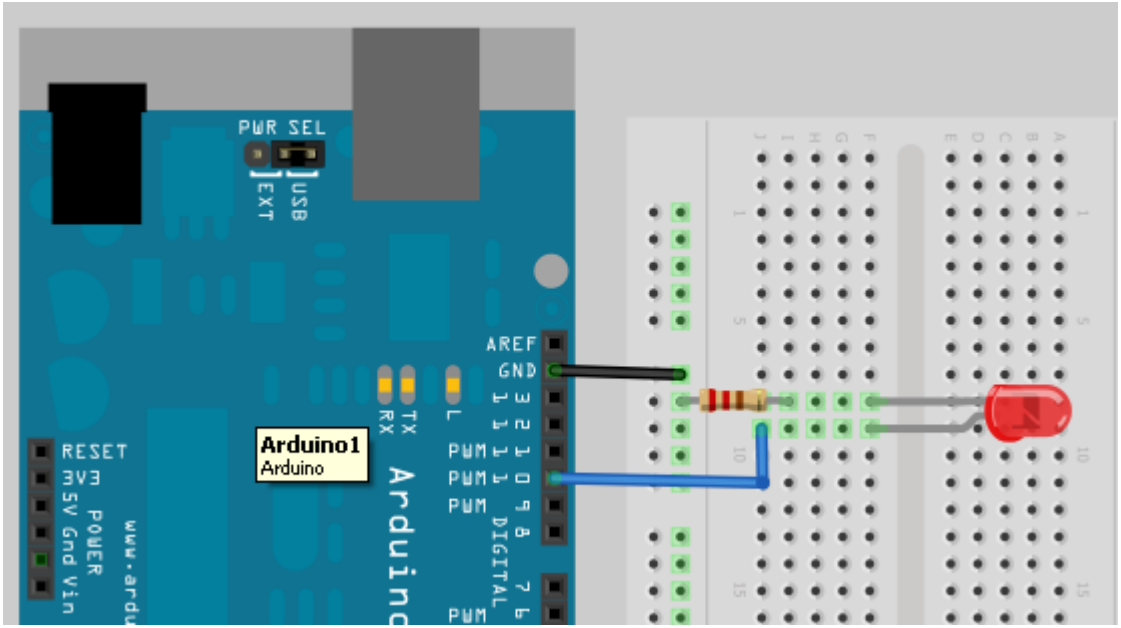
```
// Project 1 - LED Flasher
int ledPin = 10;
void setup() {
pinMode(ledPin, OUTPUT);
}
void loop() {
digitalWrite(ledPin, HIGH);
delay(1000);
digitalWrite(ledPin, LOW);
delay(1000);
}
```

- By varying the on and off times of the LED you can create any effect you want.
- For example,If you would like the LED to stay off for 5 seconds and then flash briefly (250ms), like the LED indicator on a car alarm, you could do this:

```
void loop() {
digitalWrite(ledPin, HIGH);
delay(250);
digitalWrite(ledPin, LOW);
delay(5000);
}
```

Aula 2 S.O.S. Morse Code Signaler

Hardware:

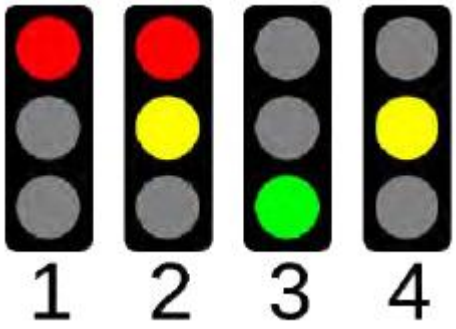


Code for “SOS”

```
// LED connected to pin 10
int ledPin = 10;
// run once, when the sketch starts
void setup()
{
// sets the pin as output
pinMode(ledPin, OUTPUT);
}
// run over and over again
void loop()
{
// 3 dits
for (int x=0; x<3; x++) {
digitalWrite(ledPin, HIGH); // sets the LED on
delay(150); // waits for 150ms
digitalWrite(ledPin, LOW); // sets the LED off
delay(100); // waits for 100ms
}
// 100ms delay between letters
delay(100);
// 3 dahs
for (int x=0; x<3; x++) {
digitalWrite(ledPin, HIGH); // sets the LED on
delay(400); // waits for 400ms
digitalWrite(ledPin, LOW); // sets the LED off
delay(100); // waits for 100ms
}
// 100ms delay between letters
delay(100);
// 3 dits again
for (int x=0; x<3; x++) {
digitalWrite(ledPin, HIGH); // sets the LED on
delay(150); // waits for 150ms
digitalWrite(ledPin, LOW); // sets the LED off
delay(100); // waits for 100ms
}
// wait 5 seconds before repeating
delay(5000);
}
```

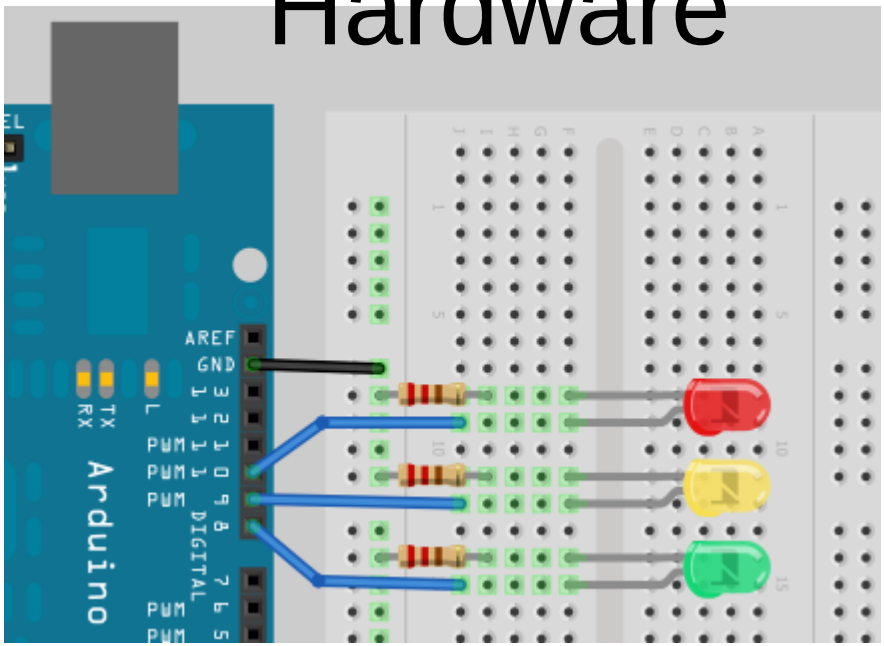
Project 3 – Traffic Lights

The project creates a set of traffic lights that will change from green to red, via amber, and back again, after a set length of time using the four-state UK system. This project could be used to make a set of working traffic lights for a model railway



The four states of the UK traffic light system

Hardware



Code for “ Traffic Lights”

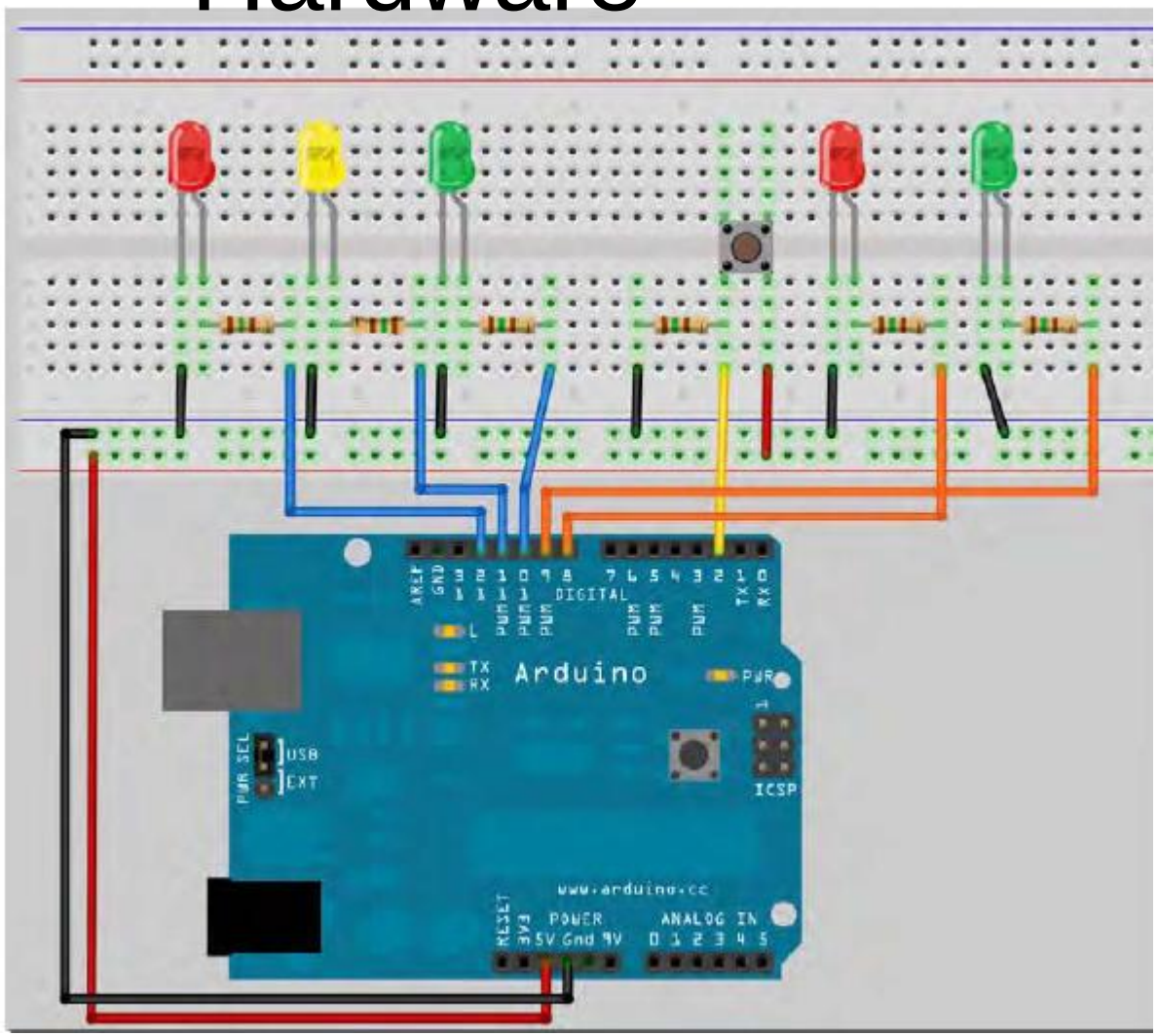
```
// Project 3 - Traffic Lights
int ledDelay = 5000; /* delay in
between changes*/
int redPin = 10;
int yellowPin = 9;
int greenPin = 8;
void setup() {
pinMode(redPin, OUTPUT);
pinMode(yellowPin, OUTPUT);
pinMode(greenPin, OUTPUT);
} void loop() {
digitalWrite(redPin, HIGH);
delay(ledDelay); // wait 5 seconds
digitalWrite(yellowPin, HIGH);
delay(2000); // wait 2 seconds
digitalWrite(greenPin, HIGH);
digitalWrite(redPin, LOW);
digitalWrite(yellowPin, LOW);
delay(ledDelay);
digitalWrite(yellowPin, HIGH);
digitalWrite(greenPin, LOW);
delay(2000); // wait 2 seconds
digitalWrite(yellowPin, LOW);
// now our loop repeats
}
```

Project 4 – “Interactive Traffic Lights”

In this project is included a set of pedestrian lights to the “traffic lights “ program and a pedestrian push button to request to cross the road.

The Arduino will react when the button is pressed by changing the state of the lights to make the cars stop and allow the pedestrian to cross safely. We also interact with the Arduino and cause it to do something when we change the state of a button that the Arduino is watching. We learn how to create our own functions in code.

Hardware



```
// Project 4 - Interactive Traffic Lights
int carRed = 12; // assign the car lights
int carYellow = 11;
int carGreen = 10;
int pedRed = 9; // assign the pedestrian
//lights
int pedGreen = 8;
int button = 2; // button pin
int crossTime = 5000; // time to cross
unsigned long changeTime; // time since
//button pressed
void setup() {
pinMode(carRed, OUTPUT);
pinMode(carYellow, OUTPUT);
pinMode(carGreen, OUTPUT);
pinMode(pedRed, OUTPUT);
pinMode(pedGreen, OUTPUT);
pinMode(button, INPUT); // button pin 2
// turn on the green light
digitalWrite(carGreen, HIGH);
digitalWrite(pedRed, HIGH);
}
void loop() {
int state = digitalRead(button);
/* check if button is pressed and it is over 5
seconds since last button press */
if (state == HIGH && (millis() - changeTime)
> 5000) {
// Call the function to change the lights
changeLights();
}
}
void changeLights() {
digitalWrite(carGreen, LOW); // green off
digitalWrite(carYellow, HIGH); // yellow on
delay(2000); // wait 2 seconds
digitalWrite(carYellow, LOW); // yellow off
digitalWrite(carRed, HIGH); // red on
delay(1000); // wait 1 second till its safe
digitalWrite(pedRed, LOW); // ped red off
digitalWrite(pedGreen, HIGH); // ped green
on
delay(crossTime); // wait for preset time
period
```

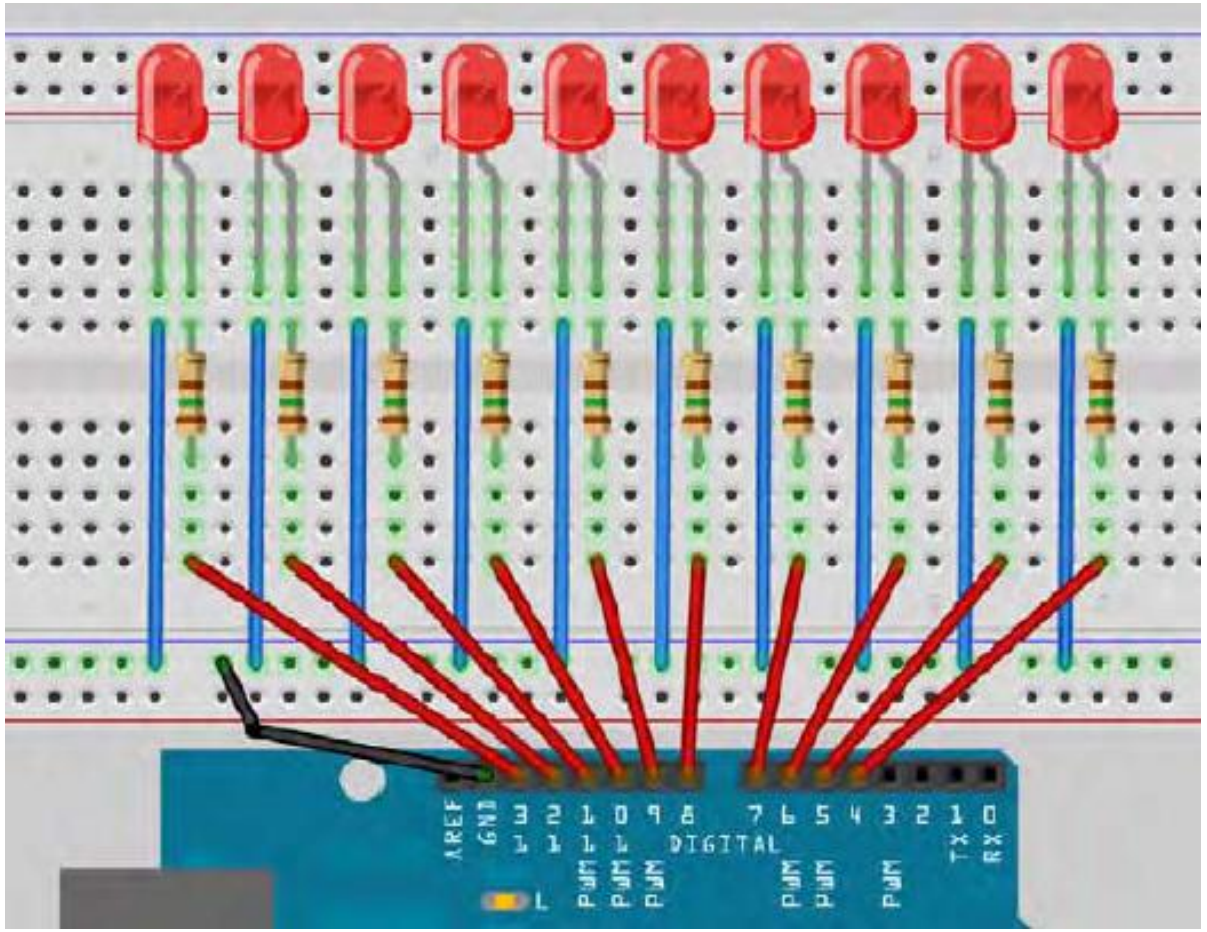
```
// flash the ped green
for (int x=0; x<10; x++) {
digitalWrite(pedGreen, HIGH);
delay(250);
digitalWrite(pedGreen, LOW);
delay(250);
}
// turn ped red on
digitalWrite(pedRed, HIGH);
delay(500);
digitalWrite(carYellow, HIGH); // yellow on
digitalWrite(carRed, LOW); // red off
delay(1000);
digitalWrite(carGreen, HIGH);
digitalWrite(carYellow, LOW); // yellow off
// record the time since last change of lights
changeTime = millis();
// then return to the main program loop
}
```

Table 2-2. Data types

Data type	RAM	Number Range
void keyword	N/A	N/A
boolean	1 byte	0 to 1 (True or False)
byte	1 byte	0 to 255
char	1 byte	-128 to 127
unsigned char	1 byte	0 to 255
int	2 byte	-32,768 to 32,767
unsigned int	2 byte	0 to 65,535
word	2 byte	0 to 65,535
long	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long	4 byte	0 to 4,294,967,295
float	4 byte	-3.4028235E+38 to 3.4028235E+38
double	4 byte	-3.4028235E+38 to 3.4028235E+38
string	1 byte + x	Arrays of chars
array	1 byte + x	Collection of variables

Project 5 – LED Chase Effect

You're going to use a string of LEDs (10 in total) to make an LED chase effect. This project will introduce the concept of arrays.



Code for Project 5

```
// Project 5 - LED Chase Effect
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11,
12, 13}; // Create array for LED pins
int ledDelay(65); // delay between
//changes
int direction = 1;
int currentLED = 0;
unsigned long changeTime;
void setup() {
for (int x=0; x<10; x++) { // set all pins
//to output
pinMode(ledPin[x], OUTPUT); }
changeTime = millis();
}
void loop() {
if ((millis() - changeTime) > ledDelay) {
// if it has been ledDelay ms since
//last change
changeLED();
changeTime = millis();
}
}
void changeLED() {
for (int x=0; x<10; x++) { // turn off all
LED's
digitalWrite(ledPin[x], LOW);
}
digitalWrite(ledPin[currentLED], HIGH);
// turn on the current LED
currentLED += direction; // increment
by the direction value
// change direction if we reach the end
if (currentLED == 9) {direction = -1;}
if (currentLED == 0) {direction = 1;}
}
```

Note that we have to stop the program, manually change the value of **ledDelay,** and then upload the amended code to see any changes.
But, it is possible to adjust the speed while the program is running. We will interact with the program and adjust the speed using a potentiometer. See project 6.

Project 6 – Interactive LED Chase Effect

4.7KΩ Rotary Potentiometer

Project 6

The code for this Project is almost identical to the previous project. We simply added a potentiometer to the hardware and the code additions enable it to read the values from the potentiometer and use them to adjust the speed of the LED chase effect.

int potPin = 2;

potentiometer is connected to Analog Pin 2. The Arduino has six analog input/outputs with a 10-bit analog to digital converter. This means the analog pin can read in voltages between 0 to 5 volts ***in integer values*** between 0 (0 volts) and 1,023 (5 volts). This gives a resolution of 5 volts / 1024 units or 0.0049 volts (4.9mV) per unit.

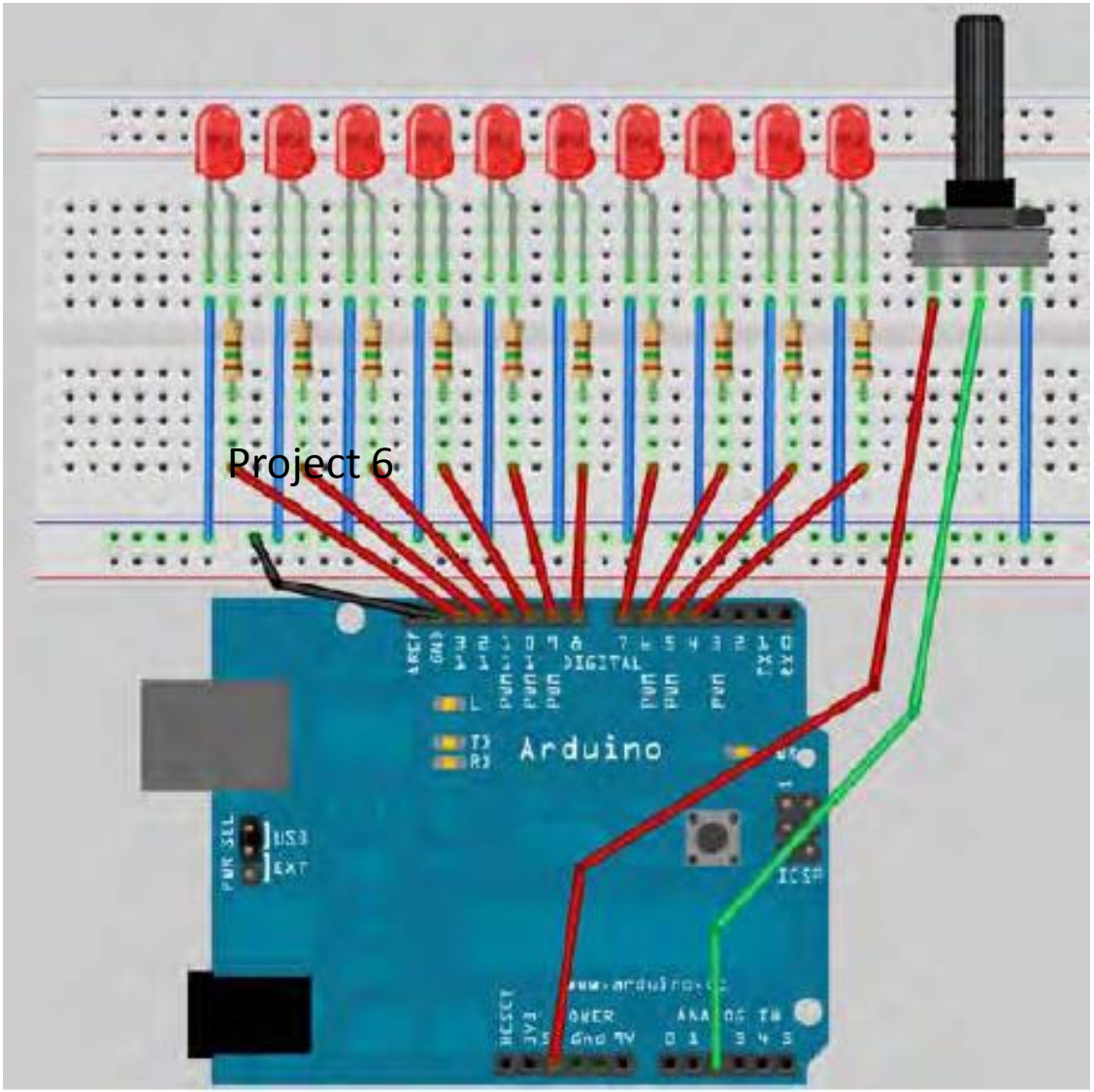
We can use the direct values read in from the potentiometer to adjust the delay between 0 and 1023 milliseconds (or just over 1 second).

ledDelay = analogRead(potPin);

This is done during the main loop and therefore it is constantly being read and adjusted. By turning the knob, we can adjust the delay value between 0 and 1023 milliseconds (or just over a second) and thus have full control over the speed of the effect.

Project 6

Hardware

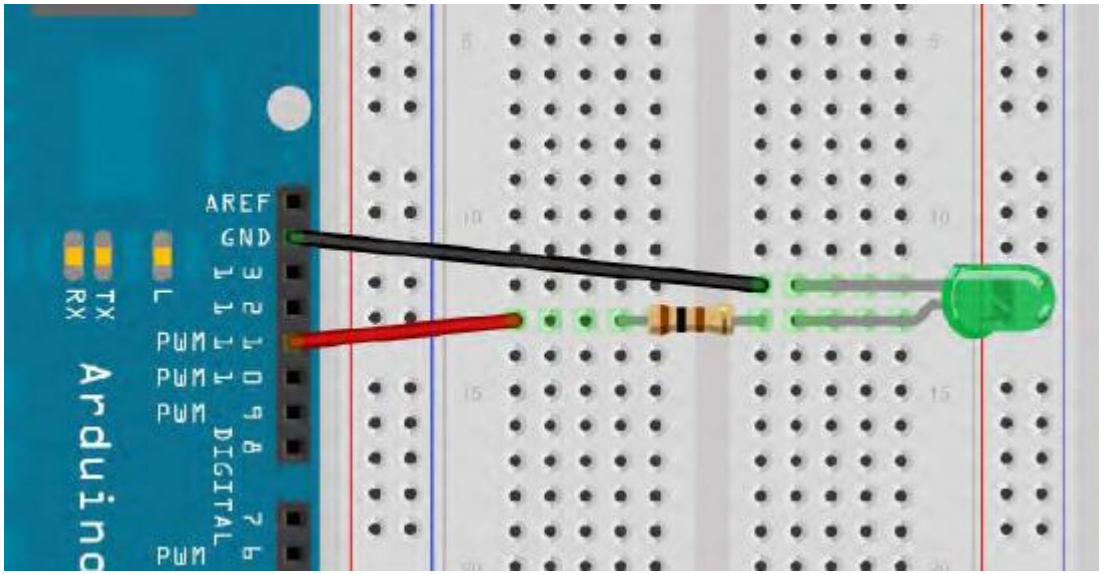


Code for Project 6

```
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11,
12, 13}; // Create array for LED pins
int ledDelay; // delay between
//changes
int direction = 1;
int currentLED = 0;
unsigned long changeTime;
int potPin = 2; // select the input pin
//for the potentiometer
void setup() {
for (int x=0; x<10; x++) { // set all pins
//to output
pinMode(ledPin[x], OUTPUT); }
changeTime = millis();
}
void loop() {
ledDelay = analogRead(potPin);
// read the value from the pot
if ((millis() - changeTime) > ledDelay) {
// if it has been ledDelay ms since
//last change
changeLED();
changeTime = millis();
}
}
void changeLED() {
for (int x=0; x<10; x++) { // turn off all
LED's
digitalWrite(ledPin[x], LOW);
}
digitalWrite(ledPin[currentLED],
HIGH); // turn on the current LED
currentLED += direction; // increment
by the direction value
// change direction if we reach the end
if (currentLED == 9) {direction = -1;}
if (currentLED == 0) {direction = 1;}
}
```

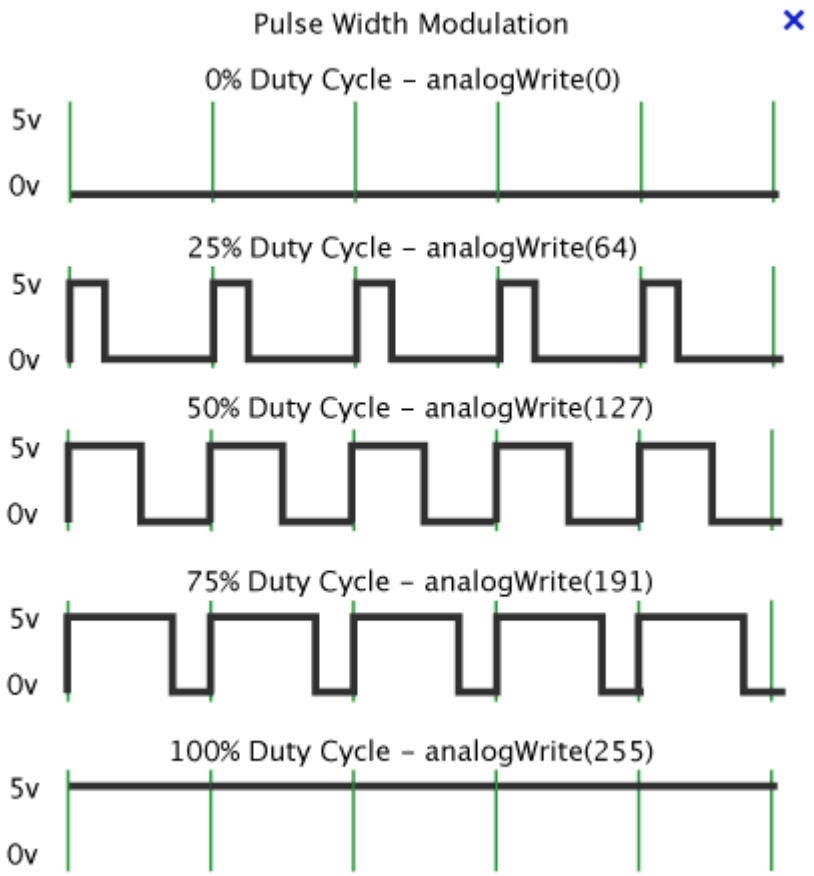
Project 7 – Pulsating Lamp

Hardware



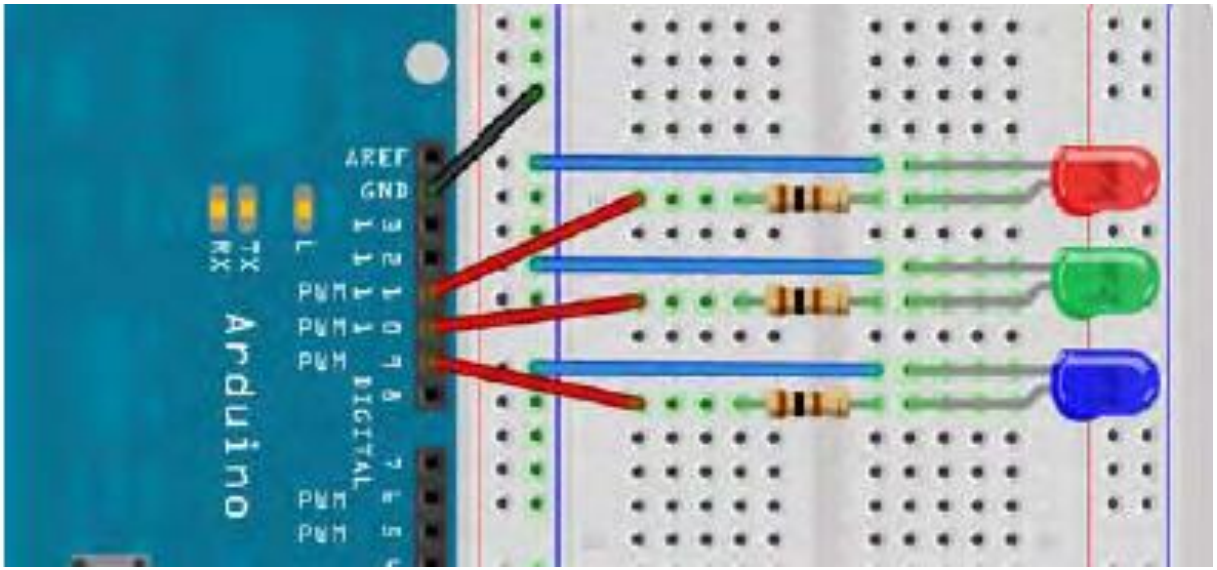
```
// Project 7 - Pulsating lamp
int ledPin = 11;
float sinVal;
int ledVal;
void setup() {
pinMode(ledPin, OUTPUT);
}
void loop() {
for (int x=0; x<180; x++) {
// convert degrees to radians then obtain sin value
sinVal = (sin(x*(3.1412/180)));
ledVal = int(sinVal*255);
analogWrite(ledPin, ledVal);
delay(25);
}
}
```

Six of the Arduino pins (3, 5, 6, 9, 10 & 11) have PWM written next to them. These pins differ from the remaining digital pins in that they are able to send out a PWM signal. PWM stands for Pulse Width Modulation, which is a technique for getting analog results from digital means. On these pins, the Arduino sends out a square wave by switching the pin on and off very fast. The pattern of on/offs can simulate a varying voltage between 0 and 5v. It does this by changing the amount of time that the output remains high (on) versus off (low). The duration of the on time is known as the *pulse width*.



Project 8 – RGB Mood Lamp

Hardware



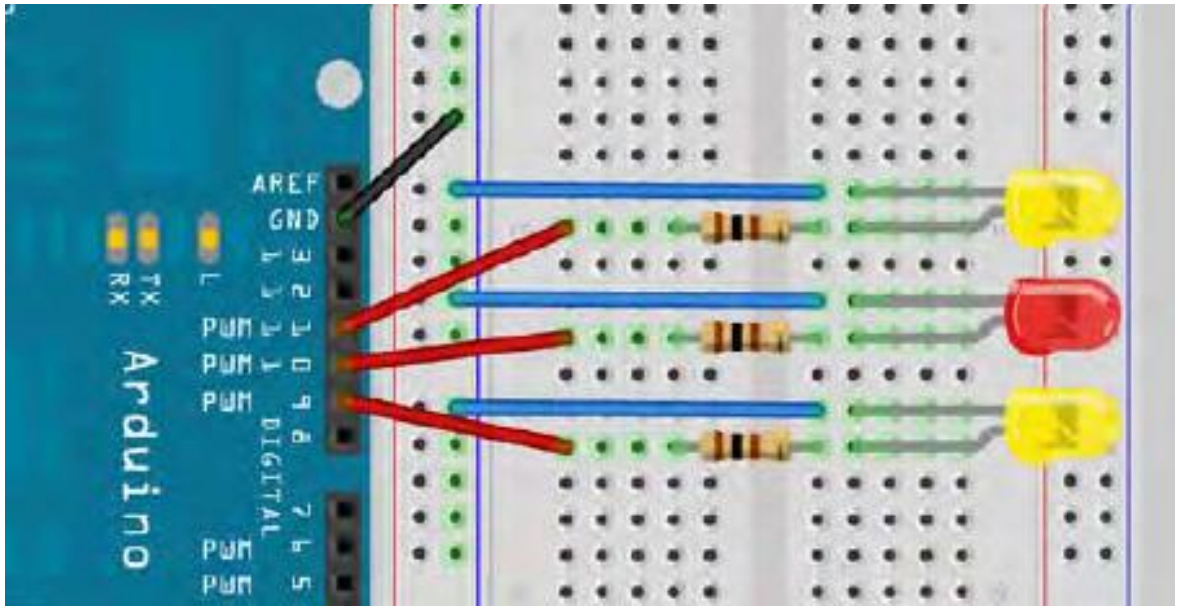
Code for Project 8

```
// Project 8 - Mood Lamp
float RGB1[3];
float RGB2[3];
float INC[3];
int red, green, blue;
int RedPin = 11;
int GreenPin = 10;
int BluePin = 9;
void setup()
{
randomSeed(analogRead(0));
RGB1[0] = 0;
RGB1[1] = 0;
RGB1[2] = 0;
RGB2[0] = random(256);
RGB2[1] = random(256);
RGB2[2] = random(256);
}
void loop()
{
randomSeed(analogRead(0));
for (int x=0; x<3; x++) {
INC[x] = (RGB1[x] - RGB2[x]) / 256; }
for (int x=0; x<256; x++) {
red = int(RGB1[0]);
green = int(RGB1[1]);
blue = int(RGB1[2]);
analogWrite (RedPin, red);
analogWrite (GreenPin, green);
analogWrite (BluePin, blue);
delay(100);
RGB1[0] -= INC[0];
RGB1[1] -= INC[1];
RGB1[2] -= INC[2];
} for (int x=0; x<3; x++) {
RGB2[x] = random(556)-300;
RGB2[x] = constrain(RGB2[x], 0, 255);
delay(1000);
}
}
```

Project 9 – LED Fire Effect

Project 9 will use LEDs and a flickering random light effect, via PWM again, to mimic the effect of a flickering flame. This simple example show how LEDs can be used to create special effects for movies, stage plays, model dioramas, model railways, etc.

Hardware

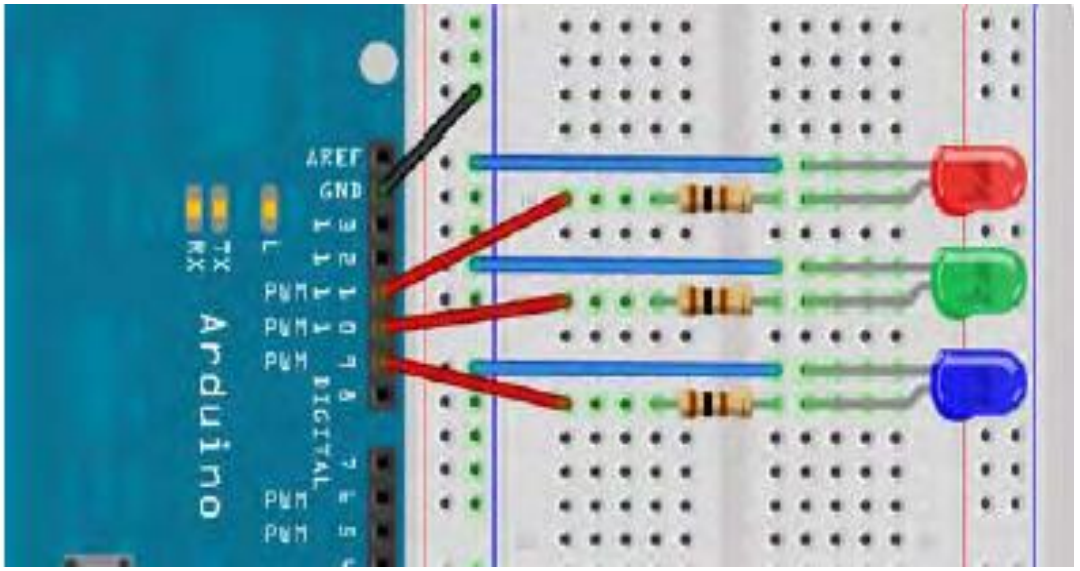


```
// Project 9 - LED Fire Effect
int ledPin1 = 9;
int ledPin2 = 10;
int ledPin3 = 11;
void setup()
{
pinMode(ledPin1, OUTPUT);
pinMode(ledPin2, OUTPUT);
pinMode(ledPin3, OUTPUT);
}
void loop()
{
analogWrite(ledPin1, random(120)+135);
analogWrite(ledPin2, random(120)+135);
analogWrite(ledPin3, random(120)+135);
delay(random(100));
}
```

Project 10 – Serial Controlled Mood Lamp

In this program we send commands from the PC to the Arduino using the **Serial Monito**r in the Arduino IDE. Serial communication is the process of sending data one bit at a time across a communication link.
It introduces how to manipulate text strings..

Hardware



Project – 10 (1º part)

```
// Project 10 - Serial controlled mood lamp

char buffer[18];
int red, green, blue;
int RedPin = 11;
int GreenPin = 10;
int BluePin = 9;
void setup()
{
        Serial.begin(9600);
        Serial.flush();
        pinMode(RedPin, OUTPUT);
        pinMode(GreenPin, OUTPUT);
        pinMode(BluePin, OUTPUT);
}
void loop()
{
        if (Serial.available() > 0) {
        int index=0;
        delay(100); // let the buffer fill
        //up
        int numChar = Serial.available();
        If (numChar>15) {
                numChar=15;
        }
        while (numChar--) {
        buffer[index++] = Serial.read();
        }
        splitString(buffer);
        }
}
void splitString(char* data) {
        Serial.print("Data entered: ");
        Serial.println(data);
        char* parameter;
        parameter = strtok (data, " ,");
        while (parameter != NULL) {
setLED(parameter);
parameter = strtok (NULL, " ,");
}
```

Project – 10 (2º part)

```
            // Clear the text and serial buffers
            for (int x=0; x<16; x++) {
            buffer[x]='\0';
            }
Serial.flush();
}
void setLED(char* data) {
     if ((data[0] == 'r') || (data[0] == 'R')) {
            int Ans = strtol(data+1, NULL, 10);
            Ans = constrain(Ans,0,255);
            analogWrite(RedPin, Ans);
            Serial.print("Red is set to: ");
            Serial.println(Ans);
}
    if ((data[0] == 'g') || (data[0] == 'G')) {
            int Ans = strtol(data+1, NULL, 10);
            Ans = constrain(Ans,0,255);
            analogWrite(GreenPin, Ans);
            Serial.print("Green is set to: ");
            Serial.println(Ans);
}
    if ((data[0] == 'b') || (data[0] == 'B')) {
            int Ans = strtol(data+1, NULL, 10);
            Ans = constrain(Ans,0,255);
            analogWrite(BluePin, Ans);
            Serial.print("Blue is set to: ");
            Serial.println(Ans);
}
}
```

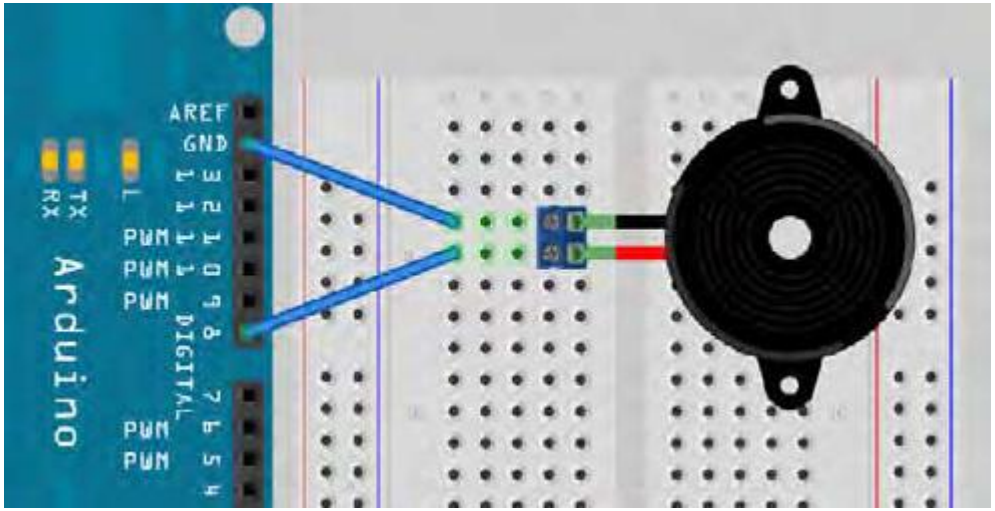
Project - 10

Start the **Serial Monitor** by clicking its icon in the **Arduino IDE** taskbar. Enter the R, G, and B values for each of the three LEDs manually. The LEDs will change to the color you have input.
For example:
r255 b100
r127 b127 g127
G255, B0
B127, R0, G255

Project 11 – Piezo Sounder Alarm

By connecting a piezo sounder to a digital output pin, we can create a wailing alarm sound.

Hardware

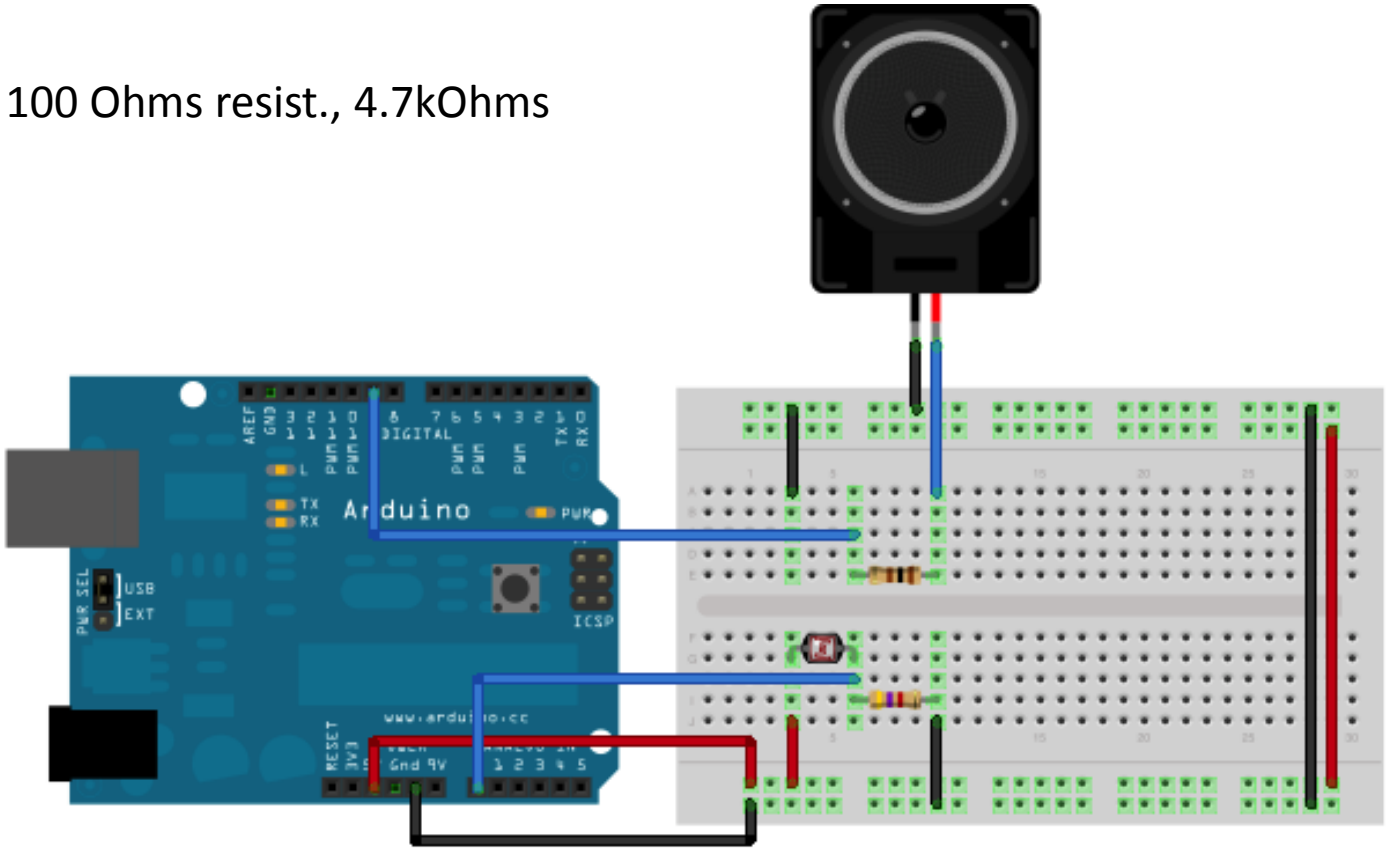


Project - 11

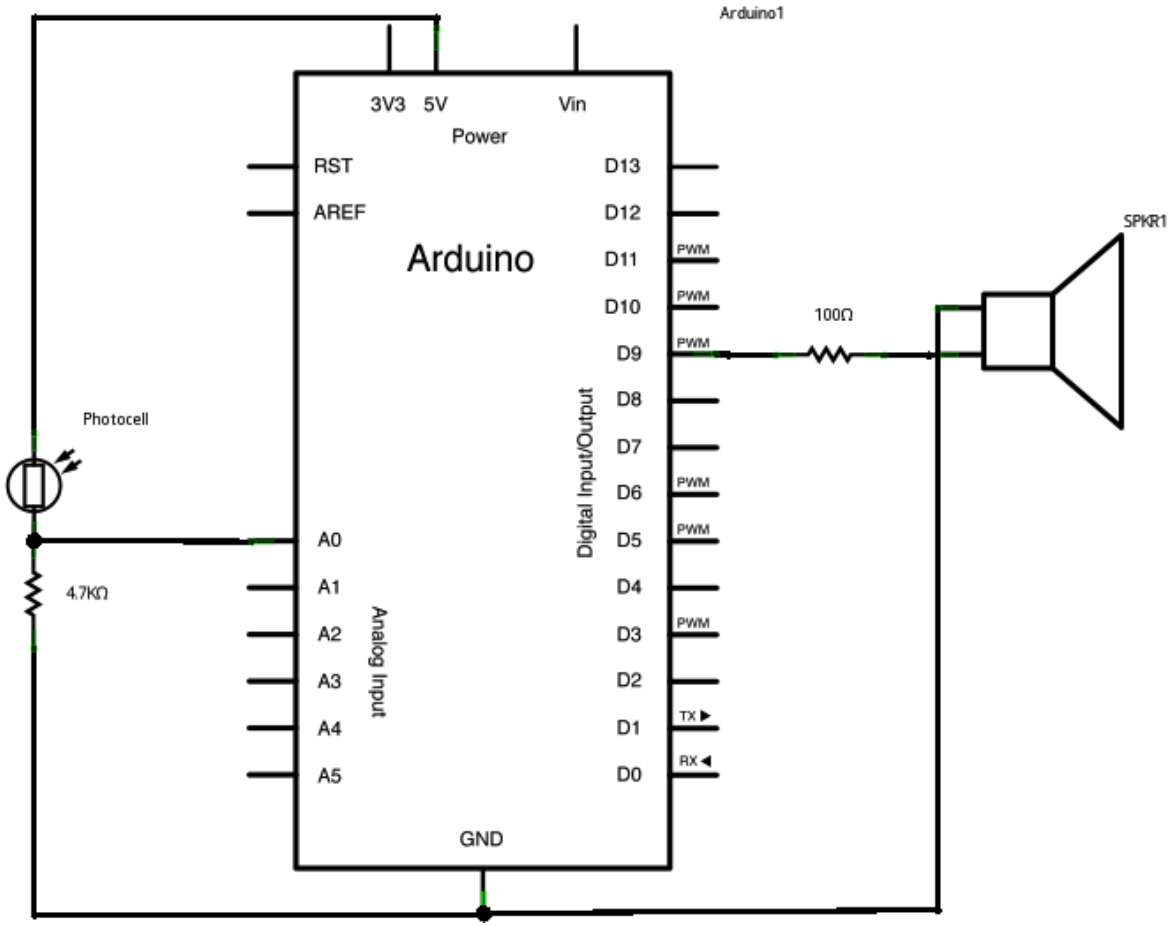
```
// Project 11 - Piezo Sounder Alarm
float sinVal;
int toneVal;
void setup() {
pinMode(8, OUTPUT);
}
void loop() {
            for (int x=0; x<180; x++) {
            // convert degrees to radians then obtain sin value
            sinVal = (sin(x*(3.1412/180)));
            // generate a frequency from the sin value
            toneVal = 2000+(int(sinVal*1000));
            tone(8, toneVal);
            delay(2);
            }
}
```

Project – 12
Pitch follower using the tone() function

Hardware:
8 Ohms speaker, 100 Ohms resist., 4.7kOhms resistor, LDR.



By using “Fritzing” software, we get the circuit layout:



```
/*  Pitch follower.  Plays a pitch that changes based on a changing analog input.
 circuit: 8-ohm speaker on digital pin 9 photoresistor on analog 0 to 5V, 4.7K resistor on
analog 0 to ground. ( created 21 Jan 2010, modified 31 May 2012, by Tom Igoe, with
suggestion from Michael Flynn. This example code is in the public domain.
 http://arduino.cc/en/Tutorial/Tone2 */
void setup() {
  // initialize serial communications (for debugging only):
  Serial.begin(9600);
}
void loop() {
  // read the sensor:
  int sensorReading = analogRead(A0);
  // print the sensor reading so you know its range
  Serial.println(sensorReading);
  // map the analog input range (in this case, 400 - 1000 from the photoresistor)
  // to the output pitch range (120 - 1500Hz)
  // change the minimum and maximum input numbers below
  // depending on the range your sensor's giving:
  int thisPitch = map(sensorReading, 400, 1000, 120, 1500);
  // play the pitch:
  tone(9, thisPitch, 10);
  delay(1);        // delay in between reads for stability
}
```

Project – 12. Function *map()*

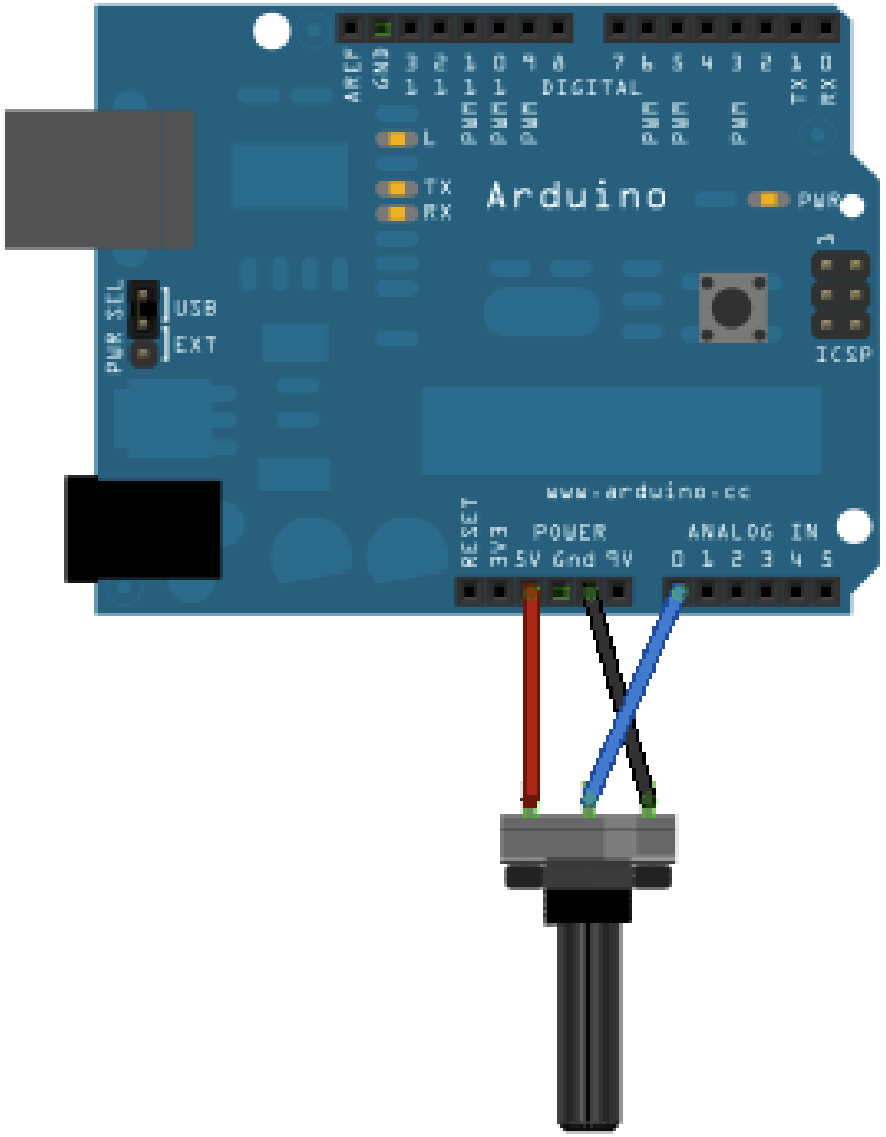
map(value, fromLow, fromHigh, toLow, toHigh)
Description
Re-maps a number from one range to another. That is, a **value** of **fromLow** would get mapped to **toLow**, a value of**fromHigh** to **toHigh**, values in-between to values in-between, etc
Example

```
/* Map an analog value to 8 bits (0 to 255) */
void setup() {}

void loop()
{
  int val = analogRead(0);
  val = map(val, 0, 1023, 0, 255);
  analogWrite(9, val);
}
```

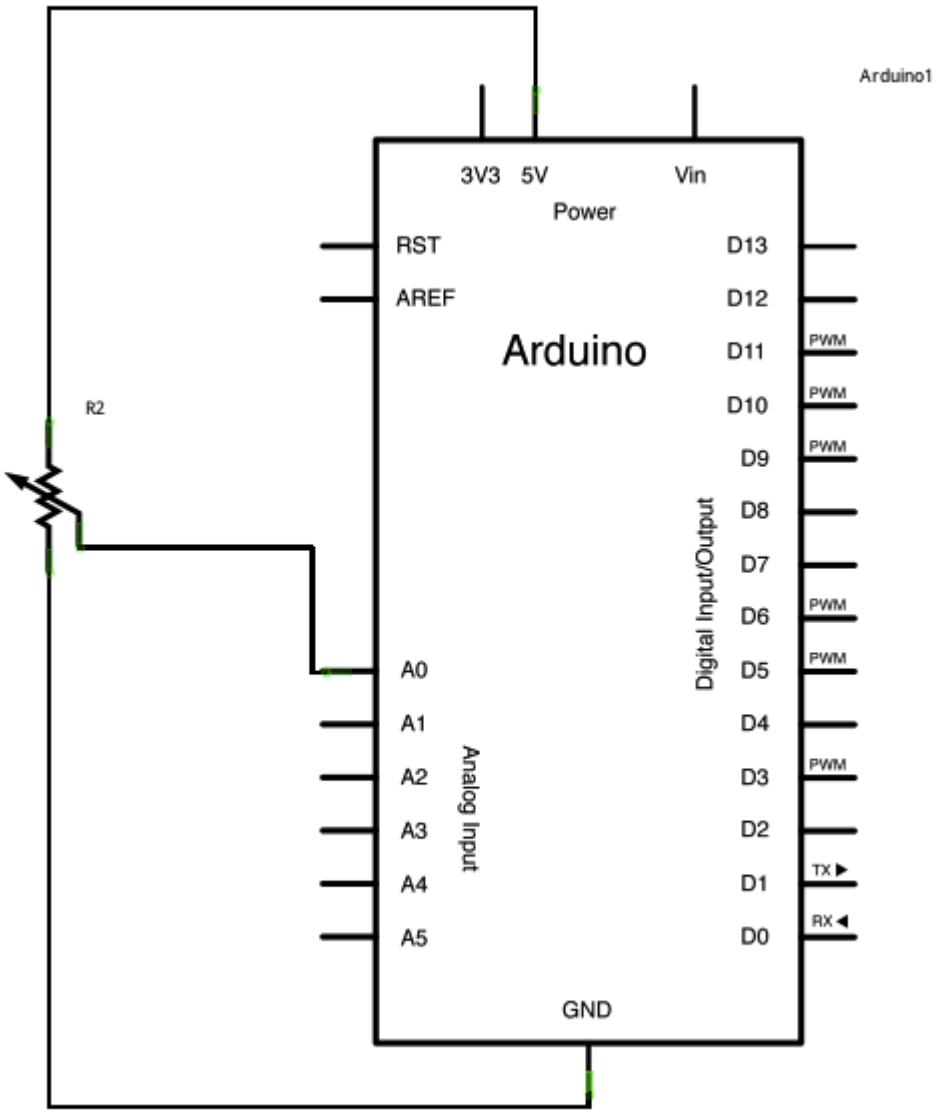
project-13
Analog Read Serial-

This example shows how to read analog input from the physical world using a potentiometer.



project-13
Analog Read Serial-

When we open the **Serial Monitor** in the **IDE** (Arduino development environment) we should see a steady stream of numbers ranging from 0-1023, correlating to the position of the pot.



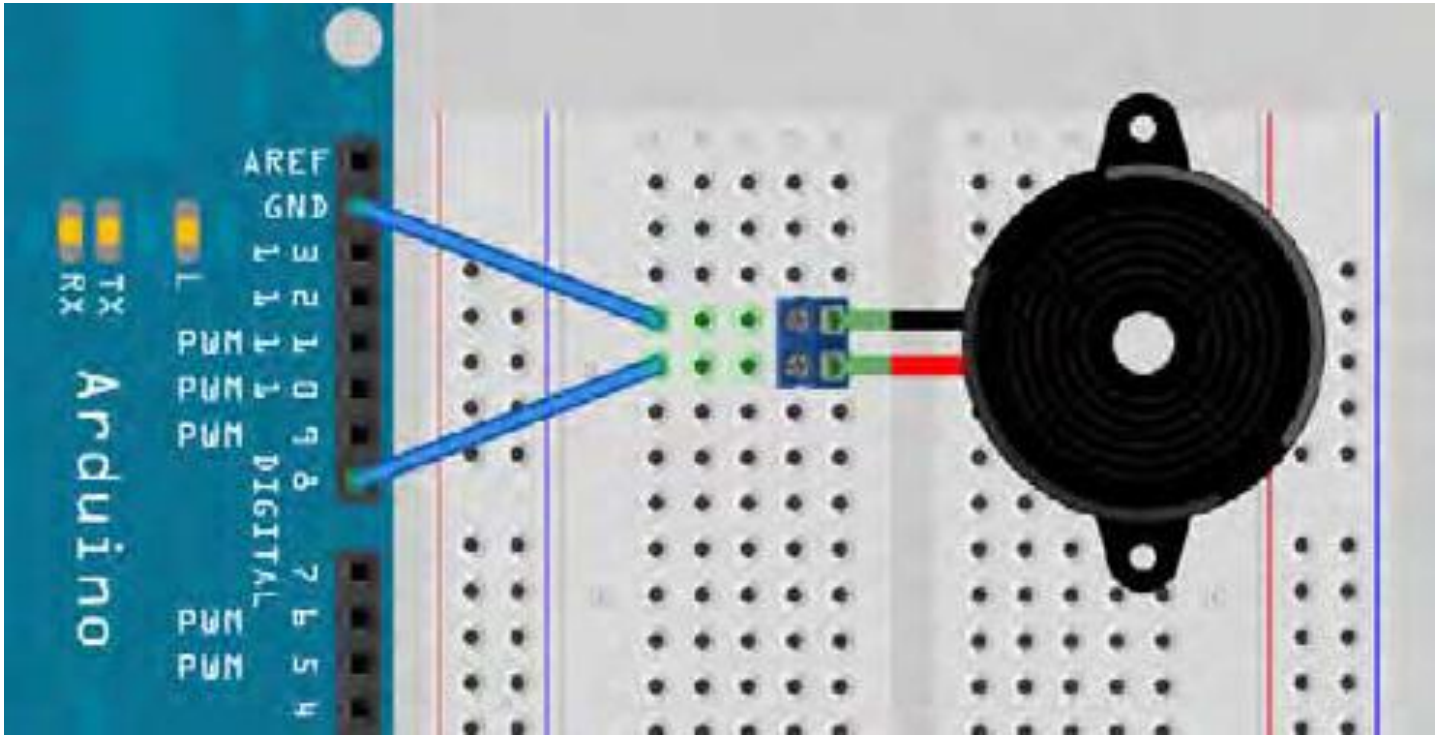
Project-13
Analog Read Serial-

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and
ground. This example code is in the public domain. */
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
                              }

// the loop routine runs over and over again forever:
void loop() {
              // read the input on analog pin 0:
               int sensorValue = analogRead(A0);
              // print out the value you read:
               Serial.println(sensorValue);
              delay(1);       // delay in between reads for stability
}
```

Project-14
Piezo Sounder Melody Player

Hardware



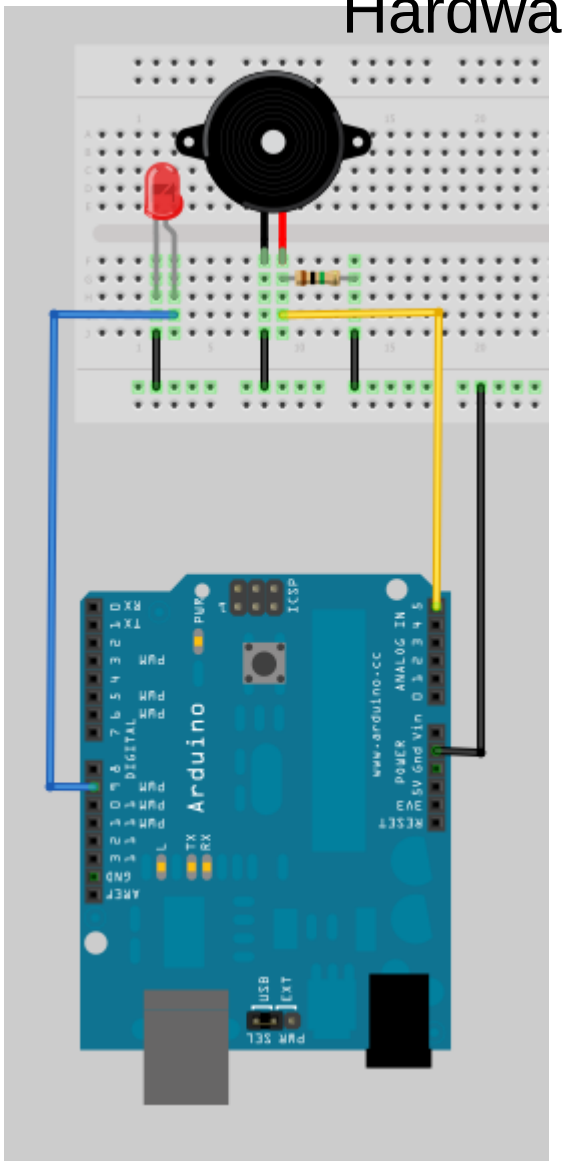
Project-14. Code

```
// Project 12 -  Melody Player
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define WHOLE 1
#define HALF 0.5
#define QUARTER 0.25
#define EIGHTH 0.125
#define SIXTEENTH 0.0625
```

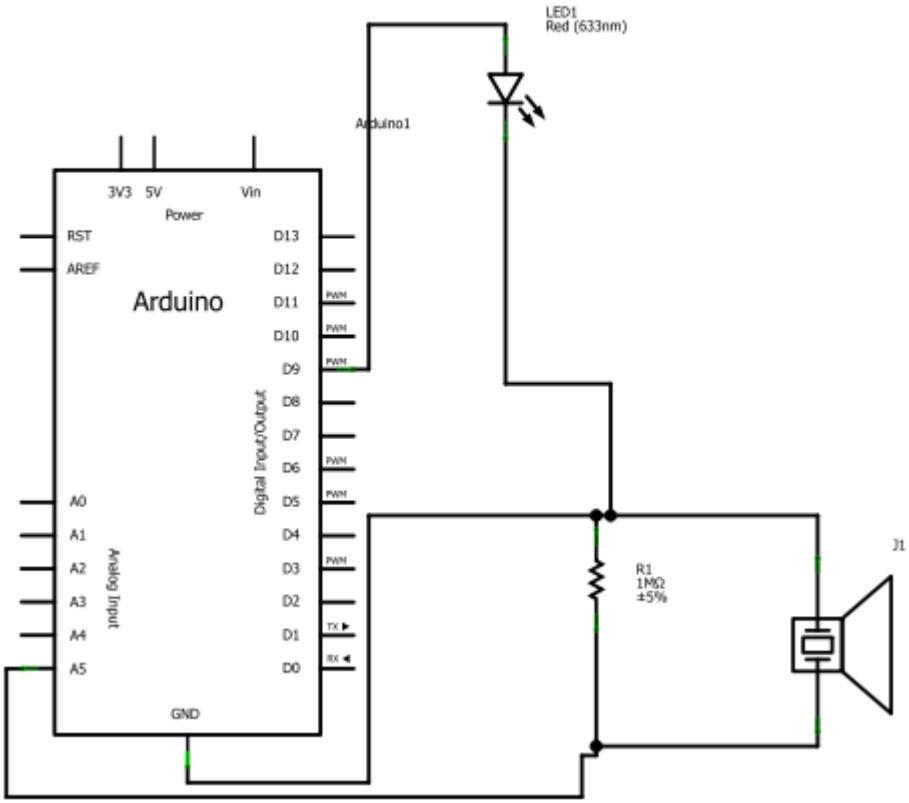
```
int tune[] = { NOTE_C4, NOTE_C4, NOTE_C4, NOTE_C4, NOTE_C4, NOTE_B3, NOTE_G3,
NOTE_A3, NOTE_C4, NOTE_C4, NOTE_G3, NOTE_G3, NOTE_F3, NOTE_F3, NOTE_G3,
NOTE_F3, NOTE_E3, NOTE_G3, NOTE_C4, NOTE_C4, NOTE_C4, NOTE_C4, NOTE_A3,
NOTE_B3, NOTE_C4, NOTE_D4};
float duration[] = { EIGHTH, QUARTER+EIGHTH, SIXTEENTH, QUARTER, QUARTER, HALF,
HALF, HALF, QUARTER, QUARTER, HALF+QUARTER, QUARTER, QUARTER, QUARTER,
QUARTER+EIGHTH, EIGHTH, QUARTER, QUARTER, QUARTER, EIGHTH, EIGHTH, QUARTER,
QUARTER, QUARTER, QUARTER, HALF+QUARTER};
int length;
void setup() {
        pinMode(8, OUTPUT);
        length = sizeof(tune) / sizeof(tune[0]);
        }
void loop() {
        for (int x=0; x<length; x++) {
        tone(8, tune[x]);
        delay(1500 * duration[x]);
        noTone(8);
        }
delay(5000);
}
```

Project 15 – Piezo Knock Sensor

Hardware - Protoboard



Schematic



Code- Project 15

```
// Project 13 - Piezo Knock Sensor
int ledPin = 9; // LED on Digital Pin 9
int piezoPin = 5; // Piezo on Analog Pin 5
int threshold = 120; // The sensor value to reach before activation
int sensorValue = 0; // A variable to store the value read from the sensor
float ledValue = 0; // The brightness of the LED
void setup() {
pinMode(ledPin, OUTPUT); // Set the ledPin to an OUTPUT
// Flash the LED twice to show the program has started
digitalWrite(ledPin, HIGH); delay(150); digitalWrite(ledPin, LOW); delay(150);
digitalWrite(ledPin, HIGH); delay(150); digitalWrite(ledPin, LOW); delay(150);
}
void loop() {
sensorValue = analogRead(piezoPin); // Read the value from the sensor
if (sensorValue >= threshold) { // If knock detected set brightness to max
ledValue = 255;
}
analogWrite(ledPin, int(ledValue) ); // Write brightness value to LED
ledValue = ledValue - 0.05; // Dim the LED slowly
if (ledValue <= 0) { ledValue = 0;} // Make sure value does not go below zero
}
```