VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

L5 + L6

IoT Fundamentals – ECE3501

Allen Ben Philipose – 18BIS0043

Lab Task – 4

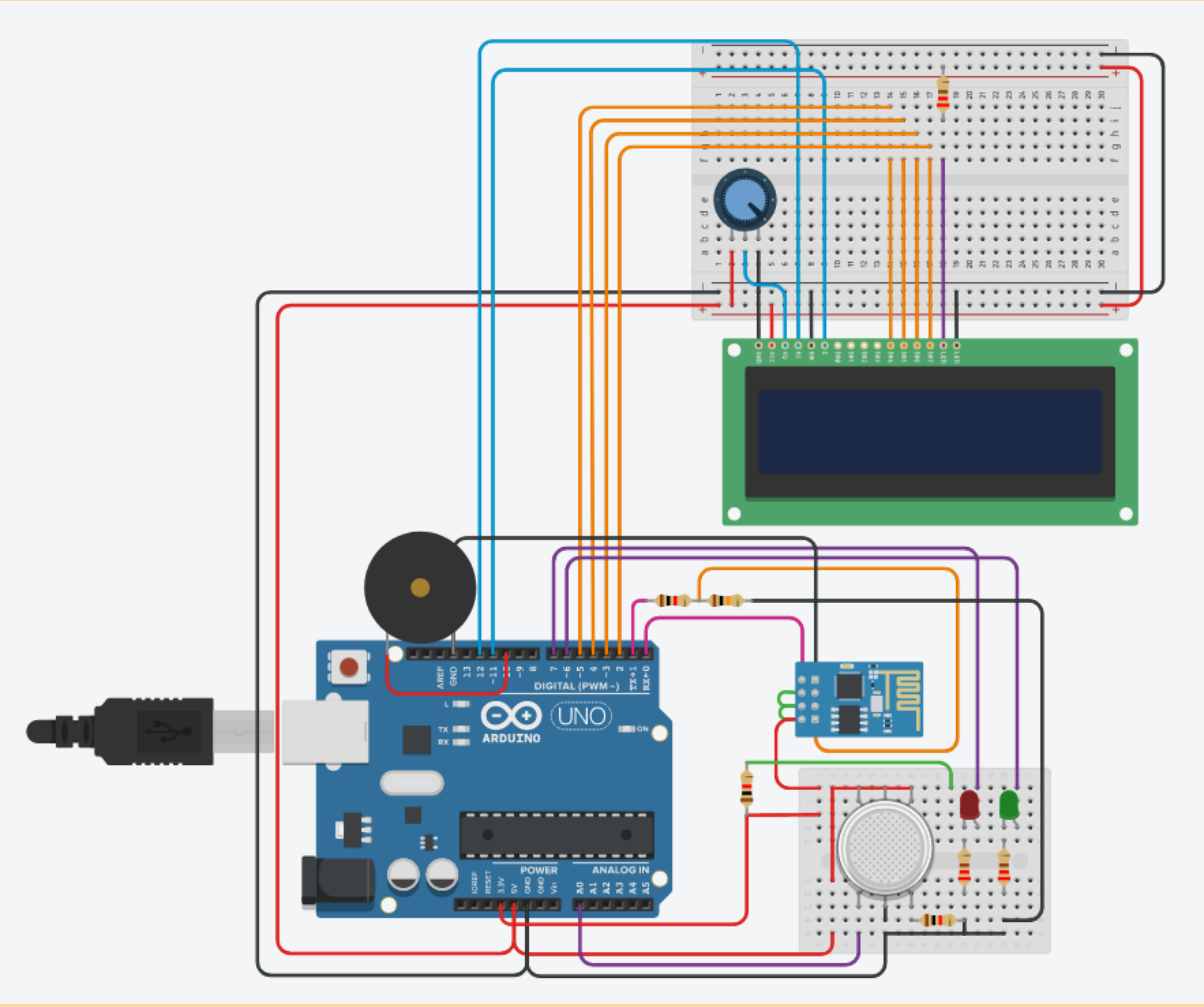
To: Prof. Suresh Chavhan

SMOKE DETECTION SYSTEM – I (OWN)

Aim

To design a circuit using Arduino for monitoring the room for any gases like smoke and if present, sound the alarm. Also transmit the detected levels to ThingSpeak for further analysis.

Circuit Diagram

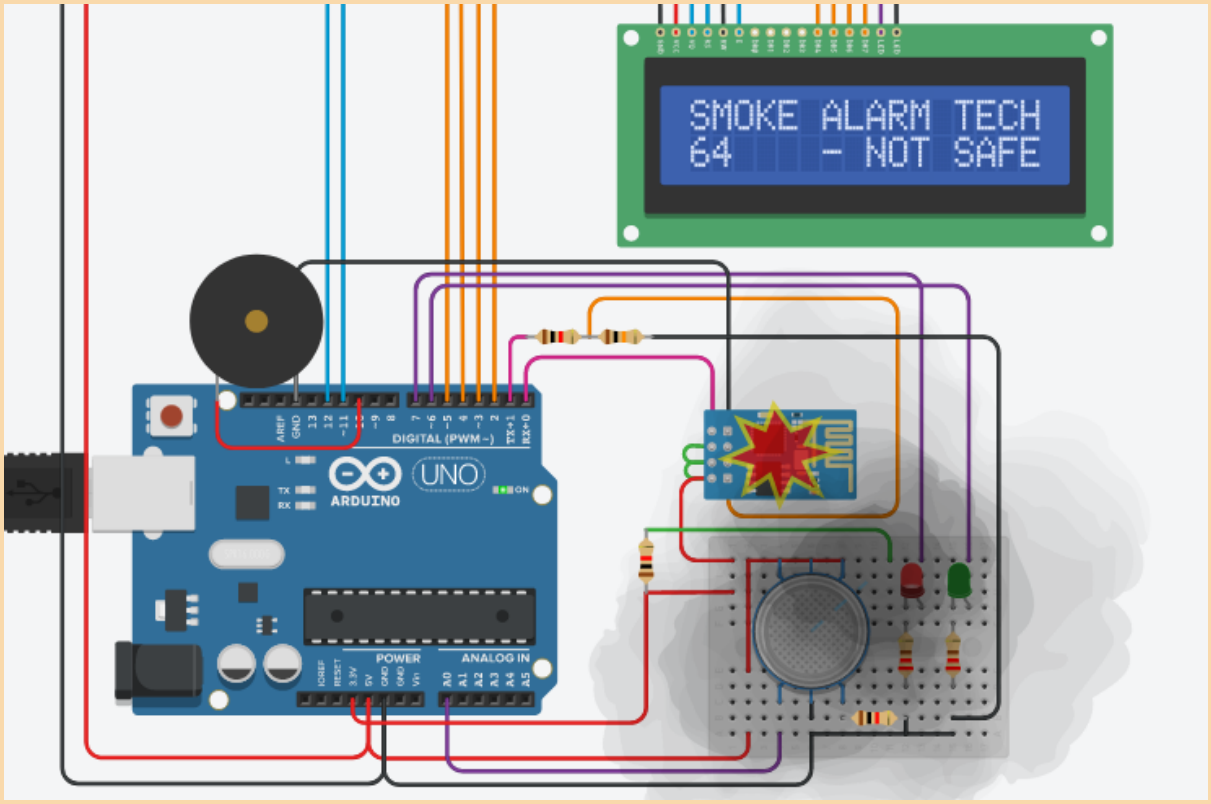


Tools Required

Tinkercad – for simulating the connection and coding of the Arduino circuit

ThingSpeak – for plotting the graph

Output from Tinkercad



Serial Monitor

```
AT+CIPSEND=85
GET /update?api_key=8XN8RYFFNZRKNYZS&field1=22 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=85
GET /update?api_key=8XN8RYFFNZRKNYZS&field1=70 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=85
GET /update?api_key=8XN8RYFFNZRKNYZS&field1=70 HTTP/1.1
Host: api.thingspeak.com
```

Code

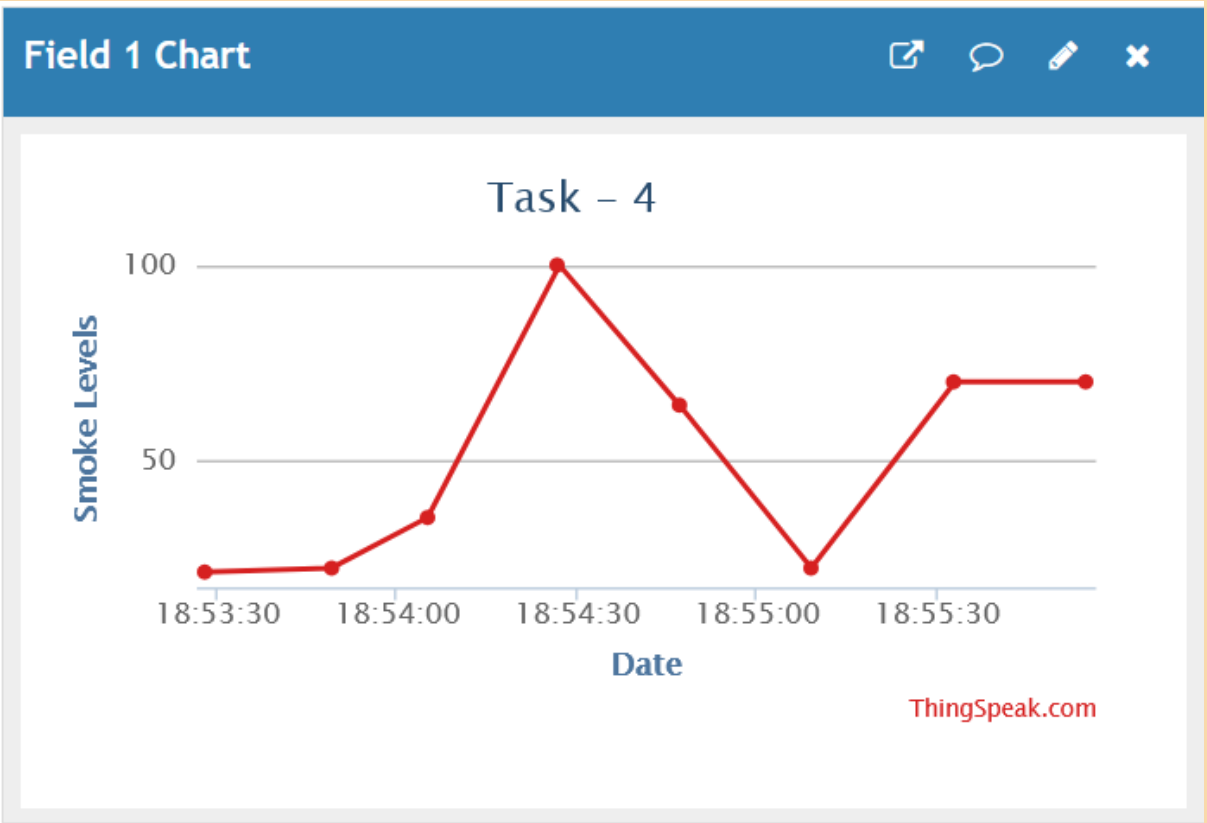
```
#include <LiquidCrystal.h>
String ssid     = "Simulator Wifi";
String password = "";
String host     = "api.thingspeak.com";
const int httpPort   = 80;
String      uri                                 =
"/update?api_key=8XN8RYFFNZRKNYZS&field4=";
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int setupESP8266(void) {
  // Start our ESP8266 Serial Communication
  Serial.begin(115200);
  Serial.println("AT");
  delay(10);
  if (!Serial.find("OK")) return 1;
  Serial.println("AT+CWJAP=\"" + ssid + "\",\""
+ password + "\"");
  delay(10);
  if (!Serial.find("OK")) return 2;
```

```
  // Open TCP connection to the host:
  Serial.println("AT+CIPSTART=\"TCP\",\""     +
host + "\"," + httpPort);
  delay(50);        // Wait a little for the ESP
to respond
  if (!Serial.find("OK")) return 3;
  return 0;
}
void anydata(int temp) {
  // Construct our HTTP call
  String httpPacket = "GET " + uri + String(temp)
+ " HTTP/1.1\r\nHost: " + host + "\r\n\r\n";
  int length = httpPacket.length();
  // Send our message length
  Serial.print("AT+CIPSEND=");
  Serial.println(length);
  delay(10); // Wait a little for the ESP to
respond if (!Serial.find(">")) return -1;
  // Send our http request
  Serial.print(httpPacket);
  delay(10); // Wait a little for the ESP to
respond
```

```
  if (!Serial.find("SEND OK\r\n")) return;
}

void setup() {
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  lcd.begin(16, 2);
  lcd.print("SMOKE ALARM TECH");
  setupESP8266();
}

void loop() {
  lcd.setCursor(0, 1);
  int temp = map(analogRead(A0),10,350,0,99);
  anydata(temp);
  lcd.print(temp);

if (temp>=30) {
    digitalWrite(6, LOW);
    digitalWrite(7, HIGH);
    tone(10, 1000, 200);
```

```
    if (temp>=100) {
      lcd.setCursor(3,1);
      lcd.print("   - NOT SAFE");
    }
    else {
      lcd.setCursor(2,1);
      lcd.print("    - NOT SAFE");
    }
  }

  else if (temp<30) {
    noTone(10);
    digitalWrite(6, HIGH);
    digitalWrite(7, LOW);
    lcd.setCursor(2,1);
    lcd.print("        - SAFE");
  }
}
```

Output from ThingSpeak



Observations

Program working as expected – minimal latency and accuracy of detecting smoke makes effective for analysis and for emergency procedures in a real-world scenario

Conclusion

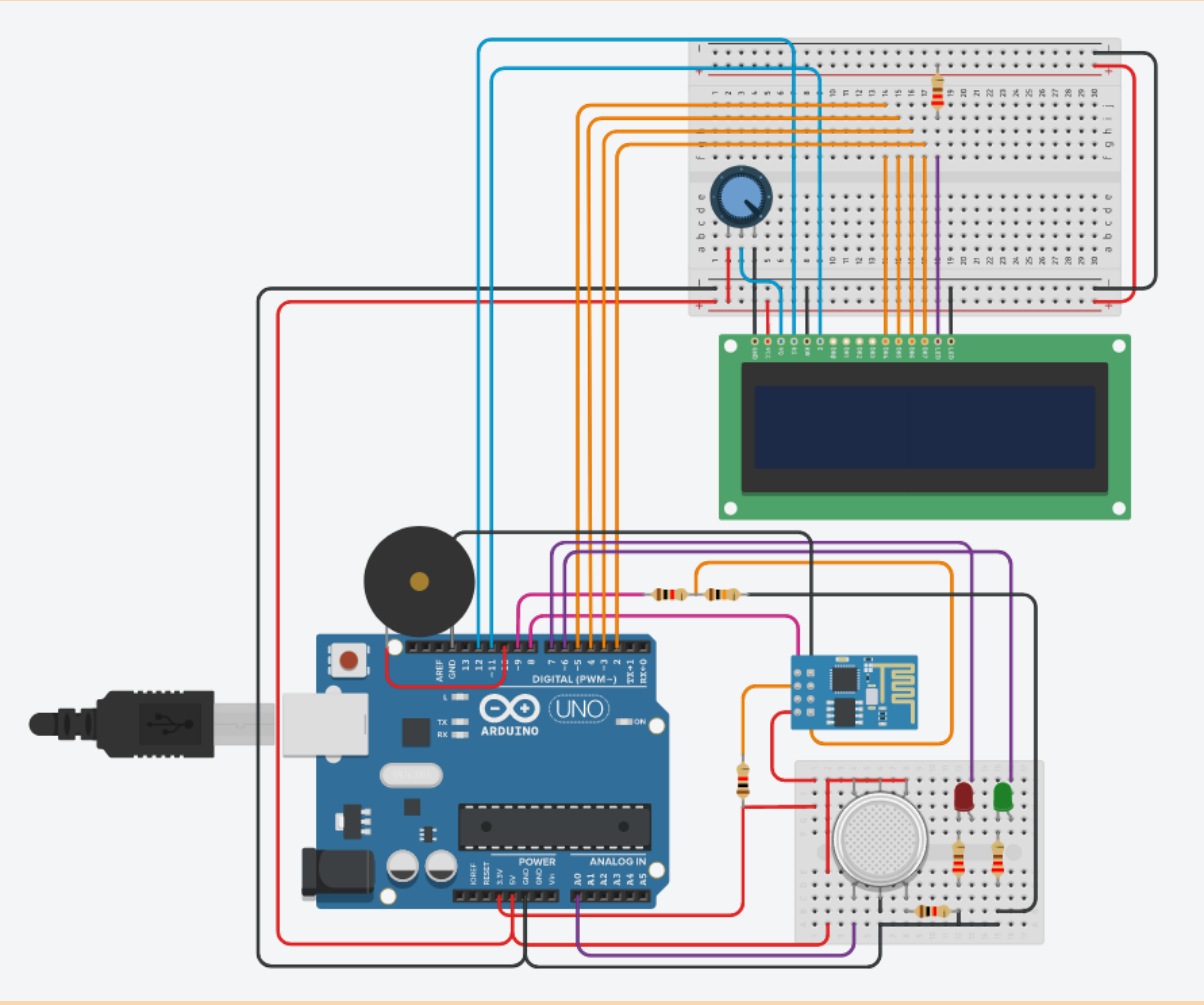
Therefore, by using Tinkercad, we simulated a circuit for a smoke detecting system and updated the live count in ThingSpeak for further analysis and storage.

SMOKE DETECTION SYSTEM – II (CLASS)

Aim

To design a circuit using Arduino for monitoring the room for any gases like smoke and if present, sound the alarm. Also transmit the detected levels to ThingSpeak for further analysis.

Circuit Diagram



Code – Will work on this after the CATs. Changes to be made in the boilerplate code

```
#include <SoftwareSerial.h>

#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // LCD Connections

SoftwareSerial SerCommESP8266(8,9); // RX, TX connect 8 to TX of ESP, connect 9 to RX of ESP


int smokeVal=0;

int smoke_sensor_pin=A0; // MQ2 Gas Sensor


int red_led_pin=7;  // Smoke indication

int green_led_pin=6; // No Smoke indication

int buzzer_pin = 10; // Buzzer


String apiKey = "8XN8RYFFNZRKNYZS";   // Write API key
```

```
void setup()
{
  pinMode(red_led_pin, OUTPUT);
  pinMode(green_led_pin, OUTPUT);
  pinMode(buzzer_pin, OUTPUT);
  pinMode(smoke_sensor_pin, INPUT);
  Serial.begin(9600);   //       serial   data
transmission at Baudrate of 9600
  SerCommESP8266.begin(9600);    // Initialize
the serial communication baud rate

  lcd.begin(16, 2);  // to intialize LCD
  lcd.setCursor(0,0);

  SerCommESP8266.println("AT");      //   Start
ESP8266 Module
  delay(100);
  SerCommESP8266.println("AT+GMR"); // To view
version info for ESP-01 output: 00160901 and
ESP-12 output: 0018000902-AI03
  delay(100);
```

```
  SerCommESP8266.println("AT+CWMODE=3"); // To determine WiFi mode
  delay(100);
  SerCommESP8266.println("AT+RST");    //    To restart the module
  delay(100);
  SerCommESP8266.println("AT+CIPMUX=1");     // Enable multiple connections 0: Single connection 1: Multiple connections (MAX 4)
  delay(100);
  String   cmd="AT+CWJAP=\"SSID   NAME\",\"SSID PASSWORD\""; // connect to Wi-Fi
  SerCommESP8266.println(cmd);
  delay(100);
  SerCommESP8266.println("AT+CIFSR"); // Return or get the local IP address
  delay(100);
}

void SetupESP8266_HA(){
   //              TCP              connection AT+CIPSTART=4,"TCP","184.106.153.149",80
```

```
    String cmd = "\nAT+CIPSTART=4,\"TCP\",\"";
// Establish TCP connection
    cmd += "184.106.153.149"; //
api.thingspeak.com
    cmd += "\",80"; // Port Number
    SerCommESP8266.println(cmd);
    Serial.println(cmd);
    if(SerCommESP8266.find("Error"))
    {
      Serial.println("AT+CIPSTART error");
      return;
    }
  String getStr = "GET /update?api_key="; //
API key
  getStr += apiKey;
  getStr +="&field1="; // Field variable as
Smoke
  getStr +=String(smokeVal);
  getStr += "\r\n\r\n";
  // send data length
  cmd = "AT+CIPSEND="; // Send data
AT+CIPSEND=id,length
```

```
  cmd += String(getStr.length());
  SerCommESP8266.println(cmd);
  Serial.println(cmd);
  delay(1000);
  SerCommESP8266.print(getStr);
  Serial.println(getStr);
  // thingspeak needs max 16 sec delay between
updates
  delay(10000);
}

void loop()
{
  delay(1000);
  smokeVal = map(analogRead(A0),10,350,0,100);
  Serial.println();
  lcd.clear();
  lcd.setCursor (0, 0);
  lcd.print (smokeVal);
  lcd.print (" In Room");
  lcd.setCursor (0,1);
```

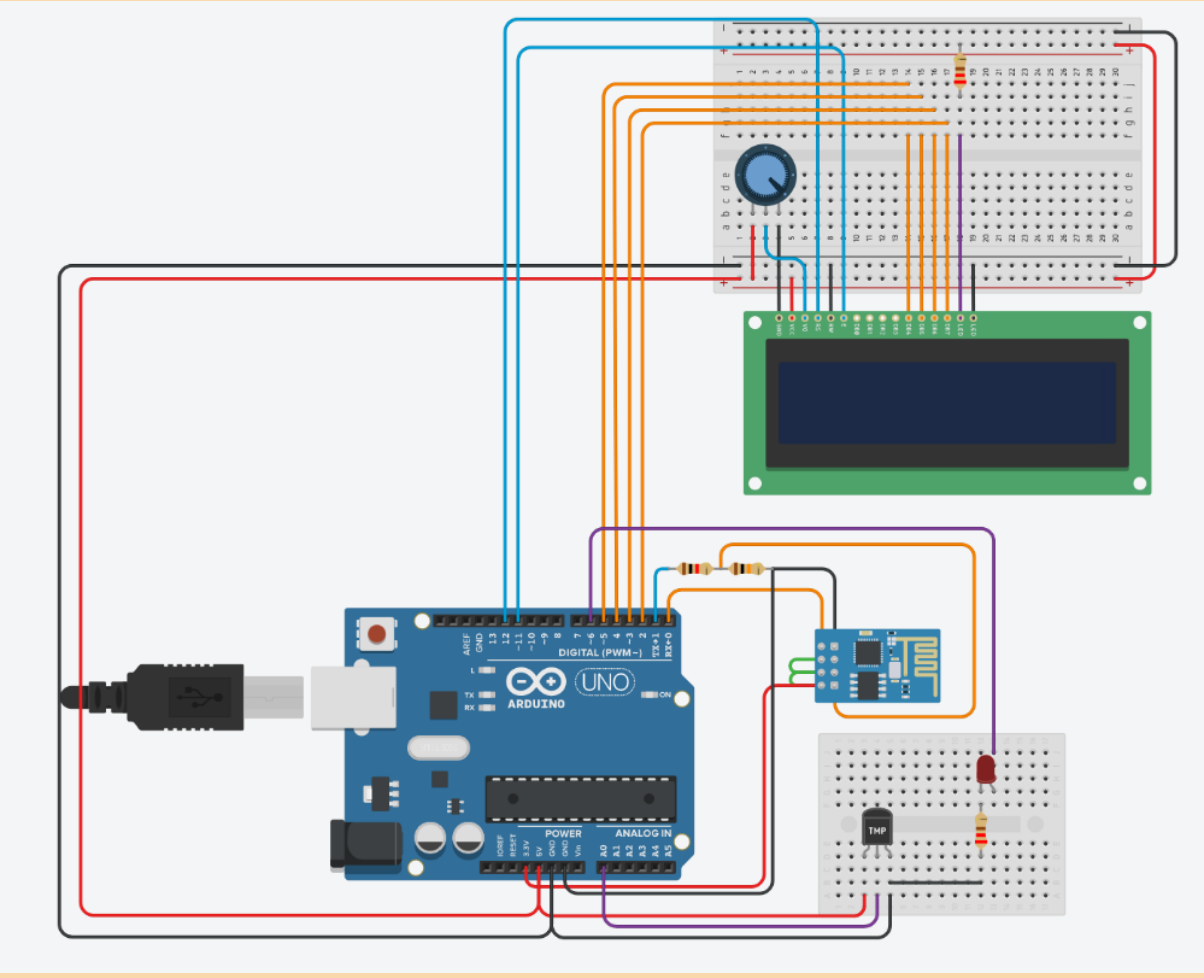
```
  if (smokeVal>30)
   {
    lcd.print("Smoke Detected");
    Serial.print("Smoke Detected");
    digitalWrite(red_led_pin, HIGH);
    digitalWrite(green_led_pin, LOW);
    tone(buzzer_pin, 1000, 200);
   }

 else
   {
    lcd.print("Safe");
    Serial.print("Safe");
    digitalWrite(red_led_pin, LOW);
    digitalWrite(green_led_pin, HIGH);
    noTone(buzzer_pin);
   }

SetupESP8266_HA();  //  For  ThingSpeak  Data
Transfer
}
```

PATIENT MONITORING SYSTEM

Aim

To design a circuit using Arduino for monitoring the temperature of the patient and transmit it to ThingSpeak for further analysis and diagnosis

Circuit Diagram

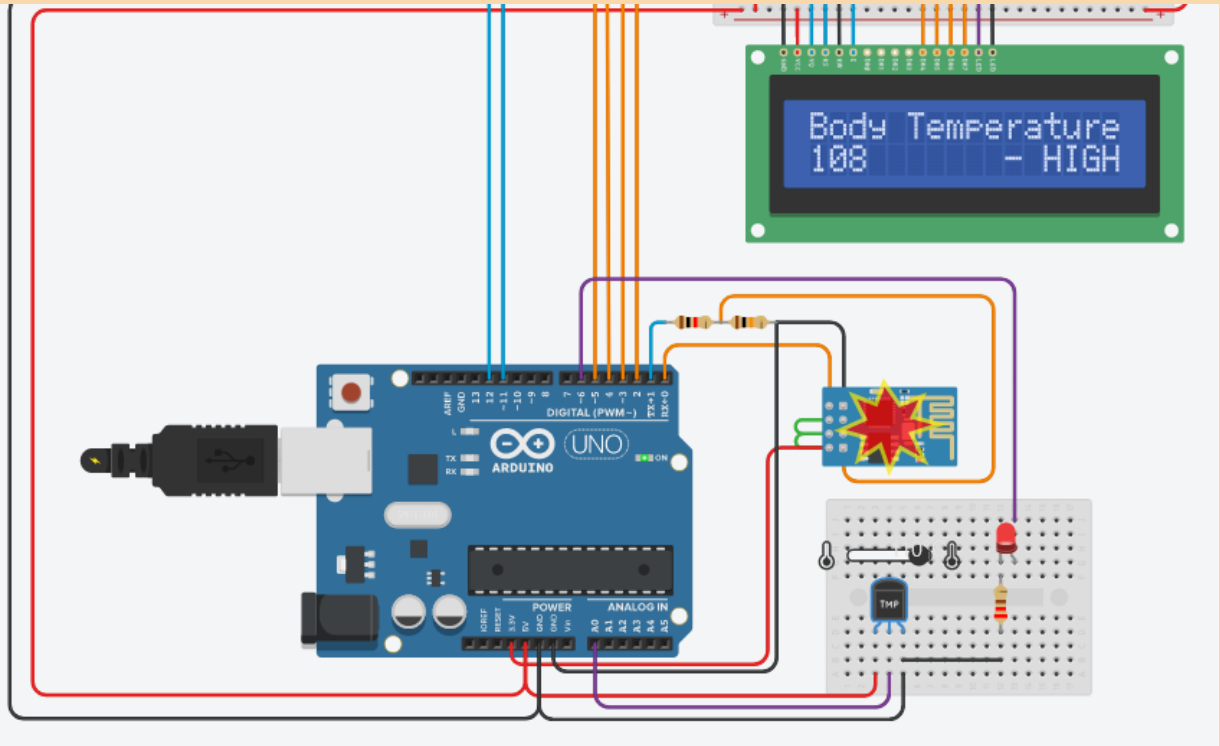


Tools Required

Tinkercad – for simulating the connection and coding of the Arduino circuit

ThingSpeak – for plotting the graph

Output from Tinkercad



Serial Monitor

```
GET /update?api_key=RFQBTTSMEQIRMKZ1&field1=97 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=85
GET /update?api_key=RFQBTTSMEQIRMKZ1&field1=98 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=86
GET /update?api_key=RFQBTTSMEQIRMKZ1&field1=100 HTTP/1.1
```

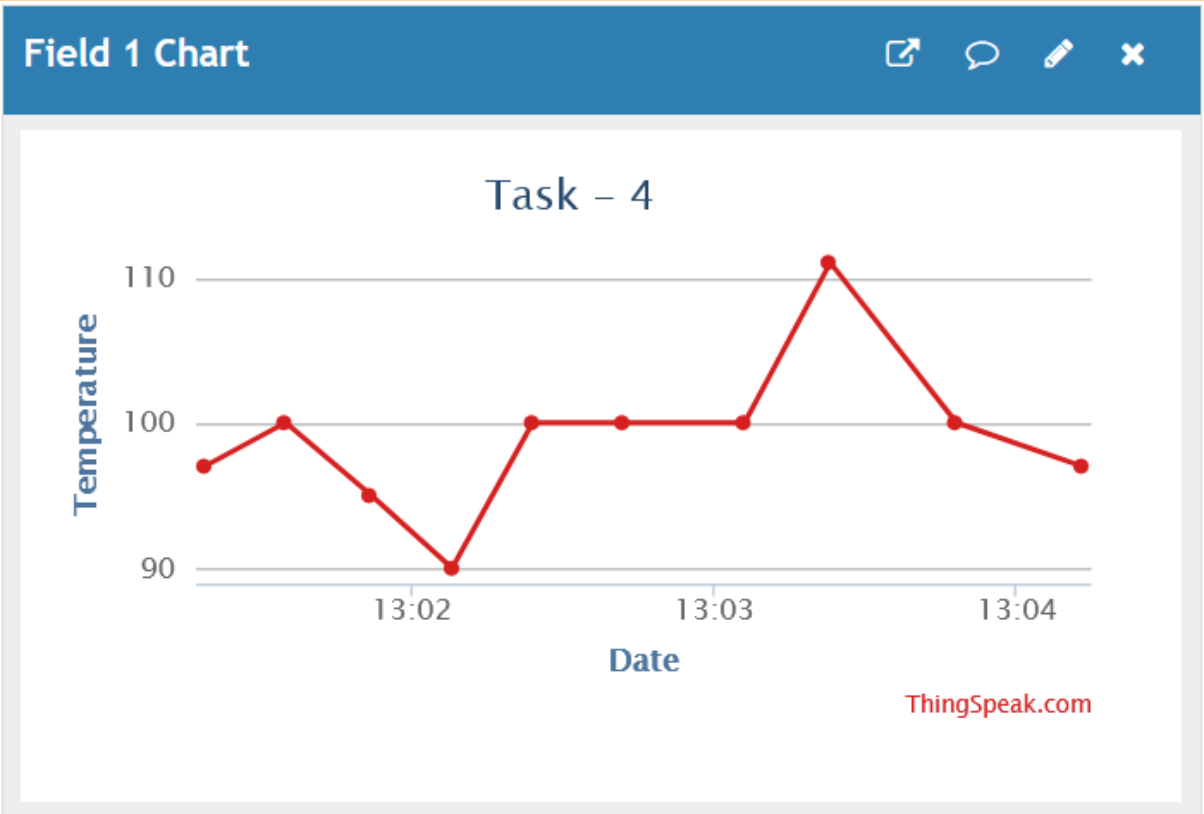
Code

```
#include <LiquidCrystal.h>
String ssid     = "Simulator Wifi";
String password = "";
String host     = "api.thingspeak.com";
const int httpPort   = 80;
String      uri                                 =
"/update?api_key=8XN8RYFFNZRKNYZS&field3=";
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int setupESP8266(void) {
  // Start our ESP8266 Serial Communication
  Serial.begin(115200);
  Serial.println("AT");
  delay(10);
  if (!Serial.find("OK")) return 1;
  Serial.println("AT+CWJAP=\"" + ssid + "\",\""
+ password + "\"");
  delay(10);
  if (!Serial.find("OK")) return 2;
```

```
  // Open TCP connection to the host:
  Serial.println("AT+CIPSTART=\"TCP\",\""     +
host + "\"," + httpPort);
  delay(50);        // Wait a little for the ESP
to respond
  if (!Serial.find("OK")) return 3;
  return 0;
}
void anydata(int temp) {
  // Construct our HTTP call
  String httpPacket = "GET " + uri + String(temp)
+ " HTTP/1.1\r\nHost: " + host + "\r\n\r\n";
  int length = httpPacket.length();
  // Send our message length
  Serial.print("AT+CIPSEND=");
  Serial.println(length);
  delay(10); // Wait a little for the ESP to
respond if (!Serial.find(">")) return -1;
  // Send our http request
  Serial.print(httpPacket);
  delay(10); // Wait a little for the ESP to
respond
```

```
  if (!Serial.find("SEND OK\r\n")) return;
}
void setup() {
  pinMode(6, OUTPUT);
  lcd.begin(16, 2);
  lcd.print("Body Temperature");
  setupESP8266();
}
void loop() {
  lcd.setCursor(0, 1);
  int temp = map(analogRead(A0),20,358,90,110);
  anydata(temp);
  lcd.print(temp);
  if (temp>98) {
    digitalWrite(6, HIGH);
    lcd.setCursor(10,1);
    lcd.print("- HIGH");
  }
  else {digitalWrite(6, LOW);}
  lcd.print("                ");
}
```

Output from ThingSpeak



Observations

Program working as expected – minimal latency and accuracy of mapping temperature makes it easily diagnosable and thus it is also effective for analysis.

Conclusion

Therefore, by using Tinkercad, we simulated a circuit for a patient temperature monitoring system and updated the live situation in ThingSpeak for further analysis and storage.