Experiment No. – 5

Title: Introduction to MQTT/ CoAP and sending sensor data to cloud using Raspberry-Pi/Beagle board/Arduino.

Aim: Interface a DHT11 sensor with Arduino and Write a program to read the values for Temperature and Humidity and send it to Thingspeak cloud using MQTT.

Hardware Requirements: Arduino uno , DHT11 Sensor, ESP8266 Module

Software Requirements: Arduino IDE

Theory:

DHT11 Sensor:

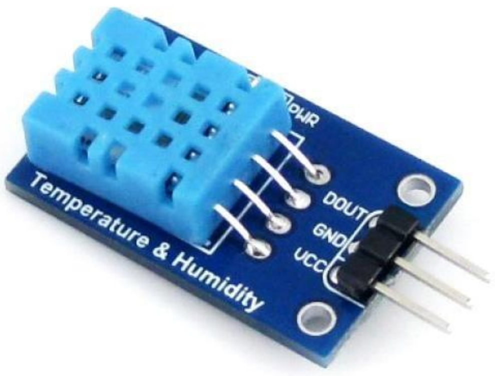
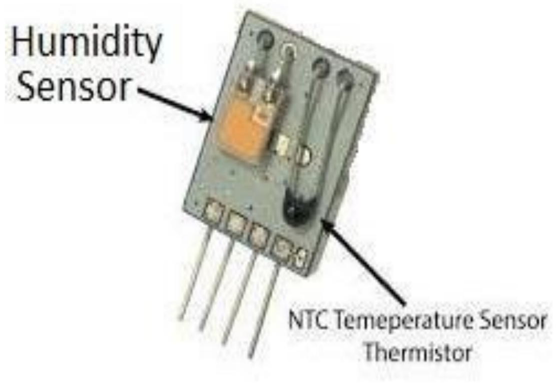


Fig: 1 DHT11 Sensor

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a



digital signal on the data pin (no analog input pins needed). It’s fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds, so when using library, sensor readings can be up to 2 seconds old.

Specifications

- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- Body size 15.5mm x 12mm x 5.5mm
- 4 pins with 0.1" spacing

Pin Configuration

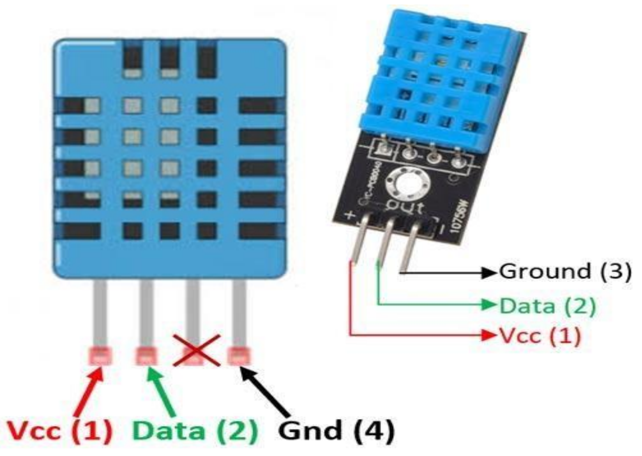


Fig: 2 Pin description of DHT11

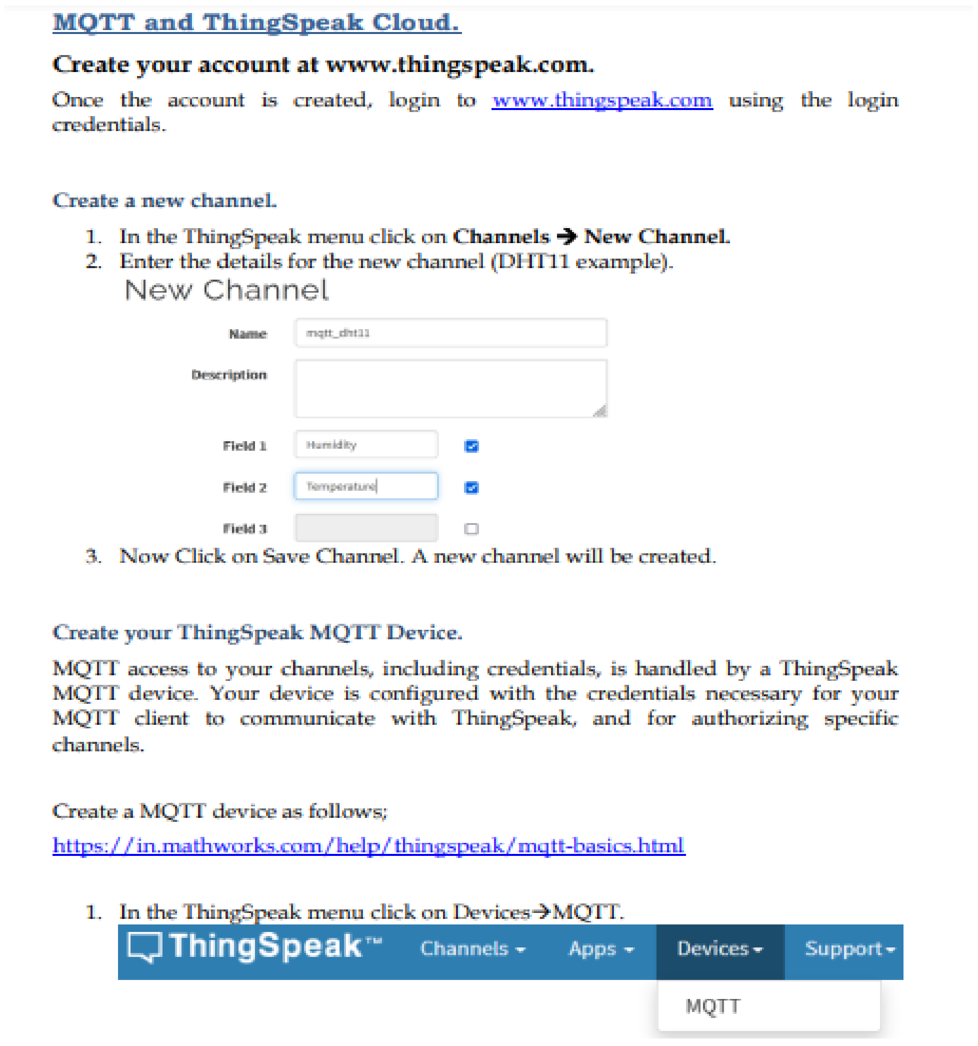
No:	Pin Name	Description
For Sensor		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	NC	No Connection and hence not used
4	Ground	Connected to the ground of the circuit
For module		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

ESP8266 Module :

The ESP8266 Wi-Fi Module is used to connect with any available internet hotspot and control the LED light. The ESP8266 Wi-Fi Module is a self contained SOC with integrated TCP/IP protocol stack that can provide access to a Wi-Fi network. The ESP8266 is capable of either hosting an application or off loading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware. This firmware can be used to communicate to the ESP8266 module via **AT command** But, if the Arduino IDE is used, this firmware will be over written. If the Arduino IDE is once used to program the ESP module, then AT commands cannot be used to configure it. Hence, the module is flashed with the default firmware so that first the AT commands can be used to configure it.

ThingsBoard –

- It is an open-source server-side platform that allows you to monitor and control IoT devices. It is free for both personal and commercial usage and you can deploy it anywhere
- This sample application performs collection of temperature and humidity values produced by DHT22 sensor and further visualization on the real-time web dashboard. Collected data is pushed via MQTT to ThingsBoard server for storage and visualization.
- The DHT22 sensor is connected to Raspberry Pi. Raspberry Pi offers a complete and selfcontained Wi-Fi networking solution. Raspberry Pi push data to ThingsBoard server via MQTT
- Setting up the ThingSpeak Account: 1. First of all, go to the following link and sign up to ThingSpeak. If you already have an account, then sign in. https://thingspeak.com/
-



MQTT and ThingSpeak Cloud.

Create your account at www.thingspeak.com.

Once the account is created, login to www.thingspeak.com using the login credentials.

Create a new channel.

1. In the ThingSpeak menu click on **Channels → New Channel.**
2. Enter the details for the new channel (DHT11 example).

New Channel

Name	mqtt_dht11	
Description		
Field 1	Humidity	✓
Field 2	Temperature	✓
Field 3		☐

3. Now Click on Save Channel. A new channel will be created.

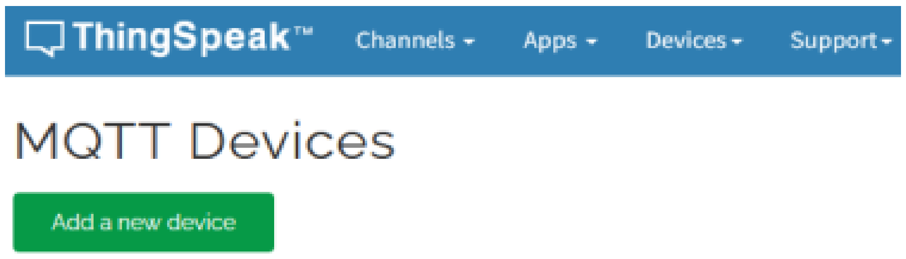
Create your ThingSpeak MQTT Device.

MQTT access to your channels, including credentials, is handled by a ThingSpeak MQTT device. Your device is configured with the credentials necessary for your MQTT client to communicate with ThingSpeak, and for authorizing specific channels.

Create a MQTT device as follows;
https://in.mathworks.com/help/thingspeak/mqtt-basics.html

1. In the ThingSpeak menu click on Devices→MQTT.

2. On the MQTT Devices page, click Add a new device.

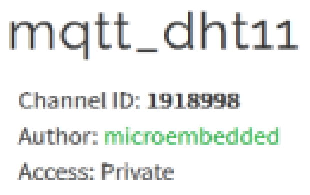


MQTT Devices

3. Fill in the Add a new device dialog:
- Provide a device name.
- Provide an optional description. You can put any information.
- In the **Authorize Channels** section, choose a channel from the list menu (chose your channel –DHT11 example channel), then click **Add Channel**. Set each to allow publish and allow subscribe as needed.
- Click **Add Device**.
- At this point, ThingSpeak generates a list of credentials for your device that includes client ID, username, and password. You can view and copy these items from this page, or click **Download Credentials** (plain text) to save the credentials in a local file. **Important**: Record or save your credentials now, as you will not get another opportunity to view or save the password.
- Click **Done** to complete the device creation.

Open the plain text file and copy the credentials and put it in our program wherever required. Also note down the Channel ID of the channel that you created.

Channels→My Channel →***your channel name***

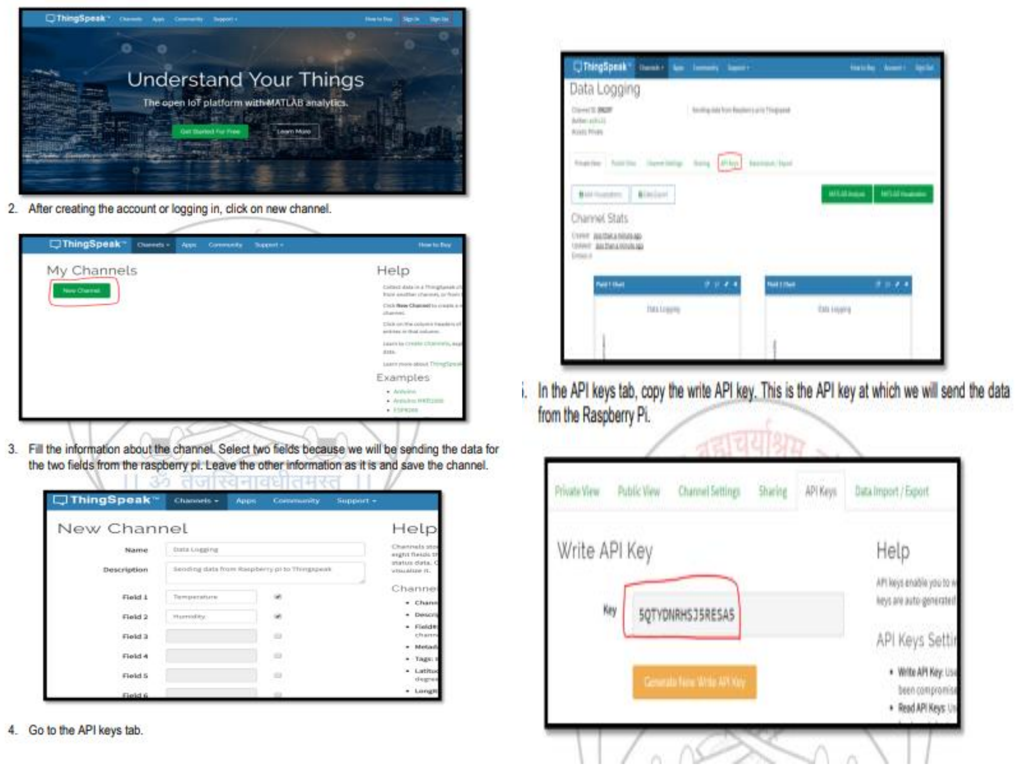


mqtt_dht11

Channel ID: **1918998**

Author: microembedded

Access: Private



2. After creating the account or logging in, click on new channel.

3. Fill the information about the channel. Select two fields because we will be sending the data for the two fields from the raspberry pi. Leave the other information as it is and save the channel.

Name	Data Logging
Description	Sending data from Raspberry pi to Thingspeak
Field 1	Temperature
Field 2	Humidity

4. Go to the API keys tab.

i. In the API keys tab, copy the write API key. This is the API key at which we will send the data from the Raspberry Pi.

Circuit Diagram:

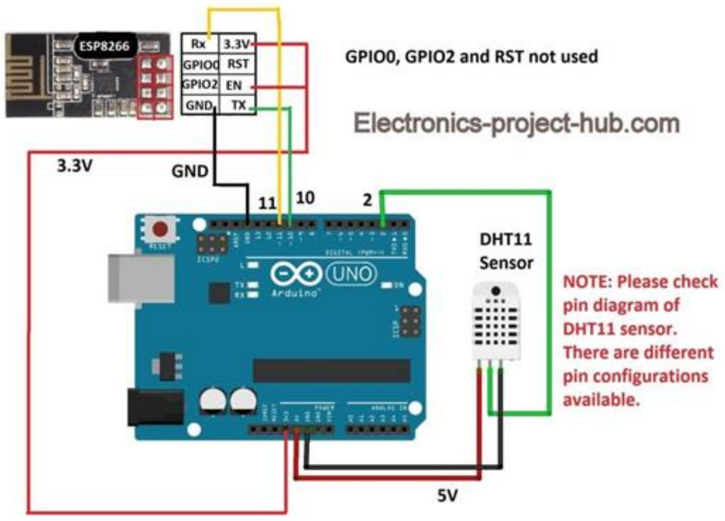
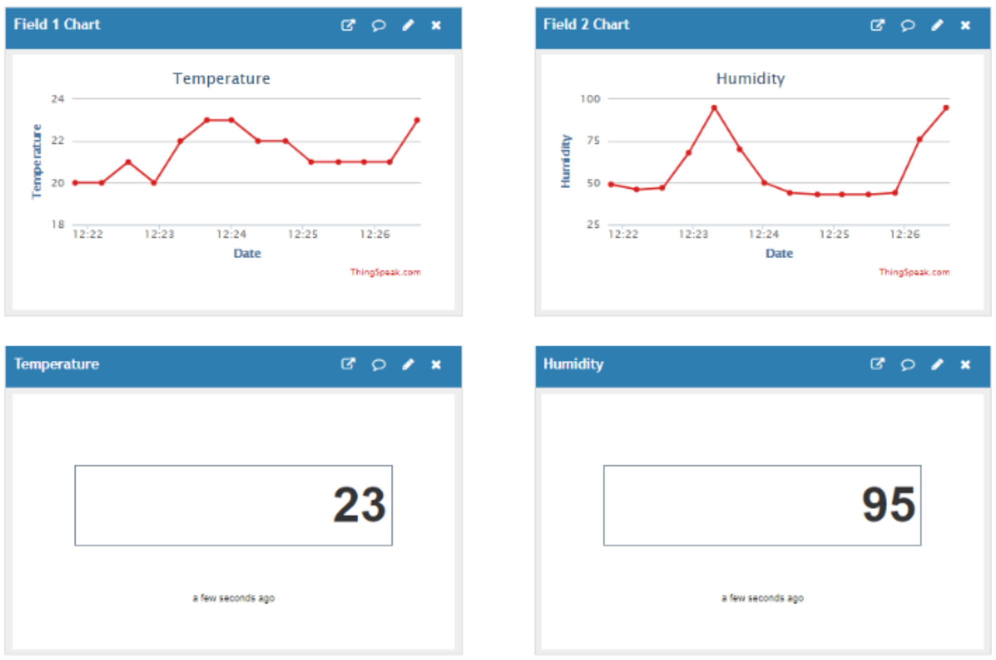

ESP8266
Rx
3.3V
GPIO0
RST
GPIO2
EN
GND
TX
GPIO0, GPIO2 and RST not used
Electronics-project-hub.com
3.3V
GND
11
10
2
DHT11
Sensor
NOTE: Please check pin diagram of DHT11 sensor. There are different pin configurations available.
UNO
Arduino
5V

Fig. 5 Interfacing of DHT11 & Arduino board



Data uploaded on Thingspeak Cloud

Procedure:

Step 1: Connect the Arduino board to the Micro-IoT Sensor board using the FRC cable provided with the board.

Step 2: Connect the Power supply adaptor and power on the circuit.

Step 3: Open Arduino IDE and create a new sketch (program) using the above pins.

Step 4: In the Arduino IDE go to tools→Port and select the appropriate COM port.

Step 5: In the Arduino IDE click on the upload button (⊙) to compile and download the code into the Arduino UNO. When successfully downloaded the code will start running.

Step 6: Connect the DHT11 sensor to CN8 (pin A1) with the Blue sensor side facing the buttons on the board.

Step 7: This experiment uses the ESP8266 Wi-Fi module. To connect the module to the circuit slide the switch SW6 between 1-2 (WiFi_ON) . Reset the ESP8266 by pressing the switch SW10. Now reset the Arduino board. Open the serial monitor @115200 baud rate and observe the output.

Step 8: Login to your Thingspeak account and check the data.

Observation:

The values for temperature and Humidity are displayed on the Thingspeak MQTT channel.

Conclusion:

Code:

```
#include "WiFiEsp.h"
#include "SoftwareSerial.h"
#include <PubSubClient.h>
#include <dht.h>

dht DHT;
SoftwareSerial Serial1(8, 7); // RX, TX

#define DHT11_PIN A1

char* wifi_ssid = "Microembedded";
char* wifi_password = "Micro@123";

char* mqtt_server = "mqtt3.thingspeak.com";
int mqtt_port = 1883;
char* mqtt_clientID = "HQYVLx8rCBApOyUxOAUwFjI";
char* mqtt_username = "HQYVLx8rCBApOyUxOAUwFjI";
char* mqtt_password = "Q1k4sz3hyM0QyurnfvKOoxuM";

char* mqtt_publish_topic = "channels/1852693/publish";

WiFiEspClient espClient;
PubSubClient client(mqtt_server,mqtt_port,espClient);
int status = WL_IDLE_STATUS;     // the Wifi radio's status

void setup_wifi() {

  // initialize serial for ESP module
  Serial1.begin(9600);
  // initialize ESP module
  WiFi.init(&Serial1);
   delay(10);
  // We start by connecting to a WiFi network
  // attempt to connect to WiFi network
  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(wifi_ssid);
    // Connect to WPA/WPA2 network
    status = WiFi.begin(wifi_ssid, wifi_password);
  }
  Serial.println("You're connected to the network");
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
```

```
 Serial.println(WiFi.localIP());

}

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

char msg[50];

void loop() {

static int counter = 0;
int chk = DHT.read11(DHT11_PIN);

  String payload="field1=";
  payload+=DHT.temperature;
  payload+="&field2=";
  payload+=DHT.humidity;
  payload+="&status=MQTTPUBLISH";

    if (client.connect(mqtt_clientID,mqtt_username,mqtt_password)) {
    Serial.println("Connected to MQTT broker");
    Serial.print("Topic is: ");
    Serial.println(mqtt_publish_topic);
    }

  if (client.connected()){
    Serial.print("Sending payload: ");
    Serial.println(payload);

    if (client.publish(mqtt_publish_topic, (char*) payload.c_str())) {
      Serial.println("Publish ok");
    }
    else {
      Serial.println("Publish failed");
    }
    client.disconnect();
  }
  ++counter;
  delay(20000);
}
```