



ÓBUDAI EGYETEM

Kandó Kálmán Villamosmérnöki Kar

Műszertechnikai és Automatizálási Intézet

Diluter tesztelő mérőrendszer

Bertók Zoltán

Tartalomjegyzék

1	Bevezetés	4
1.1	Pipettázás	4
1.2	Controller Area Network.....	6
1.3	Képfeldolgozás.....	8
1.4	TECAN cég által alkalmazott visszamérés, kalibrálás.....	10
1.5	A projekt megvalósításának célja, lehetséges alkalmazási területei	10
2	A feladat elemzése	12
2.1	Diluter műszaki leírása.....	12
2.2	Felhasználandó eszközök	13
3	Rendszertervezés.....	14
3.1	Mechanikai rendszerterv	14
3.2	Villamos terv.....	16
3.3	Program terv.....	18
4	Program kidolgozása	21
4.1	Driverrek kiválasztása	21
4.2	Felhasználói felület	21
4.2.1	Main panel.....	22
4.2.2	CAN settings	23
4.2.3	Serial settings panel.....	23
4.2.4	Calibration document settings panel	24
4.2.5	Allowed error panel.....	25
4.3	Részletes program leírás	26
4.3.1	CAN busz inicializálása	26
4.3.2	Diluter bekapcsolása	26
4.3.3	Diluter címének meghatározása	28
4.3.4	Nyugtázó üzenet küldése.....	29
4.3.5	Firmware parancsokat tartalmazó fájlok	29
4.3.6	Üzenetek beolvasása	32
4.3.7	Tesztelési lépések mérési eredményeinek tárolása	35
4.3.8	Fényforrások be- és kikapcsolása.....	36
4.3.9	Szelep tengely optikai vizsgálata	37
4.3.10	Fecskendőmozgató kar optikai vizsgálata.....	41

4.3.11	Sebességmérés.....	44
4.3.12	Mérési eredmények kiértékelése	45
4.3.13	A kalibrációt végrehajtó cég adatainak beállítása a jegyzőkönyvben	48
4.3.14	Külső program futtatása	49
4.4	Kiegészítő programok	50
4.4.1	PDF report készítése	50
4.4.2	Áram/feszültségmérés, LED villogtatás.....	53
5	Összefoglalás, elemzés.....	56
6	Abstract	57
7	Irodalomjegyzék.....	58
8	Mellékletek.....	59

1 Bevezetés

A dolgozatom címében szereplő készülék a TECAN cég folyadékszallító automatáinak központi eleme. Tekintettel arra, hogy ez a tesztelni kívánt berendezés pipettázási feladatokat lát el, a vezérléséhez felhasznált kommunikációs protokoll a Controller Area Network (CAN), valamint mérőrendszer egy optikai elven működő vizsgáló állomás, ezért a bevezetésben e három témát ismertetem röviden.

1.1 Pipettázás

A pipettázás meghatározott mennyiségű folyadék mozgását jelenti. A pipettákat kémia, biológiai és gyógyszerészeti laboratóriumokban használják. A vizsgálatok vagy tesztek során nem csak minták, hanem reagensek és adalékanyagok adagolására is szükség van [1]. Ezen folyamatok eredményességének, pontosságának és precizitásának növelésére számos készüléket, valamint pipettázási módszert fejlesztettek ki.

A pipetták igen sok formában fordulnak elő és felhasználásuk is különféle lehet. Mikro pipettáknak nevezik azokat az eszközöket, amelyek 1 és 1000 μl közötti térfogat kimérésére alkalmasak, míg makro pipettáknak amelyeket 1 ml feletti adagolásra használnak. A pipetták főbb típusai: egy-, illetve két-körjeles hasas-, valamint osztott- és mikro-pipetták [3].

A kézi pipettázás hátrányos tulajdonsága, hogy sok részfeladat és lépés végrehajtását igényli, valamint, hogy az adagolások pontossága és precizitása függ a kezelő személy képességeitől [1].

A reprodukálhatóság, az emberi beavatkozásokból származó hibák csökkentése, valamint a teljes folyamatok automatizálhatósága vezetett a folyadékkezelő rendszerek kifejlesztéséhez. Napjainkban ezen eszközöket többféleképpen nevezik: pipettázó automata, folyadékkezelő rendszer, folyadékkezelő robot, vagy egyszerűen csak folyadékkezelő.

Ezeket a rendszereket biológiai laboratóriumokban az 1980-as években kezdték el alkalmazni. Számos előnyös tulajdonságainak köszönhetően ma már kémiai, biokémiai és gyógyszeripari laborokban is használatosak [2].

Működési mechanizmus szerint három nagy csoportba sorolhatóak attól függően, hogy milyen módon állítják elő a felszíváshoz és kiadagoláshoz szükséges nyomáskülönbséget [2]:

- Folyadékkezelés dugattyús pumpával (Liquid Handlers with Piston Pump)
- Folyadékkezelés vákuum vagy levegő pumpával (Liquid Handlers with Vacuum or Air Pump)
- Folyadékkezelés perisztaltikus vagy fogaskerekes pumpával (Liquid Handlers with Flow Through Pump) [1] sz. melléklet

Mivel a dolgozatomban szereplő diluter egy dugattyús pumpának felel meg, ezért csak ennek a bemutatására térek ki.

A dugattyúpumpás rendszereket piston rendszereknek is nevezik. A dugattyú tökéletesen illeszkedik a légmenetesen zárt cső falához. A dugattyú mozgása teremti meg a felszíváshoz a negatív, míg kiadagoláshoz a pozitív nyomást. Az automatizált folyadékkezelő rendszerekben a dugattyús pumpának többféle megvalósítása létezik.

A dugattyú vagy közvetlenül, vagy egy hosszabb csővezetéken keresztül csatlakozik a pipetta hegyéhez, amelyet valamilyen folyadék tölt ki. Ez utóbbi a folyadék-elmozdulású (liquid displacement system) vagy hidraulikus elmozdulású rendszer (hydraulic displacement system) [2]. A TECAN folyadék-elmozdulású pipettázó rendszereket gyárt.

A TECAN berendezéseinek piston pumpáin, vagyis a dilutereken két mozgatott alkatrész van, egy háromállású szelep és egy fecskendő (dugattyú). A szeleppel lehet állítani a diluterrel mozgatott folyadék irányát, míg a fecskendővel történik a folyadékmozgatás. (A készülék CAN buszon csatlakozik az automata vezérlőjéhez.)

1.2 Controller Area Network

A Controller Area Network (CAN) protokollt a német autóipari beszállító, a Robert Bosch GmbH fejlesztette ki az 1980-as évek közepén. A megalkotásának célja egy olyan soros kommunikációs protokoll létrehozása volt, amellyel az autóiparban addig alkalmazott nagyméretű és drága kábelkorfacsokat nagyobb megbízhatóságú, biztonságos, kevesebb vezetékvezést igénylő hálózatra lehet cserélni. A 80-as évek után a protokollt széles körben kezdték használni, főként az ipari alkalmazásokban, az autóiparban, tehergépjármű gyártásban [4]. Ma már többek között mezőgazdasági gép gyártók és hajógyárak is alkalmazzák, valamint orvosi eszközökben, textilgyártásban és liftek vezérlésében is megtalálható [5].

A legtöbb hálózati alkalmazás követi a réteges közelítésű rendszer megvalósítást. Ez a szisztematikus megközelítés lehetővé teszi a különböző gyártójú termékek közötti együttműködést. Nemzetközi Szabványügyi Hatóság (International Standards Organization - ISO) létrehozott egy szabványt, amely egy sablonként szolgál ehhez a réteges közelítés követéséhez. Ezt OSI (Open Systems Interconnection) modellnek hívjuk [4]. Az OSI modell hét rétegből áll. Kialakításánál azt vették figyelembe, hogy minden réteg feladata jól meghatározott, a rétegek közötti kommunikáció pedig minimális legyen, valamint azt, hogy elegendő számú réteg legyen kialakítva ahhoz, hogy különböző funkciók ne kerüljenek azonos egységbe. Ezek szerint a modell a következő elemekből épül fel: fizikai réteg, adatkapcsolati réteg, hálózati réteg, szállítási réteg, együttműködési réteg, megjelenítési réteg, alkalmazási réteg [6].

Számos kommunikációs protokoll nem valósítja meg az OSI modell mind a hét rétegét, mivel a legtöbb zárt hálózatok esetében azok közül számos funkcióra nincs szükség, mint pl. az adatok megjelenítésére, bejelentkezésre, stb. Ebből kifolyólag a CAN protokoll csak két réteget implementál, a fizikai és az adatkapcsolati réteget [7].

A fizikai rétegen biztosítja a fizikai kapcsolatot a csomópontok között és a vevő által értelmezhető adatokat juttat a kommunikációs csatornára. Ehhez a réteghez tartozik az adatkapcsolat átviteli sebesség, jel szintek, az alkalmazott csatlakozó fizikai, mechanikai kialakításának meghatározása stb. [6].

Az adatkapcsolati réteg két alrétegre tagolódik: a tárgy rétegre (object layer) és az átviteli rétegre (transfer layer). A tárgy réteg végzi az üzenetek szűrését, állapítja meg, hogy melyik üzenetet kell elküldeni, dönti el, hogy mely fogadott üzeneteket kell

felhasználni, valamint ez a réteg teremt kapcsolatot az alkalmazási réteghez kötött eszközökkel. Az átviteli réteg hajtja végre az arbitrációt, ellenőrzi a kereteket, valamint végzi a hibafigyelést és többszörös hiba fellépése esetén a kommunikációból való kizárást, stb. [8].

A protokoll használatakor a csomópontok (vezérlők, érzékelők, beavatkozók, stb.) soros buszon keresztül vannak összekapcsolva. A busz egy szimmetrikus kétvezetékes áramkör, a két végén $120\ \Omega$ lezárásokkal. A busz elektromos, valamint fizikai paramétereit az ISO 11898-as szabvány tartalmazza [5].

A CAN protokoll egy üzenet alapú kommunikáció, nem címzés alapú, vagyis adatátvitelkor nincs megcímzett állomás. Minden eszköz fogadja a buszra küldött üzeneteket, majd visszaigazolást ad a beolvasás sikerességéről, ez követően döntenek csak el a csomópontok – az üzenet azonosító értékétől függően –, hogy figyelmen kívül hagyják, vagy feldolgozzák-e azokat [4]. Az azonosító nem csak az üzenet tartalmát definiálja, hanem a prioritását is. Erre a busz hozzáférés megállapításakor van szükség, a busz hozzáférés jogát az az állomás kapja meg, amelyik nagyobb prioritású – a gyakorlatban kisebb értékű – azonosítóval rendelkezik. Az üzenetorientált címzési sémának köszönhetően rugalmasan bővíthető a rendszer [5].

A CAN protokoll összesen két keretformátum típust támogat. Az egyik a standard formátumú üzenetek, amelyek 11 bites azonosítóval (ID) rendelkeznek, míg a másik a kiterjesztett üzenet formátum, ezek 29 bites azonosítóval rendelkeznek. Lényegi különbség köztük csak az azonosító hosszában van. A dolgozatban tesztelni kívánt eszköz a standard formátumot használja [5].

A standard formátumú üzenetek SOF (Start Of Frame) bittel kezdődnek, ezt követi az azonosító mező (arbitration field), amely utolsó bitje az RTR (Remote Transmission Request) bit, ez jelzi, hogy az adott keret adatot tartalmazó keret, vagy pedig üres, adatot nem tartalmazó kérés keret-e. Ezt követi a Control Field, amely tartalmazza az IDE (IDentifier Extension) bitet, ez határozza meg az üzenet formátumát. A vezérlő mező része még a DLC (Data Length Code), amely 4 bit hosszúságú, ezzel határozható meg, hogy hány bájtot tartalmaz az üzenet (maximálisan 8 bájt). A következő rész maga az adatmező (Data Field). Ezt követi a CRC mező (Cyclic Redundancy Check), amely a kerethibák ellenőrzésére szolgál. Az ACK field-nek (nyugtázó mező) két része van, az egyik ACK slot és az ACK delimiter, melyek 1-1 bit hosszúságúak. Mindkét bit recesszív (logikai nulla értékű) bitként kerül kiküldésre,

viszont ha az üzenetet bármelyik vevő hibátlannak találja, akkor felülírja ACK slot értékét. Az üzenet az End of Frame mezővel zárul [8].

A CAN protokoll szerint öt szabványos üzenetkeretet különböztetünk meg: normál, kibővített, kérés, hiba és túlterhelés üzenetkereteket [9].

1.3 Képfeldolgozás

A képfeldolgozás a jelfeldolgozás egy speciális változata. Elméletileg a képfeldolgozásba mind az analóg, mind a digitális jeleken végzett vizsgálatok beletartoznak, azonban ma már szinte csak az utóbbit értjük ez alatt.

A képfeldolgozás tehát olyan vizuális, azaz érintés nélküli számítástechnikai mérés, amely által bepillantást nyerhetünk abba, hogy a vizsgált kép milyen tárgyakat tartalmaz, a tárgyakon méréseket lehet végezni a különböző képfeldolgozási műveletek segítségével, valamint vizsgálható bizonyos (előre meghatározott) elemek jelenléte is a képen. [10]

A képfeldolgozás használata napjainkban már széles körben elterjedt, éppúgy alkalmazzák az ipari tömegtermelésben, mint a meteorológiában a műholdak felvételeinek elemzésére, az orvostechikában pl. CT koponya felvételek kiértékelésére [11], vagy akár a mezőgazdasági kísérletek értékelésében is [12].

A vizuális méréseknek számos előnyük van: a gyártási folyamat minden fázisában alkalmazhatók (természetesen csak ott, ahol vizuális információk alapján kell döntést hozni), szigorú minőségügyi és minőségbiztosítási vizsgálatokat is képesek ellátni [13], zord ipari környezetben is alkalmazhatóak, nagyobb megbízhatóságot nyújtanak, mint emberi megfigyeléssel végzett vizsgálatok.

A tömeggyártásban alkalmazott kamerás minőségellenőrző rendszerek segítségével többek között *alakzat-ellenőrzés, detektálás, nyomatellenőrzés, mintaillesztés, színellenőrzés, fényesség-ellenőrzés, felület-ellenőrzés, pozicionálás, hajlásszög-ellenőrzés, forrasztás-, ponthegesztés ellenőrzés, termékválogatás* műveleteket lehet elvégezni [14]

Pályamunkámban a képfeldolgozás megvalósítása a National Instruments (NI) szoftvereinek felhasználásával történik.

Az NI a képfeldolgozási algoritmusok létrehozására kétféle szoftvert fejlesztett ki, az egyik a *Vision Builder*, a másik pedig a *Vision Assistant*, valamint számos illesztő szoftver (driver) létezik a különböző interfésszel rendelkező kamerákkal való kommunikációra, ilyenek pl. a IMAQ for USB, IMAQ PCI-1428, IMAQdx, stb.

1.4 TECAN cég által alkalmazott visszamérés, kalibrálás

A Tecan cég a pipettázó automaták karbantartásához, teszteléséhez, valamint hiba kereséséhez az Instrument Software, vagy más néven Setup&Service szoftvert alkalmazza.

A programban a megfelelő működés ellenőrzésén túl lehetőség van a kiadagolási pontosság minősítésére, vagyis a diluterek, és a rendszerben kialakított csővezetékek együttes kalibrálására is.

Ezen mérések menete a következő [15]:

- be kell állítani a pipettázáshoz használt hegyek típusát (fix hegy, eldobható hegy, stb.)
- a készülék munkaasztalára el kell helyezni egy folyadéktárolót (amelyből történik a felszívás), valamint egy analitikai mérleget
- a minta mérlegre való kiadagolását követően a szervíz szoftver a mért tömegből visszaszámolja annak térfogatát (a mozgatót folyadék desztillált víz)
- a mérés minden egyes hegyre, így diluterre is elvégzésre kerül
- a mérési eredményekből a pontosság- és stabilitásértékeknek kiszámolásra kerülnek

Jelenleg ezen mérések a Mettler cég által gyártott AG 285 és AG 135 típusú mérlegekkel végezhetőek el [16].

A mért értékek nem csupán a dilutert jellemzik, hanem az azt követő minden egyes rendszerben lévő elemet, így a fecskendőt, a szelepet, pipettázó csővezést, a hegyet és mindezek egymáshoz való csatlakozását.

Az [2] sz. mellékletben egy ilyen kalibrálás közben készült kép látható.

1.5 A projekt megvalósításának célja, lehetséges alkalmazási területei

Pályamunkám célja egy olyan optikai elven működő vizsgáló állomás létrehozása volt, amellyel a címben szereplő készülék teljes körű, gyors, és felhasználói beavatkozást nem igénylő tesztelése végezhető el a készülékház megbontása nélkül.

Az általam tervezett mérőrendszer felhasználási helye kötetlen és nem igényli a pipettázó rendszer többi elemének meglétét. A mérések (amennyiben USB-8472 használunk) elvégezhetőek a pipettázó automatával azonos helyen is, de elsősorban nem ez a felhasználás célja, hanem a gyártás végi, illetve a javítás utáni ellenőrzés.

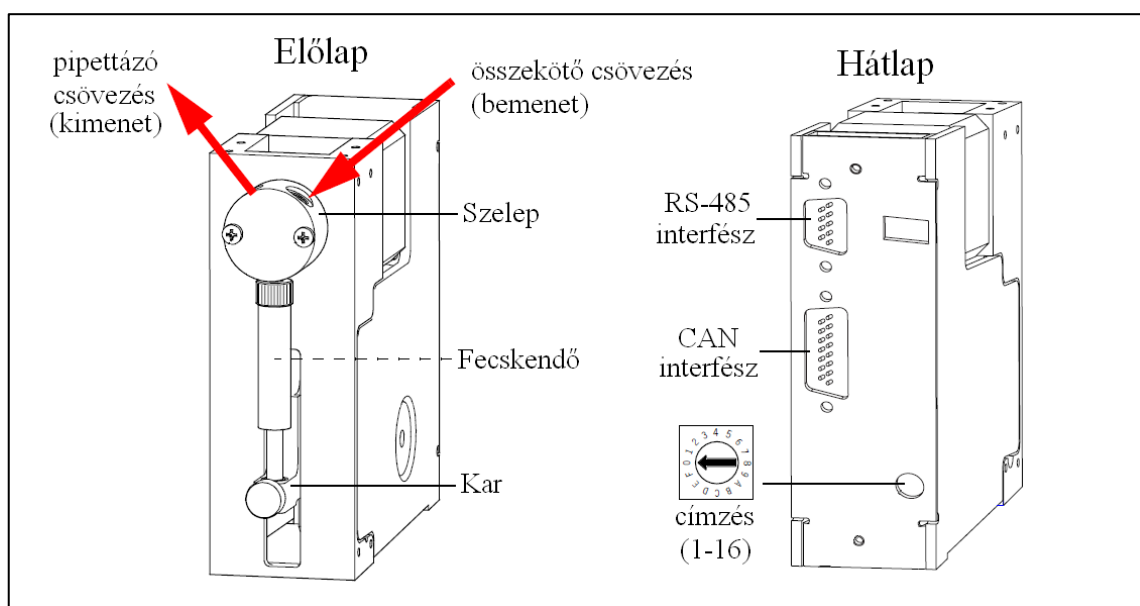
Természetesen a mérőrendszerrel elvégzett mérések csak a diluter megfelelő működéséről szolgáltatnak információt. A teszteken átment készülék egy pipettázó rendszerbe való beszerelését követően nem garantált az adott csatorna pontos kiadagolása, hiszen azt a diluteren kívül számos tényező befolyásolja (csővezeték, csatlakozások, stb. Éppen ezért az általam készített vizsgáló állomás nem helyettesíti az előző fejezetben leírt mérőrendszer használatát.

2 A feladat elemzése

2.1 Diluter műszaki leírása

A diluter tápfeszültsége 24 V, áramfelvétele mindkét léptetőmotor egyidejű mozgása esetén 500 mA körül mérhető.

Az újabb típusok (VCC, smart) nem csak CAN busz, hanem RS-485-ös interfésszel is rendelkeznek. Pályamunkámban csak a CAN buszon történő kommunikáció lesz megvalósítva.



1. ábra Diluter (XP3000plus) modul előlap és hátoldal

A CAN busz alacsony adatátviteli sebességű, 100 kbit/s-os, az üzenetek standard formátumúak (CAN 2.0A).

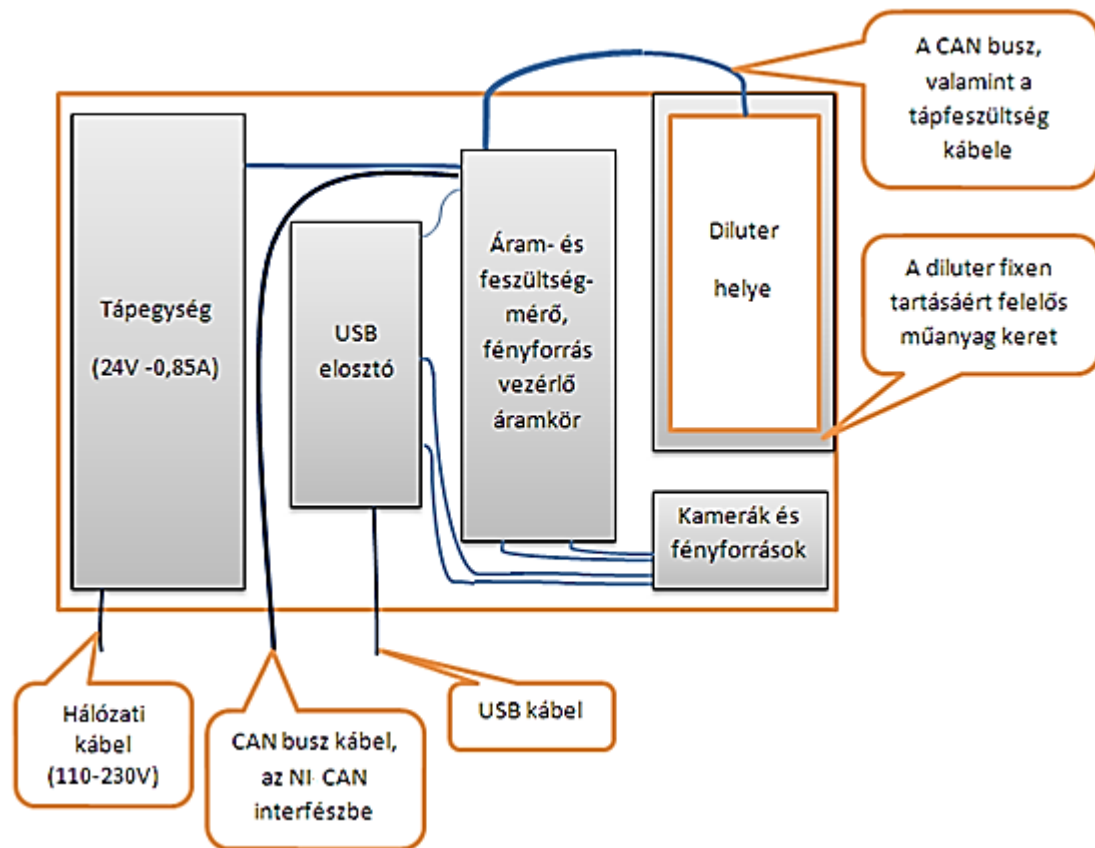
A készüléken két mozgó alkatész van. A szelep, amely tengelye üzemszerűen 360° körülfordulásra képes. Ennek három pozícióba való beállítását kell vizsgálni. A másik alkatrész a szelepmozgató kar, amelynek teljes hasznos úthossza 31,5 mm. A diluterekben felhasznált léptető motorok egy lépést 1,8°-os elfordulással tesznek meg, mivel a kar teljes úthosszán való mozgatásához a tengely 17 és fél körülfordulása szükséges, így belátható, hogy a kar 10 µm-nél kisebb pontossággal képes bárhová beállni (a motor a microstepping funkcióval ellátott vezérlőjének köszönhetően ennél pontosabb mozgásra is képes lenne). Ez azt jelenti, hogy a pontos visszamérés érdekében legalább ekkora (16:9 képarány esetén) 5,58 Mpixel felbontású USB kamerát kell felhasználni a pozíció meghatározásához.

2.2 Felhasználandó eszközök

- PXI-8511, későbbi felhasználás esetén USB-8472 interfész
 - A mérőrendszer a hordozhatóság érdekében USB CAN interfésszel lesz a későbbiekben megvalósítva. Mivel Egyetemünk rendelkezik PXI-os kártyával, így kézenfekvő volt a fejlesztést ezen megvalósítani, hiszen mindkét eszköz támogatja az NI-CAN drivert. Vagyis a későbbi hardver váltás nem vonja maga után a szoftver teljes átalakítását. Azonban a tervezéskor figyelembe kellett venni, hogy csak olyan függvények kerüljenek felhasználásra, amelyeket mindkét készülék támogat.
- 2 db USB kamera
 - A készülékek kiválasztásánál az alacsony ár játszott szerepet. Természetesen minél nagyobb felbontású kamerát választunk ki erre a feladatra, annál pontosabb mérések végezhetőek el a rendszerrel.
 - A felhasználásra kerülő kamerák típusa: *Emtec snake*.
- a fényképezendő felületek megvilágítását vezérlő áramkör
 - Erre a kiegészítőre azért van szükség, hogy a mérést ne befolyásolják a mérőrendszer környezetében lévő fényhatások.
- a tesztelés "Plug&Play" megvalósítására alkalmas mechanikai rendszer
 - A bevezetésben olvashattunk már arról, hogy a mérőrendszerrel szemben támasztott egyik követelmény, hogy a vizsgálatok úgy történjenek meg, hogy közben ne legyen szükség a készülékház megbontására. A mechanikai rendszer is hasonló célt szolgál, vagyis, hogy ne kelljen semmilyen rögzítést és egyéb kiegészítőt a diluterre szerelni a méréshez. Hanem a készüléket egyszerűen a mérőrendszer munkaasztalára helyezve és a CAN buszra csatlakoztatva már indítható legyen a tesztelés.
- USB elosztó
 - Erre azért van szükség, mert a PXI keretben lévő számítógép modul csak kettő darab USB porttal rendelkezik, a mérőrendszer pedig legalább három csatlakozási lehetőséget igényel (két webkamera, egy LED vezérlő áramkör).

3 Rendszertervezés

A felhasználandó eszközök az 2. ábrán látható módon kerülnek összeépítésre (felülnézeti kép).



2. ábra A mérőrendszer vázlatos felépítése

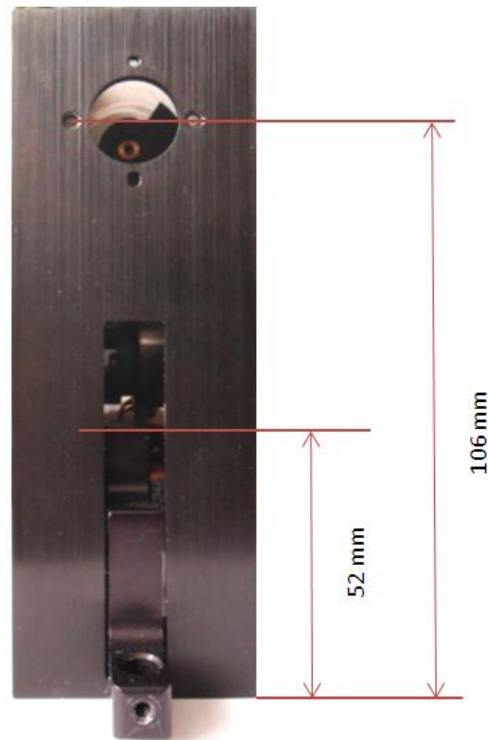
3.1 Mechanikai rendszerterv

Ebben az alfejezetben úgy gondolom, elegendő csak a kamerák elhelyezését részletesen bemutatni.

A mérőrendszer minden eleme egy 19x25 cm-es 2 mm vastagságú alumínium lapra van rászerelve. Azért ilyen anyagból készül a váz, mert ennek jó mechanikai tartása van, könnyen megmunkálható és amint a későbbiekben majd fogjuk látni, hő elvezetésére is alkalmas.

Az alaplapon "L" alakban meghajlított lemezre kerülnek a kamerák és a fényforrások. A kamerák úgy kerülnek felszerelésre erre a lapra, hogy az eszközökkel készíthető fotó

középpontja azonos magasságba essen a vizsgálni kívánt felület középpontjával, ezzel elkerülve a kamerák síkra nem merőleges felszerelésből adódó perspektivikus hibákat. Vagyis a szelep mozgását vizsgáló kamera a tengellyel azonos magasságban, míg a fecskendőmozgató kar helyzetét vizsgáló kamera a kar által bejárható teljes úthossz felével azonos magasságban kerül felhelyezésre (3. ábra).



3. ábra A kamerák felszerelési magasságai a diluteren ábrázolva

3.2 Villamos terv

Ahogy az már a felhasználandó eszközök címszó alatt is említésre kerül, a mérőrendszerben helyet kap a fényforrás be- és kikapcsolását vezérlő áramkör. Az áramkör ezen felül méri a diluter által felvett áramot, illetve a készüléken eső feszültséget is.

Az áramkörtől a mért villamos paraméterek bármikor lekérdezhetőek, így megjeleníthetőek LabVIEW-ban. Ezen felül a kontrollerben be lenne állítva a feszültségnek és áramnak egy határérték, amelyet meghaladva a dilutert ki kell kapcsolnia. Ez az érték a tápfeszültség esetén 26 V, míg a felvett áram esetén 700 mA. A készülék maximálisan 500 mA áramot vesz fel a tápegységtől (ez akkor fordul elő, mikor mindkét léptetőmotor vezérelve van, jellemzően inicializáláskor).

Természetesen a diluter számítógépről is ki- bekapcsolható, ezen a funkciók használatára a tesztelés elején illetve végén kerül sor. A fényforrások be- és kikapcsolása LabVIEW-ból történik.

Az áramkör és a számítógép közötti kommunikáció kialakítására számos lehetőség kínálkozott, az egyetlen szempont a tervezéskor az volt, hogy a későbbiekben a mérőrendszer laptopról is működtethető legyen.

A kontroller kiválasztásának szempontja az volt, hogy legalább 2 db A/D átalakítóval és 3 db I/O lábbal rendelkezzen, valamint CAN, vagy soros protokoll valamelyikét támogassa. CAN buszon történő kommunikáció kézenfekvő lett volna, hiszen ez egyébként is alkalmazva van a mérőrendszerben, sajnos azonban a legtöbb ilyen vezérlő nagyon nagy lábszámmal rendelkezik, használatuk indokolatlan lett volna, ellenben a soros kommunikációt támogató alacsony lábszámú eszközökben nagy a választék.

A felhasználásra kerül mikrokontrollert a Microchip cég gyártja, típusa: PIC16F1823, 12db I/O lábbal rendelkezik, ebből 8 db (10-bites) A/D konverterként is felhasználható.

A fényforrás (CREE XR E típusú) magas fényerejű LED-ekből áll, összesen 4 db-ból, amelyek kettésével kapcsolhatóak be. Maximális nyitóirányú áramuk 1 A.

A fényforrások tápellátását egy MAX5035 típusú Step-down konverter biztosítja, amely széles bemeneti feszültség tartománnyal (76 V) rendelkezik, így közvetlenül a diluter 24 V-os tápegységére köthető. A konverter áramgenerátorként van felhasználva, mivel

”feedback” lába nem közvetlenül a kimenetre van kötve, hanem egy feszültségosztón keresztül mindig az éppen bekapcsolt LED-en átfolyó árammal arányos feszültségre. (Az áramgenerátor kapcsolási rajza a [4] sz. mellékletben található.) A LED nyitóirányú árama 300 mA-re van beállítva.

A fényforrások az áramgenerátor kimenetére kapcsolását egy-egy kapcsolóüzemben vezérelt bipoláris tranzisztor látja el. Itt BDP949 típusú alkatrészeket használtam fel. Ezen típusok kollektor-emitter feszültsége maximálisan 60 V lehet, míg kollektor árama 3 A. A teljes disszipációs teljesítmény értéke 3 W. Kapcsolóüzemben vezérelve a tranzisztort a kollektor-emitter közötti szaturációs feszültség értéke 0,05 V-ra adódik. Vagyis a tranzisztoron 0,015 W alakul hővé a LED-en átfolyó áram hatására.

USB-soros átalakító kiválasztásánál az volt az egyik legfőbb szempont, hogy minél alacsonyabb legyen a külső alkatrészigénye. A választás az FTDI chipre esett, ennél az integrált áramkörnél ugyanis a tápszűrő alkatrészeket kívül 1 db alkatész beépítése szükséges. (Az integrált áramkört nem kell felprogramozni és a TX/RX kommunikáció működése követhető az átalakítóra kötött LED-eknek köszönhetően.) A kapcsolási rajz a [3], az alkatrészlista a [5] és a nyomtatott áramköri és beültetési rajz az [6] sz. mellékletben látható.

A tápellátás egy 160x50x35 mm nagyságú műanyag dobozban lévő 24 V-os kimenetű kapcsolóüzemű tápegység látja el. Erre az áramforrásra a diluter, illetve az áramgenerátor csatlakozik, míg a mikrokontroller, az USB-soros átalakító és az áramméréshez szükséges elektronika az USB port 5 V-os feszültségéről működik.

A mérési lépések számától függően a dilutert megvilágító LED-ek akár percekig is be lehetnek kapcsolva, 300 mA-es átfolyó áram miatt már gondoskodni kellett a hűtésükről is. Ezért mind a négy fényemittáló dióda tokja egy rézlapra van forrasztva, majd ez a rézlemez az előző fejezetben leírt módon a függőleges helyzetű alumínium lapra van rászerezelve.

3.3 Program terv

A mérőrendszer megvalósításához három különböző programozási nyelvben történő programozás volt szükséges. Természetesen a szoftver legnagyobb része LabVIEW-ban készült, viszont szükség volt olyan alkalmazásra is, amely a LabVIEW által mért és feldolgozott eredményekből egy kalibrációs adatlapot készít, illetve az előző fejezetben leírt kiegészítő hardver vezérlőjén futó firmware-re is.

Az program egyszerűsített folyamatábrája az [7] sz. mellékletben látható.

Az alkalmazás tervezésekor leginkább azt tartottam szem előtt, hogy a tesztelés független legyen a tesztelési lépések számától. Vagyis, hogy a mérések, azok kiértékelése, a kalibrációs adatlap készítése ugyanúgy történjen meg a felhasználó által megadott 10 mérési lépés alkalmával is, mint akár több ezer lépés esetén.

Az üzenetek fájlból történő beolvasásának számos előnye van az előbb leírtak mellett. Így viszonylag kisméretű és áttekinthető blokkdiagram volt készíthető. Hiszen nem kellett minden egyes üzenetküldésnél a blokkdiagramban fixen megadni az üzenet azonosítót, adat bájtokat, adat hosszt, stb. A parancsok tabulátorral tagolt fájlokban készülnek, így táblázatkezelőkkel megnyitva könnyen szerkeszthetőek, és bővíthetőek, akár a már meglévő sorok másolásával is.

A LabVIEW program elindítása előtt van lehetőség az interfészek beállítására. Amennyiben még nem készült fejléc, vagy új fejléccel akarjuk ellátni a kalibrációs adatlapot, akkor ennek a beállítását is ekkor tehetjük meg. Valamint ekkor adhatjuk meg, hogy hogyan fusson le a teszt program, megálljon-e, küldjön-e hibaüzenetet, vagy felhasználói jóváhagyás nélkül fusson-e tovább nem megfelelő CAN buszról fogadott üzenetek esetén.

Az elindítást követően a kommunikációs vonalak inicializálása történik meg. Természetesen, ha rossz interfészt adtunk meg a CAN, vagy a soros eszköznél, akkor hibaüzenet jelenik meg, majd leáll a program. A készülékek beállítása után a soros eszköz bekapcsolja a dilutert, ami minden bekapcsolást követően addig küld üres üzeneteket a CAN vonalra, míg nem kap egy a címének megfelelő nyugtázó üzenetet. Itt meg kell jegyezni, hogy a TECAN pipettázó automatáinak minden modulja ilyen módon viselkedik a tápfeszültségre kapcsolást követően. Természetesen mindegyik

különböző arbitrációs azonosítóval küld üzeneteket a buszra, amely üzenetek azonosítóiból a CU (központi egység) be tudja azonosítani, hogy melyik eszköz küldte az üzenetet, és ennek megfelelő adat byte-okkal tölti fel a jóváhagyó üzenetet. Ezen utasítások azonosítója mindig azonos (0x80).

A diluter háromféle üzenet csoportot különböztet meg. Lekérdező (Report) üzeneteket, beállító (Set) parancsokat, és mozgató (Action) parancsokat.

Ha a dilutert bekapcsolva nincs adatforgalom a buszon, akkor egy hibaüzenet jelenik meg a képernyőn, majd leáll a tesztelés, ellenkező esetben először azt ellenőrzi le a program, hogy a fogadott üres üzenet azonosítójának értéke a megfelelő határok közé esik-e. Ha nem szintén leáll a program. Ha viszont megfelelő hexadecimális értékű az azonosító, akkor ebből kiszámolásra kerülnek az előbb említett üzenetcsoportokhoz tartozó ID-k, majd egy nyugtázó üzenetet küld a LabVIEW a TECAN robot vezérlő egységéhez hasonlóan.

Ezután kerül megnyitásra az első, parancsokat tartalmazó fájl, ennek minden sorában kötelező megadni az elküldendő üzenetet, ezt csak ACSII alakban. Nincs szükség az adat byte-ok számának, vagy a byte-ok decimális alakjának megadására, mivel ezeket az ASCII-ban megadott alakból a LabVIEW mind ki tudja számolni. Megadható még hogy az adott üzenetre mi a helyes válasz, ez csak opcionális, e nélkül is tökéletesen lefut a program, viszont, ha valamilyen nem várt adatokat kapunk válasz üzenetként, akkor arról nem lesz semmilyen információnk a tesztelés közben.

Az üzenetek kiküldése, válasz beolvasása egy-egy FOR ciklusban történik, és mindaddig végrehajtódik, ameddig van nem üres sora a táblázatot tartalmazó tömbnek. Mindez négyszer ismétlődik meg az egész programban, először a **lekérdező**, majd a **szelepmozgató**, azután a **fecskendő karmozgató** parancsai kerülnek kiküldésre, végül pedig a **sebességmérés**hez szükséges parancsok.

A fényképek készítésekor a fényforrásokat soros porton keresztül bekapcsolja a mérőrendszer. Attól függően, hogy a szelep tengelyéről vagy a fecskendő mozgató karról készít képet, a felső két LED, vagy mind a négy fog világítani.

Az utolsó fénykép elkészítése után a diluter bekapcsolt állapotára már nincs szükség, mivel ekkorra már minden szükséges információ le lett kérdezve róla, így kikapcsolásra kerül.

Az USB kamerákkal készített képek azonnal feldolgozásra kerülnek, a fotók elkészítése után programban már csak a mért elfordulás-, távolság- vagy sebességértékek továbbítódnak. Ezeknek a mérési értékeknek a relatív hibája egy további blokkban kerül kiértékelésre. Ezek az értékek, illetve a tesztelés során elküldött parancsok, az arra kapott válasz üzenetek mind egy tabulátorral tagolt szöveges fájlba kerülnek. Ezt követően a LabVIEW elindítja a kalibrációs adatlapot elkészítő külső programot, ezt a fájlt adva bemenő paraméterének. A PDF fájlban nem csak áttekinthetőbb formában jelennek meg a mérési eredmények, hanem a firmware parancsokról is egy soros leírás is olvasható benne. Ha a tesztelés során volt olyan üzenet, amely többször is kiküldésre kerül, akkor az ebben a fájlban egy táblázatba rendezve fog megjelenni. Csak azok a parancsok lesznek feltüntetve benne, amelyek legalább egyszer elküldésre kerültek a mérés során, vagyis a PDF fájl szövegtörzse dinamikusan változik attól függően, hogy a felhasználó milyen tesztelési lépéseket állított be a mérést megelőzően.

4 Program kidolgozása

4.1 Driverrek kiválasztása

A CAN buszon történő kommunikáció lebonyolításához az *NI-CAN driver Frame API* library lett felhasználva. Ugyan a *PXI-8511*-es kártya számos driverrel működik, viszont az USB-s eszköz csak ezt a library-t támogatja, ezt is csak korlátozottan. A támogatott funkciók, illetve a korlátozások az [18] sz. irodalomjegyzékben található linken érhetők el.

USB kamerák használatát egyelőre a National Instruments két drivere támogatja, az *IMAQ for USB* és az *IMAQdx*. Pályamunkámban az utóbbi került felhasználásra, mert ezzel a kamerákra való hivatkozás könnyen megvalósítható. A driver telepítése után a *Measurement & Automation Explorer* szoftverben lehetőség nyílik a csatlakoztatott USB kamerák nevének megadására (ezzel a névvel történik a hivatkozás a LabVIEWs felületen) és minden egyéb paraméterének konfigurálására. A *MAX*-ban a kamerák számos paraméterét megadhatjuk és menthetjük is, mint például a video felbontását, fényerőt, kontrasztot, képélességet stb.

A program elkészítése a *LabVIEW 2010 SP1*, *NI Vision 10.0*, *NI-IMAQdx 3.7*, *NI-CAN 2.7.3*, *NI-Serial 3.6*, *Measurement & Automation 5.0* felhasználásával történt.

4.2 Felhasználói felület

Az előlapon egy *tab control* helyezkedik el, összesen 5 db füllel rendelkezik, amelyek az alábbiak:

- Main
- CAN settings
- Serial settings
- Calibration document settings
- Allowed error

4.2.1 Main panel



3. ábra A program fő ablaka

A mérés elindítását követően a rendszer az első fogadott üzenet azonosítójából kiszámolja a diluter (háttoldalán beállított) címét. Ennek értéke az *Address* decimális szám kijelzőben fog megjelenni.

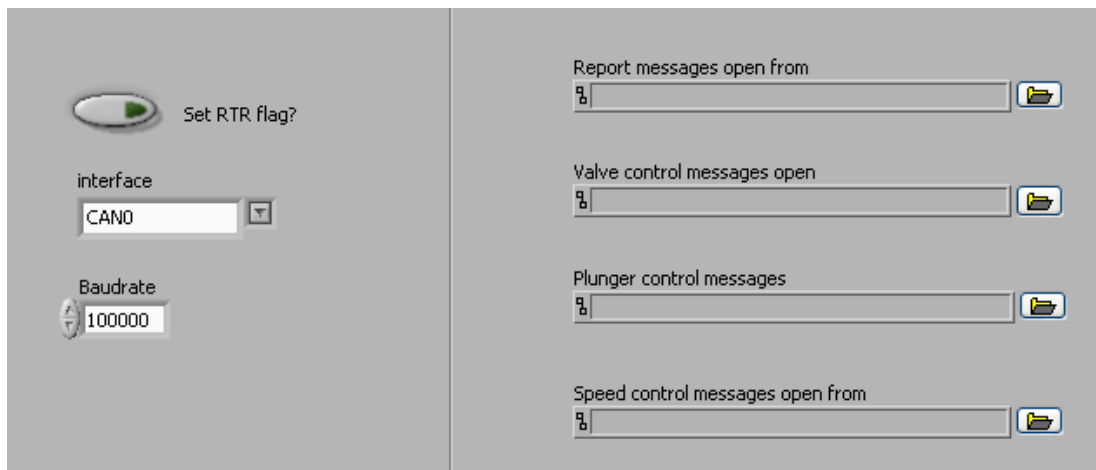
A *Diluter type* kijelzőn a csatlakoztatott eszköz típusa fog megjelenni a készülék firmware verziójának lekérdezését követően. A mérőrendszer kétféle generációjú dilutert különböztet meg, ezek XP3000 (VCC) és XP2000 (smart) típusok.

A panelon meg lehet adni, hogy mit történjen, ha valamelyik üzenetre nem a kívánt (előre megadott) választ kapjuk. Vagyis lehetőség van ebben az esetben egy hibaüzenet megjelenítésére, vagy a tesztelés leállítására.

Az előlapon kerül kijelzésre továbbá a diluter tápfeszültsége és -árama. A kijelzett adatok csak tájékoztatásul szolgálnak, nem szükséges tudnunk ezeket az értékeket a tesztelés futtatásához.

A panel jobb oldalán látható *Actual measured image* kijelzőn mindig az éppen elkészített fotó látható.

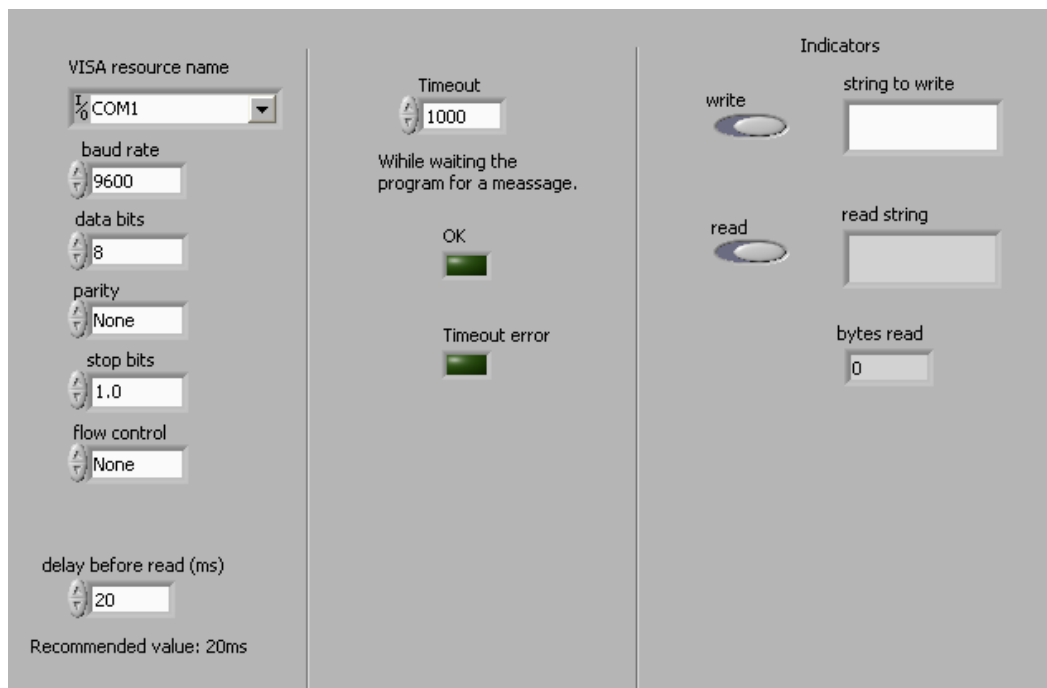
4.2.2 CAN settings



4. ábra CAN buszon történő kommunikációval kapcsolatos beállítások felülete

A panelon a CAN interfész beállítása és a parancsokat tartalmazó fájlok elérési útjának megadása történik.

4.2.3 Serial settings panel



5. ábra Soros port beállításainak felülete

Ezen a felületen kell bekonfigurálni a soros portot. A *Timeout* értékkel azt lehet megadni, hogy maximálisan mennyit várakozzon a program egy kiküldött parancsot követően a beérkező helyes válaszra.

4.2.4 Calibration document settings panel

Tested by (Company name)
University of Obuda

Company address
Tavaszmezo u. 15-17.

City
Budapest

Zipcode
1084

Phone
-

Tester
Zoltan Bertok

Purpose

Description

Company logo (max. 235x115 pixel)
K:\LabVIEW\obudai_egyetem.JPG

Image review

Image

Create property file

6. ábra A jegyzőkönyv adatit beállító felület

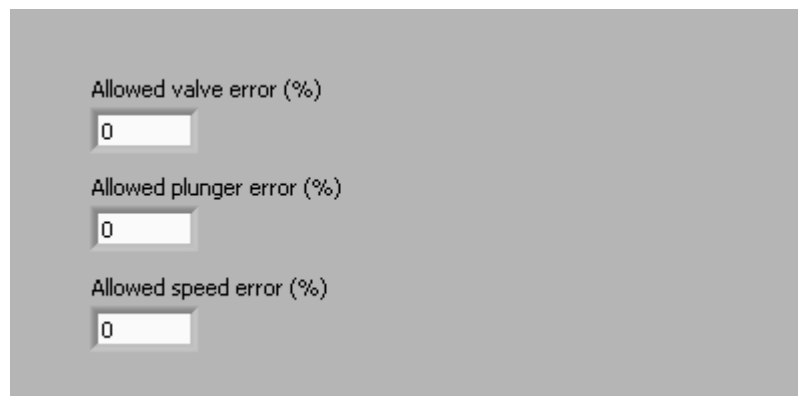
Ezen a panelon lehet beállítani a külső programmal elkészítendő jegyzőkönyvben megjelenítendő cégalapadatok. A kép elérési útvonalának megadását követően lehetőség van a kép megtekintésére is. Az 6. ábrán beállított paraméterek esetén a dokumentum fejléce az 7. ábrán látható módon fog kinézni. (Egy mérés adatait tartalmazó jegyzőkönyv egyes részei a [12] sz. mellékletben tekinthető meg.)

Company name (tested by)	University of Obuda
Address	Tavaszmezo u. 15-17.
City	Budapest
Zipcode	1084
Phone	-
Tester	Zoltan Bertok
Date	2011.10.02.

Óbudai EGYETEM
Universitas Budensis

7. ábra A jegyzőkönyvben megjelenített mérést végző szervezet adatai

4.2.5 Allowed error panel



The image shows a software interface with a light gray background. It contains three vertically stacked labels and input fields. Each label is followed by a small rectangular input box. The first label is 'Allowed valve error (%)' with a box containing '0'. The second label is 'Allowed plunger error (%)' with a box containing '0'. The third label is 'Allowed speed error (%)' with a box containing '0'.

Parameter	Value (%)
Allowed valve error (%)	0
Allowed plunger error (%)	0
Allowed speed error (%)	0

8. ábra A megengedhető hibák beállításának felülete

Ezen a felületen állítható be az, hogy maximálisan hány százalékos relatív hiba engedhető meg az egyes tesztelési lépések alkalmával.

4.3 Részletes program leírás

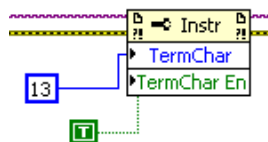
4.3.1 CAN busz inicializálása

Az interfész konfigurációja teljesen hasonló módon történik, mint ahogyan az a *Frame API* mintapéldáiban szerepel.

4.3.2 Diluter bekapcsolása

A készülék bekapcsolása soros porton kiküldött üzenettel valósul meg. A soros port lekezelését kis változtatásokkal a *Basic Serial Write and Read.vi* elnevezésű példaprogram valósítja meg. Ez a programrészlet párhuzamosan fut az egész tesztelés alatt az alkalmazás "fő" szálával. Ez azért nagyon hasznos megoldás, mert így a portot mindössze egyszer kell csak inicializálni, azt követően pedig bármikor lokális változók segítségével adatot lehet küldeni, illetve kiolvasni vonalról. Ennek köszönhetően nem csak a fényforrások és a diluter ki- és bekapcsolása valósítható meg a főprogram bármely pontján, hanem a feszültség, illetve áramértékek kiolvasása is.

Az soros eszköz úgy van beállítva, hogy a beérkező adatokat "kocsi vissza" karakterenként különböztesse meg, így egyszerűen szét lehet választani az egyes parancsokra érkezett válaszokat. (9. ábra)



9. ábra Az üzenetsomagok szétválasztására szolgáló blokk

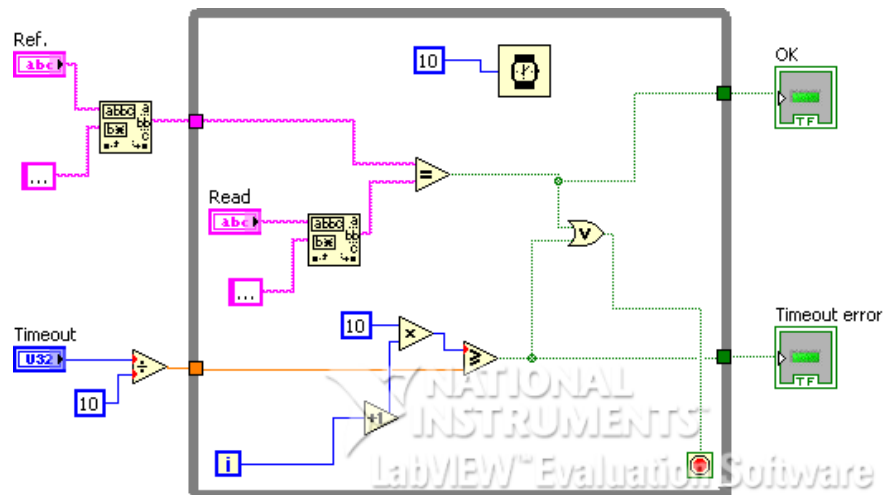
Minden üzenetküldést követően a mérőrendszer vár ezekre a válaszokra. Amennyiben nem érkezik válasz, a képernyőn egy hibaüzenet jelenik meg, amellyel leállítható a tesztelés. A várakozás idejét előre meg lehet határozni a *Serial settings* előlapi panel beállításai között.

A bekapcsoláshoz szükséges paraméter a SDO (Set Diluter On), az erre érkező nyugtázó válasz pedig a DON (Diluter On).

A várakozás az 10. ábrán látható SubVI-al van megvalósítva, blokkvázlata a 11. ábrán látható.



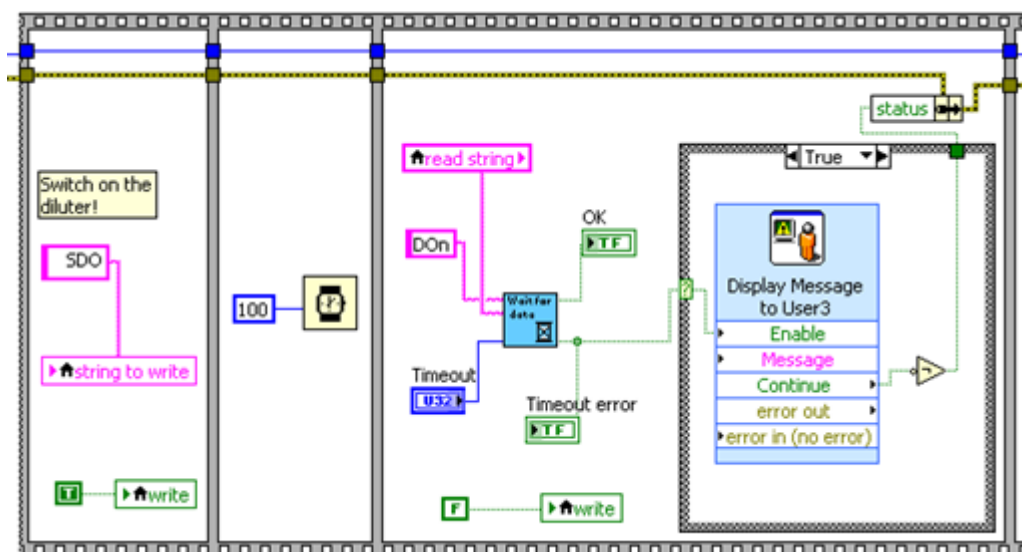
10. ábra A waitForSerialData SubVI ikonja



11. ábra A waitForSerialData SubVI blokkdiagramja

A beolvasandó és a beolvasott szöveg összehasonlítása 10 ms-oként történik meg. A késleltetés vagy akkor áll le, amikor van olyan beérkezett üzenet, amely azonos a referenciának megadottal, vagy akkor, ha a WHILE ciklus iterációjának tízszerese eléri a *Timeout* értékét.

A diluter bekapcsolását a 12. ábrán látható VI részlet végzi.



12. ábra A diluter bekapcsolását végző programrész

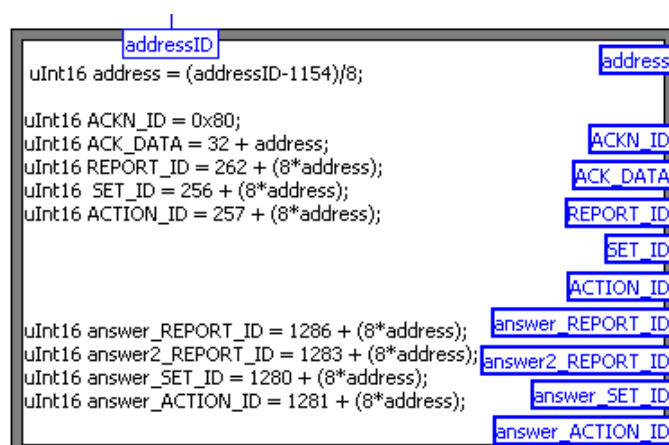
4.3.3 Diluter címének meghatározása

A készülék bekapcsolását követően, az előbbieken már említett módon a diluter a hátoldalán beállított címének megfelelő azonosítójú üres üzeneteket, azaz adat kérő üzeneteket (RTR - Remote Transmission Request) küld, jelezve, hogy rá van csatlakoztatva a buszra.

Egy TECAN pipettázó automata maximum 16 ilyen készüléket tartalmazhat. 0-ás cím esetén az arbitrációs azonosító hexadecimális értéke: 482, míg 15-ös cím esetén: 4FA. E két érték között 8 hexánként változik a címzés egyet.

Éppen ezért, az első üzenet beolvasását követően csak akkor nem áll le a program futása, ha az ID értéke ezek között a határok között van. Ha ennél nagyobb, vagy kisebb, akkor hibaüzenet jeleneik meg a képernyőn, majd leáll a tesztelés.

Megfelelő azonosító esetén viszont kiszámolásra kerülnek a nyugtázó üzenet elemei, valamint a REPORT, ACTION és SET parancstípusok ID-jai is. Ezt a számolást a 13. ábrán látható *Formula node* végzi.



13. ábra Arbitrációs azonosítók kiszámolását végző *formula node*

Jelenleg a mérőrendszer csak az **ACKN_ID** (Acknowledge ID – nyugtázó azonosító), **ACK_DATA** (Acknowledge ID – nyugtázó üzenet adat bájtt), **REPORT_ID** (lekérdező üzenetek azonosító), **SET_ID** (beállító üzenet azonosító), **ACTION_ID** (mozgató parancs azonosító) változókat használja. Az **answer_REPORT_ID**, **answer2_REPORT_ID**, **answer_SET_ID**, **answer_ACTION_ID** az egyes parancsokra érkező válaszok azonosítóit jelenti, jelenleg nincsenek felhasználva. Alkalmazásukkal a hibakezelés szigorúsága javítható.

4.3.4 Nyugtázó üzenet küldése

Minden "visszaigazoló" üzenet azonosítója 0x80 értékű. Az üzenetben két azonos értékű bájt szerepel, amely értéke az ACK_DATA.

A visszaigazolás elküldését követően a diluter abbahagyja az üres üzenetek küldését, és vár a parancsok beérkezésére.

4.3.5 Firmware parancsokat tartalmazó fájlok

Ennek az alkalmazás részletnek a magvalósításánál az volt a cél, hogy a parancs fájlokat könnyű legyen létrehozni és szerkeszteni, illetve, hogy egy-egy új parancs felvételekor ne kelljen azokat részletekbe menően leírni, vagyis ne legyen szükség az adatbájtok számának, hexadecimális értékének, stb. megadására.

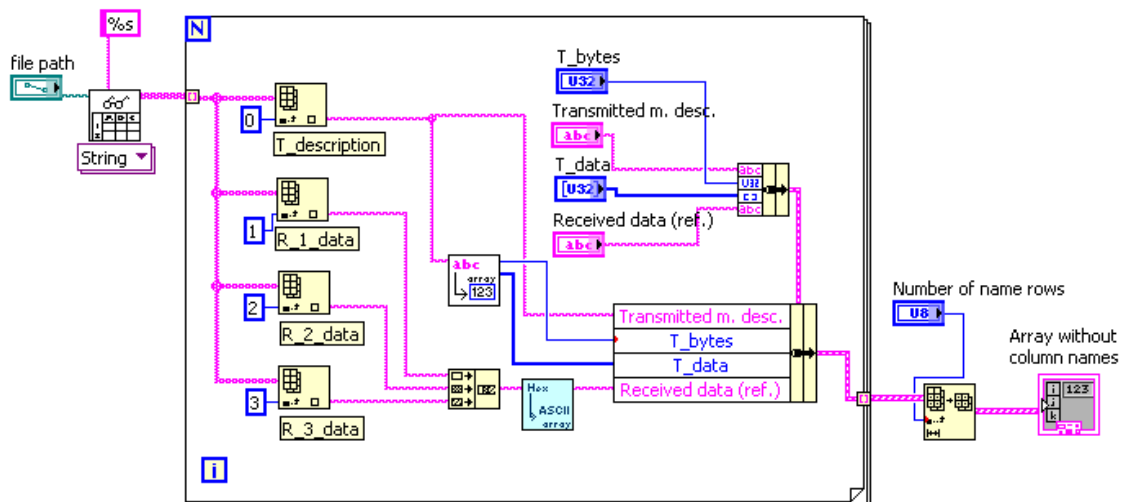
A könnyű szerkeszthetőség érdekében döntöttem úgy, hogy minden ilyen fájl táblázatkezelőben lesz létrehozható, a [8] sz. mellékletben látható formai követelményeknek megfelelően. (Természetesen az oszlop megnevezések csak tájékoztató jellegűek.)

A teszt program megfelelő működéséhez feltétlenül szükség van az elküldendő üzenet megadására, ezt csak ASCII alakban kell megtenni.

Előző fejezetekben már említett módon lehetőség van az egyes válasz üzenetek helyességének vizsgálatára is, ha tesztelés alatt ezt az opciót is szeretnénk használni, akkor a fogadni kívánt adatok oszlopait is fel kell tölteni. Ezt az egyes ASCII karakterek hexadecimális értékével tehetjük meg. Itt azért volt szükség a hexadecimális formánál maradni, mivel számos olyan karakterre is szükség lehet a feltöltés során, amelyeket nem találunk a billentyűzeten.

(Az utóbbi értékeket nem lehet megtalálni a TECAN cég egyik publikált leírásában sem, mint ahogyan az arbitrációs azonosítókat sem. Ezek kiderítéséhez egy működő automata CAN busz vonalán haladó üzenetek visszafejtésére volt szükség.)

A táblázatok beolvasását és értelmezését a következő SubVI (readCanMessagesFromSpreadSheet.vi) végzi (14. ábra):



14. ábra A diluter parancsok beolvasását végző SubVI blokkdiagramja

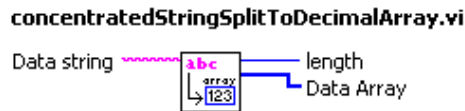
A fájl megnyitását a *Read From Spreadsheet File.vi* gyári függvény végzi. Ennél a blokknál meg kell adni, hogy milyen típusú tömböt hozzon létre a fájlban található adathalmazból (itt a String beállítására volt szükség).

A beolvasás után a kétdimenziós tömb minden egyes eleme (vagyis mindegyik 1D tömeleme) a táblázat egy-egy sorának felel meg, ezeket a sorokat a LabVIEW az előbbi képen látható módon feldarabolja, majd létrehoz a sorokból egy klasztert. A SubVI kimenete az ezeket a klasztereket tartalmazó tömb lesz.

A CAN kommunikáció lebonyolítását a *Frame API* végzi, amely üzenetküldő, *ncWriteNet.vi* gyári függvény futtatásához a következő paramétereket szükségesek:

- **IsRemote** (Lekérdező keret): amennyiben értéke 0, úgy adat keretek küldését valósítja meg. (A mérőrendszerben nincs szükség lekérdező keret használatára.)
- **ArbitrationId**: arbitrációs azonosító értékét lokális változókon keresztül kapja meg ez a bemenet, a *Formula node*-ból.
- **DataLength** (adat hossz): értéke az adat bájtok számát jelzi.
- **Data** (adat): ez a bemenet egy maximum nyolc darab előjel nélküli elemből álló tömböt vár.
- Ebből következik, hogy a beolvasott String adattípusú elküldendő üzenetet át kell alakítani decimális tömbbé. Ezt az átalakítást a *concatenatedStringSplitToDecimalArray* SubVI végzi (15. ábra). A függvény a bemenetén megjelenő szöveget először nyolc karakter hosszúságúra maszkolja a

Match Pattern Function gyári függvény segítségével. A találat kereséséhez megadott reguláris kifejezés nyolc darab pont (tehát bármilyen karakter). Az első egyezés, ami az első nyolc karakter, fel lesz darabolva karakterenként. Ezek egy tömbbe kerülnek, majd elemenként ASCII-ből decimális számmá lesznek alakítva a *Type Cast* blokk segítségével.



15. ábra A concatenatedStringSplitToDecimalArray.vi ikonja

A fájl beolvasását követően az előbb említettek szerint minden sor fel lesz dolgozva. Amennyiben a parancsokat tartalmazó táblázat oszlopainak adtunk nevet, akkor azokon is. Ahhoz, hogy ezek ne kerüljenek elküldésre az *ncWriteNet.vi* függvénnyel, ki kell törölni az(oka)t a tömbből. Ezért a megnyitást végző SubVI-ban (vagyis a *readCanMessagesFromSpreadSheet.vi*-ban) meg kell adni azt, hogy a táblázatnak hány darab ilyen nem feldolgozandó sora van (16. ábra). Az így beállított érték az *Array Subset* blokk index értéke lesz, amely annyit tesz, hogy a teljes beolvasott tömbből létrehoz egy altömböt (ez tartalmazza a parancsokat), amely az általunk megadott számú sortól kezdődik, tehát így akár több soros megjegyzések is készíthetők a dokumentumok elejére.



16. ábra A diluter parancsok beolvasását végző SubVI ikonja

A fájlok beolvasása a főprogramban:

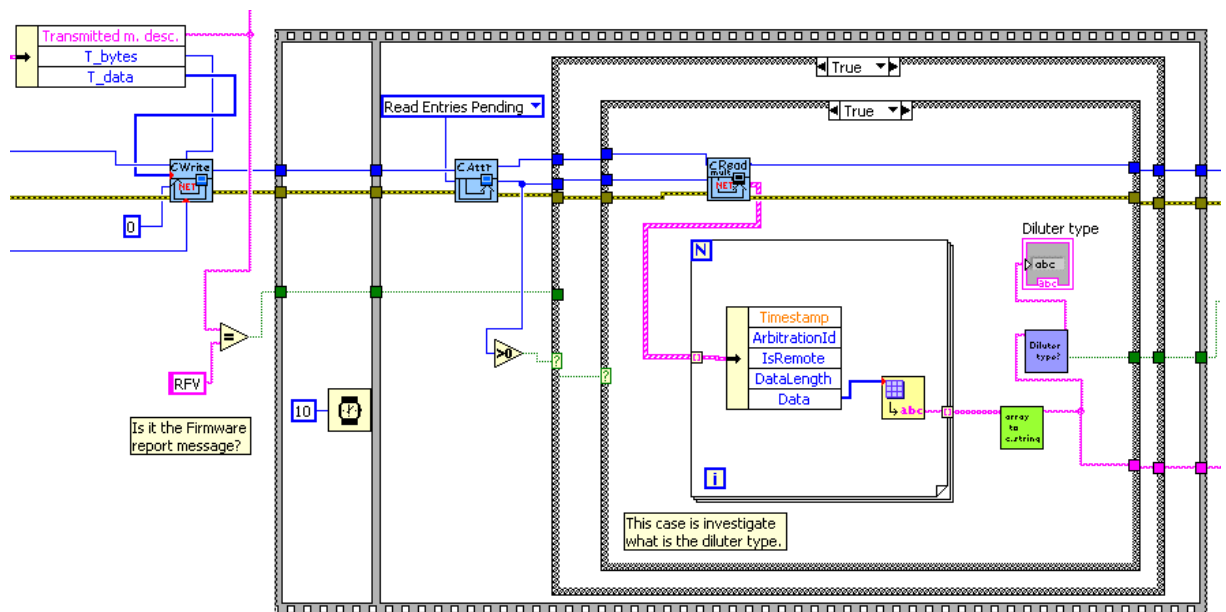
Mind a négy alkalommal teljesen hasonlóan történik, egy kivétellel, a *Lekérdező üzenetek* megnyitása esetén. Ekkor ugyanis nincs felhasználva a fogadott üzenetek eleme a kibontott klaszterben (17. ábra). Ennek az az oka, hogy a *Report* üzenetek esetén nincs értelme az összehasonlítás elvégzésének, mivel azok közül több olyan is létezik, amelyre adott válasz bekapcsolásonként, vagy inicializálásonként változik. Ilyen például az RDF (REPORT DIAGNOSTIC FUNCTIONS - Report number of power

Valve control messages open from

The diagram shows a CAN bus system. On the left, a 'Valve control' unit (represented by a blue box with a switch icon) is connected to a 'CAN Message' block (represented by a document icon). The 'CAN Message' block is connected to a CAN bus line (represented by a vertical line with a 'N' label). The bus line is connected to a PLC (represented by a grey box with a ladder logic diagram). The PLC has a 'Transmitted m. desc.' block (represented by a yellow box) which is connected to a 'Received data (ref.)' block (represented by a pink box). The 'Received data (ref.)' block is connected to a 'T_bytes' block (represented by a blue box) and a 'T_data' block (represented by a blue box). The 'T_bytes' and 'T_data' blocks are connected to a 'Received data (ref.)' block (represented by a pink box).

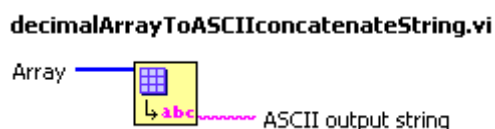
Üzenetek beolvasása

Ebben alfejezetben is különbséget kell tenni a *Lekérdező üzenetek* és a többi parancsok között. Ugyanis a *Report* utasítások esetén nem lehet előre tudni, hogy az egyes kérdésekre hány darab adatkerettel fog válaszolni a diluter, a maximum keretszám azonban minden esetben háromnál nem nagyobb.



VIEW a parancs elküldését követően 10 ms-ig vár, hogy a válasz megérkezzen, az *ncGetAttr.vi* gyári függvényével megvizsgálja, hogy a válasz hány adat keretet tartalmaz. Annak – a 18. ábrán látható beállítás esetén – az *AttValue* kimenetén az

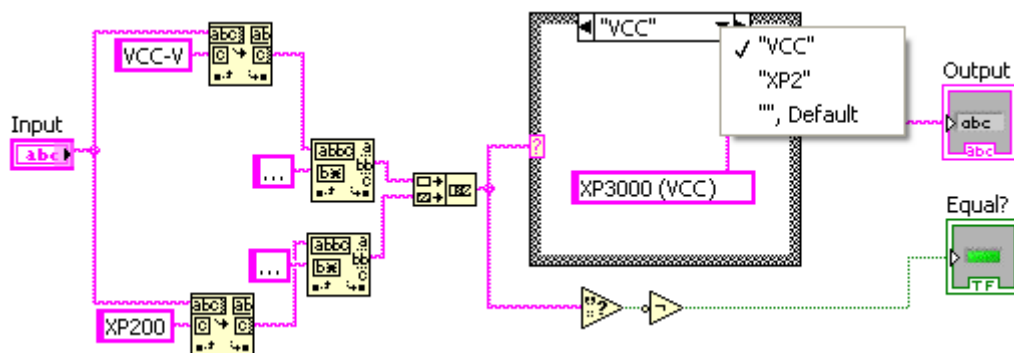
olvasási sorban elérhető keretek száma jelenik meg. Ha ez az érték nulla, akkor a blokkdiagram részlet (18. ábra) string adattípusú kimenetére egy konstansból a "No received message" kerül. Ellenkező esetben viszont az *ncReadNetMulti.vi* olvassa ki az *AttValue* által meghatározott mennyiségű keretet az interfész várakozási sorából. A kiolvasott értékek közül csak a beérkezett üzenetek adat bájtjai lesznek feldolgozva. Ezeket 19.ábrán látható SubVI alakítja át egy összefüggő stringgé, a későbbi feldolgozás számára.



19. ábra A decimális tömb - ASCII string átalakítását végző SubVI ikonja

A készülék típusának beazonosítását a *firmwareType* elnevezésű (lásd a 15. ábra 'Diluter type') SubVI végzi. A mérőrendszer az XP3000plus típusú eszközök vizsgálatára készült, mivel nagy a hasonlóság az különböző generációjú készülékek között, ezért szükségesnek láttam egy ezzel kapcsolatos ellenőrzési lépés megvalósítását is. Vagyis, hogy a tesztelés csak akkor legyen végrehajtható, ha a csatlakoztatott készülék firmware típusa egyezik a támogatott eszközével. További előnye ennek a kialakításnak, hogy a későbbi továbbfejlesztések során az itt beolvasott verzió alapján tudja majd bekonfigurálni a mérőrendszer saját magát az adott eszközhöz. Ez azért lehet szükséges, mert az egyes diluter típusok nem csak a kezelt üzenetekben, hanem geometriai méreteikben is különböznek. Az firmware típus ellenőrzését végző SubVI a 20. ábrán látható, jelenleg két különböző típus azonosítására szolgál. A mérőrendszer jelenleg a VCC (XP3000plus) készülékek vizsgálatára használható. Az egyes firmware típusok a következő szintaxis szerint épülnek fel, XP3000-es diluter esetén: 'VCC-Vx.xx-MM/YYYY', XP2000-es diluter esetén: 'XP2000-Vx.xxx-MM/YYYY'. Ebből következik, hogy a SubVI-ban csak az első öt karakter vizsgálatát szabad elvégezni. Az azonosság keresését a *Match Pattern* függvény végzi.

Amennyiben egyik összehasonlítás sem jár eredménnyel, akkor hibaüzenet jelenik meg a képernyőn ("Unknown diluter type" hibüzenettel) és leáll a tesztelés.



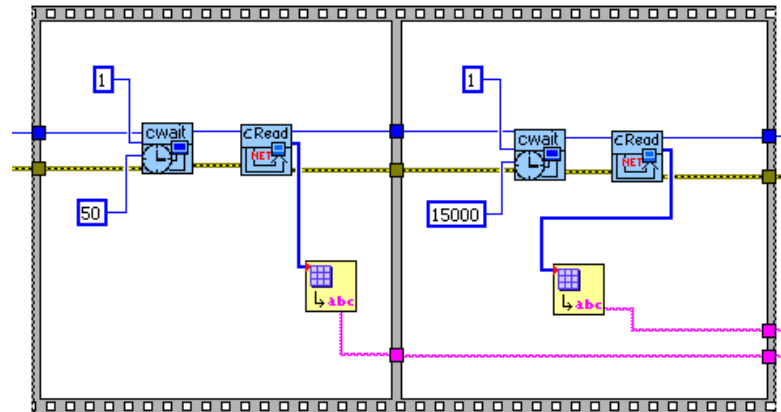
20. ábra A *firmwareType.vi* blokkdiagramja

A mozgató utasítások elküldésekor a diluter először egy üres üzenetet küld vissza, jelezve, hogy elkezdte a parancs végrehajtását, majd mikor a tengely a kívánt pozícióba fordult, vagy a kar a megadott pozícióba ért, akkor ismételt küld egy üzenetet (0x2060 tartalommal) jelezve, hogy végzett az utasítás végrehajtásával. Az egyes mozgató parancsok elvégzéséhez akár 5-10 másodperc is szükséges, ezért belátható, hogy az előző üzenetfogadási elrendezés itt nem használható, mert lemaradnánk a visszajelzésekről. A beolvasást itt az *ncReadNet.vi* gyári függvény végzi, amellyel egy-egy keret olvasható ki az interfészből. A két üzenet beérkezéséig az *ncWaitForState.vi* felhasználásával várakozik a program. A VI használatakor meg kell adni, hogy hány keret fogadásáig várakozzon, illetve, hogy a várakozás maximálisan hány másodpercig tarthat. Mivel az utasítás elküldését követően a diluter 0,7 ms-al később jelzést ad a művelet végrehajtásának megkezdéséről, ezért az első blokkban elegendő a *Timeout* értékét mindössze 50 ms-ra állítani. Ezzel szemben a következő blokkban mindenképpen legalább 10 s várakozást kell lehetővé tenni, mert a két végső pozíció közötti mozgás sok időt vesz igénybe.

A kialakítás további előnye a másik elrendezéssel szemben, hogy itt nem nekünk kell beállítani – időzítések segítségével –, hogy mennyit várakozzon a program két utasítás kiküldése között. Mivel itt mindaddig vár a mérőrendszer, míg nem kap visszajelzést a dilutertől.

Minden mozgató utasításra két adat keret válasz érkezik, vagyis amennyiben helyesen töltöttük fel a parancsokat tartalmazó táblázatot, akkor minden fogadott üzenet kettő keret hosszúságú lesz. A diluter ugyanilyen formában küldi a hibaüzeneteket is, például, ha nem tudja végrehajtani a kar mozgását valamilyen fizikai akadály miatt, vagy a

léptetőmotor meghibásodása, vagy nem értelmezhető parancs fogadása esetén is (ilyen hibaüzenetek lehetnek: 0x2360, 0x2960, stb.). Vagyis ez az elrendezés (21.ábra) még hiba esetén is alkalmazható.



21. ábra Mozgató, beállító parancsokra érkező válaszok beolvasása

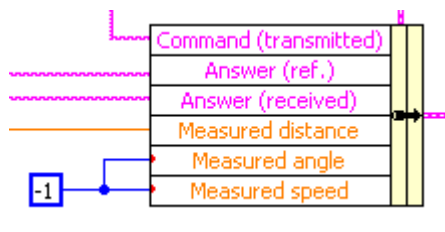
4.3.7 Tesztelési lépések mérési eredményeinek tárolása

Az előbbieken részletezett parancsok elküldését és fogadását követően (mozgató parancsok esetén) történik a tényleges mérés, vagyis a fényképek készítése és feldolgozása.

Minden egyes tesztelési lépésről a következő jellemzők kerülnek tárolásra:

- Kiküldött üzenet
- Várt válasz
- Fogadott válasz
- Mért távolság (kar esetén)
- Mért elfordulás (szelep esetén)
- Mért sebesség

Ezek az adatok egy klaszterbe kerülnek (22. ábra), a klaszterek pedig egy tömbbe a későbbi kiértékelés számára (lásd 4.3.11. fejezet).



22. ábra Mérési adatok klaszterbe rendesét ábrázoló blokkdiagram részlet

A klaszter mindazon elemei, amelyek az adott lépésnél nem kerülnek felhasználásra ”-1”-el lesznek feltöltve, az egyszerű feldolgozás érdekében, ugyanis egyik mérés eredménye sem lehet negatív szám. Például a *Lekérdező üzenetek* elküldésekor a *Command* és *Answer (received)* paraméter kivételével minden elem ”-1”-es értéket kap.

4.3.8 Fényforrások be- és kikapcsolása

A LED-ek be- és kikapcsolása teljesen azonos módon történik, mint ahogy a diluteré is (lásd 4.3.2 fejezet). A soros eszköznek kiküldendő parancsok listája a 4.4.2.2. fejezetben található. A szelep megvilágításáért felelős fényforrás bekapcsolására a *Lekérdező üzenetek* kiküldését követően kerül sor. A lámpák egészen a tesztelés végéig bekapcsolva maradnak. A kar megvilágítását végző LED bekapcsolása a karmozgató és sebességmérő lépések előtt történik meg. A diluter és a fényforrások kikapcsolására a program végén kerül sor.

4.3.9 Szelep tengely optikai vizsgálata

A szeleptengely elfordulásának és a kar helyzetének meghatározása két különböző képfeldolgozási művelet segítségével lett megvalósítva.

Jelen feladat elvégzéséhez a mintaillesztés (pattern matching) funkció felhasználására került sor.

A mintaillesztés segítségével egy előre meghatározott képrészletet (mintát, vagy template-et) kereshetünk meg a feldolgozandó fotón. A vizsgált képen a megadott mintának minden előfordulása megkereshető, függetlenül az elfordulásától, eltolásától, fényességétől, stb.

A keresés végeredménye a minta közepének helye, valamint orientációja a képen.

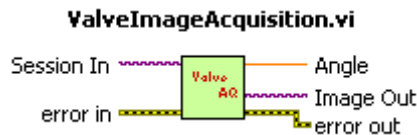
A keresendő minta kiválasztásakor fontos, hogy magas kontrasztú képet adjunk meg, mivel egy ilyen minta a forráskép fényviszonyaitól, vagy a különböző árnyaltú háttereitől függetlenül is beazonosítható [10]. További szempont, hogy olyan információkat tartalmazzon, amelyek csak ezen a képrészleten fordulnak elő és elegendő az egyértelmű beazonosításhoz. A műveletvégzések minimalizálása érdekében a kiválasztott képrészletet a lehető legkisebb méretűnek kell megválasztani.

A minta megadásának kétféle módja van: kijelölhetjük egy már meglévő képen, vagy megadhatunk mentett képeket is.

A keresés végrehajtására számos metódus létezik, ezek közül az ipari képfeldolgozásban a következőket alkalmazzák: *Cross Correlation*, *Scale Invariant and Rotation Invariant Matching*, *Pyramidal Matching*, *Complex Pattern Matching Techniques* [10]

Közülük a *Vision Assistant* két mintaillesztő eljárást használ. Az egyik a *Shift-Invariant Matching*, mellyel a minta helye bármilyen eltolásban megtalálható, a másik a *Rotation-Invariant Matching*, mellyel a minta bármely eltolásban valamint elfordulás esetén is azonosítható. [17]

Mindkét mozgó alkatrész aktuális pozíciójának meghatározására készült egy SubVI. Ennek a bloknak mindössze a *Measurement & Automation Explorer* programban beállított kamera nevét kell megadnunk. A szelep tengelyének elfordulását vizsgáló SubVI ikonja a 23. ábrán látható.



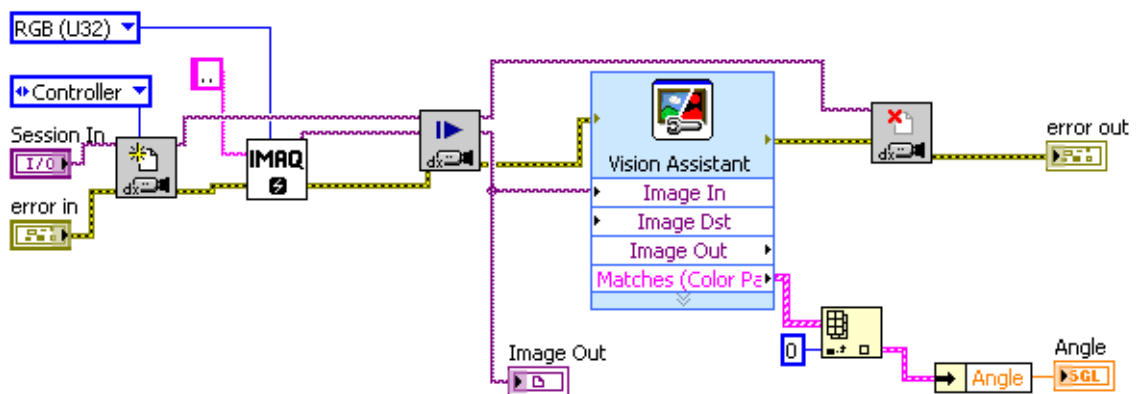
23. ábra A ValveImageAcquisition SubVI ikonja

A SubVI előlapja a [9] sz. mellékletben látható.

A képfeldolgozásnak négyféle vizsgálati módszere létezik a *Vision Toolkit*-ben. Ezek a *Snap*, *Grab*, *Sequence* és a *StillColor* [10]. A felhasznált IMAQdx driverrel ezek közül három támogatott. A *Snap* és *Grp* és a *Sequence*. A *Snap* használatakor egyszerre csak egy fotó kerül a memóriába pufferbe. A *Snap* a feldolgozás végeztével a kamerát újrainicializálva megszerez tőle egy újabb képkockát, amelyet ismét eltárol a memóriában a vizsgálat végéig. A *Snap* lassú sebességű alkalmazásokban használható [19]. Ezzel szemben a *Grp* a folyamatos, nagy sebességű képfeldolgozásra alkalmas [19], míg a *Sequence*-t akkor használják, amikor előre lehet tudni, hogy hány fotón kell a vizsgálatokat elvégezni [10].

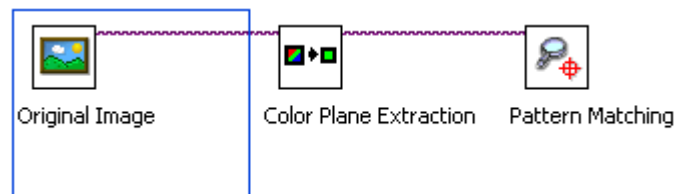
Pályamunkámban a *Snap* típus került felhasználásra, mivel elég csak egy-egy képet készíteni majd kiértékelni az alkatrészek adott pozícióba állását követően. Az USB kamera megnyitása és a fotó készítése a 24. ábrán látható.

A kép feldolgozása 'Vision Assistant Express' VI felhasználásával történt, a választás azért erre a blokkra esett, mert ezzel könnyen és gyorsan lehet nem csak létrehozni az adott képfeldolgozó algoritmust, hanem a későbbi változtatások esetén könnyen lehet elvégezni módosításokat is.



24. ábra A ValveImageAcquisition.vi blokkdiagramja

A script létrehozását megelőzően milliméterpapírról készített fotó segítségével leellenőriztem a kamera hibáit. A készített kép alapján elmondható volt, hogy az a vizsgálni kívánt felületen nem számottevő perspektivikus és nemlineáris hibával rendelkezik, ezért nem láttam szükségét az ilyen jellegű korrekciónak.



25. ábra Az elfordulás mérését végző algoritmus (Vision Assistant script)

A Vision Assistant script blokkvázlata a 25. ábrán látható. A kép megnyitását követően elsőként a fotón a program egy RGB-HSI konverziót végez el. Ez azt jelenti, hogy a színes 32 bites képet szürkeárnyaltos (HSI – Hue Saturation Intensity) képpé átalakítja át. Mivel szürkeárnyaltos képen történő képfeldolgozás kevesebb és egyszerűbb műveletvégzést igényel, ezért gyorsabban le is futtatható, mint ha színes (pl. RGB) képen végeznénk ugyanazokat a feladatokat. Természetesen egy ilyen rövid program esetén ennek nincs túl nagy jelentősége, hosszabb scriptek esetén viszont jelentős idő takarítható meg egy ilyen átalakításnak köszönhetően.

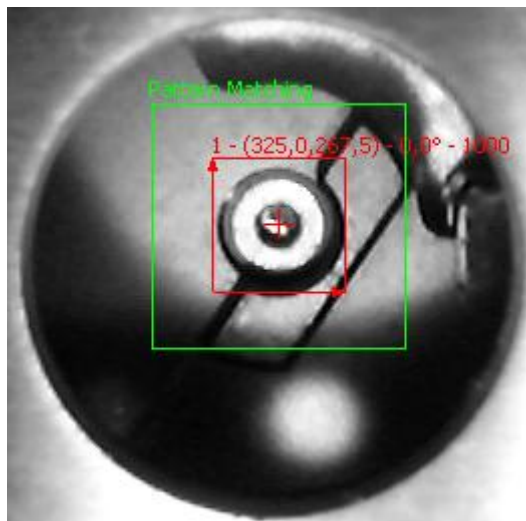
A LabVIEW-ban lehetőség van az elkészített programok futási idejének mérésére a *Performance and Memory Management* felhasználásával. A programrészlet véglegesítését megelőzően lefuttattam egy ilyen tesztet a színes és szürkeárnyaltos képen történő mintaillesztésre. Az utóbbi 25 ms-al hamarabb futott le, a teljes futási idő 125 ms volt.

A következő blokk a mintaillesztés funkcióját látja el. A mért értékek egyszerű feldolgozása érdekében a minta a szelep kiindulási, vagyis 0°-os állását ábrázoltja. A mintaillesztés művelet, ahogy arról a fejezet bevezetőjében szó volt, a képrészlet helyét (X és Y koordinátáit), az elfordulását, valamint a találati arányt (score) és az egyezések számát adja eredményül. Ezek közül csak az elfordulás lesz felhasználva, az egyezések száma pedig minden esetben egy lesz. Ezen művelet elvégzéséhez először ki kellett jelölni a képen azt a területet, ahol a képrészlet biztosan megtalálható, hogy a Vision

Assistant csak a fotó ezen szűk részletén keresse az 26. ábrán látható mintát. Minél kisebb "vizsgált területet" (angolul: Region Of Interest, ROI) adunk meg, annál gyorsabban lefut az algoritmus ezen része. A 27. ábrán látható a kijelölt ROI, valamint a keresés végeredménye.



26. ábra A mintaillesztéshez felhasznált képrészlet (referenciakép)



27. ábra A mintaillesztés funkció végeredménye

A *Pattern Matching Setup Settings* menüben a következő paraméterek lettek beállítva:

Number of Matches to Find	1
Minimum Score	500
Search for Rotated Patterns	True
Angle Range +/- (degrees)	90

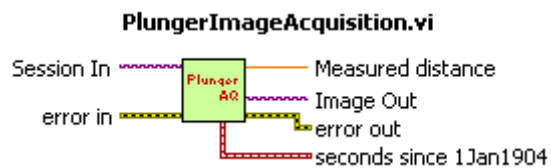
Vagyis a találatnak legalább 50%-ban kell egyezni az általunk megadott képrészlettel (Minimum score: 1000-ből legalább 500), bármilyen szögben való elfordulása esetén.

4.3.10 Fecskendőmozgató kar optikai vizsgálata

A kar elmozdulásának vizsgálata a képfeldolgozás élkeresés műveletével, lett megvalósítva. Segítségével a feldolgozandó képen látható tárgyak kontúrja kereshető meg. Ez azt feltételezve történik meg, hogy két tárgy közötti határ ott van, ahol a képen fényessége hirtelen, ugrásszerűen változik.

Az élek meghatározásához két képfeldolgozási módszer is létezik. Ezek az elsődrendű és másodrendű élkeresési módszerek. Mindkét eljárás a kép fényesség függvényének deriváltjának segítségével határozza meg az éleket, abból kiindulva, hogy a fényesség ugrások (vagyis a tárgy kontúrok) kapcsolatba hozhatóak ennek a függvénynek a deriváltjaival [20]. (A *Vision Assistant Edge detector* funkció beállításában mindkét módszer kiválasztására van lehetőség.)

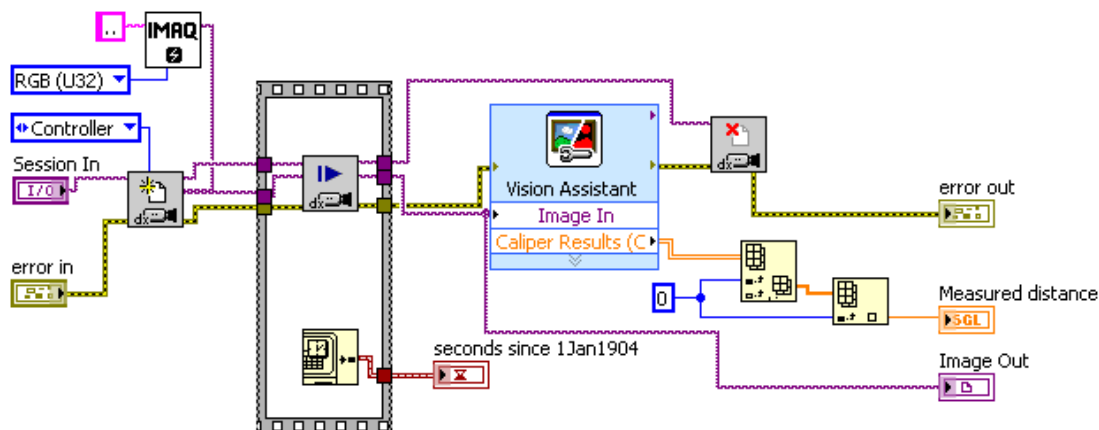
Az előző fejezetben leírtak szerint ez a vizsgálat is *Snap* képkészítési eljárás szerint történik. A fecskendő mozgató kar pozíciójának meghatározásához készül SubVI a 28. ábrán látható.



28. ábra A PlungerImageAcquisition SubVI ikonja

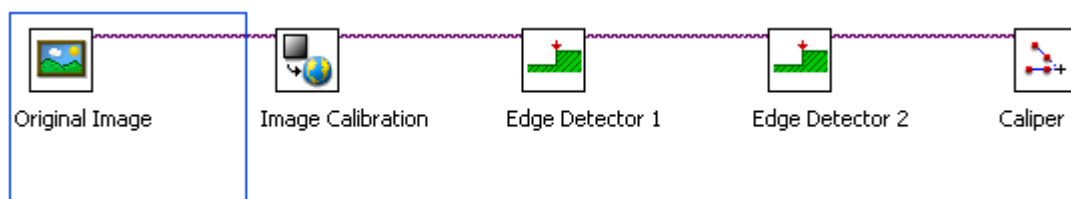
A SubVI előlapja a [9] sz. mellékletben látható.

A másodperc kimenetnek a pozíció meghatározásnál nincs szerepe, ez a sebesség mérésnél lesz felhasználva.



29. ábra PlungerImageAcquisition.vi blokkdiagramja

Természetesen itt is a *Vision Assistant Express* VI alkalmazásával valósult meg a képfeldolgozás. A pozíció meghatározáshoz készült algoritmus működési vázlata a 30. ábrán látható.



30. ábra Távolságmérést végző Vision Assitant script blokk vázlata

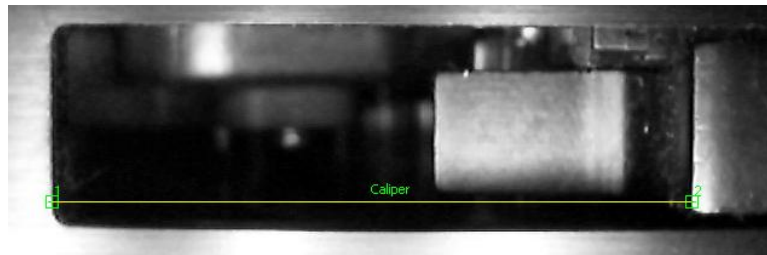
A kép megnyitását követően itt elsőként egyfajta "beállítási" műveletet¹ végzünk el, aminek eredményeként a továbbiakban a fotón végzett mérések eredményeit nem pixelben, hanem egy általunk definiált hosszmértékegységben kaphatjuk meg. Ehhez a "beállításhoz" először a választott mértékegység skálájáról² készítenünk kell egy fotót, melyen megadhatjuk, hogy egy, vagy több egysége a valóságban mekkora távolságnak felel meg. Ennek megfelelően a *Vision Assitant* átváltja a pixeleket a megadott mértékegységre.

Az általam megadott skála egy milliméterpapír volt, amelyről készített fotón 6 cm lett kijelölve. Az *Image Calibration* funkció úgy lett beállítva, hogy ezt a távolságot 60000 µm-nek értelmezze.

¹ a *Simple Calibration* funkciója lett felhasználva, ez akkor alkalmazandó, mikor a kamera perspektivikus és nemlineáris torzítása figyelmen kívül hagyható

² a skála tetszőleges lehet, ...

A távolság meghatározása a *Caliper* funkcióval a kar felső sarka és a nulla pozícióhoz tartozó perem között történik. Természetesen a kar kiindulási pozíciójában is mérhető köztük néhány pixelnyi távolság, amit a kiértékeléskor figyelembe kellett venni. Az egyes és kettes él keresés, valamint a *Caliper* funkció végeredménye a 31. ábrán látható. A mérés azért ebben az *y* pozícióban történik, mert itt az él keresésnél nem okoznak gondot a vizsgálni kívánt felszíntől mélyebben megcsillanó felületek.



31. ábra Kar elmozdulásának mérése

Az *Edge Detector* funkció *Edge Detector* panelen a következő paraméterek lettek beállítva:

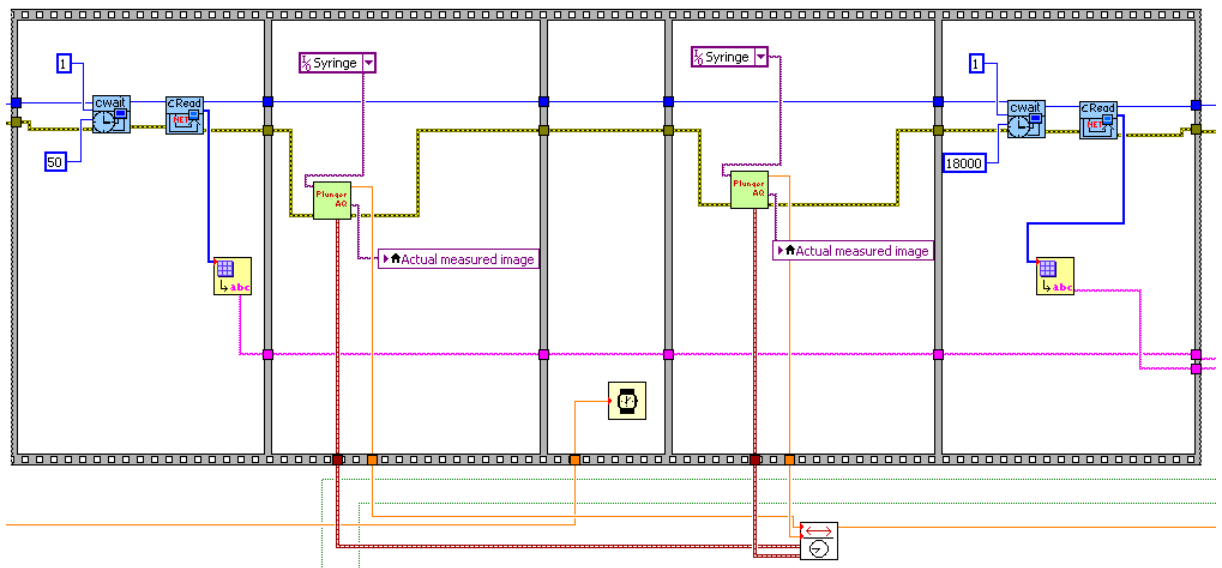
Edge Detector	Advanced Edge Tool
Look For	First Edges
Edge Polarity	All Edges
Interpolation Type	Bilinear
Kernel Size	3
Width	3
Min Edge Strength	70

4.3.11 Sebességmérés

A sebesség mérése az előző fejezetben bemutatott SubVI alkalmazásával történik. A sebességet a két távolságmérés különbsége, valamint a két kép készítése között eltelt idő hányadosa adja.

Az 29. ábrán látható, hogy az aktuális idő értéke a kép feldolgozása előtt történik, vagyis a script lefutási ideje nincs hatással a mért sebesség értékére.

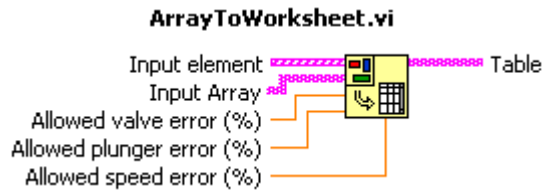
A *PlungerImageAquisition* a 32. ábrán látható módon kétszer kerül meghívásra, a két futtatás között a mozgást megelőzően beállított sebességtől függő késleltetés van. Így a két kép elkészülése között nem csak néhány ms telik el, hanem akár 6-7 másodperc is, amely így kiértékelhető és nagyobb pontosságú eredményt ad.



32. ábra A sebességmérést végző blokkdiagram részlet

4.3.12 Mérési eredmények kiértékelése

A mérési eredmények kiértékelését az 33. ábrán látható SubVI végzi. Az *Input Array* bemenetén lehet meghatározni, hogy hány darab elem feldolgozását végezze el a blokk.



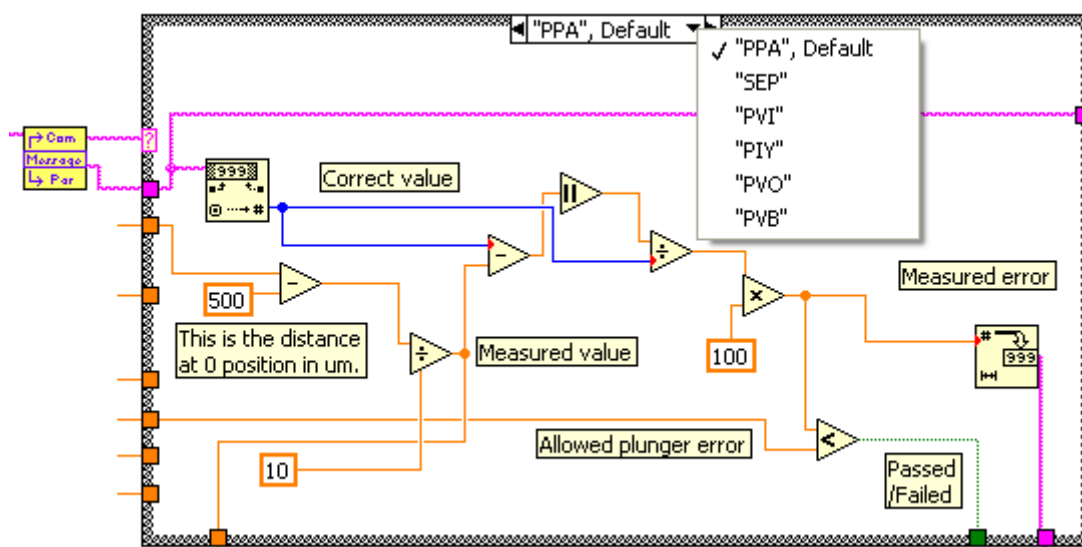
33. ábra Mérési eredmények feldolgozását végző SubVI

Az *Input Array* bemeneten kerülnek átadásra a CAN buszról kiolvasott adatok, valamint a képfeldolgozás eredményei. A főprogram *Allowed error* (megengedett hiba) beállításában megadott értékek itt kerülnek felhasználásra.

A blokk kimenete egy stringeket tartalmazó táblázat (egy két dimenziós tömb). A táblázat minden egyes sora egy kiküldött parancshoz tartozó adatokat tartalmazza, melyek a következők:

- Kiküldött üzenet
- Várt válasz
- Fogadott válasz
- Válaszok összehasonlításának eredménye
- Helyes mérendő érték
- Mért távolság
- Mért elfordulás
- Mért sebesség
- Megengedett fecskendő mozgató kar hiba
- Megengedett szög hiba
- Megengedett sebesség hiba
- Tesztelési lépés eredménye (Átment/Elutasítva)

A SubVI a kamerákkal mért eredmények kiértékelését végző részlete az 34. ábrán látható.



34. ábra Mérési eredmények kiértékelés, relatív hiba számolás

A szelepmozgatásokat kivéve minden beállító, vagy mozgató parancs paramétere jelenti a helyes mérendő értéket. Mivel a parancsok a paraméterükkel együtt egy stringben vannak tárolva, ezért szét kell választani őket parancsnévre, illetve paraméterre. Ezt a feladatot végzi a fenti ábra jobb szélén lévő SubVI. Az így levágott parancsnév vezérli a switch – case blokkot, hogy milyen kiértékelést végezzen az adott utasításnál (vagyis, hogy szög, távolság, vagy sebesség mérést).

Amint az előbb már bemutatásra került, a mérési eredmények kiértékeléséhez nem kell ismerni a vizsgálathoz használt kamera felbontását, mivel a *Vision Assistant* szoftver segítségével a képfeldolgozás útján megszerzett eredményeket nem pixelben, hanem (beállítástól függően) μm -ben kapjuk meg a távolságmérés alkalmával. Enélkül a funkció nélkül a programban nekünk kellene elvégezni az (pixel- μm) átváltásokat, amely bonyolítaná a mérőrendszerben alkalmazott kamerák esetleges más felbontására való cseréjét.

A switch - case-ben az egyes elemek jelentése:

- PPA - MOVE PLUNGER TO ABSOLUTE POSITION (a fecskendő mozgató kar abszolút pozícióba való mozgatása)
- SEP - SET THE PLUNGER DRIVE END SPEED (a kar mozgási sebességének beállítása)
- PIY - INITIALIZE PLUNGER AND VALVE DRIVE (szelep és kar inicializálása)

- PVI - TURN VALVE TO INPUT (szelep bemeneti irányba kapcsolása)
- PVO - TURN VALVE TO OUTPUT (szelep kimeneti irányba kapcsolása)
- PVB - TURN VALVE TO BYPASS (szelep bypass irányba kapcsolása)

A mért relatív hiba kiszámolásának menete:

$$h_{rel} = \frac{|X_h - X_{mért}|}{X_h} \cdot 100\% \quad , \text{ ahol } h_{rel} \text{ a relatív hiba \% -ban megadott értéke, } x_h \text{ a helyes}$$

érték, $x_{mért}$ a mért érték.

Amennyiben a LabVIEW-ban ez az érték nagyobb, mint az általunk megadott megengedett hiba értéke, akkor a tesztelési lépés eredménye "Failed" lesz, ellenkező esetben "Passed".

Az azonos típusú mérési eredmények pontosság és stabilitás értékek kiszámolása nem ebben a SubVI-ban történik, és nem is a LabVIEW-ban. Ezt a feladatot a PDF fájl készítését végző külső program végzi majd.

A blokk lefutásának eredménye, vagyis a tabulátorral tagolt szöveges fájl megtekinthető a [10] sz. mellékletben.

4.3.13 A kalibrációt végrehajtó cég adatainak beállítása a jegyzőkönyvben

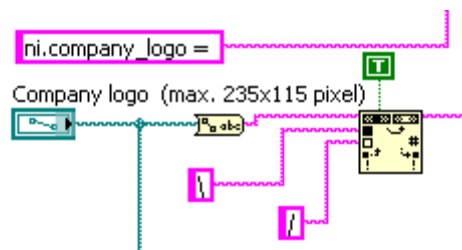
Ezzel a programrészsel a tesztelés lefuttatását megelőzően készíthető egy "properties" kiterjesztésű fájl, amelyben tárolható az összes jegyzőkönyvben feltüntetendő információ a mérés végrehajtójáról, illetve a tesztelés céljáról, leírásáról.

A választás azért erre a fájl típusra esett, mert ennek mind a létrehozása, mind pedig a későbbi feldolgozása (java környezetben) könnyen elvégezhető. A dokumentum létrehozásához csak a text fájl írásához szükséges blokkdiagramok szükségesek.

A fájl minden egyes sorának elején található a kulcs kifejezés (amellyel történik majd a tartalmára való hivatkozás), míg a baloldalon egyenlőségjellel elválasztva a hasznos tartalom, jelen esetben a cégalapok, logo, mérési leírás stb.

A dokumentum teljes tartalmának összefűzését a *Concatenate String* funkció végzi. Először a kulcs kifejezést és egy egyenlőségjelet, majd a string/path vezérlőkből beolvasott értéket, végül pedig egy sor vége konstanszt fűz össze, amelyek így a property fájl egy sorát képezik.

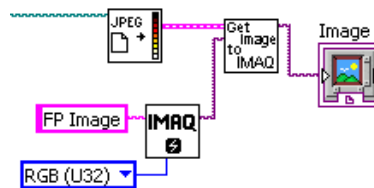
A 35. ábrán a "logo" elérési útjának megadása látható. Ez csak ezért került kiemelésre, mert az elérési út nem adható át ilyen formában a java alkalmazásnak. Ahhoz, hogy megfelelően tudja értelmezni a program a "\" karaktereket, ki kell cserélni "/" karakterekre.



35. ábra Logo elérési útjának megadása

A jegyzőkönyvön megjeleníteni kívánt "logo" megnyitására is lehetőség van a front panelen, ha esetleg a LabVIEW-s környezetben keresztül szeretnénk meggyőződni róla, hogy a megfelelő képfájlt választottuk-e ki.

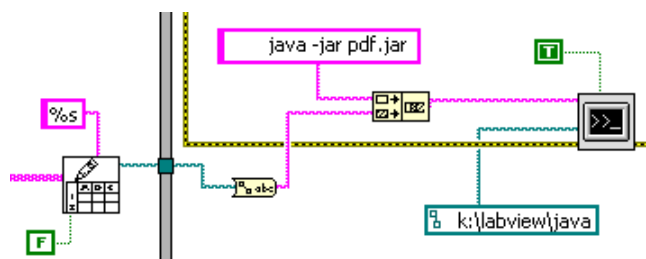
A képfájl megnyitását végző blokkdiagram részlet a 36. ábrán látható. A *GetImagetoIMAQ.vi* blokk az *image data* adattípust *Image* típusú alakítja, így válik megjeleníthetővé a kép a front panelen.



36. ábra JPEG fájl megnyitása *GetImagetoIMAQ* SubVI segítségével

4.3.14 Külső program futtatása

Külső program futtatását a *System Exec.vi* végzi, 37. ábra. Ez egy javas alkalmazás, amely futtatásához szükséges parancsok és a program munkakönyvtára fixen meg van adva a LabVIEW blokkdiagramban. Ezek változtatására nincs szükség a mérés indítása előtt. Az egyetlen változó paraméter a mérési eredményeket tartalmazó TSV kiterjesztésű fájl elérési útvonala. Ez a fájl bárhová menthető, hiszen a javas alkalmazás nem a saját munkakönyvtárában fogja keresni, hanem az általunk megadott útvonalon.



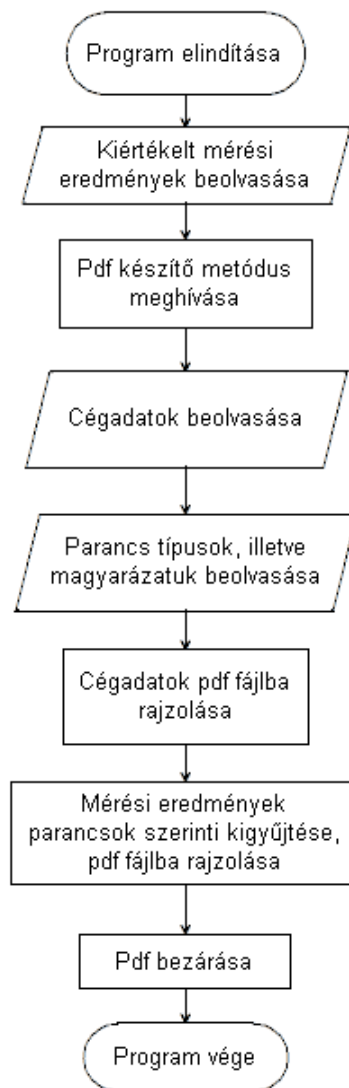
37. ábra *System Exec.vi* alkalmazása

4.4 Kiegészítő programok

4.4.1 PDF report készítése

Az alkalmazás az Eclipse SDK-3.7 fejlesztő környezetben készült. A PDF fájl kezelése az iText-5.0.5 library felhasználásával valósult meg [21].

4.4.1.1 Folyamatábra



38. ábra A java alkalmazás folyamatábrája

4.4.1.2 Működési leírás

A program három osztályból épül fel, amelyek az alábbiak:

- OpenTSV
- MeasuredItem
- PDFcreator

A *MeasuredItem.class* általános leírására szolgál a teszt eredményeknek. Tulajdonságai:

- Parancs neve
- Beolvasott üzenet
- Beolvasott üzenet azonos a megadottal
- Helyes mért érték
- Valós mért érték
- Megengedett hiba
- Mért hiba
- Tesztelési lépés eredménye

Az *OpenTSV.class* végzi a LabVIEW által megadott – elérési útvonalú – mérési eredményeket tartalmazó fájl megnyitását.

Az osztály elindulását követően példányosít egy *MeasuredItem* elemek tárolására alkalmas *ArrayList*-át. Majd a fájl soronkénti beolvasásakor hozzáad egy új *MeasuredItem* elemet. Természetesen – attól függően, hogy lekérdező, sebességbeállító, szelepmozgató, illetve hogy fecskendő kar mozgató mérési eredményt tartalmaz-e az éppen beolvasott sor – az egyes listaelemek különböző tulajdonsága lesz csak megadva. A fájl teljes beolvasását követően meghívja a *PDFcreator* osztály *createPdf* metódusát, melynek átadja a teszt eredményeket tartalmazó listát.

PDFcreator két property fájlból, valamint a *MeasuredItem*-eket tartalmazó listából hozza létre a PDF fájlt. A jegyzőkönyvnek a megelőző fejezetekben már bemutatott módon a fejléce a LabVIEW-ből konfigurálható be. Míg a dokumentum törzs létrehozásához a diluter firmware parancsait, illetve azok rövid magyarázatát a java alkalmazás a munkakönyvtárában található *commandNames.properties*-ből olvassa ki a program. A property fájl részlete a [11]sz. mellékletben található.

Az osztály meghívását követően beolvassa a beállításokat tartalmazó fájlokat, majd szerkesztésre létrehozza a *result.pdf* fájlt. Elsőkét a cégadatok és a logo kerül a dokumentumra, majd a mérés célja és rövid leírása.

A program ezt követően elemenként végigvizsgálja a mérési eredményeket tartalmazó listát, hogy van-e olyan eleme, amelynek a "Parancs neve" tulajdonsága megegyezik a parancsokat tartalmazó property fájl első elemével. Majd soronként léptetve a property fájlt, ismételten keresi az adott sorával azonos nevű objektumokat.

Ha talál egyezést, akkor azt egy táblázatba helyezve kirajzolja a dokumentum törzsére, a táblázat fölé írva egy rövid leírást (a *commandNames.properties* fájl alapján) az adott parancsról. Amennyiben több ugyanolyan parancsot talál, akkor azokat egy táblázatba rendezve jeleníti meg. Ha nincs találat, akkor az a parancs nem lesz megjelenítve a jegyzőkönyvben. Ennek köszönhetően csak olyan parancsok kerülnek kirajzolásra a dokumentumban, amelyek legalább egyszer el lettek küldve a tesztelés során. Vagyis a dokumentum törzse annak függvényében változik, hogy milyen parancsokat adtunk át mérőrendszernek.

A jegyzőkönyv végén kerülnek megjelenítésre az egyes mozgató parancs csoportokhoz tartozó pontosság és ismételhetőség értékei, melyek az alábbi képletek szerint kerülnek kiszámolásra:

- Ismételhetőség:

$$CV[\%] = \frac{s \times 100\%}{x_k}$$

ahol:

x_i – a mért érték

x_k – a közép érték

n – a mért értékek száma

s – a standard szórás

$CV[\%]$ – az ismételhetőség
(coefficient of variation)

$$s = \sqrt{\frac{\sum (x_i - x_k)^2}{n - 1}}$$

- Pontosság:

$$a = \frac{|X_v - X_k|}{X_v} \cdot 100\%$$

ahol:

a – a pontosság %-ban

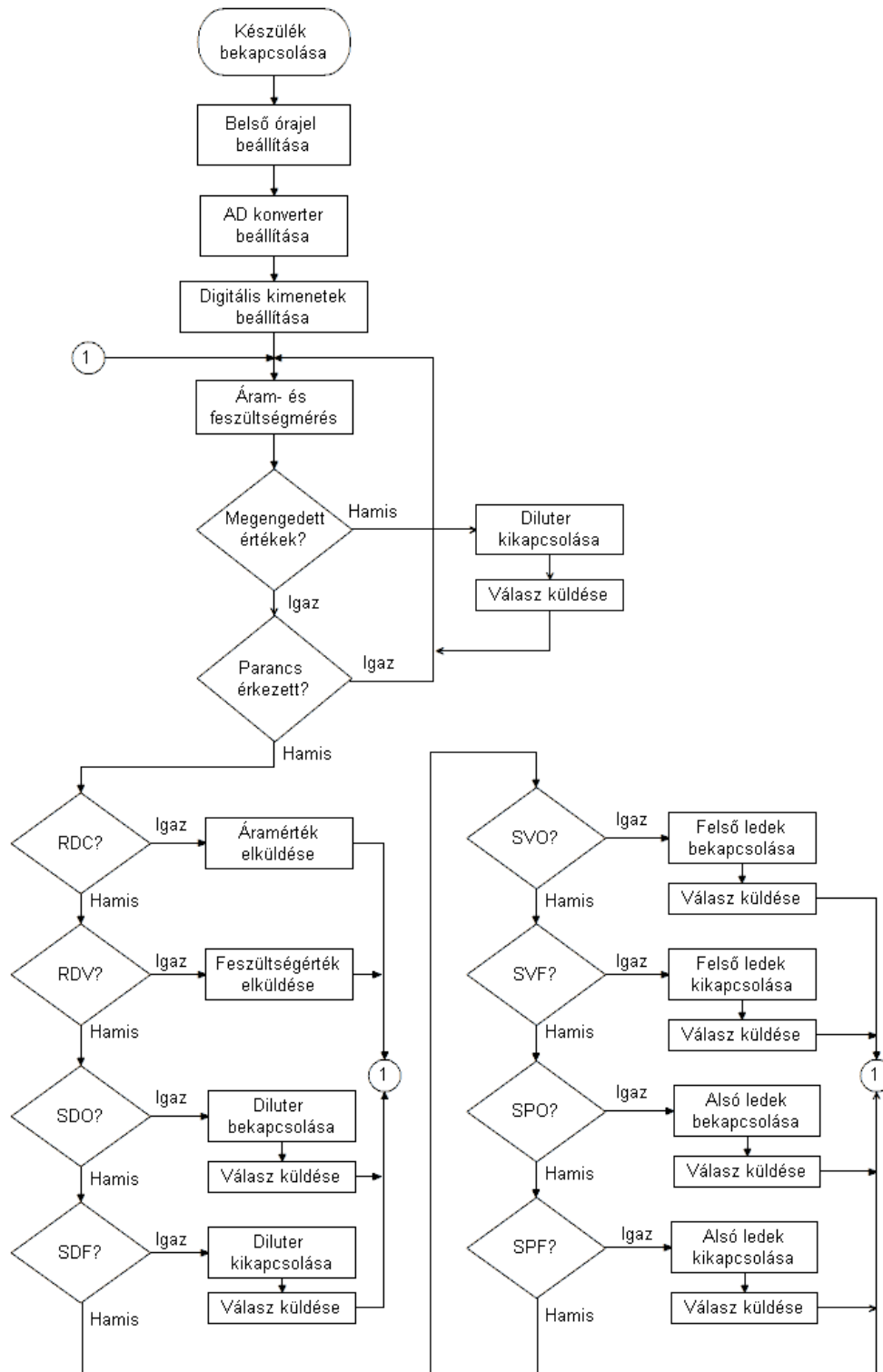
x_v – a várt (helyes) érték

x_k – a közép érték

Egy ilyen jegyzőkönyv megtekinthető [12] sz. mellékletben.

A mikrokontrollert vezérlő program az MPLAB IDE v8.80 fejlesztő környezetben készült, Hi-tech library felhasználásával.

4.4.2.1 Folyamatábra



39. ábra A mikrovezérlőn futó program folyamatábrája

4.4.2.2 *Működési leírás*

A mikrokontroller belső órajelről működik, ezért a bekapcsolást követően elsőként ennek beállítása történik meg. Majd a diluter áram- és feszültségértékeinek méréséhez az AD konverter konfigurálása következik, ezt követően pedig a LED-ek bekapcsolásához pedig a portok kimenetnek állítása hajtódik végre.

A regiszterek beállítását követően a program egy végtelen ciklusba ér, ahol minden alkalommal megvizsgálja, hogy a diluter felvett árama kisebb-e, mint 700 mA, illetve, hogy a rajta eső feszültség nem nagyobb, mint 26 V. Amennyiben valamelyik paraméter átlépi a megengedett értéket a diluter kikapcsolásra kerül. A kontroller minden ciklusban vizsgálja a bejövő üzeneteket.

Tekintettel arra, hogy az áram- és feszültségértékek – amennyiben azok a megengedett határok között vannak – nem játszanak szerepet a tesztelés kiértékelésében, hanem csak védelmi funkciót látnak el, ezért nem szükséges a pontos meghatározásuk. Mivel a feszültséget maximálisan 0,2 V, az áramot pedig 10 mA pontossággal elegendő mérni, ezért a felhasznált 10 bites AD konvertereknek csak a felső 8 bitjei lesznek feldolgozva.

A kontroller a következő üzeneteket (parancsokat) képes feldolgozni:

- RDC – Report Diluter Current (Diluter áramának lekérdezése)
- RDV – Report Diluter Voltage (Diluter feszültségének lekérdezése)
- SDO – Set Diluter On (Diluter bekapcsolása)
- SDF – Set Diluter oFf (Diluter kikapcsolása)
- SVO – Set Valve led On (Szelepet megvilágító LED bekapcsolása)
- SVF – Set Valve led oFf (Szelepet megvilágító LED kikapcsolása)
- SPO – Set Plunger led On (Kart megvilágító LED bekapcsolása)
- SPF – Set Plunger led oFf (Kart megvilágító LED kikapcsolása)

A parancsokra a következő válaszokat küldi:

- DON – Diluter ON (Diluter bekapcsolt állapotban)
- DOF – Diluter Off (Diluter kikapcsolt állapotban)
- VLO – Valve Led On (Szelepet megvilágító LED bekapcsolva)
- VLF – Valve Led oFf (Szelepet megvilágító LED kikapcsolva)
- PLO – Plunger Led On (Kart megvilágító LED bekapcsolva)
- PLF – Plunger Led oFf (Kart megvilágító LED kikapcsolva)

- ERR – ERRor diluter off (Hibából eredően a Diluter kikapcsolt állapotban)

5 Összefoglalás, elemzés

A dokumentáció készítésekor a sebesség mérés eredményeit kiértékelő programrész és a jegyzőkönyv készítő program stabilitás és pontosság kiszámolását végző függvényei még fejlesztés alatt álltak.

A mérőrendszerben felhasznált USB kamerák kiválasztásánál a dolgozat készítésekor az alacsony ár volt a szempont. A kamerákkal szemben nem az volt az elvárás, hogy képesek legyenek pontos és alacsony bizonytalanságú mérések végzésére. Ez látható a mellékletben szereplő mérési eredményekben is. Ezen készülékek alkalmazásának az volt a célja, hogy a mérőrendszer komplex funkcionális működése bemutatható legyen. Természetesen amennyiben a vizsgáló állomás tényleges felhasználásra kerül, akkor feltétlenül szükséges ezeket néhányszor nagyobb felbontású eszközökre cserélni. Ez a csere a program minimális megváltoztatásával jár.

Ugyan a sebességmérést végző programrész még nem készült el teljesen, az azonban az eddigi tapasztalatok alapján is látható, hogy a felhasznált USB kamerák nem alkalmasak pontos sebesség mérésre. Ahhoz, hogy ez a tesztelési lépés is pontosan elvégezhető legyen, valamint minimális felhasználói műveletvégzéssel járjon, két továbbfejlesztési irány jöhet szóba. Az egyik lehetséges megoldás az optokapuk felhasználása, a másik pedig a karra szerelt gyorsulásérzékelők alkalmazása. Ez utóbbi esetén a sebesség értéke a gyorsulás idő szerinti integráljából volna kiszámolható.

Véleményem szerint a vizsgálóállomás alkalmas a bevezetőben kitűzött célok elvégzésére, vagyis megfelelő felbontású és pontosságú kamerák felhasználásával elvégezhető a diluterek minősítése. A felhasználók által definiálható parancsfájloknak köszönhetően sokoldalúan is felhasználható a mérőrendszer (lehet kizárólagosan csak sebesség, karmozgató, vagy szelepmozgató méréseket végezni rajta). Valamint gyorsan lekérdezhető az összes EEPROM-ban tárolt információ a készülékekről, amelyeket TSV vagy PDF formátumú fájlokban tárolhatunk a számítógépünkön.

6 Abstract

The aim of my thesis was to build a measuring system which based on image acquisition and can perform a total, quick test without or minimal user intervention requirements. Because of the image process this system has a lot of advantages for example don't have to disassemble the device for the test, easy to increase accuracy of the measurement with a higher resolution camera, etc.

When I was writing the thesis the measure of speed, precision and accuracy functions of the report hadn't been completed.

The lowest price was the main aspect when I choosed the cameras. Unfortunately the system can't execute high accuracy measurements with these devices, but it wasn't an expectation. I used these low qualitative cameras because this was the cheapest solution to introduce how is this complex system working. Of course if we would like to use the system for high accuracy measurement we will have to change these cameras for higher resolution ones.

Tut, the full measure of speed haven't been completed, but it looks like the USB camera is not able to do it with low relative fault. To this task can be carried out with low imprecision and minimal user intervention there are two possible solutions. The first is to use opto-gates and the second is to use crash-sensors. In second case the value of speed could be calculated from the value of acceleration.

In my opinion the measuring system is suitable for perform functional test of dilter, if we suit high resolution cameras.

Because of the user defined CAN command files the system can be used different ways, for example we can do just speed, valve motor or plunger motor testing, etc. Furthermore all of the non-volatile parameters can be easily and quite required from the device. At the end of the acquisition every steps of the measurement is documented in a TSV and PDF file.

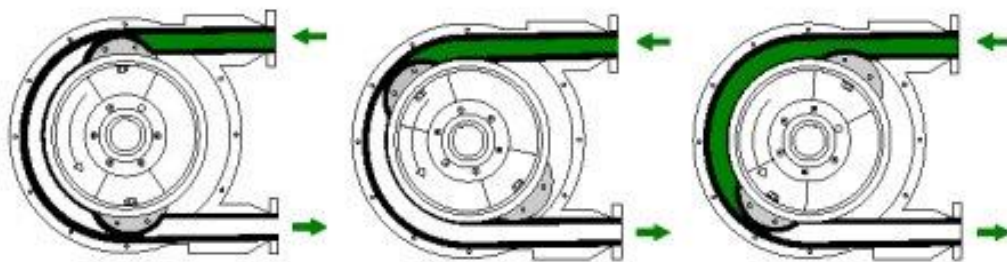
Az internetről származó források 2011. november 28-én elérhetőek voltak.

7 Irodalomjegyzék

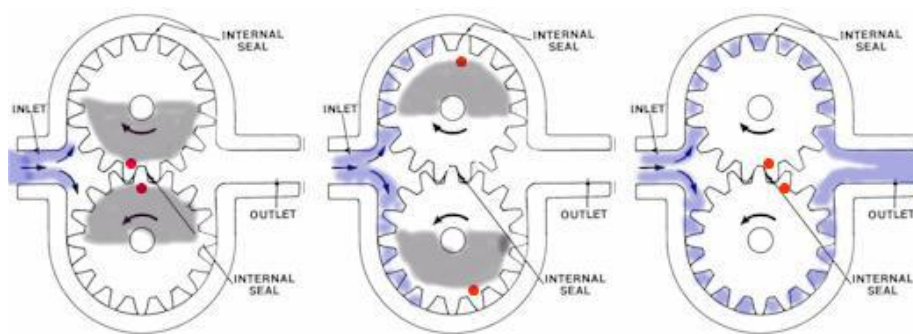
- [1] Liquid handling guide, TECAN Schweiz AG, 2011
- [2] <http://www.laboratoryequipmentworld.com/liquid-handling-system.html>
- [3] http://www.muszeroldal.hu/search_result.php?SearchDB=03&SearchType=A3&SearchText=PIPETTA&id=41
- [4] AN713 - Microchip Technology Inc., 1999
- [5] <http://www.fsz.bme.hu/traficcc/can/can.html>
- [8] CAN Specification Version 2.0, Robert Bosch GmbH, 1991
- [6] <http://www.orangesoft.hu/halozat-alapok/21>
- [7] <http://www.vlsi-world.com/content/view/65/47/>
- [9] http://ob121.com/bus_can.html
- [10] Image Acquisition and Processing with LabVIEW, Christopher G. Relf, 2004
- [11] <http://itl7.elte.hu/hlabdb.regi/kepfeld/kepf.html>
- [12] <http://www.digkep.hu/staff/BJ/Aphd/publphd.htm>
- [13] <http://www.orszaczky.hu/mvcam.html>
- [14] <http://www.ivst.hu/kamera-rendszerek/minosegellenorzes-ipari-kepfeldolgozas>
- [15] Instrument Software V4.6 Software Manual, TECAN Schweiz AG, 2003
- [16] Gravimetric pipetting precision check instructions, TECAN Schweiz AG, 1997
- [17] <http://zone.ni.com/devzone/cda/tut/p/id/3763>
- [18] <http://digital.ni.com/public.nsf/allkb/E1A44077DA6C819186257288006B76D7>
- [19] NI-IMAQdx user manual, National Instruments, 2006
- [20] Képfeldolgozás, Székely Vladimír, 2005
- [21] <http://api.itextpdf.com/itext/>

8 Mellékletek

[1] melléklet

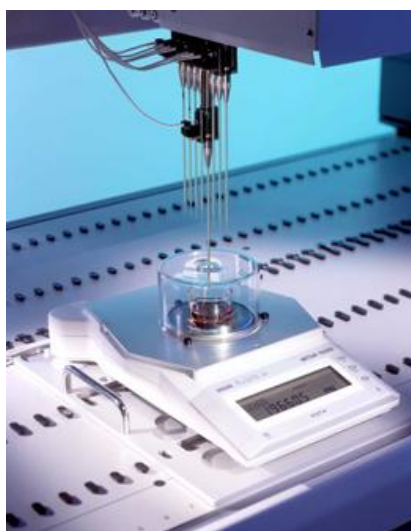


Perisztaltikus pumpa



Fogaskerékes szivattyú

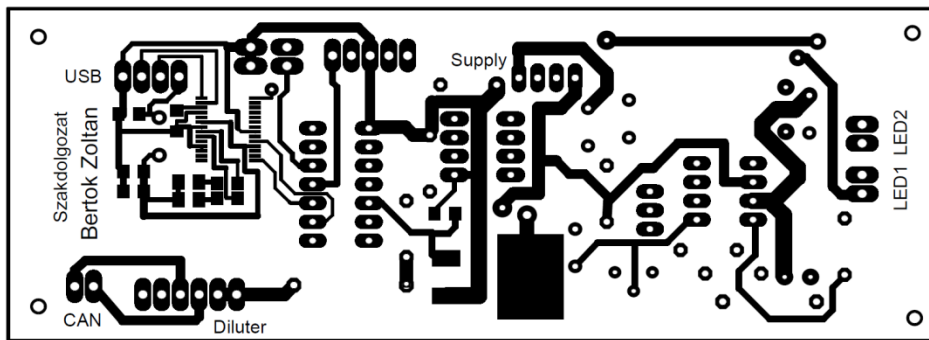
[2] melléklet



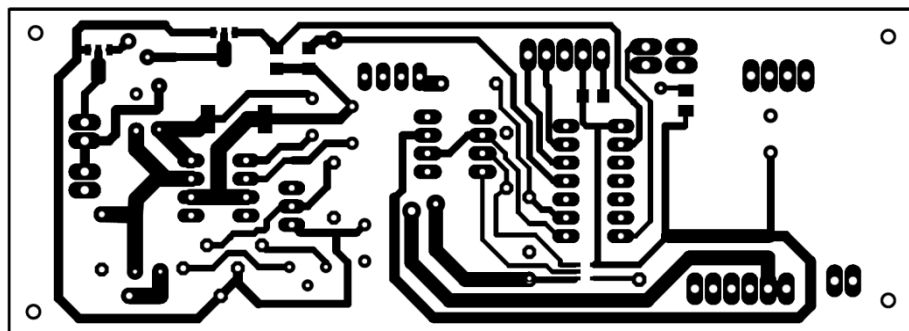
Mettler analitikai mérleg, a gravimetrikus teszt elvégzéséhez.

Part	Value	Device	Package	Library	Sheet
C1	1u	KERAMIA_KONDI	KER_KONDI	my_components	1
C2	100n	KERAMIA_KONDI	KER_KONDI	my_components	1
C3	100n	KERAMIA_KONDI	KER_KONDI	my_components	1
C4	10n	KERAMIA_KONDI	KER_KONDI	my_components	1
C5	10nF	C-EUC1206	C1206	rc1	2
C6	100nF	C-EUC1206	C1206	rc1	2
C7	100nF	C-EUC1206	C1206	rc1	2
C8	4,7uF	C-EUC1206	C1206	rc1	2
C10	100u	8X15	8X15	my_components	1
CAN2	2.54mm PIN	MPT2	2POL254	con-phoenix-254	2
D1	SK13B	SCMB36	DO214AB	my_components	1
DILUTER	2.54mm PIN	MPT6	6POL254	con-phoenix-254	2
I/O-GND	2.54mm PIN	MPT2	2POL254	con-phoenix-254	2
I/O1	2.54mm PIN	MPT2	2POL254	con-phoenix-254	2
IC1	MAX5035D	MAX5035D	DIL-08	my_components	1
IC2	FT232RL	FT232RL	SSOP28	ftdichip	2
IC6	PIC12F1823	PIC12F1823	DIL14	my_components	2
IC7	MCP601P	MCP601P	DIL08	linear	2
L	Ferrit bead	BL02RN1	BL02RN1	inductors	2
L1	220u	FEKVO_TEKERCS2	FEKVO_TEKERCS2	my_components	1
LED1	2.54mm PIN	MPT2	2POL254	con-phoenix-254	1
LED2	2.54mm PIN	MPT2	2POL254	con-phoenix-254	1
OPA2	INA138	INA138	SOT-23	my_components	2
PROG	2.54mm PIN	MPT5	5POL254	con-phoenix-254	2
Q4	STD20NF06	IRF3704S	D2PACK	transistor-power	2
R1	20k	R-EU_0204/7	0204/7	rc1	2
R2	18k	R-EU_0204/7	0204/7	rc1	1
R3	1k8	R-EU_0204/7	0204/7	rc1	1
R4	39k	R-EU_0204/7	0204/7	rc1	1
R5	33k	R-EU_0204/7	0204/7	rc1	1
R6	0R27	R-EU_0411/12	0411/12	rc1	1
R7	10k	R-TRIMM3296Y	RTRIM3296Y	rc1	1
R8	2,2k	R-EU_0204/7	0204/7	rc1	2
R9	0R16	R-EU_0411/15	0411/15	rc1	2
R10	10k	R-EU_0204/7	0204/7	rc1	2
R11	270R	R-EU_M1206	M1206	rc1	2
R12	10k	R-EU_R1206	R1206	rc1	2
R13	10k	R-EU_M1206	M1206	rc1	2
R14	10k	R-EU_R1206	R1206	rc1	1
R15	10k	R-EU_R1206	R1206	rc1	1
R16	270R	R-EU_M1206	M1206	rc1	2
R17	10k	R-EU_R1206	R1206	rc1	2
REF1	LM385-1.2	LM385-1.2	TO-92	my_components	1
SUPPLY	2.54mm PIN	M04	04P	con-amp-quick	2
T1	BDP949	BCX56-NPN-SOT89-BCE	SOT89-BCE	transistor	1
T2	BDP949	BCX56-NPN-SOT89-BCE	SOT89-BCE	transistor	1
U2	2LED1PAC	2LED1PAC	2LED_IN1PAC	my_components	2
USB	2.54mm PIN	MPT4	4POL254	con-phoenix-254	2

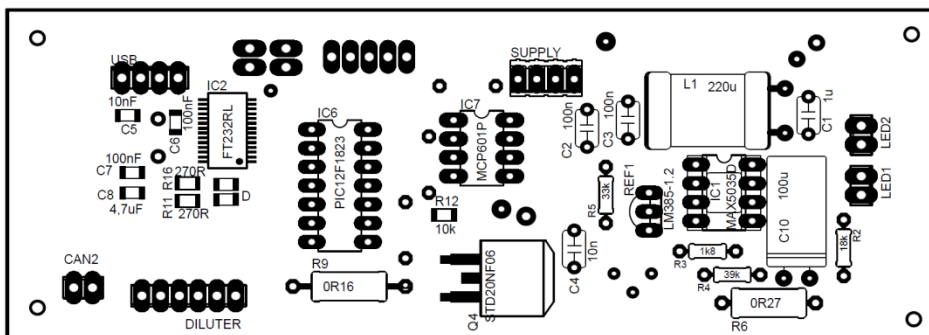
[6] melléklet



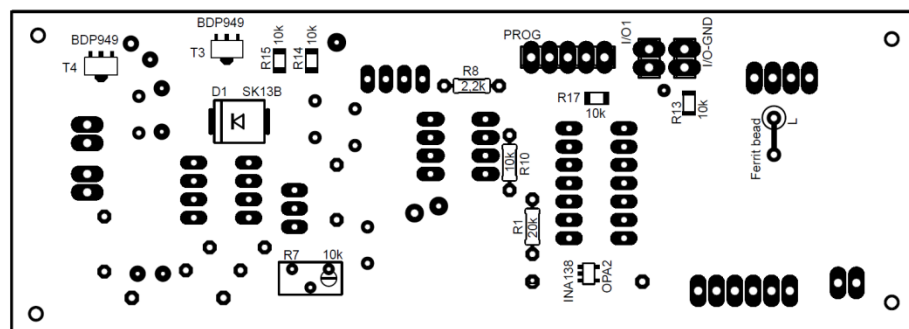
Nyomtatott áramkör top réteg



Nyomtatott áramkör bottom réteg

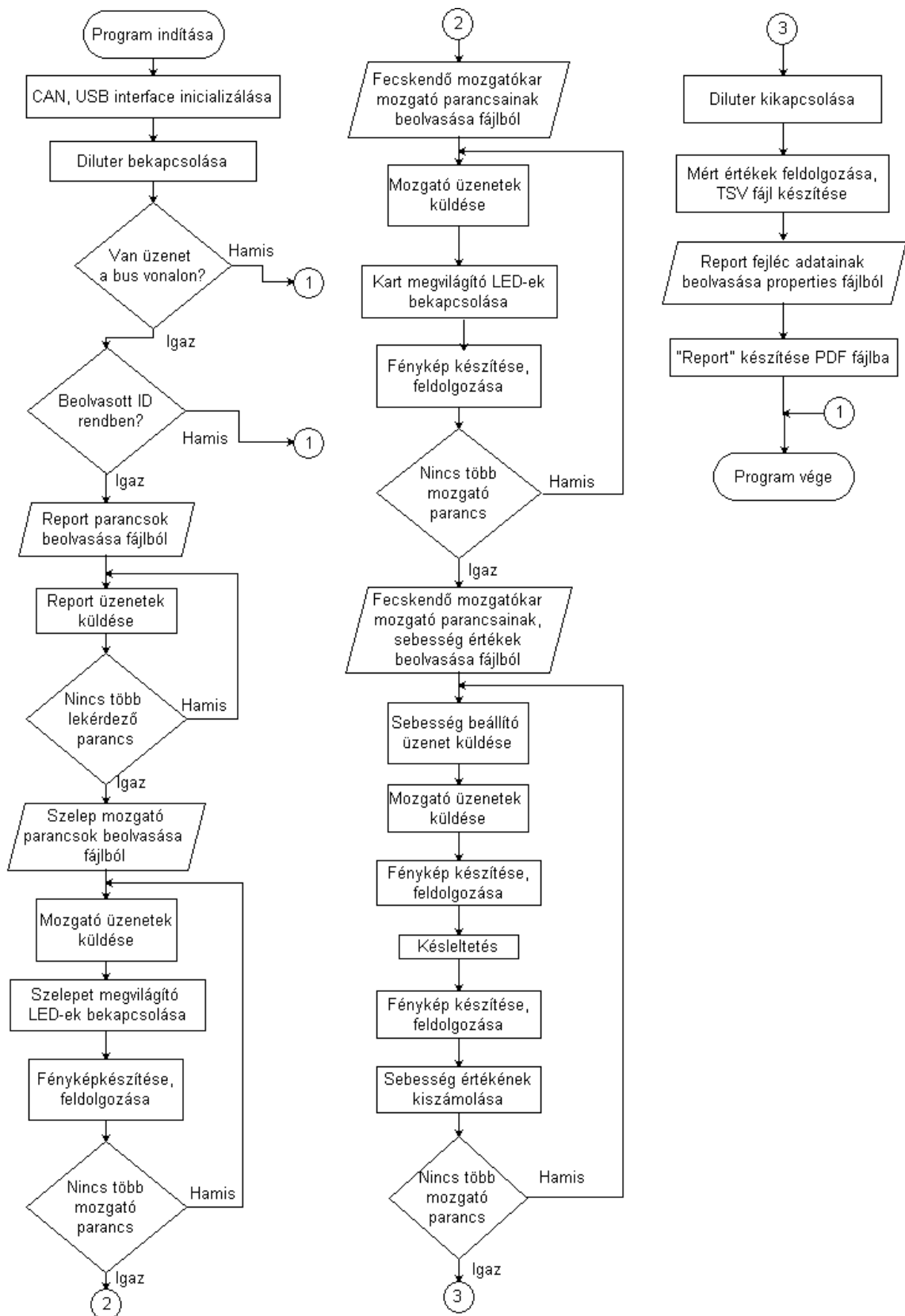


Top réteg beültetési rajz



Bottom réteg beültetési rajz

[7] melléklet



A mérőrendszer átfogó vázlatos folyamatábrája

[8] melléklet

	A	B	C	D
1	Transmitted data	Received data 1	Received data 2	Received data 3
2	RFV			
3	RFV1			
4	RFV2			
5	RSD			
6	RSD1			
7	RYV			
8	RPP			

Lekérdező (Report) parancsok

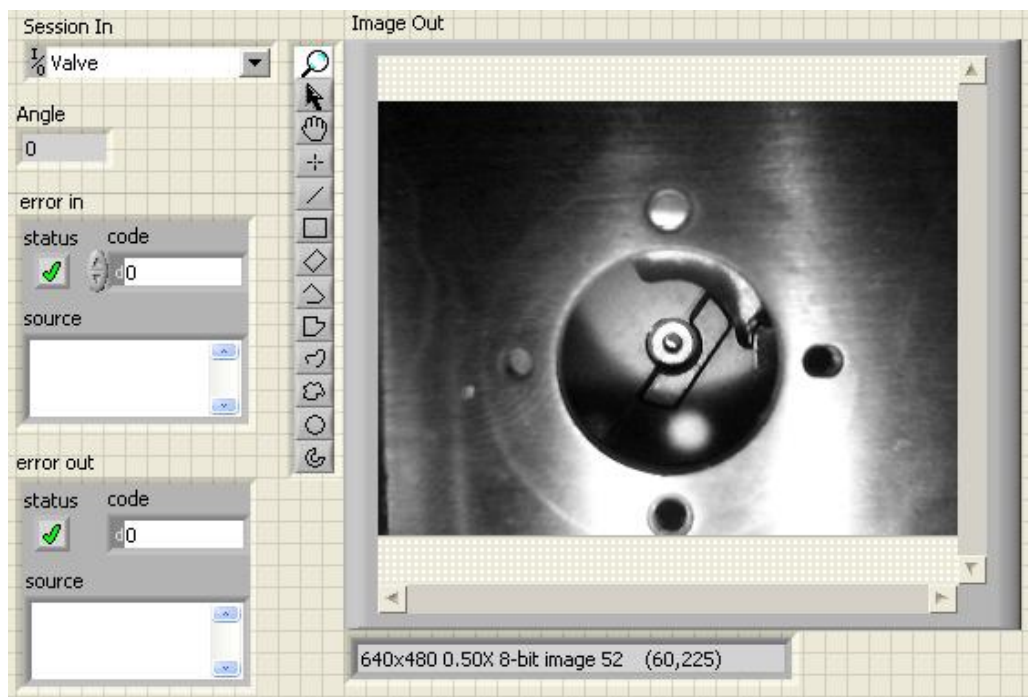
	A	B	C	D
1	Transmitted data	Received data 1	Received data 2	Received data 3
2	PIY		2060	
3	PVI		2060	
4	PVO		2060	
5	PIY		2060	
6	PVO		2060	
7	PVB		2060	
8	PVI		2060	

Szelepmozgató parancsok

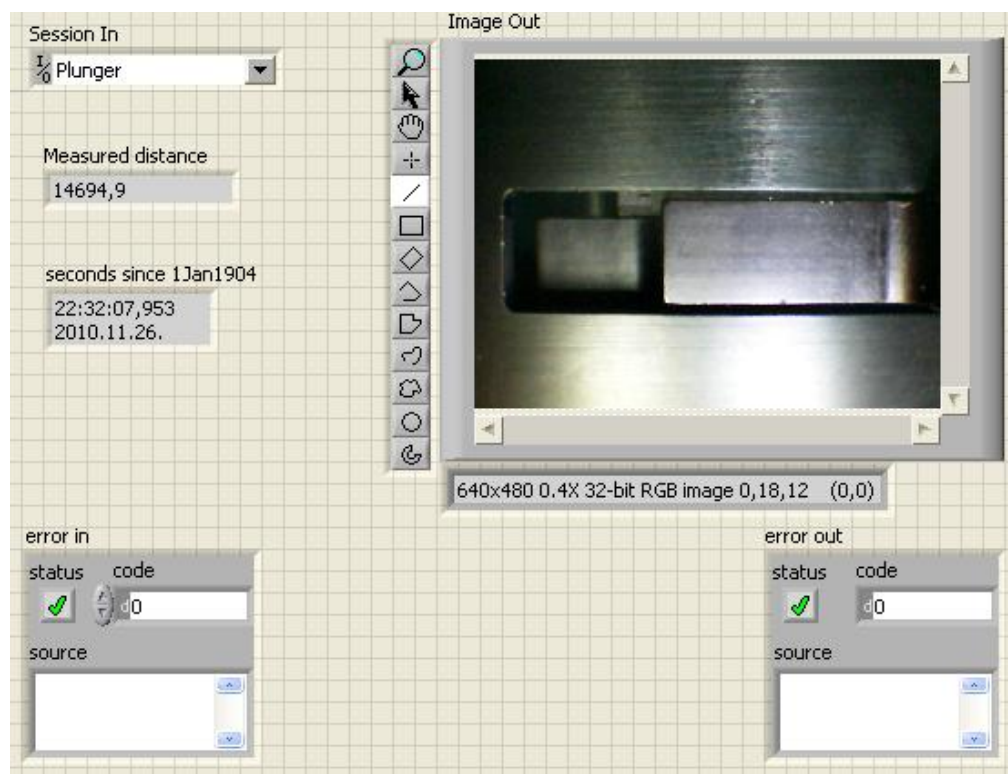
	A	B	C	D
1	Transmitted data	Received data 1	Received data 2	Received data 3
2	PPA0		2060	
3	PPA3125		2060	
4	PPA0		2060	
5	PPA100		2060	
6	PPA200		2060	
7	PPA300		2060	
8	PPA400		2060	

Fecskendőmozgató parancsok

[9] melléklet



A ValveImageAcquisition.vi előlapja



A PlungerImageAcquisition.vi előlapja

[10] melléklet

18	RPW	-1	`133	-1	-1	-1	-1	-1	-1	-1	-1	-1
19	RPN	-1	0'0	-1	-1	-1	-1	-1	-1	-1	-1	-1
20	RDV	-1	`100	-1	-1	-1	-1	-1	-1	-1	-1	-1
21	RDB	-1	`0	-1	-1	-1	-1	-1	-1	-1	-1	-1
22	RDA	-1	`0	-1	-1	-1	-1	-1	-1	-1	-1	-1
23	RRP	-1	`0	-1	-1	-1	-1	-1	-1	-1	-1	-1
24	RDF	-1	`987	-1	-1	-1	-1	-1	-1	-1	-1	-1
25	RDF1	-1	`234859	-1	-1	-1	-1	-1	-1	-1	-1	-1
26	RDF2	-1	`1876	-1	-1	-1	-1	-1	-1	-1	-1	-1
27	RDF3	-1	`217038	-1	-1	-1	-1	-1	-1	-1	-1	-1
28	RDF4	-1	`114481	-1	-1	-1	-1	-1	-1	-1	-1	-1
29	RED	-1	`1024	-1	-1	-1	-1	-1	-1	-1	-1	-1
30	PIY	`	`	OK	320	-1	320,729	-1	5	5	5	0,23 Passed
31	PVI	`	`	OK	80	-1	92,825	-1	5	5	5	16,03 Failed
32	PVO	`	`	OK	320	-1	318,221	-1	5	5	5	0,56 Passed
33	PIY	`	`	OK	320	-1	322,11	-1	5	5	5	0,66 Passed
34	PVO	`	`	OK	320	-1	319,042	-1	5	5	5	0,3 Passed
35	PVB	`	`	OK	200	-1	200,475	-1	5	5	5	0,24 Passed
36	PVI	`	`	OK	80	-1	89,221	-1	5	5	5	11,53 Failed
37	PVO	`	`	OK	320	-1	315,177	-1	5	5	5	1,51 Passed
38	PVI	`	`	OK	80	-1	87,274	-1	5	5	5	9,09 Failed
39	PIY	`	`	OK	320	-1	310,229	-1	5	5	5	3,05 Passed
40	PVO	`	`	OK	320	-1	319,042	-1	5	5	5	0,3 Passed
41	PVB	`	`	OK	200	-1	199,945	-1	5	5	5	0,03 Passed
42	PVI	`	`	OK	80	-1	91,503	-1	5	5	5	14,38 Failed
43	PVO	`	`	OK	320	-1	324,239	-1	5	5	5	1,32 Passed
44	PVI	`	`	OK	80	-1	91,451	-1	5	5	5	14,31 Failed
45	PIY	`	`	OK	320	-1	324,239	-1	5	5	5	1,32 Passed
46	PPA100	`	`	OK	100	104,302	-1	-1	5	5	5	4,302 Passed
47	PPA3125	`	`	OK	3125	3143,756	-1	-1	5	5	5	0,6 Passed
48	PPA700	`	`	OK	700	694,891	-1	-1	5	5	5	0,73 Passed
49	PPA800	`	`	OK	800	798,226	-1	-1	5	5	5	0,22 Passed
50	PPA900	`	`	OK	900	885,001	-1	-1	5	5	5	1,67 Passed
51	PPA1300	`	`	OK	1300	1293,677	-1	-1	5	5	5	0,49 Passed

Részlet a mérési eredményeket tartalmazó TSV fájlból.

Az oszlopok balról jobbra haladva: Kiküldött üzenet, Várt válasz, Fogadott válasz, Válaszok összehasonlításának eredménye, Helyes mérendő érték, Mért távolság, Mért elfordulás, Mért sebesség, Megengedett fecskendő mozgató kar hiba, Megengedett szög hiba, Megengedett sebesség hiba, Tesztelési lépés eredménye (Átment/Elutasítva)

[11] melléklet

```
RFV=REPORT FIRMWARE VERSION
RFV0=REPORT FIRMWARE VERSION
RFV1=REPORT BOOTWARE VERSION
RFV2=REPORT SERIAL NUMBER
RFP=REPORT PIEZO MODULE FIRMWARE VERSION
RSD=REPORT SYSTEM DEVICES - Low volume installed
RSD0=REPORT SYSTEM DEVICES - Low volume installed
RSD1=REPORT SYSTEM DEVICES - Piezo Module installed
RYV=REPORT SYRINGE VOLUME
RYV0=REPORT SYRINGE VOLUME - Report current position in mnemonics
RYV3=REPORT SYRINGE VOLUME - Report lost steps.
RYV11=REPORT SYRINGE VOLUME - Report high current value in increments.
RPP=REPORT PLUNGER DRIVE PARAMETER - Report calculated position in steps
RPP0=REPORT PLUNGER DRIVE PARAMETER - Report calculated position in steps
RPP1=REPORT PLUNGER DRIVE PARAMETER - Report current position in steps
RPP2=REPORT PLUNGER DRIVE PARAMETER - Report encoder position in steps
RPP3=REPORT PLUNGER DRIVE PARAMETER - Report encoder deviation in steps.
RPP4=REPORT PLUNGER DRIVE PARAMETER - Report initialization offset in steps
RPP5=REPORT PLUNGER DRIVE PARAMETER - Report absolute range in steps
RPP6=REPORT PLUNGER DRIVE PARAMETER - Report end speed in Hz
RPP7=REPORT PLUNGER DRIVE PARAMETER - Report start speed in Hz
RPP8=REPORT PLUNGER DRIVE PARAMETER - Report stop speed in Hz
RPP10=REPORT PLUNGER DRIVE PARAMETER - Report ramp frequency in 2.5kHz increments
RPP11=REPORT PLUNGER DRIVE PARAMETER - Report high current value in increments
RPP12=REPORT PLUNGER DRIVE PARAMETER - Report low current value in increments
RIS=REPORT INSTALLED PINCH VALVE SOLENOID (0 = no pinch valve installed, 1 = pinch
valve installed)
RTH=REPORT TIME TO HIT PINCH VALVE SOLENOID
RBH=REPORT TIME BEFORE HIT PINCH VALVE SOLENOID
RAH=REPORT TIME AFTER HIT PINCH VALVE SOLENOID
RPS=REPORT PIEZO MODULE STATUS
ROV=REPORT OUTPUT VOLUME
RRV=REPORT REST VOLUME
RPV=REPORT PIEZO MODULE VOLTAGE - target voltage
RPV0=REPORT PIEZO MODULE VOLTAGE - target voltage
RPV1=REPORT PIEZO MODULE VOLTAGE - actual voltage
RPF=REPORT THE PULSE FREQUENCY FOR PIEZO AKTUATION - frequency
RPF0=REPORT THE PULSE FREQUENCY FOR PIEZO AKTUATION - frequency
RPF1=REPORT THE PULSE FREQUENCY FOR PIEZO AKTUATION - frequency in counts.
RPW=REPORT THE PULSE WIDTH FOR PIEZO AKTUATION - pulse width in µs
RPW0=REPORT THE PULSE WIDTH FOR PIEZO AKTUATION - pulse width in µs
RPW1=REPORT THE PULSE WIDTH FOR PIEZO AKTUATION - pulse width in counts
RPN=REPORT PULSE NUMBER OF PIEZO MODULE
RDV=REPORT PIEZO DROP VOLUME
RDB=REPORT DELAY BEFORE RUN
RDA=REPORT DELAY AFTER RUN
RRP=REPORT LLD/PIEZO RELAY POSITION
RDF=REPORT DIAGNOSTIC FUNCTIONS - Report number of power ups
RDF0=REPORT DIAGNOSTIC FUNCTIONS- Report number of power ups
RDF1=REPORT DIAGNOSTIC FUNCTIONS - Report minute meter
RDF2=REPORT DIAGNOSTIC FUNCTIONS - Report number of initializations
RDF3=REPORT DIAGNOSTIC FUNCTIONS - Report number of plunger movements
RDF4=REPORT DIAGNOSTIC FUNCTIONS - Report number of valve movements
RED=READ EEPROM DATA
SYV=SET SYRINGE VOLUME
SEP=SET THE PLUNGER DRIVE END SPEED
STP=SET THE PLUNGER DRIVE START SPEED
SPP=SET THE PLUNGER DRIVE STOP SPEED
```

A diluter által értelmezett parancsok listáját, valamint jelentésüket tartalmazó
commandNames.properties fájl tartalmának részlete

[12] melléklet

REPORT

Company name (tested by)	University of Obuda	
Address	Tavaszmézo u. 15-17.	
City	Budapest	
Zipcode	1084	
Phone	-	
Tester	Zoltan Bertok	
Date	2011.10.31. 10:14	

Cégadatok és logo megjelenítése a jegyzőkönyvben

REPORT FIRMWARE VERSION (RFV)

Number	Received data
1	`VCC-V1.20-06/2000

REPORT BOOTWARE VERSION (RFV1)

Number	Received data
1	`CC-BOOT-V1.20-09/99

Lekérdező üzenetek megjelenítése

MOVE PLUNGER TO ABSOLUTE POSITION (PPA)

Number	Received	Received compare	Correct value	Measured value	Allowed error (%)	Measured error (%)	Test result
1	`	OK	100	104,302	5	4,302	Passed
2	`	OK	3125	3143,756	5	0,6	Passed
3	`	OK	700	694,891	5	0,73	Passed
4	`	OK	800	798,226	5	0,22	Passed
5	`	OK	900	885,001	5	1,67	Passed
6	`	OK	1300	1293,677	5	0,49	Passed

Mozgató parancsok és mért, valamint feldolgozott értékeinek kilistázása