/\*\*

\* Created by K. Suwatchai (Mobizt)

\*

\* Email: k\_suwatchai@hotmail.com

\*

\* Github: https://github.com/mobizt/Firebase-ESP32

\*

\* Copyright (c) 2023 mobizt

\*

\*/

#include <NTPClient.h>

#include <WiFiUdp.h>

#include <Arduino.h>

#include <WiFi.h>

#include <FirebaseESP32.h>

#define TRIGGER\_PIN 4

#define ECHO\_PIN 5

// Provide the token generation process info.

#include <addons/TokenHelper.h>

// Provide the RTDB payload printing info and other helper functions.

#include <addons/RTDBHelper.h>

/\* 1. Define the WiFi credentials \*/

#define WIFI\_SSID "Agent's Wifi"

#define WIFI\_PASSWORD "aannaa@303"

// For the following credentials, see examples/Authentications/SignInAsUser/EmailPassword/EmailPassword.ino

/\* 2. Define the API Key \*/

#define API\_KEY "AIzaSyClevWDzWVhSSiWjH7Qpdb2-ioF4cvks0o"

/\* 3. Define the RTDB URL \*/

#define DATABASE\_URL "fir-6fe73-default-rtdb.firebaseio.com" //<databaseName>.firebaseio.com or <databaseName>.<region>.firebasedatabase.app

/\* 4. Define the user Email and password that alreadey registerd or added in your project \*/

#define USER\_EMAIL "demo1@gmail.com"

#define USER\_PASSWORD "test@123"

// Define Firebase Data object

FirebaseData fbdo;

WiFiUDP ntpUDP;

const long utcOffsetInSeconds = 19800; // 5 hours 30 minutes in seconds

NTPClient timeClient(ntpUDP, "pool.ntp.org", utcOffsetInSeconds, 60000); // Update every 60 seconds

FirebaseAuth auth;

FirebaseConfig config;

unsigned long sendDataPrevMillis = 0;

unsigned long count = 0;

void setup()

{

Serial.begin(115200);

pinMode(TRIGGER\_PIN, OUTPUT);

pinMode(ECHO\_PIN, INPUT);

WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

Serial.print("Connecting to Wi-Fi");

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

{

Serial.print(".");

delay(300);

}

Serial.println();

Serial.print("Connected with IP: ");

Serial.println(WiFi.localIP());

Serial.println();

Serial.printf("Firebase Client v%s\n\n", FIREBASE\_CLIENT\_VERSION);

/\* Assign the api key (required) \*/

config.api\_key = API\_KEY;

/\* Assign the user sign in credentials \*/

auth.user.email = USER\_EMAIL;

auth.user.password = USER\_PASSWORD;

/\* Assign the RTDB URL (required) \*/

config.database\_url = DATABASE\_URL;

/\* Assign the callback function for the long running token generation task \*/

config.token\_status\_callback = tokenStatusCallback; // see addons/TokenHelper.h

// Comment or pass false value when WiFi reconnection will control by your code or third party library e.g. WiFiManager

Firebase.reconnectNetwork(true);

// Since v4.4.x, BearSSL engine was used, the SSL buffer need to be set.

// Large data transmission may require larger RX buffer, otherwise connection issue or data read time out can be occurred.

fbdo.setBSSLBufferSize(4096 /\* Rx buffer size in bytes from 512 - 16384 \*/, 1024 /\* Tx buffer size in bytes from 512 - 16384 \*/);

// Or use legacy authenticate method

// config.database\_url = DATABASE\_URL;

// config.signer.tokens.legacy\_token = "<database secret>";

// To connect without auth in Test Mode, see Authentications/TestMode/TestMode.ino

//////////////////////////////////////////////////////////////////////////////////////////////

// Please make sure the device free Heap is not lower than 80 k for ESP32 and 10 k for ESP8266,

// otherwise the SSL connection will fail.

//////////////////////////////////////////////////////////////////////////////////////////////

Firebase.begin(&config, &auth);

Firebase.setDoubleDigits(5);

// You can use TCP KeepAlive in FirebaseData object and tracking the server connection status, please read this for detail.

// https://github.com/mobizt/Firebase-ESP32#about-firebasedata-object

// fbdo.keepAlive(5, 5, 1);

/\*\* Timeout options.

//Network reconnect timeout (interval) in ms (10 sec - 5 min) when network or WiFi disconnected.

config.timeout.networkReconnect = 10 \* 1000;

//Socket connection and SSL handshake timeout in ms (1 sec - 1 min).

config.timeout.socketConnection = 10 \* 1000;

//Server response read timeout in ms (1 sec - 1 min).

config.timeout.serverResponse = 10 \* 1000;

//RTDB Stream keep-alive timeout in ms (20 sec - 2 min) when no server's keep-alive event data received.

config.timeout.rtdbKeepAlive = 45 \* 1000;

//RTDB Stream reconnect timeout (interval) in ms (1 sec - 1 min) when RTDB Stream closed and want to resume.

config.timeout.rtdbStreamReconnect = 1 \* 1000;

//RTDB Stream error notification timeout (interval) in ms (3 sec - 30 sec). It determines how often the readStream

//will return false (error) when it called repeatedly in loop.

config.timeout.rtdbStreamError = 3 \* 1000;

Note:

The function that starting the new TCP session i.e. first time server connection or previous session was closed, the function won't exit until the

time of config.timeout.socketConnection.

You can also set the TCP data sending retry with

config.tcp\_data\_sending\_retry = 1;

\*/

}

void loop()

{

// Firebase.ready() should be called repeatedly to handle authentication tasks.

/\* digitalWrite(TRIGGER\_PIN, LOW);

delayMicroseconds(2);

digitalWrite(TRIGGER\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIGGER\_PIN, LOW);

// Read the duration of the pulse on the ECHO\_PIN

unsigned long duration = pulseIn(ECHO\_PIN, HIGH);

// Calculate the distance in centimeters using the speed of sound

// (approximately 29 microseconds per centimeter at room temperature)

unsigned int distance = duration / 29 / 2;\*/

if (Firebase.ready() && (millis() - sendDataPrevMillis > 15000 || sendDataPrevMillis == 0))

{

sendDataPrevMillis = millis();

timeClient.update();

digitalWrite(TRIGGER\_PIN, LOW);

delayMicroseconds(2);

digitalWrite(TRIGGER\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIGGER\_PIN, LOW);

// Read the duration of the pulse on the ECHO\_PIN

unsigned long duration = pulseIn(ECHO\_PIN, HIGH);

// Calculate the distance in centimeters using the speed of sound

// (approximately 29 microseconds per centimeter at room temperature)

unsigned int distance = duration / 29 / 2;

String path = "/demo/d/1/status"; // Adjust the path accordingly

Serial.println(distance);

int value = (distance < 10) ? 1 : 0; // Determine the value to update

String startTimePath = "/demo/d/1/startTime";

if (Firebase.getString(fbdo, startTimePath)) {

String startTimeStr = fbdo.stringData();

Serial.printf("Start time: %s\n", startTimeStr.c\_str());

// Parse start time

int startHour = startTimeStr.substring(0, 2).toInt();

int startMinute = startTimeStr.substring(3, 5).toInt();

Serial.printf("Start hour %d",startHour);

Serial.printf("Start minute %d",startMinute);

// Get current time

timeClient.update();

int currentHour = timeClient.getHours();

int currentMinute = timeClient.getMinutes();

Serial.printf("Current time: %02d:%02d\n", currentHour, currentMinute);

// Calculate time difference in minutes

int startTimeInMinutes = startHour \* 60 + startMinute;

int currentTimeInMinutes = currentHour \* 60 + currentMinute;

int timeDifference = currentTimeInMinutes - startTimeInMinutes;

if (timeDifference < 0) {

timeDifference += 24 \* 60; // Handle day wrap-around

}

// Update the status if the time difference is more than 5 minutes

if (timeDifference >= 5) {

Serial.printf("Updating %s to %d\n", path.c\_str(), value);

Firebase.setInt(fbdo, path.c\_str(), value);

} else {

Serial.println("Time difference is less than 5 minutes, not updating status");

}

} else {

Serial.printf("Failed to retrieve start time: %s\n", fbdo.errorReason().c\_str());

}

}

}

/// PLEASE AVOID THIS ////

// Please avoid the following inappropriate and inefficient use cases

/\*\*

\*

\* 1. Call get repeatedly inside the loop without the appropriate timing for execution provided e.g. millis() or conditional checking,

\* where delay should be avoided.

\*

\* Everytime get was called, the request header need to be sent to server which its size depends on the authentication method used,

\* and costs your data usage.

\*

\* Please use stream function instead for this use case.

\*

\* 2. Using the single FirebaseData object to call different type functions as above example without the appropriate

\* timing for execution provided in the loop i.e., repeatedly switching call between get and set functions.

\*

\* In addition to costs the data usage, the delay will be involved as the session needs to be closed and opened too often

\* due to the HTTP method (GET, PUT, POST, PATCH and DELETE) was changed in the incoming request.

\*

\*

\* Please reduce the use of swithing calls by store the multiple values to the JSON object and store it once on the database.

\*

\* Or calling continuously "set" or "setAsync" functions without "get" called in between, and calling get continuously without set

\* called in between.

\*

\* If you needed to call arbitrary "get" and "set" based on condition or event, use another FirebaseData object to avoid the session

\* closing and reopening.

\*

\* 3. Use of delay or hidden delay or blocking operation to wait for hardware ready in the third party sensor libraries, together with stream functions e.g. Firebase.RTDB.readStream and fbdo.streamAvailable in the loop.

\*

\* Please use non-blocking mode of sensor libraries (if available) or use millis instead of delay in your code.

\*

\* 4. Blocking the token generation process.

\*

\* Let the authentication token generation to run without blocking, the following code MUST BE AVOIDED.

\*

\* while (!Firebase.ready()) <---- Don't do this in while loop

\* {

\* delay(1000);

\* }

\*

\*/