Practical no.1

Title: Digital Output

Aim: To make LED keep blinking

Apparatus: Arduino, Breadboard, USB Cable, Resistor, LED, Wire

Theory: A signal with respect to time has some value. In case of a

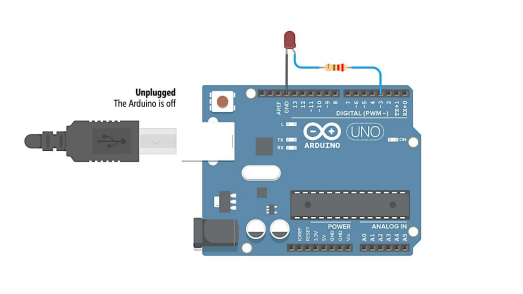
digital signal it has a discrete value. Here in the Arduino world a

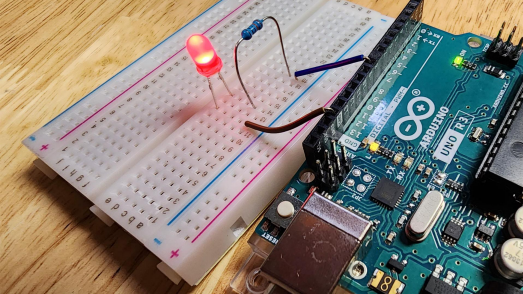
digital signal has only two options of values. The value may be ZERO or ONE. Zero represents GND voltage level where as ONE represents HIGH voltage level. In case of the Arduino HIGH Signal means +5V and Zero means GND voltage level.

If a digital signal is drawn from an Arduino It is called a digital

output. To see a digital output an LED is attached to an Arduino. If the LED glows then it is giving a HIGH voltage signal and vice-versa.

Circuit:

Picture of the circuit:

Connection :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.**  **No.** | **Component** | **Polarity** | **Connections** | |
|  |  |  | **Arduino** | **Breadboard** |
| **1** | **LED** | **Polarity**  **Matters** |  | **D25 – D26**  **(-) (+)** |
| **2** | **Resistor 1k** |  |  | **B26 – B29** |
| **3** | **Jumper** |  | **5V** | **(+)** |
| **4** | **Jumper** |  | **GND** | **(-)** |
| **5** | **Jumper** |  | **D2** | **E29** |
| **6** | **Jumper** |  |  | **(-) – A25** |

Arduino Code:

const int LED = 2;

void setup() {

pinMode(LED, OUTPUT);

}

void loop() {

digitalWrite(LED, HIGH);

delay(1000);

digitalWrite(LED, LOW);

delay(1000);

}

Output:

After successful uploading of the code, the execution of code makes the LED on for one second and off for another one second. This happens in an infinite loop. Thus the LED keeps blinking.

**Practical no. 2**

**Title: Fading LED**

**Aim: To fade an LED brightness using PWM**

**Apparatus: Arduino Uno, LED, 220 ohm resistor, breadboard, jumper wires Theory:**

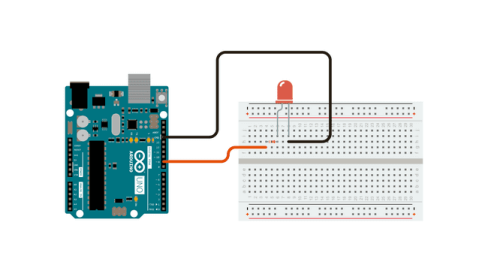
**Pulse Width Modulation (PWM) is a technique used to modulate the width of pulses in a pulse train. In Arduino, PWM can be used to control things like the brightness of an LED or speed of a motor by cycling the power on and off very quickly.**

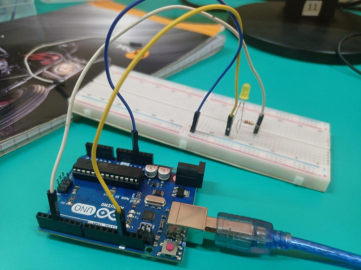
**The Arduino Uno has several PWM pins that can generate a PWM signal. By changing the duty cycle of the signal, which is the percentage of time the signal is high versus low, the brightness or power level can be varied.**

**For example, if the signal is high 50% of the time and low 50% of the time, the LED receiving this signal will be at half brightness. If the signal is high 90% and low 10%, the LED will be much brighter.**

**Using the analogWrite() function in Arduino, the duty cycle can be set between 0 (always off) to 255 (always on). This allows fading an LED up and down by slowly changing the analogWrite value.**

**Circuit:**

**Picture of the circuit:**

**Connection:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.**  **No.** | **Component** | **Polarity** | **Connections** | |
|  |  |  | **Arduino** | **Breadboard** |
| **1** | **LED** | **Polarity Matters** | **PWM Pin 3** | **B25-B26**  **(-) (+)** |
| **2** | **Resistor 220 ohm** |  |  | **B26 – B29** |
| **3** | **Jumper** |  | **5V** | **B30(+)** |
| **4** | **Jumper** |  | **GND** | **B31(-)** |
| **5** | **Jumper** |  | **PWM Pin3** | **E29** |

**Arduino Code:**

**int brightness = 0;**

**int fadeAmount = 5;**

**void setup() {**

**pinMode(3, OUTPUT);**

**}**

**void loop() {**

**analogWrite(3, brightness);**

**brightness = brightness + fadeAmount;**

**if (brightness <= 0 || brightness >= 255) {**

**fadeAmount = -fadeAmount;**

**}**

**delay(30);**

**}**

**Output:**

**After uploading the code, the brightness of the LED connected to Pin 3 will slowly fade up and down repeatedly. This is achieved by using the analogWrite() function to vary the duty cycle and PWM signal going to the LED. The fadeAmount variable controls how much the brightness changes each time through the loop.**

**Practical no:3**

**Title: Blinking LED using Switch Button**

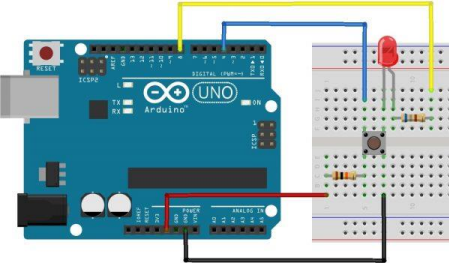
**Aim: To blink an LED using a push button switch as input**

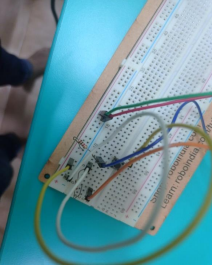
**Apparatus: Arduino Uno, LED, 220 ohm resistor, push button, 10k ohm resistor, breadboard, jumper wires**

**Theory: A push button switch provides a simple digital input to the Arduino. When the button is pressed, the pin reads HIGH. When released, it reads LOW. This can be used to control the blinking of an LED.**

**The built-in pull-up resistor is used so the pin reads HIGH when the button is released. This avoids floating values when the button is not pressed. A small delay after the button press avoids switch bouncing.**

**Circuit:**

**Picture of the circuit:**

****

**Connection:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.**  **No.** | **Component** | **Polarity** | **Connections** | |
|  |  |  | **Arduino** | **Breadboard** |
| **1** | **LED** | **Polarity**  **Matters** | **Pin 13** | **B25 – B26**  **(-) (+)** |
| **2** | **Resistor 220** |  |  | **B26 – B29** |
| **3** | **Push Button** |  | **Pin 7** | **B31** |
| **4** | **10k** |  | **Pin 7 to +5V** | **B31-B30** |

**Code:**

**int ledPin = 13;**

**int buttonPin = 7;**

**void setup() {**

**pinMode(ledPin, OUTPUT);**

**pinMode(buttonPin, INPUT\_PULLUP);**

**}**

**void loop() {**

**if(digitalRead(buttonPin) == LOW) {**

**digitalWrite(ledPin, HIGH);**

**delay(200); // debounce delay**

**digitalWrite(ledPin, LOW);**

**}**

**}**

**Output: Pressing the button will blink the LED once for a short period. The delay avoids multiple blinks from switch bouncing.**

**Practical no: 4**

**Title: Fading LED using 10k Potentiometer**

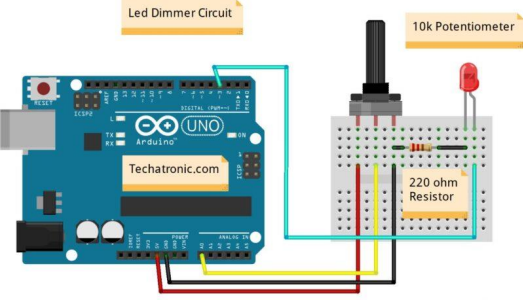
**Aim: To fade an LED based on an analog input from a potentiometer**

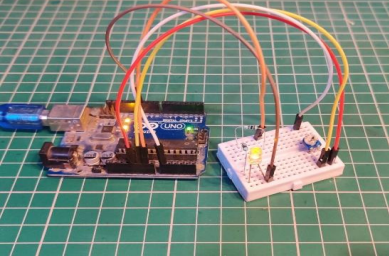
**Apparatus: Arduino Uno, LED, 220 ohm resistor, 10k potentiometer, breadboard, jumper wires**

**Theory: A potentiometer provides an analog voltage that can be read by an Arduino analog input pin. As the potentiometer knob is turned, the output voltage changes. This can be used to control the brightness of an LED.**

**The analogRead() function measures the voltage on a pin, returning a value between 0 to 1023 representing 0V to 5V. Mapping this range to 0-255 allows using it as the analogWrite value to fade an LED.**

**Circuit:**

**Picture of the circuit:**

****Connection :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.**  **No.** | **Component** | **Polarity** | **Connections** | |
|  |  |  | **Arduino** | **Breadboard** |
| **1** | **LED** | **Polarity**  **Matters** | **Pin 3 (PWM)** | **D25 – D26**  **(-) (+)** |
| **2** | **Resistor 220** |  |  | **B26 – B29** |
| **3** | **Potentiometer** |  | **Pin A0** | **E26** |
| **4** | **Jumper** |  | **Pin 2 to +5V** | **E30** |
| **5** | **Jumper** |  | **Pin 1 to GND** | **E31** |

Code:

int potPin = A0;

int ledPin = 3;

void setup() {

pinMode(ledPin, OUTPUT);

}

void loop() {

int potValue = analogRead(potPin);

int brightness = map(potValue, 0, 1023, 0, 255);

analogWrite(ledPin, brightness);

}

Output: As the potentiometer knob is turned, the LED brightness will change in response, fading up and down.

Practical no. : 5

Title: Distance Measurement using Ultrasonic Sensor

Aim: To measure distance using an ultrasonic sensor and Arduino

Apparatus: Arduino Uno, ultrasonic sensor, breadboard, jumper wires

Theory: Ultrasonic sensors work by sending out a high frequency sound pulse and then measuring the time it takes for the echo to reflect back. The Arduino provides a trigger pulse to generate the sound. The echo pin goes high to indicate when the echo is received back.

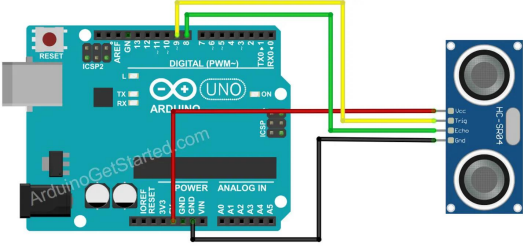
Using the speed of sound, the time between sending the pulse and receiving the echo can be used to calculate the distance to an object.

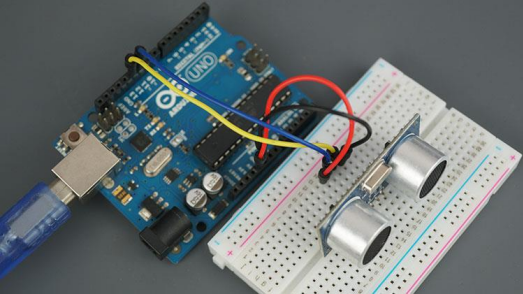
Speed of sound = 343 m/s

Distance = Time x Speed / 2

The sensor requires connection to 5V power and ground. A short 10us pulse on the trigger causes it to send the ultrasonic pulse. The echo pin then goes high until the echo is received back.

Circuit:

Picture of the circuit:

connection :

|  |  |  |  |
| --- | --- | --- | --- |
| **S.**  **No.** | **Component** | **Pins** | **Connections** |
|  |  |  | **Arduino-Breadboard** |
| **1** | **Ultrasonic**  **Sensor** | **Pin 7 - Trigger Pin 6 - Echo** | **Pin 7**  **Pin 6** |
| **2** | **Jumper Wires** | **5V, GND** | **Power rails** |

Code:

const int trigPin = 7;

const int echoPin = 6;

void setup() {

Serial.begin(9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

}

void loop() {

long duration, distance;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration/2) / 29.1;

Serial.print(distance);

Serial.println(" cm");

delay(100);

}

Output: The distance measured by the sensor is printed out to the serial monitor continually. The value changes based on proximity of objects.

Practical no. : 6

Title: Generating Melodies using Piezo Buzzer

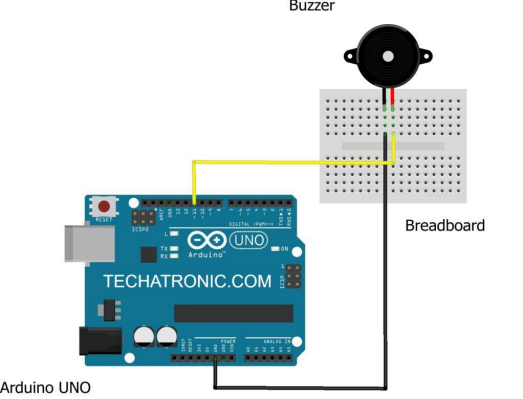
Aim: To generate musical melodies using a piezo buzzer and Arduino

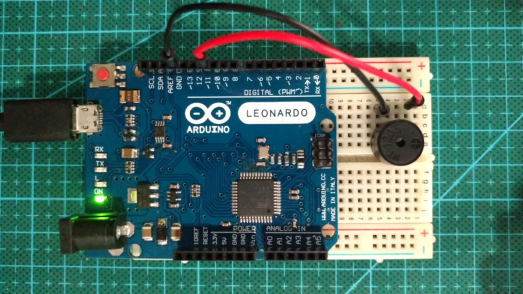
Apparatus: Arduino Uno, piezo buzzer, 220 ohm resistor, breadboard, jumper wires

Theory: A piezo buzzer can be used to generate sounds and musical notes by driving it with a square wave signal from the Arduino. The tone() function outputs a square wave on a pin at a certain frequency, corresponding to a musical note.

The duration of the note can also be specified to create musical rhythms. Notes can be strung together to form melodies. Different frequencies produce different notes - for example, 440 Hz generates an 'A' note.

Circuit:

Picture of the circuit :

connection :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.**  **No.** | **Component** | **Pins** | **Connections** | |
|  |  |  |  | **Arduino**  **Breadboard** |
| **1** | **Piezo Buzzer** | **Pin 8** |  | **Pin 8** |
| **2** | **Jumper** | **5V, GND** |  | **Power Pins** |

Code:

#define C4 262

#define D4 294

#define E4 330

#define G4 392

#define A4 440

void setup()

{

pinMode(8, OUTPUT);

}

void loop()

{

// Notes in the melody:

int melody[] = { A4, G4, E4, G4, A4, G4, E4, G4, C4};

// Note durations: 4 = quarter note, 8 = eighth note, etc.: int noteDurations[] = {4,4,4,4,4,4,4,4,2};

for (int i = 0; i < 9; i++) {

int noteDuration = 1000/noteDurations[i];

tone(8, melody[i],noteDuration);

int pauseBetweenNotes = noteDuration \* 1.30;

delay(pauseBetweenNotes);

noTone(8);

}

}

Output: A simple 9 note melody will play repeatedly on the piezo buzzer.

**Practical No.: 07**

**Title: Adjusting LED using LDR Sensor**

**Aim: To automatically adjust LED brightness based on ambient light levels sensed by an LDR. Apparatus: Arduino Uno, LED, 220 ohm resistor, LDR, 10K ohm resistor, breadboard, jumper wires. Theory:**

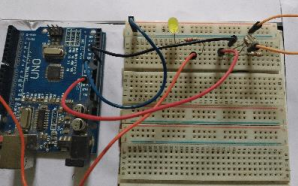
**A light dependent resistor (LDR) is a variable resistor whose value depends on the amount of light falling on its surface. In ambient light, the resistance of the LDR is low. In darkness, the resistance is high.**

**An LDR can be placed in a voltage divider circuit with a fixed pulldown resistor. As light varies, the voltage at the divider junction will vary based on the changing LDR resistance. This analog voltage can be read by an Arduino analog input.**

**The voltage can then be used directly or mapped to control the analogWrite() value of an LED, creating a LED brightness that automatically adjusts to ambient light levels.**

**Circuit:**

**Picture of the circuit:**

****

**Code:**

**const int ldrPin = A0;**

**const int ledPin = 3;**

**void setup() {**

**Serial.begin(9600);**

**pinMode(ledPin, OUTPUT);**

**}**

**void loop() {**

**int ldrStatus = analogRead(ldrPin);**

**Serial.print("LDR value: ");**

**Serial.println(ldrStatus);**

**int brightness = map(ldrStatus, 0, 1023, 255, 0);**

**analogWrite(ledPin, brightness);**

**delay(100);**

**}**

**Output: The brightness of the LED will automatically adjust based on the light falling on the LDR. More light makes the LED dimmer.**

**Experiment : 08**

**Title: Interfacing Temperature Sensor**

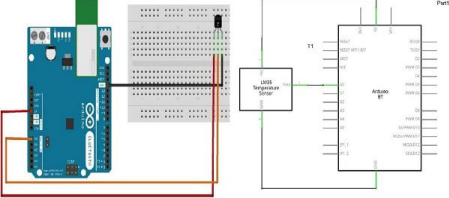
**Aim: To interface a temperature sensor with Arduino and display the temperature value on serial monitor.**

**Apparatus: Arduino Uno, Breadboard, Temperature sensor (LM35), Connecting wires Theory:**

**The LM35 temperature sensor provides analog voltage output proportional to temperature in degree Celsius. It can measure temperature from -55°C to 150°C. The sensor does not need any external calibration and provides an accuracy of ±1°C.**

**The analog output voltage from the LM35 sensor is connected to the analog input pin A0 of the Arduino. The Arduino reads this analog voltage, converts it into temperature value using the analog to digital converter (ADC), and displays it on the serial monitor.**

**Circuit:**

****

**Code:**

**const int tempSensor = A0; //LM35 connected to analog pin A0**

**void setup() {**

**Serial.begin(9600); //Initialize serial port**

**}**

**void loop() {**

**int sensorVoltage = analogRead(tempSensor); //Read analog voltage //Convert the analog voltage to temperature in degree Celsius**

**float tempC = (sensorVoltage/1024.0) \* 5 \* 100 ;**

**//Print temperature value on serial monitor**

**Serial.print("Temperature: ");**

**Serial.print(tempC);**

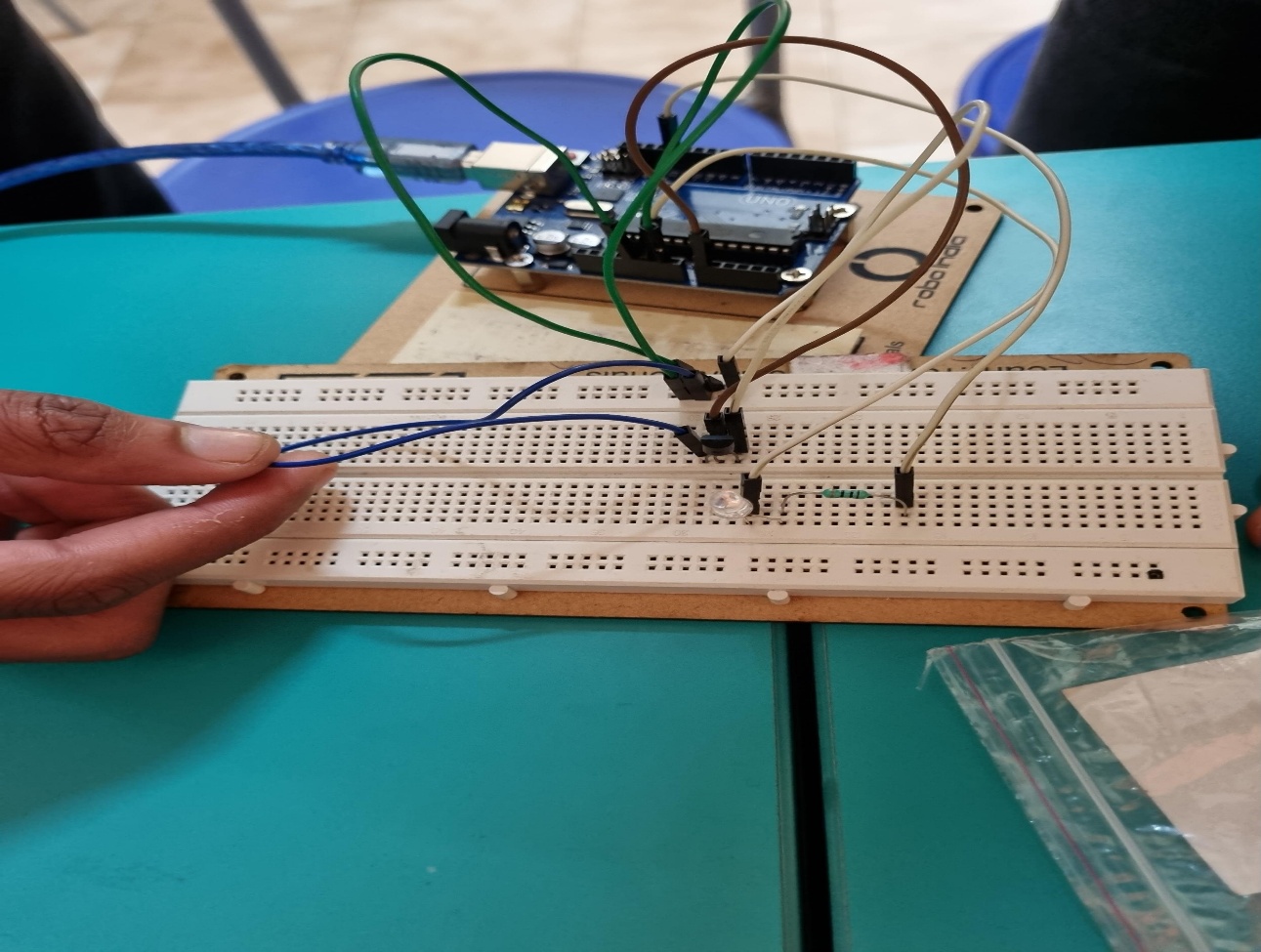
**Serial.println("°C");**

**delay(1000); //1 sec delay**

**}**

**Output**

**The measured temperature from the LM35 sensor is displayed on the serial monitor. The temperature reading gets updated continuously.**

****

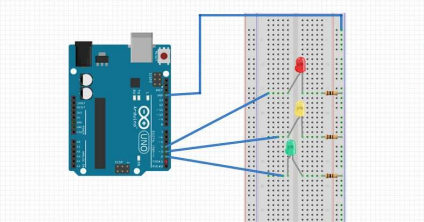
Traffic Light semulation

Aim: To create a traffic light controller that cycles through green, yellow and red LEDs.

Apparatus: Arduino Uno board, Red, yellow and green ,LEDs,220 ohm resistors (one for each LED),Breadboard, Jumper wires

Theory: The traffic light controller works by turning on the LEDs in sequence to mimic the operation of real traffic lights. The Arduino controls the timing and order of the LEDs illuminating.

Circuit:



Code:

// Pin definiƟons

const int redLed = 9;

const int yellowLed = 10; const int greenLed = 11;

void setup() {

// Set LED pins as Outputs pinMode(redLed, OUTPUT); pinMode(yellowLed, OUTPUT); pinMode(greenLed, OUTPUT); }

void loop() {

// Green light

digitalWrite(greenLed, HIGH); delay(5000); // Wait 5 seconds

// Yellow light

digitalWrite(greenLed, LOW); digitalWrite(yellowLed, HIGH); delay(2000); // Wait 2 seconds

// Red light

digitalWrite(yellowLed, LOW);

digitalWrite(redLed, HIGH);

delay(5000); // Wait 5 seconds

digitalWrite(redLed, LOW);

digitalWrite(greenLed, HIGH); // Back to green

delay(5000); // Wait 5 seconds

}

Output: The LEDs will illuminate in sequence - green for 5 secs, yellow for 2 secs, red for 5 secs, and repeat. This mimics a real traffic light changing colours.

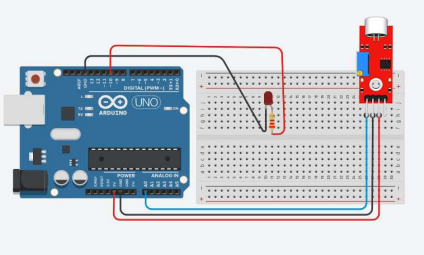
Blink LED ON/OFF using Clap Sound (Sound Sensor)

Aim: To blink an LED on and off in response to detected clap sounds using a sound sensor module.

Apparatus: Arduino Uno board, LEDs, 220 ohm ,Breadboard, Jumper wires, Sound sensor module

Theory: The sound sensor module detects loud sounds and triggers the Arduino. When a clap sound is detected, the Arduino toggles the LED state from on to off or off to on. This creates a blinking effect in response to clapping.

Circuit:



Code:

const int ledPin = 13; // LED connected to digital pin 13

const int soundSensor = A0; // Sound sensor connected to analog pin A0

void setup() {

pinMode(ledPin, OUTPUT); // Set LED as output }

void loop() {

int soundState = digitalRead(soundSensor); // Read sound sensor

if (soundState == 1) { // If clap detected

digitalWrite(ledPin, !digitalRead(ledPin)); // Toggle LED }

delay(10); // Delay

}

Output The LED remains off initially. When a loud clap sound is detected, the LED blinks on. Clapping again blinks the LED off. The process repeats for each clap..

Seven segment display(cathod)

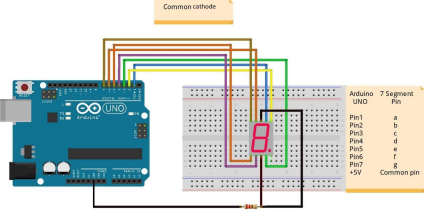
Aim: To print numbers from 0 to 9 on a single seven segment display.

Apparatus: Arduino UNO, seven segment display, Male to Female Jumper Wires, Breadboard

Theory:

A seven segment display is a simple displaying device that uses 8 LEDs to display decimal numerals. It is generally used in digital clocks, calculators, electronic meters, and other devices that displays numerical information. A seven segment display has 8 LEDs in it. Each LED is controlled through specific pin. These can be two types common anode and common cathode. In common cathode type the cathode is common for all 8 LEDs and in common anode type the anode is common for all 8 LEDs. A seven segment display has 10 pin interface, 2 pins are for commons and 8 pins are for each LED. Both common pins are internally shorted

Circuit: Cathod



Code:

int

Digit[10]={B01111111,B00001101,B10110111,B10011111,B11001101,B11011011,B11111011,B0000 1111,B11111111,B11011111}; // commmon cathode

void setup() {

DDRD = B11111110;

}

void loop() {

for(int i=0;i<10;i++)

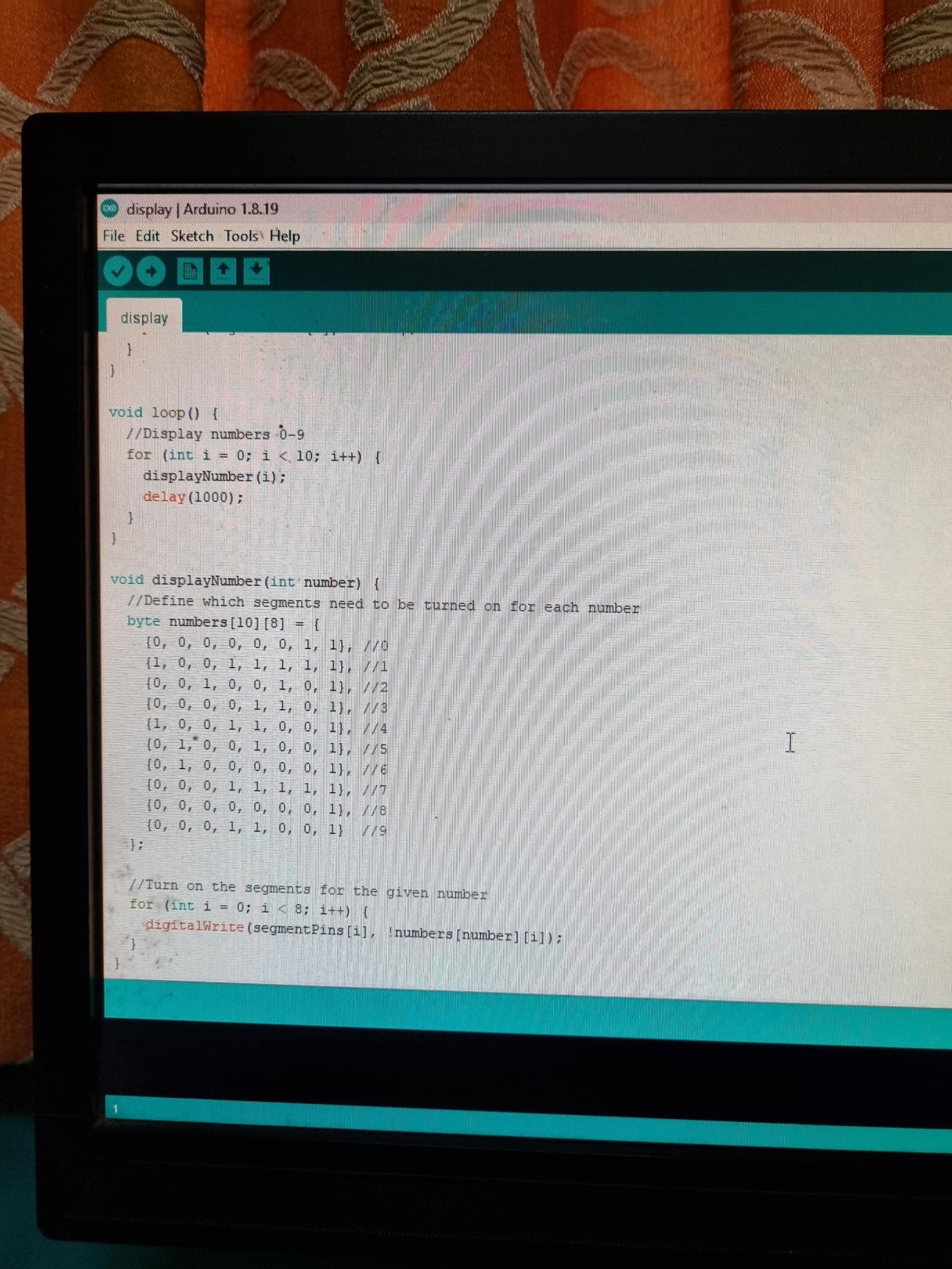
{

PORTD = digit[i];

delay(1000);

}

}



Output: After uploading the code, The single seven segment display will print numbers from 0 to 9

Seven segment display

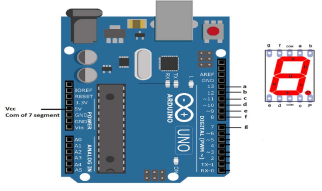
Aim: To print numbers from 0 to 9 on a single seven segment display.

Apparatus: Arduino UNO, seven segment display, Male to Female Jumper Wires, Breadboard

Theory:

A seven segment display is a simple displaying device that uses 8 LEDs to display decimal numerals. It is generally used in digital clocks, calculators, electronic meters, and other devices that displays numerical information. A seven segment display has 8 LEDs in it. Each LED is controlled through specific pin. These can be two types common anode and common cathode. In common cathode type the cathode is common for all 8 LEDs and in common anode type the anode is common for all 8 LEDs. A seven segment display has 10 pin interface, 2 pins are for commons and 8 pins are for each LED. Both common pins are internally shorted

Circuit: Anode



Code: int a = 2;

int b = 3;

int c = 4;

int d = 5;

int e = 6;

int f = 7;

int g = 8;

void setup() {

pinMode(a, OUTPUT); pinMode(b, OUTPUT); pinMode(c, OUTPUT); pinMode(d, OUTPUT); pinMode(e, OUTPUT); pinMode(f, OUTPUT); pinMode(g, OUTPUT); }

void loop() {

// G

digitalWrite(a, HIGH);

digitalWrite(b, LOW); digitalWrite(c, LOW); digitalWrite(d, HIGH); digitalWrite(e, HIGH); digitalWrite(f, HIGH); digitalWrite(g, HIGH); delay(1000);

// O

digitalWrite(a, LOW); digitalWrite(b, LOW); digitalWrite(c, HIGH); digitalWrite(d, LOW); digitalWrite(e, LOW); digitalWrite(f, HIGH); digitalWrite(g, LOW); delay(1000);

// O

digitalWrite(a, LOW); digitalWrite(b, LOW); digitalWrite(c, HIGH);

digitalWrite(d, LOW); digitalWrite(e, LOW); digitalWrite(f, HIGH); digitalWrite(g, LOW); delay(1000);

// D

digitalWrite(a, HIGH); digitalWrite(b, LOW); digitalWrite(c, LOW); digitalWrite(d, HIGH);

digitalWrite(e, HIGH); digitalWrite(f, LOW); digitalWrite(g, LOW); delay(1000);

// M

digitalWrite(a, LOW); digitalWrite(b, HIGH); digitalWrite(c, HIGH); digitalWrite(d, LOW); digitalWrite(e, HIGH);

digitalWrite(f, LOW); digitalWrite(g, LOW); delay(1000);

// O

digitalWrite(a, LOW); digitalWrite(b, LOW); digitalWrite(c, HIGH); digitalWrite(d, LOW); digitalWrite(e, LOW); digitalWrite(f, HIGH); digitalWrite(g, LOW); delay(1000);

// R

digitalWrite(a, HIGH); digitalWrite(b, LOW); digitalWrite(c, LOW); digitalWrite(d, HIGH);

digitalWrite(e, HIGH); digitalWrite(f, LOW); digitalWrite(g, LOW);

delay(1000);

// N

digitalWrite(a, HIGH); digitalWrite(b, LOW); digitalWrite(c, LOW); digitalWrite(d, LOW); digitalWrite(e, HIGH); digitalWrite(f, LOW);

digitalWrite(g, LOW); delay(1000);

// I

digitalWrite(a, LOW); digitalWrite(b, LOW); digitalWrite(c, HIGH); digitalWrite(d, HIGH); digitalWrite(e, HIGH); digitalWrite(f, HIGH); digitalWrite(g, HIGH); delay(1000);

// N

digitalWrite(a, HIGH); digitalWrite(b, LOW); digitalWrite(c, LOW); digitalWrite(d, LOW); digitalWrite(e, HIGH); digitalWrite(f, LOW);

digitalWrite(g, LOW); delay(1000);

// G

digitalWrite(a, HIGH); digitalWrite(b, LOW); digitalWrite(c, LOW); digitalWrite(d, HIGH);

digitalWrite(e, HIGH); digitalWrite(f, HIGH); digitalWrite(g, HIGH); delay(1000);

}

Output: After uploading the code, The single seven segment display will print GOOd MORNING LATTER BY LATTER