**EXPERIMENT 7**

**Objective:**

**Design an IOT system based on ESP32 using MQTT Protocol**

**ESP32 connects to the free public MQTT broker**

[MQTT](https://mqtt.org/) is a lightweight and flexible IoT message exchange and data transmission protocol, which is dedicated to achieving the balance between flexibility and hardware/network resources for IoT developers.

[ESP32](https://www.espressif.com/en/products/socs/esp32) is an upgraded version of ESP8266. In addition to the Wi-Fi module, this module also includes a Bluetooth 4.0 module. The dual-core CPU operates at a frequency of 80 to 240 MHz. It contains two Wi-Fi and Bluetooth modules and various input and output pins. ESP32 is an ideal choice for IoT projects.

In this project, we will connect ESP32 to the [free public MQTT broker](https://www.emqx.com/en/mqtt/public-mqtt5-broker) operated and maintained by EMQX [MQTT Cloud](https://www.emqx.com/en/cloud), and use the Arduino IDE to program the ESP32. EMQX Cloud is a secure MQTT IoT cloud service platform launched by [EMQ](https://www.emqx.com/en). It provides [MQTT 5.0](https://www.emqx.com/en/mqtt/mqtt5)access service with one-stop operation and maintenance management and a unique isolation environment.

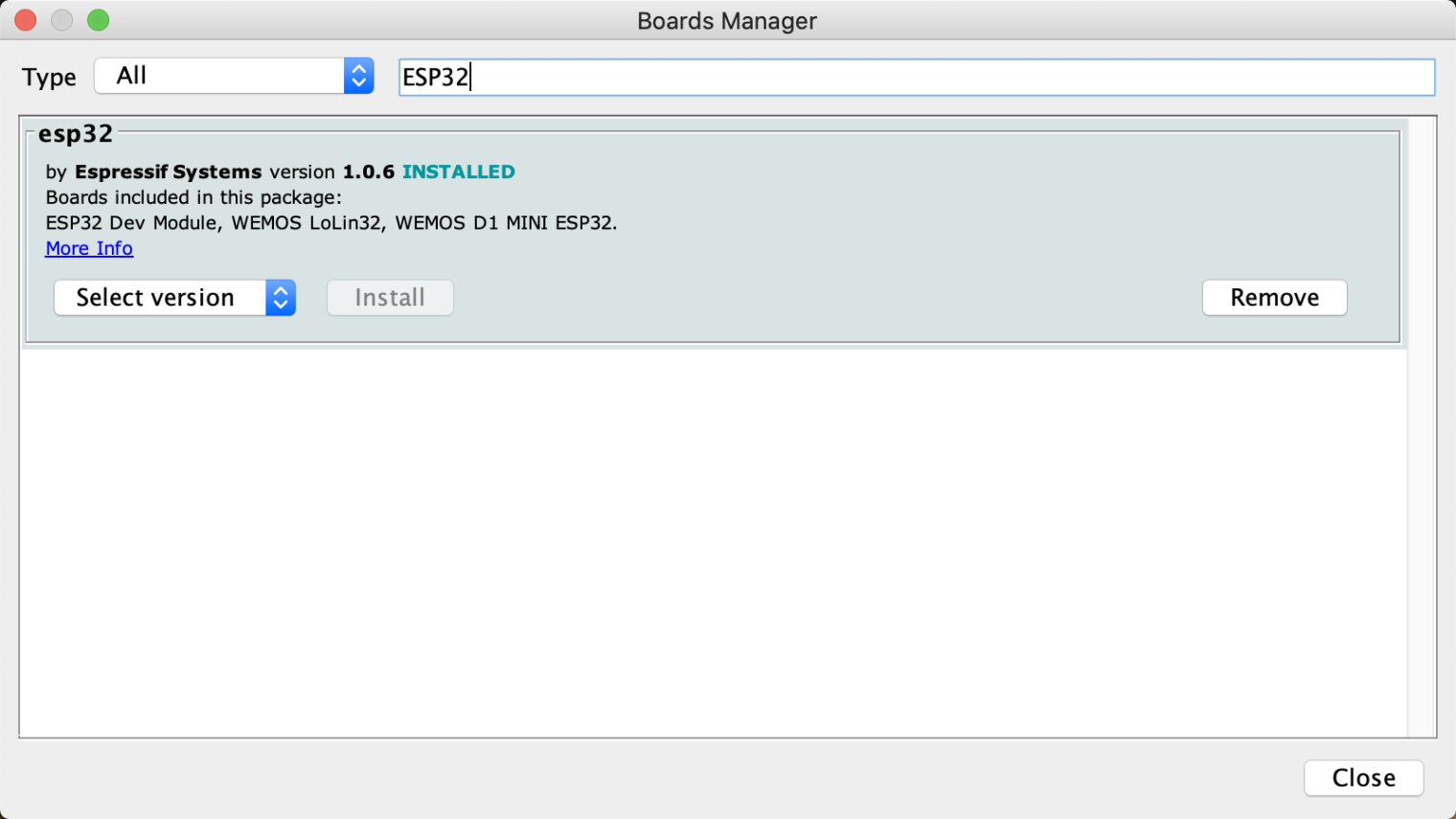
**Required IoT components**

* ESP32
* Arduino IDE
* [MQTT 5.0 client tool - MQTT X](https://mqttx.app/)
* A [free public MQTT broker](https://www.emqx.com/en/mqtt/public-mqtt5-broker) deployed on [EMQX Cloud](https://www.emqx.com/en/cloud)
  + Broker: **broker.emqx.io**
  + TCP Port: **1883**
  + Websocket Port: **8083**

**Arduino configuration**

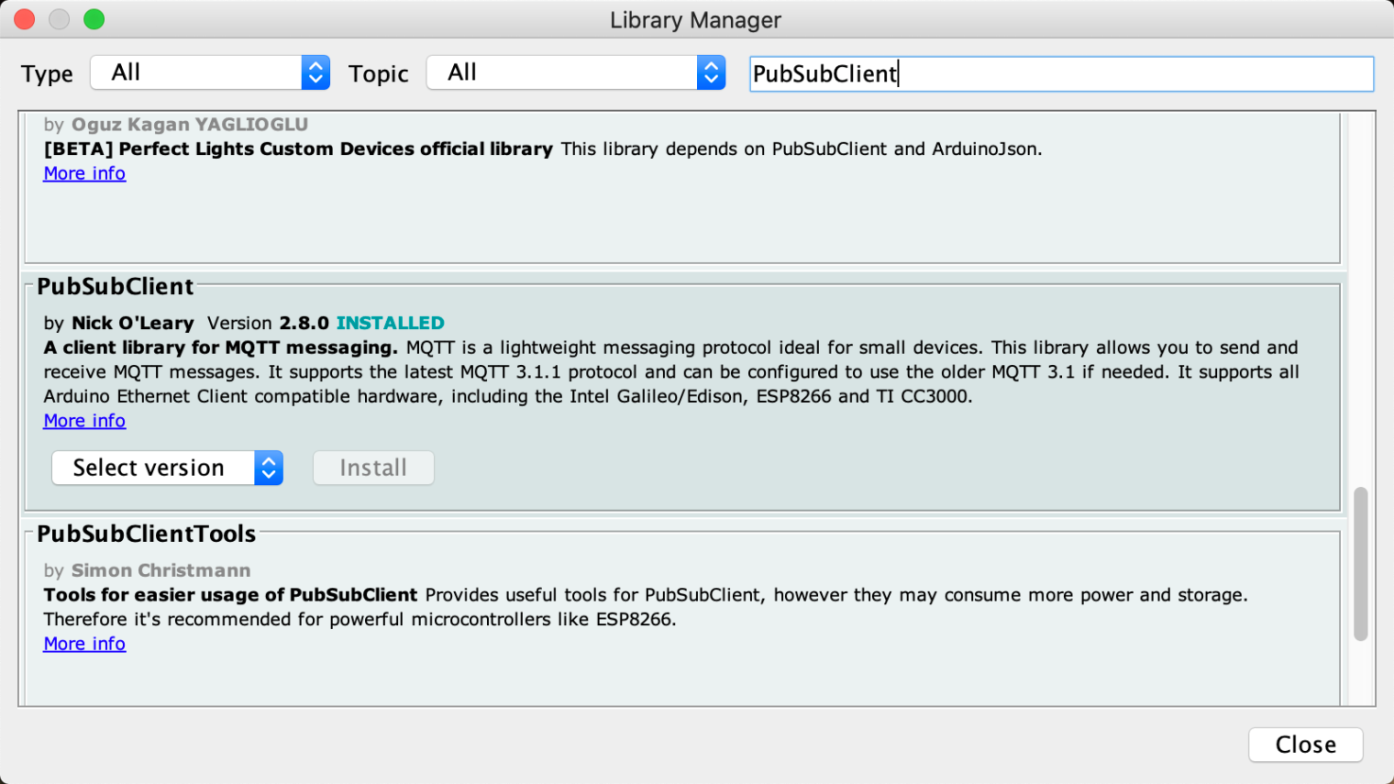
**Install ESP32 development board**

Click Tools -> Development Board -> Development Board Management -> Search ESP32 -> Install

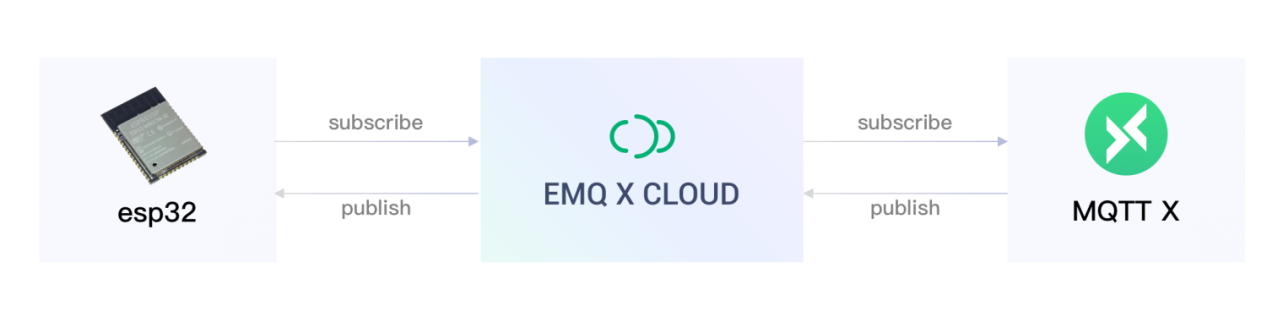


**Install PubSub client**

Project -> Load library -> Library manager... -> Search PubSubClient -> Install PubSubClient by Nick O’Leary



**ESP32 Pub/Sub diagram**



**Programming ESP32 Board with Arduino IDE**

**Connect to MQTT step by step**

1. First, we will import the **WiFi** and **PubSubClient** libraries. The ESP8266WiFi library can connect ESP32 to Wi-Fi networks, and the PubSubClient library can connect ESP32 to the MQTT server to publish messages and subscribe to topics.
2. #include <WiFi.h>
3. #include <PubSubClient.h>
4. Set the Wi-Fi name and password, as well as the MQTT server connection address and port, and set the topic to "esp32/test".
5. // WiFi
6. const char \*ssid = "mousse"; // Enter your WiFi name
7. const char \*password = "qweqweqwe"; // Enter WiFi password
8. // MQTT Broker
9. const char \*mqtt\_broker = "broker.emqx.io";
10. const char \*topic = "esp32/test";
11. const char \*mqtt\_username = "emqx";
12. const char \*mqtt\_password = "public";
13. const int mqtt\_port = 1883;
14. Open a serial connection to output the results of the program and connect to the Wi-Fi network.
15. // Set software serial baud to 115200;
16. Serial.begin(115200);
17. // connecting to a WiFi network
18. WiFi.begin(ssid, password);
19. while (WiFi.status() != WL\_CONNECTED) {
20. delay(500);
21. Serial.println("Connecting to WiFi..");
22. }
23. Use PubSubClient to connect to the [public MQTT broker](https://www.emqx.com/en/blog/popular-online-public-mqtt-brokers).
24. client.setServer(mqtt\_broker, mqtt\_port);
25. client.setCallback(callback);
26. while (!client.connected()) {
27. String client\_id = "esp32-client-";
28. client\_id += String(WiFi.macAddress());
29. Serial.printf("The client %s connects to the public mqtt broker\n", client\_id.c\_str());
30. if (client.connect(client\_id.c\_str(), mqtt\_username, mqtt\_password)) {
31. Serial.println("Public emqx mqtt broker connected");
32. } else {
33. Serial.print("failed with state ");
34. Serial.print(client.state());
35. delay(2000);
36. }
37. }
38. After the MQTT server is successfully connected, ESP32 will publish messages to the MQTT server of esp/test and subscribe to the topic messages of esp/test.
39. // publish and subscribe
40. client.publish(topic, "Hi EMQX I'm ESP32 ^^");
41. client.subscribe(topic);
42. Set the callback function to print the topic name to the serial port and print the message received from the esp32/test topic.
43. void callback(char \*topic, byte \*payload, unsigned int length) {
44. Serial.print("Message arrived in topic: ");
45. Serial.println(topic);
46. Serial.print("Message:");
47. for (int i = 0; i < length; i++) {
48. Serial.print((char) payload[i]);
49. }
50. Serial.println();
51. Serial.println("-----------------------");
52. }

**Full code**

#include <WiFi.h>

#include <PubSubClient.h>

// WiFi

const char \*ssid = "mousse"; // Enter your WiFi name

const char \*password = "qweqweqwe"; // Enter WiFi password

// MQTT Broker

const char \*mqtt\_broker = "broker.emqx.io";

const char \*topic = "esp32/test";

const char \*mqtt\_username = "emqx";

const char \*mqtt\_password = "public";

const int mqtt\_port = 1883;

WiFiClient espClient;

PubSubClient client(espClient);

void setup() {

// Set software serial baud to 115200;

Serial.begin(115200);

// connecting to a WiFi network

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.println("Connecting to WiFi..");

}

Serial.println("Connected to the WiFi network");

//connecting to a mqtt broker

client.setServer(mqtt\_broker, mqtt\_port);

client.setCallback(callback);

while (!client.connected()) {

String client\_id = "esp32-client-";

client\_id += String(WiFi.macAddress());

Serial.printf("The client %s connects to the public mqtt broker\n", client\_id.c\_str());

if (client.connect(client\_id.c\_str(), mqtt\_username, mqtt\_password)) {

Serial.println("Public emqx mqtt broker connected");

} else {

Serial.print("failed with state ");

Serial.print(client.state());

delay(2000);

}

}

// publish and subscribe

client.publish(topic, "Hi EMQX I'm ESP32 ^^");

client.subscribe(topic);

}

void callback(char \*topic, byte \*payload, unsigned int length) {

Serial.print("Message arrived in topic: ");

Serial.println(topic);

Serial.print("Message:");

for (int i = 0; i < length; i++) {

Serial.print((char) payload[i]);

}

Serial.println();

Serial.println("-----------------------");

}

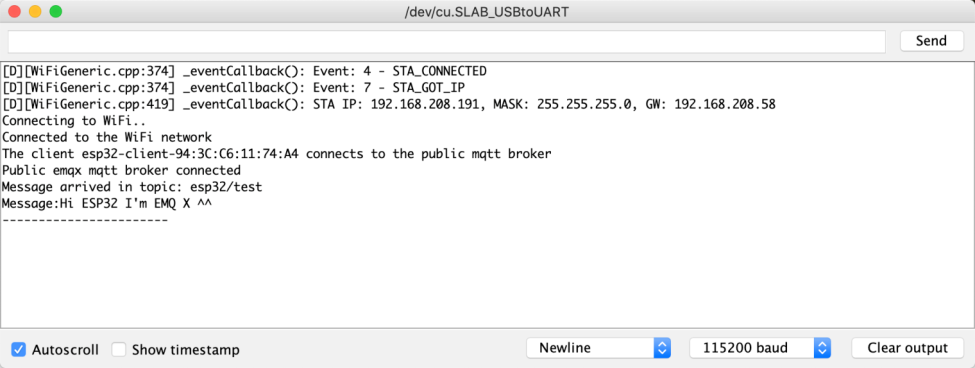
void loop() {

client.loop();

}

**Running and testing**

1. Use Arduino to upload the complete code and power on the esp32.
2. Open the serial monitor, select 115200 baud rate, and check the ESP32 connection status.



1. Use the MQTT X client to connect to the public MQTT server and publish messages to ESP32.

**Summary**

So far, we have successfully connected ESP32 to the public MQTT server provided by [EMQX Cloud](https://www.emqx.com/en/cloud). In this project, we simply connect ESP32 to the MQTT server. This is just one of ESP32's basic capabilities. ESP32 can actually connect to various IoT sensors and report sensor data to the MQTT server.

Next, you can check out [The Easy-to-understand Guide to MQTT Protocol](https://www.emqx.com/en/mqtt) series of articles provided by EMQ to learn about MQTT protocol features, explore more advanced applications of MQTT, and get started with MQTT application and service development.