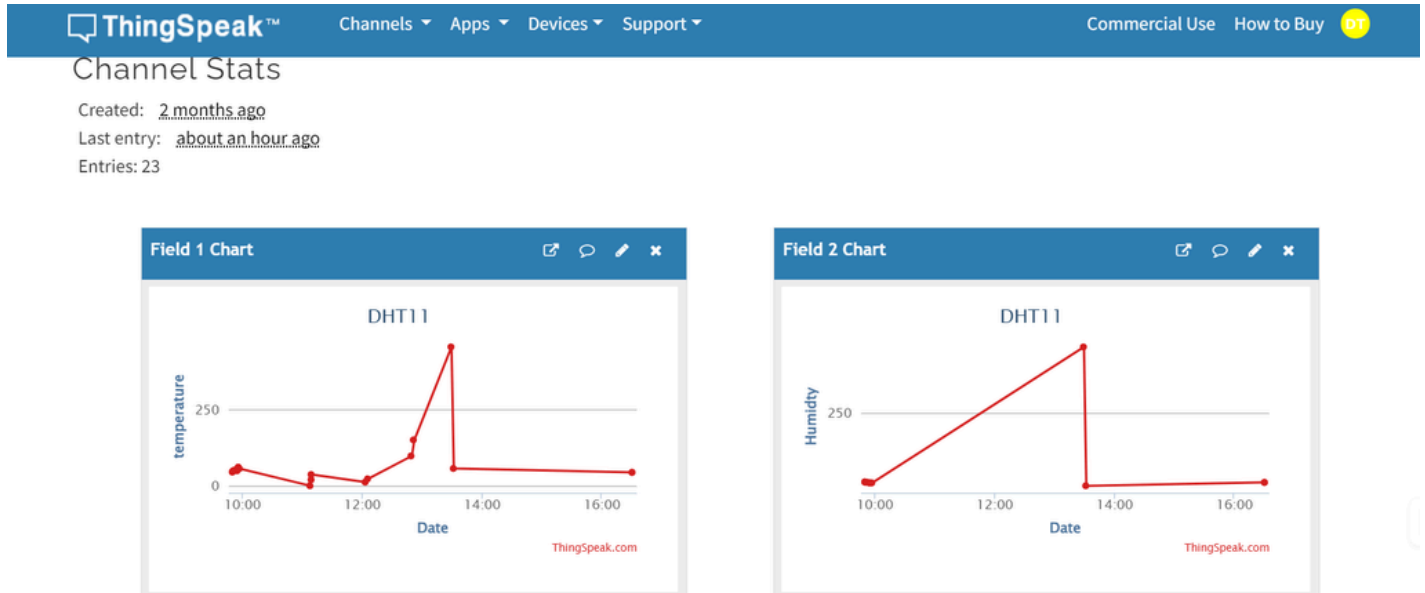
**AUTODESK**
Instructables

BharatPi 4G MQTT POST to ThingSpeak - a Practical Guide

By [bharatpi](#) in [CircuitsArduino](#)

Introduction: BharatPi 4G MQTT POST to ThingSpeak - a Practical Guide



ThingSpeak is an Internet of Things (IoT) platform that allows you to collect, analyze, and visualize data from sensors or devices. It provides an easy-to-use interface for users to send data from their IoT devices to ThingSpeak servers using various communication protocols such as HTTP, MQTT, and ThingSpeak API. Once the data is in ThingSpeak, users can perform real-time analytics, create visualizations such as charts and maps, and trigger actions based on predefined conditions. ThingSpeak is often used in various IoT applications including environmental monitoring, home automation, and industrial automation.

Supplies

Sensor

Bharat Pi Boards

<https://bharatpi.net/>

ThingSpeak


<https://thingspeak.com/>

Arduino ide

Step 1: How to Create a Account in Thingspeak

ThingSpeak™


Channels Apps Support ▾

Commercial Use How to Buy 

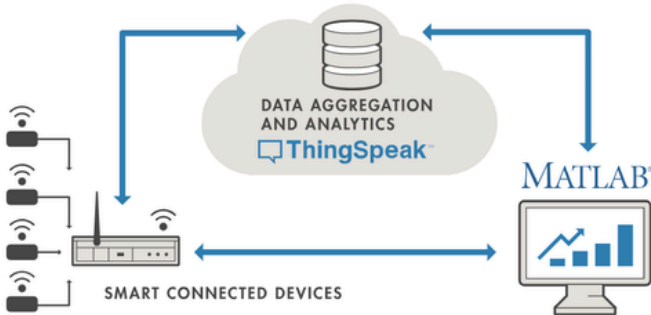
To use ThingSpeak, you must sign in with your existing MathWorks account or create a new one.

Non-commercial users may use ThingSpeak for free. Free accounts offer limits on certain functionality. Commercial users are eligible for a time-limited free evaluation. To get full access to the MATLAB analysis features on ThingSpeak, log in to ThingSpeak using the email address associated with your university or organization.

To send data faster to ThingSpeak or to send more data from more devices, consider the [paid license options](#) for commercial, academic, home and student usage.


Email

[No account? Create one!](#)
By signing in, you agree to our [privacy policy](#).
[Next](#)



Visit ThingSpeak: Go to the ThingSpeak website at [ThingSpeak thingspeak.com].

Click on Sign Up: In the top right corner of the homepage, you'll see a button labeled "Sign Up." Click on that button.

Fill out the Registration Form: On the registration page, you'll be asked to provide the following information:

- Username: Choose a unique username that you'll use to log in to your ThingSpeak account.
- Email Address: Enter a valid email address that you want to associate with your ThingSpeak account.
- Password: Create a strong password that includes a combination of letters, numbers, and special characters. Make sure it's something you can remember easily but difficult for others to guess.
- Confirm Password: Retype your chosen password to ensure there are no typos.

Create Account: Once you've filled out the form, click on the button that says "Create Account."

Verify Your Email (Optional): ThingSpeak might send a verification email to the address you provided. Check your inbox and follow any instructions in the email to verify your account.

Step 2: How to Create a Channel

Log in to ThingSpeak: Go to the ThingSpeak website ([ThingSpeak thingspeak.com]) and enter your login credentials.

Access Channels: Once logged in, navigate to the "Channels" section. You can usually find this by clicking on the "Channels" tab or a similar option on the ThingSpeak dashboard.

Create New Channel: Look for a button labeled "New Channel" or "Create Channel." Clicking on this button will initiate the channel creation process.

Channel Settings: Here, you'll define the specifics of your channel:

- **Name:** Choose a descriptive name for your channel that reflects the type of data it will store (e.g., "Home Temperature Monitoring").
- **Fields:** Select the number of data fields you want your channel to have (usually 1 to 8). Each field will represent a specific sensor measurement you plan to collect.
- You can check the boxes next to "Field 1" to "Field 8" depending on how many data streams you need.
- ThingSpeak allows you to name each field later, making it easier to understand what data is being stored in each slot.

Public View (Optional): By default, channels are private. You can choose to make your channel public if you want to share the data with others.

Save Channel: Once you've filled out the channel settings, click on the "Save Channel" button.

Step 3: How to Create a Channel for MQTT

1 go to devices -> select the mqtt

2 copy paste the mqtt client id

3 "mqtt3.thingspeak.com" change this to as per your server MQTT broker(in the code)

4 click on the edit for the user id and the password

Step 4: Code for Mqtt

```

/*****
PROJECT: Bharat Pi 4G Board Sample Code for data push to Thingspeak Cloud
AUTHOR: Bharat Pi

FUNC: 4G testing with HTTP call to Thingspeak cloud server.

SIMCARD: 4G sim cards from Airtel/Vodafone/Jio/BSNL can be used.
        To test HTTP request ensure you have data plan.

IMPORTANT: Configure the APN accordingly as per your Sim provider

GPS: If you have bought Bharat Pi wtiH GPS board then you need to connect
      the GPS antenna to the UFL connector and antenna should have clear sky visibility
      preferrably on the terrace or open field.

TODO: (Before you upload the code to Bharat Pi board)
1) Change APN as per your Sim provider
2) Change the Thingspeak API key as per your account/setup
3) Use a power adapter 9V 2amps as the 4G module requires enough power to operate

DESC: This script will connect using 4G sim and makes a http call
      to a Thingspeak cloud server setup as per your Thingspeak API key.

COPYRIGHT: BharatPi @MIT license for usage on Bharat Pi boards
*****/

#define TINY_GSM_MODEM_SIM7600 //TINY_GSM_MODEM compatible for 7672 as well
#define TINY_GSM_RX_BUFFER 1024
// #define DUMP_AT_COMMANDS true

#define SerialAT Serial1
#define SerialMon Serial

#define UART_BAUD    115200
#define PIN_DTR      25
#define PIN_TX       17
#define PIN_RX       16
#define PWR_PIN      32

#define LED_PIN 2

#include <TinyGsmClient.h> //Lib version - 0.12.0
#include "SSLCient.h" //Lib version - 2.1.7
#include <PubSubClient.h> //Lib version - 2.8

// certificate for https://mqtt3.thingspeak.com
// DigiCert Global Root G2, valid until Sun Mar 30 2031, size: 1720 bytes
// const char* rootCACertificate = \
// "-----BEGIN CERTIFICATE-----\n" \
// "MIIEyDCCA7CgAwIBAgIQDPW9BItWAvR6uFAsI8zwZjANBgkqhkiG9w0BAQSFADBN\n" \
// "MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3\n" \
// "d3cuZGlnaWlnalcnQuY29tMSAwaHgYDVQQDEXdEaWdpQ2VydCBHbG9iYWwgUm9vdCBH\n" \
// "MjAeFw0yMTAzMzAwMDAwMDBaFw0zMTAzMjkYMAZU5NTlaMFkxZCAzAjBgNVBAYTA1VT\n" \
// "MRUwEwYDVQQKEwwEaWdpQ2VydCBJbmMxMzAxAgNBVBAMTKkrPZ21ldXZJ0IEdsb2Jh\n" \
// "bCBHMmIuBTfmgUlNBIFNIQTi1NiAyMDIwIENBMTCASIAwDQYJKoZIhvcNAQEBBQAD\n" \
// "ggEPADCCAQoCggEBAMz3EGJPprtjb+2QUlbFbSd7ehJWivH0+dbn4Y+9lavyyEEV\n" \
// "cNsSAPonCrVXOFt9slGTcZUOakGUWzUb+nv6u8W+JDD+Vu/E832X4xT1FE3LpxDy\n" \
// "FuqrIVaxIhFhaZAmunjZlx/jfwardUSVCis/+9dCopZQ+GssjoP80j812s3wWPc\n" \
// "3kbw20X+fSP9K0hRBx5Ro1/tSUZUfyIXfQTnJcVPAPOotNcaQwywa8W0YUR0J8\n" \
// "osicfebutVsVqpmowQTDcd5zWSOTOEeaqJnwQ3DPP3Zr0UXjqyRewg2C/Uaoq2yT\n" \
// "zGJSQnWS+Jr6X16ysGH1Hx+5fwymY6D36g39HaaECAwEAACAIAwIGgf+MBIGA1Ud\n" \
// "EWEB/wIIMAYBAf8CAQAwhQYDVR0OBByEFHSFGMBmx9833s+9KTeqAx2+7c0XMB8G\n" \
// "A1UdIwQYMBAAFE4iVCAYlebjuYP+vq5Eu0GF485MA4GA1UdDwEB/wQEAWIBhjAd\n" \
// "BgNVHSEUFjAUBggrBgEFBQCDAQYIKwYBBQUHAUIwdgYIKwYBBQUHAQEaJBQMCGQ\n" \
// "CCSGAUQFBzABhhhdHRwOi8vb2Nzc5kaWdpY2VydC5jb20wYAYIKwYBBQUHMAKG\n" \
// "NGh0dHA6Ly9jYWNlcnRzLmRpZ21jZXJ0LmNvbS9EawdpQ2VydEdsb2JhbFJvb3RH\n" \
// "Mi5jcncwQgYDVR0fBDswOTA3ODWgM4YxaHR0CDovL2NybdMuZGlnaWlnalcnQuY29t\n" \
// "L0RpZ21ldXZJ0R2xxvmFsuUm9vdEcYLmNybdA9BgNVHSAEAnJA0MASGCWCgsAGG/\n" \
// "ATAHBgVngQwBATAIBgzNgQwBagEwCAYGZ4EMAQICMAgBmeBDAECAZANBgkqhkiG\n" \
// "9w0BAQFAAOCAQEApyyiiaXzd8dP3A+iZ7U6utZW9UpwpmIrXwkOH7U1MV1+t\n" \
// "wcW1BSAuWDH/SvWgktiwla3JLko716f2b4gp/DA/JIS7w7dkwcsr4drdjPtAFVS\n" \
// "slme5LQ89/nD/7d+mS5EHKBQRfz5eeLJJjs+aWNJXXM43AYGyZmopGrFmCW3R\n" \
// "bpD0ufovARTFXZkad19h6g4U5+LXUZtXYnhIHUfoyoMo5t558aI7Dd8KvvwVVo4\n" \
// "chDYABPPTHPbjqc1qCmbBaXzvN4Ye5DUys/vZwP9BFohFrH/6j/f3IL16/RZkiMN\n" \
// "JCqVJUzKoZhmlLesh3Sz8W2jmdv51b2EQJ8HmA==\n" \
// "-----END CERTIFICATE-----\n" \
// "";
```

```

/*****
SECTION: Set APN based on your sim card
AIRTEL: "airtelgprs.com"
BSNL: "bsnlnet"
VI: "portalnmms"
JIO: "jionet"
*****/
const char apn[] = "airtelgprs.com"; //Change this as per your Sim card operator

/*****
SECTION: Set Thingspeak cloud specific
*****/
const char broker[] = "mqtt3.thingspeak.com"; //change this to as per your server MQTT broker
const int port = 1883; //MQTT port
const char* channelID = "2456524"; //Thingspeak channel ID, set this to as per your channels on thingspeak

//thingspeak MQTT Client ID settings. Refer thingspeak MQTT settings
const char* mqttClientID = "0w0UFh47FzsqHj00JAAuGgo";
const char* mqttUsername = "0w0UFh47FzsqHj00JAAuGgo";
const char* mqttPassword = "7kq9ySBZd7oxWZln3v58/7R8";

#ifdef DUMP_AT_COMMANDS
#include <StreamDebugger.h>
StreamDebugger debugger(SerialAT, SerialMon);
TinyGsm modem(debugger);
#else
TinyGsm modem(SerialAT);
#endif

TinyGsmClient client(modem);
//SSLClient secure_client(&client);
PubSubClient mqtt(client);

/*****
SECTION: Modem operation functions
*****/
void modemPowerOn(){
  pinMode(PWR_PIN, OUTPUT);
  digitalWrite(PWR_PIN, LOW);
  delay(1000);
  digitalWrite(PWR_PIN, HIGH);
}

void modemPowerOff(){
  pinMode(PWR_PIN, OUTPUT);
  digitalWrite(PWR_PIN, LOW);
  delay(1500);
  digitalWrite(PWR_PIN, HIGH);
}

void modemRestart(){
  modemPowerOff();
  delay(1000);
  modemPowerOn();
}

void printLocalTime()
{
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
    return;
  }
  Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
}

void mqttConnect(String payload) {
  SerialMon.print("Setting MQTT broker: ");
  SerialMon.println(broker);

  mqtt.setServer(broker, port);
  mqtt.setBufferSize(2048);

  // Connect to MQTT Broker
  SerialMon.println("Connecting to MQTT broker");
  if (!mqtt.connect(mqttClientID, mqttUsername, mqttPassword)) {
    Serial.println("MQTT Client connect failed");
  } else {

```

```

Serial.println("MQTT Client connect success...");
const char* topicString = "channels/2456524/publish"; //Refer thingspeak on how to set the MQTT topics
const char* message = payload.c_str();
Serial.println("MQTT publish in progress...");
if(mqtt.connected()){
  if(mqtt.publish(topicString, message)){
    Serial.println("Data published to MQTT server: " + payload);
  } else {
    Serial.println("ERROR: Data publish failed to MQTT server!");
  }
} else {
  Serial.println("MQTT is disconnected!");
}
}
}

/*****
SECTION: Main setup
*****/
void setup(){
  // Set console baud rate
  SerialMon.begin(115200);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);

  delay(100);

  modemPowerOn();
  //secure_layer.setCACert(root_ca);
  SerialAT.begin(UART_BAUD, SERIAL_8N1, PIN_RX, PIN_TX);
  Serial.clearWriteError();
  Serial.println();
  Serial.println();
  Serial.println("/*****");
  Serial.println(" Bharat Pi Thingspeak Cloud Sync using MQTT over 4G/LTE ");
  Serial.println("");
  Serial.println(" IMPORTANT: To initialize/latch the 4G/LTE network, please make sure the antenna has been");
  Serial.println(" connected, SIM is inserted in the SIM slot (back side of the board) and 9V 2A power adapter is connected.");
  Serial.println("/*****\n\n");

  delay(2000);

  String res;
  Serial.println("Initializing Modem...");

  if (!modem.init()) {
    digitalWrite(LED_PIN, HIGH);
    modemRestart();
    delay(2000);
    Serial.println("Failed to restart modem, continue without restarting");
    digitalWrite(LED_PIN, LOW);
    return;
  }

  //Blue LED on the board use as an indicator
  //If blinking: Modem not able to boot
  //If turned ON: connected to network
  //If turned OFF: Modem booted successfully but not connected to network, check your SIM, network coverage etc.

  digitalWrite(LED_PIN, LOW);

  Serial.println("Running SIMCOMATI command...");
  modem.sendAT("+SIMCOMATI"); //Get the module information
  modem.waitForResponse(1000L, res);
  //res.replace(GSM_NL "OK" GSM_NL, "");
  Serial.println(res);
  res = "";
  Serial.println();

  Serial.println("Preferred mode selection (GSM/LTE)...");
  modem.sendAT("+CNMP?");
  if (modem.waitForResponse(1000L, res) == 1) {
    //res.replace(GSM_NL "OK" GSM_NL, "");
    Serial.println(res);
  }
  res = "";
  Serial.println();

```

```

//This section is only applicable for testing modules with NBIoT,
//for other modules the command doesn't return anything and can be ignored while testing
Serial.println("Preferred selection between CAT-M and NB-IoT...");
modem.sendAT("+CMNB?");
if (modem.waitForResponse(1000L, res) == 1) {
    //res.replace(GSM_NL "OK" GSM_NL, "");
    Serial.println(res);
}
res = "";
Serial.println();

//Get module manufacturer details
String modemName = modem.getModemModel();
Serial.println("Modem Name : " + modemName);
delay(1000);

String modemInfo = modem.getModemInfo();
Serial.println("Modem Info : " + modemInfo);
delay(1000);

/*****
SECTION: Connect to Sim network
*****/
Serial.println("Network mode connectivity testing (GSM, LTE or GSM/LTE)...");

for (int i = 0; i <= 4; i++) {
    uint8_t network[] = {
        2, /*Automatic*/
        13, /*GSM only*/
        38, /*LTE only*/
        51 /*GSM and LTE only*/
    };
    Serial.printf("Try %d method\n", network[i]);
    modem.setNetworkMode(network[i]);
    delay(3000);
    bool isConnected = false;
    int tryCount = 60;
    while (tryCount-- > 0) {
        String networkOperator = modem.getOperator();
        Serial.print("Operator: ");
        Serial.println(networkOperator);
        int16_t signal = modem.getSignalQuality();
        Serial.print("Signal: ");
        Serial.println(signal);
        Serial.print("isNetworkConnected: ");
        isConnected = modem.isNetworkConnected();
        Serial.println( isConnected ? "CONNECTED" : "NOT CONNECTED YET");
        if (isConnected) {
            break;
        }
        delay(1000);
        digitalWrite(LED_PIN, !digitalRead(LED_PIN));
    }
    if (isConnected) {
        break;
    }
}
digitalWrite(LED_PIN, HIGH); //Modem connected to network

Serial.println();
Serial.println("Yehhh....Device is connected to Sim network.");
Serial.println();

delay(1000);
Serial.println("Checking UE (User Equipment) system information...");
Serial.println();
modem.sendAT("+CPSI?");
if (modem.waitForResponse(1000L, res) == 1) {
    res.replace(AT_NL "OK" AT_NL, "");
    Serial.println(res);
}

delay(1000);
Serial.println("");
Serial.println("");

if(modem.isNetworkConnected()){
    Serial.println("Mobile Network is connected.....");
}

```

```
// GPRS connection parameters are usually set after network registration
SerialMon.print(F("DATA Connection: Connecting to APN: "));
SerialMon.print(apn);
if (!modem.gprsConnect(apn, "", "")) {
  Serial.println("APN connect failed");
  delay(10000);
  return;
}
Serial.println("APN connect success");

if (modem.isGprsConnected()) {
  Serial.println("");
  Serial.println("GPRS network is connected");
}
Serial.println("");

/*****
SECTION: Set variables and values for pushing data to cloud via MQTT.
If you want to dynamically capture data from
sensors or GPIOs then move this section to Loop
and set variables accordingly to push data to Thingspeak.
*****/
String payload;
double temperature1 = 32.9; //Dummy vairable to send temperature
double temperature2 = 36.2; //Dummy vairable to send temperature

//Prepare paylod if using the MQTT
payload = "field1=" + String(temperature1) + "&field2=" + String(temperature2) + "&status=MQTTPUBLISH";

Serial.print("Paylod length: ");
Serial.println(payload.length());

Serial.println();
Serial.println(">>>> Begin Thingspeak Cloud Sync using MQTT <<<<<");
Serial.println();

Serial.println("NOTE: Please ensure to have a Thingspeak MQTT setup for your channel.\n"
"You can find the steps on Thingspeak portal on how to set MQTT for a channel and use\n"
"the channel IDs to publish the fields (sensors) data.");

//MQTT settings
Serial.println("");
Serial.println(" Thingspeak MQTT: " + String(broker));
Serial.println(" Thingspeak Channel: " + String(channelID));

Serial.println();
Serial.println();
delay(2000);

Serial.println("Initiating MQTT connect to Thingspeak");

/*****
SECTION: Make an API call to Thingspeak server with sensor data as a Json object
*****/

Serial.println(".....MQTT connect and publish in progress.....");
//secure_client.setCACert(rootCACertificate);
//delay(5000);
//Serial.println("Root certificate set successfully.....");
mqttConnect(payload);

Serial.println("");
Serial.println(">>>> END OF Thingspeak Cloud Sync <<<<<");
}

/*****
SECTION: Loop call for capturing synamic data from sensors/GPIOs
*****/

void loop(){
  //Print data to Arduino serial monitor
  while (1) {
    while (SerialAT.available()) {
      SerialMon.write(SerialAT.read());
    }
    while (SerialMon.available()) {
```



```
        SerialAT.write(SerialMon.read());  
    }  
}
```

Step 5: Code Explanation

1 Libraries and Definitions

```
#define TINY_GSM_MODEM_SIM7600 //TINY_GSM_MODEM compatible for 7672 as well
#define TINY_GSM_RX_BUFFER 1024
//#define DUMP_AT_COMMANDS true

#define SerialAT Serial1
#define SerialMon Serial

#define UART_BAUD    115200
#define PIN_DTR      25
#define PIN_TX        17
#define PIN_RX        16
#define PWR_PIN       32

#define LED_PIN 2

#include <TinyGsmClient.h> //Lib version - 0.12.0
#include "SSLCient.h" //Lib version - 2.1.7
#include <PubSubClient.h> //Lib version - 2.8
```

- These lines define which modem to use and set up buffer sizes and serial port configurations.
- Libraries for handling GSM communication, SSL, and MQTT are included.

2. Certificate Definition (Commented Out)

```
cpp
Copy code
// certificate for https://mqtt3.thingspeak.com
// DigiCert Global Root G2, valid until Sun Mar 30 2031, size: 1720 bytes
// const char* rootCACertificate = ...
```

- The certificate for SSL connection to ThingSpeak is commented out. This can be included if secure MQTT communication is required.

3. APN and ThingSpeak Configuration

```
const char apn[] = "airtelgprs.com"; //Change this as per your Sim card operator
const char broker[] = "mqtt3.thingspeak.com";
const int port = 1883;
const char* channelID = "2456524";

const char* mqttClientID = "0w0UFh47FzsqHj00JAAuGgo";
const char* mqttUsername = "0w0UFh47FzsqHj00JAAuGgo";
const char* mqttPassword = "7kq9ySBZd7oxWZ1n3v58/7R8";
```

- The APN (Access Point Name) for the SIM card and MQTT broker details are set. These values must be updated according to the SIM card provider and ThingSpeak channel configuration.

4. Modem Functions

```
void modemPowerOn(){ ... }
void modemPowerOff(){ ... }
void modemRestart(){ ... }
void printLocalTime(){ ... }
void mqttConnect(String payload) { ... }
```

- Functions for powering the modem on and off, restarting it, and connecting to MQTT are defined. The mqttConnect function publishes data to ThingSpeak.

5. setup Function

```
void setup(){
    ...
    modemPowerOn();
    ...
    String res;
    ...
    // Initialize Modem
    ...
    // Connect to Network
    ...
    String payload;
    double temperature1 = 32.9; //Dummy variable to send temperature
```

```
double temperature2 = 36.2; //Dummy variable to send temperature

payload = "field1=" + String(temperature1) + "&field2=" + String(temperature2) + "&status=MQTTPUBLISH";
...
mqttConnect(payload);
...
}
```

- The setup function initializes the modem, connects to the network, and sets up a payload with dummy temperature data. It then calls the mqttConnect function to publish the data to ThingSpeak.

6. loop Function

```
void loop(){
  while (1) {
    while (SerialAT.available()) {
      SerialMon.write(SerialAT.read());
    }
    while (SerialMon.available()) {
      SerialAT.write(SerialMon.read());
    }
  }
}
```

- The loop function continuously reads from the modem and prints any data to the serial monitor, allowing you to debug and see real-time modem responses.

Key Points to Remember

1. **APN Configuration:** Ensure the APN is set correctly for your SIM card provider.
2. **ThingSpeak Configuration:** Update the MQTT credentials and channel ID with your own ThingSpeak details.
3. **Power Supply:** Use a 9V 2A power adapter to ensure the modem operates correctly.
4. **Debugging:** The code includes extensive serial prints for debugging purposes.

This setup is designed for testing and should be customized based on your specific requirements and environment. Make sure to replace the placeholder values (APN, MQTT credentials, channel ID) with your actual data before deploying the code.

Step 6: Wifi Code

```
#include <WiFi.h>
#include "DHT.h"

// WiFi credentials
const char* ssid = "smile123";
const char* password = "123456789";

// Thingspeak API key
const String apiKey = "20MNHFQKV1NFW83I";

// Thingspeak server
const char* server = "api.thingspeak.com";

// DHT sensor pin and type
#define DHTPIN 23
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  Serial.println("Serial Begin");

  // Initialize DHT sensor
  dht.begin();

  // Connect to WiFi
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

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

  Serial.println("");
  Serial.println("WiFi connected");
}

void loop() {
  // Read DHT sensor data
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  // Check if any reads failed and exit early (to try again)
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Send data to Thingspeak
  if (sendToThingspeak(humidity, temperature)) {
    Serial.println("Data sent to Thingspeak successfully");
  } else {
    Serial.println("Failed to send data to Thingspeak");
  }

  // Delay before next reading
  delay(10000); // Delay for 10 seconds (adjust as needed)
}

bool sendToThingspeak(float humidity, float temperature) {
  WiFiClient client;

  // Create HTTP request body
  String requestBody = "api_key=" + apiKey;
  requestBody += "&field1=" + String(humidity);
  requestBody += "&field2=" + String(temperature);

  // Send HTTP POST request to Thingspeak
  if (client.connect(server, 80)) {
    client.println("POST /update HTTP/1.1");
    client.println("Host: api.thingspeak.com");
    client.println("Connection: close");
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.println("Content-Length: " + String(requestBody.length()));
```

```
client.println();
client.print(requestBody);
delay(500); // Allow time for response
client.stop();
return true; // Data sent successfully
} else {
  Serial.println("Connection to Thingspeak failed");
  return false; // Failed to send data
}
}
```