




 knolleary / pubsubclient Public[Code](#) [Issues 459](#) [Pull requests 59](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)[New issue](#)[Jump to bottom](#)

## esp8266 TLS example #833

Open rforro wants to merge 2 commits into [knolleary:master](#) from [rforro:master](#) [Conversation 0](#) [Commits 2](#) [Checks 0](#) [Files changed 2](#)[Changes from all commits](#) [File filter](#) [Conversations](#) [Jump to](#) [187](#)  examples/mqtt\_esp8266\_tls\_auth/mqtt\_esp8266\_tls\_auth.ino 

```
...      ...      @@ -0,0 +1,187 @@
1 + /*
2 +  Basic ESP8266 MQTT over TLS example with authentication
3 +  This sketch demonstrates the capabilities of the pubsub library in combination
4 +  with the ESP8266 board/library.
5 +
6 +  Use this if you have CA certificate and you're using credentials for login.
7 +
8 +  It order to establish TLS connection with the mqtt server following steps are necessary:
9 +  - MQTT servers CA certificate has to be defined (use your own!)
10 +  - defined CA certificate has to be set
11 +  - time has to be obtained from NTP, because of CA expiration date validation
12 +  It connects to an MQTT server then:
13 +  - publishes "hello world" to the topic "outTopic" every two seconds
14 +  - subscribes to the topic "inTopic", printing out any messages
15 +  it receives. NB - it assumes the received payloads are strings not binary
16 +  - If the first character of the topic "inTopic" is an 1, switch ON the ESP Led,
17 +  else switch it off
18 +  It will reconnect to the server if the connection is lost using a blocking
19 +  reconnect function. See the 'mqtt_reconnect_nonblocking' example for how to
20 +  achieve the same result without blocking the main loop.
21 +  To install the ESP8266 board, (using Arduino 1.6.4+):
22 +  - Add the following 3rd party board manager under "File -> Preferences -> Additional Boards
    Manager URLs":
23 +      http://arduino.esp8266.com/stable/package_esp8266com_index.json
24 +  - Open the "Tools -> Board -> Board Manager" and click install for the ESP8266"
25 +  - Select your ESP8266 in "Tools -> Board"
26 + */
27 +
28 + // The hardcoded certificate authority for this example.
29 + // Don't use it on your own apps!!!!
```

```

30 + static const char ca_cert[] PROGMEM = R"EOF(
31 + -----BEGIN CERTIFICATE-----
32 + MIIC1TCCAb2gAwIBAgIJAMPt1Ms37+hLMA0GCSqGSIb3DQEBCwUAMCExCzAJBgNV
33 + BAYTA1VTMRlWEAYDVQDDAKxMjc4wLjMwHhcNMTgwMzE0MDQyMTU0WhcNMjkw
34 + NTMxMDQyMTU0WjAhMQswCQYDVQGEwJVUzESMBAGA1UEAwwJMTI3LjAuMC4zMIIIB
35 + IjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAXsa4qU/tlzn4YTcnn/I/ffsi
36 + jOPc8QRcwClKzasIZNFeye4uThl+LGZWFIfb8X8Dc+xmmBaWlPjBqtphgFKStpar
37 + DdduHSW1ud6Y1FVKxljo3UwCMrYm76Q/jNzXJvGs6Z1MDNsVZzGJaoqit2H2Hkvvk
38 + y+7kk3YbEDlcyVsL0w0zCKL4cd2DSNDyhIZxWo2a8Qn5IdjWAYtsTnW6MvLk/ya4
39 + abNeRfSZwi+r37rqi9CIs++NpL5ynqkKKEMrbeLactWgHbWrZeaMyLpuUEL2GF+w
40 + MRaAwaj7ERwT5gFJRqYwj6bbfIdx5PC7h7ucbyp272MbrDa6WNBcmwQO222t4wID
41 + AQABoxAwDjAMBGNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQCmXfrC42nW
42 + IpL3JDk8YlB2QUvd9JdMp98xxo33+xE69Gov0e6984F1Gluao0p6sS7KF+q3YLS
43 + 4hjnzuzGzF9GJmimIB7NMQ20yXKfKpmKJ7YugMakTDWdhHn5679mKVbLSQxHCUMee
44 + tEnMT93/UaDbWbJv6zu876q5vjPMYgDHODq0295ySaA71UkijaCn6UwKUT49286T
45 + V9ZtzgabNGHXfk1HgUPWoShyze+G3g29I1BR0qABoJI63zaNu8ua42v5g1Rldxsw
46 + X8yKI14mFOGxuvcygG8L2xxysW7Zq+9g+07gW0Pm6RDYnUQmIwY83h1KFctVCJdS
47 + 2PgozwwkUNyP
48 + -----END CERTIFICATE-----
49 + )EOF";
50 +
51 + #include <ESP8266WiFi.h>
52 + #include <PubSubClient.h>
53 +
54 + // Update these with values suitable for your network.
55 +
56 + const char* ssid = ".....";
57 + const char* password = ".....";
58 + const char* mqtt_server = ".....";
59 + const char* mqtt_user = ".....";
60 + const char* mqtt_password = ".....";
61 +
62 + BearSSL::WiFiClientSecure espClient;
63 + PubSubClient client(espClient);
64 + unsigned long lastMsg = 0;
65 + #define MSG_BUFFER_SIZE (50)
66 + char msg[MSG_BUFFER_SIZE];
67 + int value = 0;
68 +
69 + void setClock()
70 + {
71 +   configTime(3 * 3600, 0, "pool.ntp.org", "time.nist.gov");
72 +
73 +   Serial.print("Waiting for NTP time sync: ");
74 +   time_t now = time(nullptr);
75 +   while (now < 8 * 3600 * 2) {
76 +     delay(500);
77 +     Serial.print(".");
78 +     now = time(nullptr);
79 +   }
80 +   Serial.println("");
81 +   struct tm timeinfo;
82 +   gmtime_r(&now, &timeinfo);

```

```
83 + Serial.print("Current time: ");
84 + Serial.print(asctime(&timeinfo));
85 + }
86 +
87 + void setup_wifi() {
88 +
89 +     delay(10);
90 +     // We start by connecting to a WiFi network
91 +     Serial.println();
92 +     Serial.print("Connecting to ");
93 +     Serial.println(ssid);
94 +
95 +     WiFi.mode(WIFI_STA);
96 +     WiFi.begin(ssid, password);
97 +
98 +     while (WiFi.status() != WL_CONNECTED) {
99 +         delay(500);
100 +         Serial.print(".");
101 +     }
102 +
103 +     randomSeed(micros());
104 +
105 +     Serial.println("");
106 +     Serial.println("WiFi connected");
107 +     Serial.println("IP address: ");
108 +     Serial.println(WiFi.localIP());
109 + }
110 +
111 + void callback(char* topic, byte* payload, unsigned int length) {
112 +     Serial.print("Message arrived [");
113 +     Serial.print(topic);
114 +     Serial.print("] ");
115 +     for (int i = 0; i < length; i++) {
116 +         Serial.print((char)payload[i]);
117 +     }
118 +     Serial.println();
119 +
120 +     // Switch on the LED if an 1 was received as first character
121 +     if ((char)payload[0] == '1') {
122 +         digitalWrite(BUILTIN_LED, LOW); // Turn the LED on (Note that LOW is the voltage level
123 +         // but actually the LED is on; this is because
124 +         // it is active low on the ESP-01)
125 +     } else {
126 +         digitalWrite(BUILTIN_LED, HIGH); // Turn the LED off by making the voltage HIGH
127 +     }
128 +
129 + }
130 +
131 + void reconnect() {
132 +     char err_buf[256];
133 +
134 +     // Loop until we're reconnected
135 +     while (!client.connected()) {
```

```
136 + Serial.print("Attempting MQTT connection...");
137 + // Create a random client ID
138 + String clientId = "ESP8266Client-";
139 + clientId += String(random(0xffff), HEX);
140 + // Attempt to connect
141 + if (client.connect(clientId.c_str(), mqtt_user, mqtt_password)) {
142 +     Serial.println("connected");
143 +     // Once connected, publish an announcement...
144 +     client.publish("outTopic", "hello world");
145 +     // ... and resubscribe
146 +     client.subscribe("inTopic");
147 + } else {
148 +     Serial.print("failed, rc=");
149 +     Serial.println(client.state());
150 +     espClient.getLastSSLError(err_buf, sizeof(err_buf));
151 +     Serial.print("SSL error: ");
152 +     Serial.println(err_buf);
153 +     Serial.println(" try again in 5 seconds");
154 +     // Wait 5 seconds before retrying
155 +     delay(5000);
156 + }
157 + }
158 + }
159 +
160 + void setup() {
161 +     pinMode(BUILTIN_LED, OUTPUT); // Initialize the BUILTIN_LED pin as an output
162 +     Serial.begin(115200);
163 +     BearSSL::X509List *serverTrustedCA = new BearSSL::X509List(ca_cert);
164 +     espClient.setTrustAnchors(serverTrustedCA);
165 +     setup_wifi();
166 +     setClock(); // Required for X.509 validation
167 +     client.setServer(mqtt_server, 8883);
168 +     client.setCallback(callback);
169 + }
170 +
171 + void loop() {
172 +
173 +     if (!client.connected()) {
174 +         reconnect();
175 +     }
176 +     client.loop();
177 +
178 +     unsigned long now = millis();
179 +     if (now - lastMsg > 2000) {
180 +         lastMsg = now;
181 +         ++value;
182 +         snprintf (msg, MSG_BUFFER_SIZE, "hello world #%ld", value);
183 +         Serial.print("Publish message: ");
184 +         Serial.println(msg);
185 +         client.publish("outTopic", msg);
186 +     }
187 + }
```



242 examples/mqtt\_esp8266\_tls\_certs/mqtt\_esp8266\_tls\_certs.ino

```
...
...  @@ -0,0 +1,242 @@
1  + /*
2  +   Basic ESP8266 MQTT over TLS example with client certificate authentication
3  +   This sketch demonstrates the capabilities of the pubsub library in combination
4  +   with the ESP8266 board/library.
5  +
6  +   Use this if you have CA certificate, client certificate and client key.
7  +
8  +   In order to establish TLS connection with the mqtt server following steps are necessary:
9  +   - MQTT servers CA certificate has to be defined (use your own!)
10 +   - client certificate and client key obtained from MQTT server have to be defined (use your
    own!)
11 +   - both certificates and the key need to be set
12 +   - time has to be obtained from NTP, because of CA expiration date validation
13 +   It connects to an MQTT server then:
14 +   - publishes "hello world" to the topic "outTopic" every two seconds
15 +   - subscribes to the topic "inTopic", printing out any messages
16 +     it receives. NB - it assumes the received payloads are strings not binary
17 +   - If the first character of the topic "inTopic" is an 1, switch ON the ESP Led,
18 +     else switch it off
19 +   It will reconnect to the server if the connection is lost using a blocking
20 +   reconnect function. See the 'mqtt_reconnect_nonblocking' example for how to
21 +   achieve the same result without blocking the main loop.
22 +   To install the ESP8266 board, (using Arduino 1.6.4+):
23 +   - Add the following 3rd party board manager under "File -> Preferences -> Additional Boards
    Manager URLs":
24 +       http://arduino.esp8266.com/stable/package_esp8266com_index.json
25 +   - Open the "Tools -> Board -> Board Manager" and click install for the ESP8266"
26 +   - Select your ESP8266 in "Tools -> Board"
27 + */
28 +
29 + // The hardcoded certificate authority for this example.
30 + // Don't use it on your own apps!!!!
31 + const char client_private_key[] PROGMEM = R"EOF(
32 + -----BEGIN RSA PRIVATE KEY-----
33 + MIIeowIBAACAQEAAsRNVtVqP++YU8NrbXwE83xVsDqcB3F76xcXNKFDERfVd2P/
34 + LvyDovCcoQtT0UCRgPcxRp894EuPH/Ru6Z2Lu85sV//i7ce27tc2WRFsfuh1RxHP
35 + LJWHxTl1CEfXp/owkECQ4MB3pw6Ekcl6iTEPiezTG+T+mQ/BkiIwcIK6CmlpR9DI
36 + eYUTqv0f9NrUfAjdBrqlEO2gpgFvLFrkDEU2ntAic4aPOP7yDOym/xzfy6TiG8Wo
37 + 7nlh6M97xTZGfbEPCH9rZDjo5istym1HzF5P+COq+OTSPscjFGXoi978o6hZwa7i
38 + zxorg4h5a5lGnshRu2G1+Ybfa140wnIrv/yCswIDAQAABaoIBAHxwgbshHCriTcEoY
39 + Yx6F0VTRq6ydA5mXfuYvS/eIfIE+pp1IgMScYEXZobjrJPQg1CA1l0NyFSHS97oV
40 + JPy34sMQxcLx6KABgeVHCMJ/EeJtnv7a3SUP0GIhhsVS95Ls18RIG4hWub+EzFVK
41 + eZqAB9N9wr4Pp3wZPodbz37B38rb1QPyMFmQ0LLHjKT0moxsXhL2ot+R3+aLYSur
42 + oP01kQo7/d0UAZoy8h90QN4a2EXvawh402EvFGbc5X/yXwAdEQ4Npp9VZhknIRkV
43 + +XZ3FcIqEvOploKtRF/tvBTz3g61/lFz21L9PMmV5y8tvSafr2SpJugGVmp2rrVQ
44 + VNYglIECgYEA10JSI5gmeCU3zK6kv0fBp54hY/5dDrSUPjKkMxpm7WZQ6I1/k7A
45 + hMcLeMzHiriT7WhRIXF8AOr2MoEkHkH3DhVNN4ccieVZx2SE5P5mVkItZGLrrpFU
46 + dysR/ARAI1HYegGuiKacZtf9SrRavU0m7fOV0iYwbFRhjyX+MyuteYkCgYEA0pbz
47 + 4ZosetScP68uZx1sG1TfkcqL17i15DHk3gnj6jK1fhvC2MjeLMhNDtKeUAuY7rLQ
```

```

48 + guZ0CCghWAv0G1h5eYdfIiPhgqFfX4P5F30m4zQHVPYj8xHfHG4ZP7dKQTndr01Q
49 + fLDGDTQLVXabAUSp2YGrIjC8J9idSW1pYClvF1sCgYEAjkDn41nzYkbGP1/Swnwu
50 + AEWCL4Czoro32jVxScxSrugt5wJLNWp508VukWBTJhugtq3Pn9hNaJXeKbYqVky1
51 + pgrxwpZph7+nuxt0r5hnr02C7eppcjIoWLB/7BorAKxf8REGReBFT7nBTBMwPBW2
52 + e14U6h6+tXh2GJG1Eb/1nnECgYAYdVb0THOx7rWNkNUGggc/++why61M6kYy6j2T
53 + cj05BW+f2tkCBoctpcTI83BZb53y08g4RS2yMqNirGKN2XspwmTqEjzbhv0Klt4F
54 + X4GyW0oU0nFksXiLiFpOaQWSwWG7KJWrfGJ9kWXR0Xsf15QLoDCuNCsn3t4d43T
55 + K7ph1wKBgHDzF+50+/Wez3YHCy2a/Hg5bHCpLQjknvgwkOh1z7YitYBUM72HP8Z
56 + Ge6b4wEfNuBdlZ1l/y9BQQOZJLFvJTE5t51X9klrkGrOb+Ftwr7eI/H5xgcadI52
57 + tPYglR5fjuRF/wnt3oX9JlQ2RtSbs+3naXH8JoherHaqNn8UpH0t
58 + -----END RSA PRIVATE KEY-----
59 + )EOF";
60 +
61 + const char client_cert[] PROGMEM = R"EOF(
62 + -----BEGIN CERTIFICATE-----
63 + MIIDTzCCAjaCCQDPXvMRYOpEuDANBgkqhkiG9w0BAQsFADCBpjESMBAGA1UEAwWJ
64 + MTI3LjAuMC4xMQswCQYDVQGEwJVUzE1MCMGA1UECgwcTXkgT3duIENlcnRpZm1j
65 + YXR1IEF1dGhvcml0eTEUMBIGA1UECAwLQXJkdWlub0xhbmQxTATBgNVBACMDEFy
66 + ZHVpbm9waWxsZTEVMBMGA1UECgMRVnQODI2NlVzZXJzMRGwFgYDVQQLDA9FU1A4
67 + MjY2LUFyZHVpbm8wHhcNMTgwMzE0MDQwMDAwWhcNMjkwMjI0MDQwMDAwWjAsMRYw
68 + FAYDVQQKDA1NeSBTZXJ2ZXIgt3JnMRIwEAYDVQQDDAkxMjcUMC4wLjMwggEiMA0G
69 + CSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCxE1V0+o/75hSHw2ttfATzfFwWOpwH
70 + cXvrFxc0oUMRF9V3Y/8u/IOi8JyhC1PRQJGA9zFGnz3gS48f9G7pnYu7zmxX/+Lt
71 + x7bu1zZZEVJ+6GVHEc8s1YfFOXUIR9en+jCQQJDgwHenDoSRzXqJMQ+J7Nb5P6Z
72 + D8GSIjBwgroIyWlH0Mh5hR0q/R/02tR8CN0GuqUQ7aCmAw8sWuQMRTae0Ahzho84
73 + /vIM7Kb/HN/LpOIbxaJueWHoz3vFNkZ9sQ8If2tk00jmKy3KbUfMXk/4I6r45NI+
74 + xyMUZeil3vyjQFnBruLPGiuDiHlrmUaeyFG7YaX5ht9rXg7Cciu//IKzAgMBAAEw
75 + DQYJKoZIhvcNAQELBQADggEBAEnG+FNYNCokBvzHiUPHHpScxZqM2f+XDcewJgeS
76 + L6HkYEDIZZDnd5gduSvkHpdJtWgsvJ7dJZL40w7Ba5sxpZHPiGkJG19hzMkG+aA
77 + z5GMkjys9h2xpQZx9KL3q7G6A+C0b1170DZ1wBtY07CFMykT4Mp2oMRrQKRucMSV
78 + AB1mKujLanMRKJ3NM89RQJH4GYiRps9y/HvM51h7EIK/J0/nEZeJxY5hJngskPKb
79 + oPPdmkR97kaQn114KNsC3owVlHVU2fMftgYkgQLzyeWgzcnA39AF3B6J1c0zNyQY
80 + seoK24dHmt6tWmn/sbxX7Aa6TL/4mVlFoOgcaTJyVaY/BrY=
81 + -----END CERTIFICATE-----
82 + )EOF";
83 +
84 + static const char ca_cert[] PROGMEM = R"EOF(
85 + -----BEGIN CERTIFICATE-----
86 + MIIC1TCCAb2gAwIBAgIJAMPT1Ms37+hLMA0GCSqGSIb3DQEBCwUAMCEXCzAJBgNV
87 + BAYTA1VTMRIwEAYDVQQDDAkxMjcUMC4wLjMwHhcNMTgwMzE0MDQwMTU0WhcNMjkw
88 + NTMxMDQyMTU0WjAhMQswCQYDVQGEwJVUzESMBAGA1UEAwWJMTI3LjAuMC4zMIIIB
89 + IjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAXsa4qU/tlzn4YTcnI/ffsi
90 + jOPc8QRcwClKzasIZNFeye4uTh1+LGZWFIFb8X8Dc+xxmBaW1PjBqtphgFKStpar
91 + DdduHSW1ud6Y1FVKx1jo3UwCmrYm76Q/jNzXJvGs6Z1MDNsVZzGJaoqit2H2Hkvk
92 + y+7kk3YbEDlcyVsL0w0zCKL4cd2DSNDyhIZxWo2a8Qn5IdjWAYtsTnW6MvLk/ya4
93 + abNeRfSZwi+r37rqi9CIs++NpL5ynqkKKEmrbeLactWgHbWrZeaMyLpuUEL2GF+w
94 + MRaAwaj7ERwT5gFJRqYwj6bbfIdx5PC7h7ucbyp272MbrDa6WNBcmwQO222t4wID
95 + AQABoxAwdJAMBGNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQCmXfrC42nW
96 + IpL3JDKb8Y1B2QUvd9JdMp98xxo33+xE69Gov0e6984F1Gluao0p6sS7KF+q3YLS
97 + 4hjnzuzGzF9GJMimIB7NMQ20yXKfKpmKJ7YugMakTDWdHn5679mKVbLSQxHCUMee
98 + tEnMT93/UadbWBJv6zu876q5vjPMYgDHODq0295ySaA71UkijaCn6UwKUT49286T
99 + V9ZtzgabNGHXfk1HgUPWoShyze+G3g29I1BR0qABoJI63zaNu8ua42v5g1Rldxsw
100 + X8yKI14mFOGxuvcygG8L2xxysW7Zq+9g+07gW0Pm6RDYnUQmIwY83h1KFctYCJdS

```

```
101 + 2PgozwkkUNyP
102 + -----END CERTIFICATE-----
103 + )EOF";
104 +
105 + #include <ESP8266WiFi.h>
106 + #include <PubSubClient.h>
107 +
108 + // Update these with values suitable for your network.
109 +
110 + const char* ssid = ".....";
111 + const char* password = ".....";
112 + const char* mqtt_server = ".....";
113 +
114 + BearSSL::WiFiClientSecure espClient;
115 + PubSubClient client(espClient);
116 + unsigned long lastMsg = 0;
117 + #define MSG_BUFFER_SIZE (50)
118 + char msg[MSG_BUFFER_SIZE];
119 + int value = 0;
120 +
121 + void setClock()
122 + {
123 +   configTime(3 * 3600, 0, "pool.ntp.org", "time.nist.gov");
124 +
125 +   Serial.print("Waiting for NTP time sync: ");
126 +   time_t now = time(nullptr);
127 +   while (now < 8 * 3600 * 2) {
128 +     delay(500);
129 +     Serial.print(".");
130 +     now = time(nullptr);
131 +   }
132 +   Serial.println("");
133 +   struct tm timeinfo;
134 +   gmtime_r(&now, &timeinfo);
135 +   Serial.print("Current time: ");
136 +   Serial.print(asctime(&timeinfo));
137 + }
138 +
139 + void setup_wifi() {
140 +
141 +   delay(10);
142 +   // We start by connecting to a WiFi network
143 +   Serial.println();
144 +   Serial.print("Connecting to ");
145 +   Serial.println(ssid);
146 +
147 +   WiFi.mode(WIFI_STA);
148 +   WiFi.begin(ssid, password);
149 +
150 +   while (WiFi.status() != WL_CONNECTED) {
151 +     delay(500);
152 +     Serial.print(".");
153 +   }
```

```
154 +
155 +   randomSeed(micros());
156 +
157 +   Serial.println("");
158 +   Serial.println("WiFi connected");
159 +   Serial.println("IP address: ");
160 +   Serial.println(WiFi.localIP());
161 + }
162 +
163 + void callback(char* topic, byte* payload, unsigned int length) {
164 +   Serial.print("Message arrived [");
165 +   Serial.print(topic);
166 +   Serial.print("] ");
167 +   for (int i = 0; i < length; i++) {
168 +     Serial.print((char)payload[i]);
169 +   }
170 +   Serial.println();
171 +
172 +   // Switch on the LED if an 1 was received as first character
173 +   if ((char)payload[0] == '1') {
174 +     digitalWrite(BUILTIN_LED, LOW); // Turn the LED on (Note that LOW is the voltage level
175 +     // but actually the LED is on; this is because
176 +     // it is active low on the ESP-01)
177 +   } else {
178 +     digitalWrite(BUILTIN_LED, HIGH); // Turn the LED off by making the voltage HIGH
179 +   }
180 +
181 + }
182 +
183 + void reconnect() {
184 +   char err_buf[256];
185 +
186 +   // Loop until we're reconnected
187 +   while (!client.connected()) {
188 +     Serial.print("Attempting MQTT connection...");
189 +     // Create a random client ID
190 +     String clientId = "ESP8266Client-";
191 +     clientId += String(random(0xffff), HEX);
192 +     // Attempt to connect
193 +     if (client.connect(clientId.c_str())) {
194 +       Serial.println("connected");
195 +       // Once connected, publish an announcement...
196 +       client.publish("outTopic", "hello world");
197 +       // ... and resubscribe
198 +       client.subscribe("inTopic");
199 +     } else {
200 +       Serial.print("failed, rc=");
201 +       Serial.println(client.state());
202 +       espClient.getLastSSLError(err_buf, sizeof(err_buf));
203 +       Serial.print("SSL error: ");
204 +       Serial.println(err_buf);
205 +       Serial.println(" try again in 5 seconds");
206 +       // Wait 5 seconds before retrying
```



```
207 +     delay(5000);
208 + }
209 + }
210 + }
211 +
212 + void setup() {
213 +     pinMode(BUILTIN_LED, OUTPUT);    // Initialize the BUILTIN_LED pin as an output
214 +     Serial.begin(115200);
215 +     BearSSL::X509List *serverTrustedCA = new BearSSL::X509List(ca_cert);
216 +     BearSSL::X509List *serverCertList = new BearSSL::X509List(client_cert);
217 +     BearSSL::PrivateKey *serverPrivKey = new BearSSL::PrivateKey(client_private_key);
218 +     espClient.setTrustAnchors(serverTrustedCA);
219 +     espClient.setClientRSACert(serverCertList, serverPrivKey);
220 +     setup_wifi();
221 +     setCLOCK(); // Required for X.509 validation
222 +     client.setServer(mqtt_server, 8883);
223 +     client.setCallback(callback);
224 + }
225 +
226 + void loop() {
227 +
228 +     if (!client.connected()) {
229 +         reconnect();
230 +     }
231 +     client.loop();
232 +
233 +     unsigned long now = millis();
234 +     if (now - lastMsg > 2000) {
235 +         lastMsg = now;
236 +         ++value;
237 +         snprintf (msg, MSG_BUFFER_SIZE, "hello world #%ld", value);
238 +         Serial.print("Publish message: ");
239 +         Serial.println(msg);
240 +         client.publish("outTopic", msg);
241 +     }
242 + }
```

