



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

FUNKČNÍ HIL TESTY UNIVERZÁLNÍ ELEKTRONICKÉ ŘÍDICÍ JEDNOTKY

FUNCTIONAL HIL TESTING OF AN ELECTRONIC CONTROL UNIT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

Jan Štramberský

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Martin Brablc

BRNO 2022

Zadání bakalářské práce

Ústav: Ústav mechaniky těles, mechatroniky a biomechaniky
Student: **Jan tramberský**
Studijní program: Aplikované vědy v inženýrství
Studijní obor: Mechatronika
Vedoucí práce: **Ing. Martin Brabc**
Akademický rok: 2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Funkční HIL testy univerzální elektronické řídicí jednotky

Stručná charakteristika problematiky úkolu:

Testování signálové elektroniky je standardem ve většině odvětví průmyslu, ve kterých se vyskytují řídicí jednotky pro pohony, senzory a HMI zařízení. Nejčastějším způsobem jsou tzv. Hardware-in-the-loop (HIL) testy, nebo jiné in-the-loop varianty. Tyto testy jsou určeny především ke standardizaci kontroly kvality při sériové výrobě a pro odhalování chyb při vývoji a aktualizacích HW i FW. Tato práce se bude zabývat rozšířením existujícího HW tak, aby umožnil nejen HW, ale i funkční testy konkrétní univerzální řídicí jednotky a jejího FW, a dále metodikou návrhu testů.

Cíle bakalářské práce:

- 1) Proveďte rešerši HIL a SIL testů řídicích jednotek, uveďte příklady možných konfigurací HW, zaměřte se na metodiku návrhu testů.
- 2) Replikujte dřívější experimenty s testovacím HW a proveďte opravy a úpravy nutné pro další postup.
- 3) Navrhněte sadu testů pro vybranou řídicí jednotku a analyzujte její kompletnost.
- 4) Experimentálně proveďte sadu testů na funkční i vadné jednotce.
- 5) Navrhněte způsob návrhu testů pro případ aktualizace FW.

Seznam doporučené literatury:

JONG, Erik de et al. "European White Book on Real-Time Power Hardware in the Loop Testing" (2012).

NELLES, Oliver. Nonlinear system identification: from classical approaches to neural networks and fuzzy models. Berlin: Springer, 2011. ISBN 978-364-2086-748.

LJUNG, Lennart. System identification: theory for the user. 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, 1999. ISBN 978-0136566953.

VALÁŠEK, Michael. Mechatronika. Dot. 1. vyd. Praha: České vysoké učení technické, 1996. ISBN 80-010-1276-X.

NOSKIEVIČ, Petr. Modelování a identifikace systémů. Ostrava: Montanex, 1999. ISBN 80-722-50-0-2.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc.
ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
děkan fakulty

Abstrakt

Tato práce se zabývá funkčním HIL testováním univerzální elektronické řídicí jednotky. Řídicí jednotka je graficky programována v aplikaci, v níž je možné vytvářet stavové automaty s předprogramovanými funkcemi. V rámci práce byl vytvořen HIL test jednotky. Signály jsou vyměňovány mezi jednotkou a PC za pomoci vstupní/výstupní karty. Na PC běží skript HIL testu v programu Matlab. Tento test ověřuje firmware jednotky za pomoci kontroly očekávaného chování jednotlivých funkcí a podmínek přechodů.

Summary

This thesis deals with the functional HIL testing of a universal electronic control unit. The control unit is graphically programmed in an application in which it is possible to create state machines with pre-programmed functions. Within the scope of this work, a HIL test of the unit has been created. Signals are exchanged between the unit and PC using an input/output card. The PC runs the HIL test script in Matlab. This test verifies the firmware of the unit by checking the expected behaviour of the functions and the conditions of the transitions.

Klíčová slova

HIL, testování elektronické řídicí jednotky, funkční testy, stavový automat, simulace, Model Based Design.

Keywords

HIL, ECU testing, functional testing, state machine, simulation, Model Based Design.

Bibliografická Citace

ŠTRAMBERSKÝ, J. *Funkční HIL testy univerzální elektronické řídicí jednotky*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2022. 38 s., Vedoucí bakalářské práce: Ing. Martin Brablc.

Tímto prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Martina Brabce, s použitím materiálů uvedených v seznamu literatury.

Jan Štramborský

Brno 20.5.2022

.....

Tímto děkuji vedoucímu práce Ing. Martinu Brabcovi za cenné rady a připomínky poskytnuté při zpracování této práce. Také bych rád poděkoval Ing. Martinu Appelovi za přínosné konzultace. V neposlední řadě děkuji rodině za podporu při studiu a kolegům z mechatronické laboratoře za vytvoření motivujícího prostředí.

Jan Štramborský

Obsah

1	Úvod	8
2	Teoretická část	9
2.1	HIL Testovací přípravek	9
2.1.1	•ídící jednotka DINO master	10
2.1.2	•ídící jednotka DINO slave	10
2.1.3	Dino aplikace	11
2.2	Model Based Design	12
2.2.1	Kon gurace in-the-loop test	13
2.2.2	Varianty HIL test	14
2.3	Návrh test	16
3	Praktická část	18
3.1	Úpravy dřívějších experiment	18
3.1.1	Hardware test	18
3.1.2	HIL test DC motoru	19
3.1.3	Homing test	22
3.2	Testování jednotky	23
3.2.1	Návrh funkčního testu	23
3.2.2	Analýza test	25
3.2.3	Nalezené chyby	30
3.3	Rozřícení test	31
3.3.1	Přidání funkce	31
3.3.2	Přidání události	32
4	Závěr	34
4.1	Možnosti dalšího vývoje	35
	Literatura	36
	Seznam příloh	38

1 Úvod

V dnešní době je v průmyslu důležité inovovat a přicházet s chytrými řešeními. K tomu dopomáhají možnosti testování prototypů již v počátečních fázích vývoje. Právě proto je velice atraktivní metoda vývoje zvaná Model Based Design (MBD). Tato metoda využívá matematické modely popisující jednotlivé algoritmy, logiku a fyzické chování komponent. Na těchto modelech je možné pozorovat chování vyvíjeného systému jako celku. Postupně lze testovat jejich chování (Model-in-the-loop). Vytvořené modely slouží k vytváření reportů nebo generování kódu. Tento kód je také možné testovat ve smysle s ostatními modely simulující chování celého systému (Software-in-the-loop). Vygenerovaný a otestovaný kód je nahrán na řídicí jednotku, jež bude použita ve finálním produktu. Tuto jednotku lze testovat připojením k počítači, který v reálném čase simuluje chování ovládaného systému (Hardware-in-the-loop). Další požadavek je kladen na testování vyvíjené komponenty nezávisle na ostatních částech systému, se kterými daná komponenta interaguje.[1]

Důležité pro vytvoření efektivního testu je promyslet si, co a kolik toho vůbec testovat. Cílem této práce je naprogramovat právě takový HIL test univerzální elektronické řídicí jednotky slave. Testovanou jednotku lze programovat za pomoci grafické aplikace Dino. Ta umožňuje vytvářet stavové automaty s funkcemi jako řízení napětí, regulace natožení a rychlosti DC motoru. K využívání aplikace Dino navíc není zapotřebí pokročilé znalosti programování. Jednotky najdou uplatnění především v řízení složitých pohybů, automatizaci a v laboratořích, právě pro možnost rychlého a jednoduchého programování.

K HIL testu bude využít testovací přípravek dříve vyroben v mechatronické laboratoři Vysokého učení technického v Brně. Do tohoto přípravku se umísťuje testovaná jednotka. Měření signálů je zajištěné vstupní/výstupní kartou Humusoft MF 624. V mechatronické laboratoři byl také dříve vytvořen kontaktní hardware test jednotky a HIL test DC motoru. Po zprovoznění a úpravách těchto testů je možné z nich dále vycházet.

Vytvořený test by měl verifikovat firmware jednotky. Především otestovat jednotlivé funkce, vytváření událostí a fungování stavového automatu. Zároveň by měl být co nejvíce zautomatizovaný a uživatelsky přívětivý. Součástí práce je využití tohoto testu k prověření a popisu funkčnosti testované jednotky.

2 Teoretická část

2.1 HIL Testovací přípravek

Testovací přípravek potřebný k Hardware test-m byl vyrobený řádkem Zouharem v mechatronické laboratoři. Přípravek slouží pouze k testování jednoho určitého typu DPS. Z tohoto důvodu využívá metodu Bed of Nails, jež je výhodná pro testování většího množství jednotek. Metoda Bed of Nails spočívá v ukládání testované desky na pružné pogo piny, které zajišťují kontakt testovacího přípravku s testovanou deskou. K aretaci desky na přípravku slouží vodící kolíky.[2]

Z přípravku jsou vyvedeny měřené digitální a analogové piny do multifunkční I/O karty MF624. Pin PWM je před vstupem do karty filtrován RC filtrem s dolní propustí s časovou konstantou 10 ms, jelikož frekvence PWM testované jednotky se pohybuje v řádu desítek kHz, což není měřicí karta schopná zaznamenat. Přípravek má dále výstupní piny pro komunikaci s Pickit programátorem a výstupní piny pro napájení a komunikaci s jednotkou Master.[2]

Obrázek 2.1: Testovací přípravek

2.1.1 Řídící jednotka DINO master

Elektronické řídicí jednotky Dino jsou určeny k řízení stroje s mnoha pohony a sensory. Hlavní výhodou jednotek je možnost řízení složitých pohybů bez pokročilé znalosti programování, jelikož je k nim vyvinutá aplikace s jednoduše pochopitelným grafickým programováním. Jednotky najdou využití především v automatizaci a v laboratořích, kde bývá zapotřebí něco rychle naprogramovat a vyzkoušet. Jde tedy o zajímavou alternativu k PLC. Nyní jsou jednotky používány společností Machine planet a.s. v Dinoparcích.[3]

K hlavní řídicí jednotce master lze připojit až 16 podřízených jednotek slave za pomoci sériové komunikační sběrnice RS 485. Program pro všechny tyto jednotky se v podobě tabulek s daty obsahující informace o požadovaném chování jednotek nahrává z PC pomocí USB kabelu přímo do hlavní jednotky. V této množině může být najednou uloženo až 3 různé programy, mezi kterými lze přepínat pomocí tlačítek na desce. Ke komunikaci jednotky master s uživatelem slouží LCD display na desce. Zároveň je možné jednotku připojit k síti Internet pro vzdálenou komunikaci. Samotná hlavní řídicí jednotka je také vybavena programovatelnými digitálními vstupy a výstupy, časovači používaným k přechodem mezi stavy a komunikací pro přehrávání MP3. [3]

2.1.2 Řídící jednotka DINO slave

Primární účel podřízených řídicích jednotek slave je řízení pohonů. Piny na jednotce jsou následovné:[3]

Komunikace s Hlavní řídicí jednotkou:

- ^ 1 pin pro kanál B komunikace prostřednictvím RS 485 (B)
- ^ 1 pin pro kanál A komunikace komunikaci prostřednictvím RS 485 (A)

Vstupní piny řídicí jednotky:

- ^ 4 piny digitálních vstupů (DI)
- ^ 2 piny pro kanály digitálního enkodéru (QEI)
- ^ 2 piny pro koncové spínače (LS)
- ^ 1 pin analogový vstup pro šidlo proudu (CUR)
- ^ 1 pin analogový vstup pro referenční napětí šidla proudu (REF)
- ^ 1 pin analogový vstup pro šidlo otáček (AIM)
- ^ 1 pin obecný analogový vstup (AIG)

Výstupní piny řídicí jednotky:

- ^ 4 piny digitálních výstupů (DO)
- ^ 1 pin digitální výstup pro ovládání brzdy pohonu (BRK)
- ^ 1 pin digitální výstup pro signál enable (EN)
- ^ 1 pin digitální výstup pro určení směru pohonu (DIR)
- ^ 1 pin výstupní signál PWM 0-5 V (PWM)
- ^ 1 pin analogový výstup pro ovládání pohonu 0-10 V (AOM)
- ^ 1 pin obecný analogový výstup 0-5 V (AOG)

Napájení:

- ^ 6 pinů společné země pro připojitelné zařízení (GND)
- ^ 4 piny pro napájení zařízení 5 V (5V)
- ^ 1 pin pro napájení zařízení 3.3 V (3.3V)
- ^ 1 pin pro napájení řídicí jednotky 24 V (Vdd)

Obrázek 2.2: Testovaná řídicí jednotka Slave [3]

2.1.3 Dino aplikace

Aplikace sloužící k programování Dino jednotek je psaná v programu Matlab verze 2019a a k chodu potřebuje nainstalovaný Matlab Runtime 9.2. Při otevření aplikace se uživatel dostane do základního okna. Pro programování je nejprve zapotřebí v aplikaci nakonfigurovat Hardware, tedy určit jaké řídicí jednotky budeme programovat. Po nakonfigurování Hardware je možné v okně graficky programovat stavový automat. Uživatel vytváří v aplikaci stavy, do kterých umísťuje požadované funkce jednotek, jako nastavení a kontrola DI pinů, řízení napětí, regulace polohy a další. Tyto funkce mohou vytvářet události (event), které je možné použít s logickými funkcemi AND a OR k sestavení podmínek do přechodů mezi stavy. Celkem lze v aplikaci vytvořit 32 stavů a 16 událostí pro každou podřízenou jednotku. Aplikace také disponuje speciální funkcí pro vytváření trajektorií, které lze poté použít například pro regulaci na polohu a řízení napětí.[3]

Obrázek 2.3: Prostředí DINO aplikace

Po zhotovení programu je možné jej sestavit a nahrát za pomoci aplikace DINO uploader do hlavní řídicí jednotky. Vytvořený program se uloží ve formě souboru `.DINOBuild`. Ten se skládá z tabulek popisující požadované chování jednotek a lze jej otevřít v Matlabu jako soubor typu `.mat`. V těchto tabulkách jsou uloženy informace o konfiguraci jednotek, jednotlivých stavech a funkcích v nich, podmínkách a přechodech mezi stavy, data k trajektoriím, parametry regulátoru a podobně.[3]

2.2 Model Based Design

Model-Based Design je modelově orientovaná metoda používaná k vývoji dynamických systémů. Model systému obsahuje všechny důležité části systému k popisu jeho chování jako jsou algoritmy, logika a fyzické modely. Tato metoda využívá model systému v jednotlivých částech vývoje, konkrétně výzkumu, návrhu, implementaci a verifikaci a validaci. V těchto částech je model pokročile aktualizován a rozveden do větších detailů. Z modelu se poté může vycházet a použít jej k vytváření reportů nebo generování C kódu a HDL kódu. Při aktualizacích modelu v pokročilejších stádiích vývoje lze modely zpětně testovat za pomoci in-the-loop testů pro jednotlivé úrovně popsané v kapitole 2.2.1.[1][5]

MBD se často používá v kombinaci s takzvaným V modelem vývoje, který je hojně využíván v automobilovém průmyslu. V model používá podobné kroky vývoje jako Vodopádový model. Vodopádový model má ovšem nevýhody v tom, že části vývoje následují po sobě a jsou striktně dané, tudíž změny v pokročilejších stádiích vývoje jsou problematické. V model se namísto procházení částmi vývoje lineárně v jednom směru se po implementační fázi vývoje obrací směrem vzpětí a každé části vývoje tak odpovídá konkrétní testovací fáze. [1]

Obrázek 2.4: V model s implementovanými in-the-loop testy [4]

MBD pomáhá v modelu v automatizaci vývoje. Tato automatizace zajistí, že je jednodušší provádět změny v pozdějších částech vývoje, například díky automatickému generování reportů. Je jednodušší operovat s komplexním systémem, jelikož je možno pozorovat jednotlivé interakce mezi komponenty systému. Možnost provádět in-the-loop testy podporuje inovaci, protože všechny nové nápady lze rychle otestovat. Modelování systému v úrovních umožňuje vyvíjet celý systém již na začátku, když ještě nemusí být známy všechny proměnné.[1]

2.2.1 Konfigurace in-the-loop testů

Model-in-the-loop

Při vývoji řídicí jednotky nebo celého systému metodou MBD se začíná vytvořením modelu na hostujícím PC v prostředí pro modelování a simulaci systému, jakým je například Simulink. Model zde především slouží k verifikaci použitých algoritmů. Interakce jednotlivých subsystémů jako soustavy a regulátoru jsou zajišťovány signály v daném prostředí. Testy a simulace nemusí probíhat v reálném čase, a tak lze simulovat s variabilním krokem a s rychlostí odpovídající použitému procesoru.[7]

Software-in-the-loop

V části SIL se vývoj provádí na hostujícím PC ve stejném prostředí jako u MIL. Z modelu se za pomoci kompilátoru vygeneruje C kód a simulace testuje chování kódu v simulaci s malým krokem. Testují se zde hlavně takzvané dead-point efekty, jako jsou saturace, diskretizace signálu nebo přetečení buferu.[6][7]

Hardware-in-the-loop

V části HIL se systém již rozdělí na Controller, tedy naprogramovanou elektronickou řídicí jednotku (ECU), která se bude používat ve virtuálním zařízení a Plant, tedy model chování řízené soustavy simulovaný s ním krokem v reálném čase na hostujícím PC. Signály si testovaná jednotka fyzicky předává s hostujícím PC za pomoci sériových komunikačních sběrnic, analogových nebo diskrétních digitálních signálů a ostatních metod podle cílového použití. Po úspěšném HIL testu lze řídicí jednotku přímo připojit k ovládanému fyzickému modelu.[7]

2.2.2 Varianty HIL testů

HIL testování přináší mnoho výhod oproti validaci. Pro validaci řídicí jednotky je zapotřebí využít fyzický systém a testovat tak zařízení v reálných podmínkách, což může být jak časově náročné a drahé tak i nebezpečné. HIL naopak využívá virtuální model ovládaného systému, kde lze simulovat vnější zásahy. Tento přístup umožňuje dlelat změny a úpravy projektu dříve a levněji. [8]

Z těchto důvodů se v dnešní době HIL testování hojně využívá v průmyslu automobilovém, leteckém, medicínském, automatizaci, transportu i ve výzkumu. Vznikají se mu firmy jako dSpace, MicroNova, National Instruments a další. Tyto firmy dodávají modulární testovací stavy, které je možné skládat do skříní (rack) a vytvářet tak od malých stavů testující jednu jednotku po velké testovací stavy simulující chování například celého auta. Modulární systémy pro HIL testování vyráběné firmou dSpace lze vidět na obrázku 2.5. Díky vývoji v oblasti HIL simulací je možné toto testování rozdělit na tři typy vyvinuté pro speciální potřeby a to Control HIL, Power HIL a Mechanical HIL. [9]

Obrázek 2.5: Modulární systémy dSpace Scalexio [9]

Control Hardware-in-the-loop

Control HIL (CHIL) nebo také Signal-level HIL je základní konfigurací HIL testu, kde jsou kontrolovány informační porty testovaného zařízení. Řídící jednotka si vyměňuje prostřednictvím vstupní a výstupní karty digitální a analogové signály s počítačem, na kterém běží simulace systému v reálném čase. Tímto způsobem HIL testování je možné například hodnotit regulaci v leteckém průmyslu, testovat elektronické řídicí jednotky v automobilovém průmyslu, vyvíjet nové komponenty, akční členy a další.[10]

Power Hardware-in-the-loop

S novými přísnějšími požadavky na připojení k elektrické síti ukládanými regulačními úřady a síťovými operátory vzniká i potřeba vývoje nových způsobů certifikací a testování k dokázání splnění daných požadavků. To nám umožňuje právě Power Hardware-in-the-loop (PHIL). PHIL testuje silovou elektroniku, tedy zařízení, které přijímá nebo generuje energii. Z tohoto důvodu je zapotřebí implementovat silové rozhraní. Toto silové rozhraní komunikuje za pomoci digitálních a analogových signálů s real time počítačem a následně zesiluje signály přijímané z počítače, které si dále posílá s testovaným zařízením. Rozdíl mezi zapojením Control a Power HIL lze vidět na obrázku 2.6.[10]

Výhody Power Hardware-in-the-loop plynou z kombinace simulace a laboratorních testů. Simulace umožňuje exibilitu v návrhu testu, kde urychluje čas s potřebným zapojením při klasickém laboratorním testování. Systém je možno testovat v extrémních podmínkách s minimálními náklady a nebezpečím, jelikož nemusíme pracovat s hýbajícími částmi, či velkými momenty setrvačnosti. Tímto testováním tedy lze odhalit skryté problémy a do hloubky porozumět systému. Mezi testované systémy nejčastěji patří baterie a měniče, které je možno testovat ještě před zhotovením motoru. Dále lze testovat různé generátory, malé elektrické sítě, či celé elektro auta.[10][11]

Obrázek 2.6: Schéma simulace CHIL a PHIL

Mechanical Hardware-in-the-loop

V některých Hardware-in-the-loop testech nestačí pouze signálová komunikace s řídicí jednotkou, a je zapotřebí simulovat zatížení přímo na elektrických pohonech, fyzicky stimulovat senzory nebo testovat HMI (Human Machine Interface). V těchto případech přichází na řadu takzvaný Mechanical Hardware-in-the-loop (MHIL).[12]

Součástí MHIL simulace tedy může být například kompletní testovaná pohonná jednotka, na níž je vyvolávána zátěž pomocí dynamometru připojeného k real time počítači (viz. 2.7). MHIL najde uplatnění hlavně v testování posilovačů řízení, elektrických řepadel a hydrauliky, posilovače brzd, zpětné vazby pro simulátor řízení auta a mnoho dalších. Výhodou MHIL je také možnost testování mechaniky velkých systémů v laboratoři. U velkých systémů jako jsou třeba jeřáby by bylo testování na národním produktu drahé a v některých případech i nebezpečné.[12][11]

Obrázek 2.7: Schéma Mechanical Hardware-in-the-loop simulace[11]

2.3 Návrh testů

Hlavním úkolem při návrhu efektivního testu je si řádně promyslet, co testovat a kolik toho testovat. Toho lze dosáhnout využitím Markovových řetězců a stochastického generátoru k vybírání stavů přechodů v testovaných stavových automatech. Modely s Markovovými řetězcí jsou sestaveny k popisu očekávaného chování systému a následnému generování testů podle stavů jednotlivých přechodů. Takový model lze vidět na obrázku 2.8. Vzorový systém začíná ve stavu On Hook a končí po dosažení stavu Exit. Jeden test tedy prochází stavy, dokud nedojde do stavu konečného. Přechody mezi stavy jsou popsány stavy k jejich vykonání. Tyto stavy je možné určovat analýzou využití modelu, například za pomoci času stráveného v jednotlivých stavech, celkových stavů provedení jednotlivých událostí nebo stavů provedení událostí v jednom testu.[13]

Obrázek 2.8: Model telefonu s využitím Markovových řetězců.[13]

K generování testů takového modelu lze postupovat různými postupy.[13]

- ^ Testy lze generovat náhodnými průchody grafu, které začínají v počátečním stavu a končí v koncovém stavu. Přechody v testu mohou být ovlivněny nastavenými parametry jednotlivých přechodů. To umožňuje generování nekonečného počtu možných testů.
- ^ Test může být vygenerován s úmyslem projít všemi možnými stavy a přechody s minimálním počtem kroků (minimum coverage test). K generování takového testu se využívá takzvaný postman algoritmus.
- ^ Testy mohou být generovány podle váhy přechodů, tedy testy se budou provádět na nejvíce reprezentativních vzorcích chování testovaného systému.
- ^ Testovací inženýři mohou generovat testy ručně a přímo tak otestovat partikulární chování systému.

3 Praktická část

Hlavním cílem bakalářské práce je vytvoření komplexního HIL testu univerzální řídicí jednotky Slave, který bude možné použít při dalším vývoji a aktualizacích FW. Jednotka neslouží k jednomu účelu, ale je univerzální. Lze ji programovat za pomoci aplikace k ní navržené, v níž je možné vytvářet různé kombinace funkcí a přechodů. Kapitola 3.1 se věnuje zprovoznění HW testu a HIL testu motoru dříve provedených v laboratoři a následným úpravám k dalšímu postupu. V kapitole 3.2.1 je popsána struktura a návrh testu řídicí jednotky. Tento test kontroluje všechny nastavené funkce a vygenerované přechody popsané v 3.2.2. Po případné aktualizaci FW jednotky, lze test doplnit o přidání funkcí a podmínky podle návodu v 3.3.

3.1 Úpravy dřívejších experimentů

3.1.1 Hardware test

Kontaktní hardware test slouží ke kontrole plošného spoje a správného osazení testované řídicí jednotky. Test generuje digitální a analogové signály ve skriptu v Matlabu. Ty jsou dále posílány pomocí karty MF 624 na vstupy testované řídicí jednotky. Do této jednotky je skriptem přes programátor Pickit nahrán testovací firmware, který zpracovává vstupní piny jednotky a následně naměřené signály nastavuje na výstupní piny jednotky. Tyto výstupní piny jsou měřeny opět kartou MF 624. Naměřené hodnoty jsou v počítači porovnány s poslanými signály za pomoci směrodatné odchylky. Vypočtené směrodatné odchylky jsou porovnány s povolenými hodnotami. Pokud směrodatné odchylky vyhovují, testovaná jednotka je vyhodnocena jako funkční a je na ni nahrán Bootloader. V případě, že směrodatná odchylka nevyhovuje, je daný pin označený jako vadný. Tento test běží na frekvenci 1 kHz a schéma jeho principu lze vidět na obrázku 3.1.[2]

Obrázek 3.1: Schéma kontaktního hardware testu

Po provedení testu v mechatronické laboratoři byla testem zaznamenána chyba PWM a digitálního pinu BRK. Po kontrole kontaktů na přípravku, měřicí karty, proměnění parametrů RC filtru a změnění posílaného signálu na osciloskopu bylo zjištěno, že PWM posílaná z jednotky měla frekvenci 100 Hz oproti požadované frekvenci 20 kHz. Z tohoto důvodu nebyl signál vyhovován a neodpovídal tak očekávaným hodnotám. Pro opravu chyby PWM bylo tedy zapotřebí upravit konfiguraci frekvence PWM ve zdrojovém kódu testovacího hardwaru. Po prostudování C kódu byla také zjištěna příčina chyby digitálního pinu BRK. Ta byla způsobena tím, že firmware nepošílal signál z pinu LS na pin BRK. Tato chyba byla rovněž opravena.

3.1.2 HIL test DC motoru

HIL test DC motoru byl naprogramován v Matlabu jako součást Diplomové práce Ing. Jiřiny Zouhary. Parametry pro simulaci byly odhadnuty podle DC motoru Double Drive používaných v mechatronické laboratoři.

Obrázek 3.2: Schéma DC motoru[2]

Chování DC motoru je simulováno za pomoci rovnic 3.1 a 3.2.

$$U(t) = R i(t) + L \frac{di(t)}{dt} + c \omega(t) \quad (3.1)$$

$$J \frac{d\omega(t)}{dt} = c i(t) - b_{vis} \omega(t) - b_{such} \operatorname{sgn}(\omega(t)) \quad (3.2)$$

Funkce sgn použitá v rovnici je nespojitá v počátku soustavy souřadnic a tím zvyšuje náročnost výpočtu. Pro zjednodušení výpočtu byla tato funkce nahrazena spojitou funkcí 3.3.[2]

$$\operatorname{sgn}(\omega(t)) = \frac{2}{1 + e^{-20\omega(t)}} - 1 \quad (3.3)$$

Po spuštění kódu pro HIL test DC motoru dojde k inicializaci karty MF 624, nastavení parametrů simulace a motoru. Následuje spuštění for smyčky, v níž běží časovač a simulace chování motoru. časovač v kódu čekáním zajišťuje, že smyčka trvá daný časový krok. Ten uloží pro výpočet diferenciálních rovnic motoru. Pokud se smyčka nestihne provést a časovač tedy nemusí čekat, tak ukládá delší časové kroky. Ty poté mohou způsobit neočekávané chování v simulaci, které lze vidět v obrázku 3.3. Tímto problémem je pro funkční test zapotřebí zamezit. P-vodní Hardware-in-the-loop test DC motoru běžel na frekvenci 1 kHz s proměkanými časovými kroky v řádu až jednotek procent tik celkových. Tento problém byl vyřešen odstraněním pro běh testu nepotřebného GUI. Tím bylo dosaženo zmenšení počtu proměkaných tik na desetiny až setiny procent tik celkových. Pro správný běh testu je také zapotřebí spustit skript s HIL testem dříve, než přejde testovaná jednotka do stavu jedna, jelikož většina proměkaných tik se stává právě na začátku testu.

Obrázek 3.3: Graf s vynechaným výpočtem

Test provedený využíval pro výpočet diferenciálních rovnic motoru Eulerovu numerickou metodu. Pro přesnější výpočet je výhodné použít metodu Runge-Kutta, která je popsána rovnicemi 3.4.[14]

$$\begin{aligned}
 x_k &= x_{k-1} + \frac{h}{6}(a_1 + 2a_2 + 2a_3 + a_4); \\
 a_1 &= f(t_{k-1}; x_{k-1}); \\
 a_2 &= f\left(t_{k-1} + \frac{h}{2}; x_{k-1} + \frac{h}{2}a_1\right); \\
 a_3 &= f\left(t_{k-1} + \frac{h}{2}; x_{k-1} + \frac{h}{2}a_2\right); \\
 a_4 &= f(t_k; x_{k-1} + ha_3)
 \end{aligned} \tag{3.4}$$

Kde a_i jsou odhady derivované proměnné, x_k je derivovaná proměnná, h je časový krok a t_k čas. Díky použití metody Runge Kutta bylo možné pozorovat větší přesnost ve výpočtech a rychlejší regulaci na požadovanou hodnotu.

Dalším problémem, na který lze narazit při HIL testu motoru je nastavení příliš jemného rozlišení enkodéru zajišťujícího předání zpětné vazby s informací o změně natožení. Jelikož test běží na zvolené frekvenci, je možné posílat logické signály na kanály enkodéru právě na této frekvenci. Maximální úhlová rychlost motoru by tedy odpovídala úhlu jednoho tiků enkodéru za čas periody, tedy časového kroku. Maximální úhlovou rychlost, kterou může simulace enkodéru poslat lze tedy určit dle 3.5. Při testování jednotky bylo experimentováno se snižováním frekvence, na níž běží test pro další zmenšení počtu proměřených tiků. Právě enkodér byl limitujícím faktorem pro frekvenci testu, jelikož při frekvenci 100 Hz docházelo k porušení správnosti signálu enkodéru a následným problémům s regulací.

$$\omega_{\max} = \frac{2}{n_{\text{tiků}} T_s} \quad (3.5)$$

Kde ω_{\max} je maximální úhlová rychlost, která lze zaznamenat na enkodéru, $n_{\text{tiků}}$ je počet tiků enkodéru na jednu otáčku a T_s je časový krok simulace.

3.1.3 Homing test

V původním kódu již byla naprogramována simulace chování koncových spínačů, ale spínala koncové snímače jen v jedné konkrétní nastavené poloze. Pro simulaci navádění (Homing) je však předpokládáno, že se motor bude točit, dokud nenajde polohu snímače. Při této popsané simulaci by již najít nemusel. Z tohoto důvodu bylo chování koncových snímačů reprogramováno za pomoci funkce modulo tak, aby se spínaly v daném úhlu natožení při každé otáčce a to s hysterezí [10].

Při testování navádění byla objevena nestabilita. Pokud je navádění spuštěno, když je motor na poloze koncového spínače, dojde k rozkmitání okolo požadované hodnoty. Po nějaké době kmitání provede motor celou otáčku a úspěšně se ustálí na koncovém spínači. Tuto událost lze vidět v grafu 3.4.

Obrázek 3.4: Nestabilní homing

3.2 Testování jednotky

3.2.1 Návrh funkčního testu

První myšlenkou pro vytvoření kompletního testu jednotky bylo vytvořit program v DINO aplikaci, který by ve stavech obsahoval jak testované funkce, tak i kontrolu digitálních vstupů. Kontrola digitálních vstupů vytváří události, které lze použít k vytvoření přechodů. Tyto události by za pomoci logických funkcí AND a OR vytvářely dané zánce k vykonání přechodů. V Matlabu by se generovaly náhodné signály zajišťující ovládní těchto přechodů. Problém s tímto přístupem byl v tom, že jedna řídicí jednotka slave může být naprogramována pouze 16 událostí, které by byly pro tento způsob testování nedostatečné. Dalším problémem byla nepřehlednost v grafickém prostředí při větším počtu přechodů a celkově pomalé programování těchto přechodů.

Z těchto důvodů bylo rozhodnuto, že se testované stavy naprogramují v DINO aplikaci. V tomto programu je zapotřebí mít v každém stavu nejméně jednu funkci, která zajistí vytvoření události. Tyto události budou následně použity k vytvoření přechodů mezi stavy skriptem v Matlabu. Dále je zapotřebí do prvního stavu naprogramovat nastavení

digitálního pinu 1 na logickou hodnotu 1. Tuto informaci využívá test ke srovnání čas-začátku stavového automatu na jednotce a v simulaci. Následně do aplikací vygenerovaných tabulek budou doprogramované přechody. K naprogramování přechodů slouží skript `TransMakerV2.m` programovaný v Matlabu. Ten vytvoří náhodnou permutaci přechodů mezi stavy, která zajistí postupný přechod všemi naprogramovanými stavy. Skript dále vytvoří tabulky přechodů, do nich doplní vytvořenou permutaci a vytvoří podmínky pospojování naprogramovaných událostí s logickým operátorem AND. Nakonec doplní ostatní potřebné informace do tabulek, jako je počet přechodů, nebo který přechod odpovídá jakému stavu a výslednou tabulku uloží jako `Ready2Upload.mat` pro HIL test a `Ready2Upload.DINOBuild` pro testovanou řídicí jednotku. Po vytvoření tabulek se provozní komunikace s hlavní jednotkou master za použití objektu `D2xx_MF.m`. Do hlavní jednotky se tabulky nahrají pomocí objektu `Uploader.m`. Ke správnému fungování této funkce je zapotřebí mít ve stejné složce složky Dino aplikace `mui` a `AppData`. Jakmile je testovací program doplněný o přechody a nahraný v testované jednotce lze přejít k HIL testu.

HIL test jednotky se spustí v Matlabu skriptem `HIL_test_V12.m`, jeho fungování je znázorněno schématem 3.5. Po spuštění kódu dojde k načtení tabulek s programem jednotky. Hodnoty v těchto tabulkách jsou 8 bitové a rozdělené na High/Low bit. Program tabulky následně zpracuje do 16 bitových hodnot a vypíše dleřité hodnoty pro testování do proměnných typu `struct`. Poté dochází k inicializaci Humusoft I/O karty MF 624 a načtení parametrů simulace a parametrů simulovaného motoru.

Dále dochází ke vstupu do časované smyčky. Po vstupu do smyčky dochází k přeřtení všech měřených signálů z testované jednotky. Následuje řešení diferenciálních rovnic simulovaného DC motoru popsaných v 3.1.2, simulace spínání koncových spínačů 3.1.3 a převod vypořítané hodnoty natoření na signály posílané do enkodéru. Ve smyčce běží stavový automat `StateMachineV4.m`, který kontroluje, zda dořlo k přechodu mezi stavy. K této kontrole využívá naměřené hodnoty, které dosazuje do podmínek stavů zjiřtěných z tabulek programu jednotky. Plněním těchto podmínek se inkrementuje proměnná, která se nakonec porovná s počtem možných událostí v daném stavu. Pokud se počet možných událostí rovná této proměnné je splněna podmínka přechodu a uloří se čas vstupu a výstupu ze stavu a říslo stavu. Po dokonření výpořtů a simulací dochází k poslání signálů testované jednotce. Smyčku zakončuje časování popsané v 3.1.2.

Jakmile je měření dokonřeno přichází na řadu kontrola naměřených hodnot. Pro každý stav se urří ořekávané hodnoty ze zpracovaných tabulek ve funkci `StateChecker.m`. Ty se následně porovnávají ve funkci `MSE_check.m` s hodnotami naměřenými v řase určeném stavovým automatem v časované smyčce. Porovnání hodnot se provádí za pomoci Root Mean Square (RMS) tedy smřrodatné odchylky. Vypořtené smřrodatné odchylky se zkontrolují. Podle výsledků této kontroly program vytvoří tabulku s informacemi o probřhlých a bezchybných stavech. Nakonec program vykreslí naměřené a dolořtené hodnoty a vypíše procento proměřkaných kroků.

Obrázek 3.5: Schéma HIL testu

Návod

K provedení HIL testu řídicí jednotky slave je nejprve zapotřebí v DINO aplikaci v sekci Hardware definovat testovanou řídicí jednotku, zadat její ID číslo a vybrat konfiguraci jednotky pro test. Dále uživatel vytvoří stavy s testovanými funkcemi a minimálně jednou událostí, například stopky. Další podmínkou je, aby první stav nastavil digitální výstup 1 na logickou hodnotu 1. Test používá tuto informaci ke srovnání čas- stavových automatů. Program se sestaví a uloží jako `NoTransitions.Dinobuild`. V Matlabu se spustí skript `TransitionMaker.m`. Ten vytvoří potřebné přechody k testu a nahraje program do testované řídicí jednotky. Následuje spuštění HIL testu. Po dokončení HIL testu uživatel dostane tabulku s informacemi o úspěšnosti jednotlivých stavů a grafy naměřených a dopočítaných hodnot pro vizuální kontrolu.

3.2.2 Analýza testů

Na jednotce byly testovány především ty funkce, které byly detailněji popsány a bylo tedy u nich známé chování, a ty, které byly měřitelné na pinech přípravku. Mezi tyto funkce patří:

- ^ Kontrola digitálních vstupů
- ^ Nastavení digitálních výstupů
- ^ řízení napětí
- ^ Zpětnovazební regulace
- ^ Porovnání analogového vstupu s hodnotou
- ^ Vypnutí výkonové elektroniky při detekci fault pinu

Kontrola digitálních vstupů vytváří událost při naměření zadané logické hodnoty na zadaném pinu. Nastavení digitálních výstupů nastaví na zadaný digitální výstupní pin řídicí jednotky zadanou logickou hodnotu. Ve funkci řízení napětí si lze zvolit ovládání tří

výstupních pinů, a to analogový pro ovládání motoru, obecný analogový a PWM. Na pinu lze nastavit konstantní hodnotu nebo proměnné napětí podle trajektorie nebo senzoru. Funkce zpětnovazebná regulace může regulovat natožení motoru na zadanou konstantní hodnotu, na vytvořenou trajektorii nebo podle hodnot ze senzoru. Zpětná vazba je zajištěna digitálním enkodérem. Po skončení trajektorie může vytvořit událost. Porovnání analogového vstupu vytváří událost, pokud daný analogový vstup splňuje zadanou rovnost nebo nerovnost. Ve funkci vypnutí výkonové elektroniky je kontrolována logická hodnota na zadaném digitálním vstupu. Pokud je na vstupu zadaná hodnota, dojde k nastavení nulového napětí na pinu PWM, na všech analogových pinech a k vytvoření události.

Vyhodnocení testu

Pro kontrolu elektronické řídicí jednotky byl spuštěn HIL test s 5 stavy obsahující výše popsané funkce. K procházení těchto stavů byla vytvořena permutace určující pořadí přechodů. Přechodové stavy, které poslouží k orientaci v grafech byly zaznačeny do tabulky:

<i>f</i> as vstupu [ms]	<i>f</i> as výstupu [ms]	Stav
4280	6280	1
6281	7281	4
7282	13285	2
13286	19289	3
19290	21290	5
21291	23291	1
23292	24292	4
24293	25000	2

Obrázek 3.6: Naměřené hodnoty na obecném analogovém výstupu a digitálních výstupech

Naměřené signály a dopořtené hodnoty byly vyneseny do grafů 3.6, 3.7 a 3.8. Funkce nastavení digitálních pinů a řízení napětí lze pozorovat v grafu 3.6. Ve vrchní části obrázku je možno vidět řízení obecného analogového napětí na konstantní polohu ve stavu 4 a následné řízení podle vytvořené trajektorie ve stavu 2. Ve spodní části grafu se v prvním stavu nastaví digitální vstup 1 pro signalizování začátku testu. Dále lze vidět nastavení digitálního pinu 2 ve stavu 2.

Obrázek 3.7: Naměřené a simulované fyzikální veličiny

V grafu 3.7 můžeme vidět chování napětí, logických pinů motoru, natožení, úhlové rychlosti a proudu pro regulaci DC motoru na polohu. Ve stavu 1 je regulováno natožení na konstantní hodnotu 60° . Ve stavu číslo 3 v čase 13 a 19 s je natožení regulováno dle vytvořené trajektorie. Jednotka využívá k regulaci proporcionální a integrační složku. V čase 19 a 21 s dochází k vypnutí výkonové elektroniky po sepnutí fault pinu ve stavu 5.

Obrázek 3.8: Vypočtené natožení motoru spínající koncové spínače

Chování koncových spínačů je ukázáno v grafu 3.8. Ve vrchním grafu lze pozorovat natožení motoru v rozsahu 0 a 360 , které určuje logické hodnoty koncových spínačů. Tyto hodnoty jsou vykresleny v grafech pod grafem natožení.

Naměřené hodnoty byly porovnány s očekávanými za pomoci směrodatné odchylky. Ta byla nadále použita k vyhodnocení testu. Po vyhodnocení testu dojde k vypsání výsledků:

Results =

5x3 table

StateNumber	No_Runs	Flawless_Runs
1	2	2
2	2	2
3	1	1
4	2	2
5	1	1

Missed Ticks 0.068 [%]

V případě nesplnění zadaných kontrolovaných podmínek se vypíše, v kterém stavu co nefungovalo. Toto nesplnění se projeví i v tabulce Results. Zde se testovaná jednotka zpozdila a regulovaná trajektorie ve stavu číslo 3 tedy neodpovídala očekávané.

REG error 3

Results =

5x3 table

StateNumber	No_Runs	Flawless_Runs
1	2	2
2	2	2
3	1	0
4	2	2
5	1	1

Missed Ticks 0.044 [%]

3.2.3 Nalezené chyby

Při testování jednotky bylo zjištěno několik chyb. Zajímavou chybou je zpoždění jednotky v prvním stavu o 1 s. Tato chyba byla upozorována zprvu díky HIL testu a následně přeměřena osciloskopem. Její původ je neznámý a stává se přibližně tak v jednom ze čtyř měření. Jejím důsledkem je následné rozhození celého testu a očekávané hodnoty neodpovídají hodnotám naměřeným. To lze vidět v grafu 3.9.

Obrázek 3.9: Očekávané a naměřené natožení na opořené jednotce

Další problém byl se zpětnovazebnou regulací na úhlovou rychlost. Nastavení, zda jednotka bude regulovat na polohu nebo na rychlost, se provádí v textovém souboru, v němž se konfiguruje parametry jednotky slave. Jednotka však reguluje na polohu nezávisle na konfiguraci regulace. Podobný problém s konfigurací nastává u enable pinu, který se nepřepíná při zadané regulaci, řízení DC motoru a to nezávisle na zvolené konfiguraci. To by mohlo znemožnit provoz motoru, který tento signál k provozu potřebuje. Regulovat natožení nebo řídit napětí je možné na zadanou hodnotu a podle vytvořené trajektorie. Dále by jednotka slave měla být schopná regulovat a řídit na hodnoty přijaté ze senzoru. Tato funkce nefunguje. Po projití stavu s touto funkcí dojde k vypnutí výkonové elektroniky v dalších stavech. Dále testované funkce byly vytvořeny událostí při dosažení koncového spínače a při dosažení polohy. U těchto funkcí chybí doplněný návod k použití. Tyto funkce rovněž nevytvářejí událost při splnění zadané podmínky.

3.3 Rozšíření testu

Program k Hardware-in-the-loop testu byl navrhnout s myšlenkou jednoduché aktualizace, pro případ přidání nové funkcionality řídicí jednotky. Funkcionality řídicí jednotky může být rozšířena přidáním nových funkcí nebo vytvářením nových událostí používaných k zajištění přechodů mezi stavy. Tato aktualizace HIL testu je možné provést jednoduchou úpravou HIL testem používaných funkcí v Matlabu.

3.3.1 Přidání funkce

Pro případ přidání nové funkce do firmwaru jednotky je nutné vytvořit novou if podmínku ve funkci StateChecker.m. Tato podmínka bude hledat číslo, pod kterým je daná funkce

uložená v naprogramovaných tabulkách. Poté je zapotřebí uložit z programové tabulky data popisující chování funkce, jako například číslo trajektorie, pořadovaná hodnota, číslo pinu a další podobné informace, do proměnné `What2Check`.

Ukázkový kód z funkce `StateChecker.m` pro kontrolu napřevého řízení:

```
if ~isempty(find(States_Functions(State).state(:,4)==3)) %AO check
    position = find(States_Functions(State).state(:,4)==3);
    What2check(2).expected(:,:) = ...
    States_Functions(State).state(position(end),:);
else
    What2check(2).expected=[];
end %if
```

Ve funkci `MSE_check.m` je dále zapotřebí přidat nový case s číslem kontrované funkce z naprogramovaných tabulek do podmínky `switch`. Ve vytvořeném case je nutné naprogramovat výpočet směrodatné odchylky z naměřených a očekávaných hodnot a porovnat výsledek s povolenou hodnotou. K naprogramování tohoto výpočtu a následné kontroly stačí zkopírovat a doplnit kód z kontroly jiné funkce.

3.3.2 Přidání události

Při přidání události je zapotřebí vytvořit ve funkci `GetEvents.m` nový case s číslem funkce vytvářející událost z naprogramovaných tabulek do `switch` podmínky. V tomto case je nutné uložit potřebné informace o události z proměnné `Tables` do proměnné `StateEvents`.

Ukázka kódu z funkce `GetEvents.m` pro událost z porovnávání napřeví se zadanou hodnotou:

```
case 5
    StateEvents(Event_mat(i,2)).state(k,3) = ...
    States_Functions(Event_mat(i,2)).state(Event_mat(i,3),5);
    StateEvents(Event_mat(i,2)).state(k,4) = ...
    States_Functions(Event_mat(i,2)).state(Event_mat(i,3),6);
    StateEvents(Event_mat(i,2)).state(k,5) = ...
    States_Functions(Event_mat(i,2)).state(Event_mat(i,3),7);
```

Ve funkci `StateMachineV4.m` je potebné přidat `elseif` podmínku kontrolující splnění podmínky pro přechod. Pokud je podmínka splněna dojde k inkrementaci proměnné `a`, která je dále použita ke kontrole logické funkce `AND` mezi dvěma podmínkami ve stavu.

Ukázka podmínky z funkce StateMachineV4.m pro dokončení trajektorie v např•ovém ovládání:

```
elseif StateEvents(state).state(i,2) == 3 && ...  
(InterPol(StateEvents(state).state(i,3)).Time(end)/Ts) < t  
    a = a+1;
```

4 Závěr

Cílem této práce bylo vytvoření kompletního Hardware-in-the-loop testu univerzální elektronické řídicí jednotky Dino slave a její otestování.

V rešeršní části byl v kapitole 2.1 popsán využívaný testovací přístroj, na němž je testována elektronická řídicí jednotka. Kapitola 2.2 se zabývá metodou vývoje Model based design a s ní spojené varianty in-the-loop test. Dále pojednává o možných konfiguracích HIL testu využívaných v praxi. V kapitole 2.3 je pozornost upřena na návrh a metody HIL testování pro stavové automaty. Při testování stavových automatů se využívá modelování s pomocí Markovových řetězců. Následné metody testování používají různé šance přechodů mezi stavy k vytvoření efektivních testů.

Praktická část práce se nejprve v detailu zprovozní test řídicí elektronické jednotky slave, jež byly dříve prováděny v mechatronické laboratoři. Prvním z těchto testů je kontaktní testování jednotky takzvané In Circuit Testing (ICT). ICT se používá k odhalení možných chyb na DPS, jako jsou chybějící součástky nebo špatné kontakty. Ke správné funkci ICT testu jednotky bylo zapotřebí v programovacím jazyku C lehce upravit testovací firmware. Tento test je popsán v kapitole 3.1.1. Druhým zprovozněným testem je HIL test regulace DC motoru ověřující schopnost testované řídicí jednotky regulovat jeho natočení. Pro tento test byl zvolen vhodný časový krok 0,01 s a řešení diferenciální rovnice elektromotoru typu Runge-Kutta. To zajistilo zmenšení množství proměškaných tiků, které jsou pro použitelný průběh testu klíčové. S řešením typu Runge-Kutta šlo pozorovat i zrychlení v regulaci na požadovanou hodnotu oproti převodnímu Eulerovu typu řešení.

Navrhovaný kompletní HIL test má ověřit chování podřízené elektronické řídicí jednotky slave. Toto chování jednotky je programováno za pomoci Dino aplikace popsané v 2.1.3. V aplikaci je možné graficky sestavovat stavové automaty s přechody a funkcemi jako je řízení napětí nebo regulace natočení motoru. V této aplikaci je zapotřebí vytvořit stavy, které obsahují kontrolované funkce. Po sestrojení programu je zapotřebí vytvořit kód doplnit o přechody mezi stavy. Přechody se doplní za pomoci skriptu. Ten vytvoří náhodnou permutaci, která určí pořadí procházení všech kontrolovaných stavů a nahraje doplněný kód do hlavní řídicí jednotky komunikující s jednotkou podřízenou.

Samotný test se provádí na tením tabulek s daty k programu v jednotce, následným zpracováním těchto dat a spuštěním časovaného smyčky. V této smyčce se měří signály z jednotky, simuluje se chování DC motoru, enkodéru a koncových snímačů. Ve smyčce také běží stavový automat, který určuje přechody jednotlivých stavů. Nakonec se posílají zpětnovazební signály ze simulovaných senzorů. Po dobytí smyčky jsou naměřené hodnoty porovnané s hodnotami očekávanými prostřednictvím směřované odchylky. Vypočtené směřované odchylky jsou využity ke zhodnocení testu. Nalezené chyby při testování jednotky jsou popsány v kapitole 3.2.3.

Test byl navržen s možností jeho aktualizace v případě přidání nových funkcí jednotky. Jako návod pro tuto aktualizaci slouží kapitola 3.3.

4.1 Možnosti dalšího vývoje

Na zhotovenou práci lze navázat vytvořením programu, v němž by se přímo generovaly stavy s kontrolovanými funkcemi a událostmi. Tyto stavy by se po doplnění o přechody nahrály do testované jednotky. To by ještě více zautomatizovalo proces testování, jelikož nyní uživatel, který testuje jednotku, musí pořád vytvářet stavy s kontrolovanými funkcemi v Dino aplikaci.

Další možností vývoje je rozšíření testovacího přípravku o silové rozhraní. Toto rozhraní by umožnilo testování výkonových jednotek Electromen využívaných k řízení pohonů a dodávaných s jednotkami Dino. [3]

Literatura

- [1] AARENSTRUP, Roger. *Managing Model-Based Design* [online]. Natick: The MathWorks, 2015 [cit. 2022-05-17]. ISBN 1512036137. Dostupné z: https://ch.mathworks.com/content/dam/mathworks/ebook/gated/MBD_Book_PDF_Version.pdf
- [2] ZOUHAR, Štěpán. *HIL testovací stav pro soustavu univerzálních elektronických řídicích jednotek*[online]. Brno, 2019. [cit. 2022-05-16]. Dostupné z: <https://www.vutbr.cz/studenti/zav-prace/detail/116768>. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechanické, mechatroniky a biomechaniky. Vedoucí práce Ing. Martin Brabc.
- [3] NAJMAN, Jan. *Dokumentace řídicího systému DINO*. Brno: Mechatronická laboratoř, 2019
- [4] REYES, Victor. *Virtual Hardware "In-the-Loop": Earlier Testing for Automotive Applications* [online]. Mountain View: Synopsys, 2013 [cit. 2022-05-17]. Dostupné z: https://www.synopsys.com/cgi-bin/proto/pdfdata/docsdl/virtual_hardware_wp.pdf?file=virtual_hardware_wp.pdf/
- [5] HERNITER, Marc, Zachariah CHAMBERS. *Introduction to Model-Based System Design* (Studijní materiály k VŠ kurzu) [online]. [cit. 2022-05-17]. Dostupné z: <https://www.mathworks.com/academia/courseware/intro-to-model-based-system-design.html>
- [6] Simulating and Testing TargetLink Code.dSPACE [online]. [cit. 2022-05-06]. Dostupné z: <https://www.dspace.com/en/enc/home/medienvideos/productvideos/video-tl-simulation.cfm>
- [7] NIBERT, Jonathan, Marc HERNITER a Zachariah CHAMBERS. Model-Based System Design for MIL, SIL, and HIL. *World Electric Vehicle Journal* [online]. 2012, 5(4), 1121-1130 [cit. 2022-05-17]. Dostupné z: <https://www.mdpi.com/2032-6653/5/4/1121>
- [8] Basics of Hardware-In-The-Loop simulation. *MathWorks* [online]. Natick: MathWorks, c2022 [cit. 2022-05-17]. Dostupné z: <https://www.mathworks.com/help/physmod/simscape/ug/what-is-hardware-in-the-loop-simulation.html>
- [9] SCALEXIO. *DSPACE* [online]. Paderborn: dSPACE, c2022 [cit. 2022-05-17]. Dostupné z: https://www.dspace.com/en/pub/home/products/hw/simulator_hardware/scalexio.cfm
- [10] JONG, Erik de et al. "European White Book on Real-Time Power Hardware in the Loop Testing" (2012).

- [11] MICHNA, Michał, Filip KUTT, Łukasz SIENKIEWICZ, Roland RYNDZIONEK a Grzegorz KOSTRO. Mechanical-Level Hardware-In-The-Loop and Simulation in Validation Testing of Prototype Tower Crane Drives. *Energies* [online]. 2020, 13(21) [cit. 2022-05-17]. Dostupné z: <https://doi.org/10.3390/en13215727>
- [12] Implementing Mechatronic Test Benches. *DSPACE* [online]. Paderborn: dSPACE, c2022 [cit. 2022-05-17]. Dostupné z: https://www.dspace.com/en/pub/home/products/systems/test_bench.cfm#179_15129
- [13] PROWELL, Stacy. Using Markov Chain Usage Models to Test Complex Systems. In: *Proceedings of the 38th Annual Hawaii International Conference on System Sciences* [online]. Big Island: IEEE, 2005, 318c-324c [cit. 2022-05-17]. ISBN 0-7695-2268-8. ISSN 1530-1605. Dostupné z: <https://dx.doi.org/10.1109/HICSS.2005.663>
- [14] ZELTKEVIC, Michael. Runge-Kutta Methods. *Massachusetts Institute of Technology* [online]. Cambridge (Massachusetts): MIT, 1998 [cit. 2022-05-19]. Dostupné z: https://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node5.html

Seznam příloh

1. Matlab skript pro HIL test řídicí jednotky
2. Matlab skript pro vytváření přechodů mezi stavy