UNIVERSIDAD AUTÓNOMA DE SAN LUIS POTOSÍ

FACULTAD DE CIENCIAS

UNIDAD CENTRAL DE MONITOREO Y COMUNICACIÓN IOT PARA SISTEMAS DE RIEGO CON APLICACIÓN EN HUERTOS URBANOS

TESIS PROFESIONAL

PARA OBTENER EL TÍTULO DE

INGENIERO EN TELECOMUNICACIONES

Presenta

ADRIAN FÍVEL MARTÍNEZ RANGEL

Asesor de Tesis:

DR. JOSÉ MARTÍN LUNA RIVERA

Agradecimientos

Agradezco al **Dr. José Martín Luna Rivera** por todo el apoyo, la paciencia, la guía, la disciplina y pasión en su trabajo, pero sobre todo por su amistad en este camino de aprendizaje, ha sido un honor trabajar con él.

También agradezco a la **Facultad de Ciencias** por todas las enseñanzas, por permitirme formarme como parte de la Institución e impulsarme a crecer como profesionista y como ser humano.

A sus profesores, y maestras, al Dr. Raúl Balderas, al Dr. Marco A. Cárdenas Juárez, al Dr. Ulises Pineda Rico, la Mtra. María del Rosario Sandoval, por su tiempo, dedicación y apoyo. Y a todo su personal, por acompañarme hasta el final.

Por mostrarme que hay un universo por descubrir, que la ciencia transforma la vida de las personas y por todas las herramientas que me llevo para transformar el mundo y ser mejor persona.

Gracias por esta aventura, que después de tanto tiempo y esfuerzo ha llegado a su fin.

Descubrir lo creado, es crear la Ciencia.

GRACIAS FACULTAD DE CIENCIAS

GRACIAS UNIVERSIDAD AUTÓNOMA DE SAN LUIS POTOSI

Dedicatoria

A mi madre, por su gran esfuerzo para sacar adelante a la familia y enseñarme a hacer las cosas con amor. Gracias por ser tan fuerte.

A mi hermano, por ser un faro de luz, inspiración, mi fortaleza y motivación para salir adelante y lograr mis sueños. Gracias por tu paciencia.

A mi abuela Yolanda, por alimentar mi espíritu, mi corazón y mi estómago.
Gracias por tu Fe.

Esto es para ustedes.

A Mario Cepeda, Diana Ramírez, Melissa Berumen, Areli Flores y Ale Peña por ser los mejores amigos que han existido jamás, por su cariño y por caminar conmigo hasta el final.
Sin ustedes nada de esto hubiera sido posible.

A Ángel Gasca, Katia Hernández, Gerardo Zamudio, Jorge Pérez, Viri Hernández, Alfredo Covarrubias, Mayela Gonzales, Martín Guerreo, Coco, Andrés Rangel y Omar Flores, por su apoyo y amistad.

Y a todos los que directa e indirectamente me acompañaron en este camino.
Gracias por hacerlo realidad.

Resumen

En este proyecto de tesis se presenta el diseño, desarrollo e implementación de una unidad central de monitoreo y comunicación IoT para sistemas de Aspersión con Aplicación en Huertos Urbanos, que tiene como objetivo vincular, administrar y monitorear dispositivos periféricos de censado de humedad a tierra y humedad ambiente, mediante comunicación inalámbrica Bluetooth 4.0, así como el resguardo de la información mediante el motor de base de datos de memoria Redis, para su posterior análisis y acción en un sistema de riego automatizado conforme a las condiciones de los valores obtenidos en los huertos.

Los resultados obtenidos, muestran el funcionamiento correcto de la unidad central de comunicación y monitoreo, desarrollada en Python. Se demuestra la vinculación e intercambio de información con los dispositivos periféricos, así como el correcto almacenamiento de la información para su análisis.

Contenido

1 Introducción

1.1 Antecedentes

Estamos al comienzo de una revolución mayor a cualquiera que la humanidad haya experimentado. Una revolución tecnológica, social y cultural. La manera en que la tecnología transforma nuestra forma de vida y acciones, genera nuevos mecanismos de interacción y acción. Uno de los ejes principales que impulsa esta transformación tecnológica es la red mundial de objetos interconectados que se identifica bajo el concepto de Internet de las Cosas (IoT, Internet of Things).

A través de la interacción de elementos entre el mundo físico y digital, mediante el despliegue de sistemas de monitoreo, telecomunicaciones y procesamiento de datos, la tecnología ha adquirido un papel fundamental en el accionar de las naciones y los individuos. A nivel industrial, una cuarta revolución industrial llegó a cambiar los procedimientos en la industria y la manera de interactuar con las tecnologías. Y a nivel social, la tecnología nos brinda una nueva gamma de herramientas para hacer frente a las problemáticas actuales. Es el comienzo de una nueva era digital que abre un sin fin de posibilidades, campo verde para la exploración.

La tecnología, avanza a tal velocidad, que desarrollar hardware y software deberá suceder con la precisión necesaria para generar cambios significativos que establezcan el rumbo, ante un futuro cada vez más desafiante. Desarrollar tecnología para la innovación significa, no solo crear aparatos y sistemas, si no conectar a las personas y trabajar siempre desde el enfoque de mejora continua de la sociedad. Es un reto que existe de manera permanente, y hacerle frente con las herramientas adecuadas significará, la evolución social.

1.1.1 Internet de las Cosas

Según el Grupo de soluciones empresariales basadas en Internet (IBSG, Internet Business Solutions Group) de Cisco [1], Internet de las cosas es el punto en el tiempo en el que se conectaron a Internet más “cosas u objetos” que personas. Por otro lado, el diccionario de Oxford [2], define el término Internet de las Cosas de la siguiente manera:

Internet de las Cosas (sustantivo): Interconexión a través de Internet de dispositivos de computación integrados en objetos cotidianos, que les permite enviar y recibir datos.

El Internet de las cosas [3] considera un conjunto de herramientas que permite obtener información del entorno real, para llevar los datos al mundo digital y manipularles para realizar acciones específicas con ellos. Considera también, la interacción entre máquinas e individuos o máquinas entre si mismas.

El internet de las cosas, abre las puertas a la implementación de diversas tecnologías como la inteligencia artificial, la ciencia de datos, la simulación, la robótica, entre otras [4], considerando de forma imprescindible factores como la ciberseguridad. A nivel industrial, la Industria 4.0 considera al IoT como uno de los pilares fundamentales para la implementación de tecnologías de innovación, que permitirán a los diferentes sectores empresariales lograr sus objetivos en los próximos años.

Actualmente, el ecosistema tecnológico permite la implementación de estas tecnologías debido a la capacidad de procesamiento alcanzada, el nivel de conectividad en la red, las herramientas de comunicaciones exploradas en la actualidad que permiten generar soluciones cada vez más específicas y dedicadas para cada uno de los desarrollos, entre otros factores.

Estamos, sin lugar a dudas en un momento histórico preciso que nos brinda la oportunidad de implementar los resultados de la investigación de sistemas y dispositivos que lleva años formulándose y que actualmente pueden conectarse unos con otros para lograr resultados solo alguna vez imaginados.

Las nuevas tecnologías, nos permitirán desarrollar un potencial nunca antes visto y conectar los sistemas para analizar los datos y ejecutar acciones que influirán

directamente en el dia a día de los seres vivos en el planeta. Las nuevas generaciones se adaptan cada vez más rápido a estas herramientas y en algunos años, dejarán de ser productos innovadores para formar parte del día a día. En esas condiciones, el uso posible para cada una de las herramientas dependerá directamente de la creatividad y necesidades de cada una de los sectores a mejorar. En el campo de la domótica, por ejemplo, el internet de las cosas permitirá disminuir la cantidad de tareas domésticas realizadas por el ser humano, y centrar sus esfuerzos en tareas cada vez más sencillas. Y aunque actualmente el mercado se encuentra lleno de dispositivos interconectables, es solo el inicio.

En este documento, se hace énfasis en una de las áreas que definen directamente el rumbo de la población humana, la alimentación. Si bien, las técnicas en agricultura han evolucionado sin detenerse a lo largo de la historia debido a su importancia en la supervivencia de la especie, y han sufrido cambios importantes en cada una de las revoluciones por las que han sucedido las civilizaciones, es en este momento en el cual el internet de las cosas puede jugar un factor decisivo en su modo de operación. Obtener las características del ecosistema y analizarles para generar condiciones óptimas de manera automatizada, permitirá lograr mejores resultados y aprovechar de manera efectiva los recursos naturales cada vez más escasos para el ser humano. Nos encontramos entonces, ante herramientas que podrán definir el curso de la alimentación, y abrirán una gama de posibilidades por descubrir.

1.1.2 Bluetooth sobre Internet de las Cosas

Si bien, el protocolo de comunicación Bluetooth se ha posicionado en los últimos años como uno de los sistemas de comunicación más implementados en el desarrollo de las tecnologías inalámbricas cotidianas, su aplicación en el uso y creación de sistemas de internet de las cosas es cada vez más cuestionable, debido a las condiciones y limitantes que presenta. Sin embargo, precisamente por su implementación masiva en dispositivos de uso regular, la investigación de nuevas posibilidades para optimizar cada una de las características que presenta, le ha llevado a un nivel de presición bastante útil, obteniendo con el paso de los años una

serie de mejoras cuyas expectativas indican que seguirá siendo un elemento clave para el establecimiento de conexiones IoT en los próximos años.
Algunas de las funcionalidades del Bluetooth que en los últimos años le han hecho sobresalir, es la implementación de protocolos cada vez más precisos, que permiten desarrollar aplicaciones aprovechando a máximo las bondades que nos brinda. Otro de los puntos a considerar en la implementación de Bluetooth para IoT es, sin lugar a dudas, el rango de alcance. También, la tasa de transferencia de datos es un punto muy importante a considerar. Sin embargo, por las necesidades de los dispositivos IoT, un factor decisivo en la implementación suele ser el consumo de energía.

1.1.3 Bluetooth Low Energy

Afortunadamente, en el año de desarrollo de este proyecto, se encontraría vigente la versión 4.0 de Bluetooth, que mostraba una enorme ventaja ante sus versiones predecesoras. Bluetooth 4.0, es considerado un desarrollo que hace énfasis en su implementación para IoT, pues permite tasas de transferencia desde 25 Mbps hasta 32 Mbps, un rango de distancia de hasta 100m a la redonda. También puede conectarse de manera simultanea a diversos dispositivos y su nivel de consumo de energía se ha visto reducido notablemente. Bluetooth LE trabaja en la frecuencia de los 2.4 Ghz y para este proyecto ha sido seleccionado por las ventajas y facilidades que ofrece para el desarrollo del sistema de aspersión para huertos urbanos.

1.1.4 Huertos Urbanos

Se reconocen como huertos urbanos [5] a las áreas de cultivo ubicadas dentro de suelo urbano o urbanizable, que tienen como objetivo producir alimentos para consumo propio y tienen una finalidad social, educativa, de ocio, ambiental o paticipativa.
En la actualidad, la implementación de huertos urbanos permite a la sociedad generar sistemas de agricultura y cultivo autosostenibles en espacios reducidos,

optimizando los recursos y brindando la certeza de un consumo de alimentos con menor cantidad de agroquímicos, mejorando la calidad y el aporte nutricional de los alimentos.

1.2 Objetivos Generales y Específicos

El objetivo general de esta tesis es diseñar, desarrollar e implementar la unidad central de monitoreo como parte de un sistema IoT con implementación en huertos urbanos, con la finalidad de innovar los mecanismos y procedimientos que existen actualmente para el análisis, monitoreo, mantenimiento y riego en los huertos.

El sistema contempla el desarrollo e implementación de dispositivos para obtener información de las condiciones de los elementos y el ecosistema en el huerto, un mecanismo de comunicación y procesamiento de datos, con la capacidad de establecer una conexión y obtener información de los dispositivos, para posteriormente almacenar la información en una base de datos, a la cual podrá acceder el sistema de riego automatizado para evaluar los valores y encender el sistema de aspersión inteligente, así como un mecanismo de interacción con el usuario, innovando de esta manera el sistema manual que existe actualmente para el monitoreo y mantenimiento de los huertos.

Este proyecto, forma parte del proyecto realizado entre la empresa Bitwise Integrated Technologies SA de CV y la Universidad Autónoma de San Luis Potosí.

Para alcanzar el objetivo general de esta tesis se ha propuesto dos objetivos específicos:

1. Aprender las herramientas necesarias que faciliten la creación de software mediante el lenguaje de programación Python, aplicado en un entorno linux.

2. Desarrollar un Software e implementarle en un hardware con la capacidad de establecer un conexión e interacción con dispositivos periféricos IoT, almacenar los valores obtenidos para su posterior análisis en el sistema de riego, y mostrar los valores para el usuario.

1.3 Organización de la tesis

Este trabajo se ha organizado de la siguiente manera:

Capítulo 1. Introducción:

Se establece la definición de IoT y su evolución hasta nuestros días, el impacto social, cultural, y el estado del arte del desarrollo de tecnologías IoT aplicadas como soluciones tecnológicas a problemáticas sociales.

Capítulo 2. Diseño del Sistema

Se explica la arquitectura propuesta para el sistema IoT con aplicación en huertos urbanos, los componentes, elementos y protocolos de interacción entre las partes que lo componen. Se define la estructura y funcionamiento del sistema, así como las características de la unidad central de la unidad central de monitoreo y comunicación IoT como parte del sistema.

Se dan a conocer el entorno de desarrollo y las herramientas necesarias para el diseño y desarrollo de la propuesta, así como los requisitos necesarios para poder replicarse.

Capítulo 3. Implementación del Sistema

Explica el protocolo de implementación, procedimientos de aplicación y pruebas de funcionamiento de la unidad central de monitoreo y comunicación IoT.

Capítulo 4: Pruebas de Funcionamiento y Resultados

Muestra los resultados obtenidos en la implementación del sistema.

Capítulo 5: Conclusiones

Se detallan las conclusiones obtenidas con base en los resultados alcanzados. Se destaca su impacto con el objetivo general de este proyecto de tesis y los posibles trabajos futuros.

2 Diseño del Sistema

En este capítulo se explica la arquitectura propuesta para el sistema IoT con aplicación en huertos urbanos, los componentes, elementos y protocolos de interacción entre las partes que lo componen. Se define la estructura y funcionamiento del sistema, así como las características de la unidad central de la unidad central de monitoreo y comunicación IoT como parte del sistema.

Además, se dan a conocer el entorno de desarrollo y las herramientas necesarias para el diseño y desarrollo de la propuesta, así como los requisitos necesarios para poder replicarse.

2.1 Arquitectura del Sistema

El ecosistema inteligente, es un sistema IoT que consiste en el monitoreo de las condiciones de humedad y temperatura de un huerto, cuyos valores son enviados mediante un sistema de comunicación inalámbrica hasta una unidad central que recopilará los datos a lo largo del día y realizará el registro en una base de datos, que será posteriormente analizada por un sistema de riego inteligente, activado en el momento que los valores sobrepasen los límites establecidos. El sistema tendrá la posibilidad de conectarse a internet para enviar información, y aprovechar las herramientas de procesamiento y análisis de datos en la nube, así como implementar mecanismos de interacción con el usuario a través de diferentes plataformas y dispositivos.

En la Figura 2.1 se muestra la arquitectura general del sistema, dividido en capas según sus características y aplicaciones, así como la interacción entre las partes del sistema IoT.

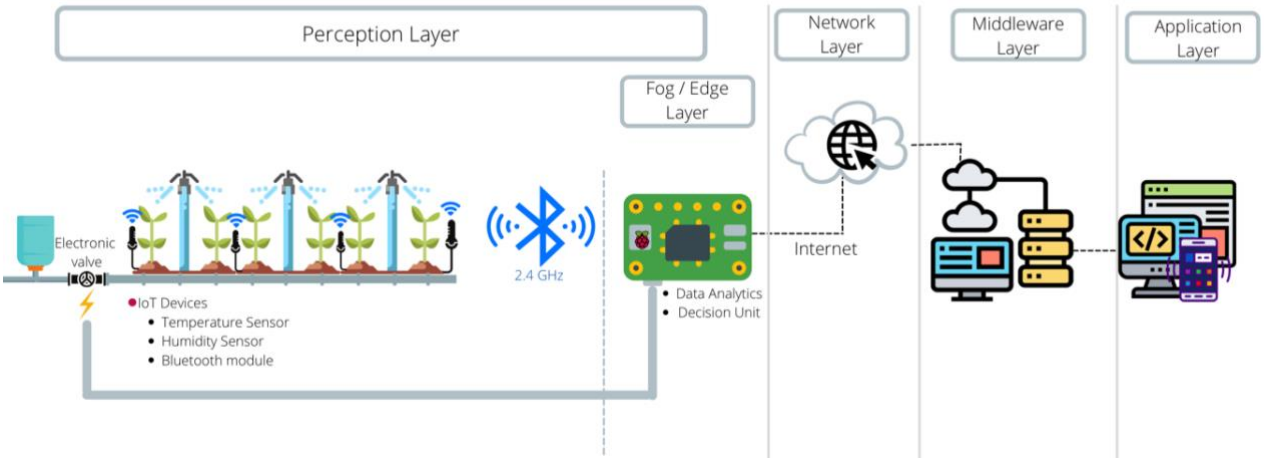

Figura 2.1 Arquitectura del Sistema IoT

2.1.1 Perception Layer

El sistema de monitoreo y riego de área local que interactúa con el entorno físico y conforma la capa de percepción, se compone de tres elementos que interactúan en tiempo real:

- **Estación Central.** Consiste en una unidad central de monitoreo y comunicación que realiza la búsqueda de sensores periféricos con conexión Bluetooth 4.0 para recibir información sobre las mediciones del entorno, y realizar el registro de los valores.
- **Dispositivos Periféricos IoT.** Consisten en sensores de humedad a tierra y temperatura distribuidos en un entorno de siembra controlado, que obtienen valores del entorno físico para posteriormente enviarlos a la unidad central, mediante conexión inalámbrica.
- **Sistema de Riego.** Consiste en un sistema de aspersión inteligente que tiene como finalidad la aplicación de agua en una zona determinada. La unidad central establece los parámetros y analiza los datos obtenidos de los sensores para abrir y cerrar la válvula electrónica, permitiendo el flujo de agua hacia el huerto.

Para ser considerado un sistema IoT, la arquitectura debe considerar tres secciones adicionales:

2.1.2 Network Layer

Consiste en la sección de transferencia de información y datos a internet. Se establece mediante la implementación de protocolos web y tecnologías de conexión.

2.1.3 Middleware Layer

Consiste en la sección de procesamiento y análisis de datos realizado en la nube, permite implementar tecnicas de procesamiento de datos de Big Data e Inteligencia Artificial, entre otras.

2.1.4 Application Layer

El mecanismo de interacción con el usuario, despliega la información en diferentes plataformas y dispositivos através de Interfaces de Usuario (UI) y experiencias de usuario (UX).

Establecer sistemas automatizados de monitoreo, comunicación y procesamiento para la implementación de actuadores en huertos urbanos, permite establecer condiciones de optimización de recursos y mejorar las condiciones de acción de aquellos lugares donde se implemente. Generar procesos de innovación para la agricultura, permite a los usuarios generar soluciones precisas aprovechando al máximo los recursos naturales.

En este proyecto de tesis, se desarrollará la capa de Percepción del sistema IoT, especificamente la unidad central de monitoreo y comunicación IoT, y su interacción con los dispositivos IoT y el sistema de riego.

2.2 Entorno de Desarrollo

Recursos de Hardware / Software

Implementar las herramientas adecuadas para la aplicación de un sistema IoT, que cubran las necesidades técnicas, optimicen los costos de desarrollo y cuenten con la información necesaria para aprovechar los recursos sin un alto nivel de especialización, es una tarea fundamental. Bajo estos argumentos, en este capítulo encontramos las herramientas de hardware y software seleccionadas para realizar la unidad central de comunicación IoT y las características que les convierten en la mejor opción.

2.2.1 Raspberry Pi Zero W

Como primer elemento de hardware, ha sido seleccionada la tarjeta programable Raspberry Pi [6] en su modelo Zero W. La Raspberry Pi Zero W es a la fecha la tarjeta más pequeña en el mercado dentro de la línea de Raspberry [7]. Es 40% más rápida que la original Raspberry Pi pero mide sólo 65 mm de largo por 30 mm de ancho y 5 mm de profundidad. Los conectores Mini de soporte ahorran espacio y el GPIO de 40 pines está vacío, lo que le da la flexibilidad de usar sólo las conexiones deseadas.

A comparación de la Raspberry Pi Zero común, el modelo “W” cuenta con la incorporación de las conexiones WiFi y Bluetooth Low Energy (BLE) 4.1 En la Figura 2.1 se muestra la apariencia física de la tarjeta Raspberry Pi Zero W.

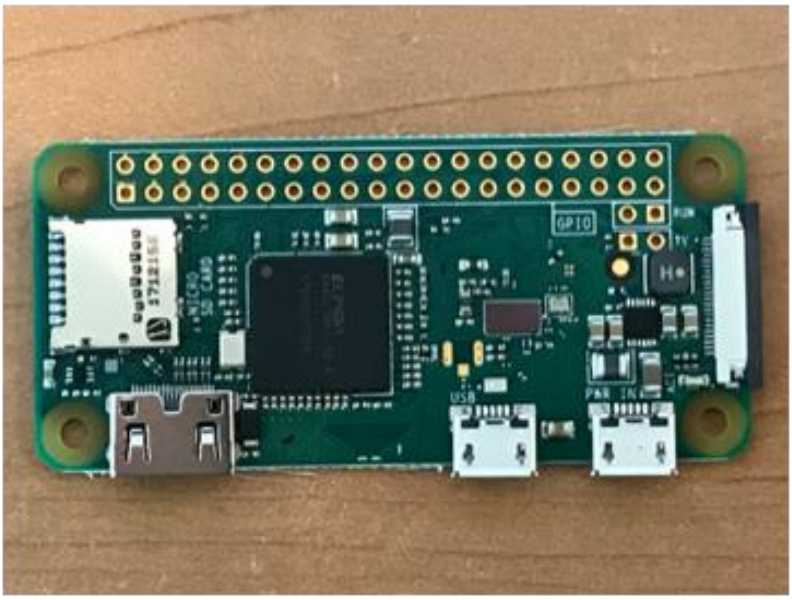

Figura 2.1 Placa Raspberry Pi modelo Zero W.

Especificaciones [8]

- Procesador Broadcom BCM2835 de un núcleo a 1 GHz
- Memoria RAM de 512MB
- Puertos Mini HDMI y US On-The-GO
- Entrada de alimentación Micro USB
- Conector 40-pin GPIO
- WiFi - IEEE 802.11 b/g/n WLAN (BCM43143)
- Bluetooth 4.1 (compatible con Bluetooth Low Energy – BLE)
- Dimensiones: 65mmx30mmx5mm
- Alimentación Micro USB 5V, 1.2A

En la Figura 2.2 se muestra un diagrama general de conexiones y puertos utilizados en la Placa de desarrollo Raspberry Pi Zero W:

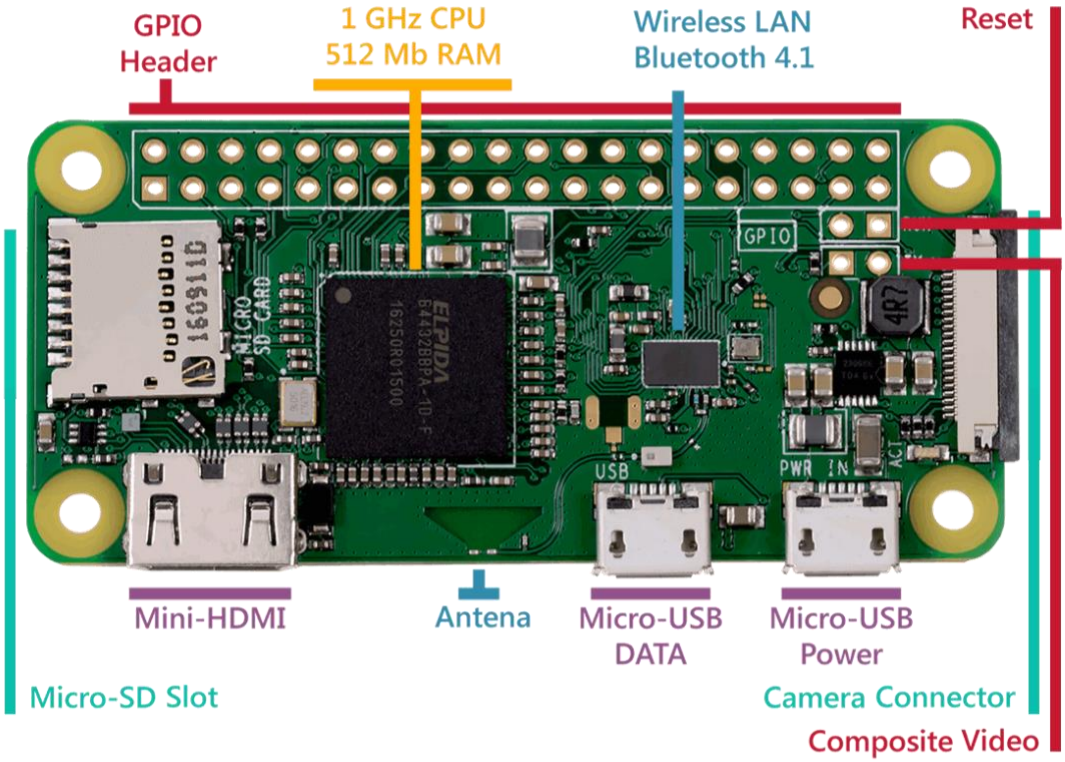

Figura 2.2 Diagrama de Uso de Raspberry Pi Zero W

2.2.2 Raspberry Pi OS

El sistema operativo seleccionado para este proyecto fue **Raspberry Pi OS** [9] (anteriormente llamado **Raspbian Os**), una distribución del sistema operativo GNU/ Linux basado en Debian [10], debido a su facilidad de implementación para trabajar mediante interfaz gráfica o línea de comandos, así como la basta cantidad de documentación y material de apoyo disponible para su aplicación. En la Figura 2.3 se muestra un ejemplo del entorno gráfico del sistema operativo. Actualmente, en el sitio oficial de Raspberry se encuentra disponible la versión 5.4 [11]

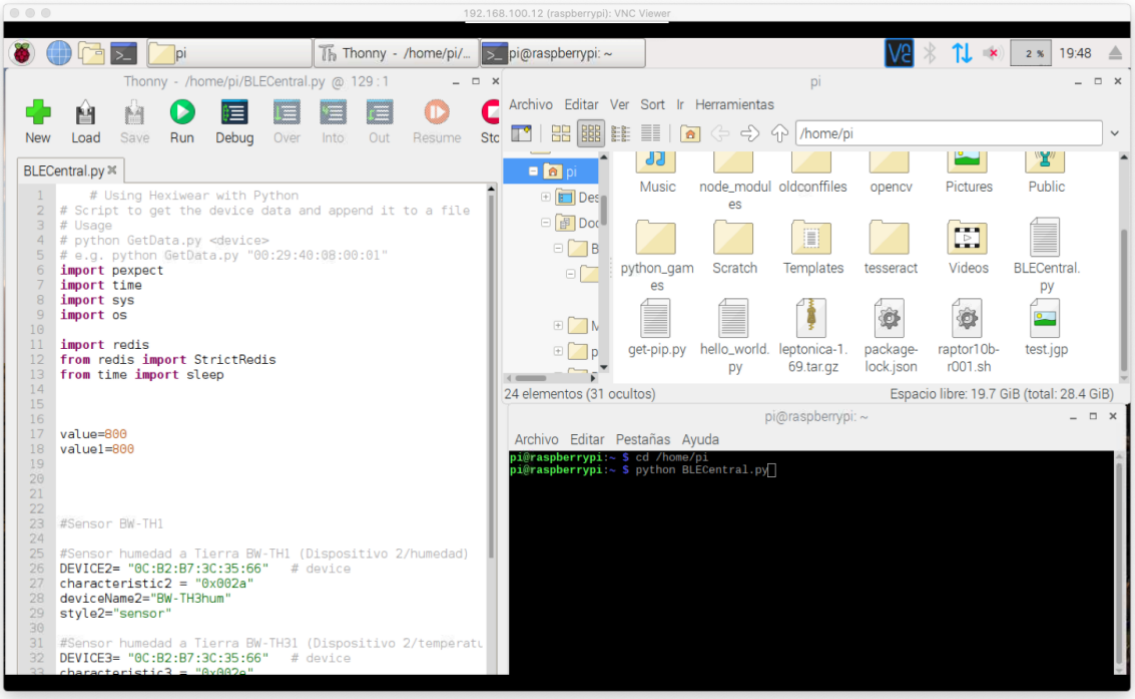

Figura 2.3 Sistema Operativo Raspbian con Interfaz Gáfica.

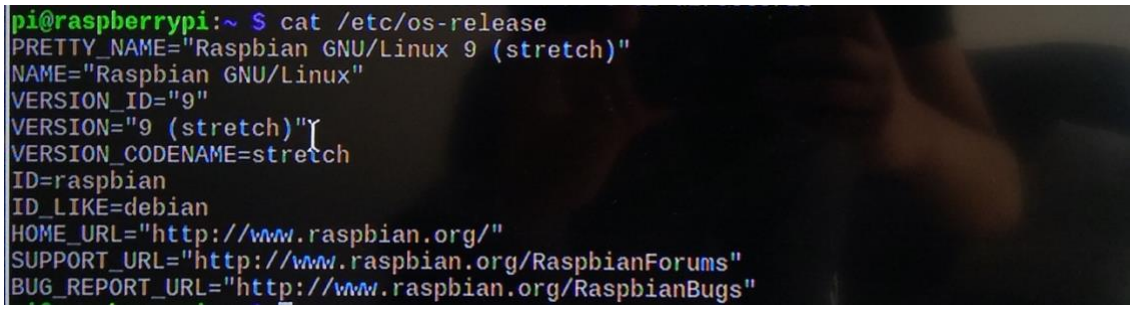

Figura 2.4 Sistema Operativo desplegado en Terminal de Raspberry.

En la Figura 2.4 se muestra la versión implentada en el sistema, basada en la versión de Debian 9 (stretch). Una vez descargado Raspbian se instala, o se copia a la tarjeta de memoria SD (se debe formatear de forma que sea booteable):

- En Windows se puede usar el programa **Win32 Disk Imager**.
- En iOS se puede usar la aplicación **Disk Utility** que se incluye en la Mac.
- En Linux, el proceso se debe poder realizar sin problemas con **Startup Disk Creator** o similares.

Una vez que el procedimiento anterior se realiza, simplemente se debe introducir la tarjeta e iniciar el sistema en la Raspberry. Posteriormente se debe ejecutar el comando *raspi-config* con permisos de *sudo* es decir:

```
sudo raspi-config
```

Este comando desplegará un menú en consola con distintas opciones, donde se podrá cambiar la contraseña, configurar el layout o distribución del teclado a español, la zona horaria etc.

Raspbian de GNU destaca ante otros sistemas debido a la poca cantidad de recursos con la que es capaz de desplegarse en la tarjeta Raspberry Pi Zero W, así como la capacidad de utilizar una gran cantidad de herramientas desarrolladas para Debian. Además, es una versión de software libre que permite disminuir los costos de desarrollo e implementar soluciones efectivas de una manera sencilla, aprovechando las bondades que la interfaz de usuario y el CLI permiten.

2.2.3 Python

Python [12] es uno de los lenguajes de programación más utilizados alrededor del mundo en la actualidad, debido a la facilidad de implementación y a la gran cantidad de recursos y librerías existentes para implemetar tareas específicas. Debido a su estructura, permite desarrollar desde pequeños script hasta sistemas complejos hospedados en la web. Algunas de las características que le destacan son la implementación como un lenguaje interpretado, lo que influye directamente en su característica multiplataforma , facilitando el desarrollo de pruebas y aplicación en diferentes sistemas operativos. Además, su tipado dinámico facilita la escritura del lenguaje y nos permite enfocarnos directamente en la creación de soluciones. La enorme comunidad que existe alrededor del mundo, da soporte al lenguaje y abre las posibilidades a analizar sistemas existentes que aporten al desarrollo de nuevas soluciones e implementeaciones [13].

Las placas de desarrollo Raspberry, permiten la implementación de diversos lenguajes de programación tales como Java, Ruby, Perl, entre otros. Para el presente proyecto se acordó con la empresa Bitwise Integrated Technologies SA de CV trabajar con el lenguaje de programación ***Python*** por su sencillez e interacción con los diferentes módulos de la tarjeta Raspberry Zero W, además de su compatibilidad entre los grupos de trabajo que colaboran para el desarrollo de este proyecto.

Para su instalación se requiere hacer los siguientes pasos en la terminal o consola del sistema:

```
sudo apt-get install python-pip
```

Python 3:

```
sudo apt-get install python3-pip
```

2.3 Librerías

Para la realización del proyecto fue necesario utilizar las siguientes librerías de Python:

- **pexpect 4.6**: Permite el control de elementos nativos del sistema.
- **redis 3.0.1**: Permite crear estructuras de datos en memoria rápido y de código abierto.

A continuación se hace una breve descripción de las librerías mencionadas.

2.3.1 Pexpect

Pexpect [14] esta diseñada para ejecutar un programa e interactuar con él por línea de comandos, principalmente usado para SSH, FTP, mencoder, passwd, etc. Pexpect, está diseñada para permitir generar una aplicación secundaria y que sea controlada mediante línea de comandos a través del script, logrando con ello interactuar directamente con módulos de control de hardware a bajo nivel.

Pexpect se encuentra dentro de **The Phyhon Package Index** o mejor conocido como **PyPi** (https://pypi.org/), que es un repositorio completamente formado por programas de *Phython*; para instalarlo se debe ejecutar en la terminal los siguientes comandos:

```
sudo apt-get install phython-pip
```

Después, se realiza la descarga Pexpect del repositorio ejecutando en la terminal:

```
pip install pexpect
```

Para consultar la información completa de dicha librería se puede acceder a los siguientes enlaces:

- https://pexpect.readthedocs.io/en/stable/install.html
- https://github.com/pexpect/pexpect

Para este sistema, seleccionamos pexpect por la facilidad y eficacia para implementar los controladores de Bluetooth 4.0 desde la tarjeta.

2.3.2 Redis

Redis [15] incorpora un conjunto de estructuras de datos en memoria versátiles que permite crear con facilidad diversas aplicaciones personalizadas. Redis significa *Remote Dictionary Server* y es un almacén rápido de datos clave-valor en memoria de código abierto que puede implementarse como base de datos, caché, agente de mensajes y cola. Redis ofrece tiempos de respuesta inferiores al milisegundo, permitiendo una gran cantidad de solicitudes por segundo.

En Redis, todos los valores residen en la memoria, haciendo una diferencia notable con las bases de datos que residen en los discos o las SSD. Además, Redis es un proyeto de código abierto [16], que cuenta con el apoyo de una comunidad activa, facilitando el uso e implementación de nuevas herramientas.

Para instalar esta potente base de datos en la Raspberry Pi Zero W, se ejecuta el siguiente comando en la terminal:

```
sudo apt-get install redis-server
```

Para consultar la información completa de instalación se puede acceder al siguientes enlace:

- https://pypi.org/project/redis/

3 Implementación del Sistema

3.1 Diagramas UML

El diseño e implementación de los diagramas UML (lenguaje unificado de modelado) aplicados a este proyecto, permite establecer los protocolos de interacción entre los elementos de hardware y software para desarrollarse de manera independiente por cada una de los equipos de desarrollo, estableciendo estándares y mecanismos de colaboración efectivos.

3.1.1 Diagrama de secuencia

El diagrama de secuencia en la Figura 3.1 muestra la actividad de los elementos del sistema a lo lagro del tiempo, representando la comunicación entre los objetos y el flujo de la información. La interacción entre los dispositivos comienza cuando la unidad central establece la búsqueda de dispositivos periféricos mediante Bluetooth, para posteriormente solicitar la información almacenada que ha sido obtenida directamente de los sensores. Después, la información obtenida es almacenada en una unidad de base de datos. Los datos, son analizados através de algorítmos de procesamiento, que generan indicadores para el encendido automatizado del sistema de riego, tal como se muestra en la Figura 3.1.

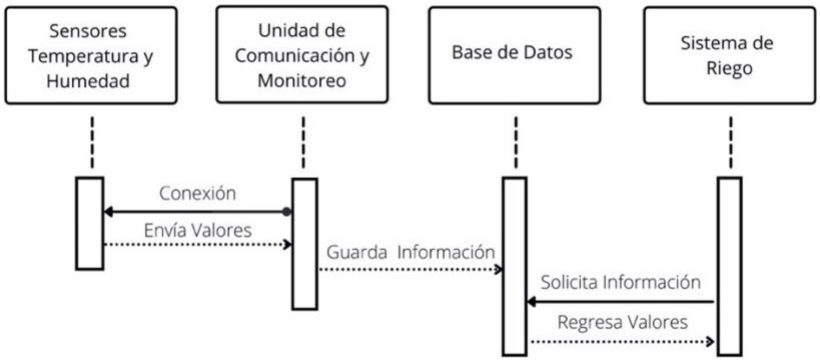

Figura 3.1 Diagrama de secuencias para el Sistema de Aspersión IoT.

3.2 Desarrollo del Sistema

La capa de percepción del sistema IoT con aplicación en huertos urbanos, que se muestra en la Figura 3.2, se conforma por cuatro elementos fundamentales, que consisten en:

- **Sensores de Temperatura y Humedad**. Los sensores son los dispositivos que permiten al sistema capturar los datos del entorno físico y enviar su información mediante un sistema de comunicación.
- **Conectividad Bluetooth 4.0.** En un sistema IoT es imprescindible establecer mecanismos de transferencia de información que permitan establecer una red local para la interacción entre los elementos. En esta capa del sistema, la implementación de comunicación Bluetooth LE permite la interacción entre los sensores y la unidad central de comunicación.
- **Unidad Central de Comunicación y Monitoreo.** Esta unidad fue desarrollada con la finalidad de establecer un eje central de búsqueda de dispositivos periféricos para recibir la información generada y tener la capacidad almacenarla. Es conocida como Fog Layer.
- **Sistema de Riego Automatizado.** Es implementado para analizar la información almacenada, procesarla, establecer las condiciones y acciones para iniciar el riego en zonas dedicadas.

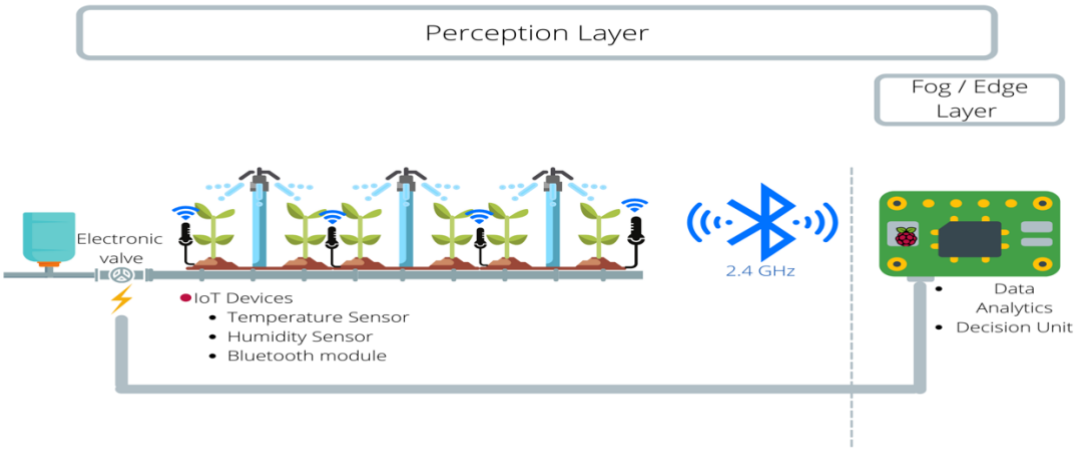

Figura 3.2 Diagrama de la capa de Percepción del Sistema.

3.2.1 Unidad Central de Monitoreo y Comunicación

La unidad central de Monitoreo y comunicación tiene como función estratégica descubrir los dispositivos periféricos de monitoreo que existen en el entorno, obteniendo directamente el último valor registrado en sus memorias y su nivel de batería para posteriormente, agregarlo en una base de datos.

El software desarrollado para iniciar la actividad de monitoreo e interacción con los dispositivos periféricos y almacenamiento de datos, es hospedado en los archivos de arranque del sistema operativo, de tal manera que será ejecutado automaticamente al iniciar el sistema y se mantendrá en funcionamiento hasta el momento que sea detenido por el usuario. En caso de suceder un error en el programa, este se reiniciará de forma automática, convirtiéndose un programa de ejecución plug and play.

En la Figura 3.3 se muestra la implementación de la unidad central de monitoreo y comunicación en la tarjeta de desarrollo Raspberry montada en el circuito para interactuar con el sistema de riego.

Figura 3.3 Ilustración de la Unidad de Monitoreo y Comunicación

3.2.2 Dispositivos Periféricos

Como parte de las especificaciones generales de hardware del sistema inteligente de aspersión, se considera la incorporación de capacidades inalámbricas integradas. En particular, se incluye el protocolo Bluetooth LE para crear enlaces inalámbricos dentro de una área local con dispositivos periféricos (sensores y actuadores).

Para el desarrollo de la etapa de comunicación Bluetooth en el sistema de aspersión se diseñó y construyó un módulo con comunicación Bluetooth LE. En la Figura 3.4 se muestra la apariencia física de la tarjeta de comunicación Bluetooth LE de diseño propio basada en el microcontrolador MCU CC2541 de la familia Texas instruments.

Figura 3.4 Módulo sensor BWTH1 con comunicación Bluetooth LE.

A continuación se describen las características generales tanto del microcontrolador MCU CC2541 como de los sensores Si7021 (Temperatura y Humedad) y FC28 (Temperatura en el suelo).

3.2.2.1 Microcontrolador MCU CC2541

El microcontrolador principal es un sistema en chip (SoC) de 8 bits con radio BLE 4.0 integrado de la familia Texas instruments [17].En la Figura 3.5 se muestra un modelo a escala del Microcontrolador de referencia.

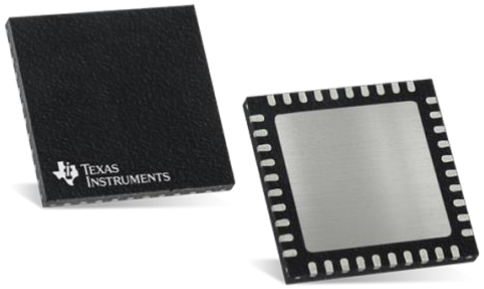

Figura 3.5 Microcontrolador MCU CC2541

El Sistema en Chip (SoC) Bluetooth CC2541 2.4 Ghz de Texas Instrument, permite optimizar la energía para las aplicaciones de Bluetooth 4.0 de manera considerable. Este dispositivo de Texas Instrument, combina el rendimiento de radiofrecuencia con un MCU 8051, una memoria flash programabre y suficiente RAM. Es ideal para sistemas de bajo consumo.

Especificaciones [17]

RF

- Sistema en chip compatible de baja energia de 2.4 GHz.
- Admite velocidades de datos de 250kbps, 1Mbps y 2 Mbps
- Sensibilidad del receptor (-94dBm a 1 Mbps), selectividad y rendimiento de bloqueo.

Microcontrolador

- Flash Programable en el sistema, 128KB o 256KB
- 8 KB de RAM con retención en todos los modos de energía
- Soporte para depuración de Hardware

3.2.2.2 Sensor de Temperatura y Humedad Si7021

En la Figura 3.6 se muestra el sensor de temperatura y humedad si7021 que está basado en una tarjeta de evaluación de la marca Adafruit con el chip si7021 [18] del fabricante Silicon Labs.

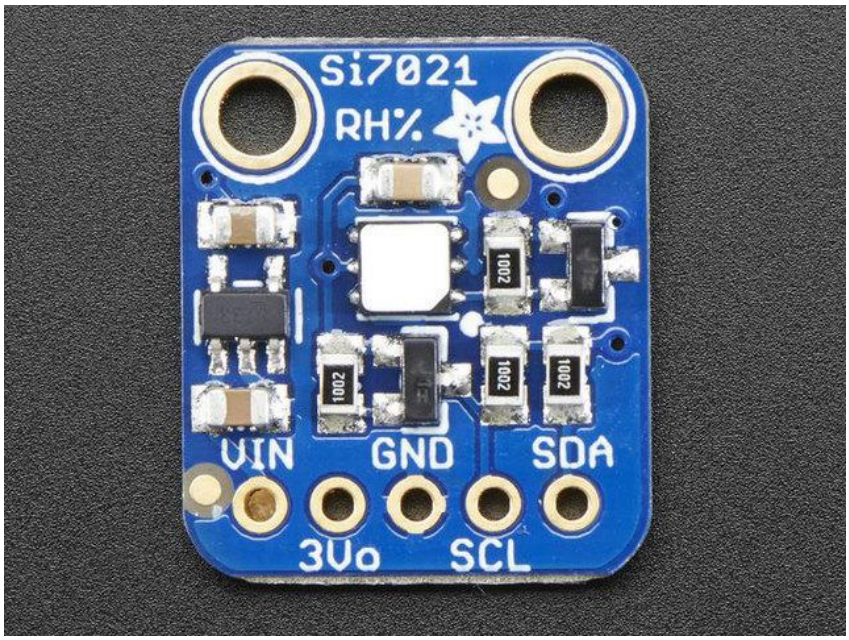

Figura 3.6 Sensor de Temperatura y Humedad Si7021.

Este sensor de Silicon Labs, tiene mediciones de humedad relativa de ± 3% con un rango de 0 a 80% de humedad relativa y una presición de temperatura de ± 0.4 ºC en un rango de -10 a +85º C. Utiliza I2C para la transferencia de datos. En la figura 3.7 podemos ver el diseño del circuito del sensor Si7021 [19].

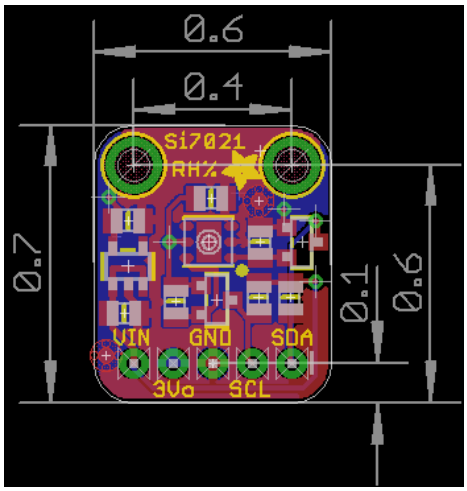

Figura 3.7 Impresión de Fabricación Version Adafruit Si7021

3.2.2.3 Sensor de Temperatura en Suelo FC28

El higrómetro FC-28 es un sensor que mide la humedad del suelo, ver Figura 3.8. Este tipo de sensor es ampliamente empleado en sistemas automáticos de riego para detectar cuando es necesario activar el sistema de bombeo. El FC-28 es un sensor sencillo que mide la humedad del suelo por la variación de su conductividad.

Figura 3.8 Sensor de Temperatura y Humedad Si7021.

En este sistema, el valor obtenido por el sensor analógico es almacenado en un registro local para su posterior obtención mediante lectura de parámetros através de protocolos Bluetootth 4.0 [20]. En la Figura 3.9 se muestra la conexión entre el sensor de temperatura y su controlador digital.

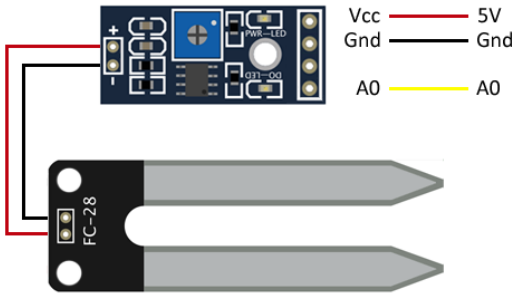

Figura 3.9 Diagrama de Sensor de Temperatura y Humedad SI7021

3.2.3 Controlador Digital para Sistemas de Aspersión con Aplicación en Huertos Urbanos

La tarjeta lógica consiste de un sistema embebido basado en la tarjeta programable Raspberry Pi en su modelo Zero W, cuya función principal es el control de arranque y frenado de un motor de 1⁄2 Hp, para un sistema de aspersión inteligente con aplicación en huertos urbanos, ver Figura 3.10.

Figura 3.10 Controlador Digital para sistema de Aspersión

En la tarjeta lógica se han añadido los siguientes periféricos para una mayor autonomía y funcionalidad: ADC externo, RTC y LED Driver. Además se incluye una unidad central de comunicación con el protocolo Bluetooth LE para crear enlaces inalámbricos dentro de una área local con dispositivos periféricos (sensores y actuadores). Bajo esta arquitectura, la unidad central de comunicación y el controlador digital para sistemas de aspersión se encuentran hospedados en la misma tarjeta Raspberry.

En este sistema, el controlador digital analiza directamente los valores almacenados en la base de datos Redis, para su análisis con parámetros establecidos para toma de desiciones. Solo entonces, activará el sistema de riego inteligente, optimizando recursos e implementando el total de las partes del sistema.

4 Pruebas de Funcionamiento y Resultados

En este capítulo, se lleva a cabo ~~desarrollan~~ la implementación de dos técnicas de conectividad e intercambio de información entre dispositivos IoT y la unidad central de monitoreo mediante comunicación Bluetooth 4.0

En la primera sección, se desarrolla la conexión Bluetooth desde consola, donde através de comandos implementados en terminal del sistema, se establece la búsqueda y conexión con los dispositivos, y se interactúa con los servicios y características para obtener los valores almacenados.

En la segunda sección, se desarrolla la comunicación Bluetooth mediante un script en python, donde se hace uso de las librerías y protocolos mencionados anteriormente.

Posteriorente, se desarrollan los procedimientos para registrar los valores obtenidos en la base de datos Redis, y la implementación del software como un proceso autónomo de arranque automatizado mediante un daemon.

Para finalizar, se muestran los resultados de la implementación de las técnicas en pruebas aplicadas al sistema.

4.1 Conexión Bluetooth mediante Terminal

Esta sección muestra la forma de efectuar un emparejamiento directo entre la Raspberry Pi Zero W y un dispositivo vía Bluetooth periférico. Para esto se usará la interfaz *bluetoothctl* mediante línea de comandos o terminal.

Para conectar el dispositivo se ejecuta sl siguiente comando en el script:

```
bluetoothctl
```

que inicializará la interacción del usuario con el control bluetooth de CLI, desplegando:

[bluetooth]#

A continuación, podrán ejecutarse el comando

[bluetooth]#help

para obtener una lista con los comandos disponibles para realizar una conexión mediante bluetooth 4.0 desde terminal, como se ilustra a continuación en la Figura 3.11:

Figura 3.11 Lista de comandos disponibles para la comunicación Bluetooth mediante terminal.

Para iniciar la exploración o escaneo de dispositivos Bluetooth es necesario asegurar que el Bluetooth del dispositivo de destino esté encendido y sea detectable, ejecutando el comando:

[bluetooth]#scan on

Será necesario esperar unos algunos segundos para completar el escaneo, los resultados de los dispositivos detectados se desplegarán en un listado que contendrá valores como Dirección MAC, Nombre del Dispositivo, UUIDs y RSSI entre otros, según las características del dispositivo, tal como se muestra a en la Figura 3.12.

```
[bluetooth]# scan on
Discovery started
[CHG] Controller B8:27:EB:E6:30:0B Discovering: yes
[NEW] Device 5A:E9:9E:2E:E3:1E 5A-E9-9E-2E-E3-1E
[NEW] Device 6E:9E:22:16:C9:17 6E-9E-22-16-C9-17
[CHG] Device 0C:B2:B7:3C:37:CB RSSI: -67
[CHG] Device 0C:B2:B7:3C:37:CB UUIDs: 00001803-0000-1000-8000-00805f9b34fb
[CHG] Device 0C:B2:B7:3C:37:CB UUIDs: 00001802-0000-1000-8000-00805f9b34fb
[CHG] Device 0C:B2:B7:3C:37:CB UUIDs: 00001804-0000-1000-8000-00805f9b34fb
[NEW] Device D8:9E:3F:7B:7C:78 iPhone de Adro
[CHG] Device 5A:E9:9E:2E:E3:1E RSSI: -67
[CHG] Device 6E:9E:22:16:C9:17 RSSI: -92
[CHG] Device 5A:E9:9E:2E:E3:1E RSSI: -79
[CHG] Device 6E:9E:22:16:C9:17 RSSI: -79
```

Figura 3.12 Lista de dispositivos Bluetooth detectado por la Raspberry Pi Zero W.

En este punto, es importante copiar la dirección MAC del dispositivo periférico que desea vincularse, para su uso posterior en la conexión.

Para establecer la conexión y emparejamiento del dispositivo Bluetooth se ejecuta el comando *connect*, seguido por la MAC del dispositivo como se muestra a continuación:

```
[bluetooth]#connect [MAC]
```

Un ejemplo del procedimiento de conexión bluetooth de la Raspberry Zero W con el dispositivo con MAC OC:B2:B7:3C:37:CB (Sensor de Temperatura y Humedad BW-TH2) es la que se muestra a continuación:

PASO 1: ESCANEO DE DISPOSITIVOS

```
[bluetooth]#scan on
```

PASO 2: CONEXIÓN Y EMPAREJAMIENTO DEL DISPOSITIVO

```
[BW-TH2]#connect 0C:B2:B7:3C:37:CB Attempting to connect to
0C:B2:B7:3C:37:CB

[CHG] Device 0C:B2:B7:3C:37:CB Connected: yes
Connection successful
[CHG] Device 0C:B2:B7:3C:37:CB ServicesResolved: yes
[CHG] Device 0C:B2:B7:3C:37:CB Name: BW-TH1
[CHG] Device 0C:B2:B7:3C:37:CB Alias: BW-TH1
[CHG] Device 0C:B2:B7:3C:37:CB Modalias: bluetooth:v000Dp0000d0110
```

PASO 3: IDENTIFICAR LA LISTA DE ATRIBUTOS DEL SENSOR (SERVICIOS Y CARACTERÍSTICAS)

Para mostrar los servicios y características Bluetooth del dispositivo BW-TH2 se ejecuta:

```
[BW-TH2]#list-attributes [NEW] Primary Service

/org/bluez/hci0/dev_0C_B2_B7_3C_37_CB/service0028 0000fff0-0000-1000-8000-
00805f9b34fb
Unknown

[NEW] Characteristic
/org/bluez/hci0/dev_0C_B2_B7_3C_37_CB/service0028/char0029 0000fff1-0000-
1000-8000-00805f9b34fb
Unknown
Characteristic User Description

[CHG] Device 0C:B2:B7:3C:37:CB UUIDs: 0000fff0-0000-1000-8000-
00805f9b34fb

[CHG] Device [CHG] Device [CHG] Device [CHG] Device

0C:B2:B7:3C:37:CB ServicesResolved: yes 0C:B2:B7:3C:37:CB Name: BW-TH1
0C:B2:B7:3C:37:CB Alias: BW-TH1
0C:B2:B7:3C:37:CB Modalias: bluetooth:v000Dp0000d0110
```

PASO 4: SELECCIONAR LA CARACTERÍSTICA

Para seleccionar una característica en el dispositivo se ejecuta, mediante el comando *select-attribute* y agregando el servicio seleccionado:

```
[BW-TH2]#select-attribute
/org/bluez/hci0/dev_0C_B2_B7_3C_37_CB/service0028/char0029
```

El indicador de lade comandos se transforma de la siguiente manera:

```
[BW-TH1:/service0028/char0029] #
```

PASO 5: LECTURA DEL VALOR OBTENIDO DE LA CARACTERÍSTICA

Finalmente se obtiene la lectura del valor de la característica ejecutando el siguiente comando *read*

```
[BW-TH1:/service0028/char0029]#read [CHG] Attribute

/org/bluez/hci0/dev_0C_B2_B7_3C_37_CB/service0028/char0029 Value: 0x02
```

4.1.1 Conexión Bluetooth mediante Script en Python

A continuación se describe el programa ***BTCentral.py*** desarrollado en Python para realizar la conexión Bluetooth entre la Raspberry Zero W y un dispositivo periférico.

El programa *BTCentral.py* ejecuta las siguientes 2 funciones:

1. Establecer la conexión Bluetooth con los dispositivos periféricos, esto por medio del reconocimiento de su MAC.
2. Registrar los valores obtenidos de los dispositivos IoT conectados, en la base de datos **Redis**, especificando el nombre del dispositivo y el valor recibido del sensor. Para realizar el registro en Redis, el valor recibido del sensor se transforma a porcentaje, facilitando su uso.

Para ejecutar el programa *BTCentral.py* se puede realizar desde el directorio raíz mediante el comando:

```
python BTCentral.py
```

A continuación se explica brevemente el código del programa *BTCentral.py* realizado en Python versión 1.0.

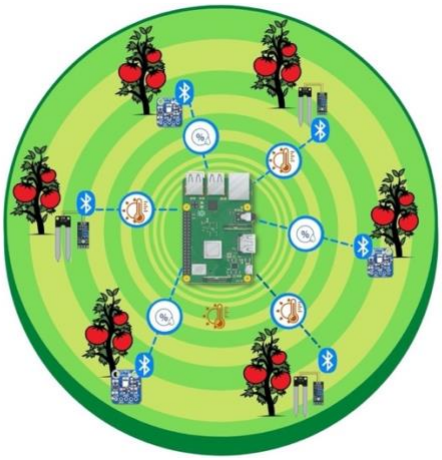

Figura 3.13 Representación Visual de Interacción entre Dispositivos periféricos y Estación Central.

4.1.2 Declaración de Dispositivos a escanear

En la primera parte del código, se define la información requerida para cada uno de los sensores de interés. Tomando como ejemplo el sensor de Humedad del suelo (BW-TH1), se puede observar la definición de su MAC (OC:B2:B7:3C:37:CB), su valor de característica (0x002a), nombre de identificación (BW-TH1hum) y tipo de variable a leer (en este caso el valor de lectura del sensor). En la Tabla 3.1 se describe el formato usado para definir los nombres de los dispositivos a conectar.

Tabla 3.1 Formato de identificación para los dispositivos de comunicación vía Bluetooth.

Dispositivo	Número del Dispositivo	Variable a leer	Identificación
Nombre	#	Tipo	Nombre#Tipo
BW-TH	1	hum	BW-TH1hum

4.1.3 Conexión y emparejamiento del dispositivo

El segundo bloque de código implementa el proceso de escaneo y conexión con los dispositivos, así como las acciones correspondientes en caso de que la conexión no sea exitosa. Los valores obtenidos del sensor serán enviados a la función dataToRedis(), de lo contrario se enviará el valor preestablecido *800* para indicar que la conexión no fue exitosa.

4.1.4 Registro en Redis

El bloque de código mostrado a continuación tiene como propósito realizar el registro de los valores provenientes de los sensores en la base de datos Redis. La variable ***r*** establece la conexión al puerto solicitado (6379) y la database 0. Si el valor que obtenido proviene de un sensor, se hace la conversión a valores de porcentaje para guardarlo en la base de redis. En caso de obtenerse desde la información de la batería, el registro se hace sin conversión. Si no se tiene una conexión bluetooth, los resultados registrados será negativos, específicamente se registrará el valor [-2.13].

Los valores que regresa la función se obtienen directamente de los valores registrados en Redis, con respecto al nombre ingresado. La función se muestra en la Figura 3.14.

```
def data_toRedis(value, value2,namedevice,style):
    # step 3: create the Redis Connection object
    try:


        r = redis.StrictRedis(host='localhost', port=6379, db=0)
        #print ("Porcentaje de Humedad registrado en Redis:")


        if style is "sensor":
            intmsg =int((float(value)))
            previo=intmsg/float(255)
            #previo=36/float(255)
            final=1-float(previo)

        else:
            if value==800:
                final=-2.13
            else:
                final=value


        r.set(namedevice,final)
        msg= r.get(namedevice)
        print(namedevice)
        print(msg)


    except Exception as e:
        print(e)
```

Figura 3.14 Fragmento de Código BTCentral.py sobre Almacenamiento en Redis

4.1.5 Lectura de sensores permanente

En la parte principal de ejecución denominada Main, se realiza el llamado a la función de Lectura de Bluetooth, estableciéndose de manera independiente por dispositivo, para una fácil manipulación. La función que forma parte del código, se muestra en la Figura 3.15:

```
try:
    while True:

        void(DEVICE2,characteristic2,deviceName2,style2);
        void(DEVICE3,characteristic3,deviceName3,style3);
        void(DEVICE4,characteristic4,deviceName4,style4);
        sleep(30)
except KeyboardInterrupt, e:
    logging.info("Stopping...")
```

Figura 3.15 Fragmento de Código BTCentral.py sobre Conenexión de Dispositivos Bluetooth.

Al llamar la función *void*, se ingresan los parámetros establecidos en el punto 1 (*Declaración de dispositivos*). Para detener la ejecución del programa, bastará utilizar *ctrl+c.*

4.2 Daemon

Un *demonio* [21], en Linux, es un proceso que se ejecuta en segundo plano y es autónomo, de manera que no necesita interacción por parte de un usuario del sistema para arrancar y funcionar. Son también conocidos con el nombre de servicios. La implantación del script como un servicio permanente, se realiza con el objetivo de tener una ejecución continua del escaneo de los sensores desde el arranque del sistema.

La Figura 3.16, muestra la estructura básica de la arquitectura del sistema donde se implementa el daemon con respecto al espacio en el disco de almacenamiento del sistema.

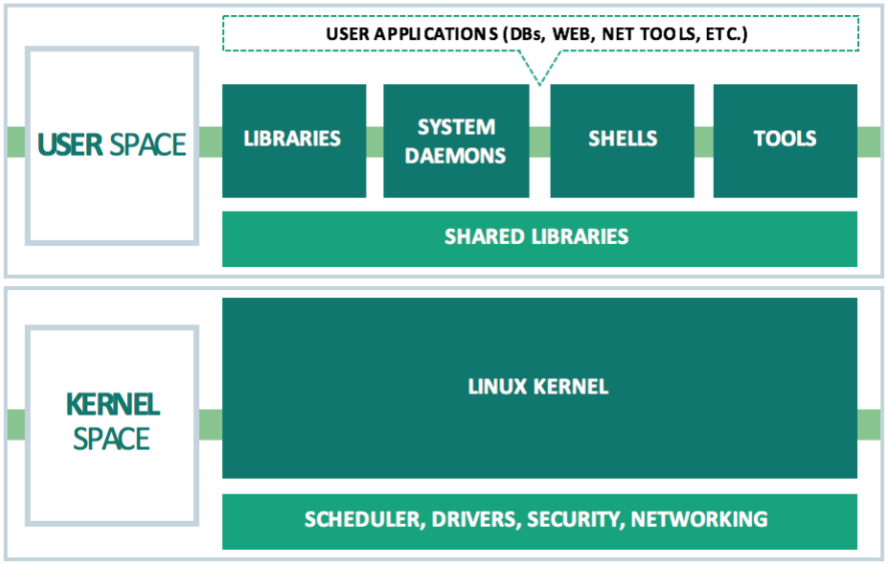

Figura 3.16 Arquitectura de la implementación de un Daemon en Linux

Para convertir el script en un servicio que podrá ser implementado en los procesos de arranque de la Central, se realiza el siguiente manera:

El script BTCentral.py en su versión final, es almacenado en la carpeta de raíz

home (home/pi/).

Después, se procederá a implementarle como servicio.

Activar modo *superusuario*: `su -`

Acceder a la librería system:

```
cd /lib/systemd/system/
```

para posteriormente, crear el servicio **CentralBTserv.service :**

```
sudo nano CentralBTserv.service
```

Dentro del servicio, se establecerán las funciones necesarias para su ejecución. La Figura 3.17 muestra la carpeta en el sistema donde fue almacenado el daemon para su correcto funcionamiento e implementación:

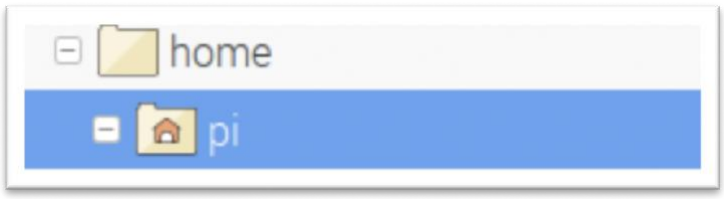

Figura 3.17 Carpeta de almacenamiento de Daemon

Las características del servicio deberán establecerse como se muestra a continuación:

Description=Central de Dispositivos Bluetooth After=multi-user.target

[Service]
Type=simple
ExecStart=/usr/bin/python /home/pi/BTCentral.py Restart=always

[Install] WantedBy=multi-user.target

El servicio es sido creado. Para activar el servicio, se implementarán los comandos:

```
sudo chmod 644 /lib/systemd/system/CentralBTserv.service chmod +x
/home/pi/BTCentral.py
sudo systemctl daemon-reload
sudo systemctl enable CenterBTserv.service

sudo systemctl start CentralBTserv.sev
```

El proceso ha sido activado. Para revisar el estado del servicio, se podrá utilizar el comando: `Sudo systemctl status CentralBTserv.service`

Para activar/ detener el servicio, podrán utilizarse los siguientes comandos:

Iniciar Servicio: `Sudo systemctl start hello.service`

Detener Servicio: `Sudo systemctl stop hello.service`

4.3 Resultados

Para validar el funcionamiento y operación de la unidad central de monitoreo se efectuaron dos tipos de pruebas, de conectividad y de disponibilidad. Para estas pruebas se consideraron el uso de los sensores Si7021 (Temperatura y Humedad) y FC28 (Temperatura en el suelo) dentro un espacio de laboratorio con escenarios controlados.

A continuación se hace una descripción de las funciones necesarias para la realización de dichas pruebas:

De los **dispositivos periféricos**:

- El **modo lectura** permite obtener los valores de temperatura y humedad a través de los sensores Si7021 y FC28
- El **modo almacenamiento** guarda los valores en la memoria local dentro de los dispositivos IoT.
- El **modo detectable** permite el emparejamiento del sensor IoT mediante Bluetooth 4.0.

En la **unidad central**:

- **Visualización**. Función para mostrar la lista de los dispositivos bluetooth detectables en el entorno.
- **Emparejamiento.** Función para activar la conexión con los dispositivos periféricos IoT desarrollados.
- **Lectura.** Función para obtener los valores capturados por los sensores y almacenados en los dispositivos periféricos.
- **Almacenamiento.** Función para registrar los valores obtenidos en la base de datos.

Prueba de conectividad

La prueba de conectividad consiste en posicionar un dispositivo periférico IoT de tipo Si7021 con conectividad Bluetooth dentro rango de alcance Bluetooth de la unidad central. El dispositivo, nombrado BW-TH3 se encuentra activo, en modo detectable y obteniendo los valores de temperatura y humedad en el entorno. De esta manera, el dispositivo se encuentra en condiciones ideales para su conectividad en la prueba.

Posteriormente, procedemos a encender la unidad central de monitoreo, donde el programa BLECentral.py será ejecutado automaticamente como un programa de inicio, al estar implementado como un daemon. En ese momento, el programa inicia la búsqueda de dispositivos Bluetooth detectables para encontrar el dispositivo BW-TH3 con MAC 0C:B2:B7:3C:35:66, valores que fueron preestablecidos en el programa. Al ser detectado correctamente, la unidad central solicita al dispositivo los valores almacenados en la memoria del dispositivo, que corresponden al porcentaje de humedad, temperatura y nivel de batería en el dispositivo. En la unidad central, los valores son procesados conforme su tipo de variable, para almacenarse en la base de datos, agregando un identificador clave. En este punto, termina un ciclo de detección, conexión y gestión de datos del dispositivo IoT desde la unidad central de comunicación y monitoreo.

En la Figura 4.1, se muestra una captura de pantalla de la búsqueda de dispositivo IoT de nombre BW-TH3. Como se puede observar, al ejecutar el programa BLECentral.py se inicia la búsqueda del dispositivo IoT con dirección MAC 0C:B2:B7:3C:35:66. La conexión se establece con éxito. Posteriormente, se obtiene el valor almacenado en la característica que corresponde a humedad del dispositivo. El valor de humedad, es procesado y almacenado de manera correcta en la base de datos con el identificador BW-TH3hum. Posteriormente, se repite el procedimiento en el dispositivo con la misma dirección MAC, solicitando los valores de temperatura y nivel batería, con identificadores BW-TH3temp y BW-TH3bat respectivamente.

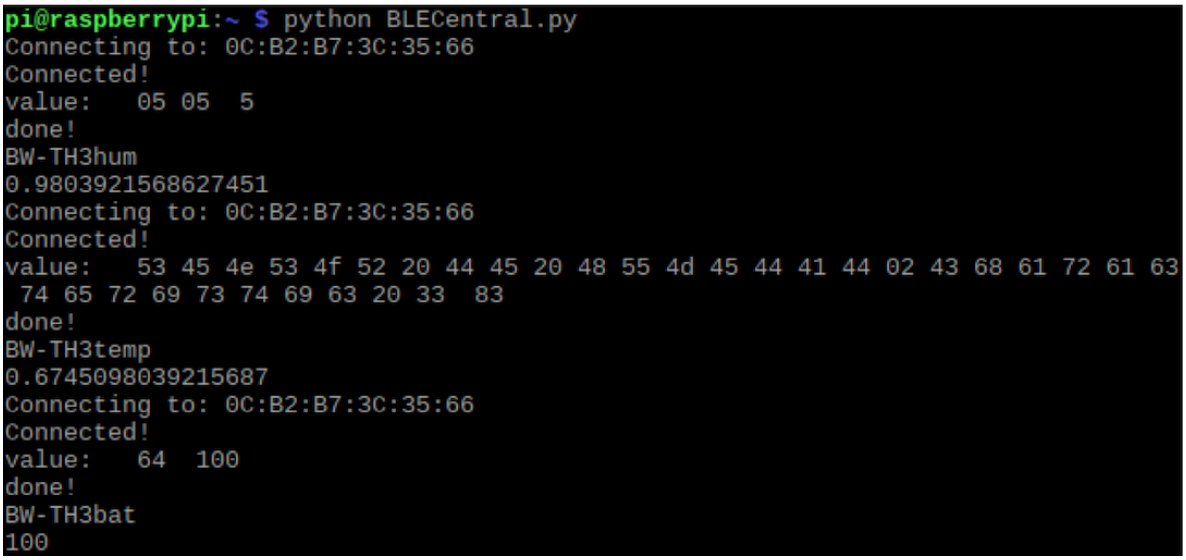

Figura 4.1 Búsqueda, Detección, Conexión y Almacenamiento de valores de dispositivo BW-TH3.

En la Figura 4.2 se muestran los valores de batería, temperatura y humedad registrados por el programa en la base de datos Redis, mostrando su llave y valor, adquiridos del sensor BW-TH3.

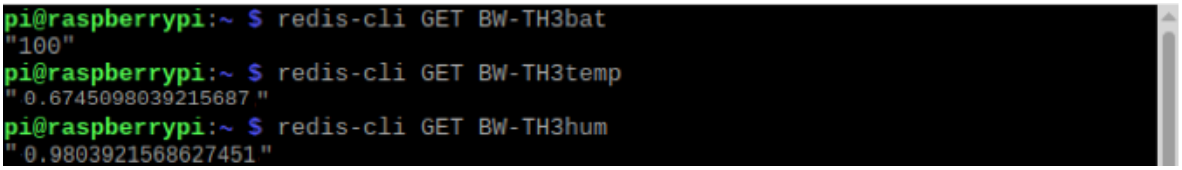

Figura 4.2 Valores Almacenados en Redis de dispositivo BW-TH3.

Prueba de Disponibilidad

La prueba de disponibilidad, consiste en ejecutar un procedimiento similar a la prueba de conectividad, considerando búsqueda y conexión con dispositivos periféricos IoT desde la unidad central de monitoreo y comunicación. Sin embargo, en esta prueba, no existirán dispositivos IoT activos y detectables dentro del rango de alcance Bluetooth de la unidad central.

El procedimiento indica que la unidad central inicia automaticamente la ejecución del programa BLECentral.py al encender, para posteriormente comenzar con la búsqueda del dispositivo BW-TH3 con MAC 0C:B2:B7:3C:35:66. Sin embargo, el sistema realiza la búsqueda del dispositivo sin éxito en 3 ocasiones, finalizando la función de búsqueda y realizando el registro de valores de advertencia en la base de datos para el identificador de BW-TH3hum. En el valor numérico de registrado, se indica la imposibilidad de obtener un parámetro del sensor solicitado, registrando un valor de advertencia como se muestra en la Figura 4.3.

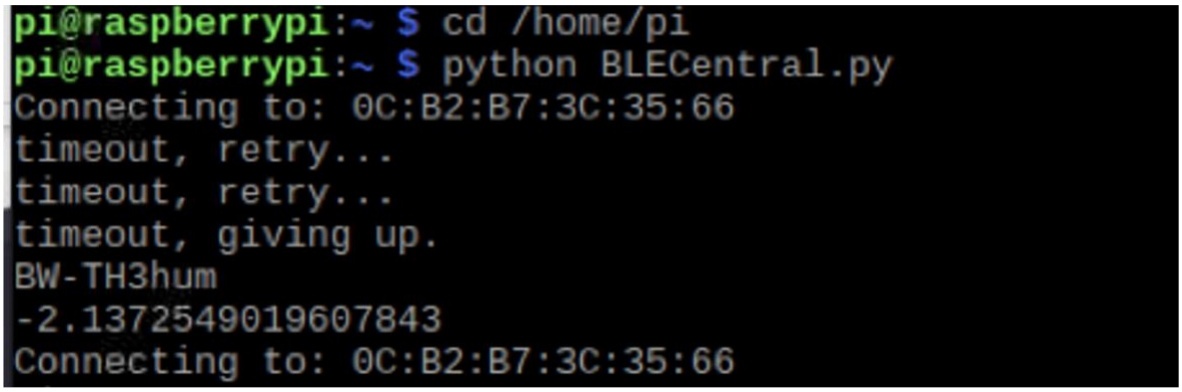

Figura 4.3 Análisis de dispositivos BW-TH3 / Dispositivo no Encontrado

Posteriormente,se realiza el la búsqueda del mismo dispositivo para obtener los valores de temperatura y nivel batería, con identificadores BW-TH3temp y BW-TH3bat respectivamente, tal como se muestra en la figura 4.4. Al no ser encontrado el dispositivo, el sistema registra un valor de advertencia en la base de datos para los indicadores de temperatura y batería. En la Figura 4.5 se muestran los valores

de advertencia almacenados en la base de datos Redis, mostrando su llave y valor, al no encontrase el dispositivo BW-TH3.

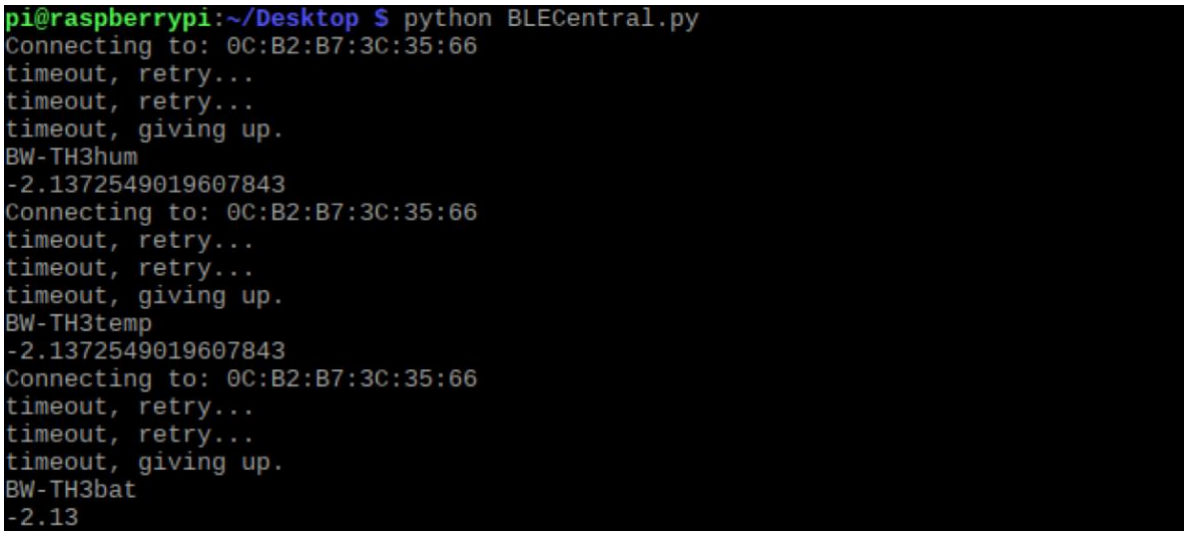

Figura 4.4 Dispositivo no encontrado BW-TH3 sobre temperatura, humedad y Batería

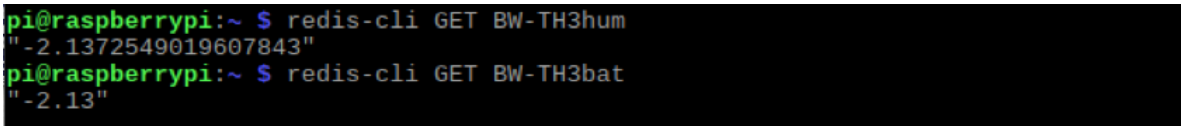

Figura 4.5 Valores de Advertencia Almacenados en Redis sobre dispositivo BW-TH3

5. Conclusiones

En el proyecto realizado, se logró desarrollar una unidad central de monitoreo para un sistema de riego con aplicación en el segmento de huertos urbanos. La integración de dicha unidad con el resto de los componentes del sistema IoT fue conseguido de forma exitosa. Una vez concluido el proyecto, se confirma que los objetivos propuestos se cumplieron satisfactoriamente. El sistema, fue entregado con éxito a la empresa Bitwise Integrated Technologies S. de RL. de CV, para su implementación en campo.

Realizar un sistema de este tipo, me ha brindado la oportunidad de entender de manera general el desarrollo de soluciones de Internet de las cosas aplicadas a problemáticas específicas. Este modelo de sistema podría ser implementado para otro tipo de soluciones de Internet de las Cosas, modificando el tipo de sensores para obtener valores de ecosistemas aplicados en otras áreas, el sistema de almacenamiento de información, los protocolos de procesamiento de los datos y/o modificando los actuadores para realizar acciones específicas dedicadas al área en cuestión.

Como actividades futuras al sistema, podrían implementarse los módulos conectividad a internet, de procesamiento de datos para su análisis y optimización en los procedimientos en la nube, tales como Machine Learning o Big Data y las aplicaciones móviles para interacción con el usuario. Todo esto, con la implementación de bases de datos externas o aplicando directamente de manera local. Otro de los factores a considerar a posterior, es la implementación de tecnologías de comunicación que permitan un mayor alcance en el perímetro de implementación en la red de área local.

Además, el sistema de interacción con el usuario ha sido resumido a acciones fundamentales. En caso de dar continuidad, adicionar al sistema un mecanismo que facilite al usuario interactuar con la información sería una excelente propuesta.

Considero que las posibilidades del Internet de las Cosas, apenas comienzan a vislumbrarse. Que la tecnología sea siempre una herramienta para hacer de este mundo un mejor lugar para vivir.

6. Bibliografía

[1] Evans, D. (2011). *Internet de las cosas. Cómo la próxima evolución de internet lo cambia todo* [Archivo PDF]. CISCO. Recuperado de *https://www.cisco.com/c/dam/global/es_mx/solutions/executive/assets/pdf/internet-of-things-iot-ibsg.pdf*

[2] *Internet of Things*. (2021). In Oxford's English online Dictionary. Oxford: Oxford University Press. Recuperado de http://www.oxforddictionaries.com/us/definition/american_english/Internet-of-things

[3]. *¿Que es Internet of things (IoT)?* (2021). Oracle. Recuperado de https://www.oracle.com/mx/internet-of-things/what-is-iot/

[4] *"Accelerate Innovation With the Power of the Internet of Things* (2015) ORACLE. Recuperado de **https://www.oracle.com/a/ocom/docs/oracle-internet-of-things.pdf**

[5] *Huertos Urbanos* (2021) In *Real Academia Española online Dictionary*. *Real Academia Española*. Recuperado de https://dpej.rae.es/lema/huerto-urbano

[6] *Raspberry Pi (2021) Raspberry Foundation*
Recuperado de
https://www.raspberrypi.org/

[7] *Annual Review 2019* (2019) Raspberry Pi Foundation [Archivo PDF].
Recuperado de
https://static.raspberrypi.org/files/about/RaspberryPiFoundationReview2019.pdf

[8] *Especificaciones de Hardware de Raspberry Zero W* (2021) Raspberry Foundation. Recuperado de
https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md

[9] *Raspberry Pi OS* (2021) Raspberry Foundation
Recuperado de
https://www.raspberrypi.org/software/

[10] Linux (2021) LINUX

Recuperado de
https://www.linux.org/

[11] *Raspberry Pi 32 bit* (2021) Raspberry Foundation
Recuperado de
https://www.raspberrypi.org/software/operating-systems/#raspberry-pi-os-32-bit

[12] *Python* (2021) PYTHON
Recuperado de
https://www.python.org/

[13] *Python Tools* (2021) W3SCHOOLS
Recuperado de
https://www.w3schools.com/python/

[14] *Pexpect* (2013) PEXPECT
Recuperado de
https://pexpect.readthedocs.io/en/stable/

[15] *Redis* (2021) REDIS LAB
Recuperado de
https://redis.io/

[16] *Redis en Amazon Web Services* (2021) AWS.AMAZON
Recuperado de
https://aws.amazon.com/es/redis/

[17] *Bluetooth® Low Energy and proprietary wireless MCU* (2021) TEXAS INSTRUMENT. Recuperado de
https://www.ti.com/product/CC2541?keyMatch=CC2541&tisearch=Search-EN-everything

[18] *TEXAS INSTRUMENT CC2541* (2021) MOUSER ELECTRONICS
Recuperado de:
https://www.mouser.es/new/texas-instruments/ti-cc2541/

[19] *Adafruit Si7021 Temperature & Humidity Sensor Breakout Board* (2021) ADAFRUIT. Recuperado de
https://www.adafruit.com/product/3251

[20] *Adafruit Si7021 Temperature + Humidity Sensor* (2021) ADAFRUIT

Recuperado de
https://learn.adafruit.com/adafruit-si7021-temperature-plus-humidity-sensor/downloads

[20] *Bluetooth smart: An enabling technology for the Internet of Things* (2021) [Archivo PDF] IEEE
Recuperado de
https://ieeexplore.ieee.org/document/7347955

[21] Daemon (2021) LINUX
Disponible en:
https://www.linux.org/docs/man7/daemon.html

Apéndice

BTCentral.py

```
# Using Hexiwear with Python
# Script to get the device data and append it to a file
# Usage
# python GetData.py <device>
# e.g. python GetData.py "00:29:40:08:00:01"
import pexpect
import time
import sys
import os

import redis
from redis import StrictRedis
from time import sleep
value=800
value1=800

#Sensor BW-TH1

#Sensor humedad a Tierra BW-TH1 (Dispositivo 2/humedad)
DEVICE2= "0C:B2:B7:3C:35:66"   # device
characteristic2 = "0x002a"
deviceName2="BW-TH3hum"
style2="sensor"

#Sensor humedad a Tierra BW-TH31 (Dispositivo 2/temperatura)
DEVICE3= "0C:B2:B7:3C:35:66"   # device
characteristic3 = "0x002e"
deviceName3="BW-TH3temp"
style3="sensor"

#Sensor humedad a Tierra BW-TH3 (Dispositivo 2/bateria)
DEVICE4= "0C:B2:B7:3C:35:66"
characteristic4 = "0x0025"
deviceName4="BW-TH3bat"
style4="battery"

def data_toRedis(value, value2,namedevice,style):

    # step 3: create the Redis Connection object
    try:
        r = redis.StrictRedis(host='localhost', port=6379, db=0)
```

```
          #print ("Porcentaje de Humedad registrado en Redis:")


        if style is "sensor":
            intmsg =int((float(value)))
            previo=intmsg/float(255)
            #previo=36/float(255)
            final=1-float(previo)

        else:
            if value==800:
                final=-2.13
            else:
                final=value


          r.set(namedevice,final)
          msg= r.get(namedevice)
          print(namedevice)
        print(msg)


    except Exception as e:
        print(e)

# ----------------------------------------------------------------------
def void (device, characteristicToread,deviceNombre,style):
    child = pexpect.spawn("gatttool -I")
    print("Connecting to:"),
    print(device)
    NOF_REMAINING_RETRY = 3
    while True:
        try:
            child.sendline("connect {0}".format(device))
            child.expect("Connection successful", timeout=5)
        except pexpect.TIMEOUT:
            NOF_REMAINING_RETRY = NOF_REMAINING_RETRY-1
            if (NOF_REMAINING_RETRY>0):
                print "timeout, retry..."
                continue
            else:
                print "timeout, giving up."
                if __name__ == '__main__':
                        data_toRedis(800,800, deviceNombre,style)
                break
        else:
                print("Connected!")
```

```
            if NOF_REMAINING_RETRY>0:

                child.sendline("char-read-hnd "+characteristicToread)
                child.expect("Characteristic value/descriptor: ", timeout=5)
                child.expect("\r\n", timeout=5)
                print("value:  "),
                print(child.before),
                print(str(int(child.before[0:2],16)))

                print("done!")
                value1=(child.before)
                value=(str(int(child.before[0:2],16)))
                if __name__ == '__main__':
                    data_toRedis(value,value1, deviceNombre, style)
                    break
                else:
                    print("FAILED!")
 # sys.exit(-1)

try:
    while True:

        void(DEVICE2,characteristic2,deviceName2,style2);
        void(DEVICE3,characteristic3,deviceName3,style3);
        void(DEVICE4,characteristic4,deviceName4,style4);
        sleep(30)
except KeyboardInterrupt, e:
    logging.info("Stopping...")
```