UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA
DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES

Máster en Ingeniería de Computadores y Redes

Trabajo Fin de Máster

Integración de smartphones en el IoT

Autor: <*Zhongyuan Ren*>

Director(es): <*Pietro Manzoni*>

<*7/2021*>

Prefacio

Las principales tareas realizadas en este trabajo son las siguientes.

1) Se han analizado las tecnologías actuales de IoT, se han analizado los requisitos y las funciones, y se ha seleccionado la tecnología y el modo de transmisión más adecuados.

2) Se han analizado las características de los diferentes protocolos de comunicación en las redes de sensores inalámbricos, y se ha seleccionado el modo de protocolo de comunicación adecuado según el escenario de aplicación de este trabajo. Implementado dos módulos funcionales básicos comúnmente usados en IoT: recepción y envío de la data basados en BLE y MQTT y realizado una aplicación.

(3) Aprender a utilizar correctamente las herramientas de software Android Studio y la plataforma Atom, y el desarrollo de los lenguajes de programación JAVA y Micro Python.

(4) Diseñar e implementar dos protocolos de transmisión basados en el sistema Android, implementar dos formas diferentes de recibir y enviar datos, y poder elegir e utilizar la conexión de red o la conexión Bluetooth para recibir datos de IoT para su propio uso, además de desarrollar aplicaciones de IoT desde cero, mostrando cómo diseñar y desarrollar aplicaciones de IoT, desde la selección de aplicaciones de IoT y todo el proceso de desarrollo de la aplicación.

Índice de contenido

简介

物联网（InternetofThings）是指通过各种信息传感器、射频识别技术、GPRS、红外感应器、激光扫描器等各种装置与技术，实时采集任何需要监控、连接、互动的物体或过程，采集其声、光、热、电、力学、化学、生物、位置等各种需要的信息，通过各类可能的网络接入，实现物与物、物与人的泛在连接，实现对物品和过程的智能化感知、识别和管理。物联网是一个基于互联网、传统电信网等的信息承载体，它让所有能够被独立寻址的普通物理对象形成互联互通的网络。

智能手机由于其具有广泛的普及性，良好的硬件水平和操作系统，以及搭载了Bluetooth、WiFi 和 3G/4G 通信方式等多种优势，可以设计并制作一个简单的应用用来接收和发送物联网硬件。

这项工作的重点是从零开始设计和开发智能手机应用程序，并在其中使用低功耗蓝牙 和 MQTT 集成 IoT 设备。 对于 BLE（bluetooth），需要实现设备发现、连接、接收的服务。 对于通过 WiFi 链路使用 MQTT 协议，实现互连的本地化、代理连接和定义方面。设备将使用 Lopy4 和 Pysense

关键词:物联网，智能手机，低功耗蓝牙，Android Studio，MQTT

Resumen

Internet de las cosas o Internet of things (IoT) se refiere a diversos dispositivos y tecnologías, como diversos sensores que proveen distinta información, tecnología de radiofrecuencia, tecnologías móviles de diversas generaciones como 2G,3G,4G e incluso muy pronto utilizando 5G, escáneres, láser, entre otras cosas. La finalidad es recoger cualquier objeto o proceso que deba ser supervisado y conectado y a su vez interactuar con él en tiempo real, recogiendo su data que puede ser sonido, luz, calor, electricidad, mecánica, química, biología, ubicación y otra información necesaria, y accediendo a ella a través de diversas redes posibles La Internet de las cosas (IoT) es una red que permite la conexión ubicua de cosas y personas para lograr la detección, identificación y gestión inteligentes de objetos y procesos. También conocida como el Internet de los objetos, es un soporte de información basado en Internet, las redes de telecomunicaciones tradicionales, etc. Permite que todos los objetos físicos ordinarios que pueden ser direccionados y que independientemente forman una red interconectada.

Los teléfonos inteligentes pueden diseñar y realizar una aplicación sencilla para recibir y enviar hardware IoT debido a su amplia popularidad, su buen nivel de hardware y sistema operativo, y a que están equipados con diversas ventajas, como los métodos de comunicación Bluetooth, WiFi y 3G/4G.

El objetivo de este trabajo es diseñar y desarrollar una aplicación para smartphones desde cero e integrar en ella dispositivos IoT utilizando Bluetooth de bajo consumo y MQTT. En el caso de BLE (bluetooth), es necesario implementar servicios de descubrimiento, conexión y recepción de dispositivos. Para utilizar el protocolo MQTT a través de un enlace WiFi, se implementan aspectos de localización, conexión proxy y definición de la interconexión. Los dispositivos utilizarán Lopy4 y Pysense.

Palabras clave: Internet de las cosas, Smartphone, Bluetooth de bajo consumo, Android Studio, MQTT.

Abstract

Internet of Things refers to the collection of any object or process that needs to be monitored, connected and interacted with in real time through various devices and technologies such as various information sensors, radio frequency identification technology, GPRS, infrared sensors, laser scanners, etc., and the collection of its sound, light, heat, electricity, mechanics, chemistry, biology, location and various other needed information, through various possible network access that enables a ubiquitous connection between things and things, and things and people, to achieve intelligent sensing, identification and management of objects and processes. The Internet of Things is an information carrier based on the Internet, traditional telecommunication networks, etc. It allows all common physical objects that can be independently addressed to form an interconnected network.

Smartphones, due to their wide popularity, good level of hardware and operating systems, as well as being equipped with a variety of advantages such as Bluetooth, WiFi and 3G/4G communication methods, can design and produce a simple application for receiving and sending IoT hardware.

The focus of this work is to design and develop a smartphone application from scratch and to integrate IoT devices within it using low power Bluetooth and MQTT. For BLE (Bluetooth), services for device discovery, connection and reception need to be implemented. For the use of the MQTT protocol over WiFi links, the localization, proxy connection and definition aspects of the interconnection are implemented. The devices will use Lopy4 and Pysense

Keywords: Internet of Things, smartphones, low-power Bluetooth, Android Studio, MQTT

I. Introducción

1.1. Introducción a los conceptos básicos del Internet de las cosas

1.1.1. La definición del Internet de las cosas

El concepto de "Internet de las cosas" es ahora muy conocido y utilizado en varios países del mundo, pero, ¿qué es el Internet de las cosas? el Internet de las cosas es el mundo real de objetos conectados a Internet, por lo que las cosas, objetos e incluso personas pueden interactuar y comunicarse entre sí.

Figura 1.1.1.1 Conectividad del IoT.

1.1.2. El estado del sistema de telefonía móvil

La aparición del smartphone se remonta al año 1999, cuando se le llamó "ordenador de mano" y se esperaba que fuera un cambio importante en la informática. Sin embargo, la introducción de estos teléfonos atrajo a los consumidores por su novedad e inteligencia, pero también por su elevado precio, lo que desanimó a muchos consumidores. Por otro lado, la teoría de los dieciocho meses en la tecnología ha permitido que todos los productos electrónicos innoven a un ritmo cada vez mayor. A día de hoy, hay casi un smartphone por persona. Los teléfonos móviles se han convertido en un producto de consumo indispensable para todas las familias. Con la popularidad de Internet, el teléfono móvil ha pasado gradualmente de ser una simple función de llamadas y mensajes de texto a una enorme plataforma para que la gente acceda a Internet y se convierta en una ventana para entender el desarrollo y los cambios del mundo. Los principales fabricantes compiten por la cuota de mercado actual con sus propios trucos.

Hay una gran variedad de sistemas de teléfonos inteligentes en el mundo, como se muestra en la Figura 1.1.2.1, seleccionamos los sistemas más representativos para presentarlos.

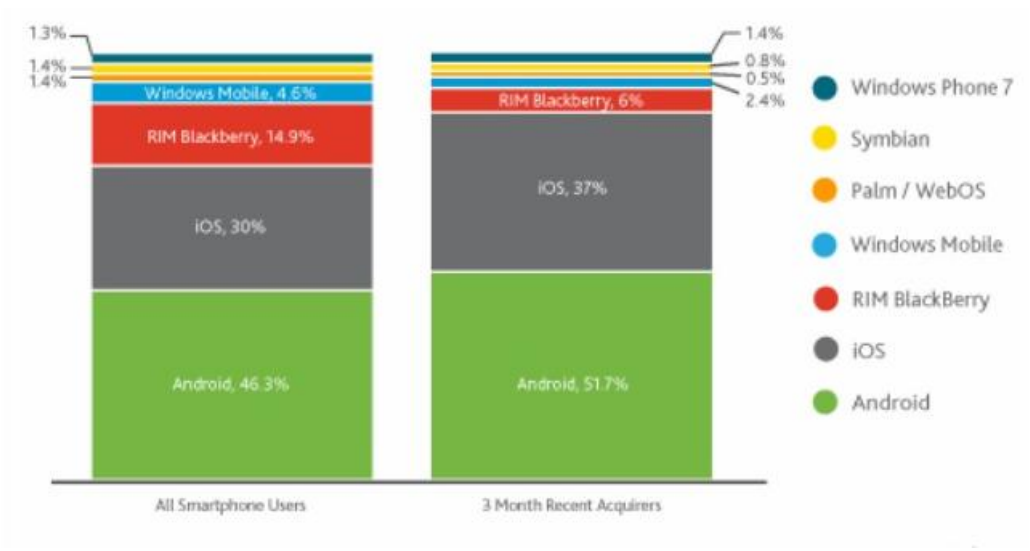

Figura 1.1.2.1 Gráfico de la cuota de los teléfonos inteligente.

1.1.2.1. Android [1]

Android es un sistema operativo gratuito y de código abierto basado en el Kernel de Linux, que no contiene componentes GNU. Utilizado principalmente en dispositivos móviles como teléfonos inteligentes y tabletas. Google fue dirigido y desarrollado en los Estados Unidos y la Alianza Móvil Abierta en noviembre de 2007, Google y 84 fabricantes de hardware, desarrolladores de software y operadores de telecomunicaciones formaron la Alianza Móvil Abierta para desarrollar y mejorar los sistemas Android. Algo adicional, es que Google lanzó el código fuente de Android en forma de una licencia de código abierto. El primer smartphone Android fue lanzado en octubre de 2008. Android se está expandiendo a tabletas y otras áreas como televisores, cámaras digitales, consolas de juegos, relojes inteligentes y más. En el primer trimestre de 2011, Android superó a BlackBerry por primera vez en el mundo, ocupando el primer lugar en el mundo.

1.1.2.2.Ios[2]

iOS es un sistema operativo móvil desarrollado por Apple, Apple presentó por primera vez el sistema en la conferencia MacWorld el 9 de enero del 2007, inicialmente diseñado para el uso del iPhone, y desde entonces se ha utilizado en iPod touch y iPad. iOS, al igual que el sistema operativo macOS de Apple, es un sistema operativo comercial que clasifica a Unix. Originalmente llamado sistema operativo iPhone, Apple cambió su nombre a iOS (iOS es una marca comercial del sistema operativo de dispositivos de red de Cisco en los Estados Unidos) en 2010 porque el iPad, el iPhone y el iPod touch utilizan el sistema operativo iPhone Autorizado por Cisco Corporation).

1.1.2.3.BlackBerry [3]

BlackBerry, un sistema operativo para dispositivos terminales inalámbricos de resolución de correo portátil lanzados por Research In Motion Canada(RIM), fue desarrollado por RIM. A diferencia de otros sistemas operativos como Symbian, Windows Mobile iOS utiliza dos por otros terminales móviles, los sistemas BlackBerry están más encriptados y seguros. Un BlackBerry con un sistema BlackBerry no es sólo un teléfono, sino un servicio de correo electrónico Push Mail en tiempo real con servidores (configuración de correo), software (interfaces operativas) y terminales (teléfonos móviles) que son lanzados por RIM.

1.1.2.4. Windows Mobile [4]

Windows Mobile (WM) es el sistema operativo de Microsoft para dispositivos móviles, este sistema operativo fue diseñado para estar lo más cerca posible de la versión de escritorio de Windows, y Microsoft diseñó el WM en el modo del sistema operativo PC para que pudiera ser idéntico al sistema operativo del PC. Las aplicaciones de WM se basan en la API de Microsoft Win32. La serie Windows Mobile salió oficialmente del mercado de teléfonos después de la aparición de un nuevo sucesor en el sistema operativo Windows Phone. En octubre de 2010, Microsoft anunció que estaba terminando todo el soporte técnico para WM.

1.1.2.5. WebOS [5]

WebOS es un sistema operativo de última generación que se extiende como nunca antes a través de un cliente de red. Por Jon A. S.Jon Rubenstein lideró el desarrollo, y Palm Pre fue el primer teléfono inteligente en adoptar web OS. En la interfaz humano-máquina, WebOS es muy amigable, mejora la experiencia táctil y mejora la conectividad inalámbrica también es una característica importante del nuevo sistema, además de admitir la función de búsqueda unificada de contenido web y datos en el teléfono, la capacidad de manejar múltiples tareas en línea al mismo tiempo también vale la pena esperar.

1.1.2.6. Symbian [6]

Symbian es el sistema operativo de Saipan diseñado para teléfonos móviles, y su predecesor fue el sistema operativo EP (Electronic Piece of cheese) de The Great British Company. Saipan fue adquirida por Nokia el 2 de diciembre de 2008. El 21 de diciembre del 2011, Nokia anunció oficialmente que abandonaba la marca Symbian. La cuota de mercado de Saipan se está reduciendo debido a la falta de apoyo de las nuevas tecnologías. En febrero del 2012, la cuota de mercado global de Saipan era de sólo el 3%. El 27 de mayo del 2012, Nokia abandonó por completo el desarrollo del sistema Saipan, pero el servicio continuaba hasta el año 2016. En la noche del 24 de enero del 2013, Nokia anunció que ya no lanzaría los teléfonos de Saipan, lo que significa que Saipan, un sistema operativo para teléfonos inteligentes, finalmente había llegado a su fin después de 14 años de historia. El 1 de enero del 2014, Nokia dejó oficialmente de actualizar las aplicaciones de Saipan en la Tienda Nokia y prohibió a los desarrolladores lanzar nuevas aplicaciones.

1.1.2.7. Windows Phone [7]

Windows Phone (WP para abreviar) es un sistema operativo móvil lanzado oficialmente por Microsoft el 21 de octubre del 2010, inicialmente llamado Windows Phone 7.0. basado en el kernel de Windows CE, utiliza una interfaz de usuario (UI) llamada Metro e integra los juegos de Xbox Live de Microsoft, música de Xbox Music y experiencias de vídeo únicas en el teléfono. En febrero del 2011, Nokia y Microsoft entraron en una alianza estratégica global y trabajaron en profundidad para desarrollar el sistema. El 27 de septiembre del 2011, Microsoft lanzó una versión actualizada de Windows Phone 7.5, la primera versión del sistema compatible con Chinese Simplified. El 21 de junio de 2012, Microsoft lanzó oficialmente Windows Phone 8, un nuevo Windows Phone 8 que abandona el antiguo kernel de Windows CE y utiliza el mismo núcleo de Windows NT que el sistema Windows, soportando muchas características nuevas. Los teléfonos de todos los sistemas Windows Phone 7.5 no podrán actualizar a Windows Phone 8 debido a un cambio en el kernel. Pero con el fin de cuidar de los usuarios de los sistemas Windows Phone 7.5,

Microsoft lanzó Windows Phone 7.8, que tiene algunas características de Windows Phone 8. El 2 de abril del 2014, Microsoft lanzó Windows Phone 8.1 en build en el 2014, agregando más características nuevas que Windows Phone 8, actualizando algunos componentes y anunciando que todos los dispositivos Windows Phone 8 podrán actualizarse a Windows Phone 8.1. En julio del 2014, Microsoft lanzó Windows Phone 8.1 Update 1, agregando algunas características a Windows Phone 8.1 y realizando algunas optimizaciones. En febrero del 2015, cuando Microsoft insertó la segunda vista previa de Windows 10 Mobile, la primera fase empujó Windows Phone 8.1 Update 2, mejorando la forma en que funcionan algunas características en Windows Phone 8.1 Update 1.

Un sucesor de Windows Phone es Windows 10 Mobile. En este proyecto se desarrollará todo para el sistema operativo Android, ya que es la mayor cuota de mercado.

1.2. La composición del Internet de las cosas

1.2.1. Componer

Todo el sistema consta de tres componentes: una placa de desarrollo basada en IoT, un teléfono inteligente con un sistema Android y sensores que integran tecnologías inalámbricas que conectan todo.

1.2.2. Principios de diseño

La aplicación básica del Internet de las cosas es conectar todos los elementos al sensor que provee la información y conectar el dispositivo a Internet para completar la transmisión de toda la data obtenida por los sensores. Por lo tanto, para diseñar una conexión a Internet de las cosas en un teléfono inteligente, con el fin de ver la información recolectada, esto se ha hecho mediante la transmisión de información a través de la red GPRS o Bluetooth del programa de teléfono móvil.

1.2.3. Diseño de métodos

El proceso de diseño consta de dos partes, una basada en el desarrollo de aplicaciones del teléfono Android, que utiliza BLE del propio sistema para comunicarse con el hardware, y la otra basada en la plataforma de proxy IoT de conexión WiFi.

1.3. Análisis estructural de aplicaciones de redes de teléfonos inteligentes

Los sistemas de telefonía móvil de hoy en día son numerosos, en este documento se prioriza el desarrollo de aplicaciones de teléfonos inteligentes basados en Android, interacción humano-ordenador de máquina real, interacción física de la transmisión de información. Debido a la naturaleza multiplataforma del lenguaje Java, los desarrollamos como un lenguaje básico, depurando programas con Android Studio en nuestros ordenadores, y luego en el sistema Android 8.0.0. Sobre la base de los programas relacionados con BLE para teléfonos inteligentes que ya existen en la red, los extraemos y llevamos a cabo un desarrollo secundario para conectar teléfonos inteligentes

a dispositivos IoT, y así que reciba datos del módulo de detección de luz sobre el hardware de IoT al mismo tiempo y envíe datos al módulo de la lámpara LED. Esto permite a los teléfonos inteligentes controlar el hardware de IoT.

Las características específicas del software incluyen las funciones principales de la búsqueda de información perimetral y el envío y recepción de confianza. El envío y la recepción nos permiten comunicarnos con los artículos. La función de búsqueda transmite la identificación de frecuencia a los elementos circundantes equipados con nodos de detección inalámbrica con el fin de satisfacer los requisitos de búsqueda del individuo.

II. Visión general de las tecnologías de IoT

El Internet de las cosas generalmente se divide en tres niveles: la capa de percepción, la capa de red y la capa de aplicación, como se muestra en la Figura 2.1, que describe la arquitectura del sistema IoT.

Capa de percepción

La capa de percepción es la parte inferior de la arquitectura de tres niveles del Internet de las cosas, compuesta principalmente por una variedad de dispositivos perceptivos y redes perceptivas, incluyendo sensores de temperatura y humedad, sensores infrarrojos, cámaras, etc.

Capa de red

La capa de red también se puede llamar la capa de transporte, la función principal es realizar la transmisión de datos percibidos e instrucciones entre la capa perceptiva y la aplicación superior, para realizar el acceso y la transmisión de equipos y datos y otras funciones, es la ruta de datos para la transmisión de información entre sí. Incluye principalmente dos tipos de red de acceso y red de transmisión.

Capa de aplicación

Como capa superior, la capa de aplicación se compone principalmente de aplicaciones y plataformas de todos los ámbitos de la vida, y los datos percibidos transmitidos a través de la capa de red pueden ser procesados por varias plataformas y aplicaciones en esta capa, lo que permite la interacción entre los usuarios y varios dispositivos.

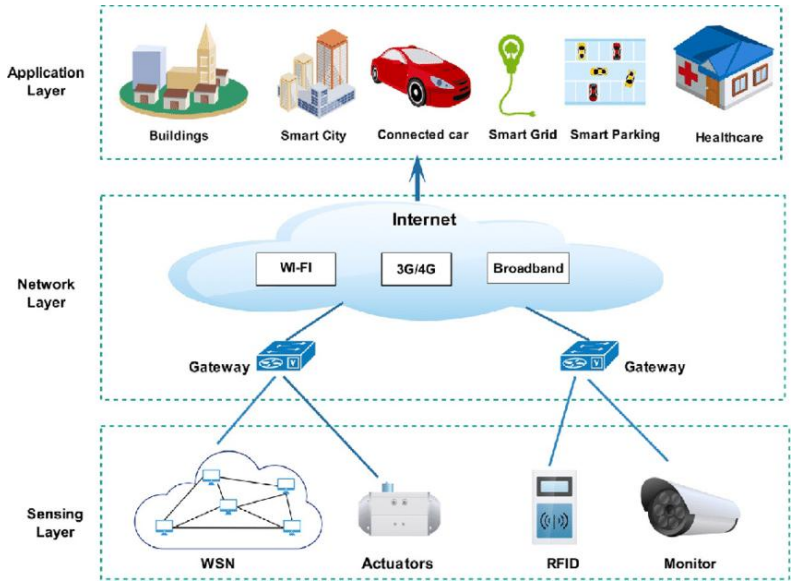

Figura 2.1 Diagrama de la arquitectura del sistema IoT.

Entre diferentes niveles, la información no se transmite en una dirección, sino que se puede interactuar y controlar, y la información que se puede transmitir también es diversa. Aunque las características de aplicación del Internet de las cosas en el hogar inteligente, transporte inteligente, agricultura moderna, energía inteligente, servicios logísticos, ecología verde, monitoreo ambiental, salud móvil, inteligencia artificial y así sucesivamente no son las mismas, pero la arquitectura básica de cada aplicación incluye la capa de percepción, la capa de red y la capa de aplicación de tres niveles, o utilizar estos tres niveles como base para mejorar la implementación del sistema y el diseño de aplicaciones.

2.1 Estructura y funcionalidad típicas de la puerta de enlace

Las puertas de enlace son una parte muy importante de las aplicaciones de IoT. La puerta de enlace se conecta al dispositivo de detección para administrar y recopilar datos para el dispositivo de detección subyacente, se conecta a la plataforma en la nube y proporciona datos para análisis y servicios posteriores. Como puente arriba y abajo del sistema de Internet de las Cosas, la puerta de enlace tiene el impacto más directo en el flujo de datos, por lo que la elección de una puerta de enlace eficiente y estable es una de las bases para el funcionamiento normal de un sistema de IoT.

Una aplicación típica de puerta de enlace de IoT tiene tres características:

(1) La función de acceso de la red perceptiva, es decir, se puede conectar al dispositivo perceptivo y a la red para obtener el estado y la información del dispositivo perceptivo.

(2) Las funciones de gestión de dispositivos y datos, es decir, la red perceptiva de información del dispositivo se puede procesar y gestionar, incluyendo el control de activación, el mantenimiento, etc. También incluye una serie de otros servicios de seguridad y gestión de derechos.

(3) Función heterogénea de conversión de protocolo de red, es decir, diferente información y datos perceptivos de dispositivos de red pueden ser embalaje unificado, y acceso a Internet, pero también puede soportar la información de Internet enviada a la red perceptiva, para lograr una comunicación estable entre aguas arriba y aguas abajo.

La arquitectura de una puerta de enlace de IoT típica consta de cuatro partes: la capa de hardware, la capa del sistema, la capa de middleware y la capa de aplicación. Como se muestra en la figura 2.2. Las principales estructuras y funciones de cada una de estas secciones incluyen:

(1) Capa de hardware: La capa de hardware es una combinación de recursos de hardware físicos, principalmente incluyendo unidades de procesamiento central (Central Processing Unit, CPU), interfaces físicas, módulos de RF, gabinetes, etc., es la base de hardware para toda la operación de puerta de enlace.

(2) Capa del sistema: La capa del sistema es principalmente el sistema operativo, responsable de invocar el controlador del sensor, llamando a la capa de hardware del dispositivo, para lograr la recepción y transmisión de datos de capa de hardware. Al mismo tiempo, también es responsable de llamar a varias pilas de protocolos de comunicación para completar el intercambio de datos entre redes. Esta es la base para que se ejecute el software de puerta de enlace.

(3) Capa middleware: La capa middleware es responsable de la gestión de dispositivos conectados a la puerta de enlace, el procesamiento y la administración de datos dentro de la puerta de enlace. Los datos de la red de detección se procesan y consolidan para su uso en una variedad de aplicaciones en el nivel superior. También incluye la gestión de interfaces de red y la transmisión de datos entre redes. Esta es la parte principal de la

ejecución de puerta de enlace.

(4) Capa de aplicación: La capa de aplicación es responsable de la gestión de una variedad de aplicaciones de IoT basadas en Gateway directo, incluyendo control de dispositivos, administración de políticas, etc. Características como diferentes interfaces de servicio también están disponibles para diferentes escenarios y usuarios. La capa de aplicación es la capa superior de la puerta de enlace de IoT, que enfrenta directamente a los usuarios y a las aplicaciones de nivel superior.

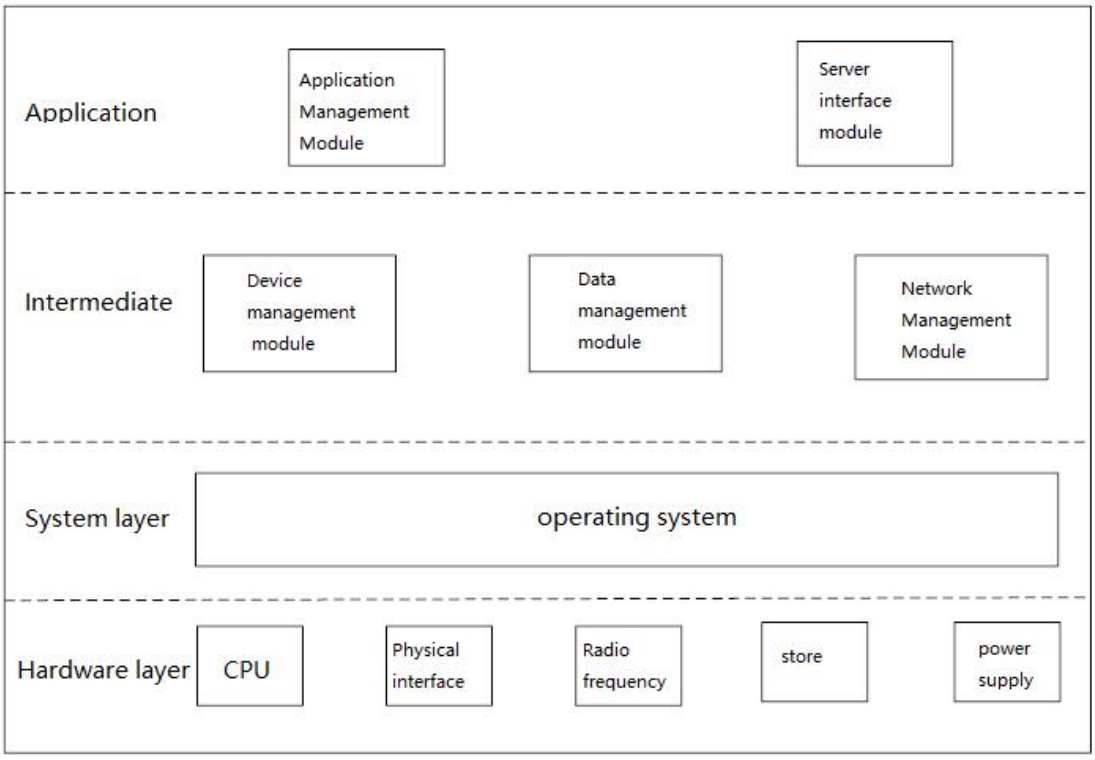

Figura 2.1.1 Arquitectura de la pasarela IoT.

2.2 Tipos de puertas de enlace comunes

En esta sección se analizan los tipos de puertas de enlace en varios entornos de aplicaciones de IoT típicos en la vida real y sus relaciones con los dispositivos subyacentes, incluidas las puertas de enlace de automatización industrial, las puertas de enlace fijas domésticas, las puertas de enlace de teléfonos inteligentes y algunos dispositivos extraíbles que pueden conectarse directamente a Internet. Al comparar los escenarios y ventajas y desventajas de los diferentes tipos de puertas de enlace, se analizan los requisitos reales de las puertas de enlace de IoT en este diseño.

2.2.1. Puerta de entrada de automatización industrial

Las aplicaciones de automatización industrial son un escenario muy común, y los dispositivos de puerta de enlace se utilizan ampliamente en entornos industriales antes de entrar en la vida de las personas.

Los primeros sistemas tradicionales de control industrial fueron diseñados generalmente para funcionar sólo en un entorno industrial particular, con capacidades de automatización débiles, y una amplia variedad de equipos y diferentes modos de comunicación, por lo que las puertas de enlace de grado industrial en este momento eran más bien un convertidor de protocolo para permitir la comunicación entre diferentes dispositivos y la sala.

Más tarde, con la mejora de la capacidad de control de automatización industrial, la función de la puerta de enlace está aumentando gradualmente, convertirse en una pequeña máquina de control industrial, en este momento la puerta de enlace no sólo puede lograr la conversión de protocolo, sino también hasta cierto punto para lograr el control automatizado de políticas, reducir la intervención manual. Es la estructura general de la red industrial, compuesta principalmente por equipos industriales, puerta de enlace industrial, bus de red y host y servidor de gestión. Los equipos industriales generales están conectados directamente a la puerta de enlace industrial por medios cableados o inalámbricos, y luego la puerta de enlace industrial está conectada al bus de red, incluyendo una variedad de autobuses de campo y redes, mientras que los diferentes hosts y servidores de gestión también están conectados al bus de red. La puerta de enlace industrial transmite información de dispositivo diferente al bus de red para su posterior gestión por parte del host de administración, al tiempo que responde a las instrucciones y datos emitidos por el host de administración.

En la actualidad, con el desarrollo de Internet e Internet de las cosas, la puerta de enlace de automatización industrial también integra las capacidades de acceso a Internet, realiza monitoreo y control de largo alcance, e incluso puede realizar la solución automática de problemas de fallas, volviéndose más inteligente. Sin embargo, las aplicaciones de grado industrial y los escenarios de aplicaciones normales no son los mismos, las aplicaciones de grado industrial generalmente tienen una alta eficiencia a gran escala, etc., y las aplicaciones industriales pueden ser un entorno de trabajo más duro, más métodos de comunicación, lo que también requiere que la puerta de enlace de automatización industrial debe tener un grado relativamente alto de estabilidad y compatibilidad. Las puertas de enlace de automatización

industrial deben cumplir estas características, por lo que, en cierta medida, las puertas de enlace de grado industrial también son más caras, más intensivas en energía, etc. Estas características de las puertas de enlace de grado industrial limitan el uso de la escena, por lo que el grado de popularidad es relativamente bajo.

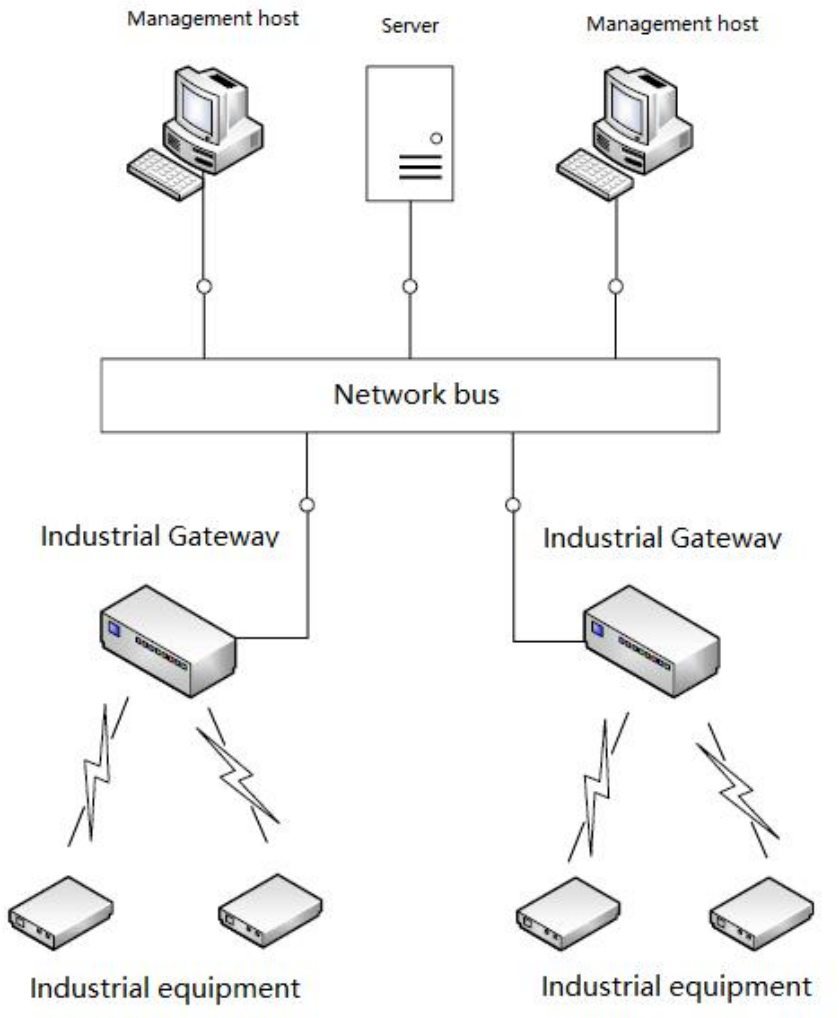

Figura 2.2.1.1 Pasarela de automatización industrial.

2.2.2. Puerta de enlace fija en casa

En la actualidad, el hogar inteligente es una de las aplicaciones típicas del Internet de las cosas, esto se debe ya que la tecnología actual está más desarrollada que antes, además el bajo costo, entre otras cosas. Una característica típica de estas aplicaciones de IoT es la necesidad de una puerta de enlace doméstica, instalada en una ubicación específica en el hogar, acceso

por cable o inalámbrico a Internet, y luego el dispositivo subyacente está conectado directamente a la puerta de enlace doméstica. Los usuarios pueden administrarlo desde una página web o una aplicación en su teléfono. Tomemos la puerta de enlace multifunción Mijia de Xiaomi Technology, por ejemplo, que puede coincidir con otro hardware inteligente lanzado por Mijia, principalmente a través de la tecnología ZigBee, y los usuarios tendrán que instalar una aplicación Mijia en sus teléfonos para administrar estos dispositivos y puertas de enlace.

La Figura 2.2.2.1 es la estructura de red del dispositivo en un escenario típico de hogar inteligente, que consiste principalmente en terminales inteligentes, puertas de enlace domésticas, routers y computadoras personales o teléfonos inteligentes. Los terminales inteligentes se conectan primero a la puerta de enlace doméstica, que luego se conecta a Internet a través de un router, normalmente utilizando métodos de comunicación inalámbrica como ZigBee y WiFi, y es administrado por la puerta de enlace doméstica para la información y los datos de estos dispositivos. Al mismo tiempo, los usuarios también utilizan ordenadores personales, teléfonos inteligentes, etc. para ver o controlar la información y el estado de estos terminales inteligentes a través de Internet.

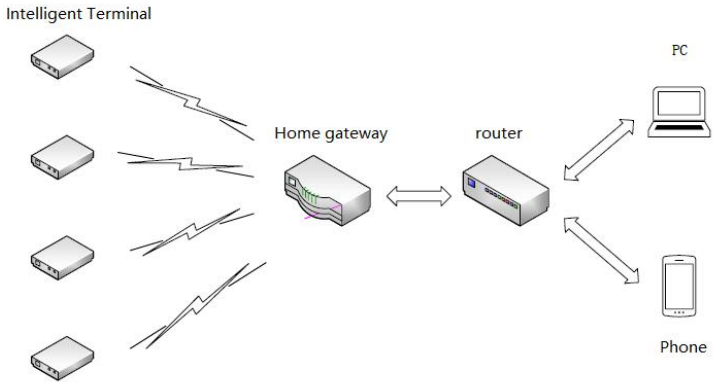

Figura 2.2.2.1 Pasarela fija doméstica.

Pero un inconveniente notable de esta puerta de enlace fija es que no se puede utilizar en hardware y sólo puede comunicarse con los dispositivos de la compañía, por ejemplo, cuando un usuario ya ha implementado una puerta de enlace multifuncional a Huawei y ha comprado un hardware inteligente producido por Huawei, es posible que el usuario también necesite comprar una puerta de enlace Huawei para el hogar, porque diferentes fabricantes pueden no ser capaces de conectarse incluso si utilizan estándares de

comunicación comunes. Por lo tanto, la incapacidad de reutilizar este hardware resulta en un desperdicio de recursos y un aumento de los costos.

2.2.3. Puerta de entrada para smartphones

En los últimos años, con el desarrollo de la sociedad y el progreso de la ciencia y la tecnología, la popularidad de los teléfonos inteligentes en el mundo es cada vez mayor, se ha integrado completamente en la vida de las personas. Al mismo tiempo, los teléfonos inteligentes de hoy en día no están menos equipados que los ordenadores personales, ya sea a nivel de hardware o en sistemas operativos y aplicaciones e incluso algunos teléfonos insignia están mejor equipados que la mayoría de los PC, y mucho menos otros dispositivos finales inteligentes. Y los teléfonos inteligentes en el mercado hoy en día están equipados con una variedad de módulos de comunicación inalámbrica, incluyendo WiFi, Bluetooth, 4G y 5G, lo que hace que sea fácil de usar teléfonos inteligentes para reunir dispositivos que utilizan estas tecnologías de comunicación inalámbrica para interactuar con los datos de una variedad de maneras. Por lo tanto, basado en la naturaleza de buen rendimiento y generalizada de los teléfonos inteligentes, es natural pensar en el uso de teléfonos inteligentes en la arquitectura IoT para permitir el uso de teléfonos inteligentes para conectar una variedad de dispositivos de sensores a Internet, especialmente dispositivos de detección utilizando la tecnología Bluetooth low energy (BLE).

Con la actualización continua y la mejora de las especificaciones estándar Bluetooth, la tecnología Bluetooth 4.0 BLE es cada vez más madura, debido a sus conocidas características excelentes de baja potencia y bajo costo, ha sido ampliamente utilizada en la capa de percepción de Internet de las Cosas de una variedad de dispositivos de detección. Los nuevos teléfonos inteligentes en el mercado hoy en día son casi 100 por ciento Bluetooth 4.0 y superior, y los dispositivos de sensores que utilizan la tecnología BLE pueden comunicarse directamente con los teléfonos inteligentes. Es por eso que hemos diseñado teléfonos inteligentes como una puerta de enlace de IoT que puede utilizar teléfonos inteligentes omnipresentes para conectarse a dispositivos de detección BLE omnipresentes, rompiendo la estructura de la "chimenea" de las aplicaciones tradicionales del sistema IoT.

En resumen, las ventajas de usar un smartphone como puerta de enlace de IoT son las siguientes:

(1) Los teléfonos inteligentes tienen una amplia gama de popularidad y

excelente rendimiento.

(2)

(2) Buena compatibilidad con la tecnología Bluetooth 4.0 BLE y redes móviles.

(3) Reduce significativamente el costo de desarrollar puertas de enlace de IoT comunes para cada proveedor.

(4)

(4) Los teléfonos inteligentes tienen una interfaz operativa visual, lo que facilita la gestión intuitiva y visualización de dispositivos y datos, así como el desarrollo y el uso de una variedad de aplicaciones de terceros.

Por lo tanto, en el mercado actual de aplicaciones de IoT, además de la puerta de enlace fija en casa, hay una categoría de aplicaciones de Internet de las cosas están utilizando teléfonos inteligentes como puertas de enlace, tales como pulseras inteligentes, relojes inteligentes y así sucesivamente. Una característica llamativa de estos dispositivos inteligentes es que no necesitan comprar una puerta de enlace por separado, solo un teléfono inteligente, e instalar una aplicación complementaria. Esto permite la reutilización de hardware de dispositivos de puerta de enlace utilizando un teléfono inteligente que puede proporcionar servicios de transferencia de datos para múltiples pulseras inteligentes y relojes inteligentes, en comparación con el uso de una puerta de enlace doméstica específica. Esto ahorra dinero a los usuarios. La Figura 2.2.3.1 es un escenario de aplicación típico que utiliza un teléfono inteligente como puerta de enlace, con pulseras inteligentes, relojes inteligentes y otros dispositivos conectados directamente al teléfono inteligente, lo que permite a los usuarios ver la información del dispositivo directamente a través de la aplicación montada en el teléfono inteligente.

Figura 2.2.3.1 Pasarela inteligente.

En cierto modo, las puertas de enlace son diferentes aplicaciones, no el teléfono inteligente en sí.

2.3 El estado actual de la comunicación inalámbrica

En los últimos años, la tecnología de comunicación inalámbrica de largo alcance es una tecnología de comunicación que se ha desarrollado rápidamente y es ampliamente utilizada en el campo de la información y la comunicación. En el rápido desarrollo de redes inalámbricas, una variedad de estándares de transmisión de datos de red inalámbrica de corto alcance, tales como: WiFi, Bluetooth, ZigBee, infrarrojos, ANT plus, comunicación de campo cercano (Comunicación de campo cercano, NFC) y así sucesivamente, diferentes estándares generalmente corresponden a diferentes áreas de aplicación. Estas diferentes normas tienen su propio entorno aplicable, pero también tienen sus propias ventajas y desventajas, como ZigBee se puede implementar auto-networking, adecuado para un pequeño número de sistema de transmisión de información de control, pero generalmente necesita el apoyo de una puerta de enlace ZigBee específica; WiFi se utiliza para la transferencia de grandes cantidades de datos, tiene la ventaja de alto ancho de banda, pero también necesita consumir mucha energía; NFC por un corto alcance de estándares de tecnología de comunicación inalámbrica de alta frecuencia, para que los dispositivos puedan llevar a cabo la comunicación de corta distancia, principalmente en escenas de transmisión cifradas, pero la prevalencia actual no es alta. Este proyecto utiliza una implementación basada en teléfonos inteligentes, es decir, la configuración ascendente y descendente de los datos que desea utilizar directamente como puerta de enlace de IoT. Sin embargo, debido a las restricciones actuales de comunicación inalámbrica del teléfono inteligente, por lo que la elección de la comunicación inalámbrica entre el dispositivo sensor subyacente y el teléfono inteligente se limita a WiFi, Bluetooth y NFC, con el argumento de que estos modos de comunicación están acompañados por la mayoría de los teléfonos inteligentes, especialmente WiFi y Bluetooth, dos modos de comunicación, se puede decir que el mercado actual de casi el 100% de los teléfonos inteligentes admiten estos medios de comunicación, los usuarios pueden utilizar estos medios de comunicación directamente entre el teléfono inteligente y la comunicación del dispositivo subyacente. Por último, mediante la selección y comparación, en este proyecto se ha seleccionado la tecnología BLE en el estándar Bluetooth, tecnología bluetooth de baja energía, como base para la comunicación entre el dispositivo de detección subyacente y el teléfono inteligente en este diseño. Esto se debe

principalmente a la tecnología Bluetooth de baja potencia con bajo costo, bajo consumo de energía, conectividad rápida y alta fiabilidad y seguridad.

III. Introducción a los requisitos y las herramientas de la aplicación IoT.

Este proyecto se centra en el sistema operativo Android de la investigación e implementación de aplicaciones IoT de teléfonos inteligentes, para la aplicación Internet de las Cosas, diferentes condiciones y diferentes escenarios de aplicación, diferentes requisitos y funciones del sistema no son los mismos, por lo que el problema específico debe ser analizado. La Figura 3.1 es la composición de las aplicaciones de IoT que normalmente utilizan teléfonos inteligentes como puertas de enlace de IoT. El dispositivo de detección subyacente envía la información recopilada a la puerta de enlace de red de teléfono inteligente, la puerta de enlace de red de teléfono inteligente procesa los datos y, a continuación, los carga en la plataforma en la nube, que almacena y procesa aún más la información y los datos del dispositivo, y luego los abre a plataformas o aplicaciones de terceros. Y el usuario en la estructura del sistema principalmente con dos partes de interacción directa, una es un servicio de plataforma de terceros, y la otra es el teléfono inteligente instalado directamente una variedad de aplicaciones de terceros.

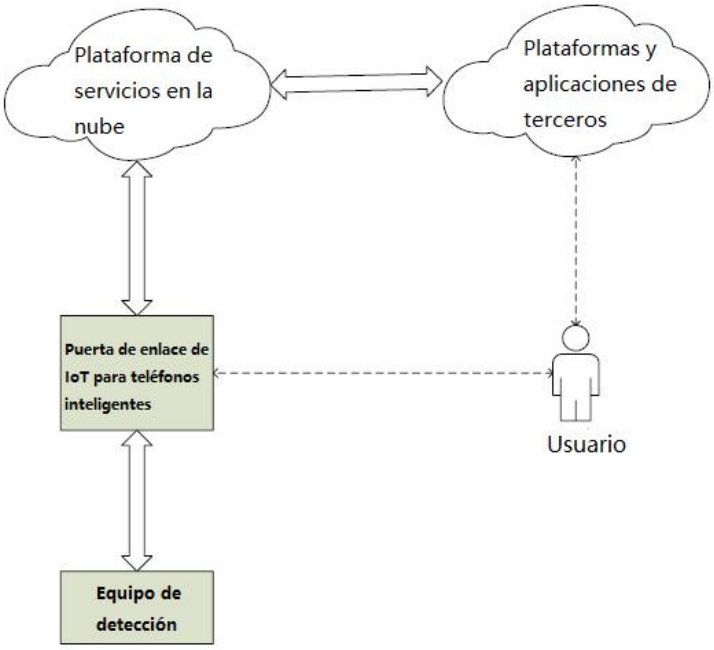

Figura 3.1 Arquitectura del sistema de aplicaciones IoT.

Los requisitos funcionales para esta aplicación de IoT basada en teléfonos inteligentes en este proyecto están relacionados principalmente con los siguientes aspectos.

(1) Este proyecto espera resolver la comunicación entre los teléfonos móviles y el hardware de IoT mediante el desarrollo de una aplicación de IoT, y que la aplicación pueda recibir y enviar datos, y que la aplicación también pueda utilizar MQTT para el funcionamiento y la gestión remotos, que se pueden utilizar para monitorear el equipo, reduciendo así el riesgo y el costo para las personas. Especialmente en la agricultura, los agricultores pueden rastrear el cambio climático, monitorear la ganadería, monitorear inventarios y analizar datos de aplicaciones centrales conectadas a múltiples dispositivos y sensores de campo.

(2) Para los dispositivos de sensores en la capa perceptiva, desarrollamos y obtenemos los valores del módulo de detección óptica y utilizamos dos medios de comunicación para lograr la interacción de datos con el dispositivo. La capa de percepción es el nivel más bajo de la arquitectura tradicional de tres niveles del Internet de las cosas y la base de todo el sistema de aplicaciones de IoT, pero hay muchas opciones diferentes en la industria en los dispositivos de detección subyacentes, la diferencia más importante es la diferencia en tecnologías de comunicación como Bluetooth, ZigBee, WiFi, Lopy, etc. Cada tecnología tiene sus propias ventajas y desventajas, y para desarrollar una arquitectura de aplicaciones de IoT, es necesario que seleccionemos dos estándares de comunicación para los dispositivos de detección subyacentes adecuados para el público en general.

En este documento se analiza y se compara diferentes tipos de puertas de enlace en diferentes escenarios, y luego se analiza la viabilidad y las ventajas de los teléfonos inteligentes como puertas de enlace de IoT en este escenario, en función de las características del entorno de red de dominio y los requisitos de la puerta de enlace. Este artículo elige usar un teléfono inteligente como puerta de enlace de IoT que permite a los usuarios ahorrar algunos costos de hardware.

3.1 Análisis del módulo de adquisición de datos

13.1.1 Protocolo del GATT

GATT Generics es una abreviatura para la comunicación entre dispositivos bluetooth conectados de baja potencia. Una vez conectados los dos dispositivos, el GATT comienza a funcionar, lo que también significa que usted tiene que completar el protocolo GAP.

GATT utiliza el protocolo ATT, que mantiene los datos de servicio y caracterización, en una tabla de búsqueda que utiliza un ID de 16 bits como índice para cada elemento.

La estructura de datos de varios niveles definida por GATT se resume brevemente como un servicio que puede contener varias entidades (Características), cada una de las cuales contiene propiedades y valores y varias descripciones.

3.1.2. Funciones en la comunicación del GATT

Desde una perspectiva del GATT, dos dispositivos conectados actúan cada uno como uno de los dos roles:

Servidor: Dispositivo que contiene datos de características leídos o escritos por el cliente del GATT.

Cliente: Un dispositivo que lee datos o escribe datos en un servidor GATT.

Los periféricos (de máquinas) actúan como servicios del GATT (Servidor), que mantiene la tabla de búsqueda ATT y la definición de servicio y caracterización. Las funciones del GATT para clientes y servidores son independientes de los roles GAP para periféricos y dispositivos centrales. Los periféricos pueden ser clientes del GATT o servidores del GATT, y los concentradores pueden ser clientes del GATT o servidores del GATT.

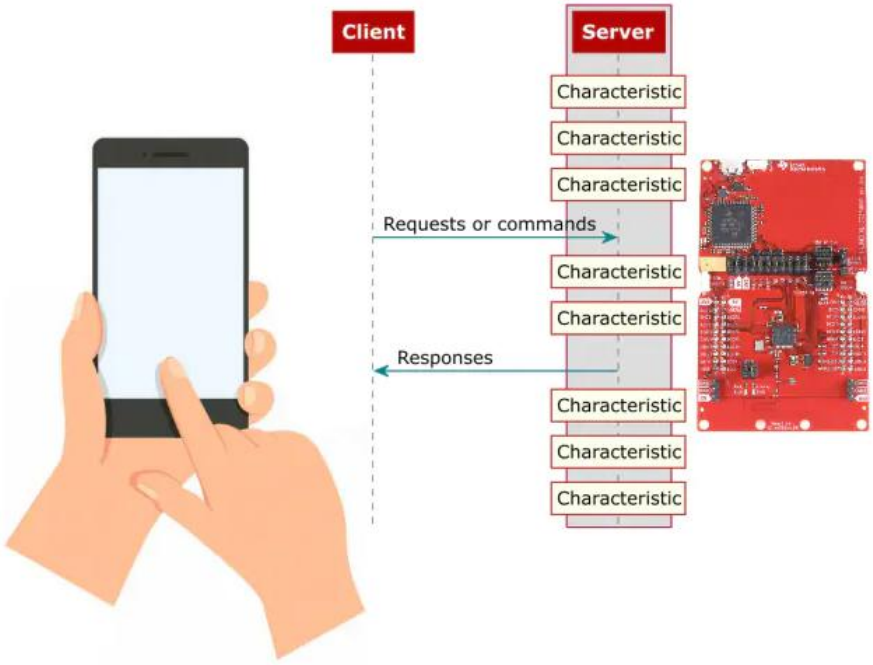

Figura 3.1.2.1 Tecnología de comunicación del GATT.

Topología Bluetooth de baja potencia

Bluetooth 4.0 BLE consiste principalmente en dos topologías de red, estrella y grupo de difusión diferentes, como se muestra en la Figura 3.1.2.2. Las estructuras de red diferentes corresponden a diferentes escenarios, y su configuración de nodos de red también es diferente.

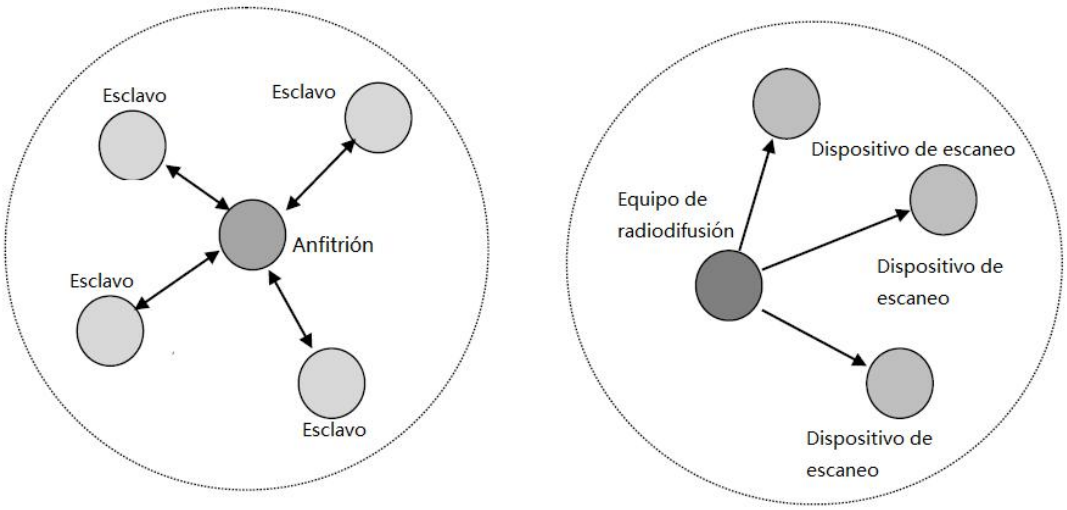

Figura 3.1.2.2 Diagrama de topología Bluetooth.

Por lo general, los nodos de red Bluetooth de baja potencia se configuran de forma diferente y de diferentes tipos, y un conjunto de nodos Bluetooth de baja potencia se pueden utilizar como servidores y clientes, así como hosts y esclavos.

3.1.3. Pila Bluetooth de baja potencia

La pila de protocolo Bluetooth de baja potencia se compone principalmente de host y controlador, en el que la parte del host se divide en tres capas, incluyendo el perfil de acceso universal y la capa de perfil de propiedad común, la capa de gestión de seguridad y protocolo de propiedad, así como el control de conexión lógica y la capa de protocolo adaptable. Al mismo tiempo, el controlador se divide en tres capas, incluida la capa de interfaz de control de host, la capa de enlace y la capa física. Su estructura específica se muestra en la Figura 3.1.2.2.

Las principales funciones de cada capa son:

(1) El perfil de acceso común y las capas comunes del perfil de propiedad definen principalmente el marco de trabajo de servicio para la siguiente capa de protocolos de propiedad.

(2) La capa de protocolo de administración de seguridad y atributos está diseñada principalmente para habilitar conexiones seguras e intercambio de datos entre otras capas y otros dispositivos, al tiempo que define cómo se asignan y emparejan las claves.

(3) El control de vínculos lógicos y la capa de protocolo adaptable se utilizan principalmente para garantizar la comunicación de datos punto a punto del dispositivo, además de proporcionar servicios de encapsulación de datos para la capa superior.

(4) La capa de interfaz de control de host es principalmente para realizar la comunicación de datos entre el host y el controlador.

(5) La capa de enlace se utiliza principalmente para controlar los cinco estados diferentes de espera del dispositivo, publicidad, escaneo, inicialización y conexión.

(6) La capa física utiliza principalmente la tecnología de salto de frecuencia adaptativa para evitar la diafonía.

3.2 Introducción de software

3.2.1 Android Studio [7]

Android Studio es la propia herramienta de desarrollo de Android de Google, creada específicamente para el desarrollo de Android, basada en IntelliJ IDEA. Esta herramienta es el entorno de desarrollo de Google, lanzado en 13 años, y ha estado en varias versiones.

Figura 3.2.1.1 Android Studio.

Android Studio ofrece:

- Soporte de construcción basado en Gradle.
- Refactorización específica de Android y correcciones rápidas.
- Herramienta tipo para capturar el rendimiento, la disponibilidad, la compatibilidad de versiones y más.
- ProGuard y las firmas de la aplicación son compatibles.
- Asistentes basados en plantillas para generar diseños y componentes de aplicaciones de Android de uso común.
- Potente editor de diseño que le permite arrastrar controles de interfaz de usuario y efectos de vista previa.

La sugerencia de código de estudio de Android es más completa que eclipse, mientras que la funcionalidad de la interfaz de usuario en tiempo real y el diseño de la interfaz de usuario de arrastrar y soltar son innovaciones y mejoras significativas. Editor de diseño: Cree una interfaz de usuario de la aplicación Android que le permita configurar rápidamente la interfaz y ajustar las propiedades visibles con el nuevo modo blueprint. En este proyecto se ha utilizado Android Studio para desarrollar Android APP.

- **Atom[8]**

Atom es un editor de texto multiplataforma en el que GitHub se especializa para programadores. Tiene una interfaz gráfica de usuario simple e intuitiva y tiene muchas características interesantes: soporte para lenguajes de programación web como CSS, HTML, JavaScript, etc. Admite macros, completa automáticamente la función de pantalla dividida e integra el administrador de archivos.

Atom se basa en Electrón (originalmente llamado Atom Shell) basado en un marco de aplicación multiplataforma que utiliza Chromium Y Node .js, y se escribe mediante CoffeeScript y Less. Atom también se puede utilizar como IDE. Sus desarrolladores lo llaman el editor de texto hacker "hackeable del siglo XXI" para el siglo XXI. A partir del 6 de mayo del 2014, los programas principales de Atom, los administradores de paquetes y el marco de programas de escritorio basado en cromo de Atom se publican bajo los términos de licencia del MIT.

Basándose en el complemento predeterminado integrado, Atom v1.5.1 admite de alguna manera los siguientes lenguajes de programación o marcado: HTML, CSS, Less, Sass, GitHub Flavored Markdown, C/C++, C#, Go, Java, Objective-C, JavaScript, JSON, CoffeeScript, Python, PHP, Ruby, Ruby on Rails, secuencia de comandos de shell, Clojure, Perl, Git, Make, lista de propiedades (Apple), TOML, XML, YAML, Bigote, Julia y SQL.

Este proyecto utiliza software de Atom para desarrollar hardware de IoT.

IV. Desarrollar procesos específicos

El nombre del software es PyBle.

4.1 Parte de Android

4.1.1. Declarar permisos

```
<utiliza-permiso android:name="android.permission.BLUETOOTH"/>
<utiliza-permiso
android:name="android.permission.BLUETOOTH_ADMIN"/>
<utiliza-permiso
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<utiliza-permiso
android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Android.permission.BLUETOOTH: Este permiso permite al programa conectarse a un dispositivo Bluetooth emparejado, solicitar una conexión/recibir o una conexión/transferir datos requiere un cambio de permiso, principalmente después del emparejamiento.

android.permission.BLUETOOTH_ADMIN: Este permiso permite al programa descubrir y emparejar dispositivos Bluetooth, que se utiliza para administrar dispositivos Bluetooth, y permite a las aplicaciones utilizar dispositivos Bluetooth nativos, principalmente para operaciones previas al emparejamiento.

Después de android.permission.ACCESS_COARSE_LOCATION y android.permission.ACCESS_FINE_LOCATION: Después de Android 6.0, ambos permisos son necesarios, y los escaneos Bluetooth alrededor de los dispositivos necesitan obtener información de ubicación difusa. Estos dos permisos pertenecen al mismo conjunto de permisos peligrosos, que deben adquirirse dinámicamente en tiempo de ejecución después de que se declaren en el archivo de manifiesto.

4.1.2. Inicialización y configuración

```
@Override
vacío protegido onCreate(Bundle savedInstanceState) {
 fenomenal. onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
BleManager. getInstance(). init(getApplication());
 BleManager. getInstance()
. enableLog(true)
. setReConnectCount(1, 5000)
. setOperateTimeout(5000);
}
```

El método de inicialización init (aplicación de aplicación) debe llamarse de antemano. Además, puede realizar algunas configuraciones personalizadas, como si desea mostrar los registros internos del marco de trabajo, el número de recombinaciones e intervalos de reconexión y el tiempo de espera del tiempo de espera de la operación.

4.1.3. Escanear periféricos

Aplicación como un dispositivo central, quiere establecer la comunicación Bluetooth con dispositivos de hardware periféricos, la premisa es obtener el objeto del dispositivo primero, la forma es escanear. Antes de llamar al método de análisis, primero debe trabajar en los siguientes preparativos.

- Determina si el dispositivo Android actual es compatible con BLE.

Después de Android 4.3, Bluetooth BLE se añadió al sistema.

```
BleManager. getInstance(). isSupportBle();
```

- Determina si Bluetooth está activado en tu dispositivo Android actual.

Puede llamar directamente al siguiente método de juicio para determinar si el nativo tiene Bluetooth activado y, si no, lanzar un mensaje al usuario.

```
BleManager. getInstance(). isBlueEnable();
```

- Activa activamente Bluetooth.

Además de determinar si Bluetooth está activado para dar al usuario un mensaje, también podemos ayudar directamente al usuario a activar el interruptor Bluetooth a través del programa, hay varias maneras de abrir: Método 1: Activar Bluetooth directamente a través del adaptador Bluetooth.

```
BleManager. getInstance(). enableBluetooth();
```

Método 2: Guía al usuario para activar Bluetooth a través de la interfaz de arranque startActivityForResult.

```
Intención = nueva
intención(BluetoothAdapter.ACTION_REQUEST_ENABLE);

startActivityForResult(intención, 0x01);
```

- Tenga en cuenta que el primer método es asincrónico y tarda un tiempo en activar Bluetooth, llamando a este método

Bluetooth no se encenderá inmediatamente. Si es necesario analizar un sucesor de este método, se recomienda mantener un subproceso de bloqueo que consulta internamente a intervalos regulares para ver si Bluetooth está activado y una pantalla externa espera a que la interfaz de usuario guíe al usuario para que espere hasta que esté activado correctamente. Con el segundo método, el usuario se guía a través de un cuadro emergente del sistema y, finalmente, una notificación de devolución de llamada en forma de onActivityResult se activa correctamente.

- 6.0 y superiores obtienen dinámicamente permisos de ubicación.

Después de activar Bluetooth, debe determinar si el dispositivo actual es 6.0 o más antes de escanear, y si es así, debe obtener dinámicamente permisos de ubicación declarados previamente en Manifiesto.

- Configurar reglas de análisis

Las reglas de análisis se pueden configurar durante 1 o más, o se pueden configurar para utilizar el valor predeterminado (escanear durante 10 segundos). Al escanear, el dispositivo escaneado se filtra según las opciones de filtrado configuradas y el resultado se devuelve al dispositivo filtrado.
El tiempo de escaneo está configurado para ser menor o igual, y se logra un

análisis infinito hasta que se llama a BleManger.getInstance(.cancelScan)) para anular el análisis.

```
BleManager. getInstance(). escanear(nuevo BleScanCallback() {
@Override
vacío público enScanStarted(éxitobooleano) {
}

@Override
vacío público enLeScan(BleDevice bleDevice) {
}

@Override
vacío público enScanning(BleDevice bleDevice) {
}

@Override
vacío público enScanFinished(Lista<BleDevice> scanResultList) {
}
});
```

Una vez completados los preparativos anteriores, puede comenzar el escaneo.

```
BleManager.getInstance().scan(new BleScanCallback() {
@Override
vacío público enScanStarted(éxito booleano) {
}

@Override
vacío público enLeScan(BleDevice bleDevice) {
}
```

```
@Override
vacío público enScanning(BleDevice bleDevice) {
}

@Override
vacío público enScanFinished(List<BleDevice> scanResultList) {
}
});
```

onScanStarted: Vuelve al subproceso principal y el parámetro indica si la operación de análisis se abrió correctamente. Debido a que Bluetooth no está activado, la última exploración no terminó y así sucesivamente, hará que el escaneo no se encienda.

onLeScan: Devolución de llamada de todos los resultados escaneados durante el análisis. Dado que el proceso de exploración y filtrado está en el subproceso de trabajo, este método también se encuentra en el subproceso de trabajo. El mismo dispositivo llevará sus propios estados diferentes (como la intensidad de la señal, etc.) en diferentes momentos, y aparecerá en este método de devolución de llamada, dependiendo de la cantidad de equipo a su alrededor y el intervalo de difusión del dispositivo periférico.

onScanning: Una devolución de llamada a todos los resultados filtrados durante el análisis. La diferencia con onLeScan es que vuelve al subproceso principal, el mismo dispositivo aparece solo una vez y el dispositivo que aparece se filtra mediante reglas de filtrado de análisis.

OnScanFinished: Una colección de todos los dispositivos escaneados y filtrados durante este período de análisis. Se remonta al subproceso principal, que es equivalente a la suma de los dispositivos onScanning.

4.1.4. Información del dispositivo

Los periféricos BLE escaneados actúan como objetos BleDevice como los objetos de celda más pequeños para operaciones posteriores. Contiene esta información en sí mismo:

String getName(): Nombre de difusión Bluetooth String.

getMac():Bluetooth Mac address.

byte?getScanRecord(): Los datos de difusión transportados cuando se escanean.

int getRssi(): La intensidad de la señal del tiempo escaneado va seguida de la conexión del dispositivo, desconexión, juicio del estado del dispositivo, operaciones de lectura y escritura, etc., utilizará este objeto. Se puede entender como el portador del dispositivo Bluetooth periférico, todo el funcionamiento del dispositivo Bluetooth periférico, a través de este objeto a realizar.

4.1.5. Conecte, desconecte y supervise el estado de la conexión

Una vez que tenga el objeto del dispositivo, puede conectarse.

```
 BleManager.getInstance().connect(bleDevice, nuevo BleGattCallback() {
@Override
vacío público enStartConnect() {
}

@Override
público void onConnectFail(excepción BleException) {
}

@Override
vacío público onConnectSuccess(BleDevice bleDevice, BluetoothGatt gatt, int status) {
}

@Override
```

```
público void onDisConnected(boolean isActiveDisConnected, BleDevice
bleDevice, BluetoothGatt gatt, int status) {
}
 });
```

OnStartConnect(): Comience a conectarse.

OnConnectFail(excepción BleException): La conexión no se realizó correctamente.

OnConnectSuccess (BleDevice bleDevice, Bluetooth Gat gatt, estado int): La conexión se realizó correctamente y se descubrió el servicio.

OnDisConnected (booleano isActiveDisConnected, BleDevice bleDevice, BluetoothGatt gatt, int status): Desconectado, por ejemplo, cuando la conexión está desconectada. Aquí puede supervisar el estado de conexión del dispositivo y, una vez desconectada la conexión, puede considerar volver a conectar el objeto BleDevice según sus propias circunstancias. Es importante tener en cuenta que hay un buen momento entre la desconexión y la reconexión, de lo contrario puede haber largos períodos de desconexión. Además, si llama al método de desconexión (BleDevice bleDevice), también se llamará al resultado de desconectar bluetooth activamente en este método, momento en el que isActiveDisConnected será true.

4.1.6. Protocolo del GATT

Las conexiones BLE se basan en el protocolo GATT (Generic Attribute Profile). GATT es una especificación común para enviar y recibir segmentos muy cortos de datos a través de una conexión Bluetooth, que se denominan atributos. Define dos dispositivos BLE que se comunican a través del servicio y el carácter. GATT es el uso del protocolo ATT(Atributo de protocolo), que mantiene los servicios, caracterizar y los datos correspondientes en una tabla de búsqueda, con la siguiente tabla de búsqueda utilizando el ID de 16 bits como índice para cada elemento.

A continuación, se destaca esta sección sobre el GATT.

En resumen, si los dispositivos centrales y los periféricos requieren una comunicación bidireccional, la única manera es establecer una conexión gatt. Cuando la conexión se realiza correctamente, se establece una conexión GATT entre el periférico y el dispositivo central.

El método connect (BleDevice bleDevice, BleGattCallback bleGattCallback) mencionado anteriormente tiene realmente un valor devuelto, que es BluetoothGatt.Por supuesto, hay otras maneras de obtener el objeto BluetoothGatt, y después de que la conexión se realiza correctamente, llame:

```
BluetoothGatt gatt = BleManager.getInstance().getBluetoothGatt(BleDevice bl
eDevice);
```

Con el objeto BluetoothGatt como puente, el dispositivo central puede obtener mucha información sobre los periféricos, así como la comunicación bidireccional.

En primer lugar, puede obtener los servicios que este dispositivo Bluetooth posee. Cada propiedad se puede definir para diferentes propósitos, a través de los cuales se puede llevar a cabo la comunicación de protocolo. El siguiente método es averiguar todos los servicios y el UUID de la característica a través de BluetoothGatt:

```
Lista<BluetoothGattService> serviceList = bluetoothGatt.getServices();
for (servicio BluetoothGattService : serviceList) {
UUID uuid_service = service.getUuid();


Lista<BluetoothGattCharacteristic> characteristicList=
service.getCharacteristics();
for(Característica BluetoothGattCharacteristic : characteristicList){
UUID uuid_chara = characteristic.getUuid();
}
 }Comunicaciones de protocolo
```

Una vez que la APP ha establecido una conexión con el dispositivo y conoce el servicio y term(que requiere la confirmación de la comunicación con el protocolo de hardware), podemos comunicarnos a través del protocolo BLE.

El puente de comunicación es principalmente a través de la característica estándar o personalizada, chino que llamamos "características". Podemos leer y escribir datos desde Feature. Esto permite la comunicación bidireccional. Desde el punto de vista de app como el dispositivo central, hay tres formas principales de comunicación comúnmente utilizadas para la interacción de datos: recibir notificaciones, escritura, lectura, además de establecer la unidad de transmisión máxima, obtener resistencia a la señal en tiempo real y otras operaciones de comunicación.

- Recibir notificaciones

Hay dos maneras de recibir notificaciones, indicar y notificar. La diferencia entre presente y notificar es que el presente está seguro de recibir datos y es probable que notifique la pérdida de datos. El dispositivo subyacente encapsula el mecanismo de respuesta, que se envía de nuevo hasta que se realiza correctamente si no recibe una respuesta del dispositivo central, mientras que notify no recibe una respuesta de los datos centrales, que puede no garantizar la exactitud de la llegada de datos y tiene la ventaja de ser rápido. Por lo general, cuando los periféricos necesitan enviar constantemente datos a la APP, como cambios de presión en el medidor de presión arterial durante la medición, la transmisión de datos en tiempo real durante la supervisión, en este caso frecuente, se da prioridad al formulario de notificación. Cuando solo se necesita enviar una pequeña e importante pieza de datos a la APP, se da prioridad al formulario de presentación. Por supuesto, desde una perspectiva de desarrollo de Android, si la colocación de hardware ha tenido en cuenta protocolos maduros y métodos de envío, todo lo que tenemos que hacer es conectar de acuerdo con cómo se envían los datos que configuramos.

- *Notificación abierta*

```
BleManager.getInstance().notify(
 bleDevice,
 uuid_service,
 uuid_characteristic_notify,
 nuevo BleNotifyCallback() {
@Override
```

```
vacío público enNotifySuccess() {
La operación de notificación abierta se realizó correctamente
}
@Override
vacío público enNotifyFailure(excepción BleException) {
Error en la operación de notificación abierta
}
@Override
vacío público onCharacteristicChanged(byte[] data) {
Después de abrir la notificación, los datos enviados desde el dispositivo aquí.
}
 });
```

- Desactivar la notificación

```
BleManager.getInstance().stopNotify(uuid_service,uuid_characteristic_notif
y);
```

- Abra el

```
BleManager.getInstance().indicate(
 bleDevice,
 uuid_service,
 uuid_characteristic_indicate,
 nuevo BleIndicateCallback() {
@Override
vacío público onIndicateSuccess() {
La operación de notificación abierta se realizó correctamente
}

@Override
vacío público onIndicateFailure(excepción BleException) {
Error en la operación de notificación abierta
```

```
}


@Override
vacío público onCharacteristicChanged(byte[] data) {
Cuando la notificación está activada, los datos enviados por el dispositivo aparecen aquí
}
 });
```

- Apague el

```
BleManager.getInstance().stopIndicate(uuid_service,uuid_characteristic_indic
ate);
```

La operación de notificación aquí utiliza dos parámetros clave, uuid_service y uuid_characteristic_notify(o uuid_characteristic_indicate), que son los servicios y características mencionados anteriormente, que se representan como cadenas y son insensibles a mayúsculas y minúsculas.

- Leer y escribir

```
BleManager.getInstance().read(
 bleDevice,
 uuid_service,
 uuid_characteristic_read,
 nuevo BleReadCallback() {
@Override
vacío público enReadSuccess(byte[] data) {
Leer datos de valor de la característica correctamente
}


@Override
público void onReadFailure(excepción BleException) {
```

```
No se puede leer los datos de los valores característicos
}
 });

BleManager.getInstance().write(
 bleDevice,
 uuid_service,
 uuid_characteristic_write,
 datos
 nuevo BleWriteCallback() {
@Override
vacío público enWriteSuccess(int current, int total, byte[] justWrite) {
Enviar datos al dispositivo con éxito (en el caso del envío de subpaquetes, puede verificar el progreso del envío a través de los parámetros devueltos en el método)
}

@Override
público void onWriteFailure(excepción BleException) {
Error al enviar datos al dispositivo
}
 });
```

Cuando se envían datos BLE entre sí, se pueden enviar hasta 20 bytes a la vez. Si necesita enviar más de 20 bytes de datos, hay dos maneras, una es intentar activamente ampliar la MTU, y la otra es subcontratar la transferencia. El método write del marco de trabajo se subcontrata de forma predeterminada cuando se encuentran más de 20 bytes de datos.

- Fije la unidad de transmisión máxima MTU

```
BleManager.getInstance().setMtu(bleDevice, mtu, new BleMtuChanged
Devolución de llamada() {
@Override
```

```
void público onSetMTUFailure(excepción BleException) {
 Sin se pudo establecer MTU
}

@Override
vacío público enMtuChanged(int mtu) {
 La MTU establece correctamente y se obtiene el valor de MTU admitido por la transmisión del dispositivo real
}
});
```

- Obtenga la intensidad de señal en tiempo real del dispositivo RSSI

```
BleManager.getInstance(). readRssi(
 bleDevice,
nuevo BleRssiCallback(){

@Override
vacío público enRssiFailure(excepciónBleException) {
No se pudo leer la intensidad de la señal del dispositivo
}

@Override
vacío público enRssiSuccess(int rssi){
La intensidad de la señal del dispositivo de lectura fue exitosa
}
 });
```

4.2 Desarrollo de hardware

4.2.1 El desarrollo

En la Figura 4.2.1.1, la línea 9 declara el archivo 12 para establecer una

conexión global para Bluetooth, la línea 16 activa la función número 1 para la conexión y desconexión exitosa, y la línea 27 recibe datos y cambia el color RGB.

```
1  import time
2  import ujson
3  import pycom
4  from network import Bluetooth
5  from SI7006A20 import SI7006A20   #濕度传感器 濕度传感器
6  from pysense import Pysense
7  from LIS2HH12 import LIS2HH12
8  from MPL3115A2 import MPL3115A2,ALTITUDE,PRESSURE
9  from LTR329ALS01 import LTR329ALS01 #光感
10
11
12 BLEConnected = False
13
14 py = Pysense()
15
16 def connectionCallback(e):
17     events = e.events()
18     global BLEConnected
19     if events & Bluetooth.CLIENT_CONNECTED:
20         BLEConnected = True
21         print("LJCG")
22     elif events & Bluetooth.CLIENT_DISCONNECTED:
23         BLEConnected = False
24         print("DKLJ")
25
26
27 def char1_cb_handler(chr, data):
28     events, value = data
29     if events & Bluetooth.CHAR_WRITE_EVENT:
30         print("GBRGB")
31         pycom.rgbled(int.from_bytes(bytearray(value), 'big'))
32     elif events & Bluetooth.CHAR_READ_EVENT:
33         print("Read Request")
34
```

Figura **4.2.1.1** Código de desarrollo parte 1.

La línea 40 de la Figura 4.2.1.2 obtiene el valor de la temperatura y almacena el valor en la función wendu. La línea 47 establece el nombre de Ble a LoPy uuid a 12345 52 comienza a empujar los datos de Bluetooth. La línea 60 está declarando dos servicios de características.

```
40  mp = MPL3115A2(py,mode=ALTITUDE)
41  wendu = str(mp.temperature())
42
43
44
45  pycom.heartbeat(False)
46
47  Bluetooth().set_advertisement(name='LoPy', service_uuid=12345)
48  # Callback event for on connected and on disconnected
49  # Llamada de retorno sobre los eventos connectado y desconectado
50
51  # Triggers Bluetooth.NEW_ADV_EVENT, Bluetooth.CLIENT_CONNECTED, or Bluetooth.CLIENT_DISCONNECTED
52  Bluetooth().callback(trigger=Bluetooth.CLIENT_CONNECTED |
53                       Bluetooth.CLIENT_DISCONNECTED, handler=connectionCallback)
54
55  # Start sending BLE advertisements.
56  # Inicia los mensajes de anuncio de BLE
57  Bluetooth().advertise(True)
58
59
60
61  # Create and init BLE service and declare 2 characteristics
62  srv = Bluetooth().service(uuid=12345, isprimary=True, nbr_chars=2, start=True)
63
64
```

Figura **4.2.1.2** Código de desarrollo parte 2.

En la figura 4.2.1.3 la fila 66 es una función activada por el envío de datos 69 la fila 71 es una función activada por la recepción de características.

```
64  # Characteristics Bluetooth.PROP_NOTIFY,
65  # Info https://www.oreilly.com/library/view/getting-started-with/9781491980550/ch04.html
66  char1 = srv.characteristic(uuid=54321, properties=Bluetooth.PROP_NOTIFY, value=ujson.dumps({}))
67
68
69  char2 = srv.characteristic(uuid=64321, value=0xff00)
70
71  char1_cb = char2.callback(
72      trigger=Bluetooth.CHAR_WRITE_EVENT, handler=char1_cb_handler)
73
74
75  while True:
76      time.sleep(10)
77      json = ujson.dumps({"h": wendu})
78      print("json  =  " + json)
79      #wd = wendu
80      if BLEConnected:
81          char1.value(json)
82
```

Figura **4.2.1.3** Código de desarrollo parte 3.

4.3 Desarrollo de hardware Mqtt

En la figura 4.3.1 9 archivo de declaración de comportamiento 12 comportamiento valores de temperatura exportados 19 comportamiento perfil de red WIFI 23 comportamiento parámetros de la plataforma IoT.

```
from network import WLAN      # For operation of WiFi network
import time                   # Allows use of time.sleep() for delays
import pycom                  # Base library for Pycom devices
from LTR329ALS01 import LTR329ALS01 #光感
from pysense import Pysense
from umqtt import MQTTClient  # For use of MQTT protocol to talk to Adafruit IO
import ubinascii              # Needed to run any MicroPython code
import machine                # Interfaces with hardware components
import micropython            # Needed to run any MicroPython code


py = Pysense()
mp = MPL3115A2(py,mode=ALTITUDE)
wendu = str(mp.temperature())


# BEGIN SETTINGS
# Wireless network
WIFI_SSID = "DIGIFIBRA-NXAP"
WIFI_PASS = "P9P6tzQbbT" # No this is not our regular password. :)

# Adafruit IO (AIO) configuration
AIO_SERVER = "io.adafruit.com"
AIO_PORT = 1883
AIO_USER = "xiaohangge"
AIO_KEY = "aio_dgEY92iAbuJrH5gl22PbpdwbDVNu"
AIO_CLIENT_ID = ubinascii.hexlify(machine.unique_id())  # Can be anything
AIO_CONTROL_FEED = "xiaohangge/feeds/lights"
AIO_RANDOMS_FEED = "xiaohangge/feeds/randoms"

# END SETTINGS
```

Figura **4.3.1** Código de desarrollo parte 4

En la Figura 4.3.2 39 comportamiento el latido del corazón de la placa de desarrollo binaria parpadea 40 comportamiento detener el multiproceso y encender la luz naranja para la alerta de error 49 comportamiento conectar wifi 52 no hay código de pausa hasta que la conexión wifi sea exitosa 56 comportamiento wifi conectado con éxito imprimir conectar wifi y encender la luz naranja para la alerta.

```
# RGBLED
# Disable the on-board heartbeat (blue flash every 4 seconds)
# We'll use the LED to respond to messages from Adafruit IO
pycom.heartbeat(False)
time.sleep(0.1) # Workaround for a bug.
                # Above line is not actioned if another
                # process occurs immediately afterwards
pycom.rgbled(0xff0000)  # Status red = not working

# WIFI
# We need to have a connection to WiFi for Internet access
# Code source: https://docs.pycom.io/chapter/tutorials/all/wlan.html

wlan = WLAN(mode=WLAN.STA)
wlan.connect(WIFI_SSID, auth=(WLAN.WPA2, WIFI_PASS), timeout=5000)

while not wlan.isconnected():    # Code waits here until WiFi connects
    machine.idle()


print("Connected to Wifi")
pycom.rgbled(0xffd7000) # Status orange: partially working

# FUNCTIONS
```

Figura **4.3.3** Código de desarrollo parte 5.

En la Figura 4.3.4 actúe la función de activación del control del interruptor 82 envíe la función de temperatura a la función some_number 83 actúe imprima el valor de la función de temperatura 84 línea determine si el envío fue exitoso 93 actúe conecte a la plataforma Adafruit IO usando el protocolo MQTT.

```
# Function to respond to messages from Adafruit IO
def sub_cb(topic, msg):          # sub_cb means "callback subroutine"
    print((topic, msg))          # Outputs the message that was received. Debugging use.
    if msg == b"ON":             # If message says "ON" ...
        pycom.rgbled(0xFF00FF)   # ... then LED on
    elif msg == b"OFF":          # If message says "OFF" ...
        pycom.rgbled(0x000000)   # ... then LED off
    else:                        # If any other message is received ...
        print("Unknown message") # ... do nothing but output that it happened.



    some_number = wendu
    print("Sending data:  ".format(some_number, AIO_RANDOMS_FEED), end='')
    try:
        client.publish(topic=AIO_RANDOMS_FEED, msg=str(some_number))
        print("DONE")
    except Exception as e:
        print("FAILED")
    finally:
        last_random_sent_ticks = time.ticks_ms()

# Use the MQTT protocol to connect to Adafruit IO
client = MQTTClient(AIO_CLIENT_ID, AIO_SERVER, AIO_PORT, AIO_USER, AIO_KEY)

```

Figura **4.3.4** Código de desarrollo parte 6.

En la Figura 4.3.5, la entrega de la suscripción de 96 actos está suscrita con éxito con luces verdes que muestran el ciclo de 103 actos enviando datos a la plataforma Adafruit IO.

```
# Subscribed messages will be delivered to this callback
client.set_callback(sub_cb)
client.connect()
client.subscribe(AIO_CONTROL_FEED)
print("Connected to %s, subscribed to %s topic" % (AIO_SERVER, AIO_CONTROL_FEED))

pycom.rgbled(0x00ff00) # Status green: online to Adafruit IO

try:                      # Code between try: and finally: may cause an error
                          # so ensure the client disconnects the server if
                          # that happens.
    while 1:              # Repeat this loop forever
        client.check_msg()# Action a message if one is received. Non-blocking.
        send_random()     # Send a random number to Adafruit IO if it's time.
finally:                  # If an exception is thrown ...
    client.disconnect()   # ... disconnect the client and clean up.
    client = None
    wlan.disconnect()
    wlan = None
    pycom.rgbled(0x000022)# Status blue: stopped
    print("Disconnected from Adafruit IO.")
```

Figura **4.3.5** Código de desarrollo parte 7

V. Resultados

5.1 Presentación de los resultados finales

En este trabajo, la aplicación IoT es estudiada y analizada en detalle desde el diseño hasta el desarrollo, y la aplicación disponible se hace de acuerdo a los requisitos de diseño, como la Figura 5.1.1 muestra la página de inicio, el usuario puede hacer una elección de BLE o MQTT para conectarse.

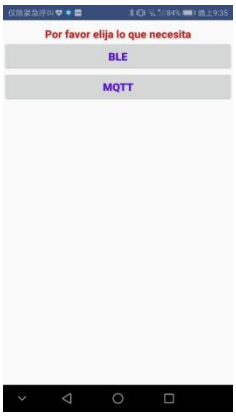

Figura **5.1.1** Aplicación desarrollada parte 1.

Toque en la conexión BLE y encienda la exploración en la parte superior, si es normal Bluetooth mostrará o sólo mostrará el nombre de Bluetooth de baja potencia, haga clic en la conexión aparecerá en la parte inferior para desconectar y entrar como se muestra en la Figura 5.1.2

Figura **5.1.2** Aplicación desarrollada parte 2.

Después de conectarse, como se muestra en la figura 5.1.3, puede seleccionar los servicios adquiridos.

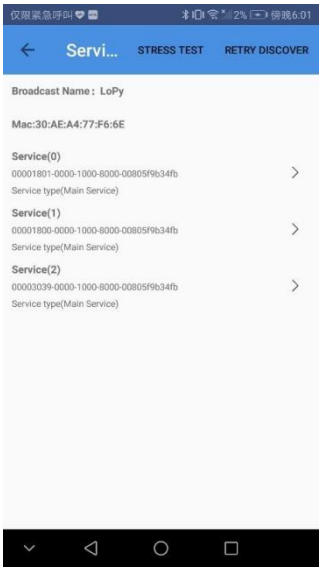

Figura **5.1.3** Aplicación desarrollada parte 3.

Como en la Figura 5.1.4 abrimos la página de recepción de mensajes y el código recibido es el hexadecimal convertido.

Figura **5.1.4** Aplicación desarrollada parte 4.

El mensaje real recibido es el mensaje escrito dentro de la figura 5.1.5{}.

```
json  =  {"h": "33.625"}
json  =  {"h": "33.625"}
json  =  {"h": "33.625"}
json  =  {"h": "33.625"}
json  =  {"h": "33.625"}
json  =  {"h": "33.625"}
json  =  {"h": "33.625"}
json  =  {"h": "33.625"}
json  =  {"h": "33.625"}
json  =  {"h": "33.625"}
```

Figura **5.1.5** Mensaje recibido.

La figura 5.1.6 muestra el envío de datos que pueden cambiar las luces LED del hardware IoT 5.1.7 muestra el hardware cambiado.

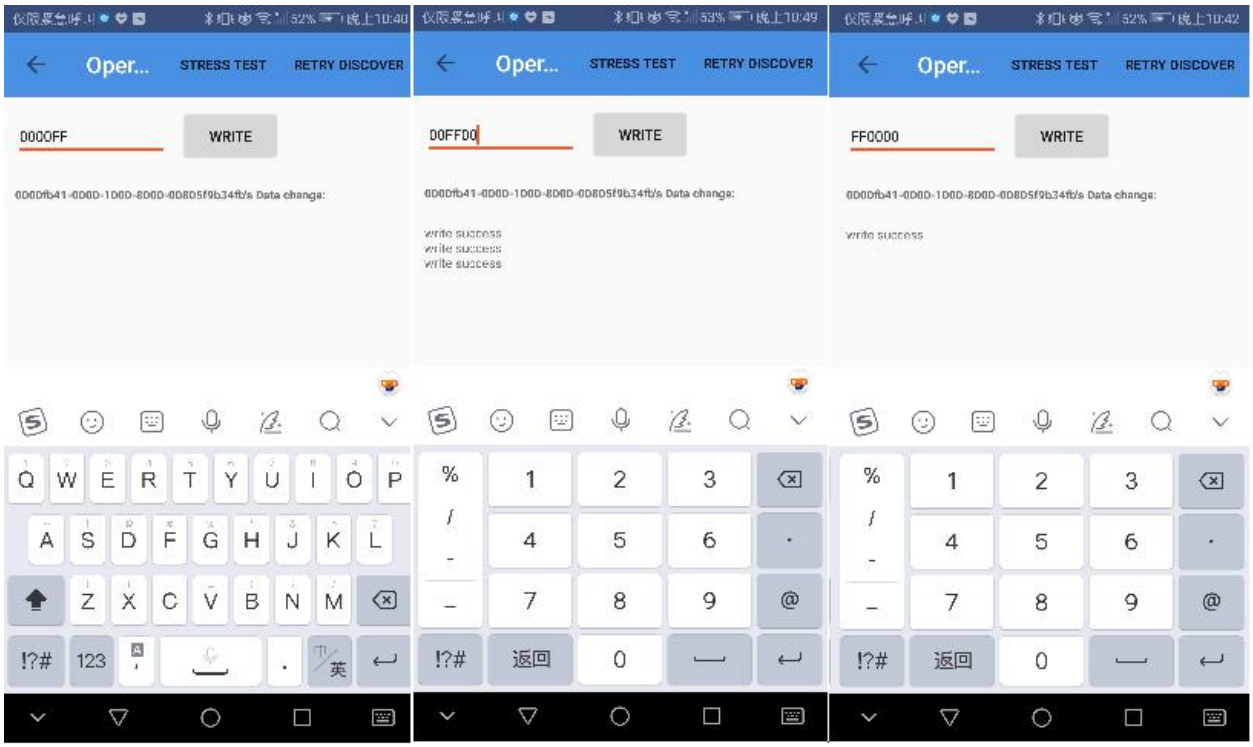

Figura **5.1.6** Envío de datos.

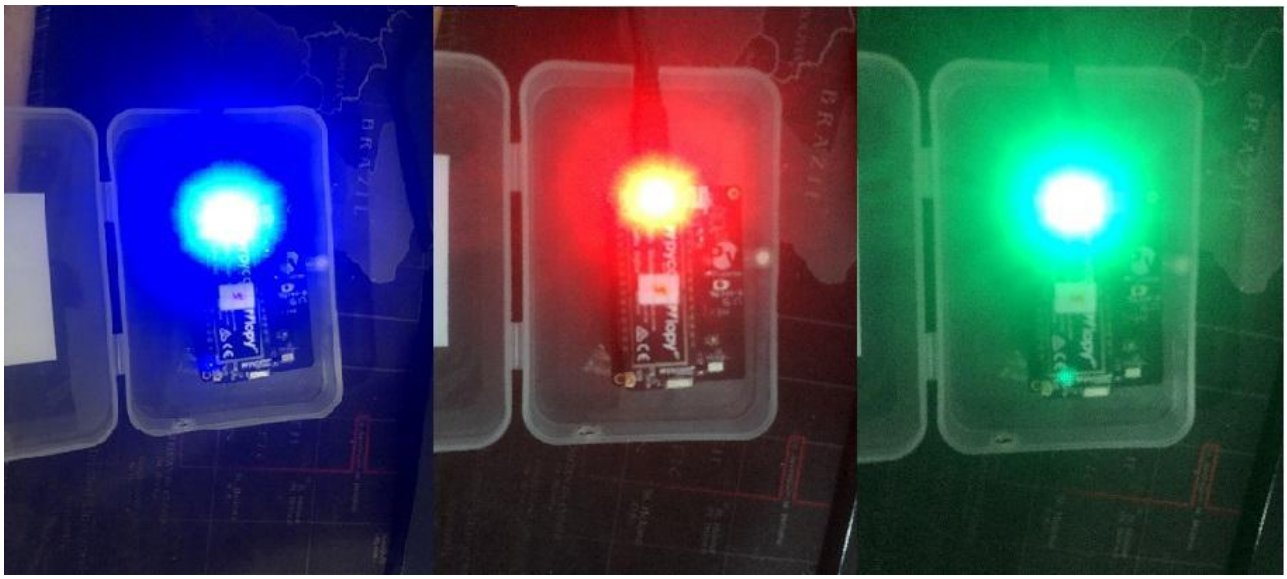

Figura **5.1.7** Control de las luces LED.

La figura 5.1.8 muestra el efecto de abrir MQTT, que le pedirá que introduzca su cuenta y contraseña.

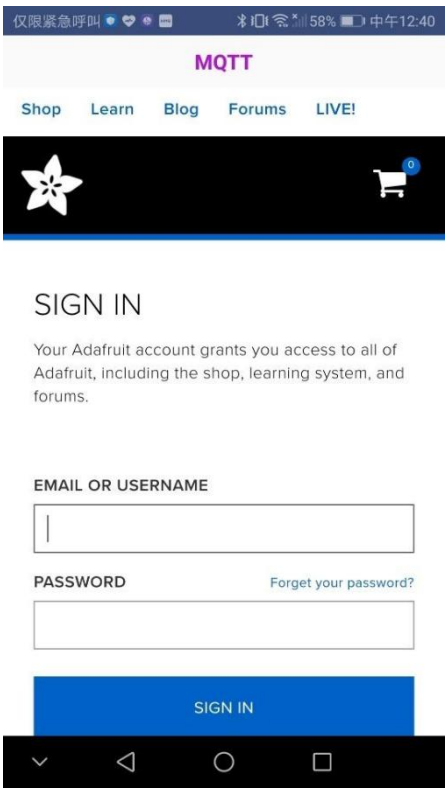

Figura 5.1.8 Ventana para inicializar MQTT.

Después de iniciar la sesión en la configuración se puede ver un botón de interruptor de un gráfico de datos, el interruptor puede controlar el interruptor de la luz LED en la presentación de Magenta LED debajo de la línea de gráfico para los datos recibidos dedo toque puede ver los datos detallados, como el tiempo y el valor.

Figura 5.1.9 Resultados finales.

Hasta ahora hemos completado todo el desarrollo del hardware de la IO, este documento se basa en la recolección de datos de hardware de la IO y la transmisión para el diseño y la implementación, el documento presenta las necesidades y la situación actual en la tecnología de la IO, lo que lleva al desarrollo de un sistema de IO APP fácil de usar, desde el desarrollo de hardware para el desarrollo de software, a través de la tecnología de microcontroladores, la tecnología Bluetooth, la tecnología de comunicación de red inalámbrica, la plataforma Android a los datos MQTT. El uso del diseño para lograr la conexión más comúnmente utilizado dos para enviar el camino, con alta escalabilidad, fácil de desarrollar, y una variedad de estilos de comunicación, y finalmente pasó la prueba. El sistema implementa la adquisición de datos en la plataforma del microcontrolador y la función de envío de datos en combinación con el módulo Bluetooth y el módulo MQTT, la función de control y visualización de datos de este sistema en la plataforma del teléfono móvil Android, y la función de almacenamiento y procesamiento de datos en la plataforma del servidor.

Las principales conclusiones de este documento son las siguientes.

（1） De acuerdo con el diseño del esquema del sistema, se realiza la función de recogida y envío de datos del extremo de recogida del sistema, principalmente el microcontrolador recoge y envía los datos del sensor, recibe los datos en serie del microcontrolador a través del modo Bluetooth y los envía al extremo del teléfono móvil Android; se realiza el desarrollo del programa cliente del teléfono móvil, principalmente la realización de cada módulo funcional del extremo del teléfono móvil.

（2） De acuerdo con el diseño de la comunicación de la parte del sistema, la comunicación entre el front-end de recogida y el cliente del teléfono móvil se realiza a través de Bluetooth, y la comunicación entre el cliente del teléfono móvil y el extremo de servicio de datos se realiza a través de la red inalámbrica.

（3） Mediante el uso de la tecnología Bluetooth y de la red inalámbrica, el sistema realiza la función de monitorización de datos del equipo de recolección a diferentes distancias, enviando algunos datos, incluyendo el uso de rango cercano del módulo Bluetooth para recibir los datos del sensor, y el uso remoto del módulo MQTT para recibir los datos del sensor.

El desarrollo de este trabajo puede superar el inconveniente de las ubicaciones múltiples, la recogida trans-regional o local sin soporte de red, facilitando el control inteligente de diferentes rangos. Los datos pueden controlarse fácilmente conectándose directamente a un microcontrolador local, o a un servidor de datos remoto a través de un smartphone móvil, y los datos pueden trazarse y analizarse directamente mediante un software de análisis gráfico en el teléfono móvil, lo que permite al usuario visualizar y comprender los datos.

El servidor de datos está basado en Adafruit IO y es totalmente abierto para su uso multiusuario. Por supuesto, debido a las limitaciones de tiempo de investigación y conocimientos personales, todavía hay algunas deficiencias en este trabajo, la parte de adquisición de front-end del sistema no implica la adquisición de sensores no es tan diversa y completa, el sistema no es tan perfecto en la aplicación de los detalles del diseño. En el futuro, podemos enriquecer aún más los dispositivos de detección en el front-end, ampliar las funciones del cliente Android, y ampliar los servicios del servidor, para que el sistema de monitorización diseñado en este trabajo tenga más valor de aplicación, y para ello es necesario seguir investigando y estudiando.

VI. Conclusiones

Durante el tiempo de desarrollo se han presentado algunos inconvenientes, como primer punto tenemos el problema en la lectura de los datos, problemas en la ejecución del programa obteniendo incluso problemas de hardware, inconvenientes con las actualizaciones del hardware, ya que no son amigables para usuarios nuevos, problemas con la comunicación específicamente con el WiFi, así como problemas en la actualización de la nueva versión de Atom, esta última porque no es compatible con la versión antigua de la consola Pymark, además para encontrar la causa se ha realizado una reinstalación de los plugin lo cual ha costado mucho tiempo. Al final, se han presentado soluciones muy sencillas a todos estos inconvenientes, en algunos de estos problemas se ha buscado ayuda en foros e incluso se ha requerido ayuda al profesor.

Se han cumplido todos los objetivos planteados para este proyecto, aunque en el proceso inicial, específicamente en la creación se presentaron muchos problemas, como algunos comandos de Python los cuales no son compatibles en el hardware ESP32, y que no permite ejecutar dos archivos al mismo tiempo. Además, el sensor no es muy sensible, puede que exista un cierto

retraso. Se recomienda realizar un mejor aprendizaje y mejorar los conocimientos para el uso correcto y óptimo de la plataforma hardware, con la finalidad de mejorar y no tener problemas. En el proceso del uso de la red de nodos, la construcción fácil de varios procesos permite que este módulo sea el más importante de todo el proyecto. Al utilizar el paquete Atom de Pymark se observó mejoras muy significativas en el desarrollo del proyecto, además que es muy amigable para desarrolladores nuevos, también que permite realizar directamente la depuración y ejecutar en el hardware sin la necesidad de cargar todos los datos.

VII. Referencias

[1] https://en.wikipedia.org/wiki/Android_(operating_system)

[2] https://en.wikipedia.org/wiki/Ios

[3] https://en.wikipedia.org/wiki/BlackBerry

[4] https://en.wikipedia.org/wiki/Windows_Mobile

[5] https://en.wikipedia.org/wiki/Symbian

[6] https://en.wikipedia.org/wiki/Windows_Phone

[7] https://developer.android.com/studio/intro

[8] https://flight-manual.atom.io/getting-started/sections/why-atom/

Fuentes de información online

Documentación de ayuda de Pycom

https://docs.pycom.io/

Lapágina oficial de github en GitHub

https://github.com/pycom

Protocolo GATT

https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html

https://docs.pycom.io/firmwareapi/pycom/network/bluetooth/gatt/

Protocolos de Internet de las cosas

https://developer.ibm.com/articles/iot-lp101-connectivity-network-

protocols/

Hoja de datosLopy4

https://docs.pycom.io/gitbook/assets/specsheets/Pycom_002_Specsheets_LoPy4_v2.pdf

Hoja de datosPysense

https://docs.pycom.io/gitbook/assets/pysense-specsheet.pdf

Un estudio ndroid

https://developer.android.com/guide

Referencia en el Texto de Referenciaseneltexto

Fuentes de información online

https://github.com/zhongyuan9/TFM