UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

Colegio de Ciencia y Tecnología

Desarrollo de un Gateway para Internet o Things implementado en la tarjeta de experimentación Raspberry Pi

Trabajo Recepcional

Que para obtener el título de:
Licenciado en Ingeniería en Sistemas Electrónicos y de Telecomunicaciones

P r e s e n t a:

Edgar David Zavala Rivera

D i r e c t o r:
M. en C. Joel Yazbek Buendía Gómez

Ciudad de México, junio de 2021.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

Resumen

Del trabajo recepcional de **Edgar David Zavala Rivera**, presentado como requisito parcial para la obtención del grado de **LICENCIADO EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS Y DE TELECOMUNICACIONES**, Ciudad de México, Julio 2019

DESARROLLO DE UN GATEWAY PARA INTERNET OF THINGS IMPLEMENTADO EN LA TARJETA DE EXPERIMENTACIÓN RASPBERRY PI

En este documento se describe el desarrollo de un Gateway para Internet of Things en el cual se habilitaron cuatro protocolos de comunicación para ser utilizado en una red de sensores que recolecten datos de diferentes protocolos de comunicación y lanzarlos a un servidor FTP

Este prototipo presenta un método de almacenamiento de datos básico para así obtener un registro de lo que llega por cada una de sus interfaces de los cuales fue dotado y lo transmite a un servidor FTP programado en Python en un Host, de esta manera lo convierte en un prototipo útil para integrarlo a la estructura Internet of Things.

Los resultados obtenidos de las pruebas realizadas en conjunto con sus protocolos de comunicación, muestran que el prototipo es posible trabajarlo como un Gateway para esta nueva tendencia (Internet of Things) que se le atribuye a la red de redes (Internet), y que la cual ya no es solo una sugerencia para un mundo digital sino una realidad.

Palabras clave: Red, IoT, Gateway, IEEE 802.11, Bluetooth, XBee, Radiofrecuencia, MQTT, FTP.

Dedicatoria

A ustedes mamá y papá, por apoyarme, porque jamás bajaron la guardia a pesar de mis tropiezos y mis errores, ustedes siempre han querido lo mejor para mí y mis hermanas, por dármelo todo sin esperar algo a cambio; todo su trabajo, esfuerzo y dedicación está aquí, porque ahora no solo soy ingeniero sino una persona que puede contribuir positivamente a esta sociedad.

A Víctor Segura Valencia "EL PAPI" porque también fuiste uno de los grandes pilares en mi vida, porque tú, tus clases de Matemáticas, tu amistad y paciencia fueron factores importantes para tomar la decisión de ser ingeniero; tu amigo mío fuiste el mejor forjador de ingenieros.

A mis amigos que encontré en la carrera, algunos los frecuento a otros ya no tanto, incluso unos ya no están aquí, no obstante les digo que en mí vive lo mejor de ustedes, porque no solo compartí experiencias, sino también aprendí de sus virtudes y con ello logré corregir mis errores, así juntos logramos hacer un cambio en nuestra persona y en nuestra vida, como decía mi mejor maestro, compañero y amigo Víctor Segura Valencia (El Papi) "No es fácil ser amigo",

Demasiada gente crece en la periferia creyendo que ellos no son los que cambiarán el mundo

Anónimo.

Agradecimientos

A la Universidad Autónoma de la Ciudad de México, por abrir sus puertas y darme la oportunidad de estudiar una carrera con alto nivel educativo, porque fue ella quien me abrigó cuando más lo necesitaba. Además no solo me ofreció una carrera sino me puso a prueba tanto en mí vida diaria, como académicamente y como persona, sin embargo nunca nos rendimos y ahora puedo decir que mi compromiso es poner su nombre en alto allá afuera, agradezco que ella creyó en mí.

A mis profesores con los que aprendí en la carrera y con los que no conviví también, porque sin ustedes este proyecto llamado UACM no fuera posible, pues ustedes creen en nosotros, en nuestras habilidades y corrigen nuestros defectos, siempre dándonos una lección tanto académicamente como de vida.

Contenido

Capítulo I, Esquema para desarrollo del proyecto. 1
1.1 Introducción 1
1.2 Planteamiento del problema 2
1.3 Justificación 3
1.4 Objetivo general 3
1.5 Objetivos particulares 4
1.6 Metodología 5
Capítulo II, Antecedentes. 6
2.1 Internet y su tendencia 6
2.2 Internet of Things y la revolución 4.0 8
2.4 Gateway IoT 9
2.5 Arquitectura IoT 10
2.6 Cloud Computing 11
2.7 Fog Computing. 11
2.6 Raspberry 12
2.7 NodeMCU 12
2.8 Arduino UNO 13
2.9 XBee 13
2.10 Bluetooth 14
2.11 Comunicación por radiofrecuencia 15
2.12 MQTT 16
2.13 Protocolo FTP (File Transfer Protocol) 18
Capítulo III, Desarrollo del prototipo. 18
3.1 Configuración e implementación del Gateway en Raspberry Pi 18
3.2 Creación de Access Point 24
3.3 Prueba transmisión de datos por MQTT 26
3.4 Habilitación de protocolo XBee 32
3.5 Habilitación del protocolo Bluetooth 38
3.6 Habilitación del protocolo por Radiofrecuencia 41
3.7 Transmisión de Datos Gateway-Servidor 44
Conclusiones 49

Anexo 52

Anexo 1.1 Eliminación de software innecesario 52
Anexo 1. 2 Instalación de DHCP y Hostapd 53
Anexo 1.3 Configuración de red inalambrica 54
Anexo 1.4 Configuracion del SSID Password para red inalámbrica 55
Anexo 1.5 Automatización para habilitación de WLAN0 55
Anexo 1.6 Actualización del HOSTAPD 57
Anexo 1.7 Cambio de versión del Kernel 58
Anexo 1.8 Instalación del driver para Red inalambrica 59
Anexo 1.9 Instalación de Mosquitto 60
Anexo 1.10 Raspberry-XBee 61
Anexo 1.11 Raspberry-Bluetooth 62
Anexo 1.12 Raspberry-ESP8266 62
Anexo 1.13 Arduino-Raspberry por XBee 64
Anexo 1.14 Arduino-Raspberry RF 65
Anexo 1.15 Raspberry-FTP 66
Referencias 68

Índice de ilustraciones

Ilustración 1 Esquema de proyecto 4
Ilustración 2 Arquitectura IoT 10
Ilustración 3 Raspberry Pi 2B+ 12
Ilustración 4 Placa NodeMCU 13
Ilustración 5 Topologías XBee 14
Ilustración 6 Logotipo Bluetooth 15
Ilustración 7 Estructura de TOPICS 17
Ilustración 8 Representación de TOPIC 17
Ilustración 9 software de preparación y respaldo de SO 19
Ilustración 10 opciones de configuración 20
Ilustración 11 habilitar puerto 22 20
Ilustración 12 Herramientas para la comunicación remota hacia Raspberry 21
Ilustración 13 Herramientas para la consulta de direcciones IP 22
Ilustración 14 Consulta de direcciones IP 22
Ilustración 15 Consulta de puertos habilitados 23

Ilustración 16 Inicio de sesión para Raspberry Pi, desde PUTTY.....23
Ilustración 17 Conexión remota desde terminal 24
Ilustración 18 Dirección IP para wlan0 25
Ilustración 19 Raspberry con tarjeta de red inalámbrica y conexión con cable UTP 26
Ilustración 20 Confirmación de conexión WiFi 26
Ilustración 21 Prueba de datos enviados por MQTT desde NodeMCU a Raspberry 28
Ilustración 22 Diagrama de función Callback 30
Ilustración 23 Diagrama de función Reconnect.....30
Ilustración 24 Diagrama de programa general.....31
Ilustración 25 Esquema de comunicación ESP y Raspberry 31
Ilustración 26 Conexión de Raspberry con radio XBee 33
Ilustración 27 Primera prueba de conexión 33
Ilustración 28 Mensaje enviado desde la terminal de XCTU y mensaje recibido en la terminal de Raspberry.....34
Ilustración 29 Esquema de comunicación con Xbee-Raspberry 35
Ilustración 30 Programa para comunicación por XBee 35
Ilustración 31 Ensamble de Arduino con Shield XBee 36
Ilustración 32 Diagrama para la recopilación de datos, python.....36
Ilustración 33 Mensaje desde Arduino-XBee a Raspberry.....37
Ilustración 34 Datos guardados en archivo .txt.....38
Ilustración 35 Diagrama para la conversión de serial a USB 38
Ilustración 36 A la izquierda HC-05 y a la derecha convertidor Serial-USB.....39
Ilustración 37 Diagrama de comunicación serial Bluetooth.....39
Ilustración 38 Primera prueba para el envío de datos por protocolo Bluetooth 40
Ilustración 39 Primera prueba para el almacenamiento, para Bluetooth 40
Ilustración 40 Esquema Arduino - Raspberry.....41
Ilustración 41 Contenido del repositorio WiringPi.....41
Ilustración 42 Error de recepción de dato y voltaje emitido en puerto GPIO 42
Ilustración 43 Diagrama para transmisión de datos por RF.....43
Ilustración 44 Prueba del dato recibido con el módulo de radiofrecuencia.....44
Ilustración 45 Esquemas Cliente-Sevidor, Python 45
Ilustración 46 Servidor Python en espera 46
Ilustración 47 Transmisión Cliente-Servidor.....47
Ilustración 48 Transmisión de archivo .txt al servidor Antes/Después 47
Ilustración 49 Consola Raspberry, enviando, consola Windows, Recibiendo 48
Ilustración 50 Transmisión y Recepción concluida.....48
Ilustración 51 Información recibida en el server Antes/Después 48
Ilustración 52 Esquema de comunicación finalizada.....49

Capítulo I, Esquema para desarrollo del proyecto.

1.1 Introducción

En este capítulo se describe el planteamiento, la justificación, el objetivo y los antecedentes de este proyecto. Tomando en cuenta la importancia que ha tenido la evolución del Internet of Things el cual comienza a ser parte del entorno en el que nos encontramos, este tema en conjunto con otras tecnologías de TI forman parte de un sistema que hace posible la interacción humano-máquina, máquina-humano y máquina-máquina.

Como bien sabemos a tecnología se mueve a una velocidad muy rápida creando nuevas corrientes y servicios que van de la mano con la investigación, el desarrollo y la implementación de tecnologías. Dentro de este crecimiento todo nuestro entorno se enlaza poco a poco con nuevas tecnologías, las que hoy se han convertido en nuestra actual base de información, conocimiento y entretenimiento, como la Internet.

Desde que inicia el concepto de la Internet, su principio es el de compartir la información, facilitando tareas y optimizando tiempo como el hecho de enviar información a lugares lo suficientemente remotos en un lapso de tiempo corto, un concepto que ha evolucionado hasta convertirse en uno de los grandes inventos que marca en la sociedad un antes y un después.

Del mismo modo crea nuevas formas de interacción de las personas, la creación de nuevos negocios, nuevas empresas y nuevas formas de ventas, gracias a la gran cantidad de datos que se generan en Internet, datos que son utilizados por las empresas para la creación y venta de productos o servicios.

Esta nueva interacción con Internet evoluciona y se aplica cada vez más incluso para otro tipo de actividades que anteriormente se encontraban separadas de Internet y que ahora tienen un trabajo en conjunto como la automatización de rutinas cotidianas en objetos como televisiones, refrigeradores, cafeteras, cámaras entre otras, donde el humano acepta dejar a cargo a las máquinas y a sus algoritmos de programación.

Esta evolución desde hace unos años se ha consolidado como un nuevo concepto llamado IoT, un término que cambia la forma en como interactuamos con cada uno de los objetos que tenemos en casa, en oficinas, en las calles, en los campos de agricultura por mencionar algunos ejemplos.

Decir que IoT cambia la interacción de las personas con el mundo pareciera una declaración arriesgada, pero es bueno recordar y tener presente que la Internet ha tenido un importante impacto en la sociedad, si ponemos atención a la forma en como interactuamos con el entorno podremos ver esas diferencias marcadas en la comunicación, la educación, la ciencia, el gobierno y las empresas que cada vez interactúan con un enlace alojado en Internet, por decirlo de una forma muy básica.

Por lo que es necesario tener en cuenta que IoT no es una ficción y que representa la próxima evolución de Internet, donde existirá una interacción de los dispositivos, los datos recolectados y los actuadores que junto con los algoritmos de programación implementados proporcionarán una automatización de actividades que anteriormente teníamos que hacer de forma manual.

Lo que ocasionará una transformación de las TI como la creación de nuevas estructuras de comunicación para estas redes de sensores y actuadores y del mismo modo la creación de nuevos perfiles de profesionistas para contribuir al funcionamiento de estas nuevas tecnologías de interacción la Internet y tener un impacto benéfico para la sociedad.

1.2 Planteamiento del problema

La definición de IoT para CISCO habla de la *"conexión de millones de dispositivos inteligentes y sensores conectados a Internet. Estos dispositivos y sensores conectados recopilan y comparten datos para que muchas organizaciones las utilicen. Estas organizaciones incluyen empresas, ciudades, gobiernos, hospitales y personas"* (Academic, 2019). Una definición que suena prometedora para la automatización trabajos rutinarios que el humano dejará de hacer pero que ahora es el turno de implementar la tecnología para que IoT funcione, ya que aún se encuentra en desarrollo.

Habrá que definir los dispositivos a utilizar y como ejecutará sus acciones para el funcionamiento esperado y para ello existen diferentes formas de trabajar, hablando desde los microcontroladores, los protocolos de comunicación, en qué tipo de red estarán interactuando y que protocolos se utilizaran.

Para la implementación de este proyecto se pudo encontrar información en la red de implementarlo de diferentes formas y diferentes protocolos para su transmisión de datos como HTTP los cuales no son meramente los mejores para su implementación en IoT debido a la carga de datos que se están

transmitiendo continuamente y la gran cantidad de sensores transmitiendo información para tomar muestras de comportamiento de algún objeto, por lo que se debe utilizar la tecnología adecuada para su funcionamiento correcto.

Por ello se crea este proyecto que define el uso de tecnologías que pueden funcionar y ser implementadas para decir que efectivamente estamos creando IoT de la forma mejor posible ya que existe mucho que profundizar y estar realizando actualizaciones, pues IoT va ligado con diferentes conceptos como Artificial Intelligence, Big Data por mencionar un par de ellos (Academic, 2019)

Aunque aún existe mucho que hablar de Internet of Things, hasta el momento uno de los objetivos básicos es la recolección de datos del entorno, analizarlos para ejecutar una acción por lo que es necesario crear una estructura definida con los estándares de comunicación definidos para su lograr su funcionamiento correcto.

1.3 Justificación

Actualmente aún no existe una estructura bien definida para el funcionamiento del IoT, pero existen varias propuestas y debido a ello en este trabajo se desea hacer funcional la comunicación *Sensor-Microcontrolador-Gateway-Cloud Computing* para el transporte y el análisis de datos para ejecutar acciones bajo este análisis realizado previamente.

En este proyecto se realizó un avance para la parte física y lógica en la creación del puente de datos entre los sensores y los servidores, posteriormente el estudiante que desee trabajar con IoT podrá unir el dispositivo a un servicio de Cloud Computing para lograr el funcionamiento de toda la estructura con mayor facilidad utilizando los dispositivos, protocolos y la comunicación punto a punto entre microcontroladores y Gateway, con ello comenzar a realizar experimentos avanzados.

1.4 Objetivo general

El objetivo general de este trabajo recepcional es crear un Gateway con capacidad de transferencia de datos por cinco protocolos de comunicación, cuatro para la implementación de redes de sensores y uno para la transferencia de datos hacia un servidor que se encarga de almacenarlos para así tener una facilidad de integración a la estructura IoT, la ilustración 1 muestra el esquema del proyecto propuesto.

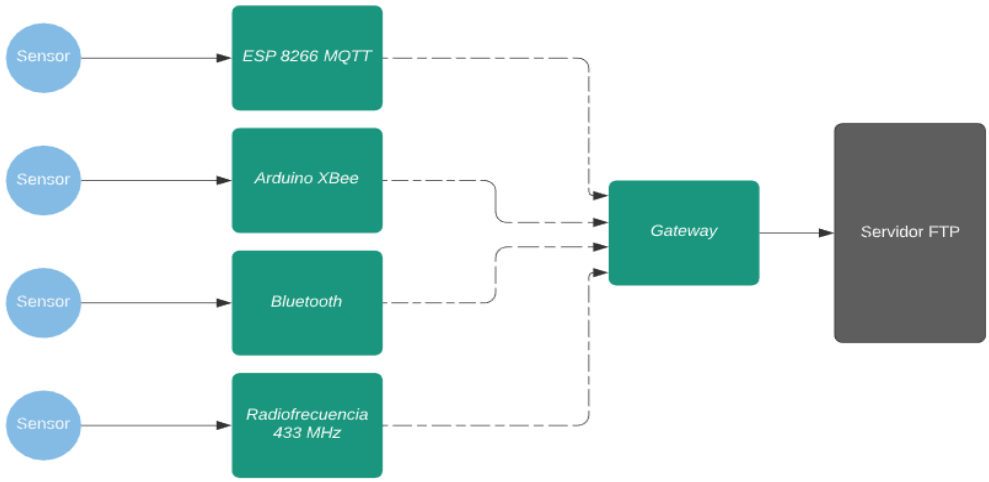

Ilustración 1 Esquema de proyecto

En la ilustración se observa la propuesta de realizar una conexión punto a punto para cada uno de los protocolos utilizados para hacer la transferencia de los datos hacia el Gateway.

1.5 Objetivos particulares

Crear un Gateway para proveer una conexión punto a punto en los cuatro protocolos de comunicación para los sensores IEEE.802.11, IEEE 802.15.4, IEEE 802.15, radiofrecuencia a 433MHz y una transferencia de datos por FTP entre Gateway y Servidor para su almacenamiento de los mismos.

El Gateway es implementado en la tarjeta de desarrollo Raspberry Pi 2B donde se habilitan cada uno de los protocolos descritos tanto a nivel hardware como a nivel software.

El protocolo IEEE.802.11 se habilita mediante la implementación de un Access Point en Raspberry Pi 2B para obtener una conexión punto a punto entre el microcontrolador NodeMCU y Raspberry.

Se Implementa el protocolo MQTT (Message Queue Telemetry Transport) en NodeMCU y Raspberry para para enviar datos desde el sensor al Gateway.

Se Agrega la comunicación con el protocolo IEEE 802.15.4, utilizando el módulo XBee S1 Pro a través de protocolo UART, para obtener comunicación entre un Arduino y el Gateway

Se implementa la comunicación con el protocolo Bluetooth al Gateway,

utilizando el módulo HC-05 y la interfaz USB, para la comunicación entre un Smartphone y Raspberry.

Se implementa el sistema de comunicación por radiofrecuencia utilizando los módulos transmisor y receptor a través de los GPIO en Raspberry para su comunicación entre Arduino y el Gateway

Por último se implementa un cliente FTP en el mismo Gateway para transmitir los datos al servidor implementado en un host dentro de una red LAN.

1.6 Metodología

El desarrollo del proyecto consta de la implementación de este Gateway el cual conlleva una serie de pasos a seguir para lograr los objetivos planteados, que van desde conexiones a nivel físico, con los dispositivos electrónicos necesarios hasta las configuraciones y programación tanto para Raspberry como para los microcontroladores como Arduino y NodeMCU, enseguida se enlistan y describen de forma breve cada uno de ellos.

- Revisión de la literatura, sobre el concepto de Internet Of Things, sus aplicaciones, su desarrollo actual, la importancia para las empresas y el tipo de ciencias y tecnologías involucradas para su desarrollo.

- Configuración memoria micro SD para la instalación del Sistema Operativo Raspbian, bifurcación de Debian para el funcionamiento de Raspberry.

- Configuración de Raspberry para su funcionamiento como un Access Point utilizando la tarjeta de red inalámbrica WN725N.

- Pruebas de comunicación entre Raspberry y NodeMCU, realizando pruebas básicas utilizando el Protocolo MQTT lanzando datos con ayuda de un push botón

- Captura de los datos entre Raspberry y NodeMCU realizando programación en Python.

- Configuración Radios Xbee, mediante el software de XCTU-DIGI y pruebas de conexión entre Raspberry Pi y Arduino con radio XBee para transferir datos de un sensor de humedad bajo tierra, los cuales son capturados en Raspberry con programación en Python

- Configuración de Raspberry para habilitar la comunicación de Bluetooth y realización de pruebas de comunicación hacia Raspberry

- con el módulo HC-05 y se captura los datos recibidos con programación en Python.

- Instalación de protocolo WiringPi, para habilitar la comunicación de Radiofrecuencia para realizar pruebas de comunicación entre Arduino y Raspberry, utilizando los módulos de comunicación por radiofrecuencia con la transmisión de datos utilizando el sensor de temperatura LM35 desde Arduino hacia Raspberry

- Implementación de la comunicación cliente-servidor FTP, mediante web sockets creados en Python.

Cada uno de estos pasos fue implementado en el proyecto llegando a su objetivo final con sus respectivos funcionamientos. En el siguiente capítulo se mencionan de forma muy breve los conceptos básicos utilizados en este proyecto, términos como Internet y su tendencia, Internet of Things y la revolución 4.0, Gateway IoT, Cloud Computing y por último Fog Computing.

Capítulo II, Antecedentes.

2.1 Internet y su tendencia

En el año 2012, según las estadísticas de CISCO, se obtuvo un registró de una cantidad de dispositivos conectados a Internet que superaron el número de habitantes en la Tierra, un dato el cual posiblemente era inimaginable en el momento en que se comenzó a construir esta infraestructura de la cual ahora un gran número de personas, instituciones y empresas hacen uso de ella.

Con ello cada una de las organizaciones se adaptan a los cambios tecnológicos debido al impacto que tiene sobre la sociedad, los servicios que ofrece con una mayor velocidad como la comunicación a distancia ya sea por telefonía celular o mensajería de texto, los cuales trabajan sobre los protocolos de Internet proporcionando una cierta comodidad al usuario.

Internet ha tenido diferentes fases antes de concretarse como la red de redes, desde sus inicios donde el proyecto comenzó como ARPANET "*(Advanced Research Projects Agency Network) donde la intención de este*

proyecto fue crear una red de computadoras dentro de la universidad de Estados Unidos, posteriormente gracias al trabajo y esfuerzo de los participantes la red fue creciendo, esta vez con más ambiciones que harían llegar más lejos el proyecto" (Velázquez, 2016)

Estas redes iniciaron como redes de información tipo LAN (Local Area Network) las que hasta ese momento fueron la red de UCLA (University of California- Los Ángeles) la red de UCSB (University of California of Santa Barbara) y la red de SRI (Standford Research-institute of California). A partir de aquí comenzó a trabajar como una red de computadoras para la ciencia y la milicia, no obstante Internet toma forma comercial en la década de los años 90 donde inicia lo que hasta hoy es posible observar las ventajas que ofrece.

El objetivo del proyecto comenzó con el único propósito de poder compartir información a distancia, entre diferentes instituciones recurriendo a la tecnología que hasta el momento se tenía, y como se convierte en una herramienta incluso podemos decir vital para las vidas de las personas que hoy en día convivimos diariamente con Internet.

La Internet que posteriormente tendría un impacto en la sociedad y en el mercado, le fue dando mayor agilidad a la comunicación, información, entretenimiento, publicidad, mejores servicios y las ventas, un impacto que aún hoy en día permanece y cada vez se fortalece conforme el paso del tiempo y lo hace muy rápido.

Debido a esto la sociedad se ve obligada evolucionar conforme los cambios que se obtienen en Internet pues la comunicación, las ventas, negocios y publicidad, cada vez se hacen con la ayuda de esta herramienta, incluso podemos atrevernos a mencionar que si algo o alguien no se encuentra en Internet no existe, pues muchas personas cada vez más buscan información en esta red mundial. (Evans, 2011)

Esta información que se guarda en los servidores toma relevancia, pues hoy en día es un factor al que se le saca provecho debido a la información que existe sobre el comportamiento de las personas que interactúan con las aplicaciones que abundan en la red de redes, esta información es utilizada para la creación de todo un mercado virtual.

Gracias a la velocidad de crecimiento de la Internet, gracias a los cambios significativos que ha tenido el hardware, el software, los procesadores, algoritmos implementados y también al análisis de datos es posible

simplificar muchas actividades cotidianas y ser productivos en actividades con mayor relevancia y mayor productividad humana y de conocimiento.

Lo que nos lleva al siguiente paso y la nueva tendencia de la Internet, el IoT, el cual en términos muy generales trata la conexión de objetos a Internet, recolección de datos y la automatización de tareas específicas proporcionando optimización de tiempo y recursos, que tendrá otro gran impacto en la sociedad en un futuro que ya se encuentra cercano.

"Cuando lo inalámbrico este perfectamente desarrollado, el planeta entero se convertirá en un gran cerebro, que de hecho ya lo es, con todas las cosas siendo partículas de un todo real y rítmico… y los instrumentos que usaremos para ellos serán increíblemente sencillos comparados con nuestros teléfonos actuales. Un hombre podrá llevar uno en su bolsillo" Nikola Tesla.

Hoy en día pudiera parecer poco significativo una revolución que vino de la mano con la popularización de la conectividad inalámbrica y que en ese entonces ya se hablaba de factores bastante avanzados para la época, ya fuese la comunicación por celular o por Wi-Fi, durante el inicio del siglo XX. Hoy vemos que esta evolución permite presenciar una primer crecimiento de los objetos conectados constatado en esta última década, donde han venido sucediendo nuevos conceptos como el WSN (Wireless Sensor Network) o el M2M (Machine to Machine), para finalmente dar paso al IoT (Internet of Things)

A pesar de todos los grandes avances, este es apenas es un relato más de un inicio que se encuentra en desarrollo, con las tecnologías existentes, pero que se está escribiendo este futuro con nuevas direcciones de red, nuevos protocolos y nuevos dispositivos, donde aún se encuentra en una fase de coexistencia pruebas, experimentación y de crecimiento.

2.2 Internet of Things y la revolución 4.0

El IoT es la conexión de dispositivos inanimados conectados a Internet o en redes de menor magnitud interactuando entre sí, estas regularmente conformadas por sensores, actuadores y microcontroladores. Estos dispositivos comparten datos que atraviesan toda una estructura desde su nivel más bajo como son los sensores hasta llegar a su nivel más alto como lo es un servicio de Cloud Computing para proporcionar una interacción con el usuario o con alguna otra máquina que pueda estar situada en alguna ciudad remota, en algún edificio o en alguna persona y estos objetos pueden ser tan comunes como una lámpara o alguna puerta, una cafetera o algún

refrigerador.

Después de la recolección de datos se realiza un análisis obteniendo información de los cambios que pudieran haber ocurrido en un lapso de tiempo como cambios de PH en algún lago, cambios de humedad en la tierra de algún cultivo, un fenómeno meteorológico, la falla en alguna luminaria o el monitoreo de los signos vitales de una persona, con ello tomar una decisión y ejecutar una acción rápida optimizando tiempo. (Azcarategui, 2018)

Aunque no existe un consenso al respecto, a estas alturas de los avances tecnológicos y científicos podemos decir que el IoT, el cómputo en la Nube, el Big Data y el análisis de datos avanzado parecen ser los pilares más importantes para la Revolución Industrial 4.0 donde el objetivo es automatizar los procesos de manufactura y digitalizarla con el objetivo de tener un constante monitoreo con capacidad de predecir fallas en sus líneas de manufactura y con ello reaccionar rápido para dar mantenimiento a las maquinas involucradas incrementando producción, ahorro de tiempo y esfuerzo (Ynzunza Cortés, 2017)además de la creación de nuevos productos y servicios.

Una vez que se puede observar el panorama que trae consigo el IoT quiero iniciar con los conceptos que hacen posible el funcionamiento de un proyecto de este tipo y con ello obtener una interacción humano-máquina, maquina-humano y máquina-máquina.

2.4 Gateway IoT

IoT funciona con sensores a través de la implementación de tarjetas de desarrollo y estas cuentan con variedad de protocolos de comunicación, un Gateway IoT puede considerarse una computadora de propósito general pero de tamaño reducido y también funciona como una puerta de enlace solamente que a diferencia de los Gateways convencionales estos dispositivos tienen la habilidad de agrupar datos que provienen de diferentes estándares de comunicación, como pueden ser, Bluetooth, Radiofrecuencia, XBee, Wi-Fi, por mencionar algunos, y los envía a un servidor de mayor procesamiento como la nube, por medio de un solo estándar de comunicación, estrictamente hablando, además de proporcionar la traducción de protocolos, agregación, filtrado, seguridad y actualizaciones.

Este enlace puede realizarse a nivel de red LAN o también a nivel WAN, dependiendo de la aplicación que se quiera utilizar y los protocolos que se

estén utilizando, ya en algunas ocasiones es necesario transmitir los datos a diferente áreas geográficas (Academic, 2019).

2.5 Arquitectura IoT

Un dato recolectado por un sensor puede pasar por diferentes tecnologías para que pueda ser observable para un usuario ya sea en una base de datos o en un gráfico para determinar la aplicación que se le vaya a proporcionar según lo recolectado.

Para este caso se propone una estructura básica donde los datos recolectados se llevan a un servidor para su almacenamiento utilizando un Gateway que recibe datos de diferentes protocolos de comunicación y los lleva a este servidor para el almacenamiento de todos esos datos recolectados y que provienen de diferentes protocolos como se muestra en la ilustración 2.

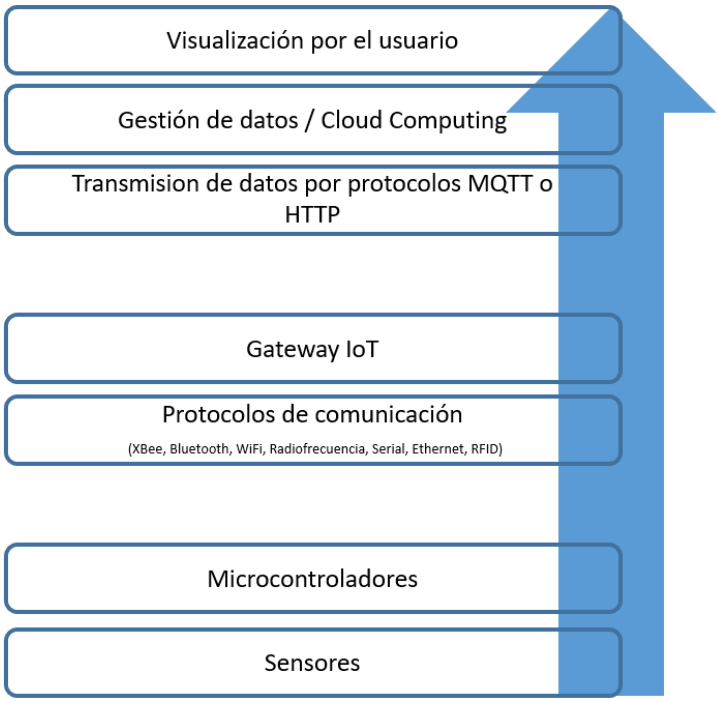

Ilustración 2 Arquitectura IoT

Los objetos procesan la información que se obtiene de los sensores, envían los datos al Gateway el cual puede realizar procesamiento de estos datos si no son demasiados y si también se encuentra dentro de sus capacidades computacionales, un proceso que se le conoce como Fog Computing

(computación en la niebla). Del mismo modo el Gateway manda toda esa información hacia un servidor de mayor capacidad computacional o hacia un servicio en la nube donde se procesan las solicitudes de servicio, en caso de que exista una comunicación hacia los microcontroladores la misma nube manda una respuesta hacia los objetos (Academic, 2019).

Para que esto funcione de la forma en que está planteada se deben tomar en cuenta diferentes factores como la cantidad de cómputo, la capacidad de procesamiento en los nodos existentes en la niebla y una comunicación dúplex, además debe existir una diferencia entre las tareas de procesamiento pesado y tareas de procesamiento ligero, para que pueda existir una distribución de datos correcta y con ello ejecutar las tareas asignadas según la aplicación.

2.6 Cloud Computing

Esta tecnología tiene la capacidad de proporcionar servicios a un usuario ya sea como infraestructura (IaaS), como software (SaaS) y como plataforma (PaaS) en el cual el usuario del servicio define cual le conviene según la aplicación que se vaya a desarrollar, hablando de aplicaciones como IoT el Cloud Computing puede proporcionar una administración y gestión de los datos con ello ejecutar algoritmos de predicción para ejecutar un evento automático (interacción máquina-máquina), así como también puede proporcionar un acceso a desde casi cualquier parte del mundo para poder observar el comportamiento de un sensor y activar un actuador de forma manual y remota (interacción humano-maquina).

2.7 Fog Computing.

El concepto de Fog Computing, aunque no es una estructura completamente implementada en este proyecto es necesario mencionarla debido a su importancia en la interacción con IoT. Cumple con una actividad muy similar a la del Cloud Computing, sin embargo esta realiza el análisis de información en menor magnitud de datos y puede encontrarse en los nodos que se encuentren libres.

Con los buenos procesamientos que se pueden hacer en computadoras muy pequeñas, incluso puede realizarse en el mismo Gateway IoT para así reducir el tráfico en la red al momento de brindar una plataforma para el filtrado de datos generados ya sea en un Gateway o un objeto, de este modo los datos enviados a la nube se reducen así como la latencia, especialmente para aplicaciones que requieren procesamiento de menor cantidad y en tiempo real. (Tárano León, 2018)

Ahora que ya se mencionaron los conceptos que conforman a la arquitectura IoT se realiza una breve descripción de cada una de las tecnologías utilizadas para el desarrollo del proyecto.

2.6 Raspberry

Es una computadora de bajo costo creada por la fundación educacional Raspberry Pi en Inglaterra, la cual ha sido distribuida comercialmente en cuatro versiones conocidas por medio de sus nombres asignados Raspberry Pi (Modelo A, Modelo B-1, Modelo B-2, Modelo B+). Cuenta con una estructura que permite entrar al hardware integrado para ejecutar diferentes tareas.

La versión utilizada en este proyecto, Raspberry 2B+, proporciona acceso a los 26 pines digitales, los cuales contienen pines de alimentación +5v, +3.3v y GND así como los puertos GPIO con posibilidad de configuración como entradas y salidas excepto pines de comunicación, alimentación y PWM, dos interfaces USB 2.0, Interfaz Ethernet 100, Conector HDMI, memoria RAM de 512MB, conector micro-USB, la siguiente ilustración muestra una foto del modelo Raspbery 2B (Wallace, 2014)

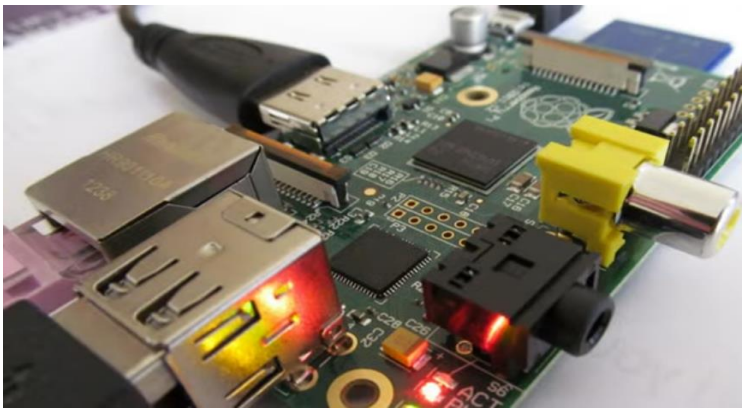

Ilustración 3 Raspberry Pi 2B+

2.7 NodeMCU

Tarjeta de desarrollo creada por Espressif Systems, empresa dedicada al desarrollo de soluciones IoT a través de sus módulos y placas de desarrollo como el ESP8266 utilizado en este proyecto aunque existen otras versiones como ESP32 y Esp32-S que proporcionan protocolos de comunicación WiFi y Bluetooth integrados.

Este dispositivo fue utilizado para la implementación de este proyecto debido

a su precio económico, fácil acceso y con documentación suficiente para crear un desarrollo con mayor velocidad.

Además que proporciona la posibilidad de comunicar por medio de las versiones 802.11 b/g/n por lo que no hay problemas de comunicación, puede programarse en entornos como Lua, Arduino y MicroPython y cada uno con el acceso a sus pines GPIO (General Proposit Input Output), entradas y salidas de propósito general, la ilustración 4 muestra una imagen del dispositivo.

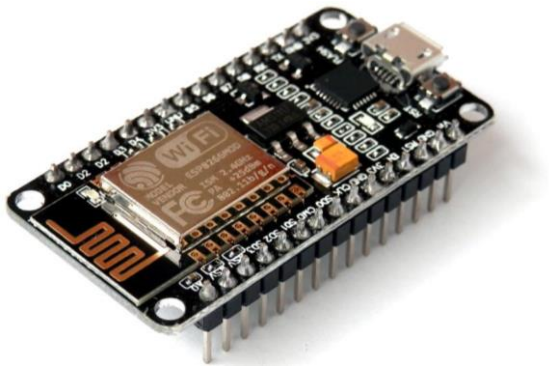

Ilustración 4 Placa NodeMCU

2.8 Arduino UNO

Del mismo modo es una tarjeta de desarrollo creada por Arduino, empresa dedicada al desarrollo de placas de desarrollo y que cualmente también desarrolla tecnología para la implementación de IoT, pero a diferencia de Espressif Systems su tecnología tiene un precio elevado.

Arduino UNO fue utilizado para la implementación de este proyecto igual que NodeMCU tiene un precio económico, fácil acceso y con documentación suficiente para crear un desarrollo con mayor velocidad. Del mismo modo proporciona la posibilidad de comunicar con shields por medio de sus protocolos de comunicación como UART y SPI así como sus pines digitales, pines analógicos y alimentación de 5v y 3.3v. (Schmidt, 2011)

Para este proyecto se utilizó el protocolo UART para el uso de shields XBee así como sus pines digitales y analógicos para los sensores.

Mencionada la tecnología utilizada se describe brevemente los protocolos de comunicación implementados XBee, Bluetooth, WiringPi, MQTT, y FTP.

2.9 XBee

Uno de los estándares implementados en este proyecto fue XBee (IEEE 802.15.4), el cual se deriva del estándar principal ZigBee ofrecido por la empresa DIGI, la cual es utilizado principalmente para realizar proyectos de Home Automation, pero que no es tan accesible al público en general debido a sus costos, los cuales van aumentando dependiendo de las posibilidades de crear proyectos de mayor escala, sin embargo no queda descartada su integración al mundo del IoT debido a las necesidades de cada proyecto

XBee es un estándar implementado para el armado de redes inalámbricas de corta distancia y baja velocidad, aunque cabe mencionar que algunos modelos de su antecesor ZigBee llegan a superar las comunicaciones de Wireless o Bluetooth, comparándolos en distancias de transmisión aunque con bajas tasas de transmisión no obstante ha sido útil para proyectos separados del direccionamiento IP.

No obstante a pesar de su tasa de transmisión éste ofrece diversas topologías de red, puede realizar conexiones punto a punto, redes tipo Mesh, además de topologías como la de árbol o tipo estrella según los alcances de cada nodo con el uso de routers XBee los cuales extienden la cobertura de la red creando rutas adicionales, en la ilustración 5 se muestran las topologías que puede ofrecer (Dignani, 2011)

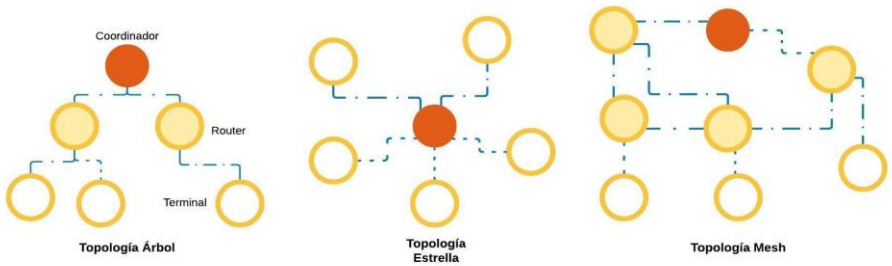

Ilustración 5 Topologías XBee

De izquierda a derecha se muestra la topología de árbol para la creación de una red jerárquica, la topología para la creación de una red centralizada y a la derecha una topología mesh para la creación de una red distribuida utilizando la tecnología XBee o ZigBee.

2.10 Bluetooth

Es una tecnología inicialmente creada por Ericsson en 1989, la cual después de treinta años de desarrollo y mejoras esta tecnología proporciona la posibilidad de crear redes de objetos las cuales toman el nombre de WPAN

(Wireless Personal Area Network) Red Inalámbrica de Área Personal, incluye dispositivos inalámbricos de corto alcance que abarcan áreas de algunas decenas de metros, la cual conecta dispositivos periféricos como impresoras, teléfonos móviles, electrodomésticos o asistentes personales digitales

Bluetooth proporciona un canal de comunicación a un máximo de 720 Kbit/s, opera en la frecuencia de radio de 2.4 a 2.8 GHz, muy similar a la tecnología de ZigBee, y con la posibilidad de transmitir en Full Duplex. (LINARES RUIZ, QUIJANO VÁSQUEZ, & HOLGUÍN LONDOÑO, 2004)

Bluetooth aún se sigue trabajando en las actualizaciones de este protocolo de comunicación como la versión 5.0, del año 2017, el cual contiene un doble de velocidad, mejor fiabilidad y amplio rango de cobertura, o la versión 5.1 del año 2019, la cual gira entorno a la localización muy parecida a GPS aunque no completamente pero si podrá determinar una ubicación con un margen de error en la escala de centímetros, podrá identificar la dirección de la señal buscada.

Ilustración 6 Logotipo Bluetooth

2.11 Comunicación por radiofrecuencia

A pesar de que las tecnologías avanzan con velocidad y con una gran cantidad de mejoras en sus estándares y protocolos que los conforman, a veces es necesario voltear a ver a tecnologías un tanto más sencillas que además de ser económicas pueden llegar a contar con una similitud a los estándares que van tomando fuerza en estos últimos días.

Los módulos de radiofrecuencia utilizados en este proyecto trabajan a una frecuencia de 433MHz, tanto transmisor como el receptor pueden ser utilizados para la comunicación punto a punto con la unión de los radios con decodificadores, estos pueden estar implementados por hardware con la ayuda del circuito integrado HT12E, o también por software con ayuda de Arduino y Raspberry, como se realizó en este proyecto.

Este sistema de comunicación, no disponen de filtros ni identificador ID por hardware, por lo que es necesario de un protocolo de comunicación implementado por software para obtener una comunicación robusta la cual se realiza en este caso utilizando Arduino con el transmisor y Raspberry con el receptor con el uso del protocolo WiringPi. (Gordon, http://wiringpi.com/, 2019)

2.12 MQTT

MQTT, Message Queue Telemetry Transport (Telemetría de Transporte para Colas de Mensajes) está basado en la pila TCP/IP como base para la comunicación y transfiere los datos por un canal abierto de forma constante hasta que el cliente finaliza dicha conexión.

MQTT define dos tipos de identidades en la red, un bróker y un cliente donde el broker es un servidor que recibe todos los mensajes publicados por los clientes origen, posteriormente puede redirigir estos mensajes a clientes destino en este caso un cliente puede ser cualquier cosa que interactúa con el broker por su canal de comunicación como un ESP8266 el cual publica mensajes.

Los mensajes de MQTT se organizan por temas o TOPICS, con esto se tiene que especificar a los clientes en que TOPIC pueden interactuar con sus mensajes que lanzan y como este protocolo proporciona un ambiente de trabajo jerárquico cada TOPIC se coloca en la punta de los SUB-TOPIC como se muestra en la ilustración 7.

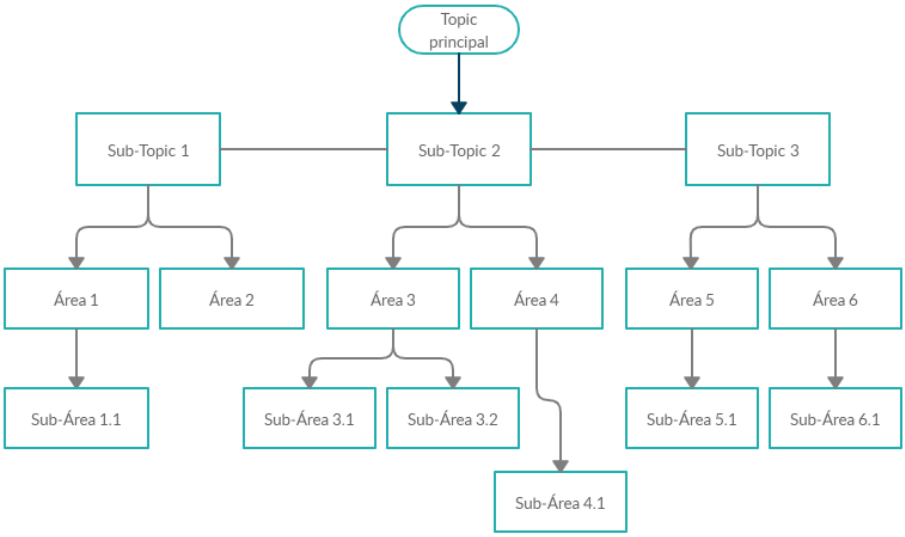

Ilustración 7 Estructura de TOPICS

Con ello recibe los mensajes publicados de cada sensor y la comunicación puede realizarse de uno a uno o de uno a muchos TOPIC, esto dependerá de la aplicación que se desee implementar.

Un TOPIC se representa mediante una cadena y tiene una estructura jerárquica, en el cual cada jerarquía es separada mediante un SLASH, ***/***, como se muestra en la ilustración 8. (Yuan, 2017) (IBM, 2017)

Ilustración 8 Representación de TOPIC

La ilustración anterior muestra la sintaxis de un TOPIC, de esta forma puede ser ejecutada tanto en la terminal como en el programa creado para la comunicación entre dispositivos.

Tanto la subscripción como la publicación deben tener la misma estructura para que el mensaje llegue exitosamente. Esta es la forma cómo funciona el protocolo MQTT donde existe flujo de mensajes sin esperar una respuesta por parte del servidor como comúnmente se ejecuta en HTTP ya que el bróker siempre se encuentra en espera de mensajes. (Jamie M, Jeremy G, Andrew D, & Bharat V)

2.13 Protocolo FTP (File Transfer Protocol)

FTP es un protocolo de transferencia de Ficheros (File Transfer Protocol), tiene diferentes características que lo vuelven bastante útil, entre ellas el uso compartido de ficheros, el uso indirecto de servidores remotos para la creación de ficheros en diferentes ordenadores y transferir datos entre sistemas conectados a una red, estas pueden ser LAN, MAN o WAN.

FTP está basado en una arquitectura cliente servidor y proporciona un mecanismo estándar para la transferencia de archivos entre sistemas de redes TCP/IP. Para el uso de este protocolo se es necesario colocar a un cliente el cual se encargará de subir, consultar o descargar los archivos que existan o puedan existir en el servidor, como bien se describe, el objetivo fundamental del protocolo FTP es intercambiar ficheros entre máquinas a través de la red.

El protocolo FTP suele usarse para manejar grandes cantidades de archivos, por lo que suele resultar útil para el desarrollo web, esto cuando se realizan cambios, es posible administrar las transferencias de archivos con una sesión FTP, este proporciona una forma sencilla de cargar archivos como ficheros, imágenes o simplemente como protocolo de transferencia de archivos para grandes lotes. (J. Postel, 1985)

Capítulo III, Desarrollo del prototipo.

3.1 Configuración e implementación del Gateway en Raspberry Pi

Previamente se expuso de forma breve conceptos básicos sobre los protocolos que fueron utilizados en este proyecto para implementarlos en Raspberry Pi 2B, posteriormente de las configuraciones se realizaron pruebas de conexión con la ayuda de Arduino, ESP8266 y Raspberry, obteniendo exitosamente cada prueba, lo mencionado se describe en los siguientes párrafos donde se explica la forma en que realizó la implementación de este dispositivo, donde fue necesario llevar a cabo las siguientes configuraciones de hardware y software, de esta forma poner en funcionamiento el prototipo .

Este sistema fue implementado en una Raspberry Pi 2B, debido su simplicidad que cuenta al trabajar como una computadora, su tamaño reducido y su bajo costo además para cuestiones de aprendizaje y comprensión propia en las configuraciones se realizan iniciando desde una capa física hasta una capa de red para cada uno de sus protocolos que

ofrece y que pueden anexarse, todo ello se detallara a continuación.

Para iniciar con las configuraciones de la tarjeta de desarrollo lo primero es preparar una memoria flash micro SD con el SO (Sistema Operativo) Raspbian Strech, versión de 2018, sin embargo no quiero dejar pasar por alto un dato importante como la versión del Kernel 4.14.79, debido a que es uno de los factores importantes que se transformó en un reto para llevar a cabo el correcto funcionamiento de la Raspberry como Acces Point.

Para realizar la instalación del SO Raspbian, primero se descargó el desde la página oficial de raspbberrypi.org, el cual fue instalada en una memoria flash Micro-SD, mejor conocido con el término en inglés como bootstraping, el cual se lleva a cabo gracias a la ayuda de software como Rufus versión 2.17 y para realizar los respaldos del SO se utilizó Win32Disklmager, los cuales son los que se presentan en la siguiente imagen.

Ilustración 9 software de preparación y respaldo de SO

Una vez que el Sistema Operativo se encuentra en la memoria flash ahora se procede a insertarla en Raspberry, para que inicie con el proceso de arranque al momento que se encienda, para ello se conecta la interfaz de alimentación y la interfaz Ethernet y en definitiva, estará lista para trabajar, aunque cabe mencionar que la versión del Kernel no permite tener un acceso remoto por SSH por políticas de seguridad, debido a ello es necesario iniciar como si se tratará de una computadora personal con monitor teclado y mouse, para habilitar los puertos de conexión SSH, por puerto 22 y VNC (Virtual Network Computing), para ello ejecutamos el comando sudo raspi-config y muestra la pantalla como se muestra en la ilustración 10.

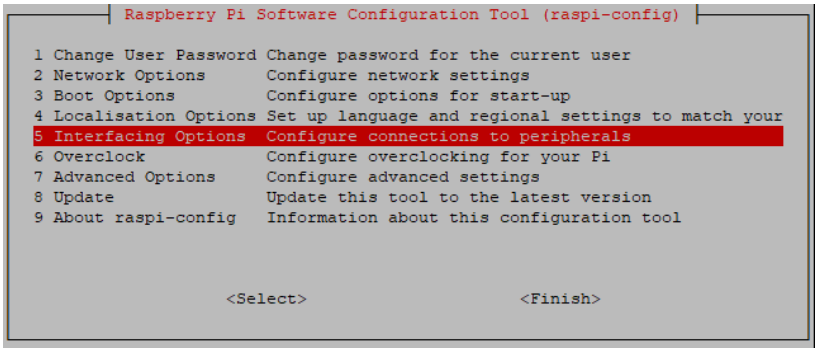

Ilustración 10 opciones de configuración

La pantalla de herramientas de configuración en la cual seleccionamos la opción Interfacing Options, esta permite el acceso a la configuración tanto de puertos como de pines GPIO, protocolos como SPI e I2C, Puerto Serial, VNC y Cámara, el ejemplo se muestra en la ilustración 11.

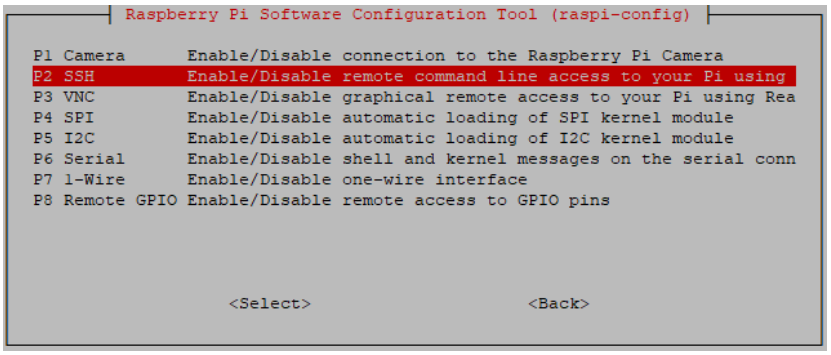

Ilustración 11 habilitar puerto 22

En ella se selecciona la opción SSH para habilitarlo, lo que posteriormente pregunta que si estamos seguros de habilitar dicho puerto, con ello selecciona la opción de <Yes> y con ello quedará habilitado el puerto para tener una conexión remota sin necesidad de utilizar monitor, mouse y teclado.

Tanto Putty como VNC son habilitados del mismo modo y para ingresar a Raspberry remotamente solicitarán credenciales de identificación para proporcionar acceso, estos fueron configurados de la forma en que se muestra, por cuestiones de simplicidad.

Usuario: pi
Password: pi

No obstante cabe mencionar que para evitar una intrusión estas no son credenciales con altos índices de seguridad para evitar una intrusión, por lo que si se requiere que este proyecto sea más robusto requerirá seguridad para proteger los datos que se obtienen de cada uno de sus protocolos y de esta forma el dispositivo permanezca fuera de ciertas vulnerabilidades informáticas.

Una vez habilitados el puerto 22 y el puerto 5900 se instala el software requerido en la computadora que tendrá acceso hacia Raspberry, en este caso se muestra VNC y PUTTY, y con ello estará lista la configuración para acceder remotamente. A pesar de que VNC proporciona una visibilidad del escritorio del ordenador al que se accedió el más utilizado fue PUTTY debido al alto uso de la terminal de comandos para la instalación, activación y administración de los servicios necesarios en este proyecto.

Ilustración 12 Herramientas para la comunicación remota hacia Raspberry

Ya configuradas las herramientas necesarias, fue posible realizar una primera conexión remota, accediendo desde Putty, sin dejar de mencionar que para ello el primer requisito es la dirección IP del de la computadora, para consultar las direcciones que se asignan a cada ordenador, se realizó un escaneo con herramientas como NMAP, ya sea para Linux o Windows, o con Fing, cada uno de sus iconos se muestra en la ilustración 13. Fing es una aplicación para Smartphone, esta herramienta proporciona las direcciones IP más rápido que NMAP y del mismo modo es de acceso eficaz y que es posible tenerlo en el Smartphone para uso rápido, estos nos proporcionaron la dirección IP junto con la dirección MAC y puertos habilitados, la cual es suficiente información para acceder al ordenador.

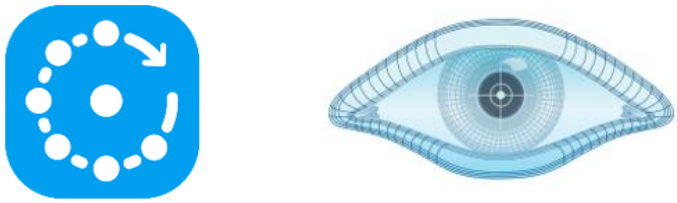

Ilustración 13 Herramientas para la consulta de direcciones IP

Cuando se realiza el escaneo de la red, aparece la lista de dispositivos que se encuentran en esta, de la cual se tomó la dirección de interés, y se ingresa en el software PUTTY para conectar con Raspberry de forma remota. En la ilustración 14 se muestra el escáner de fing, donde localiza a Raspberry.

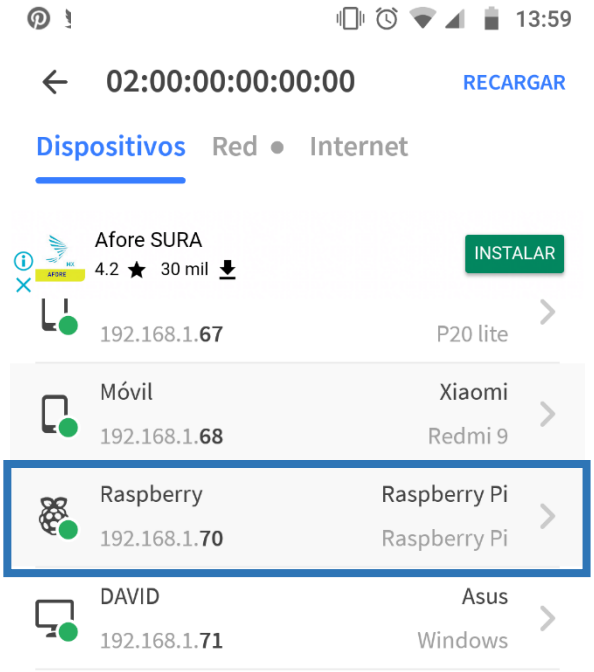

Ilustración 14 Consulta de direcciones IP

Después de realizar el escaneo de la red es posible ubicar la dirección IP proporcionada por el modem local y también es posible observar los puertos abiertos que puede tener el dispositivo, en este caso se observan dos abiertos en el dispositivo, como se muestra en la ilustración 15.

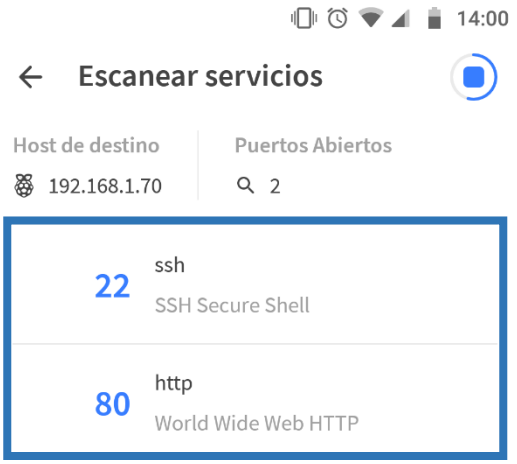

Ilustración 15 Consulta de puertos habilitados

Ya obtenida esta información se proporcionan las credenciales correspondientes en el software putty y se obtiene el acceso y ahora es posible trabajar con las configuraciones necesarias para habilitar los protocolos propuestos para este proyecto. En la ilustración 16 se muestran las credenciales y la dirección IP con la que contaba Raspberry en ese momento y realizar la conexión por ssh.

Cabe mencionar que al no tener una dirección IP estática esta cambia constantemente, por eso propongo el uso constante de Fing o NMAP cuando no existe una dirección IP estática.

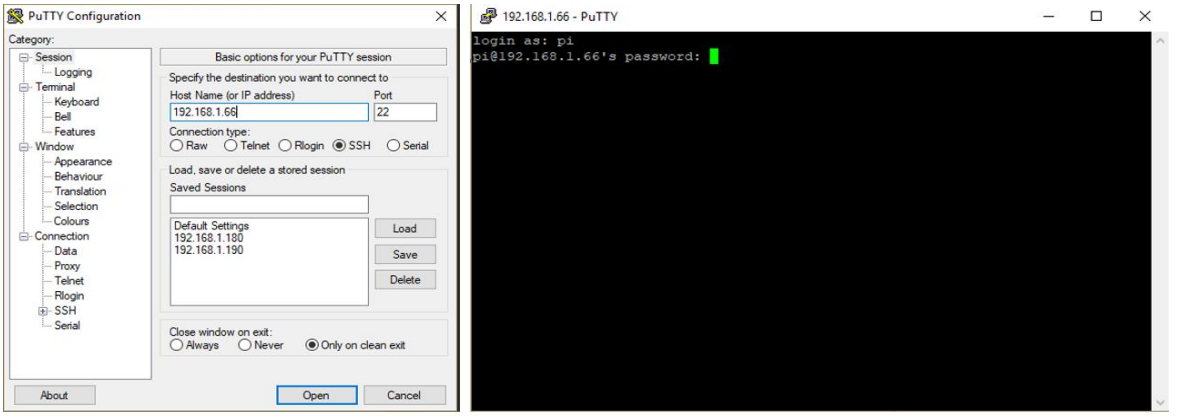

Ilustración 16 Inicio de sesión para Raspberry Pi, desde PUTTY

También se realizó una conexión remota desde una máquina virtual como experimentación de conexiones remotas desde cualquier otra máquina en la red y para obtener imágenes mejor visibles, esta máquina virtual debe estar

configurada en su interfaz de red como adaptador puente para poder establecerla dentro de la red LAN y pueda tener contacto fuera del Host real, en este caso se utilizó el sistema operativo Centos 7, donde se ejecuta el siguiente comando para realizar la conexión ssh –x pi@192.168.1.72 y posteriormente solicitar la contraseña con la que fue configurada Raspberry, el ejemplo se observa en la ilustración 17.

```
[root@localhost mqtt]# ssh -x pi@192.168.1.72
pi@192.168.1.72's password:
Linux raspberrypi 4.19.66+ #1253 Thu Aug 15 11:37:30 BST 2019 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Oct  7 16:29:17 2019 from 192.168.1.69
pi@raspberrypi:~ $ sudo su
root@raspberrypi:/home/pi# clear
root@raspberrypi:/home/pi# ls
2019-10-01-081036_1366x768_scrot.png  Downloads     Pictures      Videos
2019-10-01-081043_1366x768_scrot.png  MagPi         Public
Desktop                               Music         python_games
Documents                             oldconffiles  Templates
```

Ilustración 17 Conexión remota desde terminal

Como primera configuración a realizar en este proyecto fue eliminar ciertas aplicaciones que vienen junto con el sistema operativo para tener más espacio en la memoria micro SD y reducir un poco el tiempo cuando se requirió realizar las actualizaciones para los repositorios de Raspbian, es este caso se eliminó Wolphram Mathematica y Libre Office, ya que no fueron relevantes para utilizarlas en este proyecto. Una vez eliminadas las aplicaciones, se realizó la actualización de los repositorios e iniciar con la implementación del prototipo requerido, (Eames, 2016). Ya eliminado el software se realizó la primera configuración para el desarrollo del proyecto, en este caso se inició con la creación de un punto de acceso (Access Point) Wireless.

3.2 Creación de Access Point

El primer paso fue la configuración del Access Point, la cual tomaría la conexión a Internet a través de la interfaz Ethernet proporcionando conexión inalámbrica por medio de la tarjeta de red WiFi, para ello se realizaron los pasos correspondientes que se agregaron en el anexo 1.2 para configurar la Raspberry como un punto de acceso, donde se le asigna como SSID Wi-Pi y la contraseña como ISET-UACM además la tarjeta de red con la que trabaja

es del tipo *n* que funciona por medio de la interfaz USB. (THEORY, 2018)

Después de esta configuración se logró determinar que la versión del Kernel del SO utilizado contenía una versión exclusivamente para Raspberry Pi3 la cual no proporcionaba ninguna lectura de tarjeta de red inalámbrica en la interfaz USB por lo cual se procedió a implementar una versión de Kernel antigua. La versión que se le asignó fue la 3.10.18+, posteriormente se instaló el Driver para la tarjeta de red (TP-LINK WN725N) que también funciona para otras versiones de tarjetas.

Se realiza la configuración para que se habilite una WLAN y el Kernel busca esta red, así como el Driver de la tarjeta de red se realiza la consulta de las interfaces habilitadas con el comando ifconfig y se encuentra la red inalámbrica como wlan0 como se muestra en la ilustración 18, todos los pasos detallados se encuentran en el Anexo 1.2 al Anexo1.6.

```
root@pi:/home/pi# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.72  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::ba27:ebff:fecb:67f4  prefixlen 64  scopeid 0x20<link>
        inet6 fda4:ba76:3497:c600:ba27:ebff:fecb:67f4  prefixlen 64  scopeid 0x0<global>
        inet6 2806:107e:8:20f6:ba27:ebff:fecb:67f4  prefixlen 64  scopeid 0x0<global>
        ether b8:27:eb:cb:67:f4  txqueuelen 1000  (Ethernet)
        RX packets 19813  bytes 1283283 (1.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14257  bytes 4388435 (4.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 238  bytes 36353 (35.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 238  bytes 36353 (35.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.1  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::42a5:efff:fedc:46bf  prefixlen 64  scopeid 0x20<link>
        ether 40:a5:ef:dc:46:bf  txqueuelen 1000  (Ethernet)
        RX packets 5  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1  bytes 112 (112.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@pi:/home/pi#
```

Ilustración 18 Dirección IP para wlan0

Con esto se comprobó que el Kernel detecta a la tarjeta de red inalámbrica, la cual se encuentra lista para trabajar, y deberá estar proporcionando una red WiFi con el nombre de RPI_PA la cual se configuró. De esta forma se concluyó la habilitación de la comunicación Wireless a nuestra Raspberry Pi 2B, en la ilustración 19 se muestra físicamente a Raspberry trabajando con las dos interfaces.

Ilustración 19 Raspberry con tarjeta de red inalámbrica y conexión con cable UTP

Se realizó una conexión desde un SmartPhone, en donde aparece con el nombre que se asignó en la configuración del Access Point, y con ello se pudo confirmar el funcionamiento del protocolo de comunicación, concluyendo con la primera fase de este proyecto. En la ilustración 20 se muestra la lista de conexiones cercanas donde fue posible obtener comunicación a Internet desde un SmartPhone.

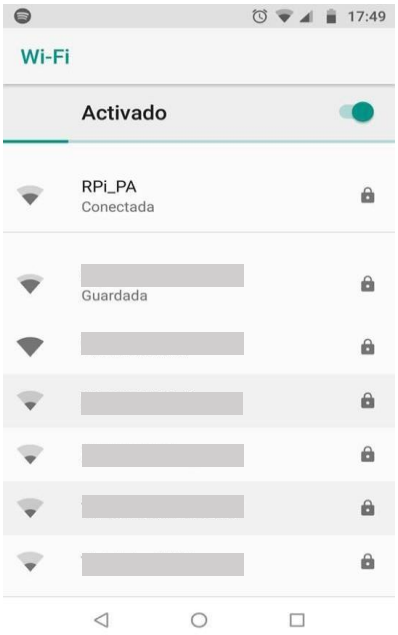

Ilustración 20 Confirmación de conexión WiFi

3.3 Prueba transmisión de datos por MQTT

Una vez habilitada la comunicación WiFi se descargó el bróker de Mosquitto en Raspberry el cual integra el protocolo de comunicación MQTT y MQTT-

Client para la creación de redes de sensores WiFi y de esta forma pueda recibir los datos que son enviados desde los dispositivos.

Para este caso se realizaron pruebas desde la tarjeta de desarrollo NodeMCU/ESP8266, los cuales fueron programados en Arduino para enviar datos por el protocolo MQTT, en Raspberry se creó un TOPIC para recibir el mensaje de ESP8266 NodeMCU y se realizó la primera prueba de comunicación.

Dicha descarga se realizó con el siguiente comando, el cual es encargado de descargar el servicio desde el repositorio en mosquitto.

sudo wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key

Posteriormente se descargó la lista de repositorios de Mosquitto para la versión del SO Raspberry Strech

sudo wget http://repo.mosquitto.org/debian/mosquitto-stretch.list

Se instala el Broker de Mosquitto

apt-get install mosquitto

Y para finalizar se instaló el cliente MQTT en Raspberry

apt-get install mosquitto-clients

Esto es una descripción resumida donde se trabajan los comandos principales para la implementación de Mosquitto, sin embargo la instalación completa se encuentra en el anexo 1.8.

Para lograr la transmisión de datos desde NodeMCU/ESP8266 se utilizan herramientas específicas, como la biblioteca PubSubClient la cual se descargó en el IDE de Arduino, dicha biblioteca ayuda a que el micro-controlador pueda crear y subscribirse a un TOPIC en el cual se manda un dato recolectado a lo cual denominamos publicación. En el brocker de Mosquitto se crea el mismo TOPIC o tema para que las tarjetas de desarrollo justamente publiquen sus datos recolectados.

La estructura del Topic para que la red de sensores se subscriba, se ejecuta de la siguiente forma, **mosquitto-h localhost –t mosquitto/sensor/temperatura**, como ejemplo. Esta dirección de

subscripción contiene dos caracteres importantes los cuales se enumeran enseguida.

- **-h**, indica el nombre del bróker o host con el que estamos trabajando
- **–t** indica lo que sucederá después de haber realizado la conexión entre sensor y Raspberry, esto es definir el TOPIC o tema a subscribir

El TOPIC a subscribirse debe estar definido tanto en ESP8266 como en Raspberry para realizar publicaciones, para este ejemplo se creó el TOPIC /sensor/temperatura, que es donde se publicarán todos los datos que recolecten de los sensores subscritos a este tema.

De esta manera se tiene la posibilidad de publicar uno o varios mensajes en el TOPIC indicado en ESP, y el resultado se obtiene de la forma en que se muestra en la ilustración 21.

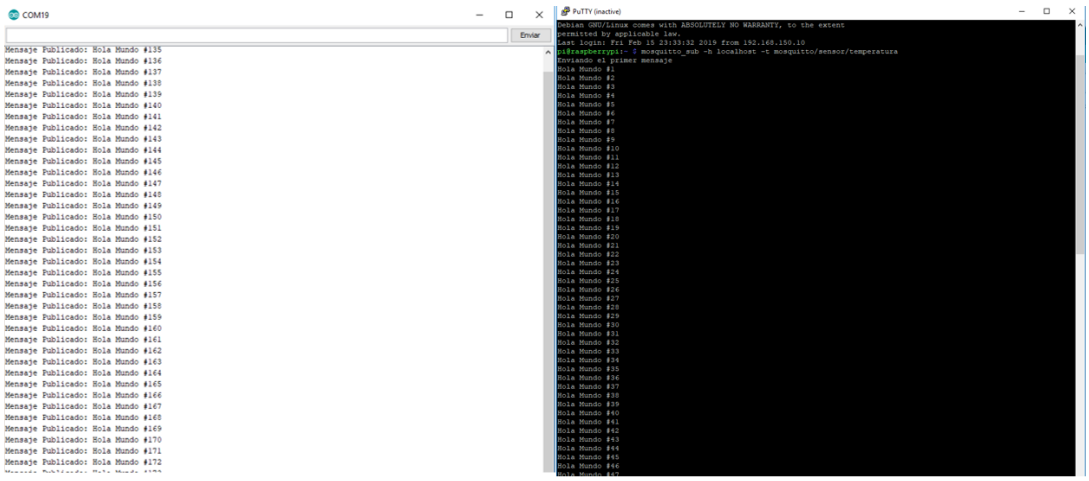

Ilustración 21 Prueba de datos enviados por MQTT desde NodeMCU a Raspberry

En ella se muestra la prueba de comunicación entre un ESP8266 y la Raspberry, del lado izquierdo se muestra un número de mensajes lanzados, los cuales al mismo tiempo se ven reflejados en la consola de Arduino mientras (lado izquierdo) mientras que del lado derecho se observa la recepción de los mensajes en la terminal de Raspberry.

En Arduino se crea un programa con ayuda de las bibliotecas WiFiClient y PubSubClient junto con su respectiva programación estructurada para obtener el resultado mostrado anteriormente, la cual es bastante sencilla y

donde se deben reconocer un par de comandos pertenecientes a las bibliotecas utilizadas.

El programa de publicación para el cliente ESP8266 se divide en cuatro secciones, definidas en el siguiente orden:

- Bibliotecas y Variables (SSID, PASSWORD, SERVER MQTT)
- Instancias y Funciones
- Setup
- Loop

En la primera sección se definen las bibliotecas ESP8266WiFi y PubSubClient, donde la primera biblioteca proporciona las funciones para que el dispositivo realiza conexión a la red inalámbrica mientras que PubSubClient proporciona las funciones para trabajar con el protocolo MQTT.

Para el caso de las variables, estas son declaradas como strings para proporcionar las credenciales al router o a la misma Raspberry para realizar una conexión con la red.

En la segunda sección se definen las instancias para ejecutar las funciones de conexión a la red, para el caso de WiFiClient, y el modo de conexión con el bróker MQTT, para la instancia PubSubClient client(espClient).

Para las funciones, se crea una primera función llamada void callback () que va de la mano con la forma en que trabaja MQTT con el método de subscripción y publicación; esta función crea strings vacíos para guardar el TOPIC a subscribirse y el mensaje, los cuales se llenan con ayuda de un contador completando el topic y el mensaje que se va a lanzar al bróker, el diagrama de flujo es el que se muestra en la ilustración 22.

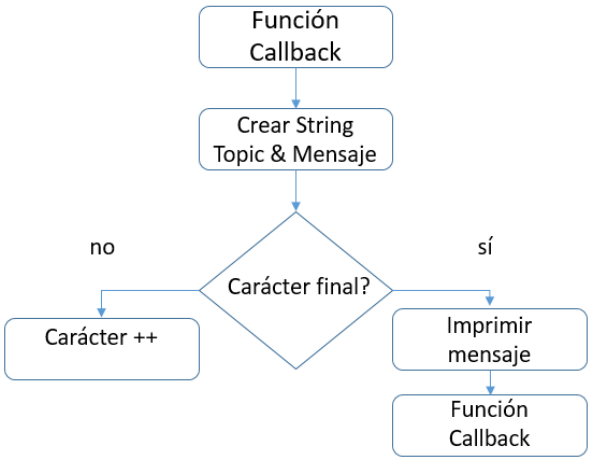

Ilustración 22 Diagrama de función Callback

Mientras que la función void reconnect () se encarga de reconectar el dispositivo al servidor MQTT y subscribirlo al TOPIC asignado para que realice la publicación del mensaje, en caso de conexión fallida, el diagrama de flujo se muestra en la siguiente ilustración.

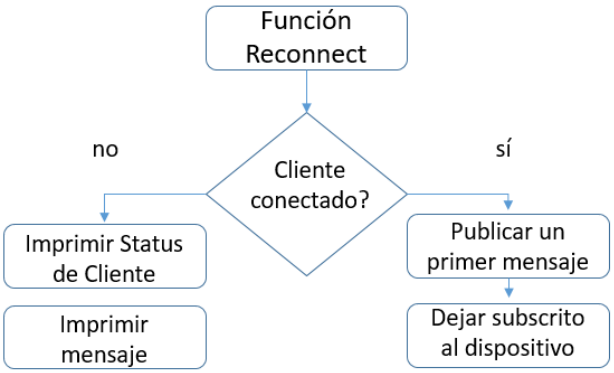

Ilustración 23 Diagrama de función Reconnect

Cuando se conecta con el servidor lanza un primer mensaje y lo deja subscrito a ese topic para lanzar el mensaje “Hola mundo” dentro del bucle de Arduino.

En la tercera sección se define la velocidad de comunicación con el puerto serie, se inicia la conexión con la red WiFi, ejecuta la conexión con el servidor MQTT, la función callback y del mismo modo notifica al usuario por la terminal de Arduino cuando existe una conexión fallida.

La cuarta sección del programa ejecuta la función reconnect () en caso de falla, y ejecuta un bucle de publicación en el bróker con el TOPIC y el mensaje Hola mundo cada que se presiona el botón. En la ilustración 24 se muestra un diagrama de flujo general del programa.

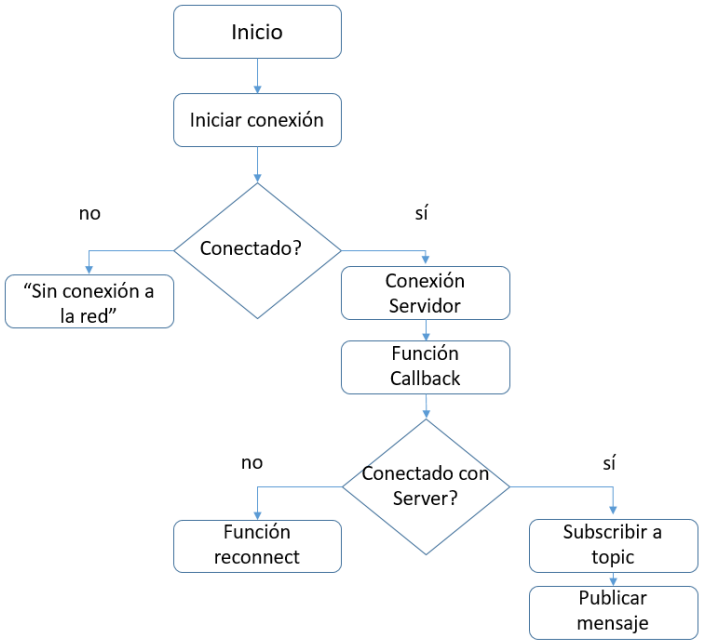

Ilustración 24 Diagrama de programa general

Una vez que se realizó la configuración y la programación en el IDE de Arduino y Raspberry es posible observar como una serie de datos son enviados a Raspberry con una conexión punto a punto (P2P), pues no se utiliza ningún intermediario entre NodeMCU y la placa de desarrollo Raspberry pi 2B, como regularmente se propone en algunos tutoriales que se pueden encontrar en la Red, en la ilustración 25 se muestra un diagrama de interacción de los elementos implementados para la creación de este prototipo donde se representa la conexión y la transmisión de los datos.

Ilustración 25 Esquema de comunicación ESP y Raspberry

El diagrama de la ilustración 25 representa la interacción del sensor con NodeMCU, directamente, en este caso se simuló un sensor con un

pushbutton para que mande un mensaje cuando este sea activado. NodeMCU transmite el dato inalámbricamente por medio de la interfaz Wireless hacia la tarjeta de red WiFi en Raspberry y por medio la subscripción MQTT que se habilita en Raspberry es como se logra imprimir el dato en consola.

3.4 Habilitación de protocolo XBee

El proceso que se llevó a cabo para realizar la habilitación del radio XBee, constó en tres pasos definidos por:

- Interconexión del hardware, este se realizó con la conexión por medio de los pines GPIO del tipo serial con el radio XBee para su funcionamiento y realizar la recolección de los datos que manda Arduino

- Configuración de Radios XBee para abrir el canal de red con ello realizar la transferencia de datos

- Por último la programación de los mismos tanto en Arduino como en Raspberry en los cuales se utilizaron dos lenguajes uno en el IDE de Arduino y para Raspberry se utilizó el lenguaje de Python (ZORRILLA, 2015)

En primera instancia, como ya se mencionó anteriormente se necesita abrir un canal de comunicación entre ambos radios, los cuales están definidos por coordinadores, enrutadores y end-points para crear una red de XBee, en este caso se realizó únicamente la configuración de un coordinador y un end-point para habilitar una comunicación punto a punto, el coordinador es el que se encarga de abrir el canal de comunicación hacia el end-point, recibir el dato transmitido inalámbricamente, y llevarlo hacía Raspberry a través de la comunicación serial en la interfaz GPIO como se muestra en la ilustración 26. El proceso completo de configuración se puede consultar en el anexo 1.9. (Gonzalez, 2016)

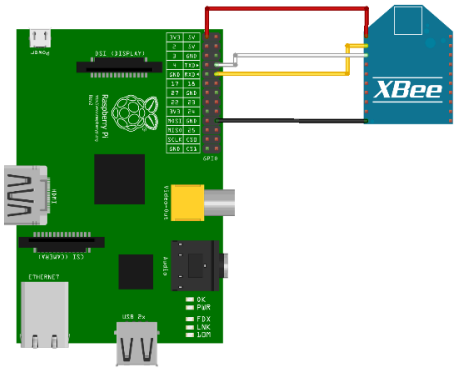

Ilustración 26 Conexión de Raspberry con radio XBee

En un principio se contemplaba usar radios ZigBee pero debido a los fallos que presentaban los radios, se consideró utilizar una tecnología básica como XBee modelo S1, el cual no proporcionó complicaciones para la lectura y la configuración de los radios y de este modo se realizó la prueba de comunicación.

Se probó el funcionamiento de los radios XBee utilizando las dos versiones del software XCTU, versión 6.3, versión 5.2.8.6. Se probaron los radios con el software XCTU y se leyeron los parámetros con los que cuenta, se configuraron para que funcionen los puertos de comunicación. (Caminati, 2016)

Una de las pruebas se realizó utilizando comandos API desde XCTU en el end-point hacia el coordinador, ambos conectados directamente a una computadora de escritorio, se envió un mensaje de “hola mundo” el cual fue recibido en la terminal del coordinador y de esta forma se confirmó el funcionamiento correcto de los radios, el resultado se muestra en la ilustración 27.

Ilustración 27 Primera prueba de conexión

La conexión de los radios XBee tanto en Arduino como en Raspberry se realizó con la comunicación serial (Tx-Rx).

Para Arduino la comunicación hacia el radio XBee se realiza con ayuda del Shield XBee el cual proporciona un ensamble plug and play y los datos enviados obedecen al comando comando Serial.print() y Serial.println() estos datos se pueden observar tanto en IDE de Arduino como en la terminal de Raspberry una vez que se realizó la comunicación.

Para la conexión del radio XBee con Raspberry se realizó una conexión cableada con los pines de alimentación (3.3v, GND) y con el puerto de comunicación Serial (Tx-Rx), ya realizada la conexión del hardware se ejecuta el programa realizado en Python con ayuda de la biblioteca python-serial, para realizar una lectura desde este puerto.

Y se realizó una segunda prueba para confirmar la comunicación desde el end-point compilando en Raspberry y ejecutando comandos desde la terminal de XCTU, donde se envió el mensaje, hola mundo como se muestra en la ilustración 28.

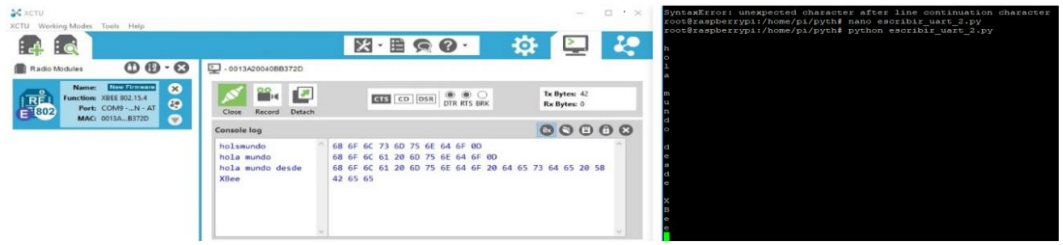

Ilustración 28 Mensaje enviado desde la terminal de XCTU y mensaje recibido en la terminal de Raspberry

Posteriormente se realiza la última prueba, haciendo una automatización en la recolección de datos de un sensor de humedad bajo tierra en conjunto de la comunicación con Arduino-Xbee quien es el transmisor con el receptor en Raspberry para que sean almacenados en un archivo de texto dentro del espacio que tiene disponible la memoria micro SD, en la ilustración 29 se muestra el esquema de la comunicación punto a punto utilizando el protocolo XBee.

Ilustración 29 Esquema de comunicación con Xbee-Raspberry

La programación de su funcionamiento trata de dos algoritmos que como bien ya se mencionó Arduino recolecta datos del sensor y transmite la información hacia Raspberry, esta los recibe y los guarda en un archivo de texto.

El funcionamiento de la ilustración 30 muestra el diagrama de flujo del programa realizado en Arduino para la comunicación con Raspberry a través del protocolo XBee.

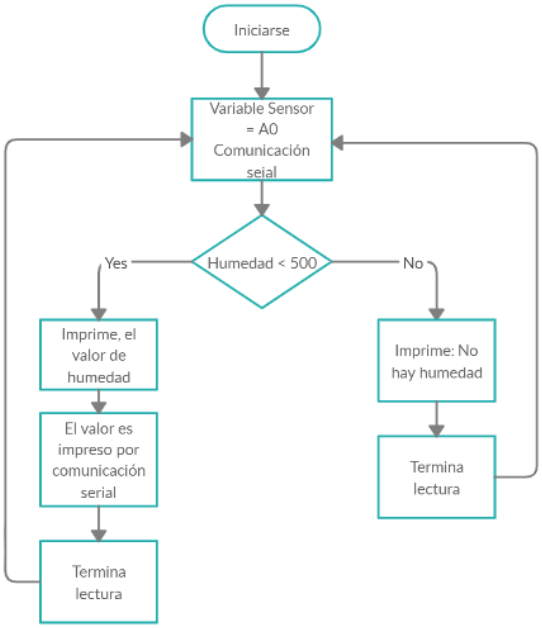

Ilustración 30 Programa para comunicación por XBee

En este programa se le agrega un sensor bajo tierra que recibe la información por el pin analógico cero (A0), e imprime en la consola el mensaje de encendido si el rango de los valores arrojados por el ADC es menor a 500 ADU (Unidades Analógicas a Digitales) y este mensaje a su vez es transmitido hacia el coordinador, en la siguiente imagen se muestra el ensamble real de Arduino y Shield XBee, el algoritmo es el que se muestra en la ilustración 31. (ELECTRONICS, 2016)

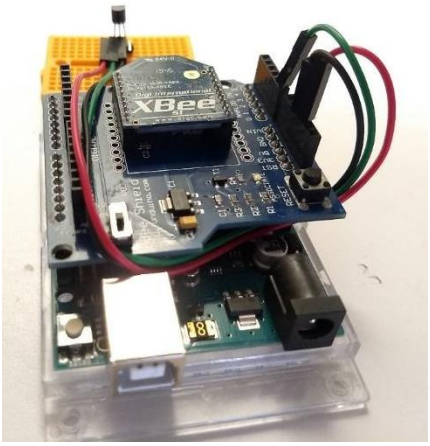

Ilustración 31 Ensamble de Arduino con Shield XBee

Para la programación en el receptor el cual se encuentra a cargo de Raspberry y el coordinador de XBee se realiza en lenguaje Python donde se le indica al programa que interfaz debe leer, en este caso la lectura es tomada desde los puertos ttyAMA0 que pertenecen a los pines Tx y RX, además de ello tomamos los datos recibidos de la interfaz guardándolos en variable serial, abre un archivo de texto nombrado xbee_otro.txt y se guardan los datos, termina programa y cierra el archivo de texto, el diagrama de flujo se muestra en la ilustración 32. (decodigo.com, s.f.)

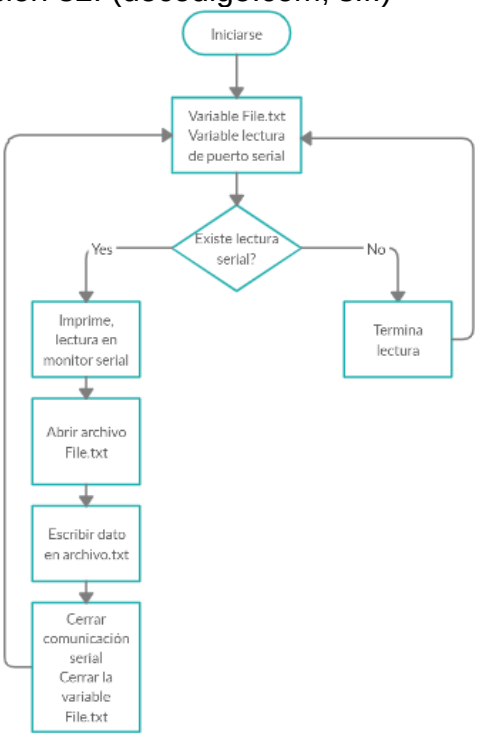

Ilustración 32 Diagrama para la recopilación de datos, python

El diagrama muestra una consulta del puerto serial constante el cual si no recibe nada vuelve a preguntar si llegó un dato y cuando este dato llega lo guarda en un archivo de texto, cierra la comunicación serial y reinicia el código. En la ilustración 33 se muestra la pantalla de la consola de Arduino a la izquierda y a la derecha se muestra la consola de Raspberry de los datos recibidos se muestra el rango de los valores del ADC entre 300 y 500 ADU (Unidades Analógicas a Digitales) para determinar que la tierra está húmeda y un rango de 600 a 900 ADU (Unidades Analógicas a Digitales) para determinar la tierra árida, estos se muestran en pantalla tanto en la terminal de Arduino como en la terminal de Raspberry.

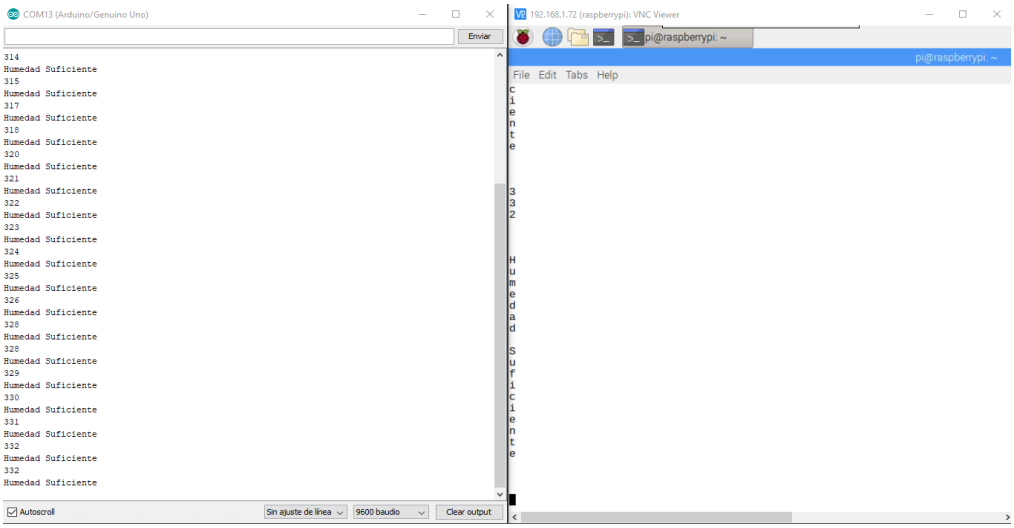

Ilustración 33 Mensaje desde Arduino-XBee a Raspberry

En la ilustración 34 se muestra una simulación de una bomba de agua activada cuando se detectan los valores del ADC mayores de 300 ADU, los datos recibidos se guardan en el documento de texto los cuales aparecen enlistados con el mensaje de Encendido.

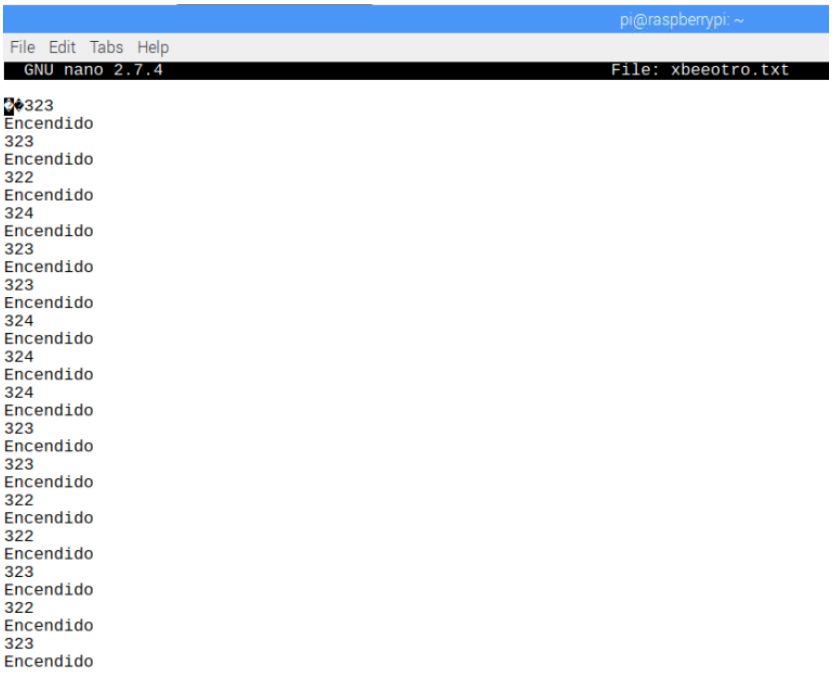

Ilustración 34 Datos guardados en archivo .txt

3.5 Habilitación del protocolo Bluetooth

Para colocar en funcionamiento este protocolo de comunicación, los pasos fueron muy similares a los que se realizaron en XBee, solo que para esta ocasión fue necesario conectar el modulo Bluetooth o HC-05 tanto en Arduino con ayuda de los pines Tx-Rx como en Raspberry en la interfaz USB, que quedaba disponible utilizando un convertidor TTL a USB, ya que Raspberry solo contiene una conexión serial por medio de las interfaces de GPIO, la cual fue ocupada en por el radio XBee. En la ilustración 35 se muestra el esquema de conexión del HC-05 con el convertidor TTL-USB (Josh, 2016)

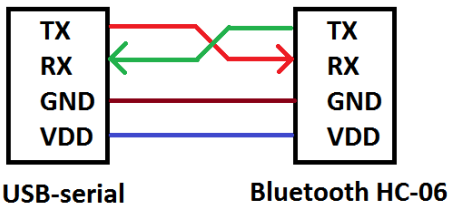

Ilustración 35 Diagrama para la conversión de serial a USB

Una vez conectados los dispositivos a Raspberry para habilitar el protocolo Bluetooth, también se realizó una programación en Python para que pudiera recibir los datos, en la ilustración 36 se muestran los dispositivos físicos utilizados.

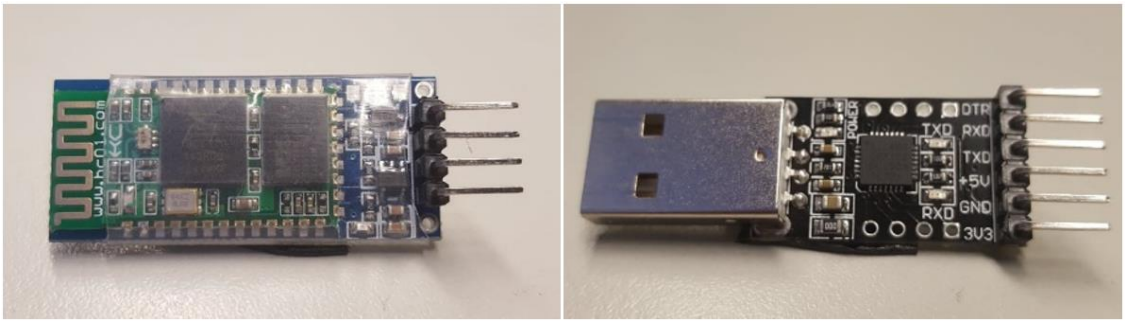

Ilustración 36 A la izquierda HC-05 y a la derecha convertidor Serial-USB

El programa realizado en Python, nombrado como hc-05.py, se encargó de realizar la lectura por medio del serial e imprimirla en la misma terminal y guardar los datos en un archivo de texto que fue nombrado bluetooth.txt, este programa trabaja de forma similar al que fue utilizado en los módulos XBee, el diagrama de flujo se muestra en la ilustración 37.

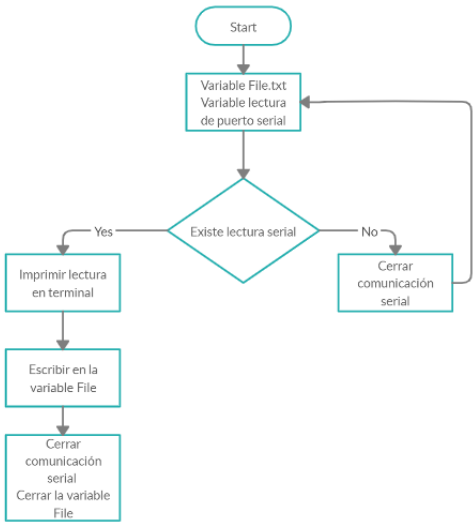

Ilustración 37 Diagrama de comunicación serial Bluetooth

Posteriormente se realizó con la aplicación APP Arduino Bluetooth para Android en el cual se manda el mensaje escrito en la interfaz de la aplicación

dicho mensaje aparece en la terminal de Raspberry con el mensaje de “Hola mundo desde Smartphone”, el funcionamiento aparece en la ilustración 38, de esta forma confirmamos el funcionamiento de tres protocolos funcionando al mismo tiempo y cada uno recibiendo datos por cada una de las interfaces asignadas hasta el momento.

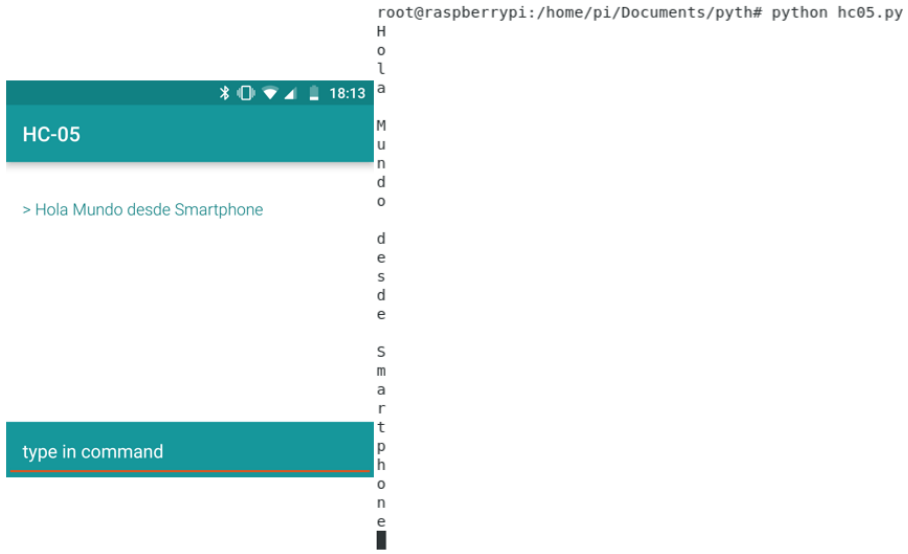

Ilustración 38 Primera prueba para el envío de datos por protocolo Bluetooth

En la ilustración 38 se observan los datos enviados a través de la aplicación del Smartphone y en el lado izquierdo se muestran los datos recibidos en la terminal de Raspberry.

Como bien se comentó anteriormente el programa, además de recibir los datos, el programa guarda los datos en un archivo de texto nombrado Bluetooth, como se muestra en la ilustración 39.

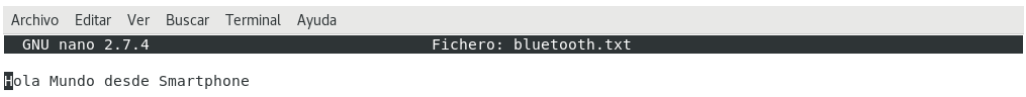

Ilustración 39 Primera prueba para el almacenamiento, para Bluetooth

Para estas alturas cada medio de comunicación recibe datos desde sus interfaces habilitadas y guardan datos en un archivo de texto para su consulta en algún otro momento posterior.

3.6 Habilitación del protocolo por Radiofrecuencia

Para la habilitación de este protocolo se utilizaron dos módulos de radiofrecuencia en este caso Raspberry utiliza el receptor conectado en los pines GPIO, mientras que para Arduino se realiza una conexión de alimentación junto con un sensor de temperatura conectado a uno de los pines analógicos, en la ilustración 40 se muestran los esquemas de conexión.

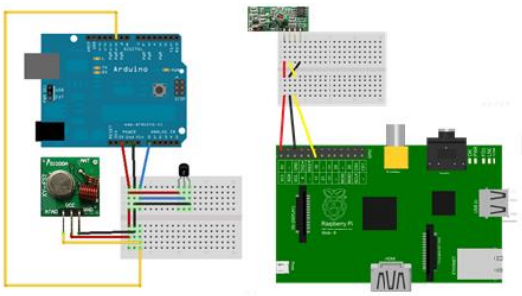

Ilustración 40 Esquema Arduino - Raspberry

Para transmitir datos por este medio es necesario de un protocolo de comunicación debido a que estos módulos, únicamente son capaces de enviar estados altos o estados bajos. Para hacer funcionar el módulo de radiofrecuencia se utilizó el protocolo de WiringPi, los pasos para realizar la descarga se encuentran en el anexo 1.13. (Gordon, Wiring Pi, 2019)

```
Reutilizando la conexión existente a [uc9edd01c8bbce36caafcce98909.dl.dropboxusercontent.com]:443.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 7781 (7.6K) [application/zip]
Grabando a: “rpi.zip”

rpi.zip                     100%[=========================================>]   7.60K  --.-KB/s    in 0s

2019-10-07 20:01:36 (19.8 MB/s) - “rpi.zip” guardado [7781/7781]

root@raspberrypi:/usr/src/rasp433# ls
rpi.zip
root@raspberrypi:/usr/src/rasp433# unzip rpi.zip
Archive:  rpi.zip
  inflating: rfreceive.cpp
  inflating: rftester.cpp
  inflating: RCSwitch.cpp
  inflating: RCSwitch.h
root@raspberrypi:/usr/src/rasp433#
```

Ilustración 41 Contenido del repositorio WiringPi

Una vez realizada la instalación de este repositorio se inició la compilación del protocolo para realizar la primera prueba de comunicación, enviando un dato que se tomó desde un sensor de temperatura utilizando Arduino, y de esta forma verificar que el dato se haya recibido.

Para este protocolo debo mencionar que el receptor no fue alimentado con el voltaje de Raspberry ya que sus pines de 5v proporcionaban valor menor por lo fue necesario alimentar con una fuente externa, para este caso con ayuda de otro Arduino.

En la ilustración 42 se observa únicamente el primer Test que se realiza al receptor, del lado izquierdo y el voltaje incorrecto que proporcionó Raspberry, del lado derecho.

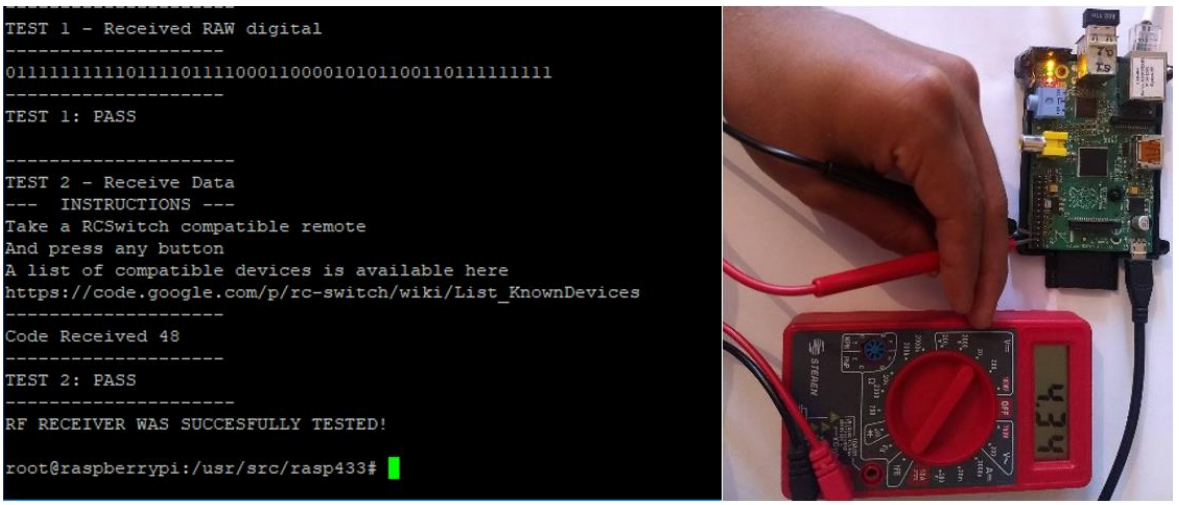

Ilustración 42 Error de recepción de dato y voltaje emitido en puerto GPIO

Una vez conectada la fuente de alimentación al receptor el dispositivo comenzó a recibir los datos.

Para enviar los datos desde Arduino utilizando los módulos de Radiofrecuencia, se realizó una programación utilizando la librería RCSWITCH, la cual proporciona comunicación con el software del protocolo que fue instalado en Raspberry, a partir de ello se desarrolló el programa para recolectar los datos de un sensor de temperatura analógico y comunicarla con el transmisor hacia Raspberry, el diagrama de flujo del programa se muestra en la ilustración 43.

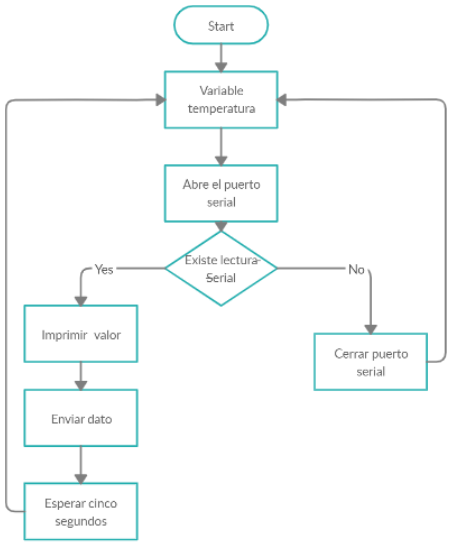

Ilustración 43 Diagrama para transmisión de datos por RF

El programa es muy similar a los programas anteriores donde se realizan lecturas a través de un pin analógico de Arduino el cual guarda sus valores en una variable y son transmitidos por el módulo de radiofrecuencia.
De esta manera se realizaron las pruebas de conexión y efectivamente el dato que fue recibido e impreso en la terminal de Raspberry, el cual mostró la temperatura que estaba captando en ese momento el sensor LM35.

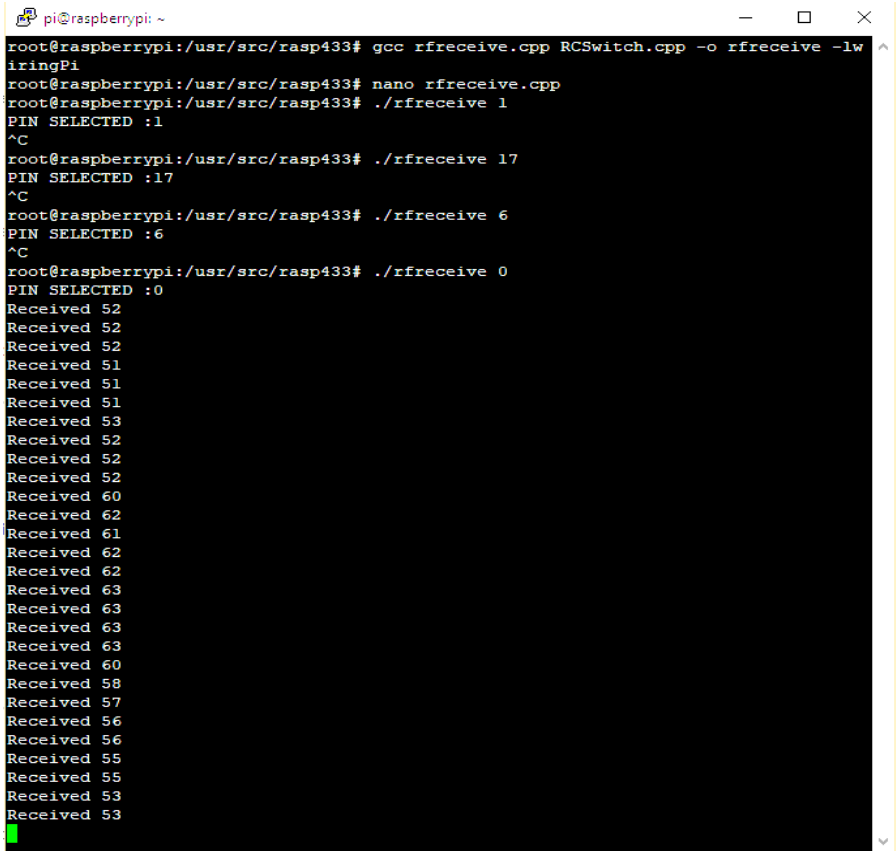

Ilustración 44 Prueba del dato recibido con el módulo de radiofrecuencia

Con estos protocolos funcionando en Raspberry Pi 2A, se ha creado una gran parte del desarrollo del Gateway, la recepción de datos que provienen de diferentes protocolos de comunicación, con ello solo falta realizar el último paso para completar el desarrollo y este trata de la transferencia de datos del Gateway haca un servidor.

3.7 Transmisión de Datos Gateway-Servidor

En este punto el Gateway recibe los datos que de los sensores y los guarda en un archivo de texto, pero para que el prototipo se finalice de forma satisfactoria se considera que estos datos provenientes de cada uno de los protocolos de comunicación sean transmitidos a un servidor, para ello se implementó de la forma más básica creando un websocket entre Raspberry y una computadora portátil con la finalidad de enviar los archivos de texto que se crean en el Gateway, utilizando un protocolo de comunicación para su almacenamiento, el websocket fue implementado con el lenguaje Python. Este se encarga de transferir archivos de texto utilizando el puerto 20 que

utiliza el protocolo FTP”. (UNDERCODE, 2013). En la ilustración 45 se describe el diagrama de flujo del funcionamiento de los programas utilizados tanto para el cliente como para el servidor.

Ilustración 45 Esquemas Cliente-Sevidor, Python

En el caso del WebSocket en Python primero se utilizó la librería socket, posteriormente se creó el objeto *der* para trabajar con WebSockets a continuación se define el puerto y el modo en que trabaja el socket, en esta ocasión en modo escucha, se define el parámetro de las conexiones que tendrá el servidor, que para este caso se definió en modo escucha.

Cuando se inicializa el funcionamiento del objeto, y la habilitación es correcta, se reciben los datos utilizando el método *recv,* se imprime en consola la notificación recibiendo con los datos que se obtuvieron y de donde provienen, posteriormente se le notifica al cliente, una vez hecho esto se cierra la instancia del socket y se imprime el estatus de conexión, esta breve descripción se encuentra ejemplificada con el diagrama de flujo izquierdo la ilustración 45.

Para el WebSocket Cliente es el que se muestra a la derecha de la ilustración 45, el cual para su creación se declara la variable host con su respectiva dirección y el puerto a utilizar, se importa la librería socket y se crea el objeto socket el cual trabajara en modo cliente, después se inicializa la conexión al servidor.

Mientras la habilitación de los parámetros sean correctos se inicializa una entrada de datos que el cliente enviara posteriormente mediante la instancia mens = raw_input (“Mensaje>>”) y se crea el objeto que se encargara de enviarlo al servidor obj.send(mens) una vez que se realizó la transmisión de los datos se cierra la instancia del objeto y se imprime notifica en consola dicha instancia. (Vela, 2019)

En este servicio, se implementa como cliente a Raspberry, ya que es ella quien solicita el servicio de transferencia para enviar archivos de texto, mientras el servidor se pone a disposición de escuchar alguna petición de algún cliente quien se le programo con un máximo de diez clientes, en la siguiente imagen se observa al servidor en la espera de una petición de cliente, estos programas igualmente se compilan y ejecutan por medio de consola en la terminal de Windows.

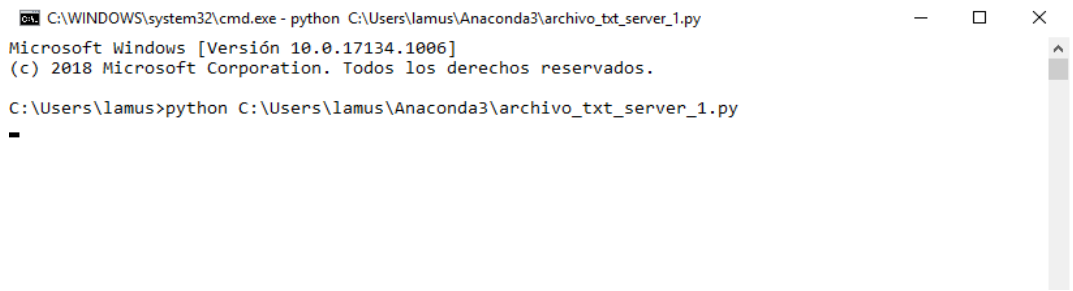

Ilustración 46 Servidor Python en espera

Posteriormente se ejecuta la petición de servicio en Raspberry y este inmediatamente enlaza una conexión con el servidor identificándose por medio de su dirección IP y transfiere el archivo de texto recogido de alguno

de sus directorios y lo transfiere, donde el servidor responde a dicha petición aceptando dicha transferencia, en la ilustración 47 se plasma la petición de cliente al momento que se compila en consola para posteriormente enviar el archivo hacia el servidor lo notifique su recepción del mismo.

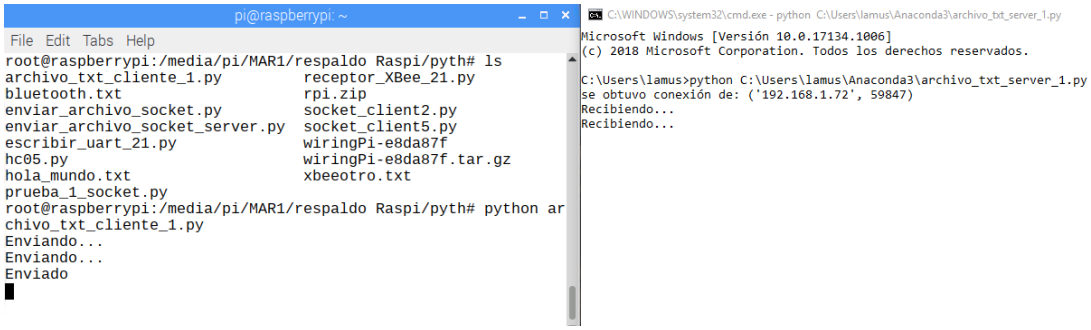

Ilustración 47 Transmisión Cliente-Servidor

La primera prueba consistió en mandar un hola_mundo.txt y este fue recibido exitosamente en el servidor el cual se almaceno directamente en el disco duro de la computadora portátil con la ruta C:\Users\lamus, en la ilustración 48 se muestra la evidencia de un antes y un después de ejecutar los programas

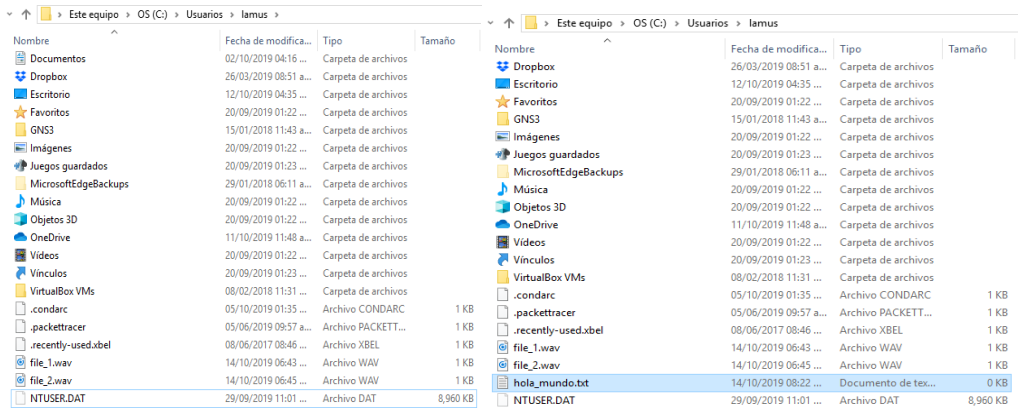

Ilustración 48 Transmisión de archivo .txt al servidor Antes/Después

El siguiente paso fue realizar la transferencia de los archivos que se generaron a partir de los datos recolectados por medio de los protocolos de comunicación que se propusieron para la implementación de este Gateway, la ilustración 49 muestra la ejecución del cliente a la izquierda y el servidor a la derecha.

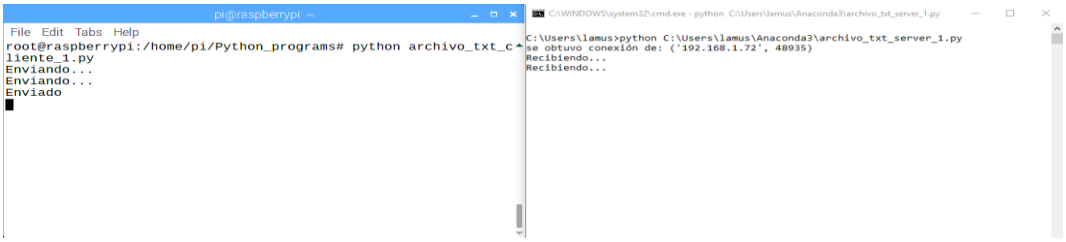

Ilustración 49 Consola Raspberry, enviando, consola Windows, Recibiendo

La ilustración 50muestra la consola de cliente-Raspberry el envío de la información y en el servidor-Windows se muestra la recepción del archivo de texto que se generó en el protocolo de comunicación Bluetooth.

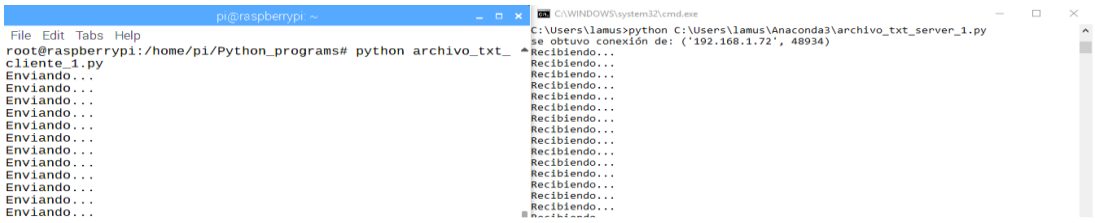

Ilustración 50 Transmisión y Recepción concluida

Posteriormente se muestran los archivos del servidor-Windows recibidos del protocolo de comunicación XBee y Bluetooth.

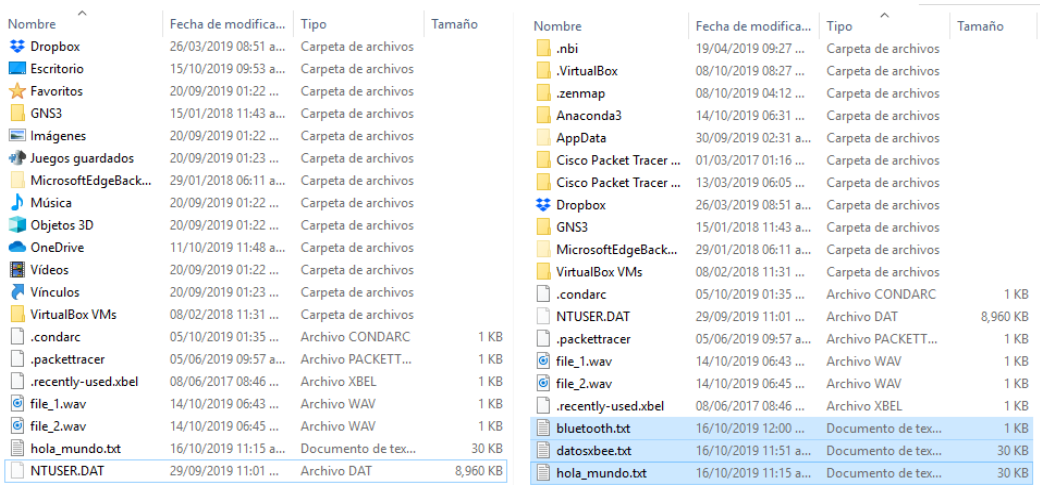

Ilustración 51 Información recibida en el server Antes/Después

Posteriormente se muestran los datos que fueron obtenidos, desde la primera prueba del hola_mundo.txt hasta los siguientes archivos como bluetooth.txt y datosxbee.txt los cuales ya se muestran en un entorno de

Windows y con estas pruebas se da por finalizado el proyecto con un resultado satisfactorio. El prototipo final se describe por el esquema que se muestra en la ilustración 52.

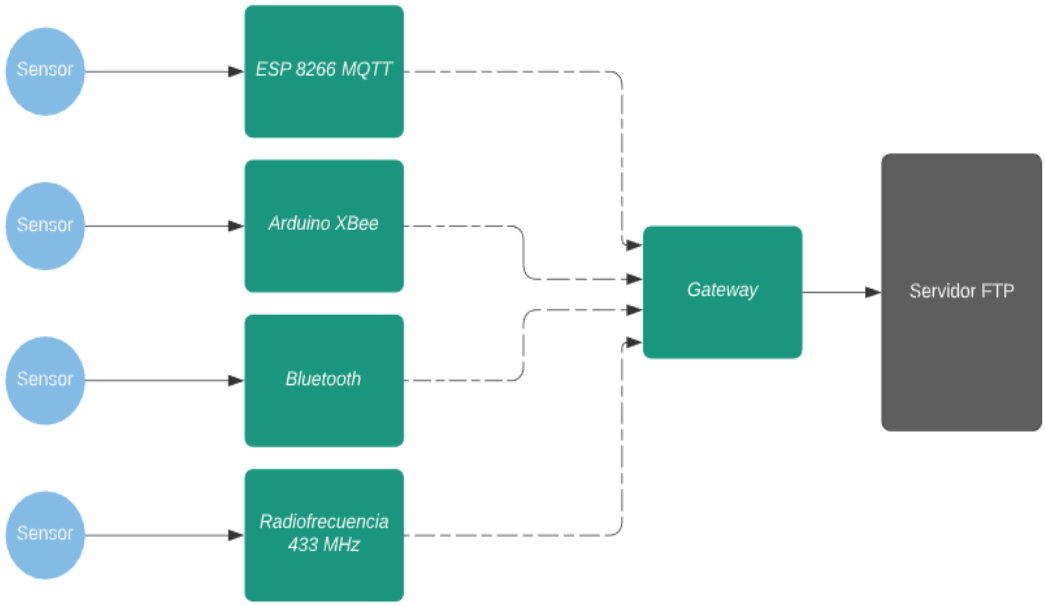

Ilustración 52 Esquema de comunicación finalizada

En esta ilustración es posible observar la interacción que existe entre dispositivos como Arduino, Raspberry Pi y ESP8266 trabajando en conjunto creando tráfico de datos hacia un servidor.

Conclusiones y alcances

La elaboración de este proyecto proporcionó una variedad de conocimientos en la implementación de cada uno de los dispositivos y protocolos de comunicación.

De los dispositivos utilizados el principal y más importante es el uso de la tarjeta Raspberry Pi 2B, que a pesar de su modelo básico fue amigable para la elaboración de este proyecto y proporcionó la posibilidad de trabajar con cada una de las tecnologías desde un nivel físico y un nivel lógico. Del mismo modo proporcionó la posibilidad de conocer un poco más los protocolos que son utilizados para aplicaciones para redes de sensores.

Se obtuvieron conocimientos de estructuras de programación tanto en

lenguaje C como en Python además del uso de diferentes tecnologías como Arduino, ESP8266/NodeMCU y Raspberry, las cuales a pesar de tener semejanzas presenta su forma de trabajar y con ello se le puede sacar provecho.

Por ejemplo, en la implementación del Access Point proporciono vastos conocimientos en la configuración de servicios desde la terminal de LINUX lo cual amplia el panorama para la implementación en sistemas más robustos. Debido al aprendizaje que se obtuvo esta implementación se consumió más tiempo de lo planeado, debido a la evolución del SO, por lo que fue necesario trabajar a prueba y error en la marcha del desarrollo para que pudiera quedar funcional y cumplir la expectativa de utilizar un modelo de comunicación punto a punto.

Hablando de la transmisión de datos por MQTT al final dependió de la programación correcta de ESP el aprendizaje del lenguaje de programación así como el funcionamiento del mismo dispositivo pues este muestra una vasta variedad de trabajo a desde el mismo microcontrolador como las prestaciones que proporciona su protocolo de comunicación.

La configuración de Mosquitto para ejecutar el servicio MQTT en Raspberry también muestra un panorama para su implementación en servicios que puedan ser alojados en un webhosting y una futura programación con lenguajes de programación con Python y su funcionamiento con una base de datos podría ser el siguiente paso a pesar de que en este proyecto solo se hizo una prueba básica puede aportar bastante información.

Con los radios Xbee, se tuvo dificultad para utilizar modelos S2PRO debido a problemas de Midleware, drivers así como problemas físicos en los mismos radios, estos fueron probados pero después se deshabilitaban debido a ello no se pudo realizar los experimentos con esta versión de radio, por ello se utilizaron modelos S1PRO que tienen de mayor facilidad de reconocimiento con la computadora.

Para habilitar la comunicación por Bluetooth se encontró el reto de realizar la conexión con Raspberry debido a que la conexión UART ya se encontraba ocupada por el radio Xbee, con ello se había propuesto la utilización de un multiplexor y realizar un programa para intercalar la lectura de los datos entre XBee y Bluetooth sin embargo se solucionó de una manera más simple con la ayuda del convertidor TTL a Serial quien nos proporcionó una segunda conexión Serial por la interfaz USB en Raspberry del mismo modo se realizó una comparación de utilizar un modelo más reciente como la Raspberry Pi3

B+ en la cual se intentó realizar la comunicación por el medio del puerto Serial , sin embargo en esta versión era necesario desactivar su módulo Bluetooth integrado para habilitar los pines GPIO Serial, sin embargo utilizando la versión Pi 2A o Pi 3b+ la solución del convertidor habría sido factible para ambas.

Para la comunicación por Radiofrecuencia, la mayor complicación que se tuvo fue un enlace que estaba roto al momento de la descarga, del protocolo de comunicación, directa desde la terminal, por lo que fue necesario descargar con Windows, guardarlo en una memoria flash y pegarla en Raspberry, lo describo ya que muchas veces estamos acostumbrados a realizar las cosas de una sola forma, tanto que en ocasiones resultamos enfrascados únicamente un solo método, por lo que debemos buscar la forma de que nuestra mente salga de ese bucle, y con ello desarrollar habilidades para modificar el rumbo de lo que estemos desarrollando y lograr un objetivo.

Y para lograr el objetivo de enlazar la Raspberry con un servidor FTP programado también fue uno de los protocolos que más trabajo costaron esto debido a que no se tenía la suficiente formación en el área programación pero al final se logró implementar proporcionando un proyecto bastante enriquecedor a nivel de aprendizaje.

Otra parte importante que quiero mencionar es la adaptación de nuestra mente al cambio de utilizar una interfaz gráfica a una de texto plano, esto provoca que tengamos que ser más atentos con lo que sucede en una descarga, pues en muchas ocasiones nos muestra un error y no lo percibimos, además los servidores se trabajan de mejor manera utilizando líneas de comando que por interfaz gráfica y de esta forma se proporciona un grado de seguridad en los sistemas proveedoras de servicios

Alcances de proyecto

Considero que este proyecto puede utilizarse para realizar un análisis de datos antes de mandar datos a otro servidor, en pocas palabras hacer Fog Computing, enlazarlo a un servicio en la nube ya sea público o privado, del mismo modo agregar otros protocolos de comunicación como la implementación de tarjetas de red celular para transmitir esos datos a distancias más lejanas utilizando esta infraestructura y que el prototipo salga pueda ser utilizado en un área libre de redes como WiFi, esto es algo que actualmente se utiliza bastante en IoT, y que tiene una mayor funcionalidad.

Considero que al servicio de Mosquitto se le puede sacar mayor provecho

instalándolo también en un servidor con mayores prestaciones, de esta forma hacer un enlace entre este servicio junto con una base de datos y un sistema que pueda mostrarlos en un gráfico hablando en términos muy básicos, no obstante se puede agregar además del enlace de comunicación diferentes tecnologías que hoy en día están siendo bastante implementadas en los servicios informáticos como AI (Artificial Intelligence) este sería un proyecto con bastante peso y de esta forma podríamos decir estamos haciendo IoT en la Universidad, así mismo se podría agregar un nuevo perfil en las especializaciones de carrera en la Universidad, una especialización en IoT considero que sería una buena rama, con ello se actualizaría la carrera con temas de Big Data, AI, desarrollador en Cloud Computing, los cuales son temas nuevos en la industria, la cual va acelerándose cada vez más.

En el proyecto solo se utilizaron protocolos comercialmente conocidos en el mercado y de fácil acceso al público en general, sin embargo existe una gran cantidad de implementaciones para llevarlo a cabo que como toda evolución en el mundo de la tecnología al final se estandarizaran los que mejores prestaciones ofrezcan a las empresas para poder trabajar en la Revolución 4.0, sin embargo esto aún se encuentra en desarrollo y como tal lo que queda es continuar implementando y así abrir el mercado, y no subestimar a los dispositivos que utilizamos como estudiantes pues muchos de estos comparten un enorme potencial siempre bajo la creatividad que tengamos en la mente, pues ella es nuestro limite.

Anexo

Anexo 1.1 Eliminación de software innecesario

Se crea un Access Point con una tarjeta de red por la interfaz USB, con el objetivo de propagar una red inalámbrica para ello se eliminan las herramientas que no utilizamos, las cuales trae incluidas el Sistema Operativo (SO) Raspbian, en este caso Wolfram matemática y open office, proporcionando más espacio libre en la memoria.

Eliminación de Wolfram Mathematica

sudo apt-get purge wolfram-engine //Desinstalar y eliminar todos los datos del programa
sudo apt-get clean //Limpiar los directorios
sudo apt-get autoremove //Remueve todas las dependencias que fueron instaladas con las aplicaciones y que no serán usadas por el sistema

Eliminación de Libreoffice

sudo apt-get remove --purge libreoffice*
sudo apt-get clean
sudo apt-get autoremove

Comando para verificar el espacio que tenemos ahora disponible.

df –h //Información mostrada en unidades Kb, MB y Gb

Con ello se puede obtener un mayor espacio para instalar nuestros recursos para la aplicación de interés debido a que no utilizaremos estos recursos y de este modo podremos dar una mejor utilidad para IoT a Raspberry. Una vez realizado esto podemos hacer una actualización del sistema Operativo.

sudo apt-get update
sudo apt-get upgrade

Esto requiere un tiempo alrededor de 60 a 90 minutos aproximadamente, esto es debido a la cantidad de memoria RAM de 512 MB con la que cuenta la versión de la Raspberry, sin embargo cabe mencionar que para un modelo más reciente como es el caso de la versión Pi 3B+, el tiempo se reduce en la actualización de las librerías. Una vez realizada la actualización del SO, se procede a implementar el Access Point, estos pasos son para implementarlo en Raspberry Pi 2, mientras que para Raspberry Pi 3 b+ es otro tipo de configuración, debido a que esta tarjeta ya cuenta con una comunicación inalámbrica integrada.

Anexo 1. 2 Instalación de DHCP y Hostapd

Después de haber realizado estos cambios se procede a hacer la habilitación del Access Point para posteriormente hacer los cambios en el Kernel e instalar los drivers para la lectura de la tarjeta de red.

sudo apt-get install hostapd isc-dhcp-server
sudo nano /etc/dhcp/dhcpd.conf

Donde en el archivo se comentan los nombres de dominio, ya que en este caso no se van a utilizar y autorizamos el servidor DHCP, así posteriormente al final del archivo agregamos la configuración de subneting, la máscara de red, el rango de direcciones IP para los dispositivos que requieran conectarse y el número de servidores de dominio, quedando de la siguiente forma.

```
subnet 192.168.100.0 netmask 255.255.255.0 {          // IP asignada a WLAN y mascara de Red
   range 192.168.100.10 192.168.100.50;       //Rango de direcciones IP para dispositivos conectados
   option broadcast-address 192.168.100.255;          //Dirección IP para hacer el Broadcast
   option routers 192.168.100.1;                      //Dirección IP para Gateway
   default-lease-time 600;
   max-lease-time 7200;
   option domain-name "local";                        //Nombre del dominio
   option domain-name-servers 8.8.8.8, 8.8.4.4;       //Servidores DNS de Google
}
```

Posteriormente se habilita la red con el nombre de wlan0, en el archivo isc-dhcp-server, INTERFACES="wlan0", posteriormente se deshabilita la tarjeta de red para configurar la dirección que tendrá como Gateway.

```
sudo nano /etc/default/isc-dhcp-server
sudo ifconfig wlan0 down
```

Ingresamos al siguiente repositorio para configurar las direcciones IP, donde la dirección para la interfaz Ethernet se deja a cargo del servidor DHCP del MODEM o Router, y la Dirección IP de la interfaz Wireless se le configura una dirección IP estática.

```
sudo nano /etc/network/interfaces
```

Anexo 1.3 Configuración de red inalámbrica

Una vez ingresado al directorio, se configuran las características de la interfaz wlan0.

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp                    //Asignación de dirección IP por DHCP en interfaz Ethernet

allow-hotplug wlan0                     //Tipo de red inalámbrica
iface wlan0 inet static                 //Tipo de dirección IP, estática
address 192.168.100.1                   //Dirección IP para el Gateway
netmask 255.255.255.0                   //Mascara de red
```

Posteriormente guardamos el archivo y se habilita la dirección IP de la interfaz inalámbrica, seguido de esto, configuramos las características que tendrá nuestra red inalámbrica, donde la red fue llamada Wi-Pi, con contraseña, iset-uacm.

Anexo 1.4 Configuración del SSID Password para red inalámbrica

sudo ifconfig wlan0 192.168.100.1
sudo nano /etc/hostapd/hostapd.conf

Dentro del repositorio, se configura de la siguiente manera, en el cual se determina driver que se utilizará la versión de operación, el nombre de la red a conectar, la clave de seguridad y se definen las características de la clave de seguridad.

interface=wlan0 //tipo de Red a la que se va a configurar
driver=rtl871xdrv //Tipo de Driver utilizar por el demond hostapd
ssid=RPi_PA **//**Nombre de la red inalámbrica
hw_mode=g //Modo de red Wireless a 54 Mbps
channel=6 //Canal de transmisión sin generar conflictos
macaddr_acl=0 //Filtro de conexión MAC, desactivado
auth_algs=1 //Método de autenticación abierta
ignore_broadcast_ssid=0 //Permite mostrar la red en todos los dispositivos
wpa=2 //Tipo de seguridad WPA
wpa_passphrase=iset-uacm **//**Llave de seguridad a WLAN0
wpa_key_mgmt=WPA-PSK //Algoritmo de gestión de claves

wpa_pairwise=TKIP //Algoritmo de cifrado utilizado, en este caso WPA
rsn_pairwise=CCMP //Algoritmo de cifrado utilizado, WPA2

Anexo 1.5 Automatización para habilitación de WLAN0

A continuación, se le indica a Raspberry dónde debe encontrar el archivo de configuración que se creó, para ello, se modifica la línea #DAEMON_CONF del archivo hostapd.

sudo nano /etc/default/hostapd
DAEMON_CONF="/etc/hostapd/hostapd.conf"

Se ingresa al repositorio sysctl.conf, el cual se encarga de controlar el sistema, y en este caso específica a Raspberry que estará haciendo una comunicación en IPV4.

sudo nano /etc/sysctl.conf
net.ipv4.ip_forward=1 //Activación del IP forwarding para que actúe como router

Estos cambios se aplicarán cuando se reinicie el sistema, pero para hacerlo inmediatamente se puede introducir e siguiente comando en consola, el cual que tiene el mismo objetivo que lo anterior:

sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward" //Activación del script para activar el enrutamiento

Posteriormente se deben configurar las *iptables* para hacer la "traducción" entre la interfaz Ethernet (Eth0) y la interfaz Wireless (wlan0), de esta forma el filtrado acepta un redireccionamiento de paquetes desde dentro hacia fuera de nuestra red y mediante NAT permitiendo que los host naveguen con la dirección IP publica del servidor.

sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE //Activa la salida por Eth0 por DHCP y enmascara los paquetes haciéndolos NAT
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT //Determina la entrada de paquetes por eth0 y salida de los mismos por medio de wlan0
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT //

Estos comandos tendrían que ejecutarse cada vez que se encienda la Raspberry, pero para que esto pueda realizarse en automático se introduce el siguiente comando, recomendable realizarlo después de insertar los comandos anteriores para que se guarde la configuración.

sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"

Volvemos al archivo de configuración de las interfaces de red que hemos modificado con anterioridad:

sudo nano /etc/network/interfaces

Al final de todo, se añade la siguiente línea.

up iptables-restore < /etc/iptables.ipv4.nat

Anexo 1.6 Actualización del HOSTAPD

Dependiendo del modelo del adaptador WiFi, existen probabilidades de que el prototipo funcione o no a la primera, en caso contrario es posible solucionarlo actualizando la versión de **hostapd** que proporciona AdaFruit.

wget http://adafruit-download.s3.amazonaws.com/adafruit_hostapd_14128.zip
unzip adafruit_hostapd_14128.zip //descomprimir el reposiortio

Posteriormente toca intercambiar la versión antigua por la nueva.

sudo mv /usr/sbin/hostapd /usr/sbin/hostapd.ORIG //Comando encargado de mover el archivo hostapd.ORIG, el cual la ruta origen depende donde se descomprimió el archivo hostapd.
sudo mv hostapd /usr/sbin //Comando encargado de mover el archivo hostapd
sudo chmod 755 /usr/sbin/hostapd //Comando encargado de cambiar los permisos del directorio hostapd, en este caso el usuario tiene acceso completo y los otros usuarios pueden enumerar el archivo pero no pueden modificarlos

Para ver si todo funciona correctamente, se ejecuta el siguiente comando y se intenta realizar una conexión a la red WiFi desde cualquier ordenador.

sudo /usr/sbin/hostapd /etc/hostapd/hostapd.conf //iniciar hostapd

Si funciona, se crea un daemon que se ejecutará cada vez que la Raspberry Pi se encienda.

sudo service hostapd start //Ejecución de los Deamons para hostapd y dhcp
sudo service isc-dhcp-server start

Es posible conocer el estado de los daemon hostapd y dhcp con los siguientes comandos.

sudo service hostapd status
sudo service isc-dhcp-server status

Finalmente, para habilitar los daemon.

sudo update-rc.d hostapd enable

sudo update-rc.d isc-dhcp-server enable

Una vez hecho esto, ya se encuentra configurado el Access Point WiFi configurado para que proporcionar acceso a Internet a través de Raspberry conectada a una interfaz Ethernet. Por último se inician los servicios restantes.

sudo systemctl start hostapd
sudo systemctl start dnsmasq

Anexo 1.7 Cambio de versión del Kernel

En este punto, a pesar de haber terminado la configuración correctamente Raspberry no logra funcionar, debido al modelo de hardware y el tipo de software instalado con Raspbian, esto sucede debido a que la versión del kernel no es compatible con el modelo de Raspberry, lo que quiere decir que no logra tener una lectura de una tarjeta de red Wireless, por lo cual es necesario cambiar el kernel por una versión más antigua para que pueda leer la tarjeta de red inalámbrica y funcionar, se ejecuta el siguiente comando.

rpi-update 8fd111f77895450323abc5b34efde19548ffc480

Con ello se reinicia Raspberry, después de ello se puede consultar la versión del kernel.

Debido a que Raspbian tiene un Kernel actualizado, 4.14.79 en Raspberry Pi 2 no es posible detectar la tarjeta de Red que le sea instalada por lo que es necesario realizar el cambio para que sea compatible con esta tarjeta inalámbrica.

uname –r comando, consultar versión del Kernel

Realizamos la actualización del Firmware

rpi-update

En caso de que no se ejecute este comando, es necesario instalar este repositorio o en caso de que se encuentre instalado, actualizarlo.

sudo apt-get
install rpi-update

Posteriormente se realiza una actualización de los paquetes nuevamente, y se reinicia el sistema.

sudo reboot

Posteriormente se instala el módulo Kernel y se genera la lista de dependencias para su correcto funcionamiento con los comandos.

insmod /lib/modules/3.10.18+/kernel/drivers/net/wireless/8188eu.ko // la función de insmod es insertar en el SO el módulo Kernel
depmod –a
//comprueba las dependencias aplicadas a todos los archivos del kernel ya que este también depende de otros programas para funcionar correctamente.

Se reinicia la tarjeta de red y se comprueban los adaptadores de red inalámbricos instalados.

/etc/init.d/networking restart //Comando para reiniciar las tarjetas de red inalámbricas
Iwconfig //Enlista todas las interfaces Wireless destacadas

Se descargan los drivers para la tarjeta de red, en este caso no fue posible hacerlo desde la terminal, debido a que Dropbox no permite darnos acceso, por lo que no se descargó manualmente en una memoria flas, se descomprimió, se copió al directorio correspondiente y se instaló con los permisos correspondientes.

Anexo 1.8 Instalación del driver para Red inalámbrica

wget https://dl.dropboxusercontent.com/u/80256631/8188eu-20140307.tar.gz // descargar el archivo comprimido desde la url de Dropbox

tar -zxvf 8188eu-20140307.tar.gz //Comando para descomprimir en el cual **z** indica el tipo de archivo gz, **x** extraer el archivo, **v** muestra el progreso y **f** permite especificar el nombre del archivo
sudo cp rtl8188eufw.bin /lib/firmware/rtlwifi //comando para realizar una copia del driver
sudo install -p -m 644 8188eu.ko /lib/modules/3.10.33+/kernel/drivers/net/wireless //comando para instalar el kernel
sudo insmod /lib/modules/3.10.33+/kernel/drivers/net/wireless/8188eu.ko
// Insertar el módulo kernel en el SO
sudo depmod –a
//comprobar las dependencias
sudo reboot //Reiniciar

Raspberry

Después de haber ejecutado el último comando la versión que aparecerá en la consola, muestra **3.10.18+** la cual es suficiente para poder trabajar con el hardware que hemos instalado. Posteriormente es necesario instalar el driver necesario para una tarjeta de red TP-LINK WN725N, ejecutando el comando.

wget https://www.dropbox.com/s/18yhvwlfz493s22/8188eu-20131110.tar.gz
//Descargar el Driver

Una vez descargado el Driver, el cual se encuentra comprimido con una extensión .tar.gz, se descomprime y se instala utilizando los siguientes comandos.

tar -zxvf 8188eu-20131110.tar.gz
//Descomprimir el archivo
install -p -m 644 8188eu.ko /lib/modules/3.10.18+/kernel/drivers/net/wireless
//Instalar driver

Anexo 1.9 Instalación de Mosquitto

Descarga de la firma (signing key) con wget sudo

wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key

Clave para autenticar el paquete que se descargara posteriormente

apt-key add mosquitto-repo.gpg.key

Entrar a la carpeta sources.list.d

cd /etc/apt/sources.list.d/

En ella se descarga la lista de repositorios de Mosquitto, en este caso para Raspbian Strech

sudo wget http://repo.mosquitto.org/debian/mosquitto-stretch.list

Realizamos una actualización de paquetes

apt-get update

Instalamos el bróker de mosquitto

apt-get install mosquito

Instalamos el cliente en mosquito

apt-get install mosquitto-clients

Anexo 1.10 Raspberry-XBee

Configuración de radios XBee y módulos Bluetooth para comunicación con Raspberry.

La habilitación de este protocolo se hizo por medio de la programación en Python, en el cual para establecer el funcionamiento de estos radios primero se descarga la librería que permite realizar lecturas por medio de la interfaz serial por medio del siguiente comando.

apt-get install python-serial

Y con la librería instalada, ahora es posible realizar la programación en un editor de texto, para este caso se realizó por medio de **nano**, donde especificamos al programa que la lectura serial se hace por medio de la lectura de uno de los archivos TTY los cuales permiten la comunicación con dispositivos físicos y estos se encuentran en la carpeta /dev como se muestra.

/dev/ttyAMA0

El cual está interfaz puede ser consultada utilizando el comando list/dev/tty donde aparecerán todas las interfaces con las que cuenta Raspberry, en este caso ttyAMA0 corresponde a la comunicación Serial por medio de los pines GPIO, con esto ahora es posible realizar el Xbee en base al algoritmo que se mostró.

Programa para leer datos desde el puerto Serial

```
import serial

ser = serial.Serial('/dev/ttyAMA0',9600)
print'Conectando puerto serie'

while True:
incoming = ser.readline().strip()
print 'Recibo: %s' % incoming
```

Programa para leer datos desde el puerto Serial y guardarlos en archivo .txt

```
import serial

ser = serial.Serial("/dev/ttyAMA0",baudrate=9600)

while True:
        read =ser.read()
        print read
ser.close()
```

Anexo 1.11 Raspberry-Bluetooth

Para habilitar Bluetooth es de forma similar, solo que en este caso ya no la librería ya se encuentra descargada y se cambia el tipo de interfaz ya que la comunicación se realiza por USB el cual puede especificarse como USB0 o USB1 dependiendo del número de interfaces, para este caso solo contiene dos y lo definimos por USB0 con la siguiente dirección.

/dev/ttyUSB0

Programa para leer datos desde la interfaz USB y guardarlos en archivo .txt

```
import serial
import os
file = open("/home/pi/pyth/bluetooth.txt", "w")
ser = serial.Serial("/dev/ttyUSB0",baudrate=9600)
while 1:
        read = ser.read()
        print read
        file.write(read)

ser.close()
file.close()
```

Posteriormente se habilita el protocolo FTP con ayuda de websockets programado en python 3

Anexo 1.12 Raspberry-ESP8266

Programa para conectar ESP8266 con Raspberry por MQTT

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>


const char* ssid = "RPi_PA";
const char* password = "iset-uacm";
const char* mqtt_server = "192.168.100.1";

WiFiClient espClient;
PubSubClient client(espClient);

void setup (){
Serial.begin (115200);
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED){
  delay (500);
  Serial.print(".");
}
Serial.println("Dispositivo conectado");
Serial.println("direccion IP:");
Serial.println(WiFi.localIP());

client.setServer (mqtt_server, 1883);
client.setCallback(callback);
}
void callback (char* topic, byte* payload, unsigned int length){
Serial.print("mensaje cargado [");
Serial.print(topic);
Serial.print("]");

for (int i=0; i<length; i++){
Serial.print((char)payload[i]);
}
Serial.println();
}
void reconnect(){
  while (!client.connected()){

    if (client.connect("testgs")){
    Serial.println("conectado");
    client.publish ("sensor/boton/luminaria1", "Lanzando primer mensaje");
```

```
    client.subscribe("sensor/boton/luminaria1");
   }else{
    Serial.print("fallo, reconectando ...");
    Serial.print(client.state());
    Serial.println ("volviendo a conectar ");
    delay (3000);
    }
  }
}
void loop(){
  if (!client.connected()){
    reconnect();
  }
  Serial.available();
  client.loop();
  if (caracter == 'Q'){
  client.publish("sensor/boton/luminaria1", "hola mundo");
  }else{
    Serial.println("caracter incorrecto");
  }
  delay (1000);
  }
```

Anexo 1.13 Arduino-Raspberry por XBee

Programa para conectar Arduino con Raspberry por XBee

```
const int sensorPin = A0;

void setup(){
  Serial.begin(9600);
}
void loop() {
  int humedad = analogRead(sensorPin);
  Serial.println(humedad);

  if(humedad < 500)
  {
  Serial.println("Humedad Suficiente");
  }
  else{
  Serial.println("Tierra Seca");
```

```
  }
  delay(1000);
}
```

Anexo 1.14 Arduino-Raspberry RF

Para Raspberry se descarga el protocolo WiringPi

Para la versión de raspberry utilizada, en este caso la versión 2B y se ejecutan los programas que vienen en el ZIP. En la página oficial http://wiringpi.com/ se descarga la versión WiringPi-e8da87f.tar.gz

El comprimido se descargará con el nombre rpi.zip el cual se descomprime con el comando
unzip rpi.zip y se ejecutan cada uno de los programas realizados en c++ con el comando

gcc recive.c –o recive

Para cada uno cambiando recive.c por el nombre del siguiente programa y usando otro nombre

recive.cpp
rftester.cpp
RCSwitch.cpp
RCSwitch.h

Programa para conectar Arduino con Raspberry por Radiofrecuencia ASK a 433MHZ

```
#include <RCSwitch.h>
RCSwitch mySwitch = RCSwitch();

long envio = 0;

// Declaración de variables
float tempC;
int tempPin = 0; // Definimos la entrada en pin A0
void setup()
{
   // Abre puerto serial y lo configura a 9600 bps
```

```
    Serial.begin(9600);
    mySwitch.enableTransmit(10);
}
void loop()
{
    // Lee el valor desde el sensor
    tempC = analogRead(tempPin);
    Serial.println(tempC);
    mySwitch.send(tempC, 8);
    // Espera cinco segundo para repetir el loop
    delay(5000);
    mySwitch.send(tempC,8);
}
```

Anexo 1.15 Raspberry-FTP

Programas para habilitar protocolo FTP

FTP Cliente

```
#VARIABLES
host = 'localhost'
port = 8050
#se importa el módulo
import socket

#Creación de un objeto socket (cliente)
obj = socket.socket()

#conexión con el servidor. Parametros : IP
obj.conect ((host, port))
print  ("conectando al servidor")

#creación un bucle para retener la conexión
while True:
                #Instanciamos una entrada de datos para que el cliente pueda enviar
mensajes
                mens = raw_input ("Mensaje desde Cliente a Servidor >>")

                #con el metodo send, se envía el mensaje
                obj.send(mens)
```

```
#Se cierra la instancia del objeto servidor
obj.close()

#se imprime la palabra adios cuando se cierra el programa
print("Conexión cerrada")
```

FTP Servidor

```
import socket

#objeto para trabajar con el socket
der = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#Puerto y servidor que debe escuchar
ser.bind(("", 8050))

#Aceptamos conexiones entrantes con el modulo listen. por parámetro las conexiones

simultaneas
ser.listen(1)

#Instanciamos un objeto cli (socket cliente) para recibir datos
cli, addr = ser.accept()

while True:
                #recibimos el mensaje, con el metodo recv recibimos datos. por el
parámetro la cantidad de bytes para recibir
                recibido = cli.recv(1024)

        #si se reciben datos, nos puestra la IP y el mensaje recibido
        print "Recibo conexion de la IP: "+ str(addr[0]) +" Puerto:" +str(addr[1])

        #se devuelve el mensaje al cliente
        cli.send(mensaje recibido)

#cerramos la instancia del socket  cliente y servidor
cli.close()
ser.close()
print("conexiones cerradas")
```

Referencias

Academic, C. N. (Enero de 2019). *Introduction to IoT*. Obtenido de Introduction to IoT: https://static-course-assets.s3.amazonaws.com/I2IoT20/es/index.html#2.2.1.1

Azcarategui, B. (16 de Abril de 2018). *La cuarta revolución converge entre IT y OT*. Obtenido de eSemnanal: https://esemanal.mx/2018/04/la-cuarta-revolucion-converge-entre-it-y-ot/#:~:text=El%20Internet%20de%20las%20cosas,son%20fundamentales%20en%20este%20sentido.

Caminati, D. (Junio de 2016). *ZigBee XB24-ZB coordinator API*. Obtenido de ZigBee XB24-ZB coordinator API: http://fritzing.org/projects/zigbee-xb24-zb-coordinator-api-with-raspberry

decodigo.com. (s.f.). Obtenido de Obtenido de Leer un archivo de texto: http://www.decodigo.com/python-leer-un-archivo-de-texto

Dignani, J. P. (2011). *ANÁLISIS DEL PROTOCOLO ZIGBEE.* La Plata.

Eames. (27 de Abril de 2016). Obtenido de How To Free Up Some Space On Your Raspbian SD Card? Remove Wolfram & LibreOffice: https://www.recantha.co.uk/blog/?p=14633

ELECTRONICS, M. (16 de Agosto de 2016). *MADNESS ELECTRONICS*. Obtenido de Obtenido de Tutorial sensor de humedad de suelo: http://www.madnesselectronics.com/tutorial-sensor-de-humedad-de-suelo/

Evans, D. (2011). *Internet de las cosas.* Cisco Internet Business Solutions Group (IBSG).

Gonzalez, C. (15 de Enero de 2016). *raspberrypi forums*. Obtenido de raspberrypi forums: https://www.raspberrypi.org/forums/viewtopic.php?t=134938

Gordon. (2019). Obtenido de Wiring Pi: http://wiringpi.com/

Gordon. (2019). Obtenido de http://wiringpi.com/: http://wiringpi.com/

IBM. (4 de Octubre de 2017). Obtenido de Conociendo MQTT: https://www.ibm.com/developerworks/ssa/library/iot-mqtt-why-good-for-iot/index.html

J. Postel, J. R. (Octubre de 1985). Obtenido de FILE TRANSFER PROTOCOL : https://tools.ietf.org/html/rfc959

Jamie M, R., Jeremy G, F. A.-C., Andrew D, R., & Bharat V, B. (s.f.). Sensor Networks and Grid Middleware for Laboratory Monitoring. 2-4.

Josh. (14 de Noviembre de 2016). Obtenido de de Raspberry Pi Bluetooth Terminal: https://www.hackster.io/user16726/raspberry-pi-bluetooth-terminal-f7e9bd

LINARES RUIZ, R., QUIJANO VÁSQUEZ, J. A., & HOLGUÍN LONDOÑO. (2004). MPLEMENTACIÓN DEL PROTOCOLO BLUETOOTH PARA LA CONEXIÓN INALÁMBRICA DE DISPOSITIVOS ELECTRÓNICOS

PROGRAMABLES. *Scientia Et Technica, vol. X*, 31.
Schmidt, M. (2011). Arduino A Quick-Start Guide. *The Pragmatic Programmers*, 26-30.
Tárano León, S. D. (2018). *Towards smarter cities taking advantage of the Fog Computing paradigm.* La Habana, Cuba.
THEORY, G. (2018). Obtenido de Tutorial Rasbperry Pi - Cómo crear un punto de acceso WiFi: https://geekytheory.com/tutorial-rasbperry-pi-como-crear-un-punto-de-acceso-wifi
UNDERCODE. (22 de Noviembre de 2013). Obtenido de Tutorial envio de archivos usando sockets: https://underc0de.org/foro/python/tutorial-envio-de-archivos-usando-sockets/
Vela, A. (23 de Junio de 2019). *escribecodigo*. Obtenido de Socket en Python – Crear una conexión Cliente/Servidor: https://www.escribecodigo.com/sockets-en-python/
Velázquez, J. I. (2016). *Redes de datos, contexto y evolución.* Ciudad de México: Samsara.
Wallace, M. R. (2014). Getting Started with Raspberry Pi. *Make*, 4-10.
Ynzunza Cortés, C. B. (2017). El Entorno de la Industria 4.0. *Implicaciones y Perspectivas Futuras*, 3.
Yuan, M. (2017). ¿Por qué MQTT es uno de los mejores protocolos de red paraInternet de las Cosas? *developerWorks*, 1-3.
ZORRILLA, J. (31 de Mayo de 2015). Obtenido de Creando un Servidor Raspberry Pi – XBee en python y conectando un Cliente Arduino – XBee: https://www.internetdelascosas.cl/2015/05/31/creando-un-servidor-raspberry-pi-xbee-en-python-y-conectando-un-cliente-arduino-xbee/