VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

INTERNET OF THINGS LAB

Submitted by

DHRUVA S (1BM21CS057)

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” was carried out by **DHRUVA S (1BM21CS057),** 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. Gauri Kalnoor
Assistant professor
Department of CSE BMSCE,Bengaluru

Dr. Jyothi S Nayak
Professor and Head Department of CSE
BMSCE, Bengaluru

Index

Sl. No.	Program Title	Page No.
1.	**LED Blinking**	1
2.	**LED ON/OFF Using Pushbutton**	3
3.	**LED Fading using Potentiometer**	6
4.	**Nightlight Simulation**	8
5.	**PIR with Arduino UNO**	11
6.	**Ultrasound with Arduino UNO**	14
7.	**Fire Alert System**	17
8.	**Automatic irrigation controller simulation**	20
9.	**Reading the code present on RFID tag**	23
10.	**Access control through RFID**	25
11.	**Bluetooth module**	29
12.	**GSM Module**	35

1. LED Blinking

Aim:

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

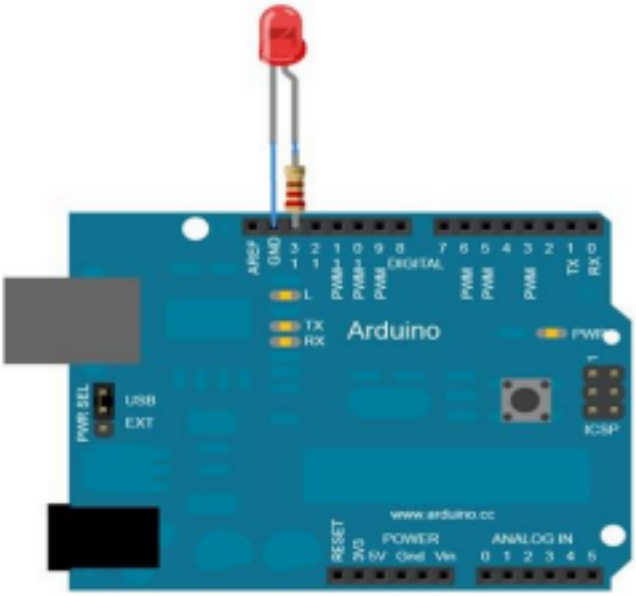
Hardware Required:

• Arduino Board

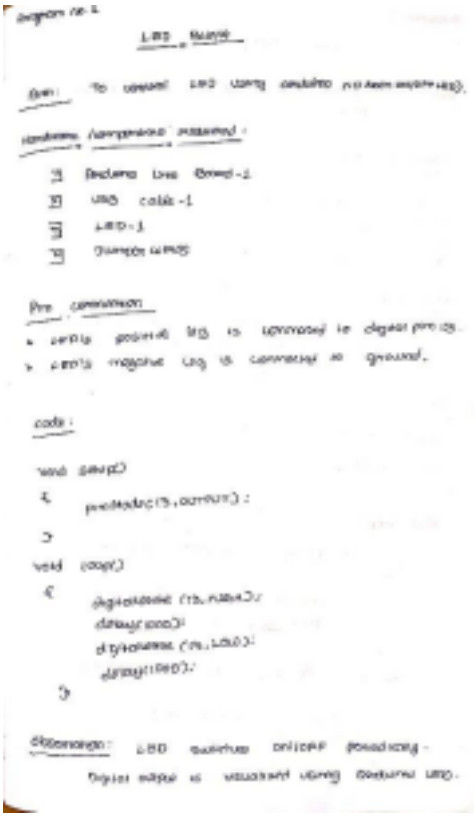
• LEDs

Pin connection:

- LED positive to pin 13.
- LED negative to ground.



Handwritten code:



Code:

```
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() {
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 LED blinks periodically.

2.LED ON/OFF Using Pushbutton

Aim:

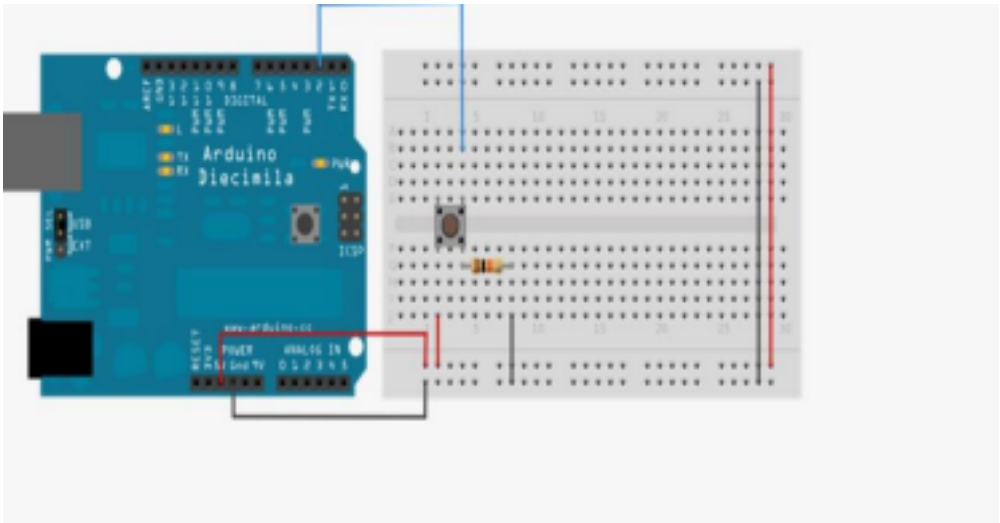
Turn an LED ON /OFF using a Pushbutton.

Hardware Required:

- Arduino Board
- LED
- Push button

Pin connection:

- LED positive to pin 13.
- LED negative to ground.
- Pushbutton leg to 5V.
- Pushbutton leg to ground.
- Pushbutton leg to pin 2.



Handwritten code:

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:

When the push button is pressed,the LED glows. When pushutton is released, LED is in OFF state.

3.LED Fading using Potentiometer

Aim:

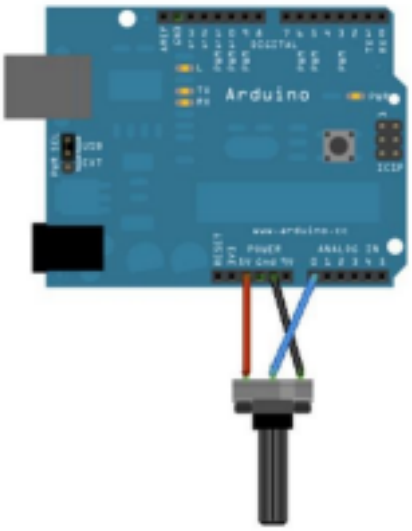
To control the brightness of an LED using a Potentiometer.

Hardware Required:

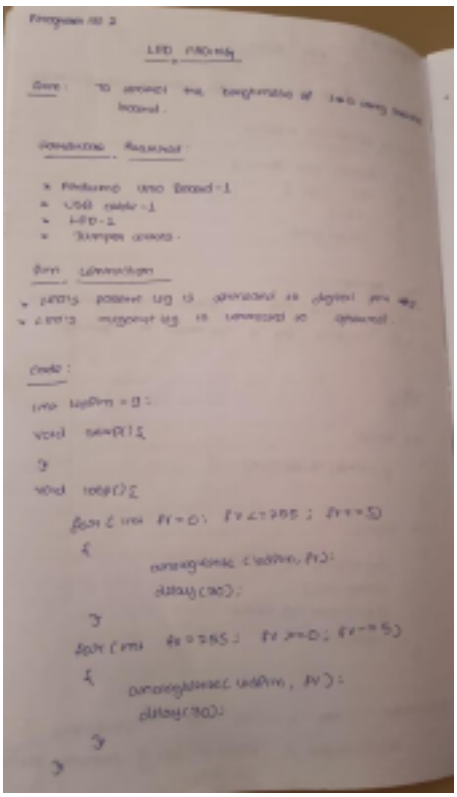
- Arduino Board
- LED
- Potentiometer

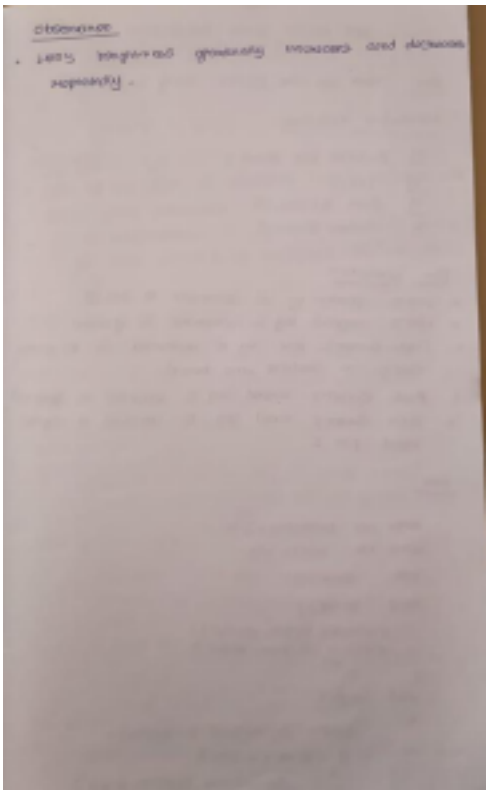
Pin connection:

- LED positive leg to digital pin 9.
- LED negative leg to ground.
- Potentiometer positive to 5V.
- Potentiometer ground to ground of arduino.
- Potentiometer to analog pin AO.



Handwritten code:





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 LED brightness is controlled by rotation of potentiometer.

4. Nightlight Simulation

Aim:

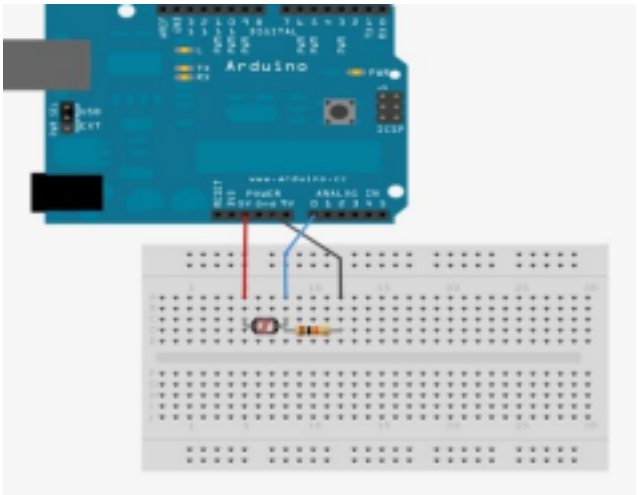
Simulating a night light using LDR and PIR

Hardware Required:

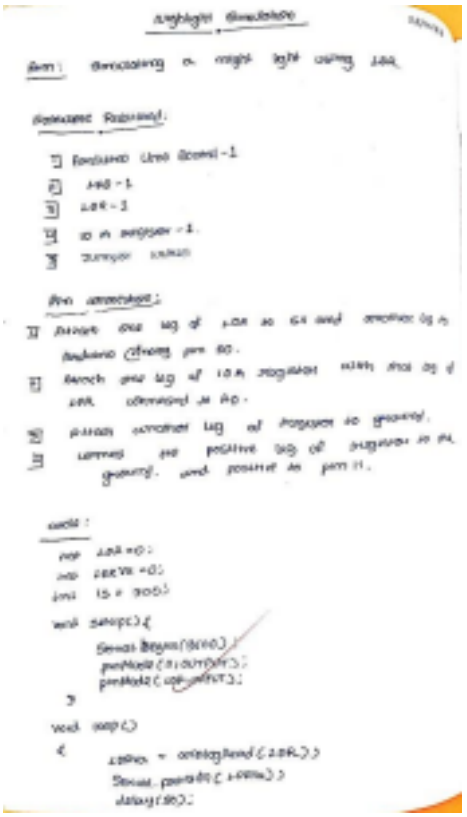
- 1 LED
- 1 LDR
- 110K register

Pin connection:

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



Handwritten code:



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 buad
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: Based on the readings from the LDR sensor, the LED light switches ON and OFF.

5.PIR with Arduino UNO

Aim:

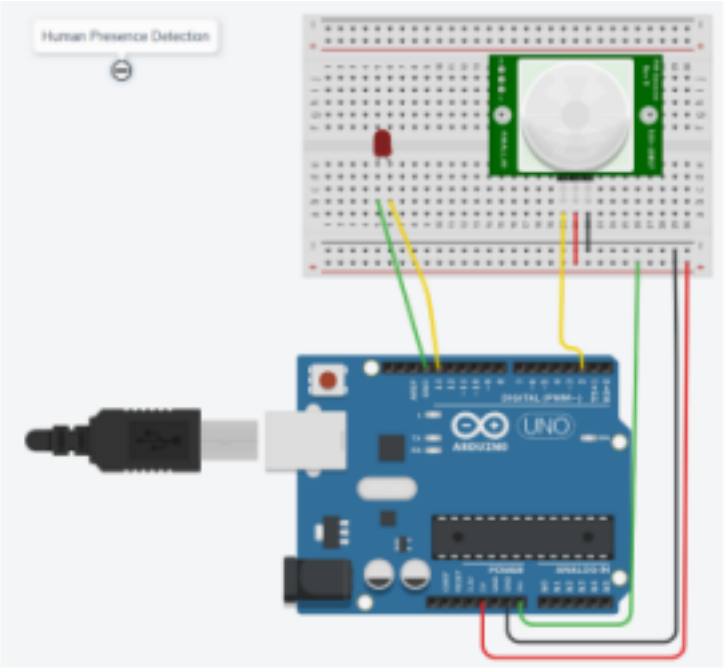
To detect motion using PIR sensor.

Hardware required:

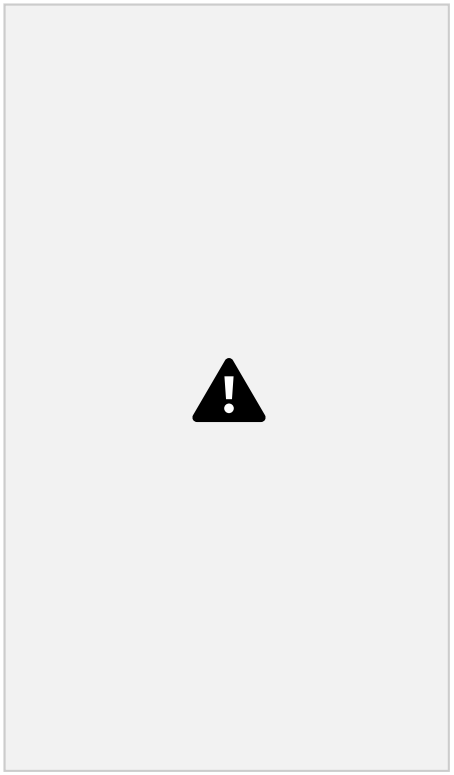
- Arduino UNO board
- PIR sensor
- LED

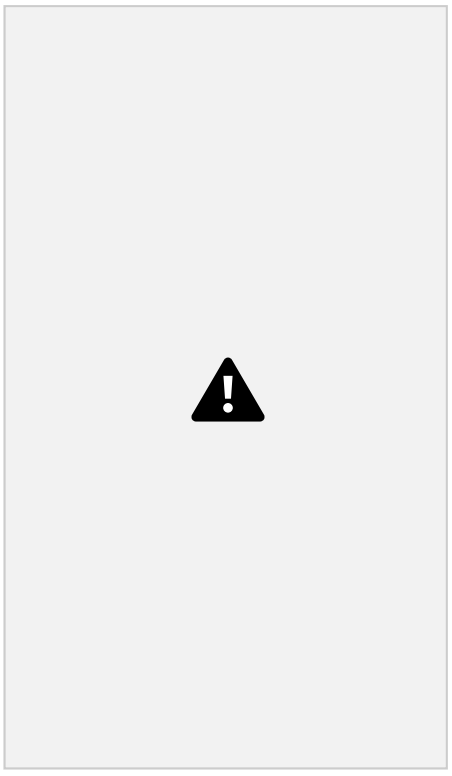
Pin connection:

- LED positive to pin 13.
- LED negative to ground.
- PIR negative to ground.
- PIR positive to 5V.
- PIR pin to A0.



Handwritten code:





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:

On detecting motion through PIR, the LED lights up.

6.Ultrasound with Arduino UNO

Aim:

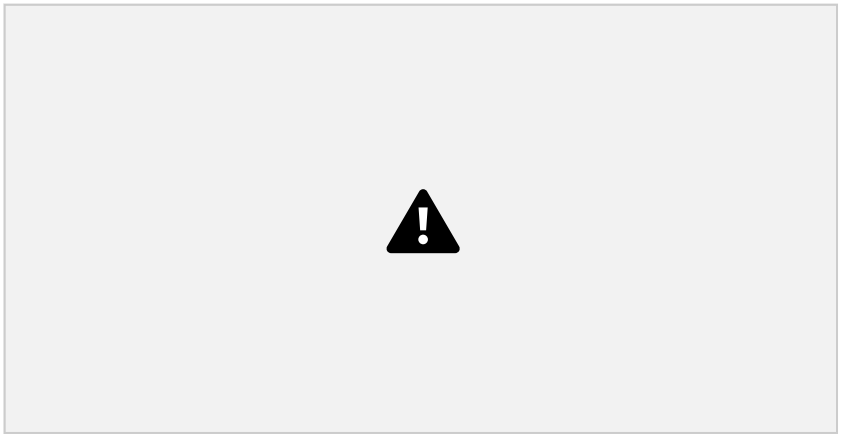
To detect proximity of objects using ultrasound.

Hardware required:

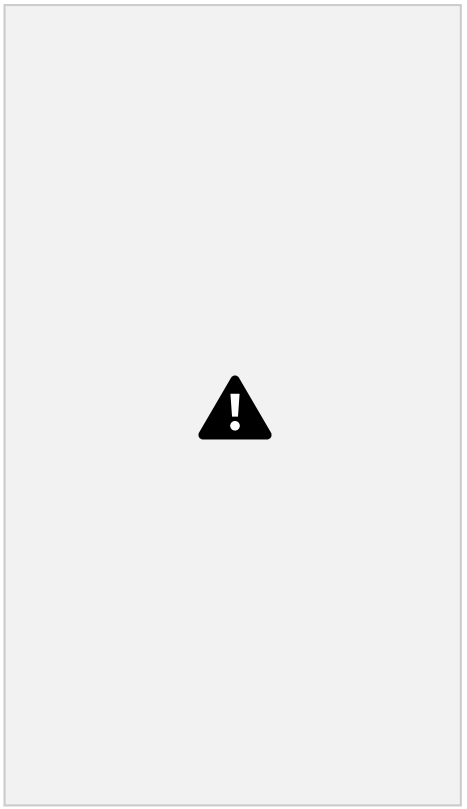
- Arduino UNO board
- Ultrasound

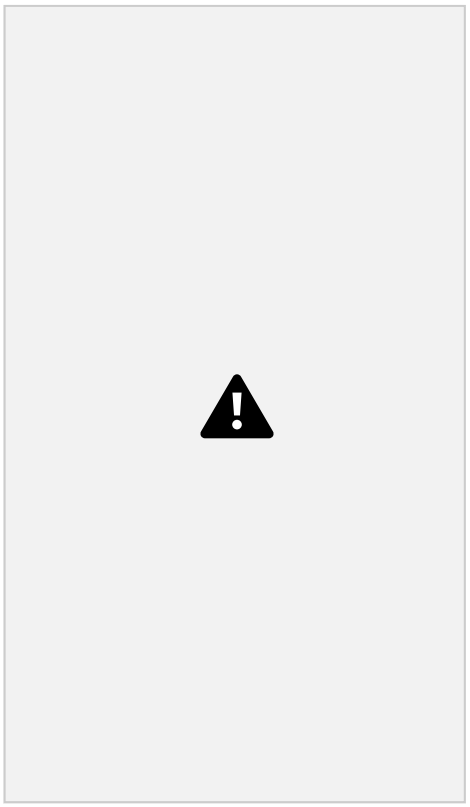
Pin connection:

- Ultrasound ground to ground.
- Ultrasound echo pin to pin 6.
- Ultrasound trigger pin to pin 7.
- Ultrasound Vcc to 5V.



Handwritten code:





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:

Distance between objects and ultrasound is printed on the monitor in centimeters and inches.

7. Fire Alert

Aim:

To simulate a fire alert system.

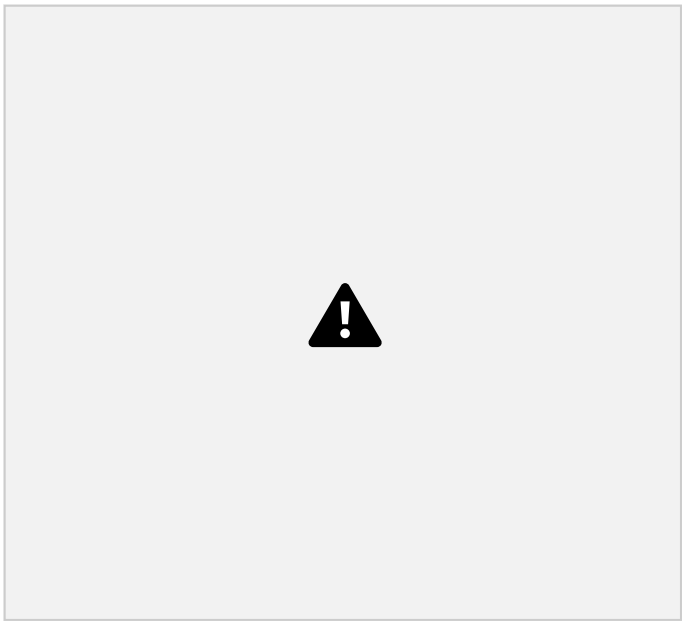
Hardware Required:

- Flame sensor (Analogue Output)
- Arduino
- LED
- Buzzer

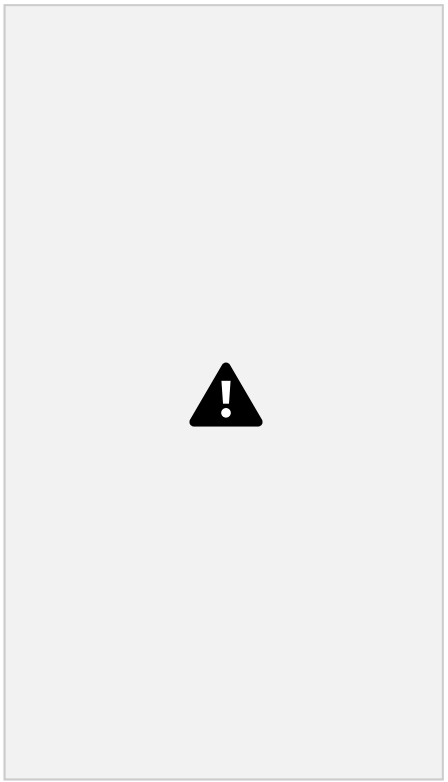
Pin connections:

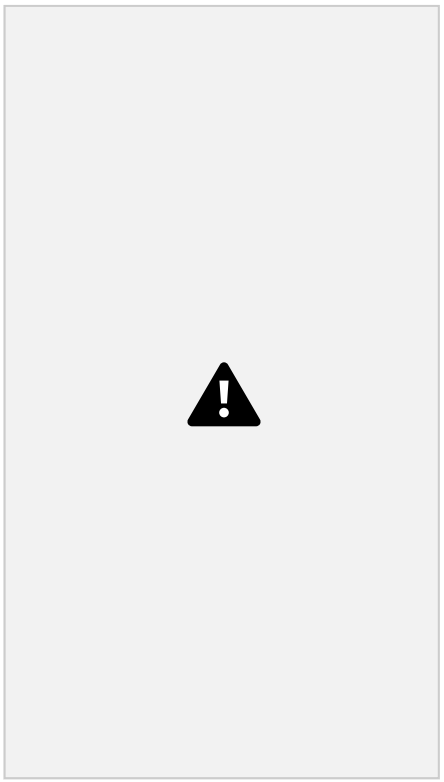
- Flame sensor Vcc to Arduino Vcc.
- Flame sensor ground to Arduino ground.
- Flame sensor A0 to Arduino A0.
- LED positive to pin 9.

- LED negative to ground.
- Buzzer positive to pin 12.
- Buzzer negative to ground.



Handwritten code:





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:

On detection of flame, the buzzer and the LED is switched on, issuing an alert.

8. Automatic irrigation controller simulation

Aim:

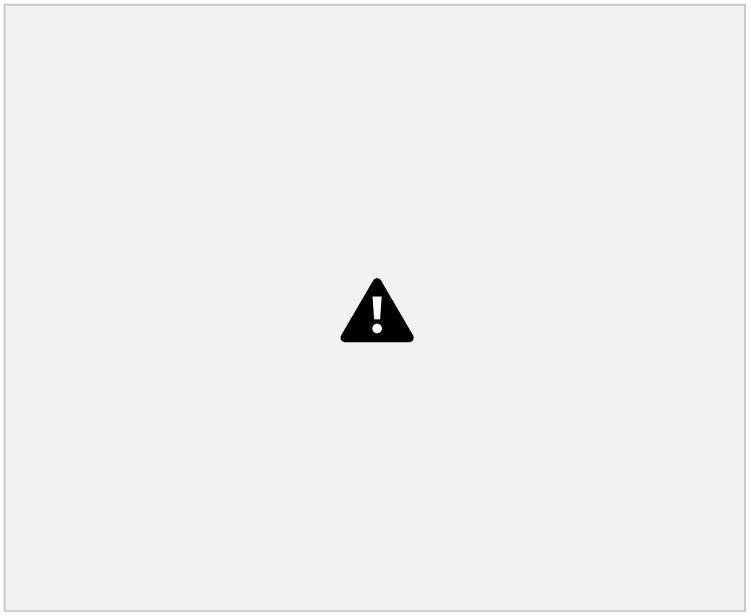
To sense the soil moisture and sprinkle water accordingly.

Hardware Required:

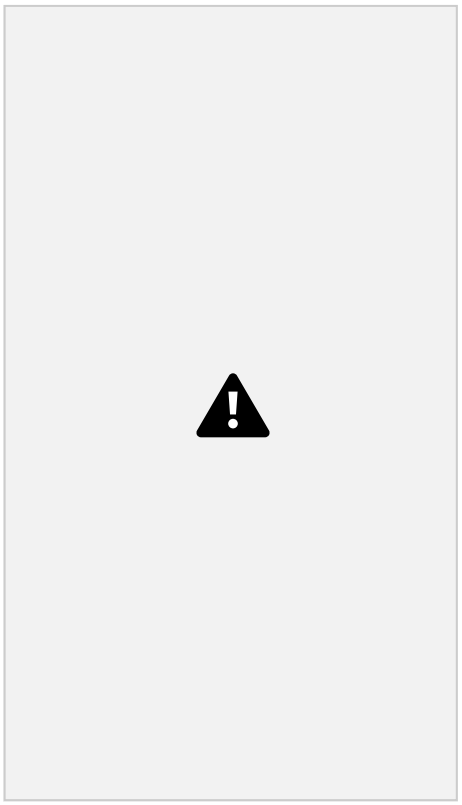
- Arduino
- Moisture Sensor
- Min servo motor

Pin 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:



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:

Based on the moisture sensor readings, the servo motor is switched on and off.

9. Reading the code present on RFID tag

Aim:

To read RFID tag number and print it onto the Serial monitor.

Hardware required:

- Arduino UNO board
- RFID tag
- RFID reader

Pin connection:

- RFID reader Vcc to 5V.
- RFID reader ground to ground.
- Tx pin of RFID reader to pin 9.

Code:

```
#include<SoftwareSerial.h>;

SoftwareSerial mySerial(9, 10);

int count = 0; // count = 0

char input[12]; // character array of size 12 rduino 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  }
}
```

10. Access control through RFID

Aim:

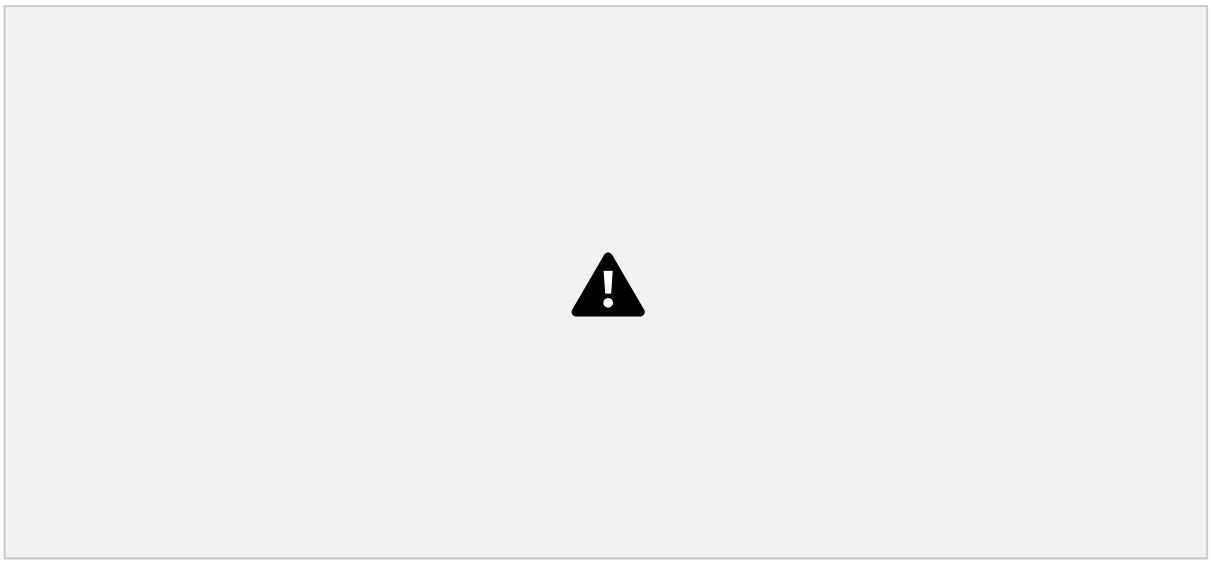
To authenticate access based on RFID tag number.

Hardware required:

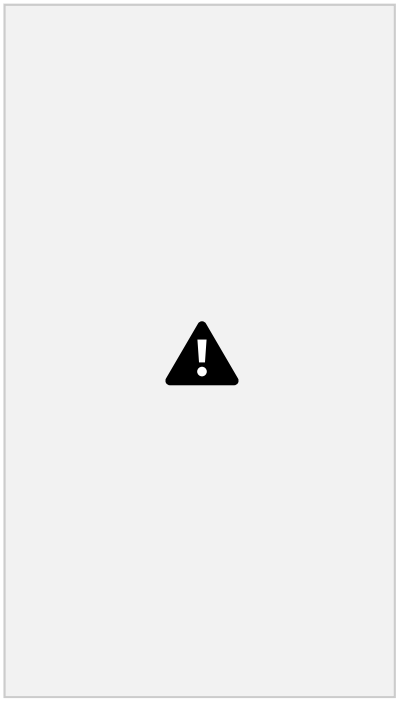
- Arduino UNO board
- RFID tag
- RFID reader

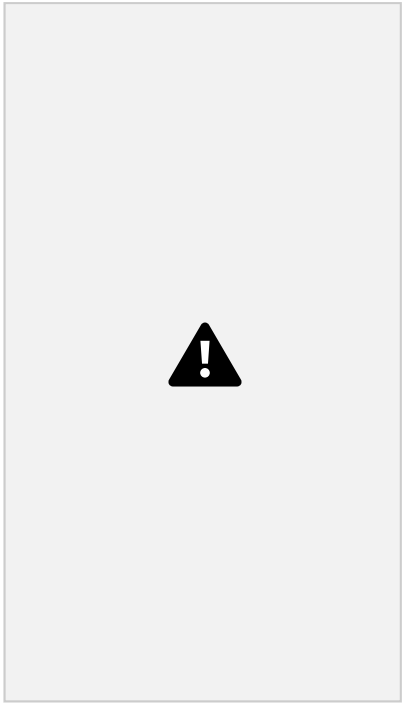
Pin connection:

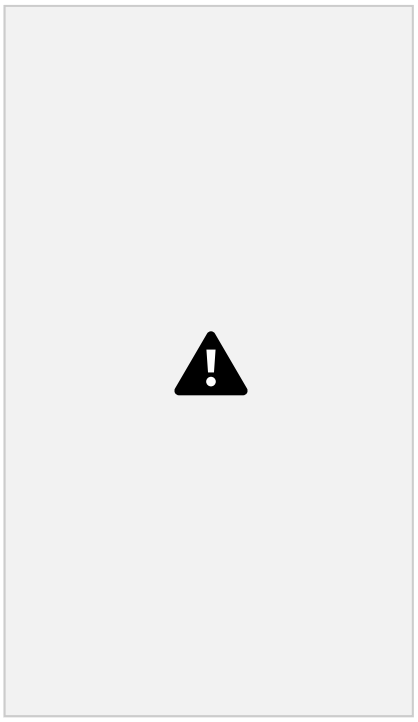
- RFID reader Vcc to 5V.
- RFID reader ground to ground.
- Tx pin of RFID reader to pin 9.
- LED positive to pin 12.
- LED negative to ground.



Handwritten code:







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
  rduino 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() &amp;amp;&amp;amp; count &amp;lt; 12)
{
input[count] = mySerial.read();
count++; // increment counter
delay(5);
}
if(count == 12)
{
count =0; // reset counter rduino 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&amp;lt;12; count++)
{
input[count]= &amp;#39;F&amp;#39;;
}
count = 0; // Reset counter variable
}
}
```

Observation:

Only registered RFID tag numbers are allowed and unregistered RFIDs are denied access.

11. HC-05 Bluetooth Module

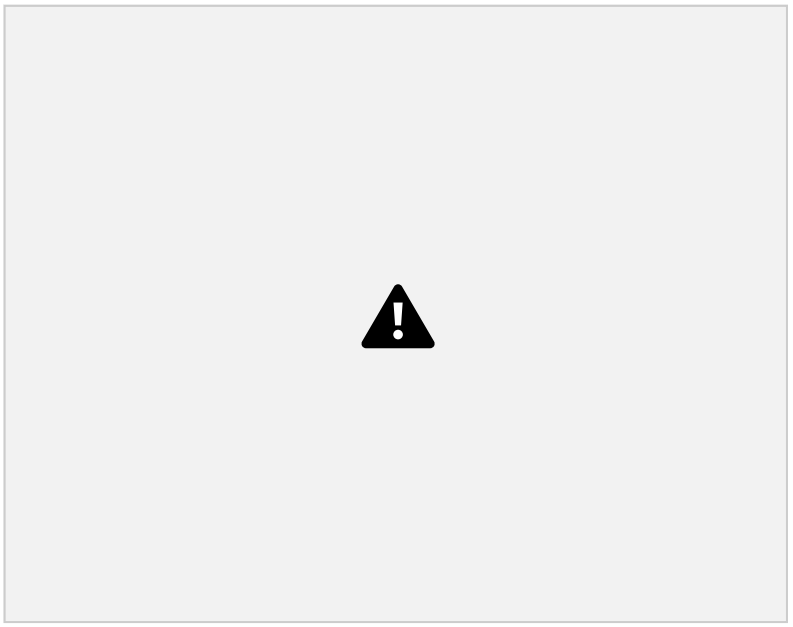
Aim:

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

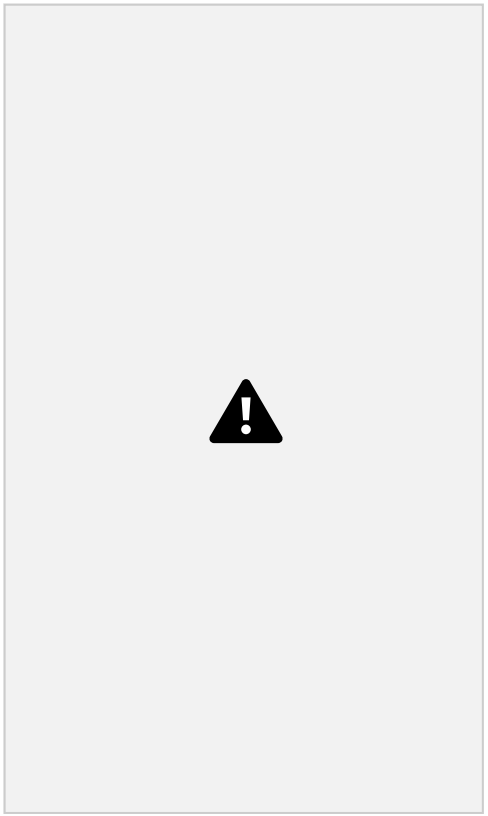
- HC-05 bluetooth module
- Arduino UNO board

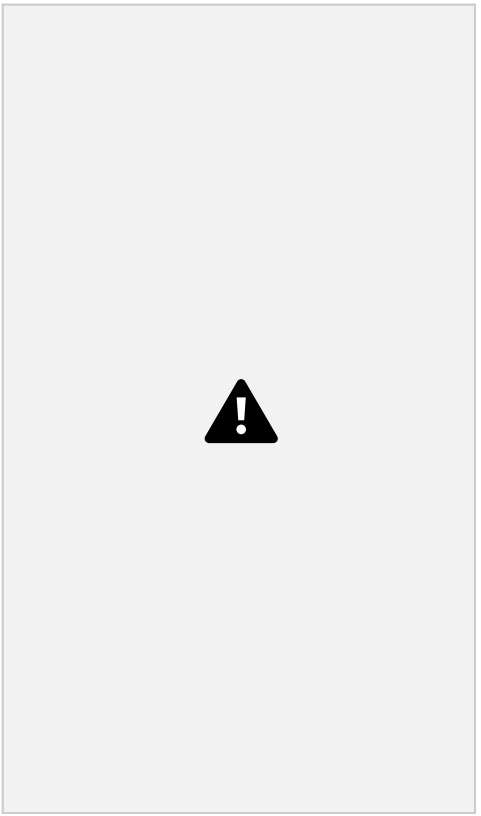
Pin connection:

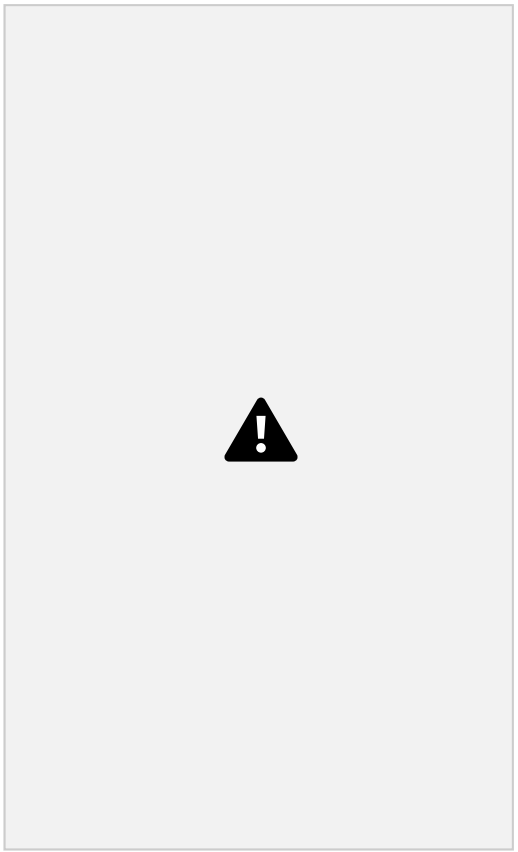
- Vcc to 5V of rduino.
- Bluetooth ground to ground of rduino.
- Tx rduino h to pin 10.
- Rx rduino h to pin 11.



Handwritten code:







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

```
}
```

HC-05 Controlled by mobile

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 I 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;
 }
}
```

BT-Master Slave

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 more
}
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:

● Commands can be sent to rduino h module to configure them. ● LED state can be controlled by rduino h module. ● Bluetooth master/slave configuration is simulated.

14. GSM Module

1. GSM Module: Call to a particular number

Aim:

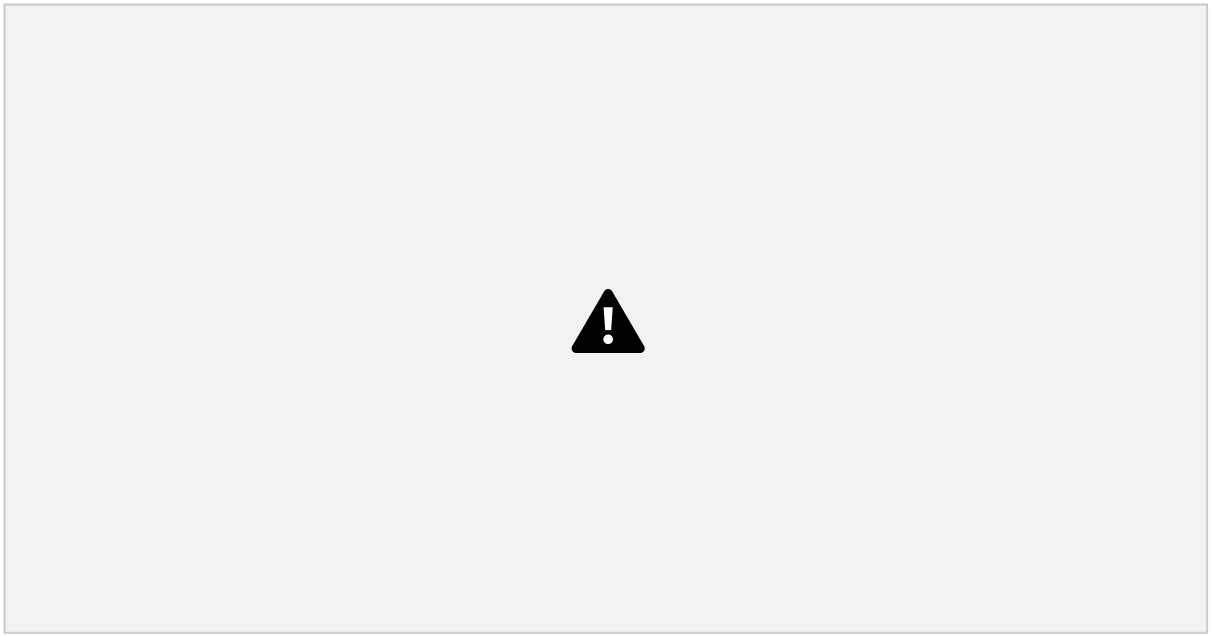
Call using Arduino and GSM Module – to a specified mobile number inside the program.

Hardware required:

- Arduino UNO board
- GSM module
- SIM card

Pin connections:

- GSM Tx to rduino pin 2.
- GSM Rx to rduino pin 3.
- GSM ground to ground of rduino.



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() {

}
```

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".

Pin connection:

- Flame sensor Vcc to Arduino Vcc.
- Flame sensor ground to Arduino ground.
- Flame sensor A0 to Arduino A0.

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

}

}
```

2. **Sending and Receiving Message**

Aim:

2) 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.

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 withyour 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);
}
```

4. Controlling LED through received messages:

Aim:

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

Pin connection:

- Attach LED to pin 13 and GND.

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...");
}
```

}

}

Handwritten code:

Observation:

● A call to a specified number is placed using the GSM module. ● SMS messages are sent and received through the GSM module. ● An LED can be controlled using a GSM module.