Sistemas Embebidos - Memoria práctica 9.

Elvi Mihai Sabau Sabau[1]
Alex Navarro Soria[2]

[1] Universidad de Alicante, Alicante, España.
emss5@alu.ua.es
[1] Universidad de Alicante, Alicante, España.
ans33@gcloud.ua.es

1 Descripción.

En esta práctica usaremos un Arduino NANO IoT con una pantalla Oled y un sensor MAX 30102 de pulsaciones, para detectar las pulsaciones, mostrar la media de pulsaciones por minuto en la pantalla oled, y en conjunto con la hora actual, mandarlas a una API Rest en Firebase en formato JSON.

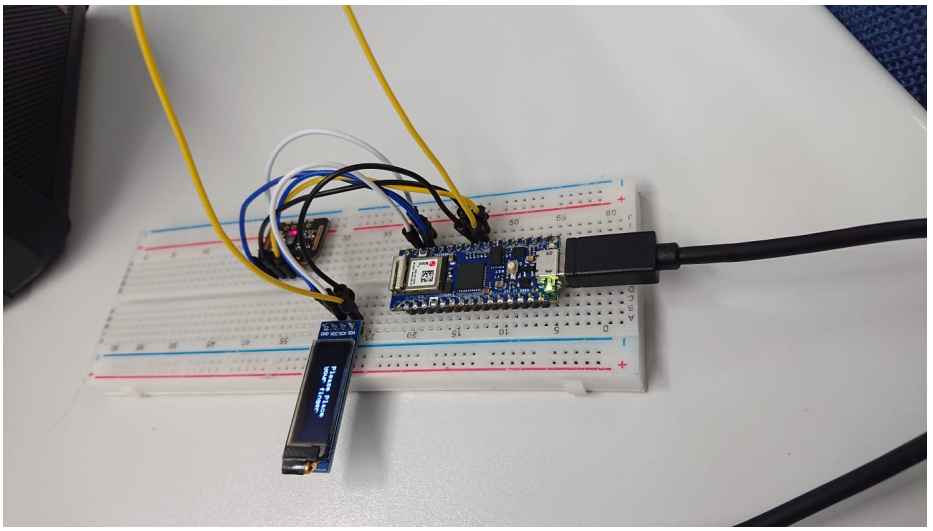
2 Librerias.

Las librerías que usaremos serán las siguientes:

- Adafruit_SSD1306: Para interactuar con la pantalla Oled.
- Adafruit_GFX: Para renderizar polígonos y textos en la pantalla Oled.
- Wire: Para usar los pines de la placa.
- NTPClient: Para obtener la hora actual.
- SPI: Para interactuar con dispositivos periféricos.
- WiFiNINA: Para interactuar con la antena, usando redes WiFi.
- WiFiUdp: Para recibir datos específicamente mediante el protocolo UDP.
- ArduinoHttpClient: Para hacer peticiones HTTP.
- ArduinoJson: Para poder parsear un String a JSON y viceversa.
- ArduinoECCX08: Para usar el chip criptográfico del arduino (Optativo en caso de usar JWT).

3 Montaje.

El montaje de la placa es el siguiente, conectando los pines del Arduino NANO IoT a la pantalla Oled y al sensor de pulsaciones.
:



4 Ejercicio.

Para este ejercicio nos hemos basado en 3 ejemplos:

- El ejemplo de la práctica anterior, para mostrar las pulsaciones por la pantalla Oled.
- Un ejemplo de interacción y solicitud de datos a un servidor NTP.
- Un ejemplo de interacción con JSONs y emisión vía HTTP POST de los datos a un servidor Firebase.

Gracias a estos ejemplos, hemos realizado un programa que, inicialmente instancia e inicializa las variables de configuración de la pantalla Oled y el sensor de pulsaciones y el cliente NTP.

```
// Declaring the display name (display)
Adafruit_SSD1306    display(SCREEN_WIDTH,    SCREEN_HEIGHT,    &Wire,
OLED_RESET);

...

const byte RATE_SIZE = 4; // Increase this for more averaging. 4 is
good.
byte rates[RATE_SIZE]; // Array of heart rates
byte rateSpot = 0;
long lastBeat = 0; // Time at which the last beat occurred
float beatsPerMinute;
int beatAvg;

...

// Define NTP Client to get time
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", utcOffsetInSeconds);
```

Una vez inicializado, se ejecutará la función “setup()”, que hará lo siguiente:

- Configura el sensor de pulsaciones.
- Se conectará a una red WiFi.
- Inicia la conexión con el servidor NTP.

```
// Initialize sensor
particleSensor.begin(Wire, I2C_SPEED_FAST); // Use default I2C port,
400kHz speed
particleSensor.setup(); // Configure sensor with default settings
particleSensor.setPulseAmplitudeRed(0x0A); // Turn Red LED to low to
indicate sensor is running

WiFi.begin(ssid, password);

while ( WiFi.status() != WL_CONNECTED ) {
    delay ( 500 );
    Serial.print ( "." );
}

timeClient.begin();
```

Una vez hecho esto, en nuestro código definiremos 2 funciones, sacadas del 3er ejemplo.

- Una función que parsee a JSON un string.

```
DynamicJsonDocument toJsonDocument(String str, int size = 0) {
  if (size == 0) {
    size = JSON_OBJECT_SIZE(1) + str.length();
  }
  DynamicJsonDocument jsonDoc(str.length() * 2);
  DeserializationError error = deserializeJson(jsonDoc, str);
  if (error) {
    Serial.print("JSON " + String(error.c_str()) + ": " + str);
    jsonDoc.clear();
  }
  return jsonDoc;
}
```

- Una función que envía un JSON a una API Rest en Firebase.

```
bool firebaseDatabasePut(String path, DynamicJsonDocument jsonDoc) {
  String jsonStr;
  serializeJson(jsonDoc, jsonStr);
  Serial.print("Saving to RTDB: " + path + " = " + jsonStr);

  String host =
String(PROJECT_ID)+".europe-west1.firebasedatabase.app";
  WiFiSSLClient wifiClient;
  HttpClient httpClient = HttpClient(wifiClient, host, 443);

  String url = "/" + path + ".json";

  httpClient.put(url, "application/json", jsonStr);
  int statusCode = httpClient.responseStatusCode();
  String response = httpClient.responseBody();

  if (statusCode != 200) {
    Serial.print(String(statusCode) + " " + response);
    return false;
  }

  Serial.print("Saved to RTDB.");
  return true;
}
```

Además, desde el código de la práctica anterior, nos traeremos 3 funciones, una que detecte si hay un dedo sobre el sensor, otra que muestre las pulsaciones, y otra que muestre un mensaje pidiendo que se ponga el dedo sobre el sensor en caso de no detectar el dedo. No vamos a mostrar el código porque ya se ha expuesto en la práctica anterior.

Una vez hecho todo esto, en nuestro “loop()” haremos lo siguiente:

- Leeremos el valor del sensor de pulsaciones.
- Detectaremos si hay un dedo.
- Si no lo hay, mostramos un mensaje y no hacemos nada más
- Si hay un dedo, comprobamos que el valor es válido, y si lo es, obtendremos la hora actual, crearemos un String en formato JSON con el nombre, la hora y el valor medio “beatAvg”, creamos un objeto JSON desde el String y por último llamamos a la función que nos permitirá mandar los datos a firebase con el objeto JSON.

```
void loop() {
  long irValue = particleSensor.getIR();

  // If a finger is detected
  if(irValue >= 7000) {
    fingerDetected();

    //If a heart beat is detected
    if (checkForBeat(irValue) == true) {
        showHeartBeatValue();
        timeClient.update();
        Serial.println(timeClient.getFormattedTime());
        String msg = "{heartRate:" + (String)beatAvg + ",
name:'alex-elvi',time:'" + (String)timeClient.getFormattedTime() +
"'}";
        DynamicJsonDocument jsonMsg = toJsonDocument(msg);
        firebaseDatabasePut(DEVICE_ID, jsonMsg);
      }
    }

  else {
     noFingerDetected();
  }
}
```

En este caso, el JSON que debería aparecer en la consola de Firebase sería el siguiente:

```
{
  "heartRate": "PulsacionesMediasPorMinuto",
  "name": "alex-elvi",
  "time": "Hora:Minutos:Segundos"
}
```

5 Ejecución.

A continuación se muestra imágenes de la ejecución: