Experiment No. 5

To Interface Motor Using a Relay with Arduino

Objective: To learn how to interface a motor with an Arduino using a relay module and control the motor operation programmatically.

Apparatus Required:

1. Arduino board (e.g., Arduino Uno)
2. Relay module (5V)
3. DC motor (6V or 12V)
4. Power supply for the motor (e.g., 6V or 12V battery)
5. Jumper wires
6. Breadboard (optional)
7. Diode (e.g., 1N4007)
8. Transistor (e.g., BC547 or 2N2222)
9. Resistor (1kΩ)

Theory: A relay is an electrically operated switch that can be used to control high-power devices like motors. The Arduino microcontroller cannot directly drive high-power motors, so a relay module acts as an interface to isolate the control circuit from the motor circuit.

Circuit Diagram: Include a clear circuit diagram showing the following connections:

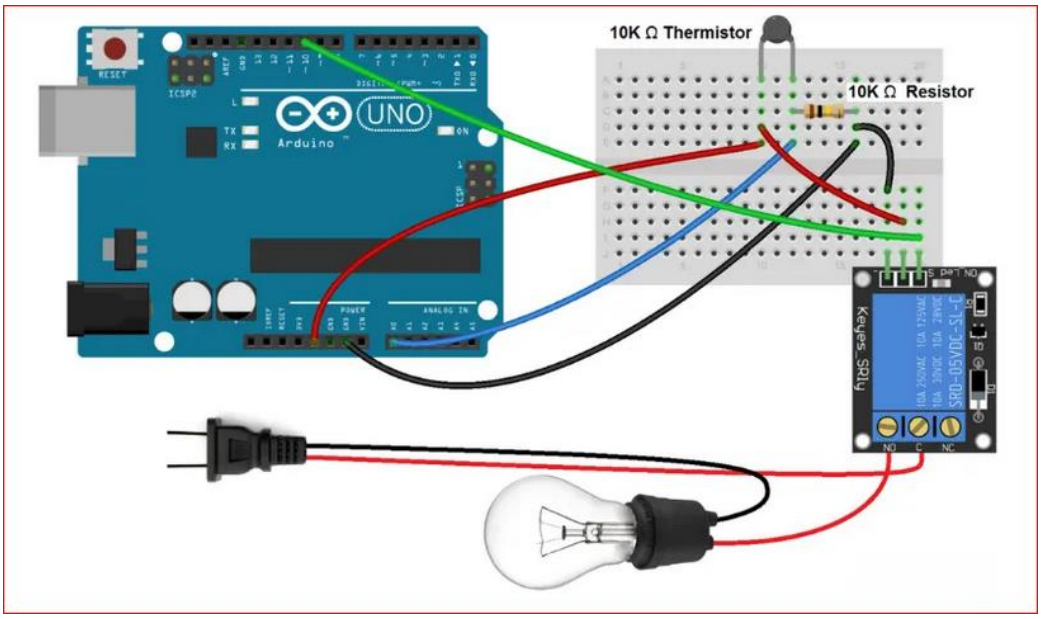
- Arduino digital pin connected to the relay module input.
- Relay module controlling the motor circuit.
- Power supply connected to the motor via the relay.
- Diode connected across the motor terminals to prevent back EMF damage.

Procedure:

1. **Hardware Setup:**

    1. Connect the VCC and GND pins of the relay module to the 5V and GND pins of the Arduino.
    2. Connect the IN pin of the relay module to one of the digital output pins on the Arduino (e.g., pin 7).
    3. Connect the common terminal (COM) of the relay to one terminal of the motor.

4. Connect the normally open (NO) terminal of the relay to the positive terminal of the motor's power supply.
5. Connect the negative terminal of the motor to the negative terminal of the power supply.
6. Place a diode across the motor terminals to protect against back EMF.



2. Arduino Programming:

1. Write a program to control the relay and motor using Arduino.
2. Load the program onto the Arduino board.

Arduino Code:

```
#define RELAY_PIN 7  // Define the pin connected to the relay

void setup() {
    pinMode(RELAY_PIN, OUTPUT);  // Set the relay pin as output
}

void loop() {
    digitalWrite(RELAY_PIN, HIGH);  // Turn on the relay (motor ON)
    delay(5000);                    // Wait for 5 seconds

    digitalWrite(RELAY_PIN, LOW);   // Turn off the relay (motor OFF)
    delay(5000);                    // Wait for 5 seconds
}
```

Precautions:

1. Ensure proper isolation between the high-power motor circuit and the low-power control circuit.
2. Double-check all connections before powering up the circuit.
3. Use a flyback diode across the motor terminals to protect the relay module from voltage spikes.
4. Avoid exceeding the current rating of the relay.

Result: The motor was successfully interfaced with the Arduino using a relay module and controlled programmatically.

EXPERIMENT NO. 6

Implementation of Mini Radar System.

Objective

To design and implement a mini radar system that detects objects within a specific range and displays the distance on an LCD or computer screen. The radar system uses an ultrasonic sensor and a servo motor controlled by an Arduino Uno.

Required Components

- **Arduino Uno**: Microcontroller board
- **HC-SR04 Ultrasonic Sensor**: To measure distance
- **Servo Motor (SG90)**: To rotate the sensor
- **LED Display** : Optional, for displaying distance data
- **Breadboard and Jumper Wires**
- **Arduino IDE**: Software for coding and uploading code to the Arduino
- **Computer**: To run the Arduino IDE and optional visualization

---

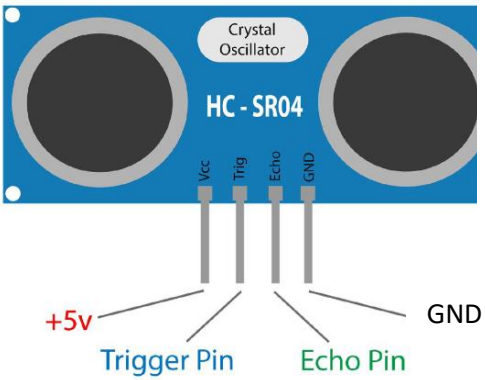
Theory

A radar system works by sending out ultrasonic pulses and measuring the time taken for these pulses to reflect back from an object. This information is used to determine the distance and, with a rotating sensor, the angular position of objects within range.

Circuit Diagram

The circuit involves connecting the ultrasonic sensor and servo motor to the Arduino Uno:

1. **Ultrasonic Sensor Connections**:
    - VCC to Arduino 5V
    - GND to Arduino GND
    - TRIG to Arduino Digital Pin 9
    - ECHO to Arduino Digital Pin 8



2. **Servo Motor Connections**:
    - VCC to Arduino 5V
    - GND to Arduino GND
    - Signal to Arduino Digital Pin 10

3. **LCD Display Connections** (Optional):
    - Connect the LCD to appropriate digital pins and set up according to the wiring for a 16x2 display.

Procedure

1. **Circuit Assembly**:
    - Connect the ultrasonic sensor and servo motor to the Arduino Uno as per the above circuit diagram.
    - If using an LCD, wire it to the Arduino pins for displaying the distance data.

2. **Code**:
    - Open the Arduino IDE and install any necessary libraries for the ultrasonic sensor, servo motor, and LCD.
    - Upload the following code to the Arduino. The code rotates the servo, triggers the ultrasonic sensor, measures the distance, and displays the information.

```
#include <Servo.h>

// Define pins

const int trigPin = 9;
```

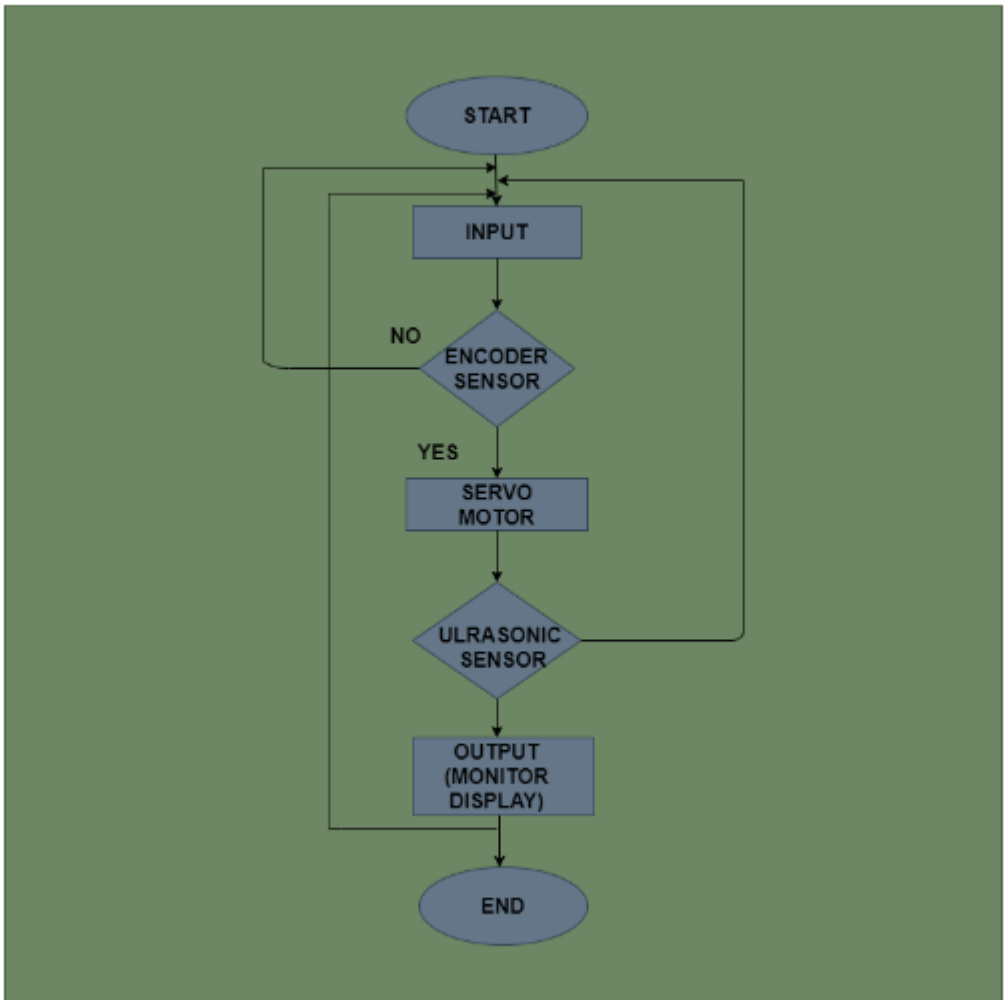
```
const int echoPin = 10;
const int servoPin = 6;
const int ledPin = 13;  // Optional LED for indicating radar activity

// Create Servo object
Servo radarServo;

// Variables for distance measurement
long duration;
int distance;
int angle = 0;  // Initial servo angle

void setup() {
  // Initialize serial communication for debugging
  Serial.begin(9600);

  // Initialize pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledPin, OUTPUT);

  // Attach servo to pin
  radarServo.attach(servoPin);
// Start the radar sweep
  radarServo.write(angle);  // Start at 0 degrees
```

```
}

void loop() {
  // Sweep radar across 180 degrees
  for (angle = 0; angle <= 180; angle += 5) {
    radarServo.write(angle);  // Move servo to the current angle
    delay(500);  // Wait for the servo to position

    // Measure distance
    distance = measureDistance();

    // Print angle and distance to Serial Monitor
    Serial.print("Angle: ");
    Serial.print(angle);
    Serial.print("° Distance: ");
    Serial.print(distance);
    Serial.println(" cm");

    // Optional: Light up the LED if something is detected
    if (distance < 25) {
      digitalWrite(ledPin, HIGH);  // Object detected within 50 cm
    } else {
      digitalWrite(ledPin, LOW);  // No object detected
    }
delay(500);  // Wait before taking another measurement
```

```
}

// Sweep back
for (angle = 180; angle >= 0; angle -= 5) {
  radarServo.write(angle);  // Move servo to the current angle
  delay(500);  // Wait for the servo to position

  // Measure distance
  distance = measureDistance();

  // Print angle and distance to Serial Monitor
  Serial.print("Angle: ");
  Serial.print(angle);
  Serial.print("° Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Optional: Light up the LED if something is detected
  if (distance < 25) {
    digitalWrite(ledPin, HIGH);  // Object detected within 50 cm
  } else {
    digitalWrite(ledPin, LOW);  // No object detected
  }

  delay(500);  // Wait before taking another measurement
```

```
  }
}

// Function to measure distance using the ultrasonic sensor
int measureDistance() {
  // Send a pulse to trigger the ultrasonic sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echo pin to determine the distance
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance (in centimeters)
  distance = duration * 0.0344 / 2;  // Speed of sound is 0.0344 cm/us

  return distance;
}
```

3. **Flow Chart**



4. Running the System:

- Connect the Arduino to your computer via USB.
- Open the Serial Monitor in the Arduino IDE to view angle and distance measurements.
- (Optional) For graphical visualization, you can use Processing or an LCD display.

5. **Testing**:
    - Place objects at different positions and observe the distance displayed.
    - Adjust code or wiring if necessary for accurate distance readings.

Observations and Calculations

- **Distance Calculation**: Distance=Duration×0.034/2\text{Distance} = \text{Duration} \times 0.034 / 2Distance=Duration×0.034/2
- **Angle Position**: The servo motor sweeps from 0° to 180° to scan the area.

Result

The mini radar system should display distances to nearby objects and, if applicable, the angle of detection.

Precautions

- Verify all connections before powering the Arduino to avoid short circuits.
- Calibrate the ultrasonic sensor if readings seem inaccurate.
- Avoid obstacles blocking the servo's range of motion.

Conclusion

The mini radar system successfully detects objects within a certain range and displays the distance. It serves as an introductory model for more advanced radar and obstacle detection systems.

Experiment No. 7

Implementation of Soil Moisture System for Agriculture

Objective: To design and implement a soil moisture monitoring and irrigation system using Arduino Uno, aimed at automating the watering process in agriculture.

Required Components

1. Arduino Uno
2. Soil Moisture Sensor
3. Relay Module
4. Water Pump
5. Jumper Wires
6. Breadboard
7. Power Supply (e.g., 12V adapter or battery)
8. Resistors (as required)
9. LED indicators (optional)
10. Buzzer (optional)

Theory

The soil moisture monitoring system uses a soil moisture sensor to measure the water content in the soil. Based on the moisture level, the system triggers a relay to control a water pump, ensuring efficient irrigation. This system helps conserve water and reduces manual effort in agriculture.

Circuit Diagram

Connections:

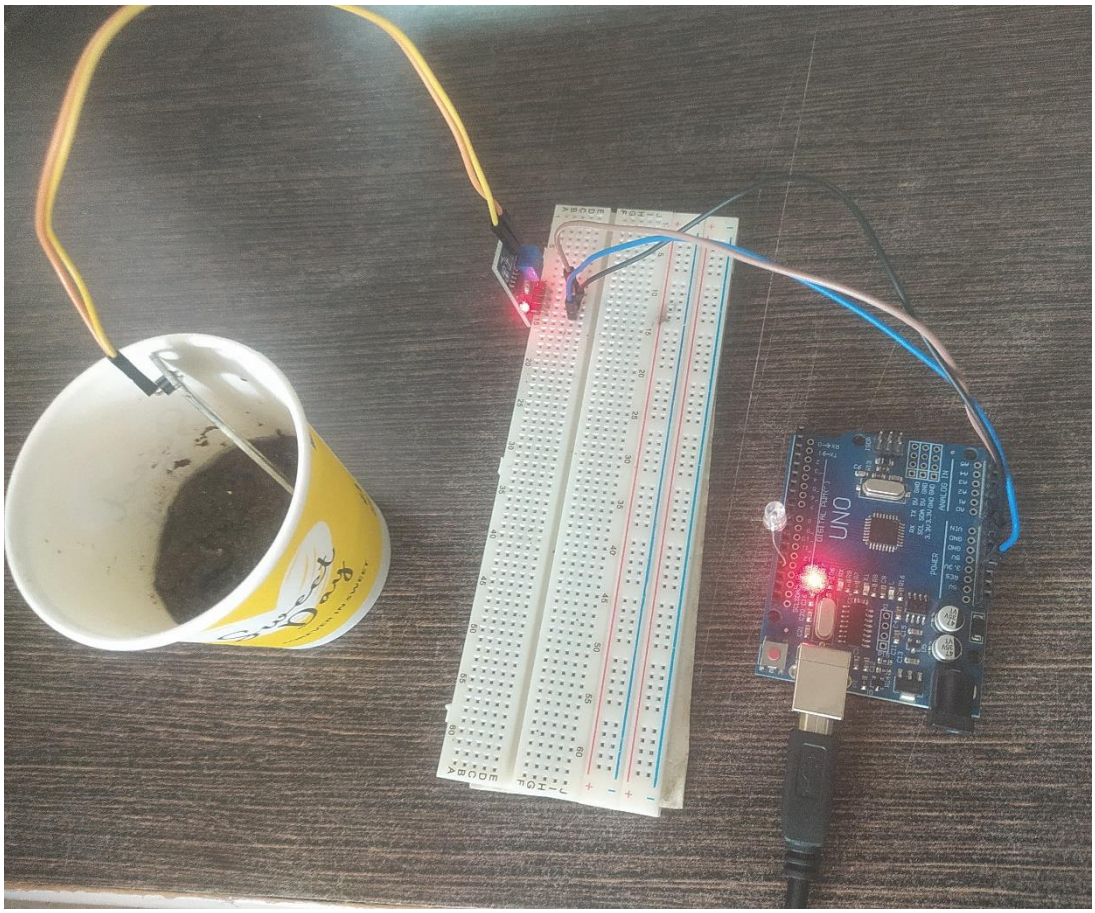
1. Connect the **Soil Moisture Sensor**:
    - VCC pin to 5V on Arduino
    - GND pin to GND on Arduino
    - Analog output pin to A0 on Arduino

2. Connect the **Relay Module**:
    - Signal pin to digital pin D7 on Arduino
    - VCC pin to 5V on Arduino

- GND pin to GND on Arduino

3. Connect the **Water Pump**:
   - Connect one terminal to the relay output.
   - Connect the other terminal to a 12V power supply.

4. Optional Components:
   - **LED**: Connect to a digital pin (e.g., D13) with a resistor in series.
   - **Buzzer**: Connect to a digital pin (e.g., D8) with proper polarity.



Procedure

Step 1: Assemble the Components

1. Connect the soil moisture sensor and relay module to the Arduino as per the circuit diagram.
2. Ensure the water pump is securely connected to the relay output.
3. Verify all connections before powering up the system.

Step 2: Program the Arduino Uno

1. Open the Arduino IDE.
2. Write the following code:

```
#define soilMoisturePin A0
#define relayPin 7
#define ledPin 13

void setup() {
    pinMode(relayPin, OUTPUT);
    pinMode(ledPin, OUTPUT);
    digitalWrite(relayPin, HIGH); // Pump off initially
    Serial.begin(9600);
}

void loop() {
    int sensorValue = analogRead(soilMoisturePin);
    Serial.print("Soil Moisture Value: ");
    Serial.println(sensorValue);

    // Adjust threshold based on soil and sensor
    if (sensorValue < 500) { // Dry soil
        digitalWrite(relayPin, LOW); // Turn on pump
        digitalWrite(ledPin, HIGH); // Turn on LED
    } else { // Wet soil
        digitalWrite(relayPin, HIGH); // Turn off pump
        digitalWrite(ledPin, LOW); // Turn off LED
    }

    delay(1000);
}
```

3. Upload the code to the Arduino Uno.

Step 3: Test the System

1. Insert the soil moisture sensor into dry and wet soil samples.
2. Observe the water pump's operation and LED indications.
3. Adjust the threshold value in the code if necessary.

Precautions

1. Ensure the soil moisture sensor is inserted properly into the soil.
2. Prevent water from coming into contact with electrical components.
3. Use components within their specified voltage and current ratings.

Observations

1. Record the soil moisture values for dry and wet soil.
2. Note the response time of the system in switching the pump on or off.
3. Check the system's reliability over multiple trials.

Result

The soil moisture monitoring and irrigation system was successfully implemented. The system effectively automates the irrigation process based on soil moisture levels, making it ideal for agricultural applications.

EXPERIMENT NO. 8

Development of IoT-based Smart Lock System using Arduino

Objective

To design and implement an IoT-based smart lock system using Arduino for secure and remote-controlled access.

Requirements

Hardware

1. Arduino Uno
2. Servo motor
3. Wi-Fi Module (e.g., ESP8266 or NodeMCU)
4. Push button or keypad module
5. Breadboard and connecting wires
6. Resistors (as needed)
7. Power supply (5V or USB cable)
8. LED (optional for status indication)
9. Smartphone or computer with internet access

Software

1. Arduino IDE
2. IoT platform (e.g., Blynk, ThingSpeak, or Firebase)
3. Wi-Fi network for connectivity

Theory

An IoT-based smart lock system leverages an Arduino microcontroller to control a servo motor acting as a locking mechanism. The Wi-Fi module enables remote monitoring and control through an IoT platform. A user can lock or unlock the system using a mobile application, ensuring security and convenience.

Block Diagram

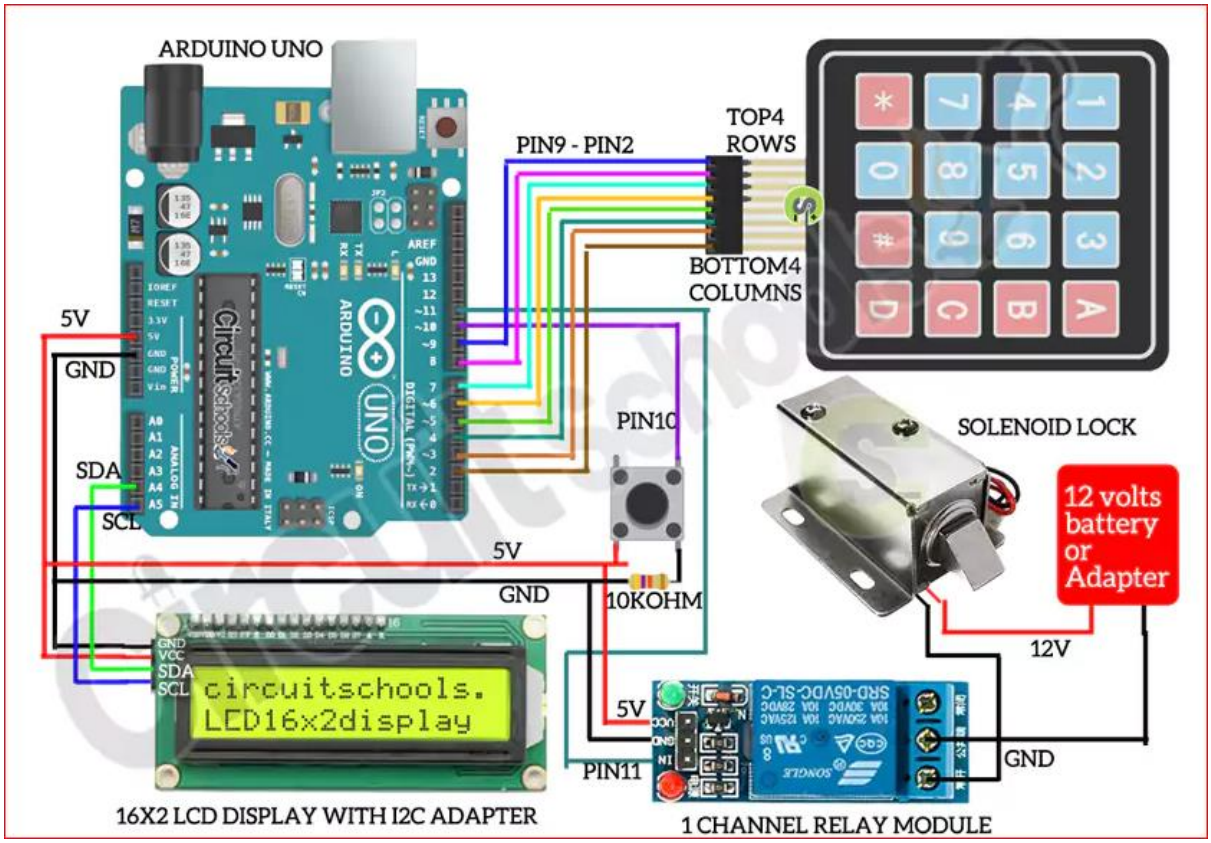
1. **Input Devices:** Push button or keypad module
2. **Processing Unit:** Arduino Uno
3. **Communication:** Wi-Fi Module

4. **Output Devices:** Servo motor and optional LED

Circuit Diagram

Create a schematic showing connections between components:

1. Connect the servo motor signal pin to a PWM pin on the Arduino.
2. Connect the Wi-Fi module to the Arduino for serial communication.
3. Connect the push button to a digital input pin of the Arduino.
4. Use resistors and LEDs for debugging or status indication.



Procedure

1. **Hardware Setup**
    1. Assemble the circuit as per the circuit diagram.
    2. Ensure proper power supply connections to avoid component damage.

2. **Programming**

    1. Write a program to control the servo motor and establish Wi-Fi connectivity.
    2. Integrate the IoT platform API to send/receive commands.
    3. Upload the program to the Arduino using the Arduino IDE.

3. **Testing and Verification**

    1. Use the IoT platform to send commands for locking/unlocking.
    2. Observe the servo motor and LED status changes as per commands.

Sample Code

```
#include <Servo.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// Define pins and objects
Servo myServo;
char auth[] = "YourBlynkAuthToken";
char ssid[] = "YourWiFiSSID";
char pass[] = "YourWiFiPassword";

void setup() {
  myServo.attach(9); // Connect to PWM pin
  myServo.write(0); // Initialize to locked position

  Blynk.begin(auth, ssid, pass);
}

BLYNK_WRITE(V1) {
  int lockState = param.asInt(); // Get button state
  if (lockState == 1) {
    myServo.write(90); // Unlock position
  } else {
    myServo.write(0); // Lock position
  }
}

void loop() {
  Blynk.run();
}
```

Precautions

1. Verify all connections before powering up.
2. Handle the Wi-Fi module and servo motor with care to avoid overheating.
3. Use proper grounding to avoid noise in signals.

Result

The IoT-based smart lock system is successfully implemented, allowing remote locking and unlocking through a mobile application.

Experiment No. 9

Implementation of pH Sensor for Smart Agriculture System Using Arduino

---

Objective: To design and implement a pH monitoring system using an Arduino microcontroller to assess soil health, crucial for smart agriculture systems.

Apparatus Required:

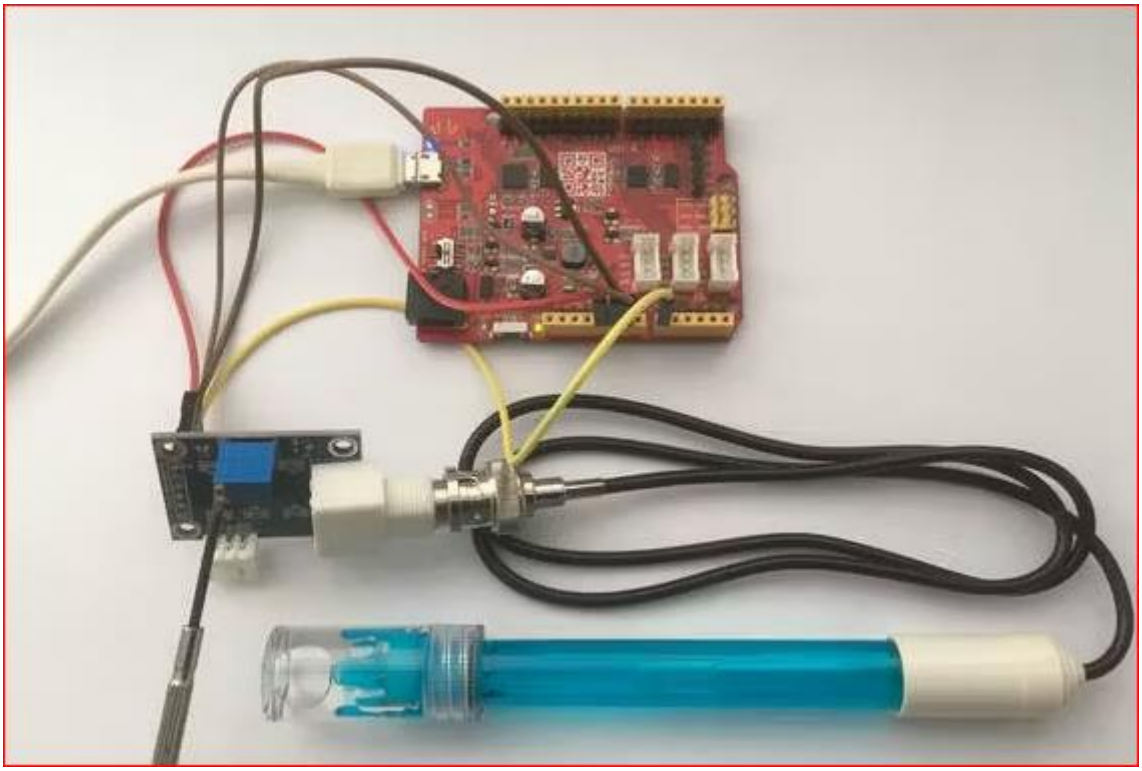
1. Arduino Uno microcontroller
2. pH sensor module (e.g., SEN0161 or equivalent)
3. Breadboard and jumper wires
4. LCD display (16x2) or OLED display
5. Potentiometer (if using an LCD display)
6. Power supply (USB cable or external battery)
7. Soil sample for testing
8. Laptop with Arduino IDE installed

Theory: Soil pH is a key factor that influences the availability of nutrients to plants. Smart agriculture systems use real-time pH monitoring to ensure optimal soil conditions for crop growth. A pH sensor measures the hydrogen ion concentration, providing a pH value that helps determine whether the soil is acidic, neutral, or alkaline.

The Arduino Uno acts as the processing unit, receiving data from the pH sensor, converting analog signals to digital readings, and displaying the pH value on an LCD or OLED display.

Circuit Diagram: (A schematic of the circuit can be included in the manual.)

1. Connect the VCC and GND pins of the pH sensor module to the Arduino's 5V and GND pins.
2. Connect the analog output pin of the pH sensor to the A0 pin of the Arduino.
3. For an LCD:
    - Connect RS, E, and D4-D7 pins to digital pins on Arduino.
    - Connect a potentiometer to adjust the contrast.
4. For an OLED display:
    - Connect SDA and SCL pins to the Arduino's A4 and A5 pins, respectively.
5. Power the Arduino through USB or an external power source.



Procedure:

1. **Hardware Setup:**
    - Assemble the circuit as per the diagram.
    - Ensure all connections are secure to avoid short circuits.
2. **Software Setup:**
    - Open the Arduino IDE.
    - Install the required libraries, such as LiquidCrystal for LCD or Adafruit SSD1306 for OLED.
    - Write the Arduino code to read the pH sensor output, process the data, and display the pH value. Use calibration formulas provided by the sensor datasheet.
3. **Calibration:**
    - Calibrate the pH sensor using buffer solutions (e.g., pH 4.0, 7.0, and 10.0).
    - Note the voltage readings for each buffer and adjust the code for accurate results.
4. **Testing:**
    - Place the pH sensor in the soil sample or buffer solution.
    - Observe the pH value displayed on the screen.
    - Record the readings and compare them with expected values.

Code:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
const int pH_pin = A0;
```

```
float voltage, pH_value;

void setup() {
  lcd.begin(16, 2);
  lcd.print("pH Sensor Test");
  delay(2000);
  lcd.clear();
}

void loop() {
  voltage = analogRead(pH_pin) * 5.0 / 1023.0;
  pH_value = 7 + ((2.5 - voltage) / 0.18); // Example calibration formula
  lcd.setCursor(0, 0);
  lcd.print("Voltage: ");
  lcd.print(voltage);
  lcd.setCursor(0, 1);
  lcd.print("pH: ");
  lcd.print(pH_value);
  delay(1000);
}
```

Precautions:

1. Handle the pH sensor delicately to avoid damaging the glass electrode.
2. Calibrate the sensor periodically for accurate readings.
3. Avoid exposing the sensor to extreme temperatures.

Result: The pH sensor system was successfully implemented using Arduino, and the pH value of the soil sample was accurately measured.

Experiment No. 10

Implementation of Smart Dustbin System for Smart City

Objective: To design and implement a smart dustbin system that utilizes Arduino Uno and sensors for automated waste disposal management in smart cities.

Required Components

1. Arduino Uno
2. Ultrasonic Sensor (HC-SR04)
3. Servo Motor
4. Jumper Wires
5. Breadboard
6. Resistors (as required)
7. Power Supply (e.g., 9V battery or USB)
8. LED indicators (optional)
9. Buzzer (optional)
10. Dustbin model

Theory

The smart dustbin system uses an ultrasonic sensor to detect the presence of an object near the dustbin. Once detected, the servo motor operates the lid of the dustbin, opening it to allow waste disposal. After a specified delay, the lid closes automatically. The system is designed to improve hygiene and convenience in waste disposal in smart cities.

Circuit Diagram

Connections:

1. Connect the **Ultrasonic Sensor**:
    - VCC pin to 5V on Arduino
    - GND pin to GND on Arduino
    - Trig pin to digital pin D9 on Arduino
    - Echo pin to digital pin D10 on Arduino

2. Connect the **Servo Motor**:
    - Signal wire to digital pin D3 on Arduino
    - VCC wire to 5V on Arduino
    - GND wire to GND on Arduino

3. Optional Components:
    - **LED**: Connect to a digital pin (e.g., D13) with a resistor in series.
    - **Buzzer**: Connect to a digital pin (e.g., D8) with proper polarity.

Procedure

Step 1: Assemble the Components

1. Mount the ultrasonic sensor and servo motor on the dustbin model.
2. Connect the components as per the circuit diagram.
3. Verify all connections before powering up the system.

Step 2: Program the Arduino Uno

1. Open the Arduino IDE.
2. Write the following code:

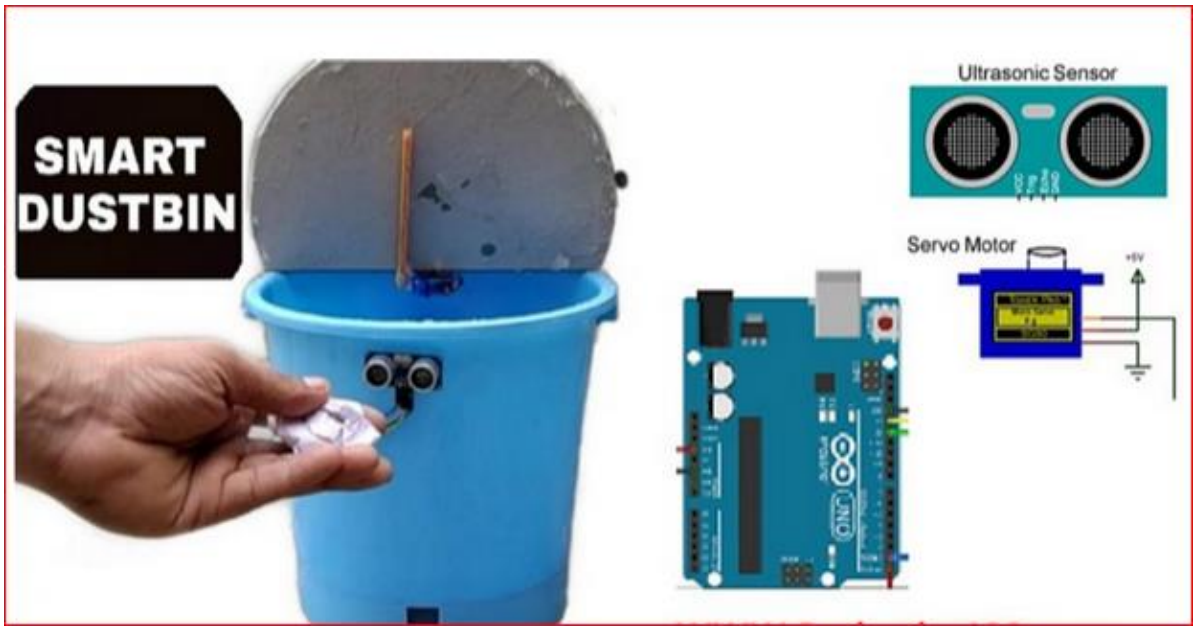
```
#include <Servo.h>

Servo servo;
const int trigPin = 9;
const int echoPin = 10;
const int servoPin = 3;
const int ledPin = 13;

void setup() {
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(ledPin, OUTPUT);
    servo.attach(servoPin);
    servo.write(0); // Lid closed initially
    Serial.begin(9600);
}

void loop() {
    long duration, distance;

    // Send ultrasonic pulse
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    // Measure pulse duration
```

```
    duration = pulseIn(echoPin, HIGH);

    // Calculate distance
    distance = (duration / 2) / 29.1; // cm

    if (distance <= 20 && distance > 0) { // If object is close
        digitalWrite(ledPin, HIGH); // Turn on LED
        servo.write(90);           // Open lid
        delay(5000);               // Wait 5 seconds
        servo.write(0);            // Close lid
        digitalWrite(ledPin, LOW); // Turn off LED
    }

    delay(100);
}
```

3. Upload the code to the Arduino Uno.

Step 3: Test the System

1. Place your hand near the ultrasonic sensor.
2. Observe the dustbin lid opening and closing automatically.
3. Optionally, check LED or buzzer indications.



Observations

1. Record the distance at which the dustbin opens.
2. Note the time taken for the lid to close automatically.
3. Check for any malfunctions or delays in response.

Precautions

1. Ensure correct connections to prevent short circuits.
2. Avoid placing the sensor in a dusty or moist environment.
3. Use components within their operating voltage range.

Result

The smart dustbin system was successfully implemented. The lid operates automatically based on object proximity, making it efficient for waste disposal management in smart cities.