Getting started with Arduino ESP8266

← BACK TO GETTING STARTED

Prerequisites

Install Arduino IDE

To write code for an Arduino you first need to install the proprietary Arduino IDE. After installation, enable support for the ESP8266 chip, which provides the WiFi functionality. To do this, start the Arduino IDE and open the *Preferences* window.

Enter `https://arduino.esp8266.com/stable/package_esp8266com_index.json` into the `File>Preferences>Additional Boards Manager URLs` field of the Arduino IDE. You can add multiple URLs, by separating them with commas. Afterwards, open `Tools>Board>Boards Manager` and install the `esp8266` platform by searching for it. Then you can select your ESP8266 board in `Tools>Board` . In testing of this example, the Wemos D1 mini was used, but any board with an ESP8266 should work.

PubSubClient library

To use HiveMQ Cloud an MQTT library has to be added first. In this example the PubSubClient is used but there are a lot of other alternatives that provide similar functionalities. Download the PubSubClient from GitHub as a `.zip` file. Then go to the Arduino IDE, open `Sketch>Include Library>Add .ZIP Library` and select the downloaded `.zip` file.

NTPClient library

The next library that is required is the NTPClient. Install this library

Here it is used to update the internal clock of the ESP8266 board, in order to use TLS certificates.

LittleFS Filesystem Uploader

The LittleFS Filesystem Uploader is needed to upload certificates to your board. This enables you to connect securely with HiveMQ Cloud. Download the `ESP8266LittleFS-X.zip` . Go to the Arduino IDE directory and open the `tools` folder. Unzip the downloaded `.zip` folder to the `tools` folder. The result will be a folder called `ESP8266LittleFS-2.6.0` or it can have a different version. This folder contains the `ESP8266LittleFS` folder. Cut the `ESP8266LittleFS` folder and paste it into the `tools` folder. Afterwards, you can delete the `ESP8266LittleFS-2.6.0` folder. The resulting folder structure should look like this:

```
<home_dir>/Arduino-<version>/tools/ESP8266LittleFS/tc
```

On OS X create the `tools` directory in `~/Documents/Arduino/` and unpack the files there. Now restart Arduino IDE and `ESP8266 LittleFS Data Upload` will appear when opening the `tools` menu.

Upload certificates to the board

The next step is to get the certificates that are needed for an encrypted connection to HiveMQ Cloud. For this purpose use this script by downloading the repository as a `.zip` folder. Only the file `certs-from-mozilla.py` is needed, you can delete the rest. Open this script in a Python IDE of your choice, for example PyCharm or Visual Studio Code. Execute the script in your IDE. A file will be generated, that is called `certs.ar` . Move this file into the directory of your Arduino IDE sketch. To find it you can go to `Sketch>Show Sketch Folder` in Arduino IDE. The folder where your sketch is saved should open. Inside this folder, create a new folder called `data` . Put your generated `certs.ar` into this `data` folder. In the Arduino IDE, in the `Tools` menu, you may want to select a bigger flash size, this depends on the size of your files. Then open `Tools>ESP8266 LittleFS Data Upload` . This will upload the files located in `data` to your board’s flash memory. After a few seconds, you should get the message `LittleFS Image Uploaded` .

Getting started with Arduino ESP8

Clusters

Billing

Help

First add your WiFi information, so that the ESP8266 can connect to the internet. The `host name` of your cluster is already inserted as `mqtt_server`. Your `username` is also inserted automatically. To fully verify your credentials, replace the variable `"your_password"` with the value you entered when creating your credentials. As HiveMQ Cloud does not support insecure connections, TLS is required. A secure connection will be established by using the certificates we downloaded earlier. The default port used for secure MQTT connections is `8883`.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <time.h>
#include <TZ.h>
#include <FS.h>
#include <LittleFS.h>
#include <CertStoreBearSSL.h>

// Update these with values suitable for your network
const char* ssid = "your_wifi_name";
const char* password = "your_wifi_password";
const char* mqtt_server = "6dd678185e194e4ca3c7ffd5fe

// A single, global CertStore which can be used by al
// Needs to stay live the entire time any of the WiFi
// are present.
BearSSL::CertStore certStore;

WiFiClientSecure espClient;
PubSubClient * client;
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (500)
char msg[MSG_BUFFER_SIZE];
int value = 0;

void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
```

Feedback

Logout

Getting started with Arduino ESP8

Clusters

Billing

Help

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

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}


void setDateTime() {
  // You can use your own timezone, but the exact tim
  // Only the date is needed for validating the certi
  configTime(TZ_Europe_Berlin, "pool.ntp.org", "time.

  Serial.print("Waiting for NTP time sync: ");
  time_t now = time(nullptr);
  while (now < 8 * 3600 * 2) {
    delay(100);
    Serial.print(".");
    now = time(nullptr);
  }
  Serial.println();

  struct tm timeinfo;
  gmtime_r(&now, &timeinfo);
  Serial.printf("%s %s", tzname[0], asctime(&timeinfo
}


void callback(char* topic, byte* payload, unsigned in
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
```

Feedback

Logout

Getting started with Arduino ESP8

Clusters

Billing

Help

```
    digitalWrite(LED_BUILTIN, LOW); // Turn the LED o
    // but actually the LED is on; this is because
    // it is active low on the ESP-01)
    delay(500);
    digitalWrite(LED_BUILTIN, HIGH); // Turn the LED
  } else {
    digitalWrite(LED_BUILTIN, HIGH); // Turn the LED
  }
}


void reconnect() {
  // Loop until we’re reconnected
  while (!client->connected()) {
    Serial.print("Attempting MQTT connection…");
    String clientId = "ESP8266Client - MyClient";
    // Attempt to connect
    // Insert your password
    if (client->connect(clientId.c_str(), "iot_ptwsma
      Serial.println("connected");
      // Once connected, publish an announcement…
      client->publish("testTopic", "hello world");
      // … and resubscribe
      client->subscribe("testTopic");
    } else {
      Serial.print("failed, rc = ");
      Serial.print(client->state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}


void setup() {
  delay(500);
  // When opening the Serial Monitor, select 9600 Bau
  Serial.begin(9600);
  delay(500);

  LittleFS.begin();
```

Feedback

Logout

Getting started with Arduino ESP8

Clusters

Billing

Help

```
    pinMode(LED_BUILTIN, OUTPUT); // Initialize the LED

    // you can use the insecure mode, when you want to
    //espclient->setInsecure();

    int numCerts = certStore.initCertStore(LittleFS, PS
    Serial.printf("Number of CA certs read: %d\n", numC
    if (numCerts == 0) {
      Serial.printf("No certs found. Did you run certs-
      return; // Can't connect to anything w/o certs!
    }

    BearSSL::WiFiClientSecure *bear = new BearSSL::WiFi
    // Integrate the cert store with this connection
    bear->setCertStore(&certStore);

    client = new PubSubClient(*bear);

    client->setServer(mqtt_server, 8883);
    client->setCallback(callback);
  }

  void loop() {
    if (!client->connected()) {
      reconnect();
    }
    client->loop();

    unsigned long now = millis();
    if (now - lastMsg > 2000) {
      lastMsg = now;
      ++value;
      snprintf (msg, MSG_BUFFER_SIZE, "hello world #%ld
      Serial.print("Publish message: ");
      Serial.println(msg);
      client->publish("testTopic", msg);
    }
  }
```

Feedback

Logout

Run this example by connecting your board to your PC via a USB-cable and uploading the sketch with the button in the Arduino IDE.

and subscribing to a topic on your HiveMQ Cloud cluster. First it sets up the WiFi by using the SSID and password you set. Then the internal clock of the board is updated by getting the present time from an NTP server. This is necessary to validate the certificates that were uploaded to the board with the LittleFS Filesystem Uploader. For `callbacks` , `Message arrived` is registered. The LED of the board will also flash when a non-empty message arrives. After `setup_wifi()` and `setDateTime()` , the certificate store (`certStore`) is being initialized and made to store the certificates of `certs.ar` . Then the WiFi client will be integrated with the `certStore` . It sets up the MQTT client by using the earlier set host name and port. Afterwards it connects to the cluster using your username and password in the `reconnect()` function, that gets executed in a loop. It subscribes to the topic `"testTopic"` to which it also publishes. Then it publishes a message with the payload `"hello world"` and incrementing numbers to this topic.

Next Steps

Get familiar with the Arduino IDE API and build your first application. Further information on the PubSubClient for Arduino can be found in our MQTT Client Library Encyclopedia. Learn more about MQTT by visiting the MQTT Essentials guide, that explains the core of MQTT concepts, its features and other essential information.

← BACK TO GETTING STARTED