

**ÍNDICE**

## Introducción

1.1 Integrantes

1.2 Descripción breve

1.3 Problemas a resolver

1.4 Público beneficiado

1.5 Tecnología utilizada

## Hardware

2.1 Raspberry Pi 4

2.2 TFmini

2.3 Fuente Step-down

2.4 Baterías

2.5 Diseño esquemático

2.6 Diseño PCB

## Firmware

3.1 Detección por IA

3.2 Detección por láser

3.2.1 Libreria TFmini

3.2.2 Código de prueba TFmini

3.2.3 Control de PWM

## Diseño

4.1

## Redes sociales

5.1 Página web

5.2 Instagram

## Facturas

# Introducción

### 1.1 Integrantes

Castillo Ramiro

Pino Octavio

Quattrocchi Tiago

### 1.2 Descripción Breve

El objetivo de este proyecto es mantener la integridad de las personas ciegas o con discapacidad visual en entornos urbanos. Para ello, se desarrollará una riñonera, equipada con una cámara, un sistema de inteligencia artificial y láseres LIDAR que avisan al usuario sobre un obstáculo en frente, contando con alertas sonoras y vibratorias para cumplir este propósito. Con este proyecto, se busca facilitar el desplazamiento de personas con discapacidad visual, brindándoles seguridad en la vía pública.

### 1.3 Problemas a Resolver

Proteger su intimidad: Poder detectar personas que están en la misma habitación sin tener que depender que la persona avise de su llegada.

Obstáculos repentinos: Dificultad para detectar postes, árboles, postes, pozos y otros peligros inesperados.

Riesgo constante y dependencia de cuidadores: La falta de autonomía y la dependencia de terceros es uno de las problemáticas más importantes a resolver la idea es darles una herramienta de independencia e integración.

### 1.4 Público Beneficiado

El proyecto está destinado a personas con impedimentos visuales. En el mundo hay 2200 millones de personas con visibilidad reducida, 43,3 millones son no videntes. Esta es la demográfica que será beneficiada directamente por el proyecto, habilitando el tránsito por la vía pública de manera más segura y sin problemas que puedan llegar a afectar su integridad o la rehabilitación a la que están sometidos. Además, al ser un accesorio portable, compacto, difícil de perder u olvidar y puede servir para guardar bienes no interrumpe en las acciones que pueda realizar el usuario en su vida cotidiana.

Este proyecto también beneficiará indirectamente a instituciones privadas como seguros de salud y ART (Aseguradoras de Riesgo del Trabajo), ya que estos pueden ofrecerlo como una ayuda al grupo de personas antes descritas. También aplica para la vida cotidiana o la vida pública, ya que se reduce el riesgo de accidentes y la carga que supone a los familiares a la hora de ayudarlo.

### 1.5 Tecnología utilizada

# Hardware

### 2.1 Tecnología utilizada

### 2.2 Raspberry Pi 4

La Raspberry Pi 4 es una computadora de placa única (SBC) compacta y de bajo consumo que sirve como el "cerebro" de su proyecto. A diferencia de las computadoras de escritorio tradicionales, está diseñada para tareas especializadas y prototipado. Utiliza un procesador Broadcom BCM2711 de cuatro núcleos y una arquitectura ARM, lo que la hace ideal para aplicaciones de robótica y automatización. Su funcionamiento se basa en la ejecución de un sistema operativo, como Raspberry Pi OS, que permite programar y controlar los componentes conectados a sus pines de GPIO (General Purpose Input/Output).



### 2.3 TFmini

El TFmini es un sensor de distancia LiDAR (Light Detection and Ranging). Funciona emitiendo un pulso de láser infrarrojo y midiendo el tiempo que tarda la luz en rebotar en un objeto y regresar al sensor. Esta técnica, conocida como tiempo de vuelo (ToF), le permite calcular la distancia con precisión. Se comunica con la Raspberry Pi a través de interfaces como UART (puerto serie) o I2C, lo que le permite obtener lecturas de distancia de forma rápida y confiable.



### 2.4 Fuente Step-down

Una fuente de alimentación step-down (también conocida como convertidor buck) es un circuito electrónico que reduce un voltaje de entrada más alto a un voltaje de salida más bajo, manteniendo la eficiencia. Esto es crucial para convertir el voltaje de las baterías (7,4V) a un voltaje seguro y estable (como 5V) que la Raspberry Pi y otros componentes puedan usar sin dañarse.



### 2.5 Baterías

Las baterías son la fuente de energía portátil del proyecto. Generalmente se utilizan baterías de iones de litio (Li-ion) o polímero de litio (LiPo) debido a su alta densidad energética y peso ligero, lo que es ideal para aplicaciones móviles. El número de celdas en serie determina el voltaje total (por ejemplo, una batería 3S de LiPo tiene 3 celdas, cada una de ~3.7V, sumando un voltaje nominal de 11.1V).



### 2.6 Motores de vibración

Los motores de vibración que vamos a utilizar son motores de corriente continua (DC) conocidos como Eccentric Rotating Mass (ERM). Su principio de funcionamiento es muy sencillo pero efectivo:

* Eje y Motor: El motor tiene un eje que gira cuando se le aplica energía eléctrica.
* Masa Descentrada: Lo que crea la vibración es un pequeño peso o masa que está unido al eje, pero no en su centro. Este desplazamiento del centro de masa hace que la rotación sea desequilibrada.
* Generación de Vibración: A medida que el motor gira a alta velocidad, la masa descentrada genera una fuerza centrífuga que cambia de dirección constantemente. Este movimiento rápido y desequilibrado es lo que produce la vibración que se siente.

En muchos dispositivos como es el caso de blindassist que requieren retroalimentación háptica, se usan comúnmente dos motores ERM de diferentes tamaños. Esto permite generar distintos niveles de vibración, desde un temblor fuerte y notorio hasta una vibración suave y sutil, mejorando la experiencia del usuario.

### 2.7 Diseño esquemático

### 2.8 Diseño PCB

# Firmware

### 3.1 Detección por IA

### El sistema de visión artificial implementado en este proyecto se basa en el modelo de aprendizaje profundo YOLOv8n.pt para la identificación de objetos en tiempo real, utilizando la cámara de la Raspberry Pi como fuente de datos.La arquitectura del software se estructura en un sistema de Programación Orientada a Objetos (POO). Cada componente lógico (cámara, IA, voz, traductor) está encapsulado en una clase dedicada, lo que facilita el desarrollo, las pruebas y el mantenimiento del sistema de manera modular.

### 3.1.1.1 La lógica principal se centra en un bucle continuo que:

### Captura frames de video.

### Analiza cada frame en busca de objetos.

### Compara las detecciones con las del frame anterior.

### Emite una alerta de voz solo cuando se detecta un cambio en el entorno (aparición o desaparición de un objeto).

### 

### 3.1.1.2 Arquitectura del Código

### El programa se organiza en un conjunto de clases principales:

### Clase Camara: Gestiona el hardware de la cámara. Utiliza la librería picamera2 para inicializar y capturar un stream de video en vivo. A diferencia de soluciones de un solo fotograma, esta aproximación optimiza el rendimiento y la tasa de fotogramas por segundo (FPS) al mantener el stream de video abierto.

### Clase IA: Actúa como el motor de inteligencia artificial. Carga el modelo yolov8n.pt y es responsable de analizar los fotogramas capturados. Identifica objetos en una lista predefinida de clases (clases\_permitidas) y filtra las detecciones por un umbral de confianza (CONFIDENCE\_THRESHOLD), asegurando que solo los objetos relevantes y con alta certeza sean considerados.

### Clase Traductor: Encapsula el diccionario de traducciones. Su única responsabilidad es convertir los nombres de las clases de YOLO (en inglés, como 'person' o 'car') a sus equivalentes en español, permitiendo que el sistema de voz dé las alertas en el idioma del usuario.

### Clase Voz: Maneja la síntesis de voz. Para garantizar la fiabilidad del audio, utiliza un método robusto basado en la librería subprocess para ejecutar el comando del sistema espeak y vocalizar las alertas. Esta solución es inmune a los problemas de configuración de audio que se presentan a menudo con otras librerías.

### Clase Main: Es el orquestador principal del sistema. Inicializa todas las demás clases y gestiona el bucle de detección en tiempo real. Su lógica más avanzada reside en el método privado \_\_anunciar\_detecciones, que compara las detecciones actuales con las del último ciclo, emitiendo una alerta de voz únicamente cuando se detecta un cambio. Esto evita la repetición constante de las mismas alertas.

### El sistema se ejecuta continuamente en un bucle while hasta que el usuario lo interrumpe con Ctrl + C. En ese momento, un bloque finally asegura que el hardware de la cámara se detenga de manera segura para evitar fallos del sistema.

### 3.1.2 Componentes de la Implementación

### 3.1.2.1 Librerías Clave:

### ultralytics: Se utiliza para cargar y ejecutar el modelo de detección de objetos YOLOv8.

### picamera2: La interfaz oficial para las cámaras de la Raspberry Pi, proporcionando acceso a un flujo de video de alto rendimiento.

### cv2 (OpenCV): Se usa para el preprocesamiento de los fotogramas, como la conversión de color necesaria antes de pasarlos al modelo de IA.

### subprocess: La herramienta estándar de Python para ejecutar comandos del sistema, en este caso, el motor de síntesis de voz espeak.

### os y sys: Permiten la gestión de archivos y el control del sistema, por ejemplo, para verificar la existencia del modelo YOLO antes de su uso.

### 3.1.2.2 Manejo de Flujo de Datos:

### El flujo de información y la lógica de control se manejan principalmente en el método iniciar() de la clase Main:

### Captura: self.\_\_camara.capturar\_frame() obtiene una imagen del stream de la cámara.

### Análisis: El frame se pasa a self.\_\_ia.analizar\_imagen(frame), que devuelve un diccionario con los objetos detectados en el fotograma actual.

### Control de Anuncios: El diccionario se envía al método \_\_anunciar\_detecciones().

### Reproducción de Audio: Si hay cambios, la clase Voz es invocada a través de self.\_\_voz.hablar(final\_message).

### La arquitectura del sistema es muy escalable. Al estar modularizado, se podrían añadir fácilmente nuevas clases sin modificar la lógica central del proyecto.

### 3.2 Detección por láser

### 3.2.1 Libreria TFmini

Esta sección describe el funcionamiento del código implementado para la lectura simultánea de tres sensores LiDAR TFmini conectados a diferentes puertos UART de una Raspberry Pi 4.

El sistema permite:

* Recepción continua de datos desde los sensores.
* Procesamiento y almacenamiento de las distancias medidas.
* Ejecución en paralelo mediante hilos (multithreading).
* Disponibilidad centralizada de los valores para otras funciones del sistema.

Descripción General del Sistema:

El programa se estructura en cuatro bloques principales:

* Configuración global: Definición de librerías, estructuras de datos y variables compartidas.
* Módulo de adquisición de datos: Funciones de lectura desde los sensores a través de los puertos UART.
* Gestión de hilos: Creación de procesos concurrentes para la lectura simultánea.
* Bucle principal: Supervisión e impresión periódica de los valores capturados.

La arquitectura se basa en la comunicación serial estándar (UART) a una velocidad de 115200 baudios, con tramas de 9 bytes emitidas por cada TFmini.

3.2.1.1 Librerías:

* serial (pyserial): Permite abrir, configurar y gestionar la comunicación UART.
* time: Se utiliza para pausas entre lecturas y evitar sobrecarga del procesador.
* threading: Facilita la ejecución concurrente de múltiples lecturas de sensores.

3.2.1.2 Estructuras Globales:

distancias = {

"uart1": None,

"uart2": None,

"uart3": None,

}

Diccionario que almacena la última distancia válida obtenida de cada sensor. Inicialmente los valores son None y se actualizan en tiempo real. Facilita la consulta de datos desde cualquier parte del programa.

3.2.1.3 Función Genérica de Adquisición de Datos:

def getTFminiData(port, key\_name):

ser = serial.Serial(port, 115200, timeout=1)

...

Flujo de la función:

* Apertura del puerto serie con parámetros: Velocidad de 115200 baudios, timeout de 1 segundo.
* Bucle infinito de lectura: Verifica si hay al menos 9 bytes disponibles

(ser.in\_waiting > 8).

* Lee un paquete completo

(ser.read(9)).

* Validación de cabecera: Primeros dos bytes deben ser 0x59 0x59. Si no coincide, se descarta la trama.

Reconstrucción de la distancia:

Byte [2] = parte baja (Low).

Byte [3] = parte alta (High).

Cálculo: distance = low + (high << 8).

3.2.1.4 Almacenamiento de datos

La distancia calculada se guarda en el diccionario global bajo la clave correspondiente (uart1, uart2, uart3). Retardo de control time.sleep(0.01) para liberar recursos del procesador.

3.2.1.5 Funciones Específicas por Puerto UART

def getTFminiData\_uart1():

getTFminiData("/dev/serial0", "uart1")

Función getTFminiData\_uart1: Llama al puerto /dev/serial0.

Función getTFminiData\_uart2: Llama al puerto /dev/ttyUSB1.

Función getTFminiData\_uart3: Llama al puerto /dev/ttyUSB2.

Estas funciones permiten escalar el sistema sin modificar la lógica principal.

3.2.1.6 Gestión de Hilos y Ejecución Principal

if \_\_name\_\_ == '\_\_main\_\_':

...

Creación de hilos: Cada hilo ejecuta una de las funciones específicas (uart1, uart2, uart3). Se utilizan hilos daemon, lo que asegura que se cierren cuando termine el programa.

Ejecución concurrente: Los tres hilos inician en paralelo con .start(). Se garantiza la lectura simultánea de los sensores.

Bucle principal de supervisión: Cada segundo imprime las distancias almacenadas en distancias y permite monitorear en consola el funcionamiento del sistema. Interrupción del usuario: Al presionar Ctrl + C, se captura la excepción KeyboardInterrupt. Se muestra un mensaje de salida segura.

### 

### 3.2.2 Control de PWM

Este programa implementa un sistema de control de motores mediante señales PWM (Pulse Width Modulation), en base a las lecturas de distancia obtenidas de tres sensores LiDAR TFmini conectados por UART.

El sistema permite:

Captura simultánea de las distancias desde tres sensores, un control dinámico del ciclo de trabajo (duty cycle) de dos salidas PWM en la Raspberry Pi 4, estrategias de control diferenciadas para cada sensor y lajecución en paralelo mediante hilos de adquisición y control.

Este diseño es útil para aplicaciones de robótica móvil, sistemas de asistencia o control adaptativo, donde las distancias detectadas influyen directamente en la potencia de motores o actuadores.

3.2.2.1Descripción General del Sistema

La lógica del sistema se organiza en tres niveles funcionales:

* Adquisición de datos: Módulo de lectura UART importado desde TFmini3.py, que actualiza un diccionario global de distancias.
* Control de motores por PWM: Tres funciones que adaptan la señal PWM en función de las distancias medidas.
* Gestión de hilos y ciclo principal: Lanzamiento en paralelo de procesos de lectura y control, garantizando funcionamiento continuo y autónomo.
* El hardware utilizado es la Raspberry Pi 4, con soporte nativo de PWM en los pines GPIO18 y GPIO13.

3.2.2.2 Librerías Utilizadas

* time: Gestión de retardos y temporizaciones.
* threading: Permite ejecutar múltiples tareas de forma concurrente (lectura + control).
* RPi.GPIO: Biblioteca oficial para controlar pines GPIO de Raspberry Pi, incluyendo PWM.
* TFmini3 (módulo propio):
  + distancias: diccionario global con los valores de los tres sensores.
  + getTFminiData\_uart1, getTFminiData\_uart2, getTFminiData\_uart3: funciones de lectura UART en hilos independientes.

3.2.2.3 Configuración de Pines y PWM

MOTOR\_PIN1 = 18 # PWM 1

MOTOR\_PIN2 = 13 # PWM 2

GPIO18 y GPIO13 son pines con soporte PWM por hardware en Raspberry Pi 4.

GPIO.setmode(GPIO.BCM)

GPIO.setup(MOTOR\_PIN1, GPIO.OUT)

GPIO.setup(MOTOR\_PIN2, GPIO.OUT)

Se configura el modo BCM (numeración lógica de pines). Ambos pines se inicializan como salidas digitales.

pwm1 = GPIO.PWM(MOTOR\_PIN1, 100)

pwm2 = GPIO.PWM(MOTOR\_PIN2, 100)

pwm1.start(30)

pwm2.start(30)

Frecuencia de PWM: 100 Hz.

Duty cycle inicial: 30% → corresponde a potencia media en motores.

3.2.2.4 Control de motores

**Motor 1:**

* Controla el motor conectado al GPIO18 (PWM1) según el valor de UART1.
* Si la distancia < 100 cm, el duty cycle aumenta proporcionalmente:
* duty = 30 + (100 - d) limitado a 100%.
* Si la distancia ≥ 100 cm, mantiene duty en 30%.
* Mensaje en consola con valores de distancia y duty aplicado.
* Refresca cada 0.1 s.
* Función: Aumentar la velocidad del motor al detectar un obstáculo cercano en UART1.

**Motor 2:**

* Controla el motor conectado al GPIO13 (PWM2) según el valor de UART2.
* Mismo esquema de control que UART1, pero aplicado al segundo canal PWM.
* Distancia menor a 100 cm → incremento proporcional del duty.
* Distancia mayor a 100 cm → duty constante en 30%.
* Refresca cada 0.1 s.
* Función: Motor secundario regulado por la distancia detectada en UART2.

**Motor 3:**

* Controla de manera intercalada ambos motores en función de UART3.
* Si la distancia < 100 cm:
* PWM alterna entre GPIO18 y GPIO13 cada 0.3 s.
* Uno de los motores se activa con 80% duty mientras el otro se apaga.
* Si la distancia ≥ 100 cm:
* Ambos motores se mantienen en 30% duty.
* Función: Generar un patrón de parpadeo/intermitencia entre los dos motores ante detección de obstáculos cercanos por UART3.

3.2.2.5 Gestión de Hilos y Ejecución Principal

if \_\_name\_\_ == "\_\_main\_\_":

…

Proceso de ejecución: Hilos de lectura UART t1, t2, t3 ejecutan en paralelo las funciones getTFminiData\_uart1/2/3.

Permiten actualización continua del diccionario distancias.

Hilos de control PWM: c1, c2, c3 ejecutan en paralelo las funciones control\_motor\_uart1/2/3.

Cada hilo ajusta el duty cycle de manera independiente según la lógica descrita.

Bucle principal: Programa se mantiene activo con un while True y time.sleep(1). Los hilos daemon garantizan que los procesos se cierren al terminar.

Gestion de interrupción y limpieza:

KeyboardInterrupt: permite salir con Ctrl + C.

finally: se detienen los PWM (pwm1.stop(), pwm2.stop()) y se libera la configuración GPIO (GPIO.cleanup()).

# Diseño

### 4.1 Prototipo 3D

* Materiales Utilizados / Programas

Filamento de impresión 3D PLA.

Tornillos ⅛

AutoCAD

* Dimensiones:

### Idea general del diseño

El diseño corresponde a un prototipo de carcasa destinado a integrar de forma eficiente los sensores y componentes del proyecto. Su geometría incluye un frente inclinado en forma triangular, que permite optimizar los ángulos de medición de los LIDAR y ampliar su campo de detección en entornos urbanos. También incorpora un orificio frontal para la cámara y tres ranuras específicas para los LIDAR, lo que asegura una orientación precisa y funcional.

El sistema de cierre se resuelve mediante anclajes en el borde del cuerpo y orejas externas para tornillos, lo que garantiza un ajuste firme y facilita el mantenimiento. A nivel general, el prototipo busca maximizar el aprovechamiento del espacio interno, mejorar la disposición de los sensores y proteger los componentes electrónicos.

Es importante remarcar que se trata de un prototipo en evolución, sujeto a mejoras tanto en la disposición interna como en la ergonomía y robustez del ensamble.

Comparaciones entre versiones:

1. Espacio interno  
   * El primer prototipo no contaba con el espacio suficiente para albergar todos los componentes electrónicos.
   * En el segundo, se corrigieron dimensiones y proporciones, logrando una organización más adecuada del volumen interno.
2. Ranuras para sensores LIDAR  
   * En el primer modelo, las ranuras eran más genéricas y no contemplaban bien la orientación de los sensores.
   * El segundo prototipo introdujo tres ranuras específicas para los LIDAR, con un frente inclinado que mejora el ángulo de detección.
3. Relieves y fijaciones internas  
   * El primer y segundo prototipo carecían de soporte interno para los componentes.
   * En el tercer prototipo, que exteriormente mantiene la misma geometría, se añadieron relieves interiores para que los LIDAR queden bien posicionados y se incorporaron finalmente los anclajes para fijar las plaquetas electrónicas, permitiendo un montaje más firme y seguro.
4. Sistema de cierre  
   * Desde el segundo prototipo ya se contemplaban orejas externas para tornillería, pero en el prototipos siguiente se mejoró la definición de los encastres en el borde del cuerpo, logrando un cierre más sólido.

## Conclusión

La serie de prototipos muestra una mejora constante en la funcionalidad y organización interna de la carcasa.

* El primer prototipo permitió validar dimensiones generales.
* El segundo resolvió la correcta orientación de los LIDAR con el frente inclinado y el aumento de espacio.
* El tercero introdujo finalmente los relieves interiores y fijaciones internas, mejorando la integración de los componentes.

Este proceso iterativo confirma que el diseño está en el camino correcto, pero aún con margen para seguir perfeccionando aspectos de robustez estructural, ensamblaje y usabilidad.

# Redes Sociales

### 5.1 Página web

### 5.2 Instagram

# Facturas