UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

Departamento de Ingeniería en Computación e Informática.

Sistema antirrobo con notificación y aviso inteligente

Autor(es): Renato Almeyda.
Jeany Aravena
Bastián Cruz.
Josue Sucso.
Académico: Diego Aracena.

Historial de Cambios.

Fecha	Versión	Descripción	Autor(es)
14/10	0.1	Versión preliminar del formato	Renato Almeyda Jeany Aravena Bastián Cruz Josue Sucso
28/10	1.0	Desarrollo de requerimientos solicitados	Bastián Cruz Josue Sucso
22/11	2.0	Integración de los modelos de diseño(caso de uso general, diagrama de clases y secuencia) y la sección de herramientas y técnicas utilizadas	Bastián Cruz Josue Sucso
24/11	2.1	Errores corregidos e integración de las secciones faltantes “conclusiones” y “referencias”	Bastián Cruz Josue Sucso

Tabla de contenidos.

Índice de tablas

Índice de figuras

I. Panorama General

1.0 Resumen del proyecto:

En la actualidad, el robo de vehículos representa un problema constante que afecta la seguridad y economía de los propietarios. Este proyecto propone una solución tecnológica basada en tecnología IoT mediante el desarrollo de un sistema de notificación inteligente que permite detectar y reportar de forma automática la conducción no autorizada de un vehículo, es decir, informar donde efectivamente el vehículo fue sustraído del dueño

Este sistema se implementa sobre una Raspberry Pi 4, la cual integra un acelerómetro, una cámara para lectura de códigos QR dinámicos, un display LCD para su interfaz, además de unas luces LED de manera complementaria y una conexión con la aplicación móvil del propietario. A través de esta infraestructura, se busca ofrecer un método moderno de autenticación vehicular y alerta inmediata, contribuyendo a la prevención del robo y al fortalecimiento de la seguridad automotriz.

1.1 Propósito del proyecto:

Proveer a los propietarios de vehículos una herramienta activa e inmediata para mitigar el riesgo de robo, transformando la seguridad automotriz de un enfoque reactivo a uno preventivo y de respuesta en tiempo real mediante la integración de la tecnología IoT.

1.2 Alcances del proyecto:

- Hardware: Raspberry Pi 4, sensor de cámara, sensor acelerómetro, módulo de red, display LCD, luces LED complementarias.
- Software: app móvil, lectura QR, base de datos y sistema de notificaciones.
- Simulación VR: entorno Unity para visualización del contexto real situado.

1.3 Objetivos específicos:

- Detectar movimiento vehicular mediante sensor acelerómetro.
- Implementar autenticación QR dinámica con cámara.
- Enviar notificaciones al propietario del vehículo.
- Crear simulación VR del sistema en el vehículo.

1.4 Suposiciones y restricciones:

1.4.1 Suposiciones:

- El propietario del vehículo dispone de un teléfono con conexión a Internet y la aplicación instalada.

- La cámara de la Raspberry Pi 4 puede leer el código QR dinámico en condiciones normales de luz.
- El acelerómetro detecta correctamente el movimiento del vehículo sin error alguno.
- El sistema cuenta con una fuente de energía estable, sea batería o conectada por cable.
- El entorno de simulación en Unity representa de forma adecuada el contexto real.

1.4.2 Restricciones:

- El desarrollo debe completarse dentro de los tiempos solicitados.
- Solo se cuenta con una Raspberry Pi 4 y sensores limitados.
- El sistema depende de conexión Wi-Fi; no incluye comunicación móvil.
- El presupuesto es limitado.

1.5 Entregables del proyecto:

Revisión	Entregables	AS	AQ	ET
Parte del 1er informe	Formulación del Proyecto Corregido	X		
Parte del 2do informe	Diagramas de diseño, herramientas y técnicas utilizadas, errores del 1er informe corregidos	X		

Tabla 1: Entregables del proyecto

II. Organización del proyecto

2.1 Personal y entidades externas

- Jefe del proyecto: Renato Almeyda
- Programador(es): Renato Almeyda , Jeany Aravena
- Diseñador: Bastián Cruz
- Ensamblador: Josue Sucso
- Documentador: Bastián Cruz

2.2 Roles y responsabilidades

- **Jefe del proyecto:** Encargado de coordinar y supervisar el correcto avance de todos los procesos que componen el desarrollo del proyecto. Es representante del equipo de trabajo ante los profesores encargados del ramo y el resto de los equipos.

- **Diseñador:** Se encarga de diseñar la interfaz de usuario y la experiencia de usuario de la *aplicación móvil de alerta*. Es responsable de crear el flujo de notificaciones que recibe el propietario.

- **Programador(es):** Encargados de escribir el código que se ejecuta en el Raspberry Pi 4. Su trabajo es asegurar que el dispositivo pueda leer correctamente la información de los sensores, tomar las decisiones de seguridad (como verificar la velocidad y el código QR), y coordinar todas las partes del sistema para que funcionen de manera sincronizada.d

- **Ensamblador:** Encargado de la preparación física y el montaje del sistema. Sus responsabilidades incluyen: ensamblar el Raspberry Pi 4 con sus componentes (caja o chasis), cablear y conectar correctamente todos los sensores (velocidad, cámara) y periféricos al microcontrolador, y asegurar que la instalación del hardware sea funcional dentro del vehículo.

- **Documentador:** Responsable de la gestión integral de la información del proyecto. Esto incluye la elaboración de informes de avance y bitácoras, así como la creación y mantenimiento de la Wiki para documentación técnica. Es el encargado de administrar la Carta Gantt dentro de la plataforma Redmine para el seguimiento y control del proyecto.

2.3 Mecanismos de Comunicación

Canales internos: correo institucional, grupo de WhatsApp, Discord.
Documentación compartida: Google Drive y **GitHub** (repositorio del proyecto).

Comunicaciones y Estándares Técnicos:

Lenguajes de Programación:

- **Python:** Para la lógica del sistema en la Raspberry Pi 4 y el procesamiento de datos del acelerómetro y la cámara.
- **Kotlin:** Para el desarrollo de la aplicación móvil nativa (Android).

Lenguajes de Interfaz y Datos:

- **JSON (JavaScript Object Notation):** Como estándar de intercambio de datos para la comunicación entre el dispositivo IoT (Raspberry Pi) y la aplicación móvil.

Comunicaciones para XR (Realidad Extendida):

- **C#:** Utilizado en el motor Unity para la creación de una maqueta virtual del proyecto, la cual será desplegada en un dispositivo Meta Quest 3.

III. Planificación de los procesos de gestión.

3.1 Planificación inicial del proyecto

3.1.1 Planificación de estimaciones

Producto	Cantidad	Costo por unidad	Costo Total
Notebook(Uso)	4	$50.000	$200.000
Raspberry PI 4	1	$90.000	$90.000
Sensor Camara	1	$5.000	$5.000
Sensor acelerómetro	1	$5.000	$5.000
Grove LCD RGB Backlight	1	$15.000	$15.000
Tarjeta SD	1	$13.000	$13.000
Total			$328.000

Tabla 2: Planificación de estimaciones

3.1.2 Planificación de Recursos Humanos.

Roles	Tarifa x Hora
Jefe de proyecto	$12.000
Programador	$10.000
Diseñador	$8.500
Documentador	$5.000
Ensamblador	$6.000

Tabla 3: Planificación de recursos humanos

Miembro	Rol	Hora x mes	Meses de utilidad	Resultado	Pago Final
Renato Almeyda	Jefe de proyecto Programador	40 40	4 4	$1.920.000 $1.600.000	$3.520.000
Bastián Cruz	Diseñador Documentador	40 40	2 4	$680.000 $800.000	$1.480.000
Josue Sucso	Documentador Ensamblador	40 40	4 2	$800.000 $480.000	$1.280.000
Jeany Aravena	Programador	40	4	$1.600.000	$1.600.000
Total					$7.880.000

Tabla 4: Planificación de recursos totales

Costo total: $8.208.000

3.2 Lista de actividades (carta Gantt)

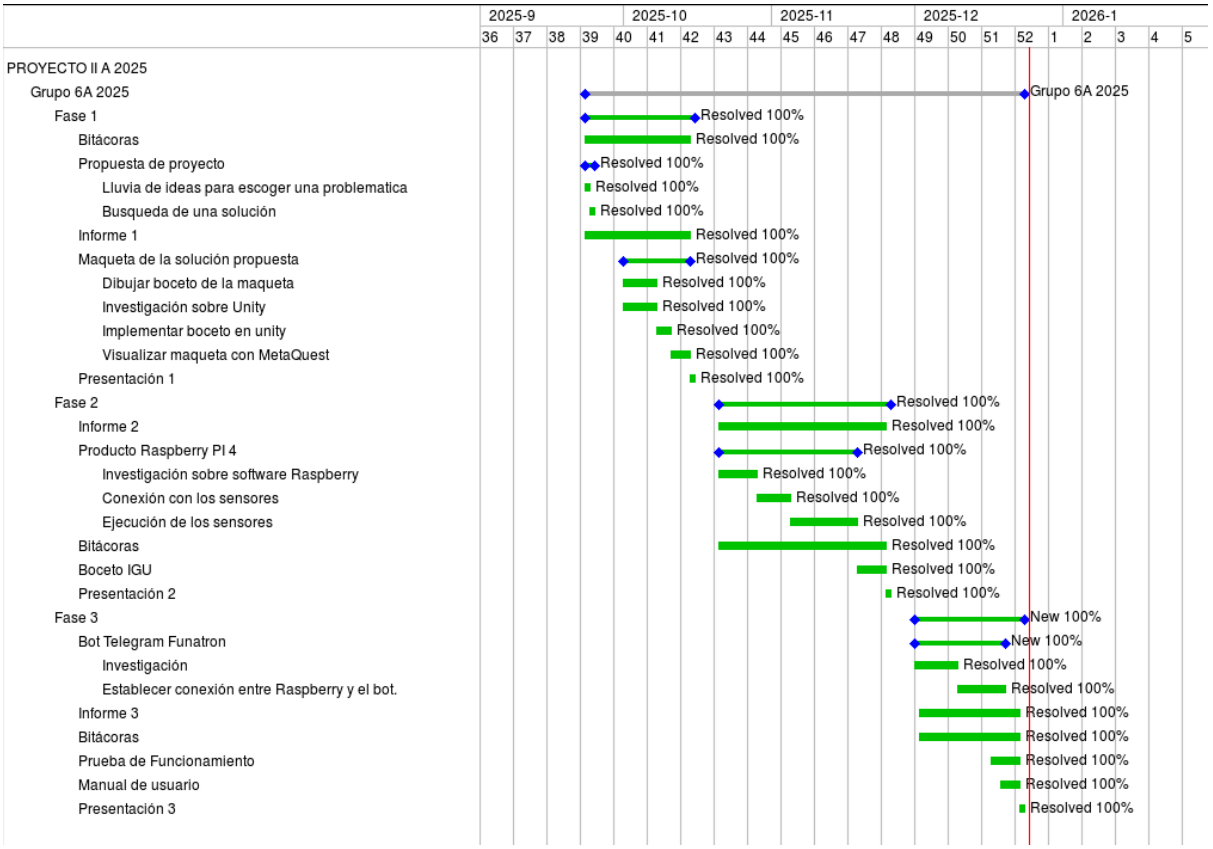

Figura 1: Carta Gantt

3.2.1 Actividades de trabajo

1) **Planificación del proyecto:**

- Revisión del formato y requisitos del plan de proyecto.
- Definición del problema, propósito y objetivos generales.
- Asignación de roles y responsabilidades dentro del equipo.
- Identificación de los recursos materiales (Raspberry Pi, sensores, cámara, etc.).
- Elaboración del panorama general, suposiciones, restricciones y entregables.
- Estimación de tiempos y costos iniciales del proyecto.

2) **Ejecución y desarrollo del proyecto:**

- **Análisis:** investigar el funcionamiento de los sensores y componentes; definir los requerimientos funcionales y técnicos del sistema.
- **Diseño:** desarrollar el modelo de simulación del contexto en Unity y posteriormente, crear la arquitectura general del sistema, diseñar las interfaces de usuario (app móvil y display LCD)
- **Implementación:** programar la lógica de lectura del QR y acelerómetro en la Raspberry Pi 4; integrar la comunicación con la aplicación móvil y la base de datos; ensamblar físicamente los componentes.
- **Pruebas:** realizar pruebas unitarias, de comunicación y de integración del sistema; validar el correcto funcionamiento de las notificaciones y la autenticación dinámica.

3) **Cierre del proyecto:**

- Elaborar los manuales de usuario e instalación.
- Redactar la documentación técnica final.
- Preparar la presentación del proyecto y la entrega del prototipo.
- Firmar el acta de conformidad final del proyecto.

3.2.2 Asignación de tiempo

- Planificación del proyecto: 3 semanas
- Ejecución y desarrollo del proyecto: 11 semanas
- Cierre de proyecto: 2 semanas

3.3 Planificación de la gestión de riesgos.

- Niveles de impacto:

  - ☐ Catastrófico
  - ☐ Crítico
  - ☐ Marginal
  - ☐ Despreciable

Riesgo	Probabilidad de ocurrencia	Nivel de impacto	Acción Remedial
Retraso en la entrega de componentes (sensores, cámara, cables).	70%	2	Reasignar tareas de software mientras se espera el hardware. Avanzar en simulación y documentación.
Fallo en la compatibilidad de librerías entre sensores Grove y Raspberry Pi 4.	60%	2	Buscar alternativas compatibles o adaptar código con librerías Python (ej. smbus, OpenCV, grovepi).
Error en la lectura del QR por condiciones de luz o enfoque.	30%	2	Implementar prueba de iluminación adicional con LED blanco o ajustar contraste por software.
Fallas en la conexión Wi-Fi durante las pruebas.	15%	4	Utilizar red local o conexión directa entre Raspberry y smartphone.
Problemas de programación en la app móvil o en la comunicación con Raspberry.	50%	2	Realizar pruebas modulares (API y comunicación). Dividir tareas por submódulos.
Dificultad del equipo para coordinar horarios o tareas.	20%	3	Planificar reuniones semanales y utilizar Google Drive y WhatsApp para actualizaciones rápidas.
Sobrecarga académica o ausencia de un integrante clave.	20%	3	Reasignar tareas temporalmente y mantener documentación actualizada para continuidad del trabajo.
Problemas de Raspberry y sensores por motivos accidentales	20%	1	Manejar con cuidado el dispositivo Raspberry y cuidar que los sensores no se quemen.
Deriva o calibración incorrecta del acelerómetro	50%	3	Establecer una rutina de calibración inicial del sensor y

			aplicar filtros digitales (ej. Filtro Complementario o Kalman) por software para suavizar las lecturas.
Problemas de seguridad en la transmisión de datos (IoT)	40%	4	Implementar cifrado (SSL/TLS) en la comunicación entre el dispositivo (Raspberry Pi) y la aplicación móvil para proteger la información.

Tabla 5: Planificación de gestión de riesgos.

3.4 Poster Asociado.

Figura 2: Poster asociado.

V. Planificación de procesos técnicos

5.1 Modelo de procesos

5.1.1 Requisitos funcionales y no funcionales

5.1.1.1 Tabla de requerimientos funcionales

	Requisitos funcionales
RF 1	El sistema debe generar un código QR dinámico que cambie periódicamente para permitir la autenticación del propietario del vehículo.
RF 2	El sistema debe capturar el QR mediante la cámara de la Raspberry Pi para validar al usuario autorizado.
RF 3	El sistema debe verificar el QR escaneado contra el código generado y determinar si es válido o no.
RF 4	El sistema debe permitir al usuario autenticado desactivar la alerta desde la aplicación móvil con el código QR.
RF 5	El sistema debe leer los valores del acelerómetro (GY-6500/9250) para detectar la aceleración del vehículo, permitiendo identificar que hubo arranque y que está en movimiento.
RF 6	El sistema debe enviar una notificación en tiempo real a la aplicación móvil del propietario cuando se detecte movimiento no autorizado al no escanear el QR.
RF 7	El sistema debe permitir al usuario, tras confirmar un evento, enviar los datos a carabineros indicando fecha, hora, modelo del vehículo y una imagen de la persona arribada en el vehículo.
RF 8	El sistema debe almacenar las alertas confirmadas en un registro histórico accesible desde la app.
RF 9	El sistema debe activar un display LCD dependiendo del estado en que se encuentre.
RF 10	El sistema debe permitir registrar información del vehículo modelo, patente, color, detalles adicionales)

Tabla 6: Requisitos funcionales.

5.1.1.2 Tabla de requerimientos no funcionales

	Requisitos no funcionales
RNF 1	El sistema debe utilizar cifrado para todas las comunicaciones entre la Raspberry Pi y la aplicación móvil para proteger el código QR y la información sensible.
RNF 2	El tiempo de respuesta desde la captura del QR (RF 2) hasta la validación y determinación de si es válido (RF 3) no debe exceder los 2 segundos.
RNF 3	El sistema debe ser capaz de mantener un registro histórico (RF 8) de al menos 6 meses de alertas.
RNF 4	El código del sistema debe ser modular y estar bien documentado para permitir que un nuevo desarrollador pueda entender y modificar la lógica.
RNF 5	El sistema debe ser actualizable de forma remota (OTA - Over-The-Air) para el software de la Raspberry Pi sin requerir acceso físico al vehículo.
RNF 6	El consumo de energía del sistema, cuando está en modo de espera (monitoreando el acelerómetro), debe ser mínimo para no descargar la batería del vehículo.
RNF 7	El tiempo de envío de la notificación de movimiento no autorizado a la aplicación móvil no debe superar los 5 segundos desde que se detecta el movimiento.
RNF 8	La interfaz de usuario (UI) de la aplicación móvil debe ser intuitiva y de fácil navegación.

Tabla 7: Requisitos no funcionales.

5.1.2 Diagrama de caso de uso general

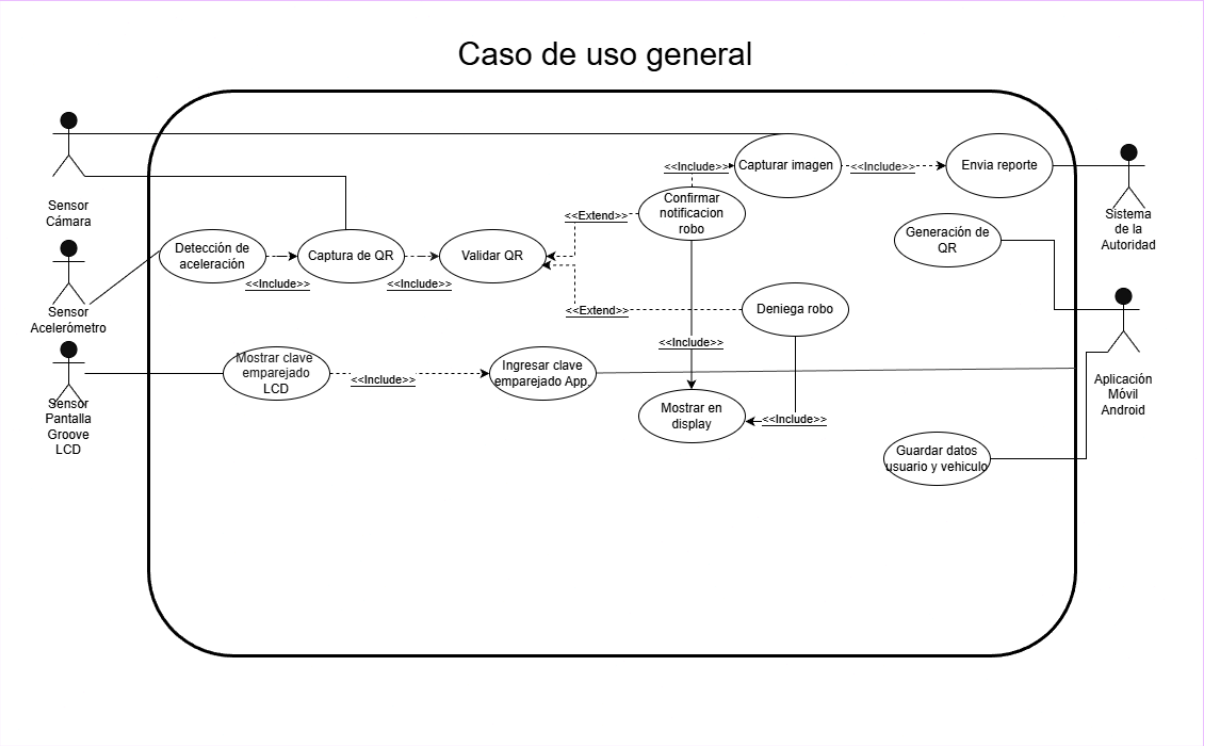

Figura 3: Diagrama de casos de uso general

5.1.3 Descripción de la arquitectura

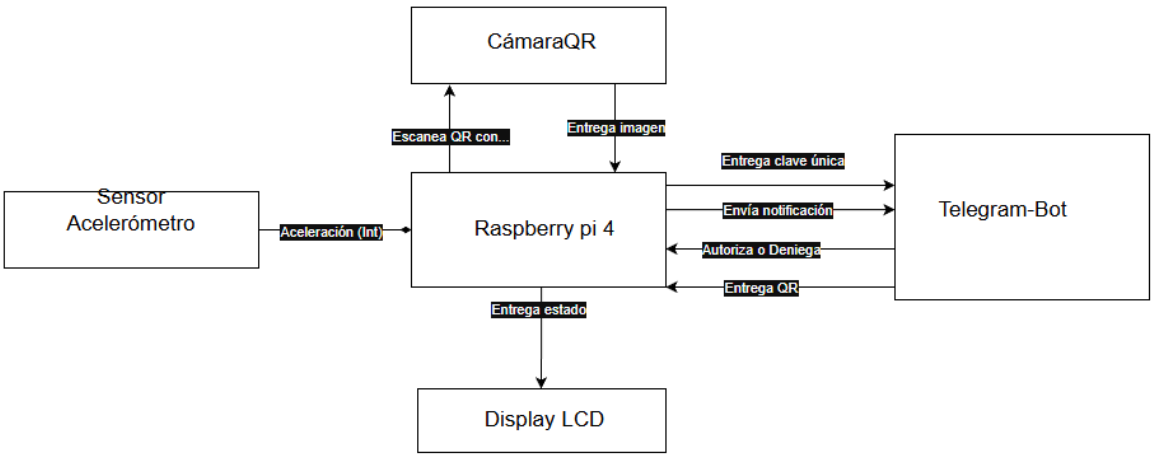

Figura 4: Diagrama de la arquitectura.

5.1.4 Diagrama de clases

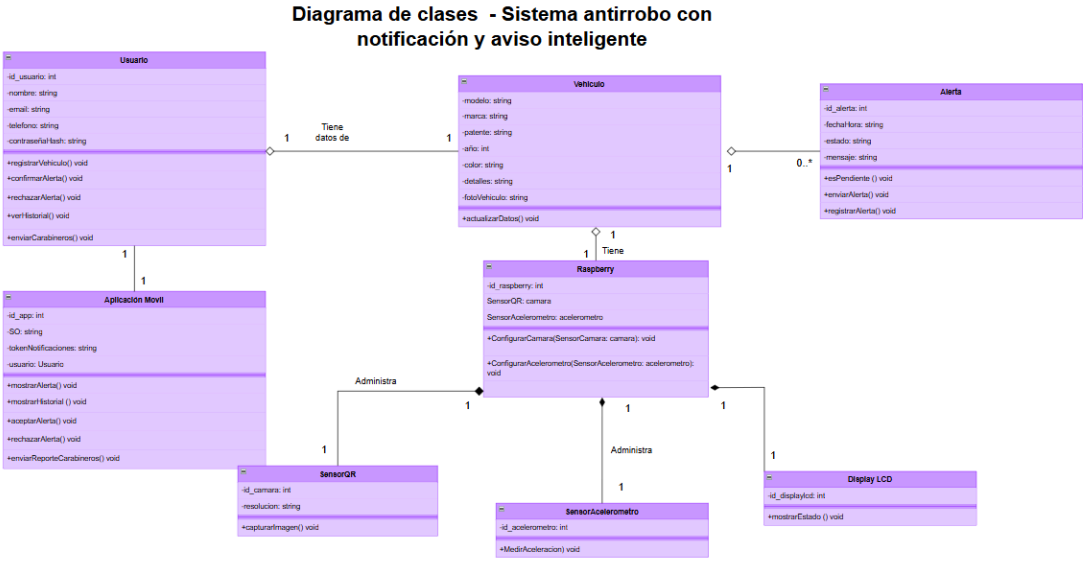

Figura 5: Diagrama de Clases.

5.1.5 Diagrama de secuencias

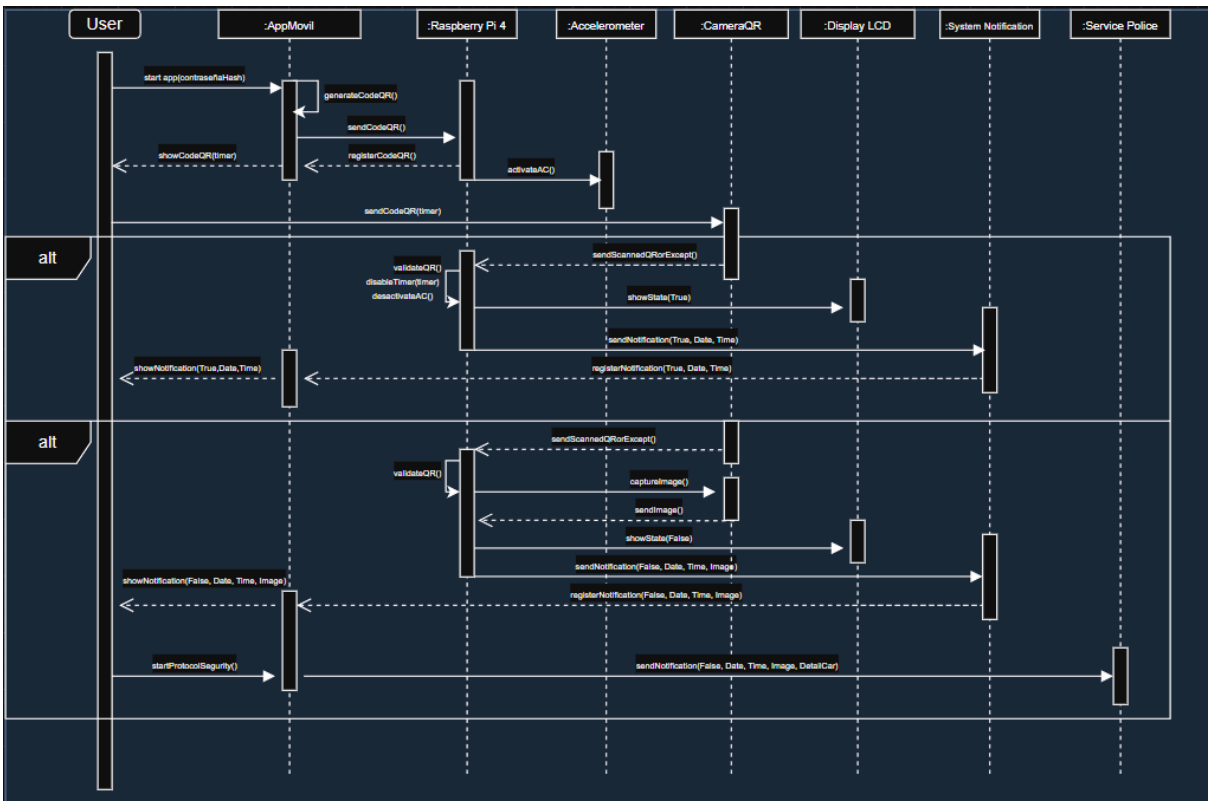

Figura 6: Diagrama de Secuencias

5.1.6 Descripción de la arquitectura con respecto a los modelos

- **Modelo de caso de uso general:**

  El diagrama de caso de uso general demuestra cómo la arquitectura del sistema integra los distintos componentes físicos y lógicos. Es así cómo se representan los sensores y actuadores del vehículo en este caso, acelerómetro, cámara y pantalla LCD, los cuales interactúan directamente con la Raspberry Pi como sistema central. Estos dispositivos permiten ejecutar funcionalidades como la detección de aceleración, la captura de QR, la validación del código y la alerta visual mediante display en base a la validación.

  Por otro lado se ubican los actores externos, la aplicación móvil, utilizada por el dueño para autenticarse, recibir notificaciones y confirmar un posible robo, entre otras funcionalidades, y el Sistema de la Autoridad, que recibe el reporte cuando el usuario confirma la alerta. Esto evidencia la comunicación entre el componente físico del sistema central y los sistemas externos mediante la app móvil.

- **Modelo de diagrama de clases:**

  En el diagrama de clases se representa la estructura interna del sistema según la arquitectura propuesta. Las clases principales se dividen entre los elementos del vehículo (Raspberry, sensores y display) y los componentes externos como el Usuario y la Aplicación Móvil. La clase Raspberry aparece como el centro de la arquitectura, ya que administra la cámara (SensorQR), el acelerómetro y el display LCD, tal como ocurre en el funcionamiento real del sistema.

  La Aplicación Móvil se relaciona directamente con el Usuario y tiene métodos para mostrar alertas, ver el historial y enviar reporte a Carabineros. Esto refleja la parte lógica de la arquitectura donde la app sirve como puente entre el dueño y la Raspberry Pi.

  La clase Vehículo contiene los datos del auto y se asocia al Usuario, mientras que la clase Alerta representa la información que se genera cuando ocurre un movimiento de aceleración. La relación entre Alerta, Vehículo y Usuario sigue

el flujo real del sistema, la Raspberry detecta un movimiento en base al sensor del acelerómetro, genera una alerta y la app del usuario decide si confirmar o no.

- **Modelo de diagrama de secuencia:**

El diagrama de secuencia representa cómo la arquitectura funciona en los dos escenarios principales del sistema. Primero se muestra el inicio, donde la Raspberry Pi deja los sensores activos, el acelerómetro comienza a detectar movimiento y la cámara queda lista para capturar el código QR.

El primer caso corresponde al ingreso normal del dueño. El acelerómetro detecta la aceleración del vehículo, lo que activa un temporizador mientras el propietario muestra su código QR a la cámara. La Raspberry valida el QR y, si es correcto, la alerta se desactiva y el display LCD muestra el acceso autorizado. Aquí se observa la coordinación entre la aplicación móvil, la Raspberry Pi, el acelerómetro y la cámara.

El segundo caso representa el escenario de intrusión. El acelerómetro detecta el movimiento igual que antes, pero como no se presenta un QR válido, la Raspberry activa la cámara para capturar una imagen del sospechoso y envía una notificación al dueño mediante la aplicación móvil. Si el usuario confirma que se trata de un robo, la app envía los datos al sistema de la autoridad. Esto muestra cómo la arquitectura reacciona automáticamente usando los sensores, la Raspberry Pi y la app, enlazándose finalmente con un servicio externo.

5.1.7 Bot de Telegram(FunatronBot)

5.2 Herramientas y técnicas

5.2.1 Herramientas:

Durante el desarrollo del proyecto se utilizaron distintas herramientas tanto de software como de hardware:

- **Visual Studio Code:** editor principal para la programación y organización del código.
- **Redmine:** gestión de tareas y seguimiento del avance del proyecto.
- **Google Document:** Elaboración del documento y bitácora.
- **Draw.io:** creación de los diagramas (casos de uso, clases, secuencia, contexto).
- **Canva:** diseño de interfaz de usuario y elementos visuales del informe y presentación.
- **Sensores y módulos físicos:** acelerómetro, cámara, display LCD y demás componentes utilizados en la Raspberry.
- **Sistema Operativo Raspberry Pi OS:** plataforma donde se ejecutará la lógica del sistema.

5.2.1 Técnicas utilizadas:

- **Dividir para conquistar:** Se separaron los módulos del sistema para facilitar el desarrollo.
- **Iteración incremental:** Cada parte del proyecto se fue avanzando en etapas, revisando antes de pasar a la siguiente.
- **Validación por escenarios:** Se analizaron los dos flujos principales (dueño e intruso) para comprobar coherencia.
- **Prototipado temprano:** Se diseñaron los modelos antes de pasar a la implementación real.

- **Modularización:** Cada componente (sensores, cámara, app móvil, alertas) se trató como módulo independiente.

- **Pruebas por componente:** Cada módulo se considerará individualmente en la integración (acelerómetro, QR, LCD).

5.3 Plan de integración

Este plan describe cómo se unificarán los distintos componentes físicos y lógicos desarrollados durante el proyecto para formar un sistema plenamente operativo.

Integración de Sensores con Raspberry Pi 4:

- Conexión física del acelerómetro GY-6500/9250 y verificación de lectura estable.

- Configuración de librerías Python (smbus, grovepi, etc.).

- Ajuste de calibración inicial para evitar falsos positivos.

Integración de la Cámara QR:

- Configuración de la cámara integrada al Raspberry Pi 4.

- Implementación del lector QR mediante Python + OpenCV.

- Validación contra el QR dinámico generado por telegram

Integración del Display LCD

- Comunicación con Raspberry mediante I²C.
- Implementación de mensajes de estado según el flujo de casos existentes.

Integración del Módulo Acelerómetro:

- Registra continuamente los valores de aceleración del vehículo.
- Envía los datos a la Raspberry Pi a través del bus I²C para su procesamiento.
- Permite identificar cuando el vehículo inicia movimiento, activando el flujo de validación del usuario mediante QR.
- Incluye calibración inicial para reducir lecturas erróneas y obtener detecciones más precisas.

Integración del Bot de Telegram:

- Creación del bot mediante BotFather y configuración del token.
- Implementación del módulo de comunicación utilizando las librerías **telegram** y **telethon** para el envío y recepción de mensajes.
- Envío de comandos y mensajes desde la Raspberry al usuario, como alertas de movimiento, solicitudes de validación o reportes.
- Recepción de respuestas del usuario para confirmar acceso, solicitar QR o confirmar un posible robo.
- Envío del QR dinámico mediante el bot, solicitado directamente por el usuario.

5.4 Descripción de la Arquitectura (vista desde los módulos del caso de uso)

La arquitectura del sistema se organiza en torno a la interacción entre los módulos principales definidos en los casos de uso. Cada uno de estos módulos se integra con la Raspberry Pi 4 como unidad central, encargada de coordinar sensores,

captura de imágenes, visualización en pantalla y comunicación con el usuario mediante el bot de Telegram.

A continuación, se describe cómo cada módulo participa dentro de la arquitectura del sistema:

Módulo de Detección de Movimiento (Acelerómetro)

Este módulo está compuesto por el sensor GY-6500/9250 conectado a la Raspberry Pi mediante I²C.

Su función dentro de la arquitectura es:

- Detectar cambios significativos de aceleración.
- Enviar los valores registrados al sistema principal.
- Activar el flujo de validación cuando se detecta inicio de movimiento.

Este módulo corresponde directamente al caso de uso donde el sistema identifica un posible arranque no autorizado del vehículo.

Módulo de Validación del Usuario (Cámara y QR)

Una vez que el acelerómetro detecta movimiento, la cámara integrada se activa para capturar y procesar el QR que el usuario debe presentar.

Dentro de la arquitectura, este módulo:

- Captura imágenes en tiempo real.
- Decodifica el QR utilizando Python + OpenCV.
- Compara el QR escaneado con el QR dinámico generado por el bot de Telegram.

Este procedimiento corresponde al caso de uso de autenticación del dueño del vehículo.

Si la validación es correcta,el sistema autoriza el acceso.
Si no se presenta un QR válido,se activa el módulo de alerta.

Módulo de Display LCD

El display LCD RGB Backlight actúa como medio visual dentro del vehículo. Su rol es mostrar al usuario el estado actual del sistema según el caso de uso activo.

Mensajes clave:

- “Autorizado”
- “Denegado”

Este módulo apoya todos los casos de uso activos dentro del vehículo, entregando retroalimentación inmediata al usuario.

Módulo de Comunicación con el Propietario (Bot de Telegram)

Utiliza las librerías telegram y telethon para establecer comunicación bidireccional con la Raspberry Pi.

Participa en los casos de uso:

- Enviar notificación de movimiento no autorizado.
- Solicitar confirmación del usuario.
- Generar y entregar el QR dinámico.
- Recibir confirmación de robo o acceso legítimo.
- Enviar imagen del sospechoso si corresponde.

Su papel dentro de la arquitectura es fundamental, ya que funciona como la interfaz principal entre el dueño y el sistema.

Módulo de Generación y Gestión de Alertas

Este módulo se activa cuando:

- No se presenta un QR válido dentro del tiempo establecido, o
- El usuario indica desde Telegram que no reconoce el movimiento del vehículo.

Funciones dentro de la arquitectura:

- Activar alerta interna en Raspberry.
- Capturar imagen del sospechoso mediante la cámara.
- Enviar notificación al usuario.
- Preparar reporte con datos del vehículo, fecha, hora e imagen.

Módulo Central (Lógica Principal en Raspberry Pi)

La Raspberry Pi 4 actúa como núcleo de la arquitectura.
Coordina todos los módulos anteriores:

- Recibe datos del acelerómetro.
- Controla la cámara y valida QR.
- Actualiza el display LCD.
- Se comunica con Telegram.
- Ejecuta la lógica de estados del sistema.

Este módulo representa la ejecución completa del caso de uso global del sistema antirrobo.

5.5 Modelo de Implementación

El modelo de implementación define cómo se estructura el software dentro de la Raspberry Pi 4 y la manera en que cada módulo se ejecuta para cumplir los casos de uso del sistema. La solución se organiza en componentes independientes, implementados en Python, y coordinados mediante un programa principal.

Estructura de módulos:

El sistema se implementa en una arquitectura modular, donde cada componente cumple una función específica:

- Módulo de Acelerómetro: encargado de obtener las lecturas del sensor y notificar cambios significativos de movimiento.
- Módulo de Cámara y QR: realiza la captura de imágenes y la decodificación del código QR.
- Módulo Bot de Telegram: administra la comunicación con el usuario mediante las librerías *telegram* y *telethon*.
- Módulo Display LCD: muestra mensajes de estado dentro del vehículo.
- Módulo de Alertas: gestiona la activación de alertas, captura de imágenes y envío de notificaciones.
- Módulo Central: coordina el flujo del sistema y la interacción entre los demás módulos.

Organización del software:

Los módulos se implementan en archivos separados para facilitar su mantenimiento:

- → coordinación general del sistema
- `accelerometer.py` → manejo del sensor
- `qr_reader.py` → lectura de QR
- `telegram_bot.py` → comunicación con el bot
- `lcd_display.py` → controlador del LCD

- `alerts.py` → envío y registro de alertas

5.6 Módulos Implementados

Durante el desarrollo del sistema se implementaron los módulos principales que permiten la ejecución completa del flujo definido en los casos de uso. Cada módulo fue desarrollado de manera independiente para facilitar su integración y asegurar un funcionamiento estable dentro de la Raspberry Pi 4.

- **Módulo de Acelerómetro**

  Encargado de obtener las lecturas del sensor GY-6500/9250 y detectar cambios significativos de movimiento.
  Implementa la lógica necesaria para determinar cuándo el vehículo inicia desplazamiento, activando la fase de validación.

- **Módulo de Cámara y Lectura de QR**

  Controla la cámara integrada y utiliza OpenCV para capturar y decodificar el código QR presentado por el usuario.
  Permite confirmar si el usuario realizó una autenticación válida.

- **Módulo Bot de Telegram**

  Implementado utilizando las librerías **telegram** y **telethon**, permite la comunicación directa con el propietario del vehículo.
  Administra:

  - Envío de alertas.
  - Generación del QR dinámico.
  - Recepción de confirmaciones o reportes de robo.

- **Módulo Display LCD**

  Permite mostrar mensajes de estado en pantalla según el flujo del sistema, entregando retroalimentación inmediata dentro del vehículo.

- **Módulo de Alertas**

  Gestiona la activación de una alerta ante fallas de validación:

  - Toma de fotografía del sospechoso.
  - Envío de notificación al usuario.
  - Registro del evento para historial.

- **Módulo Central del Sistema**

  Coordina todos los módulos anteriores, administra los estados del sistema y ejecuta el flujo completo:
  detección, validación y autorización o alerta.

5.7 Reporte de Revisión

Durante el desarrollo del proyecto, se realizaron diversas pruebas evaluando partes específicas del sistema:

5.7.1 Prueba de Integración de Hardware y Software

Durante la primera prueba integral del dispositivo finalizado (Raspberry Pi), se validó el ciclo completo de operación visualizando la ejecución de funciones a través del monitor. La etapa de generación del código QR mediante el módulo "bot-funatron" operó correctamente y sin incidencias.

No obstante, durante la fase de validación de escaneo, se detectó un fallo crítico: el sistema no reconoció el periférico de captura de imagen (cámara), resultando en una ausencia de señal en el monitor. Tras el desmontaje de la carcasa para el diagnóstico, se identificó una conexión deficiente del módulo de la cámara. Se procedió a restablecer la conexión física y se reforzó la sujeción del componente para asegurar su estabilidad y prevenir futuras desconexiones ante vibraciones o movimiento.

5.7.2 Segunda Prueba: Validación de Sensores y Lógica de Estado

Una vez solventada la incidencia de hardware (cámara), se procedió a la segunda prueba integral del dispositivo Raspberry Pi. En esta instancia, el módulo de escaneo QR funcionó correctamente, permitiendo el paso a la validación del algoritmo de detección de estado del vehículo (encendido/apagado) mediante el acelerómetro.

La lógica diseñada establece que, ante la detección de movimiento, una variable de sensibilidad (inicializada en 0) debe incrementar su valor en una unidad cada 4 segundos de actividad continua. Sin embargo, durante las pruebas de estrés con movimiento constante, se observó que la variable no acumulaba valor, estancándose en 1 de forma estática, e impidiendo el reinicio a 0 tras el cese del movimiento.

Solución: Se había implementado erróneamente una asignación directa de valor positivo (`v=+1`), en lugar del operador de incremento compuesto (`v+=1`). Tras corregir la sintaxis, se repitió la prueba, verificando que la variable incrementaba y se restablecía correctamente según los intervalos de tiempo programados.

VII. Problemas encontrados y soluciones propuestas

7.1 Tamaño insuficiente de la carcasa Raspberry

❌ Problema Identificado

Se solicitó la impresión 3D de la carcasa para albergar el dispositivo Raspberry. Sin embargo, se detectó un error de medición en el dispositivo Raspberry, lo que resultó en una carcasa de tamaño insuficiente. Específicamente, el dispositivo no encajaba correctamente en ciertas esquinas debido a la falta de holgura dimensional.

✅ Solución Propuesta

La solución consiste en:

- **Ajuste manual de las esquinas:** Se utilizó un cúter (corta cartón) y un encendedor para remover manualmente y fundir ligeramente el material plástico de las secciones de la carcasa que interferían con el correcto asiento del dispositivo.

- **Resultado:** Esta intervención permitió acomodar el dispositivo Raspberry dentro de la carcasa de manera funcional, aunque la solución es considerada temporal.

7.2 Aplicación innecesaria

❌ Problema Identificado

El desarrollo de la aplicación móvil se consideró innecesario debido a la simpleza y complejidad básica de las funciones requeridas (como la generación de códigos QR y la emisión de reportes).

✅ Solución Propuesta y Justificación

Para optimizar el proceso, el grupo, siguiendo la recomendación del profesor, decidió cambiar la estrategia tecnológica a un bot de Telegram.

Esta solución permite:

- Ejecutar todas las funciones necesarias (generación de QR, reportes, etc.) de manera más directa y sencilla.
- Reducir significativamente la complejidad de desarrollo, así como el tiempo de implementación.
- Ofrecer una experiencia de usuario fluida a través de una plataforma de mensajería ya establecida, eliminando la necesidad de que los usuarios instalen una aplicación adicional.

VIII. Trabajo Futuro

Hacia el futuro, el proyecto FUNATRON busca trascender su configuración actual de sensores periféricos para lograr una integración profunda con la tecnología propia del vehículo. Esto implica evolucionar el uso del acelerómetro y la cámara de lectura QR hacia una conexión directa con el bus CAN del automóvil, permitiendo que la lógica central ejecutada en Python pueda no solo detectar movimiento, sino también interactuar con el sistema de inmovilización electrónica y el cierre centralizado del vehículo. Asimismo, se contempla optimizar la gestión energética para que el sistema se alimente de forma permanente de la batería del auto, asegurando que la detección de movimiento mediante el sensor acelerómetro sea una función de seguridad siempre activa.

En paralelo, una línea crítica de desarrollo será la normalización del sistema ante las autoridades pertinentes para que los reportes generados tras una confirmación de robo sean aceptados como evidencia oficial. El siguiente paso técnico es estandarizar estos protocolos de comunicación para que se integren directamente con las plataformas de recepción de denuncias de instituciones como Carabineros de Chile. Al haber migrado la

interfaz de usuario hacia un bot de Telegram, FUNATRON posee una arquitectura flexible que facilita la creación de canales de comunicación bidireccional no solo con el propietario, sino con centrales de monitoreo de seguridad ciudadana, permitiendo una respuesta inmediata y coordinada ante eventos de conducción no autorizada.

IX. Conclusiones

Esta primera fase se ha completado exitosamente con la definición integral de la propuesta del proyecto. Se estableció la problemática central y se diseñó la solución tecnológica, que consiste en un sistema de detección inteligente para anticipar y prevenir el robo vehicular. Paralelamente, se consolidaron los cimientos de la gestión del proyecto, lo que incluyó la asignación formal de roles (Jefe de Proyecto, Programadores, Diseñador, Ensamblador y Documentador) , la estimación de costos y tiempos , y la determinación de los objetivos. La conclusión de esta etapa garantiza que el proyecto cuenta con la estructura organizativa y el marco técnico necesarios para abordar las siguientes fases de ejecución y desarrollo.

La segunda fase se centró exitosamente en la validación técnica y la estructuración del diseño del sistema. Se llevó a cabo una exhaustiva investigación sobre la compatibilidad y ejecución de los componentes centrales, incluyendo la Raspberry Pi 4, los sensores (cámara y acelerómetro), y los lenguajes de programación requeridos (Python, Kotlin, C#). Esto permitió asegurar la base tecnológica necesaria para la implementación. Además, se definieron los modelos de diseño esenciales para comunicar la solución, como el diagrama de casos de uso, el diagrama de clases y los diagramas de secuencia, complementados por el boceto de la Interfaz Gráfica de Usuario (IGU) de la aplicación móvil, elemento clave para la recepción de notificaciones de alerta. Finalmente, la creación formal de los requerimientos funcionales y no funcionales del proyecto garantiza una guía precisa para el desarrollo del software y la verificación de la calidad del producto final.

La finalización de este proyecto marca la entrega de un sistema integralmente operativo y funcional, capaz de cumplir con los requerimientos establecidos al inicio del desarrollo. A lo largo del proceso, nos enfrentamos a desafíos técnicos de considerable complejidad, particularmente durante la etapa de integración de los sensores y la sincronización con la unidad de procesamiento (Raspberry Pi). La fase de depuración exigió un análisis exhaustivo para garantizar la estabilidad del sistema en un entorno real.

Sin embargo, la superación de estos obstáculos no solo validó la robustez de la solución propuesta, sino que transformó el desarrollo en una experiencia de aprendizaje técnico y metodológico sumamente enriquecedora. El resultado final no es solo un prototipo funcional, sino una arquitectura sólida y escalable que sienta bases firmes para futuras investigaciones, permitiendo la incorporación de nuevas tecnologías o funcionalidades en iteraciones posteriores."

X. Referencias

Referencias de Hardware y Componentes

- Raspberry Pi 4

  https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

- Página donde se compró el sensor acelerómetro

  https://www.mechatronicstore.cl/sensor-acelerometro-y-giroscopio-gy-6500-mpu-6500/

- *Grove - LCD RGB*

  https://wiki.seeedstudio.com/es/Grove-LCD_RGB_Backlight/

- Sensores con Raspberry PI 4

  https://www.circuitbasics.com/what-is-an-accelerometer/

Referencias de Software y Desarrollo

- Meta Quest 3

  https://www.meta.com/es-es/help/quest/509273027107091/

- Software del Raspberry PI 4

  https://www.raspberrypi.com/documentation/computers/getting-started.html

- Unity

  https://docs.unity3d.com/es/530/Manual/UnityManual.html