udp UNIVERSIDAD DIEGO PORTALES

# Informe 2 - Proyecto TICS: **Cercasco**

Carcasa Inteligente Desmontable para Casco de Ciclista

Matías Vigneau Luis Valdenegro Ezequiel Morales Gabriel González

23 de Octubre de 2025

**Resumen**

Este informe actualiza el trabajo del *Informe 1*, presenta el **Modelo de Negocios**, la **identificación de cliente y hallazgos de entrevistas**, y profundiza en **requerimientos**, **decisiones tecnológicas** y **diseño de circuito**. Además, incluye una **matriz de trazabilidad** y un **checklist de cumplimiento de rúbrica**.

# Índice

**1. Actualización de Items del Informe Fase 1** **3**

**2. Modelo de Negocios** **4**

**3. Identificación del Cliente y Conclusiones de Entrevistas** **4**

**4. Requerimientos Funcionales y No Funcionales** **5**
4.1. Requerimientos Funcionales (RF) . . . . . 5
4.2. Requerimientos No Funcionales (RNF) . . . . . 5

**5. Especificación y Justificación de Decisiones Tecnológicas** **6**
5.1. Arquitectura General: Hub & Spoke . . . . . 6
5.2. Plataformas de Desarrollo y Lenguajes . . . . . 6
5.2.1. Hub de Alertas y Sensor Ultrasónico (Framework Arduino) . . . . . 6
5.2.2. Sensor de Visión (Framework ESP-IDF y TFLite) . . . . . 7
5.3. Protocolos de Comunicación . . . . . 7

**6. Diseño del Circuito y Justificación de Operación** **8**
6.1. Diseño del Pod de Sensores (Trasero) . . . . . 8
6.1.1. Justificación de Operación (Pod Trasero) . . . . . 8
6.2. Diseño del Hub de Alertas (Casco) . . . . . 9
6.2.1. Justificación de Operación (Hub Casco) . . . . . 9

**7. Lógica de Evaluación de Riesgo y Alertas (síntesis)** **9**

**8. Matriz de Trazabilidad** **10**

**9. Referencias** **11**

# 1. Actualización de Items del Informe Fase 1

Respecto al **Informe 1**, se realizaron los siguientes cambios y mejoras:

**A1**. **Arquitectura:** se formaliza un esquema *hub & spoke*: *Pod de Sensores* (trasero) y *Hub de Alertas* (en el casco). El **Hub** concentra la lógica y las salidas (vibración y LEDs).

**A2**. **Comunicación interna:** adopción de **ESP-NOW** para el enlace Pod–Hub con baja latencia y sin asociarse a un AP.

**A3**. **Comunicación externa:** el Hub expone **BLE** para configuración y consulta de historial (app móvil).

**A4**. **Percepción:** combinación de **ultrasonido** (distancia) con **detección binaria de vehículo** mediante cámara (ESP32-CAM) optimizada a nivel de firmware. Se mantiene la opción de **ToF** o **radar** como reemplazo equivalente si la validación de campo lo amerita.

**A5**. **Alertas:** se definen *niveles de alerta* (normal, preventiva, inminente) y *patrones* para vibración/LEDs.

**A6**. **Energía:** todos los módulos se alimentan con 5 V desde *powerbank*; se especifican *condensadores de bulk* y *protección* en líneas críticas.

**A7**. **Trazabilidad:** se agrega la matriz RF/RNF → decisiones → circuito → pruebas (Sección 8).

**Resumen de impacto** Los cambios priorizan *latencia*, *modularidad* y *confiabilidad* para pruebas de campo.

*Documentos previos de referencia: Informe 1.*

# 2. Modelo de Negocios

**Propuesta de valor:** aumentar la *percepción trasera* del ciclista mediante alertas hápticas y visuales en una carcasa desmontable, sin intervenir el casco.

<table>
<tr><th colspan="2">Socios clave</th><th>Actividades clave</th></tr>
<tr><td>Talleres de impresión 3D; tiendas de electrónica; ciclovías y clubes de ciclismo; voluntarios para pruebas</td><td>Proveedores de módulos ESP32, sensores y tiras LED; apoyo de laboratorio</td><td>Diseño CAD, integración HW/SW, pruebas de usuario, soporte post-venta (básico)</td></tr>
<tr><th>Recursos clave</th><th colspan="2">Relación con clientes</th></tr>
<tr><td>Pod y Hub modulares; firmware actualizable; app móvil; documentación</td><td colspan="2">Comunidad y feedback; tutoriales; garantías básicas de prototipo</td></tr>
<tr><th>Segmentos de clientes</th><th>Canales</th><th>Estructura de costos</th></tr>
<tr><td>Ciclistas urbanos; repartidores; aficionados MTB/Gravel</td><td>Venta directa en campus y ferias; difusión digital</td><td>Electrónica, carcasa, ensamblaje, validación</td></tr>
<tr><th colspan="2">Flujo de ingresos</th><th>Métricas clave</th></tr>
<tr><td colspan="2">Venta del kit; servicio de instalación opcional; accesorios</td><td>Unidades, tasa de fallas, NPS, costo por unidad</td></tr>
</table>

# 3. Identificación del Cliente y Conclusiones de Entrevistas

**Cliente objetivo principal:** ciclista urbano que circula en vías mixtas y desea aumentar su percepción trasera sin saturación auditiva.

**Entrevistas (síntesis)** 6 entrevistas semiestructuradas con alumnos repartidores y ciclistas frecuentes:

- Demanda de **alertas claras** sin ruidos; preferencia por **vibración** direccional.
- **No** desean modificaciones permanentes al casco; el accesorio debe ser *externo, extraíble*.
- Alta valoración de **autonomía** y **bajo peso**.
- Aceptación de una app solo para *configurar* y *ver eventos*, no para uso durante el pedaleo.

**Conclusiones accionables**

**C1**. Mantener carcasa *desmontable* y *universal*.
**C2**. Patrones de vibración simples y escalados por urgencia.
**C3**. App minimalista orientada a configuración e histórico.

# 4. Requerimientos Funcionales y No Funcionales

Esta sección define el ”contrato” del sistema: qué debe hacer y bajo qué condiciones de calidad y limitación debe operar.

## 4.1. Requerimientos Funcionales (RF)

Definen las acciones y funciones específicas que el sistema debe ser capaz de ejecutar.

- **RF-01:** El Pod de Sensores **debe** detectar la presencia de vehículos en la parte trasera del ciclista, incluso en condiciones de baja luminosidad (visión nocturna).
- **RF-02:** El Pod de Sensores **debe** medir continuamente la distancia a los objetos traseros en un rango efectivo de hasta 6 metros.
- **RF-03:** El Pod de Sensores **debe** enviar los datos de detección de vehículo ("SI/"NO") y de distancia (en metros) al Hub del Casco.
- **RF-04:** El Hub del Casco **debe** recibir los datos de los sensores y procesar la lógica de alerta.
- **RF-05:** El Hub del Casco **debe** activar el motor de vibración con intensidad variable (suave/fuerte) según la lógica de alerta (combinación de detección y proximidad).
- **RF-06:** El Hub del Casco **debe** activar las tiras LED RGB con patrones de parpadeo y color rojo durante una alerta activa.
- **RF-07:** El Hub del Casco **debe** permitir la conexión de una aplicación móvil de usuario.
- **RF-08:** La aplicación móvil **debe** permitir al usuario configurar los patrones de luz y colores de los LEDs en modo ”normal” (no-alerta).
- **RF-09:** La aplicación móvil **debe** permitir al usuario visualizar un historial de alertas registradas por el Hub.

## 4.2. Requerimientos No Funcionales (RNF)

Definen las características, calidades, restricciones y limitaciones del sistema.

- **RNF-01 (Latencia):** La latencia total del sistema (desde que el sensor detecta hasta que el Hub vibra) **debe** ser inferior a 500 milisegundos.
- **RNF-02 (Comunicación Interna):** La comunicación entre el Pod de Sensores y el Hub del Casco **debe** realizarse exclusivamente mediante el protocolo ESP-NOW.
- **RNF-03 (Comunicación Externa):** La comunicación entre el Hub del Casco y la aplicación móvil **debe** realizarse exclusivamente mediante Bluetooth Low Energy (BLE).
- **RNF-04 (Alimentación):** Todos los módulos del sistema **deben** ser alimentados por una fuente de poder externa de 5V. Para la fase de prototipado, se utilizará una powerbank de 5400mAh con conexión micro-USB.
- **RNF-05 (Robustez):** Los componentes expuestos a la intemperie, específicamente el sensor de ultrasonido y las tiras LED, **deben** ser resistentes al agua (certificación IP67/IP68).
- **RNF-06 (Campo de Visión):** El sistema **debe** ofrecer un amplio ángulo de detección trasero, definido por la lente de la cámara de 160 grados (FoV).
- **RNF-07 (Modularidad):** El código de lógica de decisión **debe** residir únicamente en el Hub del Casco, permitiendo que los sensores operen como unidades de reporte de información constante y sin interrupciones.

- **RNF-08 (Fiabilidad):** El sistema **debe** operar sin ‘delay()‘ en el bucle principal del Hub para garantizar la recepción continua de alertas y comandos.

# 5. Especificación y Justificación de Decisiones Tecnológicas

Para el desarrollo del proyecto ”Cercasco”, se ha optado por una arquitectura de sistema distribuido, separando las tareas de sensado y de alerta en dos módulos físicos independientes. Esta decisión busca optimizar la carga de procesamiento, la eficiencia energética y la modularidad del sistema.

## 5.1. Arquitectura General: Hub & Spoke

El sistema se divide en dos componentes principales:

- **Pod de Sensores (Trasero):** Ubicado en la bicicleta, contiene los sensores de percepción (ESP32-CAM y ESP32 con Ultrasonido). Su única tarea es ”sentir” el entorno y ”reportar” eventos (RF-01, RF-02, RF-03).
- **Hub de Alertas (Casco):** Ubicado en el casco, contiene el ESP32 principal, las tiras LED y el motor de vibración. Su tarea es ”recibir” reportes, ”decidir” el nivel de alerta y ”actuar” (vibrar/iluminar), además de comunicarse con la App móvil (RF-04 a RF-09).

Esta arquitectura *Hub and Spoke* es fundamental, ya que libera al ESP32-CAM de la carga de gestionar alertas o Bluetooth, permitiéndole dedicar el 100 % de sus recursos a la compleja tarea de detección de vehículos (RNF-07).

## 5.2. Plataformas de Desarrollo y Lenguajes

La selección de plataformas se basa en el rendimiento y las librerías disponibles para cada tarea específica.

### 5.2.1. Hub de Alertas y Sensor Ultrasónico (Framework Arduino)

- **Plataforma:** ESP32-WROOM-32. Se utiliza esta plataforma tanto para el Hub como para el control del sensor ultrasónico.
- **Lenguaje:** C++ (utilizando el framework de Arduino).
- **Justificación:** Se elige C++/Arduino por su alto rendimiento y acceso directo al hardware. Es crítico para el Hub, que debe gestionar tres tareas concurrentes con baja latencia (RNF-01): recibir datos por ESP-NOW, controlar PWM de LEDs/motor (RNF-08) y gestionar una conexión BLE.
- **Componente Sensor Clave:** Se selecciona el sensor ultrasónico **AJ-SR04M** por sus características superiores: es impermeable (cumpliendo RNF-05) y ofrece un rango de detección de hasta 6 metros (cumpliendo RF-02), superando ampliamente a los sensores HC-SR04 estándar.
- **Componentes de Alerta Clave:** Se utilizan tiras LED RGB direccionables (WS2812B) con certificación **IP68** (RNF-05) y un módulo de motor de vibración (RF-05).
- **Librerías Clave (Hub):**

- `esp_now.h`: Para la comunicación inalámbrica (RNF-02).
- `FastLED.h`: Para el control avanzado de las tiras LED IP68 (RF-06, RF-08).
- `BLEDevice.h`: Para implementar el servidor BLE (RNF-03).
- **LEDC (API nativa):** Se usará la API 'ledcWrite()' para el control PWM del motor de vibración (RF-05).

#### 5.2.2. Sensor de Visión (Framework ESP-IDF y TFLite)

- **Plataforma:** ESP32-CAM.
- **Lenguaje:** C++ (utilizando el framework ESP-IDF).
- **Justificación:** La detección de vehículos (RF-01) es la tarea más demandante. Se opta por C++ sobre el ESP-IDF (Espressif IoT Development Framework) ya que ofrece el máximo rendimiento posible para tareas de IA.
- **Especificación de Cámara:** Se utiliza un modelo de ESP32-CAM equipado con una lente de **160 grados** y capacidad de **visión nocturna**. Esto amplía drásticamente el campo de visión del sistema (cumpliendo RNF-06) y asegura su funcionamiento en condiciones de baja luminosidad (RF-01).
- **Librerías y Modelos Clave:**
  - `esp_camera.h`: Librería oficial para la captura de cuadros de la cámara.
  - **TensorFlow Lite for Microcontrollers (TFLite):** Permite ejecutar modelos de *Machine Learning* optimizados.
  - **Modelo de Detección:** Se usará un modelo de detección de objetos (como *MobileNetV2* o *YOLOv8-Nano*), cuantizado para operar eficientemente en el ESP32.

### 5.3. Protocolos de Comunicación

- **Comunicación Interna (Sensor-Hub): ESP-NOW. Justificación:** Se elige ESP-NOW para cumplir con el RNF-02. Es un protocolo de Espressif que usa Wi-Fi (2.4GHz) pero sin la sobrecarga de TCP/IP. Ofrece comunicación directa, es extremadamente rápido (cumpliendo RNF-01) y, crucialmente, **deja el módulo Bluetooth del Hub completamente libre** para la conexión con la app (RNF-03).
- **Comunicación Externa (Hub-App): Bluetooth Low Energy (BLE). Justificación:** Se utiliza BLE para cumplir con el RNF-03. Es el estándar universal para accesorios de bajo consumo en *smartphones*, permitiendo cumplir con los requerimientos de la aplicación (RF-07, RF-08, RF-09).

**Nota:** Aunque algunos esquemas simplificados del Pod muestran un enlace cableado entre placas, en la implementación real la comunicación de datos es exclusivamente por ESP-NOW (RNF-02).

# 6. Diseño del Circuito y Justificación de Operación

La arquitectura electrónica se divide en los dos módulos físicos mencionados, alimentados por una powerbank de 5400mAh (RNF-04).

## 6.1. Diseño del Pod de Sensores (Trasero)

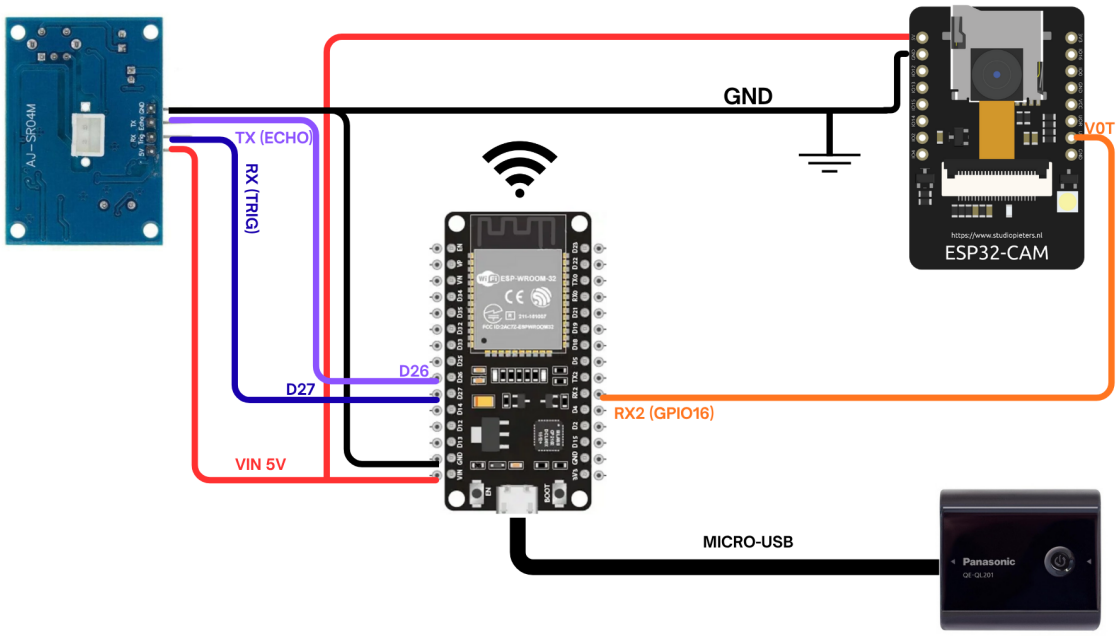


Figura 1: Pod de Sensores: ESP32-CAM (detección binaria) + ESP32 (ultrasonido). La conexión de datos entre placas es inalámbrica (ESP-NOW).

### 6.1.1. Justificación de Operación (Pod Trasero)

- **Alimentación:** Ambos ESP32 se alimentan de la fuente común de 5V (RNF-04).
- **ESP32-CAM:** Funciona de forma autónoma. Solo requiere alimentación (5V y GND). Su única "salida" es la antena integrada, a través de la cual envía reportes (`{"vehiculo":"SI"}`) vía ESP-NOW (RNF-02) donde el cable naranja a RX2 (GPIO16) es una representación visual de envío de información del ESP32-CAM más no una implementación del circuito final.
- **ESP32-Sensor (Ultrasonido):** Un ESP32-WROOM-32 estándar se conecta al sensor **AJ-SR04M**.
  - **Justificación del Sensor:** Impermeable (RNF-05) y rango de 6 m (RF-02). Se alimenta a 5V.
  - `Trig` → **GPIO 26** (D26)
  - `Echo` → **GPIO 27** (D27).

## 6.2. Diseño del Hub de Alertas (Casco)

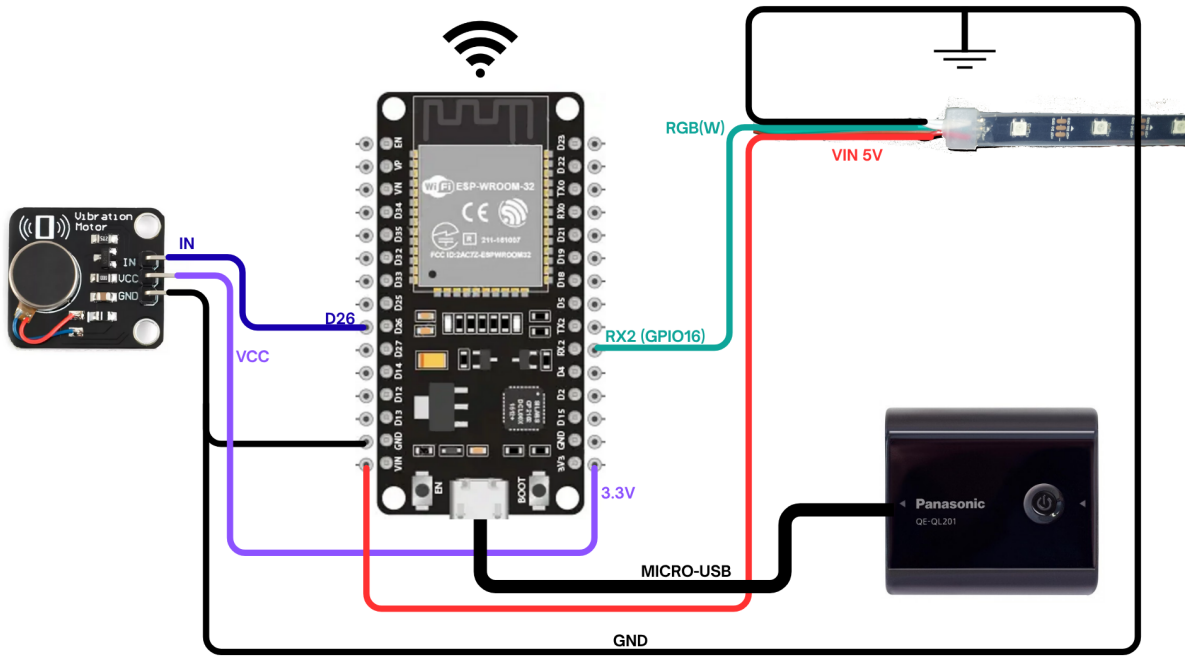


Figura 2: Hub de Alertas: ESP32 + driver de motor + tira LED direccionable + BLE.

### 6.2.1. Justificación de Operación (Hub Casco)

- **Alerta Háptica (Motor Vibracoin):**
  - Módulo controlador de motor.
  - `VCC` del módulo → **3.3V** del ESP32 (motor 3V).
  - `GND` del módulo → `GND` del ESP32.
  - `IN` del módulo → **GPIO 26** (PWM por LEDC, `ledcWrite()`).
- **Alerta Visual (Tiras LED WS2812B):**
  - **5V** del VIN placa.
  - **GND** común.
  - Datos → **GPIO 16** (RX2). Recomendación futura de implementar un LOGIC LEVEL SHIFTER para tolerancia a cambios de voltaje (Aunque actualmente no es necesario).
  - Para robustez final, usar **conversor de nivel** 3,3→5V en la línea de datos.

# 7. Lógica de Evaluación de Riesgo y Alertas (síntesis)

**L1**. **Fusión de evidencias**: presencia binaria (cámara) + distancia (ultrasonido).
**L2**. **Suavizado**: media móvil y persistencia en $N$ muestras para reducir falsos positivos.
**L3**. **Escalamiento**: umbrales por distancia/tiempo y velocidad relativa estimada.
**L4**. **Salida**: mapeo a patrón de vibración y LEDs; temporizador de *cooldown*.

# 8. Matriz de Trazabilidad

| Req. | Decisión | Circuito/Sección | Prueba / Métrica |
|---|---|---|---|
| **RF-1-2** | ESP-NOW; | Fig. 1–2 | Tasa de envío $\geq$10 Hz |
| **RF-3-4** | Lógica en Hub | Sec. 7 | Latencia $<$500 ms (**RNF-1**) |
| **RF-4** | PWM motor + patrones LED | Sec. 6 | Patrón correcto según nivel |
| **RF-5-6** | Servicio BLE (config/historial) | Sec. 5 | Conexión estable, lectura eventos |
| **RNF-2** | ESP-NOW | Sec. 5 | Pérdida $<$1 % paquetes en banco |
| **RNF-5** | Cono $\sim$75° | Sec. 6 | Validación geométrica/campo |

# 9. Referencias


- Documentación oficial de Espressif sobre **ESP-NOW API**. `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html`
- Documentación de **TensorFlow Lite for Microcontrollers**. `https://www.tensorflow.org/lite/microcontrollers`
- Librería **FastLED** (para control de LEDs WS2812B). `http://fastled.io/`
- Documentación de Espressif sobre **ESP32-CAM**. `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/camera.html`
- Documentación de **BLE en ESP32** (Arduino Core). `https://github.com/nkolban/ESP32_BLE_Arduino`
- Hoja de datos (Datasheet) del sensor ultrasónico **AJ-SR04M**.
- (Video) DroneBot Workshop. *Waterproof Ultrasonic Distance Sensors - JSN-SR04T & A02YYUW*. `https://www.youtube.com/watch?v=h6321UBATps`
- (Video) Mundo Yakara. *Crear tu propio sistema LED RGB DIRECCIONABLE, CONTROLADO POR wi-fi*. `https://www.youtube.com/watch?v=2O2TQYZ1MQA`
- (Video) Tech StudyCell. *IoT Based Water Level Monitoring system using ESP32 Blynk & Ultrasonic Sensor*. `https://www.youtube.com/watch?v=9geREeE13jc`