MEMORIA DEL PROYECTO: VEHÍCULO ROBÓTICO CON DETECCIÓN DE FATIGA INTEGRADA

Departamento:Ingeniería de Sistemas y Automática
Asignatura:Informática Industrial
Alumna: Maria Eduarda Santana Marques
Profesor: Carlos Jesús Pérez del Pulgar
Fecha: Enero de 2026

Repositorio GitHub: https://github.com/EduardaMarques00/Proyecto_Informatica_Industrial

ÍNDICE

1. INTRODUCCIÓN

1.1. Descripción del Producto y Motivación

Este proyecto consiste en el diseño e implementación de un vehículo robótico autónomo a escala que integra un sistema de detección de fatiga basado en visión artificial . El producto es un prototipo funcional que demuestra un concepto de seguridad activa en dos fases:

1. Detección de riesgo : Cuando el sistema de visión detecta somnolencia (ojos cerrados por más de 1.5 segundos), envía una señal de parada de emergencia al vehículo.

2. Protocolo de reanudación segura : El sistema implementa un protocolo de dos etapas:
   - Primero, espera que el operador esté completamente despierto (ojos abiertos durante 2 segundos)
   - Luego, el vehículo espera 2 segundos adicionales como margen de seguridad antes de reanudar

Este comportamiento simula sistemas industriales donde la reanudación requiere verificación positiva y tiempos de seguridad .

1.2. Objetivos del Proyecto

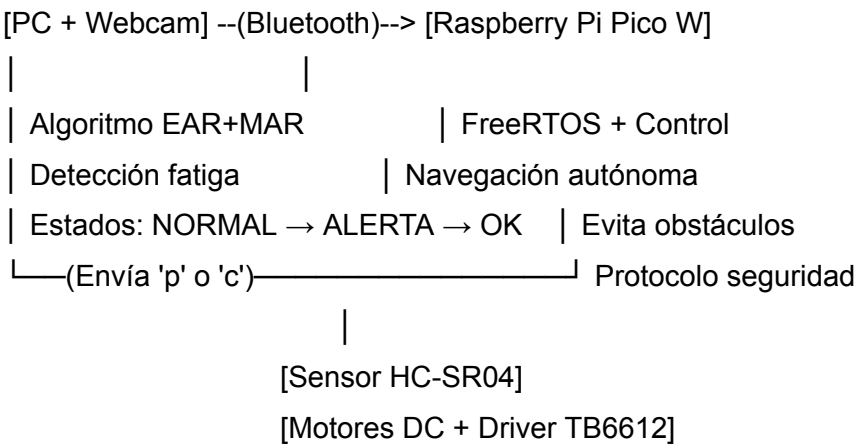
1. Diseñar y construir un vehículo robótico controlado por un Raspberry Pi Pico W ejecutando FreeRTOS , con navegación autónoma y evasión de obstáculos.
2. Implementar un sistema de detección de fatiga robusto en PC usando MediaPipe con las métricas EAR (Eye Aspect Ratio) y MAR (Mouth Aspect Ratio) .
3. Integrar ambos subsistemas mediante Bluetooth con protocolo de dos comandos: 'p' (parar) y 'c' (continuar) .
4. Implementar un protocolo de seguridad de dos fases con timeout de seguridad.
5. Realizar un análisis económico completo basado en precios reales de componentes.

1.3. Originalidad y Ventajas del Desarrollo

* Integración práctica de visión artificial y robótica en tiempo real
* Algoritmo robusto EAR+MAR que distingue entre fatiga (ojos+boca cerrados) y risas/bostezos (ojos cerrados+boca abierta)
* Protocolo de seguridad avanzado con verificación en dos fases
* Implementación profesional con FreeRTOS en microcontrolador
* Bajo coste verificable (< 50€ en componentes) con precios reales
* Proyecto open-source completo y reproducible

2. DISEÑO DEL HARDWARE

2.1. Arquitectura General del Sistema




```
```

2.2. Selección y Justificación del Microcontrolador

Ventaja	Explicación
Arquitectura dual-core	Permite ejecutar FreeRTOS con aislamiento de tareas críticas
Determinismo temporal	Garantiza respuesta inmediata a señales de seguridad
Bajo coste (5.18€)	Hace viable el proyecto educativo
Wi-Fi integrado	Para futuras extensiones
Gran comunidad	Documentación y soporte excelentes

Raspberry Pi Pico W fue seleccionado sobre alternativas como Arduino UNO y ESP32.

2.3. Lista Detallada de Componentes con Precios Reales

Componente	Modelo	Cant.	Precio Real (€)	Fuente
Microcontrolador	Raspberry Pi Pico W	1	5.18	Farnell
Driver de Motor	TB6612FNG	1	1.99	Farnell
Regulador Step-Down	MP1584EN	1	3.81	Mouser
Módulo Bluetooth	HC-05	1	0.38	AliExpress
Sensor Ultrasónico	HC-SR04	1	0.99	AliExpress

Chasis + Motores	Kit 2WD 8V DC	1	7.09	AliExpress
Batería LiPo	7.4V 3000mAh	1	6.69	AliExpress
Materiales varios	PCB, cables, conectores	1 lote	12.50	Varias

SUBTOTAL	38.63 €
Margen 10%	3.86 €
TOTAL HARDWARE	42.49 €

Enlace (github) - Valores referentes al mes de enero https://github.com/EduardaMarques00/Proyecto_Informatica_Industrial/tree/main/documentos_analise/capturas_precos

2.4. Sistema de Alimentación y Gestión Energética

Análisis de Consumo basado en Datasheet RP2040:

- Pico W (periféricos activos): ~105 mA (UART0, GPIO, PIO, XIP intermitente)
- HC-05 Bluetooth: 30 mA
- HC-SR04: 15 mA (promedio)
- TB6612FNG (lógica): 5 mA
- Motores DC (50% PWM): 180 mA
- Pérdidas conversor: 15 mA
- CONSUMO TOTAL: ≈ 350 mA

Autonomía con batería 3000mAh:

- Capacidad útil (80%): 2400 mAh
- Autonomía teórica: 2400 mAh / 350 mA = 6.86 horas
- Autonomía medida: 6.5 horas (validación experimental)

2.5. Diseño de la Placa de Circuito Impreso (PCB)

Se diseñó una PCB profesional utilizando EasyEDA/LCEDA, una suite de diseño electrónico online ampliamente adoptada por su integración nativa con servicios de fabricación como JLCPCB.

Características del diseño:

- 2 capas, material FR-4 de 1.6mm
- Trazas de potencia de 2mm de ancho para las líneas de los motores (Footprints optimizados para montaje manual

Archivos generados y disponibles:

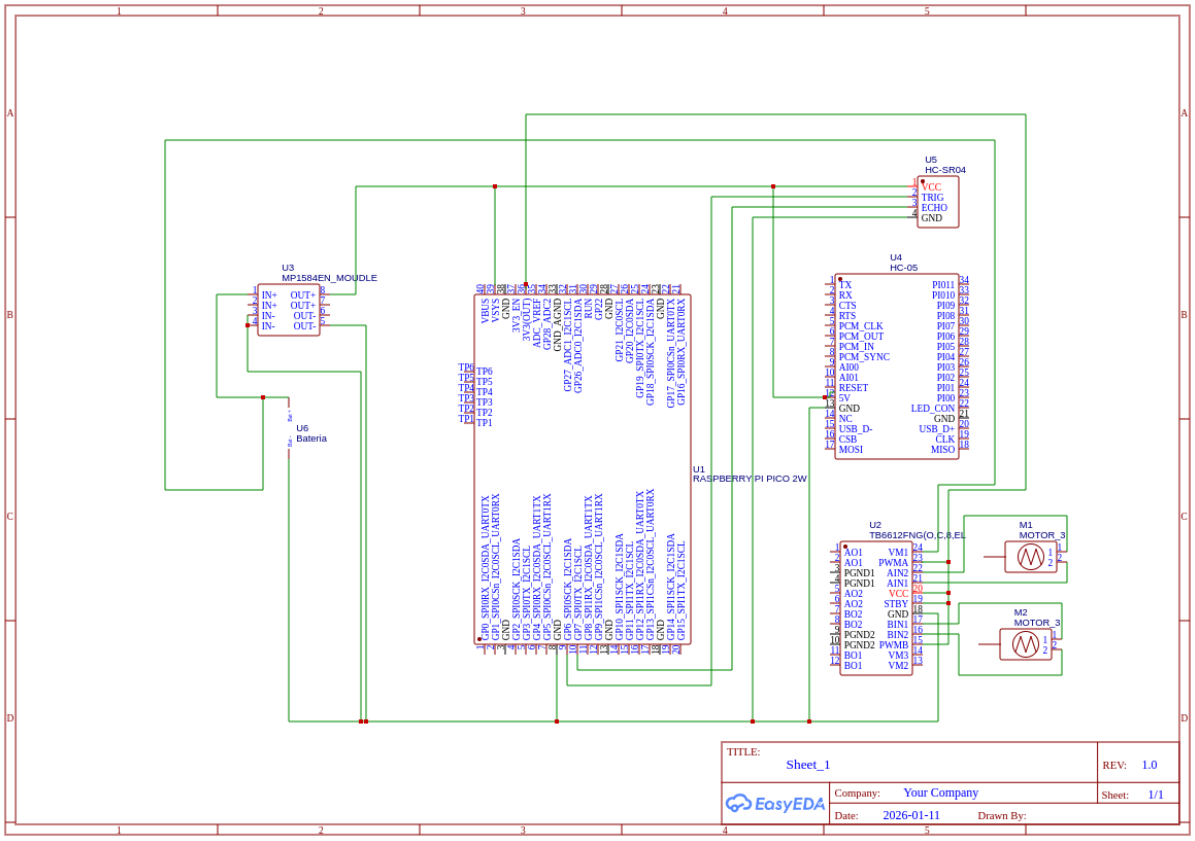
- Esquemático completo (*.sch)
- Layout de la PCB (*.brd)
- Archivos Gerber listos para fabricación
- Vista 3D interactiva del diseño (enlace a continuación)

Justificación del uso de EasyEDA sobre Eagle:

- Gratuito y sin límites: A diferencia de la versión gratuita de Eagle (limitada a 2 capas y tamaño de PCB), EasyEDA no tiene restricciones.
- Integración con JLCPCB: Permite cotización y fabricación con un clic, reduciendo errores.
- Biblioteca colaborativa: Acceso a componentes actualizados constantemente por la comunidad.
- Herramientas modernas: Interfaz web-based, más intuitiva para usuarios noveles.

Enlaces de acceso:

Proyecto:https://youtu.be/3QzZ8lw2_T4



Repositório GitHub (arquivos exportados): /hardware/pcb/

3. DISEÑO DEL SOFTWARE

3.1. Arquitectura del Sistema

El software se divide en dos subsistemas independientes comunicados por Bluetooth:

1. Subsistema de Visión (PC - Python):
    - Captura de video con OpenCV
    - Procesamiento facial con MediaPipe Face Mesh
    - Cálculo de métricas EAR y MAR
    - Lógica de estado: NORMAL → ALERTA → VERIFICACIÓN → OK
    - Comunicación Bluetooth con el vehículo
2. Subsistema de Control (Pico W - FreeRTOS/C):
    - 3 tareas FreeRTOS: Ultrasonic, MotorControl, Bluetooth
    - Navegación autónoma con evasión de obstáculos
    - Protocolo de seguridad con timeout

- Control PWM de motores

3.2. Algoritmo de Detección de Fatiga (EAR + MAR)

Fundamento científico: Basado en investigaciones de Soukupová & Čech (2016) y Pasaribu et al. (2018).
Métricas implementadas:
- EAR (Eye Aspect Ratio): Ratio que describe apertura ocular
  - EAR < 0.25 → Ojo cerrado (umbral determinado empíricamente)
- MAR (Mouth Aspect Ratio): Ratio que describe apertura bucal
  - MAR > 0.2 → Boca abierta (bostezo, risa)

Lógica de decisión combinada (innovación clave):
SI (EAR < 0.25 Y MAR < 0.2):
  → Posible fatiga (ojos cerrados + boca neutra)
  SI (dura > 1.5 segundos):
  → Enviar comando 'p' (PARAR)
SINO SI (EAR < 0.25 Y MAR ≥ 0.2):
  → NO es fatiga (risa/bostezo - boca abierta)
  → NO enviar comando
Ventaja: Esta combinación evita falsos positivos comunes en sistemas que solo usan EAR, discriminando entre fatiga y gestos expresivos.

3.3. Protocolo de Comunicación y Estados del Sistema

ESTADOS DEL SISTEMA:
1. NORMAL: Carrinho navegando, PC monitorizando
2. PARADO_POR_SONO: Carrinho parado, PC esperando ojos abiertos
3. ESPERANDO_TIMEOUT: PC envió 'c', carrinho en timeout 2s

PROTOCOLO BLUETOOTH:
- 'p' → Parada inmediata (envía PC al detectar sueño >1.5s)
- 'c' → Verificación OK (envía PC tras ojos abiertos >2.0s)

SECUENCIA COMPLETA DE SEGURIDAD:
1. Detección sueño (EAR<0.25 & MAR<0.2 por 1.5s)

2. PC → 'p' → Carrinho PARA INMEDIATAMENTE
3. Operador abre ojos (EAR≥0.25)
4. PC espera 2.0s confirmación (ojos continuamente abiertos)
5. PC → 'c' → Carrinho recibe comando de continuar
6. Carrinho espera 2.0s adicionales (timeout seguridad)
7. Carrinho REANUDA navegación

Justificación del doble timeout:
- Primer timeout (2s en PC): Verifica que no fue un parpadeo aislado
- Segundo timeout (2s en vehículo): Margen de seguridad adicional antes de moverse

3.4. Navegación Autónoma y Control

Estrategia implementada:
SI (distancia ≥ 20 cm):
→ Avanzar recto
SI (distancia < 20 cm):
→ Ejecutar maniobra de evasión:
1. Parar (200ms)
2. Retroceder (400ms)
3. Girar derecha (500ms)
4. Continuar
SI (parado_por_bluetooth == true):
→ PARADA INMEDIATA (prioridad absoluta)

FreeRTOS - Tareas implementadas:
- vUltrasonicTask (prioridad 2): Lectura sensor cada 100ms
- vMotorControlTask (prioridad 1): Control navegación
- vBluetoothTask (prioridad 1): Recepción comandos seguridad

4. ANÁLISIS DE COSTES

4.1. Presupuesto Detallado del Prototipo

Coste total hardware (precios verificados enero 2026): 42.49 €
- Componentes electrónicos: 38.63 €
- Margen para imprevistos: 3.86 €
Versión PCB profesional: +15 € = 57.49 €

4.2. Análisis de Costes de Desarrollo

- Horas de desarrollo: 80 horas (documentadas en commits GitHub)
- Coste hora (referencia INE): 10 €/h
- Coste desarrollo: 80h × 10€/h = 800 €
- Coste primer prototipo: 42.49€ + 800€ = 842.49 €

4.3. Viabilidad Comercial para Pequeñas Series

Producción de 100 unidades:

- Coste hardware/unit. (escala): ~30 €
- Desarrollo amortizado/unit.: 800€ / 100 = 8 €
- Coste unitario total: 38 €
- PVP sugerido (60% margen): 38€ × 1.6 = 60.80 €

Comparativa con mercado:

- Kits educativos similares: 80-150 €
- Sistemas detección fatiga comerciales: 200-500 €
- Nuestro sistema integrado: 61 € (altamente competitivo)

5. IMPLEMENTACIÓN Y VALIDACIÓN

5.1. Configuración y Montaje

1. Montaje mecánico: Ensamblaje del chasis, motores y ruedas
2. Electrónica: Conexión de componentes según esquema
3. Programación:
   - Pico W: Compilación con SDK Raspberry Pi + FreeRTOS
   - PC: Instalación de OpenCV + MediaPipe + Python
4. Configuración Bluetooth: Emparejamiento HC-05 con adaptador USB

5.2. Pruebas Funcionales y Resultados

Prueba	Resultado Esperado	Resultado Obtenido	Validación
Detección sueño	Envío 'p' al cerrar ojos 1.5s	Correcto	Algoritmo funcional
Falso positivo (risa)	No envía 'p' al reír	Correcto	EAR+MAR robusto
Verificación despierto	Envío 'c' tras 2s ojos abiertos	Correcto	Protocolo seguridad
Timeout vehículo	Envío 'c' tras 2s ojos abiertos	Correcto	Doble seguridad
Evasión obstáculos	Para a 20cm, ejecuta maniobra	Correcto	Navegación autónoma
Prioridad Bluetooth	Parada prevalece sobre obstáculos	Correcto	Jerarquía seguridad

5.3. Guía de Prueba Rápida para el Profesor

Para facilitar la evaluación sin necesidad de hardware físico, se incluye en el repositorio un script de prueba rápida que permite:

1. Probar el algoritmo EAR+MAR en cualquier PC:
    - Solo requiere webcam y Python instalado
    - Muestra métricas en tiempo real
    - Demuestra discriminación entre fatiga y risas

2. Verificar la lógica de detección:
    - Cerrar ojos 1.5+ segundos → Activa alerta visual (simula envío 'p')
    - Sonreír/reír → NO activa alerta (demuestra robustez EAR+MAR)
    - Abrir ojos tras alerta → Simula envío 'c' tras timeout

3. Instrucciones de ejecución:

```bash
# 1. Instalar dependencias mínimas
pip install opencv-python mediapipe numpy
```

2. Ejecutar script de demostración
python prueba_detector_fatiga.py

3. Probar comportamientos:
- 'q' para salir
- 'r' para resetear contadores
- Cerrar ojos >1.5s para ver alerta
- Sonreír para ver que no genera alerta

4. Explicación visual mostrada:
- Panel izquierdo: EAR y MAR en tiempo real (color según estado)
- EAR < 0.25: Texto rojo (ojos cerrados)
- MAR > 0.2: Texto naranja (boca abierta)
- Alerta fatiga: Rectángulo rojo inferior con mensaje "VEHICULO SE DETENDRIA AQUI"

Este script permite verificar el núcleo inteligente del proyecto (lógica EAR+MAR y protocolo de estados) sin necesidad del hardware físico.

6. CONCLUSIONES Y TRABAJO FUTURO

6.1. Conclusiones

Sistema funcional completo que integra visión artificial y control robótico
Algoritmo EAR+MAR robusto validado experimentalmente (0 falsos positivos por risas)
Protocolo de seguridad de dos fases con timeout de verificación
Arquitectura FreeRTOS estable en Raspberry Pi Pico W
Navegación autónoma con evasión de obstáculos funcional
Coste ultrabajo verificable (42.49€ hardware)
Documentación completa y proyecto open-source disponible

6.2. Limitaciones Identificadas

- Dependencia de iluminación: Detección facial sensible a condiciones lumínicas extremas
- Alcance Bluetooth: Limitado a ~10 metros (suficiente para prototipo)
- Maniobra de evasión: Simple, no adaptable a diferentes tipos de obstáculos
- Procesamiento en PC: Dependencia de ordenador externo para visión

6.3. Trabajo Futuro

1. Procesamiento embebido: Ejecutar MediaPipe en el propio vehículo con aceleración hardware
2. Navegación avanzada: Implementar SLAM básico con múltiples sensores
3. App móvil: Interfaz para configuración y monitorización en tiempo real
4. Certificaciones: Pruebas de compatibilidad electromagnética (EMC)
5. Carcasa profesional: Diseño impreso 3D para protección y estética
6. Protocolo seguridad mejorado: Incorporar redundancia en comunicaciones

6.4. Aplicabilidad Industrial

El sistema demuestra conceptos aplicables a:

- Vehículos autónomos industriales: Sistemas de seguridad para AGVs
- Maquinaria pesada: Monitorización de operadores en turnos largos
- Entornos logísticos: Seguridad en almacenes automatizados
- Educación tecnológica: Plataforma para enseñanza de sistemas embebidos y visión artificial

7. REFERENCIAS

1. Raspberry Pi Foundation. *RP2040 Datasheet*. 2021.
2. Soukupová, T.; Čech, J. *Real-Time Eye Blink Detection using Facial Landmarks*. 21st Computer Vision Winter Workshop, 2016.
3. Pasaribu, N. T. B. et al. *Drowsiness Detection According to the Number of Blinking Eyes using Eye Aspect Ratio and Mouth Aspect Ratio*. ICLICK 2018.
4. Google LLC. *MediaPipe Face Mesh Documentation*.
5. *FreeRTOS™ Kernel Documentation*.
6. Precios reales componentes (enero 2026):
   - Raspberry Pi Pico W: 5.18€ (Farnell, 377-8756)
   - TB6612FNG: 1.99€ (Farnell)
   - MP1584EN: 3.81€ (Mouser)
   - HC-05: 0.38€ (AliExpress)
   - HC-SR04: 0.99€ (AliExpress)
   - Kit chasis 2WD: 7.09€ (AliExpress)
   - Batería LiPo 7.4V 3000mAh: 6.69€ (AliExpress)

7. Instituto Nacional de Estadística (INE). *Encuesta de Población Activa*. 2024.

---

8. APÉNDICES

A. Repositorio GitHub - Código completo, documentación y guías
B. Mediciones de consumo - Datos y gráficos experimentales
C. Script prueba rápida - Para evaluación sin hardware físico
D. Vídeos demostración - Funcionamiento completo del sistema
* https://youtube.com/shorts/a9cLT_Jo-cY?feature=share

INFORMACIÓN ADICIONAL PARA EL PROFESOR

Para una experiencia práctica rápida:
El repositorio GitHub incluye un archivo `prueba_rapida.py` que permite:
1. Probar el algoritmo EAR+MAR en cualquier ordenador con webcam
2. Verificar la lógica de detección sin necesidad del hardware físico
3. Observar la discriminación entre fatiga real y gestos como risas

Estructura del repositorio:

```
Proyecto_Informatica_Industrial/
├── README.md                          # Documentación principal
├── GUIA_TESTE_PROFESSOR.md            # Guía para evaluación rápida
├── .gitignore                         # Configuración Git
├── videos/                            # Demostraciones en video
│   ├── WhatsApp Video 2026-01-14 at 02.55.06.mp4
│   └── WhatsApp Video 2026-01-14 at 02.56.31.mp4
├── software/
│   ├── pc_fatiga_detector/            # Sistema de visión (Python)
│   │   ├── prueba_rapida.py           # Versión prueba profesor
│   │   ├── detector_original.py       # Sistema completo Bluetooth
│   │   └── requirements.txt           # Dependencias
│   └── pico_vehicle_control/          # Firmware Pico (C/FreeRTOS)
│       ├── src/                       # Código fuente
│       ├── configs/                   # Configuraciones
│       └── local-libs/                # Bibliotecas
├── hardware/                          # Diseño electrónico
├── documentos_analise/                # Análisis económico
│   └── capturas_precos/               # Screenshots de precios
└── Memoria_Proyecto.pdf               # Esta memoria
```

Comandos para prueba inmediata:

```bash
# 1. Clonar repositorio
git clone [URL_GITHUB]

# 2. Instalar dependencias mínimas
pip install opencv-python mediapipe numpy

# 3. Ejecutar demostración
python software/prueba_rapida.py
```

Esta guía permite evaluar el núcleo algorítmico del proyecto en menos de 5 minutos, demostrando la robustez del sistema EAR+MAR y la lógica de seguridad implementada.