Universidad de Castilla-La Mancha

Dispositivos y redes inalámbricos

# SIGFOX: MONITOR DE SALUD

*Antonio Morán Muñoz*
*Pablo Olivas Auñón*

CURSO ACADÉMICO 2018/2019

22 de abril de 2019

# Tabla de contenidos



# Índice de figuras

# 1. Objetivos del proyecto

El objetivo del proyecto es poder monitorizar de forma sencilla y establecer un sistema de alertas para una serie de valores obtenidos de un paciente (por ejemplo número de pulsaciones por minutos, temperatura corporal, tensión o nivel de glucosa en sangre), no importando la ubicación del mismo.

# 2. Diseño del proyecto

La tecnología Sigfox nos permite enviar mensajes desde un emisor en casi cualquier parte del país hasta una desde la que podemos tratarlos y enviar alertas a multiples receptores. Esto es lo que nos ha permitido crear este monitor de salud basado en dicha tecnología.

Nuestro monitor de salud permite; tras obtener las pulsaciones, temperatura, tensión y nivel de glucosa en sangre; enviar estos dato con una cierta frecuencia a la nube de Sigfox, donde serán reenviados al centro de IOT de Azure para finalmente ser recibidos por nuestro servidor. Con los datos ya en manos del usuario final se distinguen dos usos diferenciados. Primero, tenemos una sistema de alertas para los receptores finales mediante un bot de Telegram, que recibirán si hay algún valor extraño en los datos obtenidos del paciente. El segundo uso consiste en una pequeña base de datos de fácil uso para poder visualizar el historial de todas las constantes del paciente.

# 3. Desarrollo del proyecto

## 3.1. Arduino

En el Listing 1 podemos ver el código que se encargará de aleatorizar los valores para los diferentes parámetros que vayamos a monitorizar. El programa es muy sencillo. En la función de *setup* configuramos el dispositivo establecemos la semilla para generar valores aleatorio. Tras esto, en la función *loop*, lo que tenemos que hacer es declarar los diferentes parámetros a monitorizar e iniciarlos a un valor aleatorio. Tras esto, creamos un paquete en el que meteremos toda esa información y la enviaremos al backend de SigFox. La instrucción de la línea 47 suspenderá el dispositivo durante 10 segundos antes de volver a enviar un nuevo paquete.

```
1
2 #include <SigFox.h>
3 #include <ArduinoLowPower.h>
4
5 void setup() {
6   randomSeed(analogRead(0));
7   Serial.begin(9600);
8   SigFox.begin();
9   SigFox.debug();
10
11   /*
12   short pulsaciones = random(50,130);
13   short temperatura = random(30,45);
14   short tension = random(70,130);
15   short glucosa = random(59,150);
16
17   Serial.println("Waiting for downlink");
18   SigFox.beginPacket();
19   SigFox.write(pulsaciones);
20   SigFox.write(temperatura);
21   SigFox.write(tension);
22   SigFox.write(glucosa);
23   SigFox.endPacket();*/
24   SigFox.end();
25 }
26 void loop() {
27   while (SigFox.available()) {
28   Serial.print("0x");
29   Serial.println(SigFox.read(), HEX);
30    }
31
32   SigFox.begin();
33
34   short pulsaciones = random(50,130);
35   short temperatura = random(34,41);
36   short tension = random(70,130);
37   short glucosa = random(59,150);
38
39   SigFox.beginPacket();
40   SigFox.write(pulsaciones);
41   SigFox.write(temperatura);
42   SigFox.write(tension);
43   SigFox.write(glucosa);
44   SigFox.endPacket();
45
46   //LowPower.sleep(15 * 60 * 1000);
47   LowPower.sleep(10000);
48   SigFox.end();
49 }
```

Listing 1: Codigo de Arduino

## 3.2. Configuración del callback

El siguiente paso es configurar un nuevo callback en el backend de SigFox, que se encargue de enviar la información recibida del dispositivo y la redirija a Azure. Para ello, primero debemos crear una instancia de Azure IoT, con la configuración mostrado en la Figura 1.

En la Figura 3 vemos la configuración del callback. En primer lugar, tenemos que indicar que es un uplink, ya que mandaremos la información a otro servicio. En segundo lugar, en el campo *Custom payload config* indicamos cómo se deberán interpretar los datos encapsulados en el paquete que recibamos. En el campo *Connection string* tenemos que indicar la cadena de conexión al IoT Hub de Azure donde se enviarán los datos, la cual ha sido obtenida de la instancia de Azure creada anteriormente tal y como se muestra en la Figura 2. Por último, indicaremos el formato de salida de los datos en el campo *JSON body*.


Figura 1: Configuración del callback

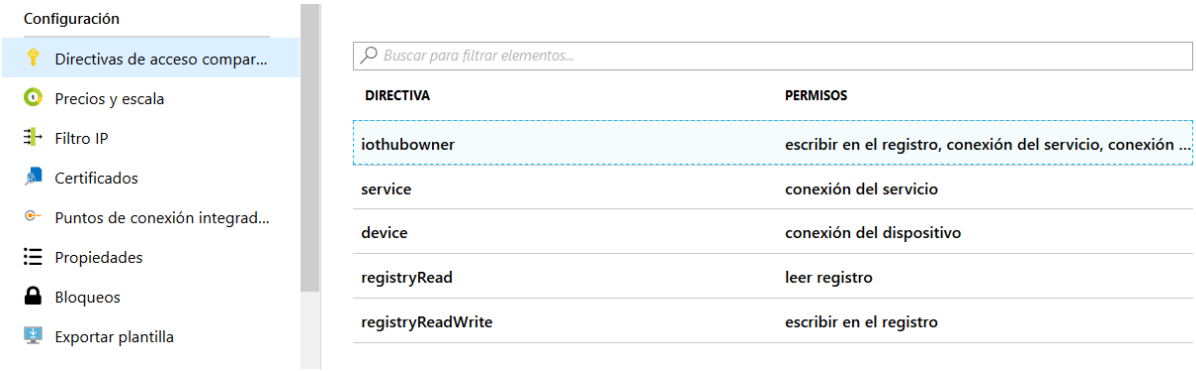


Figura 2: Configuración del callback

## 3.3. Configuración del Bot

A continuación, tenemos que crear el Bot que se encargará de la lógica del monitor. En el Listing 2 se puede ver el código, que permite enviar la información recogida a una de Excel y mandar las notificaciones correspondientes a los valores anómalos.


Figura 3: Configuración del callback

```
process.env["NTBA_FIX_319"] = 1;
const TelegramBot = require('node-telegram-bot-api');
const settings = require("./Settings.json");
const telegramToken = settings.telegramToken;
const azureToken = settings.azureToken;
const bot = new TelegramBot(telegramToken, {polling: true});
const excel = require('excel4node');
var chatID = 0;

var date = new Date();
var workbook = new excel.Workbook();
var worksheet = workbook.addWorksheet('Sheet 1');
var style = workbook.createStyle({
    font: {
      color: '000000',
      size: 12
    },
    numberFormat: '#0; (#0); -'
});
worksheet.cell(1,2).string('Pulsaciones').style(style);
worksheet.cell(1,3).string('Temperatura').style(style);
worksheet.cell(1,4).string('Tensión').style(style);
worksheet.cell(1,5).string('Glucosa').style(style);
var i = 2;

const { EventHubClient, EventPosition } = require('@azure/event-hubs');


var printError = function (err) {
    console.log(err.message);
};

var printMessage = function (message) {
    console.log('Mensaje recibido desde dispositivo Sigfox');
    console.log(message.body);
```

```
36      var current_hour = date.getHours().toString()+':'+date.getMinutes().
    toString();
37      worksheet.cell(i,1).string(current_hour).style(style);
38      worksheet.cell(i,2).string(message.body.pulsaciones).style(style);
39      worksheet.cell(i,3).string(message.body.temperatura).style(style);
40      worksheet.cell(i,4).string(message.body.tension).style(style);
41      worksheet.cell(i,5).string(message.body.glucosa).style(style);
42      i = i+1;
43      workbook.write('Excel.xlsx');
44
45      //Pulsaciones
46      if(message.body.pulsaciones < 60){
47          bot.sendMessage(chatId, current_hour+' - Alerta, pulsaciones bajas
    ('+message.body.pulsaciones+')');
48      }
49      if(message.body.pulsaciones > 120){
50          bot.sendMessage(chatId, current_hour+' - Alerta, pulsaciones altas
    ('+message.body.pulsaciones+')');
51      }
52
53      //Temperatura
54      if(message.body.temperatura < 35){
55          bot.sendMessage(chatId, current_hour+' - Alerta, temperatura bajas
    ('+message.body.temperatura+')');
56      }
57      if(message.body.temperatura > 39){
58          bot.sendMessage(chatId, current_hour+' - Alerta, temperatura altas
    ('+message.body.temperatura+')');
59      }
60
61      //Tension
62      if(message.body.tension < 80){
63          bot.sendMessage(chatId, current_hour+' - Alerta, tensión baja ('+
    message.body.tension+')');
64      }
65      if(message.body.tension > 120){
66          bot.sendMessage(chatId, current_hour+' - Alerta, tensión alta ('+
    message.body.tension+')');
67      }
68
69      //Glucosa
70      if(message.body.glucosa < 69){
71          bot.sendMessage(chatId, current_hour+' - Alerta, glucosa baja ('+
    message.body.glucosa+')');
72      }
73      if(message.body.glucosa > 140){
74          bot.sendMessage(chatId, current_hour+' - Alerta, glucosa alta ('+
    message.body.glucosa+')');
75      }
76  };
77
78  var ehClient;
79  EventHubClient.createFromIotHubConnectionString(azureToken).then(function (
    client) {
80      console.log("Conectado a Azure IOT Hub.");
81      ehClient = client;
82      return ehClient.getPartitionIds();
83  }).then(function (ids) {
```

```
84      return ids.map(function (id) {
85          return ehClient.receive(id, printMessage, printError, {
        eventPosition: EventPosition.fromEnqueuedTime(Date.now()) });
86      });
87  }).catch(printError);
88
89
90  bot.onText(/\/conectar/, (msg, match) => {
91      chatId = msg.chat.id;
92      bot.sendMessage(chatId, 'Bienvenido al monitor de salud');
93  });
94
95  bot.onText(/\/about/, (msg, match) => {
96      bot.sendMessage(chatId, 'Github: https://github.com/NecaX/
        InalambricosSigfox');
97  });
```

Listing 2: Codigo del bot

# 4. Resultados y conclusiones

Tras toda la configuración previa, podemos conectar el Arduino, meterle el código y ejecutarlo. El resultado de esto se puede ver en las siguientes figuras. En la Figura 4 podemos observar cómo se han recibido las notificaciones por valores anómalos
.

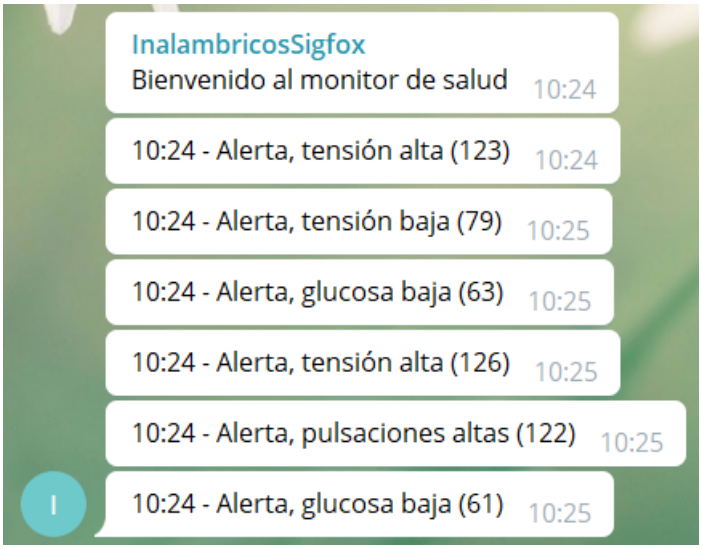


Figura 4: Notificaciones por Telegram

Una vez finalizado el proyecto, podemos establecer una serie de conclusiones acerca del uso de la tecnología SigFox. Esta tecnología es ideal para la esfera del IoT, como hemos visto por su capacidad de transmitir una pequeña cantidad de datos muy específicos que podrían representar cualquier característica del mundo real (en nuestro caso una serie de constantes vitales). Esta limitada capacidad de envío de datos permite un consumo mucho menor de energía, haciéndolo perfecto para aparatos que requieran de una gran

autonomía. Por último, hay que destacar su área de cobertura que, si bien las hay mejores, es suficiente para abarcar numerosos casos de uso.

# Referencias

[1] Microsoft. Inicio rapido: Eniío de telemetria desde un dispositivo a un centro de iot y su lectura con una aplicacion de back-end (c sharp), 2019. URL `https://docs.microsoft.com/es-es/azure/iot-hub/quickstart-send-telemetry-dotnet`. Último acceso: 14-04-2019.