

Universidad Autónoma de Baja California

FACULTAD DE INGENIERÍA

INGENIERÍA EN COMPUTACIÓN



Compilación de PROYECTO FINAL

Asignatura

Sistemas Embebidos

Presentan:

Ramses Felix Prado
Jordan Jorge Payta Sarabia

Profesor :

Dr. Adolfo Heriberto Ruelas Puente

Mexicali, B.C. México

Noviembre 2024

Resumen

Los baches representan un desafío significativo para la seguridad vial y el mantenimiento de las infraestructuras, provocando accidentes, daños a los vehículos y costos elevados de reparación. Para abordar este problema, este trabajo presenta un sistema de detección de baches en tiempo real que utiliza un microcontrolador ESP32, un sensor acelerómetro y giroscopio MPU6050, y un módulo GPS. El sistema identifica irregularidades en las carreteras, como baches, analizando patrones de aceleración y vibración, y registra los eventos detectados, incluyendo la geolocalización, en una base de datos en Firebase. Estos datos son visualizados a través de una aplicación móvil desarrollada en React Native, que permite a los usuarios monitorear las ubicaciones de los baches en un mapa interactivo. La solución propuesta tiene como objetivo asistir a los conductores en la evitación de baches al proporcionar información actualizada sobre las condiciones de la carretera, reduciendo así los accidentes y los costos de mantenimiento vehicular. A diferencia de los métodos tradicionales, este sistema ofrece un enfoque eficiente, de bajo costo y escalable para la detección y mapeo de baches en tiempo real, fomentando redes de transporte más seguras y una mejor gestión de las infraestructuras.

Dedicatoria y Agradecimientos

Este proyecto está dedicado a todas las personas que, de una u otra forma, han sido inspiración y apoyo en mi desarrollo profesional y personal. A mi familia, por su amor incondicional, por creer en mí en todo momento y por enseñarme la importancia de la perseverancia y el esfuerzo. A mis profesores, quienes con su dedicación y conocimiento me han guiado en este camino, fomentando en mí la pasión por el aprendizaje y la innovación. Y, finalmente, a todos aquellos que diariamente enfrentan los desafíos de nuestras carreteras, esperando contribuir con este trabajo a mejorar la seguridad vial y la calidad de vida de las personas.

Quiero expresar mi más sincero agradecimiento a todas las personas e instituciones que hicieron posible la realización de este proyecto.

A mi familia, por su apoyo constante, su paciencia y su aliento en cada etapa de este camino. Su confianza en mis capacidades ha sido fundamental para alcanzar este logro.

A mis docentes y mentores, quienes con su experiencia y sabiduría me han guiado en la comprensión de los conocimientos necesarios para desarrollar este trabajo. Su entusiasmo y compromiso han sido una fuente de motivación constante.

A mis compañeros, por compartir ideas, desafíos y aprendizajes, enriqueciendo este proyecto con su colaboración y perspectivas.

Índice general

Índice de figuras	1
1. CAPITULO 1	3
1.1. Antecedentes	3
1.1.1. Introducción	3
1.1.2. Problematica	4
1.1.3. Soluciones Existentes	4
1.2. Problemática	4
1.3. Justificación	4
1.4. Objetivo general	5
1.5. Objetivos específicos	5
1.6. Hipótesis	6
2. CAPITULO 2	7
2.1. Marco Teórico	7
2.1.1. ESP32	7
2.1.2. NEO-6M	8
2.1.3. Maquina CNC	9
2.1.4. Aplicación web	9
2.1.4.1. ¿Cuáles son los beneficios de las aplicaciones web? .	9
2.1.5. Javascript	10
2.1.6. Node js	10
2.1.7. ESP-IDF	12
2.1.8. React Native	12
2.2. FreeRTOS	13
2.2.1. Características	14
2.2.1.1. Conectividad	14
2.2.1.2. Protección del dispositivo, la conexión y las actualizaciones	14
2.2.1.3. Actualizaciones inalámbricas	15
2.2.2. Casos de uso:	15
3. CAPITULO 3	17
3.1. Introducción	17
3.2. Requerimiento Funcionales	17
3.2.1. Detector de Baches	17

Índice general

3.2.2. Geolocalización	17
3.2.3. Almacenamiento en la nube	17
3.2.4. Aplicación móvil	18
3.2.5. Notificaciones	18
3.2.6. Requerimientos No Funcionales	18
3.2.7. Escalabilidad:	18
3.2.8. Fiabilidad:	18
3.2.9. Usabilidad:	18
3.2.10. Portabilidad:	18
3.2.11. Tiempo de respuesta:	18
3.3. Recursos Necesarios	19
3.3.1. Hardware	19
3.3.2. Software	19
3.4. Diagrama de caso de uso	19
3.5. Diagrama de secuencia	20
3.6. Diagrama de actividades	21
3.7. Diagrama de componentes	22
4. CAPITULO 4	23
4.1. Diseño Placa en Eagle	23
4.2. Diseño electrónico	24
4.3. Diseño de la Aplicación Móvil	26
4.3.1. Tecnologías utilizadas	30
4.3.2. Flujo de Navegación	30
5. CAPITULO 5	33
5.1. Estudio experimental y pruebas	33
5.1.1. Placa del proyecto	33
5.1.2. FireBase	34
6. CAPITULO 6	37
6.1. Conclusión	37
6.2. Referencias	38

Índice de figuras

2.1. ESP32	8
2.2. Componente NEO-6M	8
2.3. Maquina cnc	9
2.4. Icono de Javascript	10
2.5. Icono de Node JS	11
2.6. Icono de React Native	12
2.7. Como se comporta Raect Native	13
2.8. Uso de FreeRTOS	14
3.1. Diagrama de caso de uso	19
3.2. Diagrama de secuencia	20
3.3. Diagrama de actividades	21
3.4. Diagrama de componentes	22
4.1. Diseño del circuito	23
4.2. Diagrama esquemático de conexiones del detector de baches	24
4.3. Diagrama esquemático del regulador de 3V	24
4.4. Diagrama esquemático del boton de RESET	25
4.5. Diagrama esquemático del boton de BOOT	25
4.6. Diagrama esquemático del esp32-Wroom-s3	26
4.7. Pagina oficial	27
4.8. Navegador de estadísticas	27
4.9. Mapa donde se registra los baches	28
4.10. Tab para reportar bache	29
4.11. Datos e Información de los reportes	30
5.1. Prototipo del circuito	33
5.2. Funcionamiento de la base de datos	34
5.3. Funcionamiento de la placa	35
5.4. Producto Final del proyecto	36

Índice de figuras

1

CAPITULO 1

**Payta Sarabia Jordan Jorge
Ramses Feliz Prado**

1.1. Antecedentes

1.1.1. Introducción

En las últimas décadas, el rápido crecimiento de la urbanización y el incremento del tráfico vehicular han llevado al deterioro significativo de las infraestructuras viales en muchas regiones del mundo. Entre los problemas más comunes que enfrentan las carreteras se encuentran los baches, que son irregularidades formadas por el desgaste continuo de la superficie debido al tránsito pesado y las condiciones climáticas adversas. Los baches no solo representan un desafío para la comodidad de los conductores, sino que también son una de las principales causas de accidentes vehiculares, daños mecánicos y elevados costos de mantenimiento.

En este contexto, surge la necesidad de desarrollar tecnologías innovadoras que permitan identificar y registrar baches de manera precisa y en tiempo real. Estas herramientas no solo tienen el potencial de alertar a los conductores para evitar accidentes, sino también de proporcionar información valiosa a las autoridades responsables del mantenimiento vial, mejorando así la planificación y priorización de las reparaciones.

Este proyecto propone un sistema de detección de baches basado en un circuito electrónico integrado por un microcontrolador ESP32, un sensor de movimiento MPU6050 y un módulo GPS. El sistema funciona de la siguiente manera: cuando el sensor detecta un bache, la información sobre su ubicación geográfica, junto con otros parámetros relevantes, se envía automáticamente a una base de datos alojada en Firebase. Posteriormente, esta información es recuperada por una aplicación móvil desarrollada en React Native, la cual permite a los usuarios visualizar los baches registrados en un mapa interactivo. Este enfoque busca no solo reducir el riesgo de accidentes, sino también optimizar el mantenimiento de las carreteras.

El presente documento detalla cada una de las etapas del desarrollo del proyecto, incluyendo el diseño, implementación y pruebas del sistema, así como los resultados obtenidos. Además, se discuten las limitaciones del sistema actual y las posibles

mejoras futuras, con el objetivo de contribuir al desarrollo de soluciones tecnológicas que aborden problemas reales y generen un impacto positivo en la sociedad.

1.1.2. Problematica

Las mujeres ciclistas que viajan solas de noche enfrentan riesgos adicionales. La falta de visibilidad, combinada con la preocupación por la seguridad personal, puede hacer que los desplazamientos nocturnos sean peligrosos y estresantes. Los sistemas de iluminación tradicionales a menudo no son suficientes para abordar estos problemas de manera efectiva.

1.1.3. Soluciones Existentes

Antes del desarrollo de GiroSAFE, se han implementado varias soluciones para mejorar la seguridad de los ciclistas, tales como:

Luces LED para Bicicletas: Las luces delanteras y traseras son esenciales para la visibilidad nocturna. Sin embargo, su efectividad puede verse limitada si no se ajustan adecuadamente o si no son suficientemente brillantes.

Ropa Reflectante: Los chalecos y bandas reflectantes ayudan a aumentar la visibilidad, pero dependen de la iluminación externa, como los faros de los automóviles, para ser efectivos.

Aplicaciones de Seguridad: Algunas aplicaciones móviles permiten a los ciclistas compartir su ubicación en tiempo real con amigos y familiares. Aunque útiles, estas aplicaciones no abordan directamente la visibilidad en la carretera.

1.2. Problemática

En las últimas décadas, el deterioro de las carreteras se ha convertido en un problema crítico para la infraestructura vial, principalmente debido al aumento del tráfico vehicular y las condiciones climáticas adversas. Entre los problemas más recurrentes y peligrosos se encuentran los baches, que no solo representan una amenaza para la seguridad de los conductores, sino que también generan altos costos de mantenimiento de vehículos y reparaciones viales.

Actualmente, los métodos tradicionales para identificar y reparar baches suelen ser reactivos, es decir, se interviene únicamente después de recibir reportes ciudadanos o inspecciones periódicas. Esto resulta en un proceso ineficiente, que no siempre atiende los puntos más críticos a tiempo. Además, la falta de información geográfica precisa dificulta la priorización de reparaciones y la asignación adecuada de recursos.

1.3. Justificación

La seguridad en el ciclismo nocturno es un tema de gran relevancia en la promoción de la movilidad sostenible y la protección de los ciclistas. A pesar de los avances en infraestructura y tecnología, los ciclistas continúan enfrentando desafíos

significativos en términos de visibilidad y seguridad personal. El proyecto GiroSAFE se justifica por la necesidad de abordar estos desafíos de manera innovadora y efectiva. El ciclismo durante la noche presenta varios riesgos, incluyendo la baja visibilidad y la vulnerabilidad a accidentes y situaciones de emergencia. Estos problemas no solo afectan a los ciclistas individuales, sino que también tienen un impacto negativo en la percepción general de la seguridad del ciclismo como medio de transporte viable. La falta de soluciones integradas y efectivas para mejorar la seguridad nocturna de los ciclistas resalta la necesidad de desarrollar nuevas tecnologías y enfoques.

GiroSAFE tiene el potencial de transformar la seguridad nocturna para todos los ciclistas mediante la implementación de un casco inteligente con características avanzadas. La integración de luces LED adaptativas, sincronización con la bicicleta y funciones automáticas de seguridad no solo mejora la visibilidad del ciclista, sino que también proporciona una respuesta rápida en situaciones de emergencia. Este enfoque holístico puede reducir significativamente el número de accidentes y aumentar la confianza de los ciclistas al viajar de noche.

El proyecto GiroSAFE está justificado por su capacidad para abordar de manera efectiva los desafíos de seguridad nocturna que enfrentan los ciclistas. Mediante la combinación de tecnología avanzada y un diseño centrado en el usuario, GiroSAFE ofrece una solución integral que mejora la visibilidad, la seguridad personal y la confianza de los ciclistas. Este proyecto no solo beneficia a los individuos que utilizan bicicletas, sino que también tiene un impacto positivo en la sociedad en su conjunto, promoviendo una movilidad más segura y sostenible.

1.4. Objetivo general

Desarrollar un sistema automatizado para la detección de baches basado en un circuito electrónico integrado y una aplicación móvil que permita identificar, registrar y visualizar baches en tiempo real, mejorando la seguridad vial y optimizando el mantenimiento de las carreteras.

1.5. Objetivos específicos

- Diseñar e implementar un circuito basado en un microcontrolador ESP32, un sensor de movimiento MPU6050 y un módulo GPS para detectar baches y registrar su ubicación geográfica con precisión.
- Desarrollar un sistema de almacenamiento en la nube utilizando Firebase para registrar y gestionar la información sobre los baches detectados.
- Crear una aplicación móvil en React Native que permita a los usuarios visualizar los baches registrados en un mapa interactivo, proporcionando información útil para la conducción.
- Implementar un algoritmo que clasifique los baches detectados según su gravedad, brindando información adicional para la priorización de reparaciones viales.

- Validar el sistema en condiciones reales mediante pruebas de campo, evaluando su precisión, confiabilidad y facilidad de uso.
- Proporcionar una solución escalable y replicable que pueda ser adoptada por entidades gubernamentales o particulares para mejorar la calidad de las carreteras.

1.6. Hipótesis

Si se implementa un sistema de detección de baches basado en sensores de movimiento, geolocalización y almacenamiento en la nube, entonces será posible registrar y visualizar los baches en tiempo real, contribuyendo a la mejora de la seguridad vial y la eficiencia en el mantenimiento de carreteras.

2

CAPITULO 2

Payta Sarabia Jordan Jorge
Ramses Felix Prado

2.1. Marco Teórico

2.1.1. ESP32

El ESP32-WROOM-32 es un módulo que incorpora el microcontrolador ESP32 de Espressif Systems junto con otros componentes necesarios para su funcionamiento. Este módulo está diseñado para facilitar la integración del ESP32 en proyectos electrónicos y para simplificar el desarrollo de productos que requieren conectividad Wi-Fi y Bluetooth.

Características:

- **Microcontrolador ESP32:** El corazón del módulo es el microcontrolador ESP32, que tiene dos núcleos de procesamiento, conectividad Wi-Fi y Bluetooth.
- **Conectividad Inalámbrica:** El módulo admite Wi-Fi 802.11 b/g/n y Bluetooth 4.2 BLE, lo que lo hace adecuado para aplicaciones de Internet de las cosas (IoT) y otros proyectos que requieren comunicación inalámbrica.
- **Memoria Flash Integrada:** Incluye memoria flash interna para almacenar el programa y datos, lo que facilita el desarrollo de aplicaciones complejas.
- **Soporte para Desarrollo con Diversos Entornos:** Puede programarse utilizando entornos de desarrollo populares como el Arduino IDE y el entorno de desarrollo Espressif IDF (IoT Development Framework).
- **Bajo Consumo de Energía:** El ESP32 está diseñado para ser eficiente en cuanto a energía, lo que lo hace adecuado para aplicaciones alimentadas por batería y otras situaciones en las que el consumo de energía es crítico.



Figura 2.1: ESP32

2.1.2. NEO-6M

GPS NEO6M V2 es un módulo receptor que puede ser utilizado en todo aquello que requiera un aplicación de geolocalización, cuenta con una antena de gran potencia, posee una memoria EEPROM para guardar datos y una batería para respaldar la configuración del módulo, lo que le permite recibir las señales de los satélites que están alrededor de la tierra. El GPS GY-NEO6MV2 es compatible con Arduino, PIC, AVR, Raspberry y otros microcontroladores del mercado y puede entregar información precisa así como ser configurado a través del puerto UART.

El Módulo GPS NEO6MV2 es ideal para proyectos de vehículos autónomos como aeronaves, quadcopters, helicópteros, robots móviles o drones y en toda aplicación que requiera de geolocalización.

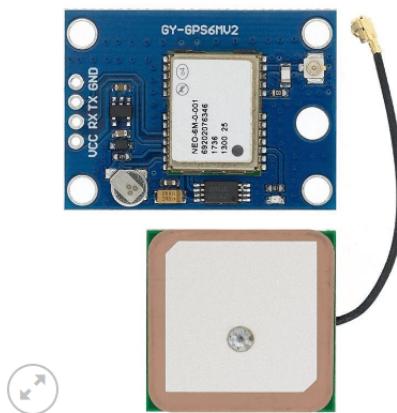


Figura 2.2: Componente NEO-6M

Para utilizar tu GPS NEO6MV2 se debe utilizar una tarjeta de desarrollo como por ejemplo puedes utilizar las placas de Arduino, ESP8266, PIC'S o cualquier otro microcontrolador que tenga puerto serie, para obtener lecturas del modulo podrás utilizar las librerías disponibles para cada entorno de programación o bien podrás obtener los datos puros que esta obtenido el modulo solo tienes que conectar

el modulo en a un puerto Serie o utilizar un Convertidor USB a TTL CP2102 y mediante un monitor serie leer los datos que esta obtenido el modulo.

2.1.3. Maquina CNC

Las máquinas CNC son dispositivos de fresado automatizados que fabrican componentes industriales sin asistencia humana directa. Utilizan instrucciones codificadas enviadas a un ordenador, lo que permite a las fábricas fabricar piezas con precisión y rapidez.

El sistema CNC, que ha transformado de manera profunda la industria, se caracteriza por su control automático, preciso y constante del movimiento. Esta maquinaria CNC, ya sea en sus formas más simples o complejas, incluye dos o más direcciones de movimiento, conocidas como ejes. Dichos ejes pueden posicionarse de manera precisa y automática a lo largo de su camino.

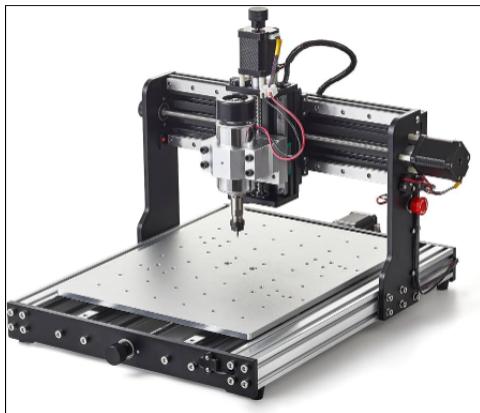


Figura 2.3: Maquina cnc

2.1.4. Aplicación web

Una aplicación web es un software que se ejecuta en el navegador web. Las empresas tienen que intercambiar información y proporcionar servicios de forma remota. Utilizan aplicaciones web para comunicarse con los clientes cuando lo necesiten y de una forma segura. Las funciones más comunes de los sitios web, como los carros de compra, la búsqueda y el filtrado de productos, la mensajería instantánea y los canales de noticias de las redes sociales, tienen el mismo diseño que las aplicaciones web. Le permiten acceder a funcionalidades complejas sin la necesidad de instalar o configurar un software.

2.1.4.1. ¿Cuáles son los beneficios de las aplicaciones web?

Las aplicaciones web tienen muchos beneficios, y casi todas las empresas grandes las utiliza como parte de sus ofertas para usuarios. A continuación se muestran alguno de los beneficios comunes asociados a las aplicaciones web.

- 1. Accesibilidad:** Las aplicaciones web son accesibles desde todos los navegadores web y desde diferentes dispositivos personales y empresariales.

Equipos de diferentes ubicaciones pueden acceder a documentos compartidos, sistemas de administración de contenidos y otros servicios empresariales a través de aplicaciones web basadas en suscripciones.

2. **Desarrollo eficiente:** Tal y como se detalló, el proceso de desarrollo para aplicaciones web es relativamente sencillo y rentable para las empresas. Los equipos pequeños pueden lograr ciclos de desarrollo cortos, lo que hace que las aplicaciones web sean una manera eficiente y asequible de desarrollar programas de computación. Además, dado que la misma versión funciona en todos los navegadores y dispositivos modernos, no tendrá que crear un número elevado de iteraciones diferentes para varias plataformas.
3. **Escalabilidad** Las empresas que utilizan aplicaciones web pueden agregar usuarios cuando sea necesario, sin necesidad de infraestructura adicional o hardware costoso. Además, la mayor parte de los datos de las aplicaciones web se almacena en la nube, lo que significa que su empresa no tendrá que invertir en capacidad de almacenamiento adicional para ejecutar aplicaciones web.

2.1.5. Javascript

JavaScript (JS) es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo (just-in-time) con funciones de primera clase. Si bien es más conocido como un lenguaje de scripting (secuencias de comandos) para páginas web, y es usado en muchos entornos fuera del navegador, tal como Node.js, Apache CouchDB y Adobe Acrobat JavaScript es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo programación funcional). Lee más en acerca de JavaScript.

Esta sección está dedicada al lenguaje JavaScript en sí, y no a las partes que son específicas de las páginas web u otros entornos host. Para información acerca de APIs específicas para páginas Web, consulta APIs Web y DOM.



Figura 2.4: Icono de Javascript

2.1.6. Node js

Es un entorno que trabaja en tiempo de ejecución, de código abierto, multiplataforma, que permite a los desarrolladores crear toda clase de herramientas de lado servidor y aplicaciones en JavaScript. La ejecución en tiempo real está pensada para usarse fuera del contexto de un explorador web (es decir, ejecutarse directamente en una computadora o sistema operativo de servidor). Como tal, el entorno omite las APIs de JavaScript específicas del explorador web y añade soporte

para APIs de sistema operativo más tradicionales que incluyen HTTP y bibliotecas de sistemas de ficheros.

En comparación con otra plataforma, Node.js tiene un flujo de trabajo particular. Funciona como un único proceso, lo que significa que no crea un nuevo hilo para cada petición. Un hilo es un conjunto de instrucciones que debe realizar el servidor. Node.js emplea operaciones de E/S no bloqueantes: cuando un cliente envía una solicitud al servidor web, el bucle de eventos de un solo hilo la recoge y la envía a un worker thread (hilo trabajador) para su procesamiento.

En lugar de bloquear el hilo y desperdiciar recursos de la CPU esperando una respuesta, Node.js continuará trabajando en la siguiente tarea. De esta manera, puede manejar una cantidad masiva de peticiones simultáneas.

Dicho esto, Node.js no es adecuado para tareas que requieran un uso intensivo de la CPU, ya que podrían impedir que el hilo principal maneje otras peticiones, bloqueándolo efectivamente.



Figura 2.5: Icono de Node JS

2.1.7. ESP-IDF

ESP-IDF (Espressif IoT Development Framework) es el marco de desarrollo oficial para los microcontroladores ESP32 de Espressif Systems. Este framework proporciona una plataforma completa para desarrollar aplicaciones de Internet de las Cosas (IoT) utilizando los microcontroladores de Espressif, incluyendo una variedad de herramientas y bibliotecas que facilitan el desarrollo de software.

- RTOS (Real-Time Operating System): Utiliza FreeRTOS, un sistema operativo en tiempo real que permite la multitarea y la gestión eficiente de recursos, crucial para aplicaciones embebidas.
- Herramientas de Desarrollo: Incluye herramientas para la configuración del proyecto, compilación, depuración y actualización de firmware. También soporta la integración con entornos de desarrollo como Visual Studio Code y Eclipse.
- Componentes Integrados: Incluye componentes para Wi-Fi, Bluetooth, manejo de energía, almacenamiento, periféricos y más, facilitando la integración de funcionalidades comunes en aplicaciones IoT.

2.1.8. React Native

Creado en Facebook y publicado en Febrero de 2015 React Native es un framework para la creación de aplicaciones nativas para iOS y Android. Para programar aplicaciones usando React Native usamos como lenguaje de programación JavaScript, en lugar de Swift, JAVA u otros lenguajes y por el resto usamos React para la creación de la interfaz del usuario.

A diferencia de otras soluciones como Ionic, Cordova o Phonegap, React Native no utiliza WebViews ni produce HTML o CSS, es decir, no usa tecnologías web para la interfaz de tu aplicación. La pregunta es, si no usa estas tecnologías cómo es que podemos usar React y JavaScript para la creación de aplicaciones nativas.

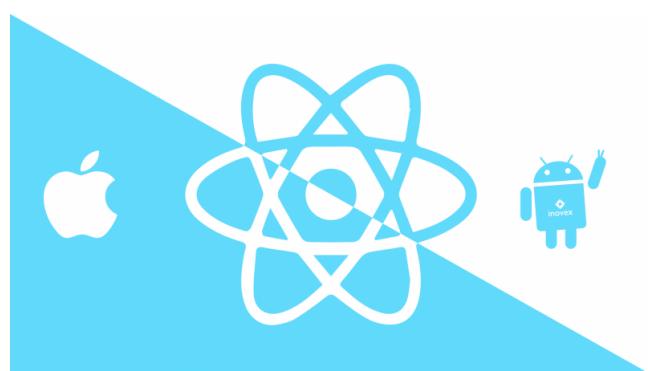


Figura 2.6: Icono de React Native

Saber que las aplicaciones de React Native no son 100

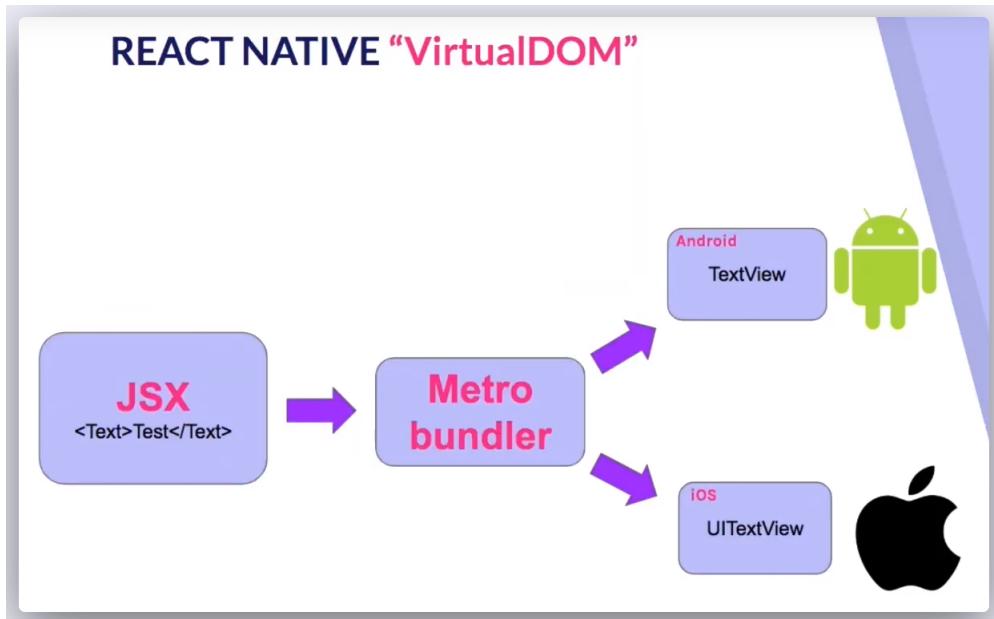


Figura 2.7: Como se comporta Raect Native

Las aplicaciones de React Native se componen de 2 elementos principalmente:

- En los navegadores web, el virtual DOM se representa a través del Document Object Model con elementos de HTML, la ventaja de que el Virtual DOM sea una capa intermedia entre el código y la representación final de nuestra app, permite que modifiquemos dicha representación basado en la descripción del virtual DOM, piensa en el Virtual DOM como una especificación textual de cómo debe verse la app, digamos, habrá un text input aquí, un botón de color rojo allá, etc. Después esta especificación debe trasladarse a los componentes de la plataforma donde se verá, de nuevo, en el caso de una página web esto significa elementos de HTML.
- Para desarrollar ReactNative se implementa un conector que tomará la descripción del Virtual DOM y basado en ésta, mandará a llamar APIs nativas de la plataforma ya sea iOS o Android y representará el DOM aprovechando dichas APIs Nativas. Esto significa que por ejemplo cuando en React Native mandes a llamar el elemento View, el conector verá cómo traducirlo a Android y cómo traducirlo a alguna API nativa de iOS.

2.2. FreeRTOS

FreeRTOS es un sistema operativo de código abierto, con neutralidad de la nube y de tiempo real que ofrece un kernel rápido, fiable y receptivo. FreeRTOS se distribuye de forma gratuita bajo una licencia de código abierto del Massachusetts Institute of Technology (MIT) y se implementa en más de 40 arquitecturas, por lo que le brinda a los desarrolladores una amplia gama de hardware, además de un conjunto de bibliotecas de software preempaquetadas. En la figura 2.6 se puede ver el uso del sistema operativo FreeRTOS.

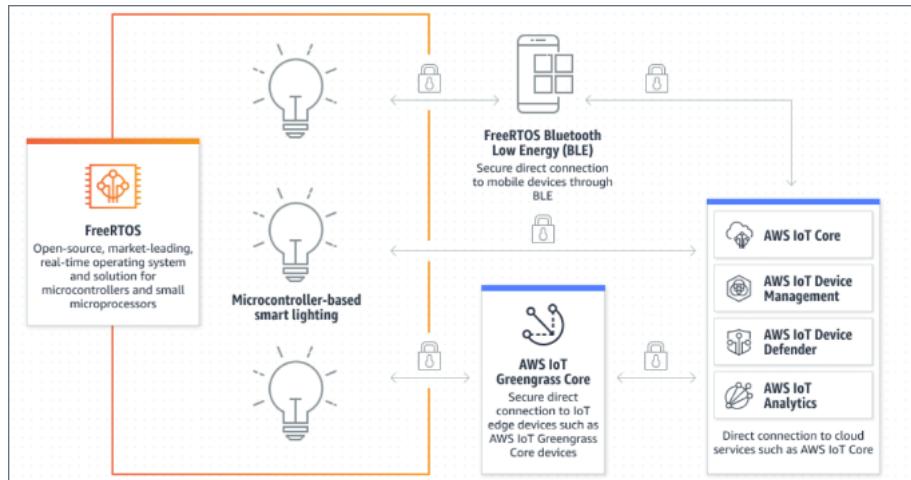


Figura 2.8: Uso de FreeRTOS

2.2.1. Características

2.2.1.1. Conectividad

Cuando los dispositivos FreeRTOS están conectados a la red local, se pueden conectar fácilmente a un dispositivo de borde local, como los dispositivos AWS IoT Greengrass Core, en la misma red local mediante la API de detección de AWS IoT Greengrass. FreeRTOS facilita que los dispositivos inicien el proceso de detección y se conecten al dispositivo de AWS IoT Greengrass Core deseado. La conectividad local permite que los dispositivos se comuniquen entre sí en el borde, como, por ejemplo, el sistema de seguridad de un edificio de oficinas que abre la puerta al deslizar una placa.

La conectividad en la nube le permite recopilar datos fácilmente y tomar medidas en dispositivos basados en microcontroladores para usar en aplicaciones de IoT y con otros servicios en la nube de AWS. Puede conectar dispositivos de FreeRTOS con AWS IoT Core mediante mensajería basada en MQTT o HTTP. MQTT es un protocolo ligero que deja poca huella y permite una comunicación eficaz en dispositivos limitados basados en microcontroladores. FreeRTOS facilita la incorporación de interfaces de bibliotecas estándar ajenas al proveedor. MQTT es un protocolo ligero que deja poca huella y permite una comunicación eficaz en dispositivos limitados basados en microcontroladores. La conectividad en la nube permite que dispositivos, como los contadores de electricidad inteligentes, devuelvan información acerca del consumo y analicen esos datos con otros servicios de AWS como AWS IoT Analytics.

2.2.1.2. Protección del dispositivo, la conexión y las actualizaciones

FreeRTOS incluye bibliotecas de seguridad, como la conexión segura a la nube, la autenticación de certificados, la administración de claves y una característica de firma de código.

FreeRTOS administra una conexión segura a la nube mediante Transport Layer Security (TLS v1.2). La biblioteca de TLS implementa una capa de abstracción

del protocolo TLS que proporciona privacidad e integridad de los datos entre dos aplicaciones en comunicación. Para poder conectarse al agente MQTT de AWS IoT Core, es necesaria la autenticación del certificado del cliente de TLS. FreeRTOS proporciona una capa de abstracción para la administración de objetos criptográficos y las operaciones de firma de claves privadas como característica de administración de claves. Los objetos criptográficos se guardan en un almacén dedicado o en la memoria flash del microcontrolador principal si no hay almacenamiento dedicado disponible. Puede utilizar la consola de AWS IoT Device Management con dispositivos FreeRTOS para la firma de código. La función de firma de código verificará la imagen firmada del dispositivo para garantizar que su código de dispositivo no corra peligro durante las implementaciones y las actualizaciones.

2.2.1.3. Actualizaciones inalámbricas

Puede utilizar AWS IoT Device Management con dispositivos FreeRTOS para obtener una solución de actualización inalámbrica integrada. FreeRTOS logra que la implementación de actualizaciones inalámbricas en dispositivos basados en microcontroladores utilice menos memoria al comunicar esas actualizaciones mediante una única conexión TLS, compartida con otras comunicaciones de AWS IoT Core. Debe facilitar una imagen de firmware, seleccionar los dispositivos que desea actualizar, seleccionar un método de firma de código y programar la actualización, todo ello en la consola de AWS IoT Device Management. Puede utilizar las actualizaciones OTA para implementar actualizaciones de seguridad, corrección de errores y actualizaciones de firmware nuevo en dispositivos en el terreno.

2.2.2. Casos de uso:

- FreeRTOS es compatible con la programación de tareas en múltiples núcleos de procesadores idénticos, como máquinas expendedoras activadas por IoT que muestran publicidades de video y las tareas de selección de bebidas en simultáneo.
- Recolecta datos sobre el rendimiento del sistema industrial de dispositivos y lleva a cabo acciones locales críticas en tiempo real para prevenir interrupciones.
- Utiliza AWS IoT Device Management con dispositivos FreeRTOS para obtener una solución de actualización inalámbrica OTA integrada para mantener y actualizar sus dispositivos de manera segura.

2. CAPITULO 2

3

CAPITULO 3

**Payta Sarabia Jordan Jorge
Ramses Felix Prado**

3.1. Introducción

En este capítulo se describen los requerimientos necesarios para el diseño, implementación y correcto funcionamiento del sistema de detección de baches. Estos requerimientos han sido definidos con base en los objetivos establecidos y buscan garantizar que el sistema cumpla con las necesidades de los usuarios y las especificaciones técnicas requeridas.

El sistema integra hardware y software, donde los componentes electrónicos y los desarrollos informáticos interactúan para proporcionar una solución eficiente, confiable y escalable. A continuación, se presentan los requerimientos funcionales y no funcionales, así como los recursos tecnológicos, humanos y operativos necesarios para la implementación del sistema.

3.2. Requerimiento Funcionales

3.2.1. Detector de Baches

- El sistema debe detectar baches en tiempo real utilizando el sensor MPU6050 para identificar cambios bruscos en la aceleración.
- Clasificar la gravedad del bache detectado como leve, moderado o grave, basado en los valores obtenidos por el sensor.

3.2.2. Geolocalización

- Integrar un módulo GPS para registrar la ubicación exacta del bache detectado.
- Almacenar las coordenadas geográficas del bache en la base de datos.

3.2.3. Almacenamiento en la nube

- Los datos recopilados (gravedad, ubicación, hora y fecha) deben ser enviados y almacenados en Firebase para su posterior consulta.

3.2.4. Aplicación móvil

- Permitir que los usuarios visualicen los baches registrados en un mapa interactivo dentro de la aplicación móvil desarrollada en React Native.
- Ofrecer información detallada de cada bache, como su ubicación, gravedad y fecha de registro.

3.2.5. Notificaciones

- Generar alertas o notificaciones visuales y auditivas en el sistema físico cuando se detecte un bache, a través de una pantalla OLED.

3.2.6. Requerimientos No Funcionales

3.2.7. Escalabilidad:

- El sistema debe permitir la integración de nuevos sensores o módulos en caso de que se requiera ampliar su funcionalidad.

3.2.8. Fiabilidad:

- Asegurar que el sistema detecte y registre baches con una precisión mínima del 80%

3.2.9. Usabilidad:

- La aplicación móvil debe ser intuitiva y accesible, permitiendo a los usuarios interactuar fácilmente con los datos registrados.

3.2.10. Portabilidad:

- El sistema físico debe ser compacto y fácilmente instalable en cualquier tipo de vehículo.

3.2.11. Tiempo de respuesta:

- Garantizar que el sistema procese y registre los datos en menos de 5 segundos desde la detección del bache.

3.3. Recursos Necesarios

3.3.1. Hardware

- Microcontrolador ESP32 Wroom S3.
- Sensor de movimiento MPU6050.
- Módulo GPS NEO-6M.
- Pantalla OLED SSD1306.
- Cables y protoboard para conexión.

3.3.2. Software

- Entorno de desarrollo ESP-IDF para la programación del ESP32.
- Firebase para almacenamiento en la nube.
- React Native para el desarrollo de la aplicación móvil.
- Bibliotecas necesarias: ssd1306, mpu6050, y axios.

3.4. Diagrama de caso de uso

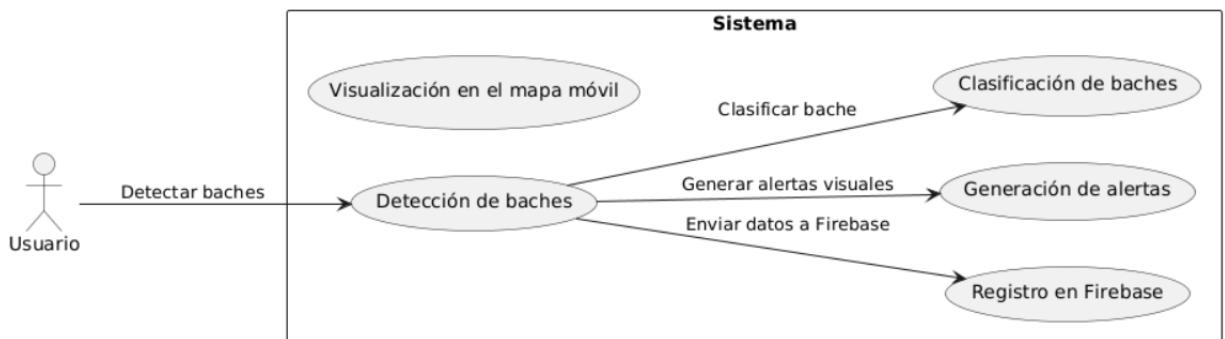


Figura 3.1: Diagrama de caso de uso

Este diagrama sirve para representar las interacciones entre los actores (usuarios o sistemas externos) y las funcionalidades principales del sistema. En el caso de tu proyecto, los actores incluyen:

Usuario del Sistema: Es el ESP32, que detecta baches y procesa la información.

3.5. Diagrama de secuencia

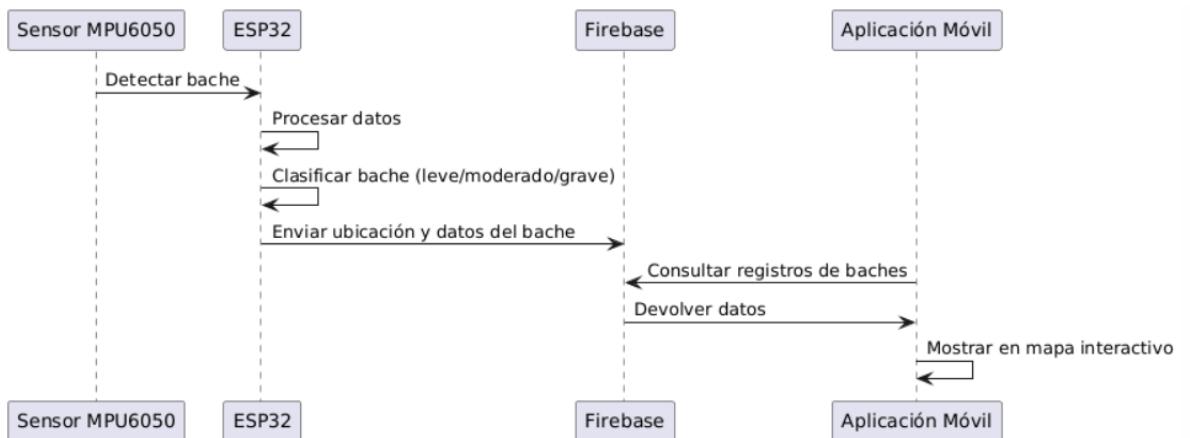


Figura 3.2: Diagrama de secuencia

El diagrama de secuencia describe cómo se comunican los diferentes componentes del sistema entre sí en el tiempo. Muestra el flujo de mensajes o datos entre:

- Los sensores (MPU6050 y GPS).
- El ESP32, que procesa la información.
- Firebase, que actúa como base de datos en la nube.
- La aplicación móvil, que consume los datos desde Firebase y los muestra al usuario.

Este diagrama es clave para entender el orden en que ocurren las operaciones, asegurando que el flujo lógico de datos sea consistente y eficiente.

3.6. Diagrama de actividades

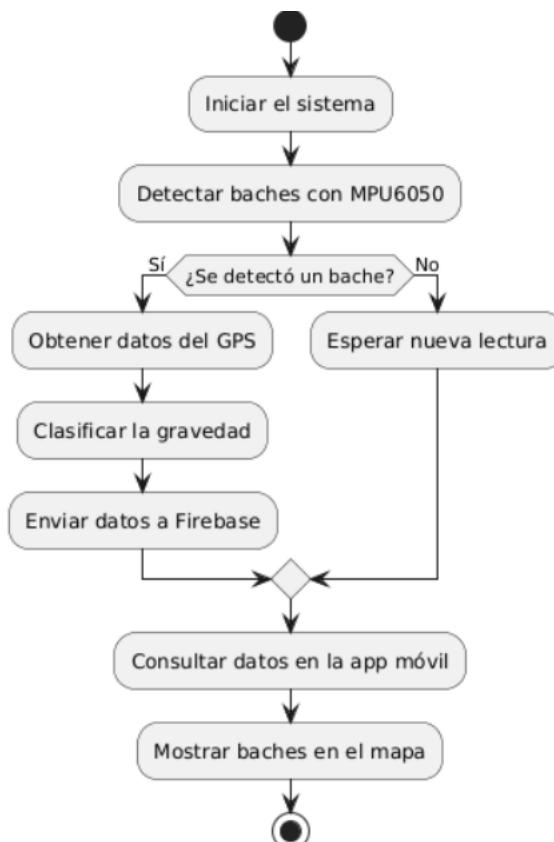


Figura 3.3: Diagrama de actividades

Ilustra el flujo de trabajo de alto nivel del sistema. Describe las acciones que el sistema realiza, así como las decisiones que toma en cada etapa.

3.7. Diagrama de componentes

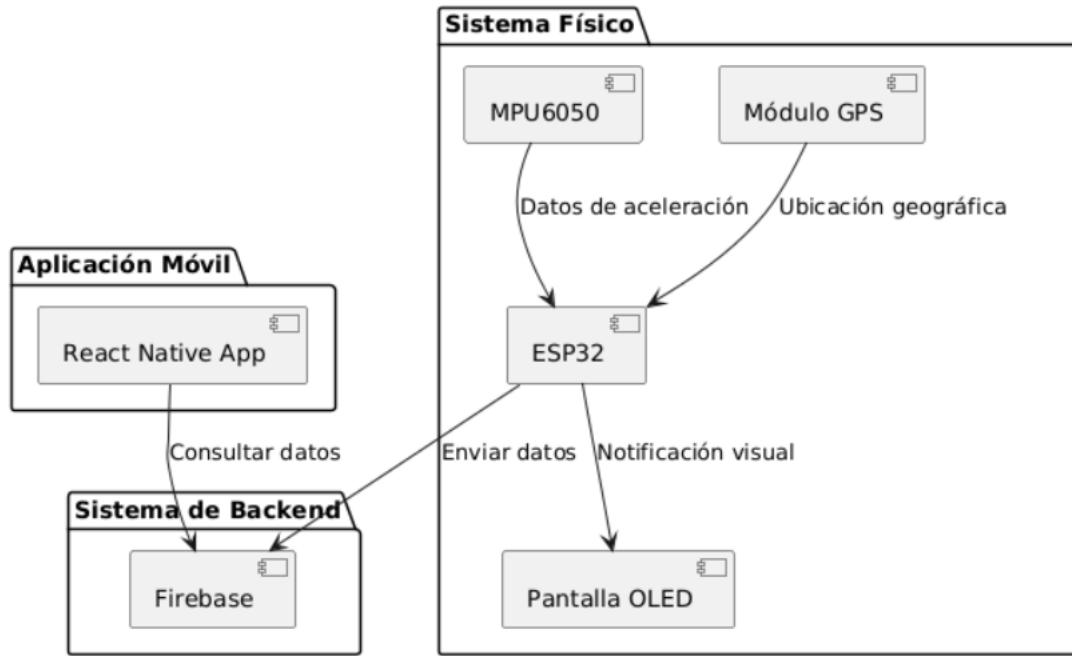


Figura 3.4: Diagrama de componentes

Representa los elementos físicos y lógicos del sistema y cómo están conectados entre sí.

- Hardware: ESP32, MPU6050, módulo GPS, buzzer y OLED.
- Software: Firebase y la aplicación móvil.

Este diagrama destaca cómo las piezas del sistema trabajan juntas, proporcionando una visión integral del diseño arquitectónico.

4

CAPITULO 4

Ramses Felix Prado
Jordan Jorge Payta Sarabia

4.1. Diseño Placa en Eagle

Se muestra el diseño hecho con el programa "Eagle" donde se hace uso de su gran facilidad de realizar las conexiones y mostrar los componentes.

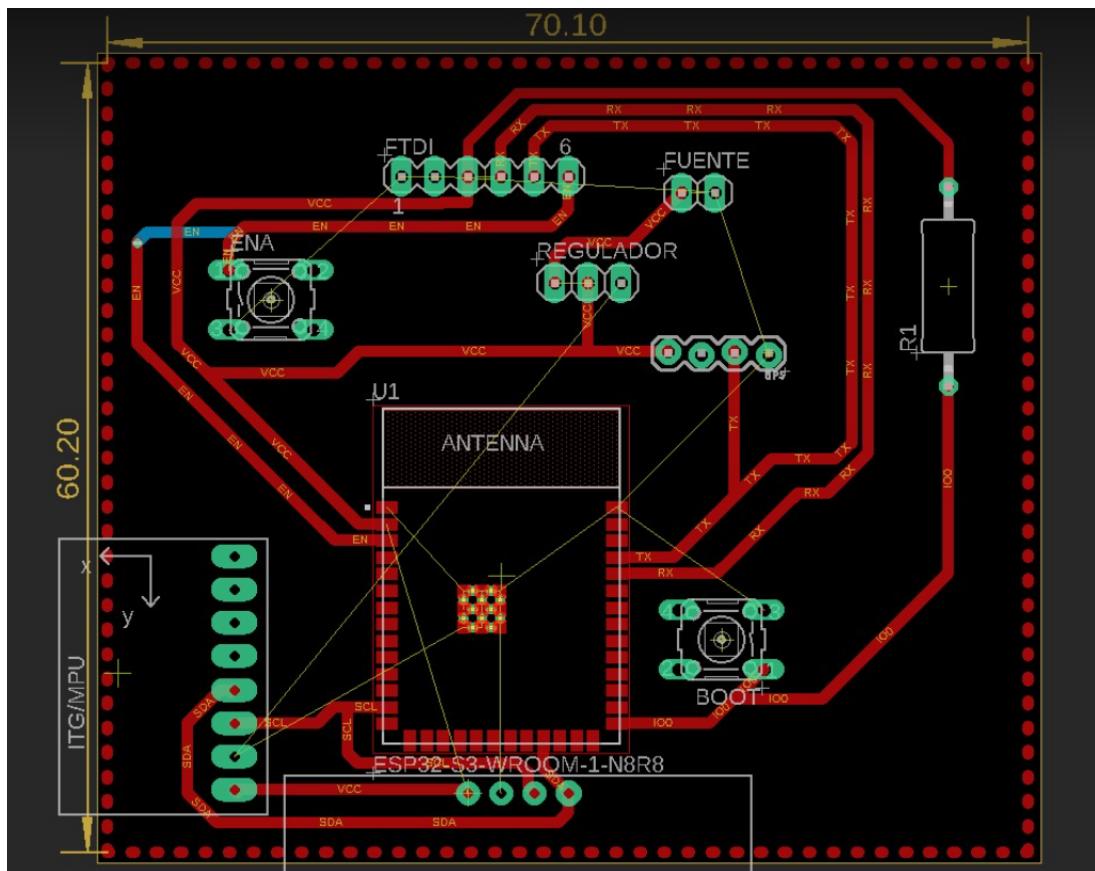


Figura 4.1: Diseño del circuito

El diseño realizado en Eagle representa el circuito electrónico encargado de implementar el sistema de detección de baches en carreteras. Este diseño integra los

principales componentes necesarios para la funcionalidad del sistema, organizados y conectados de manera eficiente para garantizar su operación.

4.2. Diseño electrónico

El diseño presentado corresponde a un esquema electrónico que integra diferentes componentes esenciales para el desarrollo de un sistema de detección y registro de baches utilizando un microcontrolador ESP32-Wroom S3 que manda las señales para el funcionamiento del prototipo.

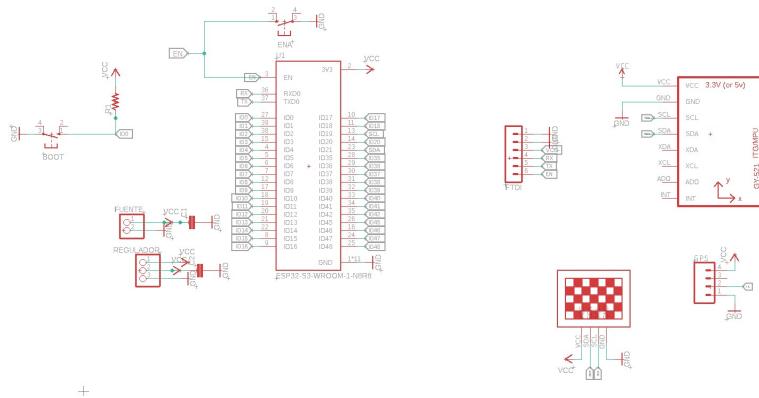


Figura 4.2: Diagrama esquemático de conexiones del detector de baches

Microcontrolador ESP32: Es el núcleo del sistema, encargado de procesar los datos provenientes de los sensores y de establecer comunicación con otros dispositivos o servicios en la nube. Se encuentra alimentado con una fuente regulada de 3.3V.

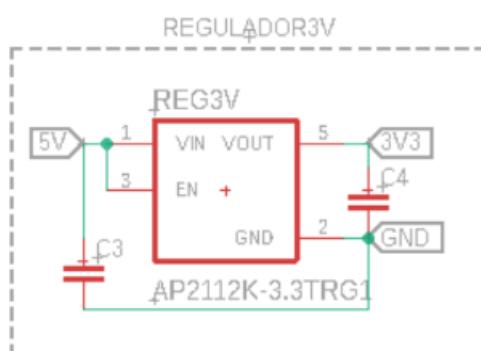


Figura 4.3: Diagrama esquemático del regulador de 3V

En la Figura 4.3 Incluye un regulador de voltaje que transforma la entrada de energía a 3.3V, necesaria para alimentar el ESP32 y los sensores conectados

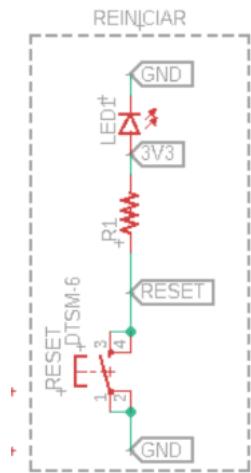


Figura 4.4: Diagrama esquemático del botón de RESET

En la figura 4.4 se incluye la parte del reset para el microcontrolador que nos permitirá reiniciar el ESP32 al momento de programarlo.

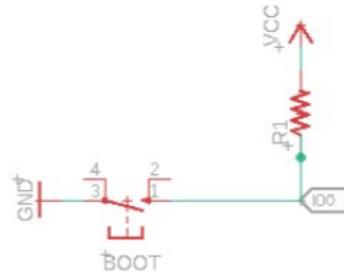


Figura 4.5: Diagrama esquemático del botón de BOOT

En la imagen anterior tenemos la estructura de conexión del botón BOOT que es el que nos permitirá subir el código al compilar el código al Esp32.

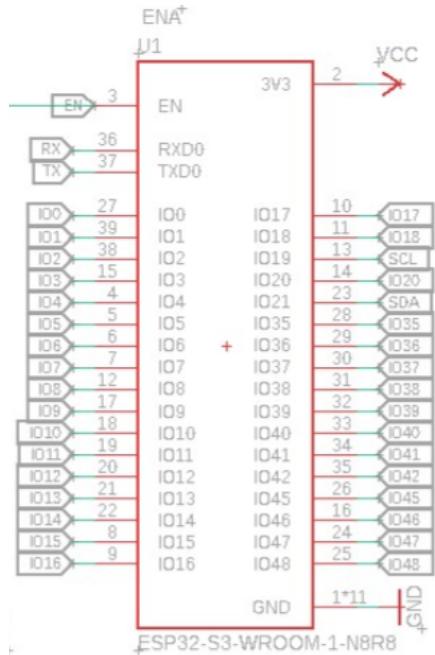


Figura 4.6: Diagrama esquemático del esp32-Wroom-s3

En la figura 4.4 se encuentra la parte del procesador en forma de chip, que en este caso se utilizó un microcontrolador ESP32 que nos permitirá recopilar los datos de los sensores para obtener la retroalimentación y dar la señal de activación para los motores.

4.3. Diseño de la Aplicación Móvil

El diseño de la aplicación móvil es una parte fundamental del proyecto, ya que permite a los usuarios finales interactuar con el sistema, visualizar los datos recolectados por el dispositivo de detección de baches y tomar decisiones informadas sobre las condiciones de las carreteras. A continuación, se describe el diseño y las características principales de la app móvil:

Funcionalidades Principales

- La app móvil está diseñada para cumplir con las siguientes funcionalidades:
- Visualización de Baches en el Mapa: Muestra los baches detectados por el sistema en un mapa interactivo, indicando su ubicación geográfica exacta mediante marcadores.
- Registro de Nuevos Baches: La aplicación permite recibir y almacenar en tiempo real los datos enviados desde el ESP32, como la gravedad del bache y su ubicación.
- Consulta de Historial: Proporciona a los usuarios un historial de los baches detectados, con detalles como fecha, hora, ubicación y gravedad.
- Notificaciones en Tiempo Real: La app envía alertas al usuario cuando se detecta un bache grave en las cercanías, mejorando la seguridad vial.



Figura 4.7: Pagina oficial

Se muestra en la parte de abajo las diferentes navegaciones que incluye la aplicación móvil, para que el usuario se le facilite para navegar la app.



Figura 4.8: Navegador de estadísticas

Proporciona a los usuarios un historial de los baches detectados, con detalles como fecha, hora, ubicación y gravedad.

4. CAPITULO 4



Figura 4.9: Mapa donde se registra los baches

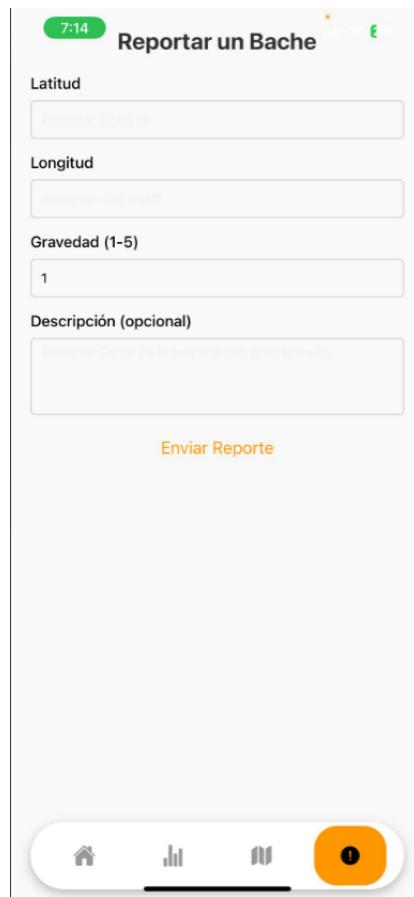


Figura 4.10: Tab para reportar bache



Figura 4.11: Datos e Información de los reportes

4.3.1. Tecnologías utilizadas

- Framework React Native: La app se desarrolló utilizando React Native para asegurar compatibilidad multiplataforma (iOS y Android) y un rendimiento eficiente.
- API de Google Maps: Se integra para proporcionar un mapa interactivo que permite visualizar la ubicación de los baches detectados.
- Conexión con Firebase: Firebase actúa como la base de datos en tiempo real para almacenar y recuperar los datos de los baches detectados.
- WebSocket para Comunicación en Tiempo Real: La app utiliza WebSockets para recibir notificaciones inmediatas de nuevos registros en la base de datos.

4.3.2. Flujo de Navegación

- Inicio de Sesión: El usuario inicia sesión en la app.
- Visualización en el Mapa: Se cargan los datos en tiempo real desde Firebase y se muestran en el mapa interactivo.
- Selección de Marcador: El usuario puede seleccionar un marcador para ver detalles del bache detectado.

- Notificaciones: Cuando se detecta un nuevo bache grave, la app muestra una alerta en tiempo real.
- Historial: El usuario puede navegar a una pantalla de historial para revisar baches detectados en días anteriores.

El diseño de la app móvil se centra en la simplicidad, accesibilidad y funcionalidad. Esto garantiza que los usuarios puedan interactuar fácilmente con el sistema, acceder a los datos relevantes de forma rápida y estar informados sobre las condiciones de las carreteras en tiempo real. La integración con Firebase y WebSocket permite que los datos fluyan de manera eficiente entre el hardware (ESP32) y el usuario final.

4. CAPITULO 4

5

CAPITULO 5

Payta Sarabia Jordan Jorge
Ramses Felix Prado

5.1. Estudio experimental y pruebas

5.1.1. Placa del proyecto

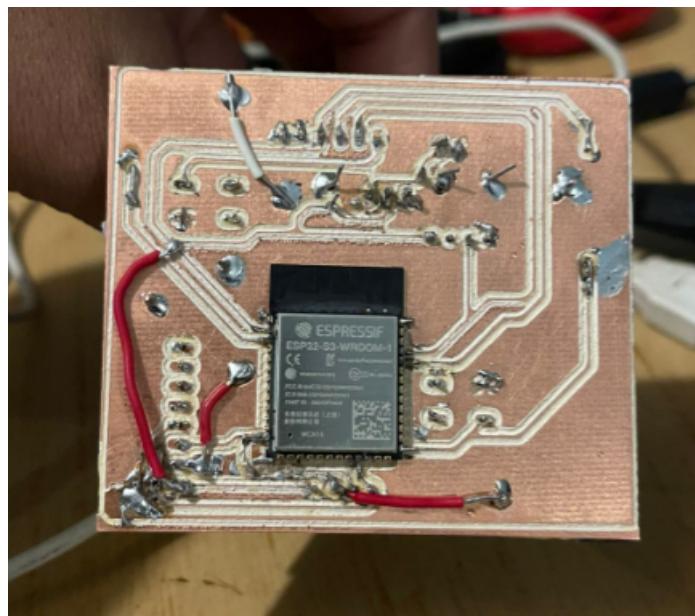


Figura 5.1: Prototipo del circuito

La imagen presentada corresponde a la placa de circuito impreso (PCB) final que fue diseñada y ensamblada como parte del sistema de detección de baches. Este PCB integra el microcontrolador ESP32-S3-WROOM-1, el cual es el núcleo del sistema. La placa fue fabricada utilizando técnicas de grabado químico y posteriormente ensamblada manualmente con los componentes necesarios para su funcionamiento.

El diseño incluye componentes como reguladores de voltaje, conectores de pines para la conexión con sensores externos, y líneas de alimentación para garantizar la operación estable del ESP32. Las conexiones de los pines del ESP32 están adaptadas para facilitar el enlace con el MPU6050 (acelerómetro y giroscopio) y el módulo GPS.

5.1.2. Firebase

Envío de Datos a Firebase: Una vez que se detectaba un bache, el ESP32 enviaba los datos (gravedad del bache y coordenadas GPS) a la base de datos en Firebase. Se verificó que los datos enviados aparecieran correctamente en la base de datos y que fueran accesibles desde la app móvil. **Resultados de Integración:** Los datos enviados por el ESP32 se almacenaron correctamente en tiempo real en Firebase. Se realizaron pruebas con diferentes niveles de señal Wi-Fi para garantizar la estabilidad de la conexión.



Figura 5.2: Funcionamiento de la base de datos

Una vez terminada la conexión del firebase con el esp32, se realiza las conexiones de los componentes, ver su funcionamiento, y ver que si se muestra en la pantalla OLED.

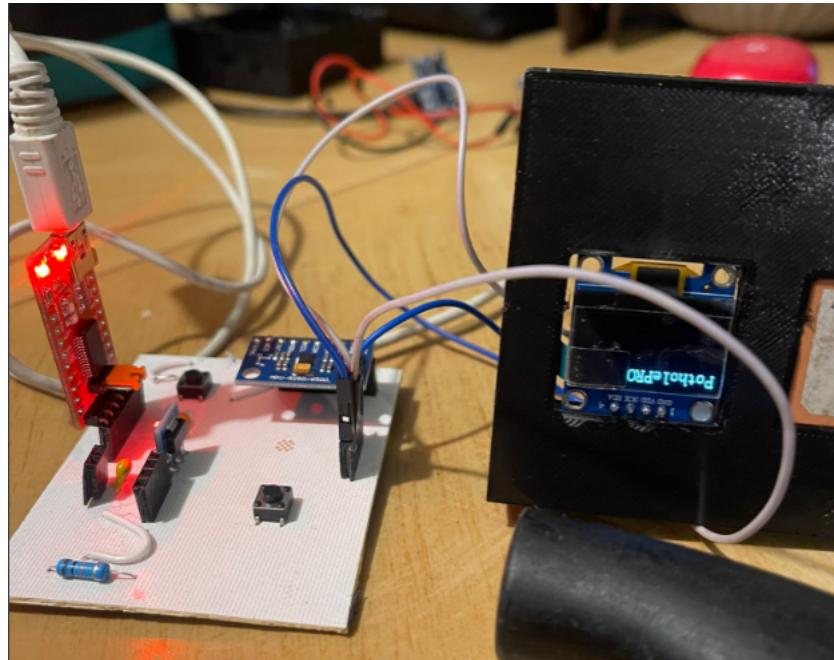


Figura 5.3: Funcionamiento de la placa

Los datos registrados en Firebase se visualizaron correctamente en la app móvil, incluyendo la ubicación de los baches en el mapa. Se realizaron pruebas con múltiples usuarios para verificar la sincronización en tiempo real.

5. CAPITULO 5

El diseño físico del Detector de Baches incluye una carcasa protectora, creada para albergar los componentes electrónicos esenciales del sistema. Este modelo fue diseñado y fabricado con el propósito de optimizar la protección y funcionalidad de los componentes electrónicos, así como de proporcionar un diseño compacto y portátil.

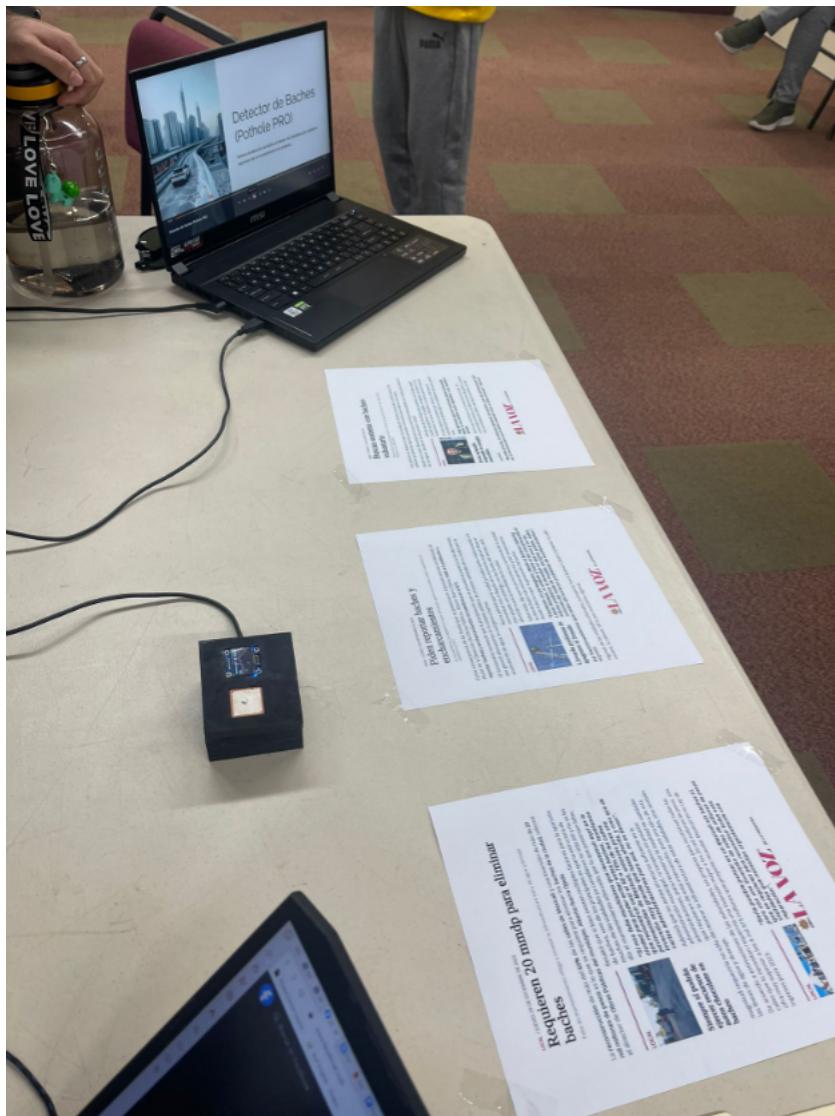


Figura 5.4: Producto Final del proyecto

Se mandaron a hacer diseños y modelos en 3D para nuestro proyecto, los resultados fueron muy buenos, cumplimos con nuestros objetivos

6

CAPITULO 6

**Payta Sarabia Jordan Jorge
Ramses Felix Prado**

6.1. Conclusión

El desarrollo del proyecto Detector de Baches representa una solución innovadora y tecnológica para abordar un problema crítico en el mantenimiento vial. A través de la integración de hardware como el ESP32, el sensor MPU6050 y el módulo GPS, junto con una infraestructura basada en Firebase y una aplicación móvil desarrollada en React Native, este sistema permite la detección, registro y visualización en tiempo real de irregularidades en la superficie de las carreteras.

El diseño del sistema no solo contribuye a mejorar la seguridad vial al alertar a los conductores sobre la presencia de baches, sino que también proporciona una herramienta valiosa para las autoridades responsables del mantenimiento vial, permitiendo priorizar y planificar las reparaciones de manera eficiente. Además, la implementación de un modelo basado en IoT garantiza una comunicación fluida entre los dispositivos, el almacenamiento en la nube y la aplicación móvil, creando un ecosistema confiable y escalable.

La experimentación y pruebas del dispositivo han demostrado su efectividad en escenarios reales, cumpliendo con los objetivos propuestos del proyecto. Este enfoque no solo aborda los riesgos de accidentes causados por baches, sino que también minimiza el impacto económico que estos generan, mejorando la calidad de las carreteras de manera significativa. El proyecto, con su diseño compacto y eficiente, sienta las bases para futuras innovaciones en el ámbito de IoT aplicado al transporte y mantenimiento vial.

6.2. Referencias

- Espressif Systems. (2023). ESP32 Technical Reference Manual. Recuperado de: <https://www.espressif.com>
- Firebase Documentation. (2023). Cloud Firestore Documentation. Recuperado de: <https://firebase.google.com/docs>
- React Native Community. (2023). React Native Documentation. Recuperado de: <https://reactnative.dev/docs>
- Bosch Sensortec. (2023). MPU6050 Product Specification. Recuperado de: <https://www.bosch-sensortec.com>
- GeeksforGeeks. (2023). Introduction to MQTT Protocol in IoT. Recuperado de: <https://www.geeksforgeeks.org>
- Autodesk. (2023). Fusion 360 Design for Electronics. Recuperado de: <https://www.autodesk.com>
- Node.js. (2023). Node.js v20.11 Documentation. Recuperado de: <https://nodejs.org>
- Mosquitto MQTT. (2023). Eclipse Mosquitto Documentation. Recuperado de: <https://mosquitto.org>
- ResearchGate. (2022). Smart Detection Systems for Road Safety: A Review of Technologies and Approaches. Recuperado de: <https://www.researchgate.net>
- Arduino. (2023). Arduino Uno Documentation. Recuperado de: <https://www.arduino.cc>