VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

INTERNET OF THINGS LAB

Submitted by

Prakhyati Bansal (1BM21CS136)

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
OCT-2023 to FEB-2024

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

CERTIFICATE

This is to certify that the Lab work entitled "Internet Of Things Lab" carried out by **Prakhyati Bansal (1BM21CS136),** 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.

Ms. Sowmya T
Assistant professor
Department of CSE
BMSCE, Bengaluru

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

`

INDEX

1. LED Blinking

Aim:

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

Hardware Required:

• Arduino Board

• LED

Circuit diagram:

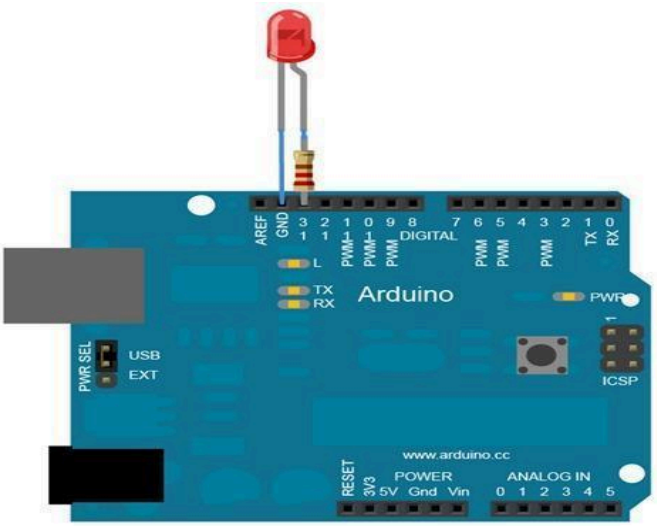
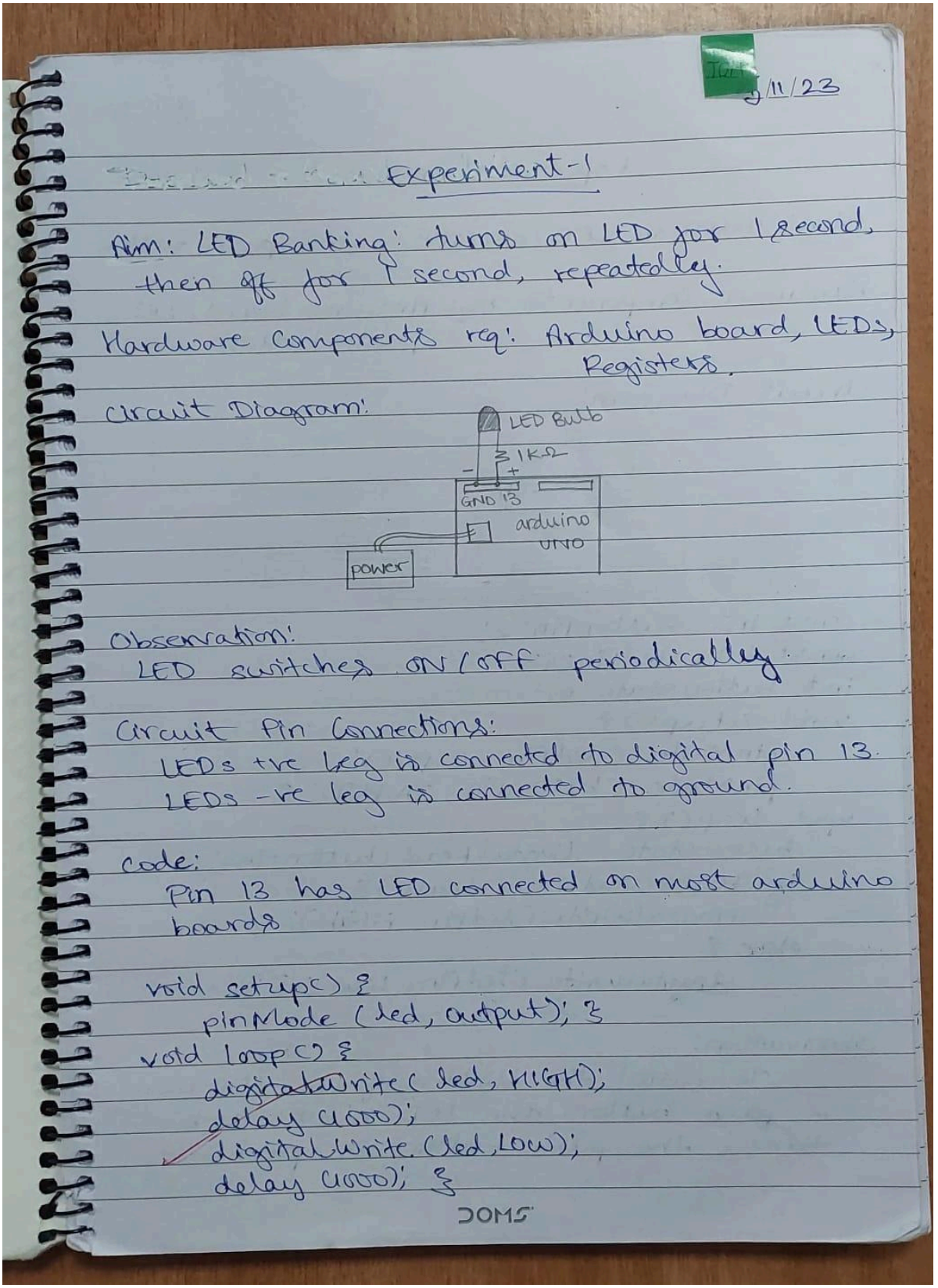


Fig.1.LED blinking

Handwritten code pic:



9/11/23

Experiment-1

Aim: LED Banking: turns on LED for 1 second, then off for 1 second, repeatedly.

Hardware Components req: Arduino board, LEDs, Registers.

Circuit Diagram:

Observation:
LED switches ON / OFF periodically.

Circuit Pin Connections:
LEDs +ve leg is connected to digital pin 13.
LEDs -ve leg is connected to ground.

Code:
Pin 13 has LED connected on most arduino boards

```
void setup() {
  pinMode (led, output); }
void loop() {
  digitalWrite( led, HIGH);
  delay (1000);
  digitalWrite (led, LOW);
  delay (1000); }
```

DOMS

Code:

```
int led = 13;
void setup()
{
pinMode(led, OUTPUT);
}
void loop() {
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
```

Observation:

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

2. LED ON/OFF Using Pushbutton

Aim:

Turn an LED ON /OFF using a Pushbutton.

Hardware Required:

• Arduino Board

• LED

• Push button

Circuit diagram:

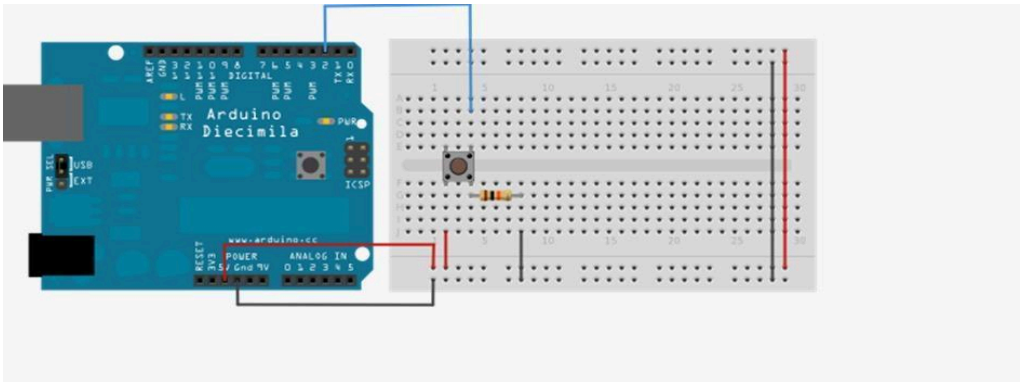
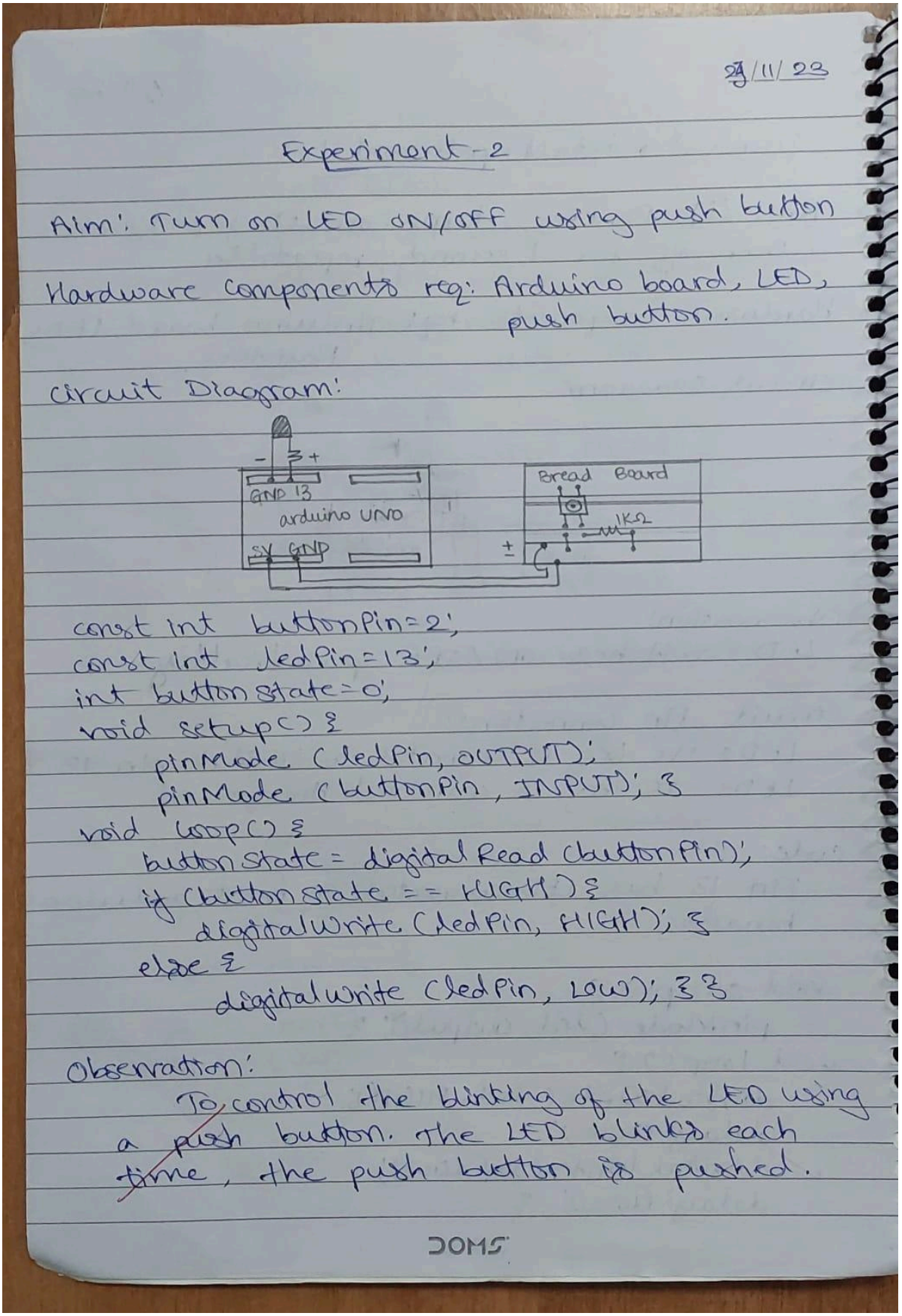


Fig.2.LED on/off using pushbutton

Handwritten code pic:



Code:

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

Observation:

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

3. LED Fading using Potentiometer

Aim:

To control the brightness of an LED using a Potentiometer.

Hardware Required:

• Arduino Board

• LED

• Potentiometer

Circuit diagram:

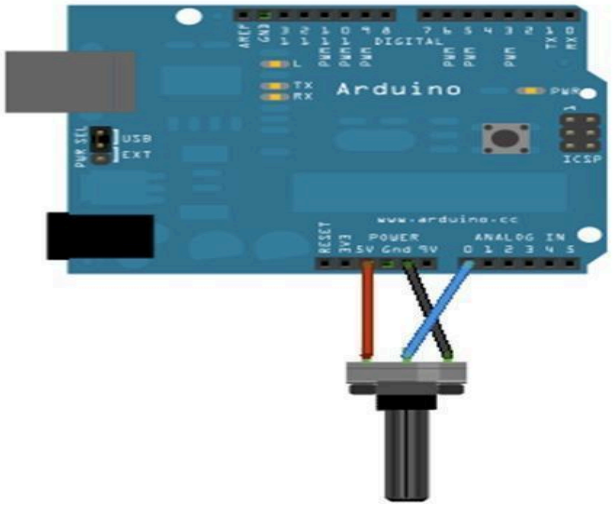


Fig.3-LED fading using potentiometer

Handwritten code pic:

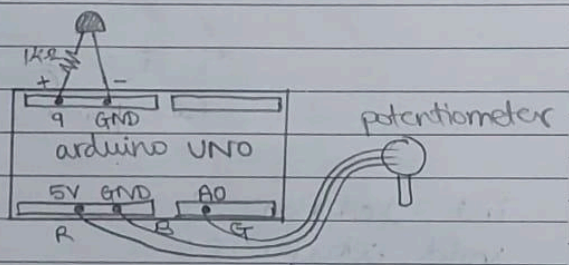
25/11/22

Experiment-3

Aim: To control brightness of LED using a potentiometer.

Hardware components req: Arduino board, LED, potentiometer.

Circuit Diagram:




```
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:

While varying the resistance using the shaft of potentiometer the LED starts to fade in brightness.

DOMS

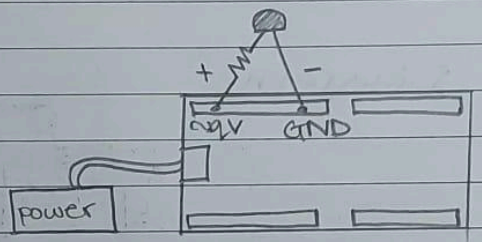
23/11/23

Experiment-4

Aim: LED fading

Hardware Components req.: Arduino Board, LED

Circuit Diagram:




```
int ledPin = 9;
void setup() { }
void loop() {
  for (int fadeValue = 0; fadeValue <= 255;
                                fadeValue++)
  {analogWrite (ledPin, fadeValue);
   delay(30); }
  for (int fadeValue = 255; fadeValue >= 0; fadeValue--)
  { analogWrite (ledPin, fadeValue);
    delay (30); } }
```

Observation:

fading of LED using the condition of the 'for' loops at a delay of 30 ms.

Sri
23/11/23

Code:

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

Observation:

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

4. Nightlight Simulation

Aim:

Simulating a night light using LDR

Hardware Required:

• 1 LED

• 1 LDR

• 110K resistor

Connection:

1. Attach one leg of LDR to 5V and another leg to Arduino Analog pin A0

2. Attach one leg of 110K resistor with that leg of LDR connected to A0

3. Attach another leg of resistor to the ground

4. Connect the positive leg of LED to pin 11 and negative to GND

Circuit diagram:

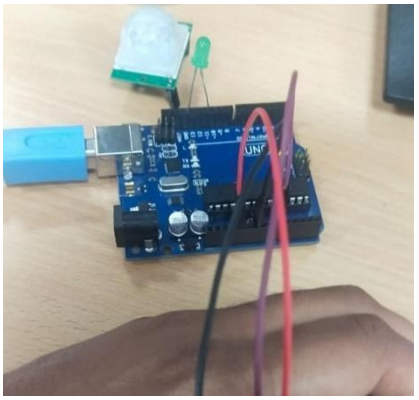


Fig 4.1- when it is bright, the LED is off.

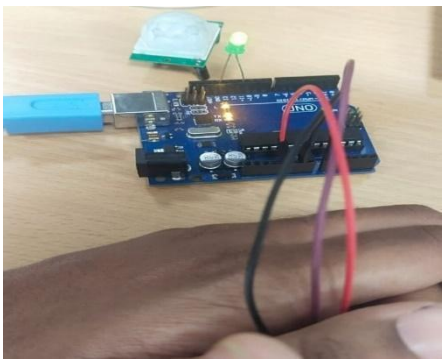
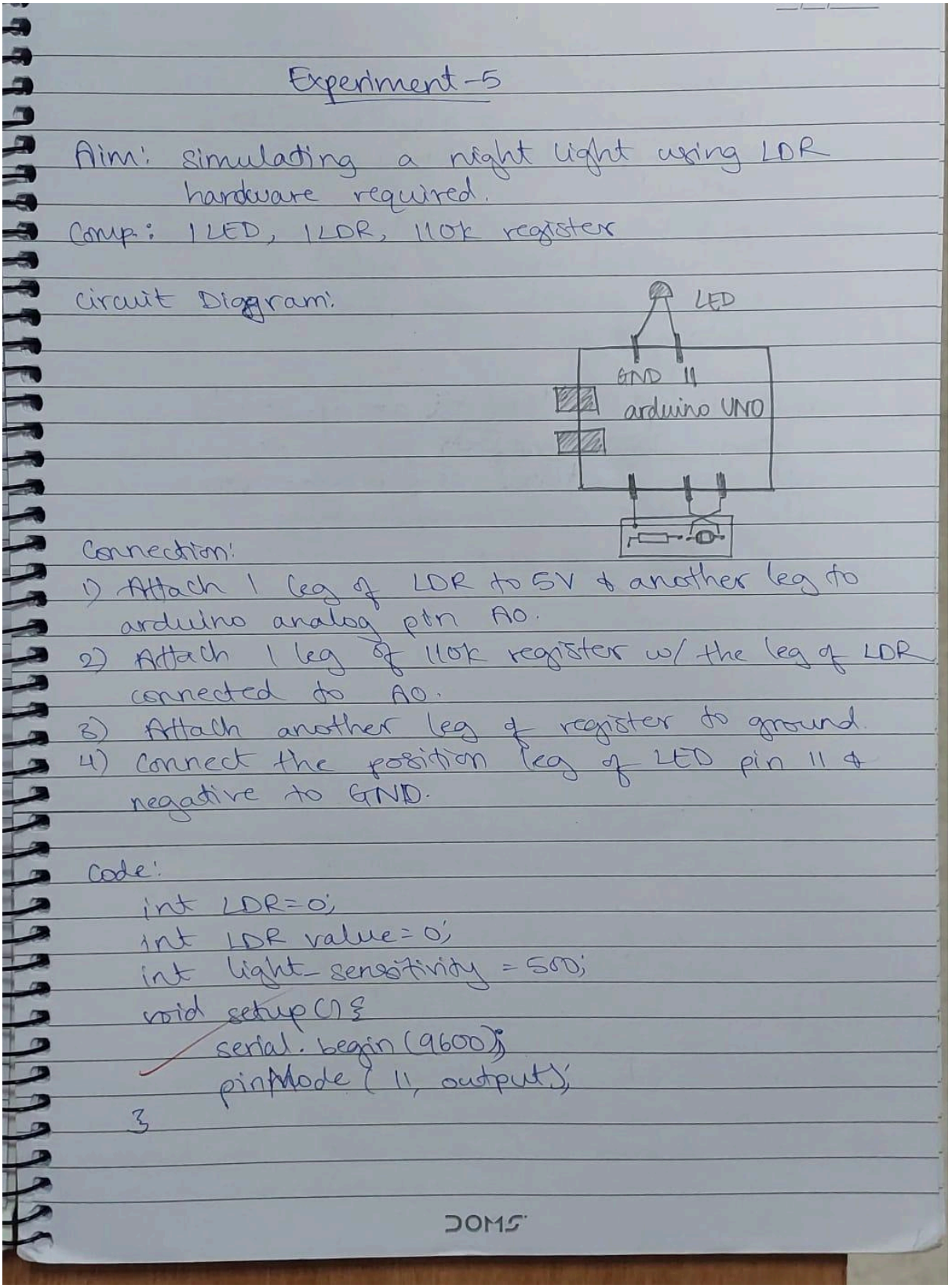


Fig 4.2- when dark, the LED turns on.

Handwritten code pic:



Experiment-5

Aim: simulating a night light using LDR hardware required.

Comp: 1 LED, 1 LDR, 1 10k register

Circuit Diagram:

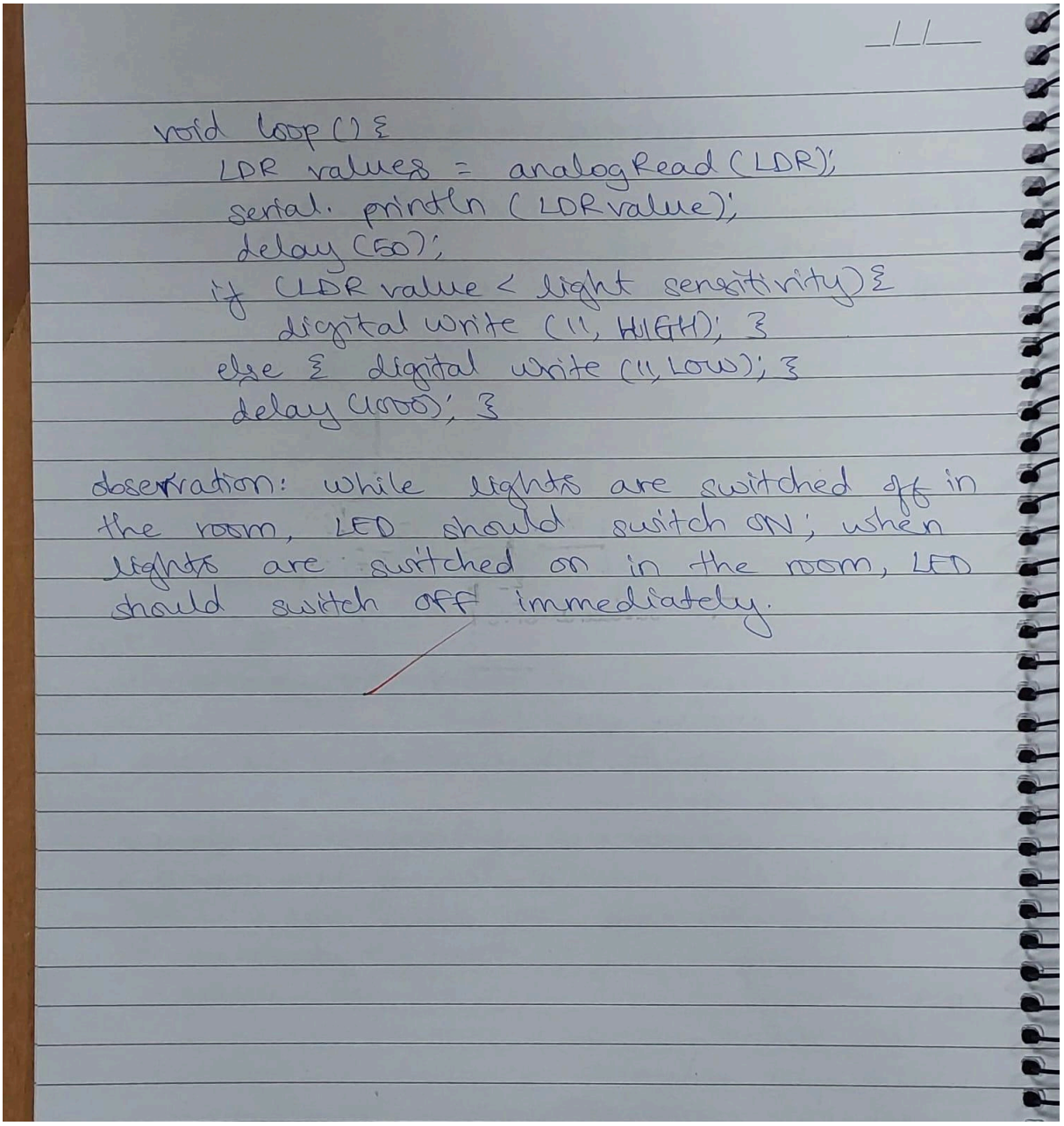
Connection:

1) Attach 1 leg of LDR to 5V & another leg to arduino analog pin A0.
2) Attach 1 leg of 10k register w/ the leg of LDR connected to A0.
3) Attach another leg of register to ground.
4) Connect the position leg of LED pin 11 & negative to GND.

Code:

```
int LDR=0;
int LDR value = 0;
int light_sensitivity = 500;
void setup() {
    serial.begin(9600);
    pinMode(11, output);
}
```

DOMS



Code:

```
int LDR = 0;
int LDRValue = 0;
int light_sensitivity = 500;
void setup()
{
Serial.begin(9600);
pinMode(11, OUTPUT);
```

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

Observation:

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

5. PIR with Arduino UNO

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

Hardware Required:

• 1 LED

• 1 PIR

• Arduino UNO

Circuit diagram:

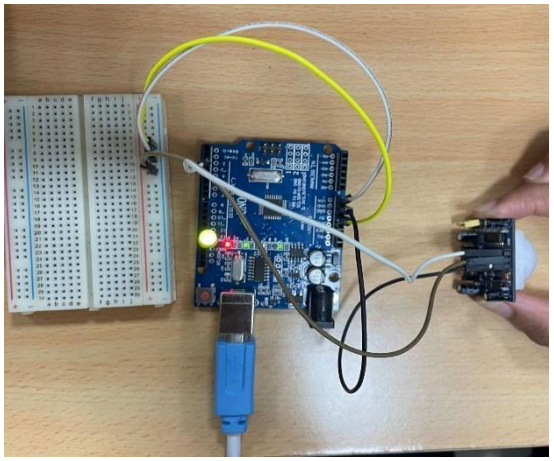
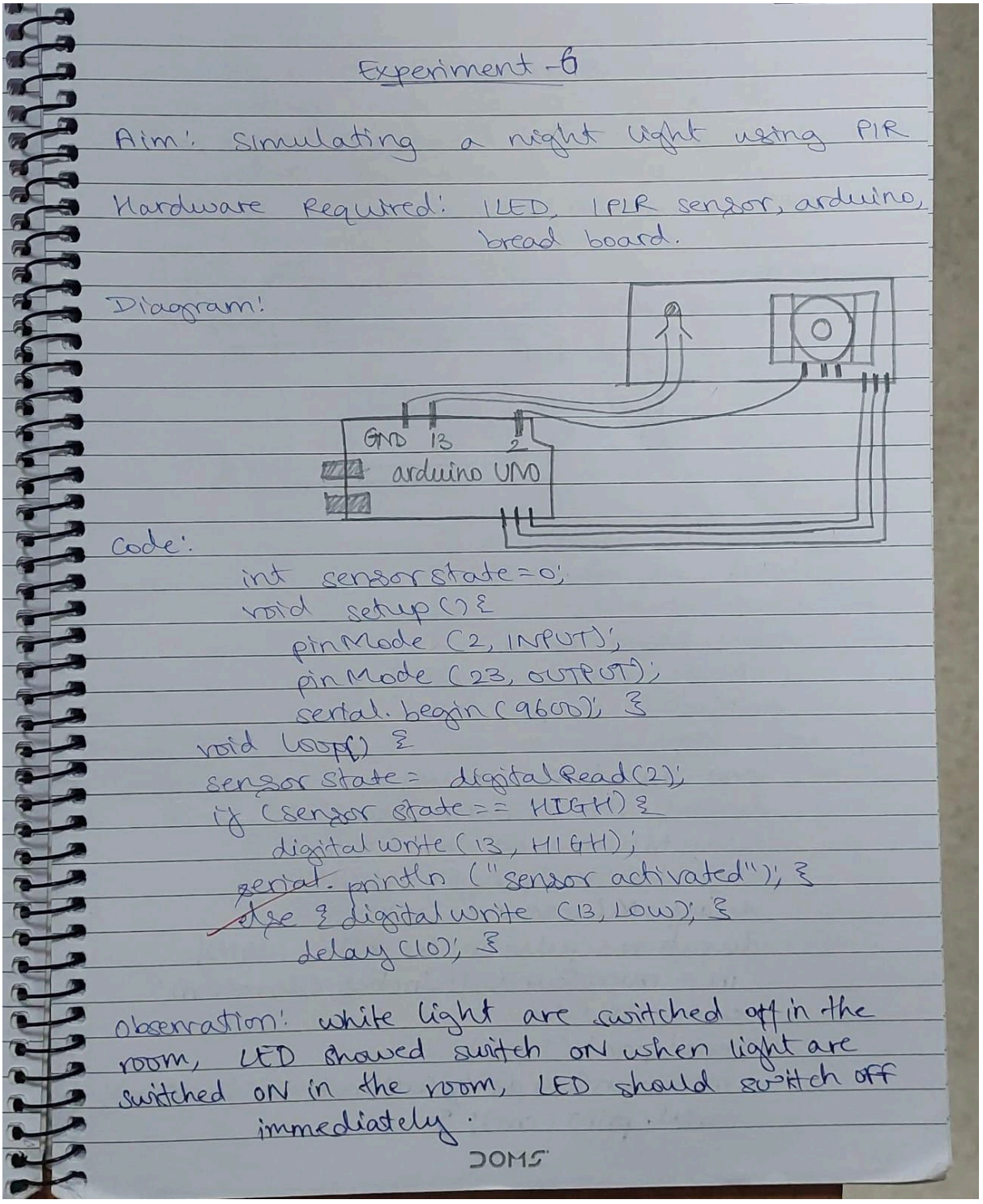


Fig 5- When motion is detected LED is high

Handwritten code pic:



Experiment - 6

Aim: Simulating a night light using PIR

Hardware Required: 1 LED, 1 PIR sensor, arduino, bread board.

Diagram:

Code:

```
int sensorState = 0;
void setup() {
  pinMode (2, INPUT);
  pinMode (23, OUTPUT);
  serial.begin (9600); }
void loop() {
sensorState = digitalRead(2);
if (sensorState == HIGH) {
   digitalWrite (13, HIGH);
   serial.println ("sensor activated"); }
   else { digitalWrite (13, LOW); }
   delay (10); }
```

Observation: white light are switched off in the room, LED showed switch on when light are switched on in the room, LED should switch off immediately.

DOMS

Code:

```
int sensorState = 0;
void setup()
{
pinMode(2, INPUT);
pinMode(13, OUTPUT);
Serial.begin(9600); }
void loop()
{
sensorState = digitalRead(2);
if (sensorState == HIGH) {
digitalWrite(13, HIGH);
Serial.println("Sensor activated!");
} else {
digitalWrite(13, LOW);
}
delay(10); }
```

Observation:

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

6. Ultrasound with Arduino UNO

Aim: To detect the distance of an object.

Hardware Required:

• Ultrasonic sensor

• jumper wires(female to male)

• Arduino UNO

Circuit diagram:

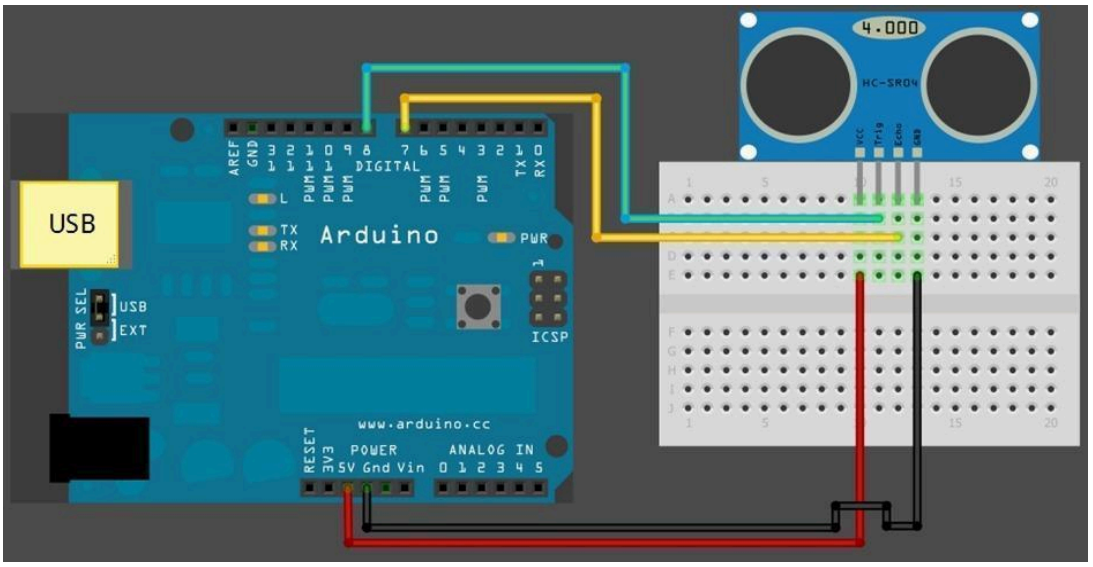
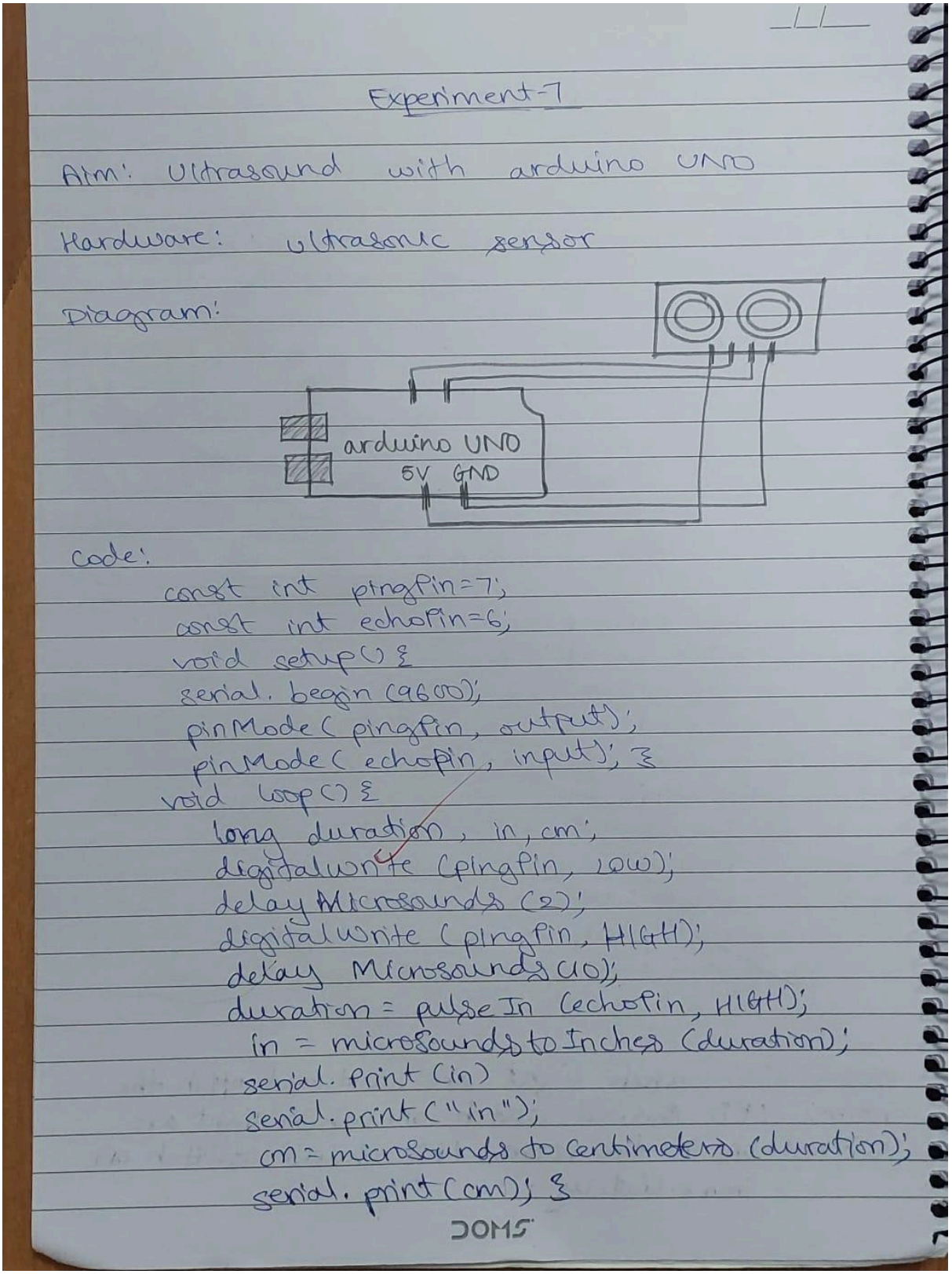


Fig 6-measures the distance of nearest object.

Handwritten code pic:



Experiment-7

Aim: Ultrasound with arduino UNO

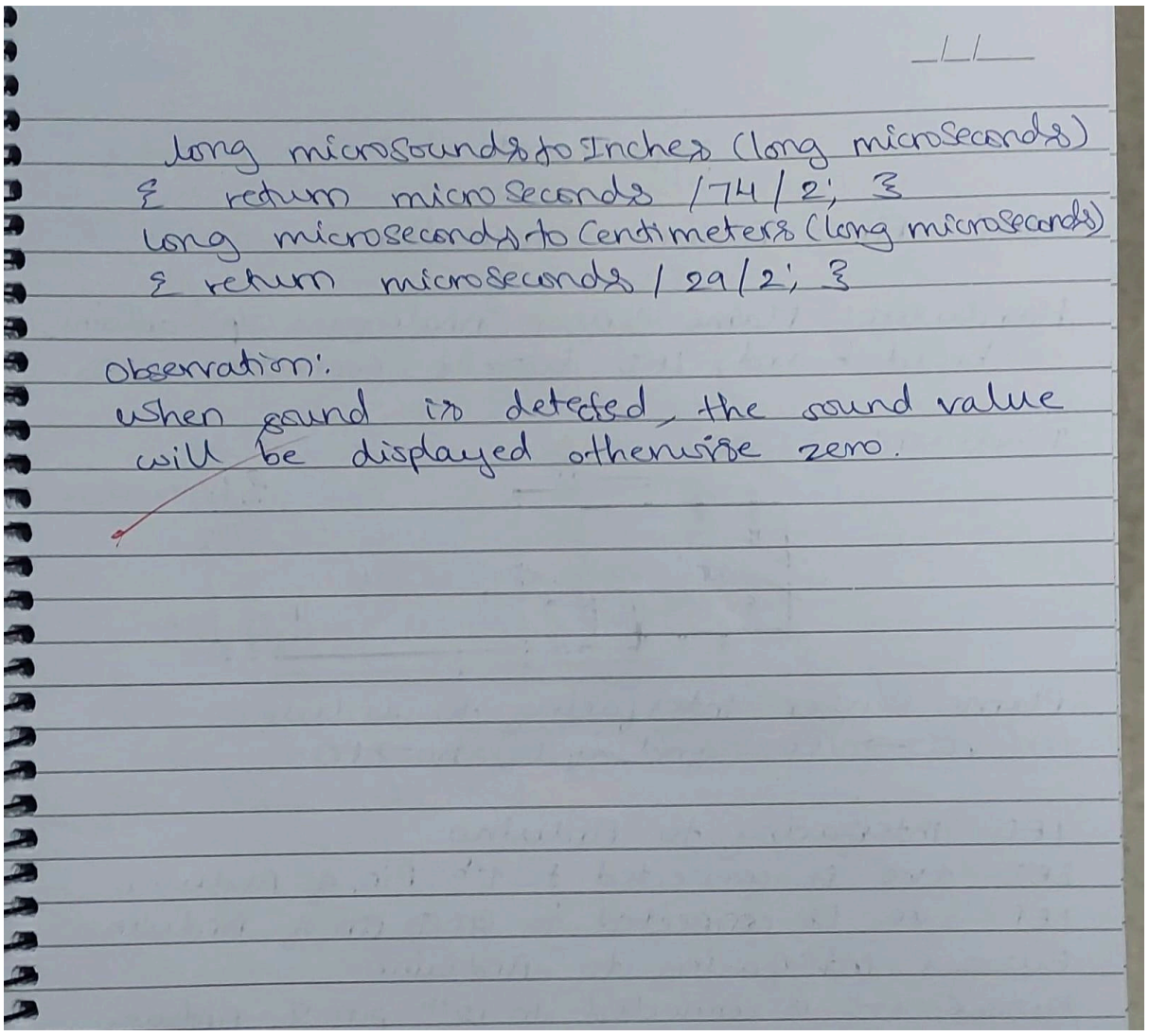
Hardware: Ultrasonic sensor

Diagram:

Code:

```
const int pingPin=7;
const int echoPin=6;
void setup() {
serial.begin(9600);
pinMode(pingPin, output);
pinMode(echoPin, input); }
void loop() {
long duration, in, cm;
digitalWrite(pingPin, LOW);
delayMicrosounds(2);
digitalWrite(pingPin, HIGH);
delay Microsounds(10);
duration = pulseIn(echoPin, HIGH);
in = microSounds to Inches(duration);
serial.Print(in)
serial.print("in");
cm = microsounds to centimeters(duration);
serial.print(cm); }
```

DOMS



Code:

```
const int pingPin = 7;
const int echoPin=6;
void setup()
{
Serial.begin(9600);
pinMode(pingPin, OUTPUT);
pinMode(echoPin, INPUT);
}
void loop()
{
```

```
long duration, inches, cm;
digitalWrite(pingPin, LOW);
delayMicroseconds(2);
digitalWrite(pingPin, HIGH);
delayMicroseconds(10);
digitalWrite(pingPin, LOW);
duration = pulseIn(echoPin, HIGH);
inches = microsecondsToInches(duration);
Serial.print(inches);
Serial.print("inches");
cm = microsecondsToCentimeters(duration);
Serial.print(cm);
Serial.println("cm");
}
long microsecondsToInches(long microseconds)
{
return microseconds / 74 / 2; }
long microsecondsToCentimeters(long microseconds)
{ return microseconds / 29 / 2;
 }
```

Observation:

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

7. Fire Alert

Aim:

Fire alarm simulation.

Hardware Required:

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

Connections:

Flame sensor interfacing to Arduino

Flame sensor to Arduino

vcc to vcc

gnd to gnd

A0 to A0

Led interfacing to Arduino

LED +ve is connected to 9th pin of Arduino

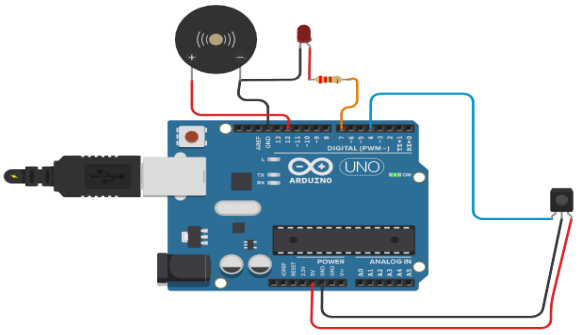
LED -ve is connected to gnd pin of arduino

Buzzer interfacing to Arduino

Buzzer +ve is connected to 12th pin of Arduino

Buzzer -ve is connected to GND pin of Arduino

Circuit diagram:



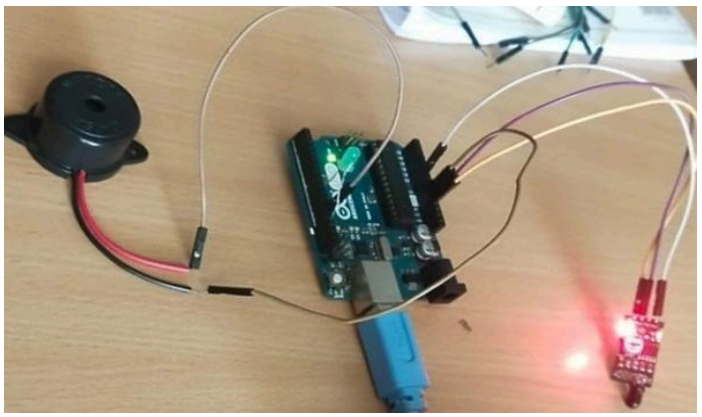
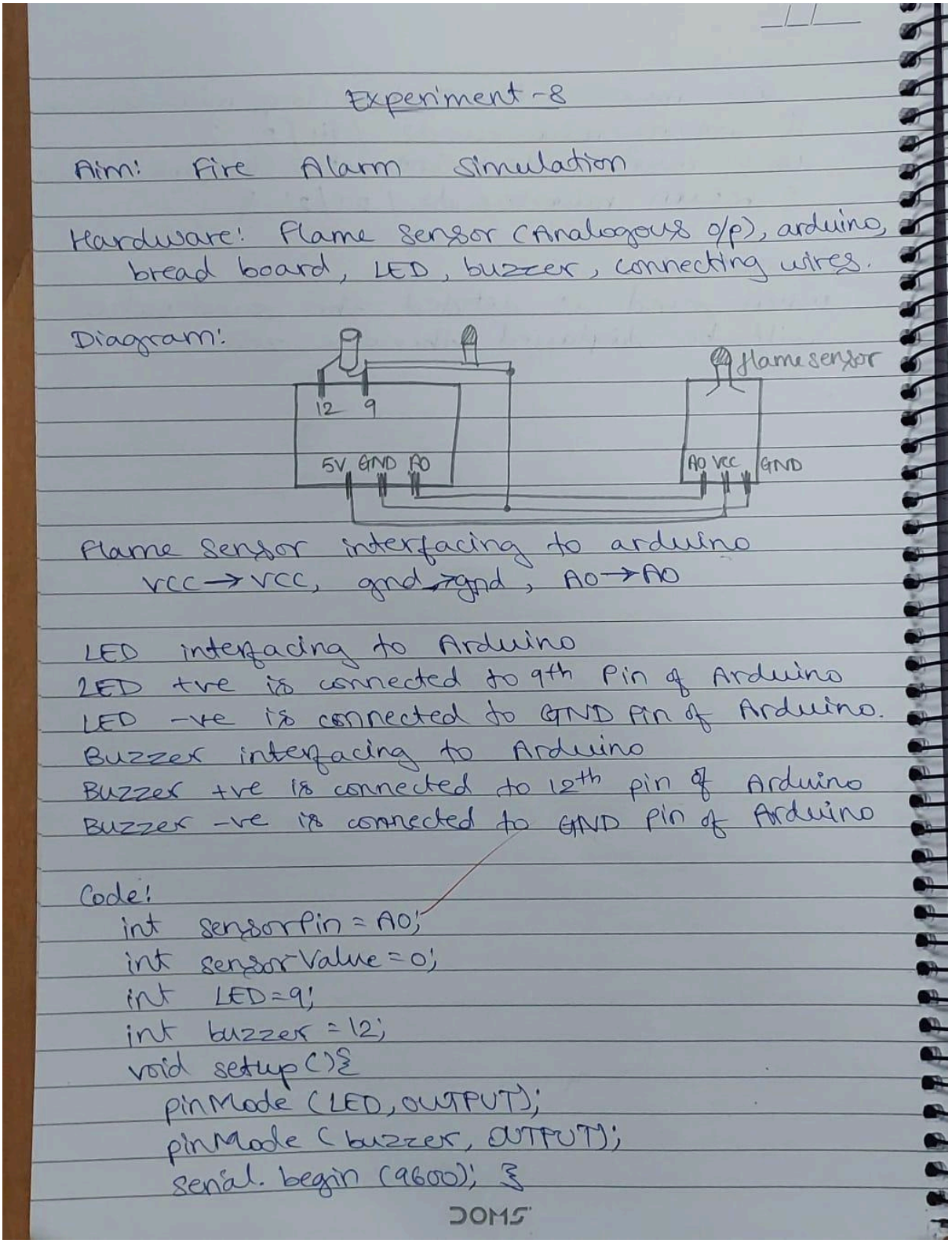


Fig 7- When the fire is detected LED turns on.

Handwritten code pic:



Experiment - 8

Aim: Fire Alarm Simulation

Hardware: Flame sensor (Analogous o/p), arduino, bread board, LED, buzzer, connecting wires.

Diagram:

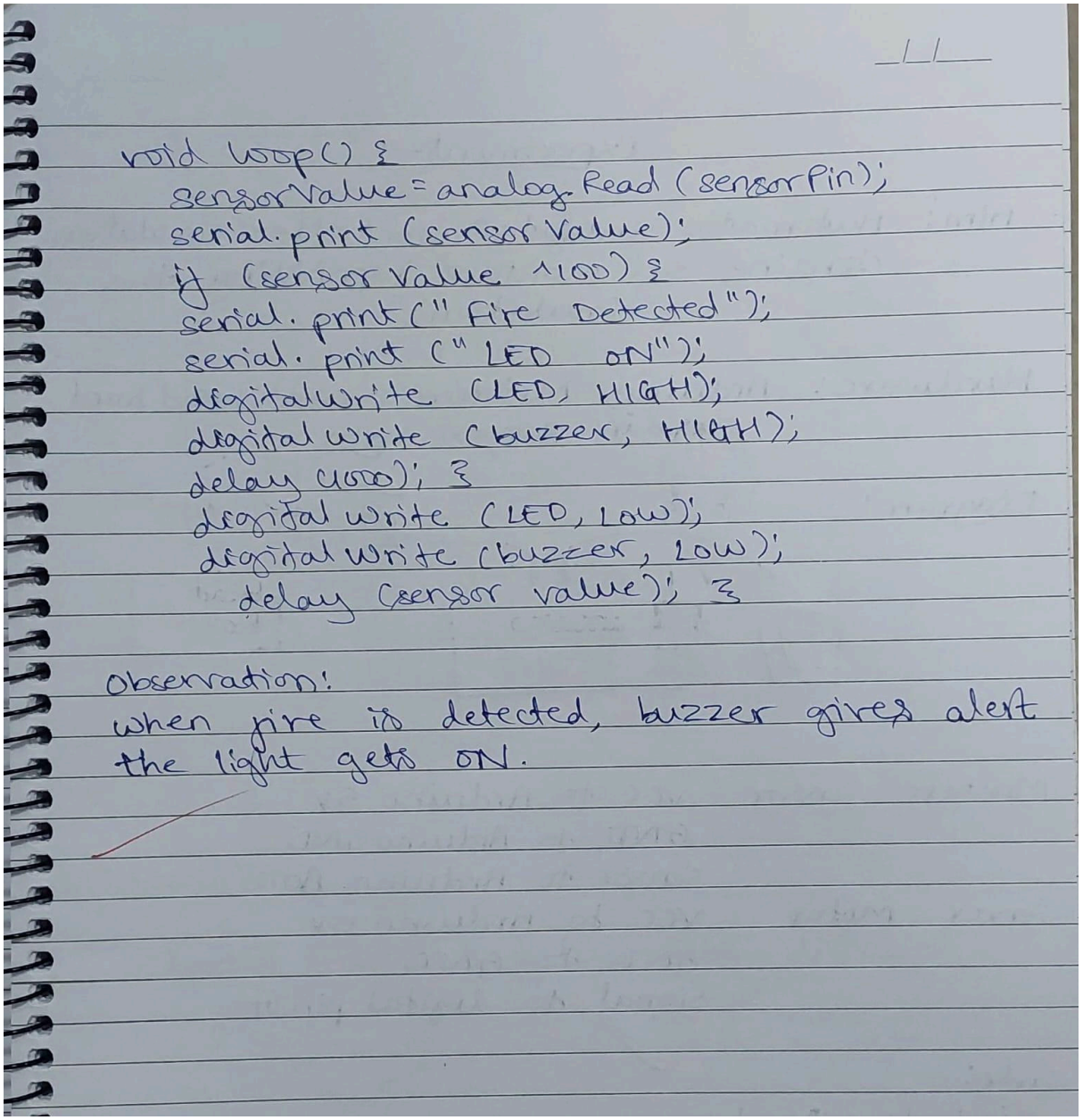
Flame Sensor interfacing to arduino
vcc → vcc, gnd → gnd, A0 → A0

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

Code:

```
int sensorPin = A0;
int sensorValue = 0;
int LED = 9;
int buzzer = 12;
void setup(){
  pinMode (LED, OUTPUT);
  pinMode (buzzer, OUTPUT);
  Serial. begin (9600); }
```

DOMS



Code:

```
int sensorPin = A0; // select the input pin for the LDR
int sensorValue = 0; // variable to store the value coming from the sensor
int led = 9; // Output pin for LED
int buzzer = 12; // Output pin for Buzzer
void setup() {
pinMode(led, OUTPUT);
pinMode(buzzer,OUTPUT);
```

```
Serial.begin(9600);
}
void loop()
{
sensorValue = analogRead(sensorPin);
Serial.println(sensorValue);
if (sensorValue < 100)
{
Serial.println("Fire Detected");
Serial.println("LED on");
digitalWrite(led,HIGH);
digitalWrite(buzzer,HIGH);
delay(1000);
}
digitalWrite(led,LOW);
digitalWrite(buzzer,LOW);
delay(sensorValue);
}
```

Observation:

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

8. Automatic irrigation controller simulation

Aim:

Sensing the soil moisture and sprinkling the Water simulation.

Hardware Required:

- Arduino
- Moisture Sensor
- Breadboard
- Min servo motor

Connections:

Moisture sensor VCC to Arduino 5V

Moisture sensor GND to Arduino GND

Moisture sensor A0 to Arduino A0

Servo motor VCC to Arduino 5V

Servo motor GND to Arduino GND

Servo Motor Signal to Arduino digital pin 9

Circuit diagram:

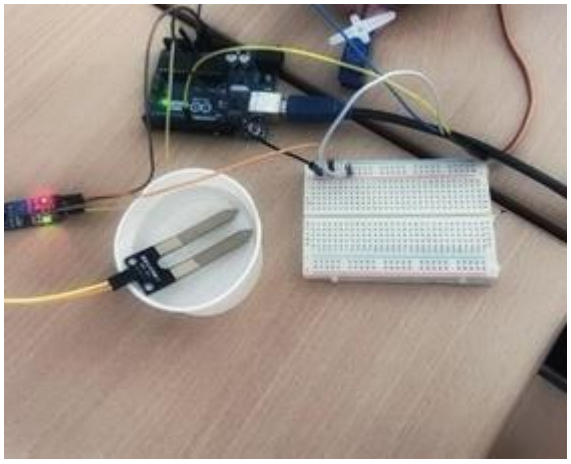
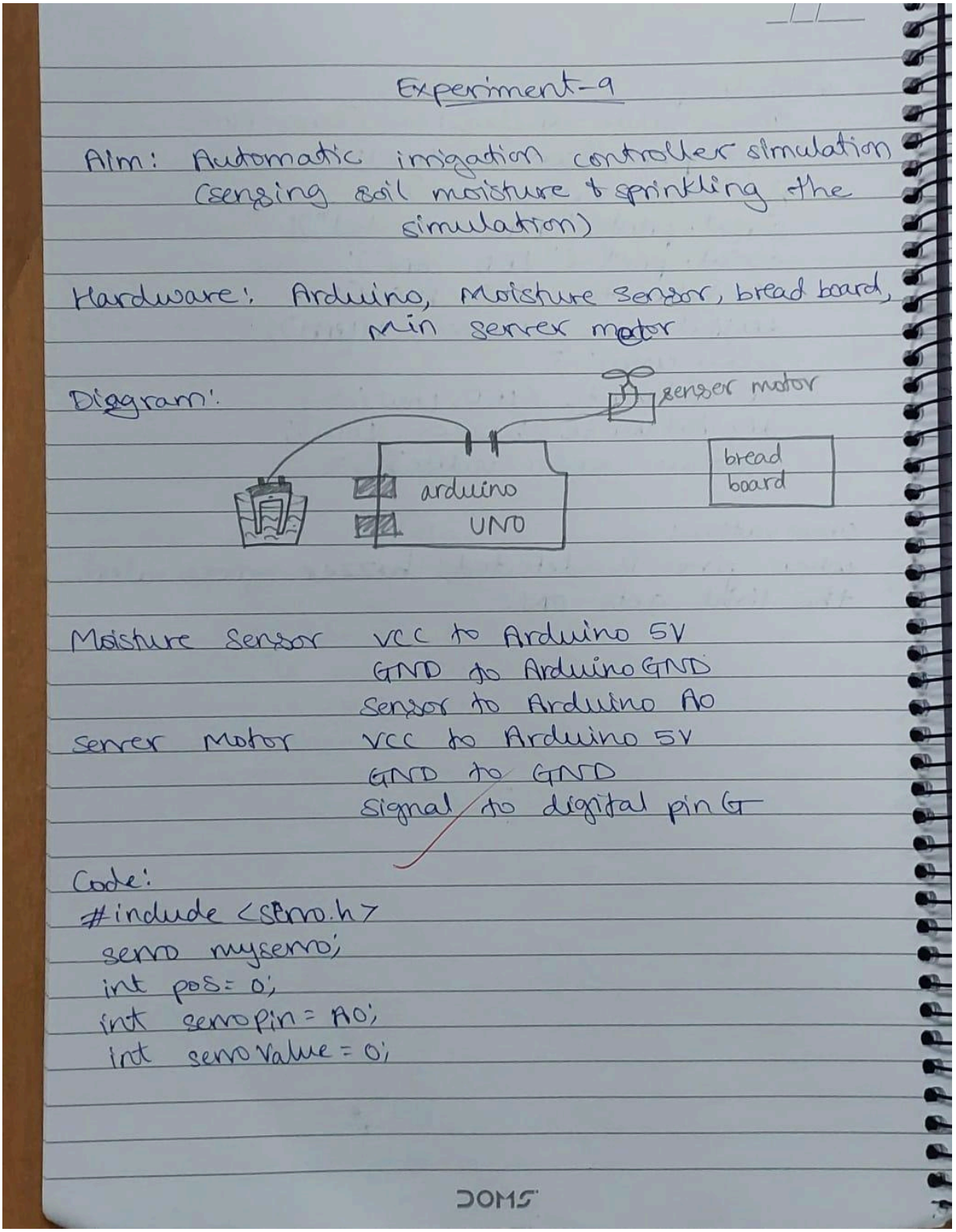


Fig 8- When moisture detected LED High, else Servo motor is on.

Handwritten code pic:



Experiment-9

Aim: Automatic irrigation controller simulation (sensing soil moisture & sprinkling the simulation)

Hardware: Arduino, Moisture sensor, bread board, Min server motor

Diggram:

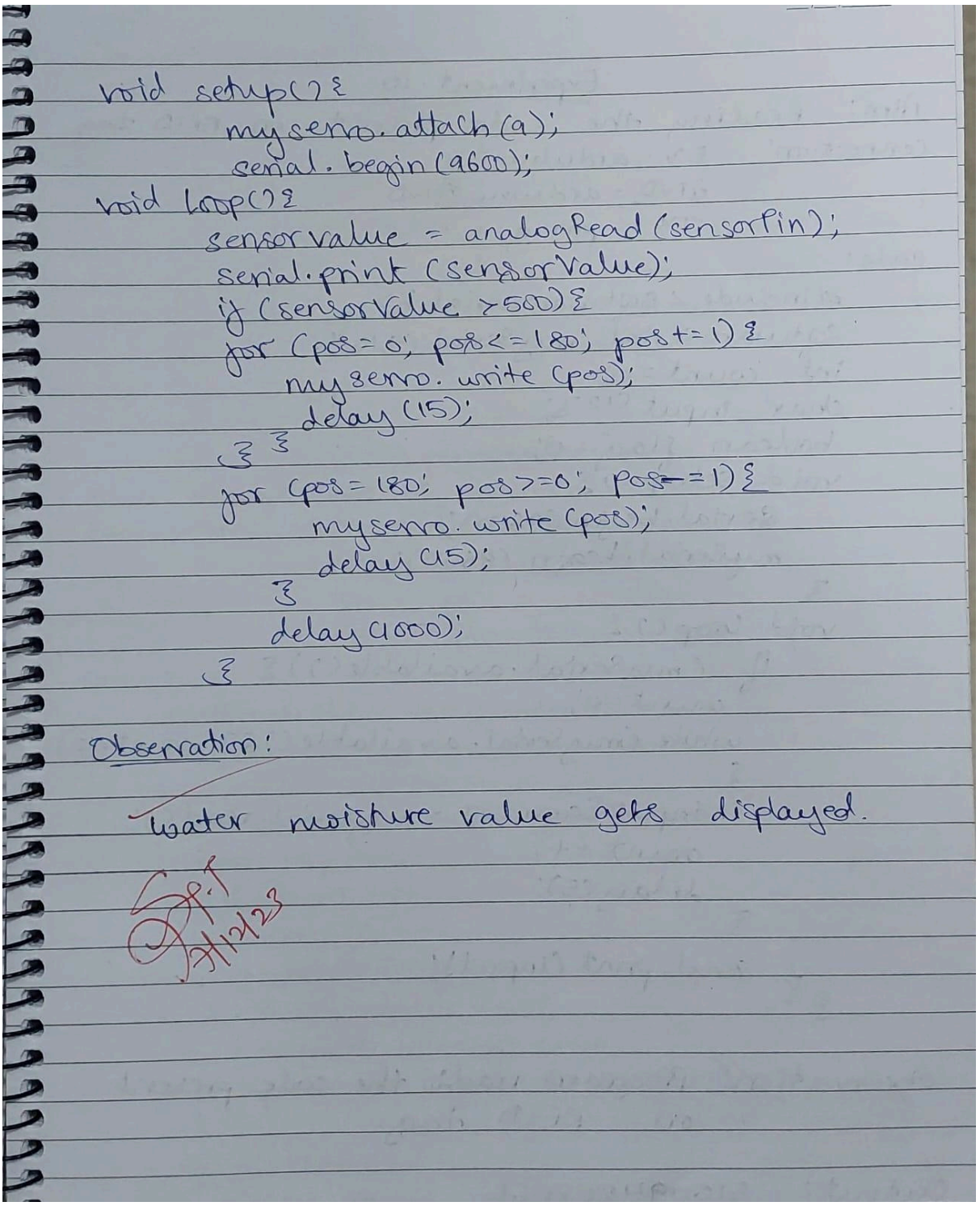
Moisture Sensor
VCC to Arduino 5V
GND to Arduino GND
Sensor to Arduino A0

Server Motor
VCC to Arduino 5V
GND to GND
Signal to digital pin 6

Code:

```
#include <Servo.h>
servo myservo;
int pos = 0;
int servoPin = A0;
int servoValue = 0;
```

DOMS



Code:

```
#include <Servo.h>;

Servo myservo;

int pos = 0;

int sensorPin = A0;
```

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

Observation:

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

9. Reading the code present on RFID tag

Aim:

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

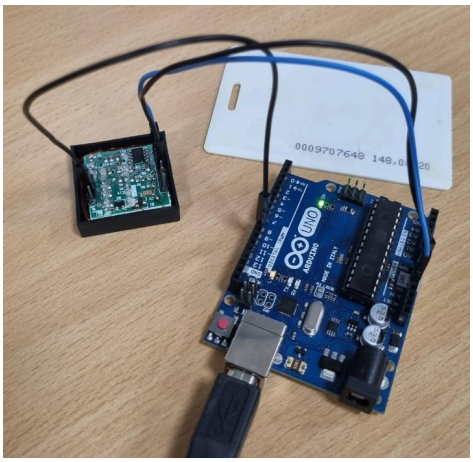
Connection:

5V-Arduino 5V

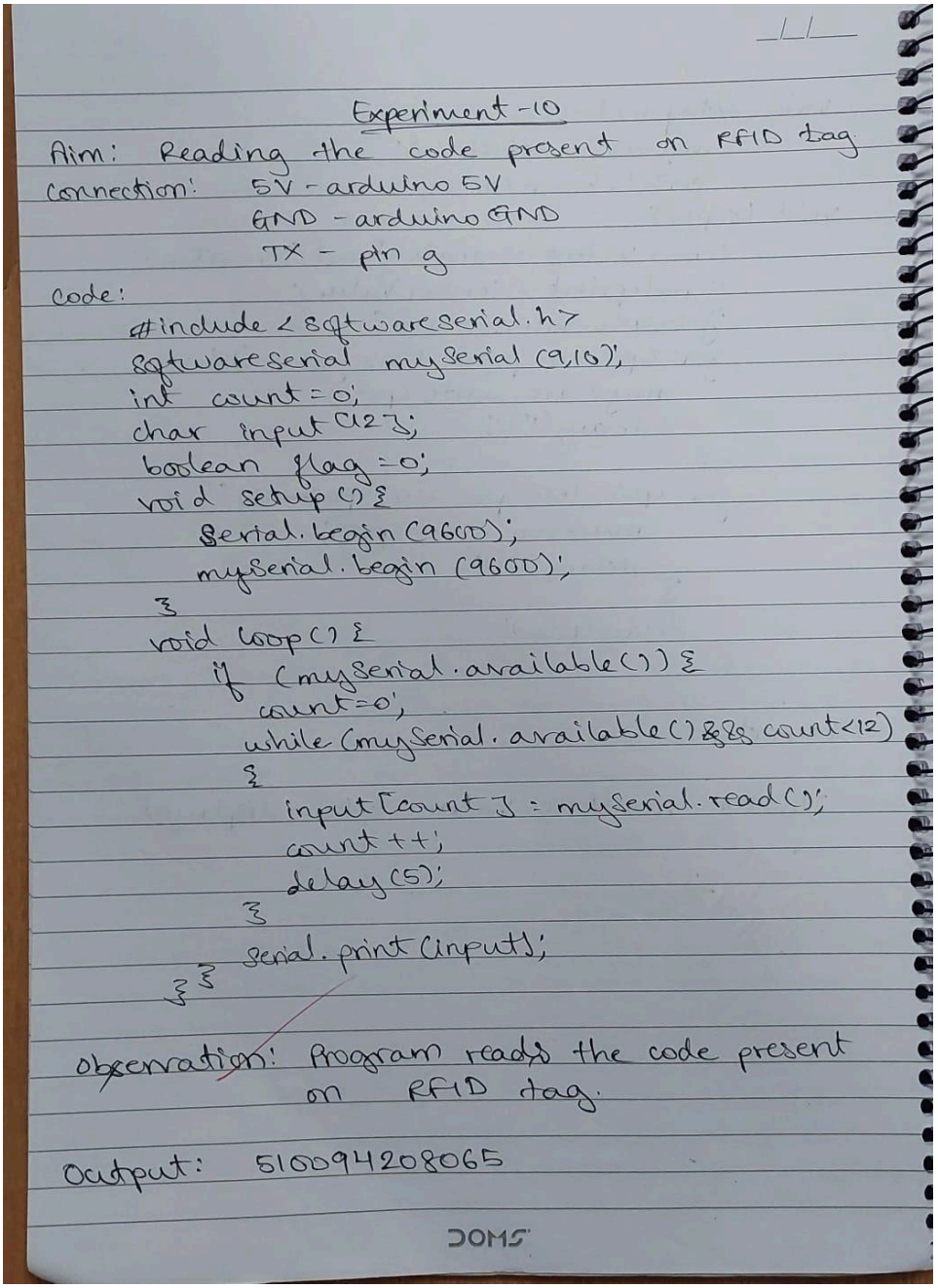
GND-Arduino GND

Tx-pin 9

Circuit diagram:



Handwritten code pic:



Experiment-10

Aim: Reading the code present on RFID tag

Connection: 5V - arduino 5V
GND - arduino GND
TX - pin 9

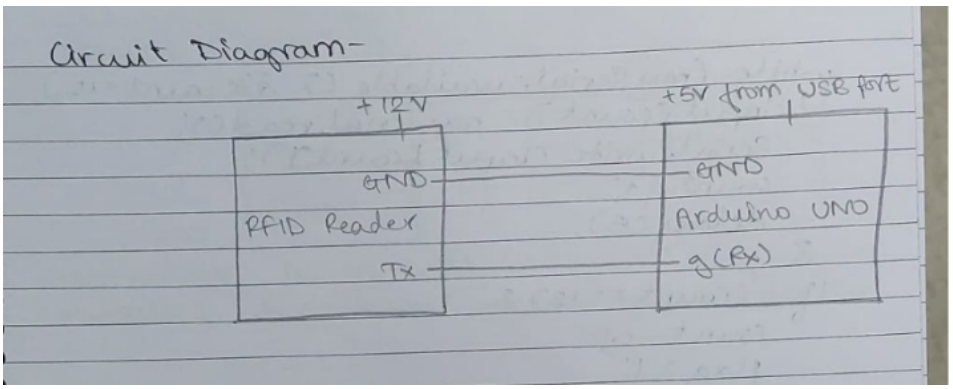
Code:

```
#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: Program reads the code present on RFID tag.

Output: 510094208065

DOMS



Code:

```
#include<SoftwareSerial.h>;
SoftwareSerial mySerial(9, 10);
int count = 0;                                  // count = 0
char input[12];                                  // character array of size 12
boolean flag = 0;                                 // flag =0
void setup()
{
  Serial.begin(9600);                             // begin serial port with baud rate 9600bps
  mySerial.begin(9600); }
void loop()
{   if(mySerial.available())
  {  count = 0;
    while(mySerial.available() && count < 12)
    {
      input[count] =mySerial.read();
      count++;
      delay(5);   }
    Serial.print(input);
}}
```

Observation:

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

10. Access control through RFID

Aim:

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

Connection:

5V-Arduino 5V

GND-Arduino GND

Tx-pin 9

Led-pin 12

Circuit diagram:

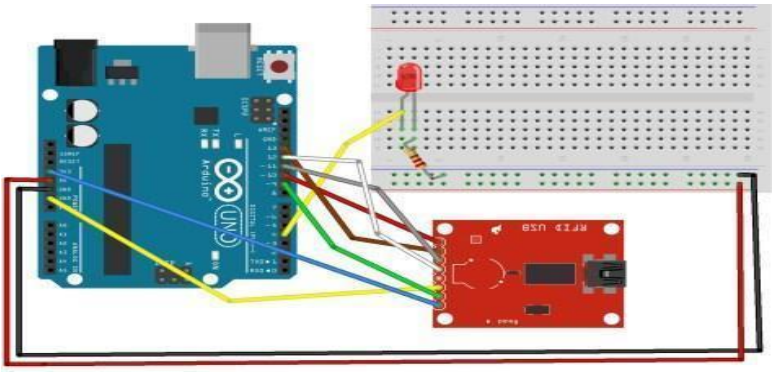
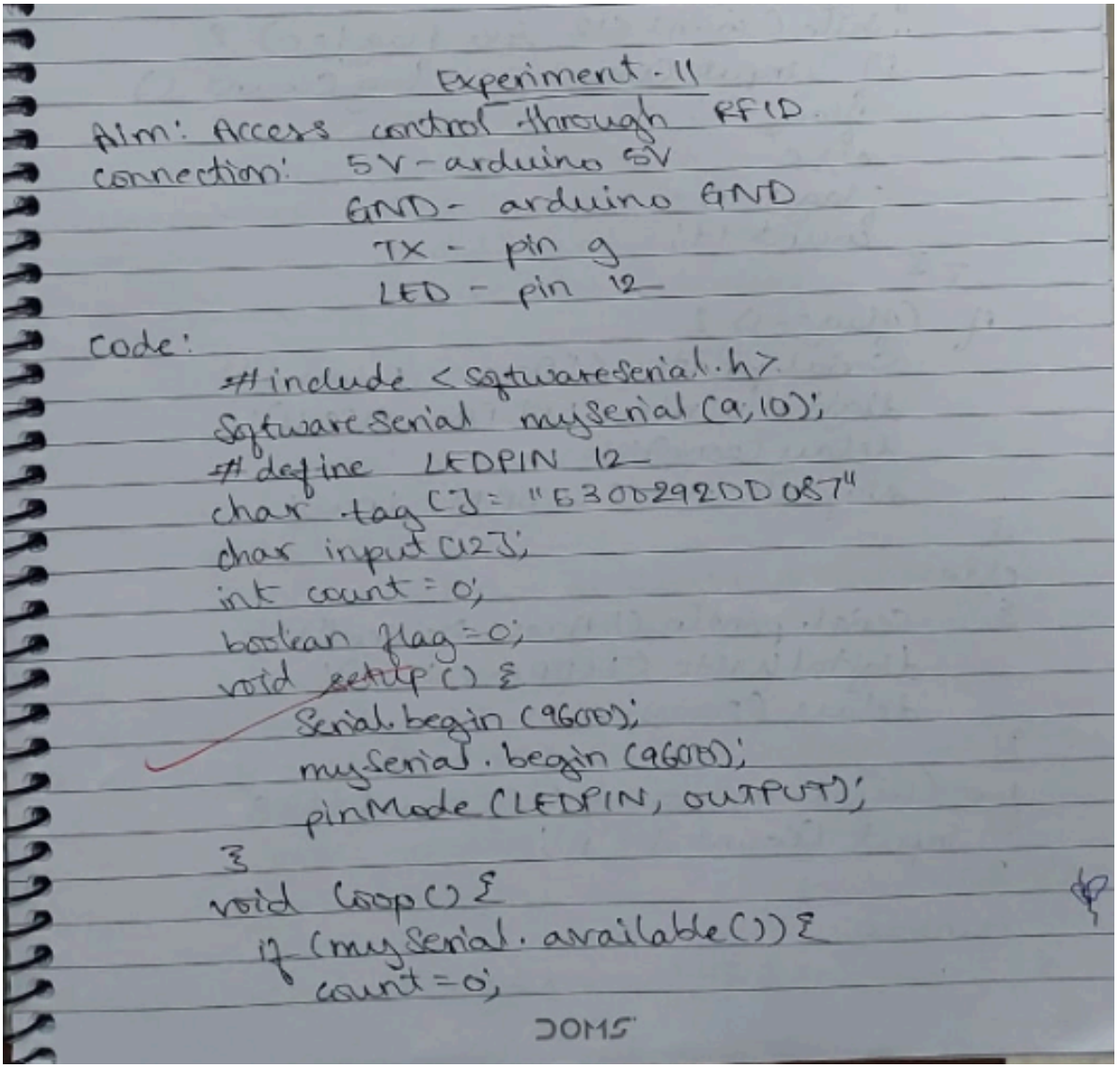


Fig.10.Access control through RFID

Handwritten code pic:



Experiment - 11

Aim: Access control through RFID

Connection: 5V - arduino 5V
GND - arduino GND
TX - pin 9
LED - pin 12

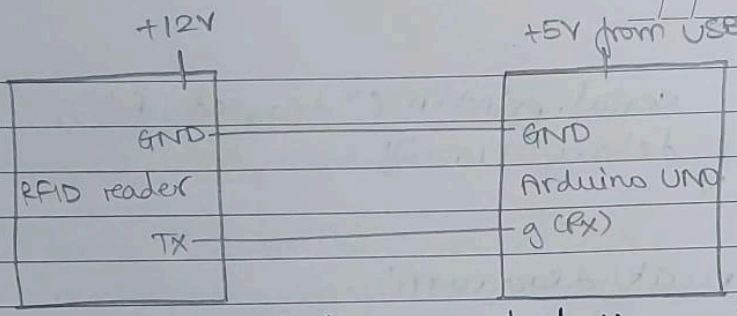
Code:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9,10);
#define LEDPIN 12
char tag[] = "5300292DD057"
char input[12];
int count = 0;
boolean flag = 0;
void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);
  pinMode(LEDPIN, OUTPUT);
}
void loop() {
  if (mySerial.available()) {
    count = 0;
```

DOMS

```
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==1) {
    Serial.println("Access Allowed")
    digitalWrite(LEDPIN, HIGH);
    delay(2000);
    digitalWrite(LEDPIN, LOW);
}
else
{   Serial.println("Access Denied");
    digitalWrite(LEDPIN, LOW);
    delay(2000);
}
for (count=0; count<12; count++) {
   input[count]='F';
}
count=0;
}
}
```

Circuit:



Observation: Access will be granted if code matches the tag, otherwise it's denied.

Experiment - 12

Aim: Temperature Sensing

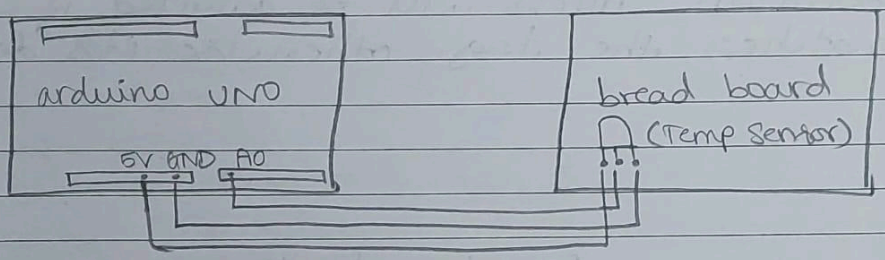
Hardware: Arduino UNO
Breadboard
Temperature Sensor

Code:

```
int SensorPin=0;
void setup() {
  serial.begin(9600);
}
void loop() {
  int reading = analogRead(sensorPin);
  float voltage = reading * 5.0/1024;
  serial.print(voltage);
  serial.println("volts");
  float temperatureC = (voltage-0.5) * 100;
  serial.print(temperatureC);
  serial.println("degrees C");
  float temperatureF = (temperatureC * 9/5) +
                       32;
  Serial.print(temperatureF);
```

```
serial.println("degrees F");
delay(1000);
}
```

Circuit diagram:



Observation:

Sensor senses the temperature of environment.

Sol:

Code:

```
#include<SoftwareSerial.h>;
SoftwareSerial mySerial(9, 10);
#define LEDPIN 12
char tag[] =”5300292DD087;” // Replace with your own Tag ID
char input[12]; // A variable to store the Tag ID being presented
int count = 0; // A counter variable to navigate through the input[] character array
boolean flag = 0; // A variable to store the Tag match status
void setup()
{
Serial.begin(9600);
mySerial.begin(9600);
pinMode(LEDPIN,OUTPUT);  }
void loop()
{
if(mySerial.available())
{     count = 0;
while(mySerial.available() && count < 12)
{
input[count] = mySerial.read();
count++; // increment counter
delay(5);
   }
if(count == 12)
 {
count =0; // reset counter varibale to 0
flag = 1;
while(count<12 && flag !=0)
```

```
{
if(input[count]==tag[count])
flag = 1;
else
flag=0;
count++; // increment i   }}
if(flag == 1) // If flag variable is 1, then it means the tags match
{
Serial.println(“Access Allowed!”);
digitalWrite(LEDPIN,HIGH);
delay (2000);
digitalWrite (LEDPIN,LOW); }
else
{
Serial.println(“Access Denied”); // Incorrect Tag Message
digitalWrite(LEDPIN,LOW);
delay(2000);    }
for(count=0; count<12; count++)
{
input[count]= 'F' ;
}
count = 0; // Reset counter variable
}
 }
```

Observation:

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

HC-05 Bluetooth Module

HC-05 PinOut (Right) :

• KEY: If brought HIGH before power is applied, forces AT Command Setup Mode.

LED blinks slowly (2 seconds)

• VCC: +5 Power

• GND: System / Arduino Ground

• TXD: Transmit Serial Data from HC-05 to Arduino Serial Receive. NOTE: 3.3V

HIGH level: OK for Arduino

• RXD: Receive Serial Data from Arduino Serial Transmit

• STATE: Tells if connected or not

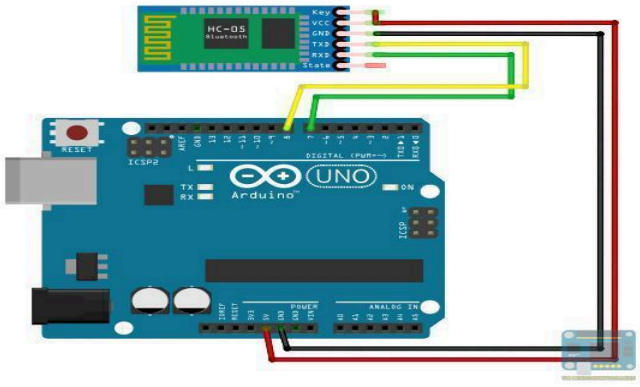
11. HC-05 at Command prompt

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

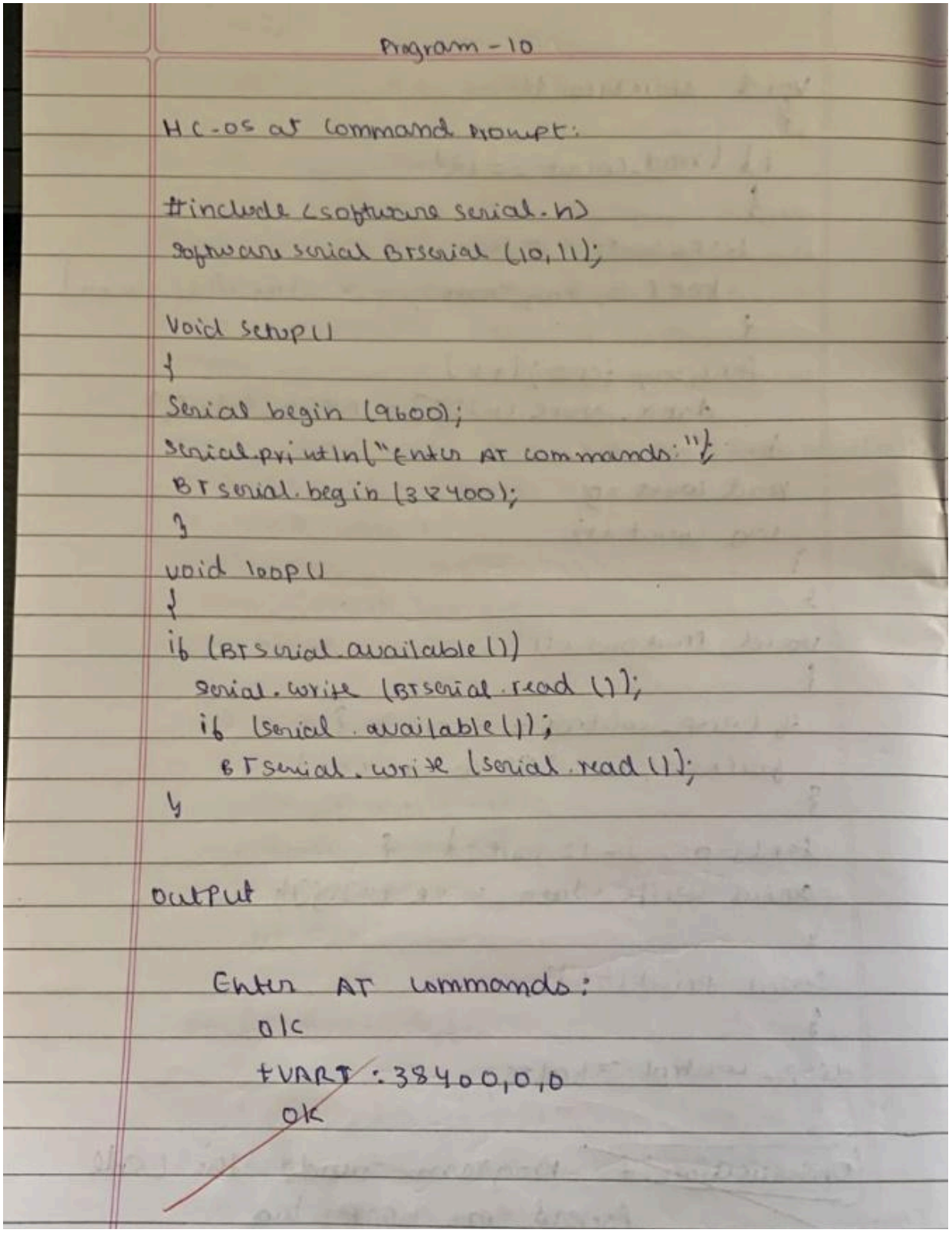
Hardware Required :

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

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



Handwritten code pic:



Program - 10

HC-05 at command prompt:

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

Void setup()
{
Serial.begin (9600);
Serial.println("Enter AT commands:");
BTserial.begin (38400);
}
void loop()
{
if (BTserial.available())
  Serial.write (BTserial.read());
  if (Serial.available());
    BTserial.write (Serial.read());
}
```

Output

```
Enter AT commands:
  OK
  +UART:38400,0,0
  OK
```

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()); }
```

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

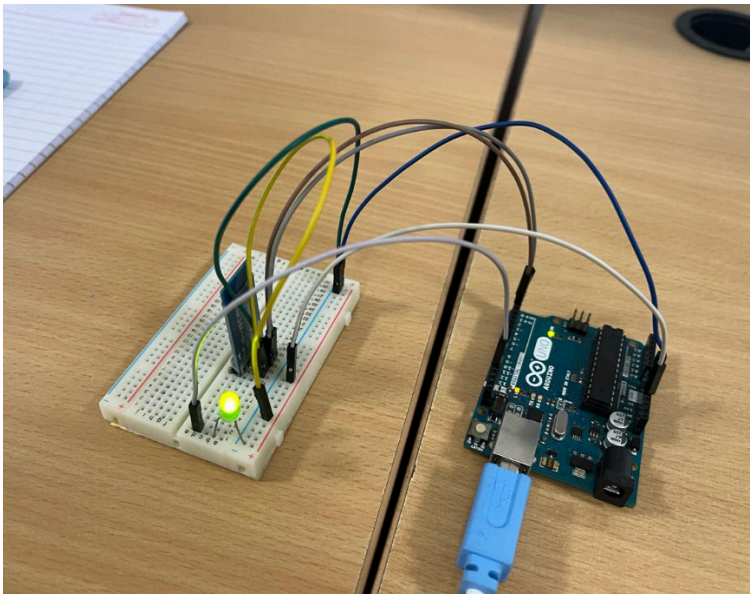
Connection:

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

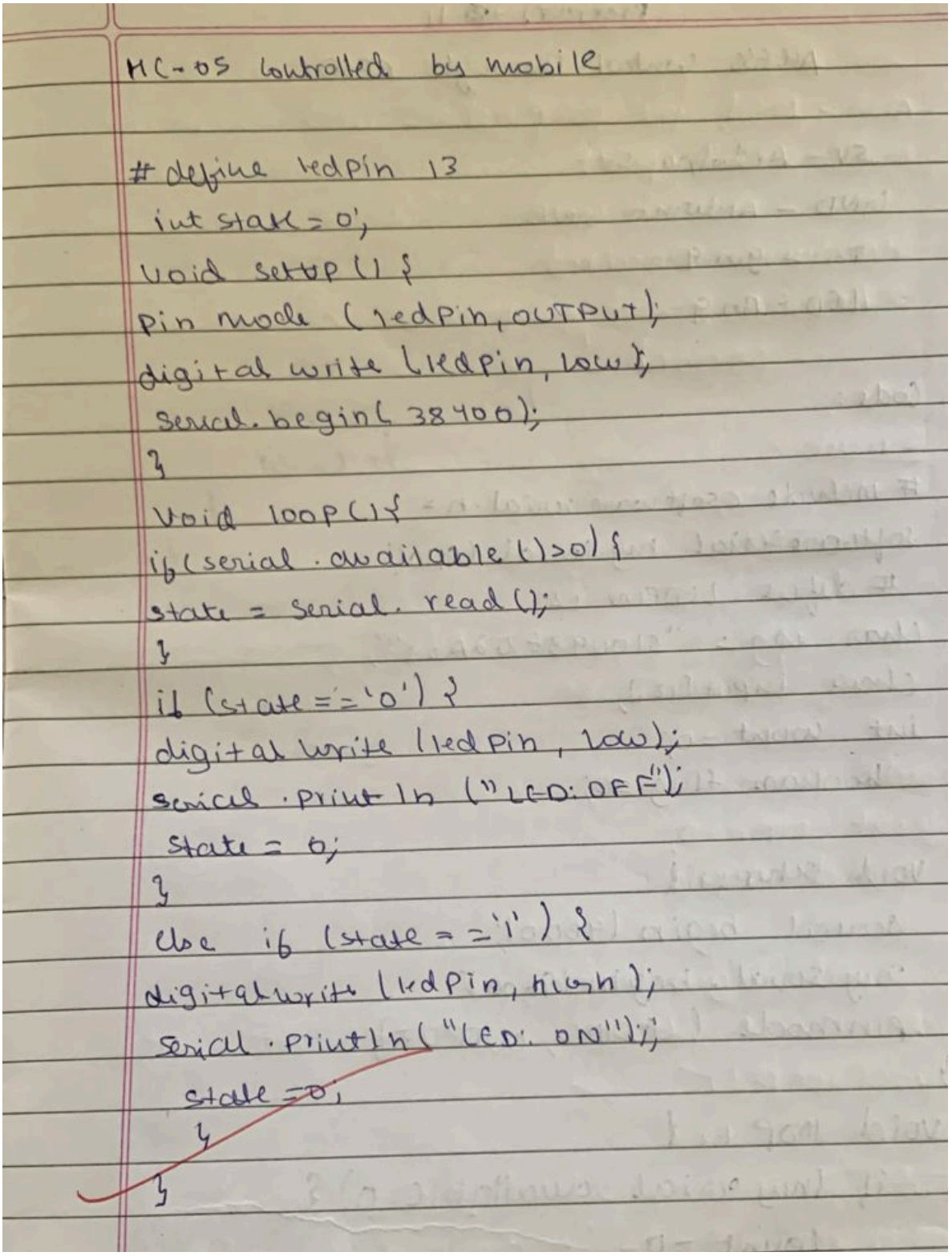
- Connect the TX pin of the HC-05 module to a digital pin on the Arduino (e.g., pin 2).
- Connect the RX pin of the HC-05 module to a digital pin on the Arduino (e.g., pin 3).
- Connect the VCC pin of the HC-05 module to the 5V pin on the Arduino.
- Connect the GND pin of the HC-05 module to the GND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



MC-05 controlled by mobile

```
#define ledpin 13
int state = 0;
void setup() {
pin mode (ledPin, OUTPUT);
digital write (ledpin, low);
Serial.begin(38400);
}
Void loop(){
if(serial.available()>0){
state = Serial.read();
}
if (state == '0') {
digital write (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);
}
void loop() {
  if(Serial.available() < 0)
{
    state = Serial.read(); // Reads the data from the serial port
 }
 if (state == “0”) {
  digitalWrite(ledPin, LOW); // Turn LED OFF
  Serial.println(“LED: OFF”);
  state = 0;
 }
 else if (state == “1”) {
  digitalWrite(ledPin, HIGH);
  Serial.println(“LED: ON”);;
  state = 0; } }
```

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:

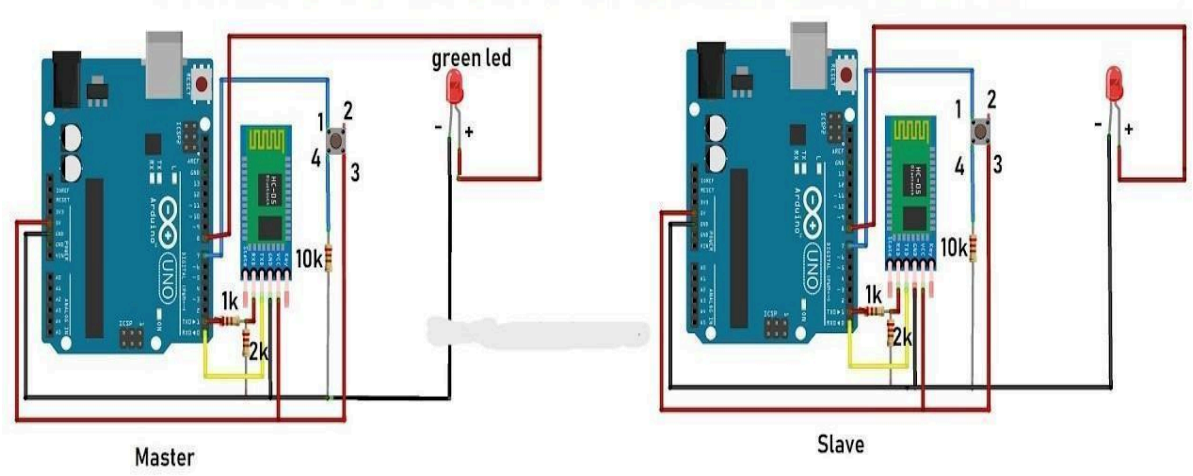
- Connect the TX pin to Arduino digital pin 10.
- Connect the RX pin to Arduino digital pin 11.
- Connect the VCC pin to Arduino 5V.
- Connect the GND pin to Arduino GND.

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.



BT-Slave Program:

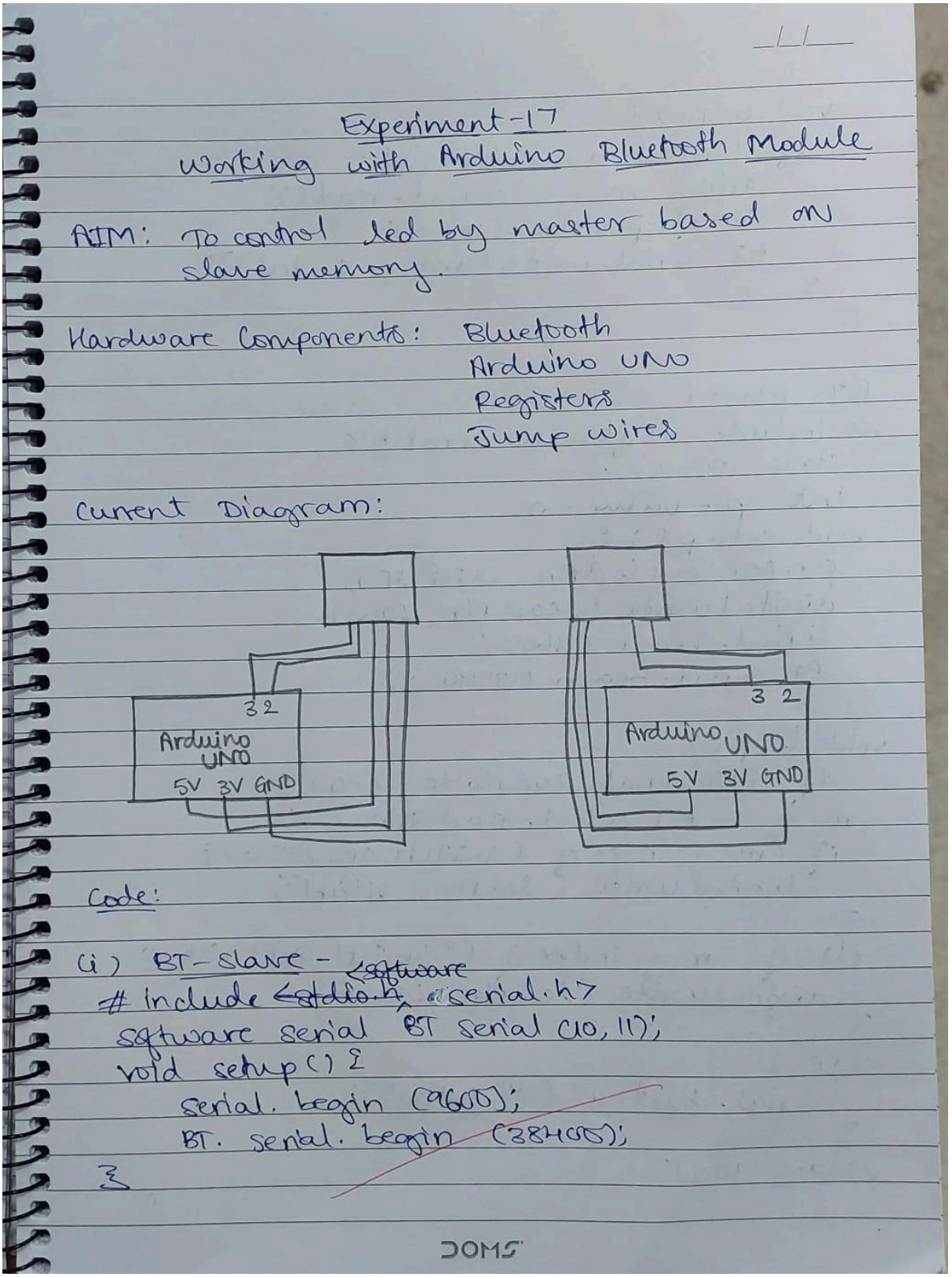
```
#include <SoftwareSerial.h>;
SoftwareSerial BTSerial(10, 11); // RX | TX
void setup() {
  Serial.begin(9600);
  BTSerial.begin(38400); // HC-05 default speed in AT command more
}
void loop() {
 if(Serial.available())
  {
   String message = Serial.readString();
   Serial.println (message);
    BTSerial.write(message.c_str());
     } }
```

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);
  }
```

Handwritten code pic:



Experiment-17

Working with Arduino Bluetooth Module

AIM: To control led by master based on slave memory.

Hardware Components: Bluetooth
Arduino UNO
Registers
Jump wires

Current Diagram:

Code:

(i) BT-Slave -

```
#include <stdio.h>, <serial.h>
Software serial BT serial (10, 11);
void setup() {
    Serial.begin (9600);
    BT. serial. begin (38400);
}
```

DOMS

```
void loop() {
  if (serial.available()) {
    string msg = serial.read();
    serial.println(msg);
    BT serial.write(msg.c_stal());
  }
}
```

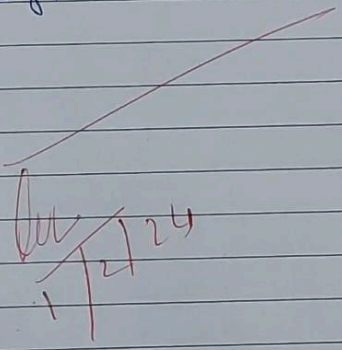
BT- Master

```
#include <software serial.h>
 sting msg;
 int pot value = 0;
void setup() {
  pinMode (ledPin, OUTPUT);
  digitalwrite (leadPin, LOW);
  Serial.begin(9600);
  BT serial.begin(38400);
}
void loop() {
   if (BT serial.available()>0) {
  msg = BT serial.readstring();
  if (msg.indexof (SWITCH ON)>=0) {
    digitalwrite (ledPin, HIGH);
  }
  else if (msg.indexof ("SWITCH OFF")>0) {
    digitalwrite (ledPin, LOW);
  }
  else {
    serial.futur ("Ating");
  }
  delay(100);
```

}
delay(10);
}

Observation

Based on msg. sent by slave-master program controls the LED.



14.GSM Module

1. GSM Module: Call to a particular number

Aim:

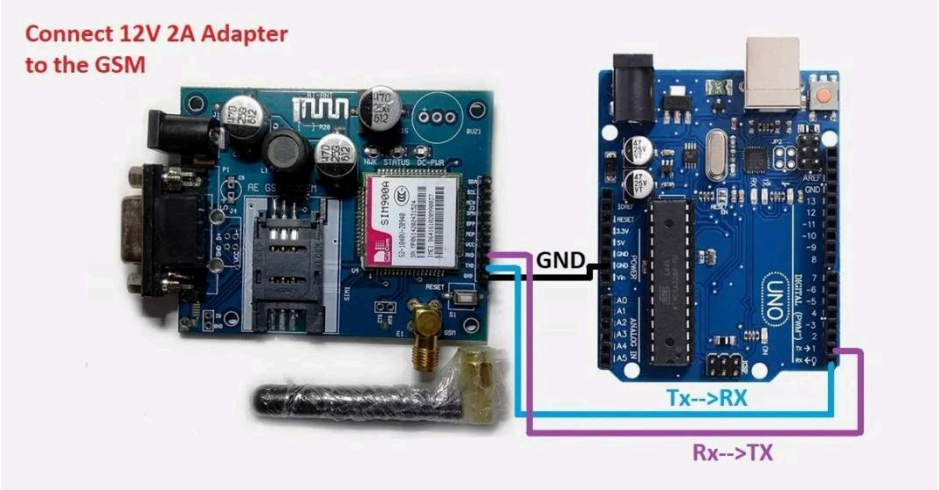
Call using Arduino and GSM Module – to a specified mobile number inside the program.

Hardware Required :

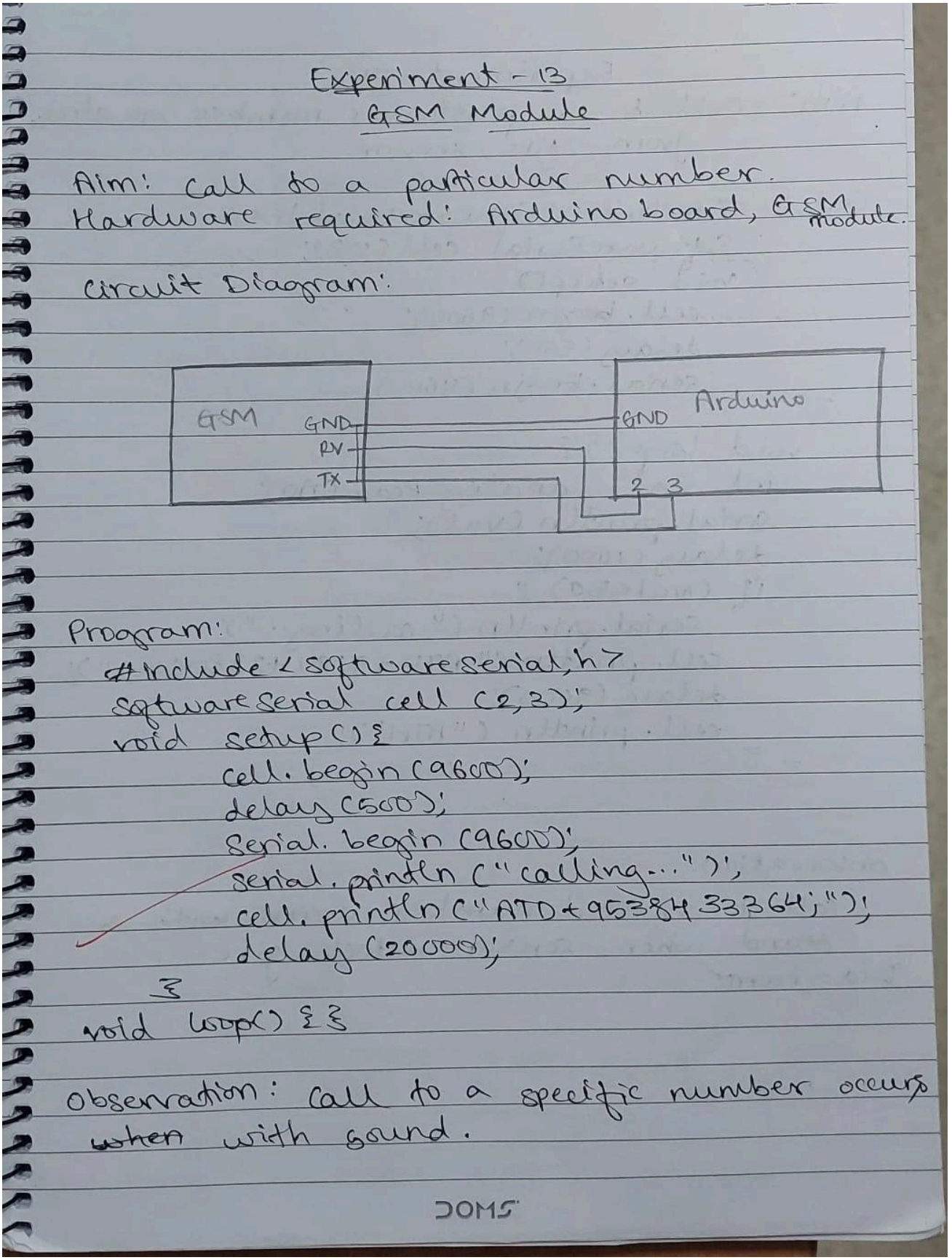
- Arduino Uno
- GSM Module
- SIM Card
- Power Supply
- Jumper wires

Connection:

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



Handwritten code pic:



Experiment - 13
GSM Module

Aim: Call to a particular number.
Hardware required: Arduino board, GSM module.

Circuit Diagram:

Program:

```
#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: Call to a specific number occurs when with sound.

DOMS

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.

2. Call to a particular number on an alert

Aim:

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

Hardware Required :

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

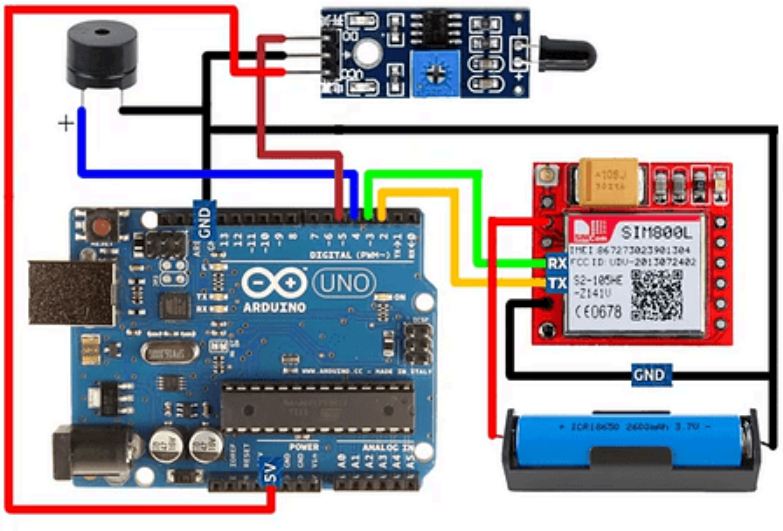
Connections :

1.GSM Module to Arduino :

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

2.Flame Sensor to Arduino :

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



Connections for flame sensor:

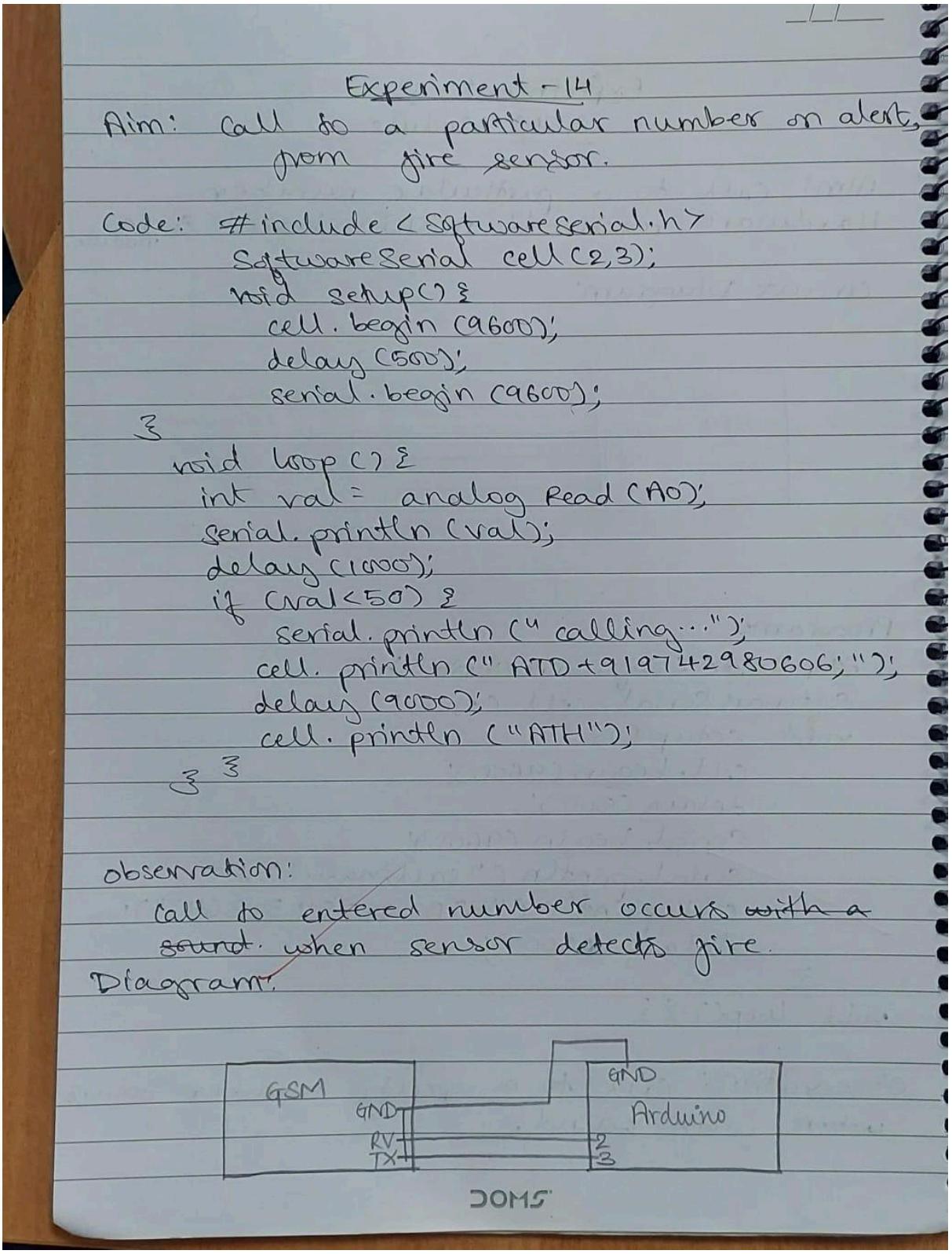
Arduino Flame Sensor

5V VCC

GND GND

A0 A0

Handwritten code pic:



Experiment - 14

Aim: Call to a particular number on alert, from fire sensor.

Code:

```
#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+919742980606;");
    delay(9000);
    cell.println("ATH");
}}
```

observation:

Call to entered number occurs ~~with a sound~~ when sensor detects fire.

Diagram:

DOMS

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&lt;50)
{
Serial.println(“CALLING.....”);
cell.println(“ATD+919742980606;”);
delay(10000);
cell.println(“ATH”); // Attention Hook Control
}
}
```

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.

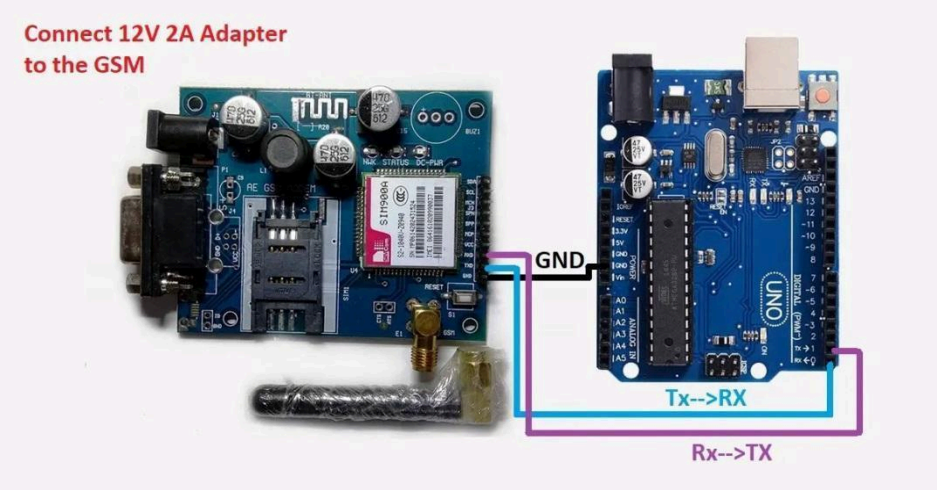
Hardware Required :

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

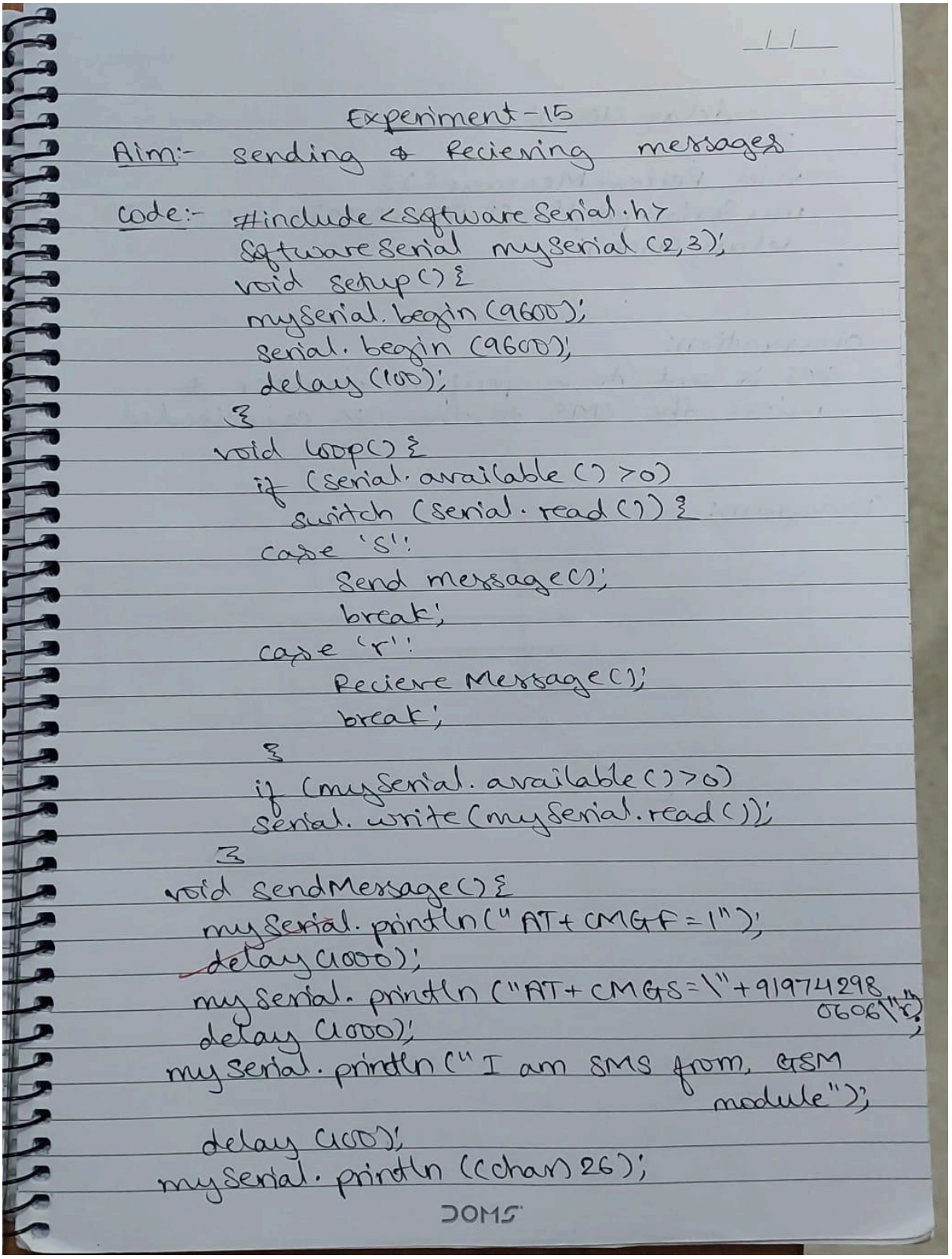
Connections :

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 pic:



Experiment - 15

Aim:- Sending & Recieving messages

Code:-

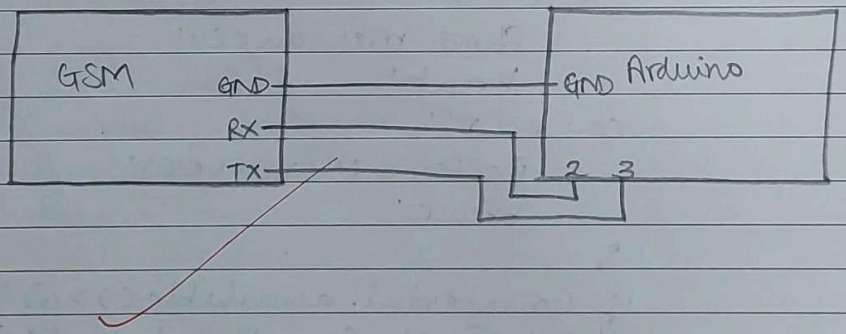
```
#include <SoftwareSerial.h>
SoftwareSerial myserial (2,3);
void setup () {
myserial.begin (9600);
Serial.begin (9600);
delay (100);
}
void loop () {
if (serial.available () >0)
switch (Serial.read ()) {
case 's':
    Send message ();
    break;
case 'r':
    Reciere Message ();
    break;
}
if (myserial.available () >0)
Serial.write (myserial.read ());
}
void sendMessage () {
myserial.println ("AT+CMGF=1");
delay (1000);
myserial.println ("AT+CMGS=\"+919974298
                                0606\"");
delay (1000);
myserial.println ("I am SMS from, GSM
                                module");
delay (100);
myserial.println ((char) 26);
```

DOMS

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

Observation:
SMS is sent to a specified number & recieve the SMS to the SIM card loaded.

Diagram:



Program:

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

through serial monitor respectively.

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3);

void setup()

{

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

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

delay(100);

}

void loop()

{

if (Serial.available()<0)

switch(Serial.read())

{

Case “s”:

SendMessage();

break;

case “r”:

RecieveMessage();

break;

}

if (mySerial.available()<0)

Serial.write(mySerial.read());

}

voidSendMessage()

{
```

```
mySerial.println(“AT+CMGF=1”); //Sets the GSM Module in Text Mode
//AT+CMGF,
SMS Format
delay(1000); // Delay of 1000 milli seconds or 1 second
mySerial.println(“AT+CMGS=\”+919742980606\”\r”); // AT+CMGS, Send Message
delay(1000);
mySerial.println(“I am SMS from GSM Module”);
delay(100);
mySerial.println((char)26);
delay(1000);
}
voidRecieveMessage()
{
mySerial.println(“AT+CNMI=2,2,0,0,0”);
delay(1000);
}
```

4. Controlling LED through received messages:

Aim:

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

LED.

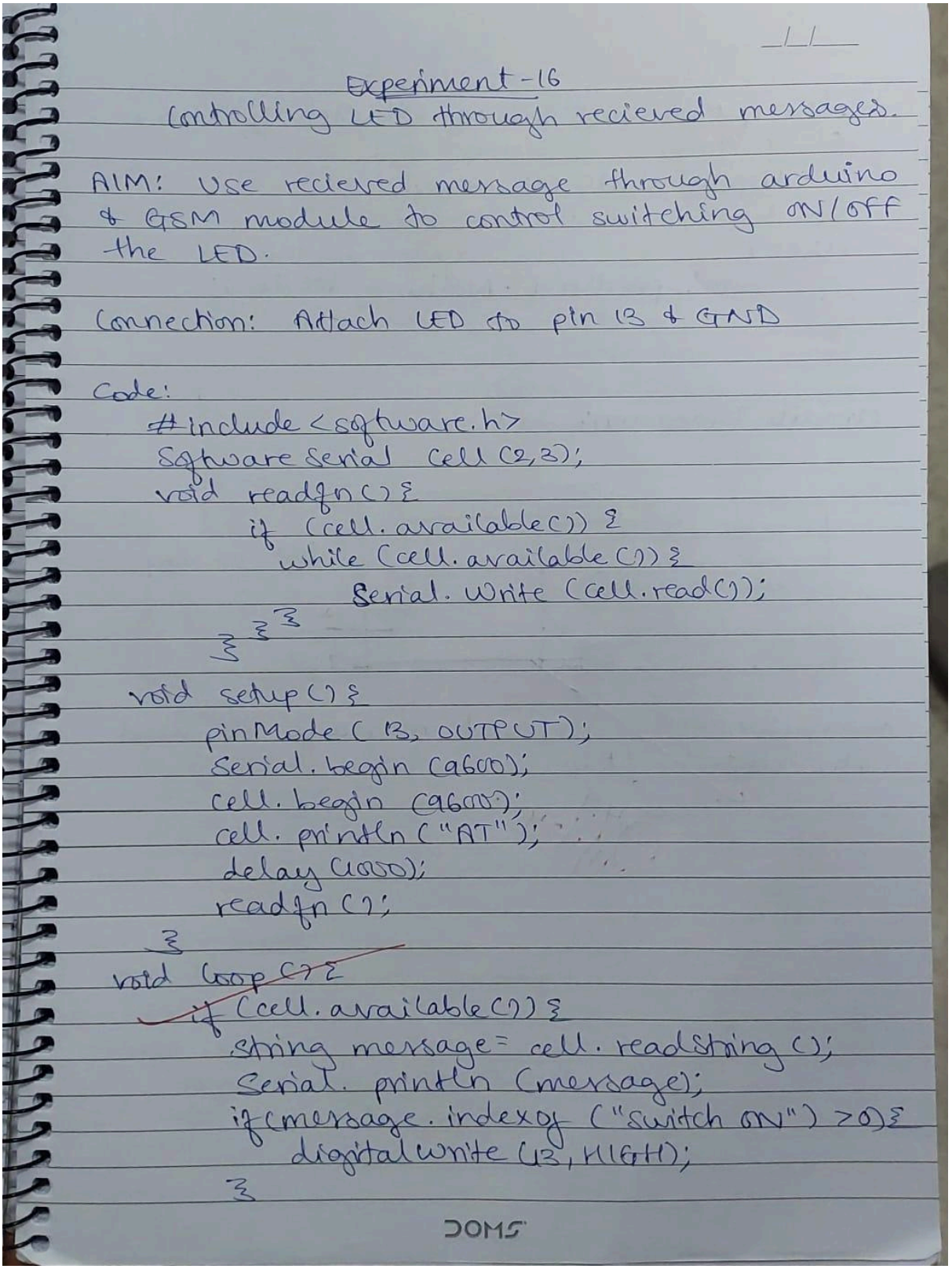
Connection: Attach LED to pin 13 and GND.

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 pic:



Experiment - 16

Controlling LED through recieved messages.

AIM: Use recieved message through arduino & GSM module to control switching ON/OFF the LED.

Connection: Attach LED to pin 13 & GND

Code:

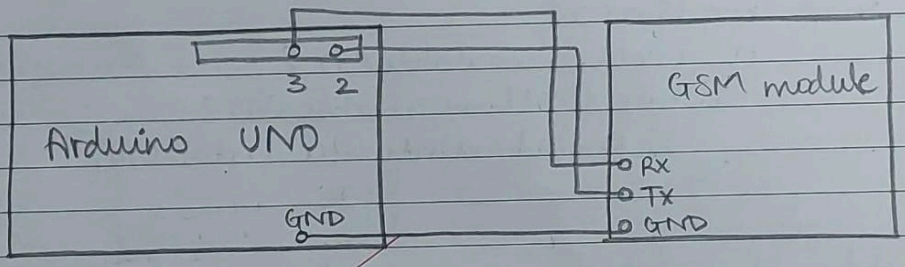
```
#include <sqftware.h>
Sqftware Serial Cell (2,3);
void readfn () {
    if (cell.available()) {
      while (cell.available()) {
          Serial.write (cell.read());
    }}}

void setup () {
    pinMode (13, OUTPUT);
    Serial.begin (9600);
    cell.begin (9600);
    cell.println ("AT");
    delay (1000);
    readfn ();
}
void loop () {
    if (cell.available()) {
      string message = cell.readString ();
      Serial.println (message);
      if (message.indexof ("switch ON") >0) {
          digitalwrite (13, HIGH);
      }
```

DOMS

```
else if (message.indexof ("switchoff") >0)
{
    digitalWrite (13, LOW);
}
else {
    serial.println (" Nothing to do");
}
}
```

Circuit Diagram:



observation: The recieved message is used to control LED.

9/1/24