

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“VISUALIZACIÓN EN TIEMPO REAL DE OCUPACIÓN DE ESPACIOS EN CAMPUS SAN JOAQUÍN MEDIANTE SISTEMA DE EDGE COMPUTING CON LORA”

JOSÉ MIGUEL QUEZADA SILVA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesora Guía: Erika Rosas
Profesor Correferente: José Luis Martí

Marzo - 2024

DEDICATORIA

Dedico este trabajo a mi papá, sin el que no estaría en este punto en el que me encuentro,
y por sus enseñanzas y los buenos tiempos que pasamos, a pesar de todo.

También se lo dedico a mi futuro ahijado, que sé que crecerá lleno de amor, apoyo y
comprensión incondicional.

AGRADECIMIENTOS

Profesora Erika: Por sus enseñanzas como profesora durante la carrera y por su apoyo y esfuerzo constante durante el transcurso de este trabajo.

Roberto León: Por tener la buena voluntad de permitirme usar su clase de Computación Científica para entrenar y poner a prueba el modelo.

Carmen Valdez: Por facilitar el mapa del campus San Joaquín.

Jeremy Ellis: Por desarrollar el código para la comunicación por LoRa, y dedicar tiempo a resolver dudas relacionadas con el Portenta H7.

Martín, César, Felipe, Pancho, Ben y Félix: Por ayudarme a etiquetar fotos para entrenar los modelos de *Crowd Counting*.

Mateo: Por instalar la batería en el Portenta H7 y por tu infinita ayuda e inspiración para seguir aprendiendo de Arduino y electrónica en general.

Edu, Octavio, Cris, Pauli y Maite: Por su inspiración e ideas cuando tenía dudas y el apañe en el FabLab.

Alberto y Martín: Por ayudarme testeando y por apañar siempre, los quiero cabros.

Luna y Flo: Por presionarme a seguir poniéndole y ayudarme a distenderme cahuineando y siendo cachos caipirinha de caja, las tkm geis.

Gracias a Rorritox, Zhuo, Max, Jorge, Gabo, Axel, Pantufla, Pancho, Gus, Ale, Felipe, Pancho, Madrid, Jadol, Joan, Miguel, Benja, Nico, Cervela, JP y Maestro Esteban, por todo el apoyo y motivarme a seguir esforzándome cada día a ser mejor. Es fácil mirarse en menos y es muy importante seguir de todas formas, para poder sentir confianza y alegría en el mundo en el que vivimos.

Seba: Por todo tu apoyo y cariño mientras crecíamos juntos, y porque sé que siempre estarás aquí con nosotros, a pesar de la distancia.

Francisco y Camila: Por el apoyo, consejos y risas en la casa donde me siento siempre acogido y querido, que tengan un futuro brillante.

Mamá: Por todo el amor, cariño y valores que me has inculcado, estoy infinitamente agradecido de ser tu hijo, te amo mucho.

Javier: Por siempre estar ahí, confiando en mí, por inspirarme y apoyarme en todo este proceso y todos los demás. Te amo mucho, chancho.

RESUMEN

Resumen— Se diseña un sistema de *Edge Computing* que utiliza LoRa como medio de comunicación, para obtener el nivel de ocupación de las salas del campus San Joaquín de la Universidad Técnica Federico Santa María, los que son visualizados en una plataforma web mediante un mapa interactivo. Esto se realiza a partir de modelos de *Machine Learning* adaptados para ser lo suficientemente ligeros para ser ejecutados en el microprocesador Arduino Portenta H7. Estos modelos son integrados en un programa en lenguaje Arduino/C++ encargado de capturar y redimensionar fotografías de tres salas del campus, en las que se entrenaron específicamente. Se realizan tres pruebas de comunicación mediante LoRa, cuyos resultados promedio corresponden a pérdida de paquetes: 63 %, RSSI: -108 [dBm], RSSI de paquete: -111 [dBm] y SNR: 2. Luego, se realizan tres pruebas de Crowd Counting en conjunto con LoRa, obteniendo un RMSE normalizado promedio de 0.2, y resultados promedio de comunicación correspondientes a pérdida de paquetes: 14 %, RSSI: -106 [dBm], RSSI de paquete: -101 [dBm] y SNR: 0. Se concluye que, a pesar de lograr los objetivos planteados en el trabajo, el reconocimiento visual y la comunicación entre dispositivos no son lo suficientemente eficaces actualmente como para ser aplicados en un sistema real en el Campus San Joaquín. Sin embargo, se proponen sugerencias para continuar con su desarrollo como trabajo futuro.

Palabras Clave— LoRa; Enumeración de personas; Aprendizaje profundo; Computación Edge; Arduino

ABSTRACT

Abstract— A LoRa-based Edge Computing system is designed, to calculate the occupancy of the classrooms of the San Joaquin Campus of the *Universidad Tecnica Federico Santa Maria*, which are then visualized on a web platform through an interactive map. This is accomplished using Machine Learning models adapted to be as lightweight as possible for execution on the Arduino Portenta H7 microprocessor. These models are integrated into an Arduino/C++ program responsible for capturing and resizing photographs of three specific rooms on the campus, where they were specifically trained. Three LoRa communication tests are conducted, yielding average results of packet loss: 63 %, RSSI: -108 dBm, Packet RSSI: -111 dBm, and SNR: 2. Subsequently, three Crowd Counting tests are conducted in conjunction with LoRa, resulting in an average normalized RMSE of 0.2, and average results of LoRa communication of packet loss: 14 %, RSSI: -106 dBm, Packet RSSI: -101 dBm, and SNR: 0. The conclusion is drawn that, despite achieving the objectives of the study, the visual recognition and communication between devices are currently not efficient enough to be applied in a real system on the San Joaquin Campus. However, suggestions are proposed for further development as future work.

Keywords— LoRa; Crowd counting; Deep learning; Edge computing; Arduino

GLOSARIO

CC: *Crowd Counting*.

CNN: Redes Neuronales Convolucionales.

DL: *Deep Learning*.

IA: Inteligencia Artificial.

IoT: Internet de las Cosas.

IWMS: Sistemas de Gestión de Lugares de Trabajo Integrados.

LPWAN: Red de Área Amplia de Baja Potencia.

MAE: Error Absoluto Medio.

ML: *Machine Learning*.

NN: Redes Neuronales Artificiales.

P2P: *Peer-to-peer*.

RSSI: Indicador de fuerza de señal recibida.

RMSE: Raíz del error cuadrático medio.

SF: *Spreading Factor*.

SNR: *Signal-to-Noise Ratio*.

UTFSM: Universidad Técnica Federico Santa María.

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	V
GLOSARIO	VI
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS	X
CAPÍTULO 1: INTRODUCCIÓN	1
1.1 Motivación	1
1.2 Definición del problema	2
1.3 Tecnologías a usar	3
1.4 Objetivos	4
1.5 Alcance	4
1.6 Impacto	5
1.7 Metodología	5
1.8 Organización del documento	7
CAPÍTULO 2: MARCO CONCEPTUAL y ESTADO DEL ARTE	9
2.1 Machine Learning	9
2.2 Deep Learning	10
2.2.1 Redes Neuronales Convolucionales	11
2.3 TinyML	11
2.4 Crowd Counting	12
2.4.1 Métodos basados en detección	12
2.4.2 Métodos basados en regresión	13
2.4.3 Métodos basados en estimación de densidad	13
2.4.4 Métodos basados en <i>Deep Learning</i>	14
2.4.5 Métricas de error	15
2.5 Edge Computing	15
2.6 LoRa	17
2.7 Nociones de desarrollo web	20
2.8 Ocupación de espacios	22
2.9 Crowd Counting	24
2.10 LoRa	25
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	27
3.1 Requerimientos	27
3.1.1 Requerimientos Funcionales	27
3.1.2 Requerimientos no Funcionales	27
3.2 Diseño del sistema	28

3.2.1	Capa Edge	28
3.2.2	Capa Gateway	29
3.2.3	Capa Web/Cloud	29
3.3	Selección de Hardware	29
3.3.1	Capa Edge	29
3.3.2	Capa Gateway	31
3.4	Reconocimiento visual y <i>Crowd Counting</i>	33
3.4.1	Diseño de CNN	33
3.4.2	Crowd Counting	34
3.5	Comunicación LoRa	35
3.5.1	Arduino IDE y librerías	36
3.6	Visualización en plataforma web	37
3.6.1	Backend y Base de datos	37
3.6.2	Frontend	38
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN		41
4.1	Pruebas preliminares de LoRa	41
4.1.1	Prueba 1: Dispositivos ubicados en extremos opuestos	42
4.1.2	Prueba 2: <i>Gateway</i> ubicado en centro (Cuarto piso)	45
4.1.3	Prueba 3: <i>Gateway</i> ubicado en centro (Segundo piso)	48
4.1.4	Promedio de pruebas de LoRa	51
4.2	Pruebas en conjunto de CC + LoRa	51
4.2.1	Sala 1: LDS	52
4.2.2	Sala 2: Sala A-016	56
4.2.3	Sala 3: Oficina de FabLab	59
4.2.4	Promedio de pruebas de CC + LoRa	61
4.3	Discusión	62
CAPÍTULO 5: CONCLUSIONES		64
5.1	Trabajo futuro	65
ANEXOS		67
5.1	Prueba 1 LoRa	67
5.2	Prueba 2 LoRa	69
5.3	Prueba 3 LoRa	71
REFERENCIAS BIBLIOGRÁFICAS		73

ÍNDICE DE FIGURAS

1	Árbol del problema, con sus causas y efectos.	3
2	Diagrama de metodología en sus distintas fases.	7
3	Arquitectura general de Redes Neuronales Convolucionales.	10
4	Crowd Counting basado en estimación de densidad. (a) Imagen de entrada, (b) Mapa de densidad real, (c) Mapa de densi- dad estimada.	13
5	Crowd counting a partir de Redes neuronales convolucionales.	14
6	Diagrama de arquitectura general de red de <i>Edge Computing</i>	16
7	Ilustración del CSS, cuya transmisión se genera mediante frecuencias variables.	18
8	Cobertura de ancho de banda dependiendo de la distancia alcanzada por distintas tecnologías. LoRa cuenta con bajos anchos de banda para un amplio espectro de distancias.	19
9	Arquitectura de red basada en LoRa.	20
10	Diagrama de aplicación web.	21
11	Visualización de ocupación de vagones en Línea 6 de Metro de Santiago.	22
12	FOMO convierte las imágenes de entrada en mapas de segmentación, los que pueden ser considerados como mapas de calor.	25
13	<i>Diagrama de arquitectura de sistema propuesto</i>	28
14	<i>Arduino Portenta H7</i> (izquierda), con la extensión <i>Vision Shield</i> (derecha).	30
15	Portenta H7 con soportes 3D y antena de transmisión LoRa.	31
16	<i>Raspberry Pi 3B+</i> con extensión <i>Dragino</i>	33
17	Herramienta VIA, utilizada para etiquetar imágenes en campus.	34
18	Formato de mensajes enviados por LoRa.	35
19	Modelo de Bases de Datos.	37
20	<i>Frontend</i> de plataforma web con niveles de ocupación de un subconjunto de salas del Campus San Joaquín.	39
21	Ubicaciones de <i>Gateway</i> (azul) y nodo (verde, para recepción exitosa y rojo, en caso contrario) al transmitir mensajes de prueba cada 10 segundos en primer experimento.	43
22	RSSI de paquetes recibidos con respecto a la distancia del nodo móvil al <i>Ga- teway</i> en prueba 1 de LoRa.	44
23	Gráfico de violín indicando la distribución de mensajes recibidos y no recibi- dos con respecto a su distancia al <i>Gateway</i> en prueba 1 de LoRa.	45
24	Ubicaciones de <i>Gateway</i> (azul) y nodo móvil (verde para recepción efectiva y rojo en caso contrario) al transmitir mensajes de prueba cada 10 segundos en segundo experimento.	46
25	RSSI de paquetes recibidos con respecto a la distancia del nodo móvil al <i>Ga- teway</i> en prueba 2 de LoRa.	47
26	Gráfico de violín indicando la distribución de mensajes recibidos y no recibi- dos con respecto a su distancia al <i>Gateway</i> en prueba 2 de LoRa	48
27	Ubicaciones de <i>Gateway</i> (azul) y nodo móvil (verde y rojo) al transmitir men- sajes de prueba cada 10 segundos en tercer experimento.	49

28	RSSI de paquetes recibidos con respecto a la distancia del nodo móvil al <i>Gateway</i> en prueba 3 de LoRa.	50
29	Gráfico de violín indicando la distribución de mensajes recibidos y no recibidos con respecto a su distancia al <i>Gateway</i> en prueba 3 de LoRa	50
30	Fotos de LDS en dos momentos distintos de la experimentación.	53
31	Funcionamiento de receptor LoRa en terminal de Raspberry Pi.	53
32	Funcionamiento de server de <i>Backend</i> recibiendo consultas HTTP para agregar y obtener datos.	54
33	Mapa de campus San Joaquín con niveles de ocupación en un conjunto de salas. El LDS se encuentra destacado con un contorno rojo.	55
34	CC calculado por Portenta en comparación con valor real aproximado, a través del tiempo, en LDS.	56
35	Fotos de la sala A-016 durante la prueba, obtenidas con celular exclusivamente para referencia de cantidad de personas.	57
36	Mapa de campus San Joaquín con nivel de ocupación de sala A-016, indicado por la etiqueta con su nombre.	58
37	CC calculado por Portenta en comparación con valor real aproximado, a través del tiempo, en sala A-016.	58
38	Foto de la oficina del FabLab durante la prueba, obtenida con celular.	59
39	CC calculado por Portenta en comparación con valor real aproximado, a través del tiempo, en Oficina del FabLab.	60
40	Mapa de campus San Joaquín indicando nivel de ocupación de la Oficina del FabLab, indicada por el recuadro con su información correspondiente.	61
41	RSSI de paquete de mensajes recibidos con respecto a la distancia del nodo móvil al <i>Gateway</i> en prueba 1 de LoRa.	67
42	SNR de mensajes recibidos con respecto a la distancia del nodo móvil al <i>Gateway</i> en prueba 1 de LoRa.	67
43	RSSI de paquete de mensajes recibidos con respecto a la distancia del nodo móvil al <i>Gateway</i> en prueba 2 de LoRa.	69
44	SNR de mensajes recibidos con respecto a la distancia del nodo móvil al <i>Gateway</i> en prueba 2 de LoRa.	69
45	RSSI de paquete de mensajes recibidos con respecto a la distancia del nodo móvil al <i>Gateway</i> en prueba 3 de LoRa.	71
46	SNR de mensajes recibidos con respecto a la distancia del nodo móvil al <i>Gateway</i> en prueba 3 de LoRa.	71

ÍNDICE DE TABLAS

1	Tabla con valores correspondientes a SF entre 7 y 10, junto con la tasa de bits, distancia y tiempo en el aire para un <i>payload</i> de 11 <i>bytes</i>	19
2	Valores de parámetros de comunicación por LoRa.	42
3	Valores promedio de resultados de prueba 1 de LoRa.	42
4	Valores promedio de resultados de prueba 2 de LoRa.	46

5	Valores promedio de resultados de prueba 3 de LoRa.	48
6	Valores promedio de resultados de las tres pruebas de LoRa.	51
7	Valores promedio de resultados de prueba de CC + LoRa en LDS.	54
8	Valores promedio de resultados de prueba de CC + LoRa en sala A-016. . . .	56
9	Valores promedio de resultados de prueba de CC + LoRa en Oficina del FabLab.	59
10	Valores promedio de resultados de las tres pruebas de CC + LoRa.	62
11	Resultados de prueba 1, con nodos ubicados en extremos opuestos del campus y envío de mensajes cada 10 segundos.	68
12	Resultados de prueba 2, con <i>Gateway</i> ubicado en una sala central del campus en cuarto piso (12 metros de altura), y Portenta enviando mensajes cada 10 segundos.	70
13	Resultados de prueba 3, con <i>Gateway</i> ubicado en segundo piso de edificio F (3 metros de altura), y Portenta enviando mensajes en dirección a <i>Gateway</i> cada 10 segundos.	72

CAPÍTULO 1

INTRODUCCIÓN

1.1. Motivación

La aparición de tecnologías emergentes con el potencial de cambiar el funcionamiento de la vida cotidiana, ha sido de interés público en los últimos tiempos. Entre ellas, el Internet de las Cosas (IoT, por sus siglas en inglés) se destaca como una alternativa prometedora, que consiste en una red interconectada de dispositivos que cumplen funciones como recopilar y compartir datos en tiempo real, con tal de mejorar la disponibilidad de información y toma de decisiones.

Los denominados *Smart Cities* y *Smart Campus* son algunas de las formas en que este concepto se ha visto aplicado recientemente, donde la información se utiliza para mejorar la calidad de vida de las comunidades, con casos como *Amazon Sidewalk*¹, una red de dispositivos propios de *Amazon*, conectados entre sí mediante tecnologías como Bluetooth Low Energy y LoRa, con usos como rastreo de mascotas y de personas con demencia². Estas aplicaciones son posibles gracias a las redes de baja potencia y largo alcance (LPWAN), como LoRa, las que, como indica el nombre, proveen comunicación a larga distancia con un consumo de energía limitada, pero con la desventaja de contar con bajas velocidades de transmisión de datos [Raychowdhury y Pramanik, 2020].

La forma tradicional de organización de estas arquitecturas es mediante el paradigma de computación *Cloud*, que se basa en el procesamiento de datos y programas en grandes dispositivos centralizados, ofreciendo servicios en servidores individuales o múltiples. Por otro lado, el *Edge Computing* lleva la capacidad de procesamiento directamente al lugar donde se generan los datos, lo que permite la reducción de la latencia y mejora la eficiencia de los sistemas en que se aplica, en contraste con la tecnología *Cloud*.

Al aplicar estos conceptos en entornos urbanos y laborales, se pueden obtener datos en tiempo real sobre la ocupación de espacios, permitiendo una gestión más eficiente y flexible. El análisis del espacio, basado en conceptos como el *space occupancy* y *space utilization*, los que describen la cantidad de personas presentes en un espacio determinado, se beneficia enormemente de estas tecnologías, facilitando una comprensión más profunda y precisa del uso de estos espacios.

Las definiciones de estos términos varían según la fuente. Según SpacelQ [Maresco, 2022], el primer término se refiere a la "medida total de empleados en un lugar de trabajo en un

¹*Amazon Sidewalk*, https://docs.aws.amazon.com/es_es/iot/latest/developerguide/amazon-sidewalk-overview.html

²*Echo, Tile, and Level devices join Amazon Sidewalk*, <https://www.aboutamazon.com/news/devices/echo-tile-and-level-devices-join-amazon-sidewalk>

momento dado, la que suele ser utilizada junto con la capacidad total para obtener la tasa de ocupación”, mientras que la utilización de espacio se referiría a “la proporción en la que los empleados utilizan un espacio en particular” en un periodo de tiempo puntual o extendido. *iOFFICE*[Miller, 2021] incluye además en su definición los recursos utilizados por las personas, junto con el tamaño necesario del área, considerando posibles modificaciones a esta.

Por otro lado, en estudios académicos tales como [Lau *et al.*, 2018] y [Manoharan *et al.*, 2020], se usan estos términos intercambiablemente, lo que da a entender que su diferenciación es más relevante en el ámbito empresarial, en los que se prioriza la optimización de recursos económicos. Sin embargo, al no ser este el enfoque de este trabajo, se utiliza únicamente el concepto de “tasa de ocupación de espacio”, entendida como la proporción entre la cantidad de personas que utilizan una zona en particular y su capacidad máxima.

1.2. Definición del problema

A partir de esto, se puede estudiar el uso ineficiente de espacios, debido a una combinación entre una utilización sub-óptima de un conjunto dado de zonas, y su excesiva o escasa ocupación. Esta problemática es identificada en el campus San Joaquín de la Universidad Técnica Federico Santa María (UTFSM), el más reciente de esta institución. Fue fundado el 2009 y cuenta actualmente con 14 carreras, la última de las cuales corresponde a la de Ingeniería Civil Física, incluida el año 2023. El campus se encuentra establecido en un terreno de aproximadamente 44 mil metros cuadrados, lo que, junto con la inclusión de 4 carreras nuevas en tan solo 2 años motivaron movilizaciones a causa del hacinamiento del estudiantado.

En este contexto, el problema tiene que ver, como se puede apreciar en la figura 1, con cómo los usuarios afectados ocupan y se distribuyen en las distintas zonas del campus en su día a día, cuyo mal uso puede incomodar e, incluso, llegar a dificultar el desarrollo de sus actividades, ya sean estudio, recreación, alimentación o descanso. Los actores afectados son identificados como las diferentes personas que ocupan habitualmente las dependencias del campus, o sea, estudiantes, docentes, administrativos y auxiliares.

Entre los principales síntomas visibles, se encuentra en primer lugar la congestión de personas en áreas y momentos específicos, como en los horarios de descanso y almuerzo. Si bien, espacialmente, esto se encuentra limitado a la cantidad de lugares disponibles, temporalmente, contar con la información de los momentos en que hay mayor disponibilidad podría ayudar en este caso. Otro efecto consiste en la dificultad en el flujo normal, al haber mayor congestión en rutas específicas al moverse de un punto a otro por el campus, a su vez, aumentando la probabilidad de accidentes, en especial en casos de emergencia, como terremotos. Además, en escenarios de pandemia, como la del COVID-19, este mal uso de espacios puede generar un aumento en la tasa de contagios, como puede pasar en un futuro no tan lejano con la propagación de otros agentes biológicos.

Por otro lado, el origen del problema puede ser explicado por la mala distribución de las personas dentro de él y por falta de conocimiento de la disponibilidad, o quizás incluso de la existencia, de otros espacios, destinados a los distintos usos ya mencionados. Otra causa notable es el hacinamiento, un problema persistente en el campus por años, que solo se ha visto aliviado por la implementación de clases en línea, debido a la pandemia. No obstante, una vez retomada por completo la presencialidad, este inconveniente seguirá siendo una preocupación, especialmente con la inclusión de más carreras en el campus y, por ende, más estudiantes. En especial, si se tiene en cuenta el aumento continuo de personas matriculadas en la Universidad y, en el caso específico del campus San Joaquín, su acotado tamaño, para la cantidad de personas que se encuentran asistiendo a él.

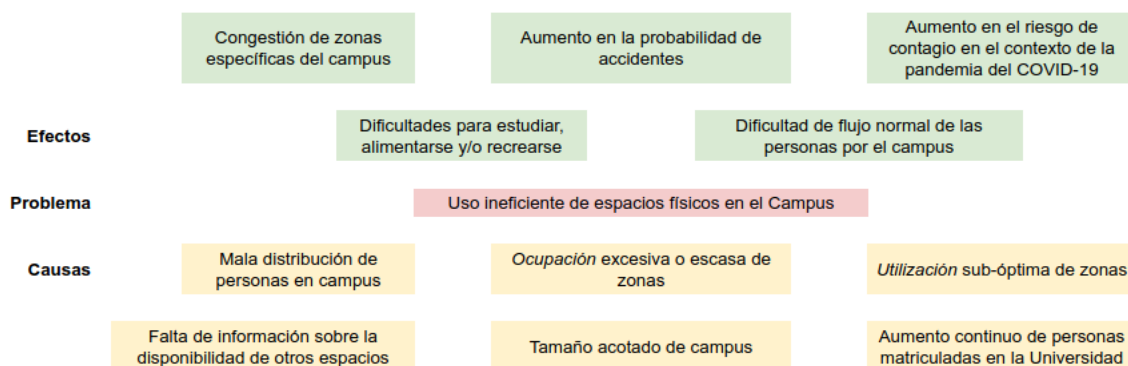


Figura 1: Árbol del problema, con sus causas y efectos.
Fuente: Elaboración propia

1.3. Tecnologías a usar

Para mitigar los problemas identificados, es necesario considerar la adopción de tecnologías y enfoques que ayuden al desarrollo de una solución que se adapte a la realidad del campus. Un ejemplo de esto es LoRa, que consiste en un protocolo de comunicación de datos a larga distancia, enfocado en dispositivos de baja potencia, ya que corresponde a un LPWAN. Sus propiedades incluyen un buen rendimiento y consumo energético reducido, lo que además de disminuir costos, puede significar una menor cantidad de contaminación producida por el sistema [Adelantado *et al.*, 2017].

LoRa tiene además la particularidad de reducir el efecto de las interferencias y otras fallas en la comunicación, permitiendo así la entrega efectiva de información en tiempo real entre lugares distantes. De esta forma, se hace factible la posibilidad de contar con tan solo un nodo central para recibir datos de múltiples edificios, optimizando el uso de material y la complejidad de la configuración del sistema.

Al utilizar LoRa como base, se hace factible la implementación de una arquitectura *Edge*, cuyo objetivo reside en la distribución del procesamiento de los datos, ubicándolo en el borde de la red, así disminuyendo la carga de los nodos centrales y, por ende, la cantidad de tráfico y la probabilidad de la generación de cuellos de botella.

En cuanto a la estimación de ocupación de espacios, se considera el uso de técnicas de *Deep Learning*, lo que consiste en el uso de modelos de Redes Neuronales Artificiales, concepto ubicado dentro del área de la Inteligencia Artificial y el Aprendizaje de Máquinas. Su particularidad consiste en el aprendizaje automático mediante capas compuestas por nodos denominados neuronas.

1.4. Objetivos

General: Desarrollar un sistema de visualización en tiempo real de ocupación de espacios del campus San Joaquín usando Edge Computing y LoRa.

Específicos:

1. Analizar e implementar un modelo de *Machine Learning* específico para dispositivos *Edge*, para reconocer y cuantificar personas usando la cámara del dispositivo.
2. Configurar una arquitectura *Edge* basada en LoRa para la transmisión de los resultados a larga distancia y con un consumo de energía reducido.
3. Desarrollar una aplicación web que muestre el nivel de congestión para cada zona del campus.

1.5. Alcance

Durante el periodo de desarrollo de este trabajo, se pretende probar y analizar un acercamiento al caso real, a partir de la construcción de un prototipo que logre satisfacer los objetivos propuestos. El prototipo será empleado en una cantidad reducida de zonas en el Campus San Joaquín, con tal de comprobar su efectividad antes de ser expandido por toda el área, a partir de *hardware* disponible actualmente en el Laboratorio de Redes de Computadores. De implementarse efectivamente, permitiría una integración completa del sistema, adquiriendo la capacidad de ampliar sus horizontes, mediante su aplicación en otras organizaciones, e incluso otros rubros.

1.6. Impacto

La pandemia del Coronavirus en Chile, iniciada a principios del 2020, ayudó a controlar el problema planteado, gracias a la posibilidad de no asistir de manera presencial a la Universidad, mediante la implementación de la metodología de clases en línea. Esto mantuvo a la gran mayoría del estudiantado fuera de esta, hasta años después, en caso de los pertenecientes a otras regiones. Sin embargo, solo es una solución temporal, tanto debido al retorno progresivo a clases de manera presencial, como al espacio limitado con el que cuenta el campus, el que se vería en la obligación de remover espacios de recreación o utilidad para continuar con su crecimiento.

Es por esto que se hace necesario tratar este problema de manera innovadora, mediante una solución que contribuya a las herramientas de información disponibles para todos los actores a los que afecta esta situación a diario, y que optimice algo más que solo la utilización de los espacios, sino que también el tiempo ahorrado al evitar dirigirse a lugares de alta ocupación.

De todas maneras, si bien puede que su impacto no sea aparente en un comienzo, en cuanto al cambio en el comportamiento de los usuarios finales, en el tiempo se espera notar un uso más informado de las instalaciones del campus en general, gracias al uso del sistema, mejorando así la convivencia y el aprovechamiento de los distintos espacios que ofrece la Universidad para su desenvolvimiento.

Además, se podrían subsanar los mismos problemas identificados en el campus San Joaquín, pero extendidos a un mayor número de instituciones universitarias, dada la gran población estudiantil de educación superior presente en el país, consistiendo de más de 1.2 millones el 2021 [Mineduc, 2021], además de organizaciones de otro tipo en que se identifique el mismo problema. De esta forma, mediante el uso de LoRa, el sistema puede ser implementado en sedes que no estén directamente contiguas, como la Universidad Diego Portales (UDP), Andrés Bello (UNAB) y la Universidad de Santiago de Chile (USACH), ya que presentan la particularidad de contar con grupos de sedes ubicadas en la misma zona de la ciudad, pero separadas incluso por calles.

1.7. Metodología

Con tal de asegurar el éxito de los objetivos propuestos, se requiere de la definición explícita de una metodología de trabajo que rija su desarrollo, especificando los pasos a seguir durante este.

En primer lugar, es necesario aclarar que esta se trata de una investigación de tipo experimental y cuantitativa, al tratarse de la evaluación de un sistema a ser diseñado y puesto a prueba, obteniendo datos numéricos, a ser comparados para buscar los valores de los distintos parámetros que logren cumplir sus objetivos de mejor forma.

A partir de esto, y del análisis de metodologías ya existentes para proyectos, tanto de investigación como de programación, se propone una inspirada en Scrum³, un marco de trabajo basado en la filosofía ágil, y que consiste en ciclos iterativos, o *sprints*, de investigación, desarrollo y pruebas de su ejecución, a través del transcurso del trabajo.

En este caso particular, se consideran etapas iniciales, seguidas de etapas cíclicas, las que se repetirán en numerosas ocasiones para cada componente del sistema a elaborar, y etapas finales (ver Figura 2), las que son definidas a continuación.

- **Investigación previa:** Investigación del estado del arte de las distintas tecnologías a utilizar. Se considera como actividad inicial, y puede pertenecer a los primeros ciclos, con tal de establecer los fundamentos de conocimiento del trabajo, y que no sean modificados a medida que este progrese.

Ciclos:

- **Investigación y obtención de datos:** Esta etapa se diferencia de la investigación previa en cuanto a que, en este caso, se estudia el funcionamiento de cada componente, integrando cada vez más características y solucionando más problemas de implementación. Esto sumado a la obtención de los datos necesarios para el entrenamiento de modelos de *Machine Learning*.
- **Desarrollo de programa y configuración de dispositivos:** Diseño y confección de la lógica y los programas de cada componente, junto con la configuración necesaria de los dispositivos a utilizar en cada caso.
- **Pruebas:** *Testing* de los programas, tanto en entornos de desarrollo, como al desplegarlos en los equipos correspondientes. Esta etapa se enfoca en los errores y el funcionamiento en sí de los componentes, más que en los valores obtenidos en cada uno.
- **Análisis de resultados:** Verificación del funcionamiento correcto de los programas y *hardware*, registrando cualquier fallo para solucionarlo en la siguiente iteración.

Etapas finales:

- **Integración de subsistemas:** Una vez en funcionamiento los distintos subsistemas, se procede a realizar su integración, que asegure el tratamiento adecuado de los datos recibidos en cada componente.
- **Pruebas de sistema integrado:** Luego de definir los parámetros de prueba, se registran los resultados obtenidos al modificarlos durante la ejecución del sistema.

³<https://www.atlassian.com/agile/scrum>

- **Análisis de resultados finales:** Al tratarse de una investigación de tipo cuantitativa, se busca comparar el desempeño promedio del sistema, determinado por las métricas numéricas de *testing*, a partir de la visualización de los resultados obtenidos.
- **Conclusiones:** Sintetizar el trabajo y análisis realizados, para llegar a las conclusiones pertinentes, el valor del estudio y el trabajo futuro.

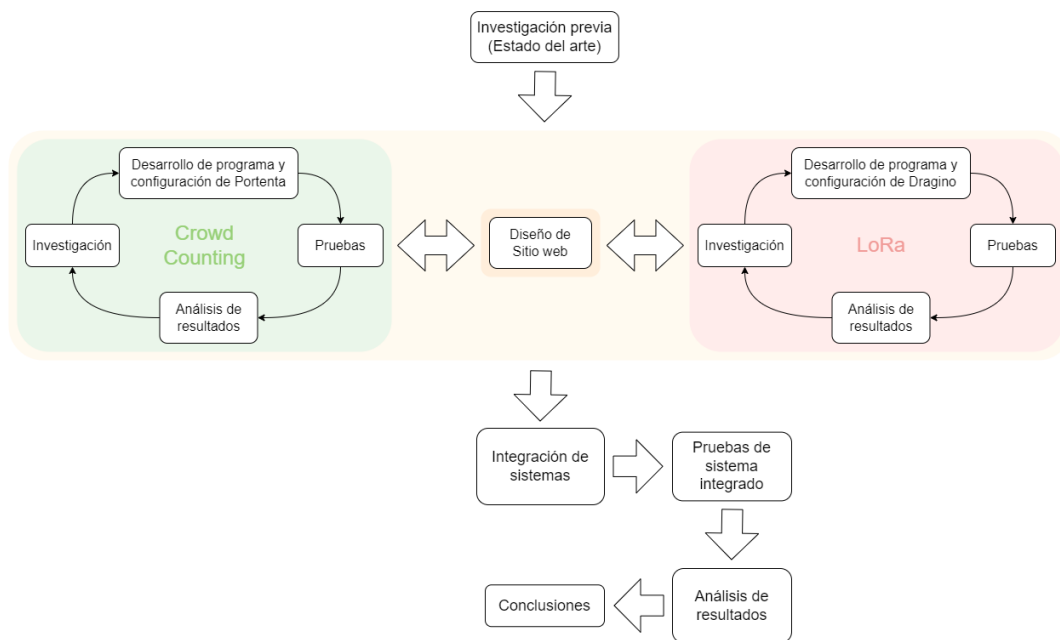


Figura 2: Diagrama de metodología en sus distintas fases.

Fuente: Elaboración propia

1.8. Organización del documento

El documento se encuentra organizado en los siguientes capítulos. El Capítulo 2, que se divide en el marco conceptual y el estado del arte, en los que se describen los conocimientos necesarios para comprender el trabajo realizado, y las investigaciones y trabajos previos que han intentado dar solución al problema planteado, respectivamente.

En el Capítulo 3 se presenta la propuesta de solución para el problema planteado, junto con el hardware específico utilizado para cumplir con los objetivos planteados, y las técnicas utilizadas para su construcción.

En el Capítulo 4, se muestra la validación del sistema diseñado, en la que se explican los experimentos realizados para ponerlo a prueba, además de discutir la metodología empleada y analizar los requerimientos establecidos.

Y por último, en el Capítulo 5, se presentan las conclusiones del trabajo realizado, a partir de los resultados obtenidos, y se aborda el trabajo futuro.

Posteriormente se incluyen los anexos con información adicional y las referencias bibliográficas utilizadas a través del documento.

CAPÍTULO 2

MARCO CONCEPTUAL y ESTADO DEL ARTE

Esta sección se compone, en primer lugar, del marco conceptual, en el que se explican de forma básica los fundamentos necesarios para comprender el desarrollo del trabajo realizado, y en segundo, del estado del arte, donde se describen soluciones ya existentes, tanto para intentar dar solución al problema, como para los distintos usos que se le ha dado a estas tecnologías en los últimos tiempos. De esta manera, recorriendo conceptos relacionados con aprendizaje de máquinas y conteo de multitudes, y seguidos por tecnologías relacionadas con la transmisión de datos, como LoRa y la computación *Edge*.

Marco Conceptual

Los temas descritos a continuación son tratados para comprender el funcionamiento íntegro del sistema a desarrollar. Para entender el funcionamiento del modelo de reconocimiento de los niveles de ocupación, se describen los conceptos de *Machine Learning*, *Deep Learning* y *TinyML*. En cuanto a la forma en que se mide la cantidad de personas a partir de ML, se aborda el concepto de *Crowd Counting* y las métricas de error utilizadas en este proceso. Por el lado de la comunicación, se describen la computación *Edge* y el protocolo LoRa. Por último, se incluye información básica sobre desarrollo web, utilizado para la plataforma de visualización de la ocupación en las distintas salas del campus.

2.1. Machine Learning

El *Machine Learning* (ML) o Aprendizaje Automático es una subárea de la Inteligencia Artificial (IA), cuyo fin consiste en el diseño de modelos que sean capaces de aprender a realizar distintos tipos de tareas, a partir de la configuración de sus parámetros. Estos son ajustados de acuerdo a los datos reales en procesos de entrenamiento, lo que produce una mejor exactitud en los resultados predictivos del modelo, obteniendo así uno cada vez más robusto. Estos conjuntos de datos, pueden o no estar etiquetados con el resultado esperado a ser predicho por el modelo. Esto último corresponde a la principal diferencia entre los métodos de aprendizaje supervisado y no supervisado, respectivamente [Badillo *et al.*, 2020].

Dentro del espacio de posibles modelos, denominado espacio de hipótesis, el objetivo es lograr escoger la alternativa que pueda representar de la manera más fiel y precisa los distintos comportamientos de los problemas. Además, se busca determinar la mejor configuración para los parámetros aprendidos, adquiriendo la capacidad de predecir resultados en situaciones no conocidas hasta el momento, mediante interpolación y extrapolación de los resultados.

Las tareas a aprender mediante aprendizaje supervisado se dividen en problemas de clasificación, los que buscan identificar las distintas clases a las que pertenecen los datos, y problemas de regresión, encargados de generar la función que mejor represente la distribución de los datos. Como se verá más adelante, el problema planteado en este trabajo se trata como uno de clasificación, considerando un enfoque basado en *Deep Learning*.

2.2. Deep Learning

El *Deep Learning* (DL) consiste, a su vez, en una subárea específica del ML, que cuenta con la particularidad de poder representar el conocimiento de manera jerarquizada, a partir de una estructura de capas, que permite la comprensión de conceptos simples en los niveles de entrada, utilizados por las capas sucesivas para lograr la síntesis de conceptos más complejos a medida que se profundiza el modelo [Goodfellow *et al.*, 2016].

Esto es logrado a través de lo que se conocen como Redes Neuronales Artificiales (ANN: *Artificial Neural Networks* en inglés o NN por simplicidad), que consisten en conjuntos de nodos, conocidos como neuronas, conectados entre sí mediante enlaces y dispuestos en capas. Estas se distribuyen en la capa de entrada, la capa de salida, y las capas ocultas, ubicadas entremedio (ver Figura 3).

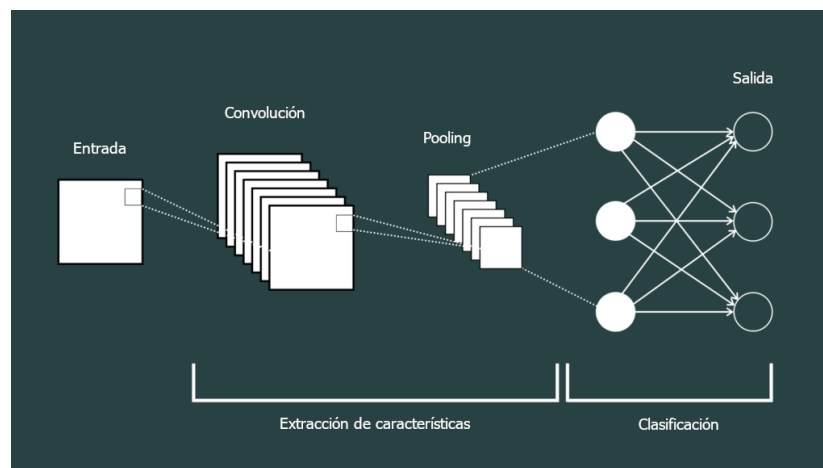


Figura 3: Arquitectura general de Redes Neuronales Convolucionales.
Fuente: [Reader, 2022] (Adaptación).

Otra de las propiedades que caracterizan este paradigma es la capacidad de realizar de manera automatizada la tarea de selección de las características relevantes para el aprendizaje de la tarea asignada, en vez de tener que ser realizada por humanos, como es el caso tradicional.

Al contar con estructuras tan complejas, se hace necesario proveer una gran cantidad de

datos de entrenamiento, por lo que se deben adoptar técnicas para obtener datos de manera automatizada. Una comúnmente utilizada para lograr esto es el *Data Augmentation*, correspondiente a la aplicación de distintas transformaciones a los datos actuales, con tal de obtener nuevos datos, que al no ser similares a los originales, ayuda en la disminución del denominado *overfitting*, o sobreajuste, correspondiente al caso en que los modelos se ajustan bien a los datos de entrenamiento, pero no son capaces de inferir resultados de datos diferentes.

2.2.1. Redes Neuronales Convolucionales

Dentro de las redes neuronales, se encuentran las Redes Neuronales Convolucionales (CNN: *Convolutional Neural Networks* en inglés), las que, a diferencia de las NN tradicionales, están diseñadas para tratar con datos representados de forma matricial de distinta dimensionalidad, siendo los casos más comunes los datos temporales en una dimensión y los visuales en dos dimensiones [Goodfellow *et al.*, 2016].

En el procesamiento de los datos en este tipo de modelos, se encuentra el uso de las funciones matemáticas de Convolución, utilizadas para procesar e interpretar las características de los datos de cada capa matricial, y *Pooling* para reducir la dimensionalidad y hacer que el análisis espacial de características sea independiente de la ubicación en la que se realiza [Gholamalinezhad y Khosravi, 2020].

2.3. TinyML

Con tal de acotar estos temas al presente trabajo, se considera el concepto de *TinyML*, correspondiente a arquitecturas, *frameworks*, herramientas y enfoques relacionados con ML, capaces de realizar análisis en el dispositivo mediante diversas modalidades de detección, ya sean visión, audio, voz, movimiento, entre otros, con bajo consumo de energía. Esto se enfoca predominantemente en dispositivos embebidos⁴, alimentados por baterías y adecuados para su aplicación en casos de uso a gran escala, preferiblemente en el ámbito del Internet de las Cosas o de redes de sensores inalámbricas [Ray, 2022].

Los lineamientos de esta subárea proponen que el software para ejecutar TinyML debería ser lo más compacto posible, con tal de lograr un ahorro energético significativo, que estos sistemas deberían especializarse en optimizar distintos modelos de ML para obtener mejor precisión a pesar de las limitaciones de recursos. Además se incentiva cumplir con otros requerimientos tales como permitir la ejecución de modelos de ML en dispositivos *Edge* que obtengan energía a partir del entorno (*energy-harvesting*) y que se alimenten por batería,

⁴Dispositivo embebido: Dispositivo, generalmente capaz de realizar procesamiento digital, responsable de realizar un subconjunto de tareas dentro de un sistema más grande con un propósito final definido y con uno o más de este tipo de dispositivos [Awati, 2022].

que estos modelos sean capaces de escalar a trillones de sensores y dispositivos *Edge*, y el desarrollo de programas que puedan ser almacenados en tan solo unos *Kilobytes* en la RAM del dispositivo.

2.4. Crowd Counting

El *Crowd Counting* (CC), traducido como conteo de multitudes, corresponde a la estimación de la cantidad de individuos que comparten una determinada región. Este constituye un desafío vital para una serie de aplicaciones en distintos ámbitos, tales como televigilancia, seguridad pública, planificación urbana, y monitoreo de tráfico [Gouiaa *et al.*, 2021].

Si bien, existen estrategias para contar personas mediante parámetros no visuales (Como *People Counting*, que consiste en la enumeración de personas que cruzan un umbral [Group, 2023]), el CC se encuentra dentro del campo del *Computer Vision*, una subárea de la IA que consiste en el uso de computadores para interpretar datos visuales para una amplia variedad de aplicaciones industriales, médicas, en vehículos automatizados, e identificación y seguimiento de objetos, entre otros [Xu *et al.*, 2021].

El CC ha sido tratado de distintas formas mediante el uso de métodos tradicionales del *Computer Vision*, sin embargo, parte importante de su investigación y desarrollo han sido realizados los últimos años gracias al explosivo avance en tecnologías basadas en DL y, particularmente, en CNNs. Esta evolución de métodos se describe a continuación.

2.4.1. Métodos basados en detección

La forma inicial para estimar la cantidad de personas en una imagen consiste en detectar a cada una de ellas mediante distintas técnicas y modelos de ML tradicional, dependiendo del modo de detección [Elbishlawi *et al.*, 2020]. Si bien es intuitiva, se trata de la técnica más costosa en lo que a recursos respecta.

Los métodos para detectar entidades se dividen en:

- *Detección monolítica o completa*: Se identifican *features* o características necesarias para detectar cuerpos completos de personas, a partir de modelos de clasificación lineales o no lineales, siendo estos últimos más lentos, pero precisos. Finalmente el clasificador revisa la imagen entera en búsqueda de personas para contarlas.
- *Detección parcial*: En vez de tener que detectar el cuerpo completo, se identifica solo la cabeza o los hombros en adición a esta. Esta técnica ayuda a disminuir parcialmente el problema de la obstaculización, producido por la existencia de obstáculos cubriendo partes de las personas a detectar.

2.4.2. Métodos basados en regresión

A pesar del intento para solucionar la obstaculización que propone la detección parcial, esta es propensa a fallos en escenarios con alta densidad de personas. Teniendo esto en consideración, surge un nuevo método, construido en torno a dos pasos. En primer lugar, la detección de características de bajo nivel, como texturas, bordes y esquinas a partir de gradientes, por ejemplo. Y en segundo, el empleo de una función de regresión, como regresión lineal, gaussiana, o de Ridge, con tal de asociar la cantidad de personas a los valores obtenidos en el primer paso.

2.4.3. Métodos basados en estimación de densidad

Estos buscan incorporar la información espacial de las imágenes como parámetro adicional de aprendizaje, mediante la producción de un mapa de densidad, que indica la probabilidad de que la cabeza, o la parte del cuerpo a identificar, de cada persona esté en una ubicación determinada. Estos mapas pueden ser calculados de distintas formas, por ejemplo, mediante la minimización de una función de pérdida, a partir del mapeo entre la imagen de entrada y su mapa de densidad real, o *ground truth* (Ver Figura 4).

Finalmente, la estimación de la cantidad de personas se realiza integrando sobre secciones locales de la imagen, en vez de contar cada persona de manera individual, logrando mejores resultados que los métodos anteriores, sobre todo en casos de una gran densidad de personas.

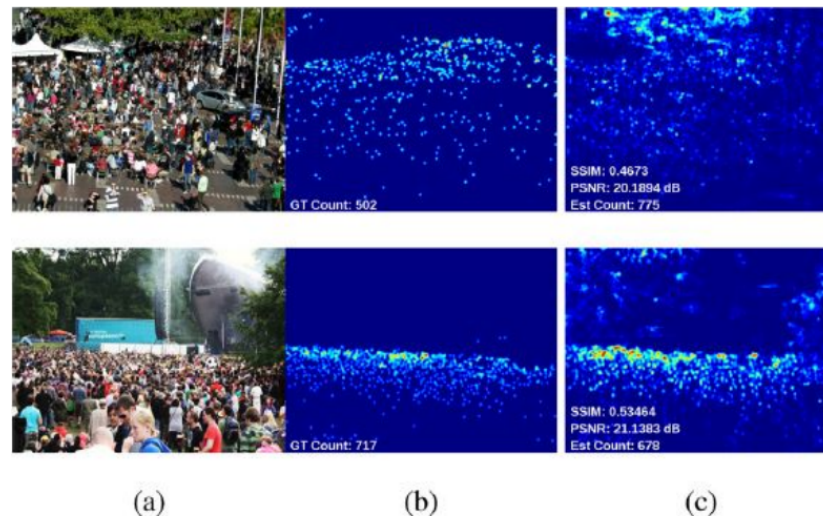


Figura 4: Crowd Counting basado en estimación de densidad.
(a) Imagen de entrada, (b) Mapa de densidad real, (c) Mapa de densidad estimada.

Fuente: [Elbishlawi et al., 2020].

2.4.4. Métodos basados en *Deep Learning*

Recientemente, debido al crecimiento del área de ML y de las CNN aplicadas a reconocimiento y análisis visual, se han desarrollado métodos para el CC basados en DL, que, como se discutió previamente, ofrecen procesos automatizados de aprendizaje de características para la identificación de personas, mejorando la precisión de los resultados y teniendo a disposición una gran cantidad de *datasets* para realizar el entrenamiento de los modelos, como el caso de JHU-CROWD++ [Sindagi *et al.*, 2020] (Ver Figura 5).

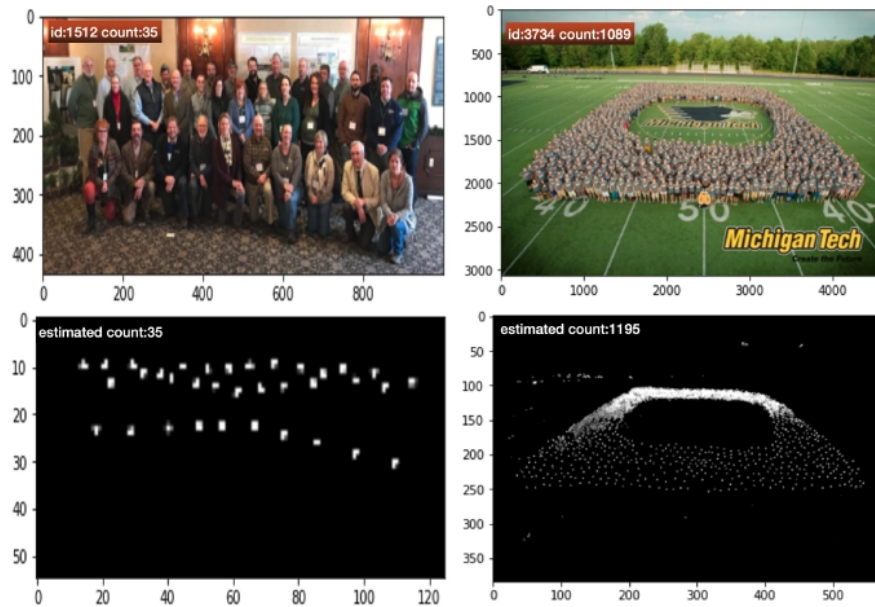


Figura 5: Crowd counting a partir de Redes neuronales convolucionales.

Fuente: [Huang *et al.*, 2021].

Otra ventaja de estos métodos es la posibilidad de utilizar modelos pre-entrenados como base para el diseño de modelos más específicos, los que pueden ser adaptados a las situaciones concretas en que se piensa hacer uso del software a producir, tales como YOLO[Redmon *et al.*, 2016], *EfficientNet*[Simonyan y Zisserman, 2015], VGG-16[Tan y Le, 2020] y *MobileNetV2*[Sandler *et al.*, 2019], entre otros. Estos modelos son capaces de discriminar, con buena precisión personas, animales, vehículos y muebles, entre varias otras entidades. Además, pueden ser adaptados para ser utilizados en dispositivos de *Edge Computing*, al reducir su tamaño, ya sea recortando capas u optimizando el uso de espacios de las variables utilizadas.

2.4.5. Métricas de error

En distintos estudios se utilizan diferentes métricas para medir el error de los modelos de CC. Entre las más populares se encuentran el MAE (Error absoluto medio, por sus siglas en inglés) y el RMSE (Raíz del error cuadrático medio).

Si bien, no hay un consenso de cuál es la mejor métrica en general, para medir los resultados de los experimentos a desarrollar más adelante, se decide utilizar el RMSE, debido a su sensibilidad a los errores grandes y *outliers*⁵. Este indicador proporciona una medida cuantitativa de la diferencia entre los valores predichos y los valores reales de las estimaciones de conteo de personas. Se define a partir de la siguiente fórmula, basada en la distancia cartesiana, como está indicado en [Gouiaa et al., 2021]:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - y_i')^2}$$

Donde N es la cantidad de datos obtenidos en la prueba, y_i es el valor de CC real para el i -ésimo dato, y y_i' es el valor de CC predicho para el i -ésimo dato.

2.5. Edge Computing

El paradigma de *Edge Computing* nace como una alternativa al tradicional *Cloud Computing*. Este último consiste en una serie de servicios de infraestructura, plataformas y software (IaaS, PaaS y SaaS, respectivamente), tales como almacenamiento y procesamiento de datos, utilizando en ocasiones múltiples servidores para asegurar su escalabilidad, resiliencia y disponibilidad. El *Edge Computing*, en cambio, busca desplazar el procesamiento de las redes hacia sus extremos, contando con nodos finales o *end devices* de potencia baja, pero suficiente para procesar los datos adquiridos por sus sensores. De esta manera, es capaz de proveer apoyo a movilidad y ubicación, baja latencia y proximidad al usuario [Khan et al., 2019].

Entre los objetivos principales de esta arquitectura de red, se encuentran una mejor distribución de potencia y consumo energético, mayor seguridad de los datos registrados en el extremo de la red, soporte para movilidad de aplicaciones y disminución de tráfico en los dispositivos de capas más altas, como servidores y *data centers*, reduciendo así la latencia y la probabilidad de congestión en la red.

Los dispositivos propios de esta arquitectura pueden variar dependiendo de los diseños de las redes y de sus objetivos, sin embargo, por lo general, como se ve en la Figura 6, está compuesta por la capa de *Cloud*, en la que se encuentran los distintos servidores, *data centers*, y otros dispositivos de almacenamiento, una capa de nodos *Edge*, compuesta por *gateways* y

⁵Valores atípicos.

routers, encargados de transmitir los datos recopilados hacia las capas superiores y, por último, una capa de sensores y otros dispositivos encargados de adquirir datos de su entorno. Estas últimas capas pueden estar unidas de todas formas, como se aplica en el caso de este trabajo.

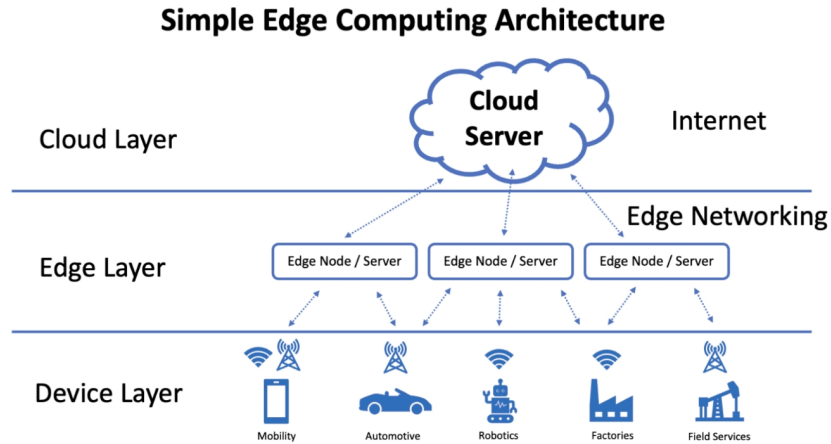


Figura 6: Diagrama de arquitectura general de red de *Edge Computing*.
Fuente: [wipro, 2023].

En el caso de los dispositivos de la capa *Edge*, se encuentran varias alternativas de modelos disponibles para realizar, entre otras tareas, el despliegue de modelos de TinyML. Algunas de estas son presentadas a continuación, en una lista no exhaustiva, con sus características respectivas.

- **Arduino Portenta H7:** Placa desarrollada por Arduino, de la línea Pro, especializada en ejecución de programas de *TinyML*. Cuenta con dos núcleos de 480 MHz y 240 MHz, SDRAM de 8 MB y conectividad por Wi-Fi y Bluetooth. Es compatible con extensiones para aumentar sus capacidades, como el *Portenta Vision Shield*, que cuenta con una cámara de 320x240 píxeles, entrada para tarjeta microSD, y módulo de comunicación mediante LoRa. Precio de \$150.000⁶, Vision Shield: US\$69⁷ (Actualmente \$60.900 aproximadamente).
- **Arduino Nicla Vision:** Placa aun más reducida, con el mismo procesador del Portenta H7, pero con cámara de 2 megapíxeles. Sin embargo, no es compatible con extensiones, y cuenta con una memoria RAM de 1 MB, permitiendo el uso de variables y modelos más pequeños que el Portenta H7. Tiene un precio de US\$95 (Actualmente \$83.860 aproximadamente)⁸.

⁶<https://arduino.cl/producto/arduino-portenta-h7/>

⁷<https://store-usa.ar.duino.cc/products/arduino-portenta-vision-shield-lora>

⁸[urlhttps://store.arduino.cc/products/nicla-vision](https://store.arduino.cc/products/nicla-vision)

- **Arduino Portenta X8:** Placa de la misma línea que el Portenta H7 (Arduino Pro), pero más avanzado, ya que cuenta con una memoria RAM 4DDR de 2 GB y procesador de 9 núcleos. Cuenta con la capacidad de ejecutar un entorno de Linux en conjunto con el motor de Arduino. No cuenta con cámara ni módulo de comunicación por LoRa u otras tecnologías de comunicación *Edge*, por lo que habría que adquirirlas por separado. Tiene un precio de \$295.800⁹.
- **Nvidia Jetson Nano:** Microcomputador desarrollado por Nvidia, cuenta con una GPU integrada de 128 núcleos, RAM de 4 GB LPDDR4, CPU de 4 núcleos, lo que lo hacen una buena alternativa para realizar proyectos de ML con *Computer Vision*. No cuenta con comunicación por LoRa ni con cámara integrada. Cuenta con un precio de \$239.990¹⁰.
- **Google Coral:** Placa de desarrollo compatible con Linux, 1 o 4 GB de RAM DDR4, 8 GB de memoria Flash, 4 núcleos y una ligera tarjeta gráfica integrada. A pesar de contar con un precio asequible, no cuenta con cámara ni módulo de tecnologías *Edge* como LoRa. Además presenta una elevada curva de aprendizaje y dificultad de desarrollo. No disponible en Chile, compra internacional por US\$129¹¹.

2.6. LoRa

LoRa (abreviatura de *Long Range* o largo alcance en inglés) es una tecnología de comunicación a larga distancia, especializada en el uso de dispositivos de baja potencia. LoRa, o *LoRa PHY*, corresponde exclusivamente a la parte física de la comunicación, o sea, la emisión y recepción de ondas electromagnéticas entre dispositivos compatibles. Para contar con operaciones a nivel de usuario, existen distintos protocolos de comunicación de más alto nivel, como LoRaWAN, creado por *LoRa Alliance* en 2015, conformando en conjunto una Red de Área Amplia de Baja Potencia (LPWAN), concepto correspondiente a redes de comunicación a larga distancia con un bajo consumo energético, al igual que bajas velocidades de transmisión de datos [Raychowdhury y Pramanik, 2020].

LoRa PHY, *LoRa-MAC* o *Raw LoRa*, es un método físico de transmisión de datos desarrollado por Semtech en 2012, cuyo nombre se atribuye a las largas distancias que puede alcanzar, teniendo límites de hasta 10 a 20 kilómetros en espacios abiertos, y de 2 a 5 kilómetros en áreas urbanas [Adelantado *et al.*, 2017]. Esto, junto con propiedades tales como un bajo consumo energético e interferencia, hacen ideal su aplicación en escenarios rurales y de emergencia, además de casos como gestión y ahorro de energía, prevención y alertas de desastres, control de comunicación y eficiencia de infraestructura [Semtech, 2022b]. De esta manera, esta tecnología cuenta con una participación clave en el Internet de las Cosas (IoT, por sus siglas en inglés), concepto asociado a la creciente red que incluye objetos de baja potencia, tales

⁹<https://cl.rsdelivers.com/product/arduino/abx00049/arduino-portenta-x8/2457769>

¹⁰<https://www.mechatronicstore.cl/nvidia-jetson-nano-kit-de-desarrollador/>

¹¹<https://coral.ai/products/dev-board>

como sensores y actuadores, conectados entre sí y con otros equipos para el tratamiento de los datos obtenidos de una variedad de sensores [Raychowdhury y Pramanik, 2020].

Para el envío de datos, estos son codificados a través de una tecnología propia, desarrollada por Semtech, basada en la modulación de espectro ensanchado de chirrido (CSS, por sus siglas en inglés). Los espectros ensanchados o dispersos corresponden a “técnicas de telecomunicación en las que una señal es transmitida en un ancho de banda considerablemente mayor que las frecuencias de la información original” [NIST, 2022], de esa manera permitiendo transmisiones con menor interferencia, y mejor inmunidad al efecto *Doppler* y al desvanecimiento multitrayecto, los que pueden producir variaciones en las frecuencias de recepción por distintos tipos de interferencia [Electronic-notes, 2022].

Por otro lado, los chirridos o *chirps*, a los que hace referencia el CSS, corresponden a las señales emitidas en formas de pulsos sinusoidales de frecuencia variante (Ver Figura 7). Esta variación depende de una serie de factores, incluyendo el valor del *Spreading Factor*, descrito más adelante.

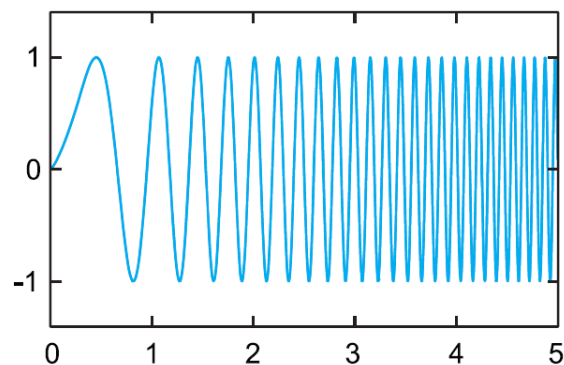


Figura 7: Ilustración del CSS, cuya transmisión se genera mediante frecuencias variables.
Fuente: [Semtech, 2022a].

Ahora, si bien, el uso de CSS no es particularmente eficiente en el uso del ancho de banda de LoRa, sigue siendo más bajo, en general, que otras tecnologías de comunicación disponibles a día de hoy, como puede ser apreciado en la Figura 8, además de proveer una amplia gama de distancias. Esto, sumado a los factores descritos previamente, hacen de LoRa una alternativa más adecuada para su aplicación en el Internet de las Cosas, con respecto a otras tecnologías como *ZigBee*, *Bluetooth*, *Z-Wave*, *WiFi*, y redes para celular como *GSM* y *LTE*, por ejemplo [Raychowdhury y Pramanik, 2020].

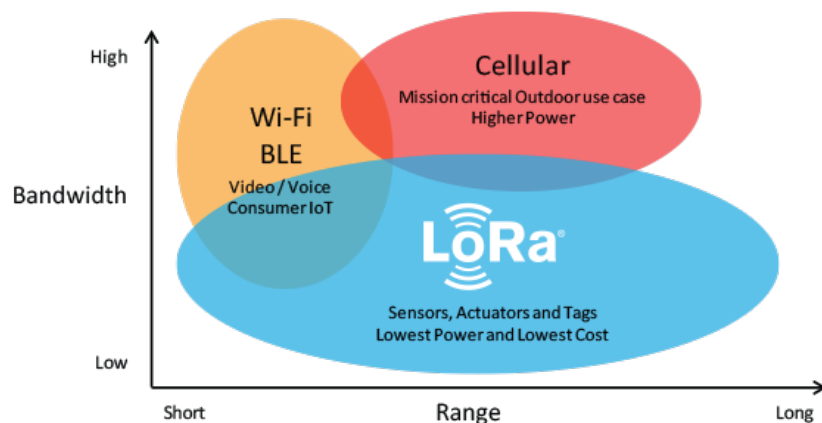


Figura 8: Cobertura de ancho de banda dependiendo de la distancia alcanzada por distintas tecnologías. LoRa cuenta con bajos anchos de banda para un amplio espectro de distancias.
Fuente: [Semtech, 2022b].

La forma en que se emiten los chirridos depende directamente del factor de esparcimiento, o *Spreading Factor* (SF) utilizado, el que define qué tan rápido se modulan y que, a su vez, afecta directamente a la tasa de bits de transmisión, y qué distancia puede alcanzar.

El SF puede tomar seis posibles valores, entre 7 y 12, dependiendo del ancho de banda utilizado y si se tratan de mensajes de *uplink* o *downlink*, en otras palabras, si el nodo está enviando o recibiendo mensajes de un *Gateway* o cualquier equipo de una capa más elevada en la red. Por ejemplo, en la Tabla 1, se encuentran los valores correspondientes a SF7 hasta SF10, para *uplink* de 125 kHz. Como se puede apreciar, mientras más alto sea el SF, más baja será la frecuencia de chirridos, disminuyendo la velocidad de transmisión de datos, pero a la vez aumentando el alcance al que puede llegar la señal antes de producir errores en la recepción.

<i>Spreading Factor</i> (Para envío a 125 KHz)	Tasa de bits	Rango (depende del terreno)	Tiempo en Aire (<i>payload</i> de 11 bytes)
SF10	980 bps	8 km	371 ms
SF9	1760 bps	6 km	185 ms
SF8	3125 bps	4 km	103 ms
SF7	5470 bps	2 km	61 ms

Tabla 1: Tabla con valores correspondientes a SF entre 7 y 10, junto con la tasa de bits, distancia y tiempo en el aire para un *payload* de 11 bytes.
Fuente: [Semtech, 2022a].

Mientras LoRa PHY se encuentra encargada de la parte física de la comunicación, existen protocolos, ubicados en las capas superiores, que proveen operaciones a niveles más altos de abstracción, siendo LoRaWAN la principal alternativa.

Su arquitectura de red está configurada con una topología de estrella, permitiendo la comunicación desde múltiples dispositivos finales hacia *Gateways* registrados, los que permiten la conexión al resto del Internet [Lora-Alliance, 2022], como puede verse en la Figura 9. Hoy en día cuenta con usos en diversos contextos, tales como *Smart Buildings*, *Smart Cities*, *Smart Industry*, *Smart Logistics* y *Utilities*.

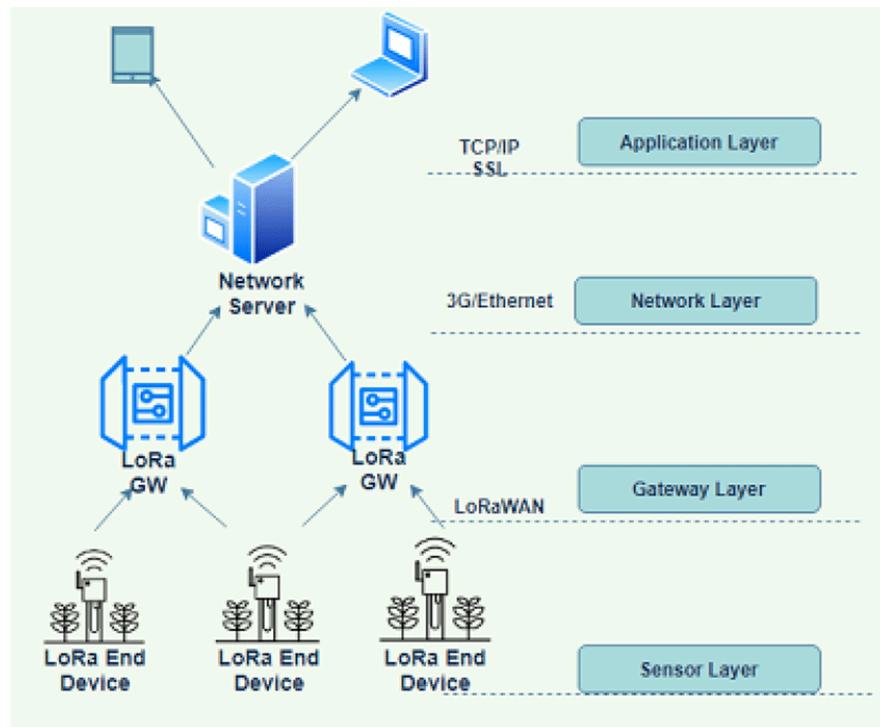


Figura 9: Arquitectura de red basada en LoRa.

Fuente: [Gupta y Snigdh, 2022].

Sin embargo, también es posible configurar arquitecturas alternativas a LoRaWAN, dependiendo del uso específico que se le quiera dar a la red. Un ejemplo sería la configuración *Peer-to-Peer* (P2P, por sus siglas en inglés), que permite la comunicación bidireccional entre los dispositivos que integran la red.

2.7. Nociones de desarrollo web

Si bien, este trabajo no se enfoca en desarrollo web, es necesario definir sus conceptos básicos, al contar el sistema con una plataforma web para visualizar los resultados obtenidos en cada sala del campus. Un diagrama simple que incluye los componentes presentados es visualizado en la Figura 10.

- **Frontend:** Componente encargado de los elementos visuales con los que interactúan

los usuarios, manejando las distintas acciones y eventos Suele estar programada en lenguajes tales como PHP o Javascript para la lógica, junto con HTML y CSS para el diseño de los elementos visibles.

- **Backend:** Responsable de la infraestructura de la plataforma, tanto para la comunicación entre el *Frontend* y la base de datos, como para la gestión de consultas enviadas al servidor.
- **Base de Datos:** Sistema de almacenamiento de datos, en este caso, organizados como una base de datos relacional, o sea, que los guarda en tablas de entidades que tienen con alguna relación particular entre sí, lo que permite una mejor interpretación de los datos y mayor facilidad para su gestión.

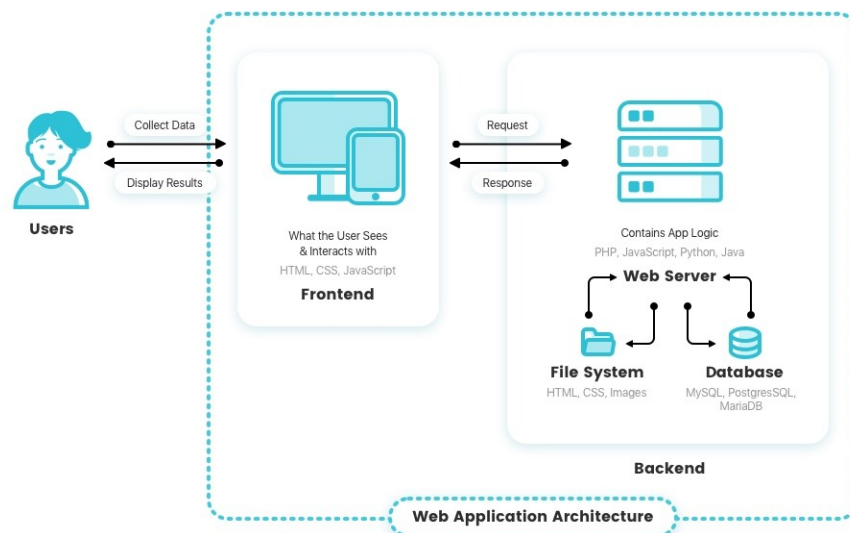


Figura 10: Diagrama de aplicación web.
Fuente: [Dabbs, 2019]

Estado del arte

2.8. Ocupación de espacios

Para solucionar el problema, se han propuesto alternativas que han sido aplicadas en distintas organizaciones y contextos, para obtener métricas de la ocupación de sus áreas disponibles, y así poder informarlas a los usuarios correspondientes.

El primer caso corresponde a la visualización de ocupación de vagones en las líneas 3 y 6 del Metro de Santiago, a través de pantallas ubicadas en los andenes (Ver Figura 11). Estas indican tres niveles de ocupación para cada vagón mediante colores. Esta métrica se calcula, según la prensa [Cooperativa.cl, 2018], estimando la cantidad de personas a partir del peso total medido por los sensores de peso a partir, considerando el peso promedio de un adulto, definido como 76 kilogramos.



Figura 11: Visualización de ocupación de vagones en Línea 6 de Metro de Santiago.

Fuente: [Cooperativa.cl, 2018]

Este caso simplemente no es aplicable a este trabajo, debido a la nula practicidad del uso de este tipo de sensores a través del campus, y a la alta imprecisión que significa estimar la ocupación a partir del peso promedio de las personas a ser enumeradas. Sin embargo, la solución propuesta utiliza un método más efectivo, que considera la individualidad de cada persona, en vez de utilizar un valor arbitrario como es la masa promedio de una persona, lo que puede llevar a importantes imprecisiones en los resultados.

Por otro lado, en el contexto empresarial, las tarjetas de identificación y escaners RFID son utilizadas en oficinas para contar la cantidad de personas presentes en sus distintas áreas, además de sensores de proximidad y movimiento, tanto en escritorios, como en aparatos de iluminación [Booqed, 2022]. Estos son comúnmente utilizados para generar estadísticas

y análisis de la ocupación de espacios posterior a la recolección de estos datos.

Otra alternativa típicamente utilizada en oficinas consiste en el uso de LiDAR o ToF (*Light detection and ranging* y *Time-of-Flight*, respectivamente) para realizar *People Counting* en entradas, y así detectar a las personas que ingresan y salen de distintas zonas a partir de sensores de infrarrojo o luz, las que miden la distancia o tiempo entre el objeto en donde rebotan estas señales y el sensor, para luego identificar si los contornos detectados corresponden a personas o entidades distintas [Terabee, 2023] [Sampath, 2019]. De esa forma se puede mantener una cuenta de cuántas personas se encuentran dentro de un espacio determinado, ya sea en salas o en un edificio completo, y enviar los resultados a dispositivos centrales, ya sea mediante WiFi o redes ubicuas, como LoRa.

De vuelta a la realidad universitaria, si bien, los estudiantes utilizan tarjetas de identificación para acceder al campus, su uso no es práctico para identificar los lugares específicos en los que se encuentra cada uno. Y, en cuanto a los sensores mencionados, se requeriría de una cantidad muy grande para estimar de manera precisa los distintos niveles de ocupación, en especial teniendo en cuenta las ocasiones en que múltiples estudiantes se suelen ubicar en un mismo escritorio. En cuanto al análisis por LiDAR o ToF en puertas, su aplicación podría ser útil en lugares cerrados. Sin embargo, dado que se busca aplicar en un campus que cuenta con varios espacios abiertos, el uso de este tipo de sensores no es suficiente, cuyo rango de alcance permite mediciones precisas solo hasta pasillos amplios, como máximo.

Por último, los Sistemas de Gestión de Lugares de Trabajo Integrados (IWMS por sus siglas en inglés), que consisten en *suites* que proveen una serie de herramientas de gestión de empresas, tales como gestión de instalaciones, bienes raíces, energía y espacios. WorkInSync¹², Planon Software¹³ e IBM Tririga¹⁴ son ejemplos de IWMS que incluyen servicios de gestión de espacios, los que pueden incluir reservas, análisis gráficos y cuantitativos de ocupación de espacios, entre otras funciones.

De estas tres, IBM Tririga es la solución más completa y efectiva para el problema planteado, siendo capaz de realizar “*supervisión de ocupación en tiempo real, planificación dinámica de espacios, mapeo de interiores y un asistente de IA*”. Sin embargo, a pesar de estas características, tienen un alcance limitado y técnicas insuficientes para contabilizar a las personas, además de altos precios, tanto para su compra, como para la capacitación de las personas encargadas de utilizarlo, puesto que además cuentan con curvas de aprendizaje elevadas.

¹²WorkInSync, <https://www.workinsync.io>

¹³Planon Software, <https://planonsoftware.com>

¹⁴IBM Tririga, <https://www.ibm.com/cl-es/business-operations/space-management>

2.9. Crowd Counting

Como se indica en la sección anterior, el *Crowd Counting* (CC) ha progresado mediante distintas etapas, llegando a las estructuras complejas que las componen hoy en día, pero, a la vez, logrando adaptarse a escenarios más restringidos en cuanto al uso de recursos.

Tanto en [Elbishlawi *et al.*, 2020] como en [Gouiaa *et al.*, 2021], se presentan estudios de distintos modelos y estructuras de NNs propuestas en la última década para tratar el CC, los que son evaluados y comparados a partir de un conjunto de *datasets* compuestos por imágenes con multitudes de distinta densidad, y etiquetadas tanto con la cantidad de personas presentes, como con la posición de las cabezas, con tal de que los modelos logren aprender a identificar a las personas, a pesar de la presencia de obstáculos visuales. Además comparan entre métodos tradicionales y basados en CNN, concluyendo que las segundas obtienen un mucho mejor rendimiento. Cada uno también presenta conclusiones con respecto a distintos conceptos, tales como el *Crowd Analysis* o análisis de multitudes, y la aplicación de CC en vehículos aéreos no tripulados.

Sin embargo, a pesar de contar con una calidad destacable, la implementación de estos modelos se mantiene ligado al uso de un elevado poder de procesamiento y de *hardware*, debido a su creciente complejidad, por lo que se investigan soluciones más “livianas”, como en el caso de [Huang *et al.*, 2021], [Wang *et al.*, 2021] y [Zhu *et al.*, 2022], quienes diseñan distintas arquitecturas de CNN orientadas a computación *Edge*, proponiendo distintos avances para alcanzar la mayor precisión posible, asegurándose al mismo tiempo de cumplir con las restricciones de los dispositivos empleados. Para esto, se incluye el uso de modelos pre-entrenados, como *EfficientNet* y *VGG-16*, comprimiendo sus estructuras para aprovechar el máximo potencial dentro de las restricciones.

De la misma forma, *Edge Impulse*¹⁵ desarrolla un modelo, denominado FOMO (Faster Objects More Objects), para detectar objetos en tiempo real, enfocándose en dispositivos de baja potencia, gracias al uso de *TinyML*, soportando oficialmente microprocesadores tales como el Arduino Portenta H7 [Louis Moreau, 2022]. Su funcionamiento se basa en disminuir el tamaño de las imágenes de entrada y convertirlas en mapas de segmentación, cuyos píxeles son clasificados como la clase de interés o como el fondo de la imagen, o sea, cualquier otro objeto.

¹⁵<https://www.edgeimpulse.com/>

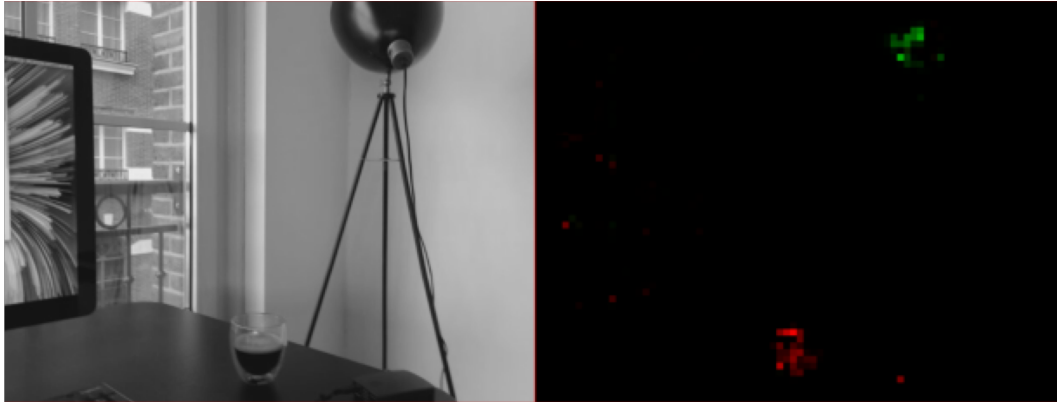


Figura 12: FOMO convierte las imágenes de entrada en mapas de segmentación, los que pueden ser considerados como mapas de calor.

Fuente: [Impulse, 2022]

Uno de los procesos fundamentales de la optimización de los modelos consta en la conversión de su formato, cambiando las dependencias de *TensorFlow* a *TensorFlow Lite*. Para reducir el tamaño del modelo, se emplea una operación denominada cuantificación de tipo entero de 8 bits, el que permite exclusivamente el uso de operaciones y variables de este tipo en el modelo, ahorrando tiempo de computo y uso de memoria.

2.10. LoRa

Como fue establecido, LoRa cuenta con características que la hacen preferible frente a otras alternativas, debido a la versatilidad y accesibilidad que han permitido su aplicación para una variedad de escenarios de la vida real, tanto para usos industriales y particulares, como en investigación. Sin embargo, existen relativamente pocos estudios sobre ocupación de lugares, y prácticamente nulos cuando se añade el factor de la computación *Edge*. En cuanto a *Crowd Counting*, no se encontró ningún estudio que utilizara LoRa. De todas formas, a continuación, se nombran algunos de los casos de ocupación de espacios que aprovechan esta tecnología.

En [Adeogun *et al.*, 2019] se diseña un sistema de medición de la ocupación de espacios exclusivamente cerrados basado en LoRa, pero sin el uso de *Computer Vision*, sino mediante una variedad de sensores, como temperatura, presión, humedad, luz ambiental y UV, entre otros. Además, tan solo se emplean modelos de clasificación binaria, para determinar si hay presencia de personas o no, y de clasificación multi-clase, para saber si no hay personas, si hay una, o si hay 2 o más, resultando bastante limitada para el problema planteado.

Por otro lado, la empresa europea *IMBuildings* ofrece un dispositivo para realizar esta tarea en oficinas¹⁶, y otro para una gama más amplia de ubicaciones¹⁷. Ambos transmiten sus resultados mediante LoRa, y utilizan sensores infrarrojos para medir el movimiento de personas, tanto en entradas de salas, como en pasillos y otros ambientes, indicando como uno de los argumentos para el uso de este tipo de sensores, el riesgo de seguridad que podría significar el empleo de cámaras. Sin embargo, este podría ser mitigado teniendo un enfoque en computación *Edge*, como ya fue descrito previamente, al no almacenar imágenes en ningún dispositivo.

De la misma forma, existen otras soluciones para *People Counting* en conjunto con LoRa, tanto a partir de ToF¹⁸¹⁹, como de LiDAR²⁰. Pero como ya fue mencionado, este tipo de soluciones no son prácticas para la aplicación que se busca en el campus, debido a la detección en espacios abiertos y el análisis en vivo de ocupación de espacios.

¹⁶Office Occupancy Counter LoRaWAN, IMBuildings, <https://www.imbuildings.com/office-occupancy-counter-lorawan/>

¹⁷LoRaWAN People Counter, IMBuildings, <https://www.imbuildings.com/lorawan-people-counter/>

¹⁸Terabee GDPR compliant People Counting L-XL LoRa Version, ExpTech, <https://exp-tech.de/en/products/terabee-gdpr-compliant-people-counting-l-xl-lora-version>

¹⁹AI ToF People Counting Sensor, Milesight, <https://www.milesight.com/iot/product/lorawan-sensor/vs133>

²⁰LoRaWAN Lidar People Counter, daviteq, <https://filerun.daviteq.com/wl/?id=KRYgdWVbNJqwu9hnIav2yrF5EjmdjXrG>

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

Para resolver el problema planteado, junto con sus efectos asociados, se propone el diseño de un sistema capaz de obtener el nivel de ocupación (*occupancy*) de distintos sectores del campus San Joaquín de la UTFSM. En primer lugar se presentan los requerimientos, tanto funcionales como no funcionales, que debe cumplir el sistema. Luego, se indica el diseño del sistema, y cómo se encuentra organizado. Posteriormente, se describe la selección del *hardware* utilizado para los distintos componentes. Y por último, se detalla el funcionamiento de cada uno de estos.

3.1. Requerimientos

3.1.1. Requerimientos Funcionales

- Detección de nivel de ocupación cíclica: El sistema debe cumplir con ciclos en los cuales se capture la imagen de la sala, se procese y obtenga el nivel de ocupación actual, para luego obtener un promedio cada tres detecciones.
- Envío de localización de sala y nivel de ocupación: Los mensajes enviados deben contener los datos del nivel de ocupación calculados y de la sala en la que se mide.
- Información en plataforma web: Al visualizar el mapa del campus, el sistema debe permitir obtener información de las salas al hacer clic en ellas.

3.1.2. Requerimientos no Funcionales

- Detección de cada nivel de ocupación: No es necesario detectar con alta precisión la cantidad de personas en una sala, pero sí que el resultado corresponda con el nivel de ocupación real. Estos niveles son definidos más adelante en esta sección.
- Tasa de actualización de datos: Los datos de cada sala deben ser actualizados con una frecuencia constante de 10 segundos, con tal de informar los niveles de ocupación más recientes, y que el sistema cuente con la holgura necesaria para procesar y enviar cada imagen.

No se consideran requerimientos relacionados con seguridad de datos, rendimiento y usabilidad, debido a que el alcance del trabajo consta de una prueba de concepto del sistema diseñado.

3.2. Diseño del sistema

Considerando el cumplimiento de los objetivos propuestos, el sistema se compone de tres partes, las que son descritas en las siguientes secciones del Capítulo.

Reconocimiento visual y CC basado en el uso de modelos de *TinyML* en la *Capa Edge*, **Comunicación mediante LoRa** de los resultados obtenidos a un nodo central, o *Gateway*²¹ en la capa del mismo nombre, y una vez recibidos, estos son subidos para permitir la **Visualización en plataforma web** de los resultados para cada área medida, en la capa *Web/Cloud*. En la Figura 13 se presenta el diagrama del diseño del sistema, en el que se indica cada componente y su rol.

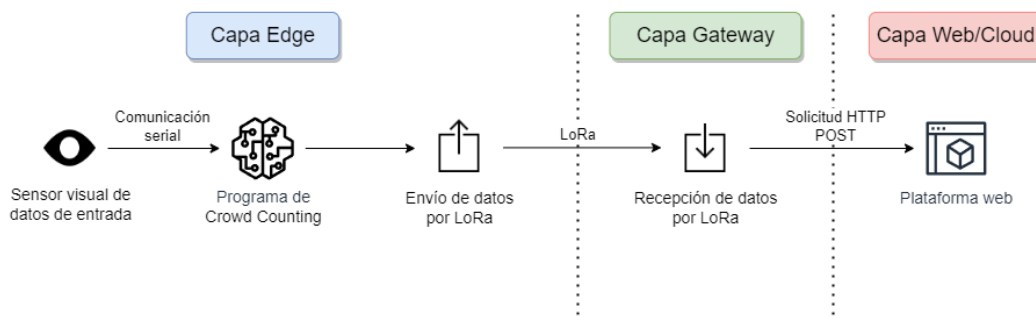


Figura 13: *Diagrama de arquitectura de sistema propuesto.*

Fuente: Elaboración propia.

3.2.1. Capa Edge

Capa ubicada en el extremo de la red, encargada de tres tareas, como se ve en la Figura 13.

- Obtención de datos brutos: Captura de imágenes en cada sala a partir de una cámara.
- Procesamiento de datos mediante modelo de *TinyML*: Los datos obtenidos son procesados por un modelo de *Crowd Counting*, previamente entrenado y desplegado.
- Envío de datos a la siguiente capa: Finalmente, los resultados del nivel de ocupación son enviados a la siguiente capa mediante LoRa, sin enviar datos adicionales, como las imágenes captadas.

Estas tareas pueden ser realizadas por una combinación de dispositivos o por uno solo que cuente con las capacidades necesarias. Lo importante para mantener el paradigma *Edge* es que en esta capa se realice tanto la obtención de datos, como su procesamiento y envío.

²¹*Gateway*: Dispositivo o nodo que conecta redes dispares, al traducir comunicaciones de un protocolo a otro.[Cisco, 2023]

3.2.2. Capa Gateway

Capa encargada tanto de recibir los resultados obtenidos y enviados por la capa de *Edge*, mediante LoRa, como de asegurar que los mensajes en los que vienen incluidos los resultados correspondan a los enviados de la capa anterior y no a externos. Esto se realiza a partir de protocolos definidos más adelante.

Luego, los datos importantes, como son el resultado e identificador del lugar en que se realiza el CC, son extraídos del mensaje y enviados a la siguiente capa, a partir de una solicitud HTTP.

3.2.3. Capa Web/Cloud

Por último, la solicitud HTTP es recibida por el servidor de *Backend* del componente web, lo que gatilla que estos datos sean almacenados en la Base de datos. Esta capa además cuenta con un servidor de *Frontend*, el que permite la visualización de los resultados.

3.3. Selección de Hardware

3.3.1. Capa Edge

Análisis de alternativas

Como se indica en la sección anterior, existen distintas alternativas para esta capa del sistema. En el caso del *Nicla Vision*, cuenta con una memoria reducida, que no permite la implementación de los modelos desarrollados. El *Arduino Portenta X8* y *Jetson Nano* cuentan con demasiada potencia para la tarea requerida, y no cuentan con cámara, por lo que implicaría un gran gasto considerando que no incluye las capacidades requeridas. *Google Coral* tampoco cuenta con cámara integrada, pero tiene un precio asequible, lo que puede hacer que el gasto total sea aceptable. Sin embargo, por otro lado, esta placa no se encuentra disponible en Chile y cuenta con una complejidad considerable de aprendizaje y desarrollo.

Finalmente, ninguna de estas tecnologías cuenta con capacidad de comunicación por LoRa, por lo que habría que incluir otro dispositivo compatible, lo que implica un aprendizaje, esfuerzo y costo adicional. Por lo tanto, se decide utilizar el *Arduino Portenta H7*, junto con la extensión *Portenta Vision Shield*, como nodo de procesamiento de la capa *Edge*, al ser un dispositivo reducido y contar con la mayoría de características deseadas ya integradas, tales como conectividad por LoRa, cámara para realizar la detección de ocupación y compatibilidad con lenguajes de alto nivel, con uno de los menores precios entre las alternativas presentadas.

Portenta H7

Este microprocesador es uno de los modelos más recientes de la línea Pro de Arduino, empresa reconocida por ser líder en el desarrollo de dispositivos reducidos como el *Nano*, *UNO*, *BLE 33 Sense*, además de contar con una comunidad activa debido a su enfoque en el código y desarrollo abierto. La línea Pro se destaca por ofrecer equipos compactos con *hardware* más potente, lo que proporciona una mayor movilidad para realizar tareas complejas como *Computer Vision* y reconocimiento de audio, entre otros. Ejemplos de estos dispositivos incluyen *Portenta X8* y *Nicla Vision*.

El Portenta H7, en particular, está equipado con una memoria Flash de 1MB y SDRAM de 8MB, además de un procesador ARM Cortex-M7 de 480 MHz y un ARM Cortex-M4 de 240 MHz, ambos basados en la arquitectura STM32, los que permiten la ejecución de tareas de forma paralela, lo que lo convierte en una buena alternativa para su aplicación en redes de *Edge computing*.

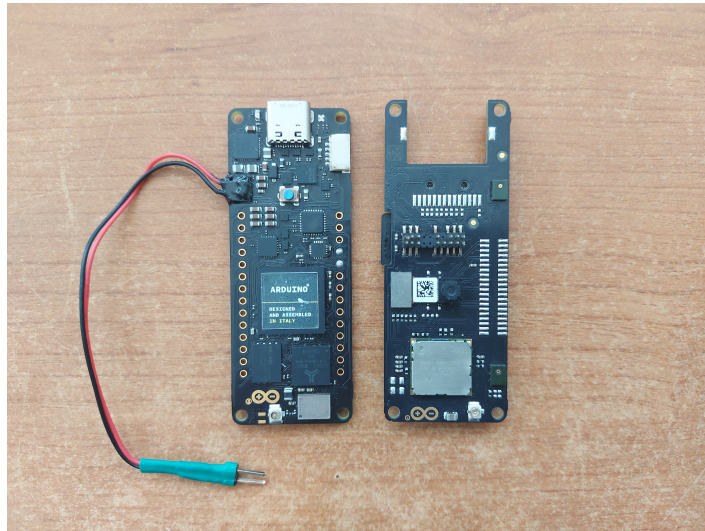


Figura 14: Arduino Portenta H7 (izquierda), con la extensión *Vision Shield* (derecha).

Fuente: Elaboración propia.

En cuanto al dispositivo encargado del envío de datos por LoRa, además de contar con capacidad de reconocimiento visual, se utiliza la extensión *Portenta Vision Shield*, la que incorpora una cámara de 320x240 píxeles en escala de grises, la que permite la ejecución de programas de *Computer Vision* en su procesador independiente, basado en arquitectura *STM32L0*.

Este equipo permite la ejecución de código tanto en *MicroPython*, como en Arduino (lenguaje basado en C/C++), sin embargo, no se puede mezclar su uso en los distintos procesadores mencionados, por lo menos en caso de ser necesario que se comuniquen entre sí. Esto, sumado al hecho que LoRa PHY no sea soportado por *MicroPython*, hace imperativo el uso íntegro del lenguaje Arduino, en conjunto con Arduino IDE como entorno de desarrollo.

Para permitir el uso del Portenta de manera independiente, se adapta e instala una batería recargable *Li-Po* de 3.7 voltios y 500 mAh, lo que proveería una vida útil de unas dos o tres horas. En caso de mantenerlo en funcionamiento durante todo el día, sería necesario mantenerlo alimentado mediante un cable USB, ya sea a un computador o directamente a la corriente. Además, de una carcasa y un soporte para protegerlo de impactos y poder obtener un mejor ángulo de visión (Ver Figura 15). La carcasa es obtenida de un repositorio público de modelos 3D²², mientras que el soporte es diseñado en la plataforma *Tinkercad*²³. Estos modelos son impresos en 3D en las impresoras del FabLab (Laboratorio de Fabricación) de la Universidad.



Figura 15: Portenta H7 con soportes 3D y antena de transmisión LoRa.
Fuente: Elaboración propia.

3.3.2. Capa Gateway

Análisis de alternativas

Los dispositivos a cumplir con las tareas descritas para esta capa deben contar con la capacidad de ejecutar los programas desarrollados. Por lo que, se podrían tomar en cuenta en principio algunas de las alternativas consideradas para la capa *Edge*, como el *Arduino Portenta X8* y *Jetson Nano*, los que tenían demasiada potencia para ser usados en esa capa. Sin embargo, sus elevados precios nuevamente hacen poner en duda su aplicación para este sistema.

²²Arduino Portenta H7 VS Case, Angel de la Rosa, <https://grabcad.com/library/arduino-portenta-h7-vs-case-1>

²³Soporte Portenta H7 esquina, desarrollo propio, <https://www.tinkercad.com/things/kdj0Fes7uAB>

Raspberry Pi²⁴, por otro lado, corresponde a una gama de computadores de tamaño reducido y precio accesible (\$58.900 según su página oficial de Chile²⁵), diseñados para ejecutar todo tipo de procesos livianos, tanto para uso convencional como industrial. Por lo tanto, estos modelos representan una solución ideal, debido a sus capacidades y precios asequibles. Una alternativa a considerar sería el Raspberry Pi 4, por ejemplo, ya que cuenta con una gran potencia en un espacio reducido. Sin embargo, para ahorrar en consumo de energía y costo, se podrían utilizar alternativas como el Raspberry Pi 3, puesto no se requiere de una capacidad tan alta para ejecutar los programas correspondientes a esta capa del sistema.

También es posible utilizar hardware de baja potencia, mediante la distribución de tareas entre distintos dispositivos conectados entre sí, por ejemplo, utilizando microprocesadores de tipo ESP32²⁶, capaces de ejecutar los programas de recepción de LoRa y los servidores de la plataforma web.

Sin embargo, debido a la disponibilidad de *hardware* en el Laboratorio de Redes del campus, se decide utilizar un Raspberry Pi modelo 3B+, que también cuenta con los requisitos mínimos para su implementación en el sistema.

Raspberry Pi 3B+

Entre los distintos modelos disponibles, se utiliza el popular *Raspberry Pi 3 Model B+*, el que incluye un procesador de 4 núcleos 64-bit de 1.4GHz, acceso a *Ethernet* y *WLAN* de banda dual, *Bluetooth* 4.2/BLE y 40 pins para alimentación y comunicación mediante señales digitales (GPIO). Estos pins pueden ser utilizado para controlar una variedad de sensores y comunicarse con otros dispositivos.

Para cumplir su función como nodo centralizado de red *Edge*, se emplea la extensión *Dragino LoRa/GPS Hat*, que permite la recepción de señales de 915 MHz, frecuencia oficial en Chile para el uso de LoRa. Esta extensión utiliza una antena omnidireccional que facilita el envío de datos desde puntos ubicados a distintas direcciones. Sin embargo, dado que se trata de un dispositivo de canal dual en lugar de múltiple, no es posible aplicar el protocolo de alto nivel LoRaWAN, limitando así al sistema al uso de LoRa PHY, el que es adaptado para el funcionamiento particular del proyecto.

²⁴Raspberry Pi, <https://www.raspberrypi.cl>

²⁵Raspberry Pi 3B+, <https://raspberrypi.cl/producto/raspberry-pi-3-modelo-b/>

²⁶ESP32, Expressif, <https://www.espressif.com/en/products/socs/esp32>

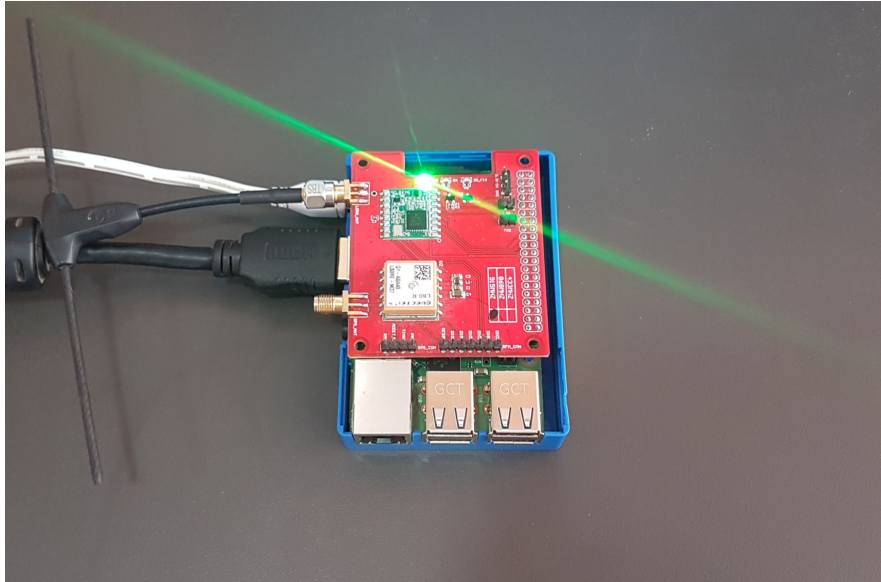


Figura 16: *Raspberry Pi 3B+* con extensión *Dragino*.
Fuente: Elaboración propia.

3.4. Reconocimiento visual y *Crowd Counting*

3.4.1. Diseño de CNN

La base del modelo de redes neuronales convolucionales es obtenida en la plataforma *Edge Impulse*, servicio que permite el desarrollo de distintos tipos de modelos de *Machine Learning*, para lo que se proporciona una cadena de procesos automatizada que permite la configuración de hiper-parámetros y la subida de los datos de entrenamiento y prueba, con sus etiquetas correspondientes.

El modelo utilizado por *Edge Impulse* corresponde a *FOMO* (*Faster Objects, More Objects* (Objetos más rápidos, Objetos más numerosos, en inglés)²⁷, basado a su vez en el modelo pre-entrenado *MobileNetV2* [Sandler et al., 2019], utilizado para reconocimiento visual de distintos tipos de objetos. Sus capas ya entrenadas son aprovechadas para conseguir un mejor rendimiento, utilizando solo una fracción de estas. Además, se aplican una serie de optimizaciones que logran disminuir la cantidad de parámetros del modelo y, por consiguiente, la potencia de procesamiento y espacio necesarios para ejecutarlo. Sin embargo, debido a la limitada capacidad de memoria SDRAM del Portenta, el modelo es adaptado para recibir imágenes de entrada de 96x96 píxeles, por lo que los datos de entrenamiento y los utilizados al probar el sistema deben ser transformados para cumplir con esta restricción.

²⁷Announcing *FOMO* (*Faster Objects, More Objects*), *Edge Impulse*, <https://www.edgeimpulse.com/blog/announcing-fomo-faster-objects-more-objects>

En cuanto al proceso de entrenamiento, se utiliza el *dataset JHU-CROWD++*, el que cuenta con fotos con la presencia de una cantidad variable de personas en distintas situaciones, junto con archivos con las etiquetas correspondientes a las coordenadas de las cabezas de cada persona. Las imágenes pueden alcanzar los miles de individuos en algunos casos, sin embargo, solo se consideran casos de hasta 80 personas, debido al alcance del sistema y su aplicación en lugares reducidos, además de tener en cuenta los recursos limitados del Portenta.

Por otro lado, también se utilizan conjuntos de imágenes obtenidas en las zonas específicas en las que se prueba el sistema, con tal de adaptar los modelos de manera *ad-hoc* a cada una, esperando obtener mejores resultados al no tener la necesidad de detectar una mayor variedad de escenarios. Estas fotografías son etiquetadas con la herramienta *VGG Image Anotator* (VIA²⁸), la que permite exportar archivos de formato .csv con las coordenadas de las etiquetas, correspondientes a la cabeza de cada persona, como se ve en la Figura 17.

Finalmente, para acceder y desplegar el modelo de CNN entrenado, se descarga el módulo específico para Arduino IDE, el que es subido al Portenta H7 para ser usado por su procesador principal.

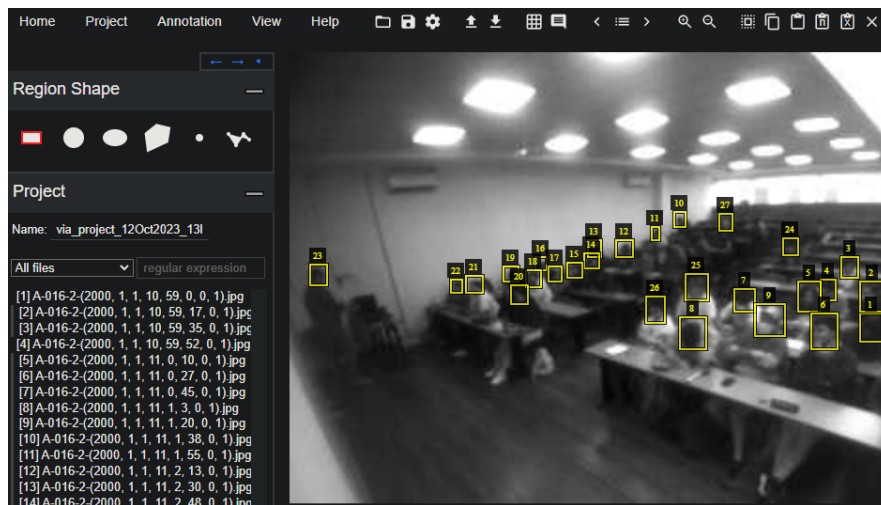


Figura 17: Herramienta VIA, utilizada para etiquetar imágenes en campus.

Fuente: VGG Image Anotator

3.4.2. Crowd Counting

Una vez obtenido el modelo de CC y subido al Portenta, este es montado en una ubicación específica y procede a capturar imágenes del lugar, las que son procesadas por el modelo de CNN, que retorna como resultado el número de personas detectadas por este, correspondiente a la ocupación del lugar. Las imágenes son obtenidas y procesadas continuamente en

²⁸VIA, <https://www.robots.ox.ac.uk/~vgg/software/via/via.html>

intervalos de 10 segundos, con tal de no sobrecargar el microprocesador, y se obtiene el promedio de ocupación cada 3 detecciones, el que es entonces enviado junto con el identificador de sala por LoRa. Con este plazo entre mensajes se intenta evitar posibles interferencias entre sus propios mensajes emitidos.

Con tal de asegurar un mayor rendimiento, considerando el tamaño y cantidad de parámetros del modelo de CNN obtenido en *Edge Impulse*, se genera una instancia del modelo para cada escenario, a partir de conjuntos de datos de entrenamiento obtenidos en cada uno de ellos. De esta manera, los modelos no requieren contar con una alta capacidad de inferencia.

3.5. Comunicación LoRa

Luego de obtener un resultado, este es enviado mediante LoRa desde el Portenta H7 al Raspberry Pi, que actúa como *Gateway*. Esto gracias al uso de una antena compatible con la frecuencia de comunicación de LoRa (915 MHz), tanto en el Portenta, como en la extensión *Dragino GPS/LoRa Hat* para el Raspberry Pi.

Con tal de asegurar que los datos recibidos por el *Gateway* correspondan a los emitidos por el Portenta, se utiliza un formato para el inicio y final de cada mensaje, definidos como los caracteres `<stL>` (*start LoRa*) y `<eL>` (*end LoRa*), respectivamente. El formato del mensaje, entonces, resulta en un segmento de inicio, tres dígitos para el identificador de la sala, correspondiente al valor de su ID en la base de datos de la plataforma web, tres dígitos para indicar la cantidad de personas presentes en esa sala, y por último, el segmento de final de mensaje.



Figura 18: Formato de mensajes enviados por LoRa.
Fuente: Elaboración propia.

Cada carácter del mensaje equivale a 1 *byte*, por lo que se utilizan en total 15 *bytes*. Esta invariabilidad en la longitud de los mensajes permite a la vez ahorrar espacio en su contenido, al no requerir especificar este valor dentro del formato.

Debido a la incapacidad del Raspberry Pi de actuar como *Gateway* compatible con el protocolo LoRaWAN, se decide seguir con una metodología *peer-to-peer* (P2P), implementada a partir del proyecto *LoRa Transceiver* de Mark Ruijter, existente en *Github* ²⁹. Este módulo permite configurar los distintos parámetros de LoRa, los que deben ser equivalentes tanto en los emisores como receptores.

Por el lado del Portenta, se utiliza como base un código de ejemplo³⁰ del repositorio “*portenta-pro-community-solutions*”, de Jeremy Ellis. Este es subido para ser usado en el procesador de la extensión Vision Shield, el que es ejecutado de forma paralela al programa de CC. Una vez este obtiene un resultado a partir del modelo de CNN, es adaptado al formato de mensajes ya definido y enviado mediante comunicación serial al programa ejecutado en el Vision Shield. Luego de recibirlo, este se encarga de emitir el mensaje mediante LoRa.

3.5.1. Arduino IDE y librerías

Como se describió previamente, el programa a ser ejecutado en el nodo *Edge* debe ser programado en Arduino nativo y C++, debido a la incompatibilidad del sistema con LoRaWAN. Por lo tanto, se utiliza Arduino IDE 1.8.19 como entorno de desarrollo para los módulos de *Crowd Counting* y comunicación por LoRa en el Portenta H7. El IDE debe ser ejecutado en una máquina virtual de Linux³¹ para ser compilado y subido exitosamente.

Para esto se utiliza un conjunto de librerías y dependencias para asegurar su funcionamiento en el Portenta H7, de las cuales se describen las fundamentales a continuación.

- **Arduino Mbed OS Portenta:** Paquete base (*board package*) compatible con el Portenta H7, utilizado para la ejecución del programa principal de CC en el núcleo principal de la placa (STM32H7 Cortex-M7).
- **STM32L0-core**³²: Paquete de placa (*board package*) utilizado para la ejecución del módulo de comunicación por LoRa en el núcleo del Vision Shield (STM32L0), mediante el uso del controlador sx1276 de Semtech, instalado también en esta extensión.
- **TensorFlow Lite**³³: Librería de *Tensorflow Lite* adaptada para funcionar con el Portenta H7, dado que los parámetros para las entradas periféricas se encontraban definidas exclusivamente para el Arduino Nano 33 BLE.

²⁹*lora_transceiver*, Mark Ruijter, https://github.com/mruijter/lora_transceiver

³⁰“dot3433-murata-lora-grumpyoldpizza”, Jeremy Ellis, <https://github.com/hpssjellis/portenta-pro-community-solutions>

³¹Se utiliza *Ubuntu 20.04* en *VMWare Workstation 17 Player*, en un computador con *Windows 10* como *host*.

³²*ArduinoCore-stm32l0*, Jeremy Ellis, basado en repositorio de GrumpyOldPizza, https://github.com/hpssjellis/ArduinoCore-stm32l0/tree/vision_shield

³³*tf-lite-micro-arduino-portenta-h7*, José Quezada, *fork* de tensorflow, <https://github.com/josemqz/tf-lite-micro-arduino-portenta-h7>

- **SDRAM:** Esta librería es usada con el fin de asignar la memoria para los tensores del modelo de CNN a ser utilizado, al igual que las imágenes capturadas por la cámara.
- **Camera y HM10B0:** Librerías necesarias para controlar el sensor visual, correspondiente al HM10B0 del Vision Shield.

3.6. Visualización en plataforma web

Finalmente, para almacenar persistentemente los datos recibidos por el Raspberry Pi y permitir su visualización, se implementa una plataforma web simple en el mismo dispositivo, compuesta por una base de datos relacional, un *Backend* y *Frontend*.

3.6.1. Backend y Base de datos

Para construir el *Backend*, se utiliza la librería *Flask* mediante *Python*, mientras que, como gestor de base de datos, se escoge *SQLite3*. Tanto este como *Flask* son escogidos debido a su liviandad y facilidad de uso, además de contar con integración completa entre ellas.

Se diseña un modelo *ad-hoc* al sistema, como el visible en la Figura 19.

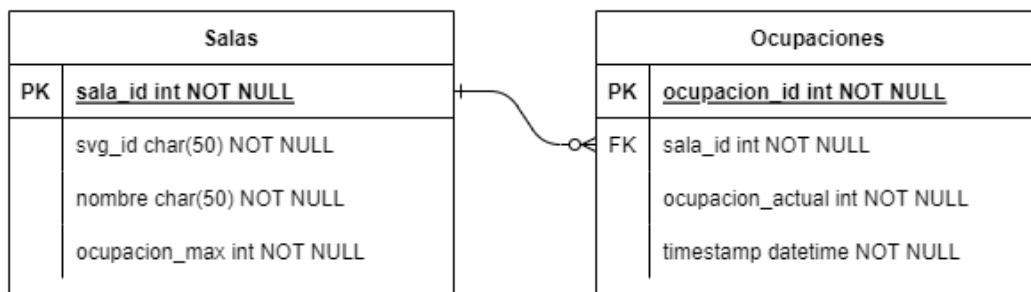


Figura 19: Modelo de Bases de Datos.

Fuente: Elaboración propia.

Esta es creada desde el *Backend* mediante *SQLAlchemy*, módulo que provee las herramientas necesarias para gestionar la base de datos a partir de comandos simples de *Python*.

A continuación, se enlistan los *endpoints*³⁴, a los que se debe hacer referencia para crear, modificar o eliminar entradas de la Base de Datos.

³⁴Parte de la URL del *backend* que permite realizar una función específica. En este caso, al realizar una petición HTTP a un endpoint determinado, el *backend* procesa la solicitud y devuelve una respuesta correspondiente.

Sala:

- crear()
- get_salas()
- get_sala_by_id(id: int)
- get_sala_by_nombre(nombre: str)

Ocupación:

- crear()
- get_ocupaciones()
- get_ultimas_ocupaciones()
- get_ocupaciones_by_sala_id(sala_id: int)
- get_ultima_ocupacion_by_sala_id(sala_id: int)
- delete_ocupacion_by_id(id: int)

3.6.2. Frontend

Mientras que el *Frontend* es desarrollado a partir del *framework* de Javascript *Vue*, el que facilita el proceso de confección e integración de los distintos componentes.

La interfaz del sitio web se compone de un mapa en formato SVG (Gráfico vectorial escalable, en español) de las distintas zonas y salas Campus San Joaquín, como se ve en la Figura 20.

Para recuperar los datos de la base de datos, se emplea la biblioteca *axios*, que facilita la realización de las solicitudes HTTP necesarias para comunicarse con el *Backend* y obtener la información requerida. Esto se realiza cada 10 segundos, con tal de mantener actualizado el mapa de ocupaciones en todo momento.

Mapa | [Acerca de](#)

Ocupación de campus San Joaquín

Hacer clic en cada área para ver detalles.

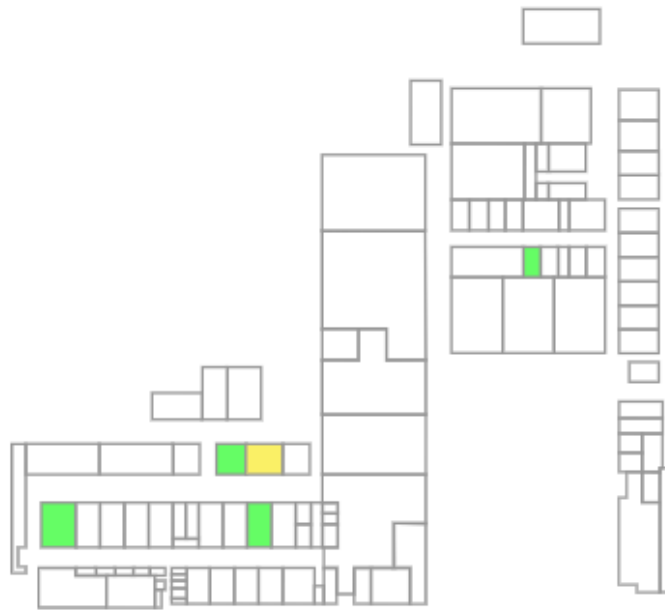


Figura 20: *Frontend* de plataforma web con niveles de ocupación de un subconjunto de salas del Campus San Joaquín.

Fuente: Elaboración propia.

De esta forma, se obtienen los valores de ocupación más recientes de cada sector, los que son categorizados a partir del valor de ocupación máxima de cada zona, en uno de los siguientes 4 niveles de ocupación.

- 0 % - 33 %: Ocupación baja.
- 33 % - 66 %: Ocupación media.
- 66 % - 100 %: Ocupación alta.
- =>100 %: Ocupación sobrepasada.

Estos niveles son, entonces, presentados en la interfaz gráfica a partir de los colores representativos de cada uno.

Es necesario recordar que no se muestran imágenes de la cámara ni la cantidad exacta de personas, tanto por asegurar la privacidad de los estudiantes y los datos, como para permitir un cierto grado de flexibilidad en cuanto a la precisión del modelo.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

Para verificar la efectividad del sistema implementado, se prueba instalar cada componente en distintas zonas del campus San Joaquín, teniendo en consideración que se trata de un producto mínimo viable, que pueda servir como base para una potencial implementación real a futuro.

Los objetivos de los experimentos a realizar consisten, en primer lugar, en la verificación de la comunicación por LoRa y de su calidad, con tal de conocer su factibilidad de aplicación en el campus, al igual que su distancia efectiva de uso, además de analizar cómo le afectan los obstáculos físicos (edificios, paredes y techos de distintos materiales). Luego, es necesario validar la precisión del modelo de CC, tanto con respecto a los niveles de ocupación definidos para el *Frontend* como en términos absolutos, en conjunto con la comunicación mediante LoRa. En estos dos casos, se miden distintas variables, definidas en cada subsección, las que puedan permitir obtener conclusiones claras con respecto a estos objetivos planteados.

La plataforma web, por otro lado, es considerada exclusivamente una herramienta para la visualización de los resultados del sistema, por lo que no es puesta a prueba en sí, ni su interfaz de usuario, ni el uso eficiente y efectivo de la base de datos y servidores.

4.1. Pruebas preliminares de LoRa

Para la primera serie de experimentos, correspondientes a la medición de efectividad de comunicación por LoRa, se consideran las distancias entre el nodo móvil y el receptor, teniendo en cuenta la altura a la que se encuentra cada uno de los nodos, con el de la capa *Edge* ubicado aproximadamente a 2 metros del suelo, y el *Gateway* con alturas variables según el experimento (Experimento 1: 1 metro, Experimento 2: 12 metros, Experimento 3: 3 metros). Estas distancias se encuentran en las tablas en la sección de Anexos, junto con las variables definidas a continuación.

- **Pérdida de paquetes:** Porcentaje de paquetes no recibidos por el *Gateway* luego de ser transmitidos por el nodo emisor.
- **RSSI:** Correspondiente al Indicador de fuerza de señal recibida (RSSI por sus siglas en inglés), el que mide la fuerza de la señal de los paquetes recibidos por el *Gateway*. Se suele usar para encontrar la distancia entre el transmisor y receptor, ya que la fuerza disminuye a medida que se alejan entre sí. [Sadowski y Spachos, 2018]
- **RSSI de paquete:** Valor promedio de los RSSI medidos durante la recepción del paquete mediante LoRa. Al contrario del RSSI de registro (rRSSI), que es el RSSI instantáneo en

su recepción. [Yang *et al.*, 2022]

- **SNR:** Como indican sus siglas en inglés (*Signal-Noise Ratio*), esta se refiere a la tasa entre la potencia de la señal de salida deseada y el ruido de fondo, lo que es descrito como $SNR\ dB = 2 \log_{10} \frac{V_{signal}}{V_{noise}}$, donde V_{signal} y V_{noise} son el voltaje de señal medido y el voltaje del ruido, respectivamente. [Dinh *et al.*, 2023]. Consiste en una escala logarítmica, lo que significa que si el valor es 0, las potencias son iguales, si el ruido es mayor, el valor es negativo, y viceversa si es positivo.

Los valores de los parámetros de LoRa utilizados en las tres pruebas corresponden a los presentados en la Tabla 2.

Variable	Valor
<i>Spreading Factor</i>	7
Potencia	14 [dBm]
Ancho de banda	125 [MHz]
Tamaño de bloques	32 [bit]

Tabla 2: Valores de parámetros de comunicación por LoRa.
Fuente: Elaboración propia.

4.1.1. Prueba 1: Dispositivos ubicados en extremos opuestos

En primer lugar, se ubican los dispositivos en puntos separados por una distancia extrema dentro del campus. El Raspberry Pi (*Gateway*) es instalado en la sala B-010, en la esquina norponiente, mientras el Portenta H7 (nodo móvil o *Edge*) es desplazado a través del campus, desde ese punto hasta llegar a la esquina sur oriente, donde se encuentra la entrada principal. Durante este trayecto, el dispositivo envía mensajes de prueba cada 10 segundos a medida que se mueve hasta su destino, los que pueden ser visualizados en el *Gateway*.

Los resultados obtenidos mediante una consola del Raspberry Pi, son registrados e indicados en la Tabla 11, presente en el anexo, la que además incluye las distancias aproximadas entre los nodos. Los valores promedio, por su parte, se presentan en la Tabla 3.

Pérdida de paquetes	RSSI	RSSIpaquete	SNR
72 %	-110	-106	1

Tabla 3: Valores promedio de resultados de prueba 1 de LoRa.

A partir de esto se analiza una de las métricas más populares en este ámbito, el RSSI, con respecto a la distancia entre nodos en la Figura 22, mientras que los gráficos de las demás métricas con respecto a las distancia (Figuras 42 y 41) se encuentran en el Anexo. Como se



Figura 21: Ubicaciones de *Gateway* (azul) y nodo (verde, para recepción exitosa y rojo, en caso contrario) al transmitir mensajes de prueba cada 10 segundos en primer experimento.

Fuente: Elaboración propia.

mencionó previamente, los valores del RSSI se expresan en números negativos, indicando una mejor calidad de señal a medida que se acercan a cero.

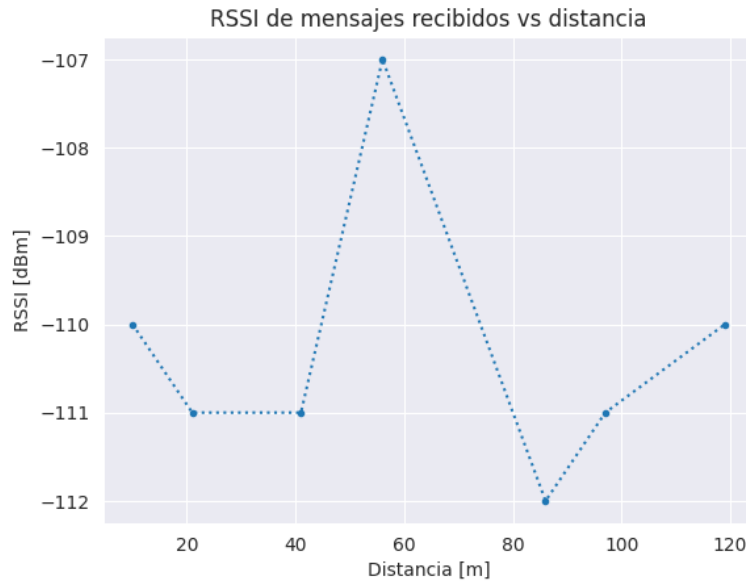


Figura 22: RSSI de paquetes recibidos con respecto a la distancia del nodo móvil al Gateway en prueba 1 de LoRa.
Fuente: Elaboración propia.

Al observar el gráfico, no parece existir una relación clara entre estas variables en este experimento, lo que sugiere la presencia de factores adicionales, como la cantidad de obstáculos entre los nodos. Esto puede ser visualizado en la Figura 21, en donde tan pronto se ingresa a un pasillo cerrado (mensajes 3 y 4), se dejan de recibir mensajes, y al encontrarse en un lugar abierto (mensajes 9, 10 y 12), se retoma la comunicación. Esto podría explicar la ausencia de recepción a partir del mensaje número 13, enviado a 134 metros del Gateway.

De todas formas, es necesario mencionar que los valores presentados son calculados solamente a partir de los mensajes que fueron efectivamente recibidos, por lo que además se considera la relación entre la recepción efectiva y la distancia entre los nodos. En la Figura 23, correspondiente a un gráfico de violín, se indica la forma en que se distribuyen los mensajes recibidos (azul) y los no recibidos (naranja), con respecto a la distancia entre el nodo móvil y el Gateway.

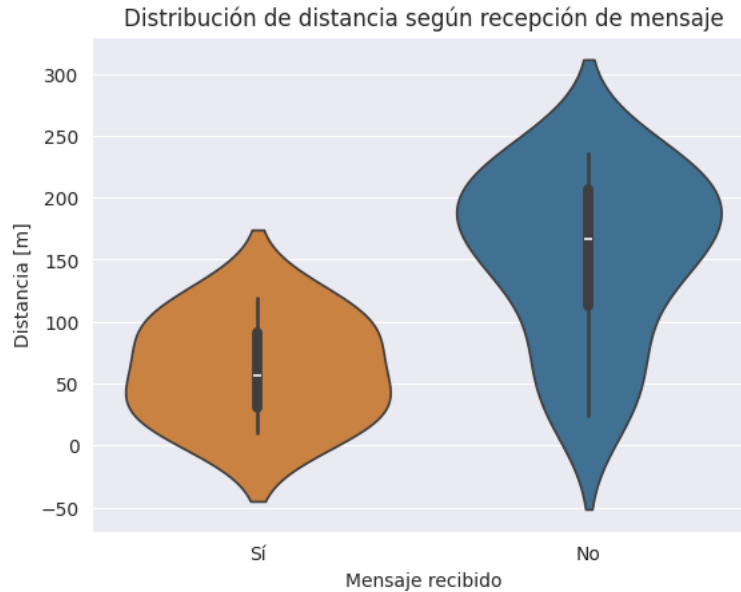


Figura 23: Gráfico de violín indicando la distribución de mensajes recibidos y no recibidos con respecto a su distancia al Gateway en prueba 1 de LoRa.

Fuente: Elaboración propia.

Como se observa, la mayor frecuencia de mensajes efectivamente recibidos se encuentra con aproximadamente 30 metros de distancia entre los nodos, mientras que la mayor cantidad de mensajes no recibidos se da pasado los 150 metros. Sin embargo, lo que destaca de esta última variable es su mayor dispersión a través de su eje, lo que podría indicar que, al contrario de los mensajes recibidos, los no recibidos pueden ocurrir en cualquier distancia, incluso menor a 100 metros.

4.1.2. Prueba 2: Gateway ubicado en centro (Cuarto piso)

Se considera una situación más realista, al ubicar el Raspberry Pi en el cuarto piso del edificio F, a unos 12 metros de altura, con tal de recibir de mejor manera los mensajes desde cualquier área del campus, y probar al mismo tiempo la efectividad de localizar el nodo receptor en una zona elevada.

El Gateway es instalado en la sala F410, mientras el nodo es desplazado a través de todo el campus, trazando una ruta de lugares de interés para transmitir datos alrededor del nodo receptor, como se puede ver en la Figura 24. Durante este recorrido, el Portenta envía mensajes cada 10 segundos desde cada uno de los puntos indicados, tal como en el caso anterior.

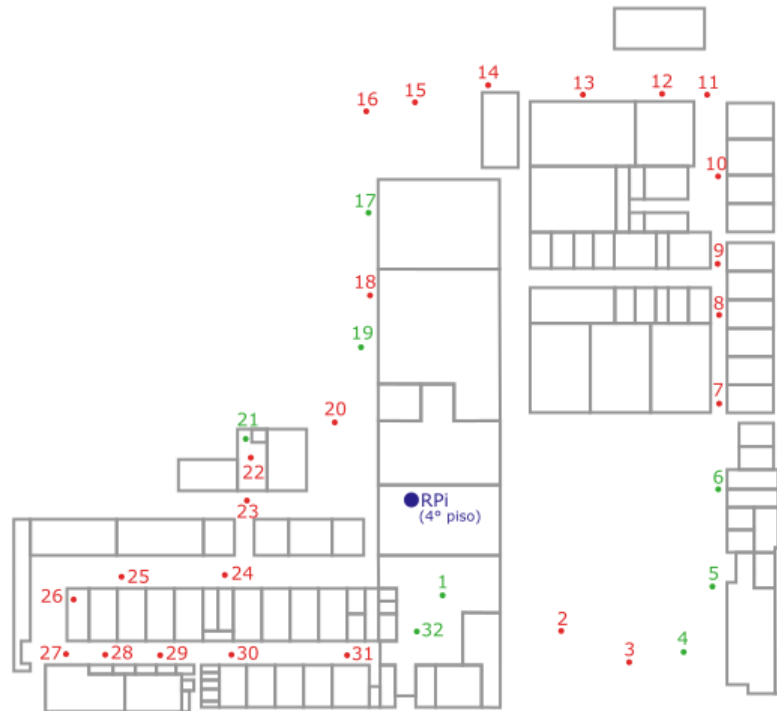


Figura 24: Ubicaciones de *Gateway* (azul) y nodo móvil (verde para recepción efectiva y rojo en caso contrario) al transmitir mensajes de prueba cada 10 segundos en segundo experimento.

Fuente: Elaboración propia.

Los resultados, junto con las distancias entre los nodos son presentados en la Tabla 12, en el Anexo, y los promedios de los demás parámetros en la Tabla 4 a continuación. En este caso destaca que la pérdida de paquetes fue similar e incluso mayor que en el experimento anterior.

Pérdida de paquetes	RSSI	RSSIpaquete	SNR
75 %	-104	-116	0

Tabla 4: Valores promedio de resultados de prueba 2 de LoRa.

En este caso, con respecto a la relación entre RSSI y la distancia de los mensajes recibidos (Ver Figura 25) sí se aprecia una correlación más evidente entre las dos variables, a pesar de la presencia de dos *outliers* cerca de los 80 metros de distancia.

Aunque los valores oscilen entre -105 y -102 dBm, rango superior al de la prueba anterior, que variaba entre los -107 a -112 [dBm], al comparar los promedios de RSSI de paquete y el SNR, se encuentran peores resultados en esta instancia. Esto, junto con la mayor pérdida de paquetes y las distancias reducidas entre los nodos, en comparación con el primer

experimento, sugiere que los obstáculos adicionales, específicamente los pisos intermedios entre el *Gateway* y cualquier punto del primer piso, son responsables de estos resultados negativos.

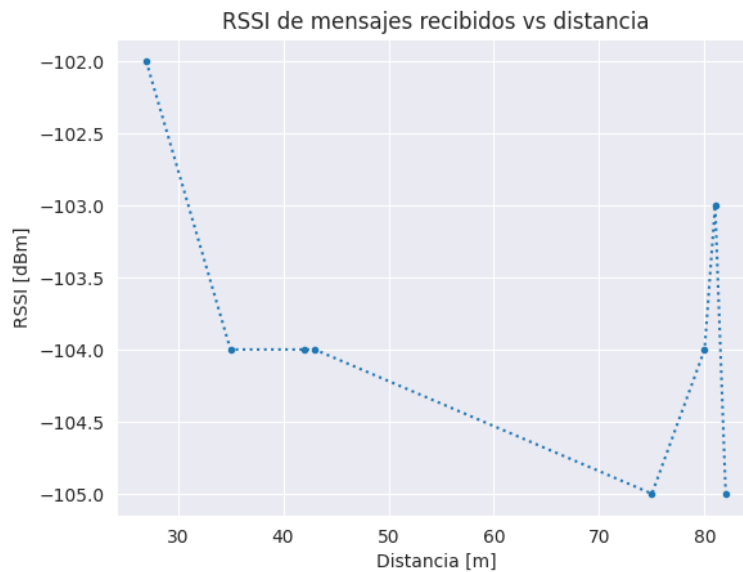


Figura 25: RSSI de paquetes recibidos con respecto a la distancia del nodo móvil al *Gateway* en prueba 2 de LoRa.

Fuente: Elaboración propia.

Por otro lado, la distribución de los mensajes recibidos y no recibidos con respecto a la distancia del nodo móvil al *Gateway* (Ver Figura 26) permanece semejante a la primera prueba. La mayoría de mensajes no recibidos se ubican sobre los 75 metros, mientras que en este caso se identifican dos cumbres en los mensajes recibidos, o sea, una distribución bimodal. Este patrón podría indicar la existencia de dos zonas en las que la recepción de datos es de buena calidad, sin embargo, al analizar estos puntos (Mensajes 4, 5, 6 y mensajes 19, 21) tanto en la Figura 24, como en la Tabla 12, se deduce que solo la primera agrupación parece corresponder a una zona válida, dado que la segunda tiene puntos alrededor y entremedio desde los cuales no se logra una recepción exitosa.

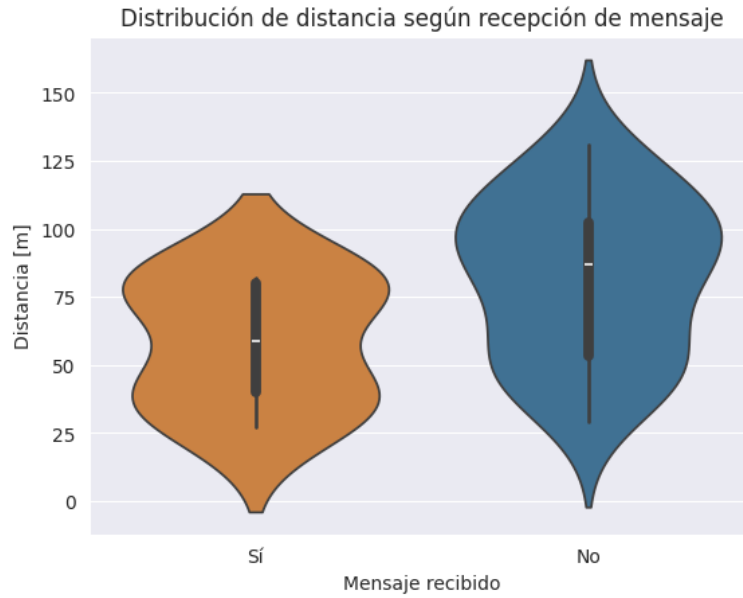


Figura 26: Gráfico de violín indicando la distribución de mensajes recibidos y no recibidos con respecto a su distancia al Gateway en prueba 2 de LoRa
Fuente: Elaboración propia.

4.1.3. Prueba 3: Gateway ubicado en centro (Segundo piso)

Al no obtener resultados satisfactorios en la prueba 2, se estudia la variación en los resultados al realizar la última prueba con el Gateway posicionado a menor altura (Ver Figura 27).

Los resultados son presentados en la Tabla 13, en el Anexo, y los promedios en la Tabla 5 a continuación.

Pérdida de paquetes	RSSI	RSSIpaquete	SNR
43 %	-109	-110	5

Tabla 5: Valores promedio de resultados de prueba 3 de LoRa.

En este caso, se cuenta con un menor porcentaje de pérdida de paquetes en comparación a los demás experimentos, además de un mayor valor de SNR, el que indica que, en promedio, la intensidad del ruido de fondo es menor que la de los mensajes transmitidos.

El RSSI de paquete es mejor que en el experimento anterior, y peor que el primero, lo que se explica al considerar que hay una mayor cantidad de mensajes recibidos y, por ende, mayor variación en la calidad de estos.

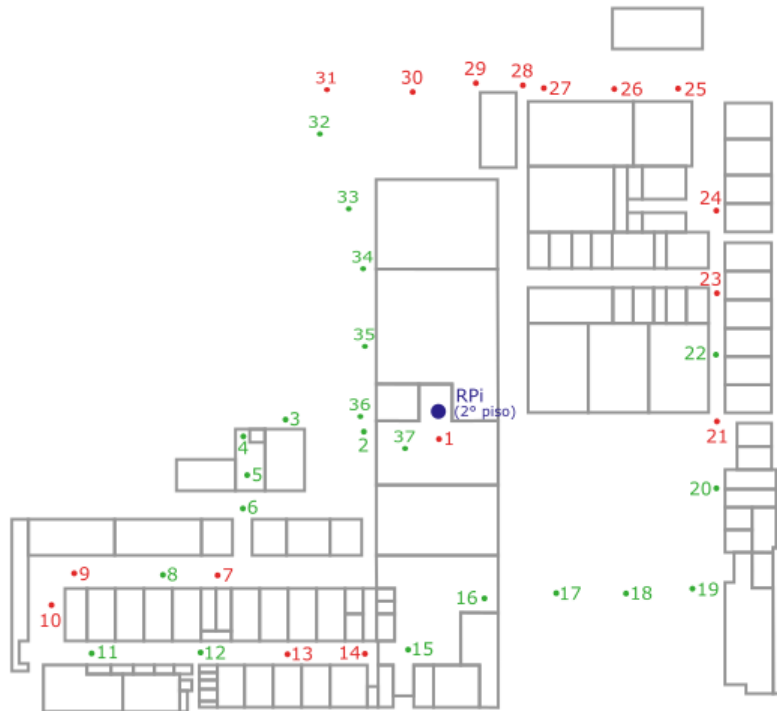


Figura 27: Ubicaciones de *Gateway* (azul) y nodo móvil (verde y rojo) al transmitir mensajes de prueba cada 10 segundos en tercer experimento.

Fuente: Elaboración propia.

En cuanto a la relación entre RSSI y la distancia entre nodos de los mensajes recibidos correctamente, se observa que el mensaje de mayor distancia, correspondiente al *outlier* del gráfico 28, se trata del mensaje 13, enviado a aproximadamente 111 metros del *Gateway*. Al ubicar ese punto en la Figura 27, se identifica como un pasillo cerrado, con una gran cantidad de salas entre ese punto y el nodo receptor, lo que reafirma la idea de que los obstáculos constituyen un factor fundamental en la recepción de los mensajes.

Finalmente, en la distribución de los mensajes recibidos y no recibidos (Ver Figura 29) se destaca una clara predominancia del primer grupo por sobre el segundo en general, extendiéndose por el eje y centrándose en los 60 metros de distancia. Mientras que la mediana de los mensajes no recibidos se encuentra por encima de la de la primera variable, como es esperable, ubicándose alrededor de los 85 metros.

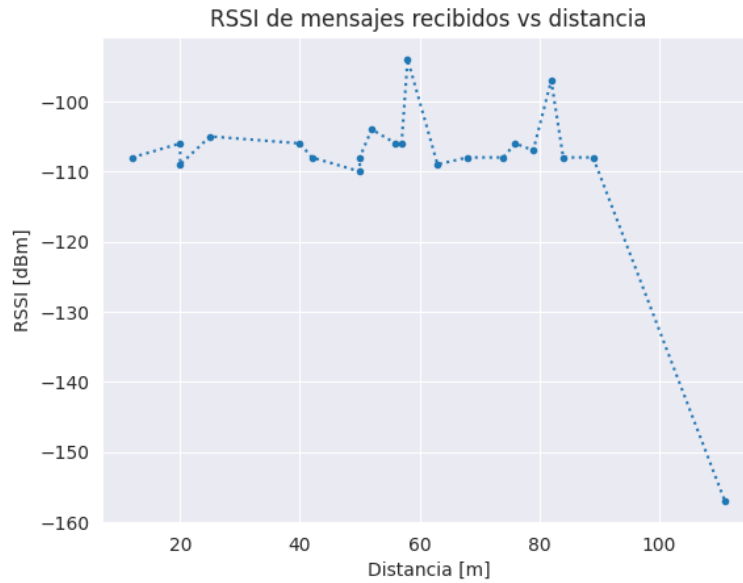


Figura 28: RSSI de paquetes recibidos con respecto a la distancia del nodo móvil al Gateway en prueba 3 de LoRa.
Fuente: Elaboración propia.

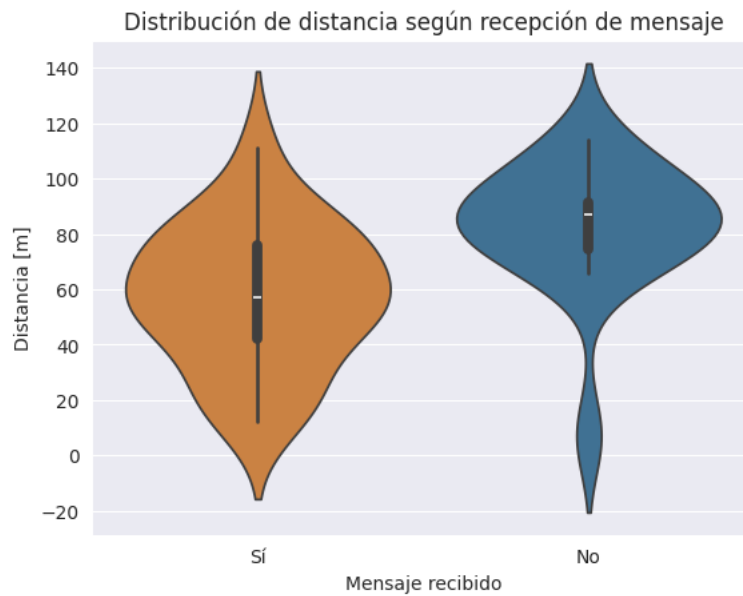


Figura 29: Gráfico de violín indicando la distribución de mensajes recibidos y no recibidos con respecto a su distancia al Gateway en prueba 3 de LoRa
Fuente: Elaboración propia.

4.1.4. Promedio de pruebas de LoRa

Los valores promedio de los resultados de los mensajes recibidos de las tres pruebas se presentan en la Tabla 6. Estos son utilizados en la siguiente sección como referencia al analizar el componente de comunicación entre nodos del sistema.

Pérdida de paquetes	RSSI	RSSIpaquete	SNR
63 %	-108	-111	2

Tabla 6: Valores promedio de resultados de las tres pruebas de LoRa.

Al considerar los resultados obtenidos en las pruebas, como se sugiere previamente, la distancia no es el único factor determinante en la calidad de la comunicación, sino que también influye la cantidad de obstáculos ubicados entre el *Gateway* y el nodo móvil, especialmente teniendo en cuenta que el *Gateway* se encontró en todo momento en áreas cerradas. Sin embargo, la medición de esta variable no resulta sencilla, ya que el mapa del campus abarca tan solo el primer piso, y no incluye los superiores, ni los obstáculos adicionales que estos pueden representar. Esto además de posibles interferencias con las demás señales electromagnéticas presentes en el aire constantemente, las que de todas formas tienen un mucho menor impacto debido al diseño de LoRa, que busca minimizar este efecto.

De todas formas es destacable la identificación de zonas de buena recepción por LoRa con las ubicaciones probadas, en particular con el último caso, razón por la que es utilizada para poner a prueba los modelos de reconocimiento visual. Estos son probados a continuación, en conjunto con el componente de comunicación por LoRa, ubicando el *Gateway* en el lugar de la tercera prueba (segundo piso de casino).

4.2. Pruebas en conjunto de CC + LoRa

Una vez asegurado el funcionamiento de la comunicación en distintas zonas del campus, se pone a prueba el *Crowd Counting* (CC), ubicando el *Gateway* en el segundo piso del Edificio F, encima del casino, ya que cuenta con los mejores resultados en las pruebas de comunicación. El Portenta H7, por su lado, es instalado en salas con distintas condiciones de amplitud y ocupación, con ayuda de los soportes diseñados. Los lugares escogidos corresponden al Laboratorio de Desarrollo de Software (LDS), la sala A-016 y un lugar más reducido, como la Oficina de los Primos.

Estos experimentos, al contrario de los anteriores, se centran principalmente en pruebas de efectividad de *software*, ya que el análisis principal se enfoca en el sistema desarrollado, a ser ejecutado en el Portenta H7 y el Raspberry Pi.

Para esto, las variables medidas corresponden al CC, o sea, la cantidad exacta de personas detectadas y, por consiguiente, el nivel de ocupación mostrado en la interfaz visual, calculado con respecto a la capacidad máxima de cada sala. Aunque el usuario no ve los resultados del CC directamente en la interfaz visual, sino más bien el nivel de ocupación correspondiente, conocer con detalle la precisión de los modelos puede ayudar a realizar un análisis más efectivo. Además, se calcula la métrica conocida como RSME (Raíz del error cuadrático medio), ya definida anteriormente. Finalmente se miden nuevamente las métricas de comunicación mediante LoRa para conocer mejor la recepción en esas zonas específicas.

4.2.1. Sala 1: LDS

El Laboratorio de Desarrollo de Software (LDS) corresponde a un espacio común de la carrera de Ingeniería Civil Informática, ubicado en el edificio B frente al edificio del DEFIDER (Departamento de Educación Física, Deporte y Recreación), en donde los estudiantes se dedican a pasar el rato, estudiar y realizar trabajos en los computadores disponibles para la comunidad, los que se encuentran dispuestos en 10 islas con 4 ordenadores cada una (Ver Figura 30).

Si bien, la capacidad oficial de ocupación es de 60 personas, este valor considera exclusivamente a personas sentadas en cada escritorio y sillones. Al tomar en cuenta que pueden haber 2 personas sentadas en cada escritorio y a las personas paradas, este número se duplica, pudiendo alcanzar los 140 en el peor de los casos. Sin embargo, para efectos de este trabajo, se utiliza el promedio entre estos dos valores, o sea, una capacidad máxima de 100 personas.

El laboratorio se encuentra abierto entre 9:30 y 18:30, por lo que se prueba durante 45 minutos en un horario intermedio en que hayan personas presentes en él, contando aproximadamente 14 personas, en promedio. Durante el transcurso del experimento, se observa el comportamiento del sistema desde el *Gateway*, a través del programa de recepción de LoRa en C, y los servidores de la plataforma web, correspondientes al *Backend* y *Frontend*.



Figura 30: Fotos de LDS en dos momentos distintos de la experimentación.
Fuente: Elaboración propia.

```
digitalRead(dio0) == 1
writeReg:
92 40
receive : 16 bytes received.
writeReg:
8D 90
Packet RSSI: -107, RSSI: -108, SNR: -1, -----
Received 16 bytes.
3C 73 74 4C 3E 30 33 34 30 35 3C 65 4C 3E 00
Mensaje recibido [upload message]: <stL>034005<el>
Verificando validez de mensaje...
strlen(message): 15
strlen(message) != message_size: 0
Mensaje válido.
sala_id recibido:0340U
ocupacion recibida:0050U
OK[1700077340] Mensaje recibido y almacenado correctamente.
```

Escritura de registros

Valores de indicadores de calidad

Tamaño y contenido de bytes recibidos

Contenido de mensaje

Verificación de validez de mensaje

Extracción de datos de mensaje

Mensaje de éxito de almacenamiento en BD

Figura 31: Funcionamiento de receptor LoRa en terminal de Raspberry Pi.
Fuente: Elaboración propia.

En el caso del programa de recepción de LoRa (Ver Figura 31), se identifica cada sección del texto impreso en el terminal, en donde se encuentran los valores de las variables medidas en los experimentos, al igual que el tamaño en bytes de los mensajes y de su contenido en hexadecimal y en texto plano. Una vez recibido el mensaje, se indica que se realiza la verificación del mensaje, para asegurar que cumpla con el formato y tamaño definidos en el capítulo anterior. Una vez almacenados los elementos extraídos en la base de datos, se imprime un mensaje de éxito.

```

127.0.0.1 - - [15/Nov/2023 16:19:47] "GET /ult_ocupaciones HTTP/1.1" 200 -
127.0.0.1 - - [15/Nov/2023 16:19:47] "GET /salas HTTP/1.1" 200 -
query: [(6, 1, 0, datetime.datetime(2023, 4, 27, 21, 19, 26)), (4, 2, 20, datetime.datetime(2023, 4, 20, 18, 10, 48)), (2, 10, 10, datetime.datetime(2023, 4, 20, 18, 10, 35)), (27, 16, 5, datetime.datetime(2023, 11, 14, 12, 2, 35, 749298)), (65, 34, 4, datetime.datetime(2023, 11, 15, 16, 19, 23, 435033)), (1, 56, 5, datetime.datetime(2023, 4, 20, 18, 10, 24)), (45, 340, 0, datetime.datetime(2023, 11, 15, 16, 0, 33, 338914))]
127.0.0.1 - - [15/Nov/2023 16:19:57] "GET /ult_ocupaciones HTTP/1.1" 200 -
127.0.0.1 - - [15/Nov/2023 16:19:57] "GET /salas HTTP/1.1" 200 -
127.0.0.1 - - [15/Nov/2023 16:20:03] "POST /ocupacion HTTP/1.1" 200 -
query: [(6, 1, 0, datetime.datetime(2023, 4, 27, 21, 19, 26)), (4, 2, 20, datetime.datetime(2023, 4, 20, 18, 10, 48)), (2, 10, 10, datetime.datetime(2023, 4, 20, 18, 10, 35)), (27, 16, 5, datetime.datetime(2023, 11, 14, 12, 2, 35, 749298)), (66, 34, 4, datetime.datetime(2023, 11, 15, 16, 20, 3, 442105)), (1, 56, 5, datetime.datetime(2023, 4, 20, 18, 10, 24)), (45, 340, 0, datetime.datetime(2023, 11, 15, 16, 0, 33, 338914))]

```

Figura 32: Funcionamiento de server de *Backend* recibiendo consultas HTTP para agregar y obtener datos.

Fuente: Elaboración propia.

Por otro lado, en el terminal del servidor de *Backend* (ver Figura 32), se indica cada operación realizada, como la obtención de los datos de ocupación de salas, de parte del *Frontend*, mediante una solicitud HTTP de tipo GET, y cuando se almacenan nuevos elementos en la base de datos, mediante una solicitud HTTP de tipo POST recibida de parte del servidor de LoRa.

De esta forma, los resultados de las métricas registradas para la comunicación por LoRa, considerando una distancia entre nodos de aproximadamente 40 metros, son presentadas en la Tabla 7. Esta incluye la pérdida de paquetes aproximada y los promedios del RSSI, RSSI de paquete, y SNR.

Pérdida de paquetes	RSSI	RSSIpaquete	SNR
9 %	-106	-105	-1

Tabla 7: Valores promedio de resultados de prueba de CC + LoRa en LDS.

Se observa que los valores de las primeras tres variables son mejores que los promedios de las pruebas de comunicación por LoRa. Esto puede ser debido a la proximidad del nodo *Edge* al *Gateway*, en comparación al radio de mayor alcance de esos experimentos. Además, el nodo se encuentra en un lugar fijo que parece estar en un área de buena recepción, contribuyendo a estos resultados. Sin embargo, el valor promedio de la SNR es negativo, lo que implica que el ruido es mayor que la señal recibida, posiblemente debido a la cantidad de obstáculos presentes entre el nodo central y la ubicación del LDS.

Por otro lado, el modelo de CC correspondiente a esta sala es entrenado a partir de un *dataset* de 122 fotografías, capturadas en distintas zonas y ángulos del laboratorio. A esto se añade un proceso de *Data Augmentation* automático, provisto por *Edge Impulse*, que permite el aumento de la cantidad de datos a usar para el entrenamiento efectivo del modelo de CC.

Como se ve en la Figura 33, correspondiente a la interfaz visual del sistema (*Frontend*), el LDS (identificado por el cuadro destacado con rojo) presenta una ocupación baja, lo que corresponde con el nivel real de ocupación observado, dado que el porcentaje real de ocupación promedio es de $\frac{\text{ocupación actual}}{\text{ocupación máxima}} \times 100 = 14/100 = 14\%$.

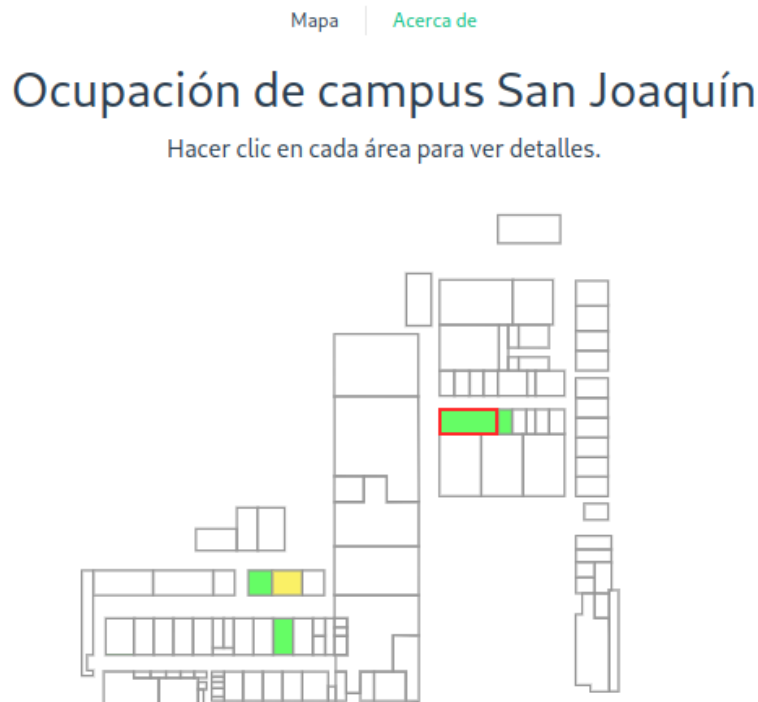


Figura 33: Mapa de campus San Joaquín con niveles de ocupación en un conjunto de salas.
El LDS se encuentra destacado con un contorno rojo.
Fuente: Elaboración propia.

Sin embargo, al comparar los valores específicos de CC calculados por el Portenta, con la cantidad real aproximada de personas presentes en el laboratorio, existe una gran diferencia, como se ve en la Figura 34. Efectivamente, el valor del RMSE corresponde a 10.8, superando el límite aceptable de error, ya que esto implica que al obtener valores de CC, el modelo puede sumar o restar 10 personas a la ocupación. Por lo tanto, es necesario seguir observando su comportamiento en otros lugares. Para posterior comparación, se obtiene además el RMSE normalizado, correspondiente a $\frac{RMSE}{Capacidad\ máxima} = \frac{10,8}{100} = 0,1$. De todas formas, por lo menos en este caso, este error es mitigado por la flexibilidad del sistema, ya que, a pesar de no calcular el valor exacto, el nivel de ocupación real de la sala es el mismo que el calculado.

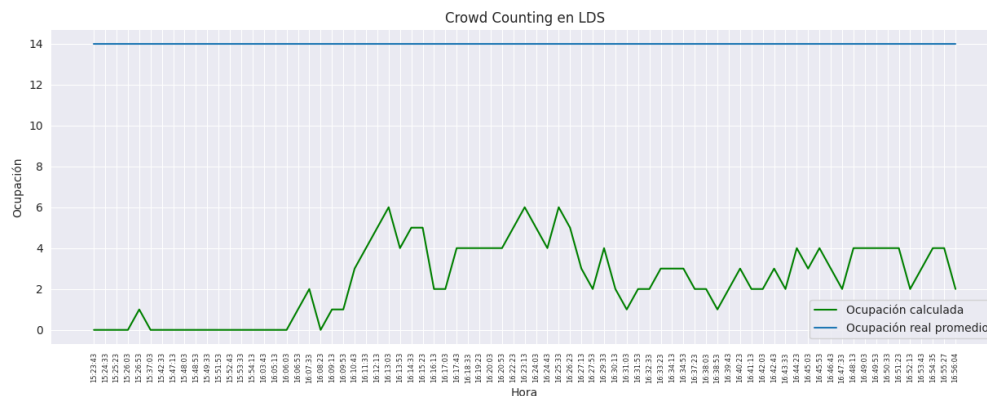


Figura 34: CC calculado por Portenta en comparación con valor real aproximado, a través del tiempo, en LDS.
Fuente: Elaboración propia.

4.2.2. Sala 2: Sala A-016

La segunda sala en la que se prueba el sistema corresponde a la A-016, que cuenta con un amplio espacio, con una capacidad de 99 personas, según datos oficiales de la Universidad, y a una distancia aproximada de 110 metros del Gateway. El experimento se lleva a cabo durante la clase de Computación Científica, realizada durante los bloques 5-6, de 10:55 a 12:05, en la que se encuentran aproximadamente 35 personas, como puede verse en la Figura 35.

En cuanto a la comunicación por LoRa, los resultados de sus métricas registradas son presentadas en la Tabla 7, la que incluye la pérdida de paquetes aproximada y los promedios del RSSI, RSSI de paquete y SNR. En este caso la distancia entre nodos es de aproximadamente 110 metros.

Pérdida de paquetes	RSSI	RSSIpaquete	SNR
15 %	-108	-99	1

Tabla 8: Valores promedio de resultados de prueba de CC + LoRa en sala A-016.

En este caso, se obtienen valores relativamente buenos, en comparación con lo medido hasta el momento. Los valores de RSSI y RSSI de paquete son iguales o mayores que los promedios de las pruebas de LoRa, y además el segundo, en particular, es mayor que en la prueba del LDS, al igual que el SNR, que llega a ser positivo, aunque aún así menor que el promedio de los experimentos de LoRa. Por otro lado, la pérdida de paquetes es ligeramente mayor al experimento previo.

De todas formas constituyen mejores resultados, a pesar de encontrarse a más del doble de distancia. Esto podría ser explicado por dos factores: La posición de la antena en el nodo



Figura 35: Fotos de la sala A-016 durante la prueba, obtenidas con celular exclusivamente para referencia de cantidad de personas.

Fuente: Elaboración propia.

Edge, en caso de haber apuntando de forma más directa al *Gateway*, y/o la presencia de una menor cantidad de obstáculos entre los nodos.

Para el entrenamiento del modelo de CC, se utiliza un *dataset* de 119 imágenes, exclusivamente de la sala de clases en distintos horarios de clase, y con distintos niveles de ocupación. Al igual que en el proceso de entrenamiento del modelo del experimento previo, se aplica *Data Augmentation* para intentar mejorar la generalidad del actual.

Como se aprecia en la Figura 36, en donde se demuestra la función de mostrar la etiqueta de cada sala al posar el *mouse* encima de ellas, la sala A-016 (ubicada en la esquina inferior izquierda) cuenta con una baja ocupación, lo que se contradice con el nivel de ocupación real del experimento, correspondiente al estado de color amarillo, entre 33 % y 66 % de ocupación. En este caso, la ocupación real es de alrededor de 35 personas, con una capacidad de 99, resultando en una tasa de ocupación real de 35 %, aproximadamente.

Mapa | Acerca de

Ocupación de campus San Joaquín

Hacer clic en cada área para ver detalles.



Figura 36: Mapa de campus San Joaquín con nivel de ocupación de sala A-016, indicado por la etiqueta con su nombre.
Fuente: Elaboración propia.

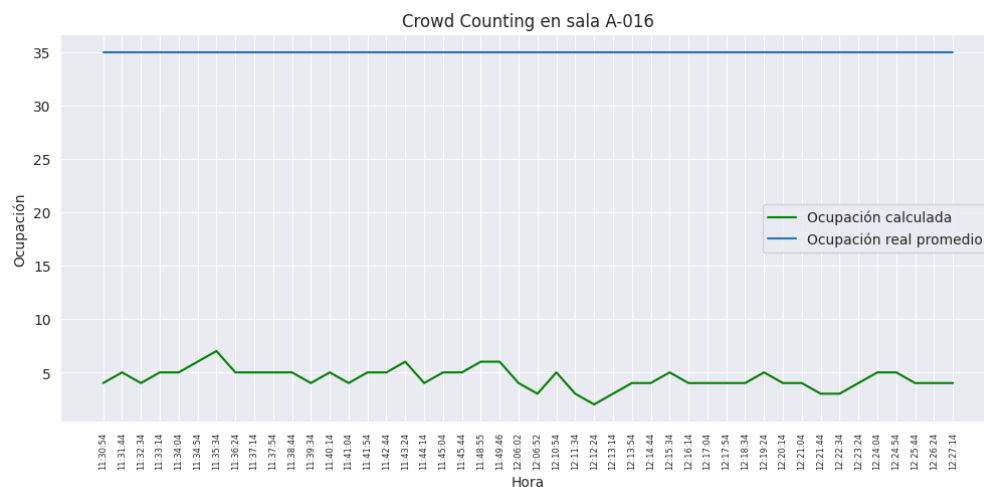


Figura 37: CC calculado por Portenta en comparación con valor real aproximado, a través del tiempo, en sala A-016.
Fuente: Elaboración propia.

Finalmente, al comparar los valores específicos del CC con los reales o *ground-truth* (Ver Figura 37), se obtiene un RMSE de 30.6, lo que indica que al realizar una medición, el modelo puede errar con 30 personas de diferencia, haciendo que el sistema no sea práctico de aplicar en este caso, a pesar de haber obtenido resultados aceptables en el envío de datos por LoRa. Además, el RMSE normalizado corresponde a 0.3.

Es necesario mencionar que el experimento cuenta con una interrupción de 32 minutos, causada por factores externos que no se pudieron controlar durante su desarrollo, pero que no afecta a los resultados.

4.2.3. Sala 3: Oficina de FabLab

Al contrario de las otras salas, esta corresponde a una pequeña oficina utilizada por los ayudantes del FabLab, laboratorio de fabricación ubicado en el Campus San Joaquín, entre el edificio A y las canchas de pasto. No tiene una capacidad oficial, pero para este caso se aproxima a unos 10, considerando la cantidad de asientos y espacio restante para personas de pie. El experimento se realiza con dos personas presentes en el laboratorio, debido a su ejecución fuera del periodo oficial de clases y de funcionamiento del laboratorio.



Figura 38: Foto de la oficina del FabLab durante la prueba, obtenida con celular.
Fuente: Elaboración propia.

Los resultados de las métricas registradas con respecto a la comunicación por LoRa, la que se da a unos 50 metros de distancia del Gateway, son presentadas en la FiguraT 9, la que incluye la pérdida de paquetes aproximada y los promedios del RSSI, RSSI de paquete, y SNR.

Pérdida de paquetes	RSSI	RSSIpaquete	SNR
17 %	-103	-100	-1

Tabla 9: Valores promedio de resultados de prueba de CC + LoRa en Oficina del FabLab.

La distancia al *Gateway* es mayor que en el experimento del LDS, pero se encuentran menos obstáculos entre los nodos en esta oportunidad, lo que podría explicar por qué los valores, excluyendo la pérdida de paquetes, son mejores que en ese caso. Una posible explicación para esta variable es que la dirección de la antena del Portenta H7 podría no haber sido la adecuada, en comparación con las demás pruebas.

Por otro lado, para el entrenamiento del modelo de CC se utilizan 247 fotos, y se incluyen las fotos del *dataset* JHU-CROWD++ que cuentan con menos de 10 personas, las que corresponden a 143 fotos. Esto con la esperanza de que el modelo pudiera generalizar de mejor manera sus resultados y que pudiera reconocer a una mayor variedad de personas.

Sin embargo, a pesar de estos esfuerzos, al observar el gráfico de la Figura 39, se evidencia que este modelo tampoco cumple con las expectativas de CC, obteniendo un valor de RMSE de 1.7, y RMSE normalizado de 0.2. Al comparar este error con la diferencia entre cada nivel de ocupación, correspondiente a 0.33, resulta una cifra significativa, ya que supera la mitad de dicho valor.

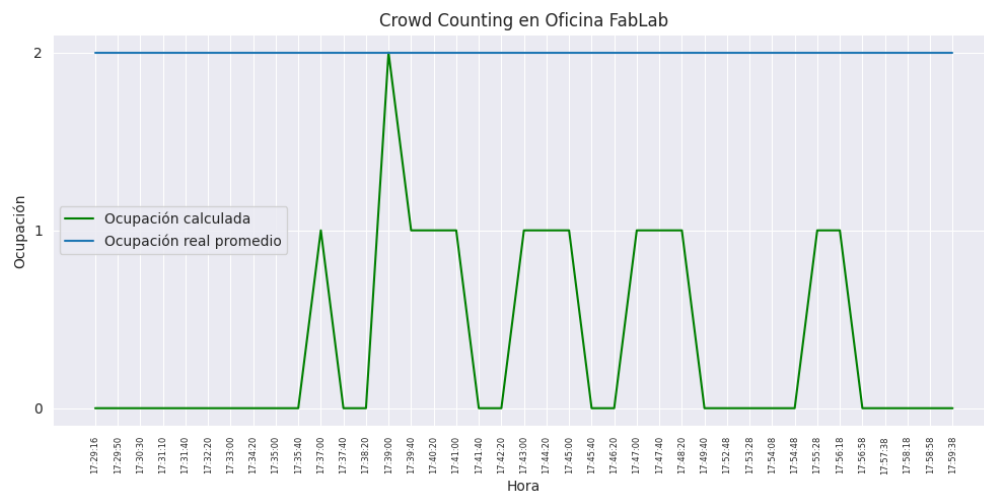


Figura 39: CC calculado por Portenta en comparación con valor real aproximado, a través del tiempo, en Oficina del FabLab.

Fuente: Elaboración propia.

En cuanto al resultado del nivel de ocupación indicado en la interfaz, visible en la Figura 40, se obtiene una clasificación baja (estado color verde, bajo el 33 % de ocupación), la que corresponde con el nivel real, obtenido a partir de la ocupación real de 2 personas y una capacidad máxima de 10 personas, lo que resulta en una tasa de ocupación del 20 %.

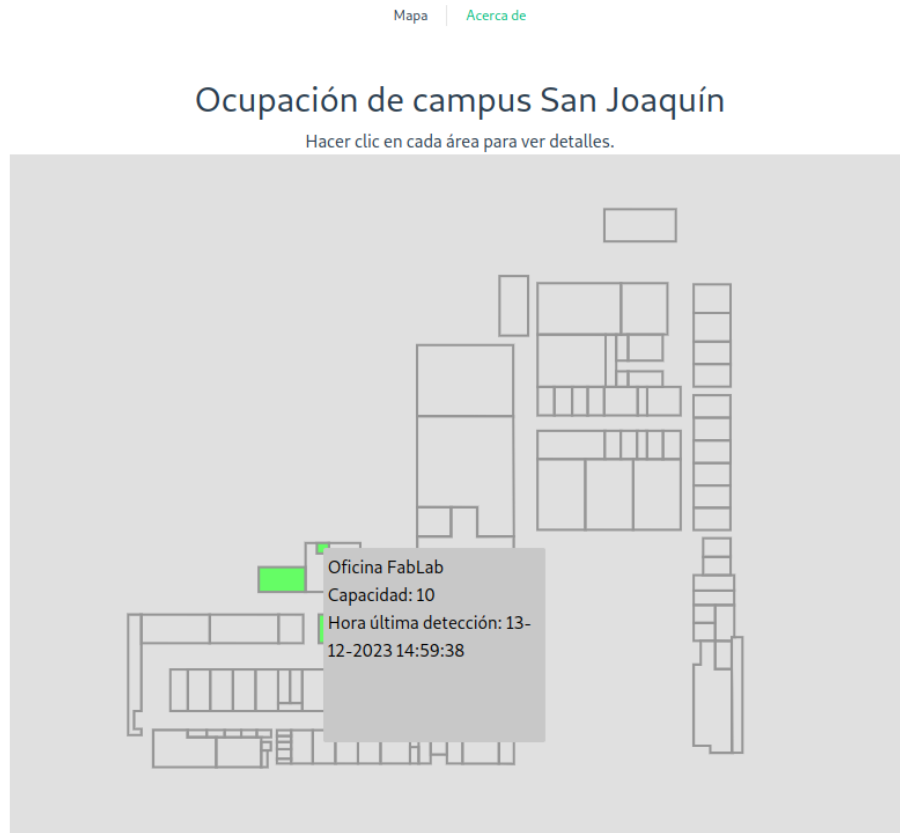


Figura 40: Mapa de campus San Joaquín indicando nivel de ocupación de la Oficina del FabLab, indicada por el recuadro con su información correspondiente.

Fuente: Elaboración propia.

Cabe destacar la implementación de una serie de mejoras a la interfaz visual, tales como un fondo que facilita diferenciar mejor la información, especialmente en caso de utilizar el tema oscuro del navegador. Además, se añade un recuadro que se activa al hacer clic sobre cada sala, el que incluye datos relevantes como el nombre del lugar, la capacidad máxima y la hora de la última detección, con tal de saber si el sistema se encuentra actualizado en cada zona.

4.2.4. Promedio de pruebas de CC + LoRa

Los valores promedio de los resultados de los mensajes recibidos de las tres pruebas se presentan en la FiguraT 10. Al compararlos con los promedios de las pruebas de LoRa, se obtienen mejores resultados en todas las variables (en especial la pérdida de paquetes), excepto por el SNR, que disminuye ligeramente.

Pérdida de paquetes	RSSI	RSSIpaquete	SNR
14 %	-106	-101	0

Tabla 10: Valores promedio de resultados de las tres pruebas de CC + LoRa.

Por otro lado, a partir de los resultados de CC, se obtiene un valor de RMSE normalizado promedio de 0.2, lo que implica que, en promedio, la detección del número de personas se desvía en un 20 %, ya sea disminuyendo o aumentando el valor. Esto no es óptimo, considerando que los niveles de ocupación utilizados en la interfaz web se definen cada 33 %, lo que fácilmente podría perjudicar la visualización de ocupación en el campus.

4.3. Discusión

A partir de los resultados, se tiene que los requerimientos funcionales definidos en el capítulo anterior, son efectivamente cumplidos, puesto que se envían los resultados del CC cada 30 segundos mediante paquetes por LoRa, con un formato que incluye el identificador de la sala, y la cantidad aproximada de personas. Finalmente, esto permite la visualización de los datos en la interfaz, junto con información de las salas al hacerles clic, como se muestra en el experimento de la oficina del FabLab.

Sin embargo, en cuanto a los no funcionales, el requerimiento “Recepción por LoRa en mayoría de salas de campus” no se cumple actualmente, como se ve en las pruebas de LoRa, ya que en las trayectorias realizadas se encontraron zonas completas en las que no funcionaba la comunicación, y en el mejor caso, correspondiente a la prueba 3 de LoRa, aún hay una pérdida de paquetes mayor al 40 %.

Para intentar solventar este problema, es necesario realizar más pruebas con el *Gateway* ubicado en distintos lugares, utilizar mejores antenas de envío en los nodos *Edge*, y/o posicionarlas de tal forma que haya una menor cantidad de obstáculos entre los nodos.

Por otro lado, el requerimiento “Detección de cada nivel de ocupación” se realiza parcialmente, debido al grado de error de parte de los modelos, en caso de presentarse más personas en las salas, como se evidenció en el experimento 2 de CC y LoRa.

Para resolver esto, no es seguro solamente mejorar los modelos para reducir el RMSE, dado que hay características propias del Portenta H7 que influyen en este aspecto. Por ejemplo, la cámara del Vision Shield, tiene una forma de ojo de pez, la que afecta negativamente en la profundidad de las imágenes captadas. De esta forma, al intentar reconocer personas a una mayor distancia, esto resulta en detecciones de menor precisión. Además de la memoria SDRAM reducida con la que cuenta que, entre otras cosas, imposibilita el almacenamiento de las imágenes procesadas durante las pruebas, y previene el uso de modelos de mayor complejidad e imágenes más grandes, que permitirían un nivel de detalle superior.

Por último, es necesario considerar que es fundamental para el correcto aprendizaje de los modelos de *Deep Learning* contar con un gran volumen de datos para su entrenamiento, lo que, debido a la lentitud del proceso de obtención, etiquetado y subida de imágenes de los lugares en los que se experimentó, no fue aplicable en la práctica. Esto, a pesar de contar con los procesos de *Data Augmentation*, cuyo aumento efectivo en la cantidad de datos, lamentablemente no se explicita en la plataforma *Edge Impulse*.

CAPÍTULO 5

CONCLUSIONES

En este trabajo se logró el desarrollo de un prototipo de sistema que cumple parcialmente los objetivos planteados en el comienzo. La solución elaborada consiste en una arquitectura *Edge* que utiliza LoRa para comunicación a distancias medias con un alto grado de obstáculos, y cuyo consumo eléctrico no pudo ser probado. Además se integran modelos de *TinyML* para contar personas en las salas del campus, pero que cuentan con un alto índice de error. Y por último se implementa una plataforma web que almacena los datos de ocupación de cada sala y, una vez clasificados en niveles, permite su visualización en una interfaz visual. Debido a esto y a pesar del cumplimiento de los objetivos, el sistema no se encuentra apto para ser aplicado de manera efectiva, ya que no cumple con su propósito por completo.

En relación a la comunicación mediante LoRa, a pesar de su funcionamiento, la presencia de una cantidad significativa de obstáculos entre el *Gateway* y varias de las salas resulta en una gran tasa de pérdida de mensajes y un bajo índice de SNR, como es evidenciado en los experimentos de LoRa. Esto implica que el sistema no pueda ser aplicado a la totalidad, o incluso a la mayoría, de las salas del campus. Entre las alternativas disponibles están buscar una posición que maximice la cantidad de regiones en las que hay una buena comunicación por LoRa, identificar las zonas de mejor recepción, dada una ubicación del *Gateway*, para aplicar CC en las salas ubicadas allí, y buscar soluciones que reduzcan la cantidad de obstáculos, por ejemplo ubicando el *Gateway* en un espacio abierto, al igual que las antenas de los nodos *Edge*, tal como en el techo de las salas.

Cabe mencionar que solo se prueba con un nodo *Edge* a la vez, lo que no permite conocer la efectividad del sistema con una mayor cantidad de dispositivos conectados, considerando la naturaleza de la arquitectura de red P2P centralizada propuesta. Además, teniendo en cuenta el costo de producción, se hace menos viable la instalación de una cantidad significativa de placas para los nodos *Edge*, por lo menos con el nivel con el que cuenta el sistema actualmente.

En cuanto a *Crowd Counting*, a pesar de su potencial para reconocer personas, no es suficiente para la aplicación requerida en este trabajo, como se evidencia en los resultados de las pruebas en cuanto al RMSE. Entre los factores responsables se encuentran las dimensiones reducidas de las imágenes analizadas por los modelos de ML, la falta de un volumen significativo de datos para entrenar los modelos y las características del Portenta H7 ya mencionadas, como la cámara de ojo de pez y su memoria reducida.

Para obtener mejores resultados y tener una mayor libertad de diseño de modelos de Redes Neuronales Convolucionales, en un comienzo se intentó diseñarlos utilizando *Python* con la librería *Tensorflow*. Esta permite la conversión del modelo al formato *Tensorflow Lite*, lo cual permite la compatibilidad con microprocesadores, al reducir su complejidad y espacio utilizado. Sin embargo, la conversión de formato no fue suficiente para optimizarlos al nivel

requerido por el Portenta, a causa de su reducida SDRAM. Por lo tanto, esta técnica no pudo ser implementada y se optó por el desarrollo en *Edge Impulse*, plataforma que cuenta con operaciones de optimización más complejas, pero privadas.

En lo que respecta al *hardware* utilizado, el Raspberry Pi no es una alternativa ideal para este sistema, debido a la incompatibilidad de la extensión *Dragino Hat* con LoRaWAN. El uso de este protocolo hubiera simplificado considerablemente el desarrollo, al no requerir el uso de C++ y el entorno nativo de Arduino, cuya dificultad de uso es significativamente mayor que la de *MicroPython* (lenguaje que integra nativamente el protocolo LoRaWAN), especialmente con respecto al empleo de modelos de *TinyML*.

Una alternativa consiste en agregar una placa (por ejemplo de tipo ESP32), con capacidad de comunicación por LoRaWAN, en vez de utilizar la extensión *Dragino*, lo que permitiría el uso de LoRaWAN en el *Gateway* sin sacrificar los beneficios del Raspberry Pi. Sin embargo, debido a disponibilidad de *hardware*, se debió utilizar la configuración descrita.

De todas formas, de esa forma se pudo desarrollar una implementación de la librería *TensorFlow Lite*, disponible públicamente para su uso en el Portenta H7, como fue indicado en la propuesta de solución. Esto aportó para poner a prueba los límites de este microprocesador, proporcionando antecedentes útiles sobre el desarrollo de *Machine Learning* en él. No obstante, es necesario considerar que, al emplear modelos de ML, y especialmente de *Deep Learning*, se requiere la implementación de técnicas de optimización para que el espacio de almacenamiento del modelo no supere el orden de los *Kilobytes*, como ya fue mencionado. De lo contrario, el programa se ejecuta durante unas cuantas iteraciones antes de agotar la SDRAM disponible, como fue comprobado.

5.1. Trabajo futuro

El Portenta H7, tiene potencial para aplicaciones variadas, además de la explorada en este trabajo, pero actualmente no es aprovechado por su baja adopción y documentación. A pesar de eso, existe una comunidad dispuesta a ayudar y seguir investigando sobre sus distintas aplicaciones, lo que puso a disposición valiosos aportes para la construcción de este sistema.

Con tal de continuar con su desarrollo en esta área, es necesario en primer lugar encontrar formas de optimizar el uso de la SDRAM, para permitir el alojamiento de una mayor cantidad de variables y funciones, y así explotar su potencial de complejidad de operaciones. Aprovechar su segundo núcleo de 240 MHz también sería valioso, ya que actualmente contiene aún menos documentación y ejemplos que el núcleo principal, dificultando su uso íntegro. Además de implementar el uso de distintos lenguajes en ambos núcleos de manera independiente, con comunicación entre ellos, lo que a pesar de ser prometido³⁵, tampoco se

³⁵“For example, it is possible to execute Arduino compiled code along with MicroPython one and have both cores to communicate with one another”[Arduino, 2023]

encuentra documentado oficialmente, sino simplemente comentado en foros³⁶ Por último, se recomienda emplear una fuente de alimentación de mayor capacidad, puesto que la actual dura unas horas, cuando debería ser capaz de mantenerse en ejecución por lo menos un día completo.

En caso de continuar con el desarrollo, es recomendado emplear otro dispositivo como *Gateway* que sea compatible con LoRaWAN, o seguir utilizando el Raspberry Pi o un dispositivo similar, y conectarlo a un microprocesador o controlador compatible con LoRaWAN, con el que pueda comunicarse para transmitir los mensajes recibidos, por ejemplo con una placa ESP32, como fue descrito previamente. En ambos casos se podrían aprovechar las ventajas propias de LoRaWAN, y además la posibilidad de emplear *MicroPython*, lenguaje de alto nivel que simplificaría considerablemente el desarrollo.

En cuanto a la calidad de la comunicación, aún existe un gran potencial por explotar para mejorar, principalmente mediante la ubicación de las antenas de los nodos en zonas abiertas, como fue comentado previamente. Además de probar distintos tipos de antenas con mayor potencia y dirigir los de la capa *Edge* más convenientemente hacia el *Gateway*.

La plataforma web, por otro lado, debería ser instalado en un entorno de producción, con tal de permitir el acceso público, pero asegurando que se mantenga restringido para las personas que frecuentan el campus San Joaquín. Una alternativa es mediante un método de inicio de sesión utilizando credenciales universitarias. Además, al implementar los servidores web en dispositivos externos, se puede reducir la carga del *Gateway*, que actualmente se encarga de ejecutar todos los servidores. De esta forma, sería posible contar con dispositivos de menor capacidad, reduciendo efectivamente costos.

En caso de lograr mejorar efectivamente los resultados de CC en el Portenta H7 y seguir las sugerencias descritas, se podrían expandir las aplicaciones del sistema, tal como la enumeración de personas en lugares reducidos, como ascensores, análisis de atención para publicidad, o conteo de otras entidades, como vehículos y productos, en cantidades limitadas.

³⁶<https://forums.openmv.io/t/portenta-micropython-on-m7-core-c-on-m4-core/6201>

ANEXOS

5.1. Prueba 1 LoRa

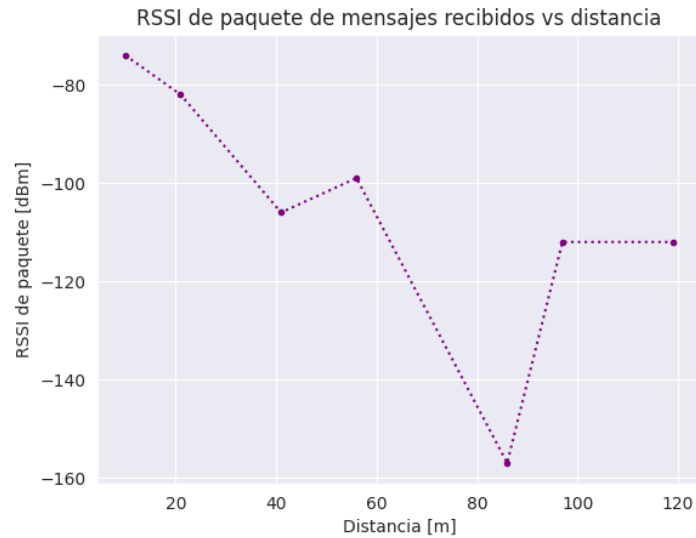


Figura 41: RSSI de paquete de mensajes recibidos con respecto a la distancia del nodo móvil al Gateway en prueba 1 de LoRa.
Fuente: Elaboración propia.

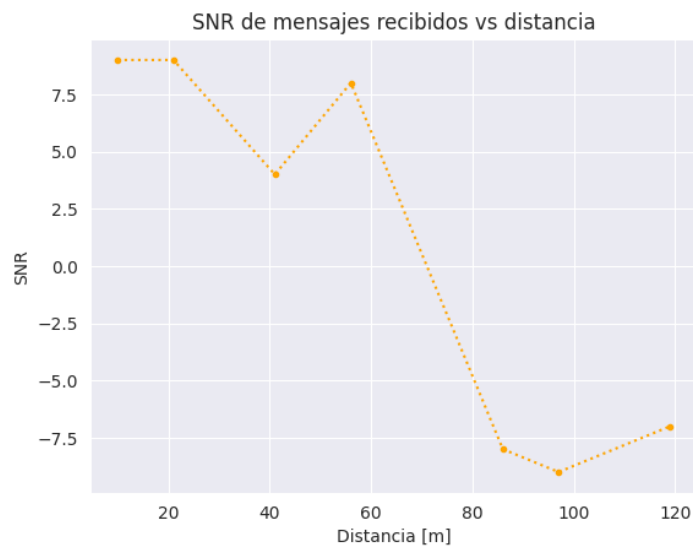


Figura 42: SNR de mensajes recibidos con respecto a la distancia del nodo móvil al Gateway en prueba 1 de LoRa.
Fuente: Elaboración propia.

# Mensaje	Recibido	Distancia aproximada [metros]	RSSI	RSSI de paquete	SNR
1	Sí	10	-110	-74	9
2	Sí	21	-111	-82	9
3	No	24	-	-	-
4	No	31	-	-	-
5	Sí	41	-111	-106	4
6	Sí	56	-107	-99	8
7	No	66	-	-	-
8	No	76	-	-	-
9	Sí	86	-112	-157	-8
10	Sí	97	-111	-112	-9
11	No	106	-	-	-
12	Sí	119	-110	-112	-7
13	No	134	-	-	-
14	No	147	-	-	-
15	No	154	-	-	-
16	No	162	-	-	-
17	No	172	-	-	-
18	No	181	-	-	-
19	No	188	-	-	-
20	No	199	-	-	-
21	No	209	-	-	-
22	No	215	-	-	-
23	No	223	-	-	-
24	No	231	-	-	-
25	No	236	-	-	-

Tabla 11: Resultados de prueba 1, con nodos ubicados en extremos opuestos del campus y envío de mensajes cada 10 segundos.

5.2. Prueba 2 LoRa

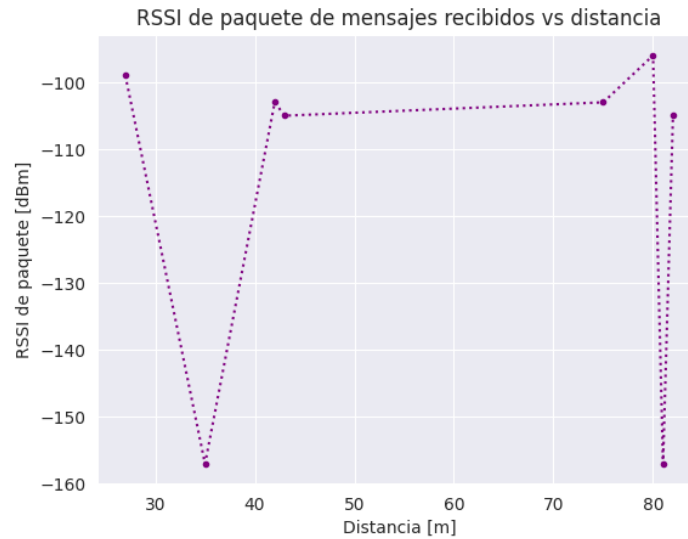


Figura 43: RSSI de paquete de mensajes recibidos con respecto a la distancia del nodo móvil al Gateway en prueba 2 de LoRa.
Fuente: Elaboración propia.

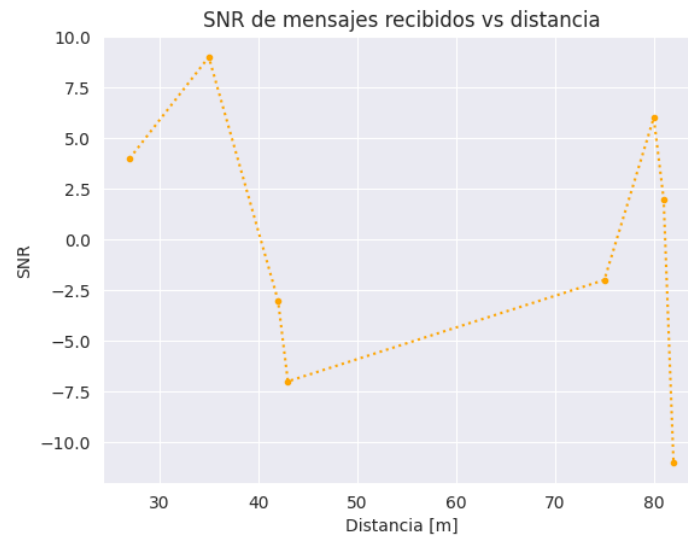


Figura 44: SNR de mensajes recibidos con respecto a la distancia del nodo móvil al Gateway en prueba 2 de LoRa.
Fuente: Elaboración propia.

# Mensaje	Recibido	Distancia aproximada [metros]	RSSI	RSSI de paquete	SNR
1	Sí	27	-102	-99	4
2	No	52	-	-	-
3	No	71	-	-	-
4	Sí	81	-103	-157	2
5	Sí	82	-105	-105	-11
6	Sí	80	-104	-96	6
7	No	84	-	-	-
8	No	94	-	-	-
9	No	101	-	-	-
10	No	117	-	-	-
11	No	131	-	-	-
12	No	125	-	-	-
13	No	115	-	-	-
14	No	110	-	-	-
15	No	104	-	-	-
16	No	102	-	-	-
17	Sí	75	-105	-103	-2
18	No	54	-	-	-
19	Sí	42	-104	-103	-3
20	No	29	-	-	-
21	Sí	43	-104	-105	-7
22	No	43	-	-	-
23	No	43	-	-	-
24	No	52	-	-	-
25	No	78	-	-	-
26	No	92	-	-	-
27	No	99	-	-	-
28	No	90	-	-	-
29	No	77	-	-	-
30	No	62	-	-	-
31	No	41	-	-	-
32	Sí	35	-104	-157	9

Tabla 12: Resultados de prueba 2, con *Gateway* ubicado en una sala central del campus en cuarto piso (12 metros de altura), y Portenta enviando mensajes cada 10 segundos.

5.3. Prueba 3 LoRa

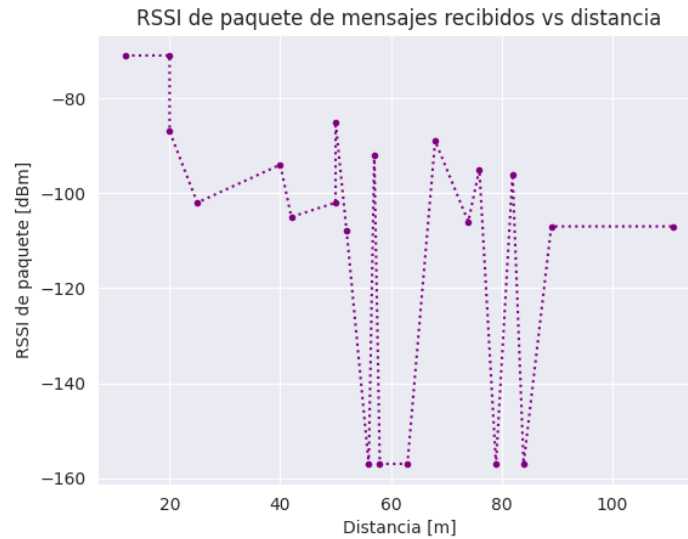


Figura 45: RSSI de paquete de mensajes recibidos con respecto a la distancia del nodo móvil al Gateway en prueba 3 de LoRa.
Fuente: Elaboración propia.



Figura 46: SNR de mensajes recibidos con respecto a la distancia del nodo móvil al Gateway en prueba 3 de LoRa.
Fuente: Elaboración propia.

VISUALIZACIÓN EN TIEMPO REAL DE OCUPACIÓN DE ESPACIOS EN CAMPUS SAN JOAQUÍN MEDIANTE
SISTEMA DE EDGE COMPUTING CON LORA

# Mensaje	Recibido	Distancia aproximada [metros]	RSSI	RSSI de paquete	SNR
1	No	7	-	-	-
2	Sí	20	-109	-87	9
3	Sí	40	-106	-94	8
4	Sí	50	-110	-102	4
5	Sí	52	-104	-108	2
6	Sí	57	-106	-92	9
7	No	72	-	-	-
8	Sí	84	-108	-157	4
9	No	105	-	-	-
10	No	114	-	-	-
11	Sí	111	-157	-107	-2
12	Sí	89	-108	-107	-3
13	No	75	-	-	-
14	No	66	-	-	-
15	Sí	63	-109	-157	6
16	Sí	50	-108	-85	9
17	Sí	56	-106	-157	9
18	Sí	68	-108	-89	9
19	Sí	82	-97	-96	8
20	Sí	76	-106	-95	8
21	No	73	-	-	-
22	Sí	74	-108	-106	0
23	No	79	-	-	-
24	No	90	-	-	-
25	No	106	-	-	-
26	No	96	-	-	-
27	No	89	-	-	-
28	No	88	-	-	-
29	No	86	-	-	-
30	No	84	-	-	-
31	No	89	-	-	-
32	Sí	79	-107	-157	7
33	Sí	58	-94	-157	4
34	Sí	42	-108	-105	1
35	Sí	25	-105	-102	4
36	Sí	20	-106	-71	0
37	Sí	12	-108	-71	9

Tabla 13: Resultados de prueba 3, con Gateway ubicado en segundo piso de edificio F (3 metros de altura), y Portenta enviando mensajes en dirección a Gateway cada 10 segundos.

REFERENCIAS BIBLIOGRÁFICAS

- [Adelantado et al., 2017] Adelantado, F., Vilajosana, X., Tuset-Peiro, P., Martinez, B., Melia-Segui, J., y Watteyne, T. (2017). Understanding the limits of lorawan. *IEEE Communications Magazine*, 55(9):34–40.
- [Adeogun et al., 2019] Adeogun, R., Rodriguez Larrad, I., Razzaghpour, M., Berardinelli, G., Christensen, P., y Mogensen, P. (2019). Indoor occupancy detection and estimation using machine learning and measurements from an iot lora-based monitoring system. pp. 1–5.
- [Arduino, 2023] Arduino (2023). Portenta h7 - features [online]. Disponible en: <https://docs.arduino.cc/hardware/portenta-h7/#features>. Accedido: 2023-01-31.
- [Awati, 2022] Awati, R. (2022). What is an embedded device? [online]. Disponible en: <https://www.techtarget.com/whatis/definition/embedded-device>. Accedido: 2023-01-31.
- [Badillo et al., 2020] Badillo, S., Banfai, B., Birzele, F., Davydov, I. I., Hutchinson, L., Kam-Thong, T., Siebourg-Polster, J., Steiert, B., y Zhang, J. D. (2020). An introduction to machine learning. *Clinical Pharmacology & Therapeutics*, 107(4):871–885.
- [Booqed, 2022] Booqed (2022). How to start analyzing your office space's data and reduce underutilized space [online]. Disponible en: <https://www.booqed.com/blog/what-is-space-occupancy-and-utilization>. Accedido: 2023-01-20.
- [Cisco, 2023] Cisco (2023). What is a network gateway? [online]. Disponible en: <https://www.cisco.com/c/en/us/products/routers/what-is-a-network-gateway.html>. Accedido: 2023-01-05.
- [Cooperativa.cl, 2018] Cooperativa.cl (2018). Metro presentó pantallas que informan sobre ocupación de vagones en trenes de línea 6 [online]. Disponible en: <https://www.cooperativa.cl/noticias/pais/transportes/metro/metro-presento-pantallas-que-informan-sobre-ocupacion-de-vagones-en/2018-04-04/202345.html>. Accedido: 2022-07-01.
- [Dabbs, 2019] Dabbs, M. (2019). The fundamentals of web application architecture [online]. Disponible en: <https://reinvently.com/blog/fundamentals-web-application-architecture/>. Accedido: 2023-12-20.
- [Dinh et al., 2023] Dinh, T., Nguyen, T., Phan, H.-P., Dau, V., Dao, D., y Nguyen, N.-T. (2023). Physical sensors: Thermal sensors. En Narayan, R., editor, *Encyclopedia of Sensors and Biosensors (First Edition)*, pp. 20–33. Elsevier, first edition edición.
- [Elbishlawi et al., 2020] Elbishlawi, S., Abdelpakey, M. H., Eltantawy, A., Shehata, M. S., y Mohamed, M. M. (2020). Deep learning-based crowd scene analysis survey. *Journal of Imaging*, 6(9).

- [Electronic-notes, 2022] Electronic-notes (2022). Multipath fading [online]. Disponible en: <https://www.electronics-notes.com/articles/antennas-propagation/propagation-overview/multipath-fading.php>. Accedido: 2023-02-15.
- [Gholamalinezhad y Khosravi, 2020] Gholamalinezhad, H. y Khosravi, H. (2020). Pooling methods in deep neural networks, a review.
- [Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- [Gouiaa et al., 2021] Gouiaa, R., Akhloufi, M. A., y Shahbazi, M. (2021). Advances in convolution neural networks based crowd counting and density estimation. *Big Data and Cognitive Computing*, 5(4).
- [Group, 2023] Group, V. (2023). The significance of people counting: Understanding the basics [online]. Disponible en: <https://vemcogroup.com/en/blog/the-significance-of-people-counting-understanding-the-basics>. Accedido: 2024-02-19.
- [Gupta y Snigdh, 2022] Gupta, S. y Snigdh, I. (2022). Clustering in lora networks, an energy-conserving perspective. *Wireless Personal Communications*, 122:1–14.
- [Huang et al., 2021] Huang, Z., Sinnott, R., y Ke, Q. (2021). Crowd counting using deep learning in edge devices. En *2021 IEEE/ACM 8th International Conference on Big Data Computing, Applications and Technologies (BDCAT '21)*, BDCAT '21, p. 28–37, New York, NY, USA. Association for Computing Machinery.
- [Impulse, 2022] Impulse, E. (2022). Fomo: Object detection for constrained devices [online]. Disponible en: <https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/object-detection/fomo-object-detection-for-constrained-devices>. Accedido: 2023-04-03.
- [Khan et al., 2019] Khan, W. Z., Ahmed, E., Hakak, S., Yaqoob, I., y Ahmed, A. (2019). Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235.
- [Lau et al., 2018] Lau, B. P. L., Wijerathne, N., Ng, B. K. K., y Yuen, C. (2018). Sensor fusion for public space utilization monitoring in a smart city. *IEEE Internet of Things Journal*, 5(2):473–481.
- [Lora-Alliance, 2022] Lora-Alliance (2022). What is lorawan® specification [online]. Disponible en: <https://loro-alliance.org/about-lorawan>. Accedido: 2022-07-06.
- [Louis Moreau, 2022] Louis Moreau, M. K. (2022). Announcing fomo (faster objects, more objects) [online]. Disponible en: <https://www.edgeimpulse.com/blog/announcing-fomo-faster-objects-more-objects>. Accedido: 2023-04-03.

- [Manoharan *et al.*, 2020] Manoharan, P., Nagarathinam, S., Vasan, A., y P. V. R. (2020). Keep it open, keep it safe: Maximizing space utilization in intelligent bio-safe buildings. En *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. Association for Computing Machinery.
- [Maresco, 2022] Maresco, D. (2022). What is occupancy and utilization? [online]. Disponible en: <https://spaceiq.com/blog/what-is-occupancy-and-utilization/>. Accedido: 2023-01-20.
- [Miller, 2021] Miller, A. (2021). 8 space occupancy and space utilization metrics for 2021 [online]. Disponible en: <https://www.iofficecorp.com/blog/space-utilization-space-occupancy-metrics-measurement>. Accedido: 2023-01-20.
- [Mineduc, 2021] Mineduc (2021). Informe 2021 matrícula de pregrado en educación superior [online].
- [NIST, 2022] NIST (2022). Glossary [online]. Disponible en: https://csrc.nist.gov/glossary/term/spread_spectrum. Accedido: 2023-02-15.
- [Ray, 2022] Ray, P. P. (2022). A review on tinyml: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 34(4):1595–1623.
- [Raychowdhury y Pramanik, 2020] Raychowdhury, A. y Pramanik, A. (2020). Survey on lo-ra technology: Solution for internet of things. En *Intelligent Systems, Technologies and Applications*, pp. 259–271. Springer Singapore.
- [Reader, 2022] Reader, T. C. (2022). Building a convolutional neural network [online]. Disponible en: <https://www.theclickreader.com/building-a-convolutional-neural-network/>. Accedido: 2022-07-01.
- [Redmon *et al.*, 2016] Redmon, J., Divvala, S., Girshick, R., y Farhadi, A. (2016). You only look once: Unified, real-time object detection.
- [Sadowski y Spachos, 2018] Sadowski, S. y Spachos, P. (2018). Rssi-based indoor localization with the internet of things. *IEEE Access*, 6:30149–30161.
- [Sampath, 2019] Sampath, S. (2019). *People counting and tracking based on LIDAR data*. Tesis doctoral.
- [Sandler *et al.*, 2019] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., y Chen, L.-C. (2019). Mobilenetv2: Inverted residuals and linear bottlenecks.
- [Semtech, 2022a] Semtech (2022a). Lora and lorawan: A technical review [online]. Disponible en: <https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan>. Accedido: 2023-02-15.
- [Semtech, 2022b] Semtech (2022b). What is lora®? [online]. Disponible en: <https://www.semtech.com/lora/what-is-lora>. Accedido: 2022-07-06.

- [Simonyan y Zisserman, 2015] Simonyan, K. y Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.
- [Sindagi *et al.*, 2020] Sindagi, V. A., Yasarla, R., y Patel, V. M. (2020). Jhu-crowd++: Large-scale crowd counting dataset and a benchmark method.
- [Tan y Le, 2020] Tan, M. y Le, Q. V. (2020). Efficientnet: Rethinking model scaling for convolutional neural networks.
- [Terabee, 2023] Terabee (2023). Time-of-flight technology for people counting: ensuring privacy and anonymity [online]. Disponible en: <https://www.terabee.com/time-of-flight-technology-for-people-counting-ensuring-privacy-and-anonymity/>. Accedido: 2024-02-19.
- [Wang *et al.*, 2021] Wang, S., Pu, Z., Li, Q., Guo, Y., y Li, M. (2021). Edge computing-enabled crowd density estimation based on lightweight convolutional neural network. En *2021 IEEE International Smart Cities Conference (ISC2)*, pp. 1-7.
- [wipro, 2023] wipro (2023). Edge computing - understanding the user experience [online]. Disponible en: <https://www.wipro.com/infrastructure/edge-computing-understanding-the-user-experience/>. Accedido: 2023-12-20.
- [Xu *et al.*, 2021] Xu, S., Wang, J., Shou, W., Ngo, T., Sadick, A.-M., y Wang, X. (2021). Computer vision techniques in construction: A critical review. *Archives of Computational Methods in Engineering*.
- [Yang *et al.*, 2022] Yang, H., Liu, H., Luo, C., Wu, Y., Li, W., Zomaya, A., Song, L., y Xu, W. (2022). Vehicle-key: A secret key establishment scheme for lora-enabled iov communications. pp. 787-797.
- [Zhu *et al.*, 2022] Zhu, F., Yan, H., Chen, X., y Li, T. (2022). Real-time crowd counting via lightweight scale-aware network. *Neurocomputing*, 472:54-67.