Proyecto: “Ojo de Van Gogh”

Escuela: E.E.S.T N°7 “T.R.Q

Curso: 7mo 2da Aviónica

Año: 2020

Link a la página web: ojodevangogh.tech

Integrantes:

- CARDINAL, Facundo Julián – DNI 44.051.731 – 7mo 2da Aviónica
- CARRO, Nahuel Agustín – DNI 43.660.810 – 7mo 2da Aviónica
- FONTE, Gonzalo Juan – DNI 43.173.573 – 7mo 2da Aviónica
- PIERRI, Matias Gabriel – DNI 44.128.118 – 7mo 2da Aviónica
- VILARDO, Theo – DNI 43.245.429 – 7mo 2da Aviónica
- VITTORE, Tobías – DNI 42.680.475 – 7mo 2da Aviónica

Tutores:

- MEDINA, Sergio
- BIANCO, Carlos

Fecha de inicio:

- 1ro de marzo de 2020

Esfuerzo del proyecto:

- 24 horas semanales repartidas entre 6 personas
- 670 horas de trabajo en total
- 28 semanas de trabajo
- Tareas abarcadas:
  - Investigación de problemática
  - Desarrollo del producto
  - Difusión y establecimiento en redes sociales[1]
  - Creación de web page[2]

Distinciones:

- Nota científica en “MasScience” ONG España
- Nota periodística en “Inquietudes de Sur”
- Entrevista de radio en “Crecer FM 106.5”
- Entrevista de radio en “Fan FM 103.9”
- Entrevista de radio en “Del Este FM 99.3”
- Medalla de Oro en Prototipos III en las ONIET

Nuestro repositorio GitHub:

- https://github.com/impatrq/ojo_de_van_gogh

[1] Para ver nuestras redes sociales buscar *ojodevangogh.tech* en Instagram
[2] Para ver nuestra web page colocar el siguiente link *ojodevangogh.tech*

Índice:

1. Introducción

1.1. **Objetivo**:

El objetivo de este proyecto es brindar a la gente con discapacidad visual un dispositivo electrónico capaz de ayudarlo a leer textos, reconocer objetos y distinguir colores sin necesidad de que necesite pedir ayuda a terceros garantizándole autonomía y bienestar emocional.

Este proyecto aborda el tema de inclusión de nuevas tecnologías para mejorar la calidad de vida de personas con discapacidad visual. Apoyándonos en Inteligencia Artificial, creamos un sistema capaz de asistir al usuario sin necesidad de que este tenga que estar dependiendo de terceros.

Para cumplir este objetivo entrevistamos personas con ceguera, tanto de nacimiento o por alguna enfermedad, para generar un sistema a medida de sus necesidades.

1.2. **Descripción general**:

Nuestro sistema se compone de dos componentes principales:

- **Sistema de reconocimiento visual**: este va colocado en el antebrazo del usuario y se encarga de tomar la foto, procesarla y dar una respuesta a través de un auricular. Además, cuenta con pequeños motores que van a vibrar dependiendo de los colores predominantes de la foto para darle un feedback al usuario.
- **Sistema de lectura cerebral**: Este es un headset que tiene un sensor apoyado arriba de la ceja del usuario que permite monitorear su actividad cerebral. Cuando este genera un doble parpadeo intencional envía una señal al sistema de reconocimiento visual para sacar la foto.

Gracias a la implementación de un sensor cerebral permitimos conectar el sistema electrónico al usuario de manera no invasiva. En caso de que el usuario no se sienta cómodo con el sensor cerebral se puede implementar un botón o sensor muscular como disparador.

1.3. **Alcance**:

Las personas que sufren una discapacidad visual, total o casi total, que no les permite poder interactuar con el entorno haciendo uso del sentido de la vista es de aproximadamente 1.000.000 de personas en Argentina.

Nuestro foco se centra en Buenos Aires, específicamente en CABA y GBA Sur, ya que nos permite hacer un seguimiento efectivo de las personas que usen nuestro sistema, además de que se encuentran posibles aliados estratégicos como la Asociación de Ayuda al Ciego (ASAC).

1.4. **Segmento de usuario**:

El segmento de usuario que buscamos son personas con una discapacidad visual total o casi total que tengan una necesidad de poder manejarse de manera autónoma debido a que sienten frustraciones al no poder desenvolverse en su día

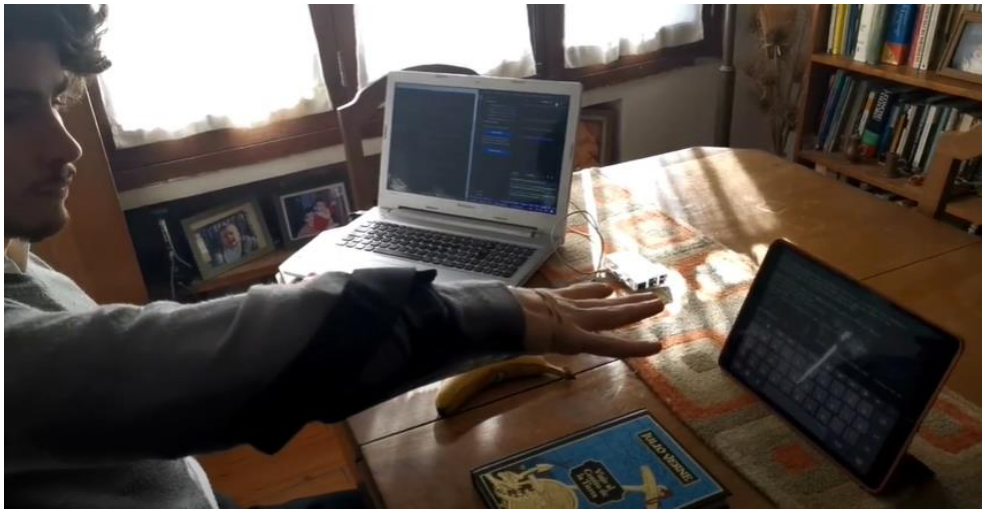
a día de manera cómoda ya que no poseen información visual de lo que está pasando.

Los beneficios que “Ojo de Van Gogh” otorga a sus usuarios son:

- Capacidad de leer textos que no están adaptados en braille. Por ejemplo: etiquetas de productos, libros, etc.
- Capacidad de reconocer objetos que no pueden distinguirse por su forma al tacto.
- Retroalimentación a la hora de interactuar con colores a través de vibraciones y sonidos. Esto se fundamenta en que las entrevistas con personas ciegas nos dijeron que les genera una gran angustia sentirse excluidos de la sociedad al no poder manejarse con colores.

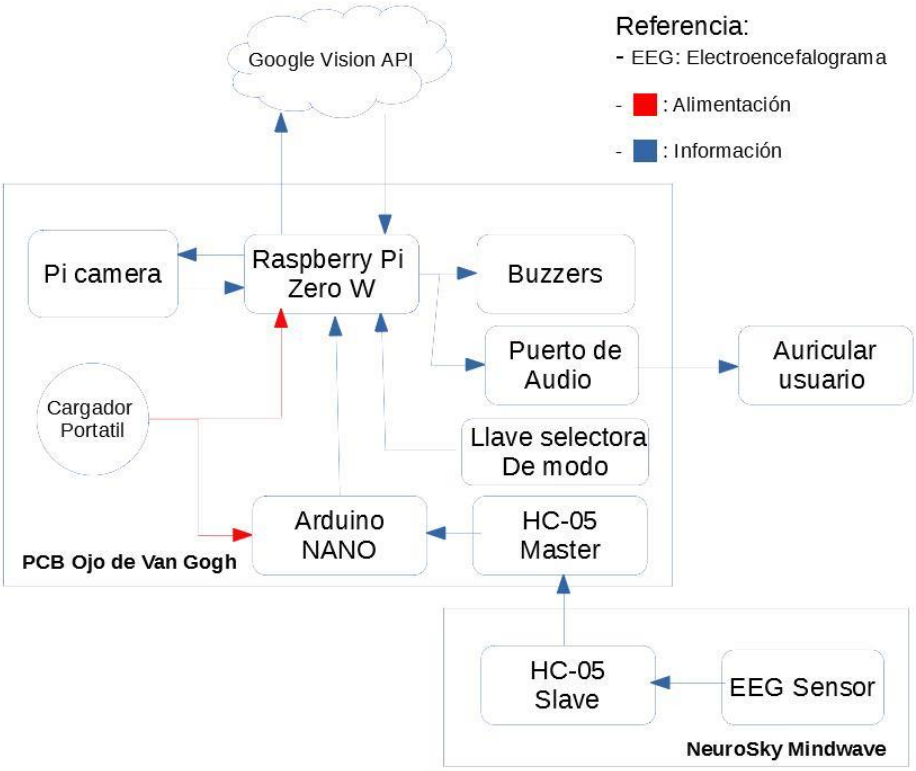
1.5. Captura del proyecto

Uno de los integrantes de nuestro equipo trabajando con el sistema de Reconocimiento de Colores, Textos y Objetos. En la mano tiene la cámara de la Raspberry Pi Zero W con la cual se saca las fotos para el reconocimiento.



Uno de los integrantes de nuestro equipo trabajando con el sistema de vibración de colores.

1.6. Diagrama en bloques



1.7. Resultado Conseguido

- Para reconocer objetos hemos usado como ejemplo, la siguiente imagen:

El resultado fue el siguiente:


```
main.py > ...
38          time.sleep(2)
39
40  elif(orden == "texto"):
41
42      text_manager = TextManager(IMAGE_PATH, CLIENT)
43
44      respuesta_texto_api = text_manager.get_response_api
45
46      texto = text_manager.get_text(respuesta_texto_api)
47
48      traducido = reproductor_audio.translate(texto, IDIOMA)
49      reproductor_audio.speak_audio(str(traducido))
50      time.sleep(2)
51
52  elif(orden == "objetos"):
53
54      object_manager = ObjectManager(IMAGE_PATH, CLIENT)
55
56      respuestas_objetos = object_manager.get_response_api()
57
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 2 1: Python

```
C:\Users\pc1\Google Drive\Cursos\ejemplos_python\proyecto>C:/Users/pc1/AppData/Local/Programs/Python/Python37/python.exe "c:/Users/pc1/Google Drive/Cursos/ejemplos_python/proyecto/main.py"
Ingrese la opcion: objetos
Plátano está en el centro derecha
```

- Para reconocer el color hemos usado como ejemplo, la siguiente imagen:

El resultado fue el siguiente:

```
25  reproductor_audio = TextoToAudio()
26
27  orden = input("Ingrese la opcion: ")
28
29  if(orden == "color"):
30
31      colors_manager = ColorsManager(IMAGE_PATH, CLIENT, DATAFRAME_COLOR)
32      datos_rgb_api = colors_manager.get_response_api()
33
34      for value_color in colors_manager.get_color(datos_rgb_api):
35
36          traducido = reproductor_audio.translate(value_color, IDIOMA)
37          print(traducido)
38          reproductor_audio.speak_audio(str(traducido))
39          time.sleep(2)
40
```

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS 2                    1: Python

C:\Users\pc1\Google Drive\Cursos\ejemplos_python\proyecto>C:/Users/pc1/AppData/Local/Programs/Python/Python37/python.exe "c:/Users/pc1/Google Drive/Cursos/eje
mplos_python/proyecto/main.py"
Ingrese la opcion: color
rojo
```

- Para reconocer el texto hemos usado como ejemplo, la siguiente imagen:

El resultado fue el siguiente:

```
main.py > ...
            reproductor_audio.speak_audio(str(traducido))
            time.sleep(2)

    elif(orden == "texto"):

        text_manager = TextManager(IMAGE_PATH, CLIENT)

        respuesta_texto_api = text_manager.get_response_api()

        texto = text_manager.get_text(respuesta_texto_api)

        traducido = reproductor_audio.translate(texto, IDIOMA)
        print(traducido)

        reproductor_audio.speak_audio(str(traducido))
        time.sleep(2)
```

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS 2        1: Python

C:\Users\pc1\Google Drive\Cursos\ejemplos_python\proyecto>C:/Users/pc1/AppData/Local/Programs/Python/Python37/python.exe "c:/Users/pc1/Google Drive/Cursos/ejemplos_python/proyecto/main.py"
Ingrese la opcion: texto
1000
BANCO CENTRAL DE LA
REPÚBLICA ARGENTINA
M
1000
MIL PESOS
HORNERO
AVE NACIONAL
```

2. Software

2.1. Información técnica:

Para lograr el objetivo que nos propusimos, decidimos utilizar una Raspberry Pi Zero W ya que nos permite trabajar con la Inteligencia Artificial de Google (Google Vision API) en el lenguaje Python.

Google Vision API nos ofrece:

- Detección de textos
- Detección de objectos
- Detección de colores predominantes en imagen (en RGB)

Estas características de Google Visión API nos permite tener una base sólida y no lidiar con el desarrollo de una IA desde cero.

Las imágenes se toman a través de una Pi Camera la cual está diseñada para funcionar perfectamente en una Pi Zero W, que cuenta con el puerto de la pi camera, haciendo uso de Raspbian OS.

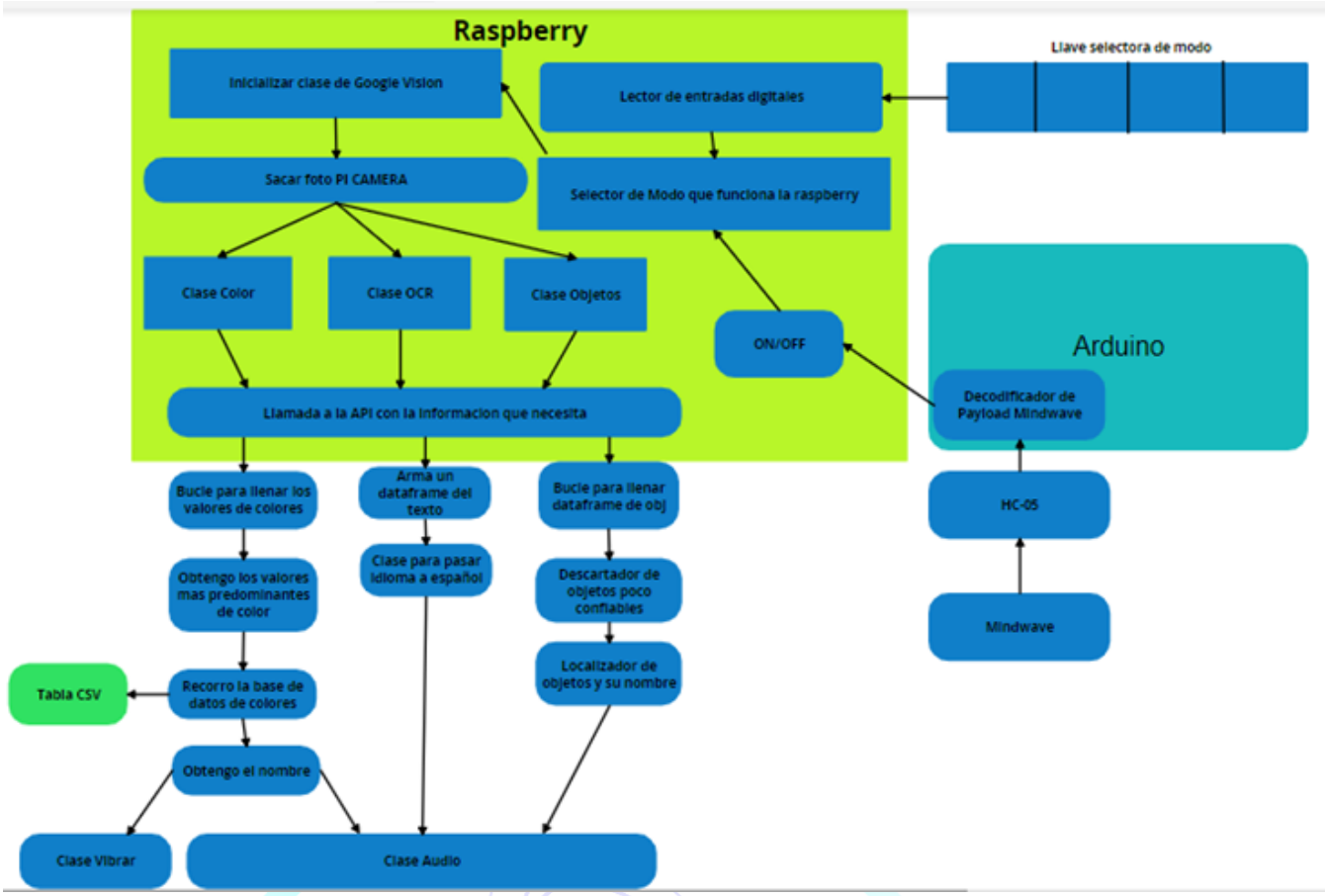
Una vez que tenemos la información hacemos uso de la librería “os” de Python que nos permite interactuar directamente con la terminal de Linux y controlar el puerto de audio para hacerle llegar la información a la persona mediante un auricular.

Todo esto es importante controlarlo y que se haga solo cuando el usuario quiera, para generar eso tenemos un sensor cerebral que se encarga de monitorear la actividad electromagnética del cerebro. En el momento que el usuario genera un doble parpadeo intencional envía un valor analógico mayor a 400 al Arduino NANO que nos permite procesar esa información para comunicarle a la Raspberry Pi Zero W que debe sacar una foto e iniciar el análisis.

Para que el usuario pueda elegir entre leer texto, reconocer objetos o detectar colores colocamos un switch de 3 posiciones donde se puede alternar que es lo que se busca saber.

2.2. Diagrama de Arquitectura

En este se definen las clases padres e hijas del código de reconocimiento de colores, textos y objetos. También se muestra las conexiones entre la Raspberry Pi Zero W, el Arduino Nano y Mindwave.



2.3. **Detección de texto:**

Para procesar la imagen y extraer los textos que tiene, conectamos a la API de Google Vision y le enviamos el contenido de la imagen usando JSON:

```
class TextManager(GoogleVisionEngine):

    def get_response_api(self, ):

        # Se lee la imagen
        with open(self.image_path, 'rb') as image:
            # Leemos las cosas de la imagen y lo guardamos en contenido
            content = image.read()
            # Enviamos la peticion a la API de google con el formato JSON
            # especifico optimizar la velocidad de respuesta
            response = self.client.annotate_image({'image': {'content': content}, 'features': [
                {'type': vision.enums.Feature.Type.TEXT_DETECTION}], })

            respuesta_texto_api = response.text_annotations
            return (respuesta_texto_api)
```

Luego que tenemos la respuesta de la API, debemos procesarla con Pandas para quedarnos con el texto puro ya que la respuesta va a llegar con datos, útiles únicamente para el desarrollador, que confundirían al usuario final.

```
def get_text(self, respuesta_texto_api):

    # Guardamos los valores de la api
    respuesta_texto_api = respuesta_texto_api

    # Creamos el dataframe con el texto y el idioma
    df_leer_texto = pd.DataFrame(columns=['description'])

    # Por cada texto detectado lo unimos al dataframe
    for text in respuesta_texto_api:
        df_leer_texto = df_leer_texto.append(
            dict(
                description=text.description
            ),
            ignore_index=True)

    # a la variable texto le metemos el contenido del texto
    texto_completo = (df_leer_texto['description'][0])

    return(texto_completo)
```

2.4. **Detección de objetos:**

Para procesar la imagen y extraer los objetos que tiene, conectamos a la API de Google Vision y le enviamos el contenido de la imagen usando JSON:

```
class ObjectManager(GoogleVisionEngine):

    def get_response_api(self, ):

        # Se lee la imagen
        with open(self.image_path, 'rb') as image:
            # Leemos las cosas de la imagen y lo guardamos en contenido
            content = image.read()
            # Enviamos la peticion a la API de google con el formato JSON
            # especifico optimizar la velocidad de respuesta
            response = self.client.annotate_image({'image': {'content': content}, 'features': [
                {'type': vision.enums.Feature.Type.OBJECT_LOCALIZATION}], })
            # Guardamos la respuesta de la API
            respuesta_objeto_api = response.localized_object_annotations
            return (respuesta_objeto_api)
```

Luego que tenemos la respuesta de la API, debemos procesarla con Pandas para quedarnos con los objetos ya que la respuesta va a llegar con datos, útiles únicamente para el desarrollador, que confundirían al usuario final. La API nos va a ofrecer una variable llamada *score* que es un float entre 0 y 1 indicando cuanta confianza tiene la IA de que el nombre del objeto es correcto, y en el caso de que sea de menos de 0.65 lo descartamos.

```
def get_object(self, respuesta_objeto_api):

    # guardamos los valores obtenidos de la api
    respuesta_objeto_api = respuesta_objeto_api

    # Creamos un dataframe con la respuesta de la API
    df_reconocer_objeto = pd.DataFrame(
        columns=['name', 'score', 'vertices'])

    # Hacemos un for que analiza cada objeto de todos los que habia en la respuesta de la API
    for obj in respuesta_objeto_api:
        df_reconocer_objeto = df_reconocer_objeto.append(
            dict(
                name=obj.name,  # Nombre del objeto
                score=obj.score,  # Confiabilidad de que el objeto sea correcto
                # Sirve para saber los vertices del objeto y poder ubicarlo
                vertices=obj.bounding_poly.normalized_vertices
            ),
            ignore_index=True)

    # Descartamos los objetos de poca probabilidad de que sean correctos
    df_reconocer_objeto = df_reconocer_objeto.query('score >= 0.65')

    # Llamamos a la funcion que localize el lugar del objeto
    return(self.locate_object(df_reconocer_objeto))
```

Además, también creamos una función que analiza la posición X, Y del objeto para dar una orientación de donde está el mismo.

```
def locate_object(self, df_reconocer_objeto):

    # leo el dataframe con todos los objetos
    df_reconocer_objeto = df_reconocer_objeto

    # Leo el tamaño del dataframe
    largo_dataframe = len(df_reconocer_objeto)

    # Hacemos un for para analizar cada objeto
    for contador in range(largo_dataframe):

        # Se crea una variable para las puntas de los objetos
        vertices_objetos = df_reconocer_objeto.loc[contador, "vertices"]

        # se calcula el valor promedio de la lozalizacion del objeto en x
        promedio_x = (
            vertices_objetos[0].x + vertices_objetos[1].x)/2

        # se calcula el valor promedio de la lozalizacion del objeto en y
        promedio_y = (
            vertices_objetos[0].y + vertices_objetos[2].y)/2

        # Se crean las variables para saber la ubicacion de x e y
        ubicacion_x = ""
        ubicacion_y = ""

        # Se define la ubicacion del objeto en el eje y
        if promedio_y < 0.33:
            ubicacion_y = " la zona superior"
        elif promedio_y < 0.66:
            ubicacion_y = " el centro"
        else:
            ubicacion_y = " la zona inferior"
```

2.5. **Detección de colores:**

Para procesar la imagen y extraer los colores predominantes que tiene, conectamos a la API de Google Vision y le enviamos el contenido de la imagen usando JSON:

```
def get_response_api(self, ):

    # Leo la imagen
    with open(self.image_path, 'rb') as image:
        content = image.read()  # Leemos las cosas de la imagen y lo guardamos en contenido
        # Enviamos la peticion a la API de google con el formato JSON
        # especifico optimizar la velocidad de respuesta
        response = self.client.annotate_image({'image': {'content': content}, 'features': [
            {'type': vision.enums.Feature.Type.IMAGE_PROPERTIES}], }).image_properties_annotation
        respuesta_rgb_api = response.dominant_colors.colors  # Respuesta de colores

        return(respuesta_rgb_api)
```

La API nos va a pasar una lista de colores en RGB ya que no puede clasificarlos por nombres.

```
def get_color(self, respuesta_rgb_api):

    # guardamos los valores de la api
    respuesta_rgb_api = respuesta_rgb_api

    # Creo dataframe para guardar los valores de la api
    df_pixel_fraction = pd.DataFrame(
        columns=['r', 'g', 'b', 'score', 'pixel_fraction'])

    # Ciclo for para analizar cada color que nos dio la respuesta
    for color in respuesta_rgb_api:
        df_pixel_fraction = df_pixel_fraction.append(
            dict(
                r=int(color.color.red),
                g=int(color.color.green),
                b=int(color.color.blue),
                score=color.score,
                pixel_fraction=color.pixel_fraction
            ),
            ignore_index=True)

    return(self.get_predominant_colors(df_pixel_fraction))
```

Puede ser que la API detecte varios colores que ocupen poquísimo espacio en la imagen y no tiene sentido decirlos, por lo tanto, los descartamos. La información

de cuanto espacio ocupa en la imagen se haya en el float *pixel_fractions* que varía entre 0 y 1.

```
def get_predominant_colors(self, df_pixel_fraction):

    # leo el dataframe con todos los valores
    df_pixel_fraction = df_pixel_fraction

    df_pixel_fraction = df_pixel_fraction.sort_values(
        ['pixel_fraction'], ascending=False).head(2)  # Agarramos los dos valores con mas predominancia
    df_pixel_fraction = df_pixel_fraction.sort_values(
        ['score'], ascending=False).head(2)  # Agarramos valores de confiabilidad altos

    # Reiniciamos el indice del dataframe
    df_pixel_fraction = df_pixel_fraction.reset_index(drop=True)

    # Nos fijamos si el primer color y el segundo son ambos confiables
    dif = df_pixel_fraction['score'][0] - df_pixel_fraction['score'][1]

    # Si los dos colores son confaibles usamos los dos
    if (dif < 0.10):
        for cont in [0, 1]:
            self.r = (df_pixel_fraction['r'][cont])
            self.g = (df_pixel_fraction['g'][cont])
            self.b = (df_pixel_fraction['b'][cont])

            # Llamamos a la funcion que hace el query para el nombre de color
            yield(self.get_color_name(self.r, self.g, self.b))

    # Si solo hay uno confiable usamos ese y ya esta
    else:
        self.r = (df_pixel_fraction['r'][0])
        self.g = (df_pixel_fraction['g'][0])
        self.b = (df_pixel_fraction['b'][0])
```

Una vez que tenemos valores RGB los procesamos con Pandas para que compare con un dataset de valores RGB y sus respectivos nombres. La función realiza un query a nuestra tabla de colores y busca cual es el color con valores de RGB más similares a la respuesta de la API.

```
def get_color_name(self, R, G, B):

    # Guardamos la tabla de colores con nombres
    dataframe_colores = self.dataframe_colores

    # Guardamos los valores de R,G,B
    self.r = R
    self.g = G
    self.b = B

    # Se busca el color en la tabla con menos distancia del color que nos da la API
    minimo = 1000
    for i in range(len(dataframe_colores)):
        # Se calcula la distancia minima con valores absolutos
        distancia_color = abs(self.r - int(dataframe_colores.loc[i, "R"])) + abs(
            self.g - int(dataframe_colores.loc[i, "G"])) + abs(self.b - int(dataframe_colores
        # Si la distancia del color es menor o igual se guarda como minimo
        if(distancia_color <= minimo):
            minimo = distancia_color
            # Se guarda el nombre del color con menor distancia al buscado
            nombre_color = dataframe_colores.loc[i, "Principal_color"]

    # Se devuelve el nombre del color
    return (nombre_color)
```

2.6. **Detección de ondas cerebrales:**

Para la programación del Mindwave se eligió usar Arduino IDE ya que es más cómodo utilizar el entorno de Arduino IDE ya que usa el conocido C ahorrando tiempo de aprender nuevos lenguajes.

El usuario se va a colocar el headset que va a estar configurado, mediante comandos AT, para conectar al Ojo de Van Gogh por Bluetooth donde un Arduino NANO procesara la información.

Primero debemos medir los valores de actividad cerebral que hay en el momento ya que los picos eléctricos cuando se está relajado son diferentes a cuando se está estresado. Una vez que tenemos estos valores definimos que el umbral de activación es 2 veces el valor de actividad cerebral en reposo.

```
void Mindwave::Calibrar_Sensor()
{
  Serial.println("Calibrando");
  Serial.println("Calibrando");
  Serial.println("Calibrando");
  Serial.println("Calibrando");
  while (Calibracion_raw <= 100 ){
    if (ReadOneByte() == 170) // AA 1 st Sync data
    {
      if (ReadOneByte() == 170) // AA 2 st Sync data
      {
        Plength = ReadOneByte();
        if (Plength == 4) // Small Packet
        {
          generatedchecksum = 0;
          for (int i = 0; i < Plength; i++)
          {
            payloadDataS[i] = ReadOneByte(); //Read payload into memory
            generatedchecksum += payloadDataS[i];
          }
          generatedchecksum = 255 - generatedchecksum;
          checksum = ReadOneByte();
          if (checksum == generatedchecksum) // Varify Checksum
          {
            if (j < 512)
            {
              Raw_data = ((payloadDataS[2] << 8) | payloadDataS[3]);
              if (Raw_data & 0xF000)
              {
                Raw_data = (((~Raw_data) & 0xFFF) + 1);
              }
              else
```

En la anterior imagen se ve como calculamos el checksum para verificar que el emisor y el receptor estén sincronizados.

Luego medimos la actividad cerebral:

```
      {
        Avg_Raw = Temp / 512;

        Calibracion_raw = Avg_Raw;
        Serial.println("La calibracion_rAW es: " + Calibracion_raw);

        Umbral_de_parpadeo = Calibracion_raw * 2;
      }
    }
  }
```

Una vez esta calibrado, se corre en loop infinito la función de activación que está constantemente midiendo el payload de la actividad cerebral y si supera el umbral envía la orden a la Raspberry Pi Zero W de que se parpadeo dos veces.

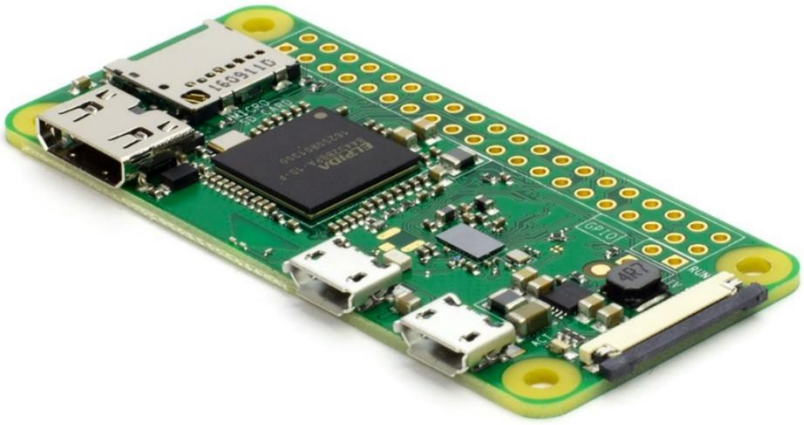
```
void Mindwave::Onesec_Rawval_Fun()
{

   Avg_Raw = Temp / 512;

   Serial.println(Avg_Raw);
   if (On_Flag == 0 && Off_Flag == 1)
   {
      if (n < 3)
      {
         Temp_Avg += Avg_Raw;
         n++;
      }
      else
      {
         Temp_Avg = Temp_Avg / 3;
         if (Temp_Avg < EEG_AVG)
         {
            On_Flag = 1;
            Off_Flag = 0;
         }
         n = 0;
         Temp_Avg = 0;
      }
   }
   Eye_Blink();
   j = 0;
   Temp = 0;
}
```

3. Sistema Embebido

3.1. Raspberry Pi Zero W



Elegimos utilizar la Raspberry Pi Zero W ya que tiene una gran variedad de características que hicieron más fácil la construcción del proyecto, su conectividad y su potencia nos permite usar el lenguaje Python y utilizar la inteligencia artificial de Google visión. También la utilizamos por sus características físicas, ya que al ser de un tamaño reducido en comparación con las demás nos permite simplificar el tamaño del equipo.

Características técnicas:

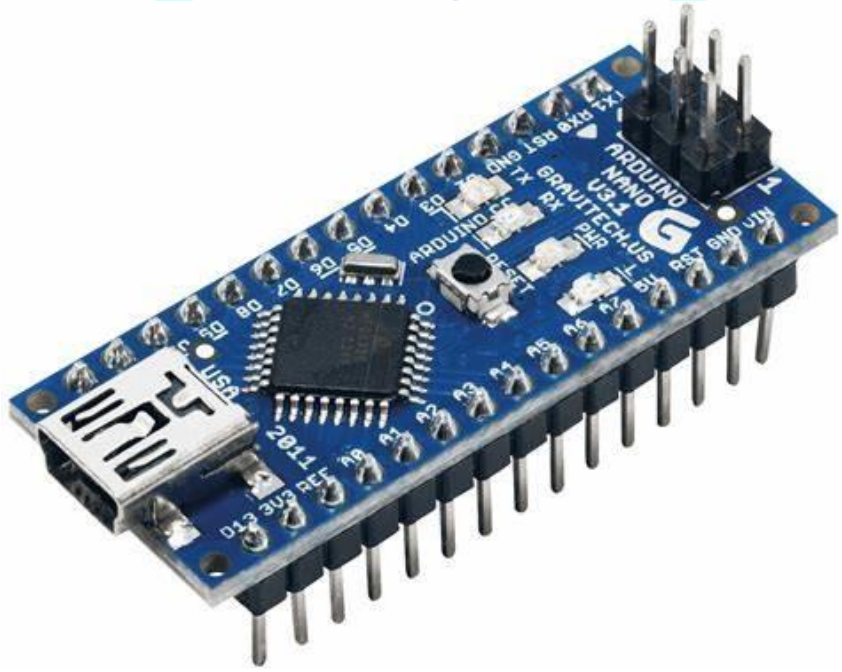
En este caso tenemos un dispositivo con las mismas características que la Raspberry Pi Zero pero que además incorpora el módulo de conectividad Cypress CYW43438, que le agrega conectividad Wi-Fi y Bluetooth, que consiste en:

- LAN inalámbrica 802.11 b / g / n
- Bluetooth 4.1
- Bluetooth de baja energía (BLE)

Al igual que el Pi Zero, también tiene:

- Procesador Broadcom BCM2835 @ 1Ghz ARM 11
- 512 MB de RAM
- Puertos Mini HDMI y USB On-The-Go
- Alimentación micro USB
- Cabezal de 40 pines compatible con HAT
- Vídeo compuesto y reiniciar encabezados
- Conector de cámara CSI

3.2. **Arduino Nano**



Arduino Nano, es una placa pequeña adaptada para trabajar con placas prototipos, que está basada en el microcontrolador ATmega328 (Arduino Nano 3.x).

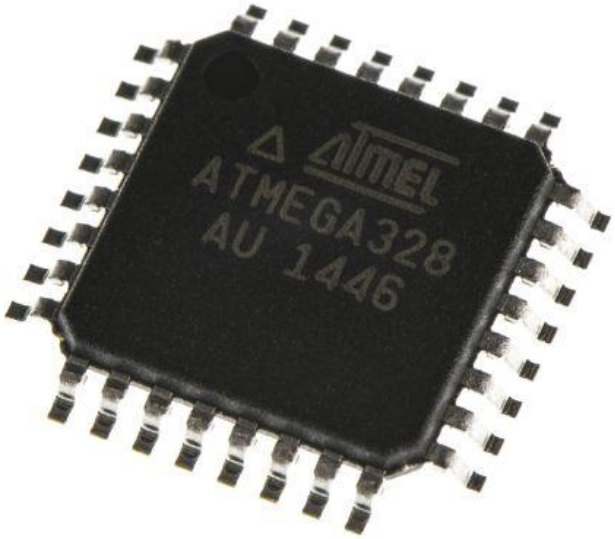
La placa Arduino está diseñada de tal manera que es muy fácil para los principiantes comenzar con los microcontroladores. Esta placa especialmente es compatible con la placa de pruebas, es muy fácil de manejar las conexiones. Tiene la misma funcionalidad que el Arduino Duemilanove, pero es más pequeño en tamaño. Sólo difiere en que Funciona a través de un USB mini-B en lugar de uno estándar.

Su pequeño tamaño y facilidad de uso hizo que eligiéramos esta placa.

Sus características técnicas son:

- Microcontrolador Arduino ATmega328
- Arquitectura, AVR
- Voltaje de operación, 5 V
- Memoria flash,32 KB
- SRAM 2 KB
- Velocidad del reloj 16 MHz
- Pines de E/S analógicas, 8
- EEPROM,1 KB
- Corriente continua por pin entrada salida, 40 mA (Pines de E/S)
- Voltaje de entrada, 7-12 V
- Pines de E/S digitales, 22
- Salida PWM, 6
- Consumo de energía, 19 mA
- Tamaño de la placa de circuito impreso,18 x 45 mm
- Peso, 7 g

3.3. **Atmega 328**



El ATmega328 es un microcontrolador de un solo chip creado por Atmel en la familia megaAVR (más tarde Microchip Technology adquirió Atmel en 2016). Tiene un núcleo de procesador RISC de 8 bits de arquitectura Harvard modificado.

Hoy el ATmega328 se usa comúnmente en múltiples proyectos y sistemas autónomos donde se requiere un microcontrolador simple, de bajo consumo y bajo costo. Tal vez la implementación más común de este chip es en la popular plataforma Arduino, en sus modelos Uno y Nano.

Características técnicas:

El microcontrolador AVR RISC de 8 bits de Atmel combina:

- Memoria flash ISP de 32 KB con capacidades de lectura durante la escritura
- 1 KB EEPROM
- 2 KB SRAM
- 23 líneas de E/S de uso general
- 32 registros de trabajo de propósito general
- 3 temporizadores/contadores flexibles con modos de comparación, interrupciones internas y externas
- USART programable en serie,
- Una interfaz serie de 2 hilos orientada a bytes
- Puerto serie SPI
- Convertidor A/D de 6 canales de 10 bits (8 canales en paquetes TQFP y QFN/MLF)
- Temporizador de vigilancia programable con oscilador interno y cinco modos de ahorro de energía seleccionables por software.
- El microcontrolador funciona entre 1,8-5,5 voltios.
- El microcontrolador logra un rendimiento que se aproxima a 1 MIPS por MHz.

3.4. Sistemas Brain Computer Interface (BCI)

Es una tecnología que se basa en la adquisición de ondas cerebrales de una persona para luego ser procesadas e interpretadas por una máquina u ordenador. Establecen un nuevo camino para interactuar con tecnología mediante nuestro pensamiento, ya que estas interfaces permiten transformarlo en acciones reales en nuestro entorno.

Podemos decir que el principal objetivo de estos sistemas es dar un uso a esta tecnología de manera asistida para aquellas personas con discapacidades o en situación de dependencia. También pueden servir de ayuda como técnica para la mejora de la comunicación y control de personas con afasia o apraxia.

Tipos de BCI

Podemos dividir los sistemas BCI en tres grupos diferentes:

- Activo: se caracteriza por poder controlar de manera consciente y voluntaria un sistema BCI, independientemente de los eventos externos.
- Reactivo: el control de la aplicación está directamente relacionado con un estímulo externo que, indirectamente, modela la actividad cerebral.

- Pasivo: se deriva del procesamiento de señales cerebrales arbitrarias y sin ninguna intencionalidad por el usuario. La información recogida se utiliza para saber nuestro estado cognitivo, como nuestros niveles de atención, relajación, etc.

Los sistemas endógenos se basan en el reconocimiento de patrones cerebrales sin la necesidad de un estímulo externo, sino se producen por la voluntad del usuario. Como ejemplo de este proceso es el uso de ritmos beta para el control de un dispositivo cuando el usuario imagina o intenta realizar movimientos.

Los sistemas exógenos, a diferencia de los anteriores, basan su control a estímulos externos y se obtiene la respuesta cerebral de los mismos. Un ejemplo de estos dispositivos sería el determinar la dirección de la mirada o la postura del usuario para obtener una respuesta cerebral.

Aplicaciones de sistemas BCIs

Al realizar un sistema BCI, deben quedar claramente definidos los objetivos sobre los cuales se lo va a efectuar, y cuáles deben ser las limitaciones sobre el mismo. Las áreas principales en las que actualmente son aplicables este tipo de sistemas son:

- Mejorar calidad de vida personas con grave discapacidad motora
- Aumento de la autonomía e independencia
- Control de dispositivos del entorno, manejo de prótesis, sillas de ruedas, herramientas de comunicación.
- Rehabilitación cognitiva

3.5. **NeuroSky Mindwave Mobile 2 EEG**

La elección del aparato capaz de leer ondas cerebrales, con el cual se realizará el sistema BCI, es crucial, ya que en función de las características de este se podrá diseñar un sistema más o menos potente en función de la información recibida.

Por esto, nos decantamos hacia el NeuroSky MindWave, que es una diadema que recoge la actividad eléctrica del cerebro y divide la señal según la frecuencia en diversos tipos de ondas, lo que nos permite inferir en nuestro estado mental. El registro de dicha actividad se hace con la configuración de referencia común, dado que solo tenemos un único canal y el electrodo esta referenciado con el potencial de nuestra oreja.

Es capaz de leer, principalmente, el estado de meditación (medido por las ondas alfa / theta) o atención/concentración (medido por las ondas beta / gamma) que nuestro cerebro emite y detectar cuando la persona realiza un parpadeo

Aspectos importantes:

- Es capaz de leer los cuatro ritmos cerebrales más importantes (beta/alfa/tetha/delta).
- Incluye los algoritmos eSense para estados de atención/meditación.
- Dispone de conexión inalámbrica.
- Es muy ligero y cómodo.
- Tiene la capacidad de hacer lecturas de EMG (electromiografía)
- Tiene la capacidad de detectar el parpadeo.

Por todos estos motivos hemos elegido el NeuroSky MindWave para la realización de nuestro sistema BCI.

La estructura es de un material plástico duro. Se adapta cómodamente a la cabeza y se puede regular su tamaño. El sensor primario se apoya en la frente y requiere de un cierto ajuste para adaptarlo a la cabeza de cada usuario.

Se conecta por radiofrecuencia a través de un stick USB, y funciona con la mayoría de los modernos sistemas operativos (Windows XP o posterior, Mac OS X 10.6.5 o posterior). La vida de la batería es de unas 8/10 horas con una sola pila AAA. En la siguiente tabla podemos observar algunas especificaciones:

Peso	Sensor frontopolar	Sensor de lóbulo	Potencia	Frecuencia RF	Potencia máx. RF	Velocidad de transmicion de datos RF
90 g	225 x 155 x 92 mm	225 x 155 x 165 mm	30mW máx 50mW	2.420 – 2.471 Ghz	6 dBm	250 kbit/s
Rango RF	**Perdida de paquetes por conexión inalámbrica**	**UART Baudrate**	**Rango máx. señal de entrada**	**Rango del filtro**	**Resolucion ADC**	**Velocidad de muestreo**
10 m	5 %	57600 Baud	1 mV pkpk EEG	3 – 100 Hz	12 bits	512 Hz

En cuanto al resto de las aplicaciones, debido a la naturaleza del sensor del MindWave, sólo dependen de tres tipos de variables utilizadas de una manera más o menos útil según el uso que el usuario quiera darles. Estas variables son:

- Concentración / no-concentración
- Meditación / no-meditación
- Parpadeo

El parpadeo será una utilidad muy importante para nuestro proyecto ya que con esta variable se le da las indicaciones al sistema.

Detección del parpadeo

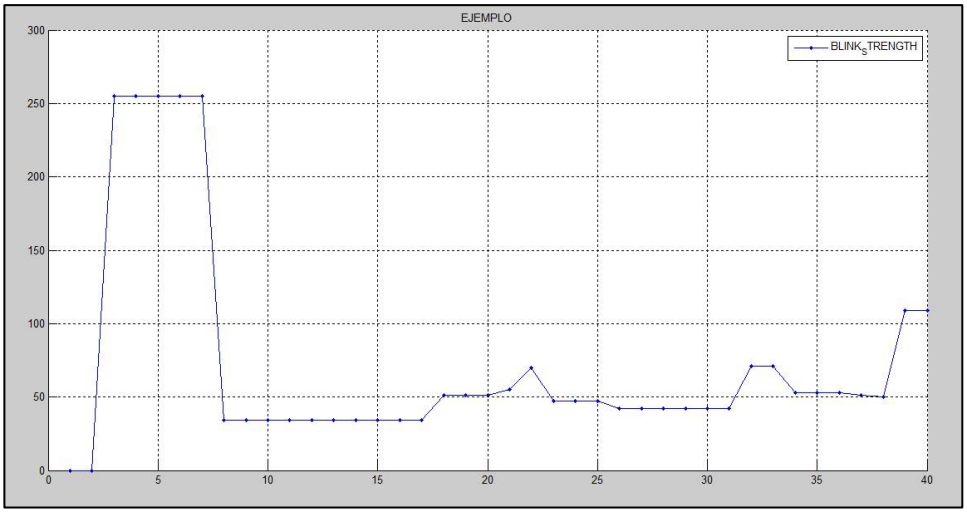
Los datos captados con el casco Mindwave pueden fluctuar mucho y el resultado no ser el esperado. Para la validación del resultado se ha usado el parpadeo.

El simple hecho de parpadear no es indicativo de un estado mental o de una onda cerebral concreta, pero como se ha comentado con anteriormente, es un potencial por acción muscular, es decir, una señal detectada por el electrodo producida por la contracción del músculo frontal.

Dentro del protocolo de transmisión de datos el chip ThinkGear incorpora un algoritmo que da un valor directo de este pico de voltaje producido por el parpadeo, DATA_BLINK_STRENGTH.

```
TG_DATA_BLINK_STRENGTH = 37; data_blink(i,1) =
calllib ('ThinkGear','TG_GetValue', connectionId1, TG_DATA_BLINK_STRENGTH);
```

Este dato se actualiza cada segundo. Usando el primer código del apartado anterior, se realiza una prueba durante 40 segundos en la que el usuario está parpadeando con diferente “fuerza”, obteniendo la siguiente figura.

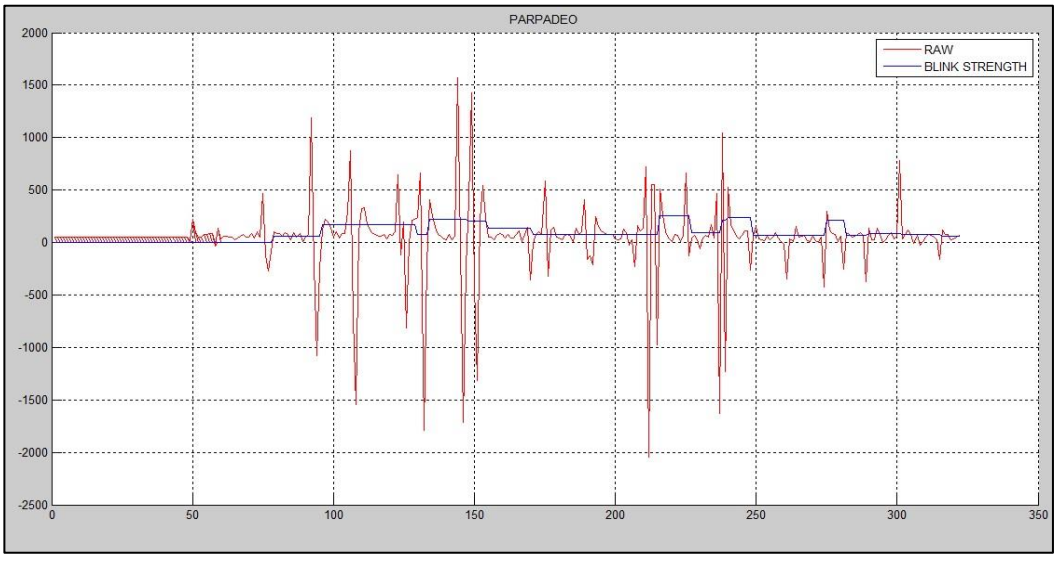


Señal parpadeo (TG_DATA_BLINK_STRENGTH)

La señal blink tiene un rango entre 0 y 255, llegándose a la conclusión que los valores más cercanos a 255, se producirán cuando el parpadeo era muy forzado. El problema de usar esta señal es que, al actualizarse cada segundo, no recoge todos los parpadeos, ya que tienes que parpadear en el momento exacto en el que se está adquiriendo el dato, y eso es problemático a la hora de tratar de usar esta señal como validación, ya que existe la posibilidad de pretender validar, y que no se reconozca ningún parpadeo.

Este artefacto interfiere la señal raw que lo está recibiendo de manera muy notable, produciendo un pico de voltaje característico. Se pueden apreciar los picos en la siguiente figura donde se muestra la señal bruta captada con Matlab y alterada voluntariamente a partir del parpadeo. Se observa que a mayor contracción muscular estos picos serán más grandes.

En la siguiente figura, se ha realizado la misma prueba anterior, pero adquiriendo también la señal raw.



Señal parpadeo y señal raw

Si el usuario de manera voluntaria realiza la contracción muscular parpadeando forzadamente, se produce un pico de voltaje en la señal raw que destaca por encima del resto, y en la señal blink se observa un valor distinto de cero. Debido a que la señal blink se actualiza cada segundo, es menos real que la señal raw, y como esta contracción se necesita para validar una respuesta realizando dos parpadeos forzados en un tiempo de tres segundos, se usará la señal raw que actualiza los datos cada 2 ms, por lo que recogerá cualquier parpadeo.

En función de la fuerza que se ejerza en el parpadeo, este pico será más o menos pronunciado. Se han realizado varias pruebas con distintos usuarios, para determinar un rango estimado de lo que se considera parpadeo voluntario e involuntario. A modo de resumen:

	UMBRAL PARPADEO (valores absolutos)
Parpadeo involuntario	0 a 599
Parpadeo forzado	600 a 1499
Parpadeo muy forzado	Valores mayores que 1500

Rango de valores del parpadeo

3.6. Protocolo de Comunicación del Mindwave

El protocolo de comunicación del caso Mindwave, esta instaurado en el chip ThinkGear. El chip ThinkGear se encarga de atenuar el ruido ambiental para poder realizar mediciones en áreas no aisladas eléctricamente. El sensor frontopolar recoge las señales, las envía al chip, y este las procesa y las convierte en una secuencia de datos digitales. Una vez filtradas las interferencias las ondas cerebrales se amplifican y son cotejadas con los algoritmos que se encuentran codificados en la memoria del chip.

Datos ThinkGear

- **POOR_SIGNAL Quality**

Se trata de un byte sin signo que describe como de “pobre” es la señal medida por el ThinkGear. El rango de valores es de 0 a 200; cualquier valor distinto de cero indica que hay ruido y si el valor es 200, quiere decir que no existe conexión entre el sensor y la piel.

Este valor es enviado cada segundo, está activo por defecto y su frecuencia de muestreo es 1 Hz. La calidad de la señal puede estar modificada por varias causas:

- Los contactos del sensor o la masa no están en contacto con la frente o la oreja respectivamente.
- Existe un contacto pobre entre el sensor y la piel.
- Excesivo movimiento del usuario
- Ruido electroestático ambiental
- Ruido generado por otras medidas bioeléctricas: EMG, EoG.

En cualquier caso, *ThinkGear* cuenta con un filtrado y un algoritmo para la medición de los valores de *Attention* y *Meditation*, que permiten detectar, corregir, compensar y tolerar muchos de estos tipos de error. eSense Meters

Cuenta con dos valores: *Attention* y *Meditation*, los cuales se recogen en una escala de 0 a 100. Cerca de cero tanto la atención como la meditación son bajas, y cerca de 80 a 100 son altas. El valor cero significa que el sensor no es capaz de calcular un valor (normalmente se da al principio, durante la calibración). Los valores de *attention* y *meditation* se actualizan cada segundo.

La razón de que el rango sea tan grande es debido a que el algoritmo que realiza estos cálculos cuenta con un aprendizaje dinámico para ajustarse a cada usuario.

- **RAW Wave Value**

Está formado por dos bytes y representa una única muestra de la señal RAW. Se trata de un entero con signo, dentro del rango -32768 a 32767. El primer byte representa la parte alta y el segundo byte la parte baja.

Por defecto este valor también se encuentra siempre activo y se recoge 512 veces por segundo, una muestra cada 2ms.

- **ASIC_EEG_POWER**

Representa el valor de 8 tipos de ondas cerebrales. Esta información va recogida en 8 grupos de 3 bytes, enteros con signo y en formato little-endian. Los 8 tipos de ondas cerebrales se recogen en el siguiente orden:

- Delta (0.5 – 2.75 Hz)
- Theta (3.5 – 6.75 Hz)
- Low-Alpha (7.5 – 9.25 Hz)
- High-Alpha (10 – 11.75 Hz)
- Low-beta (13 – 16.75 Hz)
- High-beta (18 – 29.75 Hz)
- Low-gamma (31 – 39.75 Hz)
- High-gamma (41 – 49.75 Hz)

Estos valores no tienen unidades y solo tienen sentido comparándolos con los demás o entre ellos mismos. Se recogen una vez por segundo y están activados por defecto.

- **BLINK STRENGTH**

Este valor indica la intensidad del parpadeo de ojos del usuario. Es un byte cuyo rango va de 1 a 255. Se recoge cada vez que existe un parpadeo. El dato se actualiza cada segundo.

Paquetes ThinkGear

Los componentes ThinkGear envían los datos mediante un flujo de bytes en serie asíncrono. En el siguiente apartado se va a desarrollar una explicación de los diferentes campos que representan los ThinkGear Packets, con el fin de extraer e interpretar la información transmitida.

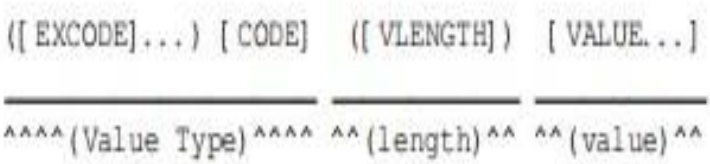
Estructura del paquete

Los paquetes se mandan en un flujo serie asíncrono de datos. El medio de transporte puede ser la UART, puerto serie COM, USB, Bluetooth, ficheros o cualquier otro mecanismo capaz de transportar bytes.

La estructura del paquete es la siguiente:

El tamaño mínimo del paquete es de 4 bytes (Header: 3 bytes, Payload: 0 bytes, Checksum: 1 byte), y el tamaño máximo 173 bytes (Header: 3 bytes, Payload: 169 bytes, Checksum: 1 byte).

- [SYNC]: 2 byte. Indica el comienzo de un nuevo paquete.
- [PLENGTH]: 1 byte. Indica la longitud de carga útil del paquete. Valores entre 0 y 169 (bytes de payload) como máximo. Cualquier valor mayor de 169 significa ERROR.
- [PAYLOAD]: carga útil, información. Este campo está formado por la siguiente estructura denominada **DATAROW**.



- [EXCODE]: se utiliza para indicar el nivel de extensión de código utilizando el byte 0x55. Nivel 1 un byte de 0x55, nivel 2, dos bytes de 0x55.
- [CODE]: identifica el tipo de dato codificado. Se usa para enviar múltiples datos o para enviar datos que no tienen cabida en una sola trama.
- [VLENGTH]: indica la longitud del dato.
- [VALUE]: valor del dato definido por [CODE].
- [CHECKSUM]: 1byte. Se emplea una tabla de códigos para verificar la integridad de la carga útil de los paquetes. El Checksum se calcula de la siguiente forma:
    - Se suman todos los bytes del paquete [PAYLOAD].
    - Se toman los 8 bits de menor peso de la suma.
    - Se realiza la inversa de los 8 bits.

Si el Checksum calculado coincide con el Checksum del paquete, el receptor puede comenzar a analizar la carga útil de datos; si no coincide, el paquete se descarta.

A continuación, se plasma un breve resumen de los códigos del ThinkGear:

TABLA DE CÓDIGOS			
EXCODE	**CODE**	**VLENGTH**	**VALUE**
-	0x02	-	POOR_SIGNAL Quality (0 a 255)
-	0x04	-	ATTENTION eSense (0 a 100)
-	0x05	-	MEDITATION eSense (0 a 100)
-	0x16	-	Blink Strength (0 a 255)
-	0x80	2	RAW Wave Value (-32768 a 32767)
-	0x83	24	ASIC_EEG_POWER
-	0x55	-	[EXCODE]
-	0xAA	-	[SYNC]

Tabla de códigos del chip ThinkGear

Paquete ejemplo:

```
Byte:    value  //  [CODE]   Explanation

[  0]: 0xAA   //   [SYNC]
[  1]: 0xAA   //   [SYNC]
[  2]: 0x20   //   [PLENGTH]   32 bytes
[  3]: 0x02   //   [POOR_SIGNAL]  Quality
[  4]: 0x00   //   Buena Señal (0/200)
[  5]: 0x83   //   [ASIC_EEG_POWER_INT]
[  6]: 0x18   //   [VLENGTH]   24 bytes
[  7]: 0x00   //   (1/3) Inicio bytes Delta
[  8]: 0x00   //   (2/3)
[  9]: 0x94   //   (3/3) Fin bytes Delta
[ 10]: 0x00   //   (1/3) Inicio bytes Theta
[ 11]: 0x20   //   (2/3)
[ 12]: 0x42   //   (3/3) Fin bytes Theta
[ 13]: 0x00   //   (1/3) Inicio bytes Low-alpha
[ 14]: 0x00   //   (2/3)
[ 15]: 0x0B   //   (3/3) Fin bytes Low-alpha

[ 16]: 0x00   //   (1/3) Inicio bytes High-alpha
[ 17]: 0x00   //   (2/3)
[ 18]: 0x64   //   (3/3) Fin bytes High-alpha
[ 19]: 0x00   //   (1/3) Inicio bytes Low-beta
[ 20]: 0x00   //   (2/3)
[ 21]: 0x4D   //   (3/3) Fin bytes Low-beta
[ 22]: 0x00   //   (1/3) Inicio bytes High-beta
[ 23]: 0x00   //   (2/3)
[ 24]: 0x3D   //   (3/3) Fin bytes High-beta
[ 25]: 0x00   //   (1/3) Inicio bytes Low-gamma
[ 26]: 0x00   //   (2/3)
[ 27]: 0x07   //   (3/3) Fin bytes Low-gamma
[ 28]: 0x00   //   (1/3) Inicio bytes Mid-gamma
[ 29]: 0x00   //   (2/3)
[ 30]: 0x05   //   (3/3) Fin bytes Mid.gamma
[ 31]: 0x04   //   [ATTENTION] eSense
[ 32]: 0x0D   //   Nivel de atencion de valor 13
[ 33]: 0x05   //   [MEDITATION] eSense
[ 34]: 0x3D   //   Nivel de meditacion de valor 61
[ 35]: 0x34   //   [CHKSUM]
```

Explicación paso a paso del *paquete ejemplo*:

1. Se lee la serie de bytes recibidos hasta que se encuentra uno de valor 0xAA [SYNC].

2. Se lee el segundo byte y se comprueba que es también 0xAA [SYNC]
- Si no lo es, se vuelve al paso 1.
- En caso contrario se continúa con el paso 3.

3. Se lee el siguiente byte correspondiente a PLENGTH:
- Si tiene valor 170 (SYNC), se vuelve al paso 3.
- Si tiene valor mayor de 170, se vuelve al paso 1, porque PLENGTH es demasiado largo.
- Si tiene valor menor de 170, se continúa con el paso 4.

4. Se leen los bytes (indicados en PLENGTH) de la carga útil, PAYLOAD, y se almacenan en una tabla o vector. A su vez se suma cada byte que se lee a un acumulador (checksum+=byte), que se comprobará con el CHECKSUM al final.

5. Se agarran los últimos 8 bits del acumulador checksum y se invierten:

```
checksum &=0xFF;
checksum= ~checksum & 0xFF
```

6. Por último, se lee el siguiente byte de la cadena, el CHECKSUM:

- Si el valor de CHECKSUM no coincide con el del acumulador, se ha producido un error y se descarta el paquete.
- En otro caso, se considera como válido dicho paquete y se pasa a analizar la secuencia de la carga útil.
- En cualquier paso, se vuelve al punto 1.

3.7. Protocolo UART en Arduino

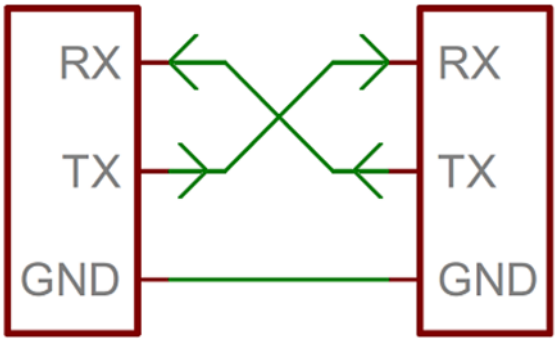
UART (Universal Asynchronous Receiver-Transmitter) significa recepción y transmisión asíncronas universales y es un protocolo de comunicación simple que permite que Arduino se comunique con dispositivos serie. El sistema UART usa los pines digitales 0 (RX) y 1 (TX) y con otro PC a través del puerto USB.

Este periférico, que se encuentra en todas las placas Arduino, permite que Arduino se comunique directamente con un PC gracias al hecho de que Arduino tiene un convertidor de USB a serie incorporado (ATmega16U2).

Con este componente, los programas escritos en Windows, Mac o Linux se pueden usar con un Arduino conectado a través de USB como si fuera un puerto serie.

Las funciones principales de chip UART son: manejar las interrupciones de los dispositivos conectados al puerto serie y convertir los datos en formato paralelo, transmitidos al bus de sistema, a datos en formato serie, para que puedan ser transmitidos a través de los puertos y viceversa.

Transmisión y recepción de datos serie



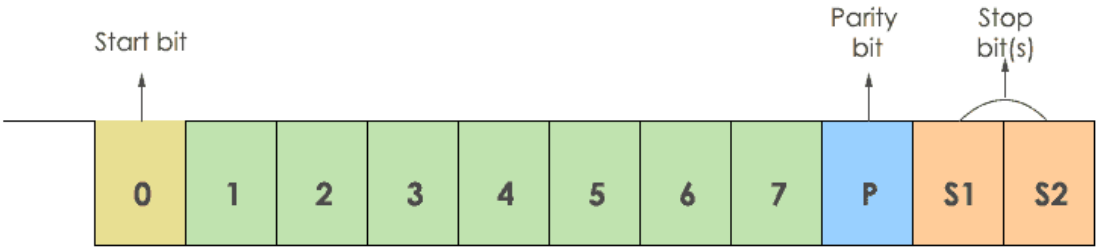
Tx: Transmisión de datos.

Rx: Recepción de datos

- Velocidad de Datos: LA comunicación máxima puede ser entre 230Kbbs a 460kbb
- Tipo comunicación: Asíncrona
- Protocolo: 8bits con un bit de start y un bit de stop.
- Ventajas:
  - Muy simple, permite conectar de forma rápida dos dispositivos, suele usarse con RS232 o RS485, por ejemplo.
- Desventajas:
  - Esta limitado a la comunicación entre dos dispositivos.
  - La velocidad se debe acordar al principio de la conexión en caso de no ser la misma dará lecturas erróneas.

Formato de protocolo

El UART inicia la comunicación con un bit de inicio '0'. El bit de inicio inicia la transferencia de datos serie y el bit de detención finaliza la transacción de dato



También se proporciona con un bit de paridad (par o impar). El bit de paridad par se representa por '0' (número par de 1's) y el bit de paridad impar está representado por '1' (número impar de 1's).

3.8. Protocolo Bluetooth Low Energy (BLE)

El Bluetooth de baja energía (Bluetooth Low Energy o BLE), es un subconjunto del estándar Bluetooth v4.0. Dispone de una pila de protocolos en referencia a la capa OSI completamente nueva y orientada a conexiones sencillas en aplicaciones de muy baja potencia (dispositivos dependientes de batería o pila).

BLE opera en la banda ISM de 2.4 GHz, sin embargo, a diferencia del Bluetooth clásico, BLE permanece en modo de suspensión constantemente, excepto cuando se inicia una conexión. Los tiempos de conexión reales son solo de unos pocos milisegundos, a diferencia de Bluetooth que tomaría más de 100 milisegundos.

Protocolos para Bluetooth Low Energy

La capa física contiene la circuitería de comunicaciones capaz de realizar los procesos de modulación y demodulación de señales analógicas y posteriormente transformarlas en símbolos digitales. La tecnología BLE es capaz de utilizar hasta 40 canales de 2MHz en la banda ISM de 2.4 GHz. El estándar emplea la técnica “frequency hopping” o “saltos en frecuencia”, siguiendo una secuencia de saltos

pseudo-aleatrorios entre los canales frecuenciales mencionados que ofrece un alto grado de robustez frente a interferencias.

La capa de enlace, se encarga de gestionar características como los requerimientos temporales del estándar, chequeo de mensajes y reenvío de mensajes erróneos recibidos, gestión, etc. Además, ofrece la definición de roles (Advertiser, Scanner, Master and Slave) que permiten identificar de forma lógica el rol de cada dispositivo en el proceso de comunicación. El nivel LL es del mismo modo responsable de procesos de control como el cambio de parámetros de la conexión o la encriptación.

HCI es un protocolo estándar que permite que la comunicación entre un host y un controlador se lleve a cabo a través de un interfaz serie. A modo de ejemplo, en la mayoría de ordenadores el host y la aplicación corren en la CPU principal mientras que el controlador está situado en hardware específico y separado, conectado mediante UART o USB. El estándar Bluetooth define HCI como el conjunto de comandos y eventos para la interacción de ambas partes (host y controlador).

La capa L2CAP (Logic Link Control and Adaptation Protocol), se responsabiliza de dos tareas fundamentales en un proceso de comunicación. En primer lugar, el proceso de multiplexación, es decir, la capacidad de dar formato a mensajes provenientes de las capas OSI superiores y encapsularlos en paquetes estándar BLE, así como el proceso inverso.

Por otro lado, la fragmentación y recombinación. Paquetes que en nivel de aplicación suponen datagramas de gran cantidad de bytes son fragmentados correctamente adecuándose al MTU de BLE (27 bytes de payload máximo).

Seguridad en Bluetooth Low Energy

Las principales características de seguridad que incorpora Bluetooth Low Energy son el cifrado de 128 bits y la autenticación. Además, como protocolo de transmisión es más robusto gracias al salto de frecuencia adaptable (AFH).

La comunicación BLE en dispositivos que ya han verificado una conexión es realmente segura. Sin embargo, para conectarse, los dispositivos deben emparejarse primero, y aquí es donde reside la principal vulnerabilidad de los sistemas BLE.

Durante la primera etapa de emparejamiento, los dispositivos intercambian información básica sobre sus capacidades.

La segunda fase de emparejamiento está dedicada a generar e intercambiar claves. Es en este punto que las conexiones BLE pueden ser manipuladas: si la conexión no está asegurada adecuadamente, los atacantes pueden tomar el control de los dispositivos y los datos que transmiten.

Esto solo puede evitarse con un método de emparejamiento apropiado. Existen 4 métodos de emparejamiento que se pueden utilizar:

- Just Works: Es el método predeterminado, más utilizado y menos seguro de todos. La clave temporal que los dispositivos intercambian durante la

segunda fase de emparejamiento se establece en 0, y los dispositivos generan el valor de la clave a corto plazo en función de eso. Es altamente vulnerable a ataques de MiTM, ya que cualquier dispositivo puede conectarse con la clave temporal 0.

- Out Of Band: En el emparejamiento fuera de banda algunos datos se transmiten a través de un protocolo inalámbrico diferente. El emparejamiento OOB se puede implementar durante la segunda fase del emparejamiento, de modo que las claves intercambiadas entre dispositivos no se transfieren a través del protocolo BLE menos seguro.
- Claves de paso: En este método la unidad central establece un pin de 6 dígitos que el usuario debe ingresar en los dispositivos periféricos. Debido a que cada dispositivo es verificado manualmente por el usuario.
- Comparación numérica: Disponible solo para BLE Secure Connections. Después de verificar automáticamente sus valores de confirmación en la segunda fase del emparejamiento, con la comparación numérica, los dispositivos usan el valor aleatorio que intercambiaron previamente para generar otro valor de seis dígitos, que el usuario debe comparar manualmente en ambos dispositivos.

4. Electrónica

4.1. **KiCad**

KiCad es un paquete de software libre para la automatización del diseño electrónico. Facilita el diseño de esquemáticos para circuitos electrónicos y su conversión a placa de circuito impreso (del inglés: Printed Circuit Board, PCB).

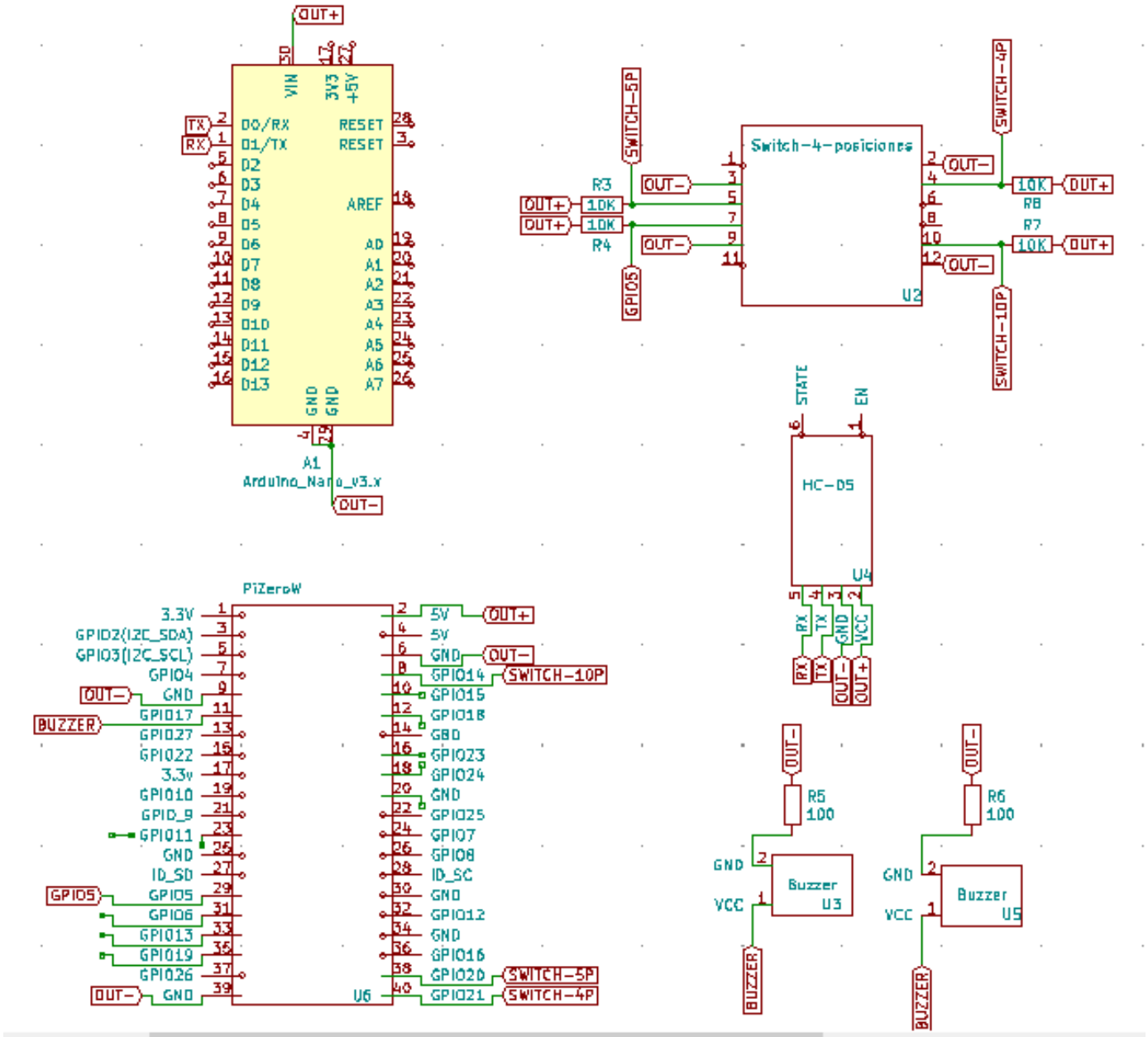
KiCad fue desarrollado originalmente por Jean-Pierre Charras. Cuenta con un entorno integrado para captura esquemática y de diseño de PCB. Existen herramientas dentro del software para crear una lista de materiales, ilustraciones, archivos Gerber y vistas 3D de la PCB y sus componentes.

Está organizado en cinco partes:

- KiCad - El administrador de proyectos.
- Eeschema - El editor de esquemáticos.
- Pcbnew - Entorno de diseño de los circuitos impresos (PCB).
- Gerbview - Visualizador de archivos Gerber.
- Bitmap2Component - Herramienta que convierte imágenes a footprints para realizar "PCB artwork".

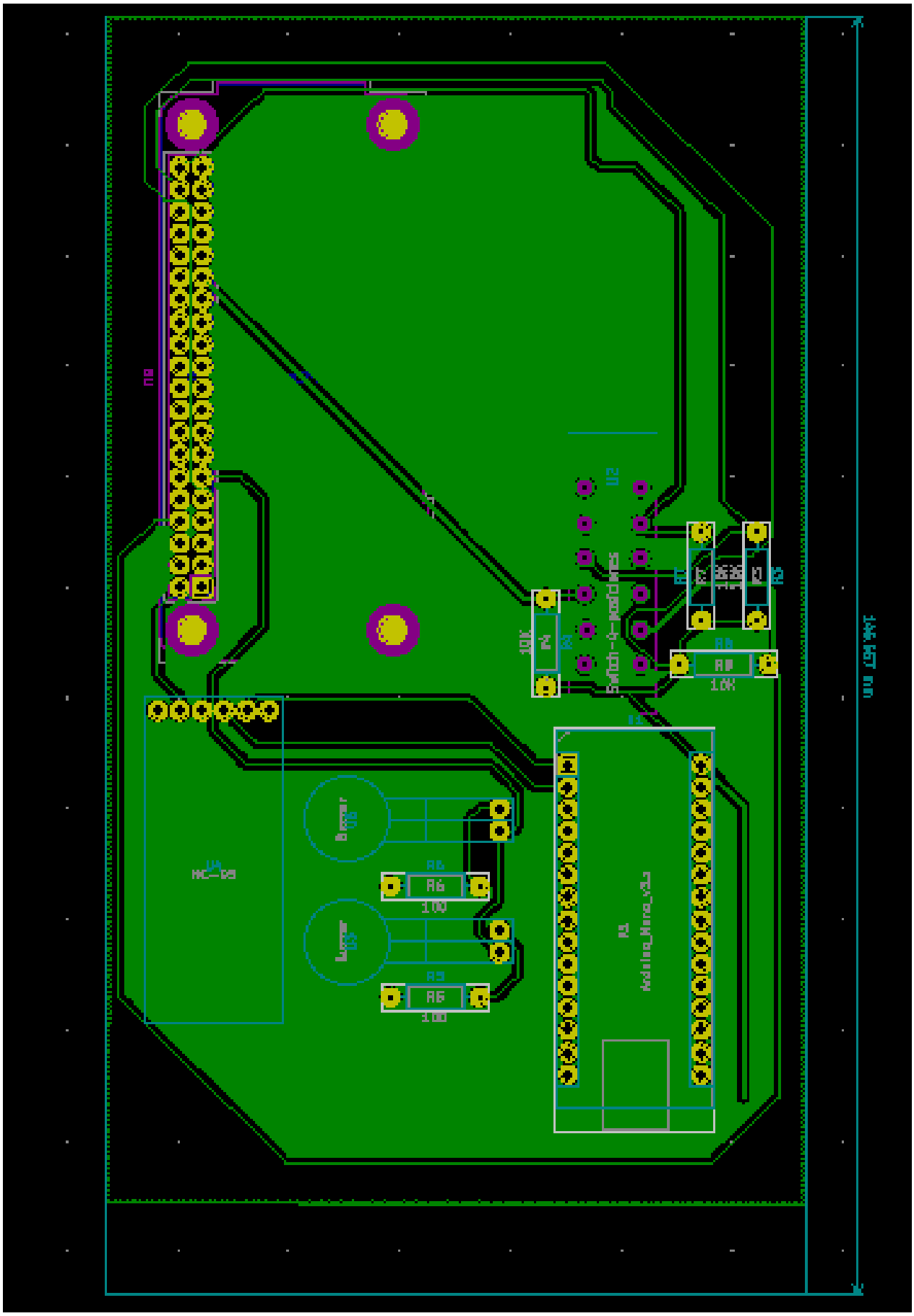
El link de descarga de KiCad es: https://kicad.org/download/

4.2. **Esquemático**

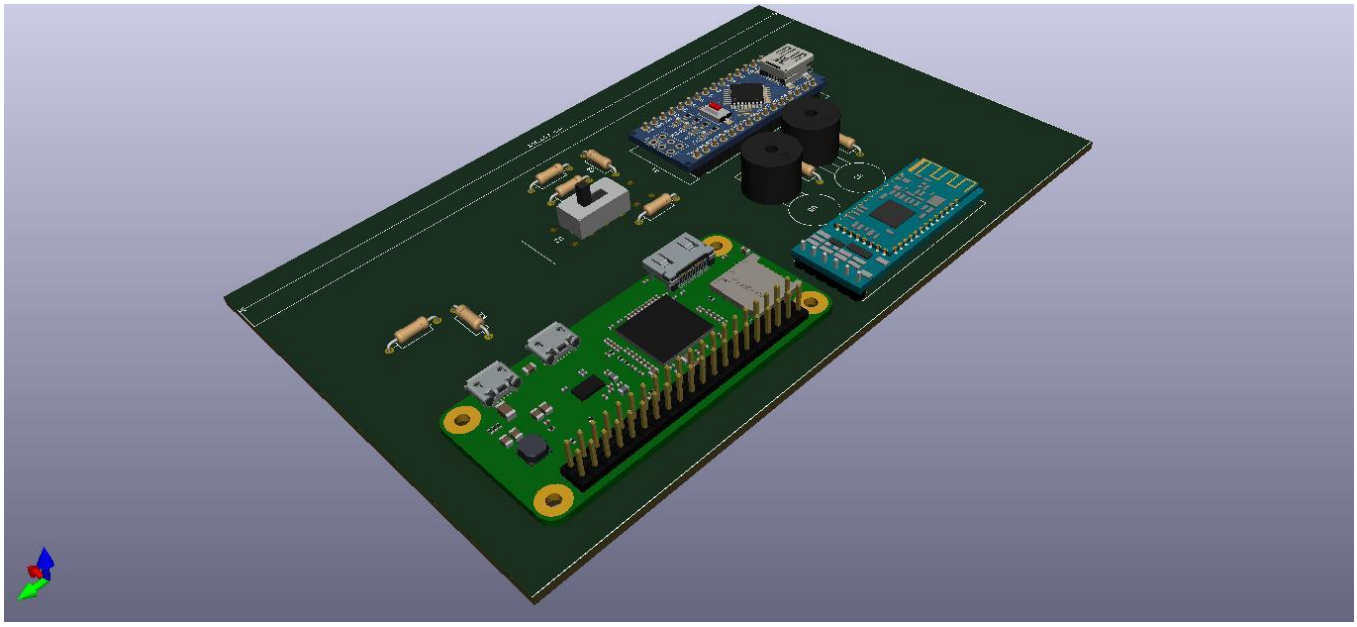


4.3. **Printed Circuit Board (PCB):**

El sistema electrónico que la persona va a tener en su antebrazo va a ser montado en un PCB para que los componentes queden conectados robustamente y se pueda usar en un ambiente cotidiano.


HC-05
Buzzer
Buzzer
Arduino_Nano_v3.x
1K
10K
100
100
1K

4.4. **Modelado 3D**



4.5. **Alimentación del Sistema**

Nuestro sistema estará alimentado por un cargador portátil Power Bank TL-PB 20000 que es de Polímero de Litio. Sus características principales son:

- Tiene 2 puertos de carga de alta velocidad con una potencia de salida adaptativa de hasta 2.1 A.
- Luz indicadora
- Alta capacidad de batería 20000 mAh
- Led indicador de carga
- 1 Puerto Type C
- 1 Puerto Micro USB
- Compatibilidad Universal
- Protección contra cortocircuito, sobretensión, sobre corriente, sobrecarga, descarga excesiva y calentamiento excesivo

4.6. Especificaciones Técnicas:

Nuestro dispositivo electrónico cuenta con las siguientes características:

- Pi camera: 5MPx 5V con cable FFC para conectar a Raspberry Pi Zero W
- Raspberry Pi Zero W: 5V,512MB RAM, WiFi 802.11b, Bluetooth 4.1, Micro-USB para alimentación, Micro-USB UART para comunicaciones
- 2 Buzzer 5V
- Cargador portátil 20000 mAH
- Carcaza hecha en ABC con Impresora 3D
- Bluetooth 4.0 module HC-05
- NeuroSky Mindwave Mobile 2 EEG

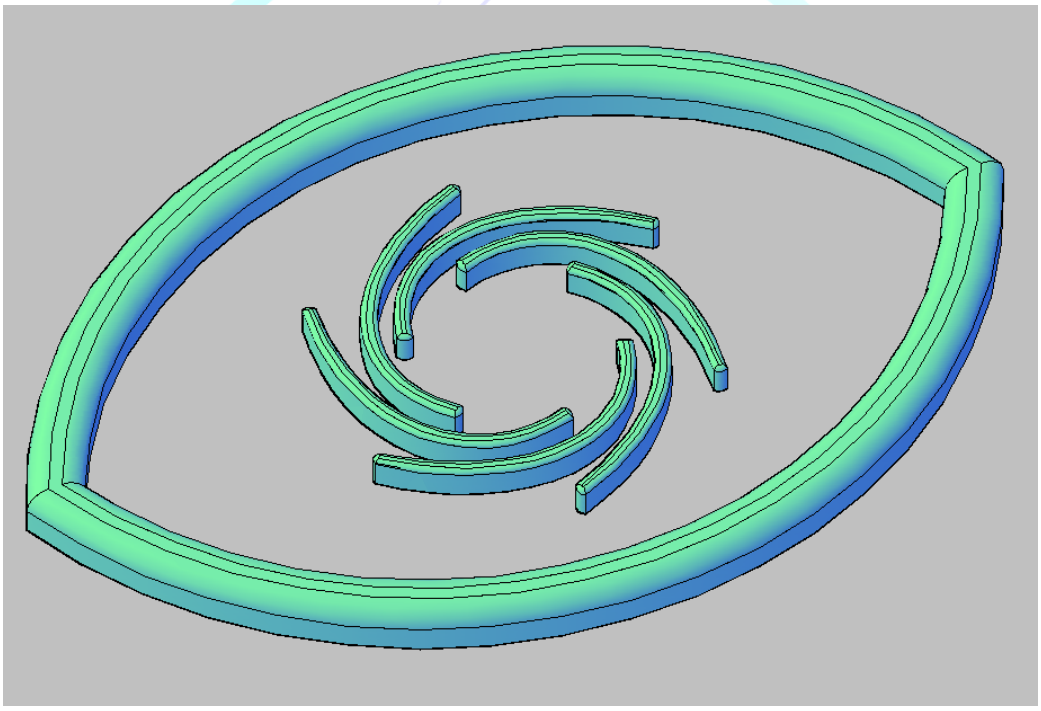
En cuanto a Software tiene las siguientes características:

- Raspberry Pi OS Lite
- Python 3.7
- Arduino IDE 1.8.10

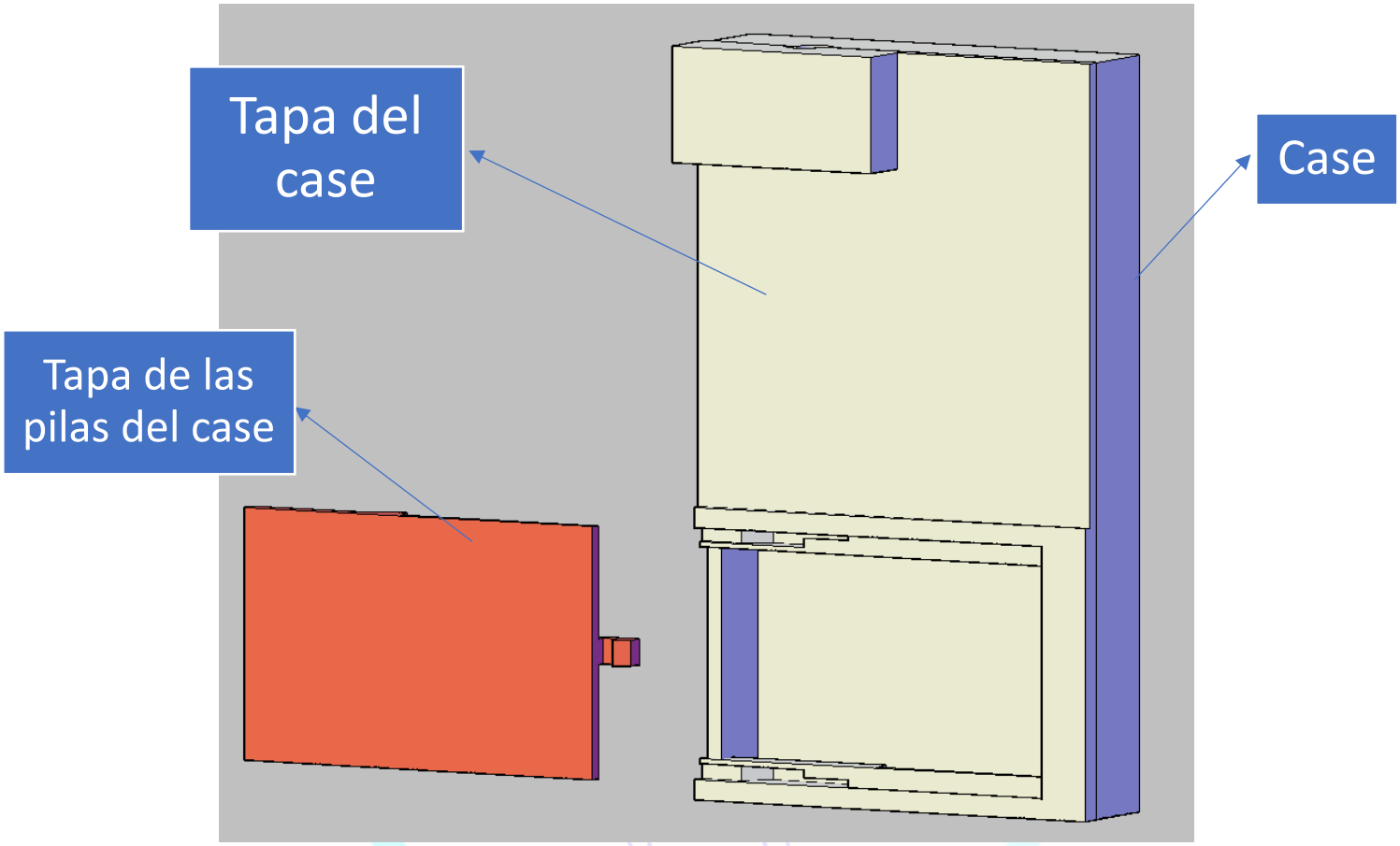
5. Estructura

5.1. Diagrama General de la Estructura

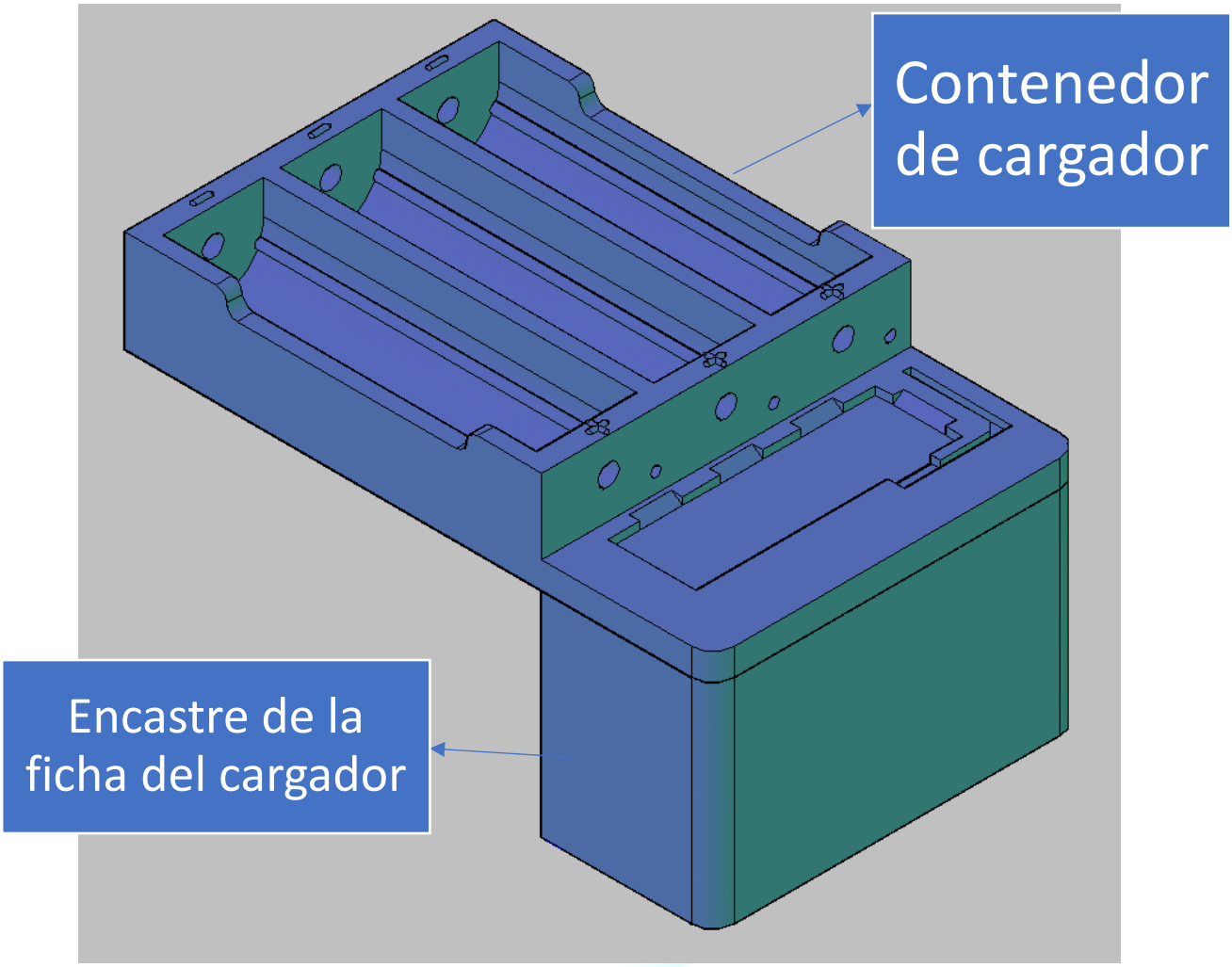
Logo



Case:



Cargador de pilas:



5.2. **Software de Diseño Utilizado:**

AutoCAD 2007

Link de descarga de AutoCAD:

https://latinoamerica.autodesk.com/products/autocad/overview?plc=ACDIST&term=1-YEAR&support=ADVANCED&quantity=1

AutoCAD es un software de diseño asistido por computadora utilizado para dibujo 2D y modelado 3D. Actualmente es desarrollado y comercializado por la empresa Autodesk. AutoCAD es un software reconocido a nivel internacional por sus amplias capacidades de edición, que hacen posible el dibujo digital de planos de edificios o la recreación de imágenes en 3D; es uno de los programas más usados por arquitectos, ingenieros, diseñadores industriales y otros.

Además de acceder a comandos desde la solicitud de comando y las interfaces de menús, AutoCAD proporciona interfaces de programación de aplicaciones (API) que se pueden utilizar para determinar los dibujos y las bases de datos.

Las interfaces de programación que admite AutoCAD son ActiveX Automation, VBA (Visual Basic® for Applications), AutoLISP, Visual LISP, ObjectARX y .NET. El tipo de interfaz que se utilice dependerá de las necesidades de la aplicación y de la experiencia en programación de cada usuario.

Usos y funcionalidades de AutoCAD

Utilizamos este programa por la cantidad de opciones y facilidades que brinda para el diseño y creación de estructuras 3D, tales como:

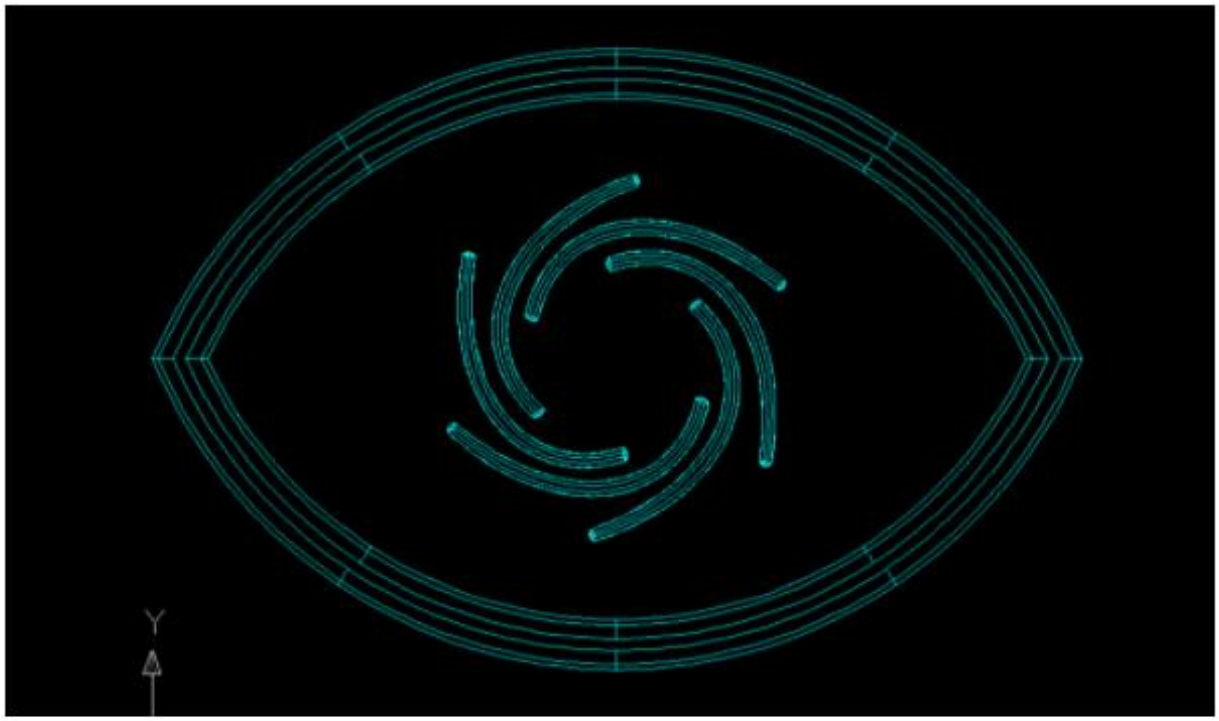
- Funcionalidades de texto, columnas y contornos.
- Acceso a comandos y menús contextuales, lo que permite tener un ritmo de trabajo más ágil y fluido.
- Rápida recuperación de vistas al permitir guardarlas por nombre.
- Extracción de vistas de un sólido tridimensional rápida.
- Proyección real de vistas creadas anteriormente.
- Para ingenieros, la orden propfis permite devolver las propiedades físicas de sólidos 3D y regiones.
- Transparencia de objetos particulares o de capas contenedoras.
- Filtrar objetos según características y propiedades.
- Posibilidad de recuperar proyectos y archivos en múltiples dispositivos gracias a **AutoCAD 360**.

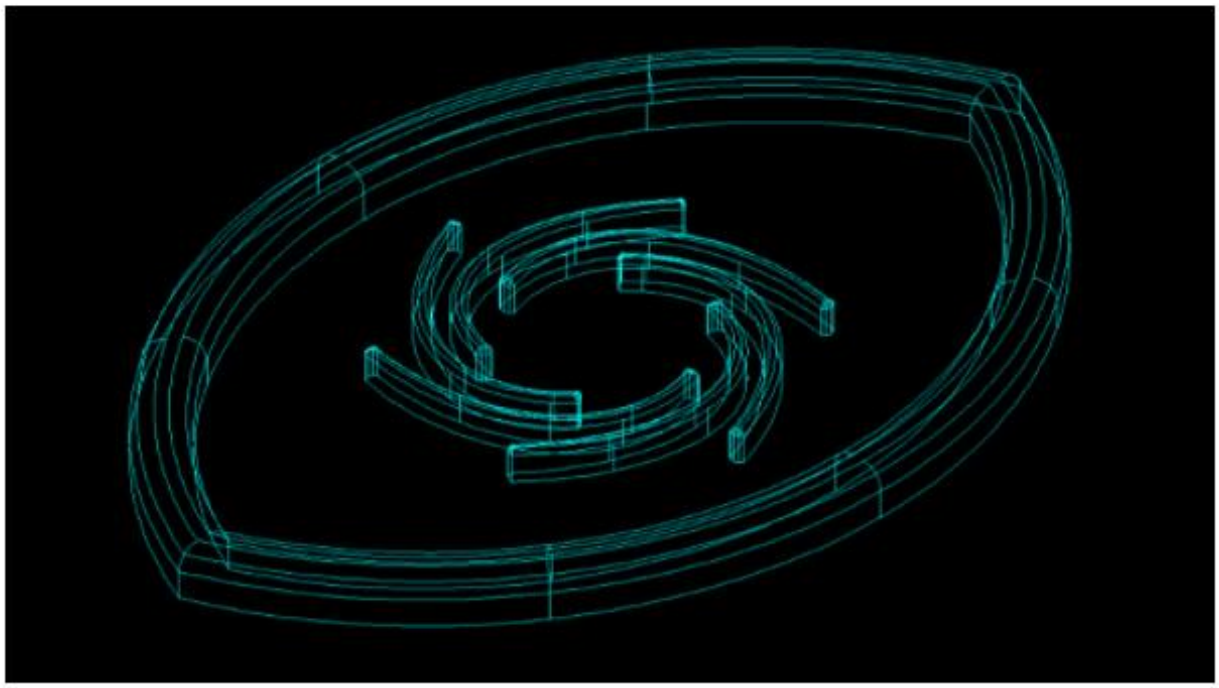
Además, este programa es famoso por sus herramientas potentes de navegación y visualización 3D. Es decir, permiten orbitar, recorrer, pivotar y volar sobre un modelo 3D con la finalidad de ver todas las perspectivas del diseño.

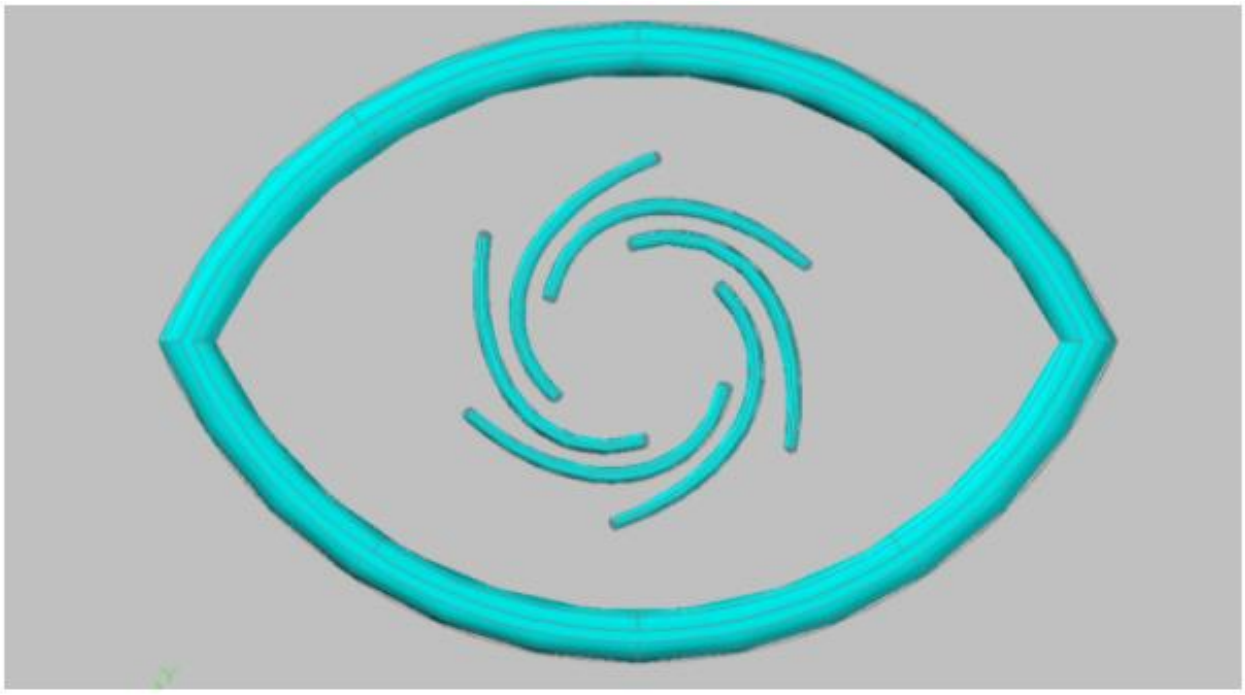
5.3. Partes de la Estructura

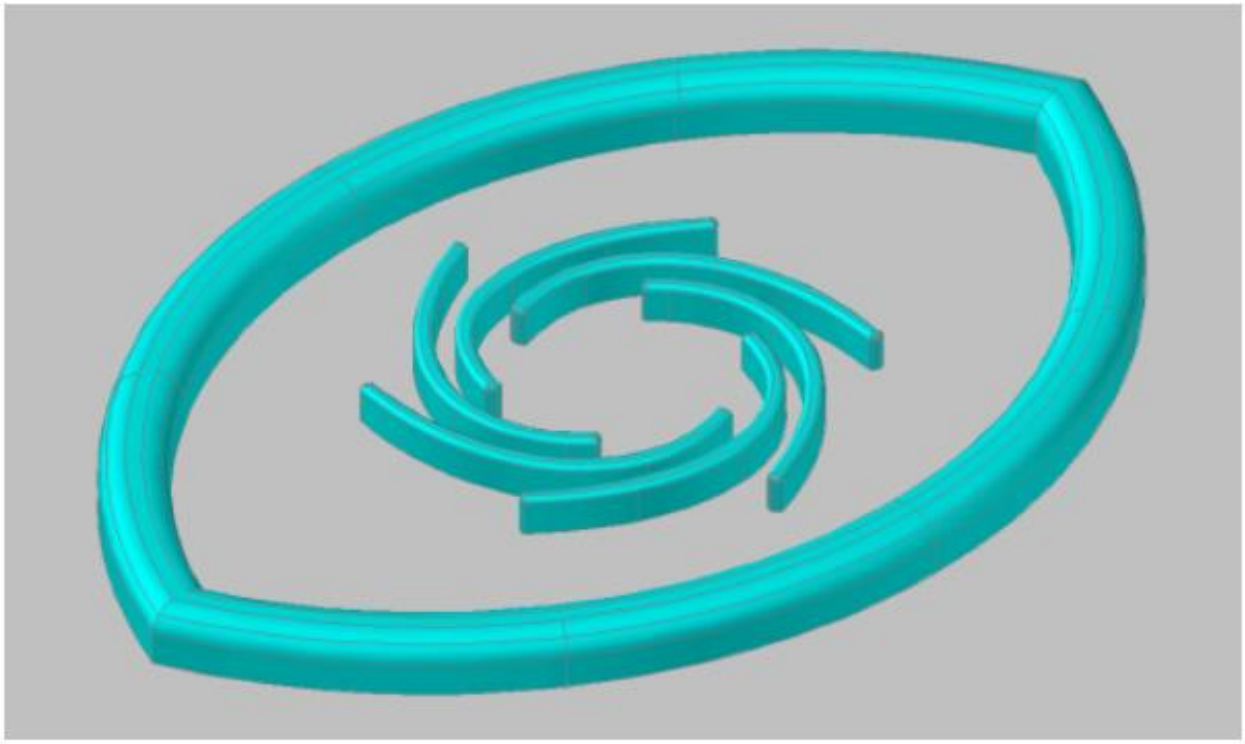
Logo:

Hicimos un modelo 3D del logo para incluirlo en el modelo del cargador de pilas.


Y





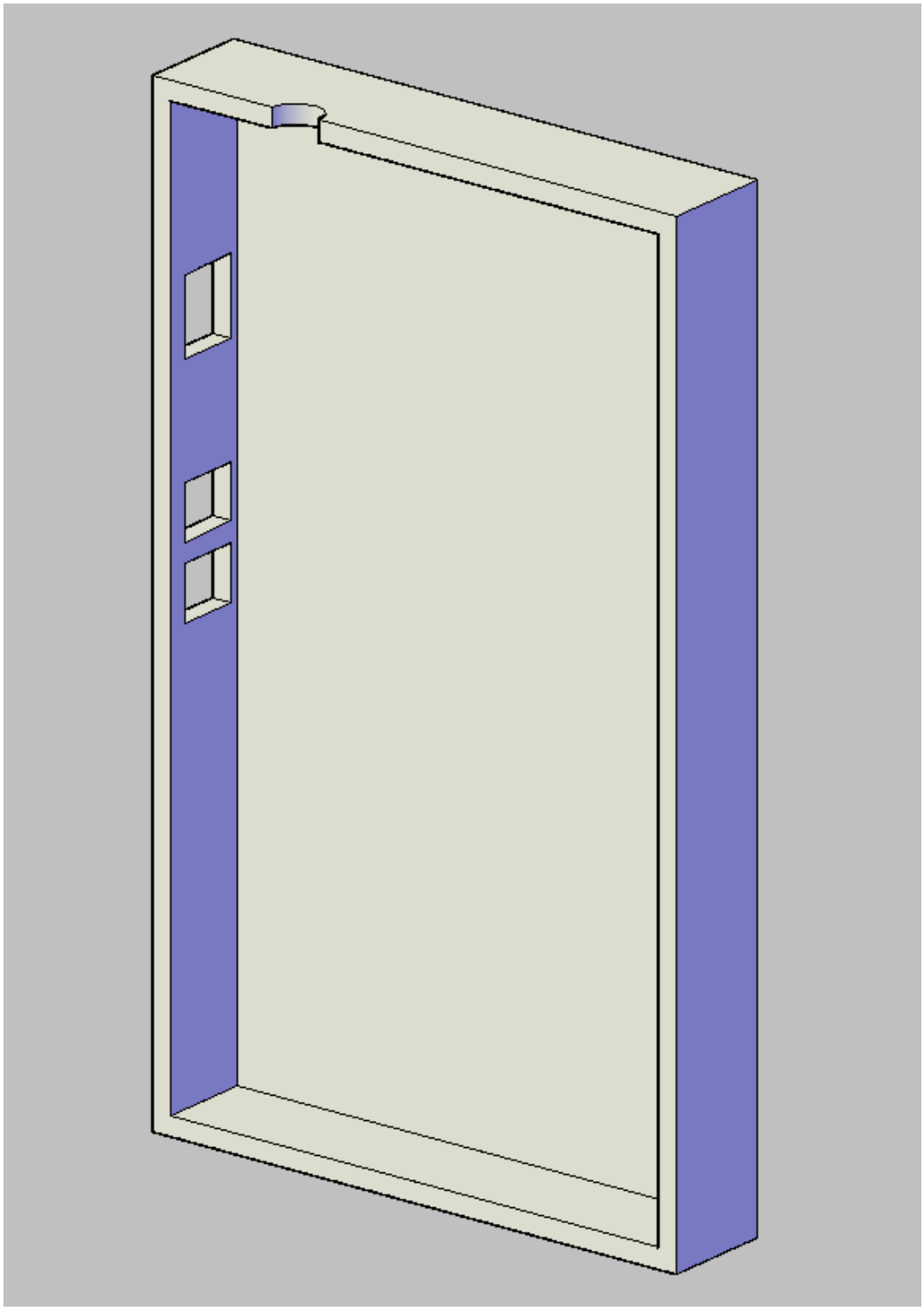


Case:

Creamos una caja que lleva adentro una placa con una Raspberry que se conecta con un Arduino, además de tener los vibradores. Está hecha en 3 partes, la inferior, la superior y la tapa que sirve para cambiarle las pilas al sistema.

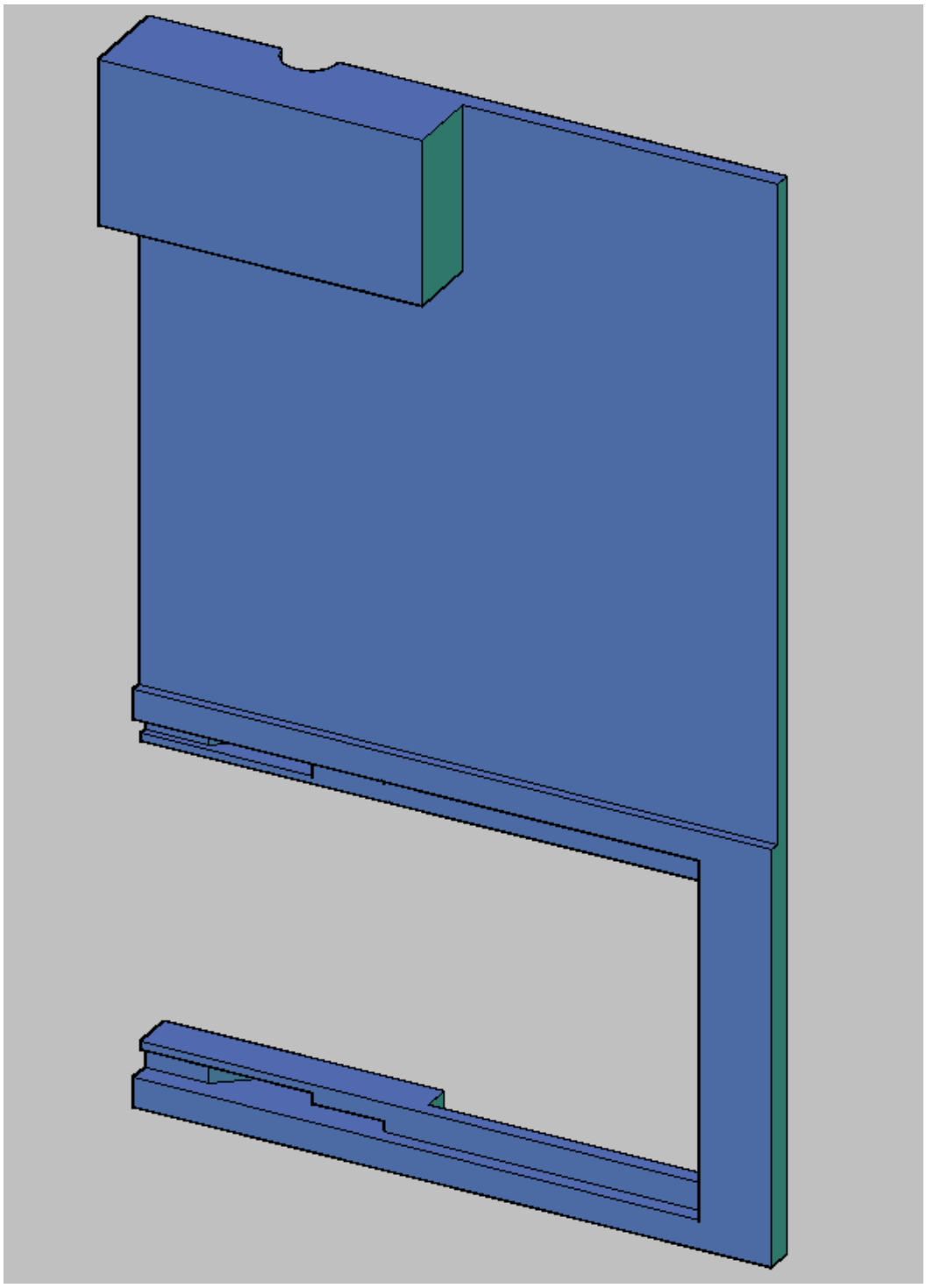
Parte inferior del case:

- Vista Isométrica

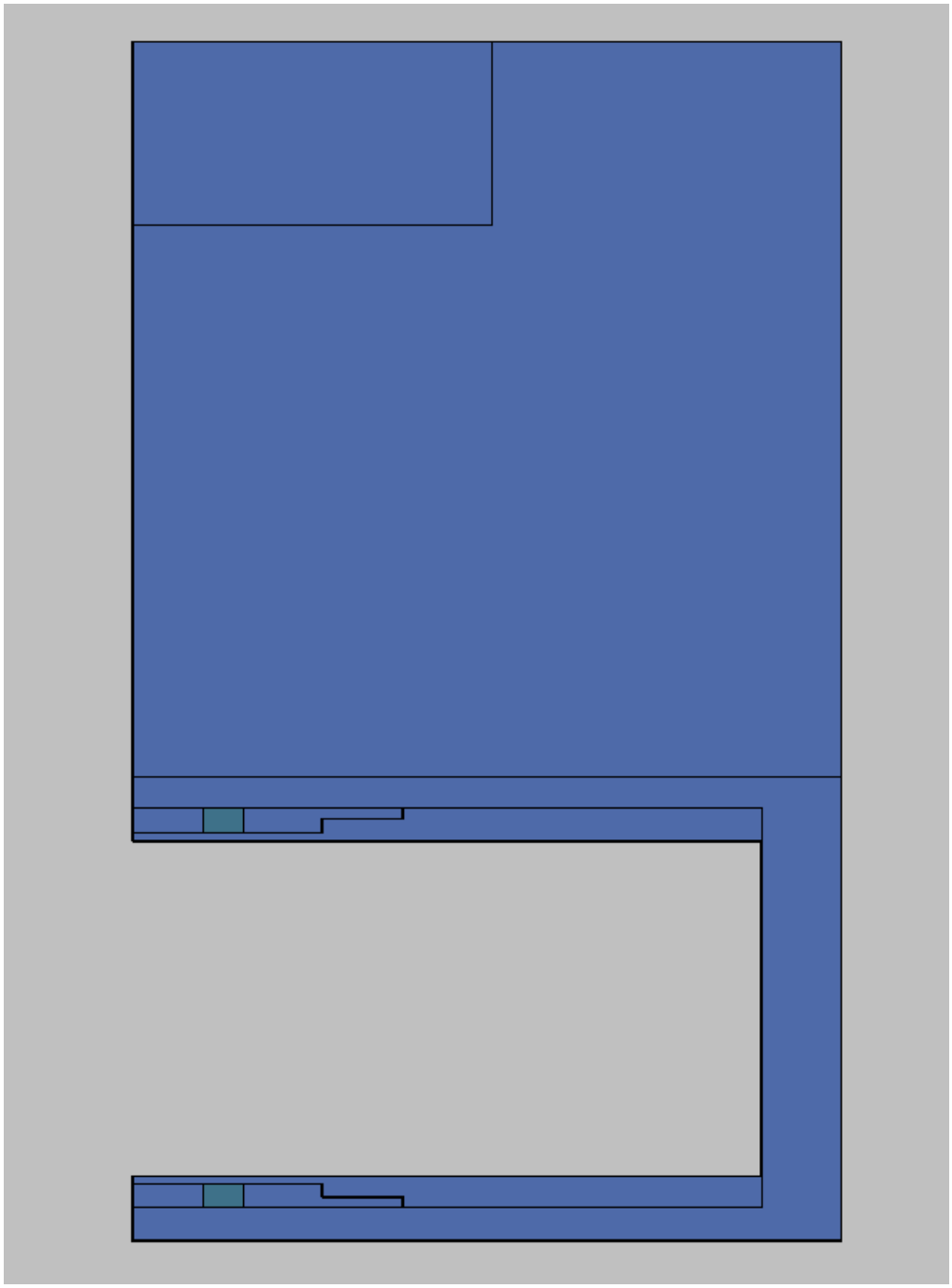


Tapa del Case:

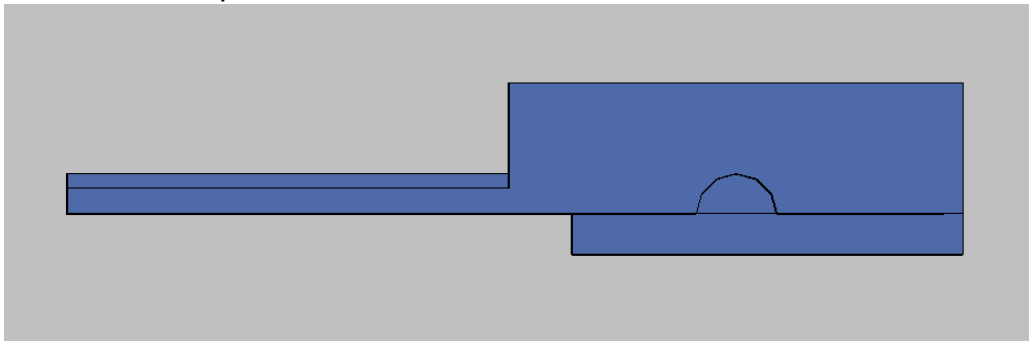
- Vista Isométrica



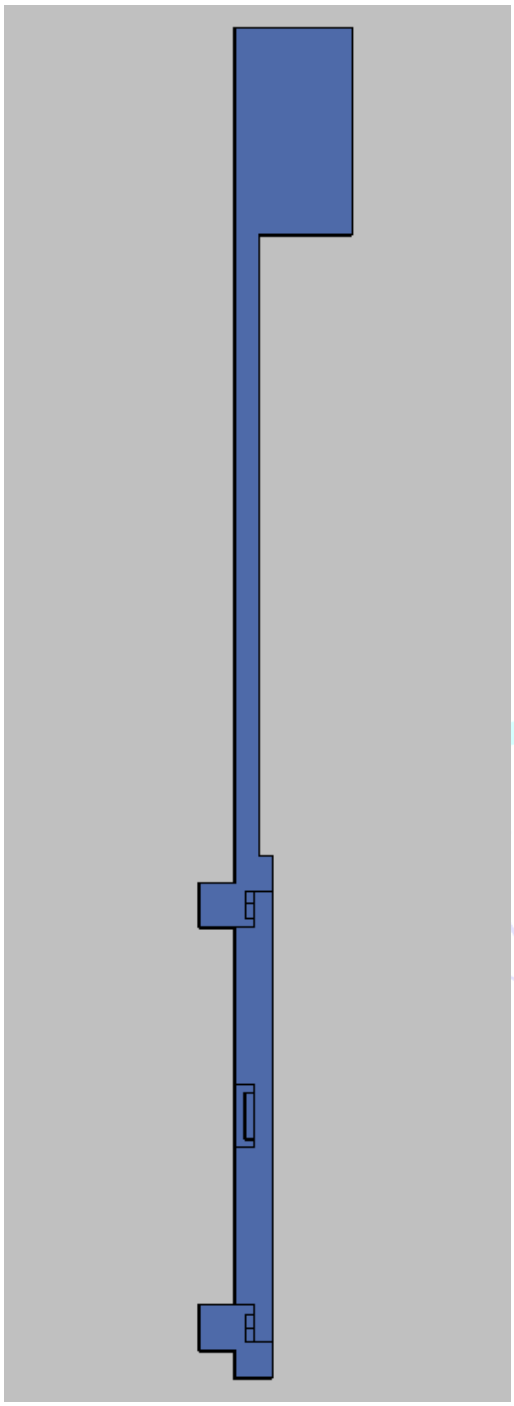
- Vista de Frente



- Vista Superior

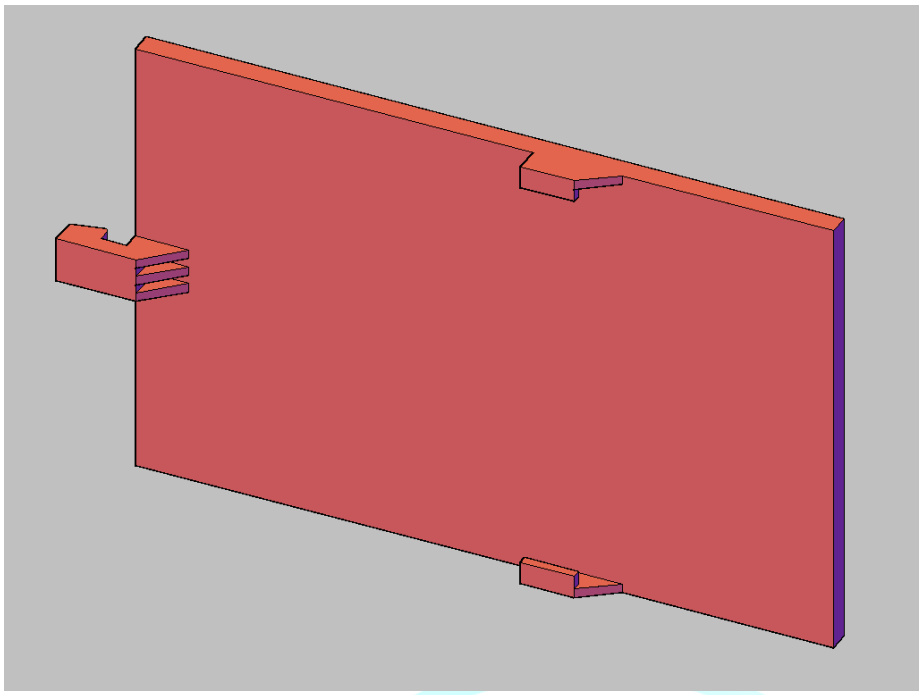


- Vista Lateral

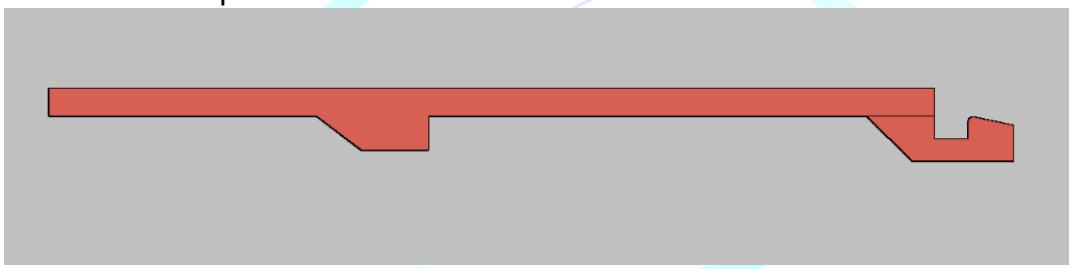


Tapa de las pilas del case:

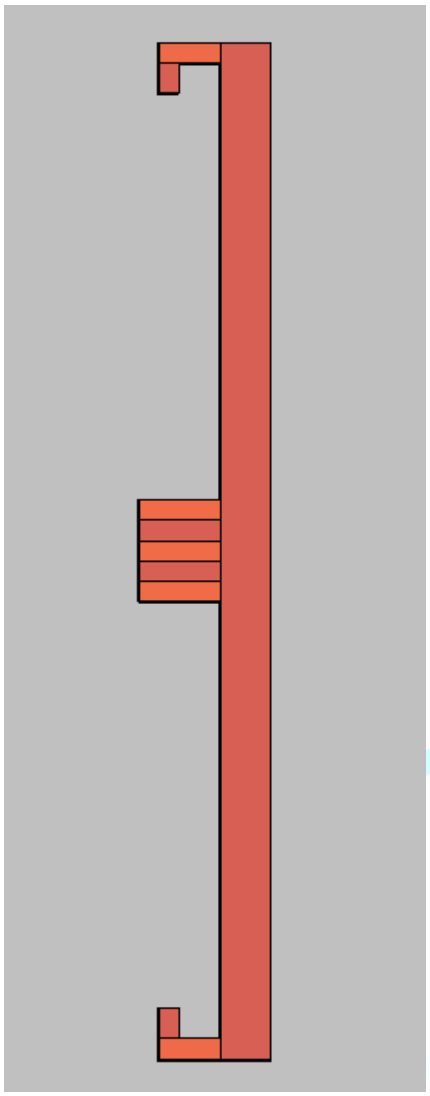
- Vista Isométrica



- Vista Superior



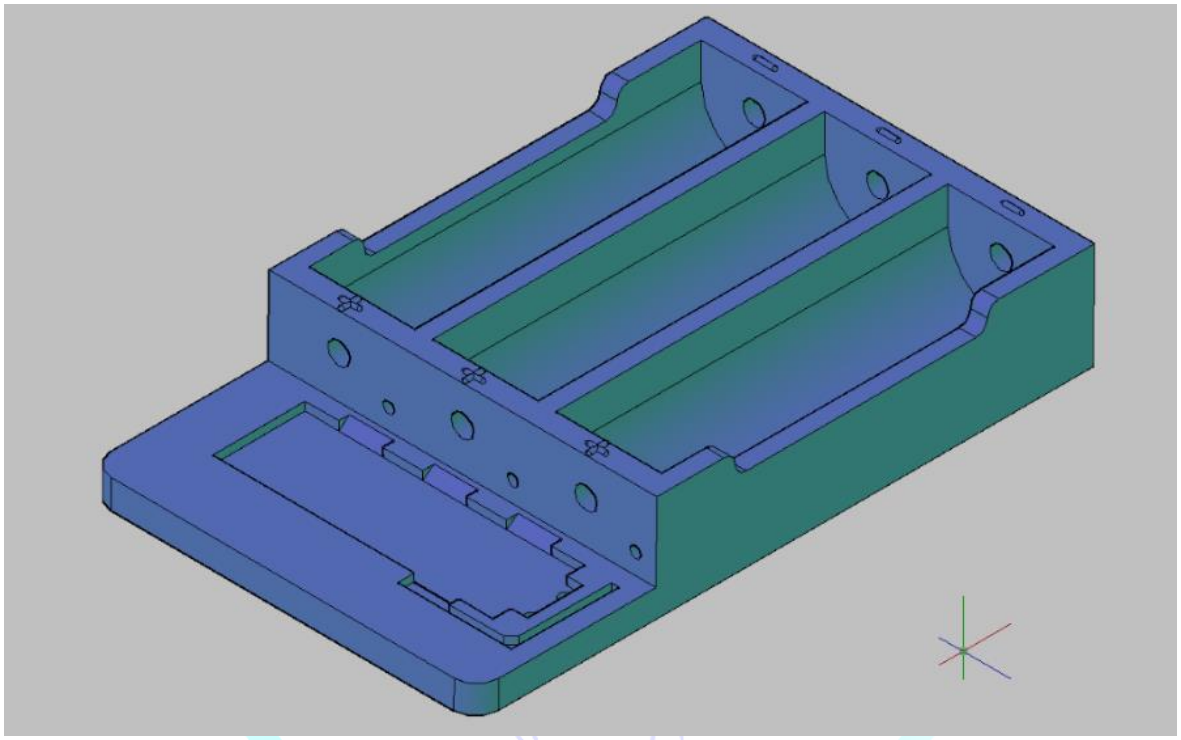
- Vista Lateral



Cargador de pilas:

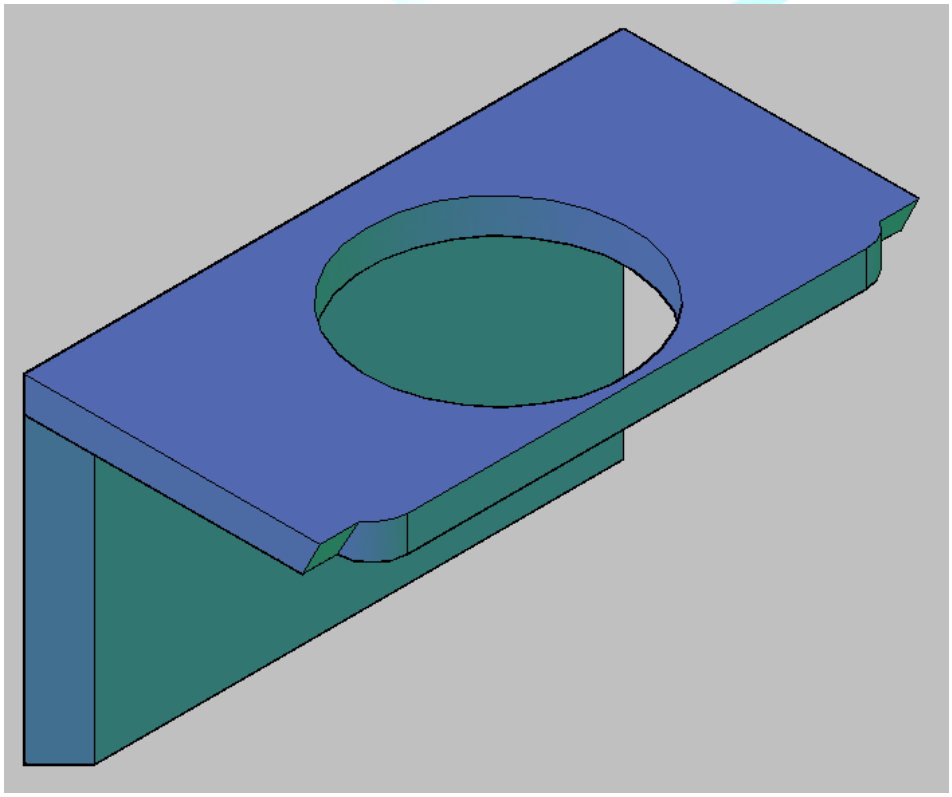
Creamos un cargador de baterías para poder cargar las pilas que utilizaría el sistema para la alimentación de la Raspberry, Arduino y elementos que haya en la placa.

- Vista Isométrica

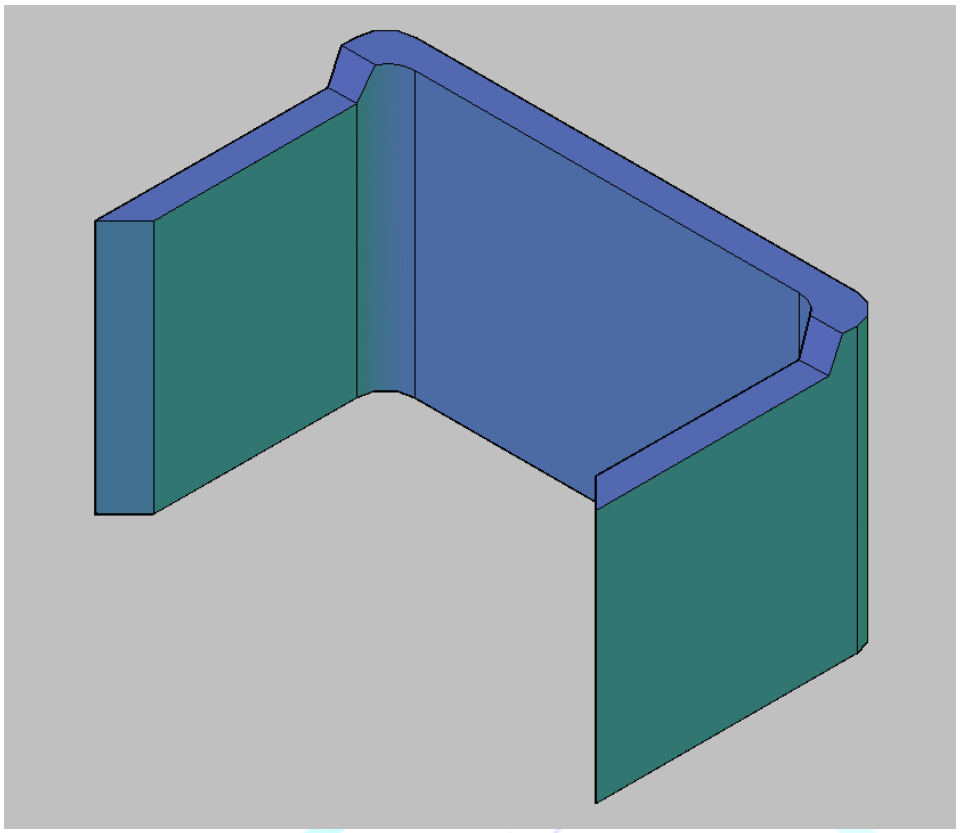


Encastre de la Ficha del Cargador

- Vista Isométrica



- Vista Isométrica



6. Anexo

6.1. Investigaciones Realizadas

ANTECEDENTES:

Hay dos emprendimientos tecnológicos importantes en el campo:

- **Antena Cyborg de Neil Harbisson**

Neil Harbisson es la primera persona “cyborg” de la historia. Definimos como cyborg a aquella persona con una cabeza que esté en constante contacto con la cibernética. Neil nació con *acromatopsia,* una enfermedad que solo lo deja ver en una escala de grises. Inspirado en movimientos artísticos, dado sus estudios en piano, decidió emprender en la idea de un implante osteointegrado, el cual coloca un microchip en el cerebro mediante una operación que vincula el oído con la antena que se ve en la foto. Mediante esta antena se analiza el espectro de colores del ambiente y se traduce en vibraciones que se diferencian según su longitud de onda. Gracias a este invento, Neil Harbisson fundó el “Movimiento Artístico Cyborg”, donde crea pinturas coloridas dependiendo de cómo se escuchen traduciendo sinfonías en cuadros.

El elevado precio y la naturaleza experimental del proyecto lo hacen inaccesible a la mayoría de la población, sumado al gran problema de tener que capacitar cirujanos de todo el mundo para que sean capaces de realizar estos implantes de vanguardia.

- **Colorino**

Colorino es un producto del cual no se encuentra mucha información en las redes. La guía de usuario nos indica que se despliegan las tarjetas que posee y su sensor compara el color de la tarjeta con el entorno, lo cual nos permite deducir el color de los objetos. Luego el resultado se anuncia por el parlante que tiene integrado. Además, cuenta con un sensor que mide si la luz es artificial o natural, lo que ayuda a la persona no vidente a saber si se deja prendida la luz de su casa y saber el estado de la luz del sol en la calle.

Este producto es más accesible y menos invasivo que el anterior. Su costo en Amazon es de 200US$. Las limitaciones deducibles de este producto son que el hecho de que haya que apuntar al objeto para saber su color complica su uso diario para el usuario no vidente, además de que un mal acomodamiento de las tarjetas puede afectar el resultado del análisis.

En pocas palabras, su uso cotidiano es viable y cómodo, pero podrían mejorarse aspectos para que sea más *user friendly*

FORMULACION DEL PROBLEMA

En Argentina, según la OMS (Organización Mundial de la Salud), son más de 900.000 las personas que sufren de discapacidades visuales que los hacen parcialmente ciegos, y 35.600 tienen una ceguera total.
Además, un 15% de la población posee incapacidad de distinguir colores con claridad debido a enfermedades congénitas como el daltonismo.
En el caso de la población daltónica, solo los que poseen un nivel leve pueden implementar lentes que corrigen esta deficiencia, cuyo valor ronda los $55.000, lo cual deja sin solución a la gente que ve en escalas de grises (acromatopsia). Hoy en día, ante la problemática que muestran estas estadísticas, no
hay ningún producto que se ofrezca en el mercado argentino (a nivel mundial hay uno solo) para poder ayudar a este sector de la población. Este sector de la población se ve con la única alternativa de aceptar que jamás podrán ser capaces de percibir el mundo de manera visualmente normal.
Nosotros proponemos como solución un sistema que mediante el uso de *Inteligencia Artificial* logre interpretar el entorno, y que mediante un sistema de vibraciones le proporcione la información al usuario no vidente de los colores de los objetos y obstáculos en su entorno, entre otros datos, cuando así lo desee.
Para lograr esto se colocará una Pi Camera y una Raspberry Pi Zero W en la mano, por lo que podemos sacar fotos y analizarlas mediante la API de *Google Vision Cloud*. Esta API ofrece la posibilidad de usar la tecnología de *Machine Learning* de Google, lo cual nos devuelve un String con la información que se halla en la imagen. Esta información se envía por Bluetooth a los sistemas de vibración distribuidos por el cuerpo.

El sistema de vibración se encarga de leer la información que nos da la Raspberry y, dependiendo del color que diga el String, vibra a menor o mayor frecuencia.
Para que este sistema sea cómodo de usar se maneja usando el *NeuroSky Mindwave 2,* una diadema que mide la actividad cerebral. Gracias a esto podemos predecir cuando el usuario desea analizar el color y solo ahí disparar el análisis y respuesta del mismo, impidiendo de este modo la posibilidad de que el sistema se encuentre trabajando constantemente y sin freno

Entrevistas realizadas

Hemos podido realizar distintas entrevistas las cuales nos sirvieron para poder verificar nuestra propuesta de valor y saber cuales son las problemáticas de las personas con discapacidad visual en el día a día y sus recomendaciones u opiniones sobre el proyecto.

- **Entrevista a Oscar:**

<u>Preocupaciones de Oscar</u>

1. Su mayor preocupación es el hecho de que no puede ser independiente y necesita mucho de su esposa para realizar algo
2. Tiene miedo de que cuando va a buscar en la heladera tirar algo. Aunque la esposa le dice donde tiene que buscar.
3. No puede desbloquear ni usar el celular solo ya que no tiene el asistente.
4. Cuando se mide la glucosa no sabe el valor que le da. La esposa le tiene que decir.
5. La computadora no la usa porque no sabe cómo manejarse en esta.
6. No sabe Braille y dice que ya es medio tarde para aprenderlo porque tiene 62 años.
7. No puede realizar compras.

<u>Opinión sobre el proyecto:</u>

- Sobre el texto dijo que sería bastante útil ya que él tiene poco manejo con la tecnología
- Propuso de que estaría bueno poder reconocer los objetos para poder manejarse solo en la calle y no tener que depender de alguien
- Sobre el color le pareció interesante y dijo que sería una buena forma de suplir su ausencia.

- **Entrevista a Lucila:**

Preocupaciones de Lucila

1. Tiene problemas para saber que dicen los tickets, cosas escritas. Billetes también es un gran problema.
2. Necesita ayuda para saber los vencimientos de productos.
3. Enseña los colores porque ellos tienen que saberlos porque viven en un mundo que se ve.
4. Problemas para saber que hay en las fotos
5. Dijo que ella ve que en cuanto a niños hay pocas cosas como para mejorar su aprendizaje y que es difícil poder comprar una maquina braille por el elevado costo y la ausencia de ayuda de parte del Estado.
6. Se viste sola, pero los colores no los sabe distinguir y le da duda e incógnita
7. En este contexto va a comprar el esposo porque si quiere ir ella necesita que alguien del supermercado o negocio la ayude.

Programas que utiliza de ayuda

Los usan para reconocer PDFs e imágenes

- Google Lens
- @ Voice Aloud.

Opinión sobre el proyecto:

- Sobre el reconocimiento de objetos, ella cree que serviría más para nenes, ya que una persona que ya tiene varios años de experiencia ya distingue bastante bien los objetos a través del tacto.
- Sobre el reconocimiento de textos: remarco que acá es un punto fuerte ya que una gran problemática para saber que dicen lo tickets, diarios, billetes, etc.
- En cuanto al reconocimiento de colores, le pareció algo bueno.

- **Entrevista a Mónica:**

Preocupaciones de Mónica:

1. No puede ir a comprar sola, ya que los supermercados no tienen ningún tipo de señalización ni información del costo del producto en braille ni nada.
2. Problemas con la conexión a videos cuando hay mensajes de alerta en el que no se informa auditivamente de ese mensaje.

3. No identifica los billetes ya que es complicado distinguir donde se encuentra el código braille y su identificación para cada billete en particular.
4. No puede hacer podas debido a los riesgos, pero le gustaría

- **Entrevista a Osvaldo**

Preocupaciones de Osvaldo

- Le molesta cuando las situaciones no tienen audios descriptivos. Por ejemplo, cuando ve la tele a veces hay silencios y quiere saber que está pasando, pero no puede
- No le gusta que la sociedad no deje pasar al ciego cuando está haciendo la fila en el banco
- Le gusta hacer los trámites, pero no puede hacerlo solo a veces porque las facturas no vienen en braille
- Le molesta mucho que la gente piense que tener una discapacidad es motivo de vivir menos alegre la vida o frustrado
- Le molesta no poder guiarse por el color de las cosas y simplemente decir que algo es tal color porque se lo dijeron y no tener un sentido asociado a eso (la vibración es importante)

6.2. Bibliografía:

- Dexter industries, “Use Google Cloud Vision on the Raspberry Pi and GoPiGo”. Recuperado de: https://www.dexterindustries.com/howto/use-google-cloud-vision-on-the-raspberry-pi/
- Google Vision, “Detectar texto en imágenes”. Recuperado de: https://cloud.google.com/vision/docs/ocr?hl=es
- Google Vision, “Detectar varios objetos”. Recuperado de: https://cloud.google.com/vision/docs/object-localizer?hl=es
- Google Vision, “Detectar propiedades de una imagen”. Recuperado de: https://cloud.google.com/vision/docs/detecting-properties?hl=es
- DataFlair, “Project in Python – Colour Detection using Pandas & OpenCV”. Recuperado de: https://data-flair.training/blogs/project-in-python-colour-detection/
- Pantech Solutions, “Blink LED with your Eye blink data using Brainwave Starter kit and Arduino”. Recuperado de: https://www.pantechsolutions.net/brain-computer-interface/blink-led-with-your-eye-blink-data-using-brainwave-starter-kit-and-arduino

- NeuroSky, “MindWave Mobile 2: User Guide”. Recuperado de: http://download.neurosky.com/public/Products/MindWave%20Mobile%202/MindWave%20Mobile%202%20User%20Guide%20.pdf
- TFG García Martin Laura 2017. Recuperado de: https://ebuah.uah.es/dspace/bitstream/handle/10017/30246/TFG%20Garc%C3%ADa%20Martin%20Laura%202017.pdf?sequence=1&isAllowed=y
- Juan Ignacio Martín Barraza – TFG. Recuperado de: https://upcommons.upc.edu/bitstream/handle/2117/78057/Juan%20Ignacio%20Mart%C3%ADn%20Barraza%20-%20TFG.pdf?sequence=1&isAllowed=y
- KiCad, Recuperado de: https://www.ecured.cu/Kicad
- Como funciona Bluetooth Low Energy: el protocolo estrella de IOT. Recuperado de: https://www.welivesecurity.com/la-es/2020/03/17/como-funciona-bluetooth-low-energy/