

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



LAB REPORT
on
INTERNET OF THINGS LAB

Submitted by

D REAVNTH (1BM21CS047)

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
JUN-2023 to SEP-2023

**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 **D REVANTH(1BM21CS047)**, 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.

Dr. Shyamala G

Assistant professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak

Professor and Head
Department of CSE
BMSCE, Bengaluru

,

Index

Sl. No.	Date	Program Title	Page No.
1.	10/11/23	LED Blinking	1
2.	10/11/23	LED ON/OFF Using Pushbutton	3
3.	17/11/23	LED Fading using Potentiometer	6
4.	17/11/23	Nightlight Simulation	9
5.	17/11/23	PIR with Arduino UNO	12
6.	24/11/23	Ultrasound with Arduino UNO	15
7.	08/12/23	Fire Alert System	19
8.	08/12/23	Automatic irrigation controller simulation	23
9.	08/12/23	Reading the code present on RFID tag	27
10.	22/12/23	Access control through RFID	31
11.	10/1/24	HC-05 Bluetooth at Command prompt	37
12.	10/1/24	HC-05 Bluetooth Controlled by mobile	40
13.	10/1/24	Bluetooth-Master Slave	42
17.	20/11/23	GSM-Module	47

1. LED Blinking

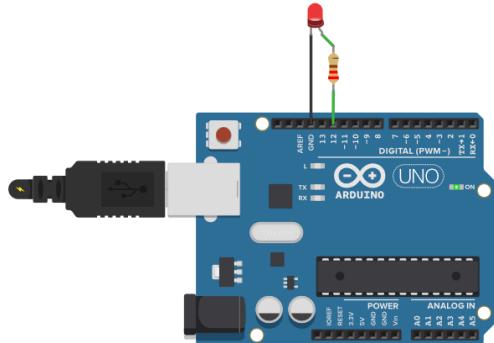
Aim: Turns on an LED on for one second, then off for one second, repeatedly

Hardware Required:

- Arduino Board
- LEDs

Connection:

1. Attach one leg (negative) of the led to ground of arduino
2. Attach other leg (positive) of led to pin 13



Handwritten code pic:

Program:

```
Void setup() {  
    pinMode(13, OUTPUT); }  
  
Void loop() {  
    digitalWrite(13, HIGH);  
    delay(1000);  
    digitalWrite(13, LOW);  
    delay(1000); }.
```



Code:

```
// Pin 13 has an LED connected on most Arduino boards  
  
int led = 13;  
  
void setup()  
{  
    pinMode(led, OUTPUT);  
}  
  
void loop() {  
    forever  
    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

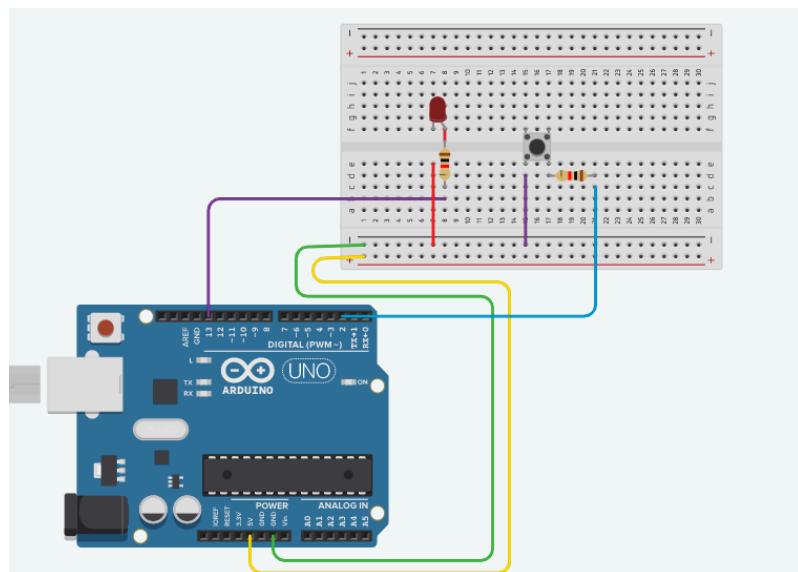
Aim: Turn an LED ON /OFF using a Pushbutton.

Hardware Required:

- Arduino Board
- LED
- Push button

Connection:

1. One end of push button is connected to 5v
2. Other end of push button is connected to ground
3. Other end is connected to digital pin 2



Handwritten code pic:

Program:

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



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

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); // Turn off the LED  
    }  
}
```

Observation :

The code effectively achieves the 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, providing a clear visual indication of the button's influence on the output. This interactive behavior enhances the user experience, creating a responsive system where the LED state is directly controlled by the push button's input.

3. LED Fading using Potentiometer

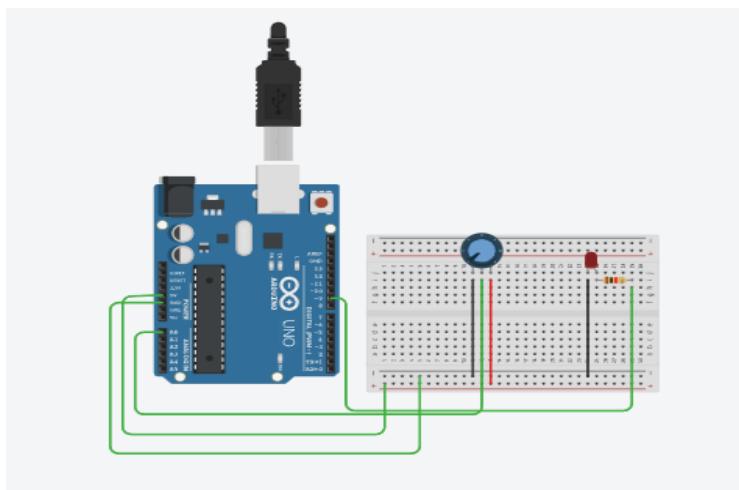
Aim: To control the brightness of an LED using a Potentiometer.

Hardware Required:

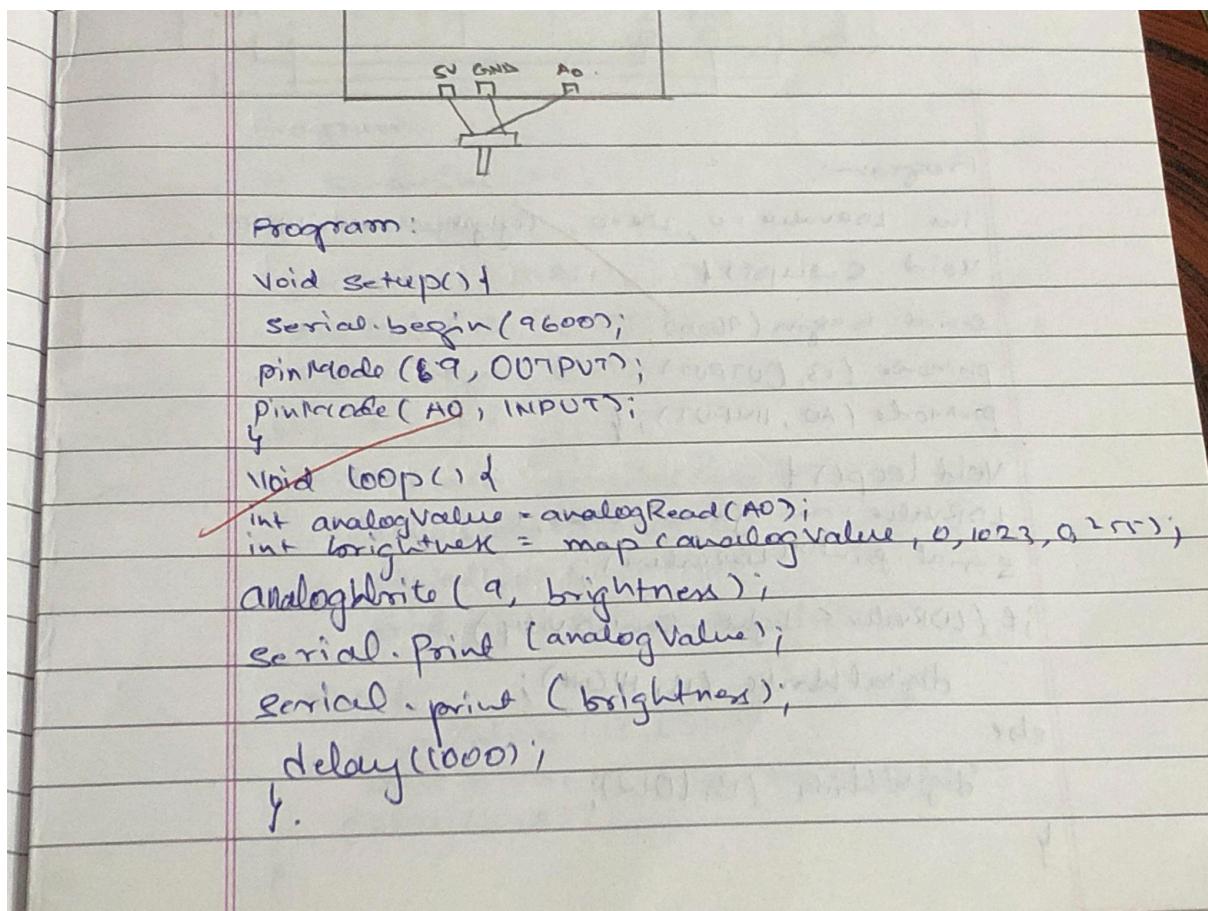
- Arduino Board
- LED
- Potentiometer

Connection:

- 1.Led's Positive leg to digital pin 9
- 2.Led's negative leg to ground
- 3.Red wire of potentiometer is connected to 5v
- 4.Black wire is connected to ground



Handwritten code pic:



Code:

```
const int potPin = A0; // Pin connected to the potentiometer  
const int ledPin = 9; // Pin connected to the LED  
void setup() {  
    pinMode(ledPin, OUTPUT); // Initialize the LED pin as an output  
}  
void loop() {  
    int potValue = analogRead(potPin); // Read the value from the potentiometer (0-1023)  
    int brightness = map(potValue, 0, 1023, 0, 255); // Map the potentiometer value to brightness  
    // (0-255)  
    analogWrite(ledPin, brightness); // Set the brightness of the LED  
}
```

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 subsequent mapping of these values to a brightness scale (0 to 255) results in a smooth and proportional adjustment of the LED's intensity.

4. Nightlight Simulation

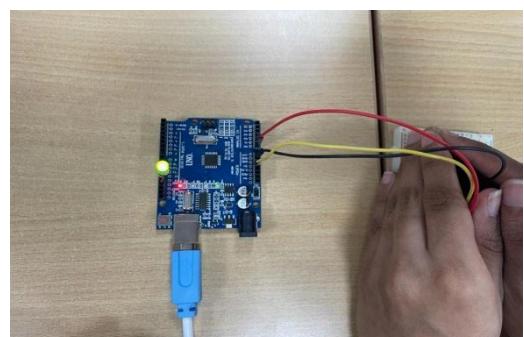
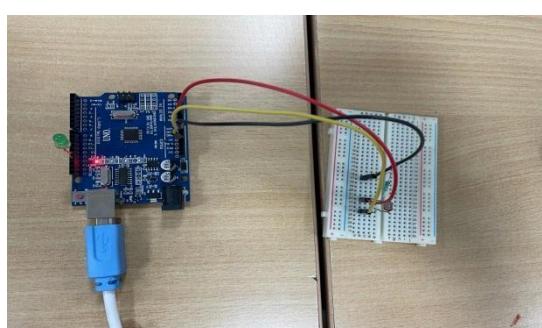
Aim: Simulating a night light using LDR and PIR

Hardware Required:

- 1 LED
- 1 LDR
- 110K register

Connection:

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



Handwritten code pic:

Program:

```

Void setup()
{
    Serial.begin(9600);
    pinMode(9, OUTPUT);
    pinMode(A0, INPUT);
}

Void loop()
{
    int analogValue = analogRead(A0);
    int brightness = map(analogValue, 0, 1023, 0, 255);
    analogWrite(9, brightness);
    Serial.print(analogValue);
    Serial.print(brightness);
    delay(1000);
}

```

Code:

```
int LDR = 0; //analog pin to which LDR is connected, here we set it to 0 so it means A0
```

```
int LDRValue = 0; //that's a variable to store LDR values
```

```
int light_sensitivity = 500; //This is the approx value of light surrounding your LDR
```

```
void setup()
```

```
{
```

```
Serial.begin(9600); //start the serial monitor with 9600 baud
```

```
pinMode(11, OUTPUT); //attach positive leg of LED to pin 11
```

```
}
```

```
void loop()
```

```
{
```

```
LDRValue = analogRead(LDR); //reads the ldr's value through LDR
```

```
Serial.println(LDRValue); //prints the LDR values to serial monitor
```

```
delay(50); //This is the speed by which LDR sends value to arduino
```

```
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 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. Conversely, if the light exceeds the threshold, the LED is turned off. The delay at the end of the loop introduces a time delay between successive readings and LED state changes

5. PIR with Arduino UNO

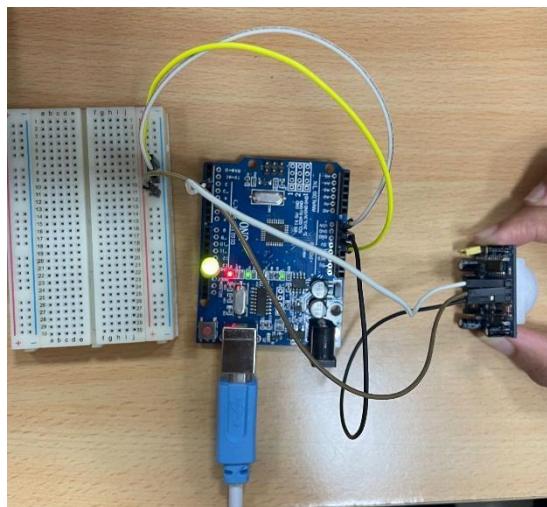
Aim: To Simulate infrared detection using PIR sensor and arduino UNO.

Hardware Required:

- Ultrasound Sensor
- Arduino Uno
- Jumper Wires

Connection:

1. Connect positive leg of LED to digital pin 13
2. Connect negative leg of LED to ground
3. Connect first leg of PIR to digital pin 2
4. Connect second leg of PIR to breadboard(a)
5. Connect first leg of PIR to ground
4. Vin of arduino to breadboard(b)
5. .5v of arduino to breadboard©



Handwritten code pic:

```
Program:  
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);  
        Serial.println("Sensor Deactivated");  
        delay(1000);  
    }  
}
```

Code:

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

```
void loop()
{
    // read the state of the sensor/digital input
    sensorState = digitalRead(2);

    // check if sensor pin is HIGH. if it is, set the
    // LED on.

    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. Conversely, when no motion is sensed, the LED is turned off. The delay of 10 milliseconds at the end of the loop helps to stabilize the sensor readings and reduce false positives.

6. Ultrasound with Arduino UNO

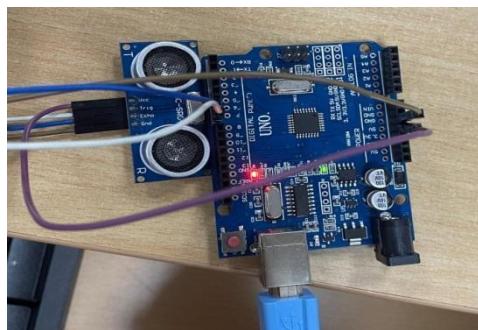
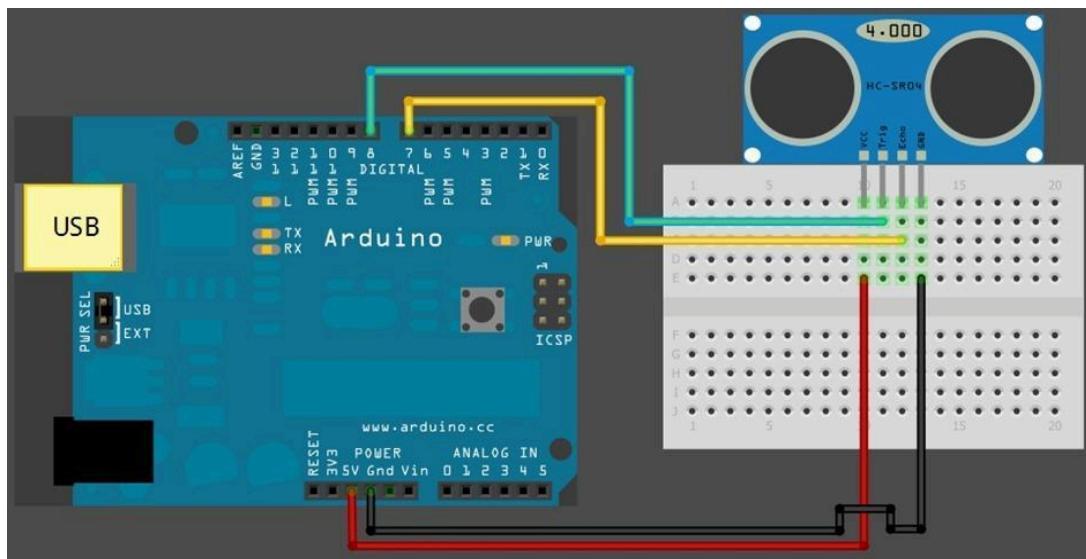
Aim: To measure distance using ultrasound sensor and Arduino UNO.

Hardware Required:

- Ultrasound sensor
- Arduino UNO
- Jump/Connection wires

Connection:

1. Connect vcc to 5v of arduino
2. Connect ground to ground of Arduino
3. Connect T of ultrasound to digital pin 7 of arduino
4. Connect ECHO of ultrasound to digital pin 11 of arduino



Handwritten code pic:

```
Program:  
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);  
        Serial.println("Sensor Deactivated");  
        delay(1000);  
    }.
```

81

```

Serial.print(inches);
Serial.print("inches");
cm = microsecondsToCentimeters(duration);
Serial.print(cm);
Serial.println("cm");
```

(long microsecondsToInches (long microseconds)
 { return microseconds / 7212; }
 (long microsecondsToCentimeters (long microseconds)
 { return microseconds / 2912; }

Observation:

Code:

```

const int pingPin = 7;
const int echoPin=6;// Trigger Pin of Ultrasonic Sensor const int echoPin = 6; // Echo Pin of
Ultrasonic Sensor
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. The microsecondsToInches() and microsecondsToCentimeters() functions convert the duration into distance measurements. The serial monitor output displays the measured distance in inches and centimeters.

7. Fire Alert

Aim: Fire alarm simulation

Hardware Required:

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

Connections:

1. Flame sensor interfacing to Arduino

Flame sensor to Arduino

vcc to vcc

gnd to gnd

A0 to A0

2. Led interfacing to Arduino

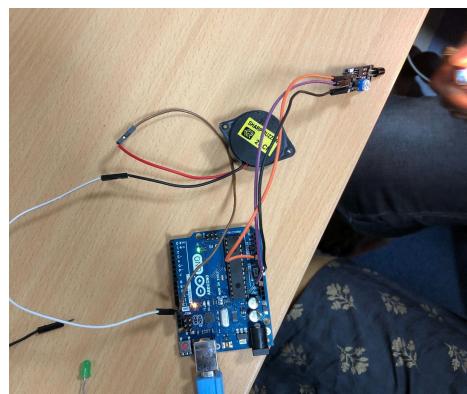
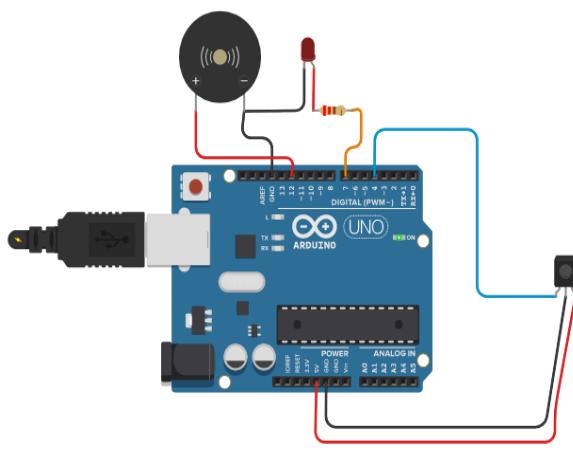
LED +ve is connected to 9th pin of Arduino

LED -ve is connected to gnd pin of arduino

3. Buzzer interfacing to Arduino

Buzzer +ve is connected to 12th pin of Arduino

Buzzer -ve is connected to GND pin of Arduino



Handwritten code pic:

Program:

```
int sensorPin = A0;  
int SensorValue = 0;  
int led = 9;  
int buzzer = 12;  
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 in LED on");  
        digitalWrite(led, HIGH);  
        digitalWrite(buzzer, HIGH);  
        delay(1000);  
        digitalWrite(led, LOW);  
        digitalWrite(buzzer, LOW);  
        delay(SensorValue);  
    }  
}
```

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() {
    // declare the ledPin and buzzer as an OUTPUT:
    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. The LED and buzzer remain active for a brief period (1 second) as part of the alarm simulation.

8. Automatic irrigation controller simulation

Aim:

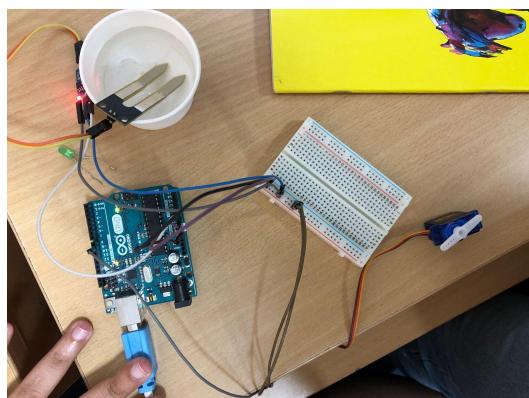
Sensing the soil moisture and sprinkling the Water simulation

Hardware Required:

- Arduino
- Moisture Sensor
- Breadboard
- Min servo motor

Connections:

- 1.Moisture sensor VCC to Arduino 5V
- 2.Moisture sensor GND to Arduino GND
- 3.Moisture sensor A0 to Arduino A0
- 4.Servo motor VCC to Arduino 5V
- 5.Servo motor GND to Arduino GND
- 6.Servo Motor Signal to Arduino digital pin 9



Handwritten code pic:

Program:

```
#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) {
        myservo.write(pos);
        delay(15);
    }
    delay(1000);
}



---



21/12/23


```

Code:

```
#include <Servo.h>

Servo myservo; // create servo object to control a servo
// twelve servo objects can be created on most boards
int pos = 0; // variable to store the servo position
int sensorPin = A0; // select the input pin for the potentiometer
int sensorValue = 0; // variable to store the value coming from the sensor
void setup() {
    myservo.attach(9); // attaches the servo on pin 9 to the servo object
    Serial.begin(9600);
}
void loop() {
    // read the value from the sensor:
    sensorValue = analogRead(sensorPin);
    Serial.println (sensorValue);
    if(sensorValue<500)
    {
        for (pos = 0; pos < 180; pos += 1) { // goes from 0 degrees to 180 degrees
            // in steps of 1 degree
            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); // tell servo to go to position in variable &#39;pos&#39;
            delay(15); // waits 15ms for the servo to reach the position
        }
    }
    delay (1000);
}
```

Observation :

The code effectively 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. The servo motor moves from 0 to 180 degrees in steps and then returns from 180 to 0 degrees. These movements represent the irrigation process.

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.

Hardware Required:

- RFID reader module
- RFID tags
- Arduino
- Jumper wires

Connection:

1.5V-Arduino 5V

2.GND-Arduino GND

3.Tx-pin 9



Handwritten code pic:

Code :

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9,10);
Void setup()
{
    mySerial.begin(9600);
    Serial.begin(9600);
}
Void loop()
{
    if (mySerial.available()>0)
        Serial.write(mySerial.read());
}
```

For multiple tags.

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9,16);
int readCount=0, tagCount=0, j=0, k=0;
```

```

Char data_temp, RFIDdata[12], data_store[10][12];
boolean disp_control;
Void setup() {
    mySerial.begin(9600);
    Serial.begin(9600); y
}
Void loop() {
    ReceiveData(); StoreData(); PrintData(); y
}
Void ReceiveData() {
    if (mySerial.available() > 0) {
        data_temp = mySerial.read();
        RFIDdata(read_count) = data_temp;
        read_count++;
    }
}
Void StoreData() {
    if (read_count == 12) {
        disp_control = true;
        for (K = Tag_Count; K <= tag_count; K++) {
            for (j = 0; j < 12; j++)
                data_store[K][j] = RFIDdata[j]; yy
        }
        read_count = 0;
        tag_count++;
    }
}
Void PrintData() {
    if (disp_control == true) {
        for (K = 0; K <= tag_count; K++) {
            for (j = 0; j < 12; j++)
                Serial.write(data_store[K][j]);
            Serial.println(); y disp_control = false; yy
        }
    }
}

```

Observation: O/p consist of characters ASCII data
 When we tap RFID tag in front of RFID module
 When it is second matches give data, access is allowed.

Code:

```
#include<SoftwareSerial.h>

SoftwareSerial mySerial(9, 10);

int count = 0;

char input[12];

boolean flag = 0;

void setup() {

Serial.begin(9600);

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. The code, when executed, continuously checks for available data on the SoftwareSerial port, captures the RFID tag code, and promptly displays it in the serial monitor.

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.

Hardware Required:

- RFID reader module
- RFID tags
- LED
- Arduino
- Jumper wires

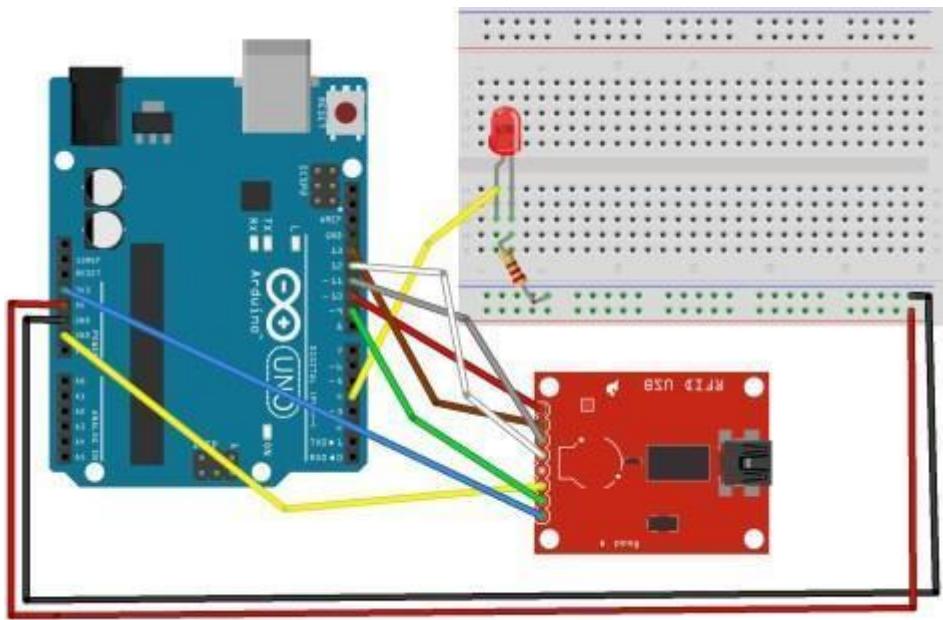
Connection:

1.5V-Arduino 5V

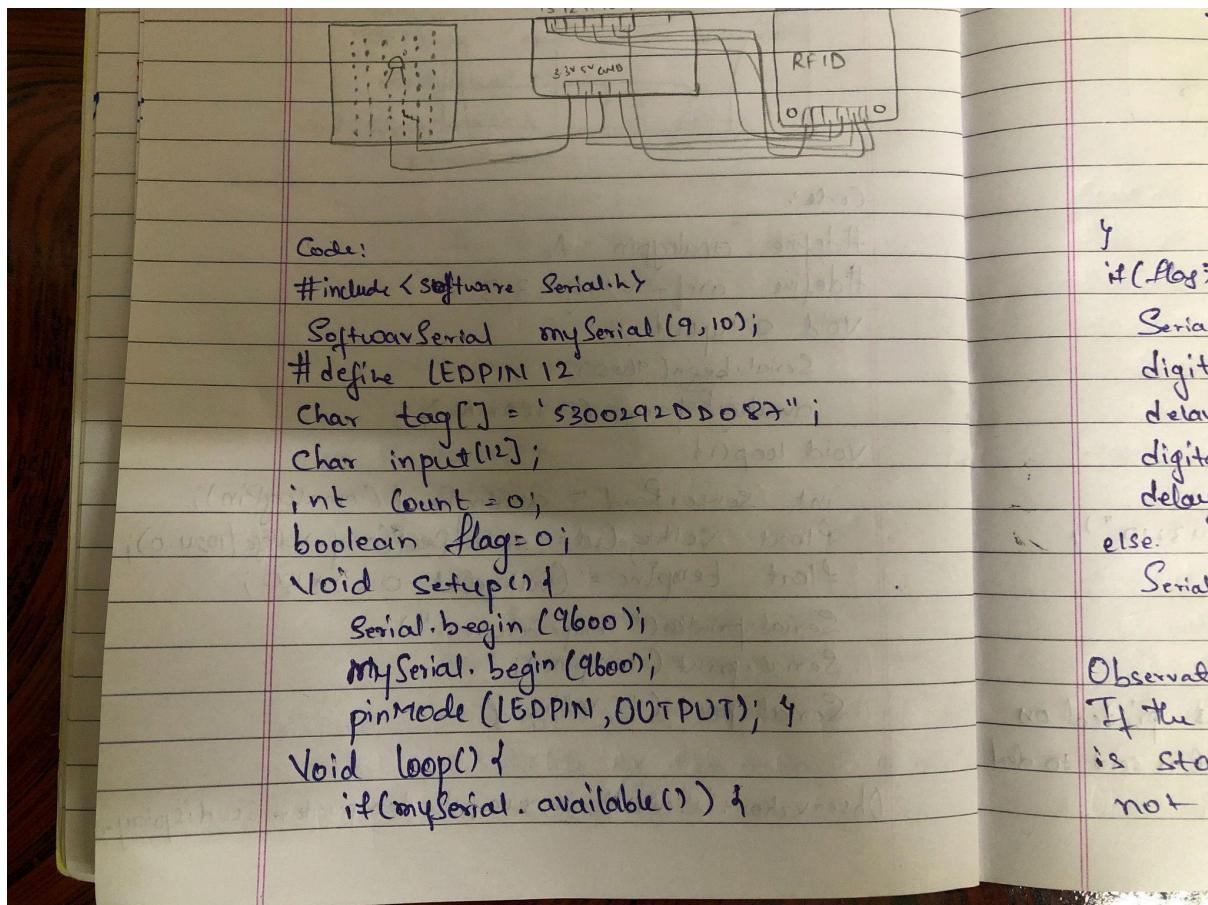
2.GND-Arduino GND

3.Tx-pin 9

4.Led-pin 12



Handwritten code pic:



```

Count = 0;
while (mySerial.available() && count < 12) {
    input[count] = mySerial.read();
    Serial.write(input[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) {
        Serial.println("Access Allowed");
        digitalWrite(LEDPIN, HIGH);
        delay(2000);
        digitalWrite(LEDPIN, LOW);
        delay(2000);
    } else {
        Serial.println("Access 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); //WRONG TAG INDICATOR
}
void loop()
{
if(mySerial.available())// Check if there is incoming data in the RFID Reader Serial
Buffer.
{
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. The output in the serial monitor provides information about the access status, whether it's allowed or denied, offering a real-time log of access attempts. The LED serves as a visual indicator, providing immediate feedback on the access control decision. This code can be expanded and adapted for various applications, such as door security systems or attendance tracking.

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 :

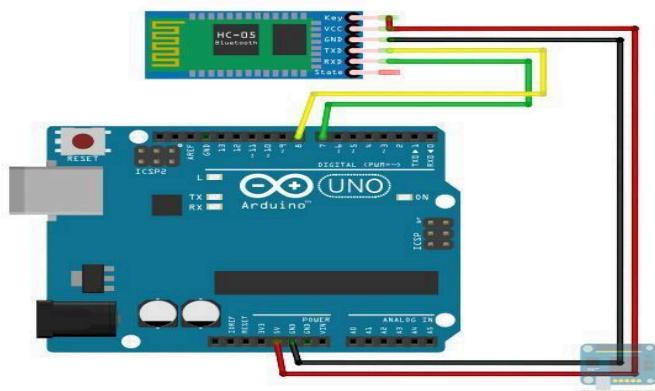
The following code will help establish communication between arduino board and HC-05 Bluetooth module

Hardware Required :

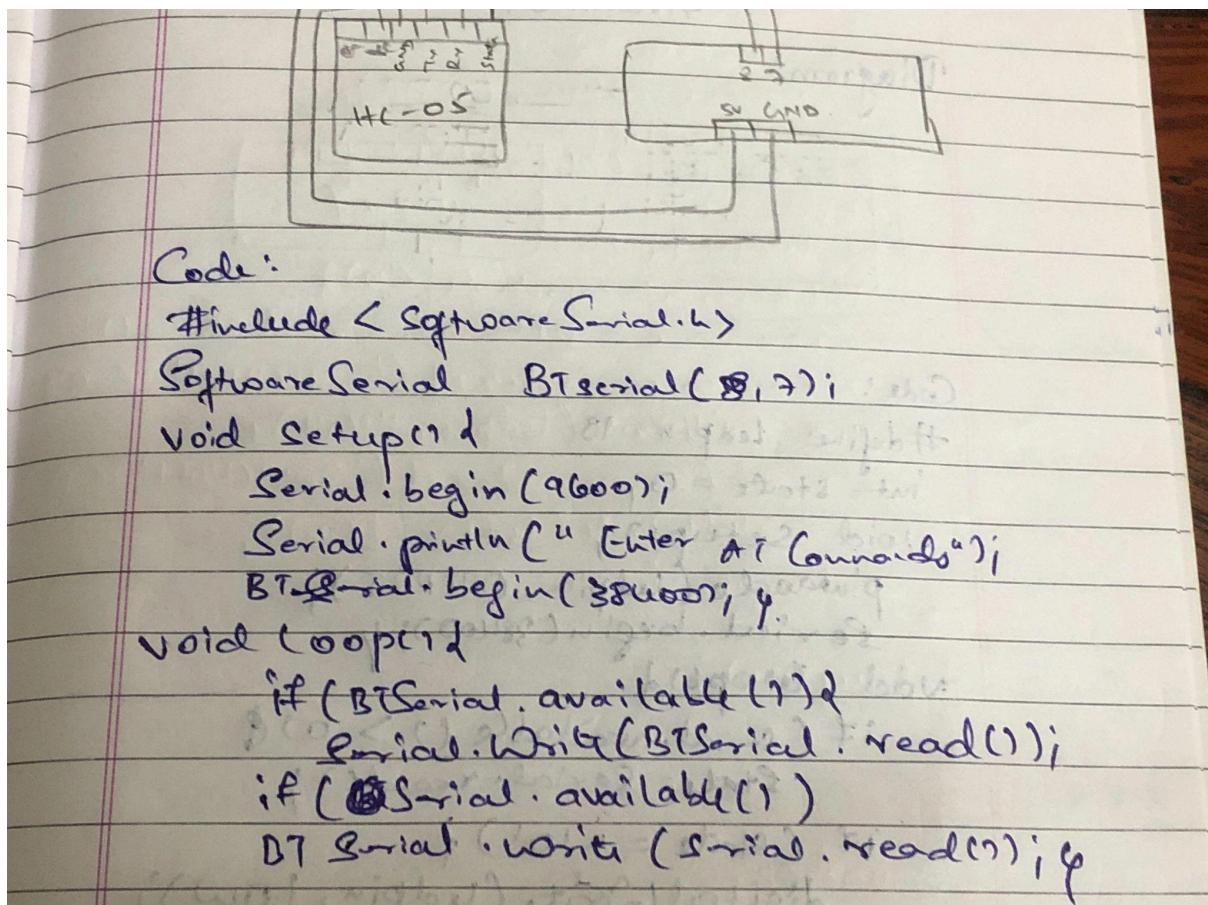
- HC-05 Bluetooth module
- Arduino uno
- Jumper wires

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:

(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());
}
```

Observation :

The code is designed for basic bidirectional communication between the Arduino and the HC-05 module using AT commands. It allows you to send commands from the Arduino to the HC-05 and receive responses, facilitating the configuration of the HC-05 module.

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:

The handwritten code is a C program for an Arduino. It defines a LED pin (13), initializes it as an output, and sets up the serial port at 9600 bps. In the loop, it checks if data is available on the serial port. If so, it reads the state (0 or 1). If the state is 1, it turns the LED off (digitalWrite(13, LOW)). If the state is 0, it turns the LED on (digitalWrite(13, HIGH)). The code also includes a comment about turning the LED off after turning it on.

```
Code:  
#define ledpin 13  
int state = 0;  
void setup() {  
    pinMode(ledpin, OUTPUT);  
    Serial.begin(9600);  
}  
void loop() {  
    if (Serial.available() > 0) {  
        state = Serial.read();  
    }  
    if (state == 1) {  
        digitalWrite(ledpin, LOW);  
        Serial.println("LED: OFF");  
        state = 0; // Turn off after turning on  
    }  
    else if (state == 0) {  
        digitalWrite(ledpin, HIGH);  
        Serial.println("LED: ON");  
        state = 1;  
    }  
}
```

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){
        // Checks whether data is comming from the serial port
        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:

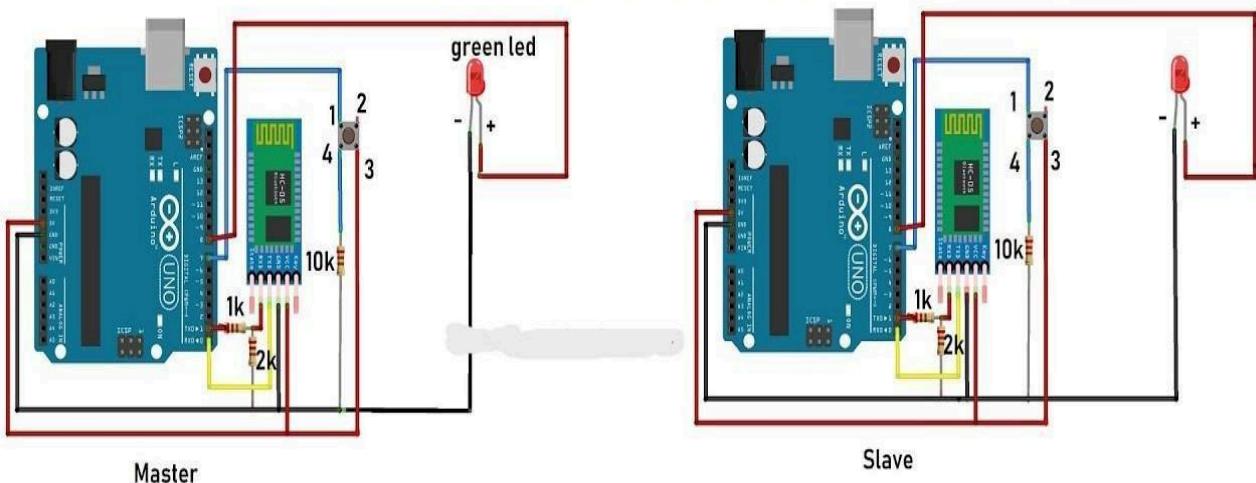
HC-05 Bluetooth Module:

- 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:

Hc-05 Vcc to arduino 5V.
 Hc-05 GND to arduino GND.
 2) LED & Resistor.
 Connect +ve of LED to arduino pin 9.
 Connect -ve of LED to resistor.
 Connect other end of resistor to GND of arduino.
BT Slave code:
 #include<SoftwareSerial.h>
 SoftwareSerial BTSerial(10, "");//pin 10
 void setup() {
 Serial.begin(9600);
 BTSerial.begin(38400);
 }
 void loop() {
 if (Serial.available()) {

```

String message = Serial.readString();
Serial.println(message);
BTSerial.write(message.c_str());

```

BT - Master Code:

```

#include <SoftwareSerial.h>
SoftwareSerial BTSerial(10, 11);
#define ledpin 9
String message;
int potValue = 0;
void setup() {
    pinMode(ledpin, OUTPUT);
    digitalWrite(ledpin, LOW);
    Serial.begin(9600);
    BTSerial.begin(38400);
}

void loop() {
    if(BTSerial.available() > 0) {
        message = BTSerial.readString();
        if(message.indexOf("SWITCHON") >= 0)
            digitalWrite(ledpin, HIGH);
        else if(message.indexOf("SWITCHOFF") >= 0)
            digitalWrite(ledpin, LOW);
        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 more
}
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. The delay(100) in the Master's loop ensured smooth processing of incoming messages. The implementation demonstrates an effective Master-Slave Bluetooth communication setup, showcasing bidirectional data transmission and control between two Arduino devices.

14. GSM Module

1. GSM Module: Call to a particular number

Aim:

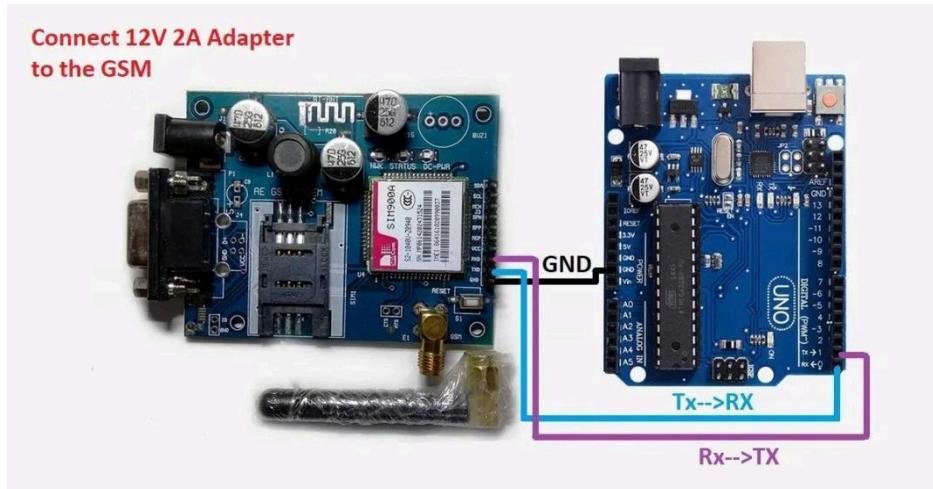
Call using Arduino and GSM Module – to a specified mobile number inside the program.

Hardware Required :

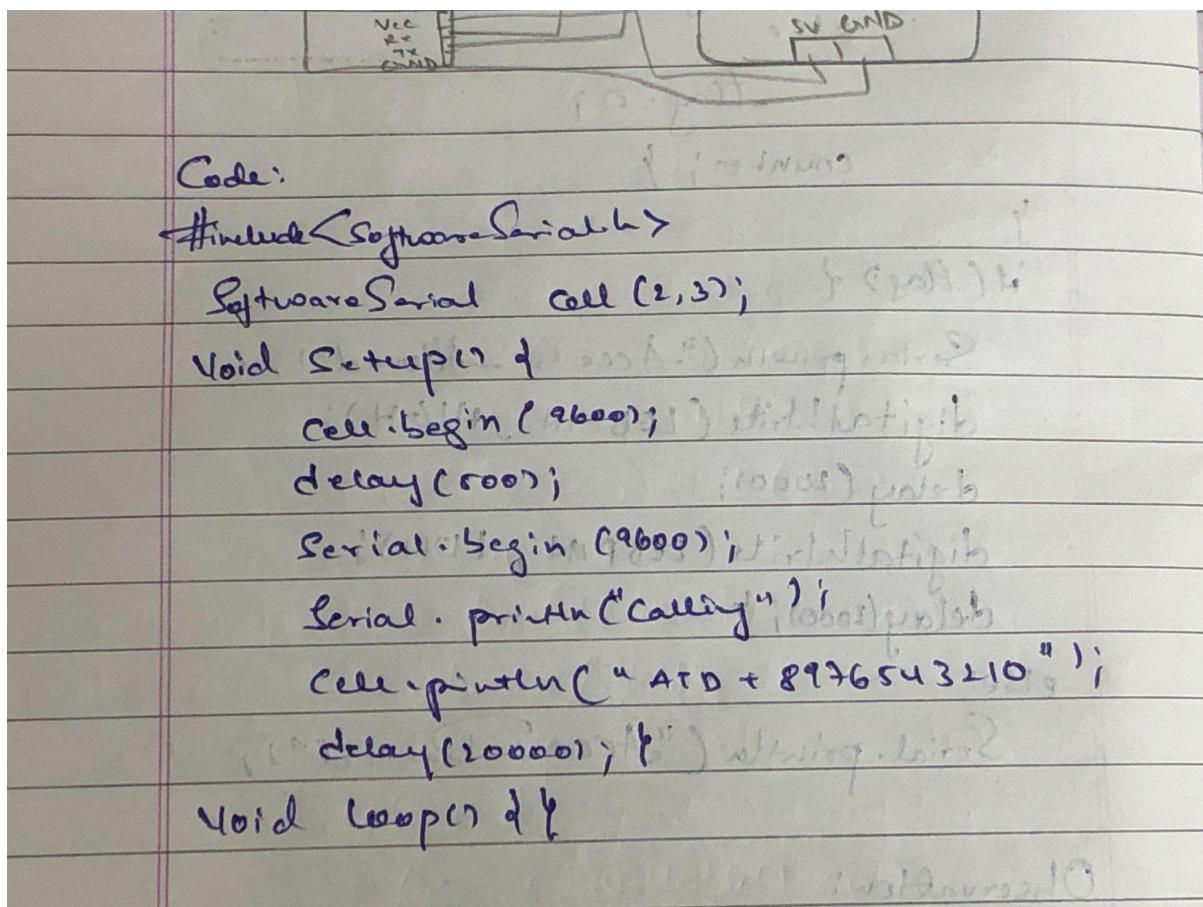
- Arduino Uno
- GSM Module
- SIM Card
- Power Supply
- Jumper wires

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:



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. The delay of 20 seconds allows for the call to be established. During this time, we observe the Serial Monitor for responses and indications of the call status.

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 :

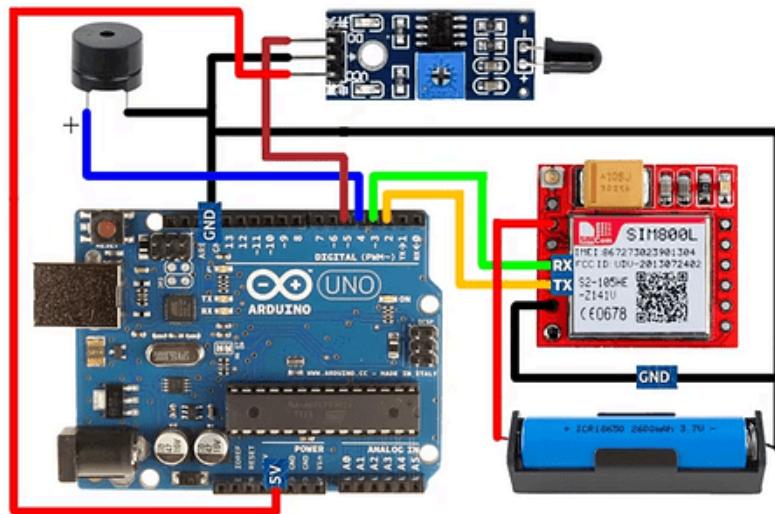
1.GSM Module to Arduino :

- 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



Handwritten Code:

```

Code:
#include <SoftwareSerial.h>
SoftwareSerial cell(2,3);
Void setup() {
    cell.begin(9600);
    delay(1000);
    Serial.begin(9600);
}
Void loop() {
    int val = analogRead(A0);
    Serial.println(val);
    delay(1000);
    if (val<50) {
        Serial.println("Callig");
        cell.println("ATD +8976543210");
        delay(10000);
        cell.println("ATH");
    }
}

```

Program:

```

#include <SoftwareSerial.h>
SoftwareSerialcell(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.

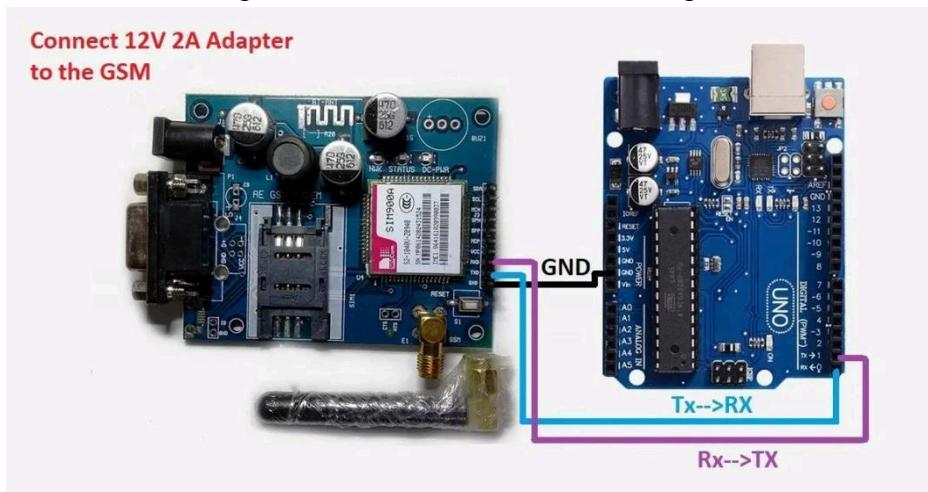
Hardware Required :

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

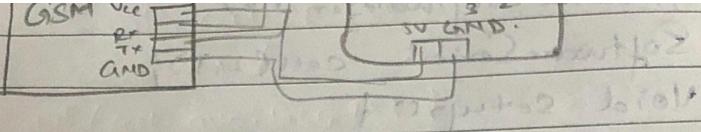
Connections :

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:



Code :

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(3,2);
void setup() {
    mySerial.begin(9600);
    Serial.begin(9600);
    delay(100);
}
void loop() {
    if (mySerial.available() > 0) {
        switch (Serial.read()) {
            case 's':
                Serial.println("Serial message");
                break;
        }
    }
}
```

```

Case 1: Receive message();
break;

if (mySerial.available < 0)
    Serial.write(mySerial.read());
}

void SendMessage() {
    mySerial.println("AT+CMGF=1");
    delay(1000);
    mySerial.println("Message from GSM");
    delay(1000);
    mySerial.println("car26");
    delay(1000);
}

void ReceiveMessage() {
    mySerial.println("AT+CNMF=2,2,0,0,0");
    delay(1000);
}

```

Observation: For send SMS by pressing 's' through Serial Monitor, the system configured GSM module to text mode & sent predefined message to Specified number. The Receive SMS is activated by pressing 'r'.

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
    // Replace with your mobile number
}

```

```

delay(1000);
mySerial.println("I am SMS from GSM Module");
// The SMS text you want to send
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.

Hardware Required :

- Arduino Uno
- GSM Module
- LED
- Resistors
- Jumper wires

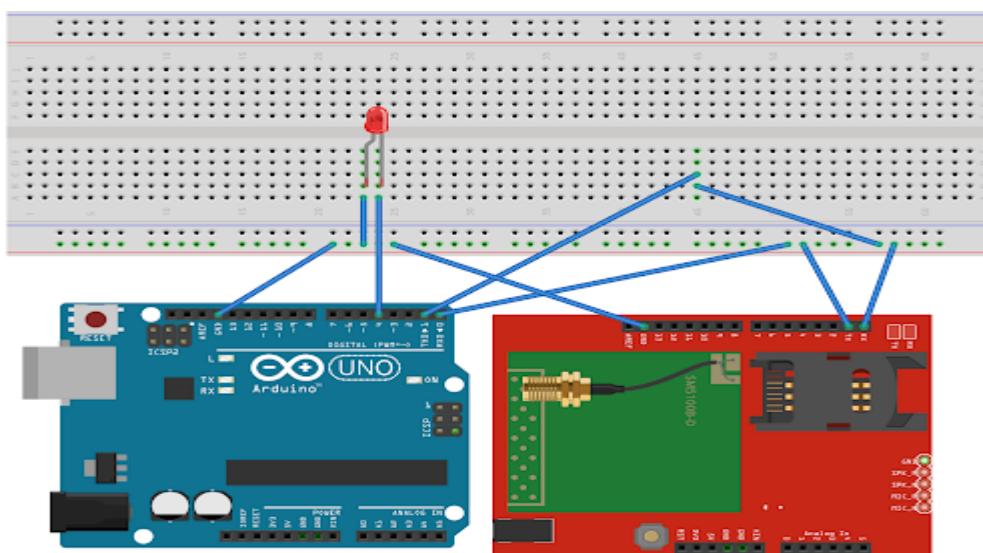
Connection :

1.Arduino to GSM Module:

- Connect Arduino TX (transmit) pin to GSM Module RX (receive) pin.
- Connect Arduino RX (receive) pin to GSM Module TX (transmit) pin.
- Connect the GSM Module GND pin to Arduino GND.
- Connect the GSM Module VCC pin to Arduino 5V.

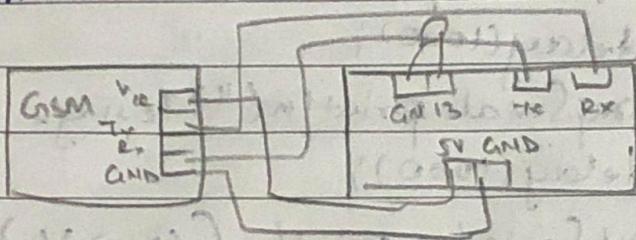
2.Arduino to LED :

- Connect the positive (longer) leg of the LED to a digital pin 13 on the Arduino
- Connect the negative (shorter) leg of the LED to a current-limiting resistor.
- Connect the other end of the resistor to the GND pin on the Arduino.



Handwritten Code:

Diagram.



Code:

```
#include <SoftwareSerial.h>
SoftwareSerial cell(2,3);
void readn() {
    if (cell.available() > 0) {
        while (cell.available() > 0) {
            Serial.write(cell.read());
        }
    }
}
void setup() {
    pinMode(13, OUTPUT);
    Serial.begin(9600);
    cell.begin(9600);
    cell.println("AT+CMND=1,2,0,0,0");
    readn();
    cell.println("AT+CMND=1,2,0,0,0");
}
```

Observation: When a message was received further checks for specific commands, LED com

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.