SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES

(Autonomous)

23ESC352L - TINKERING LAB

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

(Accredited by NBA)

III B.TECH - V SEMESTER

Prepared by

Dr.S.Sugumaran,

Professor/ECE,SITAMS

Ex-1: Parallel and Series Circuits using Bread-board

Aim: To construct and study the behavior of series and parallel circuits on a breadboard and measure voltage and current across each component and compare with theoretical values.

Apparatus Requirement :

S.No	Component	Specification	Quantity
1	Resistor	1K, 680 ohm	2
2	LED	-	2
3	RPS	-	1
4	Bread-board	-	1
5	Connecting Wire	-	As required
6	Multimeter	-	1

Theory:

Sereis Connection:

A circuit is said to be connected in series when the same current flows through all the components in the circuit. In such circuits, the current has only one path. This is nothing but a series of multiple tiny bulbs connected in series. If one bulb fuses, all the bulbs in the series do not light up.

Parallel Connection:

A circuit is said to be parallel when the electric current has multiple paths to flow through. The components that are a part of the parallel circuits will have a constant voltage across all ends.

Proceduce:

Series Connection:

1. Insert **R1, R2** in series on the breadboard by connecting the end of one resistor to the start of the next.
2. Connect the free ends to the **positive** and **negative terminals** of the power supply.
3. Use a **multimeter** to measure:
4. Voltage across each resistor.
5. Current through the circuit (same at all points in series).
6. Note the total resistance by **Rt = R1 + R2 + R3**.
7. Verify **Ohm's Law: I = V / Rt**.

Parallel Connection:

1. Connect all resistors in **parallel** on the breadboard, i.e., all one side connected to positive rail, other side to negative rail.
2. Connect to the **power supply**.
3. Measure:
    a. Voltage across each resistor (should be **same**).
    b. Current through each branch using multimeter (insert in series with each resistor).
4. Calculate total current:
    a. $\mathbf{I_{total} = I_1 + I_2 + I_3}$
5. Compute effective resistance using:
    a. **1/Rt = 1/R1 + 1/R2**

Circuit Diagram:

Series Connection

Parallel Connection:

Observation:

Connection	Resistance	Voltage Across	Current	Total Volt/Current
Series	**R1**		**Same**	
	R2			
Parallel	**R1**	**Same**		
	R2			

Result:

1. In series, current remains same, voltage divides.
2. In parallel, voltage remains same, current divides.

Exp-2: Traffic Light Circuit with Switch

Aim: To create a traffic light model using LED bulb, 9v battery, 330v resistor,push button, connecting wires and a breadboard.

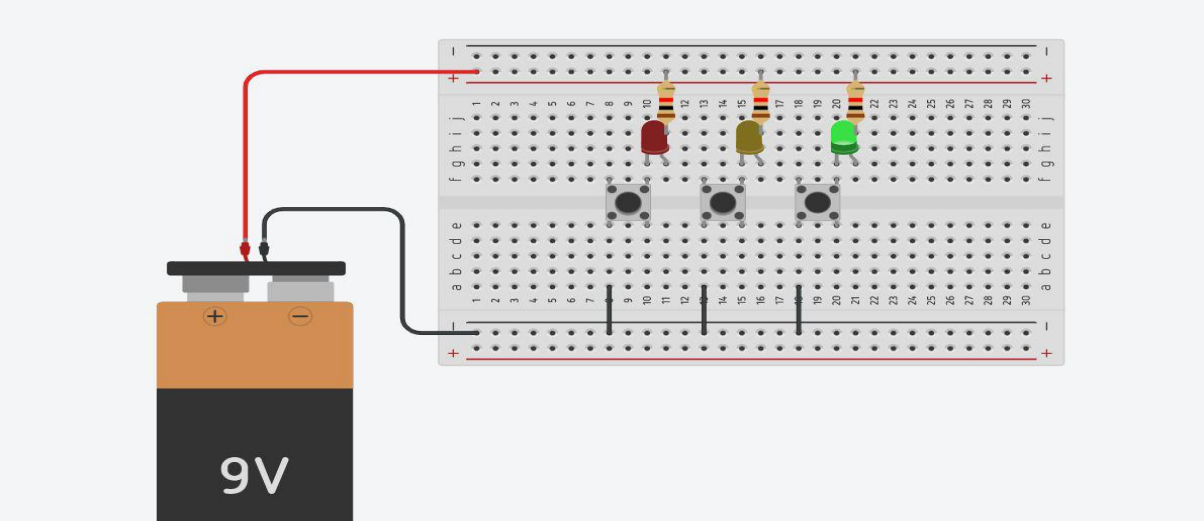
Apparatus Requirement :

S.No	Component	Specification	Quantity
1	Resistor	1K	3
2	LED	-	3
3	Push button Switch	-	3
4	Bread-board	-	1
5	Connecting Wire	-	As required

Procedure:

1. Insert a LED into the breadboard
2. Insert the resistor into the breadboard and connect it to the positive terminal of LED
3. ADD the push button and connect it to the negative side of LED like as shown in the picture.
4. Connect the 9V batter to the circuit.
5. Connect the pushbutton and neutral of the battery by using the connecting wires.
6. To turn on red light, press the push button that is connected to the Red LED.
7. To turn on yellow light, press the push button that is connected to the Yellow LED
8. To turn on green light press the push button that connected to the Green LED.

Circuit Connection:



Result: Thus the traffic light model using LED bulb, 9v battery, 330v resistor,push button, connecting wires and a breadboard is testes successfully.

Ex:3 Smart Dustbin Using Ultrasonic Sensor

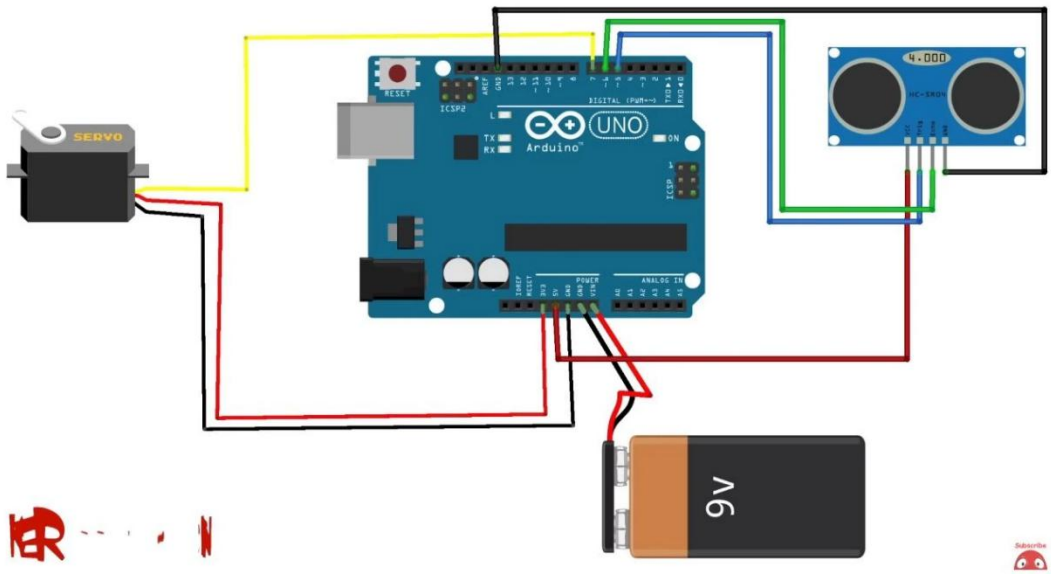
Aim

To design and test a system that detects a nearby object using an ultrasonic sensor (HC-SR04) and automatically opens/closes a lid using a servo motor.

Apparatus / Components Required

- Arduino Uno (or compatible)
- Ultrasonic sensor **HC-SR04**
- Servo motor (e.g., SG90/MG90S)
- 1 × LED (optional status indicator)
- Resistors: 220 Ω for LED (optional)
- Jumper wires and breadboard
- USB cable for programming/power
- External 5 V supply for servo (recommended)
  (common GND with Arduino)
- Laptop/PC with Arduino IDE

Circuit Connections



Program:

```
#include <Servo.h>  // Include Servo library
Servo servo;
int trigPin = 5;
int echoPin = 6;
int servoPin = 7;
int led = 10;
long duration, dist, average;
long aver[3]; // array for average

void setup() {
 Serial.begin(9600);
 servo.attach(servoPin);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 pinMode(led, OUTPUT);
 servo.write(0);  // close cap on power on
 delay(100);
 servo.detach();
}

void measure() {
 digitalWrite(led, HIGH); // turn on LED
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);  // send utrasonic pulse
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
```

```
  duration = pulseIn(echoPin, HIGH);  //Wait for echo
  dist = (duration / 2) / 29.1; // distance in cm
}

void loop() {
  for (int i = 0; i < 3; i++) {     //take 3 reading
    measure();
    aver[i] = dist;
    delay(10);  // small gap between the reading
  }
  dist = (aver[0] + aver[1] + aver[2]) / 3; //Average

  if (dist < 50) { // open lid if object is within 50cm
    servo.attach(servoPin); //connect servo to pin 7
    delay(1);
    servo.write(0);    // open
    delay(3000);      // keep open for 3 seconds
    servo.write(150);  // close
    delay(1000);
    servo.detach(); // disconnect servo to avoid jitter
  }

  Serial.println(dist); //show distance in cm
}
```

Theory (Principle)

The HC-SR04 emits an ultrasonic pulse and measures the time for the echo to return. Distance , d (cm) ≈ duration (μs) / (2×29.1)

When d< threshold (50 cm), the microcontroller commands the servo to rotate to the "open" angle and then back to "close."

Procedure

1. **Assemble the circuit** as per the connections above; double-check power and grounds.
2. **Power the servo** preferably from a stable 5 V source; connect servo GND to Arduino GND.
3. **Open Arduino IDE**, select the correct **Board** and **Port** (Tools → Board/Port).
4. **Load the sketch** and click **Upload**.
5. Open **Serial Monitor** at **9600 baud** to observe distance readings.
6. **Test with an object** (your hand) moving toward the sensor:
    - Observe distance values decrease.
    - When distance < 50 cm, the servo should rotate to open (e.g., 0°), hold ~3 s, then rotate to close (e.g., 150°).
7. **Adjust threshold** (50 cm), servo angles (servo.write() values), and delays as needed for your mechanism.

Result

The system successfully detected objects closer than the set threshold (50 cm) and actuated the servo to open and close the lid automatically. Distance readings were visible on the Serial Monitor, and the behavior matched the programmed logic.

Ex:4 Temperature& Humidity Monitoring System using DHT 11 and LCD

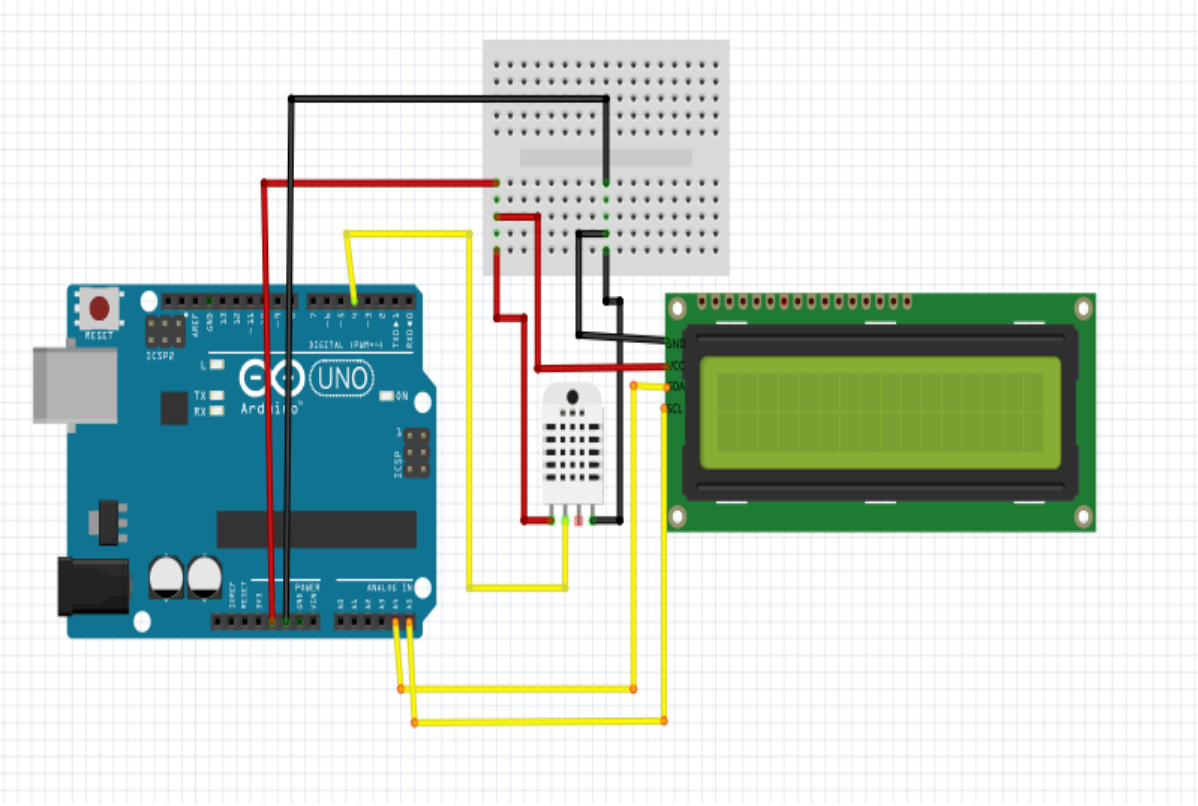
Aim

To interface a DHT11 temperature and humidity sensor with an Arduino UNO and display the measured values on a 16x2 LCD (without I^2C interface) using parallel data connection.

Components Required

- Arduino UNO board
- DHT11 temperature and humidity sensor
- 16x2 LCD display (parallel interface)
- 10k potentiometer (for LCD contrast control)
- Jumper wires & breadboard
- USB cable for Arduino programming

Circuit Connection:



LCD 16X2 Pin details:

LCD Pin	**Arduino Pin**
VSS	GND
VDD	5V
V0	Middle pin of potentiometer (contrast control)
RS	7
RW	GND
E	8
D4	9
D5	10
D6	11
D7	12
A (LED+)	5V
K (LED-)	GND

Library Tools:

- ✓ Open Arduino IDE.
- ✓ Go to **Sketch > Include Library > Manage Libraries**.
- ✓ Search and install:
  - **DHT sensor library** by Adafruit
  - **Adafruit Unified Sensor**
  - **LiquidCrystal** (built-in)

Program:

```
#include <LiquidCrystal.h>
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11 // or DHT22
// LCD pins: RS, E, D4, D5, D6, D7
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
DHT dht(DHTPIN, DHTTYPE);
void setup() {
 lcd.begin(16, 2); // 16 columns, 2 rows
 dht.begin();
}
void loop() {
 float humidity = dht.readHumidity();
 float temperature = dht.readTemperature();
 if (isnan(humidity) || isnan(temperature)) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Sensor Error");
  delay(2000);
  return;
 }
 // Display temperature
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Temp: ");
 lcd.print(temperature);
 lcd.print((char)223); // degree symbol
 lcd.print("C");
```

```
 // Display humidity
 lcd.setCursor(0, 1);
 lcd.print("Humidity: ");
 lcd.print(humidity);
 lcd.print("%");
 delay(2000); // update every 2s
}
```

Procedure

1. Connect the DHT11 sensor to the Arduino UNO as per the wiring diagram.
2. Connect the 16x2 LCD in parallel mode with RS, E, and data pins (D4–D7) to Arduino digital pins 7–12.
3. Connect the potentiometer to the LCD V0 pin to adjust display contrast.
4. Install the **LiquidCrystal** and **DHT sensor library** in Arduino IDE.
5. Write and upload the Arduino program to read temperature & humidity and display them on the LCD.
6. Power the Arduino board and observe the LCD display.

Result

The Arduino successfully reads temperature and humidity values from the DHT11 sensor and displays them in real time on the 16x2 LCD display. The first row shows **temperature (°C)** and the second row shows **humidity (%)**.

Ex:5 Fire Detection using Alarm System

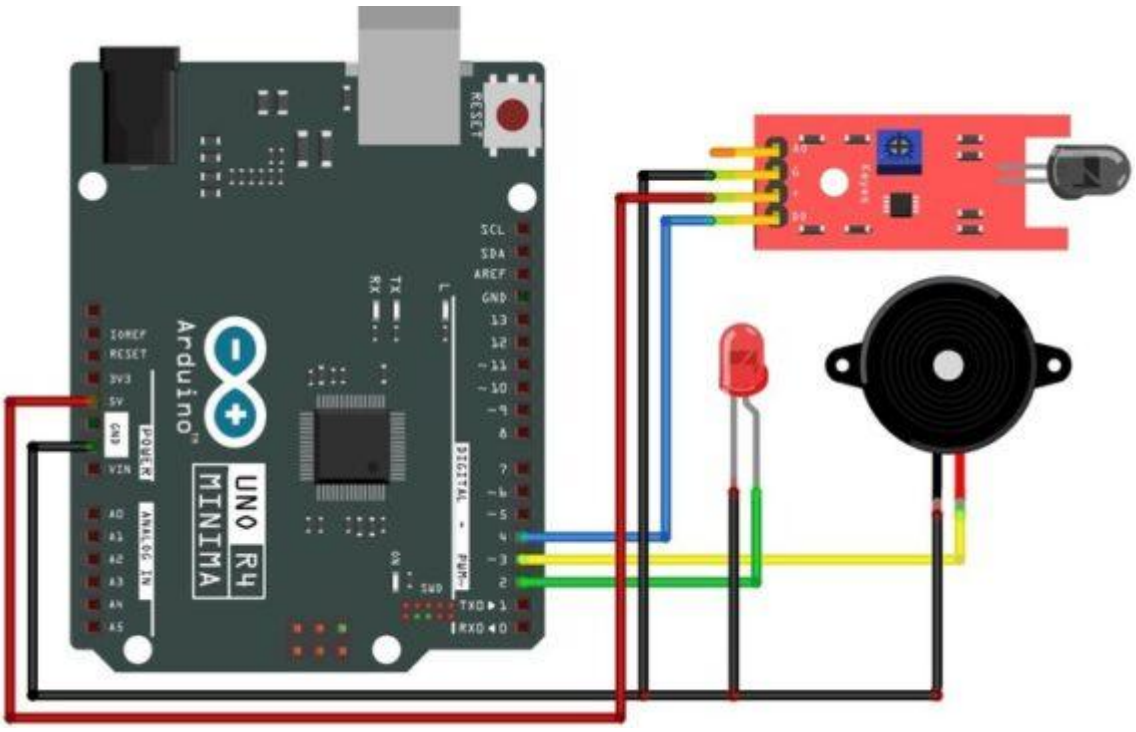
Aim

To design and implement a fire detection system using a flame sensor and Arduino that triggers a buzzer alarm when fire is detected.

Components Required

- Arduino UNO board
- Flame sensor module (digital output type)
- Buzzer module
- Jumper wires & breadboard
- USB cable for Arduino programming

Circuit Connection:



Program:

```
int flamePin = 2;   // Flame sensor digital output pin

int buzzerPin = 8;  // Buzzer pin

int flameState = HIGH; // Sensor state
```

```
void setup() {
  pinMode(flamePin, INPUT);
  pinMode(buzzerPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  flameState = digitalRead(flamePin);

  if (flameState == LOW) { // Flame detected (LOW means fire in most sensors)
    Serial.println("Fire Detected!");
    digitalWrite(buzzerPin, HIGH); // Turn buzzer ON
    delay(100);
    digitalWrite(buzzerPin, LOW);  // Buzzer beep
    delay(100);
  }
  else {
    Serial.println("No Fire");
    digitalWrite(buzzerPin, LOW); // Turn buzzer OFF
  }

  delay(200);
}
```

Procedure

1. **Hardware Setup**
    - Connect the **flame sensor**:
        - VCC → 5V on Arduino
        - GND → GND on Arduino

        - DO (Digital Output) → Digital Pin 2 on Arduino
    - Connect the **buzzer**:
        - Positive terminal (+) → Digital Pin 8 on Arduino
        - Negative terminal (−) → GND on Arduino
2. **Software Setup**
    - Open Arduino IDE and create a new sketch.
    - Write the program to:
        - Read the flame sensor's digital output.
        - Turn ON the buzzer if fire is detected (sensor output LOW).
        - Print fire status in the Serial Monitor.
    - Upload the code to the Arduino board.
3. **Testing**
    - Power the Arduino through USB or an external power supply.
    - Place a small flame (e.g., from a lighter) near the sensor to test detection.
    - Observe the buzzer sound and Serial Monitor output.

Result

The Arduino successfully detects fire using the flame sensor and activates the buzzer when the flame is within the sensor's detection range. The Serial Monitor displays **"Fire Detected!"** when fire is present and **"No Fire"** otherwise.

Ex.No:6 Water Level Detector using Arduino

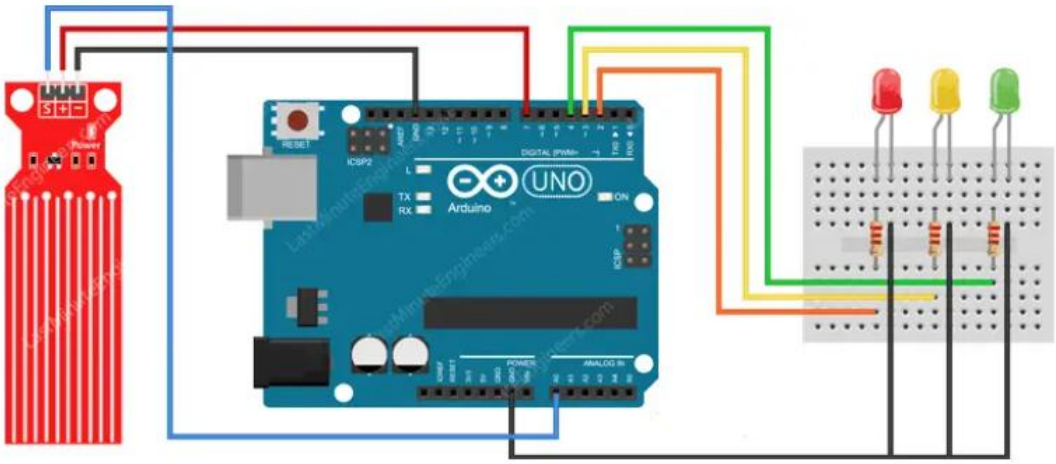
Aim

To design and implement a water level detector system using Arduino, a water level sensor (or probe), and LEDs to indicate different water levels (LOW, MEDIUM, HIGH).

Apparatus / Requirements

- Arduino board (Uno/Nano/compatible)
- Breadboard and jumper wires
- 3 × LEDs (Green = Low, Yellow = Medium, Red = High)
- 3 × current-limiting resistors (220 Ω – 1 kΩ)
- Water level sensor module (or self-made probes with conductive wires) connected to analog pin A0
- USB cable and Arduino IDE for programming
- Container with water (to test sensor readings)

Circuit Diagram:



Program:

```
#define lowOutput 13
#define mediumOutput 12
#define highOutput 11
#define sensorPin A0
void setup() {
 Serial.begin(9600);
 pinMode(lowOutput, OUTPUT);
 pinMode(mediumOutput, OUTPUT);
 pinMode(highOutput, OUTPUT);
 digitalWrite(lowOutput, LOW);
 digitalWrite(mediumOutput, LOW);
 digitalWrite(highOutput, LOW);
}
void loop() {
 int sensorValue=analogRead(sensorPin);
 if (150 > sensorValue) {
  digitalWrite(lowOutput, HIGH);
  digitalWrite(highOutput, LOW);
  digitalWrite(mediumOutput, LOW);
 }  else if (200 > sensorValue) {
  digitalWrite(mediumOutput, HIGH);
  digitalWrite(highOutput, LOW);
  digitalWrite(lowOutput, LOW);
 } else if (260 > sensorValue) {
  digitalWrite(highOutput, HIGH);
  digitalWrite(mediumOutput, LOW);
  digitalWrite(lowOutput, LOW);
 }
 else {
```

```
  digitalWrite(highOutput, LOW);
  digitalWrite(mediumOutput, LOW);
  digitalWrite(lowOutput, LOW);
 }
 delay(100);
}
```

Procedure

1. Connect the water level sensor to Arduino as per circuit.
2. Connect the LEDs to pins 13, 12, and 11 with resistors.
3. Upload the Arduino program:
    - Read analog value from the sensor (0–1023).
    - Compare the reading with threshold values.
    - Turn ON one LED according to detected level.
4. Place the sensor into the water container and vary the water level.
5. Observe sensor readings on the Serial Monitor and note which LED turns ON.
    - **Below threshold 1 (150)** → LOW LED ON (low water).
    - **Between 150–200** → MEDIUM LED ON.
    - **Between 200–260** → HIGH LED ON.
    - **Above 260** → All LEDs OFF (overflow condition or out of calibrated range).

Result

The water level detector system was successfully implemented.
The Arduino reads the analog signal from the water level sensor and indicates the level through LEDs: **Green for low, Yellow for medium, and Red for high water levels**.

Ex.No: 7 Soil Moisture Based Irrigation using Arduino Board

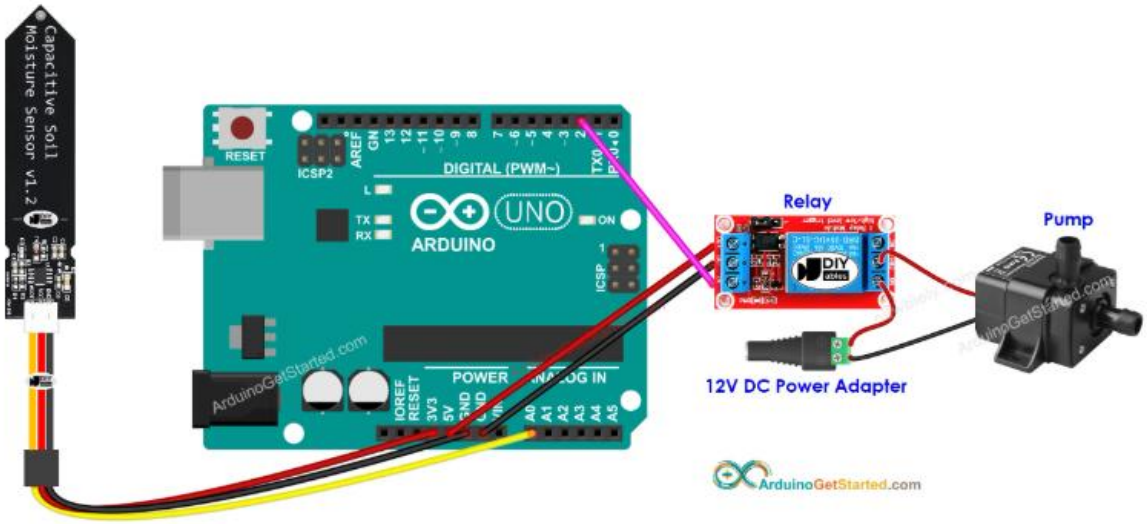
AIM

To design and implement an automatic irrigation system that uses a soil moisture sensor and Arduino UNO to control a water pump depending on soil moisture level.

Apparatus/Requirements

1. Arduino UNO board
2. Soil moisture sensor with probe
3. Relay module (5V, single channel)
4. Mini water pump (or motor)
5. External power supply for pump (e.g., 9V/12V battery or adapter)
6. Jumper wires and breadboard
7. Connecting tubes and soil pot

Circuit Connection:



Theory

- The system works on the principle of **soil moisture sensing**.
- The soil moisture sensor outputs an **analog signal** proportional to the water content in the soil:
    - Wet soil → **low resistance** → lower sensor reading.
    - Dry soil → **high resistance** → higher sensor reading.
- Arduino reads this analog signal through its **ADC (0–1023)**.

- A threshold is defined:
    - If soil is dry (value > threshold) → Arduino **activates relay**, turning ON the water pump.
    - If soil is wet (value < threshold) → Arduino **deactivates relay**, turning OFF the pump.
- This forms a closed-loop **automatic irrigation system** with minimal human intervention.

Program:

```
int sensorPin = A0;   // Soil moisture sensor analog pin
int sensorValue = 0;
int relayPin = 2;     // Relay control pin
int threshold = 600;  // Adjust based on calibration


void setup() {
 pinMode(relayPin, OUTPUT);
 Serial.begin(9600);
}
void loop() {
 sensorValue = analogRead(sensorPin);
 Serial.print("Soil Moisture: ");
 Serial.println(sensorValue);
 if(sensorValue > threshold) {
  digitalWrite(relayPin, LOW);  // Relay ON (pump starts)
  Serial.println("Soil is DRY - Pump ON");
 } else {
  digitalWrite(relayPin, HIGH); // Relay OFF (pump stops)
  Serial.println("Soil is WET - Pump OFF");
 }
 delay(1000);
}
```

Procedure

1. Connect all components as per the circuit diagram.
2. Upload the Arduino code for automatic irrigation system.
3. Place the soil moisture probe into the soil.
4. Initially keep the soil dry and observe: pump should turn ON automatically.
5. Water the soil and observe: when moisture increases, pump should turn OFF automatically.
6. Record the sensor values and corresponding pump states.

Result

- The irrigation system automatically **switches ON the water pump** when the soil is dry and **switches OFF** when the soil is sufficiently wet.
- This ensures **efficient water usage** and reduces the need for manual monitoring.

Exp. No:8 Detect theft using a PIR sensor and Arduino

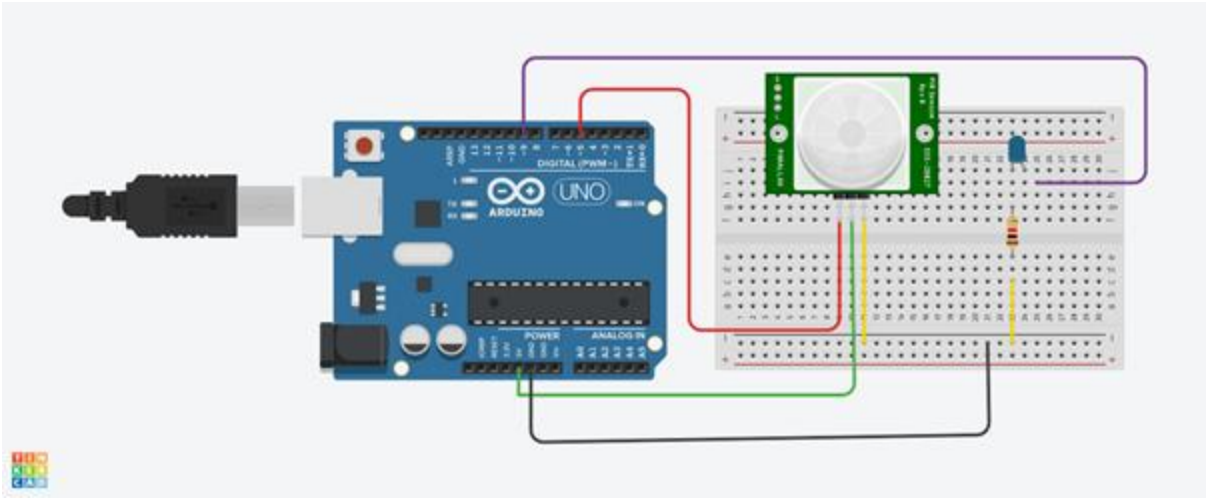
AIM:

To design a simple and cost-effective theft detection system using a **PIR (Passive Infrared) sensor** and **Arduino** that detects human motion and alerts using a buzzer or LED.

Required Components:

1. **Arduino Uno (or any compatible board)**
2. **PIR Motion Sensor**
3. **Buzzer or LED** (for alert)
4. **Breadboard and jumper wires**
5. (Optional) **Relay module** to control high-power devices like sirens or lights
6. (Optional) **GSM module** or **WiFi module** for remote alerts

Circuit Connection:



Program:

```
// PIR Sensor Theft Detection using Arduino
int pirPin = 2;      // PIR sensor output pin
int buzzerPin = 13;  // Buzzer or LED pin
int pirState = LOW;  // Default state
void setup() {
 pinMode(pirPin, INPUT);
 pinMode(buzzerPin, OUTPUT);
 Serial.begin(9600);
```

```
}
void loop() {
 int motionDetected = digitalRead(pirPin);
 if (motionDetected == HIGH) {
  digitalWrite(buzzerPin, HIGH); // Turn on buzzer/LED
  if (pirState == LOW) {
   Serial.println("Motion Detected! Possible Theft!");
   pirState = HIGH;
  }
 } else {
  digitalWrite(buzzerPin, LOW); // Turn off buzzer/LED
  if (pirState == HIGH) {
   Serial.println("Motion Stopped.");
   pirState = LOW;
  }
 }
 delay(100); // Short delay
}
```

Procedure:

1. **Connect the PIR sensor**:
   - VCC to 5V on Arduino
   - GND to GND on Arduino
   - OUT to digital pin 2
2. **Connect the buzzer or LED**:
   - Positive leg to pin 13
   - Negative leg to GND
3. **Write and upload code** to the Arduino using the Arduino IDE.
4. **Open the Serial Monitor** in the Arduino IDE to monitor the motion detection status.
5. **Test the setup** by moving in front of the PIR sensor.
6. When motion is detected, the buzzer/LED will turn on and a message will appear in the Serial Monitor.

Result:

The system successfully detects human motion using the PIR sensor. When motion is detected:

- The **buzzer or LED is activated**
- A **message is printed** to the Serial Monitor indicating that motion has been detected

Exp. No:9 - RFID-based Attendance System using Arduino

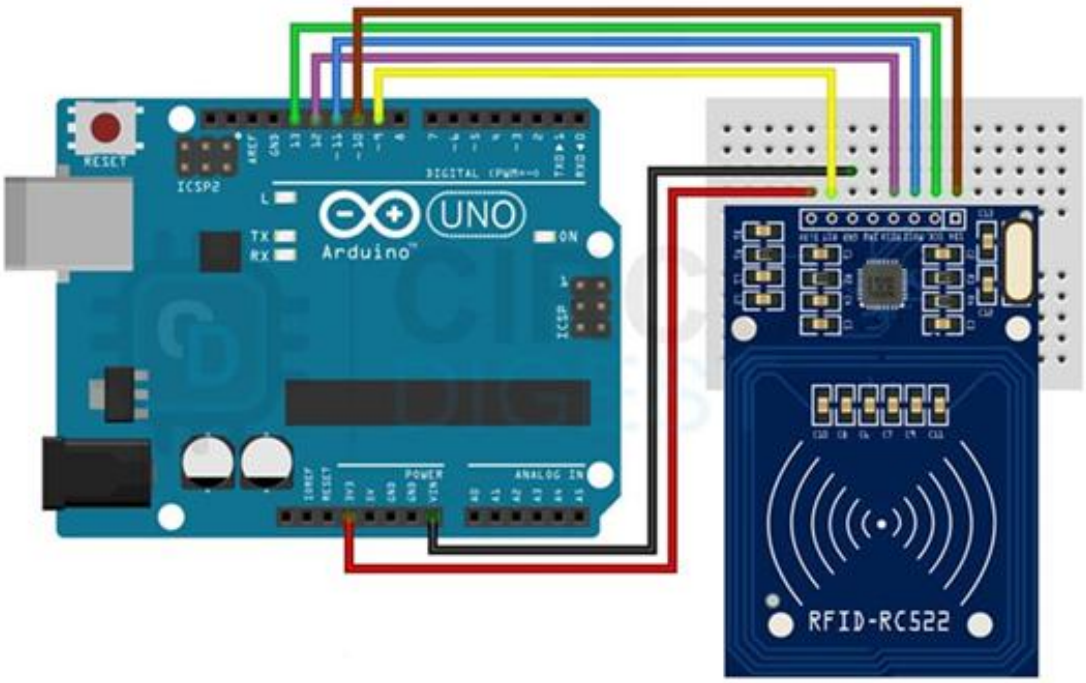
AIM:

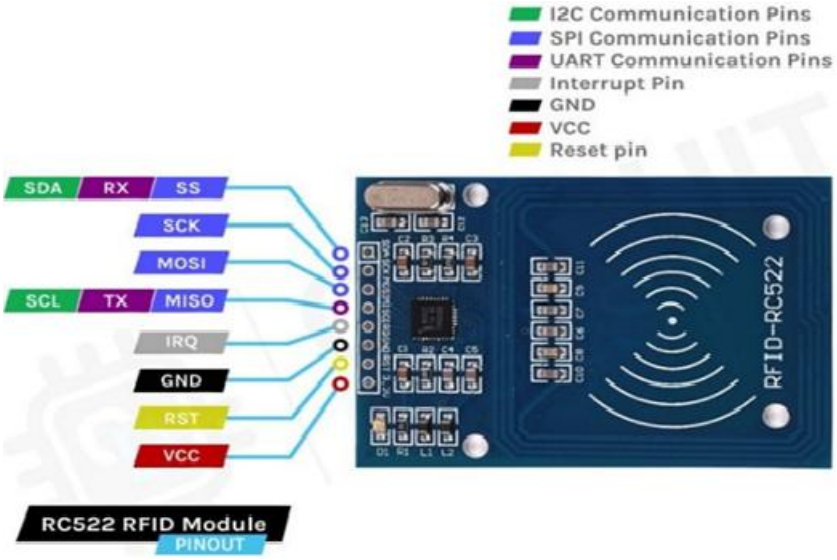
To design and implement an RFID-based attendance system using Arduino that identifies authorized cards and displays attendance status on the Serial Monitor.

Apparatus / Materials Required

Sl. No	Component	Specification / Quantity
1	Arduino UNO	1
2	RFID RC522 Module	1
3	RFID Tags / Cards	2–5
4	Jumper Wires	As required
5	Breadboard	1
6	Buzzer / LED	Optional for indication
7	USB Cable	To connect Arduino to PC
8	Arduino IDE	Software for coding

Circuit Connection:





Theory:

RFID (Radio-Frequency Identification) is a technology that uses electromagnetic fields to automatically identify and track tags attached to objects.
The **RC522 module** communicates with Arduino via **SPI protocol**, and each RFID card/tag has a unique UID (Unique ID). By comparing the scanned UID with a pre-stored list of authorized UIDs, the system marks attendance.

Code:

```
#include <SPI.h>

#include <MFRC522.h>


#define SS_PIN 10

#define RST_PIN 9

MFRC522 rfid(SS_PIN, RST_PIN);


#define LED_PIN 8      // Main indicator LED/Buzzer

#define BUZZER_PIN 7   // Optional buzzer or second LED


// Authorized card UIDs

String knownIDs[] = {"A3 A1 95 98", "B4 4D 51 21"};
```

```
void setup() {
  Serial.begin(9600);
  SPI.begin();
  rfid.PCD_Init();
  Serial.println("RFID Reader Ready. Scan your card...");

  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
}

void loop() {
  if (!rfid.PICC_IsNewCardPresent()) return;
  if (!rfid.PICC_ReadCardSerial()) return;

  // Convert UID to string
  String uidString = "";
  for (byte i = 0; i < rfid.uid.size; i++) {
    if (rfid.uid.uidByte[i] < 0x10) uidString += "0";
    uidString += String(rfid.uid.uidByte[i], HEX);
    if (i < rfid.uid.size - 1) uidString += " ";
  }
  uidString.toUpperCase();

  Serial.print("Card UID: ");
  Serial.println(uidString);

  // Check if UID is authorized
  bool authorized = false;
  for (int i = 0; i < sizeof(knownIDs) / sizeof(knownIDs[0]); i++) {
    if (uidString == knownIDs[i]) {
      authorized = true;
```

```
      break;
    }
  }

  if (authorized) {
    Serial.println("✅ Authorized - Attendance Marked!");
    // Blink LED/buzzer for authorized
    for (int i = 0; i < 2; i++) {
      digitalWrite(LED_PIN, HIGH);
      digitalWrite(BUZZER_PIN, HIGH);
      delay(200);
      digitalWrite(BUZZER_PIN, LOW);
      delay(200);
    }
    digitalWrite(LED_PIN, LOW);
  } else {
    Serial.println("❌ Unauthorized Card!");
    digitalWrite(LED_PIN, HIGH);
    digitalWrite(BUZZER_PIN, HIGH);
    delay(2000);  // long alert for unauthorized
    digitalWrite(LED_PIN, LOW);
    digitalWrite(BUZZER_PIN, LOW);
  }

  delay(2000); // Small delay before next read
}
```

Procedure:

1. **Hardware Setup**
    - Connect the **RFID RC522 module** to Arduino as follows:

RC522 Pin	Arduino Pin
VCC	3.3V
GND	GND
SDA (SS)	D10
SCK	D13
MOSI	D11
MISO	D12
RST	D9

- Connect **LED** to pin 8 (optional) and **Buzzer** to pin 7 (optional).

2. **Software Setup**
   - Install **Arduino IDE** and the required libraries:
     - MFRC522
     - SPI (built-in)
3. **Programming**
   - Open Arduino IDE and upload the **RFID attendance code** (provided earlier).
   - Store the **authorized card UIDs** in the code.
4. **Operation**
   - Open the **Serial Monitor** at 9600 baud rate.
   - Scan an RFID card/tag near the reader.
   - The Serial Monitor will display:
     - **UID of the card**
     - **Attendance status** (Authorized/Unauthorized)
   - Optional: LED/Buzzer will indicate authorization status.

5. **Recording Attendance**
   - Manually note the attendance from the Serial Monitor, or log it to a PC if needed.

Observations / Results

Card UID	Status
A3 A1 95 98	✅ Authorized

Card UID	Status
B4 4D 51 21	✅ Authorized
12 34 56 78	❌ Unauthorized

- When an **authorized card** is scanned:
    - Serial Monitor displays “Authorized – Attendance Marked”.
    - LED/buzzer blinks shortly.
- When an **unauthorized card** is scanned:
    - Serial Monitor displays “Unauthorized Card”.
    - LED/buzzer stays ON longer to indicate rejection.

Result:
The system successfully identifies authorized RFID cards, marks attendance, and provides visual/audio feedback using LED/Buzzer. Unauthorized cards are rejected.

Ex.No:10 - Measurement of Heart Rate using Pulse Sensor and Arduino

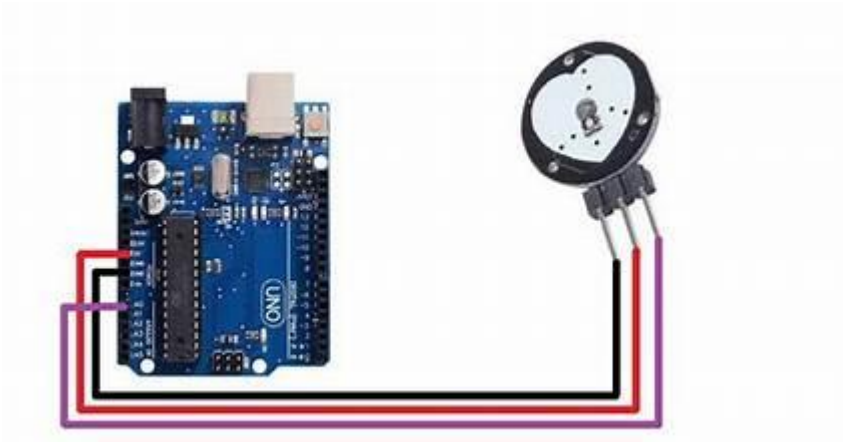
Aim

To measure and display the **heart rate (in Beats Per Minute)** using a **Pulse Sensor** interfaced with an **Arduino UNO**.

Apparatus Required

S. No.	Components / Equipment	Quantity
1	Arduino UNO board	1
2	Pulse Sensor module	1
3	Jumper wires	As required
4	Breadboard	1
5	USB cable (for Arduino)	1
6	Computer with Arduino IDE	1
7	(Optional) 16x2 LCD / OLED Display	1

Circuit Diagram



Procedure

1. **Connect the Circuit:**
    - Connect the pulse sensor to the Arduino as per the table above.

- Ensure correct orientation and firm connections.

2. **Install Library:**
   - Open **Arduino IDE**.
   - Go to **Sketch → Include Library → Manage Libraries**.
   - Search for **"PulseSensor Playground"** and install it.
3. **Write the Program**
4. **Upload the Code:**
   - Select the correct **board (Arduino UNO)** and **port**.
   - Click **Upload** to transfer the program.
5. **Observe the Output:**
   - Open the **Serial Monitor** at **9600 baud rate**.
   - Place your **index finger** gently on the pulse sensor.
   - Observe the **heart rate (BPM)** value displayed.

Program:

```
#include <PulseSensorPlayground.h>
const int PulseWire = A0;
int Threshold = 550;
PulseSensorPlayground pulseSensor;
void setup() {
 Serial.begin(9600);
 pulseSensor.analogInput(PulseWire);
 pulseSensor.setThreshold(Threshold);
 pulseSensor.begin();
}
void loop() {
 int myBPM = pulseSensor.getBeatsPerMinute();
 if (pulseSensor.sawStartOfBeat()) {
  Serial.print("Heart Beat Detected! BPM: ");
  Serial.println(myBPM);
```

```
  }
  delay(20);
}
```

Observation Table

Trial No.	Time Interval (s)	Number of Beats	Calculated BPM
1	60	72	72
2	60	75	75
3	60	74	74

Result

The **heart rate** of the subject was successfully measured using the **Pulse Sensor and Arduino UNO**.
The observed **average heart rate = 74 BPM**, which lies within the **normal range (60–100 BPM)** for a healthy adult.