Visvesvaraya National Institute
of
Technology, Nagpur

Embedded Systems (ECP403)

Lab Report

Submitted By :
Shruti Murarka (BT18ECE099)
Semester 5
Electronics and Communication Engineering Dept.

Submitted To :
Dr. Ankit A Bhurane
Course Instructor

Contents

Experiment-1: Touch sensor.LED & Serial Monitor.

Aim: To integrate Touch sensor & LED with values printed on Serial Monitor using esp32.

Problem Statement: To glow in-built LED on ESP32 with touch as input. Also check the trigger in serial monitor.

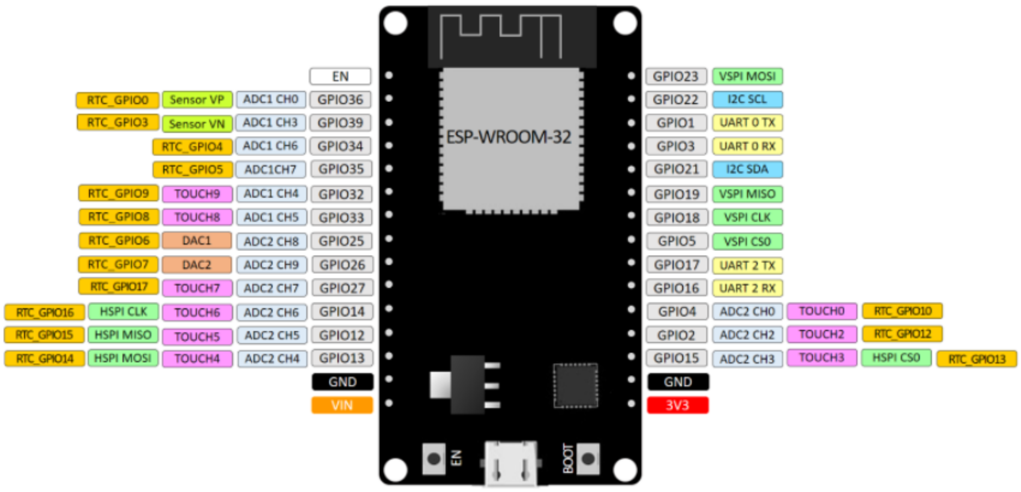
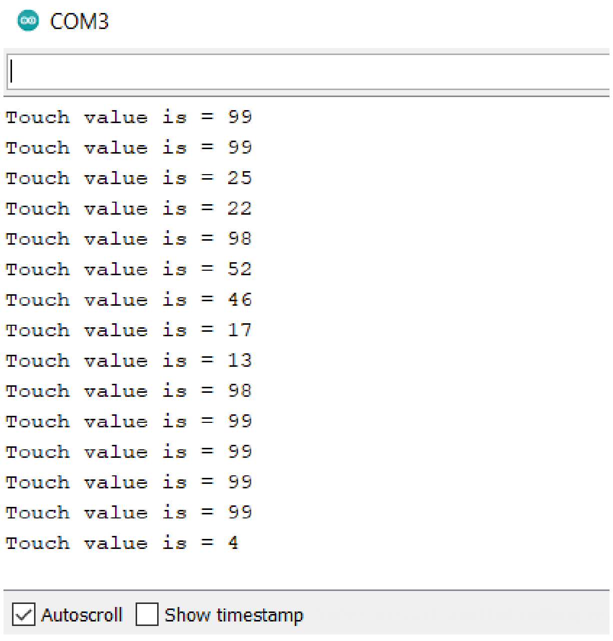


Figure 1: Figure showing **ESP32 Pinout**

Code :

```

// ESP32 Touch Test
int touch_value = 100;
void setup()
{
Serial.begin(115200);  //begin serial communication with BR 115200
pinMode (4, INPUT);   //gpio pin 4 input
pinMode(2, OUTPUT);   //2 output inbuilt led
digitalWrite (2, LOW);
}

void loop()
{
touch_value = touchRead(4);     //read from values from gpio4
Serial.print("Touch value is = ");
Serial.println( touch_value);   //print touch values

if (touch_value < 50)        //glow LED when touched
{digitalWrite (2, HIGH);}
else
{digitalWrite (2, LOW);}
delay(1000);
}
```

Output: Arduino IDE is used to compile code.



```
Touch value is = 99
Touch value is = 99
Touch value is = 25
Touch value is = 22
Touch value is = 98
Touch value is = 52
Touch value is = 46
Touch value is = 17
Touch value is = 13
Touch value is = 98
Touch value is = 99
Touch value is = 99
Touch value is = 99
Touch value is = 99
Touch value is = 4
```

Figure 2: Figure showing **Serial Port for touch sensor.**

Discussions & Observations: Fig (1) shows the pin-out diagram of ESP32. For details of ESP32 refer [Datasheet].ESP32 has 10 capacitive sensing touch sensors and 34 programmable GPIOs.These touch sensors can sense variations in anything that holds an electrical charge like the human skin. So, they can detect variations induced when touching the touch sensor with a finger.
In-built LED at port 2 was used . 'Touchread()' reads the value from the gpio. When touched, the touch value decreases. Inbuilt LED glows when touched.

Conclusion: We developed arduino code to interface ESP32 with touch sensor & LED.
The pinout diagram for ESP32 is shown in figure 1.

The output for touch sensor Serial port is shown in figure 2.
Hardware demonstration: [Video].

Experiment-2: LED Control using Bluetooth.

Aim: To explore Bluetooth features of ESP32.

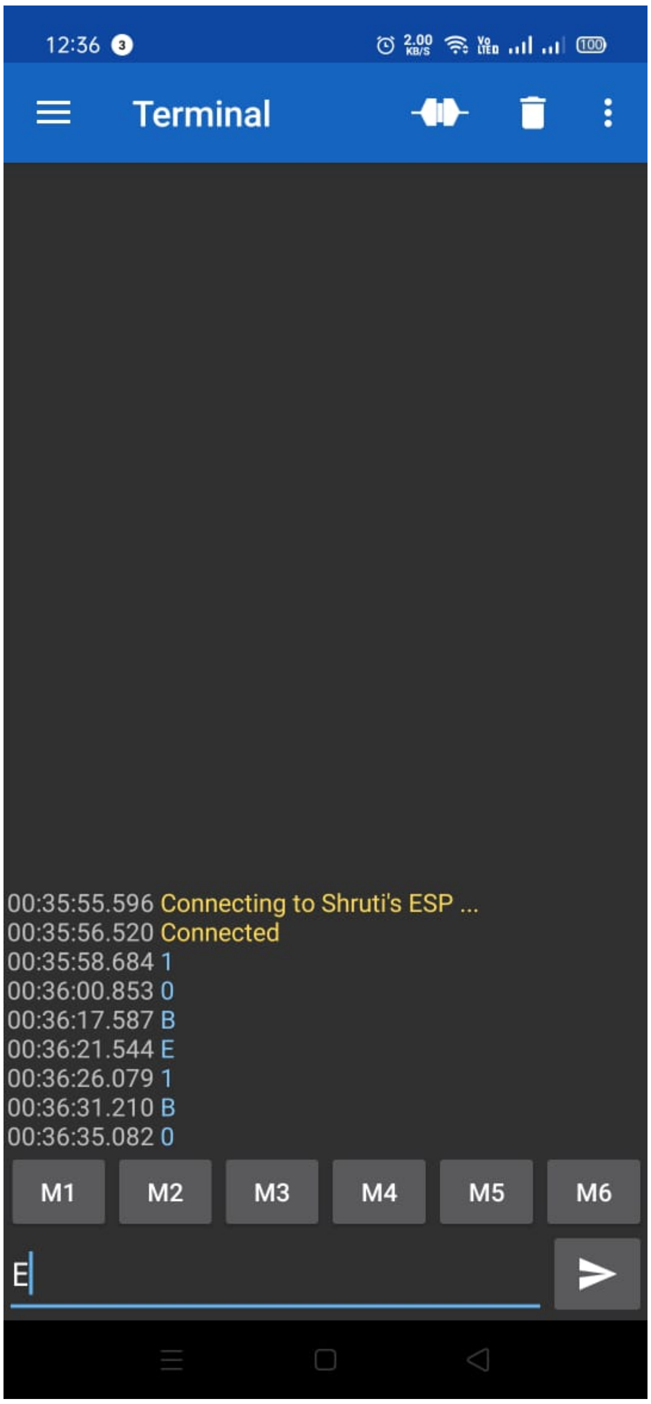
Problem Statement: To control and send information via Bluetooth using ESP32.

Code :

```
#include <BluetoothSerial.h>
BluetoothSerial BT;
char r;      //variable to read

void setup()
{
BT.begin("Shruti's ESP");    //device name to be displayed
pinMode( 2, OUTPUT);     //led as output
}

void loop()
{
  r = BT.read();     //read command sent via bluetooth
  if (r == '1')           //r=1 LED ON
  {
    digitalWrite( 2, HIGH);
  }
  if ( r == '0')          //r-0 LED OFF
  {
    digitalWrite( 2, LOW);
  }
  if(r == 'B')            //r=B LED BLINK
  {
    while(r != 'E')       //r=E END BLINKING
    {digitalWrite( 2,HIGH);
    delay(100);
    digitalWrite( 2,LOW);
    delay(100);
    r = BT.read();
    }
  }
}
```

Output: Arduino IDE is used to compile code.

Connection: Download "Serial Bluetooth Terminal" or any other Bluetooth app from the Google Playstore. Enable your phone's Bluetooth. Load the code on



```
00:35:55.596 Connecting to Shruti's ESP ...
00:35:56.520 Connected
00:35:58.684 1
00:36:00.853 0
00:36:17.587 B
00:36:21.544 E
00:36:26.079 1
00:36:31.210 B
00:36:35.082 0
```

Figure 3: Figure showing **Bluetooth Terminal.**

ESP32, and pair it with your phone. In this app, Connect to ”Shruti’s ESP“ & your ESP32 is connected.

Discussions & Observations: The ESP32 comes with Wi-Fi, Bluetooth Low Energy and Bluetooth Classic.Bluetooth Low Energy, BLE for short, is a power-conserving variant of Bluetooth. BLE’s primary application is short distance transmission of small amounts of data. Bluetooth Classic is much more simpler than Bluetooth Low Energy.
Input 1, 0, B & E make LED ON, OFF, Blink & End blink respectively. BluetoothSerial.read() which continuously reads on the port for either on/off/blink values.The BluetoothSerial.write(), transmits the on message to the phone on the terminal. Both can only read or write a character only. To transmit a message, the characters have to be transmitted one after another.

Conclusion: Hardware demonstration: [Video].

Experiment-3: LED Control using WiFi.

Aim: To initiate a local server to control and send information over WiFi using ESP32.

Problem Statement: To toggle LED via Web page which acts as Wifi Access Point using ESP32.

Code :

```
#include <WiFi.h>

const char* ssid = "Shruti-ESP";
const char* Password = "Shubh@123";

WiFiServer server(80);
String html ="<!DOCTYPE html>\
<html>\
<body>\
<form>\
<button name=\"LED\" button value=\"ON\" type=\"submit\">LED ...
    ON</button> \
<button name=\"LED\" button value=\"OFF\" type=\"submit\">LED ...
    OFF</button> \
</form> \
</body \
</html>";

void setup() {
  Serial.begin(115200);
  pinMode(2, OUTPUT);
  digitalWrite(2, LOW);
  WiFi.softAP(ssid, Password);
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP Address: ");
  Serial.print(IP);
  server.begin();
}

void loop() {
  WiFiClient client = server.available();
  if(client)
  {
    String request = client.readStringUntil('\r');
    if (request.indexOf("LED=ON")≥0)digitalWrite(2, HIGH);
    if (request.indexOf("LED=OFF")≥0)digitalWrite(2, LOW);
```

```
      client.print(html);
      request="";
    }
}
```

Output: Arduino IDE is used to compile code.

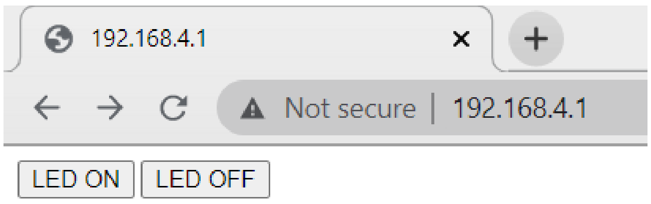


Figure 4: Figure showing **Webpage for LED Toggle via WiFi.**

Connection: After uploading above code on ESP32, connect mobile or laptop to the given network. An IP is assigned. When we go to that IP from any device on the local network, the HTML page we provided appears.

Discussions & Observations: In this experiment,ESP32 acts as local server.We define the server id and password in code.The function WiFi.softAPIP() returns the IP address through which the website can be accessed. The WiFi server is setup at port 80. A HTML form is defined in which 2 buttons are defined which send a get request to the server with the value LED=ON or LED=OFF depending on the button pressed. The request is received to ESP32 and inbuilt LED acts accordingly.

Conclusion: Hardware demonstration: [Video].

Experiment-4: Hall Sensor

Aim: To interface inbuilt hall sensor to ESP32.

Problem Statement: To glow LED in presence of magnet.

Code :

```
int val = 0;
void setup() {
  Serial.begin(9600);
  pinMode (2, OUTPUT);  // inBuilt LED output
}

void loop() {
  val = hallRead();     //get value from hallsensor
  if (val ≤0 || val≥50) digitalWrite (2, HIGH); //Led to ...
      indicate presence of magnet poles
  else digitalWrite (2, LOW);
  Serial.print("Hall sensor measurement = ");  // print the ...
      results to the serial monitor
  Serial.println(val);//to graph
   delay(500);
}
```

Output: Arduino IDE is used to compile code.

Discussions & Observations: ESP32 comes with in build hall sensor, this sensor can be used to detect presence of magnet, Like door sensor. In this tutorial we will see how to read its value and detect presence of magnet.

A Hall effect sensor is a transducer that varies its output voltage in response to a magnetic field. Hall effect sensors are used for proximity sensing, positioning, speed detection, and current sensing applications.

After uploading code, open serial plotter from tools menu, and move magnet near to ESP32 and observe readings also on board blue LED detects magnet poles. It is observed that hall sensor gives values lesser than 0 in presence of magnet and is greater than 50 for the opposite pole.

Conclusion: Hardware demonstration: [Video].

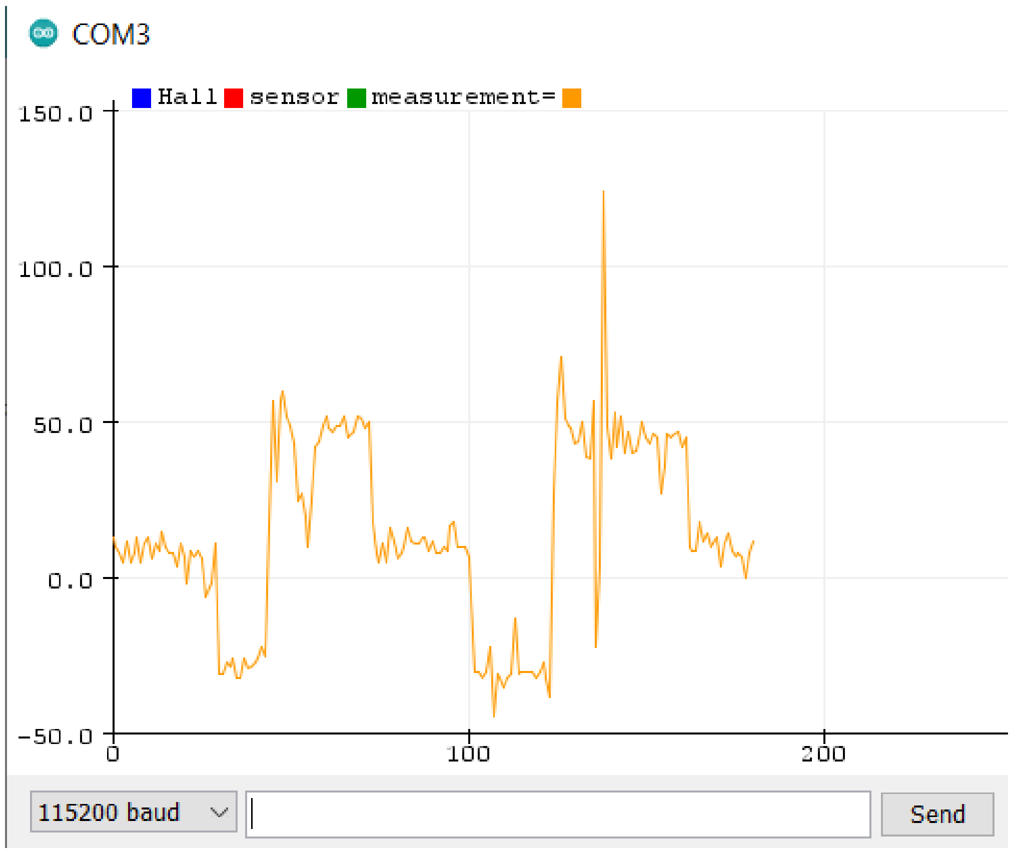


Figure 5: Figure showing **Serial Plotter**

Experiment-5: Temperature Sensor

Aim: To interface inbuilt temperature sensor to ESP32.

Problem Statement: To print temperature in Serial monitor using ESP32.

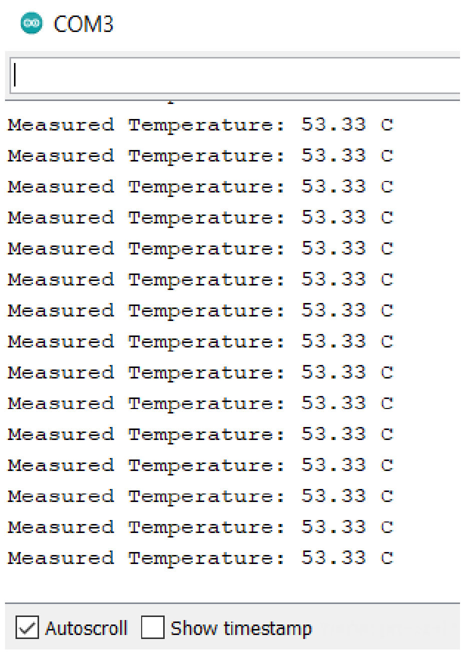
Code :

```
#ifdef __cplusplus
extern "C" {
#endif
uint8_t temprature_sens_read();
#ifdef __cplusplus
}
#endif
uint8_t temprature_sens_read();

void setup() {
Serial.begin(115200);
}

void loop() {
  Serial.print("Measured Temperature: ");
  // Convert raw temperature in F to Celsius degrees
  Serial.print((temprature_sens_read() - 32) / 1.8);
  Serial.println(" C");
  delay(100);
}
```

Output: Arduino IDE is used to compile code.

Discussions & Conclusion: It is observed the inbuilt temperature sensor gives constant value of 53.33 C(128F).
This is due to the fact that the temperature sensor is no longer included in the technical manual (since 06/2018). The fact that it always returns 128 indicates that it is obsolete.

Conclusion: Hardware demonstration: [Video].



```
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
Measured Temperature: 53.33 C
```

[✓] Autoscroll [ ] Show timestamp

Figure 6: Figure showing **Serial Monitor**

Experiment-6: Cloud API

Aim: To set up Cloud API using esp32.

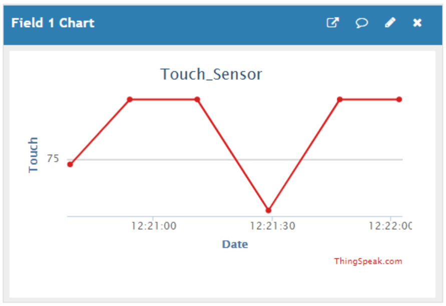
Problem Statement: To track data changes in the values of sensor like inbuilt touch sensor on an onine server(ThingSpeak).

Code :

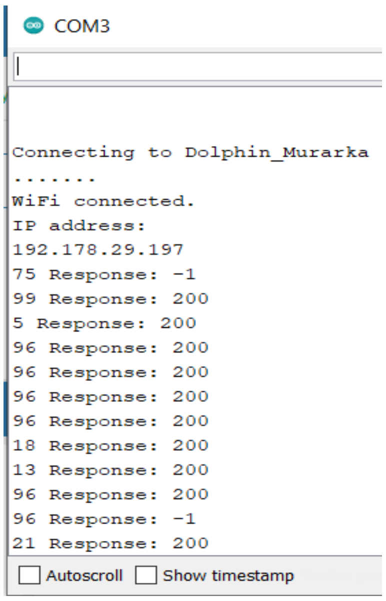
```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "Dolphin_Murarka";   //local network
const char* password = "$hubh3236";

const char* serverName = "https://api.thingspeak.com/update";
String apiKey = "CHXB4ACQ14EWY9NG";     //write key

WiFiServer server(80);
int value = 0;
void setup(){
    Serial.begin(115200);   //begin serial communication with br ...
       115200
    pinMode(2, OUTPUT);
    pinMode(4, INPUT);

    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password); // Connecting to a WiFi network

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected.");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}
void loop(){
  if(WiFi.status()== WL_CONNECTED){
   HTTPClient http;
   http.begin(serverName);
```

```
    int v = touchRead(4); //read touch sensor values
    if(v<20) digitalWrite(2, HIGH);   //led glows
    else digitalWrite(2, LOW);

    String DataSent = "api_key=" + apiKey + "&field1=" + String(v);
    int Response = http.POST(DataSent);   //datasent to Thingsspeak
    Serial.print(v);
    Serial.print(" Response: ");
    Serial.println(Response);
    http.end();
  }
}
```

Output: Arduino IDE is used to compile code.



(a) ThingsSpeak Graph.



```
Connecting to Dolphin_Murarka
.......
WiFi connected.
IP address:
192.178.29.197
75 Response: -1
99 Response: 200
5 Response: 200
96 Response: 200
96 Response: 200
96 Response: 200
96 Response: 200
18 Response: 200
13 Response: 200
96 Response: 200
96 Response: -1
21 Response: 200
```

(b) Serial Monitor

Figure 7: Connecting to Internet.

Discussions & Observations: We set up a channel in Mathwork's IoT platform ThingSpeak. Each channel has a unique Read and Write key.
We connect esp32 to the available WiFi instead of creating a local WiFi server and try to send a post request to the ThingSpeak server which receives the request and updates the graphs available on its server.

Conclusion: Hardware Demostration: [Video].

Experiment-7: IFTTT

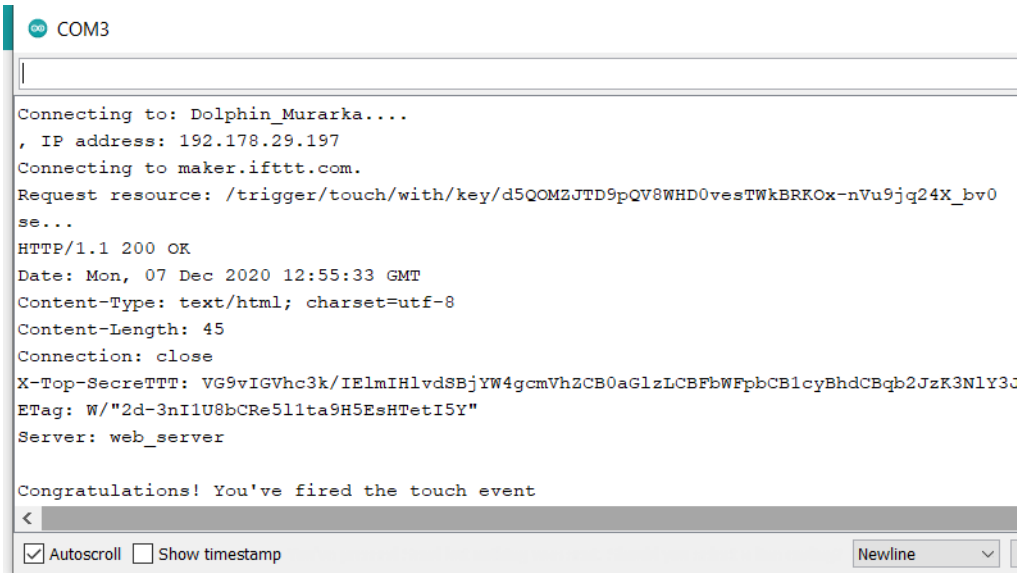
Aim: To connect Google sheets using If This Then That (IFTTT).

Problem Statement: To export and tabulate the data from the touch sensor(or any other sensor) **every 5 minutes** in Google sheets using If This Then That & ESP32.

Code :

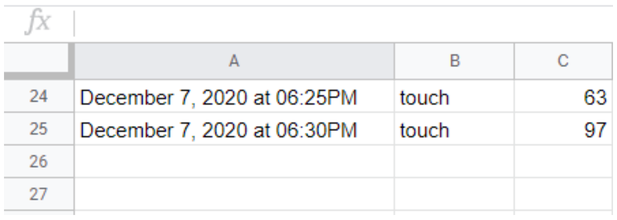
```
#include <WiFi.h>

const char* ssid     = "Dolphin_Murarka";
const char* password = "$hubh3236";

const char* resource = ...
    "/trigger/touch/with/key/d5Q0MZJTD9pQV8WHD0vesTWkBRK0x-nVu9jq24X_bv0";
const char* server = "maker.ifttt.com";

int v =100;
void setup()
{
  Serial.begin(115200);
  delay(2000);
  pinMode(2,INPUT);
  initWifi();
}

void loop()
{
  sendtoIFTTT();
  delay(300000);        //Data entry after 5 minutes.
}

// Establish a Wi-Fi connection with your router
void initWifi() {
  Serial.print("Connecting to: ");
  Serial.print(ssid);
  WiFi.begin(ssid, password);
  int timeout = 10 * 4; // 10 seconds
  while(WiFi.status() != WL_CONNECTED  && (timeout-- > 0)) {
    delay(250);
    Serial.print(".");
  }
  Serial.println("");
  if(WiFi.status() != WL_CONNECTED) {
     Serial.println("Failed to connect, going back to sleep");
  }
  Serial.print(", IP address: ");
  Serial.println(WiFi.localIP());
}

// Make an HTTP request to the IFTTT web service
void sendtoIFTTT() {
  Serial.print("Connecting to ");
  Serial.print(server);

  WiFiClient client;
  int retries = 5;
  while(!!!client.connect(server, 80) && (retries-- > 0)) {
    Serial.print(".");
  }
  Serial.println();
  if(!!!client.connected()) {
    Serial.println("Failed to connect...");
  }

  Serial.print("Request resource: ");
  Serial.println(resource);
String jsonObject = String("{\"value1\":\"") + touchRead(4) + "\"}";


  client.println(String("POST ") + resource + " HTTP/1.1");
  client.println(String("Host: ") + server);
  client.println("Connection: close\r\nContent-Type: ...
     application/json");
  client.print("Content-Length: ");
  client.println(jsonObject.length());
  client.println();
  client.println(jsonObject);
  Serial.println("len: " + jsonObject.length());
  int timeout = 5 * 10; // 5 seconds
  while(!!!client.available() && (timeout-- > 0)){
    delay(1000);
  }
  if(!!!client.available()) {
    Serial.println("No response...");
  }
  while(client.available()){
    Serial.write(client.read());
  }
  Serial.println("\nclosing connection");
//  .client.stop();
}
```

Output: Arduino IDE is used to compile code.



```
Connecting to: Dolphin_Murarka....
, IP address: 192.178.29.197
Connecting to maker.ifttt.com.
Request resource: /trigger/touch/with/key/d5QOMZJTD9pQV8WHD0vesTWkBRKOx-nVu9jq24X_bv0
se...
HTTP/1.1 200 OK
Date: Mon, 07 Dec 2020 12:55:33 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 45
Connection: close
X-Top-SecreTTT: VG9vIGVhc3k/IElmIHlvdSBjYW4gcmVhZCB0aGlzLCBFbWFpbCB1cyBhdCBqb2JzK3NlY3
ETag: W/"2d-3nI1U8bCRe5l1ta9H5EsHTetI5Y"
Server: web_server

Congratulations! You've fired the touch event
```

(a) Serial Monitor.



	A	B	C
24	December 7, 2020 at 06:25PM	touch	63
25	December 7, 2020 at 06:30PM	touch	97
26			
27			

(b) Data entry in Google Sheets after every 5 mins

Figure 8: IFTTT & Google Sheets.

Discussions & Observations: The WiFi initialisation process through a function initWifi(). The Post request is sent to the IFTTT server which is defined in the function sendtoIFTTT(). Firstly We check the connection. After that, we write to the client a POST request in the json format with the data that we want to send.

The code will send the post request to webhooks which is connected to IFTTT. The IFTTT applet is connected to Google Sheets and hence when the post request is sent with the value, this value will get appended to the google sheets. To run this every 5 minutes we can set the function in the loop with delay.

Conclusion: Hardware Demostration: [Video].