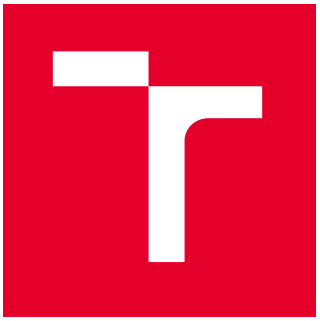


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

HIL TESTOVANÍ BATERIOVÉHO SYSTÉMU STUDENTSKÉ FORMULE

HIL TESTING OF STUDENT FORMULA BATTERY MANAGEMENT SYSTEM

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Matouš Přikryl

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Roman Adámek

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Bc. Matouš Přikryl**
Studijní program:	Mechatronika
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. Roman Adámek**
Akademický rok:	2022/23

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 diplomové práce:

HIL testovaní bateriového systému studentské formule

Stručná charakteristika problematiky úkolu:

Hardware in the loop (HIL) testování se využívá s úspěchem již řadu let v celé řadě odvětví v čele s leteckým a automobilním průmyslem pro testování řídicích systémů. Klíčovým přínosem této technologie je možnost testovat řídicí systémy bez nutnosti mít řízenou soustavu fyzicky k dispozici a dále možnost testovat řídicí systémy v celé řadě provozních podmínek. Těchto vlastností je možno s výhodou využít například u bateriového systému elektromobilu, kdy je možné řídicí elektroniku a její firmware otestovat ještě před připojením k baterii a tím předejít případným problémům nebo nebezpečným situacím. Zároveň je možno simulovat i kritické stavy, které mohou při provozu nebo nabíjení nastat a jejichž výskyt by v reálu mohl znamenat riziko poškození nebo zničení baterie.

Cíle diplomové práce:

1. Proveďte rešerši v oblasti HIL testování BMS (battery management system) vozidel z hlediska použitého HW, SW, metod a také modelů používaných pro vytvoření simulovaných částí systému.
2. Na základě rešerše z předchozího kroku a s ohledem na dostupný HW pro HIL testování, rozvrhněte, které části BMS studentské formule budou simulovány, propojení jednotlivých částí a jaké signály budou simulovány.
3. Navrhněte několik typů testů, které prověří správnou funkčnost řídicích jednotek. Simulujte běžné provozní stavy, nabíjení, vybíjení baterií při různých režimech zátěže a chybové stavy samotné baterie a připojených systémů.
4. Porovnejte chování simulovaných částí použitých pro HIL test s chováním reálných komponent použitých na studentské formuli pro validaci HIL testů.

Seznam doporučené literatury:

JOSHI, Adit. 2020. Automotive Applications of Hardware-in-the-loop (HIL) Simulation. SAE International.

PLETT, Gregory L. 2015. Battery Management Systems, Volume I: Battery Modeling. Artech House.

DAS, Shuvra. 2020. Modeling and Simulation of Mechatronic Systems using Simscape. Morgan & Claypool Publishers.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2022/23

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

Abstrakt

Práce se zabývá hardware-in-the-loop testováním bateriového systému studentské formule. Na začátku práce je provedena rešerše testování bateriových systémů. Zaměření je kladeno na hardware a software, který k tomuto účelu můžeme využít. V druhé části práce je popsáno rozvržení připojení BMS studentské formule k testovací soustavě a její následné testování. Využito bylo vlastních modelů částí formule a kód napsaný specificky pro účely této práce. Závěrečná část práce popisuje vyhodnocení testů a validaci modelů simulovaných částí.

Summary

The thesis covers the topic of hardware-in-loop testing of the battery management system of student formula. The beginning of the thesis contains a literature review on the topic of battery management systems testing with an emphasis on the hardware and software that can be used for this purpose. In the second part of the thesis, the layout of the BMS connection to the test rig and its subsequent testing is described. Custom models of the formula parts and code written specifically for this thesis were used for the testing of the BMS. The final part contains test analysis and validation of the models of the simulated parts.

Klíčová slova

Hardware-in-the-loop, HIL, Bateriový systém, BMS, Testování, Formule Student, Validace, dSPACE

Keywords

Hardware-in-the-loop, HIL, Battery management system, BMS, Testing, Fomula Student, Validation, dSPACE

Bibliografická Citace

PŘIKRYL, M. *HIL testování bateriového systému studentské formule*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2023. 63 s., Vedoucí diplomové práce: Ing. Roman Adámek.

Prohlašuji, že jsem diplomovou práci *HIL testování bateriového systému studentské formule* vypracoval samostatně pod vedením Ing. Romana Adámka, s použitím materiálů uvedených v seznamu literatury a interních zdrojů dostupných v rámci TU Brno racing.

Matouš Přikryl

Brno . . . .

Děkuji tímto Ing. Romanovi Adámkovi za vedení práce a cenné poznámky k jejímu vypracování. Také děkuji všem, kteří svými radami, připomínkami a motivací přispěli k dokončení práce. Zvláště pak členům týmu TU Brno racing za jejich podporu.

Matouš Přikryl

Obsah

1 Úvod

V době, kdy téměř všechna elektrická zařízení v sobě ukrývají řídicí systém, který určuje chování produktu, potřebujeme procedury, které zajistí správné a rychlé testování. Testováním myslíme ověřování, zda chování testovaného zařízení odpovídá požadavkům, které na ně klademe.

V minulosti bylo běžné produkt testovat, jakmile byl celý dokončen a bylo možné vyzkoušet jeho běžné provozní stavy. S rostoucím počtem a komplexitou řídicích jednotek ale tento přístup k testování nese určitá rizika, kterým je možné se vyhnout, pokud je věnována testování větší pozornost a je zvolen správný postup. Příkladem konkrétního rizika, které klasické testování přináší, je zjištění, že zařízení nefunguje podle předpokladů, až v čase, kdy je produkt dokončen. Takové zjištění často vede k potřebě přepracovat celý návrh zařízení.

Abychom rizikům předešli, je kladen důraz na vývoj nových řešení. Hardware-in-the-loop (HIL) testování je jedním z řešení, které umožňuje minimalizovat tato rizika, spojená s klasickým testováním. Umožňuje testovat dříve v rámci vývojového procesu a využívat složitější testovací sekvence. To může pomoct odhalit chybu dříve a dát tak impulz pro upravení konceptu zařízení. V neposlední řadě jsou velkým benefitem také možnosti automatizování testovacích procesů, nebo zvýšená bezpečnost při testování systémů kritických pro bezpečnost.

Právě bezpečnost je velkým tématem posledních let. S rozvojem elektromobility a autonomních systémů jsou na ni kladeny zvýšené požadavky. Se zvýšenými požadavky bezpečnosti jdou ruku v ruce zvýšené požadavky na testování.

Stejným směrem, jako vývoj v automobilovém průmyslu, se ubírá i studentská soutěž Formule Student. V rámci soutěže je úkolem každý rok postavit závodní monopost, který splňuje pravidla soutěže a je schopný závodit proti ostatním týmům. Formule se spalovacím motorem jsou v rámci soutěže pomalu upozaďovány a kapacity pro vývoj jsou směřovány k elektrickým pohonům a bateriovým systémům.

Tým TU Brno racing vyvíjí elektrické formule poslední tři roky a výsledkem vývoje během probíhající sezóny je závodní monopost Dragon e3, viz obr. 1.1. Energie pro pohon je uložena v baterii s maximálním napětím 600V. Kontakt s takto vysokým napětím je život ohrožující a bezpečnost je proto na prvním místě. Bateriový systém, který působí jako zjednodušeně řečeno jako ovladač baterie, musí být schopen reagovat na poruchy a nebezpečné stavy. Testování poruch není klasickými metodami dosažitelné a bylo proto rozhodnuto o využití HIL testování, jehož zprovoznění je hlavním cílem této práce.

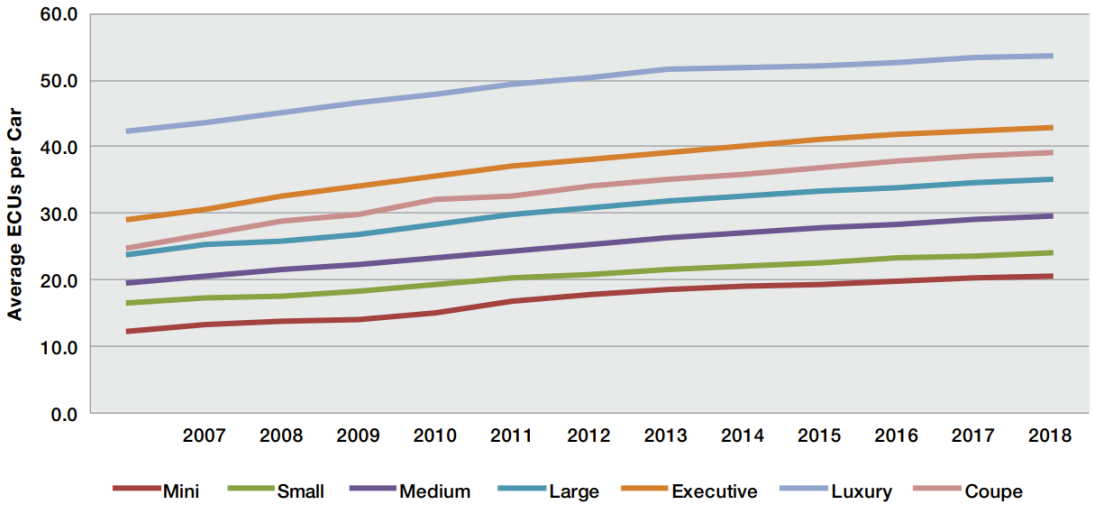
V práci jsou popsána řešení, která se nabízí pro testování a vývoj algoritmů bateriových systémů, jejich výhody, nevýhody a současné trendy této oblasti v rámci automobilového průmyslu. Následně je popsán bateriový systém studentské formule tak, abychom mohli pochopit jak bylo koncipováno jeho testování, které je popsáno v druhé polovině práce. Na závěr je také proveden popis procesu validace testů.

Obrázek 1.1: Tým TU Brno racing s formulí Dragon e3. Autor: Jan Prokopius

2 Testování řídicích jednotek

V posledních desetiletích se zvyšuje složitost a počet řídicích jednotek (ECU - electronically controlled unit), to má za následek razantní zvyšování nároků na vývojové procedury. S rozvojem autonomních technologií tento trend dále zrychluje a je potřeba dbát na bezpečnost vysoce náročných algoritmů. Z toho důvodu jsou vyvíjeny pokročilé metody testování, které reflektují rostoucí náročnost technologií a umožňují odhalit chyby spolehlivěji a dříve, než je běžné v případě klasických metod [1].

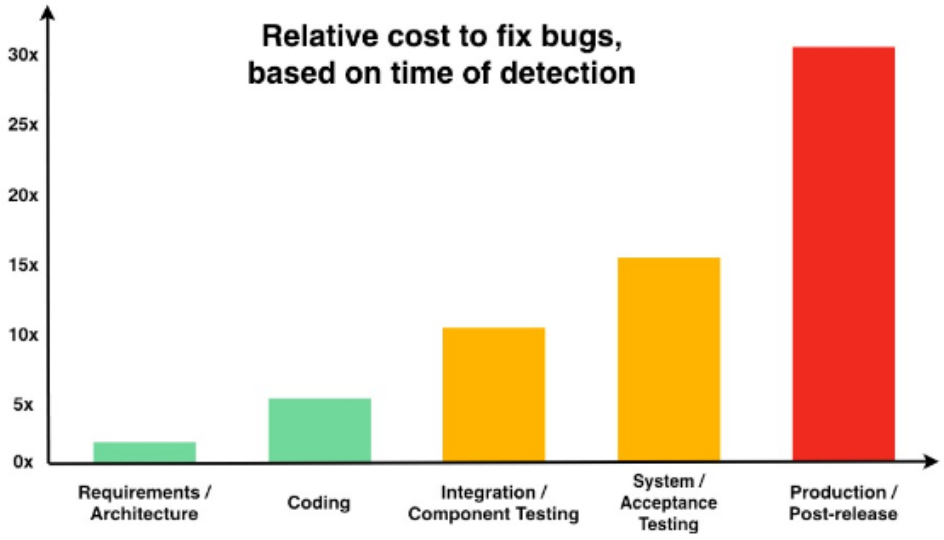
Problematiku můžeme jednoduše představit na rychle se zvyšujícím počtu řídicích jednotek využívaných v osobních automobilech. Z obrázku 2.1 je patrné, že v menších automobilech se počet řídících jednotek během deseti let téměř zdvojnásobil. U větších a dražších aut není relativní změna v počtu tak velká. U luxusních automobilů se také ve velkém rozvíjí obrazovky a další přídavné funkce, které vyžadují složité testovací procedury.



Obrázek 2.1: Počet řídicích jednotek v autech v posledních letech [2]

Hlavní omezení vývoje jsou spjaty s časem a penězi, které jsou k němu potřeba. Správná metoda testování, nebo jejich kombinace, může pomoct minimalizovat ať už potřebný čas, tak výdaje na případné opravování chyb. Pomocí toho jsme proto schopni přispět ke zdárnému dokončení vývoje a bezproblémové produkci produktu. Obecně platí, že čím později chybu v návrhu nalezneme, tím větší dopady na vývoj se dají očekávat. Na obr. 2.2 jsou zobrazeny náklady na opravu chyby podle doby jejího nálezu. Vidíme, že zvláště ve finální fázi návrhu, ve které se zařízení blíží k produkci, je nárůst nákladů vysoký. Toto pravidlo platí i v soutěži Formule Student. Pokud je chyba nalezena ve finální části testování, nebo v rámci kontroly vozidla, může tým přijít o možnost účastnit se

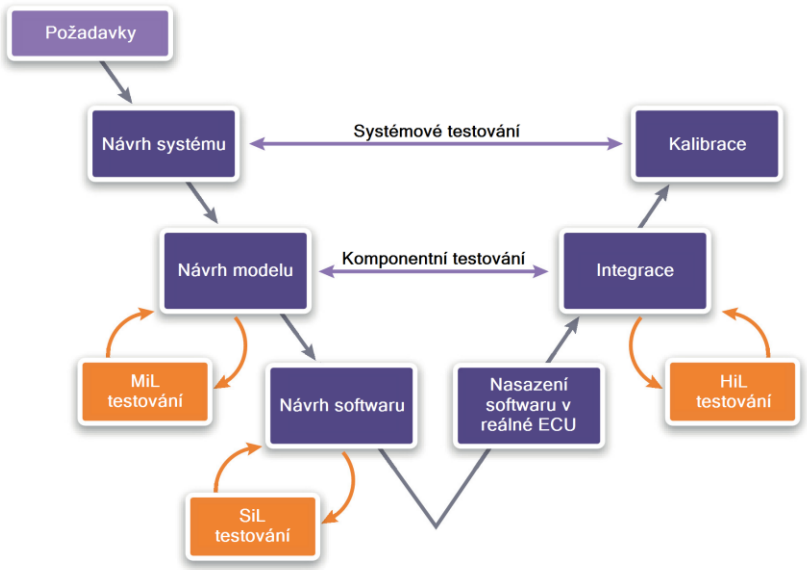
dynamických disciplín na závodech, což velmi negativně ovlivňuje hlavní výsledky sezóny.



Obrázek 2.2: Cena opravy chyby v závislosti na fázi vývoje [3]

V posledních letech zažívá velký rozmach vývoj softwaru (SW) za použití modelů (model-based design - MBD). Model je jednoduché přizpůsobovat měnícím se požadavkům a může být pro programování přehlednější alternativou velkého množství kódu. S jeho pomocí můžeme také lépe specifikovat požadavky na navrhovaný systém a řízení vývoje ve velké skupině lidí [4].

Vývojový proces můžeme dobře popsat pomocí V-modelu, viz obr. 2.3. V-model je sestaven tak, aby na levé straně byly kroky návrhu a na pravé straně kroky testování. V bodě, kde se nachází zlom, je implementace software na hardware. Pokud je v kterékoli části nalezena chyba, přesouváme se zpět k návrhu a úpravě chyby.



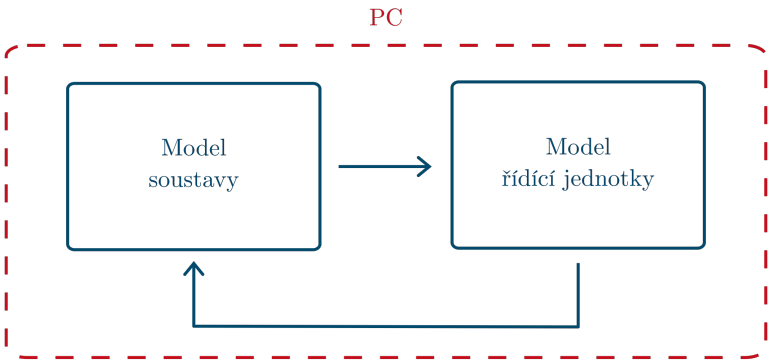
Obrázek 2.3: V-model pro MBD [5]

Testování v rámci MBD je označováno jako testování ve smyčce (in-the-loop) [6]. Testovaná řídicí jednotka svými výstupy reaguje na vstupy, které přichází ze simulované soustavy. Vytváří tak smyčku, ve které můžeme efektivně hledat chyby. V následující části práce se zaměříme právě na jednotlivé metody testování ve smyčce využívané při návrhu SW pomocí MBD.

2.1 MIL

Model-in-the-loop (MIL) je první z metod testování ve smyčce. Využíváme model soustavy a řídící jednotky. Oba modely máme simulovány v počítači, kde vytváří uzavřenou testovací smyčku, viz obr. 2.4. Pomocí MIL testujeme, zda je model řídicí jednotky správně navržen a zda splňuje všechny zadané požadavky. Klíčová je přesnost modelu soustavy, kterou má řídicí jednotka ovládat. Pokud je model dostatečně přesný, jsme schopni otestovat reakci řídicího systému na stavy, které mohou v reálné soustavě nastat. Jaká je dostatečná přesnost modelu záleží na daném případě. Některé systémy jsou citlivé na malé změny, jiné jsou naopak relativně málo citlivé a nepřesnosti modelu nemusí kvalitu řídicího algoritmu ovlivnit.

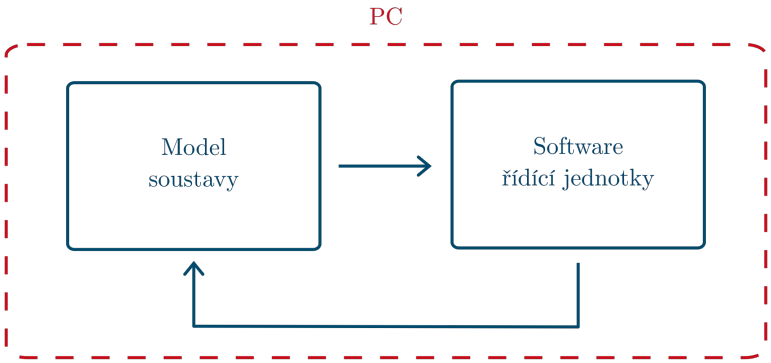
Pokud máme model, můžeme jednoduše měnit parametry obou částí, řídící i řízené, a sledovat změny které úpravy přinesou. Pomáhá nám to rychle optimalizovat návrh řídicích jednotek. Při kontrole funkčnosti modelu ECU můžeme jednoduše navodit chybu, která by mohla vytvořit nebezpečný stav a sledovat chování řídicí jednotky. Pokud je chování vyhovující, můžeme přejít k další fázi testování.



Obrázek 2.4: Schéma MIL

2.2 SIL

Software-in-the-loop (SIL) je dalším krokem testování při vývoji software pomocí metody MBD. Testovaný model řídicí jednotky je využit pro generování kódu, kterým nahrazujeme její model. Simulační prostředí, využitelné pro SIL testování, například Simulink, poskytují blok do kterého vložíme generovaný kód. U kódu testujeme numerickou ekvivalenci s modelem testovaným při MIL testování. Model soustavy zůstává stejný, jako u MIL testování. Porovnáme-li výsledky MIL a SIL měli bychom dostat stejný, nebo minimálně lišící se výsledek. Obrázek 2.5 ukazuje názorně proces SIL testování.



Obrázek 2.5: Schéma SIL

2.3 PIL

Processor-in-the-loop (PIL) testování se přibližuje k finálnímu hardware na kterém poběží kód. Na obrázku 2.6 vidíme, že software, který byl doposud simulován na osobním počítači, je nyní generován a následně nahrán na cílový procesor. Procesor je využit k nahrazení software řídicí jednotky v rámci SIL testování. Z komunikace procesoru s počítačem, na kterém stále běží simulace soustavy, můžeme určit, zda je procesor schopen spustit kód. Sledujeme například vytížení procesoru, využití paměti a odchylky od výsledků získaných v předchozích krocích testování.

Obrázek 2.6: Schéma PIL

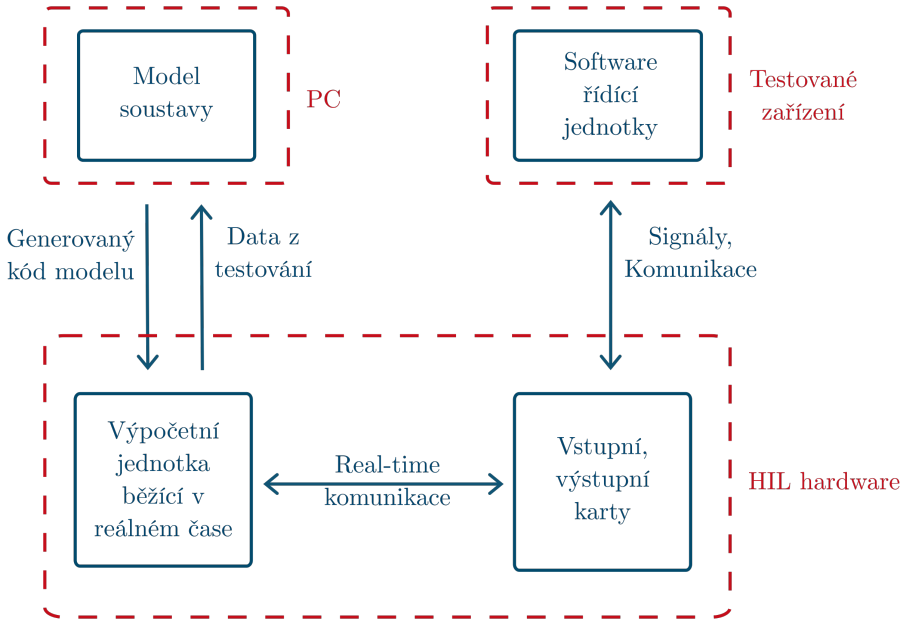
2.4 HIL

Hardware-in-the-loop (HIL) testování je hojně využívanou metodou testování. Časté je testování i řídicích jednotek, které byly programovány přímo psaním kódu, a nebyl na ně aplikován vývoj software metodou MBD. HIL testování umožňuje testování složitých algoritmů v prostředí, které je blízké finální soustavě. Na vstup ECU přivádíme stejné elektrické signály, které se objevují v systému při zapojení k reálné řízené soustavě [7]. Hodnoty analogových i digitálních signálů určujeme pomocí simulace ostatních částí systému. V

případě MBD využíváme stejného modelu, jako v předchozích případech. Přehledně máme testovací soustavu zobrazenou na schématu 2.7.

Nevýhodou této metody testování může být náročnost na technické vybavení. K její aplikaci je potřeba zařízení běžící v reálném čase (real-time). To znamená, že je časově deterministické a jsme schopni určit v jakém okamžiku se stane jaká akce. Tato real-time jednotka je propojená s vstupními a výstupními kartami (I/O kartami) generujícími signály. V ideálním případě generujeme stejné signály, které vystupují z reálného systému.

HIL testování nám poskytuje mnoho výhod oproti běžnému testování. Jednou z výhod je možnost vyvolat stejný chybový stav opakovaně tak, že navodíme totožný stav systému (zapneme stejnou testovací sekvenci znovu). To může být u reálné soustavy obtížné, až neproveditelné. Pokud navíc zaznamenáváme výstupy řídicí jednotky, jsme ze zaznamenaných dat jasně schopni určit kdy chyba nastala. Výstupy ECU v reálné soustavě je značně náročnější zaznamenávat, Často jsou k tomu potřebné speciální zařízení pro záznam signálů. Dále nám HIL umožňuje testování řídicích jednotek, pro které nemáme dostupný zbytek soustavy. Jsme proto schopni testovat dříve, i před dokončením řízené soustavy, a minimalizovat tak náklady na opravu chyb. V neposlední řadě jsme také schopni bezpečně testovat stavy systému, které by mohly vést k nebezpečným situacím. Dobrým příkladem je testování bateriových systémů, kde je náročné testovat některé stavy. Pro příklad můžeme uvést testování ochrany proti přehřátí baterií, kdy nesprávná manipulace s baterií může poškodit, nebo nenávratně zničit celý systém. Nemluvě o rizicích hrozící obsluze při provádění takových testů.



Obrázek 2.7: Schéma HIL

V literatuře bývá dále definována power hardware-in-the-loop (pHIL). Některé signály vystupující z I/O karet v tomto případě vstupují do zesilovačů a jiných výkonových částí, které jsou schopny simulovat výkonovou část systému. Velkou výhodu může tato metoda přinést například při testování měniče, kdy potřebujeme simulovat motor včetně napětí a proudů. Značné využití má také při simulování a testování bateriových systémů. BMS

často vyžaduje připojení velkého počtu izolovaných zdrojů napětí, čehož nejsme běžnými I/O kartami schopni dosáhnout.

2.4.1 Hardware

Výběr hardwaru (HW) pro HIL je důležitým krokem pro kvalitní testování. Výrobci nabízí mnoho rozdílných typů zařízení a je potřeba si před výběrem určit co k testování potřebujeme. Ukázky HW určeného pro HIL testování můžeme vidět na obr. 2.8.

(a) dSPACE real-time zařízení [8] (b) speedgoat real-time zařízení [9]

Obrázek 2.8: Ukázka real-time zařízení

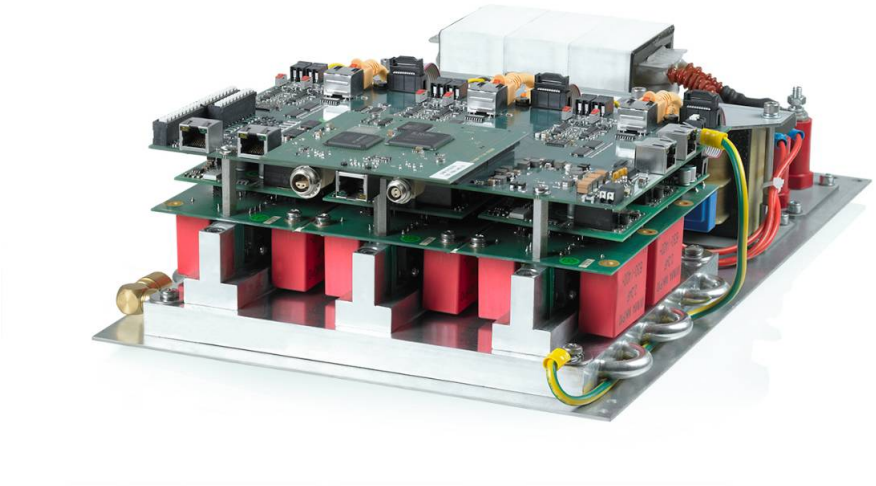
Kolik vstupů a výstupů je potřeba? Jaké komunikační sběrnice využíváme? Potřebujeme simulovat i výkonovou část ECU? To jsou některé z otázek, které je důležité před začátkem testování zodpovědět. Od náročnosti našich požadavků se následně odvíjí výše investice pro pořízení vhodného HW.

Základní varianty mívají podporu několik základních digitálních sběrnic jako Ethernet, CAN, LIN, RS232 a několik analogových a digitálních vstupů a výstupů. U náročnějších zařízení, které mohou obsahovat i stovky jednotlivých vstupů a výstupů je většinou možné připojovat či vyměňovat I/O moduly dle potřeby. Máme-li tedy projekt, u kterého si nejsme jisti potřebnými periferiemi, nebo chceme využít zařízení pro více projektů s odlišnými požadavky, je takové zařízení optimální možností.

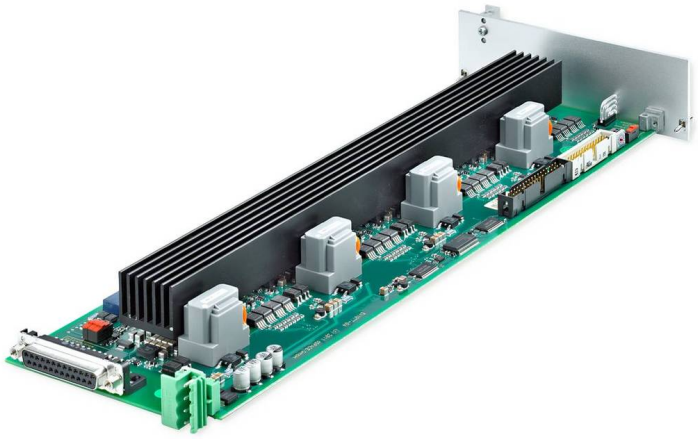
Pro aplikace náročné na výpočetní výkon, kde je žádaná rychlá a přesná kalkulace časových kroků, je možnost využít FPGA modul. Ten zaručí dostatek paralelních výpočetních procesů i pro rychlé řídicí algoritmy.

K testování řídicích systémů pro autonomní funkce automobilů jsou připraveny moduly simulující výstupy z kamer a jiných senzorů. Pokud míříme k aplikaci pHIL, můžeme si vybrat z nejrůznějších elektronických zátěží (obr. 2.9), které zvládnou i napětí pohybující se kolem 1000V a proudy přes 100A. Můžeme tak simulovat elektromotory nebo jiné výkonové spotřebiče.

Pro testování bateriových systémů je taktéž připraven specifický hardware (obr. 2.10). Můžeme si vybírat z mnoha variant izolovaných zdrojů simulujících jednotlivé články bateriového segmentu, ale také simulátory teplotních senzorů nebo jednotky schopné fyzicky simulovat chybu v systému. Například zkrat signálu na zem nebo na napájení. Vybrané karty izolovaných zdrojů jsou vypsány v tabulce 2.1. Karty lze u náročnějších HIL zařízení kombinovat a přidávat, takže je možné využít několik karet pro simulaci většího počtu článků. Moduly je potřeba vybírat s ohledem na maximální napětí článků, přesnost výstupního napětí, nebo také maximální proud, který chceme z karty odebírat.



Obrázek 2.9: Modul vysokonapěťové elektronické zátěže - dSPACE [10]



Obrázek 2.10: Modul pro simulaci napětí článků baterie - dSPACE EV1077 [11]

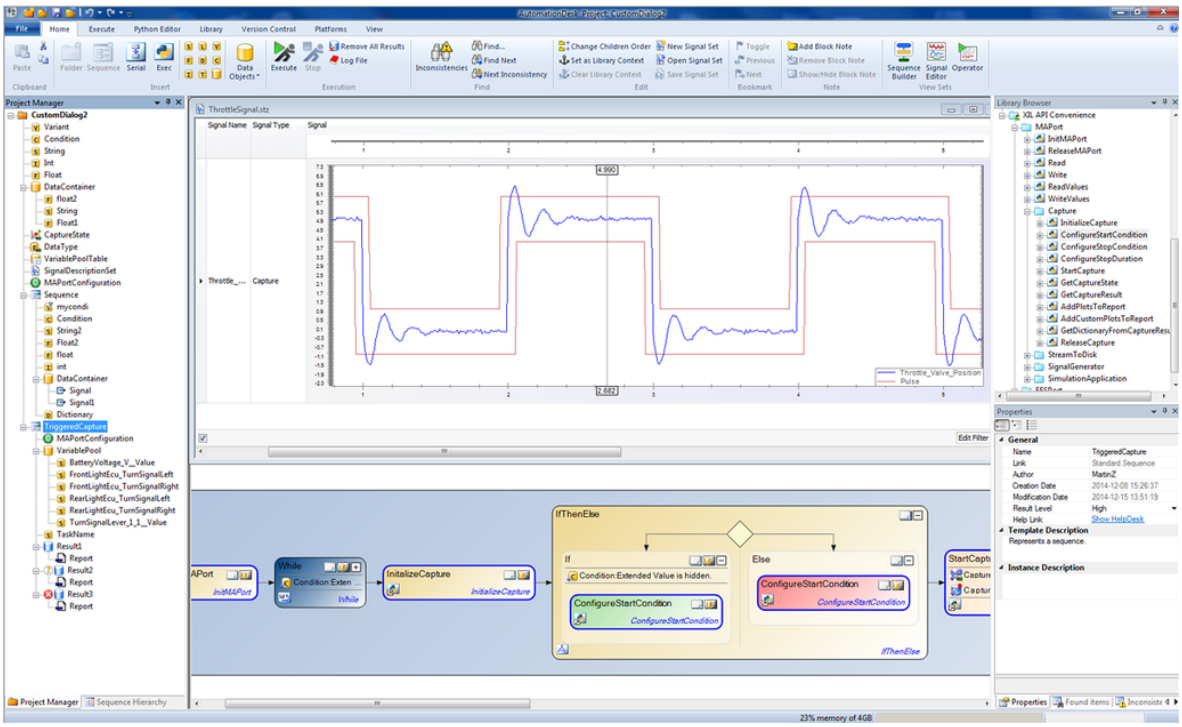
	EV1077 [11]	DS5481 [12]	IO992 [13]
Výrobce	dSPACE	dSPACE	speedgoat
Počet kanálů	4	2	6
Výstupní napětí [V]	0..6	-6..6	0..7
Rozlišení [µV]	115	10	430
Přesnost [mV]	± 0,5	± 0,3	5
Výstupní proud [A]	1 nebo 2 (volitelné)	5	0,3
Izolace mezi články [V]	60	60	750
Izolace mezi články a prostředím [V]	1000	1500	750

Tabulka 2.1: Porovnání vybraných karet pro simulaci článků baterií

2.4.2 Software

Software, který je použit při HIL testování je úzce svázaný s hardware. Společnosti vyvíjejí svoje programy, aby využily naplno potenciál jejich zařízení. Popíšeme proto alespoň několik variant software, které je možné využít pro určitý hardware.

U společnosti dSPACE je základním softwarem Simulink, do kterého integrují svoje rozšíření, real-time interface (RTI). To posiluje schopnosti Simulink Coder generovat C kód tak, abychom byli schopni ovládat vstupy a výstupy na základě signálů v modelu. Ze simulinku je tedy generován kód, který běží v reálném čase na dSPACE hardware. Simulink již není schopen přijímat zpět data z hardwaru. Komunikace je tedy obstarávána pomocí jednoho, nebo více programů od dSPACE. Např. ControlDesk, ConfigurationDesk, nebo AutomationDesk. Každý z nich dokáže obstarat jiné úkony. Na obr. 2.11 vidíme ukázku nastavení automatizovaného vyhodnocení výsledků testu v software AutomationDesk.



Obrázek 2.11: AutomationDesk - dSPACE [14]

Společnost Speedgoat spoléhá se všemi funkcemi na Simulink. Využívá Simulink Real-Time, který generuje kód pro zařízení reálného času, ale také přijímá data zpět. Můžeme proto automatizovat testování přímo v programu. V kombinaci s rozšířeními jako je App Designer, nebo Simulink Test jsme schopni dosáhnout efektivního testování stejně jako u dSPACE, ale s využitím méně samostatných programů.

Další společnosti, jako například National Instruments, nebo Vector, spoléhají na jiná řešení, která jsou specifická pro každého z nich.

3 Bateriový systém studentské formule

Bateriový systém studentské formule (BMS) je řídicí jednotka konstruovaná tak, aby kontrolovala funkci baterie, zajišťovala její bezpečný provoz a baterii v případě potřeby zcela odpojila od auta. Na starosti má také teplotní okruh baterie, zajišťující chlazení systému.

BMS každý rok navrhují členové týmu TU Brno racing takovým způsobem, aby splňoval požadavky jak na bezpečnost, tak i funkčnost a dokázala spolehlivě komunikovat s ostatními částmi formule. Návrh BMS je popsán v bakalářské práci [15]. Návrh již není možné brát doslovně vzhledem k tomu, že se zabývá návrhem BMS pro formuli eD1 ze sezóny 2020/2021 a část systému byla přepracována.

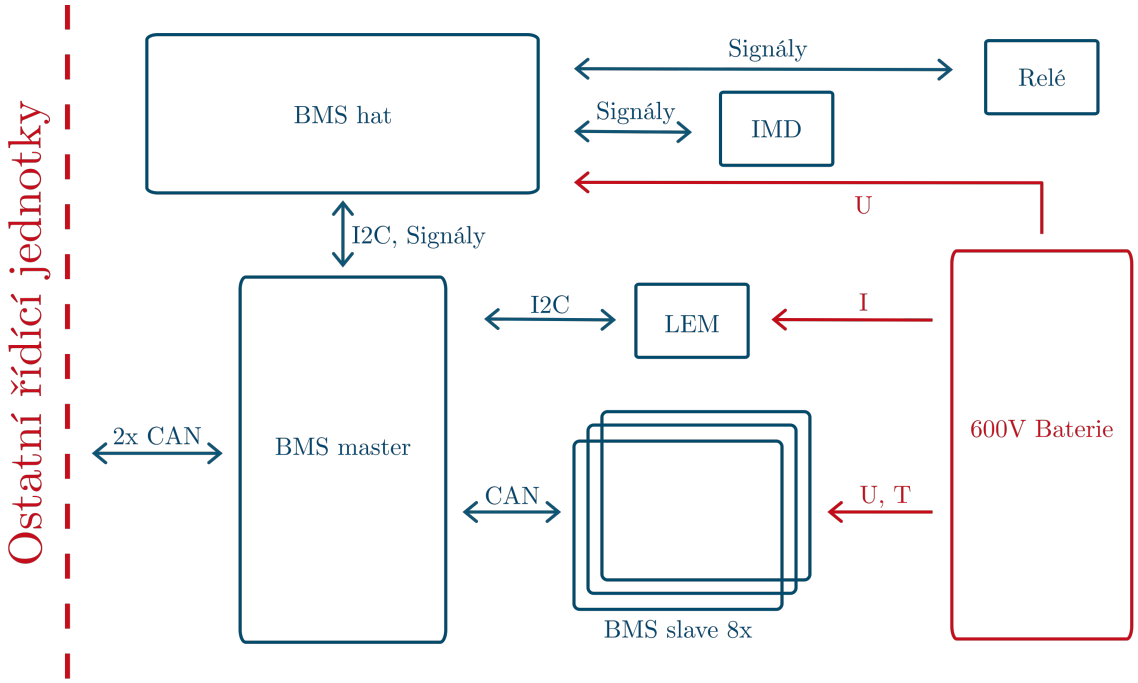
Abychom mohli popsat bateriový systém pro potřeby HIL testování, je potřeba nejdříve uvést koncept vysokonapěťové baterie. Ta se skládá ze 136 článků, které jsou seskupeny do osmi segmentů po 17 článcích. Všechny články jsou řazeny sériově, přes všechny tedy proudí stejný proud a nejslabší článek určuje kapacitu celé soustavy. Každý jednotlivý článek má maximální provozní napětí 4.35V a minimální provozní napětí 3V. Maximální dovolená teplota je 60°C. Minimální teplota není omezena pravidly. Pokud je dosažen některý z uvedených limitů, BMS reaguje a rozpojuje výkonovou část tak, aby nebylo možné dále odebírat výkon. Pokud je potřeba, reaguje také na signál ze zařízení kontrolujícího izolační odpor pólů baterie (insulation monitoring device - IMD) vůči tělu formule. V případě nízkého odporu by se mohlo vysoké napětí objevit na rámu a ohrozit tím řidiče, nebo lidi pohybující se okolo.

Bateriový systém je navržen jako distribuovaný systém. Skládá se tedy z několika částí. BMS master, BMS Hat a BMS slave. BMS slave je připojen na každém jednotlivém segmentu a stará se o měření teplot a napětí jednotlivých článků. Měření je následně posíláno na CAN sběrnici. BMS hat obstarává spínání izolačních relé (accumulator insulation relay - AIR), měření vysokého napětí a zajišťuje další funkce popsané v kapitole 3.2. BMS master zpracovává všechny příchozí data z formule a ostatních částí baterie. Ty následně využívá pro řízení stavového automatu, který určuje stav řídicí jednotky. Zjednodušené schéma BMS je zobrazeno na obrázku 3.1 a můžeme z něj vyčíst komunikaci mezi jednotlivými částmi. Modrou barvou jsou zvýrazněny digitální komunikace a řídicí jednotky, červenou barvou jsou poté zvýrazněny analogové signály a ostatní fyzické části baterie.

Důležitou součástí je také vypínací obvod (SDC - shutdown circuit). Obvod je velmi pečlivě popsán v pravidlech soutěže formule student [16]. Důležitá část SDC pro HIL je popsána v kapitole 3.4.

3.1 BMS slave

Battery management system slave je částí bateriového systému, která je v přímém kontaktu s bateriovým segmentem. V nejnovější verzi formule, Dragon e3, je rozdělena na dvě části, vrchní a spodní (Top a Bottom). Rozdělení lze názorně vidět na schématu 3.2.



Obrázek 3.1: Schéma BMS

Bottom Slave se stará o pasivní balancování článků baterie (spíná tranzistor, který připojí paralelně s článkem rezistor). Zároveň měří napětí na všech článcích a teplotu na záporných pólech dvanácti článků (ze 17ti v jednom segmentu). Měřené veličiny následně odesílá po sběrnici SPI (Serial Peripheral Interface) vrchní části BMS slave.

Top slave izoluje trakční systém formule od nízkonapěťové části a přeposílá data z SPI na CAN sběrnici. CAN sběrnici, po které komunikují všechny BMS slave s BMS master nazýváme CAN slaves.

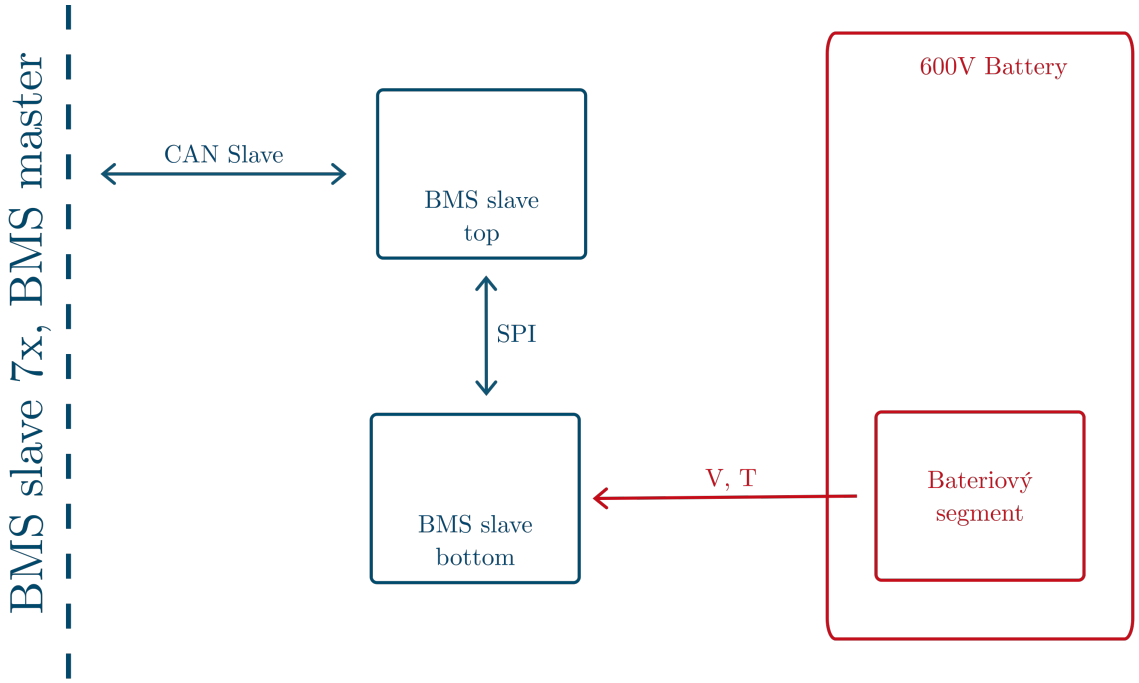
Autorem návrhu BMS slave pro Dragon e3 je Jakub Lysák.

3.2 BMS Hat

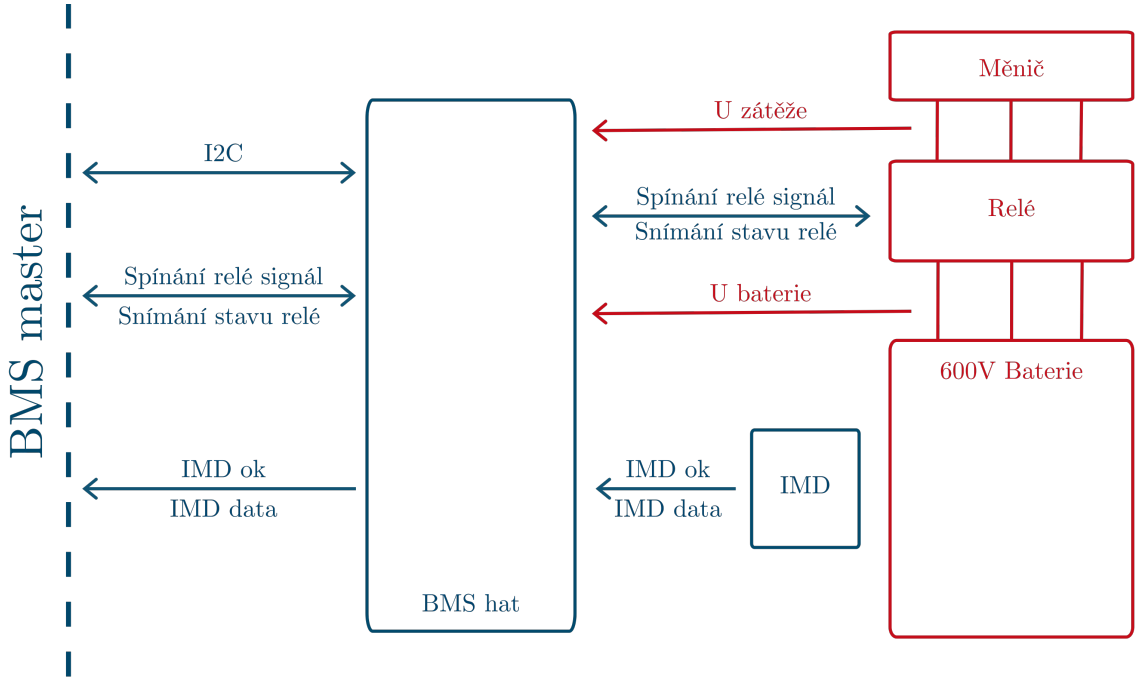
Battery management system Hat již neřeší jednotlivé články, ale celkový pohled na baterii. Zapojení je možné vidět na obrázku 3.3. Jednou z funkcí je měření celkového napětí mezi záporným a kladným pólem baterie a napětí na kondenzátorech meziobvodu měniče. Napětí může dosahovat maxima 591.6V. Je proto před měřením na analogově digitálním převodníku (ADC) nejdříve rozděleno pomocí napěťového děliče a následně galvanicky izolováno tak, aby nebyl spojen trakční systém s nízkonapěťovou částí. Měřené hodnoty jsou odesílány po sběrnici I^2C.

Druhou funkcí je kontrola stavu IMD. Stav jsme schopni zjistit ze dvou signálů. IMD OK je logický signál, který je ve stavu jedna (napětí 2V), pokud je izolační odpor baterie v pořádku. Ve stavu nula (napětí 0V), pokud není v pořádku, nebo nastala jiná chyba na IMD. IMD data je PWM signál, který mění svoji frekvenci a periodu v závislosti na velikosti izolačního odporu a stavu zařízení [17].

Poslední důležitou funkcí BMS Hat je spínání izolačních relé pro připojení baterie k měniči viz obr. 3.4. Relé jsou tři. Pro kladný pól baterie POS, pro záporný pól NEG. Třetí (PRE relé) připojuje měnič přes rezistor, který má omezit velikost proudu při nabíjení kondenzátorů meziobvodu. Relé musí být spínány a rozepínány v jasně definovaném pořadí



Obrázek 3.2: Schéma jedné BMS slave



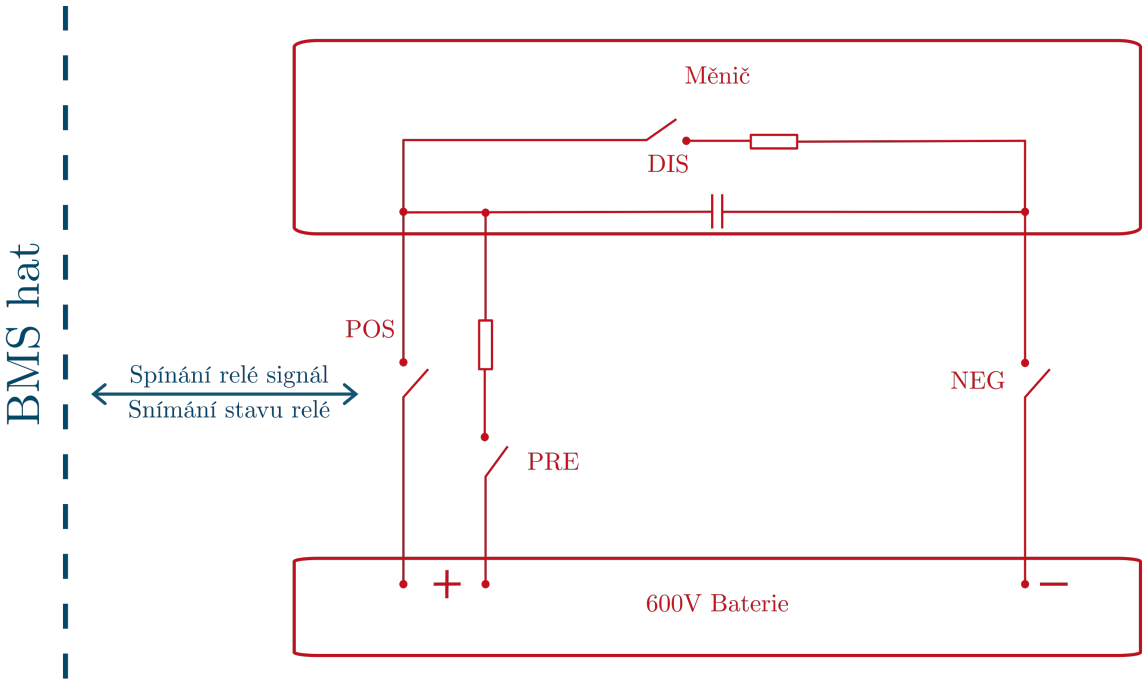
Obrázek 3.3: Schéma BMS hat

a při každém sepnutí a rozepnutí je kontrolováno, zda se systém chová podle předpokladů. Zapínací sekvence je podrobněji popsána v kapitole 3.5.

Součástí měniče je následně také vybíjecí tranzistor (DIS) s rezistorem, který má za úkol vybít kondenzátory meziobvodu, pokud je vypnut trakční systém, nebo nastal nebezpečný stav v kterékoli části formule. Spínání DIS tranzistoru je napojeno na SDC final (viz 3.4). Pokud není na SDC final napětí, je tranzistor sepnutý. V rámci měniče je

také implementován obvod, který pozdrží rozepnutí DIS tranzistoru o 3,6 sekundy, aby bylo zaručeno vybití kondenzátorů meziobvodu měniče i pokud je SDC final bez napětí jen chvíli. Návrh měniče je popsán v souběžné diplomové práci [18].

Autorem návrhu BMS Hat pro Dragon e3 je Jakub Lysák.



Obrázek 3.4: Schéma připojení baterie k měniči

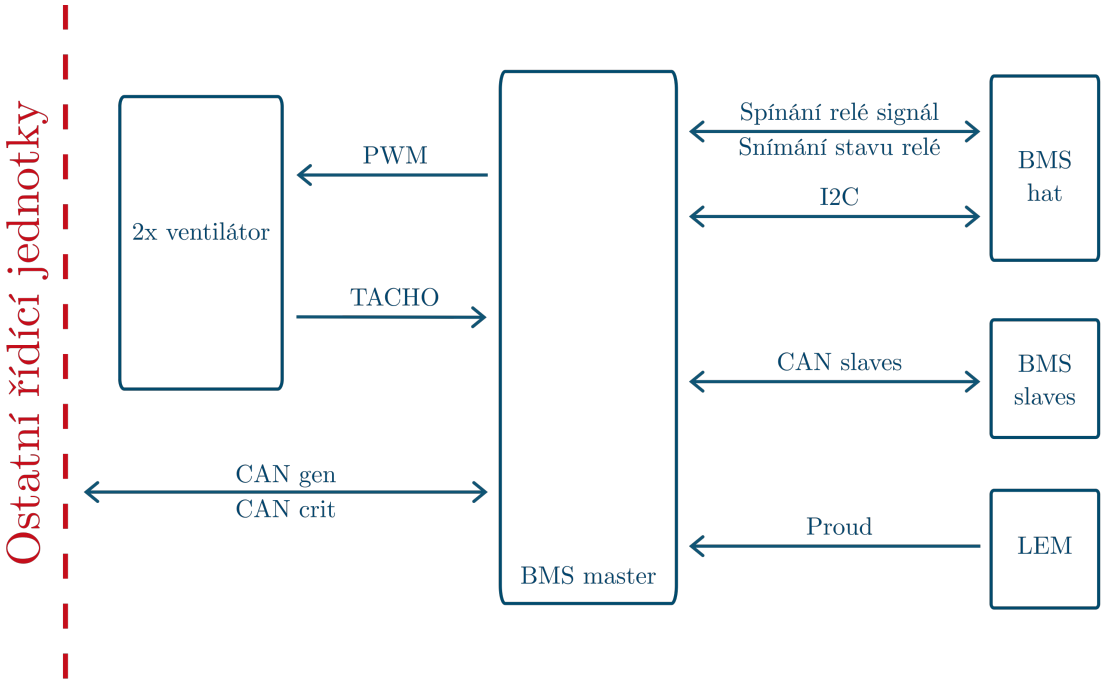
3.3 BMS Master

Battery management system Master je mozkem celé BMS. Přijímá data ze všech ostatních částí a rozhoduje o sepnutí, nebo rozepnutí AIR. Využívá dvou sběrnic I^2C. Jedné pro komunikaci s LEM senzorem, který měří proud odebíraný z baterie. Druhé pro komunikaci s BMS hat, kde měříme napětí meziobvodu měniče a napětí baterie. Ostatní digitální komunikace probíhá přes tři sběrnice CAN. CAN slaves slouží k přenosu informací o napětí a teplotách článků baterie. CAN crit slouží pro předávání důležitých dat dalším řídicím jednotkám a obsluze formule. CAN gen poté slouží k přenosu méně důležitých dat, které jsou využívány například při hledání chyb v systému. CAN gen obstarává také komunikaci mezi nekritickými částmi formule.

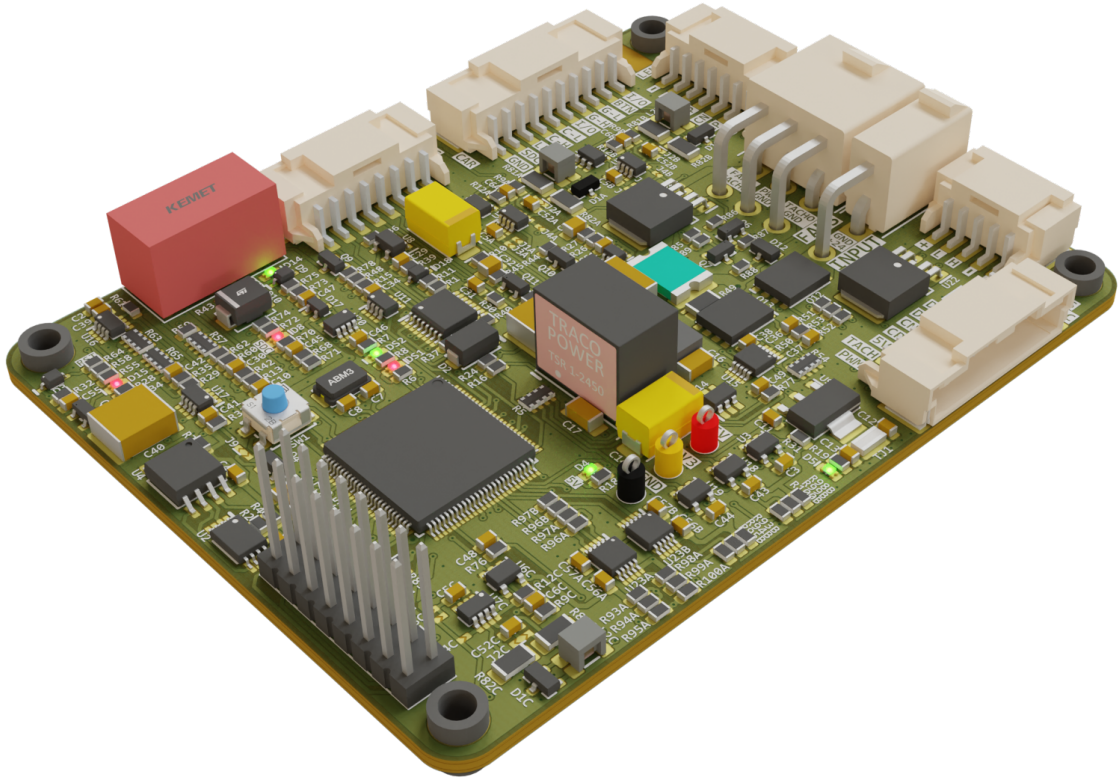
Z analogových signálů je důležité zmínit ovládání relé a snímání jejich stavu. Signály jsou posílány nejdříve přes BMS hat, kde jsou upraveny pro účely spínání a měření.

K BMS master je možné připojit počítač přes UART za pomocí programovací jednotky. K připojení je využit program PuTTY a COM port počítače. Pro správné připojení je potřeba nastavit baudrate komunikace na hodnotu 500 kbit/s. Po připojení je vypsán přehled nastavených hodnot. Této funkce je hojně využíváno při hledání chyb a sledování stavu BMS během testování. Výpis můžeme vidět na obr. 3.7.

DPS BMS master byla navrhnuta pro účely týmu TU Brno racing autorem této diplomové práce a její vizualizaci je možné vidět na obrázku 3.6.



Obrázek 3.5: Schéma BMS master



Obrázek 3.6: Render BMS master. Autor: Martin Macko

```
BMS Master
Controls: N - Next menu screen   B - Last menu screen    T - Turn on TS          R - System reset
          Q - PRE rel    W - POS rel     E - NEG rel

Master state: STATE_ERROR_RESET                 IMD State: 0    SDC Fail: 1     SDC Final: 0    SDC Monitoring: 0
CANStates:       GEN: 0          CRIT: 0        BMS: 0
CANOpen: 127 536872936 0
AIR States:      NEG: 0          POS: 0         PRE: 0
CAN TS Command: 0
CAN Charge command: 0
Error code: 8

Battery Voltage: ERROR
LOAD Voltage: ERROR
Battery Current: -47.81A

BATTERY_FAN_CURRENT: 0.000A      Slaves current 0.000A
BMS Master current: 0.078A       BMS Master temp 31.6C
SDC_FINAL voltage: 0.000V

Min cell: 6.5535V       Max cell: 0.0000V       Voltage delta: -6.5535V
Min cell: 655.35C       Max cell: 0.00C

Battery fans: 0 RPM
IMD state: NORMAL       resistance: 235kΩ
```

Obrázek 3.7: Výpis hodnot posílaný přes UART do PC

3.4 SDC

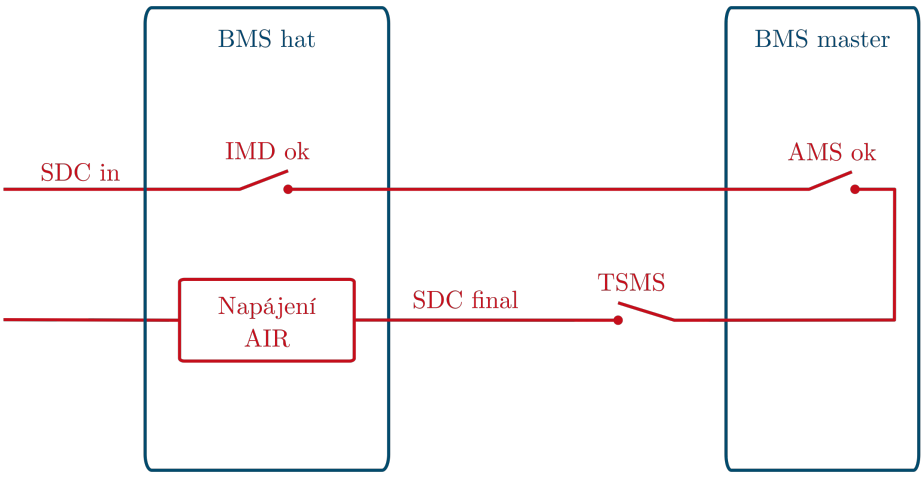
Vypínací obvod, anglicky shutdown circuit (SDC), je bezpečnostní obvod, který na svém konci napájí izolační relé. Ve formuli je několik míst, které mohou obvod rozpojit. Rozpojením obvodu zajistíme rozepnutí relé za všech okolností, i pokud se logika BMS zamrzne a bude nadále vysílat pokyn ke spojení kontaktů. Jediným nebezpečím v této chvíli mohou být svařené kontakty relé, které nelze rozpojit.

Na obrázku 3.8 vidíme vstup SDC (SDC in). Před tímto vstupem je zařazeno několik bezpečnostních prvků, jako například obvod kontrolující správnou funkčnost brzd, nebo bezpečnostní tlačítko v kokpitu řidiče. Pokud je vše před vstupem v pořádku, objeví se na něm napětí 24V. Obvod může být dále rozpojen v BMS na dvou místech. První z nich je na BMS master, kde relé rozpojuje obvod pokud je chyba v měření na BMS slave, nebo v případě vzniku jiné chyby v systému BMS. Druhé místo je na BMS hat. Zde je relé ovládáno signálem z IMD. Chyba IMD rozepne SDC.

3.5 BMS Software

Stavový automat, který běží na bateriovém systému je navrhován tak, aby byl přehledný a splňoval všechny požadavky na bezpečnost. Klade tedy důraz na bezpečnostní funkce. Základní, zjednodušenou strukturu, můžeme vidět na schématu 3.9. Modrou barvou jsou vyznačeny běžné stavy a přechody, které jsou využívány při normálním stavu BMS. Červenou barvou jsou přechody a stavy, které nastávají při detekci jakéhokoli neočekávaného chování systému. Software je naprogramován v jazyku C a nahrán na procesor BMS master.

Abychom mohli sledovat stav systému, je zavedena proměnná BMS status. Po zapnutí systému je navozen počáteční stav (status = 0) ve kterém čekáme na navázání komunikace s BMS slave. Minimální doba čekání je 2 sekundy. Pokud je komunikace po dvou sekundách v pořádku a měření ukazují správné hodnoty, přesouváme se do stavu BMS ready (status = 1). Zde čekáme na pokyn k sepnutí TS. Jestliže dojde k detekci pokynu poslaného přes komunikaci CAN crit, kontrolujeme a spínáme relé (viz 3.10). Pokud pro-



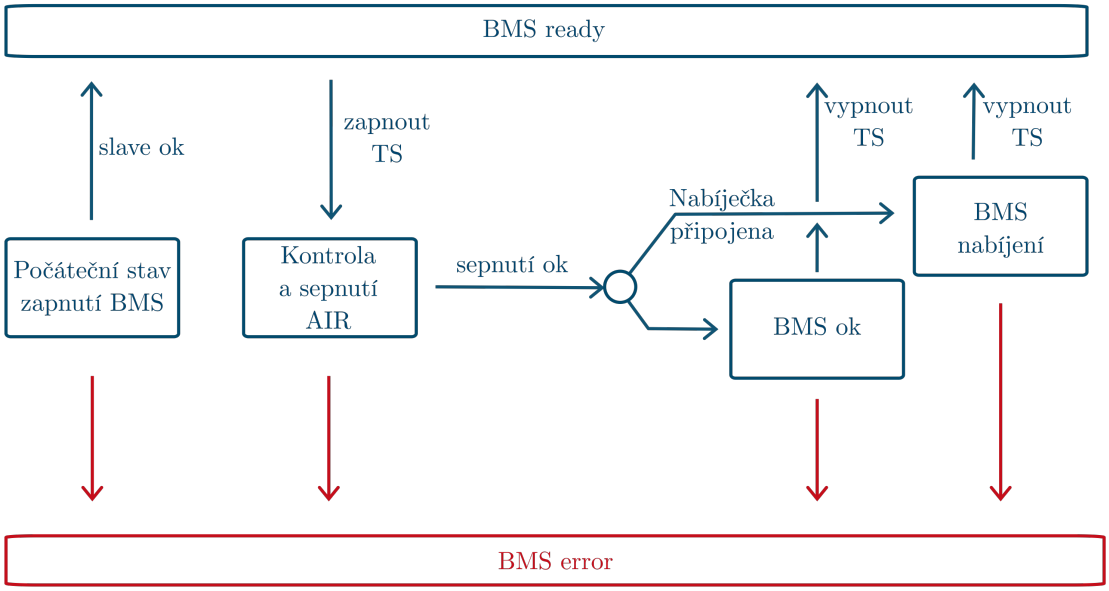
Obrázek 3.8: schéma vypínacího obvodu - SDC

běhlo připojení správně a všechny kontroly stavu byly v pořádku, rozhodujeme o dalším stavu BMS podle toho, jestli je připojená nabíječka k baterii. V případě že nabíječka není připojena, následuje stav BMS ok (status = 2). Pokud připojena je, následuje stav BMS nabíjení, ve kterém má status hodnotu 4. BMS error (status = 3) je stav, do kterého se dostáváme po detekci chyby v systému. Tento stav je možné opustit pouze pokud je procesor resetován. Restartování stavového automatu a opuštění stavu BMS error je možné provést pouze fyzickým stlačením tlačítka AMS reset z venkovní části formule. Tento úkon provádíme jen pokud jsme si jisti, že baterie není v nebezpečném stavu. Po restartování se vracíme zpět do počátečního stavu.

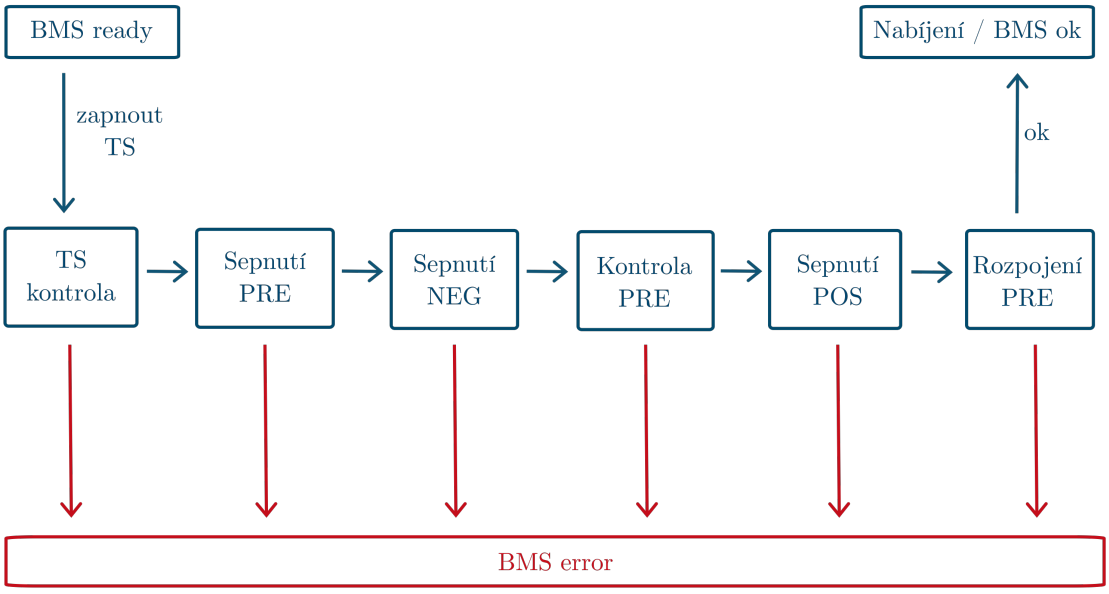
Další částí, kterou je potřeba popsat, je proces připojení baterie k formuli. Před přechodem k sekvenci spínání kontrolujeme, jestli jsou v pořádku měření na BMS slave, stav relé a napětí na kondenzátorech meziobvodu měniče, které nesmí překročit hodnotu 60V. Pokud nejsou nalezeny závady, kontrolujeme ještě, zda se shoduje součet napětí měřené na článcích s měřením celkového napětí baterie na BMS hat. Následuje sepnutí PRE AIR. Při každém sepnutí či rozepnutí jakéhokoli AIR čekáme 300 ms, abychom zajistili správnost kontroly stavu a nespoléhali se na měření během přechodového děje. Pokračujeme sepnutím NEG AIR. Tím zajistíme nabíjení kondenzátorů přes odpor zapojený sériově s PRE AIR. Během dvou až pěti sekund se musí kondenzátory nabít na hodnotu alespoň 95% napětí baterie. Proběhne-li vše v pořádku, je sepnut POS AIR a následně rozepnut PRE AIR. Takto je připojena baterie k měniči a je vše připraveno pro jízdu formule. Pro lepší představu je připraveno schéma spínání 3.10 a také schéma zapojení relé 3.4.

Z důvodu zachování přehlednosti a jednoduchosti pro obecný přehled funkce BMS schémata nezobrazují celý stavový automat, který v realitě obsahuje další pomocné stavy. Také nejsou vypsány veškeré kontroly které software obsahuje. Popis celého software by mohl být obsahem celé samostatné práce.

Ventilátor je ovládán z BMS master pomocí PWM. Hodnota PWM nastavuje SW v závislosti na nejvyšší naměřené teplotě v baterii, a to jen pokud se BMS master nachází ve



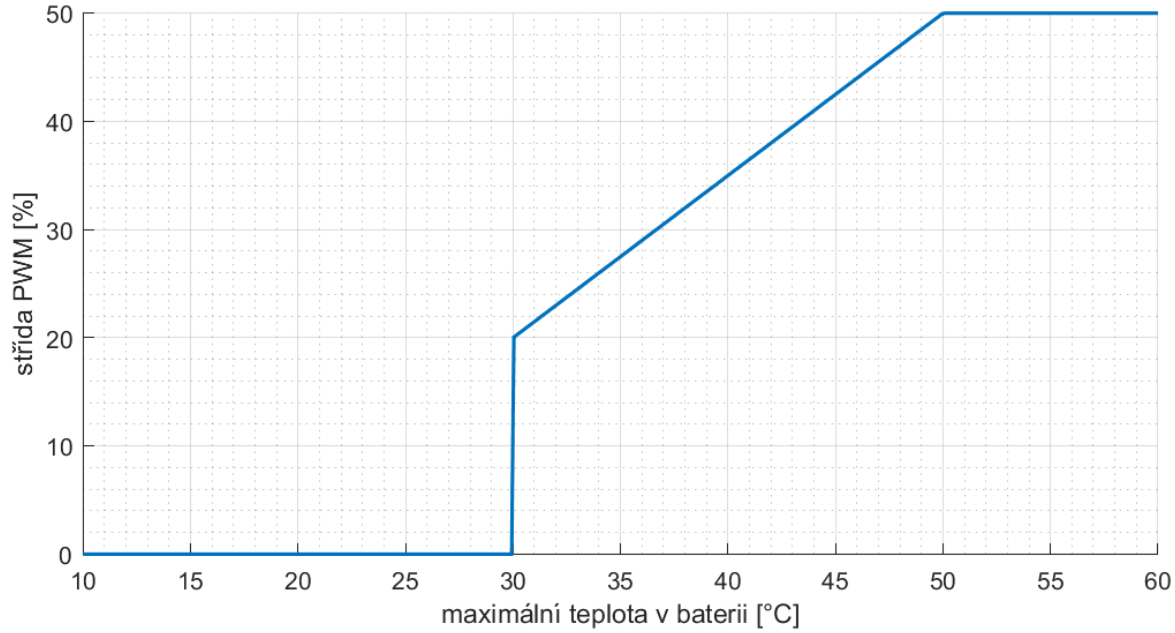
Obrázek 3.9: Zjednodušené schéma stavového automatu BMS



Obrázek 3.10: Zjednodušené schéma spínání relé

stavu BMS ok, nebo BMS charging. Závislost střídy PWM na teplotě nejvyšší naměřené teploty můžeme vidět na obrázku 3.11. Hodnota otáček je poté měřena pouze pro kontrolu řidičem, stavovým automatem není využívána pro přechod mezi stavy.

Autorem software pro BMS Dragonu e3 je Mário Harvan.



Obrázek 3.11: PWM pro ovládání ventilátoru v závislosti na maximální teplotě

4 Rozvržení HIL testování

Před začátkem testování je třeba řádně rozvrhnout, které části budeme testovat a které simulovat. Rozdělení je nutné udělat v návaznosti na dostupný HW a SW pro HIL. Před započetím vlastní diplomové práce byla proto vyvinuta snaha zajistit co nejlepší podmínky pro testování BMS.

HW platformou, kterou se podařilo pro účel této diplomové práce zařídit je dSPACE užívaný mechatronickou laboratoří. Dostupná je čtyřjádrová výpočetní jednotka DS1006, I/O karta DS2202 a FPGA modul DS5203. Systém je propojen se stolním počítačem, na kterém běží operační systém Windows XP. Model byl vytvořen v rámci programu Simulink ve verzi 2011b. Průběh simulace je kontrolován pomocí programu ControlDesk ve verzi 3.7.3. Všechny programy a HW, který byl využit pro účely DP jsou sepsány v příloze, 8. Snahy o přístup k HW speciálně určenému k testování BMS a aktualizaci SW pro dSPACE, které probíhaly před začátkem testování, selhaly. Bylo tedy potřeba přizpůsobit rozvržení HIL testů dostupnému HW.

Výslednou HIL soustavu, která byla využita, je možné vidět na obr. 4.1. Výpočetní jednotka DS1006 je určena k běhu kódu v reálném čase. Pomocí komunikační sběrnice PHS-bus je připojena k FPGA jednotce a I/O kartě [19]. Vstupně/výstupní karta DS2202 je modulem umožňujícím generovat signály běžně využívané v automobilovém průmyslu. Obsahuje tedy velké množství GPIO vstupů a výstupů, generátory PWM (pulzně šířková modulace), vstupy měřící signály PWM, ADC (analogově-digitální převodník), DAC (digitálně-analogový převodník) a základní komunikační sběrnice v automobilovém průmyslu, jako sériová komunikace UART nebo 2x sběrnice CAN.

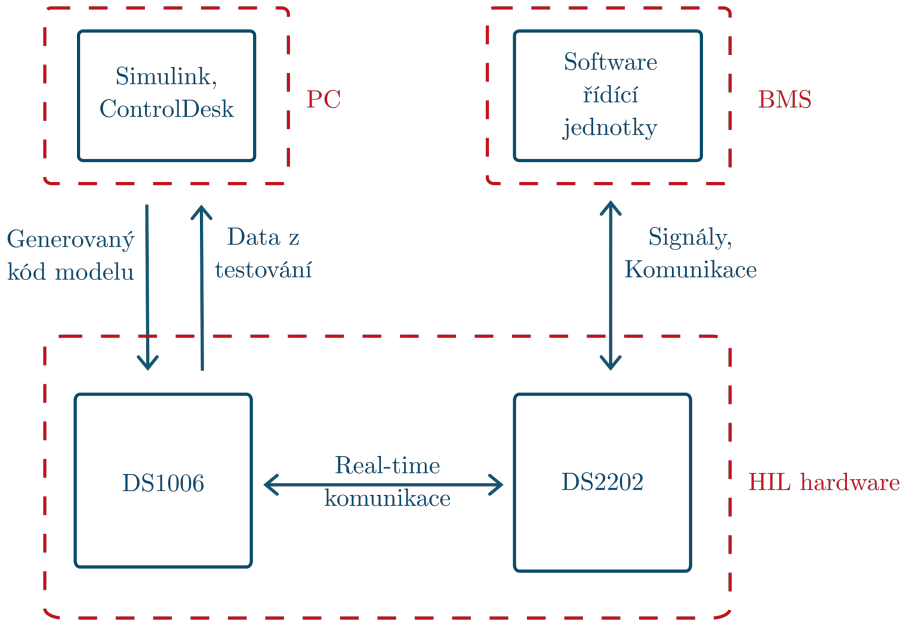
4.1 Rozvržení simulovaných a testovaných částí

Hlavním cílem, jak je zřejmé z popisu v kapitole 3, je otestovat software na BMS master. Potřebujeme tedy simulovat všechny kritické signály, které do něj vstupují. Postupně je proto projdeme a vyřešíme, zda jsme schopni je připojit, případně jakým způsobem to provedeme.

Z BMS hat probíhá komunikace po sběrnici I^2C. I^2C komunikační sběrnici karta DS2202 nepodporuje a nejsme proto jednoduše schopni simulovat probíhající komunikaci. Řešením problému může být připojení zařízení využívající tuto komunikaci a generování signálů na jeho vstupu. Je proto potřeba připojení BMS hat, kde je možné připojit se přímo na vstup ADC, které I^2C komunikaci využívá. Ostatní signály, které jsou posílány mezi oběma jednotkami, je možné simulovat a snímat až na výstupu, v konektoru. Pokud tedy připojíme BMS hat, odpadne nám starost s I^2C a získáme možnost ověřit také funkčnost analogových obvodů které se na DPS nachází.

LEM senzor je taktéž připojen přes sběrnici I^2C a vzniká nám zde stejný problém. Nabízí se tedy i stejné řešení, kterým je připojení DPS LEM k BMS master a využití přímo vstupu ADC.

BMS master dále využívá tři CAN sběrnice. K dispozici na kartě DS2202 máme ale



Obrázek 4.1: Schéma testovací soustavy pro HIL test BMS

pouze dvě. Je tedy potřeba jednu vynechat, nebo nějak nahradit. BMS slaves je nezbytné pro připojení měření na baterii a je tedy kritickou částí, kterou potřebujeme. Druhou CAN sběrnici musíme vybrat ze dvou sběrnic pro komunikaci se zbytkem formule. Po CAN gen posíláme méně kritických zpráv, které ovlivňují chování BMS a vybíráme proto BMS crit. Kritickou zprávou, kterou posíláme po CAN gen je zpráva signalizující připojení k nabíječce. Zpráva byla proto pro účel testování přesunuta na CAN crit. Ovlivníme tak mírně software BMS, zároveň ale získáme schopnost testovat důležitou část BMS. Funkčnost ostatních částí software bez přesunuté CAN zprávy je následně možné ověřit vrácením softwarových změn a spuštěním nového testu.

Pro ovládání ventilátorů, snímání jejich otáček a komunikaci s IMD je využíváno PWM signálů a GPIO pinů. Tyto signály jsme schopni generovat a nemáme tedy potřebu tyto části k BMS připojovat. Můžeme je simulovat.

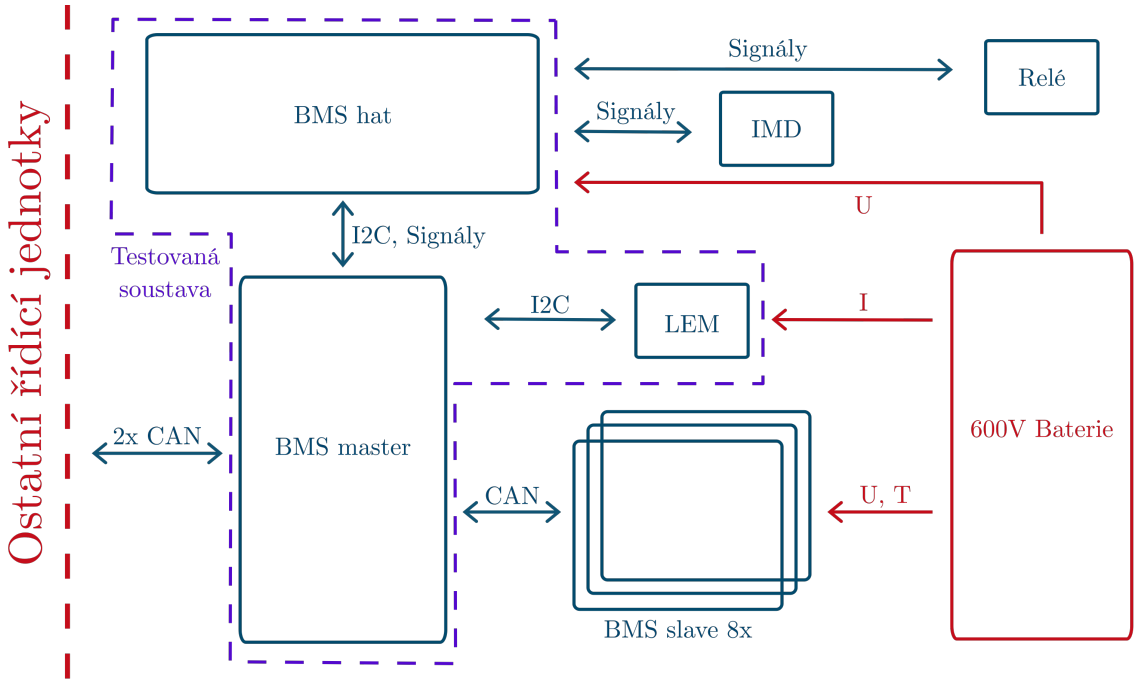
Z tohoto rozboru získáváme výsledné zapojení BMS pro HIL testování, které je názorně zobrazeno na obr. 4.2.

Dále se nabízí testovat samostatně pouze řídící jednotku BMS slave. Tohoto s dostupným HW bohužel nejsme schopni dosáhnout. K testování jsou zapotřebí izolované zdroje napětí, které simulují jednotlivé články baterie. Bez nich tato simulace nedává smysl, protože bychom vynechali klíčovou funkci DPS.

4.2 Zapojení BMS pro HIL testování

Jednotlivé vstupy a výstupy BMS mají svoje specifika a proto je v následující kapitole rozebereme tak, aby bylo možné pochopit elektrické zapojení jednotlivých částí. I/O karta DS2202 má například definovaný maximální proud, který je schopen dodat. Při připojování obvodů s větším odběrem proudu musíme na tento fakt myslet a případně vymyslet, jak daný výstup posílit.

Ukážeme také bloky RTI, které jsme pro konfiguraci vstupů a výstupů využili a pří-



Obrázek 4.2: Schéma testovaných a simulovaných částí

padně také nastavení, které bylo potřeba u jednotlivých bloků určit.

4.2.1 CAN sběrnice

CAN je robustní komunikační sběrnice, která je navržena pro aplikace i do prostředí, kde je předpokládáno rušení a ne příliš příznivé podmínky pro digitální komunikaci. Základem jsou dva vodiče, po kterých posíláme signály diferenčně. Prvnímu z dvojice říkáme CAN low (CANL), druhému CAN high (CANH). Bity označujeme jako dominantní a recesivní. U běžně používaného standardu ISO 11898-2 je při přenášení recesivního bitu rozdíl napětí mezi CANL a CANH 0V, nebo napětí blízké této hodnotě. Pokud je přenášen dominantní bit, běžné rozdílové napětí se pohybuje okolo 2V. Maximální přenosová rychlost je 1Mbit/s [20]. V případech, kdy chceme přenést velké množství dat, je možné využít verzi CAN-FD (Flexible data-rate - CAN s flexibilní rychlostí přenosu), která umožňuje rychlost až 5Mbit/s.

Pro účely týmu formule student byl v posledních letech vyvinut CANopen. CANopen je komunikační protokol běžící nad CAN sběrnicí. Implementuje funkce v rámci vyšších vrstev komunikace, které usnadňují práci s CAN. Jedním z rozdílů je například přiřazení identifikačního čísla každému uzlu na sběrnici. Následně jsme schopni adresovat konkrétní uzel a vyžádat data z jeho knihovny objektů (OD - object dictionary). Je také možné využívat zpráv určených pro monitorování stavu jednotlivých uzlů.

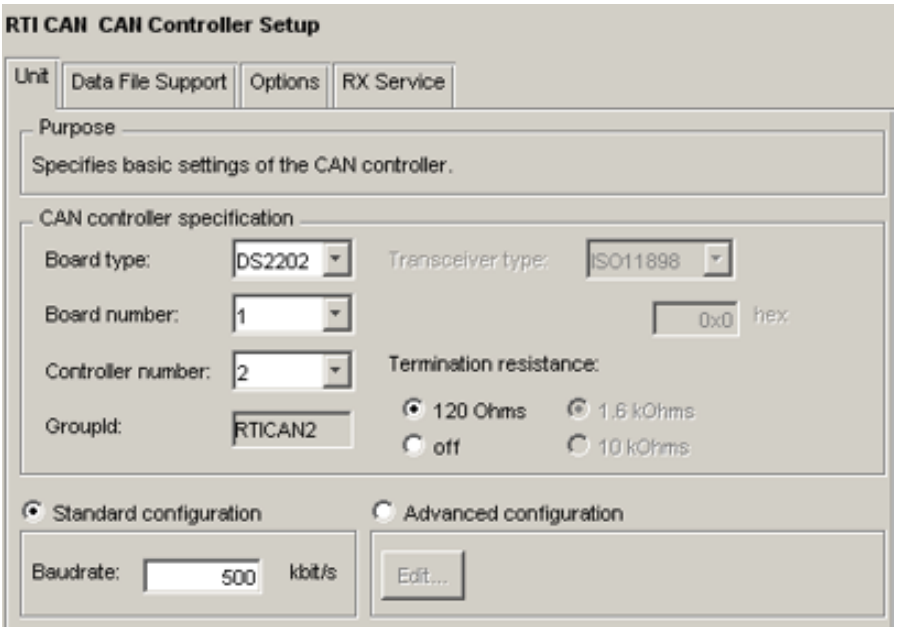
Abychom mohli pro dSPACE jednoduše definovat zprávy, které po CAN sběrnici posíláme, byl sestaven DBC soubor. Na obr. 4.3 vidíme jednu ze zpráv CAN slaves zapsanou v souboru DBC. Soubor obsahuje informace o každé zprávě. Definuje jejich délku, identifikační číslo, ale také maximální hodnotu [21]. DBC soubor byl nahrán do bloků RTI v simulinku a následně byla pro každý blok vybrána zpráva, kterou má přijímat, nebo vysílat.

V simulinku je dále potřeba upravit nastavení jednotlivých CAN sběrnic. Využíváme

```
BO_ 513 Slave1_voltage1: 8 Vector__XXX
        SG_ Voltage1 : 0|16@1+ (0.001,0) [0|65535] "V" Vector__XXX
        SG_ Voltage2 : 16|16@1+ (0.001,0) [0|65535] "V" Vector__XXX
        SG_ Voltage3 : 32|16@1+ (0.001,0) [0|65535] "V" Vector__XXX
        SG_ Voltage4 : 48|16@1+ (0.001,0) [0|65535] "V" Vector__XXX
```

Obrázek 4.3: Ukázka CAN DBC souboru

k tomu blok CAN Controller Setup, který musí být v modelu pro každou CAN sběrnici. CAN crit využívá sběrnici s baudrate 500 kbit/s, CAN slaves využívá rychlost 1 Mbit/s. Zvolíme také výstupní kartu DS2202 a příslušný CAN výstup. Zapneme také 120 ohmový terminační rezistor, kterým musí být na obou koncích CAN sběrnice.



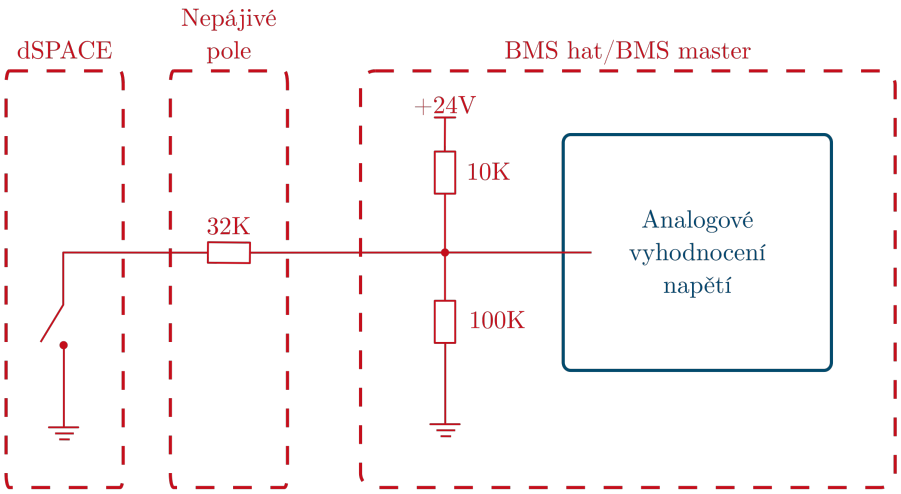
Obrázek 4.4: Ukázka nastavení CAN v Simulinku

4.2.2 SCS tlačítko

SCS (safety critical signal - signál kritický pro bezpečnost) je bezpečnostní funkce, která je využívána u kritických částí systému. U SCS signálu musíme být schopni podle pravidel detekovat chybu. Za chybu se považuje rozpojení obvodu, zkrat na zem, nebo zkrat na napájecí napětí [16]. U naší formule týmu TU Brno racing je signál považován za logickou jedna, nebo nula, pokud je v úzkém napěťovém pásu vylučujícím zem, napájecí napětí a další napěťové úrovně často vyskytující se v analogových obvodech.

Na schématu 4.5 máme zobrazeno zapojení tlačítka, které máme v testovacím zapojení nahrazeno tranzistorem simulujícím tlačítko. Pokud je tranzistor sepnut, je SCS signál uzemněn přes rezistor s hodnotou $R = 32k\Omega$. Tímto způsobem je vytvořeno napětí o velikosti 17V, které je v rozsahu vyhodnoceném jako logická jedna. Jakmile je tlačítko uvolněno, napětí stoupne mimo rozsah logické jedničky. Víme tedy, že tlačítko stlačeno není.

SCS tlačítka jsou v systému dvě. První nazýváme AMS reset (accumulator manage-



Obrázek 4.5: Zapojení SCS tlačítka pro HIL test

ment system - synonymum BMS). Tímto tlačítkem resetujeme chybu BMS. Pokud je rozpojeno SDC na BMS master stlačíme AMS reset a resetujeme tak tuto logiku. Pokud je BMS master ve stavu BMS error (viz. 3.5), je třeba podržet tlačítko na alespoň 1.4 sekundy. Tak resetujeme kromě logiky i procesor a dostaneme se do počátečního stavu stavového automatu (viz 3.5). Druhé tlačítko nazýváme IMD reset. Slouží k resetování logiky IMD. Pokud je IMD v chybě, je rozpojeno SDC. Opětovné sepnutí je možné jen pokud je signál IMD v pořádku a resetujeme logiku.

4.2.3 SDC final

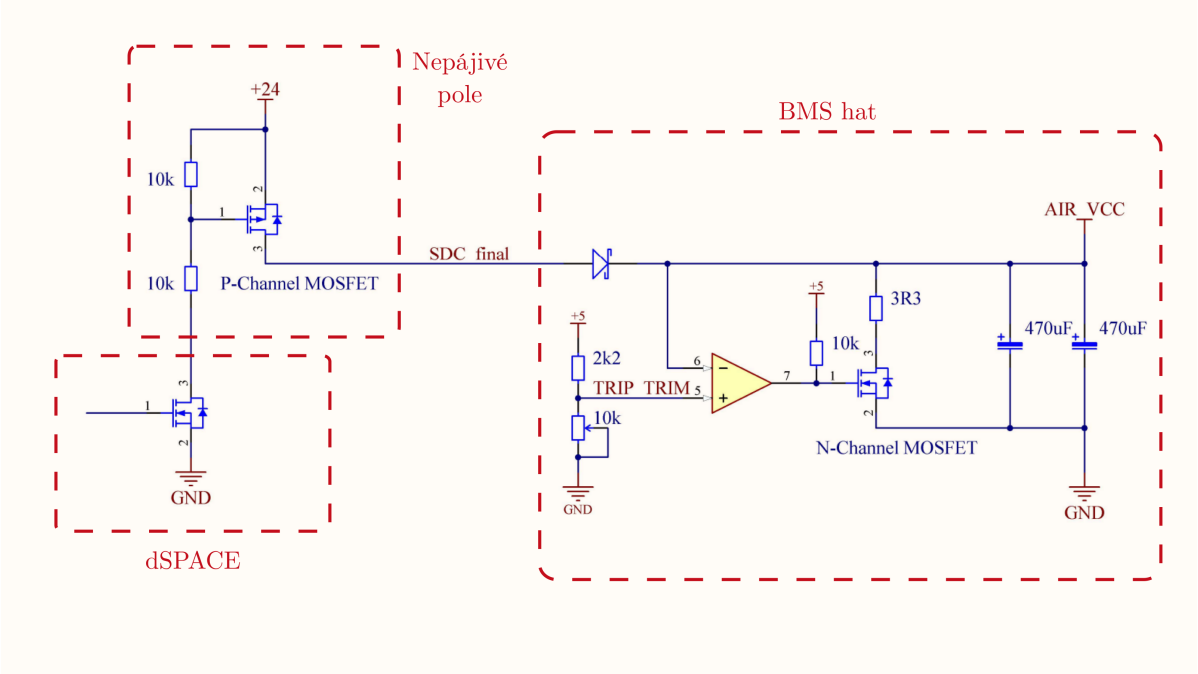
SDC final nazýváme konec SDC, který napájí izolační relé [22] (viz 3.8). Pro stabilizaci napětí jsou zde přidány kondenzátory. Pravidla formule student také definují, že před tím, než poklesne napájecí napětí na úroveň rozpojující relé (Dropout voltage - pro 24V relé je tato hranice 2,4V [22]), je potřeba náhle vybít kondenzátory a dosáhnout tak spolehlivého rozepnutí AIR. Obvod, který náhle vybije kondenzátory, pokud napětí klesne pod 3V, lze vidět na obr. 4.6.

Vybití je zaručeno komparátorem, který sepne tranzistor. Tímto je připojen paralelně ke kondenzátorům rezistor s hodnotou $R = 3,3\Omega$. S tím ale souvisí problém, který nastává při připojování SDC final. Je potřeba překonat napěťovou hranici 3V, do které je obvod vybíjen. SDC final je proto potřeba připojit přes tranzistor, který nám umožní dodat vysoký proud při jeho připojování. Tranzistor spínáme pomocí signálu z dSPACE a připojujeme tak SDC final přímo na zdroj, kterým testovanou soustavu napájíme.

4.2.4 Detekce a spínání relé

BMS ovládá tři izolační relé. Všechny tři jsou zapojeny stejným způsobem a zaměříme se tedy pouze na popis spínání a snímání jednoho z nich.

Relé, které připojujeme k BMS je spínáno pomocí tranzistoru (obr. 4.7). Pokud je



Obrázek 4.6: Zapojení SDC final pro HIL test

tranzistor sepnut, proud teče z napájení relé přes cívku, která sepne kontakty. Tuto část obvodu potřebujeme snímat, abychom zjistili, kdy BMS seplo relé. Vzhledem k tomu, že nemáme připojené AIR, musíme připojit pull-up rezistor, který nám zvedne napěťovou úroveň v době, kdy je tranzistor rozpojen. Napěťovou úroveň snímáme pomocí GPIO pinu na vstupní kartě.

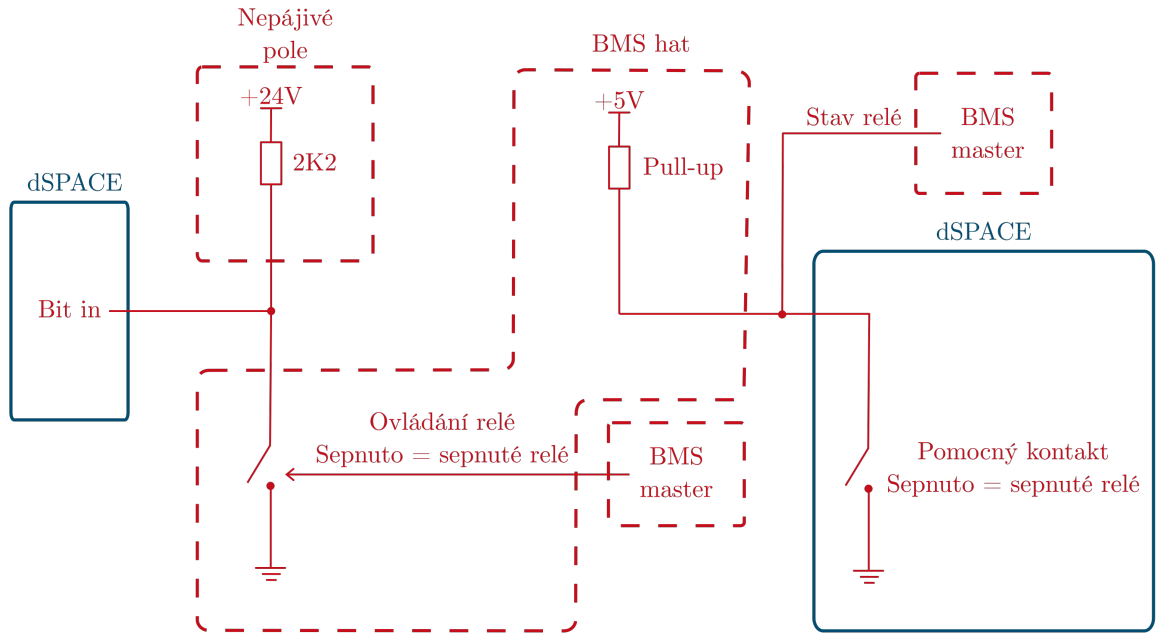
Druhou částí obvodu, kterou můžeme vidět na schématu 4.7 (na pravé straně), je kontrola, zda je relé sepnuto. Pokud je sepnuto, spojí se pomocný kontakt a stáhne napěťovou úroveň na pomocném obvodu BMS hat na zem. Stejným způsobem je tento obvod simulován pomocí dSPACE. Byl využit GPIO výstup karty DS2202 tak, aby stahoval napětí připojením výstupu na zem.

4.2.5 Simulace napětí baterie a meziobvodu měniče

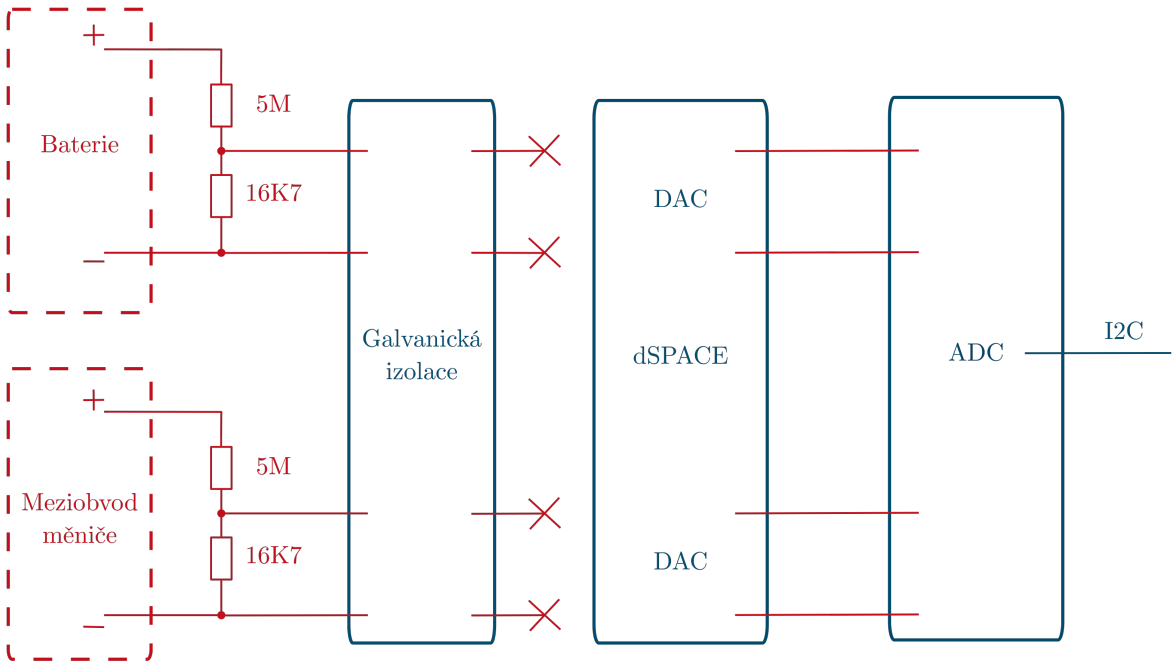
Pro simulaci napětí baterie v celém jejím rozsahu nemáme dostupný hardware. Není tedy možné generovat napětí přímo na vstupu DPS, ale je možné připojit se přímo na vstup ADC. Zde již napětí, po rozdělení napěťovým děličem, dosahuje maximální hodnoty 2V, které je možné generovat pomocí DAC na dSPACE. Přímo na DPS byl pro tento účel rozpojen signál od izolátorů a na ADC byly připojeny diferenční výstupy DAC z výstupní karty. Výsledné zapojení je zobrazeno na obr. 4.8.

4.2.6 Simulace měření proudu

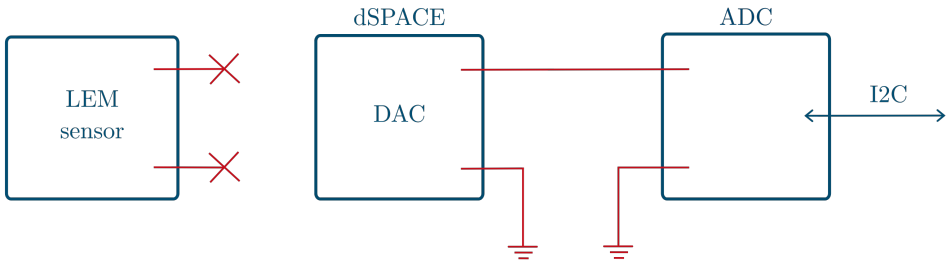
Na DPS určené pro zpracování výstupu z LEM senzoru byl využit stejný přístup, jako v případě simulace napětí na baterii. Dostupný dSPACE neumožňuje simulovat proud, který bychom mohli LEM senzorem měřit. Senzor byl proto z DPS odpojen a simulován byl jeho výstup, kterým je nízké napětí do 2,5V. Zapojení je zobrazeno na obrázku 4.9.



Obrázek 4.7: Zapojení obvodu ovládaní relé



Obrázek 4.8: Zapojení měření napětí baterie a meziobvodu měniče

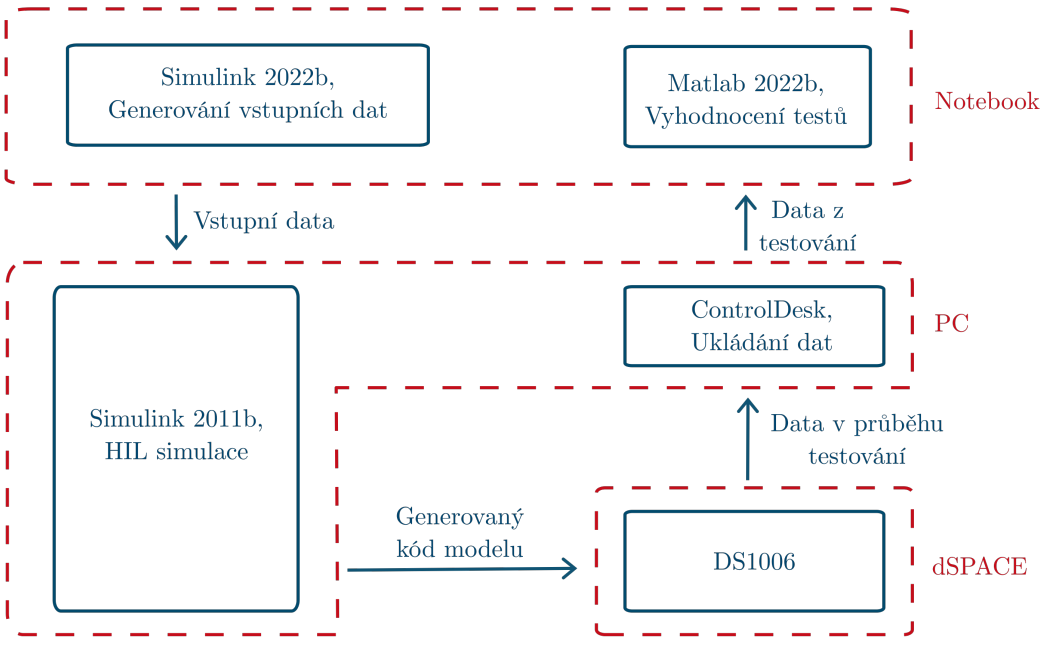


Obrázek 4.9: Zapojení měření proudu

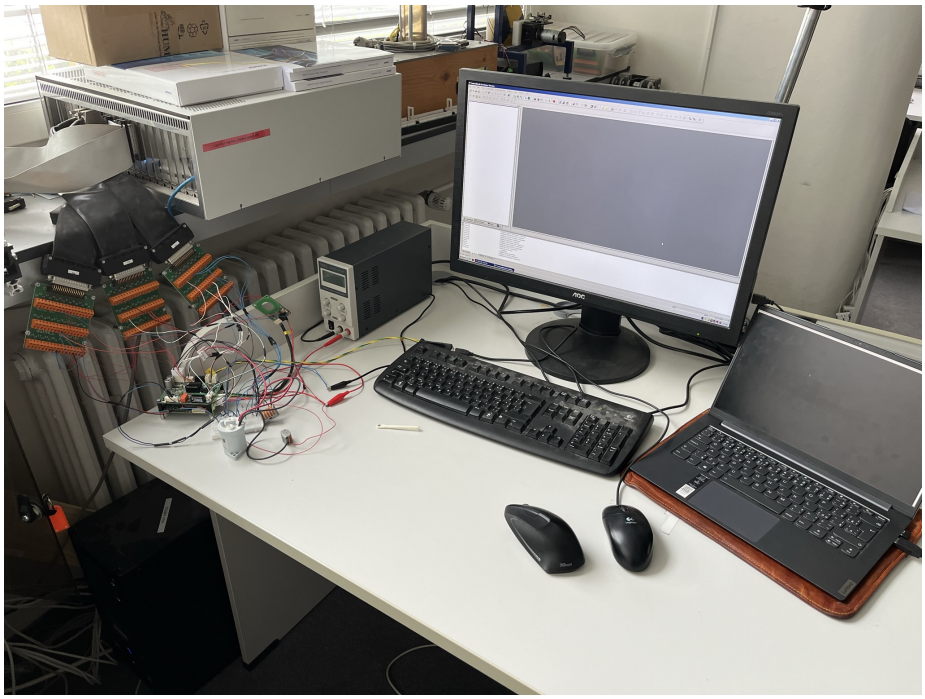
5 Návrh HIL testů BMS

Po správném připojení BMS je čas navrhnout HIL testy, které otestují co nejvíce funkcí. Zaměření testů bylo zvoleno tak, aby se otestovaly především bezpečnostní funkce a bylo možné vyhodnotit, jestli chování splňuje pravidla soutěže formule student a bezpečnostní parametry potřebné pro bezpečný provoz. Pro návrh automatizované sekvence testů bylo zvoleno rozšíření Simulinku, Stateflow. Stateflow je nejčastěji využíván pro návrh stavových automatů. Pokud ovšem definujeme přechody mezi stavy na základě času simulace, jsme schopni jednoduše generovat testovací sekvence.

Během počáteční fáze návrhu HIL testů nebylo jasné, zda bude možné testovat na verzi Simulinku 2022b, nebo bude muset být využit Simulink 2011b. Testovací sekvence byly proto v počátku navrhnuty na verzi 2022b. Protože se však nepodařilo včas zprovoznit novou verzi, byl navrhnut postup, jak jednoduše dostat testovací sekvenci na starší verzi. Data testovací sekvence jsou pro celý test uložena do souboru .mat ve formě struktury s časovým záznamem (structure with time). Následně je soubor nahrán do pracovního prostoru Simulinku 2011b a vložen do simulace ve formě vstupních dat. Výstupní data testů jsou poté v ControlDesku uloženy a nahrány zpět do Matlabu ve verzi 2022b, kde dochází k vyhodnocení. Vyhodnocení je dále popsáno v kapitole 6. Na obr. 5.1 můžeme vidět schématicky popsán testovací postup a přenos dat mezi zařízeními. Mezi počítačem a notebookem byl pro přenos souborů využit externí disk. Na obrázku 5.2 potom vidíme zapojení BMS při testování v laboratoři.



Obrázek 5.1: Testovací postup



Obrázek 5.2: Zapojení BMS pro testování v laboratoři

Pro Simulink 2011b byly následně připraveny části, které je potřeba testovat v uzavřené smyčce. Tyto části modelu není možné nahradit daty vygenerovanými, protože by je nebylo možné ovlivňovat během HIL testu výstupy z BMS. Takovými částmi jsou například izolačních relé a nabíjení meziobvodu měniče, které jsou ovládány z BMS.

Vzhledem k tomu, že byly testovací sekvence navrženy v nové verzi, je návrh připraven pro další využití do budoucna. Se zprovozněním nové verze Simulinku bude stačit připojit signály na vstupní a výstupní bloky, mírně upravit základní signály a bude možné dále testovat.

5.1 Model pro generování vstupních dat

Testovací sekvence je definována v rámci souboru HILTestSequence.slx. Pro spuštění Simulink modelu byl připraven skript v Matlab, main.m. V rámci skriptu jsou připraveny všechny proměnné potřebné ke spuštění programu, ale také odkazy na další soubory a modely, na které se hlavní model odkazuje. V main.m je možné zvolit variantu simulace. Byly definovány 3 proměnné, které nastavují jaká část modelu je simulována a co je výsledkem simulace.

První věc, kterou můžeme měnit, je typ testu, který chceme spustit. Máme na výběr 3 základní testy, které jsou připraveny k použití. Pokud nastavíme proměnnou Test = 1, spustíme základní test, viz kapitola 5.4. Test = 2 spustí dlouhodobý test (viz kapitola 5.5) a Test = 3 spustí test chybových stavů (viz kapitola 5.6).

Druhý parametr, Battery_RC, nastavuje, jaký model baterie využijeme pro simulaci. Hodnota 1 nastaví model se 136ti články a jednou RC časovou konstantou [23], [24], hodnota 2 nastaví model se 136ti články a dvěma časovými konstantami. Nakonec hodnota 3 nahradí simulaci celé baterie simulací jednoho článku. Ten nahrazuje celou baterii a má jednu časovou konstantu RC. Poslední varianta s jedním článkem byla zvolena jako finální pro všechny prezentované výsledky. Její hlavní výhodou je kratší čas potřebný

pro výpočet simulace. První dvě varianty modelu jsou výpočetně velmi náročné a byly sestrojeny Jakubem Lysákem, který model vytvářel v rámci své bakalářské práce [24]. Třetí model byl vytvořen v rámci této diplomové práce na základě dat z předchozích modelů.

Třetím parametrem (Mode) volíme, zda chceme generovat vstupní data pro HIL test, nebo chceme provést simulaci na modelu BMS. Model BMS je popsán v kapitole 6.2.1. Jak pro vstupní data, tak pro simulaci modelu jsou data uložena do souboru, který můžeme následně využít pro přenos dat na stolní počítač s HIL simulací i pro vyhodnocení testů (viz kapitola 6).

Pro definování časování testovací sekvence BMS bylo využito dvou přístupů. Změnu stavu provádíme buď v určitém čase simulace, nebo po uplynutí určitého času ve stavu předchozím. Tato kombinace dvou přístupů byla zvolena proto, aby bylo možné využít přístup, který dává pro danou sekvenci větší smysl. Pokud definujeme například délku stlačení tlačítka, chceme definovat jak dlouho je tlačítko stlačeno. Pokud na druhou stranu chceme v nějakém čase zapnout TS, je lepší definovat čas, ve kterém TS zapínáme. Pomůže nám to zvláště při vyhodnocování testů, kdy jednoduše zjistíme z průběhů sbíraných dat za jak dlouho je TS sepnuto.

5.2 Model HIL simulace

Na stolním počítači v laboratoři byl vytvořen model pro HIL simulaci. Dříve než popíšeme samotný model, je potřeba definovat nastavení samotného modelu tak, abychom byli schopni generovat C kód pro dSPACE. V nastavení modelu (configuration parameters) je důležité nastavit správně řešič. Na začátku byl nastaven řešič ode3 (Bogacki-Shampine), který měl délku kroku 0.001 sekundy. Procesor běžící v reálném čase ovšem nemá dostatek výkonu aby počítal kroky s frekvencí 1000 Hz. Proto byla upravena frekvence na hodnotu 100 Hz. Tato hodnota je dostačující pro účely našeho testování, kde nejsou simulovány vysokofrekvenční děje. Také bylo potřeba nastavit generování C kódu. V nastavení cílového souboru (Code generation/system target file) je potřeba nastavit "rti1006.tlc". Poté může být generován kód pomocí funkce incremental build.

Dále se zaměříme na popis ostatních částí modelu. Vstupní data jsou importovány pomocí bloku "From Workspace". Je tedy nutné před kompilací modelu na kód nahrát soubor s vygenerovanými daty do pracovního prostoru Matlabu. Tyto data jsou v otevřené smyčce a nemohou být ovlivněny výstupy z BMS.

V uzavřené smyčce byl vytvořen model pro izolační relé a také model ventilátorů. Model pro izolační relé byl vytvořen za pomoci popisu matematickými rovnicemi. Pokud jsou izolační relé rozpojená, nebo pouze jedno z nich sepnuté, nabíjející a vybíjející proud kondenzátorů je nulový. Pokud je rozpojené relé PRE a NEG, nabíjecí proud je vypočítán pomocí rovnice 5.1 (hodnota nabíjecího rezistoru je ($R_{nabijeni} = 1710\Omega$). Je-li sepnuto relé POS a NEG, napětí na meziobvodu je rovno napětí na baterii. Krátký přechodový stav při nabíjení kondenzátoru můžeme zanedbat, protože sepnutím kladného i záporného pólu baterie vytvoří spojení s velmi malým odporem a tím i velmi nízkou časovou konstantou. Je-li sepnutý DIS tranzistor, vybíjíme kondenzátory přes vybíjecí odpor $R_{vybijeni} = 7500\Omega$. Vybíjecí proud je poté 5.2. Celkový proud je vypočítán pomocí rovnice 5.3, napětí meziobvodu pak určíme dle rovnice 5.4. Kapacita meziobvodu měniče je $C_{meziobvodu} = 191\mu F$

$$I_{nabijeni} = \frac{U_{baterie} - U_{meziobvodu}}{R_{nabijeni}} \tag{5.1}$$

$$I_{vybijeni} = \frac{U_{meziobvodu} - U_{baterie}}{R_{vybijeni}} \tag{5.2}$$

$$I_{celkove} = I_{nabijeni} - I_{vybijeni} \tag{5.3}$$

$$U_{meziobvodu} = \frac{1}{C_{meziobvodu}} \int I_{celkove}\, dt \tag{5.4}$$

U modelu ventilátoru byla zanedbána dynamika a byla využita data z katalogového listu výrobce [25]. Ventilátor je ovládán pomocí PWM z BMS, tuto PWM měříme a na základě střídy určujeme rychlost ventilátoru. Ventilátor generuje PWM v závislosti na rychlosti otáčení. Tuto zpětnou vazbu ve formě PWM signálu naopak měří BMS master, který vyhodnocuje rychlost otáčení ventilátoru.

Maximální rychlost otáčení ventilátoru je $n_{max} = 16500 ot/min$. Otáčky jsou lineárně závislé na střídě, kterou ventilátor ovládáme. Nelinearitu, která nastává v nízkých otáčkách zanedbáváme. Ventilátor v těchto otáčkách neplánujeme provozovat. Využijeme tedy vzorec 5.5. Frekvenci PWM, kterou v simulaci generujeme pro měření BMS master, vypočítáme pomocí vzorce 5.6.

$$n = n_{max} \cdot D \tag{5.5}$$

Kde D je střída PWM pro ovládání ventilátoru.

$$f_{PWM} = \frac{2 \cdot n}{60} \tag{5.6}$$

Model LEM byl vytvořen na základě rovnice sestavené autorem DPS pro LEM senzor, kterým je člen týmu, Nikita Kolyuka. Rovnice 5.7 popisuje výpočet napětí na výstupu LEM senzoru U_{out}, které generujeme v závislosti na velikosti měřeného proudu I, za pomocí výstupní karty.

$$U = 1,875 + \frac{I}{240} \tag{5.7}$$

Zbylé části modelu tvoří vstupní a výstupní bloky pro dSPACE. Pro každý signál bylo potřeba vybrat správný vstupní, nebo výstupní blok. Poté je potřeba blok správně nastavit tak, aby na výstupu byla hodnota, kterou očekáváme. Celé zapojení, včetně vybraných periferií a výstupních pinů využitých pro daný signál je dostupné v příloze, viz 8.1.

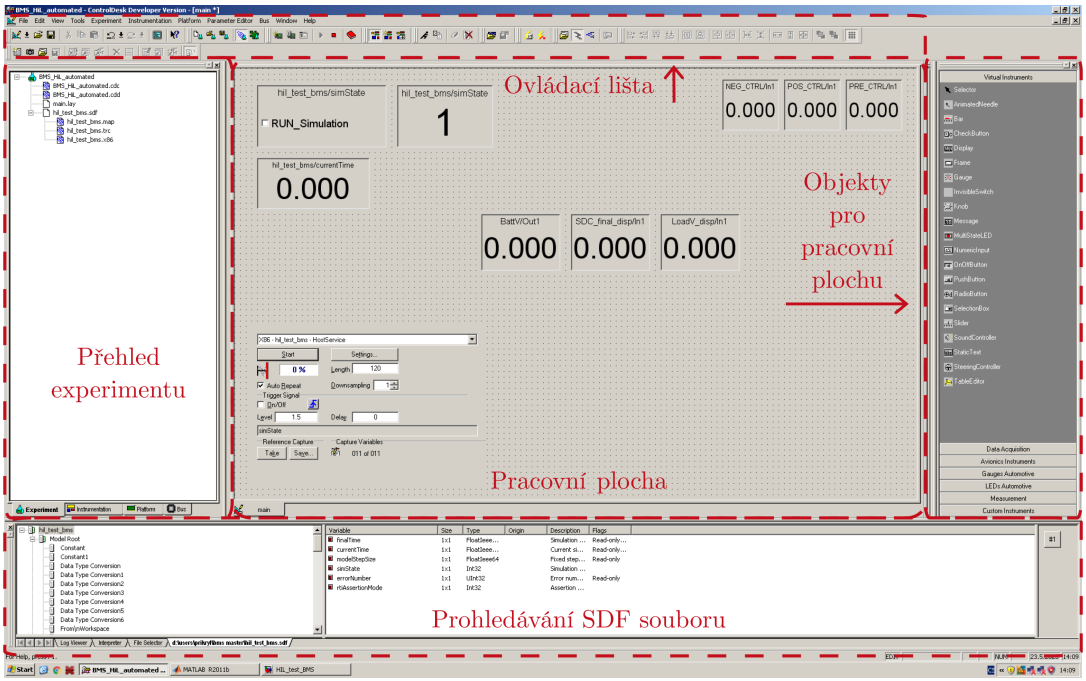
Byl také vytvořen druhý model (try_can.slx), který je možné kompilovat na C kód a můžeme jej ručně ovládat z ControlDesku pomocí nástrojů, které má tento program k

dispozici. Tento model je ideální pro kontrolu nastavení výstupních bloků. Můžeme také lépe demonstrovat, jak generování signálů funguje a jak změna proměnných ovlivňuje výstupy dSPACE.

5.3 ControlDesk

ControlDesk využíváme pro kontrolu nad hodnotami simulace v rámci HIL testování. V krátkosti proto popíšeme prostředí aplikace a experimenty, které byly vytvořeny za účelem testování BMS.

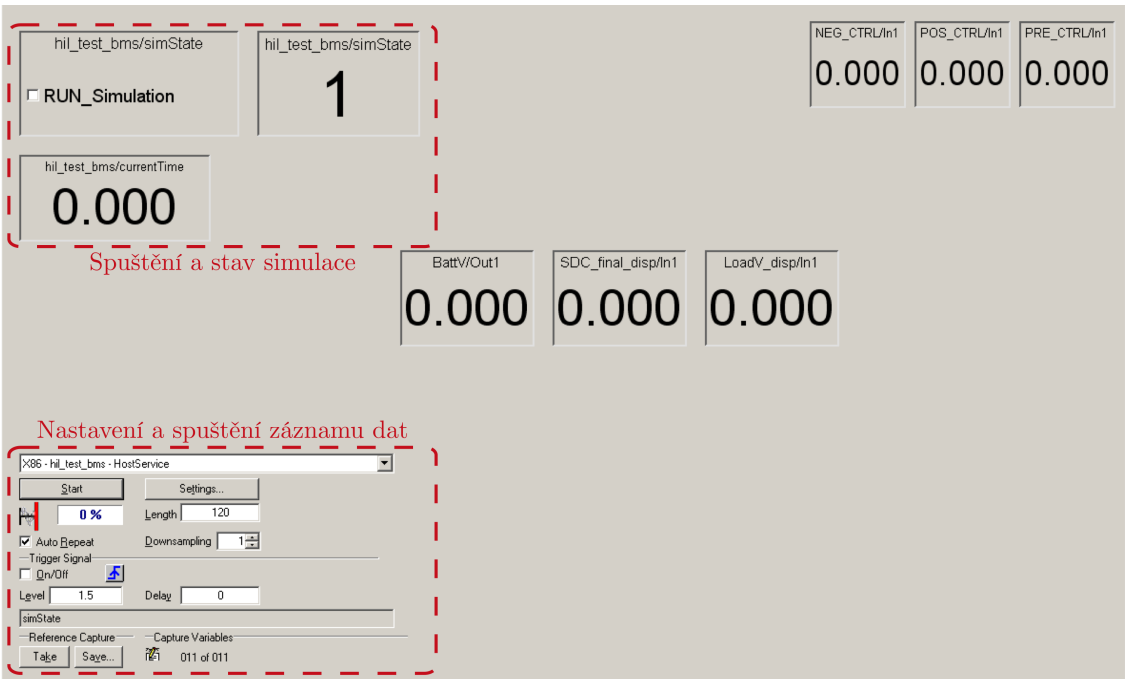
Na obr. 5.3 vidíme prostředí v módu pro úpravu (edit mode). V tomto módu je možné přidávat a upravovat jednotlivé objekty na pracovní ploše. Pokud chceme sledovat testování a zasahovat do jeho průběhu, přepneme se do módu animace (animation mode). Ikony pro přepínání se nachází na ovládací liště programu. Na pravé straně se nachází přehled objektů, které můžeme využít. Je zde například display, který umožňuje zobrazit hodnotu proměnné, nebo nejrůznější tlačítka umožňující manipulaci s hodnotami. Na levé straně se nachází přehled experimentu. Je možné se také přepnout na jiné záložky zobrazující např. konfiguraci připojené sestavy od dSPACE (záložka Platform). Ve spodní části okna se nachází prohledávání SDF souboru umožňující manipulaci s proměnnými.



Obrázek 5.3: ControlDesk v módu pro úpravu

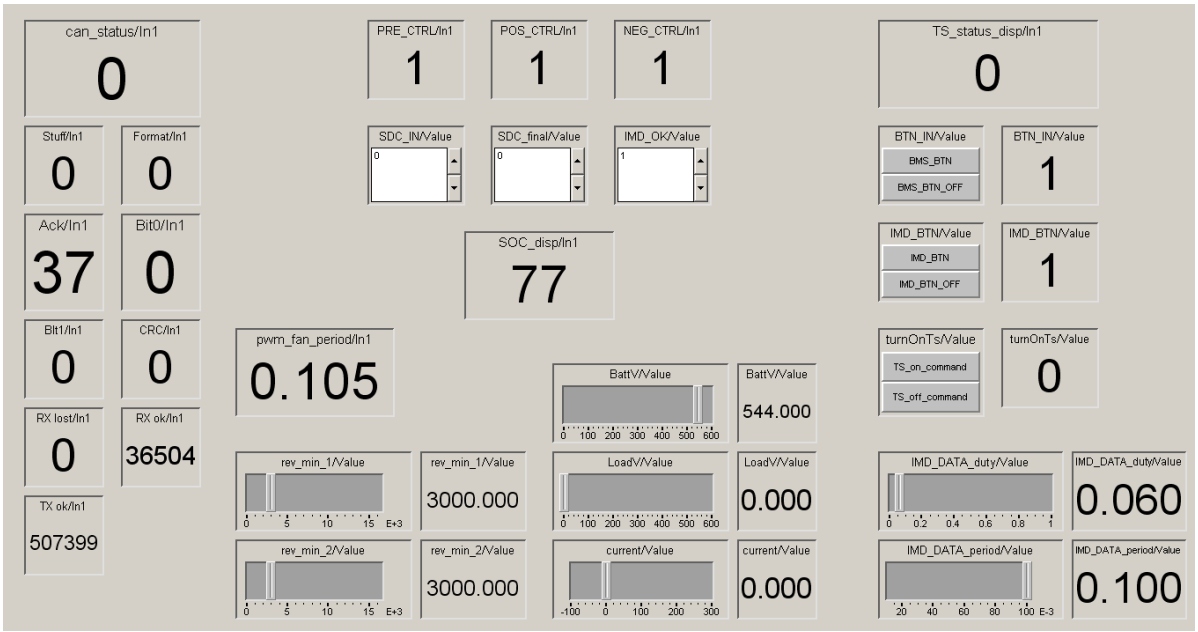
V rámci ControlDesku byly vytvořeny dva experimenty. První, BMS_HIL_automated.cdx, je optimální pro automatizovaný test. Zobrazuje přehled základních proměnných a umožňuje ovládání testu BMS. Po kompilaci Simulink modelu BMS_HIL_automated.slx se program nahraje do ControlDesku v pozastaveném stavu (možné změnit v Simulinku před spuštěním kompilace kódu - configuration parameters/Code generation/RTI simulation options). Chceme-li zaznamenat data z testování, zapneme ukládání tlačítkem Start na objektu pro záznamu dat (viz 5.4). Následně můžeme spustit simulaci potvrzením pole u RUN_Simulation. Změníme tak stav proměnné simState a test, včetně záznamu dat, je

spuštěn.



Obrázek 5.4: Pracovní prostor pro automatizovaný test

Druhý experiment, BMS_HIL_manual.cdx, nám poskytuje volnost v nastavení dat a je optimální pro kontrolu funkce všech vstupů a výstupů, případně pro experimentování s komunikací po sběrnici CAN. Není vhodný pro HIL testování. Pro využití tohoto experimentu je potřeba kompilovat kód z Simulink modelu try_can.slx. Na obr. 5.5 vidíme připravené objekty v pracovním prostoru, kterými měníme hodnoty proměnných.

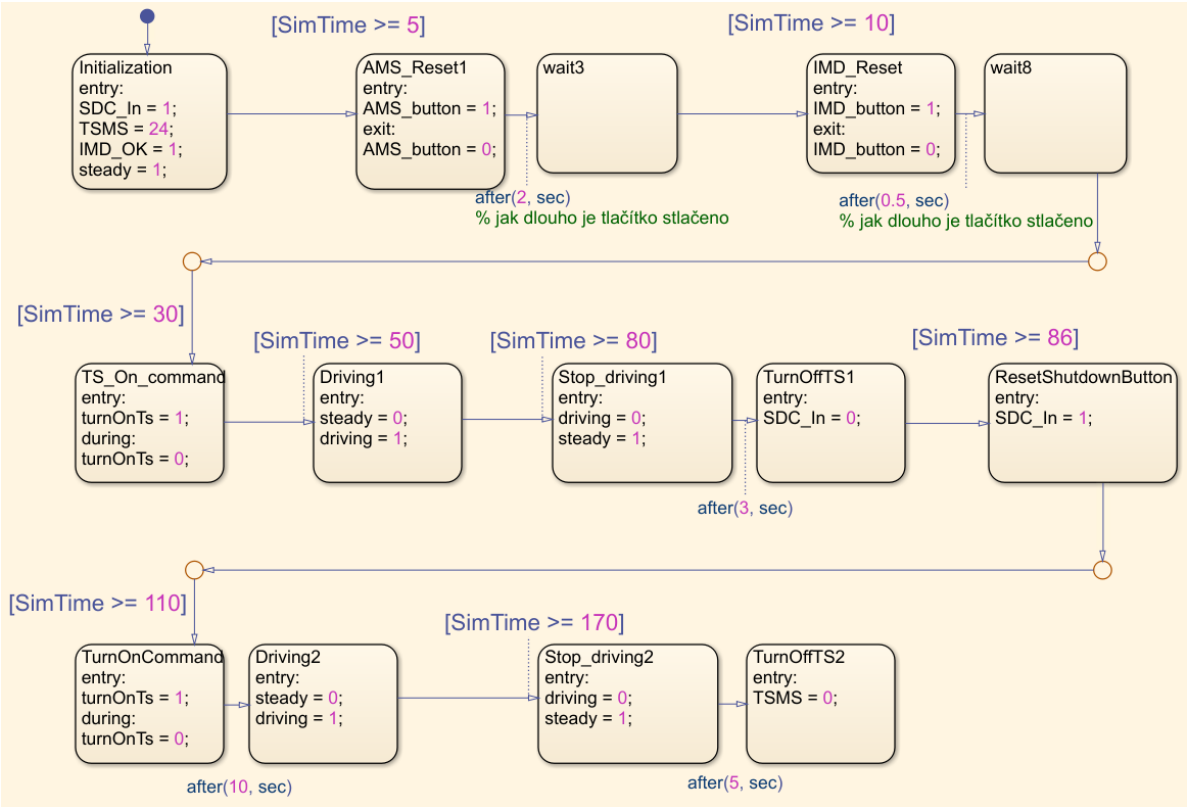


Obrázek 5.5: Pracovní prostor experimentu s ovládacími prvky

5.4 Základní test

Základní test je navržen tak, aby mohla být vyzkoušena základní funkčnost BMS. Test je proto inspirován běžným průběhem zkoušení funkčnosti formule a ladění jejího chování během testovací fáze sezóny.

Nejdříve je simulováno zapnutí formule. Je proto nastaven vstup SDC do logické jedna, zapnut trakční systém spínačem TSMS (traction system master switch - spínač pro zapnutí napájení izolačních relé), zkontrolována správná funkce IMD a stav formule nastaven do klidu (formule stojí na místě). Následně proběhne reset BMS. Reset provedeme stlačením AMS reset tlačítka na dobu delší než 1,4 sekundy. Zajistíme tím počáteční stav BMS v rámci stavového automatu. Pokud bychom reset neprovedli, nebyli bychom schopni říct v jakém stavu BMS je. Stav před restartováním závisí na tom, jaké hodnoty mají vstupní signály před spuštěním testu. Dále je resetováno IMD, tím je zajištěno sepnutí relé odpojující SDC. Jakmile je vše v pořádku, může být zapnut trakční systém formule příkazem turnOnTs, který posíláme po sběrnici CAN crit. Příkaz je nastaven tak, že je posílán během stisku tlačítka řidičem. V simulaci tedy stačí poslat turnOnTs jednou (posíláno jeden simulační cyklus). Zapnutí TS za optimálních podmínek trvá 3,2 sekundy. Je proto potřeba chvíli počkat. Následně je možné spustit odběr proudu z baterie. V simulaci tedy nastavujeme proměnnou driving do stavu jedna. Po konci ježdění je vypnuto TS odpojením SDC. V tomto okamžiku bychom při kontrole funkce formule šli k monopostu a kontrolovali stav, nebo přenastavovali jeho nastavení. Zapnutí TS provádíme ještě jednou. Provedeme další krátkou jízdu a vypínáme TS pomocí TSMS. Celá testovací sekvence je zobrazena na obr. 5.6.

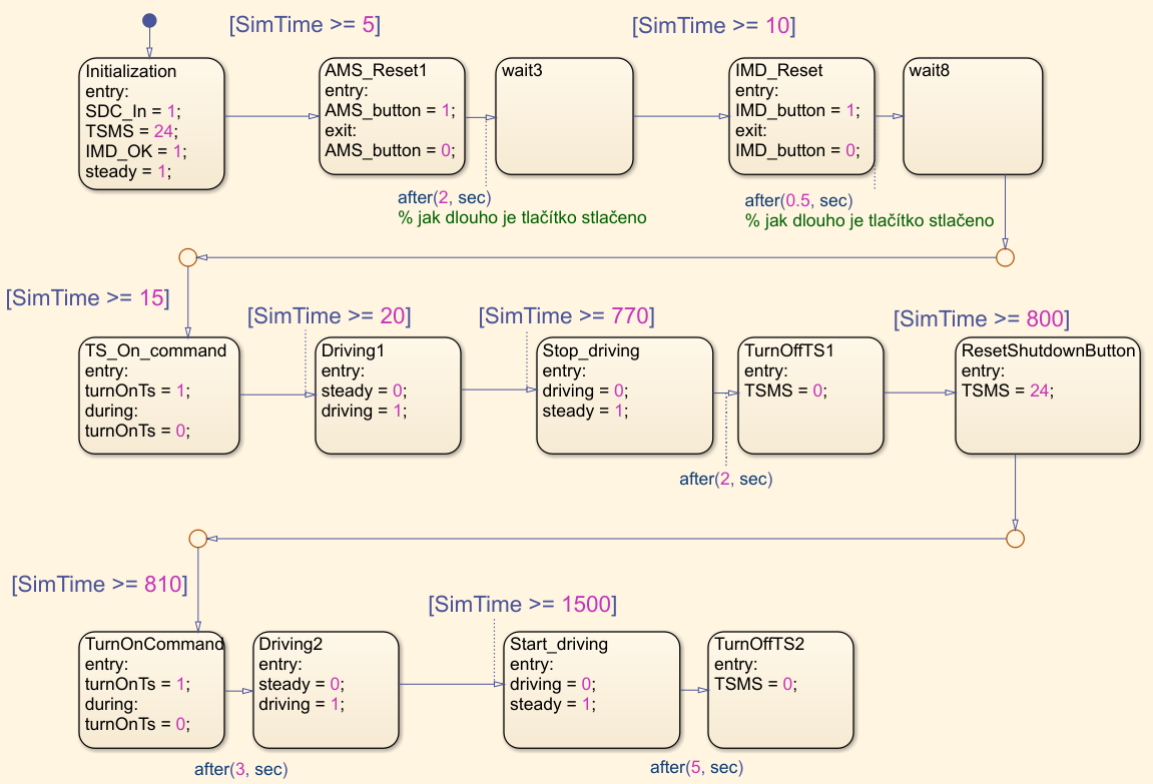


Obrázek 5.6: Testovací sekvence - základní test

5.5 Dlouhodobý test

Dlouhodobý test je proveden podobným způsobem jako test krátkodobý. Velkým rozdílem je však jeho zaměření. Úkolem je otestovat funkci BMS z dlouhodobého hlediska. To znamená, zda nevypadává komunikace s některým ze zařízení, nebo nejsou problémy např. se stoupající teplotou během dlouhého ježdění formule. Test je inspirován nejdelším závodem v rámci soutěže, endurance, kde se v polovině závodu vymění řidiči a je tedy potřeba zastavit.

Průběh testovací sekvence je zobrazen na obr. 5.7 a oproti krátkodobému testu vypínáme v obou případech TS pomocí TSMS a úseky ve kterých je odebírán proud z baterie, tzn. formule jezdí, jsou mnohonásobně delší.



Obrázek 5.7: Testovací sekvence - dlouhodobý test

5.6 Test chybových stavů

Test chybových stavů byl koncipován jako test, ve kterém chceme v krátkém čase vyzkoušet co nejvíce chyb, které by teoreticky mohly v systému BMS nastat. Využíváme zde toho, že se nemusíme o systém bát a můžeme experimentovat i se stavy, které bychom při nasazení v baterii nemohli, nebo z hlediska bezpečnosti nechtěli zkoušet.

Během testování je vyzkoušeno například co se stane pokud: je zapnuté TS a resetujeme BMS, ihned po resetu BMS zapneme TS, vypadne signál z IMD, pomocné piny u izolačních relé nezobrazují správné hodnoty, nebo pokud se objeví chyby v naměřených hodnotách teplot a napětí. Pokud chceme po chybě testovat další ochranu, musíme BMS restartovat. Schéma testu nelze zobrazit v práci a je přiloženo v elektronické příloze.

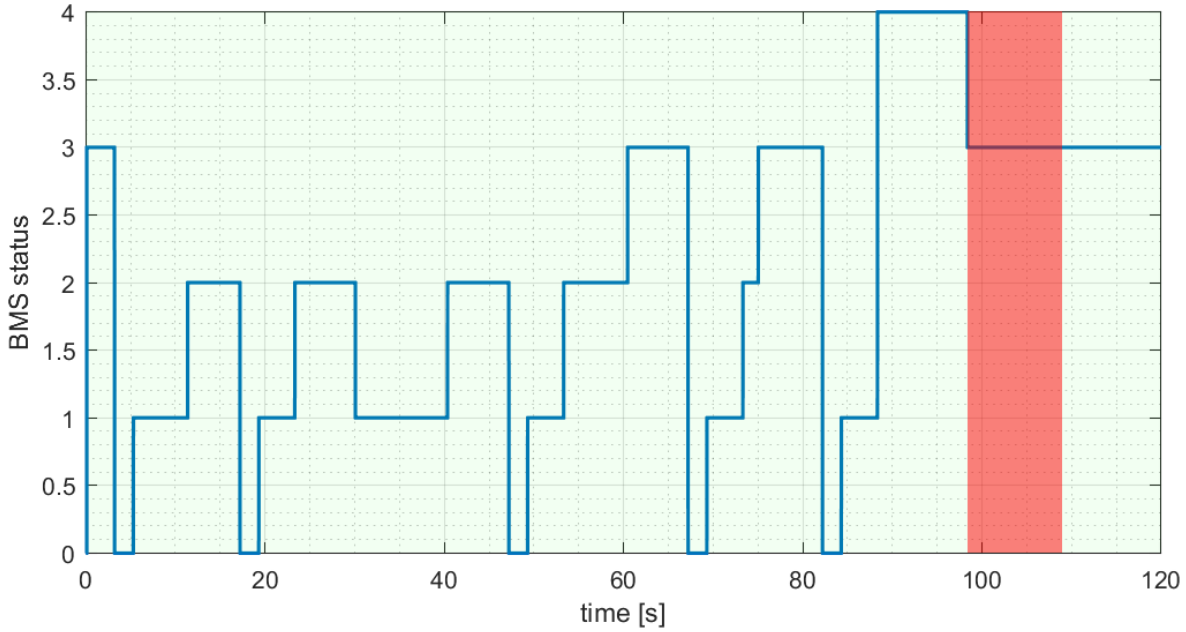
6 Vyhodnocení HIL testů

Získaná data z HIL simulace je potřeba dále zpracovat, abychom mohli určit, zda byl test proveden úspěšně, nebo v nějaké části testu řídicí jednotka selhala. V Matlabu 2022b jsme pro tento účel připravili skripty vyhodnocující test. Byly zvoleny dva přístupy. Jeden závislý na konkrétním testu, který vyhodnocujeme (viz 6.1), druhý je možné využít pro jakýkoli test (viz 6.2), i pokud jej upravíme, nebo definujeme test vlastní.

6.1 Vyhodnocení přednastavených testů

Testy, které byly přednastaveny v Simulinku odpovídají známému chování BMS, které je očekáváno. Proto je možno kontrolovat, v závislosti na čase simulace, zda je BMS ve správném stavu. Algoritmus vyhodnocení je určen pro kontrolu daného testu. Pro jednoduchost vyhodnocujeme pouze stav BMS. Ostatní proměnné jsou úzce navázány na stav a mohou být otestovány v rámci obecného vyhodnocení popsaného v kapitole 6.2.

Vyhodnocení je provedeno grafickým zvýrazněním jednotlivých částí testu podle toho, jestli jsou v pořádku, nebo nastala chyba. Části, které neodpovídají očekávání, jsou zvýrazněny barvou červenou. Části, které očekávání odpovídají, jsou vyznačeny barvou zelenou. Pro představu je vyhodnocení testu zobrazeno na obr. 6.1, kde je velká část testu v pořádku. Ke konci testu došlo ovšem k chybě, protože BMS spadla do stavu BMS error (status = 3) dříve, než jsme očekávali. Chybový stav měl nastat až po přibližně dalších deseti sekundách.

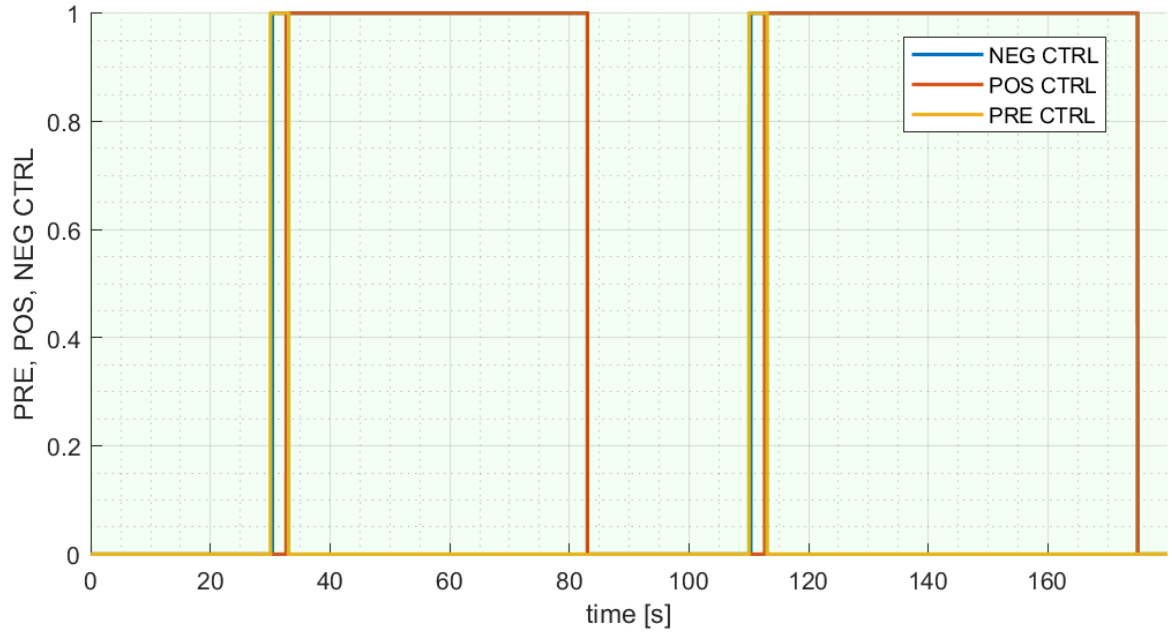


Obrázek 6.1: Vyhodnocení přednastaveného testu chybových stavů

6.2 Nezávislé vyhodnocení testů

Testy jsme schopni vyhodnotit také nezávisle na testovací sekvenci. Připraveny byly následující dva způsoby vyhodnocení.

První způsob zpracovává výstupy BMS v závislosti na jejím stavu. Víme, že pokud je BMS například ve stavu BMS error, musí být otevřené všechny izolační relé. Pokud je naopak ve stavu BMS ok, očekáváme sepnuté relé NEG a POS. Stejným způsobem může být ověřena správnost testu i v případě SDC final, napětí baterie, napětí meziobvodu měniče nebo jiných dostupných veličin. Vyhodnocení je opět provedeno uživatelem na základě zvýraznění v grafech, stejným způsobem jako v předchozí kapitole. Výsledek vidíme na obr. 6.2.



Obrázek 6.2: Vyhodnocení spínání relé, krátkodobý test

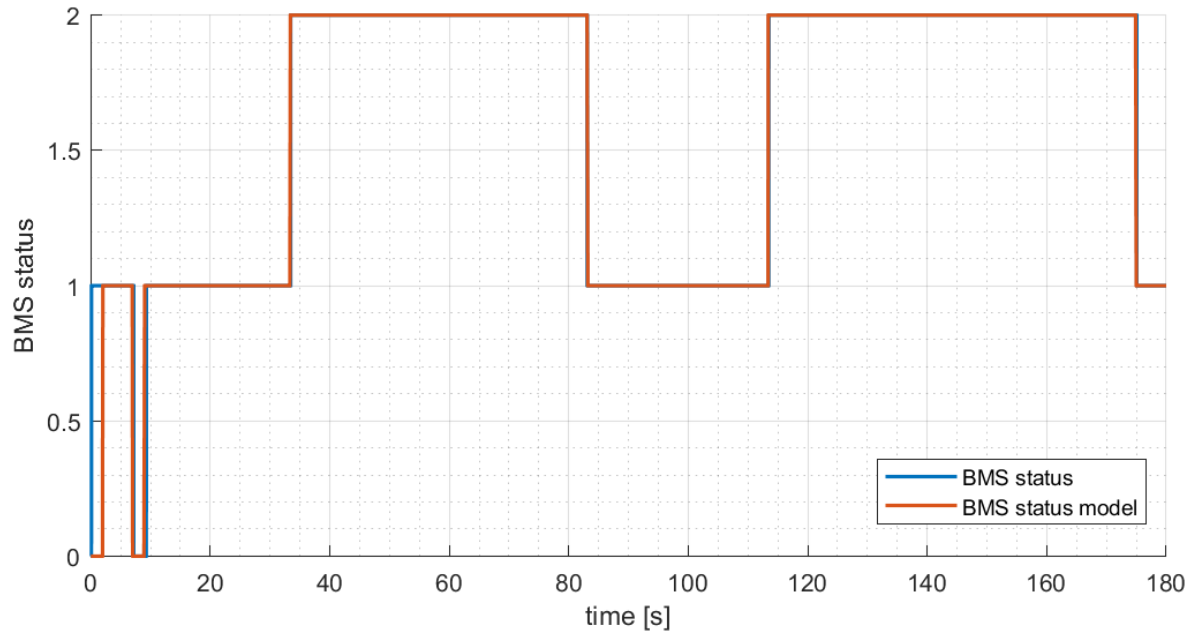
Chyby jsou také zobrazeny v příkazové řádce. Výpis obsahuje označení signálu, ve kterém byla nalezena chyba, čas ve kterém byla chyba nalezena a délka trvání chyby. Při hledání chyb uvažujeme ideální stav. Tedy například, že jsou relé rozepnuty ihned po přechodu BMS do chybového stavu. V reálném zařízení dochází k určitému zpoždění a chyby jsou nalezeny i tam, kde je výsledná akce správná. S tímto faktem je potřeba počítat a vyhodnotit, kdy je chyba v pořádku, nebo kdy již v pořádku není. Výpis s nalezenými chybami vidíme na obr. 6.3. V případě vybíjení kondenzátorů měniče může chyba, která stále reprezentuje správné chování, dosáhnout i času kolem tří sekund. Pokud vyžadujeme zcela automatické vyhodnocení testu, je v nastavení vyhodnocení možné zapnout automatické odstranění nálezů, které s velkou pravděpodobností obsahují chybnou detekci. Časové limity pro odstranění chyby lze taktéž nalézt v nastavení vyhodnocení.

Druhý způsob zahrnuje porovnání hodnot získaných HIL testem s daty získanými simulací modelu BMS (viz 6.4). Model BMS je popsán v kapitole 6.2.1. Model představuje ideální chování BMS. Sledujeme odchylky průběhů dat a na základě znalostí BMS vyhodnocujeme, zda test proběhl v pořádku, nebo výsledky již nesplňují požadavky. Při porovnání na obrázku 6.4 vidíme, že počáteční stav se liší. Počáteční stav při HIL testu

```
Chyba nalezena (SDC final), čas: 83.00 sec, trvani erroru: 0.09 sec
Chyba nalezena (SDC final), čas: 174.99 sec, trvani erroru: 0.08 sec
Chyba nalezena (AIRs), čas: 83.02 sec, trvani erroru: 0.07 sec
Chyba nalezena (AIRs), čas: 175.01 sec, trvani erroru: 0.06 sec
>>
```

Obrázek 6.3: Výpis chyb do příkazového řádku Matlabu

je závislý na stavu BMS při zapnutí testu a nejedná se o chybu.



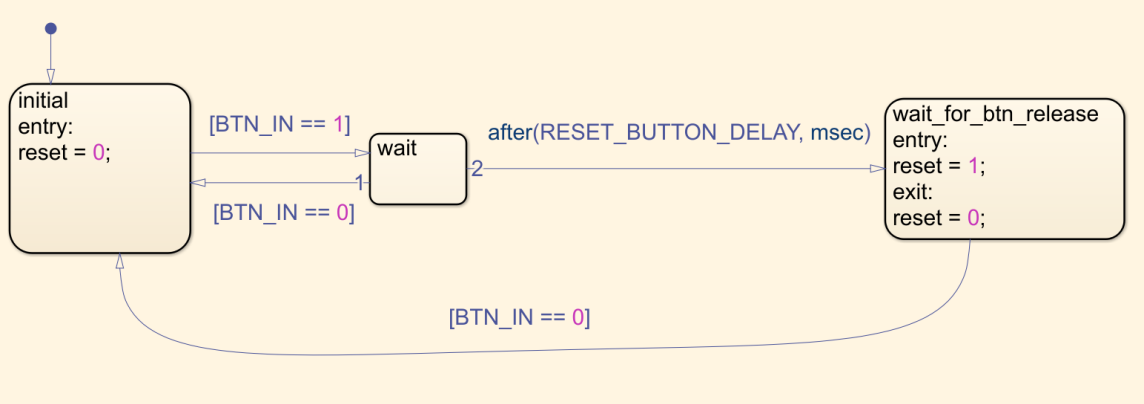
Obrázek 6.4: Porovnání výsledků HIL testu s modelem BMS

6.2.1 Model BMS

Model řídících jednotek BMS byl vytvořen za účelem vyhodnocení testů, lze jej ale využít také při navrhování testovacích sekvencí nebo hledání chyb v modelech pro testování. S modelem jsme schopni provádět MIL testování a na základě výsledků upravovat jak kontrolér (testovací sekvenci), tak model BMS.

Hlavní částí modelu je stavový automat běžící na BMS master. Pro jeho vytvoření byl využit Stateflow. Model byl sestrojen tak, aby kopíroval očekávané chování BMS. V modelu bylo možné zjednodušit velké množství částí BMS. V C kódu je potřeba velkou část programování zaměřit na časování smyček, obsluze přerušení a digitální komunikaci, která je poměrně složitá. Tyto části lze v modelu jednoduše nahradit, nebo úplně vynechat. Příkladem je právě digitální komunikace. V modelu ji nemusíme řešit a posíláme přímo hodnoty proměnných.

Kromě kódu bylo potřeba vyřešit také analogové části PCB. Např. reset BMS je prováděn analogově, stažením boot pinu mikrokontroléru na zem. Časování je řešeno komparátory a RC články s danou časovou konstantou. Model pro hardwarový reset je zobrazen na obr. 6.5.



Obrázek 6.5: Model analogového obvodu pro reset BMS

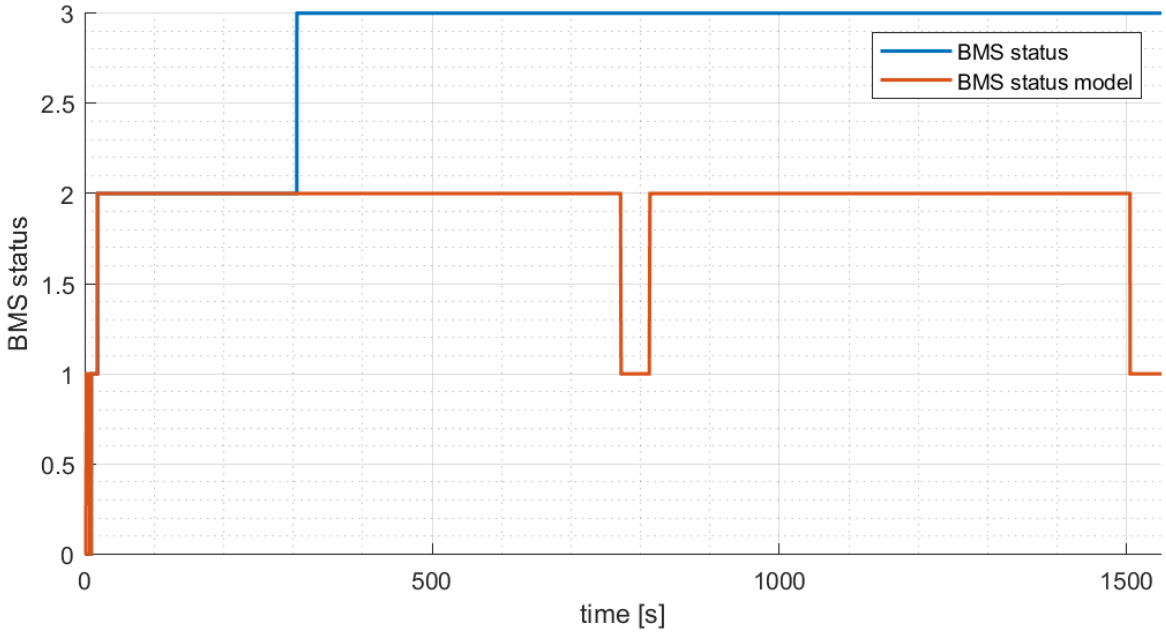
6.3 Chyby nalezené během testování

Během testování bylo nalezeno několik chyb BMS, které znemožňovaly spuštění TS. Tyto chyby byly vyřešeny a popsány níže.

Prvním problémem, který se objevil při zprovozňování HIL testů, byla chyba komunikace mezi BMS master a LEM senzorem. Po I^2C sběrnici nebyl přenášen žádný signál a BMS master vykazoval chování odpojeného senzoru. Byla proto provedena důkladná analýza sběrnice pomocí multimetru a osciloskopu, pomocí které byly nalezeny studené spoje na I^2C čipu a na straně LEM senzoru. Na straně BMS master byla taktéž nalezena chyba čipu a bylo potřeba I^2C čip vyměnit.

Další chybou, která byla nalezena, bylo špatné přiřazení pinů mikroprocesoru pro snímání sepnutí AIR. V software BMS bylo prohozeno měření relé POS a PRE. Zapínací sekvence tedy neproběhla v pořádku. Software byl následně upraven tak, aby bylo možné pokračovat v testování ostatních funkcí.

Největším problémem, který byl nalezen s pomocí HIL testování, byla chyba a tím způsobený výpadek měření ADC na BMS hat. Pro zopakování, ADC měří napětí baterie a napětí na meziobvodu měniče a následně komunikuje s BMS master pomocí I^2C. U vypadávání ADC nebyla nalezena žádná souvislost se stavem, ve kterém se BMS nachází. Během dlouhodobého testu pokaždé vypadne v jiné části. Jeden z nepodařených testů je zobrazen na obr. 6.6. Výpadek měření způsobuje chybu a BMS správně reaguje přechodem do chybového stavu. Ochrana proti této chybě funguje tak, jak očekáváme. Chybu se v době psaní diplomové práce nepodařilo vyřešit. Podařilo se pouze zmírnit četnost chyb změnou rychlosti komunikace na sběrnici I^2C.



Obrázek 6.6: HIL test s chybou ADC v porovnání s výstupem modelu BMS

7 Validace simulovaných částí

Validace je důležitý proces ověřování, zda daný systém nebo proces splňuje požadavky, které na něj klademe. V našem případě jde o validaci modelů, kterými simulujeme reálné komponenty. Validaci je potřeba provést v rámci každého HIL testování. Modely v počítači nám dovolují generovat signály, které nemusí odpovídat reálnému zařízení a je proto potřeba ověřit, zda naše simulace odpovídá chování reálných komponent. V případě, že validace není provedena, není jisté, zda se zařízení bude při připojení k reálné soustavě chovat tak, jak se chovalo při připojení k simulované soustavě.

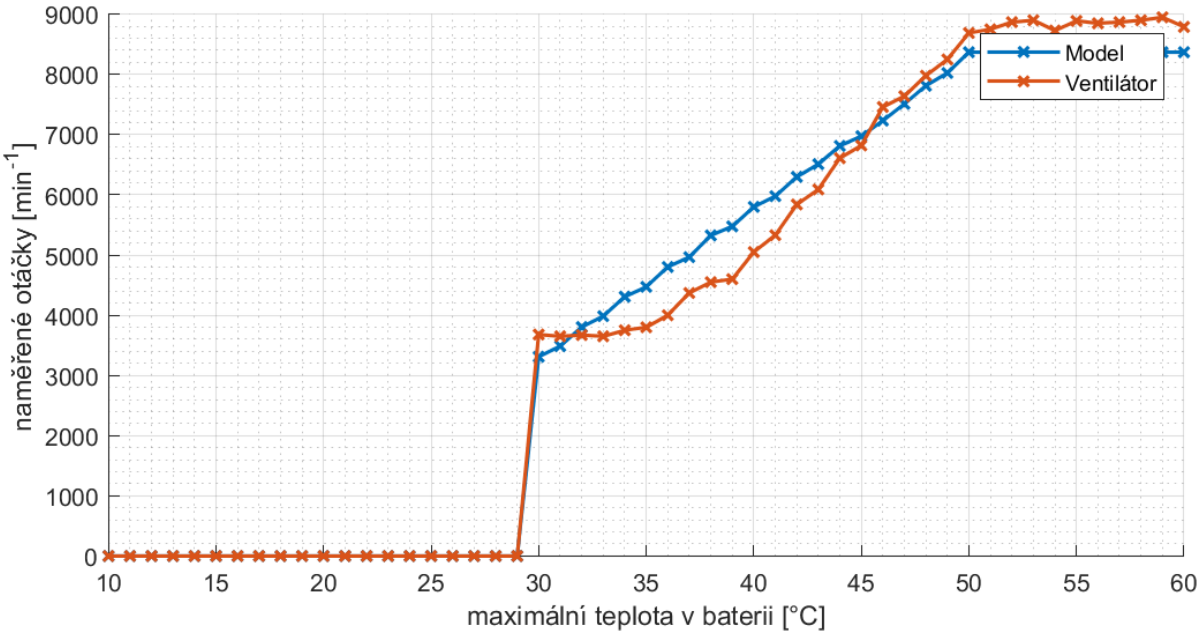
7.1 Ventilátor

Ventilátor byl před validací jeho modelu připojen k BMS master, aby byla zkontrolována jeho funkčnost. Jakmile byla nastavena střída 50%, nebyl ventilátor roztočen, pouze zareagovala podpěťová ochrana BMS a bylo odpojeno napájení DPS i ventilátoru. Po důkladné analýze systému bylo zjištěno, že v návrhu BMS master chybí dostatečná kapacita, která by pokryla velký odběr energie při roztáčení ventilátoru. Na vstup DPS byly proto přidány dva kondenzátor s celkovou kapacitou $C = 122\mu F$. Tato jednoduchá úprava vyřešila nedostatek návrhu a mohlo být přikročeno k validaci modelu.

PWM pro ovládání ventilátoru je posílána pouze pokud je BMS ve stavu BMS ok nebo BMS charging. Je proto nutné připojit BMS k dSPACE a zapnout TS. Poté byla, za účelem validace modelu, ručně, za pomocí ControlDesku, měněna hodnota maximální teploty článků, kterou posíláme po sběrnici CAN slaves. Byl zvolen rozsah měření teploty od 10°C do teploty 60°C. Vyšší teplota než 60°C vypne TS. Teplota byla zvedána po jednom stupni Celsia. Po ustálení hodnoty otáček byla odečtena hodnota rychlosti otáčení z výpisu, který posílá BMS pomocí UART komunikace do počítače. Tento postup byl proveden dvakrát. Jednou pro připojený ventilátor, podruhé pro připojený dSPACE, který generoval zpětnou vazbu podle instrukcí kódu ze Simulinku.

Naměřené hodnoty byly vyneseny do grafu (viz obr. 7.1) a propojeny čarou, která nezobrazuje naměřená data, ale je zobrazena pro lepší představení rozdílů mezi měřenými daty. Modrou barvou jsou zvýrazněná data, která byly naměřena s připojeným dSPACE. Červenou barvou jsou pak zobrazena data naměřena s připojeným ventilátorem. Vidíme, že závislost otáček ventilátoru na maximální teplotě odpovídá datům s připojeným modelem. Ventilátor do systému přináší zřetelné odchylky od optimálních otáček určených katalogovým listem výrobce. Odchylky mohou být způsobeny určitou nelinearitou, kterou může představovat například proměnná teplota, nebo proměnlivé tření v závislosti na otáčkách ventilátoru. Také je možné že bylo měření ovlivněno chybou odčítání hodnoty z displeje. Otáčky ventilátoru při měření nebyly konstantní a záleží na čase ve kterém byly odečteny. Pro přesnější výsledky by bylo potřeba provést měření více a počítat s chybami měření. Tento proces byl vynechán vzhledem k tomu že ventilátor není v systému baterie kritickou součástí a hodnota otáček je pouze orientační.

S připojeným ventilátorem byly také spuštěny všechny tři automatizované testy po-



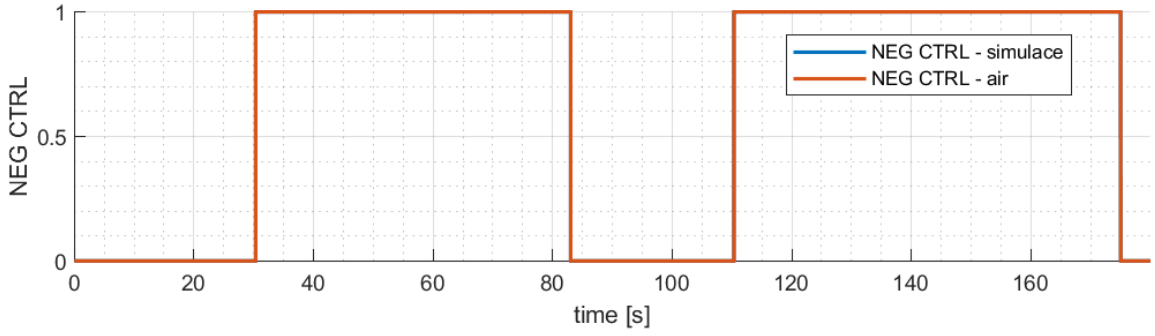
Obrázek 7.1: Porovnání naměřených otáček při připojení modelu a ventilátoru

psané v kapitole 5. V průběhu testů nebyly nalezeny rozdíly oproti testům, při kterých byl využit model.

Validaci modelu proto prohlašujeme za úspěšně provedenou a model je možné využít během dalšího HIL testování. Stav, kdy je nárazově odebírán velký proud z BMS, nemohl být testován v rámci metody HIL testování, protože nebyly dostupné výkonové moduly pro dSPACE jednotku. Výkonovou část je proto potřeba testovat zvlášť a nelze v této části spoléhat na získané výsledky.

7.2 Spínání a snímání AIR

Model izolačních relé byl validován během HIL testování. Místo simulace NEG AIR bylo fyzicky zapojeno relé a bylo sledováno, zda se systém chová stejně, jako v případě simulování. Výsledné porovnání bylo vyneseno do grafu a můžeme jej vidět na obr. 7.2.

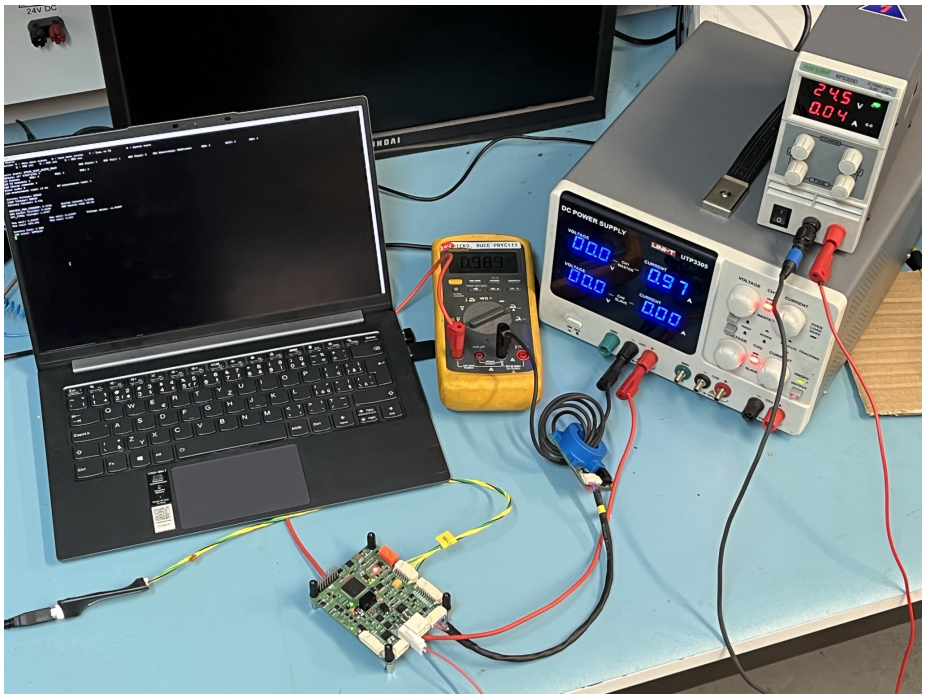


Obrázek 7.2: Porovnání chování BMS s připojeným NEG AIR vůči simulaci AIR

Automatizované testy proběhly dle očekávání a můžeme potvrdit správnost HIL simulace AIR.

7.3 Měření proudu LEM senzorem

Pro validaci měření proudovým senzorem byl využit laboratorní zdroj, ze kterého odebíráme proud. Proudovým senzorem byl šestkrát ve stejném směru provlečen výstup ze zdroje. Na výstupu proudového senzoru proto měříme napětí odpovídající šestinásobku proudu proudícího vodičem. Proud, který odebíráme byl nastaven na zdroji v režimu konstantního proudu pomocí omezovače proudu. Měření proudu je prováděno pomocí multimetru zapojeného do série s proudovým senzorem. Toto zapojení je zvoleno z důvodu přesnosti. Multimetr je schopný zobrazit hodnotu proudu s vyšší přesností, než obrazovka zdroje. Zapojení při testování můžeme vidět na obrázku 7.3. Proud byl následně odečítán z výpisu BMS master, který je posílán pomocí UART sběrnice do počítače.



Obrázek 7.3: Validace měření LEM senzorem

V rámci měření byly naměřeny hodnoty šestkrát větší, než proud odebíraný ze zdroje. Při generování výstupu z LEM senzoru bylo dosaženo stejných výsledků. Validace LEM senzoru tedy proběhla úspěšně.

7.4 Baterie

Vzhledem k tomu, že v době psaní nebyla dostupná baterie ve stavu, ve kterém by mohla být připojena k BMS a validována přímo v propojení s fyzickou soustavou, využili jsme alespoň výsledků souběžné bakalářské práce Jakuba Lysáka [24]. V rámci této bakalářské práce byly měřeny samostatné články. Za pomocí získaných dat byl vytvořen model baterie, který odpovídá chování reálné baterie.

Validace chování BMS v rámci reálné soustavy baterie bude provedena jakmile bude možné bateriový systém k baterii připojit.

8 Závěr

Hlavním náplní práce bylo testování bezpečnostních funkcí bateriového systému studentské formule. Zadání práce obsahuje čtyři cíle, které provázely postup celou prací. Cíle lze rozdělit na jednotlivé kroky, které je potřeba splnit pro plnohodnotné testování.

V rámci prvního kroku byla sepsána rešerše na téma HIL testování bateriových systémů. Zaměření bylo směřováno na HW a SW, který se pro HIL testování běžně využívá. Kromě toho byl sepsán přehled funkce bateriového systému, který bude využíván pro kontrolu baterie závodního monopostu Dragon e3.

V druhém kroku bylo na základě rešerše rozvrženo, jaké části bateriového systému studentské formule budou testovány a jakým způsobem je k testovací soustavě připojíme.

Návrh tří testovacích sekvencí, které zkouší správnou funkčnost především kritických částí BMS, je dalším krokem, který byl absolvován. Aby bylo možné data zpracovat, byl sestaven v Matlabu skript, který automaticky vyhodnotí základní předpoklady chování BMS a poskytne uživateli grafické výstupy. Ty pomáhají při manuálním vyhodnocení a vizuální kontrole výsledku. V průběhu testování byly odhaleny chyby BMS, které ovlivňovaly bezpečnost a spolehlivost systému. Poznatky byly zaznamenány a konzultovány s členy týmu TU Brno racing zodpovědnými za vývoj SW.

Nakonec byly validovány všechny modely využité pro simulaci, pro které byly v době psaní práce dostupné reálné komponenty a bylo jimi možné simulaci nahradit. Tímto byly zakončeny všechny kroky odpovídající cílům práce.

Finálním výsledkem diplomové práce je navržený a vyzkoušený proces umožňující automaticky testovat BMS, zkoušet různé nebezpečné stavy a vytvářet tak podmínky pro bezpečný provoz jednotky. To vše nezávisle na části sezóny a dostupných částech baterie. HIL testování BMS má také potenciál stát se dobrým nástrojem pro seznámení se s tímto komplexním systémem.

Řešení, které je prezentováno, je možné dále rozšiřovat. V rámci HIL testování je nepochybně potřeba validace chování BMS ve spojení s dalšími částmi baterie, které v době vypracování práce nebyly dostupné. Zvýšenou pozornost je nadále potřeba věnovat také navrženým testům a jejich vyhodnocení. BMS představuje rozsáhlý systém, ve kterém nebylo možné v rámci diplomové práce otestovat všechny možné stavy. Velkou pomocí při testování by také mohl být přehlednější způsob připojení BMS k dSPACE, které by mohlo být realizováno např. návrhem DPS určené pro tento účel.

References

[1] JOSHI, Adit. *Automotive Applications of Hardware-in-the-Loop (HIL) Simulation* [online]. Warrendale: SAE International, 2020 [cit. 2023-05-26]. ISBN 978-1-4686-0003-2.

[2] *Future Advances in Body Electronics: AMPG Body Electronics Systems Engineering Team*. 2013. Dostupné také z: https://www.nxp.com/docs/en/white-paper/BODYDELECTRWP.pdf?fbclid=IwAR0AL0MhC5M-GhKDhOqyV_Us1VT1yh5NLcZ76gmJW%5C-UjIFAPqfDhb-P5VDPI.

[3] SAURAV, Sanket. *The Exponential Cost of Fixing Bugs* [online]. [cit. 2023-05-25]. Dostupné z: https://dzone.com/articles/the-exponential-cost-of-fixing-bugs?fbclid=IwAR1G4bR5aR2vLIwLO-LtCpFRHVlk0W2JKBtCQDGj2gK-ikx_zIKleTsFXaw.

[4] *Measuring the Return on Investment of Model-Based Design* [online]. 2021. [cit. 2023-05-26]. Dostupné z: https://www.mathworks.com/content/dam/mathworks/white-paper/measuring-roi-of-model-based-design.pdf.

[5] BOHMAN, Vojtěch. *Aplikace testování založeného na modelu testovaného systému na HiL platformě*. Plzeň, 2016. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd.

[6] TULPULE, Punit; REZAEIAN, Ayyoub; KARUMANCHI, Aditya; MIDLAM-MOHLER, Shawn. *Model Based Design (MBD) and Hardware In the Loop (HIL) Validation: Curriculum Development*. Seattle: American Control Conference, 2017.

[7] BRAYANOV, Nikolay; STOYNOVA, Anna. Review of hardware-in-the-loop - a hundred years progress in the pseudo-real testing. *E+E* [online]. [B.r.], roč. 54, č. 3-4, s. 70–84 [cit. 2023-05-26]. Dostupné z: https://www.researchgate.net/publication/337880811_Review_of_hardware-in-the-loop_-a_hundred_years_progress_in_the_pseudo-real_testing.

[8] *SCALEXIO LabBox: Modular real-time system* [online]. [cit. 2023-05-26]. Dostupné z: https://www.dspace.com/en/inc/home/products/hw/simulator_hardware/scalexio/scalexio_labbox.cfm.

[9] *Mobile Real-Time Target Machine* [online]. [cit. 2023-05-26]. Dostupné z: https://www.speedgoat.com/products-services/real-time-target-machines/mobile-real-time-target-machine.

[10] *DS5386 High-Voltage Electronic Load Module: Electronic load for power HIL simulation up to 1,250 V* [online]. [cit. 2023-05-25]. Dostupné z: https://www.dspace.com/en/pub/home/products/hw/simulator_hardware/hv_electronic_load.cfm?fbclid=IwAR2bdHYN3QJepwVq9PNFM-krTUWmrW0VFGXKhlzhDO9rkQ-UVeBrWQ%5C-LPkEA.

[11] *EV1077 Battery Cell Voltage Emulation Board: Simulating high-voltage batteries at cell level for system voltages of up to 1,000 V* [online]. [cit. 2023-05-25]. Dostupné z: https://www.dspace.com/en/pub/home/products/special_solutions/battery_cell_voltage_emulation.cfm?fbclid=IwAR1-WanAoK65bOhF1GW5h59u%5C-flvD2x51bj3AARsT1UF7ZGUHqE2SIjW9_zM.

[12] *DS5481 Battery Cell Voltage Emulation Board: High-end board for simulating high-voltage batteries at cell level for system voltages of up to 1,500 V* [online]. [cit. 2023-05-26]. Dostupné z: https://www.dspace.com/en/inc/home/products/special_solutions/ds5481-battery-cell-voltage-em.cfm.

[13] *IO992: 6-Channel Battery Cell Emulation I/O Module with Current and Voltage Measurements* [online]. [cit. 2023-05-26]. Dostupné z: https://www.speedgoat.com/products/battery-cell-emulator-io992.

[14] *AutomationDesk: Powerful test authoring and automation tool* [online]. [cit. 2023-05-26]. Dostupné z: https://www.dspace.com/en/ltd/home/products/sw/test_automation_software/automationdesk.cfm.

[15] MACKO, Maroš. *Bateriový monitorovací systém pro formuli student*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií.

[16] *Formula Student Rules*. 2023. Dostupné také z: https://www.formulastudent.de/fileadmin/user_upload/all/2023/rules/FS-Rules_2023_v1.1.pdf.

[17] *ISOMETER® IR155-3203/IR155-3204* [online]. [cit. 2023-05-26]. Dostupné z: https://www.bender.de/en/products/insulation-monitoring/isometer-ir155-3203-ir155-3204/.

[18] MACKO, Maroš. *Měnič pro motory Formule Student*. Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií.

[19] *DS1006 Processor Board, Features: Release 7.0 – November 2010*. Paderborn: DSpace, 2010.

[20] *DS2202 HIL I/O Board, Features: Release 5.1 – May 2006*. Paderborn: DSpace, 2006.

[21] *CAN DBC File Explained - A Simple Intro [+Editor Playground]* [online]. [cit. 2023-05-26]. Dostupné z: https://www.csselectronics.com/pages/can-dbc-file-database-intro.

[22] *KILOVAC LEV100H Contactors* [online]. [cit. 2023-05-26]. Dostupné z: https://www.te.com/usa-en/about-te/news-center/adm-kilovac-lev100h-contactors.html.

[23] PLETT, Gregory L. *Battery management systems. Volume I, Battery modeling*. Norwood: Artech House, 2015. ISBN 9781630810238.

[24] LYSÁK, Jakub. *Baterie pro monopost Formule student*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií.

[25] *8315100134* [online]. [cit. 2023-05-26]. Dostupné z: https://cz.mouser.com/ProductDetail/ebm-papst/8315100134?qs=TuK3vfAjtkUZtMtV2d55FQ==.

Seznam zkratek

ADC Analog-to-digital converter (analogově-digitální převodník)

AMS Accumulator Management System

BMS Battery management system (Bateriový systém)

CAN Controller Area Network

CANL CAN low

CANH CAN high

DAC Digital-to-analog converter (digitálně-analogový převodník)

DBC DataBase Container file

DIS Discharge (vybíjecí)

ECU Electrically controlled unit (Elektronická řídící jednotka)

FPGA Field-programmable gate array (programovatelné hradlové pole)

HIL Hardware-in-the-loop testování

HW Hardware

I2C Inter-Integrated Circuit

I/O Input/Output (vstupní/výstupní)

MBD Model-based design

MIL Model-in-the-loop testování

NEG Negative (záporný)

pHIL Power Hardware-in-the-loop testování

PIL Processor-in-the-loop testování

POS Positive (kladný)

PRE Precharge (nabíjecí)

SDC Shutdown circuit (vypínací obvod)

SCS Safety critical signal (kritický signál pro bezpečnost)

SIL Software-in-the-loop testování

SPI Serial Peripheral Interface

SW Software

TS Tractive system (trakční systém)

TSMS Tractive System Master Switch (spínač připojující trakční systém)

UART Universal asynchronous receiver-transmitter

Seznam obrázků

Seznam tabulek

Seznam příloh

Přiložené soubory

- Laborator - složka se soubory z PC v laboratoři
  - BMS_HIL_automated - složka se soubory experimentu
  - BMS_HIL_manual - složka se soubory experimentu
  - CAN_dbc - složka obsahující CAN DBC soubory
  - modely - složka s modely ze Simulinku 2011b
- OsobniPC - složka se soubory z osobního počítače
  - m_kody - složka se skripty z Matlabu 2022b
  - modely - složka s modely ze Simulinku 2022b
  - data_hil_2_029 - výsledná data z krátkodobého HIL testu
  - data_hil_2_031 - výsledná data z dlouhodobého HIL testu
  - data_hil_2_037 - výsledná data z HIL testu chybových stavů
- SchemaTestuChybovychStavu - Obrázek se schématem

Tabulka propojení BMS s dSPACE

	název signálu	označení v dSPACE	periferie dSPACE	komentář
CAN crit	CAN crit H	P2A/47	CAN2H	-
	CAN crit L	P2A/31	CAN2L	-
CAN slaves	CAN slaves H	P2B/47	CAN1H	-
	CAN slaves L	P2B/31	CAN1L	-
AIRS	PRE ctrl	P2A/5	DIG_IN38	Pull-up 2K2
	POS ctrl	P2A/21	DIG_IN28	Pull-up 2K2
	NEG ctrl	P2A/38	DIG_IN30	Pull-up 2K2
	PRE state	P2A/7	DIG_OUT4	-
	POS state	P2A/23	DIG_OUT3	-
	NEG state	P2A/40	DIG_OUT5	-
HV_meas	Batt_L	P1A/1	GND	-
	Load_L	P1A/1	GND	-
	Batt_H	P1A/41	DAC6	-
	Load_H	P1A/42	DAC8	-
SDC	SDC IN	P2A/10	DIG_OUT12	-
	SDC Final CTRL	P2A/11	DIG_OUT15	P-ch mosfet
	SDC_to_TSMS	P1A/34	DIG_IN2	-
BTN BMS	BTN_IN	P2A/43	DIG_OUT13	22K + 10K
IMD	IMD_OK	P2A/27	DIG_OUT14	-
	IMD_DATA	P2B/6	PWM_OUT1	-
BTN IMD	BTN_IMD	P2A/44	DIG_OUT16	22K + 10K
FANs	FAN_PWM	P1A/18	PWM_IN11	Pull-up 10K
	FAN_Tacho1	P2B/23	PWM_OUT3	-
	FAN_Tacho2	P2B/24	PWM_OUT6	-
LEM	LEM_L	P1A/11	GND	-
	LEM_H	P1A/27	DAC12	-

Tabulka 8.1: Propojení BMS s dSPACE

Využitý hardware

dSPACE

- DS1006 board version 6.2.1
  - DS1006 FPGA 2.4.7
  - DS1006 CPLD 1.0.6
- DS2202 4.1
- DS821-34mm Link Board
- WibuBox/U+

Využitý software

PC v laboratoři

- Matlab 7.13.0.564 (R2011b)
- Real-Time Interface to Simulink 6.9
- RTI CAN Blockset 2.8
- Model Port Block Library 1.4
- dSPACE Real-Time Target (DSRT) 1.4
- MATLAB-dSPACE Interface Libraries 4.7.4
- Model Separation Block Library 1.0
- dSPACE MATLAB Connection 2.3 (win32) 2.3.1
- ControlDesk developer version 3.7.3

SW pro dSPACE

- DS1006 Firmware 2.5.1
  - PHS Bus Scanner 2.3
    - DS2202 4.1
      - DS2202 CAN slave firmware 1.7.5
  - Boot command server 1.0
  - Gigalink Scanner 2.0