U-VISIÓN

Alejandro Angulo Llorente
Javier Gangoso Álvarez
Alberto Tato Gómez
Patricia Sánchez Mercado
Beatriz Paula Vallecillo Martín
Mauricio Vilanova

U-VISIÓN

Índice

1. Introducción

¿Qué es?

U-VISIÓN se trata de un sistema electrónico que brinda a las personas invidentes o con escasa visión, la posibilidad de detectar obstáculos cercanos a la altura de su cabeza y así evitarlos.

Consiste en unas gafas equipadas con un sensor de ultrasonidos y una pulsera inteligente que comienza a vibrar cuando se está en presencia de un obstáculo. Asimismo, se ha contemplado dotar al sistema de un buzzer para producir una señal de alerta audible como otra forma de aviso según la preferencia del usuario.

Este sistema sería complementario a los recursos que actualmente utilizan las personas invidentes para caminar fuera de casa, como el bastón o un perro guía.

2. Análisis de Mercado

El objetivo de este apartado es demostrar la viabilidad comercial del proyecto. Para ello es necesario determinar el ámbito geográfico, agrupar el mercado en grupos homogéneos o segmentos con el mismo perfil de cliente, cuantificar el mercado potencial, y calcular su demanda potencial y de ventas, y, en la medida de lo posible una reflexión sobre la evolución futura que se espera del mercado. Por último, se incluirá un análisis DAFO.

a) Ámbito geográfico

Para esta primera etapa del proyecto, se hace un análisis de mercado en **España**.

b) Perfil de cliente

Nuestro producto está dirigido a **personas adultas invidentes o con deficiencias visuales graves** que, aún con la ayuda de un perro guía o un bastón, son susceptibles a recibir golpes con obstáculos a la altura de la cabeza.

c) Cuantificación del mercado potencial

Según datos estadísticos de la Organización Nacional de Ciegos Españoles (ONCE), en el 2017 tenían un total de 72.239 afiliados. De dichos afiliados, se conocen los siguientes porcentajes:

EVOLUCIÓN (1996-2017)		
	1996	**2017**
Afiliados	51.740	72.239
Hombres	52,3%	48,28%
Mujeres	47,7%	51,72%
0 -18 años	9,36%	6,04%
19 - 64 años	58,11%	48,96%
65 años en adelante	32,53%	45,00%
Con Ceguera*	30,23%	19,40%

Con Deficiencia Visual**	69,75%	80,60%

* Con Ceguera: personas que no ven nada en absoluto o sólo perciben luz
** Con Deficiencia Visual: personas que mantienen un resto visual cuantificable.
Como nuestro producto está orientado a personas adultas (16 años en adelante) con ceguera o deficiencias visuales considerables que caminen por la calle, de esos 72.239 afiliados se tiene que:

- el 6,04 % son niños y adolescentes (población muy pequeña).
- el 19,40 % padece ceguera.
- del 80,60 % restante, que padecen deficiencia visual, se estima que la mitad tendrán una deficiencia que se vería beneficiada con el uso de nuestro sistema.

Con estos valores, necesitamos considerar un 60% de esos afiliados, al cual hay que restarle el porcentaje de niños y adolescentes con ceguera o patologías severas, aunque al ser una parte tan pequeña, podemos seguir considerando este 60% como una buena aproximación. Este cálculo arroja una cifra de poco más de **43.000** personas.

d) Demanda potencial y de ventas

Aquí se analizan las motivaciones y comportamientos de compra de los clientes y sus necesidades, a través de algunos interrogantes:

- ¿Qué necesidad resuelve?

Resuelve la necesidad que tienen las personas invidentes o con graves problemas de visión, de evitar golpearse con objetos que se encuentren a la altura de la cabeza.

- ¿Por qué, en qué momento y dónde satisface esa necesidad?

Es una necesidad a satisfacer porque es algo común que le sucede a este grupo de personas cuando caminan fuera de casa, principalmente al hacerlo en la acera o en espacios públicos donde haya otras personas o elementos que sean un obstáculo para ellos. Al ser obstáculos que no siempre se puedan advertir con el bastón o el perro guía, este sistema cubrirá esa vulnerabilidad que de otra manera no se podría evitar.

- ¿Cómo se satisface la necesidad?

Con el simple hecho de que el usuario se coloque las gafas (y eventualmente la pulsera) y encienda el sistema, podrá hacer uso del mismo.

- ¿Qué necesidades no cubre el producto que se ofrece?

Al ser un dispositivo que sólo tiene un campo de visión limitado, no puede suplantar totalmente al bastón o al perro guía. Por lo tanto, su uso es complementario. Tampoco posee un sistema de aviso de incidencia a otras personas (como un botón de pánico), pero es algo que podría desarrollarse a futuro.

Considerando la finalidad práctica de este producto y la importante ayuda añadida para el usuario, se espera una alta demanda del mismo. En cuanto a las ventas, se planea fabricar un lote limitado para estudiarlas más en detalle, y así poder hacer una mejor estimación. Se comenzará con la producción de **1.000 unidades** (500 de cada modelo), que constituyen **menos del 2,5%** de todo el mercado potencial en España.

e) Evolución futura estimada

Según las estadísticas, la ceguera y las afecciones visuales severas tienden a disminuir, debido a los grandes avances de los tratamientos médicos. Por lo tanto, para mantener un ritmo relativamente constante de ventas, se deben fabricar en cantidades limitadas, y así ir cubriendo paulatinamente la demanda.

f) Análisis DAFO

El análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) tiene el objetivo de concretar la evaluación de los puntos fuertes y débiles de la empresa (capacidad para generar y sostener sus ventajas competitivas), y las amenazas y oportunidades externas. Esto permite crear una estrategia que logre un adecuado ajuste entre su capacidad interna y su posición competitiva externa.

Interno	**Externo**
Debilidades -Escasa experiencia empresarial del equipo. -Dificultad para conseguir financiación.	Amenazas -Productos similares en desarrollo, que podrían ofrecer mejores prestaciones y diseño.
Fortalezas -Alta formación en ingeniería. -Producto sencillo y funcional, a bajo coste. -Producto innovador.	Oportunidades -Poca competencia en el sector. -Sector poco explotado. -Ausencia de productos similares en el mercado.

3. Prototipo

El prototipo consta de dos partes:

- Gafas de detección, procesamiento, y eventual alerta sonora
- Pulsera de alerta vibratoria

Ambas se comunican vía Bluetooth Low Energy, protocolo de comunicación elegido por su simpleza de implementación, bajo consumo de energía, y venir incorporado en ciertos módulos de procesamiento.

Gráficamente, el sistema completo es:

3.1 Gafas

De acuerdo al planteamiento, las gafas poseen un sensor de detección de obstáculos, un módulo de procesamiento y transmisión Bluetooth LE, y un altavoz de alerta sonora.

Como los obstáculos que se pretenden detectar con esta versión prototipo son los que están de cara al usuario, se ha decidido ubicar el sensor en el centro de la parte frontal de las gafas. Éste recoge la distancia a la que se encuentran los obstáculos y la envía al módulo principal, que será el encargado de procesar las señales procedentes del sensor, generar las órdenes de actuación adecuadas para que el altavoz emita sonidos, y transmitir el mensaje correspondiente, vía Bluetooth, a la pulsera.

Tras hacer pruebas con diferentes tipos de sensores (infrarrojo, ultrasonido, lidar), se observó un mejor comportamiento con el de ultrasonido. Por tanto, se eligió el modelo LV-MaxSonar-EZ, que tiene un rango de medida de aproximadamente 0,20 a 6,45 metros.

Al principio, para obtener la distancia que medía el sensor se utilizó un módulo Arduino Nano por su simplicidad. Una vez comprobado que las medidas eran correctas, como se necesitaba un módulo con conexión para enviar los datos obtenidos del sensor a la pulsera, se optó por utilizar Bluetooth. Se propuso utilizar el módulo ESP32-WROOM por incluir Bluetooth Low Energy, que se caracteriza por su bajo consumo, lo que permite ahorrar energía y alargar la duración de las baterías que alimentan el sistema. Finalmente se escogió la placa SparkFun ESP32 Thing, que también posee Bluetooth Low Energy pero, a diferencia de la anterior, ésta incluye un conector JST hembra que permite conectar directamente la batería, la que además se pueden cargar mediante dicha placa.

Respecto al altavoz de alerta sonora, se considera una buena opción un buzzer (zumbador) piezoeléctrico, ya que estos altavoces son sencillos, baratos y capaces de radiar con muy poca potencia eléctrica. Además, al ser pequeños, pueden colocarse lo suficientemente ocultos junto con el módulo Bluetooth, en una pequeña caja ubicada en una patilla de las gafas.

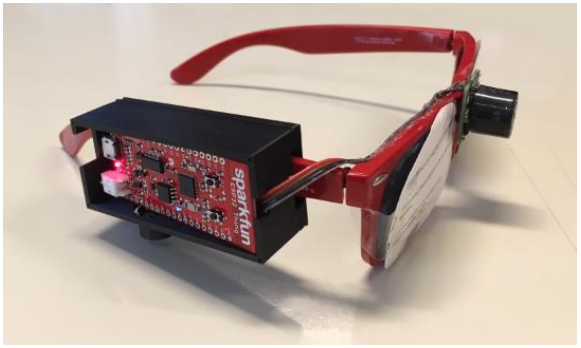
Consumo y alimentación

El consumo del SparkFun ESP32 Thing junto al sensor de ultrasonidos y el buzzer, da un valor aproximado de entre 125 mA y 135 mA.

Para alimentar este subsistema, se ha optado por emplear una batería tipo LiPo (polímero de iones de litio), por su ventaja de ser recargables (pueden llegar a más de 300 ciclos de carga y descarga), ligeras y poder almacenar gran cantidad de energía.

Para el prototipo se ha seleccionado una batería LiPo de 200 mAh, con la que se estima una duración media aproximada de 1 hora y media. Aunque la duración es corta, hemos decidido emplear esta batería de 200 mAh porque al ir incorporada en la montura de las gafas, tiene que ser lo más ligera posible, y para una primera demostración del producto es suficiente.

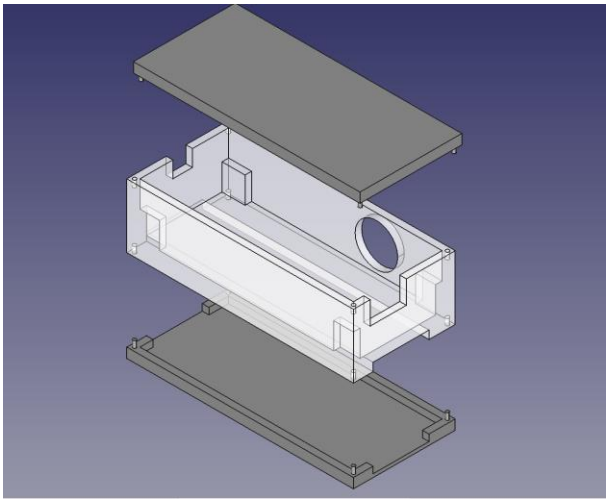
Hardware



Los componentes utilizados en las gafas para el prototipo final son los siguientes:

- Módulo SparkFun ESP32 Thing (Bluetooth)
- Sensor de ultrasonido LV-MaxSonar-EZ
- Buzzer (Zumbador piezoeléctrico) RS Pro 86dB
- Batería LiPo 3.7 V, 200 mAh

Por otra parte, para que el diseño fuera más estético y poder sujetar el módulo SparkFun con el buzzer y la batería, hemos hecho una impresión 3D de una cajita para este fin, y que se pudiera colocar en las gafas. Realizado con el programa FreeCad.



Software

El programa principal ha sido desarrollado utilizando el IDE de Arduino (configurado para su uso con ESP32) y basándose en ejemplos o librerías de código abierto que posteriormente han sido modificadas para cumplir con nuestros requisitos.

A continuación se nombran las fuentes principales:

- Ejemplo *Single Sensor Pulse Width for LV-MaxSonar* → MaxBotix Inc.
- Librería *ESP32_BLE_Arduino* → Neil Kolban (GitHub).

En el caso de las gafas, el módulo ESP32 será el encargado de leer los valores del sensor, procesarlos y activar o desactivar el buzzer en función de dichos valores.

Se ha programado para que el buzzer comience a avisar cuando se detecte un obstáculo a partir de 2 metros, y a medida que el usuario se vaya acercando, irá aumentando la frecuencia de aviso.

Además, en caso de requerir de una conexión BLE, éste módulo realiza la función de servidor, anunciándose con el nombre de "ESP32", encargado de enviar los valores leídos por el sensor en el formato adecuado.

El código completo puede ser consultado en el *Anexo I*.

3.2 Pulsera

La pulsera contiene un módulo de procesamiento y recepción Bluetooth LE igual que el de las gafas, y un motor de alerta vibratoria.

El módulo Bluetooth recibe los valores de distancia que le envía el transmisor de las gafas, y a partir de éstos controlará al vibrador, incrementando la frecuencia de vibración según el usuario se vaya aproximando al obstáculo.

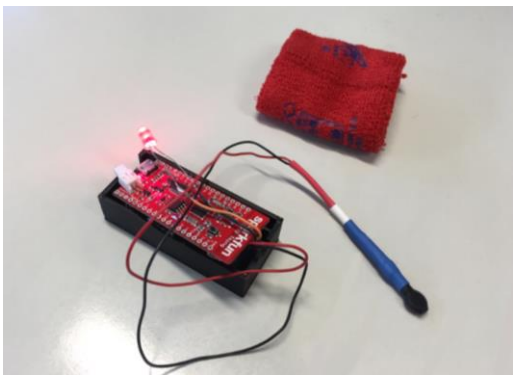
El motor vibrador cambia su período de actuación en función de la distancia a la que se encuentra el obstáculo, al igual que el buzzer situado en las gafas.

Consumo y alimentación

El consumo del SparkFun ESP32 Thing junto al motor vibrador, da un valor aproximado de entre 200 mA y 210 mA.

Nuevamente se ha optado por emplear una batería tipo LiPo para alimentar este subsistema. Para el prototipo se ha seleccionado una batería de 750 mAh, con la que se estima una duración media aproximada de 3 horas y media. Al igual que en las gafas, se decidió utilizar una batería lo más pequeña posible que cumpla con la capacidad mínima necesaria para que el prototipo funcione.

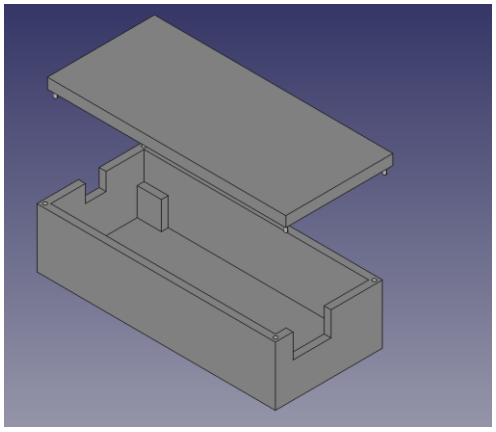
Hardware



Los componentes utilizados en la pulsera para el prototipo final son los siguientes:

- Módulo SparkFun ESP32 Thing (Bluetooth)
- Motor vibrador
- Batería LiPo 3.7 V, 750 mAh

Al igual que para las gafas, diseñamos una cajita en FreeCad y la imprimimos en la impresora 3D. En esta sólo pusimos el módulo Sparkfun junto con la batería y la unimos a la muñequera.



Software

Al igual que en el caso de las gafas, para el módulo de la pulsera se ha desarrollado un programa basándose en la librería desarrollada por Neil Kolban; en particular el ejemplo *BLE_client*.

En este caso, la pulsera hace la función de cliente realizando una búsqueda cada 30 segundos de los dispositivos BLE disponibles y conectándose automáticamente a aquel para el que se haya configurado. Una vez conectado, el módulo procesa los valores recibidos y activa o desactiva el vibrador en función de ellos.

Al igual que en el caso del buzzer, éste comenzará a vibrar cuando se detecte un obstáculo a partir también de 2 metros e incrementará la frecuencia de vibración a medida que se vaya aproximando a él.

El código completo puede ser consultado en el *Anexo I*.

4. Producto final

Ofrecemos dos opciones de producto según la preferencia del usuario, ambos con las gafas como base. La diferencia entre ellos es la forma de aviso cuando hay un obstáculo, que serían:

- Aviso mediante un *buzzer*
- Aviso mediante vibración con una *smartband*

U-Visión: Aviso mediante un *buzzer*

Para la opción del sistema de aviso mediante un *buzzer*, el producto estará formado únicamente por las gafas, las cuales llevarán incorporado el sensor de ultrasonidos, la placa Sparkfun ESP32 Thing, el buzzer y la batería.

Todos los componentes en conjunto van incorporados en la montura, y el más pesado es la batería. Por este motivo, debe de ser lo más ligera posible pero manteniendo una capacidad adecuada para un mayor tiempo de funcionamiento, por lo que se utilizará una batería de 600 mAh, con la que se estima un tiempo medio de funcionamiento de 5 horas, al ser el consumo de 125 mA aproximadamente.

Como tenemos que incluir estos componentes en la estructura de las gafas, vamos a realizar el diseño de las mismas y después contactaremos con una empresa para que lleve a cabo la fabricación. Posteriormente, nosotros incluiremos los componentes dentro de la montura de manera que queden completamente ocultos, y así quede un diseño estético y discreto.

Esta opción se compone de lo siguiente:

- Gafas
- Sensor ultrasonidos: LV-MaxSonar-EZ
- Buzzer (Zumbador piezoeléctrico) RS Pro 86dB
- Módulo BlueTooth®: Sparkfun ESP32 Thing
- Batería Li-Po 3,7V 600mAh

U-Visión Plus: Aviso mediante vibración con una *smartband*

Este producto se corresponde con la idea original de avisar al usuario mediante una pulsera que vibre ante la presencia de un obstáculo, para así dejar el oído libre.

Estará formado por las gafas anteriormente explicadas, excluyendo el componente buzzer y una pulsera programable. Respecto a ésta, consideramos que tiene que ser lo más ligera, cómoda y estética posible. Por ello, hemos decidido utilizar una *smartband* con software libre, programándola para que se conecte mediante Bluetooth LE con el módulo SparkFun y que vibre aumentando su tiempo de operación en función de la distancia a la que se encuentre el usuario del obstáculo.

En este caso la duración de funcionamiento de las gafas será de 6 horas ya que al no disponer del buzzer, el consumo del sensor y la placa SparkFun es de 100 mA, y la batería incorporada suministra 600 mAh. La smartband tendrá su propia batería de fábrica incorporada con un consumo específico, y la misma se cargará de forma independiente.

Esta opción se compone de lo siguiente:

- Gafas
- Sensor ultrasonidos: LV-MaxSonar-EZ
- Módulo Bluetooth: Sparkfun ESP32 Thing
- Batería 3,7 V 600 mAh
- *Smartband* con BlueTooth®

Por otra parte, también se considera un producto añadido:

Los clientes que compren como primera opción las gafas con aviso mediante buzzer, podrán “añadir” la funcionalidad de la pulsera. Es decir, con las mismas gafas podrán utilizar la pulsera como método de aviso, pero tendrían que actualizar el código del microcontrolador, ya sea obteniéndolo de nuestro repositorio de GitHub o bien mandándonos las gafas para que nosotros lo actualicemos.

5. Proveedores

Producto	Proveedor	Unidades	Precio total
Gafas	*	1000	10.000,00 €
Sensor ultrasonidos LV-MaxSonar-EZ1	RS Components	1000	25.700,00 €
Cables	Electrónica Embajadores	200 [metros]**	130,68 €
Batería LiPo	Farnell	1000	13.810,00 €
Cable micro-USB	DX	1000	1340 €
Sparkfun ESP32 Thing	Digikey Electronics	1000	16.380,00 €
U-Visión			
Zumbador Piezoeléctrico RS Pro	RS Components	500	885,00 €
U-Visión Plus			
Smartband	AliExpress	500	7.036,81 €

*Nosotros diseñamos la montura de las gafas, y encargamos a una empresa externa su fabricación. Estimamos que esto tendrá un coste de 10 € por cada unidad.

**Estimamos que 1 metro de cable nos sirve, aproximadamente, para 5 modelos finales por lo que cada modelo necesita 0,20m de cable.

6. Montaje, Distribución y Ventas

Hay que considerar el montaje y embalaje. Hemos decidido encargarnos nosotros del montaje y embalaje de los productos finales, puesto que esto nos permitirá ahorrar costes y obtener mayor rentabilidad, algo deseable en los primeros años tras la salida del producto.

En la caja irán incluidas las gafas, la *smartband* (en caso de solicitarla), el cable USB para cargar la batería y un manual de usuario en braille y en escritura alfabética. Por tanto, calculamos que para el embalaje se necesitará una caja de 20x14x7,5 cm, que además podremos personalizar para incluir el logotipo de la marca. Realizando una compra de 1000 unidades, el coste ascendería a 560,00 €.

Empresa proveedora: Rajapack
Referencia del producto: **FSRPT1101C**

En cuanto a la distribución de nuestro producto, una de las opciones planteadas es DHL Express, ya que nos ofrece gran seguridad en los envíos y devoluciones, así como puntos de recogida para el beneficio de nuestros clientes. Nuestro primer paso sería ponernos en contacto con ellos ya que ofrecen un presupuesto personalizado para cada empresa.

Finalmente, hemos decidido escoger **Amazon Services**, puesto que además de la seguridad en los envíos que incluye DHL Express, ofrece en su servicio el almacenaje y el envío. Además, el renombre de la empresa y sus ayudas para aparecer como productos destacados puede aumentar el volumen de ventas. El coste de este servicio es de **26€** para los meses entre Enero a Septiembre y de **36€** para los meses entre Octubre y Diciembre.

7. Coste y Precio de Venta

De acuerdo a los costes estimados en apartados anteriores (costes de material, de montaje y embalaje y costes de distribución), obtenemos los siguientes costes por unidad:

	U-Visión	U-Visión Plus
Coste de material	69,81 €	81,43 €
Coste de embalaje	0,56 €	0,56 €
Coste de distribución	2,79 €	2,79 €
Coste total	73,16 €	84,78 €
Precio de venta	89,99 €	109,99 €

*Coste de producción para **1.000** unidades (500 unidades de cada tipo).

Además se le ofrece al cliente que originalmente hubiese comprado las gafas con el buzzer la posibilidad de “mejorar” el producto y pasar del modelo estándar al modelo Plus, es decir, añadir la funcionalidad de la *smartband*. Este servicio tendría un coste de **35** € y consistiría en la actualización del código del ESP32 y el añadido de la pulsera ya lista para su uso.

8. Publicidad y Marketing

Para dar a conocer U-Visión a un mayor número de personas, vamos a contactar con varios **centros médicos** (concretamente con el área de Oftalmología) ofreciéndoles unas muestras de nuestro producto a los pacientes con escasa capacidad visual para que puedan probarlos.

Por otra parte, nos pondremos en contacto con la **ONCE** (Organización Nacional de Ciegos Españoles) para dar a conocer nuestro producto mediante correo postal o electrónico, a personas invidentes o con escasa visión afiliadas a esta asociación. También les daríamos algunas muestras de nuestros productos para que las personas interesadas puedan probarlo *in situ*.

Además, hemos creado una **página web**. En ella se explica por qué empezamos con este proyecto, cómo fue el desarrollo del prototipo y los productos que ofertamos. La web es:

https://uvision.wixsite.com/uvision

9. Conclusión

Tras el desarrollo final del proyecto, se ha comprobado que al momento de realizar un prototipo es fundamental contar con un amplio espectro de posibles soluciones para intentar resolver un problema determinado. En nuestro caso, nos dimos cuenta de que había que probar varios sensores ya que no todos funcionaban como se esperaba o no se ajustaban a las características que necesitábamos, hasta que escogimos el definitivo.
Por otra parte, lo más complicado fue implementar el software ya que en un principio no se comportaba de la manera que esperábamos, detectando falsos positivos y no avisando al usuario a tiempo para esquivar los obstáculos. Esto supuso una etapa de "prueba y error" haciendo cambios y ajustando el código según los datos que obteníamos hasta que conseguimos que el prototipo funcionara correctamente.

También es importante contar con varias opciones de proveedores, porque no todos tienen el dispositivo que se necesita, o el precio es demasiado elevado, o el tiempo de envío es muy prolongado.

En cuanto al producto, requiere un proceso de fabricación bastante sencillo, ya que sus componentes se pueden adquirir por separado y en el caso de las gafas, el montaje de los componentes sería fácil de realizar. El hecho de que sean ampliamente difundidos y basados en hardware y software libres, hace que los costes de producción sean lo suficientemente reducidos.

Por último, en cuanto a los potenciales clientes, consideramos que es una solución que puede ayudar a muchas personas invidentes a caminar con mayor seguridad por la ciudad, con un precio asequible de venta y sencillo de utilizar.

10. Bibliografía

-Organización Nacional de Ciegos Españoles (ONCE).
https://www.once.es/dejanos-ayudarte/afiliacion/datos-estadisticos

-Desglose de tarifas en Amazon Services
https://services.amazon.es/servicios/logistica-de-amazon/precios.html

Prototipo

Gafas

- ESP 32 WROOM

https://www.digikey.es/product-detail/es/espressif-systems/ESP32-DEVKITC/1904-1011-ND/8544299?utm_adgroup=RF+Boards&mkwid=s&pcrid=241858430831&pkw=&pmt=&pdv=c&productid={productid}&&gclid=Cj0KCQjw28_XBRDhARIsAEk21Fg1oNLh8xS5bdcV6eeA7xNhuhyTOtECQ1PhPQ0jXq8S2z2lNUhTjAcaAg56EALw_wcB

- Sensor ultrasonidos: LV-MaxSonar-EZ

http://tienda.bricogeek.com/sensores-distancia/40-sensor-de-proximidad-por-ultrasonidos-lv-ez3.html

- Buzzer

http://tienda.bricogeek.com/componentes/299-zumbador-piezo-12mm.html

- Batería LiPo 3.7 V, 200 mAh

http://tienda.bricogeek.com/baterias-lipo/940-bateria-lipo-200mah-37v.html 4,95€

Pulsera

- ESP 32 WROOM
- Motor vibrador
- Batería LiPo 3.7 V, 750 mAh

https://www.ebay.es/itm/BATER-A-LiPo-3-7V-750mAh-20C-DRONE-MJX-X200-X400-X500-X800-X300C-JXD-509-FY550/222830488051?hash=item33e1bb69f3:m:mIK81ha2KK5G50f1jnSAy9A

Producto final

Producto 1

- Gafas

https://es.aliexpress.com/store/product/Large-sunglasses-polarized-sunglasses-driving-glasses-classic-sunglasses/1093612_1616273277.html?spm=a219c.search0104.3.152.46dc1950S8Z2u7&ws_ab_test=searchweb0_0,searchweb201602_1_10152_10151_10065_10344_10068_10342_10547_10343_5722611_10340_10548_10341_10696_5722911_5722811_5722711_10084_10083_10618_10307_10301_10303_5711211_10059_308_100031_10103_10624_10623_10622_10621_10620_5711311_5722511-10620,searchweb201603_25,ppcSwitch_5&algo_expid=6635e78d-0b5a-4bb8-a02d-c77b0abbcc33-21&algo_pvid=6635e78d-0b5a-4bb8-a02d-c77b0abbcc33&priceBeautifyAB=0

- Módulo Bluetooth: Sparkfun ESP32 thing

https://www.digikey.es/product-detail/es/sparkfun-electronics/DEV-13907/1568-1444-ND/6419476?utm_adgroup=RF+Boards&mkwid=s&pcrid=241858430831&pkw=&pmt=&pdv=c&productid={productid}&&gclid=CjwKCAjw8r_XBRBkEiwAjWGLlI_ZVB3JKGZi5aX0letqJY5MS9vWTFP_e9OVAoMv6FStudSrnxEK0BoCU_cQAvD_BwE

- Pulsera programable

https://es.aliexpress.com/item/ID115-is-the-new-arrival-of-LUOKA-Store-Simple-and-fashion-design-5-colors-for-your/32795852007.html?spm=a219c.search0104.8.19.4ca65be7j968MC&priceBeautifyAB=0

- Cable Micro USB - USB

http://www.dx.com/es/p/micro-usb-5-pin-to-usb-2-0-short-data-sync-charging-cable-for-cellphone-tablet-pc-white-22cm-416112?tc=EUR&ta=ES&gclid=Cj0KCQjw5-TXBRCHARIsANLixNw3qucG_BlDlWySqNnqy09NxueRpmjYGy3R5CpqL8tN7EEdWr_BFVoaApe7EALw_wcB#.WvmXbtaxW18

Producto 2

- Batería LiPo 3,7V 600mAh

http://es.farnell.com/bak/lp-443440-1s-3/battery-lithium-pol-3-7v-0-6ah/dp/2077883

- Caja para envío

https://www.rajapack.es/cajas-carton-contenedores-cajas-postales/cajas-postales-personalizadas/cajas-blancas-1-color_skuFSRPT1101C.html

- Diseño de gafas 3D

ANEXO 1: CÓDIGOS CARGADOS EN EL ESP32

GAFAS

```
/** ELCO Project -- uVision
 *  Code for the ESP32 that will go on the glasses
 *  Features:
 *   - BLE Server
 *   - Sensor Read
 *   - Buzzer Control
 */

#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
#include <BLE2902.h>

#define SERVICE_UUID        "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"

const int pinSensor = 34; // GPIO number. See ESP32 board pinouts
const int pinBuzz = 18; // GPIO number. See ESP32 board pinouts
int pulse1, sensorValue; // Variables
uint8_t txValue = 0; // BL-->TX
BLECharacteristic *pCharacteristic;

// Para antirebotes
/*
const int timeThreshold = 150;
const int intPin = 2;
volatile int ISRCounter = 0;
int counter = 0;
long timeCounter = 0;
bool encendido = false;
*/
void setup () {


  //if digital read pin boton = 1 then encendido = not(encendido) ... if encendido funcion distancia else
nada
  Serial.begin(115200); //Serial communication
  pinMode(pinSensor, INPUT); //Sensor input
  pinMode(pinBuzz, OUTPUT); //Buzzer output
```

```
 /*
 //antirebote boton
 pinMode(intPin, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(intPin), debounceCount, FALLING);
 attachInterrupt(digitalPinToInterrupt(intPin), onoff, FALLING);
 */
 // Create the BLE Device
 BLEDevice::init("ESP32");

 // Create the BLE Server
 BLEServer *pServer = BLEDevice::createServer();

 // Create the BLE Service
 BLEService *pService = pServer->createService(SERVICE_UUID);

 // Create a BLE Characteristic
 pCharacteristic = pService->createCharacteristic(
                     CHARACTERISTIC_UUID,
                     BLECharacteristic::PROPERTY_NOTIFY
                   );

 pCharacteristic->addDescriptor(new BLE2902());

 // Start the service
 pService->start();

 // Start advertising
 BLEAdvertising *pAdvertising = pServer->getAdvertising();
 pAdvertising->addServiceUUID(pService->getUUID());
 pAdvertising->start();
 Serial.println("BLE Server defined!");
}

void loop () {
 //antirebote
 /*if (counter != ISRCounter)
  {
    counter = ISRCounter;
    //Serial.println(counter);
  }
 if (encendido = true)
  {*/
  read_sensor(); // Read the sensor value
  txValue = sensorValue / 4; // Value to send (8 bits, max 256)
  pCharacteristic->setValue(&txValue,1);
```

```
    pCharacteristic->notify();
    Serial.print("*** Sent Value: ");
    Serial.print(txValue);
    Serial.println(" ***");

    distance(sensorValue);


    delay(10); // This delay time changes by 50 for every sensor in the chain.  For 5 sensors this will be
250
    //}
  // else {}
}

// Function to read the ultrasonic sensor value every loop
void read_sensor() {

  pulse1 = pulseIn(pinSensor, HIGH); // Measurement for PW input in uS
  sensorValue = 2.54 * (pulse1 / 147); // Inch to cm conversion

}
/**
 * Funcion encendido/apagado
 */
 /*
void onoff()
{
  encendido = not(encendido);
}

/**
 * Funcion antirebote boton encendido/apagado

void debounceCount()
{
   if (millis() > timeCounter + timeThreshold)
   {
      ISRCounter++;
      timeCounter = millis();
   }
}
*/
/**
 * Buzzer control function
 */
```

```
void distance(int measure) {

  if (measure > 300) {
    Serial.print("No hay peligro:  ");
    Serial.print(measure);
    Serial.println(" ");
    digitalWrite(pinBuzz, LOW);

  } else if ((measure <= 300) && (measure > 200)) {
    Serial.print("Objeto detectado a:  ");
    Serial.print(measure);
    Serial.println(" cm ");

    digitalWrite(pinBuzz, HIGH);
    delay(600);
    digitalWrite(pinBuzz, LOW);


   } else if ((measure <= 200) && (measure > 150)) {
    Serial.print("Objeto cercano a:  ");
    Serial.print(measure);
    Serial.println(" cm ");

    digitalWrite(pinBuzz, HIGH);
    delay(450);
    digitalWrite(pinBuzz, LOW);


  } else if (measure <= 150 && (measure > 100)) {
    Serial.print("Cuidado, que te das en:  ");
    Serial.print(measure);
    Serial.println(" cm ");

    digitalWrite(pinBuzz, HIGH);  // Run in high speed
    delay(375);
    digitalWrite(pinBuzz, LOW);

  } else if (measure <= 100 ) {
    Serial.print("Parate boludooo que estas a:  ");
    Serial.print(measure);
    Serial.println(" cm ");

    digitalWrite(pinBuzz, HIGH);  // Run in high speed
    delay(200);
    digitalWrite(pinBuzz, LOW);
```

```
    }
    delay(measure);

} // End of distance
```

MUÑECA

```
/** ELCO Project -- uVision
 *  Code for the ESP32 that will go in the wrist
 *  Features:
 *   - BLE Client
 *   - Motor Control
 *   - Mode Selection
 */

#include "BLEDevice.h"

const int pinMotor = 18;  // Use GPIO number. See ESP32 board pinouts
int measure; // Variables
int modeSel = 1;

// The remote service we wish to connect to.
static BLEUUID serviceUUID("4fafc201-1fb5-459e-8fcc-c5c9c331914b");
// The characteristic of the remote service we are interested in.
static BLEUUID charUUID("beb5483e-36e1-4688-b7f5-ea07361b26a8");

static BLEAddress *pServerAddress;
static boolean doConnect = false;
static boolean connected = false;
static BLERemoteCharacteristic* pRemoteCharacteristic;

/**
 * Check for the BLE connection
 */
bool connectToServer(BLEAddress pAddress) {

    Serial.print("Forming a connection to ");
    Serial.println(pAddress.toString().c_str());

    BLEClient*  pClient  = BLEDevice::createClient();
```

```
  Serial.println(" - Created client");

  // Connect to the remove BLE Server.
  pClient->connect(pAddress);
  Serial.println(" - Connected to server");

  // Obtain a reference to the service we are after in the remote BLE server.
  BLERemoteService* pRemoteService = pClient->getService(serviceUUID);
  if (pRemoteService == nullptr) {
    Serial.print("Failed to find our service UUID: ");
    Serial.println(serviceUUID.toString().c_str());
    return false;
  }
  Serial.println(" - Found our service");

  // Obtain a reference to the characteristic in the service of the remote BLE server.
  pRemoteCharacteristic = pRemoteService->getCharacteristic(charUUID);
  if (pRemoteCharacteristic == nullptr) {
    Serial.print("Failed to find our characteristic UUID: ");
    Serial.println(charUUID.toString().c_str());
    return false;
  }
  Serial.println(" - Found our characteristic");

} // connectToServer

/**
 * Scan for BLE servers and find the first one that advertises the service we are looking for.
 */
class MyAdvertisedDeviceCallbacks: public BLEAdvertisedDeviceCallbacks {

  void onResult(BLEAdvertisedDevice advertisedDevice) { //Called for each advertising BLE server.

  Serial.print("BLE Advertised Device found: ");
  Serial.println(advertisedDevice.toString().c_str());

  // We have found a device, let us now see if it contains the service we are looking for.
  if (advertisedDevice.haveServiceUUID() &&
advertisedDevice.getServiceUUID().equals(serviceUUID)) {

    Serial.print("Found our device!  address: ");
    advertisedDevice.getScan()->stop();

    pServerAddress = new BLEAddress(advertisedDevice.getAddress());
    doConnect = true;
```

```
    } // Found our server
  } // onResult
}; // MyAdvertisedDeviceCallbacks

void scanner(){

  // Retrieve a Scanner and set the callback we want to use to be informed when we
  // have detected a new device. Specify that we want active scanning and start the
  // scan to run for 30 seconds.
  BLEScan* pBLEScan = BLEDevice::getScan();
  pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
  pBLEScan->setActiveScan(true);
  pBLEScan->start(10);


  } // scanner

/**
 * Setup
 */
void setup() {

  Serial.begin(115200); // Serial communication
  pinMode(pinMotor, OUTPUT); // Motor output

  // Create the BLE Device
  BLEDevice::init("");
  Serial.println("Starting BLE Client application...");

} // End of setup.


/**
 * Main loop function
 */
void loop() {

  // If the flag "doConnect" is true then we have scanned for and found the desired
  // BLE Server with which we wish to connect.  Now we connect to it.  Once we are
  // connected we set the connected flag to be true.
  if (doConnect == true) {
    if (connectToServer(*pServerAddress)) {
      Serial.println("We are now connected to the BLE Server.");
      connected = true;
```

```
    } else {
      Serial.println("We have failed to connect to the server; there is nothin more we will do.");
    }
    doConnect = false;
  } else if (connected == false) {
    scanner();
    }

  //If we are connected to a peer BLE Server, read the characteristic each time we are reached
  //with the current time since boot.
  if (connected) {
    uint8_t rxValue = pRemoteCharacteristic->readUInt8();
    measure = rxValue*4;
    distance(measure);
  }
  delay(10); // Delay between loops.
} // End of loop

/**
 * Motor control function
 */
void distance(int measure) {

    if (measure > 300) {
      Serial.print("No hay peligro:  ");
      Serial.print(measure);
      Serial.println(" ");
      digitalWrite(pinMotor, LOW);

    } else if ((measure <= 300) && (measure > 200)) {
      Serial.print("Objeto detectado a:  ");
      Serial.print(measure);
      Serial.println(" cm ");

      digitalWrite(pinMotor, HIGH);
      delay(600);
      digitalWrite(pinMotor, LOW);

     } else if ((measure <= 200) && (measure > 150)) {
      Serial.print("Objeto cercano a:  ");
      Serial.print(measure);
      Serial.println(" cm ");

      digitalWrite(pinMotor, HIGH);
      delay(450);
```

```
    digitalWrite(pinMotor, LOW);

   } else if (measure <= 150 && (measure > 100)) {
    Serial.print("Cuidado, que te das en:  ");
    Serial.print(measure);
    Serial.println(" cm ");

    digitalWrite(pinMotor, HIGH);  // Run in high speed
    delay(375);
    digitalWrite(pinMotor, LOW);

   } else if (measure <= 100 ) {
    Serial.print("Parate boludooo que estas a:  ");
    Serial.print(measure);
    Serial.println(" cm ");

    digitalWrite(pinMotor, HIGH);  // Run in high speed
    delay(200);
    digitalWrite(pinMotor, LOW);

   }
   delay(measure);

} // End of distance
```