IoT Based Early Flood Detection and Avoidance

Bharata Sai Dhanush - 180102016

November 2020

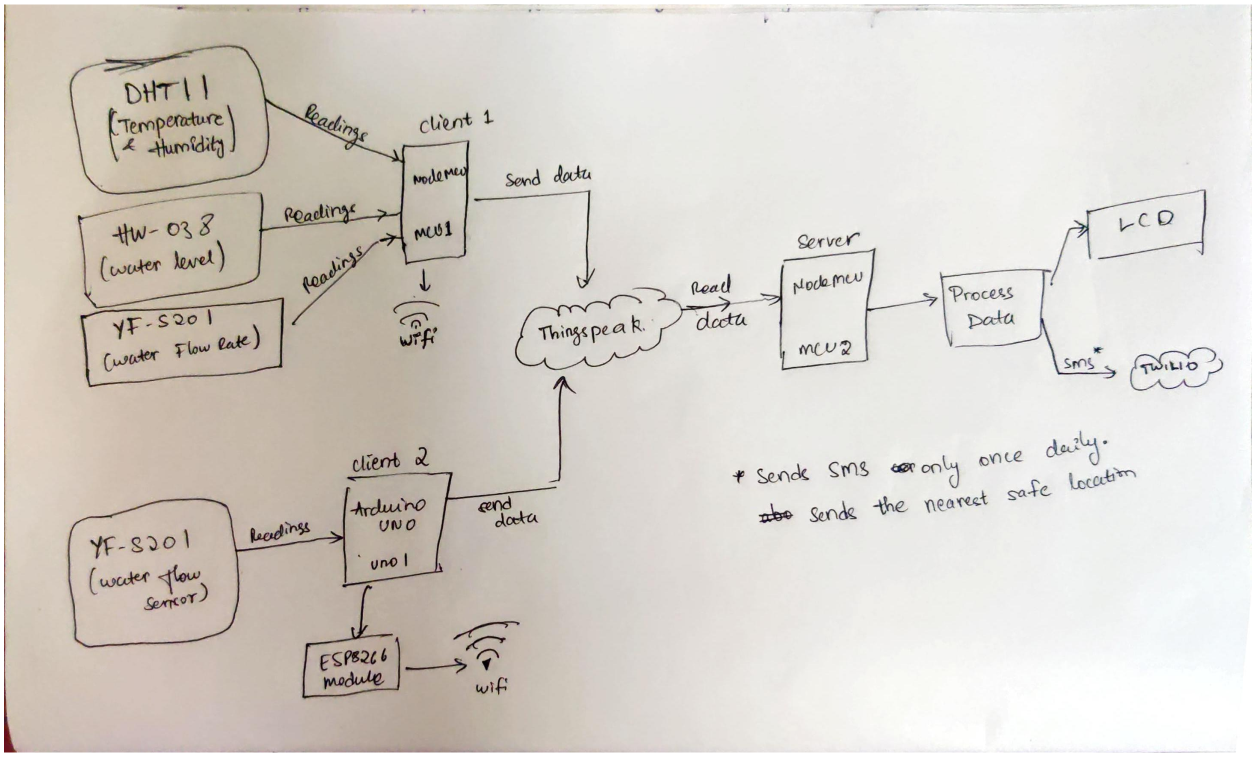
1 Main Objective

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

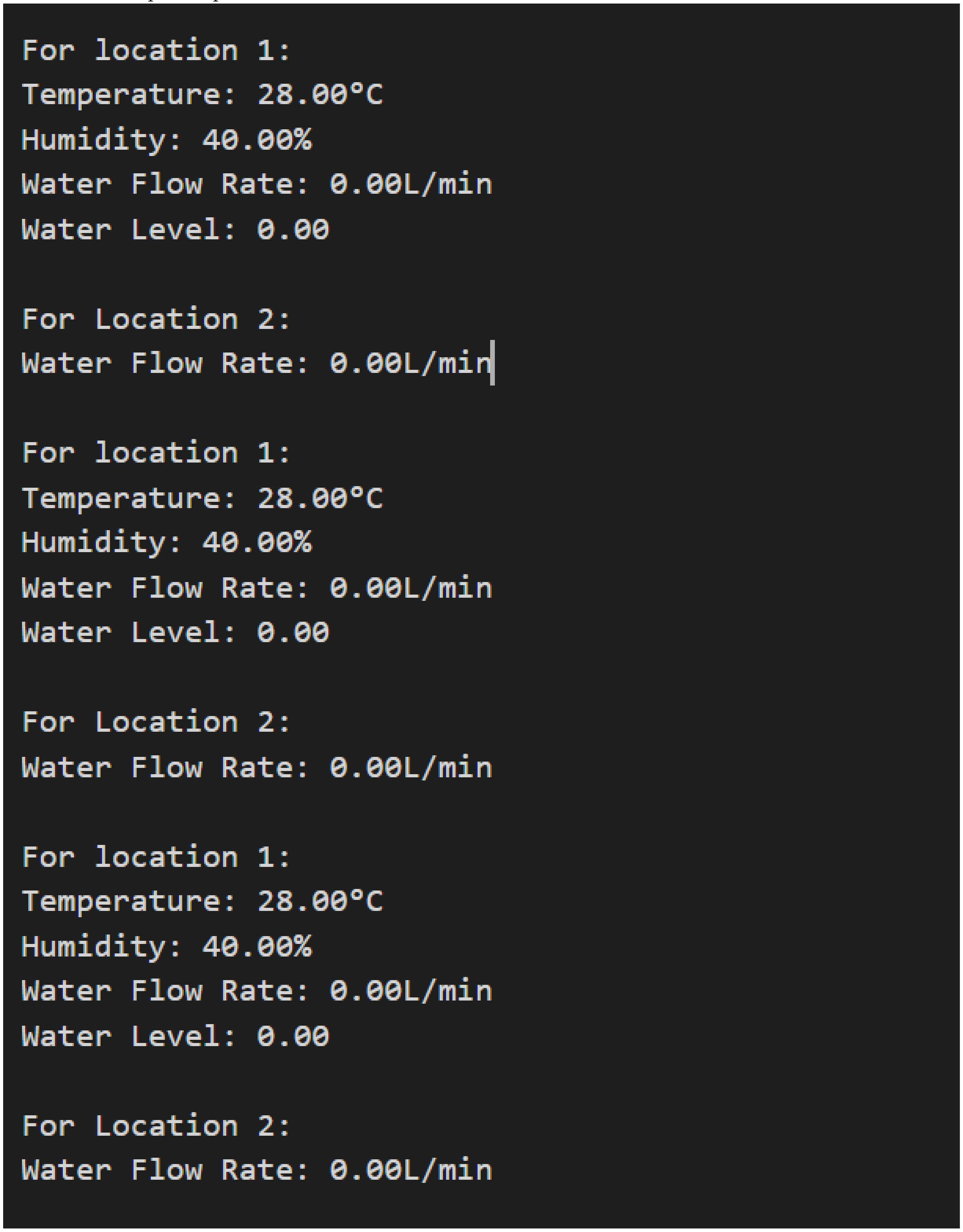
- DHT11 sensor to get the values of temperature and humidity.
- Water Level sensor and water flow sensor, using which we can develop a rough profile of the situation of water at that location.
- LCD display to display which locations are safe.
- Used ThingSpeak cloud platform to display the readings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

```
#include <DHT.h> // Including library for dht
// #include <LiquidCrystal_I2C.h>
#include <ESP8266WiFi.h>
#define DHTPIN 0 //pin where the dht11 is connected

String apiKey = "WH44RPOBWB81KXHN";             //  Enter your Write API key from ThingSpeak
const char ssid[] = "No free wifi for you :P"; // replace with your wifi ssid and wpa2 key
const char pass[] = "srini@123456789";
const char server[] = "api.thingspeak.com";
//for flow sensor
#define SENSOR 12

long currentMillis = 0;
long previousMillis = 0;
int interval = 1000;
// boolean ledState = LOW;
float calibrationFactor = 4.5;
volatile byte pulseCount;
byte pulse1Sec = 0;
float flowRate;
unsigned long flowMilliLitres;
unsigned int totalMilliLitres;
float flowLitres;
float totalLitres;

DHT dht(DHTPIN, DHT11);
// LiquidCrystal_I2C lcd(0x27, 16, 2);
//for waterlevel
#define sensorPower 13
#define sensorPin A0
int val = 0;

WiFiClient client;
void IRAM_ATTR pulseCounter()
{
  pulseCount++;
}
void setup()
{

  Serial.begin(115200);
  // lcd.clear();
  // lcd.begin();
  // lcd.print("Initializing");
  Serial.println("Initializing");
  delay(10);
  dht.begin();

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

  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  //for level sensor
  pinMode(sensorPower, OUTPUT);
  digitalWrite(sensorPower, LOW);

  //for flow SENSOR
  pinMode(SENSOR, INPUT_PULLUP);
  pulseCount = 0;
  flowRate = 0.0;
  flowMilliLitres = 0;
  totalMilliLitres = 0;
  previousMillis = 0;
  attachInterrupt(digitalPinToInterrupt(SENSOR), pulseCounter, FALLING);
}

void loop()
{

  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t))
  {
    // Serial.println("Failed to read from DHT sensor!");
    dht.begin();
    // lcd.clear();
    delay(1000);
  }
  int level = readSensor();

  // lcd.setCursor(0, 0);
  // lcd.print("temp: ");
  // lcd.print(t);
  // lcd.setCursor(0, 1);
  // lcd.print("humidity: ");
  // lcd.print(h);

  //for flow rate
  pulse1Sec = pulseCount;
  pulseCount = 0;
  flowRate = ((1000.0 / (millis() - previousMillis)) * pulse1Sec) / calibrationFactor;
  previousMillis = millis();
  flowMilliLitres = (flowRate / 60) * 1000;
  flowLitres = (flowRate / 60);
  totalMilliLitres += flowMilliLitres;
  totalLitres += flowLitres;

  if (client.connect(server, 80)) //    "184.106.153.149" or api.thingspeak.com
  {

    String postStr = apiKey;
    postStr += "&field1=";
    postStr += String(t);
    postStr += "&field2=";
    postStr += String(h);
    postStr += "&field3=";
    postStr += String(flowRate);
    postStr += "&field4=";
    postStr += String(level);
    postStr += "\r\n\r\n";

    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(postStr.length());
    client.print("\n\n");
    client.print(postStr);
    Serial.print("Temperature: ");
    Serial.print(t);
    Serial.print(" degrees Celcius, Humidity: ");
    Serial.print(h);
    Serial.println("%, Flow rate: ");
    Serial.print(flowRate);
    Serial.print("L/min, Water Level: ");
    Serial.println(level);
  }
  client.stop();

  Serial.println("Waiting...");

  // thingspeak needs minimum 15 sec delay between updates, i've set it to 30 seconds
  delay(1000);
}

int readSensor()
{
  digitalWrite(sensorPower, HIGH); // Turn the sensor ON
  delay(100);                      // wait 10 milliseconds
  val = analogRead(sensorPin);     // Read the analog value form sensor
  digitalWrite(sensorPower, LOW);  // Turn the sensor OFF
  // return val;                      // send current reading
  if (val < 100)
    return 0;
  if (val < 225)
    return 1;
  if (val < 250)
    return 2;
  else
    return 3;
}
```

Client 2 (Uno) located at Location 2:

```
#include <SoftwareSerial.h>
#include <SPI.h>
#include <Wire.h>
#define RX 10
#define TX 11
String AP = "No free wifi for you :P"; // CHANGE ME
String PASS = "srini@123456789";       // CHANGE ME
String API = "O8IV31YQWXR0RV1W";       // CHANGE ME
String HOST = "api.thingspeak.com";
String PORT = "80";
String field = "field1";
int countTrueCommand;
int countTimeCommand;
boolean found = false;
float valSensor = 1;
int p = 0;
SoftwareSerial esp8266(RX, TX);

#define SENSOR 2

long currentMillis = 0;
long previousMillis = 0;
int interval = 1000;
boolean ledState = LOW;
float calibrationFactor = 4.5;
volatile byte pulseCount;
byte pulse1Sec = 0;
float flowRate;
unsigned long flowMilliLitres;
unsigned int totalMilliLitres;
```

```
float flowLitres;
float totalLitres;
void pulseCounter()
{
  pulseCount++;
}
void setup()
{
  Serial.begin(9600);
  esp8266.begin(115200);
  pinMode(SENSOR, INPUT_PULLUP);
  pulseCount = 0;
  flowRate = 0.0;
  flowMilliLitres = 0;
  totalMilliLitres = 0;
  previousMillis = 0;
  attachInterrupt(digitalPinToInterrupt(SENSOR), pulseCounter, FALLING);
  sendCommand("AT", 5, "OK");
  sendCommand("AT+CWMODE=3", 5, "OK");
  sendCommand("AT+CWJAP=\"" + AP + "\",\"" + PASS + "\"", 20, "OK");
}
void loop()
{
  currentMillis = millis();
  if (currentMillis - previousMillis > interval)
  {

    pulse1Sec = pulseCount;
    pulseCount = 0;
    flowRate = ((1000.0 / (millis() - previousMillis)) * pulse1Sec) / calibrationFactor;
    previousMillis = millis();
    flowMilliLitres = (flowRate / 60) * 1000;
    flowLitres = (flowRate / 60);
    totalMilliLitres += flowMilliLitres;
    totalLitres += flowLitres;
    Serial.print("Flow rate: ");
    Serial.print(float(flowRate)); // Print the integer part of the variable
    Serial.print("L/min");
    Serial.println();
  }

  valSensor = flowRate;
  Serial.print("valSensor ");
  Serial.println(valSensor);
  String getData = "GET /update?api_key=" + API + "&" + field + "=" + String(valSensor);
  sendCommand("AT+CIPMUX=1", 5, "OK");
  sendCommand("AT+CIPSTART=0,\"TCP\",\"" + HOST + "\"," + PORT, 15, "OK");
  sendCommand("AT+CIPSEND=0," + String(getData.length() + 4), 4, ">");
  esp8266.println(getData);
  delay(1500);
  countTrueCommand++;
  sendCommand("AT+CIPCLOSE=0", 5, "OK");
}

void sendCommand(String command, int maxTime, char readReplay[])
{
  Serial.print(countTrueCommand);
  Serial.print(". at command => ");
  Serial.print(command);
  Serial.print(" ");
  while (countTimeCommand < (maxTime * 1))
  {
    esp8266.println(command);     //at+cipsend
    if (esp8266.find(readReplay)) //ok
    {
      found = true;
      break;
    }

    countTimeCommand++;
  }

  if (found == true)
  {
    Serial.println("OYI");
    countTrueCommand++;
    countTimeCommand = 0;
  }

  if (found == false)
  {
    Serial.println("Fail");
    countTrueCommand = 0;
    countTimeCommand = 0;
  }

  found = false;
}
```

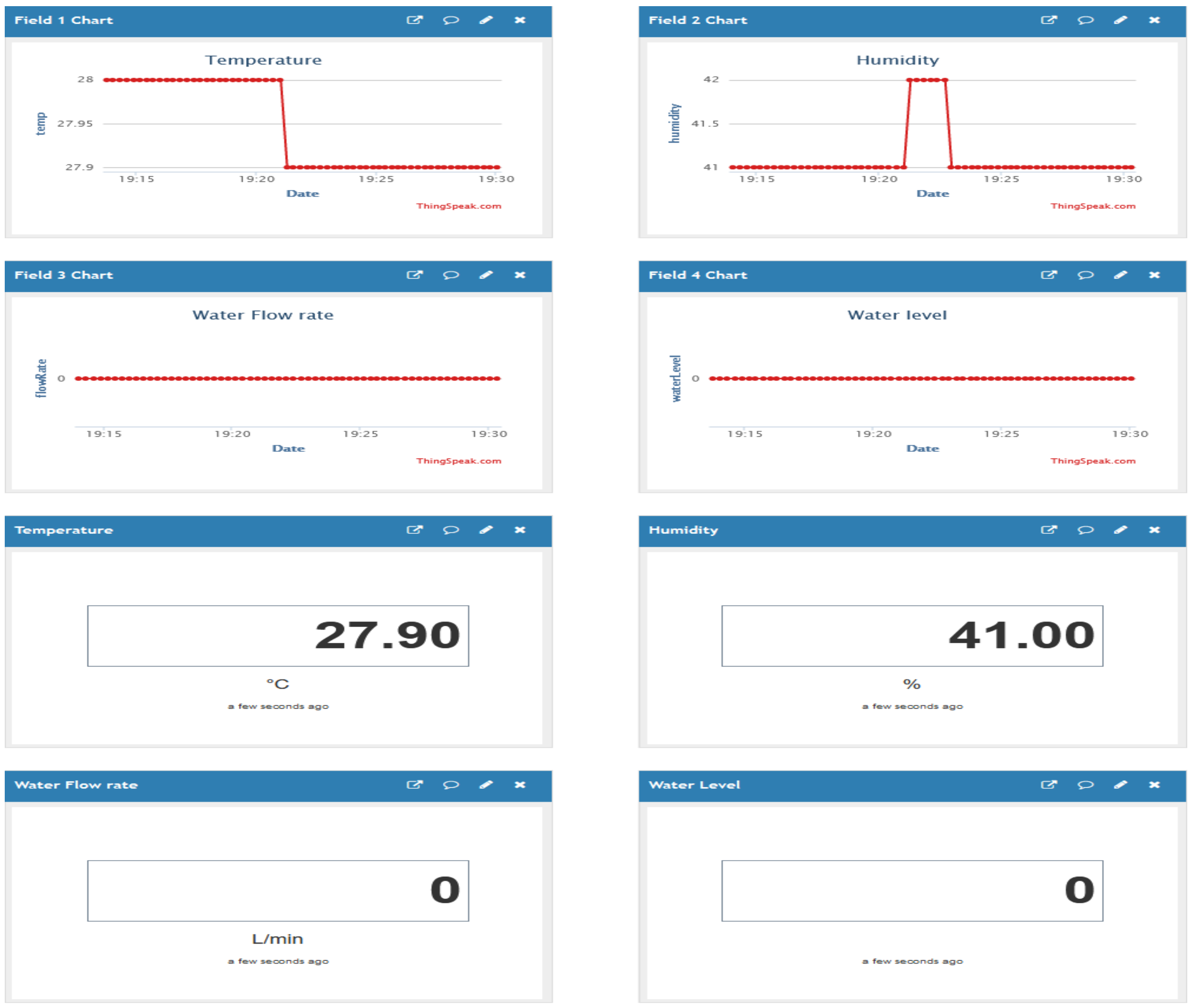
Server (NodeMCU)

```
#include <ThingSpeak.h>
#include <ESP8266WiFi.h>
#include <LiquidCrystal_I2C.h>
#include <base64.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
const char ssid[] = "No free wifi for you :P";
const char pass[] = "srini@123456789";
WiFiClient client;

//---------Channel Details---------//
unsigned long int interval = 86400000, prevSent2 = 0, prevSent1 = 0;
unsigned long counterChannelNumber1 = 1209982;
unsigned long counterChannelNumber2 = 1217400;          // Channel ID
const char *myCounterReadAPIKey = "98D2M11Y3EXC3IZJ";   // Read API Key
const char *myCounterReadAPIKey1 = "V6CPNBLFP7DZUWW7"; // Read API Key
const int FieldNumber1 = 1;                              // The field you wish to read
const int FieldNumber2 = 2;
const int FieldNumber3 = 3;
const int FieldNumber4 = 4;
bool safe1 = true;
bool safe2 = true;
//nearby numbers
String nearby = "9502215191";
//-------------------------------//
const char *account_sid = "AC05464dcb1ee0cf3258172f62eeffd0aa";
const char *auth_token = "1de03482e67b470032c297801d658c4c";
String from_number = "+18312176586";
String to_number = "+919502215191";
String message_body1 = "Alert: Go to 2";
String message_body2 = "Alert: Go to 1";
const char fingerprint[] = "BC B0 1A 32 80 5D E6 E4 A2 29 66 2B 08 C8 E0 4C 45 29 3F D0";
String urlencode(String str)
{
    String encodedString = "";
    char c;
    char code0;
    char code1;
    char code2;
    for (int i = 0; i < str.length(); i++)
    {
        c = str.charAt(i);
        if (c == ' ')
        {
            encodedString += '+';
        }
        else if (isalnum(c))
        {
            encodedString += c;
        }
        else
        {
            code1 = (c & 0xf) + '0';
            if ((c & 0xf) > 9)
            {
                code1 = (c & 0xf) - 10 + 'A';
            }
            c = (c >> 4) & 0xf;
            code0 = c + '0';
            if (c > 9)
            {
                code0 = c - 10 + 'A';
            }
            code2 = '\0';
            encodedString += '%';
            encodedString += code0;
            encodedString += code1;
        }
        yield();
    }
    return encodedString;
}

String get_auth_header(const String &user, const String &password)
{
    size_t toencodeLen = user.length() + password.length() + 2;
    char toencode[toencodeLen];
    memset(toencode, 0, toencodeLen);
    snprintf(toencode, toencodeLen, "%s:%s", user.c_str(), password.c_str());
    String encoded = base64::encode((uint8_t *)toencode, toencodeLen - 1);
    String encoded_string = String(encoded);
    std::string::size_type i = 0;
    // Strip newlines (after every 72 characters in spec)
    while (i < encoded_string.length())
    {
        i = encoded_string.indexOf('\n', i);
        if (i == -1)
        {
            break;
        }
        encoded_string.remove(i, 1);
    }
    return "Authorization: Basic " + encoded_string;
}

void setup()
{
    Serial.begin(115200);
    WiFi.mode(WIFI_STA);
    ThingSpeak.begin(client);
    lcd.begin();
    lcd.clear();
    lcd.print("Initializing...");
}

void loop()
{

    Serial.println();
    //----------------- Network -----------------//
    if (WiFi.status() != WL_CONNECTED)
    {
        Serial.print("Connecting to ");
        Serial.print(ssid);
        Serial.println(" ....");
        while (WiFi.status() != WL_CONNECTED)
        {
            WiFi.begin(ssid, pass);
            delay(5000);
        }
        Serial.println("Connected to Wi-Fi Succesfully.");
    }
    //--------- End of Network connection--------//
    Serial.println("For location 1:");
    //---------------- Channel 1 ----------------//
    float temp = ThingSpeak.readLongField(counterChannelNumber1, FieldNumber1,
    myCounterReadAPIKey);
    int statusCode = ThingSpeak.getLastReadStatus();
    if (statusCode == 200)
    {
        Serial.print("Temperature: ");
        Serial.print(temp);
        Serial.println(" C ");
    }
    else
    {
        Serial.println("Unable to read channel / No internet connection");
    }
    delay(100);
    //-------------- End of Channel 1 -------------//

    //---------------- Channel 2 ----------------//
    float humidity = ThingSpeak.readLongField(counterChannelNumber1, FieldNumber2,
    myCounterReadAPIKey);
    statusCode = ThingSpeak.getLastReadStatus();
    if (statusCode == 200)
    {
        Serial.print("Humidity: ");
        Serial.print(humidity);
        Serial.println("%");
    }
    else
    {
        Serial.println("Unable to read channel / No internet connection");
    }
    delay(100);
    //-------------- End of Channel 2 -------------//

    float flowRate1 = ThingSpeak.readLongField(counterChannelNumber1, FieldNumber3,
    myCounterReadAPIKey);
    statusCode = ThingSpeak.getLastReadStatus();
    if (statusCode == 200)
    {
        Serial.print("Water Flow Rate: ");
        Serial.print(flowRate1);
        Serial.println("L/min");
    }
    else
    {
        Serial.println("Unable to read channel / No internet connection");
    }
    delay(100);

    float waterLevel = ThingSpeak.readLongField(counterChannelNumber1, FieldNumber4,
    myCounterReadAPIKey);
    statusCode = ThingSpeak.getLastReadStatus();
    if (statusCode == 200)
    {
        Serial.print("Water Level: ");
        Serial.println(waterLevel);
    }
    else
    {
        Serial.println("Unable to read channel / No internet connection");
    }
    delay(100);
    Serial.println("\nFor Location 2:");
    float flowRate2 = ThingSpeak.readLongField(counterChannelNumber2, FieldNumber1,
    myCounterReadAPIKey1);
    statusCode = ThingSpeak.getLastReadStatus();
    if (statusCode == 200)
    {
        Serial.print("Water Flow Rate: ");
        Serial.print(flowRate2);
        Serial.println("L/min");
    }
    else
    {
        Serial.println("Unable to read channel / No internet connection");
    }
    delay(100);

    //temp,humidity,flowRate1,waterLevel,flowRate2

    if (waterLevel >= 2 || flowRate1 >= 10)
    {
        if (millis() - prevSent1 > interval || safe1)
        {
            sendsms(message_body1);
            prevSent1 = millis();
        }
        safe1 = false;
    }
    else
        safe1 = true;
    if (flowRate2 >= 10)
    {
        if (millis() - prevSent2 > interval || safe2)
        {
            sendsms(message_body2);
            prevSent2 = millis();
        }
        safe2 = false;
    }
    else
        safe2 = true;
    lcd.clear();
    if (safe1)
    {
        lcd.setCursor(0, 0);
        lcd.print("Place-1:SAFE");
    }
    else
    {
        lcd.setCursor(0, 0);
        lcd.print("Place-1:NOT SAFE");
    }
    if (safe2)
    {
        lcd.setCursor(0, 1);
        lcd.print("Place-2:SAFE");
    }
    else
    {
        lcd.setCursor(0, 1);
        lcd.print("Place-2:NOT SAFE");
    }
}

```

```
void sendsms(String message_body)
{
    WiFiClientSecure client;
    client.setFingerprint(fingerprint);
    Serial.printf("+ Using fingerprint '%s'\n", fingerprint);
    const char *host = "api.twilio.com";
    const int httpsPort = 443;
    Serial.print("+ Connecting to ");
    Serial.println(host);
    if (!client.connect(host, httpsPort))
    {
        Serial.println("- Connection failed.");
        return; // Skips to loop();
    }
    Serial.println("+ Connected.");
    Serial.println("+ Post an HTTP send SMS request.");
    String post_data = "To=" + urlencode(to_number) + "&From=" + urlencode(from_number) + "&Body=" + urlencode(message_body);
    String auth_header = get_auth_header(account_sid, auth_token);
    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";
    client.println(http_request);
    // Read the response.
    String response = "";
    while (client.connected())
    {
        String line = client.readStringUntil('\n');
        response += (line);
        response += ("\r\n");
    }
    Serial.println("+ Connection is closed.");
    Serial.println("+ Response:");
    Serial.println(response);
};
```

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