Universidad Nacional de Quilmes

Trabajo Práctico Final: Introducción a la programación de microcontroladores con Arduino

Alumno: Bertolini Agaras, Victor H.

Materia: Introducción a la programación de microcontroladores con Arduino

Profesor: Jose Luis Di Base

Cuatrimestre: 1er cuatrimestre 2026

Fecha de entrega: 16/07

Descripción del Proyecto

Sistema de Asistencia de Estacionamiento Avanzado (Radar IoT)

El presente proyecto consiste en el diseño y desarrollo de un sistema de asistencia de estacionamiento inspirado en los vehículos modernos. Tomando como base conceptual los medidores de distancia simples, este proyecto escala la complejidad técnica implementando una arquitectura de hardware y software de tres capas:

1. Arreglo Multisensor (Hardware): Utilización de tres sensores ultrasónicos (HC-SR04) montados estratégicamente en una estructura de madera (simulando un paragolpes) para barrer áreas de colisión (izquierda, centro y derecha), más una cámara de video integrada
2. Procesamiento y Transmisión (Microcontrolador): Una placa ESP32-CAM actúa como el cerebro del sistema. Procesa las distancias de forma no bloqueante, evalúa el nivel de peligro global, activa alertas sonoras (buzzer) y levanta su propia red Wi-Fi (Access Point). Emite un servidor HTTP que entrega los datos en formato JSON y realiza streaming de video (MJPEG) en tiempo real.
3. Dashboard Frontend (Software Cliente): Una aplicación web desarrollada en React.js (ejecutada en PC o móvil mediante red local) que consume la API del microcontrolador y renderiza un panel de instrumentos minimalista, adaptativo (responsive) y de baja latencia para el "conductor".

Materiales Utilizados y Adquisición

A continuación, se detallan los componentes electrónicos y herramientas utilizadas en el armado definitivo:

Componente	Función en el sistema	Adquisición	Precio Aprox.
ESP32-CAM + ESP32-CAM-MB	Microcontrolador central, Wi-Fi y streaming de video.	MercadoLibre	$ 28.500
3x HC-SR04	Sensores ultrasónicos de proximidad.	MercadoLibre	$ 3.500 (x/unidad)
Buzzer Activo	Alarma sonora de colisión.	MercadoLibre	$ 3.700
Cable USB Modificado	Cable reciclado (cortado) para extraer los filamentos de 5V y GND directamente.	Reciclado	$ 0
Base de madera	Soporte estructural rígido para la	Ya tenía en	$ 0

y Tornillos	presentación del hardware.	casa	
Cargador 5V 2A	Fuente de alimentación principal.	Reciclado	$ 0
Cables Jumper Hembra-Macho, Estaño y Termocontraíble	Conexiones, empalmes y aislamiento	Tienda de Electrónica Local	$ 3.900
Filamento PLA	Impresión 3D de la carcasa.	Impresión propia	$ 0

(Nota: La ESP32-CAM-MB es una placa complementaria que se usa para el desarrollo de programas de en placas ESP32-CAM que cuando se pasa a fase de "producción" se retira).

Esquemático y Diagrama de Conexiones

El conexionado del hardware se diseñó de forma directa empalmando las líneas de alimentación extraídas del cable USB cortado hacia todos los componentes, garantizando que los picos de corriente no pasen por pistas débiles.

- **Alimentación (Cable USB adaptado):**
  - Filamento Rojo (5V) ➔ Empalme directo ➔ Pin 5V (ESP32) y pines VCC (Sensores).
  - Filamento Negro (GND) ➔ Empalme directo ➔ Pines GND (ESP32 y Sensores).
- **Sensores HC-SR04:**
  - Todos los pines TRIG ➔ Pin 12 (ESP32).
  - ECHO Izquierdo ➔ Pin 13 (ESP32).
  - ECHO Central ➔ Pin 14 (ESP32).
  - ECHO Derecho ➔ Pin 15 (ESP32).
- **Buzzer:**
  - Pin Positivo ➔ Pin 2 (ESP32).
  - Pin Negativo ➔ Empalme a GND común.

Paso a Paso del Armado

Paso a Paso del Armado

1. **Prototipado en Protoboard: Se realizaron las pruebas iniciales de lógica y código utilizando puentes de conexión directa para validar el correcto funcionamiento de cada sensor de forma individual.**
2. **Preparación de la Alimentación Principal: Se recicló un cable USB estándar cortando uno de sus extremos. Se peló la funda exterior, se cortaron al ras los cables de datos y se expusieron los filamentos de alimentación puro: Rojo (5V) y Negro (GND).**
3. **Distribución de Energía (Nodos): A partir del cable USB pelado, se realizaron empalmes tipo "pulpo" (soldando varios cables jumper a los cables rojo y negro principales) para crear una distribución en paralelo. Todos los empalmes fueron soldados con estaño y fuertemente aislados con tubo termocontraíble para evitar cortocircuitos.**
4. **Cableado de Señal: Se soldaron/conectaron los cables para los pines TRIG comunes, ECHOs individuales y el pin del buzzer, manteniendo la misma lógica de aislamiento para evitar falsos contactos por vibración.**
5. **Ensamblaje Mecánico y Ruteo (Diseño Expuesto): En lugar de ocultar la electrónica en una caja cerrada, se optó por un diseño modular a la vista para la demostración. Se imprimieron en 3D soportes/carcasas (cases) individuales para los tres sensores, para la placa principal y para el buzzer. Estos módulos se atornillaron firmemente a una base de madera, posicionando los sensores en abanico para abarcar el área de detección sin interferencias (crosstalk). Finalmente, se realizó un ruteo ordenado de los cables a la vista con precintos, garantizando la prolijidad del conjunto y permitiendo apreciar la arquitectura del hardware de forma directa.**

Problemas que Surgieron y Cómo se Resolvieron

Durante la integración de hardware avanzado y desarrollo web, surgieron desafíos técnicos de bajo nivel que fueron resueltos de la siguiente manera:

- **Problema 1: Caída de tensión por picos de consumo (Brownout)**

- **Síntoma:** Al inicializar el servidor Wi-Fi, la placa colapsaba y se reiniciaba en bucle imprimiendo Brownout detector was triggered.
- **Solución:** Los picos de encendido de la antena superaban la estabilización inicial. Se solucionó por hardware utilizando un cargador robusto de 5V/2A y pistas directas de alimentación; y por software, inyectando la librería soc/rtc_cntl_reg.h y apagando el registro de caída de voltaje al inicio del setup: WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);.

- **Problema 2: Conflicto de Memoria PSRAM (LoadProhibited)**
  - **Síntoma:** Al asignar el Buzzer al Pin 16, la placa sufría un *Kernel Panic* (Fatal exception: LoadProhibited) al intentar levantar el streaming de video.
  - **Solución:** Se investigó el Datasheet del procesador ESP32 y se descubrió que el Pin 16 está internamente reservado para la memoria PSRAM (esencial para procesar el video de la cámara). Se migró el buzzer al Pin 2, resolviendo el colapso de memoria.
- **Problema 3: I2C Timeout al inicializar la Cámara**
  - **Síntoma:** El monitor serie arrojaba I2C hardware timeout detected. La lente no respondía a la inicialización.
  - **Solución:** Se determinó que encender el Wi-Fi y la Cámara en el mismo bloque de código generaba una doble caída de tensión, desestabilizando el bus I2C. Se solucionó agregando un delay(1000) entre el encendido del Wi-Fi y el inicio de la cámara, dándole tiempo a los capacitores de la placa para recuperar su carga nominal.
- **Problema 4: Limitación del Streaming de Video (MJPEG)**
  - **Síntoma:** Al acceder a la aplicación desde una PC y un celular simultáneamente, el video solo cargaba en el primer dispositivo conectado.
  - **Solución:** Se determinó que, debido a las limitaciones de RAM del microcontrolador, la ESP32-CAM solo puede mantener abierto un (1) socket para streaming de video MJPEG a la vez. La solución de arquitectura fue utilizar un único dispositivo principal (Ej. celular o tablet) como pantalla exclusiva ("host") durante el funcionamiento del sistema.

Software Necesario y Versiones

Para replicar, modificar o ejecutar el proyecto se requieren las siguientes herramientas:

- **Microcontrolador (C++):**
  - Arduino IDE (Versión 2.3.x o superior).
  - Gestor de placas de Espressif Systems: esp32 (Versión 2.0.11+).
  - Librería externa: ArduinoJson (Versión 6.21.3+ o v7).
- **Frontend (React.js):**
  - Node.js (Versión 18.x o superior).
  - React.js (Versión 18.2.0).
  - Vite (Versión 5.x) como empaquetador y servidor de desarrollo local.
  - *Nota de ejecución:* Se utiliza el comando npm run dev -- --host para exponer la interfaz en la red local generada por el auto.

Anexos: Medios y Referencias

- **Repositorio del Proyecto con el Front y el código Arduino:** https://github.com/Bertolini-Victor/TrabajoFinal_IntroduccionALosMicrocontroladoresConArduino_UNQ/tree/main