



RECUERDAME

INFORME DESCRIPTIVO

Recuerdame Web:

<https://recuerdamecpf.lat/>

Trello:

<https://trello.com/b/dzCC3Ma7/kanban>

GitHub:

<https://github.com/impatrq/Recuerdame>

Instagram:

<https://instagram.com/recuerdamecpf>

Índice

1. Presentación del equipo
2. Introducción
 - a. Objetivo
 - b. Descripción de la solución
 - c. Descripción de la solución
 - d. Segmento destino y alcance
 - i. Destino
 - ii. Alcance
 - e. Captura Representativa
 - f. Diagrama de bloques
 - g. Resultado conseguido
3. Software
 - a. Diagrama de bloques
 - b. Lenguajes utilizados
 - c. Capturas del código
 - i. Programa de reconocimiento
 - ii. App
 - d. Capturas Interfaces
4. Hardware
 - a. ¿Qué Micro usamos? ¿Qué Problemas trajo? ¿Soluciones?
 - b. Alimentación
 - c. Diagrama de bloques
 - d. Datasheets
5. Estructura
 - a. Diagrama general de la estructura
 - i. Carcasa superior del lente
 - ii. Patillas del lente
 - iii. Marco del lente
 - iv. Soportes de exhibición
 - b. Software de diseño utilizado
 - c. Descripción de cada parte de la estructura

Presentación del equipo:

- **Carlos Gabri Krizak**



- 8 a 10 horas semanales (300 horas aprox.)
- Encargado del software y conexiones de “Recuerdame”, incluyendo web app y software de reconocimiento. Creador de la base de datos y de la conexión entre ambos

- **Facundo Mendez**



- 8 a 10 horas semanales (300 horas aprox.)
- Encargado del diseño desde 0 de los lentes en autoCAD, de la impresión 3D y del ensamblaje de los mismos.

- **Pedro Amenta**



-
- 8 a 10 horas semanales (300 horas aprox.)
- Encargado del marketing del equipo y la búsqueda de sponsors, exponiendo el proyecto en redes sociales y desarrollando una página web propia desde 0

Introducción

Objetivo

El objetivo de Recuérdame es desarrollar un sistema de asistencia inteligente que ayude a personas con pérdida de memoria o deterioro cognitivo a reconocer a las personas de su entorno y mantener su autonomía.

El sustento del proyecto nace de una motivación personal: la experiencia con un familiar que padecía demencia y que, por vergüenza y confusión, ocultaba su dificultad para recordar.

Esta situación evidenció la necesidad de una herramienta accesible, portable y discreta que pudiera brindar apoyo al usuario sin depender permanentemente de un acompañante.

Descripción de la solución

La solución propuesta consiste en un par de **lentes inteligentes** equipados con una Raspberry Pi Zero 2, una cámara y un sistema de audio Bluetooth.

El dispositivo realiza reconocimiento facial mediante **tres modelos pre entrenados** optimizados para hardware de bajo consumo.

Cuando identifica a una persona conocida, anuncia mediante audio su nombre y relación con el usuario.

La solución se complementa con una **WebApp** donde familiares o profesionales pueden:

- Registrar personas.
- Realizar preguntas cognitivas diarias.
- Consultar estadísticas automáticas.
- Evaluar el progreso o deterioro cognitivo del paciente.

El sistema funciona de manera autónoma, inalámbrica y portable, pensado para uso cotidiano.

Segmento destino y alcance

Segmento destino:

- Personas con pérdida de memoria leve o moderada.
- Pacientes con Alzheimer u otras demencias.
- Adultos mayores en seguimiento cognitivo.
- Familias que buscan una herramienta de apoyo.
- Profesionales de la salud, cuidadores y acompañantes terapéuticos.

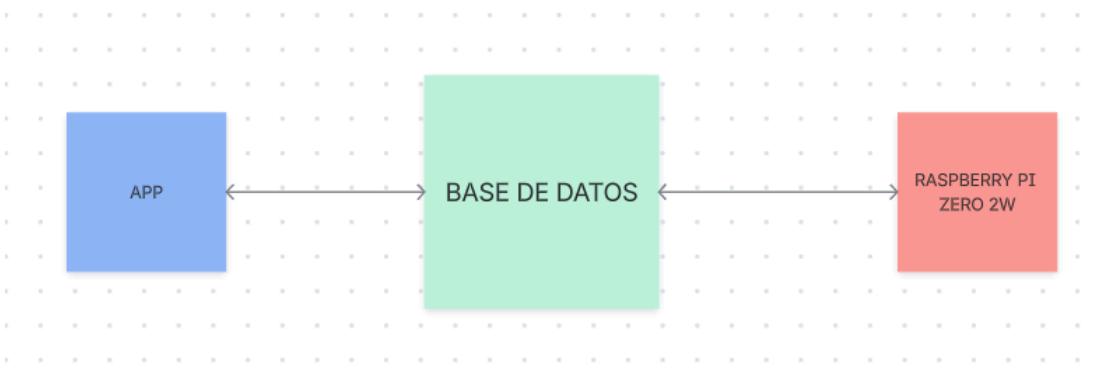
Alcance del proyecto:

- Prototipo funcional totalmente autónomo.
- Reconocimiento facial en tiempo real.
- Asistente auditivo a través de cualquier dispositivo Bluetooth.
- Plataforma web para registro, controles cognitivos y estadísticas.
- Sistema escalable para agregar más pacientes o personas.

Captura representativa



Diagrama en bloques



Resultado conseguido

El sistema *Recuérdame* funciona correctamente como prototipo de asistencia cognitiva.

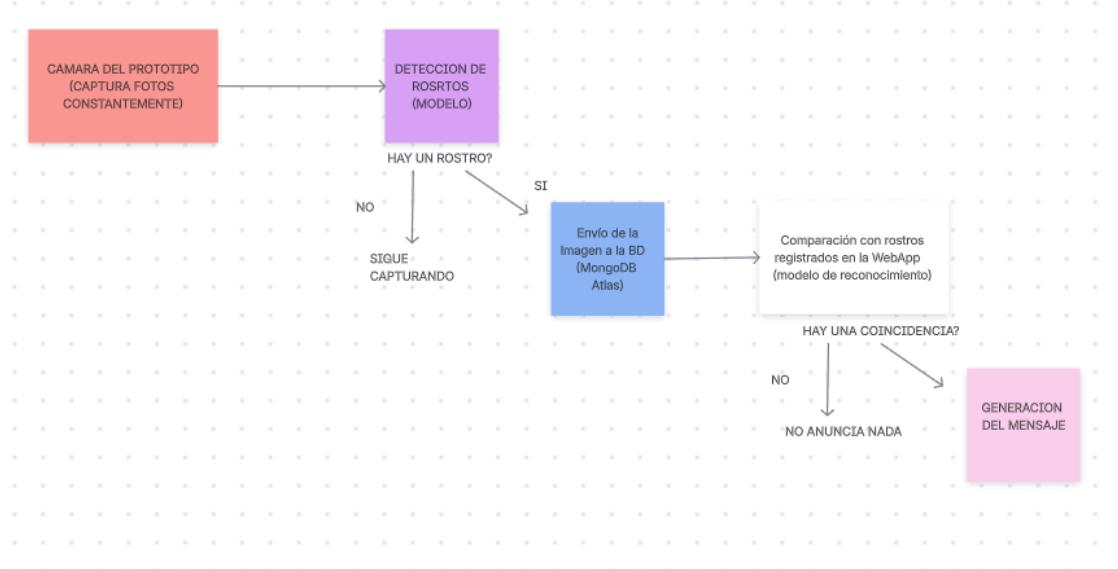
Los resultados más relevantes son:

- El reconocimiento facial obtuvo una precisión aproximada del **80%** en pruebas reales con los integrantes del equipo.
- El sistema funciona de manera **autónoma**, sin PC, con arranque automático.
- La WebApp permite registrar personas, hacer preguntas cognitivas y ver estadísticas semanales y mensuales.
- La comunicación entre los lentes, el servidor y la base de datos MongoDB Atlas funciona de forma estable.
- La página web informativa <https://recuerdamecpf.lat> se visualizó correctamente en todos los dispositivos.
- La autonomía del prototipo es de **3 horas** con una carga de 1 hora.

- El sistema es portable, ligero y suficientemente rápido para el uso diario.

SOFTWARE

Diagrama de bloques



Lenguajes Utilizados

- **Python:** utilizado en la Raspberry Pi Zero 2 para el procesamiento de imágenes, la captura continua de fotogramas, la detección de rostros y el envío de imágenes al servidor. También se utilizó para manejar los procesos internos del dispositivo.
- **JavaScript (Node.js):** empleado para el servidor backend, responsable de recibir las imágenes, compararlas con la base de datos mediante modelos de reconocimiento facial y devolver una coincidencia cuando existe.
- **HTML, CSS y JavaScript:** utilizados en la WebApp que desarrollamos para cargar personas, gestionar pacientes, visualizar estadísticas, realizar juegos cognitivos y administrar la información general del sistema.
- **MongoDB + Mongoose:** usados para almacenar los perfiles de personas, sus fotografías, las respuestas a las preguntas diarias, las estadísticas de rendimiento y los resultados de juegos cognitivos.

Capturas del código

- Programa de reconocimiento

Conexion a Mongo (Base de Datos)

```
MONGODB_URI = "mongodb+srv://recuerdamecpf:recuerdame123@recuerdamecluster.yit...  
try:  
    client = MongoClient(MONGODB_URI)  
    client.admin.command('ping')  
    db = client.recuerdame  
    colección = db.personas  
    print("☑ Python: Conectado a MongoDB Atlas")  
except Exception as e:  
    print(f"✗ Python: Error al conectar con MongoDB Atlas: {e}")  
colección = None
```

```
try:  
    image = face_recognition.load_image_file(local_path)  
    face_encodings = face_recognition.face_encodings(image)  
  
    if face_encodings:  
        fotos_codificadas.append(face_encodings[0])  
        nombres.append(nombre)  
        relaciones.append(relacion) # Guardamos la relación  
        print(f"☑ Codificación creada para: {nombre} ({relacion})")  
    else:  
        print(f"⚠ No se detectó ninguna cara en la foto de {nombre}.")  
  
except Exception as e:  
    print(f"✗ Error al procesar la foto de {nombre}: {e}")
```

- Convierte cada foto de la base de datos en un *encoding* (vector de 128 valores).

Ese encoding es lo que después se compara matemáticamente. También está la parte en la que guarda toda la información necesaria para identificar personas sin depender más de la base de datos. Esto no hace que sea completamente funcional sin wifi, pero si que cuando arranque pueda serlo.

- La Pi saca una foto, la guarda temporalmente y la procesa cuadro por cuadro.

```
try:  
    subprocess.run(["rpicam-still", "--timeout", "100", "--nopreview", "-o", "/dev/null"], check=True, capture_output=True)  
    print("☑ rpicam-still parece funcionar correctamente.")  
except FileNotFoundError:  
    print("✗ ERROR: 'rpicam-still' no encontrado. Asegúrate de que las rpicam-apps estén instaladas.")  
    exit()  
except subprocess.CalledProcessError as e:  
    print(f"✗ ERROR al probar rpicam-still: {e.stderr.decode()}")  
    exit()  
except Exception as e:  
    print(f"✗ Error inesperado al probar rpicam-still: {e}")  
    exit()
```

- Encuentra caras en la imagen. Si no hay caras, no hace nada. Si hay, pasa a comparar.

```
face_locations = face_recognition.face_locations(rgb_frame)
if face_locations:
    print(f"\n⚠ Se detectaron {len(face_locations)} cara(s) en el frame.")
```

- Esta línea es **el reconocimiento**: determina si la cara detectada coincide con alguien de la base de datos.

```
matches = face_recognition.compare_faces(fotos_codificadas, face_encoding)

nombre_identificado = "Desconocido"
relacion_identificada = ""
```

- Recupera automáticamente el **nombre** y el **parentesco** de la persona reconocida.

```
if True in matches:
    first_match_index = matches.index(True)

    nombre_identificado = nombres[first_match_index]
    relacion_identificada = relaciones[first_match_index] |
```

- El sistema informa verbalmente quién es la persona reconocida: *"Persona reconocida: Camila. Es tu hija."*

```
# Imprimir y Anunciar el resultado
if nombre_identificado != "Desconocido":
    mensaje_impresso = f" - Cara en [{top},{right},{bottom},{left}] reconocida: {nombre_identificado} (Relación: {relacion_identificada})"

# CREAMOS LA FRASE PARA LA VOZ EN ESPAÑOL
frase_voz = f"Persona reconocida: {nombre_identificado}. Es {relacion_identificada}."

# Ejecutamos el comando de voz con espeak en español
os.system(f'espeak -v es "{frase_voz}" 2>/dev/null')
```

● App

- Conexión con Mongo

```
mongoose.connect("mongodb+srv://recuerdamecpf:recuerdame123@recuerdamecluster.yitpqed.mongodb.net/recuerdame?retryWrites=true&w=majority&appName=Recuerdame")
    .then(() => console.log("✅ Conectado a MongoDB Atlas")) // <- ¡Asegurate que esté esta linea!
    .catch(error => console.error("❌ Error al conectar con MongoDB Atlas:", error));
```

- Esquema de preguntas

```
const PreguntaSchema = new mongoose.Schema({
  texto: { type: String, required: true },
  tipo: { type: String, enum: ['multiple_choice', 'text_input', 'binary'], default: 'text_input' },
  opciones: { type: [String], default: [] },
  respuestaCorrecta: { type: String },
  fechaCreacion: { type: Date, default: Date.now }
});
const Pregunta = mongoose.model("Pregunta", PreguntaSchema);
```

- Carga de Persona

```
app.get("/personas", async (req, res) => {
  try {
    const personas = await Persona.find();
    res.json(personas);
  } catch (error) {
    res.status(500).json({ error: "✗ Error al obtener personas" });
  }
});

app.post("/persona", upload.single('foto'), async (req, res) => {
  try {
    if (!req.body.nombre || !req.body.relacion || !req.file) {
      return res.status(400).json({ error: "⚠ Todos los campos son obligatorios." });
    }

    const nuevaPersona = new Persona({
      nombre: req.body.nombre,
      relacion: req.body.relacion,
      foto: `/uploads/${req.file.filename}`
    });

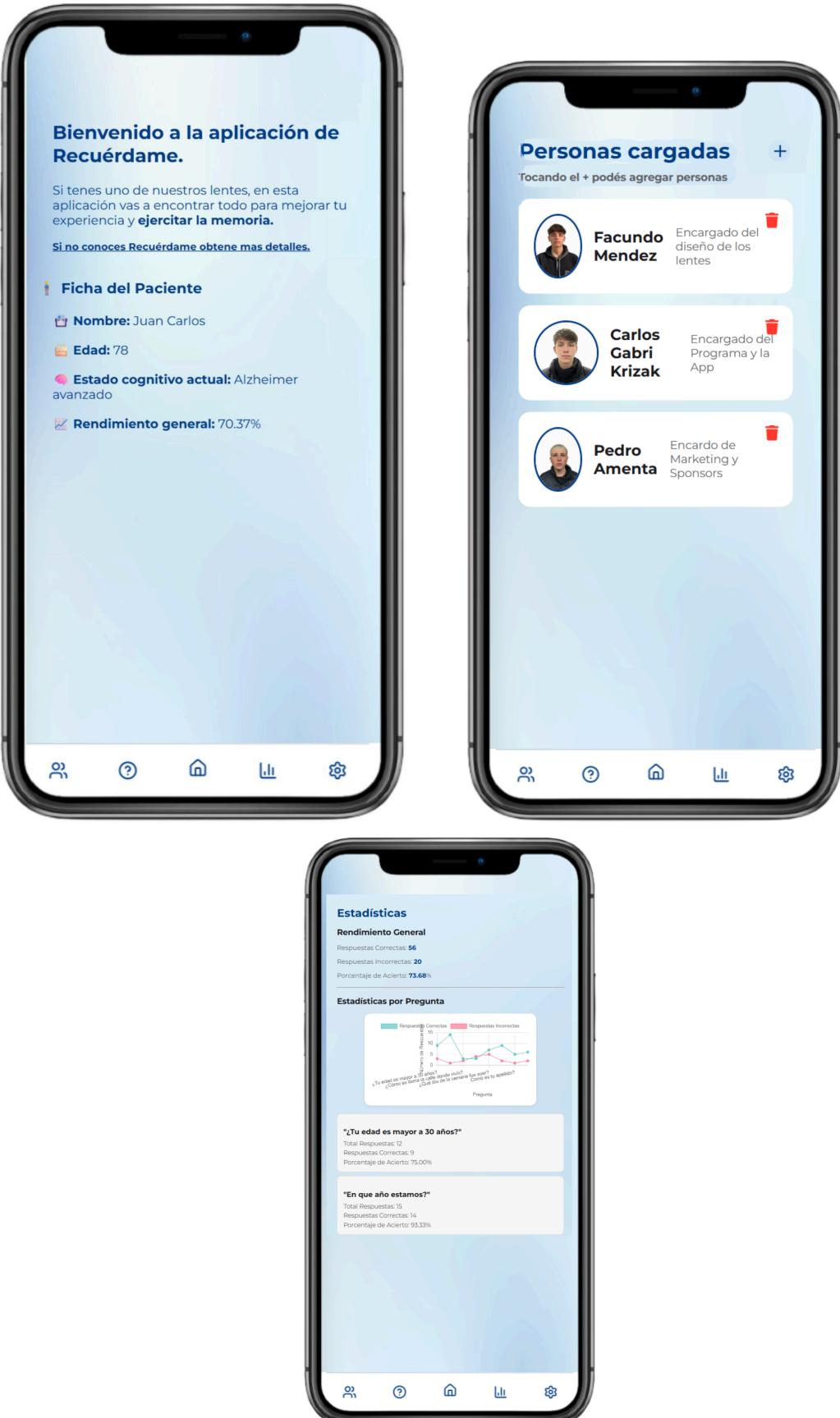
    await nuevaPersona.save();
    res.json({ message: "✓ Persona registrada correctamente", persona: nuevaPersona });
  } catch (error) {
    console.error("Error al registrar persona:", error);
    res.status(500).json({ error: "✗ Error al registrar persona: " + (error.message || "Error desconocido") });
  }
});
```

- Ruta para obtener estadisticas

```
app.get("/estadisticas", async (req, res) => {
  try {
    const totalRespuestas = await Respuesta.countDocuments();
    const respuestasCorrectas = await Respuesta.countDocuments({ acierto: true });
    const respuestasIncorrectas = await Respuesta.countDocuments({ acierto: false });

    const estadisticasPorPregunta = await Respuesta.aggregate([
      {
        $group: {
          id: "$pregunta",
          total: { $sum: 1 },
          correctas: { $sum: { $cond: ["$acierto", 1, 0] } }
        }
      },
      {
        $lookup: {
          from: "preguntas",
          localField: "_id",
          foreignField: "_id",
          as: "detallesPregunta"
        }
      },
      {
        $unwind: "$detallesPregunta"
      },
      {
        $group: {
          pregunta: "$detallesPregunta.pregunta",
          total: { $sum: 1 },
          correctas: { $sum: { $cond: ["$acierto", 1, 0] } }
        }
      }
    ]);
    res.json({ totalRespuestas, respuestasCorrectas, respuestasIncorrectas, estadisticasPorPregunta });
  } catch (error) {
    console.error("Error al obtener estadísticas:", error);
    res.status(500).json({ error: "✗ Error al obtener estadísticas" });
  }
});
```

Capturas interfaces



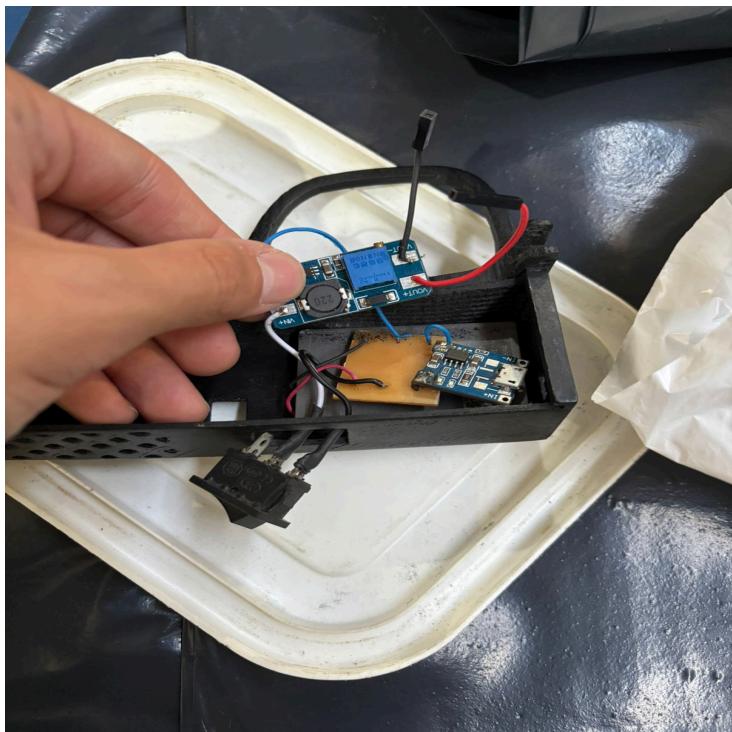
Hardware

¿Qué Micro usamos? ¿Qué Problemas trajo? ¿Soluciones?

Utilizamos una Raspberry pi zero 2w como cerebro del proyecto, esta nos pareció la mejor calidad/precio/tamaño a la hora de elegir un micro. Esto nos trajo complicaciones a la hora de desarrollar el software de reconocimiento, ya que al tener tan solo 512 mb de RAM, no logramos correr un Reconocimiento Facial convencional (Como OpenCV). La solución la encontramos con “Modelos Pre entrenados”, los cuales con solo descargar un archivo de Github y mencionarlos en el código, ya reemplaza la librería. Esto limita un poco el rendimiento, pero nos permite tener un prototipo funcional. Todo este programa lo realizamos en Visual, y para hacer la prueba lo pasábamos por Terminus.

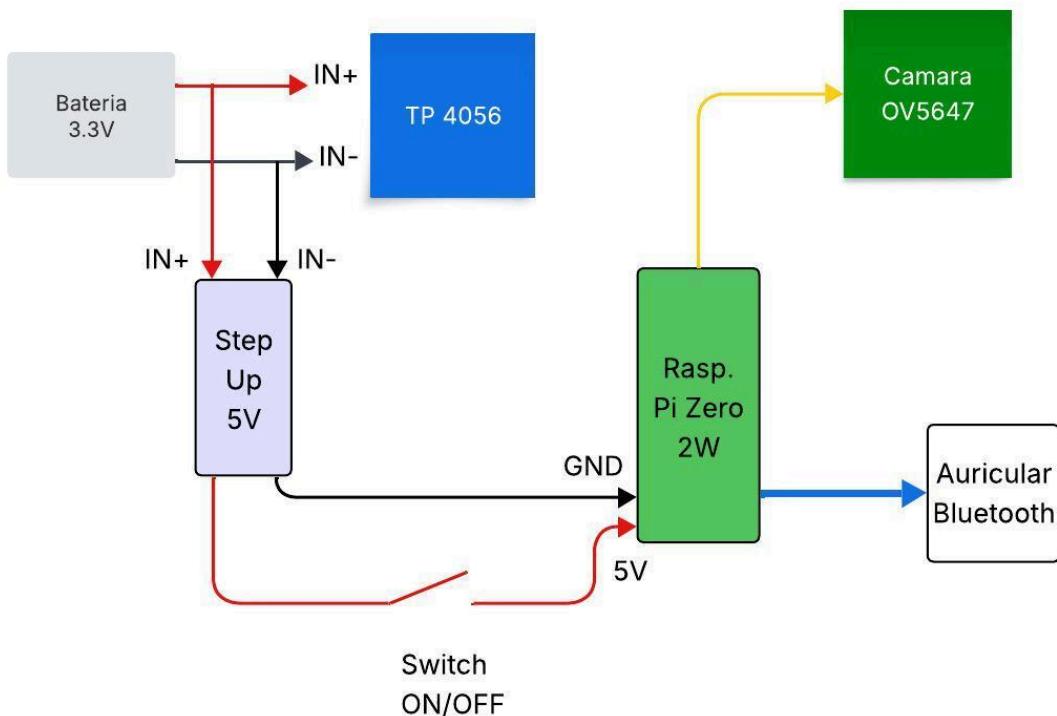


Alimentación



A la hora de alimentar el prototipo, usamos una batería Li-po cualquiera, (una de joystick por ejemplo) y un TP4056, el cual se encarga de que no se sobrecargue la batería y que sirva para cargarla. El problema fue a la hora de alimentar la raspy, ya que el voltaje que nos puede entregar la batería es de 3,7 V y necesitamos 5, por eso entra el Step-Up, el cual nos permite llegar al voltaje que necesitamos.

Diagrama de bloques



Datasheets

Step-Up:

<https://www.alldatasheet.com/datasheet-pdf/view/1131968/ETC1/MT3608.html>

TP4056:

<https://www.alldatasheet.com/html-pdf/view/1132405/ASIC/TP4056/344/3/TP4056.html>

ESTRUCTURA

Diagrama general de la estructura

La estructura del sistema Recuérdame se compone de cuatro elementos principales:

1. Carcasa superior del lente

Ubicada sobre el marco, contiene todos los componentes electrónicos y su tapa desmontable con guías.

2. Patillas del lente (patas)

Estructura liviana y convencional que proporciona soporte y equilibrio, sin alojar componentes internos.

3. Marco del lente

Base donde se apoya la carcasa superior, de diseño simple para mantener comodidad y estética.

4. Soportes de exhibición

Piezas impresas en dos partes, luego pegadas sobre el podio para sostener los lentes durante la exposición.

Software de diseño utilizado

Para el diseño estructural del proyecto se utilizó AutoCAD 2022, mediante:

- Modelado en 3D por extrusión y edición de sólidos.
- Operaciones booleanas para unión y sustracción de volúmenes.
- Diseño paramétrico para adaptación de espesores y espacios internos.
- Exportación a .STL para impresión 3D en Ultimaker Cura.

Descripción de cada parte de la estructura

1. Carcasa superior

- Pieza principal donde se integra todo el sistema electrónico.
- Cuenta con una tapa desmontable mediante guías, permitiendo acceder al interior sin desmontar todo el lente.
- Su interior está diseñado para alojar con exactitud:
 - Raspberry Pi Zero 2 W
 - Módulo de cámara Raspberry
 - Batería Li-Po 3.7V
 - TP4056 (módulo de carga)
 - Convertidor Step-Up
 - Posee perforaciones para:
 - Switch de encendido/apagado
 - Puerto micro-USB de carga
 - Incluye una rejilla de ventilación en la parte superior para disipar calor del Step-Up.
 - Internamente dispone de separadores que mantienen cada componente fijo y protegido.

2. Patillas (patas)

- Patas de diseño convencional para mantener la estética de un lente común.
 - No alojan componentes electrónicos.
 - Se ajustó su longitud mediante una reducción de 8 mm para compensar el peso frontal de la carcasa y mejorar el equilibrio.
 - Impresas en 3D con refuerzo interno para mayor resistencia.

3. Marco del lente

- Estructura frontal simple que sostiene la carcasa superior.
- Diseñada para comodidad, mantener la estética y facilitar la integración con la carcasa.
- Su forma permite una presentación natural similar a la de unos lentes tradicionales.

4. Soportes de exhibición

- Diseñados para mantener los lentes firmes y estables sobre el podio durante la exposición.
- Fueron impresos en dos piezas y posteriormente pegados en la base del stand.
- Construidos por extrusión y luego unidos en AutoCAD mediante la operación UNION.
- Impresos con infill del 20 % para obtener rigidez sin excesivo peso.