Inverted Pendulum Dev Kit

Made by: Tudor Petracovici

Grupa: 334CC

Asistent: Teodor Dicu

Introducere

Pendulul Invers este in sine o problema si un challenge in lumea sistemelor controlate si al algoritmilor de control. Exista multe variatiuni, unele pe ax fix, iar altele ce au un grad de libertate mult mai mare.

Proiectul implementeaza un Inverted Pendulum Development Kit bazat pe ESP32, ce vine cu toate feature-urile necesare pentru controlarea unui pendul invers, dar putand fii mai departe utilizat in diverse aplicatii dat fiind versatilitatea acestuia.

Ideea de la care am pornit a fost în sine echilibrarea unui pendul, vazand pe internet tot felul de self-balancing robots care mi-au trezit interesul. In plus, mereu am vrut sa step up a notch si sa cresc dificultatea implementarii unui algoritm de PID de la simplii line followers, dar aplicatii de tipul acesta in care sunt mult mai multe forte externe implicate.

Acest Development Kit este foarte facil pentru studentii care vor sa intre mai adanc in lumea algoritmilor de control, pentru ca pot vedea un feedback live la cum afecteaza cele 3 valori (K_p, K_i, K_d) din PID, sistemul.

Descriere generală

Schema bloc a proiectului:

Unghiul actual al pendulului este calculat cu ajutorul a celor 1200 de pulsuri generate de un incremental rotary encoder. Aceasta sunt impartite pentru a putea ajunge la o mapare exacta in cadranul unui cerc, 0 grade fiind pozitia de start, iar 180 de grade fiind pozitia de echilibru al sistemlui, mai exact pe verticala.

Avand 2 core-uri disponibile, am ales sa fac o aplicatie multi-threading. Unul din core-uri se va ocupa cu comunicarea user-ului cu web server-ul pentru a ne asigura ca mereu primim cele mai recente

valori pentru PID, iar al doilea core se va ocupa strict cu generarea semnalului PWM, citirea impulsurilor encoder-ului si controlul motorului. Pentru ușurarea sistemului avem urmatoarele surse de input:

Interactiunea componentelor:

Pentru alimentarea circuitului am folosit o sursa in comutatie de 12V si de 6A, deoarece motorul in load maxim poate consuma pana la 3A. Din sursa este alimentat H-bridge-ul de 6A de la Infineon si in acelasi timp un buck converter ce coboara tensiunea la 5V pentru a alimenta ESP-ul prin pin-ul de VIN. In acelasi timp, encoder-ul este alimentat tot la 5V dar cele doua faze de output ale acestuia sunt trase la 3V3 prin 2 rezistene de 2k7 pentru a fi in aceeasi logica de functionare cu cea a ESP-ului. Pendulul este pus in miscare de un motor de 12V cu cuplu mare si 280 de RPM, pentru a putea sustine si greutatea pendulului + encoder-ului, dar si pentru a putea schimba rapid directia pentru stabilizarea pendulului. Ca platforma, am ales ESP32 deoarece este un dual core rapid. Dupa cum am zis, proiectul va ajuta si ca un learning model pentru aplicarea PID-ului, putand schimba valorile K_p, K_i si k_d live.

Pentru ca ESP-ul este rapid, a trebuit sa realizez un semnal de ceas ce functioneaza la 1000Hz ce genereaza o intrerupere la fiecare milisecunda pentru a putea citi la intervale fixe valorile encoder-ului si pentru a putea calcula corect interval-ul intre citiri. Daca nu as fi folosit acest timer, diferenta de timp intre citiri ar fi fost la nivelul micro secundelor, lucru ce incurca funtionarea corecta a pendulului.

Principiu de functionare: Encoder-ul trimite pulsuri catre uC ce sunt citite pe o intrerupere. Aceste impulsuri sunt transformate in grade cu ajutorul unui calcul matematic simplu. Unghiul apoi este bagat ca feedback in PID controller, unde setpoint-ul este setat la 180 de grade (vrem ca pendulul sa stea vertical in sus).

Output-ul ete un semnal pwm generat de ESP ce functioneaza la o frecventa de 1000Hz si o rezolutie de 16 biti. Timer-ul de 1000Hz, asa cum am spus mai sus, este folosit pentru a stii exact intervalul intre citiri si de a nu avea schimbari aleatorii de timp intre citiri.

Hardware Design

Listă de piese:

- ESP32 DevKit V1 - ESP32-WROOM-32 240Mhz
- Incremental Rotary Encoder - folosit pentru calcularea unghiului
- Motor DC 12V, 280RPM
- Sursa de tensiune in comutatie Meanwell 12V 75W
- Buck-Converter LM2596HS-ADJ
- INFINEON IFX9201SG 6A H-bridge motor driver
- Pendul realizat din fibra de sticla
- Diverse bracket-uri si componente printate 3D
- Componente electronice de baza (condensatoare, rezistente, etc.)

Schematics

IFX9201SG:

Sistemul este dotat cu un H-bridge foarte potent de 6A, pentru a rezista la curentii inversi de mare putere generati de schimbarea directiei fortate si rapide de catre motor in urma controlului PID.

LM2596HVS:

Am ales folosirea unui buck converter (fast switching power supply), in combinatie cu diverse combinatii de condensatoare, pentru a putea scapa cat mai mult de zgomot-ul produs de alimentare (fie ea prin USB, priza, etc.). Cu cat exista mai putin zgomot intr-un sistem ce foloseste un algoritm de control, cu atat este mai bine.

ESP32:

Microcontroller-ul trebuie sa genereze un semnal de pwm de 1000 de Hz la 16 biti rezolutie, un timer cu un prescaler de 80 pentru a putea computa si cuantifica valorile encoder-ului la fiecare milisecunda, iar in acelasi timp sustinerea unui server web pentru utilitatea utilizatorilor. De aceea consider ca acest dual core este indispensabil.

Schema finala: schematic_invertedpendulum.pdf

Pe langa componentele de baza ale acestui dev kit (driver, buck converter), avem cate un set de pini mama pentru fiecare pin al esp-ului, pentru a putea adauga exstentii oricand si pentru a putea folosi placa in diverse proiecte.

PCB and 3D designs

PCB Design:

Pentru a realiza un design simplu si eficient, am ales sa amplasez toate componentele pe top layer, planul de masa aflandu-se atat pe top cat si pe bottom layers. Amplasarea componentelor a fost separata pe cele doua parti ale ESP32-ului, partea de alimentare si partea de putere. Am facut acest lucru deoarece am avut nevoie de o portiune mare si libera de plan de masa pentru a ajuta driver-ul sa nu se incalzeaza la curentii inversi.

Latimea traseelor de semnal este de 0.3mm, iar cele de putere sunt de 0.5mm - 0.7mm. Placa are dimensiunea de 58.7 mm* 74.4 mm.

Fiecare pin de pe ESP32 este legat la un pin mama, pentru a putea adauga extensii de senzori/drivere/orice altceva on the spot. Componentele care sunt deja venite din “fabrica” au cate un jumper pad care se poate dezlipi, astfel nemaifolosind componenta. Aceste componente sunt:

- LED on pin 22
- LED on pin 23

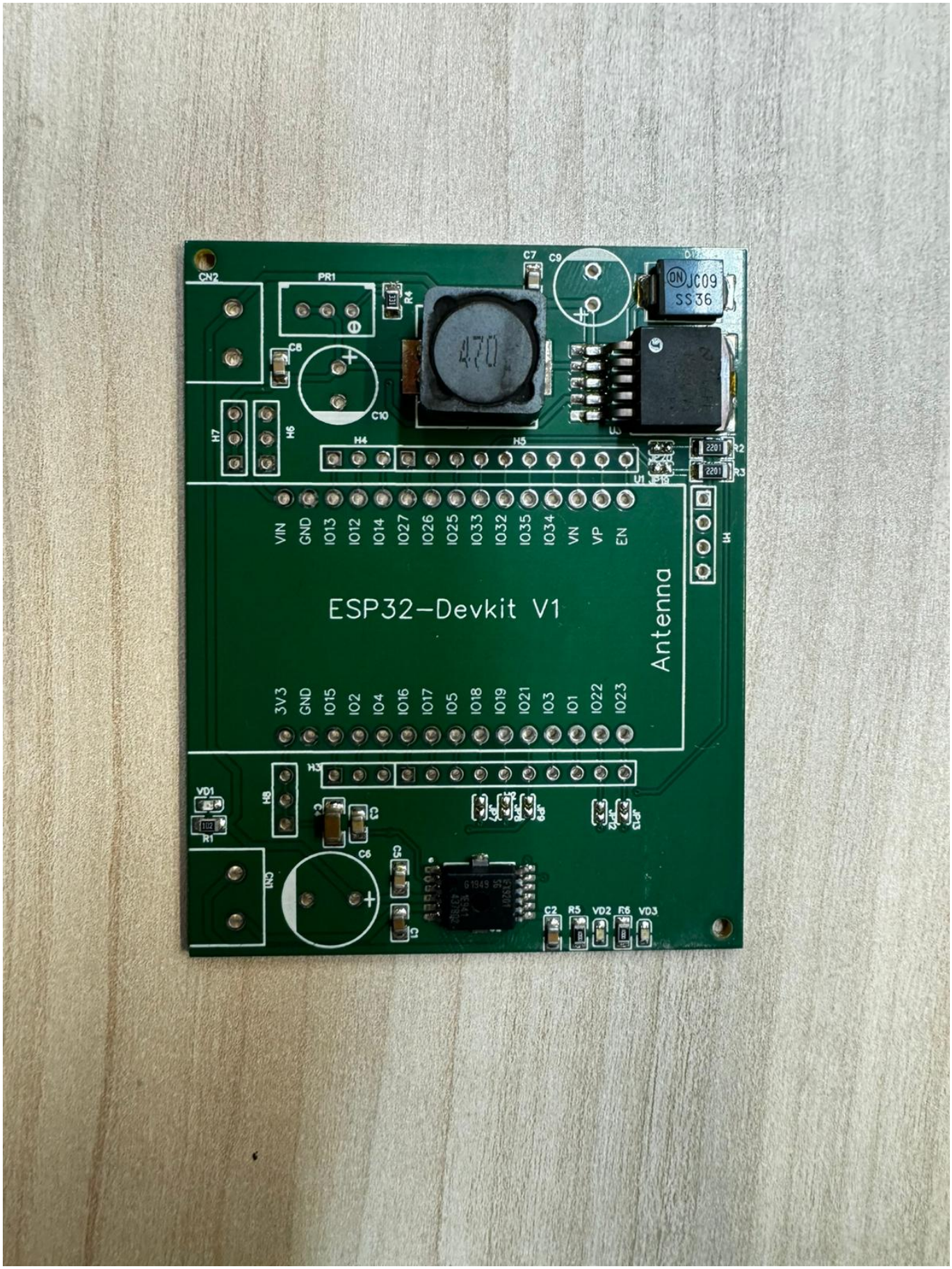
- ENCODER_A on pin 32
- ENCODER_B on pin 33
- MOTOR_DIR on pin 21
- MOTOR_EN on pin 19
- MOTOR_PWM on pin 18

Modelul 3D al PCB-ului:

Am generat toate componentele pentru a putea vedea o ampla viziune asupra lor mai din timp.

Modelele 3D ale proiectului:

Actual PCB and soldering



Soldering-ul a fost unul foarte usor si placut. SMD-urile au fost lipite cu ajutorul unui Stencil si pasta, la final bagate intr-un cuptor. (driverul de motor si integratul buck converter-ului au pad-uri foarte mari de masa, iar pentru aceasta a fost nevoie de un cuptor.

Piesele TH si connectorii au fost lipiti dupa. Din fericire nu am observat nici o problema la PCB, in schimb, nu toate placutele de tip ESP32 veneau cu o dioda intre VUSB si VIN, iar astfel am fost nevoie sa modific anumite placute on the spot.

Software Design

Descriere si aplicabilitate

Multi-threading

Core-ul 0 a fost folosit pentru controlul motoarelor, calculul PID-ului si citirea encoder-ului, iar Core-ul 1 s-a asigurat de buna functionare a placutei in sine (sunt anumite procese care trebuiesc rulate orice ar fi intern), cat si de mentinerea si rularea unui server web ce trimite catre placuta valori pentru PID.

Timere

Toate cele 4 timere de pe ESP32 sunt pe 64 de biti, astfel am putut crea un timer ce genereaza o intrerupere la fiecare mili secunda (are frecventa de 1000Hz), interval in care pot citi, procesa si aplica controlul asupra motoarelor. Am ales folosirea unui timer pentru a nu avea de facut calcule extra privind diferenta intre ultima computatie de eroare si cea curenta.

PWM

Pentru a putea mapa cat mai precis controlul la motor, am ales sa imi generez singur un semnal de PWM de 1000 de Hz, pe o rezolutie de 16 biti, pentru a avea un control cat mai mare. Astfel, pot mapa valori pana la aprox 65k, in loc de simplul si vechiul 255 de pe 8 biti.

Platform.io

Platform.io este un mediu de dezvoltare built on top of VS Code, mult mai rapid si mai versatil decat simplul Arduino IDE, ce ofera posibilitatea programarii multor environment-uri, printre care si cel de la Esspressif32. Compilarea si upload-ul catre placi este mult mai smooth, putand face si software debug prin serial.

Algoritmi si structuri

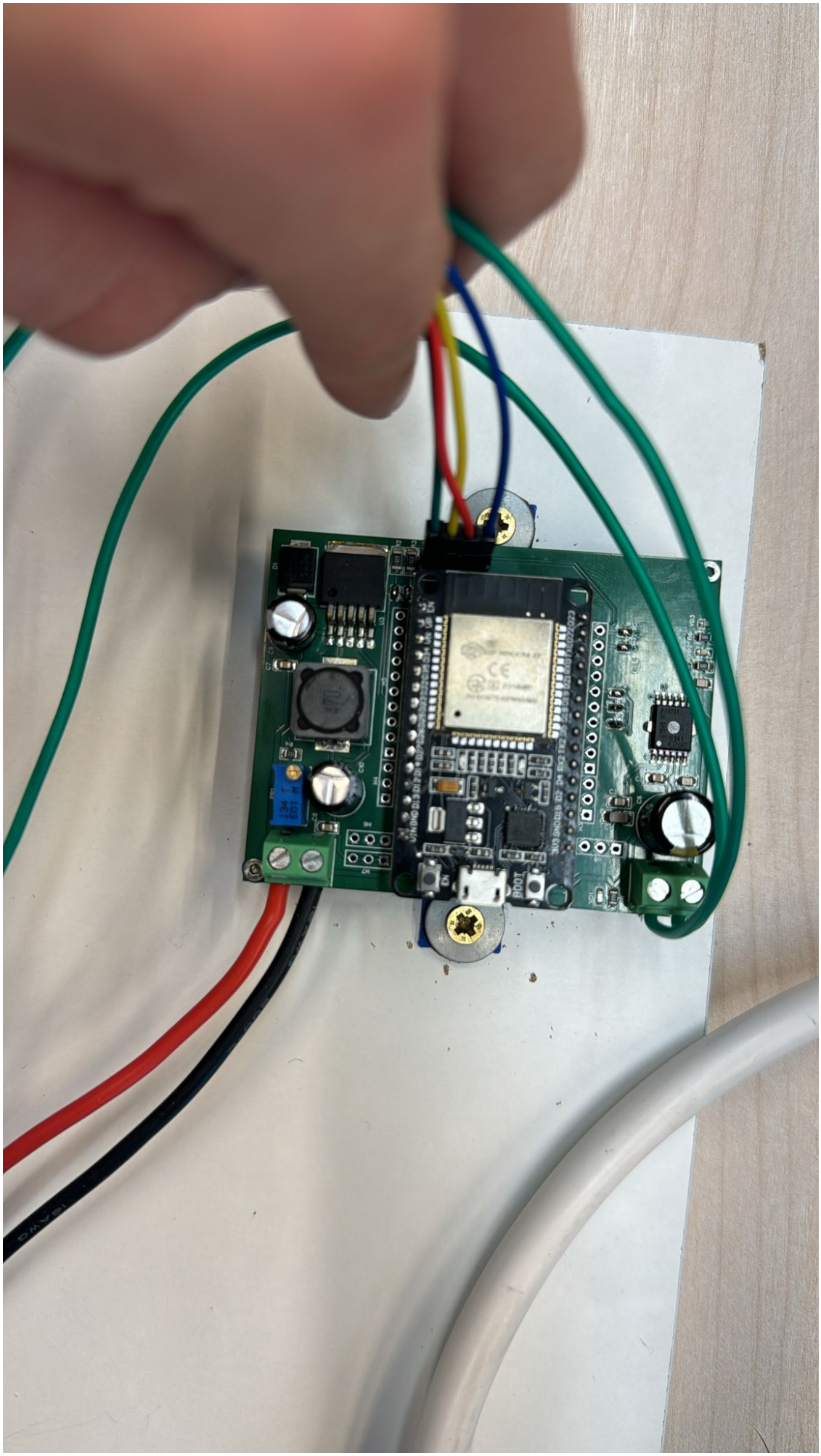
Pe partea de algoritmi, proiectul se bazeaza pe un algoritm de control, mai exact PID, ce se foloseste de unghiul la momentul T al pendulului ca input.

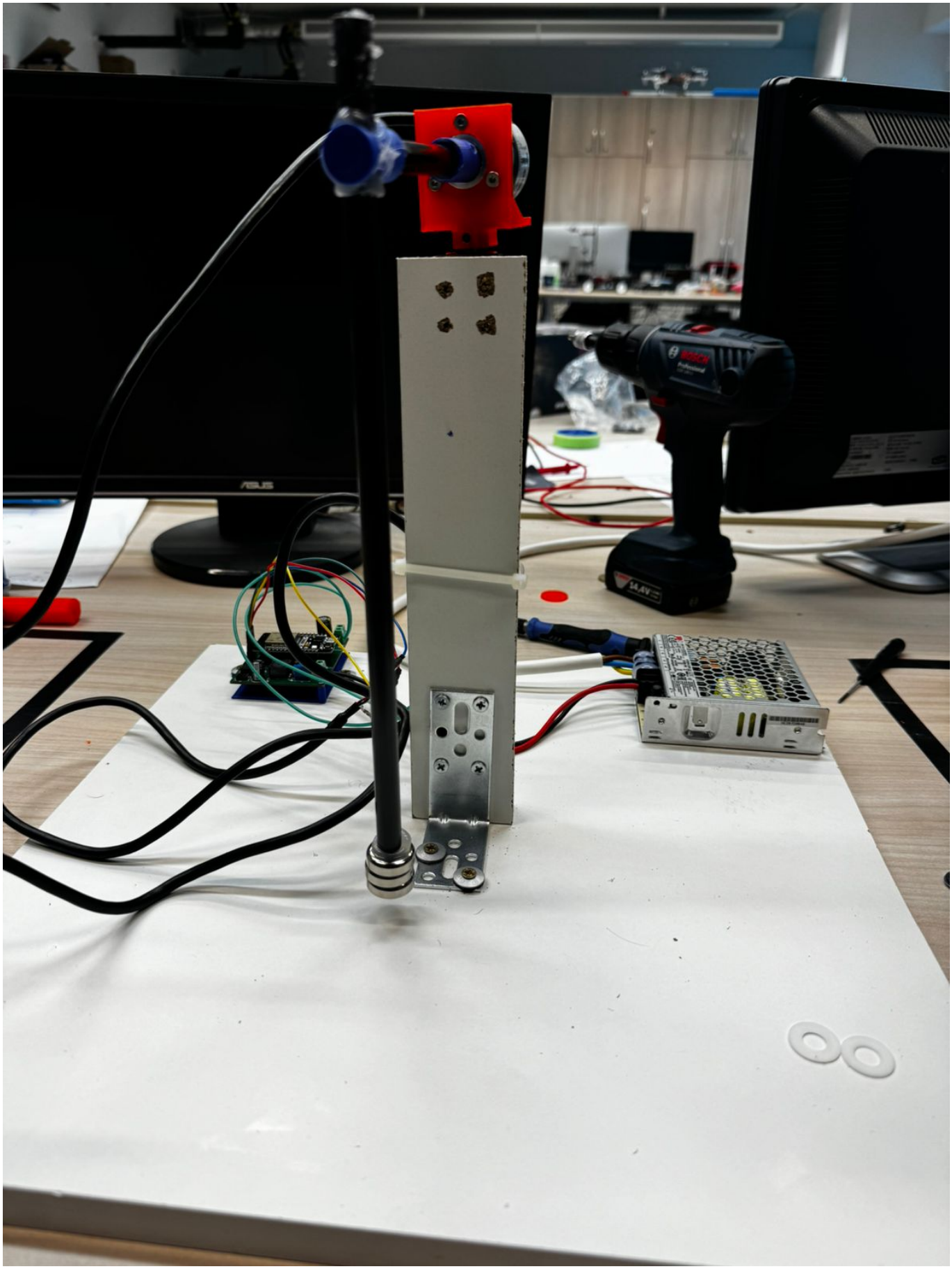
Pseudocod algoritm PID:

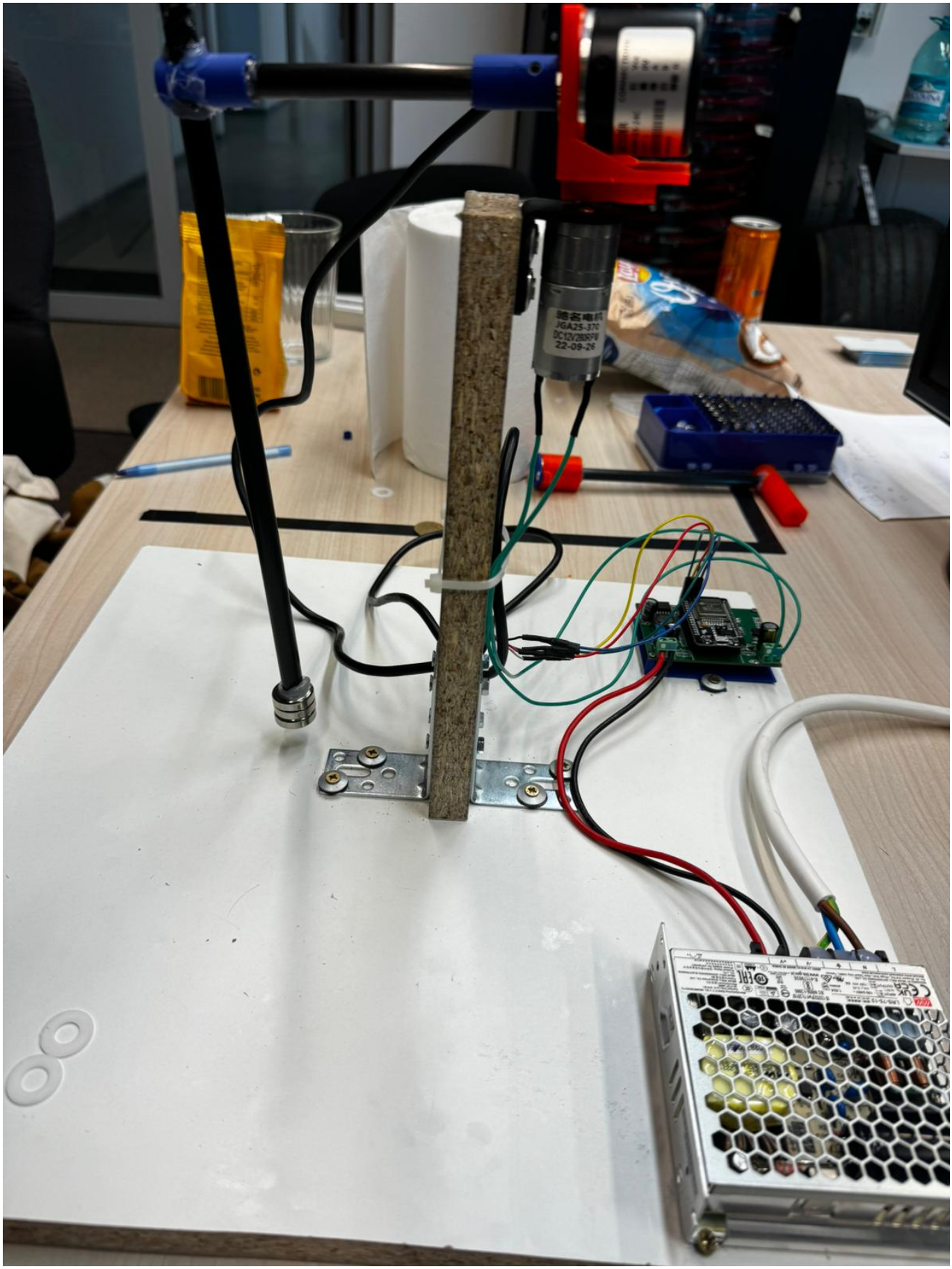
```
k_p = 1
k_i = 0
k_d = 0

# Loop forever
interval = 1 #1ms
error_prev = 0
integral = 0
setpoint = 180
while True:

    # Get angle from encoder(feedback)
    val = encoder->get_angle()

    # Calculate the PID terms
    error = setpoint - val
    integral = integral + (error * interval)
    derivative = (error - error_prev) / interval
    output = (k_p * error) + (k_i * integral) + (k_d * derivative)

    # Save value for next iteration
    error_prev = error
```

Rezultate Obţinute

The final Product




ASUS
BOSCH



I'm very proud of the final product. Este un sistem complex ce poate fi imbunatatit din foarte foarte multe puncte de vedere, incepand cu partea mecanica pana la componente, plus procesare si senzori in plus (un encoder pe motor spre exemplu).

Concluzii

Probleme

Mecanica

Am avut multe probleme legate de componentele printate 3D. Foarte multe imbinari si bracket-uri nu fac fata la soc, se indoaie si incep sa afecteze structura sistemului.

Solutie: Componente metalice taiate la laser CNC.

Bibliografie/Resurse

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

Export to PDF