DESARROLLO DE UN GATEWAY INTELIGENTE PARA EL INTERNET DE LAS COSAS (IoT)

Gustavo Adolfo Larrahondo Balanta

Director: **PhD. Miguel Ángel Niño Zambrano**

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo I+D en Tecnologías de la Información - GTI
Línea de Investigación en Gestión de la Información y Tecnologías Internet
Popayán, abril de 2021

DESARROLLO DE UN GATEWAY INTELIGENTE PARA EL INTERNET DE LAS COSAS (IoT)

Gustavo Adolfo Larrahondo Balanta

Director: **PhD. Miguel Ángel Niño Zambrano**

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo I+D en Tecnologías de la Información - GTI
Línea de Investigación en Gestión de la Información y Tecnologías Internet
Popayán, abril de 2021

DESARROLLO DE UN GATEWAY INTELIGENTE PARA EL INTERNET DE LAS COSAS (IoT)

Trabajo de grado presentado como requisito para obtener el título de
Ingeniero en Electrónica y Telecomunicaciones

Gustavo Adolfo Larrahondo Balanta

Director: **PhD. Miguel Ángel Niño Zambrano**

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo I+D en Tecnologías de la Información - GTI
Línea de Investigación en Gestión de la Información y Tecnologías Internet
Popayán, abril de 2021

AGRADECIMIENTOS

El autor expresa sus agradecimientos a:

A Dios, por permitirme hacer realidad el sueño de formarme como ingeniero y, por darme la sabiduría e inteligencia necesarias para sacar adelante este sueño.

A mis padres y hermanos, por su apoyo incondicional, constante motivación y acompañamiento que me brindaron en todo este proceso de formación.

A mis familiares y amigos, por sus palabras de aliento y motivación.

Al Doctor Ing. Miguel Ángel Niño Zambrano, director de este trabajo de grado, por su gran dedicación, constante motivación y orientación y por compartir sus conocimientos y experiencias.

A la Universidad del Cauca y al Departamento de Telemática, por haber sido cuna de todo mi proceso de formación como Ingeniero en Electrónica y Telecomunicaciones.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

CAPÍTULO 1 INTRODUCCIÓN

A lo largo de la historia, el hombre ha intentado encontrar las correlaciones entre los parámetros que definen el clima en cada área geográfica y el desarrollo de las plantas, con la finalidad de establecer en cada zona los cultivos más adecuados [1].

La agricultura es reconocida hoy en día como una de las actividades más destacadas a nivel social, ambiental y económico para todo ser humano brindándole la fuente principal de producción de alimentos y cultivo de la tierra. Según estadísticas de la FAO (Food and Agriculture Organization of the United Nations), la participación de América Latina y el Caribe (ALC) en el mercado mundial agroalimentario aumenta, en el futuro esta región desempeñará un papel aún mayor como oferente global de alimentos y materias primas agrícolas, para lo cual deberá mejorar en materia de reglas que obstruyen el comercio, infraestructura y regulaciones. Mientras que en 1990 la participación de ALC en las exportaciones agroalimentarias mundiales fue de 8,3 %, en 2015 alcanzó el 13,8 % [2].

Dado que, la agricultura tiene una dependencia directa muy acusada de las condiciones ambientales que determinan, en gran medida, la viabilidad y el desarrollo de las producciones agrícolas [1], se hace necesario utilizar nuevas técnicas para el desarrollo y crecimiento de la agricultura.

La agroclimatología, es la aplicación de los elementos del clima en la agricultura, ésta puede ayudar eficientemente a mejorar la agricultura de un país en numerosos campos, entre los que se puede destacar la planificación agrícola [3]. El no considerar elementos del clima en la planificación agrícola puede generar efectos negativos como la reducción drástica en el rendimiento de las cosechas, por lo cual se debe adoptar nuevas prácticas o mecanismos que contrarresten estos impactos climatológicos [3]. Lo anterior motiva a la investigación y el desarrollo de sistemas que permitan la captura, procesamiento y envío de datos climatológicos como temperatura, humedad relativa, radiación solar, velocidad y dirección del viento. Estas funciones son realizadas por dispositivos sensores capaces de medir cada una de las variables y varían dependiendo del tipo de cultivo, el cual determina qué datos del clima es necesario registrar. Los datos obtenidos se utilizan tanto para la elaboración de predicciones meteorológicas a partir de modelos numéricos como para estudios climáticos.

La interconexión de estos dispositivos sensores supone un reto en el desarrollo de nuevas soluciones que permitan su interoperabilidad, dada las condiciones de heterogeneidad que muchos presentan al implementar diferentes tecnologías, protocolos, conexiones, ser producidos por diferentes fabricantes y poseer características variadas, hace que su integración con soluciones y servicios de mediciones climatológicas para el monitoreo de los cultivos se dificulte. Con lo anterior, se evidencia la importancia de contar con un dispositivo que sirva como una puerta de enlace (gateway) con las aplicaciones y servicios de internet, y sea capaz de integrar esta variedad de sensores a nivel de conexiones eléctricas, de protocolos de comunicación y a nivel de datos que permitan la creación de nuevos servicios útiles para los usuarios en sus diferentes contextos de aplicación. Este dispositivo proveerá a los usuarios funciones completas de monitorización, registro, procesamiento y gestión de las soluciones implementadas en campo y los dispositivos asociadas a ellas, incluyendo también, funciones de gestión y visualización remotas, gracias a su conectividad con internet.

1.1. MOTIVACIÓN DEL TRABAJO DE GRADO

En la actualidad, se cuenta con gran variedad de dispositivos sensores y estaciones climatológicas[1] de diferentes tecnologías, marcas y fabricantes, lo cual, genera la dificultad de que no todos los sensores y sus datos sean accesibles, integrables e interoperables, dificultando la creación de aplicaciones inteligentes que generen información relevante para los usuarios (agricultores) y entidades para quienes es de vital importancia contar con información del clima. Debido a esto, se hace necesario contar con un dispositivo que proporcione una puerta de enlace inteligente[2] (Gateway Inteligente) [4], capaz de permitir la conexión a los sensores y/o estaciones, capturar sus datos, ejecutar aplicaciones que pueden recolectar información local y entregarla confiablemente a la nube de manera estándar.

La diversidad de puntos finales que una puerta de enlace debe soportar plantea preocupaciones de diseño, así, la conexión directa de un dispositivo simple tal como un sensor de humedad a Internet puede ser compleja y costosa, especialmente si el dispositivo no tiene su propio procesador.

Aunque existe en la literatura trabajos sobre el desarrollo de Gateways Inteligentes [4]–[8], se ha encontrado que existen problemas de interoperabilidad semántica entre los recursos IoT gestionados por el Gateway y otros dispositivos inteligentes que estén en la red. El grupo de investigación GTI de la Universidad del Cauca ha desarrollado trabajos al respecto, proponiendo una arquitectura de interoperabilidad semántica en IoT (Internet of Things – Internet de las Cosas) [9], para la cual se desarrolla en este trabajo un Gateway para su implementación y despliegue, que permita su aplicación en contextos como el de la agricultura, elemento primordial de necesidad social en la región. Adicionalmente, el Gateway Inteligente desarrollado podrá ser implementado en otros contextos como el de la domótica, ciudades inteligentes, aplicaciones inteligentes para la salud, entre otras.

1.2. OBJETIVOS

1.2.1. Objetivo general

Desarrollar un dispositivo Gateway Inteligente para el Internet de las Cosas, basado en la arquitectura y concepto de Objeto Inteligente desarrollado por Niño-Zambrano [9], con el fin de aplicarlo a la monitorización de variables agroclimáticas.

1.2.2. Objetivos específicos

1. Diseñar un Gateway Inteligente, que permita incorporar recursos IoT y los protocolos Ethernet, Wifi y ZigBee, con el fin de soportar su conectividad y reutilización en escenarios de interacción de objetos inteligentes [9].

[1] Una estación climatológica es una instalación que permite monitorear las diferentes variables del clima, para ello, cuenta con diferentes dispositivos sensores climáticos capaces de registrar los valores de diferentes variables como temperatura, humedad, luz, velocidad y dirección del viento, precipitación, entre otras.

[2] Una Puerta de Enlace Inteligente (Gateway Inteligente) es un dispositivo que actúa como interfaz de conexión entre aparatos o dispositivos con una red, permitiendo la conversión de datos entre los protocolos de comunicación de corta distancia a la red de comunicación tradicional.

2. Implementar un Gateway Inteligente tomando como base el diseño propuesto, utilizando tecnologías IoT.

3. Evaluar la funcionalidad del Gateway Inteligente mediante un estudio de caso en el monitoreo de las siguientes variables agroclimatológicas: temperatura, humedad relativa, precipitación (cantidad de lluvia), presión atmosférica/altitud, luminosidad, velocidad y dirección del viento; variables que constituyen los principales factores ambientales del clima en la producción agrícola [10], [11].

1.3. ESTRUCTURA DEL TRABAJO DE GRADO

El presente trabajo de grado se ha dividido en capítulos con la siguiente estructura:

- **CAPÍTULO 1. INTRODUCCIÓN**: En este primer capítulo se hace la descripción del problema y el contexto de aplicación de la solución a desarrollar, así como también, el objetivo general y objetivos específicos del presente proyecto.
- **CAPÍTULO 2. MARCO TEÓRICO**: En este segundo capítulo se presentan las bases conceptuales que son importantes y necesarias para el desarrollo del presente trabajo de grado, las cuales constituyen los núcleos temáticos definidos en la investigación, como son: Internet de las Cosas, Objetos Inteligentes, Gateway IoT, Agroclimatología y su relación con los estudios y Sistemas Meteorológicos.
- **CAPÍTULO 3. DISEÑO GATEWAY INTELIGENTE IOT**: En este tercer capítulo se presenta la construcción de la arquitectura propuesta para el desarrollo del Gateway Inteligente IoT.
- **CAPÍTULO 4. IMPLEMENTACIÓN GATEWAY INTELIGENTE IOT**: En este cuarto capítulo, se establecen las características y requisitos que debe tener el Gateway Inteligente IoT y se realiza la selección de los recursos y tecnologías utilizadas para su implementación.
- **CAPÍTULO 5. EVALUACIÓN GATEWAY INTELIGENTE IOT**: En este quinto capítulo se realiza el proceso de evaluación funcional y se realizan pruebas en el contexto de aplicación propuesto.
- **CAPÍTULO 6. CONCLUSIONES Y TRABAJOS FUTUROS**: En este último capítulo se describen las conclusiones y trabajos futuros del proyecto desarrollado, definiendo el alcance logrado de la solución propuesta y proponiendo la implementación futura de nuevos servicios y aplicaciones para el Gateway Inteligente IoT.

Además de los capítulos mencionados, se proporciona una serie de Anexos que detallan el proceso de implementación técnica y despliegue del Gateway Inteligente IoT:

- Anexo A: Instalación de Eclipse Kura.
- Anexo B: Configuración de módulos XBee desde XCTU.
- Anexo C: Configuración de Objetos Inteligentes.
- Anexo D: Configuración de AWS IoT Core.
- Anexo E: Código fuente descubrimiento de Objetos Inteligentes.
- Anexo F: instalación de RPi-Monitor en la Raspberry Pi

CAPÍTULO 2 MARCO TEÓRICO

2.1. INTERNET DE LAS COSAS

El Internet de las Cosas (en inglés, *Internet of Things*, abreviado *IoT*) es una tecnología utilizada para proporcionar servicios inteligentes a los usuarios mediante la conexión de varios dispositivos conectados a internet y permitir que estos dispositivos intercambien información entre sí [12].

Un pronóstico recientemente realizado por la Corporación Internacional de Datos (IDC) proyecta al Internet de las Cosas (IoT) y al ecosistema asociado a ser un mercado de $7,1 billones en el 2020 [13], que incluirá 50.000 millones de dispositivos conectados [14].

Además de Internet de las Cosas, existe una amplia variedad de términos para denominar este mismo concepto: Internet Físico, Internet Ubicuo, Internet de Todas las Cosas (usado por Cisco), Web de las Cosas, Internet Industrial (usado por General Electric), Internet del Futuro o Computación Física entre otros. A menudo, el IoT es comparado con otro término, el denominado Máquina a Máquina (Machine to Machine - M2M), pero son tecnologías distintas y cada una tiene sus propias funcionalidades. La Figura 1 muestra los principales elementos de la tecnología IoT.

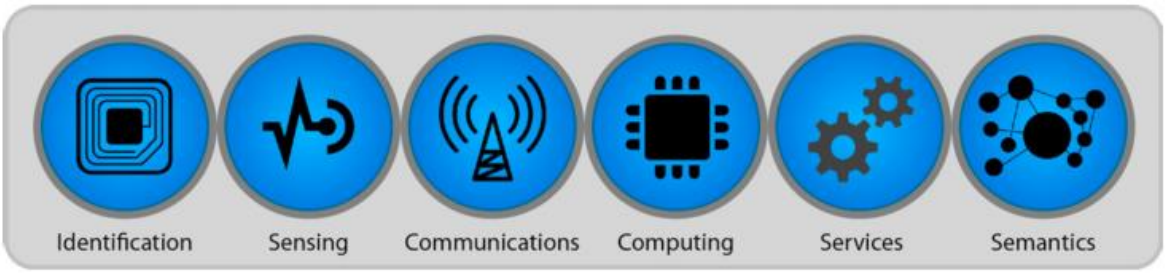

Figura 1. Elementos IoT.

Fuente: A. Glória, F. Cercas, and N. Souto, "Design and implementation of an IoT gateway to create smart environments," Procedia Comput. Sci., vol. 109, pp. 568–575, Jan. 2017.

Las soluciones basadas en IoT tienen la capacidad para hacer más que solo conectar un dispositivo a internet, en ellas, se puede mejorar notablemente la eficiencia de los dispositivos conectados y agregar nuevas características, que permiten crear nuevos servicios inteligentes para los usuarios.

2.1.1. Machine to Machine - M2M

Es un término utilizado para describir las tecnologías que permiten a los computadores, procesadores embebidos, sensores inteligentes, actuadores y dispositivos móviles comunicarse unos a otros, hacer mediciones, transferir información y tomar decisiones, incluso sin intervención humana [15]. Pero, Internet de las Cosas es mucho más amplio, es interactuar con los objetos que nos rodean, incluyendo objetos estáticos no inteligentes, y aumentar dicha interacción con algunos contextos, por ejemplo, la geolocalización.

2.1.2. Antecedentes IoT

En 1990 Jhon Romkey y Simon Hacket desarrollaron el primer objeto con conexión a internet conocido: una tostadora. Dicha tostadora se podía encender y apagar desde internet. A pesar de que Jhon Romkey y Simon Hacket crearon el primer objeto conectado a internet, no fue hasta 1999 cuando el ingeniero Bill Joy sentara las bases de Internet de las Cosas al exponer los beneficios de tener dispositivos conectados a internet. Diez años después, Kevin Ashton acuñó el término Internet de las Cosas con su artículo en RFID Journal el 12 de julio de 2009. En este artículo se exponía la idea de conectar todos los objetos que nos rodean a internet con el objetivo de poder obtener mayor información de ellos en cualquier momento [16].

Según el IBSG, Internet Business Solutions Group de Cisco, en torno a los años 2008-2009 nace Internet de las Cosas debido a que el número de dispositivos electrónicos superó al número total de habitantes en la tierra. Según reportes, para el año 2020 y a partir de esta fecha 50 mil millones de dispositivos estarán conectados a internet [14], [17]. La Figura 2 muestra el crecimiento de IoT en el mundo para el año 2020.

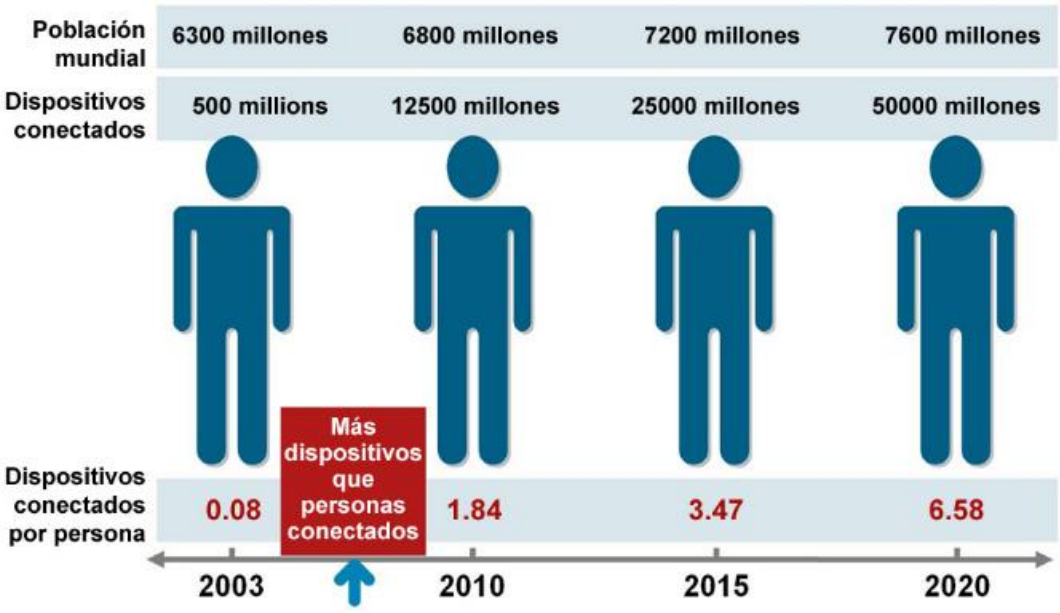

Figura 2. Crecimiento del IoT en el mundo.

Fuente: D. Evans, The Internet of Things: How the Next Evolution of the Internet is Changing Everything, vol. 1. 2011.

2.1.3. Pila de Tecnologías y Plataformas para IoT

Desde la perspectiva tecnológica, la implementación de IoT requiere de la combinación de múltiples componentes hardware y software, los cuales se ilustran en la Figura 3 y corresponden a la pila de tecnologías y plataformas necesarias para la creación de una solución de IoT. Esta pila de tecnología de IoT, generalmente se consta de tres capas centrales: Capa de Dispositivo, Capa de Conectividad, Capa IoT Cloud o Plataforma de IoT [18].

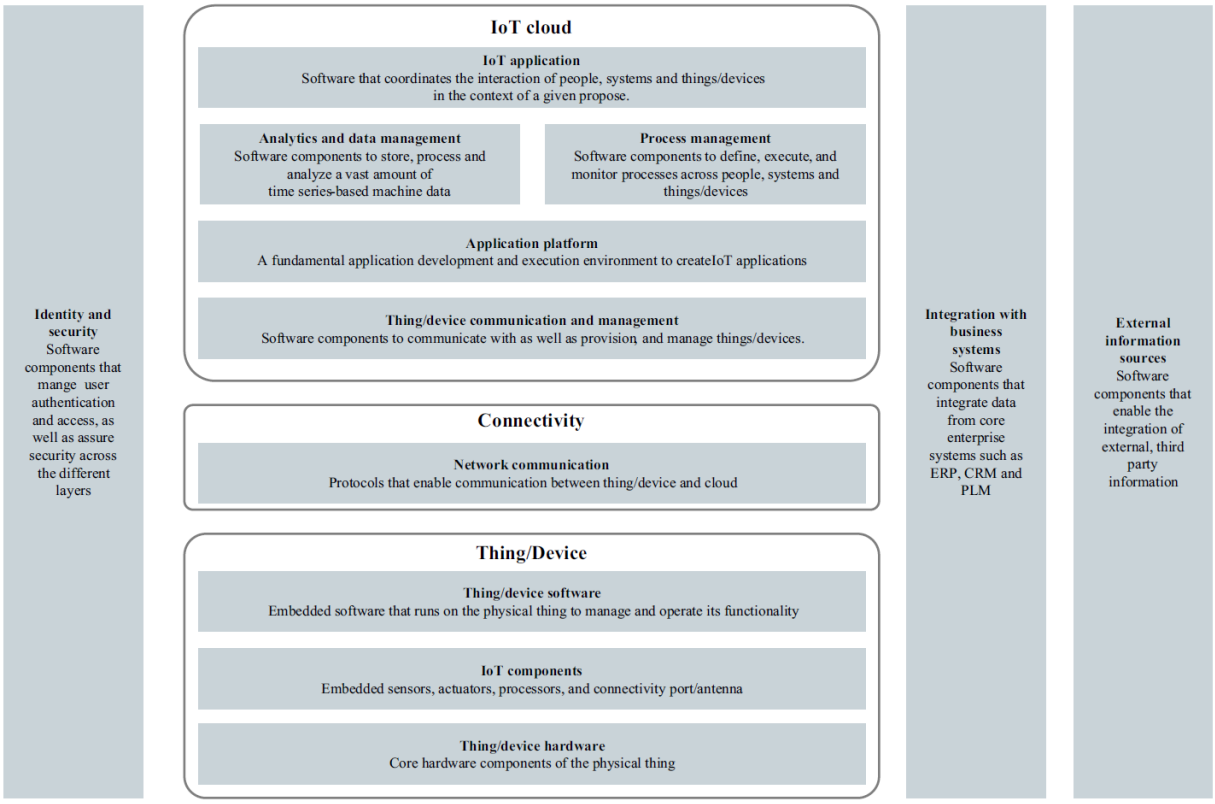

Figura 3. Pila de Tecnologías y Plataformas para IoT.

F. Wortmann and K. Flüchter, "Internet of Things: Technology and Value Added," Business and Information Systems Engineering, vol. 57, no. 3. 2015.

2.1.3.1. Capa de Dispositivo

Comprende todas las conexiones eléctricas y la interacción de los dispositivos hardware a nivel físico. En esta capa se encuentran los dispositivos como sensores, actuadores, controladores, procesadores, puertos de conexión a nivel de red y datos, entre otros. Así mismo, la capa de dispositivo, contiene también el software embebido o firmware que corre en cada dispositivo y permite su funcionamiento, interacción, análisis, gestión de datos y el monitoreo de procesos por parte de las personas.

2.1.3.2. Capa de Conectividad

La capa de conectividad, contiene todos los protocolos de comunicación que hacen posible la comunicación entre los dispositivos físicos y la red de internet.

2.1.3.3. Capa IoT Cloud o Plataforma IoT

La Capa IoT Cloud o Plataforma IoT, contiene un grupo de tecnologías que se utilizan como base sobre la que otras aplicaciones se ejecutan. Las plataformas IoT son esencialmente servicios de software que ofrecen un conjunto de funcionalidades independientes de las aplicaciones que corren en los dispositivos conectados y que pueden utilizarse para crear nuevos servicios desplegados en la nube. El análisis y gestión de datos en línea, la gestión y monitorización de variables, sistemas y procesos, y la coordinación de la interacción entre

personas con los dispositivos y sistemas, son otras de las funciones que se realizan en esta capa [18].

2.1.4. Protocolos de Comunicación para IoT

*"Un **protocolo** define el formato y el orden de los mensajes intercambiados entre dos o más entidades que se comunican, así como las acciones tomadas al producirse la transmisión y/o recepción de un mensaje u otro suceso."* [19, p. 8].

Internet y en general las diferentes redes de computadores y dispositivos, hacen un uso extensivo de protocolos para llevar a cabo las distintas tareas de comunicación. Cada protocolo de la arquitectura del sistema de IoT permite la comunicación de dispositivo a dispositivo, de dispositivo a puerta de enlace, de puerta de enlace a datos o de puerta de enlace a la nube, así como la comunicación entre centros de datos [20].

Importantes organizaciones de la industria de las redes y las telecomunicaciones, como el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE - Institute of Electrical and Electronics Engineers)[3], la Unión Internacional de Telecomunicaciones (ITU - International Telecommunication Union)[4] y el Grupo de Trabajo de Ingeniería de Internet (IETF - Internet Engineering Task Force)[5], ofrecen diferentes estándares para habilitar tecnologías que se adapten al rápido crecimiento del Internet de las Cosas. Dichas entidades son referenciadas en este trabajo en los diferentes protocolos que se definen, de acuerdo a su incidencia y participación en el desarrollo de cada uno de ellos.

Para definir los protocolos de IoT, se tomó como referencia la pila de protocolos de internet de 5 capas que deriva del Modelo de Interconexión de Sistemas Abiertos (Open Systems Interconnection - OSI) [19, p. 42], [20], [21, p. 41], [22].

La Figura 4 muestra la pila de protocolos de internet de 5 capas y las 7 capas del Modelo OSI.

[3] El IEEE (The Institute of Electrical and Electronics Engineers, Inc.) creado en Nueva York en 1884, es una asociación internacional sin ánimo de lucro con sede principal en la ciudad de Piscataway en los Estados Unidos y subsedes en más de 190 países del mundo, con alrededor de 370.000 miembros, entre profesionales y estudiantes de ingeniería, diseño, derecho, administración, medicina, biología, diseño y ciencias afines. Su objetivo principal es la permanente actualización profesional en el campo de las ciencias electromagnéticas, de la electrotecnología y de la informática [87].

[4] La ITU es el organismo especializado de las Naciones Unidas para las tecnologías de la información y la comunicación – TIC. Fundada en 1865 para facilitar la conectividad internacional de las redes de comunicaciones, atribuye en el plano mundial el espectro de frecuencias radioeléctricas y las órbitas de satélite, elabora las normas técnicas que garantizan la interconexión armoniosa de redes y tecnologías [88].

[5] El IETF es el principal organismo para el desarrollo de estándares abiertos de Internet. es una organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, enrutamiento y seguridad. Se creó en los Estados Unidos en 1986. Es mundialmente conocido porque se trata de la entidad que regula las propuestas y los estándares de Internet [89].

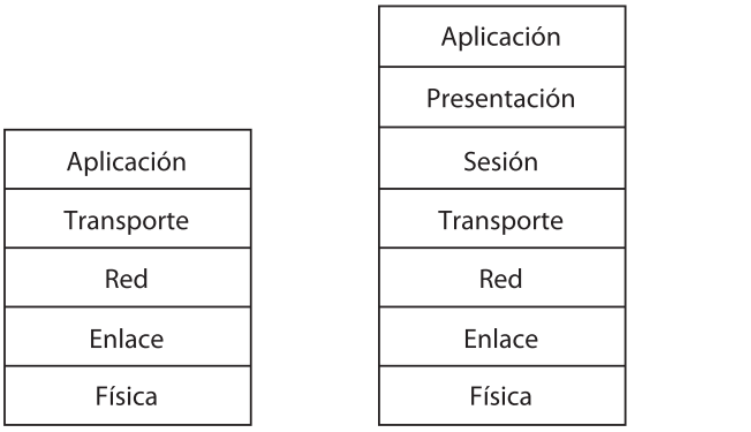

Figura 4. Pila de protocolos de Internet (a) y modelo de referencia OSI (b).

Fuente: F. James, R. Kurose, and W. Keith, Redes de computadoras : un enfoque descendente. Pearson Educación, 2010.

Complementariamente, se ilustra en la Figura 5 la pila de protocolos más comunes para IoT de acuerdo al modelo de protocolos de 5 capas de Internet.

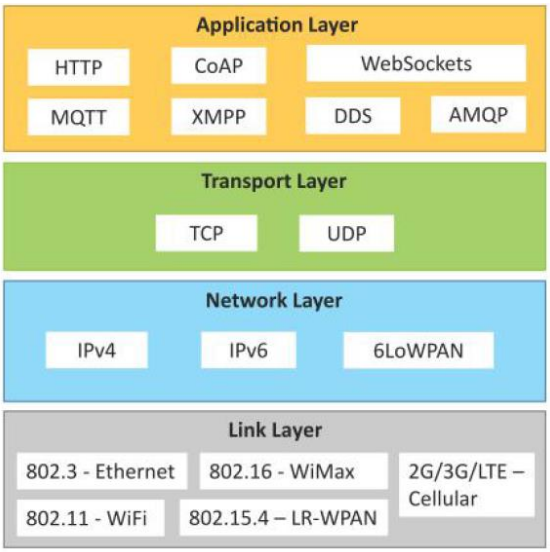

Figura 5. Pila de protocolos más comunes para IoT.

Fuente: A. Bahga and V. Madisetti, Internet of Things: A Hands-On Approach: 2014.

A continuación se definen detalladamente los protocolos más comunes para IoT, enumerado cada una de las capas del modelo de pila de protocolos de Internet.

2.1.4.1. Capa Física

El objetivo de la familia de protocolos contenidos en la capa física es definir las interfaces eléctricas y demás interfaces mediante las cuales se envían los bits como señal a través de los canales. La capa física es la base sobre la cual se construye la red. Las propiedades de

distintos tipos de canales físicos determinan el desempeño (por ejemplo, rendimiento, latencia y tasa de error). Los protocolos de esta capa dependen de los protocolos de la siguiente capa, la capa de enlace, y del medio físico por el que se transmiten los bits individuales.

Dada la estrecha relación entre la capa física y la capa de enlace, se definirán concretamente los protocolos en la siguiente sesión.

2.1.4.2. Capa de Enlace

La función de la capa de enlace de datos es proveer servicios a la capa de red. El servicio principal es la transferencia de datos de la capa de red en el dispositivo de origen, a la capa de red en el dispositivo de destino. En la capa de red del dispositivo de origen está una entidad, llamada proceso, que entrega algunos bits a la capa de enlace de datos para que los transmita al destino. La tarea de la capa de enlace de datos es transmitir los bits al dispositivo de destino, de modo que se puedan entregar a la capa de red de ese dispositivo.

Los protocolos de capa de enlace más relevantes en el contexto de soluciones IoT son:

- 802.3 - Ethernet: IEEE 802.3 es una colección de estándares para redes cableadas Ethernet. Por ejemplo 802.3 es el estándar para conexiones mediante cable coaxial 10BASE5, y también existen 802.3i y 802.3j, que son para par trenzado y para fibra óptica; 10BASE-T y 10BASE-F respectivamente.

- IEEE 802.11 (Wi-Fi) & 802.11 Low Power: Los estándares IEEE 802.11 (Wi-Fi) utilizan las bandas de 2,4 GHz, 5 GHz y 60 GHz para tener una tasa de transferencia del orden de decenas o cientos de Mbps. Proporcionan enlaces relativamente estables a cambio de un consumo elevado. Dentro de las diferentes revisiones del estándar encontramos la 802.11ª, 802.11b, 802.11g, etc. Cada una tiene unas características que las definen, como pueden ser la capacidad de transmisión y la banda de frecuencia (GHz) en la que operan.

  En el 2016 la Wi-Fi Alliance[6] anunció la salida de Wi-Fi HaLow para productos que incorporan tecnología IEEE 802.11ah. HaLow opera en bandas de frecuencias inferiores a 1 GHz, permitiendo una gama más amplia de conectividad de menor potencia que los productos Wi-Fi Certified. De esta manera HaLoW permitirá una gran variedad de casos de uso eficiente de energía, como pueden ser el hogar inteligente, agricultura de precisión y entornos de Smart City entre otros. HaLow amplía la tecnología Wi-Fi en la banda de los 900 MHz, con conectividad de baja potencia para aplicaciones y sensores portátiles. El alcance es casi el doble que el de la Wi-Fi convencional, y no sólo será capaz de transmitir más señales, sino que también permitirá conexiones más confiables en entornos hostiles donde la capacidad de penetrar en paredes y otras barreras es importante. HaLow adoptará los protocolos Wi-Fi existentes y ofrecerá numerosas ventajas para los diferentes desarrollos en IoT [23].

[6] Wi-Fi Alliance es la red mundial de empresas que impulsa la adopción y evolución global de una de las tecnologías de comunicación más valiosas del mundo. A través del liderazgo intelectual, la defensa del espectro y la colaboración en toda la industria, la Wi-Fi Alliance desarrolla tecnologías innovadoras, requisitos y programas de prueba que ayudan a garantizar que Wi-Fi brinde a los usuarios la interoperabilidad, seguridad y confiabilidad que esperan [90].

- 802.16 - WiMax: IEEE 802.16 es una colección de protocolos para conexiones de banda ancha inalámbrica. Un ejemplo de protocolo de este conjunto es el protocolo 802.16m, que permite anchos de banda de hasta 1 Gbit/s.

- 802.15.4 - LR-WPAN: 802.15.4 es un estándar que define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal con baja tasa de transmisión de datos (Low-Rate Wireless Personal Area Network, LR-WPAN) y bajo consumo energético. Sirve de base para otros protocolos de más alto nivel como Zigbee, 6lowpan y MiWi.

- Bluetooth 4.0 (Low Energy, LE): Al igual que el conjunto de protocolos anterior, Bluetooth 4.0 es una tecnología para crear redes de área personal WPAN (Wireless Personal Área Network). En sí, Bluetooth no es mucho más que un protocolo de capa de enlace, es una tecnología que provee servicios streaming de audio, transferencia de datos, redes de dispositivos y servicios de ubicación [24].

- 2G/3G/4G - Comunicación móvil: 2G, 3G y 4G hacen referencia a diferentes generaciones de estándares de comunicación mediante las redes de telefonía móvil celular convencionales. 2G hace uso de GSM y CDMA, 3G hace uso de UMTS y CDMA2000, y 4G hace uso de LTE. Todos ellos permiten que dispositivos IoT se comuniquen a través de las mismas redes que los teléfonos móviles, lo cual supone una ventaja fundamental en muchos casos donde no se tiene acceso a redes locales o con infraestructura que brinde mejores capacidades de conexión.

- LoRa: LoRa, que significa "Long Range", es un protocolo de redes de área amplia de baja potencia (LPWAN – Low Power Wide Area Network) desarrollado por la compañía Semtech. Los dispositivos LoRa y la tecnología de radiofrecuencia inalámbrica de Semtech se han convertido en la tecnología de facto para las redes de Internet de las cosas (IoT) en todo el mundo, gracias a una plataforma que permite comunicaciones de largo alcance y bajo consumo en dispositivos donde la autonomía de energía es de suma importancia.

  Los dispositivos LoRa y el protocolo abierto LoRaWAN, este último promovido por la "LoRa Alliance"[7], permiten aplicaciones inteligentes de IoT que resuelven algunos de los mayores desafíos que enfrenta nuestro planeta: gestión de la energía, reducción de recursos naturales, control de la contaminación, eficiencia de la infraestructura, prevención de desastres y más. Los dispositivos LoRa de Semtech y el protocolo LoRaWAN han acumulado varios cientos de casos de uso conocidos para ciudades inteligentes, hogares y edificios inteligentes, agricultura inteligente, medición inteligente, cadena de suministro inteligente y logística, y más. Con más de 167 millones de dispositivos conectados a redes en 99 países y creciendo, los dispositivos LoRa están creando un planeta más inteligente [25].

[7] LoRa Alliance es una alianza de compañías tecnológicas que cuenta con más de 500 miembros que han estado operativos desde finales de marzo de 2015. Entre los miembros se incluyen líderes tecnológicos como IBM, Cisco, HP, Foxconn, Semtech y Sagemcom, así como empresas de productos líderes como Schneider, Bosch, Diehl, y Mueller y muchas pymes y empresas emergentes. El objetivo principal de LoRa Alliance es estandarizar LPWAN (Low Power Wide Area Network) y, mediante la estandarización, permitir implementaciones de IoT a gran escala [25].

- Sigfox: Sigfox es la empresa que está detrás de esta tecnología con el mismo nombre, la cual, ofrece un servicio de comunicaciones con una red mundial Sigfox que ellos mismos han desplegado.

  Sigfox ofrece una solución de comunicaciones basada en software, donde toda la complejidad informática y de la red se gestiona en la nube, en lugar de en los dispositivos. Todo eso junto, reduce drásticamente el consumo de energía y los costos de los dispositivos conectados. Este servicio se basa en los siguientes principios [26]:

  - La provisión de un canal mínimo para la transferencia de pequeños mensajes, que complementaría otros protocolos de comunicación (WiFi, Bluetooth, Satélite, 3G, 4G, 5G, ADSL, Fibra…).

  - Configurar un canal de respaldo en caso de que fallen los principales enlaces de comunicación o en los casos en que una red se caiga después de un desastre natural o acto malicioso.

  - Incrementar y garantizar la seguridad de las redes y los intercambios, para estabilizar nuestra economía digitalizada.

  - Simplificar el acceso a diferentes redes para aumentar la adopción.

  - Reducir el consumo energético vinculado a las redes de telecomunicaciones.

  Pertenecen a esta capa otros de protocolos y tecnologías existentes que brindan soluciones a los retos de conectividad y transferencia de datos en los dispositivos para IoT, tal es el caso de Ocean [27], KNX [28], ZWave [29], Narrow-Band IoT [30], entre otras.

2.1.4.3. Capa de Red/Internet

La capa de red tiene como objetivo principal llevar los paquetes de datos desde una red de origen hasta otra red de destino, por medio del envío de datagramas IP. Esta capa proporciona direccionamiento y enrutamiento de paquetes.

Para lograr sus objetivos, la capa de red debe conocer la topología de la red (es decir, el conjunto de todos los enrutadores y enlaces) y elegir las rutas apropiadas incluso para redes más grandes. También debe tener cuidado al escoger las rutas para no sobrecargar algunas de las líneas de comunicación y los enrutadores, y dejar inactivos a otros. Por último, cuando el origen y el destino están en redes diferentes, ocurren nuevos problemas. La capa de red es la encargada de solucionarlos. Los protocolos más importantes de esta capa son:

1. IPv4 (Internet Protocol): Es un protocolo de interconexión de redes basado en internet. IPv4 es la cuarta versión del Protocolo de Internet que fue adaptado y ahora se utiliza ampliamente en la comunicación de datos a través de diferentes tipos de redes, utilizando direcciones de 32 bits para identificar a los dispositivos que se conectan a la red, por medio de un esquema jerárquico. La principal tarea de IPv4 es transferir los bloques de datos desde el host de envío al host de destino, donde los remitentes y los receptores son dispositivos con capacidades de conectividad que se identifican de forma única por las direcciones de protocolo de Internet. La dirección IP se utiliza como

identificador único para los dispositivos informáticos que están conectados a una red local o a Internet, para posteriormente direccionar y transmitir los datos a través de la red [21].

2. IPv6: El IPv6 es una actualización al protocolo IPv4, diseñado para resolver el problema de agotamiento de direcciones IP. La principal diferencia entre ambos es que IPv6 utiliza direcciones de 128 bits, por lo que es mayor el número de dispositivos que se pueden conectar a una red IP, frente a los 32 bits de IPv4.

3. 6LoWPAN: Las siglas de este protocolo significan “IPv6 over Low power Wireless Personal Area Network”. Este protocolo proporciona IPv6 a redes de dispositivos de bajos recursos que se conectan mediante el estándar IEEE 802.15.4.

4. Zigbee: Zigbee es un estándar que define un conjunto de protocolos para redes inalámbricas de corta distancia y baja velocidad de datos. Opera en las bandas de 868 MHz, 915 MHz y 2.4 GHz y puede transferir datos hasta 250Kbps. El estándar fue creado por la Zigbee Alliance[8], una organización sin fines de lucro, de más de 200 grandes empresas[9] (destacan Samsung Electronics, Texas Instruments, Honeywell, ARM, Microchip, Philips, Qualcomm, Siemens, Johnson Controls, Cypress, entre otras), muchas de ellas fabricantes de semiconductores.

   Zigbee desarrolla un protocolo que adopta el estándar IEEE 802.15.4 para sus 2 primeras capas, es decir la capa física y la subcapa de acceso al medio (MAC) y agrega la capa de red y de aplicación [31, p. 266]. En la Figura 6 se muestran las capas del protocolo ZigBee.

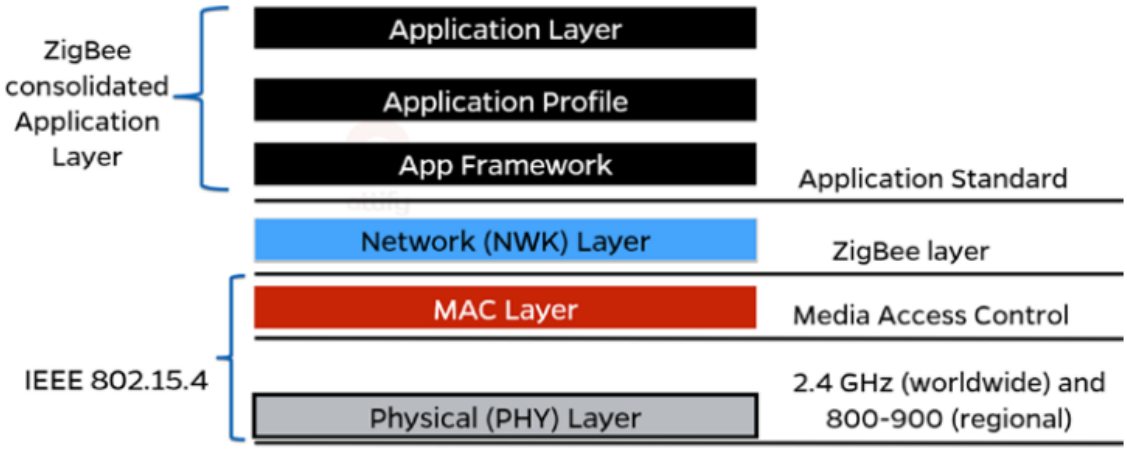

Figura 6. Pila de protocolos (ZigBee Stack).

Fuente: A. Gupta, The IoT Hacker’s Handbook. Berkeley, CA: Apress, 2019.

[8] https://zigbeealliance.org/, consultado 08/09(2020. Zigbee Alliance.

[9] https://zigbeealliance.org/members/, consultado 08/09/2020. Miembros de Zigbee Alliance, participantes y promotores.

2.1.4.4. Capa de Transporte

La capa de transporte se basa en la capa de red para proveer transporte de datos de un proceso en un dispositivo de origen a un proceso en un dispositivo de destino, con un nivel deseado de confiabilidad que es independiente de las redes físicas que se utilizan en la actualidad. Principalmente existen dos protocolos a tener en cuenta en esta capa:

1. TCP (Protocolo de Control de la Transmisión, del inglés Transmission Control Protocol): Es el protocolo más utilizado. es un protocolo confiable orientado a la conexión que permite que un flujo de bytes originado en un dispositivo se entregue sin errores a cualquier otro dispositivo en la red local. Sobre TCP se construyen protocolos de la capa de aplicación, como pueden ser HTTP, SMTP y SSH. Es un protocolo orientado a conexión y con estado, el cual asegura la entrega de mensajes, tiene control de flujo y congestión y garantiza la entrega de paquetes en orden [21].

2. UDP (Protocolo de Datagrama de Usuario, del inglés User Datagram Protocol): En contraste con TCP, UDP es un protocolo sin conexión, no confiable para aplicaciones que no desean la asignación de secuencia o el control de flujo de TCP y prefieren proveerlos por su cuenta. También se utiliza mucho en las consultas de petición-respuesta de una sola ocasión del tipo cliente-servidor, y en las aplicaciones en las que es más importante una entrega oportuna que una entrega precisa, como en la transmisión de voz o video [21].

2.1.4.5. Capa de Aplicación

Los protocolos de esta capa definen la interfaz mediante la cual los programas van a enviarse datos mediante la red. Es decir, proporciona comunicación programa a programa. El direccionamiento en programas dentro de un dispositivo se realiza a través de los puertos del mismo. Ejemplos de protocolos importantes en Internet de las Cosas son:

1. HTTP/S (Protocolo de Transferencia de Hipertexto, del inglés HyperText Transfer Protocol): Es un protocolo de comunicación del nivel de aplicación, que permite las transferencias de información en la Web. Desarrollado por el W3C (World Wide Web Consortium) y el IETF. Sigue el paradigma cliente-servidor, el cliente realiza peticiones a un servidor que sirve las peticiones a los clientes.

   Las peticiones se realizan mediante mensajes HTTP, estos se envían sobre texto plano lo que hace que los mensajes sean largos, característica que junto a otras en las que se basa su diseño, hacen que su uso no sea óptimo para redes de comunicación de bajo consumo. El protocolo define los comandos GET, PUT, POST, DELETE, OPTIONS, etc.; que el cliente envía para recibir una respuesta. El protocolo HTTP hace uso de URI's para identificar recursos. HTTP hace uso de TCP para transmitir sus mensajes [23].

2. CoAP: El uso de servicios web en Internet se ha generalizado en la mayoría de las aplicaciones, y estos servicios web dependen de REST (Representational State Transfer). CoAP es la sigla que corresponde a "Constrained Application Protocol". Este protocolo está orientado a dar soporte a aplicaciones M2M, en redes de dispositivos de recursos limitados.

Una de las principales metas de CoAP es diseñar un protocolo web genérico para los requisitos especiales de entornos restringidos, considerando especialmente las necesidades energéticas, automatización de edificios y otras aplicaciones M2M. El objetivo de CoAP no es comprimir ciegamente HTTP, sino realizar un subconjunto de REST (REpresentational State Transfer).

CoAP define un protocolo de transferencia web que se basa en la representación del estado de la transferencia encima de las funcionalidades HTTP.
REST representa una forma simple de intercambiar datos entre clientes y servidores sobre HTTP, puede verse como un protocolo de conexión que depende de la arquitectura cliente-servidor. Se usa en aplicaciones móviles y de redes sociales y elimina la ambigüedad utilizando los métodos HTTP, get, post y delete. REST permite a los clientes y servidores publicar y consumir servicios web como SOAP (Simple Object Access Protocol) de forma simple, utilizando Unifor Resource Identifiers (URIs) como nombres y los métodos HTTP get, post, put y delete como verbos.

- GET: Por medio del método GET se solicita la obtención de un recurso concreto. Una GET obtiene los datos solicitados sin afectar a los datos.

- PUT: Solicita guardar los datos transmitidos en un recurso concreto. Si el recurso ya existe, se modifica, en caso contrario se crea.

- POST: El método POST solicita almacenar los datos transmitidos y no se especifica en qué recurso se deben almacenar.

- DELETE: Este método elimina el recurso especificado.

A diferencia de HTTP, CoAP utiliza UDP para el intercambio de mensajes asíncronamente, que lo hace más adecuado para las aplicaciones IoT. CoAP modifica algunas funcionalidades HTTP para cumplir con las necesidades IoT (bajo consumo y funcionamiento en entornos ruidosos). En la Figura 7 se puede observar un entorno de interacción del protocolo CoAP [23].

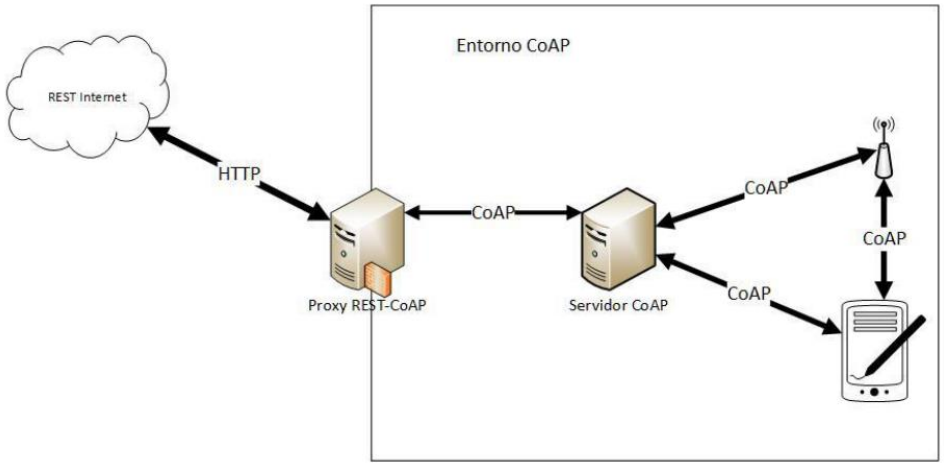

Figura 7. Entorno CoAP.

Fuente: A. J. González García, "IoT: Dispositivos, tecnologías de transporte y aplicaciones," Universitat Oberta de Catalunya, 2017.

3. SSH (Secure SHell): Es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente. A diferencia de otros protocolos de comunicación remota tales como FTP o Telnet, SSH encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas.

   SSH está diseñado para reemplazar los métodos más viejos y menos seguros para registrarse remotamente en otro sistema a través de la shell de comando, tales como telnet o ssh. El uso de métodos seguros para registrarse remotamente a otros sistemas reduce los riesgos de seguridad tanto para el sistema cliente como para el sistema remoto.

   Dentro de las principales características del protocolo SSH podemos mencionar:

   - Después de la conexión inicial, el cliente puede verificar que se está conectando al mismo servidor al que se conectó anteriormente.

   - El cliente transmite su información de autenticación al servidor usando una encriptación robusta de 128 bits.

   - Todos los datos enviados y recibidos durante la sesión se transfieren por medio de encriptación de 128 bits, lo cual los hacen extremamente difícil de descifrar y leer.

   Ya que el protocolo SSH encripta todo lo que envía y recibe, se puede usar para asegurar protocolos inseguros. El servidor SSH puede convertirse en un conducto para convertir en seguros los protocolos inseguros mediante el uso de una técnica llamada reenvío por puerto, como por ejemplo POP, incrementando la seguridad del sistema en general y de los datos [32].

   WebSocket: Websocket es un protocolo que establece una conexión bidireccional entre dispositivos sobre un único socket TCP. Está diseñado para ser implementado en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor, esto hace que pueda amoldarse a más modelos de comunicación [33].

4. MQTT (Message Queuing Telemetry Transport): MQTT es un protocolo de trasporte de mensajes introducido por Andy Standord- Clark de IBM y Arlen Nipper de Arcom (ahora Eurothech) en 1999, estandarizado más tarde en 2013 por OASIS. El objetivo de MQTT es la conexión de dispositivos embebidos, la conexión utiliza un mecanismo de enrutamiento (uno a uno, uno a muchos, muchos a muchos) convirtiendo a MQTT en un protocolo de conexión ideal para IoT y soluciones M2M [34].

   El protocolo MQTT proporciona un método ligero para el envío de mensajes mediante un modelo de publicación/suscripción. Esto lo hace adecuado para la mensajería de Internet de las cosas, como con sensores de baja potencia o dispositivos móviles como teléfonos, computadoras integradas o microcontroladores. Este protocolo se basa en la transmisión de mensajes entre diferentes dispositivos con un elemento central, el broker, que es el que gestiona las comunicaciones de manera centralizada y garantiza que en la mayoría de los casos no haya pérdida de mensajes. Hay múltiples

implementaciones de brokers compatibles con el protocolo MQTT, algunos de los más populares son Mosquitto[10], ActiveMQ[11] o RabbitMQ[12].

La Figura 8 muestra la arquitectura del protocolo MQTT, el cual se compone de tres elementos: subscriber, publisher y broker. Un dispositivo se registra como subscriber en un tema en concreto para que sea informado por el broker, cuando el publisher publique temas de su interés. El publisher transmite la información a las entidades interesadas (subscribers) a través del broker. El broker comprueba que ambos, subscribers y publishers, estén autorizados. Numerosas aplicaciones utilizan el protocolo MQTT, agricultura, salud, monitorización de procesos, mediciones de energía y notificaciones de Facebook. El protocolo es capaz de proporcionar enrutamiento en dispositivos pequeños, de bajo consumo y memoria, en entornos hostiles y de baja tasa de trasmisión [23].

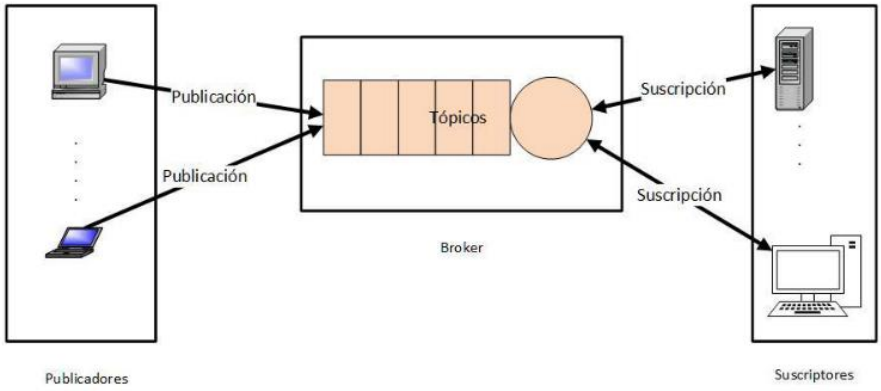

Figura 8. Arquitectura de MQTT.

Fuente: A. J. González García, "IoT: Dispositivos, tecnologías de transporte y aplicaciones," Universitat Oberta de Catalunya, 2017.

5. AMQP (Advanced Message Queueing Protocol). AMQP es un protocolo abierto para mensajería, cuya arquitectura es muy parecida a la de MQTT, aunque AMQP no está pensado para dispositivos de bajos recursos, sino para comunicación entre programas de más alto nivel de abstracción. Aparte de dar soporte a diferentes modelos de comunicación, también ofrece enrutado y encolado de mensajes [35].

6. ZeroMQ: es una librería de código abierto para mensajería universal. Proporciona mensajería distribuida y permite la implementación de todos los principales modelos de comunicación. Está pensado para comunicación entre procesos tanto en el mismo dispositivo, con propósito de paralelismo, como también en remoto. También, está pensado para comunicar programas en diferentes lenguajes y plataformas [36].

[10] https://mosquitto.org/, consultado 10/09/2020. Eclipse Mosquitto es un broker de mensajes de código abierto (con licencia EPL / EDL) que implementa las versiones 5.0, 3.1.1 y 3.1 del protocolo MQTT. Mosquitto es liviano y adecuado para su uso en todos los dispositivos, desde computadoras de placa única de baja potencia hasta servidores completos.

[11] http://activemq.apache.org/, consultado 10/09/2020. Apache ActiveMQ es el servidor de mensajería basado en Java, multiprotocolo y de código abierto más popular. Es compatible con los protocolos estándar de la industria para que los usuarios obtengan los beneficios de las opciones del cliente en una amplia gama de idiomas y plataformas.

[12] https://www.rabbitmq.com/, consultado 10/09/2020. AMQP (Advanced Message Queuing Protocol) es un protocolo de mensajería que permite que las aplicaciones cliente compatibles se comuniquen con los corredores de middleware de mensajería.

2.1.5. Plataformas para IoT

En los últimos años han surgido una gran cantidad de soluciones para interconectar dispositivos y objetos del mundo físico y la vida cotidiana a internet. Paralelo a este incremento de soluciones y dispositivos conectados a internet, se han desarrollado plataformas para visualizar, procesar, almacenar, gestionar información y desplegar aplicaciones con los datos recopilados por estos dispositivos y enviados a internet. Soluciones como las ciudades inteligentes o Smart Cities, la agricultura de precisión o Smart Farming entre otras innovaciones, generan gran cantidad de datos que deben ser procesados para obtener información de valor que permita la eficiente operación de estas tecnologías. Debido a ello, se cuenta en la actualidad con un número considerable de plataformas para IoT, cada uno con sus propias características, funcionalidades y contextos de aplicación.

Dentro de los aspectos más relevantes a considerar para seleccionar una u otra plataforma en la implementación de nuestras soluciones para IoT, se debe considerar: soporte para integración con dispositivos heterogéneos, propiedad de los datos y sus implicaciones para la seguridad y la privacidad, el procesamiento y el intercambio de datos parel soporte de nuevos servicios, soporte para el desarrollo de aplicaciones y la integración completa con ecosistema de IoT [37].

En [37] se muestra el análisis[13] de 38 plataformas IoT privadas y de código abierto que conectan dispositivos y objetos inteligentes a internet y se resume algunas características que los autores consideran fundamentales para satisfacer las expectativas de los usuarios y desarrolladores de aplicaciones. Por lo tanto, esta tabla tiene como objetivo proporcionar información visual rápida que permita seleccionar la plataforma de IoT más adecuada para implementar para un uso o necesidad específica.

En el análisis que se plantea en [37], se realiza una comparación de diferentes aspectos en cada una de las 38 plataformas para IoT. Las variables contempladas son: Soporte de dispositivos heterogéneos, tipo de plataforma, entre los cuales se contemplan plataformas M2M (Machine to Machine) PaaS (Platform as a Service - Plataforma como servicio), plataformas tipo Hub, tipo Client/Server (Cliente servidor). También, se contempla una comparación de arquitecturas de las plataformas, como son: Cloud-based (Basada en la nube), Decentralized (Descentralizada), Centralized (Centralizada) y Centralized/Cloud-based (Centralizada basa en la nube).

Finalmente, se indica si las plataformas analizadas son basadas en open source (código abierto), si soportan servicios REST, si permiten la integración con diferentes protocolos y si poseen servicio de descubrimiento de dispositivos (Service Discovery).

El análisis descrito anteriormente, no sugiere el uso en particular de una plataforma para IoT, ni sitúa a una por encima de otra en términos de preferencias para su uso o recomendación. La información aquí contenida, permite realizar una revisión rápida a las diferentes alternativas de plataformas para IoT y hacer su elección y uso de acuerdo a los requerimientos y funcionalidades necesarios para la solución específica que se esté trabajando.

[13] http://www.internetofthings.fi/results-iot-platforms-analysis.html, consultado 24/09/2020. Análisis de plataformas para IoT.

2.2. WEB SEMÁNTICA DE LAS COSAS

El concepto de Web Semántica de la Cosas (SWoT) es una extensión de la web actual, en ella, los datos tienen un significado bien definido y son comprensibles, tanto para los dispositivos como para las personas, lo que conlleva a que la información pueda ser reusada, compartida y combinada por herramientas software para la creación de nuevos servicios en entornos de interacción de dispositivos inteligentes [38]. En un escenario en el que hay dispositivos heterogéneos desplegados, la conectividad debe suplir dos necesidades, por un lado, conectar los dispositivos y por otro, lograr la cooperación entre ellos en escenarios de interacción semántica por medio del uso de ontologías para cada contexto de uso específico [39].

La implementación de técnicas semánticas ha permitido hacer abstracciones entre hardware, aplicaciones y usuarios de la WoT, con el objetivo de construir servicios inteligentes para diferentes contextos de aplicación, que suplan las necesidades de cada solución particular y los usuarios de éstas, a ello se le ha denominado la web Semántica de los Objetos o "Semantic Web of Things" – SWoT [40].

2.2.1. Ontología

"Una ontología define los términos a utilizar para describir y representar un área de conocimiento. Las ontologías son utilizadas por las personas, las bases de datos, y las aplicaciones que necesitan compartir un dominio de información. Las ontologías incluyen definiciones de conceptos básicos del dominio, y las relaciones entre ellos, que son útiles para los computadores. Codifican el conocimiento de un dominio y también el conocimiento que extiende los dominios. En este sentido, hacen el conocimiento reutilizable" [41].

2.3. OBJETOS INTELIGENTES

Son todos los entes físicos o virtuales que poseen capacidades mínimas de procesamiento, almacenamiento y comunicación de información, en un nivel de abstracción digital soportado por middleware y APIs para implementar los mensajes de comunicación entre ellos [9].

Los objetos sin capacidad de procesamiento, almacenamiento y comunicación, pueden llevarse a Internet de las Cosas a través de un dispositivo que actúe como puerta de acceso a Internet. Se trata, por ejemplo, de interactuar con sistemas de comunicación que permitan intercambiar datos entre los dispositivos, sin necesidad de contar con capacidades de comunicación como las que poseen los dispositivos basados en IP (Internet Protocol).

La visión de IoT es que todos estos dispositivos se comuniquen e interactúen entre sí, siendo capaces de recoger información, procesarla y compartirla. Sin embargo, uno de los principales obstáculos al que se enfrenta IoT es el alto grado de heterogeneidad de las diferentes capacidades de comunicación (protocolos, tecnologías y hardware) de los dispositivos. En particular, el requisito de interoperabilidad es uno de los grandes retos que debe abordarse para la integración y el desarrollo de nuevas plataformas IoT. Sin interoperabilidad el 40% de los beneficios potenciales de IoT no se podrán obtener [42].

La Figura 9 muestra una arquitectura modelo de interacción semántica entre objetos inteligentes [9].

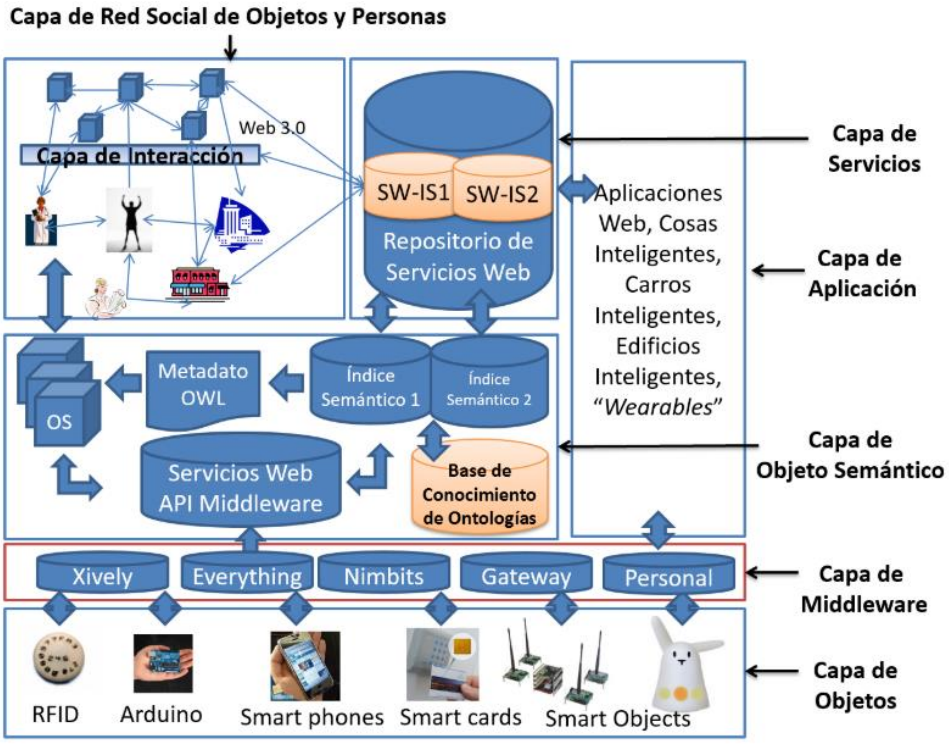

Figura 9. Arquitectura Modelo de Interacción Semántica.

M. Niño Zambrano, "Interacción Semántica de Objetos en la Web de las Cosas - Semantic Object Interaction on Web of Things," Universidad del Cauca, 2016.

En el trabajo realizado por Niño-Zambrano en [9], se propone una Arquitectura y Concepto de Objeto Inteligente mostrada en la Figura 9, sobre la cual se realiza el despliegue e interacción de objetos inteligentes a través de la web, aportando a la solución del problema de heterogeneidad que dificulta la interoperabilidad de los dispositivos en la IoT.

A continuación se definen cada una de las capas que componen la arquitectura de la Figura 9.

4.1.1. Capa de Objetos

Se encuentra la representación física de los objetos y se definen los recursos asociados.

4.1.2. Capa Middleware

Se encarga de resolver la heterogeneidad y conectividad de los diferentes objetos. Almacenará los conceptos para representar semánticamente al objeto y los recursos asociados.

4.1.3. Capa de Objeto Semántico

Encargada de alojar el motor semántico, el cual contiene los siguientes elementos:

4.1.3.1. Ontología Objeto Semántico (OOS): Ontología que contiene los metadatos del objeto.

4.1.3.2. Índice Semántico: Captura y comparte la información contextualizada de los objetos indexados.

4.1.4. Capa de Servicio

Encargado de almacenar la información del índice semántico, encapsularla y exponerla a través de servicios web, haciendo posible que las aplicaciones y los objetos puedan consumir los metadatos, datos y demás información de contexto de manera interoperable y transparente. La capa de servicio recupera información sobre los objetos en el middleware, luego la información del objeto es utilizada por el índice semántico, generando listas de objetos contextualizados, los métodos expuestos por el servidor son utilizados por el objeto semántico para recuperar sus metadatos y son utilizados por la capa de aplicación para recuperar objetos según un concepto determinado.

4.1.5. Capa de Interacción

Soporta las herramientas que permiten comunicar a los objetos con las aplicaciones de los usuarios y con otros objetos.

4.1.6. Capa de Aplicaciones

Permite al sistema integrar a los usuarios con los objetos.

2.4. GATEWAY INTELIGENTE IoT

Es un componente inteligente basado en una plataforma IoT [43], [44], que además de actuar como interfaz de conexión entre aparatos o dispositivos con una red, permite la conversión de datos entre los protocolos de comunicación de corta distancia a la red de comunicación tradicional, también admite diferentes tipos de nodos sensores, múltiples protocolos de comunicación, tanto inalámbricos como cableados, y proporciona un conjunto de información unificada para diferentes aplicaciones o el usuario, por lo cual, es también responsables del procesamiento de datos.

En las redes de datos convencionales, los Gateways, pasarelas o puertas de enlace tienen la función de posibilitar la conexión de dispositivos de la red local LAN con otras redes, generalmente de internet (WAN). Para aplicaciones de IoT, los Gateway deben permitir la conectividad entre los dispositivos o cosas (Things) con las plataformas, servicios y aplicaciones de IoT.

El objetivo básico del Gateway IoT es permitir la conectividad ubicua[14] entre dispositivos y plataformas para IoT, resolviendo la heterogeneidad entre diferentes redes de punto final y la red de internet. Para ello, debe cumplir con el requisito de permitir la interoperabilidad a nivel de dispositivo, protocolos de comunicación y datos.

[14] Para lograr la conectividad entre las cosas y la IoT, se requiere tener en cuenta la conectividad ubicua. Las capacidades de conectividad deben ser independientes de los dominios de aplicación específicos y soportar la integración de tecnologías de comunicación heterogéneas [10].

2.4.1. Interoperabilidad

Se considera como la capacidad de dos o más sistemas o componentes para intercambiar datos y utilizar la información [42]. En el contexto de IoT la interoperabilidad puede establecerse a diferentes niveles:

2.4.1.1. Interoperabilidad a nivel de dispositivo

Consiste en integrar de manera transparente dispositivos que soportan hardware y software heterogéneos [45]. A este nivel de interoperabilidad, que supone la interacción objeto a objeto, se proponen en [9] modelos de interacción entre dispositivos y objetos de la IoT, los cuales sirvan como mecanismos para crear servicios más complejos.

2.4.1.2. Interoperabilidad a nivel de protocolos

Consiste en habilitar la comunicación directa entre diferentes tecnologías de red [46]. Un estrategia para afrontar la dificultad en acceder, describir y comunicar sensores que utilizan hardware, software y protocolos de diferentes proveedores es implementar componentes software mediadores denominados middleware y servidores IoT, que por medio de anotaciones semánticas y librerías de desarrollo (Application Program Interface - APIs), se encargan de resolver las diferencias entre los diferentes formatos e interfaces [9].

2.4.1.3. Interoperabilidad a nivel de datos

Está asociada con el formato de los datos de los dispositivos. Los mensajes transmitidos por los protocolos necesitan una sintaxis y codificación definida [15], así como también, la definición de una estrategia para comunicar conceptos y conocimientos, que normalmente se realiza con técnicas de web semántica.

Solucionar el problema de interoperabilidad permite eliminar los denominados ecosistemas cerrados o silos verticales de la información [47], obteniendo el verdadero valor de IoT que reside en los datos que se crean y transmiten cuando los dispositivos interactúan entre sí. Con esto, también se obtiene la posibilidad de realizar un despliegue de diferentes tipos de dispositivos sensores en los cultivos agrícolas, sin que estos tengan que soportar la complejidad o el coste de una interfaz de alta velocidad a internet con el fin de estar conectados.

Grupos de estandarización como la ITU [48] y estudios en [49] proponen la utilización de Gateways para habilitar la interoperabilidad de IoT, incorporando en este dispositivo toda la funcionalidad requerida como, por ejemplo, el procesamiento y almacenamiento de la información de los datos de los dispositivos. Las investigaciones presentes hasta la fecha se han centrado en el diseño y desarrollo de Gateways IoT para solucionar la interoperabilidad de protocolos y tecnologías de comunicación específicos, sin considerar las capacidades limitadas de los dispositivos IoT [8].

Para la realización de este trabajo de grado se tomó como referencia el modelo de interacción semántica en la Web de las Cosas (Web of Things, WoT) presentado en [9], el cual, identifica los elementos básicos que se deben tener en cuenta al momento de realizar interacción entre los objetos de la WoT. El modelo define tres capas importantes: capa objeto semántico “Semantic Object, SO”, capa redes sociales de objetos y personas “Social

Networks of Things and People, SNTP" y capa web de las cosas y las personas "Web of Things and People, WoTP". También propone el concepto de objeto semántico (OS), que aparece como un integrador de los diferentes miembros del SNTP en un sólo elemento; por lo tanto, una persona y un objeto se pueden representar a través de este concepto único, que integra tres diferentes representaciones, pero complementarias.

El trabajo presentado por Niño-Zambrano [9] también propone una arquitectura a implementar, en la cual se hace uso de middlewares y servidores IoT (*middleware layer*), como mediadores para resolver el problema de heterogeneidad de hardware y protocolos existentes en la capa de objetos (*object layer*). Al mismo tiempo, se construye un índice semántico de los objetos explorados, basándose en ontologías de dominio específico para crear un contexto por ontología particular, clasificando el uso de los diferentes sensores en estos contextos. La arquitectura desarrollada en este trabajo se tomó como base fundamental para implementar el dispositivo Gateway Inteligente para IoT.

2.4.2. Arquitecturas y Modelos de Referencia de Gateways para Aplicaciones de IoT

En la Figura 10 se muestra una situación típica de despliegue de gateways para aplicaciones de IoT según la Rec. UIT-T Y.4101/Y.2067 (10/2017), en la cual, se puede ver que es posible conectar dispositivos heterogéneos a redes de comunicación a través de uno o varios gateways. La conectividad entre los dispositivos y los gateways puede basarse en diferentes tipos de tecnologías alámbricas o inalámbricas como ZigBee (IEEE 802.15.4), Bluetooth (IEEE 802.15.1), Wifi (IEEE 802.11) y Ethernet (IEEE 802.3) entre otras.

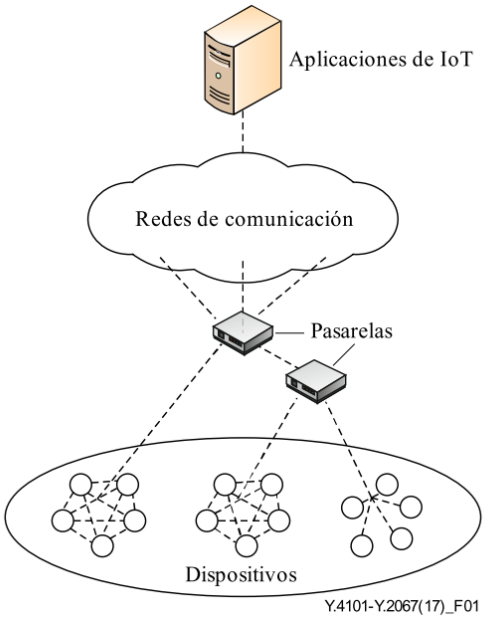

Figura 10. Aplicación típica de despliegue de gateways para aplicaciones de IoT.

Fuente: International Telecommunication Union (ITU), "Common requirements and capabilities of a gateway for Internet of Things applications," p. 26, 2017.

Basado en el "Marco técnico de referencia de una pasarela para aplicaciones de IoT" [48] y en el "Internet of Things – Architecture IoT-A Deliverable D1.5 – Final architectural reference model for the IoT v3.0" [46], se definieron los módulos, capas y las respectivas interacciones de la arquitectura propuesta para la implementación del Gateway Inteligente

IoT. El marco técnico de referencia [48], está compuesto por cuatro grupos de capacidades comunes a los gateways para aplicaciones de IoT, las cuales, establecen funcionalidades que se desarrollan en cada una de las capas del modelo:

- **Grupo de capacidades de aplicaciones**

  El grupo de capacidades de aplicaciones proporciona soporte para interactuar con aplicaciones remotas y para el procesamiento local de la lógica de la aplicación. Soporta el despliegue de múltiples aplicaciones de IoT de diferentes tipos, utilizadas en diferentes ámbitos, como la medición de variables para soluciones de agricultura de precisión, medición de energía en hogares inteligente, monitoreo de signos vitales de pacientes para soluciones de e-salud, entre otros.

- **Grupo de capacidades de soporte**

  El grupo de capacidades de soporte proporciona la gestión de dispositivos para comunicar perfiles de dispositivos con el propio gateway y las aplicaciones; también, proporciona gestión de comunicaciones; almacenamiento de datos, que incluye capacidades de almacenamiento permanente y temporal de datos recopilados de dispositivos, datos de configuración del gateway y datos de aplicaciones; procesamiento de datos, incluido el análisis de datos, la transformación de formatos de datos, la habilitación de la mediación semántica, la encapsulación de datos basada en protocolos de aplicación y la agregación de datos de dispositivos; despacho de datos, que proporciona capacidades de pre-procesamiento de datos provenientes de aplicaciones de acuerdo con las disposiciones pertinentes y de optimización de distribución de datos y gestión de servicios, con la que se administran los servicios de los dispositivos conectados al gateway.

- **Grupo de capacidades de adaptación**

El grupo de capacidades de adaptación proporciona capacidades de comunicación con dispositivos y aplicaciones, y de ocultación de diferencias entre dispositivos y aplicaciones. Este grupo incluye capacidades de abstracción de interfaces que soportan operaciones básicas (como la lectura de datos de un dispositivo) para interactuar con dispositivos y aplicaciones, y también proporciona capacidad de correspondencia entre una interfaz abstracta e interfaces específicas soportadas por dispositivos y aplicaciones. Este grupo de capacidades de adaptación también proporciona adaptación de dispositivos, que proporciona conectividad de diferentes tipos de dispositivos u otras pasarelas que se conectan a la pasarela; adaptación de red, que permite la adaptación a diferentes tecnologías de red, incluida la adaptación de la capa física/control de acceso a los medios entre la pasarela y las redes de comunicación.

- **Grupo de capacidades de gestión y seguridad**

El grupo de capacidades de gestión y seguridad proporciona capacidades de soporte a la seguridad y la gestión del propio gateway.

En la Figura 11 se muestras el marco técnico de referencia de un gateway para aplicaciones de IoT propuesto por la ITU.

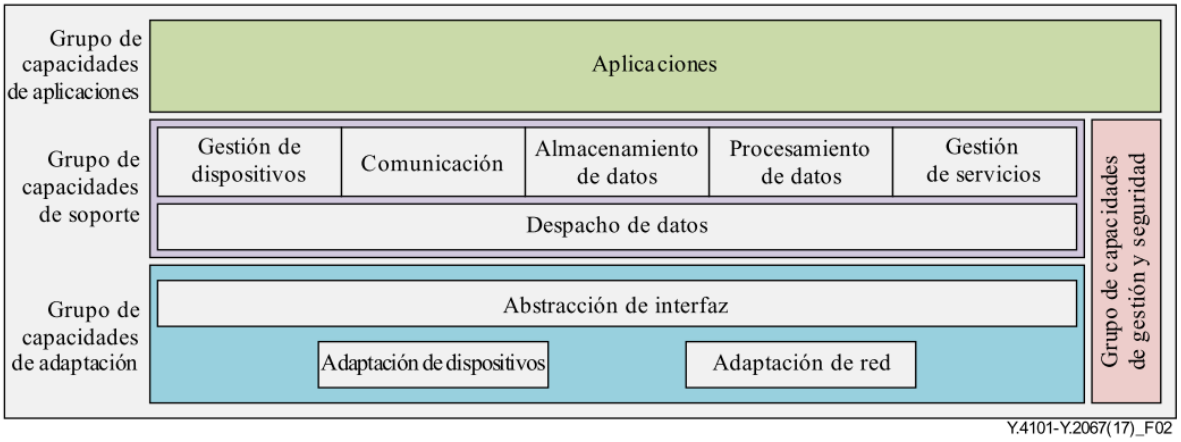

Figura 11. Marco técnico de referencia de una pasarela para aplicaciones de IoT

Fuente: International Telecommunication Union (ITU), "Common requirements and capabilities of a gateway for Internet of Things applications," p. 26, 2017.

Otro importante instrumento para la definición, diseño y construcción de la arquitectura del Gateway Inteligente IoT, es el Diagrama de Modelo Funcional de IoT [46], el cual brinda una abstracción global de todo los elementos, procesos y su interacción en soluciones de IoT para diferentes entornos. Este modelo contempla una capa de bajo nivel de dispositivos, interconectados a través de una capa de comunicación con una variedad de servicios y funcionalidades, tales como servicios de IoT, entidad para virtualizar recursos y/o servicios, administrador de procesos de IoT y un módulo de organización de servicios; todos estos interactuando con la capa de alto nivel de aplicaciones y recibiendo respaldo de procesos de seguridad y gestión dentro del modelo. En la Figura 12 se muestra la arquitectura del Modelo Funcional de IoT.

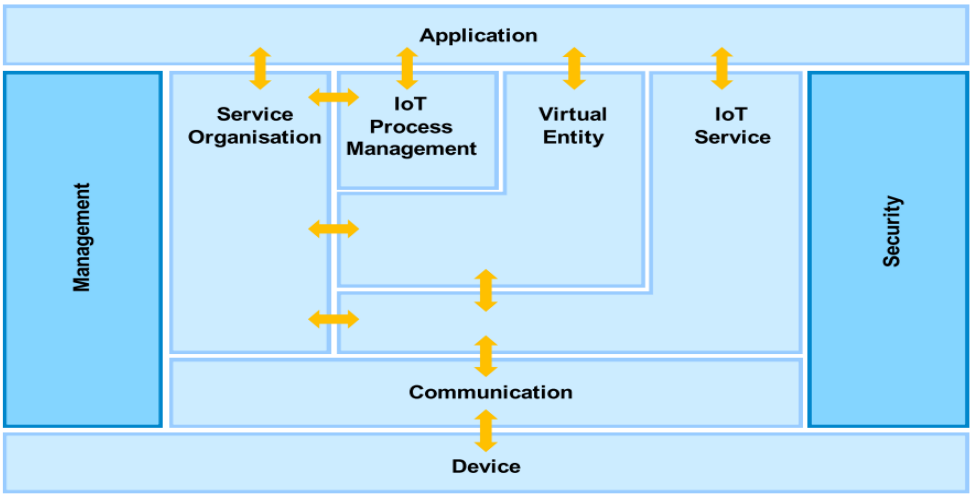

Figura 12. Modelo Funcional IoT.

Fuente: M. Bauer et al., Internet of Things – Architecture IoT-A Deliverable D1.5 – Final architectural reference model for the IoT v3.0. 2013.

Este mismo modelo mostrado en la figura anterior, representa la arquitectura de referencia IoT de [46]. En la Figura 13 se muestra una vista de la descomposición funcional de la arquitectura, que constituye una referencia estándar para el diseño de soluciones IoT.

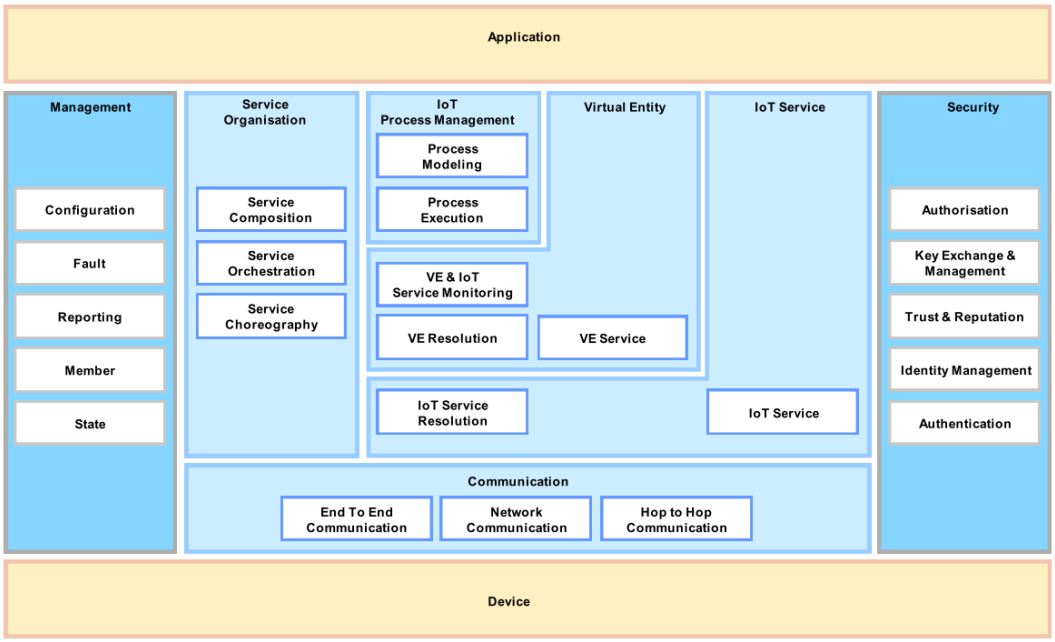

Figura 13. Vista de la descomposición funcional de la arquitectura de referencia IoT.

Fuente: M. Bauer et al., Internet of Things – Architecture IoT-A Deliverable D1.5 – Final architectural reference model for the IoT v3.0. 2013.

2.4.3. Características Generales de un Gateway para Aplicaciones de IoT

La ITU recomienda [48] la implementación de un grupo de características específicas en los dispositivos gateway para aplicaciones de IoT, esto con el objetivo de garantizar la conectividad de dispositivos heterogéneos, la conexión con redes de comunicación externas e internet, la gestión de los diferentes dispositivos y servicios desplegados en el gateway, así como también, la interacción y soporte entre aplicaciones y capacidades de seguridad. A continuación, se listan las características mencionadas:

- Conexión a redes de comunicación
- Acceso de dispositivo
- Traducción de protocolos
- Interacción y soporte de aplicaciones
- Adaptabilidad
- Soporte de las funciones de gestión
- Soporte de las funciones de seguridad

2.4.4. Requisitos Comunes de un Gateway para Aplicaciones de IoT

Esta sección corresponde a los requisitos que debe tener un gateway para aplicaciones de IoT de acuerdo a la recomendación [48] UIT-T Y.4101/Y.2067 de la ITU para su implementación. Los siguientes son requisitos obligatorios y deben aplicarse sin excepción para garantizar el funcionamiento del gateway dentro de los estándares fijados por la ITU.

- Escalabilidad
- Direccionamiento
- Apertura a extensiones funcionales
- Calidad de servicio
- Aspectos de comunicación
- Soporte de diversidad de protocolos
- Uniformidad e interacciones
- Descubrimiento de dispositivos y servicios
- Gestión de dispositivos
- Gestión de servicios
- Gestión de identificadores de dispositivos
- Almacenamiento
- Agrupación de dispositivos
- Recopilación y agregación de datos
- Envío y entrega de datos
- Integración de la lógica de aplicaciones
- Análisis de datos
- Seguridad y confidencialidad
- Autogestión y mantenimiento remoto

2.5. METEOROLOGÍA

La meteorología es la ciencia encargada del estudio de la atmósfera, de sus propiedades y de los fenómenos que en ella tienen lugar, los llamados meteoros, entendiendo por "meteoro", en el más amplio sentido, todos los fenómenos físicos naturales que tienen lugar en la atmósfera, los cuales han sido clasificados por la Organización Meteorológica Mundial (OMM) en los siguientes cuatro grupos:

- *Hidrometeoros:* los relacionados con el vapor de agua en cualquiera de sus estados físicos, tales como: la lluvia, el granizo, el rocío y la escarcha.

- *Litometeoros:* compuestos por residuos sólidos pertenecientes a la litosfera: calima, humo, polvareda, tempestad de polvo o arena.

- *Fotometeoros:* aquellos formados por la interacción de la luz solar o lunar con la atmósfera: el arco iris, la corona y los fenómenos de halo.

- *Electrometeoros:* asociados con la electricidad atmosférica: tormenta, relámpago, trueno.

El estudio de los meteoros conduce al conocimiento de los diferentes estados de la atmosfera (si llueve, si hace calor, si el viento es fuerte o si el cielo está despejado). De las variaciones de éstos estados y de las leyes físicas que los rigen, la meteorología puede explicar el por qué se producen diversas manifestaciones del tiempo y puede deducir también cual será el estado futuro. Algunos de los métodos que utiliza la meteorología son: el método empírico, que se basa en observaciones locales, el método de cálculo de probabilidades, basado en la estadística, y los modelos matemáticos, basados en las ecuaciones matemáticas de la física [50].

El objetivo de la meteorología es predecir el tiempo que va a hacer en 24 o 48 horas y, en menor medida, elaborar un pronóstico del tiempo a medio plazo, esto se logra mediante el estudio de los cambios atmosféricos que se producen a cada momento, utilizando parámetros como la temperatura del aire, su humedad, la presión atmosférica, el viento o las precipitaciones

2.6. CLIMATOLOGÍA

La climatología es la ciencia que estudia la serie de estados atmosféricos que se suceden habitualmente en un determinado lugar. Está basada en el estudio de los datos meteorológicos. Por su parte, la meteorología es la ciencia que estudia los fenómenos que tienen lugar en la atmósfera terrestre [51].

La climatología o estudio de los climas, por lo tanto, constituye una especie de “memoria” que retiene los aspectos principales de los estados del tiempo ya pasados. Utiliza métodos estadísticos por medio de los cuales describe los climas, así como métodos dinámicos, que se basan en la mecánica general y la termodinámica de la atmosfera. La climatología no se limita sólo a describir el clima a través del tratamiento estadístico, sino que también toma muy en cuenta los factores geográficos, tanto regionales como locales, por la influencia que estos ejercen en el comportamiento de los elementos del clima.

Debido al gran avance de la informática en los últimos años, ha tomado mucho auge el uso de los modelos climáticos. Estos modelos describen los climas de la Tierra y permiten analizar las posibles variaciones causadas por alteraciones en algunos de los elementos climatológicos. A los valores reales o instantáneos de los elementos que definen el estado del tiempo se les llama elementos meteorológicos y a los valores medios o normales de estos, que son los que definen los climas, se les llama elementos climatológicos [50].

2.6.1. El Clima

El clima de un lugar es el tiempo que hace normalmente en ese lugar a lo largo de los meses y los años. La Organización Meteorológica Mundial (O.M.M.) en la Conferencia de Varsovia (1935) definió como clima las condiciones meteorológicas medias para el mes y el año, calculadas sobre un período de 30 años [51].

El clima puede explicarse mediante descripciones estadísticas de las tendencias y la variabilidad principales de elementos pertinentes, como la temperatura, la precipitación, la presión atmosférica, la humedad y los vientos, o mediante combinaciones de elementos, tales como tipos y fenómenos meteorológicos, que son característicos de un lugar o región, o del mundo en su conjunto, durante cualquier periodo de tiempo [52].

2.6.2. Elementos del Clima

El clima de un lugar está definido por los elementos y los factores climatológicos, a su vez, los elementos del clima, son cada uno de los aspectos físicos que integran:

- Temperatura del aire
- Presión atmosférica
- Viento
- Humedad del aire

- Precipitaciones

Estos siempre están presentes en el clima de cualquier parte del planeta. Es decir, el clima de un punto de la Tierra siempre estará determinado por todos los elementos climáticos.

2.6.2.1. Factores del Clima

Son los agentes que influyen y/o modifican el comportamiento de cada uno de los elementos del clima. Cada factor actúa sobre todos ellos, aunque en grado desigual, por ejemplo: el relieve, la altitud, las corrientes marinas, la distribución de tierras y océanos, la continentalidad entre otros.

Los elementos y factores del clima son los aspectos que hacen posible establecer diferencias en cuanto a los climas del planeta. Los elementos son aspectos que siempre están presentes en cualquier clima; así, por ejemplo, si tomamos la ciudad de Popayán, el clima en esta localidad tendrá una cierta cantidad de radiación solar, temperatura, presión y humedad. Esto mismo se repite para cualquier otra localidad. Los factores del clima que inciden en ese lugar no son los mismos para todo el territorio, es decir, los factores pueden estar presentes o no en su totalidad o parcialmente [50].

2.6.3. El Sistema Climático

El sistema climático es un conjunto interactivo y complejo constituido por la atmosfera, la superficie terrestre, la nieve y el hielo, los océanos y otras masas de agua y organismos vivos. La Figura 14 muestra el sistema climático con sus diferentes componentes e interacción entre ellos.

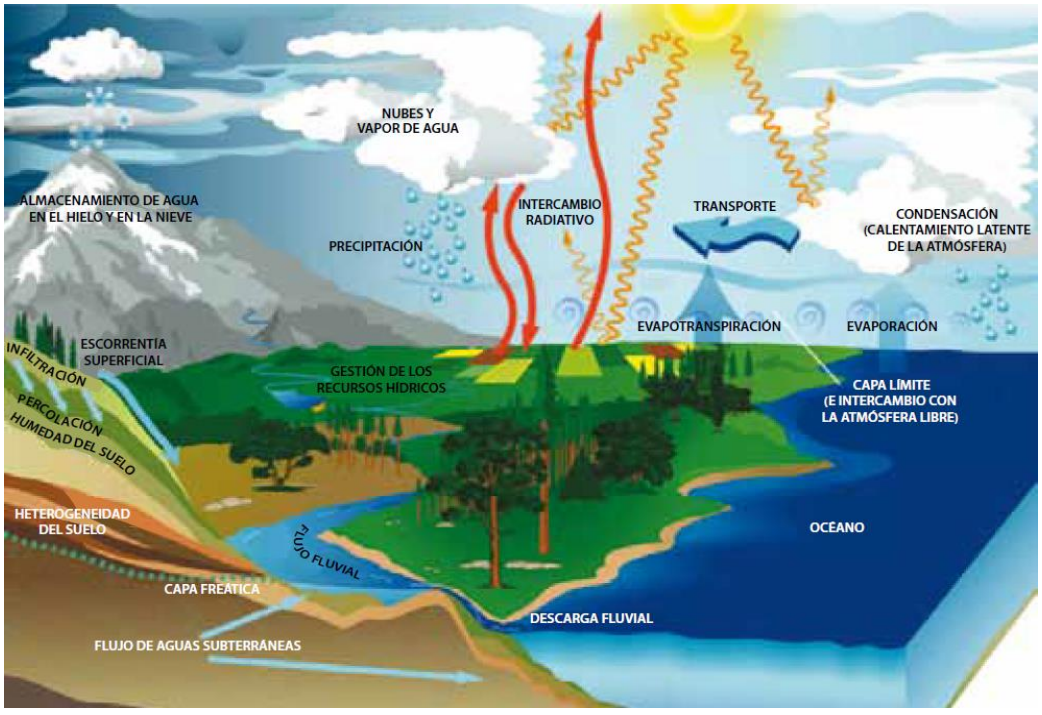

Figura 14. El Sistema Climático.

Fuente: Organización Meteorológica Mundial, "Guía de Prácticas Climatológicas," vol. 100, 2011.

- La atmosfera es la capa gaseosa que envuelve la Tierra. La atmosfera seca está compuesta casi íntegramente de nitrógeno y oxígeno, pero también contiene pequeñas cantidades de argón, helio, dióxido de carbono, ozono, metano y muchos otros gases. La atmosfera también contiene vapor de agua, gotitas de agua condensada en forma de nubes y aerosoles.

- La hidrosfera es la parte del sistema climático de la Tierra que comprende el agua líquida distribuida sobre y bajo la superficie de la Tierra en océanos, mares, ríos, lagos de agua dulce, embalses subterráneos y otras masas de agua.

- La criósfera abarca el conjunto de elementos del sistema de la Tierra que contienen agua en estado de congelación e incluye toda la nieve y el hielo.

- La litosfera es la capa superior de la parte sólida de la Tierra, que comprende tanto la corteza continental como los fondos marinos.
- La biosfera engloba todos los ecosistemas y organismos vivos presentes en la atmósfera, en tierra firme (biosfera terrestre) y en los océanos (biosfera marina), incluida la materia orgánica muerta resultante de ellos, como restos, materia orgánica del suelo o desechos oceánicos.

2.7. AGROCLIMATOLOGÍA

La agroclimatología es una ciencia relativamente reciente, se ha desarrollado como una necesidad a partir de la intrínseca relación que existe entre la agricultura[15] y los fenómenos y procesos del clima [53]. El clima de un lugar es el tiempo que hace normalmente en ese lugar a lo largo de los meses y los años. La Organización Meteorológica Mundial (OMM) en la Conferencia de Varsovia (1935) definió como clima: las condiciones meteorológicas medias para el mes y el año, calculadas sobre un periodo de 30 años [12]. Si bien el clima no es el único factor del medio físico cuya influencia se deja sentir en la actividad agrícola, sí que es cierto que es el que está sometido a variaciones más bruscas en el tiempo, y por lo tanto sus efectos son más visibles a corto plazo. En efecto, en pocas semanas e incluso en días, el tiempo puede convertirse en un importante factor de riesgo que conlleve a irremediables pérdidas sustanciales o totales en las cosechas. Organismos internacionales como la Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO) y la OMM se han propuesto como objetivo difundir el alcance de la climatología agrícola, las investigaciones que se han ido llevando a cabo y sus aplicaciones. Además de ellos, son numerosos los institutos y departamentos universitarios que incluyen la agroclimatología entre sus líneas de investigación, es el caso del Instituto Internacional de Investigaciones para el Clima y la Sociedad - IRI (https://iri.columbia.edu/) de la Universidad de Columbia, el Centro de Pronóstico del Tiempo y Estudios Climáticos - CPTEC (https://www.cptec.inpe.br/) del Instituto Nacional de Investigación Espacial en Brasil, el Instituto Nacional para la Investigación Agronómica (Institut National de la Recherche Agronomique - INRA) en Francia (https://www.inrae.fr/) y la Administración Nacional Oceánica y Atmosférica (National Oceanic and Atmospheric Administration - NOAA) que es

[15] https://es.wikipedia.org/wiki/Agricultura, consultado 11/07/2019. La agricultura es el conjunto de técnicas y conocimientos para cultivar la tierra. Las actividades relacionadas con esta, son las que integran el llamado sector agrícola. Todas las actividades económicas que abarca dicho sector tienen su fundamento en la explotación de los recursos que la tierra origina, favorecida por la acción del hombre: alimentos vegetales como cereales, frutas, hortalizas, pastos cultivados y forrajes; fibras utilizadas por la industria textil; cultivos energéticos etc.

una agencia científica del Departamento de Comercio de los Estados Unidos (https://www.noaa.gov/), por citar algunos.

En Colombia, estos estudios se han llevado a cabo de la mano de investigadores pertenecientes al Instituto de Hidrología, Meteorología y Estudios Ambientales[16] (IDEAM), institución pública de apoyo técnico y científico al Sistema Nacional Ambiental, que genera conocimiento, produce información confiable, consistente y oportuna, sobre el estado y las dinámicas de los recursos naturales y del medio ambiente.

Es importante el aporte que también realiza el sector privado y algunas universidades del país, es el caso del Instituto de Estudios Ambientales Interfacultades de la Universidad Nacional de Colombia (IDEA), que dentro de sus funciones contempla: formular, orientar y desarrollar proyectos de investigación interdisciplinaria y programas en el campo de los estudios ambientales.

De todas las entidades y organismos mencionados, cabe resaltar el trabajo que viene realizando el Centro Nacional de Investigaciones de Café - Cenicafé, división de investigación científica de la Federación Nacional de Cafeteros de Colombia[17], quienes han monitoreado el clima de la zona cafetera colombiana durante los últimos 65 años.

En el año 2014, la Federación Nacional de Cafeteros y el Centro Nacional de Investigaciones de Café lanzaron una herramienta que facilita a los caficultores tomar decisiones a partir de las condiciones meteorológicas y geográficas de cada región. La nueva plataforma se apoya con los datos históricos recopilados por Cenicafé a través de su red de 231 puntos de observación convencionales y automáticos. La moderna Plataforma Agroclimática Cafetera consolida a Colombia como país a la vanguardia en materia de desarrollos tecnológicos para el sector cafetero y servirá de apoyo a los caficultores, extensionistas e investigadores, en la toma de decisiones sobre el manejo del cultivo del café [54].

En la Figura 15 se muestra un mapa con la distribución de Estaciones Meteorológicas Automáticas instaladas en la región cafetera de Colombia, el cual permite acceder a los datos de una región específica, ya sea seleccionando la estación de interés o escribiendo el nombre de la Estación, Departamento o Municipio donde se encuentra ésta instalada. Tomando como referencia el Departamento del Cauca, se observa en la Figura 16 que cuenta con 7 estaciones distribuidas en toda la región, de las cuales al seleccionar una, se presentan los registros a nivel horario de las variables meteorológicas: lluvia (mm), temperatura (°C) y humedad relativa (%), registradas en las últimas 24 horas.

[16] http://www.ideam.gov.co/web/entidad/, consultado 11/07/2019. El Instituto de Hidrología, Meteorología y Estudios Ambientales (IDEAM) es una institución pública adscrita al Ministerio de Ambiente y Desarrollo de la República de Colombia.

[17] https://federaciondecafeteros.org/, consultado 11/07/2019. En 1927 los cafeteros colombianos se unieron con el fin de crear una organización que los representara nacional e internacionalmente, y que velara por su bienestar y el mejoramiento de su calidad de vida. Así nació la Federación Nacional de Cafeteros de Colombia (FNC), considerada hoy como una de las ONG rurales más grandes del mundo. La Federación es una entidad sin ánimo de lucro, y no está afiliada a ningún partido político.

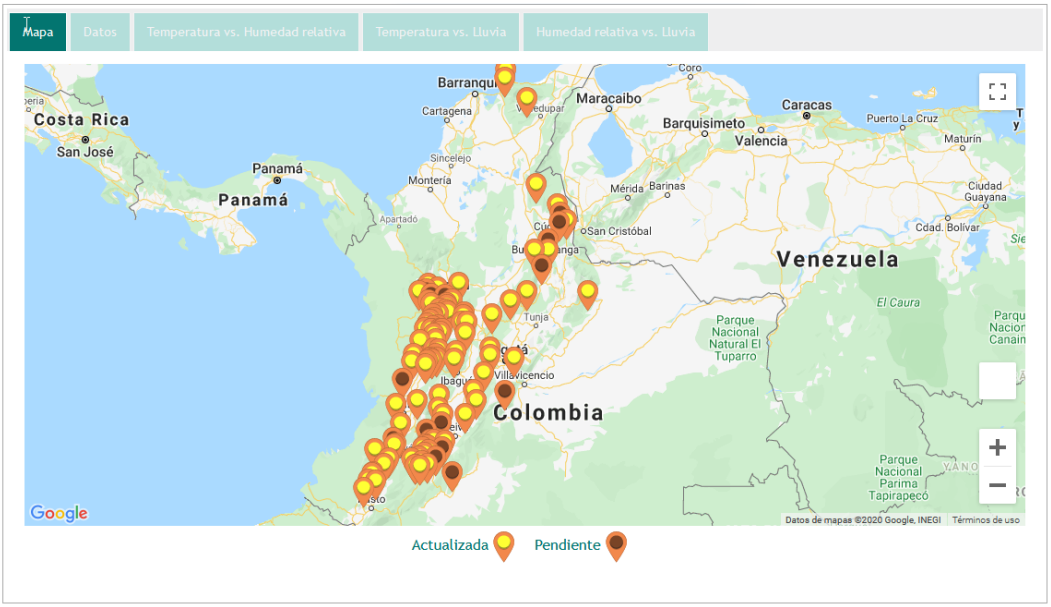

Figura 15. Plataforma Agroclimática Cafetera.

Fuente: Centro Nacional de Investigaciones de Café - Cenicafé, "Agroclima - Plataforma Agroclimática Cafetera." [Online]. Available: https://agroclima.cenicafe.org/. [Accessed: 11-Jul-2019].

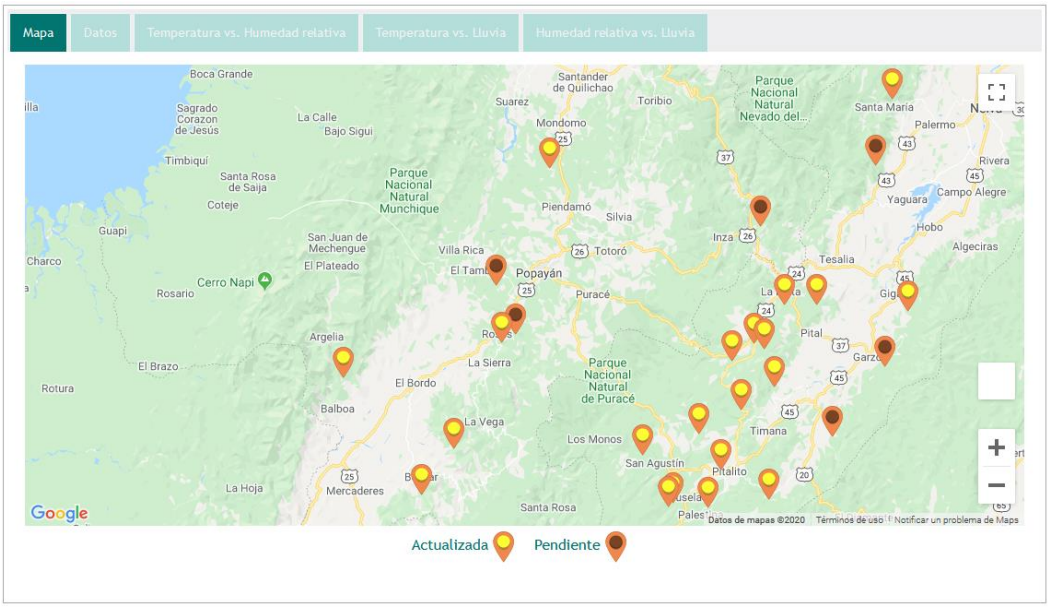

Figura 16. Estaciones Meteorológicas Automáticas instaladas en el Departamento del Cauca.

Fuente: Centro Nacional de Investigaciones de Café - Cenicafé, "Agroclima - Plataforma Agroclimática Cafetera." [Online]. Available: https://agroclima.cenicafe.org/. [Accessed: 11-Jul-2019].

En la Tabla 1 se muestran los datos registrados en las últimas 24 horas por una de las estaciones

Mapa | Datos | Temperatura vs. Humedad relativa | Temperatura vs. Lluvia | Humedad relativa vs. Lluvia

La Trinidad - Piendamó, Cauca

Día	Hora	Temperatura (°C)	Humedad relativa (%)	Lluvia (mm)
2020-01-25	17	17.7	85.7	6.9
2020-01-25	18	16.5	98.3	0.2
2020-01-25	19	16.9	95.0	0.0
2020-01-25	20	16.6	99.6	0.0
2020-01-25	21	16.7	97.5	0.0
2020-01-25	22	16.3	97.3	0.0
2020-01-25	23	16.3	98.4	0.0
2020-01-25	24	16.7	99.5	0.0
2020-01-26	1	16.7	100.0	0.0
2020-01-26	2	16.8	100.0	0.0
2020-01-26	3	16.8	100.0	0.0
2020-01-26	4	16.7	100.0	0.1
2020-01-26	5	16.5	100.0	0.0
2020-01-26	6	16.3	100.0	0.0
2020-01-26	7	16.2	100.0	0.0
2020-01-26	8	17.4	94.4	0.0
2020-01-26	9	19.8	81.7	0.0
2020-01-26	10	20.4	80.0	0.0
2020-01-26	11	22.8	68.9	0.0
2020-01-26	12	24.9	59.3	0.0
2020-01-26	13	24.0	64.0	0.0
2020-01-26	14	23.9	68.4	0.0
2020-01-26	15	23.7	68.4	0.0
2020-01-26	16	19.9	83.2	0.0

Tabla 1. Registros a nivel horario de las variables meteorológicas: lluvia (mm), temperatura (°C) y humedad relativa (%), registradas en las últimas 24 horas en la Estación Meteorológica Automática La Trinidad – Piendamó, Cauca.

Fuente: Centro Nacional de Investigaciones de Café - Cenicafé, "Agroclima - Plataforma Agroclimática Cafetera." [Online]. Available: https://agroclima.cenicafe.org/. [Accessed: 11-Jul-2019].

Además de los datos e información antes mencionada, la Plataforma Agroclimática Cafetera también permite obtener gráficas de las diferentes variables meteorológicas que en ella se registran. Estas gráficas se podrán imprimir directamente desde la plataforma o podrán ser descargadas en formatos de imagen PNG, JPEG, SVG y formato de documento portable PDF. La Figura 17 muestra la gráfica de humedad vs temperatura generada desde la plataforma.

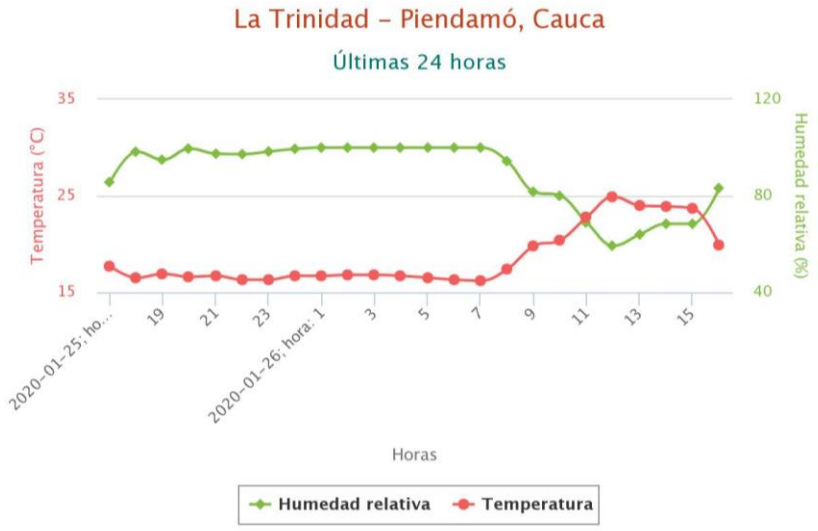

Figura 17. Gráfica de Humedad y Temperatura, Plataforma Agroclimática Cafetera.

Fuente: Centro Nacional de Investigaciones de Café - Cenicafé, "Agroclima - Plataforma Agroclimática Cafetera." [Online]. Available: https://agroclima.cenicafe.org/. [Accessed: 11-Jul-2019].

La Figura 18 muestra la gráfica de lluvia y temperatura generada desde la Plataforma Agroclimática Cafetera.

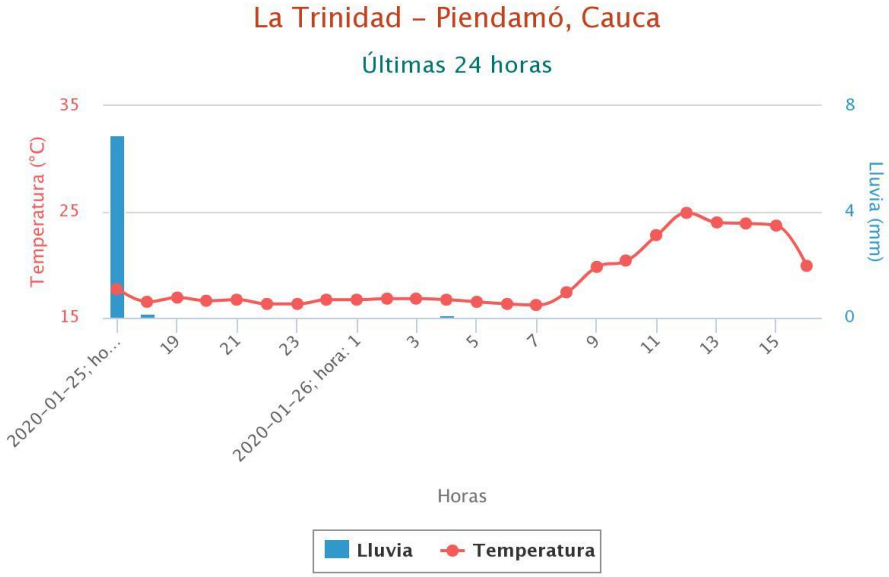

Figura 18. Gráfica de Lluvia y Temperatura, Plataforma Agroclimática Cafetera.

Fuente: Centro Nacional de Investigaciones de Café - Cenicafé, "Agroclima - Plataforma Agroclimática Cafetera." [Online]. Available: https://agroclima.cenicafe.org/. [Accessed: 11-Jul-2019].

La Figura 19 muestra la gráfica de lluvia y humedad relativa generada desde la Plataforma Agroclimática Cafetera.

Figura 19. Gráfica de Lluvia y Humedad relativa, Plataforma Agroclimática Cafetera.

Fuente: Centro Nacional de Investigaciones de Café - Cenicafé, "Agroclima - Plataforma Agroclimática Cafetera." [Online]. Available: https://agroclima.cenicafe.org/. [Accessed: 11-Jul-2019].

2.8. TRABAJOS RELACIONADOS

Dentro de la exploración realizada para este trabajo de grado se encontraron varios diseños, modelos, arquitecturas y tecnologías propuestas para la implementación de un Gateway Inteligente para IoT.

Importantes trabajos como los desarrollados en [55], [56] plantean modelos de arquitecturas con diferentes contextos de aplicación para la solución implementada, dentro de ellos, el de la agricultura de precisión [57].

Un sistema IoT para el control y monitorización de un terrario[18] con Eclipse Kura, Amazon AWS y Angular se plantea en [58]. El objetivo de este trabajo es desarrollar un sistema conformado por dispositivos sensores de temperatura y humedad conectados a una Raspberry Pi, la cual es configurada como un dispositivo IoT haciendo uso de Eclipse Kura, un contenedor de aplicaciones Java, que permite convertir un dispositivo en un Gateway IoT. Además, se utiliza un entorno de Cloud Computing con las tecnologías de Amazon Web Services (AWS), el cual recibe información de un dispositivo IoT mediante el protocolo de comunicación MQTT y permite almacenarla en una Base de Datos para ofrecerá a través de una API REST. Finalmente, la solución desarrollada en[58] cuenta con una aplicación web desarrollada en Angular con la que el usuario pueda obtener información de un determinado terrario y pueda interactuar con este. En la Figura 20 se ilustra la arquitectura propuesta para el desarrollo del sistema IoT para el control y monitorización de un terrario en [58].

[18] Instalación en la cual se mantienen artificialmente las condiciones de hábitat adecuadas para ciertos animales de tierra, especialmente reptiles y anfibios, o plantas.

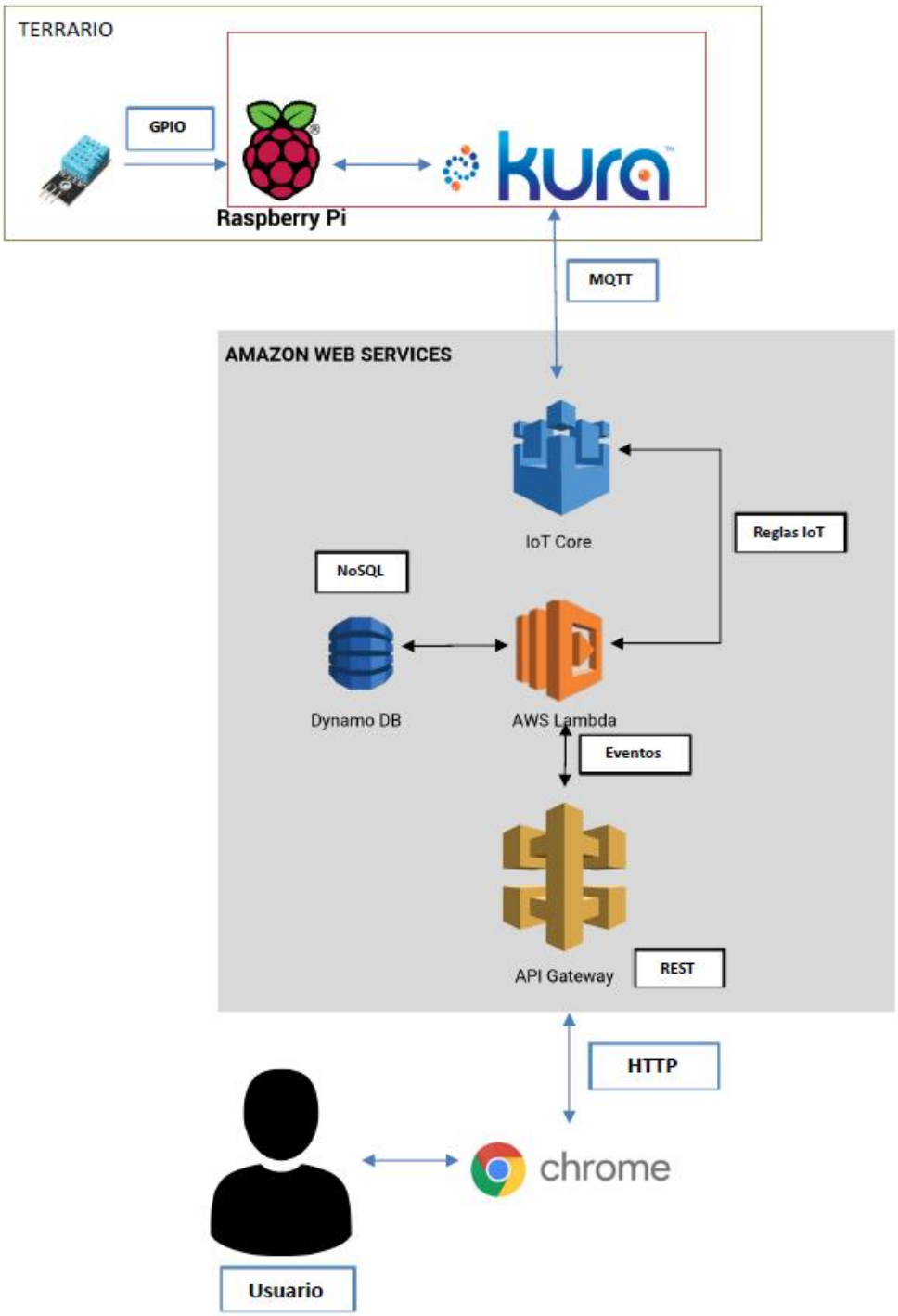

Figura 20. Arquitectura Sistema IoT para el control y monitorización de un terrario con Eclipse Kura, Amazon AWS y Angular.

Fuente: J. Jiménez Casas, "Sistema IoT para el control y monitorización de un terrario con Eclipse Kura, Amazon AWS y Angular," Universidad de Sevilla, Departamento de Ingeniería Telemática, 2019.

Otros trabajos como [59] proponen arquitecturas enfocadas en satisfacer requerimientos de seguridad para aplicaciones en sistemas Gateway para IoT. La Figura 21 muestra la arquitectura propuesta en [59].

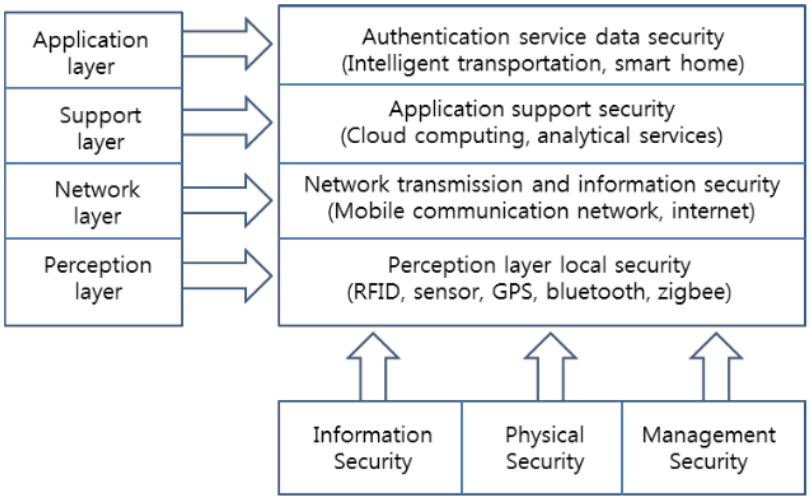

Figura 21. Arquitectura de seguridad de IoT.

Fuente: J. Kim, "Requirement of Security for IoT Application based on Gateway System," Int. J. Secur. Its Appl., vol. 9, pp. 201–208, Oct. 2015.

[60] plantean un modelo de arquitectura y su evaluación en la implementación de Gateway inteligentes para IoT. Por medio de la gestión de tareas rutinarias de aplicaciones de IoT, en este trabajo desarrolla una arquitectura que permite inteligencia descentralizada y computación distribuida, lo cual introduce una carga computacional baja al dispositivo Gateway.

En [61] se presenta un mecanismo para la interacción semántica de dispositivos IoT, el cual, dota a cada dispositivo de la capacidad de comunicarse e interactuar unos con otros sin intervención humana. Este mecanismo cuenta con tres capas en su arquitectura, es llamado IoT-Book y su uso es en escenarios de automatización para el hogar (Smart Home). La Figura 22 muestra la arquitectura del mecanismo IoT-Book.

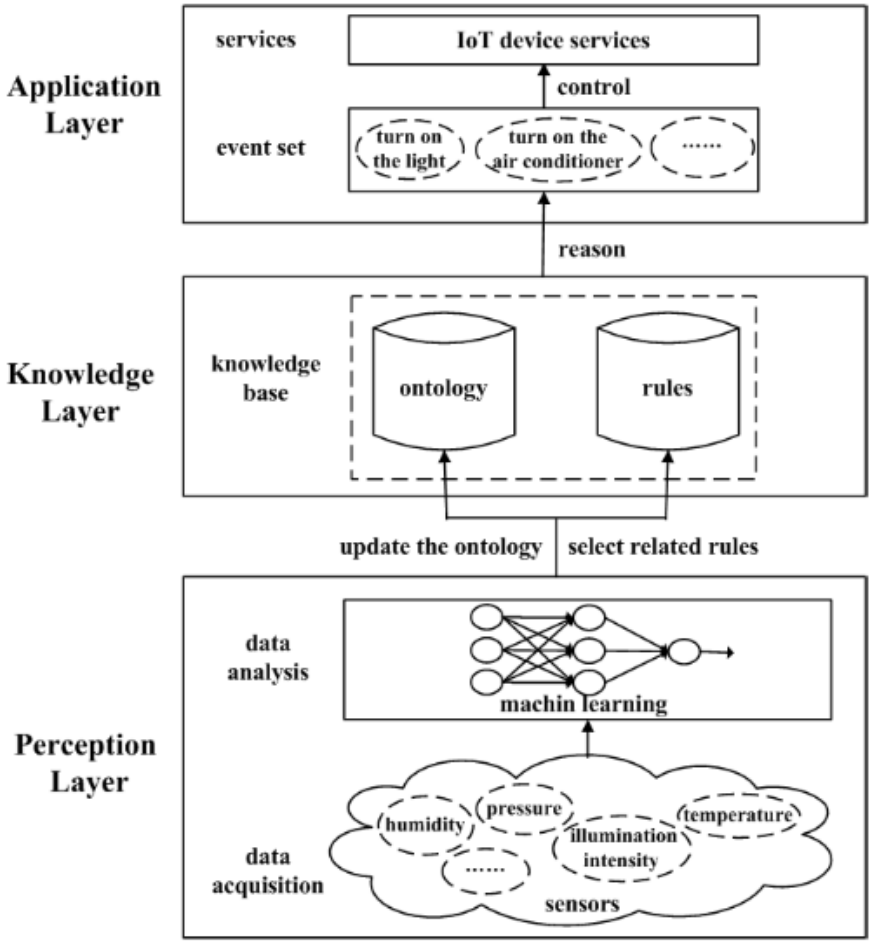

Figura 22. Arquitectura Mecanismo IoT-Book.

Fuente: S. You, X. Li, and W. Chen, "A semantic mechanism for Internet-of-Things (IoT) to implement intelligent interactions," in 2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOpt 2018, 2018.

La **Capa de Percepción** contiene dos partes, adquisición de datos y análisis de datos. La primera, es una red de sensores capturando datos del ambiente o escenario, tales como temperatura o intensidad de iluminación. La parte de análisis de datos realiza el proceso de análisis de los mismos en el mecanismo de interacción semántica, transformando los datos de sensores en diferentes tipos de estados del ambiente o escenario para posteriormente, transformar estos en información personalizada para los usuarios a través de técnicas de aprendizaje automático.

La **Capa de Aprendizaje** se basa en la toma de decisiones dentro del mecanismo de interacción semántica. En ella, se implementa una ontología y un conjunto de reglas de inferencia en la base de conocimiento. La ontología involucra definiciones de conceptos básicos y relaciones interpretables por los dispositivos organizadas en una estructura jerárquica y las reglas se manifiestan como un conjunto de condicionales "si-entonces" para proporcionar un razonamiento lógico. Los estados del entorno obtenidos de la capa de percepción se utilizan para actualizar el modelo de ontología.

La **Capa de Aplicación** consta de una serie de eventos de control obtenidos en la capa de aprendizaje, los cuales, combinados son servicios IoT realizan el proceso de control de los dispositivos en el mecanismo de interacción semántica. En el mecanismo desarrollado en [61], cada evento está vinculado a un servicio web específico, y cada servicio está vinculado a un actuador para controlarlo. Los actuadores se accionarán según los eventos obtenidos y de esta manera, el entorno se ajusta hasta satisfacer las necesidades personalizadas de los usuarios.

Un Smart IoT Gateway propuesto en [8] cumple cuatro funciones principales: Transformación de Datos, Procesamiento de Datos, Conversión de Protocolos y almacenamiento de los diferentes datos de los dispositivos. Los nodos sensores pueden establecer conexión con el Smart IoT Gateway a través de protocolos de bajo nivel como WiFi, Bluetooth, ZigBee y tradicionales como Ethernet. La Figura 23 describe la arquitectura de alto nivel del Smart IoT Gateway.

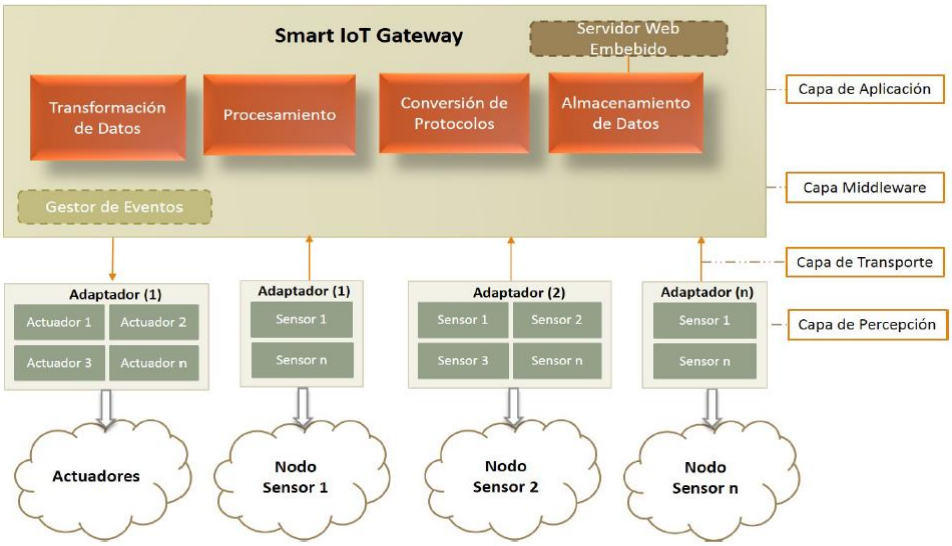

Figura 23. Arquitectura de alto Nivel de un Smart IoT Gateway.

Fuente: D. C. Yacchirema Vargas and C. E. Palau Salvador, "Smart IoT Gateway For Heterogeneous Devices Interoperability," IEEE Lat. Am. Trans., vol. 14, no. 8, pp. 3900–3906, Aug. 2016.

Este Smart IoT Gateway permite la interoperabilidad de dispositivos heterogéneos a nivel de dispositivo, protocolos y datos, mediante la incorporación de las funcionalidades de conversión de protocolos de comunicación, transformación, procesamiento y almacenamiento de datos. Dentro de los diferentes protocolos y tecnologías de comunicación tradicional e inalámbrica que permite este Gateway están Ethernet, ZigBee, Bluetooth y WiFi. El Smart IoT Gateway además de utilizar protocolos flexibles para traducir todos los datos obtenidos de los sensores en un formato uniforme, también utiliza un protocolo abierto, ligero y óptimo para el intercambio de datos entre los dispositivos con pocos recursos. En este trabajo [8] se describe un estudio de caso del Smart IoT Gateway aplicado al dominio de la sociedad, específicamente al envejecimiento activo y saludable de las personas mayores.

Otro trabajo de un IoT Home Gateway propuesto en [62] describe una arquitectura funcional que ilustra la forma en que se realiza la gestión de datos, las conexiones e interfaces de dispositivos y los protocolos soportados para acceder a dispositivos IP (Ethernet, WiFi, etc.)

y dispositivos que no son IP (ZigBee, Bluetooth, etc.) en un Gateway IoT. En la Figura 24 se ilustra la arquitectura funcional del Gateway IoT para el hogar propuesto en [62].

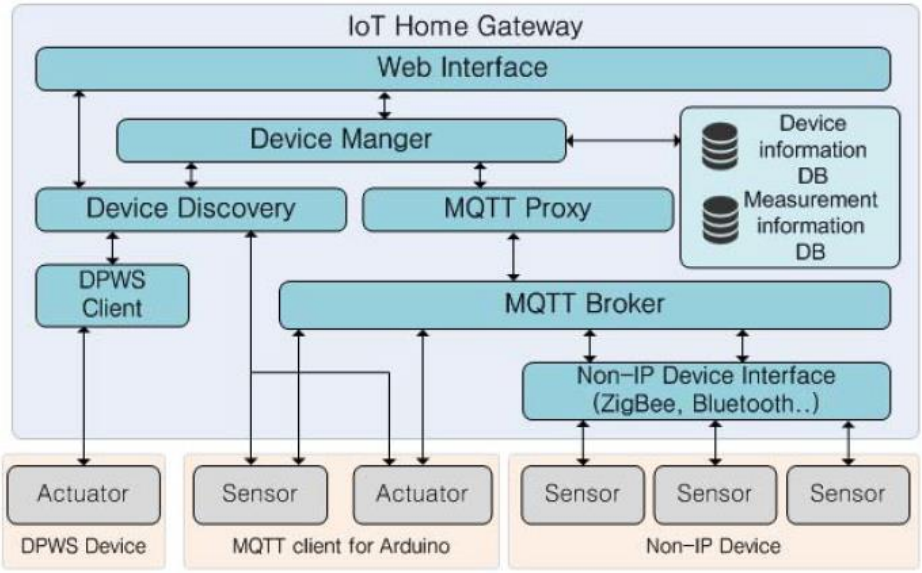

Figura 24. Arquitectura funcional Gateway IoT para el hogar.

Fuente: S.-M. Kim, H.-S. Choi, and W.-S. Rhee, "IoT home gateway for auto-configuration and management of MQTT devices," in 2015 IEEE Conference on Wireless Sensors (ICWiSe), 2015, pp. 12–17.

Por otra parte, en [5] se presenta la implementación práctica de un Gateway IoT dedicado al monitoreo en tiempo real y control remoto de redes de sensores de una piscina a través de una plataforma en línea. El Gateway IoT está basado en la placa de desarrollo Raspberry Pi, permite la comunicación bidireccional y el intercambio de datos entre el usuario y la red de sensores implementada usando una tarjeta Arduino. La Figura 25 presenta una arquitectura de alto nivel del sistema.

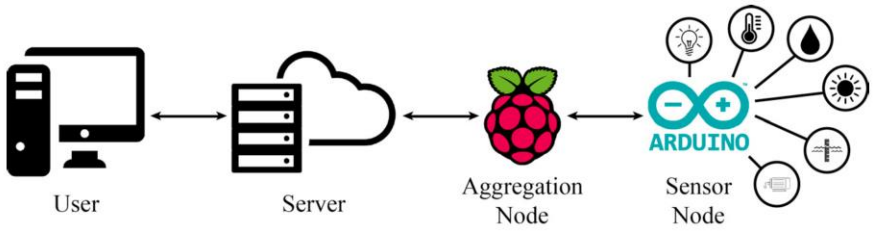

Figura 25. Arquitectura de alto nivel del sistema.

Fuente: A. Glória, F. Cercas, and N. Souto, "Design and implementation of an IoT gateway to create smart environments," Procedia Comput. Sci., vol. 109, pp. 568–575, Jan. 2017.

El propósito del sistema propuesto en [5] es usar hardware suficientemente flexible y capaz de soportar múltiples protocolos de comunicación. El componente software de este sistema se divide en dos partes, por un lado, la plataforma web consiste en una interfaz para la visualización de la información recopilada de la red de sensores; también se incluye la base de datos MySQL hospedada en un servidor privado. La otra parte del componente software son los Scripts Python presentes en el nodo de agregación (Figura 25), estos son responsables de recibir los datos del nodo sensor y enviarlos al servidor usando el protocolo

MQTT. Este sistema soporta y utiliza protocolos de comunicación como I2C, RS485, ZigBee o LoRaWAN para conectar los dispositivos sensores con el nodo sensor.
Otras contribuciones proponen Gateways para dominios de aplicación específicos, es el caso de los autores en [45], que describen un Gateway denominado “Smart e-Health” que actúa como una capa intermedia entre la red de sensores inalámbricos (WSN) e Internet, ofreciendo servicios de alto nivel como: procesamiento de datos central, almacenamiento local (temporal) y minería de datos para el despliegue de aplicaciones de monitoreo de la Salud. La propuesta en [45] apunta a establecer interoperabilidad entre las diferentes redes heterogéneas (WiFi, Bluetooth y 6LoWPAN) y los protocolos subyacentes mediante la utilización de WebSockets.

También existen propuestas de código abierto para la construcción de Gateways IoT; es el caso de Eclipse Kura [63], un proyecto Eclipse IoT que proporciona una plataforma para construir Gateways IoT. Es un contenedor de aplicaciones inteligentes que permite la administración remota de dichos Gateways y proporciona una amplia gama de APIs que facilitan el despliegue de una aplicación IoT específica.

Un estudio experimental de un Gateway Inteligente hecho en [4] describe la implementación de una puerta de enlace para IoT utilizando el framework y modelo de interacción IoTivity, es el cual es similar al modelo Cliente-Servidor de HTTP y utiliza el protocolo CoAP para las comunicaciones de dispositivo a dispositivo. Una operación IoTivity es equivalen a la de HTTP y en enviada por un cliente para solicitar una acción sobre el recurso (identificado por un URI) en el servidor. El servidor envía una respuesta con un código de respuesta. Para las pruebas se incluyeron tres tipos principales de dispositivos: el servidor maestro, los servidores esclavos y el dispositivo móvil de un usuario. Los tres dispositivos se basan en un dispositivo Raspberry Pi usando un sistema operativo Linux, lo que facilita que todos los dispositivos sean fácilmente reconfigurables para evaluar los diversos entornos de red. La puerta de enlace auto-configurable desarrollada en este proyecto fue diseñada para un contexto de ambientes en el hogar a gran escala y, básicamente los resultados de funcionamiento se evalúan midiendo el tiempo de registro de cada dispositivo en los diferentes entornos de red propuestos.

De igual forma, en la literatura encontrada, también existen proyectos donde se propone el desarrollo de estaciones para el monitoreo de variables climatológicas [64]–[66] haciendo uso de muchas de las tecnologías mencionadas anteriormente.

Aunque los trabajos citados anteriormente respecto a la implementación de Gateways IoT [4], [5], [8], [45] y estaciones para el monitoreo de variables climatológicas [64]–[66] relacionan muchos de las aspectos y tecnologías a trabajar en la actual investigación, en las investigaciones consultadas, no se ha encontrado un caso de la implementación de un Gateway Inteligente IoT con capacidades de conectar semánticamente dispositivos heterogéneos a través de protocolos Ethernet, WiFi, ZigBee y tecnologías de comunicación que permitan enriquecer la capacidad de ajustar los servicios que ofrecen los dispositivos conectados al Gateway de acuerdo al contexto de uso.

Finalmente y refiriendo el estudio de caso planteado para este trabajo de grado, cabe resaltar el desarrollo realizado en [67] de una estación climatológica basada en IoT implementada con una Raspberry Pi 3 que actúa como gateway entre los sensores de la estación y la plataforma en la nube donde se almacenan y procesan los datos. En este trabajo, se diseñó una estación climatológica que emite notificaciones en tiempo real,

permite el monitoreo de variables del clima, cuenta con una interfaz basada en una plataforma en la nube y realiza análisis de parámetros climatológicos. Los datos recolectados por los sensores de la estación climatológica conectada a una placa Arduino Uno son almacenados en la plataforma Google Cloud a través de la Raspberry Pi. La Figura 26 muestra la arquitectura del proyecto desarrollado en [67].

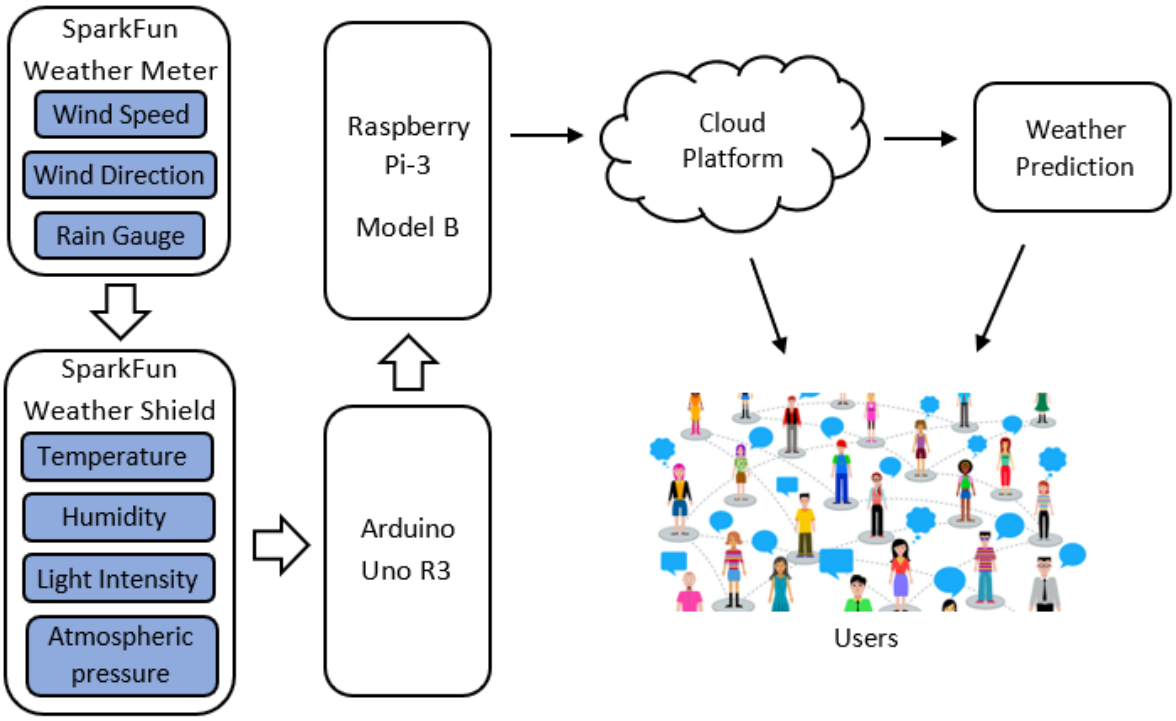

Figura 26. Arquitectura del proyecto de estación climatológica basada en IoT.

Fuente: P. Y. Muck and M. J. Homam, "IoT Based Weather Station Using Raspberry Pi 3," Int. J. Eng. Technol., vol. 7, no. 4.30, p. 145, Nov. 2018.

Los usuarios pueden visualizar las variables (temperatura, humedad relativa, precipitación, presión atmosférica, luminosidad, velocidad y dirección del viento) registradas por la estación climatológica en la plataforma Google Cloud por medio de un sitio web y una aplicación móvil, los cuales soportan funciones de monitoreo, registro de usuarios, diagnóstico de las condiciones del clima y análisis de datos. Finalmente, la estación desarrollada realiza envío de predicciones del clima vía notificaciones a los usuarios por medio de una red social, recordándoles que el clima podría tener condiciones diferentes a las actuales.

CAPÍTULO 3 DISEÑO GATEWAY INTELIGENTE IoT

En este tercer capítulo se presenta el diseño por medio de una arquitectura propuesta para el desarrollo del Gateway Inteligente IoT, la cual, fue creada teniendo en cuenta las siguientes características y funcionalidades contenidas en los objetivos de este proyecto:

- Implementa la arquitectura y concepto de Objeto Inteligente desarrollado por Niño-Zambrano [9].
- Permite incorporar recursos IoT y soporta conectividad con diferentes tipos de dispositivos a través de los protocolos Ethernet, Wifi y ZigBee.
- Soporta la reutilización de estos protocolos y sus dispositivos conectados en escenarios de interacción de objetos inteligentes con ontologías específicas para diferentes contextos de aplicación.
- Incorpora la integración de dispositivos sensores para el monitoreo de variables agroclimatológicas, como son: temperatura, humedad relativa, precipitación (cantidad de lluvia), presión atmosférica/altitud, luminosidad, velocidad y dirección del viento; variables que constituyen los principales factores ambientales del clima en la producción agrícola [10], [11] y que se consideraron dentro del estudio de caso del presente proyecto.

3.1. ARQUITECTURA GATEWAY INTELIGENTE IoT

Para la construcción de la arquitectura del Gateway Inteligente IoT, se optó por tomar como principal referencia la recomendación UIT-T Y.4101/Y.2067 “Common requirements and capabilities of a gateway for Internet of things applications” [48], emitida por el Sector de Normalización de las Telecomunicaciones de la Unión Internacional de las Telecomunicaciones (ITU), al igual que el reporte técnico [46] del Servicio de Información Comunitario sobre Investigación y Desarrollo (CORDIS)[19] “Final architectural reference model for the IoT v3.0”, del proyecto “The Internet of Things Architecture IoT-A” de la Comisión Europea. La recomendación de la ITU, proporciona los requisitos y capacidades comunes de un gateway para las aplicaciones de Internet de las cosas (IoT), misma que se refiere en particular a los gateways como dispositivos de interconexión de equipos con redes de comunicación. Adicionalmente, también fueron consultados y tomados como referencia otros trabajos [42], [46], [55]–[57], [68]–[71] sobre el diseño, desarrollo e implementación de gateways para IoT en la construcción de la arquitectura propuesta.

En las aplicaciones de IoT, la información, tanto en el mundo físico como en el mundo de la información, es recogida por dispositivos y recibida a través de redes de comunicación. Algunos dispositivos no pueden conectarse directamente a las redes de comunicación. Los gateways soportan la interconexión de esos dispositivos con las redes de comunicación.

En la Figura 27 se muestra la arquitectura propuesta para la implementación del Gateway Inteligente IoT que se contempla en el presente trabajo.

[19] https://cordis.europa.eu/about/es, consultado 23/10/2020. El Servicio de Información para la Comunidad de Investigación y Desarrollo (CORDIS, del inglés Community Research and Development Information Service), es un espacio de información dedicado a las actividades europeas de investigación y desarrollo (I+D), que tiene como objetivos, promover la difusión del conocimiento con el fin de estimular el rendimiento de las empresas en materia de innovación, sobre todo a través de la publicación de resultados de investigación obtenidos en los programas financiados por la Unión Europea.

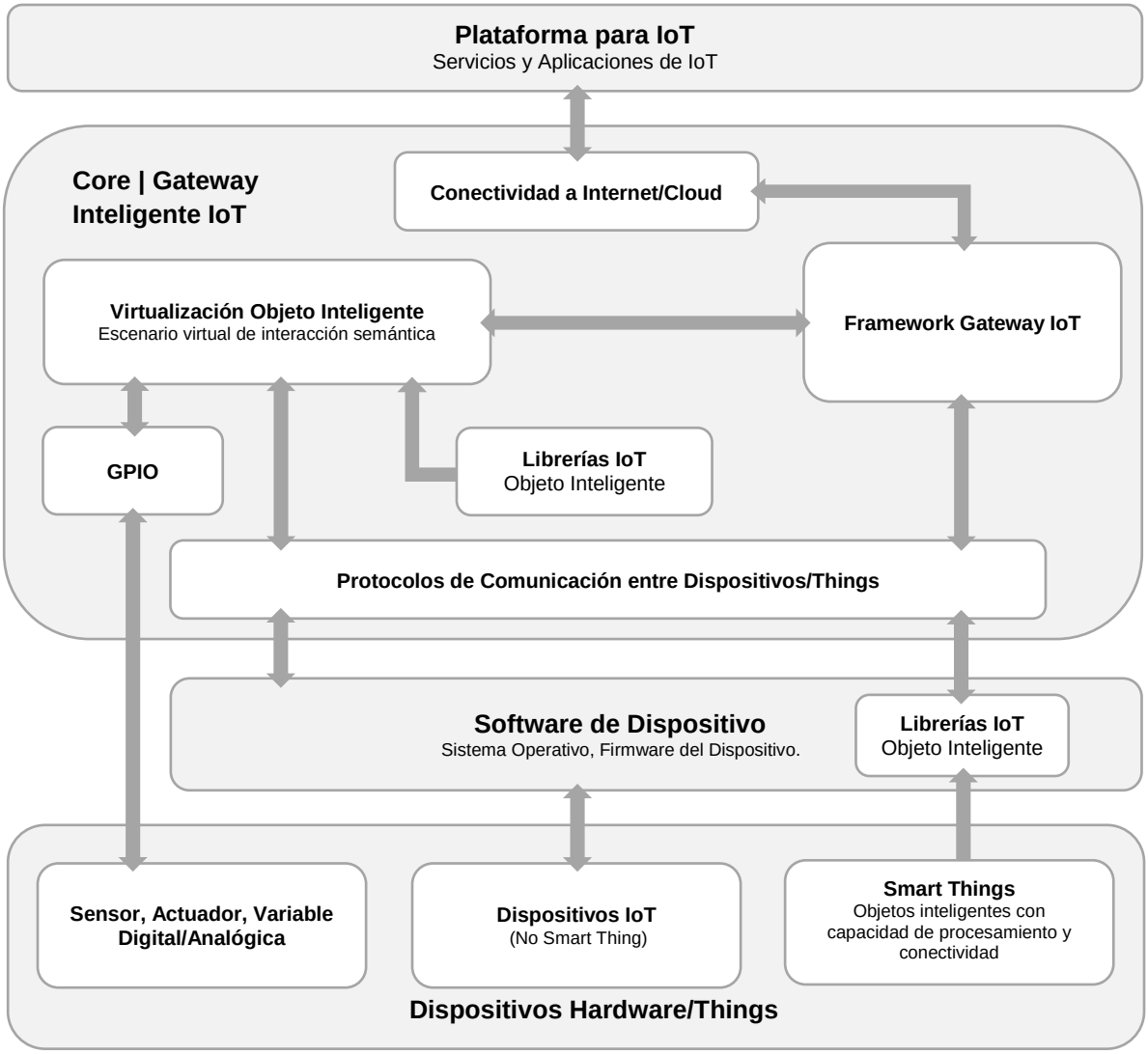

Figura 27. Arquitectura Gateway Inteligente IoT.

Fuente: Elaboración propia.

A continuación, se instanciarán cada una de las capas y módulos que componen la arquitectura propuesta:

3.1.1. Capa de Dispositivos Hardware/Things

La capa de dispositivo, también conocida como Capa de Percepción (como la llamaremos de ahora en adelante en este documento), es la que provee al Gateway Inteligente IoT la capacidad de conectarse con dispositivos del mundo físico, como lo son: diferentes tipos de sensores y actuadores.

3.1.1.1. Smart Things: Son dispositivos configurados como objetos inteligentes y que poseen capacidades de procesamiento y conectividad. Corren internamente un sistema operativo o firmware sobre el cual se pueden instalar Frameworks,

librerías o módulos de software, los cuales implementan servicios sobre el propio dispositivo. Estos Smart Things implementan diferentes protocolos y tecnologías para la comunicación con el Gateway Inteligente IoT, tales como Ethernet, Wifi y Zigbee.

3.1.1.2. Dispositivos IoT: Son dispositivos con capacidad de procesamiento y comunicación pero que no implementan la arquitectura de objeto inteligente [9], es decir, son No Smart Things. Generalmente corren un firmware en lugar de sistema operativo. En éste módulo del Gateway pueden ser conectados dispositivos microcontroladores, sistemas embebidos y/o placas de desarrollo electrónico que implementen algún código que gestione su funcionamiento y operación.

3.1.1.3. Sensor, Actuador, Variable Digital/Analógica: Es el módulo de la capa de percepción del Gateway que permite la conexión directa de un sensor, actuador o variable digital/analógica con el Gateway. A este módulo se conectan dispositivos que no tienen capacidad de conectividad y tampoco implementan ningún tipo de firmware ni sistema operativo que les permita proveer algún servicio o funcionalidad adicional para la que fueron físicamente construidos. Pueden ser conectados aquí: sensores de humedad, temperatura, interruptor digital on/off, etc., actuadores o la salida digital/analógica de otro sistema o del mismo Gateway.

3.1.2. Capa de Software de Dispositivo

Se construyó la capa de “Software de Dispositivo” para poder segmentar entre los dispositivos que son Smart Thing e implementan la arquitectura de objeto inteligente, y los dispositivos que a pesar de poseer capacidades de procesamiento y conectividad no implementan dicha arquitectura. Esto con el fin de permitir que los dispositivos Smart Things se conecten directamente al Gateway para consumir sus recursos, utilizando unos de los protocolos de comunicación definidos en este trabajo (Ethernet, Wifi, Zigbee).

3.1.2.1. Librerías IoT (Objeto Inteligente): Es el módulo que permite la implementación de la arquitectura de objeto inteligente [9] en los dispositivos con capacidad Smart Things.

3.1.3. Capa Core | Gateway Inteligente IoT

La capa Core | Gateway Inteligente IoT constituye en núcleo principal en el desarrollo de este proyecto, en ella se incorporan los protocolos Ethernet, Wifi y Zigbee, como interfaces de conexión y comunicación con los demás servicios del Gateway. También, en esta capa se cuenta con un módulo de conexiones hardware directas al Gateway que permite la virtualización de dispositivos sin capacidad de procesamiento ni conectividad por medio de librerías de objetos inteligentes. De esta forma, se facilitar la integración y gestión de estos dispositivos a través de las funcionalidades del Framework que implementa todos los servicios de una pasarela para IoT. Finalmente los servicios y funciones incorporadas en el Gateway podrán ser conectados a Internet, a otras redes o a una plataforma para IoT que permitirá el despliegue de nuevos servicios y aplicaciones de IoT.

A continuación se definen cada uno de los módulos que conforman la Capa Core | Gateway Inteligente IoT:

3.1.3.1. Protocolos de Comunicación entre Dispositivos/Things: Este módulo actúa como una capa Middleware [9], [55] entre los dispositivos de campo y el Gateway Inteligente IoT. Implementa diferentes protocolos de comunicación (Ethernet, Wifi, Zigbee, Bluetooth, etc.) y provee las interfaces para comunicar los dispositivos con el Gateway.

3.1.3.2. GPIO: Es el módulo encargado de proveer una interfaz para conexiones en pines de entradas y salidas de propósito general del Gateway con sensores, actuadores y/o variables digitales/analógicas.

3.1.3.3. Virtualización Objeto Inteligente (Escenario virtual de interacción semántica): Es el módulo que permite crear una representación virtual de los dispositivos sensores, actuadores y/o variables digitales/analógicas dentro del Gateway, de modo que se puedan implementar los servicios del escenario de interacción semántica con dispositivos que no son Smart Things (No Smart Things) y que tampoco son Dispositivos IoT.

3.1.3.4. Librerías IoT (Objeto Inteligente): Es el módulo que permite la implementación de la arquitectura de objeto inteligente [9] en los dispositivos con representación virtual de objeto inteligente (sensores, actuadores y/o variables digitales/analógicas) conectados al Gateway.

3.1.3.5. Framework Gateway IoT: implementa todas las funcionalidades, características y procesos propios de un gateway para IoT. Para la implementación de este módulo se optó por la herramienta de código abierto Eclipse Kura [63]. En la sesión 4.3 (Tecnologías para la Implementación del Gateway Inteligente IoT) del capítulo 4 del presente trabajo, trataremos más a fondo sobre Eclipse Kura.

3.1.3.6. El módulo “Plataforma para IoT (Servicios y Aplicaciones de IoT)” transporta los diferentes datos de dispositivos conectados al Gateway hacia una plataforma para IoT, la cual, permite la visualización y tratamiento de estos datos para diferentes entornos de aplicación de las variables capturadas por los sensores y actuadores conectados al Gateway.

3.1.4. Capa de Plataforma para IoT

En el más alto nivel de la arquitectura del Gateway Inteligente IoT está la Capa de Plataforma para IoT. Esta capa habilita la conexión y comunicación con plataformas para IoT, las cuales brindan la posibilidad de gestionar los recursos del Gateway, así como también, los dispositivos conectados al mismo. Dentro de las funciones más relevantes de esta capa para el estudio de caso de la agroclimatología, está la creación de nuevos servicios y aplicaciones de IoT adicionales a las generadas en el Gateway Inteligente IoT, de modo que se pueda ajustar la implementación del sistema a un contexto y necesidad específica.

CAPÍTULO 4 IMPLEMENTACIÓN GATEWAY INTELIGENTE IoT

En este cuarto capítulo, se presenta el proceso de implementación del Gateway Inteligente IoT. Al mismo tiempo, se establecen las características y requisitos que debe tener el Gateway para su funcionamiento, de acuerdo a la recomendación UIT-T Y.4101/Y.2067 de la ITU presentada en la sección 2.4.3 y 2.4.4 respectivamente del CAPÍTULO 2 del presente documento, y demás trabajos consultados. Por último, se presenta también las herramientas hardware y software, los elementos necesarios para el proceso de implementación del Gateway y la forma en que interactúan por medio de la arquitectura instanciada que se muestra en la Figura 28.

Se tuvo en cuenta las características recomendadas por la ITU, para la adopción de las tecnologías con las cuales se implementó el Gateway Inteligente IoT, mismas que sirvieron de referencia para clasificar y seleccionar las herramientas software, especialmente en lo correspondiente al Framework que implementa las funcionalidades y servicios del Gateway, así como también los componentes hardware que se usaron en el desarrollo de este proyecto. En la sesión siguiente se detallan las características generales adoptadas en la implementación del Gateway Inteligente IoT y los requisitos comunes que este debe cumplir, de acuerdo a la recomendación UIT-T Y.4101/Y.2067 de la ITU [48].

4.1. CARACTERÍSTICAS GENERALES DEL GATEWAY INTELIGENTE IoT

4.1.1. Conexión a redes de comunicación

El Gateway Inteligente IoT tiene la característica general de conectarse a redes de comunicación a través de las interfaces de red Ethernet y Wifi. Los dispositivos conectados al Gateway pueden comunicarse con otras redes a través de la conexión que éste les provee. También, el Gateway permite la conexión con otras pasarelas o puertas de enlace para permitir la interacción entre sus dispositivos conectados, y no únicamente de forma directa a redes de comunicación.

4.1.2. Acceso de dispositivo

El Gateway Inteligente IoT tiene la característica general de soportar el acceso de los dispositivos. Los dispositivos conectados al Gateway pueden establecer conexión y comunicación entre ellos mismos o con las redes de comunicación accediendo al Gateway. Diferentes tipos de tecnologías de acceso a dispositivos son soportados por el Gateway desarrollado, entre las que cabe mencionar: Wifi, Ethernet, Bluetooth, ZigBee, Modbus[20] para aplicaciones de IoT Industrial (IIoT – Industrial Internet of Things) y tecnologías de comunicación serial como USB, SPI[21] e I^2C[22], entre otras.

[20] https://modbus.org/specs.php, consultado 17/12/2020.

[21] https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all, https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html, consultado 17/12/2020.

[22] https://learn.sparkfun.com/tutorials/i2c/all, consultado 17/12/2020.

4.1.3. Traducción de protocolos

El Gateway Inteligente IoT tiene la característica general de traducir protocolos de comunicación entre dispositivos y redes de comunicación, así como también, traduce los protocolos entre diferentes dispositivos conectados a él.

4.1.4. Interacción y soporte de aplicaciones

El Gateway Inteligente IoT tiene la característica general de proporcionar interacción con las aplicaciones y soporte a las mismas, posibilita la interacción lógica entre diferentes aplicaciones y servicios por medio de la implementación de drivers de uso específico y servicios que pueden ser desplegados en el Framework principal.

4.1.5. Adaptabilidad

El Gateway Inteligente IoT tiene la característica general de tener interfaces de gestión estandarizadas e incorpora soporte de mediación semántica [9], [40], [72] de datos entre los dispositivos inteligentes que se conectan al Gateway, lo cual constituye una de las características más relevantes y principal aporte de este proyecto.

4.1.6. Soporte de las funciones de gestión

El Gateway Inteligente IoT tiene la característica general de soportar las funciones de gestión, incluida la gestión de dispositivos, la gestión de redes y conexiones establecidas en las diferentes interfaces de red, la gestión de servicios y la gestión de protocolos.

4.1.7. Soporte de las funciones de seguridad

El Gateway Inteligente IoT tiene la característica general de soportar funciones de seguridad de las redes de dispositivos conectados y proporciona mecanismos para dar soporte a los requisitos de seguridad de las aplicaciones desplegadas en él.

Dentro de los mecanismos de seguridad incorporados en el Gateway cabe mencionar, la implementación de certificados SSL para la conexión con dispositivos, servidores y plataformas para IoT, la incorporación de un Firewall para gestionar los diferentes puertos e interfaces junto con su direccionamiento IP y protocolos de comunicación, así como también, mecanismo de autenticación, autorización y gestión de identidad para acceder a las funciones y configuraciones del Gateway.

4.2. IMPLEMENTACIÓN DE LA ARQUITECTURA DEL GATEWAY INTELIGENTE IoT

Diferentes recursos hardware, software y tecnologías, fueron seleccionados para la implementación del Gateway Inteligente IoT propuesto en la arquitectura de la Figura 27. La Figura 28 muestra la implementación de la arquitectura propuesta para el desarrollo del Gateway Inteligente IoT con los recursos y tecnologías seleccionadas que permitieron dar cumplimiento a las características definidas para el Gateway y a los objetivos propuestos en este proyecto.

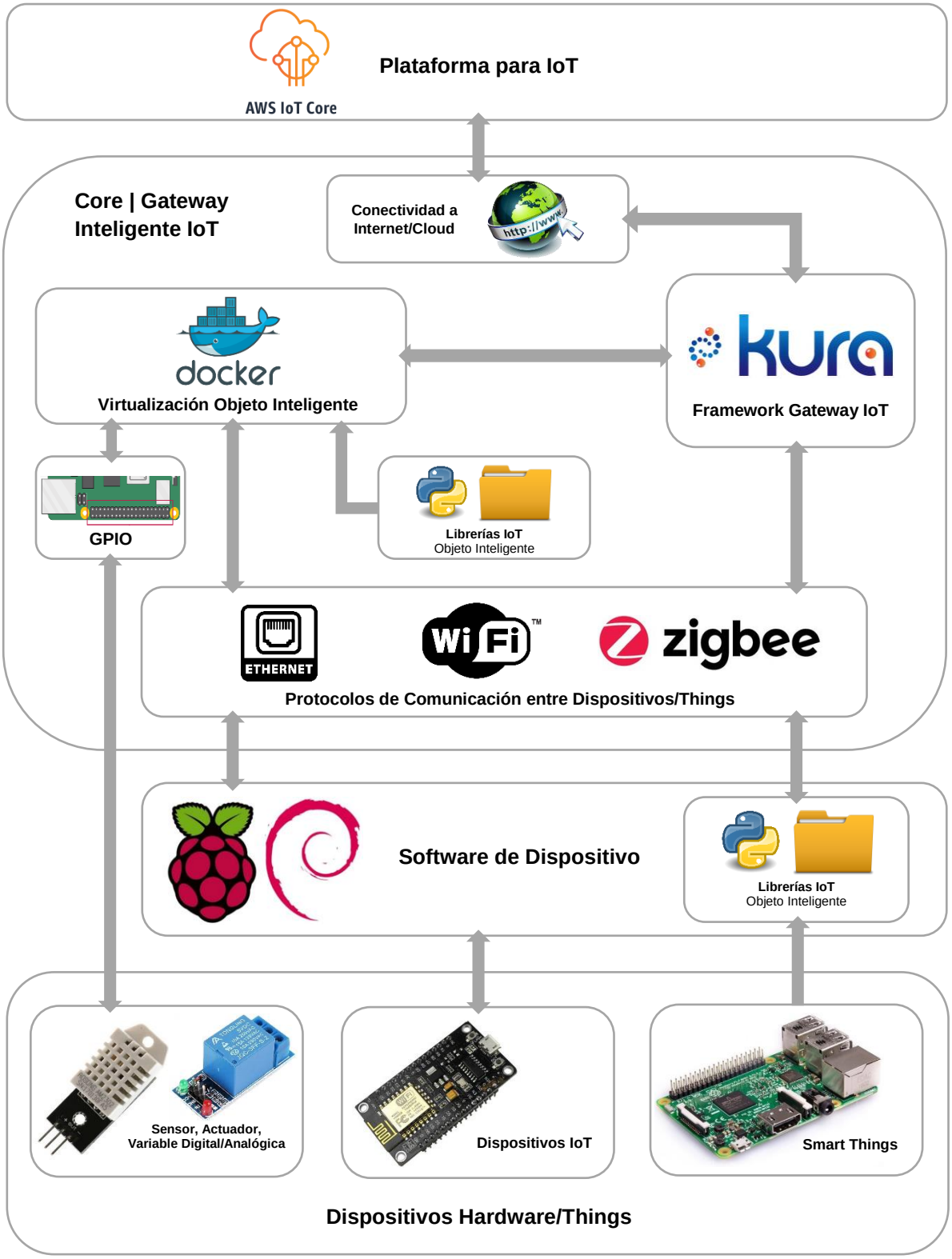

Figura 28. Arquitectura de implementación del Gateway Inteligente IoT.

Fuente: Elaboración propia.

4.3. RECURSOS UTILIZADOS

En esta sesión se presentan los recursos hardware y software utilizados para la implementación del Gateway Inteligente IoT.

4.3.1. Recursos Hardware

4.3.1.1. Raspberry Pi 3 Modelo B+

Esta es la placa principal sobre la cual se implementa el Gateway Inteligente IoT. Adicional a esto, también se usa para la implementación de un objeto inteligente conectado al Gateway. La Raspberry Pi (Figura 29) es un ordenador de placa reducida o placa única de bajo coste, desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas [73].

Figura 29. Raspberry Pi 3 Modelo B+.

Fuente: "Teach, Learn, and Make with Raspberry Pi." [Online]. Available: https://www.raspberrypi.org/. [Accessed: 15-Dec-2020].

La Tabla 2 contiene las especificaciones técnicas oficiales de la placa Raspberry Pi B+.

Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
1GB LPDDR2 SDRAM
2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
Extended 40-pin GPIO header
Full-size HDMI
4 USB 2.0 ports
CSI camera port for connecting a Raspberry Pi camera
DSI display port for connecting a Raspberry Pi touchscreen display
Micro SD port for loading your operating system and storing data
5V/2.5A DC power input

Tabla 2. Especificaciones técnicas placa Raspberry Pi 3 Modelo B+.

Fuente: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/, consultado 04/12/2020.

4.3.1.2. Arduino Uno Rev3

El Arduino Uno Rev3 es una placa de desarrollo para programación y prototipado de electrónica, basada en el microcontrolador ATmega328P [74]. Cuenta con 14 pines de entrada/salida digital (de los cuales 6 pueden ser usados como salidas PWM) y 6 entradas análogas. En la Figura 30 se muestra la placa Arduino Uno Rev3.

Figura 30. Arduino Uno Rev3.

Fuente: https://store.arduino.cc/usa/arduino-uno-rev3, consultado 30/11/2020.

El Arduino Uno cuenta con una amplia variedad de hardware complementario que se puede integrar fácilmente con la placa para dotarla de funcionalidades adicionales, como shields para conectividad de red Ethernet, Wifi, ZigBee y Bluetooth, comunicación con redes móviles celulares, control de motores y conexión de sensores, entre otros.

La Tabla 3 contiene las especificaciones técnicas oficiales de la placa Arduino Uno Rev3.

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13

Tabla 3. Especificaciones técnicas placa Arduino Uno Rev3.

Fuente: https://store.arduino.cc/usa/arduino-uno-rev3, consultado 30/11/2020.

4.3.1.3. ESP32 NodeMCU V3

El ESP32 NodeMCU es un sistema de bajo consumo y bajo costo en un chip SoC (System On Chip) con conectividad Wi-Fi y Bluetooth. La Figura 31 muestra la placa ESP32 NodeMCU versión 3.

Figura 31. ESP32 NodeMCU V3.

Fuente: https://www.nodemcu.com/index_en.html, consultado 29/11/2020.

La tarjeta NodeMCU-32 está diseñada con un regulador de voltaje que le permite alimentarse directamente del puerto USB haciendo que los pines de entradas/salidas trabajen a 3.3V. Adicionalmente cuenta el chip CP2102 que se encarga de la comunicación USB-Serial. Para su desarrollo cuenta con gran variedad de software, lenguajes de programación, frameworks, librerías, código/ejemplos y otros recursos. Los más comunes de usar son: Esp-idf (Espressif IoT Development Framework) desarrollado por el fabricante del chip, Arduino (en lenguaje C++), Simba Embedded Programming Platform (en lenguaje Python), RTOS's (como Zephyr Project, Mongoose OS, NuttX RTOS), MicroPython, LUA y Javascript [75].

La Tabla 4 contiene las especificaciones técnicas oficiales de la placa Arduino Uno Rev3.

Module with build in ESP-12E with PCB antenna
WiFi connectivity: 802.11 b/g/n
Modes: (Access Point), STA (Standalone), AP+STA
Supports: TKIP, WEP, CRC, CCMP, WPA/WPA2, WPS
Supply voltage: 3.3V (or 5V via micro USB port)
CPU: RISC 80MHz (overclockable to 160MHz)
10 GPIO - PWM / I2C / SPI / 1-Wire
Max current on I/O pins: 12mA
USB-UART converter - CH340
Build in ADC - 10-bitowy
30 pins in 2,54mm raster
micro USB socket

Tabla 4. Especificaciones técnicas placa ESP32 NodeMCU V3.

Fuente: https://nettigo.eu/products/wifi-module-nodemcu-v3-lolin-with-esp-12e-tested, consultado 04/12/2020.

4.3.1.4. Base Grove Shield para NodeMCU

La Base Grove Shield para NodeMCU es una tarjeta de extensión que facilita la conexión de sensores y actuadores Grove con el kit de desarrollo ESP32 NodeMCU V3. La Figura 32 muestra la base Grove con la cual se conectó el ESP32.

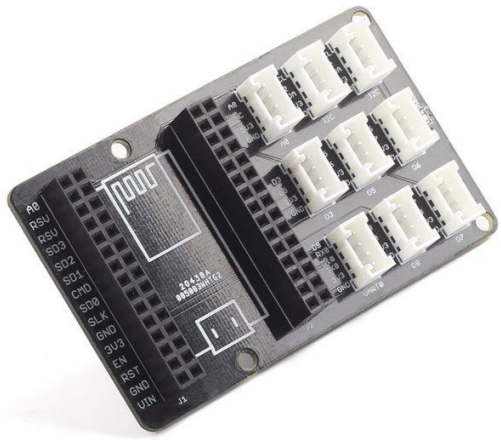

Figura 32. Base Grove Shield para NodeMCU

Fuente: https://www.seeedstudio.com/Grove-Base-Shield-for-NodeMCU.html, consultado 29/11/2020.

La Tabla 5 muestra las especificaciones técnicas de la Base Grove Shield.

Compatible with all Grove modules
5 Digital connectors (D3-D8)
1 Analog connectors (A0)
2 I2C sockets
NO SPI socket
UART/D9-D10 connector
Power indicator LED

Tabla 5. Especificaciones técnicas de la Base Grove Shield para NodeMCU.

Fuente: https://www.seeedstudio.com/Grove-Base-Shield-for-NodeMCU.html, consultado 04/12/2020.

4.3.1.5. XBee-PRO S2B

Los módulos XBee son dispositivos utilizados para enviar y recibir datos de forma inalámbrica valiéndose del protocolo IEEE 802.15.4, los cuales pueden funcionar dentro de una topología de red de diferentes maneras, incluyendo el Coordinador, Router y End Device [76]. La Figura 33 muestra el Módulo XBee-PRO S2B del fabricante Digi International Inc[23].

[23] https://www.digi.com/, consultado 04/12/2020. Módulo Xbee-PRO S2B fabricante Digi International Inc.

Figura 33. Módulo XBee-PRO S2B.

Fuente: "Digi XBee Ecosystem | Everything You Need to Explore and Create Wireless Connectivity | Digi International." [Online]. Available: https://www.digi.com/xbee. [Accessed: 01-Feb-2021].

El XBee-PRO S2B permite comunicaciones de larga distancia desde 300 ft (90 metros) en escenarios de interiores, hasta 5000 ft (1500 metros) en exteriores con linea de vista entre los módulos que genran el enlace punto a punto. Ademas, este módulo cuenta con 10 pines de entreda/salida de propósito general, 4 entredas analógicas con un conversor analógico-digital (ADC) de 10 bits de resolución. La tasa de datos es de 1200 bps (1.2Kbps) hasta 1Mbps operando en la banda de frecuencia de 2.4 GHz [76].

La Tabla 6 contiene las especificaciones técnicas de los módulos de radio frecuencia XBee2, XBee-PRO (S2) y el módulo Xbee-PRO (S2B) utilizado en el desarrollo del presente proyecto.

Specification	XBee	XBee-PRO (S2)	XBee-PRO (S2B)
Performance			
Indoor/urban range	up to 133 ft. (40 m)	Up to 300 ft. (90 m), up to 200 ft (60 m) international variant	Up to 300 ft. (90 m), up to 200 ft (60 m) international variant
Outdoor RF line-of-sight range	up to 400 ft. (120 m)	Up to 2 miles (3200 m), up to 5000 ft (1500 m) international variant	Up to 2 miles (3200 m), up to 5000 ft (1500 m) international variant
Transmit power output	2 mW (+3dBm), boost mode enabled 1.25 mW (+1dBm), boost mode disabled	50 mW (+17 dBm) 10 mW (+10 dBm) for International variant	63mW (+18 dBm) 10mW (+10 dBm) for International variant
RF data rate	250,000 b/s	250,000 b/s	250,000 b/s
Data throughput	up to 35000 b/s (see Transmission, addressing, and routing)	up to 35000 b/s (see Transmission, addressing, and routing)	up to 35000 b/s (see Transmission, addressing, and routing)
Serial interface data rate (software selectable)	1200 b/s - 1 Mb/s (non-standard baud rates also supported)	1200 b/s - 1 Mb/s (non-standard baud rates also supported)	1200 b/s - 1 Mb/s (non-standard baud rates also supported)
Receiver sensitivity	-96 dBm, boost mode enabled -95 dBm, boost mode disabled	-102 dBm	-102 dBm
Power Requirements			
Supply voltage	2.1 - 3.6 V	3.0 - 3.4 V	2.7 - 3.6 V
Operating current (transmit, max output power)	40 mA (@ 3.3 V, boost mode enabled) 35 mA (@ 3.3 V, boost mode disabled)	295 mA (@3.3 V) 170 mA (@3.3 V) international variant	205 mA, up to 220 mA with programmable variant (@3.3 V) 117 mA, up to 132 mA with programmable variant (@3.3 V), International variant
Operating current (receive))	40mA (@ 3.3 V, boost mode enabled) 38mA (@ 3.3 V, boost mode disabled)	45 mA (@3.3 V)	47 mA, up to 62 mA with programmable variant (@3.3 V)
Idle current (receiver off)	15mA	15mA	15mA
General			
Operating frequency band	ISM 2.4 GHz	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.960” x 1.087” (2.438cm x 2.761cm)	0.960 x 1.297 (2.438 cm x 3.294 cm)	0.960 x 1.297 (2.438 cm x 3.294 cm)
Operating temperature	-40 to 85° C (industrial)	-40 to 85° C (industrial)	-40 to 85° C (industrial)
Antenna options	Integrated whip antenna, embedded PCB antenna, RPSMA, or U.FL connector	Integrated whip antenna, embedded PCB antenna, RPSMA or U.FL connector	Integrated whip antenna, embedded PCB antenna, RPSMA or U.FL connector
I/O interface	3.3 V CMOS UART (not 5 V tolerant), DIO, ADC	3.3 V CMOS UART (not 5 V tolerant), DIO, ADC	3.3 V CMOS UART (not 5 V tolerant), DIO, ADC

Tabla 6. Especificaciones técnicas módulos de radio frecuencia XBee/XBee-PRO.

Fuente: https://www.digi.com/resources/documentation/digidocs/pdfs/90000976.pdf, consultado 04/12/2020.

4.3.1.6. Shield XBee SparkFun

Esta tarjeta permite añadir comunicación ZigBee a las tarjetas de desarrollo Arduino. Trabaja con todos los módulos XBee tanto en versión standard como Pro. Permiten interconectar tarjetas (shields) adicionales, si están acompañados de header (conectores o pines) hembra en la parte superior. El Shield XBee SparkFun contiene LEDs, botón de reset y un área de prototipado. La Figura 34 muestra el Shield XBee de la marca SparkFun utilizado para conectar el módulo XBee-PRO S2B.

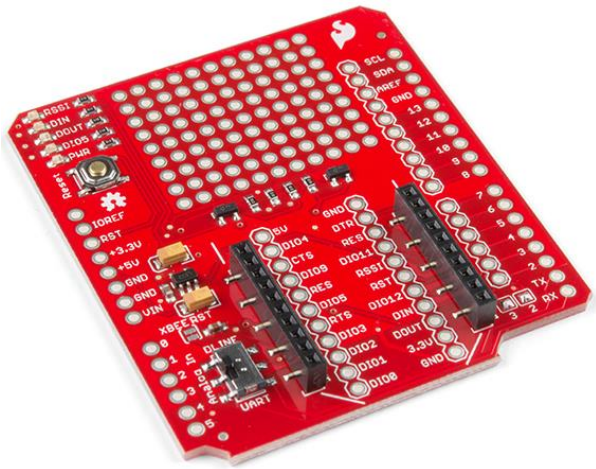

Figura 34. Shield XBee SparkFun.

Fuente: https://learn.sparkfun.com/tutorials/xbee-shield-hookup-guide/all, consultado 29/07/2020.

La Tabla 7 ilustrada a continuación relaciona las especificaciones técnicas del Shield XBee SparkFun.

Compatible con las Series 1 y 2 versiones Standar y Pro de los módulos XBee.
La tarjeta se alimenta del pin de 5V de la tarjeta Arduino.
Voltaje regulado a 3.3V para alimentar el módulo XBee.
Área de prototipado: 9x11 orificios de 0.1"
Tamaño: 54mm x 60mm
No incluye módulos XBee.
Los pines DIN y DOUT del XBee pueden conectarse a los pines UART o a pines digitales de la Arduino (D2 y D3 por defecto).

Tabla 7. Especificaciones técnicas Shield XBee SparkFun.

Fuente: https://learn.sparkfun.com/tutorials/xbee-shield-hookup-guide/all#hardware-overview, consultado 04/12/2020.

4.3.1.7. SparkFun XBee Explorer Dongle

El módulo SparkFun XBee Explorer Dongle (Figura 35) para la línea de módulos Digi XBee, permite conectar la unidad directamente a un puerto USB y hacer que actúe como una puerta de enlace entre la computadora y el XBee. Este módulo funciona con todos los módulos XBee, incluidos la Serie 1 y la Serie 2.5, versión estándar y Pro. El regulador de voltaje incorporado soporta hasta 500 mA.

Lo más destacado de esta placa es un convertidor FT231X USB a serie. Eso es lo que traduce los datos entre la computadora y el XBee. También hay un botón de reinicio y un regulador de voltaje para suministrar energía al XBee. Además, hay cuatro LED que indican el estado del módulo XBee: RX, TX, RSSI (indicador de intensidad de la señal) e indicador de encendido.

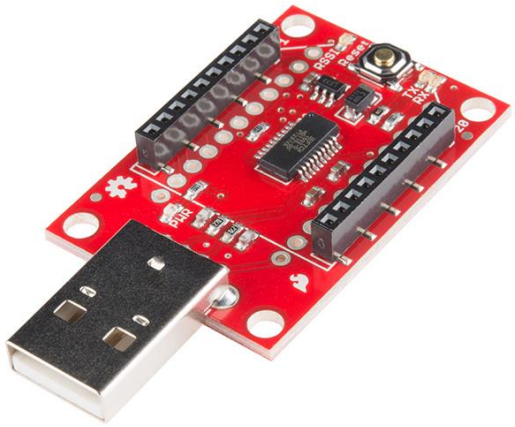

Figura 35. SparkFun XBee Explorer Dongle.

Fuente: https://www.sparkfun.com/products/11697, consultado 29/07/2020.

4.3.1.8. Estación Meteorológica Sparkfun

Para medir las variables agroclimatológicas contempladas en los objetivos de este proyecto, se empleó una estación meteorológica de la marca Sparkfun[24]. La estación meteorológica está compuesta de un shield[25] con sensores de clima incorporados a la placa y un kit de sensores[26] de viento y lluvia.

4.3.1.8.1. Shield Clima Sparkfun

El Shield Clima de la Estación Meteorológica Sparkfun, es una placa integrada que incorpora sensores de presión barométrica, humedad relativa, luminosidad, temperatura y conexiones para sensores de viento y lluvia por medio de terminales RJ-11. En cuanto a su uso, el shield clima permite una conexión rápida y fácil con la plataforma Arduino para su puesta en funcionamiento usando las librerías del fabricante. La Figura 36 muestra el Shield de Clima utilziado en este proyecto.

[24] https://www.sparkfun.com/, consultado 29/07/2020. Sitio web oficial SparkFun Electronics.
[25] https://www.sparkfun.com/products/13956, consultado 29/07/2020. Shiel clima Sparkfun.
[26] https://www.sparkfun.com/products/15901, consultado 29/07/2020. Sensores de viento y lluvia estación meteorológica Sparkfun.

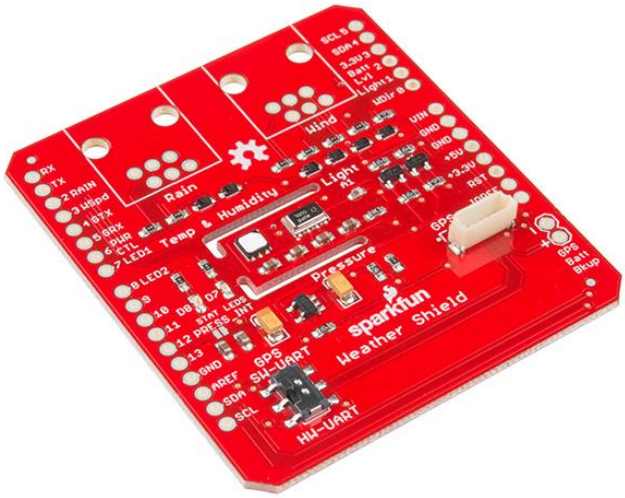

Figura 36. Shield de Clima Sparkfun.

Fuente: https://www.sparkfun.com/products/13956, consultado 29/07/2020.

La Tabla 8 muestra los sensores que conforman el shield de clima Sparkfun y las respectivas especificaciones técnicas ampliadas de cada sensor en su hoja de datos (datasheet):

Tipo de Sensor	Referencia	Ficha Técnica del Sensor
Humedad y temperatura	Si7021	https://cdn.sparkfun.com/datasheets/Sensors/Weather/Si7021.pdf
Presión barométrica	MPL3115A2	http://cdn.sparkfun.com/datasheets/Sensors/Pressure/MPL3115A2.pdf
Sensor de Luz	ALS-PT19	http://cdn.sparkfun.com/datasheets/Components/General%20IC/ALS-PT19-315C-L177-TR_datasheet.pdf

Tabla 8. Sensores Shield Clima Spakfun.

Fuente: Elaboración propia.

4.3.1.8.2. Sensores de viento y lluvia estación meteorológica Sparkfun

El kit de sensores de viento y lluvia que componen la estación meteorológica Sparkfun son:

3. **Veleta:** Permite determinar la dirección del viento en grados.
4. **Anemómetro**: Permite medir la velocidad del viento.
5. **Pluviómetro:** Permite medir la cantidad de lluvia.

La Figura 37 muestra los sensores de viento y lluvia que conforman la estación meteorológica Sparkfun.

Figura 37. Sensores de viento y lluvia estación meteorológica Sparkfun.

Fuente: https://www.sparkfun.com/products/15901, consultado 29/07/2020.

En el sitio web oficial del fabricante se puede consultar la ficha técnica[27] y la guía de ensamble[28] de los sensores de viento y lluvia de la estación meteorológica Sparkfun.

4.3.1.9. Sensor de Temperatura y Humedad DHT22 V2

El DHT22 V2 (Figura 38) es un sensor digital compuesto de un detector capacitivo para humedad y un dispositivo de medición de temperatura de alta precisión. Posee un microcontrolador de 8 bits de alto rendimiento, el cual le permite brindar una excelente calidad y una respuesta rápida de las variables censadas [77].

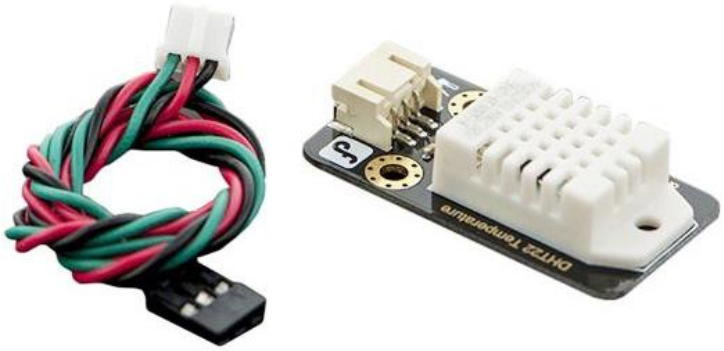

Figura 38. Sensor de Temperatura y Humedad DHT22 V2.

Fuente: "DHT22 Temperature and Humidity Sensor-DFRobot." [Online]. Available: https://www.dfrobot.com/index.php?route=product/product&product_id=1102&search=SEN0137&description=true. [Accessed: 01-Feb-2021].

[27] https://cdn.sparkfun.com/assets/d/1/e/0/6/DS-15901-Weather_Meter.pdf, consultado 29/07/2020.
[28] https://learn.sparkfun.com/tutorials/weather-meter-hookup-guide?_ga=2.180179876.1638712284.1608051624-1788102689.1608051624, consultado 29/07/2020.

4.3.1.10. Sensor de Temperatura Grove V1.2

El sensor de temperatura Grove (Figura 39) usa un termistor que detecta la temperatura ambiente. La resistencia del termistor incrementa cuando la temperatura ambiente disminuye, característica usada para calcular el valor de temperatura en un rango de -40° a 125° C con una precisión de +/- 1.5°C.

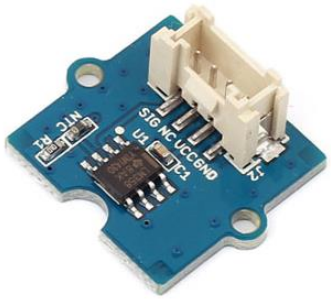

Figura 39. Sensor de temperatura Grove V1.2.

Fuente: Grove - Temperature Sensor V1.2 - Seeed Wiki (seeedstudio.com), consultado 29/07/2020.

4.3.1.11. Adaptador USB a Gigabit Ethernet Linksys USB3GIG

El adaptador USB3GIG del fabricante para equipos tecnológicos de redes y comunicaciones Linksys[29] permite agregar una nueva interfaz de red cableada Ethernet al Gateway Inteligente IoT, con la cual se obtiene una nueva conexión para dispositivos IoT gestionable desde el menú de configuraciones del Gateway. La Figura 40 muestra el adaptador Ethernet a USB Linksys.

Figura 40. Adaptador USB Gigabit Ethernet Linksys USB3GIG.

Fuente: https://www.linksys.com/co/p/P-USB3GIG/, consultado 29/11/2020.

Dentro de las especificaciones técnicas del dispositivo están, que soporta los estándares de red Estándares IEEE 802.3 e IEEE 802.3u, cuenta con un (1) puertos Gigabit Ethernet y posee una interfaz de conexión USB 3.0 con una tasa de enlace máxima de 1000 Mbps, con la cual se conecta a la placa Raspberry Pi [78].

4.3.2. Recursos Software

Una cantidad variada de recursos software fueron necesarios para el desarrollo de este proyecto. Desde aplicaciones para Windows que facilitan el alistamiento de los archivos de

[29] Linksys - Routers inalámbricos, Extensores de alcance y Cámaras IP, consultado 02/02/2020.

configuración como Win32 DiskImager[30], Raspberry Pi Imager[31], aplicaciones para el alistamiento de recursos hardware como SD Card Formatter[32], hasta aplicaciones para la gestión de conexiones y manejo de información entre los dispositivos utilizados VNC[33], Putty[34], WinSCP[35] entre otras, cabe hacer énfasis en aquellos recursos más utilizados en la configuración y despliegue del Gateway Inteligente IoT, los cuales se mencionan a continuación:

4.3.2.1. Eclipse IDE

El entorno de desarrollo integrado IDE de Eclipse se ha utilizado para probar el desarrollo de los bundles que implementa Eclipse Kura cono servicios dentro del Framework [79]. La Figura 41 muestra el logo del IDE de Eclipse.

Figura 41. Eclipse IDE.

Fuente: https://www.eclipse.org/ide/, consultado 01/02/2021.

4.3.2.2. Geany

Geany es un potente, estable y ligero editor de texto que incluye características de entorno de desarrollo integrado. Se ejecuta en Linux, Windows y MacOS, se traduce a más de 40 idiomas, y tiene soporte integrado para más de 50 lenguajes de programación [80]. La Figura 42 muestra el logo del IDE Geany.

Figura 42. Geany IDE.

Fuente: https://www.geany.org/, consultado 01/02/2021.

4.3.2.3. Pycharm

PyCharm es un entorno de desarrollo integrado (IDE) que se utiliza en programación de computadoras, específicamente para el lenguaje Python. Está desarrollado por la empresa

[30] Win32 Disk Imager download | SourceForge.net, consultado 01/02/2021.
[31] Raspberry Pi OS – Raspberry Pi, consultado 01/02/2021.
[32] SD Card Formatter 5.0.1 para Windows - Descargar (uptodown.com), consultado 01/02/2021.
[33] Download VNC Viewer | VNC® Connect (realvnc.com), consultado 01/02/2021
[34] PuTTY 0.74 (64-bit) para Windows - Descargar (uptodown.com), consultado 01/02/2021.
[35] WinSCP :: Official Site :: Free SFTP and FTP client for Windows, consultado 01/02/2021.

checa JetBrains. Proporciona análisis de código, un depurador gráfico, un probador de unidades integrado, integración con sistemas de control de versiones. PyCharm es multiplataforma, con versiones de Windows, macOS y Linux [81].

La Figura 43 muestra el logo del entorno de desarrollo integrado Pycharm.

Figura 43. Pycharm IDE.

Fuente: https://www.jetbrains.com/es-es/pycharm/, consultado 01/02/2021.

4.3.2.4. Arduino IDE

El software Arduino (IDE) de código abierto facilita la escritura de código y su carga en la placa. Este software se puede utilizar con cualquier modelo de placa Arduino [82]. La Figura 44 muestra el logo del entorno de desarrollo integrado Arduino.

Figura 44. Arduino IDE.

Fuente: https://www.arduino.cc/, consultado 01/02/2021.

4.4. TECNOLOGÍAS UTILIZADAS

En esta sección se presentan las tecnologías utilizadas para la implementación del Gateway inteligente IoT

4.4.1. Eclipse Kura

Después de una amplia revisión bibliográfica de modelos de referencia, arquitecturas, desarrollos, implementaciones y recomendaciones para desarrollo de gateways para aplicaciones de IoT [46], [48], la implementación tecnológica de la capa de Gateway Inteligente IoT, se optó por el Framework para el desarrollo de gateways para IoT Eclipse Kura [63].

Eclipse Kura tiene la finalidad de convertir un dispositivo en un Gateway IoT. En este se instalan las distintas aplicaciones haciendo uso de las API (Interfaz de programación de aplicaciones) que proporciona, permitiendo así un despliegue de forma sencilla. Además, proporciona mecanismos de conexión con distintos servicios web, gestión de

funcionalidades de enrutamiento para redes IP y pasarela de comunicaciones para aplicaciones de internet de las cosas [83].

Con Eclipse Kura se consigue estas funcionalidades, convirtiendo la placa Raspberry Pi en un Gateway IoT. Éste, es además un contenedor de aplicaciones basado en Java y OSGi [84], denominadas bundles [83], que se desarrollan haciendo uso del entorno de desarrollo Eclipse.

La Figura 45 muestra la arquitectura de servicios de Eclipse Kura. En el **ANEXO A:** del presente documento contiene la información ampliada sobre Kura y el proceso de instalación para su puesta en funcionamiento en la placa Raspberry Pi 3 modelo B V1.2.

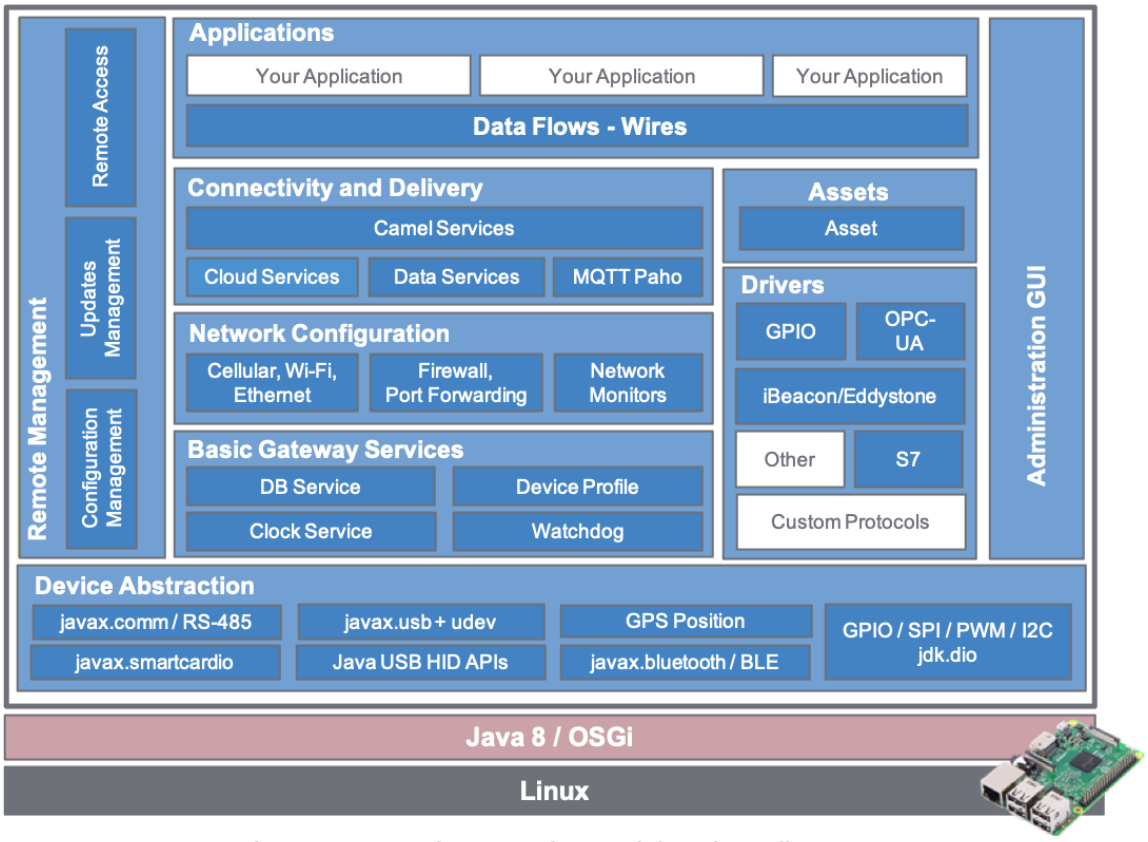

Figura 45. Arquitectura de servicios de Eclipse Kura.

E. Foundation, "Eclipse Kura - Open Source framework for IoT." [Online]. Available: http://www.eclipse.org/kura/. [Accessed: 21-May-2018].

4.4.2. AWS IoT Core

La plataforma AWS IoT Core de Amazon proporciona una comunicación bidireccional segura entre los dispositivos conectados a Internet (como sensores, actuadores, microcontroladores integrados o aparatos inteligentes) y la nube de AWS. Esto le permite recopilar datos de telemetría de varios dispositivos, y almacenar y analizar los datos. También se pueden crear aplicaciones que permitan a los usuarios controlar estos dispositivos [85]. La Figura 46 corresponde al logo de la tecnología AWS IoT Core.

Figura 46. AWS IoT Core.

Fuente: https://aws.amazon.com/es/iot-core/?nc=sn&loc=0, consultado 30/11/2020.

4.4.3. Docker

Docker es una tecnología de creación de contenedores que permite la creación y el uso de contenedores de Linux. Con DOCKER, se puede usar los contenedores como máquinas virtuales extremadamente livianas y modulares. Además, se obtiene flexibilidad con estos contenedores de poder: crearlos, implementarlos, copiarlos y moverlos de un entorno a otro, lo cual le permite optimizar las aplicaciones para la nube [86]. La Figura 47 muestra el logo de la tecnología Docker.

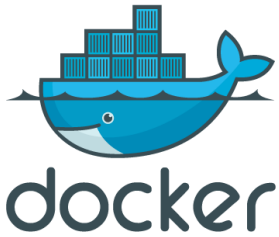

Figura 47. Docker.

Fuente: https://www.docker.com/, consultado 01/02/2021.

4.5. IMPLEMENTACIÓN DEL GATEWAY INTELIGENTE IOT

En esta sesión se muestra el proceso de implementación del Gateway Inteligente IoT de acuerdo a la arquitectura propuesta en la Figura 28.

La arquitectura del Gateway Inteligente IoT diseñado consta de 4 bloques diferenciados. La Capa de Dispositivos Hardware/Things, la Capa de Software de Dispositivo, la Capa Core | Gateway Inteligente IoT y la Capa de Plataforma para IoT. En este capítulo se profundiza en la implementación tecnológica de cada una de las capas mencionadas, apoyando algunos procedimientos técnicos con los anexos de este documento.

4.5.1. Instalación del Sistema Operativo Raspberry Pi OS

Para instalar el sistema operativo en la placa Raspberry Pi, debemos descargar la imagen del Raspberry Pi OS en su versión más reciente. Por problemas de compatibilidad detectados entre el sistema operativo y el Framework de Kura durante el desarrollo de este proyecto, se recomienda descargar la versión “Raspberry Pi OS with desktop”.

En el siguiente enlace se puede realizar la descarga de Raspberry Pi OS:

Enlace: https://www.raspberrypi.org/software/operating-systems/

Una vez descargada la imagen (archivo .iso) del sistema operativo, nos dirigimos a la página oficial de la Fundación Raspberry Pi y seguimos el paso a paso de la guía de instalación del sistema operativo: "**Setting up your Raspberry Pi**".

Enlace guía instalación: https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up

Una vez finalice la instalación del sistema operativo, al encender la Raspberry Pi se presenta el escritorio de Raspberry Pi OS, tal como lo ilustra la Figura 48.

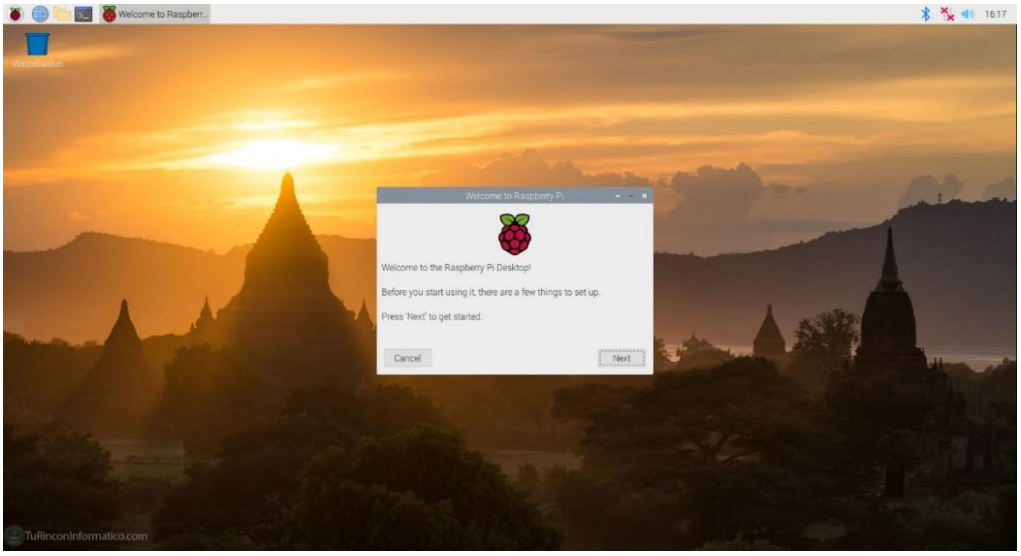

Figura 48. Escritorio Raspberry Pi OS.

Fuente: Elaboración propia.

Se debe garantizar la conexión a Internet de la Raspberry Pi, ya sea por medio de cable de red (Ethernet) o vía inalámbrica (Wifi).

Desde una terminal se actualizan la lista de paquetes disponibles y sus versiones con el siguiente comando:

```
pi@raspberrypi : ~ $ sudo apt-get update
```

Una vez actualizados los paquetes del sistema operativo, se instalas las nuevas versiones con el comando:

```
pi@raspberrypi : ~ $ sudo apt-get upgrade
```

4.5.2. Instalación del Framework Eclipse Kura

El proceso de instalación del Framework de Eclipse Kura en la placa Raspberry Pi se encuentra detallado en el documento **ANEXO A: INSTALACIÓN DE ECLIPSE KURA**. La Figura 49 muestra la interfaz principal de Kura luego del proceso de instalación.

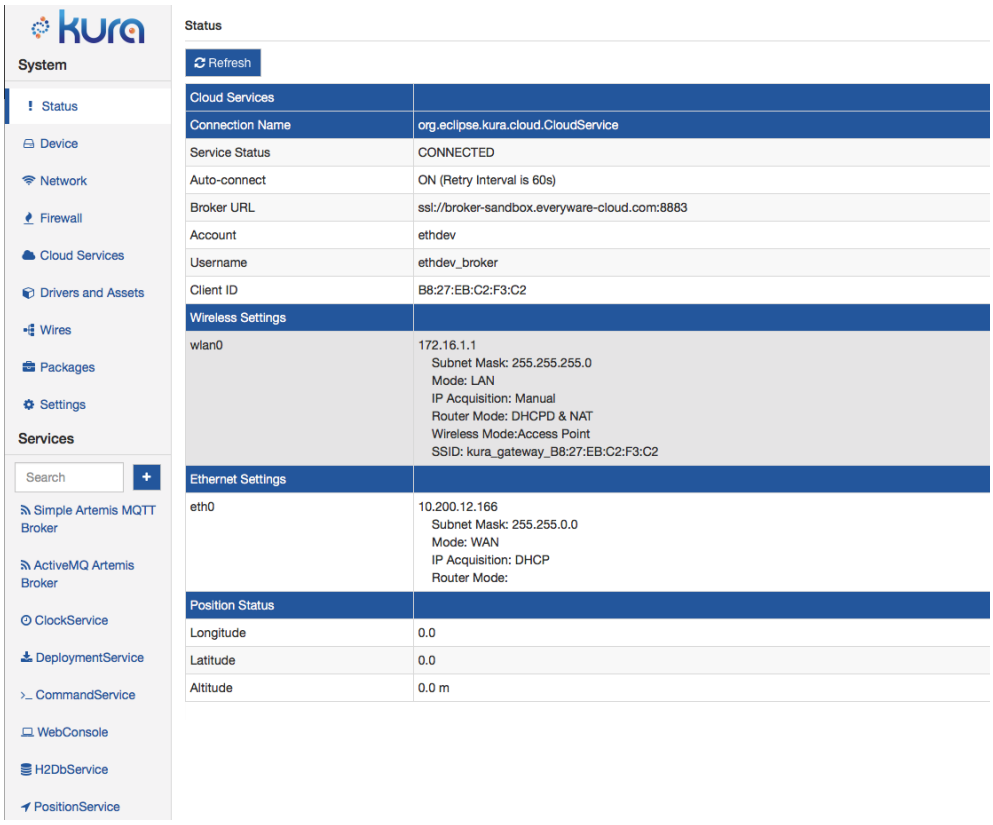

Figura 49. Interfaz principal de la instalación de Eclipse Kura.

Fuente: Elaboración propia.

Finalizada la instalación de Eclipse Kura, se pueden gestionar a través de su interfaz web los diferentes servicios, herramientas y configuraciones que el Framework ofrece.

Cabe resaltar que, en este proyecto se adoptó el desarrollo de Eclipse Kura como herramienta software principal para implementar y proveer a la Raspberry Pi las principales funciones de un Gateway IoT, de acuerdo con los criterios definidos en el diseño del Gateway, la recomendación UIT-T Y.4101/Y.2067 [48] emitida por la ITU y el reporte técnico del CORDIS “Final architectural reference model for the IoT v3.0” [46], del proyecto “The Internet of Things Architecture IoT-A” de la Comisión Europea, documentos oficiales en los cuales se establecen las capacidades, características y requisitos que debe cumplir un Gateway para aplicaciones de IoT.

En consecuencia, y para dar desarrollo a lo planteado en los objetivos de este proyecto, se trabaja sobre la base de Eclipse Kura en la implementación de los protocolos de comunicación y la arquitectura de interacción de objetos inteligente mediante un estudio de caso en el monitoreo de diferentes variables agroclimatológicas.

4.5.3. Conexión del hardware en la placa Raspberry Pi

Después de instalar Eclipse Kura, se conectan los periféricos adicionales del Gateway en la placa Raspberry Pi como lo muestra la Figura 50. En color rojo, están resaltados los protocolos definidos en el primer objetivo de este proyecto (Ethernet, Wifi y Zigbee) y los correspondientes dispositivos que permiten implementarlos en el Gateway.

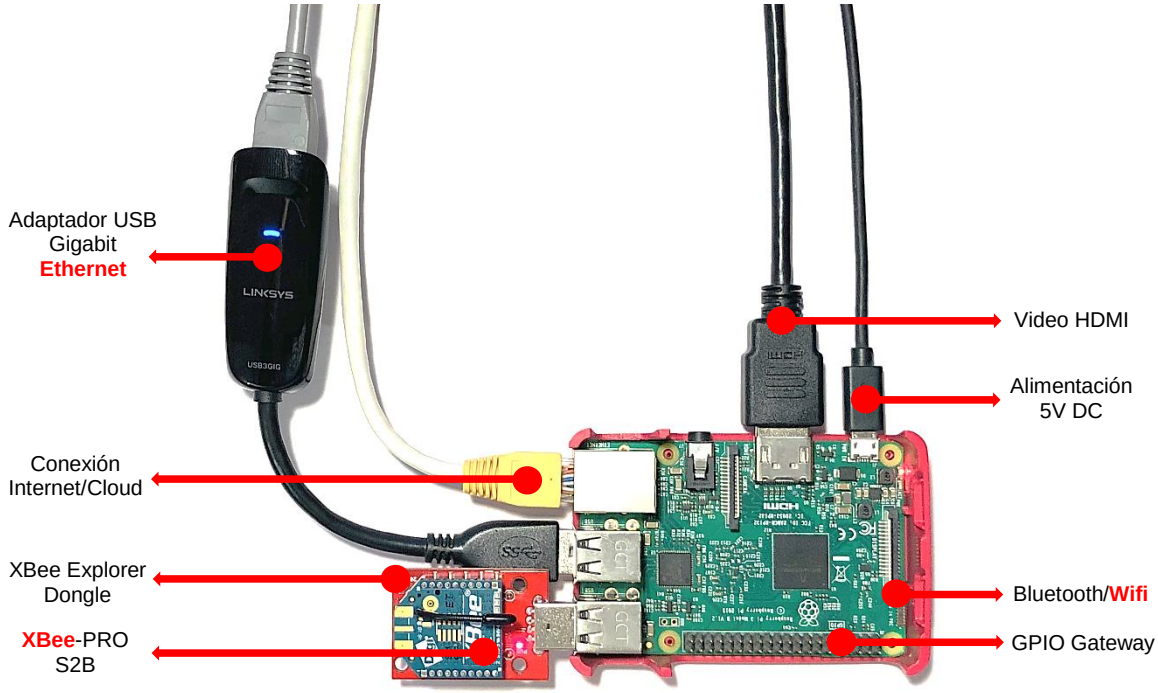

Figura 50. Conexiones hardware del Gateway Inteligente IoT.

Fuente: Elaboración propia.

4.5.4. Conexión de los módulos XBee para comunicación con el protocolo Zigbee

La implementación del protocolo de comunicación Zigbee se llevó a cabo con dos módulos XBee S2C que establecen un enlace inalámbrico punto a punto para el envío de datos.

Nota: Debido a inconvenientes presentados en los módulos XBee-PRO S2B fue necesario adquirir dos nuevos módulos XBee de la serie S2C (XBee S2C). Al conectar los módulos S2B en el Explorer no eran descubiertos por el software XCTU. También, se hicieron pruebas para reprogramarlos y tratar de restablecer el firmware pero, no fue posible hacer funcionar los módulos, por tal motivo, de aquí en adelante haremos referencia a los módulos XBee S2C como los dispositivos que proveen conectividad Zigbee entre el Gateway y el Dispositivo IoT.

Un primer módulo XBee, se conecta a la placa Arduino Uno Rev3 por medio del Shield XBee SparkFun. Los datos climatológicos de este dispositivo IoT conformado, son tomados por un Sensor de Temperatura Grove V1.2 y transmitidos al Gateway a través del protocolo Zigbee, utilizando el módulo XBee. Este conjunto de dispositivos conforman el primer

dispositivo IoT que se conecta al Gateway y que se ilustra en la Figura 28. Arquitectura de implementación del Gateway Inteligente IoT.

La Figura 51 ilustra el dispositivo IoT agroclimatológico que mide la temperatura de una planta y envía los datos medidos al Gateway a través del protocolo Zigbee. En la Figura 50, se ilustra la conexión del módulo XBee que recibe el dato de temperatura de la planta y lo integra al Gateway para ser utilizado en los servicios y aplicaciones que este implemente.

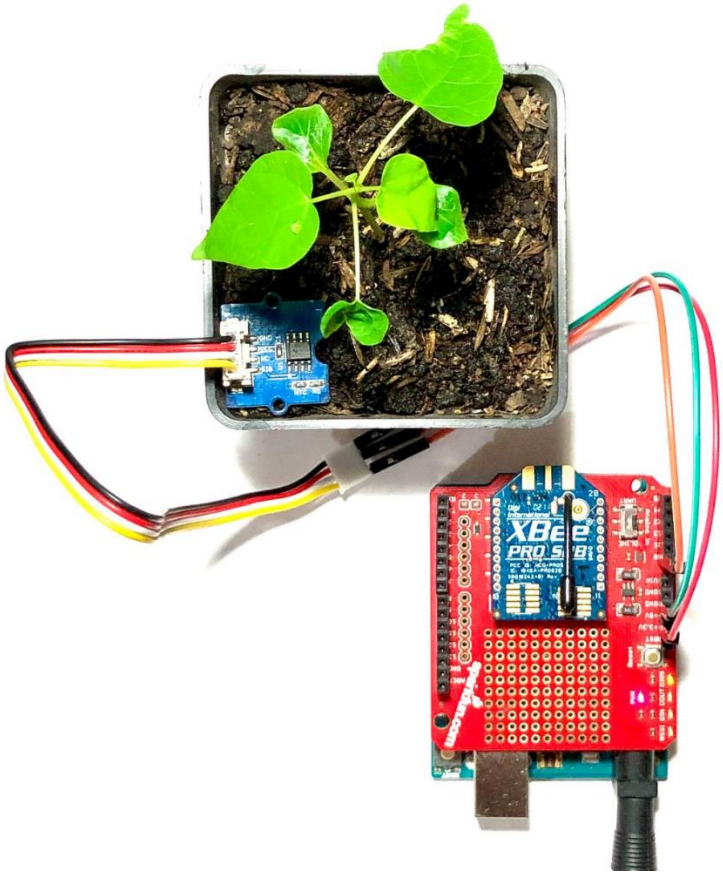

Figura 51. Dispositivo IoT sensor de temperatura con comunicación vía protocolo Zigbee.

Fuente: Elaboración propia.

4.5.5. Configuración de módulos XBee para comunicación con el protocolo Zigbee

La configuración de los módulos XBee se detalla en el **ANEXO B: CONFIGURACIÓN DE MÓDULOS XBEE DESDE XCTU.**

Una vez realizada la conexión del módulo XBee tanto en el Gateway como en el Dispositivo IoT, se procede a configurarlos para establecer comunicación entre ellos.

Para identificar (Figura 52) el módulo XBee conectado al puerto USB del Gateway, se procede a desconectarlo y hacer un escaneo de los puertos de comunicación serial del Gateway desde la terminal. Para ello se ejecuta el siguiente comando:

```
pi@raspberrypi : ~ $ lsusb
```

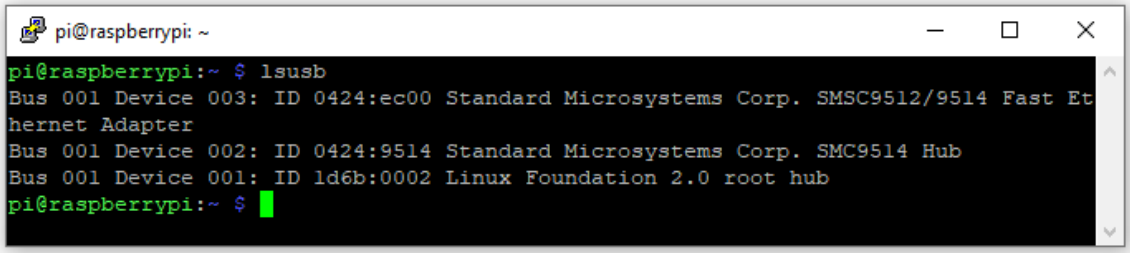

Figura 52. Dispositivos USB conectados al Gateway Inteligente IoT.

Fuente: Elaboración propia.

Se conecta de nuevo el SparkFun XBee Explorer Dongle con el módulo XBee S2C al puerto USB de la Raspberry y se listan de nuevo los dispositivos USB conectados con el comando:

```
pi@raspberrypi : ~ $ lsusb
```

Cuando se escanean los puertos USB con el módulo XBee conectado a la Raspberry Pi, se detecta una nueva conexión serial como lo ilustra la Figura 53.

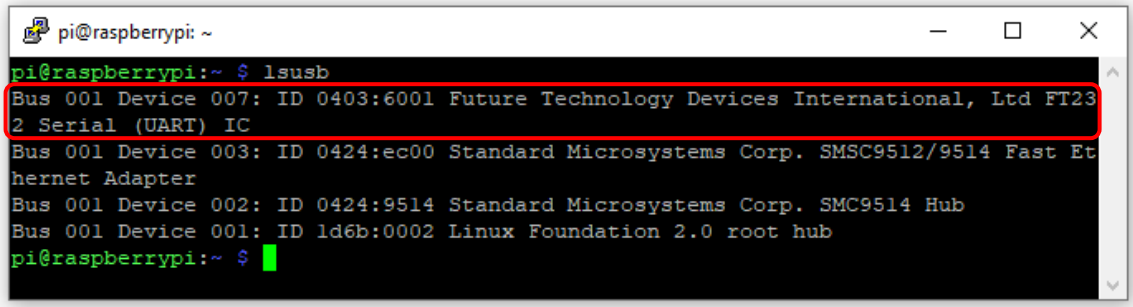

Figura 53. Dispositivos USB reconocidos en la Raspberry Pi.

Fuente: Elaboración propia.

Para conocer la identificación (nombre) del dispositivo en el puerto de comunicación serial ejecutamos el siguiente comando desde la terminal:

```
pi@raspberrypi : ~ $ ls /dev/tty*
```

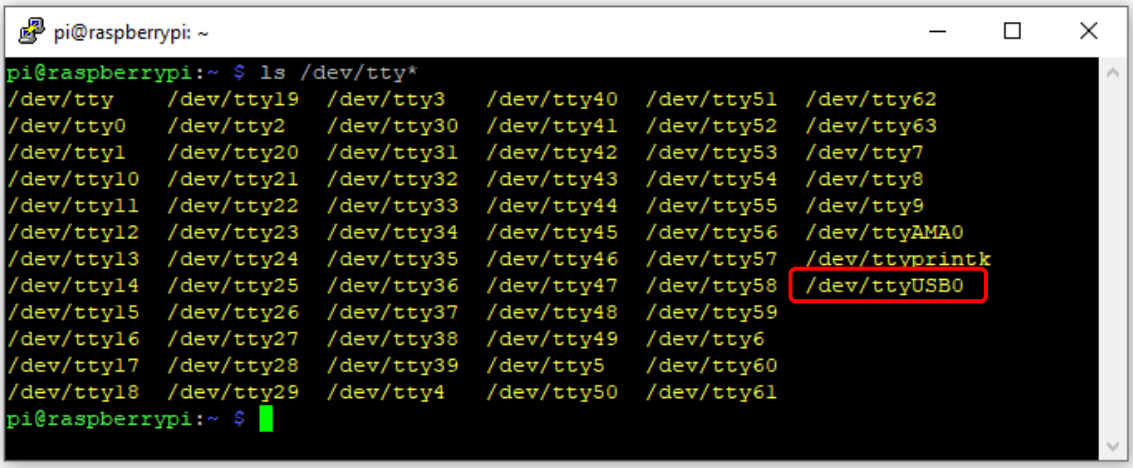

Figura 54. Identificación de dispositivos de comunicación serial.

Fuente: Elaboración propia.

Se puede observar en la Figura 54 la identificación del SparkFun XBee Explorer Dongle conectado al puerto USB, la cual es necesario conocer para implementar el módulo software encargado de realizar la lectura de los datos.

Como procedimiento de prueba y verificación, se realiza la instalación de la herramienta **Minicom**, la cual permite leer los datos de los puertos de comunicación serial del Gateway para validar que se esté recibiendo el dato de temperatura medida por el sensor conectado al dispositivo IoT que realiza el envío de datos vía Zigbee hacia el Gateway. Para la instalación de Minicom ejecutamos en una terminal los siguientes comandos:

Se actualizan los paquetes disponibles.

```
pi@raspberrypi : ~ $ sudo apt update
```

Se instala el sistema de gestión de paquetes Snap.

```
pi@raspberrypi : ~ $ sudo apt install snapd
```

Reiniciamos el Gateway.

```
pi@raspberrypi : ~ $ sudo reboot
```

Finalmente, se instala el monitor de puerto serie **Minicom**.

```
pi@raspberrypi : ~ $ sudo snap install minicom
```

El resultado de la instalación desde la terminal se ilustra en la Figura 55.

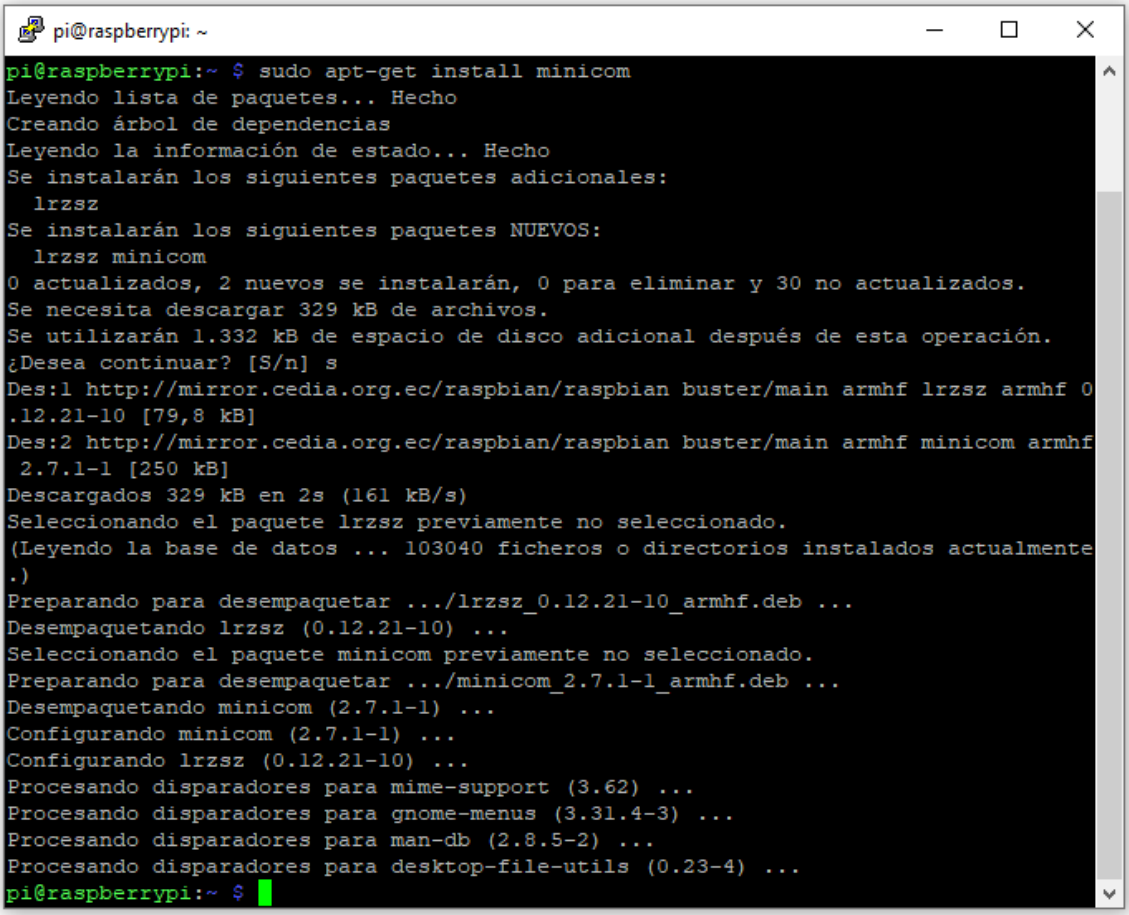

Figura 55. Instalación de monitor de puerto serie Minicom.

Fuente: Elaboración propia.

El uso del monitor de puerto seria es sencillo, se debe marcar en opciones, el puerto físico y la velocidad de en baudios (unidad de medida de la velocidad de transmisión de señales que se expresa en símbolos por segundo).el parámetro “-D” corresponde al dispositivo o puerto serie a leer y “-b” la velocidad, tal como se ilustra en el siguiente comando.

```
pi@raspberrypi : ~ $ minicom -D /dev/ttyUSB0 -b 9600
```

La Figura 56 muestra el dato recibido por el Gateway, que corresponde a la temperatura de la planta de la Figura 51, monitoreada por el dispositivo IoT con conexión Zigbee.

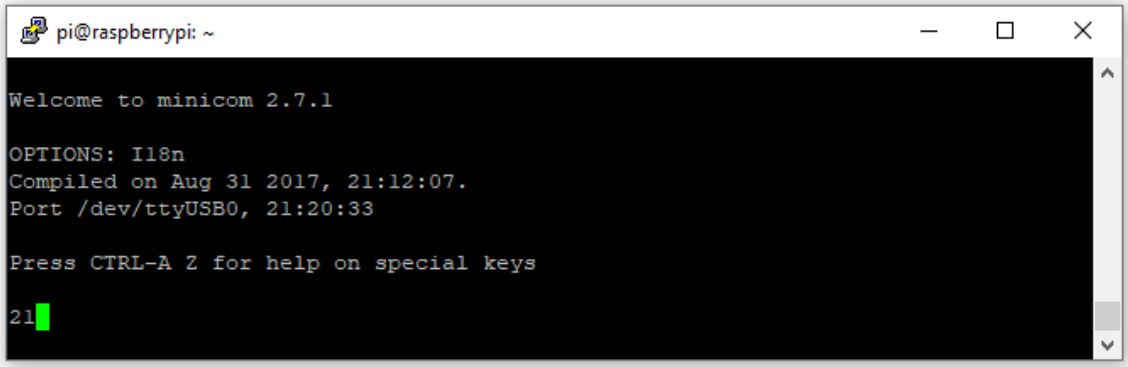

Figura 56. Dato de temperatura de la planta monitoreada por el dispositivo IoT Zigbee.

Fuente: Elaboración propia.

El código fuente que corre en el dispositivo IoT agroclimatológico sensor de temperatura, fue implementado tomando como referencia el "Demo code for Grove – Temperature sensor V1.1/1.2"[36] de la página oficial del fabricante del sensor y la librería de comunicación serial para Arduino. La Figura 57 ilustra el código fuente mencionado.

```
#include <SoftwareSerial.h>
#include <math.h>

const int B = 4275;               // B value of the thermistor
const int R0 = 100000;            // R0 = 100k
const int pinTempSensor = A0;     // Grove - Temperature Sensor connect to A0

SoftwareSerial XBee(2, 3); // RX, TX

void setup()
{
    XBee.begin(9600);
    Serial.begin(9600);}

void loop()
{
    int a = analogRead(pinTempSensor);
    float R = 1023.0/a-1.0;
    R = R0*R;
    float temperature = 1.0/(log(R/R0)/B+1/298.15)-273.15; // convert to temperature via datasheet

    Serial.print("Temperatura = ");
    Serial.println(temperature);
    XBee.write(temperature);
    delay(5000);}
```

Figura 57. Código fuente sensor de temperatura Zigbee.

Fuente: Grove - Temperature Sensor V1.2 - Seeed Wiki (seeedstudio.com)

[36] Grove - Temperature Sensor V1.2 - Seeed Wiki (seeedstudio.com), consultado 01/02/2021.

Se realizaron pruebas desde el monitor serial del IDE de Arduino para validar el funcionamiento local del sensor.
La Figura 58 ilustra el envío de datos a través del puerto serial.

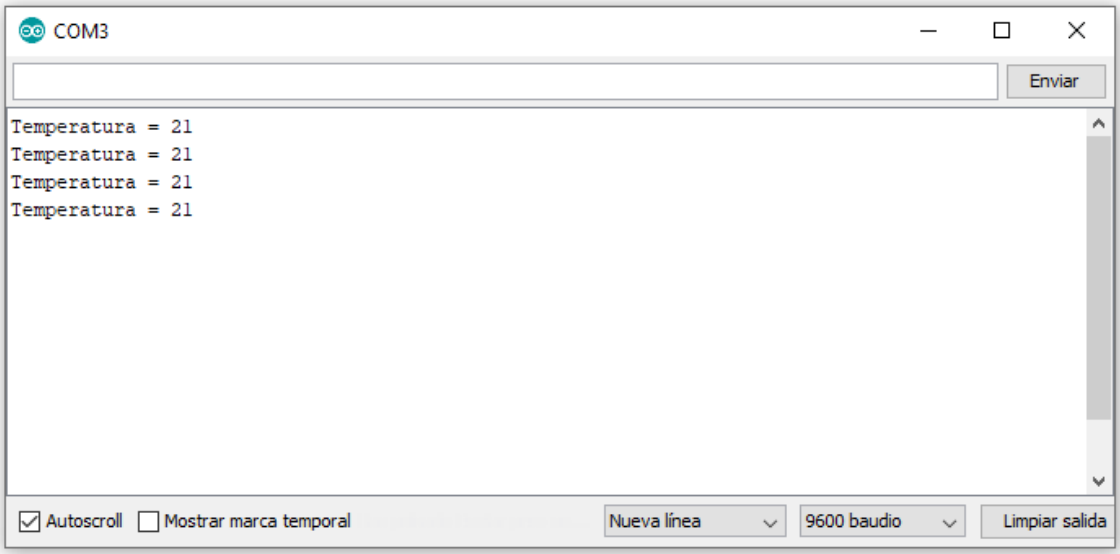

Figura 58. Dato de temperatura del dispositivo IoT agroclimatológico sensor de temperatura en el monitor serie del IDE de Arduino.

Fuente: Elaboración propia.

Por su parte, el código fuente implementado en el Gateway, corresponde al que se muestra a continuación:

```
import time
import serial
import RPi.GPIO as GPIO

ser = serial.Serial(
    port='/dev/ttyUSB0',
    baudrate=9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=1
)
while 1:
    x = ser.readline().strip()
    print (x)
```

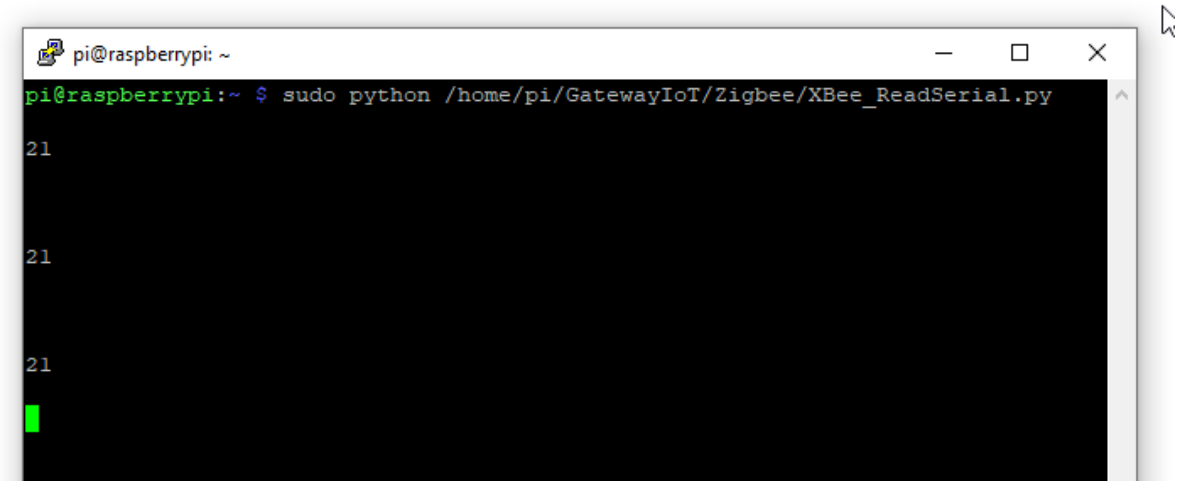

Figura 59. Ejecución de código fuente en el Gateway para sensor de temperatura Zigbee.

Fuente: Elaboración propia.

4.5.6. Configuración de Objetos Inteligentes

En el **ANEXO C: CONFIGURACIÓN DE OBJETOS INTELIGENTES**, se detalla el proceso de instalación de paquetes y librerías para implementar los Objetos Inteligentes.

Copiamos el repositorio de archivos del escenario en el escritorio.

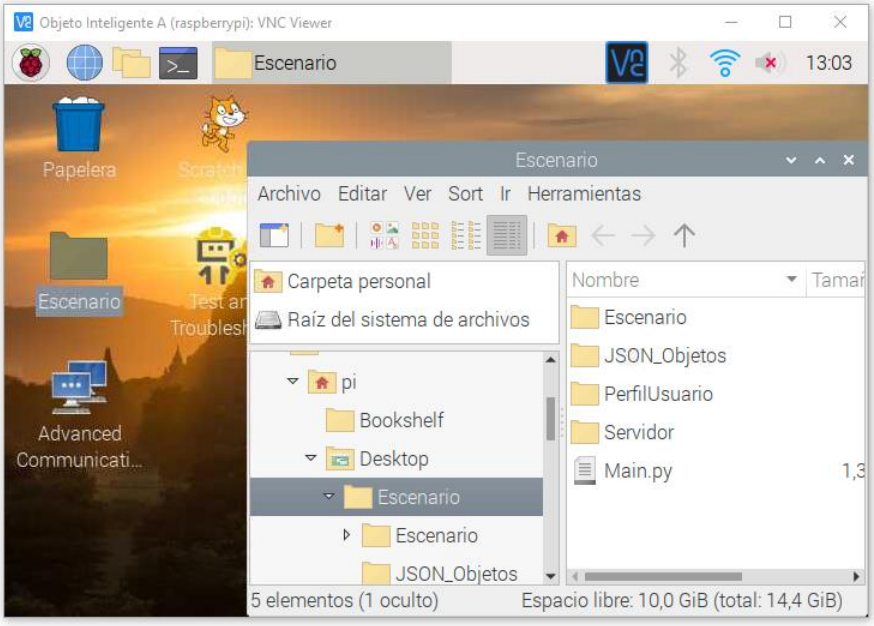

Figura 60. Configuración de Objeto Inteligente A.

Fuente: Elaboración propia.

Ejecutamos el archivo Main.py para iniciar el escenario en el Gateway.

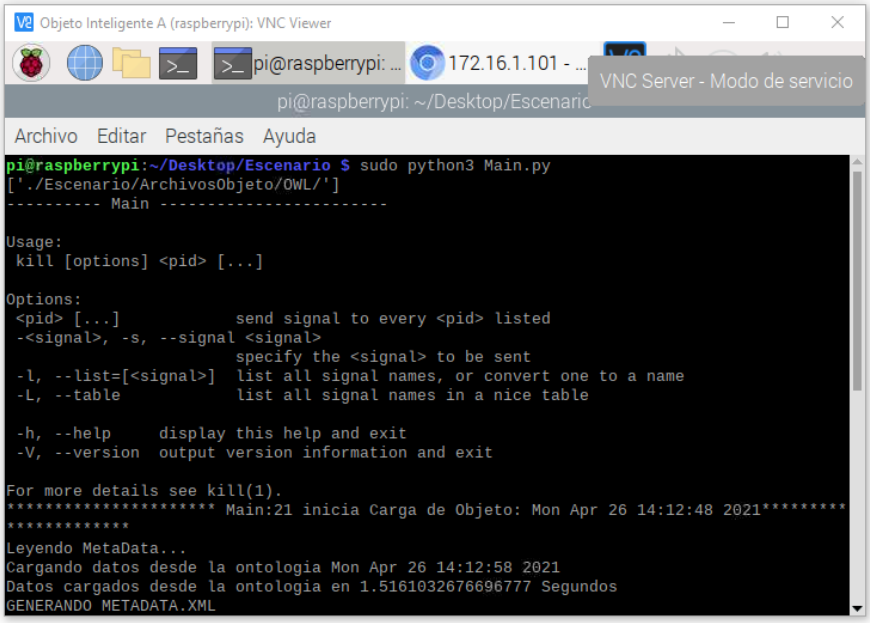

Figura 61. Ejecución del escenario de objetos inteligentes en el Objeto A.

Fuente: Elaboración propia.

Ubicamos el archivo JSON que contiene los metadatos del objeto inteligente.

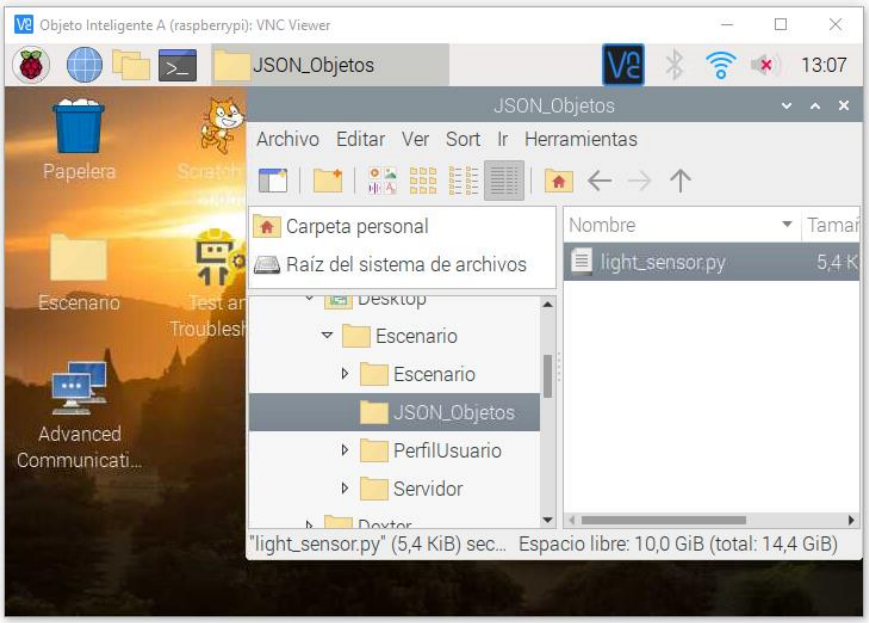

Figura 62. Archivo JSON con metadatos del Objeto Inteligente A.

Fuente: Elaboración propia.

Se configuran los metadatos del Objeto Inteligente por medio del archivo JSON.

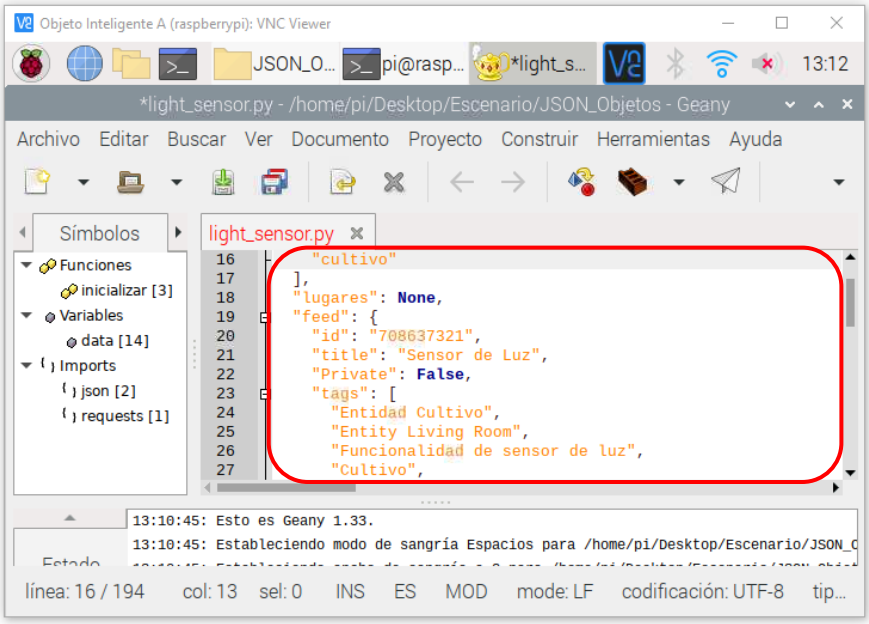

Figura 63. Configuración de Archivo JSON con metadatos del Objeto Inteligente.

Fuente: Elaboración propia.

Se establece la dirección IP del Objeto Inteligente en la red local.

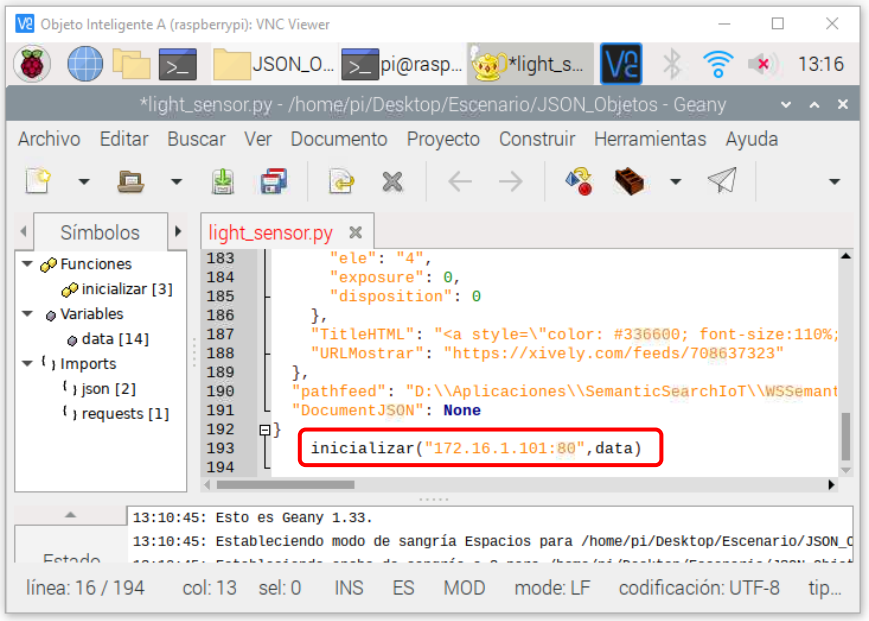

Figura 64. Dirección IP Objeto Inteligente A.

Fuente: Elaboración propia.

La configuración anterior aplica para el Objeto Inteligente B y todos los demás que se desplieguen, salvo el archivo JSON de configuración que contiene los metadatos de cada objeto de acuerdo a los sensores y/o actuadores que cada uno implemente.

4.5.7. Despliegue del Gateway con los Dispositivos IoT y Objetos Inteligentes

La Figura 65 muestra el Dispositivo IoT y los Objetos Inteligentes implementados en el desarrollo de este proyecto. De manera ilustrada se quiere representar en esta imagen una vista global de la solución, la forma en la que se conectan los Inteligentes y el Dispositivo IoT, así como también los sensores y variables del clima que cada uno captura para ser enviadas al Gateway y reutilizadas para la creación de nuevos servicios y aplicaciones inteligentes.

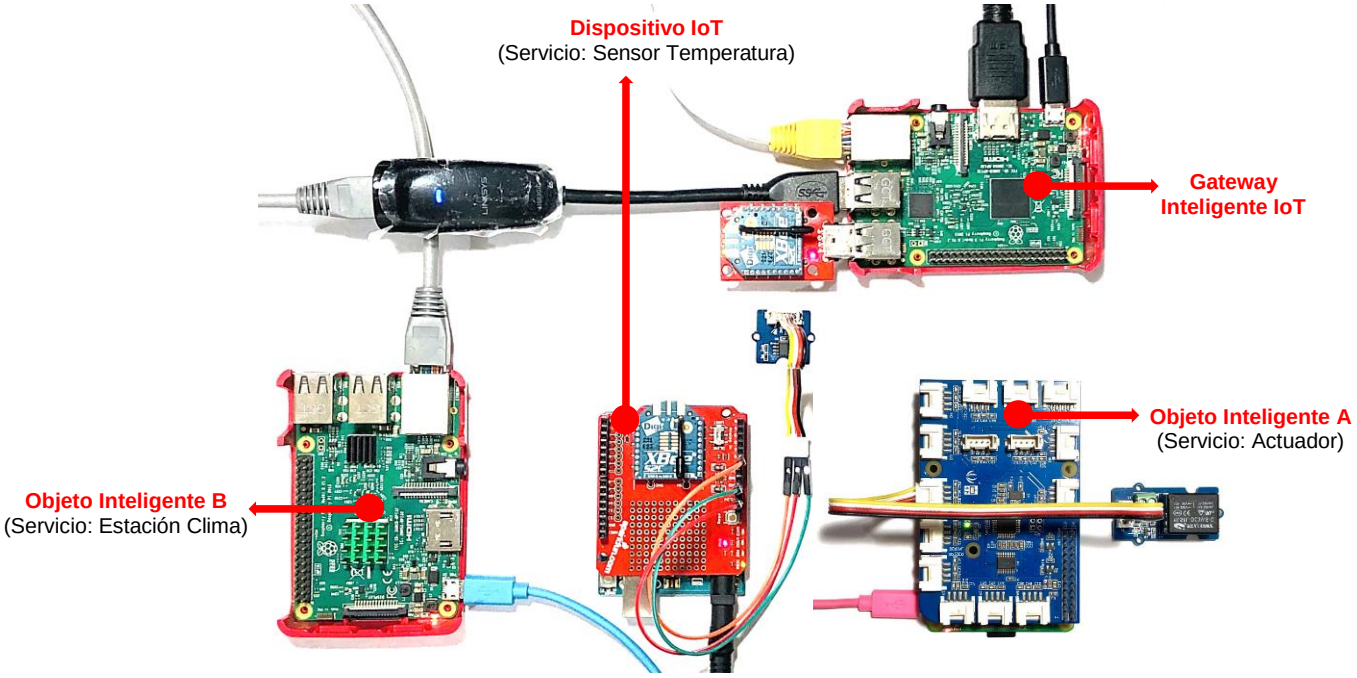

Figura 65. Gateway y Objetos Inteligentes.

Fuente: Elaboración propia.

Fueron desplegados dos Objetos Inteligentes que implementan la Arquitectura de Objeto Inteligente [9] cuya configuración se describe en el numeral 4.5.6. Un primer Objeto Inteligente A, Figura 66, incorpora un dispositivo relé (actuador) y se comunica con el Gateway por medio del protocolo Wifi.

Un segundo Objeto Inteligente B, Figura 67, incorpora la estación climatológica encarga de medir las variables del clima descritas en el tercer objetivo de este proyecto. La estación que incorpora este Objeto Inteligente, está conformada por: Figura 36. Shield de Clima Sparkfun**.** y Figura 37. Sensores de viento y lluvia estación meteorológica Sparkfun. La conexión y envío de datos entre el Gateway y el segundo Objeto Inteligente se realiza por medio del protocolo Ethernet.

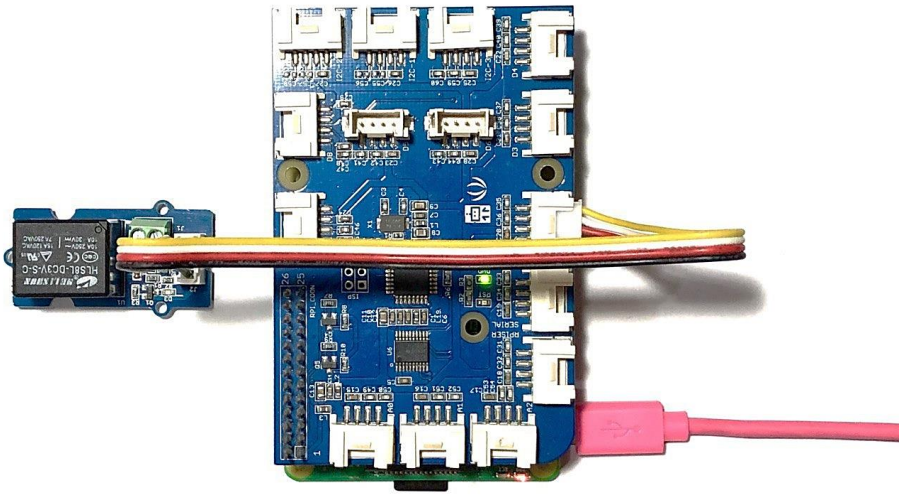

Figura 66. Objeto Inteligente A (Actuador) con comunicación vía Wifi.

Fuente: Elaboración propia.

Figura 67. Objeto Inteligente B (Estación Clima) con comunicación vía Ethernet.

Fuente: Elaboración propia.

4.5.7.1. Alistamiento de Sensores Agroclimatológicos

Se realizó el montaje de la estación climatológica Sparkfun con sus sensores de viento y lluvia y el Shield Clima. En la siguiente tabla se muestran algunas figuras del montaje realizado.

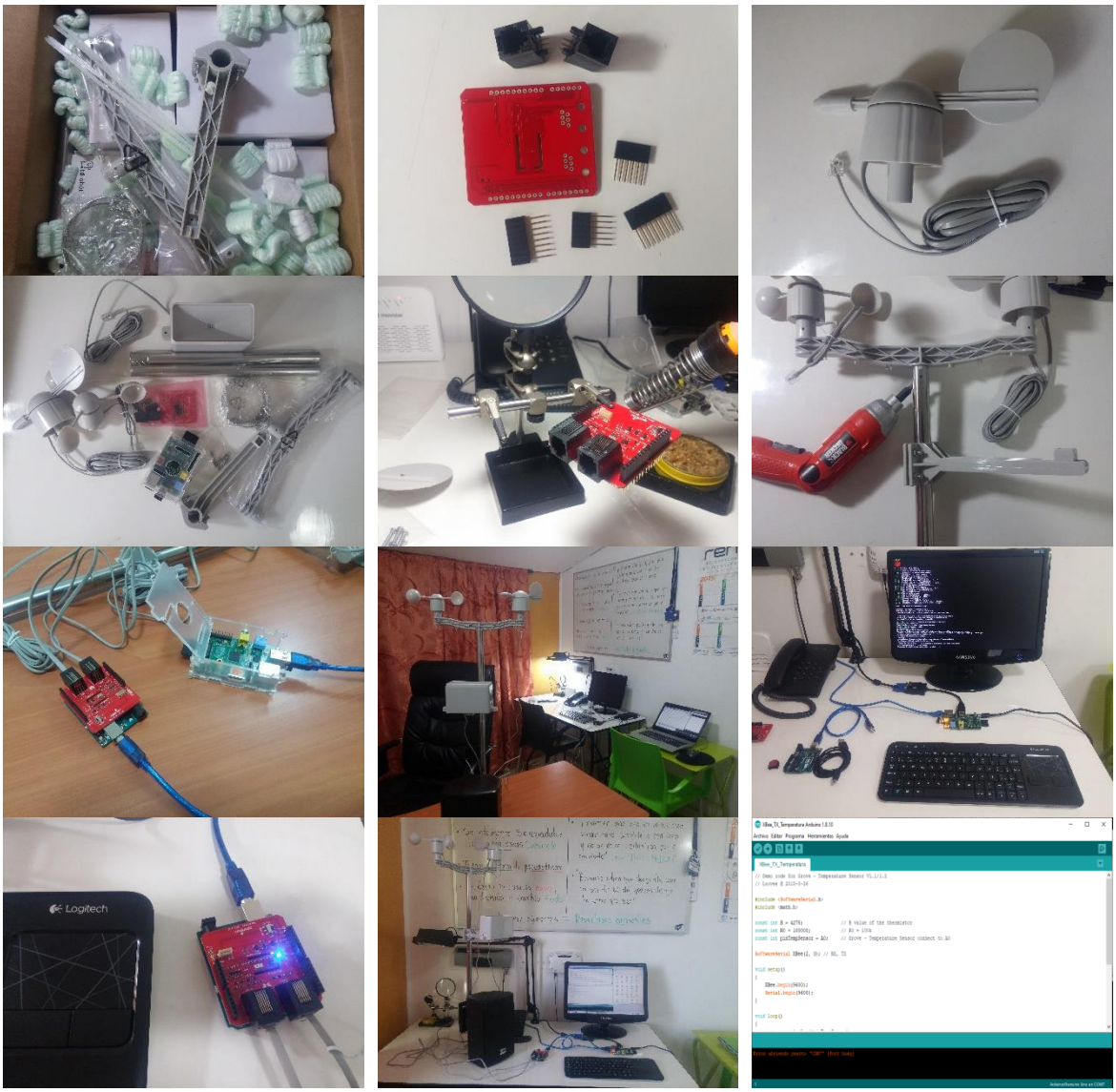

Tabla 9. Registro fotográfico de montajes y pruebas de sensores agroclimatológicos.

Fuente: Elaboración propia

4.5.7.2. Despliegue de escenario estudio de caso

En la siguiente figura se puede observar la instalación de la estación meteorológica que registra las variables de: temperatura, humedad, presión barométrica, nivel de luz, nivel de lluvia, dirección y velocidad del viento. El funcionamiento de la estación se dio en un entorno

interior con algunos fenómenos simulados, como el viento, lluvia (accionamiento manual del pluviómetro para registrar datos), veleta y sensor de luz.

Figura 68. Montaje y pruebas de sensores agroclimatológicos en un entorno simulado para el estudio de caso propuesto.

Fuente: Elaboración propia

En el CAPÍTULO 5 del presente documento se detallan las pruebas realizadas y los resultados obtenidos.

6.2.1. Agregar nueva interfaz de red Ethernet a Eclipse Kura

Para conectar los datos de la estación climatológica con el Gateway IoT, fue necesario implementar una nueva interfaz física de red por protocolo Ethernet por medio del Adaptador USB a Ethernet. A continuación se ilustra el proceso llevado a cabo para configurar la nueva interfaz de red en el Gateway Inteligente IoT y permitir así la conexión del Objeto Inteligente B a la red local del Gateway.

La Figura 69 muestra las interfaces que por defecto están listadas en el Framework Kura para la placa Raspberry Pi. En la Figura 70 se muestra la nueva interfaz agregada al conectar el adaptador de red Ethernet al puerto USB de la Raspberry.

Las Figura 71 y Figura 72, ilustran la configuración IP y de parámetros de red necesarios para el correcto funcionamiento de la interfaz **eth1**.

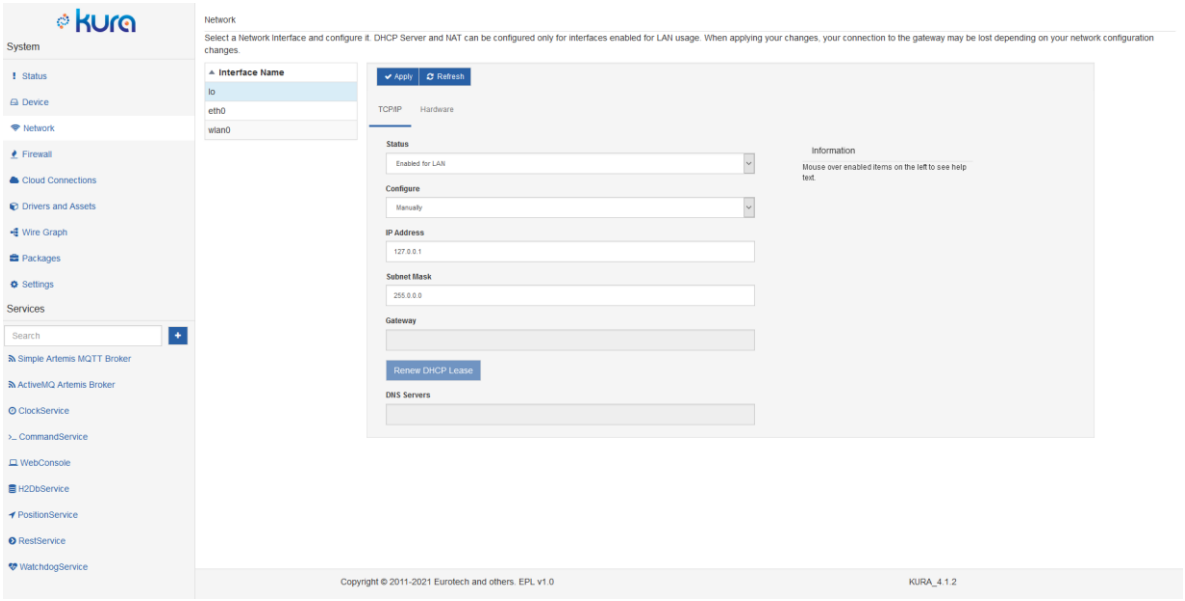

Figura 69. Interfaces de red por defecto en Kura instalado en la placa Raspberry Pi 3.

Fuente: Elaboración propia.

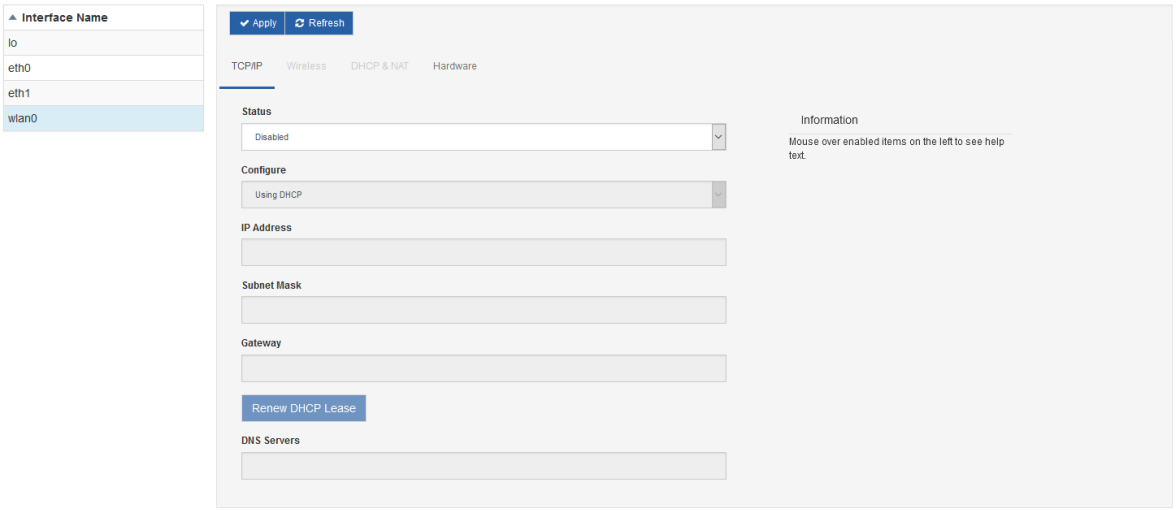

Figura 70. Interfaz de red agregada con adaptador USB Ethernet Modelo USB3GIG de Linksys.

Fuente: Elaboración propia.

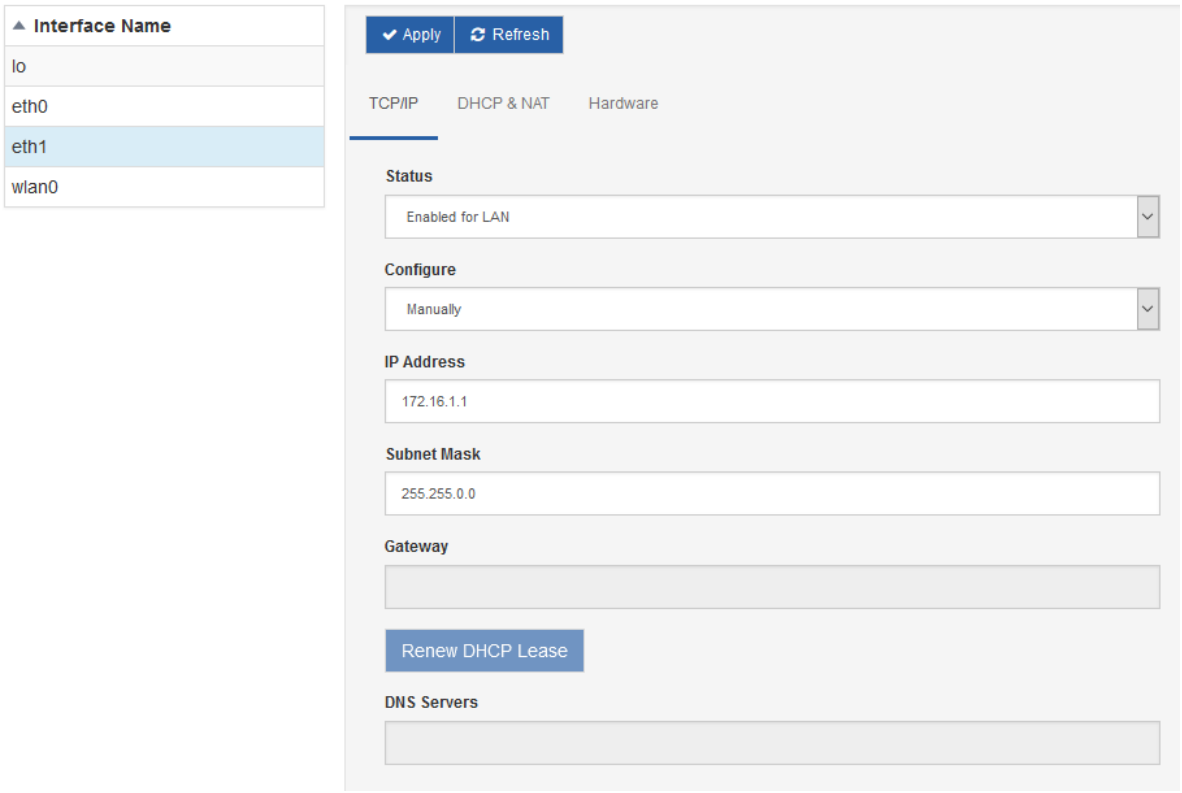

Figura 71. Configuración IP de una nueva interfaz de red Ethernet (eth1).

Fuente: Elaboración propia.

Procedemos a configurar el parámetro **Router Mode** en la pestaña **DHCP & NAT** en el parámetro **NAT Only** y tendremos lista la nueva interfaz de red agregada al Gateway para conectar dispositivos a la red local por medio de la conexión cableada que provee el protocolo Ethernet.

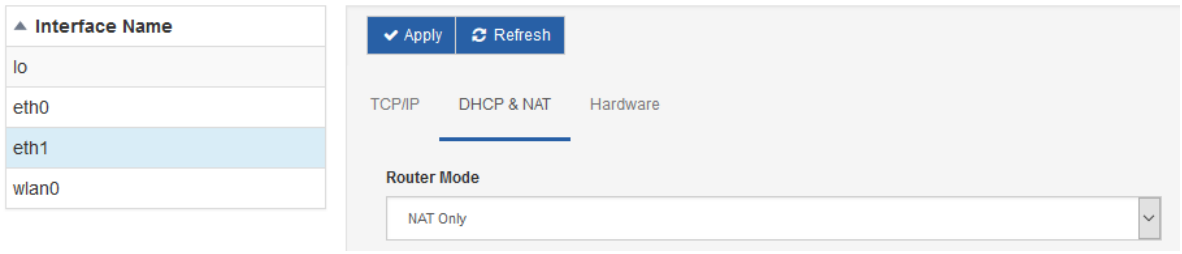

Figura 72. Configuración DHCP & NAT para la nueva interfaz de red Ethernet (eth1).

Fuente: Elaboración propia.

La Figura 73 y Figura 74, muestran las conexiones realizadas entre el Gateway, el adaptador de red que provee la nueva interfaz y el Switch que permite disponer de más puertos Ethernet para conectar dispositivos de red cableados.

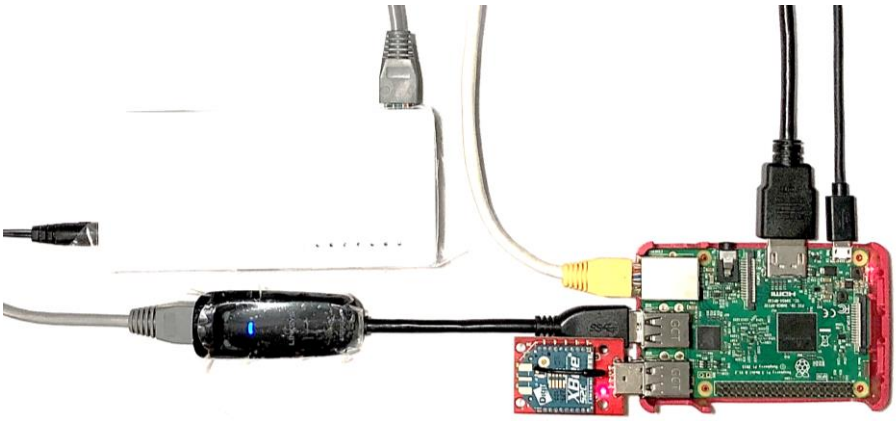

Figura 73. Switch 8-Port 10/100 Mbps con interfaz Ethernet adicionadas al Gateway Inteligente IoT.

Fuente: Elaboración propia.

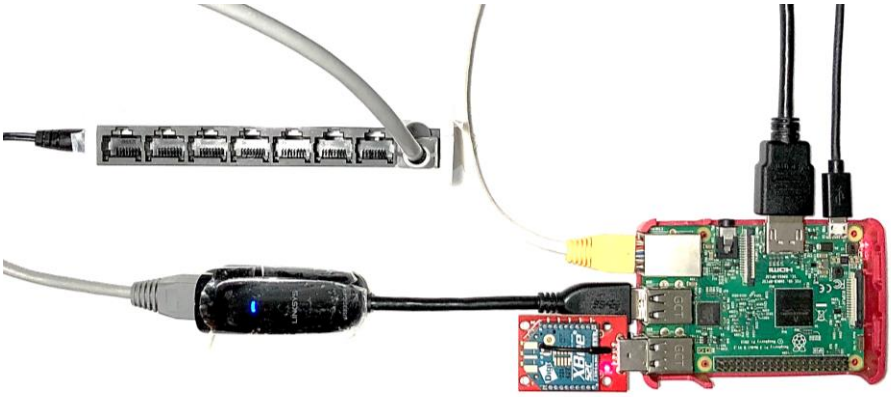

Figura 74. Puertos de red Ethernet adicionados al Gateway Inteligente IoT.

Fuente: Elaboración propia.

4.5.8. Configuración de Eclipse Kura con la Plataforma AWS IoT Core

El proceso configuración de la conexión entre Eclipse Kura y la plataforma AWS IoT Core se encuentra detallado en el documento **ANEXO D: CONFIGURACIÓN DE AWS IOT CORE**.

La Figura 75 muestra el resultado de la generación de certificados SSL/TLS[37] para establecer conexión entre Eclipse Kura y la plataforma AWS IoT Core.

[37] https://docs.aws.amazon.com/es_es/acm/latest/userguide/acm-concepts.html#concept-ssl, consultado 30/11/2020. SSL (Secure Sockets Layer) y TLS (Transport Layer Security) son protocolos criptográficos que proporcionan seguridad de comunicación a través de una red de equipos. TLS es el sucesor de SSL. Los dos utilizan certificados X.509 para autenticar el servidor. Ambos protocolos negocian una clave simétrica entre el cliente y el servidor que se utiliza para cifrar el flujo de datos entre las dos entidades.

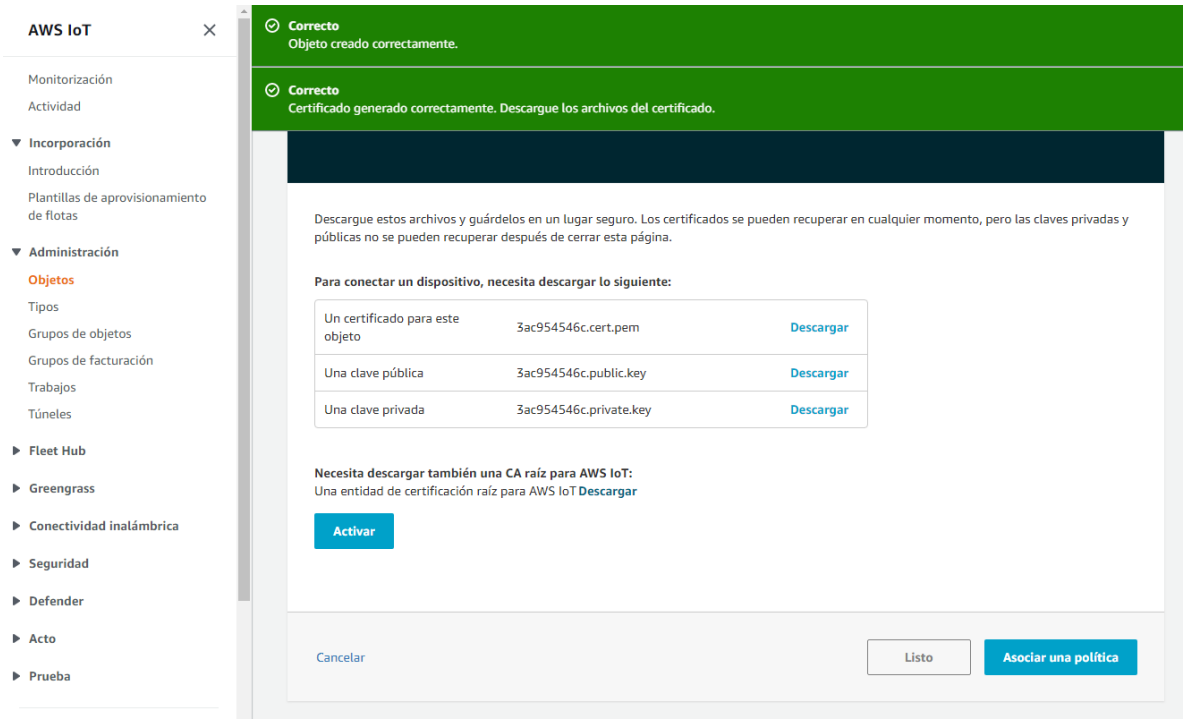

Figura 75. Certificados SSL/TLS para conexión de Kura con AWS IoT Core.

Fuente: Elaboración propia.

Al final de la configuración de la conexión entre Eclipse Kura y AWS IoT Core, debemos encontrar una nueva conexión en el parámetro **Cloud Connections** de la interfaz principal del Gateway, como lo ilustra la
Figura ***76***.

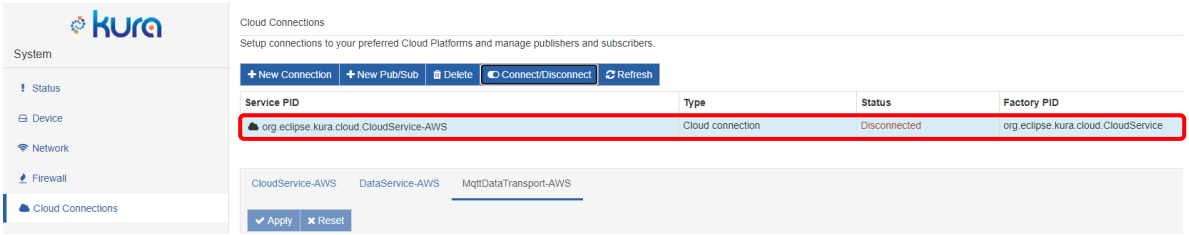

Figura 76. Servicio de conexión “Cloud Connections” Eclipse Kura y AWS IoT Core.

Fuente: Elaboración propia.

Aunque se ha creado el servicio de conexión con la plataforma para IoT, aún no se puede iniciar el proceso de comunicación entre la plataforma y el Gateway. Para que esto sea posible se debe establecer la conexión con el servicio creado, haciendo clic en el botón **Connect/Disconnect**. Finalmente se habrá establecido la conexión entre la plataforma para IoT y el Gateway como lo ilustra la Figura 77.

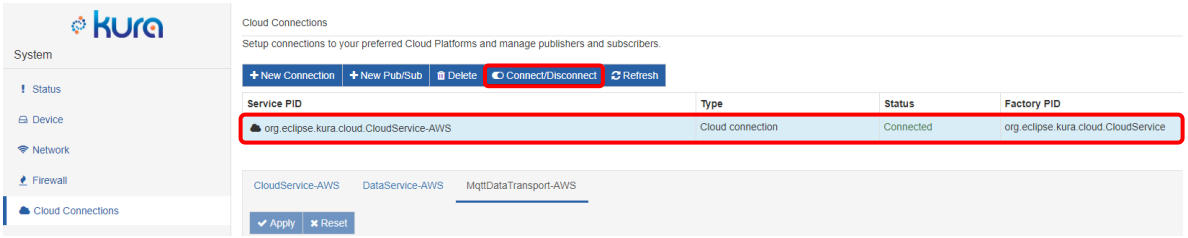

Figura 77. Establecimiento de conexión entre Eclipse Kura y AWS IoT Core.

Fuente: Elaboración propia.

4.5.9. Publicación y suscripción de la información IoT Core de Amazon AWS. Servicio CloudService.

Para la comunicación con el servicio AWS IoT Core que proporciona Amazon, utilizaremos el protocolo MQTT. Publicaremos y nos suscribiremos a los topics que el IoT Core nos proporciona. Eclipse Kura proporciona un servicio para interactuar con los topics, **CloudService** [83], cuya estructura y servicios podemos ver en la Figura 78.

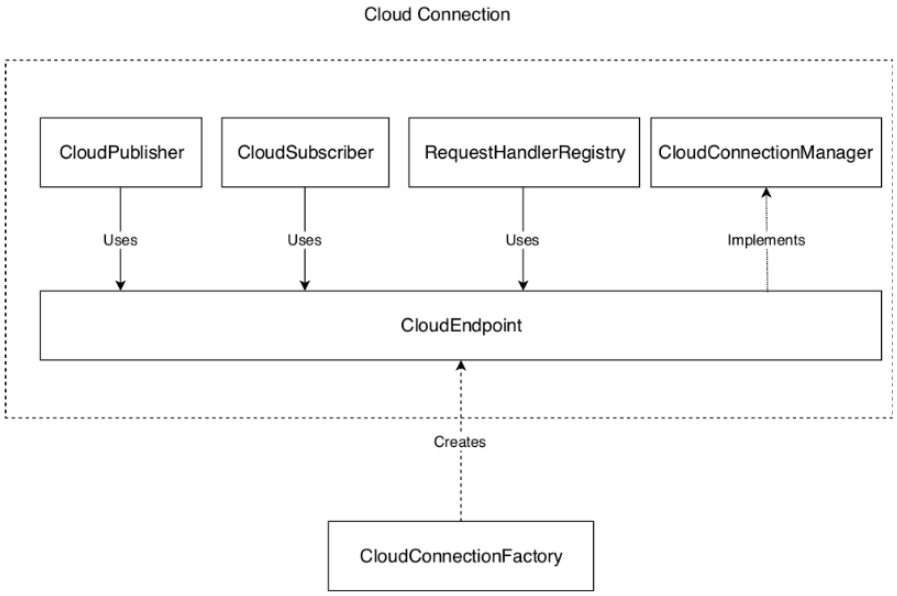

Figura 78. Servicios de CloudService

Fuente: Eclipse Foundation, "Eclipse Kura Documentation." [Online]. Available: https://eclipse.github.io/kura/. [Accessed: 26-Jul-2019].

Para la publicación de datos, en este trabajo fueron de utilidad los servicios **CloudPublisher** y **CloudSuscriber**. El primero, permite a las aplicaciones publicar en una plataforma para IoT. El segundo, permite a las aplicaciones suscribirse a una plataforma para IoT. Adicionalmente, es necesario establecer conexión con un **CloudEndpoint**, en este caso, el proporcionado por AWS IoT Core. En la documentación oficial del proyecto Eclipse Kura [83] se muestra cómo crear un **CloudService**, asociándole un **CloudEndpoint**.

Una vez se ha logrado crear el **CloudService**, por medio de la interfaz web de Kura se pueden modificar los parámetros de configuración **MqttDataTransport-AWS** en el menú **Cloud Connections**. Los parámetros configurados para este servicio son los que se muestran en la Figura 79.

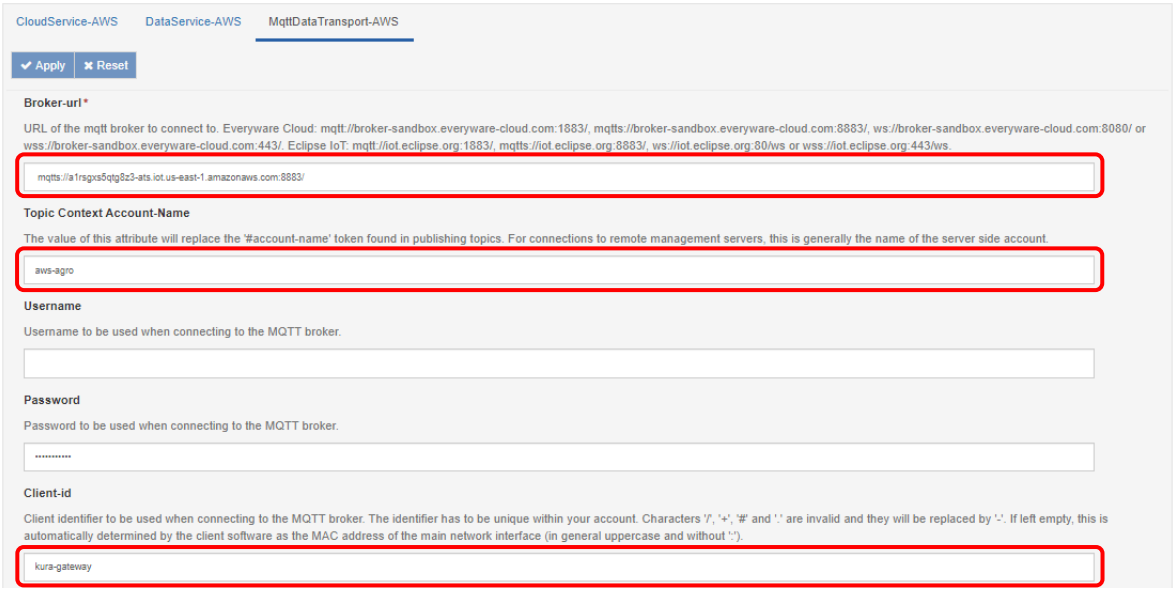

Figura 79. Parámetros MQTTDataTransport del servicio Cloud Connections.

Fuente: Elaboración propia.

- **Broker-url**: Corresponde al CloudEndpoint proporcionado por AWS IoT Core.

- **Topic Context Account-Name**: Nombre de la cuenta para establecer una conexión con AWS IoT Core, en este caso: **aws-agro**.

- **Client-id**: Corresponde al identificador único de la cuenta que se usa cuando se establece una conexión con el MQTT bróker. En este caso se llamará **kura-gateway**.

- **Topic**: Corresponde al **Topic** en el que se quiere publicar. Para este caso será '**EDC/#account-name/#client-id/MQTT/LWT**'. Los tokens '**#account-name**' y '**#client-id'** serán reemplazados por los valores establecidos en el parámetro **topic.context.account-name** y **client-id**.

Una vez realizada la configuración, los publicadores/suscriptores ejecutarán sus respectivas funciones en el **Topic** del **CloudEndpoint** indicado en el campo **Broker-url**.

En el numeral 5.5 del presente proyecto, se muestra las variables publicadas, correspondientes al porcentaje de humedad y temperatura en grados centígrados.

CAPÍTULO 5 EVALUACIÓN GATEWAY INTELIGENTE IoT

Para la evaluación del Gateway Inteligente IoT desarrollado se estableció como estudio de caso su implementación en el monitoreo de variables agroclimatológicas.

5.1. INTEGRACIÓN DEL GATEWAY CON OBJETOS INTELIGENTES

En la sesión **4.5.7. Despliegue del Gateway con los Dispositivos IoT y Objetos Inteligentes** se muestra el despliegue de los dispositivos utilizados. En una primera instancia, dado que el Objeto Inteligente B tiene conectada la estación climatológica, encargada de capturar las variables del clima definidas en los objetivos del proyecto, se realizar una evaluación cualitativa en cuanto a las funciones alcanzadas por la solución desarrollada.

Los datos climatológicos enviados a través del puerto serie de comunicaciones desde la estación hacia el Objeto Inteligente B, son definidos como variables en el archivo de configuración JSON del Objeto. Esto expone los datos del Objeto como un servicio, para toda la LAN (Local Area Network) de Objetos Inteligentes conectados al Gateway, los cuales, a través de la generación de ECAS (Evento – Condición - Acción) interactúan para generar nuevos servicios y aplicaciones inteligentes.

El siguiente código fuente en Python permite la lectura de los datos de la estación climatológica conectada vía USB al Objeto Inteligente B y los expone a través del puerto serie de comunicaciones del Objeto.

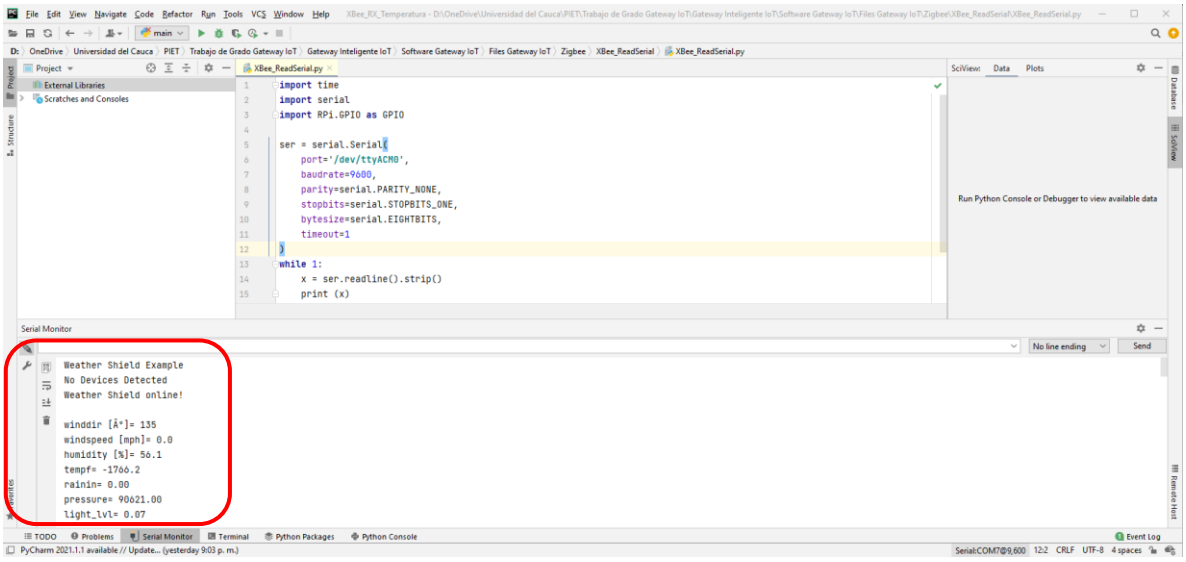

Figura 80. Código fuente lectura de datos climatológicos desde el puerto serie.

Fuente: Elaboración propia.

El código fuente de la Figura 80 es el mismo utilizado para la lectura de datos en la comunicación Zigbee, sólo que cambia el puerto serie de comunicación. En el recuadro de color rojo se pueden observar los datos climatológicos enviados por los sensores de viento

y lluvia, así como los sensores integrados en el Shield Clima Sparkfun. Cabe aclarar que estos datos son accesibles a través del protocolo IP, una vez los Objetos Inteligentes tengan implementa la arquitectura [9], lo cual soluciona el problema de interoperabilidad entre dispositivos.

La Figura 81, muestra los dispositivos IP conectados a la red local que provee la conexión de Internet al Gateway Inteligente IoT. Como se puede observar en la columna **Nombre** y **Fabricante** de los dispositivos conectados, el Gateway permite la conexión de cualquier dispositivo con una interfaz de red que implemente el protocolo IP, validando así el funcionamiento de los protocolos Wifi y Ethernet definidos en los objetivos del proyecto.

En las conexiones de red establecidas por los 12 dispositivos listados por el escáner de red, se puede observar que 3 de ellos corresponden dispositivos Raspberry Pi, dos de ellos configurados como objetos inteligentes, y el otro como Gateway Inteligente IoT.

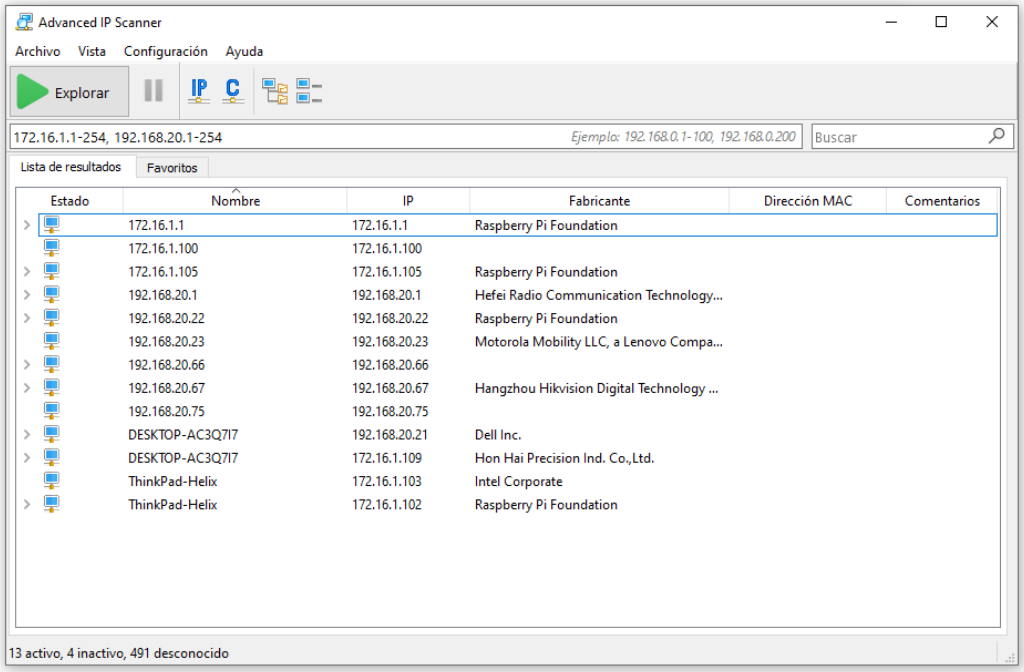

Figura 81. Dispositivos IP conectados al Gateway.

Fuente: Elaboración propia.

En la Figura 82 se puede observar que los dispositivos con las direcciones IP **172.16.1.105** y **172.16.1.102** ejecutan el servidor FTP **vsftpd**, sobre el cual se implementa la Arquitectura de Objeto Inteligente, lo cual los diferencia de los demás dispositivos como computadores portátiles, de escritorios y teléfonos inteligentes conectados al Gateway.

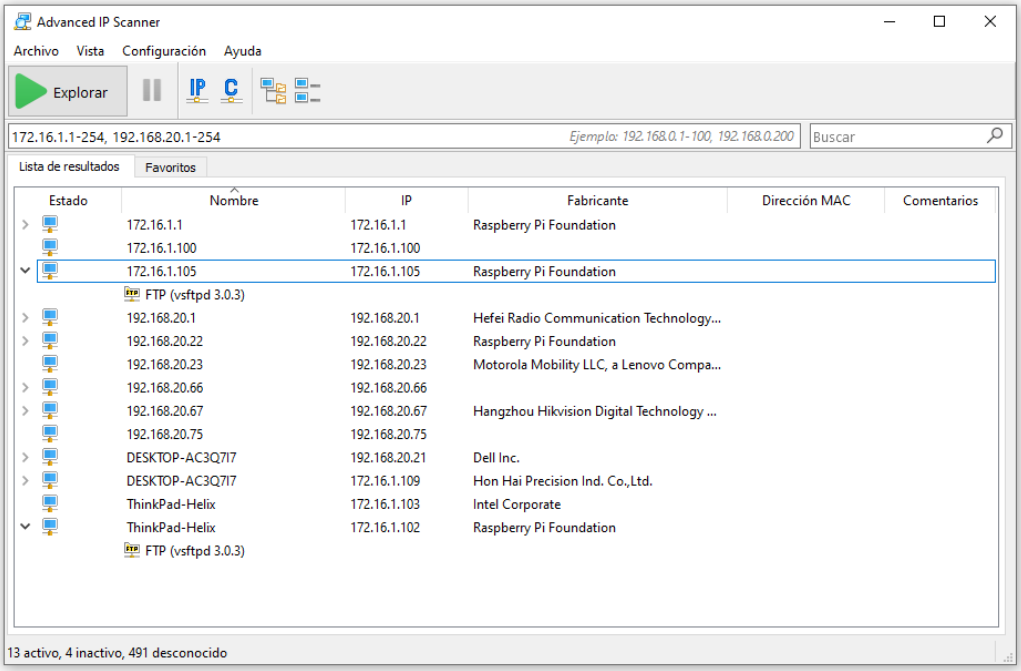

Figura 82. Objetos Inteligentes corriendo el servidor FTP.

Fuente: Elaboración propia.

Con lo descrito anteriormente, se lleva a cabo el descubrimiento general de los dispositivos de red conectados al Gateway, validando así su funcionamiento como un Router convencional, capaz de proveer conectividad local (LAN), a otras redes (WAN) y a Internet (Cloud) para los diferentes dispositivos a él.

5.2. DESCUBRIMIENTO DE OBJETOS INTELIGENTES

El descubrimiento de Objetos Inteligentes es una funcionalidad del Gateway Inteligente IoT. Este proceso, permite descubrir qué dispositivos conectados al Gateway ejecutan la Arquitectura de Objeto Inteligente, por medio de un script de Python desarrollado para tal fin.

El código fuente incluido en el **ANEXO E: CÓDIGO FUENTE DESCUBRIMIENTO DE OBJETOS INTELIGENTES**, establece el rango de direcciones a explorar, de acuerdo al segmento de red establecido en los parámetros de la Interfaz web de Kura para los dos adaptadores de red que crean la LAN del Gateway.

Se exploraron alternativas de direccionamiento IP con el objetivo de evaluar el comportamiento del protocolo DHCP en las diferentes interfaces del Gateway. En un primer caso, todos los dispositivos conectados vía Wifi, están configurados en la dirección de red local **172.16.1._**, y los dispositivos conectados vía Ethernet por la interfaz de red **eth1** toman direcciones IP privadas en el segmento **172.17.1._**. Por ello, se realiza la exploración de ambas redes con un ciclo **for** como el siguiente:

```
for ip in range(100,106):

   ipAddress = '172.16.1.'+str(ip)
```

La comprobación de, si un dispositivo conectado al Gateway es o no un Objeto Inteligente, se lleva acabo con la instrucción:

```
res = requests.get('http://''%s' %(stdout.split()[1])+'/Identificator')
```

En ésta, se realiza una petición HTTP con la dirección IP que actualmente se esté generando en el ciclo **for** y se verifica la respuesta obtenida por la **URL** (Dirección IP concatenada con el identificador del Objeto **/Identificator**).

Al momento de ejecutar la funcionalidad de descubrimiento de Objetos Inteligentes, se puede observar en la Figura 83 que, de seis dispositivos conectados al Gateway, dos de ellos son Objetos Inteligentes.

Cabe aclarar, que las direcciones IP privadas mostradas en las diferentes figuras del presente documento, no siempre corresponderán a las mismas a pesar de haber usado en todas ellas los mismos dispositivos. Lo anterior dado que, los dispositivos se conectan al Gateway por medio de DHCP y no se estableció un direccionamiento estático, lo cual, permite además probar y validar el correcto funcionamiento del Gateway a la hora de direccionar y dar de alta la conectividad a nuevos dispositivos que se conecten a él.

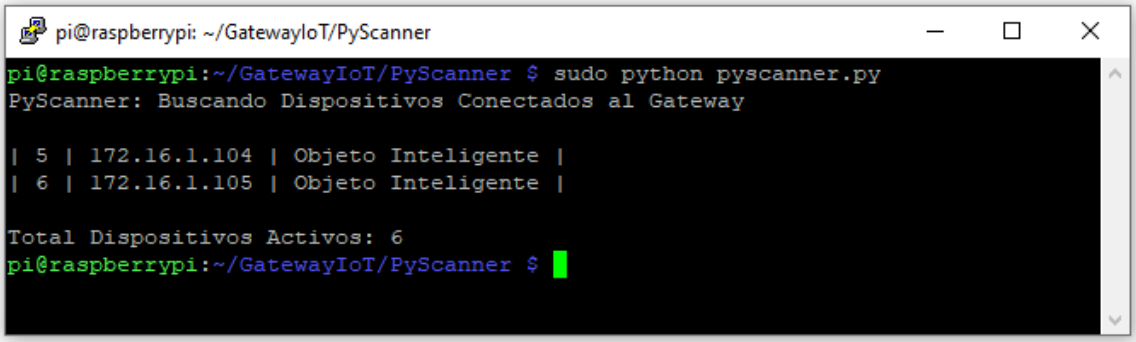

Figura 83. Servicio de Descubrimiento de Objetos Inteligentes.

Fuente: Elaboración propia.

El resultado ilustrado en la Figura 83 se obtuvo, configurando las interfaz **wlan0** y **eth1** con direcciones IP del segmento **172.16.1._**.

Una vez se ejecuta la funcionalidad de descubrimiento de Objetos Inteligentes, se genera un archivo con el listado de Objetos conectados, como se ilustra en la Figura 84. Este archivo equivale a la **Lista DHCP** que se obtiene al consultar los dispositivos conectados a un Router convencional.

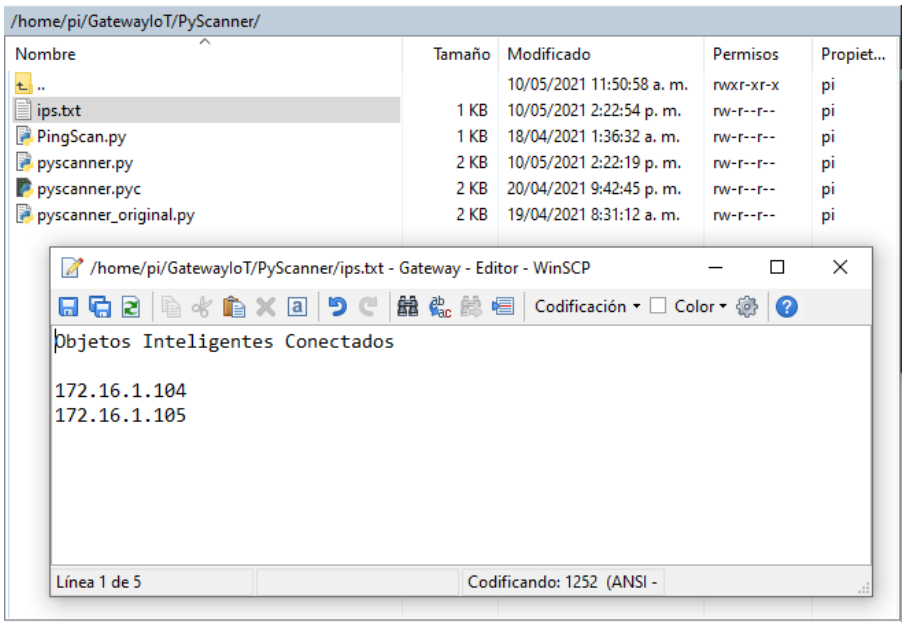

Figura 84. Archivo con lista de Objetos Inteligentes Conectados.

Fuente: Elaboración propia.

La funcionalidad de descubrimiento de Objetos Inteligentes implementada en el Gateway, posibilita la implementación de servicios e interacciones (ECAS) entre los Objetos conectados. Aquellos dispositivos que estén ejecutando la Arquitectura de Objeto Inteligente, podrán ser descubiertos por el Gateway al ejecutar esta funcionalidad.

Al tener diferentes tipos de dispositivos conectados, el Gateway crea una red LAN de esos dispositivos. Una vez listados los objetos inteligentes conectados gracias a la funcionalidad de descubrimiento, se podrá realizar la configuración de los mismos, indicando su dirección IP local en los archivos de configuración de la arquitectura, de modos que se les permitirá establecer conexión y comunicación con otros objetos por medio del Gateway.

Partiendo de la funcionalidad de descubrimiento de Objetos Inteligentes, el uso del Gateway desarrollado para el despliegue de escenarios de interacción semántica, constituye una mejora preliminar respecto a la necesidad de contar con un índice de objetos inteligentes remoto alojado en un servidor. Aunque, en el presente desarrollo se logró obtener una lista de las direcciones IP de los objetos inteligentes conectados al Gateway, aún es necesario hacer uso del índice semántico para realizar una implementación real y completa de un escenario, ya que por medio del índice se obtiene toda la información de los objetos y sus recursos. Se requiere entonces, mejorar esta funcionalidad del Gateway en futuros proyectos, de modo que se pueda obtener el identificador de los diferentes objetos y sus recursos asociados.

5.3. MONITOREO Y RENDIMIENTO DEL GATEWAY

Para monitorear el estado del hardware y rendimiento del Gateway Inteligente IoT, se utilizó el software **RPi-Monitor**[38]**. El ANEXO F: INSTALACIÓN DE RPI-MONITOR EN LA RASPBERRY PI,** muestra el procedimiento de instalación de la herramienta RPi-Monitor.

La herramienta RPi-Monitor facilitó el monitoreo del estado y rendimiento del hardware sobre el cual se desarrolló el Gateway Inteligente IoT, con el objetivo de evaluar su comportamiento como pasarela de conectividad local y conexión a Internet de los diferentes dispositivos de red conectados a él.

La Figura 85 muestra la interfaz principal de RPi-Monitor ejecutada desde el navegador web en el Gateway. La variables monitoreadas fueron: el tiempo de funcionamiento, estado de la CPU, temperatura del SoC (System on Chip), cantidad de memoria usada, cantidad de memoria de intercambio (Swap en Linux), el espacio ocupado y libre en la tarjeta de almacenamiento sobre la cual se instaló el sistema operativo, la cantidad de datos [Mb] enviados y recibidos por los adaptadores Ethernet y Wifi.

Los valores de las variables medidas por el RPi-Monitor fueron tomados para un tiempo de funcionamiento del Gateway de 9 horas, ejecutando las tareas y servicios básicos del sistema operativo Raspberry Pi OS, el Framework Kura y las conexiones con los dispositivos y Objetos Inteligentes desplegados. Aunque cabe aclarar que, durante el desarrollo de este proyecto, siempre se tuvo funcionando la Raspberry Pi como Gateway, brindando conexión a Internet a dos computadores (uno de escritorio y otro portátil), un Smart TV y un teléfono inteligente durante más de 12 horas todos los días de la semana, en un periodo de más de 9 meses. A partir de esta experimentación se puede concluir que, es confiable tener la placa Raspberry Pi 3 Modelo B V1.2 configurada como Router convencional y Gateway Inteligente para IoT haciendo uso del Framework Kura.

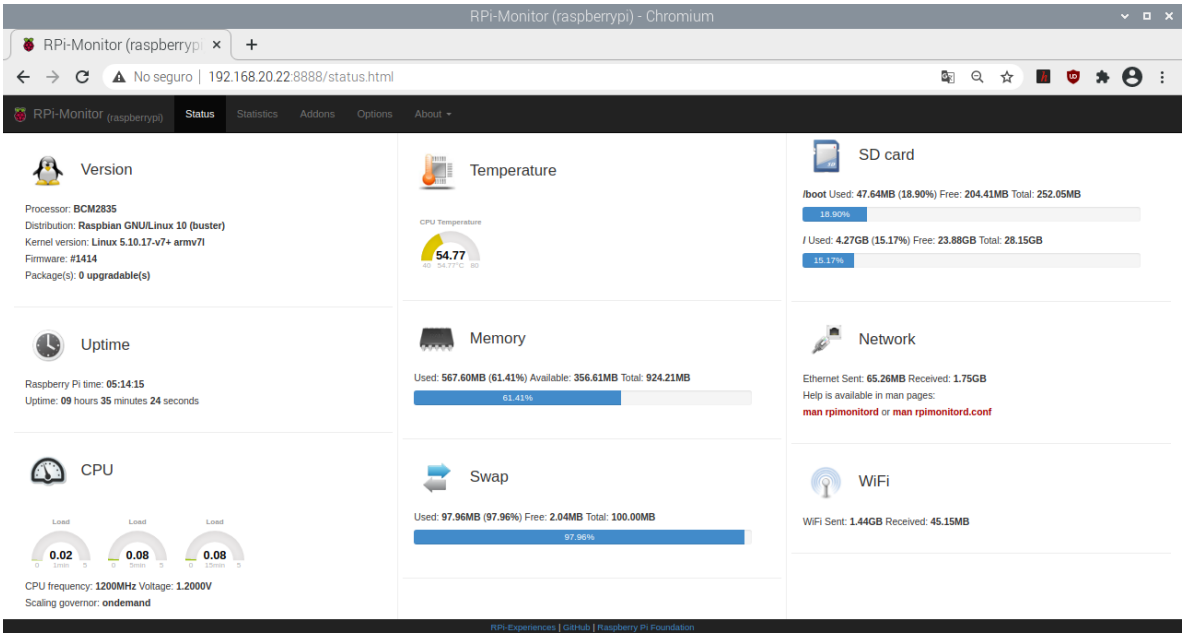

Figura 85. Interfaz principal RPi-Monitor.

Fuente: Elaboración propia.

[38] RPi-Experiences: RPi-Monitor Overview, consultado 30/11/2020.

Se evaluó el comportamiento de la CPU, la memoria y la temperatura del SoC en la Raspberry Pi incrementando la demanda de datos transmitidos a través de los adaptadores de red configurados. Para ello, se cargó un vídeo en alta definición en YouTube desde el Gateway y como resultado, evidentemente se genera un incremento en los datos enviados y recibidos vía Ethernet hacia Internet, así como también un incremento en la memoria usada, que ascendió al 80,82% (en condiciones normales descritas anteriormente 61,41%), la temperatura del SoC ascendió de 54,77 °C a 59,07 °C y el promedio de carga de la CPU ascendió de 0,08 a 1,37 en una escala de 0 a 5 durante los 5 minutos en que fueron tomados los valores monitoreados. La Figura 86 muestra los valores monitoreados durante la carga del vídeo en YouTube usando la conexión a Internet del Gateway por medio del adaptador Ethernet integrado en la Raspberry Pi.

Se considera importante esta prueba, ya que al cargar un video directamente desde el Gateway, se puede observar qué variables del hardware sufren variación en sus valores normales de funcionamiento. Con los resultados de esta prueba, se puede determinar el posible comportamiento del Gateway al ejecutar un servicios donde por ejemplo se requiera monitorear un cultivo por medio de una cámara de video o el envío y recepción de contenido multimedia (imágenes, audio y/o video), que está dentro de los posibles escenarios de aplicación del Gateway.

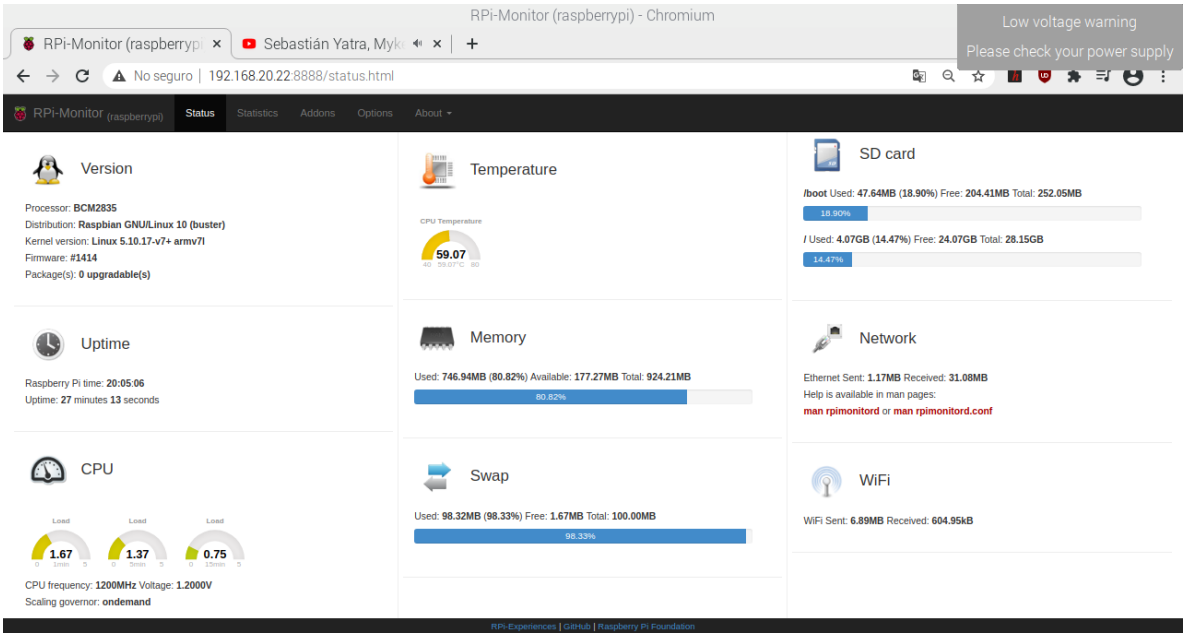

Figura 86. Monitoreo de variables bajo demanda de recursos multimedia.

Fuente: Elaboración propia.

Las gráficas mostradas desde la Figura 87 hasta la Figura 90 corresponden a las variables monitoreadas por RPi-Monitor representadas de manera individual dentro de las opciones de estadística.

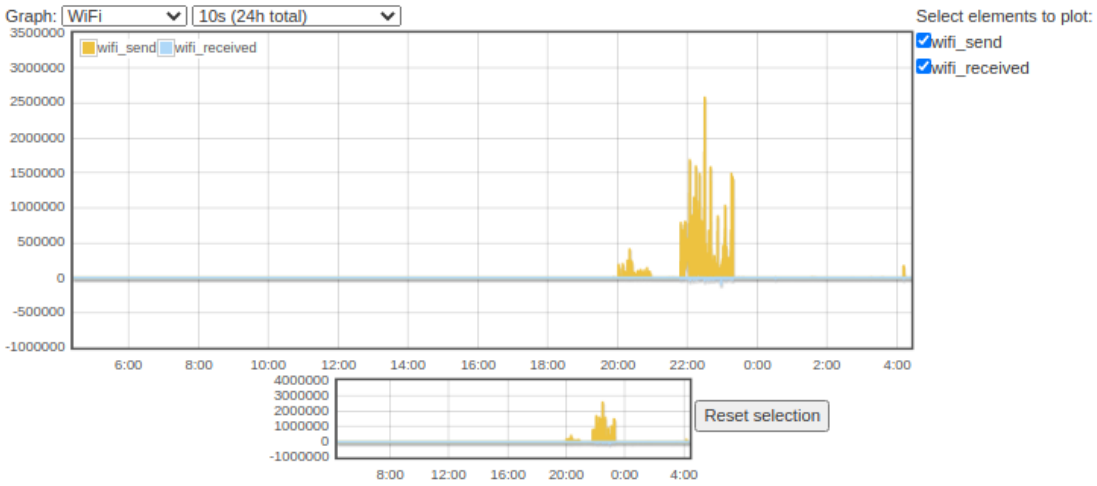

Figura 87. Gráfica de estado de conexión WiFi.

Fuente: Elaboración propia.

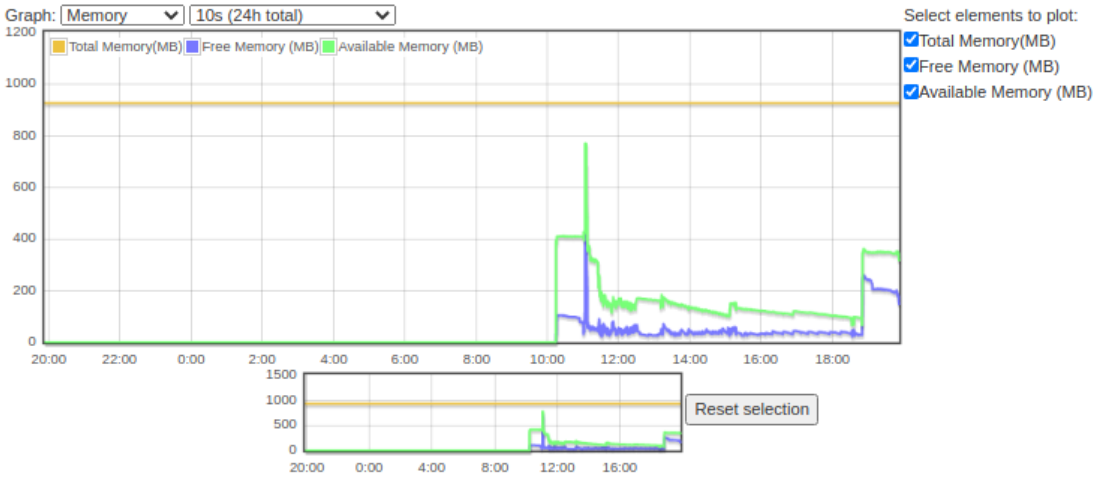

Figura 88. Gráfica de estado de la memoria RAM.

Fuente: Elaboración propia.

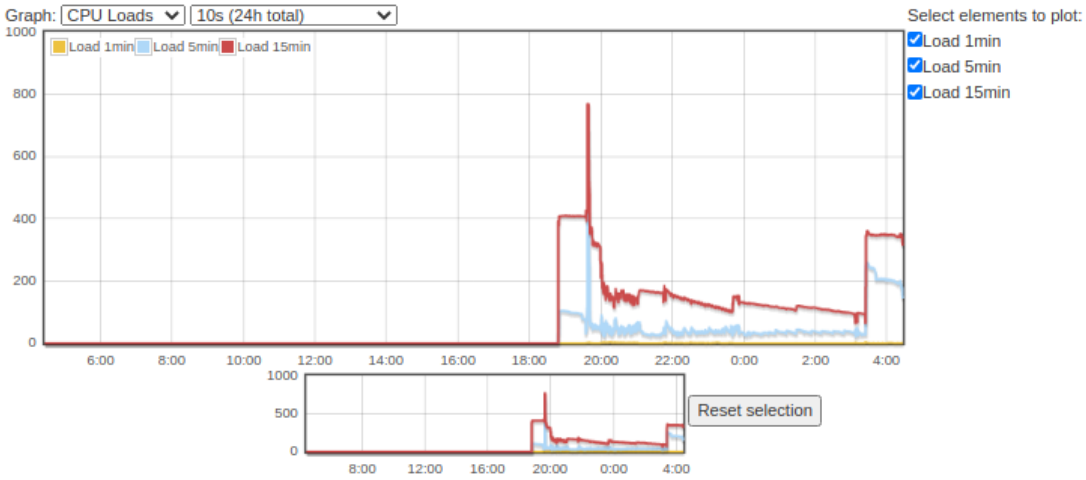

Figura 89. Gráfica del estado de carga promedio de la CPU.

Fuente: Elaboración propia.

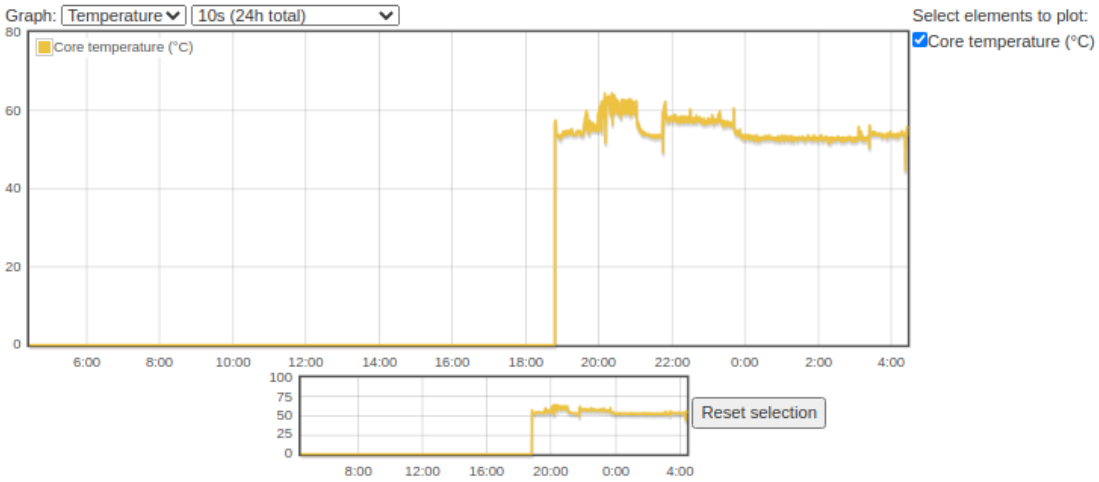

Figura 90. Gráfica de temperatura del SoC.

Fuente: Elaboración propia.

5.3.1. Pruebas para evaluar la velocidad de conexión a Internet

Dentro de las pruebas realizadas para evaluar el funcionamiento del Gateway, también se incluyeron test de velocidad de conexión a Internet, tanto del mismo Gateway, como de los equipos y Objetos Inteligentes conectados a él.

Se consideró importante la prueba de velocidad de conexión a internet, dado que el Gateway desarrollado puede ser utilziado en diferentes escenarios y puede demandar diferentes recursos a nivel de conexiones a internet. Un servicio que podría demandarse en el Gateway, es por ejemplo, el consumo en la red de contenido multimedia (audio y/o videos) en aplicaciones de agricultura inteligente donde cuente con una cámara de video para el monitoreo de los cultivos. Se toma como referencia este servicio, ya que el consumo

en recursos de la red es mucho más alto que los datos transmitidos por parte de los objetos inteligentes utilizados para este trabajo.

Se toma como referencia para las mediciones la velocidad del enlace principal que provee conexión de Internet al Gateway. Se trata de un plan de Internet de 100Mb que llega vía coaxial al predio donde se instalaron los equipos y lo provee la compañía Claro Colombia.

La Figura 91 muestra el test de velocidad realizado desde un computador de escritorio conectado por Ethernet al Router principal de la compañía Claro.

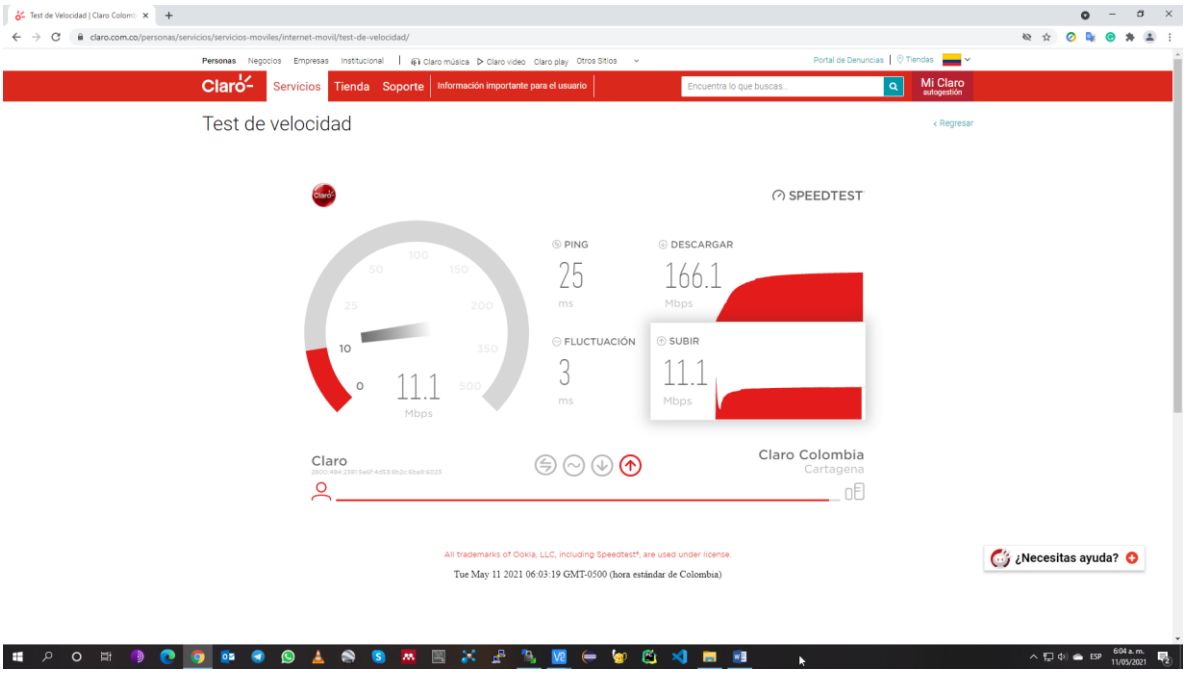

Figura 91. Prueba de velocidad de conexión a Internet computador de escritorio vía Ethernet.

Fuente: Elaboración propia.

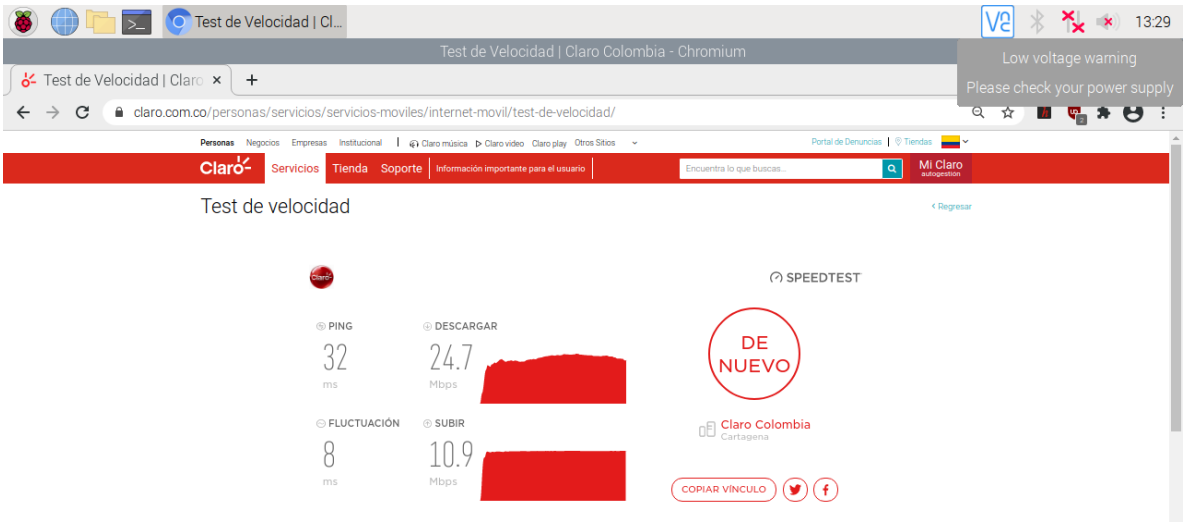

Figura 92. Velocidad de conexión a internet del Gateway Inteligente IoT vía Ethernet.

Fuente: Elaboración propia.

Se realizó también un test de velocidad a cada uno de los dos Objetos Inteligentes conectados al Gateway. Esto para validar, primero, que ambos Objetos cuentan con conexión a Internet para realizar una navegación independiente a los servicios propios del escenario de interacción semántica, y segundo, evaluar el rendimiento del enlace de red inalámbrico y cableado en los Objetos. Comparando los valores mostrados en la Figura 93 y la Figura 94, se evidencia que ambas conexiones de red, tanto Wifi como Ethernet, tiene velocidades de conexión con valores muy similares y acordes a los valores brindados por el Gateway, velocidad de descarga +/- 30Mbps y velocidad de subida +/- 11Mbps para conexión a Internet.

Figura 93. Prueba de velocidad de conexión a Internet Objeto Inteligente A vía Wifi.

Fuente: Elaboración propia.

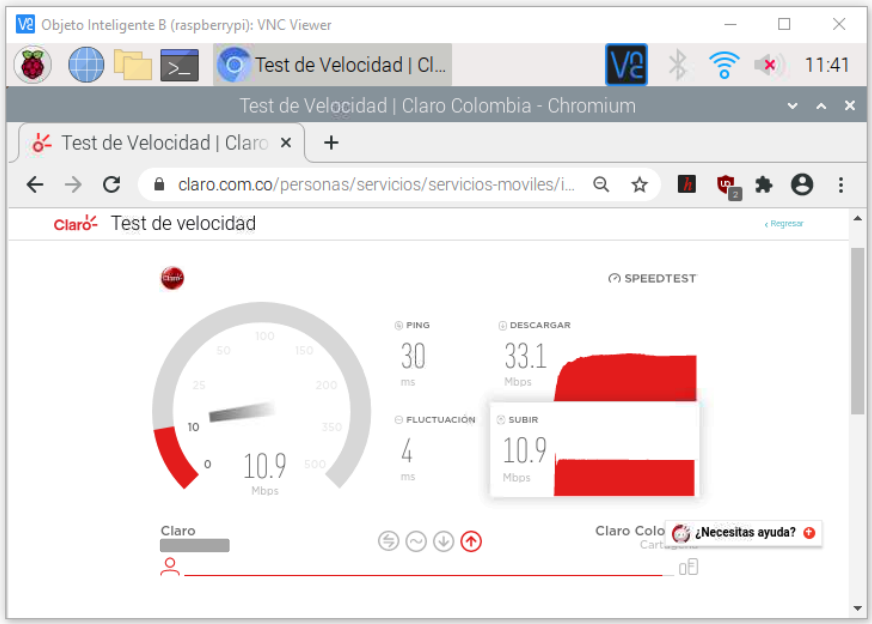

Figura 94. Prueba de velocidad de conexión a Internet Objeto Inteligente B vía Ethernet.

Fuente: Elaboración propia.

5.4. INTERFACES VIRTUALES PARA DISPOSITIVOS NO SMART THING | DOCKER

Una de las funcionalidades bien interesantes y contempladas durante el desarrollo de la arquitectura del Gateway Inteligente IoT en el numeral 3.1.3.3 del presente proyecto, es la virtualización de dispositivos que no son Smart Things, esto es, dispositivos IoT, sensores o actuadores que no tiene capacidad de implementar la arquitectura de Objeto Inteligente. Al realizar la virtualización de estos dispositivos, el Gateway permite gestionar su conectividad y comunicación con otros dispositivos y el mismo Gateway a través del protocolo IP.

Para la virtualización de dispositivos IoT, sensores y/o actuadores, así como también alguna variable digital o analógica de otro sistema o dispositivo, se utilizó la herramienta Docker.

El objetivo en este numeral es lograr generar una nueva interfaz virtual de red en el Framework de Kura, para un dispositivo No Smart Thing, donde la imagen virtual creada para dicho dispositivo permita la implementación tanto de la arquitectura como del desarrollo software necesario para la necesidad específica requerida. De esta forma, podemos evaluar el funcionamiento de la capa de virtualización propuesta en la arquitectura del Gateway.

Como primer paso, realizamos la instalación de Docker, para ello, seguiremos el procedimiento descrito en la página oficial https://get.docker.com/. Aquí un resumen de los comandos utilizados para la instalación de Docker en la Raspberry Pi:

```
pi@raspberrypi : ~ $ curl -fsSL https://get.docker.com -o get-docker.sh
```

```
pi@raspberrypi : ~ $ sh get-docker.sh
```

La Figura 95 muestra el resultado de la ejecución del script de instalación de Docker en la terminal de la Raspberry Pi.

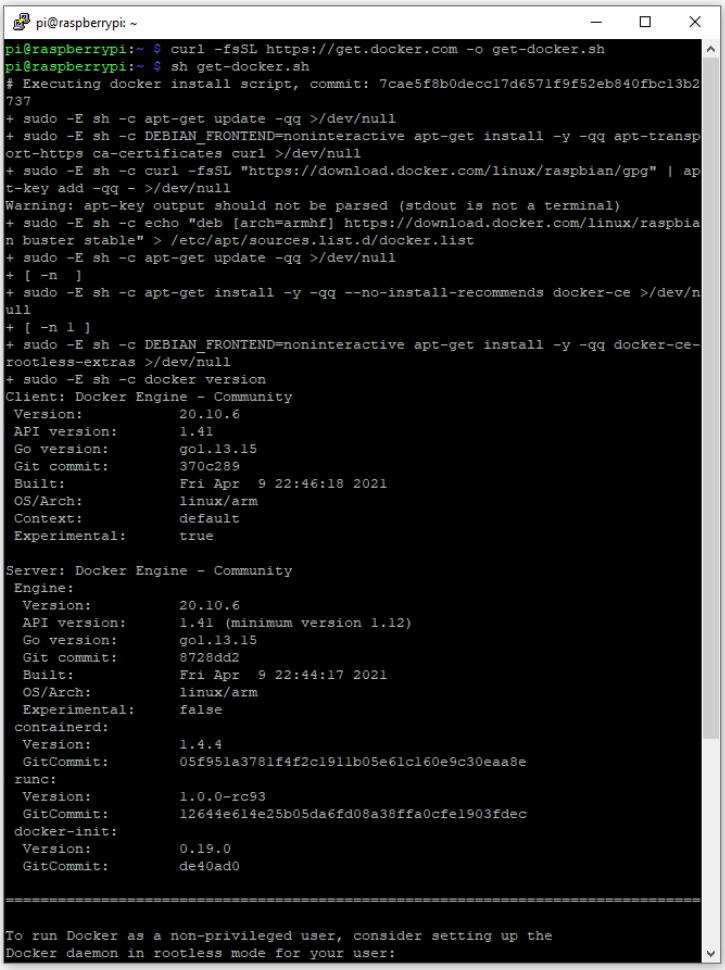

Figura 95. Ejecución script para instalación de Docker.

Fuente: Elaboración propia.

Ahora, ejecutamos el siguiente comando para permitirle a Docker ejecutarse sin el comando **sudo**:

```
pi@raspberrypi : ~ $ sudo usermod –aG docker pi
```

Finalmente, comprobamos que Docker se haya instalado correctamente, para ello, ejecutamos el comando **docker version** desde la terminal, y debemos obtener un resultado como el de la Figura 96.

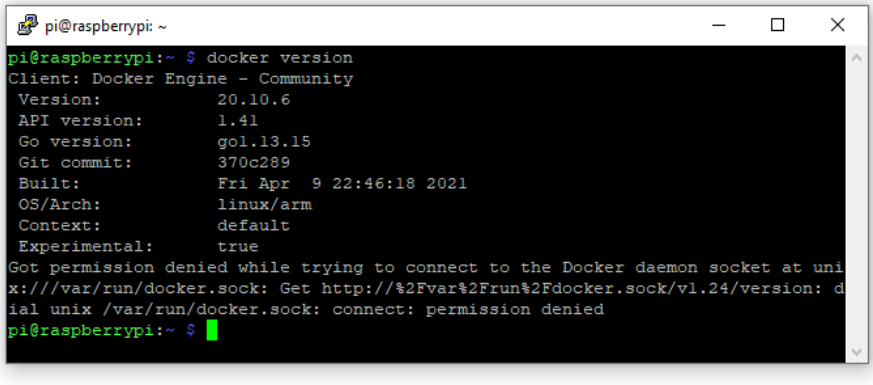

Figura 96. Versión de Docker instalada.

Fuente: Elaboración propia.

Cuando la instalación de Docker ha terminado, encontraremos en la interfaz web de configuración de Kura u nuevo adaptador de red virtual que corresponde al contenedor de aplicaciones. En la Figura 97 se muestra la configuración IP realizada al adaptador virtual **docker0** y que permite conectar vía protocolo IP con la dirección fijada, todos los servicios, aplicaciones y programas que se ejecuten en el contenedor.

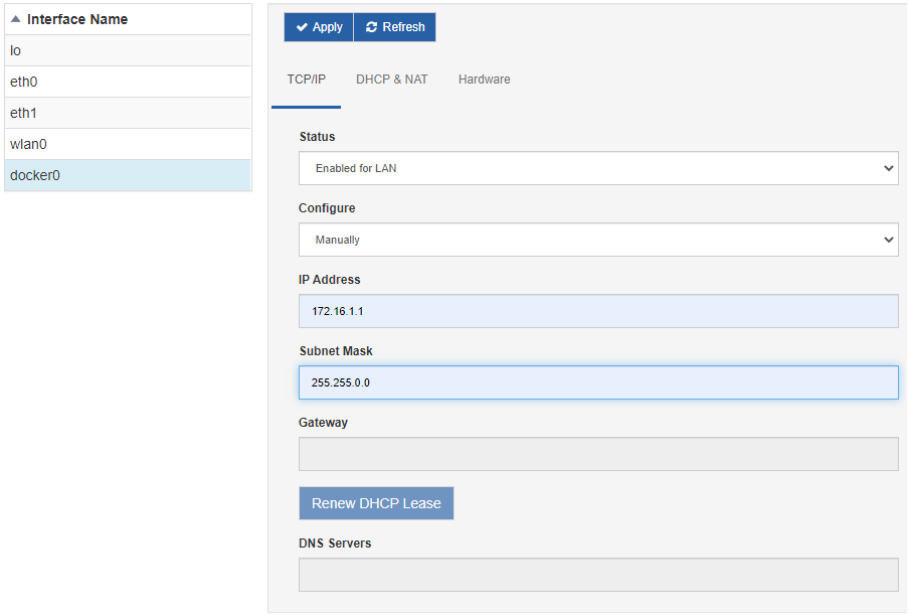

Figura 97. Configuración IP de interfaz virtual Docker desde Kura.

Fuente: Elaboración propia.

5.5. GESTIÓN DE DISPOSITIVOS Y RECURSOS IOT DESDE EL GATEWAY

La implementación que se realiza en este numeral, muestra una de las funciones más importantes que realiza el Gateway desarrollado en escenarios de IoT y que se muestra en la arquitectura propuesta, numeral 3.1.3.2. Esta función consiste en, poder tomar cualquier recurso o dispositivo actuador, sensor y/o variable física medida que se requiera integrar a

un sistema de IoT, conectarlo directamente a los pines de propósito general del Gateway (GPIO), y a partir de una representación virtual (imagen Docker y desarrollo de los componentes software que permitan su integración con el Gateway), implementar en este recurso virtual la Arquitectura de Objetos Inteligentes que permita la interacción del mismo con otros Objetos conectado. Esto es, conectar dispositivos que no tiene capacidad de procesamiento de información ni de comunicación.

Supone una ventaja entonces, poder utilizar el Gateway para gestionar dispositivos que no son inteligentes y que tampoco tienen capacidad de ejecutar un sistema operativo y por ende, tampoco ejecutarían la arquitectura de objeto inteligente. Es claro que esta funcionalidad no la permite un router convencional, por lo que el Gateway desarrollado constituye un primer avance importante en la integración de pasarelas para IoT en escenarios de interacción de objetos inteligentes.

Cabe aclarar, que dentro de los resultados obtenidos en este proyecto, no se realizó la virtualización del recurso conectado al GPIO del Gateway, lo cual constituye un trabajo futuro interesante y que daría continuidad a este desarrollo.

El Gateway Inteligente IoT desarrollado en este proyecto, da solución al problema de heterogeneidad e integración de dispositivos, permitiendo que cualquier dispositivo que cuente con conexiones físicas al GPIO o puertos del Gateway, pueda comunicar sus datos y publicarlos en una plataforma para IoT o integrarlos a soluciones IoT de más alto nivel. El recurso conectado al Gateway, se logró gestionar hasta obtener sus datos de humedad y temperatura medidos del ambiente, para ser publicados en la plataforma AWS IoT Core. Lo cual permitió la validar el funcionamiento del Gateway como integrador de dispositivos heterogéneos con plataformas para IoT.

En una primera instancia se conectó un diodo led al GPIO del Gateway para gestionar su accionamiento directamente desde su interfaz principal.

Posteriormente, se conecta el Sensor de Temperatura y Humedad DHT22 V2 que permite enviar valores capturados del ambiente. Estos valores son recibidos por el Gateway directamente desde el GPIO, el cual, se configura a través Kura utilizando los **Drivers and Assets** y se programa por medio del servicio **Wire Graph**.

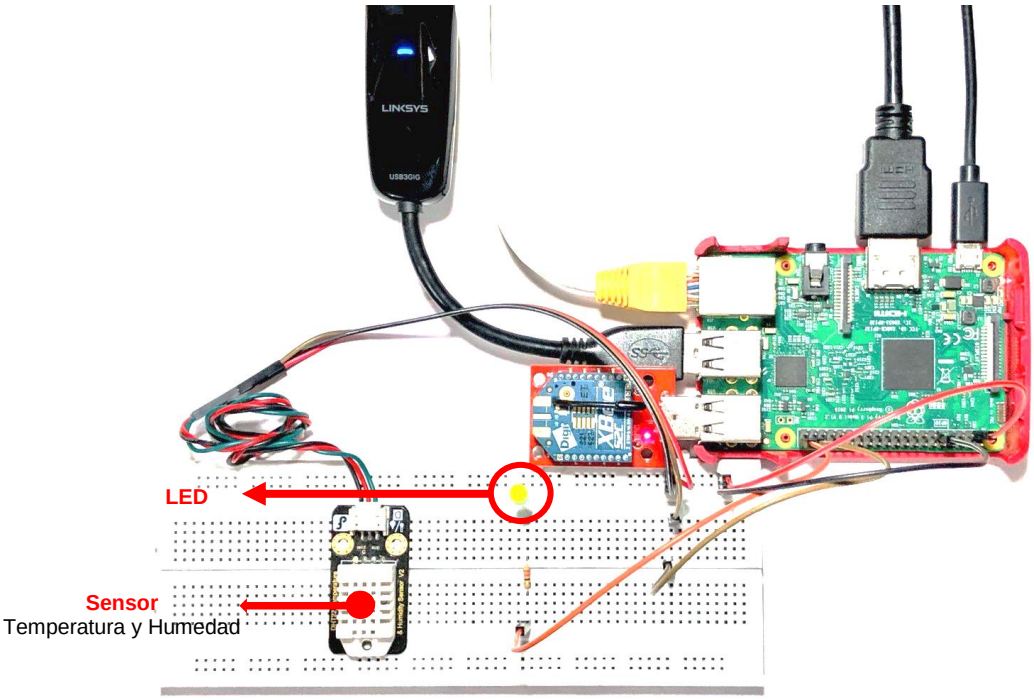

Figura 98. Gestión GPIO y sensores a través de “Drivers and Assets” de Kura.

Fuente: Elaboración propia.

En la Figura 98 se ilustra la conexión del LED y el sensor de temperatura y humedad al Gateway. Se procede a instalar el “**GPIO Driver for Eclipse Kura 4.x.y**” desde el Marketplace[39] de Eclipse.

El siguiente es el código fuente que permite la lectura de humedad y temperatura del sensor DHT22

```
import Adafruit_DHT as dht
from time import sleep
#Set DATA pin
DHT = 20
while True:
    #Read Temp and Hum from DHT22
    h,t = dht.read_retry(dht.DHT22, DHT)
    #Print Temperature and Humidity on Shell window
    print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(t,h))
    sleep(5) #Wait 5 seconds and read again
```

Posteriormente, desde Wire Graph de la interfaz Web de Eclipse Kura, se establecen las conexiones entre la información recibida del CloudSuscriber el GPIOAsset del LED.

La Figura 99 y Figura 100, muestran la configuración y programación del GPIO a través de Kura.

[39] https://marketplace.eclipse.org/search/site/gpio?f%5B0%5D=im_taxonomy_vocabulary_1%3A4397&f%5B1%5D=im_taxonomy_vocabulary_3%3A4396, consultado 01/02/2021.

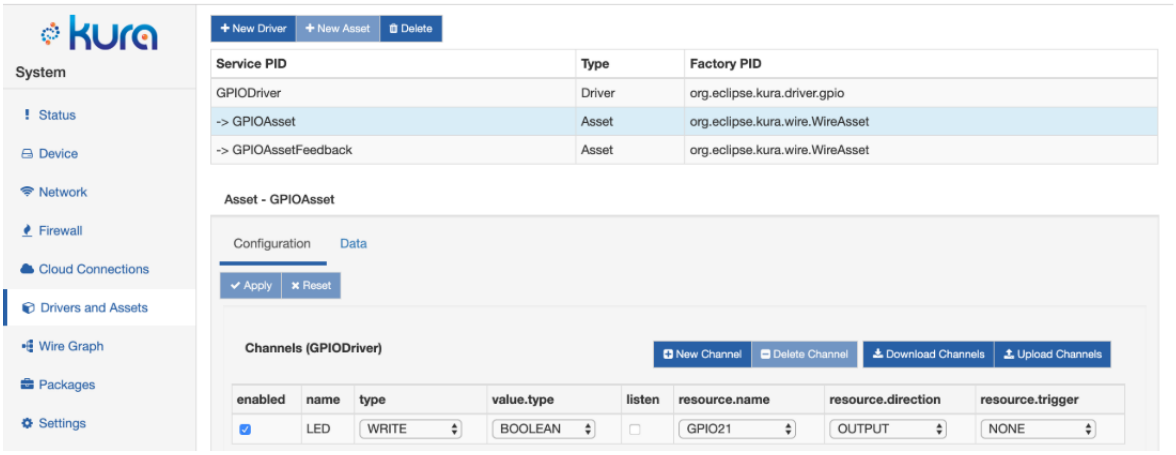

Figura 99. Configuración de GPIO desde Kura Drivers and Assets

Fuente: Elaboración propia.

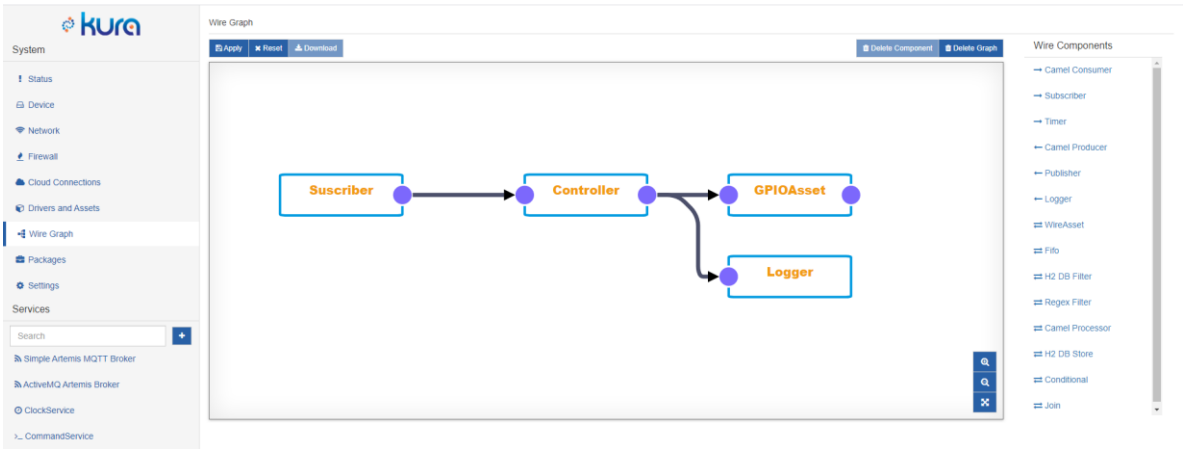

Figura 100. Programación GPIO desde Wire Graph de Kura.

Fuente: Elaboración propia.

Para que el LED se encienda o apague dependiendo de la información obtenida en la suscripción, será necesario un script que sirva como controlador. Para ello en Wire Components seleccionamos **Javascript Filter**.

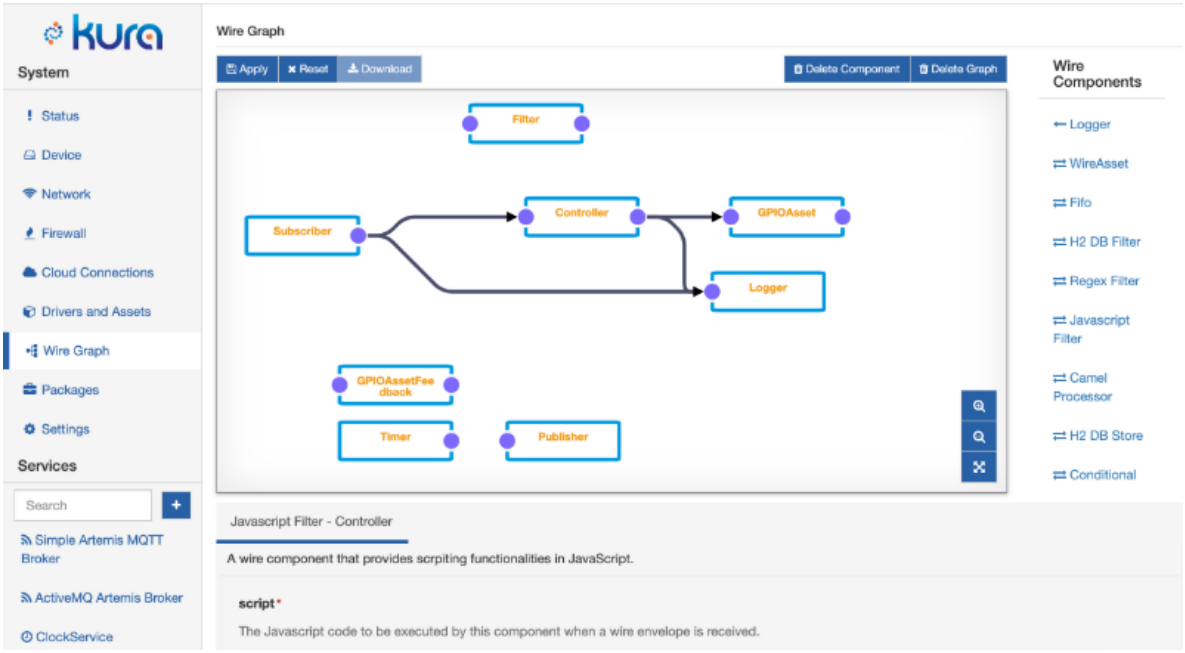

Figura 101. LED y Sensor de Temperatura con servicio CloudService de Kura.

Fuente: Elaboración propia.

La Figura 101 Ilustra la **Wire Graph** generada para publicar en la Plataforma para IoT (AWS IoT Core) los datos del sensor de temperatura y humedad y la gestión del LED. La entrada como se puede ver, es la información del CloudSuscriber y tendrá como salida una conexión con el GPIOAsset que hemos configurado anteriormente. Además, se conectará con el servicio Logger para escribir en el fichero /var/log/kura.log toda la información recibida y enviada.

En la documentación oficial de Eclipse Kura se detalla el procedimiento para la implementación del Bundle[40].

El siguiente es el código fuente en Javascript utilizado en el componente **Javascript Filter**.

```
var record = input.record[0] //Obtiene información del Suscriber
for (var prop in record){
    if (record[prop].getType()==BOOLEAN){ //Busca campo información
booleana
        logger.info('{}: {}', prop record[prop])
        var counterRecord = newWireRecord() //Flujo de salida
        counterRecord.LED = record[prop] //Se asigna el valor al LED
        output.add(counterRecord) //Fija valor en el flujo de salida
    }
}
```

Una vez se apliquen los cambios en la configuración del Wire, se tiene los valores de humedad y temperatura en AWS IoT, así como también el estado del LED. La Figura 102 ilustra los mensajes MQTT recibidos en la Plataforma IoT.

[40] http://eclipse.github.io/kura/dev/deploying-bundles.html, consultado 30/11/2020.

Figura 102. Mensaje MQTT.

Fuente: Elaboración propia.

En este numeral, se mostró de manera experimental, que es posible conectar directamente a los pines GPIO del Gateway cualquier sensor, actuador, dispositivo o variable digital o analógica. Además, en el numeral anterior 5.4, también se evidenció de manera experimental, la creación de interfaces virtuales de red que permiten conectar vía protocolo IP cualquier aplicación, código fuente o programa, incluso otro sistema operativo, por medio de la herramienta Docker en la interfaz principal de Kura. Lo cual nos muestra entonces, que es factible desarrollar el software en la imagen virtual de Docker que gestione el dispositivo conectado al GPIO, implemente en esta misma imagen la arquitectura de Objeto Inteligente y posteriormente, el Gateway será capaz de brindar conectividad y comunicación con los demás recursos que tenga conectados. Esto abre un amplio espectro de posibles soluciones, integraciones y nuevos servicios inteligentes en contextos variados de IoT.

Cabe alcarar que, en este trabajo, no se alcanzó a realizar la integración de los objetos virtuales propuestos en la arquitectura del Gateway desarrollado con la Arquitectura de Interacción Semántica de Objetos Inteligentes. Lo que si se obtuvo, fue la ejecución de una imagen virtual Docker y su respectiva configuración de las interfaces de red desde la interfaz de configuración de Kura. Lo que deja habilitada la funcionalidad de poder tomar los desarrollos que se hagan en el contenedor de aplicaciones e integrarlos con los servicios que se desplegaron en el Gateway.

CAPÍTULO 6 CONCLUSIONES Y TRABAJOS FUTUROS

Como punto final a este trabajo de grado, se reflexiona sobre los conocimientos adquiridos, la funcionalidad conseguida en la solución desarrollada, así como la capacidad de mejora y trabajos a futuro en este tema de investigación.

Desde el punto de vista del aprendizaje, se ha construido una arquitectura genérica para la implementación de Gateways Inteligentes IoT en escenarios de interacción semántica con objetos inteligentes y contextos de aplicación variados.

Se ha obtenido un modelo de Gateway Inteligente IoT y su desarrollo con tecnologías actuales, el cual implementa la arquitectura y concepto de objeto inteligente [9], basado en los estándares y recomendaciones de entidades internacionales de normalización como la ITU.

El Gateway Inteligente IoT es capaz de conectar dispositivos Smart Things y No Smart Things de manera heterogénea, gracias a la implementación de diferentes protocolos de comunicación como Ethernet, Wifi y Zigbee.

Se desplegó un escenario de objetos inteligentes con dispositivos reales para el contexto de aplicación del monitoreo de variables agroclimatológicas.

Se plantea como trabajo futuro, mejorar la funcionalidad de descubrimiento de Objetos Inteligentes de modo que se pueda obtener el identificador de los diferentes objetos y sus recursos asociados.

Dado que la implementación de las aplicaciones y servicios que corren sobre el Gateway Inteligente IoT desarrollado requieren un trabajo de programación especialmente en JAVA, se ve la necesidad de abordar trabajos futuros sobre el desarrollo de estos servicios para contextos de aplicaciones como el planeado en este proyecto que permitan explorar otras capacidades del Framework Eclipse Kura, ya que, son bastante amplias las funciones, configuraciones y servicios que este puede brindar.

Dadas las características de adaptabilidad, traducción de protocolos, interacción y soporte de aplicaciones, soporte de funciones de gestión y de seguridad, resulta bastante interesante poder desarrollar nuevos servicios y aplicaciones del Gateway dirigidos a las diferentes industrias de la IoT, como por ejemplo el IoT Industrial, de modo que se pueda implementar el Gateway desarrollado para la mejora de automatización de procesos industriales, ya que unos de los protocolos más usados (Modbus) en ésta área ya ha está implementado dentro de las tecnologías soportadas por el Framework Kura.

En la actualidad son muy variados los dispositivos que se pueden utilizar en aplicaciones para IoT, de esta variedad también surge una gran variedad de tecnologías y otros protocolos de comunicaciones. Por tal motivo, se propone explorar la implementación de aplicaciones y servicios sobre el Gateway Inteligente IoT utilizando tecnologías de comunicación no abordadas en este proyecto. Es el caso de la tecnología Bluetooth, LoraWAN, Modbus para aplicaciones de IoT industrial entre otros.

La plataforma para IoT explorada en este trabajo fue AWS IoT Core[41]. Se puede explora diferentes líneas de soluciones y contextos de aplicación del Gateway Inteligente IoT con otras plataformas para IoT.

Dado que el proyecto Eclipse Kura, que lidera el desarrollo e implementación del Framework para la creación de gateways IoT tiene un Marketplace donde podemos encontrar diferente drivers, aplicaciones, servicios y módulos ya desarrollados listos para instalar, se puede generar trabajos de investigación donde se evalúe la funcionalidad y rendimiento de estos paquetes que complementan las capacidades del Gateway, con el fin de generar mejoras en las versiones actuales de estos. Se evidenció en la bibliografía consultada muy pocos trabajos en este campo usando el Framework de Kura.

[41] https://aws.amazon.com/es/iot/, consultado 03/10/2020. Sitio web plataforma para IoT de Amazon Web Services.

ANEXO A: INSTALACIÓN DE ECLIPSE KURA

6.1. INTRODUCCIÓN

Eclipse Kura es un proyecto de código abierto del Eclipse IoT Working Group[42] que proporciona una plataforma para construir Gateways IoT. Kura, es un contenedor de aplicaciones inteligentes que permite la administración remota de dichos Gateways y proporciona una amplia gama de APIs para permitir la implementación de nuestras propias aplicaciones [63].

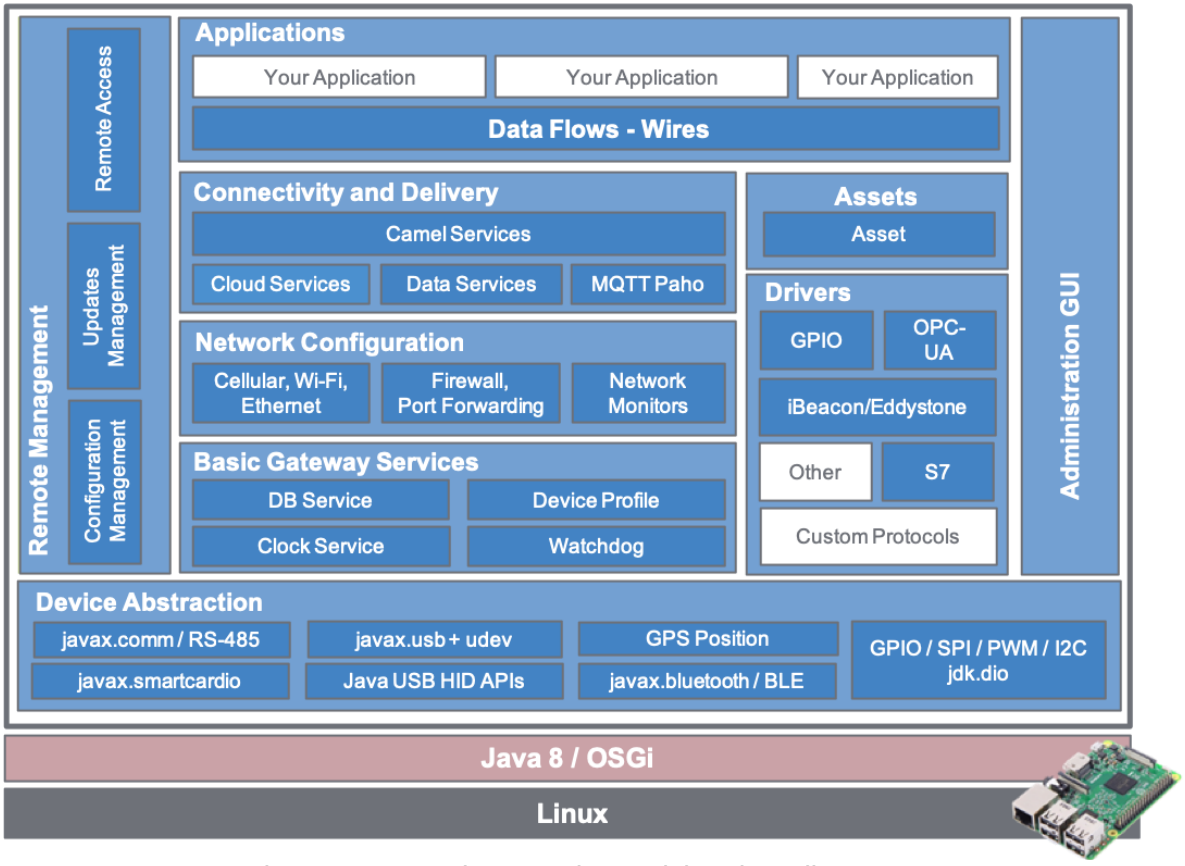

Figura 103. Arquitectura de servicios de Eclipse Kura.

Fuente: E. Foundation, "Eclipse Kura - Open Source framework for IoT". [En línea]. Disponible en: http://www.eclipse.org/kura/. [Consultado: 21-may-2018].

6.3. SERVICIOS DE ECLIPSE KURA

Eclipse Kura implementa los siguientes servicios:

6.3.1. I/O Services

[42] https://www.eclipse.org/org/workinggroups/iotwg_charter.php, consultado: 26/07/2019. Eclipse IoT Working Group es un grupo de trabajo de la Fundación Eclipse. El objetivo del grupo de trabajo de Eclipse IoT es fomentar, desarrollar y promover soluciones de código abierto, tales como herramientas de desarrollo, incluyendo simuladores y emuladores, arquitecturas de referencia y modelo de programación, sistemas e interfaces de comunicación, protocolos de comunicación abiertos y estándar, APIs y servicios para Gateways IoT y plataformas IoT Cloud.

Acceso al puerto serie a través de la API javax.comm 2.0 o la conexión de E/S OSGi.
Acceso USB y eventos a través de javax.usb, API HID, extensiones personalizadas.
Acceso Bluetooth a través de javax.bluetooth o conexión de E/S OSGi.
Servicio de posición para información GPS de una transmisión NMEA.
Servicio de reloj para la sincronización del reloj del sistema.
API de Kura para acceso GPIO / PWM / I2C / SPI.

6.3.2. Data Services

Almacena y reenvía la funcionalidad de los datos de telemetría recopilados por la puerta de enlace y publicados en servidores remotos.

Sistema de publicación basado en políticas, que abstrae al desarrollador de la aplicación de la complejidad de la capa de red y el protocolo de publicación utilizado. Eclipse Paho y su cliente MQTT proporcionan la biblioteca de mensajes predeterminada.

6.3.3. Cloud Services

En este apartado, se encuentra la Capa API que permite a la aplicación IoT comunicarse con un servidor remoto. Además de la simple publicación / suscripción, la API de servicios en la nube simplifica la implementación de flujos de interacción más complejos como la solicitud / respuesta o la gestión remota de recursos. Permita que una única conexión a un servidor remoto se comparta en más de una aplicación en la puerta de enlace, proporcionando la partición de tema necesaria.

6.3.4. Configuration Services

Aprovecha las especificaciones de OSGi ConfigurationAdmin y MetaType para proporcionar un servicio de instantáneas para importar/exportar la configuración de todos los servicios registrados en el contenedor.

6.3.5. Remote Management

Permite la administración remota de las aplicaciones IoT instaladas en Kura, incluida su administración de implementación, actualización y configuración.

6.3.6. Networking

Proporciona API para introspects y permite configurar las interfaces de red disponibles en la puerta de enlace, como Ethernet, Wifi y módems celulares.

6.3.7. Watchdog Service

Registre componentes críticos en el Servicio de vigilancia, lo que forzará un reinicio del sistema a través de la vigilancia del hardware cuando se detecte un problema.

6.3.8. Web Administration Interface

Ofrezca una consola de administración basada en web que se ejecute dentro del contenedor Kura para administrar la puerta de enlace.

6.3.9. Drivers and Assets

Se presenta un modelo unificado para simplificar la comunicación con los dispositivos conectados a la puerta de enlace. El Controlador encapsula el protocolo de comunicación y sus parámetros de configuración, mientras que el Activo, que es genérico en todos los Controladores, modela los canales de información hacia el dispositivo. Cuando se crea un activo, un espejo del dispositivo está disponible automáticamente para lectura y escritura a pedido a través de API de Java o de la nube a través de mensajes remotos.

6.3.10. Wires

Ofrece una herramienta de programación de flujo de datos modular y visual para definir la recolección de datos y las tuberías de procesamiento en el borde simplemente seleccionando componentes de una paleta y conectándolos entre sí. De esta forma, los usuarios pueden, por ejemplo, configurar un Activo, adquirir periódicamente datos de sus canales, almacenarlos en la puerta de enlace, filtrarlos o agregarlos utilizando potentes consultas SQL y enviar los resultados a la Nube. Eclipse Kura Marketplace es un repositorio desde el cual se pueden instalar componentes Wires adicionales en su tiempo de ejecución Kura con un simple arrastrar y soltar.

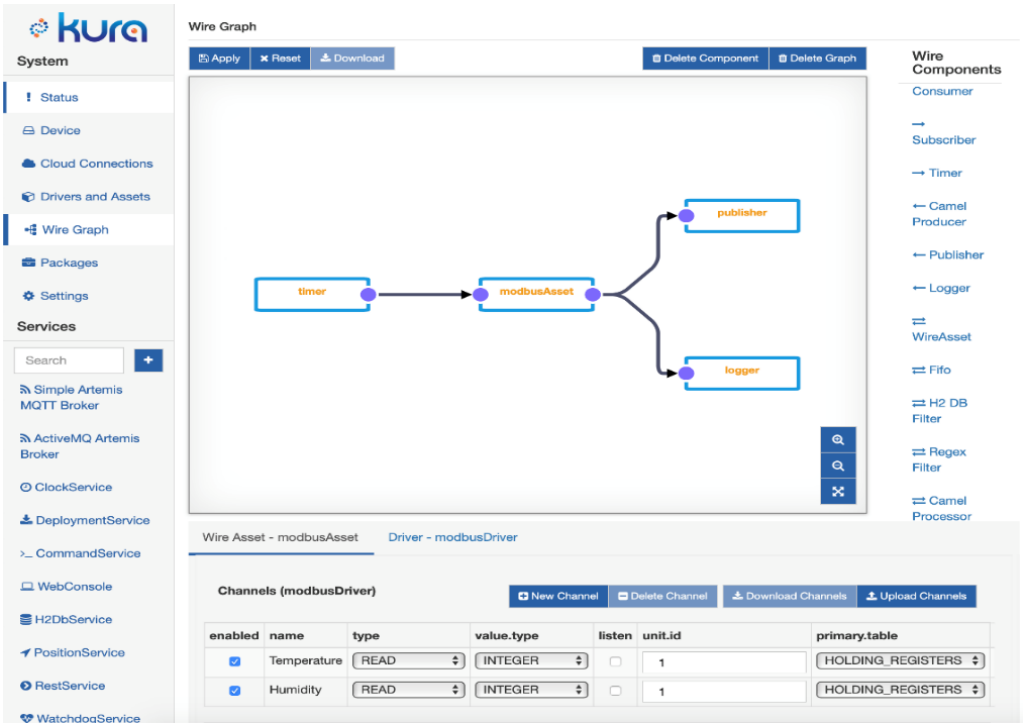

Figura 104. Servicio Eclipse Kura Wires.

Fuente: E. Foundation, "Eclipse Kura - Open Source framework for IoT". [En línea]. Disponible en: http://www.eclipse.org/kura/. [Consultado: 21-may-2018].

6.4. INSTALACIÓN DE ECLIPSE KURA EN UNA RASPBERRY PI

Instalar Eclipse Kura en una Raspberry Pi para convertirla en un Dispositivo Gateway IoT es un procedimiento sencillo que debe realizarse teniendo en cuenta algunas recomendaciones que solucionan problemas de compatibilidad entre la plataforma Kura y el sistema operativo Raspbian.

Tomaremos como referencia la guía de inicio rápido para Raspberry Pi (Raspberry Pi Quick Start) que encontramos en el sitio web oficial de Eclipse IoT, concretamente en el siguiente enlace: http://eclipse.github.io/kura/intro/raspberry-pi-quick-start.html.

Esta guía de instalación se realizó con la imagen **2020-02-13-raspbian-buster.img** del sistema operativo Raspbian.

Inicie la Raspberry Pi con la última imagen del sistema operativo Raspbian. A partir de la versión 2.1.0, Kura solo es compatible con Debian 8 o superior.

Dado que la última versión de Raspbian Stretch adoptó el nuevo Consistent Network Device Naming[43], se debe deshabilitar esta convención para ejecutar de manera correcta Eclipse Kura en la Raspberry Pi, para ello agregamos el parámetro **net.ifnames=0** al final del archivo **/boot/cmdline.txt** de nuestro sistema operativo Raspbian.

```
pi@raspberrypi : ~ $ sudo nano /boot/cmdline.txt
```

El paquete dhcpcd5 no es compatible con Kura y debe eliminarse mediante el siguiente comando:

```
pi@raspberrypi : ~ $ sudo apt-get purge dhcpcd5
```

[43] https://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames/, consultado: 26/07/2019. El Consistent Network Device Naming o Nombramiento Consistente de Dispositivos de Red en español, es una convención para nombrar adaptadores Ethernet en Linux.

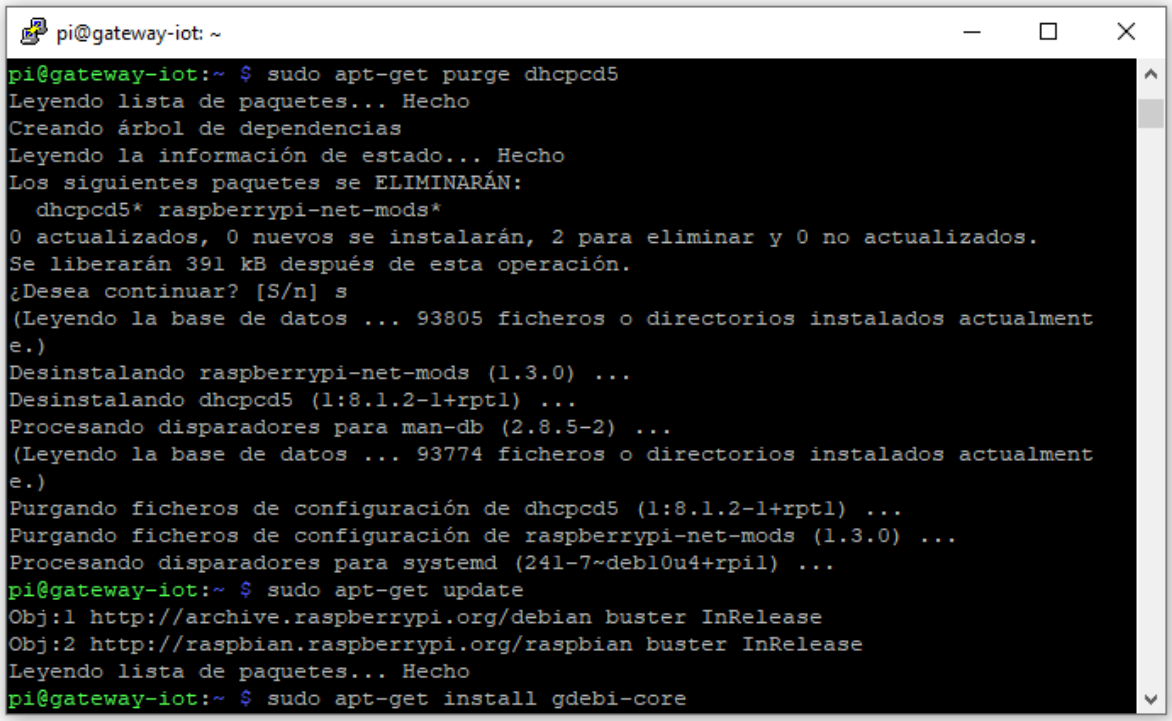

Figura 105. Desinstalación del paquete dhcpcd5.

Fuente: Elaboración propia

6.4.1. Habilitar SSH (Secure Shell)

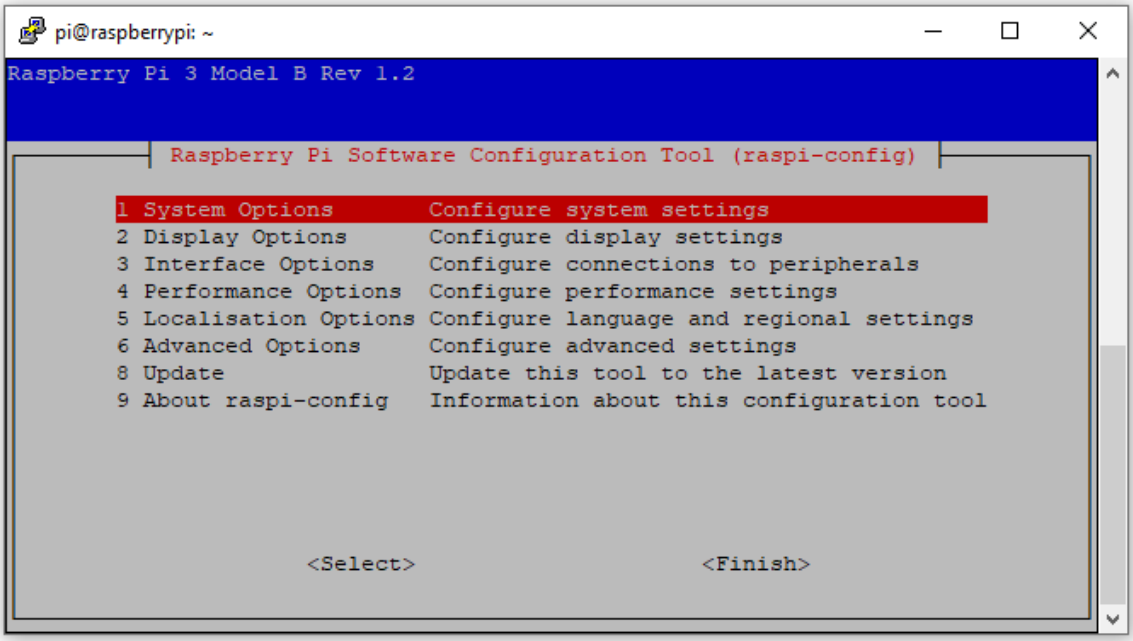

Figura 106. Habilitar SSH.

Fuente: Elaboración propia

Ingresamos a la terminal y ejecutamos el siguiente comando:

```
pi@raspberrypi : ~ $ sudo raspi-config
```

En la Figura 106 se ilustra el panel de configuración de las Raspberry Pi desde la terminal para habilitar el SSH. Seleccionamos la opción 3 “Interface Options”, luego “SSH” y habilitamos el servidor.

6.4.2. Habilitar interfaces de red para acceso a internet:

Para ver los nombres de las interfaces de red que tiene nuestra Raspberry Pi, incluso si estas no están conectadas, ejecutamos el siguiente comando:

```
pi@raspberrypi : ~ $ ls /sys/class/net/
```

Una vez hemos verificado las interfaces de red disponibles en nuestra Raspberry Pi, configuramos las conexiones en ellas según sea nuestra necesidad, por cable Ethernet o una conexión inalámbrica Wifi. Para este caso, configuraremos la interfaz de red WiFi. Editamos el archivo que contiene las interfaces de red conocidas, ejecutando el comando:

```
pi@raspberrypi : ~ $ sudo nano /etc/network/interfaces
```

Aunque no utilizaremos la interfaz Ethernet para la conexión de nuestra Raspberry a la red, escribimos las siguientes líneas para dejarla habilitada:

Configuración de direcciones IP automáticas para la interfaz Ethernet:

auto eth0
allow-hotplug eth0
iface eth0 inet dhcp

Configuración de direcciones IP estáticas para la interfaz Ethernet:

auto eth0
iface eth0 inet static
address 192.0.2.7
netmask 255.255.255.0 gateway 192.0.2.254

Fuente: https://wiki.debian.org/es/NetworkConfiguration

Luego, en la parte inferior, agregamos estas líneas que le dicen a la Raspberry Pi que permita la interfaz wlan0 como método de conexión de red automático y use /etc/wpa_supplicant/wpa_supplicant.conf como su archivo de configuración.

Para configurar una conexión donde se le fije a la Raspberry una dirección IP de manera estática (la dirección IP que se escriba no debe estar siendo usada por ningún otro dispositivo en la red donde conectaremos nuestra Raspberry Pi):

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.0.23
    netmask 255.255.255.0
    gateway 192.168.0.1
    dns-nameservers 192.168.0.1 8.8.8.8
wpa-conf ing a_supplicant/wpa_supplicant.conf
```

Abrimos el archivo de configuración wpa_supplicant.conf en el editor para configurar la conexión WiFi:

```
pi@raspberrypi : ~ $ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

6.4.3. Instalar el paquete gdebi desde la línea de comandos:

Para instalar el paquete gdebi se ejecutan los siguientes comandos en la terminal:

```
pi@raspberrypi : ~ $ sudo apt-get update
```

```
pi@raspberrypi : ~ $ sudo apt-get install gdebi-core
```

La Figura 107 muestra el resultado de la instalación del paquete gdebi.

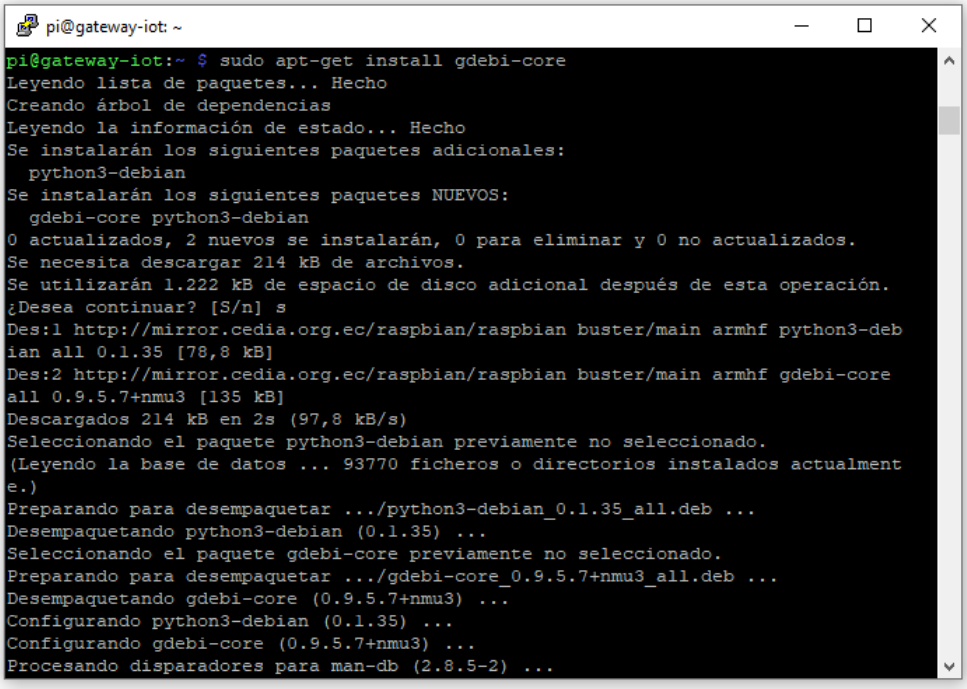

Figura 107. Instalación del paquete gdebi.

Fuente: Elaboración propia.

6.4.4. Instalación de OpenJDK

Los siguientes comandos son necesarios para la instalación de Java. Primero se comprueba si ya se encuentra instalado, en caso contrario, es necesario ejecutar el segundo comando en la terminal.

```
pi@raspberrypi : ~ $ java -version
```

Si no tenemos instalado el OpenJDK, ejecutamos el siguiente comando:

```
pi@raspberrypi : ~ $ sudo apt-get install openjdk-8-jre-headless
```

6.4.5. Descarga e instalación del paquetes

Descargamos los paquetes de instalación de la página oficial del proyecto Eclipse Kura, para ello ejecutamos en la terminal el siguiente comando:

```
pi@raspberrypi : ~ $ wget
http://download.eclipse.org/kura/releases/<version>/kura_<version>_raspberry-pi-2-3_installer.deb
```

Se debe reemplazar <version> en la URL por el número de la versión de Eclipse Kura que se desea descargar (Ej. 4.1.0).

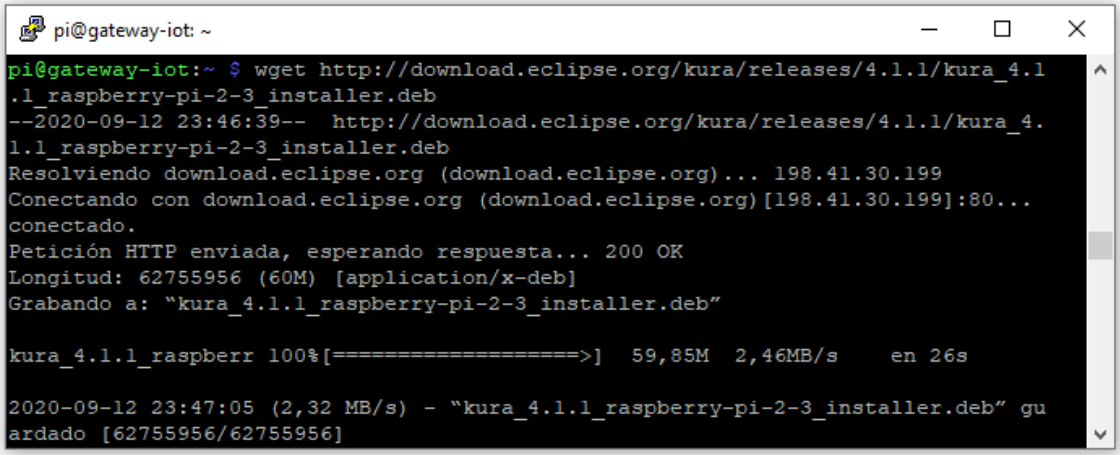

Figura 108. Descarga del paquete Kura.

Fuente: Elaboración propia.

Instalamos Kura con el comando:

```
pi@raspberrypi : ~ $ sudo gdebi kura_<version>_raspberry-pi-2-3_installer.deb
```

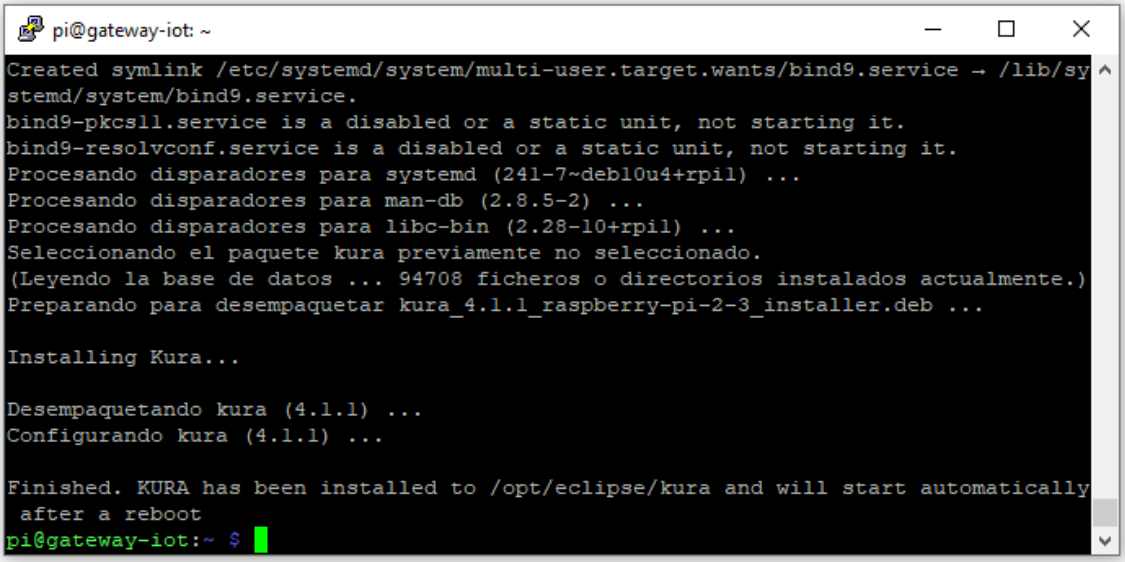

Figura 109. Instalación de Eclipse Kura desde la terminal.

Fuente: Elaboración propia.

Se debe reiniciar la Raspberry Pi:

```
pi@raspberrypi : ~ $ sudo reboot
```

Kura iniciará en nuestra placa una vez se reinicie el sistema operativo.

Compruebe la conexión a internet realizando ping a un sitio web desde el terminal o visitando un sitio web desde el navegador de internet.

Si se establece conexión a internet y desea conservar la configuración IP automática de las interfaces de red al instalar Kura, conserve el archivo de configuración original **/etc/network/interfaces**.

Si desea configurar manualmente las interfaces de red, realice la siguiente configuración (ajuste el rango de direcciones IP a las de la red local donde está conectada la Raspberry Pi):

```
# /etc/network/interfaces -- configuration file for Kura

# Virtual loopback network interface
auto lo
iface lo inet static
        address 127.0.0.1
        netmask 255.0.0.0

# Wired and wireless interfaces
auto eth0

# Ethernet configuration static
```

```
iface eth0 inet static
    address <address-ip>
    netmask <netmask-ip>
    gateway <gateway-ip>
    dns-nameservers <dns-nameserver1-ip> <dns-nameserver2-ip>

# Wifi configuration static
auto wlan0
iface wlan0 inet static
    address 172.16.1.1
    netmask 255.255.255.0
```

Kura instala una interfaz web local a la que se puede acceder a través del navegador web, usando la dirección IP local del dispositivo o la dirección Lookback del mismo.

Una vez digitamos la dirección IP del dispositivo en el navegador web, nos pedirá autenticarnos en la gateway IoT. El usuario y contraseña por defecto son: admin. La interfaz que nos presenta Kura es la que se ilustra en la Figura 110.

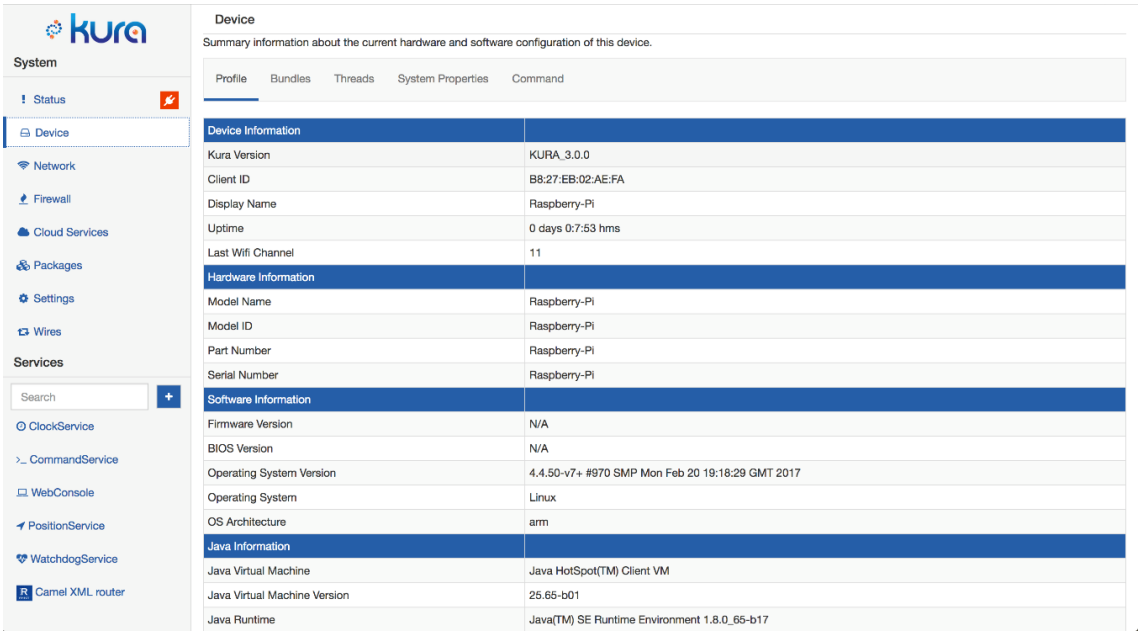

Figura 110. Interfaz de Eclipse Kura en navegador web.

Fuente: Elaboración propia.

ANEXO B: CONFIGURACIÓN DE MÓDULOS XBEE DESDE XCTU

6.5. DESCUBRIMIENTO DE MÓDULOS XBEE

La configuración inicial de los módulos XBee se realiza a través del software XCTU[44] desarrollado por la empresa DIGI, fabricante de los módulos inalámbricos. XCTU facilita el alistamiento, configuración inicial y pruebas de funcionamiento de los XBee.

El primer paso es conectar los módulos XBee al ordenador donde se tiene instalada la herramienta XCTU. Los dispositivos son descubiertos en su respectivo puerto de comunicación conectado.

Una vez termine el descubrimiento de dispositivos, se seleccionan los parámetros del puerto y se eligen los dispositivos para agregarlos a la lista.

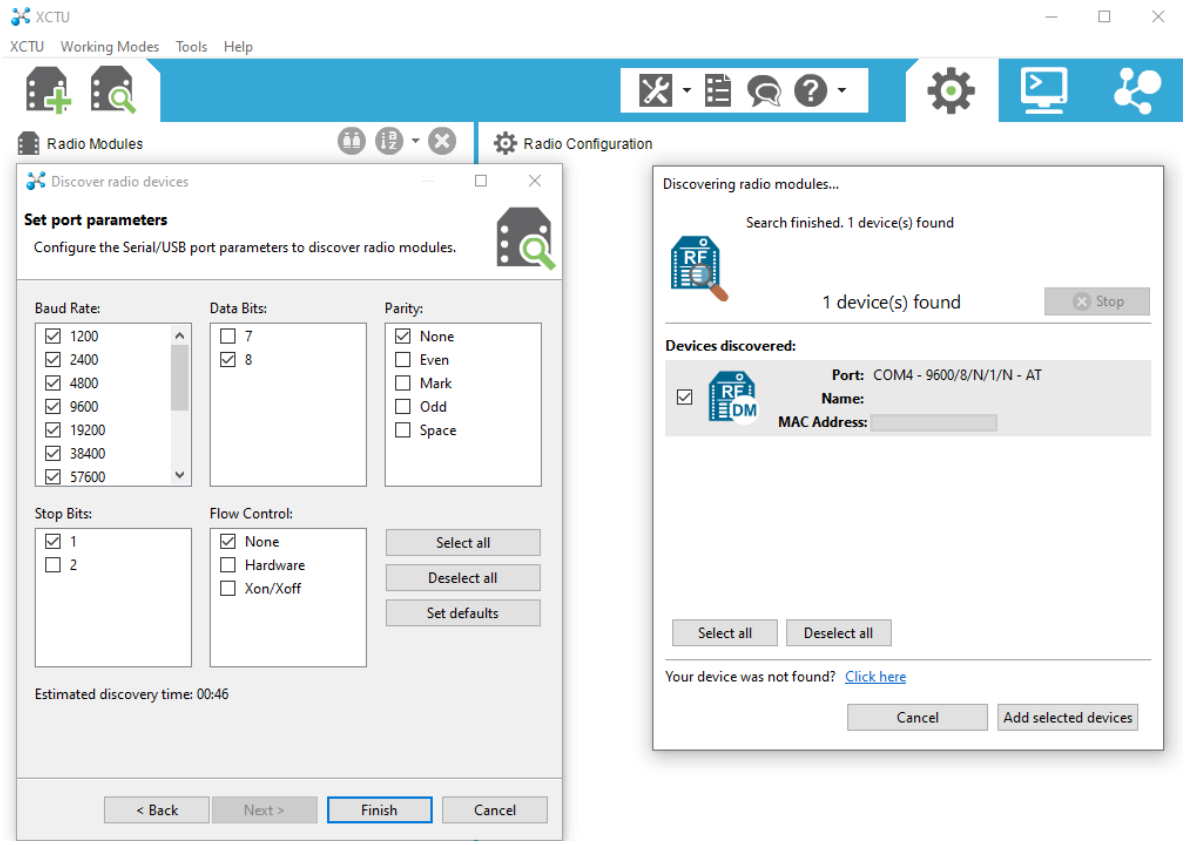

Figura 111. Descubrimiento de módulos XBee desde XCTU.

Fuente: Elaboración propia.

Se recomienda actualizar el firmware de los módulos Xbee a su versión más reciente. La Figura 112 muestra el proceso de actualización que se hizo a los módulos XBee.

[44] XCTU - Download and Install the Configuration Platform for XBee/RF Solutions | Digi International, consultado 01/02/2021.

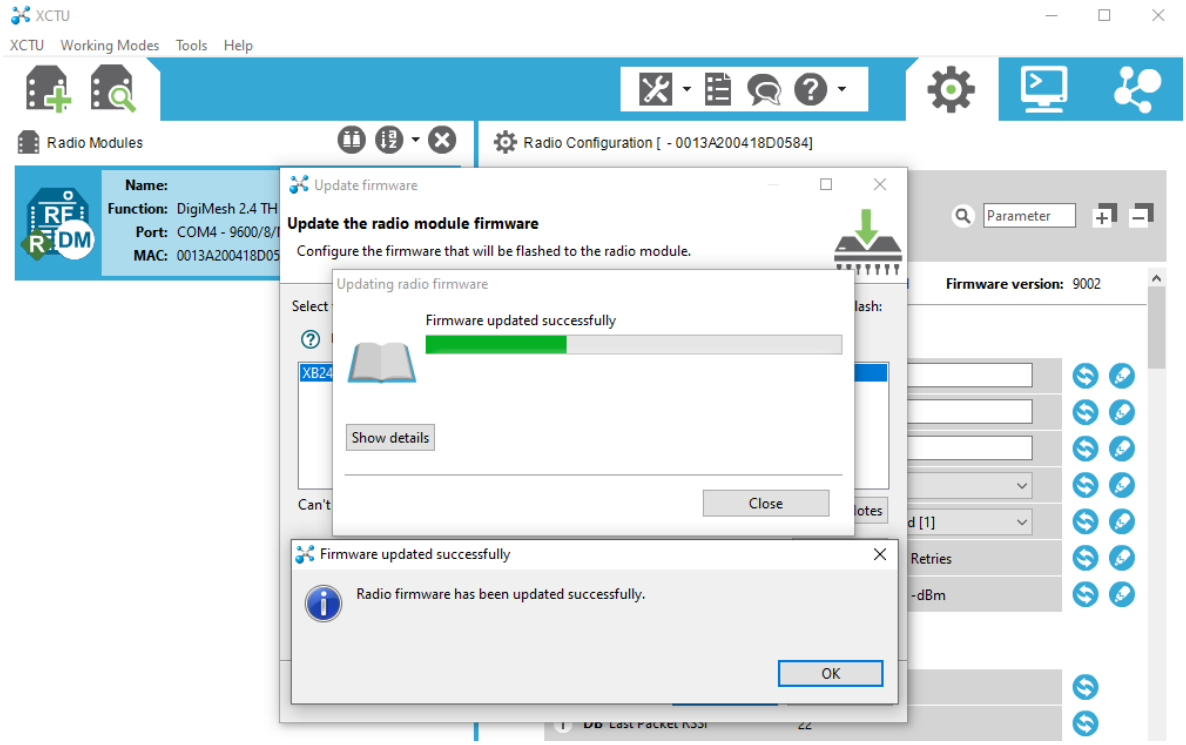

Figura 112. Actualización de Firmware en módulos XBee.

Fuente: Elaboración propia.

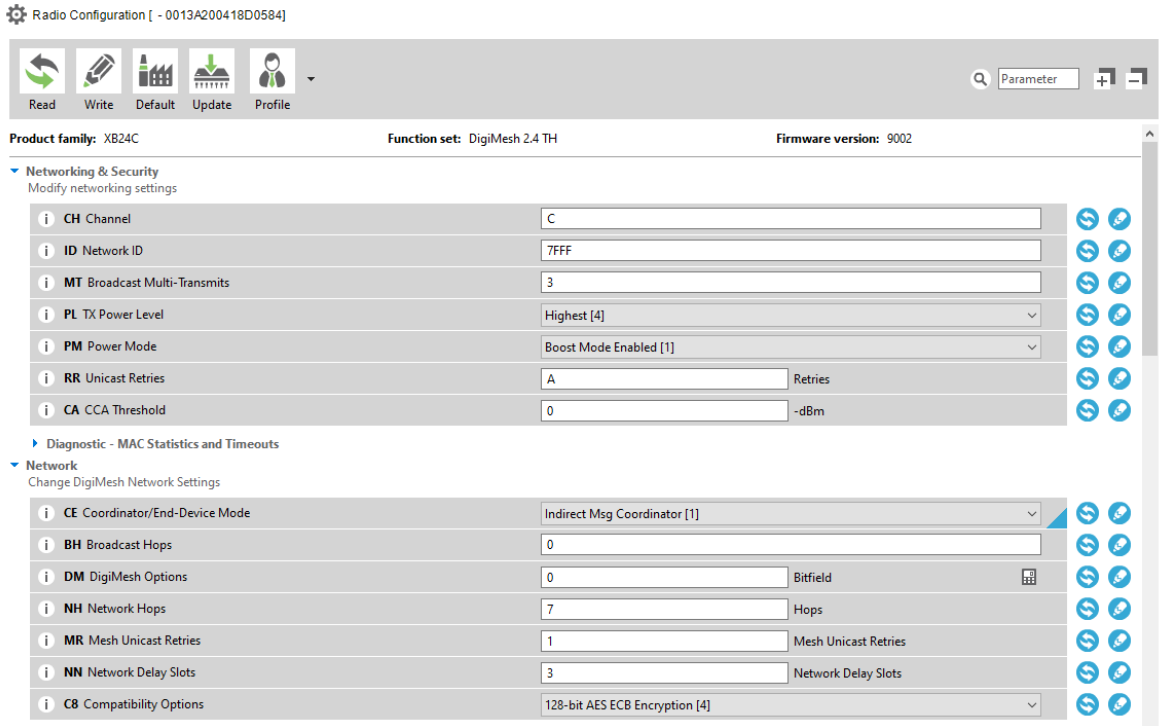

Figura 113. Interfaz de configuración de módulo XBee.

Fuente: Elaboración propia.

Seleccionando el módulo agregado se accede a la interfaz de configuración, tal como se muestra en la Figura 113. El parámetro **ID Network ID** debe ser igual en ambos módulos para poder establecer el enlace de comunicación inalámbrico.

Se configura un módulo como “Coordinador” y el segundo módulo como **Standard Router** en el parámetro **Coordinator/End-Device Mode** en el apartado de **Network**.

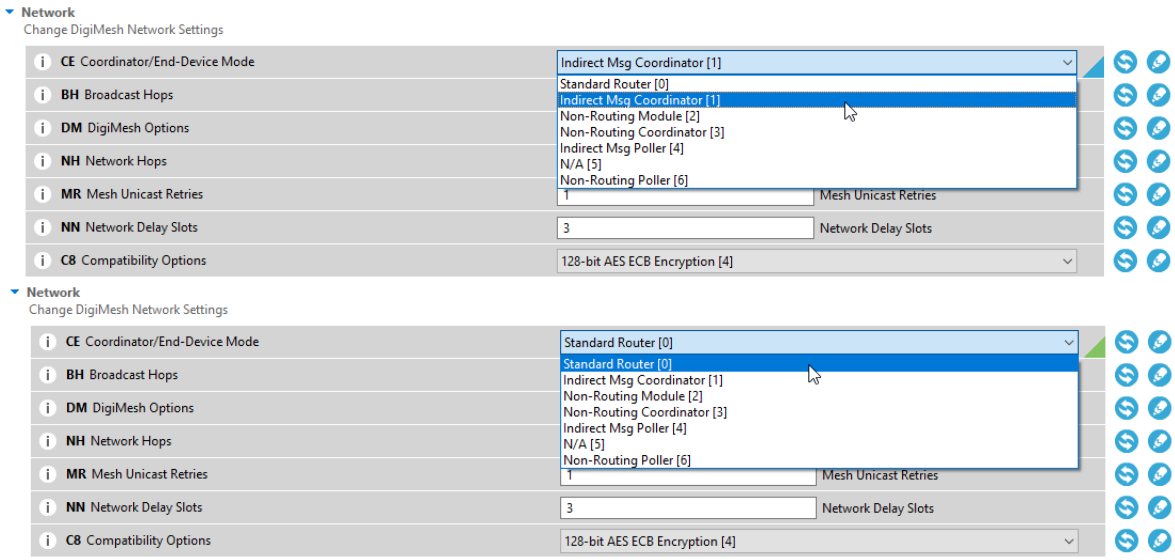

Figura 114. Configuración modos de operación XBee.

Fuente: Elaboración propia.

Se realiza la prueba de conexión y comunicación entre los dos módulos accediendo a la consola del software XCTU. Se puede observar que los mensajes escritos en el **Console log** del XBee coordinador son recibidos por el router, y viceversa.

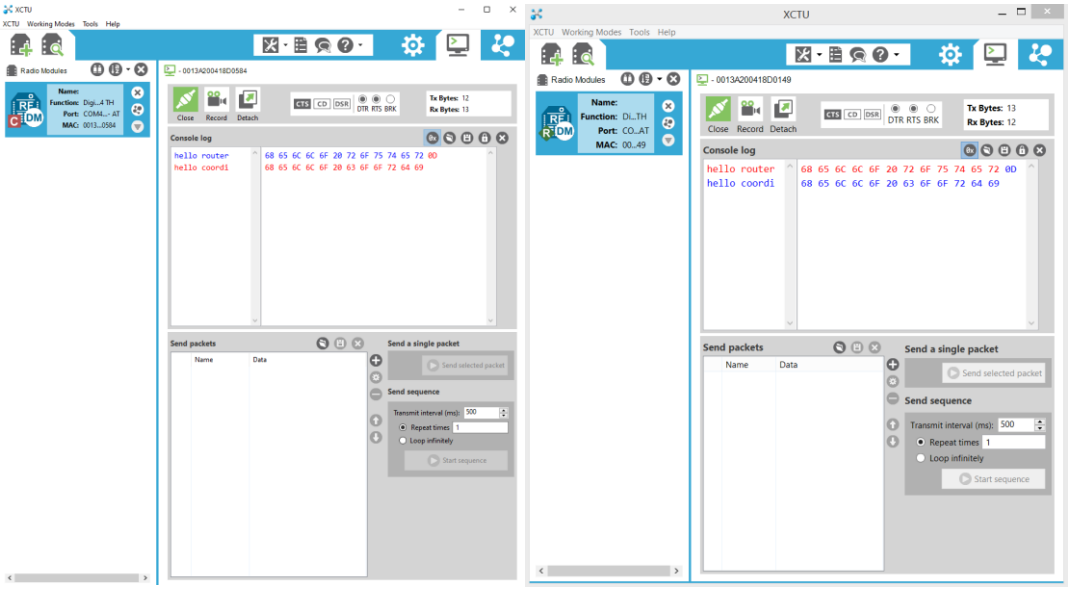

Figura 115. Prueba de comunicación entre módulos XBee.

Fuente: Elaboración propia.

Una vez se valide que los mensajes enviados desde el log de consola en el coordinador son recibidos por el dispositivo configurado como router y que a su vez, este también puede enviar mensajes al coordinador, se termina el proceso de configuración y establecimiento de conexión entre los módulos Xbee para ser conectados al dispositivo IoT y al Gateway respectivamente.

ANEXO C: CONFIGURACIÓN DE OBJETOS INTELIGENTES

En este anexo se muestra el proceso de instalación de librerías y paquetes necesarios para implementar los objetos inteligentes [40].

6.6. INSTALACIÓN DE LIBRERÍAS OBJETO INTELIGENTE

A continuación se muestran los comandos utilizados en la instalación y configuración de los objetos inteligentes:

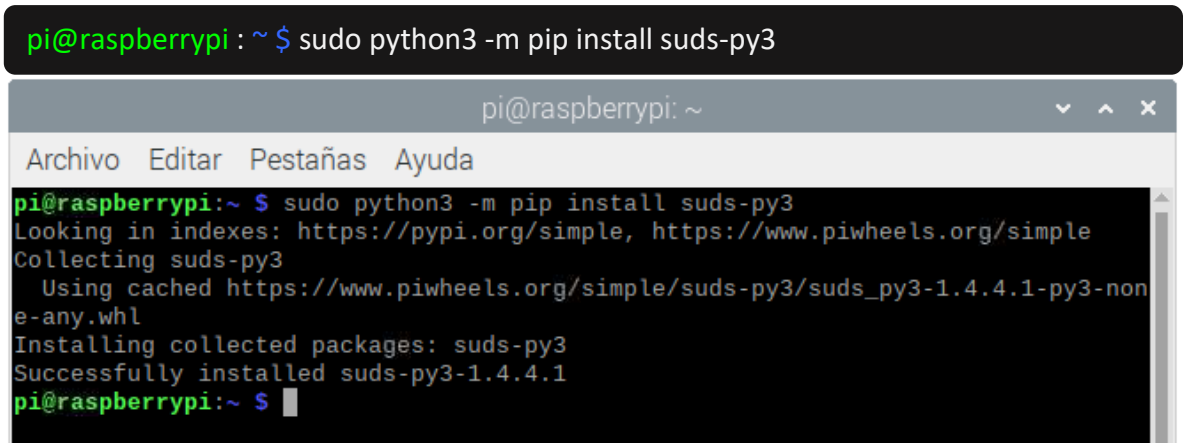

Figura 116. Instalación de librería suds.

Fuente: Elaboración propia.

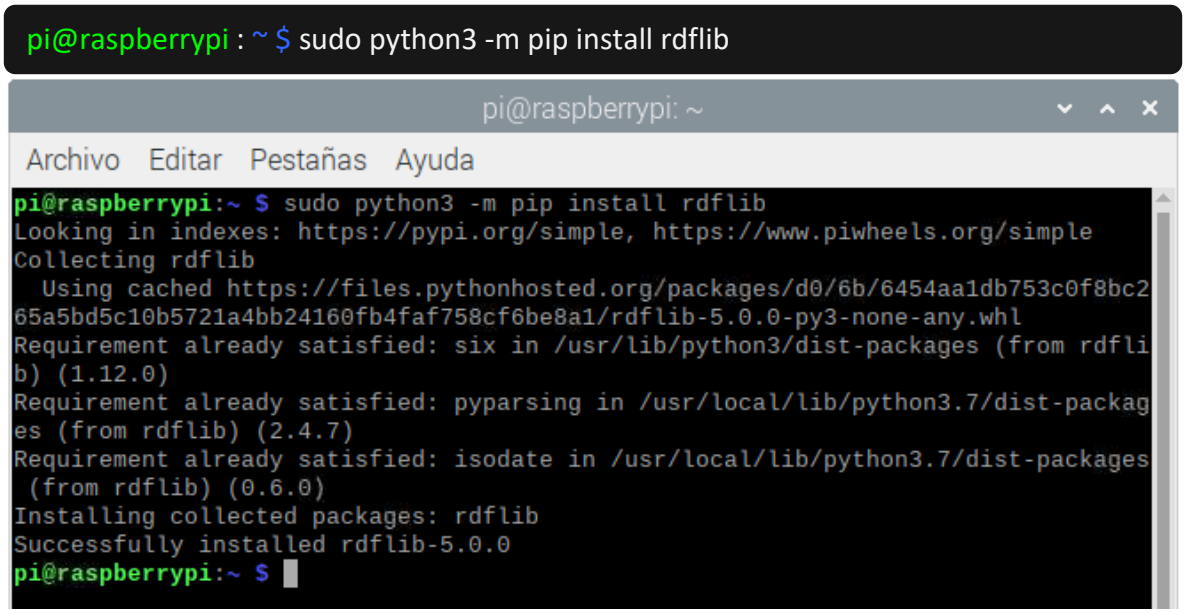

Figura 117. Instalación de librería rdflib.

Fuente: Elaboración propia.

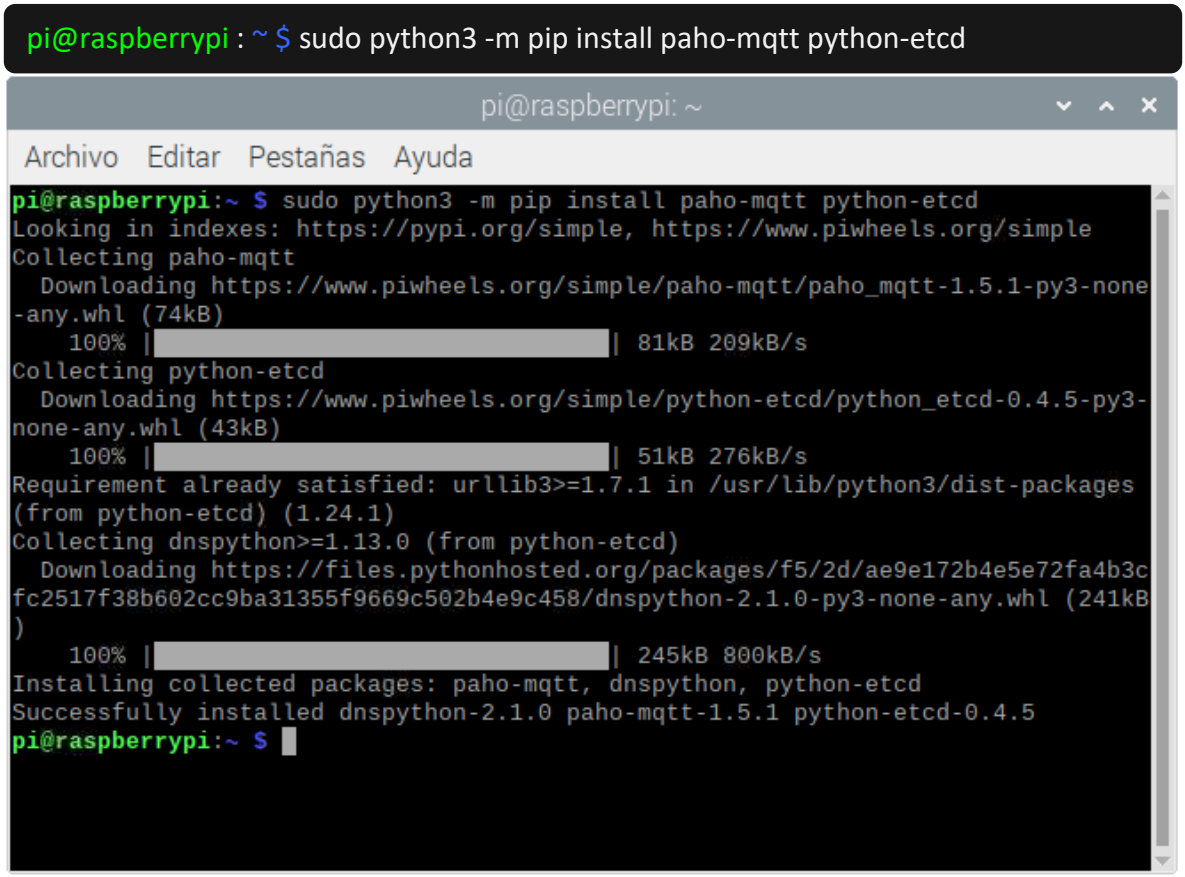

Figura 118. Instalación de librería paho-mqtt.

Fuente: Elaboración propia.

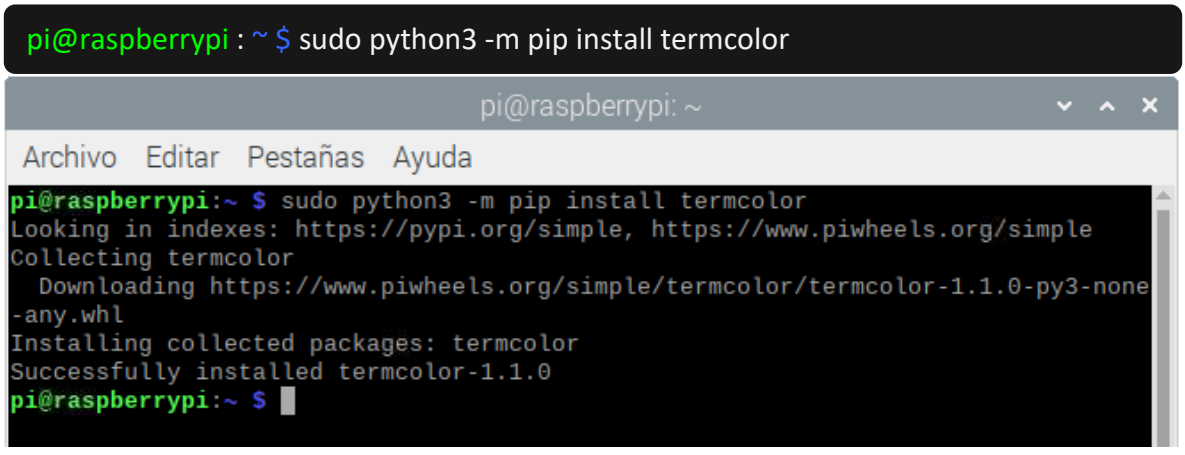

Figura 119. Instalación de librería termcolor.

Fuente: Elaboración propia.

```
pi@raspberrypi : ~ $ sudo python3 -m pip install web.py==0.40-dev1
```

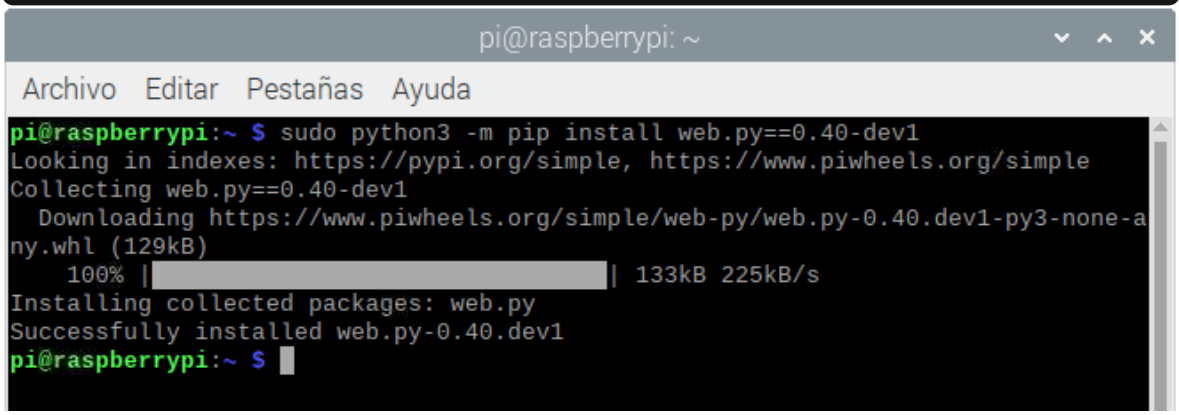

Figura 120. Instalación de librería web.py.

Fuente: Elaboración propia.

```
pi@raspberrypi : ~ $ pip install psutil
```

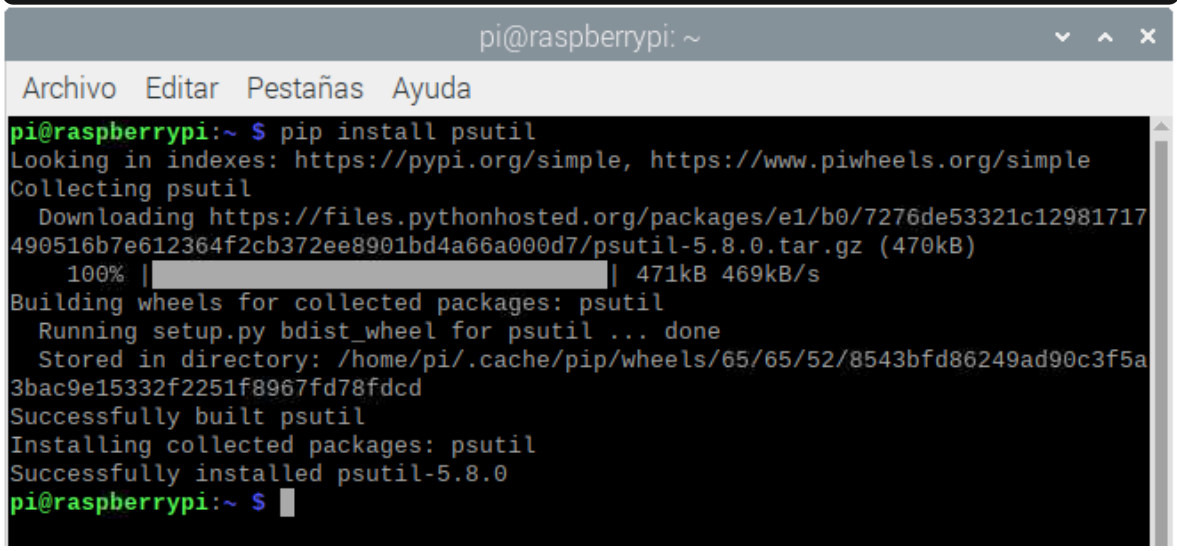

Figura 121. Instalación de librería psutil.

Fuente: Elaboración propia.

```
pi@raspberrypi : ~ $ sudo python3 -m pip install Owlready2
```

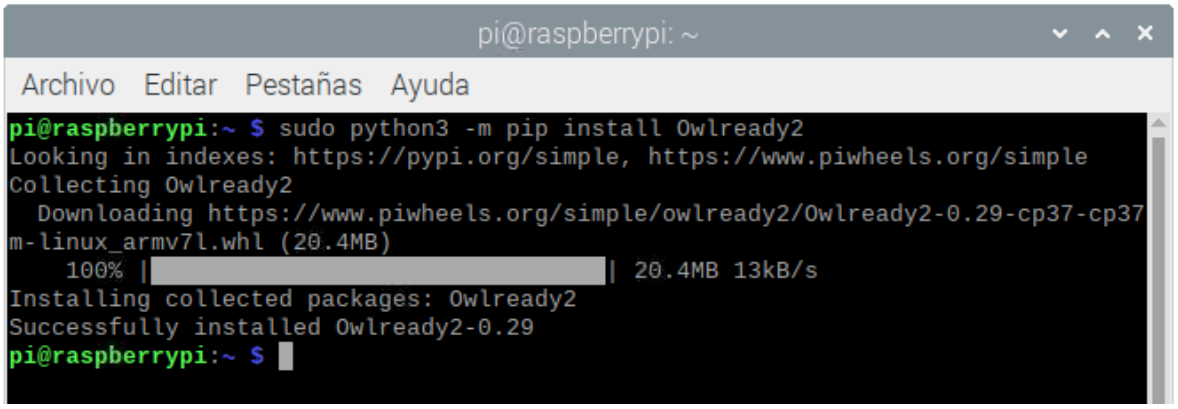

Figura 122. Instalación Owlready2 biblioteca para el manejo de ontologías.

Fuente: Elaboración propia.

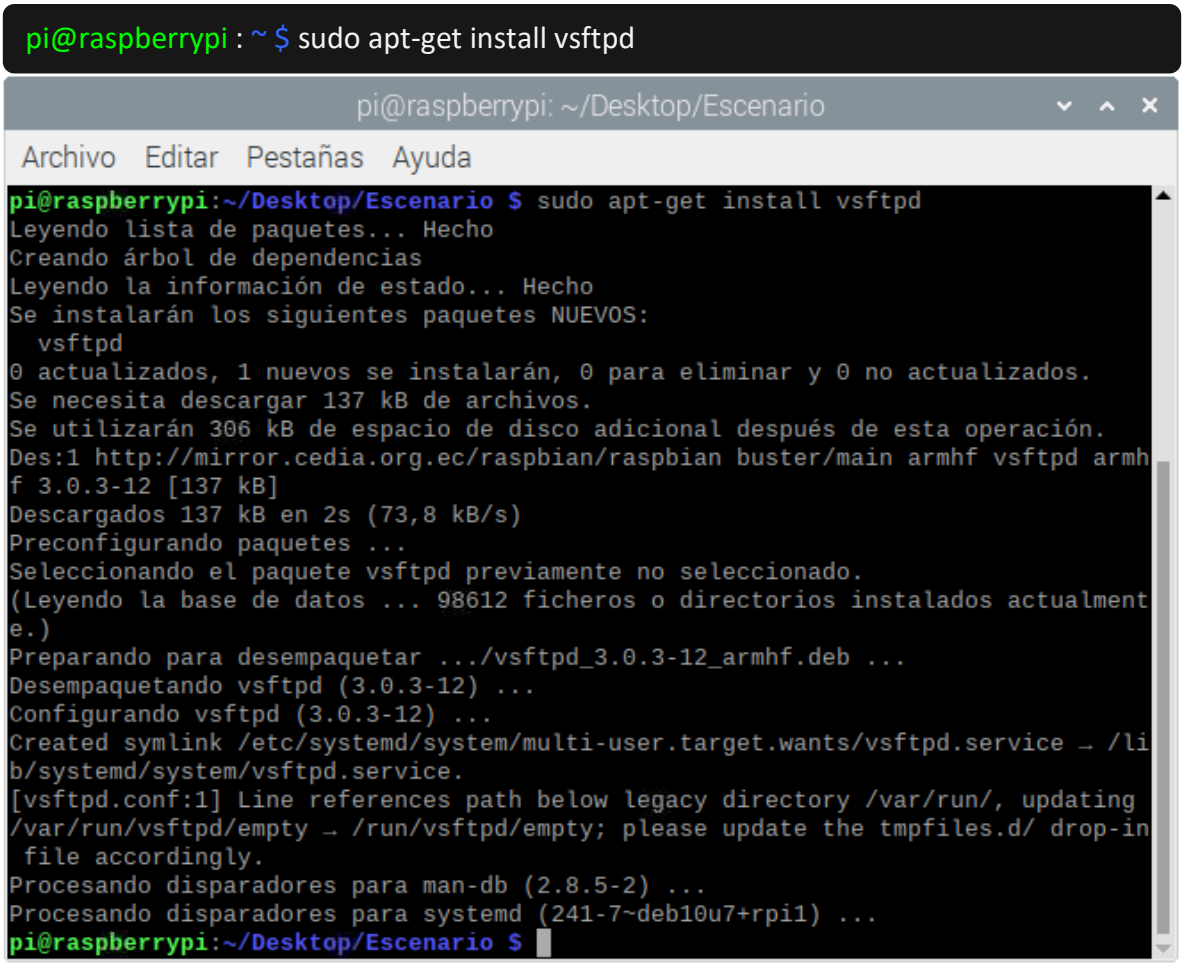

Figura 123. Instalación de servidor ftp vsftpd.

Fuente: Elaboración propia.

Se incluye en este anexo, el código fuente utilizado para la lectura de datos climatológicos de los sensores de la estación. Este código corre en la tarjeta Arduino Uno a donde se conecta la estación y los datos enviados son recibidos por el Objeto Inteligente B a través del puerto serie de comunicaciones.

```
#include <Wire.h> //I2C needed for sensors
#include "SparkFunMPL3115A2.h" //Pressure sensor - Search "SparkFun
MPL3115" and install from Library Manager
#include "SparkFun_Si7021_Breakout_Library.h" //Humidity sensor - Search
"SparkFun Si7021" and install from Library Manager

MPL3115A2 myPressure; //Create an instance of the pressure sensor
Weather myHumidity;//Create an instance of the humidity sensor

//Hardware pin definitions
//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// digital I/O pins
const byte WSPEED = 3;
const byte RAIN = 2;
const byte STAT1 = 7;
const byte STAT2 = 8;

// analog I/O pins
const byte REFERENCE_3V3 = A3;
const byte LIGHT = A1;
const byte BATT = A2;
const byte WDIR = A0;
//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

//Global Variables
//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
long lastSecond; //The millis counter to see when a second rolls by
byte seconds; //When it hits 60, increase the current minute
byte seconds_2m; //Keeps track of the "wind speed/dir avg" over last 2
minutes array of data
byte minutes; //Keeps track of where we are in various arrays of data
byte minutes_10m; //Keeps track of where we are in wind gust/dir over
last 10 minutes array of data

long lastWindCheck = 0;
volatile long lastWindIRQ = 0;
volatile byte windClicks = 0;

byte windspdavg[120]; //120 bytes to keep track of 2 minute average

#define WIND_DIR_AVG_SIZE 120
int winddiravg[WIND_DIR_AVG_SIZE]; //120 ints to keep track of 2 minute
average
float windgust_10m[10]; //10 floats to keep track of 10 minute max
int windgustdirection_10m[10]; //10 ints to keep track of 10 minute max
volatile float rainHour[60]; //60 floating numbers to keep track of 60
minutes of rain

//These are all the weather values that wunderground expects:
int winddir = 0; // [0-360 instantaneous wind direction]
```

```
float windspeedmph = 0; // [mph instantaneous wind speed]
float windgustmph = 0; // [mph current wind gust, using software specific
time period]
int windgustdir = 0; // [0-360 using software specific time period]
float windspdmph_avg2m = 0; // [mph 2 minute average wind speed mph]
int winddir_avg2m = 0; // [0-360 2 minute average wind direction]
float windgustmph_10m = 0; // [mph past 10 minutes wind gust mph ]
int windgustdir_10m = 0; // [0-360 past 10 minutes wind gust direction]
float humidity = 0; // [%]
float tempf = 0; // [temperature F]
float rainin = 0; // [rain inches over the past hour)] -- the accumulated
rainfall in the past 60 min
volatile float dailyrainin = 0; // [rain inches so far today in local
time]
//float baromin = 30.03;// [barom in] - It's hard to calculate baromin
locally, do this in the agent
float pressure = 0;
//float dewptf; // [dewpoint F] - It's hard to calculate dewpoint
locally, do this in the agent

float batt_lvl = 11.8; //[analog value from 0 to 1023]
float light_lvl = 455; //[analog value from 0 to 1023]

// volatiles are subject to modification by IRQs
volatile unsigned long raintime, rainlast, raininterval, rain;
//-------------------------------------------------------------

//Interrupt routines (these are called by the hardware interrupts, not by
the main code)
//-------------------------------------------------------------
void rainIRQ()
// Count rain gauge bucket tips as they occur
// Activated by the magnet and reed switch in the rain gauge, attached to
input D2
{
  raintime = millis(); // grab current time
  raininterval = raintime - rainlast; // calculate interval between this
and last event

  if (raininterval > 10) // ignore switch-bounce glitches less than 10mS
after initial edge
  {
    dailyrainin += 0.011; //Each dump is 0.011" of water
    rainHour[minutes] += 0.011; //Increase this minute's amount of rain

    rainlast = raintime; // set up for next event
  }
}

void wspeedIRQ()
// Activated by the magnet in the anemometer (2 ticks per rotation),
attached to input D3
{
  if (millis() - lastWindIRQ > 10) // Ignore switch-bounce glitches less
than 10ms (142MPH max reading) after the reed switch closes
  {
```

```
    lastWindIRQ = millis(); //Grab the current time
    windClicks++; //There is 1.492MPH for each click per second.
  }
}

void setup()
{
  Serial.begin(9600);
  Serial.println("Weather Shield Example");

  pinMode(STAT1, OUTPUT); //Status LED Blue
  pinMode(STAT2, OUTPUT); //Status LED Green

  pinMode(WSPEED, INPUT_PULLUP); // input from wind meters windspeed
sensor
  pinMode(RAIN, INPUT_PULLUP); // input from wind meters rain gauge
sensor

  pinMode(REFERENCE_3V3, INPUT);
  pinMode(LIGHT, INPUT);

  //Configure the pressure sensor
  myPressure.begin(); // Get sensor online
  myPressure.setModeBarometer(); // Measure pressure in Pascals from 20
to 110 kPa
  myPressure.setOversampleRate(7); // Set Oversample to the recommended
128
  myPressure.enableEventFlags(); // Enable all three pressure and temp
event flags

  //Configure the humidity sensor
  myHumidity.begin();

  seconds = 0;
  lastSecond = millis();

  // attach external interrupt pins to IRQ functions
  attachInterrupt(0, rainIRQ, FALLING);
  attachInterrupt(1, wspeedIRQ, FALLING);

  // turn on interrupts
  interrupts();

  Serial.println("Weather Shield online!");
}

void loop()
{
  //Keep track of which minute it is
  if (millis() - lastSecond >= 1000)
  {
    digitalWrite(STAT1, HIGH); //Blink stat LED

    lastSecond += 1000;

    //Take a speed and direction reading every second for 2 minute
```

```
average
    if (++seconds_2m > 119) seconds_2m = 0;

    //Calc the wind speed and direction every second for 120 second to
get 2 minute average
    float currentSpeed = get_wind_speed();
    windspeedmph = currentSpeed;//update global variable for windspeed
when using the printWeather() function
    //float currentSpeed = random(5); //For testing
    int currentDirection = get_wind_direction();
    windspdavg[seconds_2m] = (int)currentSpeed;
    winddiravg[seconds_2m] = currentDirection;
    //if(seconds_2m % 10 == 0) displayArrays(); //For testing

    //Check to see if this is a gust for the minute
    if (currentSpeed > windgust_10m[minutes_10m])
    {
      windgust_10m[minutes_10m] = currentSpeed;
      windgustdirection_10m[minutes_10m] = currentDirection;
    }

    //Check to see if this is a gust for the day
    if (currentSpeed > windgustmph)
    {
      windgustmph = currentSpeed;
      windgustdir = currentDirection;
    }

    if (++seconds > 59)
    {
      seconds = 0;

      if (++minutes > 59) minutes = 0;
      if (++minutes_10m > 9) minutes_10m = 0;

      rainHour[minutes] = 0; //Zero out this minute's rainfall amount
      windgust_10m[minutes_10m] = 0; //Zero out this minute's gust
    }

    //Report all readings every second
    printWeather();

    digitalWrite(STAT1, LOW); //Turn off stat LED
  }

  delay(5000);
}

//Calculates each of the variables that wunderground is expecting
void calcWeather()
{
  //Calc winddir
  = get_wind_direction();

  //Calc windspeed
  //windspeedmph = get_wind_speed(); //This is calculated in the main
```

loop on line 185

```
  //Calc windgustmph
  //Calc windgustdir
  //These are calculated in the main loop

  //Calc windspdmph_avg2m
  float temp = 0;
  for (int i = 0 ; i < 120 ; i++)
    temp += windspdavg[i];
  temp /= 120.0;
  windspdmph_avg2m = temp;

  long sum = winddiravg[0];
  int D = winddiravg[0];
  for (int i = 1 ; i < WIND_DIR_AVG_SIZE ; i++)
  {
    int delta = winddiravg[i] - D;

    if (delta < -180)
      D += delta + 360;
    else if (delta > 180)
      D += delta - 360;
    else
      D += delta;

    sum += D;
  }
  winddir_avg2m = sum / WIND_DIR_AVG_SIZE;
  if (winddir_avg2m >= 360) winddir_avg2m -= 360;
  if (winddir_avg2m < 0) winddir_avg2m += 360;

  windgustmph_10m = 0;
  windgustdir_10m = 0;
  //Step through the 10 minutes
  for (int i = 0; i < 10 ; i++)
  {
    if (windgust_10m[i] > windgustmph_10m)
    {
      windgustmph_10m = windgust_10m[i];
      windgustdir_10m = windgustdirection_10m[i];
    }
  }

  //Calc humidity
  humidity = myHumidity.getRH();

  tempf = myPressure.readTempF();

  rainin = 0;
  for (int i = 0 ; i < 60 ; i++)
    rainin += rainHour[i];

  //Calc pressure
  pressure = myPressure.readPressure();
```

```
  //Calc dewptf

  //Calc light level
  light_lvl = get_light_level();

  //Calc battery level
  batt_lvl = get_battery_level();
}

float get_light_level()
{
  float operatingVoltage = analogRead(REFERENCE_3V3);

  float lightSensor = analogRead(LIGHT);

  operatingVoltage = 3.3 / operatingVoltage; //The reference voltage is
3.3V

  lightSensor = operatingVoltage * lightSensor;

  return (lightSensor);
}

float get_battery_level()
{
  float operatingVoltage = analogRead(REFERENCE_3V3);

  float rawVoltage = analogRead(BATT);

  operatingVoltage = 3.30 / operatingVoltage; //The reference voltage is
3.3V

  rawVoltage = operatingVoltage * rawVoltage; //Convert the 0 to 1023 int
to actual voltage on BATT pin

  rawVoltage *= 4.90; //(3.9k+1k)/1k - multiple BATT voltage by the
voltage divider to get actual system voltage

  return (rawVoltage);
}

//Returns the instataneous wind speed
float get_wind_speed()
{
  float deltaTime = millis() - lastWindCheck; //750ms

  deltaTime /= 1000.0; //Covert to seconds

  float windSpeed = (float)windClicks / deltaTime; //3 / 0.750s = 4

  windClicks = 0; //Reset and start watching for new wind
  lastWindCheck = millis();

  windSpeed *= 1.492; //4 * 1.492 = 5.968MPH

  return (windSpeed);
```

```
}

//Read the wind direction sensor, return heading in degrees
int get_wind_direction()
{
  unsigned int adc;

  adc = analogRead(WDIR); // get the current reading from the sensor

  if (adc < 380) return (113);
  if (adc < 393) return (68);
  if (adc < 414) return (90);
  if (adc < 456) return (158);
  if (adc < 508) return (135);
  if (adc < 551) return (203);
  if (adc < 615) return (180);
  if (adc < 680) return (23);
  if (adc < 746) return (45);
  if (adc < 801) return (248);
  if (adc < 833) return (225);
  if (adc < 878) return (338);
  if (adc < 913) return (0);
  if (adc < 940) return (293);
  if (adc < 967) return (315);
  if (adc < 990) return (270);
  return (-1); // error, disconnected?
}

//Prints the various variables directly to the port
//I don't like the way this function is written but Arduino doesn't
support floats under sprintf
void printWeather()
{
  calcWeather(); //Go calc all the various sensors

  Serial.println();
  Serial.print("winddir= ");
  Serial.print(winddir);
  Serial.print("\nwindspeedmph= ");
  Serial.print(windspeedmph, 1);
  Serial.print("\nhumidity= ");
  Serial.print(humidity, 1);
  Serial.print("\ntempf= ");
  Serial.print(tempf, 1);
  Serial.print("\nrainin= ");
  Serial.print(rainin, 2);
  Serial.print("\npressure= ");
  Serial.print(pressure, 2);
  Serial.print("\nlight_lvl= ");
  Serial.print(light_lvl, 2);
  Serial.print("\n");
}
```

Fuente: Mike Grusin's USB Weather Board code: https://www.sparkfun.com/products/10586

ANEXO D: CONFIGURACIÓN DE AWS IOT CORE

Un primer paso en la configuración de AWS IoT Core para conectarse con el Gateway Inteligente IoT es darse de alta con una cuenta AWS. En el siguiente enlace se detallan las instrucciones:

https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html

Una vez nos damos de alta en la plataforma, podemos acceder a la consola de administración de AWS como se ilustra en la Figura 124.

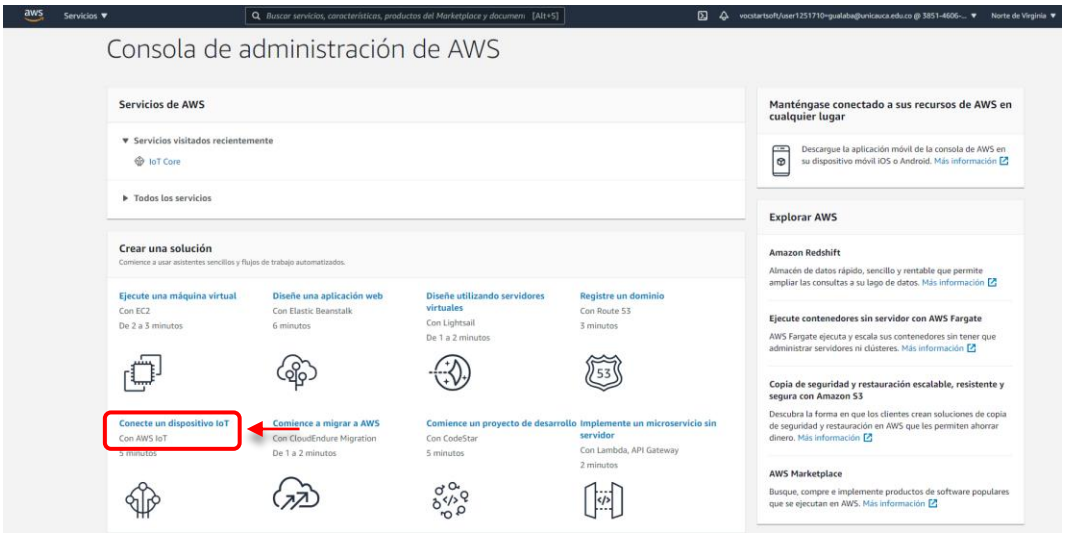

Figura 124. Consola de administración AWS.

Fuente: Elaboración propia.

Accedemos al servicio de IoT Core desde la consola, como lo ilustra la Figura ***125***.

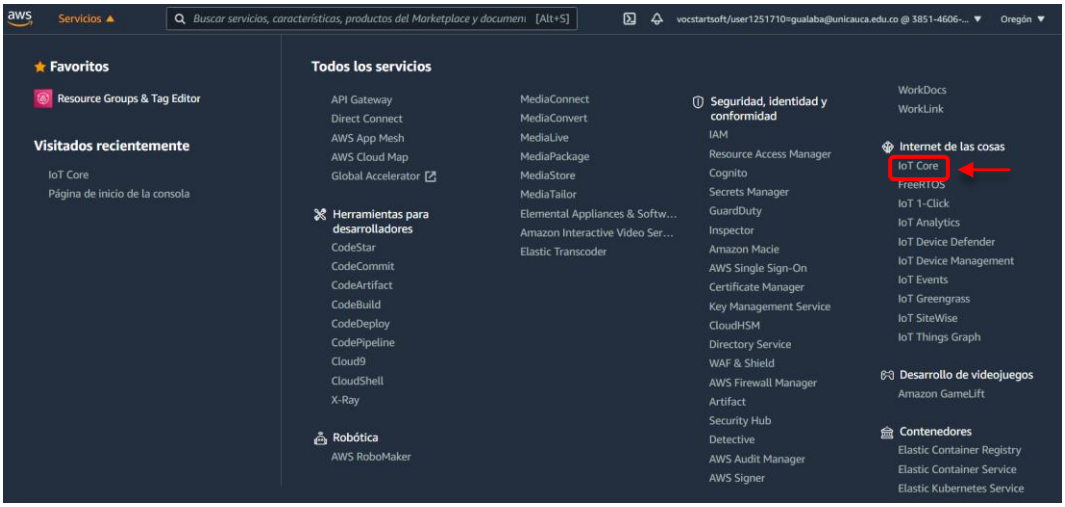

Figura 125. IoT Core.

Fuente: Elaboración propia.

Accedemos al menú de configuración de objetos como lo ilustra la Figura ***126***.

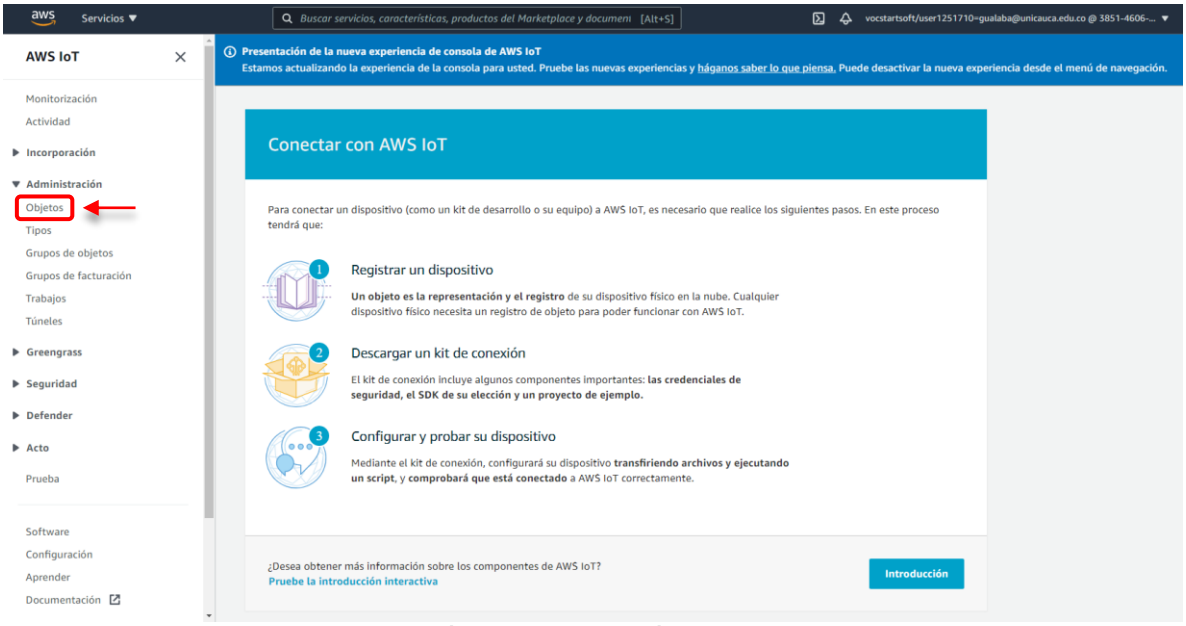

Figura 126. Menú de configuración de objetos AWS IoT.

Fuente: Elaboración propia.

Se realiza el registro de un nuevo objeto como se ilustra en la Figura 127.

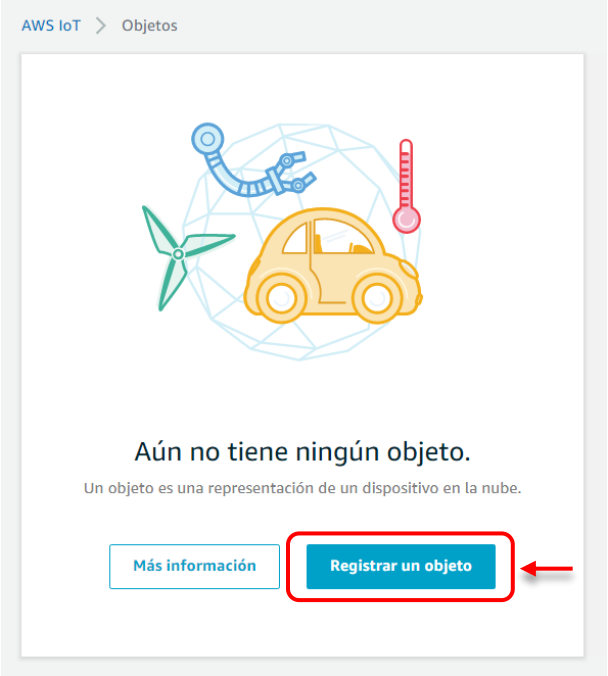

Figura 127. Registro de objetos AWS IoT.

Fuente: Elaboración propia.

Se crea el objeto en AWS IoT seleccionando la opción que se muestra en la Figura 128.

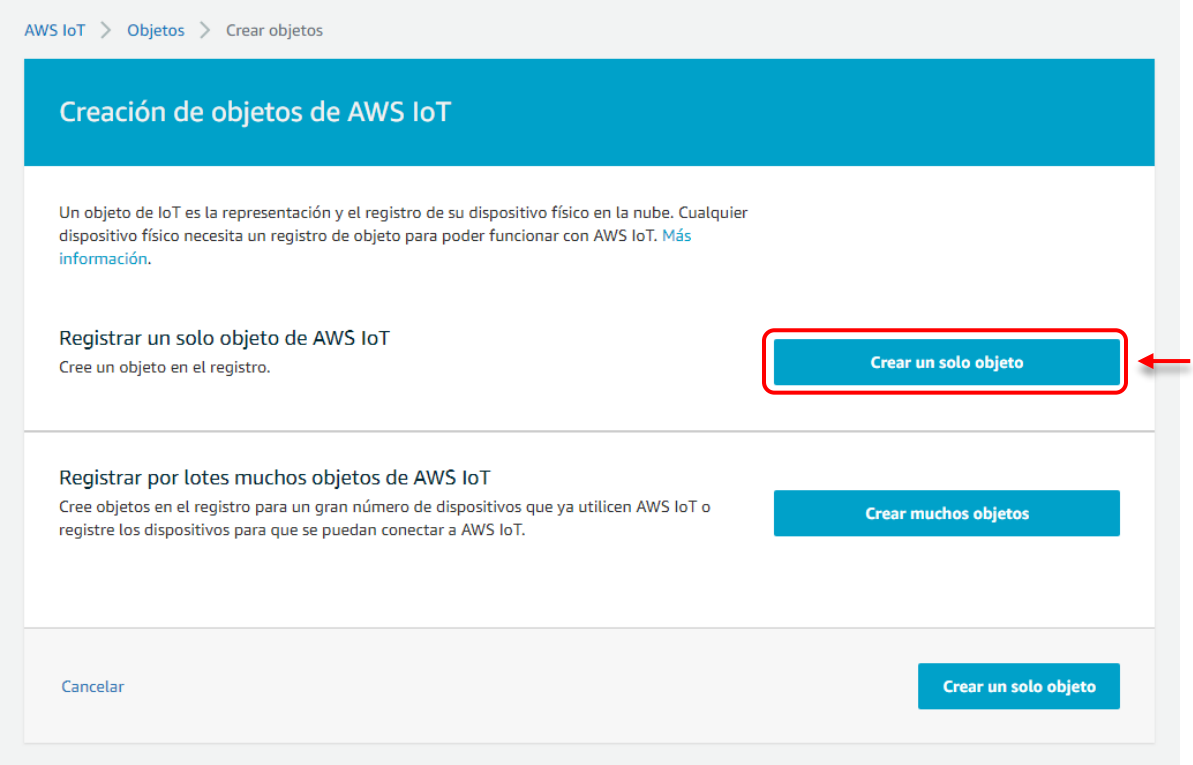

Figura 128. Creación de objetos AWS IoT Core.

Fuente: Elaboración propia.

En la Figura 129 se muestra la opción que se debe seleccionar para agregar el dispositivo al registro de objetos de AWS IoT.

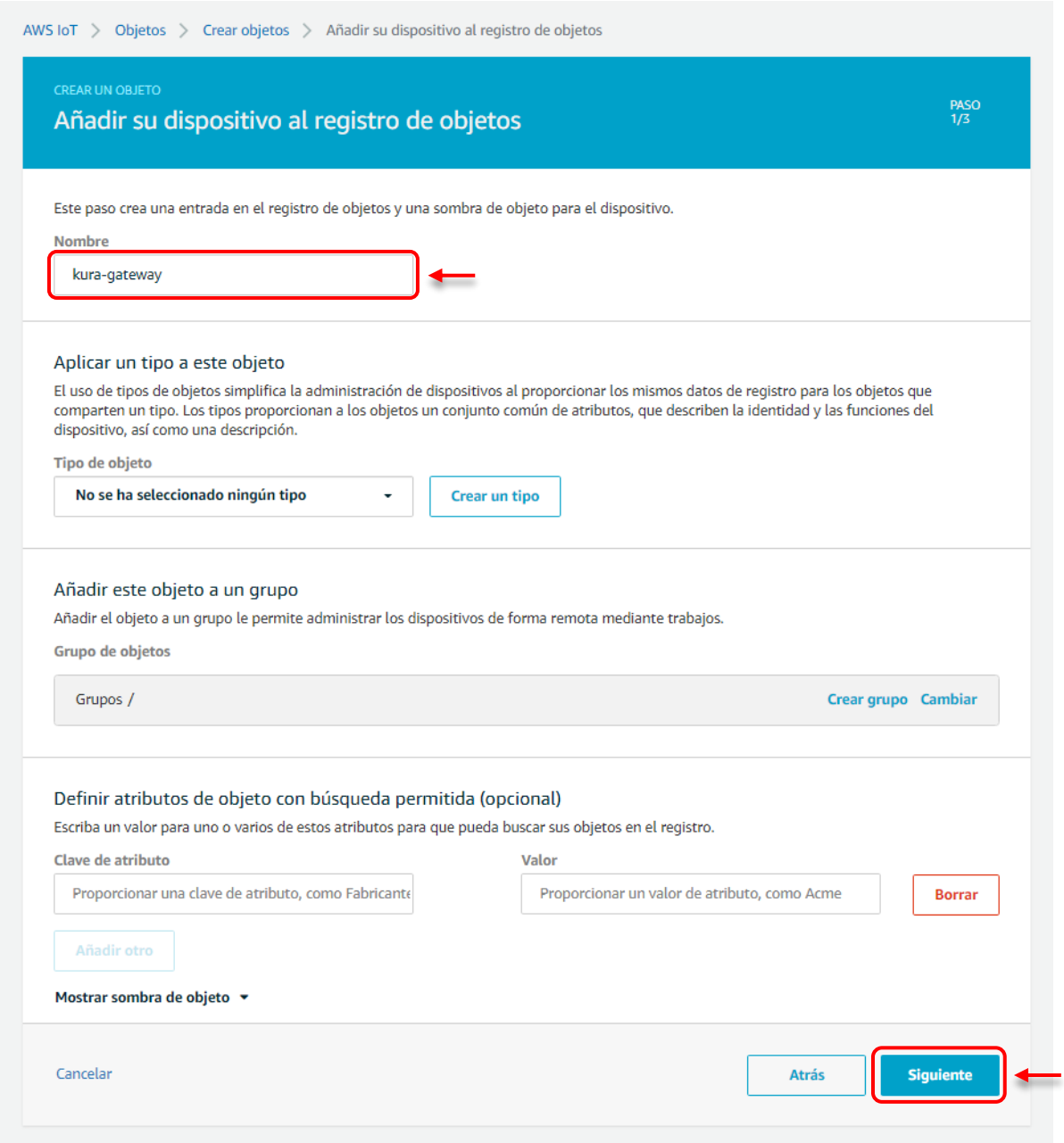

Figura 129. Agregar dispositivo al registro de objetos AWS IoT Core.

Fuente: Elaboración propia.

A continuación, se crea el certificado de seguridad para el objeto, como se ilustra en la Figura 130.

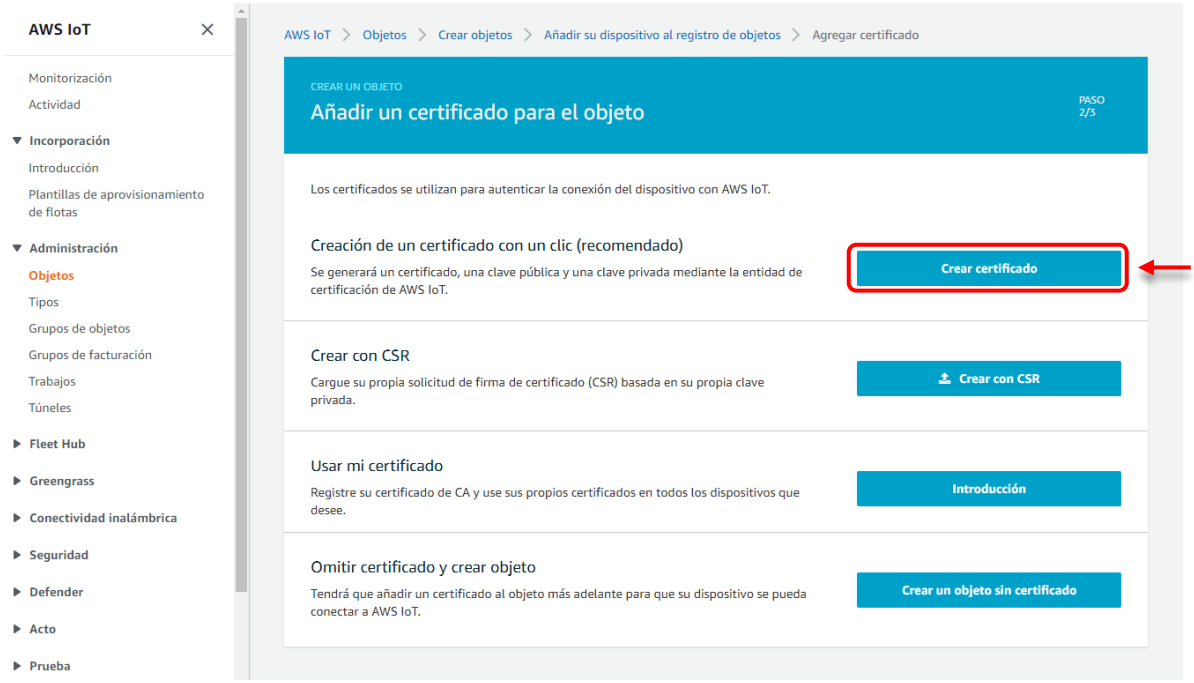

Figura 130. Creación de certificados SSL/TLS para el objeto AWS.

Fuente: Elaboración propia.

Finalmente, se deben descargar los certificados creados para el objeto y uno vez descargados a un directorio local, se procede a activarlos. La Figura 131 muestra la interfaz para la descarga y activación de los certificados de seguridad de AWS IoT Core.

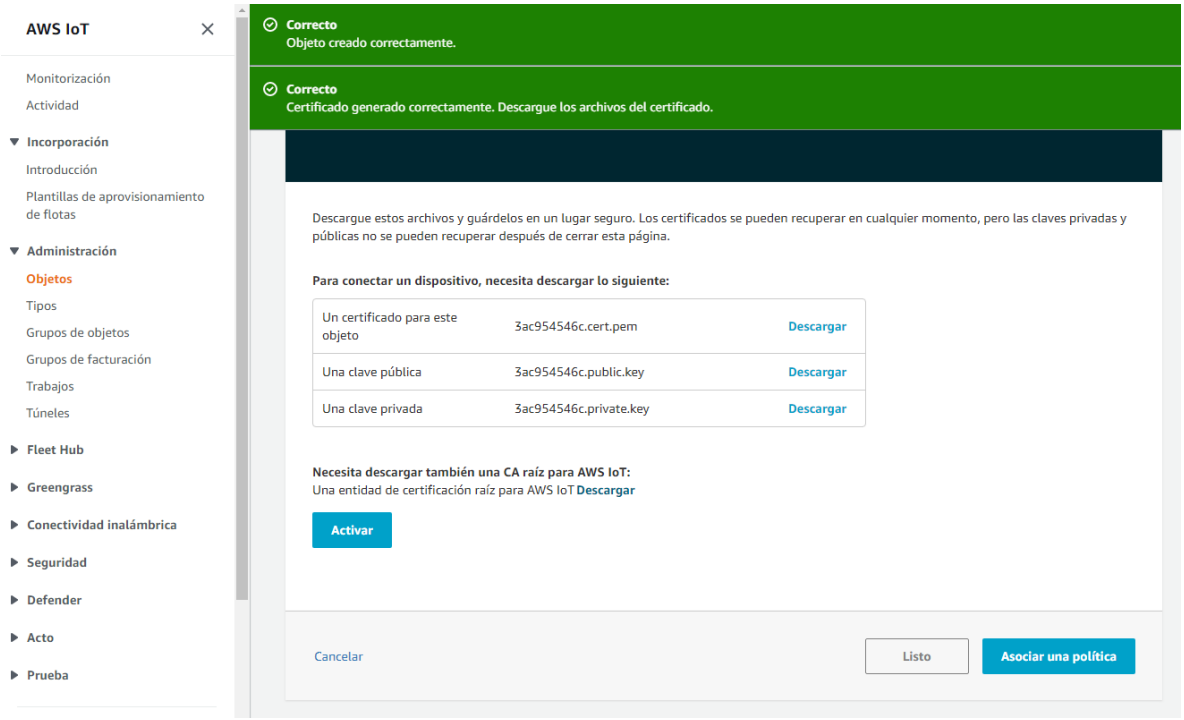

Figura 131. Descarga de certificados de seguridad de AWS IoT Core.

Fuente: Elaboración propia.

ANEXO E: CÓDIGO FUENTE DESCUBRIMIENTO DE OBJETOS INTELIGENTES

```
import requests
from requests.exceptions import HTTPError
from subprocess import Popen, PIPE

cont = 0

print "PyScanner: Buscando Dispositivos Conectados al Gateway"
print " "

with open("ips.txt", "a") as myfile:
   myfile.write('Objetos Inteligentes Conectados\n\n')

#---------------------------------------------------------------------------
----------------------
# Segmento de Red Adaptador Wifi wlan0
#---------------------------------------------------------------------------
----------------------
for ip in range(100,106):

   ipAddress = '172.16.1.'+str(ip)
   subprocess = Popen(['/bin/ping', '-c 1 ', ipAddress], stdin=PIPE,
stdout=PIPE, stderr=PIPE)
   stdout, stderr= subprocess.communicate(input=None)

   cont +=1

   if "bytes from " in stdout:

      res = requests.get('http://''%s'
%(stdout.split()[1])+'/Identificator')
      res = requests.get(stdout.split()[1])


      if res:
         print "| "+str(cont)+" | %s | Objeto Inteligente |"
%(stdout.split()[1])
         with open("ips.txt", "a") as myfile:
            myfile.write(stdout.split()[1]+'\n')
      else:
         print "| "+str(cont)+" | %s | Dispositivo IP|"
%(stdout.split()[1])

#---------------------------------------------------------------------------
----------------------
# Segmento de Red Adaptador Ethernet eth1
#---------------------------------------------------------------------------
----------------------
for ip in range(100, 106):

   ipAddress = '172.17.1.' + str(ip)
   subprocess = Popen(['/bin/ping', '-c 1 ', ipAddress], stdin=PIPE,
```

```
stdout=PIPE, stderr=PIPE)
   stdout, stderr = subprocess.communicate(input=None)

   cont += 1

   if "bytes from " in stdout:

      # res = requests.get('http://''%s'
%(stdout.split()[1])+'/Identificator')
      res = requests.get(stdout.split()[1])

      if res:
         print
         "| " + str(cont) + " | %s | Objeto Inteligente |" %
(stdout.split()[1])
         with open("ips.txt", "a") as myfile:
            myfile.write(stdout.split()[1] + '\n')
      else:
         print
         "| " + str(cont) + " | %s | Dispositivo IP|" %
(stdout.split()[1])

print " "
print "Total Dispositivos Activos: " + str(cont)
```

ANEXO F: INSTALACIÓN DE RPI-MONITOR EN LA RASPBERRY PI

Desde la Figura 132 hasta la Figura 138 se muestra el procedimiento de instalación de RPi-Monitor desde la terminal de Raspberry Pi OS, ilustrando los comandos que se deben ejecutar y su resultado.

```
pi@raspberrypi : ~ $ sudo apt-get install dirmngr
```

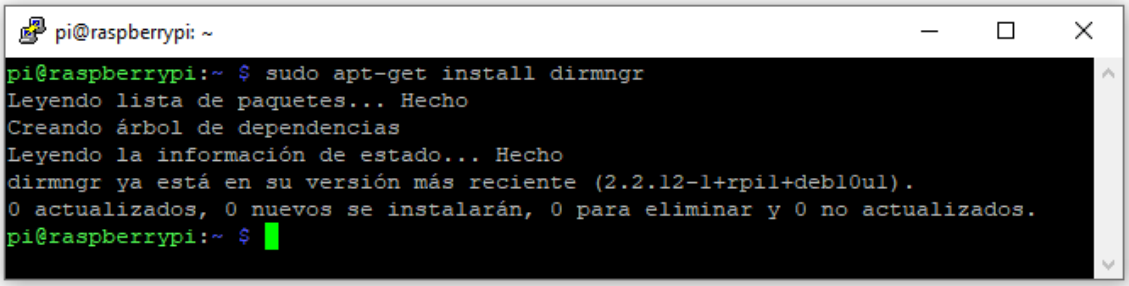

Figura 132. Ejecución del comando sudo *apt-get install dirmngr* en la terminal.

Fuente: Elaboración propia.

```
pi@raspberrypi : ~ $ sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 2C0D3C0F
```

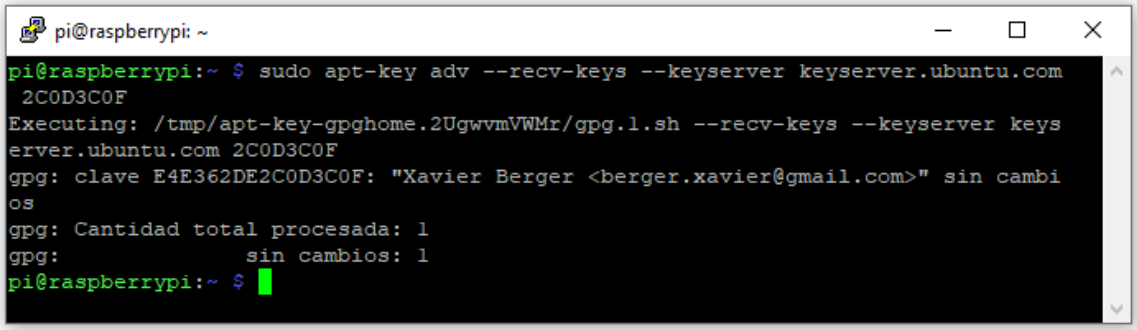

Figura 133. Instalación de claves públicas para el repositorio RPi-Monitor.

Fuente: Elaboración propia.

```
pi@raspberrypi : ~ $ sudo wget https://goo.gl/vewCLL -O /etc/apt/sources.list.d/rpimonitor.list
```

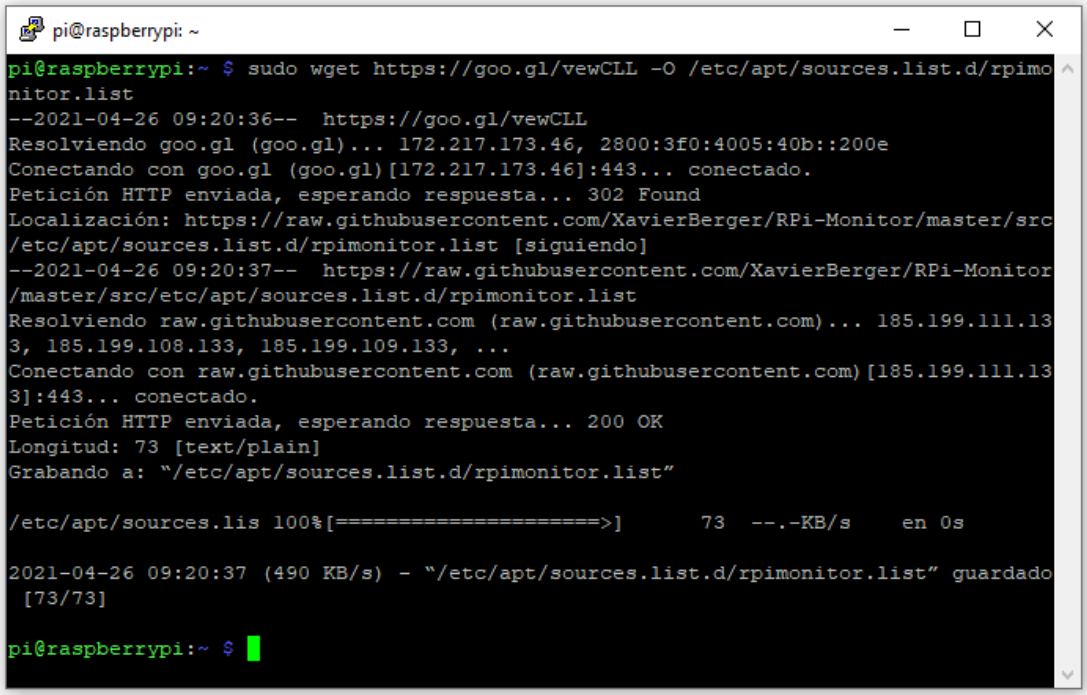

Figura 134. RPi-Monitor en la lista de repositorios.

Fuente: Elaboración propia.

```
pi@raspberrypi : ~ $ sudo apt update
```

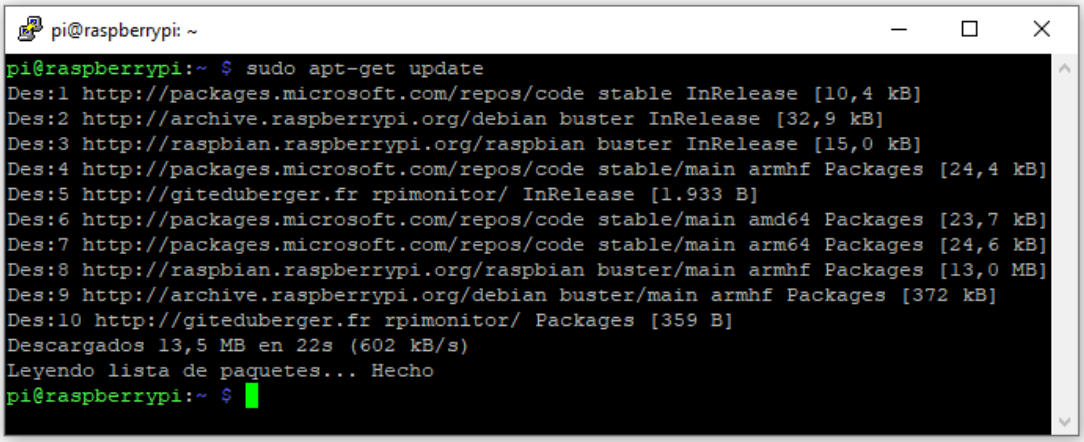

Figura 135. Actualización de paquetes RPi-Monitor.

Fuente: Elaboración propia.

```
pi@raspberrypi : ~ $ sudo apt install rpimonitor
```

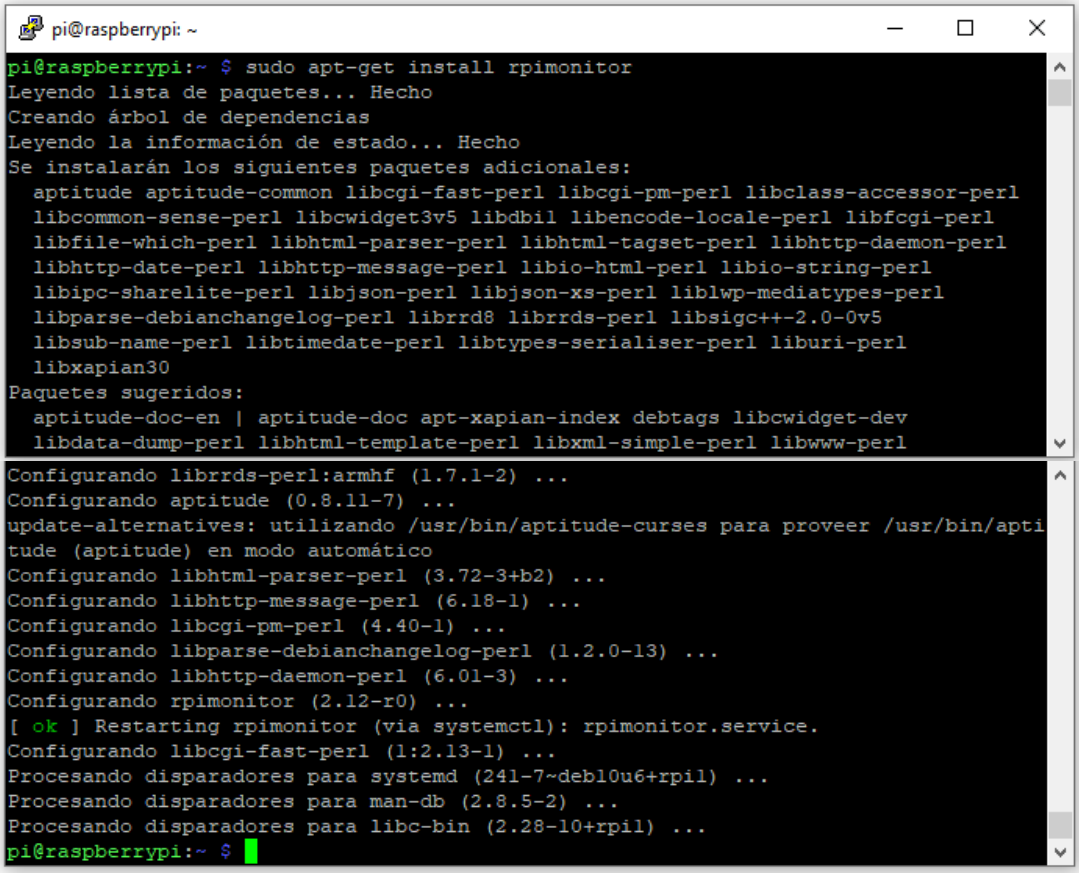

Figura 136. Instalación RPi-Monitor.

Fuente: Elaboración propia.

Listamos las configuraciones del directorio template de RPi-Monitor confirmando su correcta instalación.

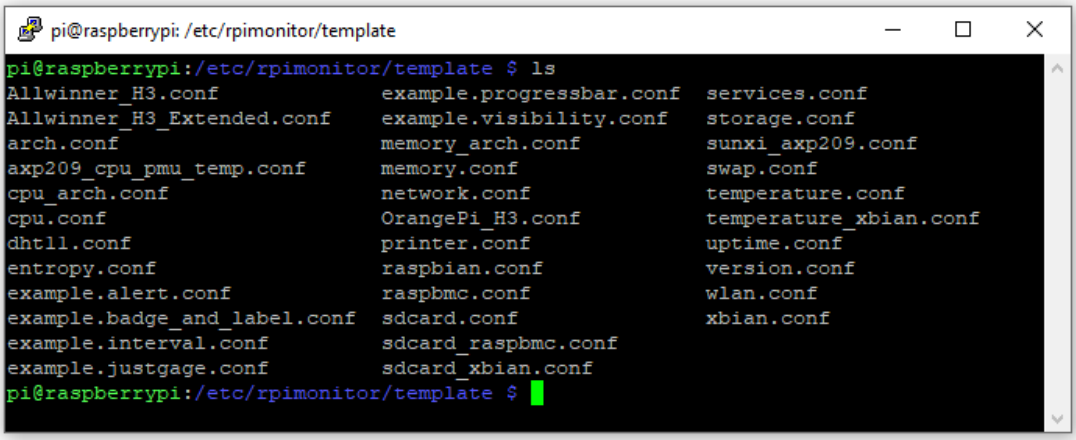

*Figura 137.*Servicios RPi-Monitor.

Fuente: Elaboración propia.

Una vez terminada la instalación de RPi-Monitor, verificamos la dirección IP del Gateway con el comando **ifconfig**, como se ilustra en la Figura 138:

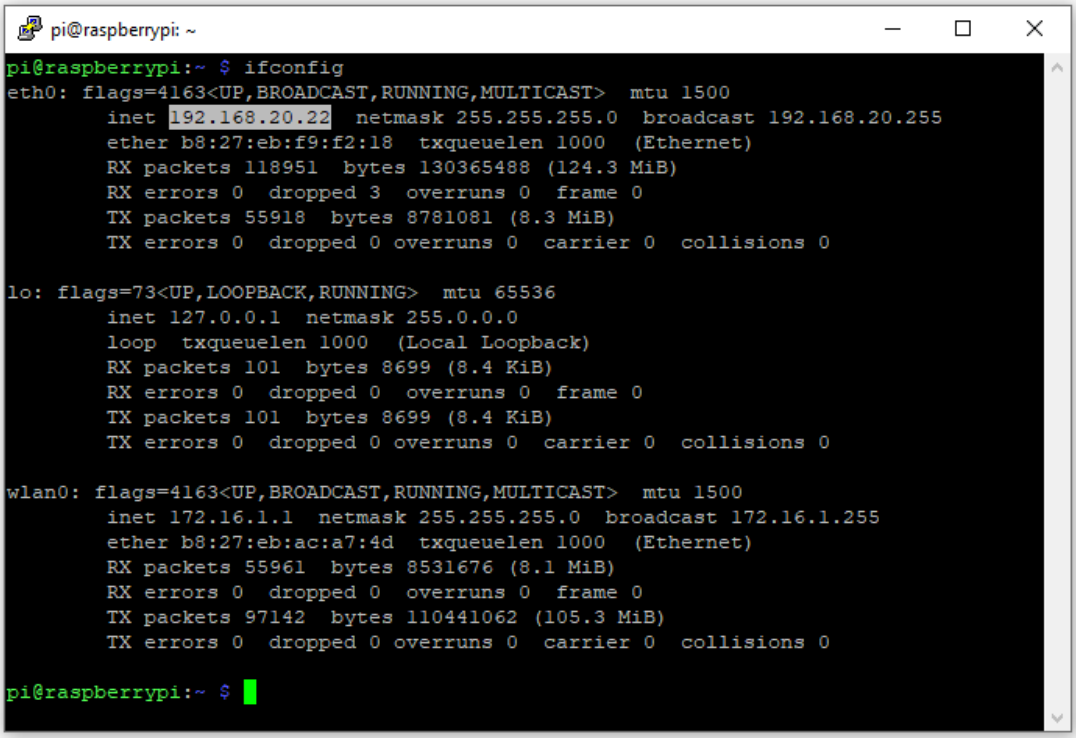

Figura 138. Dirección IP local del Gateway.

Fuente: Elaboración propia.

BIBLIOGRAFÍA

[1] C. Ferrera Fernández, “Agroclimatología,” 2002. [Online]. Available: http://www.carm.es/web/pagina?IDCONTENIDO=8376&IDTIPO=246&RASTRO=c226$m1259,20561. [Accessed: 05-Apr-2018].

[2] Instituto Interamericano de Cooperación para la Agricultura (IICA), Comisión Económica para América Latina y el Caribe (CEPAL), and Organización de las Naciones Unidas para la Agricultura y la Alimentación (FAO), “Perspectivas de la agricultura y del desarrollo rural en las Américas (2017-2018) | FAO,” 2017. [Online]. Available: http://www.fao.org/family-farming/detail/es/c/1052744/. [Accessed: 16-Apr-2018].

[3] M. R. Goyal and V. H. Ramirez Builes, *Elementos de Agroclimatología*. 2007.

[4] B. Kang and H. Choo, “An experimental study of a reliable IoT gateway,” *ICT Express*, Apr. 2017.

[5] A. Glória, F. Cercas, and N. Souto, “Design and implementation of an IoT gateway to create smart environments,” *Procedia Comput. Sci.*, vol. 109, pp. 568–575, Jan. 2017.

[6] S. Guoqiang, C. Yanming, Z. Chao, and Z. Yanxu, “Design and Implementation of a Smart IoT Gateway,” in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 2013, pp. 720–723.

[7] E. Peres *et al.*, “An autonomous intelligent gateway infrastructure for in-field processing in precision viticulture,” *Comput. Electron. Agric.*, vol. 78, no. 2, pp. 176–187, Sep. 2011.

[8] D. C. Yacchirema Vargas and C. E. Palau Salvador, “Smart IoT Gateway For Heterogeneous Devices Interoperability,” *IEEE Lat. Am. Trans.*, vol. 14, no. 8, pp. 3900–3906, Aug. 2016.

[9] M. Niño Zambrano, “Interacción Semántica de Objetos en la Web de las Cosas - Semantic Object Interaction on Web of Things,” Universidad del Cauca, 2016.

[10] Centro Nacional de Investigaciones de Café - CENICAFE, “Tecnología del cultivo del café, Interpretación de análisis de suelos para café,” no. Jun-1987, 1987.

[11] Centro Nacional de Investigaciones de Café - CENICAFE, *Anuario meteorológico cafetero*. CENICAFE, 2017.

[12] H. Park, H. Kim, and H. Joo, “Recent advancements in the Internet-of-Things related standards: A oneM2M perspective,” *ICT Express*, vol. 2, no. 3, pp. 126–129, Sep. 2016.

[13] D. Lund, C. Macgillivray, V. Turner, and M. Morales, “Worldwide and Regional Internet of Things (IoT) 2014 –2020 Forecast: A Virtuous Circle of Proven Value and Demand,” vol. 1, p. 29, 2014.

[14] D. Evans, *The Internet of Things: How the Next Evolution of the Internet is Changing Everything*, vol. 1. 2011.

[15] D. S. Watson, M. A. Piette, O. Sezgen, N. Motegi, and L. ten Hope, “Machine to machine (M2M) technology in demand responsive commercial buildings,” *2004 ACEEE Summer Study Energy Effic. Build. Pacific Grove, CA (US), 08/23/2004-08/27/2004*, Aug. 2004.

[16] J. Delgado Martín, “Adquisición de información y semántica en internet de las cosas: el problema de los objetos no conectados,” 2015.

[17] “50 mil millones de dispositivos estarán conectados en 2020, según reportes - Trends and Innovation.” [Online]. Available: https://www.galileo.edu/trends-innovation/50-mil-millones-de-dispositivos-estaran-conectados-en-2020-segun-

reportes/. [Accessed: 23-Nov-2020].
[18] F. Wortmann and K. Flüchter, “Internet of Things: Technology and Value Added,” *Business and Information Systems Engineering*, vol. 57, no. 3. 2015.
[19] F. James, R. Kurose, and W. Keith, *Redes de computadoras : un enfoque descendente*. Pearson Educación, 2010.
[20] Microsoft Azure, “Protocolos y Tecnologías de IoT.” [Online]. Available: https://azure.microsoft.com/es-es/overview/internet-of-things-iot/iot-technology-protocols/. [Accessed: 05-Feb-2021].
[21] A. S. Tanenbaum, *Redes de computadoras*. Pearson Educación, 2003.
[22] A. Bahga and V. Madisetti, *Internet of Things: A Hands-On Approach:* 2014.
[23] A. J. González García, “IoT: Dispositivos, tecnologías de transporte y aplicaciones,” Universitat Oberta de Catalunya, 2017.
[24] “Bluetooth® Technology Website.” [Online]. Available: https://www.bluetooth.com/. [Accessed: 06-Dec-2020].
[25] “LoRa Alliance.” [Online]. Available: https://lora-alliance.org/. [Accessed: 06-Dec-2020].
[26] “Sigfox - The Global Communications Service Provider for the Internet of Things (IoT).” [Online]. Available: https://www.sigfox.com/en. [Accessed: 09-Dec-2020].
[27] “Energy Harvesting Wireless Sensor Solutions and Networks from EnOcean | EnOcean.” [Online]. Available: https://www.enocean.com/. [Accessed: 09-Dec-2020].
[28] “KNX Association KNX Association [Official website].” [Online]. Available: https://www.knx.org/knx-en/for-professionals/index.php. [Accessed: 09-Dec-2020].
[29] “Z-Wave | Safer, Smarter Homes Start with Z-Wave.” [Online]. Available: https://www.z-wave.com/. [Accessed: 09-Dec-2020].
[30] “NB-IOT, Accelerating Cellular IOT.” [Online]. Available: https://www.huawei.com/minisite/hwmbbf15/en/nb-iot-accelerating-cellular-iot.html. [Accessed: 09-Dec-2020].
[31] A. Gupta, *The IoT Hacker’s Handbook*. Berkeley, CA: Apress, 2019.
[32] Red Hat Enterprise, “Red Hat Enterprise Linux 4: Manual de referencia, Capítulo 20. Protocolo SSH.”
[33] “About HTML5 WebSocket - Powered by Kaazing.” [Online]. Available: https://www.websocket.org/aboutwebsocket.html. [Accessed: 12-Dec-2020].
[34] “MQTT - The Standard for IoT Messaging.” [Online]. Available: https://mqtt.org/. [Accessed: 26-Oct-2020].
[35] “AMQP is an open Internet (or ‘wire’) Protocol standard for message-queuing communications. | AMQP.” [Online]. Available: https://www.amqp.org/product/overview. [Accessed: 12-Dec-2020].
[36] “ZeroMQ.” [Online]. Available: https://zeromq.org/. [Accessed: 12-Dec-2020].
[37] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, “A gap analysis of Internet-of-Things platforms,” *Comput. Commun.*, vol. 89–90, 2016.
[38] T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web" in Scientific American,” *Sci. Am. Mag.*, vol. 284, May 2001.
[39] S. Guerrero-Narváez, M.-Á. Niño-Zambrano, D.-J. Riobamba-Calvache, and G.-A. Ramírez-González, “Test Bed of Semantic Interaction of Smart Objects in the Web of Things,” *Futur. Internet*, vol. 10, no. 5, p. 42, May 2018.
[40] Y. A. Pabón Guerrero and L. J. Rojas Bolaños, “Modelo ontológico para manejo de perfiles de usuario en la IoT,” Universidad del Cauca, 2017.
[41] I. Torrez, J. Luna, and M. López Bonilla, “Metodologías y métodos para la construcción de ontologías Methodologies and methods for building ontologies,”

Sci. Tech., Apr. 2012.
[42] M. Serrano, P. Barnaghi, F. Carrez, P. Cousin, O. Vermesan, and P. Friess, “Internet of Things IoT Semantic Interoperability: Research Challenges, Best Practices, Recommendations and Next Steps. European Research Cluster on the Internet of Things (IERC),” p. 48, 2015.
[43] G. Yang *et al.*, “A Health-IoT Platform Based on the Integration of Intelligent Packaging, Unobtrusive Bio-Sensor, and Intelligent Medicine Box,” *IEEE Trans. Ind. Informatics*, vol. 10, no. 4, pp. 2180–2191, Nov. 2014.
[44] R. Lea and M. Blackstock, “City Hub: A Cloud-Based IoT Platform for Smart Cities,” in *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, 2014, pp. 799–804.
[45] A.-M. Rahmani *et al.*, “Smart e-Health Gateway: Bringing intelligence to Internet-of-Things based ubiquitous healthcare systems,” in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, 2015, pp. 826–834.
[46] M. Bauer *et al.*, *Internet of Things – Architecture IoT-A Deliverable D1.5 – Final architectural reference model for the IoT v3.0*. 2013.
[47] P. Desai, A. Sheth, and P. Anantharam, “Semantic Gateway as a Service Architecture for IoT Interoperability,” in *2015 IEEE International Conference on Mobile Services*, 2015, pp. 313–319.
[48] International Telecommunication Union (ITU), “Common requirements and capabilities of a gateway for Internet of Things applications,” p. 26, 2017.
[49] A. Castellani, S. Loreto, N. Bui, and M. Zorzi, *Quickly interoperable Internet of Things using simple transparent gateways*. 2011.
[50] Á. Brenes and V. F. Saborío, *Elementos de climatología: su aplicación didáctica a Costa Rica*, no. 1. EUNED, 1995.
[51] M. S. A. Rodríguez and C. Múñez, “Fundamentos de climatología,” 2004.
[52] Organización Meteorológica Mundial, “Guía de Prácticas Climatológicas,” vol. 100, 2011.
[53] H. N. María Luz, “La agroclimatología: instrumento de planificación agrícola,” 1993.
[54] Centro Nacional de Investigaciones de Café - Cenicafé, “Agroclima - Plataforma Agroclimática Cafetera.” [Online]. Available: https://agroclima.cenicafe.org/. [Accessed: 11-Jul-2019].
[55] A. Vakaloudis and C. O’Leary, “A framework for rapid integration of IoT Systems with industrial environments,” in *IEEE 5th World Forum on Internet of Things, WF-IoT 2019 - Conference Proceedings*, 2019, pp. 601–605.
[56] A. Nugur, M. Pipattanasomporn, M. Kuzlu, and S. Rahman, “Design and Development of an IoT Gateway for Smart Building Applications,” *IEEE Internet Things J.*, p. 1, 2018.
[57] S. K. Y. Donzia, H.-K. Kim, and H. J. Hwang, “A Software Model for Precision Agriculture Framework Based on Smart Farming System and Application of IoT Gateway,” in *Computational Science/Intelligence & Applied Informatics*, Springer, Cham, 2019, pp. 49–58.
[58] J. Jiménez Casas, “Sistema IoT para el control y monitorización de un terrario con Eclipse Kura, Amazon AWS y Angular,” Universidad de Sevilla, Departamento de Ingeniería Telemática, 2019.
[59] J. Kim, “Requirement of Security for IoT Application based on Gateway System,” *Int. J. Secur. Its Appl.*, vol. 9, pp. 201–208, Oct. 2015.
[60] C. Botta, L. Pierangelini, and L. Vollero, “IoT Gateways for Industrial and Medical Applications: Architecture and Performance Assessment,” in *2020 IEEE International Workshop on Metrology for Industry 4.0 and IoT, MetroInd 4.0 and IoT*

2020 - Proceedings, 2020, pp. 596–599.
[61] S. You, X. Li, and W. Chen, “A semantic mechanism for Internet-of-Things (IoT) to implement intelligent interactions,” in *2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOpt 2018*, 2018.
[62] S.-M. Kim, H.-S. Choi, and W.-S. Rhee, “IoT home gateway for auto-configuration and management of MQTT devices,” in *2015 IEEE Conference on Wireless Sensors (ICWiSe)*, 2015, pp. 12–17.
[63] E. Foundation, “Eclipse Kura - Open Source framework for IoT.” [Online]. Available: http://www.eclipse.org/kura/. [Accessed: 21-May-2018].
[64] D. Bravo, A. García, and W. Muñoz, “Diseño e Implementación de un Prototipo de Estación Meteorológica,” *Rev. Univ. en Telecomunicaciones Informática y Control*, vol. 1, no. 2, pp. 24–28, 2012.
[65] A. Sánchez García, “Desarrollo de un sistema electrónico para la monitorización de variables ambientales en huertos urbanos,” Universidad Politécnica de Cartagena, 2015.
[66] F. G. Vásconez Albán, “Diseño e implementación de un prototipo de estación de medición de parámetros meteorológicos para su incorporación en una red extendida de monitorización ambiental,” 2009.
[67] P. Y. Muck and M. J. Homam, “IoT Based Weather Station Using Raspberry Pi 3,” *Int. J. Eng. Technol.*, vol. 7, no. 4.30, p. 145, Nov. 2018.
[68] D. Fidalgo Moreno, “Framework para integración de redes de sensores y actuadores con plataformas de Internet de las Cosas,” 2016.
[69] J. Krishnan and S. K. Vasudevan, “An Extensive Survey on IoT Smart Gateways, Software Architecture, Related Protocols and Challenges,” in *Proceedings - International Conference on Vision Towards Emerging Trends in Communication and Networking, ViTECoN 2019*, 2019.
[70] A. M. Rahmani *et al.*, “Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach,” *Futur. Gener. Comput. Syst.*, 2018.
[71] Y. Lee and S. Nair, “A Smart Gateway Framework for IOT Services,” in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2016, pp. 107–114.
[72] International Telecommunication Union (ITU), “Y.2076 Requisitos y marco semánticos de la Internet de las Cosas,” 2016.
[73] “Teach, Learn, and Make with Raspberry Pi.” [Online]. Available: https://www.raspberrypi.org/. [Accessed: 15-Dec-2020].
[74] “Arduino Uno Rev3 | Arduino Official Store.” [Online]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Accessed: 29-Jan-2021].
[75] “NodeMcu - An open-source firmware based on ESP8266 wifi-soc.” [Online]. Available: https://www.nodemcu.com/index_en.html. [Accessed: 31-Jan-2021].
[76] “Digi XBee Ecosystem | Everything You Need to Explore and Create Wireless Connectivity | Digi International.” [Online]. Available: https://www.digi.com/xbee. [Accessed: 01-Feb-2021].
[77] “DHT22 Temperature and Humidity Sensor-DFRobot.” [Online]. Available: https://www.dfrobot.com/index.php?route=product/product&product_id=1102&search=SEN0137&description=true. [Accessed: 01-Feb-2021].
[78] “Adaptador Ethernet Gigabit USB 3.0 USB3GIG de Linksys.” [Online]. Available: https://www.linksys.com/co/p/P-USB3GIG/. [Accessed: 23-Feb-2021].
[79] “Eclipse desktop & web IDEs | The Eclipse Foundation.” [Online]. Available:

https://www.eclipse.org/ide/. [Accessed: 24-Feb-2021].
[80] “Home | Geany.” [Online]. Available: https://www.geany.org/. [Accessed: 24-Feb-2021].
[81] “PyCharm: el IDE de Python para desarrolladores profesionales, por JetBrains.” [Online]. Available: https://www.jetbrains.com/es-es/pycharm/. [Accessed: 24-Feb-2021].
[82] “Software | Arduino.” [Online]. Available: https://www.arduino.cc/en/software. [Accessed: 24-Feb-2021].
[83] Eclipse Foundation, “Eclipse Kura Documentation.” [Online]. Available: https://eclipse.github.io/kura/. [Accessed: 26-Jul-2019].
[84] W. Bowers, “OSGi: Simplifying the IoT Gateway,” in *EclipseCon Europe and OSGI Community Event 2015*, 2015.
[85] “Información general sobre AWS IoT Core – Amazon Web Services.” [Online]. Available: https://aws.amazon.com/es/iot-core/?nc=sn&loc=0. [Accessed: 24-Feb-2021].
[86] “Empowering App Development for Developers | Docker.” [Online]. Available: https://www.docker.com/. [Accessed: 24-Feb-2021].
[87] “IEEE - Acerca de IEEE.” [Online]. Available: https://www.ieee.org.co/acerca-de-ieee.php. [Accessed: 06-Dec-2020].
[88] “Sobre la Unión Internacional de Telecomunicaciones (UIT).” [Online]. Available: https://www.itu.int/es/about/Pages/default.aspx. [Accessed: 06-Dec-2020].
[89] “IETF | About.” [Online]. Available: https://www.ietf.org/about/. [Accessed: 06-Dec-2020].
[90] “Who We Are | Wi-Fi Alliance.” [Online]. Available: https://www.wi-fi.org/who-we-are. [Accessed: 12-Dec-2020].