



# Proyecto HealthBand Aid

## Informe Descriptivo

### Integrantes:

- Acuña, Alvaro - DNI - 7mo 1ra Aviónica
- Giulianetti, Bruno - DNI - 7mo 1ra Aviónica
- Gomez, Gonzalo - DNI 46.287.726 - 7mo 1ra Aviónica
- Pagano, Tobías - DNI 46.623.445 - 7mo 1ra Aviónica
- Perlo, Mateo - DNI 45.994.321 - 7mo 1ra Aviónica



[healthband.aid](#)



[www.healthband.com.ar](#)



[healthband.aid@gmail.com](mailto:healthband.aid@gmail.com)

# Índice:

• Docentes y Tutores a Cargo	3
• Objetivos	3
• Temática	4
• Alcance	4
• Funcionalidades	4
• Beneficios	4
• Segmento de destino	5
• Ámbito de Incumbencia	5
◦ Contribución a Grupos Vulnerables	5
◦ Aporte a la Investigación Médica y Estudios Clínicos	6
• Especificaciones Técnicas	6
◦ Diseño de Identidad	6
◦ Programación	7
■ MAX30102	7
■ MPU6050	13
■ LM335	20
■ MHL590	23
◦ Interfaces Gráficas	29
■ Aplicación Móvil	29
• Frontend	29
• Backend	35
■ Página Web	35
• Frontend	35
• Backend	43
◦ Modelado 3D	52
◦ Modelado PCB	55
◦ Redes Neuronales	57
◦ Hardware Electrónica aplicada	61
■ Alimentación	61
■ Motor de vibración	61
■ Diagrama integral circuitos electrónicos	62
◦ Hosteo Web	63

# Docentes y tutores a cargo:

Nos encontramos encantados de realizar una mención especial a aquellos docentes y tutores que han colaborado activamente en la realización del proyecto tal como se lo conoce. Sin ellos, no hubiese sido posible alcanzar los logros obtenidos:

- Bianco, Carlos Cesar
- Carlassara, Fabrizio
- Medina, Sergio
- Palmieri, Diego

**Fecha de inicio del proyecto:** 01/03/2023

**Duración:** 33 semanas al día de la fecha. Con un promedio de entre 8 y 10 horas semanales de trabajo en el proyecto, se calcula unas 300 horas totales de trabajo dedicadas a la realización del mismo individualmente, y, en conjunto unas 1500 horas. Este proceso ha involucrado a un estimado de 9 personas; 5 de ellos participantes activos de la realización del mismo y otros 4 colaboradores intermitentes, entre docentes y personal externo al equipo de trabajo.

## Objetivos

HealthBand se concibe como iniciativa de proyecto de un grupo de alumnos del último año de la E.E.S.T N7 “T.R.Q.”, con el objetivo de diseñar una herramienta capaz de contribuir al progreso y mejora de la calidad de vida de las personas en el ámbito cotidiano; especialmente para aquellas cuyas condiciones de salud se convierten en una carga adicional, sea por la necesidad de asistir a chequeos médicos recurrentes o tratamientos estrictos. HealthBand pone su foco en simplificar el seguimiento de la salud para sus usuarios a través de un sistema que cuenta con un dispositivo de tipo *wearable* capaz de sensar distintos signos vitales como el ritmo cardíaco, oxigenación en sangre, temperatura corporal, movimiento y glucosa en sangre mediante diversos circuitos sensores, haciendo hincapié en el uso de métodos no invasivos pensados en la comodidad del usuario. Además del *wearable*, HealthBand cuenta con un prototipo de aplicación desarrollada para dispositivos móviles desde la cual el usuario será capaz de visualizar su estado de salud general a través de una interfaz simple y amigable. A su vez, el sistema puede emitir alertas ante la detección de alguna anomalía desde el mismo dispositivo, en forma de vibraciones, o desde el dispositivo móvil a través de asistencia por notificaciones o mensajes de tipo bot. Los datos de salud serán visibles también desde la página web. Tanto la aplicación móvil como el sitio web recibirán información desde una base de datos diseñada por el propio equipo de trabajo.

# Temática

El proyecto aborda la temática de detección de anomalías en las personas, para la prevención de futuras enfermedades y/o fallas que pueda llegar a tener el organismo humano, mediante la utilización de diferentes tecnologías como detección infrarroja para el censado de pulso, oxígeno y glucosa en sangre, a su vez del uso de redes neuronales y un muestreo de datos automático hacia una web y aplicación móvil, proponiendo un sistema innovador para la investigación médica y el desarrollo de la tecnología en el ámbito de la salud.

## Alcance

Socialmente, Healthband puede satisfacer una amplia variedad de sectores, ofreciendo beneficios a personas con diferentes condiciones y estilos de vida, desde pacientes con preexistencias médicas, trabajadores en entornos exigentes, personas físicamente activas o profesionales en el ámbito de la salud y en la industria de la tecnología.

Geográficamente, nuestro sistema de monitoreo, con la inversión necesaria y un modelo de negocio rentable, podría ser escalable a todo el mundo, ya que los problemas de salud y la necesidad de monitoreos continuos son universales. La accesibilidad y asequibilidad de la pulsera permiten que su alcance sea global, llegando tanto a lugares poblados como una capital, o también pueblos y zonas rurales con menor densidad demográfica, siempre y cuando cuenten con acceso a la salud e internet.

## Funcionalidades

Las principales funcionalidades que ofrece el sistema de Healthband a sus usuarios se basan en el monitoreo constante de pulso, oxígeno en sangre, glucosa en sangre, temperatura corporal y movimiento como parámetros de estudio, dando la posibilidad al usuario de visualizarlas a través de la sincronización del dispositivo portátil con una aplicación móvil y página web para su seguimiento a corto, mediano y largo plazo, con un registro de los mismos datos, siendo esto muy valioso al momento de detectar anomalías tempranas. El dispositivo portátil contará con la capacidad de emitir alertas para despertar la percepción y fomentar la conciencia del usuario.

## Beneficios

Los principales beneficios de Healthband son:

- Detección temprana de problemas de salud
- Mayor seguridad para las personas con condiciones médicas preexistentes
- Apoyo a un estilo de vida saludable y activo
- Mejora de la calidad de vida al permitir un monitoreo constante de la salud
- Mayor tranquilidad para familiares y cuidadores
- Aplicaciones en la investigación médica

# Segmento de destino

En la sociedad actual, la atención a la salud es una preocupación creciente, siendo que la cantidad de personas que desarrollan patologías aumentan año tras año, en gran medida producto de los cambios en el estilo de vida de la sociedad. Esto se ve agudizado en nuestro país, con un sistema de salud colapsado, situación por la cual las visitas regulares al médico pueden ser inconvenientes, costosas y complejas, lo que lleva a un seguimiento insuficiente de la salud personal. Esta falta de monitoreo continuo puede resultar en la detección tardía de problemas de salud, lo que, en algunos casos, puede tener graves consecuencias. La propuesta del ecosistema hardware - software de HealthBand se basa en un enfoque inclusivo, siendo su sistema diseñado para estar al alcance de todo aquel que desee obtenerlo, independientemente de su edad, género o condiciones de salud; desde un individuo con una enfermedad crónica, un atleta que busca optimizar su rendimiento o a cualquier persona interesada en mantener un control continuo de su bienestar general. Todos estos perfiles se beneficiarían de la comodidad y la accesibilidad que ofrece HealthBand, promoviendo una mejora en la calidad de vida de las personas al proporcionarles un medio para el cuidado constante de su salud. Esto no solo conlleva beneficios personales, sino que también puede tener un impacto positivo en el sistema de atención médica, al reducir las cargas en los servicios de emergencia y disminuir la necesidad de tratamientos médicos costosos.

## Ámbito de Incumbencia

HealthBand pone su foco en el ámbito de la salud cotidiana y, como tal, se integra de manera directa en la rutina diaria de los individuos. Su funcionalidad consiste en la adquisición continua de datos de salud relevantes, tales como el ritmo cardíaco, el nivel de oxígeno en sangre, la temperatura corporal y el nivel de glucosa en sangre. Los datos recopilados permiten la creación de un registro que refleja el estado fisiológico del usuario.

## Contribución a Grupos Vulnerables

HealthBand se presenta como una herramienta especialmente valiosa para ciertos grupos de la población, entre los cuales se encuentran:

- **Personas con Preexistencias Médicas:** Aquellos que padecen condiciones médicas preexistentes encuentran en HealthBand una solución efectiva para monitorear de manera constante su estado de salud. Esto facilita la detección temprana de anomalías y contribuye al manejo adecuado de las condiciones médicas crónicas.
- **Adultos Mayores:** La población de adultos mayores se beneficia de la monitorización diaria proporcionada por HealthBand, ya que permite el seguimiento de su salud en una instancia de vida en la que la atención a la salud puede tornarse especialmente crítica.

- **Seguimiento Personalizado:** HealthBand brinda a cualquier usuario la posibilidad de llevar a cabo un seguimiento personalizado de su salud. Esto no solo fomenta la conciencia y el autocuidado, sino que también establece la base para la detección temprana de problemas de salud en una etapa inicial, cuando la intervención puede ser más efectiva.

## Aporte a la Investigación Médica y Estudios Clínicos

Al recopilar datos de salud de manera sistemática y precisa, HealthBand sienta las bases para posibles aplicaciones en la investigación médica y la realización de estudios clínicos. La información generada a partir de la monitorización continua puede ser valiosa para la identificación de tendencias, la evaluación de la eficacia de tratamientos y el desarrollo de enfoques de atención médica más personalizados.

En resumen, HealthBand, como proyecto destinado a la monitorización de la salud cotidiana, demuestra un alcance significativo y un potencial impacto en la mejora de la atención clínica individual y la contribución a la investigación médica y estudios clínicos. Su versatilidad y aplicabilidad lo posicionan como una herramienta valiosa en la promoción de la salud y el bienestar de diversos grupos de la población.

## Especificaciones Técnicas

La creación de HealthBand ha involucrado la utilización de diversas herramientas y programas en múltiples áreas, tales como su diseño de identidad, plataformas virtuales (web y app), circuitos electrónicos, modelado 3D y programación.

### Diseño de Identidad

Para la creación de todos los componentes que llevan un diseño de identidad sólido y fiable, tales como logotipos, paletas de colores, iconos, banners y material gráfico para redes sociales, se han utilizado diversas plataformas de diseño gráfico. La confección del logotipo consistió de una etapa inicial de bocetado, una primera digitalización realizada en el programa Sketchbook y una digitalización final realizada en Illustrator, una herramienta de diseño vectorial potente desarrollada por Adobe. En este último también se han confeccionado diferentes variantes del logotipo, con el objetivo de adaptarlas a diferentes espacios de pantalla y contextos gráficos; así como el ícono de la aplicación.

**HealthBand** 

*Keep Healthy, Save Lives.*



La producción de material gráfico para redes sociales, tales como publicaciones de tipo carrusel para Instagram, posteos de tipo Historias, iconos para Historias Destacadas y banner para LinkedIn, han sido diseñados en Canva, una potente herramienta de diseño en línea, gratuita y con funciones de autoguardado. Este último también ha sido utilizado para la confección de documentación pertinente al proyecto, tales como Carpeta de Campo, Carpeta Técnica e Informe Descriptivo; así como también del banner de presentación del proyecto para su exposición, recibiendo su posterior adaptación en resolución con Illustrator.

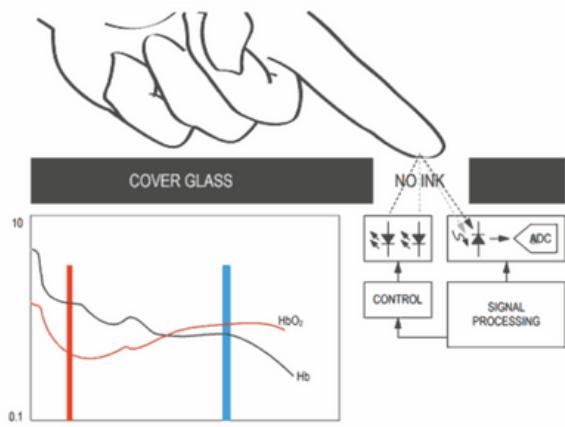
## Programación

Para la creación y prueba de códigos que interactúan con los distintos sensores del dispositivo portátil, como los de temperatura, ritmo cardíaco, movimiento y glucosa en sangre, se ha empleado una variedad de entornos de desarrollo o IDEs de carácter gratuito, siendo Thonny y Microsoft Visual Studio Code las más utilizadas. Thonny se caracteriza por su optimización en Python y Micropython, mientras que Microsoft Visual Studio Code es una IDE abierta y completa, la cual permite trabajar con una amplia gama de lenguajes de programación y ser utilizada para diversos propósitos. En menor medida se han probado códigos en la IDE de Arduino.

## MAX30102

### Módulo Sensor de Ritmo Cardíaco y Oxigenación en Sangre

Para la medición del ritmo cardíaco y oxigenación en sangre, HealthBand utiliza un módulo integrado, el cual se puede encontrar en el mercado con el nombre comercial de MAX30102, desarrollado por Maxim Integrated. El comportamiento de este sensor integrado es de tipo óptico, basado en la reflexión de señales de 2 LEDS, uno infrarrojo y otro de luz visible, sobre la piel humana, midiendo posteriormente la energía generada por la saturación sobre su receptor, la cual dependerá de la composición de la sangre y la cantidad de partículas presentes en la misma, siendo la hemoglobina de interés para el caso de este circuito, ya que presenta distintas propiedades de reflexión cuando está cargada de oxígeno (absorbiendo mayor luz infrarroja) que cuando no lo está (absorbiendo mayores cantidades de luz visible). La luz reflejada es recibida por fotodiodos, los cuales conducen energía eléctrica de forma proporcional al nivel de luz que reciben. Trabaja con una frecuencia de muestreo (lecturas de señal reflejada) de 50 sps (samples per second). También dispone de la electrónica necesaria para la amplificación y filtrado de la señal, cancelación de luz ambiental y compensación de temperatura. Es de importancia mencionar que durante las pruebas, ha presentado sensibilidad a la presión ejercida sobre su lente.



Representación gráfica del comportamiento de la hemoglobina a la radiación de distintos espectros de luz junto a un esquema de medición sobre el dedo.

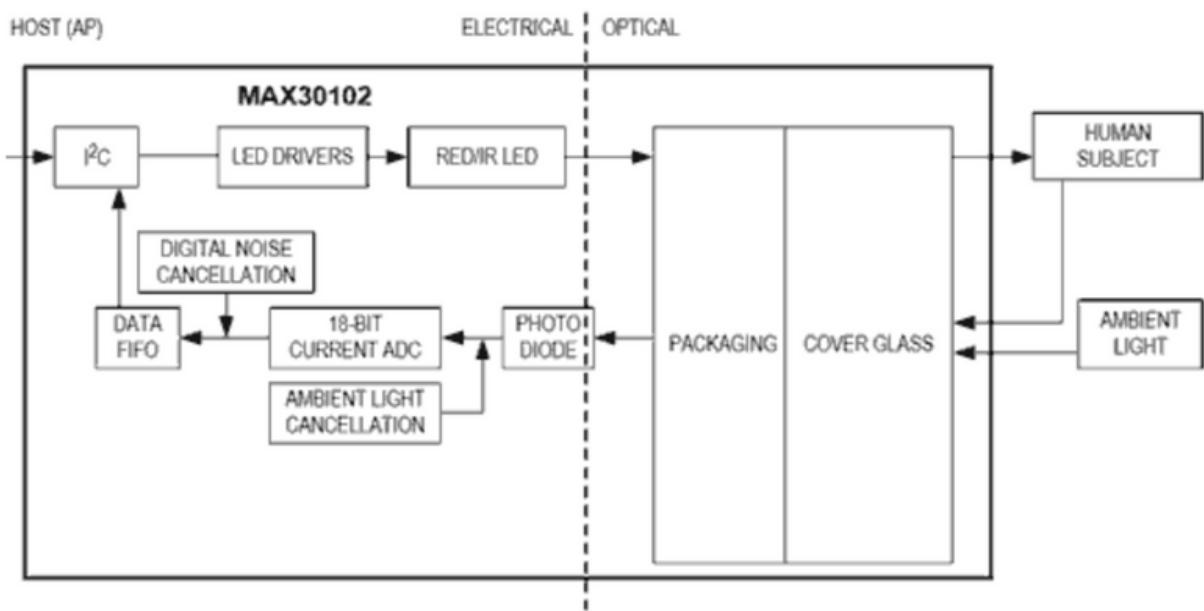


Diagrama electrónico del sensor MAX30102

La conexión entre el sensor y el microcontrolador es bastante sencilla, debiendo simplemente alimentarse el módulo desde el ESP32 mediante GND y 3.3V y conectando los pines SDA y SCL del microcontrolador con los pines de I<sup>2</sup>C correspondientes del MAX30102.

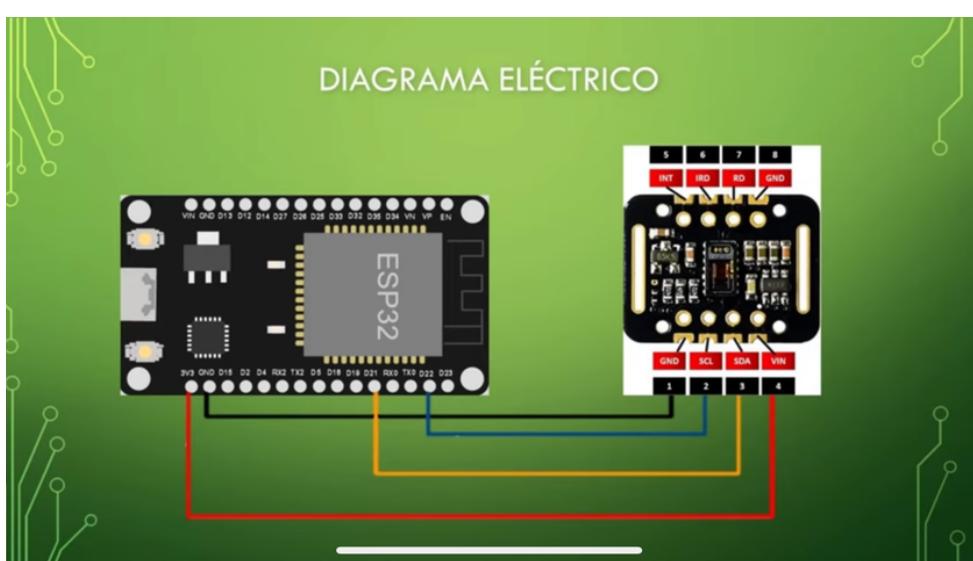


Diagrama de conexiones entre MAX30102 y ESP32

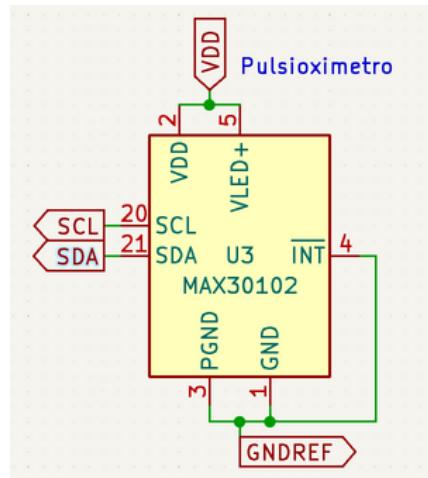
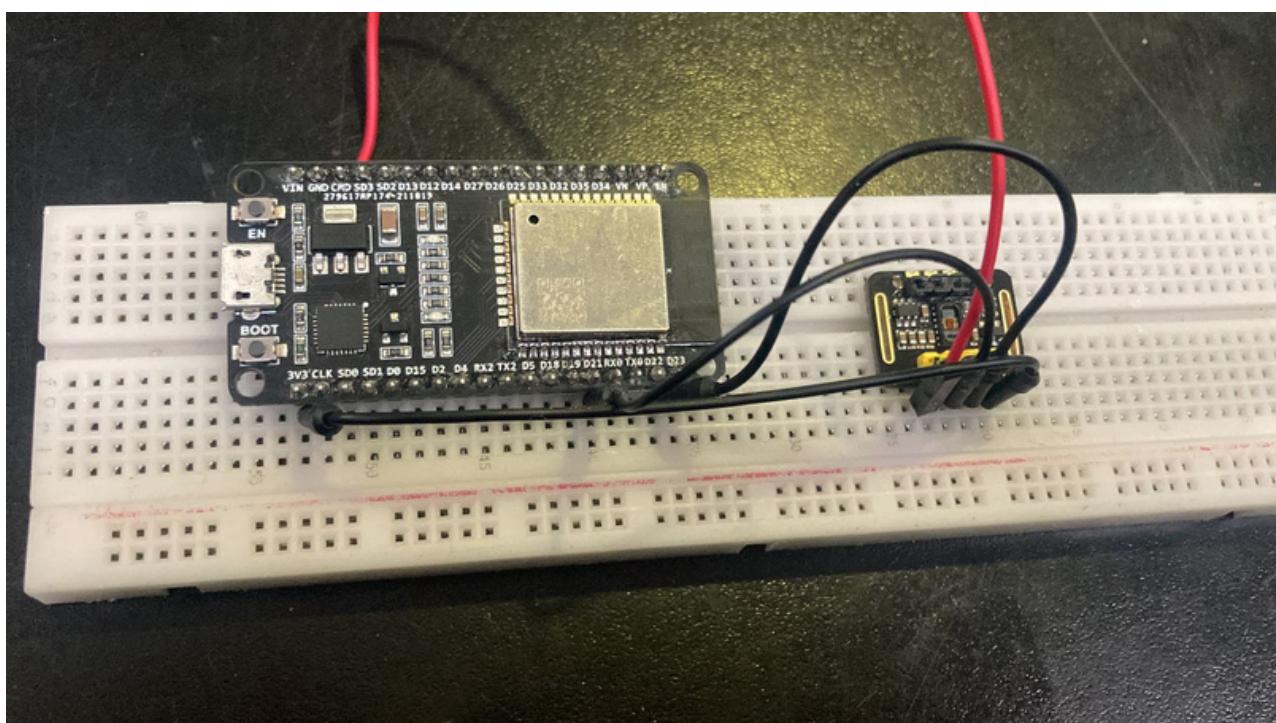


Diagrama de conexiones entre el MAX30102 y la ESP32 hecho en Kicad. Las etiquetas representan a que pines del microcontrolador van conectados los del pulsioxímetro.



Conexión entre el MAX30102 y la ESP32 para su etapa de pruebas en protoboard

Su programación fue realizada en Micropython, en la IDE Thonny. Utilizando librerías (`circular_buffer.py` y `max30102.py`) y módulos propios del mismo lenguaje (`oxi.py` y `pulsometro.py`) para la realización de cálculos y calibración, así como librerías adaptadas para este sensor. A continuación, el código implementado:

```
circular_buffer.py × max30102.py × oxi.py × pulsometro.py ×
```

```
import pulsometro
#se importa la librería nexus del max la cual contiene los cálculos y funciones para
medir pulso y oxígeno
from pulsometro import Pulso
#de pulsometro se importa la clase pulso para futuras funciones
from machine import Timer
#se importa la libreria de timer
import _thread
# se importan los hilos para hacer multihilos en simultaneo (varias tareas)
import time
# se importa la libreria de time para utilizar funciones de tiempo
import urequests as requests
#se importa la funcion urequests y se le pide que se llame "requests"
import network
#se importa la funcion network
import ujson as json
#se importa la funcion ujson y se le pide que se llame "json"

def conexion():
    #se crea la funcion conexion
    sta_if = network.WLAN(network.STA_IF)
    #se crea la variable sta_if y se le dice que va a ser una red inalambrica
    (network.WLAN). se comporta como un cliente de red asi puede conectarse a un WIFI
    if not sta_if.isconnected():
        print('Estableciendo conexión con la red')
        #se muestra en la pantalla un mensaje el cual dice "Estableciendo conexión con la
red"
        sta_if.active(True)
        #Se declara que la variable STA_IF se pueda conectar a redes wifi
        sta_if.connect('Red Alumnos', 'Educar_2023')
        #aqui se declara primero la red a conectarse y luego la contraseña
        while not sta_if.isconnected():
            # si no se establece ninguna conexión, dejarla pasar
            pass
        print('Conexión establecida')
        #cuando se establece la conexión aparece un mensaje de "conexión establecida"
    conexion()
    #se ejecuta la función conexión

Sensor = Pulso()
#Se declara la variable sensor que es = a pulso(), la cual es una clase establecida en
pulsometro.py

def test():
    #se declara la función test para luego llamar a la función muestra de pulsometro.py
    Sensor.muestra()
```

```

def categorizar_pulsaciones(pulsaciones):
    #se declara la funcion categorizar_pulsaciones con el objetivo de brindar un mensaje
    dependiendo lo medido
    if 60 <= pulsaciones <= 80:
        #Si es mayor o igual a 60 o menor o igual a 80, son pulsaciones normales
        return "Pulsaciones normales"
    elif 85 <= pulsaciones <= 90:
        #Si es mayor o igual a 85 o menor o igual a 90, son pulsaciones con ejercicio
        moderado
        return "Ejercicio moderado"
    elif pulsaciones >= 100:
        #Si es mayor o igual a 100 son pulsaciones con ejercicio intenso
        return "Ejercicio intenso"
    elif pulsaciones < 40:
        #Si es menor o igual a 40 son pulsaciones criticamente bajas
        return "Pulso criticamente bajo"
    else:
        return "Pulsaciones bajas"

def categorizar_o2(Spo2):
    #se declara la funcion categorizar_o2 con el objetivo de brindar un mensaje
    dependiendo lo medido
    if Spo2 < 95:
        #si es menor a 95 hay baja oxigenacion
        return "Baja Oxigenacion"
    elif Spo2 > 100:
        #si es mayor a 100, entonces seria una muestra no valida ya que es imposible estar
        a un 101%, por lo contrario, es una buena oxigenacion.
        return "Muestra no valida"
    else:
        return "Buena Oxigenacion"

def mediciones():
    #Se declara la funcion mediciones
    while True:
        #se declara un bucle, el cual es infinito hasta que se interrumpa manualmente
        pulsaciones = Sensor.datos
        #Se declara la variable pulsaciones, la cual almacena los datos del Sensor Bpm
        (pulsaciones)
        Spo2= Sensor.datos2
        #Se declara la variable pulsaciones, la cual almacena los datos del Sensor Spo2
        (Oxigeno)
        mensaje = categorizar_pulsaciones(pulsaciones), categorizar_o2(Spo2)
        #Se declara la variable mensaje la cual toma los datos de pulsaciones y Spo2 y devuelve
        uno de los mensajes declarados arriba en las categorizaciones

```

```

print("Pulsaciones: ",pulsaciones," BPM", " Spo2: ",Spo2, "%")
    #El bucle muestra lo medido en Pulsaciones y Spo2
    print(mensaje)
    #El bucle muestra el resultado de la categorización de Pulsaciones y Spo2
    time.sleep(4)
        #El código espera 4 segundos para volver a iniciar el bucle nuevamente
time.sleep(1)
    #El código espera 1 segundo para volver a iniciar la función test
    _thread.start_new_thread(test, ())
    time.sleep(1)
    #El código espera 1 segundo para volver a iniciar la función mediciones
    _thread.start_new_thread(mediciones, ())

def web():
    #Se declara una función llamada web para lo que sería el traspaso de datos sensados
    pulsaciones = Sensor.datos
    #Se vuelve a llamar a la variable pulsaciones que contiene lo medido en BPM
    Spo2= Sensor.datos2
    #Se vuelve a llamar a la variable Spo2 la cual contiene lo medido en Spo2
    if Sensor.datos > 40 and Sensor.datos < 130 and Sensor.datos2 > 50 and
Sensor.datos2 < 110:
        #Se le dice a la función que SI Bpm esta entre 40 y 110 Y que SI Spo2 esta entre 50
        y 110, entonces ejecute lo siguiente
        info = { 'PULSOS' : pulsaciones , 'OXIGENO' : Spo2 }
        #Se declara la variable info que contiene Pulsaciones y Spo2
        g = requests.post('http://192.168.0.51:8080/datos/', json=info)
        #Se hace un request.post, el cual envía los datos a la página web en forma de
        json, el cual este llama a la variable info
        print (g.text)
        #Se muestra en la pantalla lo enviado a la página web
    time.sleep (2)
    #El código espera 2 segundos para volver a ejecutar el if
while True:
    #Se crea un bucle para que web se ejecute constantemente
    web()
    time.sleep (2)
    #El código espera 2 segundos para volver a ejecutar web

```

Los códigos de las librerías se encuentran en el repositorio del Github de nuestro proyecto

## MPU6050 Módulo de Monitor de Actividad y Detector de Caídas

El movimiento es un factor valioso para determinar el estado de una persona. HealthBand utiliza para ello un sensor MPU6050 . Este es una IMU (Unidad de Medición Inercial) que incorpora funciones de giróscopo y acelerómetro en los 3 ejes X, Y y Z, lo que le otorga 6 grados de libertad, 3 para cada función antes descripta. Compuesto internamente de sistemas Microelectromecánicos (MEMS), es capaz de sensar la aceleración de cuerpos de masa constante, con la particularidad de que también presenta sensibilidad frente a la aceleración de la gravedad. Conociendo la aceleración es también posible determinar la velocidad y el desplazamiento del cuerpo en cuestión, aplicando diversos cálculos de integración.

La velocidad angular es un término que refiere a que tan rápido un cuerpo rota sobre su propio eje. Los MEMS del MPU6050 utilizan el principio del Efecto Coriolis, el cual describe los desplazamientos en trayectorias de diversos agentes producto de las fuerzas de rotación terrestre, para calcular velocidad angular.

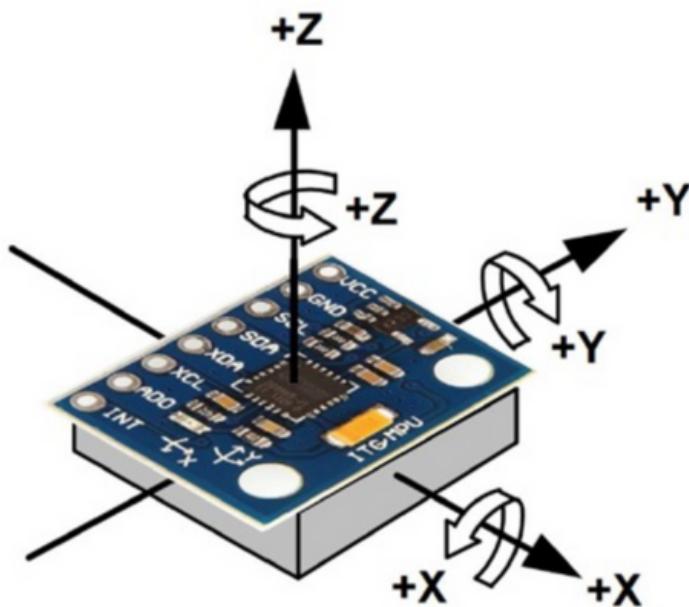


Gráfico descriptivo de los 3 ejes con los que trabajan las unidades integradas del MPU6050. Las flechas negras describen el modo de operación del acelerómetro; mientras que las blancas el sensado del giróscopo.

La comunicación entre el MPU6050 y la ESP32 se realizará, al igual que en el módulo de medición de ritmo cardíaco y oxigenación en sangre, mediante el protocolo I2C. En términos de Hardware, este se reserva los pines SCL y SDA para la transmisión de datos, conectándose a sus pares del microcontrolador. El microcontrolador es también el que le proporciona al MPU6050 la alimentación necesaria para su funcionamiento, por lo que el esquema de conexiones es representado en el siguiente gráfico:

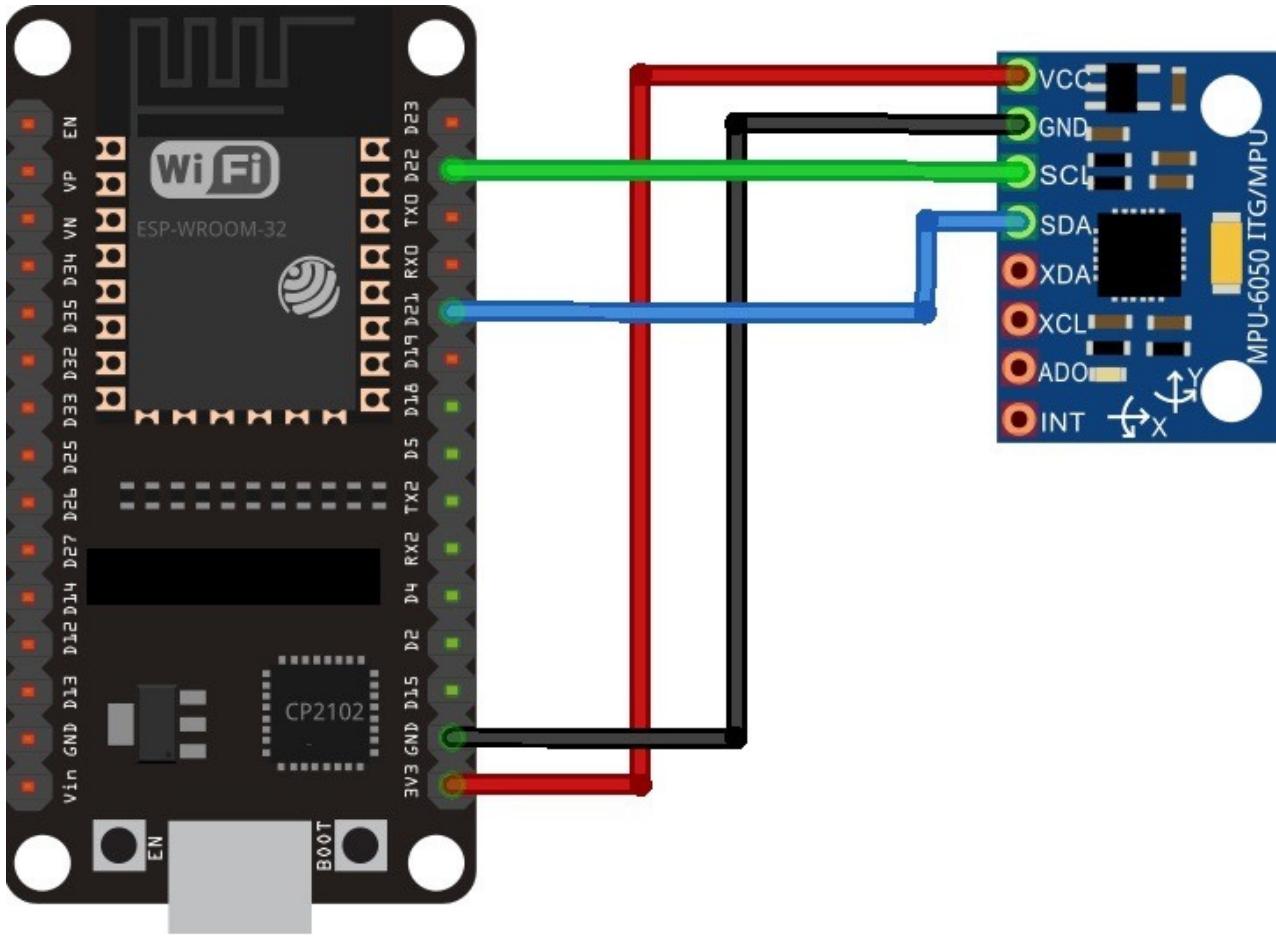
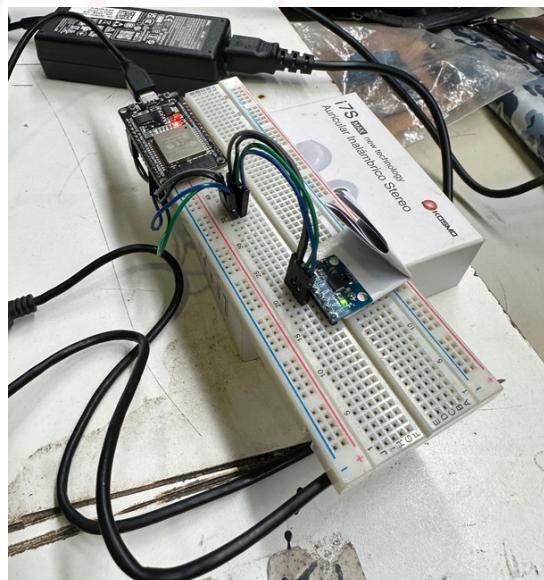
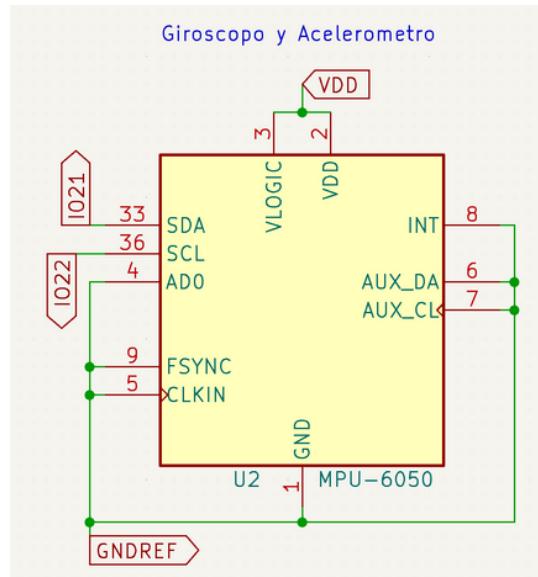


Diagrama de conexión para el módulo de monitoreo de actividad y detección de caídas. En el ESP32, el pin 21 está reservado para SDA, mientras que el pin 22 lo está para SCL.



Conecciones simuladas en protoboard durante la etapa de prueba del módulo de giroscopio y acelerómetro.



Esquema de conexión en Kicad, con sus respectivas entradas y salidas etiquetadas correspondientemente con su pin de la ESP32.

Para su programación y medición de datos se ha utilizado el lenguaje de programación Micropython, thonny Esta incluye librerías específicas para la comunicación, calculo y obtencion de las mediciones , siendo los códigos proporcionados a continuación:

```

from machine import Pin, I2C
import time
import math

MPU_ADDR = 0x68
num_samples = 100
#Número de muestras para calibración
calibration_time = 4
#Tiempo para calibrar los puntos fijos (en segundos)

i2c = I2C(scl=Pin(22), sda=Pin(21))

# Inicializar el sensor MPU6050
i2c.writeto_mem(MPU_ADDR, 0x6B, bytes([0]))

# Definir los puntos fijos iniciales
points = {'A': (0, 0, 0), 'B': (0, 0, 0), 'C': (0, 0, 0)}

def calibrate_points():
    global points

    for point in points:
        print("Coloque el sensor en el punto fijo '{}'".format(point))
        time.sleep(calibration_time)
        time.sleep(calibration_time)

```

```

i2c.writeto(MPU_ADDR, bytes([0x43]))
# Comenzar con el registro 0x43 (GYRO_XOUT_H)
data = i2c.readfrom(MPU_ADDR, 6)
# Leer 6 registros

gyro_x = (data[0] << 8) | data[1]
gyro_y = (data[2] << 8) | data[3]
gyro_z = (data[4] << 8) | data[5]

# Convertir los valores a grados por segundo
gyro_x_deg = convert_raw_to_degrees(gyro_x)
gyro_y_deg = convert_raw_to_degrees(gyro_y)
gyro_z_deg = convert_raw_to_degrees(gyro_z)

# Actualizar los puntos fijos con los valores calibrados
points[point] = (gyro_x_deg, gyro_y_deg, gyro_z_deg)

def calculate_relative_motion():
    while True:
        i2c.writeto(MPU_ADDR, bytes([0x43]))
        # Comenzar con el registro 0x43 (GYRO_XOUT_H)
        data = i2c.readfrom(MPU_ADDR, 6)
        # Leer 6 registros

        gyro_x = (data[0] << 8) | data[1]
        gyro_y = (data[2] << 8) | data[3]
        gyro_z = (data[4] << 8) | data[5]

        # Convertir los valores a grados por segundo
        gyro_x_deg = convert_raw_to_degrees(gyro_x)
        gyro_y_deg = convert_raw_to_degrees(gyro_y)
        gyro_z_deg = convert_raw_to_degrees(gyro_z)

        # Calcular el desplazamiento relativo a los puntos fijos
        displacement = {'A': [0, 0, 0], 'B': [0, 0, 0], 'C': [0, 0, 0]}
        for point in points:
            displacement[point][0] = gyro_x_deg - points[point][0]
            displacement[point][1] = gyro_y_deg - points[point][1]
            displacement[point][2] = gyro_z_deg - points[point][2]

        # Imprimir el desplazamiento relativo para cada punto
        for point in displacement:
            print("Desplazamiento relativo desde el punto '{}': X={}, Y={}, Z={}".format(point,
displacement[point][0],
displacement[point][1],
displacement[point][2]))

        # Esperar 4 segundos antes de la siguiente lectura
        time.sleep(4)

```

```
# Función de conversión de valores brutos a grados por segundo
def convert_raw_to_degrees(raw_value):
    gyro_sensitivity = 131.0
    # Sensibilidad del giroscopio
    deg_per_sec = raw_value / gyro_sensitivity
    return deg_per_sec

# Calibrar los puntos fijos
calibrate_points()

# Calcular el desplazamiento relativo a los puntos fijos
calculate_relative_motion()

MPU_ADDR = 0x68
num_samples = 100
# Número de muestras a promediar para la calibración

gyro_offset_x = 0
gyro_offset_y = 0
gyro_offset_z = 0

i2c = I2C(scl=Pin(22), sda=Pin(21))

# Inicializar el sensor MPU6050
i2c.writeto_mem(MPU_ADDR, 0x6B, bytes([0]))

def calibrate_gyro_offsets():
    global gyro_offset_x, gyro_offset_y, gyro_offset_z

    for _ in range(num_samples):
        i2c.writeto(MPU_ADDR, bytes([0x43])) # Comenzar con el registro 0x43 (GYRO_XOUT_H)
        data = i2c.readfrom(MPU_ADDR, 6) # Leer 6 registros

        gyro_offset_x += (data[0] << 8) | data[1]
        gyro_offset_y += (data[2] << 8) | data[3]
        gyro_offset_z += (data[4] << 8) | data[5]

    # Retardo entre muestras
    time.sleep(0.01)

    gyro_offset_x = gyro_offset_x // num_samples
    gyro_offset_y = gyro_offset_y // num_samples
    gyro_offset_z = gyro_offset_z // num_samples

def convert_raw_to_degrees(raw_value):
    gyro_sensitivity = 131.0 # Sensibilidad del giroscopio
    deg_per_sec = raw_value / gyro_sensitivity
    return deg_per_sec
# Calibrar los valores del giroscopio
calibrate_gyro_offsets()
```

```

while True:
    i2c.writeto(MPU_ADDR, bytes([0x43])) # Comenzar con el registro 0x43 (GYRO_XOUT_H)
    data = i2c.readfrom(MPU_ADDR, 6) # Leer 6 registros

    gyro_x = (data[0] << 8) | data[1]
    gyro_y = (data[2] << 8) | data[3]
    gyro_z = (data[4] << 8) | data[5]

    # Restar el offset calibrado a los valores brutos
    gyro_x -= gyro_offset_x
    gyro_y -= gyro_offset_y
    gyro_z -= gyro_offset_z

    # Convertir los valores a grados por segundo
    gyro_x_deg = convert_raw_to_degrees(gyro_x)
    gyro_y_deg = convert_raw_to_degrees(gyro_y)
    gyro_z_deg = convert_raw_to_degrees(gyro_z)

    # Imprimir los valores calibrados
    print("Giroscopio (°/s): X={}, Y={}, Z={}".format(gyro_x_deg, gyro_y_deg, gyro_z_deg))

    # Retardo antes de la siguiente lectura
    time.sleep(1.0)

MPU_ADDR = 0x68 # Dirección I2C del MPU-6050. Si el pin AD0 está en HIGH, la dirección I2C
será 0x69.
accelerometer_x, accelerometer_y, accelerometer_z = 0, 0, 0 # Variables para datos brutos del
acelerómetro
gyro_x, gyro_y, gyro_z = 0, 0, 0 # Variables para datos brutos del giroscopio
temperature = 0 # Variable para datos de temperatura

i2c = I2C(scl=Pin(22), sda=Pin(21)) # Configurar el bus I2C

# Inicializar el sensor MPU6050
i2c.writeto_mem(MPU_ADDR, 0x6B, bytes([0])) # PWR_MGMT_1 register (wake up the MPU-
6050)

def convert_int16_to_str(i):
    return '{:6d}'.format(i)

while True:
    i2c.writeto(MPU_ADDR, bytes([0x3B])) # Comenzar con el registro 0x3B (ACCEL_XOUT_H)
    data = i2c.readfrom(MPU_ADDR, 14) # Leer 14 registros

    # Leer y convertir los datos del acelerómetro
    accelerometer_x = (data[0] << 8) | data[1]
    accelerometer_y = (data[2] << 8) | data[3]
    accelerometer_z = (data[4] << 8) | data[5]

```

```
# Leer y convertir los datos del acelerómetro
```

```
accelerometer_x = (data[0] << 8) | data[1]
```

```
accelerometer_y = (data[2] << 8) | data[3]
```

```
accelerometer_z = (data[4] << 8) | data[5]
```

```
# Leer y convertir los datos de temperatura
```

```
temperature = (data[6] << 8) | data[7]
```

```
# Leer y convertir los datos del giroscopio
```

```
gyro_x = (data[8] << 8) | data[9]
```

```
gyro_y = (data[10] << 8) | data[11]
```

```
gyro_z = (data[12] << 8) | data[13]
```

```
# Imprimir los datos
```

```
print("Aceleración (g): X={}, Y={}, Z={}".format(accelerometer_x, accelerometer_y,
```

```
accelerometer_z))
```

```
print("Giroscopio (°/s): X={}, Y={}, Z={}".format(gyro_x, gyro_y, gyro_z))
```

```
# Retardo antes de la siguiente lectura
```

```
time.sleep(5)
```

El programa otorga en primera instancia un tiempo de ajuste inicial de 4 segundos para la inicialización del modulo en un determinado punto. La posición inicial se define en un diccionario que contiene 3 arrays y recibe el nombre de points. Luego es declarada como una variable global.

También es declarada una dirección de I2C en la variable MPU\_ADDR. Esto será imprescindible para la lectura de datos y la diferenciación del sensor MAX30102, ya que este también utiliza el protocolo de comunicación.

Luego se utiliza un bucle infinito para calcular el desplazamiento relativo a la posición inicial, donde se leen 6 registros de I2C, los cuales sumados entre si permiten obtener 3 valores enteros. Cada uno es almacenado en una variable relativa a la velocidad angular, y sus datos enteros son convertidos a angulares. Posteriormente se le resta uno de los elementos del array points a cada uno, consiguiendo la informacion de desplazamiento relativo. Tras 4 segundos de espera, se vuelve a calibrar los puntos de inicio y el ciclo descripto se repite.

Para calibrar el offset del sensor, se recorre tantas veces la frecuencia de muestreo los 6 registros del MPU6050, y se suman de la misma forma que en la obtencion de desplazamiento relativo para obtener otras 3 variables. Esos valores crudos son convertidos a grados por segundo al dividirlos por la sensibilidad del modulo. Tras un retardo de 1 segundo, el ciclo de calibracion de offset vuelve a ejecutarse.

En total se utilizan 14 registros: 6 para medición de velocidad angular, 6 para aceleración y 2 para medición de temperatura interna del modulo. Para mas información consulte en nuestro repositorio de GitHub, donde podra encontrar el código y sus librerías detalladas.

## LM335

### Módulo Sensor de Temperatura

El cuerpo humano suele regular su normal funcionamiento en una temperatura que ronda entre los 36.5 C y 37 C. Un desplazamiento de este rango puede ser signo de alerta ante la aparición de enfermedades, infecciones o posibles accidentes de salud tales como hemorragias. Un caso de temperatura más baja es conocido como hipotermia, y sus causas pueden deberse a una exposición a temperaturas muy bajas o hemorragias; mientras que una subida de temperatura por encima del rango antes mencionado puede implicar sensaciones de fiebre, generalmente asociadas a enfermedades de vía respiratoria, pero también a infecciones que deben ser tratadas con rapidez para evitar complicaciones mayores de salud.

Para la medición de temperatura corporal, el *wearable* de HealthBand utiliza un sensor LM335. Su comportamiento electrónico es similar al de un diodo zener, presentando una respuesta lineal de 10mv por grado Kelvin ante las variaciones de temperatura, por lo que será necesario realizar una conversión de la información a grados Celsius y simplificar las mediciones.

Entre sus especificaciones técnicas, trabaja en un rango de temperatura que va desde los 233.15 K (-40 C) a los 373.15 (100 C) y una tolerancia de 1% a partir de los 25 C.

Su configuración electrónica consiste en la colocación de una resistencia entre la alimentación y el pin positivo del sensor con el objetivo de polarizar el diodo y limitar su corriente, que según la hoja de datos, es capaz de soportar hasta 5mA. Este mismo terminal es sobre el cual se mide la salida de tensión del LM335 para luego ser enviada a un pin analógico - digital del microcontrolador. Para la calibración de las mediciones se coloca una resistencia paralela, entre la salida y masa.

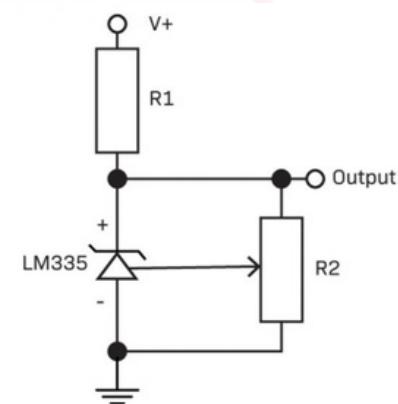
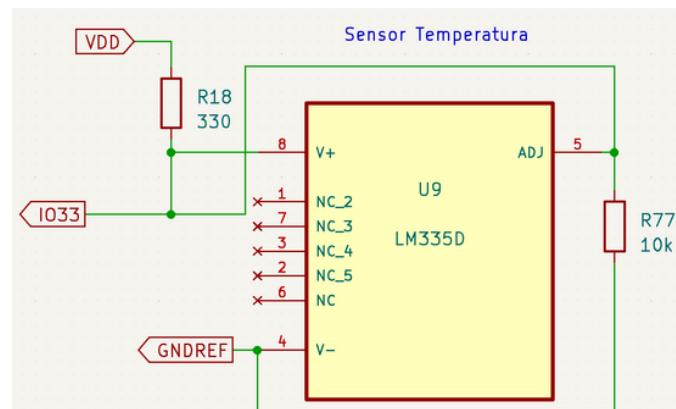


Diagrama de conexiones del circuito sensor de temperatura realizado en Kicad.



Su programación se ha realizado en Python, con el código proporcionado a continuación:

```
from machine import Pin, ADC
from time import sleep
import urequests as requests
import network
import ujson

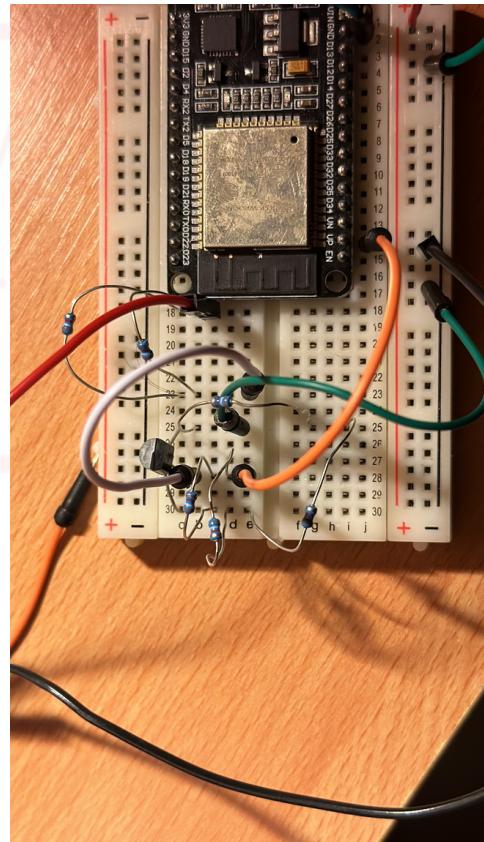
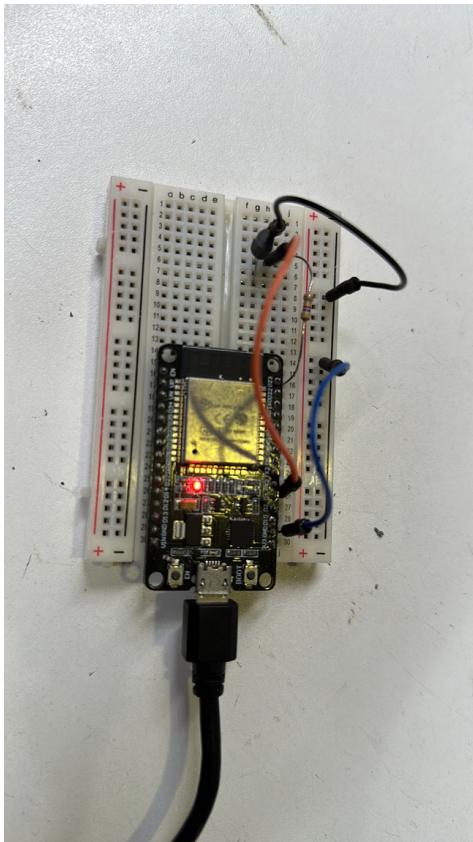
def conexion(): #se crea la funcion de conexion
    sta_if = network.WLAN(network.STA_IF) #se almacena una red inalambrica WLAN en
    #una variable y se le asigna un comportamiento de cliente para poder conectarse a wifi.
    if not sta_if.isconnected():
        print('Estableciendo conexion con la red') #se muestra en la pantalla un mensaje el cual
        #dice "Estableciendo conexion con la red"
        sta_if.active(True) #Se declara que la variable STA_IF se pueda conectar a redes wifi
        sta_if.connect('Red Alumnos', 'Educar_2023') # se pasan como argumentos de la
        #funcion la red wifi y su contraseña
        while not sta_if.isconnected(): # si no se establece ninguna conexion, dejarla pasar.
            pass
        print('Conexion Establecida') #mensaje luego de establecerse la conexión
    conexion() #se ejecuta la funcion conexion
```

```
lm = Pin(25, Pin.OUT)
adc = ADC(Pin(33, Pin.IN))
adc.atten(ADC.ATTN_11DB)
#se configuran los pines de entrada y salida del microcontrolador y la atenuacion del
#ADC

while True:
    lm.value(1) #establece el estado alto para que la salida alimente al sensor
    temp = adc.read() #lectura del pin ADC
    kelvin = temp * 0.1221 # conversion del valor ADC a escala Kelvin
    celsius = kelvin - 273.15 #conversion Kelvin a Celsius
    print(celsius)
    sleep(2)
```

```
def web(): #se crea la funcion para realizar el posteo web  
temperatura =celsius  
info = {"TEMPERATURA" : temperatura} #se crea el diccionario y se le asigna el valor de  
la variable temperatura  
g = requests.post('https://reqres.in/api/users', json=info) #se realiza el posteo y se pasa el  
diccionario como un json  
print (g.text) #se imprime por consola para visualizar  
time.sleep (2) #El codigo espera 2 segundos para volver a ejecutar el if  
while True: #Se crea un bucle para que web se ejecute constantemente  
    web()  
    time.sleep (2) #El codigo espera 2 segundos para volver a ejecutar web
```

Para mas informacion sobre las librerias utilizadas consulte a nuestro repositorio de Github



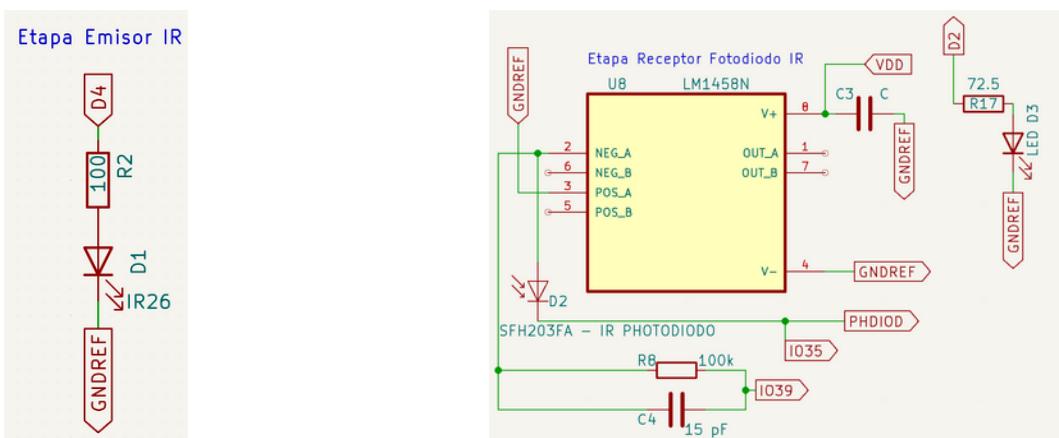
Pruebas en Protoboard del circuito sensor de Temperatura

## MHL590

### Módulo Sensor no Invasivo para la Medición de Glucosa en Sangre

La diabetes es una de las grandes epidemias de la actualidad. Se estima que entre el año y la actualidad la cantidad de personas diabéticas alrededor del mundo ha crecido un %, y se estima que para el año lo haga en otro %. Gracias a los avances de la ciencia, las personas diabéticas cuentan con métodos efectivos de seguimiento de su salud. Sin embargo, estos no resultan particularmente cómodos, ya que la gran mayoría de dispositivos medidores de glucosa en sangre disponibles en el mercado implican algún tipo de invasión en el cuerpo humano a través de pinchazos que permitan al dispositivo tener un contacto directo con la sangre. HealthBand buscó desarrollar durante su proceso de trabajo un sistema experimental de medición de glucosa no invasiva, basándose en modelos de reflexión de señales infrarrojas. El modelo de glucómetro no invasivo consta de un emisor infrarrojo de modelo MHL590, el cual emite un haz de luz infrarroja de longitud de onda de 940 nm. Esta señal es reflejada en la piel humana de forma proporcional a la composición de la sangre en niveles intersticiales, siendo la proporción de partículas de glucosa el punto de interés de este modelo, será recibida por un fotodiodo infrarrojo, el cual se excitará y permitirá la circulación de energía. Esta señal saliente del fotodiodo es muy pequeña, por lo que debe pasar por una etapa de filtrado y amplificación. Para esto se hace uso de amplificadores operacionales en configuración de circuitos divisorios de tensión para la alimentación de los operacionales desde el microcontrolador y configuraciones de circuitos RC para la construcción de un filtro pasabandas para eliminar componentes de señal indeseadas.

Puesto a que se trata de una señal analógica, su salida debe ser transmitida a un pin ADC del microcontrolador configurado como entrada, al cual llega en forma de tensión.



Esquemas de conexión en Kicad de la etapa de emisor infrarrojo (izquierda), y receptor (derecha).

Siguiendo lo implementado en un antiguo proyecto para la construcción de un prototipo de medición no invasiva de glucosa en sangre, se procedió a la incorporación de un LED de luz visible con el objetivo de acondicionar la señal infrarroja frente al ruido ambiente. Para su testeo, se diseñó un protocolo de encendido y apagado intermitente de los LEDs y la medición de la señal amplificada del fotodiodo infrarrojo en cada una de estas instancias. El protocolo consiste en un encendido inicial del LED IR por 6 segundos, luego ambos se encuentran 500 ms apagados. Finalmente, el LED visible se enciende durante 6 segundos para terminar con 500 ms de ambos apagados.

```
#define CON2_1 4 // El CON2-1 corresponde al D4
#define CON2_3 2 // El CON2-3 corresponde al D2
#define CON2_13 A3 // El CON2-13 corresponde al A3
#define CON2_14 36 // Salida filtrada y amplificada
#define CON2_15 A1 // Libre (por si es necesario)
#define CON2_16 39 // Salida del sensor
// Para hallar a qué pinMode corresponde un pin físico:
// Miramos el pin físico que queremos utilizar, por ejemplo, el pin CON-2
3.
// Lo buscamos en el esquema del micro del datasheet y miramos las
funciones que
// puede realizar, en este caso: OC5/IC5/PMWR/CN13/RD4.
// Buscamos eso en el esquema de la hoja de correspondencia del
ESP32
// y vemos que se corresponde con D2. Siguiendo la línea, nos lleva a
D2.
// Como es un pin digital, no es necesario utilizar la D, por lo que el
pinMode será 2
#define TM 10 // Define el tiempo de muestreo
#define LEDR CON2_3 // La señal del Led rojo vendrá por el CON-2 3
#define LEDIR CON2_1 // La señal del Led IR vendrá por el CON-2 1
int Te=1000*5; // Tiempo de encendido en ms
int cuentaTe=0;
int Ts=500; // Tiempo de separación en ms
int cuentaTs=0;
int valorIR;
int valoraux;
int valorLEDR;
int valor0;
int Ti=0;
int Tf=0;
int Tx=0;
int parlIR=0;
int parsep1=0;
```

```

int parLEDR=0;
int parsep2=0;
int aux;
int aux2;
unsigned char b0;
unsigned char b00;
unsigned char b1;
unsigned char b10;
unsigned char b2;
unsigned char b20;
unsigned char b3;
unsigned char b30;
void setup() {
pinMode(LED_R, OUTPUT); //salida para el led rojo
pinMode(LED_I, OUTPUT); //slida para el led IR
pinMode(CON2_16, INPUT); //ADC A0
pinMode(CON2_14, INPUT); //ADC A2

Serial.begin(9600); //para comunicar la salida
}
void loop() {
parIR=0;
parsep1=0;
parLEDR=0;
parsep2=0;
////////////////////// Enciende y apaga el LED IR /////////////////////
digitalWrite(LED_I, HIGH);
Ti=millis();
cuentaTe = millis();
//lee el tiempo en ms desde el instante en que se ha empezado a ejecutar
if (parIR <1){
Serial.print("ok");
}
while (parIR<1){

if (cuentaTe>Te){digitalWrite(LED_I, LOW);
//Anexo A 102
cuentaTe=0;
parIR=2;
}
else {
Tf=millis();
Tx=Tf-Ti;
if (Tx>TM){ aux=analogRead(CON2_16);
b00=aux / 256;
b10=aux % 256;
}
}
}

```

```
b0=(b00 << 3) | (b10 >> 5);
Serial.write(0x80 | b0);

b1=b10 & 0x1f;
Serial.write(0x0 | b1);

aux2=analogRead(CON2_14);
b20=aux2 / 256;
b30=aux2 % 256;

b2=(b20 << 3) | (b30 >> 5);
Serial.write(0xA0 | b2);

b3=b30 & 0x1f;
Serial.write(0x0 | b3);

Ti=Tf;
cuentaTe=cuentaTe+TM;
}
}

////////////////// Tiempo de separación //////////////////

//Ti=millis();
while (parsep1<1){

if (cuentaTs>Ts){cuentaTs=0;
parsep1=2;
}
else{
Tf=millis();
Tx=Tf-Ti;
if (Tx>TM){ Ti=Tf;

aux=analogRead(CON2_16);
b00=aux / 256;
b10=aux % 256;

b0=(b00 << 3) | (b10 >> 5);
Serial.write(0x80 | b0);

b1=b10 & 0x1f;
Serial.write(0x20 | b1);

aux2=analogRead(CON2_14);
b20=aux2 / 256;
b30=aux2 % 256;
```

```

b2=(b20 << 3) | (b30 >> 5);
Serial.write(0xA0 | b2);

b3=b30 & 0x1f;
Serial.write(0x20 | b3);

cuentaTs=cuentaTs+TM;
}
}

////////// Enciende y apaga el LED ROJO //////////
digitalWrite(LED_R, HIGH);
Ti=millis();
while (parLED_R<1){

if (cuentaTe>Te){digitalWrite(LED_R, LOW);
//Anexo A 104
cuentaTe=0;
parLED_R=2;
}
else{
Tf=millis();
Tx=Tf-Ti;
if (Tx>TM){ aux=analogRead(CON2_16);
b00=aux / 256;
b10=aux % 256;

b0=(b00 << 3) | (b10 >> 5);
Serial.write(0x80 | b0);

b1=b10 & 0x1f;
Serial.write(0x40 | b1);

aux2=analogRead(CON2_14);
b20=aux2 / 256;
b30=aux2 % 256;

b2=(b20 << 3) | (b30 >> 5);
Serial.write(0xA0 | b2);

b3=b30 & 0x1f;
Serial.write(0x40 | b3);

Ti=Tf;
cuentaTe=cuentaTe+TM;
}
}
}

////////// Tiempo de separación //////////
Ti=millis();
while (parsep2<1){

if (cuentaTs>Ts){cuentaTs=0;
parsep2=2;
}
}

```

```

else{
Tf=millis();
Tx=Tf-Ti;
if (Tx>TM){ Ti=Tf;

aux=analogRead(CON2_
16);
b00=aux / 256;
b10=aux % 256;

b0=(b00 << 3) | (b10 >>
5);
Serial.write(0x80 | b0);

b1=b10 & 0x1f;
Serial.write(0x60 | b1);

aux2=analogRead(CON2
_14);
b20=aux2 / 256;
b30=aux2 % 256;

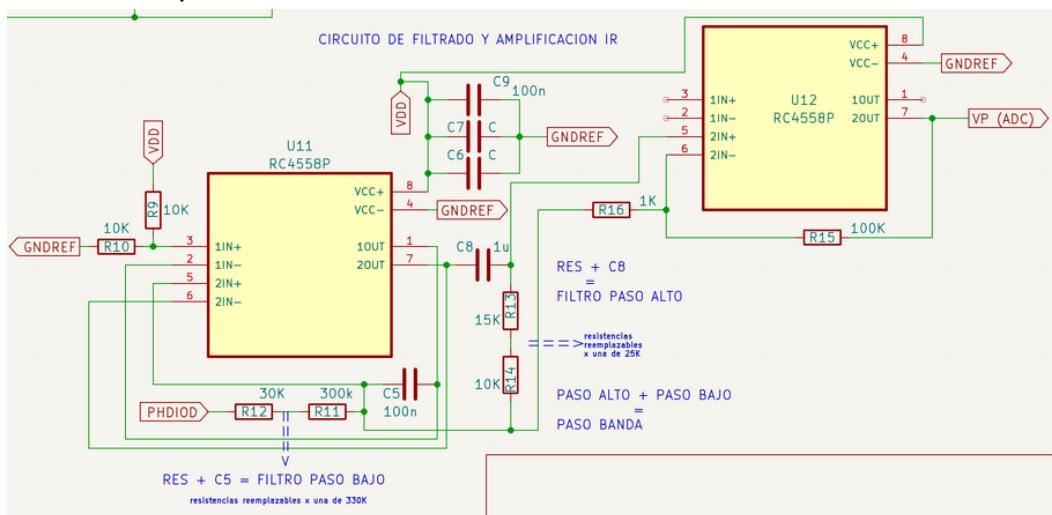
b2=(b20 << 3) | (b30 >>
5);
Serial.write(0xA0 | b2);

b3=b30 & 0x1f;
Serial.write(0x60 | b3);

cuentaTs=cuentaTs+TM;
}
}
}
}

```

Para mas informacion sobre las implementaciones del codigo para el sensor de glucosa consulte nuestro repositorio de GitHub.



Esquema de circuito de filtrado y amplificación realizado en Kicad

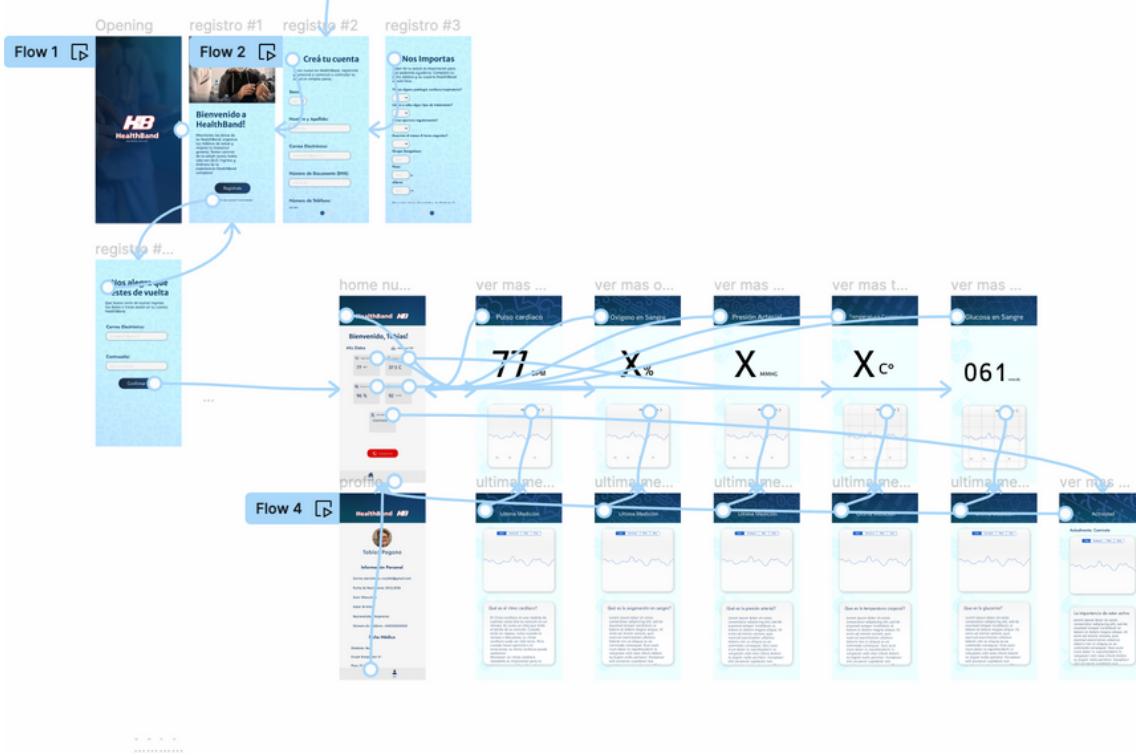
# INTERFACES GRÁFICAS

## Aplicación Móvil

La aplicación móvil es el medio por el cual los usuarios de HealthBand podrán visualizar sus mediciones de manera continua, procurando la comodidad y la simpleza de uso. Su creación implicó un desarrollo en aspectos como el diseño de interfaces y transporte de la identidad del proyecto a las mismas, así como almacenamiento y transmisión de datos. Estos dos aspectos suelen dividirse en dos grandes familias: Frontend hace referencia a la parte creativa de todo aquello que el usuario puede ver en la pantalla, mientras que el Backend engloba todas aquellas características que hacen a una aplicación funcional.

## Frontend

La construcción de una aplicación atractiva comenzó desde el diseño de interfaces de usuario. Para ello se implementó Figma, una potente plataforma web de libre acceso orientada al diseño de interfaces digitales, con la cual se crearon y conectaron entre sí las diversas pantallas que forman parte de la aplicación. Se han agregado pantallas de inicio, elementos de tipo input para texto y demás componentes activos como botones y barras de navegación. El prototipo final con su diagrama de conexiones queda visualizado de la siguiente forma:



Al abrir por primera vez la aplicación, podrá ver una pantalla de carga con nuestro logotipo sobre un fondo azul oscuro. Tras unos segundos se aparecerá una pantalla de bienvenida con una imagen en la parte superior, en la que tras un cuerpo de texto de introducción, tendrá 2 opciones para seguir avanzando: Una en forma de botón azul oscuro con un texto interno “Regístrate” le permitirá crear un usuario por primera vez. Debajo de este botón, otra opción en forma de botón de texto le permitirá navegar hasta otra pantalla e iniciar sesión con un usuario anterior.



Si escogió iniciar sesión con un usuario previo, se le solicitará únicamente su correo electrónico y contraseña. Una vez introducidos correctamente y presionado el botón de confirmación, estará de vuelta en la aplicación móvil de HealthBand. Recuerde que siempre tiene la opción de volver su decisión con las flechas al costado superior izquierdo de la pantalla.



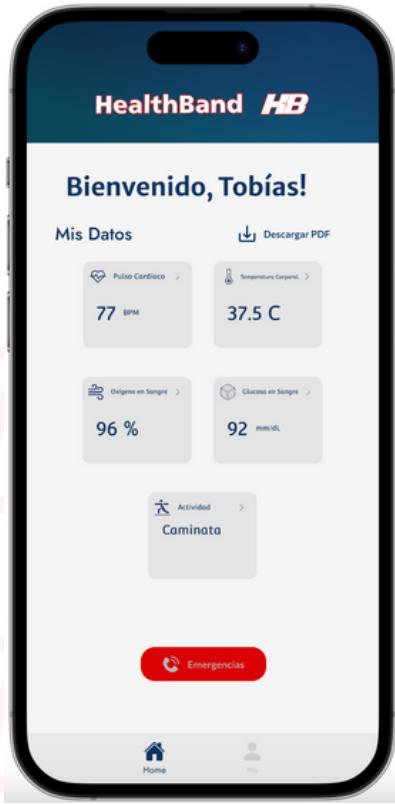
Pantalla de Inicio de Sesión

Si en cambio decidió crear una cuenta nueva, se dirigirá a una instancia de registro de 2 etapas: una primera instancia donde deberá completar datos personales, y una segunda para completar con datos médicos. De esta manera, su perfil de usuario HealthBand quedará configurado.

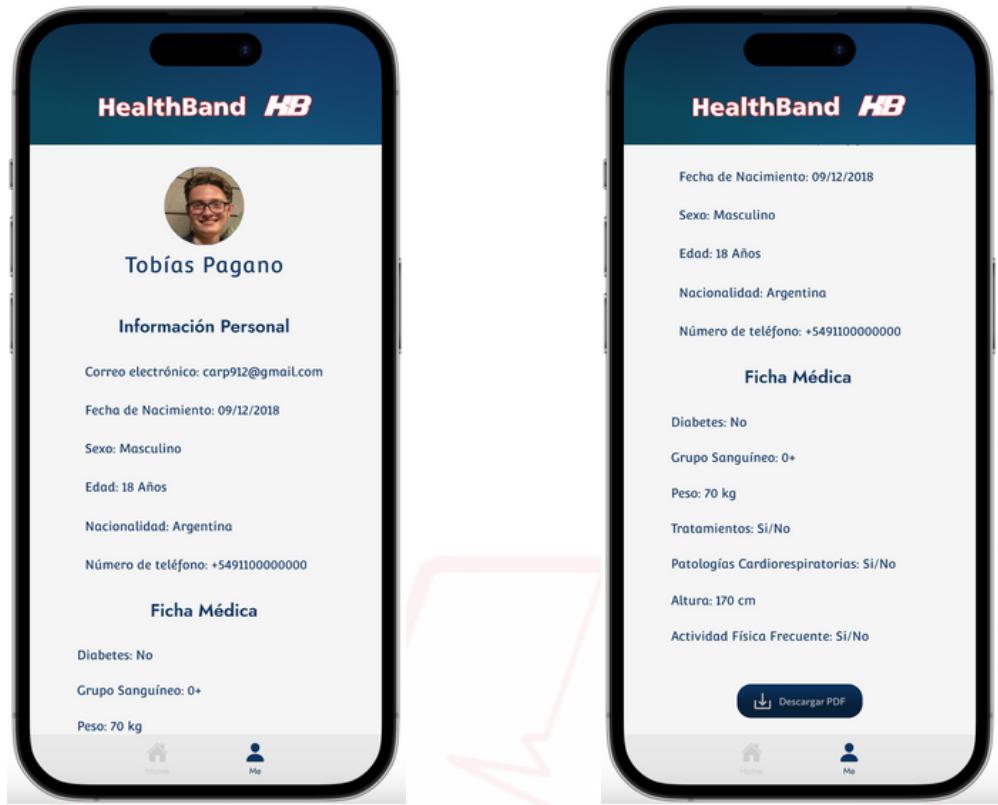


Para las pantallas de registro se optó por utilizar colores dinámicos y amigables, lo cual se puede ver con el fondo celeste con vivos aturquesados y azulados. La pantalla también se compone de elementos de texto (títulos, subtítulos y cuerpo) en un tono azul oscuro, característico de HealthBand. Los espacios blancos, conocidos como inputs, permiten al usuario introducir información personal. Un botón de confirmación con tono azul oscuro es el que permite avanzar de etapas en el registro. También se cuenta con iconos de tipo chevron cumpliendo la función de botón de regreso a la pantalla anterior, en caso de que el usuario se haya arrepentido de dirigirse a esa pantalla.

Tras darle al botón de confirmación en la segunda ventana de registro, la aplicación lo dirigirá a la siguiente pantalla de menú:

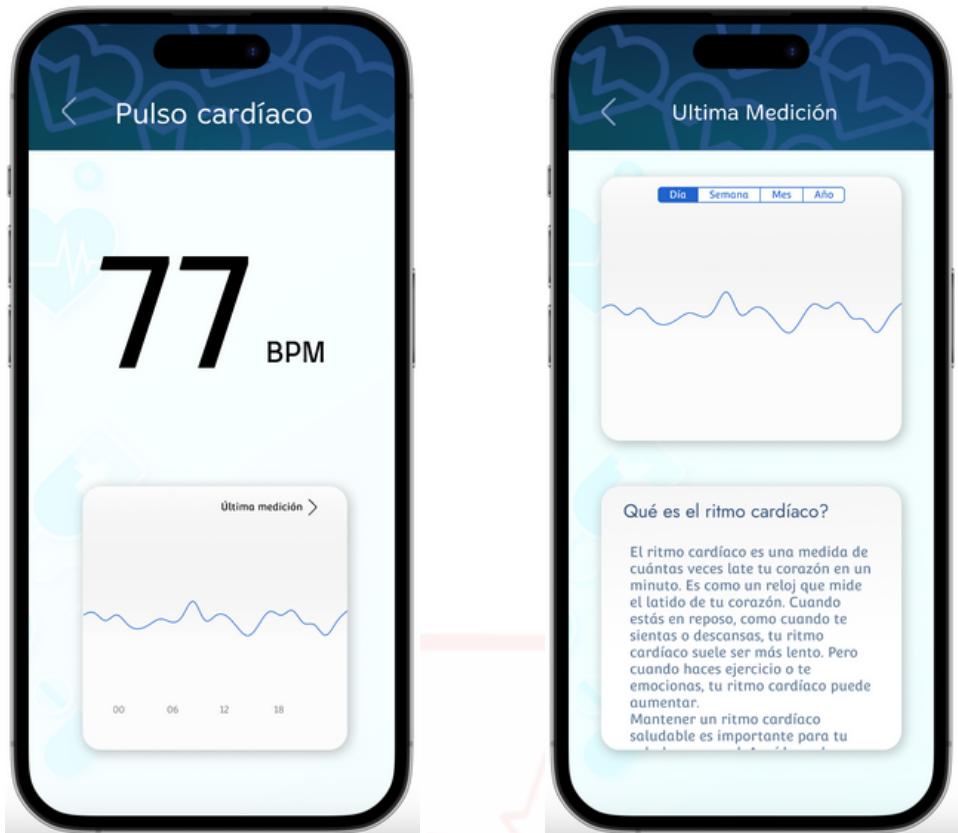


Tras ingresar a la aplicación de HealthBand, se encontrará una interfaz de usuario simple y de fácil comprensión: Una barra superior azulada con el logotipo de HealthBand le otorga la identidad de marca al menú principal. Por debajo podemos observar un titular de bienvenida con nuestro nombre registrado. En el centro de la pantalla podemos encontrar diversos bloques desde los cuales la aplicación muestra los datos medidos por el dispositivo portátil a los usuarios. Estos bloques se encuentran encima de un botón de emergencias, el cual al pulsarlo llamará al número del Centro de Emergencias Nacional (911). Sobre los mismo también hay otro botón que permite confeccionar un archivo en formato pdf con las mediciones actuales al pulsarle. Finalmente podrá ver una barra inferior donde encontrará dos posiciones: el menú, representado por el icono de una casa y la posición inicial al momento de abrir la aplicación; y otro representado con un ícono de persona, el cual al tocar sobre el dirige la aplicación al perfil de usuario, donde encontrara información personal y su ficha médica previamente cargadas en la etapa de registro. Esta información también será descargable en formato pdf mediante la pulsación de un botón, que en este caso se encuentra debajo de la ficha médica, debiendo hacer un movimiento de scroll vertical hacia abajo para llegar a el.



Vista completa de la pantalla de perfil

Volviendo a la pantalla de menú, los bloques de datos de esta contienen información adicional a la que se puede acceder pulsando sobre ellos. Se abrirá de forma directa una nueva ventana donde podrá verse los valores de medición junto a un gráfico de historial diario. Este posee una pestaña de invitación a ver más información, sobre la cual al picar dirigirá al usuario a otra pantalla, con el gráfico ocupando un mayor espacio de pantalla y un bloque descriptivo sobre el signo vital de la sección. Al igual que en las etapas de registro, las barras de navegación superiores cuentan con botones de regreso en caso de que el usuario desee volver a la sección anterior.



Pantalla de signo vital con gráfico inferior. El botón de regreso dirige al menú. Pulsar en el gráfico dirige a la pantalla informativa (derecha).

Pantalla informativa con gráfico detallado y bloque descriptivo. El botón de regreso dirige a la pantalla de la izquierda.

Como se puede apreciar, el diseño de la aplicación determinó la implementación de una cantidad reducida de pantallas para la interfaz de usuario, priorizando la simplicidad y facilidad de lectura. Para su conversión a código se decidió implementar el framework de código abierto Flutter, desarrollado por Google para la construcción de aplicaciones web, móviles y de escritorio, basado en el lenguaje Dart. Dichos códigos se podrán encontrar en el repositorio de GitHub del proyecto.

## Backend

El backend es el sector de la programación que atribuye funcionalidades a un proyecto y, de alguna forma, es lo que vuelve útil a una aplicación más allá del diseño. En el caso de HealthBand, el backend se encargaría del registro de usuario y de la transmisión de datos desde la ESP32. Para ello, se utilizó Django, un framework de Python ampliamente utilizado en el desarrollo de apps y webs para la gestión de datos y comunicación de dispositivos. El mismo se ha utilizado también para la transmisión de datos hacia la web.

Django trabaja con una arquitectura de 3 componentes esenciales: Un modelo, el cual tiene acceso a una base de datos, una vista que se conecta con este primero y es quien recibe las peticiones que luego muestra en una plantilla (aquellos que el usuario puede visualizar desde su dispositivo).

Para la transmisión de datos, en principio se debe crear un modelo. Los campos de las tablas de los modelos son configurables en términos de tipos de datos, pudiendo recibir enteros, cadenas de caracteres, flotantes, etc.

Las vistas se definen como funciones que reciben como argumento una petición. Los datos serán posteados en una determinada dirección URI en formato JSON, y las vistas evaluarán que la petición sea efectivamente un post para así obtener del JSON los datos de las mediciones e instanciarlos en objetos. Se ha elegido como práctica un posteo indirecto, realizándose en una ruta oculta, y luego conectando las rutas donde se quiere visualizar las mediciones (pagina web y aplicación) con sus respectivos objetos. Para más información sobre el manejo de datos haciendo uso de Django, consulte nuestro repositorio de GitHub.

## Página Web

La página web es otra de las plataformas virtuales con las que cuenta HealthBand. Al ingresar desde la pantalla completa de un ordenador, su pantalla principal se ve de la siguiente forma:

## Frontend

Para su construcción se utilizó Webflow, una plataforma para crear y hostear proyectos frontend, y se adquirieron conocimientos básicos de maquetado en HTML5, CSS y JavaScript. Su composición es simple, adoptando estilos modernos y dinámicos. En la parte superior nos encontramos con una barra de navegación, la cual presenta el logotipo y una serie de opciones de menú para dirigirse a otras secciones de la página. Sobre la derecha, un botón redondeado en un celeste tenue dirige al correo electrónico al pulsarlo, en caso de querer contactarse con nosotros.

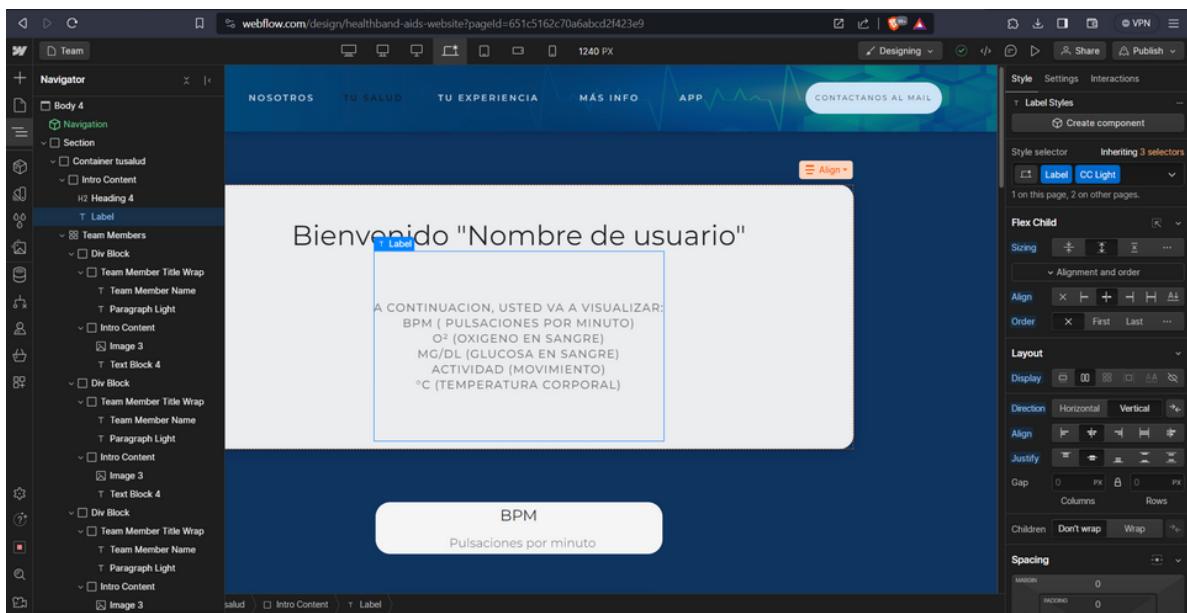
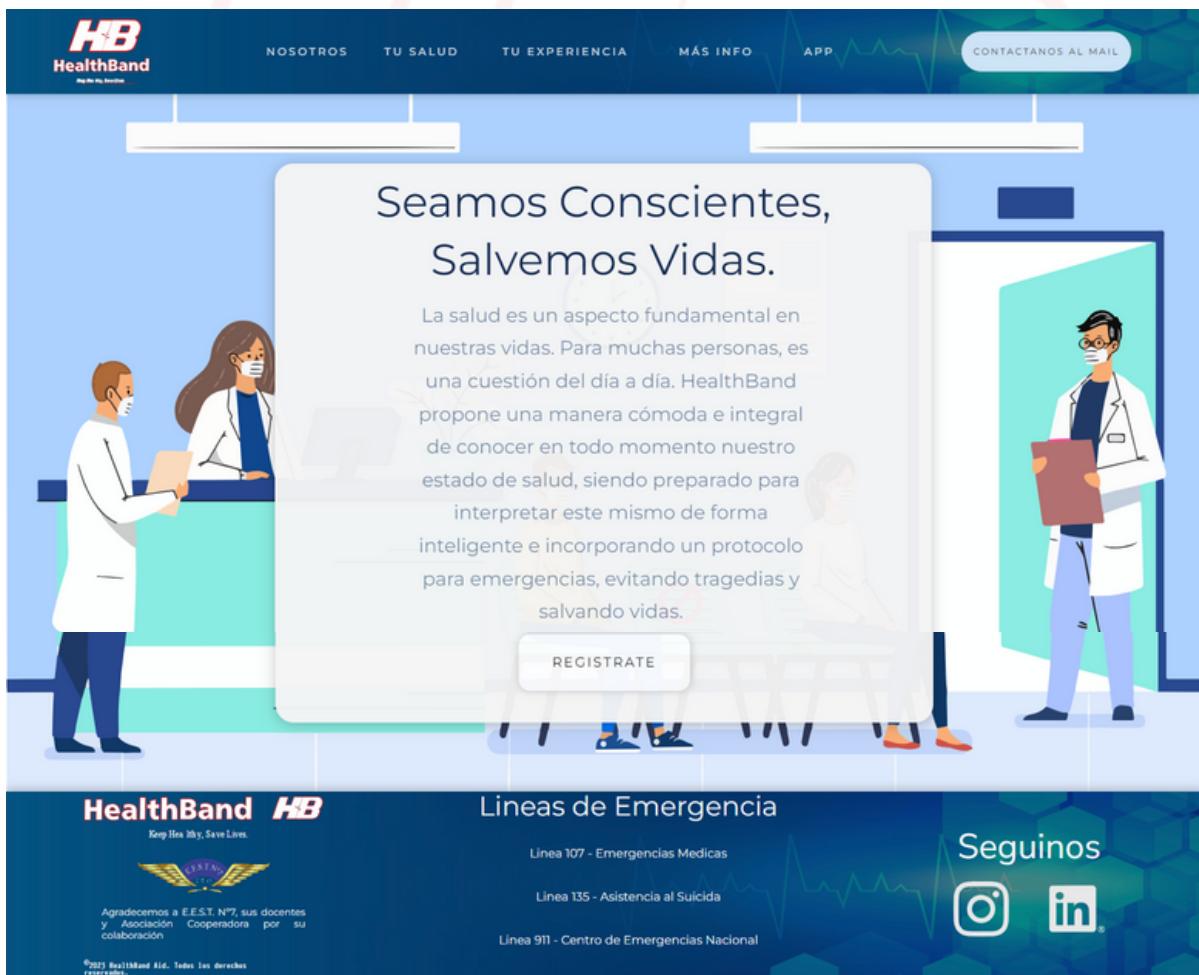


Imagen de webflow en la pestaña de "Tu Salud"



Pestaña Principal "Home"

El cuerpo presenta una ilustración decorativa hacia el fondo, con un bloque blanco de bordes redondeados, el cual contiene un título y un cuerpo de texto invitando a registrarse en la misma con un botón en la parte inferior.

Hacia abajo, el pie de pagina muestra diversos tipos de información, divididos en 3 columnas: Hacia la izquierda, presta agradecimientos a la escuela y muestra su escudo. También incluye otra versión del logotipo del proyecto. En el medio, el título “Líneas de Emergencia” despliega una serie de links para llamados a diversas líneas telefónicas. Por ultimo, en la columna de la derecha, el título “Segúinos” invita a ingresar a los perfiles de Instagram y LinkedIn del proyecto a través de sus respectivos iconos. Siendo este pie de pagina, repetido en las demás pestañas.

La pestaña de “Nosotros” muestra informacion personal del grupo y cada uno de sus miembros, así como sus perfiles de redes sociales tanto personales como profesionales. En “Tu Salud” se podrán ver los datos medidos por el wearable divididos en diferentes bloques. “Tu Experiencia” lleva a una caja de comentarios donde los usuarios podrán dejar mensajes al correo de HealthBand contando su experiencia con el servicio, “Mas info”, proporcionaría información acerca de la pulsera y como funciona y “App” es una sección para explicar y proporcionar la descarga de nuestra aplicación.



Pestaña “Nosotros” primera parte

Somos un grupo de estudiantes de Séptimo año de la escuela Técnica N°7- IMPA. Buscamos diseñar un sistema capaz de monitorear los signos vitales de una persona, Siendo capaz de activar un protocolo de emergencias cuando se presente una complicación de salud.

## Integrantes



Acuña, Alvaro

Tengo 18 años, me gusta todo lo relacionado al diseño. En un futuro pienso estudiar diseño gráfico



Julianetti, Bruno

Tengo 19 años, me gusta la biotecnología y el diseño enfocado a la electrónica, me gustaría aprender todo lo que esté a mi alcance en estos campos.



Gomez, Gonzalo

Tengo 19 años, me gusta la programación y la gestión, me gustaría combinar estas dos aficiones, estudiándolas y aplicándolas en un futuro.



Pagano, Tobias

Tengo 18 años, me considero una persona responsable y dedicada a expandir mi conocimiento y creatividad. Estoy interesado en seguir estudiando diseño gráfico y multimedia.



Perlo, Mateo

Tengo 19 años, soy una persona apasionada por el aprendizaje y la exploración, en constante búsqueda de nuevas experiencias y conocimientos.

## Contáctanos

Para cualquier consulta o comentario que nos quieras hacer.

CLICK ACA



Pestaña "Nosotros" segunda parte, final.

**Bienvenido "Nombre de usuario"**

A CONTINUACION, USTED VA A VISUALIZAR:

- BPM ( PULSACIONES POR MINUTO)
- O<sup>2</sup> (OXIGENO EN SANGRE)
- MG/DL (GLUCOSA EN SANGRE)
- ACTIVIDAD (MOVIMIENTO)
- °C (TEMPERATURA CORPORAL)

**BPM**  
Pulsaciones por minuto

aca irian los datos medidos.

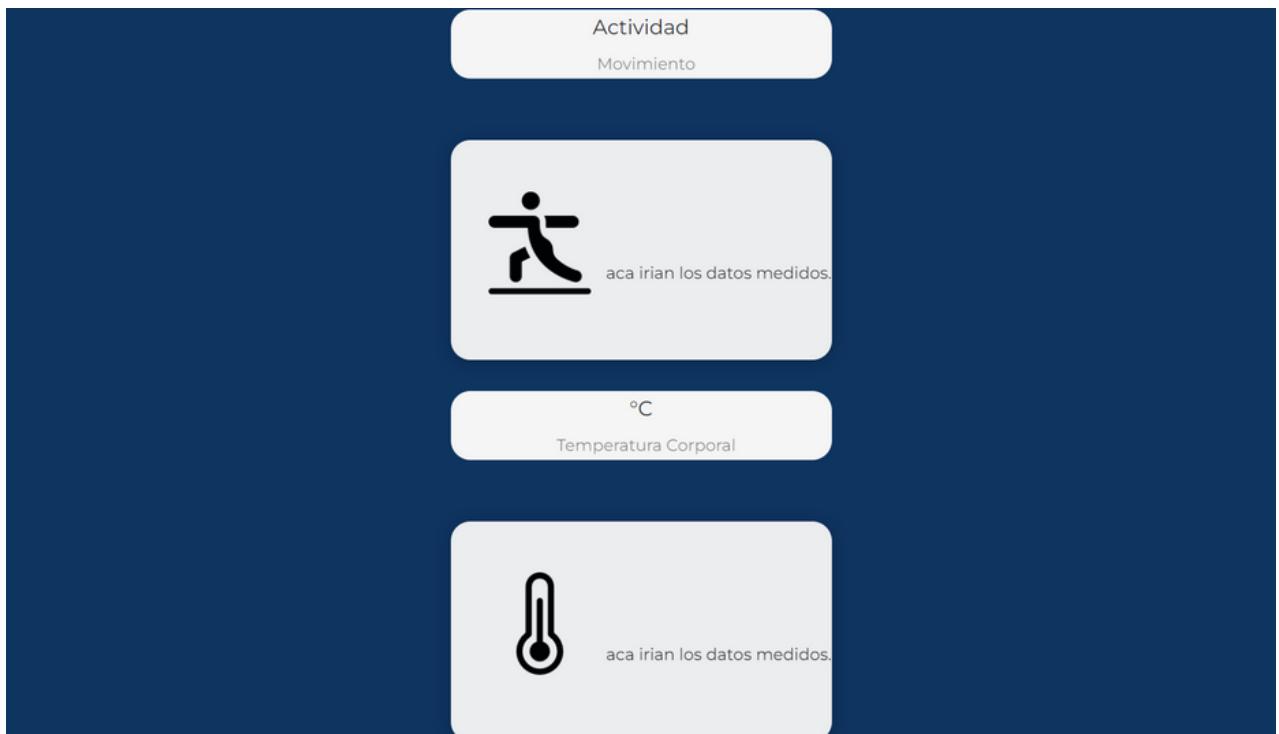
**O<sup>2</sup>**  
Oxigeno en sangre

aca irian los datos medidos.

**MG/DL**  
Glucosa en sangre

aca irian los datos medidos.

Pestaña "Tu Salud" Primera parte.



Pestaña "Tu Salud" Segunda parte y final.

**HB**  
HealthBand

NOSOTROS TU SALUD TU EXPERIENCIA MÁS INFO APP CONTACTANOS AL MAIL

## Mandanos un mensaje

Si tiene alguna consulta, algo para incorporar, dudas, etc. Mándanos un mensaje así podemos contactarnos y que usted nos ayude a mejorar.

**NOMBRE**

Ingresá tu nombre

**CORREO ELECTRÓNICO**

Ingresá tu Email

**MENSAJE**

Buenas tardes, quería consultártelos sobre algo....

**ENVIAR**

DONDE ESTAMOS  
E.E.S.T Taller regional Quilmes "T.R.Q"  
Escuela Técnica N°7 "IMPA"

CONTACT  
[healthband.aid@gmail.com](mailto:healthband.aid@gmail.com)

Pestaña "Tu experiencia"

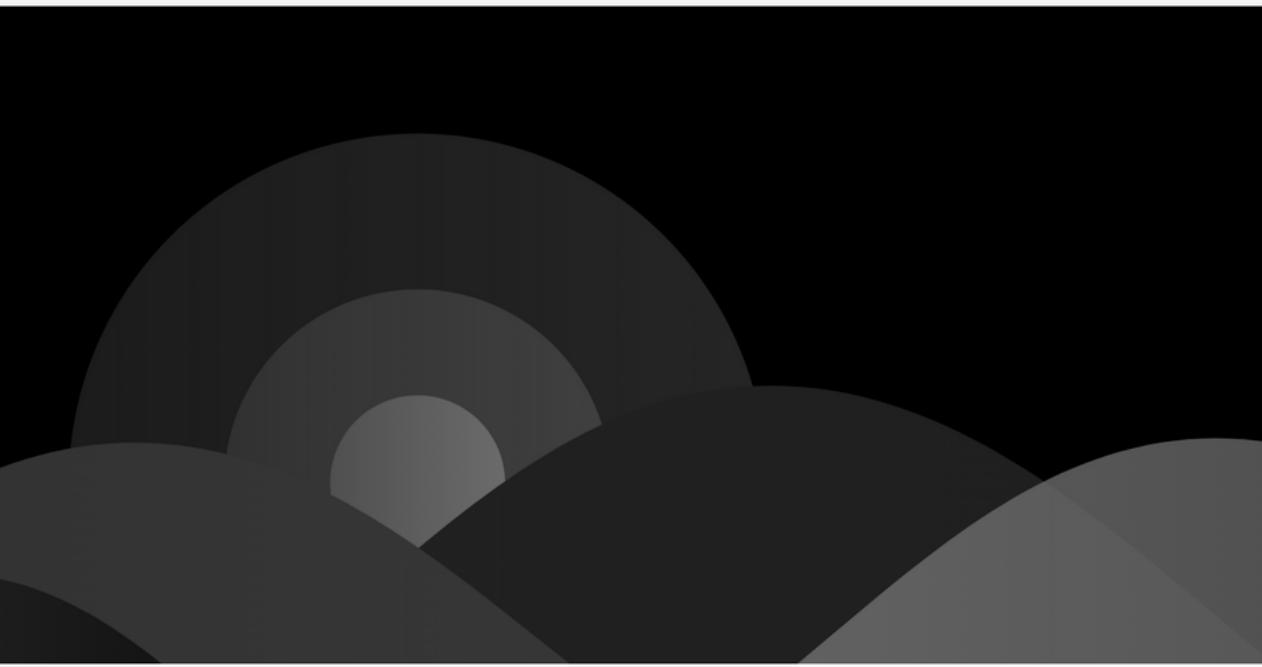


NOSOTROS TU SALUD TU EXPERIENCIA MÁS INFO APP CONTACTANOS AL MAIL

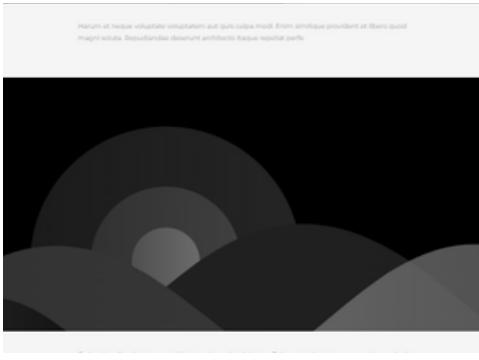
## Nuestro proyecto

COMO USAR

### HealthBand



Pestaña "Mas info" la cual contiene mas recuadros negros para llenar con nuestra información, desde como usar HealthBand hasta lo aplicado en la ONIET, y mostrar nuestros avances.



Previsualización  
en el scrolleo hacia  
abajo de la pestaña  
"Mas info"





NOSOTROS TU SALUD TU EXPERIENCIA MÁS INFO APP CONTACTANOS AL MAIL

## Team Projects

HEALTHBAND APP

Donde descargar y como funciona

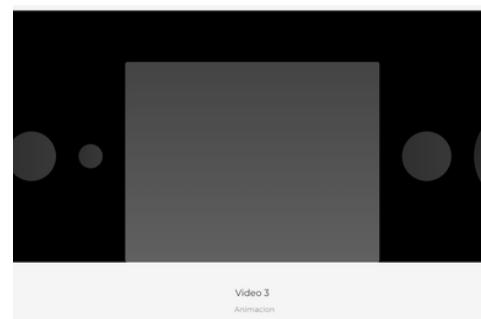
Video 1

Pestaña "App", la cual contendría videos explicativos sobre como utilizar la aplicación, como funciona y como descargarla.



Video 2

Previsualización  
en el scrolleo hacia  
abajo de la pestaña  
"App"



Video 3  
Animacion

## Backend

En el caso de HealthBand, el backend se encargaría del registro de usuario y de la transmisión de datos desde la ESP32. Para ello, se utilizó Django, un framework de Python ampliamente utilizado en el desarrollo de apps y webs para la gestión de datos y comunicación de dispositivos. El mismo se ha utilizado también para la transmisión de datos hacia la web.

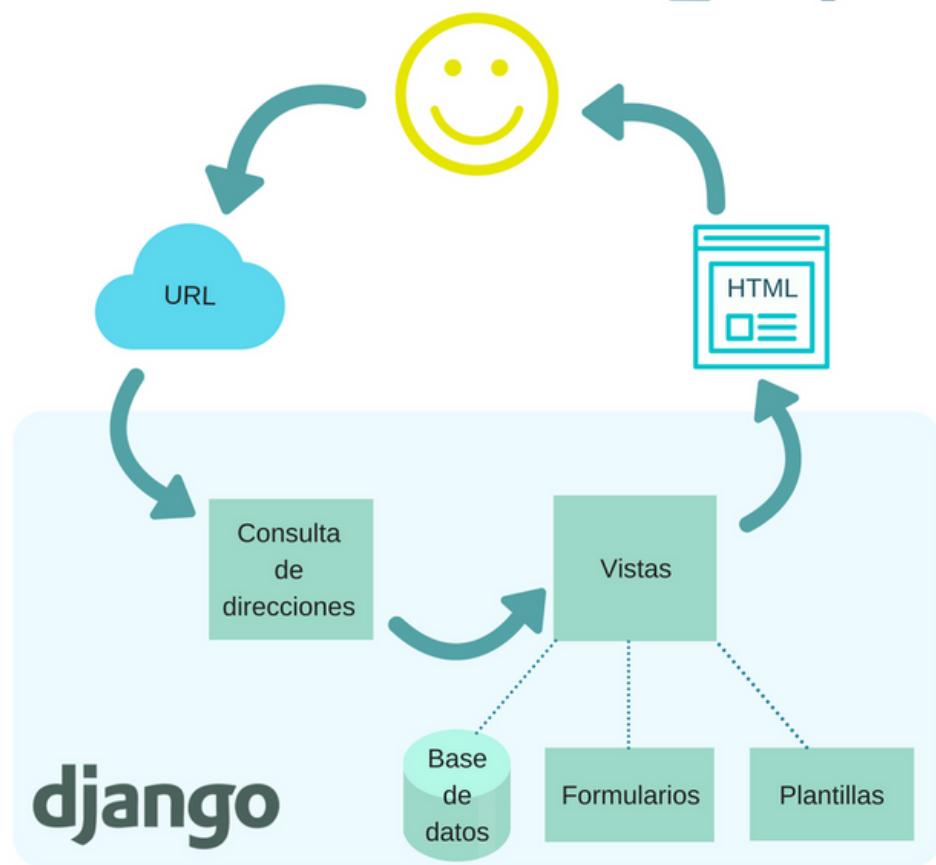
Django trabaja con una arquitectura de 3 componentes esenciales: Un modelo, el cual tiene acceso a una base de datos, una vista que se conecta con este primero y es quien recibe las peticiones que luego muestra en una plantilla (aquellos que el usuario puede visualizar desde su dispositivo).

Para la transmisión de datos, en principio se debe crear un modelo. Los campos de las tablas de los modelos son configurables en términos de tipos de datos, pudiendo recibir enteros, cadenas de caracteres, flotantes, etc.

Las vistas se definen como funciones que reciben como argumento una petición. Los datos serán posteados en una determinada dirección URL en formato JSON, y las vistas evaluarán que la petición sea efectivamente un post para así obtener del JSON los datos de las mediciones e instanciarlos en objetos. Se ha elegido como práctica un posteo indirecto, realizándose en una ruta oculta, y luego conectando las rutas donde se quiere visualizar las mediciones (pagina web y aplicación) con sus respectivos objetos.

Para mas informacion sobre el manejo de datos haciendo uso de Django, consulte nuestro repositorio de GitHub.

Debido a la cantidad de lineas de código que poseen el Style.css, Script.js y los HTML, se podrán visualizar mejor en el repositorio en GitHub del proyecto, a continuación, se procederá a explicar de manera detallada y concisa el backend detrás de la página, lo que sería la utilización de Micropython y Django para el envío, recepción, lectura, y muestreo de datos en la web. A su vez, se detallarán ciertas funciones únicas de Django para poder entender en mayor medida y por encima el funcionamiento del mismo.



Como se da entender en el diagrama, Django funciona de una manera la cual el usuario se conecta hacia una URL, siendo esta la tipica de una pagina <https://healthband.com>, luego de alli hace una consulta de direcciones hacia lo que seria en Django la carpeta views.py

```

    <ul style="list-style-type: none; padding-left: 0;">
- > Página web
- > healthweb
- > healthdatos
      <ul style="list-style-type: none; padding-left: 20px;">
- > __pycache__
- > migrations
- __init__.py
- admin.py
- apps.py
- models.py
- tests.py
- urls.py
- views.py

```

Siendo esta sección, la receptora de la base de datos, los formularios (ya sean para crear una función de registro y login en la página, y, la recepción de las plantillas, siendo estas los style.css y luego enviando todo esto hacia nuestro HTML de las diferentes urls).

Habiendo visto por encima el funcionamiento lógico de Django a continuación se explicará brevemente lo que se hizo en cada uno de los bloques para lo que sería el funcionamiento de nuestro Backend.

## Concepto proyecto/app

Para tener un proyecto en Django hay que tener claros 2 conceptos el de Proyecto y el de App.

Proyecto es el directorio raíz de una aplicación, contiene la configuración global como la configuración de la base de datos, las rutas URL principales, la configuración de autenticación, etc. Dentro de un proyecto de Django pueden haber múltiples aplicaciones y el proyecto coordina cómo se conectan y funcionan juntas.

Aplicación como tal, es un componente reutilizable y autónomo que realiza una función específica en nuestro proyecto, en este caso, para la recepción, lectura y muestra de datos.

Habiendo aclarado esto, a continuación se utilizarán estos conceptos para la explicación de URLs.

```
└── healthweb
    ├── __pycache__
    ├── __init__.py
    ├── asgi.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

Estos serían los archivos raíz de nuestro proyecto

```
└── healthdatas
    ├── __pycache__
    ├── migrations
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── tests.py
    ├── urls.py
    └── views.py
```

Estos serían los archivos App de nuestro proyecto

## URL

Para crear las diferentes URL de nuestra pagina web, se utiliza el archivo de views.py de nuestra aplicacion “urls.py”, no sin antes aclarar en el archivo url de nuestro proyecto, que incluya las url de nuestra app.

urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include("healthdatas.urls"))
    #aca se incluyen todas las url que se escriban en nuestra app (healthdatas)
]
```

A continuacion urls.py de nuestra App

```
from django.urls import path
#Importa la función path desde el módulo django.urls. Esta función se utiliza para definir las rutas URL
# y mapearlas a las vistas correspondientes.
from . import views
```

```
# Importa las vistas definidas en el mismo directorio (el directorio donde se encuentra este archivo).
# El punto (.) se utiliza para indicar el directorio actual.
```

```
urlpatterns = [
#Define una lista llamada urlpatterns que contiene una serie de patrones URL y las vistas asociadas a cada uno.
```

```
#Cada elemento de la lista es una llamada a la función path que toma tres argumentos
```

```
#El primer argumento es la ruta URL que se está configurando.
```

```
#El segundo argumento es la vista que se debe ejecutar cuando la URL coincide con este patrón.
```

```
#El tercer argumento es un nombre para esta ruta URL
```

```
#que se utiliza para referenciarla de manera única en otras partes del código, como en plantillas.
```

```
path("", views.home, name="home"),
```

```
path("nosotros/", views.about, name="nosotros"),
```

```
#ejemplo, la ruta seria healthband.com/nosotros/, la vista que cargaría sería del archivo views.py la función about
```

```
#y se le asigna el nombre "nosotros" para llamar este path en un futuro
```

```
path("team/", views.team, name="team"),
path("contact/", views.contact, name="contact"),
path("projects/", views.projects, name="projects"),
path("blog/", views.blog, name="blog"),
path("datos/", views.mediciones, name="mediciones"),
]
```

## Vistas

Las vistas únicamente se manejan desde la app, no desde el proyecto raíz. A continuación se mostrará el código hecho y comentado en el archivo "views.py"

```
from django.shortcuts import render
# Esta importación permite que tu código utilice la función render de Django.
# La función render se utiliza para renderizar plantillas HTML y devolver la respuesta
# HTML al navegador del usuario.
# En las vistas, usas esta función para renderizar las plantillas HTML correspondientes a
# cada página o función.
from .models import datos
# Importa el modelo datos desde el archivo models.py en el directorio actual.
# Los modelos en Django definen la estructura de las tablas de la base de datos y cómo
# se deben almacenar y recuperar los datos.
# Esta importación te permite trabajar con el modelo datos en tus vistas para almacenar
# y recuperar datos en la base de datos.
from django.http import JsonResponse
# Importa la clase JsonResponse de Django. JsonResponse se utiliza para devolver
# respuestas en formato JSON desde tus vistas.
# En la vista mediciones, utilizamos JsonResponse para responder con mensajes JSON
# que indican si los datos se han almacenado correctamente
# o si ha habido errores en la solicitud.
from django.views.decorators.csrf import csrf_exempt
# Al aplicar csrf_exempt a una vista, se permite que las solicitudes POST a esa vista se
# realicen sin necesidad de un token CSRF
# Lo cual es útil en casos donde no se necesita esta protección, como en tu vista
# mediciones donde se reciben datos en formato JSON.
import json
# El módulo json proporciona funciones para trabajar con datos en formato JSON
# (JavaScript Object Notation)
# que es un formato de intercambio de datos comúnmente utilizado en aplicaciones web
# y API REST para representar información estructurada.
```

```
def home(request):
#Esta línea define una función llamada home que toma un argumento llamado request.
# En Django, el objeto request contiene información sobre la solicitud HTTP que el
usuario realiza, como la URL solicitada o los datos POST
    return render(request, "home.html")
# En esta línea, la función home utiliza la función render para generar una respuesta
HTTP. La función render toma dos argumentos:
#request:Es el objeto de solicitud que se pasa como argumento a la vista.
#"home.html":Es el nombre de la plantilla HTML que se debe renderizar y devolver como
respuesta.
def about(request):
    return render(request, "about.html")

def team(request):
    return render(request, "team.html")

def contact(request):
    return render(request, "contact.html")

def projects(request):
    return render(request, "projects.html")

def blog(request):
    return render(request, "blog.html")
@csrf_exempt
#llamamos a lo importado @csrf_exempt
def mediciones(request):
    if request.method == 'POST': #le pide que reciba metodos POST solamente
        try: #Bucle que intenta infinitamente
            data = json.loads(request.body) # Analiza el JSON recibido
            pulsaciones = data.get("PULSOS") #Agarra del JSON la data que tenga de nombre
            pulsaciones
            oxigeno = data.get("OXIGENO") #Agarra del JSON la data que tenga de nombre
            Spo2
            datos.objects.create(pulsaciones=pulsaciones, oxigeno=oxigeno) #Crea
            automaticamente valores en la tabla de la base de datos
            return JsonResponse({'message': 'Datos recibidos y almacenados exitosamente'})
        except json.JSONDecodeError:
            return JsonResponse({'error': 'Error al analizar JSON'}, status=400)
    return JsonResponse({'error': 'Método no permitido'}, status=405)
```

## Base de datos

Para la base de datos utilizamos SQLite 3, la cual ya viene integrada con Django. Para la creacion de una base de datos solamente se debe de utilizar el archivo models.py, escribiendo en este el codigo necesario para la creacion de una tabla para nuestra base de datos

```
from django.db import models
# Importa el módulo models de Django, que contiene las clases y campos necesarios
para definir modelos de bases de datos.
class datos(models.Model): #Creacion de la clase datos, la cual va a contener la tabla de
la base de datos
    pulsaciones = models.IntegerField() #aca se define la tabla con nombre pulsaciones
    oxigeno= models.DecimalField(max_digits=5, decimal_places=2) #aca se define la tabla
    con nombre oxigeno
    Glu= models.IntegerField()#se define la tabla con nombre Glu para almacenar datos de
    glucosa
    Move = models.IntegerField() #se define la tabla para almacenar movimiento
    Temp = models.IntegerField() #se define la tabla para almacenar temperatura
```

healthdatos_datos	
key	<b>id</b> : integer
◆	<b>BPM</b> : text
◆	<b>Spo2</b> : text
◆	<b>Glu</b> : text
◆	<b>Move</b> : text

## Plantillas y HTML

Las plantillas serian nuestros HTML realizados, en total teniendo nosotros 6 (about, blog, contact, home, projects y team). Al contener estos demasiadas lineas de codigo, a continuacion solo se mostraran los cambios realizados dentro de los mismos para la utilizacion de Django correctamente.

El caso a ejemplificar es de home.html

```
<!DOCTYPE html>
{% load static %}
<html data-wf-dom=>
```

Para cargar el Css y el Js hay que previamente hacer una carpeta, la cual tendra dentro el Style.css y el Script.js. Siendo esa linea de codigo, la cual le dice al HTML que debe cargar luego, cuando se llamen a los respectivos Css y Js, esa carpeta, y, dentro de ella, los ya mencionados Css y JS

Luego, algo que cambia a diferencia de HTML normal, es que Django utiliza la funcion `href="{% url "home" %}"` para lo que seria el linkeo de un HTML hacia otro, en este ejemplo, nosotros usamos botones de navegacion en nuestra web, los cuales al darles click, ejecutan una funcion.

Le decimos al Logo con la funcion de boton, que cuando se presione, la pagina se dirija hacia la URL de home. Ya mencionamos anteriormente, que, el tercer argumento de views, llama a la URL la cual contiene el HTML de home.

```
<div class="navigation-wrap"><a href="{% url "home" %}" aria-current="page" class="logo-link w-nav-brand w--current">
```

```
<nav role="navigation" class="navigation-items w-nav-menu"><a href="{% url "nosotros" %}" class="navigation-item nosotros w-nav-link">NOSOTROS</a><a href="{% url "team" %}" class="navigation-item tu-salud w-nav-link">TU SALUD</a><a href="{% url "contact" %}" class="navigation-item experiencia w-nav-link">TU EXPERIENCIA</a><a href="{% url "blog" %}" class="navigation-item mas-info w-nav-link">Más Info</a><a href="{% url "projects" %}" class="navigation-item app w-nav-link">APP</a><a href="mailto:healthband.aid@gmail.com" class="navigation-item mail w-nav-link">CONTACTANOS AL MAIL</a><a
```

Ejemplos con la barra de navegacion de “Nosotros, Tu salud, Tu experiencia, Mas info, App”

## Envio de datos Sensores hacia la Web

A continuacion se hara una conexion entre los codigos de Micropython utilizados para el envio de los datos hacia la web.

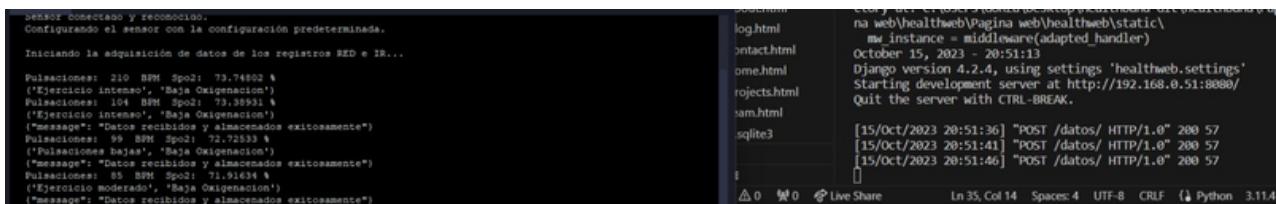
Oxi.py Micropython

```
def web():
    #Se declara una funcion llamada web para lo que seria el traspaso de datos sensados
    pulsaciones = Sensor.datos
    #Se vuelve a llamar a la variable pulsaciones que contiene lo medido en BPM
    Spo2= Sensor.datos2
    #Se vuelve a llamar a la variable Spo2 la cual contiene lo medido en Spo2
    if Sensor.datos > 40 and Sensor.datos < 130 and Sensor.datos2 > 50 and
    Sensor.datos2 < 110:
        #Se le dice a la funcion que SI Bpm esta entre 40 y 110 Y que SI Spo2 esta entre 50 y
        110, entonces ejecute lo siguiente
        info = { 'PULSOS' : pulsaciones , 'OXIGENO' : Spo2 }
        #Se declara la variable info que contiene Pulsaciones y Spo2
        g = requests.post('http://192.168.0.51:8080/datos/', json=info)
        #Se hace un request.post, el cual envia lo sensado a la pagina web en forma de json, el
        cual este llama a la variable info
        print (g.text)
        #Se muestra en la pantalla lo enviado a la pagina web
        time.sleep (2)
    #El codigo espera 2 segundos para volver a ejecutar el if
while True:
    #Se crea un bucle para que web se ejecute constantemente
    web()
    time.sleep (2)
    #El codigo espera 2 segundos para volver a ejecutar web
```

views.py Django

```
def mediciones(request):
    if request.method == 'POST': #le pide que reciba metodos POST solamente
        try: #Bucle que intenta infinitamente
            data = json.loads(request.body) # Analiza el JSON recibido
            pulsaciones = data.get("PULSOS") #Agarra del JSON la data que tenga de nombre
            pulsaciones
            oxigeno = data.get("OXIGENO") #Agarra del JSON la data que tenga de nombre
            Spo2
            datos.objects.create(pulsaciones=pulsaciones, oxigeno=oxigeno) #Crea
            automaticamente valores en la tabla de la base de datos
        return JsonResponse({'message': 'Datos recibidos y almacenados exitosamente'})
    except json.JSONDecodeError:
        return JsonResponse({'error': 'Error al analizar JSON'}, status=400)
    return JsonResponse({'error': 'Método no permitido'}, status=405)
```

Haciendo aquello la conexión entre la ESP32 y la Pagina web



The image shows two terminal windows side-by-side. The left window is a serial monitor showing data from the MAX30102 sensor. The right window is a Django development server log showing successful POST requests.

Left Terminal (Serial Monitor):

```
SENSOR CONECTADO Y RECONOCIDO.  
Configurando el sensor con la configuración predeterminada.  
Iniciando la adquisición de datos de los registros RED e IR...  
Pulsaciones: 210 BPM SpO2: 73.74902 % ("Ejercicio intenso", "Baja Oxigenacion")  
Pulsaciones: 104 BPM SpO2: 73.38931 % ("Ejercicio intenso", "Baja Oxigenacion")  
("message": "Datos recibidos y almacenados exitosamente")  
Pulsaciones: 99 BPM SpO2: 72.72533 % ("Pulsaciones bajas", "Baja Oxigenacion")  
("message": "Datos recibidos y almacenados exitosamente")  
Pulsaciones: 85 BPM SpO2: 73.91634 % ("Ejercicio moderado", "Baja Oxigenacion")  
("message": "Datos recibidos y almacenados exitosamente")
```

Right Terminal (Django Dev Server):

```
python manage.py runserver  
[15/Oct/2023 20:51:36] "POST /datos/ HTTP/1.0" 200 57  
[15/Oct/2023 20:51:41] "POST /datos/ HTTP/1.0" 200 57  
[15/Oct/2023 20:51:46] "POST /datos/ HTTP/1.0" 200 57
```

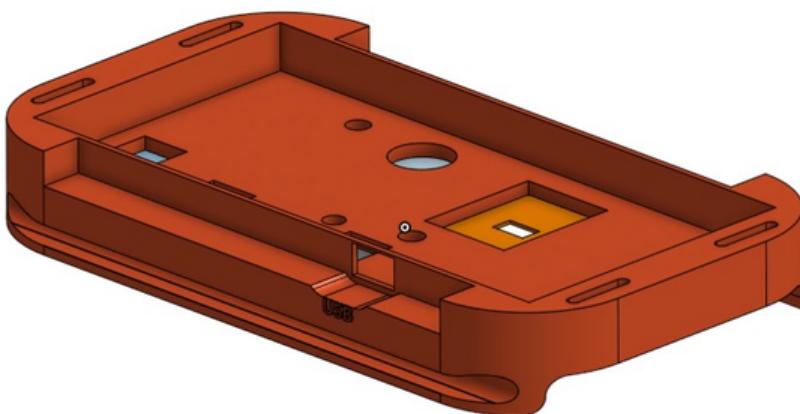
A la izquierda la consola de el MAX30102, recibiendo el mensaje de que los datos se enviaron correctamente, y, a la derecha, la consola de la pagina web la cual se le informa que le esta llegando un POST exitoso.

Para una mayor explicacion del codigo, se podra encontrá en el repositorio de GitHub de nuestro proyecto.

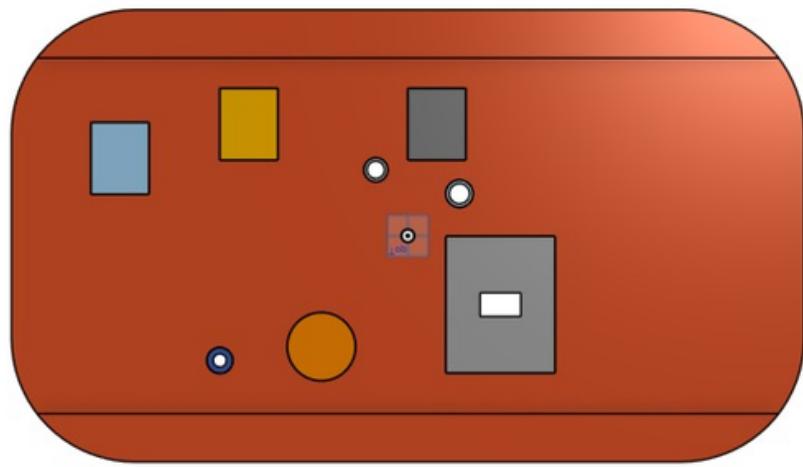
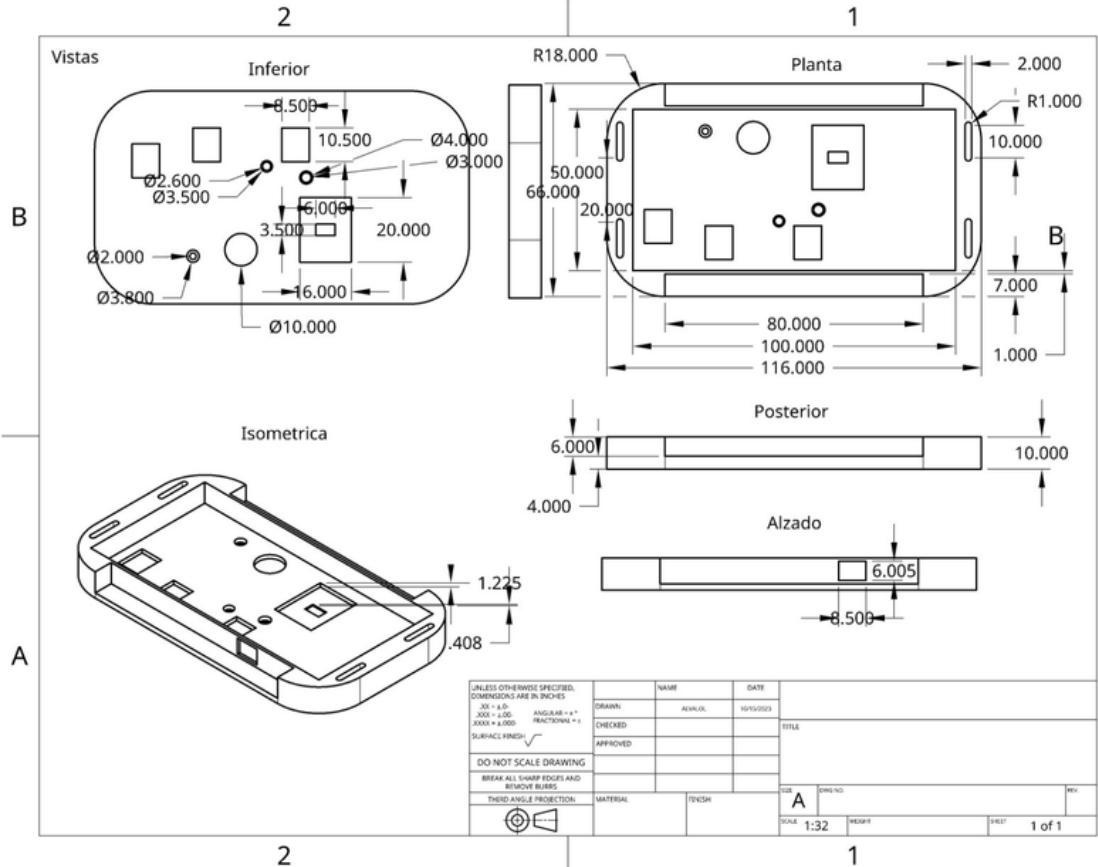
## Modelado 3D

### Estructura del Dispositivo

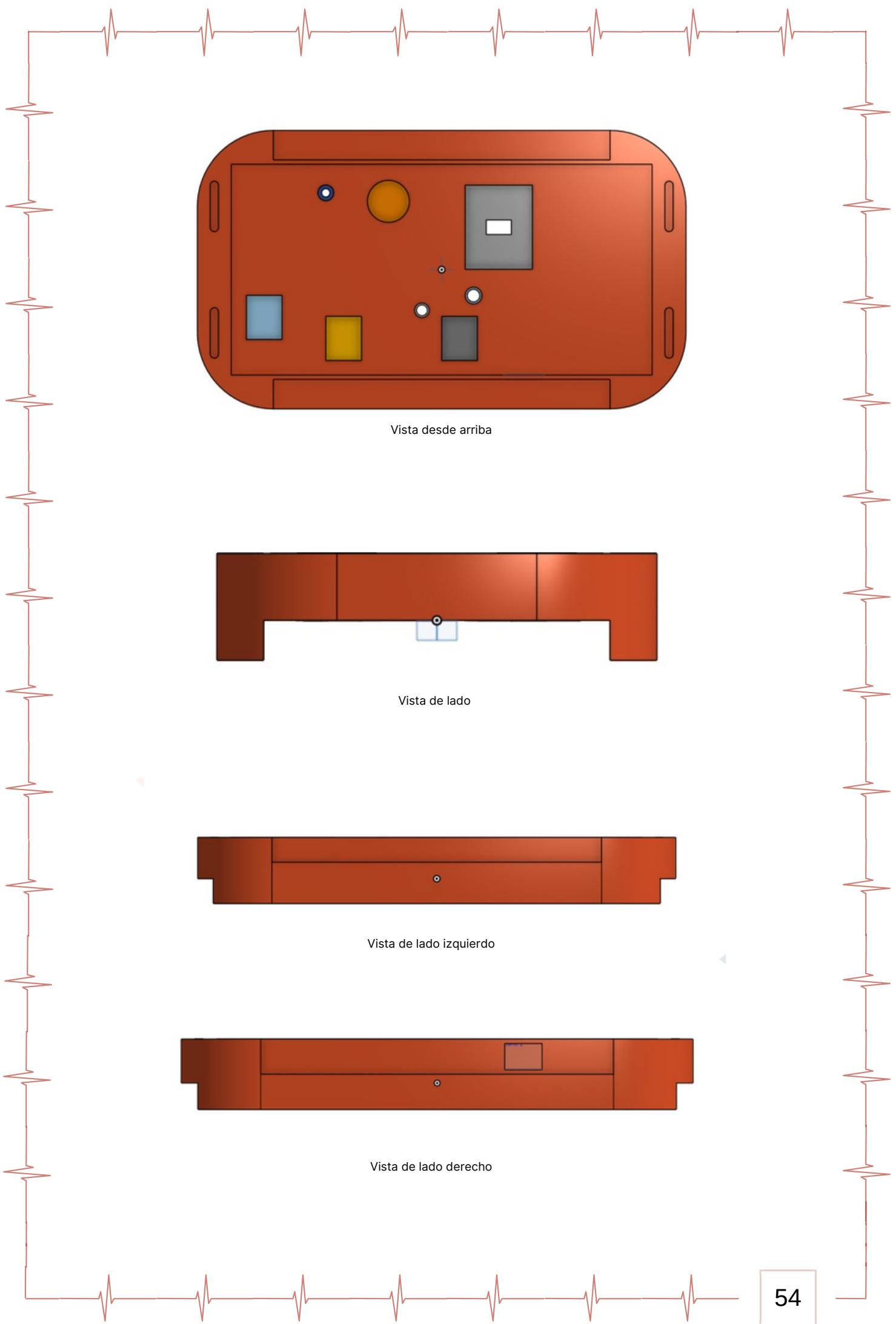
La placa diseñada en Kicad deberá ir montada sobre una estructura que sea capaz de contenerla y aislarla de las susceptibilidades del ambiente, así como ajustarla al cuerpo del usuario para un uso cómodo, priorizando tambien la ergonomía del modelo. Para ello se han utilizado herramientas de modelado 3D, comenzando por AutoCAD y luego transportando los diseños a Onshape.



La estructura se dividirá en 2 carcasa, una inferior y otra superior que se ensamblaran entre sí a través de encastres ubicados lateralmente. Al ser la placa de doble faz, con un grupo de sensores montados en su cara inferior.



Vista desde abajo



## Modelado PCB

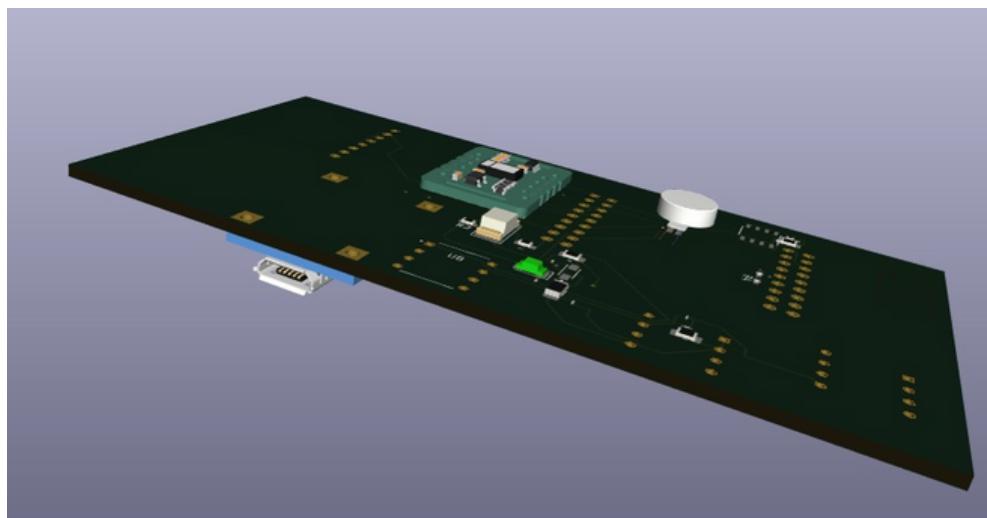
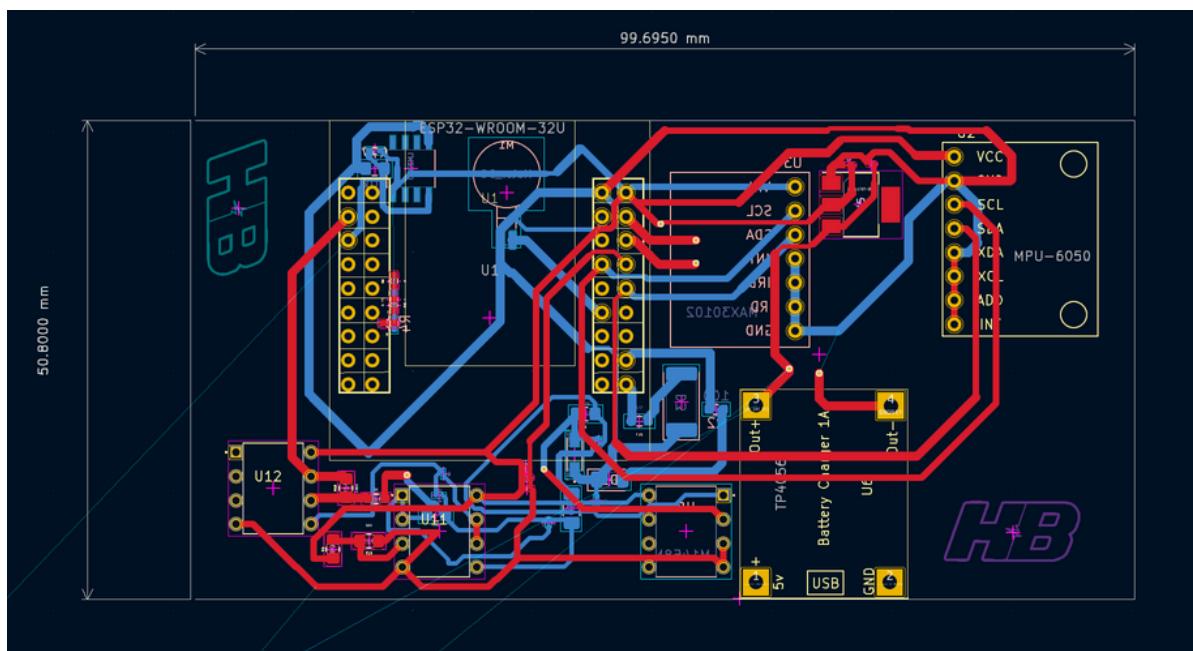
Para el desarrollo de una placa de circuito impreso (PCB) en doble faz utilizando KiCad, se han definido las siguientes disposiciones de componentes:

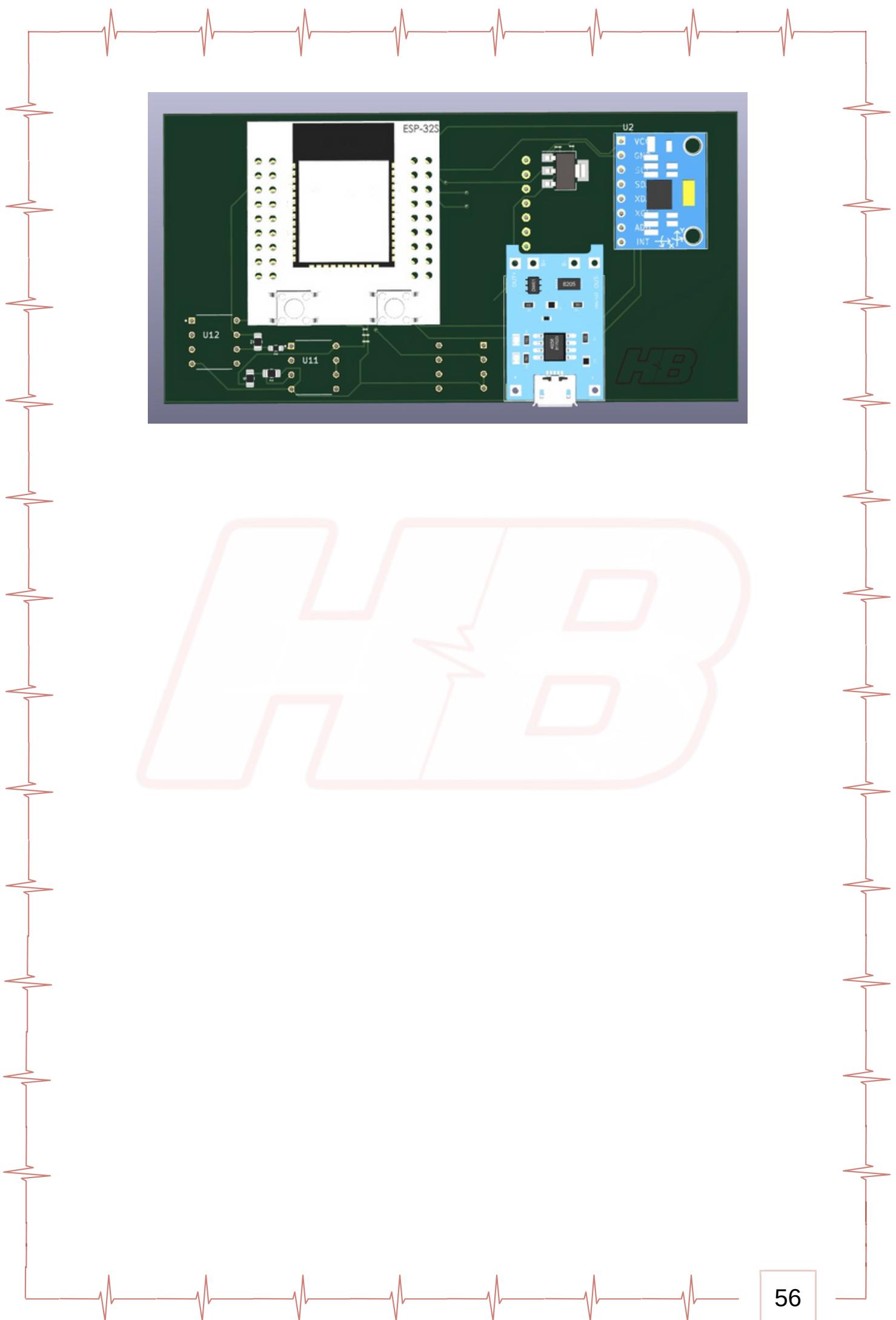
Parte Superior de la PCB:

1. ESP
2. MPU
3. Puerto de Carga
4. Parte de los Amplificadores y Filtros de Infrarrojos

Parte Inferior de la PCB:

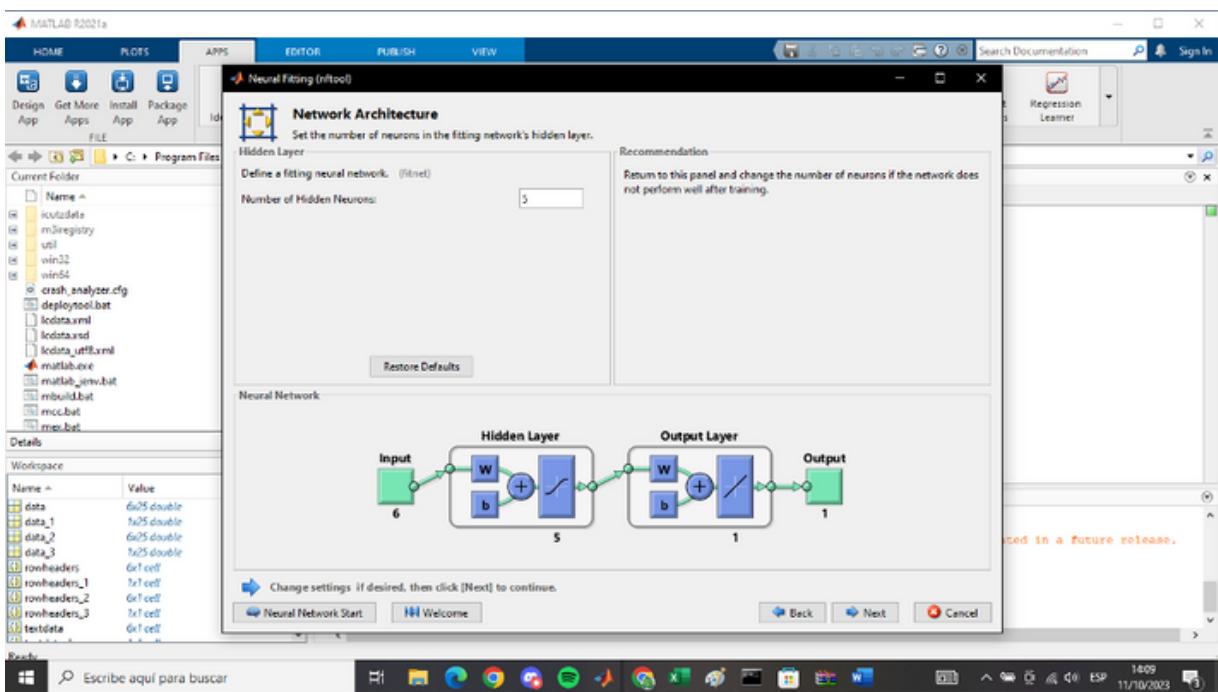
1. MAX
2. Sensor de Temperatura.
3. Sensores Infrarrojos
4. Motor de Vibración
5. Circuitos de Filtros y Amplificadores de Infrarrojos



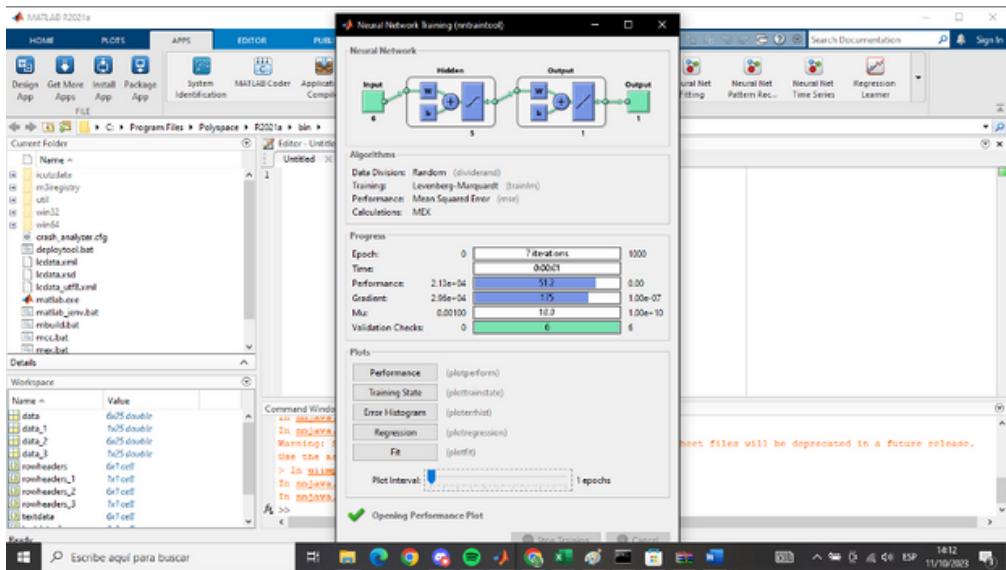


## Redes Neuronales

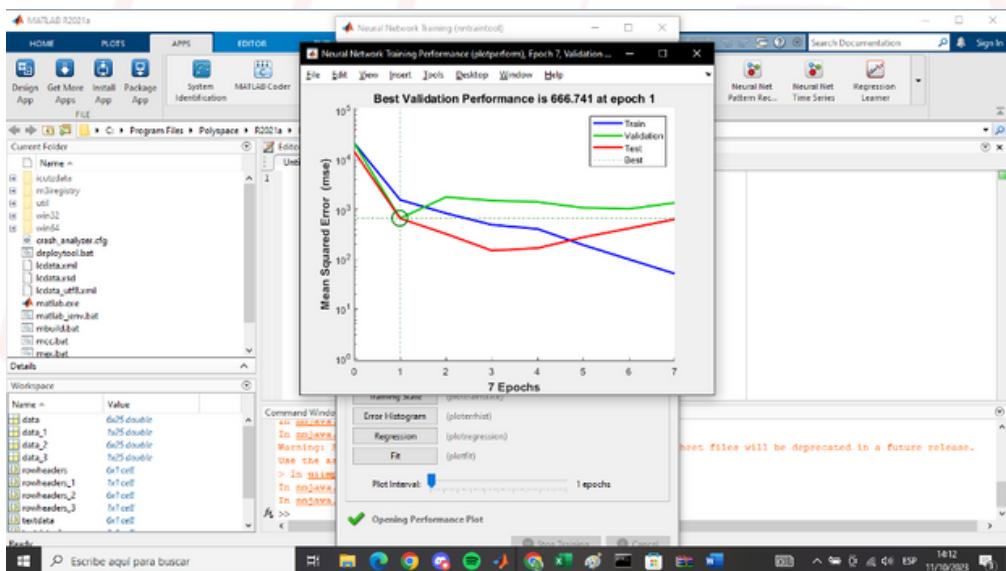
Como se ha mencionado anteriormente en el modulo MHL590. de la tensión amplificada que llega al pin de conversión analógica a digital del ESP32 es equivalente a la reflexión generada por las propiedades de la sangre. Sin embargo, la composición de glucosa en sangre se puede ver afectada por muchos factores biológicos, como el peso, la altura, el sexo o la edad. Tras un prolongado período de investigación, se encontró un proyecto similar del cual referenciarse para la construcción del sistema. Siguiendo este mismo se ha utilizado el programa de computación numérica MATLAB, para el desarrollo de un algoritmo experimental de red neuronal capaz de predecir los niveles de glucosa en sangre.



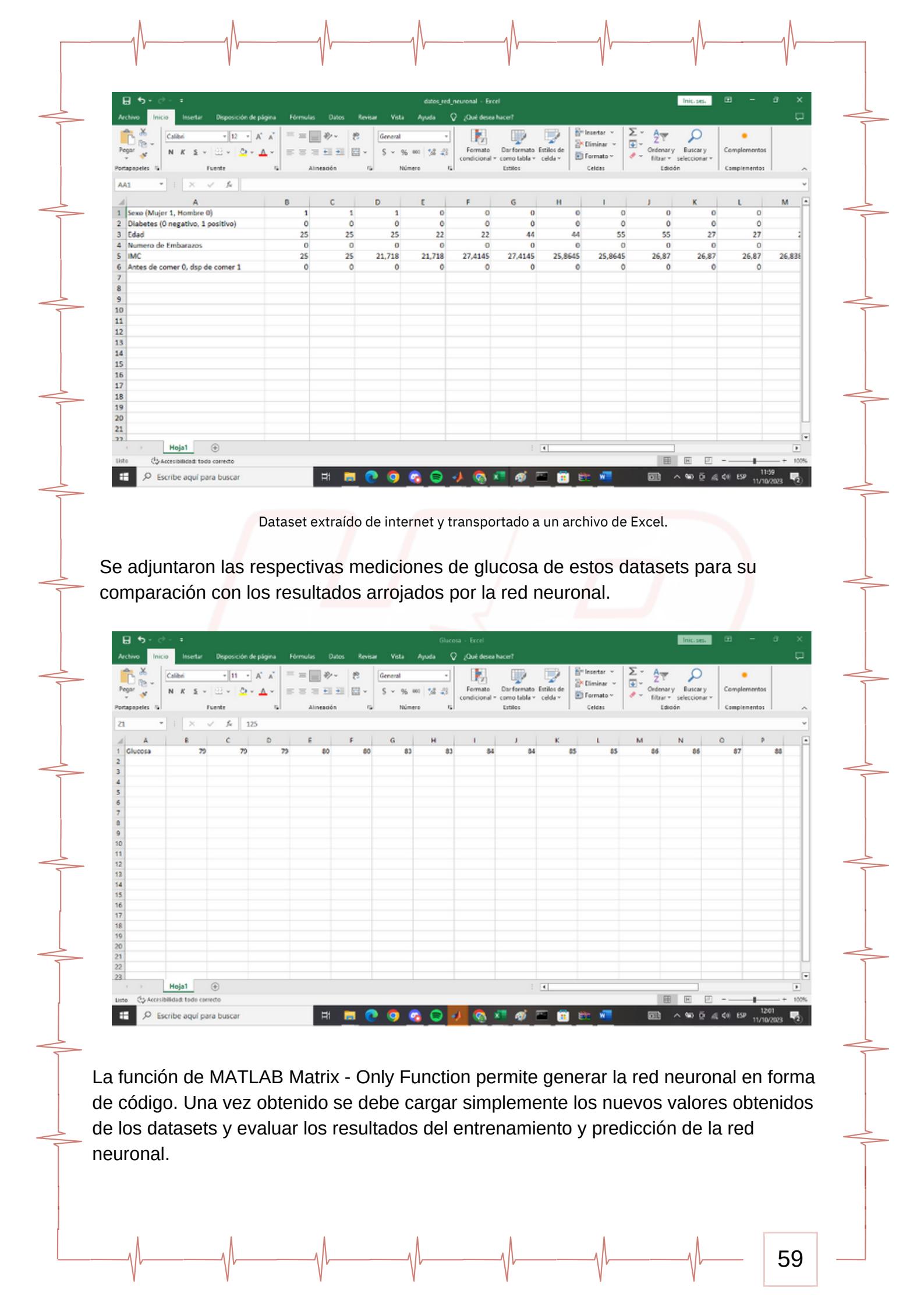
Tras diversas pruebas y evaluaciones de su rendimiento, se determinó la arquitectura de la red neuronal, compuesta de 6 entradas de datos, una capa oculta compuesta de 5 neuronas que son operadas con un algoritmo de función sigmoide, una capa de salida de una única neurona y comportamiento basado en una función lineal que se comunica con la salida de la red neuronal, la cual arroja una medición proporcional a sus entradas y el entrenamiento que recibió.



Resultados arrojados tras los períodos de entrenamiento y validación de la red neuronal visualizados desde el menú de la herramienta **Neural Network Training** (Superior), pudiendo visualizarse también a través de una gráfica con el comando **plotperform** (Inferior).



Por desgracia y debido a las dificultades técnicas en las etapas de prueba del esquema electrónico y en la previa instalación de MATLAB, el algoritmo no ha podido ser evaluado en personas físicas, siendo entrenado en cambio con datasets de carácter público pertenecientes a diversas investigaciones demográficas de la diabetes. Los datasets han sido extraídos de internet y compilados en archivos de Excel para ser cargados en la herramienta de entrenamiento de la red neuronal de MATLAB, Neural Network Training. Entre los distintos datos de entrada a evaluar se encuentran el sexo, la edad y el IMC, calculado como la relación entre peso y altura de un ser humano.

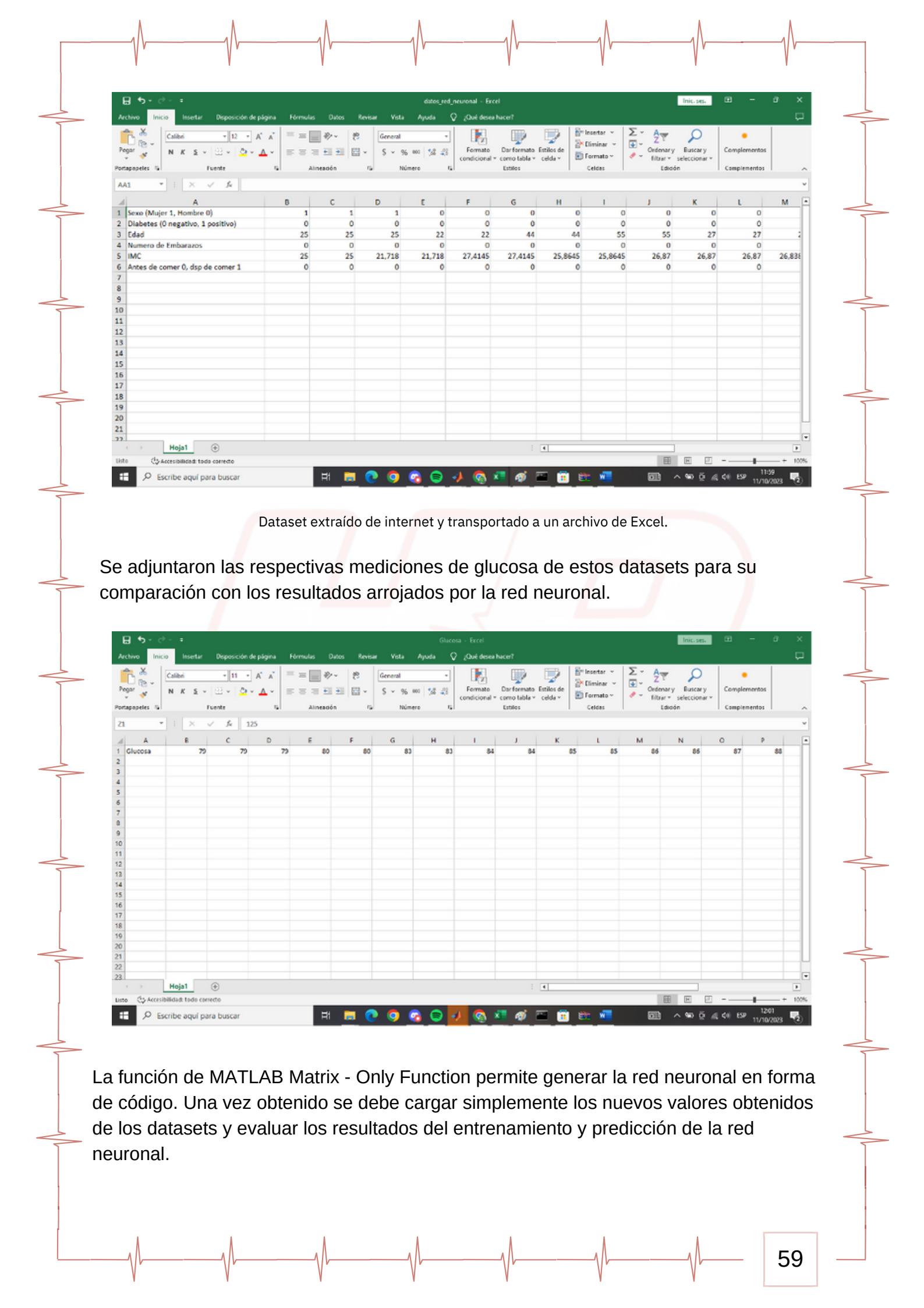


A screenshot of Microsoft Excel showing a dataset named "datos\_red\_neuronal.xlsx". The spreadsheet contains 21 rows of data across columns A through M. The data includes categorical variables like "Sexo (Mujer 1, Hombre 0)", "Diabetes (0 negativo, 1 positivo)", "Edad", and numerical variables like "Número de Embarazos" and "IMC". The last row shows values for "Antes de comer 0, desp de comer 1".

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Sexo (Mujer 1, Hombre 0)	1	1	1	0	0	0	0	0	0	0	0	0
2	Diabetes (0 negativo, 1 positivo)	0	0	0	0	0	0	0	0	0	0	0	0
3	Edad	25	25	25	22	22	44	44	55	55	27	27	
4	Número de Embarazos	0	0	0	0	0	0	0	0	0	0	0	0
5	IMC	25	25	21,718	21,718	27,4145	27,4145	25,8645	25,8645	26,87	26,87	26,87	26,838
6	Antes de comer 0, desp de comer 1	0	0	0	0	0	0	0	0	0	0	0	0
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													
21													
22													
23													

Dataset extraído de internet y transportado a un archivo de Excel.

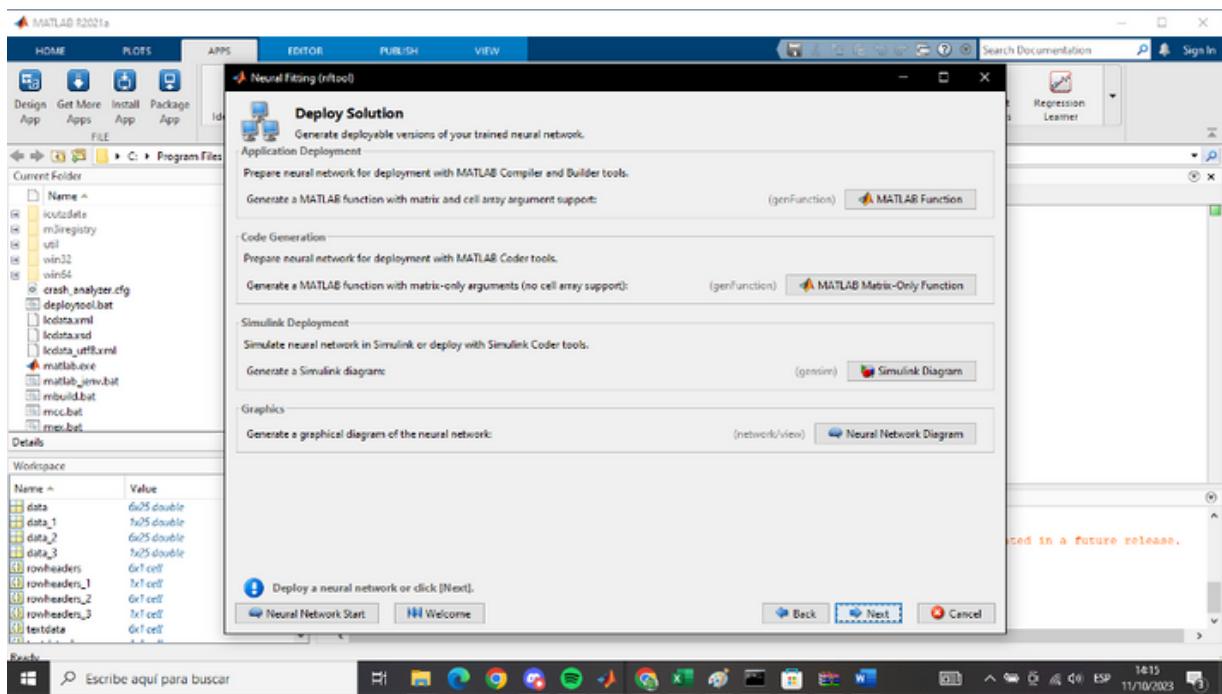
Se adjuntaron las respectivas mediciones de glucosa de estos datasets para su comparación con los resultados arrojados por la red neuronal.



A screenshot of Microsoft Excel showing a dataset named "Glucosa.xlsx". The spreadsheet contains 22 rows of data across columns A through P. The data includes a single column labeled "Glucosa" with various numerical values such as 79, 80, 83, 84, 85, 86, 87, and 88.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Glucosa	79	79	79	80	80	83	83	84	84	85	85	86	86	87	88
2																
3																
4																
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																
16																
17																
18																
19																
20																
21																
22																
23																

La función de MATLAB Matrix - Only Function permite generar la red neuronal en forma de código. Una vez obtenido se debe cargar simplemente los nuevos valores obtenidos de los datasets y evaluar los resultados del entrenamiento y predicción de la red neuronal.



## Hardware electrónica aplicada

### Alimentacion

La placa recibe alimentación desde una batería de Polímero de Litio (LiPo), capaz de entregar 3,7 V de tensión continua y hasta 380 mAh de corriente. Puesto que la placa adaptadora de la ESP32 debe recibir 3,3V de alimentación y no cuenta con reguladores de tensión integrados, se debió colocar una etapa de regulación de tensión intermedia entre la batería y el ESP32, con un LM1117, el cual baja la tensión a la necesaria para la operación del microcontrolador con la colocación de unos capacitores entre sus entradas y tierra. Para más información sobre este componente, consultar su hoja de datos.

La alimentación por parte de la batería al regulador y microcontrolador es controlada por un botón con retención mecánica, el cual al pulsar dejará pasar alimentación a los circuitos de la placa, quedando en ese estado a menos que se lo presione nuevamente.

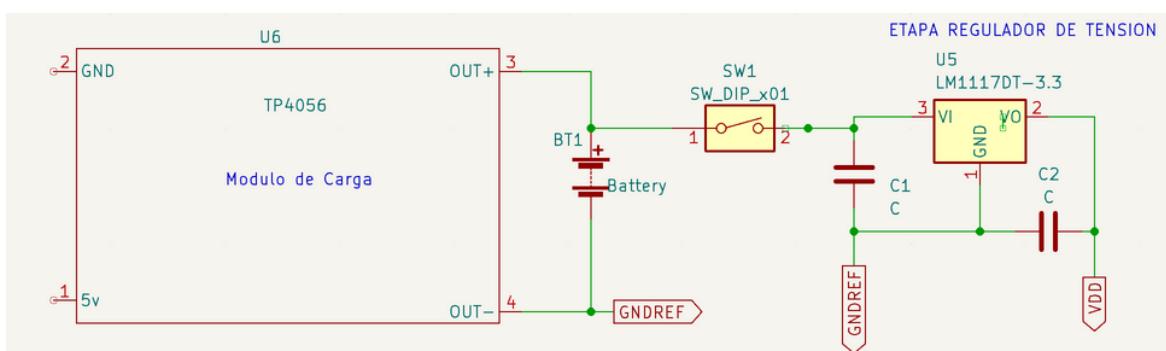
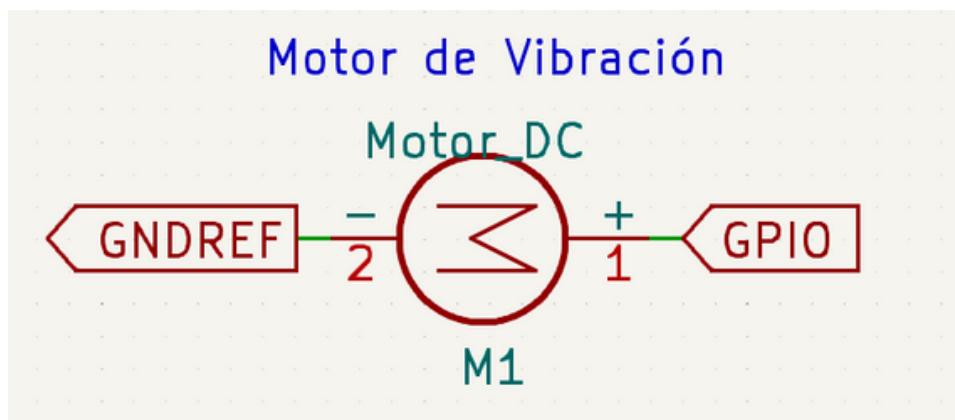


Diagrama de etapa de carga realizado en Kicad.

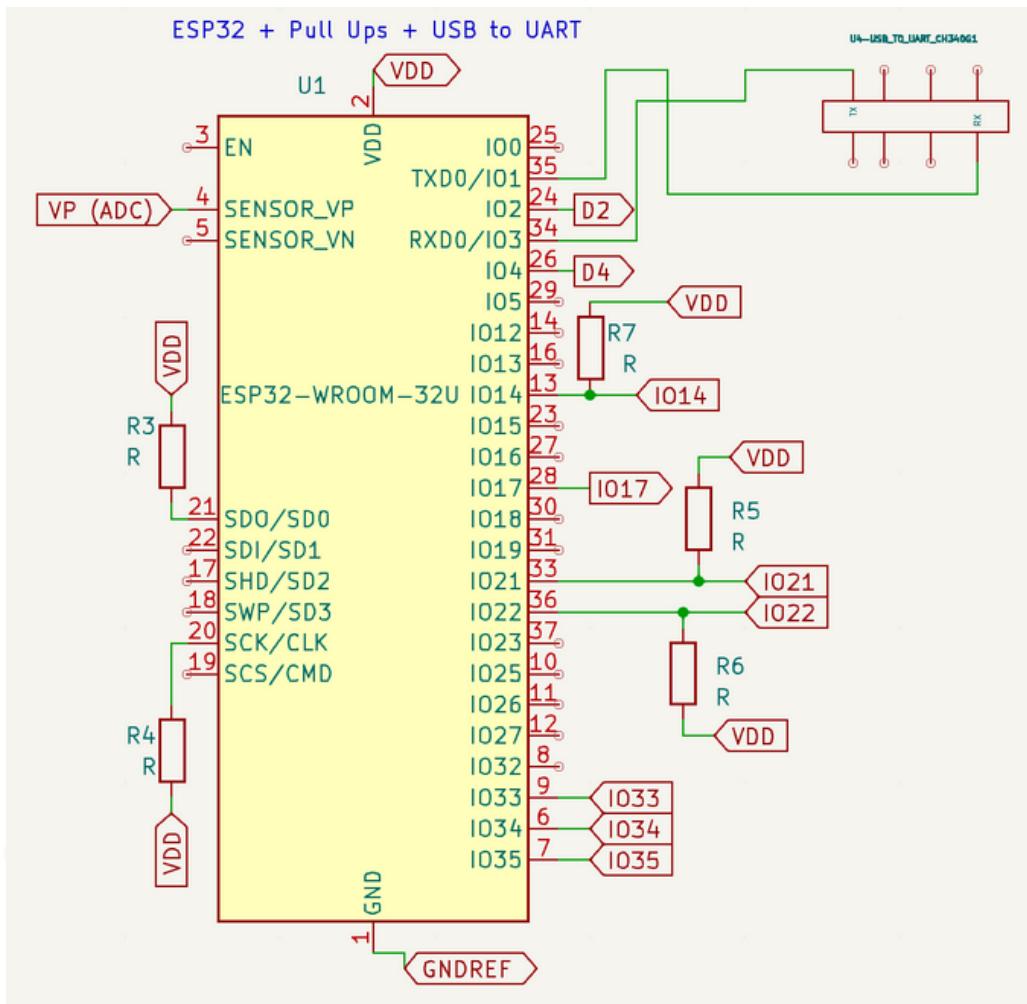
### Motor de vibracion

Para generar avisos en caso de detecciones anómalas, se incluye un pequeño motor de continua configurado para producir un movimiento vibratorio que alerte al usuario mientras lleva el dispositivo consigo.

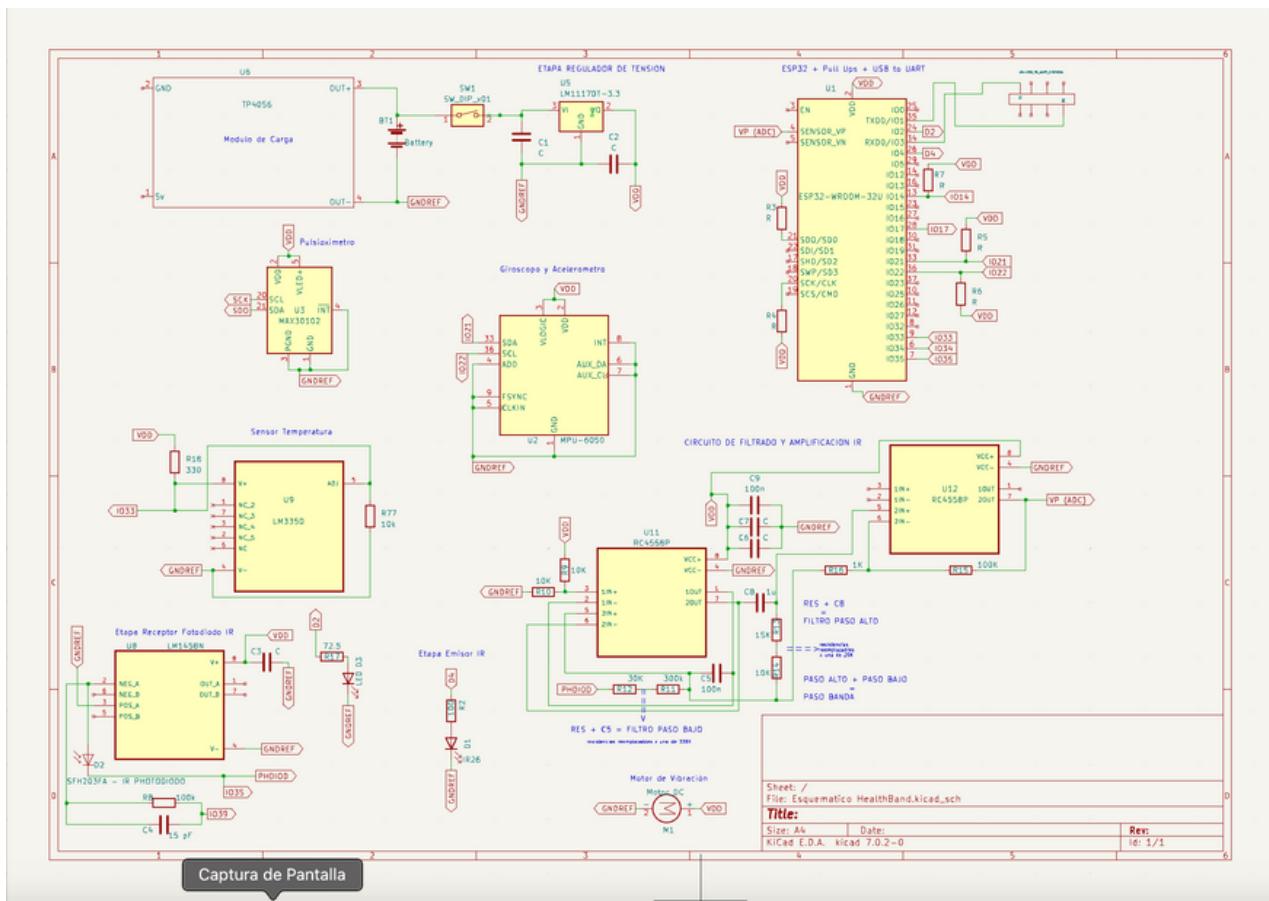


## Diagrama integral circuitos electronicos

En el apartado del microcontrolador se puede apreciar que incluye unas resistencias de tipo pullup, así como una conexión en sus terminales Rx y Tx a un puerto serial, el cual fue utilizado para configurar el microcontrolador con su firmware correspondiente y poder programar los diversos circuitos.



El diagrama integral de los circuitos electrónicos que componen la placa que irá montada internamente en el dispositivo fue realizado también en Kicad, y puede interpretar de la siguiente forma:



## Hosteo Web

### Adquisicion de dominio

Para hostear nuestra pagina web en internet, y, que esta tuviese el dominio [www.healthband.com.ar](http://www.healthband.com.ar), primeramente tuvimos que comprar el dominio.

Utilizando la plataforma de [nic.ar](http://nic.ar) se puede lograr la adquisición de un dominio solamente necesitando cuit/cuil y una clave fiscal de AFIP de nivel 2 en adelante

### Consulta de dominio

healthband .com.ar

No soy un robot  reCAPTCHA

BUSCAR

healthband.com.ar



El dominio no está disponible para registrarlo.

Podés encontrar debajo los datos del mismo. Si no son tuyos y tenés mejor derecho o interés legítimo sobre este dominio, podés iniciar una [Disputa](#). ¿Querés registrar otro nombre? Te sugerimos buscar uno que te represente.

#### Datos del dominio

Nombre y Apellido: GOMEZ GONZALO MARTIN GOMEZ GONZALO MARTIN  
CUIT/CUIL/ID: 23462877269  
Fecha de Alta: 09/10/2023  
Fecha de vencimiento: 09/10/2024

Como se logra apreciar, el dominio ya está registrado por nosotros a nombre de uno de los miembros del proyecto.



### El dominio está disponible para registrarlo

Acordate que los dominios '.bet.ar', '.coop.ar', '.gob.ar', '.int.ar', '.mil.ar', '.musica.ar', '.mutual.ar', '.org.ar', '.seg.ar', '.senasa.ar' o '.tur.ar' son zonas especiales, por lo que para obtenerlos, tenés que solicitar su habilitación. Conocé cómo hacerlo a través de nuestro [instructivo](#).

[QUIERO REGISTRARLO](#)

En el caso de tener una disponibilidad del dominio que queremos registrar, solamente hay que ir a quiero registrarlo y luego como se menciono anteriormente ingresar nuestros datos de Cuit/Cuil y clave Fiscal para que luego nos redirija a Tramites a Distancia y por ese medio poder efectuar el pago y asi mismo poder modificar el uso de nuestro ya dominio registrado

### Elegí cómo ingresar a Trámites a Distancia

Seleccioná una opción para realizar trámites de dominios.

Si contás con N° de CUIT/CUIL y Clave Fiscal nivel 2 o superior

[N° DE CUIT/CUIL Y CLAVE FISCAL →](#)

Si representás a una Persona Humana o Jurídica, recordá seleccionarla al ingresar.

Exclusivo para personas No Residentes

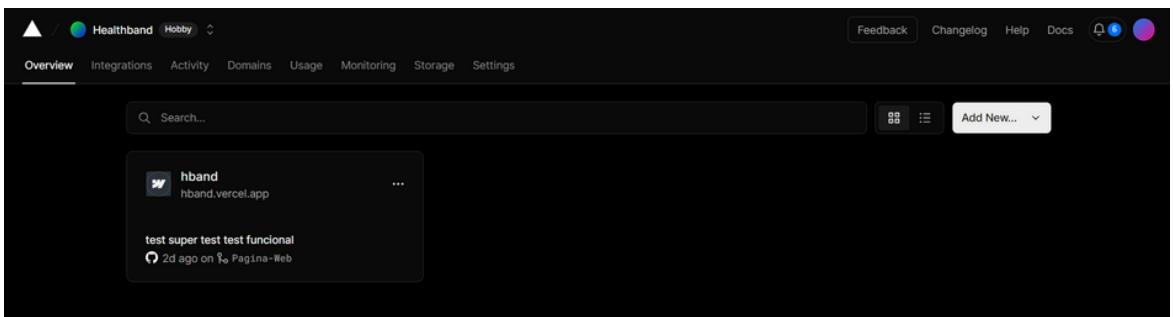
[USUARIO/ID Y CONTRASEÑA →](#)

[Olvidé mi contraseña](#)

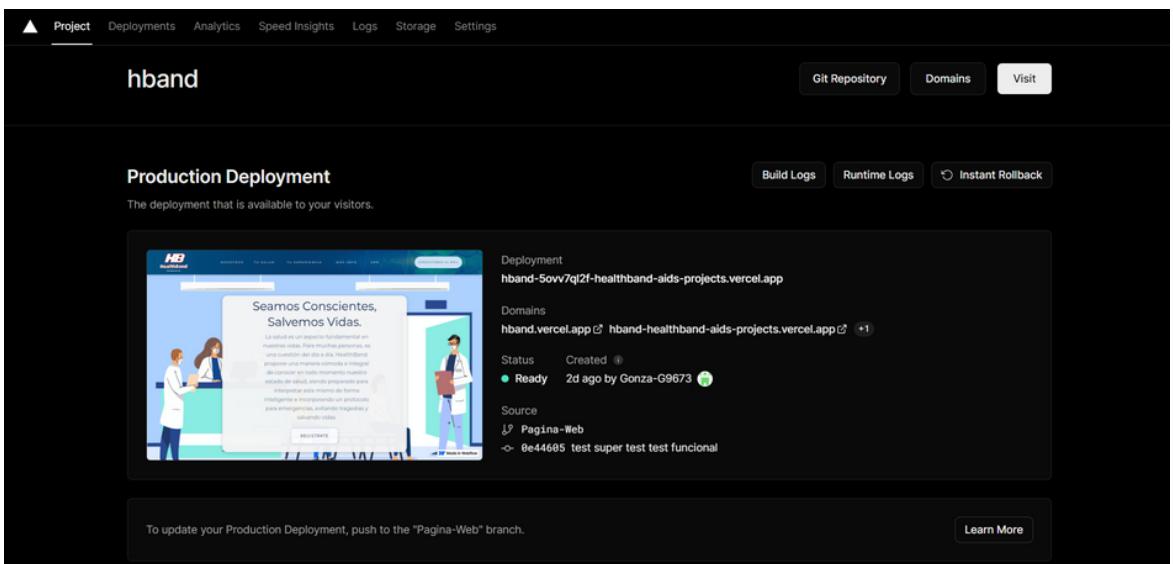
### Servidor web

Para poder tener nuestra pagina funcionando 24/7, se utilizo de un hosteo gratuito en un servidor web “Vercel”, siendo esta una pagina para el hosteo y funcionalidad de nube. Vercel funciona tomando todo el codigo de nuestra pagina web subido en GitHub, y, recrearlo en su propia plataforma

A continuacion se mostraran ejemplos de lo realizado:



Pestaña principal de vercel, la cual nos indica todos nuestros proyectos creado, en este caso el nuestro es hband.



Una vez entrado al proyecto, se logra apreciar varias cosas. Empezando por la parte superior se puede acceder a nuestro repositorio de GitHub, a los Dominios vinculados y un botón para ir hacia la pagina web, a su vez los logs, su runtime y, en el caso de tener algun cambio pendiente, actualizar la pagina con un Instant Rollback

En la parte central, se logra apreciar el deployment (siendo este un link tambien para ir hacia la pagina web alternativo a www.healthband.com.ar), tambien se logra apreciar nuevamente todos los dominios vinculados. Nos muestra el estatus de la pagina (ready, es decir, funcional y activo). Nos brinda la fecha de creacion del proyecto. Para finalizar la parte central, nos indica la Rama de GitHub en la cual esta nuestra pagina web, y, los commit y push que le hacemos frecuentemente

Mencionado ya lo de commit y push, para poder actualizar la informacion o cualquier cambio que se haya realizado en nuestra pagina web, hay que hacer un commit y un push en github para que se efectue el cambio.

Para poder sincronizar el dominio comprado en Nic.ar, con nuestra pagina ya hosteada en Vercel, habria que dirigirse a la parte superior de settings, y, en la pestaña de dominio agregarlo. no sin antes en Tramites a distancias, configurar los DNS, para que queden a corde a los solicitados por Vercel

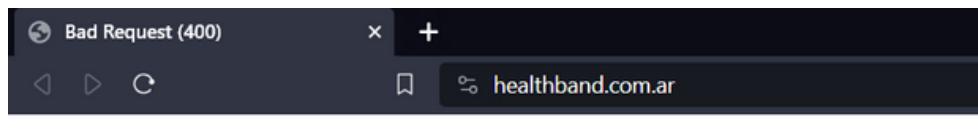
Dominio	Alta	Vencimiento	Estado	Delegado	Acciones
healthband.com.ar	09/10/2023	09/10/2024	Registrado	SI	<button>TRANSFERIR</button> <button>DELEGAR</button>

Pestaña de tramites a distancias. Primeramente hay que delegar nuestro dominio y luego registrar las DNS

+ Agregar una nueva delegación		+ Autodelegar	
Host	IPv4	IPv6	Acciones
ns1.vercel-dns.com			<button>EDITAR</button> <button>ELIMINAR</button>
ns2.vercel-dns.com			<button>EDITAR</button> <button>ELIMINAR</button>

## Permisos en Django

Con lo realizado anteriormente, si nos dirigimos hacia [www.healthband.com.ar](http://www.healthband.com.ar), nos va a aparecer un error de bad request (400)



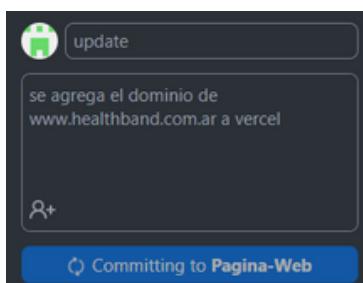
## Bad Request (400)

Para solucionar lo ocurrido, en un principio, no mencionado anteriormente porque se ajusta mas a esta situación. En Django hay que modificar el archivo settings.py de nuestro proyecto

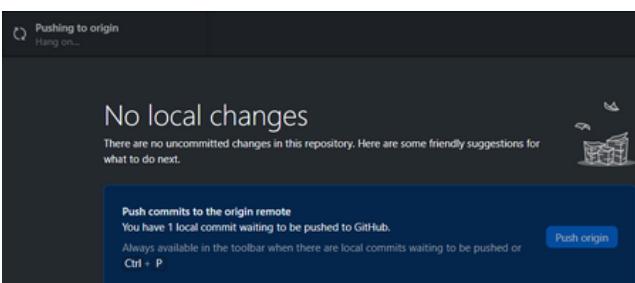
```
Página web > healthweb > healthweb > settings.py > ...
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'django-insecure-nhu$1&ic7gokzjia%zg08&lody)iy-8#y9qu^x+mj-^+9tm'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = False
27
28 ALLOWED_HOSTS = ['.vercel.app', 'healthband.vercel.app', '127.0.0.1', '192.168.111.4', '192.168.0.51', 'www.healthband.com.ar']
29
30 ##
```

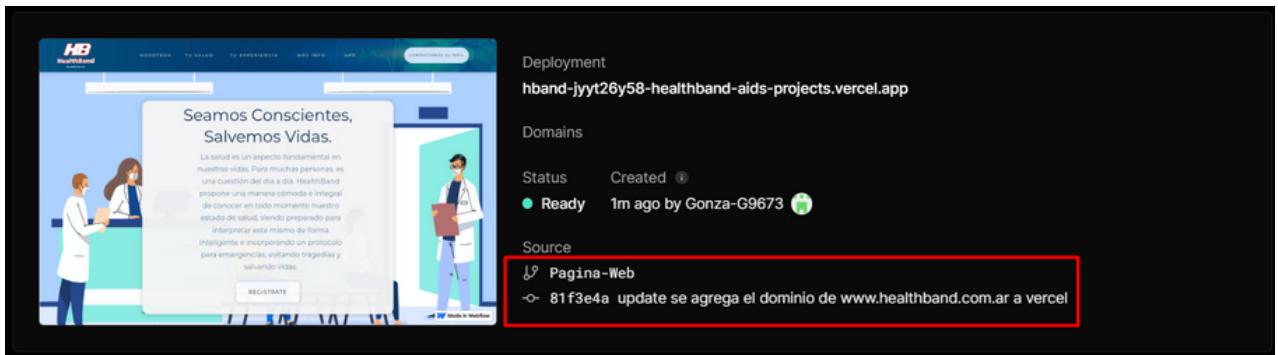
Como se logra apreciar, en ALLOWED\_HOSTS, solamente hay 5 dominios permitidos por nuestro proyecto, para que este pueda ser levantado en esas direcciones. así que únicamente hay que agregar y permitir el dominio de [www.healthband.com.ar](http://www.healthband.com.ar)

A screenshot of a GitHub interface showing a commit to the "settings.py" file. The commit message is "se agrega el dominio de www.healthband.com.ar a vercel". The code change adds "www.healthband.com.ar" to the ALLOWED\_HOSTS list.



A la izquierda, se logra apreciar el commit, y a la derecha, el push hacia nuestro repositorio de GitHub vinculado con vercel





Se logra apreciar en los commit, que a vercel le llego la actualizacion del repositorio de GitHub, asi que los cambios de Django se verian realizados, y, por ende, la pagina pasaria a ser visible y funcional.

