#include <BLEDevice.h>

#include <BLEServer.h>

#include <BLEUtils.h>

#include <BLE2902.h>

BLECharacteristic \*pCharacteristic;

bool deviceConnected = false;

float txValue = 0;

#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(relay\_1, HIGH);

digitalWrite(relay\_3, HIGH);

digitalWrite(relay\_2, HIGH);

}

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

Serial.print("Turning OFF!");

digitalWrite(relay\_1, LOW);

digitalWrite(relay\_3, LOW);

digitalWrite(relay\_2, LOW);

}

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

Serial.print("Turning ON!");

digitalWrite(relay\_3, HIGH);

}

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

Serial.print("Turning OFF!");

digitalWrite(relay\_3, LOW);

}

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

Serial.print("Turning ON!");

digitalWrite(relay\_1, HIGH);

}

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

Serial.print("Turning OFF!");

digitalWrite(relay\_1, LOW);

}

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

Serial.print("Turning ON!");

digitalWrite(relay\_2, HIGH);

}

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

Serial.print("Turning OFF!");

digitalWrite(relay\_2, LOW);

}

Serial.println();

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

}

}

};

void setup() {

Serial.begin(115200);

pinMode(relay\_1, OUTPUT);

pinMode(relay\_2, OUTPUT);

pinMode(relay\_3, 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) {

// Fabricate some arbitrary junk for now...

txValue = analogRead(readPin) / 3.456; // This could be an actual sensor reading!

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

char txString[8]; // make sure this is big enuffz

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

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

// pCharacteristic->setValue("Hello!"); // Sending a test message

pCharacteristic->setValue(txString);

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

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

Serial.print(txString);

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

delay(1000);

}