Práctica 4: Tecnologías de comunicación y sensorización BLE

Sistemas Embebidos

Francisco Javier Pérez Martínez

22 de abril de 2022

Índice

1. Trabajo a realizar

Para el presente documento, correspondiente a la práctica 4 de la asignatura, utilizaremos nuevamente el dispositivo embebido, Arduino Nano 33 IoT. En esta práctica, se probará la sensorización BLE del dispositivo en modo emisor y monitor, se calculará una tabla de distancias a partir del RSSI (indicador de fuerza de la señal recibida) y se probará el funcionamiento combinado de WiFi + BLE.

2. Funcionamiento BLE

2.1. BLE: Modo Emisor

En primer lugar debemos instalarnos la librería "ArduinoBLE". Una vez instalada, cargaremos en el entorno el ejemplo de "ArduinoBLE/Peripheral/BatteryMonitor" y conectaremos el teléfono móvil mediante Bluetooh.

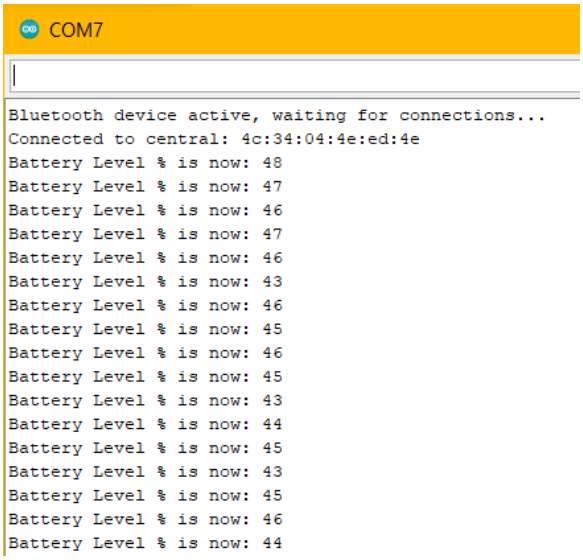

Figura 1: Salida modo emisor, dispositivo conectado

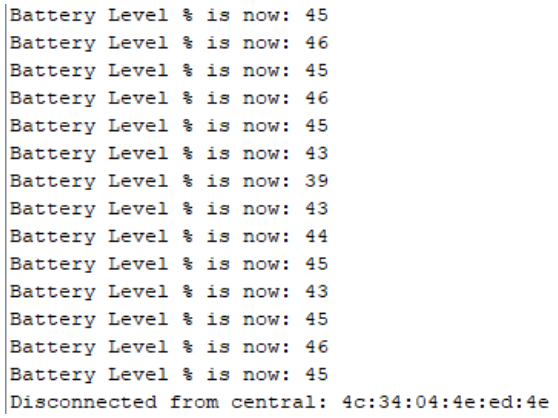

Figura 2: Salida modo emisor, dispositivo desconectado

Al ejecutar el código de ejemplo, podemos ver desde el monitor serial como se ha conectado al teléfono móvil y muestra el porcentaje de batería del dispositivo. Sin embargo, la estimación de este porcentaje no es del todo exacta debido al modo stamina del propio dispositivo móvil.

2.2. BLE: Modo Monitor

Para este modo, cargaremos el ejemplo "ArduinoBLE/Central/Scan" y comprobaremos los dispositivos que se detectan y la información que se obtiene de ellos así como la frecuencia detectada.

Al ejecutar el programa, no se detectan dispositivos Bluetooth Low Energy, esto es debido a que debemos instalarnos en nuestro dispositivo móvil una App para BLE, en mi caso, he instalado "Beacon Scope". Una vez activada la baliza, se puede ver en la figura siguiente como se detectan varios dispositivos:

Figura 3: App BeaconScope instalada

```
Discovered a peripheral
-----------------------
Address: 55:0a:ac:cb:b8:f8
RSSI: -79

Discovered a peripheral
-----------------------
Address: 67:57:40:89:ec:48
RSSI: -83

Discovered a peripheral
-----------------------
Address: 60:d7:0e:39:40:ff
RSSI: -83

Discovered a peripheral
-----------------------
Address: fe:0a:22:99:2b:39
Local Name: Mi Smart Band 5
Service UUIDs: fee0
RSSI: -82

Discovered a peripheral
-----------------------
Address: d4:99:33:e8:e3:f1
Local Name: Mi Smart Band 4
Service UUIDs: fee0
RSSI: -87
```

Figura 4: Dispositivos BLE encontrados

Para la siguiente parte del apartado del modo monitor, se nos pide instalarnos la librería "RTCZero" la cual proporciona una reloj de tiempo real para contabilizar el tiempo. Además, se nos pide que el ejemplo realice un escaneo de dispositivos cada 5 segundos. Para ello, debemos incluir las siguientes líneas de código:

```
#include <RTCZero.h>
RTCZero rtc;

void setup() {
    ...
    rtc.begin(); // inicializar rtc
    ...
}

void loop() {
    BLEDevice peripheral;
    // cada 5 segundos comprobamos si un dispositivo ha sido descubierto
    if(rtc.getSeconds() % 5 == 0) {
        // check if a peripheral has been discovered
        peripheral = BLE.available();
        }
    }
    ...
}
```

3. Cálculo de distancia relativa mediante RSSI

Para calcular la tabla de distancias, utilizaremos la App BLE instalada en el teléfono móvil configurando una de las señales BLE para que sea detectada por el sensor arduino de forma que pueda ser identificado aunque la dirección MAC cambie.

Las medidas han sido tomadas desde el aula de prácticas, por cada escaneo movemos el dispositivo móvil 25cm con respecto al arduino hasta alcanzar 2.50m. Además, se han obtenido 5 datos por cada fila de la tabla y posteriormente se ha realizado la mediana ya que ésta se ve menos afectada por los valores atípicos.

Distancia (cm)	**RSSI**
0	-26
25	-43
50	-54
75	-59
100	-63
125	-69
150	-73
175	-78
200	-82
225	-87
250	-89

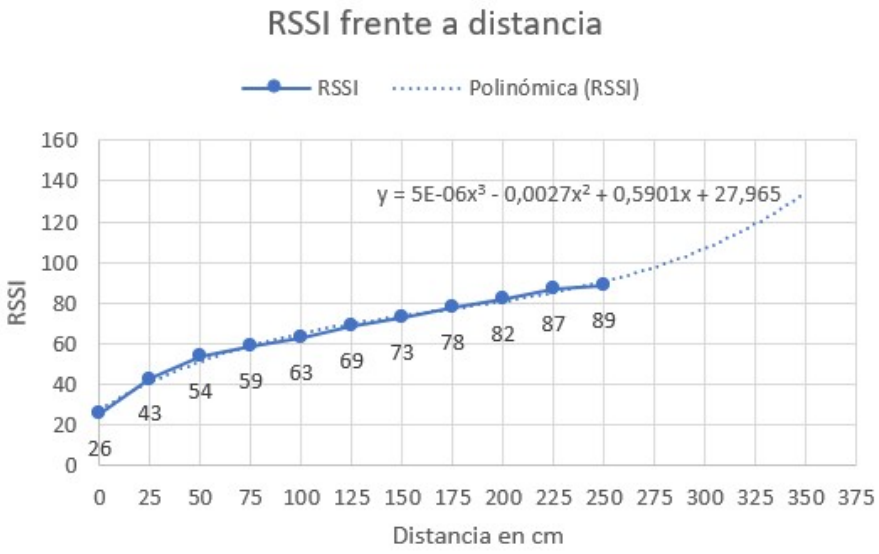

Figura 5: Gráfica RSSI frente a distancia. Tendencia polinómica de grado 3.

4. Funcionamiento combinado WiFi/BLE

4.1. WiFiPing

Para este apartado, se nos pide cargar en el entorno un ejemplo de la librería WiFiNINA, concretamente "WiFiNINA/WiFiPing" y utilizar los códigos LED para comprobar la conexión a Internet.

- LED on: inicio
- LED off: conexión correcta
- LED blink: conexión incorrecta

Una vez cargado el ejemplo, introduciremos el SSID y la contraseña de nuestro dispositivo móvil, el cual actúa como punto de acceso wifi. Para comprobar la conexión a Internet mediante los LEDs, emplearemos las funciones vistas en la práctica 2. Ver apéndice A

La salida del programa se puede ver en la siguiente figura:

Figura 6: Salida monitor serie del ejemplo de WiFiPing con LED desactivado (conexión correcta). El LED encendido es el de la corriente.

4.2. Combinación Ping y Scan

Para este apartado, se nos pide combinar los dos ejemplos vistos anteriormente, el de WiFiPing y el de Scan, de tal forma que cuando se establezca conexión WiFi, después realice un escaneo de dispositivos BLE.

Para ello, primero estableceremos la conexión WiFi con nuestro dispositivo móvil y después activaremos la función de Scan para comprobar el funcionamiento. No obstante, como se puede apreciar en la figura siguiente, la conexión se establece correctamente, en cambio cuando BLE toma el control de la antena del arduino, éste se queda bloqueado al intentar llamar a la función WiFi.ping() para hacer ping al host indicado.

Esto es debido a que el arduino solo consta de una antena de uso y por tanto aunque el módulo WiFi tenga control inicialmente sobre la antena, al arrancar el módulo BLE, el módulo de WiFi se queda sin control de ésta y por lo que si intentamos utilizar alguna función relacionada con la librería WiFiNINA podemos observar que dicha función no se ejecuta y el programa se queda bloqueado.

```
  }

  // you're connected now, so print out the data:
  Serial.println("You're connected to the network");
  printCurrentNet();
  printWiFiData();

  // begin initialization
  if (!BLE.begin()) {
    Serial.println("starting BLE failed!");

    while (1);
  }

  Serial.println("BLE Central scan");
  // start scanning for peripheral
  BLE.scan();
}

void loop() {
  Serial.print("Pinging ");
  Serial.print(hostName);
  Serial.print(": ");

  pingResult = WiFi.ping(hostName);
```

```
COM7
Attempting to connect to WPA SSID: nano33iot
Attempting to connect to WPA SSID: nano33iot
You're connected to the network
SSID: nano33iot
BSSID: C0:F4:E6:A8:AB:C4
signal strength (RSSI): -41
Encryption Type: 4

IP address : 192.168.43.35
Subnet mask: 255.255.255.0
Gateway IP : 192.168.43.1
MAC address: 4C:EB:D6:4C:B9:F8
BLE Central scan
Pinging www.google.com:
```

Figura 7: Problema al combinar Ping + Scan

A. Programa WiFiPing + LEDs

```
#include <SPI.h>
#include <WiFiNINA.h>
#include "arduino_secrets.h" // Escribimos nuestro SSID y contraseña en el fichero arduino_secrets.h
char ssid[] = SECRET_SSID;        // your network SSID (name)
char pass[] = SECRET_PASS;    // your network password (use for WPA, or use as key for WEP)
int status = WL_IDLE_STATUS;     // the WiFi radio's status
String hostName = "www.google.com"; // Specify IP address or hostname
int pingResult;

void setup() {
    pinMode(LED_BUILTIN, OUTPUT); // inicializa el pin digital LED_BUILTIN como una salida.
    digitalWrite(LED_BUILTIN, HIGH); // led on - inicio
    ...
    // attempt to connect to WiFi network:
    while ( status != WL_CONNECTED) {
        Serial.print("Attempting to connect to WPA SSID: ");
        Serial.println(ssid);
         // Connect to WPA/WPA2 network:
        status = WiFi.begin(ssid, pass);
        // wait 5 seconds for connection:
        delay(5000);
        digitalWrite(LED_BUILTIN, LOW); // led off
        blink(500); // conexión incorrecta
    }
    Serial.println("You're connected to the network");
    digitalWrite(LED_BUILTIN, LOW); // led off - conexion correcta
    printCurrentNet(); // print data
    printWiFiData(); // print data
}

void loop() {
  ...
  pingResult = WiFi.ping(hostName);
  if (pingResult >= 0) {
    Serial.print("SUCCESS! RTT = ");
    Serial.print(pingResult);
    Serial.println(" ms");
  } else {
    Serial.print("FAILED! Error code: ");
    Serial.println(pingResult);
  }
  delay(5000);
}
}
```