LABORATORIO REMOTO DE IOT - LRIOT

PROYECTO

Versión 1.0

15/5/2024

INFORMACIÓN DEL PROYECTO

Autor / Autores	
Nombre Completo del integrante 1	Mariano Agustín Zapata
Legajo	44277
e-mail	marianozapata10@gmail.com
Nombre Completo del integrante 2	Gabriel Germán Ambrosio
Legajo	44251
e-mail	ambrosiogabrielgerman@gmail.com

Tutor	Gustavo Mercado, Ing. Electrónico
Director	Gustavo Mercado, Ing. Electrónico
Jurado	Mario Sebastián Tobar, Ing. Electrónico
Año Académico	2024
Responsable de la cátedra	Ana Lattuca, Ing. Electrónica

Empresa / Cliente / Laboratorio	Gridtics
Patrocinador (Sponsor)	Gridtics

1 RESUMEN DEL PROYECTO

1.1 RESUMEN

El proyecto "Laboratorio Remoto IoT" surge como respuesta al creciente interés en la adopción de dispositivos IoT en diversos sectores, especialmente en el ámbito industrial. Según un estudio de IDC, se estima que para el año 2025 habrá más de 55,7 millones de dispositivos IoT conectados a internet, lo que subraya la necesidad de contar con sistemas y herramientas que permitan aprender y desarrollar aplicaciones efectivas en este campo.

Este laboratorio responde a la mencionada demanda, dirigido tanto a usuarios individuales como a universidades y pequeñas empresas interesadas en profundizar sus conocimientos y capacidades en el desarrollo de sistemas IoT. Ofrece un entorno completo para el estudio y desarrollo de aplicaciones para estos dispositivos, proporcionando acceso a placas IoT de la familia openmote-cc2853 y permitiendo a los usuarios probar y desarrollar programación de firmware, así como evaluar la efectividad de las comunicaciones entre dispositivos.

Ubicado en la Universidad Tecnológica Nacional - Facultad Regional Mendoza, el laboratorio está diseñado para ser accesible globalmente, con disponibilidad continua las 24 horas del día, los 7 días de la semana. Su objetivo principal es fomentar la innovación y el conocimiento especializado en IoT, formando profesionales capacitados y consolidando la universidad como un actor clave en este campo. Además, busca impulsar proyectos innovadores y atraer posibles patrocinadores para su expansión y desarrollo continuo.

1.2 SUMMARY

"The 'Remote IoT Lab' project arises as a response to the increasing demand for the adoption of IoT devices across various sectors, especially in the industrial domain. According to an IDC study, it is estimated that by the year 2025, there will be over 55.7

million IoT devices connected to the internet, highlighting the need for systems and tools that enable learning and effective application development in this field.

This lab addresses the aforementioned demand, catering to both individual users and universities and small businesses interested in enhancing their knowledge and capabilities in IoT system development. It provides a comprehensive environment for studying and developing applications for these devices, granting access to IoT boards from the openmote-cc2853 family and allowing users to test and develop firmware programming, as well as evaluate the effectiveness of communications between devices.

Situated at the National Technological University - Regional Faculty of Mendoza, the lab is designed to be globally accessible, operating continuously 24 hours a day, 7 days a week. Its main objective is to foster innovation and specialized knowledge in IoT, training skilled professionals and solidifying the university's role as a key player in this field. Additionally, it aims to drive innovative projects and attract potential sponsors for its expansion and ongoing development."

2 PALABRAS CLAVES

Laboratorio remoto, openmote, IoT/IIoT, RaspberryPI, CoAP, MQTT, IPv6, IEEE802.15.4, 6LoWPAN

3 ÍNDICE

4 INTRODUCCIÓN

4.1 IDEA Y DESCRIPCIÓN DEL PROYECTO

Se plantea la creación de un laboratorio remoto de IoT que abarcará el estudio y desarrollo de dispositivos IoT. Este se ubicará en la Universidad Tecnológica Nacional – Facultad Regional Mendoza, aunque estará accesible desde cualquier ubicación con conexión a internet.

El proyecto tiene como objetivo principal servir como plataforma para la investigación y desarrollo en el campo de IoT, así como también para la divulgación de avances y tecnologías en este ámbito. Además, se contempla su potencial como fuente de ingresos a través del alquiler del laboratorio a empresas y startups interesadas.

En términos de sus entregables, el proyecto aborda tanto aspectos de producto como de servicio. En el ámbito de producto, el laboratorio mismo suple esta característica, mientras que en el aspecto de servicio se destaca su capacidad para facilitar investigaciones futuras en el área de IoT.

La documentación asociada incluye tres manuales esenciales: uno destinado a usuarios administradores; uno para usuarios finales y un manual detallado sobre la arquitectura de los dispositivos IoT utilizados.

Una característica clave del proyecto es su capacidad para permitir la conexión remota mediante herramientas como Apache Guacamole, lo cual facilita la programación del firmware en dispositivos específicos como las placas openmote-cc2538.

Asimismo, se prevé la posibilidad de expansión del laboratorio tanto vertical como horizontalmente, lo que implica la incorporación de nuevas placas IoT o la utilización de diferentes modelos para realizar pruebas exhaustivas en términos de modelos, protocolos y arquitecturas IoT. Esto último contribuye a determinar la mejor opción para proyectos específicos en el campo de IoT.

El funcionamiento del proyecto se resume en la siguiente imagen:

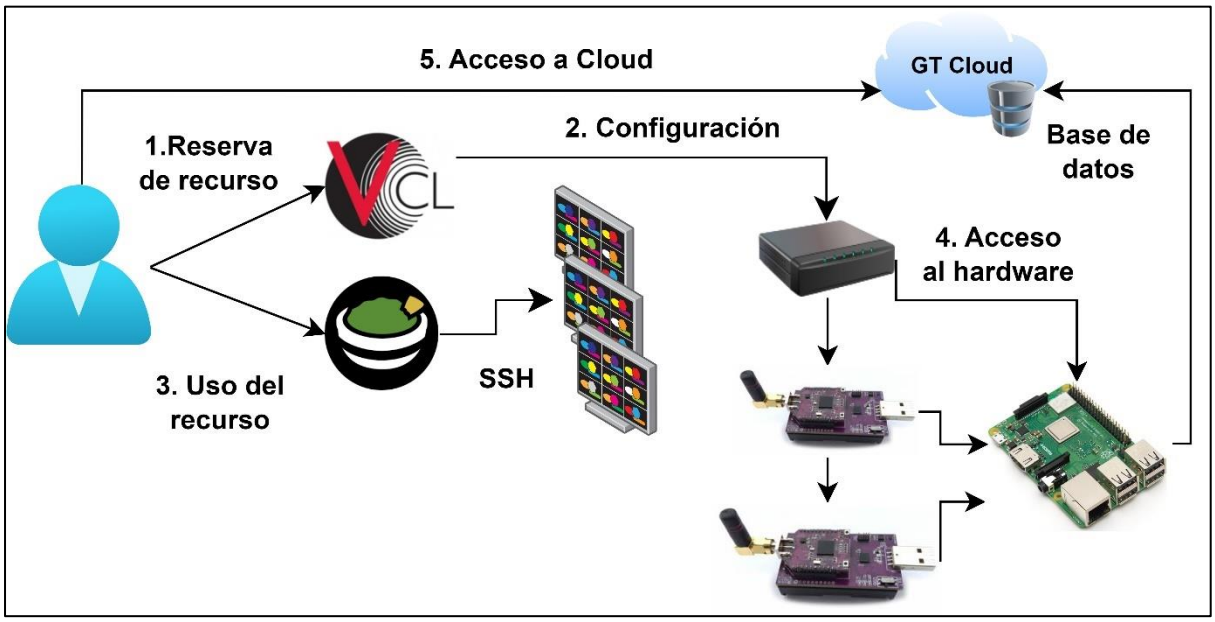

Figura 1: Representación general del proyecto

En la imagen se observa que en una primera instancia el usuario debe realizar la reserva del recurso, lo que desencadena una serie de configuraciones y comunicaciones que se desarrollan internamente en el servidor y son transparentes para el usuario final. Si la reserva del recurso es exitosa, el usuario se conecta a través del emulador de terminal mediante ssh al entorno de desarrollo que reservó en el paso uno.

Mediante este entorno de desarrollo el usuario tiene acceso al hardware que consiste en cuatro placas openmote y una placa raspberry PI. El formato de esta topología se basa en las utilizadas en la industria. En este punto el usuario puede realizar la programación del firmware y cargarla en la placa que desee.

Por último, se puede conectar a una base de datos que se encuentra en un servidor del grupo de estudio GRIDTICS, para extraer estos datos y visualizarlos en Grafana.

4.1.1 Objetivo general

El objetivo del proyecto fue lograr la construcción, configuración y programación de un laboratorio que permita poner en práctica los conocimientos de IoT, permitiendo a los estudiantes desarrollar los conocimientos apropiados en esta área. Este laboratorio posee un entorno de programación, un sistema de gestión de usuarios y recursos, un sistema de monetización del uso de recursos y hardware que permite conectar los sensores con la nube (alcance por fuera del presente proyecto).

Para lograrlo se empleó un servidor en donde se almacenaron las máquinas virtuales, entre los que se destacan los entornos de programación, el gestor de recursos y otras relacionadas con la disponibilidad y seguridad del sistema en su conjunto. Para el hardware se utilizaron cuatro placas openmote en conjunto con una placa raspberryPI. El entorno de programación y la presentación de los datos obtenidos se visualizan vía web mediante Grafana que se encuentra por fuera de las configuraciones realizadas.

4.2 JUSTIFICACIÓN DEL PROYECTO

4.2.1 Antecedentes del proyecto

El desarrollo del producto surgió para dar respuesta a una problemática observada en la universidad tecnológica nacional - facultad regional Mendoza (UTN – FRM). Actualmente se cuenta con un laboratorio remoto de RTOS, sin embargo este solamente tiene como objetivo desarrollar conocimientos de sistemas operativos en tiempo real, el cual es un campo diferente al de los sistemas IoT. Ante esta brecha en las competencias de los estudiantes, surgió la necesidad de desarrollar un nuevo laboratorio para ampliar los conocimientos en esta área de la Electrónica.

Por lo tanto el LRIOT surgió como respuesta a esta problemática al ofrecer un entorno específico para el aprendizaje y práctica de IoT. Además, se implementó un sistema de suscripción de pagos que generará ingresos al laboratorio, permitiendo financiar las actualizaciones y mantenimiento tecnológico necesario para aplicarle actualizaciones de forma periódica y convertirlo en un sistema competitivo en el mercado.

En resumen, los antecedentes del proyecto reflejan la demanda identificada en la universidad y cómo el proyecto busca solucionarla, al mismo tiempo que se proyecta mantenerse actualizado y relevante en el ámbito de las tecnologías emergentes como IoT.

4.2.2 Estado actual

Los laboratorios remotos no presentan una innovación tecnológica en si mismos, son desarrollos ampliamente estudiados y que abarcan una gran variedad de tecnologías y arquitecturas de comunicación. Sin embargo, solamente se pudo identificar un laboratorio existente que se enfoca en la enseñanza de IoT desde un enfoque más didáctico, utilizando placas ESP32 provistas por los alumnos, programadas con micropython y limitadas a la comunicación MQTT. Este laboratorio funciona como una nube MQTT para prácticas de publicación y suscripción de tópicos.

El LRIOT cuenta con la innovación de permitir la programación de dispositivos IoT sin necesidad de contar con dichas placas por parte de los estudiantes, programación en C (optimizado para dispositivos IoT), conocimiento de sistema operativo Contiki-NG (ampliamente utilizado en la industria), uso de CoAP y MQTT, conocimiento de arquitecturas IoT a muy bajo nivel, posibilidad de generar un alquiler por el uso de la plataforma, entre otras características.

Es importante destacar que el proyecto tiene una dimensión social y empresarial significativa, ya que busca no solo divulgar la tecnología IoT, sino también proporcionar herramientas y conocimientos avanzados para estudiantes, profesionales y emprendedores en este campo.

4.2.3 Necesidad del negocio y definición del problema

El proyecto surgió de la necesidad de contar con un laboratorio de dispositivos IoT que permita a los alumnos de técnicas digitales de UTN-FRM realizar pruebas y experimentos en este campo. Posteriormente a esta problemática inicial, se planteó la ampliación del alcance del laboratorio para divulgar la tecnología IoT a otras instituciones educativas,

empresas y escuelas del entorno.

En términos específicos, el problema a resolver radicó en la falta de un espacio adecuado para la práctica y experimentación en dispositivos IoT dentro de la universidad. Esta carencia limita el desarrollo de habilidades y conocimientos en esta área, tanto para los estudiantes como para el personal académico. Asimismo, se identificó la oportunidad de ofrecer este laboratorio como un servicio alquilable para pequeñas empresas y pymes interesadas en implementar dispositivos IoT en sus entornos laborales.

Por lo tanto, el objetivo principal del proyecto fue cubrir la necesidad de un laboratorio especializado en IoT para la realización de pruebas y experimentos, así como brindar la oportunidad de alquiler del laboratorio para uso externo, generando así ingresos económicos para la facultad y el laboratorio GRIDTICS.

4.2.4 Beneficios del proyecto

El proyecto está alineado con un plan para lograr un futuro mejor y más sostenible para todos, para ello se plantea cumplir con los diferentes objetivos de desarrollo sustentable, entre los que se encuentran:

1. OSD 4 – Educación de calidad: El laboratorio remoto puede proporcionar acceso a la educación científica de calidad, permitiendo a estudiantes de todo el mundo participar en experimentos y prácticas virtuales.

2. ODS 9 – Industria, innovación e infraestructura: El proyecto puede fomentar la innovación en el campo de la educación científica y tecnológica, así como en el desarrollo de tecnologías de laboratorio remoto más avanzadas.

3. ODS 10 – Reducción de las desigualdades: Al ofrecer acceso remoto a un laboratorio, se pueden reducir las desigualdades en el acceso a la educación científica entre regiones o comunidades con recursos limitados.

4. ODS 11 – Ciudades y comunidades sostenibles: Los laboratorios remotos pueden ayudar a descentralizar la educación científica, permitiendo a las personas acceder a oportunidades de aprendizaje sin tener que trasladarse a centros educativos lejanos.

5. ODS 17 – Alianzas para lograr los objetivos: El proyecto de laboratorio remoto puede fomentar la colaboración entre diferentes universidades, instituciones y países, promoviendo la transferencia de conocimientos y el intercambio de recursos.

Adicionalmente, la facultad y el grupo GRIDTICS obtendrán beneficios adicionales al contar con un laboratorio que permitirá la realización de cursos y talleres avanzados sobre tecnologías IoT, así como la generación de ingresos económicos a través de consultorías a empresas y el alquiler o venta del laboratorio como una unidad completa. Estos beneficios fortalecerán la capacidad de la organización, atraerán patrocinadores y crearán valor para todos los stakeholders involucrados en el proyecto.

4.3 ALCANCE

4.3.1 Alcance

El proyecto se enfocó en desarrollar un dispositivo funcional que abarcó diversas áreas técnicas, desde el diseño de hardware hasta la configuración de redes. Se estableció un entorno de desarrollo para conectar nodos sensores, incluyendo la instalación de Contiki-NG, un compilador y la realización de pruebas de firmware. Posteriormente, se configuró un nodo coordinador con Raspberry Pi y OpenMote-CC2538, realizando pruebas de conectividad y programación para los protocolos CoAP y MQTT. Se implementó una infraestructura virtualizada con reserva de recursos y una interfaz de interacción además de elemento transparentes al usuario final pero que ayudan al correcto funcionamiento del sistema (firewall, proy reverso, servidor de archivo, entre otros). Se incluyó soporte físico para colocar las placas openmote y RaspberryPi de manera segura, además de incorporar una cámara para realizar transmisiones en tiempo real a través de un canal de YouTube. Finalmente se entregaron manuales detallados para administradores y usuarios. El primero de ellos contiene información referida a creación de usuarios, anexo de estos al dominio y sus permisos. El segundo manual especifica como el usuario debe realizar la reserva de los entornos de desarrollo y la programación y carga de firmware en las placas openmote-cc2538.

4.3.2 Límites o fuera de alcance

En el entregable del proyecto solamente se permitirá que un usuario utilice los recursos disponibles, el desarrollo para múltiples usuarios conectados al mismo tiempo quedó fuera del alcance del proyecto.

La programación y carga del firmware del laboratorio será solamente por la interfaz CLI (command line interface) sin contar con una interfaz gráfica o GUI.

4.3.3 Soluciones y entregables principales

La siguiente tabla muestra un listado de los entregables del proyecto (productos o servicios)

Entregables principales	Descripción del entregable
Nodos sensores	Bloque encargado de la comunicación entre las placas IoT y comunicación hacía el nodo coordinador.
Nodo coordinador	Bloque encargado de recibir datos desde los nodos sensores y conectarlos a red local.
Máquinas virtuales	Incluye la estructura de reserva e interacción. Así como sistemas adicionales transparentes al usuario pero que son necesarios para el correcto funcionamiento (firewall, proxy inverso, servidor de archivos, entre otros.)
Documentación	Consiste en una serie de manuales referidos a programación por parte del usuario, configuraciones del administrador y la arquitectura de comunicaciones utilizada por las placas IoT.

Tabla 1: Entregables principales

4.4 PLANIFICACIÓN DEL PROYECTO

El proyecto se llevó a cabo dentro del marco de trabajo del laboratorio GRIDTICS, con la tutoría técnica proporcionada por el laboratorio y la colaboración de profesores de diversas cátedras relacionadas con las técnicas digitales. Se estructuró en varios bloques que requirieron pruebas de validación para garantizar su correcto funcionamiento y su integración.

En cuanto a la estrategia de procesos, se empleó inicialmente la metodología AGILE, que permitió desarrollar simultáneamente el hardware, el software y el firmware. Una vez que se logró un punto de coincidencia entre los tres grupos de desarrollo, se implementó la metodología PMI. Esto se debió a que la integración de estos componentes requería seguir una secuencia de pasos que debían iterarse y corregirse en caso de fallos antes de avanzar al siguiente paso.

4.4.1 Cronograma

						sep-23				oct-23				nov-23				dic-23			
						SEMANA				SEMANA				SEMANA				SEMANA			
ID	TAREA	FECHA DE INICIO	FECHA DE FINALIZACIÓN	HORAS DESTINADAS	PCT DE LA TAREA	1°	2°	3°	4°	5°	6°	7°	8°	9°	10°	11°	12°	13°	14°	15°	16°
1	Conexión de motes - Gateway	2/9/2023	10/10/2023	60 horas	100%																
2	Conexión de Gateway - Servidor	2/9/2023	24/10/2023	60 horas	100%																
3	Disposición de motes y configuración de sensores	25/10/2023	16/11/2023	120 horas	100%																
4	Configuración de gestor de usuarios	17/11/2023	30/11/2023	100 horas	100%																
5	Configuración de máquinas virtuales	1/12/2024	2/2/2024	420 horas	100%																
6	Configuración de redes y segmentación	3/2/2024	9/2/2024	40 horas	100%																
7	Desarrollo de sistema de monetización	10/2/2024	22/2/2024	40 horas	100%																
8	Pruebas del laboratorio en su totalidad, arreglos necesarios y anexo de dispositivos externos	25/2/2024	5/4/2024	300 horas	100%																
9	Desarrollo de documentación y entregables finales	5/4/2024	3/5/2024	100 horas	100%																

Tabla 2: Diagrama de Gantt - Parte 1

						ene-24				feb-24				mar-24				abr-24				may-24			
						SEMANA				SEMANA				SEMANA				SEMANA				SEMANA			
ID	TAREA	FECHA DE INICIO	FECHA DE FINALIZACIÓN	HORAS DESTINADAS	PCT DE LA TAREA	17°	18°	19°	20°	21°	22°	23°	24°	25°	26°	27°	28°	29°	30°	31°	32°	33°	34°	35°	36°
1	Conexión de motes - Gateway	2/9/2023	10/10/2023	60 horas	100%																				
2	Conexión de Gateway - Servidor	2/9/2023	24/10/2023	60 horas	100%																				
3	Disposición de motes y configuración de sensores	25/10/2023	16/11/2023	120 horas	100%																				
4	Configuración de gestor de usuarios	17/11/2023	30/11/2023	100 horas	100%																				
5	Configuración de máquinas virtuales	1/12/2024	2/2/2024	420 horas	100%																				
6	Configuración de redes y segmentación	3/2/2024	9/2/2024	40 horas	100%																				
7	Desarrollo de sistema de monetización	10/2/2024	22/2/2024	40 horas	100%																				
8	Pruebas del laboratorio en su totalidad, arreglos necesarios y anexo de dispositivos externos	25/2/2024	5/4/2024	300 horas	100%																				
9	Desarrollo de documentación y entregables finales	5/4/2024	3/5/2024	100 horas	100%																				

Tabla 3: Diagrama de Gantt Parte 2

Por lo tanto, el tiempo total necesario para realizar el proyecto es de 1240 horas hombre.

4.4.2 Hitos

La tabla muestra un listado de hitos generales del proyecto y el cronograma estimado de finalización:

Hitos	Fecha de finalización
Conexión de motes – Gateway	10/10/2023
Conexión Gateway – Servidor	24/10/2023
Disposición de motes y configuración de sensores	16/11/2023
Configuración de gestor de usuarios	30/11/2023
Configuración de máquinas virtuales	2/2/2024
Configuración de redes y segmentación	9/2/2024
Desarrollo de sistema de monetización	22/2/2024
Pruebas del laboratorio en su totalidad, arreglos necesarios y anexo de dispositivos externos	5/4/2024
Desarrollo de documentación y entregables finales	3/5/2024

Tabla 4: Hitos principales

4.5 Riesgo

Riesgo	Mitigación
Aumento del costo de los materiales	-Comprar materiales lo antes posible. -Rediseñar el sistema para hacerlo más económico. -Establecer un presupuesto de emergencia teniendo en cuenta la inflación.

Disminución del costo de los materiales.	-No es necesario realizar alguna maniobra de mitigación.
Proveedor no dispone de los recursos en tiempo y forma.	-Buscar otro proveedor. -Comenzar las etapas posteriores del proyecto. -Comenzar desarrollo de manuales de usuario.
Requerimiento de proyecto en tiempo mayor al propuesto.	-No es necesario realizar alguna maniobra demitigación.
Requerimiento de proyecto en tiempo menor al proyecto.	-Ajustar el diagrama de tiempos. -Desarrollar tareas en paralelo. -Avisar con antelación que no se logrará el objetivo pedido y proponer una fecha intermedia deculminación del proyecto.
Ampliación de lo inicialmente diseñado.	-Reestructurar tiempo y presupuesto y en función deello, comunicar las modificaciones pertinentes. -Notificar que se excede los objetivos anteriormente acordados.
Disminución de lo inicialmente diseñado.	-No es necesario realizar alguna maniobra de mitigación.

Tabla 5: Tabla de riesgo

Para realizar el análisis de riesgos se deben tener en cuenta 2 variables:

- Frecuencia de ocurrencia del evento.
- Impacto del evento en las operaciones de la empresa.

La escala utilizada para determinar los valores de la frecuencia de los eventos se detalla a continuación. Un evento que no se presente en un período de un año, es considerado Bajo y se le asignar un valor de 1. Por su parte, un riesgo medio, de valor 2, es aquel que puede ocurrir entre 2 meses y menos de 1 año. Finalmente, los riesgos altos son aquellos

que tienen una frecuencia menor de 2 meses, con un valor de 3.

Frecuencia del Riesgo	Valor	Descripción
Bajo	1	Más de 1 año.
Medio	2	Entre 2 meses y 1 año.
Alto	3	Menor a 2 meses.

Tabla 6: Frecuencia de Riesgo

La frecuencia de los riesgos identificados se describe luego para medir el impacto en la operación si se materializa alguno de los riesgos.

Riesgo	Frecuencia del evento		
	Alto	Medio	Bajo
Aumento del costo de los materiales	3		
Disminución del costo de los materiales.			1
Proveedor no dispone de los recursos en tiempo y forma.		2	
Requerimiento de proyecto en tiempo mayor al propuesto.			1
Requerimiento de proyecto en tiempo menor al proyecto.			1
Ampliación de lo inicialmente diseñado.		2	
Disminución de lo inicialmente diseñado.			1

Tabla 7: Frecuencia de eventos

Se determina un valor Alto, Medio o Bajo, según la severidad del impacto. Donde Bajo significa que el impacto es mínimo; la empresa puede seguir operando con normalidad,

aunque el riesgo se materializase. Medio significa que el negocio puede continuar, pero posiblemente tenga algún impacto en facturación o puede retrasar las funciones principales de la empresa. Y finalmente Alto significa que el impacto es sustancial; las operaciones se ven afectadas y causando afectación a los clientes.

Impacto del Riesgo	Valor	Descripción
Bajo	1	Impacto mínimo sobre costo, tiempo o técnico.
Medio	2	Algún impacto sobre costo, tiempo o técnico.
Alto	3	Impacto sustancial sobre costo, tiempo o técnico.

Tabla 8: Impacto del Riesgo

Basados en estos valores se clasifican los riesgos según su impacto (materialización del riesgo) en las operaciones.

Riesgo	Frecuencia del evento		
	Alto	Medio	Bajo
Aumento del costo de los materiales		2	
Disminución del costo de los materiales.			1
Proveedor no dispone de los recursos en tiempo y forma.		2	
Requerimiento de proyecto en tiempo mayor al propuesto.			1
Requerimiento de proyecto en tiempo menor al proyecto.	3		
Ampliación de lo inicialmente diseñado.	3		
Disminución de lo inicialmente diseñado.			1

Tabla 9: Frecuencia de Eventos

Como se puede observar existen riesgos con valores altos, los cuales de materializarse afectan de manera significativa a la empresa.

Por último, para determinar la dominancia del riesgo, se suman los elementos de las tablas de Frecuencia e Impacto del Evento de cada uno de los riegos. A partir de los resultados, los riesgos se categorizan en Alto, Medio o Bajo.

Nivel de Riesgo	Valor
Bajo	1 - 2
Medio	3 - 4
Alto	5 - 6

Tabla 1: Nivel de Riesgo

Riesgo	Frecuencia del evento		
	Alto	Medio	Bajo
Aumento del costo de los materiales	5		
Disminución del costo de los materiales.			2
Proveedor no dispone de los recursos en tiempo y forma.		4	
Requerimiento de proyecto en tiempo mayor al propuesto.			2
Requerimiento de proyecto en tiempo menor al proyecto.		4	
Ampliación de lo inicialmente diseñado.	5		
Disminución de lo inicialmente diseñado.			2

Tabla 2: Matriz de Riesgo

Los riesgos más altos están relacionados con el aumento del costo de los materiales y ampliación de lo que se definió en el acta de inicio.

Las posibles a cada uno de los riesgos están definidos en la primera tabla de esta sección.

5 DESARROLLO DEL PROYECTO

5.1 DESARROLLO TÉCNICO

El proyecto se basa en la integración de varios módulos y tecnologías para facilitar la reserva, conexión y programación de placas openmote, junto con la carga de firmware en dichas placas. Se requieren diversos recursos físicos, incluyendo un servidor, cuatro placas openmote-cc2538, una Raspberry Pi y una cámara IP.

La representación física de todos los recursos físicos involucrados puede visualizarse mediante la siguiente imagen:

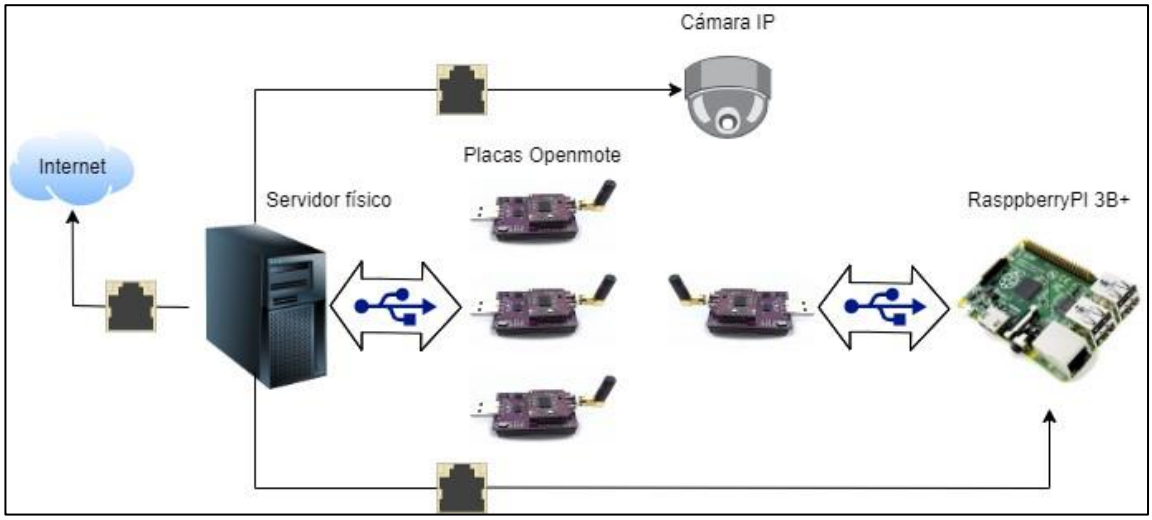

Figura 2: Recursos físicos involucrados

A continuación, se expone el desarrollo técnico del proyecto.

5.1.1 Requisitos de recursos físicos

5.1.1.1 Servidor

La configuración del sistema de reserva e interacción con el usuario requiere la presencia de un servidor físico como componente central. Este servidor debe ser capaz de interactuar con las placas openmote a través de los puertos USB, así como con la placa Raspberry Pi y la cámara mediante las interfaces de red correspondientes.

- CPU(s): 6 x AMD Fx-6300 CPU @ 3.8 GHz (1 socket)
- RAM: 16384 MB
- SSD: /dev/sda/ 480 GB
- Al menos 4 interfaces USB.
- 3 Interfaces de red (En caso de poseer solamente una, agregar dos puertos USB y utilizar conversor USB-ethernet).

5.1.1.2 Placas IIoT

Las placas utilizadas son de la familia openmote, concretamente la versión cc2538 que cuenta con las siguientes características:

- Utiliza el microcontrolador CC2538 ARM-Cortex-M3 de Texas Instrument. Puede operar de forma simultánea en las bandas de 2.4 GHz y la banda ISM de 868/915 MHz con un soporte completo del standard IEEE 802.15.4.

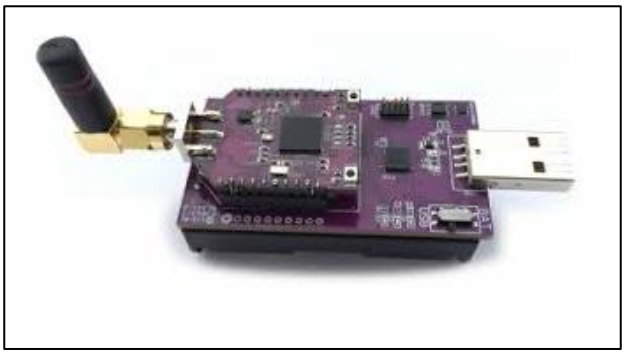

Figura 3: Placa Openmote-cc2538

- La placa generalmente se encuentra anexada a otros desarrollos que le permiten mayor versatilidad. En la siguiente foto se observa de izquierda a derecha el

openmote-cc2538, OpenBattery, OpenEthernet y OpenUSB. En estas placas se monta la primera de ellas.

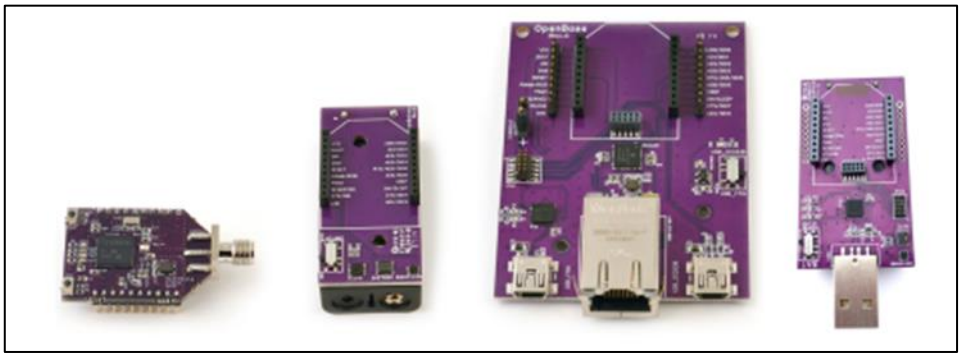

Figura 4: Versiones de placas

- La placa principal incluye 4 LEDs, un botón de usuario para propósitos de debug y un reset de hardware. Adicionalmente suelen contar con tres sensores identificados como sht21 (sensor de humedad y temperatura), max44009 (sensor de luminosidad) y adx1346 (giróscopo).
- Contiene un chip FTDI FT2232H que permite la conversión Serie-USB. Además, este permite la programación del CC2538 de forma directa usando el bootloader interno y el script cc2538-bsl de Python.
- Incluye dos conectores de antena SMA para el rango de los sub-GHz y una antena de 2.4 GHz. El conector de la antena de sub-GHz está directamente conectada al chip de radio AT86RF215. La antena de 2.4 GHz realiza una multiplexación utilizando un switch de RF entre el CC2558 y el AT86RF215.
- Por último, es posible utilizar las placas OpenMote con otros proyectos open-source como FreeRTOS, RIOT, Contiki y recientemente Contiki-ng (utilizado en el presente proyecto).

5.1.1.3 Placa RaspberryPI

En este desarrollo se utilizó específicamente el modelo 3B+, que cuenta con las

siguientes características:

- Incorpora un procesador Broadcom BCM2837B0 Cortex-A53 de 64 bits con cuatro núcleos a 1.4 GHz, una memoria RAM de 1 GB LPDDR2.
- Entre las opciones de conectividad se cuenta con: Wi-Fi 802.11ac de doble banda (2.4 GHz y 5 GHz), Bluetooth 4.2/BLE, Ethernet Gigabit y cuatro puertos USB 2.0.
- Para el almacenamiento tanto de información de manera permanente como del sistema operativo utiliza una tarjeta microSD. En el caso del presente desarrollo se utilizó una memoria de 16 GB.
- Es compatible con varios sistemas operativos, incluyendo Raspbian (basado en Debian), Ubuntu Mate, Windows 10 IoT Cores, y otras distribuciones Linux.

Figura 5: RaspberryPI 3B+

5.1.2 Infraestructura sobre servidor físico

5.1.2.1 Virtualizador de servidores – Proxmox

Se optó por utilizar Proxmox como hipervisor de nivel uno debido a su naturaleza de código abierto (open source) y su base en Linux Debian. La mayor parte de la infraestructura se construyó sobre esta plataforma, incluyendo configuraciones relacionadas con redes y networking.

Figura 6: Logo Proxmox

En Proxmox, se emplearon contenedores LxC (Linux Containers), que se definen como una versión optimizada del sistema operativo base del cual derivan. La razón detrás de esta elección radica en la minimización del consumo de recursos, evitando el uso de máquinas virtuales cuando no sea necesario. Como sistema operativo base para los contenedores se utilizaron Ubuntu 20.04 y CentOS 7. Sobre estos contenedores se realizaron las configuraciones necesarias para habilitar los servicios de reserva, interacción y programación de las placas openmote.

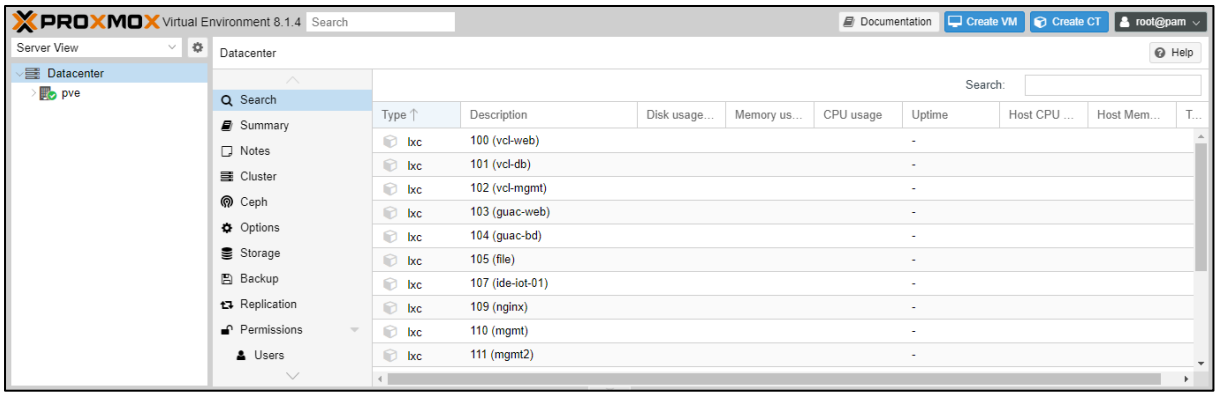

Figura 7: Interfaz web Proxmox

5.1.2.2 Gestionador de reservas – Apache VCL

VCL, abreviatura de Virtual Computing Lab, desempeña la función de gestionar la reserva de los contenedores que se conectarán a las placas Openmote-cc2538. Se trata de un sistema de gestión de laboratorios virtuales de código abierto que facilita el acceso remoto a recursos informáticos compartidos a través de internet.

Figura 8: Logo VCL

La instalación de VCL involucró tres nodos o servidores individuales (identificados como vcl-web, vcl-db y vcl-mgmt en la imagen 43):

- Nodo de gestión (vcl-mgmt): encargado de la gestión centralizada de los usuarios, grupos, horarios y política de reserva, configuración de máquinas virtuales, entre otras funciones.
- Nodo de base de datos (vcl-db): se almacena y gestiona toda la información relacionada con Apache VCL.
- Nodo web (vcl-web): es el componente que proporciona la interfaz de usuario web para que los usuarios soliciten y gestionen sus máquinas virtuales (VMs).

Para la instalación, Apache VCL proporciona un script diseñado para ejecutarse nativamente en sistemas CentOS 6.x y 7.x. Sin embargo, dado que Proxmox no cuenta con estas versiones de CentOS disponibles, fue necesario exportar la versión 7.x previamente descargada del contenedor.

Es importante destacar que los tres componentes de VCL deben ser instalados en el servidor físico para su funcionamiento adecuado.

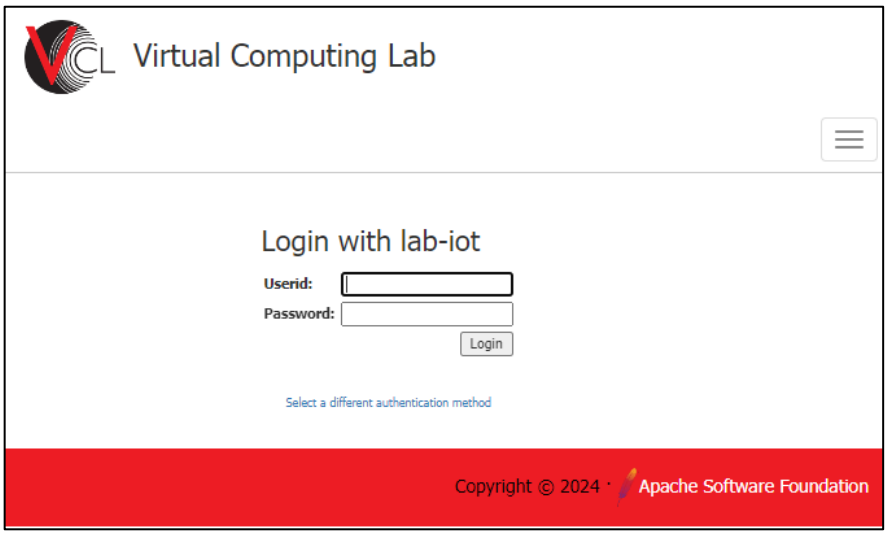

Figura 9: Interfaz web VCL

5.1.2.2.1 Anexo de entornos de desarrollo a VCL

Para poder contar con los entornos de desarrollo (identificados como ide-iot-01 en la imagen 43) que en una primera instancia se reservan para posteriormente conectarse de forma remota mediante apache Guacamole es necesario unir los primeros a Apache VCL mediante un intercambio de llave pública y privada. Posterior a esto se debe copiar desde el nodo de management una serie de scripts a cada uno de los clientes, entre estos se debe destacar el ejecutable correspondiente al demonio vcld.

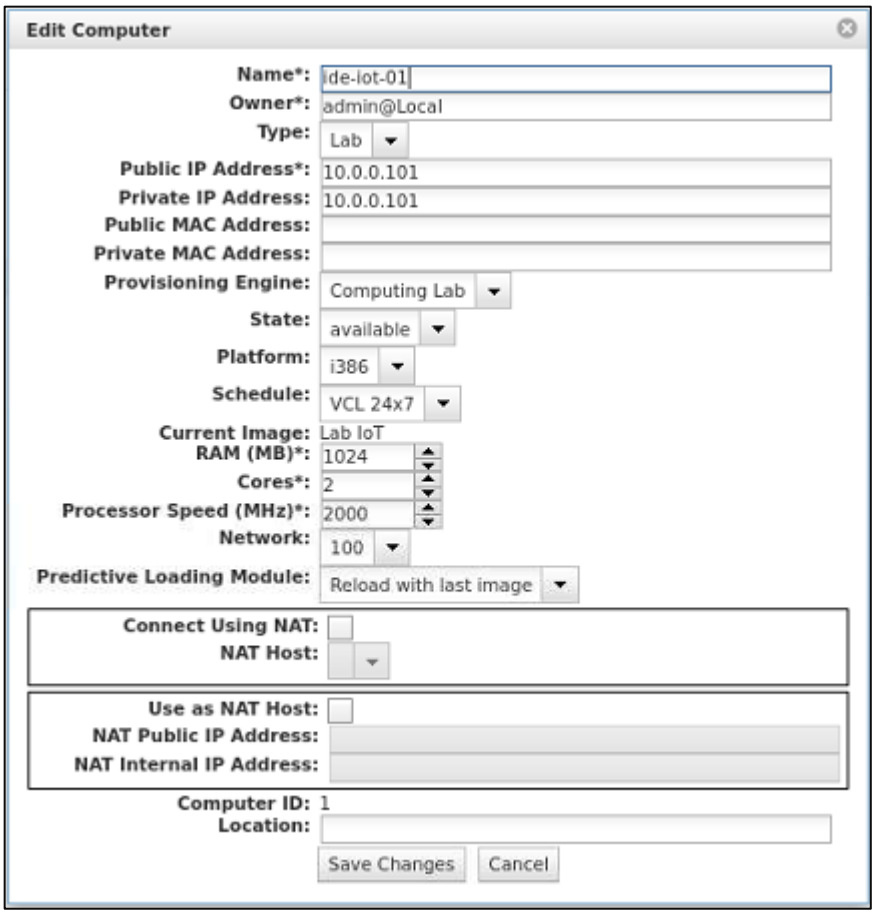

Figura 10: Anexo de entornos de desarrollo

En la configuración es necesario modificar el puerto ssh por defecto (22), ya que este es utilizado por el nodo vcl-mgmt para realizar la reserva de los entornos de desarrollo y posteriormente permitir la conexión a través de Apache Guacamole mediante puerto 24.

5.1.2.3 Emulador de terminal – Apache Guacamole

Guacamole es una aplicación de escritorio remoto de código abierto, que permite acceder a máquinas y escritorios de forma remota a través de un navegador web. Su estructura se compone de diversos elementos, como guacd (Guacamole Daemon), una interfaz web y una base de datos.

Figura 11: Logo Guacamole

El demonio guacd desempeña un papel fundamental al traducir las acciones realizadas por el usuario en comandos comprensibles para la máquina remota. Esto facilita la interacción remota utilizando una variedad de protocolos, como SSH, VNC, RDP y otros protocolos de escritorio remoto. En esencia, guacd actúa como intermediario, permitiendo que los usuarios puedan interactuar de manera efectiva con las máquinas y escritorios remotos a través de Guacamole.

Figura 12: Logueo Apache Guacamole

Apache Guacamole se instaló en dos contenedores Ubuntu (identificados como guac-db y guac-web en la imagen 43).

Es necesario realizar la conexión entre la interfaz web de guacamole (ya que esta posee el demonio guacd) y los entornos de desarrollo de forma manual.

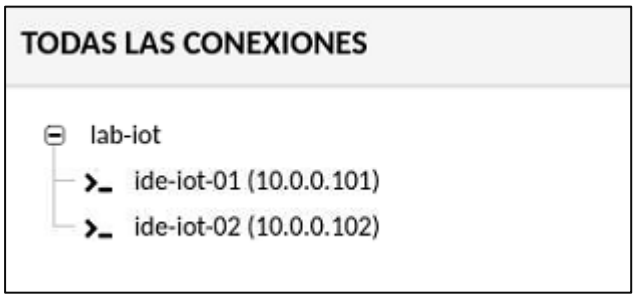

Figura 13: Conexiones Apache Guacamole

5.1.2.4 Controlador de dominio - zentyal

Zentyal se utilizó como controlador de dominio de todo el sistema, permitiendo la autenticación a través de LDAP contra el mismo servidor zentyal, así como la gestión de recursos y grupos. Dentro de este se realiza la discriminación entre usuarios de dominio y de esquema. Dichos usuarios forman parte de grupos que tienen los permisos necesarios para realizar la reserva y ejecución de los entornos de desarrollo.

Figura 14: Logo Zentyal

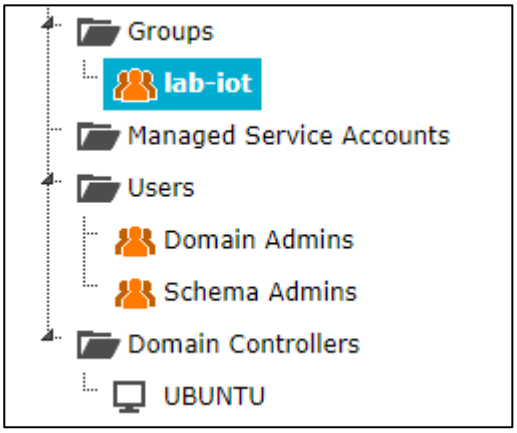

Figura 15: Grupos y Usuarios

Para acceder a las plataformas VCL y Guacamole, todos los usuarios deben disponer de credenciales de inicio de sesión creadas en Zentyal. La falta de estas credenciales impedirá el acceso a dichas plataformas. Además, es necesario que estos usuarios sean

parte del grupo lab-iot, ya que únicamente los miembros de este grupo tienen la autorización para reservar y utilizar los entornos de desarrollo.

A continuación, se presenta el flujo de información desarrollado en relación con la autenticación de los usuarios y su conexión a los entornos de desarrollo.

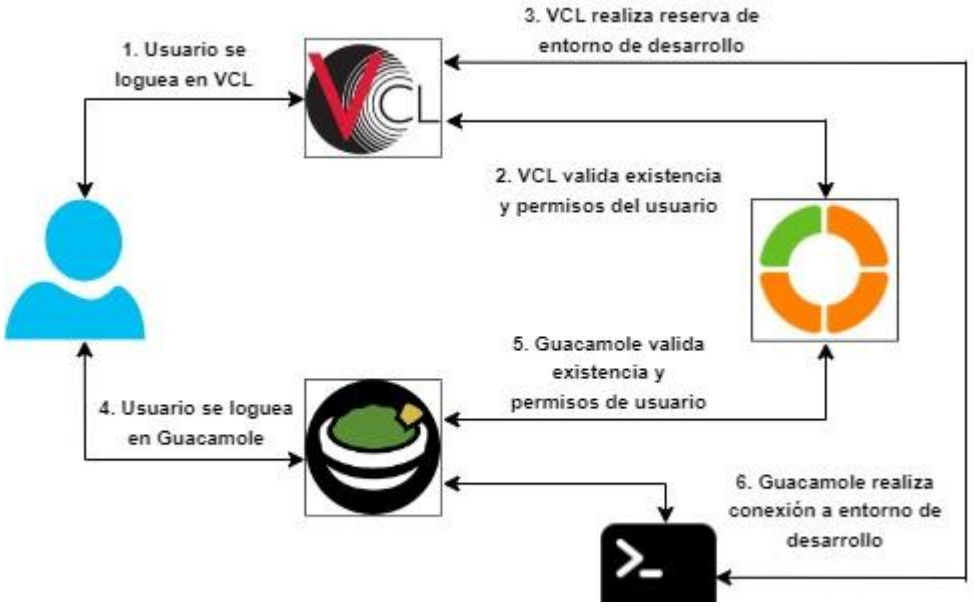

Figura 16: Flujo de comunicación reserva e interacción

5.1.2.4.1 Anexo de entornos de desarrollo al dominio - Zentyal

Para implementar la gestión centralizada de usuarios mediante Zentyal, es esencial integrar los entornos de desarrollo en dicho sistema. Esto permite que la autenticación de usuarios en el contenedor responsable de programar y cargar el firmware se realice a través del controlador de dominio. Para lograr esta integración, se debe unir una computadora al dominio de Zentyal y establecer una relación de confianza con este, lo que habilita la autenticación de usuarios con sus credenciales de inicio de sesión en el entorno de desarrollo mediante el protocolo LDAP.

En el caso de este proyecto, se empleó el demonio sssd (System Security Services Daemon). Este actúa como intermediario entre el sistema operativo y los servicios de autenticación remotos, permitiendo que los usuarios inicien sesión de manera segura y

accedan a recursos en un entorno distribuido. Después de instalar sssd, es necesario configurar el archivo sssd.conf, que almacena las configuraciones de unión al dominio y garantiza la correcta autenticación de los usuarios.

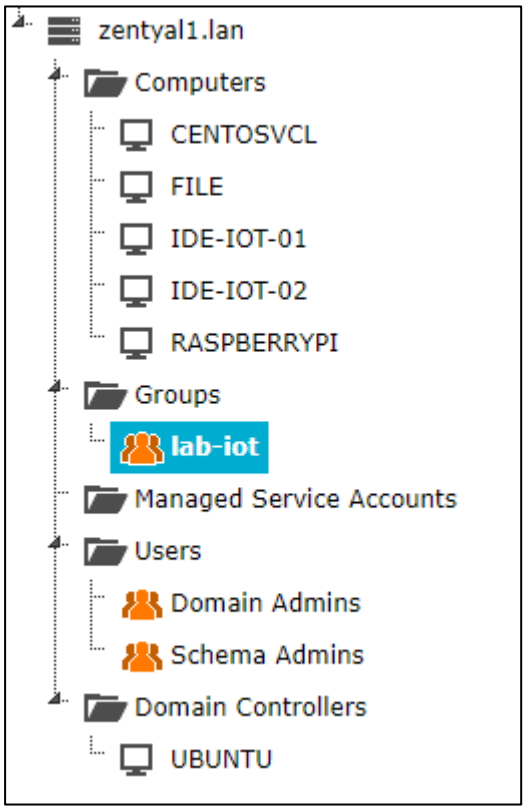

Figura 17: Gestión de grupos y Usuarios en Zentyal

5.1.2.4.2 Autenticación de usuarios del dominio en apache VCL

Para implementar la autenticación centralizada y permitir que los usuarios creados en Zentyal inicien sesión en VCL con las credenciales otorgadas, se llevaron a cabo una serie de configuraciones en el nodo web de Apache VCL, el cual fue creado en el punto 5.1.2.2 del proyecto.

Se debieron realizar una serie de pasos, entre los que se incluye tener habilitado SSL en el servidor LDAP (Zentyal), intercambio de certificados entre Zentyal y el nodo web de Apache VCL, configuraciones en scripts y base de datos dentro del nodo web de VCL. Adicionalmente, se permitió que los usuarios autenticados pudieran realizar la reserva de los recursos computacionales. Esto implicó modificaciones en la interfaz web de

Apache VCL, donde se identificaron ajustes en el grupo de usuarios (User Groups) y en el árbol de privilegios para garantizar el acceso y uso adecuado de los recursos por parte de los usuarios autorizados.

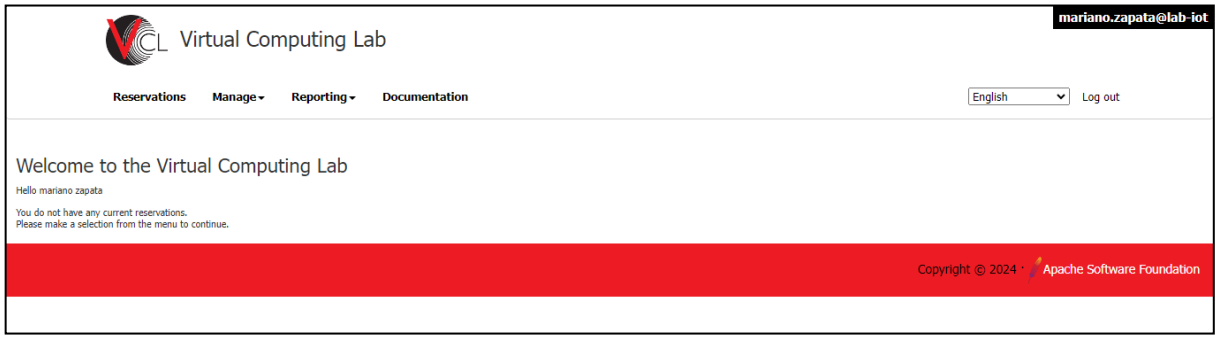

Figura 18: Logueo en VCL

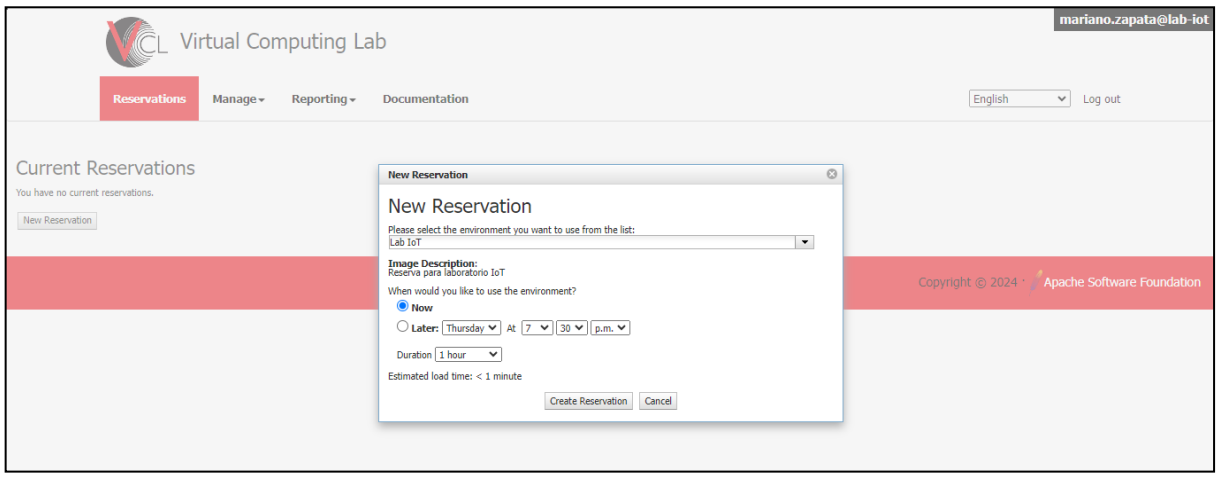

Figura 19: Ejecución de reserva

5.1.2.4.3 Autenticación de usuarios del dominio en apache Guacamole

Para permitir que los usuarios de Apache VCL se autentiquen también en Zentyal mediante LDAP, se deben realizar varias configuraciones en el nodo web creado en el punto 5.1.2.3. Entre estas se debe copiar los archivos de autenticación auth-ldap y auth-jdbc y el conector mysql al directorio de guacamole que se encuentra en /etc/guacamole. Además de reescribir el archivo guacamole.properties.

Posterior a realizar las configuraciones mencionadas se debe crear un grupo de usuario cuyo nombre sea igual al grupo creado en Zentyal. Por último, permitir el acceso a los

recursos agregados en la sección Apache Guacamole.

Figura 20: Ejecución de reserva en Guacamole

5.1.2.5 Servidor de archivos – File Server

El objetivo del file server en esta arquitectura es almacenar los directorios de inicio de los usuarios, permitiéndoles conservar sus configuraciones y programaciones independientemente del entorno de desarrollo al que se conecten. Aunque inicialmente se estableció la creación de un único entorno de desarrollo en este proyecto, se contempla la posibilidad de aumentar la cantidad de estos en el futuro, razón por la cual se ha implementado un servidor de archivos.

Para este propósito, se utilizó un contenedor LxC CentOS7 como file server, en el cual se crearon todos los usuarios junto con sus directorios de inicio. Para el montaje de estos directorios, se instaló el paquete NFS (Network File System) y se realizó la configuración correspondiente en el archivo /etc/exports.

Además, para lograr el montaje y desmontaje automático de los directorios de inicio en los entornos de programación, se configuró autofs. Esto implicó la configuración de los archivos auto.master y auto.nfs para gestionar el montaje automático de los `/homes` de los usuarios según sea necesario, proporcionando así un acceso transparente y eficiente a los archivos almacenados en el file server.

```
[root@file mariano.zapata]# ls
contiki-ng  solicitar_peticion  test.sh
[root@file mariano.zapata]#
```

Figura 21: File server

5.1.2.6 Proxy Inverso - Nginx

Figura 22: Logo NGINX

Nginx es un servidor web de código abierto ampliamente reconocido por su versatilidad, ya que puede funcionar como un proxy inverso, balanceador de carga y servidor de caché. Se destaca por su alto rendimiento, bajo consumo de recursos y capacidad para escalar eficientemente, lo que lo convierte en una opción ideal para manejar un gran volumen de conexiones simultáneas.

En este contexto, Nginx se instaló en contenedores Linux debido a su capacidad para ejecutarse de manera eficiente en este entorno. Posteriormente se modificó el archivo de configuración del proxy que se encuentra ubicado en el directorio `/etc/nginx/sites-available/default` con el objetivo de facilitar la conexión a los servidores Apache VCL y Guacamole a través de una única dirección IP. De esta manera, Nginx actúa como un punto de acceso centralizado que redirige las solicitudes entrantes hacia los servidores correspondientes, permitiendo una gestión más eficaz y optimizada de las conexiones web.

5.1.2.7 OPNsense

Figura 23: Logo OPNsense

OPNsense es un sistema operativo basado en FreeBSD que se enfoca en la seguridad y gestión de redes. Ofrece un conjunto completo de herramientas y funciones diseñadas para proteger, administrar y monitorear redes informáticas, incluyendo funciones como firewalls, VPN, balanceo de carga y filtrado de contenido, entre otras.

Dado que OPNsense es un sistema operativo completo, debe ser instalado en una máquina virtual configurada sobre Proxmox.

Una vez instalado, se deben configurar las reglas en el firewall para permitir el acceso desde la WAN (Wide Area Network) hacia la red LAN (Local Area Network) interna. El proceso utilizado en la conexión fue port forwarding, el cual implica redireccionar las solicitudes de conexión mediante el uso de puertos específicos en la interfaz de entrada (WAN) a direcciones IP internas (LAN) correspondientes a servidores específicos, garantizando la conectividad segura y eficiente entre la red externa e interna. Las reglas deben ser configuradas desde la interfaz LAN del firewall ya que es la única que permite conexión, esto se realiza teniendo en cuenta medidas de seguridad para restringir el acceso al mismo desde la WAN.

5.1.2.8 Configuración de servidor de management

El objetivo del servidor de management es proporcionar una herramienta que permita la conexión a configuraciones e instancias que, por defecto, no son accesibles desde la red exterior. Esto se logra permitiendo el acceso a servicios que contienen una interfaz web para su configuración, como es el caso de Zentyal.

Para habilitar esta funcionalidad, se instaló un servidor VNC en el servidor de management. Esto permite la interacción gráfica y la posterior conexión a diversas interfaces web. En los casos en los que los servidores no disponen de una interfaz web, se permite la conexión mediante SSH, siempre y cuando se utilice una clave de acceso previamente establecida. De esta manera, se garantiza un acceso seguro y controlado a las configuraciones e instancias relevantes desde la red exterior.

5.1.3 Entornos de desarrollo – Programación de firmware

Figura 24: Logo Contiki-NG

Los entornos de desarrollo, representados por el contenedor LxC ("ide-iot-01" en la imagen 43), son utilizados para la programación de las placas openmote y la interacción con dispositivos IoT.

El primer paso es la instalación de Contiki-NG, un sistema operativo especializado en programar dispositivos IoT. Esto implica realizar una serie de configuraciones adicionales relacionadas con la instalación de paquetes y actualización en el directorio `/home` del servidor. Es importante destacar que el compilador ARM debe ser instalado específicamente en los entornos de desarrollo y no en el file server, asegurando así la correcta configuración y funcionalidad de los entornos destinados a la programación de dispositivos IoT.

```
[root@file contiki-ng]# ls
arch                 LICENSE.md               Makefile.include  tests
CODE_OF_CONDUCT.md   Makefile.dir-variables   Makefile.tools    tools
CONTRIBUTING.md      Makefile.embedded        os
doc                  Makefile.help            README.md
examples             Makefile.identify-target SECURITY.md
```

Figura 25: Arquitectura Contiki-NG

5.1.3.1 Entorno de desarrollo – Acceso a placas Openmote

El contenedor LxC que corresponde exclusivamente a los entornos de desarrollo debe ser modificado porque con la configuración básica y por la naturaleza constructiva propia de los contenedores, estos no permiten interacción con los periféricos y los motes (placas openmote-cc2538), que están conectados mediante USB al hipervisor Proxmox.

Proxmox facilita la configuración base de los contenedores mediante el directorio `/etc/pve/lxc`.

Las configuraciones que se deben realizar son:

- Permitir que el contenedor en el cual está configurado el entorno de desarrollo pueda acceder al sistema de archivos NFS (Network File System), con el objetivo de permitir el montaje de los home de los usuarios.
- Instalar el demonio sssd.
- Permitir al contenedor acceder al dispositivo de caracteres de bajo nivel como /dev/null, dev/zero, etc.
- Realizar el montaje de los bus y los dispositivos seriales (ttyUSB*).

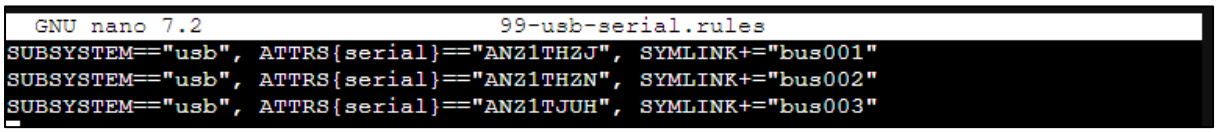

Figura 26: Creación de reglas

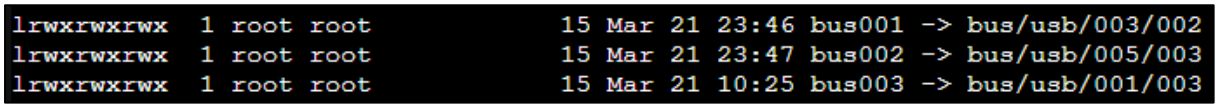

Figura 27: Validación de reglas

5.1.3.2 Entorno de desarrollo – compilador de firmware

El compilador ARM es esencial para cargar el firmware programado a través de Contiki-NG en las diversas placas conectadas a los puertos USB del sistema. Sin embargo, surge un problema con la configuración estándar: aunque los directorios home de los usuarios se encuentran en el file server, los usuarios no realizan la programación directamente en este servidor. Por lo tanto, para garantizar la escalabilidad y el acceso eficiente al compilador, se optó por instalarlo directamente en los entornos de desarrollo.

Para lograr esto, se utilizó la instrucción "apt-get install gcc-arm-none-eabi" que permite instalar el compilador ARM en los entornos de desarrollo.

5.1.3.3 Entorno de desarrollo – tunel slip

El directorio /dev/net es utilizado en sistemas Linux/Unix para la gestión de dispositivos

de red. En el contexto de Contiki-NG, este directorio es fundamental para el uso de herramientas de red, especialmente aquellas asociadas con el túnel SLIP.

Sin embargo, surgió un problema en este punto debido a que la ejecución del túnel SLIP requiere privilegios de superusuario (sudo). Esto planteaba un riesgo, ya que con permisos de superusuario se podrían realizar modificaciones en los entornos de programación de otros usuarios o incluso en el propio entorno de programación.

Para abordar esta problemática, se implementó una solución creando un archivo dentro del directorio sudoers.d del entorno de desarrollo. Este archivo permite la ejecución de comandos de superusuario únicamente en el directorio home correspondiente al usuario en cuestión, evitando así modificaciones no autorizadas en otros entornos de programación. De esta manera, se garantiza un nivel adecuado de seguridad y control en el uso de privilegios de superusuario para ejecutar el túnel SLIP y otras operaciones necesarias.

```
%domain_users ALL=(ALL) NOPASSWD:/usr/bin/make, /mnt/nfs/home/*/contiki-ng/tools/serial-io/tunslip6
```

Figura 28: Archivo sudoers.d

5.1.4 Configuración de placa Raspberry

El uso de la placa Raspberry Pi está asociado a emular de la manera más precisa posible una topología operativa que se encuentra en campo. Esta topología está compuesta por una placa openmote que desempeña el papel de router de borde, mientras que la Raspberry Pi funciona como el nodo coordinador.

En la Raspberry Pi se instaló el sistema operativo Raspbian utilizando la herramienta Raspberry Pi Imager. Para permitir que los usuarios monten su directorio home correspondiente en la Raspberry Pi, se requiere configurar el demonio sssd, siguiendo pasos similares a los realizados en etapas anteriores.

Es importante destacar que el acceso a esta placa no se realiza directamente mediante una reserva en Apache VCL. En cambio, se establece una conexión mediante un túnel SSH desde el entorno de desarrollo correspondiente. Este túnel SSH utiliza el nombre de usuario activo en el entorno de desarrollo y la dirección IPv4 de la placa Raspberry Pi.

En cuanto a la compilación de firmware en la Raspberry Pi, el proceso se llevó a cabo sin inconvenientes significativos. Únicamente se realizó la instalación de la librería necesaria utilizando el comando "apt-get install gcc-arm-none-eabi". Esto asegura que la Raspberry Pi esté lista para compilar el firmware requerido en el contexto de la topología emulada.

5.1.4.1 Placa Raspberry – acceso a puerto serie y túnel slip

A diferencia de los entornos de desarrollo, se observaron una serie de problemas al momento de ejecutar el comando login y utilizar el túnel slip, debido al montaje del home del usuario tanto en el entorno de desarrollo como en la placa raspberryPI.

El problema reside en la arquitectura de los procesadores que se están utilizando. Por lo tanto, el binario compilado utilizado en la ejecución de cada uno de los sistemas debe ser diferente para cada caso.

Frente a dicha problemática se proponen dos soluciones diferentes, una de ellas consiste en que el usuario debe eliminar el binario ya compilado para posteriormente recompilarlo. Esto debe realizarlo cada vez que se haga uso de la herramienta login y tunslip6 cuando se cambia desde el entorno de desarrollo a la raspberry.

La segunda opción es crear dos binarios compilados diferentes cuyos nombres identifiquen en cual sistema pueda ejecutarse (en la actualidad solamente puede realizarse con el binario serialdump).

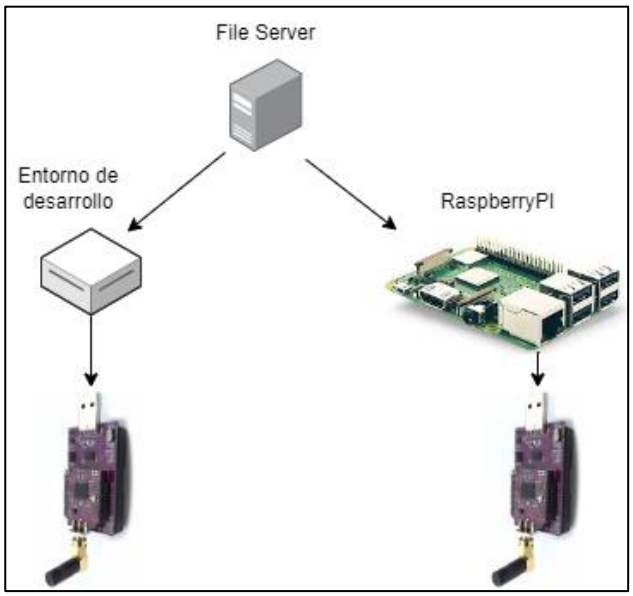

Figura 29: Montaje de /home

5.1.4.2 Conexión de RaspberryPi

Esta placa cuenta con una particularidad ya que la misma se encuentra conectada de forma externa mediante un adaptador usb-ethernet, sin embargo, como se mencionó anteriormente, es necesario que un usuario que realiza la reserva de un entorno de desarrollo pueda acceder a la Raspberry. Como la conexión es mediante SSH, el entorno de desarrollo y la placa deben compartir el segmento de red.

Para lograr esta interconexión, se realiza una configuración en Proxmox que permite que la interfaz física asociada a la red de los entornos de desarrollo tenga conectividad hacia el exterior a través de la interfaz física configurada. Esta solución habilita la conexión desde los entornos de desarrollo hacia la placa Raspberry Pi.

Adicionalmente, la placa Raspberry Pi emplea su conexión inalámbrica para almacenar los datos recopilados en una base de datos SQL alojada en un servidor dentro del laboratorio GRIDTICS. Es importante destacar que esta base de datos se encuentra externa al desarrollo actual, lo que muestra una integración compleja pero eficiente de distintos elementos dentro del sistema.

5.1.5 Servidor de cámara

Para lograr la visualización de las placas openmote-cc2538 y la Raspberry Pi, se utiliza una cámara IP que forma parte de una estructura construida específicamente para este proyecto.

La cámara se encuentra conectada a una tercera interfaz de red y configurada en un segmento de red diferente al de los entornos de programación. Esto se hace con el propósito de protegerla de posibles errores ocurridos en los entornos de desarrollo y evitar un exceso de tráfico en una única interfaz.

Para compartir la filmación en tiempo real, se emplea la aplicación ffmpeg. Se configuró un nuevo servidor dentro de Proxmox que cuenta con una doble placa de red. Esto permite la conexión a la cámara a través de su segmento de red correspondiente y, mediante la segunda placa de red virtual, llegar al canal de YouTube del proyecto a través del firewall.

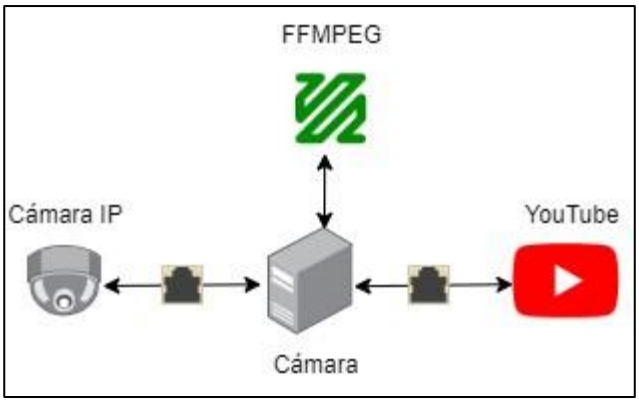

Figura 30: Arquitectura Cámara-YouTube

En las siguientes imágenes se observa el resultado de la instalación de las cámaras y las placas openmote cc-2538.

Figura 31: Instalación cámara – parte 1

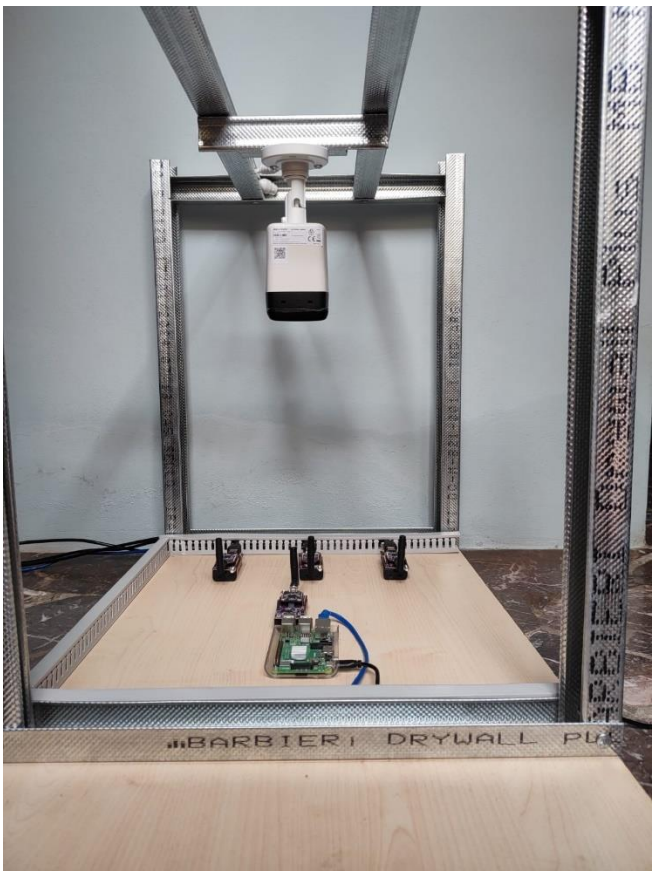

Figura 32: Instalación cámara - imagen 2

5.1.6 Conexión de clientes mediante SFTP

El objetivo poder utilizar este protocolo de intercambio de archivos se basa en que Contiki-NG cuenta con la posibilidad de realizar una captura de paquetes. Posterior a una serie de configuraciones realizadas sobre una placa openmote utilizada como capturador de paquetes, se realiza la ejecución de un programa Python, el cual genera un archivo .pcap que puede ser leído y analizado por varias herramientas como Wireshark, Network Minner, entre otras.

Para obtener dicho archivo se debe descargar desde el entorno de desarrollo del usuario, para ello se propone como solución que el servidor file-server funcione también como un servidor SFTP.

Para ello se debe realizar una serie de configuraciones sobre el archivo sshd_config que reside en el directorio /etc/ssh para permitir la comunicación.

Posterior a realizar dichas configuraciones, el usuario podrá conectarse mediante un cliente SFTP como WinSCP a su /home y obtener el archivo .pcap para abrirlo mediante wireshark y poder analizarlo.

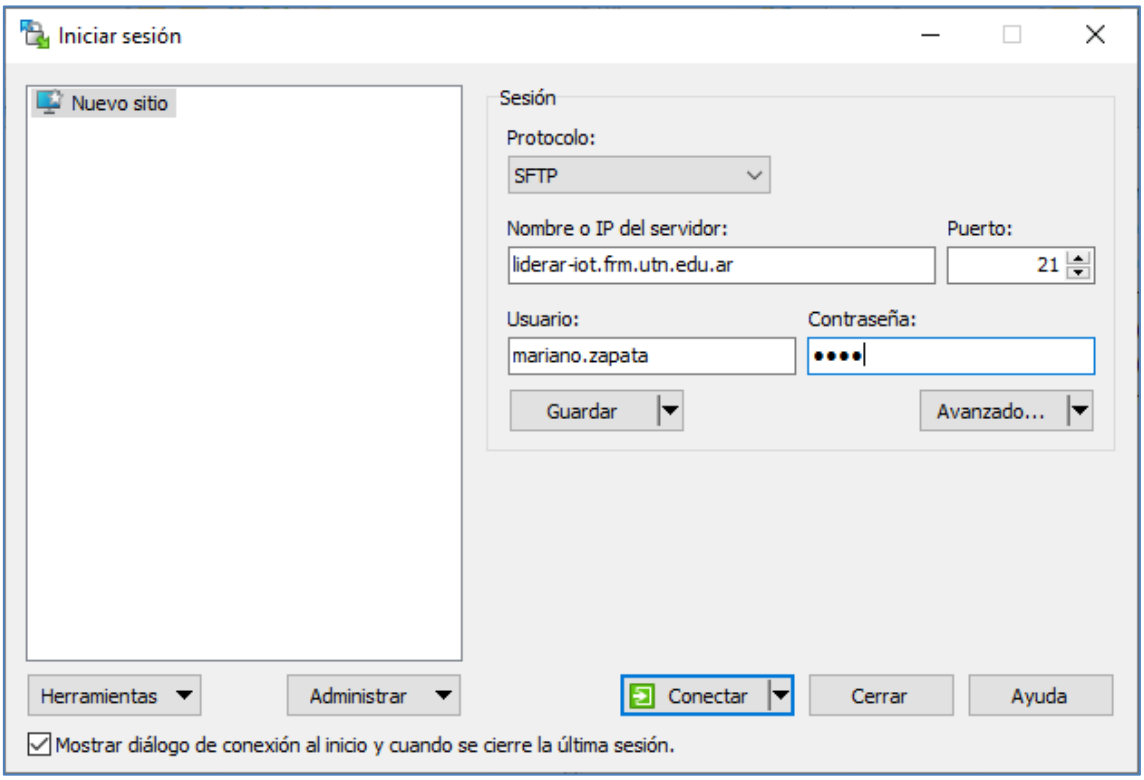

Figura 33: Cliente winscp

En el firewall OPNsense se realizó la configuración de una nueva regla NAT que permita la conexión desde el exterior al home del file server a través del puerto 21.

5.1.7 Pasarela de Pago

La propuesta del laboratorio incluye la posibilidad de generar ingresos por su uso a través de consultorías o cursos. Se plantea implementar una opción de pago utilizando la plataforma de Mercado Pago para aquellos usuarios que deseen acceder al laboratorio de forma independiente, además de una opción de inscripción previa para aquellos que participen en actividades organizadas.

En la interfaz web del nodo web de Apache VCL (vcl-web) se ha integrado un código que detiene la ejecución del flujo de programa hasta que el usuario realice el pago a través de Mercado Pago. Una vez completado el pago, el flujo de ejecución continúa normalmente, permitiendo la reserva de los recursos computacionales.

El laboratorio se ha diseñado de manera modular, lo que facilita la desactivación de la interfaz de pago eliminando o comentando una línea de código en el programa principal del nodo vcl-web. Otra opción es aplicar un descuento del 100% en el momento del pago, lo que resultaría en el uso gratuito del laboratorio.

El precio del uso del laboratorio varía según el tiempo de utilización, que puede configurarse desde media hora hasta un máximo de 8 horas. Además, se ofrece la opción de conectarse mediante SFTP al directorio /home del usuario en el servidor file server. Esto permite al usuario realizar programaciones de forma local y utilizar el laboratorio únicamente para pruebas en las placas openmote-cc2538.

El flujo de ejecución completo para la reserva del entorno de desarrollo es el siguiente:

Figura 34: Flujo de comunicación con pasarela de pago

5.1.8 Funcionamiento de las placas OpenMote

Teniendo en cuenta que uno de los objetivos por los que se realizó el laboratorio es la enseñanza y divulgación de los dispositivos IoT, se tornó necesario aprender sobre el funcionamiento de estas en concreto, su funcionamiento y protocolos involucrados.
Gran parte de este conocimiento fue volcado en el documento del usuario identificado como “manual-usuario-lab-iot” y en el de arquitectura “manual-arquitectura-lab-iot”. Sin embargo, se expondrá en los siguientes párrafos de forma resumida las principales características de funcionamiento encontradas, así como los programas desarrollados como ejemplos para los usuarios.

5.1.8.1 Protocolos y estándares utilizados

En el contexto de las redes de sensores inalámbricos (WSN), las placas openmote-cc2538 desempeñan un papel crucial al facilitar la comunicación inalámbrica entre nodos. Estas redes, conocidas como WSN por sus siglas en inglés (Wireless Sensor Network), se fundamentan en la interacción entre estos nodos, constituyendo la base de operaciones para la transmisión de datos en entornos específicos.
Un elemento distintivo de las WSN es su utilización de IPv6 como protocolo de comunicación, aprovechando las ventajas que este ofrece para redes extensas con numerosos dispositivos interconectados. Sin embargo, surge un desafío significativo debido al estándar IEEE802.15.4, comúnmente empleado en este tipo de redes, que impone restricciones en el tamaño máximo de los paquetes de datos, limitándolos a 127 bytes.
Para superar esta limitación, se desarrolló el estándar 6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks), que opera a nivel del stack TCP/IP entre la capa de red y la capa física. Su función principal radica en la compresión de los encabezados de IPv6, reduciendo así la cantidad de bytes enviados durante la transmisión de datos. Esto se logra mediante métodos de compresión que pueden reducir un encabezado de 48 bytes a tan solo 7 bytes, sin sacrificar información esencial.

Un aspecto destacado de 6LoWPAN es su capacidad para realizar esta compresión de manera eficiente y sin la necesidad de paquetes adicionales para comprimir o descomprimir la información. Todos los datos necesarios para la descompresión se incluyen en el propio paquete transmitido, lo que simplifica el proceso y optimiza el uso de recursos en entornos de WSN donde la eficiencia energética es crítica.

Figura 35: Stack TCP/IP

Gracias a este estándar es que las placas openmote pueden utilizar IPv6 y considerando que pueden haber cientos o miles de estos dispositivos, se eliminan protocolos propios de IPv4 como NAT.

5.1.8.2 CoAP vs MQTT

Los dos protocolos que pueden ser utilizados en la capa de aplicación en el stack de las placas IoT son CoAP y MQTT, sin embargo, presentan notables diferencias entre sí. CoAP, por un lado, se caracteriza por su topología de red similar a una malla, en la que múltiples nodos servidores y clientes pueden interactuar de forma conjunta o independiente. Este protocolo utiliza UDP, lo que lo hace ligero y apropiado para aplicaciones en tiempo real con baja latencia. Una característica destacada de CoAP es su soporte nativo para la observación de recursos, permitiendo a los clientes suscribirse a cambios en los recursos y recibir notificaciones cuando estos cambian.

Esto se ilustra en la sección "Ejemplo de realización de comunicación CoAP" del manual del usuario.

Por otro lado, MQTT se basa en un modelo de publicación/suscripción, donde un servidor MQTT actúa como intermediario entre los dispositivos publicadores y los suscriptores. Este protocolo utiliza TCP, lo que garantiza una entrega fiable de mensajes, pero puede tener una mayor sobrecarga debido a la conexión persistente. MQTT es ampliamente utilizado en aplicaciones donde se requiere una comunicación asincrónica y escalabilidad en la gestión de dispositivos conectados.

En resumen, CoAP es adecuado para aplicaciones en tiempo real y con baja latencia, especialmente en entornos de IoT donde la eficiencia y la velocidad son críticas. Por otro lado, MQTT es ideal para aplicaciones de IoT que requieren una comunicación asincrónica y una gestión escalable de dispositivos. La elección entre ambos protocolos dependerá de los requisitos específicos de cada aplicación y del entorno en el que se implemente.

En las siguientes imágenes se expone someramente la programación y funcionamiento de CoAP en openmote - cc2538.

```
  PROCESS_PAUSE();

  LOG_INFO("Starting Erbium Example Server\n");

  /*
   * Bind the resources to their Uri-Path.
   * WARNING: Activating twice only means alternate path, not two instances!
   * All static variables are the same for each URI path.
   */
  coap_activate_resource(&res_hello, "test/hello");
  coap_activate_resource(&res_temp_sensor, "sensor/temperatura");
  /* Define application-specific events here. */
  while(1) {
    PROCESS_WAIT_EVENT();
#if PLATFORM_HAS_BUTTON
#if PLATFORM_SUPPORTS_BUTTON_HAL
    if(ev == button_hal_release_event) {
#else
    if(ev == sensors_event && data == &button_sensor) {
#endif
      LOG_DBG("*******BUTTON*******\n");

      /* Call the event handler for this application-specific event. */
      /* Also call the separate response example handler. */
    }
#endif /* PLATFORM_HAS_BUTTON */
  }                             /* while (1) */

  PROCESS_END();
```

Figura 36: Programación de firmware

```
import logging
import asyncio
import influxdb client
from aiocoap import *
from influxdb import InfluxDBClient

logging.basicConfig(level=logging.INFO)

async def main():
    protocol = await Context.create client context()

    request = Message(code=GET, uri='coap://[fd00::212:4b00:430:5314]/sensor/temperatura')

    try:
        response = await protocol.request(request).response
    except Exception as e:
        print('Failed to fetch resource:')
        print(e)
    else:
        print('Result: %s\n%r'%(response.code, response.payload))
        temp = (response.payload.decode('utf-8'))
        print(temp)
        write to influxdb(temp)

if   name   == "  main  ":
    asyncio.run(main())
```

Figura 37: Petición de recursos mediante Python

```
mariano.zapata@raspberrypi:~$ python3 conexion.py
Result: 2.05 Content
b'30.82'
30.82
mariano.zapata@raspberrypi:~$ python3 conexion.py
Result: 2.05 Content
b'30.81'
30.81
mariano.zapata@raspberrypi:~$
```

Figura 38: Ejecución de programa para obtención de recurso mediante petición CoAP

El protocolo MQTT también es una opción en la capa de aplicación diseñada para dispositivos en redes IoT y aplicaciones M2M. A diferencia de CoAP, MQTT se basa en un modelo de publicación/suscripción en el cual los dispositivos pueden publicar mensajes en temas (tópicos) y suscribirse a otros temas para recibir mensajes pertinentes. Una desventaja a considerar es que el servidor MQTT, conocido como bróker, debe estar siempre en funcionamiento (en el caso de utilizar una topología con un solo bróker), ya que si la comunicación con este se interrumpe, los dispositivos no tendrán otra ruta de comunicación disponible.

Entre las ventajas de MQTT se destacan la retención de mensajes, que permite que los mensajes publicados se conserven para los futuros suscriptores, y su escalabilidad y flexibilidad. Esto lo hace adecuado para entornos donde se necesitan múltiples dispositivos comunicándose de forma eficiente y confiable.

En la sección "Ejemplo de realización de comunicación MQTT" del manual de usuario, se proporciona un ejemplo detallado de programación y configuración utilizando MQTT, lo que permite a los usuarios comprender mejor su funcionamiento y su aplicación práctica en entornos de IoT. En las siguientes imágenes se expone brevemente el funcionamiento de MQTT en Contiki-NG:

```
/*---------------------------------------------------------------------------*/
/* MQTT broker address. Ignored in Watson mode */
#ifdef MQTT_CLIENT_CONF_BROKER_IP_ADDR
#define MQTT_CLIENT_BROKER_IP_ADDR MQTT_CLIENT_CONF_BROKER_IP_ADDR
#else
#define MQTT_CLIENT_BROKER_IP_ADDR "fd00::212:4b00:430:5381"
#endif
/*---------------------------------------------------------------------------*/
/*
```

Figura 39: Configuración de IPv6 de Broker Mosquitto

```
1711413442: Sending PUBLISH to mosq-5qY07GPlfSFnZzGgGG (d0, q0, r0, m0, 'iot-2/ev
t/status/fmt/json', ... (182 bytes))
1711413470: Received PINGREQ from mosq-5qY07GPlfSFnZzGgGG
1711413470: Sending PINGRESP to mosq-5qY07GPlfSFnZzGgGG
1711413472: Received PUBLISH from d:quickstart:mqtt-client:00124b305381 (d0, q0,
r0, m0, 'iot-2/evt/status/fmt/json', ... (182 bytes))
1711413472: Sending PUBLISH to mosq-5qY07GPlfSFnZzGgGG (d0, q0, r0, m0, 'iot-2/ev
t/status/fmt/json', ... (182 bytes))
1711413502: Received PUBLISH from d:quickstart:mqtt-client:00124b305381 (d0, q0,
r0, m0, 'iot-2/evt/status/fmt/json', ... (182 bytes))
1711413502: Sending PUBLISH to mosq-5qY07GPlfSFnZzGgGG (d0, q0, r0, m0, 'iot-2/ev
t/status/fmt/json', ... (182 bytes))
1711413530: Received PINGREQ from mosq-5qY07GPlfSFnZzGgGG
1711413530: Sending PINGRESP to mosq-5qY07GPlfSFnZzGgGG
1711413532: Received PUBLISH from d:quickstart:mqtt-client:00124b305381 (d0, q0,
r0, m0, 'iot-2/evt/status/fmt/json', ... (182 bytes))
1711413532: Sending PUBLISH to mosq-5qY07GPlfSFnZzGgGG (d0, q0, r0, m0, 'iot-2/ev
t/status/fmt/json', ... (182 bytes))
1711413562: Received PUBLISH from d:quickstart:mqtt-client:00124b305381 (d0, q0,
r0, m0, 'iot-2/evt/status/fmt/json', ... (182 bytes))
1711413562: Sending PUBLISH to mosq-5qY07GPlfSFnZzGgGG (d0, q0, r0, m0, 'iot-2/ev
t/status/fmt/json', ... (182 bytes))
1711413590: Received PINGREQ from mosq-5qY07GPlfSFnZzGgGG
1711413590: Sending PINGRESP to mosq-5qY07GPlfSFnZzGgGG
1711413592: Received PUBLISH from d:quickstart:mqtt-client:00124b305381 (d0, q0,
r0, m0, 'iot-2/evt/status/fmt/json', ... (182 bytes))
1711413592: Sending PUBLISH to mosq-5qY07GPlfSFnZzGgGG (d0, q0, r0, m0, 'iot-2/ev
t/status/fmt/json', ... (182 bytes))
```

Figura 40: Suscripción a tópico

5.1.8.3 Router de borde y nodo coordinador

Existe una diferencia fundamental entre estos dos conceptos, el router de borde consiste en un nodo que actúa como interfaz entre la red de nodos IPv6 y el exterior. A través de este se realiza la comunicación con los nodos sensores, los administra y registra los dispositivos, entre otros.

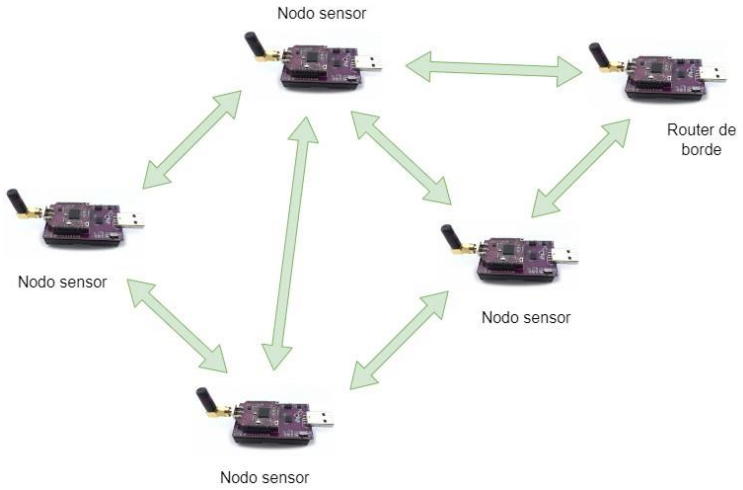

Figura 41: Topología malla

A diferencia del router de borde, el nodo coordinador actúa como interfaz entre el equipo inalámbrico y la red de internet, es decir que funciona como Gateway. Además se encarga de realizar las solicitudes de datos a los nodos sensores para luego ser almacenados en una base de datos, y posteriormente ser visualizados.

El router de borde forma parte del bloque coordinador, y generalmente en conjunto con un sistema microcontrolador forman lo que se identifica como coordinador.

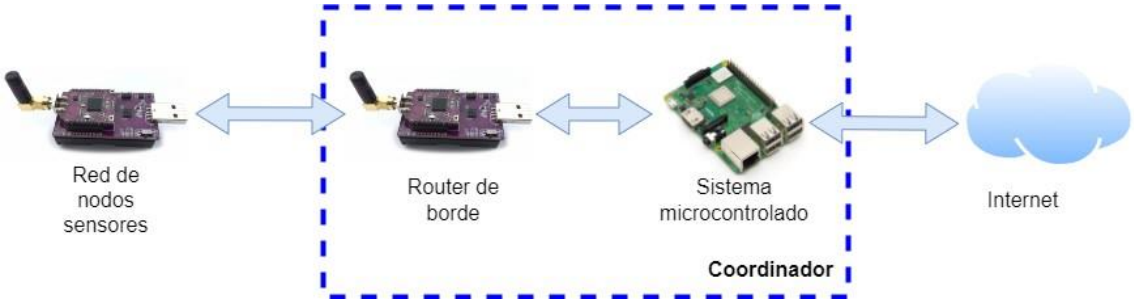

Figura 42: Router de borde y nodo coordinador

Nota1: Para información más detallada respecto al formato de programación consultar el documento: manual-usuario-lab-iiot.

Nota 2: Para información más detallada respecto a los protocolos involucrados en la comunicación de los dispositivos IoT consultar el documento: manual-arquitectura-lab-iiot.

5.1.9 Arquitectura final

En la siguiente imagen se expone y sintetiza toda la arquitectura desarrollada en las secciones anteriores:

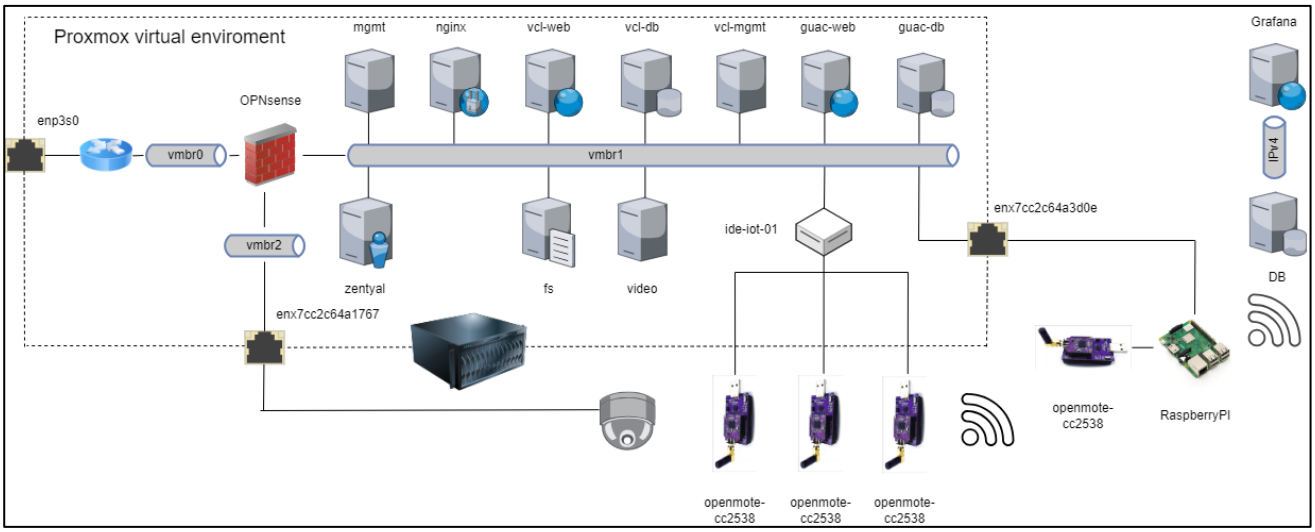

Figura 43: Arquitectura final completa

A través de la interfaz enp3s0 se realiza la comunicación con internet y redes externas al proyecto desarrollado. En la red identificada como vmbr1 se configuró el segmento 10.0.0.0/24 y todos los nodos conectados a esta red cuentan con una dirección IPv4 de este segmento, como la placa RaspberryPI forma parte de esta red, se conecta a la misma a través de la interfaz física enx7cc2c64a3d0e, la única manera de conectarse es a través de ssh y desde el entorno de desarrollo (ide-iot-01).

La red identificada como vmbr2 tiene configurado el segmento de red 192.168.2.0/24, por lo que no se puede conectar a esta desde la vmbr1, a excepción del servidor de video que cuenta con doble placa de red. La cámara es el único equipo conectado a esta red a través de la interfaz enx7cc2c64a1767. A través del servidor de video se logra "salir" a internet a través del firewall para compartir con un delay de aproximadamente 15 segundos lo que está sucediendo en tiempo real en el laboratorio de dispositivos IoT.

Por último, se puede realizar la conexión a través de VNC o SSH al servidor de management (mgmt en la figura 43), este funciona como un servidor de salto,

permitiendo la configuración del resto de los equipos de forma remota en caso de ser necesario.

5.2 FACTIBILIDAD ECONÓMICA

Mediante el análisis de factibilidad económica se busca determinar los beneficios que se pueden obtener en relación con los costos de inversión necesarios. En la siguiente tabla se presenta un resumen estimado de los gastos del proyecto, proporcionando una visión general de los costos que se prevén durante su ejecución. A continuación, se presenta el flujo de caja para un estimativo real del desarrollo de la empresa de forma anual, manteniendo las ventas año tras año. En ella se detallan los gastos estimados, brindando información precisa y relevante para evaluar la factibilidad económica del proyecto.

Flujo de caja del proyecto						
Flujo de caja (proyección anual)	0	1	2	3	4	5
Unidades promedio vendidas		12	12	12	12	12
Ingresos						
Precio por Venta		$ 15.000.000,00	$ 15.000.000,00	$ 15.000.000,00	$ 15.000.000,00	$ 15.000.000,00
Costos Fijos						
Mano de Obra		-$ 1.440.000,00	-$ 1.440.000,00	-$ 1.440.000,00	-$ 1.440.000,00	-$ 1.440.000,00
Alquiler de espacio de trabajo		-$ 180.000,00	-$ 180.000,00	-$ 180.000,00	-$ 180.000,00	-$ 180.000,00
Impuestos servicios		-$ 420.000,00	-$ 420.000,00	-$ 420.000,00	-$ 420.000,00	-$ 420.000,00
Costos Variables						
Pc Servidor		-$ 4.200.000,00	-$ 4.200.000,00	-$ 4.200.000,00	-$ 4.200.000,00	-$ 4.200.000,00
Motes (cant. 4)		-$ 5.760.000,00	-$ 5.760.000,00	-$ 5.760.000,00	-$ 5.760.000,00	-$ 5.760.000,00
Dispositivos Iot Varios (Cant. 37)		-$ 300.000,00	-$ 300.000,00	-$ 300.000,00	-$ 300.000,00	-$ 300.000,00
Cámara Hickvision		-$ 300.000,00	-$ 300.000,00	-$ 300.000,00	-$ 300.000,00	-$ 300.000,00
Rasperry PI +3B		-$ 600.000,00	-$ 600.000,00	-$ 600.000,00	-$ 600.000,00	-$ 600.000,00
Soportes		-$ 300.000,00	-$ 300.000,00	-$ 300.000,00	-$ 300.000,00	-$ 300.000,00
Cables de interconexión		-$ 60.000,00	-$ 60.000,00	-$ 60.000,00	-$ 60.000,00	-$ 60.000,00
Ganancias antes de impuestos		$ 1.440.000,00	$ 1.440.000,00	$ 1.440.000,00	$ 1.440.000,00	$ 1.440.000,00
Depreciación Pc Servidor		$ 70.000,00	$ 70.000,00	$ 70.000,00	$ 70.000,00	
Depreciación Adaptador USB-Ethernet		$ 1.000,00	$ 1.000,00	$ 1.000,00	$ 1.000,00	
Depreciación cámara		$ 2.500,00	$ 2.500,00	$ 2.500,00	$ 2.500,00	
Depreciación Raspbery PI+3B		$ 10.000,00	$ 10.000,00	$ 10.000,00	$ 10.000,00	
Depreciación Motes openmote-cc2538		$ 96.000,00	$ 96.000,00	$ 96.000,00	$ 96.000,00	
Depreciación Taladro de mano		$ 12.000,00	$ 12.000,00	$ 12.000,00	$ 12.000,00	
Depreciación Amoladora de mano		$ 11.000,00	$ 11.000,00	$ 11.000,00	$ 11.000,00	
Impuesto a las Ganancias		$ 371.249,00	$ 371.249,00	$ 371.249,00	$ 371.249,00	
Total después de impuestos		$ 1.068.751,00	$ 1.068.751,00	$ 1.068.751,00	$ 1.068.751,00	
Depreciación Pc Servidor		$ 70.000,00	$ 70.000,00	$ 70.000,00	$ 70.000,00	
Depreciación Adaptador USB-Ethernet		$ 1.000,00	$ 1.000,00	$ 1.000,00	$ 1.000,00	
Depreciación cámara		$ 2.500,00	$ 2.500,00	$ 2.500,00	$ 2.500,00	
Depreciación Raspbery PI+3B		$ 10.000,00	$ 10.000,00	$ 10.000,00	$ 10.000,00	
Depreciación Motes openmote-cc2538		$ 96.000,00	$ 96.000,00	$ 96.000,00	$ 96.000,00	
Depreciación Taladro de mano		$ 12.000,00	$ 12.000,00	$ 12.000,00	$ 12.000,00	
Depreciación Amoladora de mano		$ 11.000,00	$ 11.000,00	$ 11.000,00	$ 11.000,00	
Deshecho PC Servidor						$ 70.000,00
Deschecho Adaptador USB-Ethernet						$ 1.000,00
Deschecho Cámara						$ 5.000,00
Deshecho Raspberry PI +3B						$ 10.000,00
Deshecho Motes openmote-cc5538						$ 96.000,00
Deshecho taladro de mano						$ 12.000,00
Deshecho amoladora de mano						$ 11.000,00
Taladro de mano	-$ 60.000,00					
Amoladora de mano	-$ 55.000,00					
Pc Servidor	-$ 350.000,00					
Motes (cant 4)	-$ 480.000,00					
Adaptador USB-Ethernet	-$ 5.000,00					
Cámara Hickvision	-$ 25.000,00					
Raspberry PI +3B	-$ 50.000,00					
Capital de trabajo	-$ 1.115.000,00					$ 1.115.000,00
Flujo del proyecto	-$ 2.140.000,00	$ 1.271.251,00	$ 1.271.251,00	$ 1.271.251,00	$ 1.271.251,00	$ 1.320.000,00

Figura 44: Flujo de caja

A continuación, se expone el cálculo del capital de trabajo para el flujo de caja expuesto anteriormente.

Capital de trabajo												
Ingreso mensual primer año	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre
Fabricación												
Unidades vendidas	1	1	1	1	1	1	1	1	1	1	1	1
Precio de venta	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00
Total	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00
Egresos												
Materiales	-$ 960.000,00	-$ 960.000,00	-$ 960.000,00	-$ 960.000,00	-$ 960.000,00	-$ 960.000,00	-$ 960.000,00	-$ 960.000,00	-$ 960.000,00	-$ 960.000,00	-$ 960.000,00	-$ 960.000,00
Mano de obra	-$ 120.000,00	-$ 120.000,00	-$ 120.000,00	-$ 120.000,00	-$ 120.000,00	-$ 120.000,00	-$ 120.000,00	-$ 120.000,00	-$ 120.000,00	-$ 120.000,00	-$ 120.000,00	-$ 120.000,00
Costos fijos	-$ 35.000,00	-$ 35.000,00	-$ 35.000,00	-$ 35.000,00	-$ 35.000,00	-$ 35.000,00	-$ 35.000,00	-$ 35.000,00	-$ 35.000,00	-$ 35.000,00	-$ 35.000,00	-$ 35.000,00
Capital de trabajo												
Ingresos	0	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00	$ 1.250.000,00
Total	-$ 1.115.000,00	$ 135.000,00	$ 135.000,00	$ 135.000,00	$ 135.000,00	$ 135.000,00	$ 135.000,00	$ 135.000,00	$ 135.000,00	$ 135.000,00	$ 135.000,00	$ 135.000,00
Acumulado	-$ 1.115.000,00	-$ 980.000,00	-$ 845.000,00	-$ 710.000,00	-$ 575.000,00	-$ 440.000,00	-$ 305.000,00	-$ 170.000,00	-$ 35.000,00	$ 100.000,00	$ 235.000,00	$ 370.000,00

Figura 45: Capital de trabajo

5.2.1 Aproximación al valor actual neto

El Valor Actual Neto (VAN) es una medida financiera clave en la evaluación de proyectos. Representa la diferencia entre los flujos de efectivo futuros generados por el proyecto y la inversión inicial requerida. Al calcular el VAN, podemos determinar la rentabilidad económica del proyecto y su capacidad para generar un retorno positivo.

El Valor Actual Neto (VAN) calculado para este proyecto asciende a $ 2.470.236,84.

Se observa un VAN positivo, por lo que este proyecto otorga ganancia en comparación de la tasa de interés fijada.

5.2.2 Tasa interna de retorno

La tasa interna de retorno (TIR) calculado para este proyecto es de: 52%

5.2.3 Payback o plazo de recuperación

El payback o plazo de recuperación del presente proyecto es de 1,68 años aproximadamente (1 año y 8 meses).

5.2.4 Productos y servicios de otros fabricantes

Al momento de realizar este informe no se encontraron productos y servicios de otros fabricantes que compitan con el actualmente desarrollado. Esto se debe principalmente al bajo nivel de programación del firmware posibilitado a través de la conexión remota.

6 CONCLUSIONES Y ANEXOS

La creación del Laboratorio Remoto de Dispositivos IoT (LRIOT) supuso un desafío tanto a nivel técnico como organizacional, requiriendo la aplicación de conocimientos adquiridos a lo largo de la carrera universitaria, así como la investigación y adquisición de nuevos conocimientos sobre tecnologías y dispositivos empleados en la industria, siendo esta integración un objetivo primordial del proyecto.

El desarrollo del LRIOT permitió adquirir experiencia en la interpretación de los requisitos del cliente, la planificación y ejecución de un proyecto en sus diversas etapas, así como la adaptación a cambios de enfoque cuando resultó necesario. Asimismo, se mejoraron las habilidades comunicativas mediante la entrega de informes y la interacción con el director del laboratorio GRIDTICS.

Este laboratorio representa una innovación significativa en el campo de IoT, siendo único en el país por sus características. Su implementación abarca múltiples objetivos académicos de la facultad, como la enseñanza, investigación y desarrollo en tecnologías IoT, además de brindar oportunidades económicas mediante la impartición de cursos y consultorías.

Se proyecta que el LRIOT sirva como base para la creación de un ecosistema de Investigación y Desarrollo (I+D) en IoT/IIoT, incorporando diversas tecnologías relacionadas y potenciando la posición de la UTN-FRM como actor clave en estos campos. Se considera la integración de tecnologías operacionales (OT) junto con aspectos de ciberseguridad, Machine Learning, Inteligencia Artificial, computación en la nube, entre otros, para establecer un ecosistema sólido en el contexto de la Industria 4.0.

A continuación se exponen los documentos desarrollados durante el proyecto y su objetivo:

- Anexo_Manual-usuario-lab-iiot: especifica como un usuario genérico debe realizar la reserva de los entornos de desarrollo. Se explica mediante

ejemplos la programación, compilación y carga del firmware así como la descarga de archivos con extensión .pcap mediante SFTP.

- Anexo_Manual-administrador-lab-iiot: este manual tiene como objetivo que el usuario administrador pueda agregar nuevos usuarios para que utilicen el laboratorio así como resolver problemas generales con la reserva y conexión al laboratorio remoto.

- Anexo_Manual-arquitectura-lab-iiot: este manual está abocado a suplir el objetivo de enseñanza del laboratorio, ya que explica como están compuesta las placas openmote-cc2538, las características del estándar IEEE 802.15.4, el protocolo IPv6, RPL, la solución 6loWPAN, CoAP y MQTT.

- I+IoT Rlab Laboratorio Remoto de IoT e IIoT, para la enseñanza, la asistencia y consultoría a empresas: documento de investigación tipo short desarrollado con el objetivo de presentar el proyecto en el CACIC (Congreso Argentino de Ciencias de la Computación). El proyecto fue considerado como de interés para los evaluadores del congreso y aceptado para su exposición.

- Folleto Laboratorio IIoT: folleto desarrollado para ser utilizado en reunión con personal del INTI para ofrecer laboratorio para su uso y posterior trabajo en conjunto. Se planteó además utilizar el mismo para promocionar el laboratorio.

7 BIBLIOGRAFÍAS Y REFERENCIAS BIBLIOGRÁFICAS

A continuación, se exponen las referencias utilizadas para la configuración y despliegue de la infraestructura desarrollada, y también la bibliografía utilizada para la programación de las placas openmote:

- Infodark. (2023, febrero 02). Instalación y configuración básica de Proxmox. Recuperado de https://www.infodark.net/linux/61-instalacion-y-configuracion-basica-de-proxmox
- The Apache Software Foundation (s.f.). VCL 2.5 Installation Guide. Recuperado de https://vcl.apache.org/docs/VCL25InstallGuide.html
- Universo Digital (2023, febrero). Crear un contenedor LxC en Proxmox VE paso a paso. Recuperado de https://universodigital.org/crear-contenedor-lxc-en-proxmox-ve-paso-a-paso-2/
- Frank Aaron Peeler (2020, mayo 22) How to Provision Linux and Unix Lab Computers. Recuperado de https://cwiki.apache.org/confluence/display/VCL/How+to+Provision+Linux+and+Unix+Lab+Computers
- Matt Wildman (2021, septiembre 13) Installing Apache Guacamole on Ubuntu and Debian. Recuperado de https://www.linode.com/docs/guides/installing-apache-guacamole-on-ubuntu-and-debian/
- Infodark. (2023, febrero 04). Creación “básica” de máquinas virtuales en Proxmox VE. Recuperado de https://www.infodark.net/linux/62-creacion-basica-de-maquinas-virtuales-en-proxmox
- Zentyal (2021, julio 13). Instala Zentyal rápidamente sobre Ubuntu 20.04, Servidor o Escritorio. Recuperado de https://zentyal.com/es/news/instala-zentyal-rapidamente-sobre-ubuntu-20-04-servidor-o-escritorio/
- The Apache Software Foundation (s.f.) Apache Guacamole Manual. Recuperado de https://guacamole.apache.org/doc/gug/
- The Apache Software Foundation (s.f.). LDAP Authentication. Recuperado de https://vcl.apache.org/docs/ldapauth.html
- The Apache Software Foundation (s.f.) LDAP Authentication. Recuperado de https://guacamole.apache.org/doc/gug/ldap-auth.html

- Contiki-NG (s.f.) Toolchain installation on Linux. Recuperado de: https://docs.contiki-ng.org/en/develop/doc/getting-started/Toolchain-installation-on-Linux.html
- Erin Glass (2020, Mayo 21) Como instalar Nginx en Ubuntu 20.04. Recuperado de: https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-20-04-es
- TechExpertTIPS (s.f.) Instalación OPNsense [Paso a paso]. Recuperado de: https://techexpert.tips/es/opnsense-es/instalacion-de-opnsense-paso-a-paso/
- LEARNLINUXTV(s.f.) How to Set Up an NFS Server on Ubuntu (Complete with AutoFS!). Recuperado de: https://www.learnlinux.tv/how-to-set-up-an-nfs-server-on-ubuntu-complete-with-autofs/
- Brian Boucheron (2020, Junio 11) Como proteger Nginx con Let's Encrypt en Ubuntu 20.04. Recuperado de: https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-20-04-es
- Steve's Internet Guide (2024, Enero 20) Using The Mosquitto_pub and Mosquitto_sub MQTT Client Tools- Examples. Recuperado de: http://www.steves-internet-guide.com/mosquitto_pub-sub-clients/
- RaspberryPI (s.f.) RaspberryPi. Recuperado de: https://www.raspberrypi.com/software/
- Waher, P.: Learning Internet of Things, 1st Edition. Packt Publishing, UK (2015).
- OpenMote-cc2538. Disponible en https://doc.riot- os.org/group boards openmotecc2538.html. Julio 2023.
- Contiki: The Open Source OS for the Internet of Things. Disponible en https://www.contiki- os.org. Mayo 2019.
- Kushalnagar, N., Montenegro, G., Schumacher, C.: IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals (2007).

- Molisch, A.F., Balakrishnan, K., Chong, C.C., Emami, S., Fort, A., Karedal, J., Kunisch, J., Schantz, H., Schuster, U. and Siwiak, K.,: IEEE 802.15. 4a channel model-final report. IEEE P802, 15(04), p.0662 (2004).
- Vilajosana X., Tuset P., Watteyne T. y Pister K. OpenMote: Open-Source Prototyping Platform for the Industrial IoT. (PDF) OpenMote: https://www.researchgate.net/publication/280068237_OpenMote_Open-Source_Prototyping_Platform_for_the_Industrial_IoT