GÁZTURBINÁS SUGÁRHAJTÓMŰ TÜZELŐANYAG-SZIVATTYÚJÁNAK FORDULATSZÁM SZABÁLYOZÁSA

készítette: Bánfi Bendegúz Balázs

konzulensek: Dr. Beneda Károly Tamás

Dr. Veress Árpád

2023.

Tartalomjegyzék

1 Bevezetés

Napjaink egyik fő kihívása a jelenleg tömegek számára elérhető közlekedési eszközök olyan módon történő átalakítása és üzemeltetetése, amely megfelel az egyre szigorodó környezet- és zajvédelmi előírásoknak. Legyen szó gépjárműről vagy repülőeszközről, számtalan olyan kutatás és fejlesztés van folyamatban, amely azt igazolja, hogy a mai módszerekkel bár a jövőben egyre kevésbé gazdaságosan, de tovább folytathatjuk az életmódunkat, az egyre nagyobb és nagyobb összegzett környezeti terhelést fog a számunkra jelenteni.

Számomra a jelenlegi kihívások mellett nagyon fontos a tudatosság és az ennek jegyében eszközölt fejlesztések, szeretnék előremutatóan gondolkodni és ezáltal segíteni egy élhetőbb jövőkép létrehozásában. Ehhez az egyik út a közlekedési eszközök megreformálásán keresztül vezet, noha nem kizárólag, hiszen belátjuk, hogy nem ez lesz az egyedüli mód erre. Kiindulásként jó, ha észben tartjuk, kitartóan, de apró lépésekben érdemes haladnunk.

1.1 Környezeti és gazdasági háttér

1.1.1 Kibocsájtás

Azt gondolhatnánk, hogy a globális szén-dioxid és károsanyag-kibocsájtás nagy szereplője a repülés, azonban ez korántsem igaz. Természetesen mindenki fejében él a gondolat, hogy óriási elégetett fosszilis energiahordozómennyiségekről beszélünk a repülés terén és ez részben igaz is, hiszen a hajtómű igen nagy teljesítményét csak relatíve nagy fogyasztás kárán tudjuk elérni. Vegyük példának a CFM LEAP-1A típusú hajtóművét, amelyet például az Airbus A320 neo típusú repülőgépcsaládon láthatunk. A hajtómű képes felszálláskor $143{,}05\ kN$ tolóerőt szolgáltatni, valamint egyenletes repülés közben a tolóerő specifikus fogyasztása $14{,}4\ \frac{g}{kN \cdot s}$ [1]. Ezt, ha egy átlagos, mondjuk a példa kedvéért $t_{flight} = 2\ h$ repülési idővel számoljuk ki, akkor az 1. egyenlet értelmében az $1\ kN$ tolóerőre eső fogyasztása a hajtóműnek majdnem $104\ kg$, amely, ha a gép rossz aerodinamikai tulajdonságokkal rendelkezik és nagyon sok tolóerőt igényel, nagy fogyasztást eredményez.

$$m_{fuel\ per\ kN_{flight}} = TSFC \cdot t_{flight} = 14{,}4 \frac{g}{kN \cdot s} \cdot 7200\ s = 103{,}68\ \frac{kg}{kN} \qquad \textit{1. egyenlet}$$

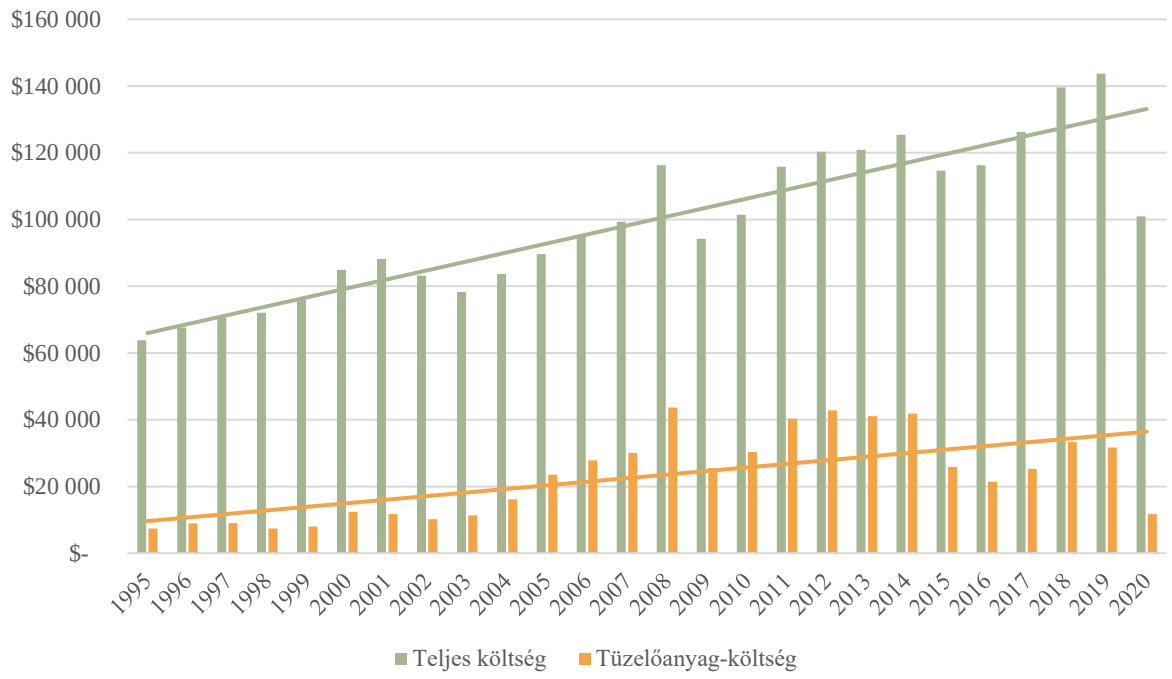
Mindezek ellenére a globális kibocsájtások okozója a legnagyobb részben nem is a repülőipar. Az ICAO (International Civil Aviation Organisation) által összeállított statisztika alapján valójában a légijárművek összességében csak a globális szén-dioxid kibocsájtások csupán 2%-át adják [2]. Ennek ellenére a légitársaságok az egy repülés árát csökkenteni szorgalmazzák, hiszen ez kevesebb kiadást eredményez számukra.

Adódik a kérdés, hogyan kapcsolódnak az alternatív tüzelőanyag-vizsgálatok a kibocsájtás csökkentéséhez? Az égéstér optimalizálásával és a tüzelőanyag égési, valamint szivattyúzási tulajdonságainak pontos, sugárhajtóműves próbapadon történő kimérésével és az utána történő, adott

anyagra indított optimalizációs eljárás után akár javítható a hajtómű fogyasztása is elviekben. Azonban nem pusztán a fogyasztás csökkentése érdekében érdemes ilyen vizsgálatot elvégezni, hanem majd a későbbiekben látni fogjuk, hogy más egyéb tényezők miatt is érdemes ilyen vizsgálatokba belefogni és ezekre megfelelő próbapadot és vezérlést tervezni.

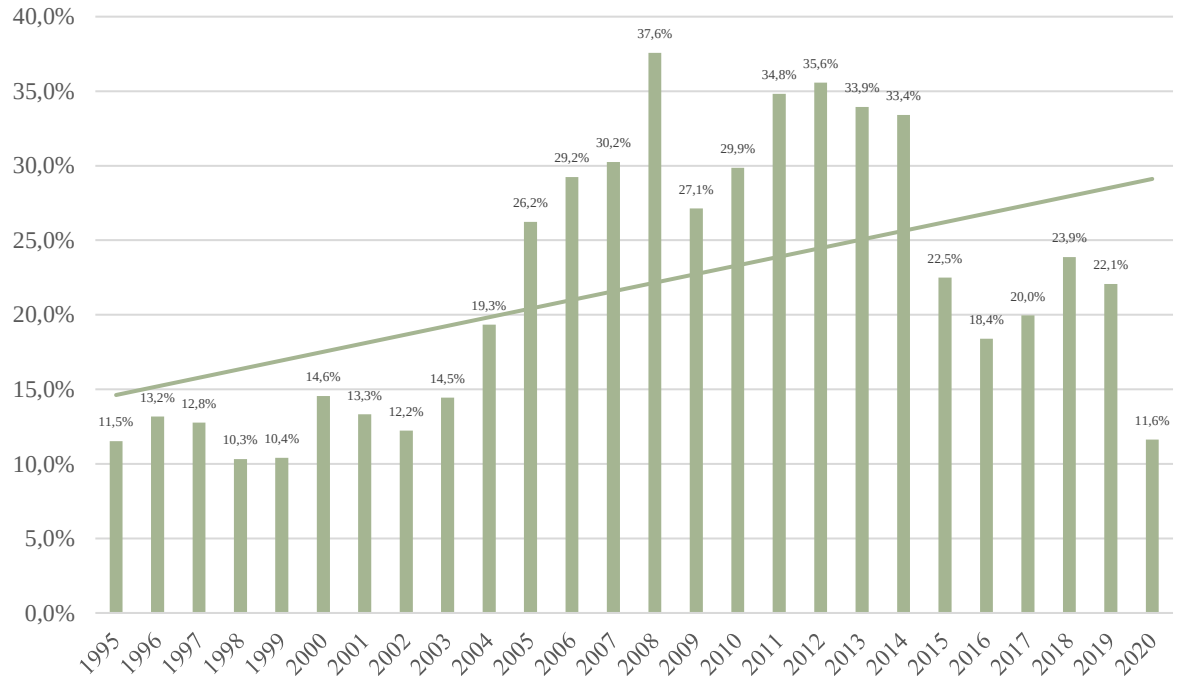
1.1.2 Az árak emelkedése

Elég, ha csak megvizsgáljuk a mai ipari trendeket a gázturbinás hajtóművek terén. Ha grafikonon összevetjük a légitársaságoknak a repülőgépek tüzelőanyaggal történő feltöltésére fordított költségeit, akkor évről-évre hatalmas számokat látunk. A Global Airline Data Project adatait használva [3] könnyen meghatározhatjuk az évről-évre légitársasági szinten a tüzelőanyagra fordított költségeket. Itt elérhető minden, légitársasági üzemeltetéssel kapcsolatos fiskális adat. Ezeket felhasználva készíthetünk egy olyan saját ábrát, amit az 1. ábra mutat. Itt egészen 1995-ig visszamenőleg találhatunk adatokat ezekkel kapcsolatban.



1. ábra Légitársaságok teljes, valamint csak tüzelőanyagra fordított költségei évekre lebontva 1995-től 2020-ig, millió USD-ben 2020-as árfolyamon, a trendeket is megmutatva [3].

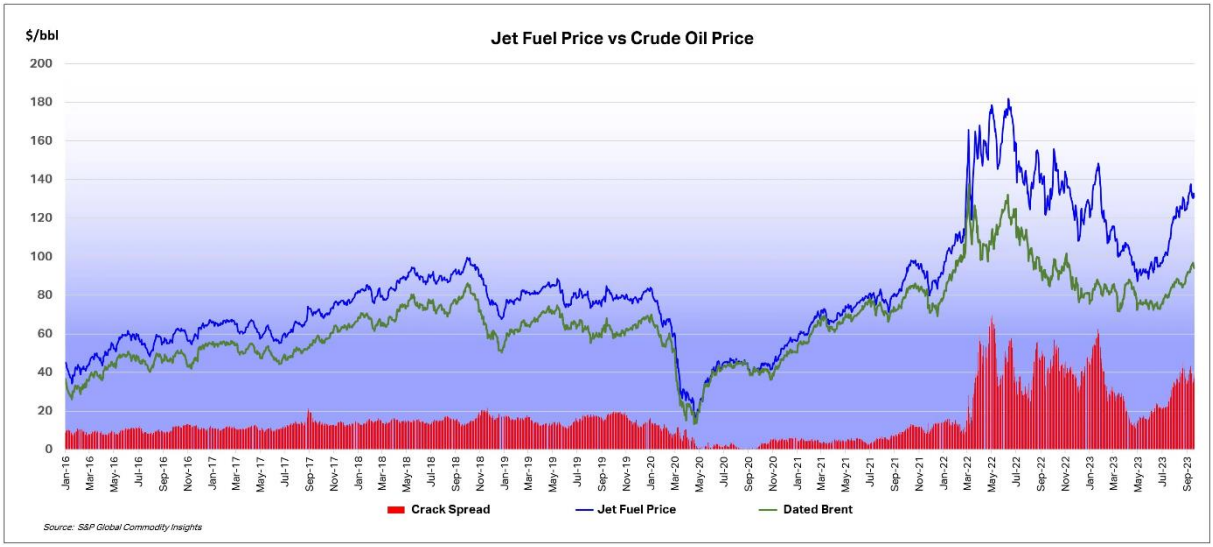
Ha vetünk egy pillantást az 1. ábra által mutatott grafikonra, akkor láthatjuk, hogy a 2020-as árfolyamra átszámolt légitársasági költségek emelkedtek az évek során, amely úgy hiszem nagyrészt az iparág növekedésével magyarázható. Ha a 2020-as adatokat vesszük figyelembe, akkor egyből szembe is tűnik a koronavírus-járvány okozta hatalmas visszaesés is a kereskedelmi utasszállító légitársaságok számain, hiszen egy földön veszteglő gép érdemleges tüzelőanyag-fogyasztás nélkül nem termel akkora költségeket, mint egy éppen utazó gép. Azonban a trendeket tüzetesebben vizsgálva láthatjuk, hogy a repülőgépekben alkalmazott kerozin ára miatti költségek is növekedtek. Itt már nagyon jól nyomon követhetőek évről-évre a nagyobb geopolitikai változások is a világgazdaságban, így ez csak hosszútávon értelmezhető, hiszen a rövid periódusú, nagy fluktuációkat okozó hatások is szerepelnek az ábrán. Ilyen például a 2020-as koronavírus járvány, amelyben, ha pusztán csak az éves költségeket nézzük, több, mint 60%-os visszaesést tapasztalunk. Említendő továbbá a 2008-as világgazdasági válság okozta gyors költségnövekedés, amelyek mind nagyon jól szemléltehetőek a diagramon.



2. ábra Légitársaságok tüzelőanyag-költségei a teljes költségeikhez képest évekre lebontva 1995-től 2020-ig a trendeket is megmutatva [3].

Ha ezt összehasonlítjuk a kizárólag tüzelőanyagra fordított költségekkel, akkor a 2. ábra diagramjához jutunk. Látható, hogy a fluktuációkat nem figyelembe véve itt is emelkedő trenddel állunk szemben, természetesen a 2020-as világjárvány okozta hatásokat nem figyelembe véve. Ez jelentheti azt, hogy az évek során egyre drágább és drágább beszerzési árral álltunk szemben a kerozin árakat figyelve, ha pedig megnézünk egy IATA (International Air Transport Association) által közzétett, olaj és JET A1 típusú kerozin árait tartalmazó összesítést, amelyet a 3. ábra mutat, akkor ez igazolódni látszik. Nagyon jól láthatóvá válik továbbá ugyanezen ábrán a 2022. februárjában történt világpolitikai változás is, amely a globális olajárak ugrásszerű növekedésével járt, ez pedig a kék színnel jelölt JET A1 típusú kerozin árát is természetesen „magával rántotta”.

A növekvő árra az egyik legkézenfekvőbb magyarázat a globális kitermelések és olajkészletek csökkenése, valamint természetesen az infláció, azonban az utóbbi jelen esetben nem mutatható meg, hiszen a diagram inflációval korrigált értékeket vetít a grafikonra, amelyek 2023-as árfolyamon mutatják be az évek változásait. Ezzel a diagrammal voltaképpen kiküszöbölhetjük az 1. és a 2. által mutatott adathiányosságot a 2020-as és a 2023-as évek között.



3. ábra A IATA statisztikája az elmúlt öt év olajárairól, illetve JET A1 típusú repülőgép-tüzelőanyag árairól. Az értékek USD (2023)/hordó-ban értendők, egy hordó pontosan 158,987 l [4].

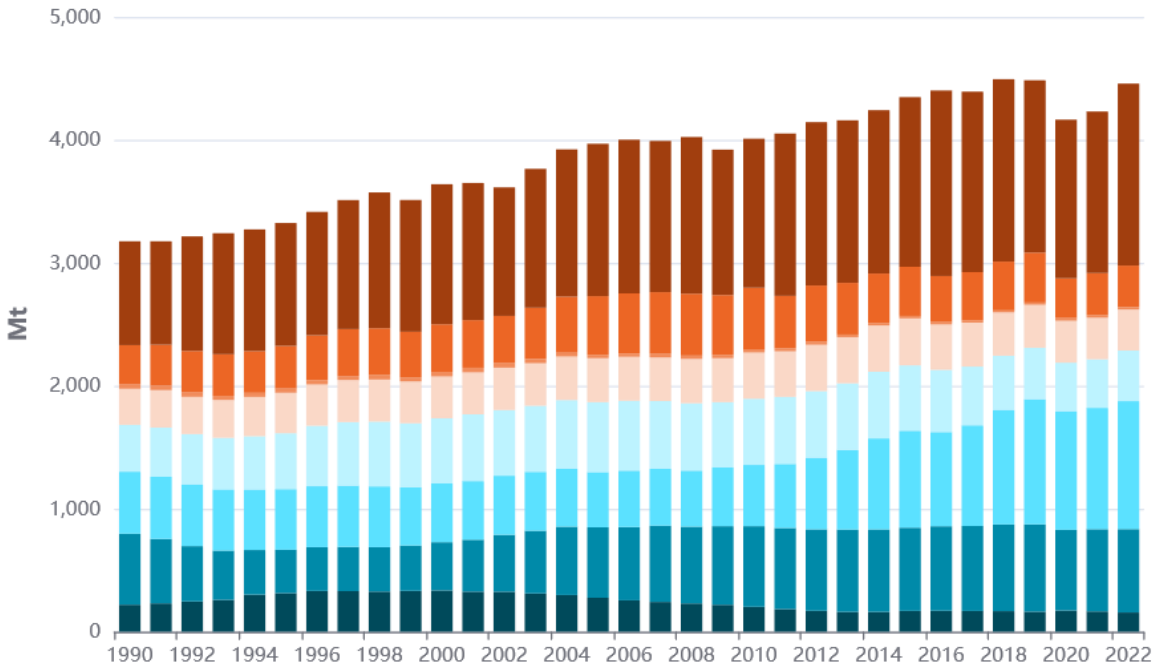
Az ábráról leolvashatóak az elmúlt évek nagy változásokat előidéző trendjei, többek között a már előbb említett koronavírus-járvány. Azonban itt is világosan látszik, hogy stabil gazdasági helyzet és kevés nagy változást okozó tényező mellett stabil emelkedés figyelhető meg a JET A1 típusú tüzelőanyag árában, ezzel is alátámasztva az elképzelésünket, hogy érdemes a jelenlegi viszonyok mellett hasonló vizsgálatokat végezni egy alternatív forrás megtalálásához.

1.1.3 Kitermelés és előállítás

A stabil monoton áremelkedést azonban nem magyarázzák az egyre csökkenő kitermelési ráták, hiszen éppen ellenkezőleg, a globális statisztikák azt mutatják, a jelenlegi kőolajkitermelésünk éppen, hogy emelkedik és teszi ezt az elmúlt években folyamatosan. A 4. ábra éppen ezt a tendenciát vizualizálja a számunkra. Az ábrán az is megfigyelhető, hogy míg az európai kitermelési mennyiségek az utóbbi évek során stabilok maradtak, addig a kitermelés javarészt eltolódott a közel-keleti régióba, ahol jelenleg is hatalmas kitermelés zajlik. Úgy gondolom, amiatt is fektetnek erre hangsúlyt, mivel az olcsó munkaerő és a kockázatvállalási hajlandóság ezekben a régiókban magasabb, mint az európai, vagy az észak-amerikai régióban. Azonban ne feledkezzünk meg arról, hogy mindezen feltételek mellett a jelenleg elérhető tartalékjaink végesek, valamint nem fog hosszútávon megoldást jelenti az, ha túlságosan függővé tesszük magunkat egyetlen forrástól. Szükség van jelenleg olyan, lehetőleg új és egyedi megoldásokra, amely függetlenít minden külső tényezőtől. Azonban ahhoz, hogy más típusú

tüzelőanyagot is képesek legyünk megfelelően használni és a repülésben megkívánt nagyon magas biztonsági és megbízhatósági követelményeknek eleget tudjunk tenni, vizsgálatokra van szükség.

A használni kívánt tüzelőanyag nagyon nagy valószínűséggel nem fog ugyanolyan égési tulajdonságokkal rendelkezni, mint egy JET-A1 típusú repülőgép kerozin, így ezeket érdemes lenne akár vizuális módon is megszemlélni, akár valós gázturbina érésterében is. Ehhez viszont szükségünk van egy olyan szabályozóra a tüzelőanyagot szállító ágban, amely a kívánt tömegáramot stabilan képes eljuttatni úgy, hogy a lehető legtisztábban biztosítsunk egyenlő feltételeket a vizsgálat során. Ezt a szabályozót szeretnénk elkészíteni a tanszéken található gázturbinás próbapadok és segédeszközök használatával ahhoz, hogy a jövőben ilyen kísérletek valósulhassanak meg.



4. ábra Globális nyersolaj-kitermelés alakulása az évek folyamán [5].

1.2 A szabályozás módja

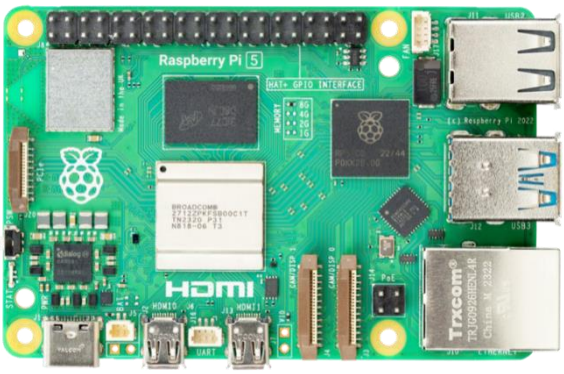
Egy szabályozott rendszernek nagyon sok követelményt kell egyszerre teljesítenie: legyen minél gyorsabb, legyen a lehető legpontosabb, mindezek mellett minél kompaktabb és beleférjen a megfelelő tárolóba. Ezen felül a repülőgépiparban a két legfontosabb kritériumnak is meg kell feleljen az eszköz: minél kisebb tömeggel rendelkezzen ahhoz, hogy ne vegyen el a hasznos teherből, valamint minél redundánsabb, minél megbízhatóbb rendszert képezzen. Ahogy az egyre bonyolultabb analóg szabályozók felé vesszük az irányt, úgy egyre inkább csak különböző kompromisszumokkal leszünk képesek mindezen követelményeket együttesen teljesíteni.

Ha nem is elektronikus úton történik a szabályozás, hanem mechanikus, hidromechanikus úton, akkor nagyon hamar vagy problémákba ütközünk vagy hatalmas tömeget kell a repülőgépünkön cipelni. Tekintsünk csak példának egy bonyolult analóg módon történő hidromechanikus szabályozó rendszert,

amely közvetett módon avatkozik be és ernyedő rugalmas visszacsatolással rendelkezik a gázturbina fordulatszámát tekintve! Ez a szabályozási módszer képes arra, hogy statikus körülmények között az egyszerűbb szabályozási módokhoz hasonlóan egy gép emelkedése közben képes a gázturbinát egy állandó és megfelelő fordulatszámon tartani. Ha a dinamikus viselkedését vizsgáljuk, akkor megfigyelhetjük, hogy az elvárt módon fog szabályozni, ha gyorsítani szeretnénk a gázturbina fordulatszámán, akkor megfelelően gyors reakcióidőt biztosít mindenféle lengések nélkül, emiatt nagyon jól használható. Azonban így egy nagyon bonyolult szerkezetet kapunk rengeteg kamrával, apró tolattyúval és kis alkatrésszel, valamint ezek együttes tömege jóval nagyobb lesz, mint azt elvárnánk egy kereskedelmi utasszállító környezetében, nem beszélve a hosszú és körülményes beszabályozási időről, amelyet elmulasztva nem fog megfelelő üzemben működni a szabályozó [6].

Adódik a lehetőség, hogy mindezek kiküszöbölésére használjunk olyan elektromos szabályozó rendszert, amely képes kielégíteni az előbb felvázolt követelményeket, azonban, ha ezeket pusztán analóg elektronikus eszközök útján szeretnénk elérni, szintén nagyon komplex és szerteágazó rendszert kapunk. Ekkor lépnek színre az otthoni környezetben már elterjednek számító számítógépek. Már a ’60-as években elkezdtek alkalmazni kisebb ilyen jellegű feladatokra számítógépeket, azonban ahhoz, hogy megfelelően integrálható eszközt kapjunk, ezt egy kezelhető méretre kell redukálnunk. Ebben az időben jelmező volt az is, hogy egyre csökkenő méreteket voltak képesek a gyártók előállítani, a korszerű tranzisztorokkal már olcsó és apró eszközök is készültek, amelyek a hatalmas potenciáljuk mellett egy kedvező árral is rendelkeztek, így alkalmasak voltak arra, hogy tömeggyártott fogyasztási cikkekben is alkalmazhatóak legyenek, ekkor születtek meg az úgynevezett mikroszámítógépek [7].

Mindezek mellett hatalmas előnyük ezeknek az eszközöknek, hogy elkészíthető belőlük egy úgynevezett beágyazott rendszer (*embedded system*), amely egy nagyon sok más komponenssel együtt integrált áramkörbe beépített mikroprocesszorból áll. Ezek a beágyazott rendszerek rendelkeznek olyan számítási kapacitással, hogy általános felhasználású számítógépek legyenek, ekkor tudjuk őket a beprogramozásuk után voltaképpen bármilyen feladatra alkalmasság tenni, mint egy italautomata, vagy akár egy szivattyú fordulatszámának szabályozása [8]. Egy ilyen korszerű mikroszámítógépet mutat be az 5. ábra.



5. ábra Raspberry Pi 5 mikroszámítógép [9].

Ha zárt hurkú szabályozásban szeretnénk elképzelni egy ilyen eszközt, akkor beláthatjuk, hogy nagyon sok előnyük van. Nagyon kis tömegűre képesek vagyunk leredukálni egy ilyen komplex eszközt, ha egyedileg terveztetjük meg, akkor akár csak a feltétlenül szükséges komponenseket hagyhatjuk rajta az integrált áramkörön, az $1\,cm^2$-es felületet is el tudjuk képzelni. Emellett nagyon komplex számításokat tudunk végeztetni velük a megfelelő számítási kapacitással, olyat, amilyet egy analóg eszköz csak nagyon bonyolult úton lenne képes végrehajtani. Ezen felül az ilyen eszközök nem igényelnek beszabályozást, csak előzetes kalibrációt, nem kopnak és nem használódnak el az idővel, kivéve persze a különleges jelfogókat, aktuátorokat.

Ehhez szükségünk van magára a mikrokontrollerre, valamint külső jelfogókra. Ahhoz, hogy valós idejű zárt hurkú szabályozást legyünk képesek megvalósítani egy digitális számítógéppel, át kell gondolnunk azt, milyen módszerrel is történik a szabályozás. Egyetlen hátrányuk az ilyen szabályozóknak, hogy nem képesek folytonos idejű szabályozásra, egy bemenetre csak adott időtartam múlva képesek reagálni, amit mintavételezési időnek hívunk. Ez az időtartam a mikrokontroller saját számítási kapacitásából és a beprogramozott algoritmus sebességéből adódik elsősorban, ennél nem lehet rövidebb, de hosszabb igen. A legfontosabb feladatunk emiatt az, hogy a lehető leggyorsabb algoritmust találjuk meg, minél jobban hasonlítson a szabályozás egy folytonos idejű szabályozásra. További adódó feladat ekkor, hogy a meglévő egyenleteket, összefüggéseket alakítsuk át oly módon, hogy azok képesek legyenek egy ilyen diszkrét idejű szabályozóval ellátott rendszer viselkedését megfelelően modellezni, leírni, úgy, hogy ne tapasztaljunk eltéréseket.

Mindezek miatt korszerű technikának tartható ez a típusú szabályozás, éppen megfelelő lehet egy tüzelőanyag-szivattyú számára. Annál is inkább, mivel, ha moduláris jelleggel építjük fel a programunkat, valamint a használt eszközeinket, sokkal egyszerűbb feladatot fog jelenteni a jövőben ennek a próbapadnak és azt ezt szabályozó programnak az integrálása alternatív tüzelőanyagok vizsgálatához.

2 Módszerek és eszközök

A dolgozat célja a TKT-1 egyáramú, változtatható fúvócsővel felszerelt gázturbinás sugárhajtómű tüzelőanyag-ellátását biztosító 924 típusú szivattyúegység vizsgálatára alkalmas próbapadon az említett berendezés fordulatszám-szabályozásának megvalósítása, így ezzel a későbbiekben további kísérletek és mérések valósulhatnak meg ezen a gázturbinán. A cél az, hogy a jelenlegi tüzelőanyag-rendszer kiegészíthető legyen egy másodlagos alternatív betápláló rendszerrel, amely a későbbiekben alternatív tüzelőanyagok vizsgálatára lesz alkalmas.

6. ábra: TKT-1 egyáramú, változtatható fúvócsővel rendelkező gázturbinás sugárhajtómű [10].

A TKT-1 gázturbinás sugárhajtómű egy külön padra felszerelt TSz-21 típusú, MiG-23 szuperszonikus vadászrepülőgépekben alkalmazott indító gázturbinából készült, oktatási célból használt hajtómű, melyet a Repüléstudományi és Hajózási Tanszék tett teljeskörűen üzemképessé 2007-ben. A gázturbina mivel eredetileg indítóhajtóműként látta el a feladatát, ezért egy fúvócsővel lett felszerelve, amit később átalakítottak egy kézi vezérlőkar segítségével módosítható keresztmetszetű fúvócsőre, amit a kívánt helyzetbe állíthatunk ahhoz mérten, hogy mekkora kiáramló gázsugár-sebességet szeretnénk elérni a hajtóművel [10].

A tanszéken korábban felmerültek ötletek olyan kísérletekre, melyek során alternatív tüzelőanyagok égési és lángtulajdonságait lehetne vizsgálni a már megépített TKT-1-es gázturbinás sugárhajtóművön, amit egy, a gázturbina égésterében készített furaton keresztül behelyezett kamerával lehetne vizuálisan is megfigyelni, lehetővé téve azt, hogy szemmel is lássuk, mi történik egy gázturbina égésterében. Ehhez azonban a teljes egységet működésképessé és aktualizálttá kell tennünk. Sajnos mivel a kézi vezérlés esetében nem lehetséges olyan jellegű pontos és minden tekintetben független, csak a tüzelőanyagtól és a közegtől függő környezetet előállítani a gázturbina égésterében, amivel az alternatív égési tulajdonságok megfelelő pontossággal és függetlenséggel kimérhetőek lennének, így szükségessé vált egy olyan szabályzórendszer készítése, ami a tüzelőanyag-szivattyút egy előre beállított gázturbina üzemállapot szerint olyan szinten tartja, amely képes a gázturbina teljesítményét változatlan szinten tartani.

2.1 Szivattyúvezérlés

A szivattyú vezérlését többféle irányból közelítettük meg a méréshez. A tanszéki laboratóriumban található egy, külön a szivattyú részére épített próbapad, amely eredetileg a szivattyú különböző üzemi pontjain történő tüzelőanyag szállítások méréséhez került megépítésre, azonban a mi méréseinkhez is tökéletesen megfelel. Mivel a próbapad hálózati feszültség felhasználására alkalmas, így ez nagyban leegyszerűsíti a szivattyú meghajtását.

A 924-es szivattyút hajtó elektromos motor egyenárammal működik, névleges feszültsége $27\,V$, ezt egy transzformátor segítségével tudjuk elérni [11]. A bemenő váltakozó hálózati feszültséget egy több kimenettel rendelkező, ezáltal változtatható szekunder feszültségű transzformátor segítségével alakítjuk át $24\,V$-ra, amely ugyan nem egyezik meg a szivattyú névleges, $27\,V$-os feszültségével, azonban mivel a majdani TKT-1-es hajtóművön is ezzel a feszültséggel fogjuk működtetni, valamint problémamentesen tud így is üzemelni, ez számunkra nem fog problémát jelenteni. Az így kapott váltakozó, $24\,V$ egyenértékű feszültséget ezután egy Graetz-híddal egyenirányítjuk, amely a szinuszosan váltakozó feszültség szinusz-hullámainak abszolút értékét veszi, azaz egy irányú, de változó nagyságú feszültséget kapunk. Ezt egy kellően nagy kapacitású kondenzátorral kisimíthatjuk úgy, hogy nagyon közel kerülünk a tökéletes egyenfeszültséghez.

Ha ezt a 24 V-ot direktben a szivattyú villanymotorjára kapcsolnánk, a szivattyú közel a névleges üzemállapotában működne közel maximális fordulatszámon, amely az eredeti rövid időre tervezett működési idő és az üresjárat miatt egyaránt nem előnyös. Emiatt szükség van egy olyan szerkezetre, amely segítségével a villanymotor fordulatszámát szabályozni tudjuk.

Az eredeti szerkezetben a Graetz híd előtti váltakozó feszültség egy vele sorba kötött TRIAC-ra (Triode for Alternating Current) érkezett, mellyel a váltakozó feszültség effektív nagyságát lehetett egy kézi potenciométerrel szabályozni. Ebben a rendszerben a feszültség szinuszosan váltakozó függvényének nullátmenetét tudtuk mozgatni, oly módon, hogy a szinuszos feszültség nullátmenetének időpillanatában még nem engedtetjük a TRIAC kapun át a feszültséget, hanem csak egy kicsiny idő után, ennek az eredménye egy csonkított szinusz-hullám lesz, mely csak akkor ér el a névleges értékére, amikor a TRIAC vezérlő jelet kap. Ezzel elérhetjük azt, hogy mekkora effektív feszültséget szeretnénk a villanymotoron láttatni [12].

2.2 Új vezérlés

Az előzetes tesztüzemek során azonban sajnos ez a rendszer meghibásodott, így egy új fajta vezérlés elkészítése vált szükségessé. Az új rendszerben egy mikrokontrollerrel vezéreljük a villanymotorra jutó egyenfeszültséget az alább ismertetett módon. A transzformátor szekunder oldaláról lecsatolt váltakozó, $24\,V$ effektív nagyságú feszültséget közvetlenül juttatjuk a Graetz hídra, így már a kör elején megvalósul az egyenirányítás és már csak ezzel a feszültséggel kell csak dolgoznunk. Egy *Arduino Mega* típusú mikrokontroller egyik analóg bemenetére egy potenciométert kapcsolunk, ez lesz

a referencia értékünk, amellyel majd a villanymotort és ezáltal a szivattyút tudjuk szabályozni. Ezzel az analóg bemenettel arányosan a mikrokontroller egyik impulzusszélesség-modulált (pulse width modulation, PWM) kimenetére egy, a potenciométer állásától függő kitöltési tényezővel rendelkező jelet bocsájt ki. Ezt az impulzusszélesség-modulált jelet vezetjük rá egy motorvezérlőre, amivel a későbbiekben fogunk foglalkozni.

2.3 Az identifikáció folyamata

Azért van szükségünk identifikációra, mivel nem állnak rendelkezésre információink arról, miként reagál pontosan az adott elektromos meghajtású szivattyú a különböző bemenetekre, valamint ennek a meghatározására felhasználható, motorra jellemző konstansok sem adottak. Az eredeti használati helyén ez a szivattyú nem volt vezérelve, mivel a TSz-21 indító gázturbinaként szolgált, így annak nem volt szükséges sok ideig üzemelnie, elegendő volt rövid ideig működtetni azt *ON-OFF* állásokban, így már csak ezért sem állhatnak a rendelkezésünkre ezzel kapcsolatban adatok. Ez egy tipikus identifikációs megoldást kínál a számunkra. Ekkor megfelelő módszerrel történő meghatározás után lehetőségünk nyílik kalibrációra, valamint irányítási rendszer tervezésére [13]. Mivel nem állnak rendelkezésre számunkra előzetes információk arról, miként viselkedik az általunk vizsgált rendszer, azaz nincsen előzetes modellünk sem a kísérlet során, ezért úgynevezett fekete doboz modellként fogjuk ezt kezelni, valamint a legtöbb szabályzástechnikai probléma is hasonló módon jár el. Ekkor egy vizsgált bemenetre egy adott kimenet viselkedését vizsgáljuk mindenféle előzetes ismeret nélkül. A modellünkön továbbá feltételezzük, hogy az időben invariáns lesz, azaz nem fogjuk figyelembe venni azt, hogy előzetesen milyen hosszan üzemelt a szivattyú. Ekkor elhanyagoljuk a szivattyú használatából adódó kopásokat, elhasználódásokat és azt feltételezzük, hogy az alatt az idő alatt, amíg mi a méréseket el fogjuk végezni, a csapágyak, a tömítések és a szűrők állapota nem fog romlani, hanem közel azonos állapotban maradnak [14].

Az identifikáció során abból a feltételezésből indulunk ki, hogy egy teljesen általános, időtartománybeli modell leírható egy általános, parametrikus egyenlettel [15].

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t - n_k) + \frac{C(q)}{D(q)}e(t) \qquad \textit{2. egyenlet}$$

Itt $u(t)$ a bemeneti, $y(t)$ a kimeneti, $e(t)$ pedig a hibafüggvényt jelenti, amely zajként jelenik meg. t az időegység, n_k pedig a bemenet késleltetése, ez általában a $B(q)$ polinom legmagasabb foka előtti zérusok számát jelenti. Az $A(q)$-$F(q)$ függvények pedig polinomiális alakban adottak, ahogy azt a 3. egyenlet mutatja. Azonban fontos azt hangsúlyozni, hogy itt q hatványai annak időbeli múltját mutatják meg, azaz például q^{-2} jelenti az x_{t-2} értéket, tehát a két lépéssel ezelőtti értékét (*backward shift operator* [16]). A többi polinom hasonló alakban írható fel, mint az $A(q)$, kivéve a $B(q)$-t (4. egyenlet). Egy modell fokszáma alatt az egyes polinomok n_x kitevőit értjük.

$$A(q) = 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a} \qquad \textit{3. egyenlet}$$

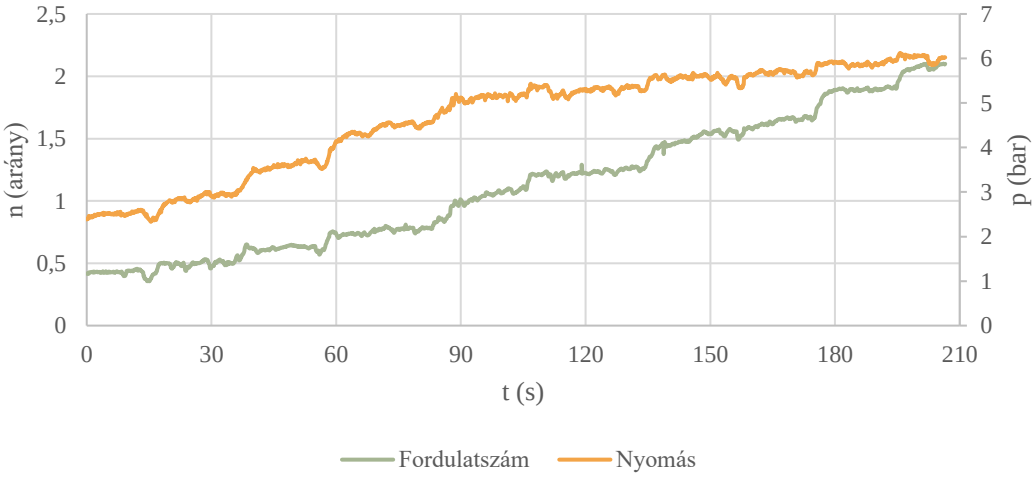
$$B(q) = b_1 + b_2 q^{-1} + \cdots + b_{n_b} q^{-n_b+1} \qquad \textit{4. egyenlet}$$

Ebből aztán felépíthető a különböző polinomok fokszámainak, valamint alapvető értelmezésének variációival maga az identifikációs művelet. A későbbiekben a MATLAB szoftvert fogjuk arra használni, hogy ezekkel könnyedén tudjunk számításokat végezni. Itt manuálisan fogjuk tudni megadni a modellünk típusát (például output-error) és annak fokszámát is. A következő fejezetben a próbamérés során egyféle modellt mutatunk be, azonban fontos, hogy egy identifikációs feladat során a lehető legtöbb ilyen modellt kipróbáljuk, hiszen könnyen lehet, hogy az egyik sokkal jobban illeszkedő eredményt fog adni, mint egy másik, nekünk pedig ezt kell megtalálnunk, így ezt azok előfordulásukkor fogjuk részletesen bemutatni.

Az identifikációhoz szükséges adatok mérését több irányból is megközelíthetjük, attól függően, idő- vagy frekvenciatartománybeli analízist szeretnénk végrehajtani. A következő próbamérésben, ugrásszerűen, lépcsőzetesen lett növelve az adatok szerint a mérés. Míg az időtartománybeli analízis a tranziensek, időállandók és a dinamikus viselkedés meghatározására szolgál, addig a frekvenciatartománybeli analízisből közvetlenül nyerhető ki a rendszer változó gerjesztésre adott válasza a teljes tartományban [13].

2.4 Identifikációs próba

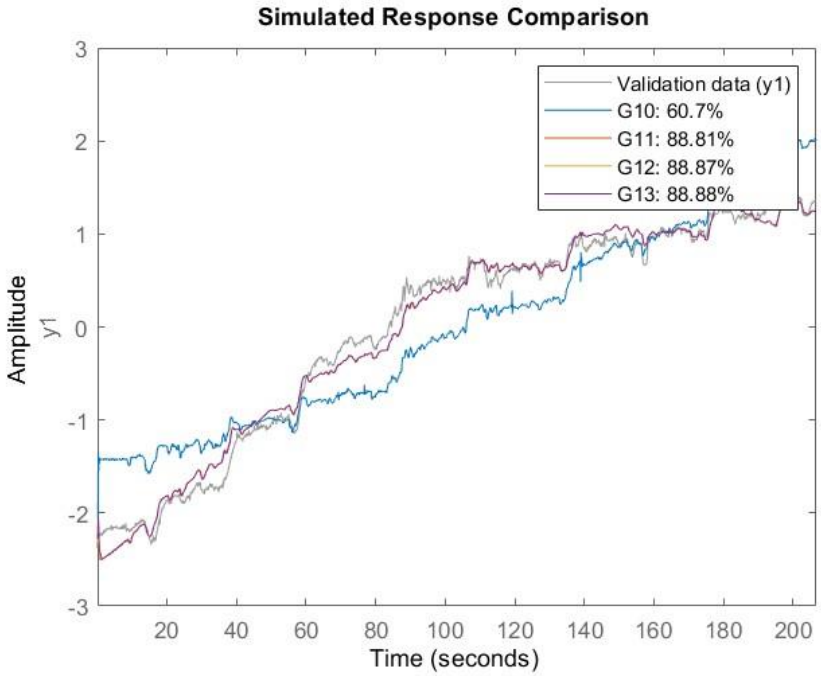
Az identifikációhoz szükséges egyenletek megoldását MATLAB programmal fogjuk végezni. A program *System identification toolbox* kiterjesztésével lehetőségünk lesz a nyers mérési adatokból könnyedén a program segítségével közelíteni a valós rendszer által látott $G(s)$ átviteli függvényt. Erre több módszert is kipróbáltunk látszólagos mérési adatokkal. A következőkben ezt a próbaidentifikációt részletezzük.



7. ábra A próbaidentifikáció során használt adatsorok, bemenetként a fordulatszám, kimenetként a nyomás olvasható le.

Adott egy korábbi, nem általunk végzett, diszkrét idejű mérési eredménysor, amelyben az adatokat $\Delta T = 0{,}1\ s$-os gyakorisággal rögzítettük. Itt szintén a már bevezetett 924-es szivattyún végeztük a mérést, azonban a rögzített adatsorok a szivattyú fordulatszáma és az általa elért tüzelőanyag-nyomás volt. A mérés körülbelül három és fél percig tartott, ekkor összesen mintegy 2000 adat keletkezett. A fordulatszám növelése lépcsős módon történt, ahogy azt az alábbi diagram is mutatja.

Ezt követően a MATLAB-ba az adatok már importálhatóak is voltak. Itt az egyik identifikációs eljárás kiválasztjuk, majd a függvényargumentumba beírjuk a megfelelő paramétereket az identifikációhoz, jelen esetben az *Output-error* módszert választjuk. Ekkor a program által megoldás az 2. egyenlet béli kiinduló paramétereket alkalmazva az *Output-error* módszer a 5. egyenlet segítségével fogja közelíteni a valódi átviteli függvényt, azaz ekkor $A(q) = C(q) = D(q) = 1$. Ezt a későbbiekben felhasználhatjuk egy szintén ebbe a programba épített segédletben, ahol pedig PID-szabályzást tudunk tervezni a PID-Tuner MATLAB-kiterjesztéssel az így kinyert átviteli függvényekre.



8. ábra Output-error módszerrel közelített szimulált válaszfüggvény MATLAB-ból exportált diagramjai. A növekvő számok egyre növekvő pontosságot mutatnak. A B és F polinomok fokszámai a legpontosabb, G13 függvény esetében rendre 2 és 4.

$$y(t) = \frac{B(q)}{F(q)}u(t - n_k) + e(t) \qquad \textit{5. egyenlet}$$

2.5 mbed mikrokontroller

A szabályozás megvalósításához elengedhetetlen egy vezérlő eszköz alkalmazása is, amelyhez mikrokontrollert fogunk használni. Ezek az eszközök nagyon jól alkalmazhatóak zárt hurkú vezérlés tervezésére, hiszen a megfelelő modellek képesek egyszerre nagyon sok adatot feldolgozni, azokkal

számolni és azokat hibával terhelt értékekként kezelve értékelni [8]. Az előnyük abban rejlik, hogy könnyen megvalósítható velük bármilyen feladat, hiszen, ha megfelelő bemeneteket használunk egy ilyen mikrokontrollerhez, akkor az könnyedén tudja az előírt számításokat végrehajtani, különféle szenzorokat, illetve aktuátorokat csatlakoztathatunk ezekhez, ezzel pedig megvalósítva a való világgal való kapcsolatot.

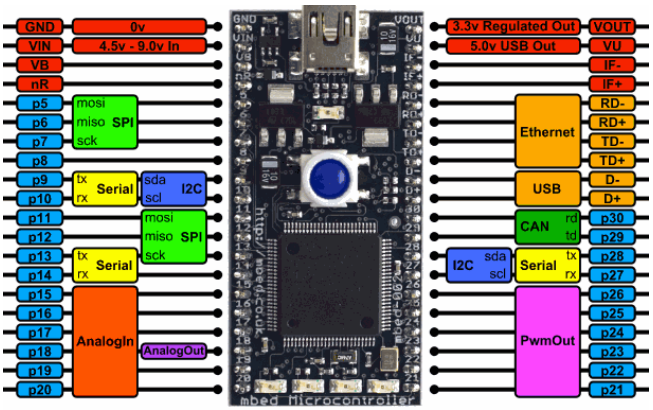
Ezek az eszközök több alegységből összeállva képezik azt a teljes számítógépet, amelyet mi saját programmal elláthatunk. Egy ilyen részegységet képez az LPC1768-as mikrokontroller integrált áramkör is, ezt ráépítve egy hordozóra, amellyel a külvilággal való kapcsolatokat meg lehet majd valósítani, kapjuk az egész egységet. Ebben található meg a programok tárolására szolgáló programmemória, a logikai és aritmetikai számításokat elvégző aritmetikai-logikai egység és egy órajelet szolgáltató kvarcoszcillátor. Az órajel határozza meg a számítások ütemét, a program lefutásának sebességét és az események megfelelő időzítését.

Az általunk használt termék az mbed LPC1768 lesz, amelyet a 9. ábra mutat. Azért erre esett a választás, mivel kis méretű, ismert, C++ programnyelvet használ, valamint már megtalálható a tanszéki laborban. Az LPC1768 esetében az órajel 96 MHz, ez nekünk megfelelő lesz a szabályozás tervezése során. Az eszköz rendelkezik továbbá 25 egyedi be- és kimenettel, azok különböző konfigurációban használhatóak (10. ábra). Számunkra az analóg bemenetként használható lábak lesznek a legfontosabbak, hiszen így majdnem közvetlenül az egy-egy szenzortól érkező információt fel tudjuk dolgozni különféle külső jelátalakító berendezések nélkül, mint például egy analóg-digitális átalakító, hiszen a kontroller belső analóg-digitális átalakítóval már gyárilag beépítetten rendelkezik. A tesztek során hasznosnak bizonyulhat az eszköz számítógéppel való, soros kommunikáción keresztül történő adatátvitele is, amellyel egy konzol programmal kiolvashatjuk az eszköz által az általunk küldésre indított jeleket, ezzel konkrétabb képet kapva arról, belül mi történik.

9. ábra mbed LPC1768 [17].

A kontroller egyik másik nagyon szembetűnő tulajdonsága, amelyet a 10. ábra mutat, hogy a kontroller képes standard soros adatátvitelre a beépített USB átalakító használata nélkül is. Például a 9-10, 13-14, illetve a 27-28 lábakra egyesével tudunk újabb és újabb adatközlő berendezéseket csatlakoztatni, amelyekkel egyéb kommunikáció is meg tud így valósulni. A mi esetünkben később

szeretnénk egy RN41 Bluetooth-modult is a tesztpadra építeni, ezzel lehetővé téve a vezeték nélküli adatátvitel lehetőségét is, így akár egy telefon segítségével soros be- és kimenet kezelésére alkalmas applikáción, úgynevezett terminálon keresztül kommunikálhatunk a berendezésünkkel, így olyan hasznos adatokat nyerhetünk ki, amelyek egy esetleges szelektív adatátvitel esetén nem állnának a rendelkezésre. Ez lesz az úgynevezett debug mód.



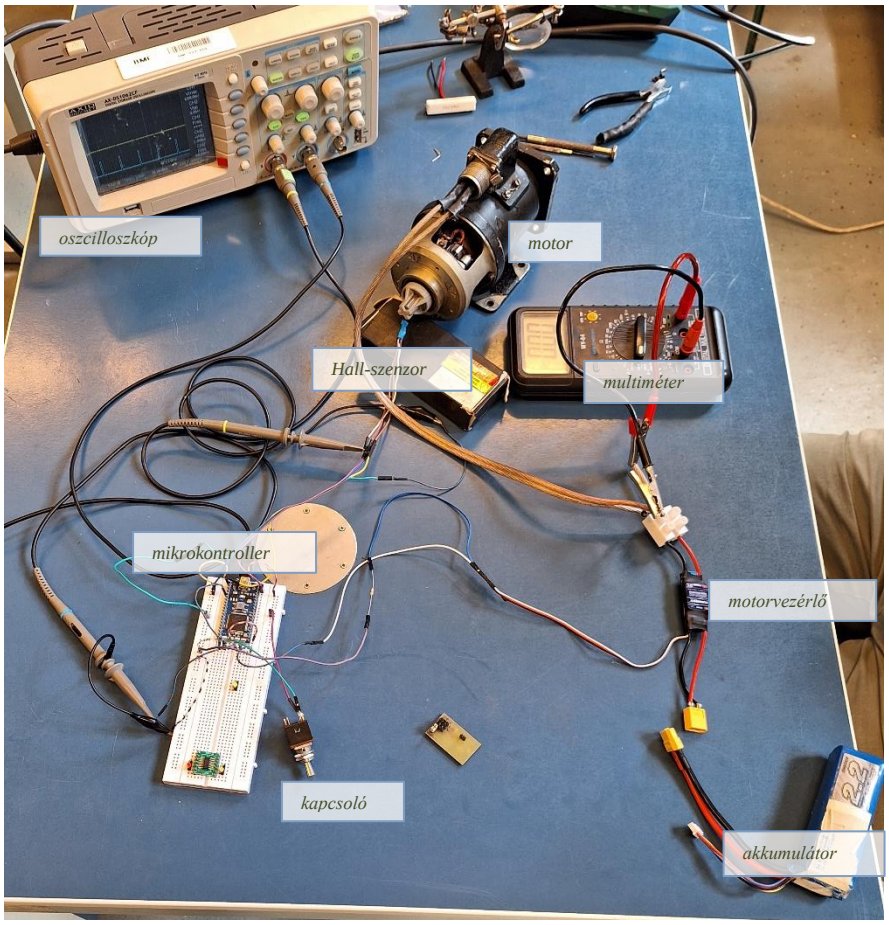
10. ábra mbed LPC1768 lábkiosztása [8].

3 Próbamérés

3.1 Kapcsolás és program

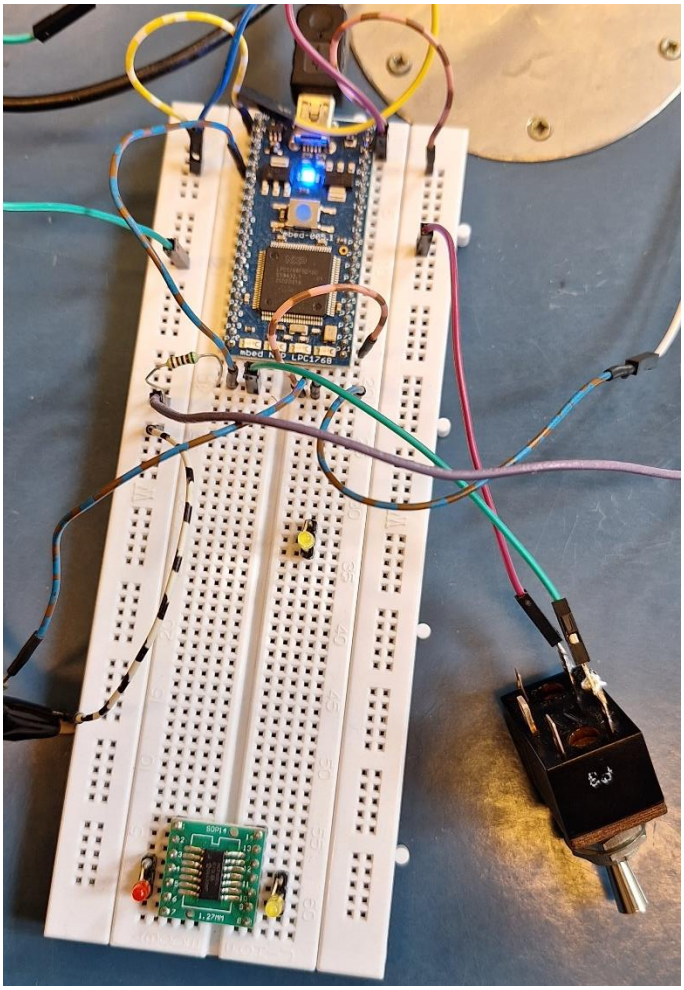
Az első alkalommal még megfelelő vezérlő nélkül kellett kipróbálnunk a szivattyút hajtó villanymotor vezérlését, hogy az megvalósítható-e az általunk gondolt eszközökkel. Ehhez a kísérlethez egy modellezéshez gyakori eszközt használtunk fel a villanymotor vezérlésére, egy $12\ V$ egyenárammal működő, kis repülőgépekhez is előszeretettel használt motorvezérlőt, a *Turnigy 30A Brushed ESC*. Ez akár egy $12\ V$-os akkumulátorról is meghajtható, amely feszültséget szaggatva, négyszögjel formájában továbbítja a villamos motor felé a vezérlő jel függvényében [18]. A vezérlő jelként egy mbed mikrokontroller által egy előre definiált és általunk kezelt PWM jelet fogunk használni, amely $0\ V$ és $+3{,}3\ V$ között váltakozik. A motorvezérlő névleges kapacitása $30\ A$. Mivel a 924-es szivattyú áramfelvétele akár $70\ A$ is lehet, valamint annak névleges feszültsége $27\ V$, így tartósan nem hajtható végre véglegesen ezzel a mérés, hiszen a valódi beépítési környezet nem ezt fogja megkívánni, azonban rövid idejű próbaüzemre megfelelő lesz ez a vezérlő is. Az összeállított kapcsolást a 11. ábra mutatja.

A PWM jel előállításásra a már említett mbed kontrollert használtuk. Erre a következő célszerű módon készült a program. A kiadott PWM jel frekvenciája a motorvezérlő gyártói specifikációja alapján $50\ Hz$ kell legyen, valamint a hasznos, kontroller által érzékelhető tartomány az 5 és 10%-os kitöltési



11. ábra A próbamérés során összeállított áramkör.

tényező közé kell essen. A későbbi valódi identifikációs méréshez előre gondolva már a programban gombnyomásra egyre növekvő kitöltési tényezőt alkalmazunk, ezzel a motor fordulatszáma az idő függvényében lépcsősen fog növekedni, ez a későbbi identifikációhoz lesz előnyös. Minden gombnyomásra 0,5%-kal növekszik a kitöltési tényező, így tudunk magunknak 10 lépcsős felbontást biztosítani.



12. ábra A mikrokontroller tényleges kapcsolása. Középen az mbed LPC1768 kontroller, felül a soros kommunikációt megvalósító számítógépes kapcsolathoz használt mini-USB kábel, jobb alul pedig a kitöltési tényező növelő kapcsoló látható.

A motor fordulatszámát egy, a motor tengelyének végére erősített mágnes és egy Hall-szezor segítségével mérjük. A mágneses tér periodikus váltakozásának hatására ennek az eszköznek a kimenetén éppen egy négyszögjel fog keletkezni. Ennek a jelnek a frekvenciája éppen meg fog egyezni a motor fordulatszámával.

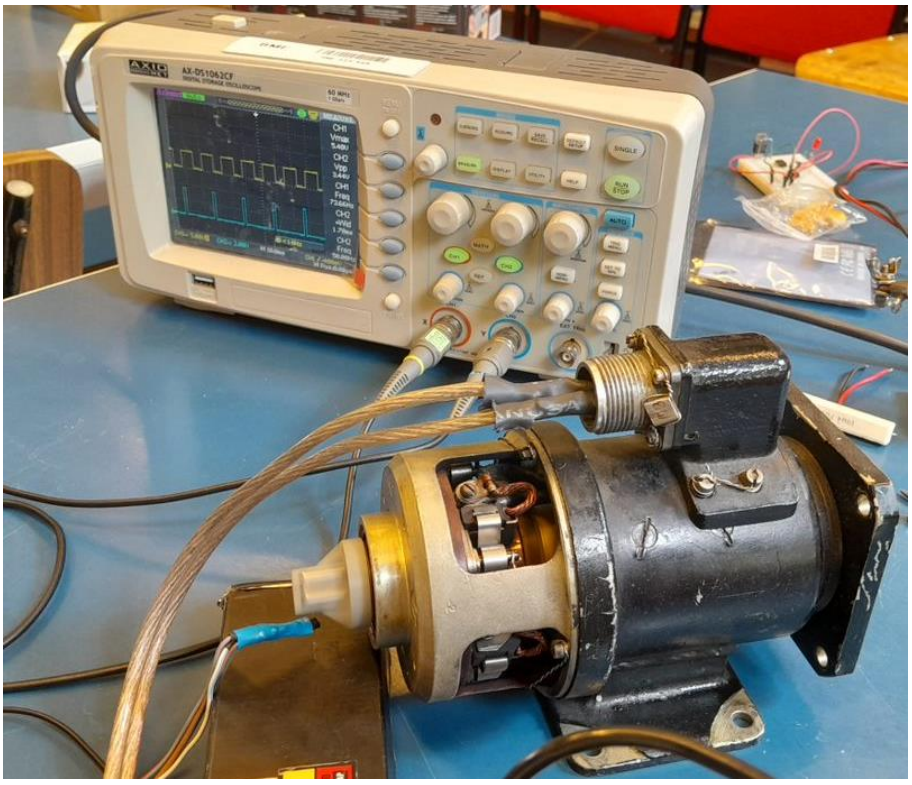
A méréshez illetve a visszajelzéshez több eszközt is használtunk. A PWM megfelelőségének a biztosítására három módszert is alkalmazunk: az oszcilloszkóp egyik szondáját a kimenetre párhuzamosan kötjük, így megbizonyosodhatunk arról, hogy a jel megfelelő frekvenciával és kitöltési tényezővel rendelkezik; a mikrokontroller soros USB-kimenetén keresztül egy számítógépre küldjük az éppen aktuális, kontroller által értelmezett kitöltési tényezőt; a kontrolleren található négy LED

segítségével azonnal visszajelzést kaphatunk a jelenlegi állásunkról négy szakaszban. Eközben a motor fordulatszámának a kijelzését az oszcilloszkóp másik szondájára bízzuk, azt a Hall-szenzor kimenetére erősítve láthatjuk az általa generált négyszögjelet, amelynek a frekvenciáját az oszcilloszkóp beépített mérési funkciójával könnyűszerrel manuálisan is meghatározhatjuk.

3.2 Eredmény

3.2.1 Identifikáció

A próbamérést végrehajtva arra a konklúzióra jutottunk, hogy az általunk elképzelt módon történő identifikáció megvalósítása megfelelő lesz a későbbiekben, így ezt a módszert fogjuk alkalmazni. A próba során felmerült több probléma is, amelyet a helyszínen tudtunk orvosolni. A mikrokontroller *p5* lábát használtuk a kapcsoló állásának meghatározására, azonban mivel ezt a program digitális bemenetként kezelte, így egy kisebb zavar, apró feszültségnövekedés is elég volt ahhoz, hogy 1 bemenetként értékelje azt, ekkor tévesen emelt a vezérlő jel kitöltési tényezőjén. Ezt egy kellően nagy ellenállás beiktatásával tudtuk orvosolni, ezt párhuzamosan a föld és a kapcsoló közé kötöttük. Arról is megbizonyosodtunk, hogy a mikrokontroller megfelelő eszköz lehet arra, hogy egy előre meghatározott bemenet alapján készítsen vezérlő jelet, ehhez megfelelő annak adatkezelési képessége.



13. ábra Az oszcilloszkóp alkotta kép a mérés során értelmezett jelekről, sárga színnel a Hall-szenzor által mért, fordulatszámmal arányos jel, kék színnel pedig a vezérlő jel látható.

Ahogy azt a 13. ábra mutatja, nagyon szép és jól értelmezhető adatokat kaptunk az oszcilloszkópos mérés során. A Hall-szenzor jól kezelhető négyszögjelet adott, ennek a kitöltési tényezője a mágneses tér szimmetrikussága miatt körülbelül 50%, így a frekvenciából már következtethető a fordulatszám is,

tehát ennek segítségével jól megvalósítható a vezérlés és szabályozás. Ez látható is sárga jelzéssel az ábrán. A próbamérés során nagyon szépen látható volt a négyszögek közötti tér csökkenése a motor fordulatszámának növekedésével. A kék színnel jelzett vonalak a vezérlő PWM jelet tartalmazzák. A majdani elkészítendő vezérlést azonban ennél nagyobb felbontásúra szeretnénk elkészíteni, ahogy látható nagyon keskeny sávokban olvasható csak ki a jel, ekkor a zavarok ezt nagyban befolyásolhatják és nem lesz megfelelő követésű a szabályozás.

3.2.2 Soros kommunikáció szükségessége

További megfigyeléseink voltak, hogy sokkal kézenfekvőbb lenne a mikrokontrollert számítógépes terminálon keresztül vezérelni saját egyénileg definiált parancskódokkal. Erre egy példakódot az tartalmaz részletesen. E felismerés során el is készítettünk egy próbát, amelyben a mikrokontroller beépített LED visszajelzőjét irányítjuk soros kommunikációval egy számítógépről. A kontrolleren a jelek megfelelő elkülönítése, valamint a zavarok kiküszöbölése végett bevezettünk egy karaktert, amely arra szolgál, hogy csak ennek a beérkezése esetén értelmezzen az bármely parancsot, ez a mi esetünkben a ’/’ jel lett.

3.2.3 Adatok értelmezése

Ahhoz, hogy a beérkező adatokat megfelelően tudjuk vizualizálni és egy számítógép képernyőjén megjeleníteni sajnos nem elég egyetlen terminál alkalmazás, ehhez valami olyasmire lenne szükségünk, amely képes valós időben mutatni az irányító összes releváns paraméterét. Emiatt a LabVIEW programhoz fogunk fordulni. Erről a későbbiekben fogunk részletesebben beszámolni.

4 A mérés bemutatása

4.1 Felépítés

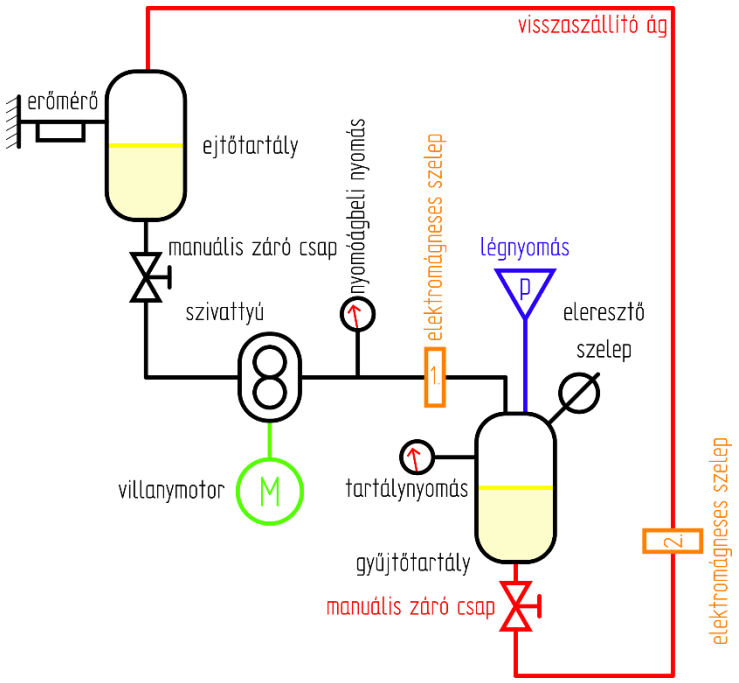
Miután összeállt a végleges kép arról, hogyan fogjuk megvalósítani az identifikációt a szivattyú próbapadon, módosítottuk a próbapad elvi felépítését aszerint, hogy a lehető leggyorsabban tudjunk érdemi méréseket végezni, hiszen fennállhat annak a veszélye, hogy egy adatsorral nem tudunk megfelelő és elfogadható eredményeket produkálni, így szükség volt arra, hogy minél könnyebben és szélesen változtatható paraméterek között tudjuk ezt végrehajtani. Eszerint készült el a mikrokontrollerre írt program is, a lehető legtöbb adatot a program módosítása nélkül beírhassunk „kézzel" és már kezdhessük is az identifikációt.

4.1.1 A közeg útja a próbapadon

A mechanikus elvi felépítést, pontosabban a szivattyúzott közeg által megtett utat több lépcsőben is biztonságossá és szivárgásoktól mentessé kellett tennünk. Annak érdekében, hogy a padra a lehető legkevesebb jusson a szivattyúzott közegből, az elektromos szelepek mellett mechanikus zárószelepeket is beépítettünk, amire azért volt szükség, hogy álló helyzetben ne történjen véletlenül sem eresztés a rendszerben, hiszen a közeg, mivel repceolajról van szó, igen ragacsos és az általuk bevont felületek tisztítása is nehézkes. A mérés elvi vázlatát a 14. ábra mutatja.

A fogaskerékszivattyú egy ejtőtartályból szivattyúzza ki a repceolajat. Ez az ejtőtartály a későbbi tömegáram méréshez elengedhetetlen erőmérő cellára van felfüggesztve, amely kimenete a mikrokontrollerbe jut. Az ejtőtartály alján található egy szorítóbilincs, amit jelenleg manuális elzáró szelepként alkalmazunk, hiszen, ha a mérés során használt műanyag szilikon csöveket elszorítjuk, akkor nem történik rajtuk átfolyás. Ezután jut a közeg az 1. számú elektromágneses szelephez, amelyet a próbapadról irányíthatunk (lásd 15. ábra), így szigorúan csak akkor fog bármilyen jellegű átfolyás is történni, amennyiben ezt engedélyezzük elektromosan, tehát a padnak mindenképpen rendelkezni kell elektromos táplálással. A szivattyú a szállított közeget ezután egy azzal egy szintben lévő gyűjtőtartályba porlasztja be. A tartályban egy fúvóka található, amibe a szivattyú az általa szállított közeget nagy nyomáson eljuttatja, ezzel is a TKT-1-be való beépítését elősegíti, hiszen ekkor sokkal közelebb képet fogunk a valós alkalmazáshoz kapni az identifikáció során. Ebben a tartályban található még több bemenő szilikon cső is, ezek közül az egyik az éppen tartályban lévő nyomás kijelzésére továbbítódik a próbapad kezelőfelületére a másik pedig egy dugattyús kompresszortól érkező légnyomást szolgáltató cső. A szivattyú üzeme és az identifikáció során a gyűjtőtartály alatt lévő elzárószelep, illetve a 2. számú elektromágneses szelep végig zárva vannak, ezzel garantálva azt, hogy az ejtőtartályba semmiképpen sem jusson vissza a már átszivattyúzott közeg. Egy ciklus végeztével és a gyűjtőtartály feltöltődése után jut szerepre a már említett sűrített levegős ág. Fontos azonban megemlítenünk, hogy ennek a levegős ágnak kettős szerepe is van, hiszen ha a gázturbina égésterében fellépő ellennyomást szeretnénk valamilyen módon szimulálni, akkor tökéletes lehetőséget biztosít ez a

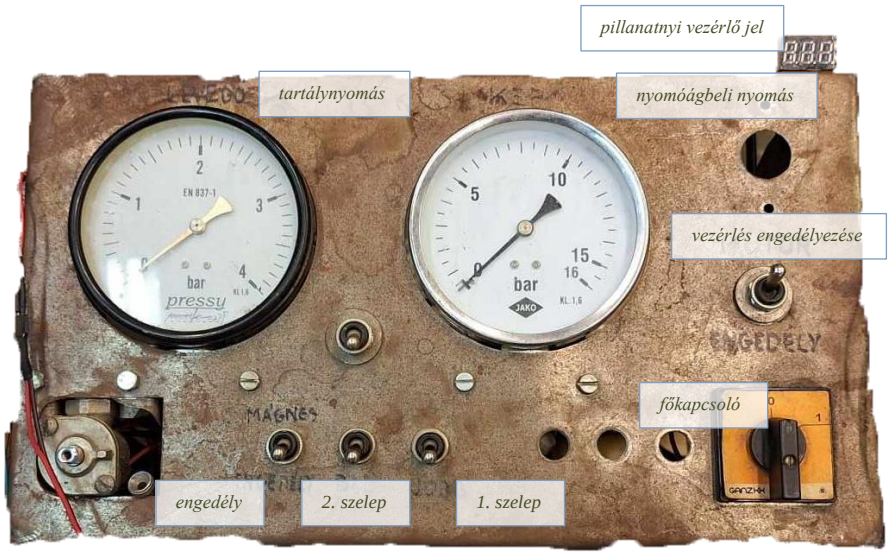
számunkra ahhoz, hogy akár több baros túlnyomást hozzunk létre a gyűjtőtartályban Ekkor a tartály alatti manuális elzáró szelepet és a 2. számú elektromos szelepet megnyitjuk, a dugattyús kompresszort üzembe helyezzük és a légnyomás segítségével a gyűjtőtartályban található repceolajat visszajuttatjuk az ejtőtartályba, ezzel zárva a kört.



14. ábra A berendezés elvi vázlata

4.1.2 A próbapad meglévő felépítése, kezelőszervei

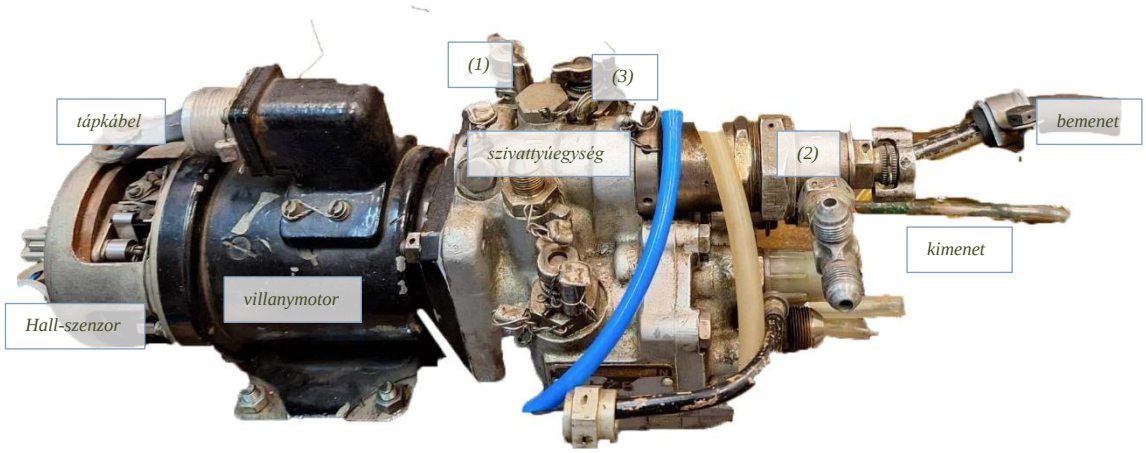
A 924-es próbapad fizikai kezelőfelületét a 15. ábra mutatja be részletesen. A bemeneti áramforrást fizikailag a főkapcsoló szakítja meg mielőtt még bármit is kezdnénk ezzel a feszültséggel. A berendezést a hálózati aljzatból szeretnénk működtetni, így az egy transzformátorra kerül, ahol 24 V-os egyenáramú feszültséggé alakítjuk át. Ezt az egyenáramot aztán alakítjuk tovább a vezérlésnek megfelelően és a szivattyúra pedig egy átalakított egyenáramot juttatunk, ezzel vezéreljük annak fordulatszámát. Ehhez tartozik még három darab hétszegmenses kijelzőből épített visszajelző kijelző, amely az éppen kiadott vezérlő jellel arányos számot mutat, azonban ezt kizárólag a saját munkánk ellenőrzésére fogjuk a mérés során használni. A vezérlést külön engedélyező kapcsolóval tudjuk elindítani. Az előbb írt fejezetben említettük az elektromos szelepek fontos szerepét, ezt a két szelepet, az 1. és a 2. számút összesen három darab kapcsolóval tudjuk működtetni. Szintén egy engedélyező kapcsoló szükséges ahhoz, hogy egy-egy szelepet működtessünk a próbapadon, enélkül nem fog áram jutni a szelepekhez bármi is legyen az állása a szelepkapcsolóknak. Ezen felül található még két darab nyomásmérő is a pulton, a bal oldali



15. ábra A próbapad kezelőfelülete

nyomásmérő a gyűjtőtartályban található túlnyomást méri, amelyet a 14. ábra alapján található kompresszor tud elsődlegesen növelni. A jobb oldali nyomásmérő pedig a nyomóoldali ágban lévő túlnyomást mutatja, erre azért van szükség, ha esetleg elmarad az 1. számú szelep nyitása, akkor ez a nyomásmérő direkt módon mutatja a számunkra, hogy miért nem jut olaj a gyűjtőtartályba.

A 16. ábra az éppen bekötött szivattyút mutatja. A 924-es szivattyú számos ki- és bemenete közül mi összesen csak a villanymotor bemeneteit, illetve a tüzelőanyag szivattyúágát fogjuk használni. A szivattyún ezen felül található még több beállítócsavar is, ezekkel a felugrási nyomást (1), a végnyomást (2), illetve a kezdeti nyomást (3) lehet beállítani. Ezeket a megfelelően állandó körülmények elérése érdekében nem fogjuk változtatni. A bemeneti cső közvetlenül a zárócsaptól érkezik, a kimeneti csővezeték pedig egyenesen az 1. számú zárószelepen áthaladva távozik a gyűjtőtartályba, ahol a porlasztási tulajdonságok jövőbeli vizsgálata miatt egy porlasztón keresztül jut a tartályba. A szivattyú fogaskerekes kivitelű, emiatt mindig megfelelő tüzelőanyag, vagy jelen esetben olajellátással kell rendelkeznie annak érdekében, hogy folyamatos kenést biztosítsunk annak, szárazon nem futtathatjuk, különösen kell ügyelni a szívóoldali csapok megnyitására.

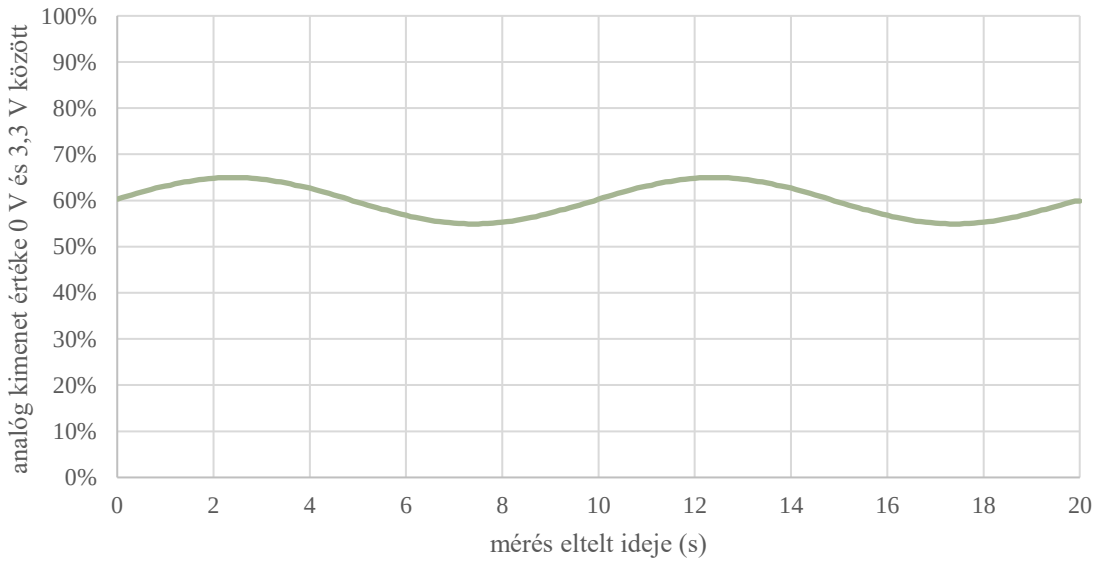


16. ábra A 924-es szivattyú és annak villanymotorja

4.1.3 Méréshez használt kimenő jel

Mivel a mikrokontroller sokrétűen alkalmazható a mérés során, így azt gondoltuk, hogy a lehető legtöbb feladatot bíznánk rá, legyen szó vezérlésről vagy adatgyűjtésről. A legújabb fejlesztés során a próbapad már rendelkezik egy olyan, az elektromos motort vezérlő tranzisztoros vezérlővel, amely egy potenciométer állása alapján képes egy adott kimenetet adni a motornak. Fontos megemlítenünk, hogy ez nem egy visszacsatolt szabályzó, hanem egy nyílt hurkú vezérlő, amely analóg bemenet alapján küld a motornak egy adott feszültségű jelet. Ezt az analóg jelet könnyűszerrel helyettesíteni tudjuk egy, a kontrollerből kijövő analóg jellel. Az mbed LPC 1768 p18-as kimenetén lehet egyedül ilyet véghez vinni, ekkor is $0\ V$ valamint $3{,}3\ V$ közötti kimenetek választhatóak a belső limitációk miatt. A kontroller ezen kimenetének felbontása megegyezik az előjel nélküli 16 bites egész szám típusú belső változó (unsigned 16-bit integer) maximális értékével, amely $2^{16} = 65536$, azaz arányosan elosztva 0 és $65\,535$ közötti szám segítségével megfelelően nagy felbontással generálhatunk kimenetként szinusz függvény szerint változó analóg jelet. Az adatok begyűjtésére egyelőre terminál alkalmazást használunk (TeraTerm), amire a kontroller csak kiküldi az adatokat, ezeket pedig mi feldolgozzuk. Erre a függvényre egy példát a 17. ábra, illetve egy példakódot pedig az 3. melléklet tartalmaz. A kimenetre küldött szinuszjel értékét a 6. egyenlet szerint számíthatjuk, ahol az *offset* értékét mi állítjuk majd be.

$$AnalogOut(rads) = amplitude \cdot \sin(frequency \cdot rads) + offset \qquad \textit{6. egyenlet}$$



17. ábra Az első szinuszhullám a ciklus során (0,1 Hz – 20 s, 60%-os középértékkel). A diagram valós, kontroller által számítógépre visszaküldött adatok alapján lett elkészítve a 6. egyenlet alapján.

Azonban nem tudunk identifikációt végrehajtani akkor, ha csak egyetlen frekvencián vizsgálódunk. A frekvenciatartománybeli analízis szerint a rendszer erősítése egy egységnyi amplitúdójú, egyre növekvő frekvenciájú szinuszjellel történő gerjesztésre adott válaszjel lesz [14], adódik tehát az összefüggés, hogy nekünk is lehetőleg nagy tartományban készítsünk egymás után egyre növekvő frekvenciájú gerjesztő jelet. Ezt úgy tudjuk a legegyszerűbben leképezni, hogy veszünk tíz

diszkrét frekvenciát, amelyen megfelelő ideig gerjesztjük a rendszert, ekkor elég adatunk keletkezhet. Ezt a tíz diszkrét értéket az 1. táblázat tartalmazza. Ekkor a villanymotor ezt csak egy ideig tudja majd lekövetni a fordulatszámával és egy bizonyos frekvencia felett azt várjuk el tőle, hogy beálljon egy fordulatszámra, amely a szinuszjel függőleges eltolásából adódik majd.

1. táblázat A méréshez használt egyre növekvő frekvenciájú szinuszhullámok adatai

N	Frekvencia (Hz)	Ciklusszám (-)	Időtartam (s)	N	Frekvencia (Hz)	Ciklusszám (-)	Időtartam (s)
1	0,1	2	20	6	2	10	5
2	0,25	2	8	7	3	12	4
3	0,5	4	8	8	5	25	5
4	0,75	6	8	9	7,5	30	4
5	1	5	5	10	10	50	5
						Összesen	72

4.1.4 UART kommunikáció

Az UART, vagyis univerzális aszinkron adóvevő (*universal asynchronous receiver-transmitter*) egy kommunikációs forma, egy kommunikációs eszköz adatok átvitelére. A kommunikáció jelen esetben USB-n (*universal serial bus*) keresztül fog történni, hiszen alapvetően erről kapja az mbed is a szükséges tápfeszültséget, ekkor nincs is szükség a két eszköz közötti esetleges feszültségkülönbséggel való bánásra sem. Az aszinkron kommunikáció lényegi része, hogy az adattovábbítás csak akkor történik, ha van továbbítandó adat, azaz, ha van például egy karakter, mondjuk ’c’ az író pufferben [7]. Két karakter között egy folyamatos „*break signal*” kerül továbbításra, amely általában egy magas jel, ezzel tudja a két eszköz azonosítani, hogy mikor indul a valós adat továbbítása. Digitális kommunikációról van szó, azaz az adatok továbbítása bináris formátumban történik, azaz, ha az eredeti példánál maradunk, a ’c’ karakter bináris 8-bites ASCII kódja nem más, mint a 01100011_2 [19].

Mivel aszinkron kommunikációról van szó, így minden egyes adatcsomag magával kell cipelje a szükséges szinkronizáló információit, ez lesz a kerete az adatnak, ezt nevezik *framing*-nek. Ez a *framing* három adatból áll, egy startjelből, egy paritásjelből és egy stopjelből. A startjel általánosan 0 szokott lenni. A paritásjel egy olyan bit, ami lehet úgynevezett páros vagy páratlan paritás, ami azt a célt szolgálja, hogy a küldött bitek számában hány darab 1-es található. Ez a nyolc biten felül történik továbbításra. A páros paritásbit 0, ha páratlan számú 1-es található a továbbított adatcsomagban, 1, ha páros számú 1-es található az adatcsomagban. A páratlan paritásbit pedig éppen ennek az ellenkezőjeként működik. Azt a célt szolgálja, hogy belső ellenőrzésképp meg lehessen határozni, hogy a fogadott adat megfelelő minőségben került-e továbbításra, ha a paritásbit eltér a fogadott jelben

18. ábra UART kommunikáció által használt kódolás

számolt bitektől, akkor valami hiba történhetett. A stop jel pedig nem más, mint az adatcsomag végét jelző jel, ez minden esetben 1 értékű kell legyen, általában egy, vagy két darab 1 értékű jel is használatos stop jelként. Amennyiben az adat kerete nem megfelelő, azaz a start vagy stopjelek nem kerülnek érzékelésre megfelelően, *framing error* lép fel a két kommunikáló eszköz között, és ezt jelezni is fogja [7]. Az előbb bevezetett 'c' karakter továbbításához használt kód tehát nem más, mint amit a 18. ábra is mutat nekünk. Ezt fogja majd az mbed beépített USB kommunikációs modulja UART kommunikációvá átalakítani.

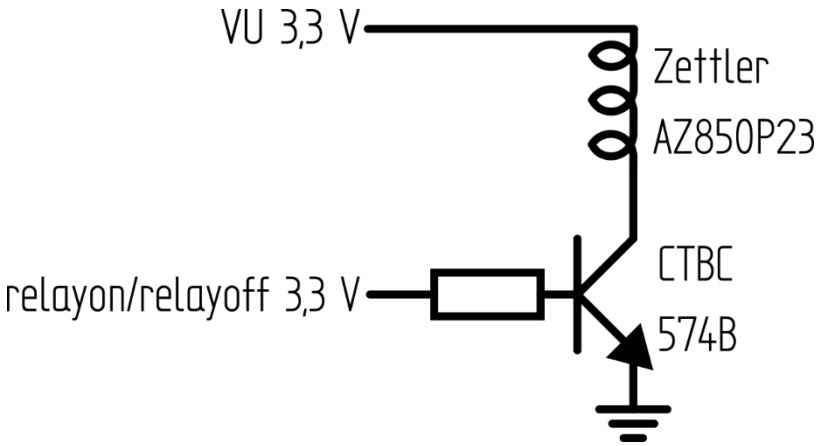
Ezen felül még egy fontos tulajdonságot érdemes megemlítenünk, az úgynevezett *baud rate*-et, azaz a másodpercenként továbbított bitek számát. Mivel 11 bitből áll egy karakter, így ezzel kell tovább számolnunk. Azért fontos számunkra ez a tulajdonság, mert ekkor megfelelő időben tudjuk a mérendő paramétereket a számítógép képernyőjére küldeni és megfelelően kicsi időközönként fog történni a mintavétel, pontosabban annak továbbítása. Ezek előre definiált sebességeket használnak, például egy úgynevezett teleprinter, azaz karakterek vezetékkel történő továbbítására használt eszköz standard *baud rate*-je 110 volt, amely a 11 bit miatt 10 karaktert jelent másodpercenként. Ez az 1970-es évekre jellemző mennyiség, ennél manapság jóval gyorsabb átvitelre is képesek vagyunk, jelenleg az mbed felületén elérhető standard *baud* 9600, azaz több mint 800 karakter másodpercenként. Mi azonban ennél sokkal nagyobb rátát fogunk használni, 115 200-at, amit azért fontos megemlíteni, mert minden, ezzel az eszközzel használt és kapcsolatban lévő program vagy más eszköz is ezen a rátán fog adatot várni, illetve adatot küldeni, ha pedig ezek nem egyeznek, problémába fogunk ütközni.

Ezen felül a túl kicsi *baud* is problémát okozhat, a 19. ábra éppen ezt mutatja be. Oszcilloszkópon vizualizáltuk az analóg kimeneten kiadott szinusz jel valós idejű függvényét, és meglepődve tapasztaltuk, hogy a magasabb frekvenciákon problémát okozhat, ha túl lassan küldünk adatot a vele tulajdonképpen semmilyen hardveres összefüggést nem mutató UART-on. A kapcsolat pusztán szoftveres volt, 10 Hz-es szinusz frekvencia esetén a lassú adatküldés túlságosan lefoglalja a kontroller erőforrásait, így az általunk 65536-os felbontással közelített jelet nem tudjuk megfelelően leképezni, csak rendszertelen jel marad belőle, amely alkalmas lehet közelítésre, de mivel van lehetőségünk ennél pontosabb közelítésre, így ezt nem fogjuk elfogadni és emiatt 115 200-as *baud*-ot fogunk használni.

19. ábra Kiadott analóg 10 Hz-es szinusz jel valós időbeli értéke alacsony, 9600 -as (bal) és magas, 115 200-as (jobb) baud értékek esetén egy oszcilloszkópon mérve.

4.1.5 Parancskódok

A legrugalmasabban úgy tudjuk kezelni a bemenő paramétereket, ha magunk állítjuk be és nem fixen kerülnek a mikrokontroller memóriájába beégetésre. Mivel az újragondolt vezérlés egy analóg bemenet, illetve egy kapcsoló alapján működik, vészelállításként magunk tudjuk be- és kikapcsolni a kimenetet. Egy bistabil relét kötve a mikrokontroller két digitális kimenetére, a p7, p8 kimenetekre és azokat parancsokkal vezérelve elérhetjük, hogy magunk tudjuk engedélyezni az egész vezérlési kimenetet. A relé egy Zettler AZ850P2-3 típusú relé, amihez sajnos nem volt elég a 3,3 V vezérlő feszültségen a kontroller kimenet által kiadott vezérlőáram, így a *set* és *reset* bemeneteket egy-egy tranzisztorral kell megtámogatnunk, két CTBC 547B típussal, ezt a 20. ábra mutatja be. Azonban ahhoz, hogy ne kézzel kelljen ezeket állítanunk, felmerült az ötlet, hogy magáról a számítógép termináljáról lehessen mindezeket vezérelni, így megszülettek a parancskódok, amelyekkel a mérés, illetve a majdani szabályozás is sokkal könnyedebben folyhat.



20. ábra A relé kapcsolásához használt erősítő kapcsolás

Úgynevezett parancskódokat definiáltunk, amiket a terminálba beírva tetszőleges eredményt érhettünk el a kontrollerünkön. A parancskódok listáját a 2. táblázat tartalmazza. Ehhez viszont a kapcsolatot át kellett alakítanunk. A programban kezelt változók listájából két típusú soros kapcsolat értelmezett, az egyik az úgynevezett *Serial* a másik pedig a *RawSerial* változóval értelmezett kapcsolat. Alapvető különbség a kettő között, hogy az UART kommunikáció során, hogy a *RawSerial* kapcsolat a

beépített *printf()* és *getc()* parancsokat módosítottan implementálja, így azok megfelelően lesznek megszakításkori (*interrupt*) bemenetek értelmezésére [20]. Ezek a megszakítások a megírt folyamatos kódot nem, vagy csak elhanyagolható mértékben befolyásolják, így nem fognak gondot okozni a valós idejű visszacsatolt vezérlés kiépítése során, azonban lehetőséget adnak arra, hogy ne a fő ciklusokon belül vizsgáljuk folyamatosan a bejövő adatokat, hanem csak akkor, ha ténylegesen van mit vizsgálnunk, ezzel meggyorsítva a programunkat. Erre egy kidolgozott mintaprogramot a dolgozat végén a 2. melléklet tartalmaz, amely az ’1’-es billentyű megnyomásakor az mbedbe beépített LED1-es digitális kimenet értékét állítja át.

2. táblázat A mikrokontroller által értelmezett parancskódok

/r	A szabályozó engedélyezéséhez szükséges relé átkapcsolása
/mr	Fordulatszám mérési eredményének kiírása (igen/nem)
/mf	Ejtőtartályban maradt folyadék tömegének mérési eredményének kiírása
/sa#/	Nyílt hurkú vezérlő konstans értékre való beállítása
/sr#/	Fordulatszám-visszacsatolt szabályozó referencia értének beállítása
/ss#/	Szinuszgerjesztés amplitúdójának beállítása
/ca	Nyílt hurkú vezérlési mód
/cr	Fordulatszám-visszacsatolt vezérlési mód
/d	Debug mód be- és kikapcsolása
/gs	Állandó frekvenciájú szinuszgörbe kiadása a vezérlő kimeneten
/gi	Egyre növekvő frekvenciájú szinuszgörbe kiadása, a ciklus 72 s-ig tart
/gr	Szinuszgörbe megállítása, visszatérés kézi folyamatos szabályozáshoz
/h	Segítség, az elérhető parancsok listája

Ekkor, ha bármilyen hiba történne, vagy elszállna a vezérlő algoritmus, két biztonsági funkciónk is lesz arra, hogy kézzel bármikor beavatkozzunk ennek megállítására, a parancsok közé beépített */r* parancsot puha, míg a kezelőfelületen található, áramellátó kapcsolót a kemény leállításnak fogjuk bélyegezni.

A */mr* paranccsal elindíthatjuk, vagy megállíthatjuk tetszőlegesen a program által belsőleg folyamatosan mért fordulatszám értékek kijelzését a számítógép képernyőjére. Erre a tesztelés során azért volt szükségünk, hogy nem pazaroljunk feleslegesen erőforrásokat arra az esetre, amikor nincsen erre szükségünk, csak amikor kíváncsiak vagyunk erre külön. A mérés során azért sem lesz erre szükségünk, mivel a növekvő frekvenciájú szinuszgörbe kiadásakor beprogramozhatjuk úgy az eszközt, hogy folyamatosan továbbítsa a fordulatszám mért értékeit is a számítógép felé, így gyakorlatilag minden identifikációhoz szükséges paraméter a kezünkben lesz, különösen, ha a terminál programon az úgynevezett *log* funkciót is igénybe vesszük, ami voltaképpen egy szöveges fájlba menti el az összes terminál által kezelt karaktert.

Az analóg kimeneten szabályozási esetben egy referencia bemenet alapján szeretnénk szabályozott, visszacsatolás utáni kimenetet láttatni, ezt a referenciát a */sa#/* paranccsal tudjuk kezelni. A szabályozás megtervezése előtt ezt a bemenetet arra fogjuk felhasználni, hogy konstans vezérlő jellel lássuk el a villanymotort. Ekkor a majd elindított szinuszgörbe az általunk indított jelről tud elindulni, azaz a szinuszgörbe nullátmenete ebben a pontban lesz, ez lesz az alapjel. Ezen kívül egyedi beállításokra is lesz lehetőségünk, a 6. egyenlet alapján értelmezett *amplitude* változó értékét a */ss#/* parancs segítségével állíthatjuk be, így könnyedén rekonfigurálhatjuk a gerjesztő paramétereket anélkül, hogy a program újrakompilálásával vennénk el az időnket. A majdani zárt hurkú szabályozó referencia értékét a */sr#/* paranccsal írhatjuk be, amely esetén egy fordulatszámot adunk a követő szabályozás tartandó értékeként.

A szabályozás aktuális módjának kiválasztására két parancsot is létrehoztunk, a */ca* és */cr* kódokat. Az előbbi az identfikációnál is nagyon fontos nyílt hurkú vezérlést valósítja meg egy konstans analóg jel kiadásával, az utóbbi pedig a valódi szabályozó lesz. Ahhoz, hogy problémamentesen zajlódjék le az átállás, két *bool* típusú változót deklarálunk, amelyek igaz értéke esetén módosítjuk a mindenkori szabályzási módnak megfelelően a kiadott analóg jelet. Ehhez a főciklusban *if* blokkok közé illesztjük a szabályozó algoritmust.

A tényleges identifikációhoz a */gi* fog nagyon fontos szerepet játszani, ő fogja az 1. táblázat értelmében létrehozott szinuszjelek generálását végrehajtani, ez előtt érdemes beállítanunk az alapjelet, valamint az amplitúdót. Ennek a demonstrációjára született az állandó frekvenciájú folyamatos szinuszjel generálása is, amely /*gs* beírásával indítható el. Ha valami hiba csúszik be a mérés során, a */gr* paranccsal leállíthatjuk a szinuszjelek kiadását és visszatérhetünk konstans vezérlésre, legvégső esetben is elérhető lesz a */r*, vagy nemes egyszerűséggel a főkapcsoló.

4.1.6 Debug-mód

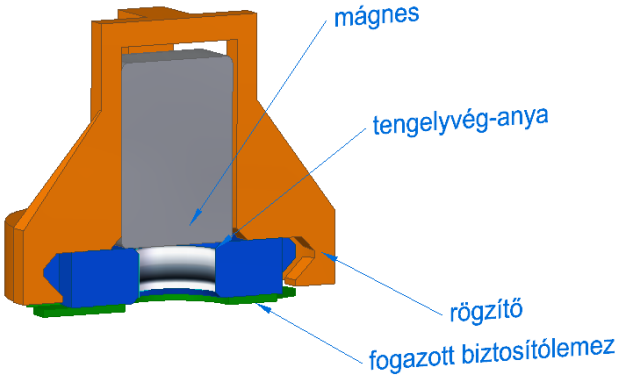
A */d* paranccsal aktiválható a korábban említett debug-mód, amely egyfajta ellenőrzőként vesz részt a folyamatban, feltéve, ha egyáltalán szükségünk van ilyenre. Ekkor például telefon kijelzőjén valós időben olvashatunk minden paramétert, általunk választott szűrők nélkül, hiszen a funkció fontos jellemzője, hogy minden adat rendelkezésre álljon ahhoz, hogy a hibákat kiküszöböljük a programból. Ezt egy RN41 típusú Bluetooth-modullal tudjuk megvalósítani, amely a már említett UART soros kapcsolaton keresztül kommunikál a mikrokontrollerrel.

Ez egy külső kis eszköz 6 lábbal, amelyek közül mi négyet, a táp és föld, valamint a TX és RX lábat fogjuk használni. Tápfeszültségként az eszköz $3{,}0\,V$ valamint $3{,}6\,V$ közötti értékek esetén működik megfelelően, tehát tökéletes a számunkra, tudjuk használni az mbed standard $3{,}3\,V$-os egyik tápfeszültség kimenetéről. A kontroller árammal való ellátása esetén azonnal működésbe lép és máris csatlakozhatunk rá elsősorban okostelefon segítségével. Az eszköz kimeneteit, valamint az általunk akart bemeneteket bármely ilyen célra fejlesztett alkalmazáson keresztül el tudjuk érni, ekkor akár

számítógép segítsége nélkül is képesek vagyunk megfigyelni, mi történik a vezérlés vagy éppen egy adott mérési ciklus során. Az eszköz Bluetooth 2.1-es szabványt használ, így már nem kapható. A modul további fontos előnye, hogy a modultól függő megfelelő kód nélkül nem lehetséges arra csatlakozni, így kerülve el azt, hogy idegen módon férjenek hozzá a vezérlőhöz [21].

4.1.7 A fordulatszám mérése

A fordulatszám mérése mágneses úton történik. A 924-es szivattyút meghajtó villanymotor tengelyének anyával ellátott részére egy mágnest rögzítünk. Ez a mágnes kellően erős ahhoz, hogy könnyen érzékelhető változást keltsen a mágneses térben, amelyet érzékelni lehet. A mágnes azonban sajnos önmagában nem maradt a villanymotor tengelyének a végén, a mágnes által keltett erő nem volt ehhez elég erős, nagyjából 1500 és 2000 $\frac{1}{min}$-es fordulatszám tartományban az váratlanul lerepült a tengely végéről az előző méréseknél, így más módszerhez kellett folyamodnunk. Mivel a Vasúti Járművek és Járműrendszeranalízis Tanszéken található egy 3D-nyomtató, így adódott a lehetőség, hogy tervezzünk ide egy ABS-ből egyedileg legyártható alkatrészt, amely a mágnest a helyén tartja és bepattanó kötéssel rögzíti a tengely végén. Az alkatrész a 21. ábra modelljén látható. Axiálisan a mágnest a lerepülés ellen az apró körmök, valamint a mágnes biztosítják, ezek a tengelyvéganya letöréssel rendelkező geometriájába illeszkednek, ekkor pontosan a forgástengelybe illeszthető a mágnes. Radiálisan a fogazott biztosítólemez egy felhajtott foga fogja biztosítani, hogy a mágnes ne változtassa a szögét, hiszen ekkor nem lenne pontos a mért fordulatszám értékünk.



21. ábra A mágnes rögzítése a tengely végén

A mágnes által változtatott mágneses tér érzékelhető egy Hall-cellával, hasonlóan a 3.1 fejezhez, a cella által adott digitális négyszögjelet a mikrokontrollerbe vezetjük és ott mérjük annak frekvenciáját. A különbséget itt a mérésben az fogja adni, hogy módosítjuk aszerint a programkódot, hogy két felfutó él közötti időtartamot mérjen, hiszen nem tudjuk mindig pontosan meghatározni, hogy az adott PWM-szerű jel kitöltési tényezőjének is felfogható aktív periódusának aránya a teljes jelhez képest biztosan mindig 50% lesz-e. Így a fordulatszám-mérés leredukálódik egy egyszerű időmérésre, amelyet a mikrokontroller *interruptként*, azaz megszakításként könnyedén tud meghatározni. Azaz a kontroller folyamatosan mér időt, amennyiben van bejövő jel, úgy a kontroller tárolja az adott időpontot egy

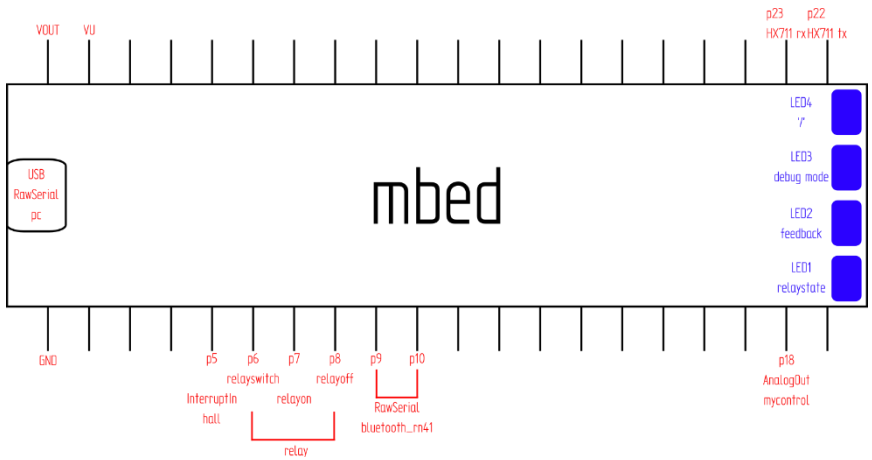
változóba, majd amint újabb magas jel érkezik egy másik változóban tárolja azt, így a kettőt egymásból kivonva μs nagyságrenddel tudjuk meghatározni a fordulatszámot. Az ideális eset az lenne, ha nem csak egy körülfordulás után érkezne jel, hanem ez idő alatt minél többször, azonban egyelőre az identifikációhoz ez megfelelő lesz.

Mivel ekkor a program, ha nem kap újabb jelet a Hall-szenzortól, nem indít újabb fordulatszám-meghatározási ciklust, ez azt eredményezi, hogy ameddig nem forog a villanymotor, folyamatosan egy adott legutolsó alkalommal meghatározott értéken marad az. Ezt egyszerűen ki tudjuk küszöbölni azzal, ha beépítünk egy *Timeout* típusú rutint is, amely meghatározott időtartamokban zérusra állítja be a fordulatszám értékét. Ezt a rutint minden alkalommal újraindítjuk, amikor fordulatszám jelet kapunk, ezzel biztosítva, hogy szabályozás vagy mérés során ne legyen képes elindulni, ezzel nem fogja befolyásolni a kapott adatok minőségét. Ezt az időtartamot $1{,}5\ s$-ban állapítottuk meg, ennyi idő bőségesen elegendő ahhoz, hogy még nagyon alacsony fordulatszám értékeken is megfelelő mérésünk legyen, azonban elég rövid ahhoz, hogy közel álljon a villanymotor valós periódusidejéhez.

4.1.8 Kapcsolás

A mikrokontrollert a különböző eszközökhöz és berendezésekhez egy dugaszolható próbapanelen tudjuk összekapcsolni, itt tudunk továbbá kezdetleges kapcsolásokat, teszteket is folytatni. Ez egy nagy műanyag panel, amibe forrasztás nélkül megfelelő kapcsolatot tudunk létesíteni két vezeték között. Ebben fogjuk az egész mérést lefolytatni a könnyebb átláthatóság és a könnyebb módosíthatóság érdekében, ebbe helyezzük be a kapcsoló relét is, amely a vezérlést majd pedig a szabályozást fogja engedélyezni. Ahhoz, hogy mégis jól lássuk, mi történik, valamint egyértelmű legyen, milyen szerepek jutnak a kontrollerre és a környezetére, elkészítettük a 22. ábra által mutatott kapcsolási rajzon, amelyen a belső változók nevei is szerepelnek.

Külön érdemes megemlítenünk, hogy nem csak soros terminálos leszünk képesek információt kinyerni a kis eszközből, hanem a 4 beépített LED-et igénybe vesszük. Lentről felfelé haladva 1-4-ig tartó számozással tudunk ezekre a LED-ekre hivatkozni a programunkon belül, mint *DigitalOut* típusú

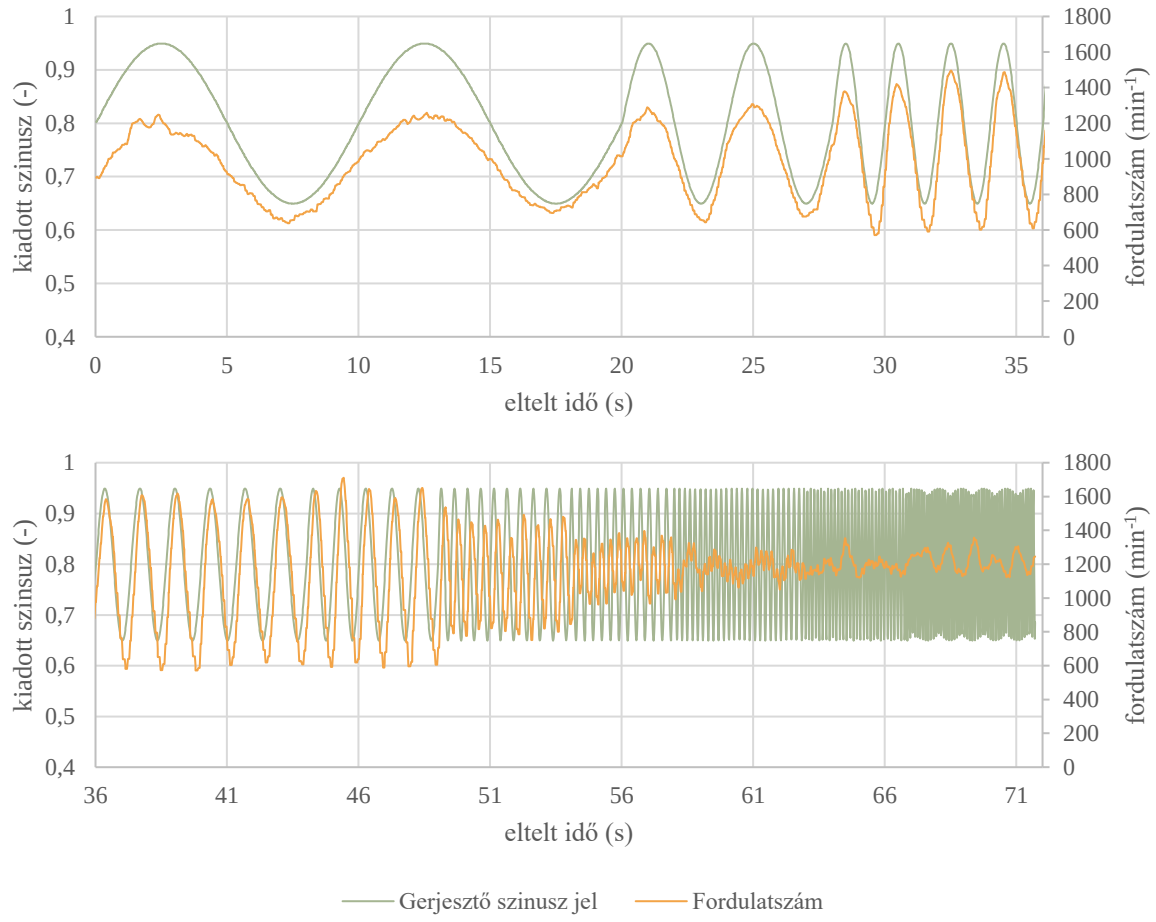


22. ábra Az mbed mikrokontroller lábainak bekötése a belsőleg használt változók neveivel együtt, valamint a négy darab, visszajelzésre szolgáló LED

kimenet. Az 1. számú LED fogja a tudtunkra adni a relé pillanatnyi állapotát, azaz, hogy éppen engedélyezve van-e a vezérlő berendezés, amennyiben világít, úgy a relé zárt állapotban van. A 2. számú fénykibocsájtó fogja azt jelezni, hogy éppen milyen típusú algoritmust futtat a kontroller: sötét állapotban a */ca* utáni visszacsatolás nélküli vezérlési algoritmus zajlik éppen, */cr* parancs után pedig a LED felgyullad, jelezvén, hogy éppen visszacsatolt szabályozóként működik a program. A 3. számú LED akkor lesz aktív állapotban, amikor éppen a Bluetooth-modulon keresztül adattovábbítás történik, azaz debug módban vagyunk éppen. Végül a 4. számú LED jelzi a parancssor állását, mindaddig, amíg kontroller bemenő parancsra vár a LED égni fog, ezzel jelezve, hogy nem fejeztük be a parancsok bevitelét és vár a *'/'* jel utáni kód beolvasására.

4.2 Végrehajtás

A mérést több kiindulási ponttal végeztük el, többféle alapjel és amplitúdó értékek megadásával, miközben figyeltünk arra, hogy a lehető legjobban kezelhető adatsorokat kapjuk. Minden mérés után a lehető legegyenlőbb körülmények biztosítása érdekében a kompresszor bekapcsolásával visszajuttattuk a közeget az ejtőtartályba. Összesen három olyan mérést végeztünk, amely értékeit összehasonlítottuk, ebből csak a harmadik mérés eredményeit mutatjuk be a 23. ábra segítségével, a többi diagramot a 4. melléklet tartalmazza.



23. ábra 80%-os alapjel esetén 15%-os amplitúdóval történő mérés. A fordulatszám értékek 3 pontos mozgóátlaggal lettek az identifikáció során beszámítva.

Ezen alkalommal megfigyelhető mindaz, amit az identifikáció ezen lépése során vártunk. A szivattyú mért fordulatszáma pontosan leköveti jó darabig a kiadott szinuszjelet, amint az ábrán megfigyelhető, még egészen a mérés 59-ik másodpercéig képes a szivattyú majdhogynem azonnal reagálni a bemenet változására, miután pedig a szivattyú már nem képes lekövetni az egyre növekvő frekvenciával változó jelet, egy lehetőleg konstans fordulatszámra áll be, ami az ábrán a zavarások miatt nem látható első ránézése, azonban a kezdeti lengés alapján kikövetkeztethető, hogy nem tud már azonos mértékben változni a mért érték. Mivel nagyon ingadozó mért fordulatszám értékeket kaptunk, így ahhoz, hogy az identifikáció során megfelelő minőségben tudjuk a rendszer átviteli függvényét meghatározni, a nyers mért fordulatszám adatokat 3 pontos mozgóátlaggal fogjuk kezelni a 7. egyenlet értelmében.

$$RPM_{mean} = \frac{\sum_{i=1}^{n} RPM_i}{n} \qquad \textit{7. egyenlet}$$

Fontos továbbá kiemelnünk, hogy mind a mellékletben található diagramokon, mind az előző ábrán látható fordulatszám értékeken látszik a mérés vége felé egy enyhe növekedés, azonban ez csak az utolsó 15-20 másodpercben volt ez árulkodó. Ez talán annak az eredménye, hogy az üzem során a szivattyú enyhén felmelegedett, mivel nem rendelkeztünk előfűtőkkel ahhoz, hogy a tüzelőanyagot előmelegítsük, így az kis részben a szivattyúban melegedett fel, ekkor azonban a viszkozitása csökkent. A csökkent viszkozitás miatt viszont a szivattyúnak sokkal könnyebb dolga akadt a szállítással, így nagyon enyhén képes volt felgyorsulni. Ezt a hatást azonban nem fogjuk tudni figyelembe venni, mivel nem mértük a tüzelőanyag hőmérsékletét, azonban mivel nem is jelentős növekedésről van szó, így elhanyagolhatjuk.

A mérés során nagyon fontos volt a megfelelő mintavételezési idő megválasztása és annak minél pontosabb értéken tartása. A *System identification toolbox* vagy csak úgy képes elvégezni az identifikációs lépéseket, ha a bemeneti változók bizonyítottan egy adott állandó mintavételezési értékkel lettek kimérve, vagy ha tartozik minden egyes értékhez egy saját mintavételezési idő. Mivel az utóbbi módszer nem bizonyult ígéretesnek, azaz már az *Output-error*-módszer (MATLAB-ban *oe()* függvény) sem volt képes lekövetni és felhasználni az időbélyegzővel ellátott mérési adatokat, így arra jutottunk, igyekezzünk biztosítani, hogy a mérés során az egymást követő mintavételek közötti időtartamok eltérései az $5\,ms$-os eltérést ne haladják meg, ekkor úgy gondoljuk, ez tekinthető még egyenközű mintavételezésnek. Mivel a $20\,ms$ túl hosszú lenne, a $10\,ms$-ot pedig a kontroller nem volt képes pontosan tartani, így az egységes mintavételezési idő a $12\,ms$-os időtartam volt, amit sikerült is elérnünk különféle programbéli belső késleltetésekkel vagy egyszerűsítésekkel.

4.3 Eredmények

A nyers mérési eredmények után ideje volt valós körülmények között is megfigyelnünk, hogyan illeszkednek a mért adatok, valamint szabályozót is tervezzünk erre a már identifikált függvényre.

4.3.1 MATLAB identifikáció

A MATLAB programban készítettünk egy mintaprogramot, amellyel az összes lemért értéket a 2.4 fejezet alapján fogjuk felhasználni, a program az 5. melléklet részleteiben olvasható. Importálás után, amennyiben specifikáltuk a mintavételezési időtartamot ($12\ ms$) már neki is állhattunk a különféle módszerek alapján történő rendszeridentifikációnak. Négyféle identifikációs eljárást is kipróbáltunk, mivel nem tudhattuk, melyik módszer fogja a legjobb eredményt szolgáltatni. A négy leggyakrabban használt eljárás az *output-error, ARX (autoregressive with exogenous input), ARMAX (autoregressive moving average with extra input)*, és Box—Jenkins-módszerek, így ezekkel dolgoztunk tovább. A 2. egyenlet szerinti polinomok értelmezése alapján az egyes módszerek a következő egyenleteket alkalmazzák. A módszerek bemeneti paraméterek száma, azaz az egyes polinomok általunk megadott tetszőleges fokszámok darabszáma ebben az esetben pedig rendre 3, 3, 4 és 5.

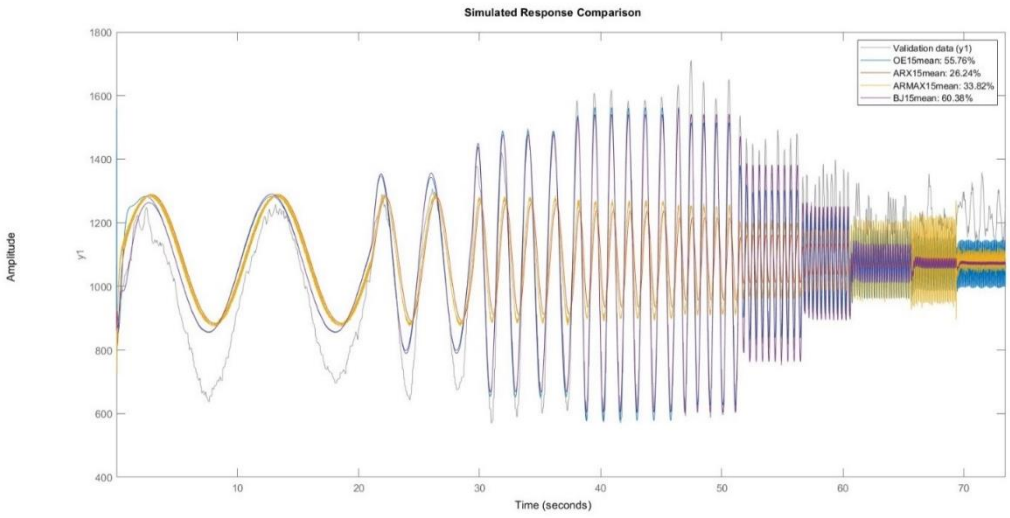
$$\text{Output-error:}\quad y(t) = \frac{B(q)}{F(q)}u(t-n_k) + e(t) \qquad \textit{8. egyenlet}$$

$$\text{ARX:}\quad A(q)y(t) = B(q)u(t-n_k) + e(t) \qquad \textit{9. egyenlet}$$

$$\text{ARMAX:}\quad A(q)y(t) = B(q)u(t-n_k) + C(q)e(t) \qquad \textit{10. egyenlet}$$

$$\text{Box—Jenkins:}\quad y(t) = \frac{B(q)}{F(q)}u(t-n_k) + \frac{C(q)}{D(q)}e(t) \qquad \textit{11. egyenlet}$$

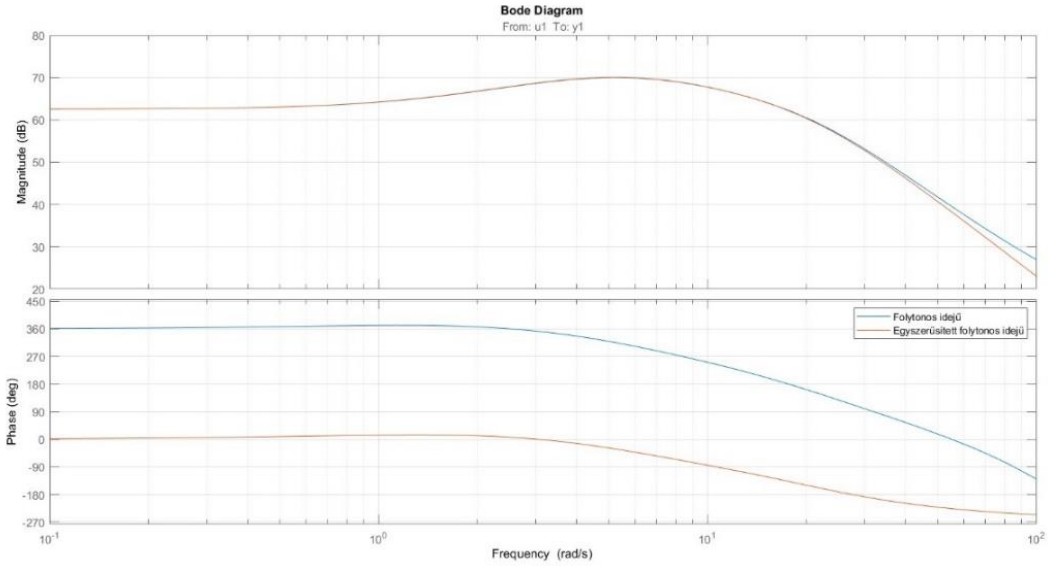
Ezen bemenő paramétereket első körben tetszőlegesen választottuk meg, ekkor minden esetben egységvektort adtunk a függvények számára. Ezt az eredménytől függően utána kézzel módosítottuk az alapján, hogy a lehető legjobb illeszkedést érjünk el a megjelenített összehasonlító ábrán. Több körben végeztettünk számításokat a programmal, a lehető legtöbb variációt kipróbálva, így alkottuk meg a 24. ábra által bemutatott teljes diagramot, amin az összes eddig részletezett módszer alapján megpróbáltuk



24. ábra A 80%-os alapjel és 15%-os amplitúdó, valamint 3 pontos mozgóátlaggal kezelt fordulatszám közös értékeinek identifikációja utáni eredeti függvénnyel történő összehasonlítás MATLAB programban. A megnevezések és az identifikációs eljárások eredeti értékekkel való illeszkedési arányai a jobb felső sarokban találhatók.

meghatározni az átviteli függvényt. Változó eredményeink születtek, az összes mérés minden teljes identifikációs diagramját a 6. melléklet tartalmazza. Az, hogy mi számít megfelelő illeszkedésnek, változó módon lehet meghatározni, mi a MATLAB beépített vizsgálója által mutatott értékek közül úgy határoztunk önkényesen, hogy a megfelelőséget az 50%-os illeszkedési küszöbhöz tesszük. Amennyiben a modellt nagyon sok próbálkozás után sem sikerül efölé vinni, úgy azt elvetjük a későbbiekben. Ezen a mérés során két illesztés is elérte ezt a megfelelőségi határt, az Output-error és a Box—Jenkins, ezek rendre $55{,}76\%$ és $60{,}38\%$ voltak. Sajnos az ARX és az ARMAX-modellek nem voltak képesek lekövetni a bemeneti paraméterek gyors változásait és nem sikerült ezekkel megfelelő minőséget elérnünk. Itt fontos azt is megjegyeznünk, hogy egy határon túl az identifikációs polinomok fokszámának növelése nem jár lényeges eredménnyel, nem tudunk velük szignifikáns pontosságnövekedést elérni, emiatt is maradtunk a Box—Jenkins-módszer során a $[2\ 1\ 2\ 4\ 5]$ bemeneti vektornál, amely rendre a $B(q)$, $C(q)$, $D(q)$ és $F(q)$ polinomok fokszámait, valamint az n_k értékét tárolja, még így is az $F(q)$ polinomunk negyedrendű lett, amely a későbbi számítások során lesz nehezítő körülmény.

Megkaptuk a szivattyú diszkrét idejű átviteli függvényét az identifikációs eljárással, ezt a 12. egyenlet mutatja be nekünk, azonban ez nem lesz érvényes folytonos idejű szabályozásra. Annak ellenére, hogy a mikrokontrollerrel történő szabályozás diszkrét idejű lesz, nekünk a PID behangolásához folytonos idejű átviteli függvénnyel érdemes dolgoznunk, mivel ekkor később magunk választhatjuk meg tetszőlegesen a mintavételezési időt, valamint a későbbiekben használt egyenletek is ezt kívánják meg. Emiatt a $G_d(z)$ függvényt úgynevezett Z-transzformációval (bilineáris transzformációval) [13] átalakítjuk, ezt MATLAB-ban egy lépésben megtehetjük a *d2c()* függvénnyel és így kapjuk a folytonos idejű rendszer $G(s)$ átviteli függvényét. A $G(s)$ esetében az exponenciális tényező az átviteli függvényben a bemenet késleltetéséből adódik, ezt a 13. egyenlet mutatja.



25. ábra A $G(s)$ egyszerűsítés előtti és a $G_e(s)$ egyszerűsítés utáni átviteli függvények Bode-diagramjai.

$$G_d(z) = \frac{19{,}19z^{-1} - 18{,}87z^{-2}}{1 - 3{,}506z^{-1} + 4{,}622z^{-2} - 2{,}718z^{-3} + 0{,}602z^{-4}} \quad \textit{12. egyenlet}$$

$$G(s) = e^{-0{,}048s} \cdot \frac{-3{,}058s^4 - 514s^3 + 84\,220s^2 + 1{,}428 \cdot 10^7 s + 1{,}996 \cdot 10^7}{s^4 + 42{,}42s^3 + 821{,}8s^2 + 5609s + 14\,880} \quad \textit{13. egyenlet}$$

Láthattuk a 23. ábra és a 24. ábra diagramjain, hogy a fordulatszám mérés közel sem lett annyira egyenletes, mint azt vártuk volna, ez elsősorban a mérési zajok miatt adódott. Mivel digitális eszközökkel mértünk, így a kontroller a digitális bemenetén érzékelhetett olyan változásokat, olyan nagyobb feszültségingadozásokat, amelyek torzították a fordulatszám mérésünket, és ekkor nem megfelelő eredményt kaptunk. Ez azonban az identifikált átviteli függvényben is megjelenik, a számláló polinom, amely a rendszer kimeneti hatásait magába foglalja, is tartalmaz első foknál nagyobb tagokat. Emiatt mi egyszerűsítéssel fogunk élni, egyszerűen elhanyagoljuk ezeket a magas fokszámú tagokat, valamint az exponenciális tényezőt is az átviteli függvény legelejéről, ezt az egyszerűsített $G_e(s)$ függvényt az 14. egyenlet mutatja. Ezt az egyszerűsítést meg is tudjuk erősíteni a két rendszer Bode-diagramjának összehasonlításával, a 25. ábra láttatja ezeket. Ezen az ábrán leolvasható, hogy egy ilyen nagy mértékű egyszerűsítés után is nem változik számottevően az erősítés a frekvencia függvényében, legalábbis abban a tartományban, amit mi fogunk használni, hiszen már most kijelenthetjük, hogy nem fogjuk a bemeneti paramétereket $100\ \frac{rad}{s}$-nál gyorsabban változtatni. Az egyszerűsített rendszer fáziseltolása azonban már sokkal nagyobb eltérést mutat, mint az eredeti, itt már a $360°$-os különbség is megjelenik. Azonban mivel az egyszerűsített modell görbéje realisztikusabb viselkedést mutat, a fáziseltolás inkább jobban hasonlít egy két tárolós proporcionális rendszerre, és mivel elsősorban ilyet vártuk egy villanymotortól, így ezt az egyszerűsítést elfogadjuk.

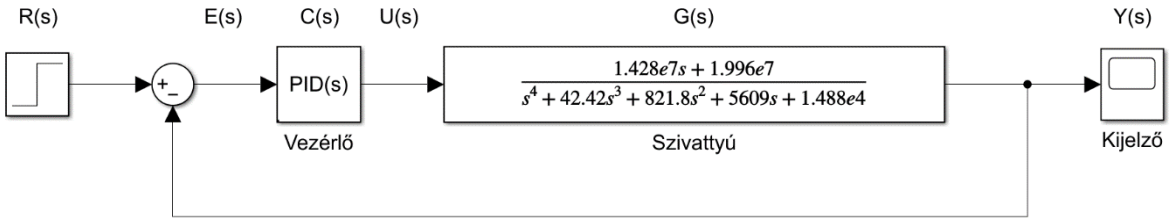
$$G_e(s) = \frac{Y(s)}{U(s)} = \frac{1{,}428 \cdot 10^7 s + 1{,}996 \cdot 10^7}{s^4 + 42{,}42s^3 + 821{,}8s^2 + 5609s + 14\,880} \quad \textit{14. egyenlet}$$

Az előállított átviteli függvényben megjelentek az identifikáció során harmad- és magasabb rendű tagok is a függvény nevezőjében. Mindezen tagok nagy valószínűséggel onnan származhatnak, hogy a fogaskerékszivattyút meghajtó villanymotor tekercselése visszahat az őt megtápláló egységre. A tekercselés induktivitása a rendszert gyorsítja, így ezáltal kissé instabillá is teszi azt és ez úgy jelenik meg a Bode-diagramon, mintha két tárolósként szeretne viselkedni az eszköz, apró frekvenciákra konstans erősítéssel válaszol, majd a frekvenciát növelve növekszik míg el nem érünk egy csúcsot, ahonnan pedig intenzív csökkenésbe kezd a frekvencia növekedésének hatására.

4.3.2 Simulink-modell és hangolás

Miután megkaptuk a $G_e(s)$ egyszerűsített átviteli függvényt, azonnal be is ültetjük azt a MATLAB Simulink kiterjesztésébe. A PID-szabályozás behangolásához szükségünk lesz mindenekelőtt egy *Transfer function* blokkra, egy *PID-control* blokkra, valamint a referencia jelet szolgáltató $1(s)$ egységugrás függvényre, a visszacsatoló jel és a referencia különbségét képző blokkra, illetve egy

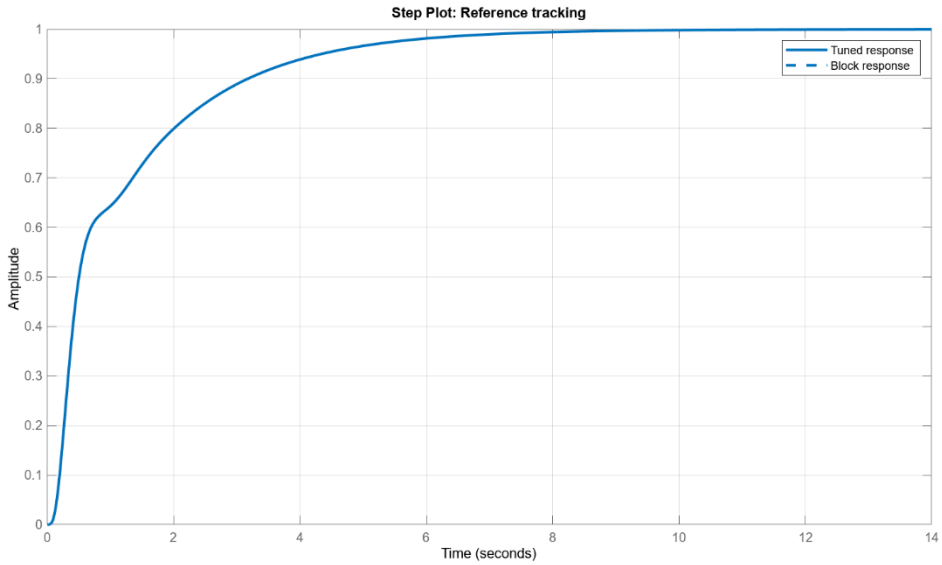
kijelzőre. Ezt a komplett szabályozási hatásvázlatot a 26. ábra mutatja be. Az $R(s)$ referencia jelünk lesz a mindenkori elérni kívánt fordulatszám érték, a PID-szabályozó elkészítéséhez és a szabályozó minőségi jellemzőinek meghatározásához. A referencia jelből ezután kivonjuk a visszacsatolt kimeneti jelet, ekkor kapjuk meg a hibajelet, azaz $E(s) = R(s) - Y(s)$. A szabályozó az ábrán $C(s)$-sel jelölt PID szabályozó, ezt kell nekünk majd behangolnunk a program segítségével úgy, hogy számunkra



26. ábra A Simulinkben felépített modell. Jobbról balra olvasható: referencia jel, hibajel, vezérlő, beavatkozó jel, szabályozott rendszer, kimeneti jel [5].

megfelelő tulajdonságokkal rendelkezzen, illetve a keletkező rendszer stabil legyen. Most szerencsés helyzetünk akadt, ugyanis a szabályozni kívánt rendszer önmagában is stabil. Az innen távozó jel lesz az $U(s)$ beavatkozó jel, ami közvetlenül a szabályozni kívánt $G(s)$ rendszerre fog hatni, amelyből a rendszer számunkra egy $Y(s)$ kimenetet szolgáltat.

Mivel fontos számunkra a jelkövető szabályozás, hiszen pontosan szeretnénk a kívánt fordulatszámot folyamatosan tartani, így elsősorban integráló tagot szeretnénk látni az elkészített szabályozóban, hiszen ez fogja tudni biztosítani nekünk a zérus követési hibát. A PID-tuner blokkot megnyitva automatikusan fog nekünk egy szabályozót felkínálni, azonban mi ezt átalakítjuk oly módon, hogy ne legyen túlzottan rövid, de túl nagy sem a rendszer válaszideje. A mikrokontroller nem képes a folytonos rendszer változásait folytonos módon lekövetni, a visszacsatolásra csak diszkrét időben képes reagálni, ami miatt a későbbiekben át is alakítjuk a szabályozót felépítő egyenleteket. Mivel a szabályozási cél abból áll, hogy a majdani gázturbinához szállítsunk lehetőleg folyamatosan adott



27. ábra A behangolt rendszer egységugrás-függvényre adott válasza.

fordulatszámon tüzelőanyagot, kevés időbeli rövid idejű változtatással, így elsősorban nem a legrövidebb szabályozási idő a cél, azonban ezt túl nagyra sem hagyhatjuk meg, mert ekkor a szabályozás már nem lesz pontos. A mikrokontroller viszont képes lesz nagyon finom felosztású vezérlő jelek küldésére is, nagyon kis változtatásokat is képes lesz a rendelkezésre álló mintavételezési idő alatt végrehajtani.

A behangolt rendszer egységugrás függvényre adott válaszát a 27. ábra mutatja, a minőségi tulajdonságai pedig a 3. táblázat elemeiből olvashatóak ki. Érdemes megfigyelni, hogy a szabályozáshoz nem szükségesek nagy értékek, az integráló tag értéke is 10^{-4} nagyságrendű, valamint a deriváló tag egészen pontosan 0, amely abból a szempontból nem meglepő, hogy nem szeretnénk a rendszert gyorsítani, pusztán jelkövetővé szeretnénk tenni azt. Az arányos tag sem vesz fel nagy értéket, azonban ez sem meglepő, hiszen mivel a rendszer önmagában stabil és a szabályozó csak a visszacsatolás miatt szükséges, voltaképpen nem is lenne rá szükségünk elviekben, azonban a használt implementációhoz szükséges egyenletek miatt érdemes hagynunk egy kis hatást a proporcionális tagra nézve. A szabályozott rendszer emiatt is fog ekkora fázistartalékkal rendelkezni, amely sokkal nagyobb, mint amekkora egy hagyományos szabályozó tervezésénél előírás szokott lenni, ez az érték tipikusan $30°$ vagy $45°$ attól függően, hogy a használt elemek milyen minőségben képesek az elvárt paramétereiket stabilan tartani.

3. táblázat A behangolt szabályozó paraméterei

Proporcionális tag (K_p)	10^{-7}	Integráló tag (K_i)	$6{,}1613 \cdot 10^{-4}$
Deriváló tag (K_d)	0	Deriváló tag szűrőtényezője (N)	100
Válaszidő (T_R)	$2{,}98\ s$	Szabályozási idő (T_S)	$5{,}87\ s$
Fázistartalék (φ_t)	$104°$		

4.3.3 Szabályozó implementálása

A mikrokontroller programján már elérhetők a */ca* és */cr* parancsok, amelyekkel a vezérlés vagy a szabályozás közül tudunk választani, így az utóbbit most feltöltjük valós tartalommal, valódi programmal. A szabályozás rutinja aszerint fog futni, hogy mekkora mintavételezési időt szeretnénk megvalósítani. Ez a mintavételezési idő sajnos a szabályozás során egyelőre biztonsági okokból nem lesz módosítható, azonban sokkal valósághűbb lenne az eszköz, ha parancs útján módosítanánk a szabályozó pontosságát, és ezáltal egy testreszabhatóbb eszközt kapnánk. Előzetesen az implementáláshoz most a programban $T = 0{,}1\ s$-os mintavételezési időt fogunk alkalmazni. Az általunk megvalósított szabályozás típusa az úgynevezett digitális diszkrét idejű elsőrendű PID-szabályozó lesz [22], az ehhez használt egyenleteket pedig a következőkben részletezzük.

Ismert a PID-szabályozás általános, Laplace-tartománybeli egyenlete, amelyből készítünk inverz Laplace-transzformációval időtartománybeli egyenletet, ekkor máris a rendelkezésünkre fog állni az alapegyenlet, amiből kiindulhatunk a szabályozás tervezése során, ez az átalakítást a 15. egyenlet

mutatja be. Itt a K_p-vel jelölt változó jelenti a proporcionális tagot, az $e(t)$-vel jelölt változó jelenti a hibajelet, a T_d és T_i betűkkel jelölt változók pedig a deriválási és integrálási időállandókat. Ebből egyszerűsítés után a belső integrálásból, mivel diszkrét idejű szabályozót készítünk, egy egyszerű összegzést készítünk, azaz a bejövő korábbi hibajelek alapján csak egyszerűen összeadjuk azon diszkrét értékeket, így máris egyszerűbb képlethez jutunk. Ezen felül, mivel mintavételezési idővel dolgozunk, a folytonos idejű szabályozó paramétereit súlyoznunk kell ezzel az idővel, hogy az képes legyen diszkrét időben is lekövetni a szabályozni kívánt rendszert. Ha mindezeket figyelembe vesszük, a 16. egyenlet által leírt változatra jutunk [22]. Itt a mintavételezési időt T betűvel jelöltük.

$$u(t) = \mathcal{L}^{-1}\left(K_p + T_d s + \frac{1}{T_i s}\right) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \qquad \textit{15. egyenlet}$$

$$u(n) = K_p \cdot \left[e(n) + \frac{T_d}{T}\cdot\big(e(n) - e(n-1)\big) + \frac{T}{T_i}\cdot\sum_{i=0}^{n} e(i)\right] \qquad \textit{16. egyenlet}$$

Ahhoz, hogy a programban nem kelljen folyamatos összegzéseket végeznünk, átalakíthatjuk az előző egyenletet úgy, hogy az aktuális beavatkozó jelből kivonjuk az előző iterációban kiadott beavatkozó jel értékét, az ekkor kapott Δu értékét pedig egyszerűen hozzáadhatjuk az előző lépésben kiszámolt $u(n-1)$ jelhez, a végeredményt pedig a 17. egyenlet mutatja be nekünk [22]. Az ekkor kapott összefüggés szerint pedig a mikrokontrollerrel könnyedén, rekurzív módon leszünk képesek a kiadott beavatkozó jel értékét módosítani, ezt fogjuk implementálni elsődleges szabályozó algoritmusként a kontrollerbe, azonban az összehasonlítás végett elkészítjük a 16. egyenlet szerinti előző variációt is.

$$\begin{aligned}&\mathrm{u(n)} = u(n-1) + \Delta u = \mathrm{u(n-1)} + \\ &+K_p \cdot \left[\big(\mathrm{e(n)} - \mathrm{e(n-1)}\big) + \frac{T_d}{T}\cdot\big(\mathrm{e(n)} - 2\mathrm{e(n-1)} + \mathrm{e(n-2)}\big) + \frac{T}{T_i}\cdot \mathrm{e(n)}\right]\end{aligned} \qquad \textit{17. egyenlet}$$

Ekkor a korábban már definiált */cr* paranccsal átválthatunk bármikor visszacsatolt zárt hurkú szabályozási módba, azonban ehhez előbb meg kell határoznunk a T_d, T_i értékeket, ehhez a PID-tuner szoftverből kinyerhető K_p, K_d és K_i változókat fogjuk kiindulási alapnak tekinteni, az átalakításokat a 18. egyenlet és a 19. egyenlet mutatja be. Érdemes megfigyelnünk, hogy mindkét értékben fontos szerepet játszik a mintavételezési idő értéke, ez is meg fogja határozni az értékek nagyságát, hiszen, ha rövidebb idő alatt képesek vagyunk beavatkozni, akkor nem szükséges ezt akkora mértékben megtenni, mintha hosszabb időt hagynánk a rendszernek ahhoz, hogy attól az állapottól tovább álljon. Az így kapott integrálási időállandó $1{,}6230 \cdot 10^{-5}$ lesz a deriválási, mivel K_d értéke korábban is zérus volt, így 0 lesz.

$$T_i = \frac{T}{K_i} \cdot K_p \qquad \textit{18. egyenlet}$$

$$T_d = \frac{K_d \cdot T}{K_p} \qquad \textit{19. egyenlet}$$

A program lefutását tekintve ekkor, ha a */cr* parancsot kiadjuk a kontrollernek, az a belső állapotjelző változók értékét átállítja, majd a program a fő ciklusban képes belépni egy feltételes függvénybe, amely feltétel a visszacsatolt szabályozási mód bekapcsolásához kötött. Tehát, ha a belső *feedbackcontrol* változó értéke igaz, akkor történik visszacsatolt módon a szabályozás, ellenkező esetben csak nyílt hurkú vezérlést fogunk kapni az analóg kimeneten.

A teszteléshez az üzembe helyezés előtt fontos, hogy lássuk, hogyan viselkedne a szabályozó abban az esetben, ha megkapná az irányítást, azonban a biztonság kedvéért először csak a vezérlő mellett engedjük majd, azaz folyamatosan egy adott analóg kimenetet adunk, eközben figyeljük, hogy a bemenő adatok alapján erre mit tenne a szabályozó algoritmus. Ehhez szükséges az, hogy a szabályozó paramétereinek kiszámítása ne csak a *feedbackcontrol* változó igaz értéke esetén történjen meg, hanem minden esetben kivétel nélkül, legfeljebb nem használjuk fel azt. Emiatt a számításokat csak a főciklusba helyezzük el, pontosabban egy szubrutint készítünk, amelyet a főciklusban minden feltétel nélkül hívunk majd meg.

5 LabVIEW kezelőfelület készítése

Ha a mikrokontroller UART kapcsolatban áll egy számítógéppel, így adódott a lehetőség, hogy készítsünk ehhez egy olyan felületet, amely grafikusan is képes megjeleníteni minden számunkra szükséges információt, valamint képes arra, hogy parancsokat adjon a kontroller számára és ezzel voltaképpen a számítógép képernyőjéről könnyedén tudnánk azt vezérelni, valamint kiértékelni. Ezt a felületet már identifikációra nem használnánk, azaz elég lenne nekünk csak a szabályozáshoz szükséges paramétereket megjeleníteni. Ehhez egy megfelelő szoftver a LabVIEW, amely képes nemcsak grafikus megjelenítésre, hanem a beérkező információk valós idejű feldolgozására és akár átalakítására.

A LabVIEW (*Laboratory Virtual Instrument Engineering Workbench*) a National Instruments által fejlesztett programozási környezet, amely elsősorban automatizált kutatásokhoz, valamint rendszerek validációjához és teszteléséhez használatos [23]. A programban a kódok és nyelvek helyett grafikus programozási felület használatát kell elsajátítanunk, ez az úgynevezett grafikus kód („G” nyelv), a program maga pedig C nyelven íródott. A megírt programok lesznek az úgynevezett virtuális műszerek, amelyeket be is ágyazhatunk al-műszerekként tetszés szerint egymásba, mintha külön szubrutinokat alkalmaznánk egy hagyományos programozási feladat során. Ennek szerves részét képezi a felhasználói felület is, amellyel minden programozási feladatunk indulni fog, ez lesz a virtuális műszer kezelőfelülete, itt fogunk tudni interakcióba lépni a programmal, azon keresztül pedig majd a mikrokontrollerrel [24].

Egy virtuális műszer nem csak a kezelőfelületéből áll, hanem ahhoz tartozik egy blokkdiagram is, ebben tudunk G nyelven programot készíteni. A műszerben két típusú kezelőfelületi eszköz lehetséges: amelyik adatok bevitelére szolgál és amelyik adatok kivitelére, megjelenítésre szolgál. Ez láthatóan jelezve is van a program blokkdiagramján, a bemeneti eszközökhöz jobb oldalról tudunk csatlakozni, a kimeneti eszközökhöz bal oldalról tudunk.

5.1 Kontroller oldala

A LabVIEW alapvetően más struktúrában fog a kontrollerrel UART kommunikáción keresztül „beszélgetni”, ehhez kérdés-válasz párok szükségesek, ugyanis nincsen szükségünk arra, hogy minden egyes adandó alkalommal lefoglaljuk azzal a kontrollert, hogy erőforrásait felhasználva adatokat küldjön a számítógép felé, erre csak akkor lenne szükség, ha mi kérdjük az adatot. Így születtek meg a programban a kérdező kódok, azaz amit a későbbiekben csak q-kódoknak fogunk hívni. Ezek a kódok alapvetően teljesen ugyanazt a struktúrát követik mint az előzőekben megismert, billentyűzetről begépelhető *’/’* karakterrel kezdődő parancsok, azonban ahhoz, hogy billentyűzetről ne így irányítsuk az eszközt, egy nehezebb karaktert, a függőleges elválasztóvonalat (*’|’*) tettük meg parancsindító és néhol lezáró karakternek. Ekkor lehetőségünk nyílik arra, hogy új és egyedi parancsokat definiáljunk, amelyek bár kódjukban majdnem megegyeznek az eredetiekkel, azonban teljesen más funkció ellátására is képesek, valamint ami ennél is fontosabb, nem küldenek visszajelzést a számítógépnek direkt

formában azonnal, hanem csak a visszajelző LED-ek segítségével kommunikálnak a kezelő személlyel. A q-kódok listáját a 4. táblázat tartalmazza.

4. táblázat A LabVIEW program által UART kommunikáción keresztül kiküldött kérdés-válasz, az általunk q-kódnak nevezett parancsok teljes listája.

|r0	A szabályozó engedélyezéséhez szükséges relé kikapcsolása
|r1	A szabályozó engedélyezéséhez szükséges relé bekapcsolása
|ca	Nyílt hurkú vezérlési mód
|cr	Fordulatszám-visszacsatolt vezérlési mód
|qr	Fordulatszám mérés eredményének „megkérdezése” a kontrollertől, erre a válasz az „R” betűvel fémjelzett számsor
|qa	Pillanatnyi analóg kimenet értékének „megkérdezése” a kontrollertől, erre a válasz az „A” betűvel fémjelzett számsor
|sa#|	Szabályozó konstans értékre való beállítása
|sr#|	Szabályozó referencia fordulatszámának beállítása
|d0	Debug mód kikapcsolása
|d1	Debug mód bekapcsolása

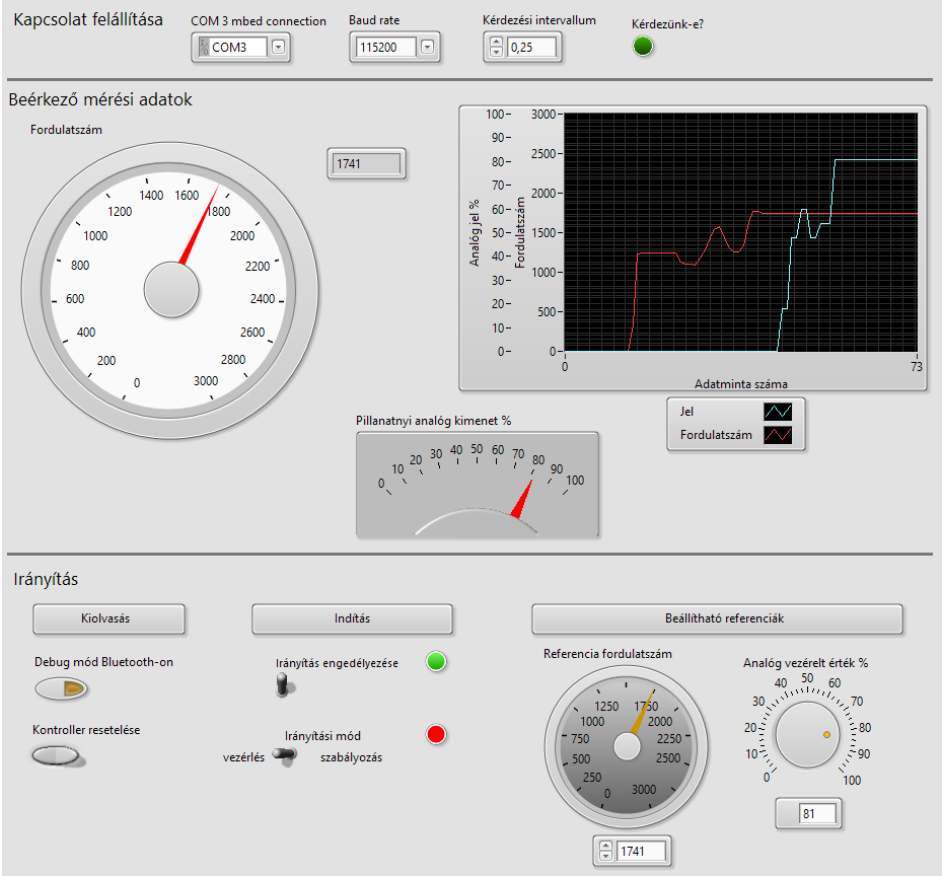
5.2 Kezelőfelület

Ha már elkészültünk a kontroller programjával, akkor ideje elkészítenünk a szabályozóhoz tartozó virtuális műszert. Először hozzuk létre a kezelőfelületünket, amelyet a 28. ábra mutat be teljes egészében. Itt három szakaszra bontottuk a feladatokat és a kezelőfelületet: az első szakasz az inicializációért felel, itt tudunk kapcsolatot létesíteni a mikrokontrollerrel UART-on keresztül; a második szakasz felelős a kontroller által a q-kódokra küldött válasz megjelenítésért; a harmadik szakaszban pedig mi magunk tudunk beavatkozni a különböző kapcsolókon és tekerőkön keresztül a szabályozásba. A harmadik szakaszban definiált kezelőszervek fogják küldeni a parancskódokat a kontroller számára.

Először szükségünk van a kapcsolat inicializásásra, itt beállíthatjuk, milyen kontroller az, amivel kapcsolódni szeretnénk, valamint a *baud rate*-et is beállíthatjuk, azonban ennek egyeznie kell a kontrollerben definiált rátával. Az első szakaszba tudjuk a q-kódok közötti intervallumot is beállítani, valamint visszajelzést kapunk arról, hogy zajlik-e éppen kérdezési művelet.

A második szakaszt a beérkező adatok megjelenítésére használjuk, a két legfontosabb paraméterünk a fordulatszám és az analóg vezérlő jel lesz. A mérés során használt diagramok gyors szemrevételezést biztosítani, emiatt definiálunk egy gráfot is, amin a beérkező analóg kimeneti értékeket, valamint a fordulatszámot jelenítjük meg az idő függvényében, ezzel lehetőséget biztosítunk magunknak arra, hogy rögtön kiértékeljünk egy adott eseményt. Ezen felül viszont a pillanatnyi értékekre is szükségünk van, így két nagy mutatós műszert is elhelyezünk a második szakaszban, amelyek a pillanatnyi fordulatszám és az analóg kimenet értékeit mutatják meg.

A harmadik szakaszban pedig az 5.1 fejezetben definiált parancskódokat tudjuk kiadni. Itt kapnak helyet a különböző kapcsolók, amik a vezérlést engedélyezik és a vezérlés módját választják ki. Beállíthatjuk tekerős gombok segítségével a vezérelt esetben kiadott analóg értéket, vagy szabályozott esetben a referencia fordulatszámot, amely ugyanolyan stílusú mutatós műszerben állítható be, így is segítve a könnyű tájékozódást a virtuális műszerben. Mivel nem szeretnénk az UART kommunikáció nyers szövegét olvasni, mint egy terminál szoftver esetében, így be kell építenünk egy debug-módot kapcsoló gombot, amellyel elindítható vagy megállítható a Bluetooth-on keresztül történő adatküldés. Utolsó lépésben hozzáadunk egy olyan gombot, amely megnyomásával az mbed-et újraindíthatjuk, így ha valamilyen problémába ütközünk, például megtelne a kontroller író puffere vagy új programot készítettünk, akkor így is bele tudjuk programozni azt.

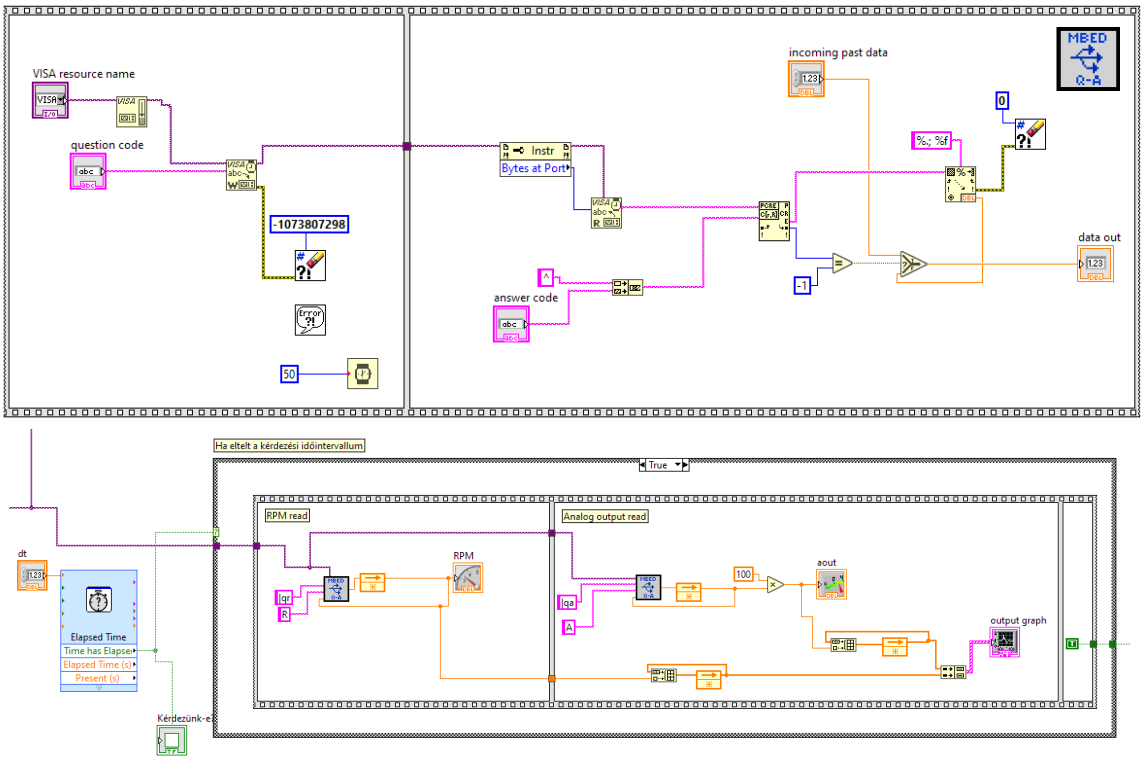


28. ábra Az elkészített vezérlési felület a beállítási lehetőségekkel és kijelzőkkel, az interfész. A kép a program működése közben készült.

5.3 Program és alműszer

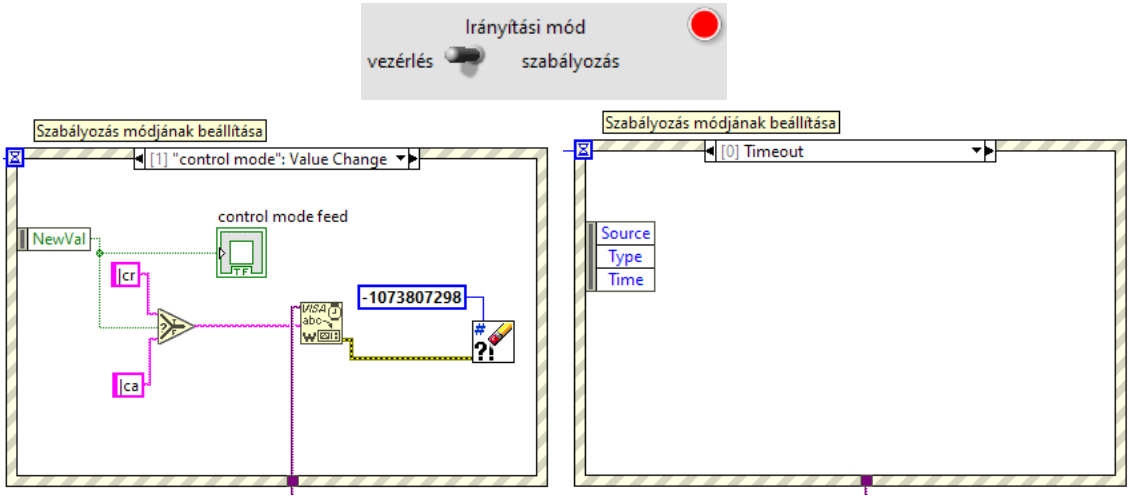
A legfontosabb a kérdés-válasz párok értelmezése, ehhez létrehozunk egy új alműszert a mikrokontrollerrel való kommunikálásra, ugyanis többször meg kell majd ezt a programrészletet hívnunk, így célszerű moduláris módon hozzáállnunk, ez lesz az *mbed Q/A* nevű program. Ebben a programban definiálunk bemeneteket és kimeneteket, így tudunk ezzel fordulatszámot és a pillanatnyilag kiadott analóg jelet is kérdezni. Speciális formázást fogunk a kontrolleren használni, így, ha egy fordulatszám értéket kérdezünk, az *R####* formátumban fogja azt elküldeni, ha pedig az analóg kiadott jel nagyságára vagyunk kíváncsiak, *A##* formátumban fogunk jelet kapni, ezt is egy bemenetre

kötjük. Az alműszernek a további bemeneteire fogjuk adni a kérdező kódot, valamint amennyiben nem megfelelő a kiérkező kapott érték formátuma, akkor az előző alkalommal kapott értéket küldjük tovább újbóli megjelenítésre. Ezt a programrészletet a 29. ábra mutatja be a G-nyelven. Az ábráról leolvasható, hogy egyszeri futtatás történik a meghívásakor, a kérdés szekció a bal oldalon, a válasz értelmezése a jobb oldalon látható, a keretezés arra szolgál, hogy a LabVIEW ezeket a parancsokat meghatározott sorban futtassa le. A két részlet közé beékeltünk egy 50 *ms*-os késleltetést is így biztosak lehetünk benne, hogy a válasz megérkezik.



29. ábra Az elkészített alműszer, az mbed-del való kommunikálásra (felül) és annak valós környezetbe való beépítése a kérdezési időintervallummal együtt (alul). A hibák kezelésére azért van szükség, mert a működőképesség és problémamentesség ellenére a futtatás közben a szoftver hibát jelzett.

Az elkészített program kezelése egyszerű lesz, a megadott és beállított vezérlőket az mbed programozásánál már megismert *interrupt*-szerű módon fogjuk elkészíteni, úgynevezett eseménystruktúraként (*event structure*). Ekkor, ha lenyomunk egy gombot, vagy tekerünk az egyik beállítógombon, az kis szükséges késleltetéssel továbbítja a parancsot az mbed felé, amely reagál is erre. Emellé megadhatunk egy Timeout esetet is, ha nem érkezik az általunk megadott időben parancs, akkor az eseménykezelő egyszerűen lemond az eseményről és nem fog semmi sem történni. A tekerőgombok esetében fontos beállítanunk a maximális felsorakoztatott műveletek számát is, amelyet mi 1-nek határozunk meg, ha ez túl sok lenne, azzal feleslegesen sok művelettel foglalnánk le a kontrollert. Egy ilyen eseménykezelést mutat meg a 30. ábra, amely az irányítási mód változtatásának példáján keresztül mutat be egy eseménykezelést.



30. ábra A felül látható kezelőszerv eseménykezelője, ahol a control mode feed a jelenleg pirosan világító lámpát jelenti, bal oldalt, ha bekövetkezik az esemény, jobb oldalt pedig, ha timeout helyzet alakul ki.

Amint elkészült az egész program, a LabVIEW két opciót kínál fel futtatáshoz: egyszeri vagy folyamatos futtatás. Ha az utóbbi mellett döntünk és így szeretnénk a programunkat indítani, akkor azt tapasztalhatjuk, hogy a beépített késleltetések, valamint *Ticker*-szerű objektumaink nem működnek. Ez azért történik így, mert ezek csak cikluson belül tudnak operálni, azaz egy folyamatban eltelt időt mérik. Éppen emiatt az egész programot egy nagy *while*-ciklussal fogjuk össze annak érdekében, hogy csak meghatározott intervallumokban fusson le a kívánt kód és nem minél hamarabb, amint végzett az előző ciklussal a program kezdje elölről a futtatást, így nem foglaljuk le a kontrollert.

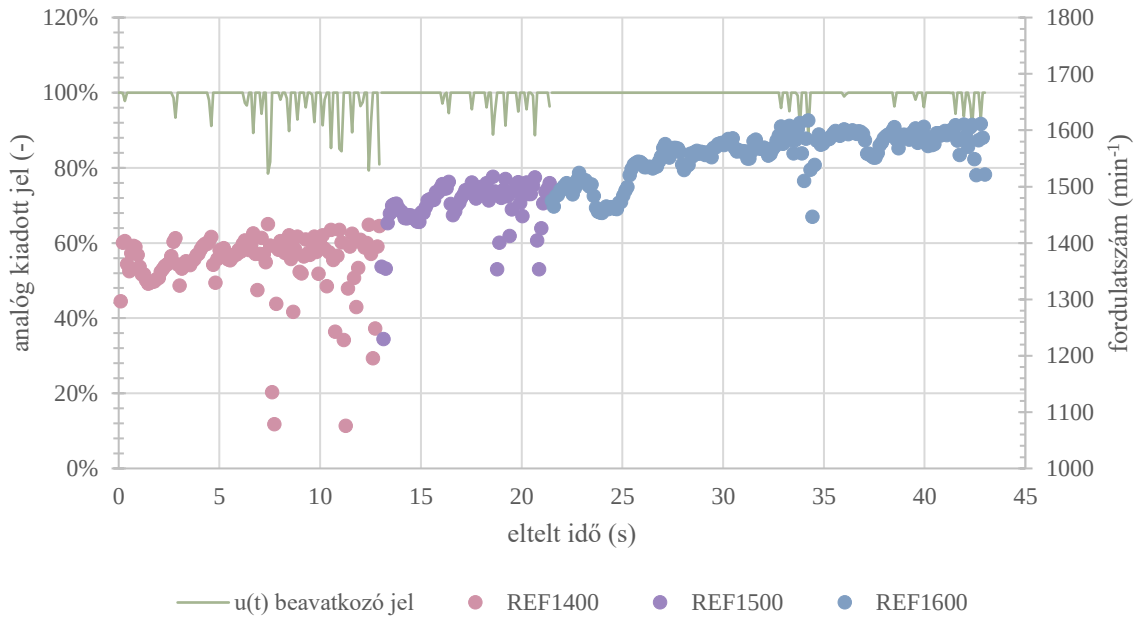
Minden futtatás előtt azonban meg kell győződjünk arról, hogy kapcsolódtunk-e a mikrokontrollerre, különben hibába fog a program ütközni és meg fog állni. A hibák ezen felül jelentkeznek még számos helyen a program futása során, azonban ezeket nagyon sok munka árán sem sikerült elhárítanunk. Ezek a hibák azonban nem akadályozzák a program megfelelő működését, a kért feladatokat ellátja, az adatokat maradéktalanul kijelzi, a kért parancsokat sikeresen továbbítja, valamint a beállítandó értékeket a kontroller megkapja és be is építi azokat.

Mindezzel már egy olyan programot kaptunk, amit tudunk éles környezetben is használni az elkészített szabályozóval, azt grafikusan megjeleníteni és amennyiben szükséges, olyan fejlesztéseket beleépíteni a programba, amellyel akár belső számításokat is végezhetünk és a különböző szenzorok értékeit már közvetlenül itt a LabVIEW környezetben értékeljük ki, majd ez alapján küldjünk a beavatkozókra saját kimenetet. A program erre is alkalmas, viszont a szabályozó eredeti célja nem a kizárólag számítógéppel való működés, hanem fontosnak tartjuk, hogy e nélkül is képes legyen végrehajtani a feladatát, így egyelőre ilyen módon nem építjük ki a virtuális műszerünket.

6 Összegzés

6.1 Észrevételek

A szabályozó megépítése és a kontrollerbe történő implementálása után azt azonnal ki is próbáltuk. A tapasztalataink alapján a szabályozó sajnos nem végez olyan pontos munkát, mint amilyet mi szerettünk volna. A 31. ábra mutatja be a tervezett szabályozóval végzett mérést, ahol több referencia fordulatszámot is felhasználtunk. Itt leolvasható, hogy a szabályozó a magasabb fordulatszámokon üzemel megfelelően, mi is magasabb referenciával végeztük el a próbát. Alacsony fordulatszámokon nagy lengések következtek be a szabályozott rendszerben, azonban ahogy növeltük a referenciát, látható az ábrán, hogy a valós fordulatszám enyhén növekedik addig, amíg egy kisség meg nem haladja a referencia értékeket, így a követési feltételünk teljesül. Azonban a kontroller által kiadott beavatkozó jel nem mutat nagy változásokat, folyamatosan 1 körüli értékeket vesz fel, emiatt az $1600\ \frac{1}{min}$-es fordulatszámot is csak nagyon lassan tudja elérni a szivattyú, amint látszik is, hogy váltunk, nagyon enyhe emelkedést tapasztalunk csak.



31. ábra Az előző fejezetekben tárgyalt szabályozó beépítése utáni rögzített fordulatszám értékek, a színek a különböző beállított referencia fordulatszámokat jelölik.

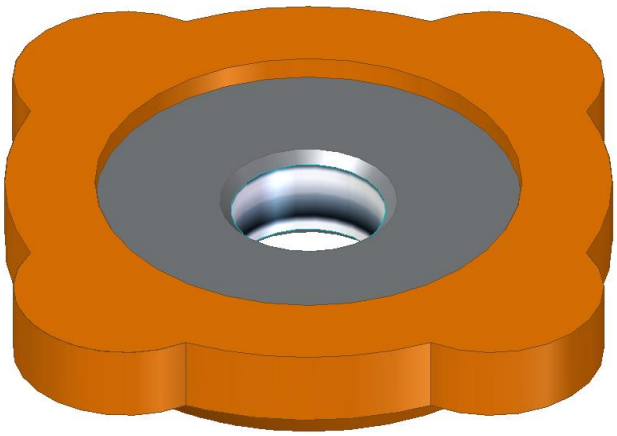
A kis fordulatszámokon bekövetkező lengéseknek több oka is lehet, akár az adatgyűjtés módja is. A fordulatszám mérés *interrupt* típusként kerül kiszámításra, így voltaképpen csak akkor kapunk új fordulatszám értéket a kontrollerben, amikor a motor egy körülfordulást már megtett, tehát csak ekkor tudjuk a beavatkozó jel értékét is változtatni. Ha túl alacsony a szivattyú fordulatszáma, például $600\ \frac{1}{min}$, akkor a $10\ Hz$ esetén nem elégséges időben változik a bemenő fordulatszám értéke a hibajelek számításánál, vagy akár el is csúszhat időben, nem akkor kap fordulatszám bemenetet, amikor éppen a beavatkozó jel értékét frissíti és ez okozhatja akár a szabályozó nem megfelelő viselkedését.

Egy másik ok lehet, hogy sajnos ezt a nagyon nagy lengést nem képes lekövetni a szabályozó rendszerünk a jelenlegi formában, mivel nincsen információja arról, hogy a nyomóágban milyen feltételek uralkodnak, így nem tudjuk ehhez igazítani a kimenetet. Ez a jelenség jórészt indításkor, valamint alacsonyabb fordulatszámokon számottevő, ahogyan növeljük a referenci fordulatszám értékét, a szivattyút annál kevésbé érinti a periodikus lengések hatása. Ez a jelenség adódhat akár a porlasztó okozta ellenállás miatt is, ahhoz, hogy a szállított folyadék a gyűjtőtartályba beporlasztásra kerüljön, szükséges egy megfelelően nagy nyomóoldali nyomás, ennek a hiányában a folyadék csak mint egy kis sugár kerül be a tartályba. Amint beindul a porlasztás, sokkal stabilabb rendszert kapunk, előzőleg a fúvóka csak küzd azon, hogy képes legyen porlasztani és ekkor periodikusság is kialakulhat. A lengések kis fordulatszámon nagyon jól megfigyelhetőek, itt a nyomóágban akár 2-3 bar-os nyomáskülönbségeket is fel tudunk fedezni a nyomásmérőn.

6.2 Módosítási javaslatok

6.2.1 Fordulatszám mérési pontosságának javítása

A jövőre nézve mindenképpen szeretnénk pontosabbá tenni a fordulatszám méréseket. Ehhez elsősorban növelnünk kell az egy körülfordulás alatt beérkező adatok számát a jelenlegi 1-ről például 30-ra. Mivel a jelenlegi mérőegység egy Hall-szenzor, így adódik a lehetőség, hogy az egy darab mágnest helyettesítsük több kisebb mágnessel. A tengely végére erősíthetnénk egy szintén egyedi tervezésű tokban apró, de lehetőleg erős mágneseket, hogy a szenzor a mágneses tér változását érzékelni tudja megfelelő távolságból. Ráadásul ez lenne a legkézenfekvőbb megoldás, hiszen a jelenlegi felépített eszközeinket nem, vagy csak nagyon kis mértékben kellene módosítani. A programhoz pedig egyetlen osztást szükséges ekkor csak beépíteni, ahol az osztó a mágnesek számával lesz egyenlő. Egy ilyen előzetes tervet a 32. ábra mutat 4 darab apró mágnessel a rögzítőben.



32. ábra Fordulatszám-mérő változtatásának koncepciója.

6.2.2 Nyomásingadozások, -lengések kiküszöbölése

Több ötlet is felmerült, hogyan lehetne úgy javítani a modellünkön, hogy az ne reagáljon ilyen módon a nyomások ingadozásaira. A nyomóágban el lehetne helyezni egy aktív ellennyomás-szabályozót, amely képes lenne a nyomóágban lévő nyomást állandó értéken tartani, ezzel kiküszöbölve

annak visszahatását a szivattyúra. Ekkor mind a palackban, mind pedig a nyomóágban kellene az éppen ott uralkodó közeg nyomását tudnunk, hiszen itt differenciális nyomásérték lesz az, ami az aktív szerepet fogja játszani, ugyanis a palackban nem elhanyagolható nyomás uralkodik. Ehhez már van a birtokunkban két nyomásmérő cella, amelyek a porlasztó előtt és a palackban uralkodó nyomások mérését teszi lehetővé. Ekkor elérhetővé válna az is, hogy különböző ellennyomásokat beállítva tudnánk különböző karakterisztikákat meghatározni a szivattyúra, amely a majdani TKT-1-es gázturbinás próbapaddal történő integrációt segítené elő.

Másik megoldásként akár elképzelhető lenne a több paraméterrel történő szabályozás kérdése is, akár MISO (*multiple input, single output*) vagy MIMO (*multiple input, multiple output*) rendszerként is fel lehetne fogni a szabályozást. Ehhez az eszközeink a MATLAB-ban továbbra is a rendelkezésünkre állnak, hiszen a szoftver képes arra, hogy több paraméterrel rendelkező rendszert is identifikáljon, azonban erre egyelőre még nem indítottunk kísérletet. A módszer ugyanígy működne, viszont az identifikációra váró *iddata* típusú adat két vektor formájában kerülne megadásra mind a be- mind a kimenetek oldaláról.

6.2.3 Nyomástól függő szabályozás és új paraméterek

A jövőben aszerint lehetne fejlesztéseket eszközölni a szivattyús próbapadon, hogy az immár működőképes szabályozóval ellátott szivattyú nyomástól függő karakterisztikáját is meg lehetne határozni, így elérhetővé válna a számunkra, hogy a visszacsatolt szabályozó paraméter már ne is a fordulatszám, hanem egyenesen a nyomóágbeli, szivattyú által keltett nyomás legyen. Azért lenne ez a megközelítés a jövőben relevánsabb, mert hosszabb távon pontosabb lehetőséget biztosít majd egy gázturbinával való összekapcsolásra.

Az identifikáció során továbbá nem vettük figyelembe a tüzelőanyag szivattyún történő átszállítása során elszenvedett hőmérsékletnövekedését, amely hosszabb távon azonban már szembetűnő. A szivattyú utáni nyomóágban uralkodó nyomás és a palackba történő beporlasztás nyomásesése erősen függ a szállított közeg viszkozitásától, amely hőmérsékletfüggő. Ahhoz, hogy egy adott alternatív tüzelőanyagot minden tekintetben vizsgálni tudjunk, elengedhetetlen annak ideális porlasztási tulajdonságainak meghatározása, ezt azonban teljeskörűen csak a hőmérséklet növelésével vagyunk képesek kielégítően meghatározni. Ezen felül a repülőgépekben alkalmazott tüzelőanyag a hajtóművek égésterébe való belépése előtt rendszerint előmelegített állapotban jut egyáltalán a szivattyúhoz is, egy alternatív tüzelőanyaggal üzemelő repülőgép esetében ez még hangsúlyosabb lenne a porlasztási tulajdonságok eltérése miatt.

Felhasznált irodalom

[1] V. Karnozov, „Aviation International News Online,” 2019. augusztus 19. [Online]. Elérhető: https://www.ainonline.com/aviation-news/air-transport/2019-08-19/aviadvigatel-mulls-higher-thrust-pd-14s-replace-ps-90a [Hozzáférés dátuma: 2023. szeptember 28.]

[2] „ICAO Offical page,” ENSZ, [Online]. Elérhető: https://www.icao.int/environmental-protection/pages/aircraft-engine-emissions.aspx [Hozzáférés dátuma: 2023. szeptember 28.]

[3] „Airline Data Project - expenses Excel tables,” [Online]. Elérhető: https://web.mit.edu/airlinedata/www/Expenses&Related.html [Hozzáférés dátuma: 2023. szeptember 26.]

[4] IATA, „IATA Jet fuel price monitor,” 22. szeptember 2023. [Online]. Elérhető: https://www.iata.org/contentassets/9036deaf9c984009a3515fd6aa1c5e24/jet_fuel_price_devt_longterm.jpg [Hozzáférés dátuma: 2023. szeptember 28.]

[5] Enerdata, „Enerdata World energy & climate statistics,” 2023. [Online]. Elérhető: https://yearbook.enerdata.net/crude-oil/world-production-statistics.html [Hozzáférés dátuma: 2023. szeptember 28.]

[6] K. T. Dr. Beneda, Repülőgép-hajtóművek elmélete II., Budapest: Akadémiai Kiadó Zrt., 2018.

[7] A. Osborne, An Introduction to Microcomputers Volume 1: Basic Concepts, Berkeley, Kalifornia, USA: Adam Osborne and Associates Incorporated, 1976.

[8] R. Toulson és T. Wilmshurst, Fast and effective embedded systems desing applying the ARM mbed, Oxford: Elsevier Ltd., 2017.

[9] „Raspberry Pi webpage,” Raspberry Pi Ltd., 2023. október [Online]. Elérhető: https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf. [Hozzáférés dátuma: 2023. november 05.]

[10] K. T. Beneda, „Kisméretű gázturbinás sugárhajtóműves berendezés kísérleti és oktatási célokra,” Szolnok, 2008.

[11] Ismeretlen szerző, Паспорт 924 ПС (1. melléklet), 1982.

[12] J. P. Stengl és J. Tihanyi, Teljesítmény-MOSFET-ek és alkalmazásaik, Budapest: Műszaki Könyvkiadó, 1990.

[13] R. Johansson, System modeling and identification, New Jersey: Prentice-Hall Inc., 1993.

[14] J. Bokor, Irányítástechnika járműdinamikai alkalmazásokkal, Budapest: Typotex, 2008.

[15] L. Andersson, U. Jönsson, K. H. Johansson és J. Bengtsson, „A manual for system identification”

[16] R. J. Hyndman és G. Athanasopoulos, „Forecasting: principles and practice, 2nd edition,” OTexts, Melbourne, 2018.

[17] „ARM mbed áttekintő oldal,” [Online]. Elérhető: https://os.mbed.com/platforms/mbed-LPC1768/ [Hozzáférés dátuma: 2023. június 26.]

[18] „Hobbyking.com,” [Online]. Elérhető: https://hobbyking.com/en_us/turnigy-30a-brushed-esc.html?___store=en_us [Hozzáférés dátuma: 2023. június 27.]

[19] „ASCII-code,” Injosoft AB., [Online]. Elérhető: https://www.ascii-code.com/ [Hozzáférés dátuma: 2023. október 23.]

[20] ARM Limited, „ARM mbed Keil Studio Cloud,” [Online]. Elérhető: https://os.mbed.com/docs/mbed-os/v5.15/apis/rawserial.html [Hozzáférés dátuma: 2023. október 23.]

[21] „Microchip Technology RN41 module overview,” Micorchip Technology Inc., [Online]. Elérhető: https://www.microchip.com/en-us/product/rn41 [Hozzáférés dátuma: 2023. október 23.]

[22] M. R. Avazpour, F. Piltan, M. H. Mazloom, A. Sahamijoo, H. Ghiasi és N. B. Sulaiman, „Research on First Order Delays System Automation,” International Journal of Grid Distribution Computing Volume 8. No. 4., 2015.

[23] National Instruments webpage „What is LabVIEW?” National Instruments Corp., [Online]. Elérhető: https://www.ni.com/en/shop/labview.html [Hozzáférés dátuma: 2023. november 05.]

[24] National Isntruments, LabVIEW Documentation, 2016

Mellékletek

1. melléklet 924-es szivattyú útlevele

Таблица результатов контроля параметров

Дата	Причина контроля	Наработка с начала эксплуат., час.	Результаты контроля	Подпись проводившего контроль
1	2	3	4	5
	Приемо-сдаточные испытания		ГОДЕН	

НАСОС МАСЛОТОПЛИВНЫЙ 924

ПАСПОРТ

924 ПС

на изделие № 21450325

19 81

1. ОСНОВНЫЕ ТЕХНИЧЕСКИЕ ДАННЫЕ

Наименование параметра	Характеристика
1	2
1. Режим работы	[illegible]
2. [illegible]	[illegible]
3. [illegible]	27 [illegible]
4. [illegible]	[illegible]
5. [illegible]	2
6. [illegible]	40
7. [illegible]	3,4...5
8. [illegible]	6...15
9. [illegible]	14...22
10. [illegible]	1,5...4,2
11. [illegible]	от минус 50 до плюс 60
[illegible]	до плюс 120

1. ОСНОВНЫЕ ТЕХНИЧЕСКИЕ ДАННЫЕ

Наименование параметра	Характеристика
1	2
Давление впрыска топлива при Рвх=2 кгс/см², U=27 В (на состояние поставки), кгс/см²	8±1
13. Давление топлива на входе в изделие, кгс/см²:	
1) на режиме разгона	1,5±0,3
2) на режиме конечного давления и в подаче	1,5 $^{+1}_{-0,5}$
14. Начальное давление топлива при Рвх=2 ати, Рк=0 кгс/см², U= 27 В (на состояние поставки), кгс/см²	3,1 +0,4
15. Конечное давление при Рк=2,5 кгс/см², U=27 В, Рвх.л=2 кгс/см² (на состояние поставки), кгс/см²	17,7±0,25
16. Изменение давления топлива на выходе из агрегата в зависимости от давления Рк.	См. приложение
17. Производительность масляного насоса при давлении на входе 100...1500 мм столба рабочей жидкости, на выходе 2,5 кгс/см² и напряжении на клеммах электродвигателя 27 В, л/ч, не менее	120
18. Давление полного перепуска масла при U=27 В, кгс/см²	6
19. Герметичность: просачивание отпотевание по разъемам и станкам	Не допускается

1. ОСНОВНЫЕ ТЕХНИЧЕСКИЕ ДАННЫЕ

Наименование параметра	Характеристика
1	2
20. Дренажные утечки при давлении топлива на входе 2,5 кгс/см², не более:	
1) при неработающем изделии, см³/мин.	3
2) при работе изделия за три включения, см	13
3) дренажные утечки при неработающем изделии и отсутствии подкачки на входе, см³/мин.	0,3
21. Масса изделия, кг	6,6

2. КОМПЛЕКТ ПОСТАВКИ

Наименование	Шифр	Кол-во	Заводской номер	Примечание
1	2	3	4	5
Насос маслотопливный	924	1	—	
ЗАПЧАСТИ:				
Кольцо уплотнительное	2262А-175	3	—	
Кольцо уплотнительное	2262А-176	1	—	
Кольцо уплотнительное	2262А-177	2	—	
Кольцо уплотнительное	2267А-17	2	—	
ДОКУМЕНТАЦИЯ:				
Паспорт на электродвигатель ЭМ-660Т		1	0680100200	
Паспорт (этикетка) на электромагнит ЭМТ-503А		1	068002058	

3. РЕСУРСЫ, СРОКИ СЛУЖБЫ, СРОК ХРАНЕНИЯ

Ресурс изделия до первого ремонта — [illegible] включений в течение срока службы 12 лет, в том числе срок хранения в упаковке изготовителя в складских помещениях ______ год (а) в условиях тропического и морского климата — 1 год.

Межремонтный ресурс — [illegible] включений в течение срока службы 12 лет.

Назначенный ресурс — 4500 включений.

Количество ремонтов в пределах назначенного ресурса не ограничивается.

Указанные ресурсы, сроки службы и срок хранения действительны при соблюдении потребителем требований действующей эксплуатационной документации.

4. КОНСЕРВАЦИЯ И РАСКОНСЕРВАЦИЯ

Дата	Наименование операции	Срок действия	Подпись
1	2	3	4
15.06.90	Внутренняя консервация с применением масла МК-8 ГОСТ 6457-66 или масла трансформаторного ТК ГОСТ 982-68. Наружная консервация с применением масла МК-22 или МС-20 ГОСТ 21743-76 с добавлением 6–10% церезина ГОСТ 2488-79	1 год	[signature] ОТК 30
25.07.07	Расконсервация		[signature]

5. СВИДЕТЕЛЬСТВО О ПРИЕМКЕ

Изделие изготовлено [illegible] действующей технической документацией и признано [illegible] ации.

Главный контролер ____________ (подпись)

М. П.

« __ » ____________ 19 81 г.

6. ДВИЖЕНИЕ ИЗДЕЛИЯ В ЭКСПЛУАТАЦИИ

Дата установки	Шифр и номер объекта	Дата снятия	Наработка с начала эксплуат., час., мин.	Причина снятия	Подпись за установку (снятие)
1	2	3	4	5	6
13.6.90	ТС-21 030230	21.5.90	1396 [illegible]	Ремонт двиг.	[illegible]
		1134			[illegible]
		34 07.07.	436 [illegible] / 1832 [illegible]	[illegible]	[illegible]

7. ПРОВЕДЕННЫЕ РЕМОНТЫ И ДОРАБОТКИ ПО БЮЛЛЕТЕНЯМ И УКАЗАНИЯМ

Дата	Наименование работы	Основание (наименование, номер и дата документа)	Организация, проводившая работу	Новый ресурс и срок службы изделия	Подпись ответственного лица
1	2	3	4	5	6
13.6.90	[illegible] АРП [illegible]	[illegible]	[illegible]		[illegible]

7.1. Краткие записи о произведенном ремонте

(ТТ Кап ремонт)

Произведён [illegible] АРП [illegible] в [illegible] действующей [illegible] документацией.

7.2. Свидетельство о приемке после ремонта

(П Кап ремонт)

Изделие отремонтировано ремонтным предприятием (предприятием-изготовителем) в соответствии с действующей Технической документацией, принято и признано годным для эксплуатации.

Ресурс изделия до очередного капитального ремонта 1500 ч. после

в течение срока службы 12 лет, в том числе срок хранения — лет

(года)

в упаковке исполнителя ремонта, в складских помещениях,

на открытых площадках и т. п.

Указанные ресурс, срок службы и срок хранения действительны при соблюдении потребителем требований действующей эксплуатационной документации.

Начальник ОТК (подпись)

М. П.

«13» 06 1980 г.

8. ЗАМЕТКИ ПО ЭКСПЛУАТАЦИИ И ХРАНЕНИЮ

8.1. [illegible] (предприятия-изготовителя во время действия гарантийных обязательств) не должны, кроме пломб на транспортировочных заглушках, [illegible] регулирования в соответствии с Руководством по эксплуатации.

8.2. В полевых условиях изделие разбирать не допускается.

8.3. Не допускать попадания консервирующих и промывочных жидкостей на электрические разъемы, в дренажный штуцер, на разъем между корпусом изделия и электрожгутом и в суфлирующее отверстие в корпусе сильфона клапана-регулятора.

8.4. Перед установкой изделия проверить:

1) отсутствие внешних повреждений;

2) наличие контровок и пломб;

3) расконсервацию изделий.

ХАРАКТЕРИСТИКА $P_т = f(P_к)$

$P_к$, кгс/см²	Давление $P_т$, кгс/см²	
	расчетное	замеренное
0,1	$3^{+0,75}$	3,3
0,2	$3,2^{+0,75}_{-0,2}$	3,6
0,4	$3,9^{+0,75}_{-0,9}$	4,0
0,6	$4,7^{+0,75}_{-1,2}$	5,0
0,8	$6,5^{+0,75}_{-1,5}$	6,2
1,0	$8,25^{+0,75}_{-1,5}$	8,2
1,2	$10,1^{+0,75}_{-1,5}$	9,4
1,4	$11,0^{+0,75}_{-1,5}$	11,4
1,6	$13,75^{+0,75}_{-1,35}$	13,6
1,8	$15,5^{+0,75}_{-1,35}$	15,8

ХАРАКТЕРИСТИКА $P_т = f(P_к)$

$P_к$, кгс/см²	Давление $P_т$, кгс/см²	
	расчетное	замеренное
2,0	$17^{+0,75}_{-1,5}$	16,8
2,2	$17,6^{+0,75}_{-1,5}$	17,2
2,4	$17,7 \pm 0,25$	17,6
2,5	$17,7 \pm 0,25$	17,6
3,0	$17,7 \pm 0,25$	17,7

Примечания: 1. Характеристику снимать на топливе Т-1 и масле МК-8 при $P_{вх}$ топлива 2 кгс/см², $P_{вых}$ масла $2,0^{+0,5}$ кгс/см², температуре рабочей жидкости (топлива и масла) плюс 25 ± 5°С и напряжении 27 В.

2. Точность замера давления ±1% от текущего значения.

Подпись ОТК

2. melléklet UART kommunikáció mintaprogram

A kód nyelve a C++.

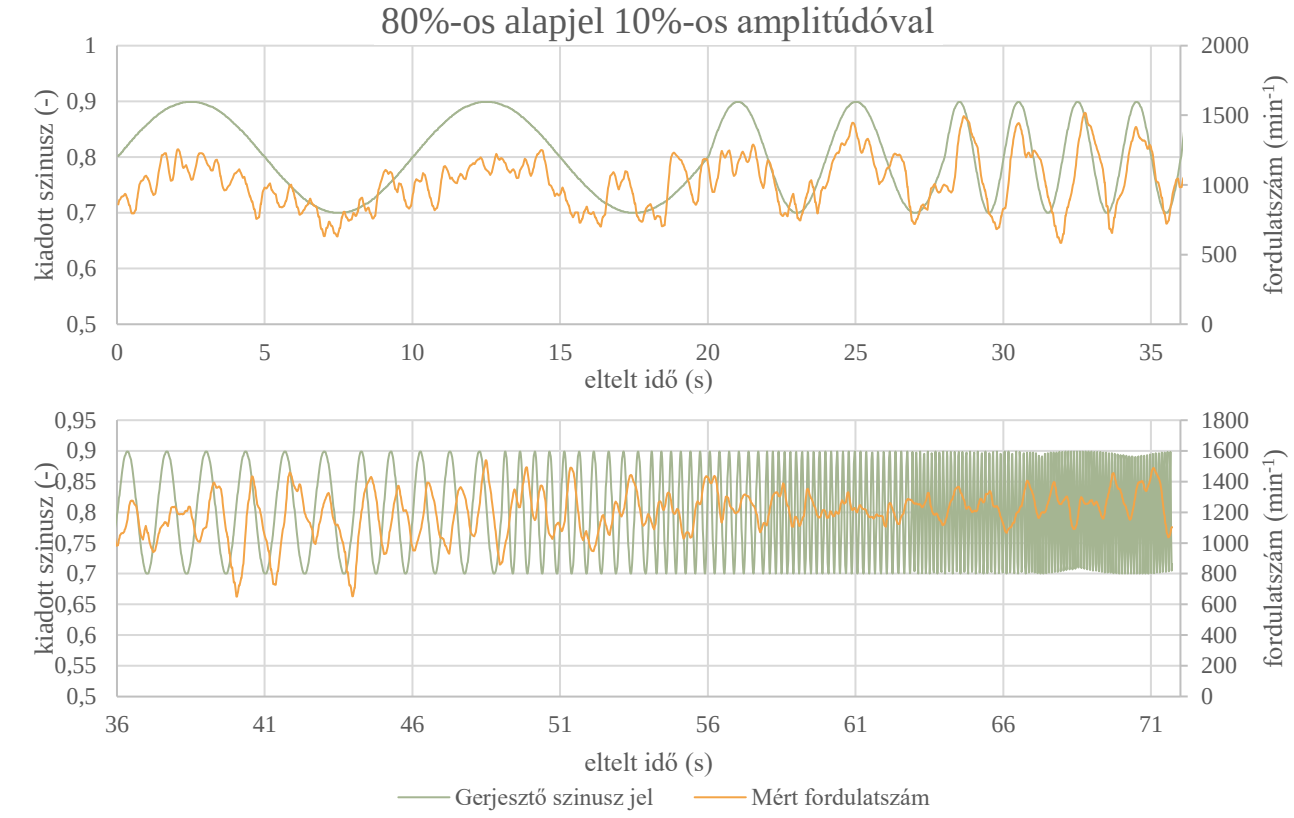
```
//A programmal a LED1 kimenetet tudjuk vezérelni 1-es gomb megnyomásával úgy, hogy a PC-től érkező karakter esetén fut csak le a kijelölt kódrészlet (interrupt)

#include „mbed.h”
RawSerial pc(USBTX, USBRX);
DigitalOut led(LED1);

void PCCommand()
{
        if (pc.readable() && pc.getc() == ’1’)
        {
                led = !led;
                pc.printf(„LED átállítva!\n”);
        }
}

int main()
{
        pc.baud(115200);
        pc.attach(&PCCommand, Serial::RxIrq);
        pc.printf(„Soros kommunikációs program\n1-es gomb megnyomására a LED1 állapotot vált\n”);
        while (true)
        {
                pc.printf(„Bemenetre várok\n”);
                wait_us(1e6);
        }
}
```

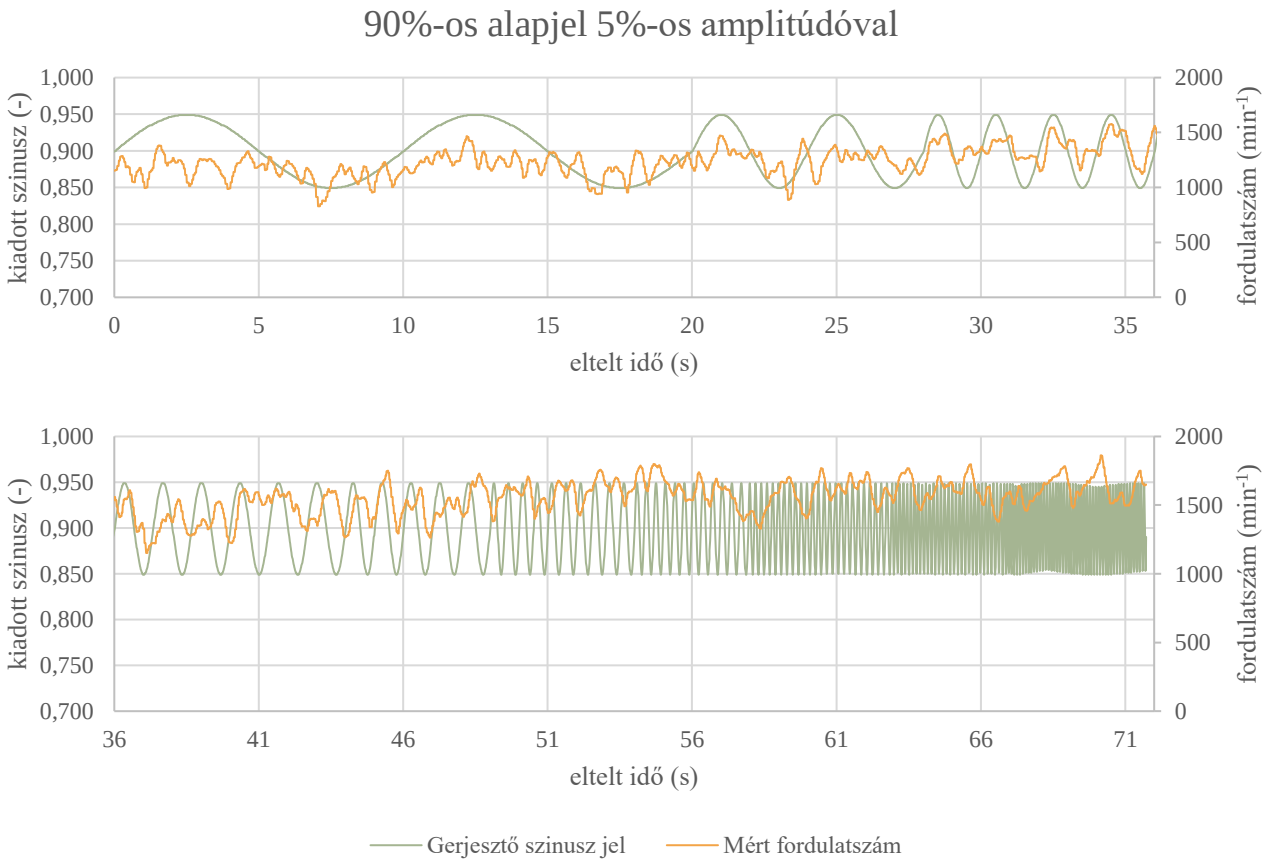
3. melléklet Analóg szinusz hullám létrehozása

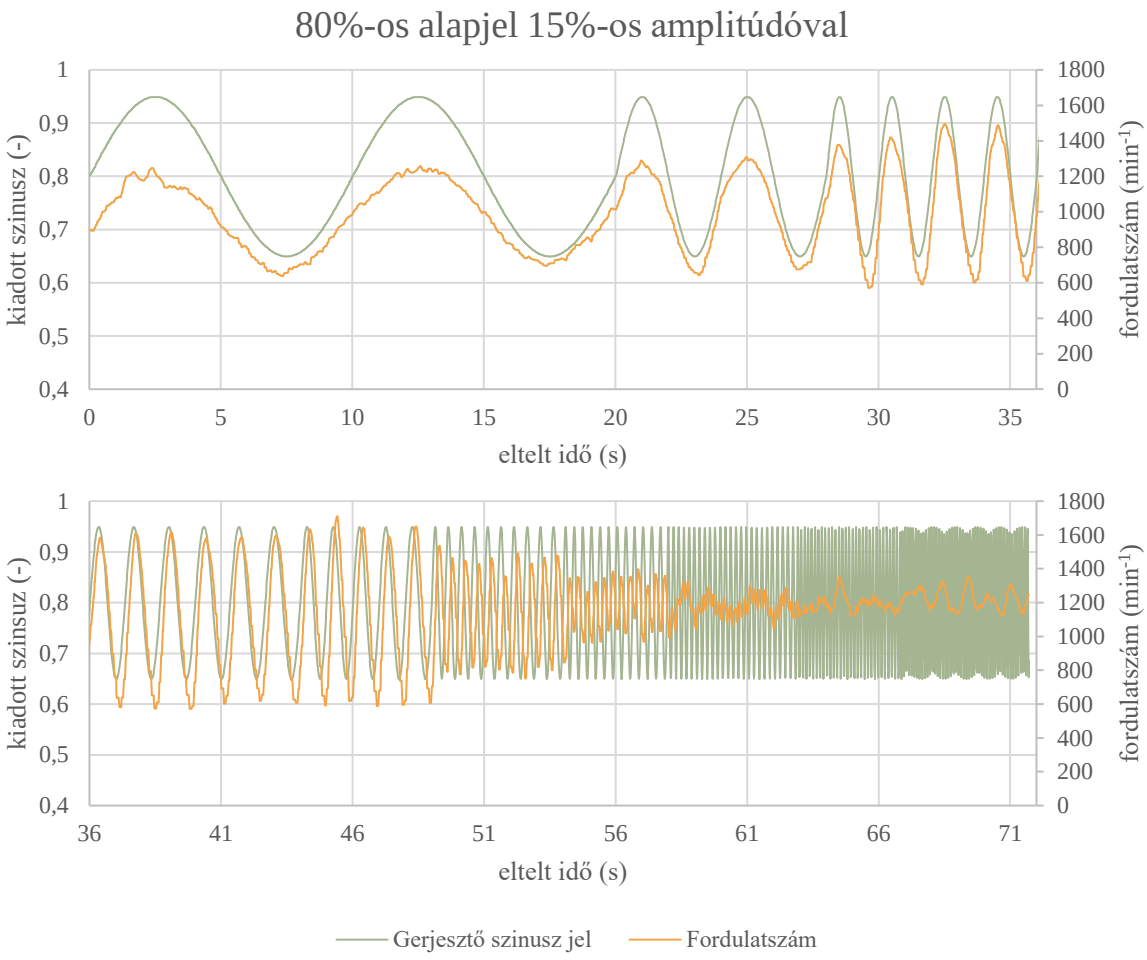
A kód nyelve a C++.

```
/*A mellékelt kód 0,5 Hz frekvenciájú, 5%-os amplitúdójú, 70% körül ingadozó szinuszjel
létrehozására alkalmas mbed LPC 1768 kontrolleren. Az alábbi kód teljes értékű, mbed Online
Compiler-ben azonnal futtatható.
Készült 2023-ban B3B*/

#include „mbed.h”

AnalogOut sine(p18);
Ticker sinetick;
RawSerial pc(USBTX, USBRX);

float rads = 0;
float amplitude = 0.05;
float offset = 0.7;
float freq = 0.5;
int samplerate = 1000;
const float pi = 3.141592653589793238462;
bool sine_wave_const = 0;

void TickStaticSine()
{
        sine_wave_const = 1;
        if (rads < ((2*pi)) / 0.5)
                rads += (2*pi)/0.5/200;
        else
                rads = 0;
}

int main()
{
        pc.printf(„Static sine wave program\nF = 0.5hz\nA = 0.05\nDel = 0.7\n”);
        sinetick.attach_us(&TickStaticSine, 1e6 * (1 / freq) / samplerate);
        while (true)
        {
                if (sine_wave_const)
                {
                        sine.write(((amplitude * (sin(rads * freq))) + offset));
                        pc.printf("%f\n", sine.read());
                        sine_wave_const = 0;
                }
        }
}
```

4. melléklet Három mérés mért adata az identifikáció során diagramokba szedve





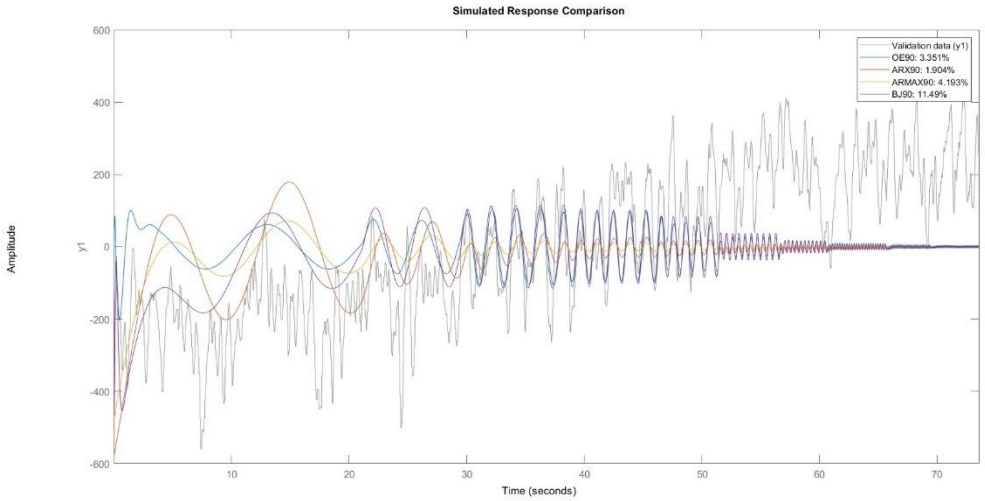

80%-os alapjel 15%-os amplitúdóval
kiadott szinusz (-)
fordulatszám (min-1)
eltelt idő (s)
1
0,9
0,8
0,7
0,6
0,5
0,4
1800
1600
1400
1200
1000
800
600
400
200
0
0
5
10
15
20
25
30
35
36
41
46
51
56
61
66
71
Gerjesztő szinusz jel
Fordulatszám

5. melléklet MATLAB identifikációs program

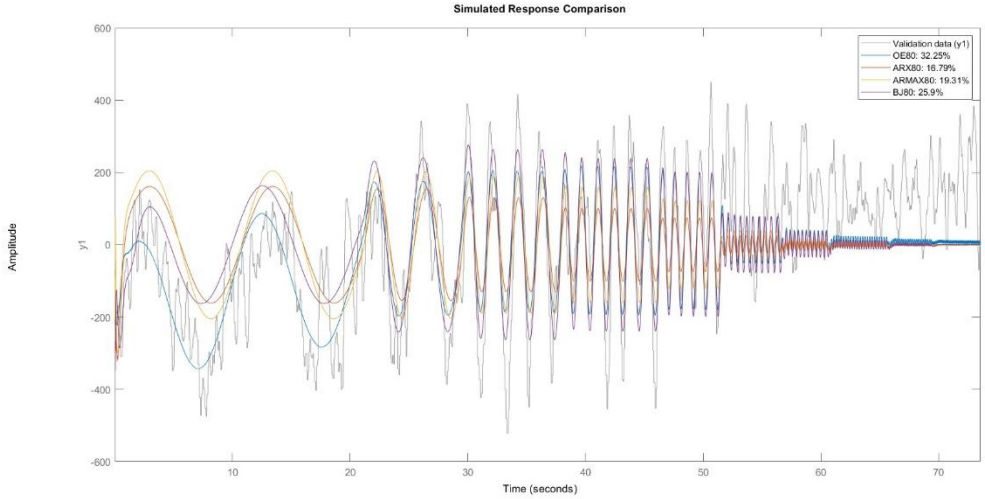
A kód nyelve a MATLAB.

```
% TDK Bánfi Bendegúz Balázs
% 2023

% Identifikáció 80%-os jel mellett 10%-os amplitúdóval
load('80_10.mat')
id80 = iddata(rpm80 - mean(rpm80), sine80 - mean(sine80), 12/1000)
OE80 = oe(id80, [3 4 2]);
compare(id80, OE80)
ARX80 = arx(id80, [7 1 0]);
compare(id80, ARX80)
ARMAX80 = armax(id80, [5 1 3 3]);
compare(id80, ARMAX80)
BJ80 = bj(id80, [5 1 3 3 4]);
compare(id80, BJ80)
compare(id80, OE80, ARX80, ARMAX80, BJ80)

%Identifikáció 90%-os jel mellett 5%-os amplitúdóval
load('90_05.mat');
id90 = iddata(rpm90 - mean(rpm90), sine90 - mean(sine90), 12/1000);
OE90 = oe(id90, [3 3 3]);
compare(id90, OE90)
ARX90 = arx(id90, [2 1 0]);
compare(id90, ARX90)
ARMAX90 = armax(id90, [1 1 4 1]);
compare(id90, ARMAX90)
BJ90 = bj(id90, [4 2 1 4 0]);
compare(id90, BJ90)
compare(id90, OE90, ARX90, ARMAX90, BJ90)

%Identifikáció 80%-os jel mellett 10%-os amplitúdóval, de most 5-ös mozgóátlaggal
load('80_10_mean5.mat');
id80_mean5 = iddata(rpm80mean5, sin80mean5, 12/1000)
OE80MEAN = oe(id80_mean5, [4 4 6]);
ARX80MEAN = arx(id80_mean5, [5 1 4]);
ARMAX80MEAN = armax(id80_mean5, [4 3 2 4]);
BJ80MEAN = bj(id80_mean5, [4 4 1 3 3]);
compare(id80_mean5, OE80MEAN, ARX80MEAN, ARMAX80MEAN, BJ80MEAN)

%Identifikáció 80% és A15% esetén 3 pontos mozgóátlaggal
load('80-15.mat');
id80_15 = iddata(rpm15mean, sin15mean, 12/1000);
OE15mean = oe(id80_15, [3 2 3]);
ARX15mean = arx(id80_15, [3 3 4]);
ARMAX15mean = armax(id80_15, [1 5 2 1]);
BJ15mean = bj(id80_15, [2 1 2 4 5]);
compare(id80_15,OE15mean, ARX15mean, ARMAX15mean, BJ15mean)
BJCont = d2c(BJ15mean, 'tustin')
BJSimp = tf([1.428e7 1.996e7], [1 42.42 821.8 5609 1.488e4])
nyquist(BJSimp)
bode(BJSimp)
```

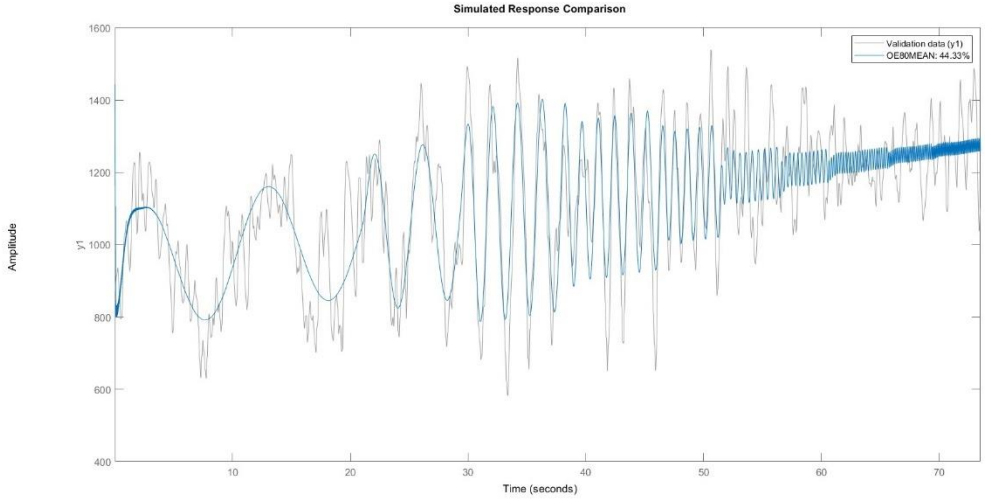
6. melléklet MATLAB-ban identifikált összes eredmény



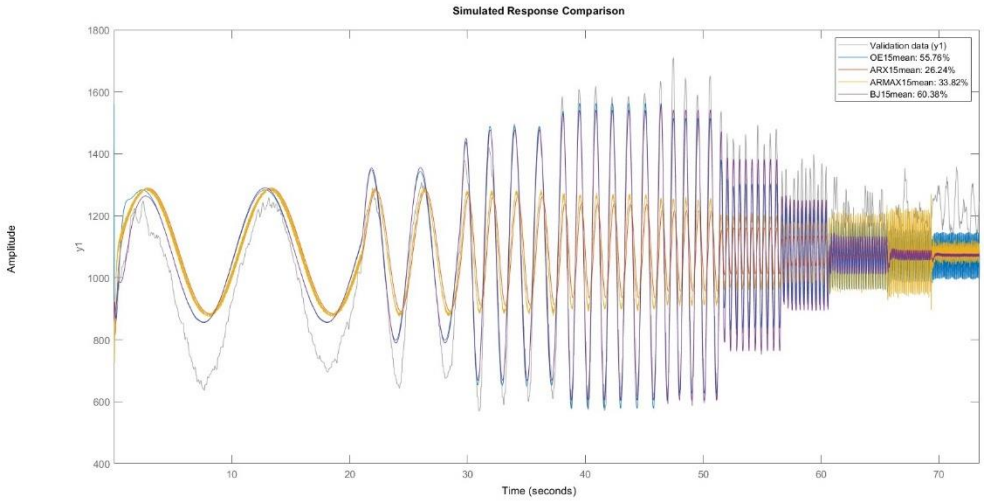
90%-os alapjel 5%-os amplitúdóval. Az egyes identifikációs eljárások megnevezései a jobb felső sarokban olvashatók.



80%-os alapjel 10%-os amplitúdóval. Az egyes identifikációs eljárások megnevezései a jobb felső sarokban olvashatók.



80%-os alapjel 10%-os amplitúdóval, de a fordulatszám 5 pontos mozgóátlaggal kezelve. Az egyes identifikációs eljárások megnevezései a jobb felső sarokban olvashatók.



80%-os alapjel 15%-os amplitúdóval, a fordulatszám 3 pontos mozgóátlaggal kezelve. Az egyes identifikációs eljárások megnevezései a jobb felső sarokban olvashatók. Ezt az identifikációt használtuk a szabályozó megtervezéséhez.

7. melléklet A program verzióinak listája

BETA

- β1.0 PWM generálás és négyszögjel beolvasása
- β1.1 Négyszögjel két bemenete között eltelt idő utáni kiolvasása
- β1.2 Analóg kimenet potenciométer alapján
- β1.3 Relé állapotának változtatása
- β2.0 Soros kommunikáció egyetlen karakterrel
- β2.1 Soros kommunikáció parancssorral
- β2.2 Rendes parancsok definiálása és ráncfelvarrás
- β2.3 HX711 erőmérő cella beépítése
- β2.4 PC-ről adatbekérés int formátumban, maximális karakterlánc-hiba kiküszöbölése
- β2.5 Bluetooth-kapcsolat két irányba, debug mode implementálása Bluetooth-irányba, soros kapcsolat Androiddal
- β3.0 Szinusz jel generálása
- β3.1 LabVIEW kommunikáció fejlesztése, új típusú (|) parancsok definiálása, kérdés-válasz párok beépítése
- β3.2 Egyre növekvő frekvenciájú szinuszjel generálása, parancsok létrehozása
- β3.3 Első mérési iteráció, mérésbe való beépítés
- β3.4 Azonos kijelzési időközök használata az identifikációhoz
- β3.5 Első identifikációs kísérlet
- β4.0 Zárt hurkú szabályozás próbája
- β4.1 Fordulatszám-mérő belső problémáinak kiküszöbölése, timeout beépítése
- β4.2 Apró módosítások a zárt hurkú szabályozásban
- β4.3 Két típusú zárt hurkú szabályozó implementálása
- β 4.4 Valódi 100 ms-os mintavételezési idejű szabályozó működőképessége

RELEASE

- 1.0 A szabályozó letesztelt állapota