MIT World Peace University

Internet of Things
Second Year B. Tech, Semester 2

MQTT Protocol Implementation using Tinkercad on Arduino UNO

Assignment 6

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

May 2, 2023

Contents

1 Aim

To understand the working of MQTT Protocol and implement it using Tinkercad on Arduino UNO.

2 Objectives

1. To understand the working of MQTT Protocol.
2. To use Arduino UNO to implement the MQTT Protocol.
3. To sense data from sensors and send it to cloud.

3 Equipments Used

Name	Quantity	Component
U1	1	Arduino Uno R3
U2	1	Temperature Sensor [TMP36]
U3	1	Wifi Module (ESP8266)
R1 R2	2	1 kΩ Resistor

4 Theory

Our project involves sensing the temperature using the TMP36 sensor and sending the data to the cloud using the ESP8266 module and MQTT protocol. This is an example of **sensing**.

4.1 MQTT

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol that is widely used in IoT (Internet of Things) applications for sending and receiving data between devices and the cloud. It uses a publish/subscribe messaging model, where devices (publishers) send messages to a central server (broker), which then distributes the messages to other devices (subscribers) that have subscribed to the topic of the message.

4.2 Connection

In our project, we use the ESP8266 module to connect to the Internet and publish the temperature data to a cloud platform called Thingspeak using the MQTT protocol. Thingspeak is a platform that allows users to collect, analyze, and visualize data from IoT devices. It provides an MQTT broker that can be used to publish and subscribe to data streams, as well as a web interface for displaying and analyzing the data in real-time.

To use MQTT in our project, we would need to first establish a connection to the broker using the ESP8266 module and then publish the temperature data to a specific topic on the broker. We could then use the Thingspeak API to retrieve the data from the broker and display it in a chart or graph on the Thingspeak web interface.

5 Circuit Diagram

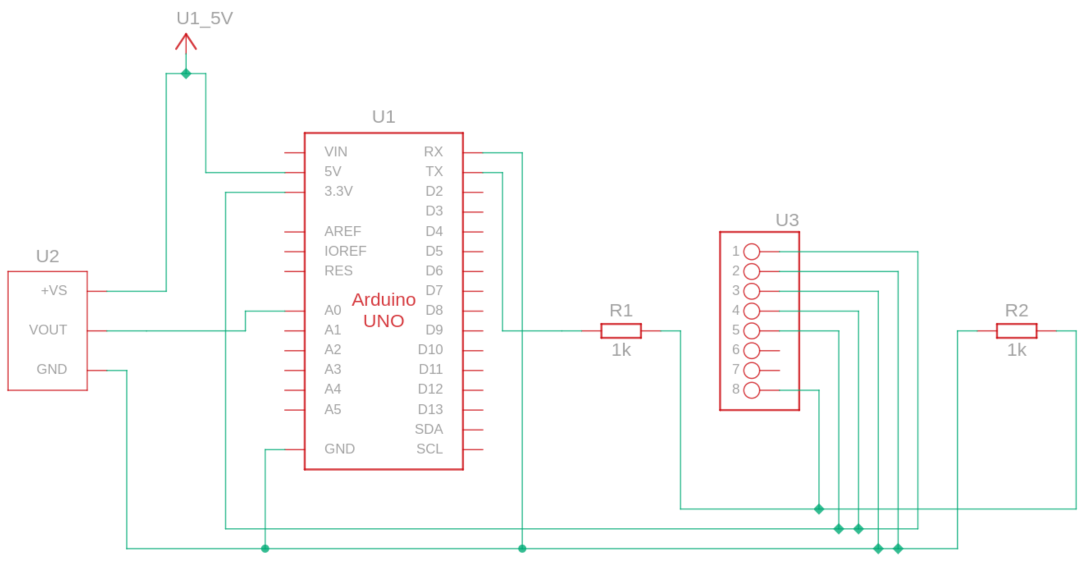

Figure 1: Circuit Diagram

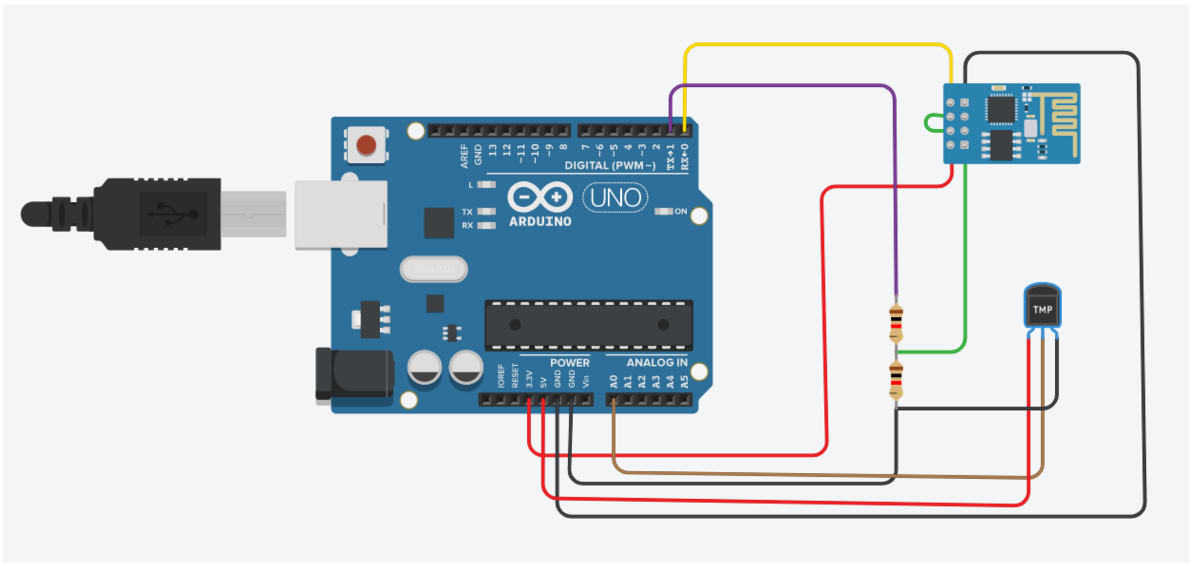

Figure 2: Circuit

6 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code and Tinkercad

7 Input

Data from the Temperature Sensor is sent to the cloud using the MQTT Protocol.

8 Output

```
CIPSEND=89
GET /update?api_key=SGXV201YIK6KJ2SC &field1=76.47 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=89
GET /update?api_key=SGXV201YIK6KJ2SC &field1=76.47 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=90
GET /update?api_key=SGXV201YIK6KJ2SC &field1=199.52 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=90
GET /update?api_key=SGXV201YIK6KJ2SC &field1=256.65 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=90
GET /update?api_key=SGXV201YIK6KJ2SC &field1=108.99 HTTP/1.1
Host: api.thingspeak.com

AT+CIPSEND=90
GET /update?api_key=SGXV201YIK6KJ2SC &field1=-40.42 HTTP/1.1
Host: api.thingspeak.com
```

9 Code

```
float val, voltage, temp,tempf;
String ssid     = "Simulator Wifi";  // SSID to connect to
String password = ""; //virtual wifi has no password
String host     = "api.thingspeak.com"; // Open Weather Map API
const int httpPort   = 80;
String url     = "/update?api_key=SGXV201YIK6KJ2SC";
String url1     = " &field1=";
//Replace XXXXXXXXXXXXXXXX by your ThingSpeak Channel API Key

void setupESP8266(void) {

  // Start our ESP8266 Serial Communication
  Serial.begin(115200);   // Serial connection over USB to computer
  Serial.println("AT");   // Serial connection on Tx / Rx port to ESP8266
  delay(10);        // Wait a little for the ESP to respond
  if (Serial.find("OK"))
  Serial.println("ESP8266 OK!!!");

  // Connect to Simulator Wifi
  Serial.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\"");



  delay(10);        // Wait a little for the ESP to respond
  if (Serial.find("OK"))
  Serial.println("Connected to WiFi!!!");

  // Open TCP connection to the host:
  //ESP8266 connects to the server as a TCP client.

  Serial.println("AT+CIPSTART=\"TCP\",\"" + host + "\"," + httpPort);
  delay(50);        // Wait a little for the ESP to respond
  if (Serial.find("OK"))
  Serial.println("ESP8266 Connected to server!!!") ;

}

void anydata(void) {

  val=analogRead(A0);
  voltage=val*0.0048828125;
  temp = (voltage - 0.5) * 100.0;
  tempf=(temp*9/5)+32;
  //distance=12;

  // Construct our HTTP call
  String httpPacket = "GET " + url  + url1 + String(tempf) + " HTTP/1.1\r\nHost: "
    + host + "\r\n\r\n";
  int length = httpPacket.length();

  // Send our message length
  Serial.print("AT+CIPSEND=");
  Serial.println(length);
  delay(10); // Wait a little for the ESP to respond if (!Serial.find(">")) return
    -1;

  // Send our http request
  Serial.print(httpPacket);
  delay(10); // Wait a little for the ESP to respond
  if (Serial.find("SEND OK\r\n"))
  Serial.println("ESP8266 sends data to the server");

}

void setup() {
  pinMode(A0, INPUT);
  setupESP8266();

}

void loop() {

  anydata();

  delay(100);
}
```

10 Conclusion

Thus, we have successfully sent data from the Temperature Sensor to the cloud using the MQTT Protocol.

11 FAQ

Details of the main components used in this project and their pin specifications:

1. Arduino Uno R3 (Code: U1)
   - Type: Microcontroller board
   - Digital pins: 14
   - Analog input pins: 6
   - Operating voltage: 5V
   - Input voltage (recommended): 7-12V
   - Input voltage (limits): 6-20V
   - Flash memory: 32 KB (ATmega328P microcontroller)
   - SRAM: 2 KB (ATmega328P microcontroller)
   - EEPROM: 1 KB (ATmega328P microcontroller)
   - Clock speed: 16 MHz (ATmega328P microcontroller)
2. Temperature Sensor [TMP36] (Code: U2)
   - Type: Analog sensor
   - Pin 1: Output voltage (analog)
   - Pin 2: Ground
   - Pin 3: Input voltage (3.3V - 5V)
   - Temperature range: -40°C to +125°C
   - Output voltage range: 0V to 1.75V (at 25°C)
3. Wifi Module [ESP8266] (Code: U3)
   - Type: Digital communication module
   - Pin 1: VCC (3.3V)
   - Pin 2: Ground
   - Pin 3: TXD (Transmit Data)
   - Pin 4: RXD (Receive Data)
   - Operating voltage: 3.3V
   - Communication protocol: UART
   - Wireless standard: IEEE 802.11 b/g/n
4. 1 kΩ Resistor (Codes: R1 and R2)
   - Type: Resistor
   - Resistance: 1 kΩ (1000 ohms)
   - Tolerance: 5%
   - Power rating: 1/4 watt
   - Quantity: 2