

ROBOT PROTOTIPO PARA SINIESTROS CONTROLADO POR INTERNET DE LAS COSAS

Bárbara Emma Sánchez Rinza¹, Aquino Regalado Mariano Aldair²
Benemérita Universidad Autónoma de Puebla
Facultad de Físico Matemáticas¹
Facultad de Ciencias de la Computación²
Puebla, Mex.
barbara.sanchez @correo.buap.mx, ar223470493@alm.buap.mx

RESUMEN

El principal desafío para los equipos de rescate y protección civil es el poder brindar una respuesta rápida y segura ante siniestros, esto ya que enfrentan situaciones de alto riesgo en el que el poder salvaguardar vidas se vuelve mucho más difícil. Teniendo esto se propone que este trabajo pueda dar solución a esta problemática con ayuda del diseño y desarrollo de un robot asistido y controlado por internet de las cosas (IoT), con la capacidad de trabajar en entornos peligrosos y de difícil acceso tras un siniestro como derrumbes, fugas de sustancias tóxicas entre otras. El sistema está integrado por sensores ambientales, tarjetas ESP32, así como una comunicación inalámbrica con un dispositivo móvil y un sistema distribuido con conexión por sockets e hilos.

Palabras Clave: Robot, Sistema Distribuido, Hilos, IoT.

ABSTRACT

The main challenge for rescue and civil protection teams is providing a rapid and safe response to disasters, as they face high-risk situations in which safeguarding lives becomes much more difficult. With this in mind, it is proposed that this work can provide a solution to this problem with the help of the design and development of a robot assisted and controlled by the Internet of Things (IoT), capable of working in dangerous and difficult-to-access environments after a disaster, such as landslides, toxic substance leaks, and others. The system is integrated by environmental sensors, ESP32 cards, as well as wireless communication with a mobile device and a distributed system connected by sockets and wires.

Keywords: Robot, Distributed System, Threads, IoT.

INTRODUCCIÓN

En la actualidad, la atención a situaciones de emergencia, como incendios, explosiones, derrumbes o ambientes contaminados, plantea desafíos significativos para la seguridad de los equipos de rescate. En este contexto, la robótica se ha consolidado como una herramienta esencial para intervenir en zonas de alto riesgo, minimizando la exposición humana a condiciones extremas.

De acuerdo con Siciliano & Khatib, un robot es un mecanismo accionado programable en dos o más ejes con un grado de autonomía, que se mueve dentro de su entorno para realizar tareas previstas [1].

Por lo que un robot para siniestros controlado mediante Internet de las Cosas (IoT) combina las capacidades autónomas y de manipulación de los robots con la conectividad y el control remoto en tiempo real que ofrece la infraestructura IoT.

La idea general de este concepto es la presencia generalizada a nuestro alrededor de una variedad de cosas u objetos como lo son: microcontroladores, sensores, actuadores, teléfonos móviles, etc. que, mediante protocolos y topologías de direccionamiento únicos, pueden interactuar entre sí y cooperar con sus vecinos para alcanzar objetivos comunes [2].

Con todo esto es posible monitorear el entorno, recopilar información crítica, y operar el robot de forma remota desde ubicaciones seguras, permitiendo así optimizar las labores de rescate, logrando que las personas capacitadas para el rescate no tengan que entrar a estos entornos peligrosos a ciegas, sobre todo teniendo en cuenta que pueden existir peligros como incendios, gases dañinos para la salud, polvo, humedad que provoque derrumbes, etc.

Este trabajo aborda el diseño, implementación y potencial impacto de un sistema robótico para siniestros que aprovecha la conectividad IoT, destacando sus aplicaciones, beneficios y desafíos tecnológicos asociados.

Se implementa un robot móvil con múltiples sensores para medir el ambiente, así como motores, tarjetas ESP32, una aplicación móvil desarrollada en Android y una comunicación por sockets TCP, permitiendo el desarrollo de un sistema distribuido entre las tarjetas ESP32 y la aplicación móvil.

1. DESARROLLO

1.1. Arquitectura del sistema

El sistema está compuesto por dos motores controlados por la tarjeta L298N, sensores de temperatura, humedad, gas, 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 como se muestra en la Figura 1.

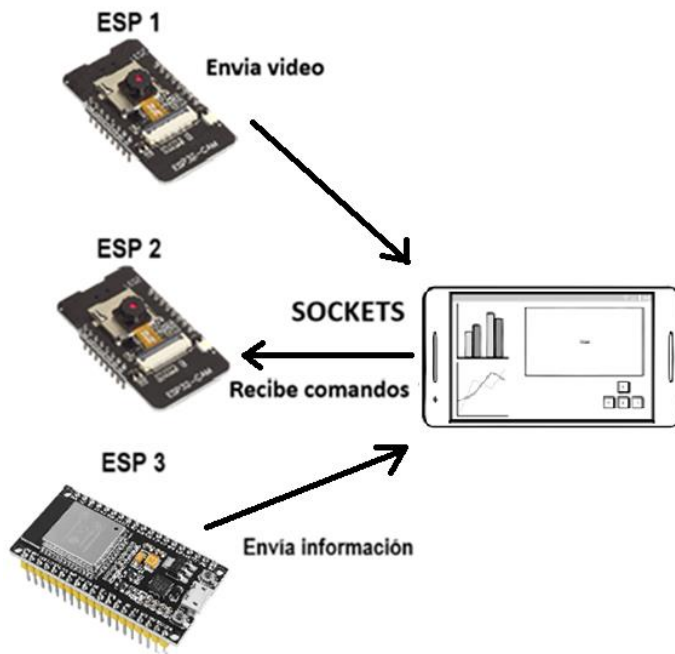


Figura 1. Arquitectura.

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 [3]

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.

En el caso de este proyecto que se usarán 3 controladores ESP32 (Figura 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 Wifi 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 [4]. 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 pérdida de información.

1.2. Hardware y software

El cerebro del robot por así decirlo serán las tarjetas ESP32, dentro de estas se cuentan con 2 tarjetas Esp32-CAM y una ESP32-DevKit, y a estas se conectan los sensores y motores del robot.

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).



Figura 2. Tarjeta ESP32-CAM.

Sensor MLX90614: Es un sensor de temperatura sin contacto que mide la temperatura a través de radiación infrarroja ver Figura 3, además de tener un precio relativamente económico e ideal para medir temperatura corporal a distancia.



Figura 3. Sensor de temperatura a distancia.

Sensor de temperatura DHT11: Detecta temperatura, así como humedad del ambiente

Sensor MQ-2: detecta gases inflamables, permitiendo alertas en tiempo real

Sensor de luz (fotorresistor) es un componente electrónico cuya resistencia se varía dependiendo del aumento de intensidad de luz incidente en el ambiente.

Motores DC: 2 motores unidireccionales con voltaje de 9 a 12 volts con ruedas tipo oruga para poder moverse por terrenos irregulares, ver Figura 4.



Figura 4. Motor DC

Y como alimentación para el robot se usa una **batería Lipo** de 2200 mah 3s 30c, ver figura 5.



Figura 5. Batería Lipo

Y como tecnología de conectividad para el robot con el celular se usa:

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.

1.3. Interfaz de visualización

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.

En la Figura 6 se puede visualizar la interfaz principal en la cual se visualizarán los datos del medio capturados por el robot, como lo son la temperatura, humedad, luz, polvo, etc.

Y así mismo 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.



Figura 6. Aplicación móvil

1.4. Comunicación y paralelización

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 [5]. Esto se puede al paralelizar cada función de cada esp32 con ayuda de hilos, de acuerdo con Tanenbaum es la unidad más pequeña de utilización de CPU, que posee su propio contador de programa, registros y pila [6].

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 librería WifiClient.

De modo que cada ESP32:

1. Se conecta a la misma red wifi 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().

En la funcionalidad se puede ver en el siguiente código (Figura 7) donde cada ESP32 cuenta con esta función que permite realizar los pasos mencionados anteriormente para la conexión con el servidor.

```
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 7. Función de conexión por socket ESP32

Para el caso del servidor, el celular se convierte en él, con ayuda de los datos móviles de cualquier compañía de comunicaciones, esto con la librería ServerSocket.

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.

Estos pasos realizados con el siguiente código (Figura 8).

```

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

```

Figura 8. Función de conexión para el servidor

El proceso final se puede ver de mejor manera en la siguiente Tabla 1, en la cual se ven los pasos que sigue tanto el servidor como los clientes.

Tabla 1. Proceso de conexión cliente-servidor.

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

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.

El programa cuenta con 4 hilos dentro de funciones descritos en la siguiente tabla 2.

Tabla 2. Hilos del programa.

Hilo (Thread)	Función	Método	Función principal
Hilo 1	UI/Main Thread	onCreate()	Muestra la interfaz gráfica, actualiza TextViews, responde a clics de usuario.
Hilo 2	setupServer()	new Thread(..)	Abre un ServerSocket para recibir temperatura corporal de la ESP32-CAM.
Hilo 3	setupDHTServer()	new Thread(..)	Recibe temperatura ambiente, humedad y presión desde la ESP32 DevKit por sockets.
Hilo 4	sendCommand()	new Thread(..)	Envía comandos de movimiento a la ESP32-CAM (arriba, abajo, etc.).

1.5. Detección de calor

La función de detección de calor se realiza con ayuda del sensor MLX90614 el cual es un sensor infrarrojo de bajo costo que detecta calor a distancia y que con ayuda de la programación en Android con el hilo 2 (Tabla 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 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, esto tomando en cuenta que, al estar en una situación de riesgo en un siniestro, una persona puede sufrir temperaturas anormales, por ejemplo, ya sea hipotermia o hipertermia.

El proceso es el siguiente:

1. El ESP32-CAM (2) está conectado a un sensor de temperatura a distancia (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, hipotermia o una fuente de calor externa peligrosa.

Visto de manera más sencilla, el robot va midiendo la temperatura en un punto específico, si en algún momento ese valor supera un umbral crítico (con un valor no muy común por el entorno donde se está midiendo) se muestra en pantalla un punto rojo como señal de alerta, y este puede ser una fuente de calor externa peligrosa como un incendio o una persona y si lo es, si su temperatura está por debajo de los 36 grados podría sufrir hipotermia y en caso contrario hipertermia y en los dos casos la información sería de mucha importancia para que los rescatistas le puedan dar prioridad a estas personas en riesgo. Esto se puede ver de igual manera en la siguiente Figura 9 en la cual se muestra el código donde se obtiene la información por parte de la ESP32 y la validación con el umbral dado para posteriormente visualizar el punto rojo.

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

Figura 9. Función para el punto rojo.

2. RESULTADOS

2.1. Hardware

Para la construcción del robot de manera física se tomó en cuenta que el robot intervendría en entornos de siniestros donde el

espacio es reducido, como pasillos estrechos, escombros o zonas de difícil acceso, por lo que se buscó un tamaño compacto que permitiera movilidad y maniobrabilidad y a su vez tomara en cuenta todos los componentes necesarios para la recolección de información en los siniestros, por lo que el tamaño estuvo condicionado por el factor del volumen de los componentes internos, como las tarjetas esp32, los motores, y la batería siendo la de mayor tamaño, por lo que buscando mantener un equilibrio entre **autonomía, estabilidad y que fuera compacto**, el robot se construyó con un tamaño de 18 centímetros de largo, 17 centímetros de ancho y 18 centímetros de alto, esto se puede ver mejor en la siguiente Figura 10.



Figura 10. Robot físico.

2.2. Software

En el aspecto del software se tiene la conclusión de la aplicación móvil con las funcionalidades antes mencionadas como lo son la cámara en tiempo real, la detección del medio mediante sensores y estos mostrándolos en pantalla, así como el control del robot de forma remota con ayuda de los botones de control para ir hacia todos los sentidos (arriba, izquierda, derecha) visto en la Figura 11.



Figura 11. Aplicación móvil en funcionamiento.

Y para la detección de calor y por ende de posibles personas en riesgo se tiene la visualización de la pantalla en funcionamiento con la prueba acercando una mano al robot y visualizando el punto rojo en pantalla visto en la Figura 12.



Figura 12. Aplicación móvil en funcionamiento.

Y finalmente en la Tabla 3 se muestra el desempeño del robot mediante algunas métricas promedio clave, se observó una autonomía que varía entre 30 y 90 minutos, teniendo una latencia aceptable entre la aplicación móvil y el robot, así como el alcance de la transmisión de los datos siendo de 20 a 40 metros en interiores y de 40 a 60 metros en exteriores y con tolerancia a fallos al desconectarse y en un rango de 2 a 10 segundos volver a conectarse.

Tabla 3. Métricas del desempeño del robot.

Métrica	Valor aproximado
Autonomía (movimiento normal)	≈ 90 min
Autonomía (uso intensivo)	≈ 30–45 min
Latencia: RTT promedio (Android ↔ ESP32)	30–80 ms
Latencia: Envío de comando (UI → motor)	100–300 ms
Alcance operativo interior	20–40 m
Alcance en línea de vista exterior	40–60 m
Tolerancia a fallos (nodo desconectado)	Funcionalidad degradada pero operativa
Tiempo de reconexión automática	2–10 s

3. CONCLUSION

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, 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 ESP32 (ESP32-CAM y ESP32 DevKit V1), sensores (DHT11, ultrasonido HC-SR04, sensor de temperatura corporal MLX90614), un puente H L298N y motores, formando un sistema funcional.

Así como la aplicación que permite controlar los motores, visualizar video en tiempo real, y recibir datos ambientales y de temperatura corporal en tiempo real mediante sockets, todo esto tomando en cuenta que **el sistema ha sido diseñado pensando en recursos limitados**, utilizando hardware de bajo costo y bajo consumo energético, por lo que dentro de las limitaciones del proyecto están la necesidad de trabajar bajo un esquema de **bajo costo**, así mismo garantizar una **baja latencia** en la transmisión de video y en el control remoto de los motores, De igual forma, **el tamaño del prototipo** se definió con el objetivo de mantener un equilibrio entre maniobrabilidad en espacios reducidos y la posibilidad de integrar los sensores necesarios, lo que limita la incorporación de baterías de mayor capacidad y componentes más robustos.

Como posibles trabajos a futuro con la ventaja de que el proyecto es escalable se pueden agregar más funcionalidades importantes al robot, como lo son, mejorar el alcance, poder controlar el robot desde cualquier parte del mundo, manejo de datos en la nube, más sensores para detección del medio, una posible comunicación por voz, así como la posibilidad de funcionalidades con IA que permitan reconocer personas de manera más eficaz.

Es un **proyecto distribuido** ya que cada ESP32 opera como un nodo con responsabilidades específicas, comunicándose con la aplicación móvil mediante una red local y las tareas están paralelizadas y asignadas entre diferentes dispositivos, mejorando la **eficiencia** y **resiliencia** del sistema.

3.1. Referencias

- [1] B. Siciliano and O. Khatib, Eds., "Springer Handbook of Robotics". Springer, 2016.
- [2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010, doi: 10.1016/j.comnet.2010.05.010.
- [3] J. F. Kurose and K. W. Ross, "Computer Networking: A Top-Down Approach", 8th ed. Pearson, 2021.
- [4] A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating System Concepts", 10th ed. Wiley, 2018.
- [5] A. S. Tanenbaum and M. van Steen, "Distributed Systems: Principles and Paradigms", 2nd ed. Pearson Prentice Hall, 2007.
- [6] A. S. Tanenbaum, "Modern Operating Systems", 2nd ed. Prentice Hall, 2001.