



## TRABAJO AUTÓNOMO 4

APLICACIÓN MÓVIL CON CONEXIÓN A REDES “LPWAN” Y SENSORES IOT

ELABORADO POR: ADRIANA COLLAGUAZO JARAMILLO/LUIS ZUÑIGA ROSADO

ITINERARIO: APLICACIONES MÓVILES Y SISTEMAS TELEMÁTICOS

CARRERA DE INGENIERÍA EN TELEMÁTICA

FIEC-ESPOL

**Trabajo Autónomo:**

**Objetivo de Aprendizaje:** Desarrollar aplicaciones móviles sencillas considerando las características de la programación de dispositivos móviles.

**Recursos:** Android Studio, Firebase, Sigfox DEV Kit Thinxtra, Ubidots.

**Duración:** 8 horas

**INTRODUCCIÓN:**

Sigfox es una alternativa de amplio alcance, que en términos de alcance está entre Wifi y la comunicación móvil. Utiliza bandas ISM, que se pueden utilizar sin necesidad de adquirir licencias. Sigfox responde a las necesidades de muchas aplicaciones M2M que funcionan con una batería pequeña y solo requieren niveles menores de transferencia de datos, allí donde Wifi se queda demasiado corto y la comunicación móvil es muy cara y consume demasiada energía. Los tres pilares de Sigfox son: bajo coste, eficiencia y alcance global.

**Beneficios de Sigfox**

- Bajo precio en los costes de operación
- Un gran rango de kilómetros y una excelente cobertura

**Desventajas de Sigfox**

- Velocidad baja de bits, alrededor de unos 100 bit/s
- Comunicación unidireccional (conexión de subida, conexión de bajada muy limitada)
- Protección inferior a las interferencias

**Limitaciones Sigfox**

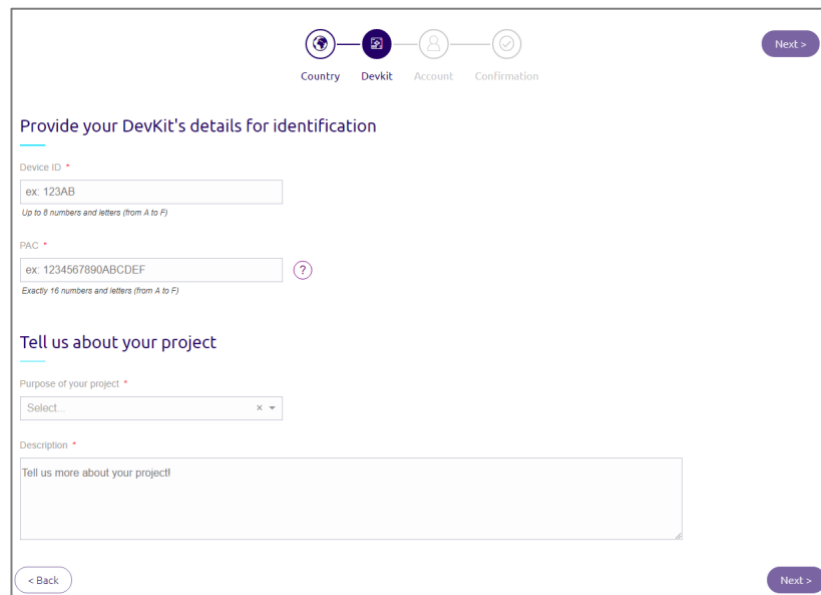
- Longitud máxima del mensaje de 12 Bytes
- Número máximo de mensajes de 140 al día
- Velocidad de transferencia de datos de 100 bit/s

**ACTIVIDADES:****Paso 1 (Informativo): Registro del dispositivo Thinxtra en una cuenta del backend de SIGFOX.**

- Ingresar a la siguiente URL <https://buy.sigfox.com/activate>.
- Colocar en el buscador el país donde se desea activar el módulo y visualizar si hay cobertura.

The screenshot shows the Sigfox 'buy' page during the activation process. The progress bar indicates the 'Country' step is current. The search for 'ecuador' has returned results for WND Ecuador, showing their location and contact information. The page also provides instructions on how to proceed based on whether the country is active, inactive, or not listed.

- Seleccionar la opción *Next*, completamos un formulario indicando el identificador del módulo (Device ID) y el código de autorización (PAC), los cuales son provistos por el fabricante.



Country Devkit Account Confirmation

**Provide your DevKit's details for identification**

Device ID \*  
ex: 123AB  
Up to 8 numbers and letters (from A to F)

PAC \*  
ex: 1234567890ABCDEF ?  
Exactly 16 numbers and letters (from A to F)

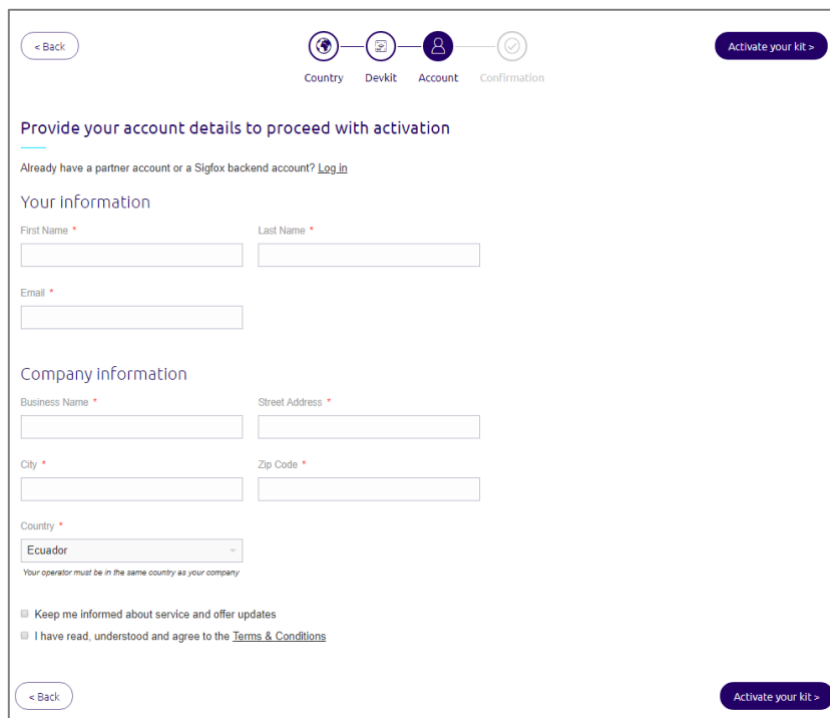
**Tell us about your project**

Purpose of your project \*  
Select...

Description \*  
Tell us more about your project!

< Back Next >

d) Se crea la nueva cuenta en el backend de SIGFOX o se ingresa una cuenta creada anteriormente.



< Back Country Devkit Account Confirmation Activate your kit >

**Provide your account details to proceed with activation**

Already have a partner account or a Sigfox backend account? [Log in](#)

**Your information**

First Name \* Last Name \*  
Email \*

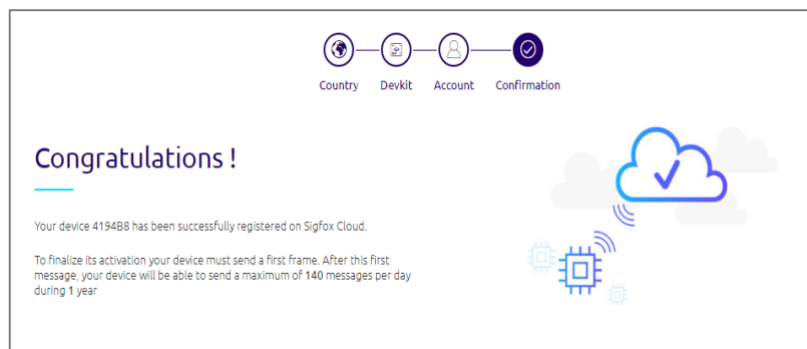
**Company information**

Business Name \* Street Address \*  
City \* Zip Code \*  
Country \*  
Ecuador  
Your operator must be in the same country as your company

☐ Keep me informed about service and offer updates  
☐ I have read, understood and agree to the [Terms & Conditions](#)

< Back Activate your kit >

e) Al seleccionar “*Activate your kit >*” aparecerá una ventana indicando registro exitoso.



Country Devkit Account Confirmation

**Congratulations !**

Your device 4194B8 has been successfully registered on Sigfox Cloud.

To finalize its activation your device must send a first frame. After this first message, your device will be able to send a maximum of 140 messages per day during 1 year

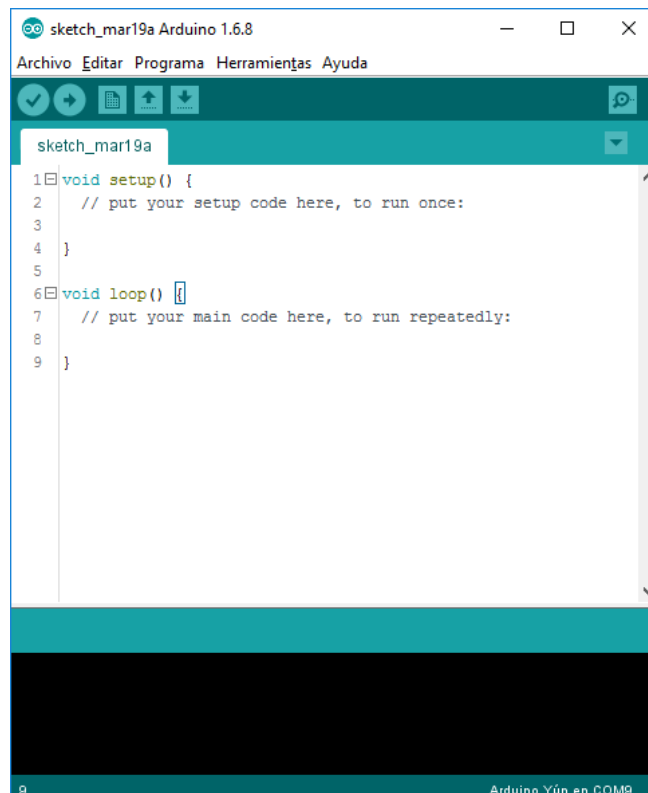
**Paso 2: Creación de código Arduino.**

Nota: Si usted no tiene instalado el IDE de Arduino lo puede descargar en este enlace <https://www.arduino.cc/en/Main/Software>

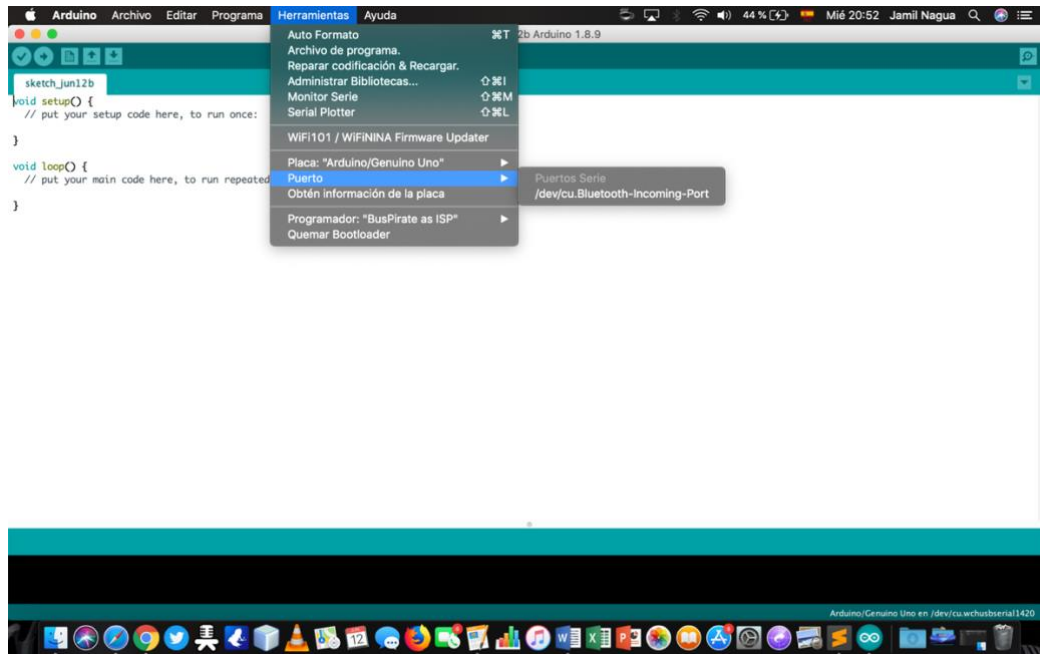
- a) Verificar la conexión del Arduino con el Thinxtra. Los jumpers deben estar conectado como se muestra en la imagen, así funcionará el Arduino con el Thinxtra. Para compilar el código en el Arduino es necesario desconectar los jumper del recuadro rojo de la imagen, ya que estos están conectados al pin TX y RX del Arduino y si estos se encuentran en uso no se puede compilar código en el Arduino. En este paso, conecte la antena al Thinxtra.



- b) Abrir el IDE de Arduino.



- c) Seleccionar la opción "Herramientas" y elegimos el puerto y modelo de nuestra placa. *Es posible que no se identifique en que puerto esta el arduino esto se debe a que el Thinxtra tiene un chip que no es compatible con el IDE de arduino, por eso elija el puerto COM (comunicaciones) que le aparezca en su máquina, en este caso es el COM4 (muy posible sea el que este conectado el arduino ya que es el com libre que tiene la maquinas del laboratorio).*



- d) Copiar el siguiente código y cargarlo al Arduino.

```
void setup() {
  Serial.begin(9600);
  Serial.println("AT$RC"); //Se indica el tipo de conexión para nuestra zona
  delay(100);
  Serial.println("AT$SF=0123CAFE"); //Se envía el mensaje en formato hexadecimal
}
void loop() {}
```

- e) Acceder al backend de Sigfox con las siguientes credenciales:

- <https://backend.sigfox.com/auth/login>
- Usuario: lab-telematica@fiee.espol.edu.ec
- Contraseña: L4bt3l3fiee@

- f) En la pestaña DEVICE se encuentran todos los dispositivos registrados en la cuenta.  
g) Seleccione en la columna ID su dispositivo de acuerdo con el Device Type.

Count: 7 / 7

Communication status	Device type	Group	Id	Last seen	Name	Token state
<input type="radio"/>	Thinextra_DevKit_7	ESPOL-TELEIATICA	3E3E17	2019-06-05 11:41:40	Thinextra_DevKit_7-device	<input checked="" type="checkbox"/>
<input type="radio"/>	Thinextra_DevKit_1	ESPOL-TELEIATICA	41241F	2019-06-05 12:28:27	Thinextra_DevKit_1-device	<input checked="" type="checkbox"/>
<input type="radio"/>	Thinextra_DevKit_4	ESPOL-TELEIATICA	4146E1	2019-06-05 12:01:53	Thinextra_DevKit_4-device	<input checked="" type="checkbox"/>
<input type="radio"/>	Thinextra_DevKit_5	ESPOL-TELEIATICA	414805	2019-06-05 12:13:57	Thinextra_DevKit_5-device	<input checked="" type="checkbox"/>
<input type="radio"/>	Thinextra_DevKit_3	ESPOL-TELEIATICA	417922	2019-06-05 12:19:25	Thinextra_DevKit_3-device	<input checked="" type="checkbox"/>
<input type="radio"/>	Thinextra_DevKit_2	ESPOL-TELEIATICA	418E45	2019-06-05 11:41:56	Thinextra_DevKit_2-device	<input checked="" type="checkbox"/>
<input type="radio"/>	Thinextra_DevKit_6	ESPOL-TELEIATICA	4194B8	2019-06-05 11:42:46	Thinextra_DevKit_6-device	<input checked="" type="checkbox"/>

h) Entre las opciones de la derecha, seleccione MESSAGES y observe si su mensaje llego correctamente.

page 1

Time	Data / Decoding	LQI	Callbacks	Location
2019-06-05 11:41:40	Scbf2410090c9bf temp: 26.32			
2019-06-04 14:18:54	Scbf2410090c9bf temp: 26.32			
2019-06-04 13:57:43	ae47db41 temp: 27.41			

### Paso 3 : Sensar y enviar la temperatura.

a) Descargar las librerías en el directorio `../Documentos/Arduino/libraries/` de la siguiente ruta <https://github.com/Thinextra/Xkit-Sample>

b) Utilizar el siguiente código y revisar los comentarios:

```
// Include libairies
#include <WISOL.h>
#include <Tsensors.h>
#include <Wire.h>
#include <math.h>
Isigfox *Isigfox = new WISOL();
Tsensors *tSensors = new Tsensors();
typedef union{
    uint16_t number;
    uint8_t bytes[2];
} UINT16_t;

void setup() {
```

```

Wire.begin();
Wire.setClock(100000);
// Init serial connection between Arduino and Modem
Serial.begin(9600);
uint16_t tempt;
// WISOL modem test
Isigfox->initSigfox();
Isigfox->testComms();
// Init sensors on Thinxtra Module
tSensors->initSensors();
// Init an interruption on the button of the Xkit
tSensors->setButton(buttonIR);
}
// Infinite loop of the program
void loop() {
//obtenemos la aceleración en el eje x del thinxtra
float axeX = getAxeX();
Serial.print("Check Axe X: "); Serial.println(axeX);
if (axeX <= -0.4 or axeX >= 0.4) {
//obtenemos la temperatura que nos da un tipo de dato float
float t = tSensors->getTemp();
//hacemos un arreglo de bytes para poder enviar byte por byte mas adelante
byte *float_byte = (byte *)&t;
byte *float_axeXb = (byte *)&axeX;

//indicamos el tamaño de nuestro mensaje sabiendo que el máximo tamaño es
de 12 bytes
const uint8_t payloadSize = 8;
uint8_t buf_str[payloadSize];
buf_str[0] = float_byte[0];
buf_str[1] = float_byte[1];
buf_str[2] = float_byte[2];
buf_str[3] = float_byte[3];
buf_str[4] = float_axeXb[0];
buf_str[5] = float_axeXb[1];
buf_str[6] = float_axeXb[2];
buf_str[7] = float_axeXb[3];

Send_Pload(buf_str, payloadSize);
// Wait 20s
delay(20000);
}
delay(1000);
}
// Return the acceleration on Axe X
float getAxeX() {
acceleration_xyz *xyz_g;
xyz_g = (acceleration_xyz *)malloc(sizeof(acceleration_xyz));
tSensors->getAccXYZ(xyz_g);
float axeX = (float)xyz_g->x_g;
free(xyz_g);
return axeX;
}
// Get the temperature
float getTemp() {
float temp = round(tSensors->getTemp() * 10) / 10.0;
Serial.print("Sending Temp: "); Serial.println(temp);
return temp;
}
// SendPayload Function => Send messages to the Sigfox Network
void Send_Pload(uint8_t *sendData, int len) {

```

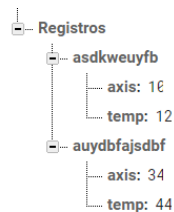
```

recvMsg *RecvMsg;
RecvMsg = (recvMsg *)malloc(sizeof(recvMsg));
Isigfox->sendPayload(sendData, len, 0, RecvMsg);
for (int i = 0; i < RecvMsg->len; i++) {
    Serial.print(RecvMsg->inData[i]);
}
Serial.println("");
free(RecvMsg);
}
// Button Interruption
void buttonIR(){
    float temp = getTemp();
    Send_Pload((const char*)&temp, sizeof(temp));
}

```

#### Paso 4: Creación de mensajes Callbacks hacia Base de datos externa

- a) Primero accedemos al proyecto en Firebase 'trabajo-autonomo-3' (asegurese tener permisos de acceso). Nótese que cada grupo de registros tiene un id aleatorio.



- b) Nos dirigimos al backend de sigfox seleccionamos la pesta Device Type y escoger el nombre correspondiente a su grupo.

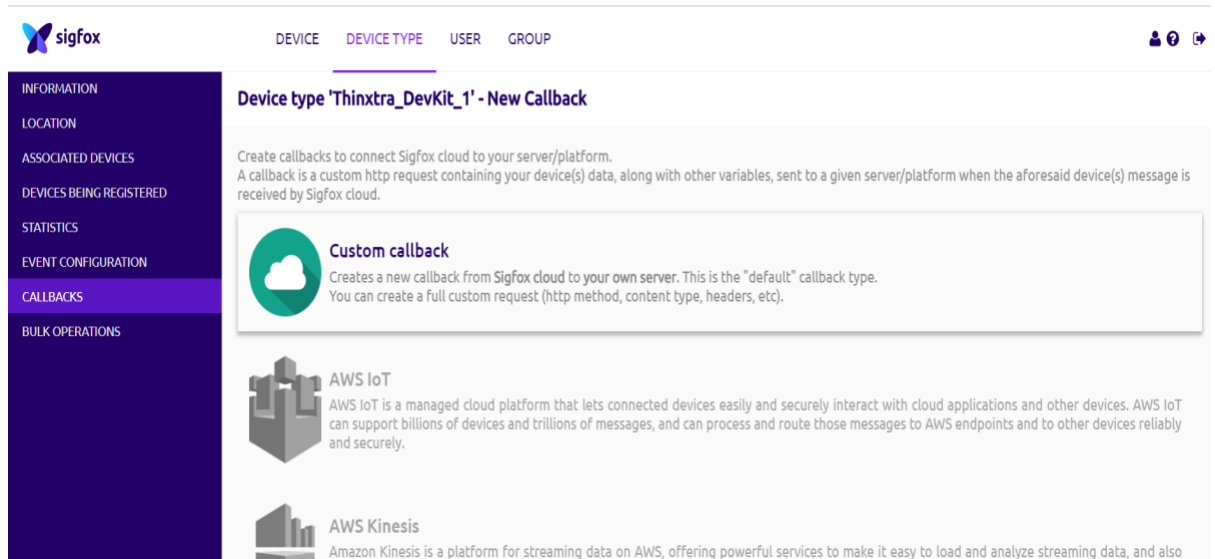
page 1

Description	Display type	Group	Keep alive	Name
DevKit 1 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_1
DevKit 2 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_2
DevKit 3 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_3
DevKit 4 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_4
DevKit 5 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_5
DevKit 6 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_6
DevKit 7 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_7

page 1

- c) Seleccionar la opción CALLBACKS y luego en New
- d) Seleccionar Custom callback





e) Configurar el callback de la siguiente manera

Donde dice Custom Payload config

Indicamos el nombre que tendrá nuestra variable y el tipo de dato que es

The screenshot shows the 'Callbacks' configuration form. It has several fields: 'Type' (set to DATA), 'Channel' (set to URL), 'Send duplicate' (unchecked), and 'Custom payload config' (set to 'temp::float:32:little-endian axi::float:32:little-endian'). There is a help icon (?) next to the custom payload config field.

La variables tomaron los datos en orden, temp es la primera tomara los primeros byte 4 en este caso ya que es un float en la imagen de abajo se ven los datos en formato hexadecimal por lo que cada 2 números representa un byte

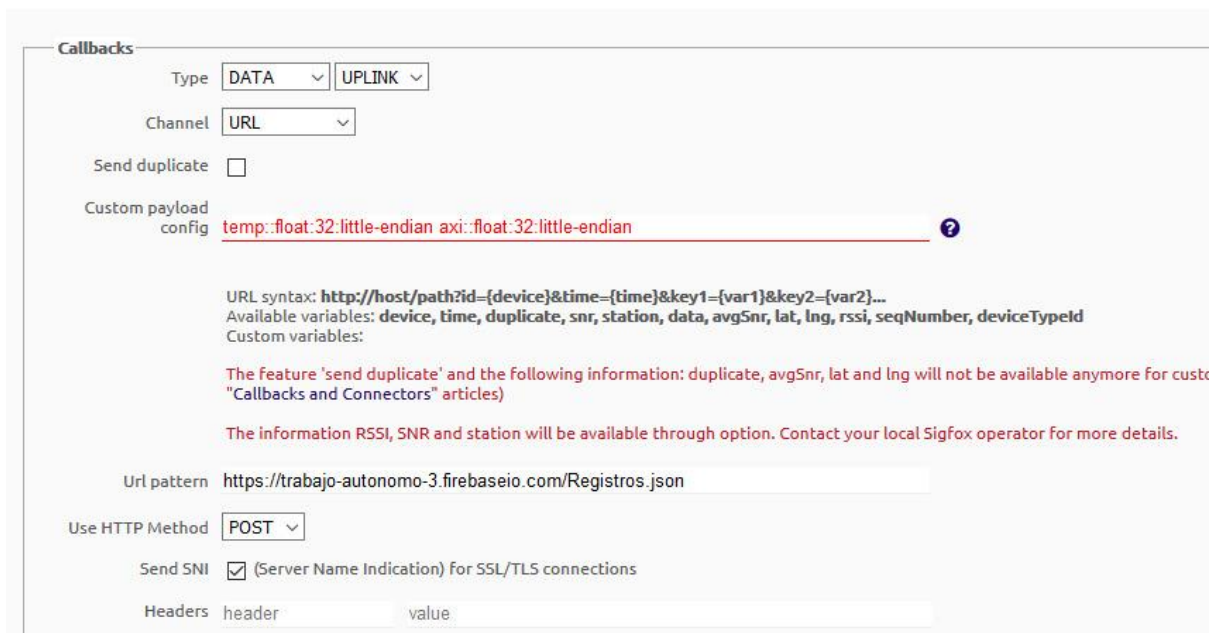
Data / Decoding	LQI	Callbacks	Location
1f85e54100b01d40 temp: 28.69			

En Url pattern ponemos la dirección hacia la cual enviaremos nuestra información será la siguiente:

<https://trabajo-autonomo-3.firebaseio.com/Registros.json>

El método HTTP que usaremos es **POST** para guardar datos

## Device type Thinxtra\_DevKit\_5 - Callback edition



Callbacks

Type: DATA UPLINK

Channel: URL

Send duplicate: ☐

Custom payload config: temp::float:32:little-endian axi::float:32:little-endian ?

URL syntax: <http://host/path?id={device}&time={time}&key1={var1}&key2={var2}...>  
 Available variables: **device, time, duplicate, snr, station, data, avgSnr, lat, lng, rssi, seqNumber, deviceTypeId**  
 Custom variables:

The feature 'send duplicate' and the following information: duplicate, avgSnr, lat and lng will not be available anymore for custom "Callbacks and Connectors" articles)

The information RSSI, SNR and station will be available through option. Contact your local Sigfox operator for more details.

Url pattern: https://trabajo-autonomo-3.firebaseio.com/Registros.json

Use HTTP Method: POST

Send SNI: ☒ (Server Name Indication) for SSL/TLS connections

Headers: header value

Indicamos que el tipo de conexión es por una aplicación json en el body  
 Cada variable las publicamos como {customData#Variable}

### Body:

```
{
  "temp": "{customData#temp}",
  "axi": "{customData#axi}"
}
```



Content type: application/json

Body:

```
{
  "temp": "{customData#temp}",
  "axi": "{customData#axi}"
}
```

Después de ser enviado debe presentarse en la base de firebase.

### Paso 5: observar el mensaje desde la aplicación del teléfono

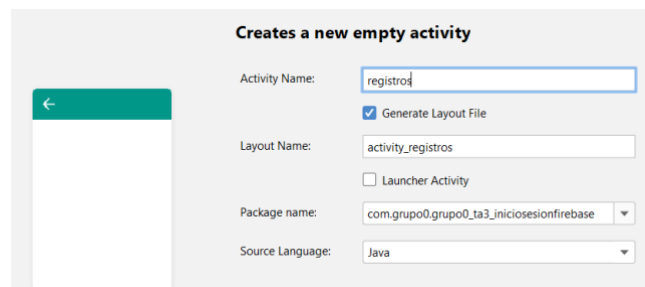
**NOTA:** Utilizamos la aplicación del taller autónomo n3 (Se requiere haber iniciado sesión con Firebase para poder acceder a la base de datos.)

- Agregamos un botón en el botón de perfil para poder acceder a una **nueva actividad**. En la actividad veremos los registros de la base de datos.

*App/res/layout/activity\_perfil\_usuario.xml*



- b) Creamos una nueva actividad, llamada: registros.



Agregamos la función `irRegistros()`. Y se la asignamos al boton Ver registros.

### App/java/PerfilUsuario.java

```
public void irRegistros(View view){
    Intent intent = new Intent(this, registros.class);
    startActivity(intent);
}
```

### App/res/activity\_perfil\_usuario.xml

```
<Button
    android:id="@+id/btnVerRegistros"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="irRegistros"
    android:text="Ver Registros" />
```

- c) Dentro de la nueva actividad Registros.java obtenemos los datos dentro de nuestra rama

```
public class registros extends AppCompatActivity {

    DatabaseReference db_reference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registros);
        db_reference = FirebaseDatabase.getInstance().getReference().child("Registros");
        leerRegistros();
    }

    public void leerRegistros() {}
}
```

- d) Diseñamos la vista para poder incluir los valores de temperatura. Modificamos el template `activity_registros.xml` (Usamos Layouts como contenedores.)



Código:

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".registros" >

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Registros"
        android:textSize="30sp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <LinearLayout
            android:id="@+id/TituloTemp"
            android:layout_width="211dp"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <TextView
                android:id="@+id/textView3"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="temperaturas" />
        </LinearLayout>

        <LinearLayout
            android:id="@+id/TituloAxis1"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <TextView
                android:id="@+id/TituloAxis2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Axis" />
        </LinearLayout>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <LinearLayout
            android:id="@+id/ContenedorTemp"
            android:layout_width="211dp"
            android:layout_height="match_parent"
            android:orientation="vertical" />

        <LinearLayout
            android:id="@+id/ContenedorAxis"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" />
    </LinearLayout>
</LinearLayout>
```

e) Implementamos la función leer registros para cargarlos desde la base de datos.

```
public void leerRegistros() {
    db_reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                mostrarRegistrosPorPantalla(snapshot);
            }
        }

        @Override
        public void onCancelled(DatabaseError error) {
            Sytem.out.println(error.toException());
        }
    });
}

public void mostrarRegistrosPorPantalla(DataSnapshot snapshot){}
```

f) Implementamos la función mostrar Registros para imprimir los resultados.

```
public void mostrarRegistrosPorPantalla(DataSnapshot snapshot){
    LinearLayout contTemp = (LinearLayout) findViewById(R.id.ContenedorTemp);
    LinearLayout contAxis = (LinearLayout) findViewById(R.id.ContenedorAxis);

    String tempVal = String.valueOf(snapshot.child("temp").getValue());
    String axisVal = String.valueOf(snapshot.child("axis").getValue());

    TextView temp = new TextView(getApplicationContext());
    temp.setText(tempVal+" C");
    contTemp.addView(temp);

    TextView axis = new TextView(getApplicationContext());
    axis.setText(axisVal);
    contAxis.addView(axis);
}
```

RESULTADO:



## TAREA DESAFIO

### Creación de mensajes *Callbacks* hacia Ubidots.

- Desde el backend seleccionar la pestaña Device Type y escoger el correspondiente a su grupo.

page 1

Description	Display type	Group	Keep alive	Name	
DevKit 1 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_1	
DevKit 2 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_2	
DevKit 3 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_3	
DevKit 4 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_4	
DevKit 5 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_5	
DevKit 6 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_6	
DevKit 7 (Thinextra)	Custom	ESPOL-TELEMATICA	N/A	Thinextra_DevKit_7	

page 1

- b) Seleccionar la opción CALLBACKS y luego en New
- c) Seleccionar Custom callback

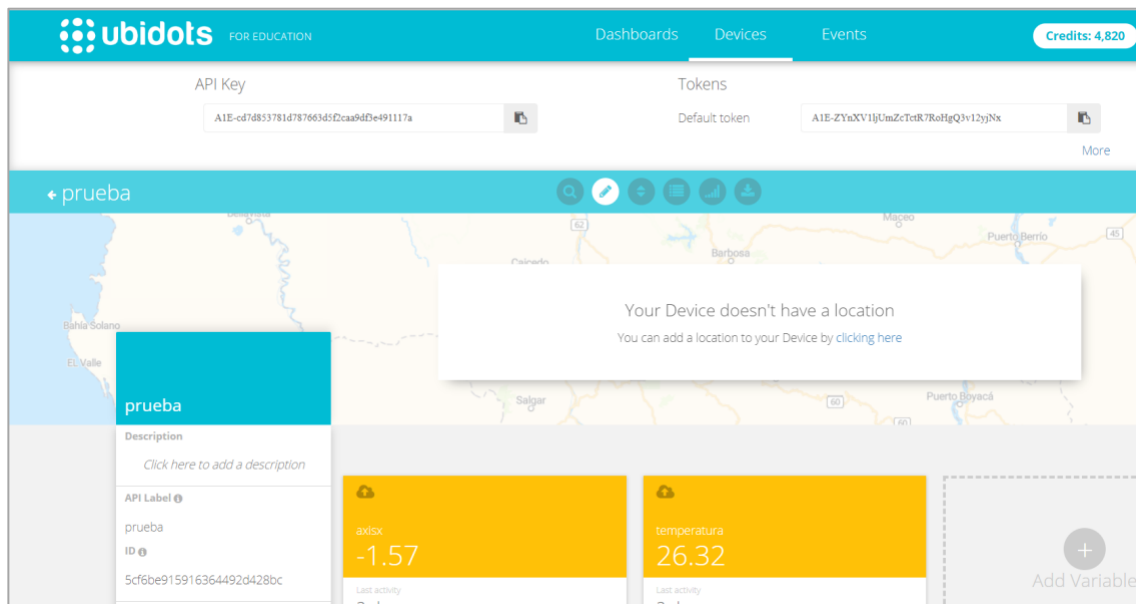
The screenshot shows the Sigfox web interface. The top navigation bar includes 'DEVICE', 'DEVICE TYPE', 'USER', and 'GROUP'. The left sidebar has a menu with 'CALLBACKS' highlighted. The main content area is titled 'Device type 'Thinextra\_DevKit\_1' - New Callback'. It contains a description of callbacks and three main options: 'Custom callback' (described as the default), 'AWS IoT', and 'AWS Kinesis'.

- d) Configurar el callback de la siguiente manera, donde pone YYYY ponga un nombre y XXXX es el token generado de su cuenta de Ubidots.

The screenshot shows the 'Callback edition' page for device type 'Thinextra\_DevKit\_7'. The form includes the following fields and values:

- Type: DATA
- Channel: URL
- Send duplicate: ☐
- Custom payload config: temp:float:32 little-endian axi:float:32 little-endian
- URL pattern: https://things.ubidots.com/api/v1.6/devices/prueba/?token=A1E-ZYnXV1lUmZcTcf
- Use HTTP Method: POST
- Send SNI: ☒ (Server Name Indication) for SSL/TLS connections
- Headers: header value
- Content type: application/json
- Body:
 

```
{
    "temperature": { "value": "{customData#temp}" },
    "axisx": { "value": "{customData#axi}" }
  }
```



## FORMATO DEL TRABAJO

El trabajo autónomo será desarrollado en el siguiente formato:

- Nombre del archivo: AMST\_Trabajo Autónomo A\_Grupo B\_Apellido1\_Apellido2\_Apellido3
- (\*) Siendo A el número del trabajo y B el número del grupo
- Nombre de la materia
- Título del trabajo: Ejemplo: Trabajo Autónomo A - Tema
- Nombre de la profesora
- Número de grupo
- Nombres/Apellidos de los integrantes del grupo que hayan desarrollado el trabajo
- Fecha de inicio y fin del trabajo
- Resultados de las actividades planteadas: Explicación de las actividades ejecutadas, incluyendo las imágenes del proceso. Además, incluir el enlace del repositorio del proyecto en Github y el archivo ejecutable (apk) de la aplicación móvil.
- Conclusiones y Recomendaciones: Respecto a lo aprendido durante el desarrollo del trabajo.
- Referencias bibliográficas: Colocar los documentos, enlaces web o libros consultados.