Budapesti Műszaki és Gazdaságtudományi Egyetem
Közlekedésmérnöki és Járműmérnöki Kar
Közlekedés- és Járműirányítási Tanszék

Járműmechatronikai vezérlőegységek oktatási célú emulációs keretrendszerének kifejlesztése

TDK Dolgozat 2013

Készítette: Gáldi Péter, járműmérnöki (Bsc) szak

Konzulens: Dr. Bécsi Tamás adjunktus

2013. november

Tartalomjegyzék

Rövidítések jegyzéke

ABS – Anti-lock Braking System

ACC – Adaptive Cruise Control

AFL – Adaptive Forward Lightning

AI – Analog Input

AO – Analog Output

BSA – Blind Spot Alert

CAN – Controller Area Network

CL – Central Lock

Cmd – Command

Cont – Control

DAU – Data Aquisition Unit

DI – Digital Input

DO – Digital Output

ECC – Electronic Cruise control

ECU – Electronic Control Unit

EEPROM – Eletronically Erasable Programmable ROM

ESP – Electronic Stability Program

EWL – Electronic Wind Lifter

GPS – Global Positioning System

HLC – Headlight Control

HMDS – Helmet Mounted Display System

LDW – Line Departure Warning

LIN – Local Interconnect Network

MCU – Micro Control Unit

MOST – Media Oriented Systems Transport

Op – Operability

PWM – Pulse-Width Modulation

Stat – Status

TCU – Trip Computer Unit

TS – Turn Signal

1. Bevezetés

A Budapesti Műszaki Egyetem Közlekedés- és Járműmérnöki Karán folyó Járműmérnök képzés Járműmechatronika szakirányához kapcsolódóan a diákok megismerkednek a járműveken alkalmazott hidraulikus, pneumatikus és elektromechanikus egységek jellemzőivel, ezek gépészeti, üzemeltetési, tervezési és vezérlési sajátosságaival. Megtanulják az eszközök irányításának, vezérlésének elméleti alapjait, valamint foglalkoznak ezek implementálásának gyakorlati kérdéseivel mind érzékelők, mind beavatkozó és vezérlő egységek esetén.

Az oktatás során a járművek különböző elektronikai vezérlő rendszereire való szoftverek fejlesztése a Járműfedélzeti rendszerek 2 tantárgy keretei között valósul meg, a dolgozat témája az ezekhez készült mikrokontrolleres vezérlő egységeket tesztelő rendszerek bemutatása. Ezen tesztrendszerek a különböző ki- és bemenetek szimulálásával képesek a teljes lehetséges eseményhorizontot (a különböző rendszerhibákat is ide értve) lefedni, abban a vezérlőegység megfelelő működését vizsgálni, hibáit felderíteni.

Annak érdekében, hogy a mérendő rendszerek autóipari viszonylatban történő elhelyezését megkönnyítsem, a dolgozat a fedélzeti elektronikus vezérlőberendezések rövid történetét is tartalmazza az első motorvezérlő elektronikáktól napjainkig. Ezen túlmenően kitekint az előttünk álló időszak kihívásaira a biztonsági és vezetőt segítő rendszerek területén, mely rendszerek tesztelési igényeinek növekedésével elkerülhetetlenné válnak az automatizált teszteljárások.

Ezt követően bemutatom az általam fejlesztett tesztrendszerek általános hardveres és szoftveres jellemzőit, ezek a tesztelt eszközökkel való összekötésének mikéntjét, valamint a legtöbb szoftverben közös funkcionalitásokat. Kiemelem a teljes eseményhorizont lefedéséhez szükséges hardveres és szoftveres implementációs kérdések megoldásait, valamint a későbbi analízishez szükséges, továbbá a szoftverbe épített különböző validációs és verifikációs lehetőségeket.

A dolgozat az egyes alkalmazások részletes leírásával, egyedi hardveres specifikációjával, a szoftver sajátosságainak bemutatásával folytatódik. Kitér a felhasználói felület elemeire, és ezek futás közbeni letiltására/engedélyezésére a teszt módjának függvényében, mely funkció jelentősen csökkenti a hardveres ütközés lehetőségét, a szoftver erőforrásigényének minimális növelésével. A fejezet továbbá

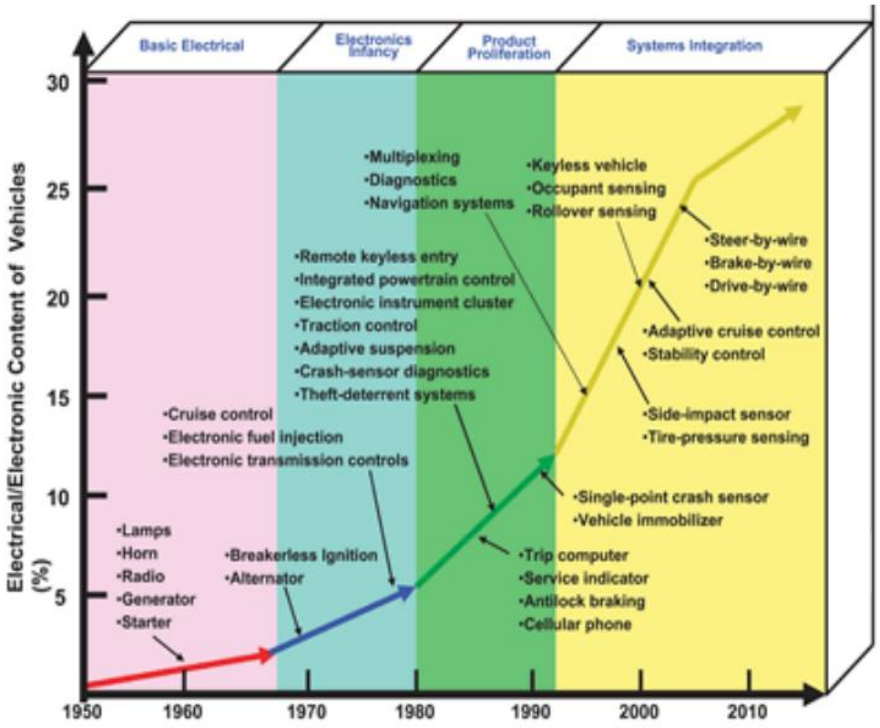
tartalmazza az adott alkalmazások specifikus jellemzőit, az általuk készített log fájlok bemutatását, és az automatikus tesztszekvenciák paramétereinek leírását.

2. Járműelektronika fejlődése [1]

A járművek fedélzetén hozzáférhető fedélzeti elektronika színvonala, komplexitása, teljesítménye és megbízhatósága az 1950-es évekbeli megjelenését követően folyamatosan nőtt, és ez a folyamat az elkövetkező évtizedekben sem látszik megtörni. Dolgozatomban ennek a fejlődésnek rövid történeti összefoglalását, a jelenlegi technológiák alapvető funkcióinak példákkal történő bemutatását, valamint az iparág várható fejlődésének irányát is kifejtem.

2.1. Rövid történelmi áttekintés

Az első, járművek fedélzetén megjelent egyszerű elektronikus berendezésektől a jelenleg elérhető összetett, vezetőt támogató, a jármű utasainak biztonságát aktív és passzív módon, valamint az általános komfortérzetet növelő intelligens rendszerek elterjedéséig közel öt évtized telt el. Ez alatt az idő alatt ezen rendszerek teljesítménye egyre nőtt, miközben folyamatosan szigorodó biztonsági előírásoknak felelnek meg.
Ezt a folyamatot az 1. ábra szemlélteti:



1. ábra Az autóelektronika fejlődése [2]

2.1.1. Korai fedélzeti rendszerek

Kezdetben a jármű fedélzeti elektronikája az alapvető funkciók megvalósítására, automatizálására és vezérlésére terjedt ki, mint a

- világítás;
- hajtáslánc;
- indító motor / Generátor;
- tüzelőanyag-befecskendező rendszer;
- és gyújtásvezérlés.

Irányításuk analóg és logikai áramkörökkel volt megvalósítva, mivel a digitális irányításhoz szükséges, bonyolultabb technológia nem állt rendelkezésre.

2.1.2. 80-as évek

Az 80-as években a járművek fedélzetén is megjelenő mikrokontrollereknek (Micro Contoller Unit – MCU) köszönhetően a számítási teljesítmény ugrásszerűen megnőtt, ezzel a lehetséges felhasználások köre is kiterjedt. Megvalósíthatóvá vált a meglévő szabályozási feladatok digitális vezérlésűvé alakítása, ezen keresztül azok bonyolultságának, megbízhatóságának és hibatűrésének növelése, a gépjárművek fogyasztásának csökkentése [3]. Sorozatgyártásba került az elektronikus vezérlésű automataváltó és az adaptív felfüggesztés, melyek kényelmesebbé és az úttartás javításával, valamint a vezetőre jutó feladatok csökkentésével könnyebben irányíthatóvá tették a járműveket. A biztonsági rendszerek területén megjelentek az ütközésérzékelő rendszerek és a légzsák (1981, Mercedes S-osztály) [4], továbbá a blokkolásgátló és az övfeszítő. Az utasok épségének megóvását segítő figyelmeztető rendszerek (mint a biztonsági öv becsatolt állapotát, illetve a keréknyomást figyelő) kezdtek elterjedni.

Az utasok kényelmét szolgáló elektronikus eszközök első példányai is ekkor jelentek meg, többek között a fedélzeti számítógép, elektronikusan vezérelt klímaberendezések, első ülések és fűtött külső tükrök. Parkolás közben a jármű biztonságát megvédeni hivatott elektronikus lopásgátlók váltak elérhetővé.

A fedélzeten található ECU-k közti kommunikáció szabványosítására a Robert Bosch GmbH 1986-ban bemutatta a CAN (Controller Area Network) protokollt, melynek továbbfejlesztett változatát, a CAN 2.0-t a ma gyártott járművekben is használják.

2.1.3. Modern idők

A modern gépjárművek elektronikájának bonyolultsága messze meghaladja akár a 2000-es évek elején látott szinteket is: egy átlagos modern gépjármű ECU-inak száma 60-hoz közelít. Mindezen eszközökre a folyamatosan szigorodó környezetvédelmi és biztonsági előírások teljesítéséhez, illetve a felhasználók által támasztott egyre magasabb követelmények kielégítéséhez van szükség. Ezek a mikrokontrollerek több, különböző technológiájú kommunikációs hálózatot használnak, mint például

- CAN,
- LIN,
- MOST
- és FlexRay.

Egy korszerű személygépjármű biztonsági rendszereivel a benne utazók és a környezete épségét is védi, és a balesetek hatásainak mérséklésén (passzív biztonság) túl magát a balesetet is segít megelőzni, ha szükséges a vezető beavatkozása nélkül is (aktív biztonság). Folyamatosan figyeli környezetét, és figyelmezteti a sofőrt amennyiben valami elkerülte annak figyelmét (holttér-figyelés, táblafelismerés, sávelhagyás-figyelmeztetés). A modern járműelektronika képes öndiagnosztikára, kritikus alkalmazások esetén többszörösen redundáns felépítésű. Ezen túl kiterjedt multimédiás szolgáltatásokkal rendelkezik, képes a benne utazók elektronikus eszközeivel vezeték nélkül kommunikálni, illetve lehetővé teszi a jármű viselkedésének, egyes kényelmi beállításainak testre szabását.

Ezen széles szolgáltatási kör létrehozását (melynek bővebb kifejtése a 2.2. fejezetben olvasható) az elektronikai ipar fejlődése tette lehetővé. Ez a fejlődés – és a vele járó adott számítási teljesítmény árának jelentős csökkenése – a hajtóereje a járműfedélzeti elektronika egyre erőteljesebb térhódításának. Példaként hadd hozzam fel a General Motors által használt erőátviteli vezérlő (PCM) mikrokontollerét, mely az 1. táblázatban szerepel.

GM PCM MCU	*1979*	*2005*
Típus	Custom Motorola 6800 (8 bit)	Motorola Power PC (32 bit)
Órajel	1 MHz	56 MHz
Memória	4 kbyte	2000kbyte
Programnyelv	Assembly	C+kevés Assembly
Kalibrációs adatok	256 byte	256 kbyte
Szenzorok és beavatkozók	5 darab	20 darab
Csatlakozók lábszáma	<50	>200

1. táblázat Motorvezérlő MCU-k összehasonlítása

2.2. Szolgáltatási szintek

Egy jármű fedélzeti elektronikáját több, egymással kapcsolatban lévő hálózatban működő ECU (Electronic Control Unit) összessége alkotja, ezen ECU-k tevékenységét három fő csoportba sorolhatjuk, melyek a következők:

2.2.1. Működtető rendszerek

Ehhez a területhez tartoznak a jármű alapfunkcióit kezelő, irányító mikrokontrollerek és programok. Ezeket a feladatokat automatizálták először. A terület természetesen komoly fejlődésen ment keresztül az első digitális motorvezérlések és automata váltók megjelenése óta, azonban az alapvető funkciók nem változtak. Ide tartozik az erőátvitel, a kormányzás és a futómű vezérlése. A működtető rendszereknek, mivel meghibásodásuk komoly balesetveszélyt hordoz magában, szigorú, a DIN 31000 szabványban leírt előírásoknak kell megfelelni. [5]

Például egy elektronikus gázpedálnak, ami az AK4/SIL2 kategóriába (meghibásodása rendkívül kis valószínűségű, esemény ideje rövidtől közepesig, következménye több személy sérülése vagy halála) tartozik, az IEC 61508-ban [6] meghatározott feltételeknek (struktúra, hardver, szoftver, aktuátorok és szenzorok) kell megfelelnie. Mint az a példából is látszik, ezen rendszerek meghibásodása rendkívül komoly következményekkel járhat, így fejlesztésükkor a redundáns felépítés, hibaészlelés és hibakezelés alapvető.

A kritikus rendszerekben alkalmazott hibafelismerő rendszerek a következők lehetnek:

- Referenciaellenőrzés: a rendszernek egy kérdés előre már ismert választ kell helyesen megadnia azon eszközeinek segítségével, melyeket normál üzem közben használ. Amennyiben a válasz nem helyes, a rendszer adatait annak módosításáig hibásként kell kezelni.
- Redundáns ellenőrzés: két alapvetően különböző algoritmus azonos bemenő jelet dolgoz fel. Ezek futhatnak külön hardveren, azonos hardveren, illetve lehetséges ugyanazon algoritmus többszörös futtatása a korábbi hibák kiszűrésének érdekében.
- Kommunikáció ellenőrzése: Tipikusan paritás- és redundancia-kontroll, illetve megerősítő válaszjel küldése.
- Fizikai tulajdonságok figyelembe vétele: Ekkor a processzor a szenzor adatai közül csak azokat dolgozza fel, amelyek korrelálnak a szenzor, illetve a szenzor által mért tulajdonság lehetséges értékeivel. Tipikus felhasználása a határértékekkel (melyek vonatkozhatnak az adat szélső értékeire vagy értékének időbeli megváltozására) történő plausability-szűrés, például egy hőmérséklet-szenzor esetén.

A felismert hibákat a következő módokon, illetve ezek kombinációjával lehetséges kezelni:

- Redundáns értékek használata: a vezérlés több, egymástól független bemenő jelet dolgoz fel, melyeket egymás ellenőrzésre is felhasznál. További lehetőség a bejövő adatok egy előre meghatározott, biztonságos, állandó értékkel történő helyettesítése.
- A rendszer vagy egyes részeinek leállítása
- Hibaállapotban maradás vagy működés hibához alakítása
- Hibatárolás az eszköz vagy a központi egység EEPROM-jában
- Hiba kiküszöbölése, például az MCU újraindításával

2.2.2. Biztonsági rendszerek

A biztonsági rendszerekhez tartozik minden olyan funkció, mely a jármű irányíthatóságának fenntartását, a baleset megelőzését, illetve elkerülhetetlen baleset esetén a személyi sérülések, anyagi és természeti károk mérséklését szolgálja. Ezek a

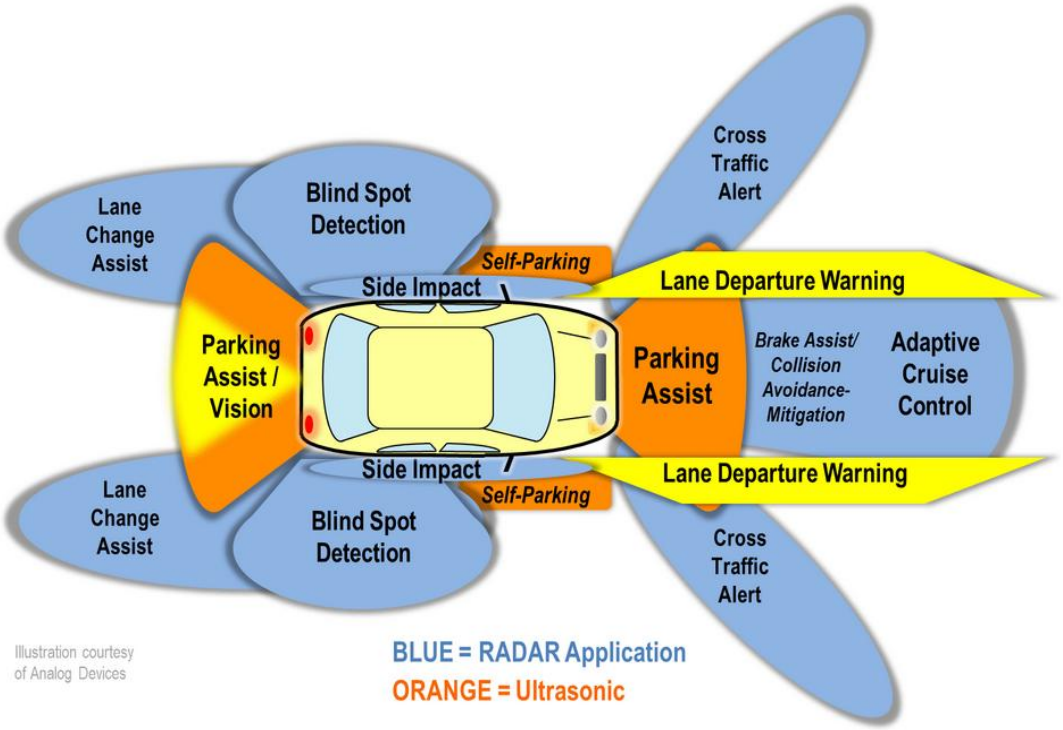
rendszerek a működtető rendszerek automatizálása után kezdtek megjelenni az autókban, előbb a passzív, majd fokozatosan az aktív rendszerek. Ez utóbbiak alkalmazását azonban hátráltatja az esetlegesen általuk okozott vagy ki nem került ütközés esetén a jogi felelősség kérdése, illetve különböző jogi szabályozások.

A biztonsági rendszereket két alapvető csoportra lehet osztani:

- Aktív (ütközést megelőző, a jármű irányítását segítő) – pl. ABS, ESP, Emergency Brake Assistant [7] [8]
- Passzív(ütközés esetén a résztvevők sérüléseit mérséklő) – pl. légzsákvezérlés, övfeszítők

Mindkét csoportba tartozó rendszerek a többitől fizikailag is elkülönített adatbuszon kommunikálnak, és a megengedhető kereteken belül nagy teljesítményűek, bizonyos esetekben (ABS vezérlés) speciális gépészeti megoldásokkal az aktuátorhoz lehető legközelebb helyezik el őket, így is csökkentve a veszély észlelése és a beavatkozás közötti időt.

Egy modern gépjármű szenzorai nem csupán a jármű viselkedését figyelik, de annak a környezetben elfoglalt helyét is folyamatosan monitorozzák. A jármű külső érzékelőinek rendszere a 2. ábraán látható:



2. ábra Korszerű jármű környezetérzékelése

2.2.3. Kényelmi berendezések

A jelenleg forgalomba kerülő járművekben bizonyos kényelmi szint elvárt, ezen rendszerek száma és minősége természetesen a jármű kategóriájának megfelelő. Ilyen eszközök a légkondicionáló, elektromos ablakemelők, különböző hang-, esetenként videórendszerek, vagy a beépített navigáció lehetősége. Ezek esetleges kiesésének önmagában alacsony a biztonsági kockázata (ezért kevésbé szigorú szabályozás vonatkozik fejlesztésükre, üzembiztonságukra és kommunikációjukra), azonban hibás működésük a vezetőt megzavarhatja, ami növeli egy baleset kockázatát.

Ezek az eszközök képezik a szolgáltatási szintek legmagasabb lépcsőjét, ezekkel baleset bekövetkezése nélkül is minden utas kapcsolatba kerül, míg a jármű működtető rendszereinek esetleges beavatkozása jóformán észrevehetetlen. A biztonsági rendszerek néhány jelét ugyan minden utas észlelheti (akár egy intenzívebb fékezés során is), azonban az azzal történő interakció ideje meg sem közelíti a szórakoztató és kényelmi berendezésekkel töltöttet.

Ebből következik, hogy egy gépjármű vásárlásakor a kényelmi berendezések mennyisége, színvonala ha nem is elsődleges, de fontos helyen szerepel a követelmények között, így fejlesztésük nem szorítható háttérbe.

2.3. Jövőbeli trendek, lehetőségek

A járműfedélzeti rendszerek az elkövetkező években, évtizedekben várhatóan tovább szofisztikázódnak majd, egyre újabb és újabb, még precízebb és megbízhatóbb vezérlések fognak megjelenni. Az elérhető számítási teljesítmény növekedésével mindhárom szolgáltatási szint tovább fog fejlődni: jobban időzített és több paramétert monitorozó motorvezérlések, pontosabban működő ESP-k és a jelenleg elérhetőnél is kellemesebb utazási környezet nagy biztonsággal kijelenthető, hogy megvalósul.

A jelenleg elérhető rendszerek továbbfejlesztésén túl szeretnék bemutatni néhány olyan megoldást, amelyek a jövőben új szolgáltatásokat hozhatnak a járműelektronikába.

2.3.1. X-by-wire

A drive-by-wire rendszerrel megszűnik a vezérlőszervek (kormány, pedálok, kapcsolók) és a jármű rendszerei közötti mechanikai kapcsolat, ennek helyébe elektronikus vezérlés lép. A vezető itt már nem az eszközöknek ad közvetlen utasítást, hanem egy vezérlő számítógépnek, amely – ismerve a jármű technikai korlátait, és a külső viszonyokat – a vezető által kívánt végeredményt állítja elő, amennyiben ez lehetséges. Ez a technológia egyáltalán nem ismeretlen a gépjárműiparnál is szigorúbb biztonsági előírásokkal rendelkező űrkutatásban és repülésben. Ilyen rendszer üzemel többek között az Airbus, az új generációs Boeing repülőkben, és az amerikai űrsiklóban is (felújításukat követően). Mint sok más technológiai fejlesztés, ez is a hadiiparban jelent meg először sorozatgyártásban, az F-14-es - azóta már hadrendből kivont- amerikai vadászgép fedélzetén, 1974-ben. [10] Ez a technológia lehetővé teszi az irányítószervek méretének és súlyának csökkentését (hidraulikus vezetékek, fémszálak szerepét egy adatbusz veszi át), mely környezetvédelmi szempontból rendkívül előnyös, főleg olyan, korlátozott hatótávolságú meghajtási módok esetén, mint az elektromos hajtás.

Továbbá többszörös redundancia építhető ki ennek használatával, ami növeli a menetbiztonságot és a megbízhatóságot. [11] Szintén megkönnyíti az aktív biztonsági rendszerek beépítését, mivel kezdettől fogva minden utasítás ellenőrzésen esik át, így a biztonságot veszélyeztető utasítások nem is hajtódnak végre. Ezen túl a telemetriai adatok folyamatos rögzítése/továbbítása is egyszerűbbé válik, ami egyrészt a flottakezelésnél jelent előrelépést, másrész segíti a balesetanalízist.

Azt azonban világosan kell látni, hogy egy ilyen rendszer rendkívül magas biztonsági elvárásoknak kell, hogy megfeleljen. Erre kézenfekvő megoldásnak tűnik a DO-178B minőségbiztosítási rendszer – mely egy repülő- és űrtechnológiában használt szoftverbiztonsági szabvány – az autóipar jellemzőinek figyelembe vételével történő átalakítása és használata. [12] Ezen túlmenően a törvényi előírásoknak megfelelően továbbra is biztosítani kell valamilyen mechanikai kapcsolatot a kezelőszervek és a jármű vezérelt elemei között, elkerülendő hogy az elektromos rendszer üzemképtelensége a jármű irányíthatatlanságát vonja maga után.

2.3.2. Vezetést segítő rendszerek, self-drive

Míg az x-by-wire rendszerek elterjedése jogi akadályokba ütközik, a vezetőt segítő rendszerek területén folyamatos a fejlesztés. Ezek közé tartoznak a felső-középkategóriában ma is jelen lévő sávelhagyás-figyelmeztetés (Line Departure Warning – LDW), kanyarkövető fényszórók (Adaptive Forward Lightning – AFL), adaptív tempomatok (Adaptive Cruise Control – ACC). A jövőben ezek tovább finomodnak, és egyre több emberi hibát lesznek képesek kiszűrni (holttér-figyelés, ütközést megakadályozó szoftverek), mely információ felhasználásával, folyamatos adatgyűjtéssel egyre jobban lehet modellezni az emberi vezetők viselkedését, hibáit. Ez a lépés elengedhetetlen ahhoz, hogy a rendszerek összekapcsolásával autonóm járművet alkossunk.

Több ilyen program is létezik, a „Google self-driving car" projekt immáron több mint 300000 mérföldet tett meg a szoftver hibája által okozott baleset nélkül. Ehhez több szenzor (többek között a jármű tetejére szerelt forgólézeres távolságmérő, GPS és GPRS-alapú helymeghatározó rendszer) együttműködése szükséges, azonban a rendszer létjogosultsága megkérdőjelezhetetlen. Lévén, hogy a közúti balesetek többsége nem műszaki, hanem emberi (vezetői) hibából következik be (gyorshajtás, elsőbbségi viszonyok figyelmen kívül hagyása), és a közlekedési szabályok megszegésére a rendszer természeténél fogva képtelen, a baleseti károk mérséklésének eszközeként a számítógép által vezetett autót komolyan számításba kell venni.

Megjegyzendő azonban, hogy az autót irányító rendszer erőforrásigénye messze meghaladja a jelenleg jármű fedélzetén elérhető számítási kapacitást, így a félvezetőipar további fejlődése nélkül a rendszer annak ellenére sem lenne megvalósítható, ha a gyerekbetegségeit, mint:

- az ideiglenes forgalmi rend kezelésével kapcsolatban felmerülő problémákat,
- a folyamatos emberi felügyelet igényét,
- a többi sofőrrel vagy autóval való kommunikáció kérdését

orvosolná a műszaki tudomány. Ezek a jövőben várhatóan megoldható problémák [13], így a „self-driving car" néhány évtizeden belül megvalósíthatónak látszik. [14]

2.3.3. Kényelmi berendezések

A kényelmi berendezések területén az utóbbi időben megszokott kategóriákon belüli fokozatos elterjedésen túl (vagyis a kizárólag felső kategóriás járművekben megtalálható rendszerek idővel alsóbb szegmensekben is az alapfelszereltséghez tartoznak) néhány új technológia megjelenése is várható. Ezek előrejelzése természetesen – jellegükből fakadóan – sosem lehet teljes, azonban a lehetőségek skálájának megjelenítése szükséges.

Nagy biztonsággal állítható, hogy a jövőben az olyan szervizelést, karbantartást megkönnyítő rendszerek, mint a szenzorhibára figyelmeztetés, az égők kiégését jelző rendszerek, illetve a gumiabroncs nyomását szabályozó rendszerek csakúgy elterjednek, mint a motor- és fékolaj mennyiségét, tulajdonságait monitorozó, ezek mennyiségét folyamatosan optimális szinten tartó szolgáltatások (mely adatokat a vezető/üzemben tartó online is el tudja majd érni, a szervizt időben értesíteni). Ezek elsősorban a vezetőt segítik majd a karbantartásban, azonban nem csak ezen a területen várható fejlődés.

Az utasok, utazás kényelmét szolgáló rendszerek közül a folyamatos internetkapcsolatot lehetővé tévő fedélzeti Wi-Fi ugyanúgy valószínűsíthetően elterjed majd, mint a hordozható eszközök töltését lehetővé tévő csatlakozók, vagy az USB-képes személyi audió-, és videórendszerek. A fenti funkcionalitások a jelenleg elérhető technológiákkal is megvalósíthatók, az még eldöntésre vár hogy a vezérlés alapvetően a jármű, vagy a vezető/utas mobiltelefonján keresztül történik-e. Ezen túlmenően a felső kategóriában várható az összes ülés elektromos állíthatóságának megjelenése, természetesen programozható formában, illetve valamilyen szintű hangvezérlés implementálása.

Egy-két évtizeden belül olyan technológiák is megjelenhetnek, amik a kamerák segítségével a külső világ képét az autó belső felületeire vetítik – ezáltal megszüntetve a holttér fogalmát, illetve lehetővé téve az utasoknak a panoráma teljes körű élvezetét. A technológia – primitívebb formában – jelenleg is létezik, a legmodernebb harci repülők pilótái olyan sisakot viselnek, mely „eltünteti” a repülőt, így bármely irányban megfigyelhető a környezet. Ezen technológiának legújabb, még fejlesztés alatt álló képviselője a VSI által fejlesztett Helmet Mounted Display System-e (HMDS), mely az F-35-ös vadászbombázóhoz készül. [15]

3. Járműmechatronika oktatási igényei

A Közlekedés- és Járműmérnöki Karon folyó Járműmérnök képzés Járműmechatronika szakirányához kapcsolódóan a diákok megismerkednek a járműveken alkalmazott hidraulikus, pneumatikus és elektromechanikus egységek jellemzőivel, ezek gépészeti, üzemeltetési, tervezési és vezérlési sajátosságaival. Megtanulják az eszközök irányításának, vezérlésének elméleti alapjait, valamint foglalkoznak ezek implementálásának gyakorlati kérdéseivel mind érzékelők, mind beavatkozó és vezérlő egységek esetén.

Az oktatás során a járművek különböző elektronikai vezérlő rendszereire való szoftverek fejlesztése a Járműfedélzeti rendszerek 2 tantárgy keretei között zajlik. E programok a 0 fejezetben tárgyalt szolgáltatási szintek közül a működtető és a kényelmi rendszerek közé sorolható funkciókat valósítanak meg egy fejlesztői panelen elhelyezett Atmel AT90CAN128 mikrokontrollerrel. A feladatok közé tartoznak többek között a következő rendszerek:

- Elektromos ablakemelő (EWL)
- Fedélzeti adatgyűjtő (DAU)
- Fedélzeti számítógép (TCU)
- Holttér- és sávelhagyás-figyelő (BSA)
- Irányjelző (TS)
- Kanyarkövető fényszóró vezérlése (HLC)
- Központi zár (CL)
- Tempomat (ECC)

Ezen alkalmazásoknak a saját funkcionalitásukon túlmenően képesnek kell lenniük egymással is kommunikálni: meghatározott időzítéssel státuszüzeneteiket a CAN hálózatra (melynek specifikációit a II. Függelékben leírt BME_LAB_Network.dbc fájl tartalmazza) kiküldeni, illetve onnan parancsokat fogadni.

A hallgatók által készített programok fejlesztés közbeni folyamatos teszteléséhez, valamint a félév végi validációjához és munka értékeléséhez szükséges tesztprogramok számítógépen futnak, melyek National Instruments, Vector és Inventure teszthardvereket használnak.

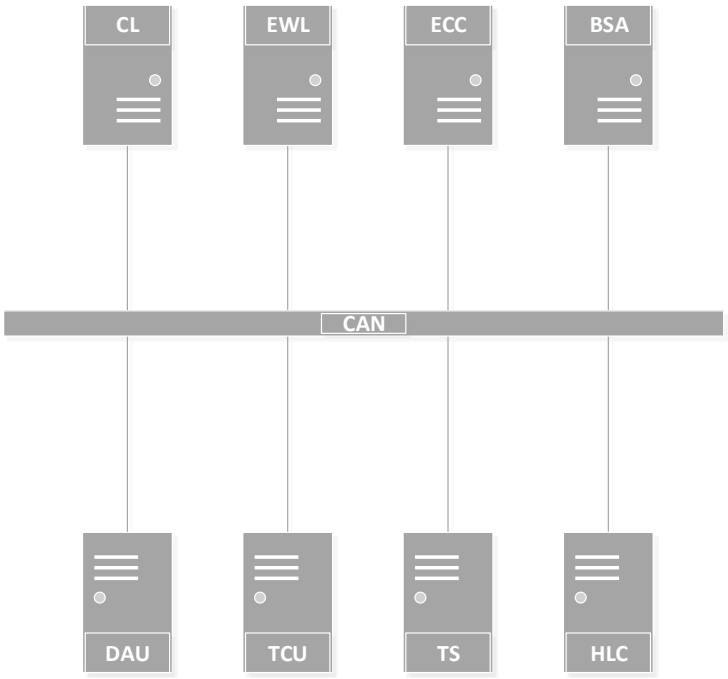
Ezen túlmenően más egyetemi tárgyakhoz, illetve kutatáshoz-fejlesztéshez is szükséges egyedi tesztprogramok készítése, amelyeknél szintén az előbbiekben említett, és a 4. fejezetben részletesen tárgyalt tesztarchitektúrát használjuk.

4. Labor CAN hálózata

A hallgatók által fejlesztett vezérlő egységek 500 kbit/s sebességű CAN hálózaton kommunikálnak, illetve küldenek státuszüzeneteket. Ez a hálózat a CAN 2.0 szabványnak megfelelő, kiterjesztett címzéssel rendelkezik, és a következő rendszeres CAN státuszüzenetek találhatók meg rajta üzemszerű működés közben (zárójelben az üzenet küldésének gyakorisága):

- Status_BSA (50 ms)
- Status_CL (50 ms)
- Status_DAU (500 ms)
- Status_ECC (50 ms)
- Status_EWL (50 ms)
- Status_HLC (50 ms)
- Status_TS (50 ms)

A hálózat elvi felépítése, a rajta kommunikáló hardverelemekkel a 3. ábraán látható.

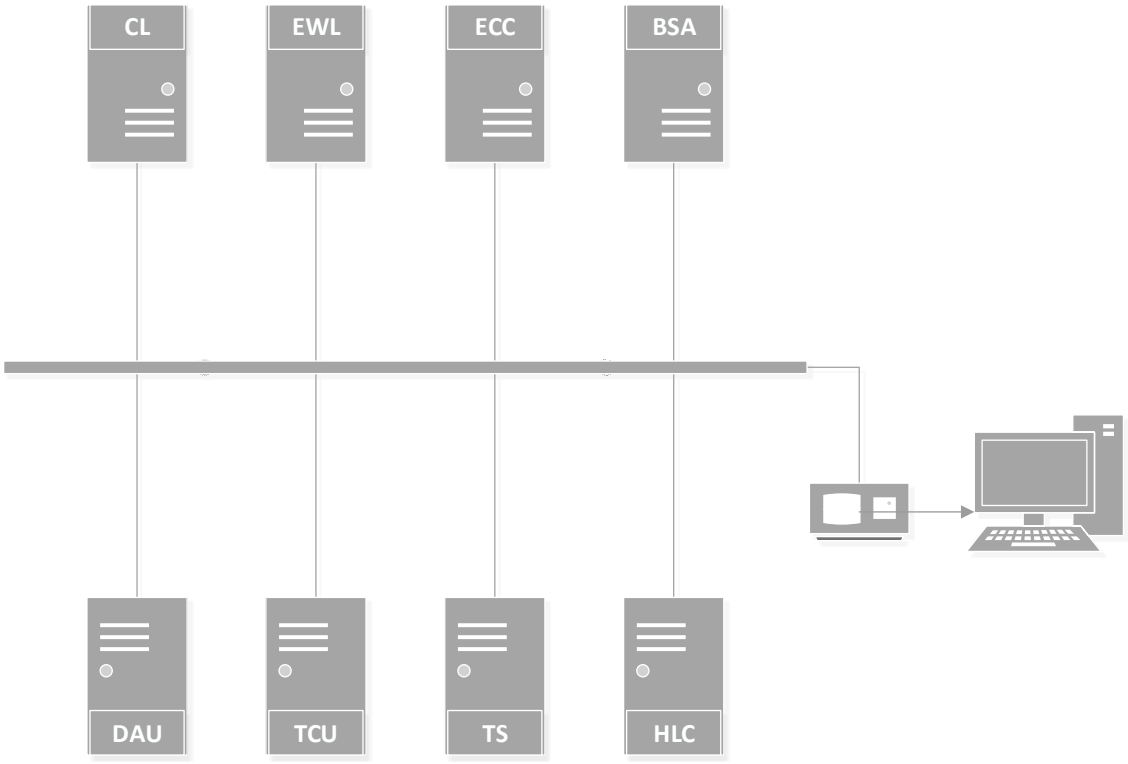


3. ábra BME_Lab_network CAN hálózat felépítése

Annak érdekében, hogy a hálózati forgalmat kellőképpen felügyelni, és így a hibákat észlelni lehessen, szükséges a kommunikáció valós idejű monitorozása, ennek elvi felépítése a 4. ábraán látható. Ezt különböző eszközökkel vagyunk képesek megtenni:

- National Instruments CompactDAQ (CAN kártyával)
- Inventure WeCAN [16]
- Vector CANalyser

Ezek közül gyors felhasználáshoz a Vector eszközeit, párhuzamos fejlesztésekhez pedig az Inventure hardveres megoldásait szoktuk használni. A National Instruments CAN hardvereszközeit alacsony számuk és a hozzá írt CAN felügyelő alkalmazás hiánya miatt, amíg megfelelő alternatívát jelentenek a fentiek (vagyis nincs szükség magasabb szintű tesztprogramba integrálásra az 6. fejezetben tárgyaltaknál) várhatóan nem is fogjuk a hallgatói laborban használni.



4. ábra BME_Lab_Network számítógépes CAN monitorozással

A fejlesztett vezérlőegységek a 2. táblázatnak megfelelően olvassák egymást státuszüzeneteit bejövő adatokért, illetve adnak parancsokat egymásnak:

	BSA	CL	DAU	ECC	EWL	HLC	TCU	TS
BSA				Read ST		Read ST		Read ST
CL								
DAU							Get CMD	
ECC								
EWL		Get CMD						
HLC				Read ST				
TCU		Read ST	Read ST	Read ST				
TS		Get CMD						

2. táblázat BME_Lab_Network kommunikáció

A fenti táblázatban szereplő kifejezések:

- Read ST: Státuszüzenet olvasása bejövő információért
- Get CMD: Parancsüzenet olvasása utasításért

Ahhoz, hogy a táblázat első oszlopában szereplő vezérlőegység fejlesztéséhez szükséges emuláció teljes körű lehessen, szimulálni kell a CAN hálózati kommunikációt leíró táblázat vezérlőhöz tartozó sorában szereplő adatokat és parancsokat; továbbá olvasni kell a mikrokontroller által küldött státuszüzeneteket.

5. Emulátor szoftverek általános specifikációja

Ahhoz, hogy a vezérlő mikrokontrollereket megfelelően lehessen tesztelni, szükséges azok minden ki- és bemeneti jelének kontrollálása, de legalábbis monitorozása. A mérőrendszernek a következő portokkal kell rendelkeznie:

- Analóg ki- és bemenet (Analog I/O)
- Digitális ki- és bemenet (Digital I/O)
- CAN csatlakozó

Továbbá szükséges a fentieket megfelelően kezelni képes, megfelelő teljesítményű számítógépes hardveres és szoftveres architektúra.

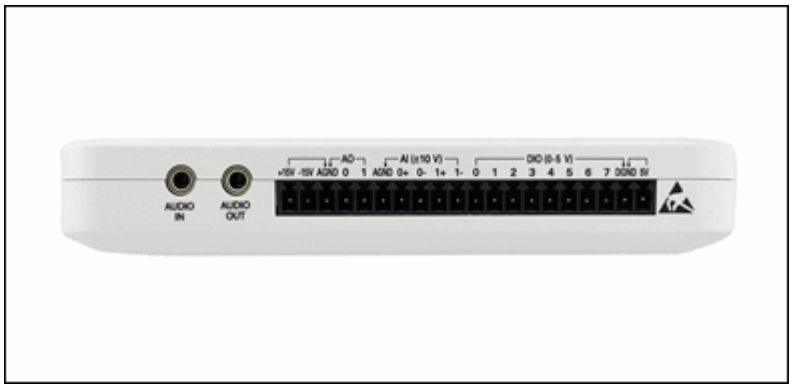
Célszerű továbbá a tesztszoftverek lehetőségek szerinti uniformizálása, mely mind a tesztszoftver fejlesztését, esetleges hibáinak gyorsabb azonosítását és javítását; mind a felhasználást megkönnyíti.

Az általam fejlesztett programok futása közben (amennyiben ezt a funkciót a felhasználó nem kapcsolja ki) minden ki- és bemenet az adott felhasználásnak

megfelelő gyakorisággal mentésre kerül, így lehetővé téve a teszteredmények későbbi megjelenítését, elemzését és összehasonlítását. Ezen túlmenően a szoftverek fejlett hibakezeléssel és ahol szükséges, hardveres konfliktuskezeléssel rendelkeznek, valamint képesek automatikus tesztszekvenciák végrehajtására.

5.1. Hardveres felépítés

A hallgatók a fejlesztést Atmel AT90CAN128 típusú mikrokontrolleren végzik, egy Mikroelektronika [15] által fejlesztett BIGAVR6-os fejlesztői boardon, melynek képe a I. Függelékben található. A vezérlőegységek analóg és digitális be- és kimeneteit National Insruments MyDAQ [16] eszközökkel állítom elő. Ezek száma applikációtól függően változik, a jelenleg tesztelésre használt rendszerek esetében (a használt NI MyDAQ mérőeszköz sajátosságai miatt) egyszerre legfeljebb 2 analóg kimenetet, 3 analóg bemenetet, és összesen 8 digitális ki- és bemenetet használható. Az eszköz oldalnézete az 5. ábraán látható.

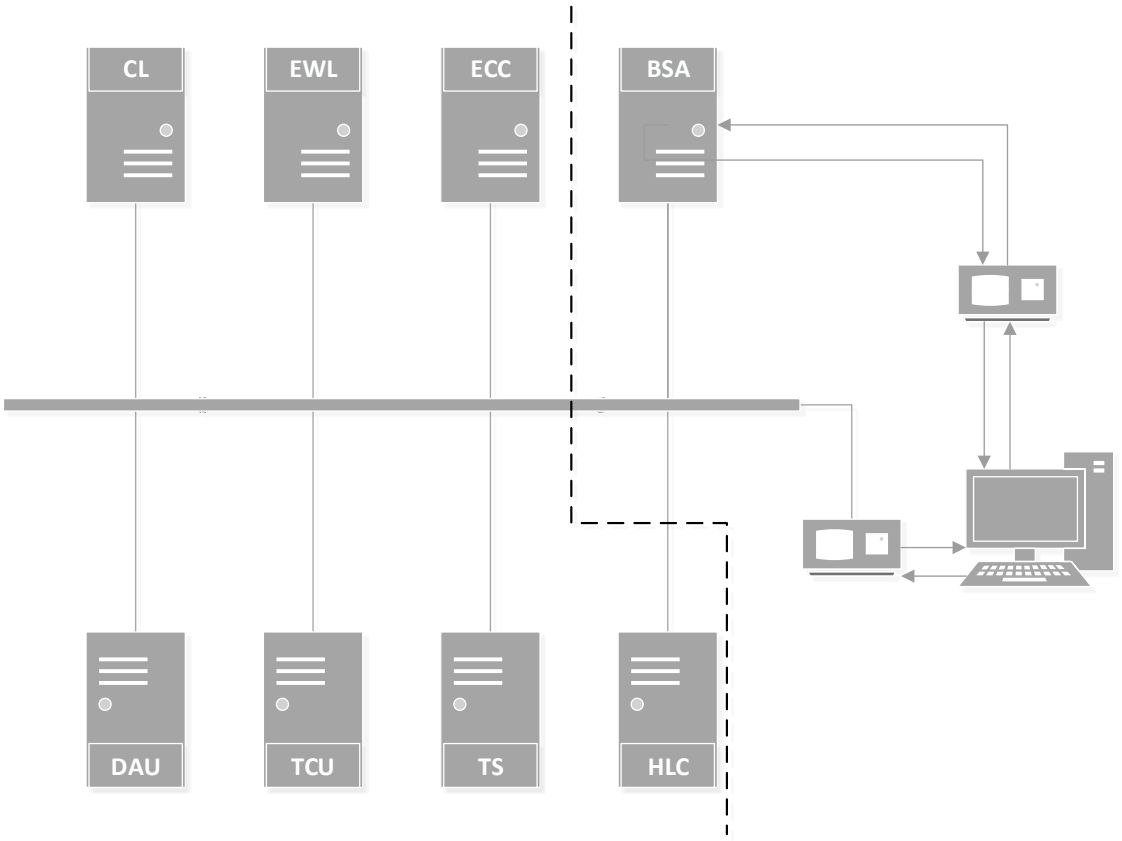


5. ábra National Instruments MyDAQ ki- és bemenetei

A fejlesztéshez, illetve különféle tesztek elvégzéséhez bizonyos alkalmazások esetén szükséges, hogy az egyes vezérlőket a CAN hálózatról leválasztva, az azon szereplő releváns adatokat emulálva legyenek képesek a hallgatók a viselkedését tanulmányozni. Ennek elméleti összeállítását a **6**. ábra szemlélteti.

Ennek több előnye is van:

- Az egyes komponensek fejlesztésének üteme nem függ egymástól
- A vezérlők által kiadott hibás jelek nem okoznak további hibát a rendszerben, így
- A hibajelenségek szeparálásával azok okainak felderítése is felgyorsul, javításuk egyszerűsödik



6**. ábra Tesztelés BME_Lab_Network CAN hálózat és a BSA jeleinek emulálásával**

5.2. Kezelőszervek szoftveres megvalósítása

Ahhoz, hogy a különböző mérési feladatokat megfelelően időzítve és dokumentálva lehessen végrehajtani, szükséges a rendszer összes bemenő és kimenő jelének monitorozása, valamint célszerű a bemenetek számítógépen való kontrollálása. Ez utóbbinak több előnye is van:

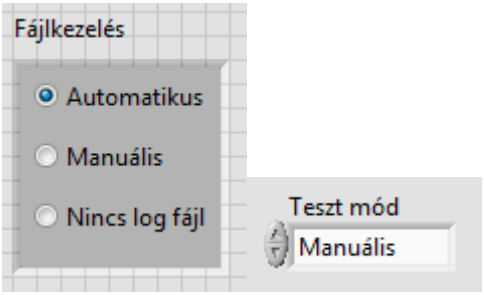
- Lehetővé teszi a felhasználási környezet jobb szimulálását.
- Egy szoftverben egyesíteni tudja az összes szükséges tesztfeladat vezérlését, ezzel Az esetleges szoftveres és hardveres konfliktusokat is megelőzi.
- A bemenő jelek időzítése pontosabbá tehető.
- Lehetségessé válik a rendszer gyorsított tesztelése.
- A fejlesztői panelen lévő digitális és analóg kezelőszervek bizonytalan nyomáspontjából, valamint esetlegesen nem megfelelő működéséből adódó bemeneti hibák kikerülhetők.

Természetesen ez a megvalósítás a tesztelésre használt hardverek igénybevételét, valamint a teszthardver és a panel nem megfelelő összekötésének kockázatát növelik.

További kérdésként megjelenik, hogy a tesztelésre használt hardver képes-e megfelelően gyors működésre, valamint elegendő ki- és bementettel rendelkezik, vagy szükség van-e további hardverelemek rendszerbe építésére.

A programok továbbá különböző hibajelenségek szimulálására is képesek, melyek ki-be kapcsolása szintén a számítógépes felületen történik. Ezen hibák közvetlenül nem jelennek meg a kimeneteken, a rendszer belső tulajdonságainak módosításával azonban közvetve jelentősen befolyásolják a bemenő jelekre adott választ.

5.3. Logolás, automatikus tesztek



7. ábra Fájlkezelés és teszt mód választó gombok

A tesztprogramok futása közben háromféle jelrögzítési lehetőséget választhat a felhasználó, melynek kezelőszerveit az 7. ábra ismerteti:

- Nincs log fájl
- Automatikusan írásvédett log fájlt készít a program, melynek nevében szerepel a mérés típusa, dátuma és ideje
- A felhasználó nevezi és helyezi el az írásvédett log fájlt

Ezen fájlok mind szövegszerkesztőben egyszerűen olvashatók, mind táblázatkezelő programban gyorsan megjeleníthetők és grafikusan feldolgozhatók. A hibaanalízis megkönnyítése érdekében minden bemeneti és kimeneti jel értékét menti a program a fájlba.

Erre egy példa az ablakemelő mikrokontrollerhez tartozó emulátor által készített log fájl részlete. Ezen fájlok a mérés jellegét, idejét a log fájl fejlécében az alábbi módon tartalmazzák:

Window lift test results

Date and time of measurement: 2013.09.27. 8:24

Test software written by BME-KJIT

Egy mérési sor részlete látható a 3. táblázatban:

Time [s]	Delay [ms]	Window pos [mV]	PWM [V]	Err_PWM	Block	Poz_OK	AV_OK	FV_OK	Stat_AV	Stat_FV	CMD_Up	CMD_Dwn
0	100	3000	NaN	0	0	1	1	1	0	1	0	0
0	100	3000	NaN	0	0	1	1	1	0	1	0	0
0	100	3000	NaN	0	0	1	1	1	0	1	0	0
0	100	3000	NaN	0	0	1	1	1	0	1	0	1
0	100	2962	0	0	0	1	1	1	0	0	0	1
0	100	2921	0	0	0	1	1	1	0	0	0	1
1	100	2888	0	0	0	1	1	1	0	0	0	1

3. táblázat Ablakemelő emulátor log fájljának részlete

A táblázat oszlopainak jelentését a 6.2.3. fejezetben részletesen leírom.

A program továbbá képes előre megírt tesztszekvenciák végrehajtására (ezek kiválasztása a program elején történik meg, a felhasználó választhatja a kézi vezérlést is mint lehetőséget), akár futás közben változó időzítéssel is. Ez a funkció képes bizonyos jellemző szituációk szimulálására, a rendszer hibatűrésének elemzésére, valamint egy komplex szekvencia használatával a rendszer validálására is. Ezen túlmenően – amennyiben a tesztelt vezérlés jellege ezt megengedi, a technikai korlátok figyelembe vételéve – képes gyorsított tesztek végrehajtására.

Az automatizált tesztekhez használt adatfájlokat egy külön programban irányítottan és a paraméterek pontos megnevezését ismerve lehet létrehozni, a program ezen túlmenően képes tabulátorral elválasztott szöveges fájlokat is bemenetként értelmezni. Ebben az esetben a fájlhiba lehetősége azonban sokkal nagyobb, mivel a beolvasott értékek csak bizonyos határok között értelmezhetők, valamint számuk és elhelyezésük a soron belül rögzített, ahogy a sor hossza is.

Automatikus tesztek esetében kötelező a log fájl készítése, mely továbbra is lehet automatikusan létrehozott, illetve manuálisan elnevezett és elhelyezett.

5.4. Szabványos kinézet, hibakezelés

Mivel minden tesztprogram azonos szoftveres, és hasonló hardveres architektúrára íródott, fejlesztői és felhasználó szempontból is célszerű a működésüket és kezelőszerveiket a meglévő kereteken belül uniformizálni.

Erre olyan, minden, vagy legalább több tesztprogramban használt funkciók ideálisak, mint:

- A használandó hardver kiválasztása,
- A mérési fájlok (logok) kezelése,
- Az automatikus teszt szekvenciák fájljainak kezelése,
- A felhasználói bemenetek engedélyezése/tiltása, illetve
- Több feladatban is megjelenő azonos bemenetek kezelő szervei.

Az egyes elemek képernyőn való elhelyezkedése a tesztprogramok között változik (amint ez a 6. fejezetben látható lesz), viselkedésük azonban minden egyes esetben azonos marad.

Egy tesztrendszernél különösen nagy figyelmet kell fordítani arra, hogy az esetleges hibákat a lehető leghamarabb kijelezzük, továbbterjedésüket megelőzzük. Ennek hiányában lehetséges lenne, hogy egy hibás tesztrendszerrel a mikrokontrollerre írt vezérlő program a teszteket kielégítően működik, azonban a célhardverrel, vagy más ECU-kkal összekötve már nem megfelelő eredményeket ad.

Ezért a tesztprogramok futás közben korlátoznak, illetve újraengedélyeznek felhasználói beavatkozásokat. Ennek legszembetűnőbb példái a hardverelemek, valamint a fájlkezelés választó felületei, melyeket csak a futás előtt lehet beállítani. Automatikus módban a rendszerbe a felhasználó nem tud adatokat bevinni a futás kezdete után, azt viszont bármikor leállíthatja, ekkor a program megszakítja a hardveres működést, újraengedélyezi a felhasználói vezérlést, majd leáll.

Mivel nem csak felhasználói beavatkozás okozhat nem megfelelő működést, hanem bármilyen szoftveren belüli hiba is, szükségessé vált az egymástól függetlenül futó ciklusok közötti hiba-, illetve vészleállítási kommunikáció megvalósítása. Ez a kommunikáció teszi lehetővé, hogy bármely ciklusban bekövetkező hiba bekövetkezése esetén az egész program leáll, és tájékoztatja a felhasználót a hiba jellegéről és forrásáról. Ezen túlmenően ugyanezen kommunikációs csatornán keresztül állítható le az egész program futása, amennyiben arra a felhasználó parancsot ad, vagy az automatikus tesztszekvencia lejár. Ezen esetekben is végrehajtódik azonban a program végén az erőforrás-felszabadítás, valamint a bemenetek engedélyezése.

6. Tesztprogramok megvalósítása

A 3. fejezetben felsorolt rendszerek teszteléséhez használt szoftvert a National Instruments által fejlesztett LabView környezetben készítettem. Ez a magas szintű, vizuális programozási nyelv elsősorban tesztelési feladatok végrehajtására és automatizálására készült, azonban képes különböző vezérlési feladatok megvalósítására is mind statikus (például gyártósori), mind mobil (különböző mobil robotok, robotautók) alkalmazásokban.

A dolgozatban terjedelmi korlátok miatt csak három tesztprogram részletes bemutatására kerül sor, ezek a három típusú emulátort egy-egy példával szemléltetik:

- Kizárólag analóg és digitális ki- és bemenetekkel;
- Csak CAN üzenetekkel;
- Analóg, digitális ki-és bemenetekkel, valamint CAN üzenetekkel való emuláció.

A tesztprogramok mind fejlett hibakezeléssel rendelkeznek, valamint futás közben szeparáltan leállítható a log fájlok készítése. Ezen túlmenően a felhasználó által okozott hibák számának csökkentése érdekében a futás egyes fázisaiban a korábbiakban megadott hardveres paraméterek kezelőszerveit letiltja, majd a teszt végeztével újraengedélyezi. Ez a funkció fokozottabban jelenik meg az automatikus futás közben, ahol a hardvereszközök és a log fájl kiválasztása után (ebben az esetben nem választható a „nincs log fájl” opció) az eszköz összes bemenete letiltásra, majd a tesztszekvencia végeztével engedélyezésre kerül, majd a program befejezi a futását.

6.1. Irányjelző

Az irányjelző vezérlési feladat esetében a hallgatók egy jármű irányjelző készülékeit működtetik, 1 Hz-es gyakorisággal. Vezérlése a tesztprogramon keresztül történik, amelynek kimenete a kormány elfordulási szöge, az irányjelző lámpák működőképessége, illetve a rendszer vezérlő gombjai.

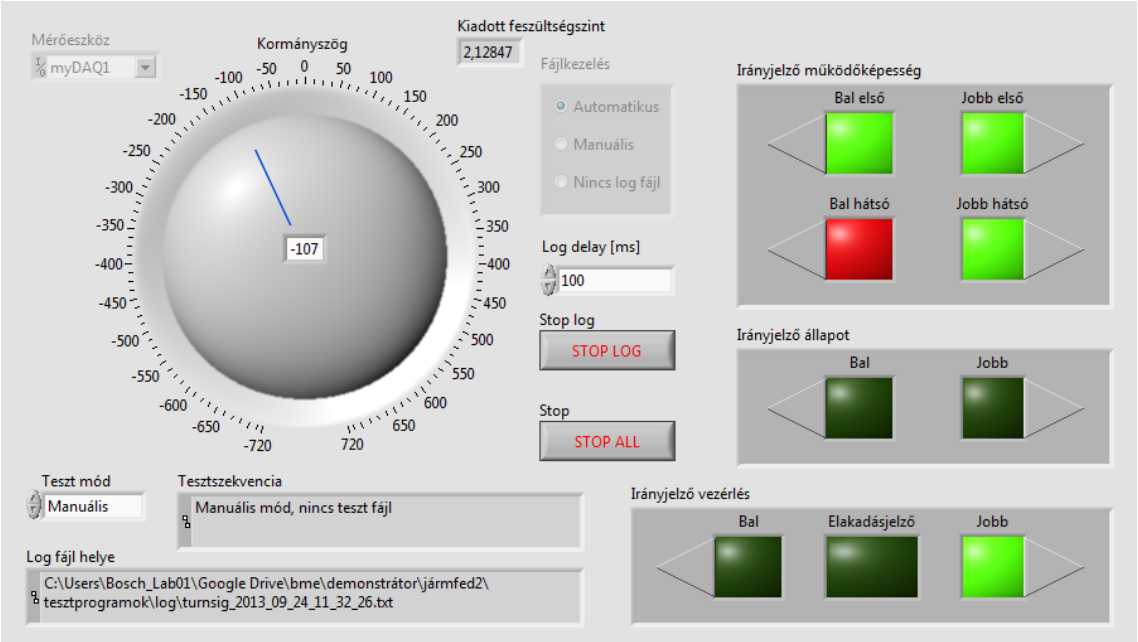
A rendszernek képesnek kell lennie normál körülmények között hibamentesen működni, melybe beletartozik:

- Alapvető irányjelzési feladatok végrehajtása (adott oldal, vészjelző)
- A kormány visszatérítésekor az irányjelzés kikapcsolására
- Státusz üzenetek küldése CAN hálózaton
- CAN hálózaton érkező utasítások végrehajtása

Ezen túlmenően képesnek kell lennie a fellépő hardveres hibák, egymásnak ellentmondó bemenetek biztonságos kezelésére, e problémák jelzésére mind közvetlenül a vezető (a fényjelzési frekvencia csökkentésével), mind a CAN hálózaton keresztül az adatgyűjtő felé.

Ezen feladat megoldásához szükséges szoftveresen emulált hardver különböző analóg és digitális ki- és bemenetekkel rendelkezik, CAN hálózaton azonban nem küld és nem is fogad adatokat, így azok tesztszoftveres kezelése sem szükséges.

Az alkalmazás futási képe a 8. ábraán látható:



8. ábra Irányjelző emulátor futási képe

Az átláthatóság növelése érdekében a működőképességhez, állapothoz és vezérléshez tartozó kezelőszervek és visszajelzők a képernyő jobb oldalán vannak csoportosítva. A képernyő jobb oldalán vizuálisan és numerikusan is megjelenik a kormányelfordulás szöge, valamint ennek közelében visszajelzési céllal az ez alapján kalkulált, és analóg kimeneten kiadott feszültségszint. A jobb oldalon lévő összekötési segédlet a képen nem szerepel.

A 5.4.-es pontban már említett, minden tesztprogramban visszatérő elemek közül a hardver kiválasztásáért felelős a bal felső sarokban, a tesztelésért és fájlútvonalak megjelenítéséért felelősek alul, a logolásért és a leállításért felelős eszközök pedig középen találhatók.

6.1.1. Ki- és bemenetek

Az emulációs feladat végrehajtásához a program a National Instruments MyDAQ 4. táblázatban szereplő ki- és bemeneteit használja:

Csatorna	Jel
DI - 0	Bal világít (1: aktív)
DI - 1	Jobb világít (1: aktív)
DO - 2	Bal irányjelző vezérlés (1: aktív)
DO - 3	Jobb irányjelző vezérlés (1: aktív)
DO - 4	Bal első világítótest visszajelzés (1: működőképes)
DO - 5	Jobb első világítótest visszajelzés (1: működőképes)
DO - 6	Bal hátsó világítótest visszajelzés (1: működőképes)
DO - 7	Jobb hátsó világítótest visszajelzés (1: működőképes)
AO - 0	Kormánykitérés (0-5V)
AO - 1	Elakadásjelző vezérlés (digitális jel) (1: aktív)

4. táblázat Irányjelző ki- és bemenetei

6.1.2. Szimulációs lehetőségek

Az emulátor program képes egy irányjelző működése közben általánosságban fellépő vezérlési helyzetek, valamint hardveres eredetű hibák szimulálására. Az emulációval létrehozható, és a validációs tesztszekvenciában szereplő vezérlések a következők:

- Elakadás-jelzés vezérlés
- Hosszú irányjelzés vezérlés egy irányba,
- Rövid irányjelzés vezérlés egy irányba (3 villantás)
- Irányjelzés egy irányba, majd kormányelfordulással való oldás
- Irányjelzés egy irányba, majd elakadás-jelzés – *ekkor az elakadás-jelzésnek felül kell írnia az irányjelzést*
- Irányjelzés olyan irányba, ahol kiégett egy lámpatest – *az irányjelzést ki kell vezérelni, a hibát a státuszüzenetben CAN hálózaton továbbítani, valamint a hibát tárolni kell*
- Elakadás-jelzés kiégett lámpatesttel – *az elakadás-jelzést ki kell vezérelni, a hibát státuszüzenetben CAN hálózaton továbbítani, valamint a hibát tárolni kell*
- Irányjelzés olyan irányba, ahol minden lámpatest kiégett – *a hibát státuszüzenetben CAN hálózaton továbbítani, valamint a hibát tárolni kell*

- Elakadás-jelzés, miközben minden lámpatest kiégett – *a hibát CAN hálózaton továbbítani, valamint a hibát tárolni kell*

A dőlt betűvel szedett részek a hallgatók által fejlesztett rendszertől elvárt működést írják le adott bemenetek, illetve állapot esetén.

6.1.3. Fájlkezelés

Az irányjelző emulátor program a későbbi hibaanalízis megkönnyítése érdekében log fájlt készít (amennyiben ezt a funkciót nem kapcsolták ki). A készített adatfájlban a 5. táblázatban ismertetett paraméterek kerülnek rögzítésre:

Oszlop neve	Jelentés
Time	A program indulása óta eltelt másodpercek száma
Delay	Két mintavétel között eltelt ezredmásodpercek száma
Angle	Kormány állásszöge (0V = -720°)
V_out	Kiadott feszültségszint [V]
Op_RF	Bal első irányjelző működőképes (1: működőképes)
Op_LF	Jobb első irányjelző működőképes (1: működőképes)
Op_RR	Bal hátsó irányjelző működőképes (1: működőképes)
Op_LR	Jobb hátsó irányjelző működőképes (1: működőképes)
Cont_L	Bal irányjelzés vezérlés (1: aktív)
Cont_R	Jobb irányjelzés vezérlés (1: aktív)
Cont_EM	Elakadásjelző vezérlés (1: aktív)
Stat_L	Bal irányjelzés állapot (1: aktív)
Stat_R	Jobb irányjelzés állapot (1: aktív)

5. táblázat Irányjelző log fájljának tartalma

Az emulátor program képes automatizált teszt szekvencia végrehajtására is, ekkor a 6. táblázatban szereplő bemeneteket kell beolvasnia az erre szolgáló fájlból:

Jel	Érvényes tartomány
Log delay [ms]	[10;1000]
Angle [°]	[-720;720]
Op_RF	0;1
Op_LF	0;1
Op_RB	0;1
Op_LB	0;1
Cont_L	0;1
Cont_R	0;1
Cont_EM	0;1

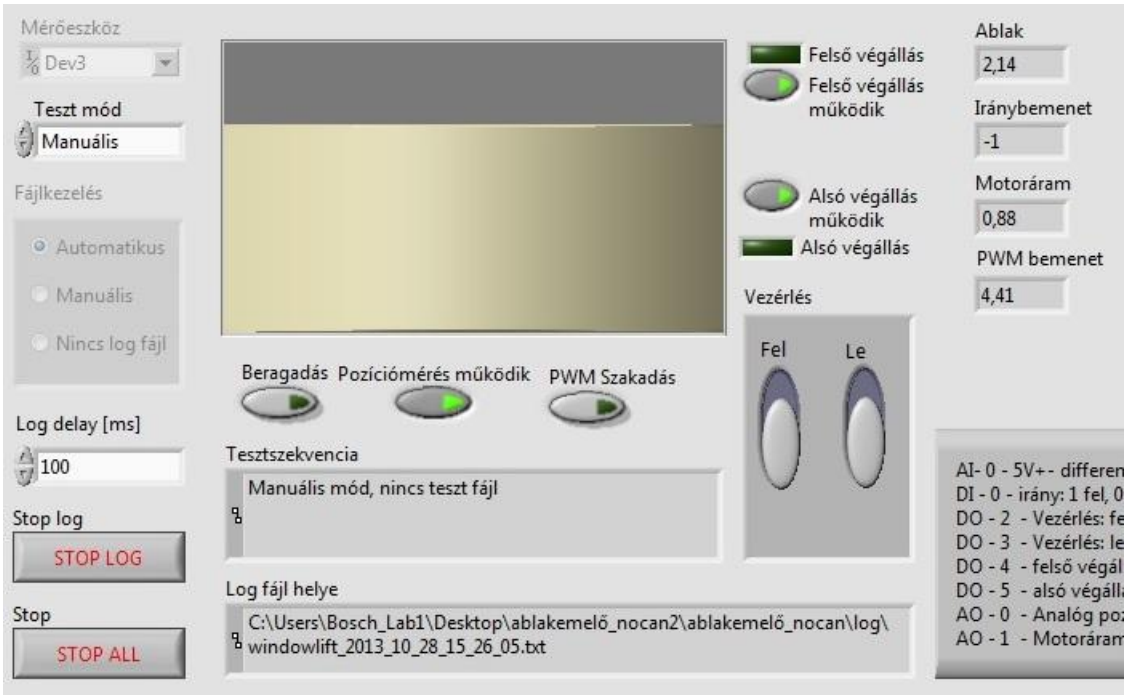
6. táblázat Irányjelző automatikus futtatási fájljának tartalma

6.2. Ablakemelő

Az ablakemelő hallgatói feladatnál a cél egy elektromos motorral mozgatott ablak vezérlő programjának megírása. Vezérlése a tesztprogramon keresztül történik, amelynek kimenete az ablak pozíciója, a két végállás-érzékelő állapota, illetve a rendszer vezérlő gombjai. Az emulátor bemenetei a vezérlőegységtől érkező utasítások.
A rendszernek képesnek kell lennie normál körülmények között hibamentesen működni, melybe beletartozik:

- Az ablak mozgatásának biztonságos megvalósítása
- Státusz üzenetek küldése CAN hálózaton
- CAN hálózaton érkező utasítások végrehajtása

Ezen túlmenően képesnek kell lennie a fellépő hardveres hibák, egymásnak ellentmondó bemenetek biztonságos kezelésére, a felmerülő problémák kezelésére, és jelzésére a CAN hálózaton keresztül az adatgyűjtő felé.
Ezen feladat megoldásához szükséges szoftveresen emulált hardver különböző analóg és digitális ki- és bemenetekkel rendelkezik, CAN hálózaton azonban nem küld és nem is fogad adatokat, így azok tesztszoftveres kezelése sem szükséges.
Az alkalmazás futási képe a 9. ábraán látható:



9. ábra Ablakemelő emulátor futási képe

A fájlkezeléshez, futási mód és hardver választáshoz valamint a logoláshoz tartozó kezelőszervek az ablak bal oldalán találhatók. Középen az ablak helyzetét vizualizáló kijelző, melyet a különböző állapotokat és működőképességeket jelző ledek és kapcsolók vesznek körül. Ellenőrzésképpen a jobb felső sarokban láthatóak a különböző kiadott és beolvasott feszültségek. Középen alul a fájlok elérési útvonalai találhatók, jobb alul pedig a vezérlésért felelős kapcsolók és egy összekötési segédlet (mely csak részben látszik).

6.2.1. Ki- és bemenetek

Az emulációs feladat végrehajtásához a program a National Instruments MyDAQ 7. táblázatban megjelölt ki- és bemeneteit használja:

Csatorna	Jel
AI - 0	5V -os PWM motorhajtás
DI - 0	iránybemenet (1:fel, 0:le)
DO - 2	Vezérlés: fel (1: aktív)
DO - 3	Vezérlés: le (1: aktív)
DO - 4	felső végállás jelző (1: aktív)
DO - 5	alsó végállás jelző (1: aktív)
AO - 0	Analóg pozíció [0V;3V]
AO - 1	Motoráram [0;1] normál üzemben, *5 beragadás esetén

7. táblázat Ablakemelő ki- és bemenetei

6.2.2. Szimulációs lehetőségek

Az emulátor program képes egy elektromos ablakemelő üzemszerű működése közben általánosságban fellépő vezérlési helyzetek, valamint hardveres eredetű hibák szimulálására. Az emulációval létrehozható, és a validációs tesztszekvenciában szereplő vezérlések a következők:

- Ablak teljes le- és felhúzása bármely pozícióból, akár ellenkező irányú mozgás megváltoztatásával is, minden szenzor működőképessége esetén
- Ablak rövid távú mozgatása lefelé és felfelé, minkét irányban (ez a funkció a vezérlőgomb hosszan nyomva tartásával érhető el)
- Ablak közepes távú mozgatása lefelé és felfelé, mindkét irányban (ez a funkció az egyik, majd másik vezérlés rövid kiadásával érhető el)

- Ablak végállásba történő vezérlése a végállás-érzékelők működésképtelensége esetén – *ekkor az ablaknak végállásban meg kell állnia, hibaüzenetet CAN hálózaton továbbítania, valamint tárolnia a hibát*
- Ablak beragadása mozgás közben – *ekkor az ablakot meg kell állítani, hibaüzenetet CAN hálózaton továbbítani, és a hibát tárolni*
- PWM szakadás bármely vezérlés esetén – *a mozgás kivezérlését meg kell szüntetni, a hibát CAN hálózaton továbbítani és tárolni*
- Ablak végállásba történő vezérlése a végállás-érzékelők és a pozíció szenzor működésképtelensége esetén – *az ablakot végállásban meg kell állítani, a hibát CAN hálótaton továbbítani és tárolni*

A dőlt betűvel szedett részek a hallgatók által fejlesztett vezérlőegységtől elvárt működést írják le adott bemenetek, illetve állapot esetén.

6.2.3. Fájlkezelés

Az ablakemelő emulátor program a későbbi hibaanalízis megkönnyítése érdekében log fájlt készít (amennyiben ezt a funkciót nem kapcsolták ki). A készített adatfájlban a 8. táblázatban leírt paraméterek kerülnek rögzítésre:

Oszlop neve	Jelentés
Time	A program indulása óta eltelt másodpercek száma
Delay	Két mintavétel között eltelt ezredmásodpercek száma
Window	Ablak helyzete [0;3V], 0 az alsó végállás
PWM	PWM jel [0;5V]
Err_PWM	PWM szakadás (1: szakadás)
Block	Ablak beragadás (1: beragadás)
Pos_OK	Helyzetérzékelő működőképessége (1: működik)
AV_OK	Alsó végállás-érzékelő működőképessége (1: működik)
FV_OK	Felső végállás-érzékelő működőképessége (1: működik)
Stat_AV	Felső végállás-érzékelő értéke (1: aktív)
Stat_FV	Alsó végállás-érzékelő értéke (1: aktív)
CMD_Up	Felhúzás vezérlés (1: aktív)
CMD_Dwn	Lehúzás vezérlés (1: aktív)

8. táblázat Ablakemelő log fájljának tartalma

Az emulátor program képes automatizált teszt szekvencia végrehajtására is, ekkor a 9. táblázatban ismertetett bemeneteket kell beolvasnia az erre szolgáló fájlból:

Jel	Érvényes tartomány
Log delay [ms]	[10;1000]
PWM szakadás	0;1
Block	0;1
Pos_OK	0;1
AV_OK	0;1
FV_OK	0;1
CMD_Up	0;1
CMD_Dwn	0;1

9. táblázat Ablakemelő automatikus futtatási fájljának tartalma

6.3. Központi zár

A központi zár hallgatói feladatnál a cél egy három ajtós jármű elektronikus aktuátorokkal működtetett zárszerkezeteit vezérlő központi egység programjának megírása. A felhasználói bemenetek megadása a tesztprogramon keresztül történik, amelynek kimenetei az egyes ajtók és zárak állapotai, valamint a vezérlő jelek. Az emulátor bemenetei a vezérlőegységtől érkező utasítások.

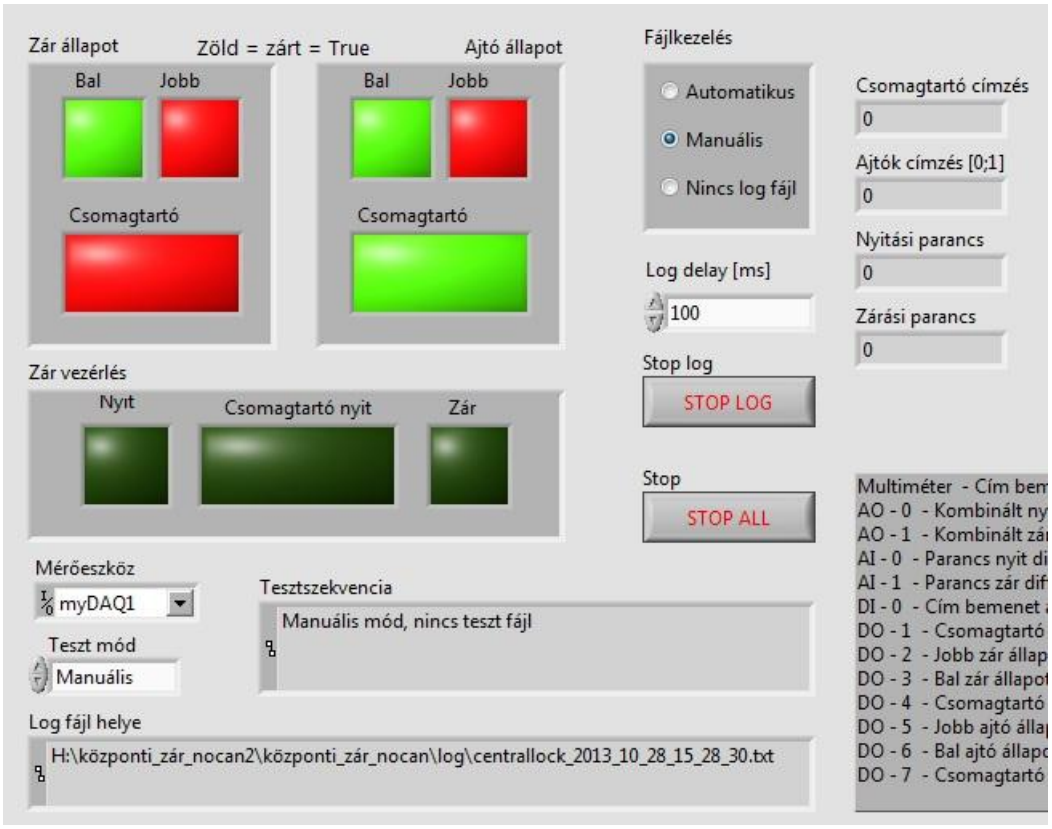
A rendszernek képesnek kell lennie normál körülmények között hibamentesen működni, melybe beletartozik:

- Az egyes zárak megfelelő vezérlése
- Státusz üzenetek küldése CAN hálózaton
- CAN hálózaton érkező utasítások végrehajtása

Ezen túlmenően képesnek kell lennie a fellépő hardveres hibák, egymásnak ellentmondó bemenetek biztonságos kezelésére, a felmerülő problémák kezelésére, és jelzésére a CAN hálózaton keresztül az adatgyűjtő felé.

Ezen feladat megoldásához szükséges szoftveresen emulált hardver különböző analóg és digitális ki- és bemenetekkel rendelkezik, CAN hálózaton azonban nem küld és nem is fogad adatokat, így azok tesztszoftveres kezelése sem szükséges.

Az alkalmazás futási képe a 10. ábraán látható:



10. ábra Központi zár emulátor futási képe

A hardverkezeléshez, teszt mód választáshoz tartozó kezelőszervek, valamint a fájl elérési utak a képernyő alján találhatók, a logoláshoz és fájlkezeléshez tartozók pedig középen. A zárak, ajtók állapota, illetve a vezérlés a képernyő bal oldalán helyezkedik el, jobb oldalt az ablak legszélén pedig a beolvasott és kiadott feszültségeket ellenőrző numerikus indikátorok, valamint az összekötést segítő panel (mely csak részben látszik).

6.3.1. Ki- és bemenetek

Az emulációs feladat végrehajtásához a program a National Instruments MyDAQ 10. táblázatban megjelölt ki- és bemeneteit használja:

Csatorna	Jel
Multiméter	Cím bemenet csomagtartó (1: vezérlés)
AO - 0	Kombinált nyitó gomb (1: vezérlés)
AO - 1	Kombinált záró gomt (1: vezérlés)
AI - 0	Parancs nyit (digitális jel) (>3V nyit)
AI - 1	Parancs zár (digitális jel) (>3V zár)
DI - 0	Cím bemenet ajtók (1: vezérlés)
DO - 1	Csomagtartó nyitó gomb (1: vezérlés)
DO - 2	Bal zár állapot (0: nyitva)
DO - 3	Jobb zár állapot (0: nyitva)
DO - 4	Csomagtartó zár állapot (0: nyitva)
DO - 5	Bal ajtó állapot (0: nyitva)
DO - 6	Jobb ajtó állapot (0: nyitva)
DO - 7	Csomagtartó állapot (0: nyitva)

10. táblázat Központi zár ki- és bemenetei

6.3.2. Szimulációs lehetőségek

Az emulátor program képes egy elektromos központi zár üzemszerű működése közben általánosságban fellépő vezérlési helyzetek, valamint hardveres eredetű hibák szimulálására. Az emulációval létrehozható, és a validációs tesztszekvenciában szereplő vezérlések a következők:

- Ajtók zárszerkezetének nyitása, zárása a csomagtartóval együtt
- Csomagtartó zárszerkezetének nyitása, zárása az ajtóktól függetlenül
- Ajtók zárszerkezetének zárása nyitott ajtónál, *ekkor a program nem zárhatja a nyitott állapotban lévő ajtókat*
- Csomagtartó zárszerkezetének zárása nyitott csomagtartó esetén, *ekkor a program nem zárhatja a csomagtartót*
- Egyszerre kiadott nyitó és záró utasítások, *ekkor a programnak az első kell végrehajtania, a hibát CAN hálózaton továbbítania és az EEPROM-ban tárolnia*
- Zárt ajtók állapotának módosítása zártról nyitottra, *ekkor a programnak a hibát CAN hálózaton továbbítani és EEPROM-ban tárolni kell.*

6.3.3. Fájlkezelés

Az központi zár emulátor program a későbbi hibaanalízis megkönnyítése érdekében log fájlt készít (amennyiben ezt a funkciót nem kapcsolták ki). A készített adatfájlban a 11. táblázatban ismertetett paraméterek kerülnek rögzítésre:

Oszlop neve	Jelentés
Time	A program indulása óta eltelt másodpercek száma
Delay	Két mintavétel között eltelt ezredmásodpercek száma
Lock_L	Bal zár állapot (1: zárt)
Lock_R	Jobb zár állapot (1: zárt)
Lock_T	Csomagtartó zár állapot (1: zárt)
Unlock	Zár vezérlés: nyit (1: aktív)
Lock	Zár vezérlés: zár (1: aktív
Trunk	Zár vezérlés: csomagtartó nyit (1: aktív)
Stat_L	Bal ajtó állapot (1: zárt)
Stat_R	Jobb ajtó állapot (1: zárt)
Stat_T	Csomagtartó ajtó állapot (1: zárt)

11. táblázat Központi zár log fájljának tartalma

Az emulátor program képes automatizált teszt szekvencia végrehajtására is, ekkor a 12. táblázatban szereplő bemeneteket kell beolvasnia az erre szolgáló fájlból:

Jel	Érvényes tartomány
Log delay [ms]	[10;1000]
Stat_L	0;1
Stat_R	0;1
Stat_T	0;1
Unlock	0;1
Lock	0;1
Trunk	0;1

12. táblázat Központi zár automatikus futtatási fájljának tartalma

6.4. Trip computer

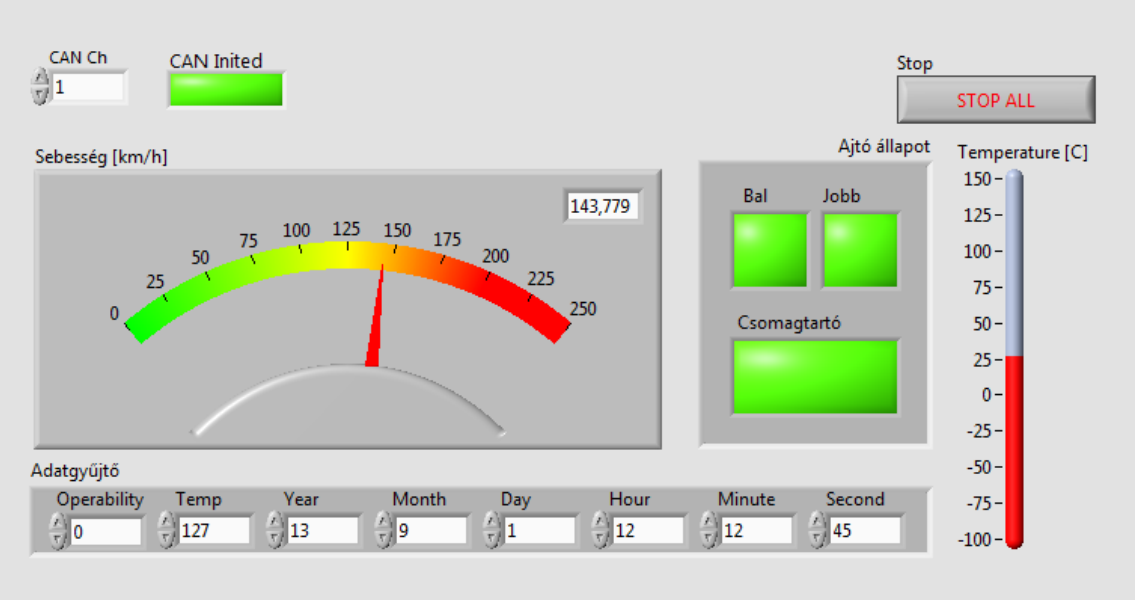
A Trip Computer feladat során a hallgatók célja egy olyan fedélzeti egység készítése, mely a jármű CAN hálózatán lévő adatok közül az központi zár, a tempomat és az adatgyűjtő által szolgáltatott információkat jeleníti meg grafikus kijelzőn. Ezek a(z)

- dátum, idő,
- hőmérséklet,
- sebesség, és

- ajtók állapota.

Ezen túlmenően átlagsebességet, összes megtett utat is számít, valamint rendelkezik egy nullázható napi kilométer számlálóval is.

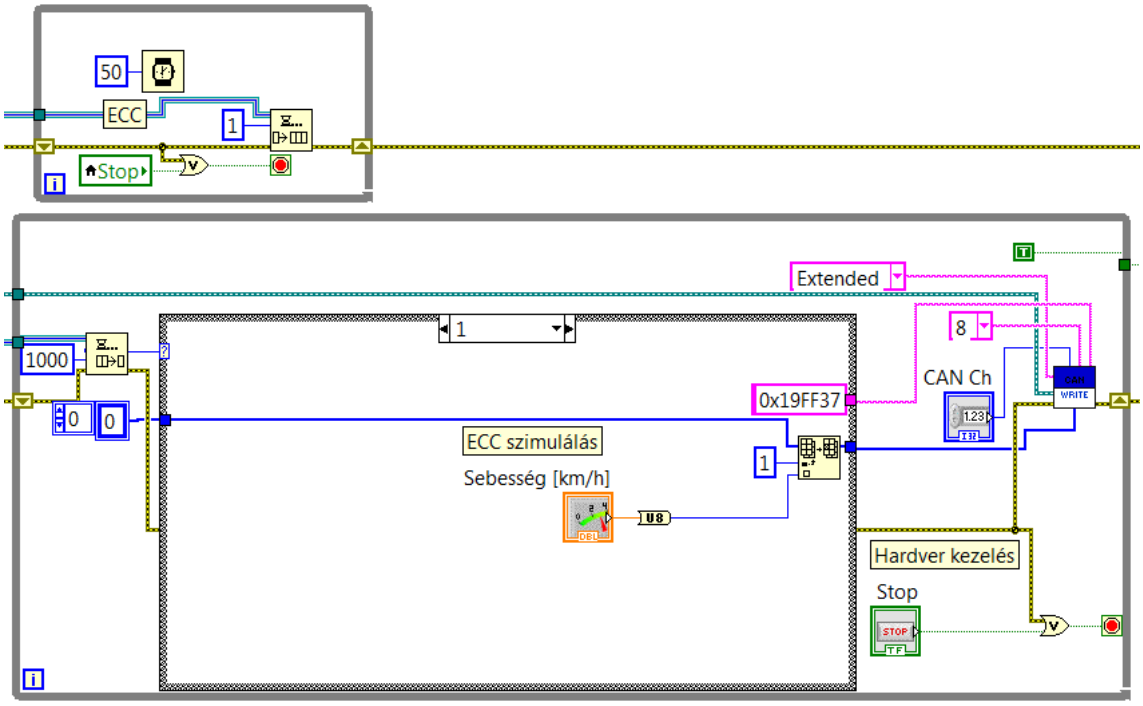
Ezen feladat esetében értelemszerűen a képernyőkép szoftveres monitorozása feleslegesen bonyolult és teljességgel szükségtelen plusz feladatot jelentene, így a tesztprogram feladata a megfelelő státuszüzenetek dbc fájl szerinti elkészítésében és ütemezett kiküldésében (más szóval a hardverek CAN üzeneteinek szimulációjában) merül ki. Vagyis a tesztszoftver a CAN interfészen kívül nem használ semmilyen más analóg vagy digitális ki- vagy bemenetet. A 5.4. pontban említettnek megfelelően a trip computer által monitorozott kezelőszervek kinézete és működése a tesztszoftveren belül teljesen megegyezik az egyes egységekhez készült egyedi tesztszoftvernél találhatóval. Az alkalmazás futási képe a 11. ábraán látható:



11. ábra Trip computer tesztszoftver futási képe

A szoftver elkészítésében a legnagyobb kihívást a különböző időzítésű státuszüzenetek megfelelő gyakoriságú és ugyanazon hardveren történő kiküldése jelentette.

Ahogy az a 12. ábraán látható, egy megfelelően időzített ciklus (minden CAN üzenethez saját ciklus tartozik) egy várólista végére fűzi az üzenethez tartozó ID-t, amit egy szeparált ciklus dolgoz fel. Ez a ciklus egészen addig (vagy 1000 ms-os timeout-ig) vár, míg valamely időzítő ciklustól parancsot nem kap. Ekkor feltölti a CAN üzenetet az emulációhoz releváns adatokkal, megfelelően címzi, majd kiküldi a hálózatra.



12. ábra Időzített CAN üzenetküldés kódrészlet

Így a feladatot a hardveres ütközések még elméleti lehetőségének kizárásával sikerült megoldani, miközben a rendszer a célhardver erőforrásait optimálisan használja fel, valamint rugalmasan bővíthető. Ennek az ára a szoftver számítógép-oldali hardverigényének növekedése, ez azonban minimális szinten maradt: az alkalmazás mérete, illetve futás közbeni processzoridő- és műveleti memória-igénye az 5.1 alfejezetben tárgyalt irányjelző tesztprogram alatt maradt.

6.4.1. Szimulációs lehetőségek

A program a következő hibajelenségek emulálására alkalmas:

- Egyes egységektől érkező adatok az értelmezési tartományukon kívül esnek vagy hiányosak
- Egy vagy több mikrokontroller nem forgalmaz
- Hibás a CAN hálózathoz tartozó Bode Rate beállítása a hálózaton

6.5. Kanyarkövető fényszóró vezérlés

A kanyarkövető fényszóró vezérlési feladatnál a hallgatók a jármű külső világítását kezelik, a következő vezérlési módokkal:

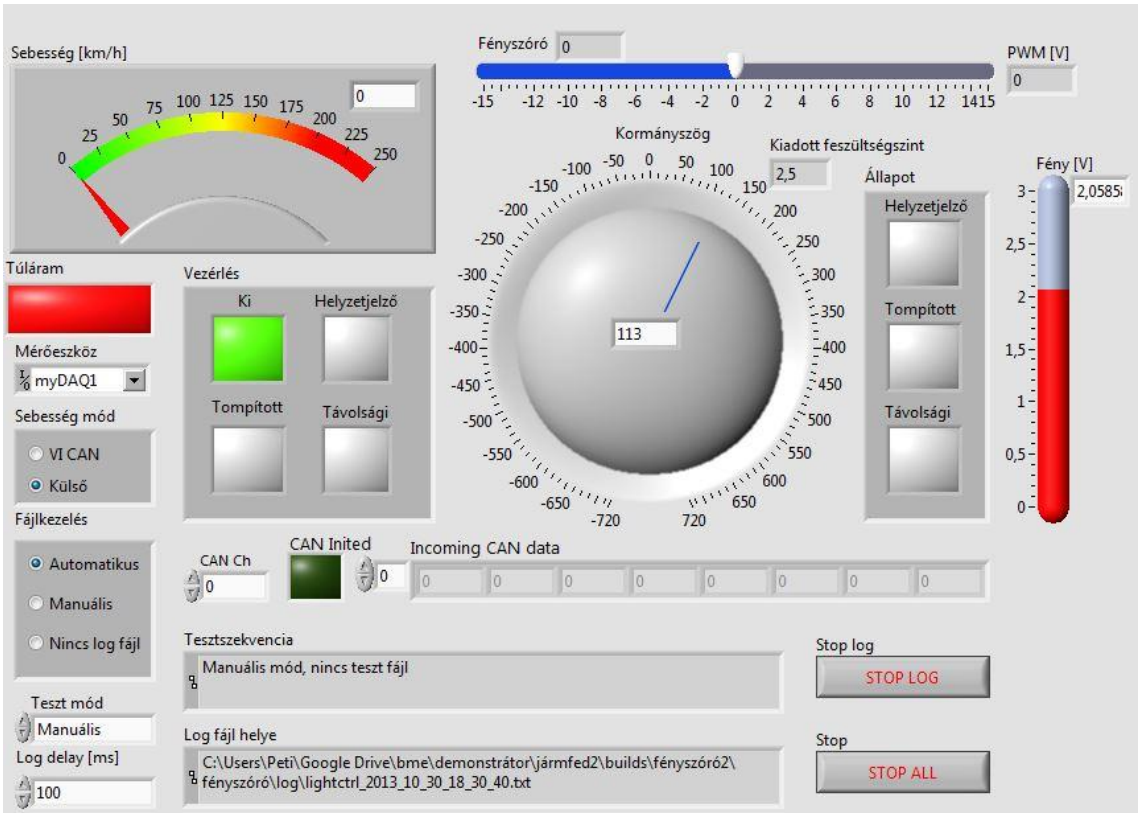
- Ki
- Helyzetjelző
- Tompított fényszóró
- Távolsági fényszóró

Ezen funkcionalitást bővítik egyrészt a külső fényviszonyoktól függő fényszóróvezérléssel, valamint 70 km/h fölötti sebességtartományban kanyarkövető fényszóró vezérléssel.

Mivel ennyiféle különböző analóg és digitális jel szimulálása megnöveli a szoftverben elkövetett, helyes működést alapvetően befolyásoló hibák valószínűségét, jelentősen bonyolítja a helyes bekötést, további hardverelemeket (NI myDAQ), vagy a hardverszisztéma cseréjét teszi szükségessé, valamint az iparban alkalmazott műszaki megoldások sem ezen séma alapján készülnek (pontosan a fenti okok miatt). A szimuláció ezért komplex módon, analóg és digitális ki- és bemenetekkel, valamint CAN üzenetek küldésével és fogadásával emulálja a mikrovezérlő környezetét.

Az emulátor program futási képe a 13. ábraán látható.

A fájlkezeléshez, hardverválasztáshoz és futási mód választáshoz tartozó kezelőszervek az ablak bal oldalán helyezkednek el, míg a kezelt fájlok elérési útjai alul találhatók. Középen az emulátor program különböző állapot-kimeneteinek kezelőszervei láthatók, felül és jobb oldalt a vezérlőegység által adott jelek. Szintén középen látható a kimenő CAN üzenet tartalma (ezt a program automatikusan kitölti, amennyiben a CAN mód választó „VI CAN” állásban van).



13. ábra Fényszóró vezérlés futási képe

6.5.1. Ki- és bemenetek

A szimuláció így a 13. táblázatban szereplő ki- és bemenetekkel rendelkezik (melyek értelemszerűen be- és kimenetek a mikrovezérlő oldalán):

Csatorna	Jel
AI - 0	PWM fényszóró elfordítás (0-5V)
AO - 0	Kormányszög (0-5V)
AO - 1	Környezeti fény (0-3V)
DO - 0	Vezérlés: kikapcsolás (1: aktív)
DO - 1	Vezérlés: helyzetjelző (1: aktív)
DO - 2	Vezérlés: tompított fényszóró (1: aktív)
DO - 3	Vezérlés: távolsági fényszóró (1: aktív)
DI - 4	Visszajelzés: helyzetjelző
DI - 5	Visszajelzés: tompított fényszóró (1: aktív)
DI - 6	Visszajelzés: távolsági fényszóró (1: aktív)
DO - 7	Túláram jelző (1: túláram)
CAN IN: 0x19FF37	Tempomat (ECC) állapotvektor
CAN OUT: 0x19FF40	Fényszóró (HLC) állapotvektor

13. táblázat Fényszóró vezérlés ki- és bemenetei

Az analóg és digitális jeleket az 5.1. fejezetben leírtaknak megfelelően National Instruments MyDAQ-val, míg a CAN üzeneteket Inventure WeCAN eszközzel emulálom.

6.5.2. Szimulációs lehetőségek

Az emulátor program képes egy irányjelző működése közben általánosságban fellépő vezérlési helyzetek, valamint hardveres eredetű hibák szimulálására. Az emulációval létrehozható, és a validációs tesztszekvenciában szereplő vezérlések a következők:

- Világítás ki – helyzetjelző világítás – tompított fényszórók kivezérlése
- Tompított és távolsági fényszóró egyidejű kivezérlése – *csak távolsági fényszórót kell kivezérelni*
- Kormányszög változtatása a kanyarkövető fényszóróvezérlés kapcsolási sebessége alatt és felett – *a kapcsolási sebesség (70 km/h) felett a kormányszög függvényében el kell fordítani a tompított fényszórót*
- Környezeti fény változtatására, akár ugrásszerűen is, minden vezérlési módban – *automatikus mód esetén kikapcsolt állapotból a kapcsolási határ (1,3V) alatti fényerősségnél fel kell kapcsolni a tompított fényt*
- Túláram jelzése bármely vezérlési állapot alatt – *adott vezérlésről el kell váltani, státuszüzenetben a hibát továbbítani és EEPROM-ban tárolni*
- Sebességadat CAN hálózaton küldésének leállítása futás közben – *ekkor a fényszórót középállásba kell állítani*

A dőlt betűvel szedett részek a hallgatók által fejlesztett rendszertől elvárt működést írják le adott bemenetek, illetve állapot esetén.

6.5.3. Fájlkezelés

Az irányjelző emulátor program a későbbi hibaanalízis megkönnyítése érdekében log fájlt készít (amennyiben ezt a funkciót nem kapcsolták ki). A készített adatfájlban a 14. táblázatban leírt paraméterek kerülnek rögzítésre:

Oszlop neve	Jelentés
Time	A program indulás óta eltelt másodpercek száma
Delay	Két mintavétel között eltelt ezredmásodperce száma
Speed	Jármű sebessége [km/h]
Light	Külső fény erőssége
Off!	Vezérlés: Kikapcsolás (1: aktív)
Pos!	Vezérlés: Helyzetjelző (1: aktív)
Dimmer!	Vezérlés: Tompított fényszóró (1: aktív)
Head!	Vezérlés: Távolsági fényszóró (1: aktív)
Ex_Curr	Túláram jelző (1: túláram)
St_Pos	Státusz: Helyzetjelző (1: aktív)
St_Dimm	Státusz: Tompított fényszóró (1: aktív)
St_Head	Státusz: Távolsági fényszóró (1: aktív)
CAN_OUT	Kimenő CAN üzenet
CAN_IN	Bejövő CAN üzenet

14. táblázat Fényszóró vezérlés log fájljának tartalma

A program képes automatikus tesztszekvenciák végrehajtására is, ehhez a beolvasott szöveges fájlnak a 15. táblázatban szereplő adatokat kell tabulátorral határolt szöveges fájl formátumban tartalmaznia:

Jel	Érvényes tartomány
Log delay [ms]	[10;1000]
Velocity [km/h]	[0;250]
Angle [°]	[-720;720]
Ambient light [V]	[0;3]
Off!	0;1
Position!	0;1
Dimmer!	0;1
Headlight!	0;1

15. táblázat Fényszóró vezérlés automatikus futtatási fájljának tartalma

7. Összefoglalás

A modern járműelektronikában – az integrált áramköri technológiák gyors fejlődésének köszönhetően – sorra jelennek meg a vezetést segítő, biztonságot illetve kényelmet növelő technikai berendezések. A dolgozat bemutatta a járművek elektronikus vezérlőegységeinek fejlődősét, mely indokolja a fejlett emulációs keretrendszerek létrehozását. Kitér e rendszerek fejlesztésével kapcsolatos alapvető kívánalmakra, valamint ismerteti az ezek teljesítéséhez általam használt módszereket.

Az dolgozat hangsúlyozza a tesztszoftverek közös vonásait, melyek nagyobb használhatóságot és könnyebb átjárást eredményeznek, valamint a fejlesztés során elkövetett hibák számát csökkentik. Külön kiemeli a szoftverek validációs és verifikációs képességeit, a log fájlok készítésének különböző módjait, valamint a hardveres ütközések elkerülése érdekében tett lépéseket, mind a felhasználó által is érzékelhető, mind a program kódjában lévő megoldásokat ismertetve.

Ezek után bemutatja az egyes tesztszoftverek felhasználói felületét, a többitől eltérő elemeit, azok egyedi szoftveres megoldásait, hardveres megvalósítását és összekötésének módját a mérendő vezérlőegységgel. Megismerteti az olvasót az egyes szoftverek által készített adatfájlokkal, valamint az automatikus futtatáshoz szükséges fájl tartalmával.

A jövőben – a szakirányra jelentkező hallgatók számának növekedésével – várhatóan szükségessé válik több, hasonló emulációs feladatot végrehajtó tesztprogram elkészítése. Ezen keretrendszer használatával az újabb programok elkészítése jelentősen könnyebbé válik, a készített log fájlok hasonlóságait felhasználva lehetséges olyan analizáló alkalmazás fejlesztése, mely minden, a keretrendszerben meghatározott specifikációk alapján készült emulátor adatfájljait képes feldolgozni és grafikusan megjeleníteni.

8. Ábra- és táblázatjegyzék

9. Forrásjegyzék

[1] Gáldi Péter: *Autonóm párhuzamos parkolási rendszer fejlesztése.* 2012, 4.-13. o.

[2] BME Járműfedélzeti rendszerek I. Előadás

[3] Trevor O. Jones, Joseph F. Ziomek: *The word automotive electronics market*. 1980, 4.o.

[4] Conceptcarz, *1988 Mercedes-Benz 560 SL new, pictures, and information*. online: http://www.conceptcarz.com/vehicle/z12163/Mercedes-Benz-560-SL.aspx

[5] Jörg Schäuffele, Thomas Zurawka, *Automotive Software Engineering*. Vieweg. 2006, 98.-103. o.

[6] IEC International Electrotechnical Commision: IEC 61508 – *Functiona lSafety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*. 1998

[7] ADAC et all, *ADAC Test Notbremsenassistenten 2012*. 2012, online: http://www.adac.de/infotestrat/tests/assistenzsysteme/notbremsassistent_2012/default.aspx?ComponentId=142547&SourcePageId=31968

[8] ADAC et all, *Comparative test of advanced emergency braking systems*. 2011

[9] Eetimes at all, online: http://www.eetimes.com/ContentEETimes/Images/Schweber/NP%20ADI%20AD8283%20auto%20radar/ADI%20auto%20electronics.gif

[10] Wikipedia: *Grumman F-14 Tomcat*. 2012, online: http://en.wikipedia.org/wiki/Grumman_F-14_Tomcat#Background

[11] BOSCH et all, *Szenzorok a gépjárművekben*.2008, 17. o.

[12] Dr.-Ing. J. Kreuzinger, *Softwareentwicklung im Automobilbau.*VDI-Berichte Nr. 1907, 2005, 75.o.

[13] Dipl.-Ing. S. Thissen, Dipl.-Ing. R. Faller, *Flexible Sicherheitskonzepte an einem Beispiel für Fahrerassistenzsysteme*. VDI-Berichte Nr. 1907, 2005, 95.o.

[14] The Atlantic, *Google's Self-Driving Cars: 300,000 Miles Logged, Not a Single Accident Under Computer Control.* 2012, online: http://www.theatlantic.com/technology/archive/2012/08/googles-self-driving-cars-300-000-miles-logged-not-a-single-accident-under-computer-control/260926/

[15] http://www.mikroe.com/bigavr/

[16] http://sine.ni.com/np/app/main/p/ap/academic/lang/hu/pg/1/sn/n17:academic,n21:16781/fmid/6307/

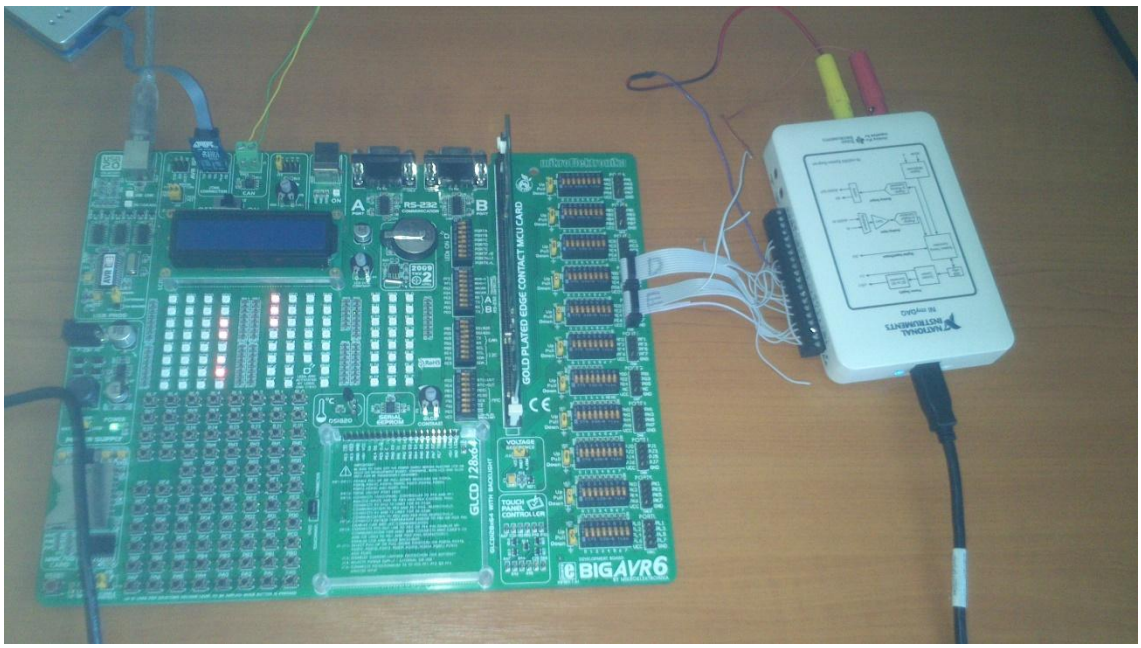
[17] http://www.inventure.hu/wecan_usb_hu

[18] http://physics.wku.edu/phys318/?attachment_id=38

I. Függelék Teszteléshez használt eszközök

14. ábra Mikroelektronika BIGAVR6 fejlesztői panel [15]

15. ábra National Instruments MyDAQ [18]



16. ábra Hardvereszközök összekötése központi zár feladatnál

II. Függelék BME_Lab_Network dbc fájl tartalma

```
BS_:

BU_: Body_computer TS_ECU EWL_ECU CL_ECU
VAL_TABLE_ VT_TS_CNT 15 "Infinite" 0 "Stop" ;
VAL_TABLE_ VT_TS_ADR 3 "Both" 2 "Right" 1 "Left" 0 "None" ;
VAL_TABLE_ VT_CL_Lock 10 "Close All" 12 "Open Front" 15 "Open All" 3
"Open Rear" 0 "Do Nothing" ;


BO_ 2148138807 MSG_CLR_Errors: 1 Body_computer
 SG_ CLR_Wnd_Error : 2|1@1+ (1,0) [0|0] ""  EWL_ECU
 SG_ CLR_TS_Error : 1|1@1+ (1,0) [0|0] ""  TS_ECU
 SG_ CLR_CL_Error : 0|1@1+ (1,0) [0|0] ""  CL_ECU

BO_ 2149187380 MSG_ST_EWL: 8 EWL_ECU
 SG_ St_Wnd_Sw_Up : 27|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_Wnd_Sw_Down : 26|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_Wnd_EP_Low : 25|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_Wnd_EP_High : 24|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_Wnd_Position : 16|8@1+ (0.4,0) [0|102] "%"  Body_computer
 SG_ St_Wnd_Operability : 8|8@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_Wnd_Current : 0|8@1+ (0.04,0) [0|0] "A"  Body_computer

BO_ 2149187381 MSG_ST_TS: 8 TS_ECU
 SG_ St_TS_Read_RR : 13|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_TS_Read_RL : 12|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_TS_Read_FR : 11|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_TS_Read_FL : 10|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_TS_Out_Right : 9|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_TS_Out_Left : 8|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_TS_Operability : 0|8@1+ (1,0) [0|0] ""  Body_computer

BO_ 2149187382 MSG_ST_CL: 8 CL_ECU
 SG_ St_CL_Operability : 8|8@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_CL_Lock_Right : 5|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_CL_Lock_Rear : 4|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_CL_Lock_Left : 3|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_CL_Door_Right : 2|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_CL_Door_Rear : 1|1@1+ (1,0) [0|0] ""  Body_computer
 SG_ St_CL_Door_Left : 0|1@1+ (1,0) [0|0] ""  Body_computer

BO_ 2148138806 MSG_CMD_CL: 8 Body_computer
 SG_ Cmd_CL_Door_Lock : 0|4@1+ (1,0) [0|0] ""  CL_ECU

BO_ 2148138805 MSG_CMD_TS: 8 Body_computer
 SG_ Cmd_TS_Count : 2|4@1+ (1,0) [0|0] ""  TS_ECU
 SG_ Cmd_TS_Address : 0|2@1+ (1,0) [0|0] ""  TS_ECU

BO_ 2148138804 MSG_CMD_EWL: 8 CL_ECU
 SG_ Cmd_Wnd_Pos : 0|8@1+ (0.4,0) [0|100] ""  EWL_ECU

BO_TX_BU_ 2148138805 : CL_ECU,Body_computer;
BO_TX_BU_ 2148138804 : Body_computer,CL_ECU;


CM_ BU_ TS_ECU "Turn Signal ECU";
```

```
CM_ BU_ EWL_ECU "Electric Window Lift ECU
";
CM_ BU_ CL_ECU "Central Locking ECU";
CM_ SG_ 2148138807 CLR_Wnd_Error "Window ECU Clear Errors ";
CM_ SG_ 2148138807 CLR_TS_Error "Error Clearing Turn Signal ECU";
CM_ SG_ 2148138807 CLR_CL_Error "Central Locking ECU Clear Errors ";
CM_ SG_ 2149187380 St_Wnd_Sw_Up "Window Switch Up";
CM_ SG_ 2149187380 St_Wnd_Sw_Down "Window Switch Down";
CM_ SG_ 2149187380 St_Wnd_EP_Low "Window Lower End Position 1- TRUE";
CM_ SG_ 2149187380 St_Wnd_EP_High "Window Higher End Position 1-TRUE";
CM_ SG_ 2149187380 St_Wnd_Position "State of the Window 0-100%";
CM_ SG_ 2149187380 St_Wnd_Current "Window drive motor current";
CM_ SG_ 2149187381 St_TS_Read_RR "Turn Signal Check Rear Right";
CM_ SG_ 2149187381 St_TS_Read_RL "Turn Signal Check Rear Left";
CM_ SG_ 2149187381 St_TS_Read_FR "Turn Signal Check Front Right";
CM_ SG_ 2149187381 St_TS_Read_FL "Turn Signal Check Front Left";
CM_ SG_ 2149187381 St_TS_Out_Right "Turn Signal Right Output";
CM_ SG_ 2149187381 St_TS_Out_Left "Turn Signal Left Output";
CM_ SG_ 2149187382 St_CL_Lock_Right "Central Locking Lock State
Right";
CM_ SG_ 2149187382 St_CL_Lock_Rear "Central Locking Lock State Rear";
CM_ SG_ 2149187382 St_CL_Lock_Left "Central Locking Lock State Left";
CM_ SG_ 2149187382 St_CL_Door_Right "Central Locking Door State Right
1 - Opened 0 - Closed";
CM_ SG_ 2149187382 St_CL_Door_Rear "Central Locking Door State Rear 1
- Opened 0 - Closed";
CM_ SG_ 2149187382 St_CL_Door_Left "Central Locking Door State Left 1
- Opened 0 - Closed";
CM_ SG_ 2148138806 Cmd_CL_Door_Lock "Door Lock Command";
CM_ BO_ 2148138805 "Turn Signal Command message";
CM_ SG_ 2148138805 Cmd_TS_Count "Blink count Command (together with
address)";
CM_ SG_ 2148138805 Cmd_TS_Address "Blink Command address (together
with count)";
CM_ BO_ 2148138804 "Command Set For Window Operations";
CM_ SG_ 2148138804 Cmd_Wnd_Pos "Window Position Command";
BA_DEF_ "Baudrate" INT 0 1000000;
BA_DEF_ "BusType" STRING ;
BA_DEF_ BU_ "NodeLayerModules" STRING ;
BA_DEF_ BU_ "ECU" STRING ;
BA_DEF_ BU_ "CANoeJitterMax" INT 0 0;
BA_DEF_ BU_ "CANoeJitterMin" INT 0 0;
BA_DEF_ BU_ "CANoeDrift" INT 0 0;
BA_DEF_ BU_ "CANoeStartDelay" INT 0 0;
BA_DEF_DEF_ "Baudrate" 500000;
BA_DEF_DEF_ "BusType" "";
BA_DEF_DEF_ "NodeLayerModules" "";
BA_DEF_DEF_ "ECU" "";
BA_DEF_DEF_ "CANoeJitterMax" 0;
BA_DEF_DEF_ "CANoeJitterMin" 0;
BA_DEF_DEF_ "CANoeDrift" 0;
BA_DEF_DEF_ "CANoeStartDelay" 0;
BA_ "BusType" "CAN";
BA_ "Baudrate" 500000;
VAL_ 2148138806 Cmd_CL_Door_Lock 10 "Close All" 12 "Open Front" 15
"Open All" 3 "Open Rear" 0 "Do Nothing" ;
VAL_ 2148138805 Cmd_TS_Count 15 "Infinite" 0 "Stop" ;
VAL_ 2148138805 Cmd_TS_Address 3 "Both" 2 "Right" 1 "Left" 0 "None" ;
```