Posgrado en Robótica e Inteligencia Artificial

Proyecto final

RiveroGlasses, una solución IOT con visión artificial para la asistencia a ciegos

Autor : Néstor Sequeira

Supervisor :
Vinícius Menezes de Oliveira

5 de septiembre de 2021

Agradecimientos

En primer lugar quiero agradecer a UTEC por permitirme iniciar en este mundo que me apasiona, me hace perder el sueño sin penas, uno se adentra en el estudio y el tiempo transcurre de una manera diferente, desaparece. A los docentes que me han enseñado tanto en tan poco tiempo, permitiéndome entender cosas que al principio pensaba que eran mágicas, soportando mis cuestionamientos y contestando a mis inquietudes.
A la Armada Nacional en especial al SELAR, que me brindo su apoyo incondicional, permitiendo que le dedique tiempo al estudio, y siendo mi campo de entrenamiento para la aplicación de lo aprendido en el PRIA.
A mis compañero de generación que varios ya los puedo considerados Amigos.
A quienes me ayudaron desinteresadamente a escribir esta tesis que tanto me cuesta.
A Nicolás quien me invito a participar de este proyecto que permite ayudar a las personas, a Federico quien me mostró este mudo tan diferente y desconocido.
A mis Hijas Clara y Renata que me regalaron el tiempo de ellas para que pueda especializarme en esto que tanto me gusta y para finalizar a mi esposa Isolda quien tuvo que hacer sus tareas más las mías para que yo pueda hacer esto que tanto me gusta. Solo me queda decir ¡Muchas gracias!

Resumen

Existen en el mundo diversos dispositivos y tecnologías que asisten a las personas ciegas a tener mayor autonomía, dichos dispositivos tienen altos costos económicos, además de que se comercializan principalmente en países económicamente desarrollados, lo que genera una situación social y económica difícil para las personas con ceguera, pues el acceso a dicha tecnología requiere de recursos que no están al alcance de todos. A su vez, si se tiene en cuenta que gran parte de la población ciega o de baja visión se concentra en países económicamente subdesarrollados, el acceso es aún más difícil, no permitiendo una mejora en el bienestar de salud y social de la población afectada por la ceguera. Este proyecto consiste en el diseño de un prototipo que interactúa con RiveroPi; mediante el algoritmo YoloV5 se detectará objetos y se medirá su distancia haciendo uso de un dispositivo Time of Fly (ToF) VL531X; para luego informar al usuario que objetos son detectados, su distancia y ubicación mediante el uso de audio. Se generan tres modos de operación, el primero es especializado en caminar; se dará la información al usuario de que objetos se encuentran a menos de dos metros de distancia, esto ayudará a tomar decisiones de cual movimiento o dirección es mejor. El segundo modo, tiene como objetivo la búsqueda de un objeto especificado por el usuario, se solicita al asistente que comience la búsqueda, al detectar dicho objeto informará la ubicación y la distancia del mismo. El tercer modo, tiene como función la búsqueda de objetos que hay en una habitación, el sistema saca una foto, hará uso del algoritmo de reconocimiento y de esta manera le informará al usuario la ubicación de los objetos que reconoce.

Palabras clave: Raspberry Pi, ESP32, Bluetooth, Opencv, Yolo V5, lentes para ciegos, deficiencia visual, ciegos

Índice

Índice de figuras

Índice de tablas

1. Introducción

Según la OMS, en el mundo existen más de 2200 millones de personas con deficiencia visual o ceguera [OMS, 2019]. La mitad de estos casos pudieron haber sido prevenidos, mientras que otros no; más de 39 millones son ciegas totales. En particular esto se da en los países de ingresos bajos y medios. En Uruguay, según datos del censo 2011 [REDACCIÓN180, 2012], existen más de 300 mil personas con baja visión, de las cuales más de 4 mil son ciegas totales. En Brasil según datos del IBGE más de 1.5 millones de personas padecen de ceguera total [BR, 2010].

Situaciones que parecen triviales para cualquier persona vidente, puede generar estrés y depresión en ciegos. Identificar billetes, el color de la ropa para combinar, cocinar o hacer las compras. Todo genera dependencia.

Para un ciego que desea tener una vida independiente, social y completa, cada día se transforma en un periplo con muchas dificultades. Es muy complejo tomarse un ómnibus, esquivar salientes, carteles y pozos en las veredas, cruzar en un semáforo, identificar un local específico en un shopping o un andén en la terminal de autobuses.

A partir de todo esto, surge la pregúnta sobre cómo podemos mejorar la independencia de los ciegos en nuestra región. Esta independencia viene dada por dos factores clave: movilidad y visión.

La aparición de soluciones y dispositivos tecnológicos han reducido la brecha enormemente. Entre estos encontramos teléfonos inteligentes con comandos de voz (Siri, Alexa, Android), dispositivos dedicados (Orcam, My Eyes, EyeSynth), sistemas de navegación basados en audio (Wayfindr). Sin embargo, estos dispositivos son difíciles de alcanzar económicamente y no brindan información localizada y personalizada. Tampoco se encuentra una solución completa y adaptable. Por ejemplo, Wayfindr es un sistema para navegación pero no incluye funcionalidades de visión artificial como Orcam. Este proyecto tiene como objetivo diseñar un sistema IOT dedicado a las personas ciegas, que sea extensible, integral y de bajo costo relativo.

2. Objetivos y metas

2.1. Objetivo general

Este proyecto se enmarca en las actividades del Posgrado de Inteligencia Artificial y Robótica de UTEC y FURG; tiene como objetivo general diseñar un sistema que ayude a mejorar la independencia de los ciegos en nuestra región.

2.2. Objetivo especifico

El objetivo específico de este proyecto será realizar un prototipo que interactúe con la plataforma RiveroPi desarrollado en el proyecto de Nicolas Acerenza, proporcionando al usuario la dirección y la distancia de los objetos u obstáculos que se encuentran delante.

2.3. Metas

- Diseñar un soporte con forma de lentes que permitan portar una cámara, un láser y los auriculares para la interacción con el usuario.
- Diseñar un sistema que permita el control de la dirección del láser que será usado para medir la distancia a objetos.
- Interacción manual del sistema por medio de una interfaz gráfica.
- Detección de objetos mediante el uso de algoritmos de visión artificial. Medir distancia a objetos detectados.
- Informar la ubicación del objeto detectado mediante audio al usuario dando la dirección y la distancia al mismo
- Integración con la plataforma RiveroPi

3. Revisión Bibliográfica

3.1. Dispositivos Relacionados

ORCAM My eye

"Es un dispositivo revolucionario activado por voz que se acopla prácticamente a cualquier gafa. ¡Es capaz de leer instantáneamente el texto de un libro, la pantalla de un teléfono inteligente o cualquier otra superficie, reconocer rostros, ayudarle a comprar por su cuenta, trabajar más eficientemente y vivir una vida más independiente! OrCam MyEye transmite información visual de forma audible, en tiempo real y sin conexión a Wifi."[OrCam, 2021]

IrisVision

" A diferencia de OrCam, IrisVision mejora su visión útil restante, para que pueda ver las cosas de forma clara e independiente."[IrisVision, 2021]

EyeSynth
"Eyesynth es un sistema de comprensión visual para invidentes. Se compone de unas gafas que registran el espacio que nos rodea en 3 dimensiones. Un micro-ordenador procesa la información y la convierte en audio comprensible para el invidente. Nosotros lo llamamos "experiencia de sentido aumentado."[eyesynth, 2021]

3.2. Revisión bibliográfica

Este proyecto pretende dar asistencia a las personas ciegas para detectar objetos que se encuentran a su alrededor o obstaculizan su andar, para realizar esta tarea es necesario generar un prototipo que sea capaz de detectar objetos en tiempo real. Dentro de las asignaturas que fueron impartidas en el PRIA se destacan dos de ellas en el área de detección de objetos, Visión computacional e Inteligencia Artificial. En visión computacional se hace uso de la biblioteca para python[**?**] denominada OpenCV[team, 2021], con la que se aprenden diferentes métodos que permiten la detección de objetos, e identificación de patrones de los objetos con lo que se puede mejorar. En Inteligencia Artificial se brindan las herramientas necesarias para poder entender cómo están compuestas las redes neuronales, su funcionamiento y cómo se realiza el aprendizaje de máquina. Es importante destacar el estudio de diferentes tipos de redes, entre ellas las convulsiones y el acercamiento a diferentes arquitecturas de redes que han mejorado la detección de objetos en tiempo real.

3.2.1. Introducción a la visión computacional

El cerebro humano interpreta la información que le llega a través de la vista, identificando objetos y ubicando su posición, esto permite a las personas realizar las actividades cotidianas. La visión computacional pretende emular las acciones antes mencionadas por medio del procesamiento de imágenes y vídeos mediante el uso de tecnología. Esta tarea no es para nada trivial y tampoco nueva, pues desde los años 60 se viene evolucionando en esta área de investigación con pequeñas mejoras[**?**], pero recién en los últimos años se han logrado grandes avances. Esto último que se menciona se debe a varios factores, dentro de los cuales se pueden citar los avances teóricos en los en los algoritmos, mejoras en en la tecnología de procesamiento de datos, innovaciones en inteligencia artificial, deep learning y la gran cantidad de imágenes que se están generando diariamente, solamente en la red social Instagram en el año 2019 se suben aproximadamente 95 millones de fotos por día.[Brandwatch, 2019] Dentro de la visión computacional debemos de sepa-

rar dos problemas, el procesamiento de imágenes y la visión computacional. El procesamiento de imágenes se encarga de mejorar una imagen para posteriormente ser interpretada por un ser humano. Dentro de las acciones que se pueden realizar se puede encontrar con la remoción de defectos en la imagen, mejoras en los desenfoques, modificación de colores, alteración de imágenes, etc. En cambio la visión computacional debe de extraer características de una imagen para su descripción e interpretación por la computadora, como por ejemplo la detección y localización de objetos en una imagen, construir un representación tridimensional de un objeto, descomposición de una imagen u objeto en diferentes partes, etc.[Sucar and Gómez, 2011]

3.2.2. Redes neuronales artificiales

En la investigación del estado del arte de esta tecnología se evidencia que no existe una única definición para esta temática, tal cual como expresa Simon Haykin (1999) en su libro; *"Una red neuronal es un procesador distribuido masivamente en paralelo compuesto por una unidad de procesamiento simple, que tiene una capacidad natural para almacenar conocimientos de manera experimental y hacerlos disponibles para su uso. Se parece al cerebro en dos aspectos:*

1. *El conocimiento es adquirido por la red de su entorno a través de un proceso de aprendizaje.*

2. *Las fuerzas de conexión entre neuronas, conocidas como pesos sinápticos, se utilizan para almacenar el conocimiento adquirido".*

Otras formas de definirlo pueden ser las expresadas por José Hilera y Víctor Martínez en su libro (Rede Neuronales Artificiales. Fundamentos, modelos y aplicaciones)[Hilera González et al., 2000] *"Las redes neuronales artificiales son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico"* o *"Un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles", estas definiciones tiene mucho en común, pero para entenderlas tendremos que definir varios conceptos.*

3.2.3. Red neuronal biológica

Las redes neuronales artificiales se inspiran en las redes neuronales biológicas. Estas redes están compuestas por un conjunto de células especializadas

en la recepción, integración y transmisión de información llamadas neuronas. Estas células se encuentran altamente interconectadas entre sí, su cantidad estimada en el sistema nervioso central ronda las 10^{11} neuronas y sus interconexiones andan en el orden de 10^{15} [Hilera González et al., 2000]. Como se puede observar en la Figura 1 la neurona biológica se compone básicamente por:

- El Soma o cuerpo de la neurona en donde está contenido el núcleo, su forma puede ser muy variable y desde esta parten las dendritas y el axón.
- Las dendritas tienen forma de árbol de donde proviene su nombre, son las encargadas de la recepción de la información a través de las espinas dendríticas, siendo el elemento post-sináptico de la sinapsis .
- El axón es una prolongación delgada del soma, puede tener una longitud variable de menos de 1 mm a varios metros, es el responsable de transportar y transmitir la información, que fue procesada en la neurona.
- La Sinapsis es una estructura celular donde se intercambia información entre dos neuronas. Existen dos tipos de sinapsis la química y las eléctricas. [de Vigo, 2018]

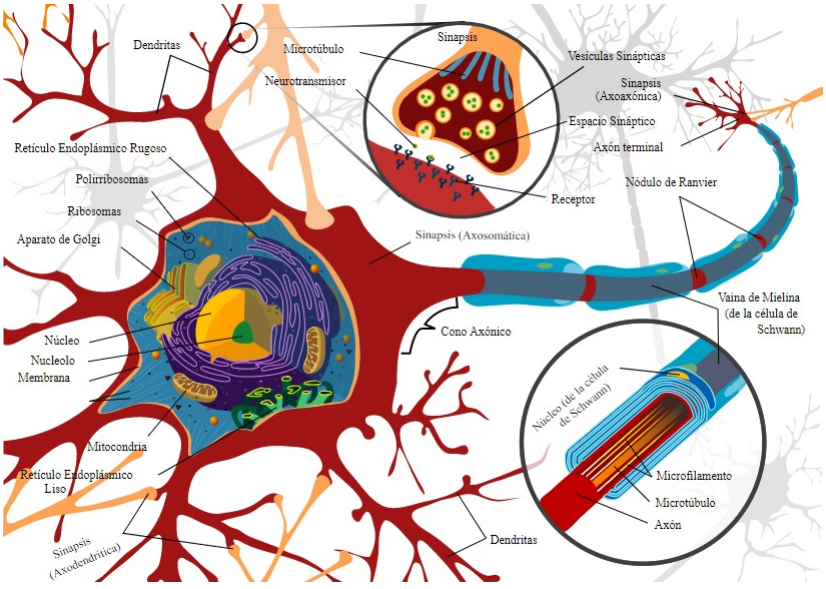

Figura 1: Composición de una Neurona Biológica[LadyofHats, 2007]

3.2.4. Neurona Artificial

Es la unidad de procesamiento fundamental de una red neuronal artificial, se puede dividir en cuatro elementos que podemos observar en la Figura 2:

- Las sinapsis o conexiones son la entrada a la neurona, se conectan a la información de entrada de la red neuronal o a la salida de otra neurona, cada sinapsis tiene un peso que la pondera y multiplica el valor que le llega a través de ella.
- Este peso puede ser positivo o negativo. Un sumador que se encarga de sumar todos los resultados de cada sinapsis multiplicada por cada peso ponderador.
- El Bias tiene como función generar un offset positivo o negativo antes del ingreso a la función de activación. La función de activación limita la amplitud de la salida de la neurona, esto genera que la salida quede contenida dentro de ciertos límites en su amplitud.

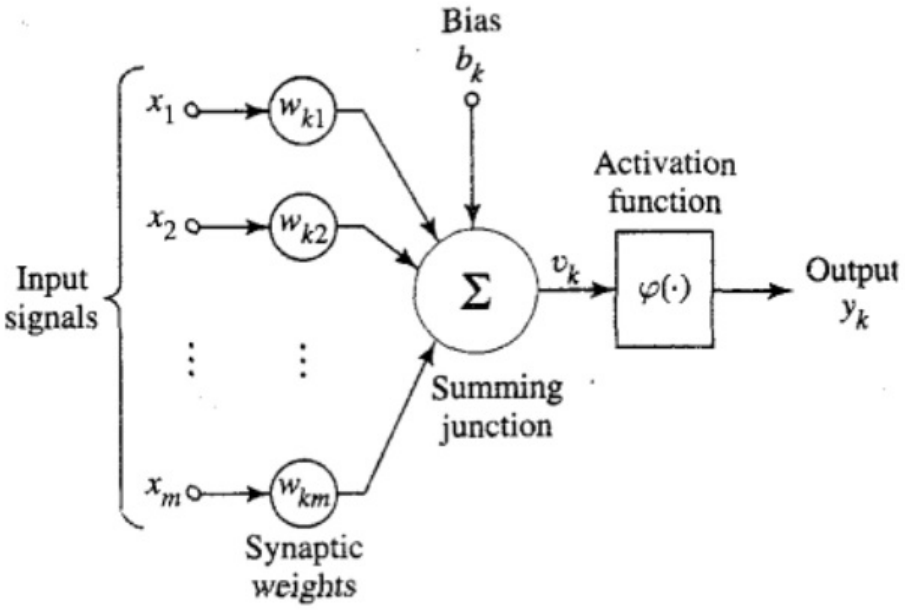

Figura 2: Modelo no linear de una neurona [Haykin and Network, 2004]

3.2.5. Definición de aprendizaje de máquina

Hay múltiples definiciones de aprendizaje, no obstante ello, para este proyecto se focaliza en la definición de aprendizaje de máquina dada por Mendel

y Maclaren (1970): *"El aprendizaje es un proceso por el cual lo parámetros libres de una red neural se adaptan a través de un proceso de estimulación por parte del entorno en el que está inmersa la red. El tipo de aprendizaje está determinado por la forma en que se producen los cambios de parámetros."* Se debe puntualizar de esta definición, que la red es estimulada por el ambiente, los cambios en los parámetros son el resultado de esos estímulos y la red responde de manera diferente a los estímulos debido a los cambios ocurridos en su estructura interna, estas reglas definen el algoritmo de aprendizaje[Haykin and Network, 2004]

3.2.6. Modelo de una Red Neuronal Perceptron

La red neuronal perceptrón es denominada por Frank Rosenblant para relacionar su modelo de redes neuronales a la resolución de problemas de percepción.
En 1943 McCulloch y Pitts desarrollan un modelo bioinspirado en el cerebro humano que es capaz de resolver problemas de una manera alternativa usando dispositivos formados por una red de unidades interconectadas a las que llaman neuronas. Desarrollan las siguientes suposiciones para el funcionamiento de una red neuronal: las salidas de las neuronas son binarias, tienen un umbral de activación, se deben de activar un número determinado de sinapsis para que se realice el cambio de estado, la demora que experimentan las señales de activación en su propagación se produce en las sinapsis, la actividad de una sinapsis inhibidora evita la excitación de la neurona y estructura de la red de interconexión no cambia con el tiempo.[McCulloch and Pitts, 1943]
El modelo antes definido se denomina neuronas binarias con umbral [binary threshold neurons] o LTU [linear threshold units], para el cual no es propuesto un algoritmo de entrenamiento, siendo necesario ajustar los pesos sinápticos manualmente. El funcionamiento de esta neuronas artificiales se realiza haciendo la suma ponderada de las sinapsis de entrada (z) multiplicada por su peso sináptico, $z = \sum_i \omega_i x_i$ si esta suma supera el umbral la salida es igual a 1 de lo contrario la salida es igual a 0.
En 1947 Alan Turing publica una charla para la London Mathematical Society denominado "Intelligent Machinery" donde considera el entrenamiento automático de las redes neuronales, conectadas inicialmente de forma aleatoria.[Turing, 1948]
En los años siguientes varios investigadores siguen trabajando en esta área desarrollando algoritmos de aprendizaje automático, pero el más destacado es Frank Rosenblatt quien desarrolla el algoritmo de aprendizaje del perceptrón. Se inspira en el modelo de aprendizaje hebbiano propuesto por Donald Hebb en 1949 y su mecanismo de aprendizaje es muy similar al que desa-

rrolla Clark y Farley con la diferencia de que logra generalizarlo para redes multicapas denominado back-propagating error correction. [Berzal, 2019]

3.2.7. Batch Normalization

La velocidad de aprendizaje en las diferentes capas de una red neuronal profunda es diferente, por lo general las capas de más cercanas a la salida de la red aprenderán más rápido que las capas más cercanas a la entrada, ya que el gradiente del error de aprendizaje irá disminuyendo, llegando a ser insignificante en las primeras capas. Este problema es llamado gradiente evanescente [vanishing gradient], ya que el gradiente del error de las capas anteriores se obtiene como producto de los términos correspondientes a todas las capas siguientes. Para solucionar este problema, todas las capas deben aprender más o menos a la misma velocidad, generando gradientes del error balanceados y así su producto sea, aproximadamente, de la misma magnitud. Esto es lo que hace, la normalización por lotes [batch normalization]. [Berzal, 2019]

3.2.8. Convolutional Neural Network

Este tipo de red neuronal ha tenido gran éxito por su gran capacidad de resolver problemas de visión artificial. Se basan en el uso de convoluciones. En los años 70 Hubel y Wiesel generaron su modelo ice cube model, este planteaba la posible organización del córtex visual. Unos años más tarde Kuhiniko Fukushima, propuso el cognitrón [Fukushima, 1975] y el neocognitrón [Fukushima and Miyake, 1982], modelos de red neuronal artificial bioinspirados.
El modelo está basado en una red neuronal compuesta por una retina, que sirve de capa de entrada y una jerarquía de módulos o niveles. Cada módulo o nivel está compuesto de dos capas: una de células simples [S-cells: simple cells] y otra de células complejas [C-cells: complex cells]. Este modelo es muy similar a las de las redes convolutivas actuales: las células S corresponden a las capas convolutivas, y células C son similares a las capas de pooling. Este modelo carece de un algoritmo de entrenamiento debiéndose diseñar manualmente a medida. [Berzal, 2019]

La convolución Es una operación matemática que se realiza sobre dos funciones, produciendo otra función que se interpreta como un filtro de una de las funciones originales. Se define matemáticamente como la integral del

producto de dos funciones después de que una de ellas se refleje y se desplace

$$(f \star g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau \tag{1}$$

Si se utilizan señales discretas se puede ver como

$$(f \star g)[n] = \sum_{-\infty}^{\infty} f[m]g[n-m] = \sum_{-\infty}^{\infty} f[n-m]g[m] \tag{2}$$

Si $x[n]$es la señal que queremos procesar y la otra función $h[n]$ es el filtro que queremos procesar finito con un dominio 0,1,...,k-1, la operación de convolución deberá realizarse para cada valor de la señal, k multiplicaciones y k-1 sumas.

$$(x \star h)[n] = \sum_{k=0}^{k-1} h[k]x[n-k] \tag{3}$$

Esto es generalizable para un caso multidimensional. Si tenemos señales discretas le aplicaremos un filtro k1 x k2 la expresión de la convolución es

$$(x \star h)[n1, n2] = \sum_{k1=0}^{k1-1} \sum_{k2=0}^{k2-1} h[k1, k2]x[n1-k1, n2-k2] \tag{4}$$

En procesamiento de imagen las variables [n1,n2] se sustituyen por las coordenadas de los píxeles de la imagen [x,y] y el signo negativo se sustituye por un signo positivo y se denomina correlación cruzada.

$$(x \star h)[n1, n2] = \sum_{k1=0}^{k1-1} \sum_{k2=0}^{k2-1} h[k1, k2]x[n1+k1, n2+k2] \tag{5}$$

Es interesante interpretar esto gráficamente, si se supone que se tiene una imagen en blanco y negro de un tamaño 7x7 píxeles representado en forma de matriz.

$$x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Si a esta imagen se le aplica un filtro o kernel que se define como

$$h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Para realizar la convolución se va desplazando la matriz h por cada posición de la matriz x, y se sitúa la matriz h para calcular la posición $x[0, 0]$.

$$x = \begin{bmatrix} 0 & 0 & 0 & & \\ 0 & 0 & 0 & & \\ 0 & 0 & 1 & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

La posición filtrada es
$x[1, 1] = 0x0+(-1x0)+0x0+(-1x0)+(4x0)+(-1x0)+0x0+(-1x0)+0x1 = 0$

Ahora si se calcula la posición $x[3, 2]$

$$x = \begin{bmatrix} & & & & \\ & & & & \\ & 0 & 0 & 0 & \\ & 1 & 1 & 1 & \\ & 1 & 1 & 1 & \\ & & & & \\ & & & & \end{bmatrix}$$

La posición filtrada será
$x[3, 2] = 0x0+(-1x0)+0x0+(-1x1)+(4x1)+(-1x1)+0x1+(-1x1)+0x1 = 1$
Si se realiza la convolución para todas las posiciones se obtiene la siguiente matriz

$$(x \star h) = \begin{bmatrix} 0 & -1 & -1 & -1 & 0 \\ -1 & 2 & 1 & 2 & -1 \\ -1 & 1 & 0 & 1 & -1 \\ -1 & 2 & 1 & 2 & -1 \\ 0 & -1 & -1 & -1 & 0 \end{bmatrix}$$

Si se observa, mediante la convolución se logra identificar las esquinas y las fronteras del objeto que hay en la imagen original. Si se modifica el kernel se puede obtener diferentes filtros que resaltan características importantes en la imagen.[Berzal, 2019]

Capas Convolucionales Se caracteriza por incorporar en su arquitectura dos importantes características, el uso de conexiones locales que restringen la conectividad de la red y pesos compartidos por varias neuronas [weight sharing], lo que permite disminuir el número de pesos que se deben de ajustar facilitando el entrenamiento. En una capa convolutiva se incluyen distintos tipos de detectores de características, estos se replican en toda la capa, para poder detectar estas propiedades en diferentes partes de la señal de la entrada. Cada tipo de detector generará un mapa diferente, dada una señal de entrada por ejemplo una imagen, su salida se forma por K imágenes que representan las K características buscadas por los detectores en la capa de convolución.[Berzal, 2019]

Capas de Pooling Su función es reducir el tamaño de la entrada para que las capas posteriores puedan trabajar con datos reducidos. La reducción puede ser espacial [spatial pooling] o de profundidad [cross-channel pooling]. La versión más usada es la de combinar una serie de píxeles de entrada utilizando la función máximo, se denomina max pooling. [Berzal, 2019] Podemos observar la siguiente matriz un ejemplo, si a la matriz 4x4 se le quiere realizar max pooling 2x2 el resultado del max pooling es:

$$\begin{bmatrix} 2 & 3 & 5 & 10 \\ 6 & 1 & 1 & 2 \\ 3 & 9 & 3 & 8 \\ 5 & 6 & 4 & 11 \end{bmatrix} = \begin{bmatrix} 6 & 10 \\ 9 & 11 \end{bmatrix}$$

DenseNet Las redes convolucionales pueden ser más precisas y eficientes de entrenar si contienen conexiones directas entre capas cercanas a la entrada y aquellas cercanas a la salida. Para esto se desarrolla la Red convolucional densa (DenseNet), que conecta cada capa con todas las demás en una forma de retroalimentación. Las redes convolucionales tradicionales con L capas tienen L conexiones entre cada capa y su capa posterior, la DenseNet tiene conexiones directas $L(L+1)/2$. Para cada capa, los mapas de características de todas las capas anteriores se utilizan como entradas, y sus propios mapas de características se utilizan como entradas en todas las capas posteriores.[Huang et al., 2017]

CSPNet [Cross Stage Partial Network] Esta arquitectura logra una mejoras en el gradiente, reduciendo la cantidad de cálculo para realizar el proceso. Este objetivo se logra dividiendo el mapa de características de la capa base en dos partes y luego fusionándose a través de una jerarquía propuesta de etapas cruzadas. El flujo de gradiente se debe propagar a través de

diferentes rutas de red dividiéndose. CSPNet puede reducir en gran medida la cantidad de cálculo y mejorar la velocidad de inferencia y la precisión. [Wang et al.,]

3.2.9. Detección de objetos

R-CNN [Region Based Convolutional Neural Networks] Se descompone el problema en tres etapas, primero se extraen posibles objetos utilizando un método de proposición de regiones, se generan 2000 regiones diferentes en las que exista una alta probabilidad de que un objeto se encuentre presente. Luego, para cada una de estas 2000 regiones propuestas, se extraen características utilizando una red convolucional entrenada con el dataset ImageNet, con lo que se intenta clasificar las imágenes. Finalizando se clasifica cada región utilizando LSVMs [Linear Support Vector Machines] a partir de las características extraídas se emplea un algoritmo de regresión para intentar delimitar mejor las fronteras del objeto en la imagen [bounding box regressor]. Si se detectan varios objetos solapados se eliminan los de menor probabilidad.[Berzal, 2019]

Fast R-CNN [Fast Region Based Convolutional Neural Networks] Sa crea una red convolutiva que sustituye el clasificador SVM. La red se aplica sobre la imagen completa y haciendo uso de un mecanismo de pooling RoI [region of interest pooling] se selecciona las regiones de interés y luego se utiliza para delimitar los límites del objeto identificado en esa región.[Berzal, 2019]

Fastest R-CNN Se elimina el algoritmo de búsqueda selectiva que genera las regiones de interés en las Fast R-CNN y lo sustituye por una red de proposición de regiones RPN [region proposal network], la que genera las propuestas de regiones y los límites de los objetos, para luego continuar con el proceso como una Fast R-CNN. [Ren et al., 2015]

Yolo [You Only Look Once] Existen 5 versiones de YOLO publicadas. Las mismas, presentan una mejora, lo que evidencia la investigación y la inclusión de ideas más innovadoras del estado del arte, tal cual lo menciona Joseph Redmon en [Redmon and Farhadi, 2018]. La versión que se usa para la detección de objetos en este proyecto es YoloV5. Yolo plantea una manera nueva para la detección de objetos como un único problema de regresión, directamente desde la imagen. Para esto una sola red convolucional predice simultáneamente múltiples cajas delimitadoras y probabilidades de clase para esas cajas. Su estructura se observa en la Figura 3.

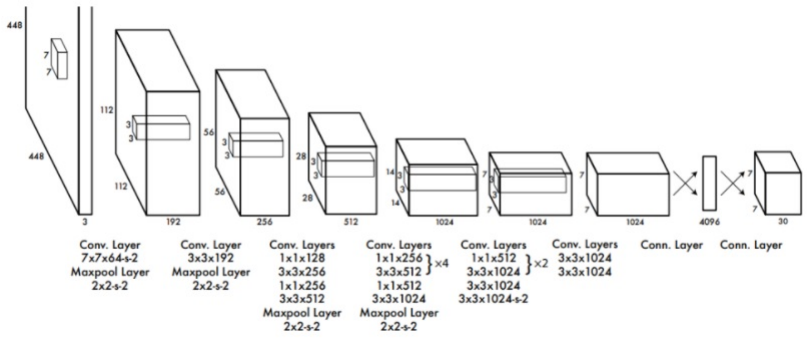

Figura 3: Estructura del algoritmo YOLO con 24 capas convolucionales seguidas por dos capas totalmente conectadas.[Redmon et al., 2016]

El algoritmo plantea dividir la imagen de entrada en una grilla SxS cómo se puede observar en la Figura 4, mediante el uso de una red convolucional predice múltiples cajas delimitadoras y propiedades de clase para esas cajas.

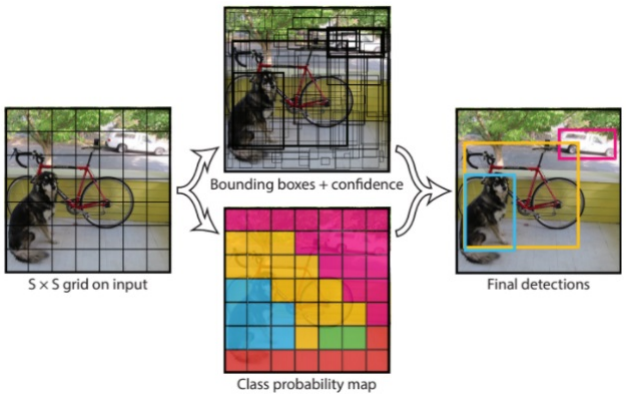

Figura 4: Esquema de funcionamiento de YOLO, 1 - Se divide la imagen en cuadrículas más pequeñas. 2A- Se calculan las diferentes cajas delimitadoras o bounding boxes y su confidence. 2B- Se calcula las probabilidades de las distintas clases. 3-Se obtienen las detecciones.[Redmon et al., 2016]

Si el centro de un objeto está en una celda de la cuadrícula, esa celda es la responsable de detectar el objeto. Cada celda detecta B cajas delimitadoras y niveles de confianza para dichas cajas, estos niveles generan probabilidades de que en la caja se encuentre un objeto y si esa caja contiene al objeto.Se define la como:

$$confidence = P(Object) * IOU[IntersectionoverUnion] \tag{6}$$

Si el objeto no se encuentra en la caja tiene un $confidence = 0$, de no ser así dependerá de el IOU [Rosebrock, 2005] entre la caja detectada y la caja real. Cada caja delimitadora consiste en 5 predicciones $x, y, w, h, confidence$. (x, y) es el centro de la caja, w es el ancho y h la altura de la caja. En paralelo se crea un mapa de probabilidades por cada celda de la grilla, compuesto por un vector de probabilidades C con todas las clases de objetos que pueden ser detectados $P(Clasei|Objeto)$, este proceso se realiza independiente de las cajas detectadas. En la inferencia, se multiplican las probabilidades condicionales de cada clase y la confianza de cada caja detectada para obtener la propiedad específica de cada clase por cada caja detectada.

$$P(Clasei|Objeto) * P(Objeto) * IOU = P(Clasei) * IOU \tag{7}$$

Esta medida indica la probabilidad de que cada clase aparezca en las cajas detectadas y cuál fue el acierto de la caja delimitadora.[Redmon et al., 2016]

4. Metodología

Para organizar el trabajo se divide en las siguientes etapas.

- **Diseño del prototipo**
  Esta etapa se caracteriza por el diseño del hardware a ser utilizado.

- **Adquisición de materiales**
  Al tener un mercado muy pequeño de plaza los componentes necesarios para la implementación pueden ser difíciles de encontrar por este motivo se han de traer del exterior, además de disminuir el costo de adquisición de los mismos.

- **Comunicación con ESP32-CAM**
  Al desconocer este microcontrolador, se debe de generar un aprendizaje a medida que se avanza con la implementación, su desenlace es la implementación de un servidor que permita transmitir las imágenes con las características necesarias para hacer factible el prototipo.

- **Interfaz para interacción con RiveroGlasses.**
  Esta etapa permite la interacción con el hardware creado de manera manual, permitiendo verificar el correcto funcionamiento de la implementación.

- **Detección de Objetos mediante python**
  Esta etapa se encarga de la implementación del algoritmo YoloV5 que

permite al prototipo detectar objetos que se encuentren delante del usuario.

- **Medir distancia a objeto detectado**
  Luego de ser detectados los objetos dentro del campo visual de la cámara se debe orientar el láser al objeto para medir su distancia.

- **Enviar información a RiveroPi para que informe al usuario.**
  Esta etapa comprende el final del proceso, luego de ser detectado el objeto/obstáculo, se mide su distancia y si la misma se encuentra en un radio menor al definido como distancia de seguridad, el prototipo debe informar, mediante comando de voz, la ubicación en azimut con respecto al frente del usuario y la distancia a la que se encuentra el obstáculo.

5. Diseño e implementación

5.1. Identificación del problema

La gran cantidad de posibilidades que brinda la robótica, la inteligencia artificial, la visión computacional entre otras áreas de la ciencias, pueden ser de gran ayuda al colectivo de ciegos. En este proyecto en particular, se focaliza en una solución concreta, y de esta manera sea realizable. Para esto se cuenta, tal como se menciona anteriormente, con la ayuda de Federico Rivero, persona ciega que ayuda a discernir cuáles son sus principales necesidades. En el Anexo A se encuentra la entrevista que se le realiza. El objetivo principal de la entrevista es lograr hacer una lista de prioridades de los problemas a resolver mediante el uso de las tecnologías desarrolladas durante el PRIA. Se habló en la misma de la interface de comunicación entre el prototipo y el usuario, se le plantea la posibilidad de realizar un sistema de comunicación que haga uso de actuadores mecánicos que actúan sobre la piel generado estímulos de las imágenes que eran captadas por la cámara, de esta manera no interferir en el sentido del oído. Rivero plantea que los ciegos perfectamente pueden movilizarse con un oído obstruido por un auricular, pero existen sistemas como el Orcam que el parlante no está dentro del oído, sino que se encuentra afuera del mismo y se regula el volumen del parlante dependiendo de las necesidades del lugar donde se encuentra el usuario. Otro factor que se consultó fue la forma que se debería dar la información de ubicación del obstáculo al ciego, Rivero nos comenta que en el entrenamiento que realizan al ir a un restorán, la ubicación de los objetos en la mesa les es dada como si fuera las horas del reloj.

Las prioridades ordenadas según su importancia derivadas de la entrevista con Rivero fueron las siguientes:

1. Detectar obstáculos e identificar su ubicación y distancia.
2. Detectar objetos en una habitación
3. Detección de texto.
4. Detección de billetes
5. Detección de colores de ropa
6. Detección de caras

5.2. Componentes que integran el sistema

5.2.1. Armazón de Gafas para soporte de componentes

Las gafas tienes sus orígenes en Italia antes del siglo VIII[Puyuelo, 2002], desde ese entonces evolucionan hasta el presente. Esta evolución, permite, a lo largo del tiempo, generar mejoras en los diseños permitiendo cumplir su objetivo y generar un grado de confort que permite ser un producto exitoso. Por estos motivos se decide que el soporte para los componentes del prototipo debe de ser un armazón modificado para las prestaciones requeridas. Se hace uso de un témplate modificado de lentes similares a google glasses [joris March, 2013] para desarrollar el armazón. El mismo es impreso haciendo uso de impresora 3D en PLA[ácido poliláctico].

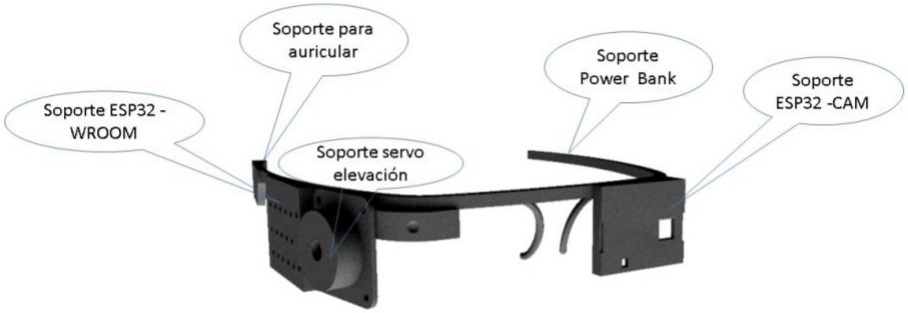

Figura 5: Gafas impresas en 3D.Contenido Propio.

5.2.2. Esp32-CAM

Microcontrolador que implementa un servidor web, es el encargado de generar un stream de video en formato mjpeg, también será capaz de enviar fotografías instantáneas en alta y baja calidad según sea requerido por el algoritmo.

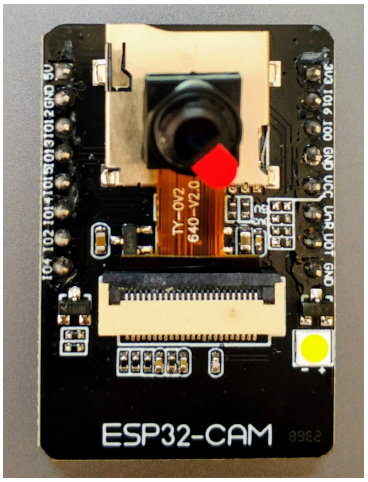

Figura 6: ESP32-CAM [Nowforever, 2019]

5.2.3. Esp32-WROOM

Este microcontrolador es el encargado de controlar y comunicar todos los sensores y actuadores que se encuentran en el RiveroGlasses. Genera un servidor websocket que permite:

- Controla los servomotores que posicionan al láser VL53L1X al ser detectado el objeto.
- Medir la distancia haciendo uso del láser VL53L1X mediante el protocolo i2c.
- Detección de botones touch que permiten la interacción con el sistema sin uso de la voz.

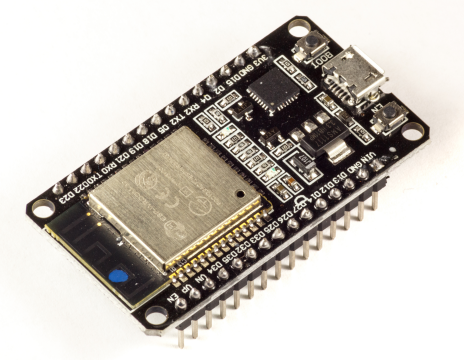

Figura 7: ESP32-WROOM [Ubahnverleih, 2018]

5.2.4. PowerBank

Batería de litio que proporciona la energía necesaria para que el sistema funcione. Característica principal capacidad 2600mAh, 5v dc output.

Figura 8: PowerBank. Contenido Propio

5.2.5. Servomotor de ronza y elevación para el control de la posición del sensor VL53L1X

Este sistema funciona en conjunto, haciendo uso de dos servos SG90. posicionados como se puede observar en la Figura 9 Al detectar un objeto, ellos deben de posicionarse orientando el láser al mismo, para permitir medir su distancia.

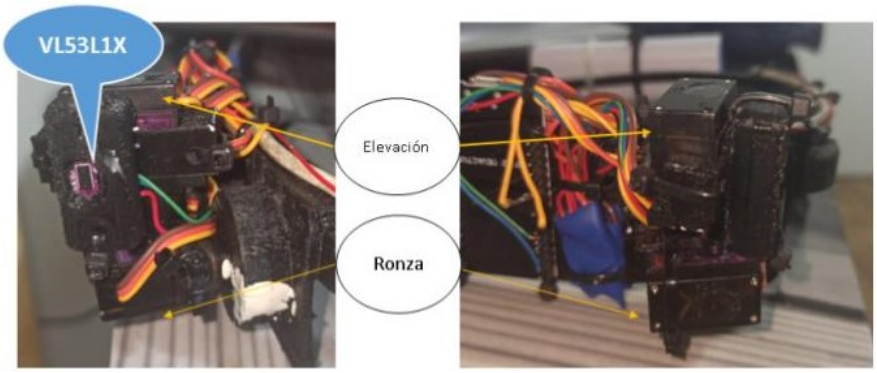

Figura 9: Sistema de posicionamiento del láser en ronza y elevación. Contenido Propio

5.2.6. Sensor VL53L1X

Sensor Time of Flight [TOF], este medidor láser tiene la capacidad de medir hasta 4m de distancia dependiendo de las condiciones de luz, en este momento se está utilizando el sensor VL53L0 a la espera del arribo del sensor antes mencionado. El sensor VL53L0 tiene un alcance de 2 m en interiores y 1,5 m en exteriores.

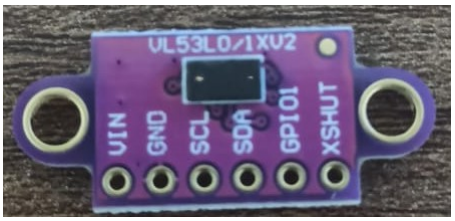

Figura 10: Sensor ToF VL53L1X. Contenido Propio

5.2.7. Auricular Bluetooth

Su función es la de transmitir la información procesada al usuario para que este pueda actuar en función de la información recibida.

5.2.8. Raspberry Pi

Es un ordenador de placa única de software libre, su sistema operativo oficial es Raspbian una versión adaptada del Debian. Permite el uso de python y otros lenguajes de programación.

5.3. Interfaz para interacción con RiveroGlasses de modo manual

Como primera etapa de desarrollo, luego del ensamble del hardware se crea una interfase web en el ESP32-WROOM que permiten el control manual de los servomotores de azimut y elevación, además proporciona la distancia que está midiendo el ToF en tiempo real y el stream de vídeo de la ESP32-CAM.

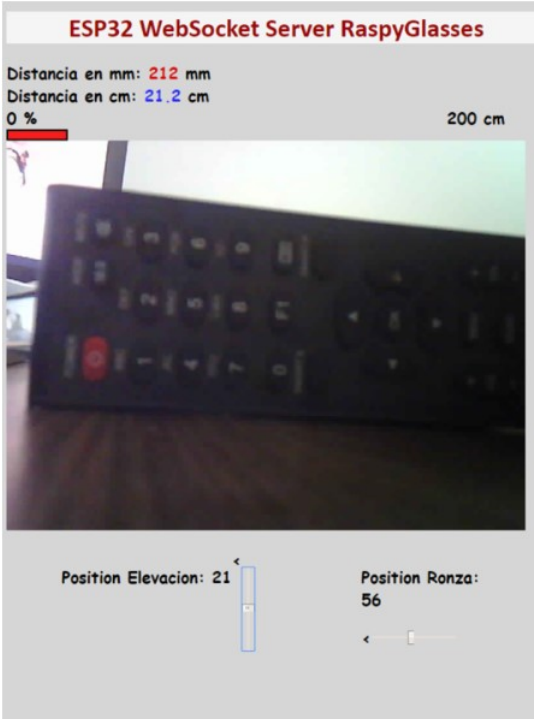

Figura 11: Interfaz de control Web. Contenido Propio

5.4. Resumen del funcionamiento de RiveroGlasses

El prototipo interactúa con el sistema **RiveroPi**, proyecto de fin de carrera de Nicolás Acerenza, mediante un comando de voz se genera una interrupción en el sistema, que activa a **RiveroGlasses**, seleccionando una de las tres opciones desarrolladas.

Modo Walk Permite al usuario moverse en un entorno desconocido para él. Se pretende que al caminar el sistema busque los objetos que se encuentran a una distancia que pueden obstaculizar el movimiento del usuario y avisar de los mismos, **RiveroGlasses** mediante el algoritmo de YoloV5 detecta objetos sin importar la clasificación de los mismos, la información de la posición de cada objeto, que es enviada a los servomotores de elevación y de azimut donde está colocado el láser, de esta manera se logra medir la distancia al objeto. Si la distancia es menor a 1,9 metros se almacena y se prepara el dato para ser enviado a RiveroPi. La información de los objetos almacenados se clasifica según el porcentaje de acierto, si este es superior a 60 % se comunicará la información al usuario haciendo uso del nombre del objeto que tiene en frente, de lo contrario, solo se informa que hay un objeto y no se lo clasifica. La ubicación en azimut se hace en sentido horario como se puede observar en la Figura 12, la cámara en azimut cuenta con un ángulo de 42 grados, se usa las 12 hs para orientar hacia el frente del usuario, las 10 horas es el máximo que simboliza 21 grados hacia la izquierda, y las 14hs simboliza 21 grados hacia la derecha.

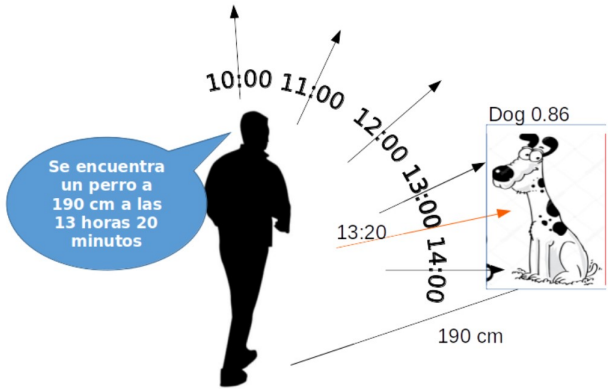

Figura 12: Descripción modo Walk. Contenido Propio

Modo findObject Permite al usuario encontrar un objeto que él desee, funciona de manera similar a la función walk, pero su diferencia es que no debe detectar otro objeto que no sea el especificado. El stream que se entregará a **RiveroPi** contendrá dos posibles sentencias, si supera el porcentaje de acierto en un 60 % **Riveroglasses** le dirá al usuario que *el objeto se encuentra a una distancia X cm y su ubicación está a Y horas Z minutos*, de lo contrario le informará que *el objeto puede estar a una distancia X cm y su ubicación está a Y horas Z minutos.*

Modo detectObject Permite al usuario saber qué objetos se encuentran en una habitación. Esta opción no otorgará la distancia a los diferentes objetos encontrados. Si el porcentaje de acierto es mayor a un 60 % se genera un stream que informa al usuario que objeto es y su ubicación, ésta última se informa con la codificación horaria antes mencionada, de lo contrario el stream generado informa que *puede haber un objeto y su ubicación codificada en formato horario.*

5.5. Comunicación entre los componentes del sistema

La comunicación entre los componentes se divide en 3 etapas para su mejor comprensión, puede ser visualizado en la Figura 13.

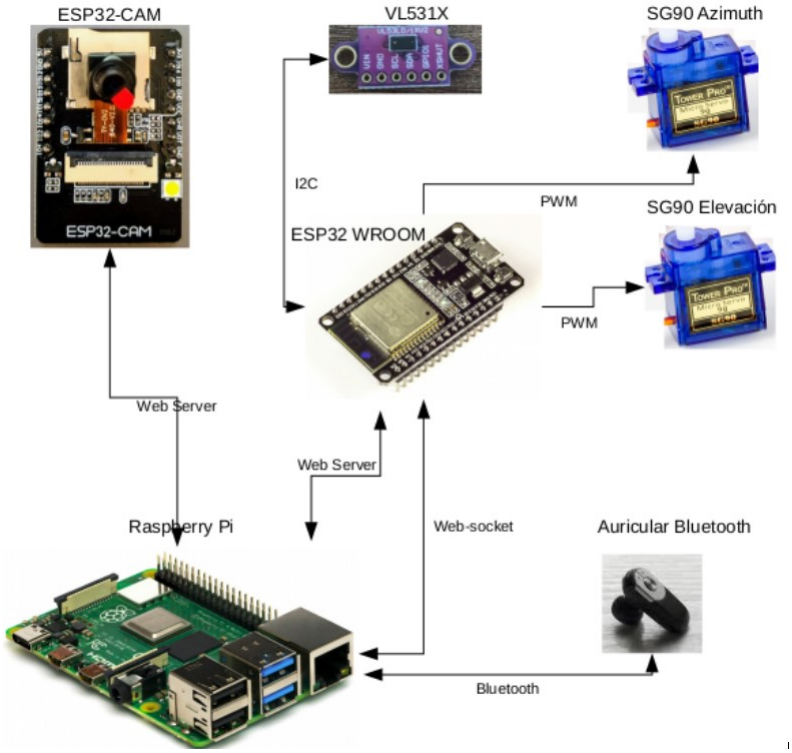

Figura 13: Diagrama de comunicación de datos. Contenido Propio

5.5.1. Obtención de la imagen para poder ser procesada

La comunicación entre el ESP32-CAM y el Raspberry se realiza por medio de un enlace wifi 2.4 ghz. En el ESP32-CAM se genera un servidor web que permite, la configuración de los parámetros de la cámara ingresando a la dirección 192.168.1.128, la obtención de imágenes en formato jpg mediante

la dirección 192.168.1.128/capture, y la obtención de un stream de vídeo en formato MJPEG mediante la dirección 192.168.1.128:81/stream.

5.5.2. Obtención de la distancia a los objetos

La comunicación entre el ESP32-WROOM y el Raspberry se realiza por medio de un enlace wifi 2.4 ghz. En el ESP32-Wroom se crea un servidor web, que permite el acceso a una página web en donde se puede realizar el control de los servomotores de manera manual, obtener la medición de distancia y el stream de vídeo de la ESP32-CAM para esto se debe de ingresar a la dirección 192.168.1.148. Además se genera un servidor websocket que permite al algoritmo generado en python acceder como cliente a la información de distancia del VL531X, el ángulo al que deben moverse los servomotores es enviado mediante el servidor websocket. La posición de los servomotores se controla por medio de una señal PWM generada en el ESP32-WROOM. La comunicación entre el TFT VL531X y ESP32-WROOM se realiza por medio del protocolo I2C.

5.5.3. Comunicación con el usuario

Esta se realiza por medio de un auricular Bluetooth asociado con la raspberry.

5.6. Algoritmo en python

El algoritmo completo se encuentra en Anexo B. Para poder realizar la ejecución se debe de contar con el siguiente software instalado: Python 3.8, torch 1.8.1, torchaudio 0.8.0, torchvision 0.9.1, tensorflow 2.4.1, YoloV5, gTTS 2.2.2 El diagrama de flujo del algoritmo se encuentra en el anexo E. Para explicar el algoritmo se divide en cuatro etapas facilitando la comprensión.

5.6.1. Etapa uno: interacción con RiveroPi

RiveroPi selecciona el modo de operación de RiveroGlasses mediante la función Run, se deben de pasar 4 parámetros:

- El parámetro **argument** permite seleccionar el modo que se usa **RiveroGlasses** *(walk y detectObjects)*.
- El parámetro **objects** define si se va a buscar un objeto específico o busca todos los objetos que puede detectar el algoritmo de visión, para esta última opción se debe dejar como **None**.

- El parámetro **threshold** define el límite inferior de porcentaje al cual el algoritmo YoloV5 detectara un objeto.
- El parámetro **View** activa la visualización de imágenes en una pantalla.

El retorno de información a **RiveroPi** es mediante la función *self.sendDatatoRivero*, que está compuesta por un stream de texto con la información que se debe de transmitir al usuario por medio de los auriculares.

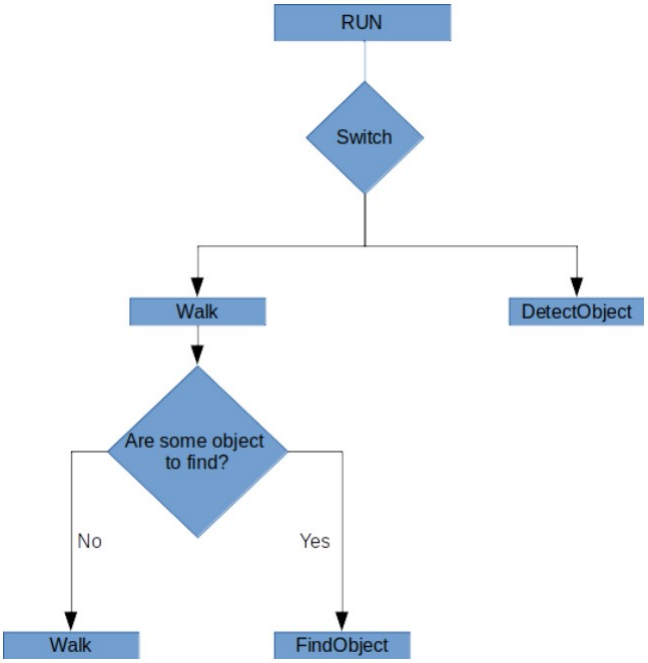

Figura 14: Diagrama de flujo de la función Ran. Contenido Propio.

5.6.2. Etapa dos: modo *walk*

Los parámetros que recibe la función **Run** son:

- Parámetro 1 = 1, selecciona modo walk.
- Parámetro 2 = None, habilita a que sean detectados todos los objetos por medio de YoloV5.
- Parámetro 3 = 0.05, permite que se detecten objetos con un porcentaje de acierto muy bajo, lo importante en este modo es que se detecten objetos que puedan obstaculizar al usuario en su caminar y no su clasificación.

- Parámetro 4 = False, aumenta la velocidad de predicción al no cargar el sistema con mostrar las imágenes generadas.

Al ejecutar la función se habilitar tres hilos como se observa en la Figura 15

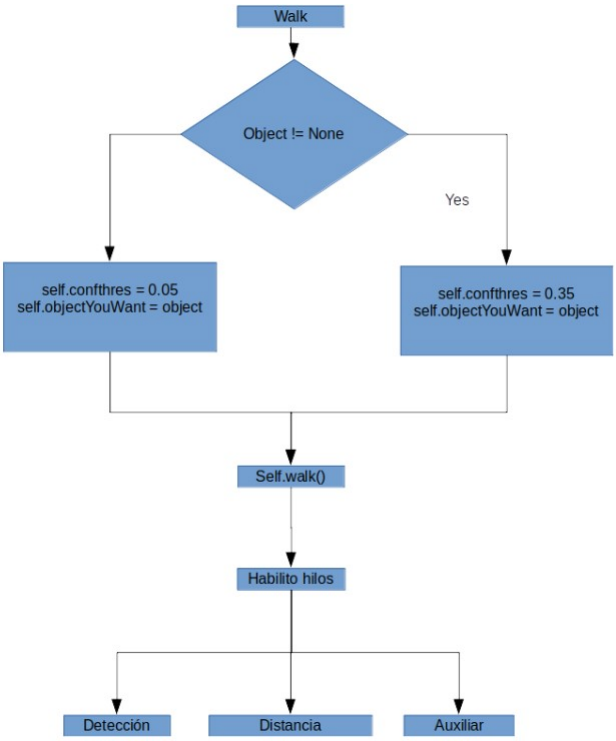

Figura 15: Diagrama de flujo de la función walk. Creación propia.

Hilo detección Es el encargado de ejecutar el algoritmo YoloV5, detectar los objetos, obtener el centro del rectángulo que enmarca el objeto y almacenar los objetos en un lista global que es usado por los otros 2 hilos. Al ejecutarse la función *detectStream()* recibe un parámetro que es la dirección de la cámara a ser usada. Dentro de la función se carga el modelo de red *resnet101* y sus parámetros, se detecta si está correctamente la inicialización de la cámara y se ejecuta el algoritmo de detección. Por cada imagen recibida desde el stream de vídeo el sistema detecta los objetos con una velocidad aproximada de 150 a 300 ms. Para cada objeto encontrado se ubica su centro y se almacena en la lista global *self.objetosEncontrados*. *self.objetosEncontrados*, es compartida con el hilo Auxiliar y se encuentra conformada por diccionarios con la siguiente sintaxis.

- 'OBJECTINDEX': Índice de objeto según su clase

- 'X': Posición x en la imagen
- 'Y': Posición y en la imagen
- 'DIST': Distancia desde el usuario al objeto
- 'MEDIDO': variable booleana de control para saber si fue medida
- 'OBJECTNAME': Nombre del objeto encontrado
- 'OBJECTPERCENTAGE': Porcentaje de certeza de que objeto es
- 'ENVIADO':variable booleana de control para saber si fue enviado a RiveroPi

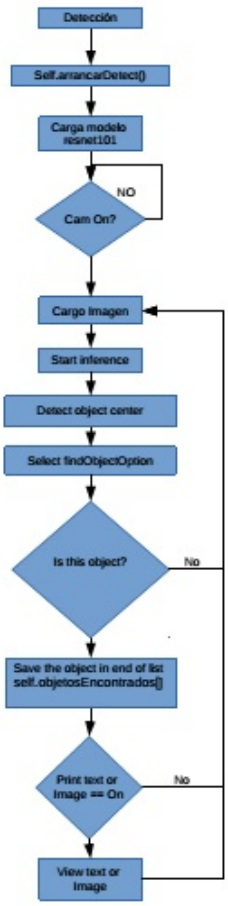

Figura 16: Diagrama de flujo del hilo detección. Creación propia.

Hilo auxiliar Este hilo se encarga de las funciones auxiliares a la detección de objetos, si *self.objetosEncontrados* no es una lista vacía, se envía la lista *self.objetosEncontrados* a la función *laserPositionServoSender*. La función *laserPositionServoSender* , por medio de un websocket envía la posición de azimut (*self.objetosEncontrados[objeto]['X']*) y elevación (*self.objetoEncontrado[objeto]['Y']*) a cada servo en el ESP32 Wroom, además modifica la variable global *self.objetosEncontrados* con el índice del objeto dentro de la lista *self.objetosEncontrados*, luego de enviarse la posición se generará un delay que está definido en la variable *self.tiempoEsperaMedida*, este permite el almacenamiento de la distancia en la lista *self.objetosEncontrados [self.contadorObjetos]['DIST']* para el objeto enviado. Al finalizar con todos los objetos se inicia la función *self.distanceControlObjectWalk* y se pasa como parámetro la lista de objetos a controlar la distancia (*self.objetosEncontrados*), se devuelve una lista con los objetos (*nearbyobjects*) que se encuentran a menos de la cota fijada en la variable *distanceMax*. La lista *nearbyobjects* es enviada como parámetro a la función *self.preparateDataToSend*, en donde se prepara los stream de texto que son enviados a **RiveroPi** y devolverá una lista con los stream generados (*preparatedData*). Podemos identificar dos tipos de streams distintos que se generan dependiendo del porcentaje de acierto en el objeto, si se supera un 60 % se genera el stream con el nombre del objeto, su distancia, y la posición del mismo con respecto al frente del usuario con formato horario. De ser un porcentaje menor se genera el stream de la forma antes mencionada con la diferencia que no se define cual es el objeto encontrado. La lista *preparedData* se pasa como parámetro a la función *self.sendDatatoRivero()* que es encargada de enviar los streams generados a **RiveroPI**.

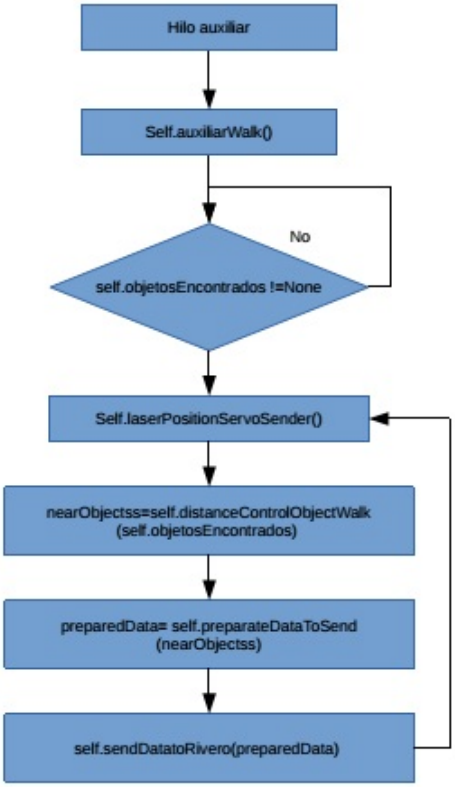

Figura 17: Diagrama de flujo del hilo auxiliar. Creación propia.

Hilo distancia Al iniciar la función *Run* se genera una conexión con el servidor websocket que se está ejecutando en el ESP32-Room y se envía el stream *"/?startWebpageDistance="* esto inicia el envío de un stream a todos los clientes de un objeto Json *{"DIST": distancia}* cada 250 ms. Por este motivo el hilo distancia ejecuta la función *self.websocket_scan_thread* si hay datos almacenados en *self.objetosEncontrados*, se habilita la recepción y se almacenará los datos en la lista *self.objetosEncontrados[self.contadorObjetos] ["DIST"]*, la variable *self.contadorObjetos* es modificada en el hilo Auxiliar por la función *laserPositionServoSender*. Sobre este hilo corre la función *self.variableReset()* esta se ejecuta cada *self.tiempoReset*, su objetivo es borrar la lista *self.objetoEncontrado* y la variable *self.objetoNuevo*. Al realizar los proceso de medir objetos y enviar el stream a **RiveroPi** se modifican las variables *'Medido'* y *'ENVIADO'* de la lista *self.objetoEncontrado*, estas son variables de control que si están en True no se vuelve a medir ni a enviar el dato por ser una tarea ya realizada.

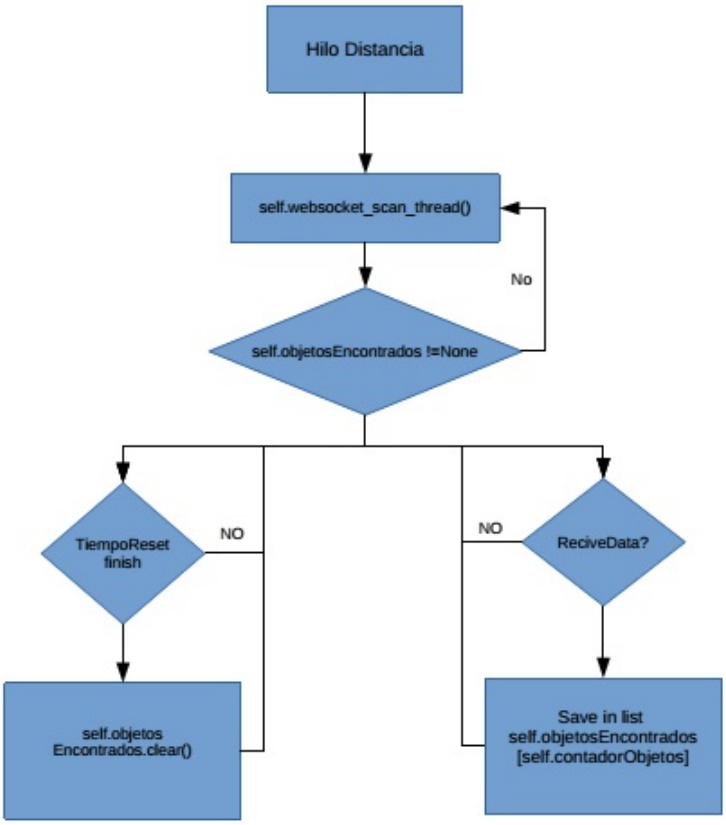

Figura 18: Diagrama de flujo del hilo distancia. Creación propia.

5.6.3. Etapa tres modo *findObject*

Este modo de operación funciona igual al modo walk la diferencia radica en los parámetros de pasado a la función Run, al funcionar de manera similar al modo walk solo se explicaran las diferencias entre los mismo.

- Parámetro 1 = 1, selecciona modo walk.
- Parámetro 2 = Objeto, habilita la búsqueda de uno de los 80 objetos entrenados en YoloV5.
- Parámetro 3 = 0.35, aumenta la certeza para buscar objetos específicos.
- Parámetro 4 = False, aumenta la velocidad de predicción al no cargar el sistema con mostrar las imágenes generadas.

Hilo detección El algoritmo YoloV5 solo intentara ubicar el objeto que sea seleccionado por el parámetro 2 de la función *Run*, guardándolo en la lista *self.objectosEncontrados*.

Hilo auxiliar No se hace uso de la función *self.distanceControlObjectWalk*. Todos los objetos encontrados en la lista *self.objetosEncontrados* son procesados por la función *self.preparateDataToSend*, para luego ser enviados a **RiveroPi** por medio de la función *self.sendDatatoRivero*.

Hilo distancia El funcionamiento se describe en el Párrafo 5.6.2, Hilo distancia

5.6.4. Etapa cuatro modo *detectObjects*

Los parámetros recibidos en la función self.Run serán:

- Parámetro 1 = 2, selecciona modo *detectObjects*.
- Parámetro 2 = None, habilita a que sean detectados todos los objetos por medio de YoloV5.
- Parámetro 3 = 0.35, permite que se detecten objetos con precisión bastante buena, sin ser detectados de manera múltiple.
- Parámetro 4 = False, aumenta la velocidad de predicción al no cargar el sistema con mostrar las imágenes generadas.

Esta invocará a la función *self.detectObject*, esta función correrá en un solo hilo. Inicialmente llamara a la función *self.takePicture*, esta se encarga de solicitar al ESP32-CAM una fotografía haciendo una llamada a la dirección *192.168.1.128/capture*, y la almacenara en **/fotos/capture.png*, esta dirección es la que le pasaremos al algoritmo YoloV5 para que identifique los objetos que hay en esa imagen. Luego de finalizar la captura se llamará a la función *self.detectImage()*, explicación realizada en el **Párrafo 5.6.2, Hilo detección**, se generará nuevamente la lista *self.objetosEncontrados* siendo enviada como parámetro a la función *self.preparateDataToSendObjectDetect*, su funcionamiento es similar a *self.preparateDataToSend* explicada en el **Párrafo 5.6.2, Hilo auxiliar**, pero se diferencia en el texto generado ya que no se informara la distancia del objeto al usuario, si la probabilidad de acierto es superior al 60 % se generará un stream con el objeto encontrado como se muestra en el siguiente ejemplo *"Se encuentra un perro (objeto identificado)*

a las doce horas cincuenta minutos" de no alcanzar el porcentaje se informará de la siguiente manera *"Es posible que el perro(objeto identificado) se encuentra a las doce horas cincuenta minutos"*

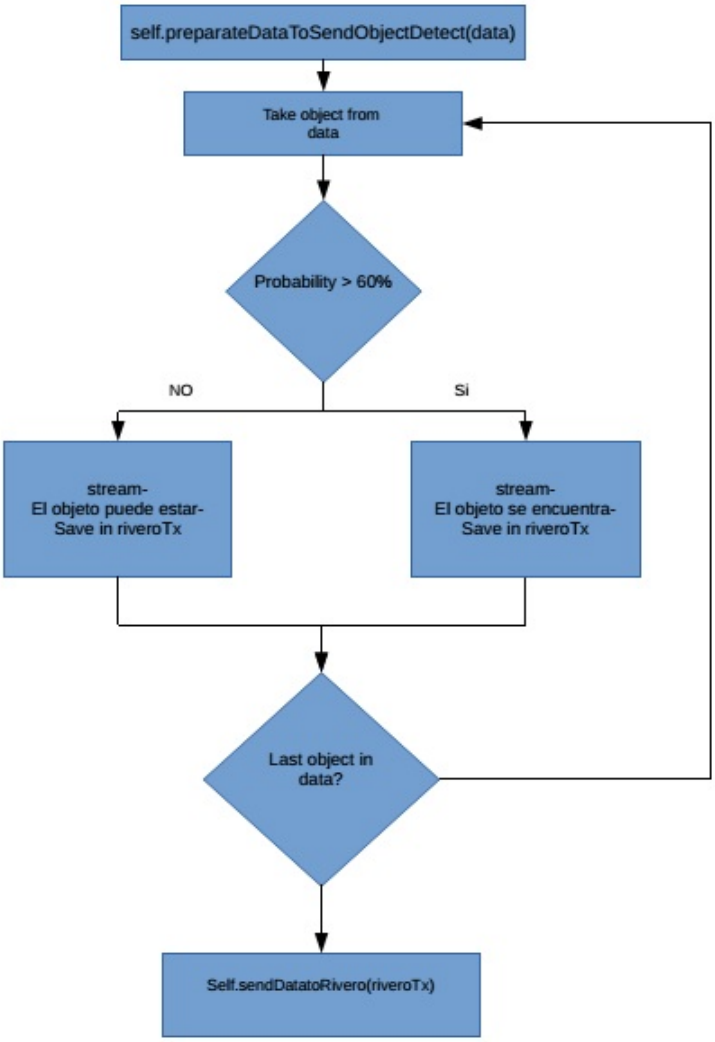

Figura 19: Diagrama de flujo de la función detectObject. Creación propia.

5.7. Algoritmo ESP32-WROOM

El algoritmo se encuentra en el Anexo D y su diagrama de flujo en el Anexo E. Su función es la de control de los servomotores y almacenar la medición de distancia del sensor VL531X que se encuentran instalados en el **RiveroGlasses**, para esto se generan tres proceso que funcionan en paralelo,

estos son la medición del láser, un servidor websocket y un servidor web que se observan en la Figura **??**.

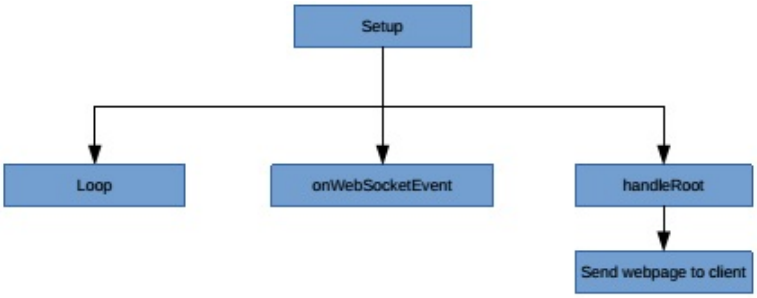

Figura 20: Diagrama de flujo Algoritmo ESP32-WROOM. Creación propia.

5.7.1. Función loop

Dentro de esta función se realizará la acción de medición de láser realizada cada 100ms almacenándose en la variable global *DistvalString*, para el filtrado de la información se hace uso de un filtro de media móvil. Si un cliente se conecta al websocket se habilitará la variable *SendWebpage* permitiendo el envío de la información de distancia a todos los clientes cada 250ms.

5.7.2. Función onWebSocketEvent

Esta función controla las solicitudes de los clientes en el websocket, por medio de un Switch-Case. Se utilizaron tres de los tipos de solicitudes permitidas.

Solicitud WStype_CONNECTED Informa la conexión de un cliente nuevo.

Solicitud WStype_DISCONNECTED Informa la desconexión de un cliente.

Solicitud WStype_TEXT Almacena un stream de texto en la variable pyload, la que se gestiona para la comunicación de información entre el algoritmo de Python y el ESP32-WROOM Posibles sentencias.

- **Sentencia *`/?startWebpageDistance=`*** Encargada de iniciar el envío de datos a los clientes websocket al modificar la variable *SendWebpage*.

- **Sentencia** ***/?distancia=*** Solicitud que realiza un cliente para que le sea enviada la distancia almacenada en ese momento en la variable global DistvalString.
- **Sentencia** ***/?servoElevacion=*** Posición a la que se debe de mover el servomotor de elevación para hacer que el láser se mueva a la orientación necesaria para iluminar el objeto.
- **Sentencia** ***/?servoRonza=*** Posición a la que se debe de mover el servomotor de azimut para hacer que el láser se mueva a la orientación necesaria para iluminar el objeto.

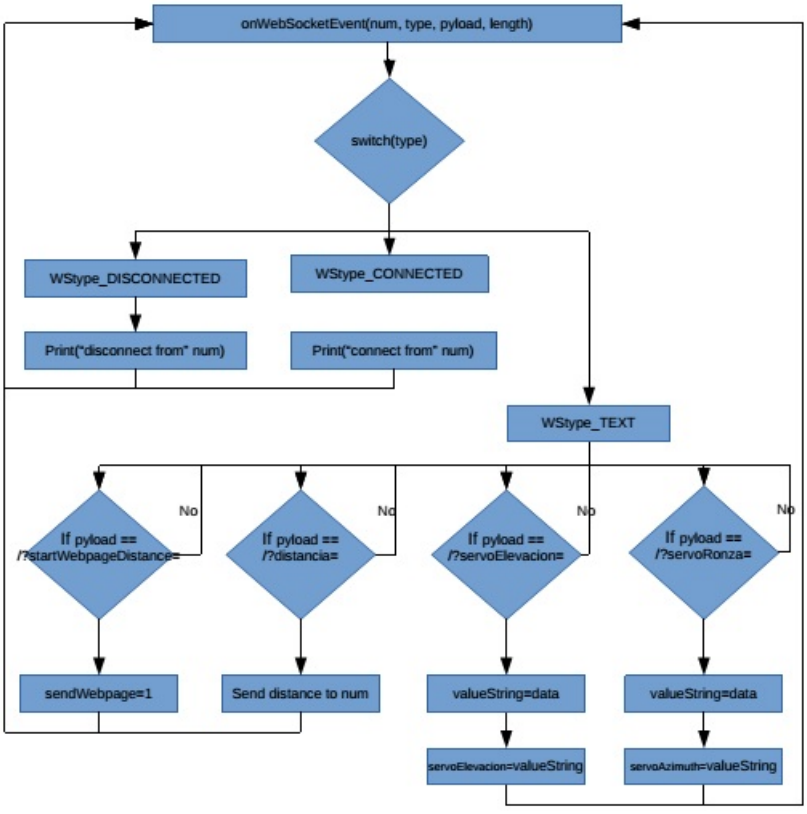

Figura 21: Diagrama de flujo Función onWebSocketEvent. Creación propia.

Función handleRoot Envía al cliente la página web para la interacción manual con los servomotores y control de distancia y visualización de la ESP32-CAM

5.8. Algoritmo ESP32-CAM

Se hace uso del ejemplo incorporado por ESPRESSIF [Espressif, 2021]. Se accedera al stream de video a requerimiento del algoritmo realizado en Python. A continuación se describen las funciones que serán usadas. En el ejemplo se genera un servidor web al que se accede generando un get, las funciones que se utilizan son:

- **index_handler** Permite la configuración de la cámara OV2640 incluida en el ESP32-CAM haciendo uso de una interfaz web. Se accede haciendo uso de una solicitud get a la dirección 192.168.1.128.
- **capture_handler** Hace el envío de captura en formato JPEG siendo configurado para una resolución de UXGA de 1600x1200 pixeles, para hacer uso del stream se debe hacer una solicitud get a la dirección 192.168.1.128/capture.
- **stream_handle**r Permite hacer un envío de stream de imágenes en formato MJPEG siendo configurado para una resolución de UXGA de 1600x1200 pixeles, para hacer uso del stream se debe hacer una solicitud get a la dirección 192.168.1.128/stream.

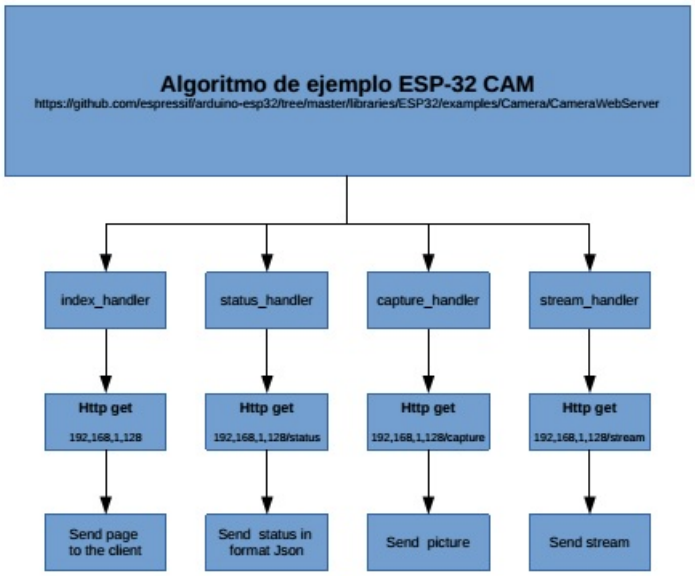

Figura 22: Diagrama de flujo ESP32-CAM Server. Creación propia.

5.9. Procedimiento de alineación entre imagen y láser

Es importante la alineación entre los objetos que son ubicados por el algoritmo YoloV5 y la posición que tiene el láser para que estos iluminen el mismo objeto. El láser que compone el sensor VL531X no es visible al ojo humano por este motivo se debió desarrollar un procedimiento para ajustar la interacción entre imagen y láser.

5.9.1. Componentes necesarios para realizar la alineación

- **Celular** - El láser es visible para las cámaras de los celulares, como se puede observar en la Figura 23 Por este motivo se debe de usar un celular que tenga instalada alguna aplicación que permita realizar un stream de vídeo a una ordenador, ejemplo Droidcam[dev47apps, 2021].
- **Cinta métrica** - Se usará para medir la distancia entre **iveroGlasses** y la pared.
- **Pared uniforme** - Se debe colocar a **RiveroGlasses** a una distancia de 2 metros de una pared que no genere ruido en la detección de imagen y de esta manera simplificar la detección del celular.
- **Cinta adhesiva** - Se usará para adherir el celular en la pared en la posición que sean requeridas.

Figura 23: Comparación entre el láser energizado y no energizado. Creación propia.

5.9.2. Procedimiento

- Colocar **RiveroGlasses** a dos metros de distancia de la pared uniforme.
- Abrir la página web de control de los servomotores en el ESP32-WROOM con la dirección 192.168.1.148.
- Iniciar aplicación en el celular, ejemplo Droidcam.
- Pegar el celular en la pared con cinta adhesiva, en la esquina inferior izquierda de la imagen que se encuentra en la página web 192.168.1.148.
- Mover la ubicación del láser en la página web hasta ver en la imagen del celular (aplicación Droidam), que es iluminado por el láser.
- Modificar las variables *self.anguloXl=Posición Ronza* y *self.anguloYl=Posición Elevación.*
- Pegar el celular en la pared con cinta adhesiva, en la esquina superior derecha de la imagen que se encuentra en la página web 192.168.1.148.
- Mover la ubicación del láser en la página web hasta ver en la imagen del celular que es iluminado por el láser.
- Modificar las variable *self.anguloXh=Posición Ronza* y *self.anguloYh=Posición Elevación*
- De ser necesario realizar algún ajuste, se deben de modificar las variables *self.offsetY* y *self.offsetX* de la clase **RiveroGlasses**

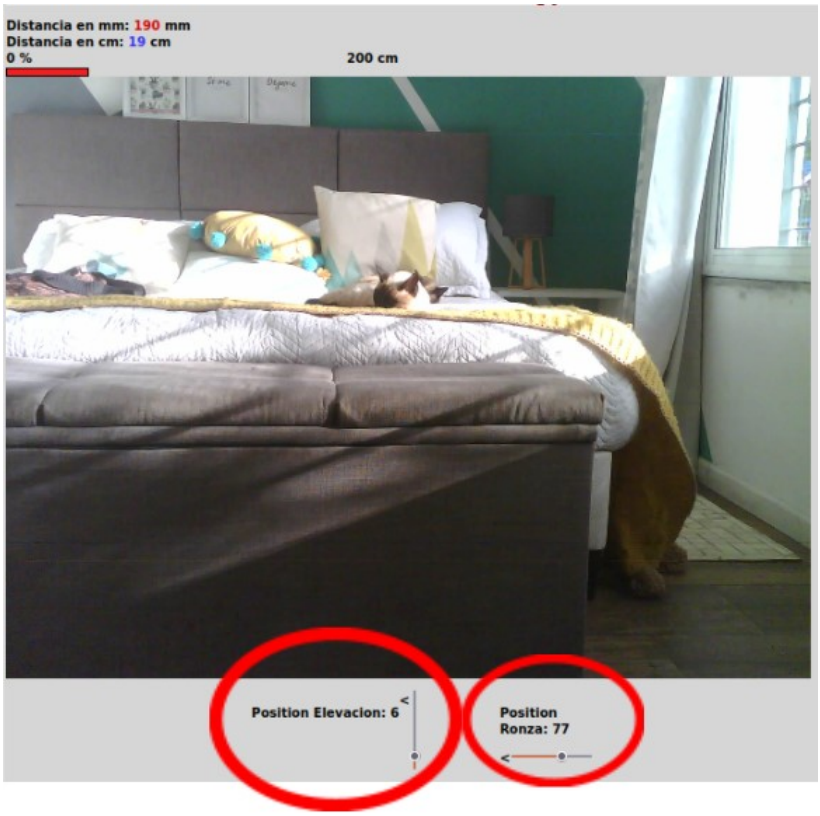

Figura 24: Página web 192.168.1.148, control de los servomotores y ángulo de los mismos para modificar las variables. Creación propia.

5.9.3. Comprobación de los datos obtenidos por el procedimiento 5.9.2

Al realizar la resta *self.anguloXh-self.anguloXl* se puede obtener el ángulo máximo que tendría la cámara medido por el láser en azimut, en el procedimiento realizado los valores fueron los siguientes *self.anguloXh=88 self.anguloXl=46*, al realizar la resta el ángulo máximo de la cámara es 42 grados. Para esto se puede imprimir un semicírculo como se observa en las imágenes 21 y 22, y de manera manual se puede usar un lápiz para marcar el ángulo máximo y mínimo en azimut y de esta manera corroborando los 42 grados que fueron calculados mediante el servomotor de azimut.

Procedimiento de verificación 111 grados Procedimiento de verificación 69 grados

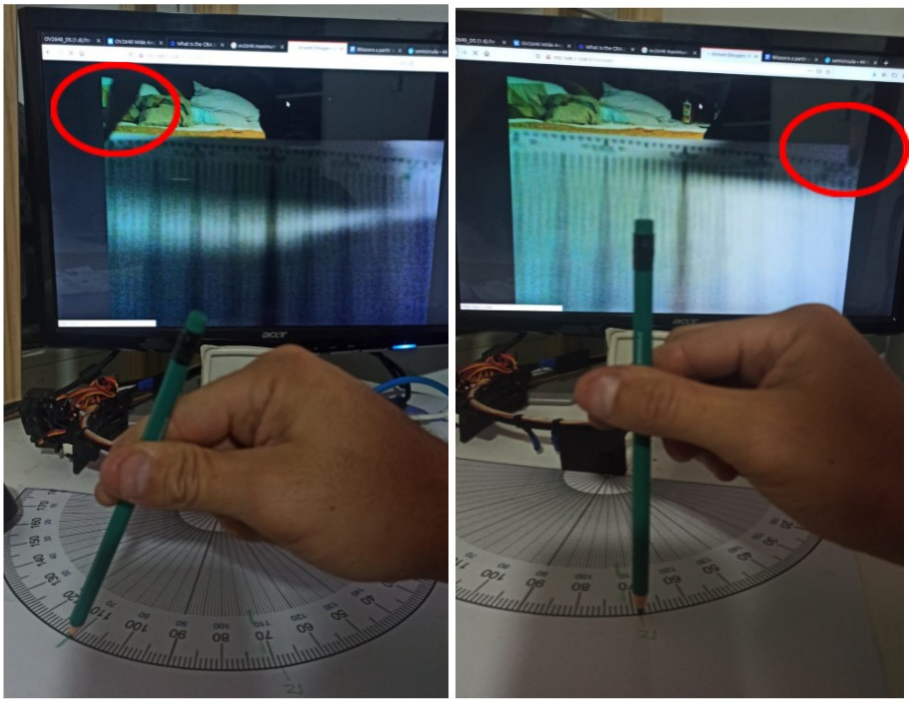

Figura 25: Procedimiento de verificación 111 grados izquierda | Procedimiento de verificación 69 grados derecha. Creación propia

5.10. Integración RiveroPi

Se preparó el dispositivo y la implementación de la aplicación para integrarse con la solución **RiveroPI** de Nicolás Acerenza. Para esto se debe de seguir los siguientes pasos.

Librerías Las librerías utilizadas ya están disponibles en el sistema RiveroPI, principalmente se hace uso de opencv, pytorch y torchvision. Además de las más comunes como numpy y pandas, entre otras requeridas por estas.

Definición de comandos Se definen los comandos como se indica en el documento de integración de RiveroPI. Se crea un archivo *riveroglasses_intents.ini* que contiene la siguiente configuración:

Extensión de clase RiveroApp Se implementa la aplicación extendiendo la clase **RiveroApp** y utilizando las funciones provistas para interactuar con el sistema. Por ejemplo para comunicar un resultado al usuario se utiliza la función speak("...").

Acceso Wifi al Acces Point brindado por RiveroPi Para que los dispositivos de **RiveroGlasses** puedan comunicarse con **RiveroPI**, se debe de configurar la red wifi a la que se conectan, en este caso **RiveroAP**, que es el access point provisto por la solución RiveroPI.

Por más información consultar la sección 9 del proyecto RiveroPI

5.11. Problemas conocidos del prototipo

5.11.1. Problema de diseño en el soporte de los lentes

En el diseño original no se tuvo en cuenta los triángulos angulares que se forman entre la cámara y el láser. El centro de cada sensor se encuentra separado 15 cm en el eje horizontal y 2,5 cm en el eje vertical. Lo antes mencionado genera que si un objeto ingresa en la zona A y es menor a las dimensiones antes mencionadas es posible que no sea visualizado por la cámara o en la zona B donde es posible que el láser no lo localice, lo antes mencionado dependerá de la distancia al objeto como se puede observar en la 26, la solución para este prototipo sin modificar su diseño es que los objetos a detectar sean como mínimo de 20x10 cm para de esta manera a distancias cercanas no tener problemas como se puede observar en la Figura 26.

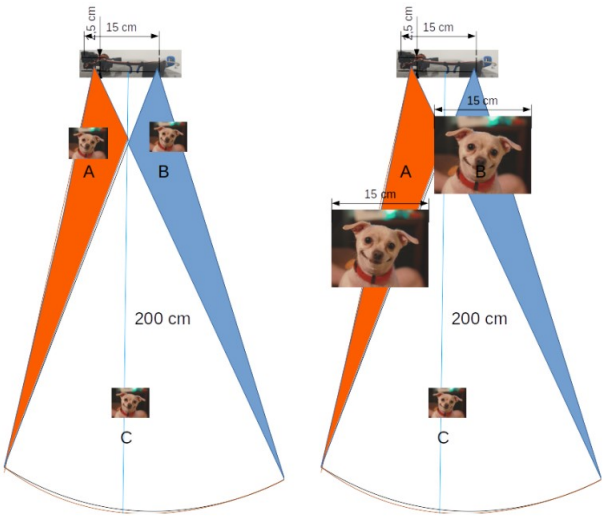

Figura 26: Objetos menores a 15x3 cm. Creación propia

5.11.2. Ruidos por el movimiento en los servomotores

Al detectar muchos objetos, el movimiento de los servomotores producen ruido que pueden molestar al usuario.

5.11.3. Detección de objetos

El algoritmo YoloV5 no siempre detecta todos los objetos que se encuentran dentro de la imagen como se muestra en la Figura 27.

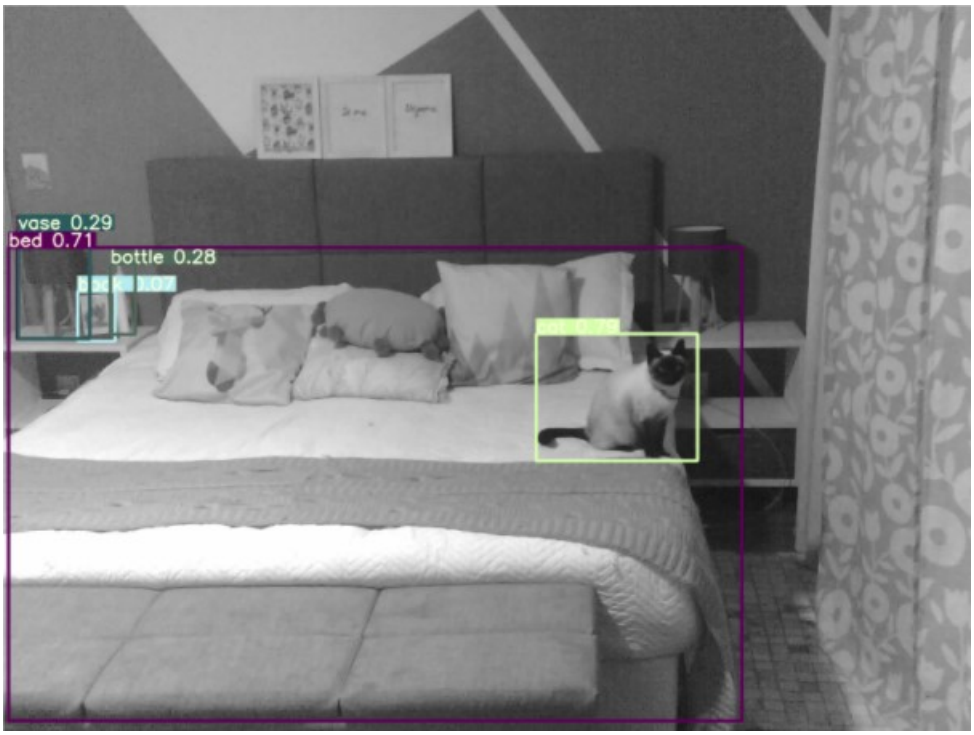

Figura 27: Objetos detectados por el algoritmo YoloV5. Creación propia

5.11.4. Medición de objetos

Algunos objetos pueden ser medidos erróneamente si se encuentran muy cerca unos de otros.

5.11.5. Velocidad del audio

Se debe de ensayar con el usuario cual debe de ser la velocidad y el largo del stream de audio para que cumpla con el requerimiento de la movilidad, si es muy lento no cumplirá con el objetivo de ayudar a la movilidad del usuario.

6. Resultados

6.1. Prototipo

En la Figura 28 se observa el prototipo finalizado desde varios ángulos diferentes, en ella se puede contemplar los componentes que lo integran.

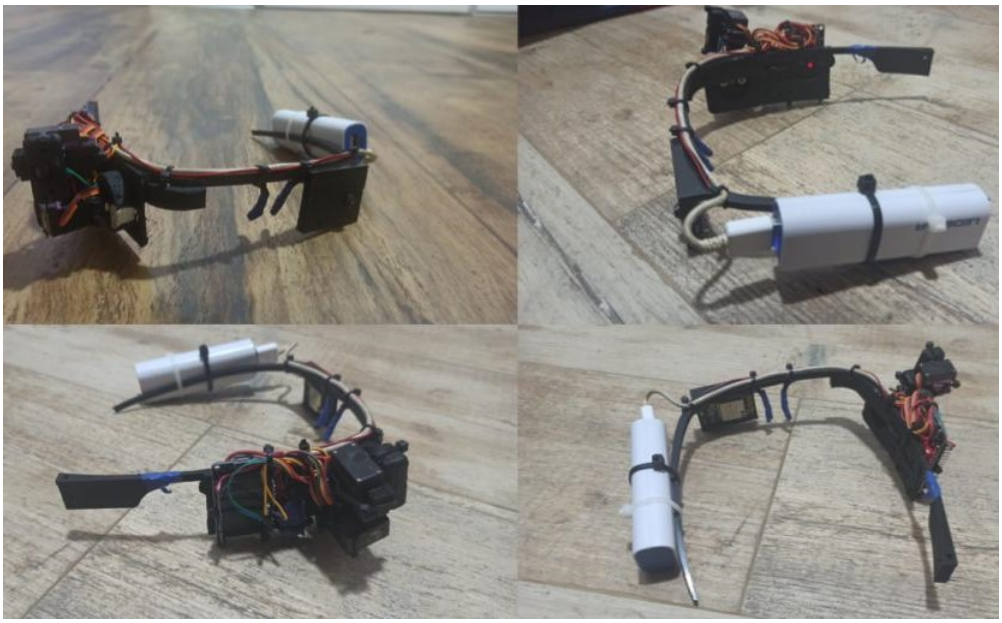

Figura 28: Prototipo Final. Creación propia

6.2. Ergonomía

Por lo mencionado en la **Subsubsección 5.2.1, Armazón de Gafas para soporte de componentes** se adoptó una forma de armazón de gafas, al ser impresa en una impresora en 3D, la terminación de la misma tiene imperfecciones en las plaquetas nasales como se puede observar en la Figura 29 para disminuir las molestias que se produce sobre la nariz se recubren las mismas, permitiendo su uso de manera aceptable.

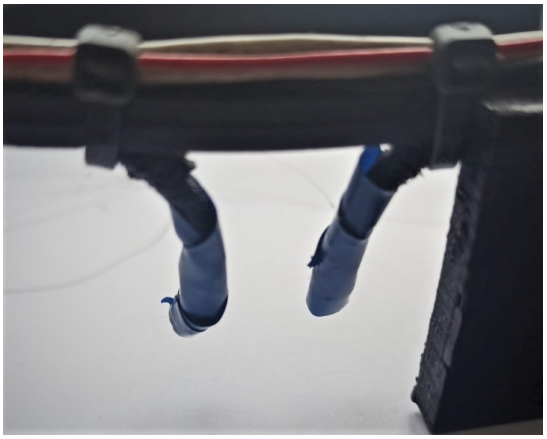

Figura 29: Plaquetas Nasales. Creación propia

6.3. Peso

La masa del prototipo es de 240 gramos, permite portarlo sin que sea una molestia excesiva.

6.4. Robustez del prototipo

El armazón es impreso en PLA, esto genera que su resistencia mecánica sea reducida.

6.5. Consumo de energía y autonomía

Para medir el consumo de energía que utiliza el prototipo en su funcionamiento, se conecta una fuente de 5V 4A que suministra la energía para la prueba, además se coloca un multímetro ET-1953 en paralelo a la fuente que mide la caída de tensión y un multímetro EX350 en serie para medir la corriente. El multímetro EX350 tiene una función que permite medir la corriente máxima y mínima durante un periodo de tiempo designado.
En primera instancia se procede a ejecutar el modo **Walk** en donde se produce el mayor consumo de energía. Se detectaron varios objetos como se puede observar en la Figura 30, esto permite que el sistema haga uso intensivo de los servomotores que orientan el láser.

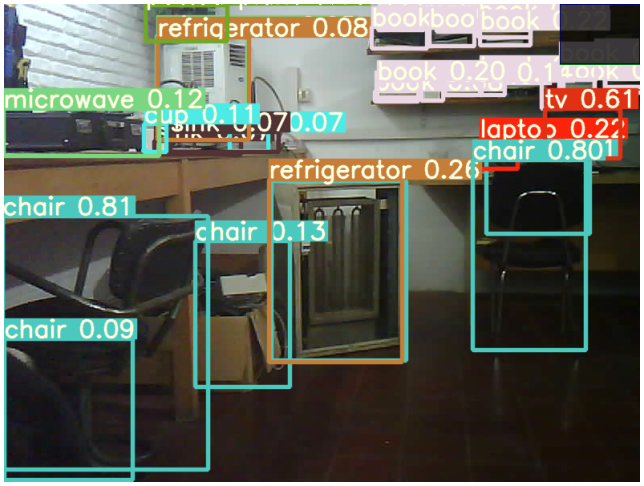

Figura 30: Detección de objetos usada para medir el consumo instantáneo de corriente en modo Walk

Al realizar las pruebas se observa que el consumo depende de varios factores entre ellos el movimiento de los servomotores, la comunicación de datos, la medición de distancia, el stream de vídeo entre otros, produciendo que no sea constante la medida, por este motivo se procede a medir durante un periodo de tiempo de 30 segundos el consumo máximo y mínimos de corriente con el multímetro EX350 generando la Tabla 1. Además se realiza una medición durante un periodo de treinta minutos en donde se pudo observar un consumo instantáneo máximo de 0.848 A.

Tiempo medido en segundos	Voltaje	Corriente maxima	Corriente minima	Promedio corriente
30	5,2	0,707	0,403	0,555
30	5,2	0,792	0,401	0,5965
30	5,2	0,68	0,391	0,5355
30	5,2	0,711	0,403	0,557
30	5,2	0,68	0,403	0,5415
30	5,2	0,707	0,39	0,5485
30	5,2	0,684	0,403	0,5435
30	5,2	0,742	0,393	0,5675
30	5,2	0,758	0,4	0,579
30	5,2	0,742	0,396	0,569
			Promedio	0,5593

Tabla 1: Corrientes máximas y mínimas medidas en un periodo de 30 segundos haciendo uso del modo Walk

Como segunda prueba se realizan diferentes medicaciones de consumo para conocer como se comporta el prototipo en diferentes acciones, como por ejemplo en modo Stand-by, accediendo al servidor de configuración que se ejecuta dentro del Esp3-Cam, entre otras. Estas mediciones se realizan en un periodo de tiempo de un minuto, tomando el dato de consumo de corriente máximo y mínimo de cada prueba. Este procedimiento genera la Tabla 2

Estado	Observaciones	Voltaje V	Corriente A	Potencia W
Stand-by	Corriente máxima	5,22	0,36	1,88
Server Esp32-CAM	Al entrar al servidor es en la Esp32-CAM	5,22	0,37	1,93
Cambio de Resolución en Esp32-CAM	SXGA a UXGA instantáneo llegó a 0,397 luego pasa a ser 0,369	5,23	0,37	1,93
Stream Cámara	Corriente máxima	5,23	0,43	2,22
	Corriente mínima	5,23	0,37	1,91
	Promedio	5,23	0,40	2,07
Stream + distancia	Corriente máxima	5,23	0,43	2,24
	Corriente mínima	5,23	0,37	1,91
	Promedio	5,23	0,40	2,08
Stream + distancia Servomotor ronza	Corriente máxima	5,23	0,68	3,55
	Corriente mínima	5,23	0,37	1,91
	Promedio	5,23	0,52	2,73
Stream + distancia + Servomotor Elevación	Corriente máxima, solo se da de 0 a 5 grados en el Servo de elevación	5,23	0,68	3,57
	Corriente mínima	5,23	0,37	1,91
	Promedio	5,23	0,52	2,74
Stream + distancia + Servomotor Elevación + Servomotor Ronza	Corriente máxima en un periodo de 10 minutos, solo se da de 0 a 5 grados en el Servo de elevación	5,23	0,77	4,03
	Corriente mínima	5,23	0,37	1,91
	Promedio	5,23	0,57	2,97
Funcionando con Algoritmo YoloV5 Modo Walk	Modo Walk, es el modo de mayor consumo Medida máxima tomada en un periodo de 30 minutos de funcionamiento	5,23	0,85	4,44
	Corriente mínima	5,23	0,37	1,94
	Promedio realizado cada 30 segundo entre minima y maxima lectura	5,23	0,56	2,93

Tabla 2: Consumos durante diferentes acciones del prototipo.

Haciendo uso de los datos observados anteriormente podemos desarrollar la siguiente Tabla 3, en donde el prototipo nos da una autonomía de uso continuo con un Powerbank de 2,6 Ah de aproximadamente 4hs.

Estado	Potencia Consumida W	Voltaje Power Bank V	Corriente consumida A	Horas de funcionamiento Powerbank 2,6Ah	Horas de funcionamiento PowerBank 10Ah
Stand-by	1,88	4,70	0,40	6,50	25,01
Funcionando con Algoritmo YoloV5 Modo Walk	2,93	4,70	0,62	4,18	16,07

Tabla 3: Tiempos aproximados de uso de manera continúa del prototipo.

6.6. Materiales constructivos y mantenimiento

Todos los materiales que componen el prototipo se encuentran en plaza, permitiendo el mantenimiento del prototipo en un plazo menor a 24hs.

6.7. Costos

En la Tabla 4 podemos observar los costos en dólares estadounidenses de los componentes que integran al prototipo, se decide poner los costos de plaza y los costos de compra en el exterior a la fecha de fabricación del prototipo.

Item	Cantidad	Exterior	Uruguay
Esp32-cam	1	10,00	22,22
VL53L1X	1	10,00	20,00
ESP32-WROOM	1	10,00	28,00
URSEC	1	2,91	0,00
Envío	1	26,68	0,00
PLA (plastico) x kg	1	40,00	40,00
SG90 (servo)	2	10,00	20,00
Varios	1	20,00	20,00
PowerBank	1	22,00	22,00
	Total	151,59	172,22

Tabla 4: Costos del prototipo, precios en dólares estadounidenses. Creación propia

La diferencia en costos para realizar una única unidad no son significativamente diferentes pero si se requiere realizar más de un prototipo es recomendable la compra de los componentes en el exterior.

6.8. Detección de objetos

6.8.1. Cantidad de objetos detectados en una misma imagen

Como se observa en la Figura 31 el algoritmo YoloV5 tiene una gran capacidad de detección de objetos. Dependiendo de la aplicación que se requiera ejecutar se deberá de ajustar el parámetro confidence threshold, por ejemplo en el modo Walk es ajustado a confidence threshold 0.05, permitiendo detectar objetos que no son correctamente clasificados por el algoritmo pero sí existen y pueden ser un obstáculo para el usuario. Para los otros modos es preferible aumentar el confidence threshold a 0.3 de esta manera mejoramos

la precisión de los objetos encontrados pero solo se afirmara que objeto es si la probabilidad es superior al 60

Figura 31: Parametro confidence threshold 0.05. Creación propia

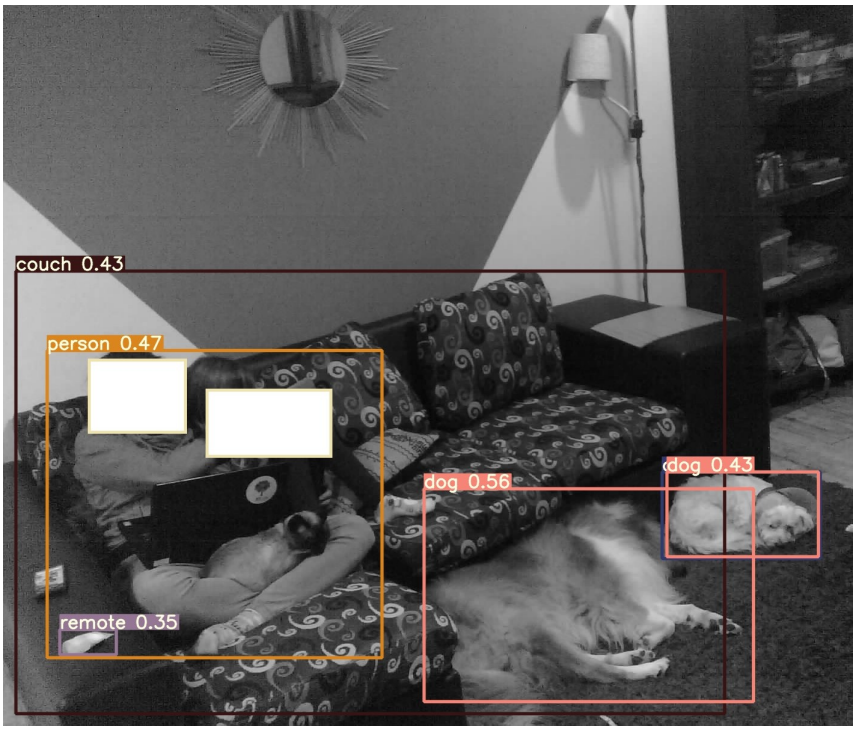

Figura 32: Parametro confidence threshold 0.4. Creación propia

6.8.2. Capacidad de discriminación de objetos

Como se observa en la Figura 33 ,el algoritmo YoloV5 tiene una gran capacidad para clasificar objetos, en la parte izquierda de la Figura 33 podemos observar la clasificación de objetos con un confidence threshold de 0.20, donde se observa que pudo discernir correctamente a un perro que se encuentra entre diferentes objetos, aunque la Figura 33 debido a la iluminación no cuenta con una buena definición. A la derecha de la Figura 33 se puede observar al perro desde otro punto de vista.

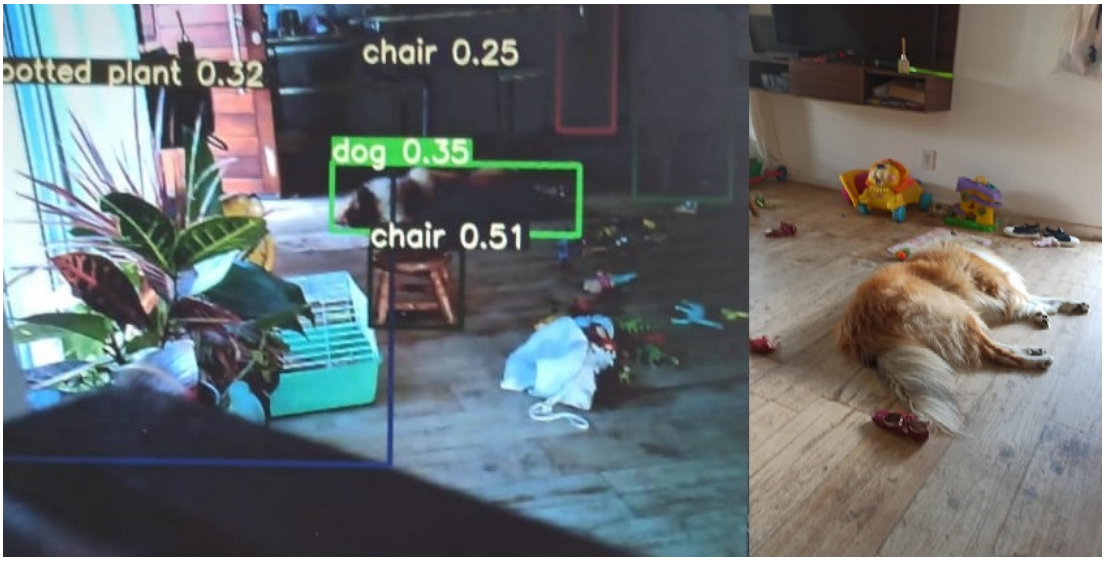

Figura 33: Clasificación de objetos mediante el algoritmo YoloV5 en comparación con imagen tomada desde otro punto de vista. Creación propia

6.9. Medición de distancia a objetos

6.9.1. Medición de distancia

En el lado izquierdo de la Figura 34 podemos observar la interfaz de control manual descrita en la **Subsección 5.7, Algoritmo ESP32-WROOM** en comparación con la medición realizada de manera manual en el lado derecho de la imagen. La medición realizada por el sistema ToF tiene una precisión aceptable pero debe de ser realizada apuntando el Láser de frente al objeto que se desea medir su distancia, de no ser realizada la medición correctamente la distancia medida sera errónea.

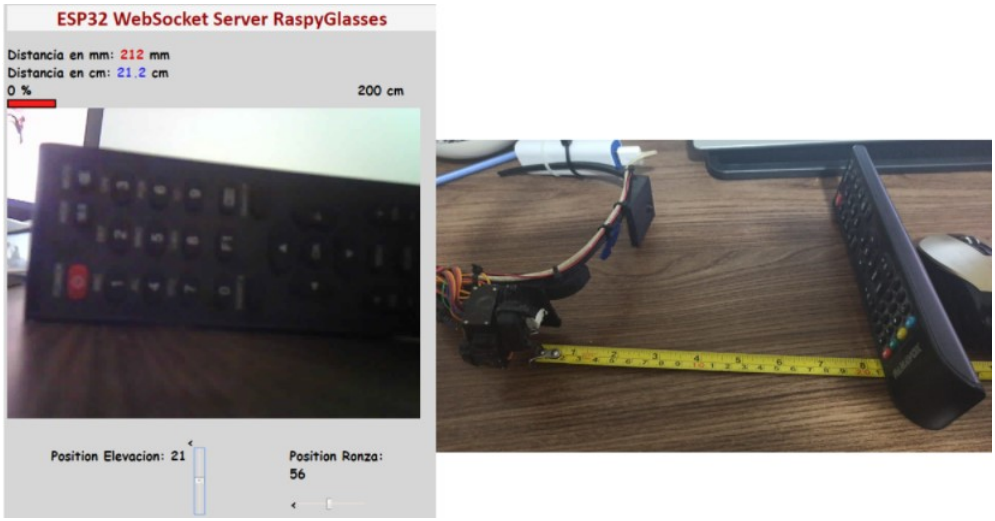

Figura 34: Comparación entre medición por sensor ToF vs medición realizada por un flexómetro. Creación propia

6.10. Capacidad de medir el objeto detectado

La siguiente secuencia de imágenes que observamos en las Figuras 35,36 es parte de un vídeo realizado para la prueba del modo de *detectObjects*, fue creada con la superposición de las siguientes imágenes [interfaz de control manual (izquierda abajo), salida del modo *detectObjects* (izquierda arriba y derecha abajo) y una cámara que registra las acciones llevadas a cabo que para luego comparar lo realizado (derecha arriba)]. La prueba que se realiza es la detección de un objeto, en esta ocasión una Tablet. Se observa que el mismo es detectado y el algoritmo es capaz de medir su distancia y la dirección con una buena precisión. Como se menciona en la **Subsubsección 5.11.1, Problema de diseño en el soporte de los lentes** el objeto a ser detectado debe de tener un tamaño mayor a 20 x 3 cm. Es importante aclarar que en la prueba no se reconoce el celular el 100 % del tiempo.

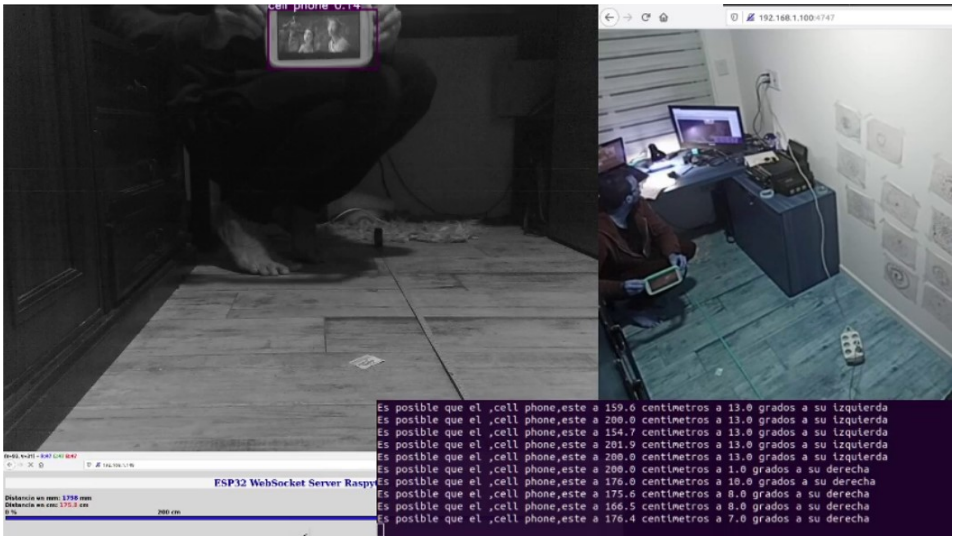

Figura 35: Detección de objetos y medición de distancia. Creación propia

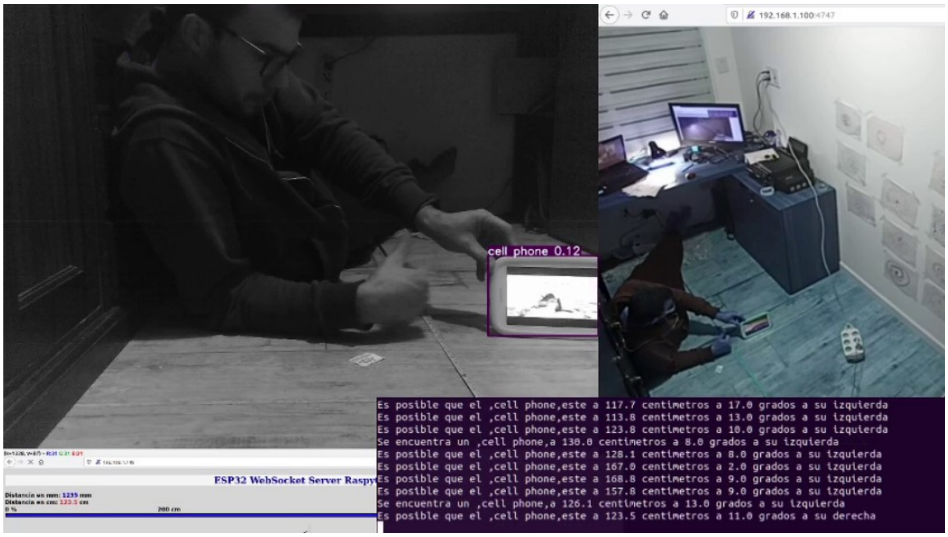

Figura 36: Detección de objetos y medición de distancia. Creación propia

6.10.1. Modos de operación

Se crean tres modos de operación, descritos en **Subsección 5.4, Resumen del funcionamiento de RiveroGlasses**. En el **anexo G** se adjuntan los vídeos que se crean al ejecutar las pruebas de los diferentes modos de operación.

Modo *Walk* Ayuda a detectar los objetos que se encuentran a menos de dos metros de distancia del usuario, su objetivo es la detección de objetos no la clasificación de los mismos. Informando al usuario si existe un obstáculo, otorgándole la ubicación en distancia y posición con respecto al frente de él.

Figura 37: Modo *Walk*. Creación propia

La Figura 37 pertenece a un vídeo, en el que se realiza la prueba del modo *Walk*, en él se puede observar la gran cantidad de objetos detectados pero solo son informados al usuario los que están a menos de 2m de distancia.

Modo *findObject* Ayuda al usuario a buscar un objeto que necesite encontrar. Al ser detectado el objeto se informará al usuario su ubicación con respecto al frente del usuario y distancia al mismo.

Figura 38: Modo *findObject*. Creación propia

Figura 39: Modo *findObject*. Creación propia

La secuencia de imágenes que podemos observar en las Figuras 38, 39 pertenece al vídeo de *findObject* adjunto en el **Anexo G**, en el observamos la detección del objeto que se ha sido requerido desde diferentes ángulos.

Modo *detectObject* Ayuda al usuario a saber qué objetos se encuentran en la habitación, el objetivo de este modo no es informar su distancia, solo se dará su ubicación.

Figura 40: Modo *detectObject*. Creación propia

La Figuras 40 pertenece al vídeo del **anexo G**, en donde se puede observar el funcionamiento de este modo, en el se solicita al algoritmo que busque los objetos que se encuentran en la habitación. Después de ejecutar la tarea se informa al usuario los objetos encontrados, si superan una confianza del 60 % se afirma su ubicación, si es menor la confianza en la clasificación se informa que es posible que se encuentre un objeto en esa ubicación.

7. Conclusiones y trabajos futuros

Tal cual se visualiza a lo largo del informe de proyecto, el protagonista, la persona destino de este proyecto es el señor Federico Rivero, que en su condición de ciego desde los 15 años de edad, representa, en gran medida, al colectivo que es beneficiado por el desarrollo de este dispositivo. El valor inherente de sus opiniones, de su experiencia, de las dificultades que encuentra para su normal desarrollo en la sociedad, tanto a nivel doméstico en el entorno cercano, como al desenvolvimiento como persona en la comunidad en la cual se insertan y en definitiva en el mundo, dado que dicha condición, la de la ceguera, en la gran mayoría de los casos, no es tenida en cuenta en la urbanización o diseños de soluciones socio-culturales y tecnológicas. Por lo tanto, el valor de sus entrevistas para este proyecto es importante, pues permite enfocar la nueva tecnología a la solución de problemas reales y significativos. Se debe tener en cuenta, que la asistencia que dicho dispositivo aporta a poder detectar los objetos, buscar y localizar, a su vez, puede permitir que el usuario logre integrarse mejor en la sociedad y el resto de las personas con quienes interactúan también logren empatizar e integrarse con el usuario ciego, resolviendo a su vez desafíos sociales propios del desconocimiento de

las dificultades de los demás. No queda ahí la experiencia de Federico Rivero, sumado a todo lo anterior, también comparte anécdotas basadas en el uso de otros dispositivos tecnológicos que buscan el mismo objetivo de asistencia, pero en este aspecto, Federico remarca que dichas tecnologías no tienen en cuenta los requisitos o capacidades del usuario. En este aspecto último es importante mencionar como una conclusión o quizás corolario de este proyecto, si bien la ceguera es una condición que puede tener diferentes orígenes, cada persona que la padece es única y por tanto, su experiencia, su necesidad de asistencia, su entorno tanto físico como humano es diferente para cada ser; en este aspecto, este dispositivo, puede tener la capacidad de la versatilidad, tanto en su robustez y simpleza de sus materiales como en la capacidad de cambiar sus prestaciones, adaptándose a las necesidades finales del usuario, volviendo al dispositivo altamente funcional.

Por lo anterior, es que se plantean dichas necesidades como objetivos a cumplir o resolver, los cuales se detallan a continuación.

7.1. Metas y objetivos alcanzados

Como se ha desarrollado a lo largo de la **Sección 6, Resultados** en donde se describe la implementación del prototipo, podemos afirmar que se logra alcanzar:

- El desarrollo del diseño del soporte que permite tener todos los componentes necesarios para el correcto funcionamiento del prototipo.
- El control de la dirección a la que mide el láser, haciendo uso de un sistema compuesto por dos servomotores que permiten la orientación en azimut y elevación.
- Se creó una plataforma que permite la interacción de forma manual con el prototipo, permitiendo controlar los servomotores de azimut y elevación, ver el stream de vídeo que genera la cámara ESP32-Cam y la medición del láser VL53X1.
- Haciendo uso del algoritmo YoloV5 se ha logrado la detección de objetos con una gran probabilidad de acierto y capacidad de discriminación entre objetos como se muestra en la **Sección 6, Resultados**.
- Mediante el uso del sistema de control en azimut y elevación y la detección de objetos por intermedio del algoritmo YoloV5, el prototipo es capaz de realizar mediciones a los objetos clasificados como se observa en la **Sección 6, Resultados**.

- Al estar integrado el sistema con **RiveroPi** es posible la interacción mediante el uso de audio con el usuario.

Haciendo referencia a lo anterior se concluye que se ha alcanzado un prototipo que permite cumplir con el objetivo específico propuesto, permitiendo la interacción con la plataforma **RiveroPi** y que proporciona al usuario la dirección y la distancia de los objetos u obstáculos que se encuentran delante del mismo. Estos resultados permiten aportar al objetivo general de mejorar la independencia de las personas ciegas.

7.2. Madurez tecnológica

El sistema usado por la Nasa para medir el grado de madurez en la tecnología se denomina TRL [Technology Readiness Levels]. Se consideran 9 niveles desde TLR 1 idea Básica, hasta el TLR 9 - Pruebas con éxito en entorno real [Ibáñez, 2014]. Si se tiene en cuenta lo antes mencionado y los resultados obtenidos, podemos afirmar que el prototipo es un TLR 3 - Prueba de concepto. El prototipo ha logrado cumplir con los requerimientos que permiten alcanzar las metas y objetivos propuestos, pero se debe de realizar pruebas de laboratorio por usuarios ciegos, validar el correcto funcionamiento de los algoritmos, mejorar el diseño y el confort, para de esta manera poder aumentar de nivel en el TRL.

7.3. Percepción de Federico Rivero

"Me alegra que haya podido aportar al menos un granito de arena en esto, así sea un prototipo no importa ya es mucho, como yo le decía a Nicolás, han logrado muchísimo y bueno han puesto el interés que de repente otras personas no han puesto en investigar que poder hacer para mejorar la calidad de vida de la gente ciega, eso es muy valorable y para mí más que valorable es invaluable. Yo estoy muy agradecido por lo que están haciendo cualquiera de los dos ... Así sea un prototipo en el momento que vos quieras que yo lo pruebe, va a ser un placer poder probarlo y poder transmitir lo que se siente... " (F. Rivero, comunicación personal, 8 de agosto de 2021)
Estas palabras demuestran las posibilidades que da el desarrollo tecnológico enfocado a la solución concreta de necesidades, en este caso en especial, necesidades humanas de un colectivo de la población mundial. Se puede concluir que esta tecnología tiene un impacto positivo en la vida de las personas, a través de la mejora de la calidad de sus vidas en su aspecto más amplio, desde la independencia hasta su inclusión en la sociedad. Como corolario de lo anterior y como conclusión en lo referente a la metodología utilizada para

el desarrollo de esta tecnología, un aspecto crucial y quizás, desencadenante de los resultados, es contemplar de manera adecuada la matriz de interesados. Si bien lo anterior, puede interpretarse un poco exento de empatía, en contraposición con las expresiones sinceras de Federico Rivero, no es así su proceso de investigación. La apertura de Federico, para volcar cuestiones privadas de su vida cotidiana tiene que ver con, no sólo contemplar el interés de la población ciega, sino de la comunicación y empatía para comprender los desafíos que deben vencer en una sociedad diversa y la propia necesidad humana de la independencia en sus movimientos en la sociedad. Por esa razón, es importante incluir en esta conclusión, las palabras textuales de Federico Rivero.

A lo anterior, se puede mencionar otro corolario que demuestra la importancia de no perder el objetivo principal de este proyecto: El prototipo desarrollado no compone un fin en sí mismo, sino que es la herramienta (camino) para resolver problemas significativos en las vidas de las personas ciegas, mejorando su calidad de vida, y que el desarrollo tecnológico del mismo está subordinado a la continua retroalimentación con el destino del servicio; y por tanto, el dispositivo desarrollado se vuelve funcional, innovador y mejorable.

7.4. Problemas detectados en el prototipo

Durante la implementación (**Subsección 5.11, Problemas conocidos del prototipo**), se detectaron varios problemas que permiten realizar mejoras al prototipo, entre ellas podemos citar la ubicación del centro de la cámara y el centro del láser que generan que el objeto detectado tenga un tamaño mínimo de 20x3 cm para poder ser medido. El ruido realizado por el movimiento de los servomotores al detectar muchos objetos puede ser molesto para el usuario. La velocidad de reproducción del audio podría limitar la velocidad a la que se deba mover la persona ciega por ser un canal único para transmitir todos los objetos detectados. Lo anteriormente mencionado presenta oportunidades de mejoras para el prototipo.

7.5. Trabajos futuros

Presentar el sistema a organizaciones interesadas, como la Unión de ciegos del Uruguay, de forma de evaluar la solución y obtener más información sobre las necesidades de los ciegos esto permitirá con la metodología apropiada incrementar el nivel TRL alcanzado. Se debe de estudiar la posibilidad de realizar cambios al prototipo, que permitan mejorar las características del mismo, algunas posibles mejoras son:

- Integración de leds infrarrojos para permitir a la cámara tener visión nocturna.
- Entrenamiento del Algoritmo YoloV5 a imágenes que provengan de cámaras térmicas.
- Mejorar la distancia entre el centro de la cámara y láser.
- Mejorar la ergonomía de los lentes.
- Mejorar la estética del producto.
- Mejora en los algoritmos de clasificación.

Es interesante plantear otros posibles usos para el prototipo, ya que el prototipo puede ser un asistente que se entrena para la detección de los requerimientos de la tarea. Un ejemplo puede ser el de un bombero que entra a un incendio con el sistema entrenado para detectar personas en condiciones extremas, al identificar una persona, debe informar la ubicación de las mismas al bombero, o transmitir información al equipo encargado del control del incendio. Otro ejemplo puede ser la detección de embarcaciones en zonas de interés para el estado, donde el prototipo puede estar a la espera de que una embarcación sea detectada, al suceder este hecho informa al centro de control su ubicación, distancia y generar una captura de la embarcación.

A. Entrevista con Federico Rivero

Nombre - Federico Rivero
Edad 53 años

Preguntas:

1. P: ¿Qué pasaría si obstaculizamos el oído con auriculares?

   R: No habría problema, si solo se obstaculiza un oído y si se puede controlar el nivel de la intensidad del audio. Orcam utiliza un parlante que no se introduce en el oído.

2. P: ¿Has usado auriculares cuando te mueves por la casa?
   R: Si, si tengo un oído libre no tengo problemas

3. P: ¿Podrías moverte de la misma manera que lo haces sin auriculares?
   R: Si, si tengo un oído libre no tengo problemas.

4. P:¿Sería mejor que usáramos otro canal de comunicación contigo?
   R:No es necesario, porque vas a complicar la investigación.

5. P: ¿Qué pasaría si obstaculizamos el oído con auriculares?

   R: No habría problema, si solo se obstaculiza un oído y si se puede controlar el nivel de la intensidad del audio. Orcam utiliza un parlante que no se introduce en el oído.

6. P: ¿Has usado auriculares cuando te mueves por la casa?
   R: Si, si tengo un oído libre no tengo problemas

7. P: ¿Podrías moverte de la misma manera que lo haces sin auriculares?
   R: Si, si tengo un oído libre no tengo problemas.

8. P:¿Sería mejor que usáramos otro canal de comunicación contigo?
   R:No es necesario, porque vas a complicar la investigación.

Posible investigación:
Durante la entrevista le propongo a Federico lo siguiente:
Estoy investigando el uso de tecnología para generar en tu piel diferentes sensaciones, la idea que tengo por ahora y que estoy lejos de poder concretar

es generar una pantalla en tu piel formada por una matriz de puntos que la cual represente de manera muy aproximada a lo que vemos nosotros, la idea sería como realizar un dibujo en la piel de lo que hay enfrente de ti, esto es muy difícil inicialmente de realizar por la gran cantidad de punto que debería representar en tu piel y tu tendrías que aprender a procesar esa imagen además. Otra posibilidad, inicialmente más sencilla es generar una cuadrícula más chica que pueda escribir símbolos o letras en tu piel, por ejemplo si hay un árbol a tu derecha, el sistema procese esta información y te pueda avisar que hay un árbol a tu derecha con un símbolo que sea derecha y otro que sea árbol. Otra posibilidad es generar varias de estas matrices y se colocarían en varias partes del cuerpo y cada una te podría mostrar cosas que hay en esos cuadrantes. Hay que ver a que podemos llegar a hacer, pero es importante tener un feedback tuyo para organizar y acorralar lo que voy a investigar para que sea de utilidad.

R:*Me es muy interesante esto que mencionas pero, no creo que sea necesario que te compliques tanto en desarrollar eso, ya que con un solo oído nosotros nos podríamos manejar apropiadamente.*

Mayores problemas citados por Federico Rivero

Alto costo de las soluciones que ayudan a los ciegos y muchos de estos productos no salen del país de desarrollo.

Cuenta que una de las cosas que más le gusta hacer en verano es ir a la playa y si no tiene acompañante no puede realizarlo, ya que no pueden encontrar las escaleras, o encontrar la salida de la playa, se nota una gran sensibilidad al hablar de esto, comenta que por diferentes razones este año no va a poder ir entre una de ellas es que su madre estaba quebrada y no lo iban a poder acompañar.

Moverse en una terminal de ómnibus

En un Shopping ¿Dónde está el local y cómo es y cómo llegar al mismo?

Color de la ropa es muy difícil para ellos poder combinar la ropa adecuadamente al no ver los colores, lo que hacen es doblar de manera especial la ropa, o le piden a alguien que los ayude.

Leer billetes las marcas que traen los billetes normalmente se borran por este motivo los doblan de manera diferentes dentro de la billetera

Leer texto si no es en braille no lo pueden leer y deben de pedir ayuda a otra persona.

Al caminar en la calle se enfrentan a la dificultad y el estrés de orientarse en situaciones desconocidas, generando que muchos ciegos no quieran salir solos. Otro problema es que por desconocimiento y descuido del resto de la población les pueden pisar el bastón y rompérselo quedando indefensos en esa situación, Rivero contó que él lleva un bastón de respeto ante estas situaciones porque ya van varias veces que le pasa.

Cuenta que la óptica Estela Jinchuk es la importadora del producto Orcam my eye en el país y que su costo ronda los 6 mil dólares. Además cuenta que están haciendo una colecta entre la familia para poder regalarle a su sobrino (ciego de nacimiento) para de esta manera pueda ser más fácil su ida a la facultad. Dentro de las cosas interesantes que cuenta Rivero sobre el Orcam es que el parlante no está dentro del oído sino que sobre el armazón del lente y se puede regular el volumen dependiendo de las necesidades del usuario.

Lista de prioridad de problemas a solucionar ordenada de mayor a menor prioridad que me permitan enfocar mi investigación a sus necesidades

1. Detectar obstáculos y medir distancia
2. Detectar objetos en una habitación
3. Detección de texto
4. Detección de billetes
5. Detección de colores
6. Detección e identificación de personas

B. Algoritmo Python

C. Algoritmo ESP32-CAM

D. Algoritmo ESP32-WROOM

E. Diagrama de flujo

F. Modelo 3d del soporte

G. Vídeos del prototipo

Referencias

[Berzal, 2019] Berzal, F. (2019). *Redes neuronales & deep learning: Volumen II*. Independently published.

[BR, 2010] BR (2010). Pessoas-com-deficiencia. `https://educa.ibge.gov.br/jovens/conheca-o-brasil/populacao/20551-pessoas-com-deficiencia.html`. Censo.

[Brandwatch, 2019] Brandwatch (2019). 126 amazing social media statistics and facts. `https://www.indec.gob.ar/indec/web/Nivel4-CensoNacional-3-2-Censo-2010s`. Medicion de imagen en facebook e instagram por dia.

[de Vigo, 2018] de Vigo, U. (2018). Atlas de histología vegetal y animal tipos celulares neurona. `https://mmegias.webs.uvigo.es/descargas/tipos-cel-neurona.pdf`. Departamento de Biología Funcional y Ciencias de la Salud. Facultad de Biología.

[dev47apps, 2021] dev47apps (2021). Droidcam. `https://www.dev47apps.com/`. DroidCam.

[Espressif, 2021] Espressif (2021). Camera web server. `https://github.com/espressif/arduino-esp32/tree/master/libraries/ESP32/examples/Camera/CameraWebServer`. Algoritmo CamaraWebServer para Esp32-Cam-Arduino.

[eyesynth, 2021] eyesynth (2021). Smartglasses para invidentes. `https://eyesynth.com/`. EXPERIENCIA DE SENTIDO AUMENTADO.

[Fukushima, 1975] Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3):121–136.

[Fukushima and Miyake, 1982] Fukushima, K. and Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer.

[Haykin and Network, 2004] Haykin, S. and Network, N. (2004). A comprehensive foundation. *Neural networks*, 2(2004):41.

[Hilera González et al., 2000] Hilera González, J. R., Martínez Hernando, V. J., et al. (2000). *Redes neuronales artificiales: fundamentos, modelos y aplicaciones*.

[Huang et al., 2017] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

[Ibáñez, 2014] Ibáñez, J. (2014). Niveles de madurez de la tecnología technology readiness levels. trls. una introducción. *Revista Economia Industrial*, 393:165–170.

[IrisVision, 2021] IrisVision (2021). Irisvision. `https://irisvision.com/orcam/`. Make the Most of Your Remaining Vision.

[joris March, 2013] joris March (2013). Google glasses. `https://www.thingiverse.com/thing:65706`. licensed under theCreative Commons - Attributionlicense.

[LadyofHats, 2007] LadyofHats (2007). Complete neuron cell diagram. [Online; acceso 18-8-2021].

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[Nowforever, 2019] Nowforever (2019). Esp32-cam. `https://commons.wikimedia.org/wiki/File:ESP32-CAM.jpg`. [Online; acceso Agosto 19, 2021].

[OMS, 2019] OMS (2019). La oms presenta el primer informe mundial sobre la visión. `https://www.who.int/es/news/item/08-10-2019-who-launches-first-world-report-on-vision`. Ceguera y discapacidad visual.

[OrCam, 2021] OrCam (2021). Orcam. `https://www.orcam.com/es/`. OrCam.

[Puyuelo, 2002] Puyuelo, Francisco Javier, C. B. J. (2002). Sobre lentes, espejuelos, anteojos, gafas o antiparras. *Archivos de la Sociedad Española de Oftalmología*, 77(12):689–691.

[REDACCIÓN180, 2012] REDACCIÓN180 (2012). Censo sorpresa por alto número de discapacitados. `https://www.180.com.uy/articulo/28890_Censosorpresapor-alto-numero-de-discapacitados`. Censo.

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

[Redmon and Farhadi, 2018] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99.

[Rosebrock, 2005] Rosebrock, A. (2005). Intersection over union (iou) for object detection (2016). *URL https://www. pyimagesearch. com/2016/11/07/intersection-over-unioniou-for-object-detection*.

[Sucar and Gómez, 2011] Sucar, L. E. and Gómez, G. (2011). Visión computacional. *Instituto Nacional de Astrofísica, Optica y Electrónica. Puebla, México*.

[team, 2021] team, O. (2021). Vl53l1x. `https://opencv.org`. OpenCV.

[Turing, 1948] Turing, A. (1948). Intelligent machinery. `http://www.alanturing.net/intelligent_machinery/`. bot de Twitter para fomentar el uso de textos alternativos.

[Ubahnverleih, 2018] Ubahnverleih (2018). Deutsch: Esp32 espressif esp-wroom-32 dev board. `https://commons.wikimedia.org/wiki/File:ESP32_Espressif_ESP-WROOM-32_Dev_Board.jpg`. [Online; acceso Agosto 19, 2021].

[Wang et al.,] Wang, C., Liao, H., Yeh, I., Wu, Y., Chen, P., and Hsieh, J. Cspnet: A new backbone that can enhance learning capability of cnn. arxiv 2019. *arXiv preprint arXiv:1911.11929*.