/\*

Video: https://www.youtube.com/watch?v=oCMOYS71NIU

Based on Neil Kolban example for IDF: https://github.com/nkolban/esp32-snippets/blob/master/cpp\_utils/tests/BLE%20Tests/SampleNotify.cpp

Ported to Arduino ESP32 by Evandro Copercini

Create a BLE server that, once we receive a connection, will send periodic notifications.

The service advertises itself as: 6E400001-B5A3-F393-E0A9-E50E24DCCA9E

Has a characteristic of: 6E400002-B5A3-F393-E0A9-E50E24DCCA9E - used for receiving data with "WRITE"

Has a characteristic of: 6E400003-B5A3-F393-E0A9-E50E24DCCA9E - used to send data with "NOTIFY"

The design of creating the BLE server is:

1. Create a BLE Server

2. Create a BLE Service

3. Create a BLE Characteristic on the Service

4. Create a BLE Descriptor on the characteristic

5. Start the service.

6. Start advertising.

In this example rxValue is the data received (only accessible inside that function).

And voltageValue is the data to be sent, in this example just a byte incremented every second.

\*/

#include <BLEDevice.h>

#include <BLEServer.h>

#include <BLEUtils.h>

#include <BLE2902.h>

BLECharacteristic \*pCharacteristic;

bool deviceConnected = false;

float txValue = 0;

const int readPin = 32; // Use GPIO number. See ESP32 board pinouts

const int RED = 25; // Could be different depending on the dev board. I used the DOIT ESP32 dev board.

const int GREEN = 26;

const int BLUE = 2;

int adder = 0;

//std::string rxValue; // Could also make this a global var to access it in loop()

// See the following for generating UUIDs:

// https://www.uuidgenerator.net/

#define SERVICE\_UUID "6E400001-B5A3-F393-E0A9-E50E24DCCA9E" // UART service UUID

#define CHARACTERISTIC\_UUID\_RX "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"

#define CHARACTERISTIC\_UUID\_TX "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"

class MyServerCallbacks: public BLEServerCallbacks {

void onConnect(BLEServer\* pServer) {

deviceConnected = true;

};

void onDisconnect(BLEServer\* pServer) {

deviceConnected = false;

}

};

class MyCallbacks: public BLECharacteristicCallbacks {

void onWrite(BLECharacteristic \*pCharacteristic) {

std::string rxValue = pCharacteristic->getValue();

if (rxValue.length() > 0) {

Serial.println("\*\*\*\*\*\*\*\*\*");

Serial.print("Received Value: ");

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

Serial.print(rxValue[i]);

}

Serial.println();

// Do stuff based on the command received from the app

if (rxValue.find("SYS\_ON") != -1) {

Serial.print("Turning ON!");

digitalWrite(RED, HIGH);

digitalWrite(BLUE, HIGH);

digitalWrite(GREEN, HIGH);

}

else if (rxValue.find("SYS\_OFF") != -1) {

Serial.print("Turning OFF!");

digitalWrite(RED, LOW);

digitalWrite(BLUE, LOW);

digitalWrite(GREEN, LOW);

}

else if (rxValue.find("A") != -1) {

Serial.print("Turning ON!");

digitalWrite(BLUE, HIGH);

}

else if (rxValue.find("B") != -1) {

Serial.print("Turning OFF!");

digitalWrite(BLUE, LOW);

}

else if (rxValue.find("C") != -1) {

Serial.print("Turning ON!");

digitalWrite(RED, HIGH);

}

else if (rxValue.find("D") != -1) {

Serial.print("Turning OFF!");

digitalWrite(RED, LOW);

}

else if (rxValue.find("E") != -1) {

Serial.print("Turning ON!");

digitalWrite(GREEN, HIGH);

}

else if (rxValue.find("F") != -1) {

Serial.print("Turning OFF!");

digitalWrite(GREEN, LOW);

}

Serial.println();

Serial.println("\*\*\*\*\*\*\*\*\*");

}

}

};

void setup() {

Serial.begin(115200);

pinMode(RED, OUTPUT);

pinMode(GREEN, OUTPUT);

pinMode(BLUE, OUTPUT);

// Create the BLE Device

BLEDevice::init("ESP32"); // Give it a name

// Create the BLE Server

BLEServer \*pServer = BLEDevice::createServer();

pServer->setCallbacks(new MyServerCallbacks());

// Create the BLE Service

BLEService \*pService = pServer->createService(SERVICE\_UUID);

// Create a BLE Characteristic

pCharacteristic = pService->createCharacteristic(

CHARACTERISTIC\_UUID\_TX,

BLECharacteristic::PROPERTY\_NOTIFY

);

pCharacteristic->addDescriptor(new BLE2902());

BLECharacteristic \*pCharacteristic = pService->createCharacteristic(

CHARACTERISTIC\_UUID\_RX,

BLECharacteristic::PROPERTY\_WRITE

);

pCharacteristic->setCallbacks(new MyCallbacks());

// Start the service

pService->start();

// Start advertising

pServer->getAdvertising()->start();

Serial.println("Waiting a client connection to notify...");

}

void loop() {

if (deviceConnected) {

if(adder <= 10){

txValue = adder; // Test sensor reading

}

else{

adder = 0;

}

// Let's convert the value to a char array:

char txString[8];

dtostrf(txValue, 1, 2, txString); // float\_val, min\_width, digits\_after\_decimal, char\_buffer

// pCharacteristic->setValue(&voltageValue, 1); // To send the integer value

pCharacteristic->setValue("120.0|1.23|2.45|3.40|00|05|45"); // Sending a test message

// pCharacteristic->setValue(txString);

pCharacteristic->notify(); // Send the value to the app!

Serial.print("\*\*\* Sent Value: ");

Serial.print(txString);

Serial.println(" \*\*\*");

}

adder += 1;

delay(1000);

}