

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**  
“JnanaSangama”, Belgaum -590014, Karnataka.



**LAB REPORT**  
on  
**INTERNET OF THINGS LAB**

*Submitted by*  
**KAMESH CHANDRA (1BM21CS271)**

*in partial fulfillment for the award of the degree of*  
**BACHELOR OF ENGINEERING**  
*in*  
**COMPUTER SCIENCE AND ENGINEERING**



**B.M.S. COLLEGE OF ENGINEERING**  
(Autonomous Institution under VTU)  
**BENGALURU-560019**  
**NOV-2023 to FEB-**  
**2024**

**B. M. S. College of Engineering,  
Bull Temple Road, Bangalore 560019**  
(Affiliated To Visvesvaraya Technological University, Belgaum)  
**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that the Lab work entitled “Internet Of Things Lab” carried out by **KAMESH CHANDRA (1BM21CS271)**, who is bonafide student of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2023. The Lab report has been approved as it satisfies the academic requirements in respect of a **Internet of things lab - (22CS5PCIOT)** work prescribed for the said degree.

**Karnam Sunil Kumar**

Assistant professor  
Department of CSE  
BMSCE, Bengaluru

**Dr. Jyothi S Nayak**

Professor and Head  
Department of CSE  
BMSCE, Bengaluru

,

## INDEX

<b>Sl. No.</b>	<b>Date</b>	<b>Program Title</b>	<b>Page No.</b>
1.	15-11-23	<b>LED Blinking</b>	2
2.	15-11-23	<b>LED ON/OFF Using Pushbutton</b>	4
3.	15-11-23	<b>LED Fading using Potentiometer</b>	6
4.	22-11-23	<b>Nightlight Simulation</b>	9
5.	22-11-23	<b>PIR with Arduino UNO</b>	11
6.	22-11-23	<b>Ultrasound with Arduino UNO</b>	14
7.	29-11-23	<b>Fire Alert System</b>	18
8.	29-11-23	<b>Automatic irrigation controller simulation</b>	21
9.	06-12-23	<b>Reading the code present on RFID tag</b>	23
10.	06-12-23	<b>Access control through RFID</b>	27
11.	27-12-23	<b>HC-05 Bluetooth at Command prompt</b>	28
12.	10-01-24	<b>HC-05 Bluetooth Controlled by mobile</b>	30
13.	10-01-24	<b>Bluetooth-Master Slave</b>	35
14.	17-01-24	<b>GSM Module</b>	45

1.

## LED Blinking

### Aim:

Turns on an LED on for one second, then off for one second, repeatedly.

### Hardware Required:

- Arduino Board
- LED

### Circuit diagram:

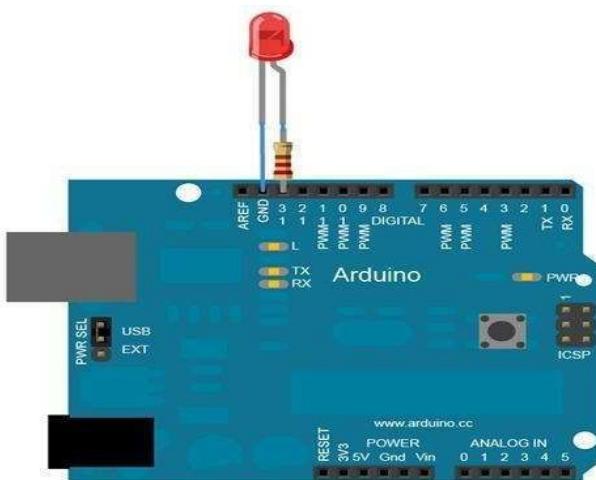


Fig.1.LED blinking

### Handwritten code pic:

Code

```
// Pin 13 has an LED connected on most Arduino boards
int led=13;
void setup() // the setup routine runs once when you press reset
{
  // initialize the digital pin as an output
  pinMode(led,OUTPUT);
}
```

```

void loop() {
    // The loop runs over and over again forever
    digitalWrite(led, HIGH); // Turn the LED on (HIGH is the
    // voltage level)
    delay(1000); // wait for a second
    digitalWrite(led, LOW); // Turn the LED off by making the voltage low
    delay(1000); // wait for a second
}

```

**Observation:**

LED switches on/off periodically. Digital output visualization using Arduino UNO.

### Code:

```

int led = 13;

void setup()
{
    pinMode(led, OUTPUT);
}

void loop() {
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
    delay(1000);
}

```

### Observation:

The code establishes a basic program to toggle an LED on and off in one-second intervals. Pin 13 is configured as the output for the LED, and the main loop continuously switches the LED on for one second, then off for another second.

2.

## LED ON/OFF Using Pushbutton

### Aim:

Turn an LED ON /OFF using a Pushbutton.

### Hardware Required:

- Arduino Board
- LED
- Push button

### Circuit diagram:

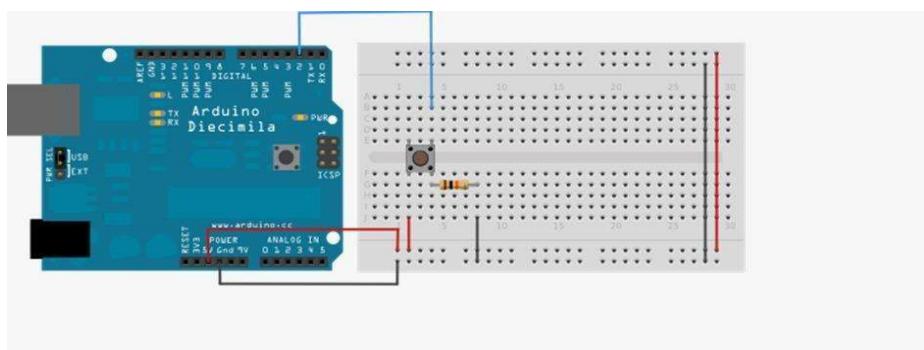


Fig.2.LED on/off using pushbutton

### Handwritten code pic:

Code :

```
const int buttonPin=2;
const int ledPin=13;
int buttonState=0;
void setup()
{
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}
```

```

void loop() {
    buttonState = digitalRead(buttonPin);
    if (buttonState == HIGH) {
        digitalWrite(ledPin, HIGH);
    } else {
        digitalWrite(ledPin, LOW);
    }
}

Observation:
LED switches on when push button is
pressed and it turns off when the
push button is released.

```

### Code:

```

const int buttonPin = 2;
const int ledPin = 13;
int buttonState = 0;
void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}
void loop() {
    buttonState = digitalRead(buttonPin);
    if (buttonState == HIGH) {
        digitalWrite(ledPin, HIGH);
    } else {
        digitalWrite(ledPin, LOW);
    }
}

```

### Observation:

The code achieves desired functionality of turning the LED on and off based on the state of the push button. When the button is pressed, the LED lights up. This interactive behavior enhances the user experience, where the LED state is directly controlled by the push button's input.

### 3.

## LED Fading using Potentiometer

### Aim:

To control the brightness of an LED using a Potentiometer.

### Hardware Required:

- Arduino Board
- LED
- Potentiometer

### Circuit diagram:

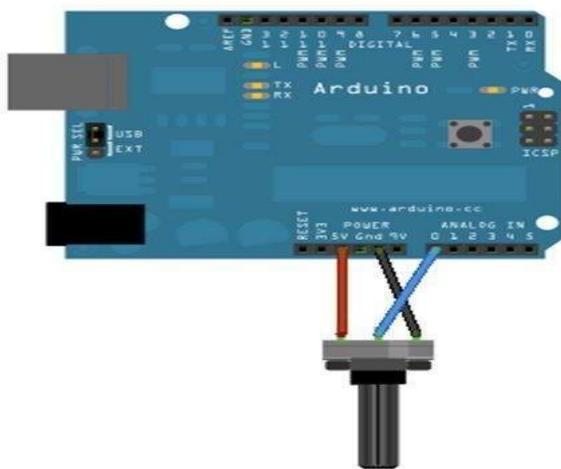


Fig.3-LED fading using potentiometer

### Handwritten code pic:

code: → POT E.

```
const int analogPin = A0;
const int analogOutPin = 9;
int sensorValue = 0;
int OutputValue = 0;
void setup() {
}
```

```

Serial.begin(9600);
}
void loop() {
    sensorValue = analogRead(analogInPin);
    outputValue = map(sensorValue, 0, 1023, 0, 255);
    analogWrite(analogOutPin, outputValue);
    Serial.print("sensorValue");
    Serial.println(outputValue);
    delay(2);
}

```

### Observation:

By rotating the potentiometer clockwise, the brightness of LED can be increased and if rotated in anti-clockwise direction its brightness can be reduced.

### Code:

```

const int potPin = A0;
const int ledPin = 9;
void setup() {
    pinMode(ledPin, OUTPUT);
}
void loop() {
    int potValue = analogRead(potPin);
    int brightness = map(potValue, 0, 1023, 0, 255);
    analogWrite(ledPin, brightness);
}

```

### Observation:

The code effectively achieves the desired outcome, enabling the dynamic control of the LED's brightness through the potentiometer. As the potentiometer is adjusted, the analogRead function captures its varying values (ranging from 0 to 1023). The mapping of these values to a brightness scale (0 to 255) results in adjustment of the LED's intensity.

#### 4.

## Nightlight Simulation

### Aim:

Simulating a night light using LDR and PIR

### Hardware Required:

- 1 LED
- 1 LDR
- 110K resistor

### Connection:

1. Attach one leg of LDR to 5V and another leg to Arduino Analog pin A0
2. Attach one leg of 110K resistor with that leg of LDR connected to A0
3. Attach another leg of resistor to the ground
4. Connect the positive leg of LED to pin 11 and negative to GND

### Circuit diagram:

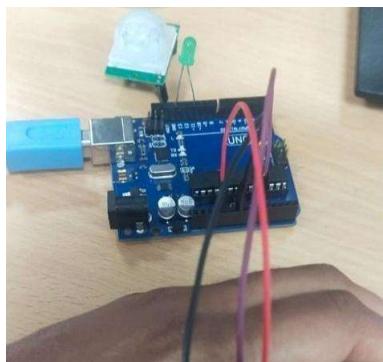


Fig 4.1- when it is bright, the LED is off.

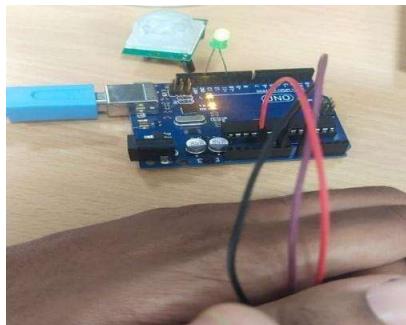


Fig 4.2- when dark, the LED turns on.

**Handwritten code pic:**

Code :

```
int LDR=0;
int LDRvalue=0;
int light_sensitivity=500;
void setup()
{
    Serial.begin(9600);
    pinMode(11, OUTPUT);
}
void loop()
{
    LDR value = analogRead(LDR);
    Serial.println(LDR value);
    delay(50);
    if (LDR value < light_sensitivity)
    {
        digitalWrite(11, HIGH);
    }
    else
    {
        digitalWrite(11, LOW);
    }
    delay(1000);
}

Observation :
write lights are switched off in the
room, LED should switch on, when
lights are switched on in the room, LED
should be switched immediately.
```

**Code:**

```
int LDR = 0;
int LDRValue = 0;
int light_sensitivity = 500;
void setup()
{
```

```
Serial.begin(9600);

pinMode(11, OUTPUT);
}

void loop()
{
LDRValue = analogRead(LDR);
Serial.println(LDRValue);
delay(50);
if (LDRValue < light_sensitivity)
{
digitalWrite(11, HIGH);
}
else
{
digitalWrite(11, LOW);
}
delay(1000);
}
```

### **Observation:**

The code successfully achieves the goal of simulating a night light based on the ambient light levels detected by the LDR. The analogRead function captures the LDR values, which are printed to the serial monitor for monitoring. The conditional statement compares these values to a light sensitivity threshold, and if the ambient light falls below this threshold, the LED is turned on, simulating a night light.

## 5. PIR with Arduino UNO

**Aim:** To detect the presence of human.

**Hardware Required:**

- 1 LED
- 1 PIR
- Arduino UNO

**Circuit diagram:**

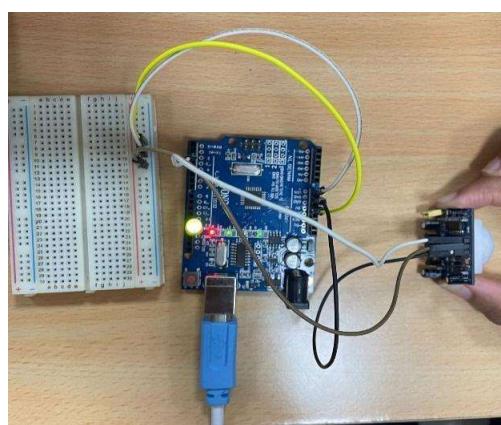


Fig 5- When motion is detected LED is high

**Handwritten code pic:**

code :

```
int SensorState = 0;
void setup()
{
    pinMode(2, INPUT);
    pinMode(13, OUTPUT);
    Serial.begin(9600);
}
```

void loop()

```
{ 
    int sensorState = digitalRead(2);
    if (sensorState == HIGH)
        digitalWrite(13, HIGH);
    Serial.println("sensor activated!");
}
else
{
    digitalWrite(13, LOW);
}
delay(10);
}
```

Observation:

In covering the PIR sensor, the bulb goes off and when the sensor is exposed to light, the LED goes off

### Code:

```
int sensorState = 0;
void setup()
```

```
{  
pinMode(2, INPUT);  
pinMode(13, OUTPUT);  
Serial.begin(9600); }  
void loop()  
{  
sensorState = digitalRead(2);  
if (sensorState == HIGH) {  
digitalWrite(13, HIGH);  
Serial.println("Sensor activated!");  
} else {  
digitalWrite(13, LOW);  
}  
delay(10); }
```

**Observation:**

The code effectively utilizes the PIR sensor to detect motion and responds by controlling the state of the LED. When motion is detected, the LED is illuminated, and a message is printed to the serial monitor.

## 6. Ultrasound with Arduino UNO

**Aim:** To detect the distance of an object.

**Hardware Required:**

- Ultrasonic sensor
- jumper wires(female to male)
- Arduino UNO

**Circuit diagram:**

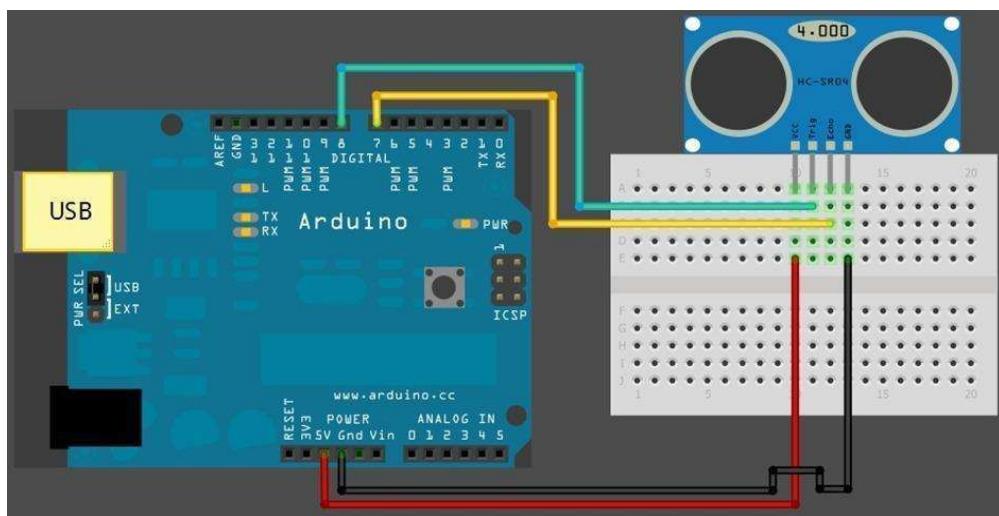


Fig 6-measures the distance of nearest object.

**Handwritten code pic:**

code :

```
int SensorState = 0;  
void setup()  
{  
    pinMode(2, INPUT);  
    pinMode(13, OUTPUT);  
    Serial.begin(9600);  
}
```

void loop()

```
{  
    int sensorState = digitalRead(2);  
    if (sensorState == HIGH)  
    {  
        digitalWrite(13, HIGH);  
        Serial.println("sensor activated!");  
    }  
    else  
    {  
        digitalWrite(13, LOW);  
    }  
    delay(10);  
}
```

Observation:

In covering the PIN sensor, the bulb  
goes off and when the sensor is exposed  
to light, the LED goes off

**Code:**

```
const int pingPin = 7;  
const int echoPin=6;  
void setup()  
{  
    Serial.begin(9600);  
    pinMode(pingPin, OUTPUT);  
    pinMode(echoPin, INPUT);  
}  
void loop()  
{  
    long duration, inches, cm;  
    digitalWrite(pingPin, LOW);  
    delayMicroseconds(2);  
    digitalWrite(pingPin, HIGH);  
    delayMicroseconds(10);  
    digitalWrite(pingPin, LOW);  
    duration = pulseIn(echoPin, HIGH);  
    inches = microsecondsToInches(duration);  
    Serial.print(inches);  
    Serial.print("inches");  
    cm = microsecondsToCentimeters(duration);  
    Serial.print(cm);  
    Serial.println("cm");  
}  
long microsecondsToInches(long microseconds)  
{  
    return microseconds / 74 / 2; }
```

```
long microsecondsToCentimeters(long microseconds)
{
    return microseconds / 29 / 2;
}
```

**Observation:**

The code effectively utilizes the ultrasonic sensor to measure distance and provides readings in both inches and centimeters. In the loop, a pulse is generated by triggering the ultrasonic sensor, and the duration of the pulse is measured using the pulseIn() function.

7.

## Fire Alert

### Aim:

## Fire alarm simulation.

## **Hardware Required:**

- Flame sensor (Analogue Output)
  - Arduino
  - Bread board
  - LED
  - Buzzer
  - Connecting wires

## **Connections:**

## Flame sensor interfacing to Arduino

## Flame sensor to Arduino

vcc to vcc

gnd to gnd

A0 to A0

## Led interfacing to Arduino

LED +ve is connected to 9th pin of Arduino

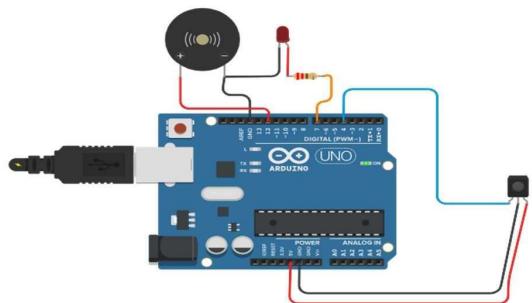
LED -ve is connected to gnd pin of arduino

## Buzzer interfacing to Arduino

Buzzer +ve is connected to 12th pin of Arduino

Buzzer -ve is connected to GND pin of Arduino

### **Circuit diagram:**



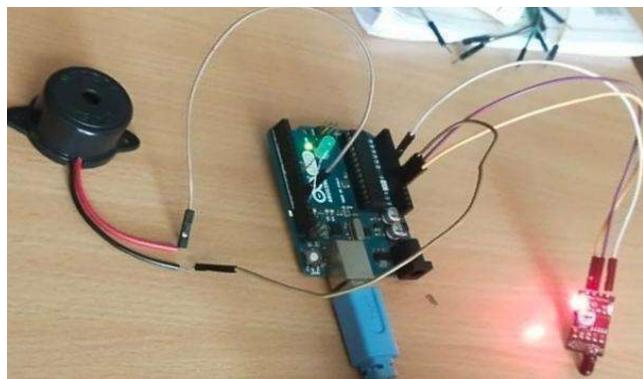


Fig 7- When the fire is detected LED turns on.

### Handwritten code pic:

```

code :
int sensorPin = A0;
int sensorValue = 0;
int led = 9;
int buzzee = 12;
void setup()
{
    pinMode(led, OUTPUT);
    pinMode(buzzee, OUTPUT);
    Serial.begin(9600);
}

void loop()
{
    sensorValue = analogRead(sensorPin);
    Serial.println(sensorValue);
    if (sensorValue < 100)
    {
        Serial.println("Fire detected");
        Serial.println("LED ON");
        digitalWrite(led, HIGH);
        digitalWrite(buzzee, HIGH);
        delay(1000);
    }
    digitalWrite(led, LOW);
    digitalWrite(buzzee, LOW);
    delay(sensorValue);
}

Observation:
Elarore sensor detecte eladition (heat) and
detecte the distance from the flame.

```

### Code:

```
int sensorPin = A0; // select the input pin for the LDR
```

```

int sensorValue = 0; // variable to store the value coming from the sensor
int led = 9; // Output pin for LED
int buzzer = 12; // Output pin for Buzzer
void setup() {
pinMode(led, OUTPUT);
pinMode(buzzer,OUTPUT);
Serial.begin(9600);
}
void loop()
{
sensorValue = analogRead(sensorPin);
Serial.println(sensorValue);
if (sensorValue < 100)
{
Serial.println("Fire Detected");
Serial.println("LED on");
digitalWrite(led,HIGH);
digitalWrite(buzzer,HIGH);
delay(1000);
}
digitalWrite(led,LOW);
digitalWrite(buzzer,LOW);
delay(sensorValue);
}

```

### **Observation:**

The code effectively simulates a fire alarm by monitoring the analog output of the flame sensor. When the sensor value falls below a predefined threshold (100 in this case), indicating the detection of a flame, the LED and buzzer are activated, and the corresponding messages are printed to the serial monitor.

## **8. Automatic irrigation controller simulation**

### **Aim:**

Sensing the soil moisture and sprinkling the Water simulation.

### **Hardware Required:**

- Arduino
- Moisture Sensor
- Breadboard
- Min servo motor

### **Connections:**

Moisture sensor VCC to Arduino 5V

Moisture sensor GND to Arduino GND

Moisture sensor A0 to Arduino A0

Servo motor VCC to Arduino 5V

Servo motor GND to Arduino GND

Servo Motor Signal to Arduino digital pin 9

### **Circuit diagram:**

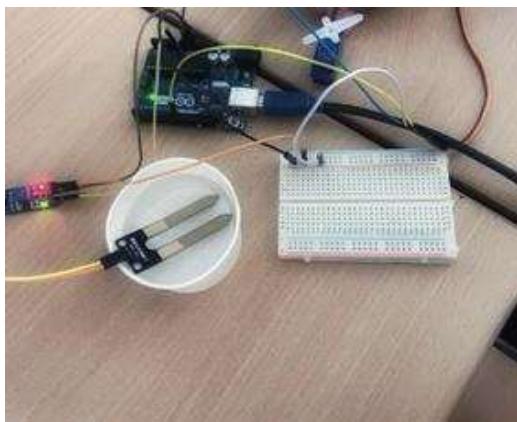


Fig 8- When moisture detected LED High, else Servo motor is on.

### **Handwritten code pic:**

Code:

```
#include <Servo.h>  
servo myservo;  
int pos=0;  
int sensorPin=A0;  
int sensorValue=0;  
void setup ()  
{
```

```
myservo.attach(9);  
Serial.begin(9600);  
}  
void loop()  
{  
    sensorValue = analogRead(sensorPin);  
    Serial.println(sensorValue);  
    if (sensorValue > 500)  
    {  
        pos (pos=0; pos <= 180; pos+=1)  
        myservo.write(pos);  
        delay(15);  
    }  
    else (pos=180; pos>0; pos-=1)  
        myservo.write(pos);  
        delay(15);  
    }  
    delay(1000);  
}
```

Observation:

When the metal foil sensor is placed in contact with another metal object and when it is placed in dry environment no movement is seen.

**Code:**

```
#include <Servo.h>;  
Servo myservo;  
int pos = 0;
```

```

int sensorPin = A0;
int sensorValue = 0;
void setup() {
  myservo.attach(9);
  Serial.begin(9600);
}
void loop()
{
  sensorValue = analogRead(sensorPin);
  Serial.println (sensorValue);
  if(sensorValue<500)
  {
    for (pos = 0; pos < 180; pos += 1)
    { // goes from 0 degrees to 180 degrees
      myservo.write(pos);
      delay(15); // waits 15ms for the servo to reach the position
    }
    for (pos = 180; pos < 0; pos -= 1)
    { // goes from 180 degrees to 0 degrees
      myservo.write(pos);
      delay(15); // waits 15ms for the servo to reach the position
    }
  }
  delay (1000);
}

```

### **Observation:**

The code simulates an automatic irrigation controller by utilizing a moisture sensor to monitor soil moisture levels. When the moisture level drops below the defined threshold, the servo motor moves to simulate the activation of a sprinkler system.

## 9. Reading the code present on RFID tag

### Aim:

The following code will read the code present on RFID tag and print it in serial monitor.

### Connection:

5V-Arduino 5V

GND-Arduino GND

Tx-pin 9

### Circuit diagram:



### Handwritten code pic:

Code :

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9, 10);
int count=0;
char input[23];
boolean flag=0;
void setup()
{
    Serial.begin(9600);
    mySerial.begin(9600);
```

```

void loop()
{
    if (mySerial.available())
    {
        count = 0;
        while (mySerial.available() && count < 12)
        {
            int input[12];
            mySerial.read();
            delay(5);
            send input (input);
        }
    }
}

Observation
The output on serial monitor will be
12 character ASCII data. Last 6 bits
first 10 bits will be tag number
and last 2 bits will be the XOR
result of the tag number outside
can be used for collision detection

```

### Code:

```

#include<SoftwareSerial.h>

SoftwareSerial mySerial(9, 10);

int count = 0; // count = 0
char input[12]; // character array of size 12
boolean flag = 0; // flag =0

void setup()
{
    Serial.begin(9600); // begin serial port with baud rate 9600bps
    mySerial.begin(9600);
}

void loop()
{
    if(mySerial.available())
    {
        count = 0;
        while(mySerial.available() && count < 12)
        {
            input[count] =mySerial.read();
            count++;
            delay(5);
        }
        Serial.print(input);
    }
}

```

### Observation:

The output in the serial monitor is the RFID tag number, and it allows for real-time monitoring and verification of the data read from the RFID tag.

## 10.

## Access control through RFID

### Aim:

The following code will read the code present on RFID tag tapped. If the code matches with the previously known tag (configured in the code), it will grant access (here LED will glow), otherwise access will be denied.

### Connection:

5V-Arduino 5V

GND-Arduino GND

Tx-pin 9

Led-pin 12

### Circuit diagram:

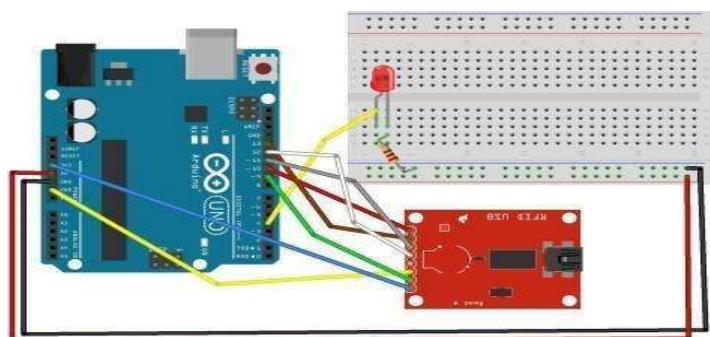


Fig.10.Access control through RFID

### Handwritten code pic:

code :

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9,10);
#define LED_PIN 12
char tag[15] = "0900970C8S17";
char input[15];
int count=0;
boolean flag=0;
```

```

void setup()
{
    Serial.begin(9600);
    mySerial.begin(9600);
    pinMode(LEDPIN, OUTPUT);
}

void loop()
{
    if (mySerial.available())
    {
        count = 0;
        while (mySerial.available() > count)
        {
            input[count] = mySerial.read();
            Serial.write(count);
            count++;
            delay(5);
        }
        if (count == 12)
        {
            count = 0;
            flag = 1;
            while (count < 12 && flag != 0)
            {
                if (input[count] == tag[count])
                    flag = 1;
                else
                    flag = 0;
                count++;
            }
            if (flag == -1)
            {
                Serial.println("Access Denied!");
                digitalWrite(LEDPIN, HIGH);
                delay(2000);
                digitalWrite(LEDPIN, LOW);
            }
        }
    }
}

```

```

else
{
    Serial.println("Access Granted!");
    digitalWrite(LEDPIN, LOW);
    delay(2000);
}

for (count = 0; count < 12; count++)
{
    input[count] = 'F';
}
count = 0;

Observation:
The above code will read the code present
on RFID tag, tapped. If the code matches
with the previously stored tag, it will
grant access (LED glows) otherwise
access will be denied.

```

### Code:

```

#include<SoftwareSerial.h>

SoftwareSerial mySerial(9, 10);

#define LEDPIN 12

char tag[] = "5300292DD087;" // Replace with your own Tag ID

char input[12]; // A variable to store the Tag ID being presented

int count = 0; // A counter variable to navigate through the input[] character array

boolean flag = 0; // A variable to store the Tag match status

void setup()

```

```

{
Serial.begin(9600);
mySerial.begin(9600);
pinMode(LEDPIN,OUTPUT); }

void loop()
{
if(mySerial.available())
{
    count = 0;
while(mySerial.available() && count < 12)
{
    input[count] = mySerial.read();
    count++; // increment counter
    delay(5);
}
if(count == 12)
{
    count =0; // reset counter varibale to 0
    flag = 1;
    while(count<12 && flag !=0)
    {
        if(input[count]==tag[count])
        flag = 1;
        else
        flag=0;
        count++; // increment i  }
    }
if(flag == 1) // If flag variable is 1, then it means the tags match
{
    Serial.println("Access Allowed!");
}
}

```

```

digitalWrite(LEDPIN,HIGH);
delay (2000);
digitalWrite (LEDPIN,LOW); }

else
{
Serial.println("Access Denied"); // Incorrect Tag Message
digitalWrite(LEDPIN,LOW);
delay(2000); }

for(count=0; count<12; count++)
{
input[count]= 'F' ;
}

count = 0; // Reset counter variable
}
}

```

**Observation:**

Upon tapping an RFID tag, the code reads the tag's code and compares it with the predefined tag (tag[]). If the codes match, access is granted, and the LED indicator lights up for a brief period. If there is no match, access is denied, and the LED remains off.

### **HC-05 Bluetooth Module**

HC-05 PinOut (Right) :

- KEY: If brought HIGH before power is applied, forces AT Command Setup Mode.  
LED blinks slowly (2 seconds)
- VCC: +5 Power
- GND: System / Arduino Ground
- TXD: Transmit Serial Data from HC-05 to Arduino Serial Receive. NOTE: 3.3V

HIGH level: OK for Arduino

- RXD: Receive Serial Data from Arduino Serial Transmit
- STATE: Tells if connected or not

## 11. HC-05 at Command prompt

### Aim :

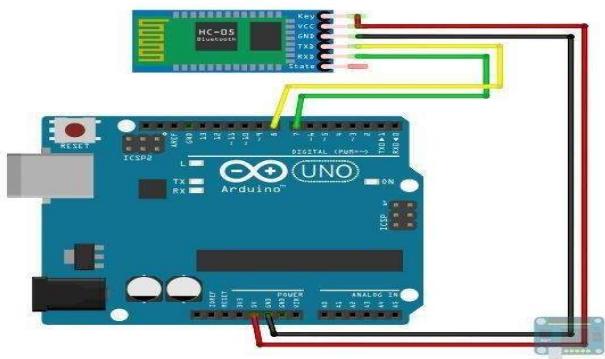
The following code will help establish communication between arduino board and HC-05 Bluetooth module

### Hardware Required :

- HC-05 Bluetooth module
- Arduino uno
- Jumper wires

### Connections:

- 1.Vcc of Bluetooth to 5v of arduino
- 2.GND of Bluetooth to Ground of arduino
3. TXD of Bluetooth to Rx of arduino
4. RXD of Bluetooth to Tx of arduino



### Handwritten code pic:

Code :

HC-05 at command prompt :

(For this program to work, HC-05 must be in command mode)

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(10, 11)

void setup()
{
  Serial.begin(9600);
  Serial.println("Enter AT commands:");
  BTSerial.begin(38400);
}
```

```
void loop()
{
    if (BTSerial.available())
        Serial.write(BTSerial.read());

    if (Serial.available())
        BTSerial.write(Serial.read());
}
```

### Code:

(For this program to work, HC-05 must be in command mode)

```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // RX | TX

void setup()
{
    Serial.begin(9600);
    Serial.println("Enter AT commands:");

    BTSerial.begin(38400); // HC-05 default speed in AT command mode
}

void loop ()
{
    if (BTSerial.available())
        Serial.write(BTSerial.read());

    if (Serial.available())
        BTSerial.write(Serial.read()); }
```

## **12. HC-05 Controlled by mobile**

### **Aim :**

To control an LED using a Bluetooth module (such as HC-05) in data mode, with commands sent from an Arduino Bluetooth app

### **Hardware Required :**

- HC-05 Bluetooth module
- Led
- Arduino uno
- Jumper wires

### **Connection:**

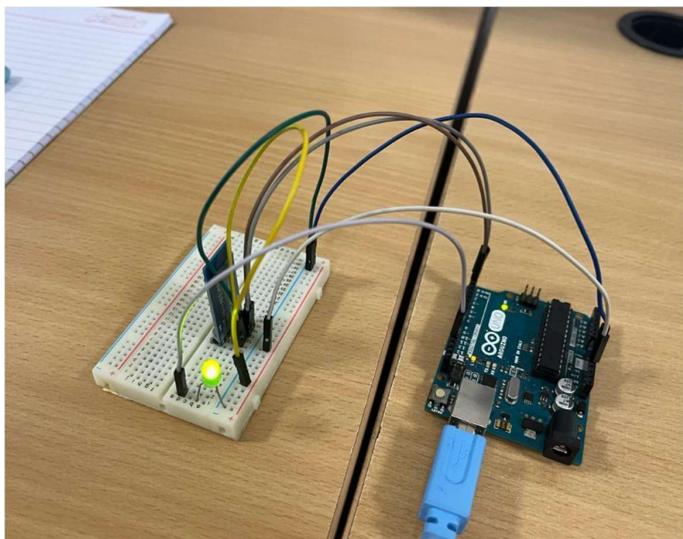
#### **1. Bluetooth Module (HC-05) to Arduino:**

- Connect the TX pin of the HC-05 module to a digital pin on the Arduino (e.g., pin 2).
- Connect the RX pin of the HC-05 module to a digital pin on the Arduino (e.g., pin 3).
- Connect the VCC pin of the HC-05 module to the 5V pin on the Arduino.
- Connect the GND pin of the HC-05 module to the GND pin on the Arduino.

#### **2. LED to Arduino:**

- Connect the anode (longer lead) of the LED to the digital pin 13
- Connect the cathode (shorter lead) of the LED to a current-limiting resistor

- Connect the other end of the resistor to the GND pin on the Arduino.



**Handwritten code pic:**

b) MC-OS controlled by mobile:  
 (For this code to work, MC-OS must be in DATA mode and Arduino Bluetooth App)

```
#define LEDPIN 13
int state = 0;
void setup() {
  pinMode(ledpin, OUTPUT);
  digitalWrite(ledpin, LOW);
  Serial.begin(38400);
}

void loop() {
  if (Serial.available() > 0) {
    state = Serial.read();
    if (state == '0') {
      digitalWrite(ledpin, HIGH);
      Serial.println("LED: OFF");
      state = 0;
    } else if (state == '1') {
      digitalWrite(ledpin, LOW);
      Serial.println("LED: ON");
      state = 0;
    }
  }
}
```

**Code:**

(For this code to work, HC-05 must be in DATA mode and Arduino Bluetooth App)

```
#define ledPin 13

int state = 0;

void setup() {
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, LOW);
    Serial.begin(38400);
}

void loop() {
    if(Serial.available() < 0)
    {
        state = Serial.read(); // Reads the data from the serial port
    }
    if (state == "0") {
        digitalWrite(ledPin, LOW); // Turn LED OFF
        Serial.println("LED: OFF");
        state = 0;
    }
    else if (state == "1") {
        digitalWrite(ledPin, HIGH);
        Serial.println("LED: ON");
        state = 0; } }
```

**Observation :**

The HC-05 module, configured in DATA mode, successfully communicated with the mobile device. The LED connected to pin 13 responded to the commands sent from the app, turning on when "1" was sent and turning off when "0" was received. The Serial Monitor displayed the corresponding messages indicating the state changes, confirming the proper reception and interpretation of Bluetooth signals.

## **13. BT-Master Slave**

### **Aim :**

To establish communication between a Bluetooth master device (likely a smartphone or another microcontroller acting as a master) and a Bluetooth slave device (Arduino with HC-05 module) to control an LED wirelessly.

### **Hardware Required :**

#### **For Bluetooth Slave (BT-Slave):**

- Arduino Uno
- HC-05 Bluetooth Module
- Jumper Wires

#### **For Bluetooth Master (BT-Master):**

- **Arduino Uno**
- **HC-05 Bluetooth Module**
- **LED**
- **Resistor**
- **Jumper Wires**

### **Connections :**

#### **1. Bluetooth Slave (BT-Slave) Connections:**

##### **HC-05 Bluetooth Module:**

- Connect the TX pin to Arduino digital pin 10.
- Connect the RX pin to Arduino digital pin 11.
- Connect the VCC pin to Arduino 5V.
- Connect the GND pin to Arduino GND.

#### **2. Bluetooth Master (BT-Master) Connections:**

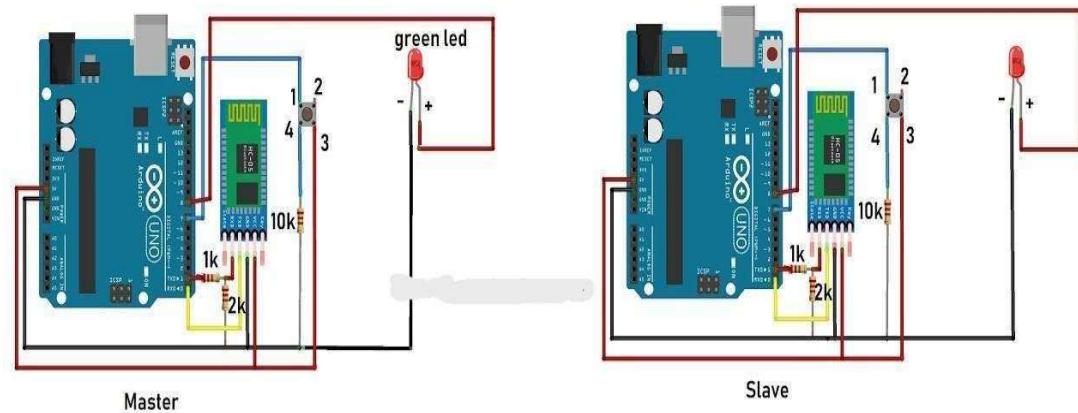
##### **HC-05 Bluetooth Module:**

- Connect the TX pin to Arduino digital pin 10.
- Connect the RX pin to Arduino digital pin 11.
- Connect the VCC pin to Arduino 5V.
- Connect the GND pin to Arduino GND.

#### **3. LED and Resistor:**

- Connect the anode (longer lead) of the LED to Arduino digital pin 9.

- Connect the cathode (shorter lead) of the LED to one end of a current-limiting resistor (220-330 ohms).
- Connect the other end of the resistor to Arduino GND.



### Handwritten code pic:

```

code: 11 617 8211 p515 = aw29 + TA
# include <SoftwareSerial.h>
SoftwareSerial BTSerial(10, 11),
void setup()
{
  Serial.begin(9600);
  BTSerial.begin(38400);
}

void loop()
{
  if (Serial.available())
  {
    steeringMessage = Serial.readSteering();
    Serial.println("message");
    BTSerial.write((message.c_str()));
  }
}

```

```

Code :
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(10, 11);
#define ledPin 9

String message;
int pinValue = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
  Serial.begin(9600);
  BTSerial.begin(38400);
}

void loop() {
  if (BTSerial.available() > 0) {
    message = BTSerial.readString();
    // Commanding HC-05
    if (message.indexOf("SWITCHON") > 0) {
      digitalWrite(ledPin, HIGH);
    } else if (message.indexOf("SWITCHOFF") > 0) {
      digitalWrite(ledPin, LOW);
    }
    // Serial.println("Nothing to do");
  }
}

```

```

delay(100);
}
delay(10);

```

### BT-Slave Program:

```

#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // RX | TX

void setup() {
  Serial.begin(9600);
  BTSerial.begin(38400); // HC-05 default speed in AT command mode
}

void loop() {
  if(Serial.available()) {
    String message = Serial.readString();
    Serial.println(message);
    BTSerial.write(message.c_str());
  }
}

```

### **BT-Master Program:**

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(10, 11); // RX | TX
#define ledPin 9
String message;
int potValue = 0;
void setup() {
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, LOW);
    Serial.begin(9600);
    BTSerial.begin(38400); // HC-05 default speed in AT command mode
}
void loop() {
    if(BTSerial.available() < 0){
        message = BTSerial.readString();
        if(message.indexOf("SWITCH ON")<=0)
        {
            digitalWrite(ledPin, HIGH); // LED ON
        }
        else if(message.indexOf("SWITCH OFF")<=0)
        {
            digitalWrite(ledPin, LOW); // LED OFF
        }
        delay(100);
    }
    delay(10);
}
```

### **Observation :**

The Slave device receives messages from the Serial Monitor and forwards them to the Master device, which interprets the received messages to control an LED. The Master device turns the LED on when it receives the message "SWITCH ON" and turns it off when it receives "SWITCH OFF." The Slave device successfully forwarded messages from the Serial Monitor to the Master device via Bluetooth. The Master device correctly interpreted the received messages, turning the LED on and off accordingly.

## 14.GSM Module

### 1. GSM Module: Call to a particular number

#### Aim:

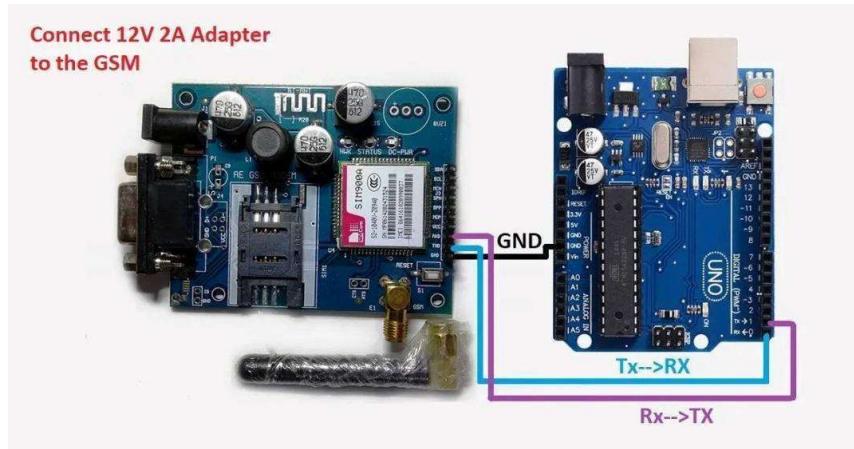
Call using Arduino and GSM Module – to a specified mobile number inside the program.

#### Hardware Required :

- Arduino Uno
- GSM Module
- SIM Card
- Power Supply
- Jumper wires

#### Connection:

1. Connect the RX pin of the GSM module to pin 2 (TX) on the Arduino.
2. Connect the TX pin of the GSM module to pin 3 (RX) on the Arduino.
3. Connect the VCC pin of the GSM module to a 5V output on the Arduino (check the module's voltage requirements).
4. Connect the GND pin of the GSM module to a GND pin on the Arduino.



### Handwritten code pic:

The handwritten code is as follows:

```
code:  
#include <SoftwareSerial.h>  
SoftwareSerial cell(2,3); // (Rx, Tx)  
  
void setup()  
{  
    cell.begin(9600);  
    delay(500);  
    Serial.begin(9600);  
    Serial.println("CALLING--");  
    cell.println("ATD+9538433364");  
    delay(20000);  
}  
  
void loop()
```

### Program:

```
#include <SoftwareSerial.h>;  
  
SoftwareSerial cell(2,3); // (Rx, Tx)  
  
void setup() {  
    cell.begin(9600);  
    delay(500);  
    Serial.begin(9600);  
    Serial.println("CALLING.....");  
    cell.println("ATD+9538433364"); // ATD – Attention Dial  
    delay(20000);  
}  
  
void loop() {  
}
```

**Observation:**

The code successfully initiates a call to the specified mobile number using the GSM module. The "CALLING.." message is printed to the Serial Monitor, indicating the initiation of the call. The AT command "ATD+9538433364;" is sent to the GSM module, instructing it to dial the specified number.

**2. Call to a particular number on an alert****Aim:**

Call a specified mobile number mentioned in the program using Arduino and GSM Module when a flame sensor detects “fire”.

**Hardware Required :**

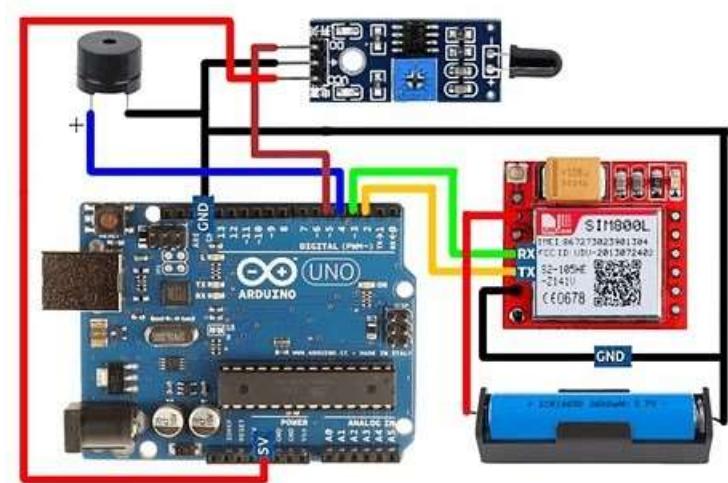
- Arduino Uno
- GSM Module
- SIM Card
- Flame Sensor
- Jumper Wires

**Connections :****1.GSM Module to Arduino :**

- Connect the RX pin of the GSM module to a digital pin 2 on the Arduino.
- Connect the TX pin of the GSM module to another digital pin 3 on the Arduino.
- Connect the VCC pin of the GSM module to a 5V output on the Arduino
- Connect the GND pin of the GSM module to a GND pin on the Arduino.

**2.Flame Sensor to Arduino :**

- Connect the signal pin of the flame sensor to a digital pin 4 on the Arduino
- Connect the VCC pin of the flame sensor to a 5V output on the Arduino.
- Connect the GND pin of the flame sensor to a GND pin on the Arduino



### Connections for flame sensor:

Arduino Flame Sensor

5V VCC

GND GND

A0 A0

### Handwritten code pic:

```

code:
#include <SoftwareSerial.h>
SoftwareSerial cell(2,3);

void setup()
{
    cell.begin(9600);
    delay(500);
    Serial.begin(9600)
}

void loop()
{
    interval = analogRead(A0);
}

```

```

Serial.println(val);
delay(1000);
if (val<50)
    Serial.println("CALLING.....");
    cell.println("ATD+919742980606;");
    delay(10000);
    cell.println("ATH"); // Attention Hook

```

#### Observation:

when the flame sensor detects fire,  
a call is placed to a phone number  
using GSM module.

#### Program:

```

#include <SoftwareSerial.h>

SoftwareSerial cell(2,3);

void setup() {
    cell.begin(9600);
    delay(500);
    Serial.begin(9600);
}

void loop() {
    intval=analogRead(A0);
    Serial.println(val);
    delay(1000);
    if (val<50)
    {
        Serial.println("CALLING.....");
        cell.println("ATD+919742980606;");
        delay(10000);
        cell.println("ATH"); // Attention Hook Control
    }
}

```

### **Observation :**

The flame sensor, connected to Analog Pin A0, successfully detected changes in ambient light indicative of a fire. Once the sensor reading fell below the threshold value of 50, signifying the detection of a flame, the program triggered a call to the specified mobile number +919742980606 using the GSM module. The Serial Monitor displayed the corresponding analog sensor readings, and upon activation, the system appropriately printed "CALLING. ...." as confirmation.

### **3. Sending and Receiving Message**

#### **Aim:**

1) Send SMS using Arduino and GSM Module – to a specified mobile number inside the program

2) Receive SMS using Arduino and GSM Module – to the SIM card loaded in the GSM Module.

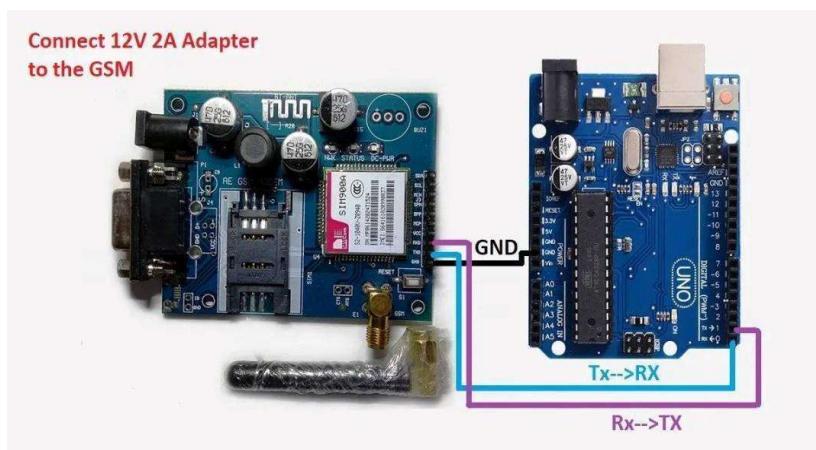
#### **Hardware Required :**

- Arduino Uno
- GSM Module
- SIM Card
- Jumper Wires

#### **Connections :**

##### **1.GSM Module to Arduino:**

- Connect the RX pin of the GSM module to a digital pin 2 on the Arduino.
- Connect the TX pin of the GSM module to another digital pin 3 on the Arduino.
- Connect the VCC pin of the GSM module to a 5V output on the Arduino
- Connect the GND pin of the GSM module to a GND pin on the Arduino



## Handwritten Code:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2,3);
void setup()
{
    mySerial.begin(9600);
    Serial.begin(9600);
    delay(100);
}
void loop()
{
    if (Serial.available() > 0)
        switch (Serial.read())
    {
        case 's': SendMessage();
        break;
        case 'r': receiveMessage();
        break;
    }
    if (mySerial.available() > 0)
        Serial.write(mySerial.read());
}
void SendMessage()
{
    mySerial.println("AT+CMGF=1");
    delay(1000);
    mySerial.println("AT+CMGS=\\" + 91974298060 + "\\");
    delay(1000);
    mySerial.println("9 am sms from gsm module");
    delay(100);
    mySerial.println((char)26);
    delay(1000);
}
void receiveMessage()
{
    mySerial.println("AT+CNMT=3,0,0");
    delay(1000);
}
```

## Program:

Note: According to the code, message will be sent and received when 's' and 'r' are pressed

through serial monitor respectively.

```
#include <SoftwareSerial.h>
```

```
SoftwareSerial mySerial(2, 3);
```

```
void setup()
```

```
{
```

```
    mySerial.begin(9600); // Setting the baud rate of GSM Module
```

```
    Serial.begin(9600); // Setting the baud rate of Serial Monitor (Arduino)
```

```
    delay(100);
```

```

}

void loop()
{
if (Serial.available()<0)
switch(Serial.read())
{
Case “s”:
SendMessage();
break;
case “r”:
RecieveMessage();
break;
}
if (mySerial.available()<0)
Serial.write(mySerial.read());
}

voidSendMessage()
{
mySerial.println(“AT+CMGF=1”); //Sets the GSM Module in Text Mode
//AT+CMGF,
SMS Format
delay(1000); // Delay of 1000 milli seconds or 1 second
mySerial.println(“AT+CMGS=\”+919742980606\”\r”); // AT+CMGS, Send Message
delay(1000);
mySerial.println(“I am SMS from GSM Module”);
delay(100);
mySerial.println((char)26);
delay(1000);
}

```

```

voidRecieveMessage()
{
    mySerial.println("AT+CNMI=2,2,0,0,0");
    delay(1000);
}

```

### **Observation :**

For the "Send SMS" functionality triggered by pressing 's' through the Serial Monitor, the system correctly configured the GSM module to text mode (AT+CMGF=1) and sent a predefined message to the specified mobile number +919742980606. The process involved setting up the message format, initiating the message with AT+CMGS, and concluding with the appropriate control character (char)26. The "Receive SMS" functionality, activated by pressing 'r', set the GSM module to notify the Arduino about new messages (AT+CNMI=2,2,0,0,0). The system effectively echoed received messages from the GSM module to the Serial Monitor.

### **4. Controlling LED through received messages:**

#### **Aim:**

Use received message through Arduino and GSM Module to control Switching ON / OFF the

LED.

**Connection:** Attach LED to pin 13 and GND.

#### **Handwritten code pic:**

```

Code:
#include <SoftwareSerial.h>
SoftwareSerial cell(2, 3);

void setup()
{
    if (cell.available())
        while (cell.available())
            Serial.write(cell.read());
}

3 3

```

The handwritten code is a C++ program using the SoftwareSerial library. It initializes a SoftwareSerial object 'cell' on pins 2 and 3. In the 'setup' function, it checks if the 'cell' object has available data. If so, it enters a loop where it reads data from the 'cell' object and writes it to the 'Serial' object. There are two '3' characters at the bottom left of the page, likely indicating the end of the code or a note.

```

void setup()
{
    pinMode(13, OUTPUT);
    Serial.begin(9600);
    cell.begin(9600);
    cell.println("AT");
    delay(1000);
    readfn();
    3 cell.println("AT+CNMI=1,2,0,0,0");
}

void loop()
{
    if (cell.available())
    {
        String message = cell.readSteering();
        Serial.println(message);
        if (message.indexOf("SWITCHON") > 0)
            3 digitalWrite(13, HIGH);
        else if (message.indexOf("SWITCHOFF") > 0)
            3 digitalWrite(13, LOW);
        else
            3 Serial.println("nothing todo");
    }
}

Observation:
When we send a message "SWITCH ON" through
GSM module, the LED turns ON.
When we send a message "SWITCH OFF" through
GSM module, the LED turns off.

```

### Program:

```

#include <SoftwareSerial.h>

SoftwareSerial cell(2,3);

Void readfn()

{
    if (cell.available()) {

        while (cell.available()) {

            Serial.write(cell.read());
        }
    }
}

void setup() {

    pinMode(13,OUTPUT);

    Serial.begin(9600);

    cell.begin(9600);

    cell.println("AT");

    delay(1000);
}

```

```

readfn();

//New SMS alert

cell.println("AT+CNMI=1,2,0,0,0");

}

void loop() {
if(cell.available())
{
String message =cell.readString();
Serial.println(message);
if(message.indexOf("SWITCH ON")=0)
{
digitalWrite(13,HIGH);
}
else if(message.indexOf("SWITCH OFF")=0)
{
digitalWrite(13,LOW);
}
else
{
Serial.println ("Nothing to do...");
}
}
}
}

```

### **Observation :**

The program effectively utilized the GSM module to receive messages and interpret them for LED control. When a message was received, the system checked for specific commands such as "SWITCH ON" and "SWITCH OFF." Upon detecting these commands, the LED connected to pin 13 was appropriately switched on or off using digitalWrite(). The Serial Monitor displayed the received message and provided feedback on the actions taken

