BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

ROBOT PARA SINIESTROS CONTROLADO POR INTERNET DE LAS COSAS

ENERO 2026

TESIS

PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

MARIANO ALDAIR AQUINO REGALADO

DIRECTOR DE TESIS

BARBARA EMMA SÁNCHEZ RINZA

ASESOR DE TESIS

BARBARA EMMA SÁNCHEZ RINZA

RESUMEN

Los sistemas distribuidos y el internet de las cosas son áreas muy importantes en el avance científico y en el desarrollo de soluciones para el mundo de hoy, permitiendo crear elementos como los robots que pueden aplicarse a distintas áreas como lo son medicina, industria, entre otros.

En este trabajo de tesis se desarrolla un prototipo de robot móvil de bajo costo diseñado para operar en escenarios de siniestro. El funcionamiento se basa principalmente en una arquitectura que se fundamenta en sistemas distribuidos e internet de las cosas, donde se integran múltiples módulos (tarjetas ESP32) que funcionan como nodos independientes, responsables de tareas específicas como comunicación, control de motores, captura de video y monitoreo en tiempo real, con esto se logra reducir la latencia y mejorar la tolerancia a fallos, mediante la distribución de la carga de trabajo.

El sistema incorpora múltiples sensores: humedad, luz y temperatura del ambiente y a distancia, además de contar con funciones importantes como lo son: la comunicación por voz y la transmisión de video en tiempo real. Visualizándose a través de una aplicación móvil desarrollada desde cero, donde se presentan los datos capturados por el prototipo, así como controlar el movimiento del robot mediante una red local basada en sockets TCP/UDP.

En la etapa final del desarrollo se realizaron pruebas de funcionamiento, evaluando resistencia, conectividad, integración y rendimiento, demostrando que el robot puede operar de manera estable en entornos adversos. Esto demuestra que es posible usar tecnología accesible y de bajo costo, para ayudar en tareas tan importantes como lo son la búsqueda y rescate de personas. Este proyecto ofrece una alternativa viable y reproducible que une IoT, robótica y sistemas distribuidos.

AGRADECIMIENTOS

Expreso mi más sincero agradecimiento a la **Secretaria de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI),** por el apoyo económico brindado durante mis estudios de maestría, sin lo cual no habría sido posible la realización de esta tesis.

A mi asesora, la Dra. Barbara Sánchez Rinza, por su guía, dedicación y orientación a lo largo del desarrollo de este proyecto, gracias por confiar y por las enseñanzas que dejo en mí.

A los miembros del jurado, que contribuyeron con sus observaciones, sugerencias y aportaciones para fortalecer este proyecto: Dr. Mario Rossainz López, Dr. José Arturo Olvera López y Dr. Iván Olmos Pineda

A todos los docentes, por el conocimiento transmitido a lo largo de mi formación dentro del posgrado.

A la Benemérita Universidad Autónoma de Puebla (BUAP) y a la Facultad de Ciencias de Computación (FCC), por el espacio, recursos y herramientas para poder crecer tanto en lo personal como en lo académico.

DEDICATORIA

A mis padres, Carmen Regalado Toledo y Mariano del Carmen Aquino Ortiz, por su apoyo y amor incondicional e inquebrantable, por los valores que me enseñaron, gracias por su esfuerzo, por motivarme y por siempre creer en mí, los amo.

A mis queridas abuelas, Catalina Ortiz “*Na Catalina*” y Juana Toledo Santiago “*Na Juanita*” por su educación, apoyo y amor eterno.

A mi compañera de vida, Zaira Gabriela Sierra Meléndez, por motivarme cada día y compartir conmigo este camino lleno de retos y aprendizajes, gracias por creer en mí y por el amor incondicional.

A toda mi familia, a todos y cada uno de ellos, gracias.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

CAPÍTULO 1
INTRODUCCIÓN

En este primer capítulo del proyecto se presenta la introducción donde se muestran la motivación, una descripción e identificación del problema a abordar, los objetivos del proyecto, así como el estado del arte referente a robots de ayuda manejados con internet de las cosas.

1.1. Planteamiento del problema

La tecnología ha ido avanzando de forma acelerada, tanto que hoy en día se puede contar con elementos que antes no se tenían o no eran tan accesibles como lo son a día de hoy, uno de esos elementos son los robots, los cuales se verán en este capítulo, lo que son, para qué son y su importancia, asimismo, lo que los forman y la importancia de los sensores que los componen, así como su uso en lugares de riesgo.

México es uno de los países con más terremotos por año, lo cual ocasiona derrumbes no solo en ciudades, sino también en lugares de difícil acceso como lo son las minas, edificios, y los barrancos, los cuales después de un derrumbe se vuelven un lugar de mucho más difícil acceso, también la infraestructura tanto económica como industrial, no permite una búsqueda tan exhaustiva de personas en estas circunstancias difíciles, por lo que se plantea el uso de herramientas (sensores, tarjetas, etc.) para la creación de un robot que cubra la necesidad de la búsqueda de personas en momentos de difícil acceso y esto, con recursos mucho más reducidos, tanto económicos como humanos.

1.2. Objetivos

1.2.1. Objetivo general

El principal objetivo de este proyecto es el desarrollo de un robot capaz de ingresar a lugares donde hayan sucedido siniestros, esto en circunstancias adversas donde el humano no puede y realizar actividades específicas, como comunicación y localización de personas, lo cual es de suma importancia en momentos de catástrofes como derrumbes.

El propósito central de este proyecto es la creación de un robot prototipo que tanto en software (programación y comunicación) y hardware sea suficientemente robusto para la ubicación donde se espera pueda funcionar, como lo son los derrumbes en las minas.

1.2.2. Objetivos específicos

Los objetivos específicos del proyecto son los siguientes, donde el prototipo es capaz de:

1. Evitar obstáculos
2. Detectar calor corporal
3. Detectar humedad
4. Detectar temperatura
5. Detectar luz
6. Brindar apoyo en la búsqueda de personas
7. Comunicación por voz
8. Tener conectividad constante
9. Proporcionar información constante
10. Enviar video en tiempo real del interior del siniestro

1.3. Trabajos relacionados

Actualmente existen múltiples robots de ayuda o soporte para personas, ya sea de manera directa o indirecta, dicho de otra forma, algunos son capaces de ayudar de forma física siendo ellos los que sacan a las personas en riesgo y otros a la distancia, necesitando que las personas hagan el trabajo físico, enfocándose en informar sobre el contexto del siniestro, algunos de los más importantes basan su funcionalidad en poder detectar personas, por ejemplo, el trabajo que presentan Hashimoto et al. [1] en donde auxilian a personas que se encuentren en una habitación detectando que estas hayan pasado por un percance, esto con ayuda de láseres integrados a múltiples robots móviles, de igual manera, el trabajo de Volkhardt y Gross [2] realizan detección de personas para su auxilio, pero la diferencia es que lo realizan mediante un solo robot móvil que busca a las personas por una habitación realizando un movimiento constante.

Siguiendo la base de los robots anteriores, Lovon et al. [3], presentan un trabajo similar en donde la diferencia radica en cómo se mueve el robot por la habitación ya que este, se basa en algoritmos de navegación para poder tener un robot capaz de moverse por el mejor lugar.

Los trabajos mencionados anteriormente, presentan robots para la detección de personas que requieran ayuda, pero se basan en informar a personal que realiza la ayuda física, Yang et al. [4] proponen un robot de apoyo en medicina, el cual es un brazo robótico con sensores para evitar chocar con obstáculos dentro de una sala de hospital y siendo la principal diferencia con los demás proyectos que este brinda el soporte físico directamente, sin requerir de personal humano.

Un trabajo importante y similar a los que se busca en este proyecto es el de Zhao et al. [5], el cual es un robot móvil con conectividad internet de las cosas y que implementa todas sus funcionalidades con ayuda de sensores y tecnología de la mejor calidad, pero con un costo económico elevado comparado a este proyecto.

Estos trabajos se pueden ver de mejor manera con su año de publicación, títulos y objetivos respectivos en la siguiente Tabla 1.3.1.

Tabla 1.3.1: Marco referencial sobre robots de apoyo a personas

	Año	**Titulo**	**Objetivo**	**Forma de búsqueda**
1	2011	*Seguimiento de personas mediante láser mediante múltiples robots móviles [1]*	Seguimiento de personas	Basado en láser, realizado por un grupo de robots móviles ubicados cerca uno del otro.
2	2013	*Finding People in Apartments with a Mobile Robot [2]*	Buscar y localizar a una persona que no se encuentre en las proximidades del robot moviéndose por un apartamento	Detecta las partes del cuerpo de manera separada (para las piernas un láser, para la cara un clasificador tipo AdaBoost con el método de detección de caras de Viola-Jones, para el torso un detector basado en Histograma de Gradientes Orientados)
3	2016	*People Detection and Localization in Real Time during Navigation of Autonomous Robots [3]*	Detección y localización de personas con respecto al plano del robot durante la navegación de un robot autónomo	Algoritmo de detección de rostros basado en HOG (Histograma de Gradientes Orientados).
4	2024	Coordinating Obstacle Avoidance of a Redundant Dual-Arm Nursing-Care Robot [4]	Brazos de cuidados de enfermería, que evitan obstáculos	No buscan personas, pero brinda soporte a los pacientes y al personal médico.
5	2025	Development of an autonomous fire rescue and surveillance robot integrating LiDAR, thermal imaging, and AI-based navigation [5]	Robot de rescate en lugares de riesgo como incendios	Basado en sensores de calor y geolocalización.

1.4. Alcances y limitaciones

1.4.1. Alcances del prototipo

- Desarrollo de un prototipo de robot para entornos de siniestro: se construyó un prototipo físico capaz de desplazarse en espacios reducidos, irregulares o parcialmente colapsados.
- Desarrollo de un sistema que opera con internet de las cosas: el usuario final puede controlar el prototipo desde una aplicación móvil mediante comunicación inalámbrica (Wi-Fi), logrando visualizar en tiempo real la información recolectada por el robot.
- Integración de sensores: el prototipo mide temperatura, humedad, intensidad de luz y presencia de gases lo que permite evaluar el entorno.
- Localización de personas: el prototipo con ayuda de la cámara de la tarjeta ESP32-CAM puede realizar una búsqueda básica de personas esto, mediante calor a distancia con ayuda de un sensor de temperatura.
- Comunicación por voz: el prototipo está integrado con un micrófono y una bocina, lo que permite tener una comunicación básica por voz con las personas que puedan estar en riesgo en el siniestro.
- Desarrollo de una aplicación móvil: se diseñó una aplicación donde se pueden ver los datos que el prototipo captura en tiempo real.

1.4.2. Limitaciones

Dentro de las principales limitaciones para el desarrollo del proyecto se encuentran:

- Recursos: el sistema ha sido diseñado pensando en que se tienen recursos limitados en cuestión económica y de componentes.
- Bajo costo: ligado a lo anterior, pero buscando que el prototipo pueda ser replicable debido al costo reducido y a la accesibilidad de los componentes

- Baja latencia: el prototipo aun con las limitaciones anteriores debe ser capaz de tener una conectividad eficaz.
- Tamaño: el tamaño del robot debe ser reducido para poder acceder a las zonas de difícil acceso, esto también tomando en cuenta el peso.

De igual manera existen limitaciones técnicas las cuales son:

- Conexión Wi-Fi local: el prototipo depende de estar dentro del rango de la red Wi-Fi, ya que opera conectándose a la red del dispositivo móvil y no a internet público.
- Transmisión de video: la cámara de la tarjeta ESP32-CAM cuenta con un ancho de banda limitado, por lo que el video no es HD y puede presentar retrasos o interferencia.
- Autonomía: teniendo en cuenta las limitaciones principales, el bajo costo y el tamaño principalmente, el prototipo cuenta con una batería acorde al costo y al peso necesario, pero sacrificando más autonomía del robot, por lo que el tiempo de operación del robot se ve afectado.
- Movilidad: el prototipo está diseñado para poder desplazarse por terreno irregular, pero no puede escalar grandes obstáculos o escombros inestables.
- Sensores: la capacidad de lectura del ambiente se limita a la capacidad de los sensores que están integrados al robot, por lo que los datos recolectados pueden tener falsos positivos.

Otras limitaciones importantes son las del entorno:

- Interferencias: pueden surgir interferencias en la señal Wi-Fi y provocar perdida temporal o total de comunicación, dependiendo de algunas circunstancias del entorno, como lo puede ser la humedad, metales pesados y estructuras colapsadas donde, por ejemplo, muchas edificaciones en México, en la estructura de los techos cuentan con una malla la cual en un derrumbe puede funcionar como una jaula de Faraday.

- Condiciones extremas: aunque el prototipo se diseñó pensando en que sea robusto, no está preparado para entornos muy extremos como lo son, temperaturas muy elevadas o fuego directo, inundación completa o ambientes totalmente tóxicos.
- Campo de visión: la cámara de la ESP32-CAM tiene un campo de visión limitado, por lo que la exploración depende de poder maniobrar el prototipo.

Finalmente, las limitaciones operativas del robot son:

- Rescate: no realiza el rescate de manera física, solo localiza, monitores y comunica, permitiendo que personal de rescate si pueda entrar al lugar y realizar el auxilio, asimismo no puede levantar escombros.
- Navegación: no cuenta con navegación autónoma, el movimiento requiere control manual y no genera rutas automáticas ni mapas del entorno.
- Comunicación: puede establecer una comunicación por voz, pero la calidad del audio puede verse afectada por múltiples cuestiones como ruido, interferencias o baja potencia de la bocina.

1.5. Organización del documento de tesis

La tesis se organiza en 5 capítulos, los siguientes capítulos se organizan de la siguiente manera:

- Capítulo 2: se presenta el marco teórico, con conceptos básicos sobre temas que son importantes para este trabajo de tesis: qué son los robots, los sensores y la tecnología internet de las cosas.
- Capítulo 3: se presenta el diseño y desarrollo del proyecto el cual se divide en cinco secciones principales: el diseño del prototipo (requerimientos), diagramas, que van ligados al diseño y que plantean de manera visual el funcionamiento del robot, la arquitectura del sistema que se usó, el hardware y software seleccionado para el

desarrollo y finalmente la sección de la implementación tanto del robot como de la aplicación.

- Capítulo 4: se presentan los resultados finales del robot, después de la implementación mostrada en el capítulo 3, esto con métricas sobre su funcionalidad, esto mediante pruebas de funcionalidad, integración, rendimiento, conectividad y resistencia.
- Capítulo 5: se presentan las conclusiones del proyecto.

CAPITULO 2
MARCO TEORICO

En este capítulo se presenta el marco teórico, mostrando conceptos importantes como lo son los robots, sensores e internet de las cosas, todo lo que se aplicó como antecedente para el desarrollo del prototipo.

2.1. ¿Qué son los robots?

A lo largo del crecimiento del humano, tanto en conocimiento como en cuestiones físicas y de civilización ha ido avanzando la creencia de nueva vida, como bien menciona John M. Jordan [6] *"una historia continua de esfuerzos para crear vida artificial, replicar, mejorar o superar las características humanas"* asimismo llevando al campo de estos; la robótica que se puede definir como *"una disciplina científica que aborda la investigación y desarrollo de una clase particular de sistemas mecánicos"* [7].

Subir Kumar Saha [8] menciona la existencia de unas leyes de la robótica:

1. Un robot no debe dañar a un ser humano.
2. Un robot debe obedecer las órdenes que le son dadas por un ser humano, excepto las que entran en conflicto con la primera ley.
3. Un robot debe proteger su propia existencia, a menos que ésta entre en conflicto con las dos primeras leyes.
4. Un robot podrá tomar el trabajo de un ser humano, pero no debe dejar a esta persona sin empleo.

Por lo que un robot es algo virtual o mecánico, desarrollado para un fin, que permita cubrir una necesidad o tarea, mejorando, superando o incluso reemplazando la función y características humanas para dicha tarea, esto se puede ver en la

industria, seguridad, medicina, entre otras áreas, en medicina, por ejemplo, en el reemplazo de extremidades en pacientes como se puede ver en la Figura 2.1.1.

Figura 2.1.1: Brazo robótico

Con los ejemplos mencionados anteriormente surge una necesidad de volver a los robots mucho más autónomos, permitiendo la movilidad y extender el campo de acción de los robots, por ejemplo, en el caso de esta tesis, en donde se busca la exploración de lugares de difícil acceso como son las minas o lugares de siniestros, esto con ayuda de un robot móvil, con este contexto, es pertinente considerar la definición propuesta por Anival Ollero Baturone [9], quien señala que *"se trata también de incrementar la autonomía limitando todo lo posible la intervención humana"* asimismo menciona que *"la autonomía de un robot móvil se basa en el sistema de navegación automática. En estos sistemas se incluyen tareas de planificación, percepción y control".*

Kelly, R. y Santibáñez, V. [10] señalan una categorización de los robots en dos grupos:

- Robots manipuladores
- Robots móviles

Las principales diferencias entre estos son que los manipuladores interactúan con objetos en un entorno fijo, mientras que los robots móviles tienen la capacidad para desplazarse por todo su entorno, asimismo otra diferencia es la autonomía, ya que los robots móviles por su funcionalidad pueden adaptarse a los cambios en su entorno.
Por ende estos últimos pueden clasificarse de acuerdo con el terreno donde se desempeñan: robots terrestres, robots acuáticos y robots aéreos.

Los robots están compuestos por diferentes elementos los cuales se encargan de ser los sentidos de este, y pueden verse como objetos que emulan la vista, tacto y oído del humano y a estos se les conoce como sensores.

2.1.1. Robots de rescate

Los robots implementan funciones que pueden emular a un humano y dentro de esas funciones también pueden realizar actividades que sustituyan a una persona, y más cuando se habla de entornos donde alguien puede estar en riesgo, y ahí es cuando los robots de rescate entran, bajo ese contexto, en la obra de Murphy, Tadokoro y Kleiner se indica que un robot de rescate es aquel sistema mecánico o robótico para ser operado en entornos destructivos o de desastre, con la finalidad de ayudar en misiones criticas [11], de igual manera, en las capacidades que puede tener un robot de rescate se encuentran que, sea capaz de explorar entornos peligrosos, inestables o inaccesibles para un rescatista, proporcionando observación, recolección de datos, evaluación de riesgos y asistencia operativa durante emergencias [12].

Las misiones críticas que se deben cumplir son [13]:

- Reconocimiento
- Búsqueda
- Mapeo

- Detección de víctimas
- Proporcionar conocimiento de la situación del entorno

También, un punto importante es la evaluación de riesgos, lo que implica el análisis de distintos factores como lo son [14]:

- La estabilidad de la estructura colapsada.
- La presencia de materiales peligrosos.
- El nivel de accesibilidad.
- La probabilidad de fallos mecánicos o de comunicación.
- El impacto sobre la seguridad de los rescatistas y las víctimas.

2.2. Sensores

Un sensor es un dispositivo que detecta y mide diferentes tipos de variables físicas o químicas [15], como la temperatura, la luz, la distancia, la presión, el sonido, etc.

Por lo general, convirtiendo el medio en magnitudes visibles y cuantitativas, como señalan C. Romero, F. Vázquez, y C. de Castro, *"la misión de un sensor es la conversión de magnitudes de una determinada naturaleza a otra, generalmente eléctrica"* [16] asimismo, describen una clasificación de estos (Tabla 2.2.1):

Tabla 2.2.1: Clasificación de sensores

Tipo	**Atendiendo al tipo de señal**
Activos	Alimentados eléctricamente a los niveles apropiados
Pasivos	Sin necesidad de alimentación eléctrica
Continuos	Señales continuas proporcionadas
Discretos	Señales discretas proporcionadas

Dentro de los diferentes tipos de sensores que se pueden utilizar para los sentidos del robot y los cuales se dividirán de acuerdo con sus características son:

2.2.1. Sensores de proximidad

Brindan soporte en la detección de obstáculos como, por ejemplo, el **sensor de proximidad ultrasónico** el cual mide la distancia mediante el uso de ondas ultrasónicas manda una onda ultrasónica y espera a recibir la onda que rebota en el objeto y midiendo la distancia contando el tiempo entre la emisión y recepción. Este se puede visualizar en la Figura 2.2.1

Aplicaciones: Útil en robótica para evitar obstáculos, en sistemas de aparcamiento y en sistemas de seguridad.

Figura 2.2.1: Sensor de proximidad ultrasónico

2.2.2. Sensor de inclinación (SW-520D)

El módulo de inclinación está equipado con un sensor de inclinación y un potenciómetro. Puede adjuntarlo a cualquier objeto y determinará si el objeto está inclinado o no. Es simple, ya que devuelve una salida digital, ver Figura 2.2.2.

Aplicaciones: Monitoreo de objetos que deben permanecer en una orientación específica, como dispositivos electrónicos, o en sistemas de seguridad para detectar caídas.

Figura 2.2.2: Sensor de inclinación

2.2.3. Sensor de proximidad infrarrojo

Está basado en un emisor y receptor infrarrojo, emite el haz infrarrojo de manera constante y detecta obstáculos si algún objeto interrumpe el haz, se puede ver en la Figura 2.2.3.

Figura 2.2.3: Sensor de proximidad infrarrojo

2.2.4. Sensor de temperatura y humedad (DHT11)

Permite obtener la información de la temperatura y humedad del aire convirtiendo las magnitudes recabadas en señales eléctricas, se puede ver en la Figura 2.2.4.

Aplicaciones: Ideal para estaciones meteorológicas, control ambiental en invernaderos y sistemas HVAC (calefacción, ventilación y aire acondicionado).

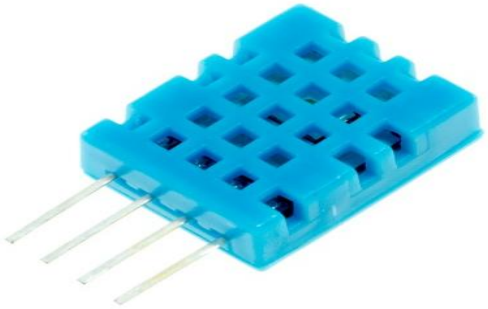

Figura 2.2.4: Sensor de temperatura y humedad

2.2.5. Sensor de gas (MQ)

Los sensores MQ están compuestos por un sensor electroquímico que varía su resistencia al estar en contacto con las sustancias. Están diseñados para detectar sustancias como gases inflamables, calidad del aire o detección de alcohol en aire, ver Figura 2.2.5.

Aplicaciones: Detección de fugas de gas en hogares o industrias, monitoreo de la calidad del aire y sistemas de alarma.

Figura 2.2.5: Sensor de gas

2.2.6. Sensor de luz (fotorresistor)

Es un componente electrónico cuya resistencia se varía dependiendo del aumento de intensidad de luz incidente en el ambiente, ver Figura 2.2.6.

Aplicaciones: Control de iluminación automática, detección de presencia de luz y aplicaciones en dispositivos que necesitan ajustar su brillo según la luz ambiental.

Figura 2.2.6: Fotorresistor

2.2.7. Sensor de temperatura a distancia

Es un sensor de temperatura sin contacto que mide la temperatura a través de radiación infrarroja, ver Figura 2.2.7.

Aplicaciones: Ideal para medir temperatura corporal a distancia.

Figura 2.2.7: Sensor de calor corporal

2.2.8. Micrófono

Para el sentido del oído del robot se puede integrar un micrófono el cual recoge las vibraciones del aire y las transforma en una señal eléctrica que puede procesarse mediante un algoritmo, ver Figura 2.2.8.

Aplicaciones: Sistemas de reconocimiento de voz, activación de dispositivos mediante sonido, y detección de ruidos ambientales.

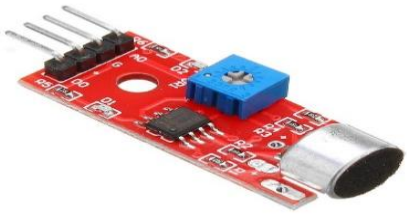

Figura 2.2.8: Micrófono

2.2.9. Bocina

Para la voz del robot se puede implementar una bocina, creando tonos de diferentes frecuencias y reproduciéndolos mediante el altavoz, ver Figura 2.2.9.

Figura 2.2.9: Bocina

2.2.10. Cámara (ESP-32 CAM)

Para la visión del robot se implementará con una cámara está incluida dentro de la placa ESP-32 la cual es muy similar a la placa Arduino, solo que está orientada al internet de las cosas (IoT) permitiendo conectividad sin necesidad de otras placas (Bluetooth o Wi-Fi) y permitiendo usar la cámara con la que ya cuenta., ver Figura 2.2.10.

Especificaciones

- Voltaje de Alimentación de la placa: 5V en pin 5V
- Voltaje de trabajo de la placa: 3.3V
- CPU 32 bits de doble núcleo de baja potencia Xtensa LX6
- Wi-Fi 802.11b/g/n Access point & Station
- Bluetooth 4.2 2.4Ghz; BT 2.0 y 4.0 BLE
- Tipo de cámara: OV2640 2MP
- Frecuencia principal de hasta 240 MHz

Figura 2.2.10: ESP32 CAM

Limitaciones: la principal limitación de la tarjeta ESP32-CAM es acerca de la memoria ya que esta posee aproximadamente 520 KB de SRAM, de los cuales una parte significativa es usada por el sistema operativo, el manejo del Wi-Fi y los buffers, por lo que al final tiene una cantidad de memoria muy limitada, por esta razón la ESP32-CAM no puede procesar ni transmitir video de alta resolución, esto para evitar saturar la memoria.

Ventajas: la tarjeta tiene dos principales ventajas, el primero es su bajo costo y su gran capacidad de integración en proyectos de internet de las cosas, ya que está compuesto por sensores para la conectividad (Bluetooth y Wi-Fi) y también es muy sencillo de programar al contar con la mayoría de las bibliotecas disponibles en Arduino.

Tomando en cuenta esa limitación y las ventajas, se vuelve muy importante el poder dividir las responsabilidades del prototipo en diferentes módulos, ya que al cargar a una sola ESP32-CAM con todos los procesos esperados que haga el robot provocaría:

- Saturación del microcontrolador
- Múltiples perdidas de conexión
- Inestabilidad del video en tiempo real
- Latencia mayor en la respuesta del robot

Por esto mismo se implementa una arquitectura distribuida para este proyecto, donde cada tarjeta funciona como una unidad independiente pero coordinadas mediante internet de las cosas, ofreciendo múltiples ventajas como:

- Robustez
- Escalabilidad
- Balance de carga
- Mantenimiento sencillo

Tomando en cuenta todos los sensores mencionados, se pueden elegir algunos para obtener un robot que complete todos los sentidos humanos, pero con la posibilidad de tener más y mejores capacidades comparadas con las de una persona o mejor dicho sin ciertas limitaciones, así como un objetivo específico para lo cual sea diseñado, en la Tabla 2.2.2 se puede ver una comparativa entre todos los sensores mencionados anteriormente:

Tabla 2.2.2: Comparativa de sensores

Sensor	Función	Modelo Común	Rango de Medición	Voltaje de Operación	Señal de Salida
Sensor de Proximidad	Detecta la presencia de objetos cercanos	HC-SR04 (ultrasónico)	2 cm - 4 m (HC-SR04)	5V	Digital (pulso)
Sensor de Temperatura y Humedad	Mide la temperatura y la humedad del ambiente	DHT11, DHT22	Temperatura: 0-50°C (DHT11), -40-80°C (DHT22) Humedad: 20-90% RH (DHT11), 0-100% RH (DHT22)	3.3V - 5V	Digital (datos seriales)
Sensor de Inclinación	Detecta la inclinación en una o más direcciones	SW-520D	0°-180°	3.3V - 5V	Digital (on/off)
Sensor de Gas	Detecta la presencia de gases específicos (e.g., CO, metano)	MQ-2, MQ-135	300-10000 ppm (MQ-2)	5V	Analógica (voltaje)
GPS	Compatible con sistemas GPS estándar	NEO-6N	±2.5 metros	2.7V - 3.6V	Digital
Sensor de calor corporal	Mide temperatura corporal a distancia	MLX90614	-40°C a +85°C.	3.3V o 5V DC.	Digital
Fotorresistor	Mide la intensidad de la luz	LDR (Light Dependent Resistor)	1-500kΩ	3.3V - 5V	Analógica (resistencia)
Micrófono para Arduino	Capta sonidos del entorno y mide su intensidad	Electret	0-1023 (ADC del Arduino)	3.3V - 5V	Analógica (voltaje)

2.3. Internet de las cosas

Otro componente importante para el robot es la conectividad, existen diferentes formas en las que un robot puede comunicarse o recibir comandos, para esto existe un paradigma que es el internet de las cosas, este puede permitir el acceso e interacción con una amplia variedad de dispositivos como sensores, actuadores, teléfonos móviles, etc. Asimismo, ofrece la capacidad de medir, inferir y comprender indicadores del entorno, recopilar datos y realizar tareas sofisticadas [17].

Conforme a la perspectiva de Atzori, Iera y Morabito [18] existen 3 enfoques principales para definir el paradigma IoT:

1. Orientado a internet: middleware
2. Orientado a las cosas: sensores
3. Orientado a lo semántico: conocimiento

Para este proyecto se toman en cuenta los tres enfoques, para integrar el paradigma, permitiendo realizar una conexión segura entre el robot y la aplicación móvil para su control logrando así facilitar la supervisión del robot, dicho de otra forma, permite que el robot y la aplicación convivan en una red local de internet, compartiendo así información y comandos, que no deben perderse en estos entornos de difícil acceso como lo son los siniestros.

CAPITULO 3
DISEÑO E IMPLEMENTACIÓN

El desarrollo del prototipo, en cuestión de ingeniería de software, se basó primero en un diseño que tiene en cuenta las limitaciones, realizando una búsqueda de requerimientos y realización de diagramas, lo que permitió servir como base del proyecto en todos los aspectos importantes: hardware, software y comunicación.

3.1. Diseño del prototipo

3.1.1. Requerimientos

Para los requerimientos del proyecto, desde la postura teórica de Somerville [19] estos se pueden dividir en:

Requerimientos funcionales: servicios que el sistema debe proveer, su funcionamiento de acuerdo con entradas o situaciones específicas.

Requerimientos no funcionales: limitaciones sobre los servicios o funciones que ofrece el sistema o proyecto, incluyendo limitaciones de tiempo y de funcionalidad.

Aunque si bien se pueden clasificar en estos grupos, los requerimientos no son necesariamente independientes, ya que un requerimiento puede restringir a otro, por lo que los requerimientos no solo indican que servicios o funciones debe realizar un sistema, sino que también detallan la funcionalidad que se requiere para que los servicios sean realizados de manera correcta.

De acuerdo con [20] la obtención de requisitos consiste en capturar el propósito y funcionalidades del sistema desde la perspectiva del usuario.

Y para esto se deben tomar en cuenta los siguientes puntos: los objetivos generales, el dominio del problema, los actores del proceso y el entorno de operación.

- Dominio del problema: para este proyecto el dominio será el de robótica en entornos adversos, como lo son los siniestros.
- Actores: usuarios administradores del robot, así como las personas involucradas en el entorno (mineros)
- Entorno de operación: lugares de difícil acceso, como lo son minas, derrumbes de edificios, etc. Tomando en cuenta la señal y complejidad de acceso.

Una vez definidos los puntos anteriores se especifica la técnica para la obtención de requisitos, para este caso se usará la técnica de planteamiento de escenarios, con la cual se pueden contextualizar los requisitos de acuerdo con escenarios específicos, planteando la pregunta “¿Qué pasaría sí?” o “¿Cómo realizar esto?”, por lo que la pregunta principal para la obtención de requerimientos es ¿Cómo hacer que el robot pueda entrar a lugares de difícil acceso y pueda comunicarse y hallar personas? para esto se toman en cuenta los sensores de la sección 1.2 los cuales servirán para la implementación de las funciones del robot, donde se planteará un sensor como un sentido especifico del robot y por ende ligado a una función que debe realizar el robot, de modo que se realizan las siguientes preguntas:

1. ¿Cómo buscar el mejor camino tomando en cuenta los obstáculos?
2. ¿Qué pasaría si no hay luz?
3. ¿Cómo buscar a las personas?
4. ¿Qué pasaría si hay gases peligrosos?
5. ¿En caso de encontrar a personas cómo comunicarse?
6. ¿En caso de que las personas encontradas no puedan comunicarse cómo actuar?
7. ¿Cómo realizar la comunicación en lugares de difícil acceso?
8. ¿Qué pasaría si el lugar es muy estrecho?
9. ¿Qué pasaría si el lugar es demasiado extenso?
10. ¿Qué pasaría si el lugar es muy húmedo?
11. ¿Cómo establecer el proceso en casos de defunciones?

Una forma de realizar esto es con ayuda de UML, con el diseño de diagramas, donde se modelarán las funciones antes mencionadas, así como los actores y como proceden ante cada posible evento que ira transcurriendo en el tiempo.

Por lo que los requerimientos de la tesis son:

3.1.2. Requerimientos funcionales

- Localizar personas: Detectar personas mediante la cámara, así como el calor corporal o sonidos.
    - Se requiere el sensor MLX90614 para medir calor corporal a distancia.
- Movilidad: Moverse de forma guiada controlada por una aplicación móvil, así como evitar obstáculos.
    - Se requiere de:
    - Tarjeta ESP-32 CAM: como cerebro y antena para la conexión a la aplicación móvil mediante Bluetooth y Wi-Fi.
    - Sensor de proximidad HC-SR04: para detectar obstáculos.
- Comunicación: Transmitir la imagen en tiempo real, así como la ubicación y todas las mediciones de los sensores del robot.
    - Se requiere de:
    - ESP32-CAM: la cámara con la que cuenta permitirá visualizar imágenes en tiempo real.
    - Micrófono
    - Bocina
- Resistencia: Contar con la capacidad de soportar condiciones extremas como poca luz, humedad y gases peligrosos.
    - Se requiere de:
    - Sensor DHT11: proporciona valores de humedad y temperatura.
    - Sensor de LDR: mide la intensidad de la luz en el ambiente.

3.1.3. Requerimientos no funcionales

1. Seguridad: Debe operar sin riesgo de provocar más daños en el entorno

2. Autonomía: consumo de energía eficiente y operar por largos periodos de tiempo

3. Robustez: Construido con materiales resistentes para soportar las condiciones adversas.

4. Conectividad: procesar y transmitir la información de manera fluida y constante

5. Tamaño: Debe ser compacto para poder pasar por espacios estrechos

6. Facilidad de uso: Contar con interfaces intuitivas para que el equipo de rescate pueda monitorear o controlar al robot.

3.2. Diagramas

"El diagrama no es el diseño: el diagrama es una representación de (parte de) un modelo de diseño, que captura un aspecto del diseño de una manera que pueda ser discutido" [21]. para este proyecto se realizan los siguientes diagramas:

3.2.1. Casos de uso

El primer diagrama es de casos de uso teniendo en cuenta a los actores y funciones del robot, *"un caso de uso modela la interacción entre el producto y los usuarios del producto"* [22]. En la Figura 3.2.1. se observa el diagrama de casos de uso:

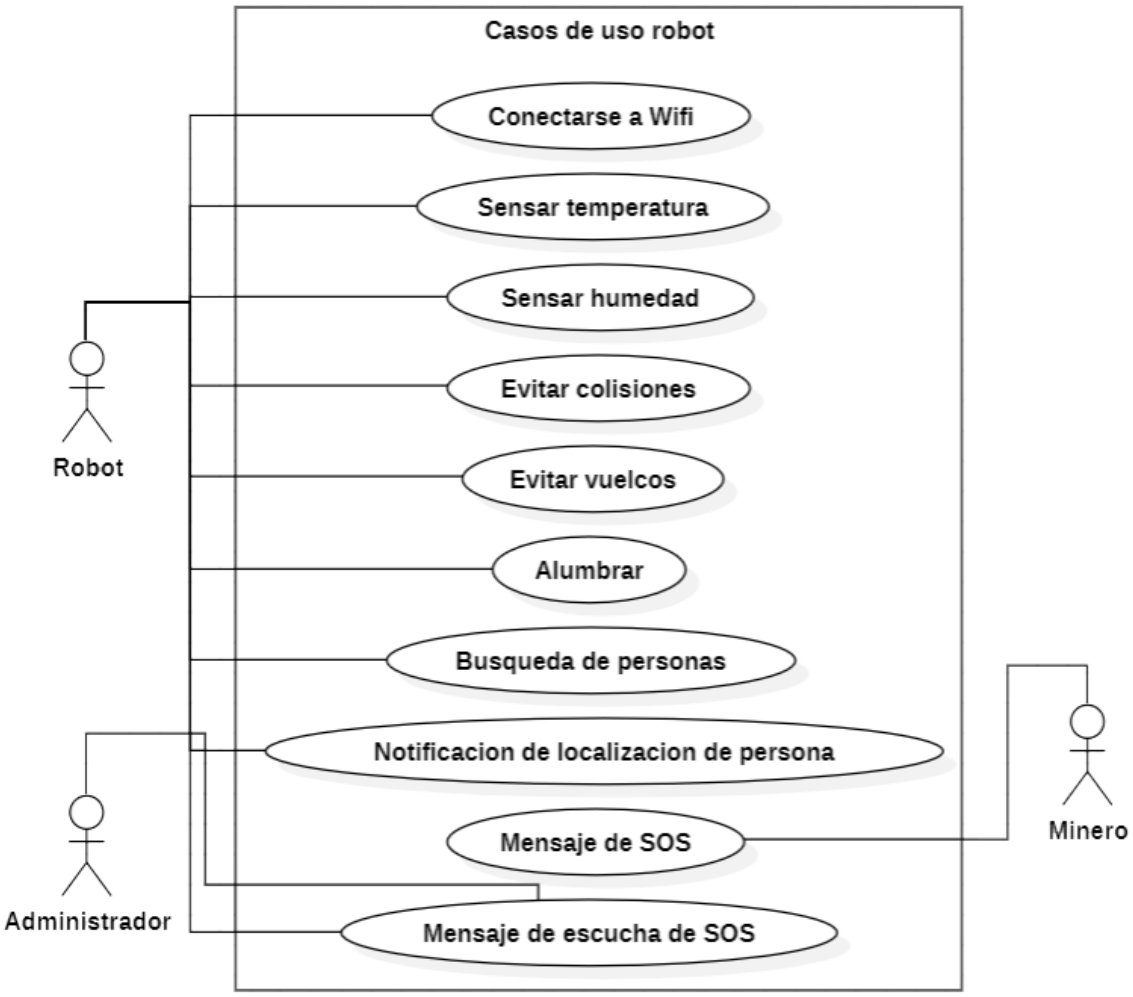

Figura 3.2.1: Diagrama de casos de uso robot para minas

En el diagrama de la Figura 3.2.1 se puede observar a los actores, los cuales son el robot, el administrador que estará a cargo de ir observando los avances del robot dentro del lugar del siniestro, en este caso de ejemplo, una mina y los mineros o personas dentro del desastre, los cuales podrán mandar mensajes, el primordial de auxilio llamado S.O.S. y el administrador podrá responder a este mensaje con ayuda del robot para comenzar el proceso adecuado de extracción por personas calificadas.

3.2.2. Diagrama de estados finitos

En este apartado se muestra el diagrama de estados finitos del robot, el cual es una representación gráfica de los estados y todas las posibles transiciones entre los estados de un sistema, protocolo, proceso, etc.

Por ende, se plantean los estados posibles del robot y las transiciones que se esperan realizar en los casos que se presenten.

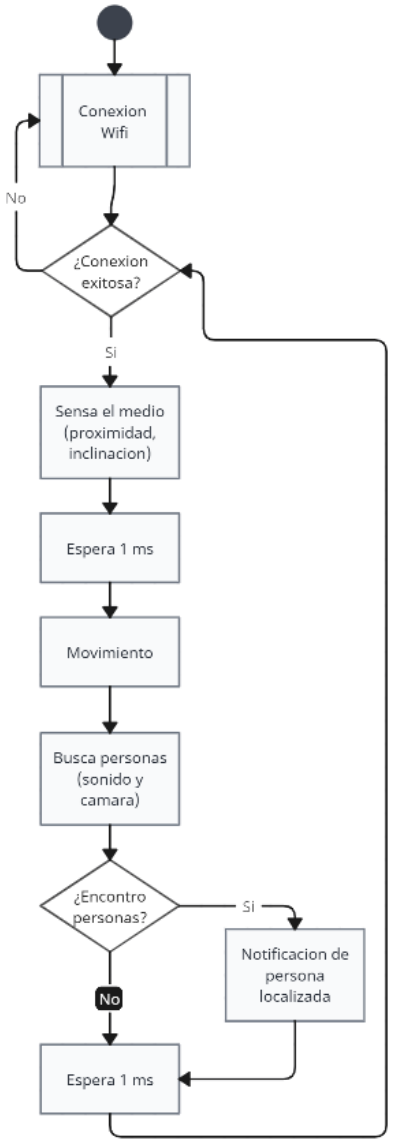

Figura 3.2.2: Máquina de estados finitos robot

En el diagrama de la Figura 3.2.2 se observa los pasos (estados) que atraviesa el robot para la localización de las personas, desde el establecimiento de la conexión inicial donde las tarjetas ESP32-CAM se conectan a una conexión Wi-Fi determinada, hasta el censado y espera para el movimiento del robot que se podrá ver desde la aplicación para finalmente determinar si encontró personas o no (con ayudad el sensor de temperatura a distancia) y realizar una validación de la conexión.

3.2.3. Diagrama de bloques

En el siguiente diagrama de bloques de la Figura 3.2.3. se representa en bloques lo distintos sensores, así como su orden de ejecución dentro del proceso del robot para la búsqueda de persona, en donde inicia recibiendo 5 volts de entrada en este caso para las tarjetas ESP-32 la cual puede recibir los 5 volts en su pin de acuerdo con las especificaciones mencionadas en el punto 1.2, una vez recibido el voltaje inicia el sensor de proximidad, así como el de luz, para poder determinar si el lugar es apto para moverse y posteriormente inicia el motor para encaminarse a la búsqueda, asimismo se realiza el censado de gases nocivos y de humedad y temperatura, y finalmente la búsqueda con la cámara y sensor de temperatura a distancia y la bocina así como la posible comunicación con el individuo con ayuda del micrófono.

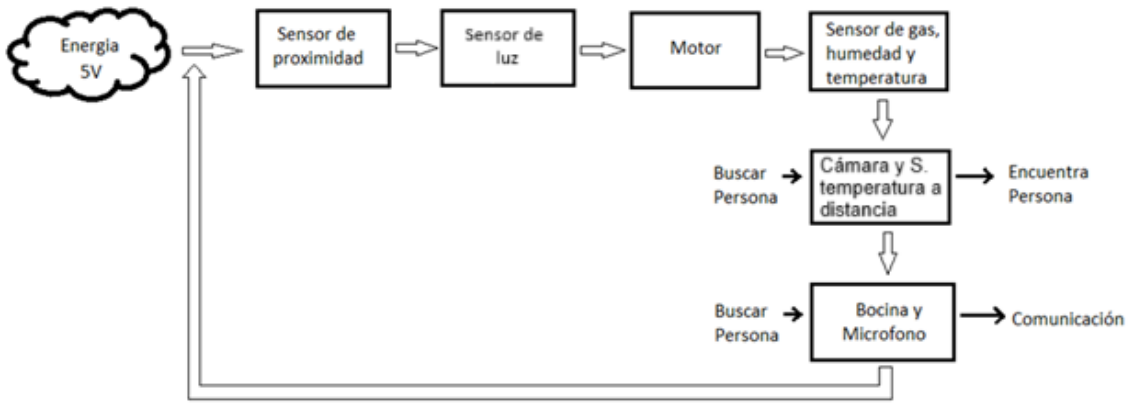

Figura 3.2.3: Diagrama de bloques

3.2.4. Diagrama general

El diagrama general muestra los sensores que se usaran, así como la conexión con los pines de la tarjeta principal la cual para este caso es la ESP32-CAM, teniendo de referencia este para la implementación práctica, donde se usan 3 ESP32-CAM ver Figura 3.2.4.

Para la primera figura la cual sería el primer ESP32-CAM este estaría dedicado a la cámara y a la luz led que viene integrado a la tarjeta, esto debido a la cantidad de memoria que necesita la transmisión de video en tiempo real.

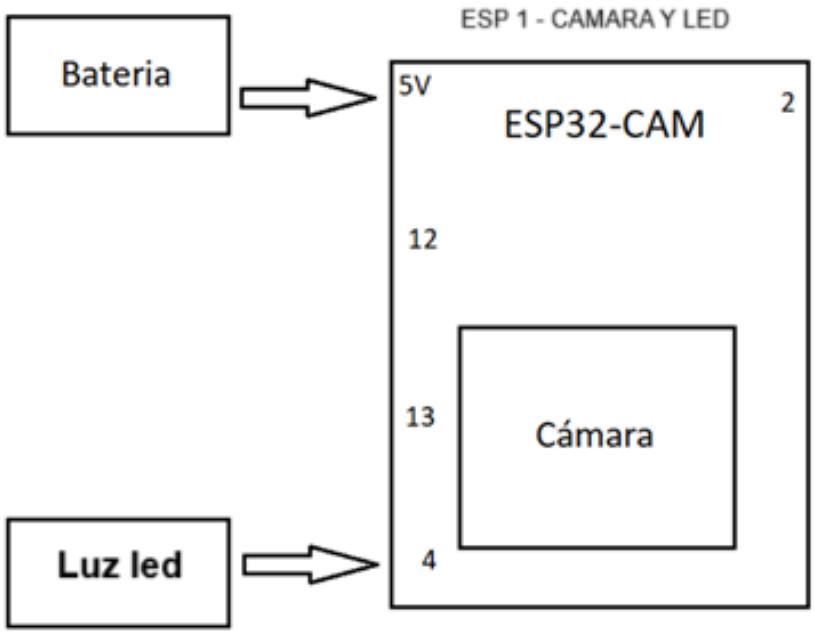

Figura 3.2.4: Diagrama general conexiones de la cámara y el led de la tarjeta ESP

Para la segunda imagen (Figura 3.2.5) la cual sería la segunda ESP32-CAM, este sería el responsable del control de los motores con ayuda de una tarjeta L298N y del sensor de proximidad para evitar obstáculos y parar los motores en caso de ser necesario, así como del sensor de temperatura a distancia.

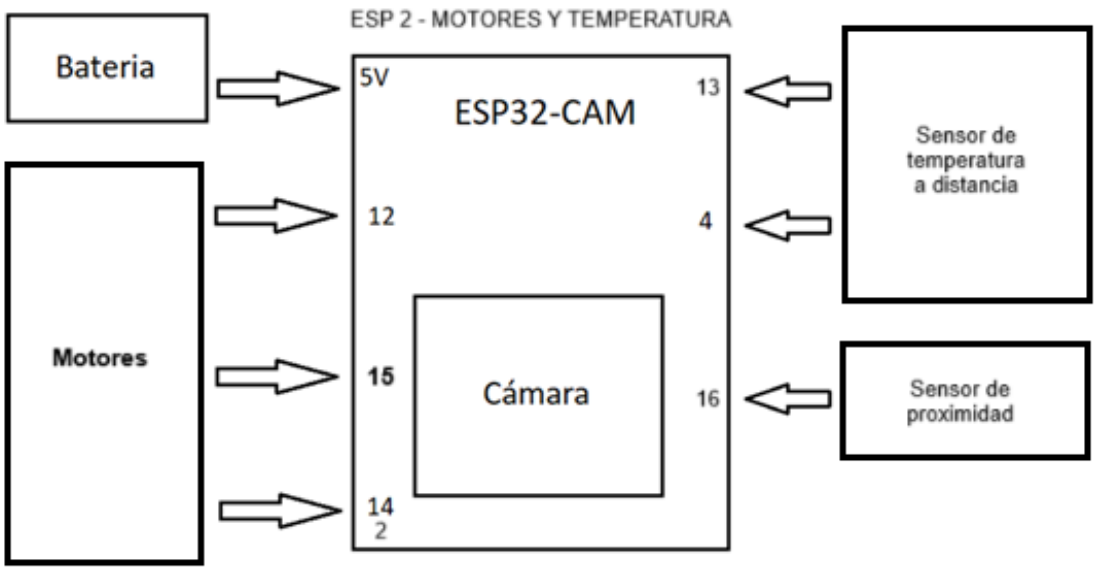

Figura 3.2.5: Diagrama general de conexiones de los motores y sensor de temperatura

Y para la tercera ilustración (Figura 3.2.6) la cual sería la tercera ESP32-CAM, en este estarían los sensores faltantes para la recopilación de información del entorno, así como para la comunicación con personas dentro del siniestro en caso de ser necesario, esto con un micrófono y una bocina.

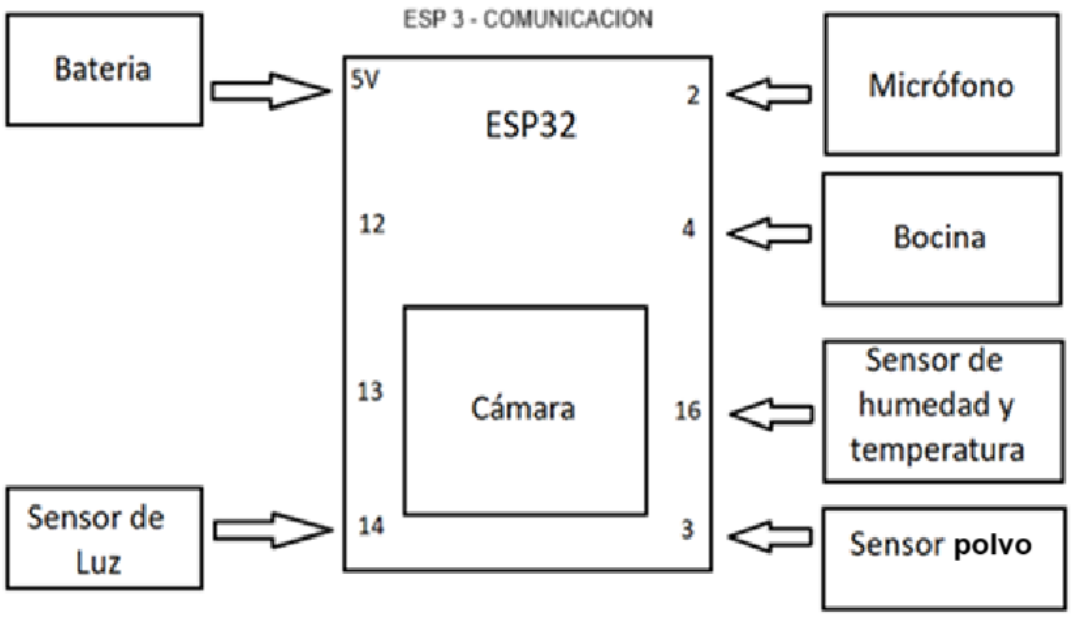

Figura 3.2.6: Diagrama general de conexiones de comunicación

La comunicación entre las tres tarjetas será mediante sockets con el dispositivo móvil (smartphone) desde la aplicación, donde se podrá visualizar la información de cada ESP32-CAM sin interferencias o errores.

3.2.5. Diagrama de secuencia

En el diagrama de secuencia (Figura 3.2.7) se puede ver la interacción desde la conexión hasta la notificación a los usuarios conectados, esto visto desde el punto de software en el cual se tienen 3 interfaces:

Interfaz de conexión: esta para realizar la conexión correcta del robot con el Wi-Fi o alguna otra tecnología inalámbrica.

Interfaz de búsqueda: en esta el usuario podrá monitorear en tiempo real todos los sensores incorporados al robot, con una sección de cámara, otra para los distintos parámetros de los demás sensores, como la humedad, gas, luz, etc.

Interfaz de comunicación: en caso de poder localizar a una persona y esta pueda comunicarse se tendrá una interfaz que permita realizar esto, con ayuda del micrófono y bocina implementados en el robot.

Asimismo, se realizan las notificaciones respectivas de cada interfaz, tanto de conexión perdida como de personas encontradas y notificaciones de la posible comunicación.

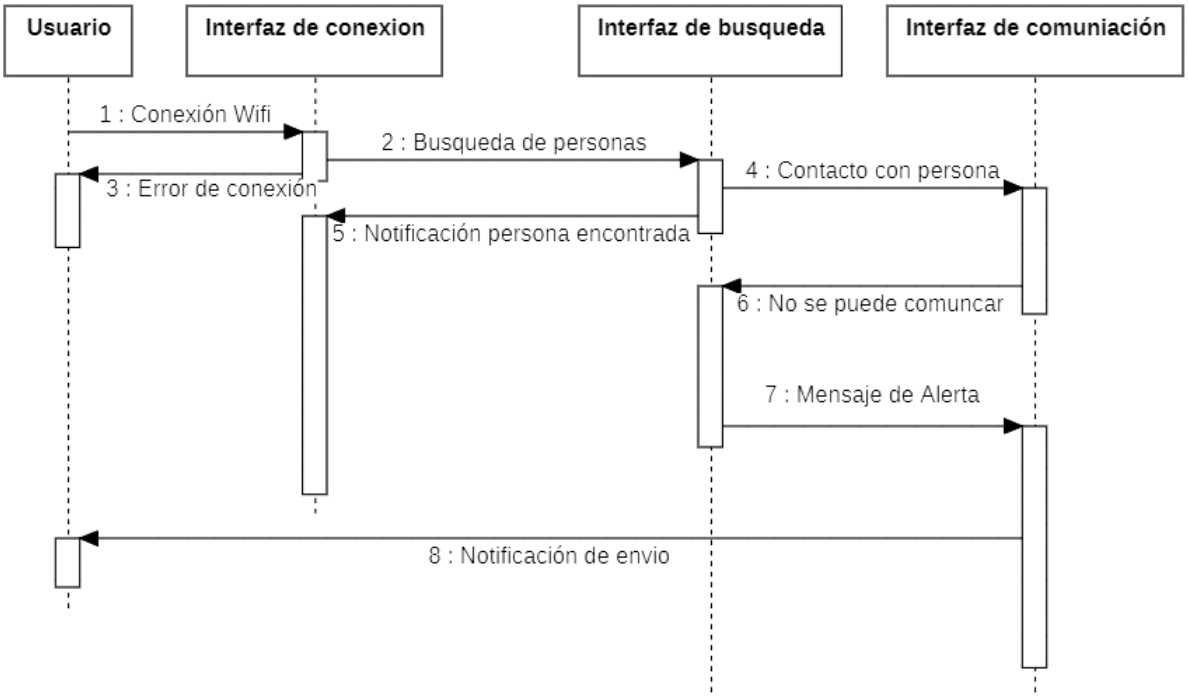

Figura 3.2.7: Diagrama de secuencia

3.2.6. Diagrama de flujo

El diagrama de flujo para la aplicación de seguimiento del robot (Figura 3.2.8) con el cual se visualiza de mejor manera el flujo que lleva el proceso de la interacción usuario-robot para con los individuos encontrados, así como el proceso de notificaciones del robot por medio de la aplicación.

Donde el robot primero establece una conexión con la aplicación móvil, para posteriormente comenzar a realizar la medición del medio en donde se obtienen los parámetros para poder determinar si puede haber personas en riesgo y si la hay realiza una comunicación mediante la interfaz de comunicación de la aplicación.

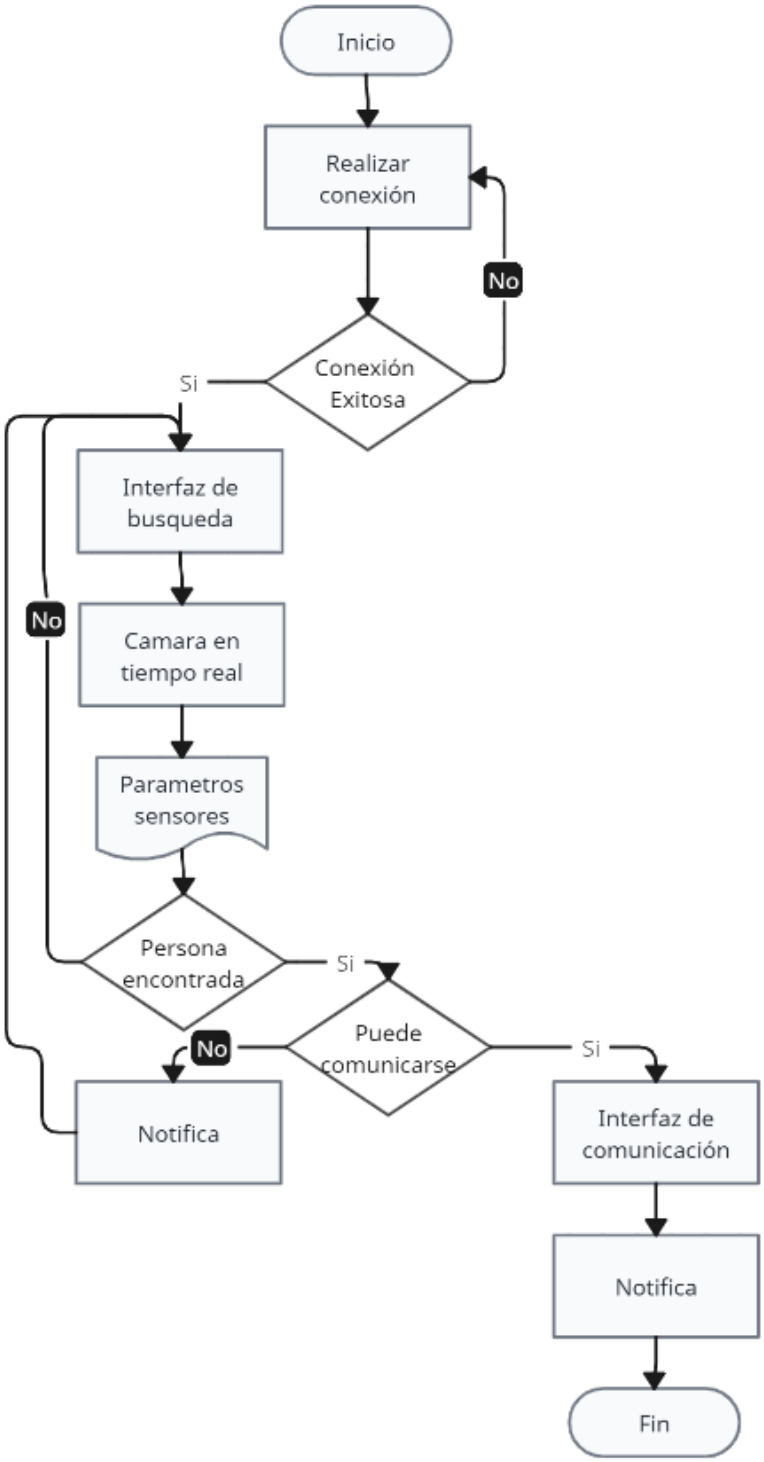

Figura 3.2.8: Diagrama de flujo de la aplicación

3.2.7. Diseño de aplicación

En este apartado se visualizan los diseños de las pantallas de la aplicación móvil, para el primero (Figura 3.2.9), se muestra en esta los datos recolectados por el prototipo (con los sensores), así como el video en tiempo real y el control para el robot mediante botones.

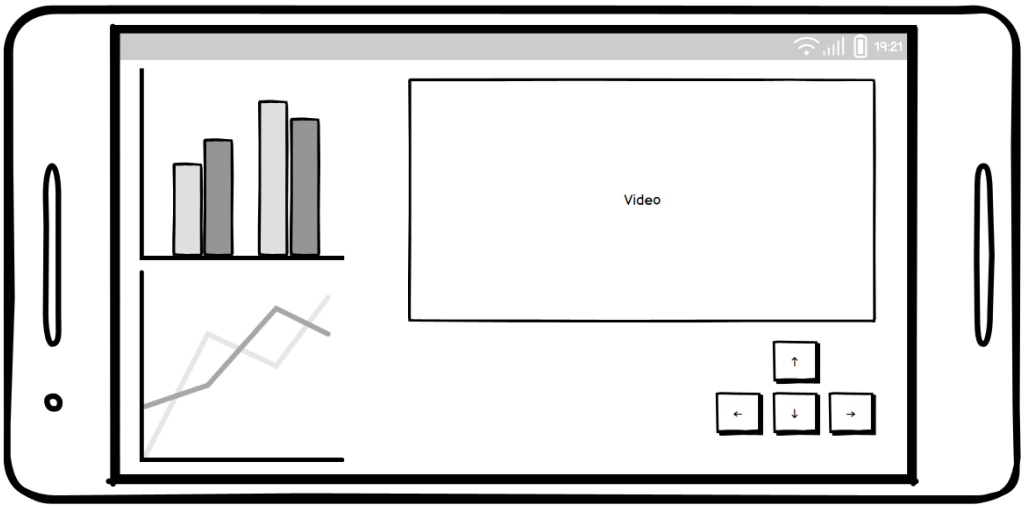

Figura 3.2.9: Pantalla de interfaz de control principal

Para la segunda pantalla (Figura 3.2.10), en caso de poder localizar personas poder realizar la comunicación mediante los micrófonos del dispositivo móvil y el prototipo.

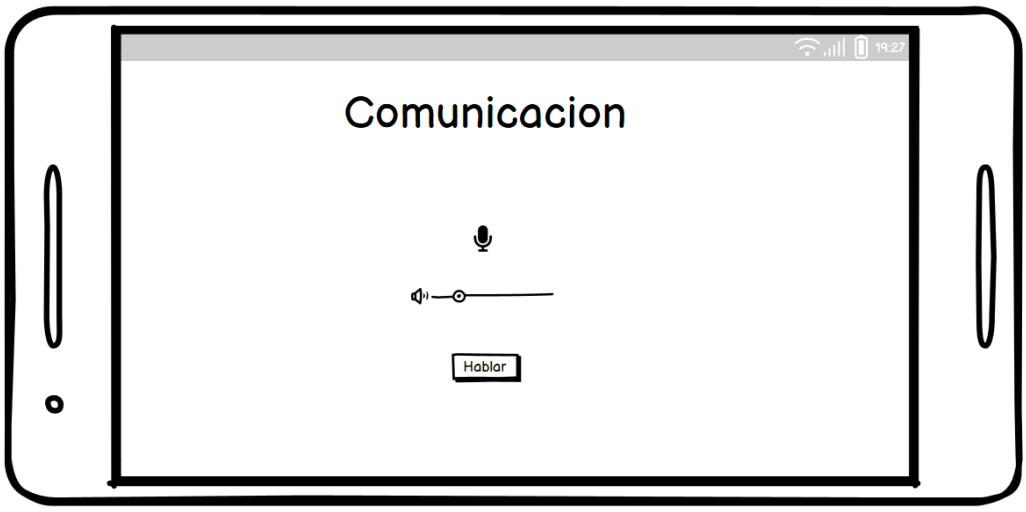

Figura 3.2.10: Pantalla de interfaz de comunicación

3.3. Arquitectura del sistema

La arquitectura de un sistema es la estructura o estructuras del sistema, que comprenden componentes de software, las propiedades externamente visibles de esos componentes, y las relaciones entre ellos [23].

El sistema está compuesto por dos motores controlados por la tarjeta L298N, sensores de temperatura, humedad, así como tres microcontroladores ESP32-CAM conectados mediante sockets a una aplicación móvil en donde se visualiza la información recolectada y se controla al robot.

La comunicación entre los componentes se realiza mediante sockets (figura 3.1.1) Un socket es la interfaz entre la aplicación y la capa de transporte dentro de un host. De manera análoga a una puerta, el socket actúa como el punto de entrada y salida a través del cual los datos fluyen hacia y desde la red hacia el proceso de aplicación [24]. Por lo que un socket es la puerta que tiene cada computadora, programa o tarjeta para poder comunicarse con dos o más dispositivos o usuarios a través de una red, como internet, cada usuario abre su “puerta” (socket) y con ello puede enviar o recibir datos, dicho de otra manera, es el punto de conexión que usa una computadora o programa para hablar con otro, ya sea en la misma maquina o con otra que se encuentre muy lejos. Asimismo, al poder crear una puerta por dispositivo se pueden evitar problemas de comunicación.

Ventajas de los sockets:

- Los sockets permiten **una conexión persistente** entre el cliente (ESP32) y el servidor (app Android).
- Permiten **enviar y recibir datos** sin necesidad de abrir y cerrar conexiones cada vez.
- los sockets TCP son **ligeros y eficientes**.

- Todo el sistema puede operar en una **red local** sin necesidad de acceso a Internet ni servidores en la nube.
- **Android permite usar sockets en hilos** (Thread, ServerSocket, etc.), manteniendo el rendimiento del interfaz fluido.
- La librería **WiFiClient de Arduino** facilita el uso de sockets TCP sin configuración compleja.

Estos sockets se basan en el protocolo de capa de transporte del modelo OSI, El Protocolo de Control de Transmisión (TCP) es un protocolo estándar en internet que garantiza la transmisión fiable de datos entre dispositivos en una red [25]. Este le da preferencia a la transmisión fiable de los datos, de modo que lleguen sin errores a su destino, esto al ser un servicio con conexión y con reconocimiento.

En el caso de este proyecto que se usan 3 controladores ESP32 (Figura 3.3.1) en donde cada ESP realiza una conexión mediante sockets con el dispositivo móvil, estos conectados a la misma red generada por un punto de acceso por el mismo dispositivo móvil (red Wi-Fi con el chip del móvil) y con ayuda de hilos, Un hilo es una unidad básica de utilización de CPU que forma parte de un proceso. Cada hilo tiene su propio contador de programa, conjunto de registros y pila, pero comparte la memoria y los recursos del proceso al que pertenece. Usar hilos permite que múltiples tareas se ejecuten concurrentemente dentro del mismo programa [26]. Por lo que permiten que un programa pueda realizar distintas funciones al mismo tiempo, cada ESP se encarga de funcionalidades específicas de por lo que:

- El robot escucha en su puerto 5000, 5001 y 5002
- ESP 1 se conecta → el robot crea un socket A.
- ESP 2 se conecta → el robot crea un socket B.
- ESP 3 se conecta → el robot crea un socket C.

De manera que cada ESP puede mandar información o recibir comandos del dispositivo móvil en su socket correspondiente, el primero envía video en tiempo

real, el segundo recibe comandos para movilidad y el tercero envía información de los sensores, todo esto sin colisiones o perdida de información.

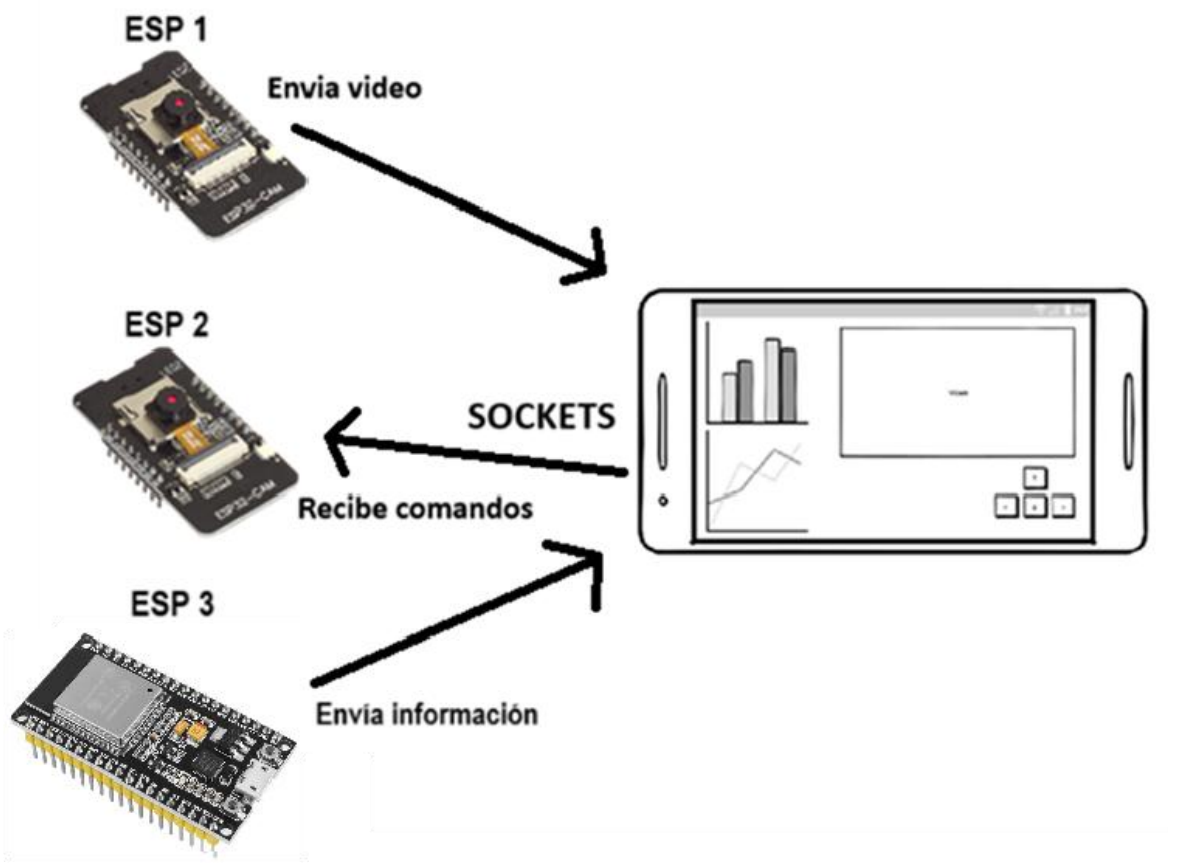

Figura 3.3.1: Comunicación del sistema

3.4. Selección de hardware y software

3.4.1. Microcontroladores

- **ESP32-CAM**: permite conectividad inalámbrica y un procesamiento rápido de los datos de los sensores, así como su conectividad Wi-Fi, el uso de la cámara integrada y su bajo consumo energético (Figura 2.2.10)
- **ESP32-DEVKIT V1:** de igual manera que la ESP32-CAM tiene conectividad inalámbrica y procesamiento veloz, la diferencia está en la perdida de la

cámara, pero ganando más puertos usables para la conexión de los sensores que miden el medio (Figura 3.4.1)

Figura 3.4.1: ESP32-DEVKIT

- **PUENTE H L298:** este componente se encarga de controlar los motores del robot, de modo que es un intermediario entre las ESP32 y la aplicación móvil que controla el movimiento del robot (Figura 3.4.2)

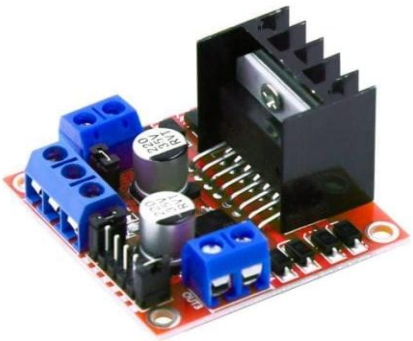

Figura 3.4.2: Puente H

3.4.2. Sensores utilizados

- **Sensor MLX90614**: Es un sensor de temperatura sin contacto que mide la temperatura a través de radiación infrarroja, además de tener un precio relativamente económico e ideal para medir temperatura corporal a distancia (Figura 3.4.3)

Figura 3.4.3: Sensor de temperatura a distancia

- **Sensor de temperatura DHT11**: Detecta temperatura, así como humedad del ambiente (Figura 2.2.4)
- **Sensor de luz**: (fotorresistor) su resistencia varía dependiendo de la intensidad de la luz en el ambiente (Figura 2.2.6)

3.4.3. Actuadores

- **Motores DC**: 2 motores unidireccionales con voltaje de 9 a 12 volts con ruedas tipo oruga para poder moverse por terrenos irregulares (Figura 3.4.4).

Figura 3.4.4: Motor DC

3.4.4. Tecnologías de conectividad

Los protocolos de comunicación remota utilizados son:

- **Wi-Fi (IEEE 802.11 b/g/n)**: Permite conectar el ESP32-CAM a una red inalámbrica, así como poder enviar datos o video a servidores o recibir comandos.

- **Sockets**: es un punto final de comunicación bidireccional que permite que dos aplicaciones puedan intercambiar información mediante una red ya sea local o remota.

3.4.5. Plataforma de control y visualización

Se desarrolló una aplicación móvil con Android y java para poder controlar el movimiento del robot, así como el estado de este y del entorno, la interfaz se puede visualizar en la siguiente imagen (Figura 3.4.5).

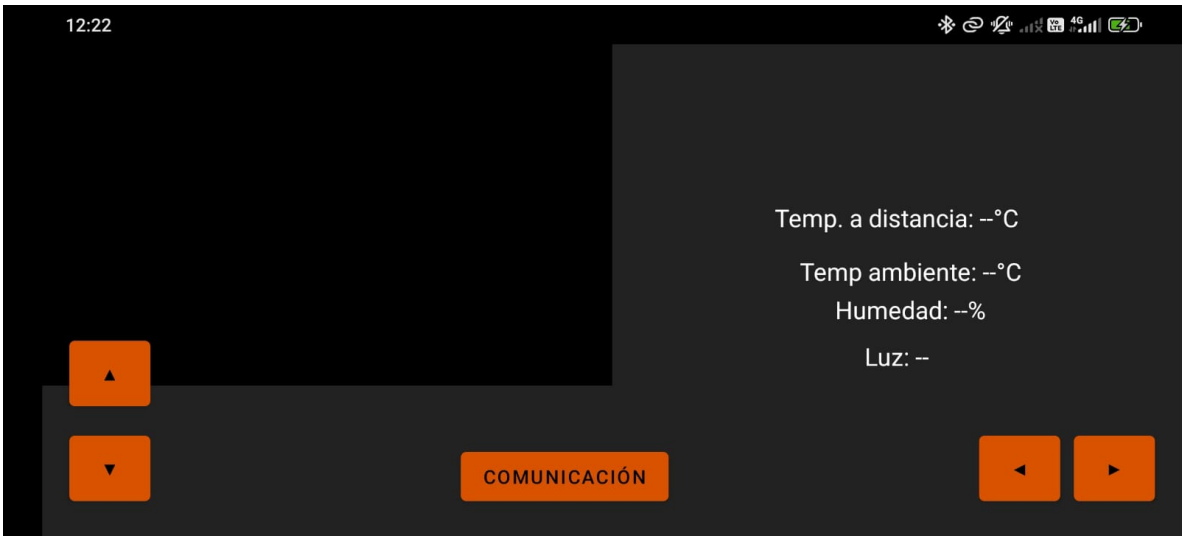

Figura 3.4.5: Interfaz principal de control del robot

3.5. Implementación del sistema

Para el control y conectividad del robot se desarrolló una aplicación en Android desde cero, esto con elementos importantes para que el manejo del robot sea intuitivo para el usuario final.

Permitiendo interfaces de control sencillas, así como alertas visuales y mensajes importantes en pantalla.

3.5.1. Interfaz de visualización

La aplicación cuenta con dos interfaces para el control y monitoreo del robot, la primera (Figura 3.4.5) es la interfaz principal en la cual se visualizan los datos del medio capturados por el robot, como lo son la temperatura, humedad, luz, etc. Asimismo se tienen los botones del control del robot con los cuales se maneja el movimiento del robot y por último en la parte superior derecha la visualización de la cámara en tiempo real del robot.

La segunda interfaz será para el manejo de la comunicación por voz de los usuarios, esto permitiendo con ayuda de grabaciones de voz, que exista una comunicación eficaz entre los usuarios que se encuentran fuera del siniestro con los que están dentro (Figura 3.5.1)

Figura 3.5.1: Interfaz para la comunicación por voz

3.5.2. Comunicación

Para la comunicación del robot con la aplicación móvil se hace uso de un sistema distribuido, el cual es una colección de computadoras independientes que aparenta ante los usuarios finales del sistema como una única computadora coherente [27].

Esto se puede al paralelizar cada función de cada esp32 con ayuda de hilos, en la obra de Tanenbaum se menciona que es **la unidad más pequeña de utilización de CPU**, que posee su propio contador de programa, registros y pila [28]. Por lo que se puede decir que es una unidad de ejecución independiente dentro de una aplicación, programa o sistema.

También con ayuda de los sockets y el protocolo TCP se puede hacer la comunicación de modo que cada esp32 es un cliente con ayuda de la biblioteca WifiClient [29]. De modo que cada ESP32:

1. Se conecta a la misma red Wi-Fi del servidor
2. Usa WiFiClient client; para intentar conectarse al IP del celular en el puerto abierto por el ServerSocket.
3. Si la conexión se logra, puede enviar datos usando client.print() o client.println(), visto en el siguiente código (Figura 3.5.2).

```
void reconnect() {
    Serial.println("Intentando conectar al servidor...");
    if (client.connect(serverIP, serverPort)) {
        Serial.println("Conectado al servidor");
    } else {
        Serial.println("Fallo en la conexión");
    }
}
```

Figura 3.5.2: Código para la conexión y comunicación (wificlient)

Para el caso del servidor, el celular se convierte en el con ayuda de los datos móviles de cualquier compañía de comunicaciones, esto con la biblioteca ServerSocket [30], iniciada en el código del programa (Figura 3.5.3)

1. La aplicación en Android abre un puerto TCP para cada canal de comunicación (uno por cada ESP32) por ejemplo el 5000, 5001 y 5002
2. Luego el servidor escucha continuamente solicitudes de conexión de algún cliente (ESP32)
3. Cuando una solicitud de conexión llega y es aceptada por el servidor entonces se acepta con *.accept()* y se crea un canal de comunicación.

```
new Thread(() -> {
    try {
        ServerSocket serverSocket = new ServerSocket(SERVER_PORT);
        socket = serverSocket.accept();
        outputStream = socket.getOutputStream();
```

Figura 3.5.3: Código de conexión para la aplicación Android (ServerSocket)

La funcionalidad final entre ambos elementos se puede ver en la Tabla 3.5.1

Tabla 3.5.1: Comunicación entre prototipo y aplicación

Android (ServerSocket)	**ESP32 (WiFiClient)**
Abre un puerto TCP	Se conecta al puerto
Espera conexiones	Inicia la conexión
Recibe datos por socket	Envía datos por socket

3.5.3. Paralelización

Para evitar problemas de colisión de los datos recibidos o que sea necesario detener el robot para activar el uso de los sensores y evitar que sea de manera secuencial con ayuda de hilos en Android se paraleliza las funciones de cada tarjeta ESP32 de manera que trabajan de manera independiente una de otra, teniendo un canal de comunicación diferente para cada uno con los sockets, el programa consta de 4 hilos (Tabla 3.5.2):

Tabla 3.5.2: Hilos en Android

Hilo (Thread)	Método donde se inicia	Función principal
UI/Main Thread	onCreate()	Muestra la interfaz gráfica, actualiza TextViews, responde a clics de usuario.
setupServer()	new Thread(...)	Abre un ServerSocket para recibir temperatura corporal de la ESP32-CAM.
setupDHTServer()	new Thread(...)	Recibe temperatura ambiente, humedad y presión desde la ESP32 DevKit por sockets.
sendCommand()	new Thread(...)	Envía comandos de movimiento a la ESP32-CAM (arriba, abajo, etc.).

3.5.4. Detección de personas por medio de calor

La función de detección de calor se realiza con ayuda del sensor MLX90614 el cual es un sensor infrarrojo que detecta calor a distancia y que con ayuda de la programación en Android con el hilo 1 (Tabla 3.5.2) de manera independiente a las demás ESP32 obtiene toda la información del sensor y lo envía por el socket al servidor, este a su vez realiza lo siguiente para crear un mensaje de alerta (un punto rojo) en la visualización de la cámara en pantalla.

El **punto rojo** (Figura 3.5.4) en la interfaz de la aplicación Android **es un indicador visual de alerta** que se activa cuando se detecta una **temperatura corporal anormalmente alta o baja** desde el sensor conectado al ESP32-CAM.

Figura 3.5.4: Punto rojo en la cámara

El proceso es el siguiente:

1. El ESP32-CAM (2) está conectado a un sensor de temperatura corporal (MLX90614).
2. Este ESP32-CAM mide la temperatura y la envía mediante sockets (TCP) a la aplicación Android.
3. En la aplicación Android, el método setupServer() crea un ServerSocket, el cual recibe constantemente los datos de temperatura enviados por el ESP32-CAM.
4. Cada vez que se recibe una nueva lectura de temperatura, se verifica si la temperatura supera un umbral crítico:
5. Si se supera el umbral (por ejemplo, **33.0 °C**), se **muestra el punto rojo**, indicando una posible situación de emergencia, como una persona con fiebre o una fuente de calor peligrosa.

Este proceso se basa en la función setupServer() la cual se puede ver en el siguiente código (Figura 3.5.5)

```
private void setupServer() {
    new Thread(() -> {
        try {
            ServerSocket serverSocket = new ServerSocket(SERVER_PORT);
            socket = serverSocket.accept();
            outputStream = socket.getOutputStream();
            BufferedReader reader = new BufferedReader(
                new InputStreamReader(socket.getInputStream())
            );

            runOnUiThread(() ->
temperaturaTextView.setText("Conectado"));

            String line;
            while ((line = reader.readLine()) != null) {
                float temperatura = Float.parseFloat(line.trim());
                runOnUiThread(() -> updateTemperature(temperatura + 5));
            }

        } catch (Exception e) {
            e.printStackTrace();
            runOnUiThread(() -> temperaturaTextView.setText("Error de
conexión"));
        }
    }).start();
```

Figura 3.5.5: Código para la visualización del punto rojo en pantalla

¿Por qué es útil este punto rojo?

- Sirve como una alerta visual inmediata para el usuario sin necesidad de leer los valores que se encuentran a un costado, evitando distracciones.
- Es intuitivo ya que el color rojo se asocia normalmente a peligro o alerta.
- Puede permitir una asistencia remota más eficaz, de modo que el operador que ve la aplicación puede actuar con mayor rapidez si se detecta una persona con fiebre en un entorno de desastre.

3.5.5. Comunicación por voz

Una funcionalidad muy importante en situaciones de siniestros es el poder comunicarse directamente con las personas que se encuentran dentro del escenario de riesgo, esto ya que podría permitir al personal calificado conocer mejor el entorno, ya que se tendría un contexto más de lo que sucede desde dentro del siniestro.

Esta funcionalidad se implementa con el robot, de modo que no solo tramite video y datos del medio (sensores), sino que además permite comunicación por voz en tiempo real, permitiendo así ampliar sus capacidades en el apoyo de rescate de personas.

Arquitectura de comunicación: la comunicación a diferencia que con el video y los datos de los sensores que se usan TCP sockets, está basada en UDP sockets para audio en tiempo real esto ya que cuenta con una menor latencia que TCP.

Se hace uso de dos canales, para evitar conflictos con los tipos de datos que se deben recibir en los dispositivos:

- Canal de comandos **(puerto 3333)** → para iniciar/detener grabación y reproducción.
- Canal de audio **(puerto 4444)** → para transmitir las muestras de voz.

Hardware utilizado: los componentes físicos utilizados para permitir la comunicación por voz fueron los siguientes:

Tarjeta ESP32 devkit v1(Figura 3.5.6), ya que a diferencia de las tarjetas ESP32-CAM esta cuenta con DAC (Convertidor Digital Analógico), además de ser el cerebro y capturar audio del micrófono (I2S) y enviar el audio al celular mediante UDP con sockets, además de recibir audio desde el celular y reproducirlo en la bocina.

Figura 3.5.6: Tarjeta ESP32 Devkit V1

Asimismo, se utilizó el **amplificador digital MAX98357**(Figura 3.5.7) para poder convertir las señales de audio I2S que envía la ESP32 en una señal analógica y amplificada que pueda reproducir directamente la bocina.

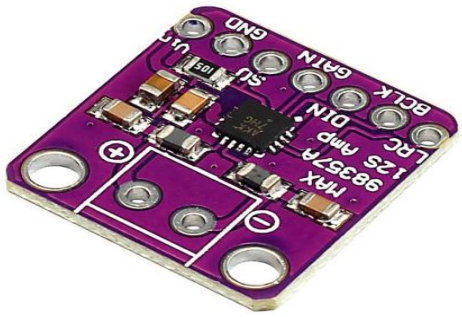

Figura 3.5.7: Amplificador MAX98357

Micrófono INMP441 (Figura 3.5.8) captura la voz o sonidos comunicándose con la ESP32 y usando I2S.

Figura 3.5.8: Micrófono INMP441

Y finalmente una **bocina** (Figura 2.1.10) siendo el dispositivo final donde se reproduce el audio que llega desde el celular a la ESP32.

Software para la comunicación

Para realizar la comunicación entre el celular y el robot se desarrolló un apartado extra en la Aplicación Android (Figura 3.5.9) que usa la API de AudioRecord y AudioTrack para grabar y reproducir.

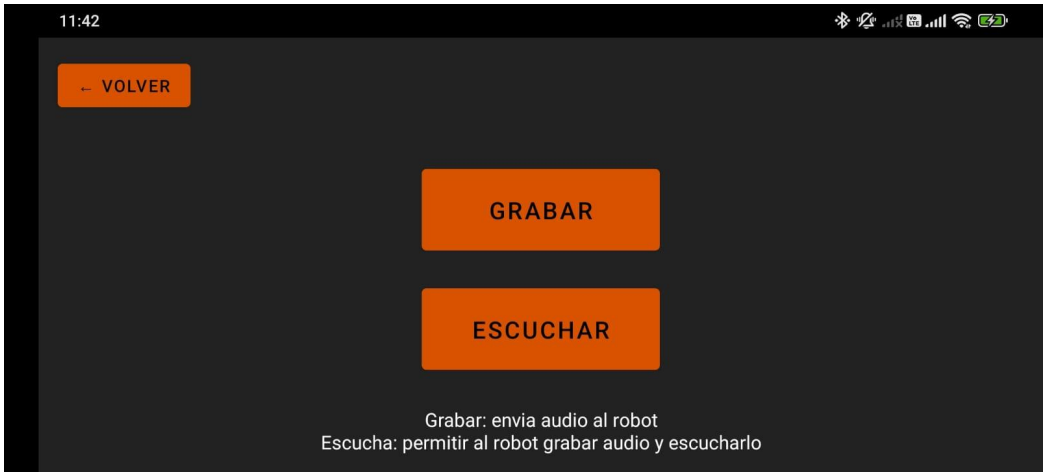

Figura 3.5.9: Interfaz de comunicación por voz

El Funcionamiento de la comunicación entre los dispositivos se puede visualizar en la siguiente tabla (Tabla 3.5.3)

Tabla 3.5.3: Funcionamiento de la comunicación por voz

Acción	**Respuesta**
El usuario presiona “GRABAR” en la aplicación	El audio del micrófono del celular se envía a la ESP32 y lo reproduce
El usuario presiona una segunda vez “GRABAR”	Se detiene él envió de audio a la ESP32
El usuario presiona “ESCUCHAR” en la aplicación	La ESP32 envía el audio capturado por su micrófono al celular.
El usuario presiona por segunda vez “ESCUCHAR”	Se detiene la recepción de audio desde la ESP32

El funcionamiento desde el robot con la ESP32 es la siguiente:

Primero se configura el periférico I2S del ESP32, luego se define que trabajará como maestro, capaz de recibir audio (RX, micrófono) y enviar audio (TX, bocina), la frecuencia de muestreo es de 16 kHz y la resolución de 16 bits, suficiente para comunicación de voz clara visto en el siguiente código (Figura 3.5.10).

```
i2s_config_t i2s_config = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX | I2S_MODE_TX),
    .sample_rate = 16000,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
    .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
};
i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);
```

Figura 3.5.10: Código I2S ESP32

Para la transmisión del micrófono hacia la aplicación móvil, se capturan datos desde el micrófono con i2s_read, los datos se envían en un paquete UDP hacia el celular (IP y puerto previamente definidos), esto permite que el usuario escuche lo que capta el micrófono del robot en tiempo real (Figura 3.5.11).

```
if (enableTx) {
    i2s_read(I2S_PORT, samples, sizeof(samples), &bytesRead,
portMAX_DELAY);
    udpAudio.beginPacket(phoneIP, audioPort);
    udpAudio.write((uint8_t*)samples, bytesRead);
    udpAudio.endPacket();
}
```

Figura 3.5.11: Código para la transmisión de audio ESP32

Para la recepción (Figura 3.5.12) el ESP32 escucha en el puerto UDP si hay datos entrantes, si recibe audio desde el celular, se escribe directamente en la salida I2S, reproduciéndose en la bocina del robot, con esto el usuario puede hablar al robot y que su voz se reproduzca en el altavoz.

```
if (enableRx) {
    int len = udpAudio.read((char*)buffer, sizeof(buffer));
    i2s_write(I2S_PORT, buffer, len, &bytesWritten, portMAX_DELAY);
}
```

Figura 3.5.12: Código para la recepción audio ESP32

Asimismo, el funcionamiento de la comunicación desde el celular (aplicación móvil) sigue lo siguiente:

Él envió de audio se realiza desde el celular al robot donde se inicializa AudioRecord para capturar el micrófono del celular, los datos grabados se guardan en un buffer y se envían por paquetes UDP al ESP32, esto activa el modo "Grabar" en la app, permitiendo que el robot escuche la voz del usuario (Figura 3.5.13).

```
AudioRecord recorder = new AudioRecord(MediaRecorder.AudioSource.MIC,
    SAMPLE_RATE, AudioFormat.CHANNEL_IN_MONO,
    AudioFormat.ENCODING_PCM_16BIT, BUFFER_SIZE);

while (isRecording) {
    int read = recorder.read(buffer, 0, buffer.length);
    socket.send(new DatagramPacket(buffer, read, espAddress,
AUDIO_PORT));
}
```

Figura 3.5.13: Código para él envió de audio aplicación móvil.

Para la recepción de audio desde la ESP32 se inicializa AudioTrack para reproducir el audio en el altavoz del celular, el programa se queda escuchando datos entrantes (UDP) que vienen del micrófono del ESP32, cuando llegan, se reproducen inmediatamente, permitiendo al usuario escuchar lo que el robot capta en su entorno (Figura 3.5.14).

```
AudioTrack speaker = new AudioTrack(AudioManager.STREAM_MUSIC,
    SAMPLE_RATE, AudioFormat.CHANNEL_OUT_MONO,
    AudioFormat.ENCODING_PCM_16BIT, BUFFER_SIZE, AudioTrack.MODE_STREAM);

while (isListening) {
    socket.receive(packet);
    speaker.write(packet.getData(), 0, packet.getLength());
}
```

Figura 3.5.14: Código para la recepción de audio en la aplicación móvil.

En resumen, la tarjeta ESP32 captura y reproduce audio usando el periférico I2S y transmite/recibe mediante sockets UDP, asimismo, la aplicación móvil usa las clases ***AudioRecord*** y ***AudioTrack*** para grabar/reproducir, enviando y recibiendo también por UDP, permitiendo que juntos formen un canal de comunicación bidireccional de voz en tiempo real.

CAPITULO 4
PRUEBAS Y RESULTADOS

En este capítulo se presentan los resultados con base en la funcionalidad del prototipo, evaluando apartados importantes como lo son: la conectividad, la resistencia, la integración y el rendimiento.

4.1. Estrategia de pruebas

Una **estrategia de pruebas** es un documento de alto nivel que define el enfoque general de pruebas dentro del ciclo de desarrollo de software. Establece los objetivos, métodos, herramientas, responsables y recursos necesarios para asegurar que el producto cumpla con los requisitos de calidad [31].

Dentro de las distintas pruebas para comprobar la funcionalidad del robot se tienen las funcionales para validar la movilidad del robot, así como la de integración, que evalúa que el robot funciona correctamente con los distintos módulos con los que el prototipo cuenta, en la siguiente Tabla 4.1.1 se pueden ver esos distintos tipos de pruebas a las que se sometió el robot.

Tabla 4.1.1: Tipos de prueba

Tipo de prueba	**Descripción**
Pruebas funcionales	Verifican que el robot cumple con lo que debe hacer: moverse, detectar, enviar datos.
Pruebas de integración	Verifican que los módulos (ESP32, sensores, app) funcionen bien juntos.
Pruebas de rendimiento	Evalúan el tiempo de respuesta, consumo de energía, velocidad de transmisión.
Pruebas de conectividad	Validan la estabilidad de la comunicación inalámbrica en distintos entornos.
Pruebas de resistencia o estrés	Observan el comportamiento del sistema ante cargas altas o fallos.

4.2. Pruebas funcionales

En esta etapa se validan las funcionalidades principales del robot, si este se mueve correctamente, captura la información mediante los sensores y envía esa información a la aplicación móvil, en la Figura 4.2.1 se puede observar el resultado obtenido desde la aplicación con la lectura de los sensores (temperatura a distancia, temperatura y humedad del ambiente y la luz del mismo) y el video en tiempo real capturado por el robot.

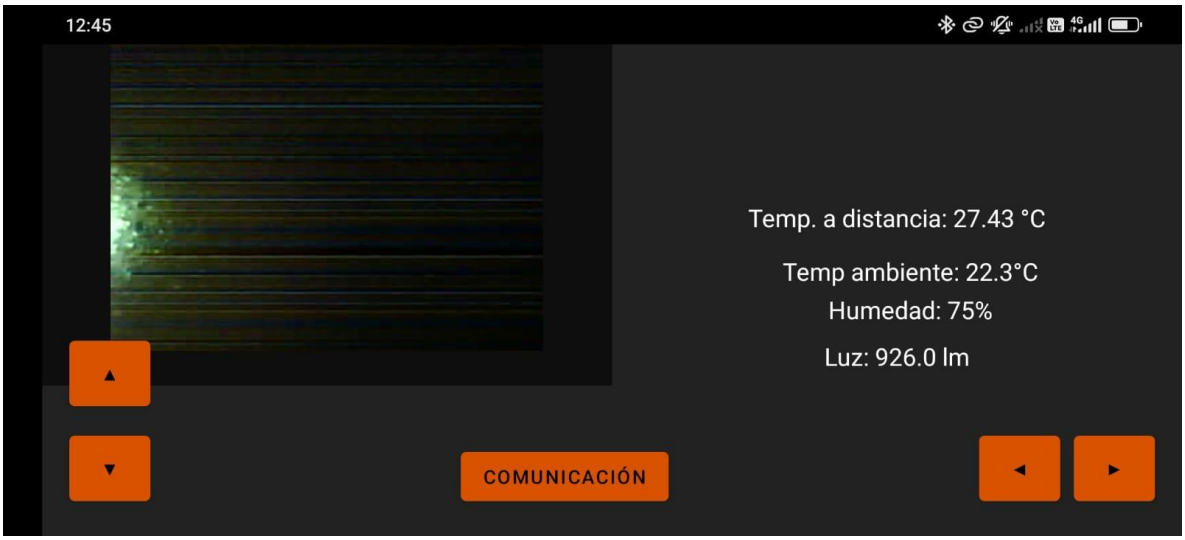

Figura 4.2.1: Funcionalidad prototipo

De acuerdo con esa funcionalidad se puede visualizar de mejor manera en la siguiente Tabla 4.2.1, donde se mide el comportamiento del robot en aspectos como: el movimiento del robot, detección de obstáculos, lectura de temperatura corporal y ambiente y comunicación por voz.

Para la detección de obstáculos se usó un sensor ultrasónico el cual al detectar un objeto a 10 cm de distancia se detiene y envía una notificación a la aplicación la cual se visualiza en tiempo real (Figura 4.2.2)

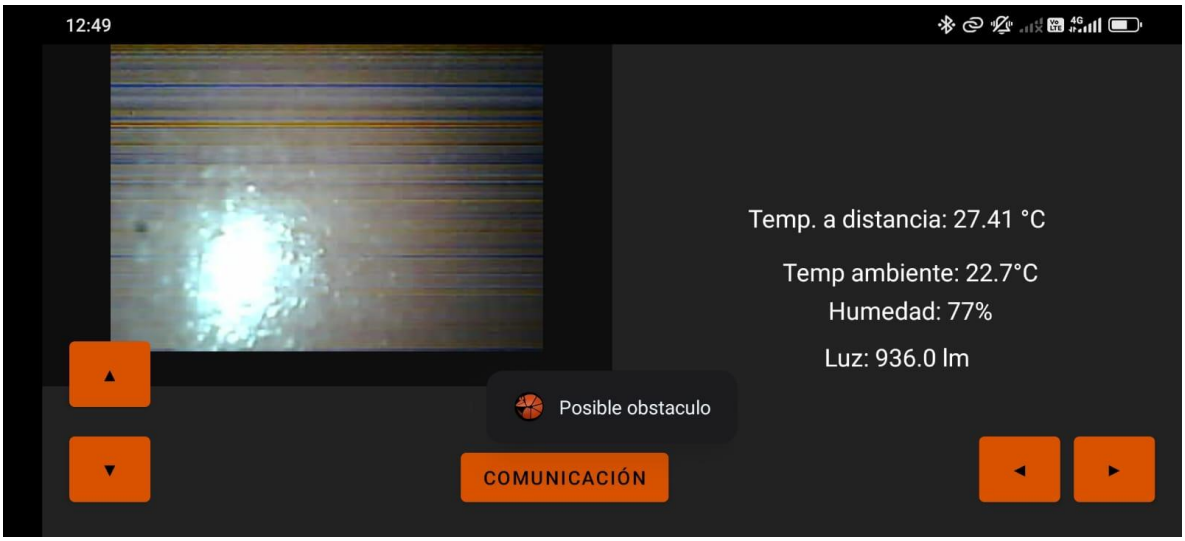

Figura 4.2.2: Detección de obstáculos

Asimismo, para la detección de calor a distancia se pueden visualizar los resultados de manera grafica en las siguientes imágenes, la primera (Figura 4.2.3) se muestra la detección de calor corporal de una mano, mostrando un punto rojo y una notificación.

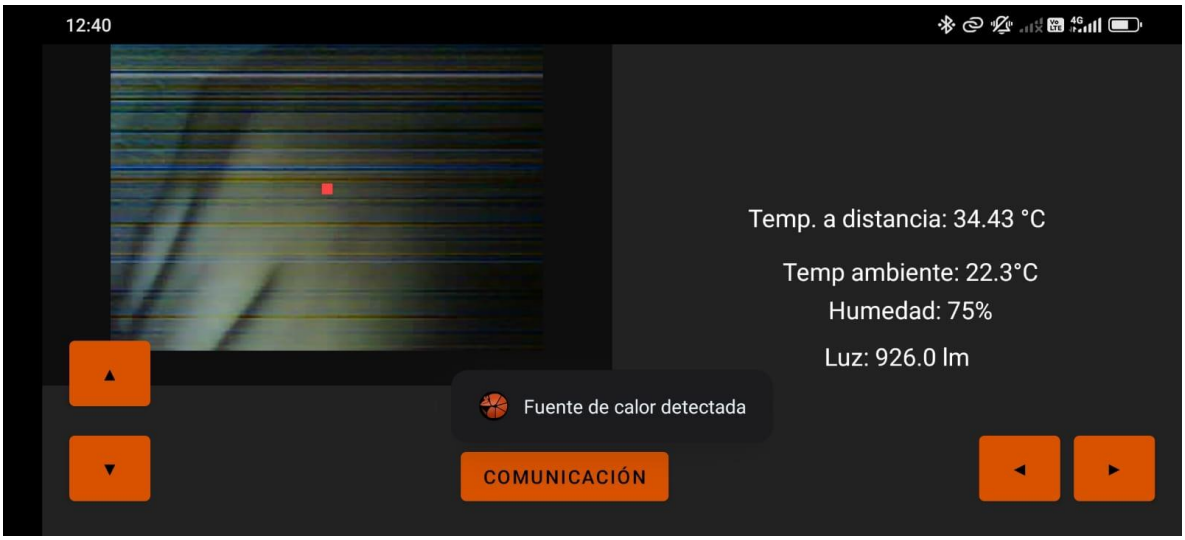

Figura 4.2.3: Calor corporal

La segunda imagen (Figura 4.2.4) muestra una fuente de calor distinta, en este caso fuego, el cual el robot detecta y muestra en pantalla.

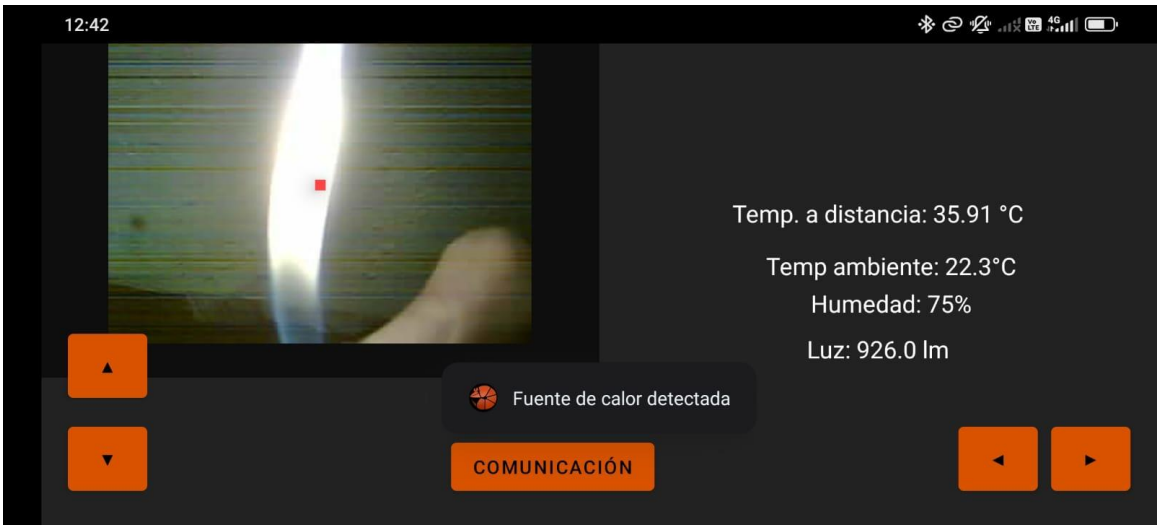

Figura 4.2.4: Fuente de calor externa

La comunicación por voz se realizó mediante dos botones, uno para la transmisión de la voz, en la siguiente imagen (Figura 4.2.5) se visualiza como al presionar este, se muestra una notificación.

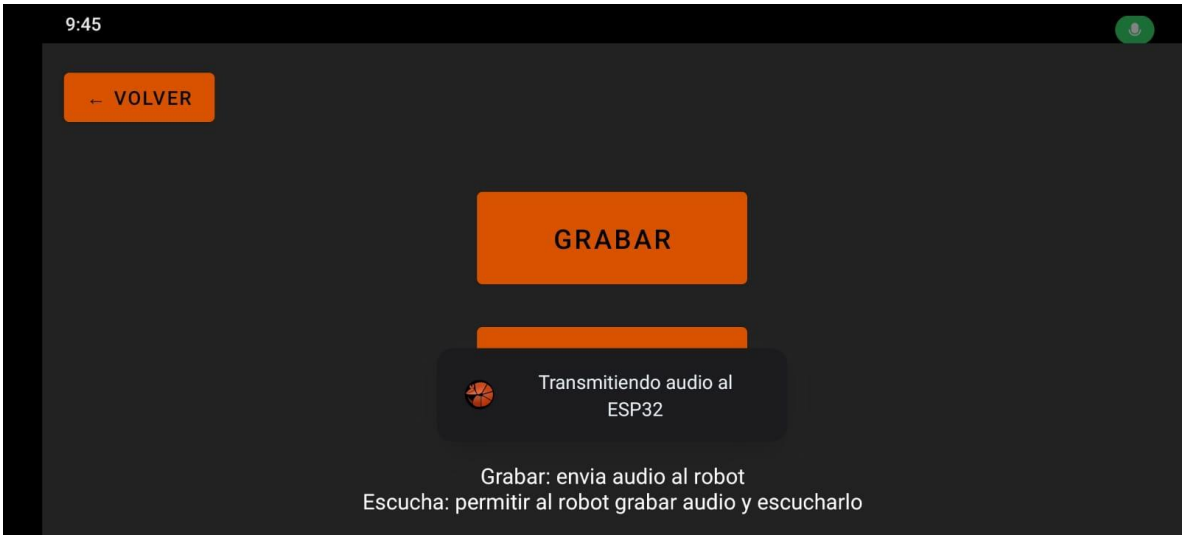

Figura 4.2.5: Transmisión de audio

El segundo botón realiza la función inversa, que es recibir el audio que recopila el robot y de igual manera al activarlo se notifica en pantalla (Figura 4.2.6).

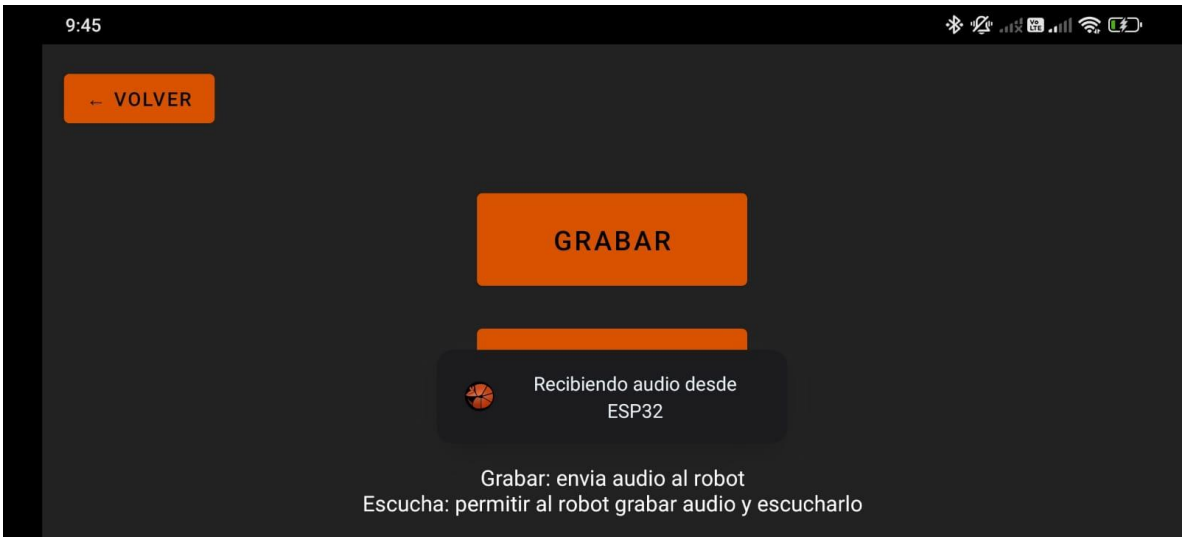

Figura 4.2.6: Recepción de audio.

Teniendo como resultados un rendimiento correcto por parte del robot en las funciones mencionadas.

Tabla 4.2.1: Pruebas funcionales

Función probada	**Descripción**	**Resultado esperado**	**Resultado obtenido**	**Estado**
Movimiento del robot	Al presionar botones en la aplicación, el robot debe moverse en la dirección indicada.	Robot responde mientras se presiona el botón en la dirección correcta	Responde con una latencia de: **100–300 ms.**	Correcto
Detección de obstáculos	El sensor ultrasónico detiene el robot al detectar un objeto a < 10 cm.	Los motores se detienen automáticamente.	Se detiene a ~8 cm.	Correcto
Lectura de temperatura corporal	La ESP32-CAM 2 recibe temperatura y la muestra en la aplicación.	Visualización en °C con punto rojo si se supera el umbral de 33°C.	Muestra 34°C y se activa el punto rojo.	Correcto
Lectura de ambiente	La ESP32 DevKit envía humedad, temperatura y luz.	Se visualizan los datos en tiempo real en la aplicación.	Actualiza cada 2 s.	Correcto
Comunicación por voz	Envío y recepción de audio por UDP entre el robot y la aplicación.	Audio reproducible en ambas direcciones.	Retardo de entre ~150-700 ms, buena calidad.	Correcto

4.3. Pruebas de integración

En las siguientes pruebas (Tabla 4.3.1) se determina que todos los componentes del prototipo funcionen correctamente.

Tabla 4.3.1: Pruebas de integración

Módulos integrados	**Objetivo de la prueba**	**Resultado esperado**	**Resultado obtenido**	**Estado**
ESP32-CAM + puente H + motores	Control de movimiento en tiempo real.	Movimiento del robot sin interferir en los demás componentes	Movimiento correcto al mismo tiempo que se lee el ambiente	Correcto
ESP32 DevKit + DHT11 + App	Sincronización de datos ambientales vía sockets.	Recepción continua sin pérdidas	Recepción de la información de manera continua.	Correcto
ESP32-CAM + App Android	Transmisión de video sin interrupciones.	Video en tiempo real con visibilidad.	Video continuo, pero con calidad mediana	Aceptable
Módulo de voz + App Android	Comunicación bidireccional.	Transmisión en tiempo real audible.	Transmisión correcta, aunque con un retardo de ~0.5s. y audio calidad media	Correcto

4.4. Pruebas de rendimiento y conectividad

Se mide en valores cuantitativos la funcionalidad del robot en cuestión de latencia y autonomía. Visto en la Tabla 4.4.1.

Tabla 4.4.1: Conectividad

Métrica	Descripción	Resultado obtenido	Observaciones	Estado
Latencia promedio	Tiempo entre comando y acción (Android ↔ ESP32)	20–30 ms.	Medida mediante timestamp en sockets TCP.	Aceptable
Latencia UI → Motor	Tiempo total entre la acción del usuario en la aplicación y el movimiento físico del robot.	100–300 ms.	Incluye procesamiento de app, transmisión Wi-Fi y respuesta mecánica.	Aceptable
Autonomía (movimiento normal)	Duración promedio con operación moderada de sensores y motores.	≈ 90 min.	Calculado midiendo tiempos de operación del robot en el entorno	Aceptable
Autonomía (uso intensivo)	Duración bajo operación continua de motores y transmisión constante (figura 4.4.1).	≈ 30–45 min.	Depende del tipo de terreno y actividad de los sensores.	Aceptable

Asimismo, otras métricas sobre la resistencia del robot en cuestión de su operabilidad como el alcance y la tolerancia a fallos se pueden ver en la siguiente Tabla 4.4.2.

Tabla 4.4.2: Rendimiento

Métrica	Descripción	Resultado obtenido	Observaciones	Estado
Alcance operativo (interior)	Distancia máxima con señal Wi-Fi estable dentro de edificaciones (figura 4.4.2).	20–40 m	Disminuye por interferencias, paredes o materiales metálicos.	Aceptable
Alcance operativo (exterior)	Distancia máxima con línea de vista directa (figura 4.4.3).	40–60 m	Calculada en un entorno controlado en línea recta sin obstáculos.	Aceptable
Tolerancia a fallos	Capacidad del sistema de seguir operando ante desconexiones parciales.	Funcionalidad degradada pero operativa	Si un nodo falla, los demás continúan enviando datos o control.	Aceptable
Tiempo de reconexión automática	Tiempo promedio que tarda en restablecerse la conexión tras una pérdida temporal.	2–10 s	Depende del tipo de reinicio y calidad de la red Wi-Fi.	Aceptable

Para el uso intensivo del robot (Figura 4.4.1) se puso a prueba bajo circunstancias adversas como barrancas, baches, sobre charcos no muy profundos, por lo que los motores fueron exigidos de manera constante permitiendo validar la autonomía y la resistencia bajo estas situaciones.

Figura 4.4.1: Prueba de prototipo en uso intensivo

De igual manera para validar el alcance operativo del robot en interiores se puso a prueba dentro de una edificación, con paredes y objetos como obstáculos, permitiendo validar que opera correctamente, aunque con limitaciones en el alcance máximo del robot, visto en la Figura 4.4.2.

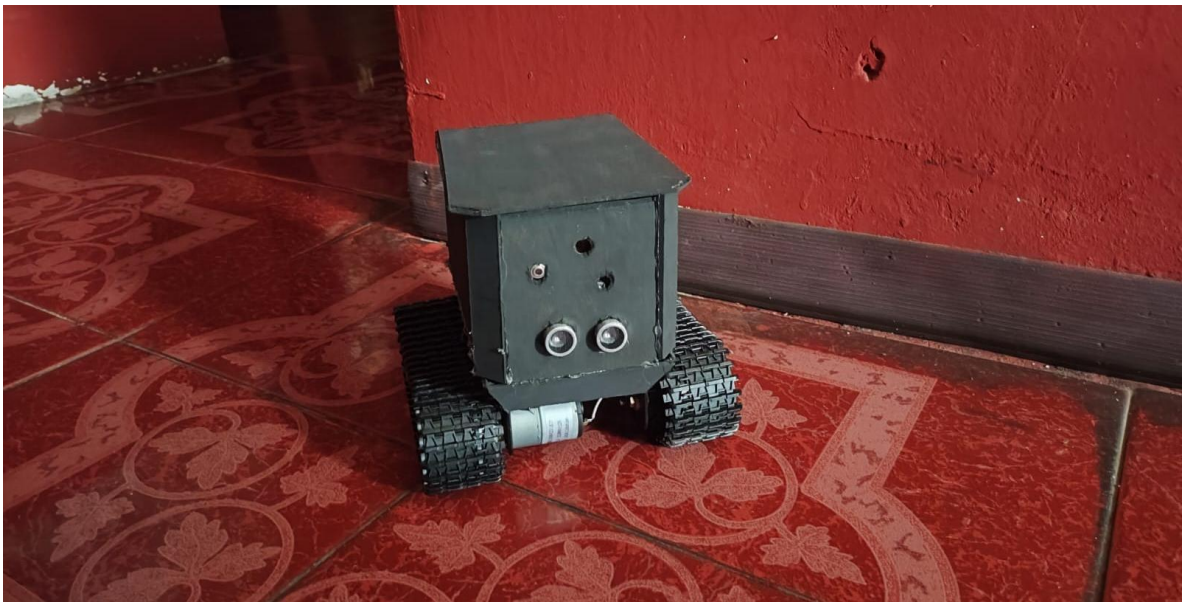

Figura 4.4.2: Prueba de prototipo en operación en el interior

Para el otro caso en cuestión de la operatividad del robot, pero en exteriores, se puso a prueba en una calle recta, validando la distancia máxima del robot (con movimiento frontal constante), esto sin obstáculos (Figura 4.4.3).

Figura 4.4.3: Prueba de prototipo en operación en el exterior

4.5. Pruebas de resistencia

Y finalmente tomando en cuenta situaciones de riesgo donde operará el robot se tienen la prueba de resistencia vistos en la siguiente Tabla 4.5.1.

Tabla 4.5.1: Resistencia

Escenario	**Descripción**	**Resultado esperado**	**Resultado obtenido**	**Estado**
Uso continuo 30 min	Movimiento + sensores + transmisión video	Sin reinicios ni bloqueos.	Sin fallos detectados.	Correcto
Sobrecarga de comandos	Presionar botones simultáneamente en la aplicación	Robot debe priorizar último comando.	Último comando ejecutado.	Correcto
Interrupción Wi-Fi	Pérdida de conexión por 5 s	Reconexión automática.	Reconexión en ~2-10 s.	Correcto
Temperatura ambiente alta (40°C)	Evaluar estabilidad de ESP32	Sin sobrecalentamiento crítico.	Sin operación.	Incorrecto

4.6. Comparación

Tomando en cuenta las limitaciones del prototipo de este proyecto se puede decir que cumple, sobre todo, al poner en balance el costo del proyecto con las funcionalidades que puede realizar, en la Tabla 4.6.1 se hace una comparación con otro proyecto similar mencionado en el estado del arte, el cual incorpora sensores y cómputo de gama alta (LiDAR RoboSense M1, cámaras térmicas HIKMICRO, NVIDIA Jetson Orin) por lo que tiene un costo mucho mayor.

Tabla 4.6.1. Comparativa con otro proyecto

Métrica	**Robot para siniestros controlado por IoT (proyecto propuesto)**	**Robot avanzado de extinción y autonomía (Nature / Scientific Reports, 2025)**
Autonomía (uso intensivo)	**≈ 30–45 min**	**≈ 30–90 min**, dependiendo del uso del procesamiento (LiDAR, Jetson Orin)
Latencia: RTT promedio (Android ↔ ESP32)	**30–80 ms** (TCP sockets locales)	**< 50 ms** (LAN dedicada o edge computing con Jetson y ROS2)
Latencia: Envío de comando (UI → motor)	**100–300 ms** (incluye procesamiento en app Android y envío Wi-Fi)	**50–200 ms** (ROS o control directo mediante módulos dedicados)
Alcance operativo interior	**20–40 m** (Wi-Fi 2.4 GHz estándar)	**>100 m** (con antenas externas, Wi-Fi industrial o enlaces 5G)
Alcance en línea de vista exterior	**40–60 m**	**50–200 m**
Tolerancia a fallos (nodo desconectado)	**Funcionalidad degradada pero operativa** (sigue activo de forma local)	**Alta tolerancia** mediante control distribuido, fusión de sensores y redundancia
Tiempo de reconexión automática	**2–10 s** (reconexión Wi-Fi y socket TCP)	**2–30 s**, dependiendo del stack de comunicación y protocolo ROS
Costo total estimado	**≈ 3,700 MXN (USD ~200)**	**> USD $10,000** — componentes de alta gama: Jetson Orin, LiDAR RoboSense M1, cámara térmica HIKMICRO, sensores avanzados y chasis reforzado
Arquitectura de comunicación	**Sistema distribuido basado en Wi-Fi y sockets TCP/UDP** entre ESP32 y app Android	**Sistema híbrido con ROS2 + edge computing + conexión a nube para supervisión remota**
Enfoque principal	**Operación remota en entornos de siniestros locales con recursos limitados**	**Autonomía avanzada y percepción 3D para rescate o entornos industriales**

CAPITULO 5
CONCLUSIÓN

El desarrollo de robots para siniestros controlados mediante internet de las cosas representa un avance significativo en la forma en que se gestionan y atienden situaciones de emergencia y desastres, permitiendo disminuir el riesgo tanto para las personas que se encuentran dentro como de las que intervendrán para salvaguardar esas vidas, a lo largo del desarrollo del presente proyecto se han logrado avances significativos que permiten que las funcionalidades del robot como lo son el monitoreo y comunicación en entornos de siniestros puedan funcionar de manera satisfactoria, integrando con éxito múltiples dispositivos.

Se logró el desarrollo de un proyecto distribuido ya que cada ESP32 opera como un nodo con responsabilidades específicas, comunicándose con la aplicación móvil mediante una red local con ayuda de sockets TCP/UDP e hilos para que las tareas estén paralelizadas y asignadas entre diferentes dispositivos, mejorando la eficiencia y resiliencia del sistema. Asimismo, se integró de manera satisfactoria la operación de múltiples sensores para la medición del ambiente (humedad, temperatura y luz).

Se consiguió integrar al prototipo algo de suma importancia como lo es la transmisión de video y control en tiempo real, lo que permite saber en todo momento lo que el robot esté “viendo” por así decirlo, permitiendo al posible personal de rescate tener más contexto sobre el interior de los siniestros.

De igual manera, se implementó la funcionalidad de comunicación por voz de manera correcta, permitiendo entablar una comunicación entre las personas que se puedan encontrar dentro del siniestro con las que están fuera, por lo que esta comunicación mediante la voz representa una funcionalidad importante en el prototipo en su objetivo de poder ayudar a salvaguardar vidas, asimismo, dentro de la implementación se concluyó con el desarrollo de una aplicación móvil desde cero,

donde se pueden visualizar todas las funcionalidades del robot, permitiendo el acceso más sencillo a estas.

Al evaluar el desempeño del prototipo se pudo notar que el sistema demostró un desempeño estable y consistente de acuerdo con el objetivo del proyecto y los resultados esperados.

Se mantuvo el tiempo promedio de respuesta entre la aplicación Android y el microcontrolador ESP32 dentro del rango de 30 a 80 milisegundos, lo cual permitió un control remoto sin interrupciones ni demoras muy notorias al mandar algún movimiento (comando) o durante la transmisión de datos.

El tiempo de envío y ejecución de los comandos desde la interfaz de usuario hasta los motores, presentó una latencia promedio de entre 100 a 300 milisegundos, aceptable para operaciones en tiempo real.

En cuanto a la autonomía energética del robot, pudo mantener un funcionamiento continuo de cerca de 90 minutos, esto bajo condiciones normales y de entre 30 y 45 minutos bajo un uso intensivo, resultados que son acordes al consumo energético promedio estimado para los componentes electrónicos y motores empleados.

El alcance en que puede operar el prototipo fue de entre 20 a 40 metros en entornos de interiores con obstáculos, y de hasta 60 metros en línea de vista exterior (entornos sin obstáculos y rectos), lo cual permitió validar la eficiencia del módulo Wi-Fi del ESP32 para aplicaciones de campo.

Además, el sistema presentó una tolerancia apropiada a errores cuando alguno de sus componentes (tarjetas ESP32) tuvo alguna pérdida temporal de conexión, lograron reconectarse automáticamente en un tiempo promedio de 2 a 10 segundos, manteniendo operativas las funciones esenciales.

Estos resultados reflejan que el desempeño general del robot es suficiente para escenarios de monitoreo remoto, garantizando un equilibrio entre la respuesta, el alcance y la autonomía.

El desarrollo de este proyecto demostró que es posible implementar alternativas más viables y de menor costo a problemáticas como lo es el rescate de personas en siniestros, además poder contar con prototipos de libre acceso y escalables.

Asimismo, se pudo observar la posibilidad de integrar múltiples tecnologías y modelos como lo son IoT y sistemas distribuidos a fin de tener un sistema funcional, permitiendo tener un prototipo que visto desde el punto de vista económico puede ser viable y reproducible, ya que tomando en cuenta el costo total del hardware de este proyecto se puede decir que tiene un costo muchísimo menor al de sistemas comerciales equivalentes.

El impacto del proyecto radica en que se puede aplicar a situaciones reales, como lo son derrumbes, zonas con presencia de humo o fuego y lugares donde la visibilidad y el acceso son limitados, contando también con una aplicación móvil con una interfaz sencilla lo que permite que la curva de aprendizaje del posible operador sea muy baja, dicho de otra manera, es sencilla de usar, así como intuitiva y segura.

En conjunto, este proyecto evidencia que el paradigma IoT junto con el modelo de sistemas distribuidos, aplicados a la robótica móvil, puede ofrecer soluciones accesibles y eficientes para la atención y monitoreo en escenarios de emergencia, logrando aportar una herramienta útil tanto para la protección civil como para la investigación tecnológica.

Esta iniciativa proporciona evidencia práctica acerca de la posibilidad de integrar internet de las cosas con sistemas distribuidos en robots móviles para escenarios de emergencia. Contribuye principalmente al cuerpo de conocimiento a través de un enfoque de arquitectura modular reproducible, de bajo costo y enfocado a situaciones de riesgo.

Gracias a la arquitectura distribuida, que se basa en nodos (tarjetas ESP32), el sistema se puede generalizar hacia otras aplicaciones como el monitoreo industrial, la agricultura, la vigilancia autónoma, entre otros, todo bajo la misma estructura que se siguió para este proyecto: modularidad y bajo costo.

La limitación principal del proyecto fue el costo, ya que se determinó que el proyecto se realizaría con el menor posible a modo de que fuera fácil de reproducir y usarse más fácilmente, por lo que muchas de las características del robot se determinaron por los componentes que se podían conseguir acorde al costo bajo, por ejemplo, la autonomía del robot fue determinada por el tamaño de la batería que se podría conseguir, de igual manera otra limitación fue la latencia que se debía tener, la cual debería ser baja ya que el robot debería operar en entornos donde esto es de mucha importancia, y tomando en cuenta estas dos limitaciones fue determinado el tamaño del robot, ya que se debía tener un equilibrio entre el tamaño, peso y costo.

Dentro de las limitaciones operativas del robot, tomando en cuenta las limitaciones iniciales mencionadas anteriormente, se tuvo que el alcance del robot en distancia no es mucho sobre todo al funcionar en una red local, esto debido a dar prioridad a la latencia permitiendo al sistema funcionar como un punto a punto, otra limitación es que el prototipo depende de una red estable además de no contar con integración a plataformas más globales como lo es la nube por ejemplo, y finalmente la limitación de la autonomía es importante ya que el robot no puede operar durante sesiones prolongadas de más de 45 minutos.

Asimismo, durante el desarrollo se lograron identificar riesgos operativos, como: una posible pérdida temporal de comunicación debido a entornos con interferencias, potenciales fallos mecánicos a causa de una sobrecarga del hardware, así como el riesgo de sobrecalentamiento por un uso intensivo del prototipo, estos riesgos se tomaron en cuenta a la hora de realizar las pruebas de funcionamiento.

Por lo que como trabajos a futuro se podrían dar solución a estas limitaciones y riesgos, permitiendo al prototipo contar con conectividad IoT global a la nube, así como poder integrar IA para tener una detección de personas más autónoma y confiable, además de una optimización energética ya sea en los componentes de hardware o mediante software y finalmente a fin de mejorar el alcance poder usar redes más avanzadas como 4G y 5G.

Como modelo de comunicación distribuida, en este trabajo se propone una arquitectura basada en nodos, donde cada uno de ellos se especializa en determinadas tareas, interactuando mediante sockets TCP/UDP en una red local y con ayuda de los hilos para paralelizar las labores, a fin de distribuir la carga entre los módulos, este esquema puede servir como referencia para futuros trabajos donde sea necesario mantener comunicación robusta, baja latencia y resiliencia a fallos.

En conclusión, el prototipo desarrollado representa una base sólida para el desarrollo de robots de bajo costo para poder dar solución a problemáticas del mundo real, donde es muy importante brindar más herramientas y técnicas para que cada día exista menos riesgo para las personas, contribuyendo así al avance de soluciones inteligentes dentro del campo del Internet de las cosas y los sistemas distribuidos.

REFERENCIAS

.

[1]. M. Hashimoto, M. Ozaki, T. Yokoyama, and K. Takahashi, *Seguimiento de personas mediante láser mediante múltiples robots móviles,* in Proc. IEEE/ASME Int. Conf. Advanced Intelligent Mechatronics (AIM), Budapest, Hungary, 2011, pp. 37–42, doi: 10.1109/AIM.2011.6026995.

[2]. M. Volkhardt and H.-M. Gross, *Finding People in Apartments with a Mobile Robot,* in Proc. IEEE Int. Conf. Systems, Man, and Cybernetics, Manchester, U.K., 2013, pp. 4348–4353, doi: 10.1109/SMC.2013.742.

[3]. P. W. Lovon-Ramos, Y. Rosas-Cuevas, C. Cervantes-Jilaja, M. Tejada-Begazo, R. E. Patiño-Escarcina, and D. Barrios-Aranibar, *People Detection and Localization in Real Time during Navigation of Autonomous Robots,* in Proc. XIII Latin American Robotics Symp. and IV Brazilian Robotics Symp. (LARS/SBR), Recife, Brazil, 2016, pp. 239–244, doi: 10.1109/LARS-SBR.2016.47.

[4]. Z. Yang, H. Lu, P. Wang, and S. Guo, *Coordinating Obstacle Avoidance of a Redundant Dual-Arm Nursing-Care Robot*, Bioengineering, vol. 11, no. 6, p. 550, 2024.

[5]. X. Zhao et al., *Development of an autonomous fire rescue and surveillance robot integrating LiDAR, thermal imaging, and AI-based navigation*, *Scientific Reports*, 2025.

[6]. J. M. Jordan, *Robots*. MIT Press, 2016.

[7]. F. Reyes, *Robótica: Control de robots manipuladores*, 1st ed. México: Alfaomega, 2011.

[8]. S. K. Saha, *Introducción a la robótica*, 1st ed. México: McGraw-Hill, 2010.

[9]. A. Ollero Baturone, *Robótica: Manipuladores y robots móviles*. México: Alfaomega, 2007.

[10]. R. Kelly and V. Santibáñez, *Control de movimiento de robots manipuladores*. España: Prentice Hall, 2003.

[11]. Murphy, R. R., S. Tadokoro and A. Kleiner, *Disaster robotics*. in Springer Handbook of Robotics, B. Siciliano and O. Khatib, Eds. Springer International Publishing, 2016, pp. 1577–1604, doi: 10.1007/978-3-319-32552-1_60.

[12]. Tadokoro, S., *Rescue Robotics: DDT Project on Robots and Systems for Urban Search and Rescue*. Springer-Verlag, 2009. doi: 10.1007/978-3-642-00196-2.

[13]. R. R. Murphy, *Disaster Robotics*. Cambridge, MA, USA: MIT Press, 2014.

[14]. R. R. Murphy, *Trial by fire [rescue robots]*. IEEE Robotics & Automation Magazine, vol. 11, no. 3, pp. 50–61, Sep. 2004.

[15]. R. C. Ramírez Estrada, A. L. Hernández-Sánchez, D. A. Aguilar-Díaz, and J. M. Pérez-Sánchez, "Sensores y tipos de sensores," *ESTR*, vol. 11, no. 21, 2024, doi: 10.29057/estr.v11i21.11779.

[16]. C. Romero, F. Vázquez, and C. De Castro, *Domótica e inmótica: Viviendas y edificios inteligentes*, 3rd ed. España: Alfaomega, 2011.

[17]. J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013, doi: 10.1016/j.future.2013.01.010.

[18]. L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010, doi: 10.1016/j.comnet.2010.05.010.

[19]. Somerville, *Ingeniería de software*, 9th ed. México: Pearson Educación, 2011.

[20]. S. Sanchez, M. A. Sicilia, and D. Rodríguez, *Ingeniería de software: Un enfoque desde la guía SWEBOK*, 1st ed. México: Alfaomega, 2012.

[21]. P. Stevens and R. Pooley, *Utilización de UML en ingeniería de software con objetos y componentes*. Madrid: Pearson Educación, 2002.

[22]. S. R. Schach, *Ingeniería de software clásica y orientada a objetos*, 6th ed. México: McGraw-Hill, 2002.

[23]. L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Addison-Wesley, 2013.

[24]. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 8th ed. Pearson, 2021.

[25]. A. Frenkel, "Transmission Control Protocol (TCP)," *TechTarget*, 2024. Online. Available: https://www.techtarget.com

[26]. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 10th ed. Wiley, 2018.

[27]. S. Tanenbaum and M. van Steen, *Distributed Systems: Principles and Paradigms*, 2nd ed. Pearson Prentice Hall, 2007.

[28]. S. Tanenbaum, *Modern Operating Systems*, 2nd ed. Prentice Hall, 2001.

[29]. Arduino SA, “WiFiClient.cpp,” *Arduino WiFiNINA Library*, GitHub, 2018. Online. Available: https://github.com

[30]. Android Developers, “ServerSocket,” *Android API Reference*, Google. Online. Available: https://developer.android.com

[31]. R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner’s Approach*, 9th ed. McGraw-Hill Education, 2020.