Example Codes

WiFi Events

The ESP32 WiFi library provides several events that allow you to monitor the WiFi connection status and respond to changes in network conditions. These events can be handled using event handlers in your code to manage WiFi connections more effectively.

ARDUINO_EVENT_WIFI_STA_GOT_IP

- Triggered when the ESP32 station got IP from connected AP

```
#include <WiFi.h>

const char* ssid = "ssid";
const char* password = "password";

void WiFiGotIP(WiFiEvent_t event, WiFiEventInfo_t info){
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

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

    WiFi.onEvent(WiFiGotIP, WiFiEvent_t::ARDUINO_EVENT_WIFI_STA_GOT_IP);

    WiFi.begin(ssid, password);

    Serial.println();
    Serial.println();
    Serial.println("Wait for WiFi... ");
}
```

```
void loop(){
    delay(1000);
}
```

NTP Server

The ESP32 can be configured to fetch the current time and date from NTP servers over the internet. This is particularly useful in projects requiring accurate timestamps, scheduling, or clock synchronization. With the built-in WiFi capabilities of the ESP32, connecting to an NTP server is simple.

Using the configTime() function, the ESP32 can communicate with an NTP server to get the current time and date. This function takes the timezone offset and NTP server URLs as parameters. Once configured, the ESP32 will maintain the time internally, even if it disconnects from WiFi.

```
#include <WiFi.h>
#include <time.h>

const char* ssid     = "ssid";
const char* password = "password";

const char* ntpServer = "id.pool.ntp.org";
const long  gmtOffset_sec = 25200; // GMT offset in seconds (e.g., +25200 for WIB because 7 * 3600)

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

    // Connect to Wi-Fi
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected.");

    // Init and get the time
```

```
    configTime(gmtOffset_sec, 0, ntpServer);
    printLocalTime();
}

void loop(){
    delay(1000);
    printLocalTime();
}

void printLocalTime(){
    struct tm timeinfo;
    if(!getLocalTime(&timeinfo)){
        Serial.println("Failed to obtain time");
        return;
    }
    Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
    Serial.print("Day of week: ");
    Serial.println(&timeinfo, "%A");
    Serial.print("Month: ");
    Serial.println(&timeinfo, "%B");
    Serial.print("Day of Month: ");
    Serial.println(&timeinfo, "%d");
    Serial.print("Year: ");
    Serial.println(&timeinfo, "%Y");
    Serial.print("Hour: ");
    Serial.println(&timeinfo, "%H");
    Serial.print("Hour (12 hour format): ");
    Serial.println(&timeinfo, "%I");
    Serial.print("Minute: ");
    Serial.println(&timeinfo, "%M");
    Serial.print("Second: ");
    Serial.println(&timeinfo, "%S");
}
```

HTTPClient

The HTTPClient library on the ESP32 enables you to perform HTTP requests (GET, POST, PUT, DELETE) with ease. This is especially useful for applications where the ESP32 interacts with web servers, APIs, or cloud services over HTTP or HTTPS. The library provides a straightforward API for

making HTTP requests, handling responses, and managing errors.

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <HTTPClient.h>

const char ssid[] = "ssid";
const char password[] = "password";
const char *server_cert = R"(-----BEGIN CERTIFICATE-----
Get the server's root CA by running the command below

openssl s_client -connect {Host}:{Port} -showcerts
-----END CERTIFICATE-----)";

const char *client_cert = R"(-----BEGIN CERTIFICATE-----
Generate client certificates by running the command below

openssl req -newkey rsa:2048 -nodes -keyout client_key.pem -x509 -days 365 -out client_cert.pem

Fill client_cert with client_cert.pem
-----END CERTIFICATE-----)";

const char *client_key = R"(-----BEGIN PRIVATE KEY-----
Fill client_key with client_key.pem
-----END PRIVATE KEY-----)";

void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);
    Serial.printf("Connecting to WiFi with SSID : %s\n", ssid);
    while(!WiFi.isConnected());
    Serial.println("Connection succesful");

    delay(1000);
}

void loop() {
    WiFiClientSecure *client = new WiFiClientSecure;
```

```
if(client) {
    client->setCACert(server_cert);
    client->setCertificate(client_cert);
    client->setPrivateKey(client_key);

    {
        // Add a scoping block for HTTPClient https to make sure it is destroyed before WiFiClientSecure *client is

        HTTPClient https;

        Serial.print("[HTTPS] begin...\n");
        if (https.begin(*client, "https://httpbin.org/get")) {  // HTTPS
            Serial.print("[HTTPS] GET...\n");
            // start connection and send HTTP header
            int httpCode = https.GET();

            // httpCode will be negative on error
            if (httpCode > 0) {
            // HTTP header has been send and Server response header has been handled
                Serial.printf("[HTTPS] GET... code: %d\n", httpCode);

                // file found at server
                if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY) {
                    String payload = https.getString();
                    Serial.println(payload);
                }
            } else {
                Serial.printf("[HTTPS] GET... failed, error: %s\n", https.errorToString(httpCode).c_str());
            }
            https.end();
        } else {
            Serial.printf("[HTTPS] Unable to connect\n");
        }

        // End extra scoping block
    }

    delete client;
} else {
```

```
        Serial.println("Unable to create client");
    }

    Serial.println();
    Serial.println("Waiting 10s before the next round...");
    delay(10000);
}
```

ArduinoJSON

ArduinoJSON is a library for the Arduino/ESP32 to serialize and de-serialize JSON documents. Below is an example of JSON de-serializing from the endpoint https://httpbin.org/get

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>

const char ssid[] = "ssid";
const char password[] = "password";
const char *server_cert = R"(-----BEGIN CERTIFICATE-----
Get the server's root CA by running the command below

openssl s_client -connect {Host}:{Port} -showcerts
-----END CERTIFICATE-----)";

const char *client_cert = R"(-----BEGIN CERTIFICATE-----
Generate client certificates by running the command below

openssl req -newkey rsa:2048 -nodes -keyout client_key.pem -x509 -days 365 -out client_cert.pem

Fill client_cert with client_cert.pem
-----END CERTIFICATE-----)";

const char *client_key = R"(-----BEGIN PRIVATE KEY-----
Fill client_key with client_key.pem
-----END PRIVATE KEY-----)";
```

```
void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);
    Serial.printf("Connecting to WiFi with SSID : %s\n", ssid);
    while (!WiFi.isConnected());
    Serial.println("Connection successful");

    delay(1000);
}

void loop() {
    WiFiClientSecure *client = new WiFiClientSecure;
    if (client) {
        client->setCACert(server_cert);
        client->setCertificate(client_cert);
        client->setPrivateKey(client_key);

        {
            // Add a scoping block for HTTPClient https to ensure it is destroyed before WiFiClientSecure *client is
            HTTPClient https;

            Serial.print("[HTTPS] begin...\n");
            if (https.begin(*client, "https://httpbin.org/get")) {  // HTTPS
                Serial.print("[HTTPS] GET...\n");
                // start connection and send HTTP header
                int httpCode = https.GET();

                // httpCode will be negative on error
                if (httpCode > 0) {
                    // HTTP header has been sent and Server response header has been handled
                    Serial.printf("[HTTPS] GET... code: %d\n", httpCode);

                    // file found at server
                    if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY) {
                        String payload = https.getString();
                        Serial.println(payload);

                        // Parse JSON payload
```

```
                DynamicJsonDocument doc(1024);
                DeserializationError error = deserializeJson(doc, payload);

                if (!error) {
                    // Extract "origin" and "url" values
                    const char* origin = doc["origin"];
                    const char* url = doc["url"];

                    // Print extracted values
                    Serial.printf("Origin: %s\n", origin);
                    Serial.printf("URL: %s\n", url);
                } else {
                    Serial.print("JSON parsing failed: ");
                    Serial.println(error.c_str());
                }
              }
            } else {
              Serial.printf("[HTTPS] GET... failed, error: %s\n", https.errorToString(httpCode).c_str());
            }
            https.end();
          } else {
            Serial.printf("[HTTPS] Unable to connect\n");
          }

          // End extra scoping block
        }

        delete client;
    } else {
        Serial.println("Unable to create client");
    }

    Serial.println();
    Serial.println("Waiting 10s before the next round...");
    delay(10000);
}
```

MQTT (PubSubClient)

Basic MQTT example

This sketch demonstrates the basic capabilities of the library. It connects to an MQTT server then:

- publishes "hello world" to the topic "outTopic"
- subscribes to the topic "inTopic", printing out any messages it receives. NB - it assumes the received payloads are strings not binary

It will reconnect to the server if the connection is lost using a blocking reconnect function.

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>

const char ssid[] = "ssid";
const char password[] = "password";
const char *server_cert = R"(-----BEGIN CERTIFICATE-----
Get the server's root CA by running the command below

openssl s_client -connect {Host}:{Port} -showcerts
-----END CERTIFICATE-----)";

const char *client_cert = R"(-----BEGIN CERTIFICATE-----
Generate client certificates by running the command below

openssl req -newkey rsa:2048 -nodes -keyout client_key.pem -x509 -days 365 -out client_cert.pem

Fill client_cert with client_cert.pem
-----END CERTIFICATE-----)";

const char *client_key = R"(-----BEGIN PRIVATE KEY-----
Fill client_key with client_key.pem
-----END PRIVATE KEY-----)";

const char mqttServer[] = "broker.hivemq.com";
const int mqttPort = 8883;

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
```

```
    for (int i=0;i<length;i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

WiFiClientSecure wifiClient;
PubSubClient client(wifiClient);

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect("arduinoClient")) {
            Serial.println("connected");
            // Once connected, publish an announcement...
            client.publish("outTopic","hello world");
            // ... and resubscribe
            client.subscribe("inTopic");
        } 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()
{
    Serial.begin(115200);

    client.setServer(mqttServer, mqttPort);
    client.setCallback(callback);

    WiFi.begin(ssid, password);

    wifiClient.setCACert(server_cert);
    wifiClient.setCertificate(client_cert);
```

```
    wifiClient.setPrivateKey(client_key);

    delay(1000);
}

void loop()
{
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}
```

Revision #4
Created 3 November 2024 16:38:06 by Giovan Christoffel Sihombing
Updated 7 November 2024 03:08:06 by Giovan Christoffel Sihombing