Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
TMIT Tanszék

Bányai Klaudia

AUTOMATIZÁLT KALIBRÁCIÓS ESZKÖZ FEJLESZTÉSE ELEKTROMOS KORMÁNYRENDSZER NYOMATÉKSZENZORÁHOZ

KONZULENSEK

Paragi László

(thyssenkrupp Components Technology Hungary Kft.)

Dr. Varga Pál

(Távközlési és Médiainformatikai Tanszék)

BUDAPEST, 2022

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Bányai Klaudia**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2022. 05. 29.

...…….

Bányai Klaudia

Összefoglaló

Az autók egyre komplexebb funkciói miatt jelentősen megnőtt az alkalmazott szenzorok száma. Növekvő számuk miatt több megfigyelési ponttal szolgálnak és nagyobb működési biztonságra adnak lehetőséget. Egyik ilyen biztonságkritikus pont a kormányoszlopon fellépő nyomaték ismerete. Diplomatervem célja ezen nyomatékszenzor kalibrációjához egy kalibrációt segítő eszköz fejlesztése, mind hardveres, mind szoftveres oldalról.

A kormányrendszerek architektúrájának megismerésével kezdtem a munkám, amit egy rövid áttekintő követ, majd a feladatom megértését segítő részletekre térek ki, a nyomatékszenzor működésére, és a jelenlegi kalibrációra. A tervezés fejezetben felsorolom megoldási lehetőségeket az automatizálásra, majd részletezem a választott módszert, annak hardveres és szoftveres kihívásait. Nyomtatott áramkört terveztem a választott megoldáshoz, melynek részleteit is ebben a fejezetben fejtem ki. A választott megoldáshoz mikrokontrollert programoztam, ez a mikrokontroller az adatok előfeldolgozását és az eredmények továbbküldését végzi a számítógép felé. Emellett Python nyelven fejlesztettem egy felhasználói felületet a mérési folyamat egyszerűsítéséhez. Az új, kalibrált értékek számolása is Pythonban valósult meg. Miután az esetleges hardveres és szoftveres módosításokat és teszteléseket elvégeztem, a rendelkezésre álló eszközökből mérési elrendezést állítottam össze. A mérés folyamán elvégeztem a kézi kalibráció menetét, valamint az automatizáltat, és összevetettem a pontosságukat. Emellett más, aszimmetrikus is forgatásokat végeztem, hogy vizsgáljam miként hatnak ezek az automatizált kalibráció-számolásra.

Az elkészített eszköz segítségével az automatizált kalibráció sokkal rövidebb időt vesz igénybe az előző eljáráshoz képest, valamint a mért adatok alapján pontosabb eredményt is hoz. Dolgozatomban javaslatokat nyújtok az ideális kormányforgatásra, valamint arra, miként lehet a számolás pontosságát a forgatás folyamatától függetleníteni a jövőben. Az eszköz hosszútávon könnyíti a kalibráció menetét, és kialakítása miatt további fejlesztésekre is használható.

Abstract

Due to the increasingly complex features of the cars, the number of sensors used has increased significantly. Caused by their increasing number, they provide more monitoring points and greater operational safety. One of these safety-critical points is the knowledge of the steering wheel's torque. The aim of my thesis is to develop the hardware and software of a calibration tool for the calibration of the torque sensors.

I began my work by learning about the architecture of steering systems, followed by a brief historical overview and then I would go into details that will help understand my task, such as the operating principle of the torque sensor, and its present calibration method. In the design chapter, I list solution options for the automation and then detail the chosen method as well as its hardware and software challenges. I designed a printed circuit board for the chosen solution, its details are also explained in this chapter. For the chosen solution, I programmed a microcontroller to process and forward the data to the computer. I also created a user interface in Python to simplify the measurement process. The calculation of the new, calibrated values was also performed with Python. After performing the required hardware and software modifications and testing, I assembled a measurement layout from the available tools. With the measurement setup, I performed the previous calibration process as well as the automated one and compared their results. In addition, I performed other asymmetric – not so ideal - rotations to investigate how they affect the calculation of the automated calibration.

With the support of the new calibration device the automated calibration requires much less time compared to the previous method, and based on the measured data it also provides more accurate results. In this thesis I also presented suggestions for the ideal steering rotation, as well as how the calculation can be separated from the rotation process in the future. In long term the tool facilitates the calibration process and it can be used for further improvements due to its design.

1 Bevezetés

Az életünk mindennapi részévé váltak az autók, a maguk kényelmével, gyorsaságukkal, „okosságukkal” mechanikai remekműnek számítanak. Azonban folyamatos fejlődésük még nem érte el a csúcspontját. Napjainkban az elektromos kormányrendszerek már elengedhetetlen részeit képezik egy olyan szerkezetnek, amely a kezdetek kezdetén elképzelhetetlennek tűnt.

A mai fejlesztések olyan céllal formálódnak, hogy minél több információt szerezzünk a jármű állapotáról. A sok adat segítségével a korábbiaknál pontosabb döntés hozható valós időben, valamint átláthatóbb képet kapunk arról, hogy mi is történik a szerkezeten belül és közvetlen környezetében. Ezen igény kielégítéséhez elengedhetetlenek a szenzorok, többek között a kormányoszlop nyomaték szenzora.

A vezető által a kormánykerékre kifejtett nyomaték ismerete meghatározza a rásegítés mértékét, ezáltal javítja a vezetés minőségét, biztonságát. Mivel már autó-specifikus nemzetközi szabványoknak is meg kell felelni, ezért egy szenzor végleges beépítését nagyon sok tesztelés előzi meg. Ezek a szabványok a biztonság szempontjából kritikus alkatrészekre különösképpen összpontosítanak, a technológiai kereteket viszont nem szabnak a megvalósításhoz.

Dolgozatomban röviden bemutatom a kormányrendszert, azon belül a nyomaték szenzort és annak fontos paramétereit, valamint jelenlegi kalibrációs metódusát. Ezután a megvalósítás menete kerül levezetésre, annak kihívásai, választott technológiái. A kommunikációs interfész megvalósítására is kitérek majd dolgozatomban. Végül egy megállapítással fejezem be, arra vonatkozóan, hogy az összetett eszköz képes-e a kalibrációra, milyen minőségben, valamint milyen további fejlesztések eszközölhetőek a kész rendszerben.

2 Műszaki háttér

A kormányzás lényege, hogy a kormánymű a kormánykerék forgó mozgását a nyomtávrudak hosszirányú mozgásává alakítsa. Ezek felelősek a kerekek fordításáért, miközben a kerékben fellépő erőket a kormánykeréknek adják át. Ezen hatások összessége határozza meg a jármű irányíthatóságát, kormányozhatóságát, valamint a kormányzási érzetet.

Napjainkban a korszerű autók alapvető részét képezi a kormányrásegítő rendszer vagy más néven a szervokormány. Ezen rendszer fő célja a kormányzás emberi erőszükségletét lecsökkenteni egy külső rásegítő egység segítségével.

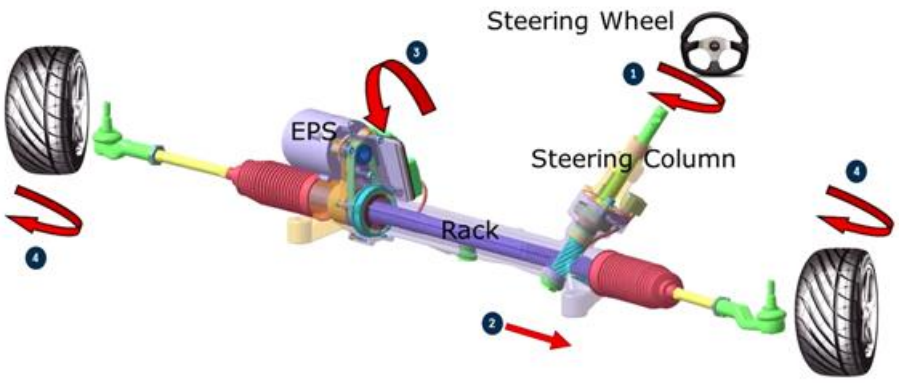
Kezdetben ezt hidraulikus megoldásokkal érték el, melyekben a jármű motorja egy szivattyút hajtott, és a kormányszervo szelepei szabályozták a rásegítő erő mértékét. Ezt követően megjelent az elektrohidraulikus, vagyis az elektromosan szabályozott hidraulikus szervo, majd az elektromos szervo is. Ezekben a beavatkozást villanymotor végzi, a szükséges mértéket pedig a rendszer szenzorai figyelik [1].

Elengedhetetlen, hogy ezek az eszközök megbízhatóak legyenek, tekintve, hogy a jármű irányításában érintettek. A jelenlegi műszaki előírások szerint a rendszer esetleges tönkremenetele esetén a biztonsági szempontoknak való megfelelést kell előtérbe helyezni. Tehát a kormányozhatóságnak akkor is meg kell maradnia, ha a rásegítő rendszer meghibásodik. Ezért kell, hogy legyen egy biztonsági, az eredetit helyettesíteni képes rendszer.

A thyssenkrupp Components Technology Hungary Kft. budapesti fejlesztő központjában olyan elektromos kormányrendszerek tervezése zajlik, melyek elektromos rásegítést alkalmaznak.

2.1 Kormány rendszer felépítésének áttekintése

Az angol „Electric Power Assisted Steering” kifejezés rövidítéseként az elektromos kormányrendszereket EPAS-nak nevezzük/rövidítjük. Ezen rendszer több alrendszerből áll, melyek önmagukban is komplexek. Számos EPAS rendszert különböztetünk meg, melyek más és más konfigurációban jelennek meg. A thyssenkrupp Components Technology Hungary Kft.-nél használt három ilyen rendszert mutatom be röviden a következőkben, az egyik mentén az egész felépítés bemutatásra kerül.



1. ábra Kormányrendszer bemutatása [3]

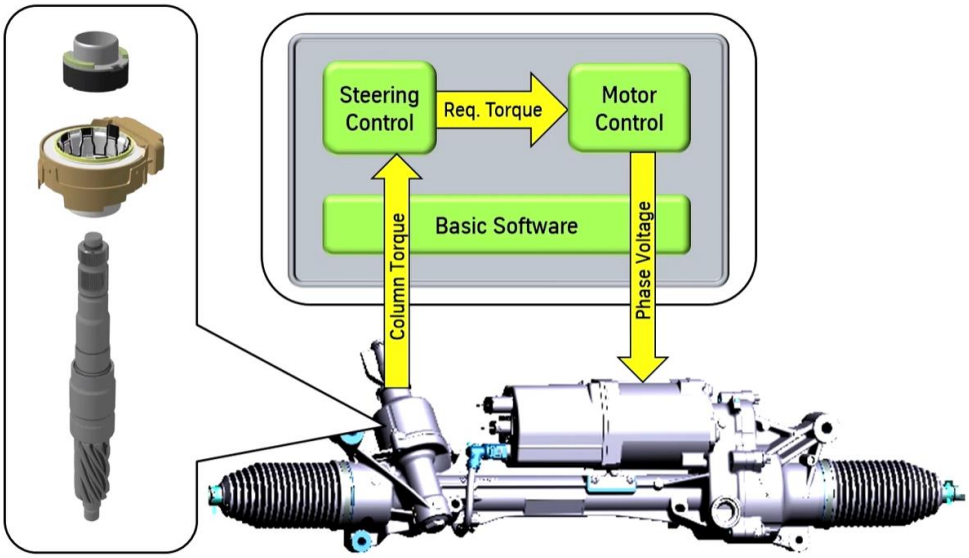
Az 1. ábra kormányrendszer felépítésének főbb részeit az mutatja [2]. A három EPAS típus melyet megkülönböztetünk, a Column-EPAS (C-EPS), a Pin-EPAS (P-EPS), valamint a Rack-EPAS (R-EPS) vagy másnéven ParPas. A rövidítések később kerülnek kifejtésre. A felépítés a Rack-EPAS mentén lesz szemléltetve, az 1. ábra [3] alapján, ahol az egyes számok a nyilakkal együtt mozgásirányra értendők. Az elsődleges bemenet a kormánykerék (1) forgása, amely a kormányoszlophoz kapcsolódik. A kormányoszlop alján található a közbenső tengely, ami mechanikusan kapcsolódik a fogasléc mechanikához, itt továbbítja a kormányoszlop forgása a fogasléc (rack) felé. Ennek hatására az ábrán látható módon (2) a fogasléc elmozdul. Fogasléc mechanikán található a PowerPack, ami a vezérlőegységből (ECU – Electrical Control Unit), a rotor pozíció szenzorból (RPS – Rotor Position Sensor), valamint motorból (EPS – Electric Power Steering) áll. A vezérlőegységben történik az érzékelt vezetői beavatkozás alapján a szükséges rásegítés kiszámítása, valamint ez alapján a motor vezérlése (3). A fogasléc két szélén található nyomtávrúdhoz csatlakoznak az első kerekek melyek a fogasléctől függően forognak a kívánt irányba (4). Az EPAS rendszerben található még egy nyomaték szenzor (TSU – Torque Sensor Unit), amely a kormánykerék nyomatékát méri, pozíciója a rendszerben belül változhat, leggyakoribb alkalmazása a kormányoszlopon történik [3].

A ParPas rendszerek neve abból adódik, hogy paralell, azaz párhuzamosan a fogasléccel helyezkedik el a motor. A RackPas rendszerben az elektromos rásegítő motor a fogasléc mechanikáján van, olyan elrendezésben a szíjjal és gömbcsavarral, hogy a forgó mozgás longitudinálissá (hosszantivá) alakulhasson. Ebben az elrendezésben a nyomaték szenzor olyan közel helyezkedik el az fogasléchez, amennyire lehetséges. Ide tartozik a PinPas rendszer is, melyben a rásegítés közvetlenül a csigahajtáson van.

A Column-EPAS (ColPas) rendszer struktúrájában a Power Pack (az elektromos rásegítő motor és a vezérlő egysége) a kormányoszlopon található. A rásegítő motor egy csigahajtással csatlakozik a kormányoszlophoz. A nyomaték szenzor valamivel ezen csatlakozási pont fölött helyezkedik el.

ParPas és ColPas rendszerek ismerete átfogó képet ad a szenzorok lehetséges elhelyezkedéséről, valamint az autóiparban elterjedt kormányrendszerek struktúrájáról [4][5].

2.2 Nyomatékszenzor



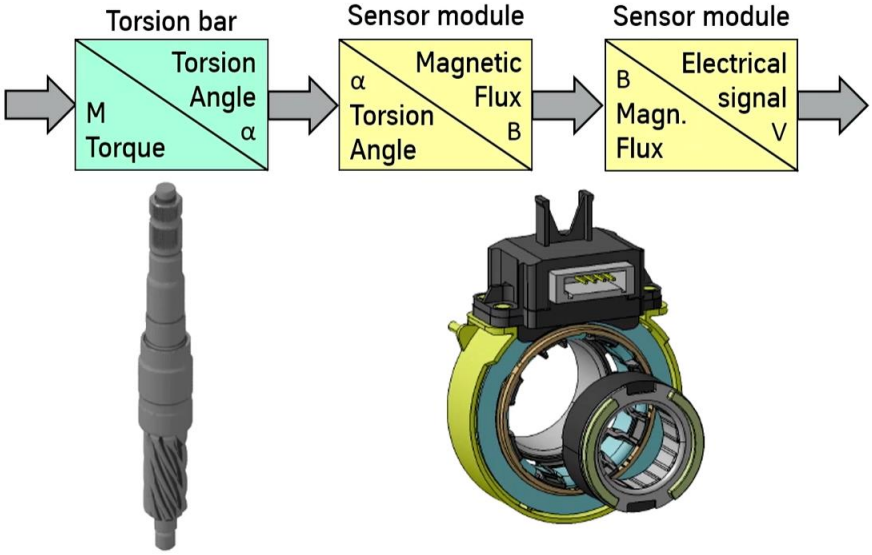
2. ábra Kormányra kifejtett nyomaték hatása a rendszer többi részére

Az EPAS rendszerekben feltétlenül van egy nyomaték szenzor (TSU) - melynek az oszlopnyomaték nagyságának utóbbiénak a motor pozíciójának mérése - valamint egy rotorpozíció szenzor (RPS) – mely pedig a motor pozícióját méri. Ezen szenzorok kimenetei szolgáltatják az adatot a vezérlőegységnek, ami ezáltal a szükséges beavatkozásokat eszközöli. A 2. ábra a szenzorok és a vezérlőegység közötti kommunikációt mutatja, ez esetben egy Parpas rendszer sematikájával. A nyomaték szenzor feladata a vezető kormányzási szándékának mérése, valamint ezen információ továbbítása a kormányrendszer felé. A mérésben résztvevő mechanikai komponensek a torziós rúd (torsion bar) és a szenzor egység. A rendelkezésre álló információk alapján a kormányvezérlő kiszámolja a motortól kikérendő nyomatékot, amelyet a motor-vezérlő az előállításához szükséges fázisfeszültséggé alakít.

A nyomaték szenzor egysége három típusra osztható:

- TOS – Torque Only Sensor, vagyis nyomatékmérő szenzor;
- TAS – Torque and Angle Sensor, azaz nyomaték és szögmérő szenzor;
- TIS – Torque and Index Sensor, azaz nyomaték és index szenzor.

A nyomaték szenzor egység (TSU – Torque Sensor Unit) egy elektromechanikai struktúra, ahol a fenti kép bal oldalán látható módon van egy mágnes, szenzor, valamint egy fluxuscső (lásd: státor lemezek, illetve flux lemezek). Ez az egység védelmi szempontokból házba van tokozva. A ház csapágyakkal csatlakozik a kormányrendszer vázához, ami az alábbiakat foglalja magába: fogaskerék (TAS alkalmazás esetében), torziós rúd, bemenő rúd (input shaft). Nyomaték hatására a torziós rúd bemenete és kimenete között különbségi szög keletkezik. A differenciális szög (különbségi szög) az input és output shaft közötti szögkülönbség, amely a torziós rúd nyomaték hatására történő elhajlásából jön. A differenciális torziós szög mágneses fluxussá történő átalakítása mágneses módon történik. A mágnes és a fluxus-cső státor lemezei közötti szögkülönbség fluxus különbséghez vezet, ami lineáris Hall elemekkel mérhető. A különbségi szög hatására a mágnes és a státor lemezek elfordulnak egymáshoz képest, ami mágneses fluxust generál a szenzoron. Ezt a mágneses fluxust alakítja a szenzor elektromos jellé. Ezt a folyamatot mutatja a 3. ábra.



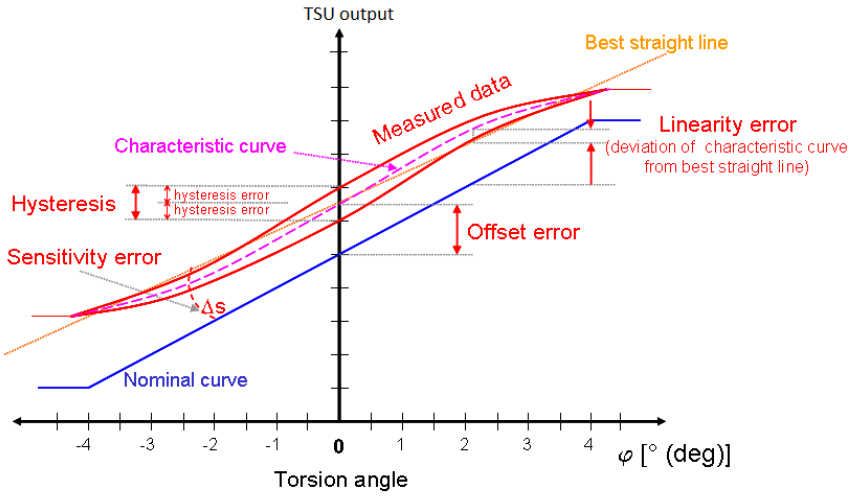
3. ábra Nyomaték elektromos jellé átalakítása

A nyomatékszenzor fontos paraméterei, példákkal:

- Mérési tartomány: ± 10 Nm
- Torziós rúd merevség: 2....3 Nm/°

- Pontosság: ± 0,4 Nm

Az ideális kimenettől eltér a valós kimenet. A 4. ábra4. ábra mutatja milyen hibákkal találkozhatunk egy analízis során. Az x tengelyen a torziós szöget látjuk, y tengelyen pedig a szenzor kimenetét. Látható, hogy a torziós szög a ± 4°-os tartományon mozog. A kék ’nominal curve’ jelöli az ideális működés vonalát, míg a szaggatott rózsaszín, egy polinomiális illesztés a hibával terhelt szenzor kimenetét. A kalibráció során tapasztalható illetve figyelembe vett hibák a következők: hiszterézis, offset, gain, illetve S-curve. A hiszterézis által behozott hiba a mérési eredményeken is látszani fog: egy adott bemeneti nyomatékértékhez tartozó kimeneti érték attól függ, hogy melyik irányba történik a kormányforgatás. Az offset hiba a 0Nm nyomaték kimeneti értékének eltéréseként jelenik meg, az ábrán az y tengely metszéspontjának eltérése. Mivel szimmetrikus a mérési tartomány, ezért 0 Nm-nél a kimenetet teljes tartományának 50%-os megfelelőjén várjuk. A gain hiba az ábrán ’Sensitivity error’-ként látható, ami a két vonal (sárga és kék) meredekségének különbségeként jelenik meg. Az utolsó, kalibráció szempontjából fontos hiba az S-curve hiba, ami a karakterisztikus vonalban (’characteristic curve’) jelenik meg, vagyis, hogy nem lineáris a kimenet, hanem szélső értékeknél nem lineáris karakterisztikát mutat. Erre azonban az 5.3.2 Automatizált kalibrációs fejezetben térek ki részletesebben.



4. ábra: A nyomatékszenzor kimenetén látható hibák

A szenzorra vonatkozó biztonsági koncepció elsősorban a redundanciával valósul meg. A kommunikációs interfész szenzorfüggő, lehet:

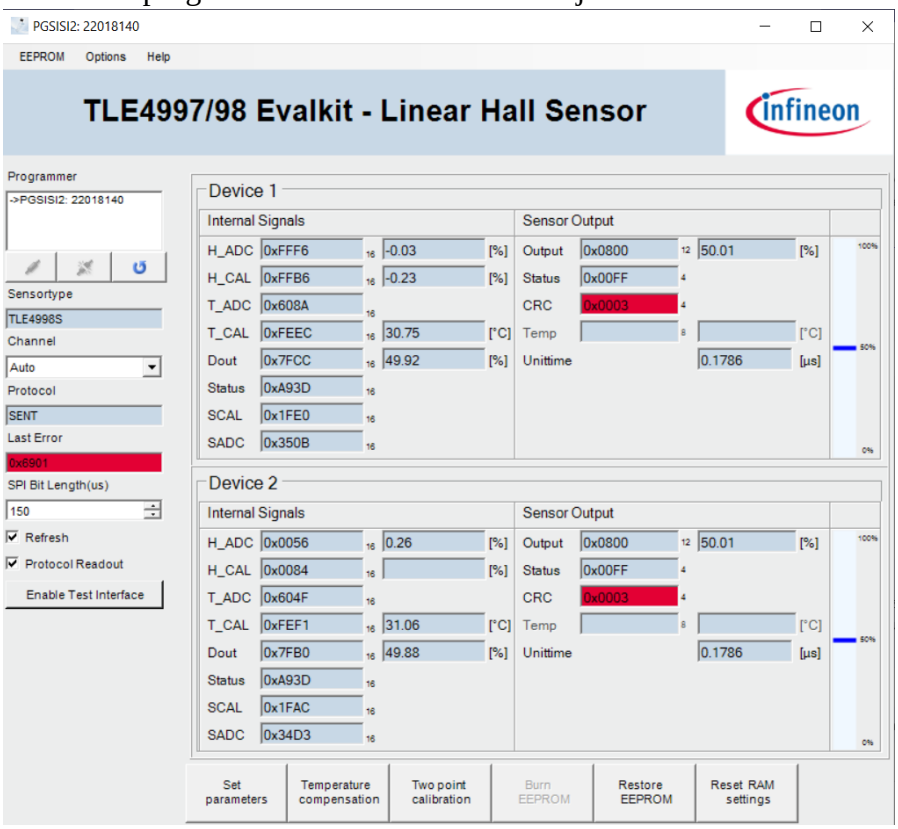
- PSI5: Aszinkron áramerősség modulált kommunikáció, amelyhez elegendő két kábel (föld, táp).
- SENT protokoll: 3 kábel kell, aszinkron.
- SPC protokoll: Triggerre küldött SENT jel, ezáltal szinkron kommunikáció.

Vannak analóg kimenetű nyomaték szenzorok is, de mivel a TSU nem a kormányvezérlővel van egybeépítve, így a hosszú jelút (kábel) miatti analóg zajok okozta hibák elkerülésére, a nyomaték szenzorok többnyire digitális interfésszel rendelkeznek.

2.3 Nyomatékszenzor jelenlegi programozása

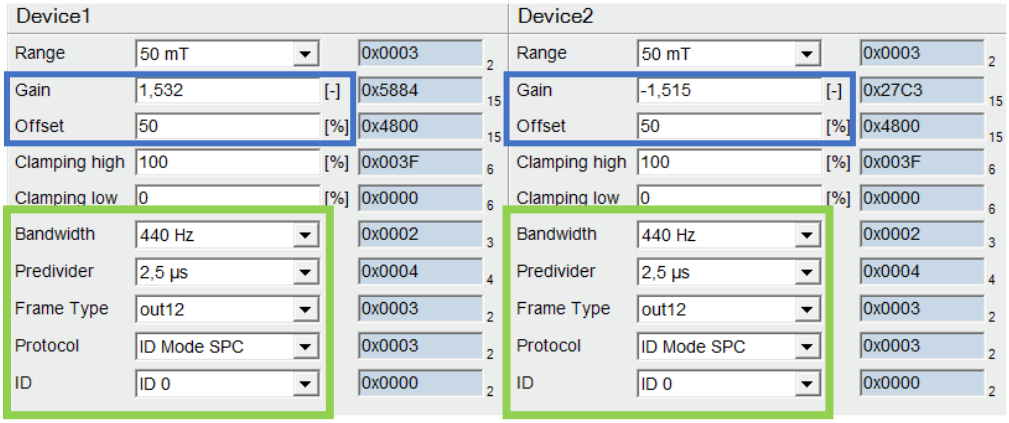
Az Infineon PGSISI2 programozó (továbbiakban PGSISI) eszközzel lehetséges a TSU-t a számítógéphez csatlakoztatni és a TLE4997_98 Evalkit Software-el felprogramozni.

A PGSISI programozó képes a szenzor EEPROM írására és a jelei kiolvasására, amely adatokat a számítógépnek USB COM porton keresztül szolgáltatja. Még az adatátvitel előtt, a fentebb említett program használata szükséges az eszközhöz való csatlakoztatáshoz. A program felületét az 5. ábra mutatja.



5. ábra A két szenzor megjelenítése a programban

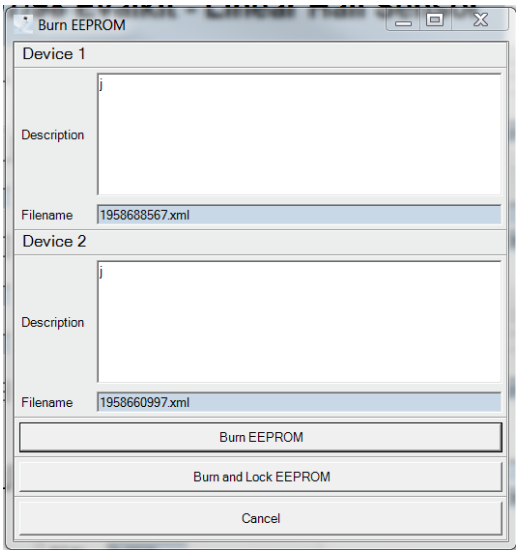
Amennyiben a Protocol Readout lehetőség be van pipálva (ábra bal oldalán), akkor aktuális mért értékeket olvasunk ki a szenzorból, azonban csak ennek kikapcsolásával tudunk a paraméterek beállítására (Set Parameters) lépni. A paraméterek beállítását a 6. ábra mutatja, ahol a zöld keretben foglalt értékek projekt specifikus



6. ábra Paraméter beállítás

beállítások, amelyekben a kalibrálás során nem történik változtatás. A kék keretben található értékek a gain és offset értékek, amelyeik ideális értékét a kalibráció során határozzuk meg. A range is változtatható volna projektenként, ahogy a clamping high és low paraméterek is, azonban ezeket nem változtatjuk.

Ha beállítottuk a paramétereinket a RAM-ba, a szenzor kalibrált kimenettel rendelkezik, de tápvesztés után visszaáll az EEPROM-ban található értékekre. Így a módosításainkat az EEPROM-ba is be kell írni. Ezért a ’Burn to EEPROM’ gombra kattintva, ami a paraméterek alatt található a programban, a 7. ábra ablakára jutunk.



7. ábra Felugró ablak az EEPROM-ba írás folyamatában

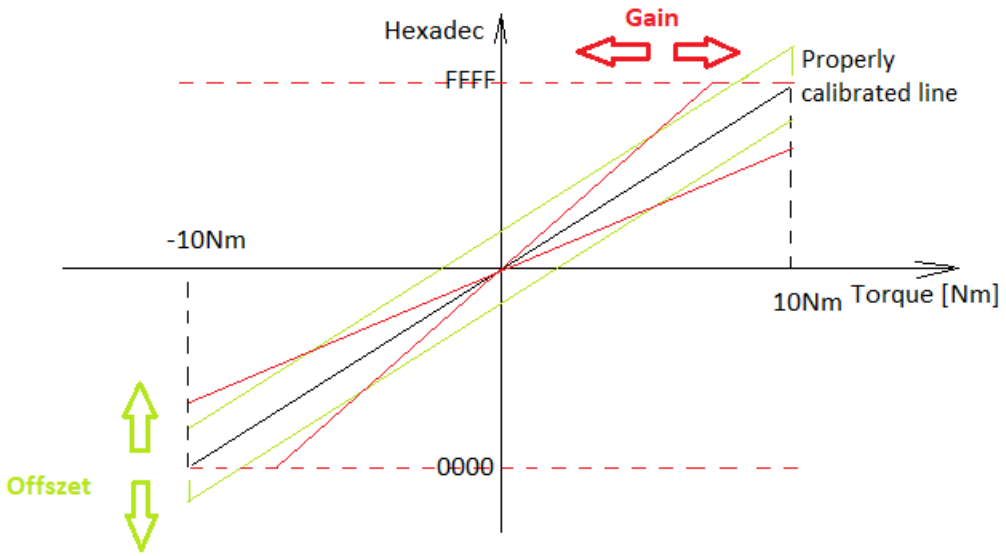
Egy felugró ablak tájékoztat az EEPROM programozás sikerességéről a margin feszültség ellenőrzésével, itt egy ’OK’ gomb nyomásával be is fejeződött a felprogramozása a nyomatékszenzorban található Hall szenzoroknak. Általában 4 darab Hall szenzor van egy TSU-ban, azonban az Evalkit programmal egyszerre csak kettőt tudunk bekötni, ezért egy átkötés szükséges a maradék kettő programozása előtt.

2.4 Jelenlegi kalibráció

Ebben a fejezetben a jelenlegi kalibrációs eljárást mutatom be, aminek minden lépése külön eszközzel történik. A rack-et a mérőpadra úgy kell rögzíteni, hogy a mérőkormány mozgatásával a fogasléc ne mozduljon el, ezt a rögzítést a 8. ábra világoszölddel karikázott részei mutatják. Ekkor a kormány forgatása által elért nyomaték éri csak a TSU-t, melynek kiolvasása PGSISI-n keresztül számítógépen történik. Ha a számítógépen olvasható nyomaték nagyban eltér a mérőkormány által mért nyomaték értékétől, a gain és offset értékek változtatásával kell elérni a két érték azonosságát.

8. ábra Rögzített fogasléc a mérőpadon

A szenzornak többféle kimenete is leolvasható az Evalkit segítségével, hexadecimális értékben is megjelenik a kimenet, valamint százalékértékben is. A nyomatékszenzort a bemutatott kalibrációs példában ±10 Nm-es tartományon fogják használni, ezért a kalibráció is erre a tartományra érvényes. Ez azt jelenti, hogy 0%, vagy a hexadecimális 0x00 érték a -10Nm-nek felel meg. Az állítható gain és offset paraméterek hatását a kimenetre a 9. ábra mutatja. A gain paraméter a kimeneti egyenes meredekségét, míg az offset az y tengellyel való metszéspontját befolyásolja.



9. ábra Gain és Offset paraméterek és a nyomatékszenzor kimenete közötti összefüggés

A használt mérőkormány mérési tartományát korlátozhatjuk ± 100 vagy 10 Nm-re. A mérőkormány megtáplálásához szükséges 12 V-ot külső táppal biztosítottuk. A mérőkormány analóg kimenettel rendelkezik, a kiadott nyomaték multiméter segítségével leolvasható. A legtöbb projekt mérési tartománya ±10 Nm, ez van a mérőkormány által kiadott ±5V feszültségen elosztva. Ennek megfelelően -5V = -10 Nm, +5V =10Nm. A nyomatékszenzor százalékos kimenetét nézzük, tehát -5V mért feszültség esetén a kimeneten 0%, míg +5V esetén 100%-ot vártunk, és ezzel arányosan 0V esetén 50%-ot. Ideális offset beállítás esetén ezt a kimeneti értéket várjuk 0 Nm nyomaték bemenet esetén. A kalibrálás a gain és offset paraméterek hatásának ismeretében, illetve a várt és leolvasott nyomaték értékek különbsége alapján történik. Ezek ismeretében iteratív módon változtatva a paramétereket közelítjük meg a kívánt ideális egyenest. Az ideális gain érték kiszámolására automatizált esetben, a 4.4 Gain és Offset számolás fejezetben térek ki. Az 5. fejezetben bemutatásra kerül a kézi kalibrációra egy példa, ahol ugyanazon a szenzor egységen megy végbe az automatizált kalibráció is.

3 Tervezés

Az előző fejezetben ismertetett kalibráció a következő eszközök használatával történik:

- Tesztpad: Ehhez rögzítjük a szervoegységet.
- Mérőkormány: Referencia nyomatékszenzor.
- Külső táp: A mérőkormány tápja.
- PGSISI: A szenzor programozásához.
- Kábelek: A szenzor két oldalának külön programozásához.
- Számítógép (PC): Az értékek kiolvasásához.
- Multiméter: A mérőkormány kimenetének leolvasására.
- Szenzoros szakértő: A kalibrációs paraméterek kiszámolásához, ellenőrzéséhez és a méréshez szükséges nyomaték képzéséhez.

Több opció felmerült az automatizálás tekintetében, melyekre a következőkben kitérek, ahogy arra is, milyen nehézségekbe ütköztek az adott lehetőségnek.

A tesztpad és a mérőkormány semmiképpen nem helyettesíthető áramköri elemmel, mivel a kormányrendszer mechanikájának megléte szükséges a kalibrációhoz. A jövőben lehet, hogy szimulációval, a mechanika megfelelő modellezésével megoldhatóvá válik a kalibrációs paraméterek meghatározása. A szimulációs megoldás azonban nem ésszerű, mivel sok projekt különböző mechanikával rendelkezik. A szimuláció időigényesebb megoldásnak bizonyul, mint a gyártósorról érkező, több szenzor mérésével történő kalibrációs paraméter meghatározás. A gyártósori adatok statisztikáit felhasználva is meghatározhatók az ideálist közelítő paraméterek, viszont ezen paraméterek darabszórásának mértéke indokolja a TSU egyesével történő kalibrációját. Ez hasonló a gyártósoron tapasztaltakhoz, ahol minden egyes TSU kalibrálásra kerül. Az emberi tényező a nyomaték kifejtéséhez mindenképp szükséges, azonban a további emberi beavatkozás automatizálása megoldható. Értendő az automatizálás alatt a számolás, mintavételezés, amit a kód biztosít, valamint a kábelek átkötésének megoldása relékkel.

Korábban felmerült a folyamat integrálása a PGSISI eszközbe, ezáltal a paraméterekhez kellő emberi fejszámolást a helyettesítve. Azonban ennek az eszköznek az analóg-digitális átalakítója pontatlan, aminek a kiküszöbölésére illesztőáramkörre lett

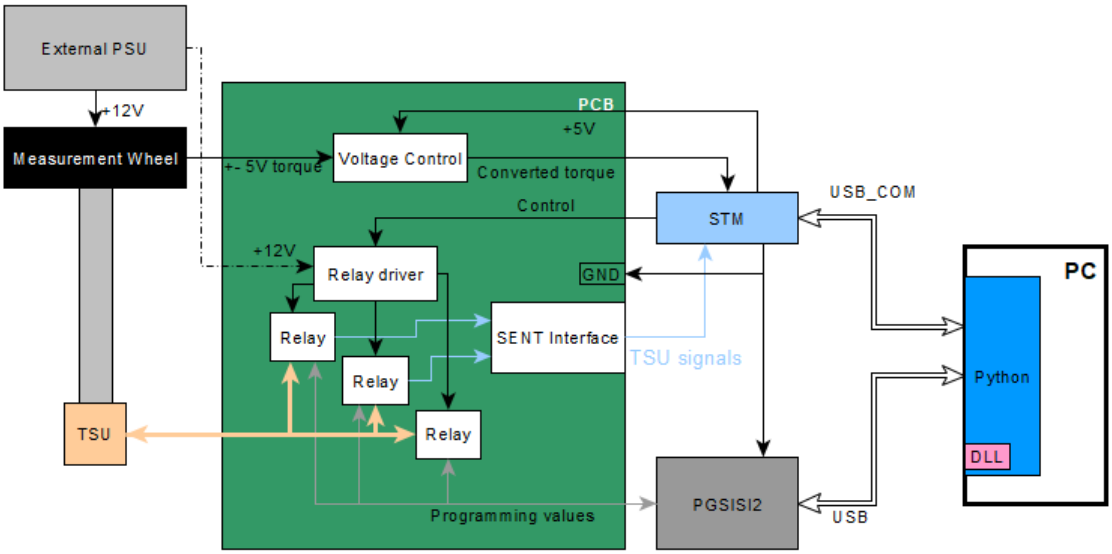
volna szükség. A PGSISI analóg mintavételezési ideje túl alacsony a mérés tervezett futásához képest. Emellett a PGSISI-nek a grafikus felhasználói felületéhez nincs támogatás, a programozása körülményes, az átalakítás pedig nagy munka lenne, így a bele fektetendő idő meghaladná, a feladat megoldásának hasznosságát.

Másik lehetőségként felmerült mikrokontrolleres integrált áramköri összevonás. Ennél a megoldásnál a PGSISI, és az emberi tényező került volna helyettesítésre, azonban a PGSISI és a TSU közötti programozási protokoll megvalósítása túl bonyolult. A 18V-os tápellátást is szükségessé tett volna, mivel a nyomatékszenzor programozásához ekkora feszültség szükséges. Ez a programozó protokoll implementálása miatt túl nagy munka lett volna, illetve a működés közben fellépő magas feszültség miatt illesztőáramkört is igényelt volna.

Felmerült, hogy kábelezést gombokkal vagy kapcsolókkal helyettesítsük, azonban ez plusz nyomtatott áramkört igényel, és a szakszerű szenzoros hozzáértés továbbra is szükséges volna a számoláshoz. Emellett ez a megoldás semmilyen egyéb berendezést sem váltana ki.

A fentebb említett lehetőségek legjobban automatizált opciója és a befektetett idő optimumaként kerestünk megoldást. A következő megoldás mellett döntöttünk: az emberi tényező csak a kalibráció elsődleges összekötésénél, illetve a kormány forgatásához szükséges, azonban a nyomatékszenzor beolvasása, a kalibrálandó értékek kiszámítása, illetve felprogramozása automatizáltan fog megtörténni egy mikrokontroller segítségével. A PGSISI része marad a mérési összeállításnak, viszont egy mikrokontroller irányítja. A kalibráció vezérléséhez egy grafikus felhasználói felületet (GUI – graphical user interface) biztosítunk, ami ugyan nem szükséges a kalibráció működéséhez, de megkönnyíti a TSU-ban lévő adatok ellenőrzését, illetve felhasználó baráttá teszi a kalibrációt. A megoldás összeállítását a 10. ábra mutatja be a következő oldalon.

3.1 Alkalmazott technológia választása

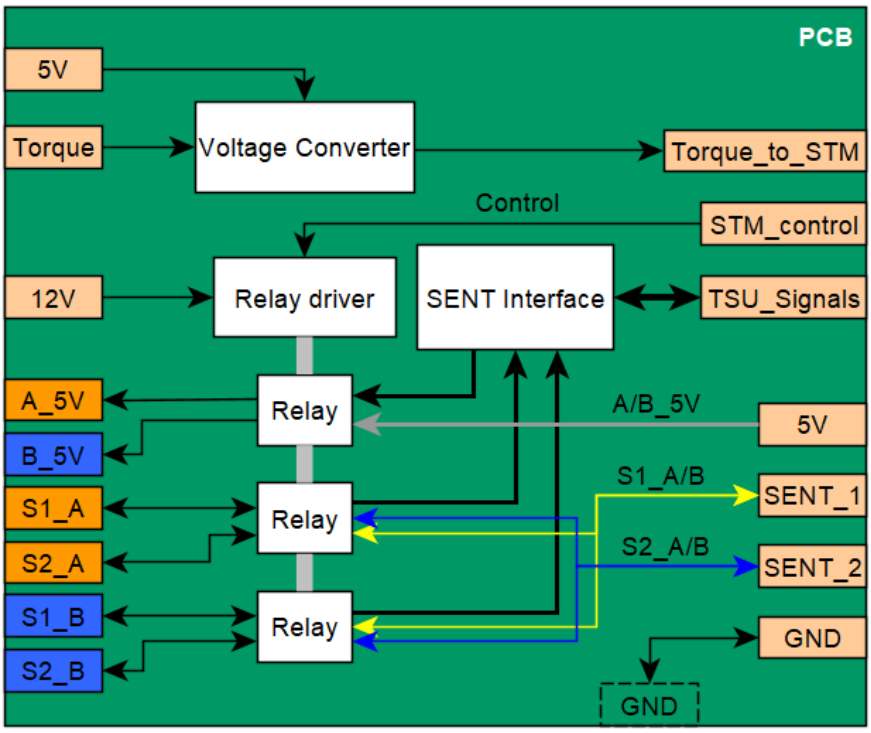


10. ábra: Az összeállítás sematikus ábrája

A kiválasztott megoldás részegységei részben adottak, részben tervezésre várnak. A korábban felsorolt eszközökön kívül a kommunikáció és vezérlés egy részét a mikrokontroller fogja ellátni. Az összeállításhoz választott fejlesztő panel: STM32H743ZI mikrokontrollerrel ellátott Nucleo-144-es panel. A választott mikrokontrollerrel pozitív tapasztalatok vannak csoporton belül, emellett a specifikációja alapján képes az adott feladat végrehajtására. Az STM32 H7-sorozat nagy teljesítményű STM32 mikrokontrollerek csoportja, magja a 480 MHz-es a működési frekvenciára is képes. Többféle STM alapú fejlesztő panel típus is rendelkezésre állt, a Nucleo-144-esből több darab is, így a hiánya nem számottevő, valamint a megvalósított rendszerünk duplikálható is lehet.

3.2 Hardware illesztés (NYÁK tervezés)

A NYÁK, bemenet és kimenet szerint részletezett logikáját, a 11. ábra mutatja.



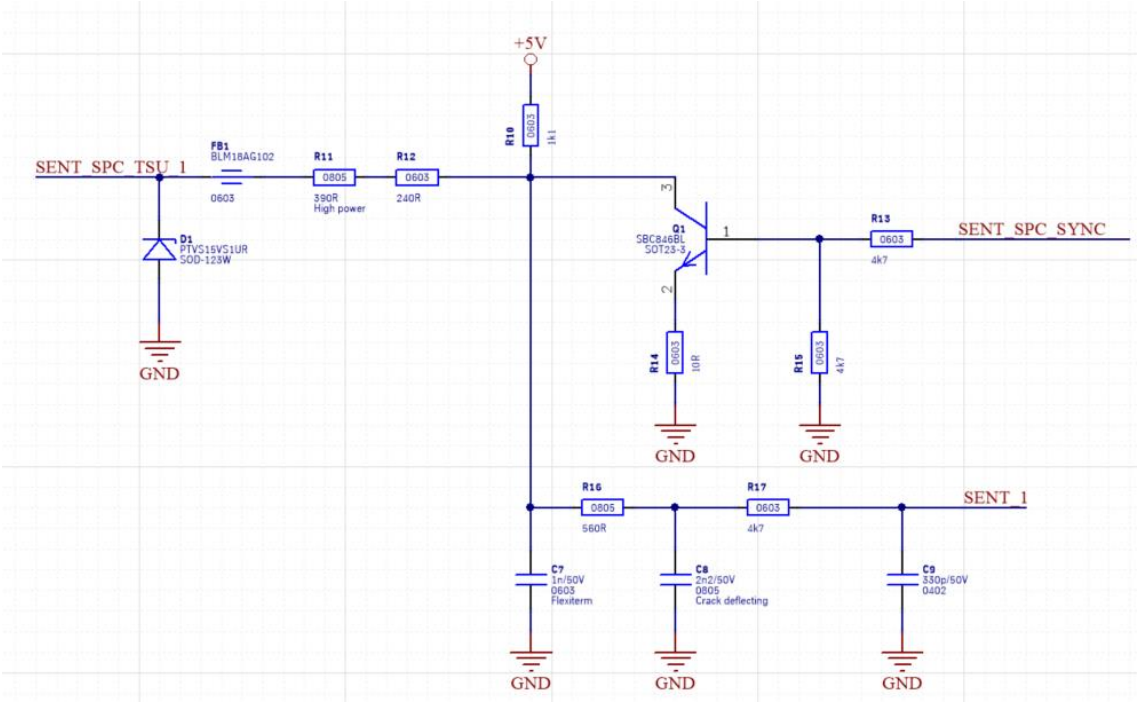
11. ábra: A nyák logikai, egyszerűsített képe

A nyomtatott áramkör tervezése a cégen belül elérhető, Altium Desinger programban történt Az egyes részek szerepére, fontosságára a következő alfejezetekben térek ki.

3.2.1 SENT/SPC interfész

A nyomatékszenzor projekttől függően SENT vagy SPC protokollal kommunikálja a mért adatokat a rendszer felé. Ezek bemutatására az STM - TSU kommunikáció kihívásai fejezetben (3.3.1) térek ki. A bejövő jelek, mikrokontroller által értelmezhetővé alakításáról gondoskodik a 12. ábra által mutatott áramköri részlet. A bal oldalon jön be a nyomatékszenzor jele, a jobb oldalon a ’SENT_SPC_SYNC’ bemenetet a STM küldi, mert SPC protokoll esetében trigger jelre történik az adatküldés. A kimeneti jel a ’SENT_1’ lábra van kötve, ide csatlakozik a mikrokontroller (MCU) egyik lába. Ez az áramköri részlet kisebb módosításokkal ugyan, de már évek óta rendszerszintű alkalmazásban van. Bizonyos elemei biztonsági célzattal vannak beépítve - például a dióda, ami a ESD védelemre szolgál, valamint a ferrit bead, ami zajszűrőként van az

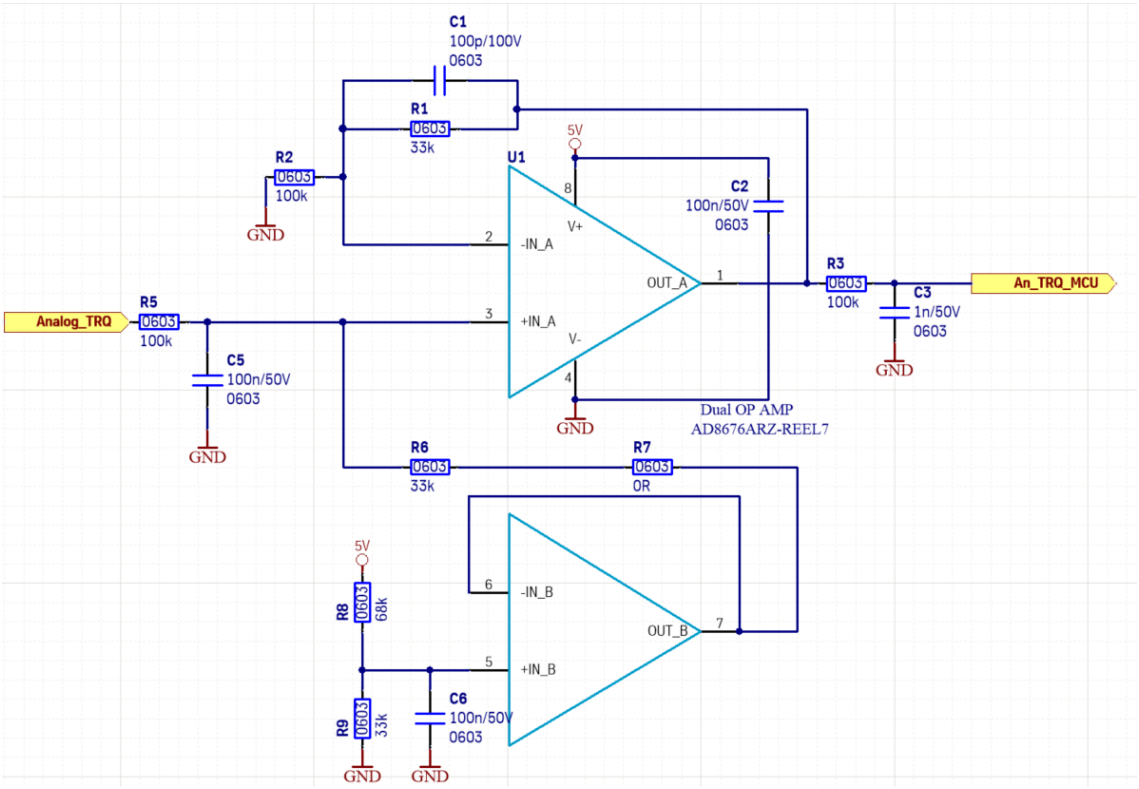
áramkörben. A tranzisztor oldalán, mint említettem a trigger jeltől függően nyit vagy zár a tranzisztor, az áramkör alsó ágában pedig a többszörös RC szűrés.



12. ábra: SENT, SPC interfész sematikus ábrája

3.2.2 Feszültség átalakító működésének bemutatása

A mérőkormány ±5V-os tartományra konvertálja a mért nyomatékot, ezért kell egy feszültségátalakító, ami a bemenetet az STM bemenetének megfelelően alakítja át. Felmerült külső analóg digitális átalakító implementálása a tervezett nyákra, azonban az STM board-nak van saját ADC-je, ami 16 bites pontosságra képes a 0-3.3V-os tartományon. Ez konfigurációtól függ, lehet 12, 10 illetve 8 bitre is konfigurálni az analóg-digitális átalakítót, azonban a pontosság növelése érdekében a 16 bites felbontást választottam. A mérőkormány feszültségének az STM ADC-jével feldolgozható feszültégtartományára történő transzformálására szolgál a lenti ábrán (13. ábra) látható erősítő kapcsolás.



13. ábra: Feszültség átalakító

A fenti áramkör a következő számítás alapján alakítja a bejövő jelet (bejövő $\pm 5V$ esetén):

$$U_{out} = 5\ \text{V} \cdot \frac{R9}{R8+R9} \cdot \frac{R5}{R6+R5} \cdot \left(1+\frac{R1}{R2}\right) + U_{in} \cdot \frac{R6}{R6+R5} \cdot \left(1+\frac{R1}{R2}\right) =$$

$$= 5\ \text{V} \cdot 0.3317 + U_{in} \cdot 0.3300$$

U_{in}= 5 V, U_{out}= 3.3085 V

U_{in}= - 5 V, U_{out}= 0.0085 V

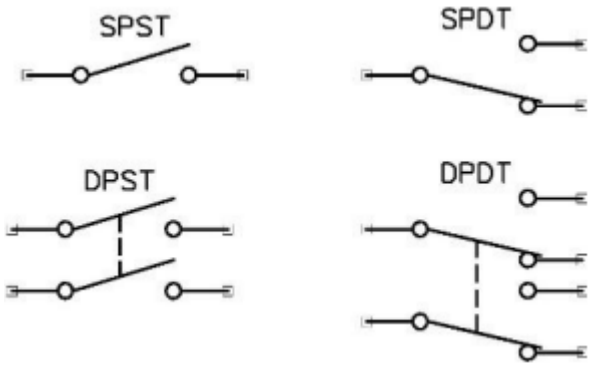
Az STM board AD-átalakítója 0-3.3V-os tartományon működik, ezért a számolt kimenetek alapján célnak megfelelő a feszültség átalakító.

3.2.3 Switch

A nyomatékszenzor jeleit az STM board-on keresztül fogom kiolvasni, azonban mikor a TSU felprogramozása történik, a PGSISIvel való összeköttetés szükséges. Ha a PGSISI programozó jelei az STM boardon keresztül jutnának a szenzorhoz, a 18V-os programozó feszültség miatt sérülne a fejlesztő panel. Kapcsoló nyújt megoldást a leválasztásra. Működésük szerint megkülönböztetünk mechanikus, elektromágneses, illetve elektronikusan kapcsoló kapcsolókat. A két állapot közötti automatizált

(mikrokontrollel által vezérelt) kapcsolás elérése érdekében, az elektromechanikus relé használata mellett döntöttem. A mechanikus kapcsolót egyszerű vezérelni, illetve bármekkora feszültséget és áramot könnyen kapcsol. A következőkben kitérek a fajtákra, amik közül a kiválasztás megtörtént. Felmerült a szilárdtest relé (solid state relay), ahol a kapcsolást fotoszenzor végzi, illetve a mechanikus relé, ahol egy tekercsre áramot kapcsolva az indukált tér vonzó hatása végzi a kapcsolást. Megkülönböztetünk reteszelő, illetve nem reteszelő tekercses fajtákat. A reteszelő tekercses megtartja az aktivált pozícióját egészen amíg az áram nincs fordítva kapcsolva a tekercsre, míg a nem reteszelő tekercs visszatér eredeti pozíciójába, amint a vezérlő áram megszűnik. Az egyszerűbb kapcsolás miatt nem reteszelő megoldást választottam.

A billenőkapcsolók között megkülönböztetünk egyállású, illetve többállású kapcsolót, valamint az irányuk szerint is egyirányút, illetve többirányút. Ezen kapcsolók mechanikáját a 14. ábra mutatja. Az állás, angol irodalomban pólusnak hívják (single pole – SP, double pole – DP), arra utal, hogy hány kapcsoló átváltása történik egyszerre. Az irány, angolul throw (single throw – ST, double throw – DT), arra vonatkozik, hogy hány kapcsoló állás van, egy esetén on-off kapcsolásról beszélünk, a kapcsoló vagy zárja az áramkört, vagy szakadás van az áramkörben. Kétállású kapcsoló esetén két kimenet között történik a kapcsolás [6][7].

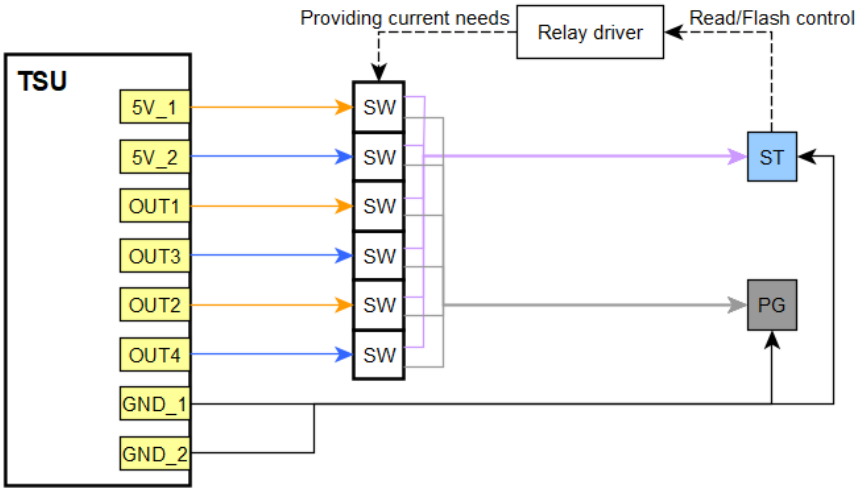


14. ábra: Különböző billenőkapcsoló fajták ábrázolása [6]

A kiválasztás folyamatát a következő paraméterek is szűkítik: tekercs áramfelvétele aktív állapotban - az STM board 100 mA áramot képes a kimenetén kiadni. Az adatlapban specifikáltak szerint, ez a 100 mA a perifériák összességére értendő, nem perifériánként. Mivel külső táp kell a mérőkormányhoz, és az több kimenettel is el van látva, akár 8-10A is adható lenne a relék irányába a második csatornáján. Tekercsfeszültséget tekintve 24V alatti értéket kerestünk. A névleges áram jelöli az érintkezéseken maximálisan átengedhető áram nagyságát, ami esetünkben minimum

50 mA. A kapcsolási feszültség alatt azt a maximális feszültséget érjük, amelyen az érintkezők kapcsolása azok károsodása nélkül még megtehető. A működési idő jelenti a feszültség rákapcsolása és a relé kapcsolójának kontaktusa között eltelt idő, erre különösebb megkötésünk nem volt, ameddig nem hosszú másodpercek szükségesek az átkapcsolásához [8].

Ezek az információk segítették a relék kiválasztását, azonban a korábban már említett, az STM board áramkorlátai miatt egy ’relé driver’-re volt szükség. A relé driver működésének alapja, hogy kis bemeneti áram hatására (pár mA) egy Darlington kapcsolás segítségével nagy kimeneti áramot (akár 500mA) tesz lehetővé. Ez a nagy áram kapcsolja majd a reléket. A relé driver bemenetére kötjük az mikrokontroller vezérlőjelét, a kimenetére pedig a különböző reléket, a tápolásához pedig ugyanazt a tápot fogjuk használni, amit a mérőkormányhoz is használunk, hogy a szükséges eszközök számát ne növeljük. A programozási vagy mérési fázis határozza meg a vezérlőjelet. A relék nem reteszelők, így visszaállnak eredeti állapotukba, ha nem áll fenn az átkapcsoláshoz szükséges tekercsen a feszültség. Az idő nagy részében az adatkiolvasással foglalkozunk, és a programozás csak kis része a mérésnek, ezért az alacsony vezérlőjelű lállásban a nyomatékszenzor jelek az STM board felé lesznek továbbítva. Ekkor csak programozás alatt lesz nagyobb áramfelvételre szükség, illetve vezérlésre. A 15. ábra mutatja ennek a kapcsolásnak egy egyszerűsített változatát. A relé driver kimenete külön-külön lesz bekötve az egyes relékhez, az ábrán az átláthatóság miatt van ez egy vonallal jelezve.



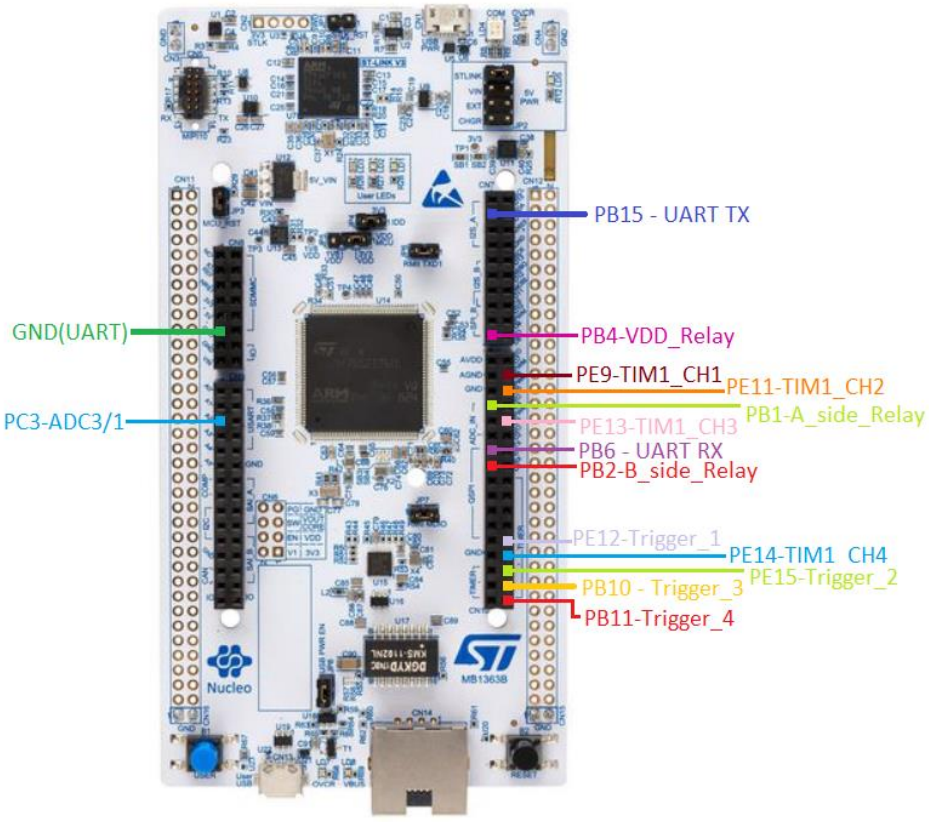
15. ábra Kapcsoló bekötése egyszerűsített ábrán

Leteszteltem, hogy a nyomatékszenzor programozása során okoz-e problémát, hogy ha össze vannak kötve a két oldali (az A és B oldali) tápok, amikor az egyik oldali csatornán programozás folyik. Ennek a célja az volt, hogy ha esetleg olyan relét

használunk, aminek egy bekapcsolása kettő átkapcsolást csinál egyszerre (DPDT 14. ábra), akkor ezek összekötése jel szempontjából a programozást megzavarja-e. Mivel ekkor is lehetőség van a felprogramozás zavarmentes lefolyására az A és B oldal kábeleket összeköthetjük a PGSISI felé.

3.2.4 Nyák tervezése Altium Designer programban

A PCB tervezéséhez a cég által használt Altium Designer programot használtam. A tervezés célja az volt, hogy az STM board ráilleszthető legyen a tervezett nyomtatott áramkörünkre, ezért látható a sok csatlakozási pont, azonban a panel kialakítása azt is lehetővé teszi, hogy pinekkel felülről csatlakozzunk az STM boardra. Az elrendezés tervéhez segített a 16. ábra.



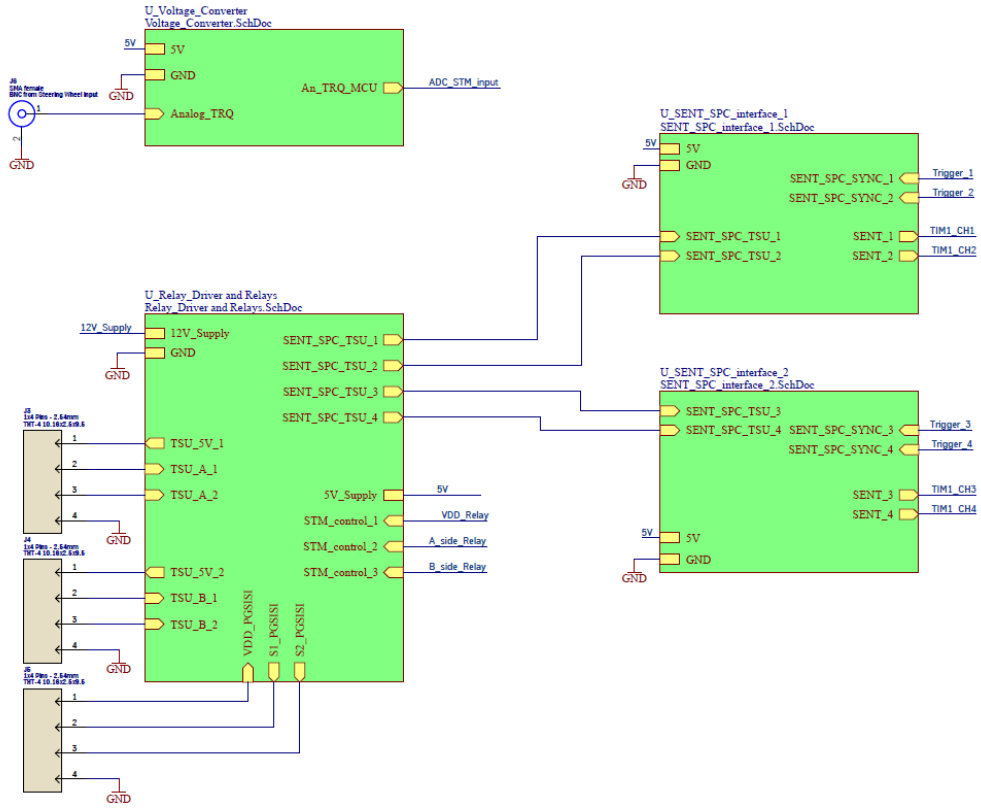
16. ábra: Bemeneti és kimeneti jelek az STM board-on [16]

Emellett szükség volt az STM board pontos méreteihez. Rendelkezésre állt az általunk használt board teljes Altium formátumú forrás fájlja, amit megnyitva bármely kérdéses két pontja között megtudtuk mérni a távolságot. A magas alkatrészek pl.: relék, jack csatlakozók, pinek, BNC, elhelyezése külön figyelmet igényelt, hogy a board körvonalával ne ütközzön. A layout készítésénél GND kitöltést alkalmaztam, ami mikrokontroller paneljén a GND csatlakozásokon keresztül kapcsolódik az STM board

földkitöltéséhez. A csatlakozási pontok sokasága miatt a kósza zavaráramok elenyészők lesznek.

Az alkatrész méreteket igyekeztem 0603-as tokozással egységesíteni, ami a komponensrendelést könnyíti, a tervezett nyákhoz. A reléhez nem volt a létező könyvtárban lábkiosztás (footprint), ezért az adatlap alapján külön kellett létrehozni. A konnektor/pin lábak, ahova az STM board csatlakozik majd, is külön távolságmérést szükségeltek a nem konvencionális pintávolság miatt (nem mindenhol 2.54mm-es kiosztás volt).

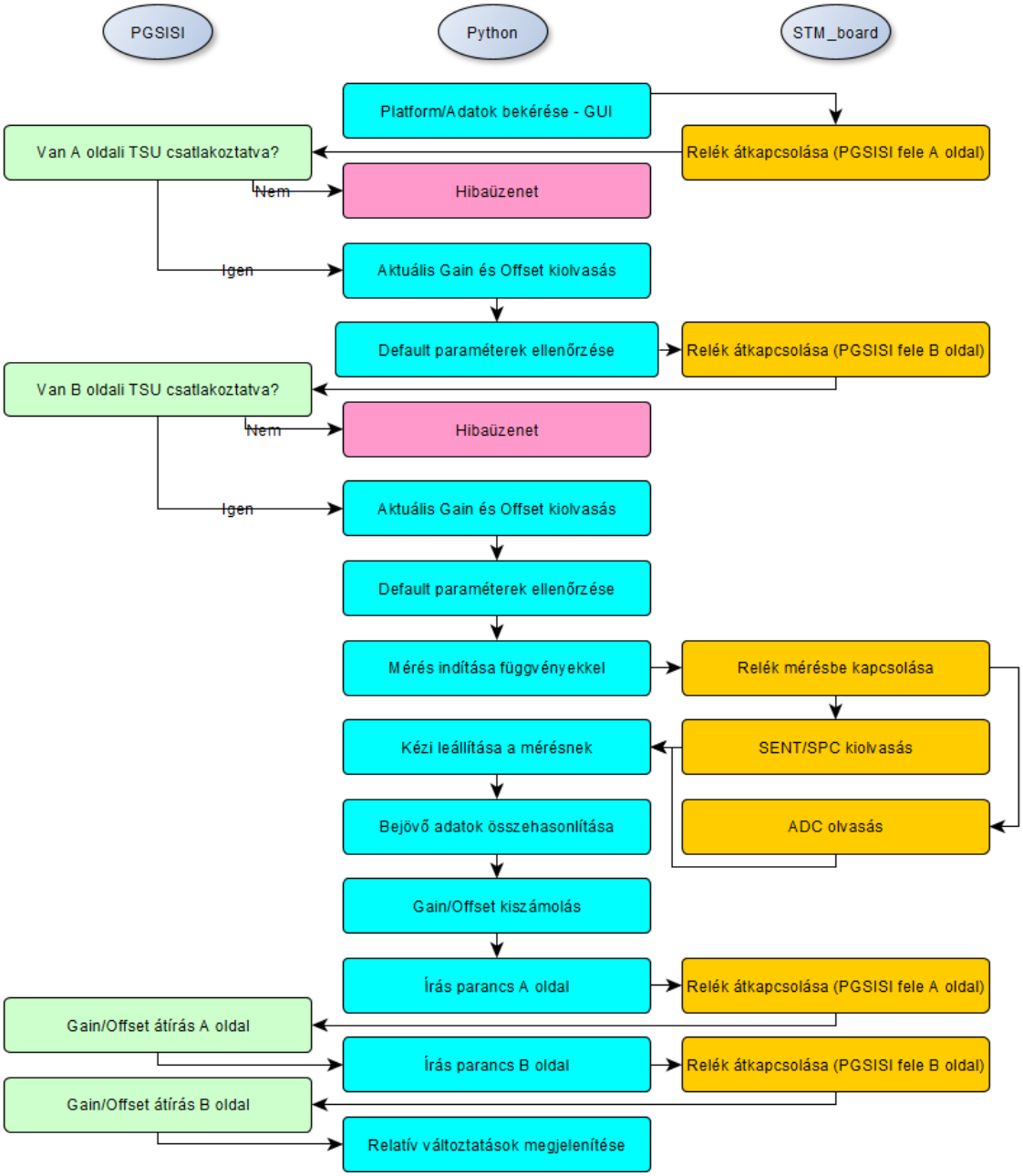
A 17. ábra az Altium terv sematikus ábráját mutatja, hogy az egyes részegységek közötti összeköttetések hogyan valósulnak meg. Emellett az STM board pinkiosztásának megfelelő csatlakozók bekötése is megtörtént a tervezés alatt. Egy plusz jack csatlakozót tettem a nyomtatott áramkörrel azzal a céllal, hogy ha az áramfelvételünk bármilyen oknál fogva sok lenne a boardnak, akkor külső forrásból is lehetséges legyen megoldani tápellátást. Ez lehetőséget ad az erősítő és SENT/SPC interfész +5V-os tápjának külső helyettesítésére. Az STM board 5V-os kimenetének külső tápellátásának elkerülése érdekében egy pinpárral választottam el a két lehetőséget egymástól. Ez a függelékben található három dimenziós áramköri ábráról jobban látható.



17. ábra: Kész PCB sematikus ábrája

3.3 Kommunikációs interfész illesztés

A szoftveres háttérműködést a következő folyamatábra írja le.



18. ábra: Szoftveres háttérfolyamatok folyamatábrája

Az STM board programozásához az Atollic TrueSTUDIO [17] programot használtam. Ez ugyan nem olyan egyszerű preiféria konfiguráló felülettel rendelkezik, mint a STM32CubeIDE [18] (vagy STM32CubeMX), de az Atollic-ra több csoporton belüli technikai támogatottságunk van. A céges környezetben az STMH7-es könyvtárának telepítése minden alkalommal hibára futott, emiatt a Cube-os kódgenerálás

nem használható. A programozás C nyelven történik. Programozás után tápot adva a board-nak, az írt program folyamatosan fut.

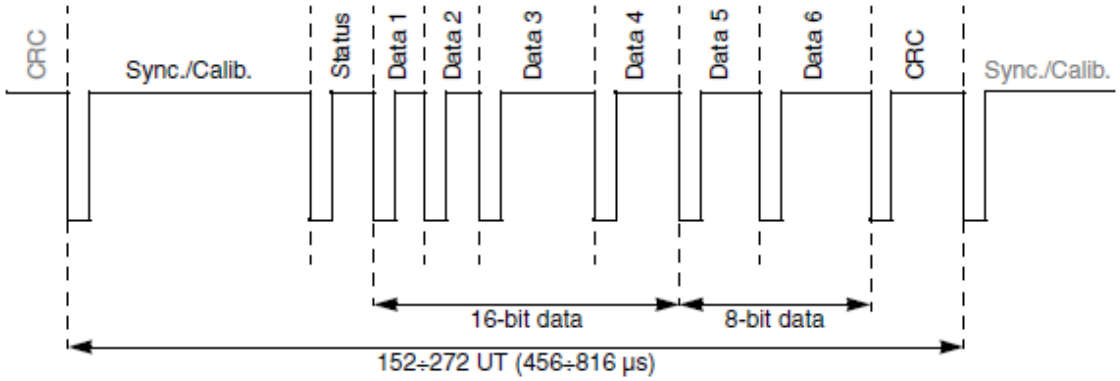
Az otthoni gépre sikerült az STM32CubeIDE telepítése, így a kezdeti ismerkedés a board-dal ezen a felületen keresztül történt meg. Ebben a programban van egy plusz absztrakciós réteg 'HAL library', ami segítette a gyorsabb, átláthatóbb programozást, nem regiszterszintű átlátást igényelt, hanem már kész függvények használatát. Ez egy ideig könnyebbé tette, egy komplexitás után viszont nehezítette a haladást. Nem teljesen lehetett átlátni, hogy egy-egy HAL függvény a háttérben hogyan működik, ezért tértem át az Atollic programra. A CubeIDE viszont arra nagyon hasznos volt, hogy a perifériák inicializálását könnyítse. Egy grafikus felület segítette a lábkiosztás beállítását, így láthatóak voltak a korábban beállított értékek is. Ezzel szemben Atollic-ban az inicializációs kódrészletet kell átnézni, hogy az adott kimenet éppen használatban van-e.

3.3.1 STM – TSU kommunikáció kihívásai

A következőkben bemutatásra kerül, hogy milyen kommunikációs protokollokat használhat a nyomatékszenzor.

3.3.1.1 SENT protokoll

A Single Edge Nibble Transmission (SENT) [9] protokoll egy olyan egyirányú kommunikáció, melyben az szenzoroktól az adatokat aszinkron módon továbbítják a fogadó eszköz felé, annak bármilyen beavatkozása nélkül. Megvalósítható csupán 3 vezeték segítségével (táp, föld, adat). A fogadó eszköz feladata az aszinkron feldolgozás. Az érzékelőktől továbbított jel egy impulzussorozat, ahol az egymást követő lefutó élek közötti idő meghatározza az átadott 4-bites adatot, ezt nibble-nek nevezzük. A teljes átviteli idő az adatok értékeitől és a szenzor órajelétől függ. Az egymást követő SENT nibble-ök átvitele azonnal történik, az előző átvitel befejezése után. A SENT kommunikáció alap időegysége (egység idő – unit time) esetünkben általában 2,5-3µsec. Az órajel maximálisan megengedett eltérése ± 20% lehet a nominális időegységhez képest.



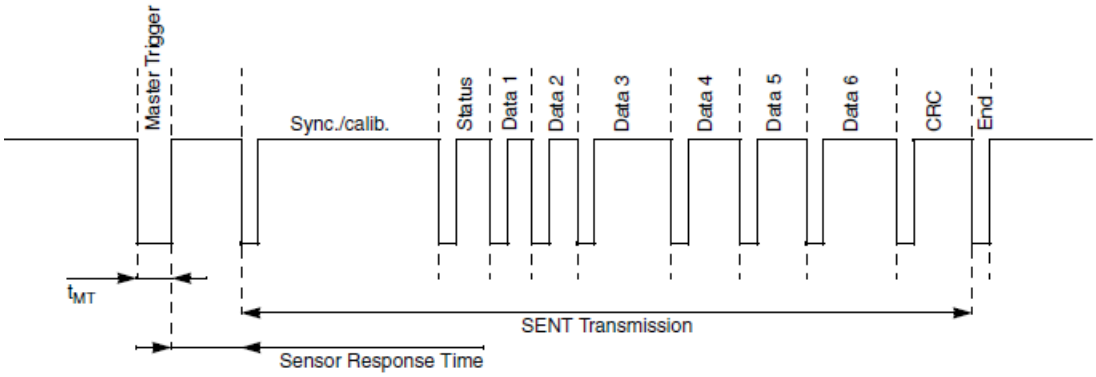
19. ábra: SENT adatátvitel példa 16 + 8 bitnyi adatra [9]

A kommunikáció 4 pulzusra bontható (19. ábra). Első a szinkronizációs vagy kalibrációs pulzus, ami 56 időegység (unit time) hosszú. Ez a pulzus az adatküldő szenzor egység idejéről ad információt, azonban az órajel 20%-os szórására rá tudjon szinkronizálódni a fogadó eszköz. Az első 3 vagy több időegység alacsony állásban, digitális nullában van, a pulzus további része digitális magas értékben van. Ezután jön a státusz pulzus, amely magában hordoz 4-bitnyi információt, ami szenzoronként változhat, tartalmazhat pl.: a szenzor hibajelzési, üzemmódbeli információiról üzenetet. A harmadik megjelenő kommunikációs egység az adatok pulzusait tartalmazza. Ezekben 4 bitnyi szenzor adat van, a nibble-k száma pedig akár a hatot is elérheti. Maximum 6 ilyen nibble kerülhet átadásra egy SENT adatátvitelben. Az adatátvitelben használt nibble-k száma érzékelőtől és gyártótól függően változik. Az utolsó ilyen nibble az ellenőrző összeg pulzus, amely 4 bit CRC-t (ciklikus redundancia ellenőrző, hibadetektáló kódot) tartalmaz. Ez egy polinomiális képlettel számolja ki az adatokat küldő pulzus sorozatok összegét [9][10][12]. Így a fogadó oldal ellenőrizni tudja, hogy valóban a küldött adatot olvasta e hiba nélkül.

3.3.1.2 SPC protokoll

Az Short PWM Code (SPC) [9] protokoll egy módosított SENT protokollos kommunikáció, amely egy félduplex (half duplex) szinkron kommunikáció (20. ábra). Half duplex kommunikációról beszélünk, ha a felek csak felváltva kommunikálhatnak, teljes duplexről beszélünk, ha egyszerre. Az adat vevője (MCU) generálja a trigger (Master Trigger) impulzust a kommunikációs vonalon úgy, hogy azt meghatározott ideig alacsony szintre húzza. Az impulzus szélességét méri az adó (érzékelő) és akkor indítja a SENT átvitelt, ha annak szélessége bizonyos határokon belül van. A csatorna a

kommunikáció lezajlása után magas állapotban marad egészen a következő Master Trigger impulzusig.



20. ábra: SPC adatátvitel példa [9]

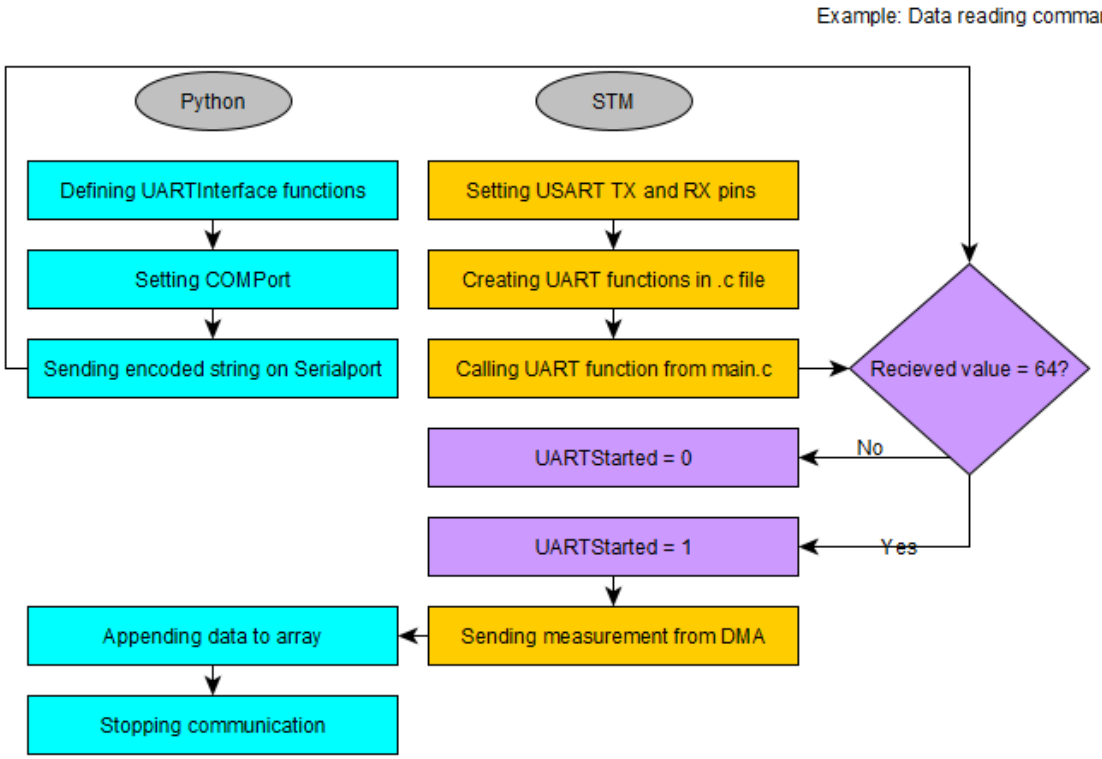
Az SPC protokoll használatakor többféle triggerelési mód választható. Ilyen a Szinkron mód, mikor egy szenzor csatlakozik a vevőhöz, és egy rövid Master Trigger pulzus adott szélességgel indítja az átvitelt. Egy másik mód a szinkron mód tartomány kiválasztással, ahol szintén egy szenzor csatlakozása esedékes, azonban a trigger szélessége megadja a kiváltott átvitel mágneses tartományát. A harmadik pedig az ID választási szinkron átviteli mód, ahol akár 4 szenzor is csatlakozhat párhuzamosan a kommunikációs csatornára és a trigger szélesség határozza meg, hogy melyik szenzor kezdheti az adatátvitelt [9][11].

A mikrokontroller programjában az SPC protokoll megvalósítása egyszerűbb, mert bizonyos, dedikált trigger pineket kell kiolvasáskor "magas" majd "alacsony" állásba. Amennyiben a pineket push-pull beállítással használtuk, hogy teszteléskor egy felhúzó ellenállással is tudjuk tesztelni a kimenetet. Ekkor a trigger pin-ek 3.3V-ra ugrottak, ami a tranzisztor adatlapja szerint elég ahhoz, hogy az nyitott állásba menjen át. A beültetés módosítások a 4. fejezetben kerülnek kifejtésre.

Az SPC-vel szemben a SENT protokollnál a szinkronizáció a Sync impulzus és a Unite time hosszának ismeretéből valósul meg. Itt az adatoknak azonban egy cirkuláris buffert kell fennttartani, hogy kiolvasás után a bejött adatokból utólag értelmezzük a konkrét jeleket. Itt a kommunikáció létrejöttéhez nem kell külön, az STM board által kiadott (trigger) jel. A dolgozat további részében SPC kommunikációt használó szenzorral dolgoztam.

3.3.2 STM – PC kommunikáció kihívásai

A számítógép és az STM board közötti kommunikáció soros porton keresztül UART interfészen valósul meg. Atollic segítségével az STM board USART kommunikációs pinjei beállításra kerültek. Ehhez rendelkezésre állt egy korábban használt kód, csak a megszakítás kérés (Interrupt Request Handler) részt kellett módosítani annak megfelelően, hogy számítógép felől érkező parancsokra hogyan reagáljon a mikrokontroller. A számítógép oldalán Python kódot használok, ahol egy osztályon belül definiálom a függvényeket, amik kiküldik az adott jelet, aminek különböző értékére a board az implementált módokon reagál. A jelek értékei ASCII táblázatból olvassuk ki, mivel a Python soros kommunikációs könyvtárának használt verziója karakter bemeneteket vár el.



21. ábra: Példa adatkiolvasás STM-PC kommunikáció

A 21. ábra az a példa látható, amint a kiolvasás folyamán az adat eltárolódik a PC-ben, annak hatására, hogy a PC-vel kiküldtük, milyen adatra van szükségünk. A folyamatos Pythonos új adatküldés között eltelt idő a PC-s környezet korlátai miatt nagy varianciával rendelkezik és átlagban 16 ms körül van, viszont például 250 µs-enként is képes üzeneteket fogadni. Így az adatgyűjtés nem kérésenként egy mérési pontot tartalmaz. Hanem az adat kérésétől kezdve folyomatosan érkeznek az üzenetek az STM

boardtól, majd egy másik paranccsal megállítható a mérés (kommunikáció). Így nagyobb mintavételi sebesség mellett több adatból pontosabb kalibráció végezhető. Ezen felül más ASCII kódok kiküldésével állíthatjuk a relék állapotát is, vagy módosíthatjuk a protokoll kiolvasásának módját is (pl.: amennyiben a projekt nem SPC hanem SENT protokollal rendelkezik, az UARTon kiküldött kód módosul).

3.3.3 PGSISI - PC kommunikáció kihívásai

Ez a kommunikáció Python scripttel valósult meg. Már rendelkezésre állt egy korábban használt Python driver modul a PGSISI-2 eszközhöz, amit a nyomatékszenzor programozásához használunk. Abban a script-ben egy PGSISI osztály van definiálva, amiben már vannak metódusok pl.: getFwVersion, getError, setprotocol stb.. A driverben ezek mellett, található függvény az EEPROM memória cím szerinti írására és olvasására, ez azért fontos, mert a nyomatékszenzor beállításainak adatait az EEPROM mapből olvashatjuk ki, illetve annak átírásával programozhatjuk át. A szenzor adatlap alapján tudjuk milyen memóriacímen milyen adatok vannak tárolva, lásd: 22. ábra, ez azonban csak az SPC-s szenzorokra érvényes. A SENT-es EEPROM map annyiban különbözik, hogy a 0x16-os címen nincs a protokolloknak 2 bit, hanem a nem módosítható (’do not modify’) bitsor van meghosszabbítva.

ADDR	Description	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
10_H	Parity of each column	P_C	P_C	P_C	P_C	P_C	P_C	P_C	P_C	P_C	P_C	P_C	P_C	P_C	P_C	P_C	P_C
11_H	IC lock high, clamping high/low	P_I	LH	CH - Clamping high (bit 5...0)						CL - Clamping low (bit 5...0)						F (1...0)	
12_H	Gain	P_I	G - Gain (bit 14...0)														
13_H	Offset	P_I	OS - Offset (bit 14...0)														
14_H	ID, precal status, predivider, TQ value	P_I	ID		PC	Prediv (bit 3...0)				TQ - quadratic temperature coefficient (bit 7...0)							
15_H	Bandwidth, Range, TL value, IC lock low	P_I	BW (bit 2...0)			R (1...0)		TL - linear temperature coefficient (bit 8...0)									LL
16_H	Prot, TT value (do not modify)	P_I	Prot (1...0)		Reserved - do not modify								TT - cubic temp. coefficient (4...0)				
17_H	Reserved	P_I	Reserved - do not modify														
18_H	Reserved	P_I	Reserved - do not modify														
19_H	Reserved	P_I	Reserved - do not modify														
$1A_H$	Reserved	P_I	Reserved - do not modify														

22. ábra: Nyomatékszenzor EEPROM Map az SPCs szenzoron belül

A kiolvasáshoz és az íráshoz is először el kell érnünk az EEPROM-ba írt biteket. Kiolvasáskor ebből fejtjük vissza a beállításokat, azonban íráskor az eredetihez képest szeretnénk a pár bitnyi módosítást véghez vinni, mivel nem szabad módosítani a többi

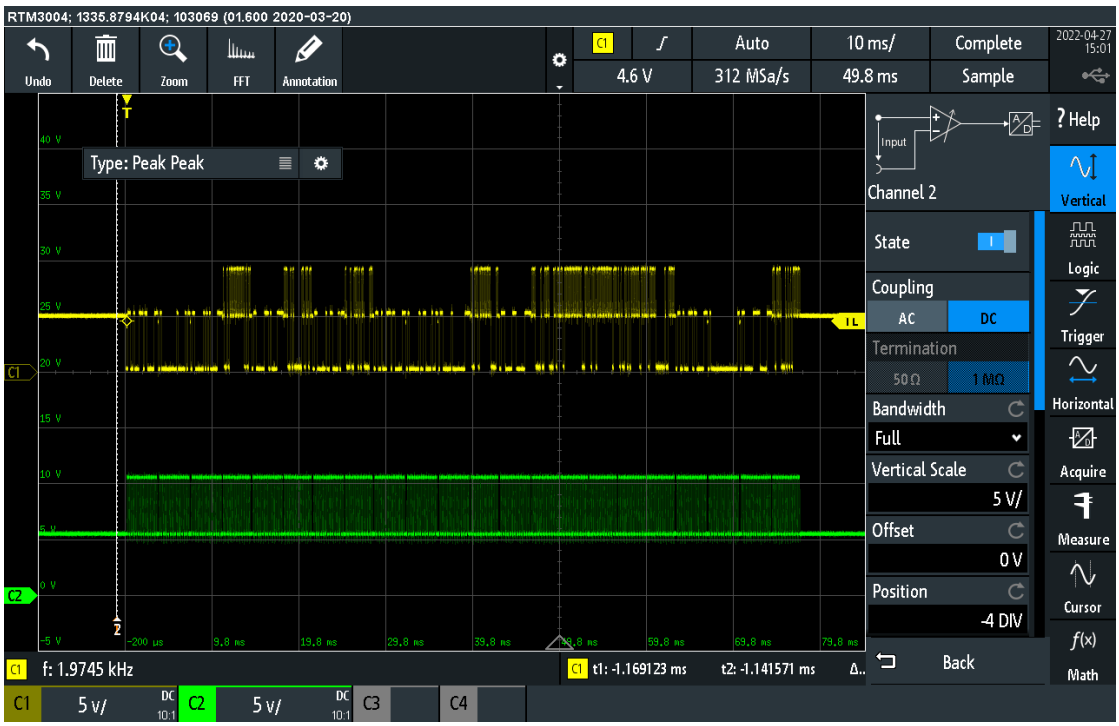
részt. Az EEPROMban minden memóriacím első bite paritás bit, ez a sorokban lévő egyesek számára vonatkozik, emellett a 0x10 memóriacímen található bitek az oszlopok paritásbitjeinek felelnek meg. Amennyiben bármelyik paritás bit eltér a sorban vagy oszlopban tapasztalhatótól, akkor paritás hibát kapunk.

Különböző hibaüzeneteket kapunk, ha paritás (parity) hibás egy EEPROM sor, ha nem tudunk csatlakozni a COM porthoz, mert az eszköz nincs a USB portra kötve, illetve, ha a PGSISI ugyan csatlakoztatva van, de a VDD és GND bemeneteket felcseréltük. Emellett paritás-hibák esetén az EEPROM-ban tárolt adatot nem tudjuk kiolvasni az Evalkittel.

Paritás hibát tapasztaltam, mikor a nyomatékszenzorral egy buszon volt a szögszenzor. Az Evalkiten keresztüli nyomatékszenzor megszólításkor nem lépett fel ez a probléma, azonban a Python kódon keresztüli EEPROM kiolvasásnál megjelent. Több projekt szögszenzor nélküli, ezért olyan szenzorral folytattam a kalibráció tesztelését, ahol ez a probléma nem jelent meg. Ennek oka, hogy a szögszenzor programozásáról és működéséről kevés információnk van, a kimeneti jelek értelmezésén kívül. Látszólag parity hibát kapunk, amit azzal próbáltam megoldani, hogy egy hibahatár (error limit) segítségével addig próbálkoztam az EEPROM Map újrabeolvasásával, ameddig az hiba nélküli sort nem kapok. Ha ez nem sikerül, akkor pedig visszakapok egy „over error limit” üzenetet. Feltételeztem, hogy a PGSISI is hasonló módon kerülheti ki a tapasztalt kiolvasási hibát. Azonban, ha az error limitet 20-ra tettem, majd 50-re, ezzel lényegesen késleltetve a kiolvasás gyorsaságát, akkor sem szűnt meg a probléma. Oszcilloszkóppal vizsgáltam a jeleket, feltűnő szintkülönbséget keresve a PGSISI-n keresztüli, illetve a Pythonon keresztüli megszólítás között, azonban jelentős különbséget nem találtam az ábrák között lásd: 23. ábra, 24. ábra.

23. ábra: Pythonon keresztüli kommunikáció jelei a nyákon keresztül

Mindkét ábrán a C1-es csatorna (citromsárga) az egyik, szögszenzoros csatorna jeleit, a C2-es csatorna (zöld színnel) a táp oldal jeleit mutatja.



24. ábra: Közvetlenül a PGSISI-re kapcsolt kiolvasási jelszintek

Az oszcilloszkóp képen nem látszott különösebb eltérés a szintek között, láthatóan 5 és ~10 V között váltakozik a feszültségszint. Azonban a különbséget a felfutásokban,

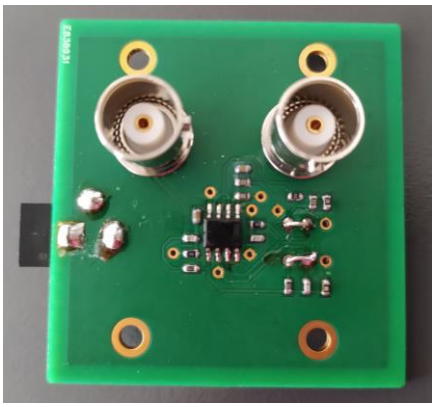
azok meredekségében, esetleg időzítésében kell keresni, mert ha látszólag nem feltűnő a különbség, a részletes vizsgálat más eredményt hozhat ki. A PGSISI közvetlen csatlakozásakor nincs probléma, így az eszközön belül vagy hardveresen vagy szoftveres úton oldódik meg a probléma. Előbbi azért zárható ki, mert plusz áramköri elem a nyomtatott áramkörre sem került a PGSISI és a Szenzor közé a kapcsolón kívül. Egy relé kapcsolásával jön létre közöttük az összeköttetés, semmilyen plusz szűrés nem került rá a NYÁKra, a hasznos jelek kiszűrésének elkerülése miatt. Ez a probléma még megoldásra vár, azonban a kalibráció pontosságának meghatározásához olyan projektre fókuszáltam, amiben nincs szögszenzor. Viszont a szögszenzor jelenléte a nyomaték szenzorral párhuzamosan kötve csak a szenzor és PGSISI közötti kommunikálást befolyásolhatja. Amely mind az Evalkit mind a Pythonos PGSISI vezérlés közben megegyezik. Tehát, mivel az Evalkittel képes párhuzamosan között nyomatékszenzort jól olvasni a PGSISI, akkor megfelelő PGSISI-re vonatkozó utasításokat kiadva a Python kódból a buszos szenzorbekötések is kezelhetőek kell, hogy legyenek a jövőben.

4 Implementáció

A következő fejezetben a részegységek külön-külön történő tesztelését, illetve a felhasználói felület, a háttérben futó program és a számolás részleteit fejtem ki.

4.1 Külön tesztelt áramkörök

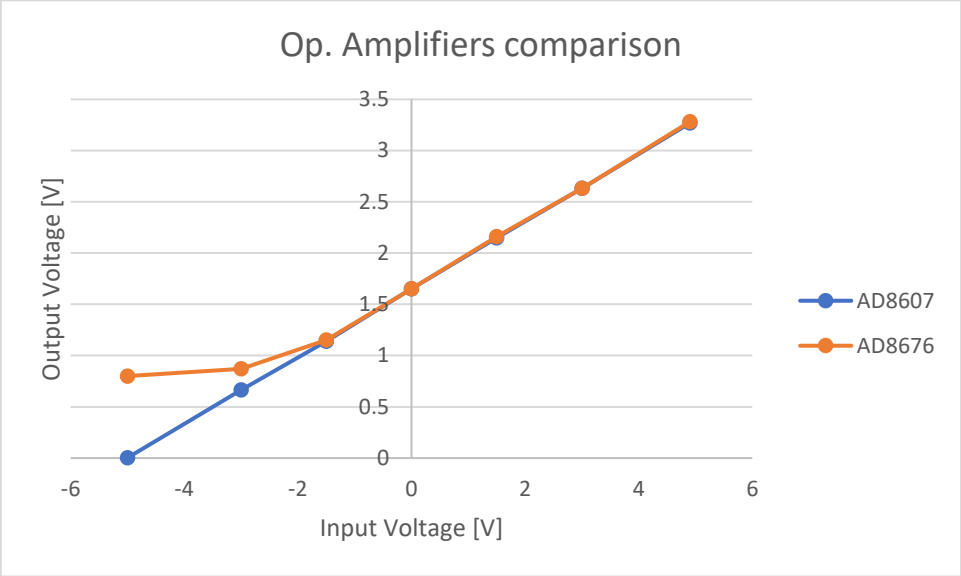
Mind a feszültség átalakító, mind a SENT/SPC interfész esetében már rendelkeztünk NYÁK-kal, ezért ezek a részegységek programozása hamarabb elindult, minthogy a tervezett NYÁK megérkezett volna.



25. ábra: Feszültség átalakító panel

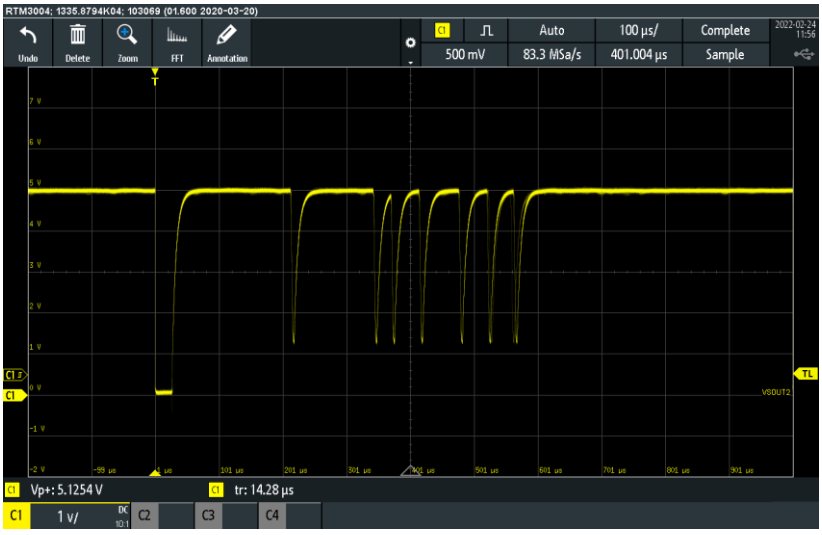
A feszültség átalakítót megvalósító board-ot (25. ábra) tápra kötöttem és a kimenetét oszcilloszkópra, majd a bemenetet változtatva olvastam le, hogy a kimenet miként változik. A táp tizedes pontosságú kijelzésre képes. A kimenetünk 5V-ra 3.3-3.4V volt, míg 0V-ra 1.6V körüli értéket adott ki, ezzel mutatva, hogy a teljes tartomány valóban az STM board számára feldolgozható tartományba került át. A panel eredetileg egy AD8676-os sorszámú dual erősítőre lett tervezve, az LTSpice [19] szimulációban lévő .cir modell is ezt valósítja meg, valamint minden dokumentációban ez az integrált áramkör jelenik meg. A mérések azonban nem igazolták a szimulációt, habár a megépített áramkör megegyezett a szimulálttal. A szimulált kimenet lineáris kapcsolatot mutatott a bemenet és a kimenet között, azonban a mért értékek, ahogy azt a 26. ábra mutatja, a negatív szélső érték felé szaturációt mutattak. Az adatlap alapján nem kellett volna ilyen kimenetet kapnunk, azonban a korábbi munkatársak által épített panel a kívántak szerint működött. az AD8607-es sorszámú dual erősítővel. Az adatlapok között lényeges eltérést a differenciális bemeneti feszültségtartományban találtam. A 8676-os modellnél ez a paraméter ± 0.7V a 8607-os modellnél ± 6V. A negatív tartomány miatt ez a küszöbérték

átlépésre kerül, ezért nem teljesült az ígért karakterisztika. A Spice modellben azért működhetett a szimuláció, mert a modell a karakterisztikának megfelelő működést produkálja, azonban a abszolút maximum alkalmazási tartományokat a használónak kell betartania. Ezen megfontolások mentén, kicseréltem a két erősítőt. Ezután a verifikáló mérések során is teljes tartományon tudtuk mérni a mérőkormány ±5V-os kimenetét. Az eredeti erősítő működésének egy ábrája is bekerült a Függelékbe (55. ábra).



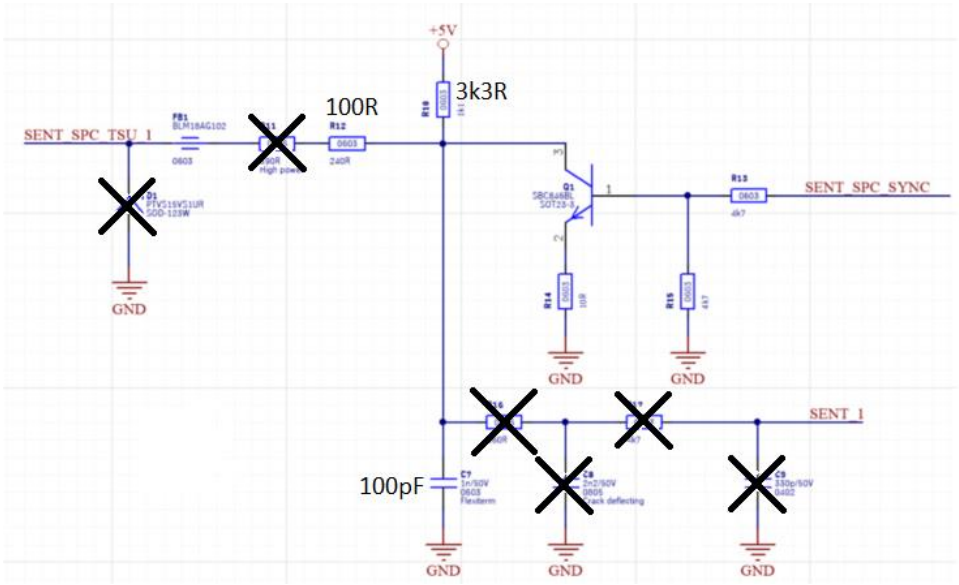
26. ábra: Két erősítő összehasonlítása

Az SENT/SPC interfészen keresztüli kommunikáció felállításában probléma lépett fel, mikor a szenzor nem tudta 0V-ra lehúzni a jelek alacsony szintjét (lásd 27. ábra). Amikor pedig nagyobb értékű felhúzóellenállás került beültetésre, a jel magas szintje nem haladta meg a mikrokontroller bemeneti magas szintjét.



27. ábra: Oszcilloszkóp kép a nem megfelelő alsó jelszintről

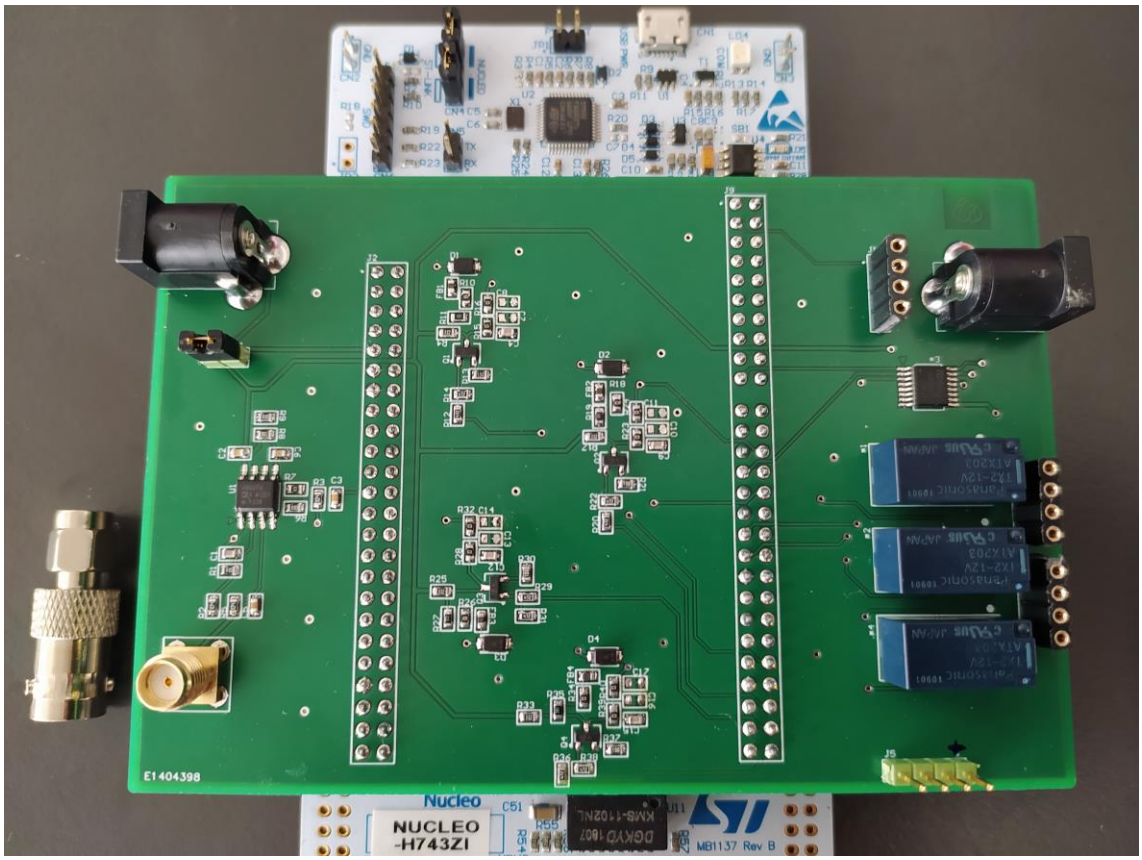
LTSpice szimulációt futtattam, különböző felhúzó ellenállásokra illetve az illesztő áramkör egyéb komponenseinek cseréjére. A módosítások előtti oszcilloszkóppal mért működése az áramkörnek a 27. ábraán látható. A ki nem olvasható jelszintek, ahhoz vezethetők vissza, hogy az interfész az ECU-khoz használt processzor jelszintjeihez volt igazítva, melyek különböznek az STM digitális jelszintjeitől. A korábbi jelszintek 0-5V küszöbértékeire (threashold) lettek beállítva, azonban az STM board 0-3.3V-os tartomány küszöbértékei máshova esnek. Ennek megfelelően kellett az interfészt módosítani. A módosításokat a 28. ábra fekete feliratai, illetve X jelölései mutatják. Az LTSpice szimuláció módosítás előtti és utáni lefutásának eredményeit a Függelékben található 50. ábra és 51. ábra mutatja.



28. ábra: Szimulált SENT/SPC interfész

A módosításokat a végső nyákon is eszközöltük, azonban a végső nyákterven megmaradt a korábbi áramkör. Annak megtartása mellett szólt, hogy az ültetési opciók megmaradjanak, és ne kelljen új NYÁK-ot tervezni, hogy tesztelhessünk. A kihúzott ellenállások helyére rövidzár került, a kondenzátorok helyére nem ültettünk semmit. Az SPC-s kommunikáció kiolvasása is megtörtént, a szenzor fölött mágnest forgattam, ekkor kicsit a mérési körülményekből adódóan torzulva, de a várt jeleket láttam (lásd: 52. ábra).

A Relék működése multiméterrel volt leteszteltve, hogy adott tápellátás mellett az átkapcsolás megtörténik-e, vagyis a megfelelő két láb van-e rövidzárban. Az áramfelvétele a reléknek 10 mA volt, míg a minimális feszültség, ahol még átkapcsol a relé 8V-nál volt. A felső feszültséghatárt roncsolódás lehetősége miatt nem teszteltem le.



29. ábra: Beültetett nyák, mellette SMA – BNC átalakító

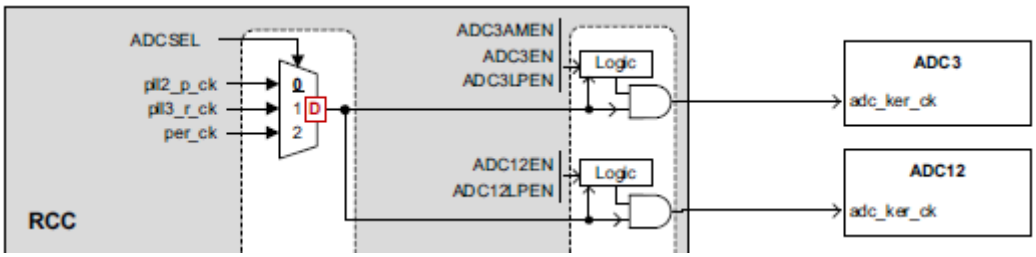
A fenti ábrán (29. ábra) látszik, hogy a tervezett nyák miként illeszkedik az STM board kimeneteire. Ez lehetővé teszi, hogy akár később felprogramozott kimenetek is használhatóak legyenek.

4.2 STM vezérlés

A NYÁK-hoz szükséges részegységeket külön-külön teszteltem, így az STM-hez kellő C kód is részegységeiben került megírásra. Az STM board az adatok továbbküldésére szolgál a PC felé, a számolás Python kódokra van bízva, hogy az STM board ne legyen túlterherve, illetve ne veszítsünk felesleges időt számolással, mivel az adatok közel egyidejűsége fontosabb tényező. Az adatok egyidejűsége miatt, egyedül az analóg-digitális konverter beállítása az, ami számottevően hathat a mérőkormányból nyert értékekre.

Az ADC beállításaihoz szükségem volt az órajelek ismeretére, ehhez a CubeIDE programnak clock setting felületét használtam. Megnéztem az Atollicon belül az egyes kérdéses bitek beállítását, hogy mennyivel osztódik le vagy szorzódik fel az aktuális órajel.

Az órajel értékek ismeretében, illetve az adatlap alapján megállapítottam, hogy az ADC-nek mekkora órajel jut. Ehhez az adatlapban található útmutató segített, lásd: 30. ábra.

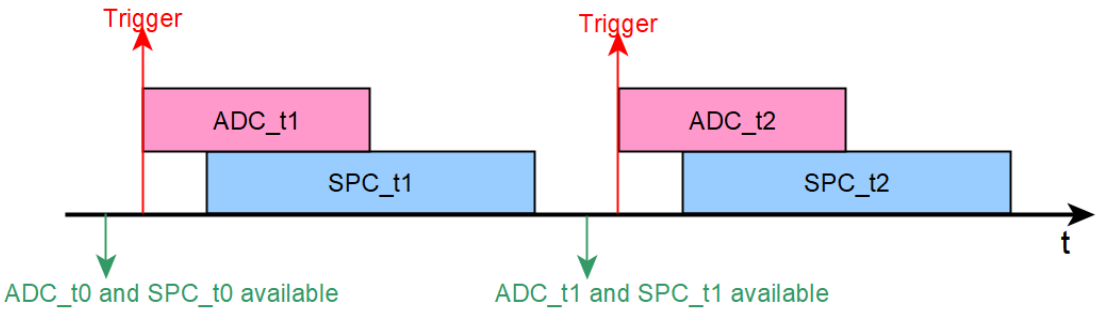


30. ábra: Clock for ADCs

Az ADC órajelét a per_ck-tól kapjuk, ami 200 MHz, ezt kettővel osztva kapjuk meg az ADC órajelét, mivel ez 100 MHz, vagyis 20 MHz-nél nagyobb, a Boost bitet ennek megfelelően kell beállítani. A referencia feszültséget beállítottam, hogy külső referencia feszültségen legyen, ami a VREFBUF értékeinek default módra állításával történt. Ekkor a buffer, ami a belső referencia feszültség buffer-t jelenti, kikapcsolásra került, és a Vref+ pin pedig bementi módba váltódott. A pontosabb értékhez túlmintavételezést alkalmazunk, ez esetben a bevitt értékek kiolvasása pontosabb lett, bár így is volt benne egy nagyobb szórás, de ezt a tápegység pontatlanságának tudtam be. Az ADC inicializáció lefut az STM kód elején, a többi részegység (periféria pl.: órajel beállítások, GPIO, stb) inicializációjával együtt. Ebben a részben történik az ADC kalibrációja is.

Az ADC kiolvasása úgy zajlik, hogy az ADSTART bit 1-essé állítása után az ADC mintavételezi, konvertálja, majd beírja a DR regiszterbe a mérés eredményét és onnan kiolvasva továbbítja az STM a mért értéket UART-on keresztül. Az ADC mintavételezése is eredetileg ez az ADC_Read() függvényben valósult meg, amit az SPC-nek szánt GPIO trigger után hívtam. Azonban azt tapasztaltam, hogy az SPC-s adatkiolvasás nem megfelelő. A kiolvasás és triggerelés task 1 ms-enként hívódik, amíg az UART-on utasítást nem kap az STM board a folyamat megszakítására. A probléma az volt, hogy az ADC mérés nagyon sokáig tartott, a ciklus lefolyása több időt igényelt, mint amilyen gyorsan lefutni képes, viszont egy while ciklussal vártam az ADC mérés végét, hogy az eredmény tovább küldhető legyen. Így az említett függvényt két részre bontottuk, külön ADC trigger és külön az adatkiolvasás részre, majd a kiolvasást a ciklus elejére tettük, amikorra biztosan lefut az ADC mérése, az SPC megvalósításához hasonlóan. A problémánk megoldódott, valóban az ütemezést túllépő futásidő volt a probléma. A 31.

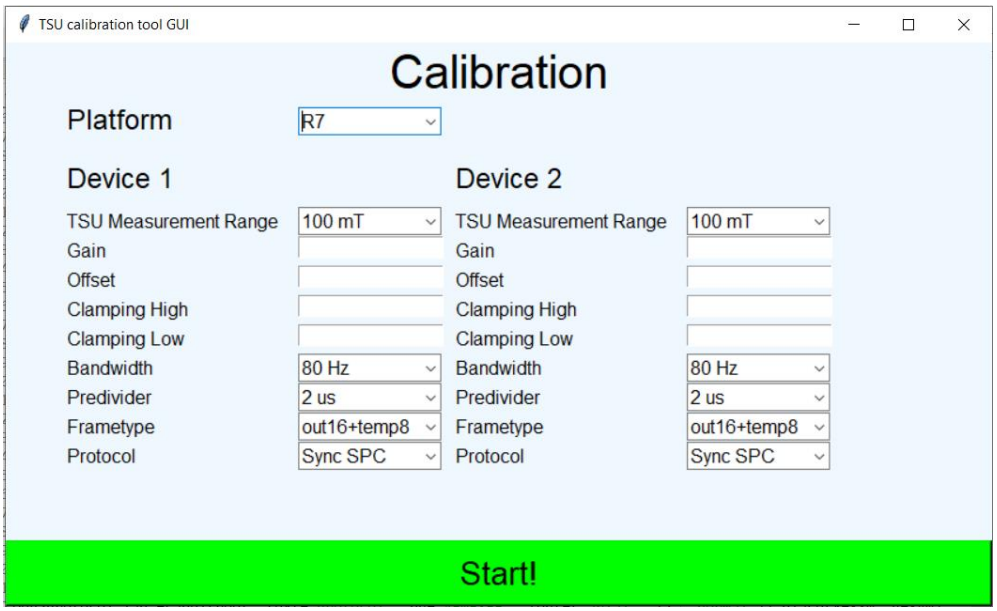
ábra szemlélteti a mérés és kommunikáció időzítését. Ezen az ábrán látszik, hogy a Trigger után megtörténik az adatok továbbítása, de ezen folyamat alatt mindig az eggyel korábbi adathoz férünk hozzá, melyek koherens időpontban voltak mintavételezve, ezzel csökkentve a késleltetést a nyomaték szenzor és referenciája között.



31. ábra: SPC, ADC időzítés

4.3 Grafikus felhasználói felület

A kalibrációhoz általam létrehozott felhasználói felület célja (ld.: 32. ábra), hogy a kalibrációt végző személy, a projekt kiválasztásával, megadja, mely paraméterekre számíthatunk a nyomatékszenzortól. Ekkor beolvasódik a nyomatékszenzor mérési tartománya, ami elengedhetetlen a kalibrációhoz. Ez a felület, még fejlesztés alatt lévő projektnél is lehetővé teszi a különböző értékek beírását. A megvalósításhoz a Python tkinter nevű könyvtárát használtam.



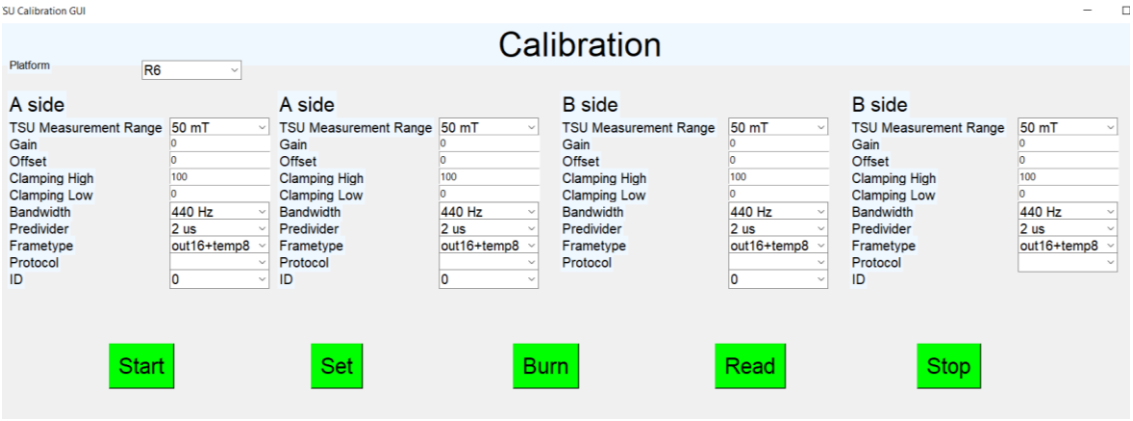
32. ábra: Felhasználói felület a kalibrációhoz

Az 32. ábra mutatja az eredeti prototípust, ami a munka első félévében alakult ki. Ennek a felületnek az alapja az 6. ábra által mutatott Evalkit program.

Ez a felület a későbbiekben módosult, hogy láthassuk egyszerre az összes programozandó szenzor értékeit. Erre azért van itt lehetőség, mert a relékkel váltva megtörténik az átkapcsolás a felprogramozó irányába. Korábban, az egyik oldal kiolvasása után le kellett csatlakozni a PGSISI eszközről, majd a kábelek átkötése után a másik oldalra csatlakozni, így nem láttuk az összes beállítást egy helyen. Emellett a következő gombok is megjelent, az alábbi funkciókkal:

- Set gomb: A felületre vitt (akár kézzel) értékek eltárolás egy mátrixban, aminek értékei már programozhatók a PGSISI-vel.
- Burn: A Set-elt értékek paritásának ellenőrzése után, azok beírása a szenzor EEPROM-jába.
- Read: Az A (Device 1, 2) és B (Device 3, 4) oldal kiolvasása, mátrixban tárolása, illetve a GUI felületen történő megjelenítése.
- Start: A kalibrációhoz kellő mérés kezdetét jelzi, ezután folyamatos az adatkiolvasás az STM board-on keresztül, mind az ADC (referencia) és az nyomatékszenzorok felől.
- Stop: A mérés leállításának gombja, ezután kiíródik a mérés hossza, a kiszámított gain és offset értékek delta értéke a korábbi beállításhoz képest.

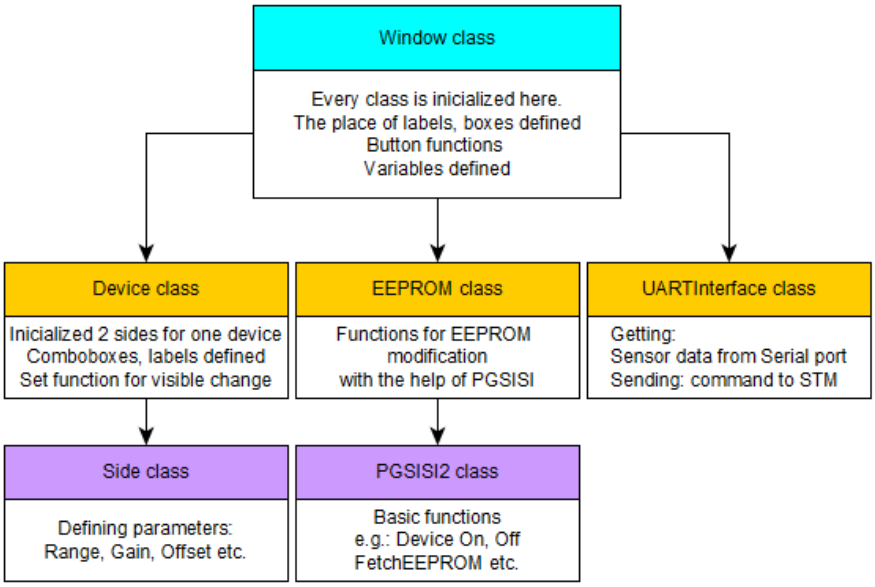
Mivel van külön A és B oldalunk, a relé kapcsolásra szükség van, a Burn és a Read parancs használatakor. A Set gomb funkciójához azért nincs rá szükség, mert a kiolvasott értékeket egy lokális mátrixban tároljuk. Ennek oka, hogy ha a programozóban beállítunk egy értéket, anélkül, hogy az EEPROM Burn megtörtént volna, a táp elvesztésével, a paraméter beállítás elveszik. A relék kapcsolásánál ez a tápvesztés megtörténik, ezért kell a mátrixos eltárolás.



33. ábra: Változtatott, javított felület

Gombnyomás után a háttérben futó folyamat blokkolja a felhasználó felületet. Azonban a mérés közben el kell érni a Stop gombot, hogy a mérés leállítható legyen. Ennek megoldására a threading könyvtárt használtam, a folyamatok párhuzamosítására. A mérési adatok kimentése fut másik szálon, így a GUI még mindig elérhető, és a Stop gomb megnyomható. A Stop gomb hatására a egy változó értéket változtatunk, ami a mérés while ciklusában kerül vizsgálatra, ezek után fejeződik be vagy folytatódik.

A program, amit a felület elindításához használok, több Python scriptből hív meg osztályokat. A fő programban inicializálom a Window, a Device és a Side osztályokat. Egy másik kód tartalmazza az EEPROM osztályt, ami a PGSISI2 osztály egy objektumának létrehozásával kezdődik. A harmadik kódban az UARTInterface osztály van definiálva, ami a soros porton az STM felé történő kommunikáció lebonyolításáért felelős.



34. ábra: Osztályok és funkciójuk a Pythonon belül

A 34. ábra mutatja az egyes osztályok viszonyait és funkcióit. A 3.3.3 PGSISI - PC kommunikáció kihívásai fejezetben említettem a PGSISI osztályt tartalmazó script-et, ami az eszközhöz szükséges funkciókat tartalmazza például: fetchEEPROM, DeviceOn, DeviceOff, getEEPROM, chack_and_correctEEPROM stb. Az EEPROM osztályban már teljes EEPROM Map mátrixban tárolására írt függvény van, ami egy dictionary segítségével fejti vissza a kiolvasott bitek beállítás megfelelőjét. Emellett itt található még olyan függvény, ami a megadott beállítás (pl.: Frametype, Protokol stb.) alapján az adott mátrix megfelelő részeibe átírja a biteket. A Side osztályon belül az Evalkit programhoz hasonlóan paramétereket definiálok például: mérési tartomány, gain, offset, protokoll stb. A Device osztályon belül kettő Side objektum van létrehozva, ami az A oldal SPC1, illetve SPC2 csatornájának feleltethető meg. A Device osztályban már a megjelenéshez szükséges, viszont eszközönként külön kiírandó paraméterekhez tartozó label, textbox és combobox elemek vannak definiálva. Itt található egy beállító függvény, ami beolvasás esetén állítja a felhasználói felületeken látható értékeket. A Window osztályon belül a, már említett gombok egyes függvényei vannak definiálva, illetve a Device-on belül létrehozott tkinter könyvtár elemeit helyezi el a felületen belül. Itt vannak azok a változók is inicializálva, illetve állítva, amik Platform választás után módosulnak például, hogy hány csatorna van az adott platformon, mekkora a mérési tartomány, mi a kiolvasott mátrix, amit a későbbiekben módosítani kell.

4.4 Gain és Offset számolás

Ez a fejezet a kalibrációs számolásokról szól. A kézi kalibrációtól eltérően itt figyelembe vesszük, hogy az offset gain-függő érték. Ehhez útmutató egyenleteket a szenzor Használati útmutatójából olvastam ki [15]. Kitérek a gain setting (4.2) illetve az offset setting (4.3) fejezetekre is, hogy az egyenlet levezetés érthetőbb legyen.

$$Gain = \frac{(G-16384)}{4096} \tag{1}$$

Ahol G az aktuális gain értéke, ami az EEPROM -ban tárolva van, Gain pedig a [-4, 3.9998] tartományon belül elhelyezkedő, és a felprogramozáskor megjelenő érték. A gain habár 16 biten van tárolva a memóriában, az első bitje paritásbit, ezért 15 biten tárolódik az értéke. Az 16384 a pozitív-negatív tartományba konvertáláshoz szükséges, az osztás pedig a ~8-as sávhoz kell.

$$OUT_{OS} = OS - 16384 \tag{2}$$

Ahol az OUT_{OS} a számolt offset értéket jelöli LSB_{12} formában. A Gain-hez hasonlóan a negatív és pozitív tartomány miatt került levonásra a 16384.

$$DOUT_{16bit} = \frac{2*(G_{EEPROM}-16384)*HCAL}{4096} + 16 * (OS - 16384) \quad (3)$$

A $DOUT_{16bit}$ a szenzor által a kimeneten kiküldött 16bitet jelenti. Itt a HCAL (Hall Calculated) érték a szenzor által mért belsőleg tárolt mért érték, de láthatóan a kimeneti 16 bitbes értéket az offset és gain befolyásolja. A 3. egyenletből kiindulva, tudjuk, hogy ideális esetben a 0 Nm helyen a kimenet 12 biten 2048-at várunk, ekkor ideális a gain és offset érték. A képlet 16 bitre érvényes, ezért kap egy $2^4 = 16$ –os szorzós az érték:

$$HCAL_{zero} = \frac{\left(\left(2048*16-\frac{Offset_{polyfit}}{Gain_{polyfit}}\right)-16(OS-16384)\right)*4096}{2*(G-16384)} \quad (4)$$

A számoláshoz rendeztük a (3.) egyenletet, ebből lett a (4.). Ebből a HCAL értéket az (5.) egyenletébe téve kiszámolható lesz az ideális offset.

$$2048 * 16 = \frac{2*(G_{ideális}-16384)*HCAL_{zero}}{4096} + 16 * (OS_{ideális} - 16384) \quad (5)$$

A gain értékről tudjuk, hogy az ideális értéknek az ismeretében, a következő számolásra jutunk:

$$Gain_{ideális} = Gain_{EEPROM} * Gain_{polyfit} \quad (6)$$

Az (6) egyenlet mutatjak, hogy az aktuális gain ($Gain_{EEPROM}$) érték, miként alakítható az ideális gain ($Gain_{ideális}$) értékre (ezt fogjuk majd az EEPROMba írni), a polyfit által meghatározott elsőrendű koefficiens alapján. A $Gain_{polyfit}$ a mérési adatokon alkalmazott polyfit függvény meredekségét takarja. Az ideális gain számolása azért lehetséges, mert tudjuk, hogy itt nincs offset értéktől való függése a paraméternek. A korábban említettek szerint a gain-t, mint meredekséget értelmezzük, az offset-et pedig az y tengely metszéseként, belátható, hogy az, hogy az y tengely hol kerül metszésre, nem befolyásolja a meredekséget, míg a meredekség változtatása befolyásolja az y tengely metszését lásd: 37. ábra. A segédszámítással rendezett (5.) egyenletből, az offset kiszámolása a következő képpen alakul:

$$OS_{ideális} = \frac{2048*16 - 2*(G_{ideális}-16384)*\frac{HCAL_{zero}}{4096}}{16} + 16384 \quad (7)$$

A számolásokhoz az ideálistól való eltérést a bejövő adatokra használt polyfit segítségével állapítottuk meg. Ez a függvény nyomatékszenzor adatait a referenciához

hasonlítva egy elsőrendű polinomot illeszt az adatokra. Visszatérési értékként pedig megkapjuk az egyenes meredekségét, illetve a metszéspontját az y tengelyen.

A nyomatékszenzor kimenetébe bele kell számítani a gyűrűmágnes hiszterézisét, ezt kalibrációval nem tudjuk kikompenzálni. A hiszterézisre példá a mérési eredmények között fogjuk látni. Ez úgy fog mutatkozni, hogy a balról jobbra tekert kormány nyomatékjához tartozó szögszenzor értékek akár 0.3 Nm-el is eltérhetnek, mint az ugyanahhoz a nyomatékhoz tartozó, azonban másik irányba tekert kormány esetén, a szenzor értékek. A szenzor kimenetének szaturációja miatt az adatok szélsőérték közeli szakaszait ki kell szűrni, (ki kell hagyni a számolásból), mivel befolyásolják a számításokat. Ha a szenzor kimenete szaturálódik, de a referencia nyomaték változik, nem valós nyomatékhibát fogunk mérni, hiszen ez csak a 12bites kimenet korlátja. Például az utolsó 10 adatpontnál ugyanazt az értéket olvassuk ki, pedig a ráadott nyomaték változik. A 4. ábra által mutatott S-curve hiba miatt is indokolt a szélsőértékekhez közeli mérések maszkolása. Ez viszont nem okoz nagyobb pontatlanságot, mint a gyártósori beállításoknál, ahol szintén egy középérték közeli lineáris szakaszon kerül meghatározásra a gain és offset érték.

Az alkalmazott egyenleteket kipróbáltam olyan adatokra, amik kellően pontatlanok, de még bemenetként elképzelhetők. Ekkor a szimulált értékekre kalibrálás előtti hibára a referencia és a szenzor jelek között peak-to-peak értékre 15 Nm, átlag értékre -2 Nm-ert kaptunk. Ez a számolás után lecsökkent nagyságrendileg 0-ra, pontosabban 10^{-14}-es nagyságrendre. Ez az elméleti határ persze nem elérhető, mert a beírható gain és offset értékek a felbontásnak megfelelően korlátozottak, azonban még a ’round’ kerekítési függvény segítségével is elérthető volt egy 0.001-es pontosság.

5 Verifikáció

A mérések előtt a mérőkormány kimeneteit vizsgálva az derült ki, hogy az ígért ±5V-os kimenet valójában mindkét irányba 5.38 V. Illetve, a mérőkormányra táp adás után nullázni kell azt, különben a 0 Nm-nél 1.82 V a kimenet. A táp kimeneti feszültségét pedig csak tizedesjegy pontossággal tudjuk megadni. Ekkor az áramfelvétel 2 tizedesjegy pontossággal jelenik meg.

5.1 Tesztelési terv

Különböző tesztesetek folyamán a felépített rendszer határait kerestem, működési tartományainak feltételeit vizsgálva, az egyes kalibrációkból adódó százalékos hibák végül összehasonlításra kerülnek. A következő tesztelési lehetőségek merültek fel a verifikáció érdekében:

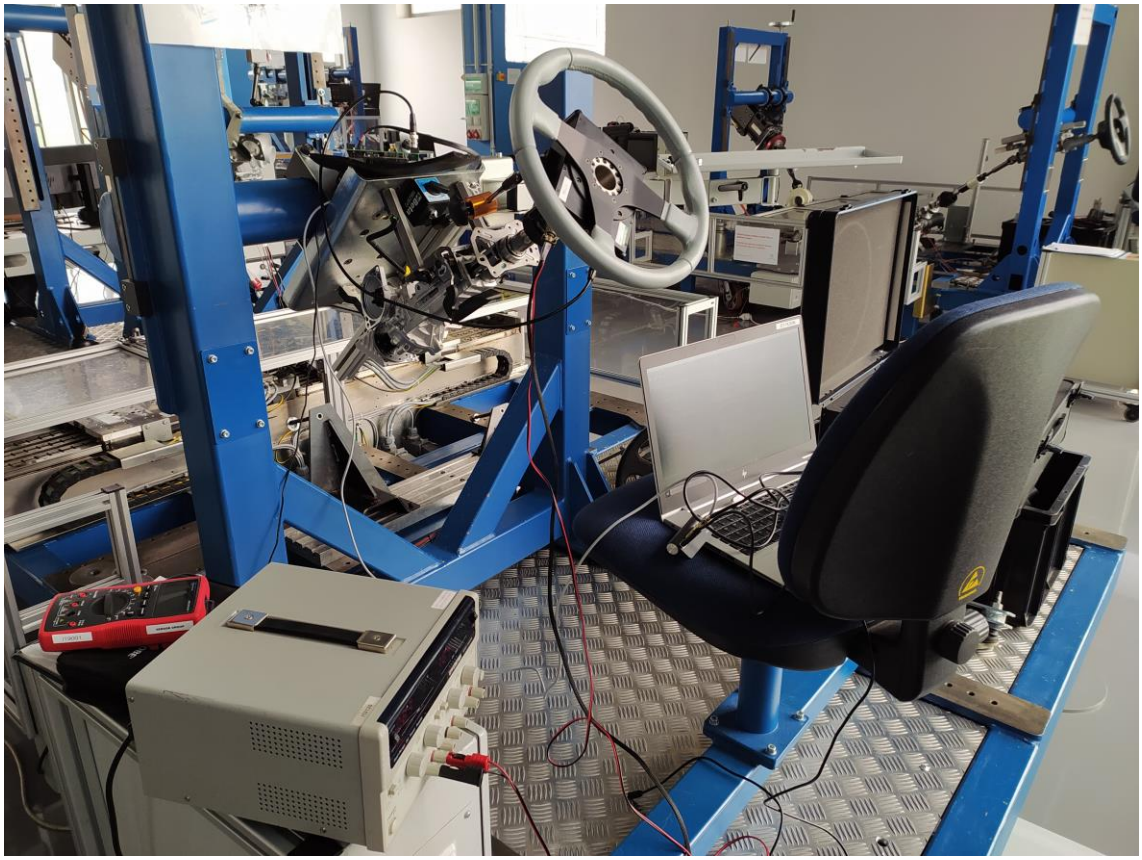
1. Kézi kalibráció, majd ugyanezen a TSU-n automatizált kalibráció elvégzése
2. Forgatás nélküli mérés eredménye
3. Aszimmetrikus forgatások és eredményei

Előzetes ismereteink alapján az offset és gain minimális hibái már a felprogramozás kvantálási lépéseiből jönnek, mivel az általunk beírt, és a program által kerekített értékek nem azonosak. A szenzor User Manual-ja alapján a teljes tartomány gain beállítására ±4.0, amire 15 bit áll rendelkezésre. Ennek a beállításnak a pontossága így 8/32768. Ezt a hibát százalékban kifejteni nem lehet, mert az adott projekt gain értékétől függ, hogy ez a lépték mekkora pontatlanságot eredményez például: egyik projekt 2.5 –es gaint vár, ekkor ez a kvantálási hiba 0.01%-os hibának felel meg.

Offset esetén százalékos értéket tudok programozáskor beállítani, nem úgy, mint a gain esetében, ahol dimenziótlan mennyiségről beszélünk. A kimenetnek 0 Nm nyomatékra 50%-ot kell mutatnia, ennek pontosítására szolgál az offset. Az offsetnél egy 0.4%-os hiba már elhanyagolható a kézi kalibrálás folyamán, mivel ez a 0.4% nyomatékban 0.08 Nm-nek felel meg egy ±10 Nm-es tartományon, ami megfelelően pontos.

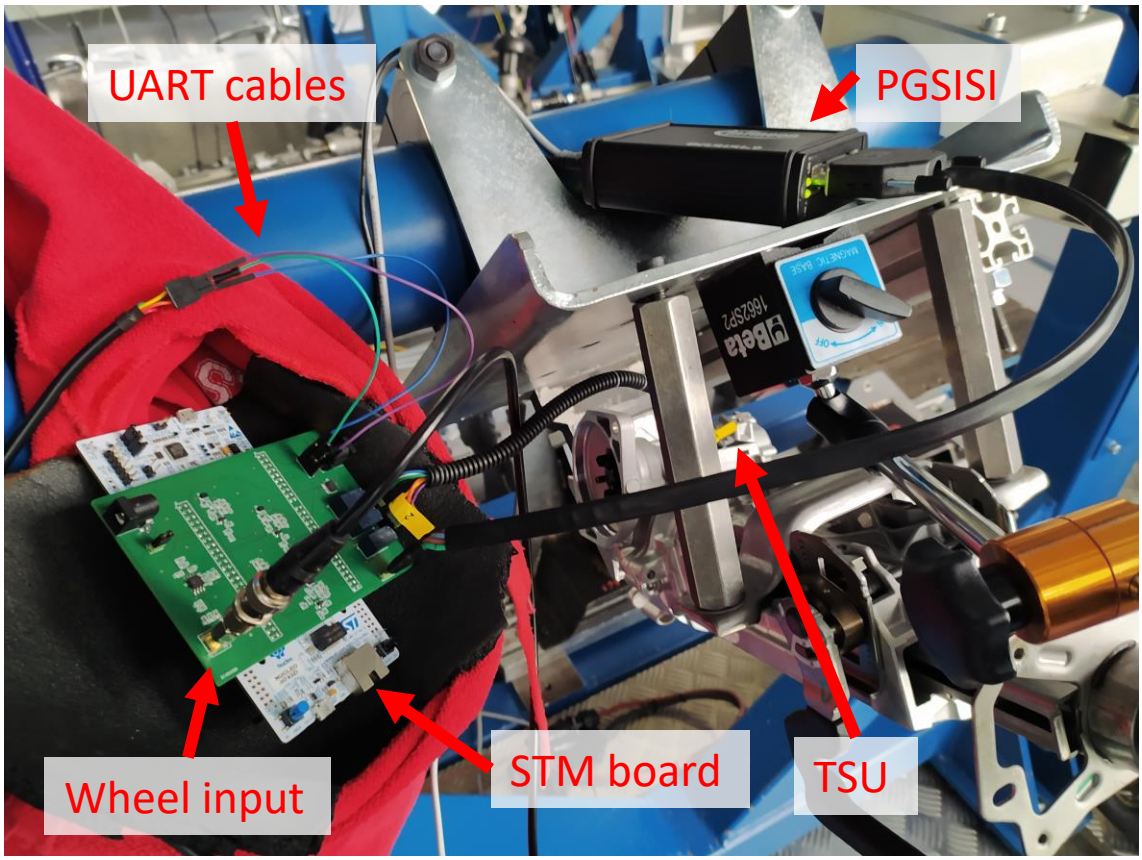
5.2 A mérési összeállítás bemutatása

A teszteléshez használt összeállítás annyiban tér el az 2.4-es fejezetben említettől, hogy a mérés servo unit-on történt, míg a 2.4-es fejezet képén gear látható. A méréshez olyan nyomaték szenzor verziót használtam, ami SPC protokollal kommunikál és nincs rajta szögszenzor, hogy kiküszöböljük azt a hibát, ami egyelőre javításra vár.



35. ábra: Mérési összeállítás a kézi kalibrációhoz

A jelenlegi kalibrációhoz hasonlóan szükségünk van a tápra, ami a multiméter mellett a 35. ábra bal alsó sarkában látható. A multiméterre azért volt szükség, hogy a ’régi’, illetve az automatizált kalibráció különbségeit tudjuk mérni. A szervo unit esetén a kormányoszlop rögzítésre került. A képen még nincs jelen az automatizáláshoz használt panel és STM board nincs az összeköttetésben, mert először a kézi kalibrációt végeztük el. Ennek sorrendisége azért volt indokolt, hogy rosszul kalibrált egységen hiteles legyen a kézi kalibráció. A kézi kalibráció után viszont már a 36. ábra által mutatott panelünk is rész vett a számolásban.



36. ábra: A mérési folyamatba helyezett panel

A 36. ábra bal oldalán látható, hogy az 5 V-os táp nincsen bekötve, vagyis a feszültség átalakító és az interfészek is az STM board 5V-os kimenetéről vannak tápolva. A relékhez szükséges 12V-ot egy hálózati tápkábel adta, de a tápegységnek 2 kimenete van, amiből az egyik 12V-ra van állítva és a mérőkormány felé megy, azonban a másik csatornája még használaton kívül van. Innen is megoldható lenne a relék vagy az STM board 5V-ja helyett a tápellátás.

5.3 Végeredmény

Ebben a fejezetben bemutatom az egyes mérések eredményeit, a felhasználói felület működését, illetve értékelem a tapasztaltakat.

5.3.1 Kézi kalibráció

Az offset beállításához, a korábbiak szerint említve, kell a 0 Nm pontban az 50%-ot beállítani, ezt még a gain vizsgálata előtt megtettem. Az ideális gain értéket a következő képpen számoljuk: ±10 Nm-es nyomatéktartományt vizsgálva az ideális meredekség (gain) amit két mérési pontból már megállapíthatunk, kézi kalibráció esetén ez 2Nm/V (±10/±5=2). Példa számokkal:

	Kimenet [%]	Kimenet [Nm]	Referencia érték [V]	Referencia érték [Nm]
A side	95,28	9,065	3,82	7,64
B side	94,46	8,892	3,82	7,64
A side	37,4	-7,48	-3,05	-6,1
B side	13,65	-7,27	-3,06	-6,1

1. Táblázat: Kalibráció előtti értékek

A 1. Táblázatban látszik az Evalkit programmal %-osan kiolvasott jelek illetve azok nyomaték megfelelői. Ezek mellett pedig a multiméterrel mért referencia érték, ami a mérőkormánnyal a rögzített kormányoszlopra adott nyomaték feszültségbeli megfelelője, valamint ennek a nyomatékra átszámolt értéke is látható a táblázatban. Látszik, hogy az eszközre adott illetve kiolvasott nyomatékértékek között jelentős eltérés van, tehát kalibráció szükséges. A számoláshoz használt egyenletek a következő képpen alakulnak [14]:

$$gain_{mért} = \frac{sensor\ kimenete\ [Nm]}{kormány\ analog\ kimenete\ [V]} \tag{7}$$

$$gain_{korrigált} = gain_{aktuális} * \frac{gain_{ideális}}{gain_{mért}} \tag{8}$$

$$gain_{coeff} = \frac{gain_{ideális}}{gain_{mért}} \tag{9}$$

A fentebb látható adatok alapján :

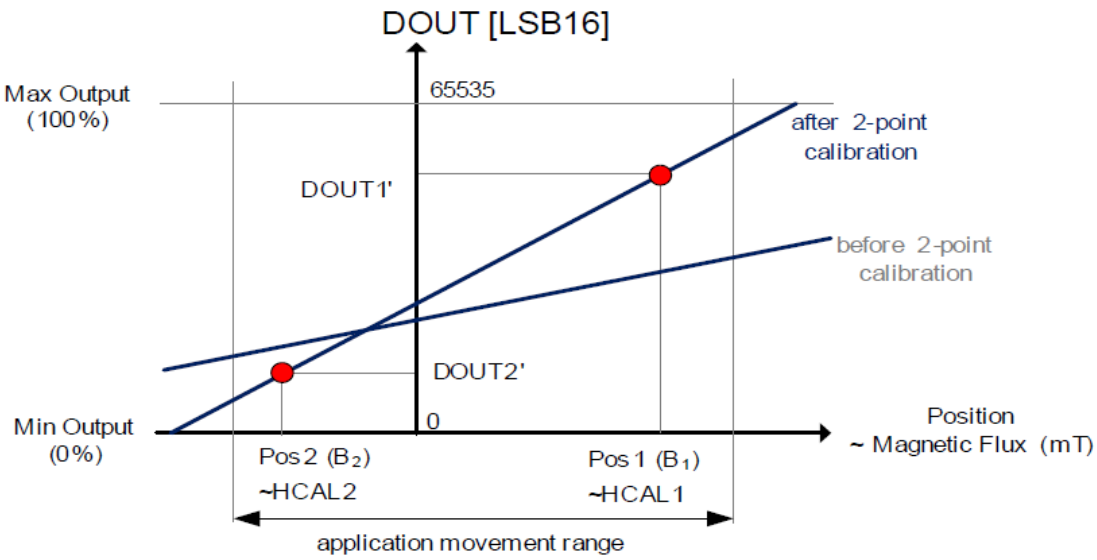
A oldali gain: $\frac{(9{,}065-(-7{,}48))}{(3{,}82+3{,}06)} = 2{,}408.$

Ebből az A oldali koefficiens: $\frac{2}{2{,}408} = 0.83.$

B oldali gain: $\frac{(8{,}892-(-7{,}27))}{(3{,}82+3{,}06)} = 2{,}3525.$

Ebből az B oldali koefficiens: $\frac{2}{2{,}3525} = 0.85.$ Ezekkel a számokkal kell megszorozni a szenzorban található gain értékeket. A korrigálás után az offset értéke változhat, lásd: 37. ábra. Látható, hogy a gain értékének változtatásával az y tengely metszéspontját is változtatjuk, ami a 0 Nm –es nyomaték bemenet pontja, vagyis az offset. Ekkor az offset szélsőségesen változik, ha a gain esetén nagyon nagy hiba volt, ennek elkerülésére azonban gyártósori korlátok szolgálnak. Az azonban előfordulhat, hogy a

PGSISI által használt Evalkit-es programmal valaki kézileg átírja az értéket. Az egyes projektek worst-case analíziséből kiderül, hogy mekkora a tipikus gain érték, illetve, hogy mennyi a worst case érték is, aminél nagyobbat nem szoktunk beállítani. A Gain és Offset számolás fejezetben láthattuk, hogy a szenzornak nem a mért értéke kerül a kimenetre, hanem a gain és offset függvényében, ezért, ha túl nagy az offset hiba, lehet, hogy a mérési tartományból kikerül az ábrázolni kívánt szenzor kimenet.



37. ábra: Kétpontos kalibráció sémája [15]

A gain érték beállítása után még egy ellenőrző mérést szoktunk csinálni, hogy lássuk, az új gain értékekkel mennyire érjük el az ideális beállítást. Az ellenőrzési adatokat nem részletezem, csak arra térnék ki, hogy a meredekség kalibráció után az „A” oldalon 1.98 Nm/V, míg a „B” oldalon 1.964 Nm/V. Ennél a kézi mérésnél minden adat leolvasás kétes pontosságú, mert szimultán kell a multiméter értéket leolvasni, miközben a mérőkormányt az ember egy állásban tartja, és a másik kezével egy gyors ’print screen’-t nyom a számítógépen, hogy meglegyenek az Evalkit által mutatott értékek, ezért túl nagy a tévedés és pontatlanság a mérés alatt. Ennél a projektnél a kalibráció 4 szenzor esetén 53 percig tartott. Ebben nincs benne a mérési összeállítás előkészítése, csak tisztán a kalibrálás ideje.

5.3.2 Automatizált kalibráció

A kézi kalibráció után futtattam egy automatizált kalibrációt, ahol a mérés időtartalma, valamint a Delta gain és offset értékek kiíródnak. A delta értékek arra utalnak, hogy mekkora „igazítás” kell, a gain tekintetében (itt maga a koefficiens szorzó íródik ki), illetve offsetnél ez Nm-ben adódik.

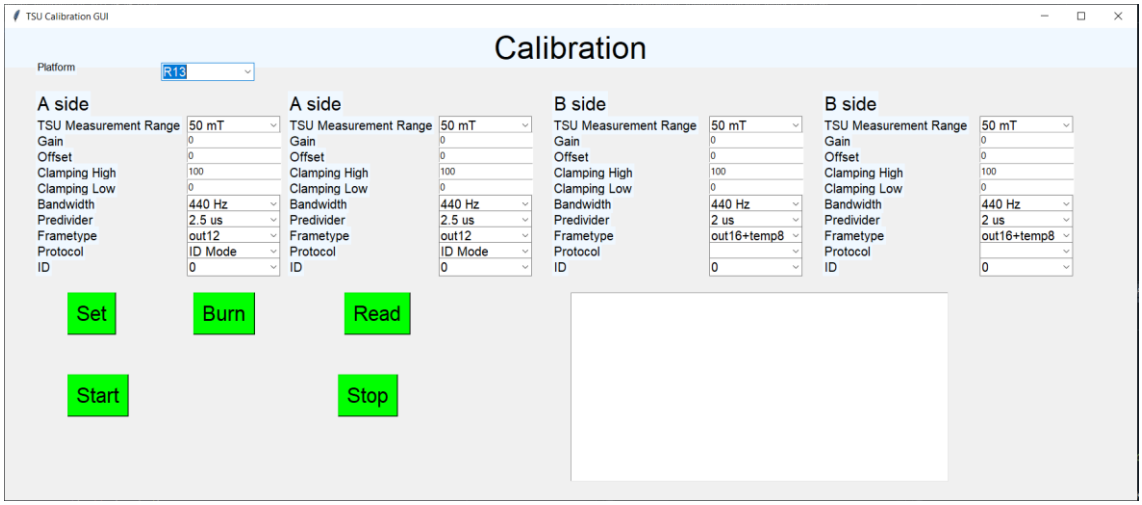
Az automatizált mérés során mentésre kerülnek a nyomatékszenzor jelei, a referencia jel, valamint a későbbiekben a számolás pontosításának érdekében, a jelen beállítás mátrixai is. A lementett adatok feldolgozására van egy Evaluation script, ami a .mat kiterjesztésű mérési eredményeket ábrázolja illetve javítja ahol kell. Habár az alapprogram megcsinálja, amit az Evaluation script, ez csak utólagos ellenőrzésre szolgál, illetve használható arra, hogy az eredeti értékeket és az eredeti nyomatékhibát is kiolvassuk, ami miatt a kalibrációra küldték a nyomaték szenzort. Az adatok javítása alatt maszkolást értünk. Az adatok maszkolásra kerülnek, hogy a referencia értéknek csak a [-8, +8] Nm-es tartományát vizsgáljuk. Ennek oka az úgynevezett S-curve hiba. A gyűrűmágnes mágneses tere szinusz alakú, aminek lineáris tartományában mérjük a mágneses teret, ez a lineáris szakasz a torziós szög ±4°-os tartományán található. A torziós rúd merevsége [Nm/°] befolyásolja, hogy mennyire maradunk a lineáris tartományban úgy, hogy egy merevebb torziós rúd kisebb szögtartományon éri el a kívánt Nm-es tartományt. Projektenként változó, hogy a torziós rúd merevség mekkora, ezért az adatokból, mindig kivesszük a szélsőértékektől ±2 Nm-re levő értékeket. A maszkolás lehetővé tette, hogy a különböző mérési tartománnyal rendelkező projektek hasonló pontosságú kalibrációt eredményezzenek. Mivel a számolás parametrizálva van, amely paraméter a platform kiválasztásánál állítódik, a kisebb nyomatéktartományú projektekre is alkalmazható a számolás.

A maszkolás nem csak Nm tartományban az mérőkormánytól jött jelre történt meg, hanem a sikertelen kiolvasások is. A digitális protokollok nagy előnye az ellenőrizhetőség, erre szolgál a CRC Checksum nibble, minden üzenet végén. Ezzel kivédhetők a hibás üzenetek, amelyek a kábel hosszúsága vagy zavaró tér miatt

```
TLE_CRCTable = [0, 13, 7, 10, 14, 3, 9, 4, 1, 12, 6, 11, 15, 2, 8, 5]
def TLE4998_crc(message, lendata=4):
        """
        check TLE4998 CRC for message
        message=[status,data1,data2,data3,crc] nibble value
        """
        crc = 5;
        for i in range(lendata):
                crc = crc ^ ((message>>((4-i)*4)) & 0xF)
                crc = TLE_CRCTable[crc]
        return crc == (message & 0xF)
```

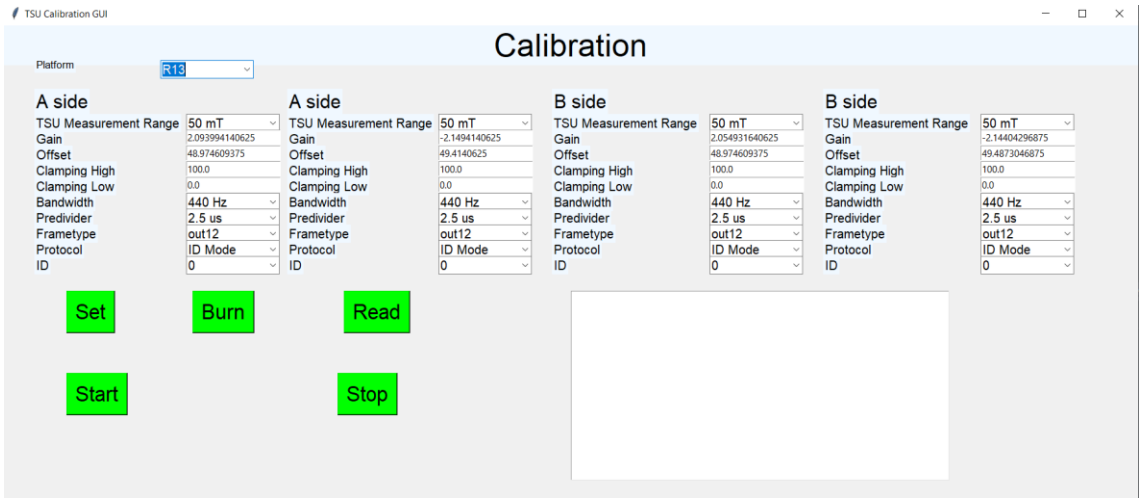
bekövetkeztek. Így a hibás üzenetek az esetleges kiugró nyomaték értékeikkel nem befolyásolják a mérés eredményét. Ennek számolása a következő lentebb olvasható.

A felhasználói felület kiolvasás előtti állapotát a 38. ábra mutatja. „A” megszokottól eltérő „B” oldali értékeket állítottam be a következőkre: Predivider, Frametype, Protocol, ID, hogy látszódjon, a kiolvasás tényleg sikeresen végbemegy. Az „A” oldali szenzorok ezzel szemben felvették a platform kiválasztása után ismert alapértelmezett értékeket. A platform kiválasztásra azért van szükség, hogy ismert legyen a csatornaszám.



38. ábra: Kiolvasás előtti állapot

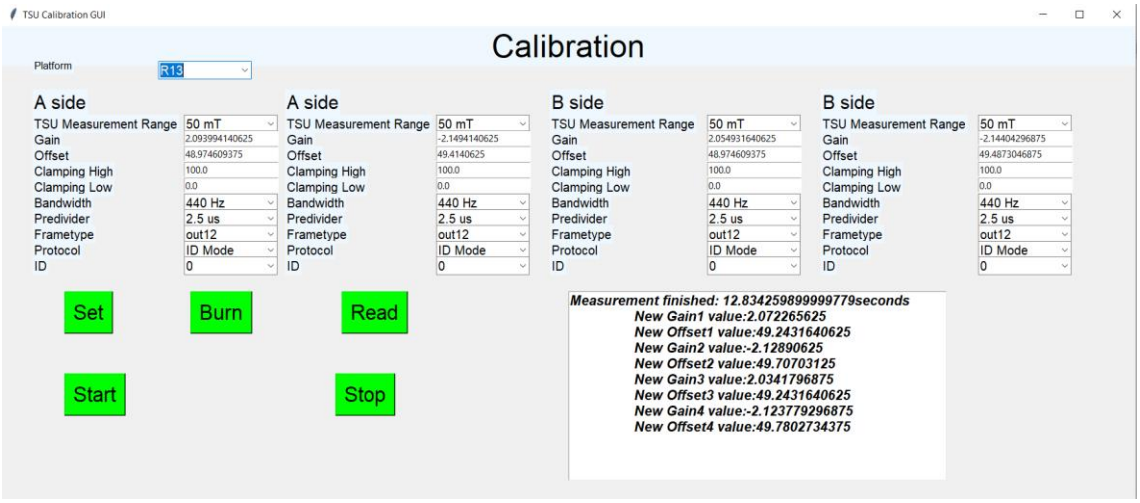
Ezután a Read gombra nyomva hallatszik a relék kapcsolása, egyik, másik irányba, majd vissza az STM board irányába, a 39. ábra által mutatott felületet látjuk.



39. ábra: Kiolvasásból ismert értékek ábrázolása

A sikeres kiolvasás után egy mérést indítottam, ahol a kormány óra járásával ellentétes (CCW-counter clockwide) irányba forgattam egészen a szélső értékig, majd

vissza a másik oldali szélső értékig. Ezt kétszer ismételtem, hogy több mérési pontunk legyen, majd középállásban megállítottam a mérést a Stop gomb megnyomásával. Ezután a felületen végbemenő változást a 40. ábra mutatja. Kiírásra került a mérés hossza, valamint az újonnan kiszámolt gain és offset értékek.



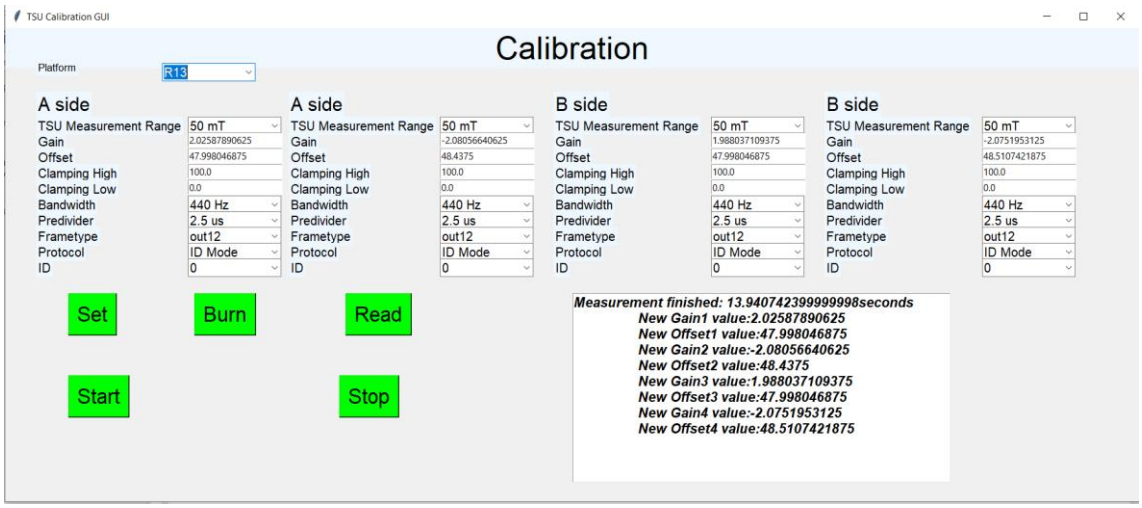
40. ábra: Mérés utáni felhasználói felület

A Python konzoljába kiírásra került a delta gain és offset érték, ami a kiolvasott és a beírni kívánt értékek közötti különbség eredménye. Innen tudjuk, hogy a kézi kalibráció mennyiben tért el az automatizált mérés eredményétől.

	ΔGain	**ΔOffset [Nm]**
1. csatorna	0.94730675	-0.03648351
2. csatorna	0.94903651	-0.00504711
3. csatorna	0.94250526	0.01845775
4. csatorna	0.93940874	0.03132717

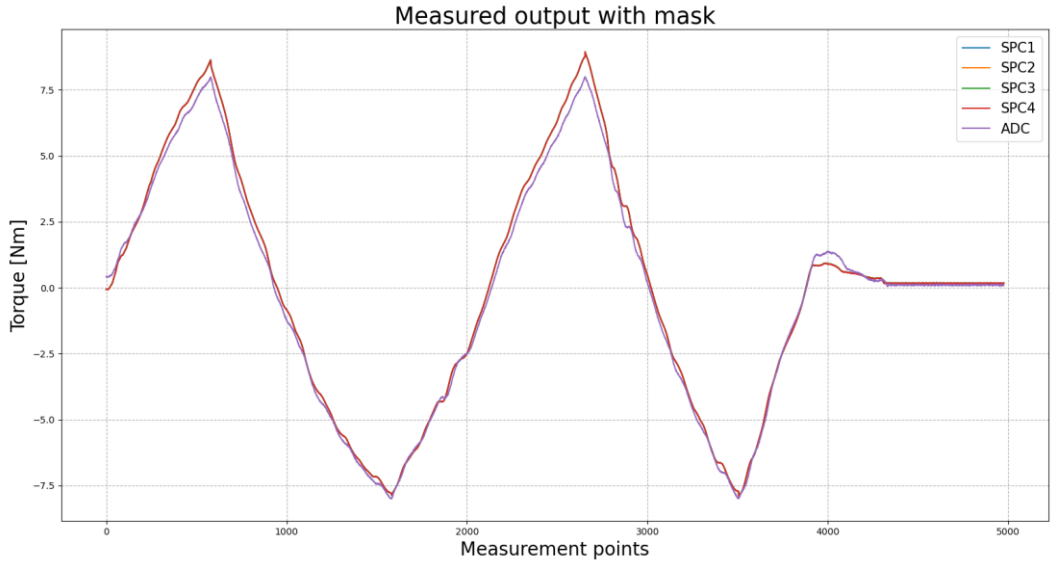
2. Táblázat: A kézi kalibráció pontossága az automatizált alapján

Az új gain és offset érték kiíródik a felületre, illetve a teljes mérési fájlt lementjük. Kevésbé tartottam fontosnak a delta értékek kiírását, mint az új számolt értékeket, mert ettől függ, hogy a felhasználó beírja az új értéket a szenzor EEPROM-jába vagy nem. A Burn függvény helyes működésének teszteléseképpen a gomb megnyomása után ismét kiolvastam a szenzor értékeket, ezt a 41. ábra mutatja, ahol látszik, hogy a kiolvasott értékek megegyeznek a beírtakkal.



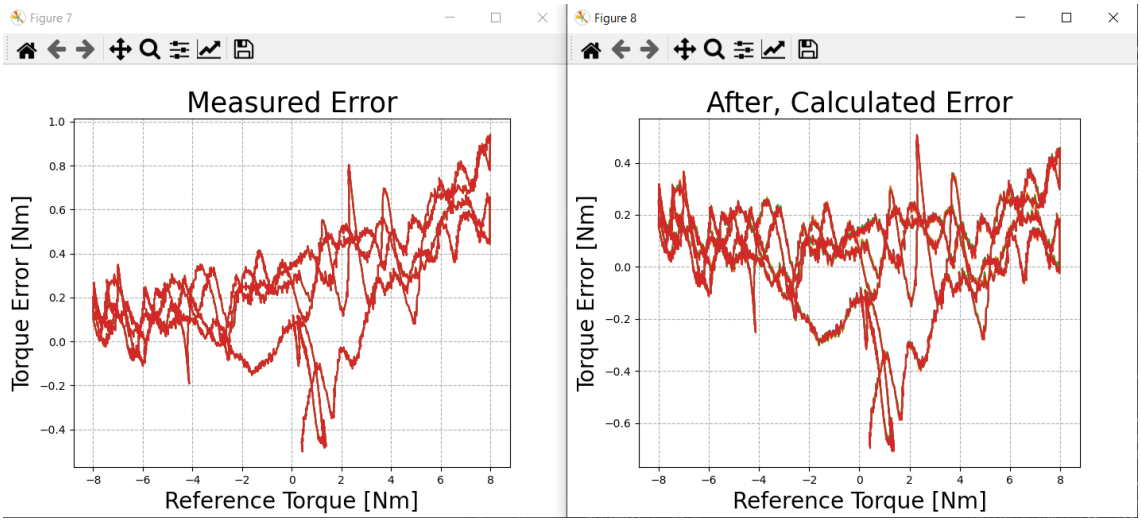
41. ábra: Mérésből kapott gain offset értékek a szenzorban

A kalibráció gain és offset értékeinek számolását korábban már taglaltam, azonban a függvény amit használunk a numpy könyvtáron belül található polyfit függvény. A függvény egy adott rendű, esetünkben egy elsőrendű, polinomot illeszt az adatok sokaságára, és ebből számolva adja vissza annak meredekségét és a metszéspontját. A függvényt azonban befolyásolja, hogy a mennyi adatpontunk van a hiszterézis miatt CW és CCW nyomaték képzésből. Amennyiben az egyik oldalról túl sok adatunk van, az elmozdíthatja a számolt offset értéket az egyik irányba. A kalibráció során, ebből kifolyólag, a 42. ábra alapján alakul a kormányforgatás. Itt már a maszkolt jelek láthatók, az SPC jelek a piros vonalként lapolódnak egymásra, míg a mérőkormányból vett jel a lila vonalként jelenik meg, mindkét vonal már Nm-re váltva.



42. ábra: Automatizált kalibráció adatpontjai maszkolás után.

A fentebb látható forgatásnak megfelelő hibás értékeket ábrázolva, ami nem más, mint a mért SPC jel és a referencia ADC közötti különbség, a 42. ábra által mutatott kalibráció és előtti és utáni hibát kapjuk.



43. ábra: Mérési hiba kalibráció előtti és utáni értékei

A bal oldalon látható különbség dőléséből látszik, hogy gain hiba terheli a mérést, azonban a korrigált értékekkel láthatóan ez kiküszöbölődik. A jobb oldalon látszik, hogy minták átlagértéke is módosult, a 0 Nm felé, vagyis az offset kalibráció is megfigyelhető az ábrán.

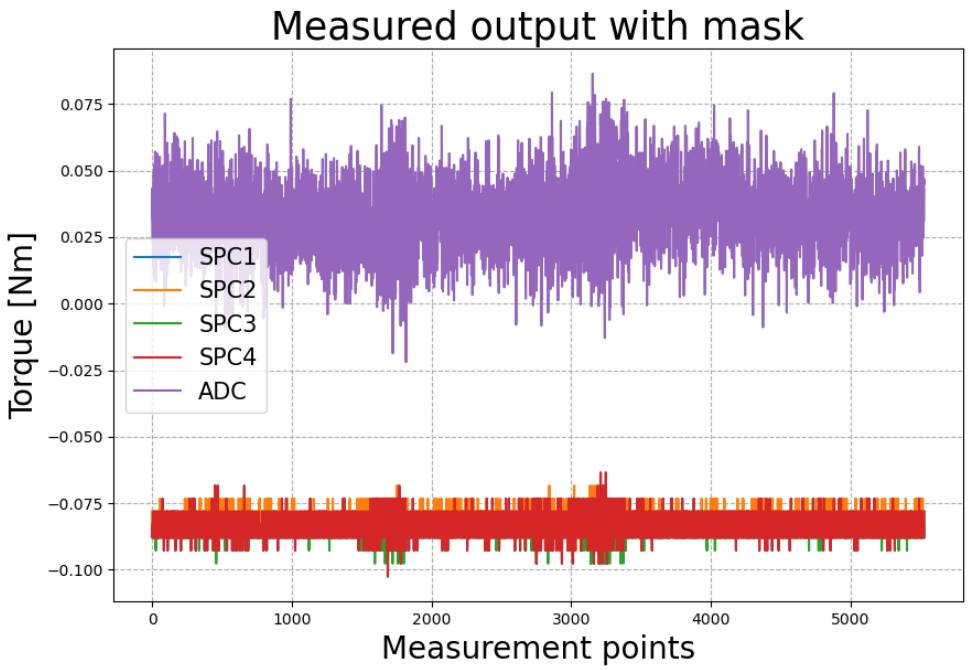
Az újra számolt hibaérték az automatizált kalibráció után offset esetén a [-0.002, 0.00016] Nm-es tartományban volt a csatornák hibája. A gain javított hibáját nehezebb mérni a hiszterézis miatt, de a két ábra peak-to-peak értékét összehasonlítva is sikerült 0.5 Nm-t javítani, ami jelentős.

5.3.3 Forgatás nélküli mérés

A forgatás nélküli mérés folyamata ugyanazokat a lépéseket igényli, mint a fentebbi fejezet, azonban a mérőkormány forgatása elmarad, ezáltal a gain érték nem lesz kiszámolva, hibás adatot kapunk. Az offset beállítása a gain függése miatt szintén pontatlan lesz, függetlenül attól, hogy 0 Nm-es nyomaték kimenetét mérjük.

A 44. ábra mutatja a mért jeleket. Látszik, hogy az STM board ADC bemenetére érkező jel nagyon zajos, illetve nem is teljesen 0 Nm értéket vesz fel. Az utóbbi oka, hogy bizonyos, mechanikából fakadó nyomaték a rendszerben marad, ezt a mechanikai feszültségek lazításával tudjuk kivenni a mérőkormányból. A zajosság nagyobb

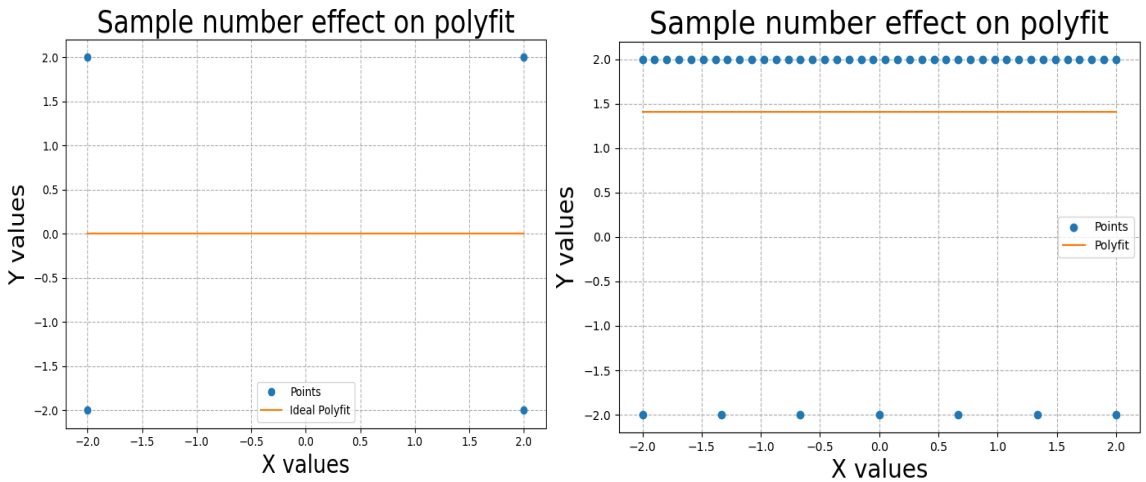
tartományon nem volt szembetűnő, így nem foglalkoztam vele, azonban ezt a problémát a referencia-jel mozgóátlagolása megoldaná.



44. ábra: Kormány tekerés nélküli mérés

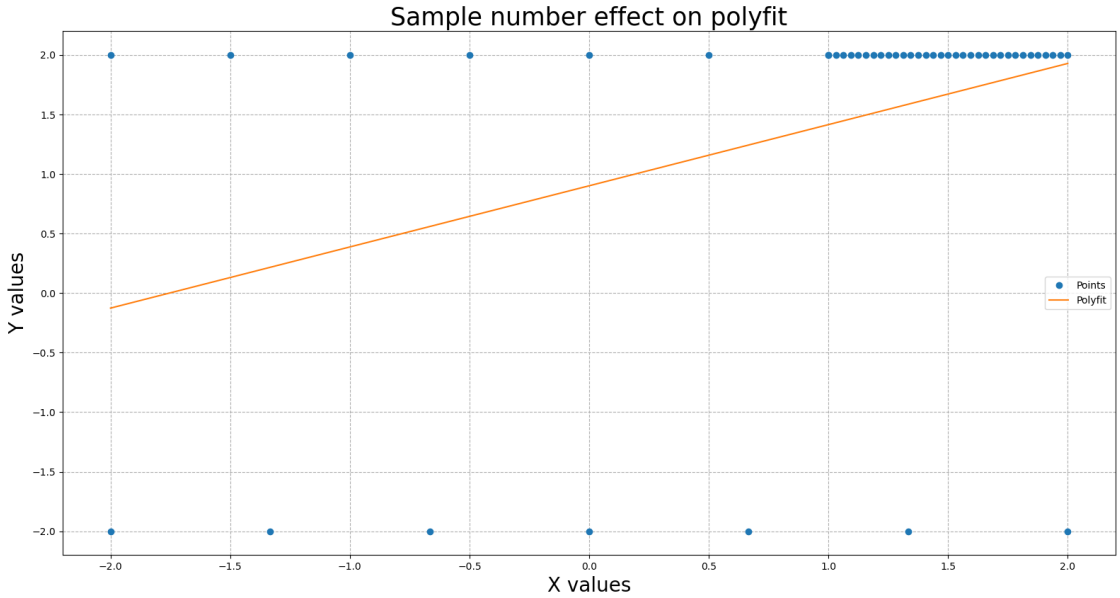
5.3.4 Aszimmetrikus forgatások eredményei

A kalibrációs számoláshoz használt polyfit eredménye, ahogy azt az Automatizált kalibráció fejezetben is említettem, nagyban függ a minták számától CW és CCW nyomaték kiadási irányok között. Mielőtt rátérnék az ezzel kapcsolatos méréseimre, ábrákkal illusztrálom, miképpen függ a mintaszámtól, illetve azok helyzetétől a polyfit függvény kimenete. Ennek oka, hogy a polyfit a legkisebb négyzetek módszerével illeszt.



45. ábra: Polyfit eredménye, kiegyenlített és kiegyenlítetlen mintaszám esetén

A 45. ábra bal oldala mutatja, azt az ideális helyzetet, mikor mind a négy síknegyedben egy darab mintánk van, mindkét tengelytől ugyanakkora távolságra. Az ábra jobb oldali része ezzel szemben azt mutatja, hogy ha y =2 vonal mentén megnöveljük a mintaszámot, az y=-2 vonalon lévőkhöz képest, akkor a polyfit kimenete a nagyobb mintaszám felé tolódik el. Ennek oka, hogy függvény a hiba négyzetgyökét minimalizálja a végeredményben. A jobb oldali ábrához hasonló kimenetet akkor tapasztalhatunk, ha a mérőkormány egyik irányú tekerése sokkal lassabban megy végbe, mint a másik irányba.



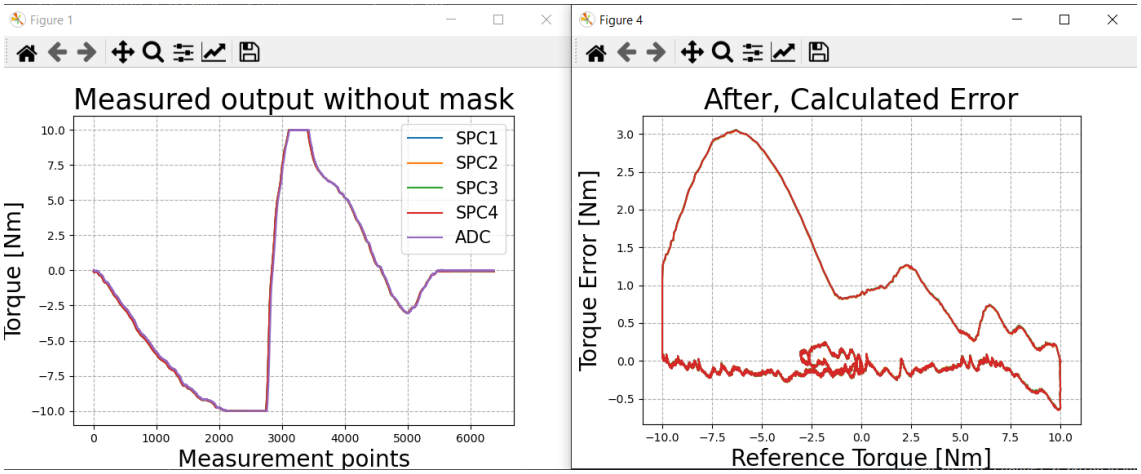
46. ábra: Polyfit eredménye, egy tartományon több mintaszámmal

A 46. ábra a polyfit eredményét ábrázolja, mikor az y=2, illetve x = (1,2) tartományon megnöveltem a mintaszámot, ekkor a kék színnel a minták, narancssárgával a polifyt illesztett görbéje látható. Látszik, hogy a felső mintatartomány sokasága miatt a polyfit eltolódik felfelé, azonban az y=2 vonal x=[1, 2] tartományban a korábbiaknál is több minta van. Ez a különbség egy meredekség változást eredményez a polyfit kimenetén. Ez a mérés során a mérőkormány szélsőértékben tartásával egyezik meg.

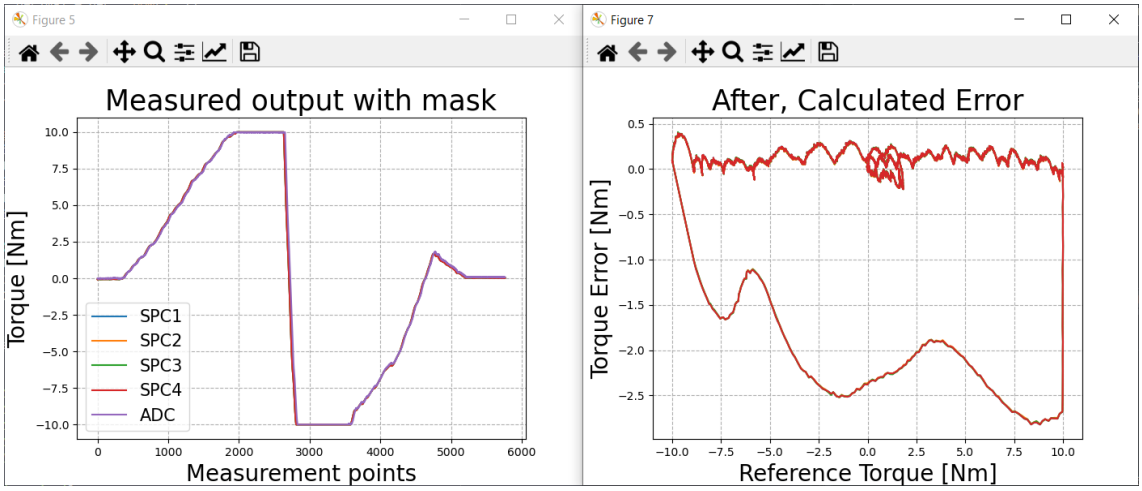
Az itt látható jelenségeket a következő mérésekkel próbáltam a mérési eredményeken láthatóvá tenni:

a. CW (clockwise) tartomány mintaszámának növelése
b. CCW (counter clockwise) tartomány mintaszámának növelése
c. 0 Nm-ben tartás hosszan, majd a standard kalibrációs kormány útvonal

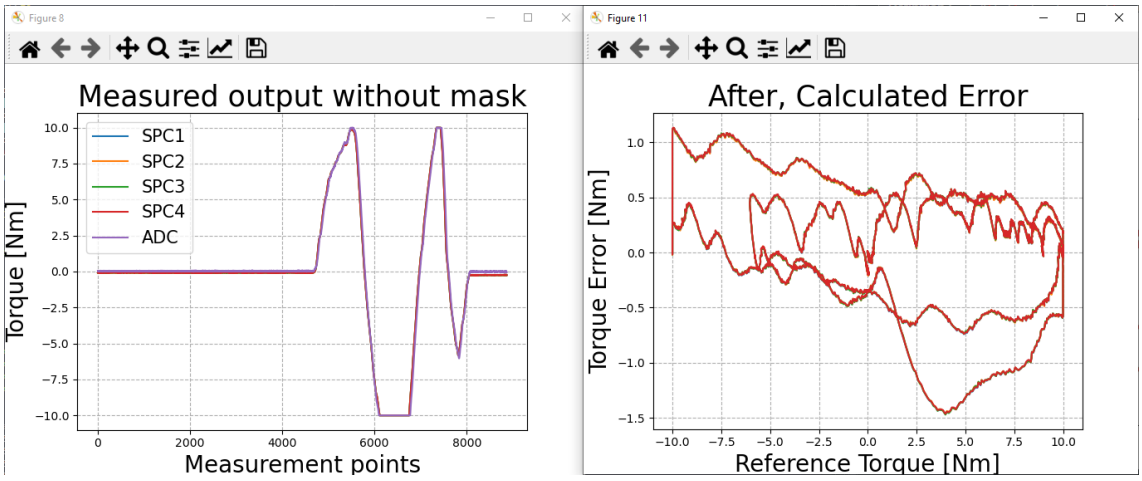
Érdekességképpen olyan méréseket is végeztem, amik előfordulása nem valószínű, viszont nem várt működési mód felderítésére alkalmasak lehetnek. Ilyen például a mérés folyamán a kormányra ütés egyik és másik irányból, valamint a kormányrúd rögzítése nélküli automatizált kalibrációja. Ez utóbbinak létjogosultsága a valós kalibrációk alatt nincs, mert a kormányrúd forgása miatt, alig adódik nyomaték a mérőkormányból a szenzorra, mert a mechanika leköveti a mozgás.



47. ábra: Megnövelt CW oldali mintaszám, és polyfitelt eredménye



48. ábra: Megnövelt CCW oldali mintaszám, és polyfit eredménye



49. ábra: 0 Nm-hez tartozó mintaszám növelésével elért eredmény

Mind a három ábrapár mutatja a várt tendenciát. A 47. ábra, illetve a 48. ábra jobb oldali ábráján a várt eltolódás jelenik meg. A 49. ábra jobb oldalán látható, hogy a kezdeti 0Nm-hez tartozó mintahalmaz annyira nagy számú (ezáltal nagyobb súllyal rendelkezik), hogy a polyfit funkció az illesztés jelentősen figyelembe vette.

A 47. ábra és a 48. ábra által mutatott hiszterézis nagysága, nem konzisztens a korábban említett 0.3 Nm-es hiszteréssel. Ennek oka, a referencia és a kiolvasott nyomaték közötti 32 ms-os késleltetés, ami az erősítő előtti RC nagysága miatt jelenik meg. Gyorsabb tekerésnél a késleltetés miatt nagyobb hibát mérünk a két jel között. LTSpice szimulációt végeztem a szűrőn, ahol egy impulzus jelre adott választ vizsgáltam, ez a Függelékben a 54. ábra. Ez az ellenállás cseréjével (100kΩ helyett 1kΩ-ra) megszűnik.

A polyfit által tapasztalt torzítás kiküszöbölhető, ha a nincs nagy szünet a mérőkormány forgatása és a mérés kezdete vagy vége között (túlzottan növelve egy adott ponton a mintaszámokat). Ugyanez elérhető, ha ugyanakkora sebességgel tekerünk mindkét nyomatékirányba. A tekerés minőségének hatása szoftveresen is kiküszöbölhető, erre megoldás, ha a referencia nyomatékot felosztjuk egyenlő távolságú pontokra, majd külön kis ablakonként átlagoljuk a CW és CCW nyomatékoldali mintákat a referencia nyomatékhoz. Ezután mérésből generált egyenletes eloszlású adatokra kell futtatnunk a polyfit függvényt.

6 Összegzés és további javaslat

A nyomatékszenzorok kalibrációja a pontosabb működés miatt elengedhetetlen. A kézi kalibráció korábban kicsit kevesebb, mint egy órát vett igénybe, olyan szakembernek, aki már elvégzett hasonló mérést. A diplomám keretein belül egy olyan eszközrendszer létrehozása volt a célom, ahol a folyamat egyszerűen elvégezhető és nem igényel szakembert. Ez az eszköz emellett rövidebb mérési időt és pontosabb kalibrációt is lehetővé tesz. Ennek megvalósítására egy mikrokontrollert használtam, amihez plusz nyomtatott áramkört kellett tervezni az egyes jelek csatlakozásához.

Az év során megtörtént a részegységek hardveres tervezése és mérése, valamint ezek összeállítása egy nyomtatott áramkörön. Ezután a mikrokontrolleres programozással ismerkedtem, majd írtam, illetve módosítottam kódot a kiolvasás, kommunikáció és az analóg-digitális átalakítás megvalósítására az Atollic programban. Ezen programok megírásához az SPC és az UART protokollok ismeretére volt szükség. Részleteiben megismerkedtem a használt nyomatékszenzor EEPROM tartalmával, majd Python kódot írtam annak kiolvasására és átírására. Ezután a felhasználói felületet alkottam meg a Python tkinter könyvtárával. A mérés és az ablak párhuzamos megjelenítéséhez betekintettem a több szálon futó programok működésébe, majd alkalmaztam a saját programomban. Amikor az írás olvasás a szenzor és a számítógép között megvalósult, rátértem a kalibrációhoz szükséges gain és offset értékek számolására. Ehhez segítségül szolgált a szenzor felhasználói kézikönyve. Amint a hardveres és szoftveres működés megvalósult, a verifikációhoz mérési összeállítást készítettem. Ehhez egy kalibrálatlan szervo unitot használtam egy mérőpadon.

Az automatizált kalibráció a kézi kalibrációhoz képest nagyobb pontossággal és rövidebb idő alatt végbement. A felhasználói felület eleget tesz a kívánt funkcióknak. A tervezett eszköz megfelel a kívántaknak.

A további fejlesztési lehetőségként javaslom a mikrokontrollertől kapott referencia jel átlagolását, a referencia jel előtti szűrő módosítását, valamint a mintaszám szoftveres kiegyenlítését a negatív és pozitív nyomatéktartományban. Így erősebb megkötések nélküli mérőkormány forgatásból is lehetséges pontos kalibráció. Emellett a kalibrációs eszköz bővítése szükségszerű más és új nyomaték szenzorok kalibrációjához szükséges kompatibilitáshoz, több projekt lefedésének érdekében.

Irodalomjegyzék

[1] Dr. Lévai Zoltán, *Szervokormányok – kormányszervók*
http://lezo.hu/szerkezettan/futomuvek/kormany/szervok/szervo.html
Letöltve: 2019.10.01

[2] Thyssenkrupp Components Technology Hungary Kft.
https://www.thyssenkrupp.hu/hu/telephelyek/budapest
Olvasva: 2019.10.01.

[3] Thyssenkrupp Components Technology Hungary Kft.
Hermann Bressner: EPAS Academy – Introduction video

[4] Thyssenkrupp Components Technology Hungary Kft.
Philippe Steck (2015): Steering Gear (Rack-EPS)

[5] Michael Fernie: *How Electric Power Assisted Steering Works, and Why it's better Than Hydraulic*
https://www.carthrottle.com/post/electronic-power-assisted-steering-how-does-it-work/
Letöltve: 2019.10.16

[6] Govindan Unni1, Deepak S: *SPST to DPDT Switching Conversion Module for Solid State Relays (SSR)*
https://www.ijireeice.com/upload/2017/march-17/IJIREEICE%2021.pdf
Letöltve:2021.10.25

[7] Littelfuse: *What do SPST, SPDT, DPST, and DPDT mean?*
https://www.littelfuse.com/technical-resources/technical-centers/commercial-vehicle-technical-center/poles-and-throws.aspx
Letöltve: 2021.10.26

[8] Digikey: Switch Basics
https://www.digikey.hu/hu/ptm/n/nkk-switches/switch-basics/tutorial
Letöltve: 2021.10.18

[9] Josef Kramolis: *SENT/SPC Driver for the MPC5510 Micricintriller Family*
Letöltve: 2021.11.15

[10] Tim White: *A Tutorial for the Digital SENT Interface*
https://www.idt.com/document/whp/tutorial-digital-sent-interface-zssc416xzssc417x
Letöltve: 2019.10.20

[11] Teach Computer Science: *Simplex, Half Duplex, Full Duplex*
https://teachcomputerscience.com/simplex-half-duplex-full-duplex/
Letöltve: 2019.10.20

[12] Roland Gamper: *Analysis of a SENT (Single Edge Nibble Transmission) Signal using an oscilloscope,* 2015

http://www.lahniss.com/_u/_upublications/sentdecodev7.pdf
Letöltve: 2019. 11.02

[13] Allegro microsystems: *USING SENT COMMUNICATION OUTPUT PROTOCOL WITH A17700 PRESSURE INTERFACE SENSOR*
https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjq6r_rjqL2AhXPuaQKHU7EAB0QFnoECAgQAQ&url=https%3A%2F%2Fwww.allegromicro.com%2F-%2Fmedia%2Ffiles%2Fapplication-notes%2Fan296244-using-sent-output-protocol-with-a17700.pdf%3Fsc_lang%3Dzh-cn%26hash%3D38473ED18731D9139CED9E4190D0C336&usg=AOvVaw1SzPbHpFqthOZlKKWUby9r
Letöltve: 2022.02.28

[14] Thyssenkrupp Components Technology Hungary Kft.
Árpád Lipkovics: *Torque Sensor Unit Gain and Offset Correction*
Letöltve: 2022.01.30

[15] Infineon: *TLE4998 User's Manual, Rev. 1.3*
Leöltve: 2022.02.01

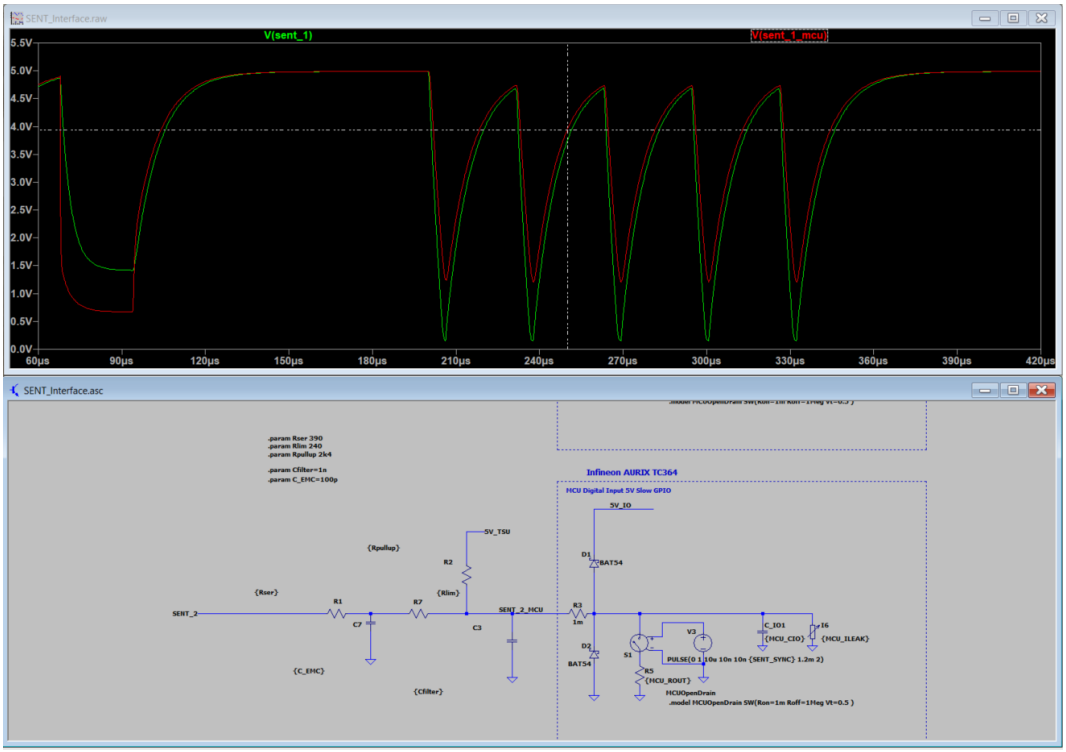
[16] ST: *STM32 Nucleo-144 development board with STM32H743ZI MCU, supports Arduino, ST Zio and morpho connectivity*
NUCLEO-H743ZI - STM32 Nucleo-144 development board with STM32H743ZI MCU, supports Arduino, ST Zio and morpho connectivity - STMicroelectronics
Letöltve: 2022.02.14

[17] Texas Instruments: *Atollic, Atollic Software Development Tools*
ATOLLIC IDE, configuration, compiler or debugger | TI.com
Letöltve: 2022.01.12

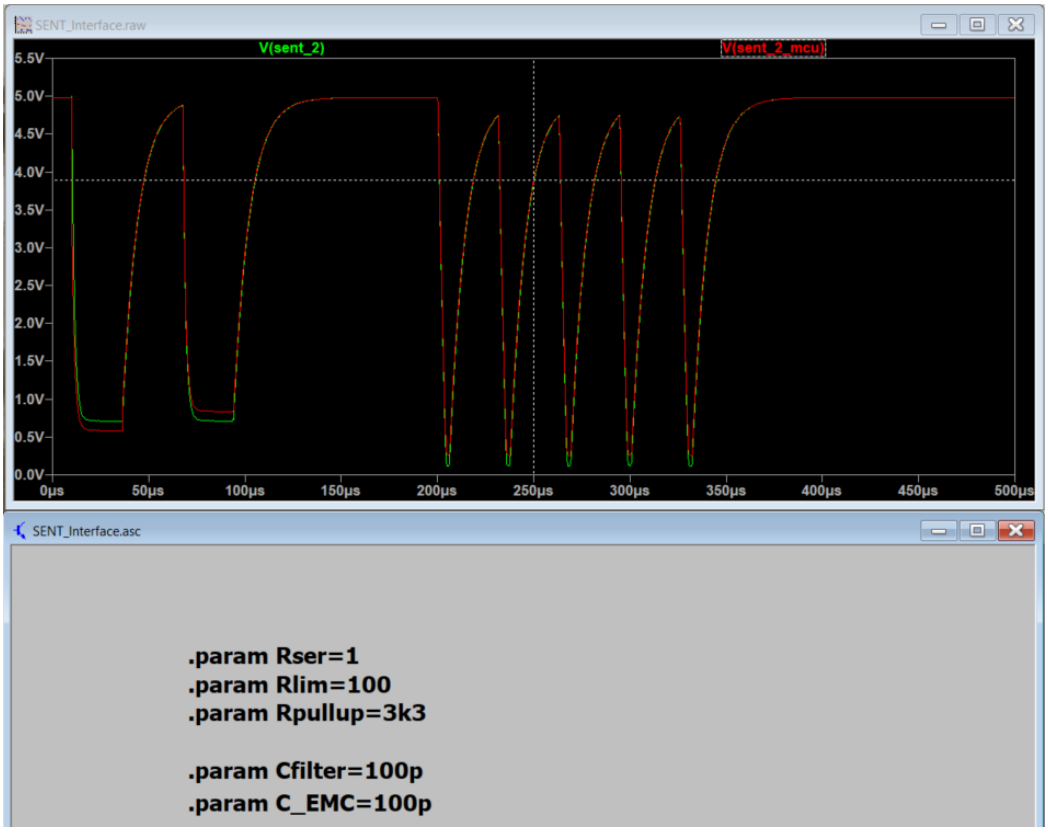
[18] ST: *TrueSTUDIO - A powerful eclipse-based C/C++ integrated development tool for your STM32 projects - STMicroelectronics*
TrueSTUDIO - A powerful eclipse-based C/C++ integrated development tool for your STM32 projects - STMicroelectronics
Letöltve: 2022.01.10

[19] Analog Devices: *LTspice Simulator*
LTspice Simulator | Analog Devices
Letöltve: 2021.12.01

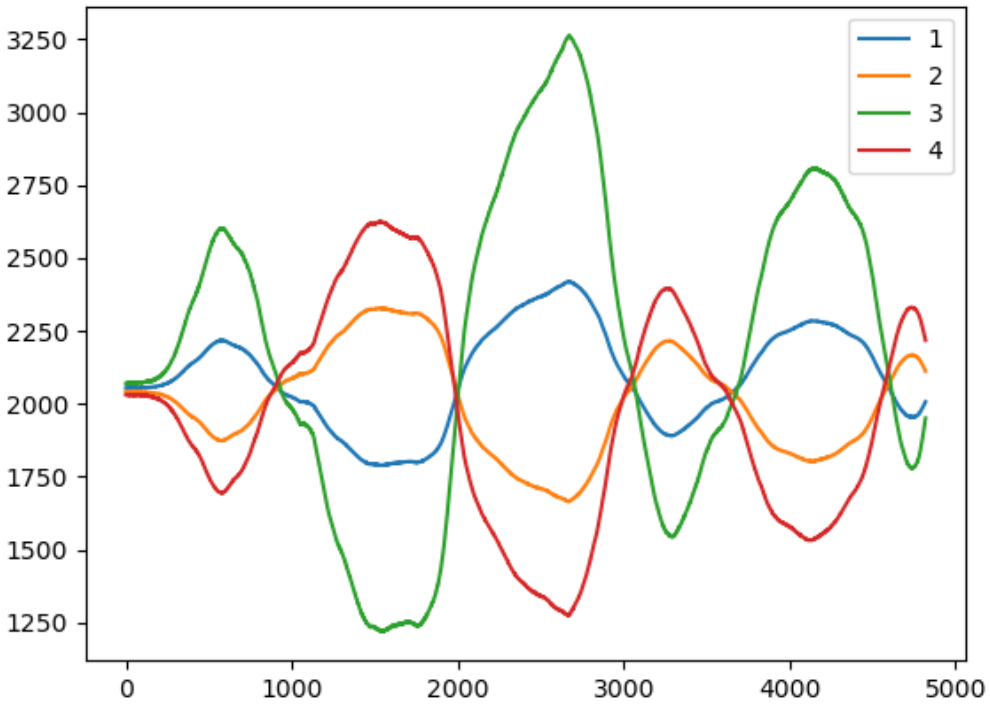
Függelék



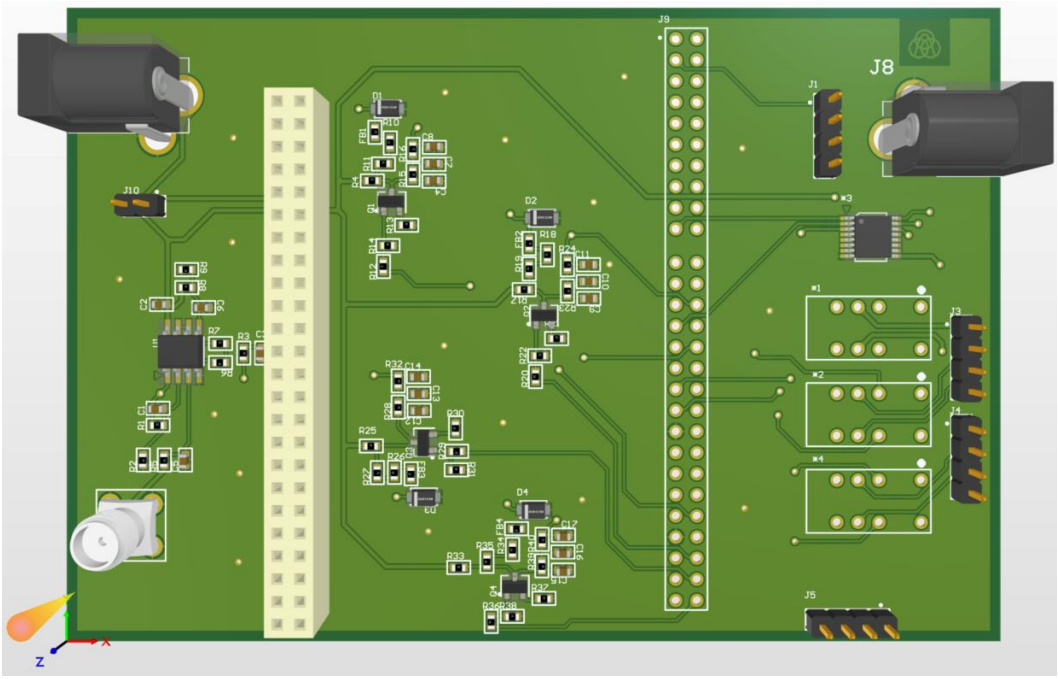
50. ábra: SENT/SPC interfész szimulációja meglévő értékekkel



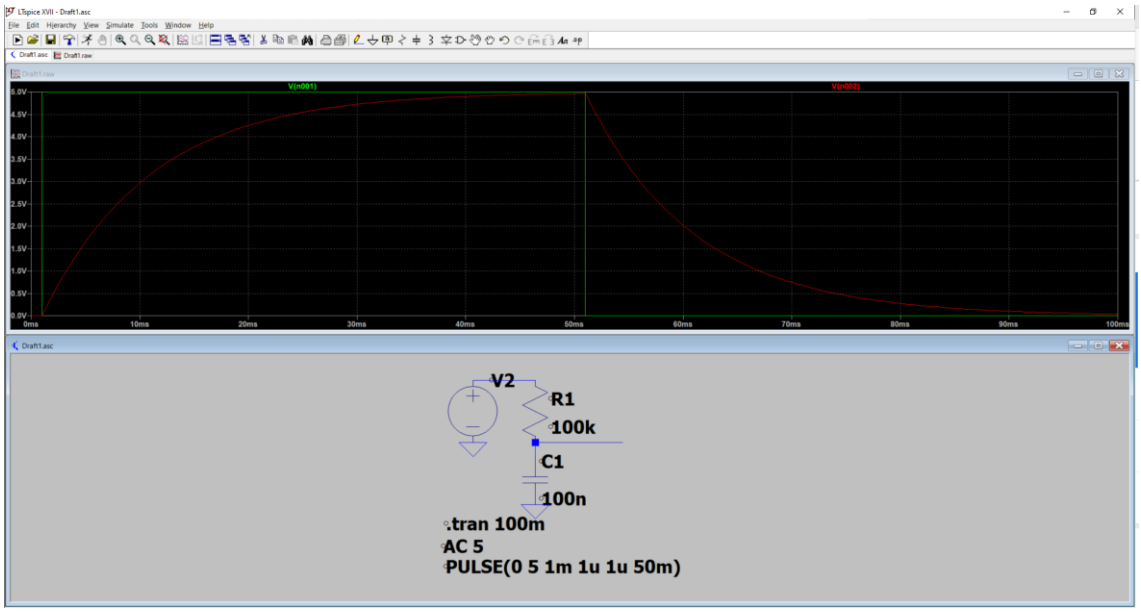
51. ábra: SENT/SPC interfész szimulációja módosított értékekkel



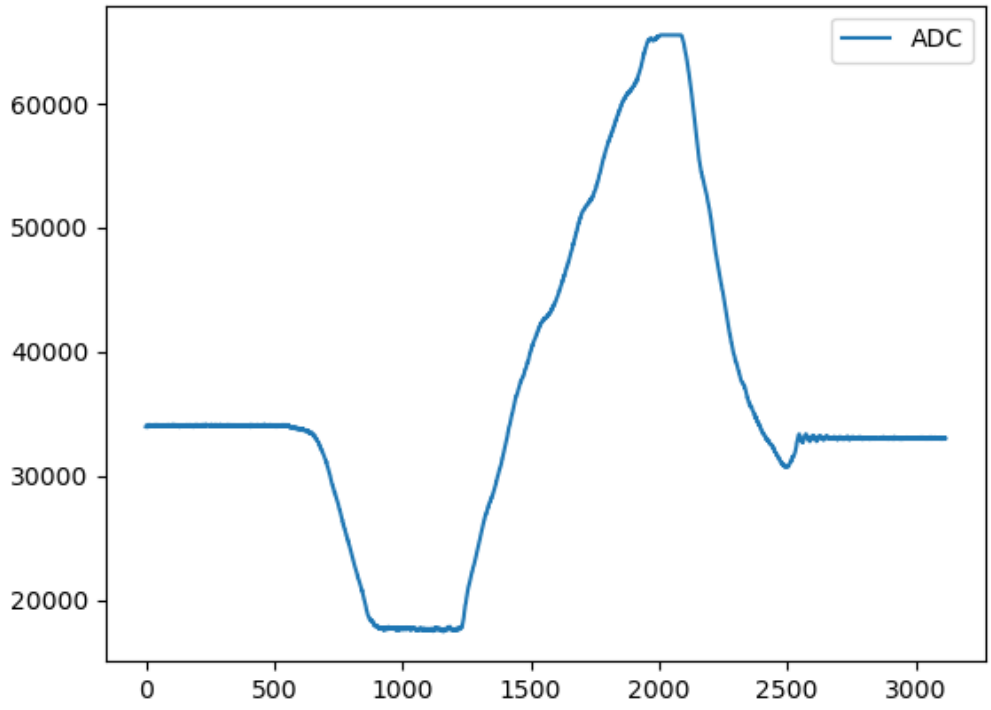
52. ábra: Módosított nyomatékszenzor kimenete



53. ábra: A tervezett PCB 3D-s árbája



54. ábra: Az RC szűrő LTSpice-os szimulációja



55. ábra: ADC korlátozott kimenete