**Firmware for ESP32-BLE Broadcasting Services**

**By Nirvan Tamhane**

/\*

  modified 10 July 2024

  by Nirvan Tamhane

  Ref.

  Rui Santos, Neil Kolban.

  Parts required:

  - 1x ESP32 Devkit V1 Development Board

  - 1x 10 kilo Ohm Resistor (For Pull-up)

  - Jumpers

  Connections:

  | 3v3 -> VCC          |

  | GND -> GND          |

  | GPIO2 -> S/OUT      |

  | VCC -> 10K -> S/OUT |

  Complete project detail @

  Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files.

  The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

\*/

#include <BLEDevice.h>

#include <BLEServer.h>

#include <BLEUtils.h>

#include <BLE2902.h>

#include "DHT.h"

// UUID for the Unique User Service

#define SERVICE\_UUID "00000002-0000-0000-FDFD-FDFDFDFDFDFD"

// Define the Temperature Characteristic and Descriptor using default UUIDs

BLECharacteristic TEMP\_Characteristic(BLEUUID((uint16\_t)0x2A1C), BLECharacteristic::PROPERTY\_NOTIFY | BLECharacteristic::PROPERTY\_READ);

BLEDescriptor TEMP\_Descriptor(BLEUUID((uint16\_t)0x2902));

// Define the Humidity Characteristic and Descriptor using default UUIDs

BLECharacteristic HDT\_Characteristic(BLEUUID((uint16\_t)0x2A6F), BLECharacteristic::PROPERTY\_NOTIFY | BLECharacteristic::PROPERTY\_READ);

BLEDescriptor HDT\_Descriptor(BLEUUID((uint16\_t)0x2902));

BLEServer\* pServer;  // Creating a BLE Server

#define DHTPIN 2       // (GPIO2) Digital pin connected to the DHT sensor

#define DHTTYPE DHT11  // Type of DHT sensor

DHT dht(DHTPIN, DHTTYPE);  // Creating an instance for DHT class

bool deviceConnected = false;     // Flag to track connection status

bool oldDeviceConnected = false;  // Previous connection status

float temp = 0;   // Variable to store temperature

float humid = 0;  // Variable to store humidity

// Structure for IEEE-11073 32-bit float

struct ieee11073\_32bit\_float {

  uint8\_t mantissa[3];

  uint8\_t exponent;

};

// Function to convert float to IEEE-11073 32-bit float

ieee11073\_32bit\_float float\_to\_ieee11073(float value) {

  ieee11073\_32bit\_float result;

  int32\_t mantissa = (int32\_t)(value \* 100);

  result.mantissa[0] = mantissa & 0xFF;

  result.mantissa[1] = (mantissa >> 8) & 0xFF;

  result.mantissa[2] = (mantissa >> 16) & 0xFF;

  result.exponent = -2;  // The exponent part of the float (2 decimal places)

  return result;

}

// Callbacks for BLE server connection events

class MyServerCallbacks : public BLEServerCallbacks {

  void onConnect(BLEServer\* pServer) {

    deviceConnected = true;

    Serial.println("Device Connected");  // Print on Connection

  };

  void onDisconnect(BLEServer\* pServer) {

    deviceConnected = false;

    Serial.println("Device Disconnected");  // Print on Disconnection

  }

};

void setup() {

  // Start serial communication

  Serial.begin(115200);

  //Start reading from the DHT11 sensor

  dht.begin();

  // Initialize the BLE Device

  BLEDevice::init("Weather");

  // Create the BLE Server and set callbacks

  pServer = BLEDevice::createServer();

  pServer->setCallbacks(new MyServerCallbacks());

  // Create the BLE Service

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

  // Add Characteristics and Descriptors to the Service

  ENVM\_Service->addCharacteristic(&TEMP\_Characteristic);

  TEMP\_Characteristic.addDescriptor(&TEMP\_Descriptor);

  ENVM\_Service->addCharacteristic(&HDT\_Characteristic);

  HDT\_Characteristic.addDescriptor(&HDT\_Descriptor);

  // Start the BLE Service

  ENVM\_Service->start();

  // Start advertising the service

  pServer->getAdvertising()->start();

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

}

void loop() {

  readData();     // Read temperature and humidity data

  sendData();     // Send data over BLE

  checkStatus();  // Check the connection status

  delay(1000);    // Wait for a second before repeating

}

// Function to read data from the DHT sensor

void readData() {

  temp = dht.readTemperature();  // Read temperature as Celsius

  humid = dht.readHumidity();    // Read humidity

  // Check for failed readings

  if (isnan(temp) || isnan(humid)) {

    Serial.println("Failed to read from DHT sensor!");

  }

  // Print readings to the serial monitor

  Serial.print("Temperature: ");

  Serial.print(temp);

  Serial.print("°C | ");

  Serial.print("Humidity: ");

  Serial.print(humid);

  Serial.println("%");

  Serial.println("x----------------x--------------x");

}

// Function to send data over BLE

void sendData() {

  if (deviceConnected) {

    Serial.println("Device Connected");

    // Convert temperature to IEEE-11073 32-bit float

    ieee11073\_32bit\_float tempVal = float\_to\_ieee11073(temp);

    // Prepare data buffer for Temperature Measurement characteristic

    uint8\_t tempBuf[5];  // Flags (1 byte) + Celsius (4 bytes)

    // Flags byte: bit 0 for Celsius present

    uint8\_t flags = 0x00;  // Fahrenheit present

    tempBuf[0] = flags;

    // Copy the Celsius value (4 bytes) into buffer

    memcpy(tempBuf + 1, &tempVal, sizeof(tempVal));

    // Set temperature Characteristic value and notify connected client

    TEMP\_Characteristic.setValue(tempBuf, sizeof(tempBuf));

    TEMP\_Characteristic.notify();

    // Convert and notify humidity reading

    uint16\_t humidity = (uint16\_t)humid \* 100;

    // Set humidity Characteristic value and notify connected client

    HDT\_Characteristic.setValue(humidity);

    HDT\_Characteristic.notify();

  } else {

    Serial.println("Device Not Connected");

  }

}

// Function to check the connection status and restart advertising if necessary

void checkStatus() {

  // If device is disconnected, restart advertising

  if (!deviceConnected && oldDeviceConnected) {

    delay(500);                   // Give the Bluetooth stack time to get ready

    pServer->startAdvertising();  // Restart advertising

    Serial.println("Restart Advertising");

    oldDeviceConnected = deviceConnected;

  }

  // Update old connection status

  if (deviceConnected && !oldDeviceConnected) {

    oldDeviceConnected = deviceConnected;

  }

}