

MONITOREO DE UN TRANSMISOR FM

MANUAL TÉCNICO



- **INTRODUCCIÓN**

La propuesta de proyecto trata sobre un sistema de monitoreo de parámetros básicos, como lo son la potencia y potencia reflejada de un transmisor FM para radio. Este sistema funcionará en base a una aplicación desarrollada en Android Studio en el cual fue realizado todo el código con el fin de que se tenga un diseño de fácil interacción para el usuario. Así mismo, se hizo uso de herramientas como lo es un ARDUINO UNO en donde se tiene el código basado en Python para poder mandar la información recolectada a la aplicación móvil con el fin de que toda dicha información se mantenga almacenada y así el usuario pueda realizar sus respectivos análisis comparando mediciones pasadas.

Este proyecto nace de la idea de ayudar a las empresas radiodifusoras con el objetivo de reducir el tiempo que los empleados de este tipo de lugares necesitan en tomar medidas de ciertos parámetros puesto que, en varias ocasiones, para poder realizar dicha acción, es necesario que ellos se desplacen una gran distancia entre su lugar de trabajo y su destino, siendo que en ocasiones dicho destino se encuentra en lugares muy elevados teniendo como resultado un gran gasto de tiempo solo en el trayecto. Todo esto también conlleva a mayores gastos económicos como puede ser el transporte, ya sea público o que se tenga vehículo propio y en la adquisición de herramientas adicionales que pueden llegar a necesitar para realizar mediciones exactas.

- **DISEÑO**

Diseño de Hardware



Figure 1: Diseño Hardware terminado

Se tiene una sección que representa a un transmisor FM el cual está compuesto por un excitador y un amplificador. En general, estos equipos tienen un puerto remoto en el cual se sacarán los parámetros a medir para luego ser calibrados a través de la tarjeta Arduino en el cual hay cables que van conectados a los pines analógicos del mismo. Esta calibración es posible debido a que,

se ha creado un código que leerá los datos enviados para luego escalarlos. tUna vez que los datos hayan sido procesados, estos serán enviados a la Raspberry Pi, el cual a través de un código realizado en Python mandará información a la nube de Google, pero adicionalmente, también se han impreso los resultados en el terminal de la Raspberry.

Diseño de Software

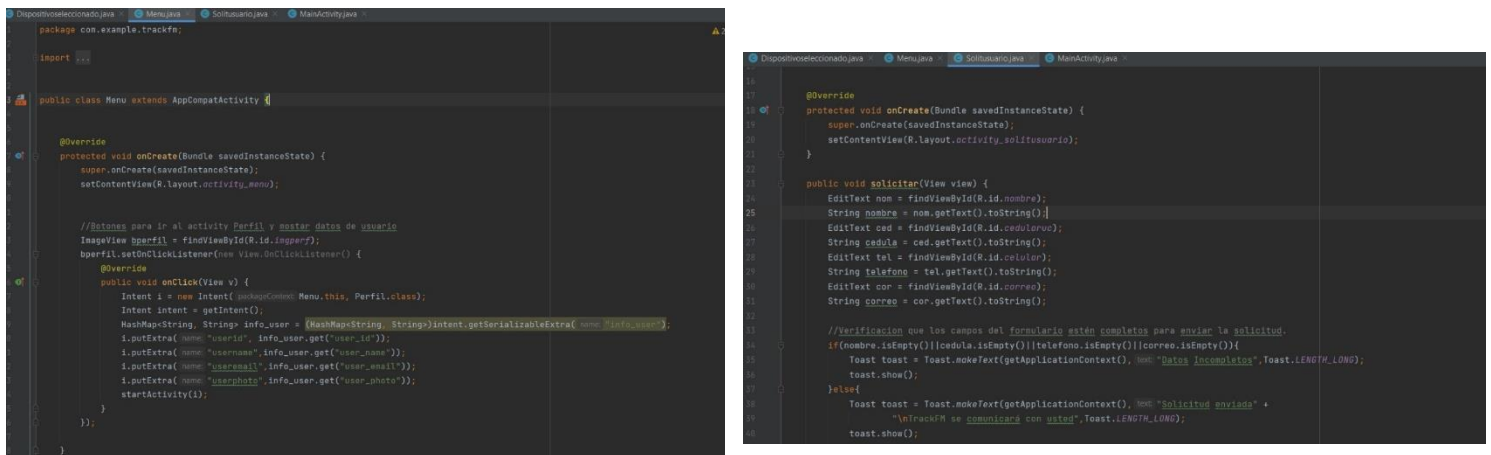


Figure 2: Código del software realizado en Android Studio

En cuanto a los archivos .java, se tiene al ActivityIniciodeSesion que corresponde al inicio de sesión en pantalla al dar click en el botón “Iniciar Sesión” y se revisa en la base de datos si el usuario existe o no, para saber si se le da permiso y que se le abra una pestaña e inicie sesión con Google, le aparezca la ventana de Google}, si no existe, aparecerá un mensaje de que no tiene permisos para entrar a la aplicación.

Luego, el ActivitySolicitarUsuario, que se ingresa mediante el botón “Aquí” que pide los datos necesarios para que una persona se pueda registrar en el sistema y son recibidos por los administradores para que su cuenta sea aprobada.

El ActivityMenu que mostrará el menú de la pantalla de bienvenida en el que se tiene el botón de perfil del usuario, el de dispositivos disponibles y el de seleccionar el parámetro a analizar; esta interfaz le permite al usuario indicarle que acción en específico requiere realizar ya que, deberá elegir un dispositivo y parámetro para el respectivo análisis o si solamente quiere cerrar sesión.

El ActivityListaDispositivos mostrará al haber seleccionado el botón “Dispositivos disponibles” del Activity anterior los dispositivos disponibles a monitorear en el área junto con su respectivo parámetro seleccionado

Finalmente, el ActivityDispositivosSeleccionados permite visualizar el nombre del dispositivo elegido, el parámetro seleccionado y los datos que se encuentran en la base de datos, no solo como texto, sino también mostrará un gráfico lineal de los parámetros para que el usuario pueda ver como los valores cambian a lo largo del tiempo.

Descripción de los tipos de diagramas

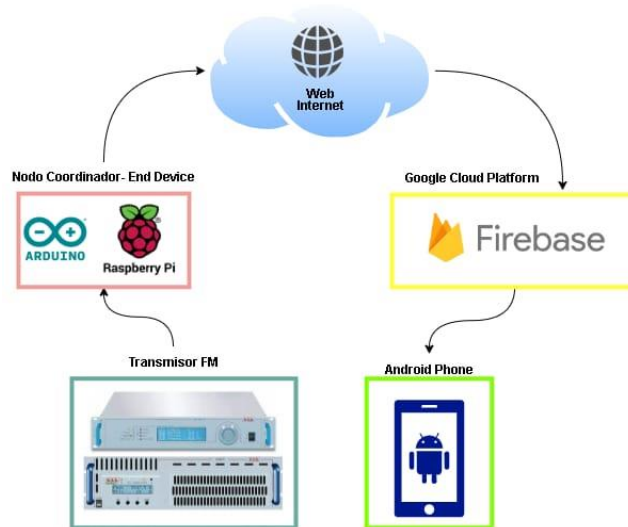


Figure 3: Diagrama de proyecto

Primero, se tiene el Transmisor FM que se encuentra compuesto por un amplificador y un excitador, luego a través del Arduino y Raspberry Pi se captura la información que el transmisor emite para luego mandar esos datos a través Internet al servicio de datos Firebase para finalmente mediante este servicio de Google mostrar los resultados usando la aplicación móvil desarrollada en un dispositivo Android.

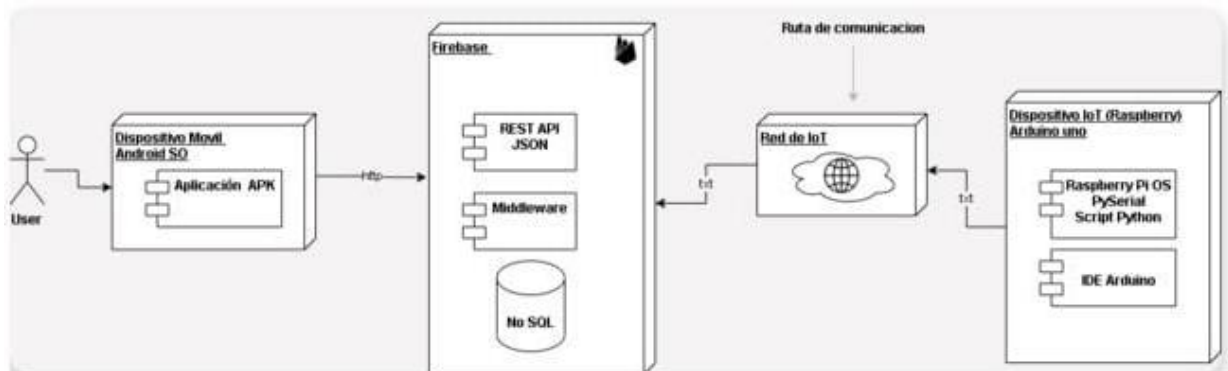


Figure 4: Diagrama de despliegue

Se tiene al usuario que ingresa a través de su dispositivo Android a la aplicación a través del archivo APK y hace peticiones HTTP al servidor de Firebase en la nube. Luego se tienen las librerías en la placa de desarrollo, siendo estas Arduino y Raspberry Pi para entablar una conexión serial entre ellas facilitando el flujo de datos. La Raspberry Pi contiene Script hecho en Python y códigos realizados en la IDE del Arduino para que esos datos que se sacan del transmisor se puedan enviar a través de la Red de IoT a los servidores de Firebase.

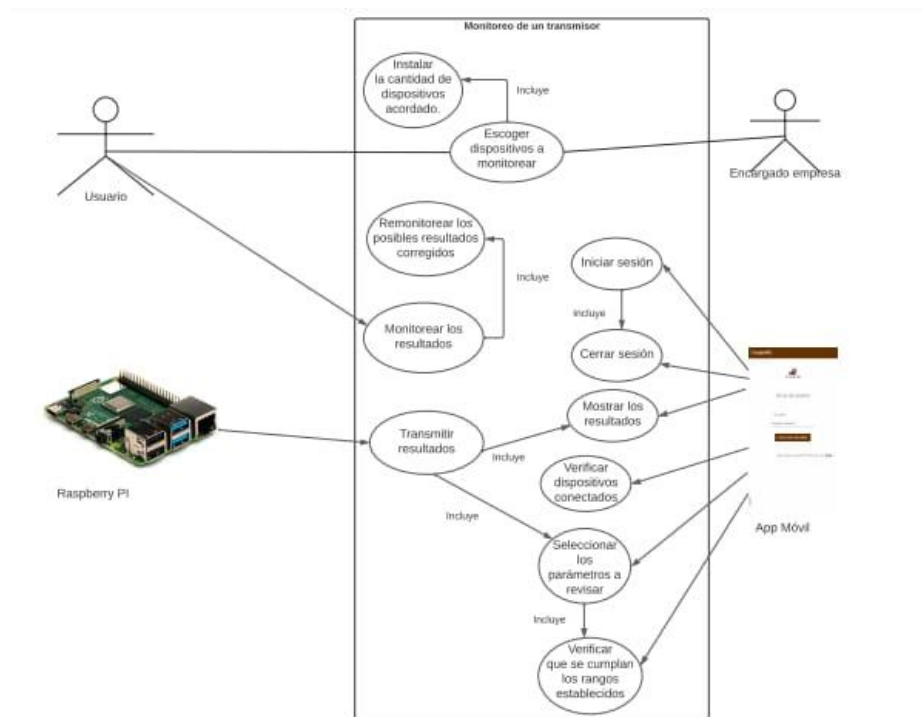


Figure 5: Diagrama de caso de uso

Para este diagrama, se tienen los principales actores que participarían en un escenario real que incluya el monitoreo de transmisor usando la aplicación móvil desarrollada. Primero se tiene al usuario quien será el encargado de empezar todo el procedimiento al escoger cual dispositivo desea monitorear, pero para esto se debe llegar a un acuerdo con el encargado de la empresa que se encuentra en posesión del producto desarrollado en este proyecto. Cuando haya un acuerdo mutuo, entonces se podrá proceder a instalar la cantidad de dispositivos acordados. Una vez completada esa sección, la Raspberry Pi será la encargada de transmitir los resultados de las medidas realizadas para luego mandar toda esa información a la base de datos y se pueda visualizar en la aplicación móvil en donde se especifican los parámetros que se deseen mostrar solamente y verificar si es necesario mandar una alerta al usuario que se encuentra monitoreando los resultados en caso de que algún parámetro se salga de un rango establecido.

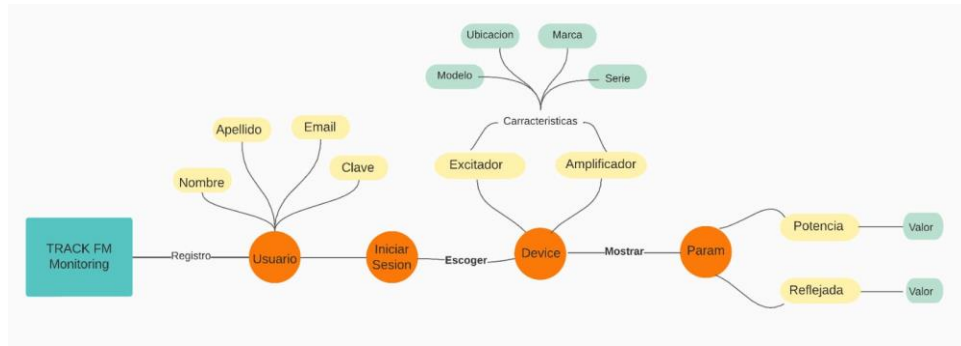


Figure 6: Diagrama de árbol

Se empieza con el aplicativo Track Fm puesto que, el usuario al momento de ingresar, deberá registrarse en el sistema. Luego, la cuenta del usuario tendrá las componentes de nombre, apellido, email y clave. Después podrá iniciar sesión en la aplicación para escoger el dispositivo que desee monitorear. Se pasa al dispositivo, el cual es un transmisor que normalmente se compone de un amplificador y un excitador en donde el usuario hará su elección. La aplicación permitirá que se puedan visualizar las características del componente seleccionado como lo es el modelo, ubicación, marca y serie. Finalmente, con el dispositivo elegido, se pasa a la visualización de los parámetros de este mismo como lo son la potencia y la potencia reflejada con sus respectivos valores.

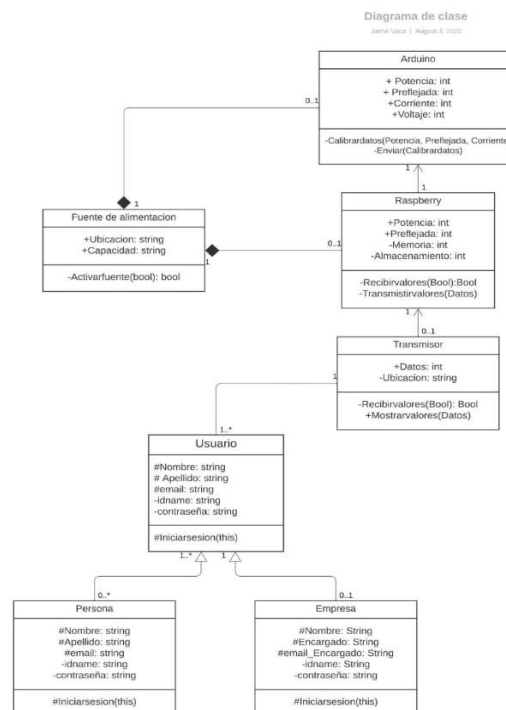


Figure 7: Diagrama de clases

En la Figure 5 se tienen las principales clases que participarán en el monitoreo de un transmisor con sus respectivas características y funciones. Un usuario puede ser tanto una persona como toda una empresa que mande a un encargado de ellos que deje la información esencial para el registro de sus nuevas cuentas al sistema. Luego de estar registrados, podrán visualizar la información deseada puesto que, el transmisor mandará todos esos datos, pero para que eso ocurra, el Arduino debe haberle mandado la información a la Raspberry Pi después de haber calibrado y procesado los parámetros medidos para que este último a través de un código Python mande la información a la nube de Google. Para que tanto Arduino Y Raspberry Pi funcionen, deben tener una fuente de alimentación activa, caso contrario, estas no tendrán funcionalidad alguna.

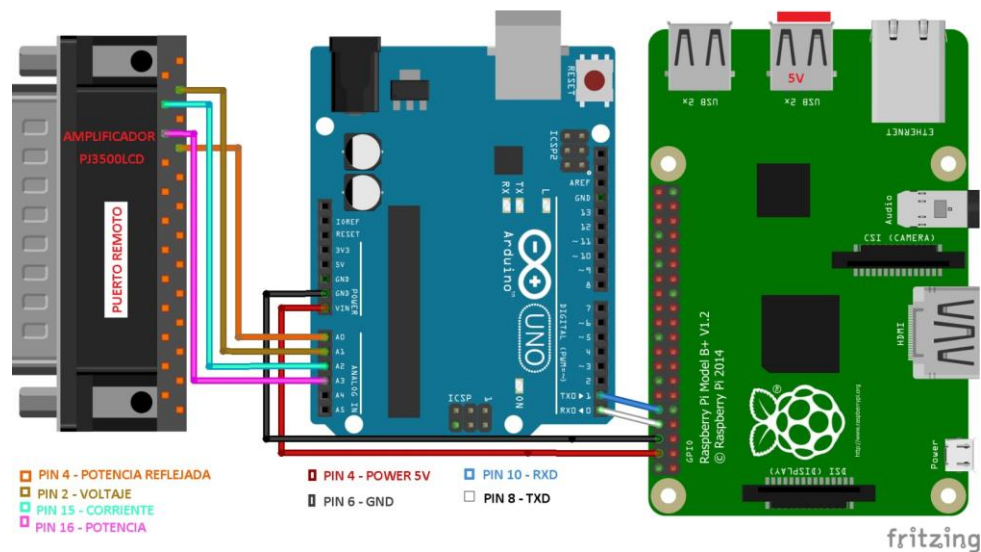


Figure 8: Diagrama de circuito

Se tiene el puerto remoto del amplificador RVR de 3500W donde se utiliza los pines detallados en la imagen para extraer los voltajes representativos de las variables, estos van a ser leídos y procesados por la tarjeta Arduino a través de un código que permita escalarlos a sus valores de potencia, potencia reflejada, voltaje y corriente según lo indica el manual para posteriormente ser enviados a la Raspberry Pi que transmite esta data estructurada a la nube de google para ser consultada por la APP.

- **METODOLOGÍA**

En el vídeo demostrativo, se explicó el origen de la idea de realizar el monitoreo de un transmisor FM, el cual tiene como objetivo reducir los inconvenientes que varias empresas radiodifusoras como lo son el gasto de dinero y tiempo al tener que trasladarse en ocasiones grandes distancias al tener que mandar a sus empleados a realizar las medidas especificadas en lugares muy lejanos e inhóspitos como lo son los cerros montañas. Por ello, se realizó una aplicación móvil que tiene como función mostrar los dispositivos disponibles dentro del área y los parámetros que se pueden medir en el sistema en el cual podrán visualizar y especificar los datos que desean analizar. Luego de elegir el dispositivo y parámetro a monitorear, existirá una tabla de datos de los valores que han sido ingresados a la base de datos para que el usuario asignado pueda ir comparando la variación en los valores a lo largo del tiempo y así saber si existe algún inconveniente.

Con respecto a la parte física del proyecto, se tiene una sección que representa a un transmisor FM el cual está compuesto por un excitador y un amplificador. En general, estos equipos tienen un puerto remoto en el cual se sacarán los parámetros a medir para luego ser calibrados a través de la tarjeta Arduino. Una vez que los datos hayan sido procesados, estos serán enviados a la Raspberry Pi, el cual a través de un código realizado en Python mandará información a la nube de Google, pero adicionalmente, también se han impreso los resultados en el terminal de la Raspberry.

- **RESULTADOS**

A continuación, se muestra la aplicación resultante del código fuente realizado en Android Studio explicado anteriormente:



Figure 9: Pantalla de inicio de sesión

Primero, se tiene la pantalla de inicio de sesión en donde el usuario podrá ingresar con sus datos en caso ya de tener una cuenta registrada o en caso contrario, podrá registrarse al señalar la palabra ‘Aquí’ en la parte inferior.

03:03 94%

TrackFM

Track Fm

Solicitud de usuario

Nombre Completo:

Ingrese nombre

Cédula/RUC:

Ingrese cédula/RUC

Celular:

Ingrese número

Correo electrónico:

Ingrese correo

SOLICITAR USUARIO

< □ ≡

Figure 10: Solicitud de usuario

En caso de que el usuario haya solicitado la creación de su cuenta, se le pedirá ingresar la información mostrada en la Figure 8, para que luego un administrador le revise y pueda aprobar su solicitud.

03:03 94%

TrackFM

Track Fm

Bienvenido

Perfil

DISPOSITIVOS CONECTADOS

PARÁMETROS

CERRAR SESIÓN

Grupo 4

< □ ≡

Figure 11: Pantalla de bienvenida

Estando en la pantalla de bienvenida, el usuario podrá ingresar a su perfil para poder asegurarse que los datos de su cuenta se encuentren correctamente registrados o también podrá cerrar sesión con la opción que se encuentra en la parte inferior.



Figure 12: Dispositivos disponibles

Para iniciar el procedimiento de análisis de un parámetro en específico, el usuario deberá antes ingresar a la opción de ‘dispositivos conectados’, para poder visualizar todos los dispositivos que puede analizar sus parámetros en el área.



Figure 13: Selección de parámetro

Luego de seleccionar el dispositivo a monitorear, se le pedirá al usuario que elija un parámetro que desee visualizar del dispositivo anteriormente mencionado.

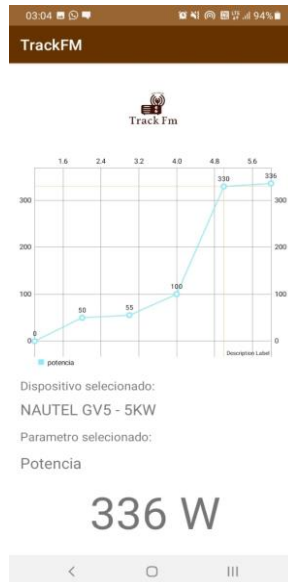


Figure 14: Datos del dispositivo y parámetro elegido

El usuario podrá observar los datos del dispositivo seleccionado, el parámetro elegido junto con su respectivo valor medido y una tabla con los valores que han sido ingresados a la base de datos a lo largo del tiempo para sus respectivos análisis.

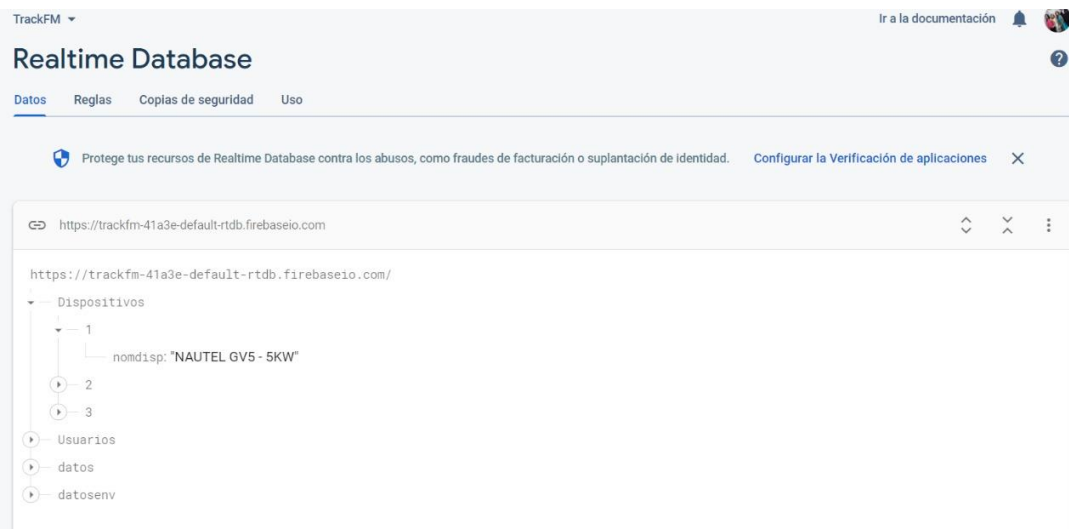


Figure 15: Grupo "Dispositivos" de la base de datos

Los datos que han sido enviadas a la nube por medio de la Raspberry son guardados en una base de datos la cual tiene diferentes grupos, tal y como se muestra en la Figure 13. El grupo “Dispositivos” almacena los dispositivos disponibles en el área que algún usuario quiera realizar su respectivo análisis

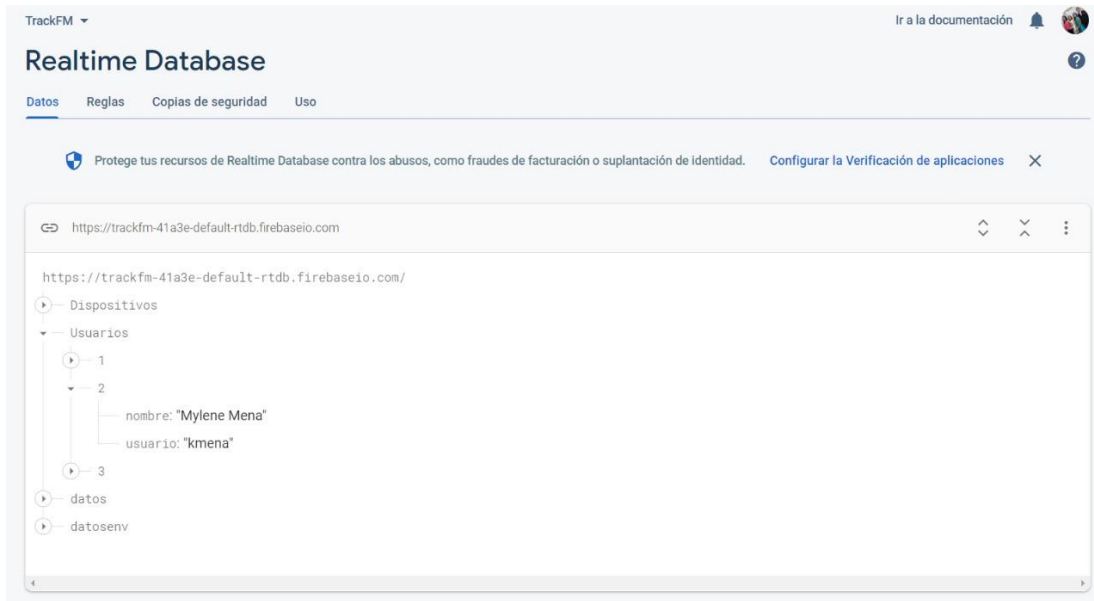


Figure 16: Grupo "Usuarios" de la base de datos

El grupo “Usuarios” almacena a los usuarios que tienen acceso a la aplicación móvil luego de haberse aceptado su solicitud de creación de cuenta.

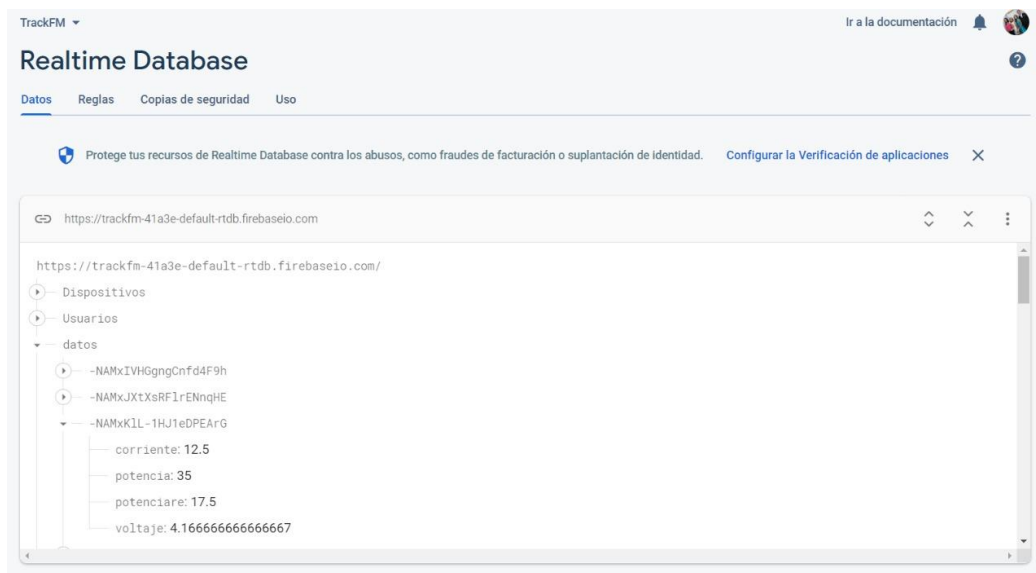


Figure 17: Grupo "datos" de la base de datos

Finalmente, el grupo “datos” almacena todos los parámetros que son enviados en tiempo real por la Raspberry Pi.

Analizador de código LINT:

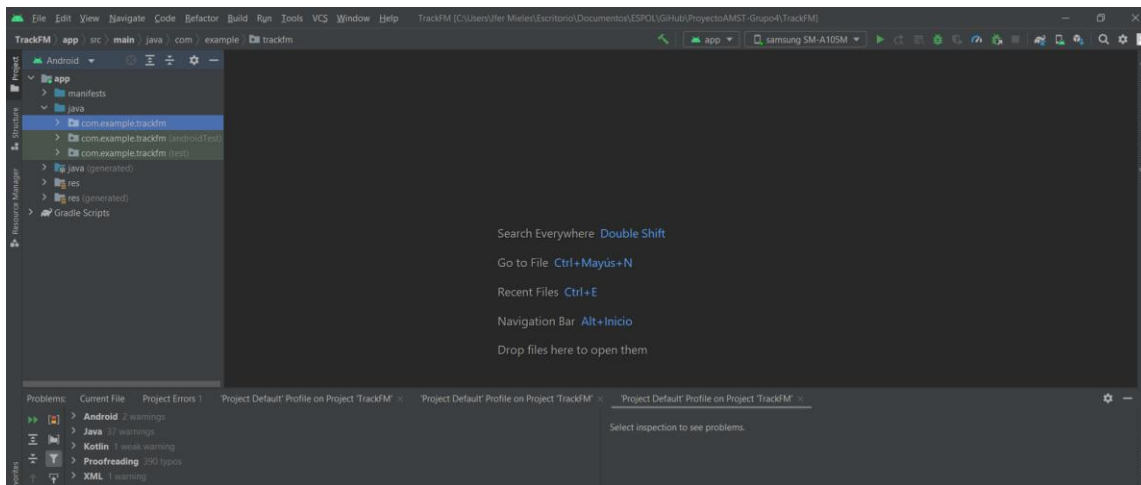


Figure 18: Analizador de código LINT

Aquí se pueden visualizar el número de errores de Android y Java en la parte inferior de la Figure 7, siendo 2 y 37 respectivamente.

Conteo de líneas de código usando cloc:

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1889]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Ifer Miele>cd C:\Users\Ifer Miele\Escritorio\Documentos\ESPOL\GitHub\ProyectoAMST-Grupo4

C:\Users\Ifer Miele\Escritorio\Documentos\ESPOL\GitHub\ProyectoAMST-Grupo4>cloc TrackFM
 499 text files.
 455 unique files.
 458 files ignored.

github.com/AlDanial/cloc v 1.94 T=79.03 s (5.8 files/s, 972.3 lines/s)
-----
Language             files      blank      comment      code
-----
XML                   262        4219        391        32420
JSON                  160          0          0        24191
Text                   9          0          0        14395
Java                  10         145         31         526
Bourne Shell           1          23         36         126
Properties             7          0         31          92
Gradle                 3          26          1          77
DOS Batch             1          21          2          66
Markdown              1          0          0           1
ProGuard              1           3         18           0
-----
SUM:                  455        4437        510       71894
-----

C:\Users\Ifer Miele\Escritorio\Documentos\ESPOL\GitHub\ProyectoAMST-Grupo4>
```

Figure 19: Conteo de líneas de código usando el comando cloc

Procesamiento de Resultados:

$$\%Calificación = \frac{EA + EJ}{T} * 100$$

En donde:

T = Total de líneas de código Java

EA = Número de errores de Android

EJ = Número de errores de Java

$$\%Calificación = \frac{2 + 37}{526} * 100 = 7.414\%$$

Utilizando el analizador de código LINT, se buscó que el porcentaje de calificación sea lo más cercano a 0 posible puesto que, eso significaría que la aplicación realizada está programada de la mejor forma. Para esta ocasión, el porcentaje quedó en 7.414%, siendo este un valor cercano a 0, por lo que se podría decir que si bien es mejorable el código realizado en Android Studio también este cumple al no tener muchos errores que afecten el rendimiento de la aplicación.

• CONCLUSIONES

- El hecho de que el proyecto pueda funcionar sin problemas depende de la red en la que se encuentre conectado debido a que, la red puede tener restricción de puertos lo cual causa que el código en Python de la Raspberry lance un error al no poder acceder a la base de datos de la nube de Google.
- Se decidió almacenar los datos en una nube de Google puesto que, en varias ocasiones, las empresas radiodifusoras requieren analizar cómo se comporta un cierto parámetro a lo largo del tiempo para poder visualizar si existe algún inconveniente que requiera su atención.
- Tener un registro de los usuarios que tienen acceso a la aplicación, permite disponer de un mayor control en caso de que surja algún tipo de análisis erróneo en la aplicación para poder determinar en el menor tiempo posible si fue una falla de la app o un error de análisis de alguno de los usuarios registrados en el sistema.

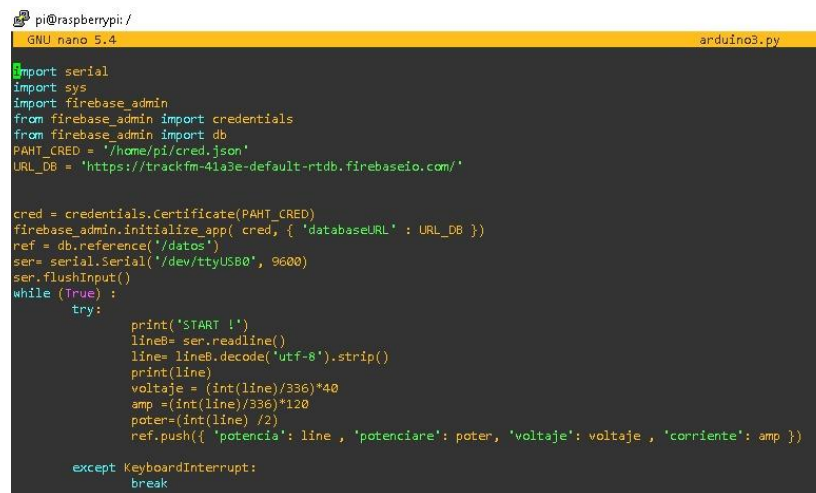
- **PRESUPUESTO**

| Dispositivo | Precio |
|------------------|----------|
| Arduino UNO | \$15.00 |
| Raspberry Jumper | \$343.00 |
| Blower | \$30.00 |
| Total a pagar | \$388.00 |

Tabla 1: Presupuesto del proyecto

- **APÉNDICE**

Fuente de código Hardware:



```

pi@raspberrypi: /
GNU nano 5.4 arduino3.py
import serial
import sys
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
PAHT_CRED = '/home/pi/cred.json'
URL_DB = 'https://trackfm-41a3e-default-rtdb.firebaseio.com/'

cred = credentials.Certificate(PAHT_CRED)
firebase_admin.initialize_app( cred, { 'databaseURL' : URL_DB })
ref = db.reference('/datos')
ser= serial.Serial('/dev/ttyUSB0', 9600)
ser.flushInput()
while (True):
    try:
        print('START !')
        lineB= ser.readline()
        line= lineB.decode('utf-8').strip()
        print(line)
        voltaje = (int(line)/336)*40
        amp =(int(line)/336)*120
        poten=(int(line) /2)
        ref.push({ 'potencia': line , 'potenciare': poter, 'voltaje': voltaje , 'corriente': amp })
    except KeyboardInterrupt:
        break

```

Figure 20: Código de Raspberry Pi en Python

El código realizado en la Raspberry Pi tiene como objetivo mandar los resultados medidos que viene desde el Arduino a la nube de Google. Adicionalmente, se ha programado de tal modo que se podrán mostrar los valores en el terminal de esta misma.


```

ProyectoAM

int Por = A0;
int Vo = A1;
int Co = A2;
int Po = A3; //
int pr = 0;
int vo = 0;
int co = 0;
int po = 0; //

void setup() {
  pinMode(Por, INPUT);
  while (!Serial) {
    ;
  }

  Serial.begin(9600);
}

void loop() {

  pr = ((analogRead(Por)* 0.0049)/1.81)*350;
  vo = ((analogRead(Vo)* 0.0049)/1.81)*40;
  co = ((analogRead(Co)* 0.0049)/1.81)*120;
  po = ((analogRead(Po)* 0.0049)/1.81)*3500;
  if (Serial.available() > 0) {
    Serial.write(pr);
    delay(5000);
    Serial.println(co);
    delay(5000);
    Serial.println(vo);
    delay(5000);
    Serial.println(po);
    delay(5000);
  }
}

```

Figure 21: Código realizado en el Arduino

Para el código programado en el Arduino, este tiene como función leer los datos enviados que vienen al sacarlos del puerto remoto del transmisor FM para después de terminar la lectura, estos puedan ser calibrados y procesados a la Raspberry Pi.

Fuente de código Software:

En el Activity “Dispositivoseleccionado” se muestra la información del parámetro seleccionado con el dispositivo escogido. Para identificar cuál ha sido el parámetro que el usuario desea saber se utiliza la condición if e if else y se usan los métodos leerparametros() y grafico() con los ids respectivos de cada parámetro.

```

//Análisis de parametro seleccionado para mostrar valores asociados de la Base de Datos

    if (pa.equals("Potencia Reflejada")){

        leerparametros(valor,id1,ulu2);
        grafico(id1);

    }
    else if (pa.equals("Potencia")) {

        leerparametros(valor,id2,ulu2);
        grafico(id2);

    }
    else if (pa.equals("Corriente")) {

        leerparametros(valor,id3,u3);
        grafico(id3);

    }
    else if (pa.equals("Voltaje")) {

        leerparametros(valor,id4,u4);
        grafico(id4);

    }

```

Figure 22: Activity "Dispositivoseleccionado"

El método leerparametros() extrae los datos de la RealDataBase, indexando el id del parámetro y setea el valor obtenido en un TextView.

```

private void leerparametros(TextView valor, String id, String unidad){
    db_reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                String val = String.valueOf(snapshot.child(id).getValue());
                valor.setText(val+unidad);
            }
        }
        @Override
        public void onCancelled(DatabaseError error) {
            System.out.println(error.toException());
        }
    });
}

```

Figure 23: Método leerparametros()

El método grafico() recoge todos los datos que se encuentran en la base de datos, y los almacena en un ArrayList<> cuyos datos se agregan a un LineChart. El objetivo de esta gráfica es que el usuario pueda ver cómo han variado en el tiempo los valores.

```
private void grafico(String id){

    db_reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {

            ArrayList<Entry> linelist = new ArrayList();
            int cont = 1;

            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                String val = String.valueOf(snapshot.child(id).getValue());
                float ejex = (float) cont;
                float ejey = (float) Integer.parseInt(val) ;
                linelist.add(new Entry(ejex*1,ejey*1));
                cont++;
            }

            LineDataSet lineDataSet = new LineDataSet(linelist, id);
            LineData lineData = new LineData(lineDataSet);
            LineChart lchart = findViewById(R.id.line_chart);
            lchart.setData(lineData);

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

Figure 24: Método grafico()