EEG Analyser

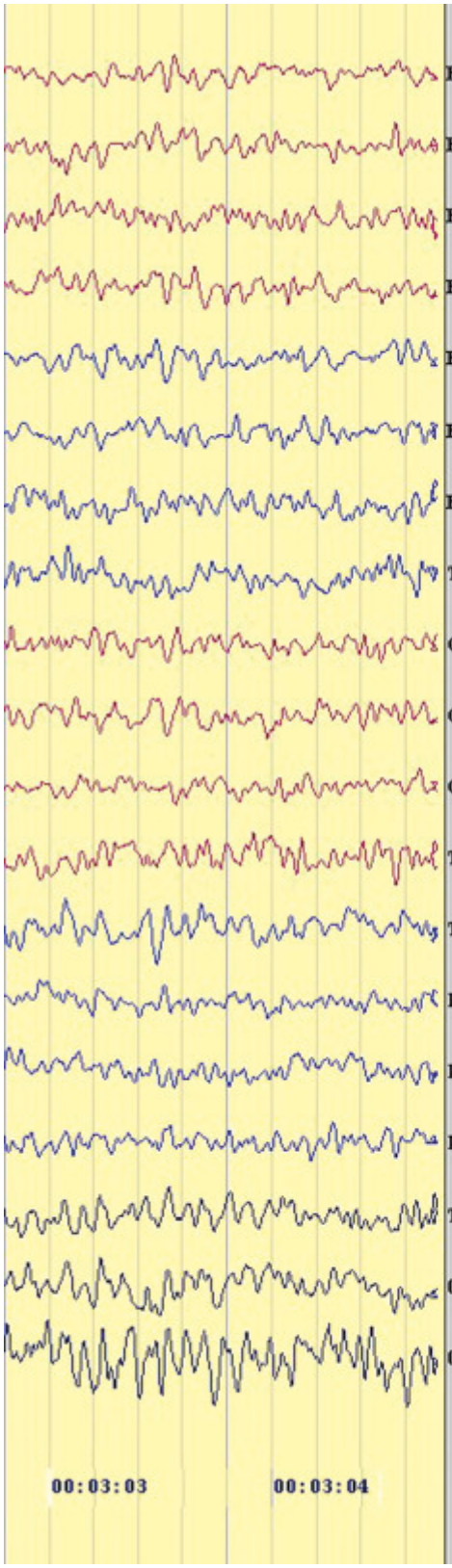
Introducere

- Proiectul propus consta intr-un sistem de monitorizare a activitatii cerebrale (EEG), realizat cu ajutorul unui convertor analog-digital de precizie ADS1115 si un set de filtre analogice pasive. Semnalul EEG este preluat de la electrozi si amplificat cu un circuit specializat, apoi transmis catre microcontrolerul ESP32. Acesta proceseaza si afiseaza in timp real forma de unda pe un ecran OLED si clasifica activitatea cerebrala in functie de frecventa in benzi caracteristice: delta, theta, alfa, beta si gamma.
- Scopul acestui proiect este dezvoltarea unui sistem portabil si accesibil de monitorizare a activitatii cerebrale, care poate afisa in timp real semnalele captate si poate clasifica tipurile de unde cerebrale.
- Ideea a pornit de la dorinta de a intelege mai bine cum functioneaza activitatea cerebrala in diferite stari (odihna, concentrare, somn) si cum poate fi aceasta masurata cu hardware relativ simplu si accesibil.
- Poate fi folosit in scopuri educationale, pentru cercetari de baza in neurostiinte, pentru aplicatii de relaxare si meditatie, dar si ca punct de plecare pentru dezvoltarea unor interfete creier-dispozitiv.

Frecventele la care functioneaza semnalele cerebrale:

Frequency band	Frequency	Brain states
Gamma (γ)	>35 Hz	Concentration
Beta (β)	12–35 Hz	Anxiety dominant, active, external attention, relaxed
Alpha (α)	8–12 Hz	Very relaxed, passive attention
Theta (θ)	4–8 Hz	Deeply relaxed, inward focused
Delta (δ)	0.5–4 Hz	Sleep

Mai jos am inclus un exemplu grafic care ilustreaza cum arata undele cerebrale, fiecare cu frecventele si amplitudinile specifice gamei din care face parte. Fiecare linie reprezinta o componenta a semnalului compus, iar pe baza acestor forme de unda este posibila identificarea si clasificarea activitatii cerebrale in gamele mentionate:

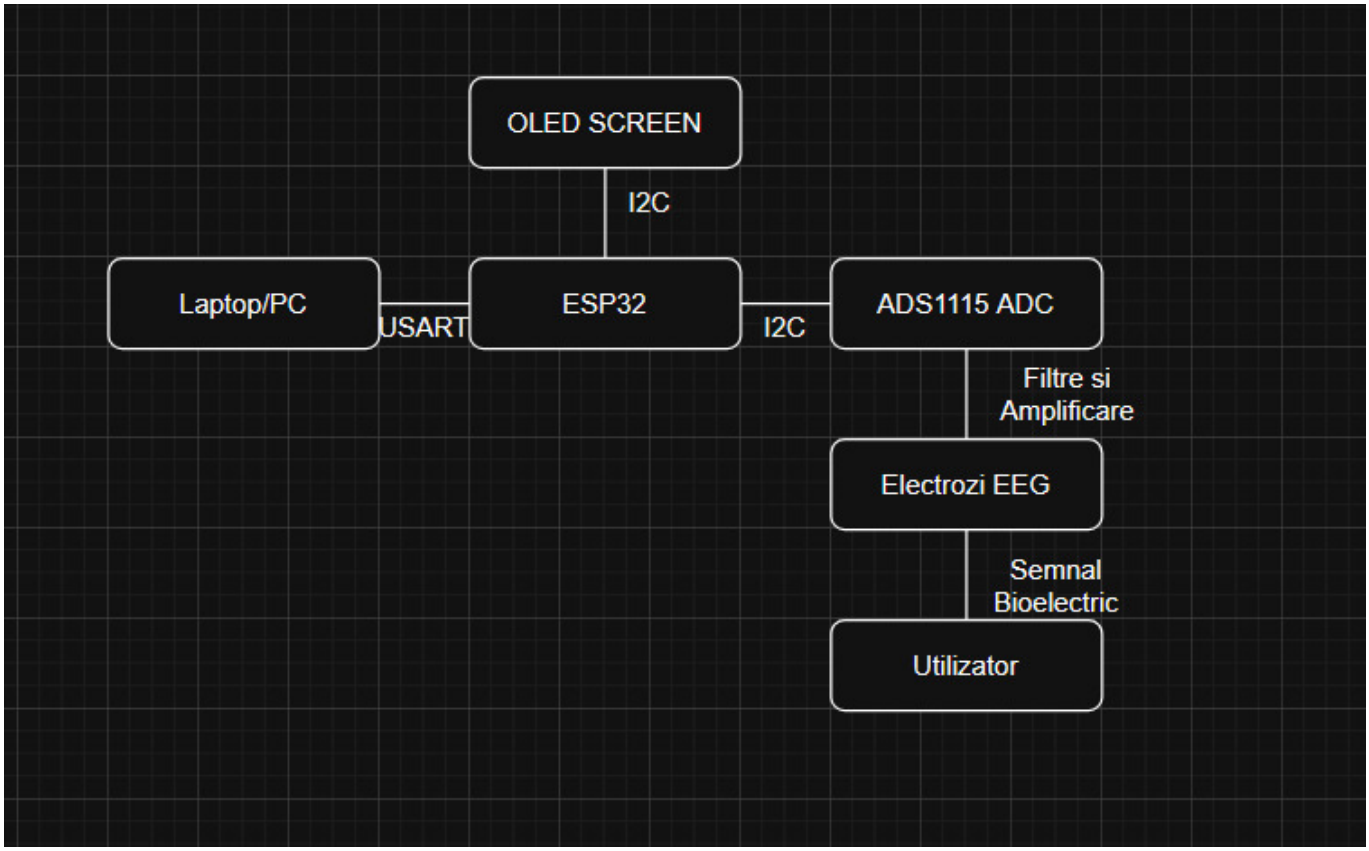


Electrode	F-Range (Hz)	Abs Power (µV ** 2)
FP1 - CAR	δ	4.20
	θ	2.29
	α	6.70
	β	3.37
FP2 - CAR	δ	5.72
	θ	4.17
	α	6.77
	β	2.24
F7 - CAR	δ	6.95
	θ	1.94
	α	2.86
	β	0.64
F3 - CAR	δ	4.08
	θ	4.54
	α	11.63
	β	1.96
FZ - CAR	δ	2.78
	θ	8.53
	α	7.55
	β	2.34
F4 - CAR	δ	4.82
	θ	4.19
	α	5.10
	β	2.68
F8 - CAR	δ	4.90
	θ	4.33
	α	7.02
	β	2.06
T3 - CAR	δ	33.79
	θ	1.72
	α	2.41
	β	4.26
C3 - CAR	δ	2.34
	θ	2.69
	α	5.28
	β	2.46
CZ - CAR	δ	5.35
	θ	11.80
	α	5.50
	β	1.88
C4 - CAR	δ	2.99
	θ	3.20
	α	2.72
	β	1.37

Descriere generală

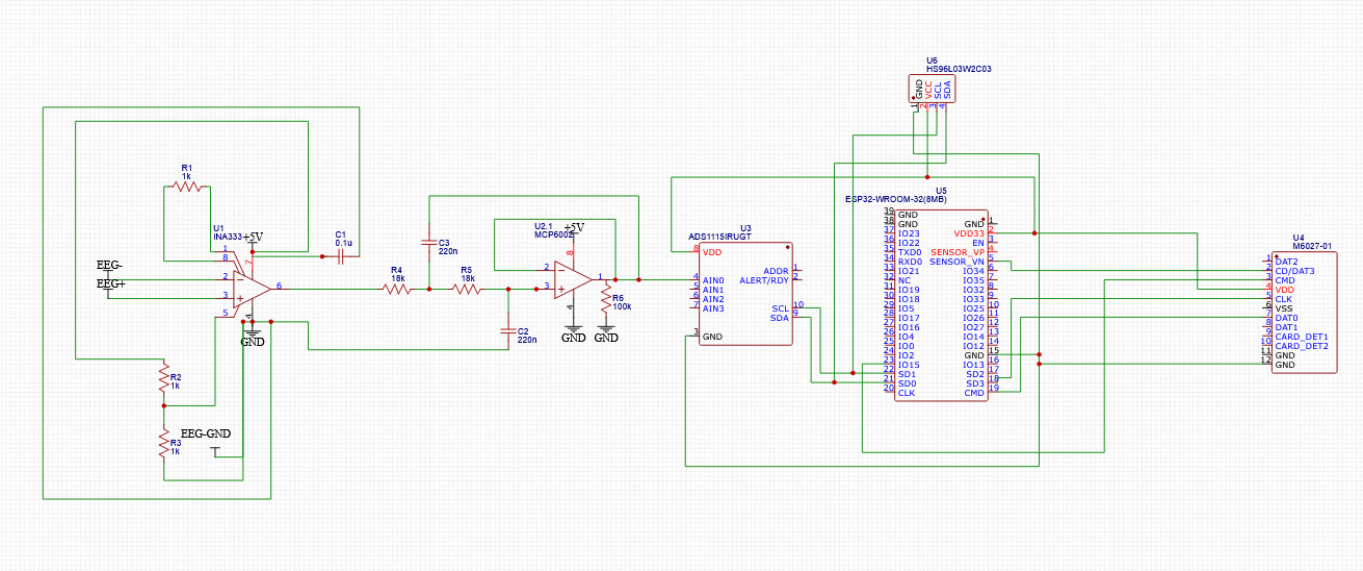
Sistemul EEG propus capteaza semnalul cerebral prin electrozi conectati la un amplificator si apoi la convertorul ADS1115. Semnalul este transmis catre ESP32, care il proceseaza in timp real. Datele apoi sunt afisate pe un ecran OLED si vor fi salvate pe un card microSD. Zgomotul de 50/60 Hz si

altele sunt filtrate digital pentru a obtine o clasificare precisa in benzi cerebrale.



Hardware Design

- Lista piese:
  - ADS1115 ADC
  - ESP32 DevKit-C
  - INA333
  - MCP6002
  - OLED SSD1306
  - Electrozi
  - Placa PCB pentru prototipare
  - Condensatoare, rezistente, breadboard si power bank
- Interfete Hardware:
  - I2C cu dispozitivele conectate (oled si ads1115) are rolul pentru a transmite date intre microcontroler si senzori/afisaj.
  - USART pentru transmiterea datelor de pe esp32 catre laptop, apoi prelucrate in python.
  - ADC pentru citirea semnalului EEG. Am pornit de la faptul ca semnalul preluat de la electrozi este foarte slab, de ordinul microvoltilor. Ca sa-l pot duce intr-un interval masurabil, am folosit un amplificator de instrumentatie INA333, care amplifica diferenta de potential dintre cei 2 electrozi in zona milivoltilor. Dupa amplificare, semnalul trece printr-un filtru trece-jos de tip Sallen-Key, implementat cu amplificatorul MCP6002, configurat cu o frecventa de cut-off de 40 de Hz, pentru a elimina zgomotul de retea si alte interferente. Astfel, raman doar gamele dorite ale EEG-ului: Delta, Theta, Alpha, Beta, pe care ulterior le poate analiza convertorul ADC.



Software Design

Acest cod de Arduino are rolul de a afisa in timp real graficul evolutiei tensiunii EEG preluate de la convertorul analog-digital ADS1115. Tensiunea este citita de pe canalul A0 si transmisa atat catre ecranul OLED pentru vizualizare. Prelucrarea ulterioara a semnalului EEG (incadrarea in benzi de frecventa) este realizata de codul python.

eeg_final_code.ino

```
#include <Wire.h>
#include <Adafruit_ADS1X15.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define GRAPH_WIDTH 128
#define GRAPH_HEIGHT 40
#define GRAPH_Y_OFFSET 20

Adafruit_ADS1115 ads;
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

float dataPoints[GRAPH_WIDTH];
int currentX = 0;
float visualGain = 10.0;

unsigned long lastSample = 0;
unsigned long sampleCount = 0;

void setup() {
  Serial.begin(115200);
```

```
  Wire.begin();

  if (!ads.begin()) {
    Serial.println("ADS1115 error!");
    while (1);
  }

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("Display error!");
    while (1);
  }

  ads.setGain(GAIN_ONE);
  ads.setDataRate(RATE_ADS1115_860SPS);

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println("EEG ready");
  display.display();

  Serial.println("timestamp,raw,voltage");
}

void loop() {
  unsigned long now = micros();

  if (now - lastSample >= 4000) {
    lastSample = now;
    sampleCount++;

    unsigned long timestamp = millis();
    int16_t raw = ads.readADC_SingleEnded(0);
    float voltage = raw * 4.096 / 32767.0;

    Serial.print(timestamp);
    Serial.print(",");
    Serial.print(raw);
    Serial.print(",");
    Serial.println(voltage, 6);

    float amplified = voltage * visualGain;

    dataPoints[currentX] = amplified;
    currentX = (currentX + 1) % GRAPH_WIDTH;

    drawWaveform();
  }
}
```

```
void drawWaveform() {
  display.clearDisplay();

  float minVal = dataPoints[0];
  float maxVal = dataPoints[0];
  for (int i = 1; i < GRAPH_WIDTH; i++) {
    if (dataPoints[i] < minVal) minVal = dataPoints[i];
    if (dataPoints[i] > maxVal) maxVal = dataPoints[i];
  }

  float range = maxVal - minVal;
  if (range < 0.001) range = 0.001;

  display.setTextSize(1);
  display.setCursor(0, 0);
  display.print("V:");
  display.print(dataPoints[(currentX - 1 + GRAPH_WIDTH) % GRAPH_WIDTH],
2);
  display.setCursor(65, 0);
  display.print("R:");
  display.print(range, 2);

  int baselineY = GRAPH_Y_OFFSET + GRAPH_HEIGHT / 2;
  for (int x = 0; x < GRAPH_WIDTH; x += 10) {
    display.drawPixel(x, baselineY, SSD1306_WHITE);
  }

  for (int i = 0; i < GRAPH_WIDTH - 1; i++) {
    int x1 = i;
    int x2 = i + 1;

    int y1 = GRAPH_Y_OFFSET + GRAPH_HEIGHT - ((dataPoints[i] - minVal)
/ range * GRAPH_HEIGHT);
    int y2 = GRAPH_Y_OFFSET + GRAPH_HEIGHT - ((dataPoints[x2 %
GRAPH_WIDTH] - minVal) / range * GRAPH_HEIGHT);

    y1 = constrain(y1, GRAPH_Y_OFFSET, GRAPH_Y_OFFSET + GRAPH_HEIGHT);
    y2 = constrain(y2, GRAPH_Y_OFFSET, GRAPH_Y_OFFSET + GRAPH_HEIGHT);

    display.drawLine(x1, y1, x2, y2, SSD1306_WHITE);
  }

  int markerY = GRAPH_Y_OFFSET + GRAPH_HEIGHT - ((dataPoints[currentX]
- minVal) / range * GRAPH_HEIGHT);
  markerY = constrain(markerY, GRAPH_Y_OFFSET, GRAPH_Y_OFFSET +
GRAPH_HEIGHT);
  display.fillCircle(currentX, markerY, 1, SSD1306_WHITE);

  display.display();
}
```

In implementarea mea am folosit doua coduri de python.

Primul script de python are rolul de a colecta datele transmise de ESP32, folosind biblioteca pyserial. Prin aceasta metoda pot capta in timp real tensiunile masurate cu o rata de esantionare de 250 Hz. Aceasta valoare este cea mai potrivita pentru analiza EEG (teorema Nyquist). Datele apoi sunt salvate intr-un fisier CSV, pentru a fi procesate de o biblioteca cu metode numerice mai avansate comparativ cu Arduino, in special pentru prelucrarea semnalelor.

Al doilea cod realizeaza analiza de frecventa a semnalului EEG prin aplicarea Transformatei Fourier. Pe baza acesteia am incadrat fiecare valoare in banda ei specifica, apoi la final facandu-se o analiza prompta a tot ce s-a intamplat pe parcursul analizei. Pentru observarea mai clara a semnalului, deoarece initial am fost pacalit si de osciloscop si de semnalul afisat pe oled, am crezut ca este zgomot, dar daca am amplificat artificial in codul arduino cu 10 semnalul, se poate observa foarte clar diferenta dintre zgomot si montajul electrozilor pe frunte si mastoida.

get_data.py

```
import serial
import time
import csv

def get_serial_port(serial_port, baud_rate, timeout=1):
    ser = serial.Serial(serial_port, baud_rate, timeout=timeout)
    time.sleep(2)
    return ser

def write_to_csv(serial_conn, csv_file_path):
    with open(csv_file_path, mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(["timestamp", "raw", "voltage"])

        while True:
            line = serial_conn.readline().decode(errors='ignore').strip
()
            if line.count(',') == 2:
                timestamp, raw, voltage = line.split(",")
                writer.writerow([timestamp.strip(), raw.strip(),
voltage.strip()])
                print(timestamp, raw, voltage)

def main():
    serial_port = "COM5"
    baud_rate = 115200
    csv_file = "eeg_data.csv"

    ser = get_serial_port(serial_port, baud_rate)
    write_to_csv(ser, csv_file)

if __name__ == "__main__":
    main()
```

processing_data.py

```
import numpy as np
import csv
import matplotlib.pyplot as plt
import pandas as pd
from scipy.fft import rfft, rfftfreq

def read_csv(file_path):
    timestamps = []
    raw_values = []
    voltage_values = []
    with open('eeg_data.csv', mode = 'r') as f:
        reader = csv.reader(f)
        next(reader)
        for row in reader:
            if len(row) == 3:
                timestamps.append(float(row[0]))
                raw_values.append(float(row[1]))
                voltage_values.append(float(row[2]))
    return timestamps, raw_values, voltage_values

def fourier_transform(voltage_values, samples, T):
    signal = np.array(voltage_values)
    fited_signal = signal * np.hamming(samples)

    xf = rfftfreq(samples, T)
    yf = rfft(fited_signal)

    return xf, yf

def compute_band_powers(xf, yf, eeg_bands):
    powers = {}
    energy = np.abs(yf) ** 2
    for band_name, (f_low, f_high) in eeg_bands.items():
        band_power = 0
        for i in range(len(xf)):
            if f_low <= xf[i] <= f_high:
                band_power += energy[i]
        powers[band_name] = band_power
    return powers

def plot_eeg_bands_heatmap(band_powers):
    plt.figure(figsize=(6, 1.5))
    band_labels = list(band_powers.keys())
    power_values = list(band_powers.values())

    plt.imshow([power_values], cmap='viridis', aspect='auto')
    plt.xticks(ticks=np.arange(len(band_labels)), labels=band_labels)
    plt.yticks([])
    plt.colorbar(label="Putere relativa")
```

```
    plt.title("Putere pe benzi EEG")
    plt.tight_layout()
    plt.show()
def main():
    T = 1.0 / 250.0
    timestamps = []
    raw_values = []
    voltage_values = []

    timestamps, raw_values, voltage_values = read_csv('eeg_data.csv')
    samples = len(voltage_values)

    xf, yf = fourier_transform(voltage_values, samples, T)

    eeg_bands = {
        "Delta": (0.5, 4),
        "Theta": (4, 8),
        "Alpha": (8, 12),
        "Beta": (12, 35),
        "Gamma": (35, 50)
    }

    band_powers = compute_band_powers(xf, yf, eeg_bands)
    magnitude = np.abs(yf)

    print("\nPutere pe benzi EEG:")
    for band, power in band_powers.items():
        print(f"{band}: {power:.2f}")

    plot_eeg_bands_heatmap(band_powers)

if __name__ == "__main__":
    main()
```

Rezultate Obţinute

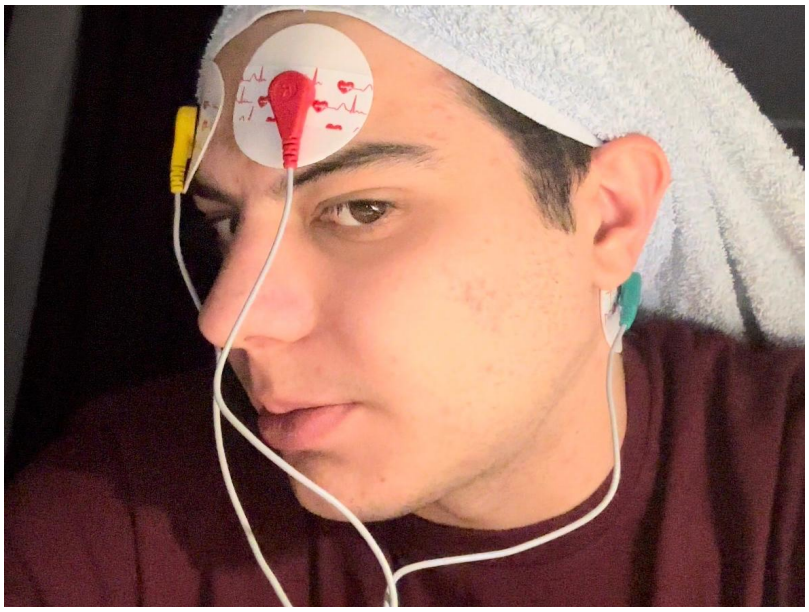
In urma experimentelor, se observa o diferenta clara intre zgomot si semnalul EEG valid, obtinut prin montarea corecta a electrozilor.

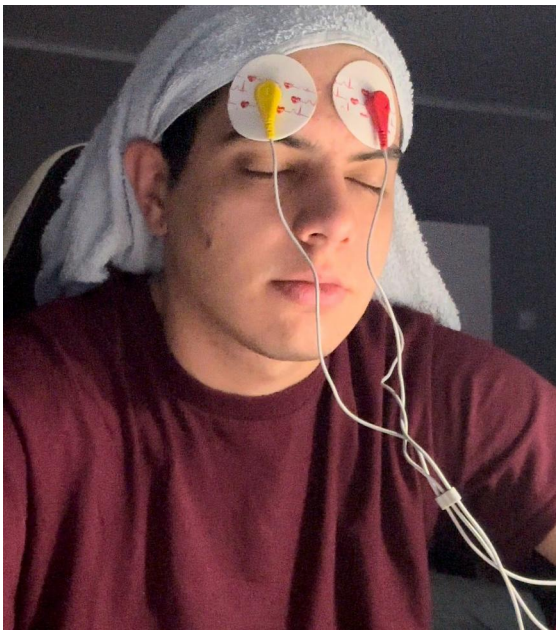
De exemplu, atunci cand electrozii au fost lasati in aer, analiza in frecventa a aratat o valoare extrem de ridicata in banda Delta (~11400) semnaland un zgomot evident.

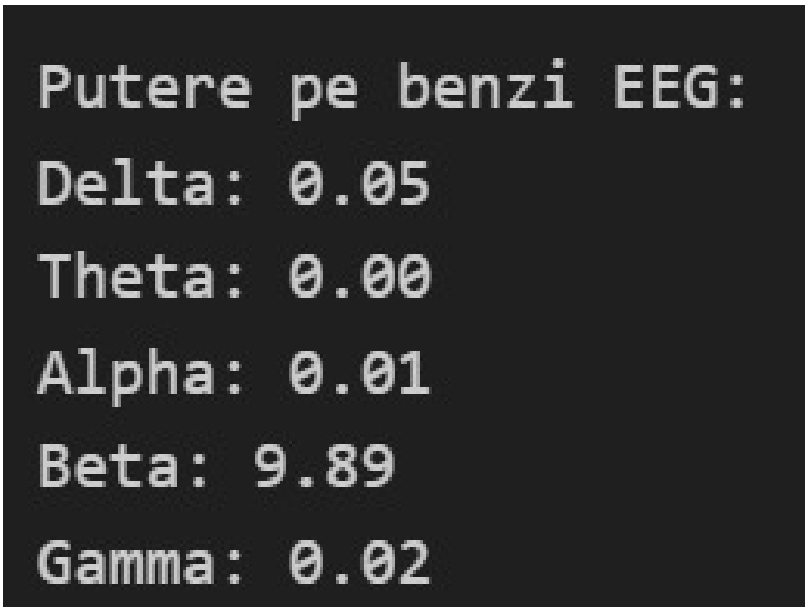
In schimb, in conditii reale, cu montajul plasat corect, s-au putut observa variatii semnificative in benzi, in functie de comportament (gama beta creste atunci cand vorbim sau gandim, alpha cand inchidem ochii si delta posibil un mic zgomot sau stare de oboseala).

Filmulet cu electrozii neconectati, zgomot: https://www.youtube.com/shorts/eSiJnOe5d7s

Mai jos imagini ale testarii efective cu electrozii conectati, dura foarte mult ca videoclip sa inregistrez o varietate de benzi.







Concluzii

Chiar daca filtrarea hardware nu elimina complet toate sursele de zgomot, se vede clar diferenta intre semnalul zgomotos si semnalul real cu montajul aplicat corect. Atunci cand am testat, se observa variatii corelate cu activitate, de exemplue, Beta creste cand vorbesc sau ma concentrez, iar Alpha apare usor cand sunt relaxat cu ochii inchisi. Am incercat sa stau cat mai nemiscat si relaxat in timpul inregistrarii si am certitudinea ca valorile reflecta favorabil activitate cerebrala reala, nu zgomot sau miscari musculare.

Totusi, exista si niste cazuri care ne spun ca exista zgomot: componente mici din banda Gamma sau usoare variatii la Delta pot proveni din zgomot, interferente electromagnetice sau chiar miscari musculare subtile. Putem imbunatati aceste defecte printr-o filtrare hardware mai eficienta sau prelucrare digitala mai avansata. Zgomotul ramas mai poate fi redus si prin contactul bun al

electroziilor cu pielea, mai ales la persoanele cu par, unde amplasarea electrozilor influenteaza mult calitatea semnalului.

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

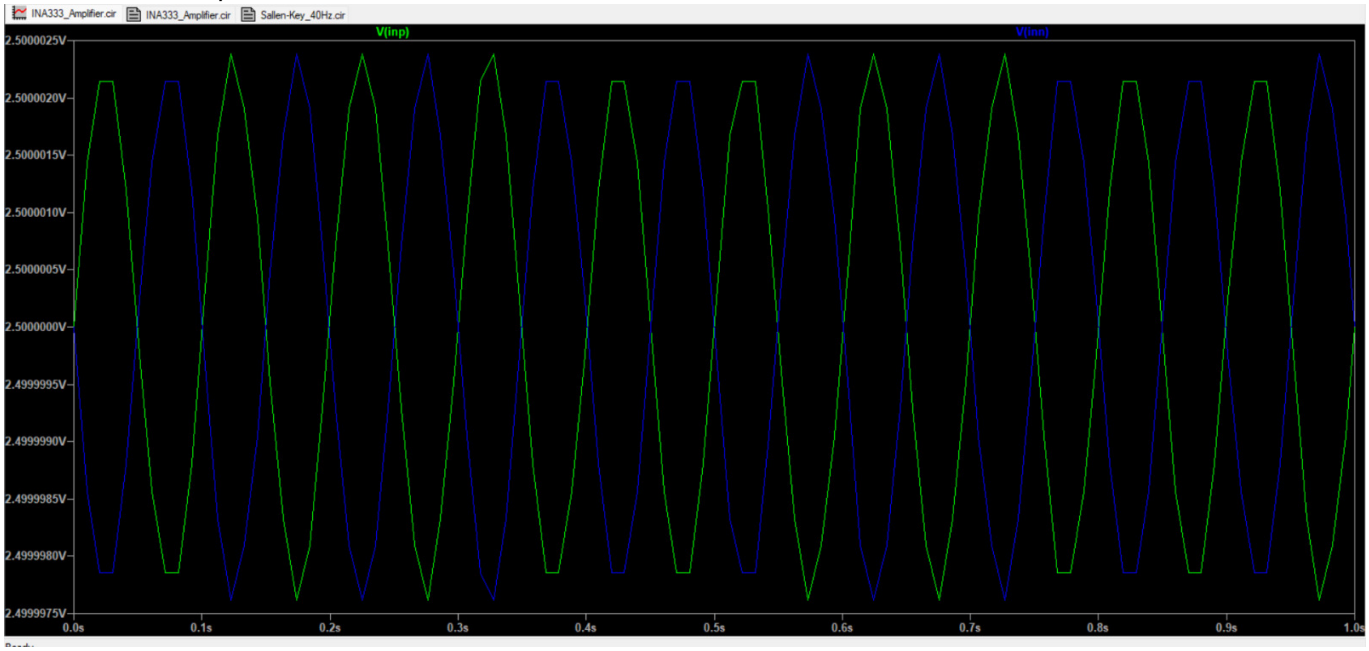
In aceasta sectiune voi evidentia cateva aspecte suplimentare. Saptamana aceasta a fost dedicata componentei hardware, am realizat mai multe simulari in LTSpice pentru a anticipa comportamentul circuitului odata ce va fi implementat fizic. Am testat semnale care acopera majoritatea gamelor EEG ce pot fi observate realist, pentru a evalua atat amplificarea cat si filtrarea acestora. Mai jos am pus codurile celor doua etape prin care obtin un semnal EEG pregatit pentru a fi citit de ADC:

ina333_test.cir

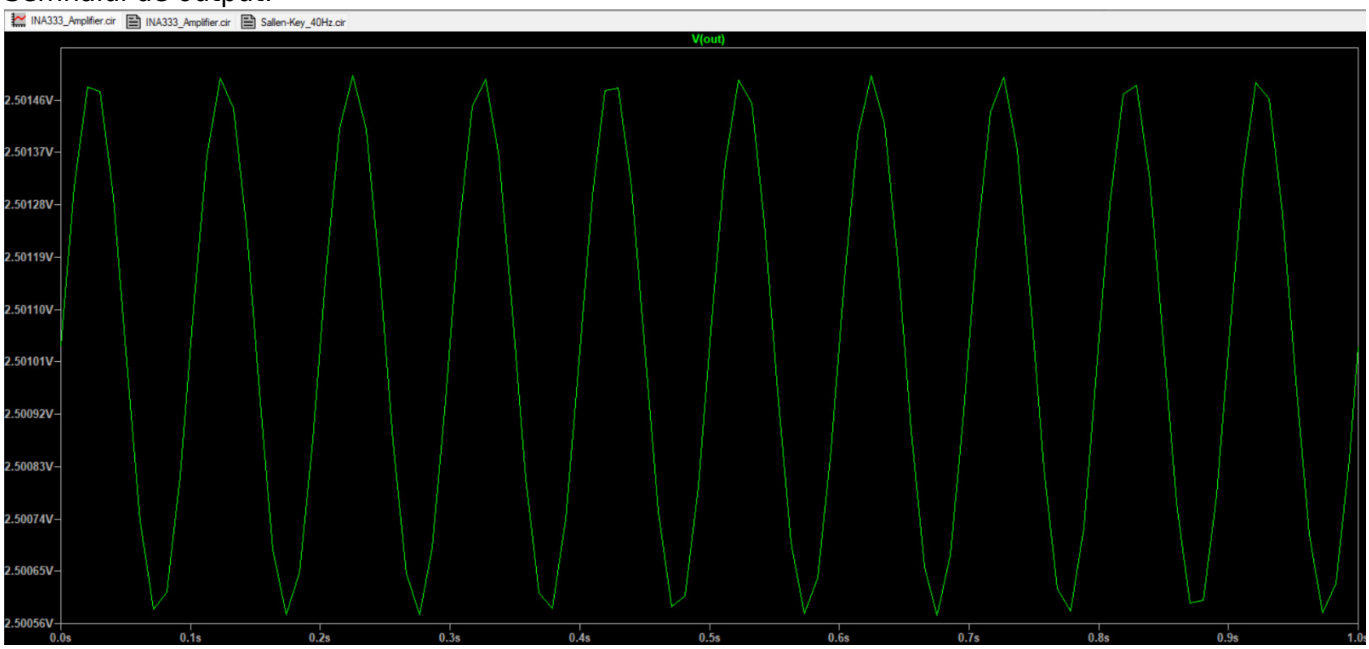
```
* Test INA333 EEG Amplifier

; Delta (1Hz, amplitudine 5µV)
; V1 INP 0 SIN(2.5 5u 1)
; V2 INN 0 SIN(2.5 -5u 1)

; Theta (6Hz, amplitudine 2µV)
; V1 INP 0 SIN(2.5 2u 6)
; V2 INN 0 SIN(2.5 -2u 6)

; Alpha (10Hz, amplitudine 2.3µV)
V1 INP 0 SIN(2.5 2.3u 10)
V2 INN 0 SIN(2.5 -2.3u 10)

; Beta (20Hz, amplitudine 1.8µV)
; V1 INP 0 SIN(2.5 1.8u 20)
; V2 INN 0 SIN(2.5 -1.8u 20)

VCC VCC 0 DC 5
VREF REF 0 DC 2.5
```

```
CDECOUPLE VCC 0 0.1u

RGAIN RG+ RG- 1k

XU1 INP INN VCC 0 OUT REF RG+ RG- INA333

.lib INA333.LIB

.options abstol=1n reltol=1u
.tran 0 1000ms 0 10us
.backanno
.end
```

Semnalele de input:



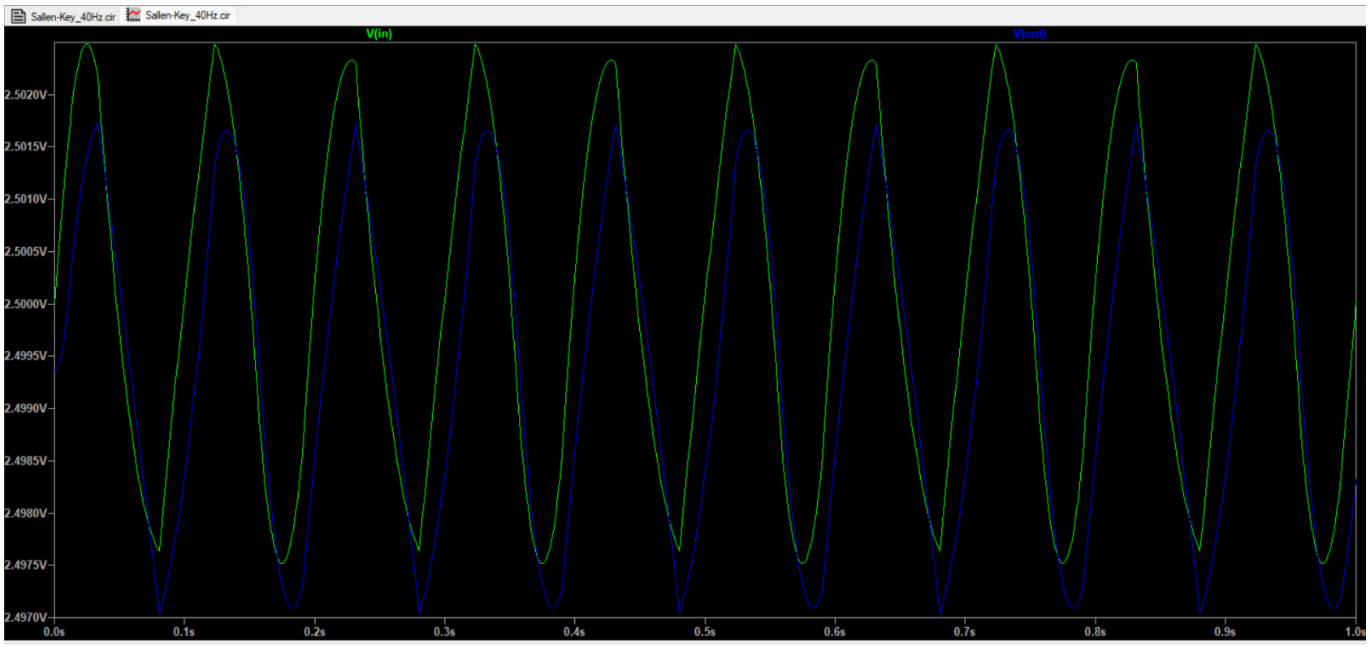
Semnalul de output:



sallen-key_test.cir

```
* Test Sallen-Key MCP6002 - Low-Pass Filter (~40 Hz cutoff)

* Delta - 1Hz, 5mV
* V1 IN 0 SIN(2.5 5m 1)
* Theta - 6Hz, 3mV
* V1 IN 0 SIN(2.5 3m 6)
* Alpha - 10Hz, 2.5mV
V1 IN 0 SIN(2.5 2.5m 10)
* Beta - 20Hz, 2mV
* V1 IN 0 SIN(2.5 2m 20)
* Zgomot 50Hz - 5mV
* V1 IN 0 SIN(2.5 5m 50)

VDD VDD 0 DC 5
R1 IN N1 18k
R2 N1 N2 18k
C1 N2 0 220n
C2 N1 OUT 220n
XU1 N2 OUT VDD 0 OUT MCP6002

Rload OUT 0 100k

.lib MCP6002.LIB
.tran 0 1s 0 1m
.options reltol=1e-3 abstol=1e-6
.backanno
.end
```

Semnalul nefiltrat/filtrat:



Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF