VU VISHWAKARMA UNIVERSITY

Maximising Human Potential

Practical Lab Book

MCA 2025-2026 SEM I

Subject – Internet Of Things

Submitted by-

SRN:31242488

Tushar Betala

Guided By-

Dr. Bharati Pandurang Kawade

INDEX -1

Practical 1

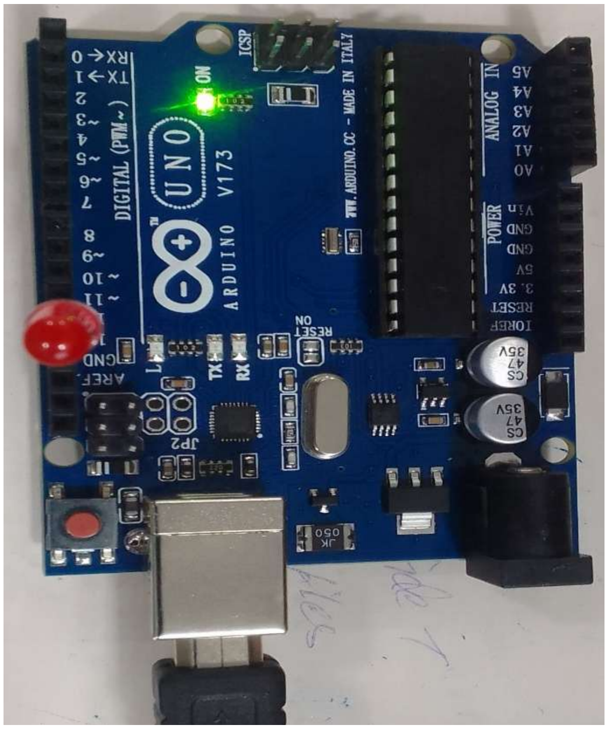
1. To interface Light Emitting Diode(LED)- Blinking LED.

   Code:

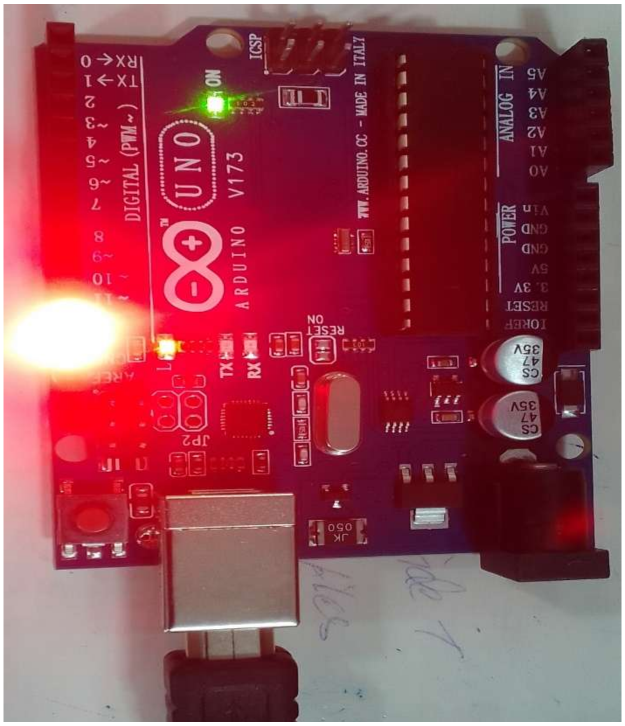
```
const int ledPin = 13; // Pin connected to the LED
void setup() {
      pinMode(ledPin, OUTPUT); // Set the LED pin as an output
}
void loop() {
   digitalWrite(ledPin, HIGH); // Turn on the LED
   delay(1000); // Wait for 1 second
  digitalWrite(ledPin, LOW); // Turn off the LED
  delay(1000); // Wait for 1 second
}
```

   Output :

Turn Off the LED



Turn On the LED



2. To interface Button and LED – LED blinking when button is pressed.

Code:

```
  const int buttonPin = 2; // Pin connected to the button
  const int ledPin = 13; // Pin connected to the LED
  int buttonState = 0; // Variable to store the button state
  void setup() {
  pinMode(ledPin, OUTPUT); // Set the LED pin as an output
  pinMode(buttonPin, INPUT); // Set the button pin as an input
 }
void loop() {
 buttonState = digitalRead(buttonPin); // Read the state of the button (HIGH or LOW)
 // Check if the button is pressed (buttonState is LOW)
 if (buttonState == LOW) {
 digitalWrite(ledPin, HIGH); // Turn on the LED
 delay(500); // Wait for 500 milliseconds (0.5 seconds)
 digitalWrite(ledPin, LOW); // Turn off the LED
 delay(500); // Wait for another 500 milliseconds
 }
 }
```

Output :

Turn Off the LED

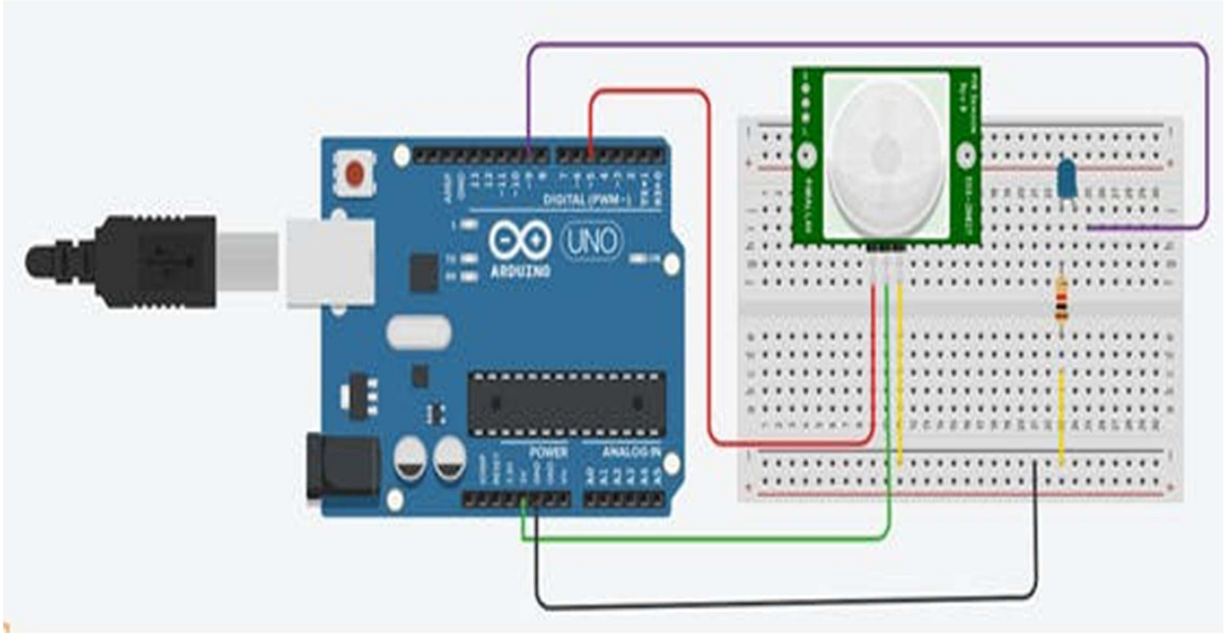
Turn On the LED

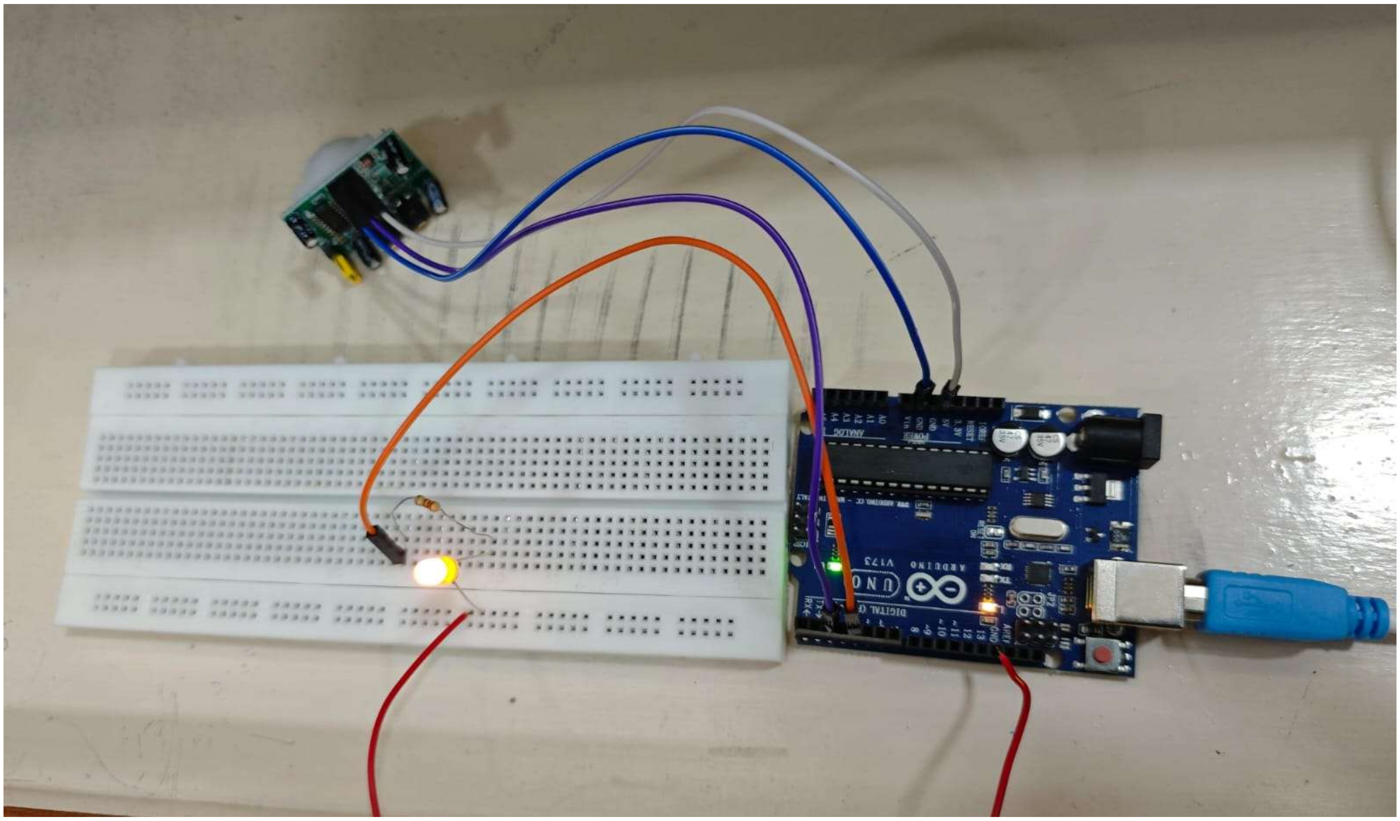
3. To interface a PIR Sensor with Arduino to Drive an LED Indicator.

Code:

```
  int pirPin = 2;       // PIR sensor OUT pin
  int ledPin = 4;      // LED pin
  int pirState=0;
  void setup() {
     pinMode(pirPin, INPUT);   // Set PIR pin as input
     pinMode(ledPin, OUTPUT);  // Set LED pin as output
     Serial.begin(9600);       // Start serial communication
 }

 void loop() {
    int pirState = digitalRead(pirPin);  // Read PIR state

    if (pirState == HIGH) {
          digitalWrite(ledPin, HIGH);  // Turn LED on
          Serial.println("Motion Detected!");
} else {
  digitalWrite(ledPin, LOW);   // Turn LED off
}

delay(1000);  // Small delay to stabilize reading
}
```

Output :



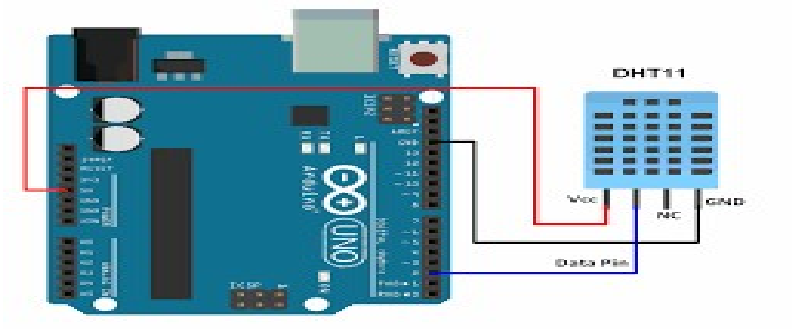


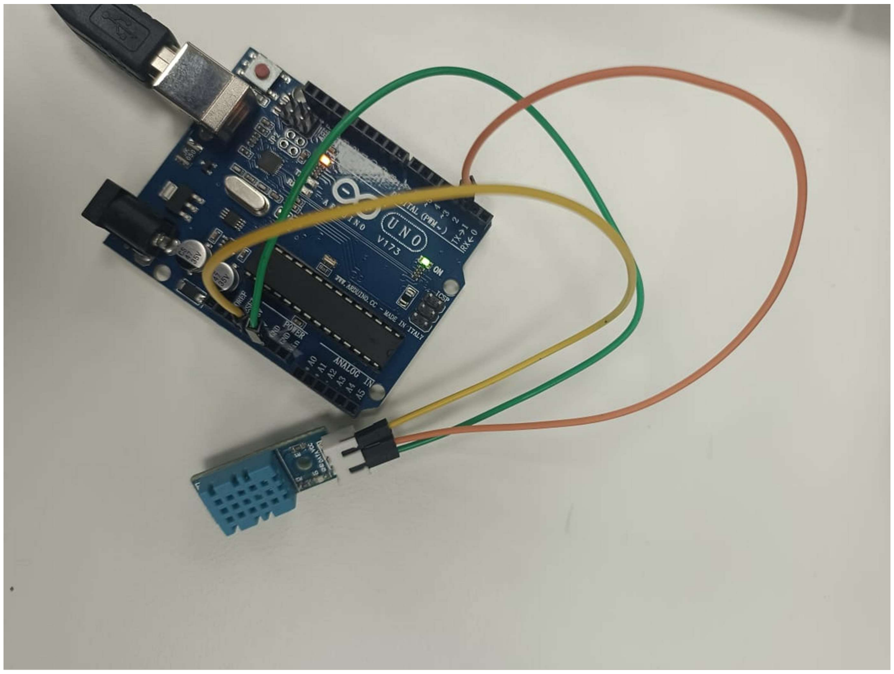
4. To use the DHT11 temperature & humidity sensor with Arduino.

Code:

```
#include "DHT.h"   // Include DHT library
#define DHTPIN 2     // Pin connected to DATA
#define DHTTYPE DHT11   // Define sensor type (DHT11 or DHT22)
DHT dht(DHTPIN, DHTTYPE);
void setup() {
Serial.begin(9600);
dht.begin();
Serial.println("DHT11 Sensor Test");
}
void loop() {
// Wait a few seconds between measurements
delay(2000);
float h = dht.readHumidity();
float t = dht.readTemperature(); // Celsius
// Check if reading failed
if (isnan(h) || isnan(t)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %\t");
Serial.print("Temperature: ");
Serial.print(t);
Serial.println(" °C");
}
```

Output :


DHT11
Vcc
NC
GND
Data Pin

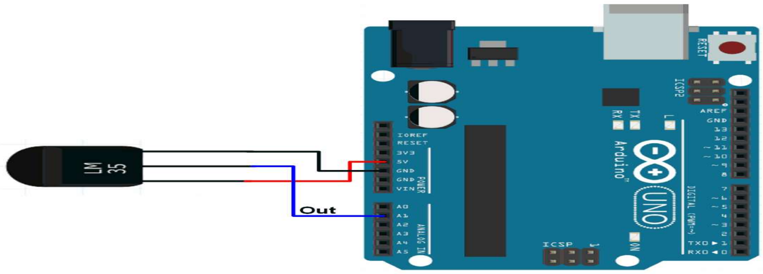


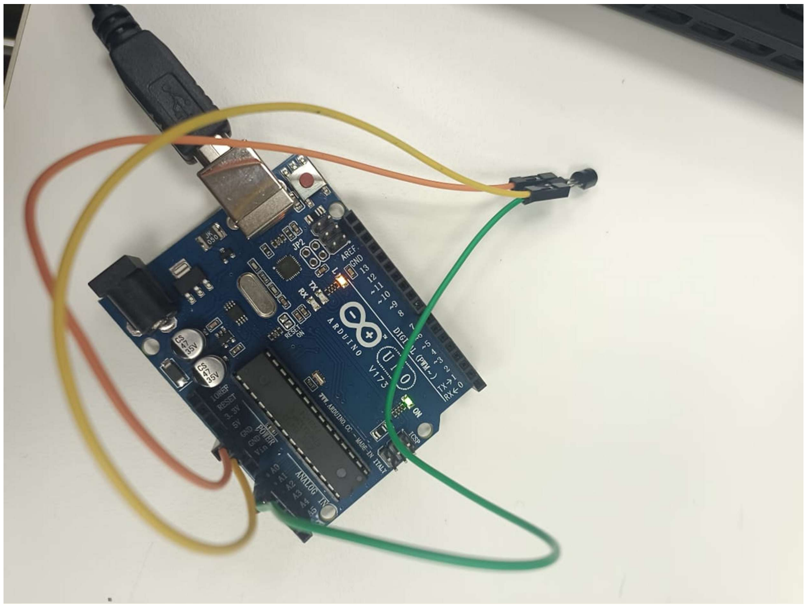
5. To interface the LM35 temperature sensor with Arduino.

Code:

```
int sensorPin = A0;   // LM35 output connected to A0
float sensorValue;       // to store ADC value
float temperature;       // calculated temperature
void setup() {
Serial.begin(9600);
}
void loop() {
sensorValue = analogRead(sensorPin);   // read analog value (0-1023)
// Convert to voltage (Arduino is 5V with 10-bit ADC)
float voltage = sensorValue * (5.0 / 1023.0);
// LM35 gives 10mV per °C → 0.01V = 1°C
temperature = voltage * 100;
Serial.print("Temperature: ");
Serial.print(temperature);
Serial.println(" °C");
delay(1000); // read once per second
}
```

Output :


LM 35
Out
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
RESET
ICSP2
AREF
GND
TX
RX
L
Arduino
UNO
DIGITAL (PWM~)
TX0
RX0
ICSP
ON



6. To interface the Traffic Signal Simulation with Arduino.

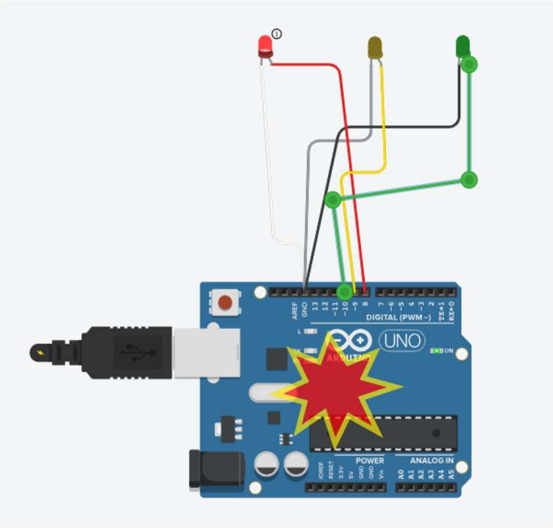
Code:

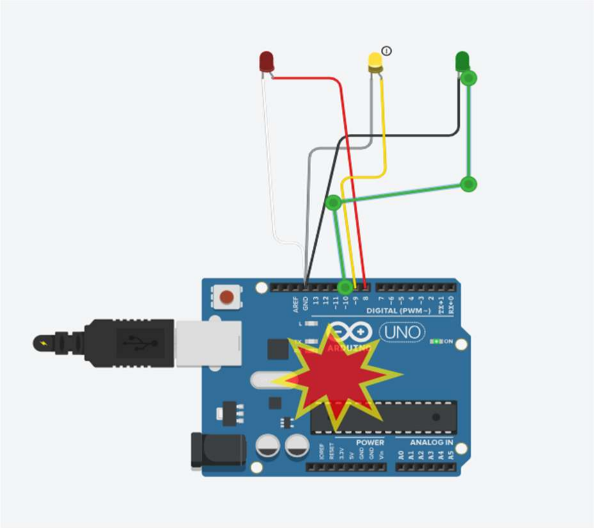
```
int redLed = 8;
int yellowLed = 9;
int greenLed = 10;

void setup() {
  pinMode(redLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
}

void loop() {
  // Red ON (Stop)
  digitalWrite(redLed, HIGH);
  digitalWrite(yellowLed, LOW);
  digitalWrite(greenLed, LOW);
  delay(5000);  // 5 seconds

  // Green ON (Go)
  digitalWrite(redLed, LOW);
  digitalWrite(yellowLed, LOW);
  digitalWrite(greenLed, HIGH);
  delay(5000);  // 5 seconds

  // Yellow ON (Caution)
  digitalWrite(redLed, LOW);
  digitalWrite(yellowLed, HIGH);
  digitalWrite(greenLed, LOW);
  delay(2000);  // 2 seconds
}
```

Output :

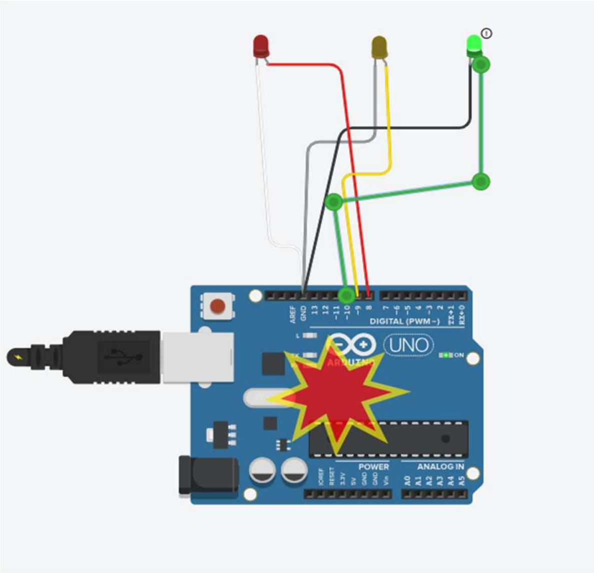
RED

YELLOW





GREEN

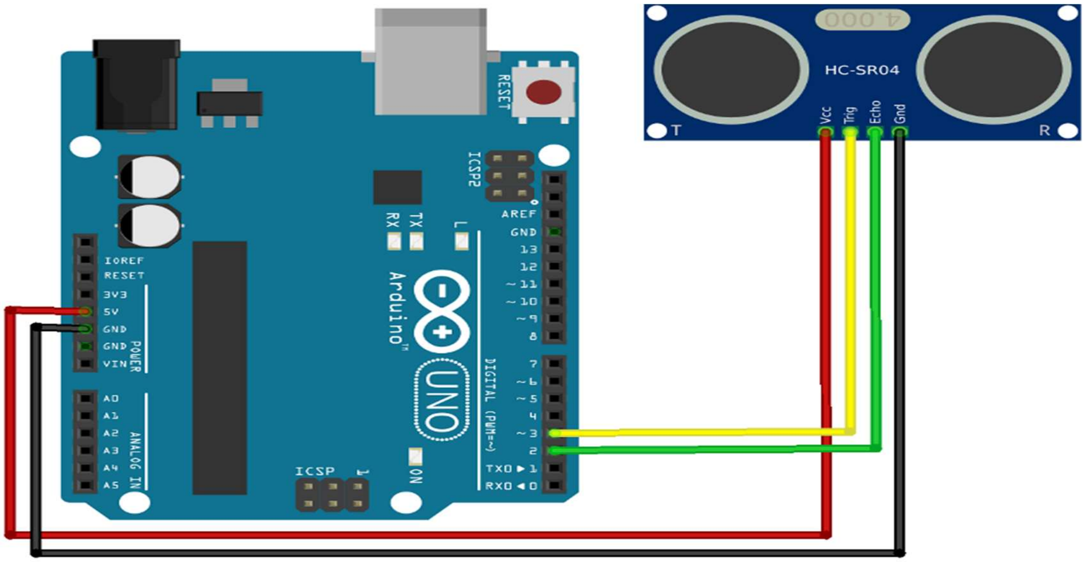


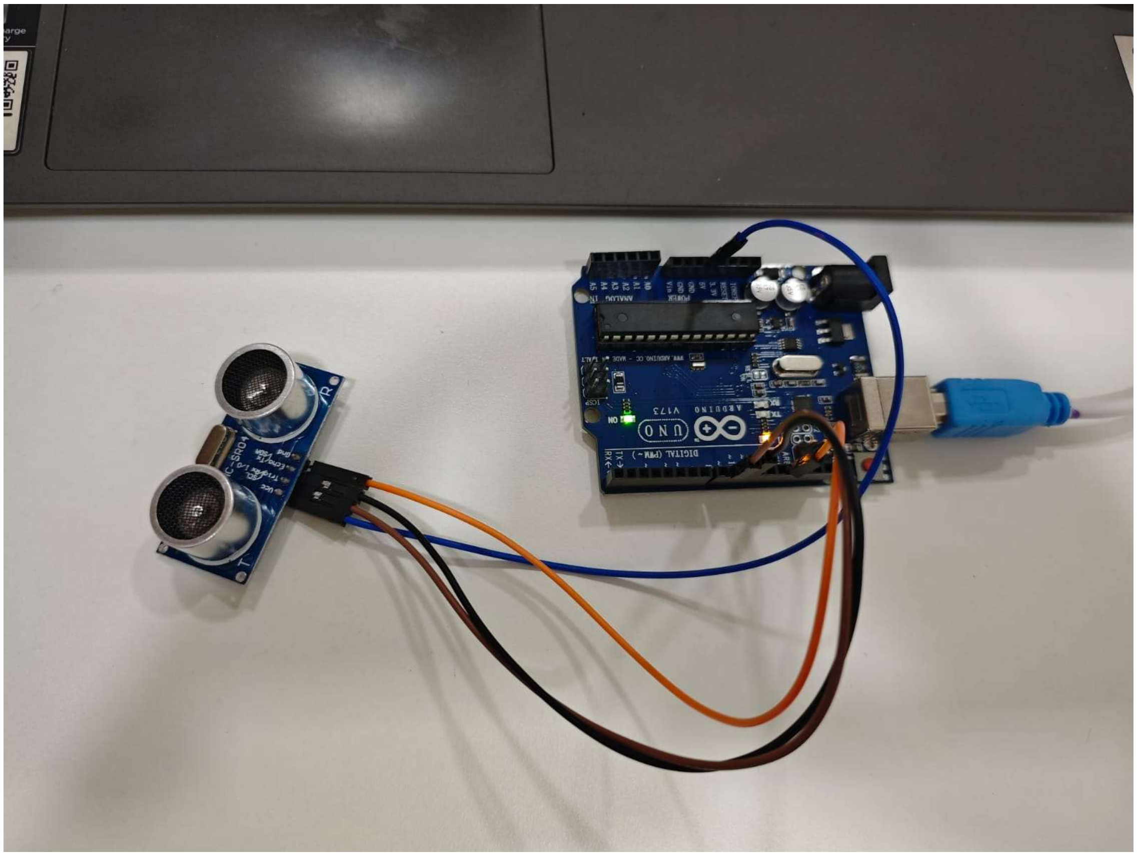
7. To interface the Ultrasonic Sensor with Arduino.

Code:

```
// HC-SR04 simple example (Trig=9, Echo=10)
const int trigPin = 9;
const int echoPin = 10;
unsigned long duration;
float distanceCm;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // Make sure Trig is LOW
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Send 10 microsecond pulse to trigger
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the time for echo (timeout 30000 us ~ 5 m)
  duration = pulseIn(echoPin, HIGH, 30000);

  if (duration == 0) {
    Serial.println("Out of range");
  } else {
    // distance in cm: duration (us) * 0.01715
    distanceCm = duration * 0.01715;
    Serial.print("Distance: ");
    Serial.print(distanceCm);
    Serial.println(" cm");
  }

  delay(200); // 5 readings/second
}
```

Output :


HC-SR04
4.000
Vcc
Trig
Echo
Gnd
T
R
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX ▶ 1
RX ◀ 0
DIGITAL (PWM=~)
RX
TX
L
Arduino
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN

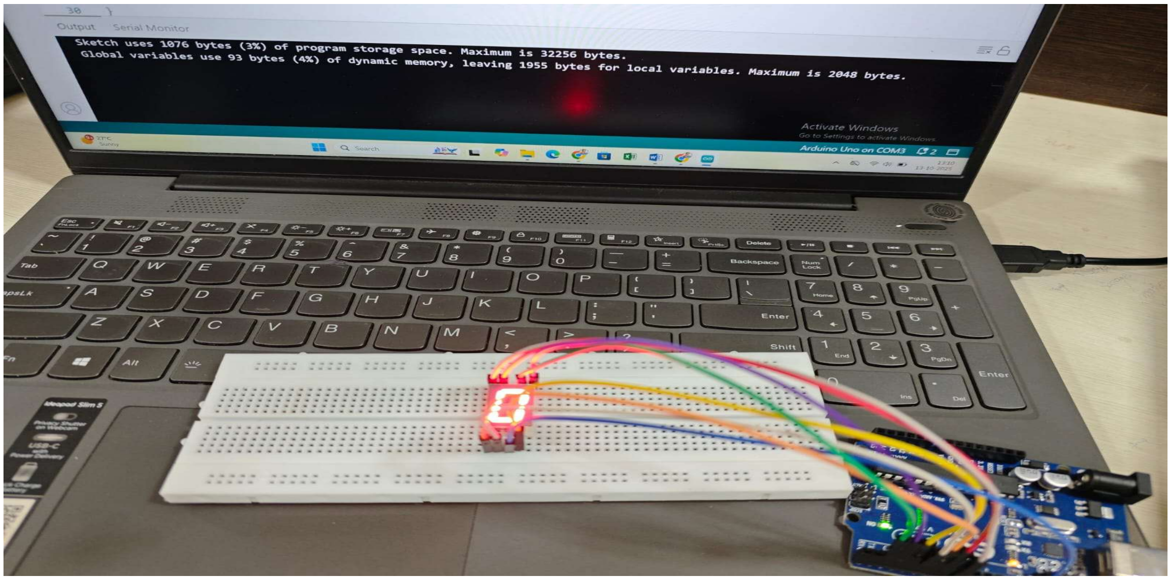


8. To interface 7 segments step by step in Arduino.

<u>Code:</u>

```
// Pin mapping for segments a-g
int segPins[7] = {2, 3, 4, 5, 6, 7, 8};

// Digit patterns for 0-9 (1 = ON, 0 = OFF)
byte digits[10][7] = {
  {1,1,1,1,1,1,0}, // 0
  {0,1,1,0,0,0,0}, // 1
  {1,1,0,1,1,0,1}, // 2
  {1,1,1,1,0,0,1}, // 3
  {0,1,1,0,0,1,1}, // 4
  {1,0,1,1,0,1,1}, // 5
  {1,0,1,1,1,1,1}, // 6
  {1,1,1,0,0,0,0}, // 7
  {1,1,1,1,1,1,1}, // 8
  {1,1,1,1,0,1,1}  // 9
};

void setup() {
  for (int i = 0; i < 7; i++) {
    pinMode(segPins[i], OUTPUT);
  }
}

void loop() {
  for (int num = 0; num < 10; num++) {
    showDigit(num);
    delay(1000); // 1 second delay
  }
}

void showDigit(int num) {
  for (int i = 0; i < 7; i++) {
    digitalWrite(segPins[i], digits[num][i]);
  }
}
```

Output :



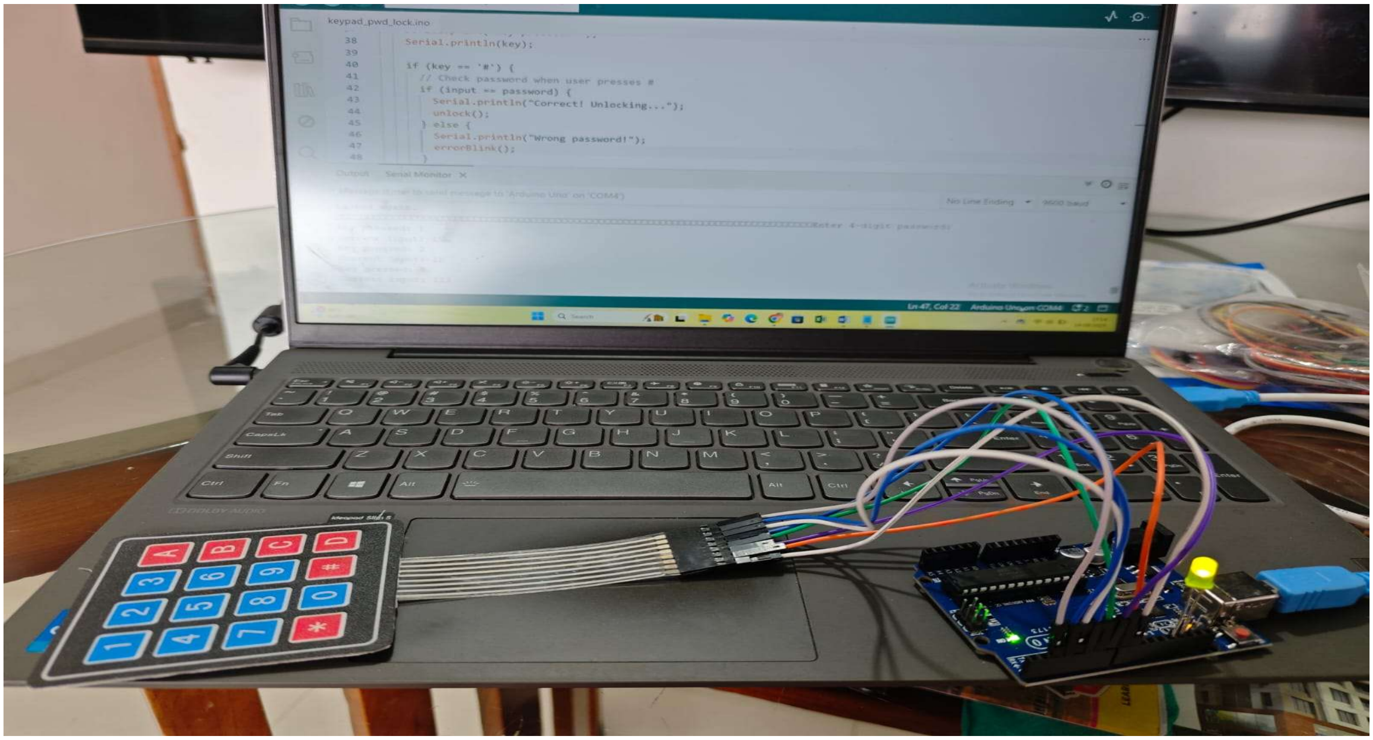
```
Sketch uses 1076 bytes (3%) of program storage space. Maximum is 32256 bytes.
 Global variables use 93 bytes (4%) of dynamic memory, leaving 1955 bytes for local variables. Maximum is 2048 bytes.
```

9. To interface a 4×4 matrix keypad with Arduino and develop a program to detect and display the pressed key using the serial monitor.

Code:

```
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

// change these pin numbers to match your wiring
byte rowPins[ROWS] = {9, 8, 7, 6};    // R0, R1, R2, R3
byte colPins[COLS] = {5, 4, 3, 2};    // C0, C1, C2, C3

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup() {
  Serial.begin(9600);
}

void loop() {
  char k = keypad.getKey();
  if (k) {
    Serial.print("Key: ");
    Serial.println(k);
  }
}
```

Output :



keypad_pwd_lock.ino

```
    Serial.println(key);

    if (key == '#') {
      // Check password when user presses #
      if (input == password) {
        Serial.println("Correct! Unlocking...");
        unlock();
      } else {
        Serial.println("Wrong password!");
        errorBlink();
      }
```

Serial Monitor

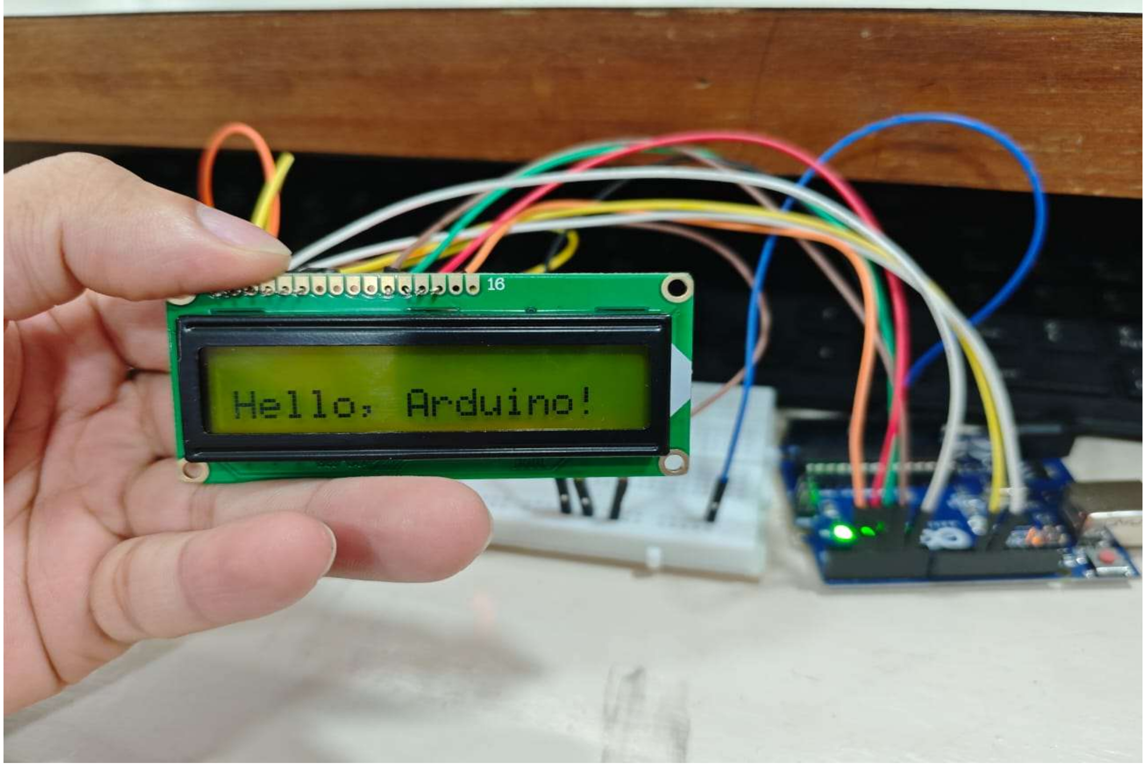
```
Enter 4-digit password:
```

10. To interface an LCD display with Arduino.

Code:

```
#include <LiquidCrystal.h>;

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2, ct=9;
LiquidCrystal mylcd(rs, en, d4, d5, d6, d7);

void setup() {
analogWrite(ct,50);
//ct- this sets the brightness level. Sets PWM brightness for the LCD backlight (50/255 ≈ 20% brightness).
mylcd.begin(16, 2);  // Initializes the LCD with 16 columns and 2 rows.
mylcd.print("Hello, Arduino!");   // Displays the text “Welcome” on the first line of the LCD.

}

void loop()
{
}
```

Output :



```
Hello, Arduino!
```