



Carpeta de Campo BIAS

Integrantes del Proyecto:

- Adell, Nicolas Fabian
- De Blasi, Luca
- Diaz Melion, Danilo Sebastian
- Gil Soria, Ian Lucas
- Montenegro, Luciano Nahuel
- Sojka, Santiago Alejandro

0.1. Adell, Nicolás Fabián



0.2. De Blasi, Luca



0.3. Diaz Melión, Danilo Sebastián



0.4. Gil Soria, Ian Lucas



0.5. Montenegro, Luciano Nahuel



0.6. Sojka, Santiago Alejandro



1. Marzo de 2024

Inicio de anteproyecto e investigación sobre la utilización de MATLAB, Python y cómo vincular el Neurosky Mindwave Mobile 2 a una computadora o celular. Decisión de elección de microcontrolador y hardware a utilizar.

<https://theses.hal.science/tel-04011453v1/document>

Realización y testeo del código para poder mover la silla de ruedas mediante un joystick. Investigación sobre los posibles materiales a utilizar en el proyecto ya existentes en la escuela. Hemos visto las posibles sillas de ruedas a utilizar en el proyecto. Investigación del sensor ultrasónico a utilizar en el proyecto.

Hemos empezado a realizar la programación de los motores y la programación del ultrasonido y se han probado en protoboard, probando que funcionan. También se han hecho ciertas modificaciones al anteproyecto, y se ha empezado a hacer el diagrama temporal.

```
1 from machine import Pin, ADC
2 from time import sleep, sleep_us, ticks_us
3 from math import atan2, degrees
4
5 # Establecer variables globales para ADC
6 NUM_STEPS = 2 ** 16
7 MAX_RADIUS = (NUM_STEPS - 1) / 2
8 TOLERANCE_STEP_IN_REST = 7500
9
10 def main():
11     # Setear los pines de ADC
12     pinx = ADC(Pin(26))
13     piny = ADC(Pin(27))
14
15     # Setear los ultrasonidos
16     trig_forward = Pin(16, Pin.OUT)
17     echo_forward = Pin(15, Pin.IN, Pin.PULL_DOWN)
18     trig_back = Pin(14, Pin.OUT)
19     echo_back = Pin(13, Pin.IN, Pin.PULL_DOWN)
20     trig_right = Pin(12, Pin.OUT)
21     echo_right = Pin(11, Pin.IN, Pin.PULL_DOWN)
```

Hemos hecho el logo del proyecto, hemos encontrado las baterías necesarias para poder mover la silla de ruedas, discutimos, tanto entre nosotros como con otros profesores, sobre la utilización de 1 o 2 motores, discutimos sobre los drivers y el puente H.



En el día de la fecha hemos creado la cuenta de la red social instagram junto al mail mismo del proyecto.

proyecto.bias

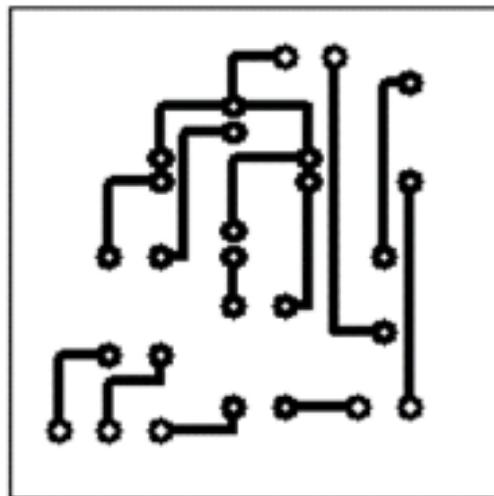
Editar perfil Ver archivo Herramientas

15 publicaciones 629 seguidores 38 seguidos

BIAS
Estudiantes del IMPA 7º 2º Avionica
Silla de ruedas con movimiento asistido por actividad cerebral
linktr.ee/biasproject

4.2 mil cuentas alcanzadas en los últimos 30 días. Ver insights

Hemos realizado el esquemático junto al diseño del pcb del sistema de emergencias, por lo que quedaría imprimirlo y ya ir armando la placa.



Investigamos la datasheet del driver del motor, empezamos a investigar el circuito interno y las posibilidades a la hora de manejar el motor.



Empezamos a desarrollar el programa para conectar el Neurosky a un módulo HC-05.

```
from machine import Pin, UART
from utime import sleep_ms

md = False

def main():
    uart = UART(0, baudrate=115200, tx=Pin(0), rx=Pin(1))

    enable = Pin(4, Pin.OUT)

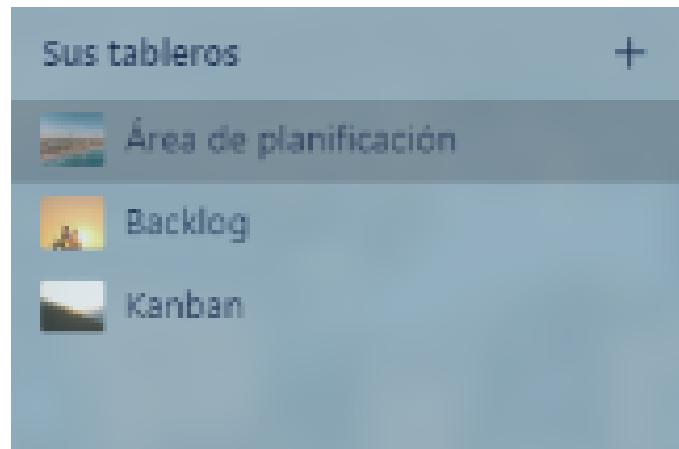
    # uart = UART(0, baudrate=38400, tx=Pin(0), rx=Pin(1))

    # dirección MAC: (recomienda usar la real)
    mac_address = "E819_A1_0007ed"
    # mac_address = "hc10_7b_0x189f"
    # mac_address = "acca_fc_0x6d9b"
    uart.write("AT\r\n".encode())
    sleep_ms(1000)
    print(uart.read().decode())
```

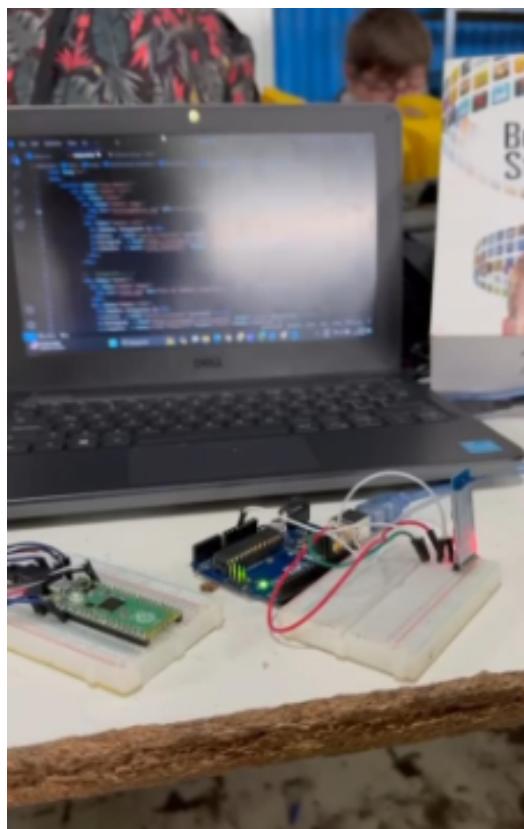
```
ble to connect to COM6: port not found
process ended with exit code 1.
```

2. Abril de 2024

Hemos empezado a organizarnos con la plataforma trello para tener buena percepción sobre las tareas a realizar.



Estuvimos buscando la forma de conectar el Neurosky a la computadora a través de bluetooth, y terminamos optando por seguir con el HC-05, ya que no se conecta directamente y las versiones no son compatibles.



Hicimos las imágenes que vamos a publicar al Instagram a la hora de que verifiquen el LinkedIn, seguimos desarrollando el programa para usar el módulo bluetooth (HC-05), empezamos a hacer un plano de la silla de ruedas para hacer las modificaciones de la estructura que tengamos que hacer.



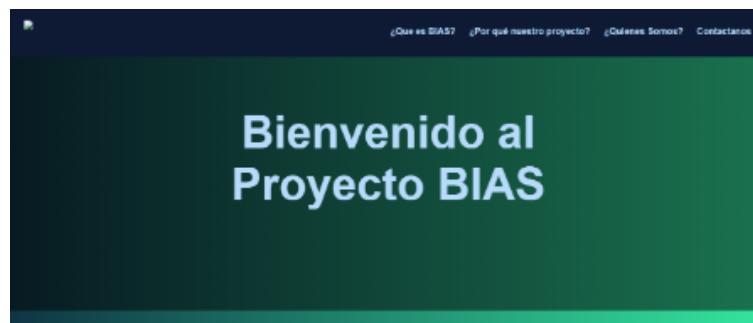
Se consiguió con éxito poder establecer la conexión Bluetooth del HC05 a una computadora



Cargamos las baterías y empezamos a probar los motores junto al driver, investigamos sobre otras opciones para reemplazar al Neurosky, ya que no hay forma de hacerlos vincular..



Desarrollamos la página del proyecto, creamos el GitHub donde estará todo tipo de documento y código relacionado al proyecto y una cuenta de TikTok.

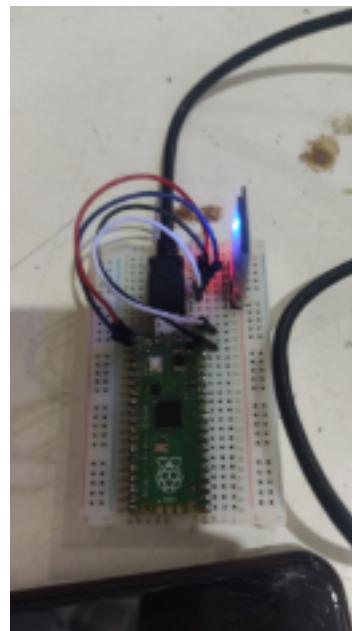


Seguimos con la programación del Neurosky Mobile Mindwave 2, además de crear publicaciones para difundirnos por Instagram.

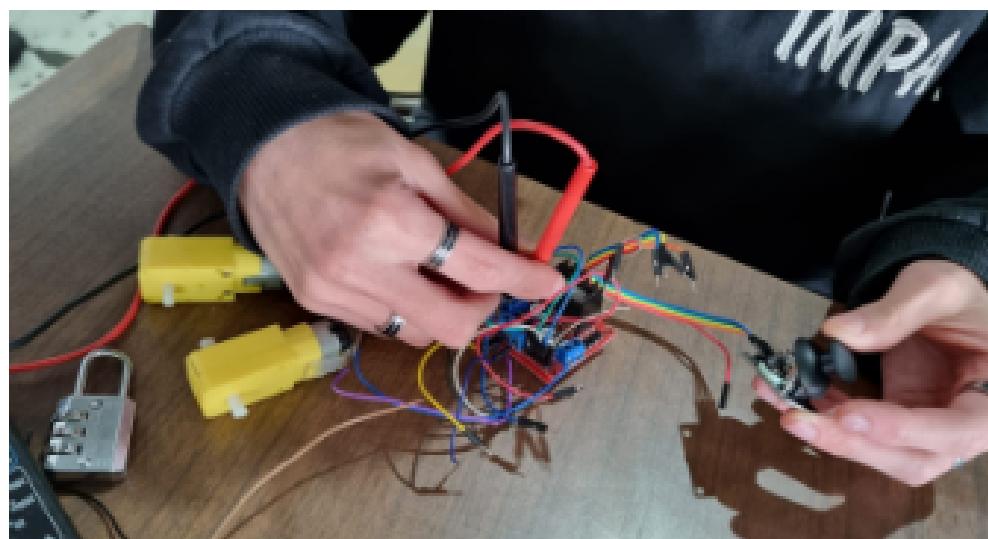


3. Mayo de 2024

Probamos conectar el Neurosky con Arduino pero sigue sin funcionar. Por eso buscamos otras alternativas para el EEG, ya que debido a los intentos reiterados no funciona.



Se investiga el driver si puede funcionar y si es viable utilizarlo o hacer el propio.



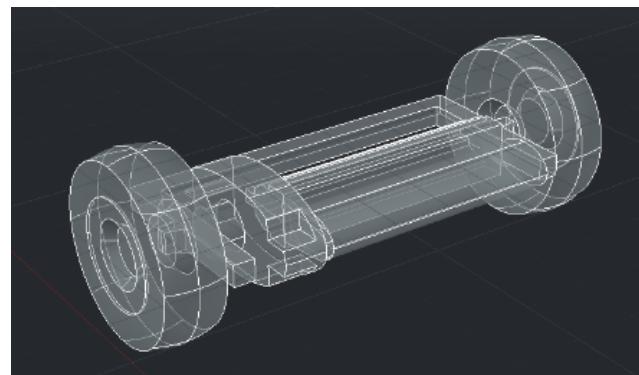
Fuimos a la radio en Solano a hablar del proyecto. Nos contactamos con la radio FM Sur.



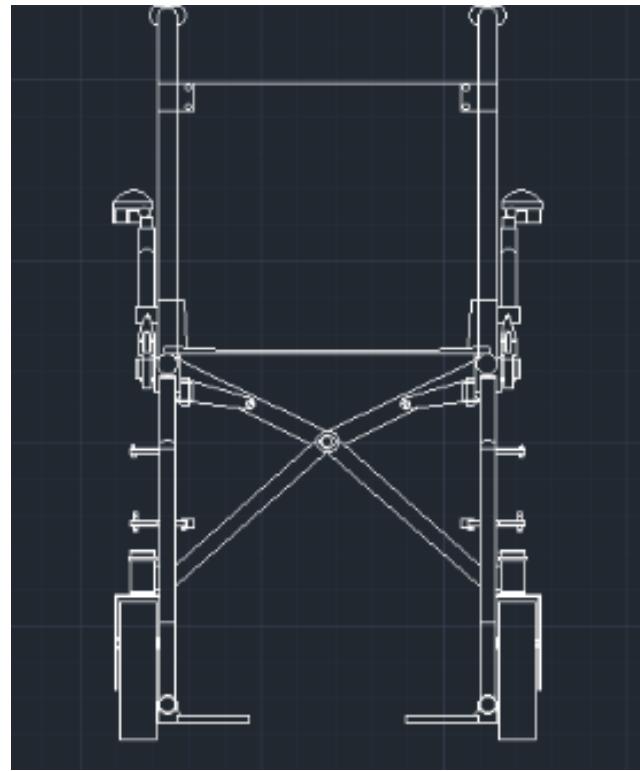
Probamos el driver y el motor originales que venían con la silla de ruedas. Se realiza un informe sobre el Driver ya que el mismo no funciona.



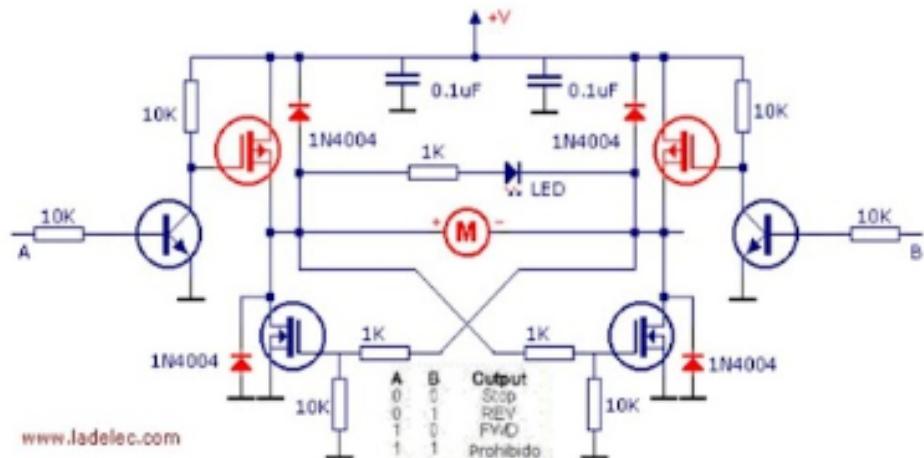
Se realizó el plano de la silla de ruedas en Autocad. Se busca la forma de diseño del Driver. Nos contactamos con sponsors para darnos difusión. Organizamos el Trello.



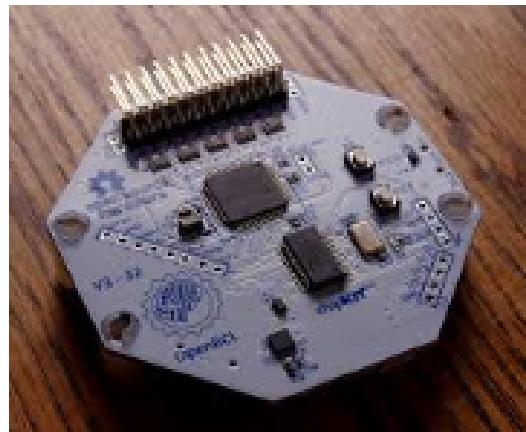
Tuvimos la entrevista radial en FM Sur. Se realiza el reel n.º 3. Se continúa con el plano de la silla. Se organiza el Github. Se buscan alternativas al Neurosky.



Se realiza el diseño del driver para el motor. Finalmente se toma el Open BCI como alternativa para realizarla por nosotros mismos debido a los costes de fabricación. Para eso se busca la documentación.



Nos contactamos con ALAPA, con la Fundación Esteban Bullrich y con la política María Sotolano. Vemos la alternativa del Open BCI



Realizamos una entrevista para que Clarín haga una nota sobre nosotros.

Clarín

● Envíos | Ley Kicillof: paquete fiscal de Millet | Desaparición de Lucas | Dólar blue hoy | Gran Hermano | Eurocopa y Copa América

Ingresar

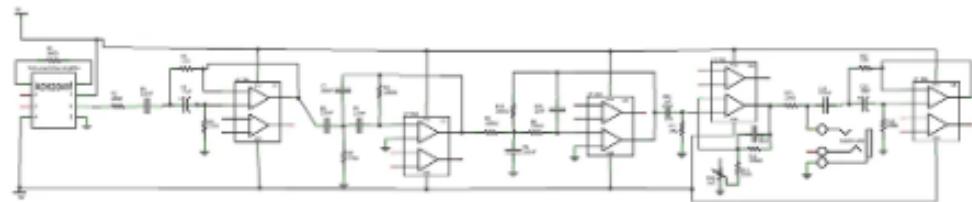
Zonales

Una silla de ruedas que se controla con la mente: la creación de estudiantes argentinos

Son alumnos de una escuela técnica que están desarrollando un prototipo que utiliza Inteligencia Artificial.
Aseguran que con sólo imaginar el movimiento, la silla se podrá mover en la dirección deseada.

Buscamos de un EEG de reemplazo que tenga un coste aceptable. Elegimos entre el Open BCI y otro circuito. Tomamos el ejemplo de la página siguiente:

<https://www.instructables.com/DIY-EEG-and-ECG-Circuit>



Fuimos al Congreso a hablar con la diputada María Sotolano. Logramos asegurar el contacto con la asociación Esteban Bullrich.

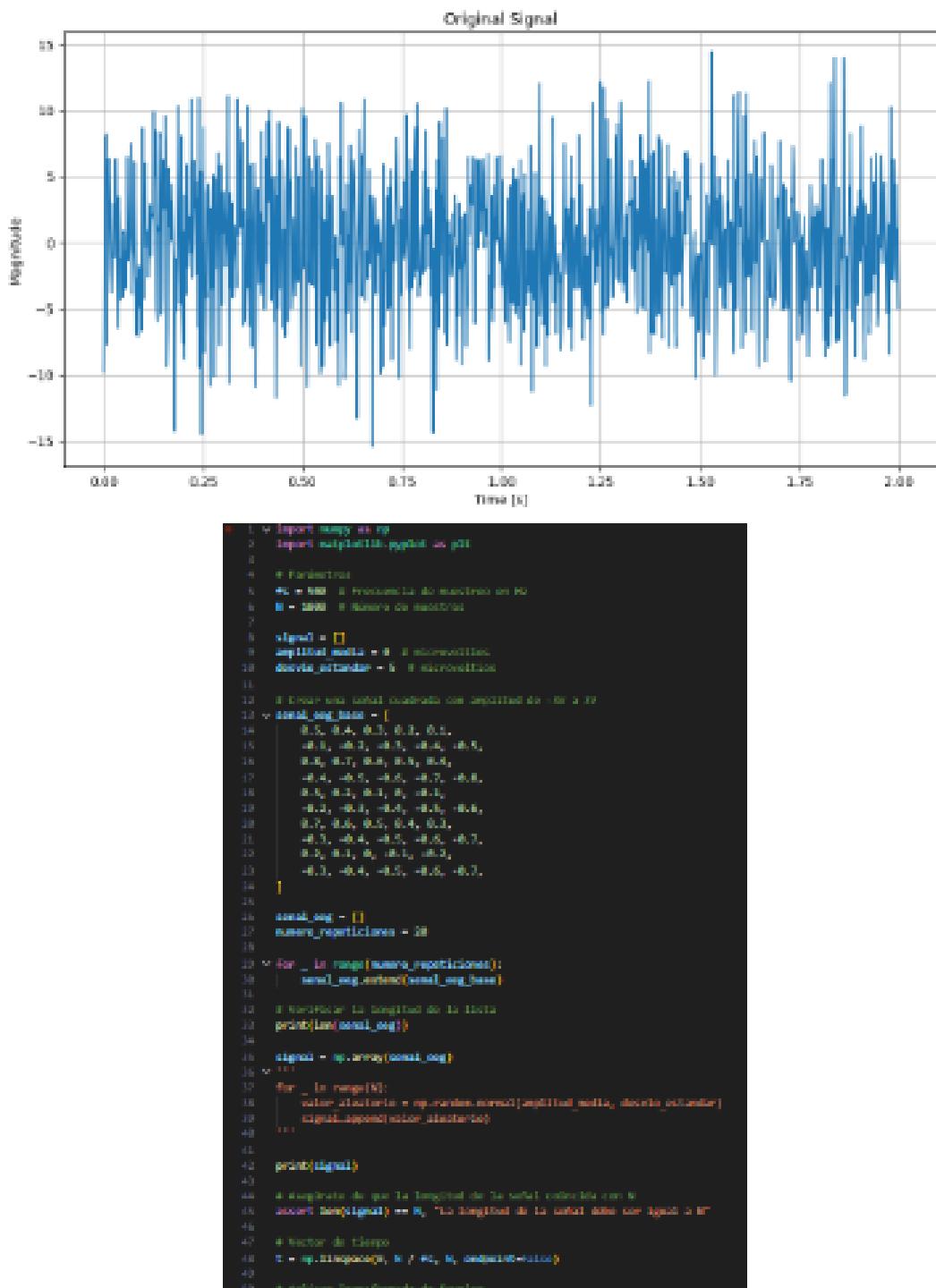


TN nos entrevista, por lo que preparamos un stand para presentar en la televisión.



4. Junio de 2024

Damos entrevista con Radio Nacional. Se comunica con nosotros TV pública y Radio Halley. Se hace diseño en KiCad del EEG. Se empieza a programar EEG mediante transformada de Fourier. Luego se filtra la señal y se discrimina por tipo de frecuencia. Se consigue joystick del motor.



Se dio una entrevista en Radio Eco. Nos contactó la Radio de la Jauretche. Se optimiza el código y se lo hace aplicando transformada de Fourier inversa. Se prueba el motor y no funciona. Se empieza a programar en la Raspberry Pi 4 con respecto a los motores.



También se dio una entrevista en Radio Halley. Se armó el tríptico, textos para imprimir y las presentaciones para el Stand en la Técnica 3 de Solano. Se desestima el ADC 1115 para utilizar la Raspberry Pi Pico mini, ya que tiene una mayor frecuencia de switcheo. Se empieza a armar el código de la Inteligencia Artificial y se arma el código de I2C para conectar la Raspberry Pi Pico con la Pi 4. Luego se programa a la Raspberry Pi Pico para utilizar su ADC.

```
#include "pio/pio.h"
#include "hardware/i2c.h"
#include "hardware/adc.h"
#include "hardware/led.h"

#define I2C_SLAVE_ADDRESS 0x04
#define I2C_PORT I2C0

void i2c_slave_init(uint8_t *i2c, uint8_t address);
void i2c_slave_handler();
void initialize_pins_i2c(uint8_t *i2c, uint8_t sda, uint8_t scl, int baudrate);
void initialize_i2c(uint8_t *i2c, uint8_t address, uint8_t sda, uint8_t scl, int baudrate);
void initialize_adc(uint8_t *adc_pins, uint8_t adc_input);
uint8_t read_analog_value(uint8_t adc_input);

int main() {
    // Set ADC pins and channels
    const uint8_t ADC_PIN_1 = 26;
    const uint8_t ADC_INPUT_1 = 0;
    const uint8_t ADC_PIN_2 = 27;
    const uint8_t ADC_INPUT_2 = 1;
    const uint8_t ADC_PIN_3 = 28;
    const uint8_t ADC_INPUT_3 = 2;
}

// DUTSUT DEBUG CONSOLE TERMINAL PORTS MEMORY SERIAL MONITOR JUPITER RTOS
```

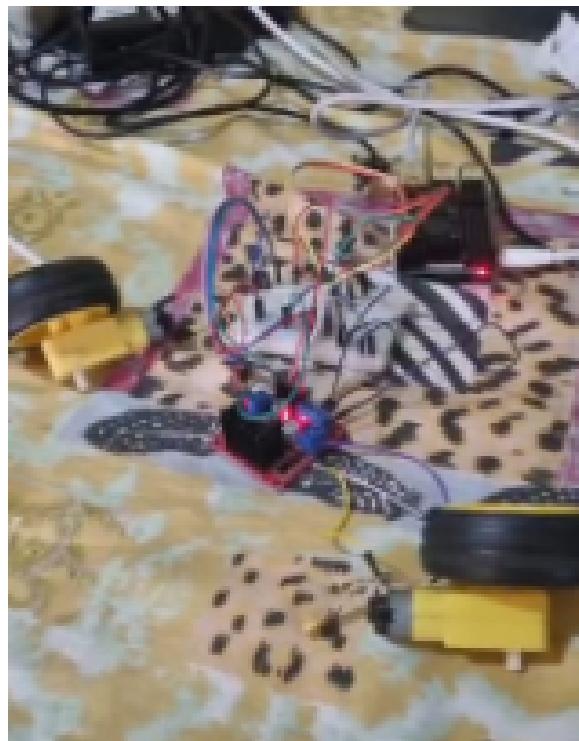
Fuimos a la muestra de la Técnica 7 de Solano y se arma el Stand. Se sigue el código de I2C y ADC.



Se realiza videollamada con CILSA y se realiza nota con TV pública.



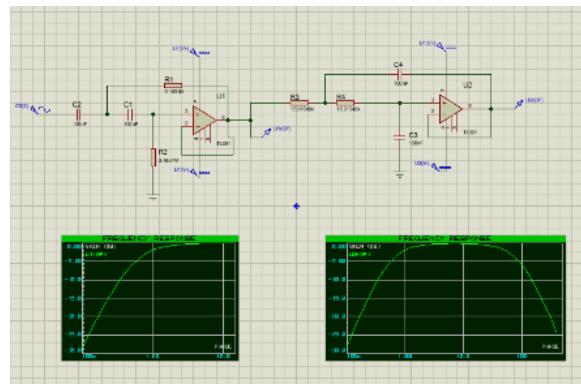
Se prueban los motores Arduino con la Raspberry Pi 4 para el sistema de emergencia.



Avances en el código para la realización de modelo de predicción. Se conectan los códigos para recepción de EEG en tiempo real. Se prueba el motor, se verifica que está quemado y se verifica la funcionalidad.

```
1 import tensorflow as tf
2 import numpy as np
3 from keras.models import Sequential
4 from keras.layers import Dense, Dropout, Activation
5 from keras.utils import to_categorical
6 from keras.preprocessing import image
7 from keras.callbacks import EarlyStopping, ModelCheckpoint # these imports
8 import shutil
9 import re
10
11 def prepare_data(img_data, n, duration, fs, online=True):
12     try:
13         sr = img_data.sr
14         dt = img_data.duration
15         n_samps = int(sr * dt)
16         y = np.zeros(n_samps)
17         x = np.zeros((n_samps, 128)) # 128 = number of filters
18         instant_k = 0
19         instant_k_2 = 0
20         except Exception as e:
21             print("Error preparing data: (%s)" % e)
22             raise
23
24     out_train, out_test = train_test_split(x, y, test_size=0.3, random_state=42)
25
26     return train_test_split(x, y)
```

Simulación de circuito EEG y código para el filtrado digital. Se empieza a realizar la carpeta de campo. Búsqueda de puentes H y módulo PWM. Refactorización de códigos EEG. Prueba de motor.



Se soldó un potenciómetro al motor para luego probarlo. Se revisan los códigos de IA. Se saca la silla con dos motores para probarla.

```
cdigo > ERG > prediction.py > ~
1 import extraction
2 import preprocessing
3 from keras.models import load_model # type: ignore
4 import joblib
5 import reception
6 import pandas as pd
7
8 def load_model_and_scaler():
9     try:
10         model = load_model('wheelchair_model.h5')
11         scaler = joblib.load('scaler.save')
12         return model, scaler
13     except Exception as e:
14         print(f"Error loading model and scaler: {e}")
15         raise
16
17 def classify_ecg(model, scaler, eeg_data, n, duration, fs):
18     try:
19         t, alpha, beta, gamma, delta, theta = preprocessing.preprocess_signal(eeg_data, n, duration, fs)
20         features = extraction.extract_features(alpha, beta, gamma, delta, theta)
21         features_df = pd.DataFrame([features])
22         features_scaled = scaler.transform(features_df)
23         prediction = model.predict(features_scaled)
24         predicted_class = prediction.argmax(axis=1)
```

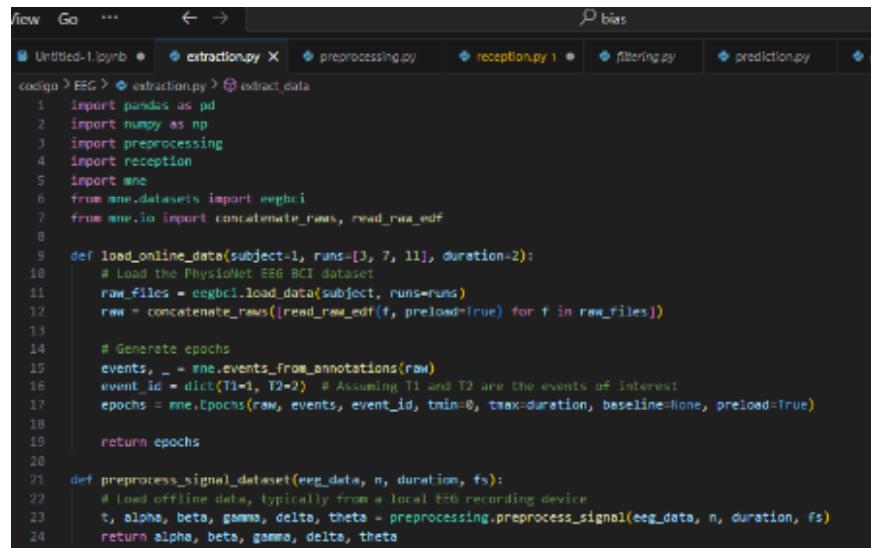
Se empieza a hacer funcionar al motor. El driver aun no funciona pero se detectó un código de estado del error. También se verifica el estado del freno de mano, el cual está cortocircuitado lo que produjo olor a quemado.



Se debuggea el código de filtrado.

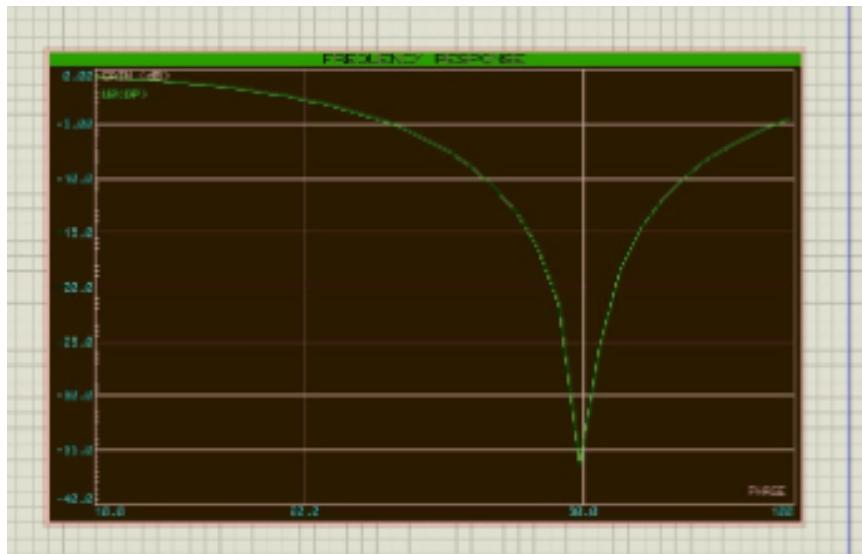
```
codigo > EEG > filtering.py > main
 1 ~ import numpy as np
 2   from scipy.signal import butter, filtfilt
 3   from sklearn.decomposition import FastICA
 4   import preprocessing
 5   import pandas as pd
 6   import matplotlib.pyplot as plt
 7   import graphing # type: ignore
 8
 9 ~ def digital_filtering(eeg_data, fs, notch, bandpass, car=False, ica=False):
10 ~     try:
11         # Handle Nan and infinite values
12         eeg_data = preprocess_data(eeg_data)
13
14         # Print data shape
15         print(f"Original data shape: {eeg_data.shape}")
16
17         # Check the dimentions of the eeg_data
18         if eeg_data.ndim == 1:
19             eeg_data = eeg_data.reshape(1, -1)
20
21         if notch:
22             # Remove power line noise
23             eeg_data = butter_notch_filter(eeg_data, notch_freq=50, fs=fs)
24             print(f"Data shape after notch filter: {eeg_data.shape}")
```

Avances en el debugging de extraction.py debido a inteligencia artificial utilizando datasets de la web. Descubrimos que lo más probable de hacer es realizar un dataset propio.

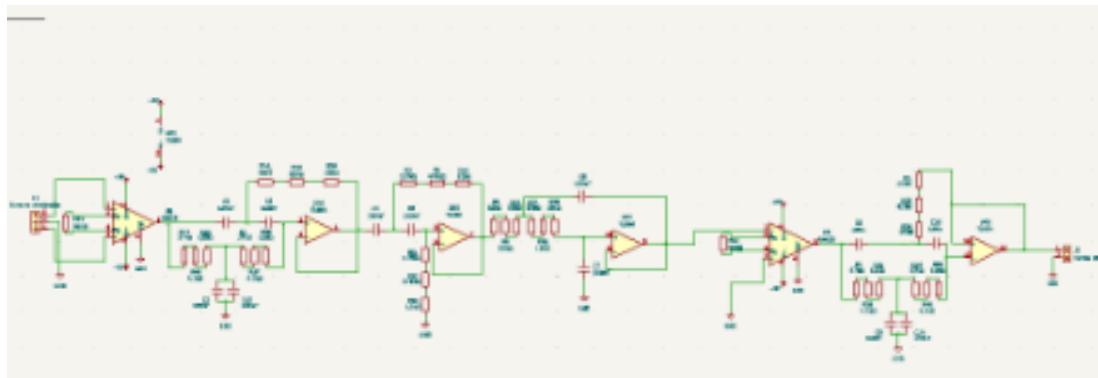


```
View Go ... ← → ⌂ bias
Untitled-1.lynk • extraction.py X preprocessing.py reception.py 1 filtering.py prediction.py ⌂ o
codigo > EEG > extraction.py > extract_data
1 import pandas as pd
2 import numpy as np
3 import preprocessing
4 import reception
5 import mne
6 from mne.datasets import eegbci
7 from mne.io import concatenate_raws, read_raw_edf
8
9 def load_online_data(subject=1, runs=[3, 7, 11], duration=2):
10     # Load the PhysioNet EEG BCI dataset
11     raw_files = eegbci.load_data(subject, runs=runs)
12     raw = concatenate_raws([read_raw_edf(f, preload=True) for f in raw_files])
13
14     # Generate epochs
15     events, _ = mne.events_from_annotations(raw)
16     event_id = dict(T1=1, T2=2) # Assuming T1 and T2 are the events of interest
17     epochs = mne.Epochs(raw, events, event_id, tmin=0, tmax=duration, baseline=None, preload=True)
18
19     return epochs
20
21 def preprocess_signal_dataset(eeg_data, n, duration, fs):
22     # Load offline data, typically from a local EEG recording device
23     t, alpha, beta, gamma, delta, theta = preprocessing.preprocess_signal(eeg_data, n, duration, fs)
24
25     return alpha, beta, gamma, delta, theta
```

Búsqueda de modelos de puente H. Simulación de filtrado de circuitos y rediseño de los filtros. Se empieza a realizar el diseño de KiCad.



Llegan algunos materiales de cooperadora y continuamos el diseño en Kicad. Se intenta probar la recepción por medio de I2C y la Raspberry Pi Pico con la Pi 4

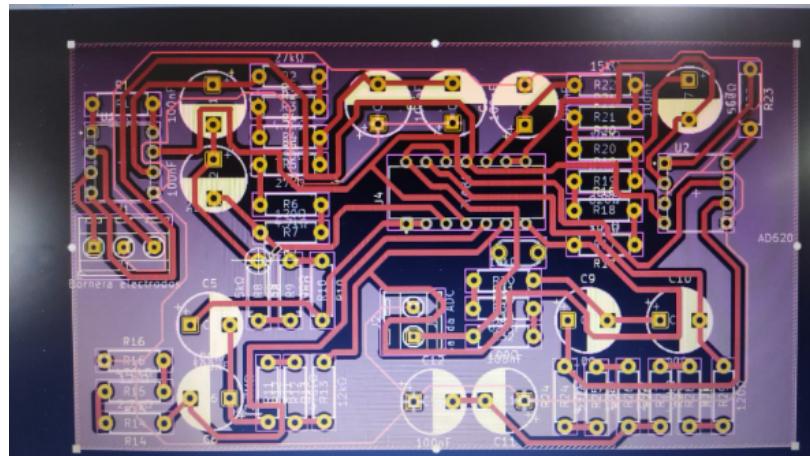


En el día de la fecha se ordenaron los materiales ordenados por cooperadora y se pidieron aquellos faltantes en el pañol y lo que no había se agregó a la lista de compra. También hicimos pruebas con los motores y logramos hacerlos funcionar, además de ponerlos en condiciones para su uso. Recibimos los electrodos del proyecto Brainstream del año pasado.

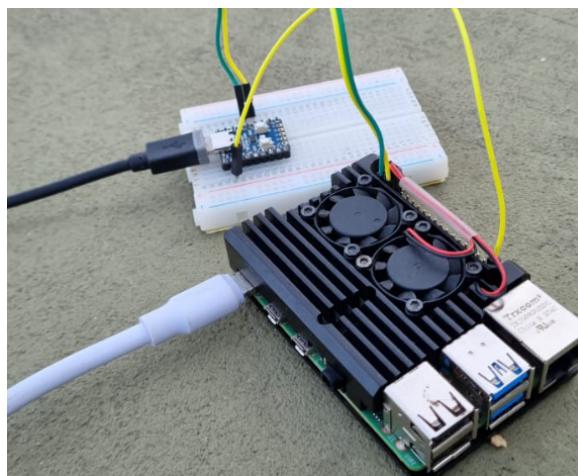


5. Julio de 2024

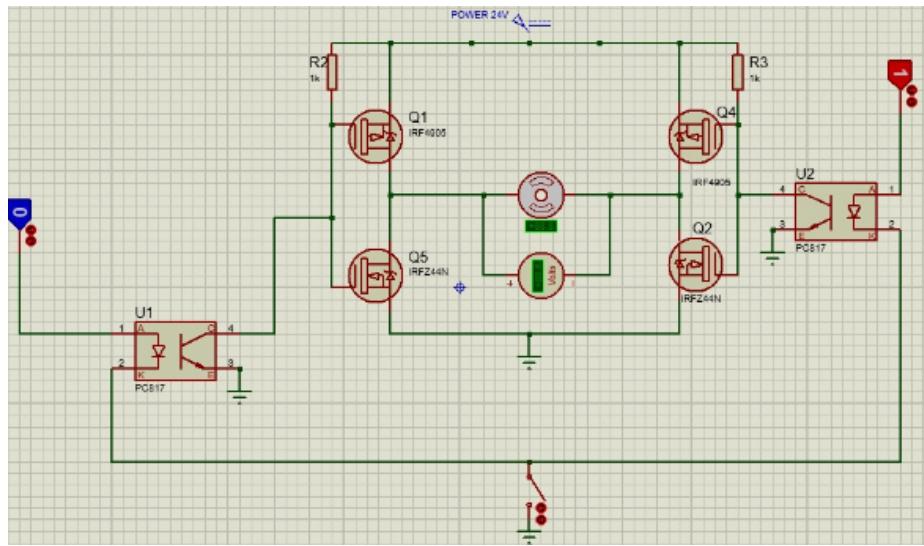
Se empieza a realizar el circuito en el PCB. Se empieza a debuggear el código de recepción, aunque aún con inconvenientes en la conexión.



Se encuentra un Puente H que supuestamente aguanta hasta 48 A. Se finaliza el pcb del circuito EEG. Se establece la conexión I2C entre los micros. Se prueba el circuito de filtro pasabajos. Se busca alternativa al Driver Arduino por transistores que aguanten la corriente y hacerlo por nosotros mismos.



Se realiza la conexión I2C entre la Pi 4 y la Pi Pico. Se van refactorizando los códigos y mejorando el rendimiento de la conexión para mejorar la rapidez. Por otro lado, ya se hizo el pedido de los transistores Mosfet que soporten la corriente de 24 A para realizar el puente H.



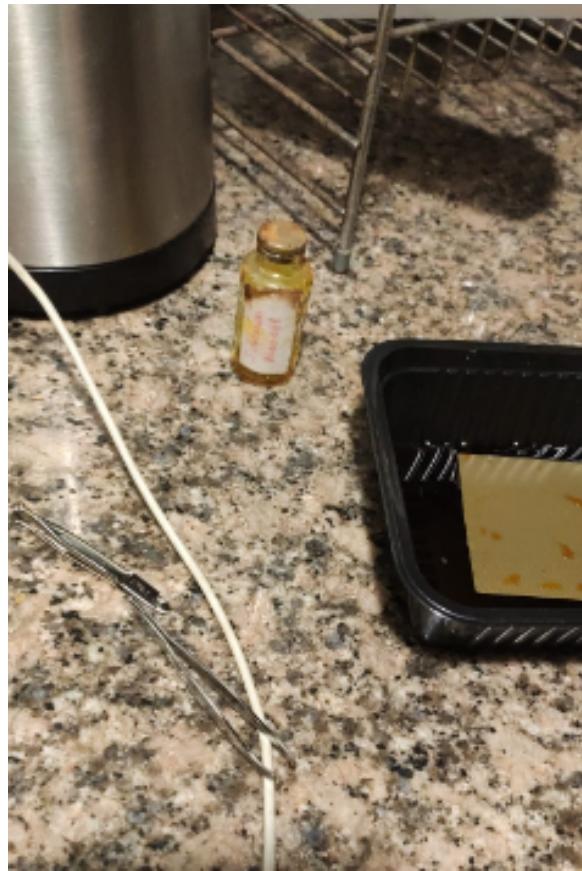
Se comenzó a realizar la carpeta técnica, tanto en word como en LaTeX. Conexión entre i2c y los microcontroladores, mejorando la velocidad para las mediciones y refactorizando el código. Creación de modelo 3D del PCB y finalización del EEG. Realización de puente H por señales PWM.

```

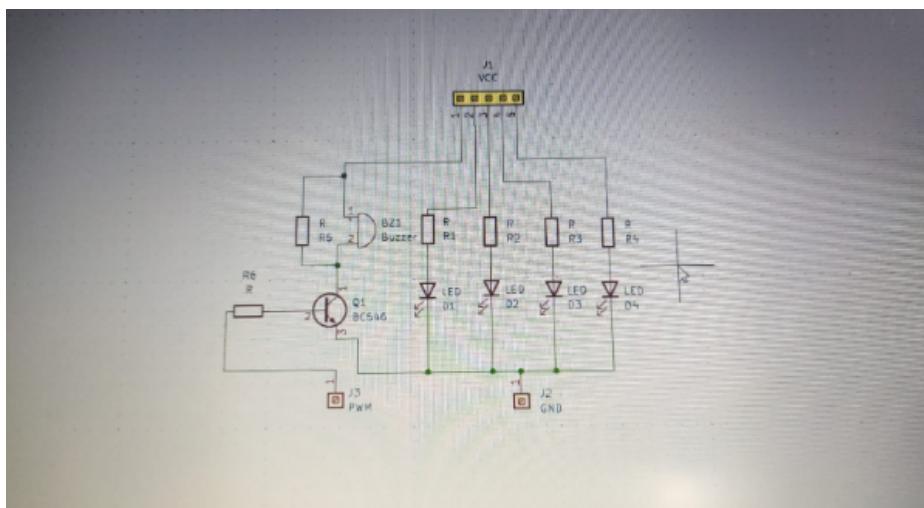
1 // Main code for the project
2 #include "hardware/adc.h"
3 #include "hardware/i2c.h"
4 #include <stdio.h>
5
6 #define I2C_ADDR 0x3E
7 #define ADC_DELAY_US 1000
8 #define I2C_PORT i2c0
9 #define NUMBER_OF_BYTES_TO_READ 3
10 #define NUMBER_OF_CHANNELS 4
11 #define NUMBER_OF_BYTES_TO_SEND 8
12
13 // Function prototypes
14 bool read_adc();
15 void init_adc(uint8_t adc_channel_0, uint8_t
16 void init_i2c(uint8_t sda_pin, uint8_t scl_pi
17
18 // Array to store ADC values
19 uint16_t adc_values[NUMBER_OF_CHANNELS];
20
21 int main(void) {
22     // Set baudrate
23     const int BAUDRATE = 100 * 1000;

```

Se empieza a hacer una placa modelo para el EEG la cual luego se enviará a China para la posterior confección. Se sigue intentando con la conexión i2c, primero probándola en una Pico sola y luego la comunicación entre micros; sin embargo, siguen habiendo errores y retrasos.



Se modifica el protocolo a utilizar de i2c a UART debido a una mayor simplicidad en la comunicación y menos problemas. Además, por otra parte se investigan los disipadores a utilizar en el puente H. Se confecciona finalmente el sistema de emergencia para realizar el PCB. Se descubre que se puede simular PWM por software.



6. Agosto de 2024

Se comienza a realizar el codigo para mover los motores mediante PWM. Se toma las medidas del cráneo de Nicolas Fabian Adell para crear la vincha/casco para colocar los electrodos. Se soluciona el problema del refresco de las muestras. Se sigue realizando la carpeta técnica. Se comenzó a grabar un video para reels.

```
from gpiozero import PWMOutputDevice, DigitalOutputDevice
import time

# GPIO Pin setup for Motor 1
motor1_in1 = PWMOutputDevice(12) # Use PWM-capable pin
motor1_in2 = PWMOutputDevice(13) # Use PWM-capable pin

# GPIO Pin setup for Motor 2
motor2_in1 = PWMOutputDevice(18) # Use PWM-capable pin
motor2_in2 = PWMOutputDevice(19) # Use PWM-capable pin

def set_motor_speed(motor1_in1, motor1_in2, speed):
    if speed > 0:
        motor1_in1.value = speed / 100.0
        motor1_in2.value = 0
    elif speed < 0:
        motor1_in1.value = 0
        motor1_in2.value = abs(speed) / 100.0
    else:
        motor1_in1.value = 0
        motor1_in2.value = 0

def move_forward(speed):
    set_motor_speed(motor1_in1, motor1_in2, speed)
    set_motor_speed(motor2_in1, motor2_in2, speed)

def move_backward(speed):
    set_motor_speed(motor1_in1, motor1_in2, -speed)
    set_motor_speed(motor2_in1, motor2_in2, -speed)
```

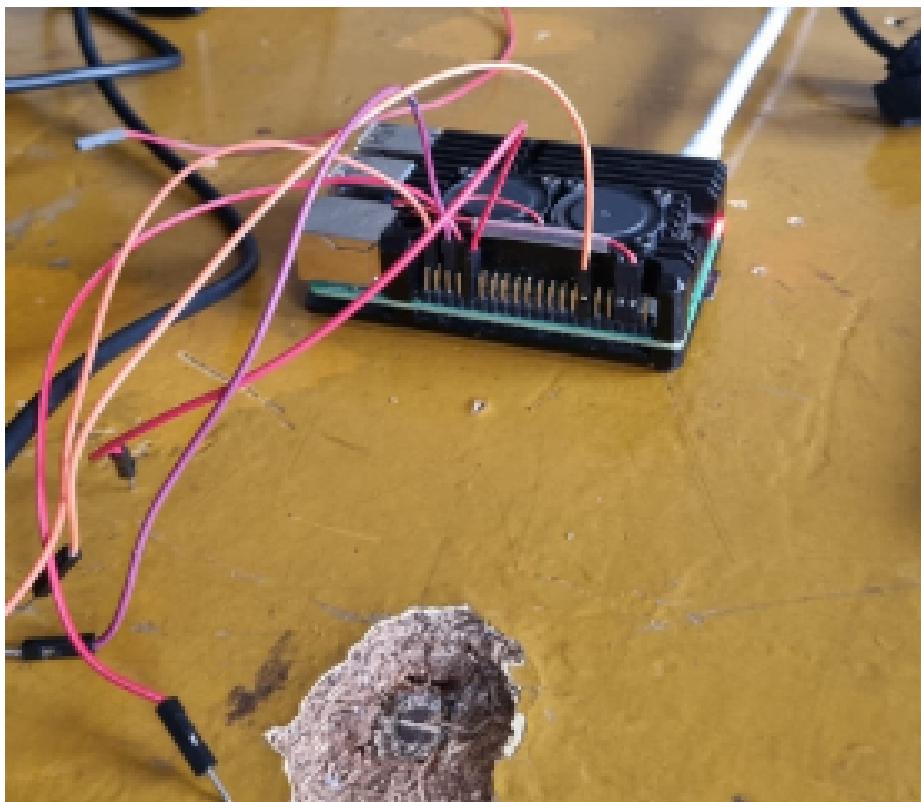
01, 020 : [200, 0, 422, 602], '021': [1447, 217, 2307, 2507], '022': [1887, 131, 2117, 2007], '023': [1607, 221, 201, 2077], '024': [1607, 160, 201, 1991], '025': [1607, 384, 160, 397], '026': [1607, 1449, 1711], '027': [1893, 317, 1948, 2386], '028': [1996, 3621, 2621, 2311], '029': [1481, 1321, 1361, 1717], '030': [1886, 3124, 1958, 2183], '031': [1992, 3689, 2921, 2318], '032': [1383, 1316, 1362, 1718], '033': [1484, 10, 238, 189], '034': [1266, 10, 238, 189], '035': [1996, 3139, 313, 257], '036': [220, 139, 313, 257], '037': [553, 8, 469, 632], '038': [315, 44, 318, 257], '039': [1885, 10, 238, 189], '040': [220, 139, 313, 257], '041': [388, 231, 386, 390], '042': [1555, 671, 566, 499], '043': [1483, 3006, 1447, 1711], '044': [1888, 3124, 1958, 2183], '045': [1887, 218, 2304, 338], '046': [1393, 1316, 1362, 1718], '047': [546, 8, 468, 634], '048': [375, 45, 323, 362], '049': [27, 22, 241, 195], '050': [218, 145, 213, 261], '051': [293, 218, 294, 338], '052': [1401, 3071, 1448, 1274], '053': [1486, 3111, 1951, 7186], '054': [1988, 3626, 2618, 2311], '055': [1399, 1313, 1367, 1718], '056': [1547, 8, 471, 632], '057': [378, 36, 322, 358], '058': [264, 13, 246, 198], '059': [337, 149, 313, 2], '060': [337, 149, 313, 260], '061': [367, 225, 391, 391], '062': [1554, 662, 570, 500], '063': [1401, 3071, 1448, 1274], '064': [1486, 3111, 1951, 7186], '065': [1988, 3626, 2618, 2311], '066': [1399, 1313, 1367, 1718], '067': [545, 8, 469, 633], '068': [372, 36, 324, 359], '069': [273, 26, 237, 195], '070': [337, 149, 313, 2], '071': [337, 149, 313, 260], '072': [1554, 671, 565, 586], '073': [1402, 3066, 1447, 1275], '074': [1486, 3024, 2018, 2310], '075': [1487, 3086, 1368, 1718], '076': [549, 8, 468, 635], '077': [358, 44, 331, 362], '078': [263, 39, 294, 195], '079': [317, 159, 269, 265], '080': [1885, 3124, 1958, 2183], '081': [1988, 3626, 2618, 2311], '082': [1399, 1313, 1367, 1718], '083': [1401, 3071, 1448, 1274], '084': [1486, 3111, 1951, 7186], '085': [1988, 3626, 2618, 2311], '086': [1399, 1313, 1367, 1718], '087': [546, 8, 468, 634], '088': [375, 45, 323, 362], '089': [273, 26, 237, 195], '090': [337, 149, 313, 2], '091': [337, 149, 313, 260], '092': [1554, 671, 565, 586], '093': [1402, 3066, 1447, 1275], '094': [1486, 3024, 2018, 2310], '095': [1487, 3086, 1368, 1718], '096': [549, 8, 469, 633], '097': [358, 44, 331, 362], '098': [263, 39, 294, 195], '099': [317, 159, 269, 265], '010': [1885, 3124, 1958, 2183], '011': [1988, 3626, 2618, 2311], '012': [1399, 1313, 1367, 1718], '013': [1401, 3071, 1448, 1274], '014': [1486, 3111, 1951, 7186], '015': [1988, 3626, 2618, 2311], '016': [1399, 1313, 1367, 1718], '017': [546, 8, 468, 634], '018': [375, 45, 323, 362], '019': [273, 26, 237, 195], '020': [337, 149, 313, 2], '021': [337, 149, 313, 260], '022': [1554, 671, 562, 586], '023': [1402, 3024, 2018, 2310], '024': [1487, 3086, 1368, 1718], '025': [549, 8, 468, 635], '026': [358, 44, 331, 362], '027': [263, 39, 294, 195], '028': [317, 159, 269, 265], '029': [1885, 3124, 1958, 2183], '030': [1988, 3626, 2618, 2311], '031': [1399, 1313, 1367, 1718], '032': [1401, 3071, 1448, 1274], '033': [1486, 3111, 1951, 7186], '034': [1988, 3626, 2618, 2311], '035': [1399, 1313, 1367, 1718], '036': [546, 8, 469, 634], '037': [375, 45, 323, 362], '038': [273, 26, 237, 195], '039': [337, 149, 313, 2], '040': [337, 149, 313, 260], '041': [1554, 671, 563, 586], '042': [1402, 3066, 1447, 1275], '043': [1486, 3024, 2018, 2310], '044': [1487, 3086, 1368, 1718], '045': [549, 8, 468, 633], '046': [358, 44, 331, 362], '047': [263, 39, 294, 195], '048': [317, 159, 269, 265], '049': [1885, 3124, 1958, 2183], '050': [1988, 3626, 2618, 2311], '051': [1399, 1313, 1367, 1718], '052': [1401, 3071, 1448, 1274], '053': [1486, 3111, 1951, 7186], '054': [1988, 3626, 2618, 2311], '055': [1399, 1313, 1367, 1718], '056': [546, 8, 469, 632], '057': [375, 45, 323, 361], '058': [273, 26, 237, 194], '059': [337, 149, 313, 2], '060': [337, 149, 313, 260], '061': [1554, 671, 562, 585], '062': [1402, 3066, 1447, 1275], '063': [1486, 3024, 2018, 2310], '064': [1487, 3086, 1368, 1718], '065': [549, 8, 468, 631], '066': [358, 44, 331, 360], '067': [263, 39, 294, 193], '068': [317, 159, 269, 264], '069': [1885, 3124, 1958, 2183], '070': [1988, 3626, 2618, 2311], '071': [1399, 1313, 1367, 1718], '072': [1401, 3071, 1448, 1274], '073': [1486, 3111, 1951, 7186], '074': [1988, 3626, 2618, 2311], '075': [1399, 1313, 1367, 1718], '076': [546, 8, 469, 630], '077': [375, 45, 323, 360], '078': [273, 26, 237, 193], '079': [337, 149, 313, 2], '080': [337, 149, 313, 260], '081': [1554, 671, 561, 584], '082': [1402, 3066, 1447, 1275], '083': [1486, 3024, 2018, 2310], '084': [1487, 3086, 1368, 1718], '085': [549, 8, 468, 629], '086': [358, 44, 331, 359], '087': [263, 39, 294, 192], '088': [317, 159, 269, 263], '089': [1885, 3124, 1958, 2183], '090': [1988, 3626, 2618, 2311], '091': [1399, 1313, 1367, 1718], '092': [1401, 3071, 1448, 1274], '093': [1486, 3111, 1951, 7186], '094': [1988, 3626, 2618, 2311], '095': [1399, 1313, 1367, 1718], '096': [546, 8, 469, 628], '097': [375, 45, 323, 358], '098': [273, 26, 237, 192], '099': [337, 149, 313, 2], '100': [337, 149, 313, 260], '101': [1554, 671, 560, 583], '012': [1402, 3066, 1447, 1275], '013': [1486, 3024, 2018, 2310], '014': [1487, 3086, 1368, 1718], '015': [549, 8, 468, 627], '016': [358, 44, 331, 357], '017': [263, 39, 294, 191], '018': [317, 159, 269, 262], '019': [1885, 3124, 1958, 2183], '020': [1988, 3626, 2618, 2311], '021': [1399, 1313, 1367, 1718], '022': [1401, 3071, 1448, 1274], '023': [1486, 3111, 1951, 7186], '024': [1988, 3626, 2618, 2311], '025': [1399, 1313, 1367, 1718], '026': [546, 8, 469, 626], '027': [375, 45, 323, 356], '028': [273, 26, 237, 191], '029': [337, 149, 313, 2], '030': [337, 149, 313, 260], '031': [1554, 671, 559, 582], '032': [1402, 3066, 1447, 1275], '033': [1486, 3024, 2018, 2310], '034': [1487, 3086, 1368, 1718], '035': [549, 8, 468, 625], '036': [358, 44, 331, 355], '037': [263, 39, 294, 190], '038': [317, 159, 269, 261], '039': [1885, 3124, 1958, 2183], '040': [1988, 3626, 2618, 2311], '041': [1399, 1313, 1367, 1718], '042': [1401, 3071, 1448, 1274], '043': [1486, 3111, 1951, 7186], '044': [1988, 3626, 2618, 2311], '045': [1399, 1313, 1367, 1718], '046': [546, 8, 469, 624], '047': [375, 45, 323, 354], '048': [273, 26, 237, 190], '049': [337, 149, 313, 2], '050': [337, 149, 313, 260], '051': [1554, 671, 558, 581], '052': [1402, 3066, 1447, 1275], '053': [1486, 3024, 2018, 2310], '054': [1487, 3086, 1368, 1718], '055': [549, 8, 468, 623], '056': [358, 44, 331, 353], '057': [263, 39, 294, 189], '058': [317, 159, 269, 260], '059': [1885, 3124, 1958, 2183], '060': [1988, 3626, 2618, 2311], '061': [1399, 1313, 1367, 1718], '062': [1401, 3071, 1448, 1274], '063': [1486, 3111, 1951, 7186], '064': [1988, 3626, 2618, 2311], '065': [1399, 1313, 1367, 1718], '066': [546, 8, 469, 622], '067': [375, 45, 323, 352], '068': [273, 26, 237, 189], '069': [337, 149, 313, 2], '070': [337, 149, 313, 260], '071': [1554, 671, 557, 580], '072': [1402, 3066, 1447, 1275], '073': [1486, 3024, 2018, 2310], '074': [1487, 3086, 1368, 1718], '075': [549, 8, 468, 621], '076': [358, 44, 331, 351], '077': [263, 39, 294, 188], '078': [317, 159, 269, 259], '079': [1885, 3124, 1958, 2183], '080': [1988, 3626, 2618, 2311], '081': [1399, 1313, 1367, 1718], '082': [1401, 3071, 1448, 1274], '083': [1486, 3111, 1951, 7186], '084': [1988, 3626, 2618, 2311], '085': [1399, 1313, 1367, 1718], '086': [546, 8, 469, 620], '087': [375, 45, 323, 350], '088': [273, 26, 237, 187], '089': [337, 149, 313, 2], '090': [337, 149, 313, 260], '091': [1554, 671, 556, 579], '092': [1402, 3066, 1447, 1275], '093': [1486, 3024, 2018, 2310], '094': [1487, 3086, 1368, 1718], '095': [549, 8, 468, 619], '096': [358, 44, 331, 349], '097': [263, 39, 294, 186], '098': [317, 159, 269, 258], '099': [1885, 3124, 1958, 2183], '100': [1988, 3626, 2618, 2311], '101': [1399, 1313, 1367, 1718], '102': [1401, 3071, 1448, 1274], '103': [1486, 3111, 1951, 7186], '104': [1988, 3626, 2618, 2311], '105': [1399, 1313, 1367, 1718], '106': [546, 8, 469, 618], '107': [375, 45, 323, 348], '108': [273, 26, 237, 185], '109': [337, 149, 313, 2], '110': [337, 149, 313, 260], '111': [1554, 671, 555, 577], '112': [1402, 3066, 1447, 1275], '113': [1486, 3024, 2018, 2310], '114': [1487, 3086, 1368, 1718], '115': [549, 8, 468, 617], '116': [358, 44, 331, 347], '117': [263, 39, 294, 184], '118': [317, 159, 269, 257], '119': [1885, 3124, 1958, 2183], '120': [1988, 3626, 2618, 2311], '121': [1399, 1313, 1367, 1718], '122': [1401, 3071, 1448, 1274], '123': [1486, 3111, 1951, 7186], '124': [1988, 3626, 2618, 2311], '125': [1399, 1313, 1367, 1718], '126': [546, 8, 469, 616], '127': [375, 45, 323, 346], '128': [273, 26, 237, 183], '129': [337, 149, 313, 2], '130': [337, 149, 313, 260], '131': [1554, 671, 554, 575], '132': [1402, 3066, 1447, 1275], '133': [1486, 3024, 2018, 2310], '134': [1487, 3086, 1368, 1718], '135': [549, 8, 468, 615], '136': [358, 44, 331, 345], '137': [263, 39, 294, 182], '138': [317, 159, 269, 256], '139': [1885, 3124, 1958, 2183], '140': [1988, 3626, 2618, 2311], '141': [1399, 1313, 1367, 1718], '142': [1401, 3071, 1448, 1274], '143': [1486, 3111, 1951, 7186], '144': [1988, 3626, 2618, 2311], '145': [1399, 1313, 1367, 1718], '146': [546, 8, 469, 614], '147': [375, 45, 323, 344], '148': [273, 26, 237, 181], '149': [337, 149, 313, 2], '150': [337, 149, 313, 260], '151': [1554, 671, 553, 574], '152': [1402, 3066, 1447, 1275], '153': [1486, 3024, 2018, 2310], '154': [1487, 3086, 1368, 1718], '155': [549, 8, 468, 613], '156': [358, 44, 331, 343], '157': [263, 39, 294, 180], '158': [317, 159, 269, 255], '159': [1885, 3124, 1958, 2183], '160': [1988, 3626, 2618, 2311], '161': [1399, 1313, 1367, 1718], '162': [1401, 3071, 1448, 1274], '163': [1486, 3111, 1951, 7186], '164': [1988, 3626, 2618, 2311], '165': [1399, 1313, 1367, 1718], '166': [546, 8, 469, 612], '167': [375, 45, 323, 342], '168': [273, 26, 237, 179], '169': [337, 149, 313, 2], '170': [337, 149, 313, 260], '171': [1554, 671, 552, 573], '172': [1402, 3066, 1447, 1275], '173': [1486, 3024, 2018, 2310], '174': [1487, 3086, 1368, 1718], '175': [549, 8, 468, 611], '176': [358, 44, 331, 341], '177': [263, 39, 294, 178], '178': [317, 159, 269, 254], '179': [1885, 3124, 1958, 2183], '180': [1988, 3626, 2618, 2311], '181': [1399, 1313, 1367, 1718], '182': [1401, 3071, 1448, 1274], '183': [1486, 3111, 1951, 7186], '184': [1988, 3626, 2618, 2311], '185': [1399, 1313, 1367, 1718], '186': [546, 8, 469, 610], '187': [375, 45, 323, 340], '188': [273, 26, 237, 177], '189': [337, 149, 313, 2], '190': [337, 149, 313, 260], '191': [1554, 671, 551, 572], '192': [1402, 3066, 1447, 1275], '193': [1486, 3024, 2018, 2310], '194': [1487, 3086, 1368, 1718], '195': [549, 8, 468, 609], '196': [358, 44, 331, 339], '197': [263, 39, 294, 176], '198': [317, 159, 269, 253], '199': [1885, 3124, 1958, 2183], '200': [1988, 3626, 2618, 2311], '201': [1399, 1313, 1367, 1718], '202': [1401, 3071, 1448, 1274], '203': [1486, 3111, 1951, 7186], '204': [1988, 3626, 2618, 2311], '205': [1399, 1313, 1367, 1718], '206': [546, 8, 469, 608], '207': [375, 45, 323, 338], '208': [273, 26, 237, 175], '209': [337, 149, 313, 2], '210': [337, 149, 313, 260], '211': [1554, 671, 550, 571], '212': [1402, 3066, 1447, 1275], '213': [1486, 3024, 2018, 2310], '214': [1487, 3086, 1368, 1718], '215': [549, 8, 468, 607], '216': [358, 44, 331, 337], '217': [263, 39, 294, 174], '218': [317, 159, 269, 252], '219': [1885, 3124, 1958, 2183], '220': [1988, 3626, 2618, 2311], '221': [1399, 1313, 1367, 1718], '222': [1401, 3071, 1448, 1274], '223': [1486, 3111, 1951, 7186], '224': [1988, 3626, 2618, 2311], '225': [1399, 1313, 1367, 1718], '226': [546, 8, 469, 606], '227': [375, 45, 323, 336], '228': [273, 26, 237, 173], '229': [337, 149, 313, 2], '230': [337, 149, 313, 260], '231': [1554, 671, 549, 570], '232': [1402, 3066, 1447, 1275], '233': [1486, 3024, 2018, 2310], '234': [1487, 3086, 1368, 1718], '235': [549, 8, 468, 605], '236': [358, 44, 331, 335], '237': [263, 39, 294, 172], '238': [317, 159, 269, 251], '239': [1885, 3124, 1958, 2183], '240': [1988, 3626, 2618, 2311], '241': [1399, 1313, 1367, 1718], '242': [

Se empieza a corregir los otros códigos para adaptar esta nueva forma de lectura al resto de código. Se usa un template en LaTex para la documentación.

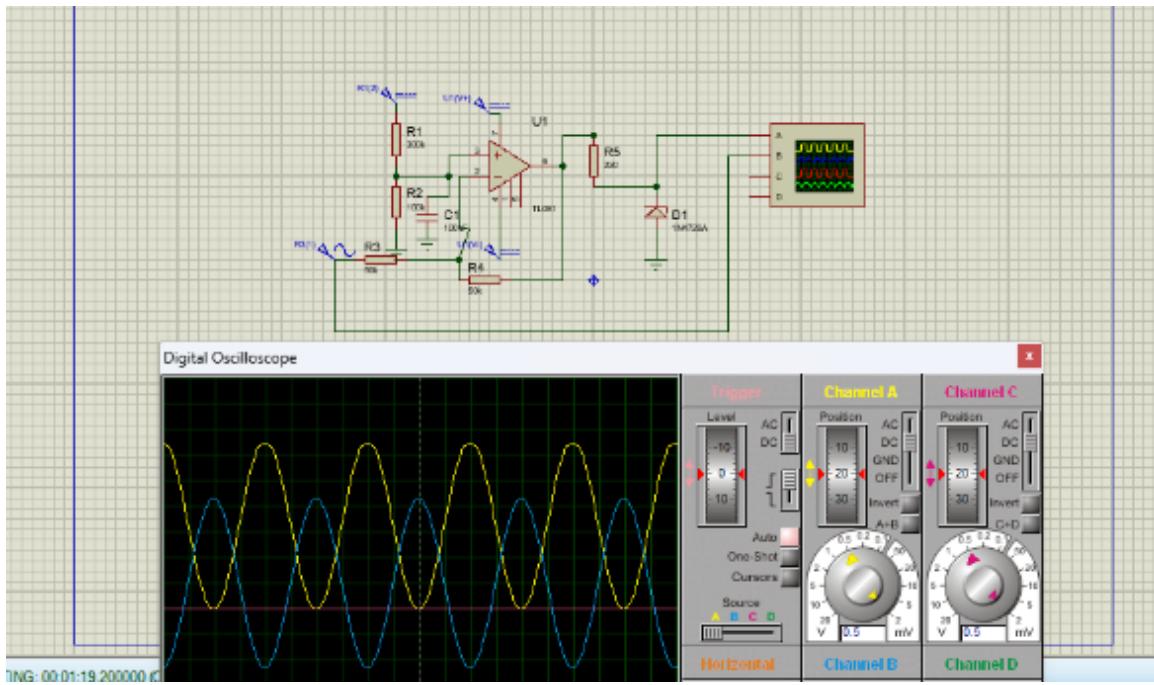
```
1 \documentclass[article]{}
2 \usepackage[utf8]{inputenc}
3 \usepackage[spanish]{babel}
4 \usepackage{graphicx}
5 
6 \date{}
7 
8 \begin{document}
9 
10 \begin{center}
11   \includegraphics[width=1\textwidth]{LogoBIAS.png}\\
12 \end{center}
13 
14 \title{BIAS}
15 
16 \vspace{1cm}
17 
18 \textbf{Integrantes del Proyecto:}
19 
20 \begin{itemize}
21   \item \textcolor{red}{Viter Adolfo}, Nicolas Fabian
22   \item \textcolor{red}{Viter De Blas}, Luca
23   \item \textcolor{red}{Viter Diaz Mellion}, Danilo Sebastian
24   \item \textcolor{red}{Viter Gil Soria}, Ian Lucas
25   \item \textcolor{red}{Viter Montenegro}, Luciano Nahuel
26   \item \textcolor{red}{Viter Sojka}, Santiago Alejandro
27 \end{itemize}
28 \end{document}
```

Integrantes del Proyecto:	
■ Adel, Nicolás Fabián	2
■ De Blasi, Luis	2
■ Díaz Melero, Darío Sebastián	2
■ GI Sotelo, Inés Lucía	2
■ Montenegro, Luciano Nahuel	2
■ Suárez, Santiago Alejandro	2
 Índice	
1. Introducción	2
1.1. Contexto	2
1.2. Metodología	2
1.3. Objetivos	2
2. Desarrollo del Proyecto	2
2.1. Metodología	2
2.2. Herramientas	2
2.3. Resultados	2
3. Conclusiones	2
3.1. Limitaciones de Investigación	2
3.2. Limitaciones	2
3.3. Trabajo Futuro	2
4. Referencias	2

Se trabaja con los códigos de recepción y se empieza a modificar el circuito de filtro. Se sube reel a instagram. Se modifica el código para graficar.



Se hace el código de PWM y se añaden dos filtros nuevos para el código de filtrado digital (filtro FIR y filtro IIR). Se comienza a mejorar la IA. Se mejora el pcb del puente H. Se simula circuito de offset.



Investigación de combinación de las señales en el EEG. Soldado de componentes placa de sistema de emergencia y filtro EEG. Se unen los códigos del Puente H y del main del sistema de emergencia. Se encuentran las formas de combinar las señales de los canales en el EEG. Se sigue realizando la carpeta técnica. Se arma el programa app.py el cual va a ser el encargado de conectar todos los códigos. En el mismo ya se implementaron la recepción constante y el procesamiento aplicado con filtros.

```

if choice == '3':
    n = int(input("Enter number of data points: "))
    fs = int(input("Enter sampling frequency: "))
    duration = n / fs
    number_of_channels = int(input("Enter number of channels: "))
    while True:
        signals = reception.get_real_data(channels=number_of_channels, n=n, fs=fs, filter=True)
        for ch, signal in signals.items():
            t = np.arange(len(signals[ch])) / fs
            graphingTerminal.graph_signal_voltage_time(t=t, signal=np.array(signal), title="Signal {}".format(ch))

        t0, alpha0, beta0, gamma0, delta0, theta0 = preprocessing.preprocess_signal(n=n, duration=duration, fs=fs, eeg
        t1, alpha1, beta1, gamma1, delta1, theta1 = preprocessing.preprocess_signal(n=n, duration=duration, fs=fs, eeg
        t2, alpha2, beta2, gamma2, delta2, theta2 = preprocessing.preprocess_signal(n=n, duration=duration, fs=fs, eeg
        t3, alpha3, beta3, gamma3, delta3, theta3 = preprocessing.preprocess_signal(n=n, duration=duration, fs=fs, eeg

    print("EEG data processed")

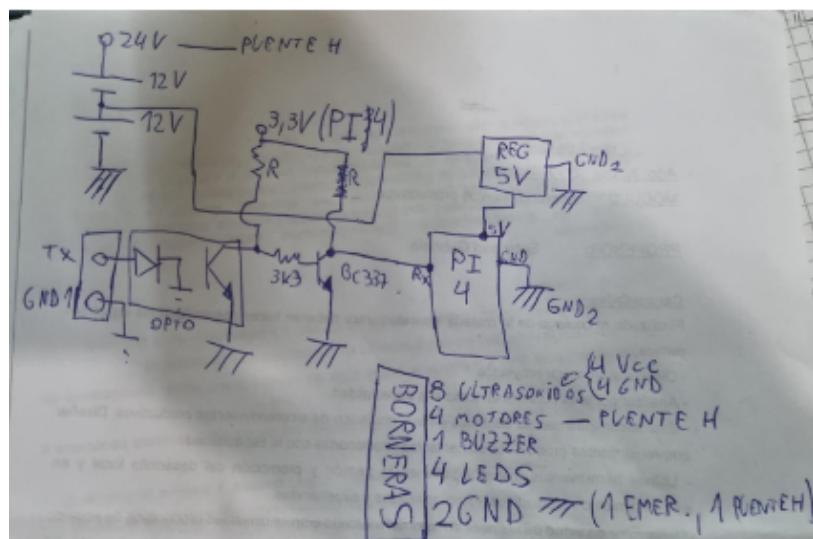
```

Se decide organizar el código en programación orientada a objetos para, de esta manera, tener una mejor organización. De esta manera, se crean clases tales como DspBias, Bias, ReceptionBias y ProcessingBias, FilterBias.



Estuvimos en la feria regional de la UNQ. Se crea clase graphing y se termina diseño de la placa con la RP2040 Zero.

Se crea la clase BiasAI para empezar a codificar. Se adapta el código para que exista un PWM invertido. Se averigua para comprar los AD620AN. Se diseña la unión de las tres partes del circuito para evitar que se filtre ruido y pueda afectar el desempeño de la Pi 4. De esta forma, se tienen dos masas distintas en todo el circuito.



Se prueban el código de los motores. Se sueldan los componentes de los circuitos de EEG y del sistema de emergencia. Se arreglan los errores del código de Emergencia.

Se termina de soldar los componentes de los circuitos. Se decide como hacer la estructura del EEG. Se sigue desarrollando la estructura de la silla. Se investiga sobre las jaulas de Faraday. Se codifica la clase BiasAI.

Se obtiene el dinero para comprar amplificadores AD620AN. Se siguen soldando las plaquetas para luego leer señales cerebrales.

7. Septiembre de 2024

Se trabaja con la extracción de features para el entrenamiento de la IA, utilizando señales sintetizadas. Se intenta lograr que tengan las mismas dimensiones que la capa de entrada. Se diseña el circuito de alimentaciones para la Pi 4 (etapa de control).

```
def extract_features(self, eeg_data):
    features = []
    # Iterate over each channel in eeg_data
    for ch, signals_per_channel in eeg_data.items():
        channel_features = []
        assert(len(signals_per_channel) == self._number_of_waves_per_channel)
        # Iterate over the signals of each channel
        for band_name, signal_wave in signals_per_channel.items():
            signal_wave = np.array(signal_wave)

            # Statistical Features
            mean = np.mean(signal_wave)
            variance = np.var(signal_wave)
            skewness = skew(signal_wave)
            kurt = kurtosis(signal_wave)
            energy = np.sum(signal_wave ** 2)

            # Frequency Domain Features (Power Spectral Density)
            freqs, psd = welch(signal_wave, fs=self._fs) # Assuming fs = 500 Hz
            band_power = np.sum(psd) # Total power within this band

            # Use scipy.signal.cwt instead of pywt
            scales = np.arange(1, 31)
            coeffs = cwt(signal_wave, morlet, scales)
            wavelet_energy = np.sum(coeffs ** 2)

            # Entropy
            signal_entropy = entropy(np.histogram(signal_wave, bins=10)[0])
            list_of_features = [mean, variance, skewness, kurt, energy, band_power, wavelet_energy, signal_entropy]
            channel_features.append(list_of_features)
    features.append(channel_features)
    return features
```

Se trabaja con los modelos de inteligencia artificial, viendo que ya se puede entrenar y va prediciendo comandos. El modelo utiliza una red neuronal convolucional.

```
14/14 ━━━━━━━━━━ 0s 13ms/step
[[48 1 36 16]
[41 0 39 22]
[49 2 29 23]
[47 0 48 18]]
Training complete.
Original data shape: (1000,)
Data shape after notch filter: (1, 1000)
Data shape after bandpass filter: (1, 1000)
Original data shape: (1000,)
Data shape after notch filter: (1, 1000)
Data shape after bandpass filter: (1, 1000)
Original data shape: (1000,)
Data shape after notch filter: (1, 1000)
Data shape after bandpass filter: (1, 1000)
Original data shape: (1000,)
Data shape after notch filter: (1, 1000)
Data shape after bandpass filter: (1, 1000)
Original data shape: (1000,)
Data shape after notch filter: (1, 1000)
Data shape after bandpass filter: (1, 1000)
/home/bias/Documents/adeLL/bias/codigo/RaspberryPi4/bias_ai.py:202: DeprecationWarning: scipy.signal.cwt is deprecated in SciPy 1.12 and will be removed in SciPy 1.15. We recommend using PyWavelets instead.
    coeffs = cwt(signal_wave, morlet, scales)
1/1 ━━━━━━━━━━ 0s 43ms/step
Predicted Command: right
```

Se sigue realizando la carpeta técnica. Se suelda el primer canal del EEG. Se investiga sobre el multiprocessing y el multithreading con python. Se realizan códigos para el multithreading. Se mide la señal de la salida de la placa de los filtros. Se prueba la corriente del motor para calcular las dimensiones de los capacitores.

```

import threading
import importlib
import numpy as np
import scipy.signal as signal
import matplotlib.pyplot as plt

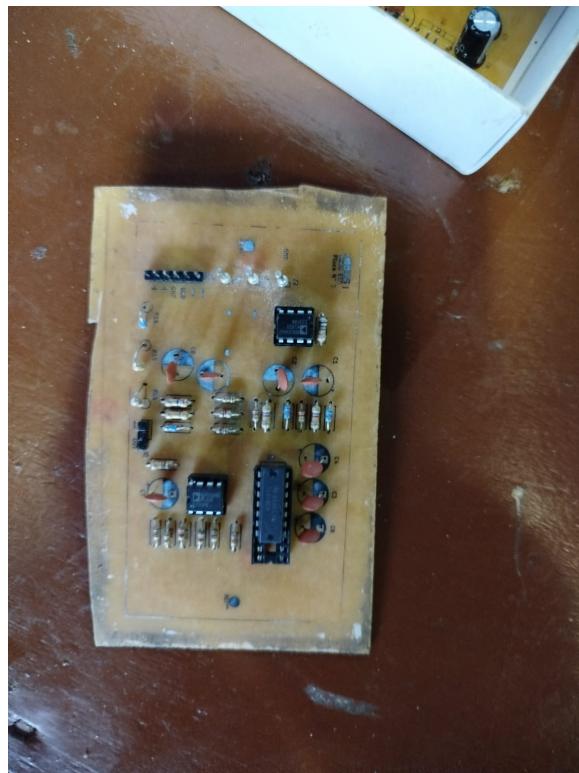
def run_motor_task():
    motor_module = importlib.import_module('motor_task')
    motor_module.motor_task()

def run_processing_task():
    processing_module = importlib.import_module('processing_task')
    # Proporciona señales de ejemplo y otros parámetros necesarios
    n = 512
    fs = 256.0
    eeg_signals = [
        1: [0] * n, # Señal de ejemplo
        2: [0] * n, # Señal de ejemplo
        3: [0] * n, # Señal de ejemplo
        4: [0] * n # Señal de ejemplo
    ]
    times, processed_signals = processing_module.processing_task(eeg_signals, n, fs)
    print("Processing task completed.")

def run_filter_task():
    # Parámetros de ejemplo
    fs = 256.0 # Frecuencia de muestreo
    f_notch = 50.0 # Frecuencia a eliminar con el filtro Notch
    quality_factor = 30.0 # Factor de calidad para el filtro Notch
    f_low, f_high = 1.0, 50.0 # Frecuencias para el filtro Bandpass

```

Probamos el circuito de filtrado y se encontró un problema. El último notch tenía las entradas inversoras y no inversoras invertidas



Probamos el circuito de filtrado y se solucionó el problema que encontramos el día anterior. Probamos el circuito de emergencia desde el PCB. Se sigue realizando la carpeta técnica. Se encuentra un error en la última etapa de notch, que por alguna razón empieza a amplificar luego de los 100 Hz.

Se avanza con el soldado de la placa de offset. A su vez se termina el diseño del puente H para imprimir y finalmente se descarga el dataset BCI IV 2a. para luego utilizarlo para el entrenamiento del modelo.



Se programa un Arduino Nano 3.0 para usar un joystick y mover los motores. Por lo tanto, se crea un programa de slave para la arduino y un programa de master para la Raspberry Pi 4.

```
// Pines para los motores
const int motorIzqIN1 = 3;
const int motorIzqIN2 = 4;
const int motorDerIN3 = 5;
const int motorDerIN4 = 6;

// Pines del joystick
const int joyX = A0; // Eje X
const int joyY = A1; // Eje Y

int xValue, yValue;
int motorSpeed = 255;

void setup() {
    pinMode(motorIzqIN1, OUTPUT);
    pinMode(motorIzqIN2, OUTPUT);
    pinMode(motorDerIN3, OUTPUT);
    pinMode(motorDerIN4, OUTPUT);

    // Inicializar comunicación serial
    Serial.begin(9600);
}

void loop() {
    xValue = analogRead(joyX);
    yValue = analogRead(joyY);
```

Se prueba la placa de emergencia, se ve que hay un error en el buzzer y posteriormente se termina la placa de alimentación para la Pi 4. Se termina de realizar la lista para la compra de materiales. Se descubre que la plaqueta de alimentación contiene el LED rojo al revés. Se terminan de planchar las pistas del puente H y se estañan las mismas.



Se prueba el circuito de offset y se descubre que el diodo Zener práctico no es igual a la simulación ya que en el mismo circula corriente aunque sea menor a la tensión esperada. Se prueba circuito de transmisión por UART y se ve que a la salida no le llega nada a la Pi 4. Se descubre que el problema en la plaqueta de la comunicación UART es la velocidad de transmisión. Se puede transmitir hasta 9600 baudios. Por esa razón, se reemplaza por un optoacoplador 6n137 y por un transistor 2n2222.

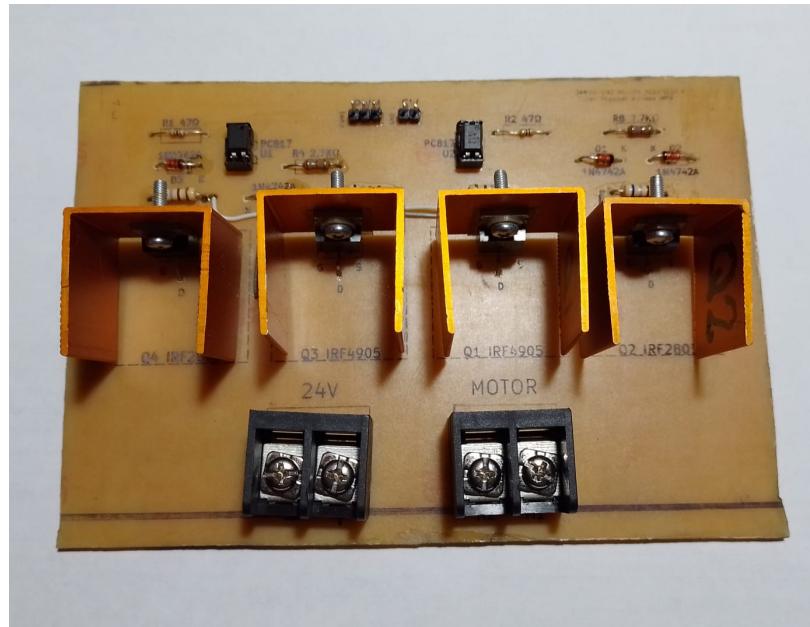


Se va a hablar con una técnica neuróloga en el hospital del cruce con el objetivo de que nos muestre las instalaciones EEG y nos despeje algunas dudas de la localización de los electrodos..

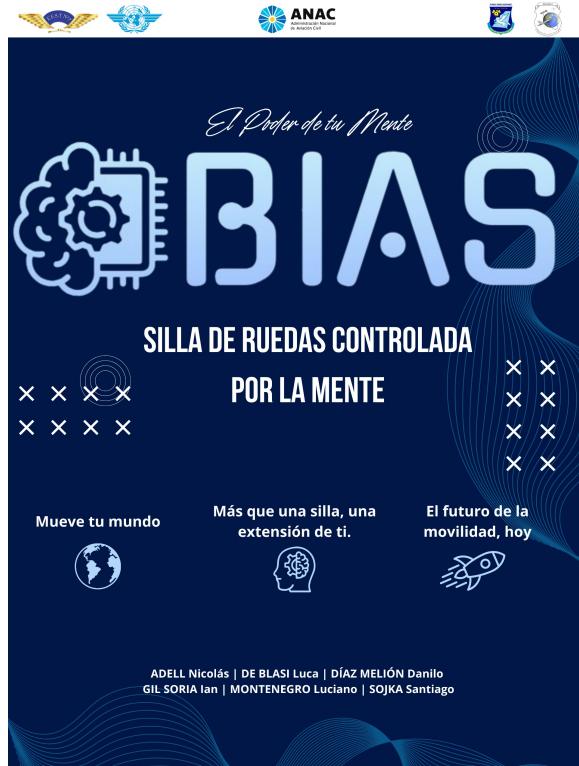
Se van a comprar los componentes restantes y se resuelve el problema de la velocidad de comunicación UART mediante el reemplazo de los materiales nombrados anteriormente. Se rediseñan las placas por tema de alimentaciones y, por último, se rediseña la placa de recepción. Además, se

sueldan los últimos componentes del puente H. Por otro lado, se piensa en hacer una silla de diseño 3D chiquita.

Se prueba el puente H y nos damos cuenta de un error, y es que circulan 160 A cuando ambas entradas PWM están en cero. Se prueba el circuito de filtrado en protoboard pero termina un AD620AN quemado debido a que las alimentaciones estaban invertidas. Por otro lado, se realiza la placa de control de la Pi 4.



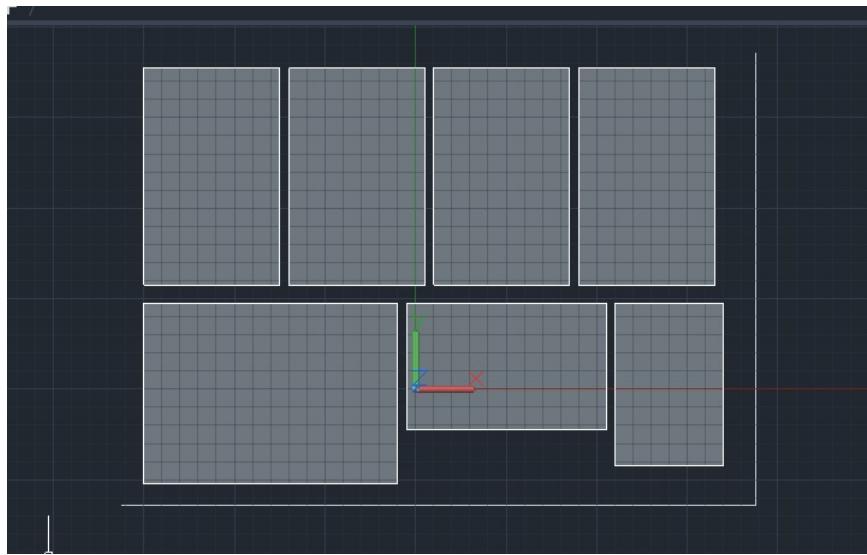
Se arreglan ciertas incoherencias en la “shape” del dataset del código de la IA. Se realiza el banner y se manda a cooperadora.



Se prueba el circuito de filtrado sin el último filtro Notch. Este último distorsionaba la señal al punto de aumentarla hasta los 120 Hz y disminuye luego de esta frecuencia. Sin este último nos aseguramos que no haya distorsión. Se termina de soldar el circuito de control y recepción de datos. Se prueba el circuito de offset y nos damos cuenta que el zener empieza a conducir antes de que la señal se recorte, por lo que sugerimos cambiar la resistencia. De esta forma, la tensión de la resistencia será menor.

8. Octubre de 2024

Se comienza a diseñar la jaula de Faraday. Se miden los PCBs. La silla de madera termina de ser fabricada.



Se comienza a realizar el Informe Técnico del proyecto para presentar en las ONIET.

BIAS Informe Descriptivo

Code Editor Visual Editor Review Share Submit History Layout Ch

```
1 \documentclass[article]
2 \usepackage{RawInputEncoding}
3 \usepackage[utf8]{inputenc}
4 \usepackage[T1]{fontenc}
5 \usepackage{selinput}
6 \usepackage[spanish]{babel}
7 \usepackage{hyphenat}
8 \usepackage{graphicx}
9 \usepackage{subcaption}
10 \usepackage{hyperref}
11 \usepackage{listings}
12 \usepackage{xcolor}
13 \usepackage{float}
14 \usepackage[a4paper, total={6in, 9in}]{geometry}
15 \begin{document}
16 \date{}
17
18 \begin{center}
19   \includegraphics[width=1\textwidth]
20   {Imagenes/Logo/LogoBIASbn.png}\\
21 \end{center}
22 \title{}
23
24 \vspace{1cm}
25
26 \textbf{Integrantes del Proyecto:}
27
28 \begin{itemize}
```

BIAS

Integrantes del Proyecto:

- Adel, Nicolas Fabian (DNI 47069172)
- De Biasi, Luca (DNI 47382528)
- Dian Melisa, Daniel Sebastian (DNI 47307165)
- Gil Soria, Ian Lucas (DNI 47371514)
- Montenegro, Luciano Nahuel (DNI 47256543)
- Seijas, Santiago Alejandro (DNI 47306411)

Se probó el circuito de filtrado con los electrodos, usando la cabeza de Santiago Sojka. Se sigue realizando la carpeta técnica. Se buscan soluciones a los errores del puente H.

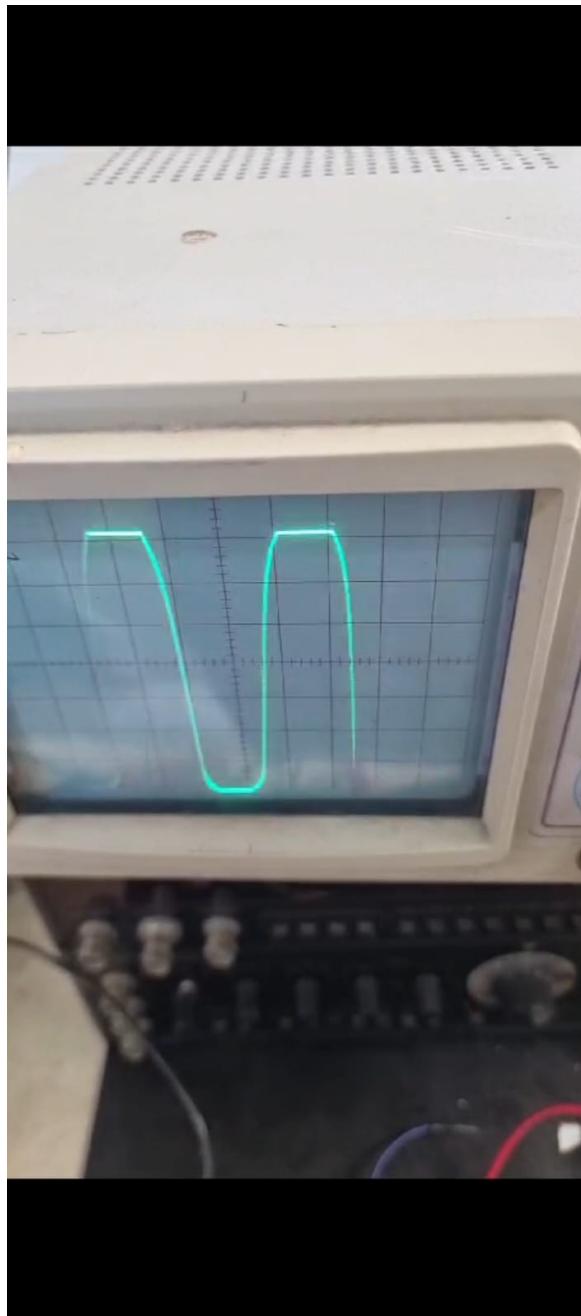


Se prueba nuevamente la placa de filtrado con osciloscopio digital. Se prepara todo para entregar el informe descriptivo y el informe de redes sociales para las ONIET.

Se diseña un nuevo circuito de puente H que tiene más probabilidades de funcionar. Se termina y luego se refactoriza el programa que permite entrenar al algoritmo de machine learning.

Se hace funcionar el puente H con un nuevo circuito. En este caso, lo que hicimos fue hacer un cruce entre los gates de los mosfets y, de esta forma, vimos que la carga podía tener +24 V, -24 V y 0 V. Por último, se intentó probar todas las etapas de filtros y nos dimos cuenta que las baterías estaban descargadas.

Se prueban todas las placas juntas y nos damos cuenta que hay un error en una placa debido a que había una pata de resistencia haciendo cortocircuito con masa. Después, se empieza a hacer la jaula de Faraday de madera. Por último, se terminan todas las placas de filtros y se empiezan a probarlas. Se prueban todas las placas juntas para ver la gráfica de señales en la terminal con los 9 electrodos juntos.





Se termina de realizar la placa del puente H. Se empieza a diagramar las conexiones. Se avanza en realizar la carpeta técnica. Se ve que el modelo de inteligencia artificial posee un 25 % de precisión.

```

53/53      - 2s 36ms/step - accuracy: 0.9710 - loss: 0.1305 - val_accuracy: 0.3389 - val_loss: 3.7916
Epoch 91/100
53/53      - 2s 36ms/step - accuracy: 0.9677 - loss: 0.1371 - val_accuracy: 0.3270 - val_loss: 3.8902
Epoch 92/100
53/53      - 2s 36ms/step - accuracy: 0.9582 - loss: 0.1652 - val_accuracy: 0.3317 - val_loss: 3.7651
Epoch 93/100
53/53      - 2s 36ms/step - accuracy: 0.9735 - loss: 0.1258 - val_accuracy: 0.3389 - val_loss: 3.9592
Epoch 94/100
53/53      - 2s 36ms/step - accuracy: 0.9585 - loss: 0.1499 - val_accuracy: 0.3317 - val_loss: 4.2479
Epoch 95/100
53/53      - 2s 36ms/step - accuracy: 0.9738 - loss: 0.1015 - val_accuracy: 0.3365 - val_loss: 4.4690
Epoch 96/100
53/53      - 2s 36ms/step - accuracy: 0.9745 - loss: 0.1076 - val_accuracy: 0.3270 - val_loss: 4.4815
Epoch 97/100
53/53      - 2s 36ms/step - accuracy: 0.9671 - loss: 0.1073 - val_accuracy: 0.3246 - val_loss: 4.3911
Epoch 98/100
53/53      - 2s 36ms/step - accuracy: 0.9706 - loss: 0.1065 - val_accuracy: 0.3150 - val_loss: 4.4487
Epoch 99/100
53/53      - 2s 36ms/step - accuracy: 0.9643 - loss: 0.1192 - val_accuracy: 0.3126 - val_loss: 4.4732
Epoch 100/100
53/53      - 2s 36ms/step - accuracy: 0.9681 - loss: 0.1163 - val_accuracy: 0.3413 - val_loss: 4.5056
14/14      - 0s 7ms/step - accuracy: 0.3330 - loss: 4.5519
Test Accuracy: 0.3412887752056122
14/14      - 1s 46ms/step
[[33 22 22 24]
 [ 9 35 23 35]
 [20 20 30 33]
 [22 22 24 45]]
Training complete.

```

Lo siguiente que hicimos fue probar el puente H con un motor de poco consumo. Se prueba el modelo con SVM y CSP para ver cómo se comportan antes las señales. Se modifican las dimensiones de las señales y de las features para poder entrenarlas.

Lo que se prueba en este día es la utilización del filtrado PCA para ver el comportamiento de la IA.

Fuimos a las ONIET en Córdoba y ganamos una medalla de oro en la competencia de Prototipos de accesibilidad e inclusión social.



En las competencias se nos quemó un puente H, por lo que realizamos pruebas de los motores para ver si seguían en buen estado

