VISOKA TEHNIČKA ŠKOLA STRUKOVNIH STUDIJA U SUBOTICI
SZABADKAI MŰSZAKI SZAKFŐISKOLA SZABADKA
SUBOTICA TECH – COLLEGE OF APPLIED SCIENCES

DC MOTOR POZICIÓ SZABÁLYZÓ LABVIEW SCADA KÖRNYEZET FEJLESZTÉSE

DC MOTOR POSITION CONTROL LABVIEW SCADA ENVIRONMENT DEVELOPMENT

RAZVOJ SCADA OKRUŽENJA ZA KONTROLU POZICIJE DC MOTORA U LABVIEW-U

Hallgató

NAGY RICHÁRD
25223020

Mentor

DR. SIMON JÁNOS

Szabadka, 2025

Tartalom

1. Bevezető

A DC motorok pozíciószabályozása számos valós ipari és hétköznapi alkalmazásban jelen van. Ilyen például az automatizált gyártósorokon működő szállítószalagok rendszere, ahol a motorokat pontos helymeghatározással vezérlik, hogy a termékek megfelelő pozícióba kerüljenek címkézéshez vagy csomagoláshoz. További gyakori felhasználási területek a robotkarok pozícióirányítása, CNC gépek tengelymozgatása, 3D nyomtatók fejpozíció-szabályozása, valamint kameramozgató rendszerek, ahol fontos a sima és precíz elmozdulás. Ezekben az esetekben a motor pozícióját szenzorok és szabályozó algoritmusok segítségével folyamatosan figyelik és módosítják, biztosítva a pontos működést.

A projekt célja egy egyedi beágyazott rendszer kifejlesztése, amely lehetővé teszi DC motorok precíz pozíciószabályozását valós időben. A fejlesztett rendszer három fő komponensből áll: hardver, firmware és a SCADA (Supervisory Control and Data Acquisition) környezet. A hardveres tervezés az Altium Designer környezetében zajlott, ahol a nyomtatott áramkör (PCB) és az elektronikai alkatrészek pontos elrendezése lett megtervezve a motor vezérlésére. A firmware fejlesztésére az ESP32 mikrokontroller platformját használtuk az ESP-IDF környezetben, C++ nyelven. Ez lehetővé teszi a motor gyors és pontos vezérlését, valamint a rendszer széleskörű kommunikációját.

A vezérléshez kapcsolódó SCADA rendszer fejlesztése LabVIEW alkalmazásban történt, amely biztosítja a felhasználói interfészt és lehetőséget ad a PID paraméterek dinamikus beállítására. A rendszerben a motor pozícióját folyamatosan monitorozzuk és módosítjuk a SCADA rendszerből, így biztosítva a kívánt pozíciópontok pontos elérését. A SCADA rendszer lehetővé teszi a kommunikációs beállítások konfigurálását is, így biztosítva a különböző eszközök közötti zökkenőmentes adatáramlást.

A következő linken elérhető a projekt hardver-es, firmware-es és LabVIEW SCADA interfész része: https://github.com/Ricsi1231/DC-Motor-Controller

2. Irányítástechnika alapjai

Az irányítástechnika célja, hogy a rendszerek viselkedését meghatározott szabályozó rendszerekkel irányítsuk, így biztosítva azok kívánt működését. Az irányítástechnikai rendszerek legfontosabb jellemzője, hogy folyamatosan figyelik a kimeneti változókat, és a bemeneti jelek módosításával szabályozzák őket. Az egyik legismertebb és legelterjedtebb szabályozó típus a PID szabályozó, amely három paraméter segítségével irányítja a rendszert: arányos (P), integráló (I) és differenciáló (D).

2.1 PID szabályozó és DC motor szabályozás

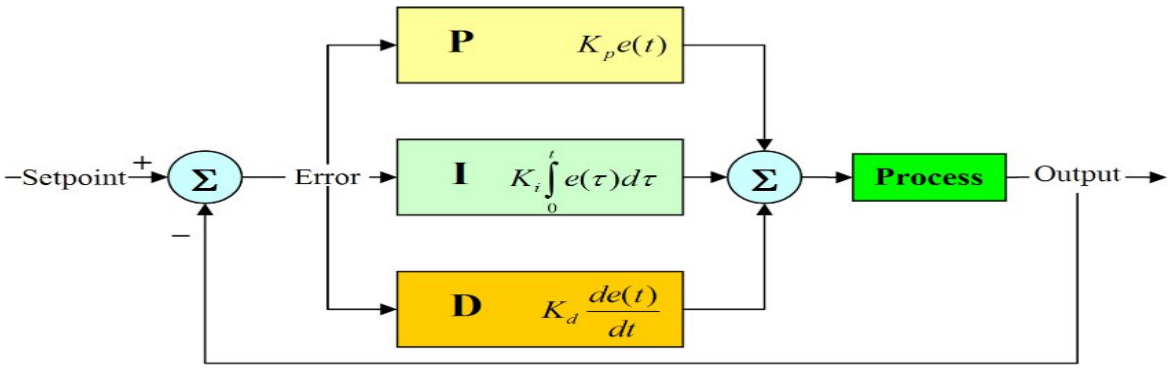
A PID szabályozó egy olyan algoritmus, amely az arányos (P), integráló (I) és differenciáló (D) tagok kombinálásával képes a kívánt kimeneti értéket elérni. A DC motorok szabályozásában a PID szabályozót gyakran alkalmazzák a motorok pozíciójának, sebességének és nyomatékának pontos beállítására.

A DC motorok szabályozásánál a PID algoritmus célja, hogy minimalizálja a pozíció hibát, azaz biztosítsa, hogy a motor pontosan elérje a kívánt szögállást. A PID szabályozó három paramétere – a P, I és D – együttesen lehetővé teszik a gyors reakciót, a rendszer stabilitását és a hiba minimalizálását.

A P (proporcionális) tag a hiba közvetlen arányos részeként működik, tehát minél nagyobb a hiba, annál nagyobb a válasz. Ez gyors reakciót biztosít a hiba csökkentésére, de ha túl nagy, instabilitást és oszcillációt okozhat a rendszerben.

Az I (integráló) tag a hiba időbeli összegét kezeli, és hosszú távon kiküszöböli a maradék hibákat. Előnye, hogy képes a folyamatos hibák eltüntetésére, azonban hátrányos lehet, ha túl magas értéket kap, mert lassan reagálhat, és túlkompenzálhatja a rendszert, ami instabilitáshoz vezethet.

A D (differenciáló) tag a hiba sebességét veszi figyelembe, tehát a jövőbeli változások előrejelzésére szolgál. Ez csökkenti a túlkompenzálást és stabilizálja a rendszert, ugyanakkor érzékeny a zajra, ami hibás válaszokat eredményezhet.

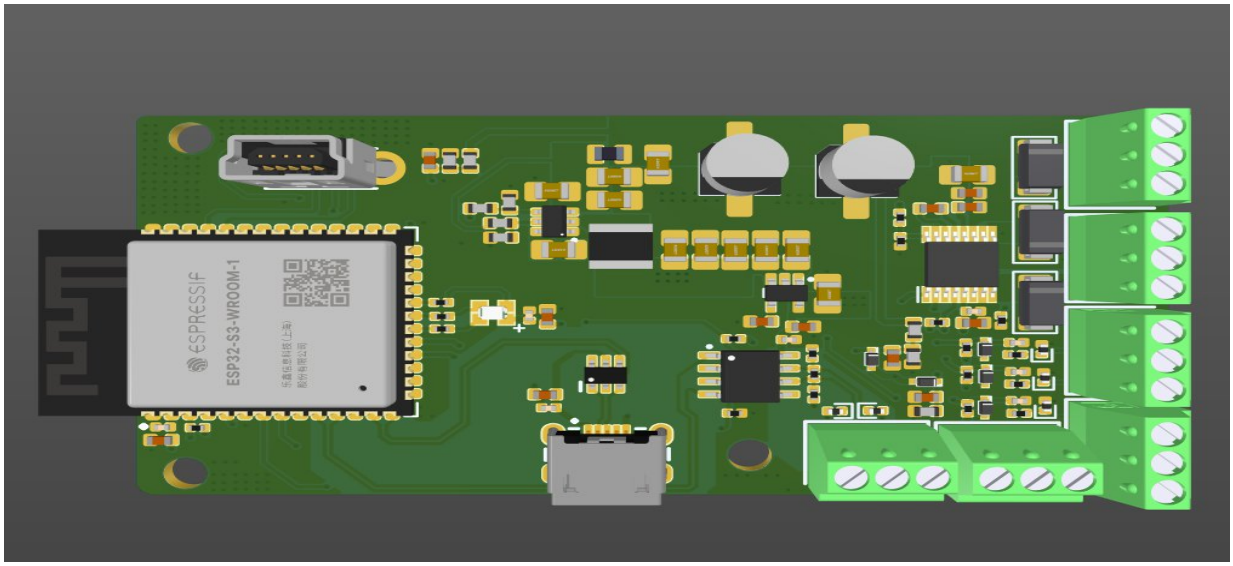


2. Hardver

A rendszer hardveres tervezése Altium Designer környezetben történt, ahol az áramkörök pontos elrendezése és a megfelelő alkatrészek kiválasztása biztosította a rendszer stabil működését. A rendszer három fő komponensből áll: a vezérlő áramkörök, a DC motor, és az enkóder. Mindezek a komponensek elengedhetetlenek a motor pozíciószabályozásához és a rendszer megfelelő működéséhez.

A PCB felépítése a következő főbb jellemzőkkel rendelkezik:

- **Tápellátás:** A PCB 10-18V közötti bemeneti feszültséget képes kezelni, és maximálisan 3A áramot biztosít a DC motor terhelésére. Ez lehetővé teszi a motorok stabil működését különböző alkalmazásokhoz.
- **Enkóder támogatás:** A rendszer támogatja mind a 5V, mind a 3.3V enkódereket, így különböző típusú visszajelzéseket képes fogadni, biztosítva a motor pozíciójának pontos mérését.
- **Modbus IC:** A PCB tartalmaz egy Modbus IC-t, amely lehetővé teszi a Modbus protokoll használatát a rendszer kommunikációjában. Bár a Modbus integráció jelenleg nem lett implementálva a firmware-ben, a hardveres támogatás biztosítja a jövőbeni fejlesztés lehetőségét.
- **Micro USB csatlakozó:** A rendszer rendelkezik egy micro USB csatlakozóval, amely lehetővé teszi a számítógép és az ESP32-S3 közötti kapcsolatot (native USB).
- **Hőmérséklet mérés:** A rendszer lehetőséget biztosít külső hőmérséklet mérő szenzor (NTC) csatlakoztatására a motor hőmérsékletének figyelemmel kísérésére. Továbbá a belső elektronika hőmérséklete is mérhető. Jelenleg egyik hőmérséklet mérés sem lett implementálva a firmware-ben, de a hardveres támogatás készen áll arra, hogy a felhasználó igényei szerint bővíthető legyen.
- **Feszültség mérés:** A rendszer lehetőséget biztosít a laphoz csatlakoztatható bemeneti feszültség mérésére. Jelenleg ez a funkció sem lett implementálva a firmware-ben.
- **ESP32-S3 programozó port:** A rendszer egy dedikált ESP32-S3 USB portot tartalmaz, amely egyszerűsíti a programozást és a firmware frissítést.
- **RGB LED visszajelzés:** Az RGB LED segítségével a rendszer állapotáról kapunk vizuális visszajelzést. Bár a LED visszajelzés nem lett implementálva a firmware-ben, a hardveres támogatás lehetővé teszi annak későbbi aktiválását a rendszer állapotának indikálására.



3. DC Motor és vezérlés

A DC motor (egyenáramú motor) az egyik legelterjedtebb motor típus, amelyet számos alkalmazásban használnak, mivel egyszerű és könnyen vezérelhető. A DC motor működését a rotor és a stator közötti mágneses tér váltakozása biztosítja, ami forgómozgást eredményez. Az irányított áram a motor tekercseiben váltakozó mágneses mezőt hoz létre, amely a rotor elforgatását okozza. A DC motorok előnye, hogy könnyen szabályozhatók sebességük és irányuk tekintetében, ha megfelelő vezérlő áramkört használnak.

A motor sebessége a tápfeszültség növelésével vagy csökkentésével, míg a forgásirány a feszültség polaritásának megváltoztatásával állítható. A precíz motorvezérléshez szükség van egy motorvezérlő IC-re, amely a feszültséget és áramot szabályozza a kívánt sebesség és irány eléréséhez.

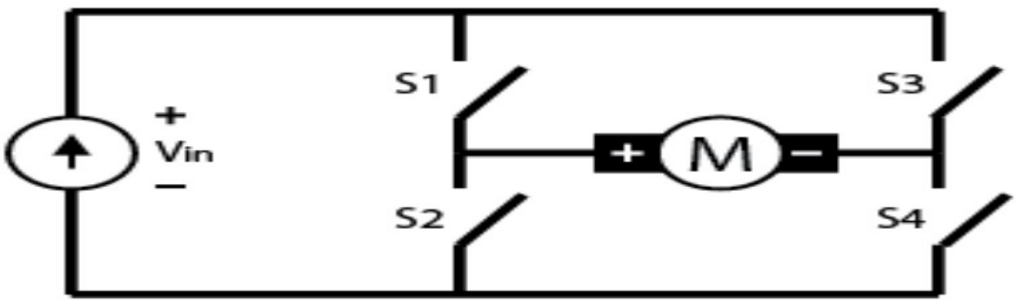
A H-híd (vagy H-Bridge) egy elektronikus áramkör, amelyet általában DC motorok irányítására használnak. A H-híd négy kapcsolóval (általában tranzisztorokkal vagy MOSFET-ekkel) van megvalósítva, és ezek egy H-alakú elrendezésben vannak összekapcsolva. A H-híd lehetővé teszi, hogy a motorra áramot vezessünk két különböző irányba, így képesek vagyunk a motor forgásirányát változtatni.

A H-híd lehetővé teszi a motor sebességének és irányának precíz szabályozását. A motor forgásirányát úgy változtathatjuk meg, hogy a H-hídon áthaladó feszültséget különböző irányban engedjük át, ami lehetővé teszi a motor előre vagy hátra forgatását.

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.

A DRV8876PWPR IC ezen működési módja egyszerűsíti a motorvezérlés implementálását, mivel a PWM és PH jelek segítségével a felhasználó könnyedén irányíthatja a motor sebességét és irányát anélkül, hogy bonyolult vezérlő áramkörökre lenne szükség. Ezen felül az IC beépített védelmi mechanizmusokkal is rendelkezik, így biztosítva a rendszer megbízható működését hosszú távon.



3. Enkóder Szerepe

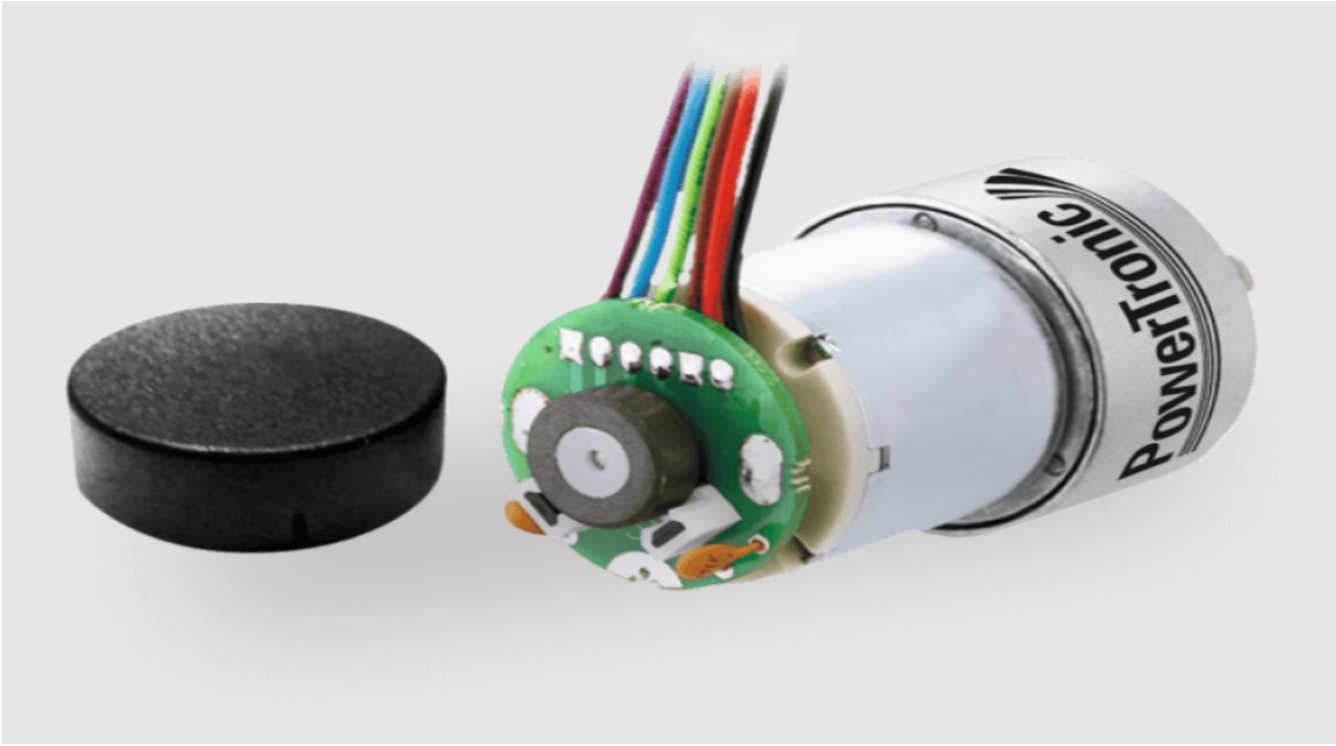
Az enkóder egy olyan eszköz, amely a forgó alkatrészek pozícióját méri, és folyamatos visszajelzést ad a vezérlő rendszernek a motor aktuális helyzetéről. Az enkóder működése alapvetően impulzusok generálásán alapul, amelyeket a motor tengelyének elforgása vált ki. Ezek az impulzusok lehetővé teszik a motor pontos helyzetének, sebességének és irányának meghatározását.

A teljes elfordulásra az enkóder egy meghatározott számú impulzust ad vissza, amely az enkóder típusától függ. Például egy incrementális enkóder esetén egy teljes fordulat (360°) során 360 vagy akár több ezer impulzus is generálódhat, attól függően, hogy milyen felbontással rendelkezik az eszköz. Minél több impulzust generál az enkóder egy teljes fordulat során, annál nagyobb a pozicionálás pontossága, és annál precízebben tudja a vezérlő rendszer követni a motor mozgását.

Két fő típusú enkóder létezik:

- Incrementális enkóder: Az incrementális enkóder folyamatos impulzusokat generál minden egyes elforgásnál. A vezérlő rendszer az impulzusok számát figyelve határozza meg a motor pozícióját és sebességét. Az enkóder visszajelzése alapján a rendszer pontosan számolja a motor helyzetét.
- Abszolút enkóder: Az abszolút enkóder nemcsak az elfordulás számát, hanem az adott pozíciót is közvetíti, így pontosan tudja a rendszer, hogy a motor hol helyezkedik el.

A pontos pozíció és sebesség számításához azonban nemcsak az enkóder visszajelzése szükséges, hanem a motor paraméterei is fontosak. A motor típusának (pl. DC motor) és egyéb jellemzőinek (pl. fogaskerék áttétel, motor áttétel) figyelembevételével a vezérlő rendszer képes kiszámítani a motor pontos szögét és sebességét. Az enkóder jeleit felhasználva és a motor paramétereket figyelembe véve a rendszer képes pontosan szabályozni a motor mozgását, biztosítva ezzel a kívánt pozíció elérését.



4. Firmware

A firmware fejlesztése az **ESP-IDF** környezetben történt, amely az **ESP32-S3** mikrokontrollerhez kínál hivatalos fejlesztői platformot. Az **ESP-IDF** környezet biztosítja az alacsony szintű hardvervezérlést, interfészek kezelését és a magas szintű alkalmazás logika megírását. A rendszer minden komponensét **C++** nyelven implementáltam, figyelembe véve a rugalmas és moduláris kód struktúrát.

A a firmware **layered firmware architecture** alapján lett fejlesztve, ahol az egyes funkcionális modulok külön rétegekben kommunikálnak egymással. Az alkalmazás főbb komponensei a következőképpen vannak strukturálva:

4.1 Komponensek és alkalmazás logika

- **USB**: A rendszer USB kommunikációját egy magas szintű **C++ USB wrapper** modul kezeli. Ez biztosítja a kényelmes és rugalmas adatátvitelt a rendszer és a PC, illetve más eszközök között, mint például a LabVIEW alkalmazás.
- **MotorCommHandler**: Ez a komponens felelős a **LabVIEW**-val való kommunikációért. A **MotorCommHandler** a **USB wrapper**-t használja a kommunikációs feladatok ellátására. Segítségével a felhasználó beállíthatja a **PID paramétereket**, vezérelheti a motor pozícióját, valamint lekérheti a rendszer aktuális állapotát.
- **PID**: A **PID szabályozó** komponens a motor pozíciójának szabályozásához lett implementálva. A **PID algoritmus** folyamatosan figyeli a motor pozícióját, és az aktuális hiba alapján kiszámítja a megfelelő jelet, hogy a motor pontosan elérje a kívánt helyzetet.
- **DRV8876**: A **DRV8876** motor driver IC-t használjuk a motor vezérlésére. A komponens biztosítja a motor forgásirányának változtatását, és sebesség finomhangolását. A motor pozíciójának szabályozásához a **PWM** és a **forgásirány** jelek szolgálnak.
- **Encoder**: Az **encoder** komponens a motor pozícióját és sebességét mérő eszközként működik. Az **ESP32-S3** beépített **PCNT (Pulse Counter)** perifériáját használva az enkóder visszajelzéseit feldolgozza, és biztosítja a motor helyzetének pontos mérését. A komponens képes visszaadni a motor szögét, amit a vezérlő rendszer felhasznál a pozíció szabályozásához.
- **MotorControl**: A **MotorControl** komponens a rendszer központi vezérlőegysége, amely a **PID**, **DRV8876**, és **Encoder** komponensek együttműködésével valósítja meg a motor pozíciószabályozását. Ez a modul felelős a motor pozíciójának szabályozásáért, a rendszer által biztosított visszajelzések alapján.

A fő alkalmazás (main.cpp) célja a DC motor pozíciószabályozása, amely a **MotorControl** és a **MotorCommHandler** komponensek használatával valósul meg. Az alkalmazás folyamatosan figyeli a motor aktuális pozícióját és sebességét, és a felhasználó által megadott paraméterek alapján szabályozza a motor működését.

5. LabVIEW SCADA környezet

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** grafikus programozás erejével biztosítja, hogy minden vezérlési folyamat vizuálisan és valós időben elérhető legyen.

A **LabVIEW** szerepe kulcsfontosságú, mivel:

- **Felhasználói felület biztosítása**: Grafikus eszközök (pl. gombok, diagramok) segítségével a felhasználó kényelmesen állíthatja be a kívánt paramétereket, mint a motor sebessége, pozíciója és a PID paraméterek.
- **Valós idejű adatkommunikáció**: A rendszer figyeli és frissíti a motor aktuális pozícióját, valamint a vezérlési paramétereket. 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 mind a **LabVIEW** alkalmazásban történik, amely megjeleníti a szükséges adatokat és lehetővé teszi a rendszer finomhangolását.

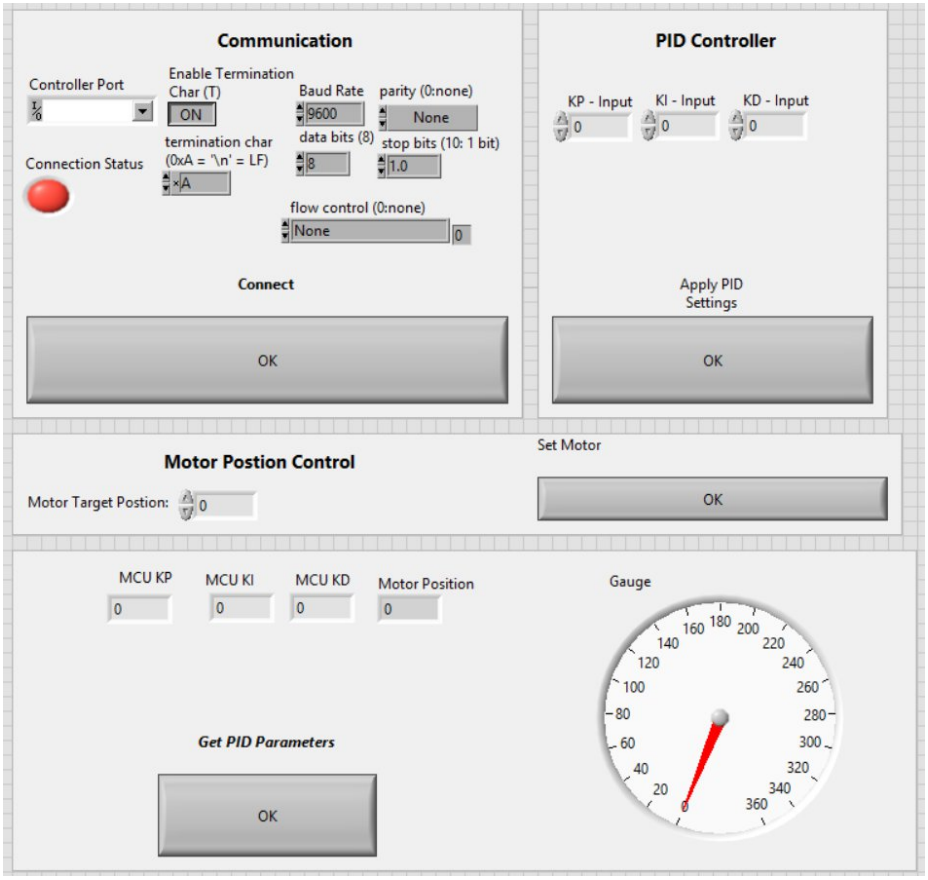
Ezen felül a **LabVIEW SCADA környezet** biztosítja:

- A motor pozíciójának pontos szabályozását.
- A PID paraméterek dinamikus beállítását a felhasználó igényei szerint.
- A motor visszajelzéseinek folyamatos figyelemmel kísérését.
- A kommunikációs és vezérlési interfész biztosítását a PC és a mikrokontroller között.

A felhasználó számára könnyen érthető módon jeleníti meg az adatokat és vezérlési lehetőségeket, és biztosítja a megfelelő kommunikációs csatornákat.

Eszköz csatlakozási állapotának jelzése

A felhasználói felületen található egy lámpa a rendszer csatlakozási állapotát jelzi. Ha a lámpa **piros**, az azt jelenti, hogy a rendszer nem csatlakozott az eszközhöz (pl. az **ESP32**-S3 mikrokontrollerhez), és a kommunikáció nem működik megfelelően. Ha a lámpa **zöld**, akkor a rendszer sikeresen csatlakozott az eszközhöz, és az adatátvitel működik, így a felhasználó továbbra is vezérelheti és monitorozhatja a motor működését.



5.1 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 adatcserét, 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 pozícióját, 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.

5.2 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 (Kp, Ki, Kd), és a rendszer lehetővé teszi ezek lekérdezését is a vezérlő mikrokontrollerből. Az alábbiakban részletesen bemutatjuk, hogyan történik a PID paraméterek küldése és lekérése a LabVIEW és az ESP32 mikrokontroller között.

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 (Kp, Ki, Kd), 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:
    - Kp
    - Ki
    - Kd

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 (Kp, Ki, Kd) 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.
- Vesszők: 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.

5.3 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 leellenőrizzük, 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 Gauge (mérőóra) és egy egyszerű indikátor segítségével ábrázoljuk.
- Ha a motor elérte a kívánt pozíciót, a rendszer vizuálisan is jelzi a felhasználó számára hogy a pozíciót sikeresen elérte.

6. A rendszer működése és jövőbeli irányok

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

Tesztelési eredmények:

1. **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 **±2°** tolerancián belül maradt a tesztelt forgások során.

2. **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.
3. **Felhasználói felület**: A **LabVIEW** felhasználói felület megfelelően kezelte a PID paraméterek beállítását, a motor pozícióját és a rendszer állapotát. A felhasználói interakció gördülékenyen zajlott, és az értékek pontosan frissültek a vezérlő rendszerben.
4. **Tesztelés különböző paraméterekkel**: A rendszer különböző **PID paraméterek** (Kp, Ki, Kd) beállításával is stabilan működött. A tesztek azt mutatták, hogy a megfelelő paraméterek kiválasztása és finomhangolása biztosítja a rendszer optimális működését.

Működési megbízhatóság:

A rendszer megbízhatóságát és hosszú távú működését teszteltük különböző körülmények között, például:

- **Különböző motor terhelés mellett**.

A tesztelés során nem tapasztaltunk komoly hibákat, és a rendszer stabilitása az elvárásoknak megfelelően működött.

A 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:

1. **Kommunikáció és adatátvitel**: A **Visa kommunikációs modulok** megbízhatóan működtek, de a rendszer további fejlesztéséhez javasolt lenne más kommunikációs protokollok, mint például **Wi-Fi** vagy **Bluetooth** integrálása, hogy a rendszer távoli vezérlésére is lehetőség legyen.
2. **PID paraméterek finomhangolása**: A PID algoritmus alapvetően jól működött, azonban néhány finomhangolás szükséges a motor dinamikus viselkedésének pontosabb szabályozásához, különösen gyors vagy hirtelen pozícióváltások esetén.

Jövőbeli irányok:

1. **Távoli vezérlés**: A rendszer távoli vezérlése **Wi-Fi** vagy **Bluetooth** segítségével, amely lehetővé tenné a felhasználók számára, hogy bárhonnan vezérelhessék a motor pozícióját és egyéb paramétereket.
2. **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 beépíthetjük a felhasználói felületbe.
3. **Fejlettebb hibakezelés és védelmi mechanizmusok**: Bár a rendszer már tartalmaz alapvető védelmi mechanizmusokat, további fejlesztések lehetségesek a **feszültségcsúcsok**, **áramkimaradások** és **rendszertúlterhelések** kezelésére, hogy biztosítsuk a rendszer hosszú távú megbízhatóságát.

7. Felhasznát irodalom

Espressif Systems – ESP-IDF Programming Guide,

https://docs.espressif.com/projects/esp-idf/

National Instruments – LabVIEW Basics Manual,

https://www.ni.com/en/support/documentation

Texas Instruments – DRV8876 Motor Driver Datasheet,

https://www.ti.com/product/DRV8876

Altium Designer – PCB Design Documentation,

https://resources.altium.com