UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

SEDE VIÑA DEL MAR – JOSÉ MIGUEL CARRERA

MONITOREO REMOTO DE VARIABLES FÍSICAS UTILIZANDO RASPBERRY PI Y XBEE

Trabajo de Titulación para optar al Título de Ingeniero de Ejecución en CONTROL E INSTRUMENTACIÓN INDUSTRIAL

Alumno:

Carlos Pizarro Fajardo

Profesor Guía:

Mag. Guelis Montenegro Zamora

2024

RESUMEN

KEYWORDS: Telemetría, Base de datos, Raspberry, Xbee, Código abierto.

En el siguiente trabajo se presenta la creación e implementación de un sistema de telemetría de variables físicas basado en módulos Xbee y Raspberry Pi. Para esto, se captura el valor de un tipo de variable a través de un sensor, esto pasa a un módulo Xbee configurado como dispositivo final. Luego las 3 variables pasan por un Xbee configurado como coordinador que a través de comunicación serial enviará los datos a una base de datos configurada en una Raspberry Pi. Además, para que un usuario visualice los datos en tiempo real, estos serán mostrados en un dashboard que permita ver tendencias en línea con un historial de tiempo para mejorar la toma de decisiones. Finalmente, este trabajo no responde a una problemática específica, por lo que al final de este informe se presentan ejemplos de aplicaciones en las cuales esta tecnología podría ser implementada.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

SIGLAS Y SIMBOLOGÍA

A. SIGLAS

CPU	:	Central Processing Unit
ED	:	End Device
HDMI	:	High-Definition Multimedia Interface
HVAC	:	Heating Ventilation Air Conditioning
IEEE	:	Institute of Electrical and Electronical Engineers
IT	:	Information Technology
LDR	:	Light-Dependent Resistor
MLP	:	Minera Los Pelambres
PVM	:	Python Virtual Machine
RAM	:	Random Access Memory
SAG	:	Semi Autogenous Grinding
SD	:	Secure Digital
SBC	:	Single Board Computer
USB	:	Universal Serial Bus

B. SIMBOLOGÍA

%	:	Porcentaje
GHz	:	Gigahertz
Km	:	Kilometro
Kbps	:	Kilobits por segundo
Mbar	:	Milibar
MHz	:	Megahertz
Ms	:	Milisegundo
V	:	Volt

INTRODUCCIÓN

En el avance de las tecnologías en esta década, han surgido dos requerimientos para cada proyecto innovador: Telemetría y almacenamiento de información en sitios en línea. La gran cantidad de datos que se maneja, el carácter privado de estos, junto a la necesidad de crear una conectividad de monitoreo en línea para operadores y usuarios, ha llevado a que cada vez se creen sistemas integrados con protocolos de una seguridad mayor, con posibilidad de expandir la estructura de su red y con un coste económico de implementación menor. Estos desarrollos rápidamente se han unido a servidores y bases de datos que hoy se conocen como nubes, ya que pueden consultarse en tiempo real desde cualquier parte mientras se tenga un acceso a red. La unión de todos estos conceptos en un solo sistema ha revolucionado la industria para ejecutivos, operadores y usuarios ya que hubo un salto desde sistemas con infraestructuras complejas y costosas a la digitalización por completo de una organización.

CAPÍTULO 1: ANTECEDENTES GENERALES

1. ANTECEDENTES GENERALES

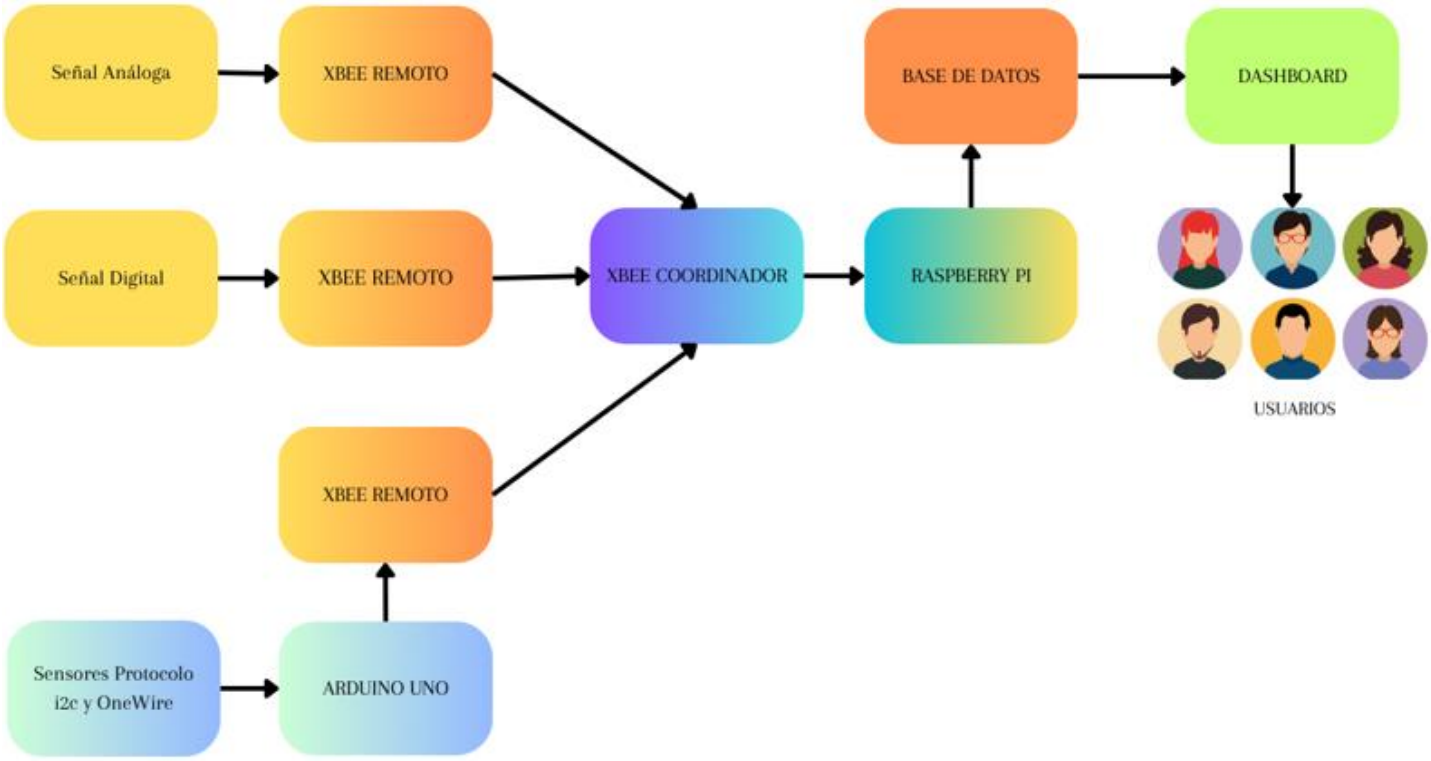
Este capítulo presenta antecedentes para el desarrollo de este proyecto tales como la problemática, componentes electrónicos a utilizar, configuraciones del sistema y además se definen los objetivos a cumplir.

1.1. PROBLEMÁTICA

El presente trabajo fue solicitado por el profesor Guelis Montenegro Zamora para la asignatura de Trabajo de Título de la carrera de Ing. de ejecución en Control e Instrumentación en la Universidad Técnica Federico Santa María. Por lo tanto, no resuelve una problemática como tal inicialmente. Sin embargo, la aplicación de este sistema de telemetría podría incluirse en varios procesos que necesiten obtener mediciones a distancia, para luego enviar estos datos a un servidor online tipo nube que permita visualizarlos en un dashboard. Ejemplos de estos pueden ser la agricultura, minería, celulosas, puertos, aplicaciones domóticas, monitoreo en visión por computador, etc.

1.2. TRABAJO ENCARGADO

El sistema de telemetría a realizar consiste en la combinación del módulo Xbee con Raspberry Pi. El proyecto medirá 3 variables físicas a través de sensores: Un sensor análogo, uno digital y uno en protocolo One Wire. Cada uno de estos estará conectado a un módulo Xbee configurado como End Device (*ED*) y estos 3 módulos transmitirán las tramas de datos a un módulo Coordinador. Finalmente, el coordinador a través de comunicación serie enviará los datos a la Raspberry Pi, la cual funcionará como servidor. Este trabajo se puede ver en la figura 1-1.



Fuente: Elaboración propia

Figura 1-1: Diagrama de bloques para proyecto

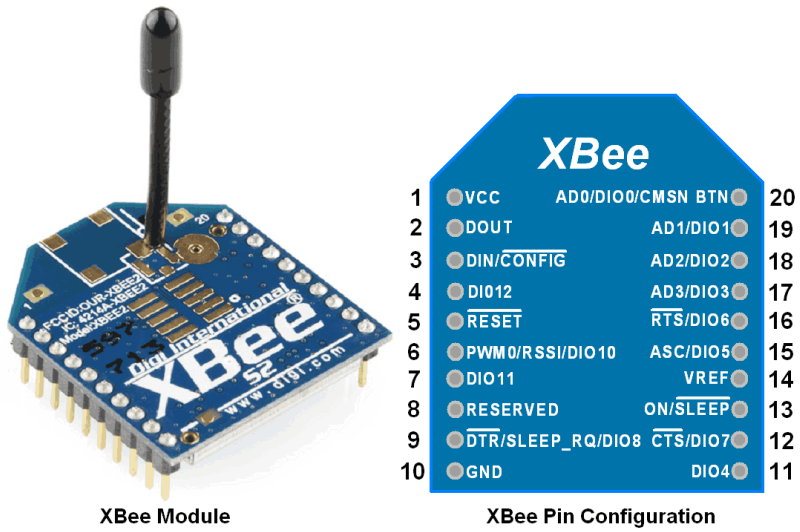
1.3. REQUERIMIENTOS DE HARDWARE Y SOFTWARE

Previo a la implementación física del proyecto es necesario describir y definir los componentes electrónicos, sus configuraciones posibles y los conceptos relacionados al estado del arte para este trabajo.

1.3.1. XBEE

El módulo Xbee, que se puede observar en la figura 1-2, es un dispositivo integrado que brinda un medio inalámbrico para la interconexión y comunicación. El módulo utiliza un protocolo de red IEEE 802.15.4 para crear redes punto a multipunto o

redes punto a punto. Cuenta con las capacidades para manejar un gran volumen de datos con una baja latencia, ya que basa su funcionamiento en el estándar de redes MESH llamado Zigbee. Dentro de los tipos de módulos Xbee existen diferentes series, en el caso particular de este trabajo, el Departamento de Electrotecnia e Informática de la Universidad dispone de la serie 2, la cual puede ser configurada por el usuario como Coordinador, Router o dispositivo final (end device).



Fuente: www.electronicwings.com

Figura 1-1: Módulo XBEE Serie 2

1.3.2. Protocolo Zigbee

El protocolo ZigBee es un estándar que define una serie de protocolos para el armado de redes inalámbricas de corta distancia y baja velocidad de datos, operando en bandas de 868 *MHz*, 915 *MHz* y 2.4 *GHz*. Puede transferir datos de hasta 250 *Kbps*.

Este estándar permite un bajo consumo para uso de equipos a batería, bajo costo de dispositivos y mantenimiento y está optimizado para ciclos de transmisión efectivos menores a 0.1%, entre otras especificaciones.

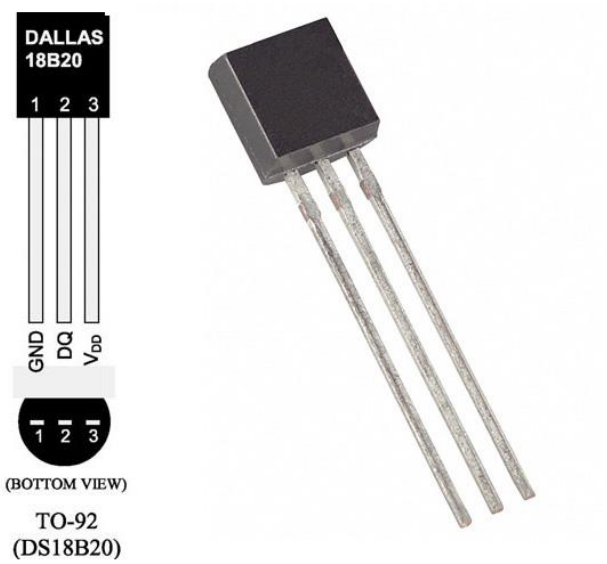
Sus aplicaciones más comunes son para automatización de hogares e industrias, entre otros.

1.3.3. Sensores

Este trabajo pretende visualizar 4 variables físicas a través de 4 sensores diferentes. Uno con protocolo one-wire, uno I2C, uno digital y uno del tipo análogo. Los cuales se describen a continuación:

1.3.3.1. Sensor de temperatura DALLAS 18B20

Para el protocolo One-Wire se ha seleccionado el sensor DS18B20, el cual puede ser observado en la figura 1-3. Es un sensor de temperatura digital que utiliza este protocolo para enviar datos a través de un solo pin. Debe polarizarse con voltajes de 3 a 5 volt y el pin de datos estará conectado a uno de los XBEE.

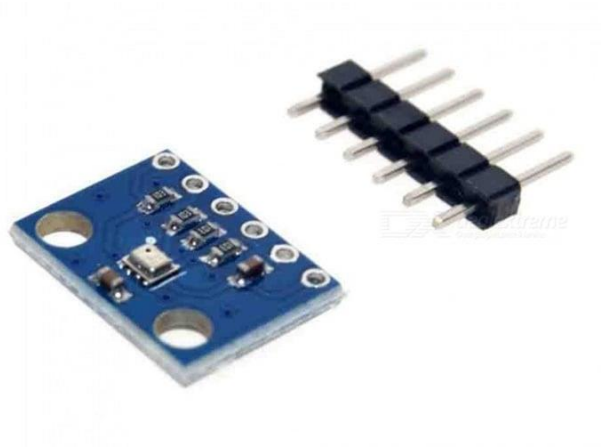


Fuente: https://electronicaemedin.cl/producto/sensor-de-temperatura-ds18b20/

Figura 1-2: Sensor DS18B20

1.3.3.2. Sensor de presión BMP280

El Sensor BMP280, que puede visualizarse en la figura 1-4, es un sensor de presión y temperatura. Está diseñado para aplicaciones que requieren una medición precisa de presión atmosférica y temperatura en simultáneo. Este sensor utiliza una interfaz digital para comunicarse con otros dispositivos a través de protocolo I2C. Por lo tanto, es un sensor fácil de integrar a microcontroladores, ya que incluye una calibración interna que simplifica el proceso de integración.

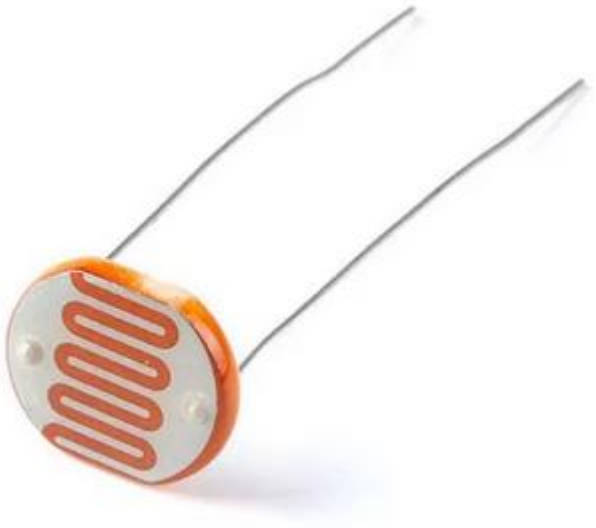


Fuente: https://afel.cl/producto/sensor-barometrico-bmp280/

Figura 1-3: Sensor BMP280

1.3.3.3. LDR

Para ingresar una variable análoga desde un sensor se ha escogido usar una LDR, cuya forma se puede ver en figura 1-5. Este es un resistor dependiente de luz que junto a un circuito acondicionador permite medir esta variable física. Este variará su resistencia según la intensidad lumínica. Además, es un componente pasivo por lo que su aplicación al circuito es sencilla.



Fuente: https://http2.mlstatic.com/D_NQ_NP_2X_659997-MLC47737364958_102021-Fwebp

Figura 1-4: LDR para circuito

1.3.4. Python

Python es un lenguaje de programación utilizado en aplicaciones web, desarrollo de software, bases de datos, aprendizaje de máquinas, etc. Permite su ejecución en diferentes plataformas y se integra a softwares por lo que aumenta la velocidad del desarrollo. Esto gracias a su fácil uso y comprensión, utilizando un lenguaje de datos similar al idioma inglés.

Soporta el uso de módulos, librerías y paquetes de código que son intercambiables, en lugar de una larga lista de instrucciones como en otros lenguajes de programación. Esto permite la reutilización de código en varios proyectos, siendo escalable y fácil de importar y exportar.

La implementación estándar de Python se llama “cpython”, que no convierte su código en lenguaje de máquina, sino que es convertido en un “código de byte”. Este código no puede ser entendido por la *CPU*, por lo que se necesita un intérprete llamado “Python Virtual Machine” (*PVM*) para ser ejecutado.

1.3.5. Raspberry Pi

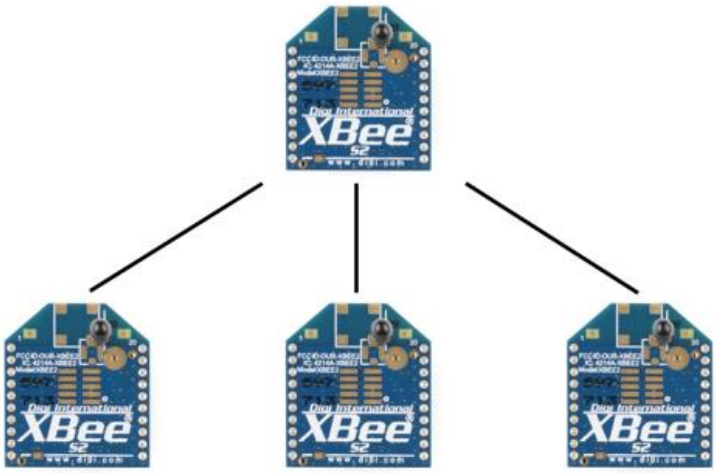
Raspberry Pi es un computador de placa única o *SBC* (Single board computer) compuesto por una *CPU*, memoria *RAM*, puertos de entrada/salida de audio y video, conexión de red, wifi, ranura *SD* para almacenamiento, conexión de periféricos vía *USB* y visualización a través de *HDMI*. Entre sus principales ventajas se encuentra su sencillez de uso ya que cuenta con una serie de funcionalidades que pueden ser manipuladas con Python. Esto junto a su precio reducido y tamaño pequeño la hace un equipo muy conveniente de escoger para aplicaciones tecnológicas e innovadoras. Su sistema operativo es Raspbian que es basado en Debian y está optimizado para el hardware de la Raspberry Pi. Este sistema ofrece herramientas de ofimática, navegación, gestión de correos, programación, etc. Su énfasis está en dar estabilidad en el desempeño de los paquetes que originalmente están en la distribución de Linux.

Fuente:https://es.wikipedia.org/wiki/Raspberry_Pi#/media/Archivo:Raspberry_Pi_4_Model_B_-_Side.jpg

Figura 1-5: Raspberry Pi 4 Modelo B

1.3.6. Estructura de Red Estrella

La configuración de comunicación para los módulos Xbee tendrá una topología del tipo estrella. Una estructura de red en estrella es aquella en la que los equipos terminales están conectados directamente a un dispositivo central y todas las comunicaciones se han de hacer necesariamente a través de este. Esto permite ciertas características y ventajas como: facilidad de administración, agilidad para modificaciones y ampliaciones, optimización de recursos de transmisión. Mientras que desventajas asociadas son: en caso de descomponerse el nodo central la red queda fuera de servicio y el módulo central puede convertirse en un cuello de botella. Un ejemplo de esta topología se puede observar en la figura 1-7.



Fuente: Elaboración propia

Figura 1-6: Topología Estrella XBEE

1.3.7. Base de datos MariaDB

MariaDB es uno de los más populares sistemas de gestión de bases de datos relacionales en el mundo. Es un sistema gratuito de código abierto de los mismos creadores de MySQL. Cuenta con las mismas características que este, pero tiene mejoras

respecto al rendimiento y opciones de seguridad, por lo que puede ser usada para tareas de procesamiento pequeñas como de nivel empresarial o industrial.

Para este trabajo en particular MariaDB brindará una ventaja para la creación de la base de datos, ya que se puede manipular con Python que es el lenguaje principal con el que se implementará este sistema.

1.3.8. Grafana para dashboard

Grafana es una plataforma de código abierto para visualización de datos, monitoreo de infraestructuras *IT* y aplicaciones. En esta los usuarios pueden ver datos a través de tablas y gráficos que se unifican en un panel de control, facilitando su comprensión y análisis. Permite realizar consultas e identificar información con mayor facilidad, posibilitando la eficiencia de los procesos.

Al automatizar la monitorización de grandes cantidades de datos temporales, con la opción de generar alertas ante posibles problemas y desviaciones, permite que una organización o industria tome decisiones y aplique soluciones con mayor facilidad, logrando un mejor equilibrio entre rendimiento y costo. Un ejemplo de panel de datos se puede ver en la figura 1-8.

Fuente: www.grafana.com

Figura 1-7: Dashboard de muestra Grafana

1.4. OBJETIVOS DEL PROYECTO

Para el correcto desarrollo del proyecto, se definirá un objetivo general y objetivos específicos.

1.4.1. Objetivo General

Implementar una red basada en Raspberry y XBEE para monitorear variables físicas en dashboard.

1.4.2. Objetivos Específicos

a. Estudiar cada uno de los dispositivos que conforman el proyecto.
b. Implementar topología estrella para la conexión de XBEE.
c. Realizar la comunicación entre Raspberry Pi y XBEE coordinador.
d. Visualizar las variables físicas en un dashboard.
e. Presentar resultados prácticos del proyecto.
f. Presentar aplicaciones prácticas (agricultura, telemetría, minería, etc.)

CAPÍTULO 2: IMPLEMENTACIÓN DEL PROYECTO

2. IMPLEMENTACIÓN DE PROYECTO

Este capítulo presenta las configuraciones de software y hardware para que el proyecto funcione de acuerdo con los objetivos de este trabajo. Esto considera asignar modos de operación para cada dispositivo, las direcciones, rutinas de muestreo para XBee, configuración de Raspberry, generación de comunicación entre XBee y RPi y la implementación de un circuito que incluya los componentes para realizar mediciones y almacenar las muestras en una base de datos. Además, se incluyen descripciones teóricas relacionadas al envío y procesamiento de los datos que son relevantes en el desarrollo.

2.1. DIAGRAMA DE BLOQUES PARA PROYECTO

El diagrama de bloques para este proyecto se puede observar en la figura 1-1.

2.2. TIPOS DE CONFIGURACIÓN PARA MÓDULOS XBEE

Anteriormente, se mencionó que un dispositivo XBee puede tener diferentes configuraciones dependiendo el propósito por el que se incluya en un proyecto. A continuación, se describe brevemente cada una de estas opciones y cuales se usan para este trabajo.

2.2.1. XBee como Coordinador

Para crear una red de tipo estrella o punto a multipunto, uno de los dispositivos debe ser configurado como Coordinador. Este se encarga de recibir cada uno de los datos transmitidos por los otros 3 XBee y se comunica con la Raspberry Pi a través del puerto UART.

2.2.2. XBee como Router

Un módulo XBee configurado como Router cumple la función de extender el tamaño de la red. Para esto tiene la capacidad de enrutar paquetes de datos entre dispositivos. Esto significa que puede recibir un dato y enviarlo a otro destino, actuando como intermediario extendiendo la cobertura. Este parámetro puede configurarse directamente a través del XCTU de legado.

2.2.3. XBee como End Device

Un módulo configurado como End Device cumple la función de comunicarse directamente con sensores y/o actuadores. Opera en redes Zigbee, por lo que se podrá comunicar con un Router o con un Coordinador directamente.

2.2.4. Modo AT

En el modo AT o modo transparente todos los datos recibidos se encuentran de la misma manera que fueron enviados. La comunicación en modo transparente da los mismos resultados como si dos módulos estuvieran conectados a través de un cable, pero claramente la ventaja de transmisión por radio frecuencia hace de la conexión física innecesaria.

Las ventajas principales de este modo son:

- En modo AT, el XBee funciona como una línea serial: lo que se envía es exactamente lo que el otro módulo recibirá.
- Es compatible con cualquier dispositivo que pueda comunicarse en serie.
- Es conveniente para comunicaciones punto a punto.

Por otro lado, las desventajas por las cuales este modo no se utilizará son:

- Trabajando con más de 2 módulos, es necesario configurar la dirección de destino antes de enviar cada mensaje.
- No es posible identificar la fuente de cada mensaje recibido.

2.2.5. Modo API

API (Application Programming Interface), es una sigla para referirse al modo Interfaz para programar aplicaciones. El modo API requiere de una comunicación con el módulo de una manera estructurada en tramas o tal como se puede encontrar en la documentación: “API FRAMES”.

El modo API cuenta con comandos con posibilidad de respuestas por consulta (polling), envío de paquetes y verificación del estatus de comunicación. Este modo amplía posibilidades de uso por ejemplo en:

- Configurar el XBee.
- Configurar dispositivos remotos en una misma red.
- Transmisión de datos a múltiples destinos (red punto a multipunto).
- Estatus de cada paquete transmitido en radio frecuencia.
- Identifica la dirección origen de cada paquete recibido.
- Manejo y diagnóstico de red.

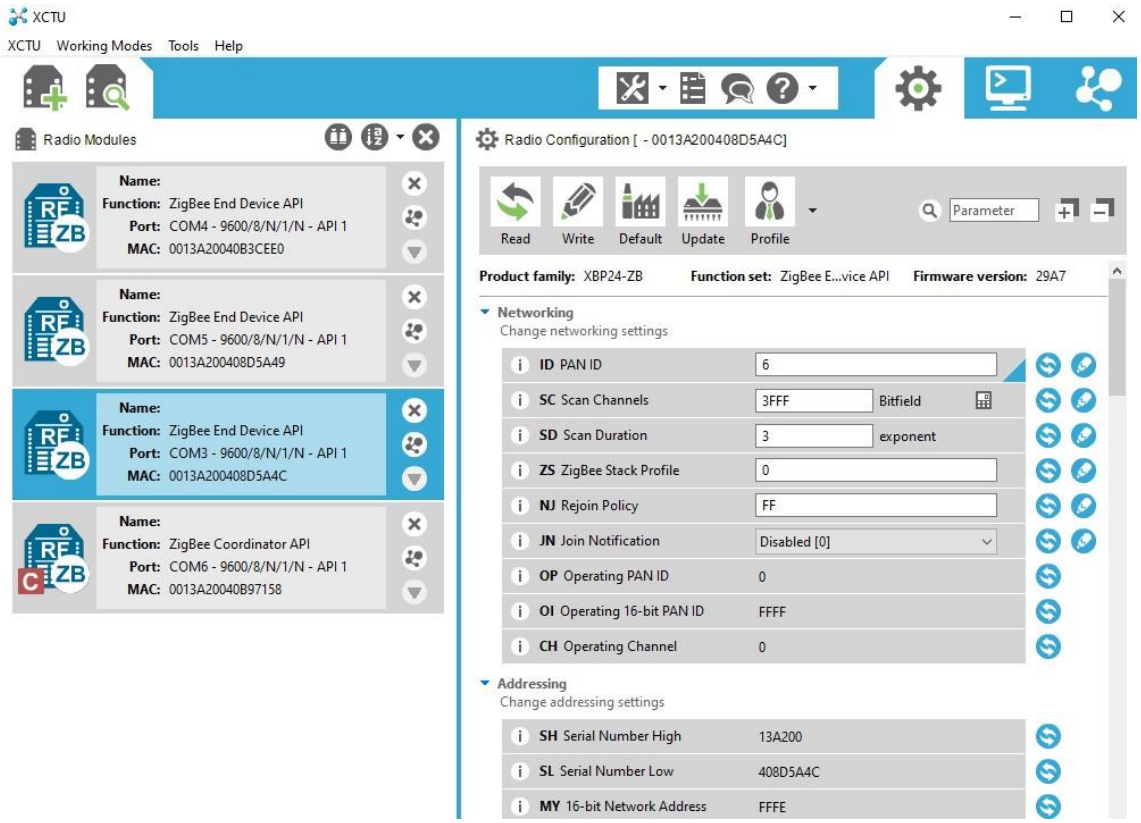
Dependiendo el valor del parámetro AP, el dispositivo puede operar en API 1 o API 2 (escaped).

Ambos modos operan de la misma manera y la única diferencia será que el modo 2 incluirá excepciones para bytes como “0x7E”, que es el primer byte que identifica el inicio de un nuevo paquete.

Para este trabajo, el Coordinador se configurará en modo API 1 ya que solo recibirá datos y los 3 End Device operarán en API 2.

2.3. SOFTWARE XCTU

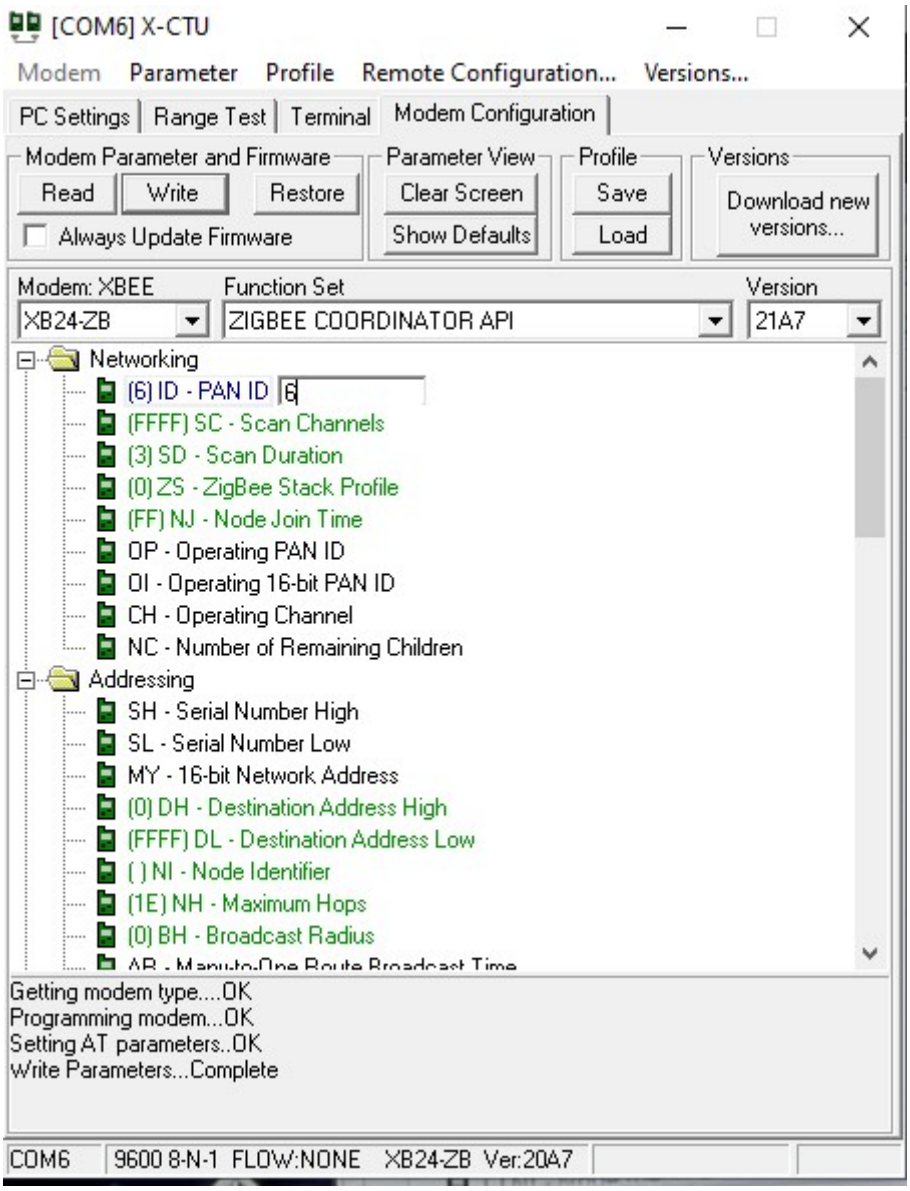
El Software XCTU, visible en figura 2-1, puede ser descargado directamente desde la página de Digi. Este viene en dos versiones, una actualizada y una de legado, donde ambas permiten hacer las mismas configuraciones, la única diferencia será la interfaz hacia el usuario.



Fuente: Captura de pantalla.

Figura 2-1 : Software XCTU 1

Al momento de comenzar a configurar los módulos para realizar pruebas de comunicación, puede ser un tanto abrumador la cantidad de parámetros que pueden configurarse en el software. Sin embargo, para comenzar a enviar información de un punto a otro bastará configurar en modo AT y asignarle un valor a los parámetros PAN ID, DH y DL, que se mencionan a continuación en este capítulo.



Fuente: Captura de pantalla.

Figura 2-2 : Software XCTU de Legado

El software de legado es útil en caso de tener algún módulo conflictivo que no permita restablecer el Firmware. Para esto, se puede usar la pestaña “restore” ubicada en Modem Configuration, como se ve en la Figura 2-2. Además, la ventaja que entrega la versión de legado es que se puede seleccionar directamente un protocolo Zigbee como Coordinador, Router o End Device y en modo AT, API1 o API2.

Para este trabajo los módulos fueron configurados con ambos softwares, ya que con la versión nueva no hubo respuesta, pero con la de legado sí.

2.3.1. Parámetro PAN ID

El parámetro PAN ID (Personal Area Network Identifier) establecerá una red asociada a un número hexadecimal que puede ir desde el 0x0000 al 0xFFFF, ofreciendo un amplio rango de redes posibles. Este parámetro es importante porque le da seguridad al sistema, ya que es un número único y permitirá interactuar desde el Coordinador con otros elementos en la red.

2.3.2. Parámetro DH

El parámetro Destination High Address es el valor de un tramo de la dirección de destino al momento de enviar un mensaje a otro XBee. Junto al parámetro DL completan la dirección final. El parámetro está expresado en un valor decimal de 4 bytes (32 bits), por lo que al unirse al DL formarán los 64 bits de la dirección única que tiene cada XBee. Tiene una importancia mayor a la hora de crear redes MESH porque permite saltarse unos destinos y dirigir mejor los datos.

2.3.3. Parámetro DL

El parámetro Destination Low Address, al igual que el parámetro DH, define un tramo de la dirección de destino. Completa los 4 bytes (32 bits) restantes para formar 64 bits.

2.3.4. Parámetro NI

El parámetro Node Identifier (NI) funciona como la asignación de un nombre para el dispositivo. Esto es conveniente para un usuario al momento de configurar una red con muchos dispositivos y en el caso particular de este trabajo, al utilizarse la librería digi-xbee, la cual se describe más adelante; este parámetro será muy relevante al momento de consultar de que dispositivo proviene un mensaje para este poder clasificarlo dentro de la base de datos.

2.3.5. Parámetro AP

El parámetro Api Enable permite definir por configuración si el Router o End Device funcionará en modo Api 1 o modo Api 2, los cuales fueron previamente descritos en este capítulo.

2.3.6. Parámetros para entradas y salidas

Los parámetros de Input y Output son configurables desde el software y también desde la librería digi-xbee. Para este trabajo se configuró en la programación por python cada una de las entradas y salidas, por lo que en el xctu no se necesita seleccionar ninguna de estas.

2.4. SOFTWARE PARA RASPBERRY PI

Para poder implementar la parte del procesamiento de datos del proyecto, se requiere contar con software con la capacidad de interpretar las tramas provenientes de los módulos, almacenar en una BDD las mediciones y que finalmente sean procesadas para visualizarse. Para esto, se han seleccionado entre una variedad de opciones en el código abierto, los siguientes softwares descritos a continuación. Además, la Raspberry ha sido configurada de una manera para usuario básica, activando el uso de VNCViewer para poder trabajar con esta desde el computador y no instalarle periféricos adicionales.

2.4.1. Instalación base de datos MariaDB

Para la instalación de la base de datos se utiliza el siguiente comando:
$ sudo apt install mariadb-server

Luego, es necesario desde la terminal ingresar a la base de datos, para esto se escribe:

$ sudo mysql -u root

El comando permite entrar como usuario a mariadb.

Se crea la base de datos con el nombre “xbees” con el comando “CREATE XBEES;”

2.4.2. Instalación Grafana

Para la instalación del software que interpreta la base de datos y los traduce a un dashboard, se siguen los pasos disponibles en la página, copiando los siguientes comandos directo a la terminal de la Rpi:

Se agrega la apt-key:

$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add –

Se añade el repositorio:

$ echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list

Se instala Grafana con:

$ sudo apt-get update

$ sudo apt-get install -y Grafana

Ya se encuentra instalado el software que interpretará la base de datos, pero aún no se encuentra habilitado el servidor web. Para habilitarlo, se siguen las siguientes instrucciones por terminal:

$ sudo /bin/systemctl enable grafana-server

$ sudo /bin/systemctl start grafana-server

Ahora Grafana está funcionando como servidor en la Rpi. Para acceder bastará con entrar desde un navegador a la dirección: `http://<ip raspberry>:3000` e ingresar las credenciales que para este trabajo serán usuario: admin y clave: raspberrypi.

Es importante considerar que el dispositivo que intente ingresar al dashboard debe estar en la misma red a la que se encuentre conectada la Rpi y que esta tiene que estar energizada.

2.4.3. Instalación de librería Digi-XBee

Convenientemente, Python cuenta con una librería relacionada a los módulos XBee, la cual puede ser instalada a través de terminal en la Raspberry Pi con el siguiente comando:

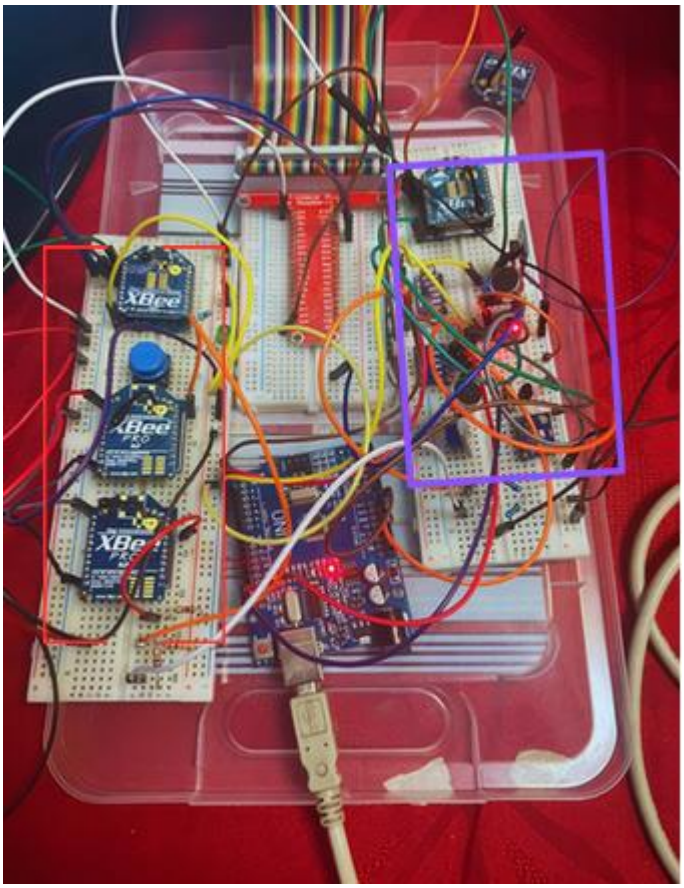
```
$ pip3 install digi-xbee
```

Esta librería solo funciona con python3, por lo que es necesario desinstalar cualquier versión anterior de Python que pudiera estar en la RPi. Además, junto con la instalación de la librería, se instalará automáticamente PySerial3 y el paquete SRP, los cuales son necesarios para generar una comunicación entre el XBee coordinador y la RPi a través del puerto UART.

La documentación de la librería con los comandos para interactuar con los XBee se encuentra disponible online de manera gratuita y se adjunta en las referencias de este trabajo.

Hacer uso de esta librería tiene una variedad de beneficios para el proyecto, ya que el software XCTU pasaría a un segundo plano permitiendo realizar la configuración de entradas/salidas, identificación de red, lectura de parámetros y comunicaciones directamente desde el código. Optimizando el tiempo para el estudiante tanto en programación como en lectura de documentación.

2.5. CIRCUITO ELECTRÓNICO EN FÍSICO



Fuente: foto propia.

Figura 2-3 : Circuito Electrónico

En la figura 2-3 se puede observar en el cuadro rojo el XBee coordinador, el de señal análoga, digital y en el morado se puede observar el Xbee para sensores de protocolo que pasan por el Arduino.

2.5.1. Sensor de sonido KY-037 para señal digital

Para recibir una señal digital en uno de los módulos se ha escogido el sensor KY-037, ya que tiene la capacidad de entregar una salida análoga y otra digital, por lo que se acondiciona a los 3,3 volts de entrada para el Xbee.

2.5.2. Puente de Wheatstone para LDR en señal análoga

Para utilizar la LDR es necesario colocar este sensor en un circuito acondicionador, para poder obtener una variación de voltaje que será la entrada del XBee para señal análoga. Este circuito es un puente de Wheatstone con 3 resistencias iguales y polarización de 3,3 volts para proteger el XBee de sobre tensiones. El circuito se explica con la figura 2-4 a continuación:

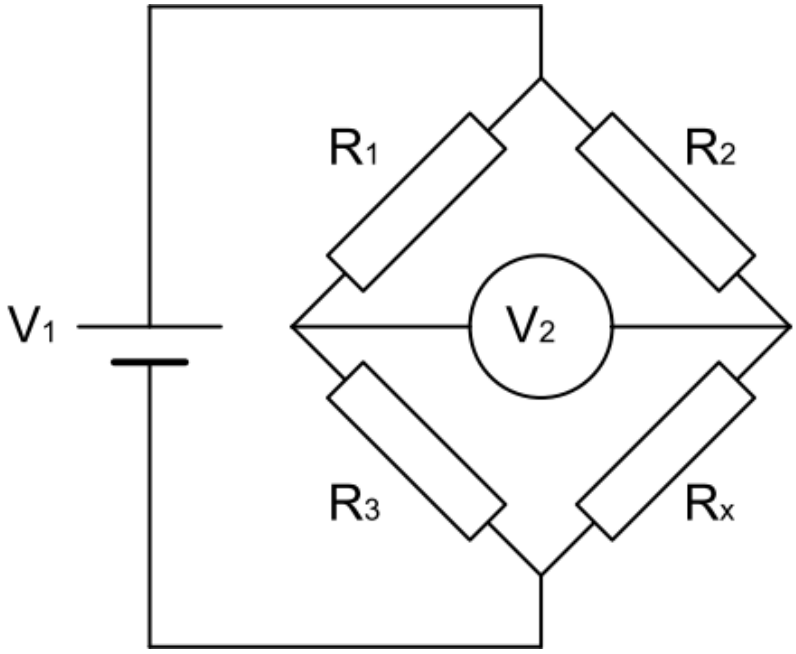


Figura 2-4 : Puente de Wheatstone

Fuente: https://josecasares.com/puente-de-wheatstone/

La LDR correspondería a Rx en la figura 2-4 y V2 variante que es la señal análoga del sistema.

2.5.3. Sensores para protocolo OneWire e I2C en Arduino

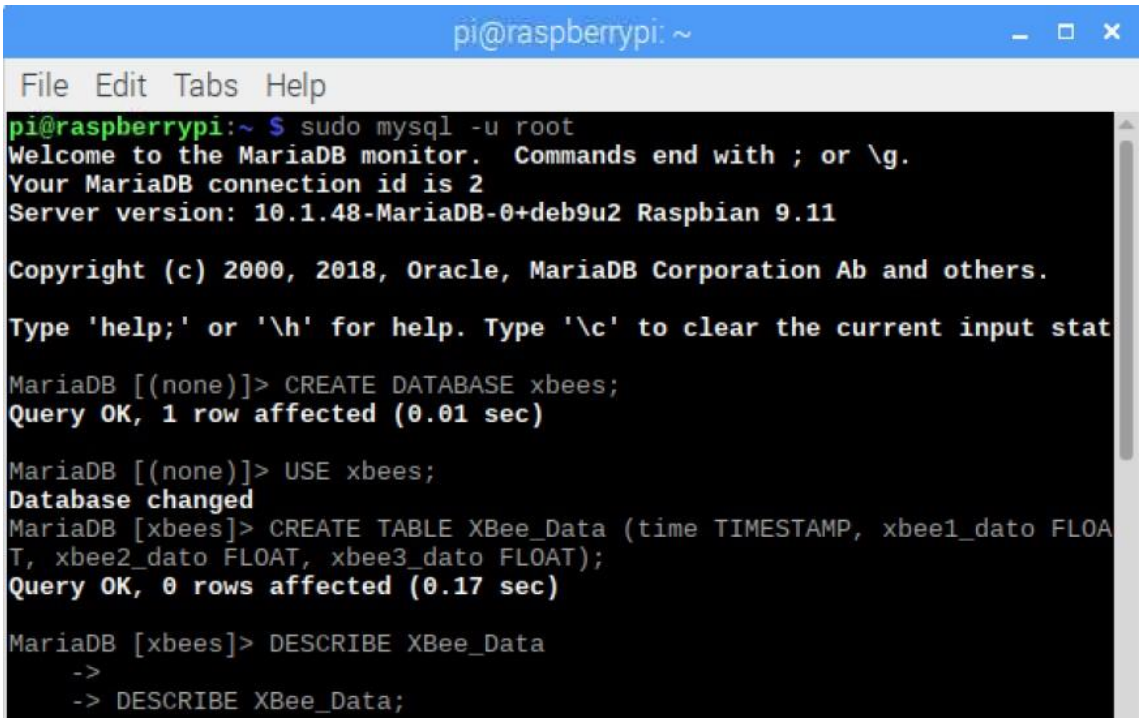
A modo de cumplir con los objetivos de este trabajo sobre realizar mediciones de variables físicas, se han incluido dos protocolos de comunicación para sensores. El

dispositivo XBee no tiene la capacidad de reconocer ninguno de estos, por lo tanto, para interpretar estos datos se ha incluido un Arduino UNO solamente con el propósito de actuar como interfaz para cada sensor. Como se mencionó anteriormente en el capítulo 1 de este trabajo, los sensores respectivamente para protocolo onewire e I2C escogidos son un Dallas 18B20 para temperatura y un BMP280 para presión barométrica. Ambos sensores cuentan con librerías de desarrollo amigables por lo que basta con polarizar cada sensor con 3,3 volts y ejecutar el código.

2.6. REGISTRO DE DATOS Y ALMACENAMIENTO EN MARIADB

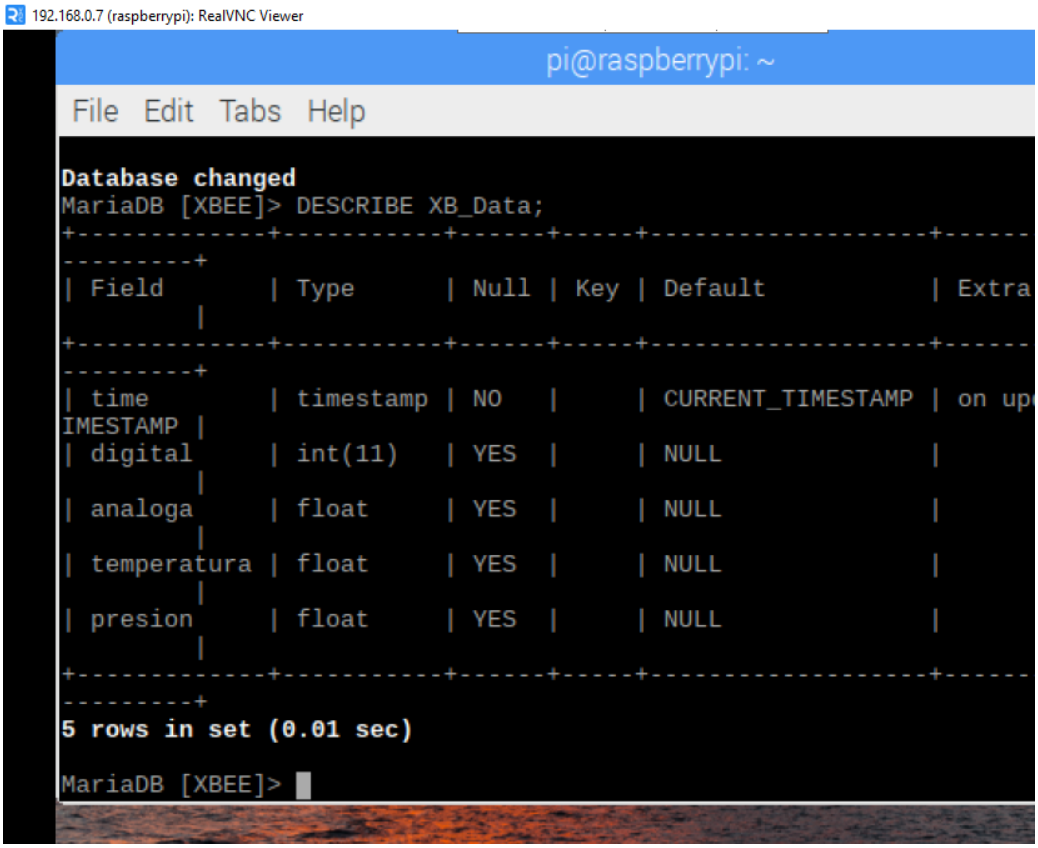
Para almacenar los datos es necesario crear la base de datos desde la terminal de la Raspberry en lenguaje de consulta SQL, luego crear la tabla comentando el tipo de variable a asignar para cada columna y finalmente, desde Python, ingresar el nombre de esta base para que se almacene la información en esta. Los comandos son los siguientes:

```
$ sudo mysql -u root
CREATE DATABASE “nombrebase”;
USE “nombrebase”;
CREATE TABLE “nombretabla” (time TIMESTAMP, “nombredato” tipo dato, etc );
DESCRIBE “nombretabla”;
```



Fuente: Captura de terminal.

Figura 2-5 :Ejemplo de comandos base de datos



Fuente: Captura de pantalla terminal.

Figura 2-6 : Descripción de tabla base de datos

Luego, ya creada la tabla para almacenar la información, como se ve en figura 2-5 y 2-6, hay que agregar las siguientes líneas de código en Python para que reconozca a donde se enviaran los datos:

```
import MySQLdb   ##Base de datos

db = MySQLdb.connect(host="localhost",user="raspi", passwd="raspberry",db="XBEE")

cur = db.cursor() ###Crea cursor para consulta
```

Estos comandos permiten reconocer la base de datos desde Python. Finalmente, para registrar los datos se ingresan las siguientes líneas en donde se esté ejecutando el programa:

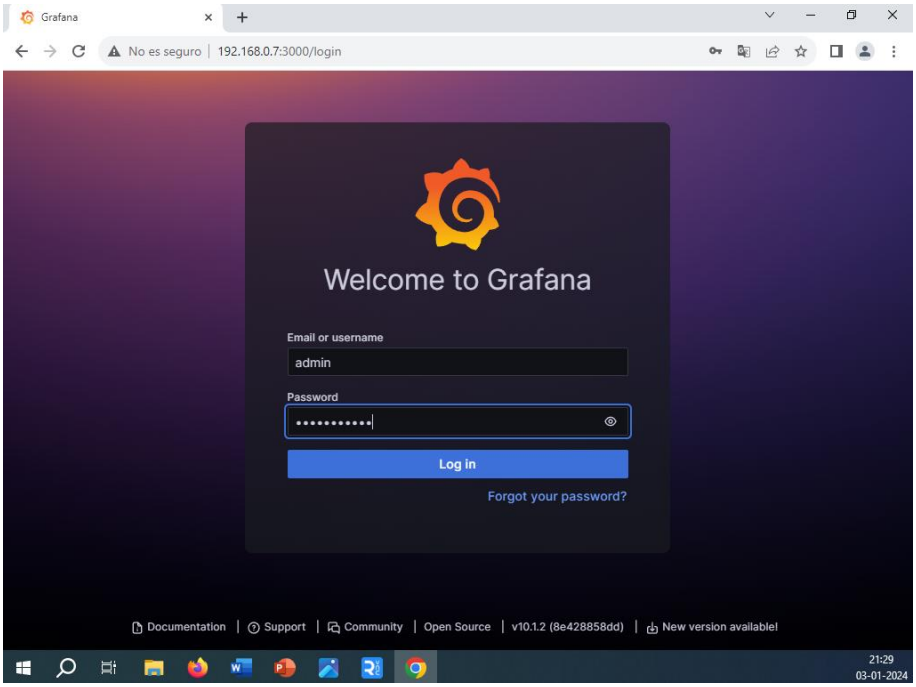
```
cur.execute('''INSERT INTO XB_Data(digital,analoga,temperatura,presion) VALUES(%s,%s,%s,%s);''',(med_1,med_2,temp_ard,presion_ard))

db.commit()
```

Para comprender de mejor manera el contexto de los comandos lo ideal sería referirse a la documentación de MySQL/MariaDB disponible en internet.

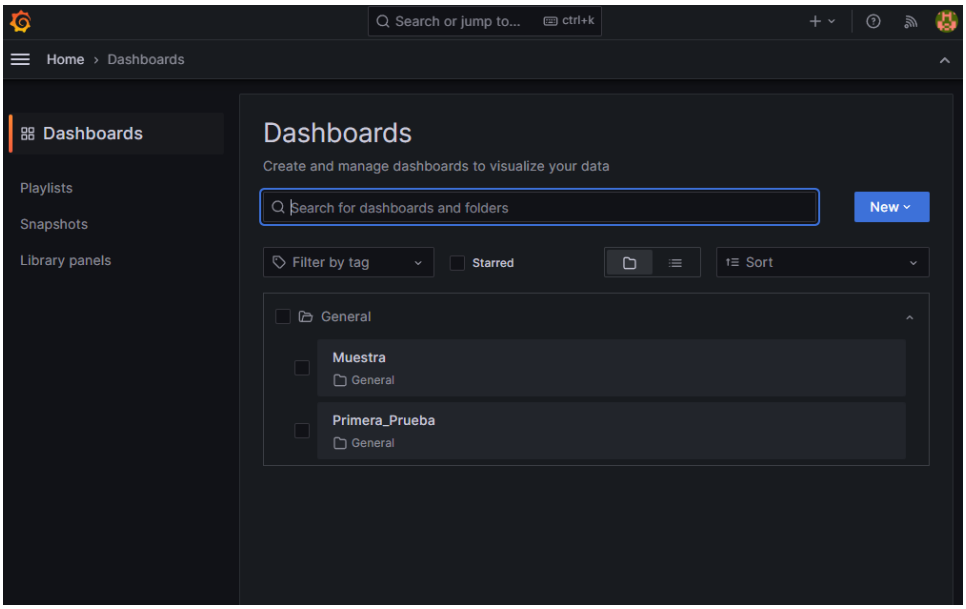
2.7. USUARIO GRAFANA, CONFIGURACIÓN DASHBOARD Y CONSULTA

Como se comentó en el punto 2.4.2, Grafana ya fue instalado en la raspberry en la dirección IP:3000, por lo que solo bastaría ingresar desde un navegador en la misma red que la raspberry, como se muestra en la figura 2-7 con credenciales admin y raspberrypi



Fuente: Captura de pantalla Google Chrome.

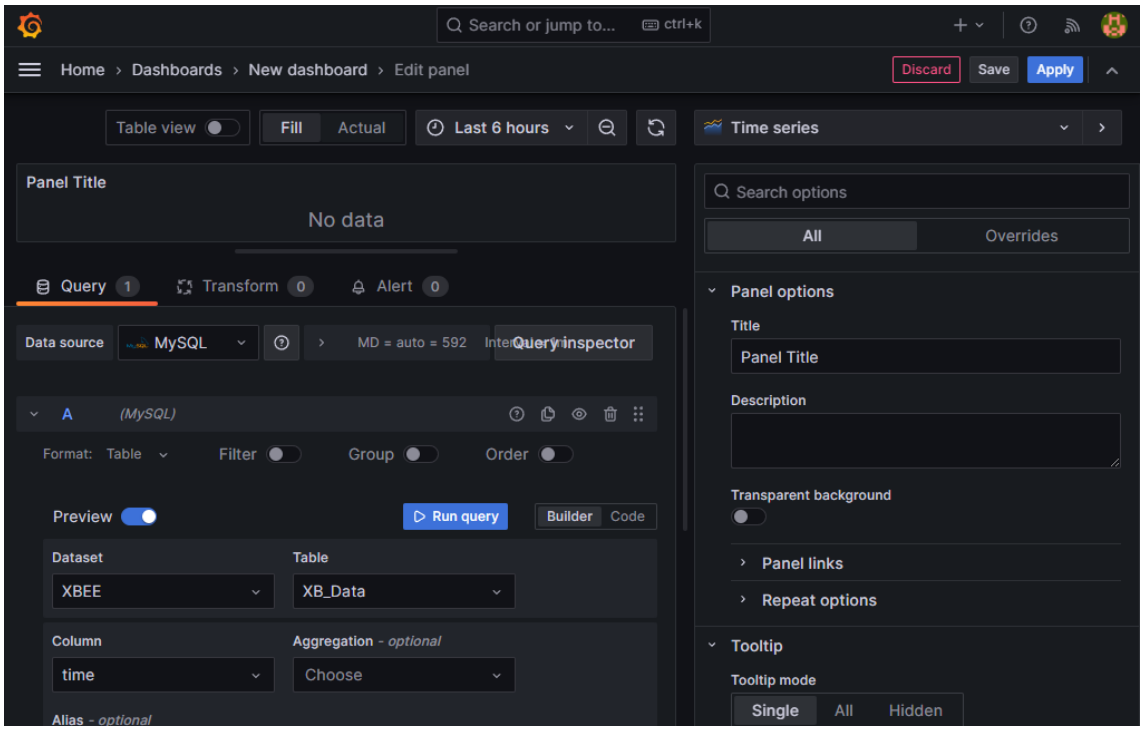
Figura 2-7 : Entrada a Grafana



Fuente: Captura de navegador.

Figura 2-8 : Creación de dashboard

En el panel de dashboard, visible en figura 2-8, se selecciona nuevo y se agrega la base de datos de MySQL.



Fuente: Captura de pantalla.

Figura 2-9: Configuración Dashboard

Para terminar de crear el dashboard se selecciona la base en Dataset y la tabla. Luego se coloca que datos se quiere por columna y se da click en Run Query como se observa en la figura 2-9, esto da por finalizado el proyecto. Los resultados se presentan en el capítulo 3.

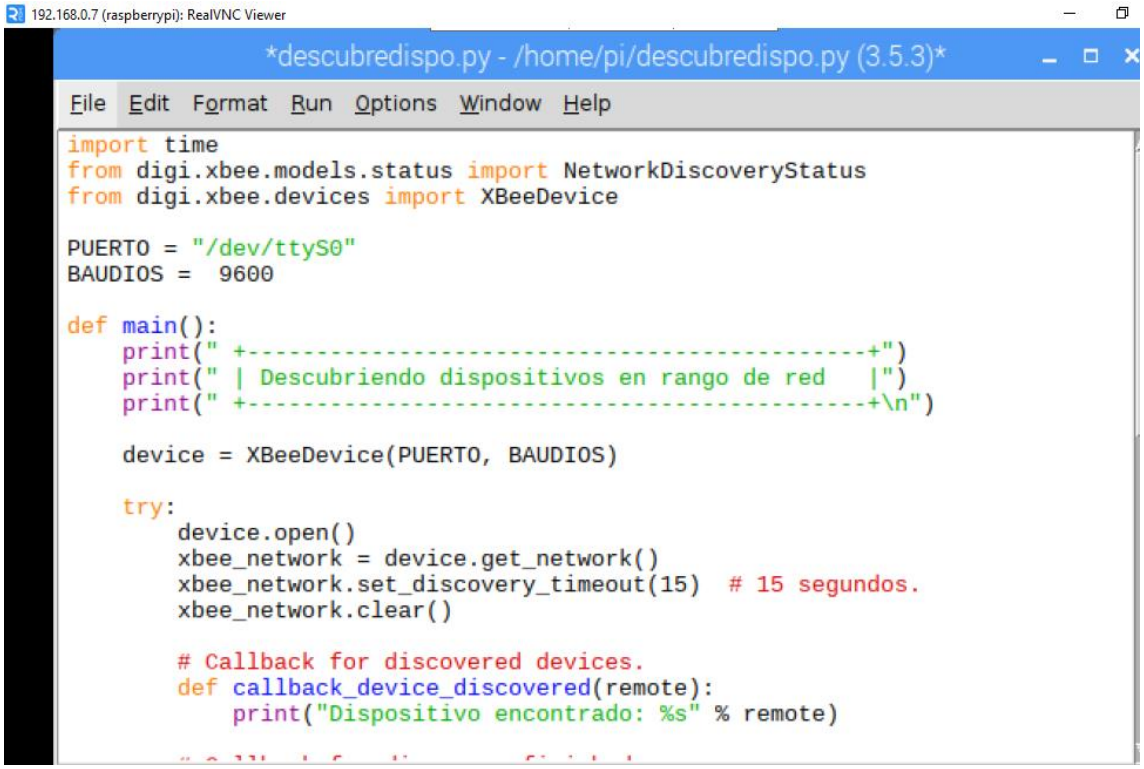
CAPÍTULO 3: RESULTADOS, PRESUPUESTOS Y CONCLUSIONES

3. RESULTADOS, PRESUPUESTOS Y CONCLUSIONES

En este capítulo se presentan los resultados finales del desarrollo de este proyecto. Cada uno de estos resultados responde a los objetivos que inicialmente se plantearon para este trabajo, por lo que se da una conclusión a este demostrando su funcionalidad y dando un ejemplo de una posible aplicación real en un entorno laboral conocido. Además, se presenta una Carta Gantt que sirvió de guía para los inicios de este proyecto junto a presupuestos relacionados a materiales para que esta red pueda ser implementada por alguna persona, organización o entidad.

3.1. RESULTADO DE TOPOLOGÍA ESTRELLA

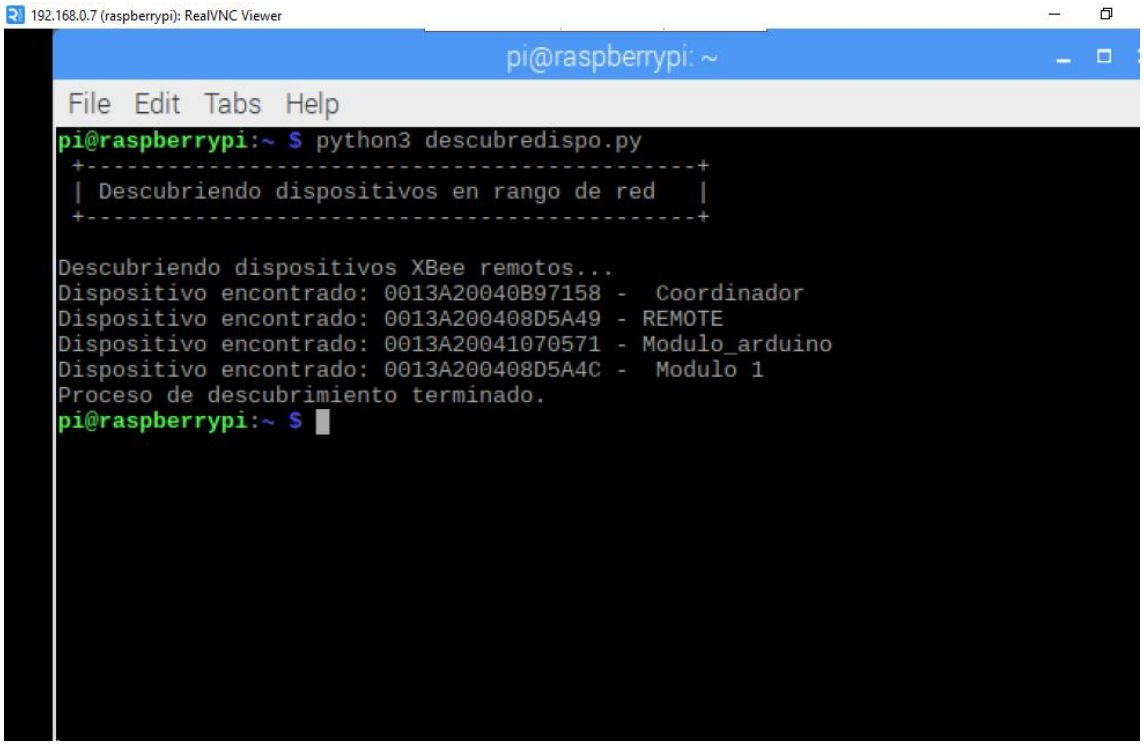
Para crear la topología estrella se han configurado todos los XBee en la dirección PAN ID: 2. Por lo tanto, estos deberían ser reconocibles al hacer la consulta de la network del coordinador a través del comando que provee la librería digi-xbee. En la figura 3-1, se muestra un ejemplo de código para verificar la topología:



Fuente: Captura de pantalla VNC Viewer RPi.

Figura 3-1: Captura código red estrella

En el código, es posible identificar que a comienzo se define la dirección del puerto UART de la Raspberry y la velocidad de comunicación en Baudios. Luego según la documentación de la librería se abre el dispositivo coordinador y se usa la función get_network() para identificar la red en la cual está configurado y qué otros dispositivos se encuentran en esta. El resultado de la ejecución del código anterior a través de la consola es el siguiente:



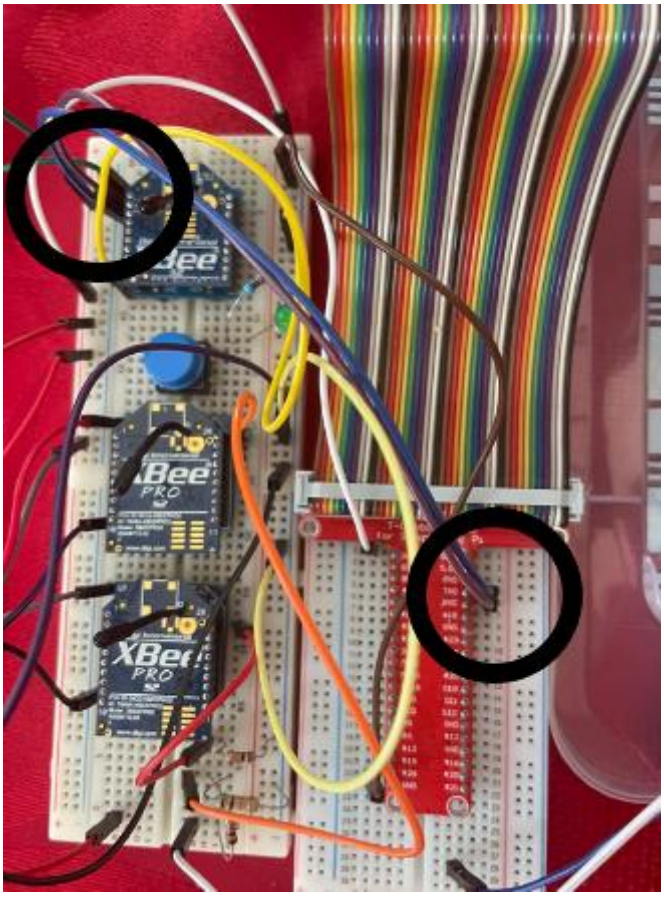
Fuente: Captura de pantalla de consola RPi.

Figura 3-2: Resultado de código para topología estrella.

En la figura 3-2 se puede ver que el código realiza una búsqueda de dispositivos disponibles en la red pan ID del coordinador. Luego, retorna el tag único asociado a cada uno de los XBee junto al nombre que se le asignó a cada uno de los dispositivos en la configuración. Para hacer la programación un tanto más amigable, se utilizó el nombre para el código del funcionamiento principal. Sin embargo, como se ve en la figura 3-2, se podría utilizar el tag único como opción para programar.

3.2. RESULTADO DE COMUNICACIÓN ENTRE RPI Y XBEE COORDINADOR

En el punto 3.1 se puede ver que a nivel de programación la Raspberry reconoce el puerto del XBee, por lo que si existe una comunicación entre ambos dispositivos. Además, en los círculos negros de la figura 3-3 se puede ver cómo están unidos físicamente estos dos puertos a través de dos cables dupont, lo que permite que al momento de ingresar la dirección del puerto de la Raspberry esta reconozca al coordinador del otro lado.



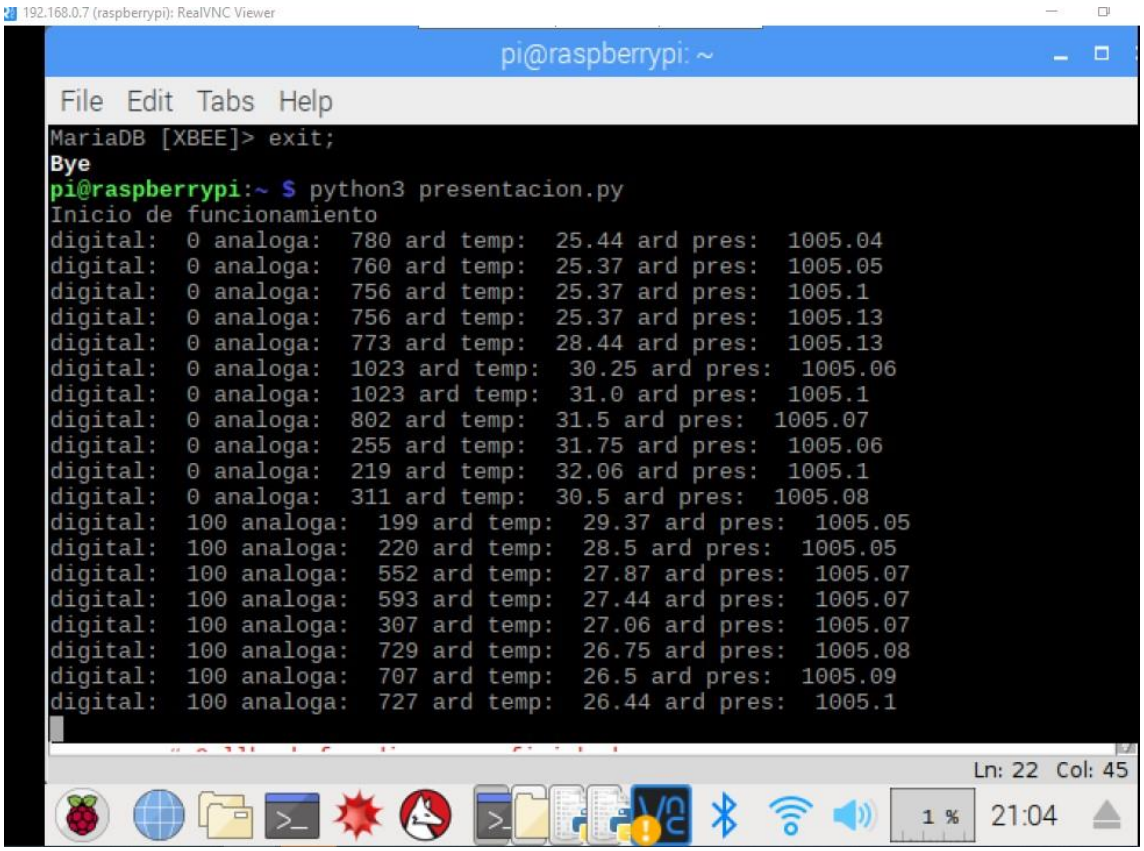
Fuente: Fotografía propia.

Figura 3-3: Cableado puerto UART

Es importante recalcar que el cableado del puerto UART entre un dispositivo y otro tiene que ser cruzado. Ósea que el pin de transmisión (TX) tiene que ir conectado al de recepción (RX) y viceversa.

3.3. VISUALIZACIÓN DE MEDICIONES EN DASHBOARD

A modo de cumplir con el objetivo de visualizar los datos en Dashboard, a continuación, en la figura 3-4 se presenta una muestra de mediciones que es cargada en la base de datos y puede verse a través de un navegador de cualquier dispositivo conectado a la misma red wifi de la Raspberry Pi. La visualización final de estos datos se presenta en la figura 3-5 y 3-6.

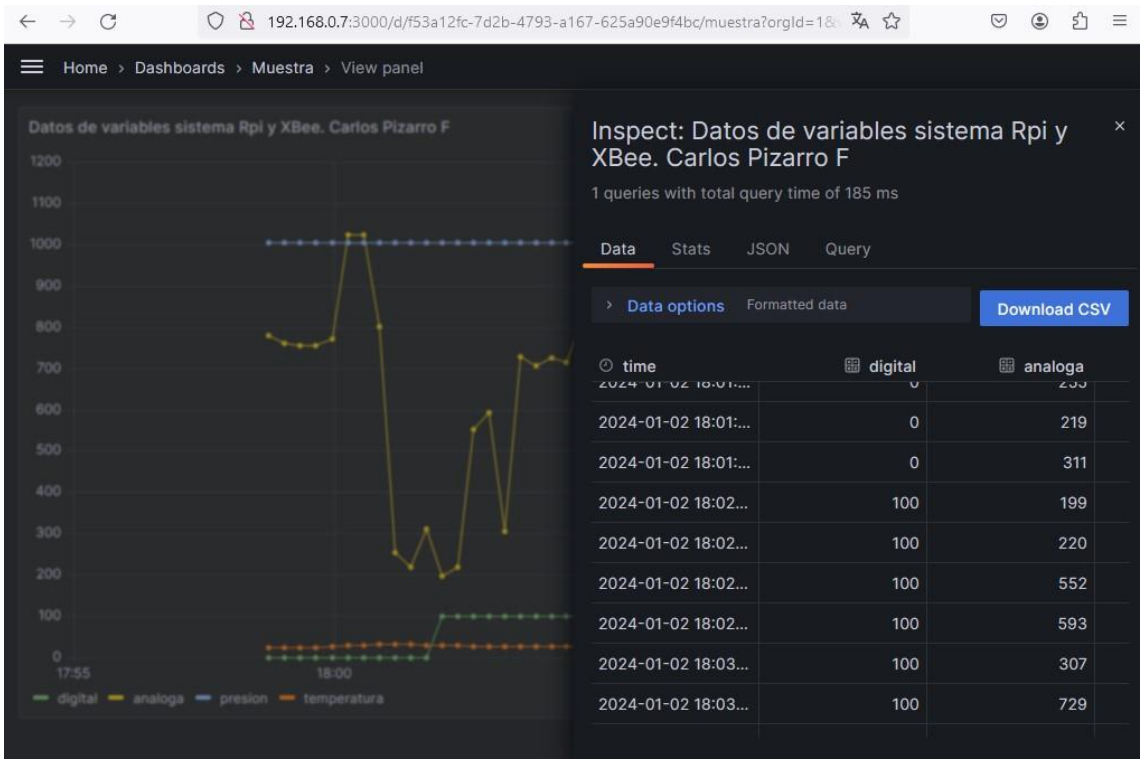


Fuente: Captura de Terminal RPi.

Figura 3-4: Mediciones de programa ejecutándose

Fuente: Captura de pantalla Grafana

Figura 3-5: Datos en dashboard navegador Mozilla Firefox



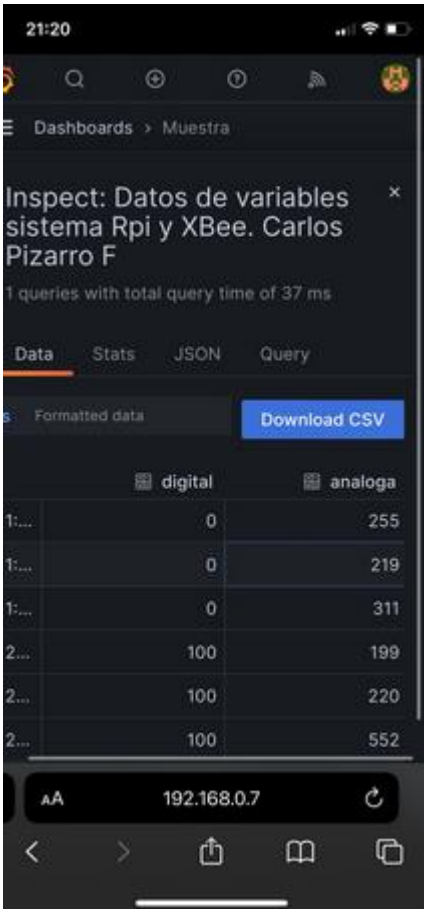
Fuente: Captura de pantalla Grafana.

Figura 3-6: Datos en el tiempo v/s tabla de valores

Fuente: Captura de pantalla teléfono navegador Safari.

Figura 3-7: Captura teléfono navegador Safari

Para el caso de la figura 3-7, los datos que se ven en el tiempo son los mismos de la figura 3-6, solo que desde el teléfono no está la posibilidad de hacer un zoom en el eje temporal. A continuación, en la figura 3-8, se presenta la visualización de la tabla de la figura 3-7 desde el teléfono.

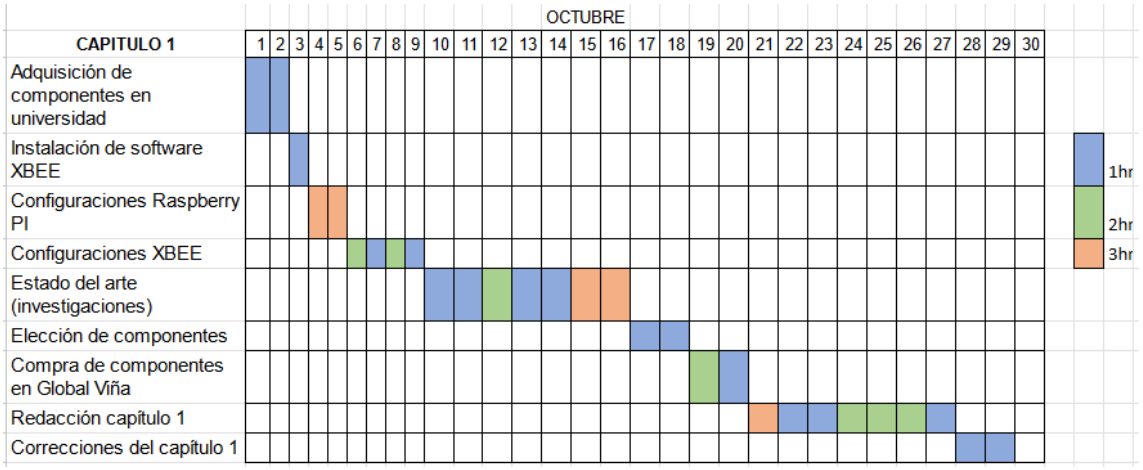


Fuente: Captura de pantalla teléfono

Figura 3-8: Tabla de datos en navegador Safari

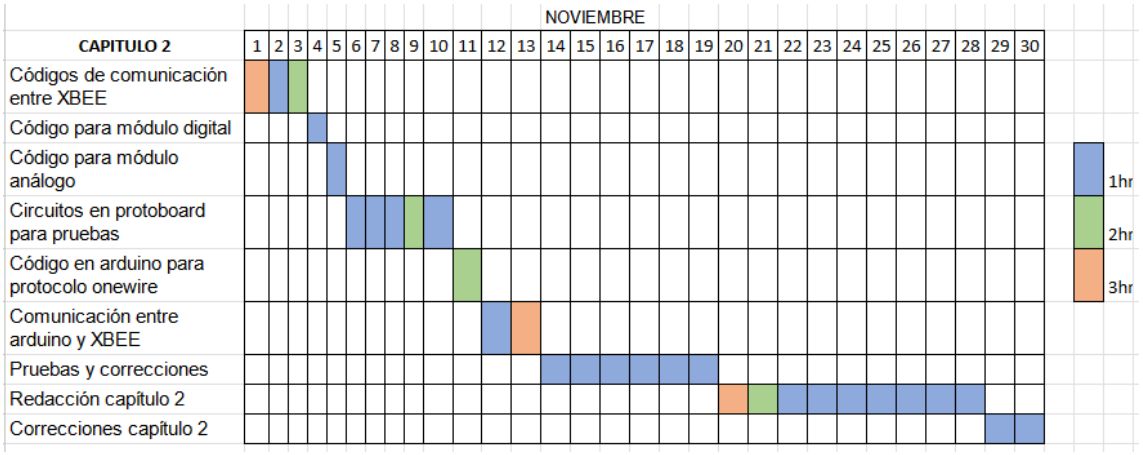
3.4.- CARTA GANTT Y PRESUPUESTO DE MATERIALES

La asignatura de trabajo de título tiene la duración de un semestre regular, por lo que durante el desarrollo del proyecto se escribió una lista de tareas necesarias para cumplir con los objetivos planteados. Estas tareas fueron dimensionadas en hora para tener una cuantificación del trabajo realizado en lo que duró el ramo y se presentan en formato Carta Gantt en las figuras 3-9, 3-10 y 3-11.



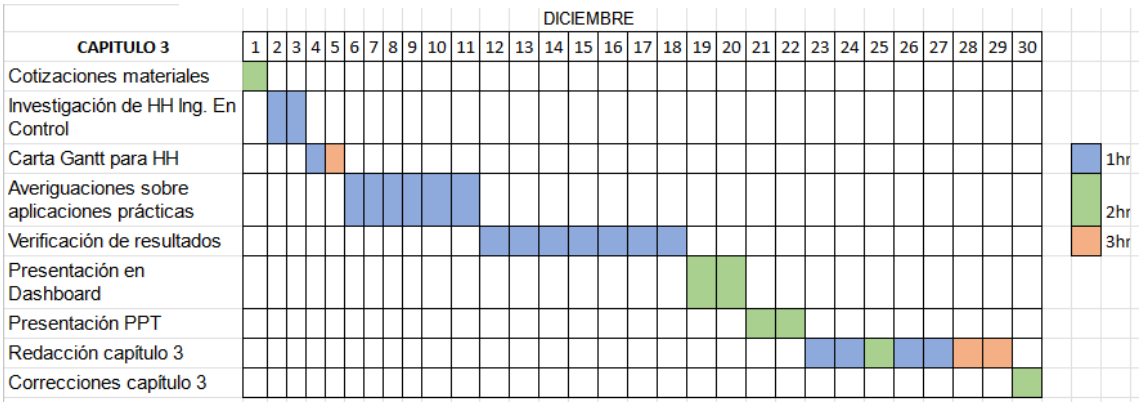
Fuente: Elaboración propia en Google Sheets.

Figura 3-9: Carta Gantt de Octubre



Fuente: Elaboración propia en Google Sheets.

Figura 3-10: Carta Gantt de Noviembre



Fuente: Elaboración propia en Google Sheets.

Figura 3-11: Carta Gantt de Diciembre

Además de la cuantificación de las tareas en horas, se ha creado un presupuesto de los materiales que se utilizaron para este trabajo, dando como resultado un valor total de aproximadamente 10 UF, con una UF de 36.789,36 el 31 de diciembre del año 2023.

Tabla 3-1. Presupuesto de proyecto

	Valor unitario ($)	**Cantidad**	**Costo total (CLP)**	**Costo total (UF)**
XBEE	23.467	4	93.868	2,55
Raspberry Pi	72.863	1	72.863	1,98
GPIO Raspberry	6.319	1	6.319	0,17
Arduino UNO	8.047	1	8.047	0,22
Sensor BMP280	4.062	1	4.062	0,11
LDR	1.278	1	1.278	0,035
Sensor KY-037	1.094	1	1.094	0,03
Cables dupont	2.826	30	84.780	2,3
Resistencias 10k	3.340	5	16.700	0,45
USB Explorer XBEE	17.190	4	68.760	1,87
Dallas 18b20	3.139	1	3.139	0,085
LM-555	1.200	1	1.200	0,033
Total			**362.110**	**9,833**

Fuente: Elaboración propia.

3.5. SUGERENCIA DE APLICACIÓN DE PROYECTO EN MINERIA

El objetivo final de este proyecto es presentar alguna aplicación de este sistema en la realidad. Para esto se ha escogido una experiencia personal del alumno que escribe este trabajo: A finales del año 2020 el estudiante tuvo la oportunidad de trabajar como

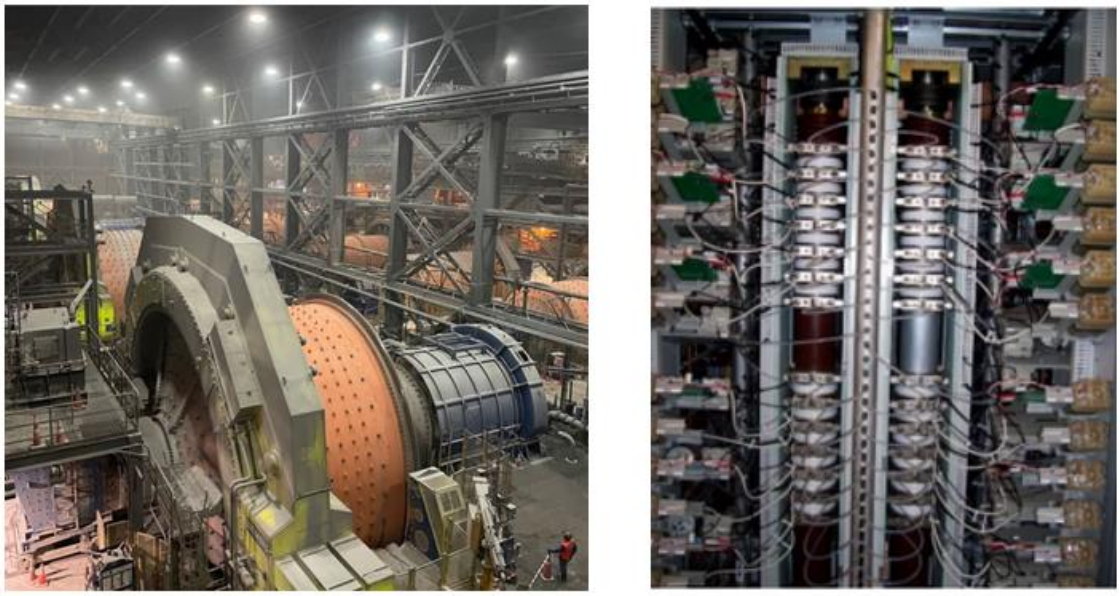
Ayudante en la empresa contratista Engie que se encarga del sistema de HVAC para el sistema de Molienda en la planta concentradora de Minera Los Pelambres, la cual se presenta en la figura 3-12.

Fuente: www.mineriaydesarrollo.com

Figura 3-12: Concentradora Minera Los Pelambres

Dentro de las labores que tenía el equipo estaba la de dar mantención a equipos Chiller que se encargan de enfriar el ciclo conversor de Molinos SAG asociados a la línea de molienda 1 y 2. El equipo hace circular agua fría desmineralizada a través del circuito del ciclo conversor lo que permite reducir la temperatura y evitar que el proceso de molienda se detenga inesperadamente por sobre calentamiento de alguna de sus partes. Esta labor tiene una gran relevancia de cara al cliente, ya que asegura una producción continua que en caso de ser detenida afectaría económicamente en altas cifras a la empresa mandante y por consiguiente a la contratista.

A la izquierda en la figura 3-13 se puede observar la nave de molienda de Minera Los Pelambres, la cual contaba con 3 líneas de molinos SAG en el año 2020. En esa foto en particular, de frente se observa el molino SAG 2. Mientras que los que se encuentran detrás de los pilares de metal son molinos de bola por lo que esta empresa no se involucra con ese tramo del proceso. Además, a la derecha se presenta el ciclo conversor correspondiente a un molino SAG, el cual permite modificar la velocidad del motor del molino.



Fuente: Fotos propias del año 2020.

Figura 3-13: Molienda MLP y Ciclo conversor Molino SAG

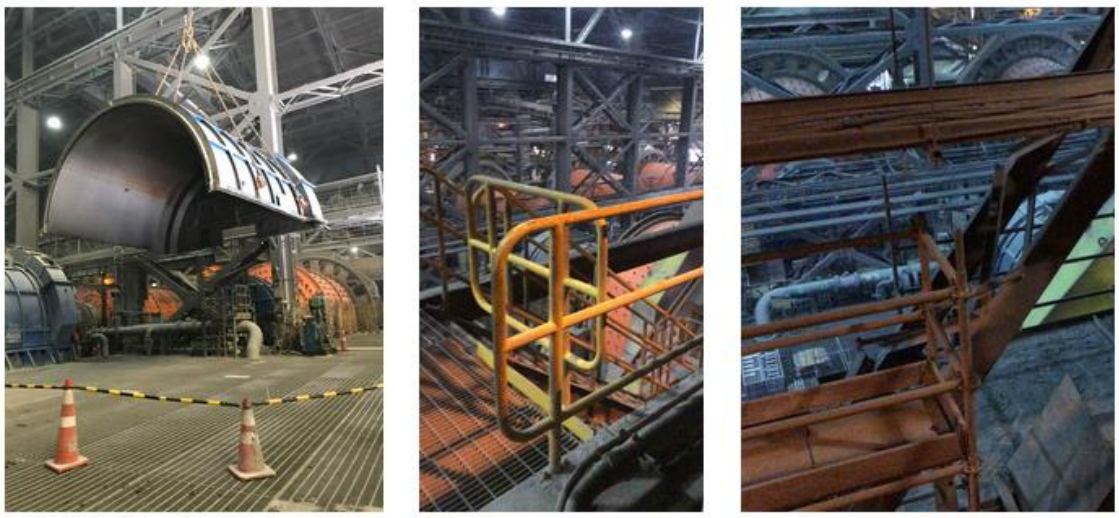
Por el entorno en el que se encuentran distribuidas las máquinas, es necesario las mantenciones recurrentes fuera de programación, especialmente en verano.

Fuente: Fotos propias de 2020.

Figura 3-14: Mantención a Chiller

Debido a que no es posible detener la producción fuera de programación para hacer una mantención adecuada a la máquina, la solución que encontró la empresa fue hacer una limpieza superficial mostrada en la figura 3-14 que incluye el soplado de tierra de ventiladores y rociar los serpentines del chiller con agua de proceso. Además, la máquina cuenta con un monitoreo remoto a través de PI SYSTEM como en varias faenas. El problema de esto es que esa temperatura solo se encuentra accesible desde la sala de control o desde un computador ubicado en la oficina de la contratista la cual está situada a unos 8 km de la planta concentradora, por lo que las únicas opciones para monitorear la máquina terminan siendo: llamar por radio constantemente saturando las líneas de comunicación generales o mantenerse junto a la máquina toda la jornada.

Mantenerse junto a la máquina parece lo lógico ya que está dentro de las labores del cargo, pero entre el polvo generado por la molienda, el ruido en altos decibeles provocado por los molinos y las altas temperaturas a causa del verano, esto se hace muy pesado para el trabajador. Un alivio sería que se pudiera salir por momentos de la nave de molienda, pero dado que los equipos se encuentran en el tercer piso, mantenciones de otra área y trabajos cruzados como se ve en la figura 3-15, se hace imposible alejarse de la máquina sin arriesgar la detención del proceso en el cual el Molino semi autógeno demoraría alrededor de 15 minutos en volver a partir.



Fuente: Fotos propias del 2020.

Figura 3-15:Trabajos cruzados y piso 3 MLP

Como solución a esta condición del trabajador lo que se propone es acoplar este proyecto de título al presostato de la máquina Chiller que se puede observar en la figura 3-16:

Fuente: Fotos propias del 2020.

Figura 3-16: Presostato y mantención a Chiller

Esto permitiría que el trabajador pueda monitorear desde su teléfono conectado a la misma red de MLP la presión y la temperatura del sistema pudiendo alejarse de la máquina controladamente sin arriesgar la detención de la producción, generando un alivio directo para el personal.

CONCLUSIONES

- Los módulos XBee son robustos al momento de querer monitorear procesos industriales, ya que la posibilidad de extender la red con routers en el protocolo Zigbee permite expandir el monitoreo en grandes distancias.
- Este sistema tendría varias limitaciones a la hora de querer usar señales de control en actuadores, ya que el voltaje con el que trabaja XBee es de máximo 3,3 volts. Lo cual podría presentar algunos problemas al momento de querer acondicionar las señales, necesitando de fuentes externas o reductores de voltaje.
- Raspberry es un buen sistema para crear la base de datos y almacenarlos, ya que al poder programarse con Python brinda varias ventajas en librerías y posterior procesamiento de datos. Sin embargo, el sistema podría ser algo caro si es que solamente se pretende monitorear valores y no hacer un análisis de datos. Podría reemplazarse la Raspberry por una ESP-32.
- El sistema puede ampliarse bastante ya que teniendo 3 módulos XBee con 15 E/S cada uno tendría la opción de conectar 45 sensores de distintos tipos, por lo tanto, ya con un solo módulo se pueden monitorear bastantes cosas en simultáneo. Junto a esto, MariaDB permite crear más tablas de datos y Grafana cargar paneles de datos con otras tablas. Por lo que este mismo sistema ya ofrece grandes oportunidades desde el punto de vista de la IoT.
- A pesar de las planificaciones con Carta Gantt, este trabajo tomó más tiempo de lo presupuestado por problemas que se presentaron con los sensores, por lo que en la planificación faltó considerar tiempos muertos por errores.

LINKOGRAFÍA

Raspberry Pi. (s.f). *Raspberry Pi hardware*. Raspberry Pi. Revisado el 21 de octubre de 2023 desde https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

Cassero, S. (2016). *Design and Analysis of Arduino, Raspberry Pi, and Xbee based Wireless Sensor Networks*. eScholarship. Revisado el 25 de octubre de 2023 desde https://escholarship.org/uc/item/5q8323dv#article_main

Merello, A., Fernández, L. (2023). *Monitor temperature and humidity with Grafana and Raspberry Pi*. Grafana Labs. Revisado el 1 de noviembre de 2023 desde https://grafana.com/blog/2023/10/23/monitor-temperature-and-humidity-with-grafana-and-raspberry-pi/?utm_source=grafana_news&utm_medium=rss

Malmsten, P. (2017). *Python-XBee Documentation*. Revisado el 4 de noviembre de 2023 desde https://readthedocs.org/projects/python-xbee/downloads/pdf/stable/

Digi International Inc. (2017). *XBee Python Library*. Digi XBee Python Library. Revisado el 25 de septiembre de 2023 desde https://xbplib.readthedocs.io/en/latest/

Ijorquera. (s.f). *Configuración XBEE Serie 2*. XBee.cl. Revisado el 25 de septiembre de 2023 desde https://xbee.cl/xbee-serie-2-configuracion/

Vera Romero, C. A., Barbosa Jaimes, J. E., & Pabón González, D. C. (2015). *Parámetros de configuración en módulos XBEE-PRO® S2B ZB para medición de variables ambientales*. SciELO. Revisado el 4 de octubre de 2023 desde http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0123-921X2015000300012

Digi XBee Python Library. (s.f). Github.com. Revisado el 7 de octubre de 2023 desde https://github.com/digidotcom/xbee-python

Jain, R. (2019). *How to interface XBee module with Raspberry Pi*. Circuit Digest. Revisado el 7 de octubre de 2023 desde https://circuitdigest.com/microcontroller-projects/raspberry-pi-xbee-module-interfacing

Harrell, J. (2023). *Sending Data Strings to XBee using Python Library: A Guide*. CopyProgramming. Revisado el 15 de octubre de 2023 desde https://copyprogramming.com/howto/how-can-i-send-strings-of-data-to-an-xbee-with-a-python-library#google_vignette

Grafana Labs Team. (s.f). *Install Grafana on Raspberry Pi*. Grafana Labs. Revisado el 21 de octubre de 2023 desde https://grafana.com/tutorials/install-grafana-on-raspberry-pi/

Dignani, J. (2011). *Análisis del protocolo ZigBee*. Universidad Nacional de la Plata. Consultado el 13 de septiembre de 2023 desde: https://postgrado.info.unlp.edu.ar/wp-content/uploads/2014/07/Dignanni_Jorge_Pablo.pdf

PureStorage. (s.f). *¿Qué es MariaDB y cómo funciona?*. PureStorage.com. Consultado el 13 de septiembre de 2023 desde: https://www.purestorage.com/es/knowledge/what-is-mariadb.html

ANEXOS

A.- Código Arduino

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define BMP_SDA 21  // Cambiar estos valores según tus conexiones I2C
#define BMP_SCL 22

const int pinOneWire = 2;
OneWire oneWire(pinOneWire);
DallasTemperature sensors(&oneWire);
float temp;
float pres;
Adafruit_BMP280 bmp;  // Instancia del sensor BMP280
char cadena[20];
void setup() {
  Serial.begin(9600);
  //Wire.begin(BMP_SDA, BMP_SCL);
  sensors.begin();
  // Inicializar el BMP280 con dirección I2C 0x77
  if (!bmp.begin(0x77)) {
    Serial.println("No se pudo inicializar el BMP280. Comprueba las
conexiones y la dirección I2C.");
    while (1);
  }

  Serial.println("BMP280 iniciado correctamente");

}

void loop() {
  //Serial.print("Temperatura = ");
  //Serial.print(bmp.readTemperature());
  //Serial.println(" *C");
  sensors.requestTemperatures();
  float temperatura = sensors.getTempCByIndex(0);

  // Verifica si la lectura es válida (no es -127)
  if (temperatura != -127) {
    // Imprime la temperatura en el Monitor Serie
    //Serial.print("Temperatura: ");
```

```
    pres=(bmp.readPressure() / 100.0F);
    Serial.print(temperatura);
    Serial.print(temp);
    Serial.print(",");
    Serial.println(pres);
    //Serial.println(" °C");
  } else {
    //Serial.println("Error al leer la temperatura");
  }

  //delay(1000);
  //temp=bmp.readTemperature();
  //pres=(bmp.readPressure() / 100.0F);
  //Serial.print("Presión = ");
  //char cadena[50];
  //sprintf(cadena, "temp: %.2d, pres: %.2f", temp, pres);
  //Serial.print(temp);
  //Serial.print(",");
  //Serial.println(pres);
  //Serial.println(pres);
  //Serial.println(" hPa");

  //Serial.print(F("Approx altitude = "));
  //Serial.print(bmp.readAltitude(1013.25)); /* Adjusted to local
forecast! */
  //Serial.println(" m");

  delay(1000);
}
```

B.- Código principal Python

```
from digi.xbee.devices import XBeeDevice  ##libreria XBee
from digi.xbee.devices import ZigBeeDevice  ##Libreria para conexiones
remotas
from digi.xbee.io import IOLine, IOMode, IOValue  ##Libreria para
entradas/salidas

import time
import datetime
from datetime import date    ##Fechas para BDD
import MySQLdb   ##Base de datos
from enum import Enum


puerto = '/dev/ttyS0'  ##PUERTO RPI

baud = 9600

#NODO_REMOTO_ID="NODO NULO"  ## Declaración para identificar nodo

IOLINE_IN = IOLine.DIO1_AD1   ## Se declara como entrada
```

```
db = MySQLdb.connect(host="localhost",user="raspi",
passwd="raspberry",db="XBEE") #conecta con  MySQL/MariaDB
cur = db.cursor() #crea el cursor para las peticiones de
MySQL/MariaDB
presion_ard=None
temp_ard=None

def main():

    #stop=False
    #th = None

    #local_device = XBeeDevice(puerto,baud)
    #inicio = time.time()
    bandera=1
    while (bandera<4):

        if (bandera==1):

              medicion_digital=registro_digital()
              #print(type(medicion_digital))
              med_1=float(medicion_digital)

              bandera=2
        if (bandera==2):
              #medicion_digital=registro_digital()
              medicion_analoga=registro_analogo()
              med_2=float(medicion_analoga)
              #print('info ',med_2,type(med_2))

              bandera=3

        if (bandera==3):
              medicion_arduino=registro_arduino()
              #print('tipo valor arduino, ', type(medicion_arduino))
              #print('temp: ', temp_ard, 'pres: ', presion_ard)
              bandera=1

        fecha = date.today()
        hora = datetime.datetime.now().time()
        #print(fecha),"-",hora.replace(microsecond=0)
        print('digital: ',medicion_digital,'analoga:
',medicion_analoga,'ard temp: ', temp_ard,'ard pres: ', presion_ard)
        time.sleep(0.5)
        cur.execute('''INSERT INTO
XB_Data(digital,analoga,temperatura,presion)
VALUES(%s,%s,%s,%s);''',(med_1,med_2,temp_ard,presion_ard))
        db.commit()
    #fin = time.time()

def registro_analogo():

    puerto = '/dev/ttyS0'
    #stop=False
    #th = None   threading opcional

    xbee_analogo = XBeeDevice(puerto,baud)
    xbee_analogo.open()
    #print("reconocio el xbee: ", xbee_analogo)
```

```
    NODO_REMOTO_ID1=" Modulo 1"  ##Nombre de nodo análogo, digital
"REMOTE"

    xbee_network = xbee_analogo.get_network()
    remoto_analogo = xbee_network.discover_device(NODO_REMOTO_ID1)
    nombre_nodo = xbee_analogo.get_node_id()


    if remoto_analogo is None:
        print("No se encuentra el dispositivo remoto")
        exit(True)
    remoto_analogo.set_io_configuration(IOLINE_IN, IOMode.ADC)

    def leer_entrada_analoga():
        a=1
        while (a<2):
            # Lee valor analogo de la linea input
            valor_analogo = remoto_analogo.get_adc_value(IOLINE_IN)
            #print("%s: %d" % (IOLINE_IN, valor_analogo)) Revisa
printeo
            a+=1
            analogo=valor_analogo
            time.sleep(0.2)
        return analogo
    lectura_analoga=leer_entrada_analoga()

    bandera=2
    if xbee_analogo is not None and xbee_analogo.is_open():
        xbee_analogo.close()
    return lectura_analoga


def registro_digital():

    #stop = False
    #th = None
    puerto = '/dev/ttyS0'
    xbee_digital = XBeeDevice(puerto,baud)
    ENTRADA_IO = IOLine.DIO3_AD3
    SALIDA_IO = IOLine.DIO4_AD4
    NODO_REMOTO_ID2="REMOTE"  ##NOMBRE DE NODO
    xbee_digital.open()
    red_digital=xbee_digital.get_network()
    remoto_digital=red_digital.discover_device(NODO_REMOTO_ID2)

    if remoto_digital is None:
        print("No se encuentra el dispositivo digital")
        exit(1)
    def deteccion_io():
        n=0
        #print("en deteccion")
        while (n<1):
            valor_io=remoto_digital.get_dio_value(ENTRADA_IO)
            #print("%s: %s" % (ENTRADA_IO, valor_io))

            xbee_digital.set_dio_value(SALIDA_IO, valor_io) ##prueba
            n+=1
            time.sleep(0.2)
            digital=valor_io
            #class digi.xbee.io.IOValue(code):
            #IOValue.LOW=0
```

```
            #IOValue.HIGH=1
            #print(digi.xbee.io.IOValue)



            if(valor_io == IOValue.LOW):
                #print('success')
                digital=0
            else:
                digital=100
        return digital
    #print("max",read_io_sample().digital_values.keys())
    remoto_digital.set_io_configuration(ENTRADA_IO, IOMode.DIGITAL_IN)
    xbee_digital.set_io_configuration(SALIDA_IO,
IOMode.DIGITAL_OUT_LOW)

    lectura=deteccion_io()
    #print("la funcion es: ", lectura)
    if xbee_digital is not None and xbee_digital.is_open():
        xbee_digital.close()
    time.sleep(0.2)

    return lectura


def registro_arduino():
    puerto = '/dev/ttyS0'
    xbee_arduino = XBeeDevice(puerto,baud)
    #ENTRADA_IO = IOLine.DIO3_AD3
    #SALIDA_IO = IOLine.DIO4_AD4
    NODO_REMOTO_ID3="Modulo_arduino"  ##NOMBRE DE NODO
    #xbee_arduino.open()
    #red_arduino=xbee_arduino.get_network()
    #remoto_arduino=red_arduino.discover_device(NODO_REMOTO_ID3)
    global temp_ard
    global presion_ard
    try:
        xbee_arduino.open()

        xbee_arduino.flush_queues()

        #print("Probando datos por UART")
        flag=1

        while (flag<2):
            xbee_message = xbee_arduino.read_data()
            if xbee_message is not None:
                #print("From %s >> %s" %
(xbee_message.remote_device.get_64bit_addr(),
                                    #      xbee_message.data.decode()))
                msje=xbee_message.data.decode()
                #print(msje,type(msje))
                msje_separado=msje.split(",")
                temp=msje_separado[0]
                presion=msje_separado[1]
                presion_editada=presion[:-2]
                temp_editada=temp.rstrip(".0")
                temp_flotante=float(temp_editada)
                temp_ard=temp_flotante
                #print(msje_separado[0])
                #print(msje_separado[1])
```

```
                #print(presion_editada)
                #print(temp_flotante)
                #print(type(temp_flotante))
                presion_flotante=float(presion_editada)
                presion_ard=presion_flotante
                #print(presion_flotante)
                flag=2
    finally:
        if xbee_arduino is not None and xbee_arduino.is_open():
            xbee_arduino.close()

    return temp_ard, presion_ard


if __name__ == '__main__':
    print("Inicio de funcionamiento")
    main()
```

C.- Código topología estrella Python

```
import time

from digi.xbee.models.status import NetworkDiscoveryStatus
from digi.xbee.devices import XBeeDevice

PUERTO = "/dev/ttyS0"
BAUDIOS =  9600


def main():
    print(" +---------------------------------------------+")
    print(" | Descubriendo dispositivos en rango de red   |")
    print(" +---------------------------------------------+\n")

    device = XBeeDevice(PUERTO, BAUDIOS)

    try:
        device.open()

        xbee_network = device.get_network()

        xbee_network.set_discovery_timeout(15)  # 15 segundos.

        xbee_network.clear()

        # Callback for discovered devices.
        def callback_device_discovered(remote):
            print("Dispositivo encontrado: %s" % remote)

        # Callback for discovery finished.
        def callback_discovery_finished(status):
            if status == NetworkDiscoveryStatus.SUCCESS:
                print("Proceso de descubrimiento terminado.")
            else:
                print("Hubo un error descubriendo dispositivos: %s" %
status.description)
```

```
xbee_network.add_device_discovered_callback(callback_device_discovered
)


xbee_network.add_discovery_process_finished_callback(callback_discover
y_finished)

        xbee_network.start_discovery_process()

        print("Descubriendo dispositivos XBee remotos...")

        while xbee_network.is_discovery_running():
            time.sleep(0.1)

    finally:
        if device is not None and device.is_open():
            device.close()


if __name__ == '__main__':
    main()
```