

# DC MOTOR POZÍCIÓSZABÁLYOZÓ RENDSZER FEJLESZTÉSE LABVIEW ALAPÚ FELHASZNÁLÓI FELÜLETTTEL

# RAZVOJ SISTEMA ZA POZICIONU REGULACIJU DC MOTORA SA LABVIEW INTERFEJSOM

# DEVELOPMENT OF A DC MOTOR POSITION CONTROL SYSTEM WITH A LABVIEW-BASED USER INTERFACE

Hallgató

NAGY RICHÁRD  
25223020

Mentor

MR. PÓTH MIKLÓS

Szabadka, 2025

## Absztrakt

Ebben a munkában egy egyenáramú (DC) motor pozíciószabályozására alkalmas beágyazott platform fejlesztése és megvalósítása került bemutatásra. A rendszer alapját egy ESP32-S3 mikrovezérlő képezi, amely FreeRTOS valós idejű operációs rendszer alatt futtatja a szabályozási algoritmust. A motor meghajtásáról egy DRV8876 H-híd meghajtó IC gondoskodik, míg a pozíció-visszacsatolást egy 1024 PPR felbontású inkrementális enkóder biztosítja. A szabályozás magját egy diszkrét idejű PID algoritmus adja, amelyet mozgásprofil-generátor (trapéz- és S-görbe), elakadás-detektor, beállás-érzékelő, lágy pozíciókorlát és drift-felügyeleti egészít ki. A platform USB CDC soros interfészen keresztül kommunikál a felügyeleti rendszerrel, lehetővé téve a célpozíció beállítását, a PID paraméterek valós idejű hangolását, valamint a motor állapotának folyamatos visszajelzését. A moduláris, rétegzett firmware-architektúra és a hardverabsztrakciós interfések révén a rendszer alkalmas oktatási, kutatási és prototípus-fejlesztés célokra egyaránt, mivel lehetővé teszi különböző szabályozási stratégiák és paraméterek hatékony vizsgálatát valós hardverkörnyezetben.

## Kulcsszavak:

DC motor, PID szabályzó, pozíciószabályozás, ESP32-S3, DRV8876, beágyazott rendszer, FreeRTOS, mozgásprofil, kvadratúra enkóder

## Tartalom

Absztrakt.....	2
Tartalom.....	3
Jelmagyarázat.....	4
Köszönnetnyilvánítás.....	5
1. Bevezető.....	6
2. DC Motor És Enkóder Működése.....	7
2.1. DC Motor Működése .....	7
2.2. H-Híd Kapcsolás.....	7
2.3. DC motor matematikai modellje.....	8
2.4. Enkóder Működése.....	11
3. PID Irányítás.....	12
3.1. PID felépítése .....	12
3.2. Valós PID irányítás.....	13
4. Rendszer hardveres felépítése .....	14
4.1. Tápellátás .....	14
4.2. Mikrovezérlő .....	15
4.3. DC motor irányítása .....	16
4.4. Analóg mérőáramkörök és jelkondicionálás .....	17
4.5. Pozicíció Mérése .....	19
4.6. RS485 interfész .....	20
4.7. Számítogépes interfész és programozás .....	21
4.8. RGB Led .....	23
4.9. Nyomtatott áramköri lap kialakítása .....	24
5. A firmware architektúrája és működése .....	27
5.1. Áttekintés .....	27
5.2. Perfériák .....	27
5.3. FreeRTOS taskok .....	27
5.4. Architektúra .....	28
5.5. Modulok .....	29
5.5.1. Hardver meghajtó réteg .....	29

5.5.2.	Szabályzási Modulok.....	30
5.6.	Alkalmazás réteg.....	31
6.	LabView Felhasználói interfész.....	32
6.1	Visa Kommunikációs Modul.....	34
6.2	3D vizualizáció LabVIEW-ban.....	35
6.3	PID paraméterek beállítása és lekérése .....	36
6.4	Szög elfordolás megjelenítése.....	38
7.	A rendszer működése és jövőbeli írányok .....	39
8.	Összefoglaló.....	40
9.	Ídegen nyelvű Összefoglaló.....	41
10.	Irodalom jegyzék .....	42

## Jelmagyarázat

Jel/Rövidítés	Értelmezés
DC Motor	Egyenáramú motor (angolul: Direct Current motor)
PID	Proporcionális-Integráló-Deriváló szabályzó (angolul: Proportional-Integral-Derivative controller)
PPR	Impulzus per fordulat, az enkóder felbontása (Pulses Per Revolution)
PWM	Impulzusszélesség-moduláció, motorvezérlésre használt jel (Pulse Width Modulation)
H-híd	Motorirányváltó teljesítménykapcsolás (angolul: H-Bridge)

## Köszönetnyilvánítás

Szeretnék köszönetet mondani mindeneknek, akik támogatásukkal hozzájárultak ennek a projektnek a megvalósításához. Külön köszönöm Mr. Póth Miklós-nak a szakmai útmutatást és a hasznos tanácsokat, amelyek nagyban segítették a munkám fejlődését. Hálával tartozom a családomnak is, akik folyamatosan biztattak és biztosították a szükséges hátteret a projekt elkészítéséhez. Támogatásuk nélkül ez a munka nem jöhetett volna létre.

## 1. Bevezető

A modern ipari automatizálás és robotika fejlődésével az irányítástechnikai rendszerek egyre fontosabb szerepet töltnek be a mérnöki gyakorlatban. A pontos pozícionálás, sebesség- és nyomatékszabályozás elengedhetetlen követelménye számos alkalmazásnak, legyen szó akár egy ipari robotkar mozgatásáról, egy CNC megmunkálógép szerszámvezetéséről, vagy egy elektromos jármű hajtásláncának vezérléséről. Ezen feladatok megoldásában kulcsszerepet játszanak az egyenáramú (DC) motorok, amelyek egyszerű felépítésük, könnyen szabályozható jellemzőik és széles körű elérhetőségük miatt az ipar egyik leggyakrabban alkalmazott hajtástípusává váltak.

Az irányítástechnika területén számos szabályozási módszer áll rendelkezésre a kívánt rendszerviselkedés elérésére. A klasszikus megközelítések közül kiemelkedik a PID (Proporcionális-Integráló-Deriváló) szabályozó, amely matematikailag jól megalapozott, könnyen hangolható és robusztus működést biztosít. A PID szabályozók évtizedek óta bizonyítják hatékonysegüköt az ipari környezetben, és ma is a legszélesebb körben alkalmazott visszacsatolt irányítási megoldásnak számítanak.

A szabályozó algoritmus hatékony működése azonban önmagában nem elegendő – annak valós hardveren történő megvalósítása és a teljes rendszer integrációja legalább olyan fontos mérnöki feladat. Egy beágyazott motor szabályzó platform tervezése és fejlesztése magában foglalja a megfelelő mikrovezérlő kiválasztását, a motorhajtó fokozat kialakítását, a visszacsatoló szenzor illesztését, a valós idejű szoftverarchitektúra megtervezését, valamint a felügyeleti kommunikáció megvalósítását. Ezek együttese határozza meg, hogy az elméleti szabályozási teljesítmény a gyakorlatban is realizálható-e.

Jelen projekt célja egy DC motor pozíciós szabályozására alkalmas beágyazott platform tervezése és megvalósítása. A rendszer egy ESP32-S3 mikrovezérlőre épül, amely FreeRTOS valós idejű operációs rendszer alatt futtatja a diszkrét idejű PID szabályozó algoritmust. A motor meghajtásáról DRV8876 H-híd meghajtó IC gondoskodik, a pozíció-visszacsatolást pedig egy 1024 PPR felbontású inkrementális kvadratúra enkóder biztosítja. A szabályozó algoritmust mozgásprofil-generátor, elakadás-detektor, beállás-érzékelő és lágy pozíciókorlát egészíti ki, amelyek együttesen megbízható és biztonságos működést nyújtanak. A platform USB soros interfészen keresztül kommunikál a felügyeleti rendszerrel, lehetővé téve a célpozíció beállítását, a PID paraméterek valós idejű hangolását és a motor állapotának folyamatos megfigyelését.

A kidolgozott rendszer oktatási és prototípus-fejlesztési célokat egyaránt szolgál. Moduláris felépítésének és hardverabsztraktiós rétegeinek köszönhetően alkalmas különböző szabályozási stratégiák és paraméterekek hatékony vizsgálatára és összehasonlítására valós hardverkörnyezetben, ami az elméleti ismeretek gyakorlati alkalmazásának mélyebb megértését segíti elő.

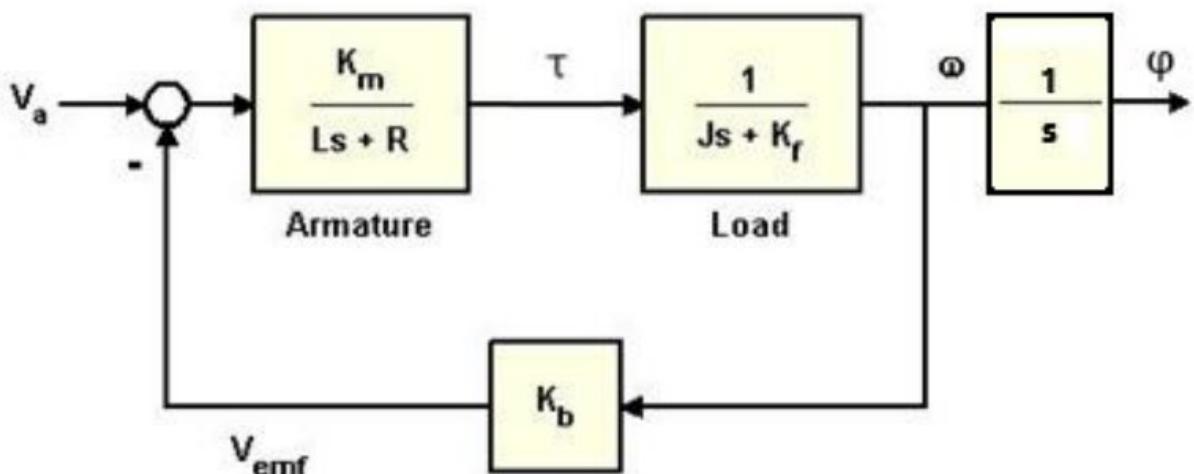
## 2. DC Motor És Enkóder Működése

### 2.1. DC Motor Működése

Az egyenáramú (DC) motor az egyik legelterjedtebb villamos hajtás, amelyet egyszerű felépítése és jó szabályozhatósága miatt széles körben alkalmaznak ipari és fogyasztói berendezésekben. Működése azon alapul, hogy az árammal átájtart forgórész-vezetők a stator mágneses terében Lorentz-erő hatására mechanikai nyomatékot hoznak létre.

A forgórész tekercseiben folyó áram mágneses teret kelt, a kommutátor pedig a forgás során úgy kapcsolja át az áramirányt, hogy a keletkező forgatónyomaték iránya állandó maradjon, ezáltal biztosítva a folyamatos forgómozgást.

A DC motorok fontos előnye, hogy sebességük és forgásirányuk egyszerűen szabályozható. A fordulatszám a motorra jutó átlagos feszültség módosításával állítható – gyakorlati megvalósításban ez pulzusszélességg-modulációval (PWM) történik. A forgásirány a tápfeszültség polaritásának megfordításával változtatható. A precíz vezérléshez motorvezérlő áramkörre van szükség, amely a motoráramot és a kapocsfeszültséget a kívánt üzemi paraméterekhez igazítja.

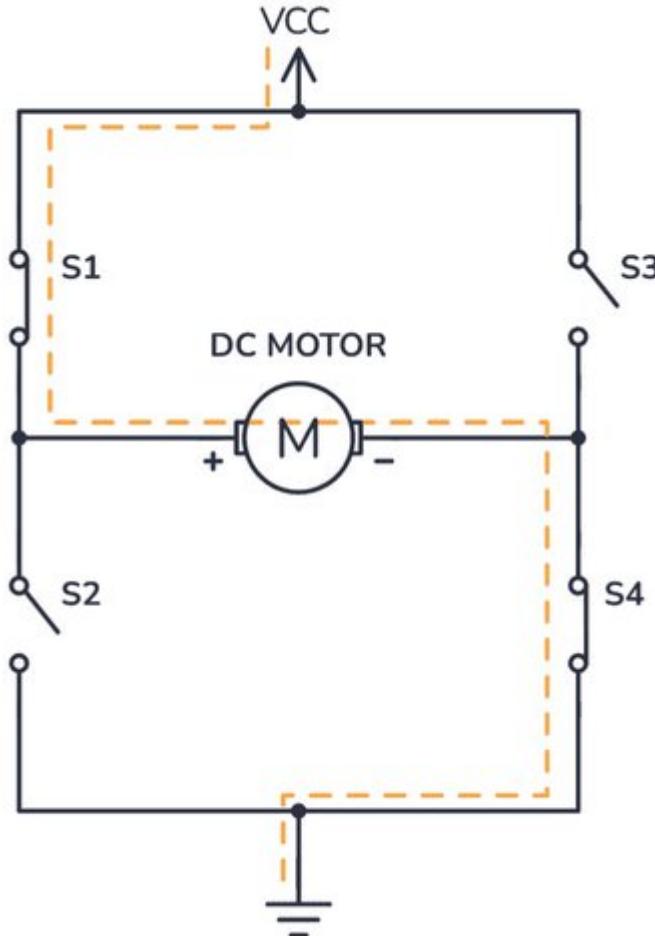


1. Ábra: DC Motor Általános Modellje

### 2.2. H-Híd Kapcsolás

A DC motorok irányításának egyik legelterjedtebb módja a H-híd (H-Bridge) áramkör. A kapcsolás négy félvezető elemből – általában tranzisztorokból vagy MOSFET-ekből – épül fel, amelyek H-alakú elrendezésben helyezkednek el. A H-híd működése az áramirány megfordításán alapul: a megfelelő két átlós kapcsoló bekapsolásával a motoron átfolyó áram iránya megváltozik, így a forgásirány szabályozható. Ha az egyik diagonális kapcsolópárt aktiváljuk, a motor az óramutató járásával megegyező irányba forog; a másik diagonális pár bekapsolásával pedig az ellentétes irányba. A két felső vagy két alsó kapcsoló egyidejű bekapsolása tilos, mert rövidzárat okozna a táp felé.

A motor fordulatszáma pulzusszélesség-modulációval (PWM) szabályozható: a kitöltési tényező módosításával az átlagos feszültség változik, így a motor sebessége finoman állítható.



2. Ábra: H-híd kapcsolás

### 2.3. DC motor matematikai modellje

A szimulációban a DC motor viselkedését differenciálegyenletekkel írjuk le, amelyek az elektromos és mechanikai alrendszert modellezik.

$$H(s) = \frac{\Phi(s)}{V_{app}(s)} = \frac{K_m}{s[(Js + K_f)(Ls + R) + K_b K_m]}$$

$v_{app}(t)$  - kapocsfeszültség

$\tau(t)$  - forgató nyomaték.

$i(t)$  - a forgórész árama

$\omega(t)$  - A forgórész szögsebessége.

$L$  - a forgórész induktivitása

$K_f$  - súrlódási együttható.

$R$  - a forgórész ellenállása

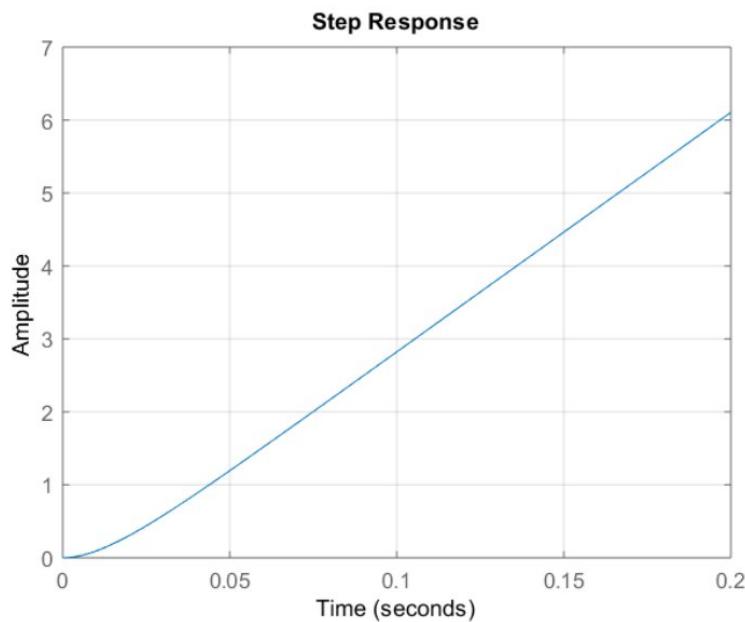
$J$  - tehetetlenségi nyomaték

$K_b$  - elektromos állandó

$K_m$  - motorállandó

### Rendszer Matlab Szimulációja

```
s = tf('s');  
J=3.2E-6;  
Kf=3.5E-6;  
Km=0.03;  
Kb=Km;  
R=4; L=3E-6;  
H1 = tf(Km,[L R]);  
H2 = tf(1,[J Kf]);  
P_motor = Km/(s*(J*s+Kf)*(L*s+R)+Km*Kb));  
step(P_motor,0.2)  
grid;
```



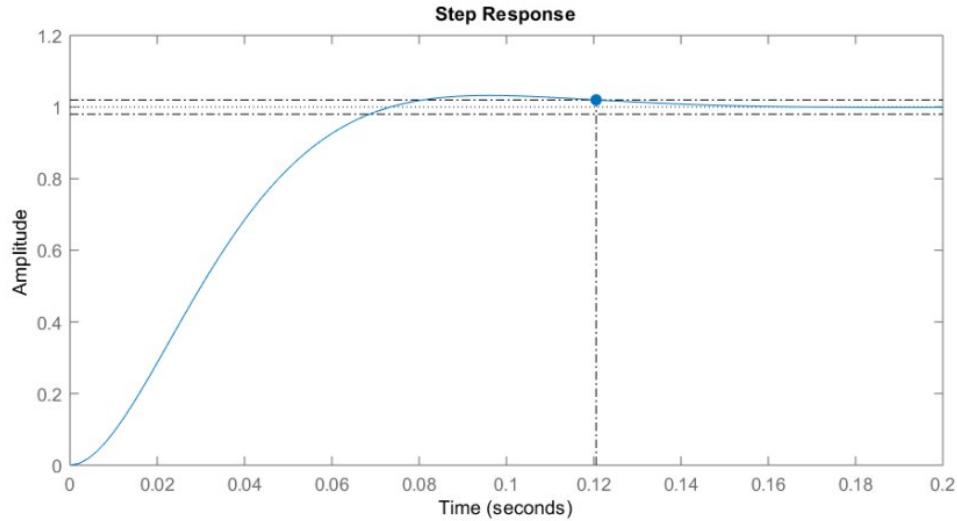
3 Ábra: DC motor rendszer szimulációja

Mivel a rendszer integráló jellegű, nem stabil, egységugrásra adott válasza egy folyamatosan növekvő értékű jel, mely összhangban van azzal, hogyha egy egyenáramú motorra egy fix feszültséget kapcsolunk, a tengely elfordulási szöge folyamatosan nőni fog.

### Mereven Vissza csatolt rendszer

`sys_cl = feedback(P_motor,1)`

`step(sys_cl,0.2)`



4. Ábra: DC motor vissza csatolt rendszer szimulációja

damp(sys_cl)			
Pole	Damping	Frequency (rad/seconds)	Time Constant (seconds)
-3.57e+01 + 3.27e+01i	7.37e-01	4.84e+01	2.80e-02
-3.57e+01 - 3.27e+01i	7.37e-01	4.84e+01	2.80e-02
-1.33e+06	1.00e+00	1.33e+06	7.50e-07

5. Ábra: Rendszer tulajdonságai

A zárt rendszer egy domináns komplex konjugált gyökpárral rendelkezik, melyek időállandója 28 ms, illetve egy nagyon gyors, nagyon kicsi (0.75 us) időállandójú valós gyökkel.

## 2.4. Enkóder Működése

A rotációs enkóder a tengely szögelfordulását digitális impuluszjelekké alakítja. Működése optikai vagy mágneses érzékelésen alapul: a forgó réselt (vagy mágnesesen jelölt) tárcsa elhaladásakor minden jelölés egy impulzust hoz létre. Az enkóder felbontását a PPR (pulses per revolution) érték adja meg. Egy 1000 PPR-es inkrementális enkóder  $0,36^\circ/\text{impulzus}$  szögfelbontást biztosít, amely négyzetezett számlálással (A és B csatorna fel- és lefutó élei) akár  $0,09^\circ$ -ra is finomítható.

A folytonos szög diszkrét impulzusokká történő alakítása kvantálási hibát eredményez, amely tipikusan  $\pm 0,5$  impulzusnyi. Az A és B fázisjelek  $90^\circ$ -os fázistolása lehetővé teszi a forgásirány meghatározását. Az enkóder jelei pozíció-visszacsatolásként szolgálnak, így zárt hurkú szabályozás valósítható meg.

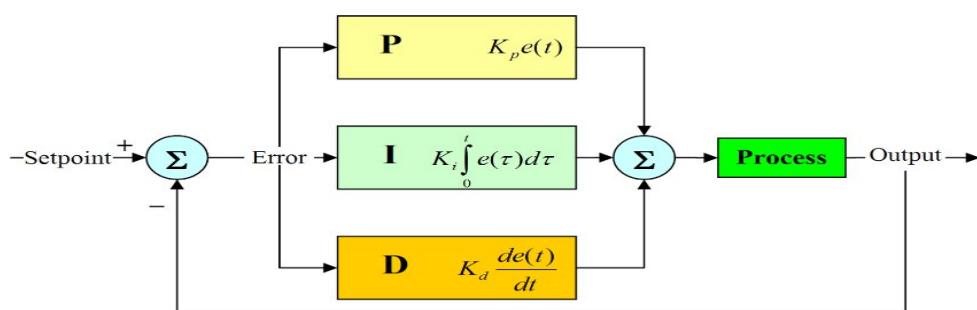


6. Ábra: DC Motor felszerelt enkóderrel

### 3. PID Irányítás

A PID (Proporcionális-Integráló-Deriváló) szabályozó az iparban az egyik legelterjedtebb visszacsatolt irányítási algoritmus. Egyszerű matematikai felépítése ellenére rendkívül hatékony eszköz a dinamikus rendszerek viselkedésének befolyásolására és stabilizálására. A DC motorok szabályozásában a PID algoritmust gyakran alkalmazzák a pozíció, a sebesség vagy a nyomaték pontos beállítására, mivel képes kompenzálni a rendszer hibáit, külső zavarásait és nemlineárításait.

#### 3.1. PID felépítése



7 Ábra. – PID Szabályzó általános modelje

##### Proporcionális (P) tag:

A proporcionális tag a pillanatnyi hibával arányos beavatkozást hoz létre: minél nagyobb a hiba, annál nagyobb a kimeneti jel. A P tag gyors reakciót biztosít és javítja a rendszer dinamikáját, azonban önmagában nem képes a maradó hiba teljes megszüntetésére. Túl nagy erősítés esetén a szabályozott rendszer instabillá válhat, és oszcilláció léphet fel.

##### Integráló (I) tag:

A integráló tag a hiba időbeli összegét veszi figyelembe. Fő szerepe a tartós, kis hibák (maradó hiba) eltüntetése, amelyeket a P tag nem képes önmagában kompenzálni. Hátránya, hogy túl nagy integráló erősség a rendszer lassúvá válását, túllandulést és akár instabilitást is okozhat.

##### Deriváló (D) tag:

A deriváló tag a hiba változási sebességére reagál, vagyis a hiba deriváltját használja a beavatkozás meghatározásához. A D tag kvázi „előrejelzi” a rendszer várható viselkedését, csökkenti a túllandulést és növeli a stabilitást. Ugyanakkor érzékeny a mérési zajra, ami hamis vagy ugráló derivált értékeket eredményezhet, ezért gyakran szűréssel együtt alkalmazzák.

##### Teljes PID egyenlet

A három tag összegzésével kapjuk a beavatkozójelet:

$$u_{PID}(t) = K_p e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de(t)}{dt}$$

**PID Átviteli függvénye:**

$$C_{PID}(s) = K_p + K_I \frac{1}{s} + K_D s = A_p \left( 1 + \frac{1}{T_I s} + T_D s \right) = \frac{A_p}{T_I} \frac{T_I T_D s^2 + T_I s + 1}{s}$$

### 3.2. Valós PID irányítás

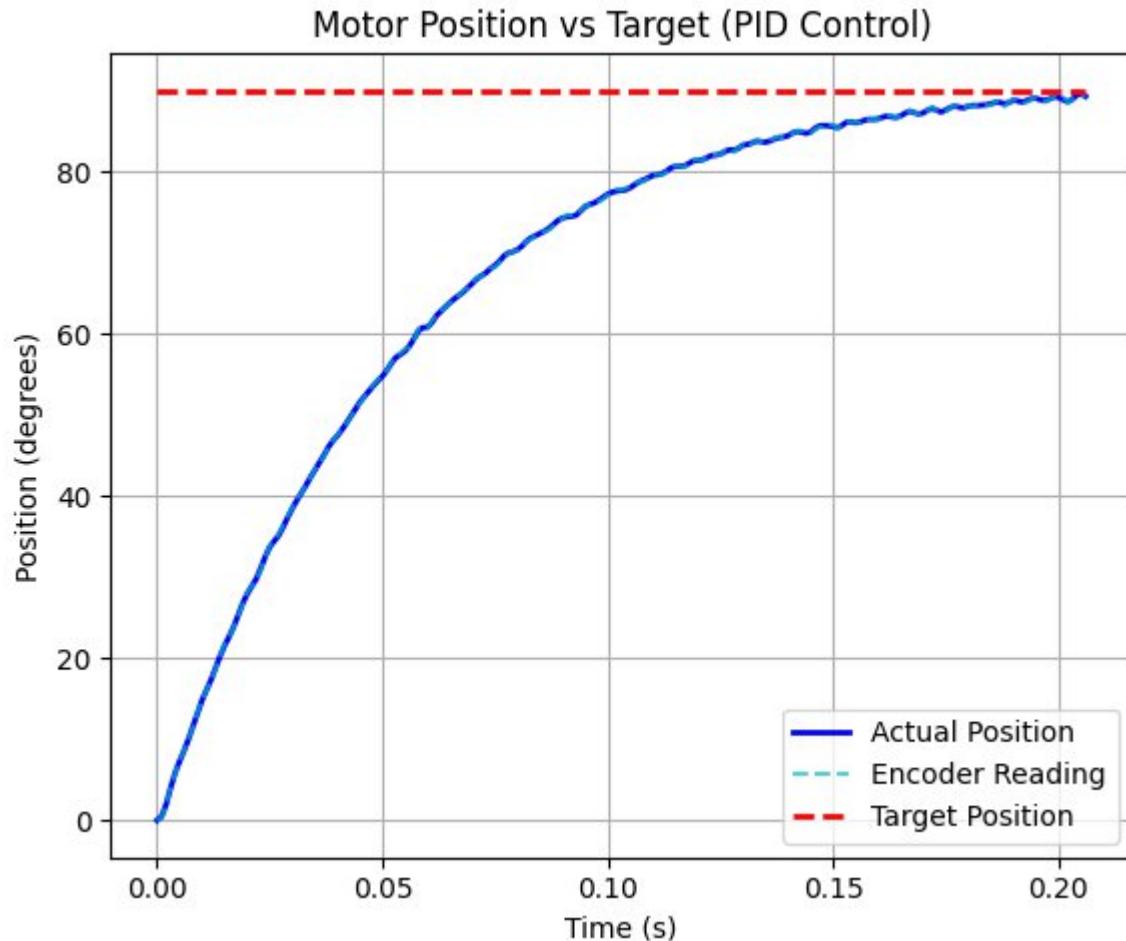
Az ideális PID szabályozó nem realizálható, a valós vagy közelítő PID szabályozóba egy plusz pólust kell beiktatnunk T időállandóval.

**Valós PID megvalósítás**

$$\hat{C}_{PID}(s) = A_p \left( 1 + \frac{1}{T_I s} + \frac{T_D s}{Ts + 1} \right) = \frac{A_p}{T_I} \frac{T_I(T_D + T)s^2 + (T_I + T)s + 1}{s(Ts + 1)}$$

$$T = \frac{T_D}{N}$$

ahol N a pólusáthelyezési arány.



8 Ábra. – PID Szabályzó ki menete aperiodikus jel esetén

## 4. Rendszer hardveres felépítése

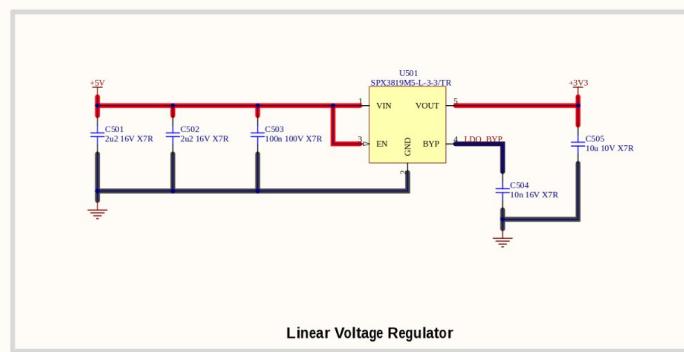
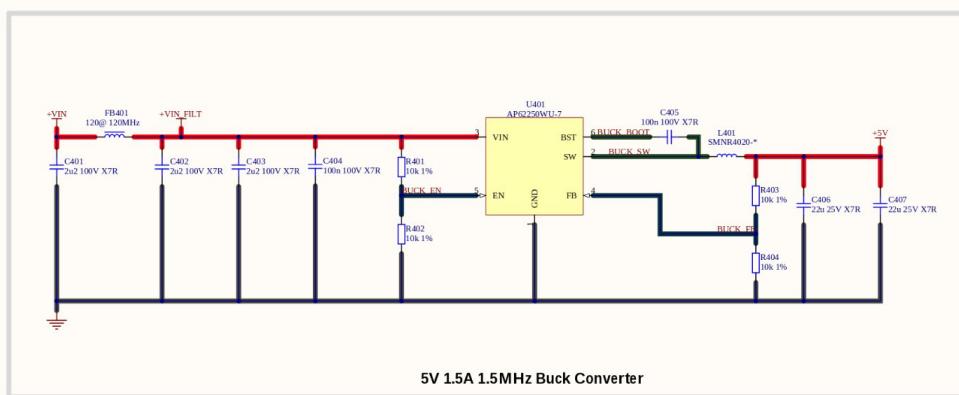
A rendszer hardverének tervezése Altium Designer környezetben történt. A PCB felépítése a következő főbb jellemzőkkel rendelkezik:

### 4.1. Tápellátás

A PCB 10–18V közötti bemeneti feszültséget fogad. A tápellátás kétlépcsős felépítésű: az első fokozatban egy Diodes Incorporated AP62250WU-7 szinkron buck konverter a bemeneti feszültségből 5V-ot állít elő, legfeljebb 1,5A terheléssel, 1,5 MHz-es kapcsolási frekvencia mellett. Az integrált szinkron kapcsolóelemekként diódát nem igényelnek, a magas kapcsolási frekvencia pedig kompakt szűrőelemek alkalmazását teszi lehetővé. Az IC TSOT-26 tokozásban kerül beépítésre. Az 5V-os sín táplálja a DRV8876 motorhajtó IC logikai részét és az USB interfészét.

A második fokozatban egy MaxLinear SPX3819M5-L-3-3 alacsony zajszintű LDO stabilizátor az 5V-os sínből 3,3V-os tápfeszültséget állít elő az ESP32-S3 mikrovezárló és a 3,3V-os perifériák számára, maximálisan 500mA kimeneti árammal. Az LDO alacsony nyugalmi áramfelvételle (90 µA) és SOT-23-5 tokozása kompakt elhelyezést biztosít.

A motorhajtás számára a bemeneti feszültség közvetlenül, a DRV8876 H-híd meghajtó IC-n keresztül kerül a motorra, amely maximálisan 3A terhelőáramot képes kezelní.



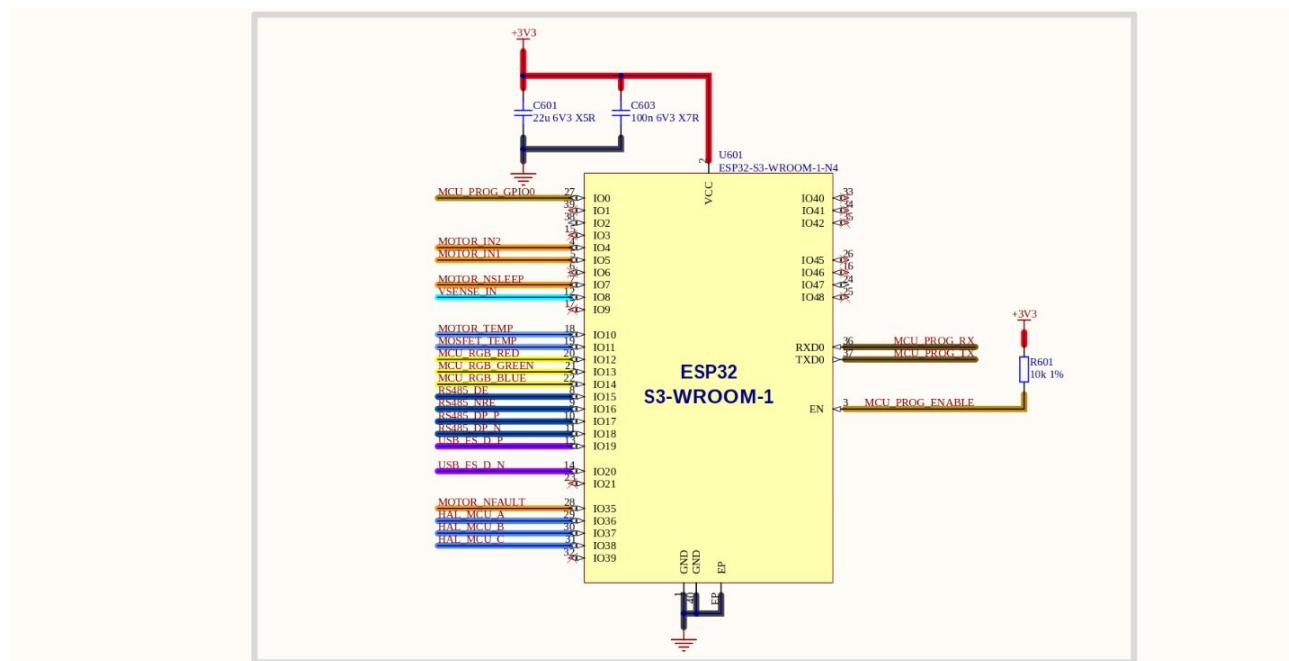
9 Ábra. – Elektronika tápelátására szolgáló áramkörök

## 4.2. Mikrovezérlő

A rendszer központi vezérlőegysége az Espressif ESP32-S3 rendszer-a-chipen (SoC) mikrovezérlő, amely két darab Xtensa LX7 processzormagot tartalmaz, legfeljebb 240 MHz-es órajelen. A chip 512 KB beépített SRAM-mal rendelkezik, valamint külső SPI flash és PSRAM memória illesztését is támogatja. A jelen alkalmazás szempontjából az ESP32-S3 az alábbi perifériáit használjuk ki:

- PCNT (Pulse Counter): hardveres impulzusszámláló modul, amely az enkóder A és B kvadratúra jeleit külön beavatkozás nélkül, hardverszinten dolgozza fel. A modul önállóan képes a felfutó és lefutó élek számlálására, beépített zajszűrővel (glitch filter) rendelkezik, és a GPIO mátrixon keresztül tetszőleges lábra konfigurálható. Ez a megoldás a szoftveres megszakítás-alapú számlálással szemben pontos és CPU-terhelés nélküli pozícióérzékelést biztosít.
- LEDC (LED PWM Controller): nyolccsatornás, konfigurálható frekvenciájú és felbontású PWM generátor, amelyet a DRV8876 motorhajtó EN bemenetének vezérlésére használunk. A jelen rendszerben 20 kHz-es frekvencián, 10 bites felbontással működik, ami 1024 fokozatú sebességszabályozást tesz lehetővé.
- USB OTG: a chip beépített USB 2.0 Full-Speed tranceiverrel rendelkezik, amely CDC-ACM (virtuális soros port) üzemmódban biztosítja a kommunikációt a felügyeleti rendszer felé, külső USB-UART átalakító IC nélkül.

A chip 3,3V-os tápfeszültségről működik, mélyszunyadási módban fogyasztása akár 7 µA-re csökkenhető.



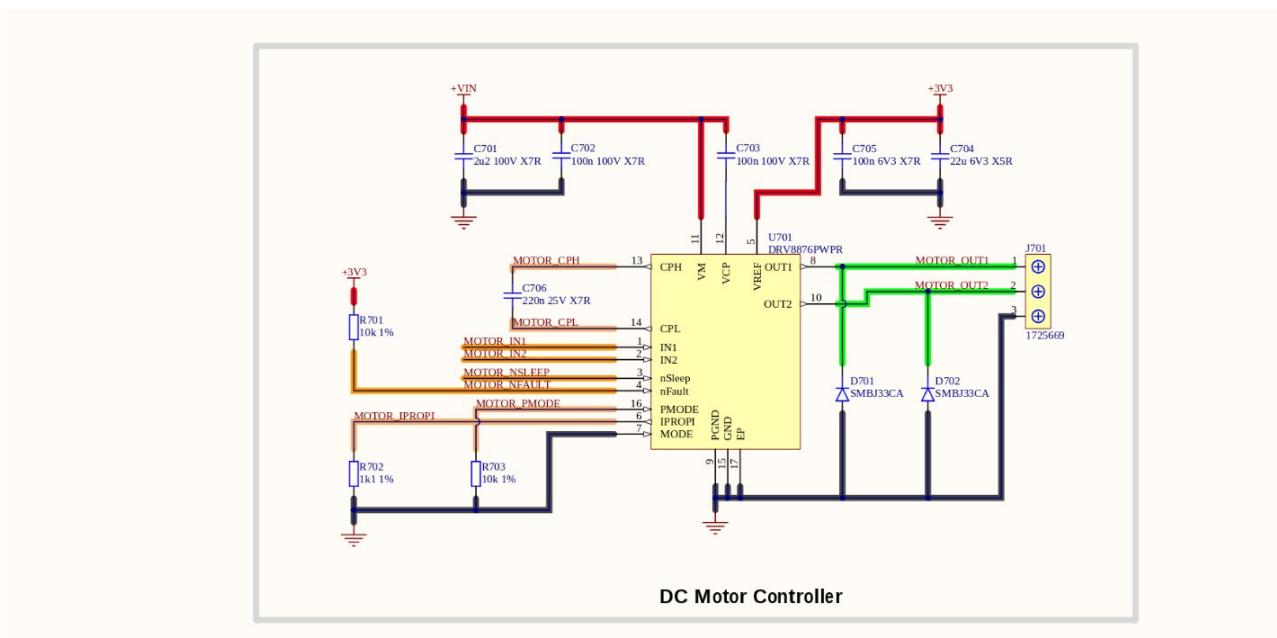
10 Ábra. – ESP32-S3 Modul

### 4.3. DC motor irányítása

A rendszerben a DRV8876PWPR motorvezérlő IC-t használjuk, amely a PWM és PH pinjein keresztül vezérli a DC motor működését. Az IC egy teljes híd kapcsolású vezérlő, amely lehetővé teszi a motor sebességének és irányának finomhangolását.

- PWM vezérlés (EN pin): Az EN (Enable) pin a PWM jelet fogadja, amely szabályozza a motor sebességét. A PWM jelet a mikrokontroller generálja, és ezen keresztül szabályozható a motor sebessége az impulzusok szélességének változtatásával. Minél magasabb a PWM ciklus, annál nagyobb a feszültség átengedése a motorra, így nagyobb sebességet eredményez.
- Forgásirány (PH pin): A PH (Phase) pin a motor forgásirányát szabályozza. A PH pin magas vagy alacsony szintre állítása határozza meg, hogy a motor előre vagy hátra forog-e. Ha a PH pin magas, a motor előre, ha pedig alacsony, akkor hátra fog forogni. Így a vezérlő rendszer képes a motor forgásirányának dinamikus változtatására, miközben a PWM a sebességet szabályozza.

Ezen felül az IC beépített védelmi mechanizmusokkal is rendelkezik, így biztosítva a rendszer megbízható működését hosszú távon.



11 Ábra. – DRV8876 Motor Áramköre

#### 4.4. Analóg mérőáramkörök és jelkondicionálás

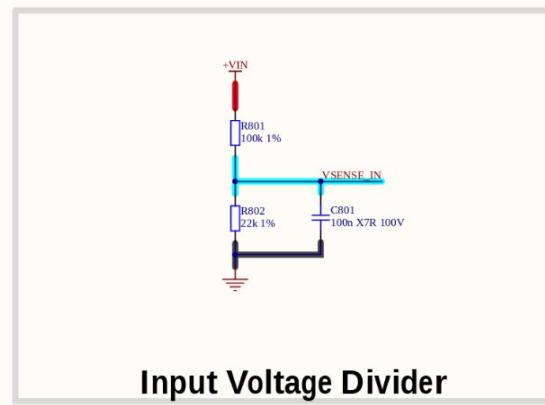
A hardver három analóg bemeneti csatornát tartalmaz, amelyek az ESP32-S3 beépített ADC-jére csatlakoznak. Mindhárom csatornán a jelkondicionálás célja a mérendő jel ADC-kompatibilis feszültségtartományba (0–3,3V) történő illesztése és a nagyfrekvenciás zaj kiszűrése.

##### Bemeneti tápfeszültség mérése

A 10–18V bemeneti tápfeszültség monitorozását egy rezisztív feszültségesztő valósítja meg. A felső ágban R801 = 100 kΩ, az alsó ágban R802 = 22 kΩ ellenállás található, az osztópont és a föld között pedig egy 100 nF kerámia kondenzátor biztosítja a nagyfrekvenciás zajszűrést.

$$V_{out} = V_{in} * \frac{R802}{R802 + R801}$$

10V-os bemenetnél 1,80V, a 18V-os bemenetnél 3,25V jelenik meg az ADC bemenetén, ami a teljes üzemi tartományban a 0–3,3V-os mérési sávon belül marad. A firmware az ismert osztási arány alapján visszaszámítja a valós bemeneti feszültséget.



12 Ábra. – Bementi feszültség mérése

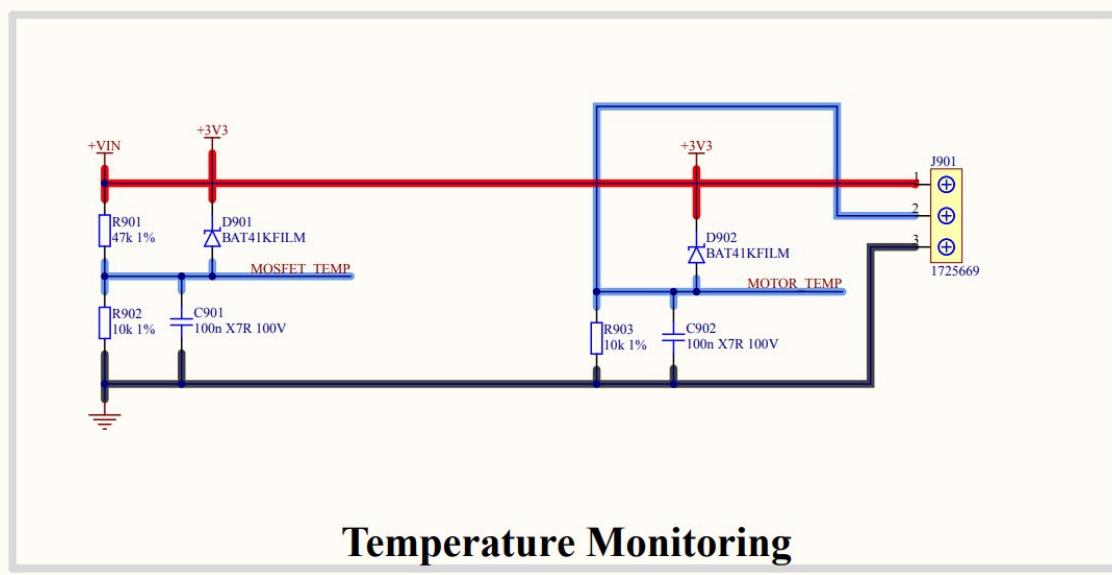
### Hőmérsékletmérés NTC termisztorral

A rendszer hőmérsékletét egy  $47 \text{ k}\Omega$  névleges ellenállású ( $25^\circ\text{C}$ -on) NTC termisztor méri. Az NTC (R901) a bemeneti tápfeszültségre csatlakozik felső ágként, az alsó ágban egy  $R902 = 10 \text{ k}\Omega$  ellenállás található a föld felé, az osztópontot pedig  $100 \text{ nF}$  kondenzátor szűri. Az NTC ellenállása a hőmérséklet emelkedésével csökken, így az osztópontron megjelenő feszültség emelkedik.

$$\begin{aligned} V_{\text{out}} &= V_{\text{in}} * \frac{R902}{R_{NTC} + R902} \\ R_{NTC} &= R902 * \left( \frac{V_{\text{in}}}{V_{\text{out}}} - 1 \right) \\ \frac{1}{T_k} &= \frac{1}{T_0} + \frac{1}{B} * \ln \left( \frac{R_{NTC}}{R_0} \right) \\ T_{\text{C}} &= T_k - 273,15 \end{aligned}$$

- $T_k$  a keresett hőmérséklet Kelvinben
- $T_0 = 298,15 \text{ K}$  ( $25^\circ\text{C}$ , a referencia-hőmérséklet)
- $R_0 = R_{NTC} = 47 \text{ k}\Omega$  (az NTC névleges ellenállása  $25^\circ\text{C}$ -on)
- $B$  az NTC anyagállandója (az adatlapból olvasható ki, jellemzően  $3000\text{--}4500 \text{ K}$  közötti érték)
- $R_{NTC}$  a mért aktuális ellenállás

Magas hőmérsékleten az NTC ellenállása jelentősen lecsökkenhet, ami a bemeneti feszültségtől függően az ADC tűréshatárát meghaladó feszültséget eredményezne. Ennek megakadályozására egy BAT41KFILM Schottky dióda védi a bemenetet: anódja a mérőpontra, katódja a  $+3,3\text{V}$ -os sínrre csatlakozik. Ha a jelszint meghaladja a  $3,3\text{V}$  + a dióda nyitófeszültsége értéket, a dióda kinyit és a többletáramot a tápfeszültségre elvezeti, megvédve az ESP32-S3 ADC bemenetét a túlfeszültségtől.



13 Ábra. – Hőmérséklet mérése

#### 4.5. Pozicíció Mérése

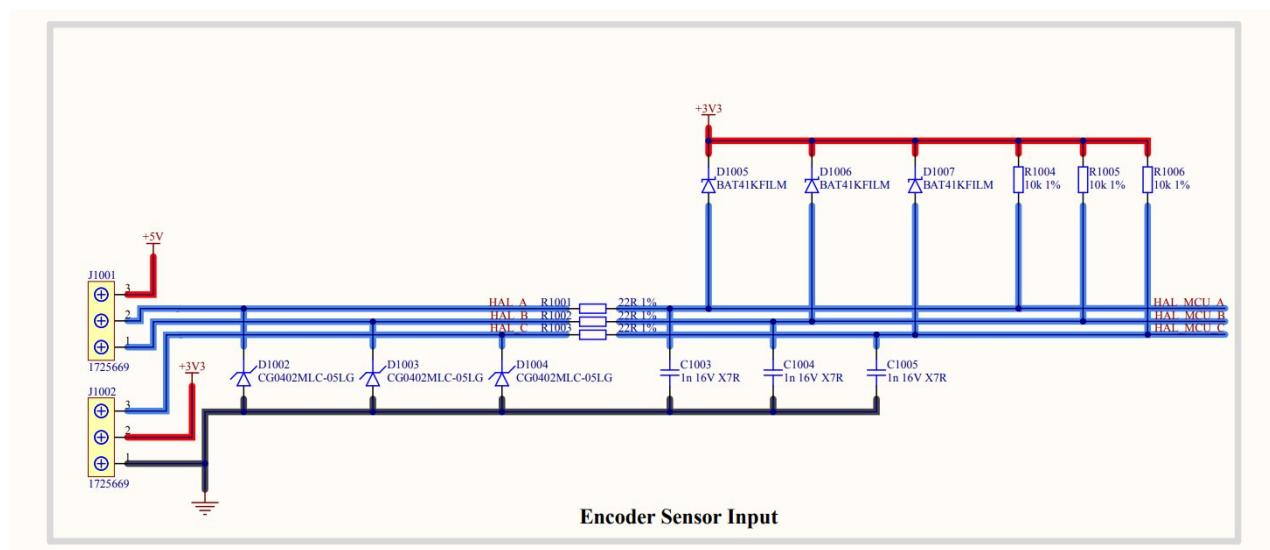
Az enkóder bemenete sorkapocson keresztül érhető el, és az A, B valamint az opcionális Z (index) csatorna fogadására alkalmas. A Z csatorna fordulatonként egyetlen impulzust ad, amely abszolút referencia-pontkeresést (homing) tesz lehetővé, így a rendszer bekapcsolás után képes meghatározni a tengely kiindulási pozícióját.

Az enkóder bemenetei ESD védelmi diódákkal vannak ellátva, amelyek az elektrosztatikus kisülésekkel eredő feszültségtükörökkel szemben védekeznek meg, megvédve az ESP32-S3 GPIO bemeneteit a károsodástól. Ez különösen fontos, mivel az enkóder jelvezetékei a külső sorkapocson keresztül csatlakoznak, így ki vannak téve a környezeti elektrosztatikus hatásoknak.

Az áramkör 3,3V-os és 5V-os jelszintű enkóderek fogadására egyaránt alkalmas. A jelszint-illesztést – az NTC mérőáramkörhöz hasonlóan – BAT41KFILM Schottky diódák biztosítják: anódjuk a jelvezetékre, katódjuk a +3,3V-os sínre csatlakozik. 5V-os enkóder használata esetén a dióda a 3,3V + V<sub>f</sub> szint felett kinyit, és a többletfeszültséget a 3,3V-os tápra vezeti el, így az ESP32-S3 bemenete nem kap káros túlfeszültséget. 3,3V-os enkóder esetén a dióda zárt állapotban marad, és nem befolyásolja a jelet.

A bemeneti vonalakon gyenge felhúzó (pull-up) ellenállások találhatók a 3,3V-os sín felé. Ezek biztosítják, hogy nyitott drain (open-drain) kimenetű enkóderek esetén is stabil logikai szint alakuljon ki a jelvezetéken. Az ilyen típusú enkóderek kimenete csak föld felé húz, a magas szintet a felhúzó ellenállás állítja elő.

Minden csatornán egy  $R = 22 \Omega$  soros ellenállás és egy 100 nF föld felé kapcsolt kondenzátor RC aluláteresztő szűrőt alkot, amelynek vágási frekvenciája körülbelül 72,3 kHz. Ez a szűrő a nagyfrekvenciás zajt és az élek körüli csengést (ringing) csillapítja, miközben az enkóder hasznos jelét még magas fordulatszámon is veszteségmentesen átengedi. A kis értékű soros ellenállás az ESD védiódákkal és Schottky feszültségkorlátozó diódákkal együttműködve áramkorlátozó szerepet is betölt.



14 Ábra. – Enkóder bemenet

#### 4.6. RS485 interfész

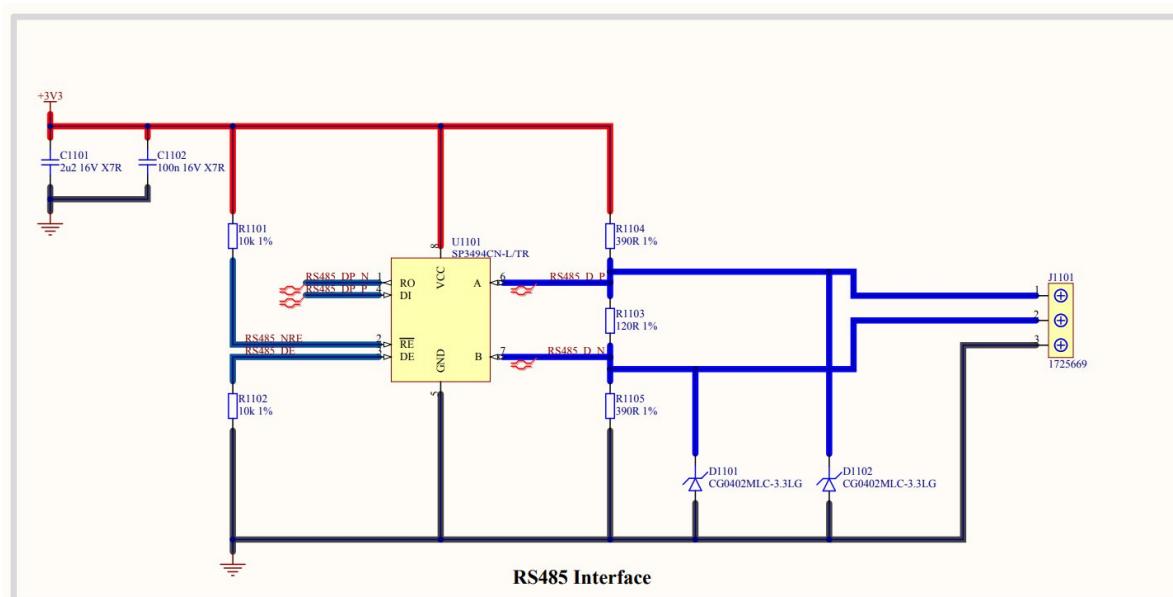
RS-485 félvezetékes (half-duplex) kommunikációs interfészt is tartalmaz, amely ipari környezetben elterjedt, zajálló differenciális soros kommunikációt biztosít. Az interfész lehetővé teszi a motorvezérlő integrálását többeszközös buszrendszerekbe, ahol akár 256 csomópont is elhelyezhető egyetlen vezetékpáron.

Az átalakítást egy SIT3485EEA RS-485/RS-422 half-duplex transceiver végzi SOIC-8 tokozásban. Az IC 3V-5,5V tápfeszültség-tartományban működik, legfeljebb 12 Mbps adatátviteli sebességet támogat, és beépített  $\pm 15$  kV ESD védelemmel rendelkezik. Az alacsony, 750  $\mu$ A nyugalmi áramfelvétel energiahatékony működést biztosít.

A transceiver UART oldala az ESP32-S3 egyik UART perifériájára csatlakozik. Az adásirány-vezérlő lábak alapállapota úgy van konfigurálva, hogy a DE (Driver Enable) láb lehúzó ellenállással alacsony szinten, az NRE (Receiver Enable, aktív alacsony) láb pedig felhúzó ellenállással magas szinten áll. Ezzel az IC alapállapotban sem adás, sem vétel módban nem aktív – minden funkciót a firmware-nek explicit módon kell engedélyeznie a kommunikáció megkezdésekor, ami megakadályozza a busz nem kívánt meghajtását inicializálás előtt.

A differenciális A és B jelvezetékek a következő áramköri elemekkel vannak ellátva:

- 120  $\Omega$  lezáró ellenállás (termináció): az A és B vonalak között elhelyezett ellenállás a busz jellemző impedanciájához illeszti a végpontot, csökkentve a jelvisszaverődéseket. Hosszabb vezetékhossz és magasabb adatsebesség esetén a termináció elengedhetetlen a megbízható kommunikációhoz.
- 390  $\Omega$  felhúzó és lehúzó ellenállások (bias): az A vonalra a tápfeszültségre kapcsolt 390  $\Omega$  felhúzó, a B vonalra a föld felé kapcsolt 390  $\Omega$  lehúzó ellenállás biztosítja, hogy a busz üresjáratú állapotában (amikor egyetlen adó sem aktív) a differenciális feszültség meghatározott, érvényes logikai szinten maradjon. Ez megakadályozza a hamis adatvételt és a busz lebegő állapotát.
- ESD védelem: a csatlakozónál ESD védelmi elemek találhatók, amelyek a külső kábelezésen érkező elektrosztatikus impulzusokat a transceiver beépített védelmén túl további védelmi szintet biztosítva elvezetik.

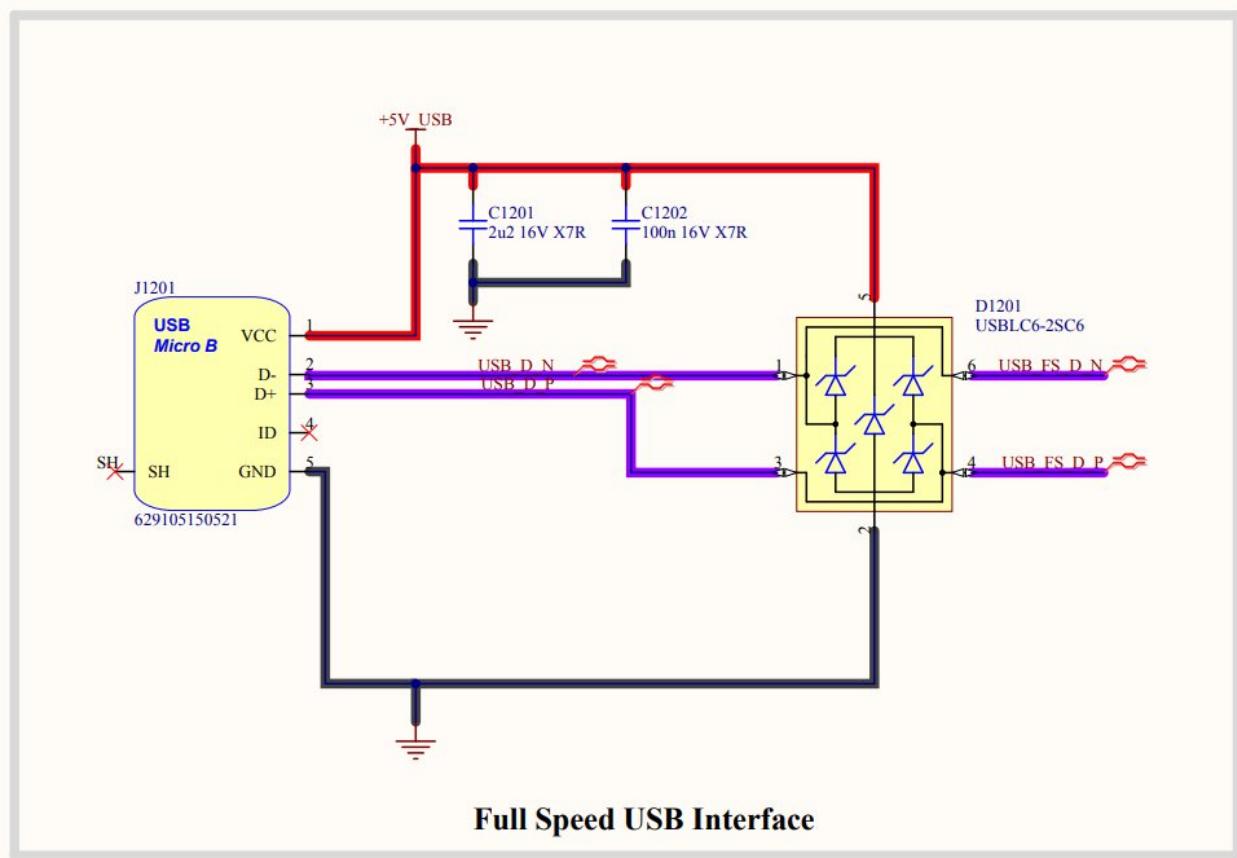


15 Ábra. – RS485 Interfész

#### 4.7. Számítogépes interfész és programozás

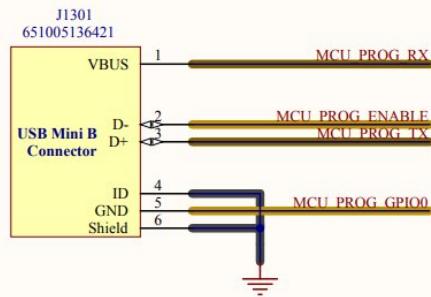
A PC és az ESP32-S3 MCU közötti összeköttetést egy Micro USB csatlakozó biztosítja, USB 2.0 Full Speed (12 Mbps) üzemmódban. Az USB szabvány szerinti D+ és D- differenciális jelvezetékek 90 Ω differenciális impedanciával kerültek kialakításra a nyomtatott áramköri hordozón. Az impedanciaértéket a PCB rétegfelépítése, a dielektrikum tulajdonságai alapján számítottuk ki, biztosítva a jelintegritást és az USB specifikációnak való megfelelést a teljes jelúton.

Az USB csatlakozó és az ESP32-S3 USB adó-vevője közé egy USBLC6-2SC6 ESD védelmi IC került beépítésre. Az IC a D+ és D- vonalakat védi a csatlakozón keresztül érkező elektrosztatikus kisülések től, kiegészítve az MCU saját belső védelemét. Az USBLC6-2SC6 alacsony parazita kapacitásának köszönhetően a jelintegritást nem rontja, miközben az IEC 61000-4-2 szabvány szerinti ESD védelmet nyújt a buszvezetékeken.



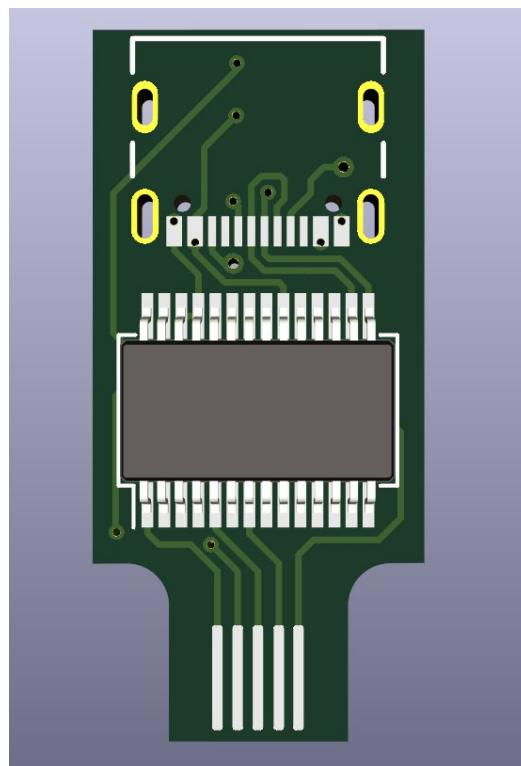
16 Ábra. – USB 2.0 kapcsolat

A eszköz egy saját fejlesztésű, FTDI chipen alapuló programozó áramkör lehet programozni, amely egy Mini USB-B csatlakozón keresztül csatlakoztatható. Ez az interfész biztosítja az ESP32-S3 soros programozását és debug kommunikációját a fejlesztői számítógép felől, a fő USB porttól függetlenül.



USB Programming Interface

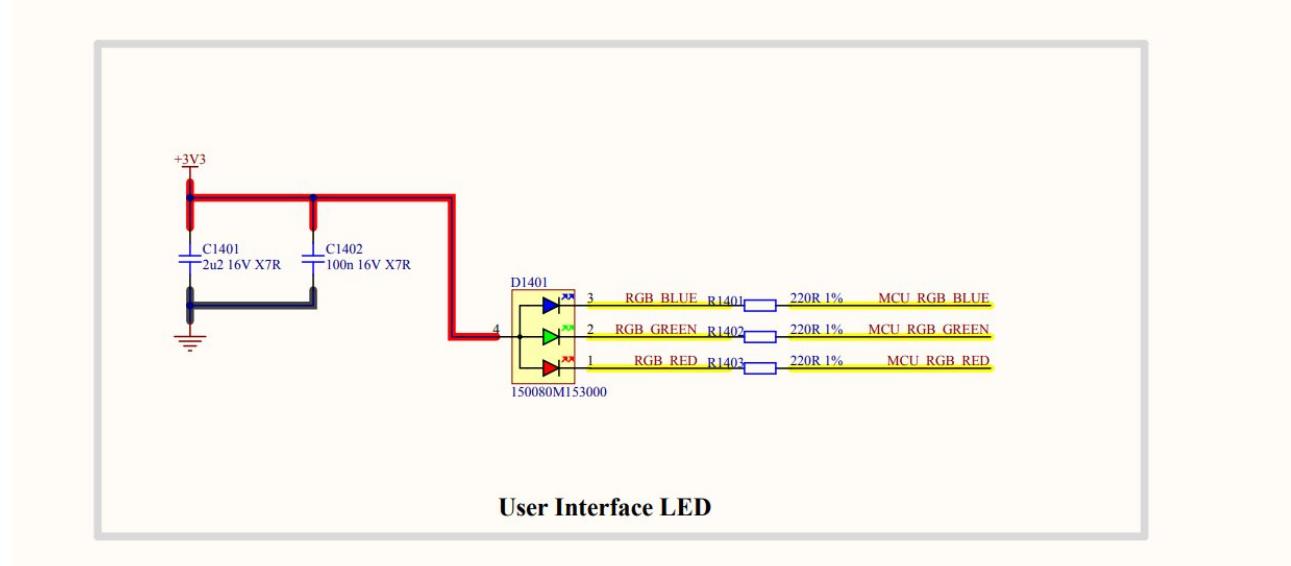
17 Ábra. – Programozó interfész



18 Ábra. – Programozó Áramkör

#### 4.8. RGB Led

Az áramkör egy közös anódos RGB LED-et tartalmaz állapotjelzés céljából. A LED három színcsatornáját (piros, zöld, kék) az ESP32-S3 három független LEDC PWM timer csatornája vezérli, lehetővé téve az egyes színkomponensek fényerejének önálló, szoftveres szabályozását és tetszőleges színkeverést. Az egyes színcsatornákon egy-egy áramkorlátozó ellenállás biztosítja a LED névleges üzemi áramát és védi mind a LED-et, mind az MCU GPIO kimeneteit a túláramtól.



19 Ábra. – RGB Led áramkör

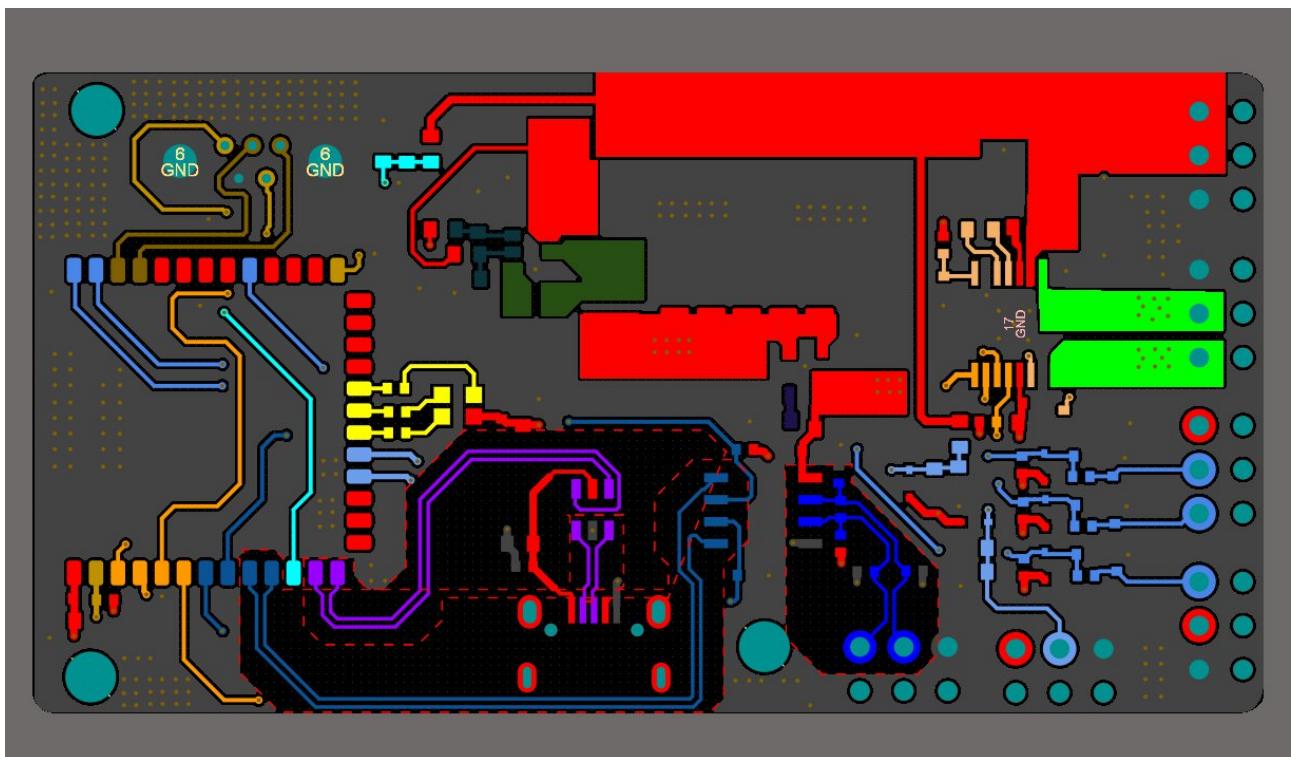
#### 4.9. Nyomtatott áramköri lap kialakítása

A nyomtatott áramköri hordozó 4 rétegű felépítéssel készült, az alábbi rétegsorrenddel (stackup):

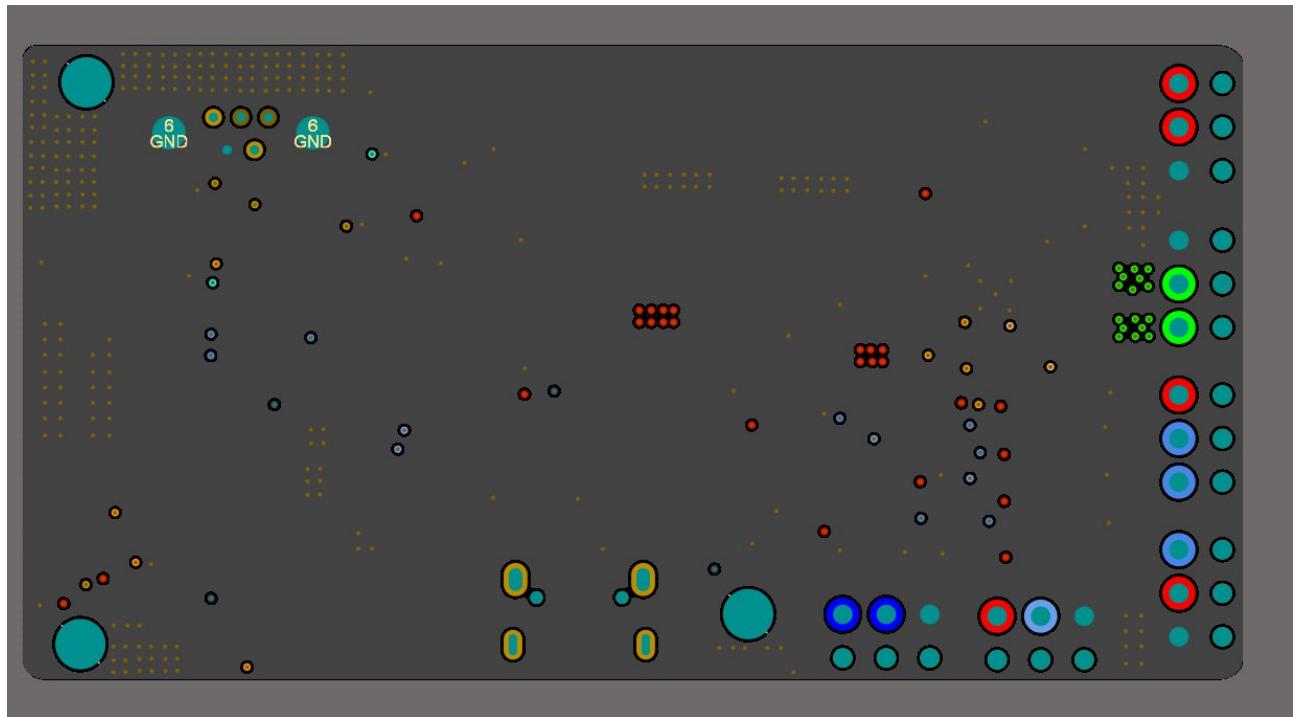
Réteg	Funkció
L1	SIG/PWR
L2	GND
L3	PWR
L4	SIG/PWR

A belső L2 réteg folytonos földsíkként szolgál, amely referenciajelkötőket biztosít a felső jelréteg vezetékei számára. Ennek köszönhetően a kritikus jelvezetékek visszatérési áramútja rövid és jól meghatározott, ami alacsony hurok induktivitást és kiszámítható impedanciakörnyezetet eredményez. Az L3 réteg a tápellátás síkok és kiegészítő földterületek elhelyezésére szolgál, hozzájárulva a tápellátás szűréséhez és a rétegek közötti kapacitív csatlás kihasználásához.

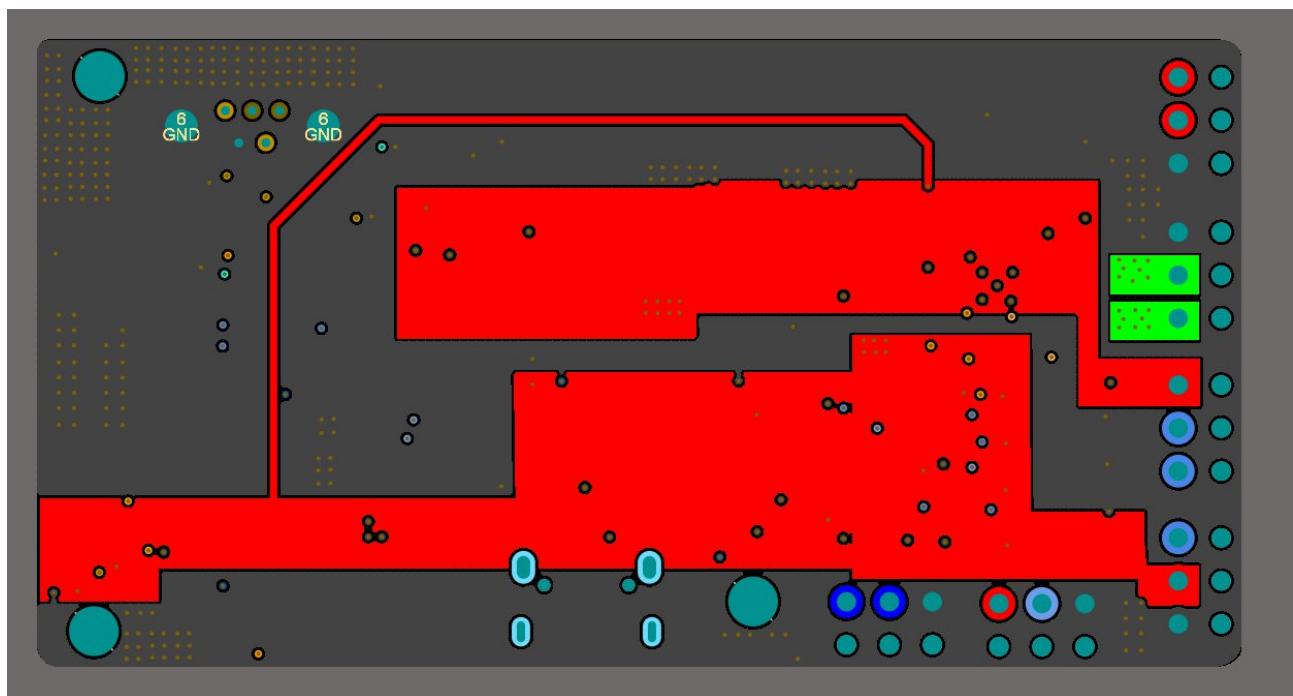
A tervezés során impedanciaillesztett differenciális jelvezetékek kerültek alkalmazásra: az USB D+/D- vonalak 90 Ω, az RS-485 A/B vonalak pedig 120 Ω differenciális impedanciára lettek méretezve, a rétegefélépítés és a hordozó dielektrikus tulajdonságai alapján. A kritikus jelútvonalak elrendezésénél a hurok induktivitás minimalizálása volt az elsődleges szempont, csökkentve ezzel a vezetett és sugárzott zajkibocsátást. A folytonos belső földsík, a rövid visszatérési utak és az impedanciaillesztett vonalvezetés együttesen kedvező EMI teljesítményt biztosítanak.



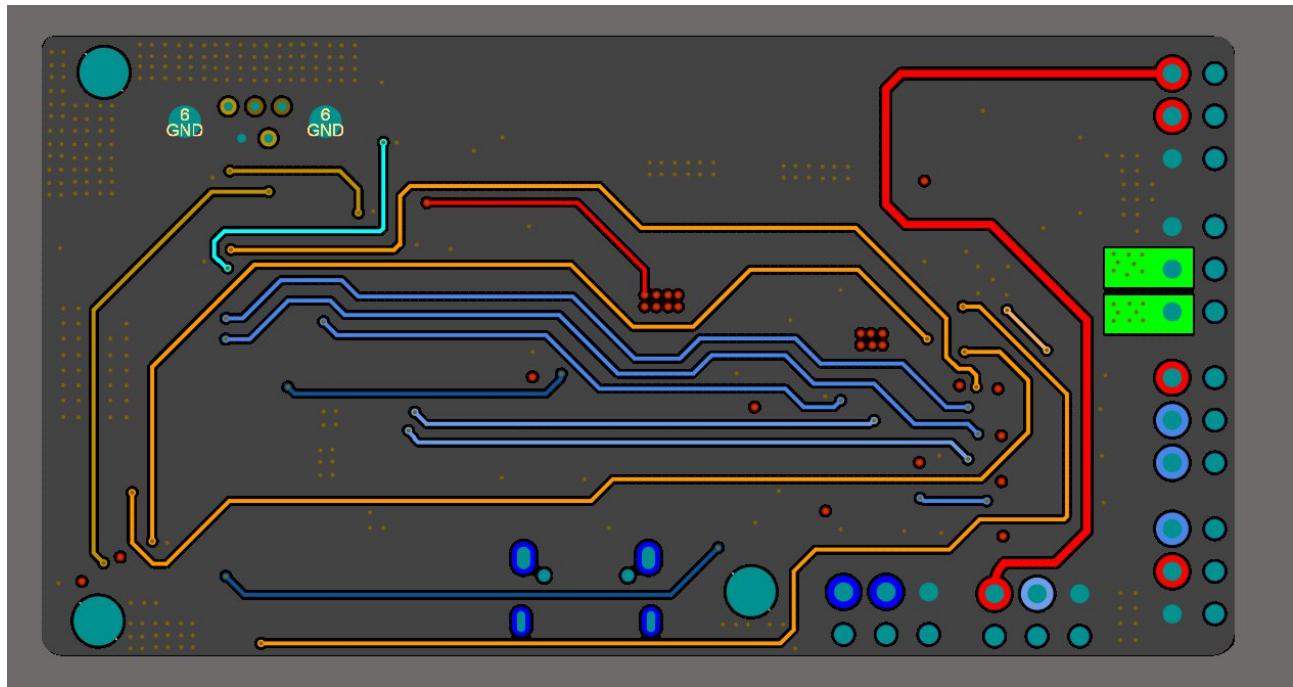
20 Ábra. – L1 réteg



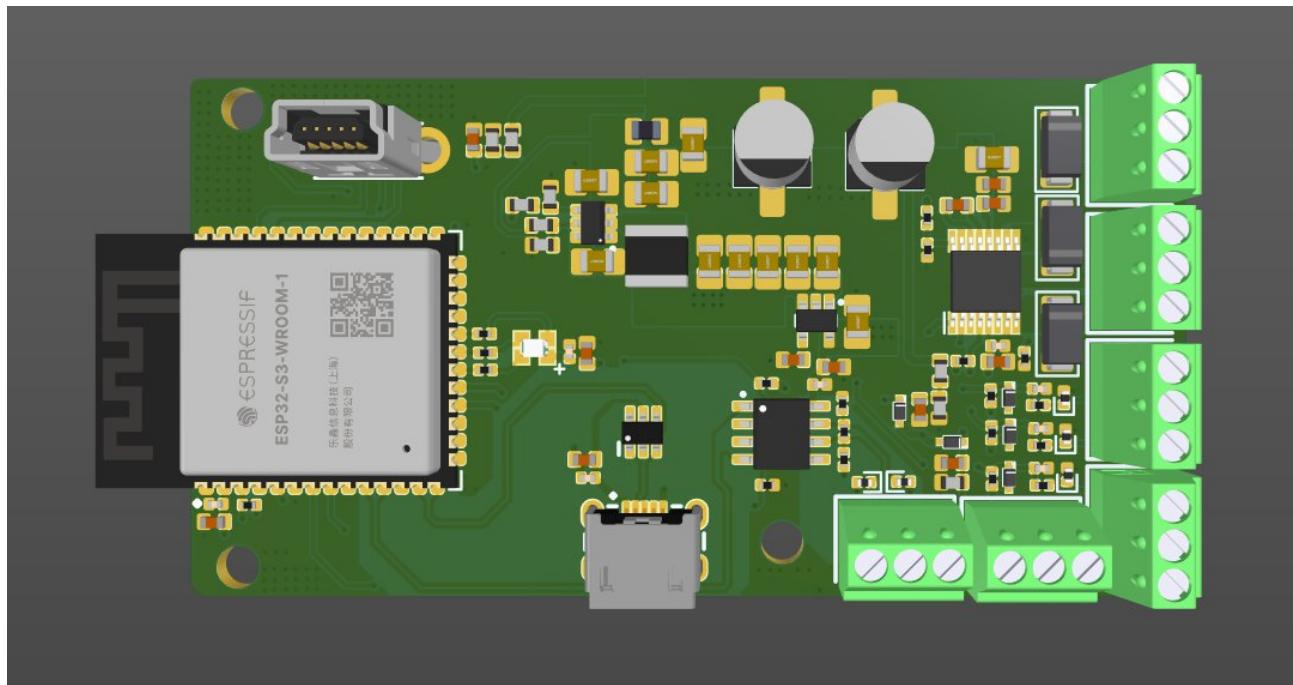
21 Ábra. – L2 réteg



22 Ábra. – L3 réteg



23 Ábra. – L4 réteg



24 Ábra. – Hardver 3D nézete

## 5. A firmware architektúrája és működése

### 5.1. Áttekintés

A firmware moduláris, rétegzett architektúrával rendelkezik, amely az ESP-IDF keretrendszer komponens-alapú projektstruktúráját követi. A tervezés alapelve az interfész-alapú absztrakció: minden hardverközeli és algoritmikus modul egy tisztán virtuális (abstract) C++ interfészt valósít meg, amely elválasztja a felhasználói logikát a konkrét hardvermegvalósítástól. Ez lehetővé teszi az egyes komponensek független tesztelését, cseréjét és újrafelhasználását.

### 5.2. Perfériák

A firmware az ESP32-S3 következő perfériáit használja:

Periféria	Felhasználás	GPIO/Csatorna
LEDC PWM	Motor sebességszabályozás (DRV8876 EN)	Channel 0, GPIO 5, 20 kHz, 10 bit
LEDC PWM	RGB LED vezérlés (3 csatorna)	Channel 1-3
GPIO kimenet	Motor irányváltás (DRV8876 PH)	GPIO 4
GPIO kimenet	Motor meghajtó alvás/ébresztés (nSLEEP)	GPIO 7
GPIO kimenet	Motor meghajtó hibajel (nFAULT)	GPIO 6
PCNT	Kvadratúra enkóder pozíciósámlálás	GPIO 2 (A), GPIO 1 (B)
USB CDC-ACM	Soros kommunikáció (TinyUSB)	Beépített USB
ESP Timer	RPM számítás, időmérés	-

### 5.3. FreeRTOS taskok

A firmware két fő FreeRTOS taskot hoz létre:

#### 1. MotorControlTask (prioritás: 10, stack: 4096 B, maghoz nem kötött)

- 100 Hz-es ciklusban fut (vTaskDelayUntil)
- Enkóder pozíció olvasás (PCNT)
- PID szabályozó számítás
- Mozgásprofil generálás (trapéz/S-görbe)
- Lágy pozíciókorlát ellenőrzés
- Sebesség becslés
- Beállás- és elakadás-detekció
- Motor parancs kiadás (PWM kitöltési tényező + irány)

## 2. MotorCommTask (prioritás: 10, stack: 4096 B, maghoz nem kötött)

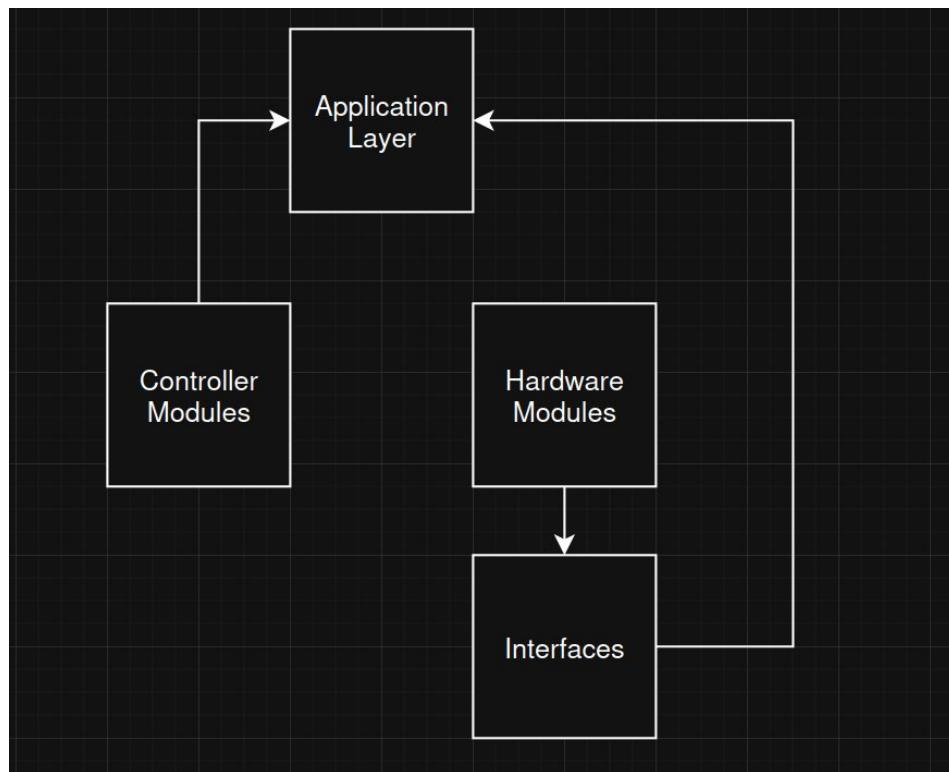
- 100 Hz-es ciklusban fut (vTaskDelay)
- USB parancsok fogadása és értelmezése (SET\_DEG, SET\_PID, GET\_PID, STOP, ENABLE, DISABLE)
- Célpozíció és PID paraméterek továbbítása a MotorControlTask felé
- Visszajelzés küldése a PC felé (pozíció, MOTOR\_REACHED, PID értékek)

A két task között mutex (targetMutex, configMutex) és spinlock biztosítja a szálbiztonságot.

### 5.4. Architektúra

Az architektúra négy fő rétegből épül fel:

Réteg	Komponensek	Szerep
Interfészek	IMotorDriver, IEncoder, IPIDController, IComm, IMotorController, IMotorCommHandler, IRGBLed	Absztrakt osztályok
Hardver meghajtók	DRV8876, Encoder, USB, RGBLed	Perifériakezelés
Szabályozási modulok	IDController, MotionProfiler, SettleDetector, StallDetector, SoftLimiter, MotionGuard	Zárt hurkú algoritmusok
Alkalmazási réteg	MotorController, MotorCommHandler	Összeállítás és vezérlés



25 Ábra. – Firmware architektúra

## 5.5. Modulok

### 5.5.1. Hardver meghajtó réteg

A DRV8876 osztály az IMotorDriver interfészt valósítja meg, és négy belső alkomponensből áll:

- PwmController – Az ESP32-S3 LEDC perifériáját kezeli: timer és csatorna inicializálás, kitöltési tényező beállítás mutex-védett hozzáféréssel.
- GpioController – A PH (irány) és nSLEEP (alvás) GPIO kimenetek vezérlése.
- FaultHandler – Az nFAULT bemenet negatív ére triggerelt megszakítás-kezelője. Az ISR atomi flag-et állít, amelyet a feladatkontextusban dolgoz fel a rendszer, elkerülve a megszakításon belüli hosszú feldolgozást.
- MotionRamp – Hardverfüggetlen rámpázási logika, amely fokozatos sebességváltoztatást biztosít irányváltásnál. Callback mintát használ, így nem függ közvetlenül a motormeghajtótól.

#### Enkóder

Az Encoder osztály az IEncoder interfészt valósítja meg. Az ESP32-S3 PCNT hardveres impulzusszámlálóját használja a kvadratúra jelek dekódolásához, szoftveres megszakítás nélkül. Két belső alrendszert tartalmaz:

- SpeedEstimator – Hibrid fordulatszám-becslő, amely két módszert ötvöz: magas fordulatszámon az impulzusszám-különbség módszert, alacsony fordulatszámon az élperiódus-mérés módszert alkalmazza. A két érték közötti átmenet lineáris interpolációval történik egy konfigurálható küszöb körül, majd EMA vagy IIR aluláteresztő szűrő simítja az eredményt.
- DirectionDetector – Debounce-olt irányérzékelő hiszterézissel: megköveteli, hogy a pozícióváltozás meghaladjon egy küszöbértéket, és az irányváltás óta meghatározott idő teljen el, mielőtt a forgásirányt ténylegesen megváltoztatja.

Az enkóder támogatja a szoftveres A/B csatorna cseréjét, polaritás-invertálást és áttételi arány beállítást.

#### USB kommunikáció

Az USB osztály az IComm interfészt valósítja meg, és a TinyUSB CDC-ACM (virtuális soros port) veremre épül. Atomi flag-ek jelzik a kapcsolat állapotát, a fogadott adatokat konfigurálható méretű pufferben tárolja.

#### RGB LED

Az RGBLed osztály az IRGBLed interfészt valósítja meg. Hárrom független LEDC PWM csatornán keresztül vezérli a közös anódos RGB LED színcsatornáit. Támogat fade (fokozatos átmenet), villogás és fényerő szabályozás funkciókat.

### 5.5.2. Szabályzási Modulok

#### PID szabályozó

A PIDController osztály az IPIDController interfészt valósítja meg. A diszkrét idejű PID algoritmust a következő jellemzőkkel egészíti ki:

- Anti-windup: az integráló tag értéke konfigurálható maximumra van korlátozva
- Derivált szűrés: aluláteresztő szűrő a derivált tagon (derivativeAlpha), amely csökkenti a mérési zaj hatását
- Deriválás a mért értékből: a derivált tagot nem a hiba, hanem a mért pozíció változásából számítja, elkerülve az alapjel-ugrásokból eredő derivált-csúcsokat

#### Mozgásprofil-generátor

A MotionProfiler modul a kívánt sebességharmonikát formált profillá, két módban:

- Trapéz profil: konstans gyorsulási korláttal
- S-görbe profil: rángás-korláttal (jerk limit), amely simább átmenetet biztosít

A profiler kimenete referenciaiként szolgál a PID szabályozó és az előrecsatolás számára.

#### Beállás-érzékelő

A SettleDetector meghatározza, hogy a motor elérte-e a célpozíciót. Két feltétel együttes teljesülését figyeli konfigurálható számú egymást követő ciklusban:

- A pozícióhiba kisebb, mint a tűréshatár (posTolDeg)
- A sebesség kisebb, mint a sebességtűrés (velTolDegPerSec)

#### Elakadás-detektor

A StallDetector felismeri, ha a motor elakadt (parancsot kap, de nem mozdul). Működése:

1. Bemelegítési fázis: új parancs után a PID szabályozónak szüksége van néhány ciklusra a felépüléshez  
– ez idő alatt az elakadás-detekció szünetel
2. Elakadás-figyelés: ha a pozícióváltozás a küszöb alatt marad, miközben a hiba a minimális mozgatási határ felett van, az elakadás-számláló növekszik
3. Riasztás: a számláló elérésekor STALLED állapotot jelez

#### Lágy pozíciókorlát

A SoftLimiter szoftveres pozícióhatárokat valósít meg. Két funkciója van:

- Célpozíció korlátozás: a kívánt célpozíciót a megengedett tartományra szűkíti
- Futásidőjű blokkolás: megakadályozza a motorparancs végrehajtását, ha az a határ felé mozdítaná a motort

#### Mozgás-időkorlát

A MotionGuard két védelmi funkciót lát el:

- Időtúllépés: ha a mozgás nem fejeződik be a megadott időn belül, hibát jelez
- Drift-ébresztés: ha a motor a beállás után a holtszav-küszöböt meghaladóan elmozdul, újraindítja a szabályozási hurkot

## 5.6. Alkalmazás réteg

### MotorController

A MotorController osztály az IMotorController interfészt valósítja meg, és kompozíció útján egyesíti az összes modult. Saját FreeRTOS taskban fut (MotorControlTask, 100 Hz), és minden ciklusban a következő lépéseket hajtja végre:

1. Enkóder pozíció és sebesség olvasás
2. Pozícióhiba számítás
3. Lágy korlát ellenőrzés
4. Mozgásprofil alkalmazás
5. PID kimenet számítás
6. Előrecsatolás hozzáadás (feed forward)
7. Motorparancs korlátozás és kiadás
8. Beállás-, elakadás- és időkorlát-detekció
9. Esemény-visszahívások (callback-ek) meghívása

### MotorCommHandler

A MotorCommHandler osztály az IMotorCommHandler interfészt valósítja meg. Saját FreeRTOS taskban fut (MotorCommTask, 100 Hz), és szöveges protokollon keresztül kommunikál az USB soros interfészen:

#### Fogadott parancsok:

- SET\_DEG:<fok> – célpozíció beállítása
- SET\_PID:<kp>:<ki>:<kd> – PID paraméterek hangolása
- GET\_PID – aktuális PID értékek lekérdezése
- STOP – motor leállítás
- ENABLE / DISABLE – motor engedélyezés/tiltás
- RESET\_ALL – állapotjelzők törlése

#### Küldött üzenetek:

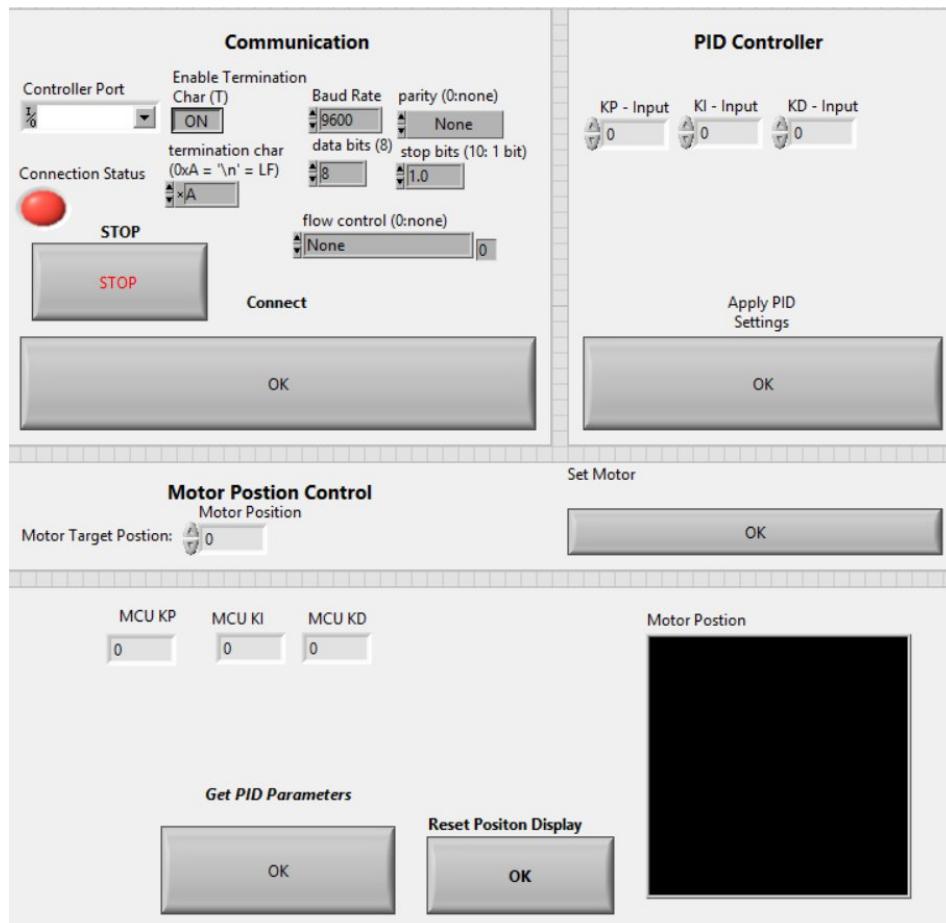
- PID:<kp>,<ki>,<kd> – PID paraméterek válasz
- MOTOR\_REACHED – motor beállt a célpozícióra

## 6. LabView Felhasználói interfész

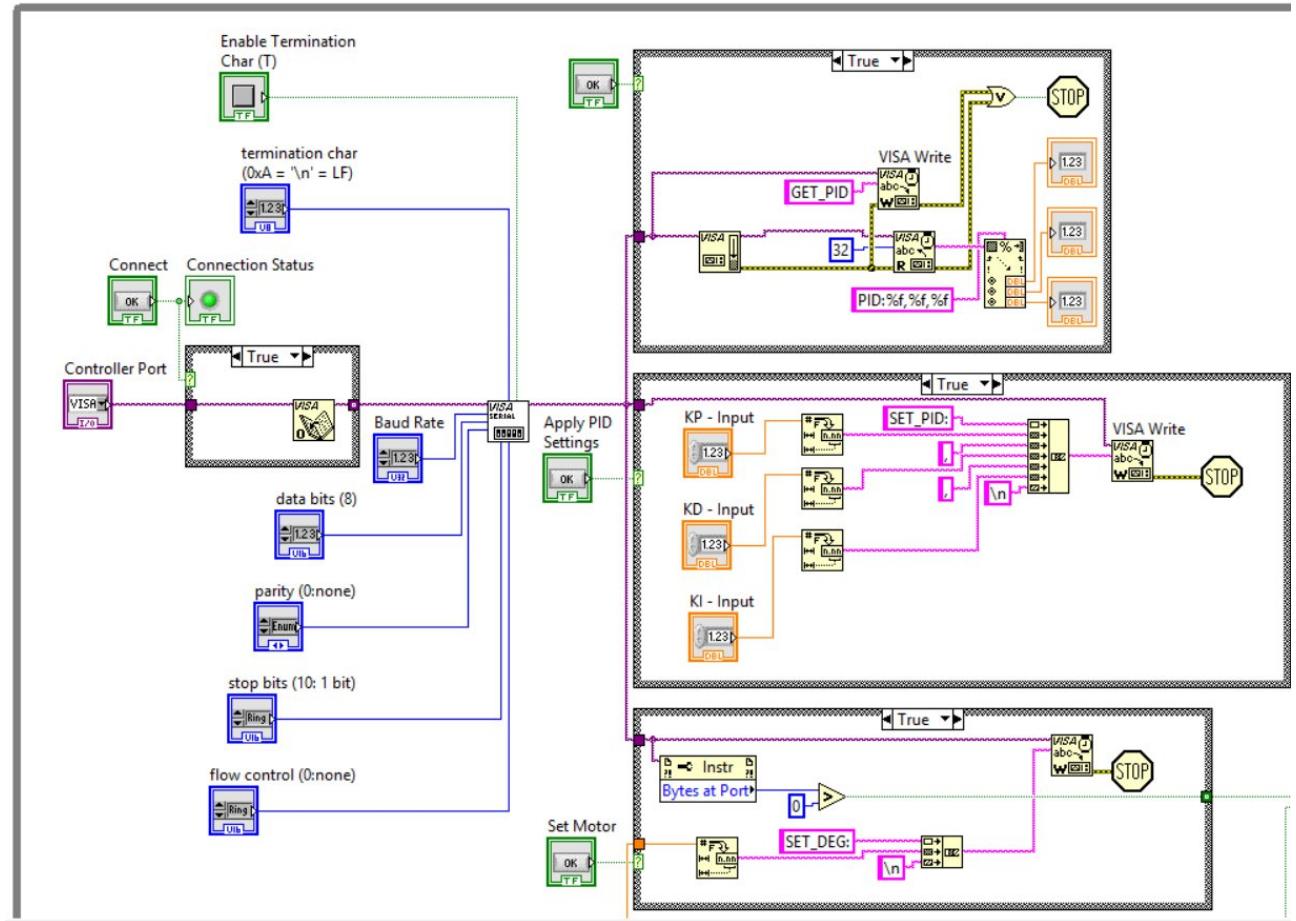
A **LabVIEW SCADA környezet** a rendszer központi vezérlőeleme, amely lehetővé teszi a motor vezérlését és az összes szükséges paraméter beállítását a felhasználó által. A rendszer célja, hogy egy könnyen használható, intuitív és hatékony felületet biztosítson a vezérléshez, monitorozáshoz és hibakereséshez.

A **LabVIEW** szerepe kulcsfontosságú, mivel:

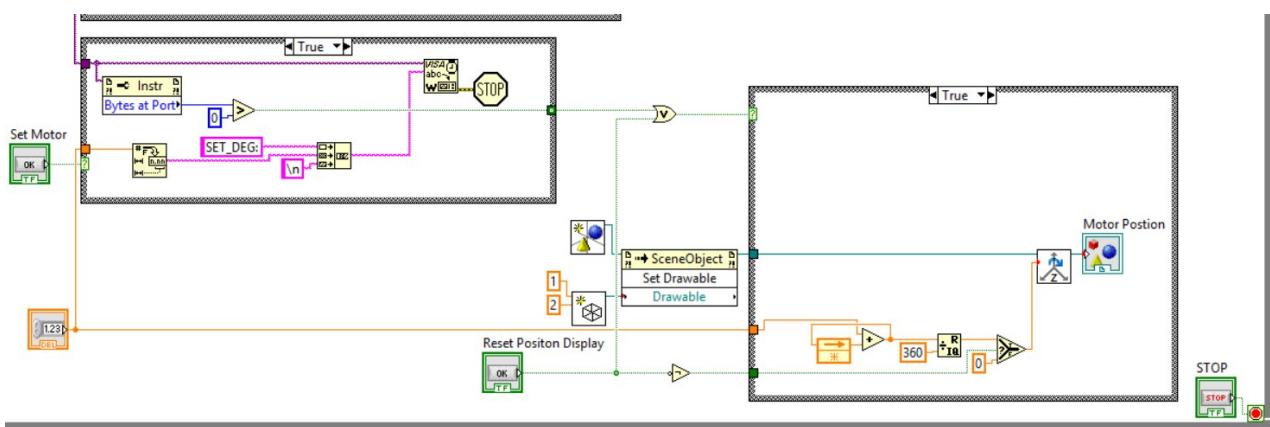
- Felhasználói felület biztosítja:** Grafikus eszközök (pl. gombok, diagramok) segítségével a felhasználó könnyelmesen állíthatja be a kívánt paramétereket, mint a motor pozíciója és a PID paraméterek.
- Valós idejű adatkommunikáció:** A rendszer figyeli és frissíti a motor aktuális pozíóját, valamint a vezérlési paramétereit. A LabVIEW és az ESP32-S3 mikrokontroller közötti kétirányú kommunikáció biztosítja, hogy minden adat szinkronban legyen.
- Rendszer irányítása:** A motor beállításainak módosítása, a PID paraméterek szabályozása és a motor állapotának lekérdezése minden a LabVIEW alkalmazásban történik, amely megjeleníti a szükséges adatokat és lehetővé teszi a rendszer finomhangolását.



26 Ábra. – LabView Felhasználói Inteface



27. Ábra - LabView alkalmazás blokk diagramja



28. Ábra - LabView alkalmazás blokk diagramja

## 6.1. Visa Kommunikációs Modul

A Visa kommunikációs modul alapvető szerepet játszik a LabVIEW SCADA környezet és az ESP32-S3 mikrokontroller közötti adatátvitelben. A Visa (Virtual Instrument Software Architecture) egy szabványos protokoll, amely lehetővé teszi a két eszköz közötti adatcsérét, és biztosítja a kétirányú kommunikációt. A Visa kommunikációs modul felelős a vezérlési és adatfeldolgozási feladatokért, mint például a PID paraméterek küldése, a motor pozíciók lekérése és a rendszer visszajelzéseinek kezeléséért. Az alábbiakban bemutatjuk, hogyan történik a kommunikációs kapcsolat létesítése és adatcsere.

### Kommunikációs kapcsolat létesítése a LabVIEW és az ESP32-S3 mikrokontroller között

#### Port kiválasztása és megnyitása (Visa Open):

- Az első lépés a kommunikációs port kiválasztása. Ehhez a LabVIEW felhasználói felületén létrehoztunk egy vezérlőt a main panel-en, amely lehetővé teszi a felhasználó számára, hogy kiválassza az elérhető portot (pl. COM port vagy USB).
- A felhasználó a kívánt portot kiválasztva a Visa Open blokk segítségével megnyitja a kapcsolatot a mikrokontrollerrel. A Visa Open blokk biztosítja, hogy a megfelelő portot használja a rendszer, így a kommunikáció folytatódhat.

#### Port konfigurálása (Visa Configure Serial Port):

- Miután a portot sikeresen megnyitottuk, be kell állítani a kommunikációs paramétereket. Ehhez a Visa Configure Serial Port blokkot használjuk, amely lehetővé teszi a port beállításait, mint például a baud rate, paritás, adatbitek, stop bitek, és egyéb kommunikációs paraméterek.
- A Visa Configure Serial Port blokk biztosítja, hogy a rendszer megfelelő módon konfigurálja a portot, és felkészíti a kommunikációra.

#### Adatok írása (Visa Write):

- A következő lépés a vezérlési parancsok és paraméterek elküldése az ESP32-S3 mikrokontroller felé. Ehhez a Visa Write blokkot használjuk, amely lehetővé teszi, hogy a LabVIEW alkalmazás egy string formájában küldjön adatokat a mikrokontrollernek.
- Ha az írás nem sikerül (például ha a port nem elérhető vagy hibás az adatátvitel), a program leáll, és hibaüzenetet küld a felhasználónak.

#### Adatok olvasása (Visa Read):

- A következő lépés az adatvisszajelzés olvasása a mikrokontrollertől. Ehhez a Visa Read blokkot használjuk, amely az ESP32-S3 mikrokontroller által küldött válaszokat olvassa be.
- A válaszok formátuma szintén string, amelyet a Visa Read blokk olvas be. Az olvasott adatokat ezután feldolgozzuk és megjelenítjük a felhasználói felületen, például a motor pozíóját, hibakódokat, vagy egyéb visszajelzéseket.
- Ha az olvasás nem sikerül (például ha nincs válasz a mikrokontrollertől), a program befejeződik, és a felhasználó tájékoztatást kap a hibáról.

## 6.2. 3D vizualizáció LabVIEW-ban

A LabVIEW-ban létrehozott 3D megjelenítő modul célja, hogy szemléletesen vizualizálja a motor elmozdulását, különösen a tengely körüli forgását. A megvalósítás a LabVIEW beépített 3D Picture vezérlőjére épül, és az alábbi főbb blokkok felhasználásával történik:

### Create Object VI

- Ez a blokk felelős a 3D objektum példányosításáért. Lehetőséget biztosít az objektum elnevezésére, valamint a későbbi vizualizációhoz szükséges Scene Object Node-ra történő csatlakoztatásra.

### Create Box VI

- Ezzel a blokkal hozható létre maga a geometriai elem (téglatest), amely a motor tengelyének elfordulását reprezentálja. A blokk lehetőséget biztosít a box méreteinek meghatározására (X, Y, Z irányban), valamint a színének konfigurálására is. Az objektumot szintén a Scene Object Node-hoz kell csatlakoztatni.

### Scene Object Property Node

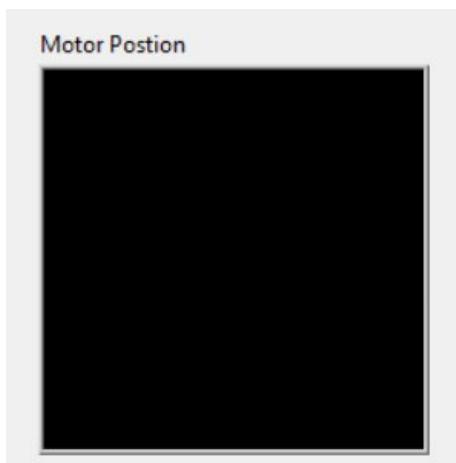
- Ez a node összeköti a Create Object és Create Box blokkok által létrehozott elemeket a 3D megjelenítő rendszerrel. Ezáltal létrejön a megjeleníthető 3D jelenet, amelyen a grafikus elemek interaktívan frissíthetők.

### Rotate Z Axis

- A létrehozott objektum elforgatása ezzel a blokkal valósítható meg. Az Angle bemeneti paraméter segítségével a test a Z tengely mentén forgatható el, amely a motor elmozdulását reprezentálja valós időben.

### 3D Picture

- Ez maga a megjelenítő felület, amely a Scene Object Property Node által összerakott jelenetet rendereli. A frontpanelen jeleníti meg az objektumok térbeli állapotát, így biztosítva a felhasználó számára a vizuális visszacsatolást a motor mozgásáról.



29. Ábra - 3D modul front panelen

### 6.3. PID paraméterek beállítása és lekérése

A PID paraméterek beállítása és lekérése a LabVIEW SCADA rendszerének egyik kulcsfontosságú funkciója. A felhasználó a felületen keresztül könnyen módosíthatja a motor vezérléséhez szükséges PID paramétereket ( $K_p$ ,  $K_i$ ,  $K_d$ ), és a rendszer lehetővé teszi ezek lekérdezését is a vezérlő mikrokontrollerből.

A PID paraméterek küldése a következő lépésekben történik:

#### PID lekérése

- Ha a felhasználó rákattint a PID lekérése gombra, a rendszer aktivál egy Case Structure blokkot. A gomb megnyomása azt jelzi, hogy a felhasználó szeretné lekérni a mikrokontroller aktuális PID paramétereit.
- A LabVIEW egy constans string üzenetet küld a rendszernek, amelyben a GET\_PID parancs szerepel. Ezt az üzenetet a Visa Write blokk segítségével küldjük el a mikrokontrollerhez.

#### Mikrokontroller válasza:

- Miután a GET\_PID üzenet elérte a mikrokontrollert, az ESP32-S3 visszaküldi a PID paramétereket az alábbi formátumban: **PID:%f,%f,%f**. Itt a három %f jelzi a három PID paramétert ( $K_p$ ,  $K_i$ ,  $K_d$ ), amelyek a válaszban szerepelnek.

#### Adatok olvasása (Visa Read):

- Miután a GET\_PID üzenetet elküldtük, a Visa Read blokk segítségével fogadjuk az ESP32-S3 válaszát. A válasz maximálisan 32 byte hosszú, mivel az üzenet csak három float értéket tartalmaz.
- A Visa Read blokkban beállítjuk, hogy a maximális byte szám 32 legyen, hogy biztosítsuk, hogy csak a válaszunk teljes formátuma kerüljön feldolgozásra.

#### String feldolgozása (Scan From String):

- A válasz feldolgozása érdekében egy Scan From String blokkot használunk. Ennek a blokkja a következő formátumot várja: **PID:%f,%f,%f**.
- A Scan From String blokkot úgy konfiguráljuk, hogy az megfelelően olvassa be a három float értéket, és szétbontja azokat a három különböző paraméterre. Ezek a paraméterek a következő értékekre lesznek hozzárendelve:
  - $K_p$
  - $K_i$
  - $K_d$

#### PID paraméterek megjelenítése:

- Miután a Scan From String blokk sikeresen feldolgozta a választ, a három float értéket hozzárendeljük három indikátorhoz a felhasználói felületen.
- A felhasználó így azonnal láthatja a mikrokontroller által visszaadott aktuális PID paramétereket.

## PID paraméterek küldése

### Felhasználói interakció:

- A felhasználó rákattint egy gombra a LabVIEW felületen, amely aktiválja a Case Structure blokkot, hogy a program belépjen a PID paraméterek küldéséhez szükséges szakaszba.

### PID paraméterek átalakítása:

- A három PID paramétert ( $K_p$ ,  $K_i$ ,  $K_d$ ) az input kontrollok segítségével kérjük le. A paramétereket Number To Fractional String blokk segítségével alakítjuk át string formátumra, hogy később azokat egyesíthessük.

### String összeállítása:

- Az átalakított PID paraméterekből egy megfelelő string üzenetet kell létrehozni. Ehhez a Concatenate Strings blokkot használjuk. Az üzenet formátuma a következő: SET\_PID:%f,%f%,f\n.
- SET\_PID: Ez egy konstans, amely jelzi, hogy a PID paraméterek beállítása következik.
- Veszők: A három paraméter között elválasztásul vesszők szerepelnek.
- \n: Ez a karakter a sor végi karakter (új sor), amelyet a Concatenate Strings blokk segítségével hozzáadunk a string végéhez.

### PID paraméterek küldése:

- Miután a string üzenet elkészült, a Visa Write blokk segítségével elküldjük azt az ESP32-S3 mikrokontroller felé.

#### 6.4. Szög elfordolás megjelenítése

A **motor szögének elfordulása** a rendszer egyik kulcsfontosságú visszajelzési paramétere, amely lehetővé teszi a felhasználó számára, hogy nyomon kövesse a motor aktuális pozícióját és annak változását a vezérlés alatt. A **LabVIEW SCADA** környezetben a szög elfordulás megjelenítése folyamatosan frissül és vizuálisan jeleníti meg a motor aktuális pozícióját.

##### MOTOR\_REACHED üzenet fogadása:

- Miután a LabVIEW elküldi a motor pozícióját, folyamatosan figyeljük, hogy a mikrokontroller visszaküldi-e a MOTOR\_REACHED üzenetet. Ez az üzenet jelzi, hogy a motor elérte a kívánt pozíciót. Ehhez nem használunk Visa Read blokkot, hanem egy Property Node-ot alkalmazunk a byte beérkezésének figyelésére. A Property Node segítségével ellenőrizzük, hogy a válasz sikeresen megérkezett-e.

##### Byte beérkezésének ellenőrzése:

- A Property Node folyamatosan figyeli, hogy érkezett-e byte a mikrokontrollertől. Ha a MOTOR\_REACHED üzenet megérkezett, akkor a rendszer továbblép a pozíció frissítéséhez.
- Ha a válasz nem érkezik meg, a rendszer nem frissíti a pozíciót, és nem folytatja a vezérlési ciklust.
- Miután az üzenet megérkezett, a Comparison Greater blokk segítségével ellenőrizzük, hogy a visszaküldött érték nagyobb-e, mint 0.

##### Pozíció frissítése és számítása:

- A motor pozícióját a Feedback Node segítségével frissítjük. A visszaküldött elmozdulás (pl. 30°) hozzáadódik az előző pozícióhoz.
- A Quotient & Remainder blokkot használjuk annak biztosítására, hogy a pozíció ne lépje túl a 360°-os határt. Ha a pozíció meghaladja a 360°-ot, akkor a számítás nullázódik, és újra 0°-ról indul a számítás.

##### Pozíció frissítése a felhasználói felületen:

- Miután a motor pozícióját kiszámítottuk, azt vizuálisan megjelenítjük a LabVIEW felhasználói felületén.
- A motor aktuális szögét egy 3D segítségével ábrázoljuk.
- Ha a motor elérte a kívánt pozíciót, a rendszer vizuálisan is jelzi a felhasználó számára hogy a pozíciót sikeresen elérte.

##### Pozíció alaphelyzetbe állítása:

- A felhasználói felület lehetőséget biztosít arra is, hogy a felhasználó szükség esetén **visszaállítsa a téglatest pozícióját alaphelyzetbe**, azaz **0 fokos szögállásba**. Ez különösen hasznos a rendszer újraindításakor vagy referenciahelyzet beállításakor.

## 7. A rendszer működése és jövőbeli irányok

A rendszer a tervezett céloknak megfelelően működik, és az elvégzett tesztelések megerősítik, hogy a motor pozíószabályozás megfelelő pontossággal és megbízhatósággal történik. A tesztelési folyamatok különböző forgási sebességeken, különböző terhelésekkel történtek, hogy biztosítani tudjuk a motor precíz irányítását.

### Tesztelési eredmények:

- Motor pozíció szabályozása:** A motor pozíciója folyamatosan és pontosan követhető volt a rendszerben. A PID paraméterek finomhangolása után a motor precízen elérte a kívánt pozíciókat, és a kívánt  $\pm 2^\circ$  tolerancián belül maradt a tesztelt forgások során.
- Kommunikáció stabilitás:** Az **USB** és **Visa kommunikációs modulok** stabil kapcsolatot biztosítottak a **LabVIEW** és az **mikrokontroller** között. Az adatátvitel zökkenőmentesen működött, és a visszajelzések valós időben érkeztek a felhasználói felületre.
- Felhasználói felület:** A **LabVIEW** felhasználói felület megfelelően kezelte a PID paraméterek beállítását, a motor pozíóját és a rendszer állapotát. A felhasználói interakció gördülékenyen zajlott, és az értékek pontosan frissültek a vezérlő rendszerben.
- Tesztelés különböző paraméterekkel:** A rendszer különböző **PID paraméterek** ( $K_p$ ,  $K_i$ ,  $K_d$ ) beállításával is stabilan működött. A tesztek azt mutatták, hogy a megfelelő paraméterek kiválasztása és finomhangolása biztosítja a rendszer optimális működését.

### Jövőbeli irányok:

- Távoli vezérlés:** A rendszer távoli vezérlése **Wi-Fi** vagy **Bluetooth** segítségével, amely lehetővé tenné a felhasználók számára, hogy bárhonnán vezérelhessék a motor pozíóját és egyéb paramétereket.
- Hőmérséklet monitorozás fejlesztése:** A motor hőmérsékletének és a rendszer elektronikai hőmérsékletének monitorozása további biztonságot adhat a rendszer működéséhez, különösen a hosszú távú terheléses tesztelésnél. A hőmérsékletadatokat egy kényelmes módon beépítve fejlesztés során szerzett tapasztalatok és a jövőbeli irányok segítenek a rendszer további fejlesztésében és finomhangolásában. A következő lépések és lehetőségek felmerültek a fejlesztés során:

### Fejlesztési tapasztalatok:

- Kommunikáció és adatátvitel:** A **Visa kommunikációs modulok** megbízhatóan működtek, de a rendszer további fejlesztéséhez javasolt lenne más kommunikációs protokollok, mint például a felhasználói felületbe.
- Fejlettebb hibakezelés és védelmi mechanizmusok:** Bár a rendszer már tartalmaz alapvető védelmi mechanizmusokat, további fejlesztések lehetségesek a **feszültségcsúcsok**, **áramkimaradások** és **rendszertárterhelések** kezelésére, hogy biztosítsuk a rendszer hosszú távú megbízhatóságát.

## 8. Összefoglaló

A jelen munka keretében egy DC motor pozíciós szabályozására alkalmas beágyazott platform került megtervezésre és megvalósításra, amely stabilan működő, valós idejű szabályozást biztosít.

A rendszer az ESP32-S3 mikrovezérőlő számítási teljesítményére és perifériakészletére, valamint a FreeRTOS valós idejű operációs rendszer determinisztikus ütemezési lehetőségeire épül.

A firmware rétegzett, interfész-alapú architektúrával rendelkezik, amelyben a hardvermeghajtók, a szabályozási algoritmusok és az alkalmazási logika jól elkülönített komponensekként valósulnak meg. Ez a felépítés nemcsak az átláthatóságot és a karbantarthatóságot biztosítja, hanem lehetővé teszi az egyes modulok független cseréjét és bővítését is anélkül, hogy a rendszer többi részét módosítani kellene.

A hardveres motorvezérlés a DRV8876 H-híd meghajtón, a pozíció-visszacsatolás az 1024 PPR felbontású inkrementális enkóderen, a zárt hurkú szabályozás pedig egy diszkrét idejű PID algoritmuson alapul, amelyet mozgásprofil-generátor, elakadás-detektor, beállás-érzékelő és lágy pozíciókorlát egészít ki.

Az USB CDC soros interfészen megvalósított kommunikációs protokoll lehetővé teszi a célpozíció beállítását, a PID paraméterek valós idejű hangolását és a motor állapotának folyamatos megfigyelését.

A LabVIEW-alapú felhasználói felület 3D vizualizációja tovább növeli a rendszer demonstrációs és oktatási értékét, mivel valós időben jeleníti meg a motor pozíóját és a szabályozás viselkedését.

A projekt sikeresen demonstrálja, hogy egy mikrovezérőlő-alapú, alacsony költségű platformmal hatékony és megbízható motorpozíció-szabályozás valósítható meg. A moduláris felépítésnek köszönhetően a rendszer készen áll további funkciók integrálására – például sebességszabályozási üzemmód, hálózati kommunikáció vagy több tengelyes koordinált vezérlés ami lehetővé teszi ipari, oktatási és kutatási célú továbbfejlesztését egyaránt.

## 9. Idegen nyelvű Összefoglaló

In this work, an embedded platform for DC motor position control was designed and implemented, providing stable, real-time closed-loop regulation. The system is built upon the computational performance and peripheral set of the ESP32-S3 microcontroller, combined with the deterministic scheduling capabilities of the FreeRTOS real-time operating system.

The firmware features a layered, interface-based architecture in which hardware drivers, control algorithms, and application logic are implemented as clearly separated components. This structure ensures not only transparency and maintainability, but also allows individual modules to be independently replaced or extended without modifying the rest of the system.

The hardware motor control relies on the DRV8876 H-bridge driver, position feedback is provided by a 1024 PPR incremental quadrature encoder, and the closed-loop regulation is based on a discrete-time PID algorithm supplemented by a motion profile generator, stall detector, settle detector, and soft position limiter.

The communication protocol implemented over the USB CDC serial interface enables target position setting, real-time PID parameter tuning, and continuous monitoring of the motor state.

The LabVIEW-based user interface with 3D visualization further enhances the demonstration and educational value of the system by displaying the motor position and control behavior in real time.

The project successfully demonstrates that effective and reliable motor position control can be achieved using a low-cost microcontroller-based platform. Thanks to its modular design, the system is ready for the integration of additional features — such as velocity control mode, network communication, or multi-axis coordinated control — enabling further development for industrial, educational, and research purposes.

## 10. Irodalom jegyzék

- [1] Espressif Systems, „ESP32-S3-WROOM-1 Datasheet,” 2023.  
[https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1\\_wroom-1u\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf)
- [3] Texas Instruments, „DRV8876 H-Bridge Motor Driver Datasheet,” SLVSEB3B, 2021.  
<https://www.ti.com/lit/ds/symlink/drv8876.pdf>
- [4] Diodes Incorporated, „AP62250 2.5A Synchronous Buck Converter Datasheet,” 2022.  
<https://www.diodes.com/assets/Datasheets/AP62250.pdf>
- [5] MaxLinear, „SPX3819 500mA Low Noise LDO Regulator Datasheet,” 2019.  
<https://www.maxlinear.com/ds/spx3819.pdf>
- [6] 3PEAK, „SIT3485E RS-485/RS-422 Transceiver Datasheet,” 2022.  
<https://ptelectronics.ru/wp-content/uploads/SIT3485-1.pdf>
- [7] Espressif Systems, „ESP-IDF Programming Guide – PCNT (Pulse Counter),” 2024.  
<https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/api-reference/peripherals/pcnt.html>
- [8] Espressif Systems, „ESP-IDF Programming Guide – LEDC (LED PWM Controller),” 2024.  
<https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/api-reference/peripherals/ledc.html>
- [9] R. Barry, Mastering the FreeRTOS Real Time Kernel – A Hands-On Tutorial Guide. Real Time Engineers Ltd., 2016. [https://www.freertos.org/Documentation/02-Kernel/07-Books-and-manual/01-RTOS\\_book](https://www.freertos.org/Documentation/02-Kernel/07-Books-and-manual/01-RTOS_book)
- [10] Espressif Systems, „TinyUSB – USB Device Stack for ESP-IDF,” 2024.  
[https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/api-reference/peripherals/usb\\_device.html](https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/api-reference/peripherals/usb_device.html)
- [11] National Instruments, „LabVIEW User Manual.”. [https://www.ni.com/docs/en-US/bundle/labview/page/user-manual>Welcome.html?srsltid=AfmBOopeYnMnJR4RXobUCJ12RPW2wdpXR\\_JU7hnaULYA3jZHJshMf8HY](https://www.ni.com/docs/en-US/bundle/labview/page/user-manual>Welcome.html?srsltid=AfmBOopeYnMnJR4RXobUCJ12RPW2wdpXR_JU7hnaULYA3jZHJshMf8HY)