

IoT Based Early Flood Detection and Avoidance

Bharata Sai Dhanush - 180102016

November 2020

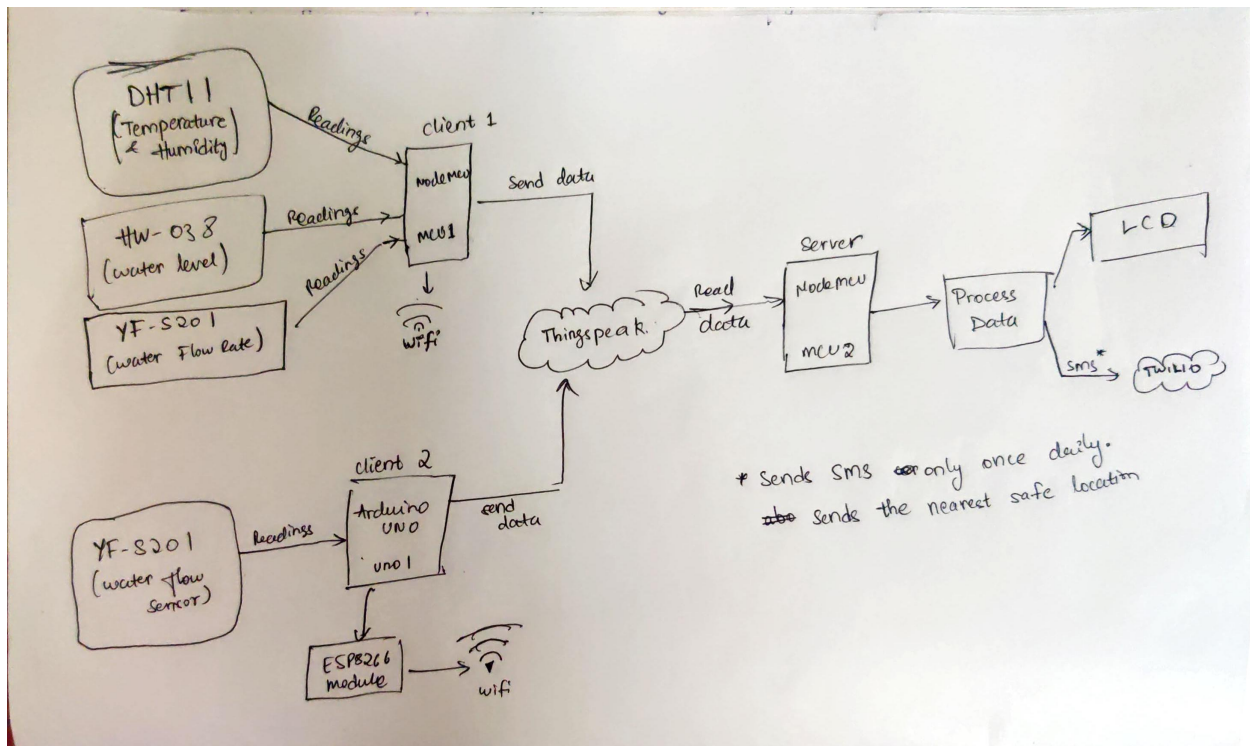
1 Main Objective

The main objective of this project is to build a simple IoT based system to detect the possibility of flood in its early stage by monitoring the conditions at the place so that necessary precautions can be taken.

2 Implemented Attributes

- DHT11 sensor to get the values of temperature and humidity.
- Water Level sensor and water flow sensor, using which we can develop a rough profile of the situation of water at that location.
- LCD display to display which locations are safe.
- Used ThingSpeak cloud platform to display the readings for monitoring.
- Used Twilio as a cloud service to send SMS.

3 Configuration Diagram



4 Sample Outputs

This is the sample output of the Serial monitor for the server which shows data from both the locations.

```
For location 1:  
Temperature: 28.00°C  
Humidity: 40.00%  
Water Flow Rate: 0.00L/min  
Water Level: 0.00
```

```
For Location 2:  
Water Flow Rate: 0.00L/min
```

```
For location 1:  
Temperature: 28.00°C  
Humidity: 40.00%  
Water Flow Rate: 0.00L/min  
Water Level: 0.00
```

```
For Location 2:  
Water Flow Rate: 0.00L/min
```

```
For location 1:  
Temperature: 28.00°C  
Humidity: 40.00%  
Water Flow Rate: 0.00L/min  
Water Level: 0.00
```

```
For Location 2:  
Water Flow Rate: 0.00L/min
```

5 Codes

The codes are attached with the document.

Client 1 (NodeMCU) located at Location 1:

```
1 #include <DHT.h> // Including library for dht
2 // #include <LiquidCrystal_I2C.h>
3 #include <ESP8266WiFi.h>
4 #define DHTPIN 0 //pin where the dht11 is connected
5
6 String apiKey = "WH44RPOBWB81KXHN"; // Enter your Write API key from ThingSpeak
7 const char ssid[] = "No free wifi for you :P"; // replace with your wifi ssid and wpa2 key
8 const char pass[] = "srini@123456789";
9 const char server[] = "api.thingspeak.com";
10 //for flow sensor
11 #define SENSOR 12
12
13 long currentMillis = 0;
14 long previousMillis = 0;
15 int interval = 1000;
16 // boolean ledState = LOW;
17 float calibrationFactor = 4.5;
18 volatile byte pulseCount;
19 byte pulse1Sec = 0;
20 float flowRate;
21 unsigned long flowMilliLitres;
22 unsigned int totalMilliLitres;
23 float flowLitres;
24 float totalLitres;
25
26 DHT dht(DHTPIN, DHT11);
27 // LiquidCrystal_I2C lcd(0x27, 16, 2);
28 //for waterlevel
29 #define sensorPower 13
30 #define sensorPin A0
31 int val = 0;
32
33 WiFiClient client;
34 void IRAM_ATTR pulseCounter()
35 {
36     pulseCount++;
37 }
38 void setup()
39 {
40
41     Serial.begin(115200);
42     // lcd.clear();
43     // lcd.begin();
44     // lcd.print("Initializing");
45     Serial.println("Initializing");
46     delay(10);
47     dht.begin();
48
49     Serial.print("Connecting to ");
50     Serial.println(ssid);
51
52     WiFi.begin(ssid, pass);
53
54     while (WiFi.status() != WL_CONNECTED)
55     {
56         delay(500);
57         Serial.print(".");
58     }
59     Serial.println("");
60     Serial.println("WiFi connected");
61
62     //for level sensor
```

```

63  pinMode(sensorPower, OUTPUT);
64  digitalWrite(sensorPower, LOW);
65
66  //for flow SENSOR
67  pinMode(SENSOR, INPUT_PULLUP);
68  pulseCount = 0;
69  flowRate = 0.0;
70  flowMilliLitres = 0;
71  totalMilliLitres = 0;
72  previousMillis = 0;
73  attachInterrupt(digitalPinToInterrupt(SENSOR), pulseCounter, FALLING);
74 }
75
76 void loop()
77 {
78
79  float h = dht.readHumidity();
80  float t = dht.readTemperature();
81
82  if (isnan(h) || isnan(t))
83  {
84      // Serial.println("Failed to read from DHT sensor!");
85      dht.begin();
86      // lcd.clear();
87      delay(1000);
88  }
89  int level = readSensor();
90
91  // lcd.setCursor(0, 0);
92  // lcd.print("temp: ");
93  // lcd.print(t);
94  // lcd.setCursor(0, 1);
95  // lcd.print("humidity: ");
96  // lcd.print(h);
97
98  //for flow rate
99  pulse1Sec = pulseCount;
100 pulseCount = 0;
101 flowRate = ((1000.0 / (millis() - previousMillis)) * pulse1Sec) / calibrationFactor;
102 previousMillis = millis();
103 flowMilliLitres = (flowRate / 60) * 1000;
104 flowLitres = (flowRate / 60);
105 totalMilliLitres += flowMilliLitres;
106 totalLitres += flowLitres;
107
108 if (client.connect(server, 80)) // "184.106.153.149" or api.thingspeak.com
109 {
110
111     String postStr = apiKey;
112     postStr += "&field1=";
113     postStr += String(t);
114     postStr += "&field2=";
115     postStr += String(h);
116     postStr += "&field3=";
117     postStr += String(flowRate);
118     postStr += "&field4=";
119     postStr += String(level);
120     postStr += "\r\n\r\n";
121
122     client.print("POST /update HTTP/1.1\n");
123     client.print("Host: api.thingspeak.com\n");
124     client.print("Connection: close\n");
125     client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
126     client.print("Content-Type: application/x-www-form-urlencoded\n");
127     client.print("Content-Length: ");
128     client.print(postStr.length());
129     client.print("\n\n");
130     client.print(postStr);

```

```

131     Serial.print("Temperature: ");
132     Serial.print(t);
133     Serial.print(" degrees Celcius, Humidity: ");
134     Serial.print(h);
135     Serial.println("%, Flow rate: ");
136     Serial.print(flowRate);
137     Serial.print("L/min, Water Level: ");
138     Serial.println(level);
139 }
140 client.stop();
141
142 Serial.println("Waiting...");
143
144 // thingspeak needs minimum 15 sec delay between updates, i've set it to 30 seconds
145 delay(1000);
146 }
147
148 int readSensor()
149 {
150     digitalWrite(sensorPower, HIGH); // Turn the sensor ON
151     delay(100);                      // wait 10 milliseconds
152     val = analogRead(sensorPin);     // Read the analog value form sensor
153     digitalWrite(sensorPower, LOW);  // Turn the sensor OFF
154     // return val;                   // send current reading
155     if (val < 100)
156         return 0;
157     if (val < 225)
158         return 1;
159     if (val < 250)
160         return 2;
161     else
162         return 3;
163 }
164 }

```

Client 2 (Uno) located at Location 2:

```

1 #include <SoftwareSerial.h>
2 #include <SPI.h>
3 #include <Wire.h>
4 #define RX 10
5 #define TX 11
6 String AP = "No free wifi for you :P"; // CHANGE ME
7 String PASS = "srini@123456789";      // CHANGE ME
8 String API = "08IV31YQWXRORViW";     // CHANGE ME
9 String HOST = "api.thingspeak.com";
10 String PORT = "80";
11 String field = "field1";
12 int countTrueCommand;
13 int countTimeCommand;
14 boolean found = false;
15 float valSensor = 1;
16 int p = 0;
17 SoftwareSerial esp8266(RX, TX);
18
19 #define SENSOR 2
20
21 long currentMillis = 0;
22 long previousMillis = 0;
23 int interval = 1000;
24 boolean ledState = LOW;
25 float calibrationFactor = 4.5;
26 volatile byte pulseCount;
27 byte pulse1Sec = 0;
28 float flowRate;
29 unsigned long flowMilliLitres;
30 unsigned int totalMilliLitres;

```

```

31 float flowLitres;
32 float totalLitres;
33 void pulseCounter()
34 {
35     pulseCount++;
36 }
37 void setup()
38 {
39     Serial.begin(9600);
40     esp8266.begin(115200);
41     pinMode(SENSOR, INPUT_PULLUP);
42     pulseCount = 0;
43     flowRate = 0.0;
44     flowMilliLitres = 0;
45     totalMilliLitres = 0;
46     previousMillis = 0;
47     attachInterrupt(digitalPinToInterrupt(SENSOR), pulseCounter, FALLING);
48     sendCommand("AT", 5, "OK");
49     sendCommand("AT+CWMODE=3", 5, "OK");
50     sendCommand("AT+CWJAP=\"" + AP + "\",\"" + PASS + "\", 20, \"OK\");
51 }
52 void loop()
53 {
54     currentMillis = millis();
55     if (currentMillis - previousMillis > interval)
56     {
57
58         pulse1Sec = pulseCount;
59         pulseCount = 0;
60         flowRate = ((1000.0 / (millis() - previousMillis)) * pulse1Sec) / calibrationFactor;
61         previousMillis = millis();
62         flowMilliLitres = (flowRate / 60) * 1000;
63         flowLitres = (flowRate / 60);
64         totalMilliLitres += flowMilliLitres;
65         totalLitres += flowLitres;
66         Serial.print("Flow rate: ");
67         Serial.print(float(flowRate)); // Print the integer part of the variable
68         Serial.print("L/min");
69         Serial.println();
70     }
71
72     valSensor = flowRate;
73     Serial.print("valSensor ");
74     Serial.println(valSensor);
75     String getData = "GET /update?api_key=" + API + "&" + field + "=" + String(valSensor);
76     sendCommand("AT+CIPMUX=1", 5, "OK");
77     sendCommand("AT+CIPSTART=0,\"TCP\", \"" + HOST + "\", " + PORT, 15, "OK");
78     sendCommand("AT+CIPSEND=0," + String(getData.length() + 4), 4, ">");
79     esp8266.println(getData);
80     delay(1500);
81     countTrueCommand++;
82     sendCommand("AT+CIPCLOSE=0", 5, "OK");
83 }
84
85 void sendCommand(String command, int maxTime, char readReplay[])
86 {
87     Serial.print(countTrueCommand);
88     Serial.print(". at command => ");
89     Serial.print(command);
90     Serial.print(" ");
91     while (countTimeCommand < (maxTime * 1))
92     {
93         esp8266.println(command); //at+cipsend
94         if (esp8266.find(readReplay)) //ok
95         {
96             found = true;
97             break;
98         }

```

```

99
100     countTimeCommand++;
101 }
102
103 if (found == true)
104 {
105     Serial.println("OYI");
106     countTrueCommand++;
107     countTimeCommand = 0;
108 }
109
110 if (found == false)
111 {
112     Serial.println("Fail");
113     countTrueCommand = 0;
114     countTimeCommand = 0;
115 }
116
117 found = false;
118 }

```

Server (NodeMCU)

```

1 #include <ThingSpeak.h>
2 #include <ESP8266WiFi.h>
3 #include <LiquidCrystal_I2C.h>
4 #include <base64.h>
5 LiquidCrystal_I2C lcd(0x27, 16, 2);
6 const char ssid[] = "No free wifi for you :P";
7 const char pass[] = "srini@123456789";
8 WiFiClient client;
9
10 //-----Channel Details-----//
11 unsigned long interval = 86400000, prevSent2 = 0, prevSent1 = 0;
12 unsigned long counterChannelNumber1 = 1209982;
13 unsigned long counterChannelNumber2 = 1217400; // Channel ID
14 const char *myCounterReadAPIKey = "98D2M11Y3EXC3IZJ"; // Read API Key
15 const char *myCounterReadAPIKey1 = "V6CPNBLFP7DZUWW7"; // Read API Key
16 const int FieldNumber1 = 1; // The field you wish to read
17 const int FieldNumber2 = 2;
18 const int FieldNumber3 = 3;
19 const int FieldNumber4 = 4;
20 bool safe1 = true;
21 bool safe2 = true;
22 //nearby numbers
23 String nearby = "9502215191";
24 //-----//
25 const char *account_sid = "AC05464dcb1ee0cf3258172f62eeffd0aa";
26 const char *auth_token = "1de03482e67b470032c297801d658c4c";
27 String from_number = "+18312176586";
28 String to_number = "+919502215191";
29 String message_body1 = "Alert: Go to 2";
30 String message_body2 = "Alert: Go to 1";
31 const char fingerprint[] = "BC B0 1A 32 80 5D E6 E4 A2 29 66 2B 08 C8 E0 4C 45 29 3F D0";
32 String urlencode(String str)
33 {
34     String encodedString = "";
35     char c;
36     char code0;
37     char code1;
38     char code2;
39     for (int i = 0; i < str.length(); i++)
40     {
41         c = str.charAt(i);
42         if (c == ' ')
43         {
44             encodedString += '+';

```

```

45     }
46     else if (isalnum(c))
47     {
48         encodedString += c;
49     }
50     else
51     {
52         code1 = (c & 0xf) + '0';
53         if ((c & 0xf) > 9)
54         {
55             code1 = (c & 0xf) - 10 + 'A';
56         }
57         c = (c >> 4) & 0xf;
58         code0 = c + '0';
59         if (c > 9)
60         {
61             code0 = c - 10 + 'A';
62         }
63         code2 = '\\0';
64         encodedString += '%';
65         encodedString += code0;
66         encodedString += code1;
67     }
68     yield();
69 }
70 return encodedString;
71 }
72
73 String get_auth_header(const String &user, const String &password)
74 {
75     size_t toencodeLen = user.length() + password.length() + 2;
76     char toencode[toencodeLen];
77     memset(toencode, 0, toencodeLen);
78     snprintf(toencode, toencodeLen, "%s:%s", user.c_str(), password.c_str());
79     String encoded = base64::encode((uint8_t *)toencode, toencodeLen - 1);
80     String encoded_string = String(encoded);
81     std::string::size_type i = 0;
82     // Strip newlines (after every 72 characters in spec)
83     while (i < encoded_string.length())
84     {
85         i = encoded_string.indexOf('\\n', i);
86         if (i == -1)
87         {
88             break;
89         }
90         encoded_string.remove(i, 1);
91     }
92     return "Authorization: Basic " + encoded_string;
93 }
94
95 void setup()
96 {
97     Serial.begin(115200);
98     WiFi.mode(WIFI_STA);
99     ThingSpeak.begin(client);
100     lcd.begin();
101     lcd.clear();
102     lcd.print("Initializing...");
103 }
104
105 void loop()
106 {
107
108     Serial.println();
109     //----- Network -----//
110     if (WiFi.status() != WL_CONNECTED)
111     {
112         Serial.print("Connecting to ");

```



```

113     Serial.print(ssid);
114     Serial.println(" ...");
115     while (WiFi.status() != WL_CONNECTED)
116     {
117         WiFi.begin(ssid, pass);
118         delay(5000);
119     }
120     Serial.println("Connected to Wi-Fi Succesfully.");
121 }
122 //----- End of Network connection-----//
123 Serial.println("For location 1:");
124 //----- Channel 1 -----//
125 float temp = ThingSpeak.readLongField(counterChannelNumber1, FieldNumber1,
126 myCounterReadAPIKey);
127 int statusCode = ThingSpeak.getLastReadStatus();
128 if (statusCode == 200)
129 {
130     Serial.print("Temperature: ");
131     Serial.print(temp);
132     Serial.println(" C ");
133 }
134 else
135 {
136     Serial.println("Unable to read channel / No internet connection");
137 }
138 delay(100);
139 //----- End of Channel 1 -----//
140 //----- Channel 2 -----//
141 float humidity = ThingSpeak.readLongField(counterChannelNumber1, FieldNumber2,
142 myCounterReadAPIKey);
143 statusCode = ThingSpeak.getLastReadStatus();
144 if (statusCode == 200)
145 {
146     Serial.print("Humidity: ");
147     Serial.print(humidity);
148     Serial.println("%");
149 }
150 else
151 {
152     Serial.println("Unable to read channel / No internet connection");
153 }
154 delay(100);
155 //----- End of Channel 2 -----//
156 float flowRate1 = ThingSpeak.readLongField(counterChannelNumber1, FieldNumber3,
157 myCounterReadAPIKey);
158 statusCode = ThingSpeak.getLastReadStatus();
159 if (statusCode == 200)
160 {
161     Serial.print("Water Flow Rate: ");
162     Serial.print(flowRate1);
163     Serial.println("L/min");
164 }
165 else
166 {
167     Serial.println("Unable to read channel / No internet connection");
168 }
169 delay(100);
170 float waterLevel = ThingSpeak.readLongField(counterChannelNumber1, FieldNumber4,
171 myCounterReadAPIKey);
172 statusCode = ThingSpeak.getLastReadStatus();
173 if (statusCode == 200)
174 {
175     Serial.print("Water Level: ");
176     Serial.println(waterLevel);
177 }

```

```

177     else
178     {
179         Serial.println("Unable to read channel / No internet connection");
180     }
181     delay(100);
182     Serial.println("\nFor Location 2:");
183     float flowRate2 = ThingSpeak.readLongField(counterChannelNumber2, FieldNumber1,
184     myCounterReadAPIKey1);
185     statusCode = ThingSpeak.getLastReadStatus();
186     if (statusCode == 200)
187     {
188         Serial.print("Water Flow Rate: ");
189         Serial.print(flowRate2);
190         Serial.println("L/min");
191     }
192     else
193     {
194         Serial.println("Unable to read channel / No internet connection");
195     }
196     delay(100);
197     //temp,humidity,flowRate1,waterLevel,flowRate2
198
199     if (waterLevel >= 2 || flowRate1 >= 10)
200     {
201         if (millis() - prevSent1 > interval || safe1)
202         {
203             sendsms(message_body1);
204             prevSent1 = millis();
205         }
206         safe1 = false;
207     }
208     else
209     {
210         safe1 = true;
211     }
212     if (flowRate2 >= 10)
213     {
214         if (millis() - prevSent2 > interval || safe2)
215         {
216             sendsms(message_body2);
217             prevSent2 = millis();
218         }
219         safe2 = false;
220     }
221     else
222     {
223         safe2 = true;
224     }
225     lcd.clear();
226     if (safe1)
227     {
228         lcd.setCursor(0, 0);
229         lcd.print("Place-1:SAFE");
230     }
231     else
232     {
233         lcd.setCursor(0, 0);
234         lcd.print("Place-1:NOT SAFE");
235     }
236     if (safe2)
237     {
238         lcd.setCursor(0, 1);
239         lcd.print("Place-2:SAFE");
240     }
241     else
242     {
243         lcd.setCursor(0, 1);
244         lcd.print("Place-2:NOT SAFE");
245     }
246 }

```

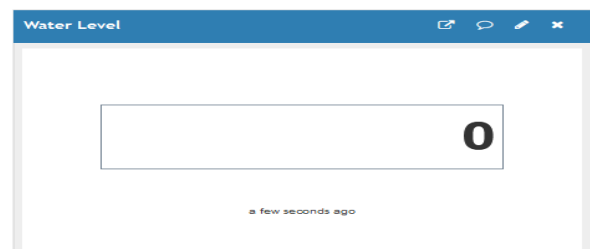
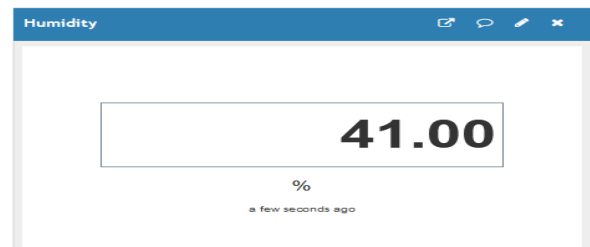
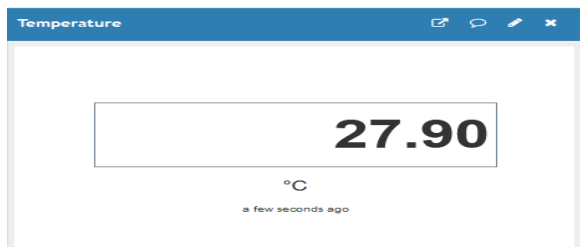
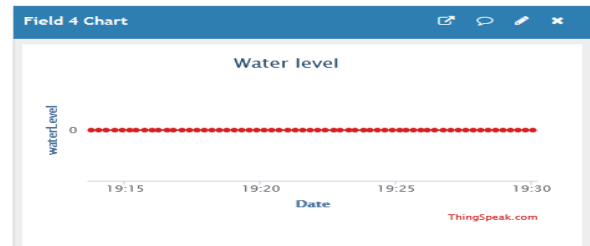
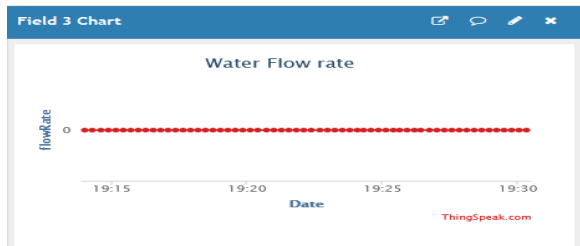
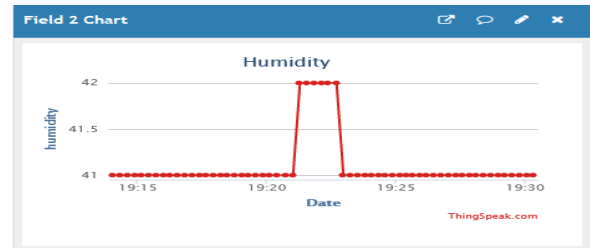
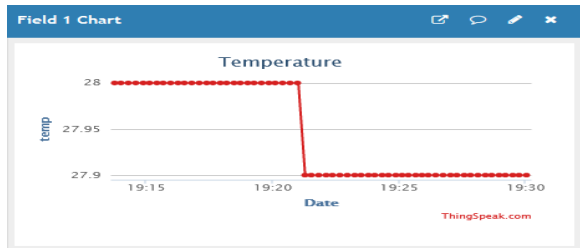
```

244 void sendsms(String message_body)
245 {
246     WiFiClientSecure client;
247     client.setFingerprint(fingerprint);
248     Serial.printf("+ Using fingerprint '%s'\n", fingerprint);
249     const char *host = "api.twilio.com";
250     const int httpsPort = 443;
251     Serial.print("+ Connecting to ");
252     Serial.println(host);
253     if (!client.connect(host, httpsPort))
254     {
255         Serial.println("- Connection failed.");
256         return; // Skips to loop();
257     }
258     Serial.println("+ Connected.");
259     Serial.println("+ Post an HTTP send SMS request.");
260     String post_data = "To=" + urlencode(to_number) + "&From=" + urlencode(from_number) + "&
Body=" + urlencode(message_body);
261     String auth_header = get_auth_header(account_sid, auth_token);
262     String http_request = "POST /2010-04-01/Accounts/" + String(account_sid) + "/Messages
HTTP/1.1\r\n" + auth_header + "\r\n" + "Host: " + host + "\r\n" + "Cache-control: no-
cache\r\n" + "User-Agent: ESP8266 Twilio Example\r\n" + "Content-Type: application/x-www
-form-urlencoded\r\n" + "Content-Length: " + post_data.length() + "\r\n" + "Connection:
close\r\n" + "\r\n" + post_data + "\r\n";
263     client.println(http_request);
264     // Read the response.
265     String response = "";
266     while (client.connected())
267     {
268         String line = client.readStringUntil('\n');
269         response += (line);
270         response += ("\r\n");
271     }
272     Serial.println("+ Connection is closed.");
273     Serial.println("+ Response:");
274     Serial.println(response);
275 };

```

6 User Manual

- Power the two clients (one is nodeMCU and other is Arduino uno) and one server(NodeMCU).
- Place these in areas where Wifi signal is not weak.
- You should start seeing all the stats in your ThingSpeak app as well as in the server's serial monitor.
- You can monitor the data using ThingSpeak (attached below is for Location 1)



7 Demo

The video is in the drive link

<https://drive.google.com/file/d/15BfMhqgfqN4mu5fjoA7pQUWaWM5qfLFg/view?usp=sharing>