

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



LAB REPORT

on

INTERNET OF THINGS LAB

Submitted by

Vibha Hugar (1BM21CS255)

in partial fulfilment 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 “Computer Networks Lab” carried out by **Vibha Hugar (1BM21CS255)**, who is a 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.	18/11/23	LED Blinking	1
2.	18/11/23	LED ON/OFF Using Pushbutton	3
3.	18/11/23	LED Fading using Potentiometer	13
4.	25/11/23	Nightlight Simulation	24
5.	13/12/23	PIR with Arduino UNO	29
6.	13/12/23	Ultrasound with Arduino UNO	33
7.	13/12/23	Fire Alert System	38
8.	13/12/23	Automatic irrigation controller simulation	42
9.	20/12/23	Reading the code present on RFID tag	45
10.	20/12/23	Access control through RFID	49
11.	20/8/23	HC-05 Bluetooth at Command prompt	53
12.	27/12/23	HC-05 Bluetooth Controlled by mobile	55
13.	27/12/23	Bluetooth-Master Slave	57
14.	27/12/23	GSM Module	63

1. LED Blinking

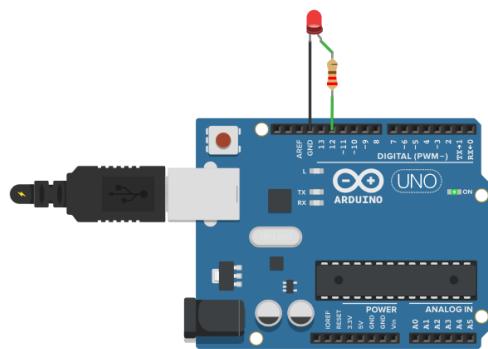
Aim:

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

Connections:

1. Attach one leg (negative) of the led to ground of arduino

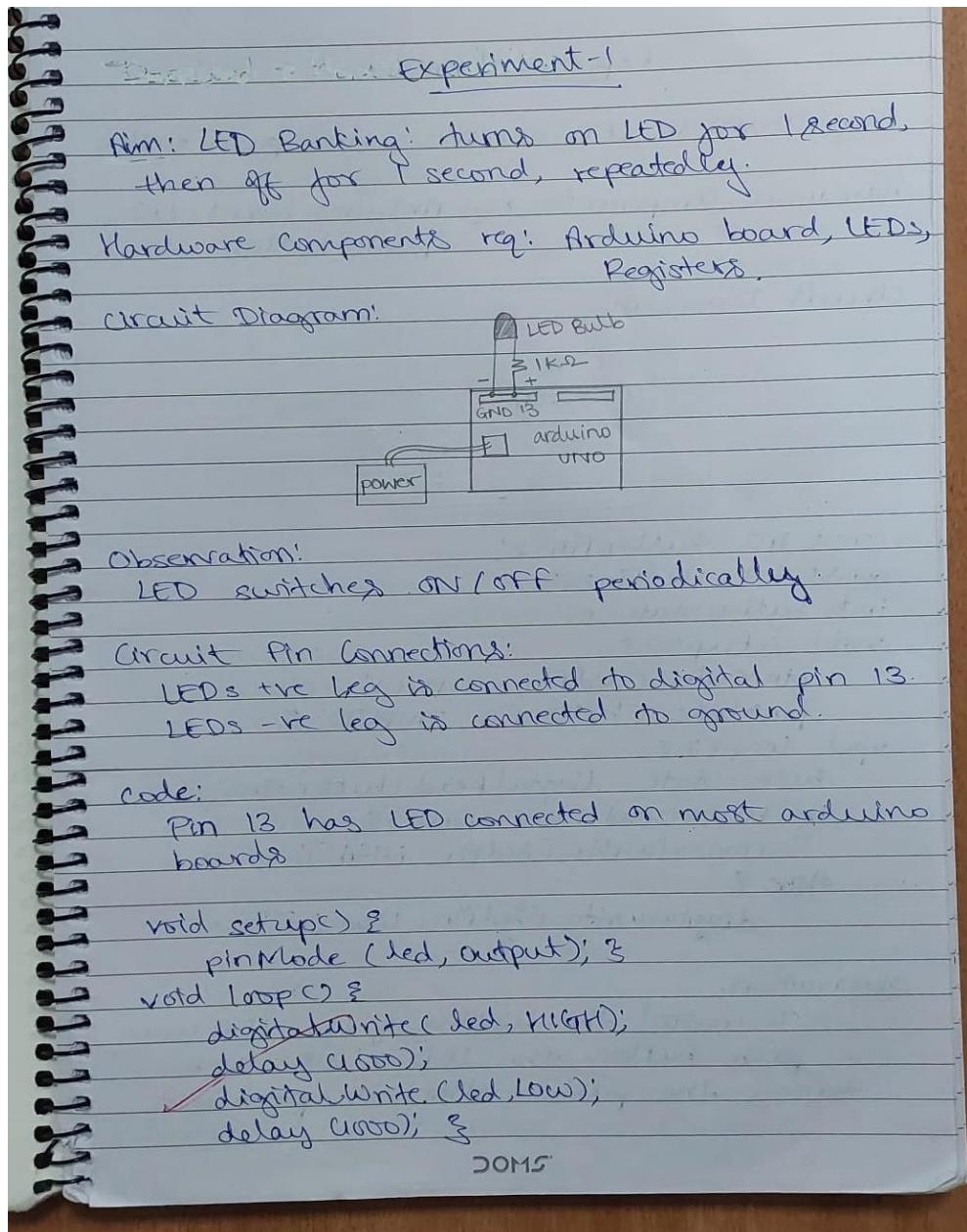
2. Attach other leg (positive) of led to pin 13



Hardware Required:

- Arduino Board
- LEDs

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 routine 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:

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

Connections:

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

Circuit diagram:

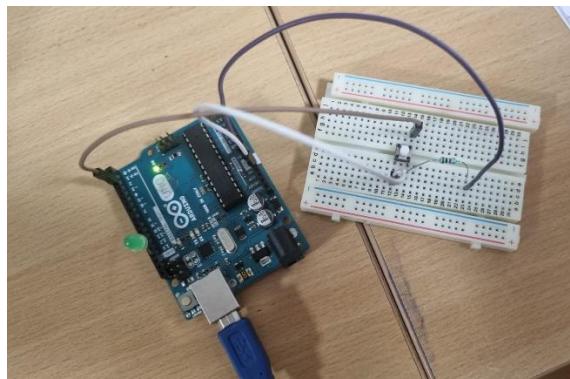


Fig.2.1. LED using push button

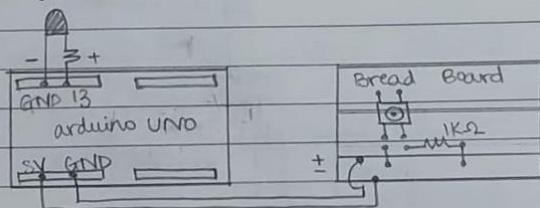
Handwritten Code Pic:

Experiment-2

Aim: Turn on LED on/off using push button

Hardware components req: Arduino board, LED, push button.

Circuit Diagram:



```
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:

To control the blinking of the LED using a push button. The LED blinks each time, the push button is pushed.

DOMS

Code:

```
const int buttonPin = 2; // Pin connected to the push button  
const int ledPin = 13; // Pin connected to the LED  
int buttonState = 0; // Variable to store the state of the push button
```

```

void setup() {
    pinMode(ledPin, OUTPUT); // Initialize the LED pin as an output
    pinMode(buttonPin, INPUT); // Initialize the push button pin as an input
}

void loop() {
    buttonState = digitalRead(buttonPin); // Read the state of the push button
    if (buttonState == HIGH) { // If the button is pressed
        digitalWrite(ledPin, HIGH); // Turn on the LED
    } else { // If the button is not pressed
        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

Connections:

- 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

5.Blue wire is connected to Analog

Circuit diagram:

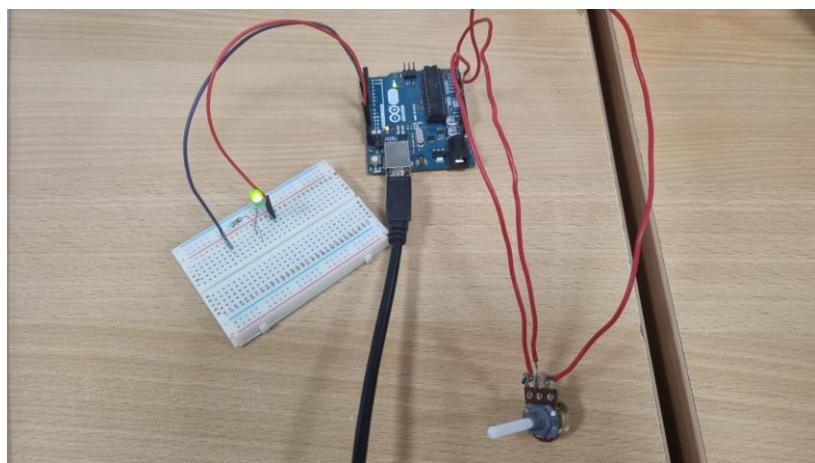


Fig. 3.1. LED ON by using potentiometer

Handwritten Code Pic:

4) LED Fading using Potentiometer

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

Hardware Required:

- Arduino board, USB cable
- LED
- Potentiometer

Circuit diagram:

- LFO shorter leg - GND, longer leg - pin 9
- Potentiometer red wire - 5V
- blue wire - A0
- black wire - GND

Code:

```
const int analogInPin = 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.print(outputValue);  
    delay(2);  
}
```

Observation: Turning the shaft of the potentiometer decreases/increases the intensity of LED light

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

Connections:

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:

Nightlight simulation

Aim: Simulating a night light using LDR & PIR

Hardware Required :-

- LDR
- LED
- 10K resistor
- Bread Board, wires, Arduino

Connection:

- Attach short LED leg - GND, long - pin 11
- One LDR leg - 5V another leg A0 (from BreadBoard)
- 1 leg resistor to LDR another to GND

Code:

```
int LDR=0;
int LDR value =0;
int light_sensitivity =500;
void setup()
{
    serial.begin(9600);
    pinMode(11, OUTPUT);
}
void loop()
{
    LDR Value =analogRead(LDR);
    serial.print(LDRValue);
    delay(50);
    if (LDRValue < light_sensitivity )
    {
        digitalWrite(11, HIGH);
    }
}
```

```
else {  
    digitalWrite(11, LOW);
```

```
    delay(1000);  
}
```

observation:

while lights are switched off in the room, it
should switch ON; else, OFF immediately

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);
}

```

Circuit diagram:

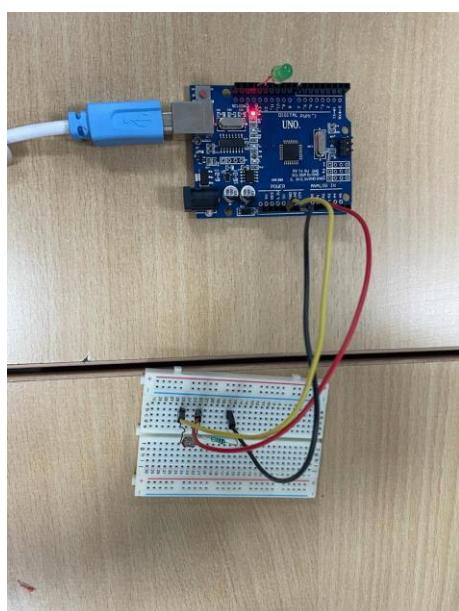


Fig 4.1- When it is bright, LED is off.

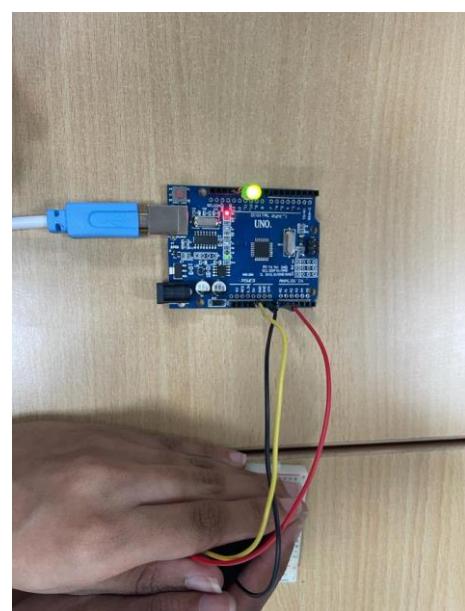


Fig- When it is dark, LED is on.

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. 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:

Simulating a motion sensor light using PIR.

Hardware Required:

- PIR
- Arduino UNO
- Bread Board
- Jumper Wires
- LED

Connections:

LED – Pin 13 and GND

PIR 1st leg - PIN 2 Arduino

PIR 2nd leg – Bread Board

PIR 3rd leg – GND Arduino

5V Arduino – Bread Board

Vin Arduino – Bread Board

2nd leg, Vin, 5V all in the same line of Bread Board in that order

Handwritten Code Pic:

PIR with Arduino UNO

Aim: simulating a motion sensor light using PIR

Hardware Required:-

- PIR
- Arduino UNO, Bread Board, wires
- LED

connections:

- Short leg (LED to GND), long leg to pin 13
- PIR using female to male,
 - 1st leg pin 2
 - 2nd leg to breadboard
 - 3rd leg to GND
- 5V of Arduino to Bread Board
- Vin of Arduino to Bread Board
- 2nd leg, Vin, 5V all in same line of Bread Board in that order.

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
  serial.println("Sensor activated!");
}
else {
  digitalWrite(13, low);
}
delay(10);
```

observation:
PIR detects motion in 360° around it.
any motion lights up the LED.

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);
}

```

Circuit diagram:

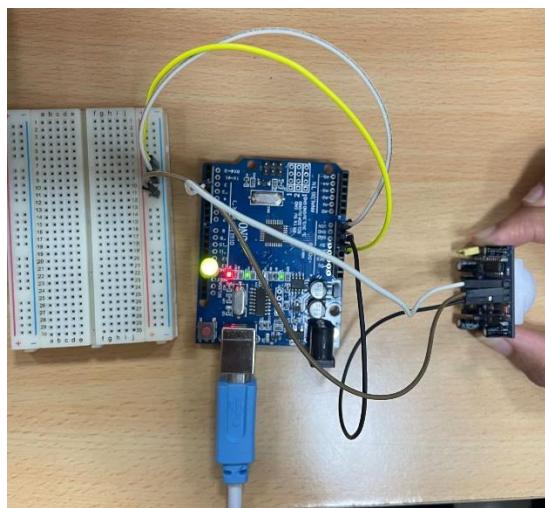


Fig 5.1- When motion is detected LED is high

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:

Measuring distance of object from ultrasound and printing it.

Hardware Required:

- Ultrasound HC-5R04
- Arduino Uno
- Wires

Connections:

VCC Ultrasound – 5V Arduino

TRIG Ultrasound – PIN 7 Arduino

ECHO Ultrasound – PIN 6 Arduino

GND Ultrasound – GNDArduino

Handwritten Code Pic:

Ultrasonics with Arduino Uno

Aim: Measuring distance of object from ultrasonic by printing it.

Hardware Required:

- Ultrasonic HC-SR04
- Arduino Uno, wires, USB

Connections:

- VCC of ultrasound - 5V arduino
- TRIG of ultrasound - digital pin 7 arduino
- ECHO of ultrasound - digital pin 6 arduino
- GND of ultrasound - GND of arduino

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");
delay (2000);
Serial.print ("inches");
cm = microsecondsToCentimeters (duration);
Serial.print (cm);
Serial.print ("cm");
delay (2000);

long microsecondsToInches (long microseconds)
{
    return microseconds / 71 / 2;
}

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

Observation -
The distance of the object is continuously
printed in serial monitor in both cm/inches.
Moving the object, distance can be changed &
different outputs can be observed.

```

✓

22/11/17

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;
}
```

Circuit diagram:

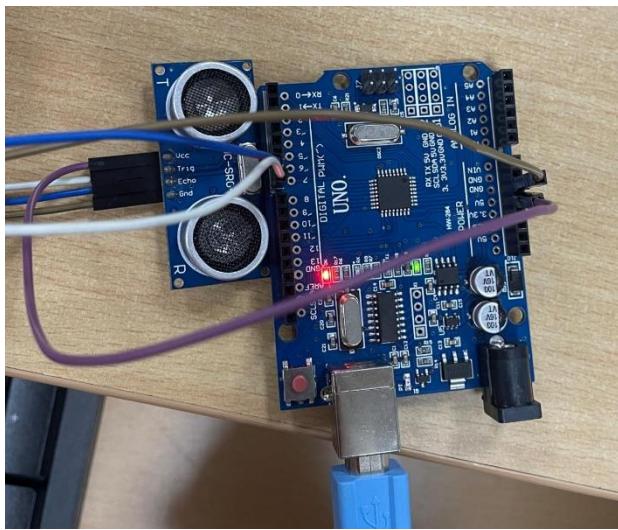


Fig 6.1-Measures distance of nearest object

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:

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

Handwritten Code Pic:

Fire Alert

Aim: Design and implement fire alarm system using flame sensor and buzzer

Hardware Requirements:

- Flame Sensor (Analogue Output)
- Arduino, Bread Board, wires, USB
- LED
- Buzzer

Connections:

- LED long +ve to 9th pin of Arduino
- LED -ve to ground pin of Arduino
- Flame sensor Vcc to Arduino Vcc (5V)
- Flame sensor ground to Arduino ground
- Flame sensor AO to Arduino AO
- Buzzer +ve to 12th pin of Arduino
- Buzzer -ve to Ground pin of Arduino

Code:

```
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");
    serial.println("LED on");
    digitalWrite(led, HIGH);
    digitalWrite(buzzer, HIGH);
    delay(1000);

    digitalWrite(led, LOW);
    digitalWrite(buzzer, LOW);
    delay(sensorValue);
}

```

Observation:

When the sensor value dips below 100 it
detects flame. It prints Fire Detected, LED on
if the sensor value is 0.
Normally it prints values in the range of 90
above.
As there is a flame, the buzzer buzzes and
the LED turns on.

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);
}
```

Circuit diagram:

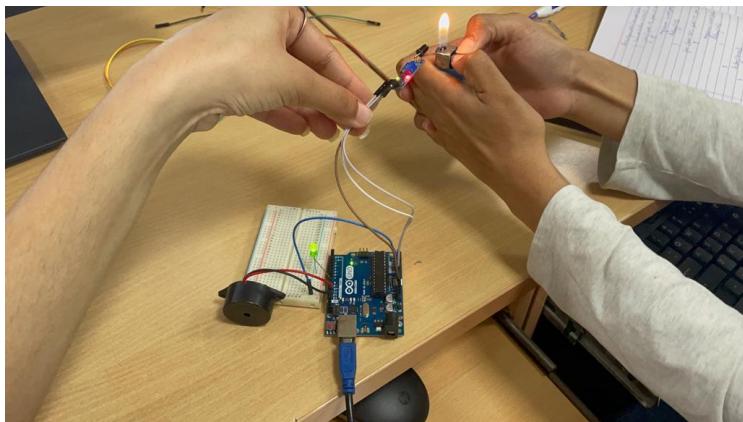


Fig 7.1- When fire is detected LED is on

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:

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

Handwritten Code Pic:

Automatic Irrigation controller simulation

Aim : Design and implement smart irrigation system using soil moisture sensor and servo motor for sprinkling the water simulation

Hardware Required :

- Arduino, BreadBoard, wires, USB
- Moisture sensor
- Mini servo Board
- Water cup

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

Code :

```
#include <sensor.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) // Put 1000 for
{ for (pos=0; pos <= 180; pos+=1)
    {
        myServo.write(pos);
        delay(15);
    }
    for (pos=180; pos>=0; pos-=1)
    {
        myServo.write(pos);
        delay(15);
    }
    delay(1000);
}

```

Observation :

We observe that when we place the moisture detector in a cup of water, the servo motor stops rotating.

Once we remove it, the moisture level drops and the servo ^{motor} starts rotating again, simulating a sprinkler system.

S.T.
29/11/23

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 'pos'
            delay(15); // waits 15ms for the servo to reach the position
        }
    }
    delay (1000);
}

```

Circuit diagram:

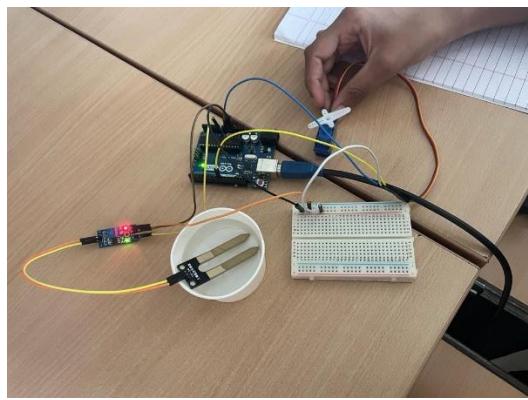


Fig 8.1- When moisture detected LED High, else Servo motor is on

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

Connections:

5V-Arduino 5V

GND-Arduino GND

Tx-pin 9

Circuit Diagram:



Fig. 9.1. RFID with tag

Handwritten Code Pic:

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);
    }
}
```

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);                      // Print RFID tag number
    }
}

```

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

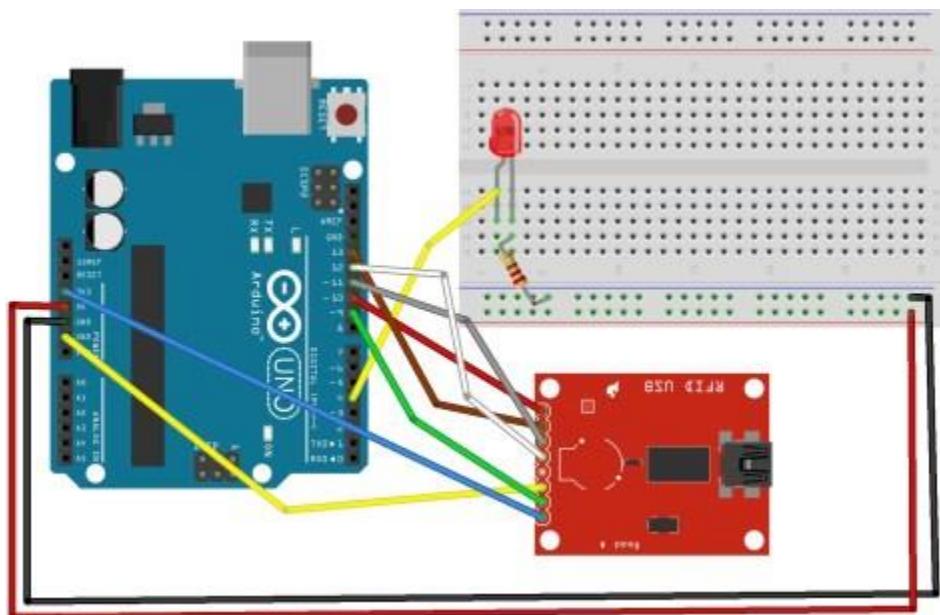
Connections:

5V-Arduino 5V

GND-Arduino GND

Tx-pin 9

Led-pin 12



Handwritten Code Pic:

Code:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9, 10);
#define LEDPIN 12
char tag[] = "530029200087";
char input[12];
int count = 0;
boolean flag = 0;
void setup()
{
    Serial.begin(9600);
    mySerial.begin(9600);
    PinMode(LEDPIN, OUTPUT);
}
```

Date _____
Page _____

```
void loop() {
    if (mySerial.available()) {
        count = 0;
        while (mySerial.available() && count < n) {
            input[count] = mySerial.read();
            Serial.write(input[count]);
            count++;
            delay(5);
        }
        if (count == n) {
            count = 0;
            flag = 1;
            while (count < n && flag != 0) {
                if (input[count] == flag) {
                    flag = 1;
                } else {
                    flag = 0;
                }
                count++;
            }
        }
        if (flag == 1) {
            serial.println("Access Allowed!");
            digitalWrite(LEDPIN, HIGH);
            delay(2000);
            digitalWrite(LEDPIN, LOW);
            delay(2000);
        }
        for (count = 0, count < n; count++) {
            input[count] = 'F';
        }
        count = 0;
    }
}
```

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++;
}

}

if(flag == 1)
{
    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:

To facilitate communication with the HC-05 Bluetooth module in command mode using an Arduino.

Hardware Required :

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

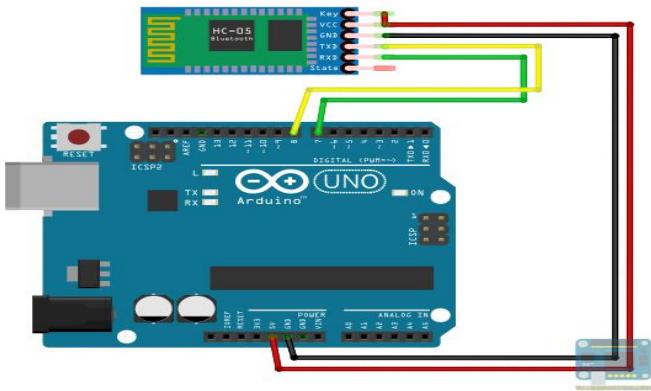
Connections:

VCC Bluetooth – 5V Arduino

Txd Bluetooth – 0 pin Arduino

Rxd Bluetooth – 1 pin Arduino

GND Bluetooth – GND Arduino



Handwritten Code Pic:

Code:

HC-05 at Command prompt:
in command mode.

```
#include <SoftwareSerial.h>
SoftwareSerial BTserial(10, 11);
void setup()
{
    Serial.begin(9600);
    Serial.println("Enter AT command:");
    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());
}

```

Observation:

This program enables users to interact with the HC-05 module in command mode through the Arduino's Serial Monitor. The Arduino acts as a transparent interface, relaying AT commands entered in the Serial Monitor to the HC-05 module and displaying the module's responses. This facilitates the configuration and customization of the HC-05 module for specific Bluetooth communication requirements.

12. HC-05 Controlled by mobile

Aim:

Establish a successful communication link between an Arduino board and a mobile device using the HC-05 Bluetooth module in data mode, enabling remote control of an LED through a dedicated Arduino Bluetooth app.

Hardware Required :

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

Connections:

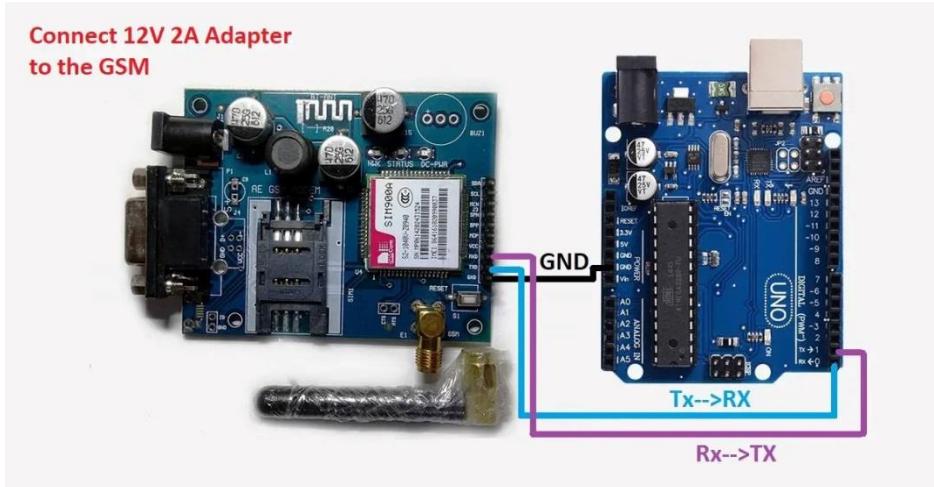
VCC Bluetooth – 5V Arduino

Txd Bluetooth – 0 pin Arduino

Rxd Bluetooth – 1 pin Arduino

GND Bluetooth – GND Arduino

LED – Pin 13 and GND



Handwritten Code Pic:

Microcontroller controlled by mobile:

Microcontroller in Data Mode of Arduino Bluetooth

```
#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, LOW);  
    Serial.println("LED: OFF");  
    state = 0;  
}
```

```
else if (state == '1')
```

```
{  
    digitalWrite(ledPin, HIGH);  
    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);
    // Default communication rate of the Bluetooth module
}

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: Design and implement a system to realize Bluetooth Master/Slave scenario.

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

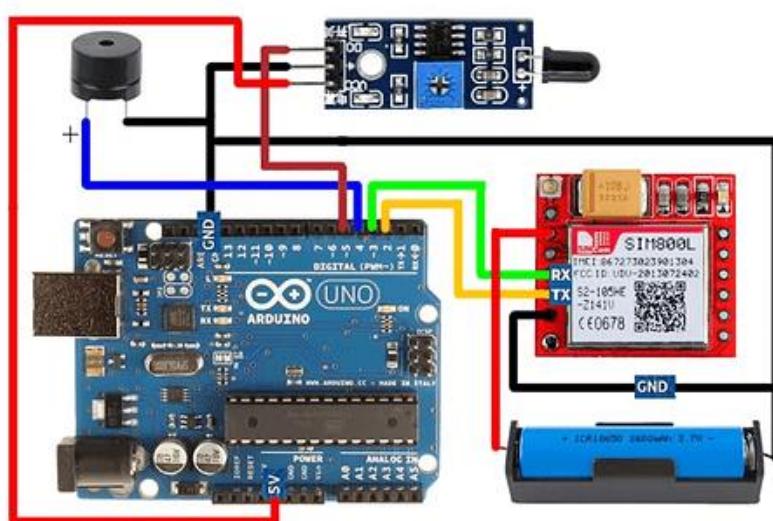
Connections:

VCC Bluetooth – 5V Arduino

Txd Bluetooth – 0 pin Arduino

Rxd Bluetooth – 1 pin Arduino

GND Bluetooth – GND Arduino



Handwritten Code Pic:

BT - Slave Program:

```
#include <SoftwareSerial.h>
SoftwareSerial BTserial(10,11);
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 Program:

```
#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("switch on") >= 0)
{
    digitalWrite(ledPin, HIGH);
}
else if (message.indexOf("switch off") >= 0)
{
    digitalWrite(ledPin, LOW);
}
else {
    serial.println("Nothing to do");
}
delay(100);
}
delay(10);

```

AT Commands:

Slave Mode:

The HC-05 bluetooth module can also act as a slave. There are fewer commands to set this up:

AT+ORGL Reset to defaults

AT+RMAAD Clear any paired devices

AT+ROLE=0 Set mode to SLAVE

AT+ADDR Display SLAVE address //+ADDR:98d3:33:807822

Master Mode:

To configure the module as Bluetooth Master and to pair with another bluetooth module follow these steps. First we need to put the module into command mode.

Enter these commands in order:

AT+RMAAD Clear any paired devices

AT+ADCN

AT+ROLE=1 Set mode to Master

AT+CMODE=0 Allow master to ONLY connect to bound address (slave). This allows the master to automatically connect to the slave when switched on

AT+PSWD=1234 Set PIN. Should be the same as the slave device

AT+BIND=<address> Set bind address to the slave address

AT+LINK=<address> Connect to slave.

AT+INIT

Note: If it shows any error, then check if both the Bluetooth modules are blinking in sync. If so then both the Bluetooth modules are synchronized.

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. 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 :

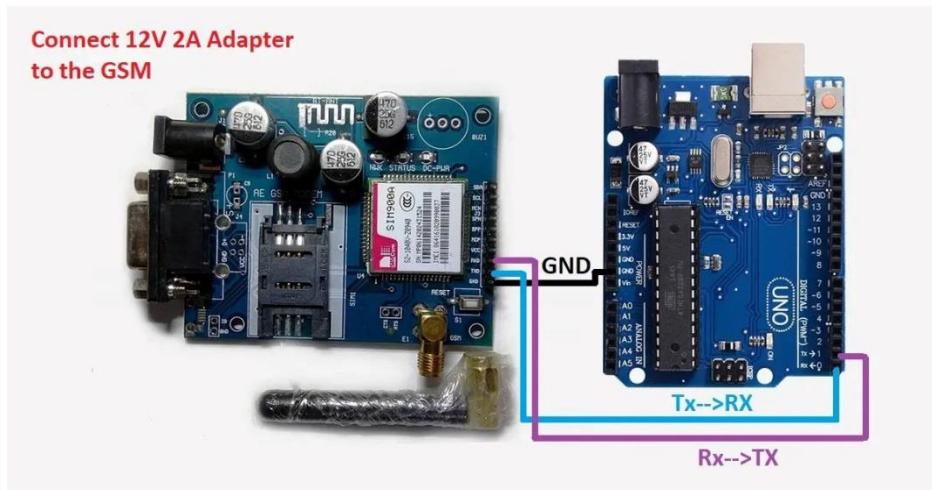
- Arduino Uno
- GSM Module
- SIM Card
- Power Supply
- Jumper wires

Connections:

GSM Tx - Arduino Rx

GSM Rx - Arduino Tx

GSM Ground - Arduino Ground



Handwritten Code Pic:

GSM Module

1) Aim: Call using Arduino and GSM module - to a specified mobile number inside the program.

Connection:

GSM TX → Arduino RX
RX → TX
GND → GND

f count < 12

1. read()

Code:

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

void loop() { }

Observation:

We insert a sim into GSM module & try calling the GSM module, if the ringtone can be heard the GSM module is said to be working.

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;"); // 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

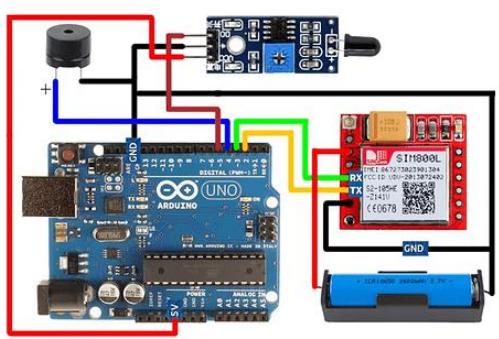
Connections for flame sensor:

Arduino - Flame Sensor

5V - VCC

GND - GND

A0 - A0



Handwritten Code Pic:

2) Aim: call a specified mobile number in the program using Arduino & GSM module when a flame sensor detects fire.

connection:

Arduino	flame sensor
5V	VCC
GND	GND
A0	A0

Program:

```
#include <SoftwareSerial.h>
SoftwareSerial cell(2,3);
void setup() {
    cell.begin(9600);
    delay(500);
    Serial.begin(9600);
}
```

Void loop()

```
{ int val = analogRead(A0);
    Serial.println(val);
    delay(1000);
    if (val < 50)
}
```

```
    Serial.println("(CALLING...")
```

```
    cell.println("ATD+919742986666");
    delay(10000);
}
```

```
    cell.println("ATM");
}
```

Code:

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

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

```
void setup() {
```

observation
when te
let up
from

3) Aim: ca
specifi

Reive s
card l
connec
GSM

code:

```
# inclu
Software
void s
{ }
```

Void

```

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

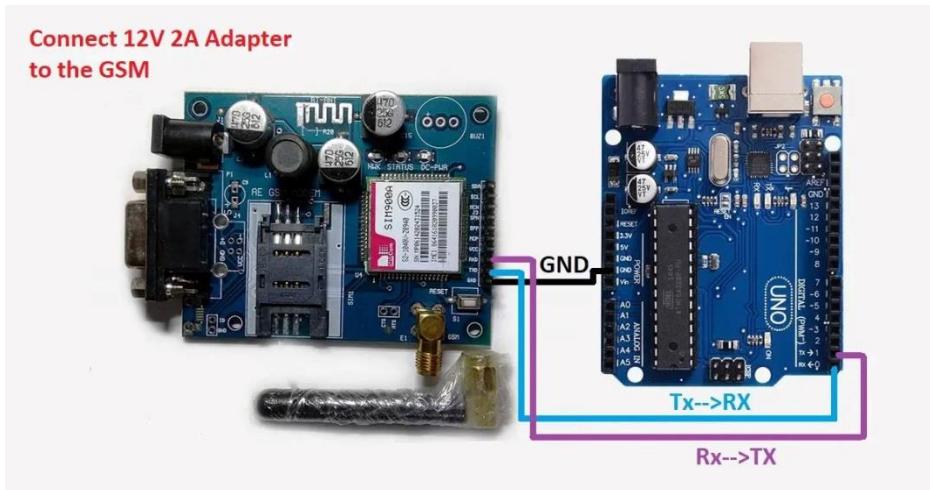
- GSM Module
- SIM Card
- Jumper Wires

Connections:

GSM Tx - Arduino Rx

GSM Rx - Arduino Tx

GSM Ground - Arduino Ground



Handwritten Code Pic:

observation:

when the flame sensor detects fire, the led will be lit up and a notification via call is made from the arduino to the specified mobile number

- 3) Aim: send SMS using Arduino & GSM Module - to a specified Mobile number track the progress

Receive SMS using Arduino & GSM module - to the SIM card loaded in the GSM Module

connection:

GSM TX → Arduino RX
RX → TX
GND → GND

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=1" + 9/9709116);
    delay(1000);
    mySerial.println("I am SMS from GSM
    module");
    mySerial.println((char)26);
    delay(1000);
}

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

```

Observation

We receive and send messages from Arduino
 GSM module. If in the serial monitor
 we press 'r' we receive a message.
 If we press 's' a message can be sent from
 GSM module.

Code:

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

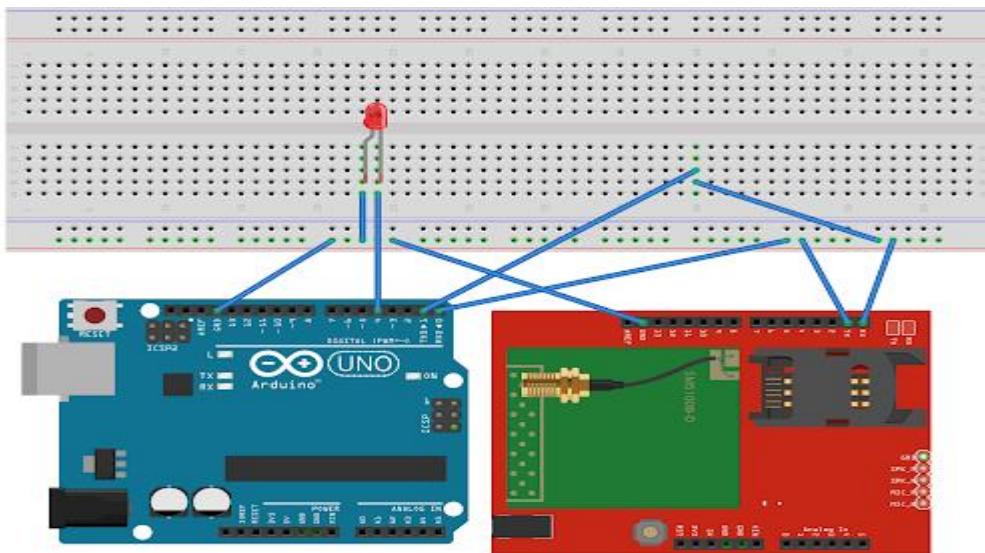
Connections:

Attach LED to pin 13 and GND.

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.



Handwritten Code Pic:

Q) Aim: Use received message through Arduino GSM module to control switching ON/OFF the LED.

Connection:

Attach LED to pin 13 of GND

Code:

```
#include <SoftwareSerial.h>
SoftwareSerial cell(2,3);
void readfun()
{
    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);
  readfun();
  cell.println("AT+CNMI=1,2,0,0,0");}
```

void loop()

```
{ if (cell.available())
    {
        string message = cell.readString();
        serial.println(message);
        if (message.indexOf("SWITCHON"))
            }
```

```
    } digitalwrite(13, HIGH);
} else if (message.indexOf("SWITCH OFF") > 0)
{
    digitalwrite(13, LOW);
}
else
{
    serial.println("Nothing to do...");
```

}

Observation:

We are able to control if LED is switched off/on by sending a message from phone 'SWITCH ON' to GSM module & similarly 'SWITCH OFF' for switching off LED.

Code:

```
#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.