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

ŘÍZENÍ SOUSTAVY ELEKTRICKÝCH MOTORŮ PŘES CAN BUS

CONTROL OF SYSTEM WITH ELECTRIC MOTORS BY CAN BUS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Petr Hamřík

AUTHOR

VEDOUCÍ PRÁCE — Ing. Rostislav Huzlík, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Petr Hamřík**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Rostislav Huzlík, Ph.D.**
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 bakalářské práce:

Řízení soustavy elektrických motorů přes CAN BUS

Stručná charakteristika problematiky úkolu:

Cílem práce je navrhnout a realizovat program pro řízení soustavy pohonů na platformě Arduino Due a pomocí sběrnice CAN bus. K řízení budou využité standy, které již obsahují motory a potřebnou elektroniku.

Cíle bakalářské práce:

Rešerše na téma řízení stejnosměrných motorů a sběrnice CAN.
Seznámení s předloženým hardwarem.
Vytvoření simulace předloženého hardwaru.
Vytvoření programu pro řízení na předloženém hardwaru.
Vyhodnocení možností využití řízení pohonů pomocí směrnice CAN.

Seznam doporučené literatury:

ROUBÍČEK, Ota, 2004. Elektrické motory a pohony: příručka techniky, volby a užití vybraných druhů. Praha: BEN – technická literatura. ISBN 80-730-0092 VODA, Zbyšek, 2015. Průvodce světem Arduina. Bučovice: Martin Stříž. ISBN 978-80-87106-90-7.

Termín odevzdání bakalářské 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

Cílem této bakalářské práce bylo zpracovat řízení stejnosměrných motorů na předem vytvořených standech za použití sběrnice CAN.V práci je rešeršní část, která se zabývá řízením stejnosměrných motorů. V práci je zpracován model DC motoru v MATLAB Simulink. Poslední část se zabývá fyzickým řízením stejnosměrných motorů a vytvořením uživatelského rozhraní.

KLÍČOVÁ SLOVA

CAN BUS, DC motor, uživatelské rozhraní, enkodér, proudový snímač, H – můstek, Arduino Due

ABSTRACT

The aim of this bachelor's thesis was to work the control of DC motors on already created stands using the CAN BUS. There is a research part in the thesis that deals with the control of DC motors. The DC motor simulation was created in MATLAB Simulink. The last part deals with the physical control of DC motors and the development of the user interface.

KEYWORDS

CAN BUS, DC motor, user interface, encoder, current sensor, H – bridge, Arduino Due

BIBLIOGRAFICKÁ CITACE

HAMŘÍK, Petr. *Řízení soustavy elektrických motorů přes CAN BUS* [online]. Brno, 2023 [cit. 2023-05-24]. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/145659. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce: Ing. Rostislav Huzlík, Ph. D.

PODĚKOVÁNÍ

Zde bych chtěl poděkovat svému vedoucímu bakalářské práce Ing. Rostislav Huzlík, Ph. D. za jeho přístup a pomoc při řešení problémů v průběhu zpracovávání práce.

Petr Hamřík

V Brně dne …… ……

Podpis autor

PROHLÁŠENÍ AUTORA O PŮVODNOSTI PRÁCE

Prohlašuji, že bakalářskou práci jsem vypracoval samostatně, pod odborným vedením Ing. Rostislav Huzlík, Ph.D. Současně prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským.

Petr Hamřík

V Brně dne …… ……

Podpis autora

OBSAH

1 ÚVOD

Tato práce je zaměřena na řízení stejnosměrných motorů řízených pomocí CAN bus na předem připravených standech. Tato soustava pohonů bude využita jako výukové zařízení pro názornou ukázku řízení jako je například regulace polohy, rychlosti elektronický hřídel a elektronická převodovka. Každý stand bude mít přiřazené vlastní ID zpráv, které jej budou ovládat přes uživatelské rozhraní.

V první kapitole je zpracována stručně rešerše na téma řízení stejnosměrných motorů a sběrnice CAN bus. Jelikož pro následný návrh programu potřebujeme mít bližší pohled jednotlivých komponent využitých pro řízení.

Další kapitoly se zabývají následnou simulací a programem. Simulace byla provedena v prostředí MATLAB Simulink vytvořením modelu stejnosměrného motoru za použití dynamických rovnic a kaskádním regulátorem.

Hlavní program byl vytvořen pomocí aplikace Arduino IDE, která je open – source a vložením potřebných knihoven k řízení pomocí CAN bus. Požadavky pro ovládání standů se budou zadávat do uživatelského rozhraní, které je vytvořeno pomocí nástroje MATLAB App Designer.

2 REŠERŠE

2.1 Stejnosměrný motor

Tento typ motoru se řadí mezi nejstarší elektrické pohony. V dnešní době má na trhu pořád hojné zastoupení, jelikož je cenově dostupný oproti jiným elektrickým motorům. Základními typy jsou kartáčové a bezkartáčové DC motory. Skládá se ze statoru a rotoru. Kartáčové motory jsou náročné na pravidelnou údržbu, jelikož v místě komutátoru dochází k jiskření, opotřebení kartáčů (prach, nečistoty). Otáčky můžeme jednoduše měnit změnou napájecího napětí kotvy a změnou budícího proudu. Je založen na jednoduchém principu, kdy do vodiče (cívky), který se nachází v magnetickém poli přivádíme elektrický proud, což vede k následnému otáčení rotoru. Může pracovat jak motor, tak i generátor. Zásadní rozdíl v tom není, jelikož když začneme pohybovat rotorem stejnosměrného motoru, tak se nám na vodičích bude indukovat napětí úměrné rychlosti otáčení.

Stejnosměrný motor s permanentními magnety můžeme popsat rovnicemi, pro indukované napětí

$$U_i = C\Phi \cdot \omega \tag{2.1}$$

a moment, který vytváří působení proudu a magnetického toku

$$M = C\Phi \cdot I \tag{2.2}$$

Přechodný stav stejnosměrného motoru je popsán dynamickými rovnicemi pro napětí

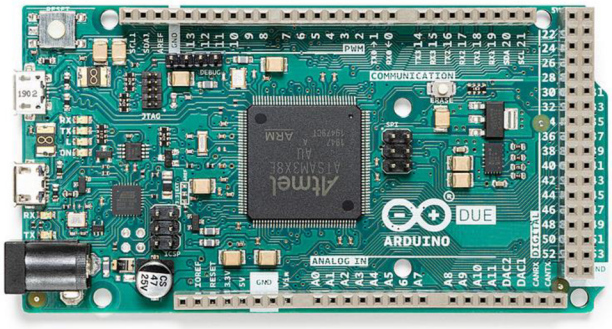
$$U = R_a \cdot i + L_a \cdot \frac{di}{dt} + C\Phi \cdot \omega \tag{2.3}$$

a moment

$$C\Phi \cdot i = J \cdot \frac{d\omega}{dt} + B \cdot \omega + M_0 \tag{2.4}$$

2.2 Arduino

Arduino je open – source platforma pro programování mikrokontrolérů. Umožňuje vytvářet interaktivní projekty a experimenty. Hardwarová platforma je založena na mikrokontrolerech, které jsou vybaveny vstupními a výstupními piny na kterých lze přenášet jak analogové, tak digitální signály. Tyto piny lze připojit k různým senzorům, motorům, displejům atd. Výchozím programovacím nástrojem je Arduino IDE (integrated development environment), který je open – source. Díky jednoduchosti a přístupnosti Arduino platformy můžeme použít i další programovací nástroje jako je např. Matlab, VisualStudio, AtmelStudio7 po stažení náležitých knihoven. Arduino je velice populární při vytváření různých projektů a experimentů, jako je například robotizace, domácí automatizace, ale také vzdělávacími institucemi pro výuku elektroniky a programování. [1]



Obrázek 1 - Arduino Due

2.3 CAN Bus

Firma BOSCH začala jako první používat CAN BUS ke komunikaci ve vozidlech. Hlavní příčinou byly příliš objemné elektrické svazky ve vozidlech. CAN Bus je digitální datová sběrnice dvou elektrických vodičů (CAN_Low a CAN_High), kde jsou informace odesílány do a z ECU (Electronic Control Unit). Řídící jednotky komunikují mezi sebou a předávají si informace pomocí sítě, která se nazývá CAN (Controller Area Network).

Sběrnice CAN je sériová komunikační sběrnice navržená pro vysoký výkon a použití v náročných podmínkách. V prvopočátku se využívala pro komunikaci ve vozidlech, ale používá se v nesčetných aplikacích včetně automatizace továren, budov, letadel a kosmonautiky. Jde v podstatě o standard sběrnice vozidel, který umožňuje vzájemnou komunikaci mikrokontrolerů a zařízení. [2]

2.3.1 CAN Bus – typy

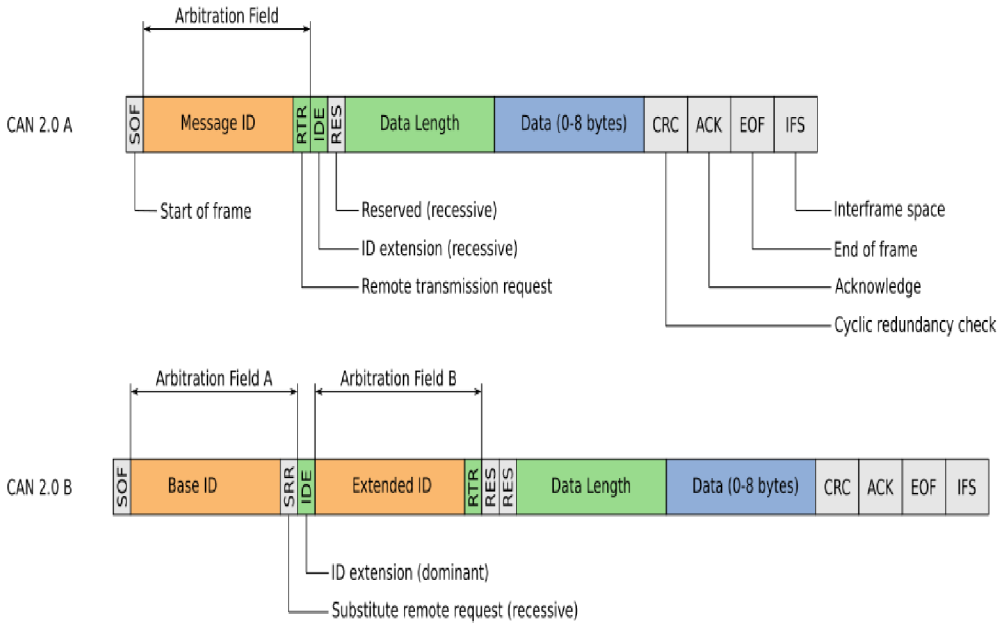
Bez CAN sběrnice je komunikace mnohem obtížnější, jelikož jednotlivé větve nejsou propojeny mezi sebou.

Existují různé typy CAN sběrnice:

1. High speed CAN podporuje přenosovou rychlost mezi 40 kbit/s až 1 Mbit/s. Využívá se pro protokoly jako je OBD2, CANopen, j939 atd. Je nejpoužívanější.
2. Low speed CAN podporuje přenosovou rychlost mezi 40 kbit/s až 125 kbit/s. Jestliže se poškodí jeden ze dvou vodičů, je schopný pokračovat v komunikaci.
3. CAN FD je rozšířením původního protokolu sběrnice. Hlavním důvodem bylo zvýšení přenosové rychlosti až na pěti násobek původní rychlosti CAN.

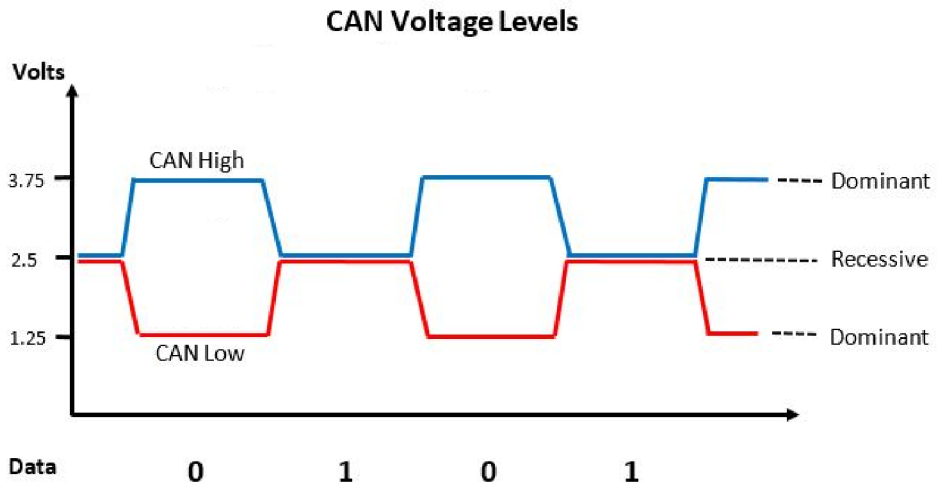
2.3.2 CAN Bus – přenos dat

Ke komunikaci se sběrnicí jsou v rámci dva logické stavy – dominantní a recesivní. Využívá negativní logiku. Typicky používá rámec s 11bitovou identifikací. Lze využít i rozšířený rámec který má 29bitovou identifikaci. Každá zpráva nese číselnou hodnotu, která řídí prioritu na sběrnici. Všechny uzly v síti CAN musí pracovat se stejnou nominální přenosovou rychlostí, jinak dojde k chybě na přijímací straně. [2]



Obrázek 2 - identifikace dat CAN zprávy [2]

1. Dominantní stav – Can HI 3,75 V, Can Low 1,25 V – rozdílové napětí je větší než práh nebo je dosažen logickou 0
2. Recesivní stav – Can Hi and Low 2,5 V – rozdílové napětí je menší než práh nebo je dosažen logickou 1



Obrázek 3 - CAN High/Low [3]

2.3.3 Výhody CAN Bus

Sběrnice CAN je ideální pro bezpečnostní aplikace, díky své spolehlivosti a odolnosti. K dispozici máme také několik kroků pro detekci chyb v protokolu, jako je vyplnění a monitorování bitů, kontrola rámců a kontrola cyklické redundance.

Je to nízko – nákladový systém za účelem zrychlení komunikace mezi elektronickými zařízeními a moduly, ale za účelem snížení chyb a velikosti kabeláže.

Rychlost je definována pomocí dvou fyzických vrstev High Speed CAN A Low Speed CAN.

CAN bus je protokol založený na své flexibilitě. Můžeme libovolně přidávat nebo odebírat uzly bez provádění aktualizace systému. Pomocí toho lze snadno přidávat do systému elektrická zařízení bez složité integrace.

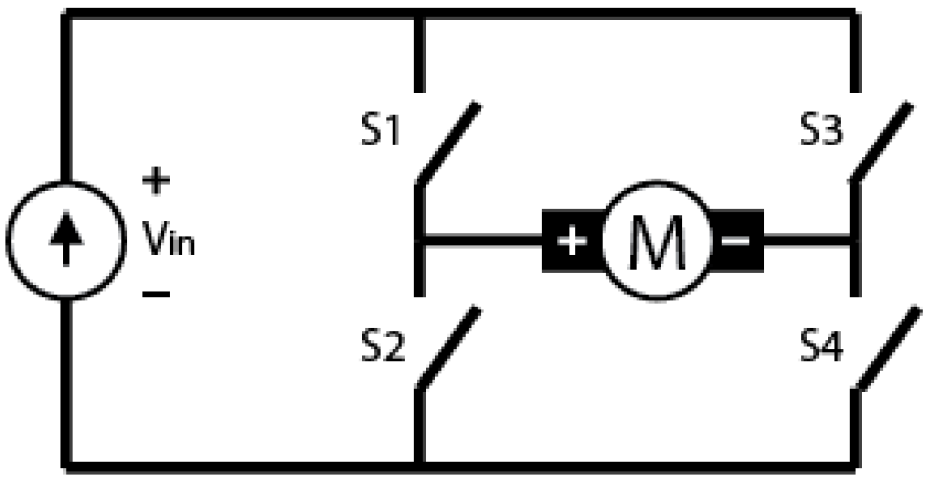
Data jsou upřednostňována podle priority. Nejvíce prioritní data získají okamžitý přístup ke sběrnici, ale bez přerušení ostatních rámců. [2]

2.4 H – můstek

Schéma zapojení vypadá jako písmeno H, odtud H – můstek. Obvod se skládá ze 4 spínačů umožňujících volbu směru proudu procházejícího součástkou. Nejjednodušší H – můstek se skládá ze čtyř ručně ovládaných spínačů. Dnes jsou dostupné jako integrované obvody. Využívají se hlavně pro změnu polarity/směru nebo brždění motoru.

H – můstky jsou řízeny buď na bázi polovodičů nebo pomocí relé. Polovodičový je vhodný tam, kde nemáme vysokorychlostní spínání nebo kde je mechanické opotřebení nežádoucí. Pro spínání se využívají hlavně PNP, NPN tranzistory nebo MOSFETy. Pro řízení využíváme PWM signál.

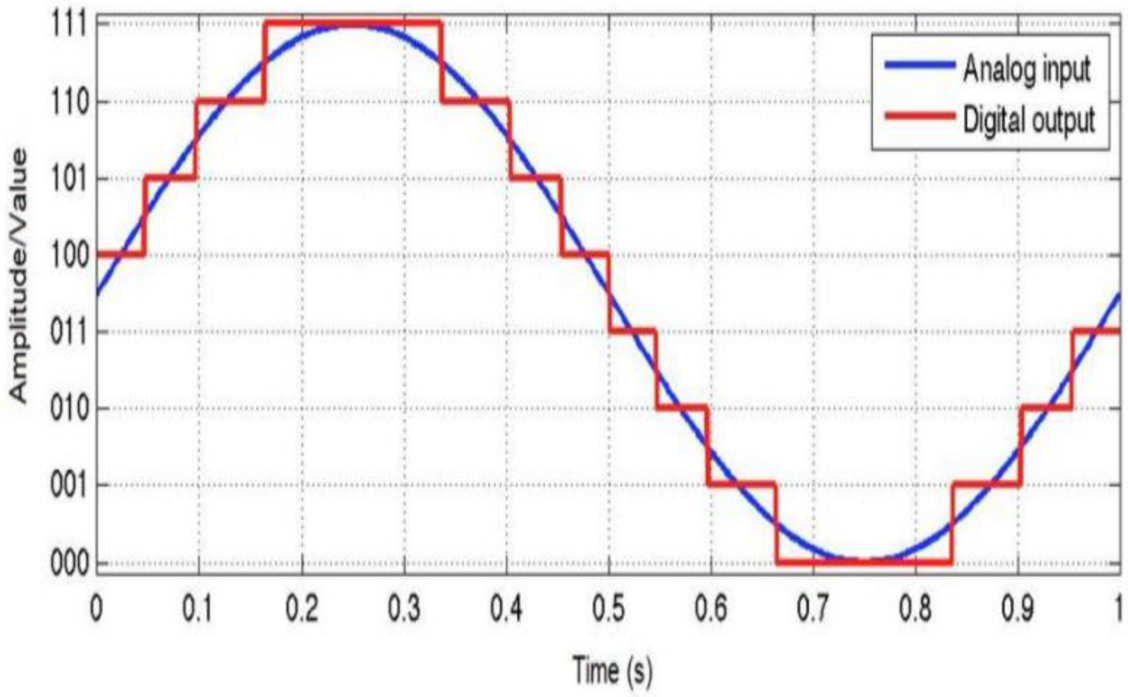
Odlišujeme také poloviční a celý H – můstek. Poloviční H – můstek není schopen přepínat polaritu napětí aplikovanou na zátěž na rozdíl od celého H – můstku.



Obrázek 4 - přepínání H – můstku pomocí mechanických spínačů [4]

2.5 A/D převodník

Analogově – digitální převodníky jsou důležitou součástí v digitálních systémech komunikujících v reálném čase. Analogový spojitý signál obsahuje sekvenci hodnot, které mohou pocházet ze zvuku, teploty, světla atd. Diskrétní digitální signál je reprezentován sekvencí diskrétních hodnot, která je závislá na vzorkovací frekvenci.



Obrázek 5 - vzorkování analogového signálu [5]

2.5.1 Vzorkovací frekvence

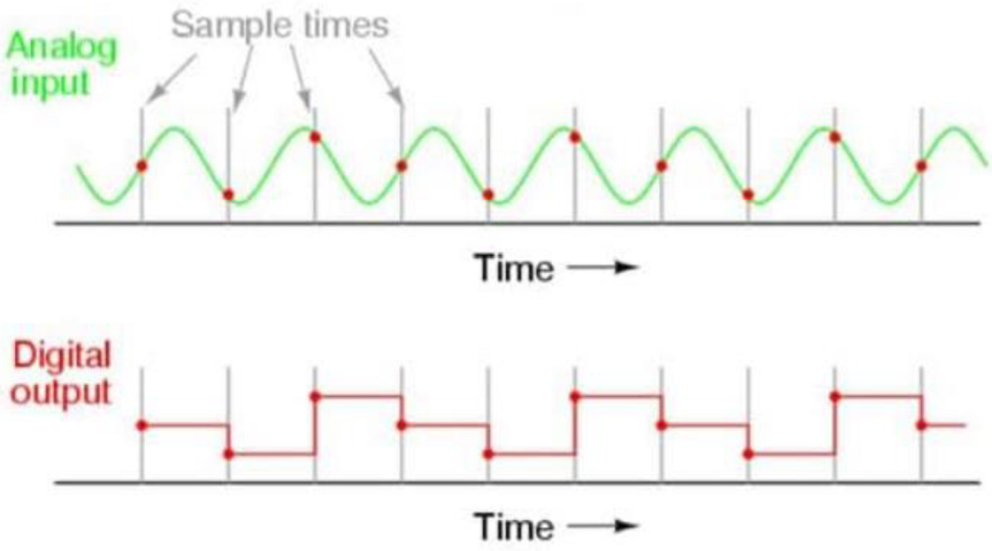
Mikrokontrolery nedokážou číst data, pokud se nejedná o digitální hodnoty. ADC nejprve navzorkuje analogový signál, následně jej kvantifikuje, aby určil rozlišení a na konec jej převede na binární hodnoty a pošle do systému.

Vzorkovací frekvence je rychlost vzorkování signálu tzv. kolik uděláme vzorků za jednu sekundu. Čím větší vzorkovací frekvence dosáhneme tím budeme mít přesnější data.

Výpočet vzorkovací frekvence

$$f_S = \frac{1}{T} \tag{2.5}$$

Jestli nedosáhneme dostatečně velké vzorkovací frekvence a frekvence signálu bude vyšší, tak nebudeme schopni rekonstruovat původní analogový signál, což povede k nesprávnému načtení dat.



Obrázek 6 - nesprávná vzorkovací frekvence [5]

Nesprávnou vzorkovací frekvenci můžeme vidět na Obrázek 6. Rekonstruovaný signál se neblíží původnímu analogovému signálu, jelikož nemáme dostatečně vysokou vzorkovací frekvenci. Digitálnímu systému bude chybět úplný obraz analogového signálu. Jako správnou vzorkovací frekvenci bychom měli alespoň používat dvojnásobek nejvyšší frekvence vstupního signálu, abychom vytvořili původní signál.

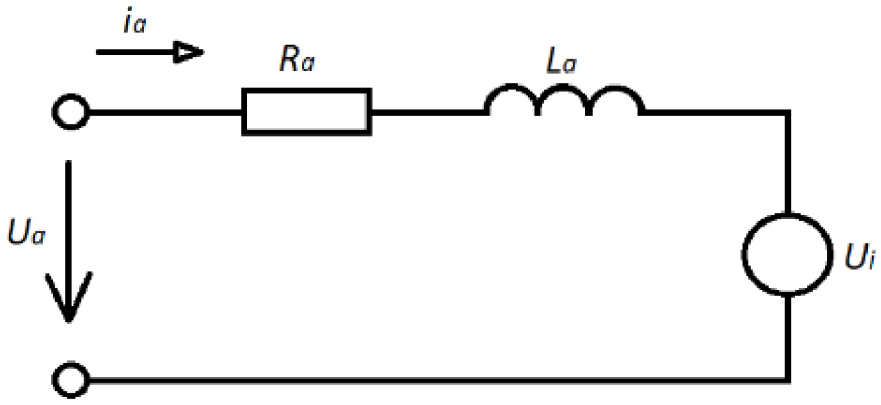
Může dojít k narušení vstupního signálu z důvodu vnějšího šumu. Abychom zamezili vstup těchto neočekávaných frekvencí do signálu používá se na vstupu antialiasingový filtr (dolní propusť) k zamezení vysokých frekvencí. [5]

2.5.2 Převodník proudu

V našem zadání využíváme proudový 20 A snímač k měření velikosti protékajícího proudu motorem. Tento snímač je dimenzován na analogový signál napětí 5 V. Z důvodu toho, že Arduino Due má analogové piny na 3,3 V se použil napěťový dělič pro snížení maximální hodnoty napětí. Oproti ostatním mikrokontrolerům používá 10 – bit hodnotu analogového signálu.

3 SIMULACE

Pro vytvoření modelu v aplikaci MATLAB Simulink bylo nutné si změřit a dopočítat některé hodnoty motoru pro použití dynamické rovnice napětí (2.3) a momentu (2.4).



Obrázek 7 - náhradní schéma stejnosměrného motoru

Nejprve byl změřen odpor vinutí motoru a následně indukčnost vinutí. Pro změření byl použit přístroj CEM DT - 9935. Pro výpočet další potřebné konstanty $C\phi$, ze vzorce (2.1) bylo zapotřební roztočit motor a změřit otáčky při daném napětí. K tomu posloužil vytvořený kód pro měření otáček motoru za pomoci enkodéru. Odpor vinutí byl při výpočtu zanedbán.

R_a [Ω]	L_a [μH]	$C\phi$ [−]
2,2	703	0,0592

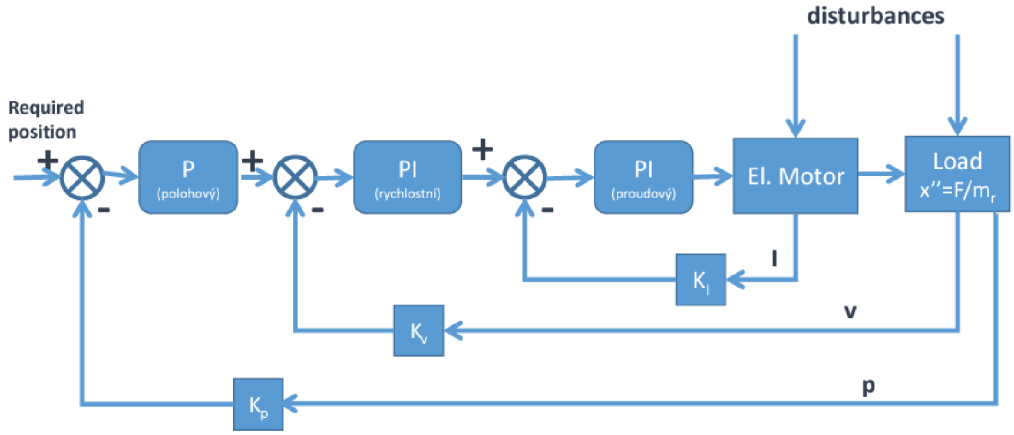
Tabulka 1 - DC motor

Posledními parametry byl moment setrvačnosti motoru, který byl obsažen v datasheetu pro daný motor a konstanta viskózního tření byla zanedbána při simulaci.

Následně byl vytvořen model stejnosměrného motoru pro řízení otáček za použití kaskádního regulátoru.

3.1 Kaskádní regulátor

Pro přesné naladění soustavy je zapotřebí vytvořit regulátor pro každou zpětnovazební smyčku regulátoru. Využíváme tzv. pomocnou regulovanou veličinu. Chceme snížit zpoždění mezi akční a pomocnou regulovanou veličinou. V podřazené regulační smyčce je regulátor proudu, v nadřazené je regulátor rychlosti nebo polohy.

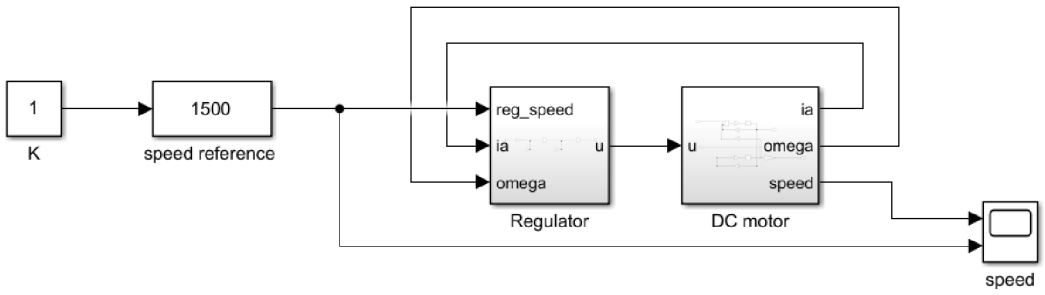


Obrázek 8 - kaskádní regulace [7]

Pro naladění regulátoru byla využita Ziegler – Nicholsonova metoda ladění. Matematický model soustavy nemusíme znát. Spočívá v experimentální metodě postupného získání P, PI a PID složky regulátoru.

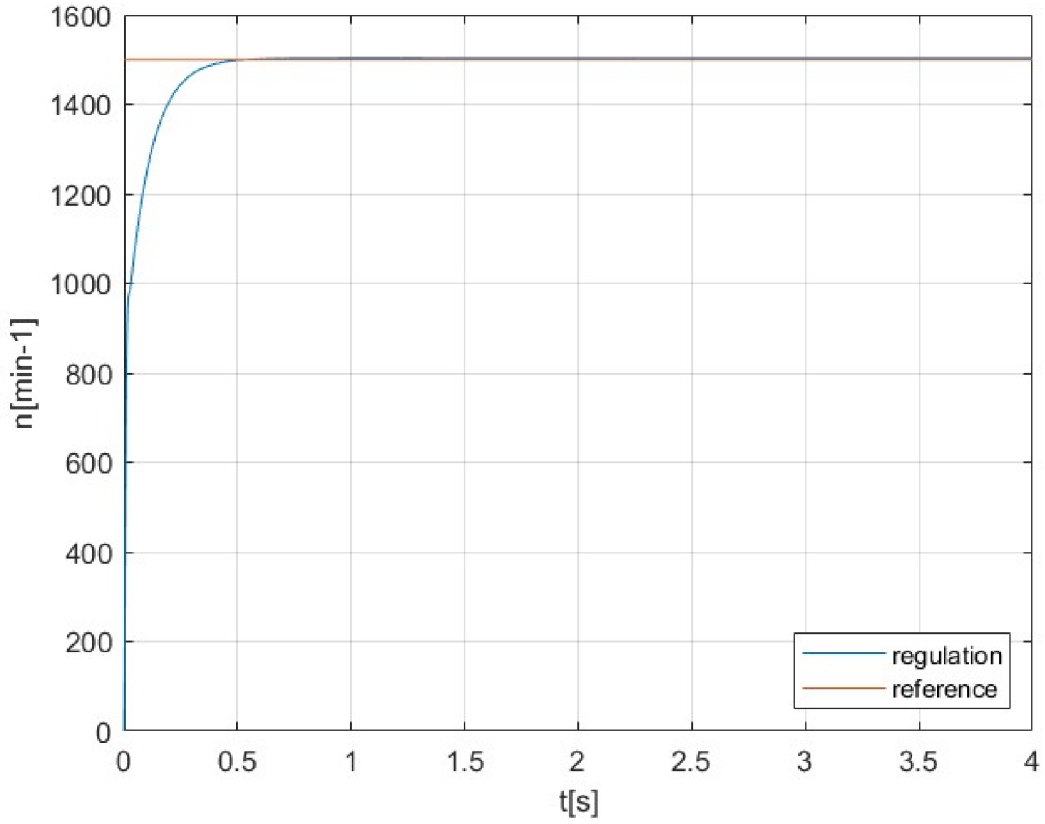
3.2 Rychlostní regulace

Prvním krokem bylo vytvoření modelu stejnosměrného motoru v prostředí MATLAB Simulink, kde byly využity získané hodnoty pro využití dynamických rovnic (2.3) a (2.4). Po vytvoření bylo zapotřebí rozšířit model o kaskádový regulátor, s tím že nám stačila proudová smyčka a jako nadřazená rychlostní. V Simulink slouží k vložení hodnot regulátoru PID blok. Bylo využit regulátorů PI, tak že se nejprve experimentálně naladila složka P a následně složka I k proudovému regulátoru. Dalším krokem bylo přidání rychlostního regulátoru, ale postup byl opět stejný. Nejprve došlo k získání složky P a následně I. Vstupním parametrem systému byly požadované otáčky a výstupem byly otáčky zregulované na požadovanou hodnotu. Celý systém můžeme vidět na obrázku níže.



Obrázek 9 - model rychlostní regulace

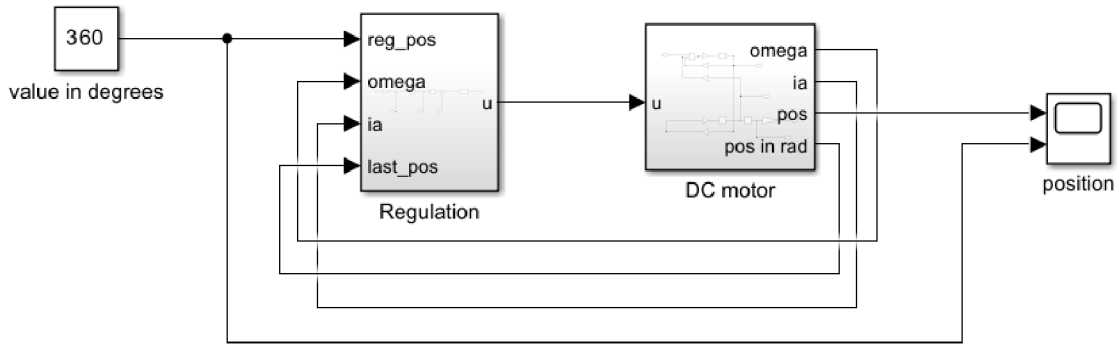
Model je navržen tak že vstupním parametrem je $n = [min^{-1}]$, jelikož tuto veličinu používáme v následujícím programu pro řízení. Požadovanou hodnotu můžeme měnit přes blok speed reference který má funkci slider gain. Ten pak vstupuje přes regulátor do DC motoru. Výslednou zregulovanou rychlost pak můžeme vidět v následujícím grafu v porovnání s referenční hodnotou.



Obrázek 10 - žádaná a zregulovaná hodnota

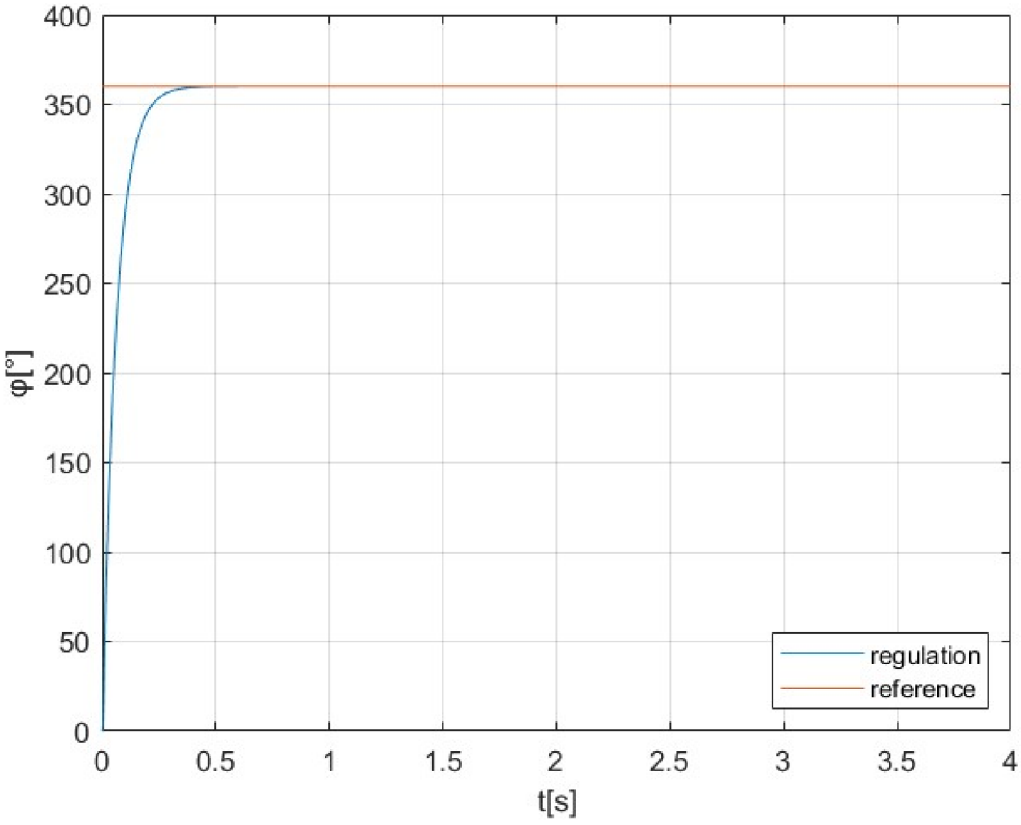
3.3 Polohová regulace

Jako vzor pro polohovou regulaci využijeme předchozí vytvořený model. Pro polohovou regulaci budeme potřebovat jenom P složku, jelikož je pro tuto aplikaci dostačující. Zbytek kaskádové soustavy zůstane stejný z předešlé rychlostní regulace. Vytvořený systém můžeme vidět na obrázku níže.



Obrázek 11 - model polohové regulace

Jako vstupní parametr modelu bylo použito $\varphi = [°]$, jelikož tuto veličinu používáme v následujícím programu pro řízení. Přes blok pojmenovaný jako value in degrees můžeme měnit požadovanou hodnotu natočení motoru. Ten pak vstupuje přes regulátor do DC motoru. Výslednou zregulovanou polohu pak můžeme vidět v následujícím obrázku v porovnání s referenční.

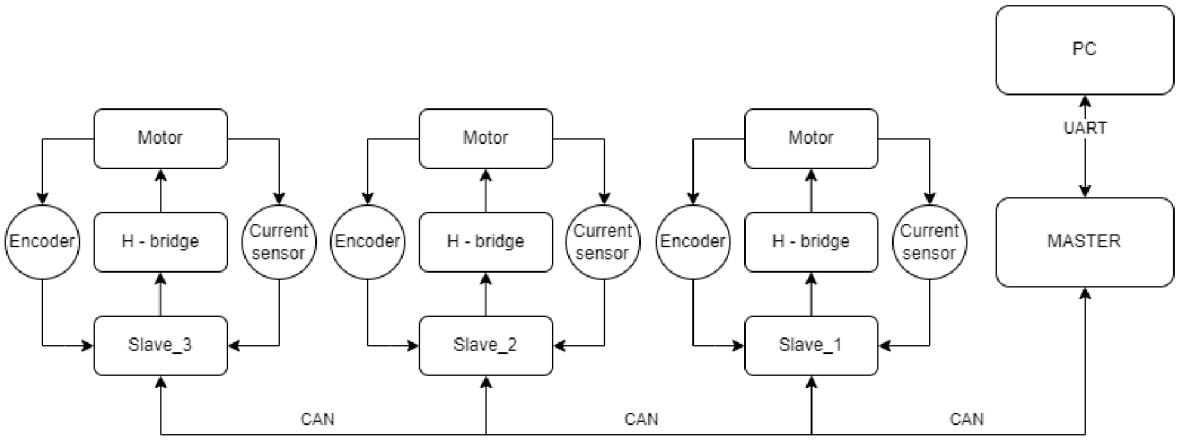


Obrázek 12 - žádaná a zregulovaná veličina

4 PROGRAM PRO ŘÍZENÍ

4.1 Hardware

Celou naši soustavu můžeme vidět na obrázku níže **Chyba! Nenalezen zdroj odkazů.**. Ta se skládá ze 3 motorových soustav. Každá soustava se skládá se shodných komponent pro řízení stejnosměrných motorů. Všechny komponenty už byly obsaženy na předem vytvořených standech. Všechny Slave a Master zařízení komunikují mezi sebou pomocí sběrnice CAN BUS. Každé jednotlivé zařízení má přiřazené své ID zpráv. Jako identifikace slouží číslo motoru. Nadřazenou smyčkou všech Slave zařízení je Master. Ten je využit jako spojovací uzel s PC. Komunikuje zároveň přes CAN BUS tak i UART. Jelikož z PC jsou odesílány zprávy po UART a následně se zakódují do CAN zprávy a odešlou. Z jednotlivých Slave zařízení chodí informace o aktuálních hodnotách do Master, zde se dekódují a opět odešlou po UART sběrnici do PC, kde jsou zpracovávány dál.



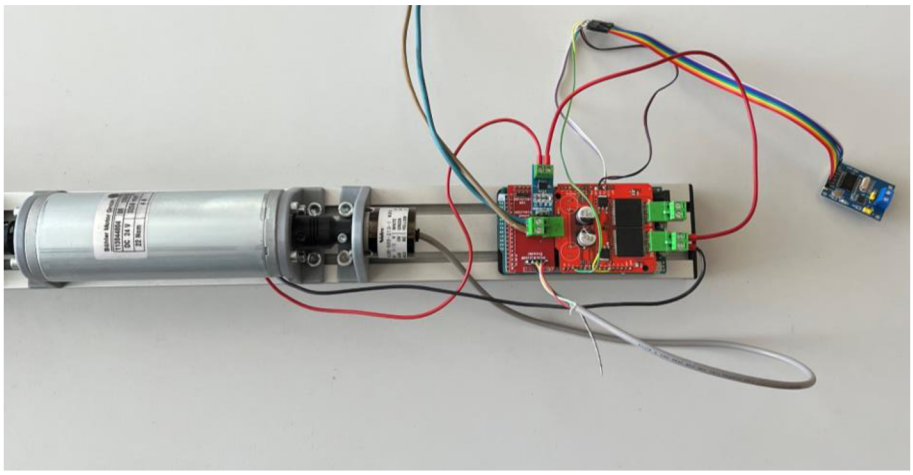
Obrázek 13 - soustava zařízení

Každý stand obsahuje jedno Arduino Due, na kterém je nahraný program pro Slave zařízení. Ten ovládá DC motor, který je na jmenovité napětí 24 V. Avšak je spínán H – můstkem jehož maximálním napětím je maximálně 16 V. Což nám lehce snižuje škálu využitelných otáček. Motor musí být zapojen na každém můstku do druhého páru svorek napájení, jelikož PWM pin na desce Arduino Due pro první motor neumí větší frekvenci PWM signálu než 1 kHz. Bylo to nedostačující pro řízení stejnosměrného motoru. Ten druhý dokáže zvládnout maximální frekvenci H – můstku, což je 20 kHz.

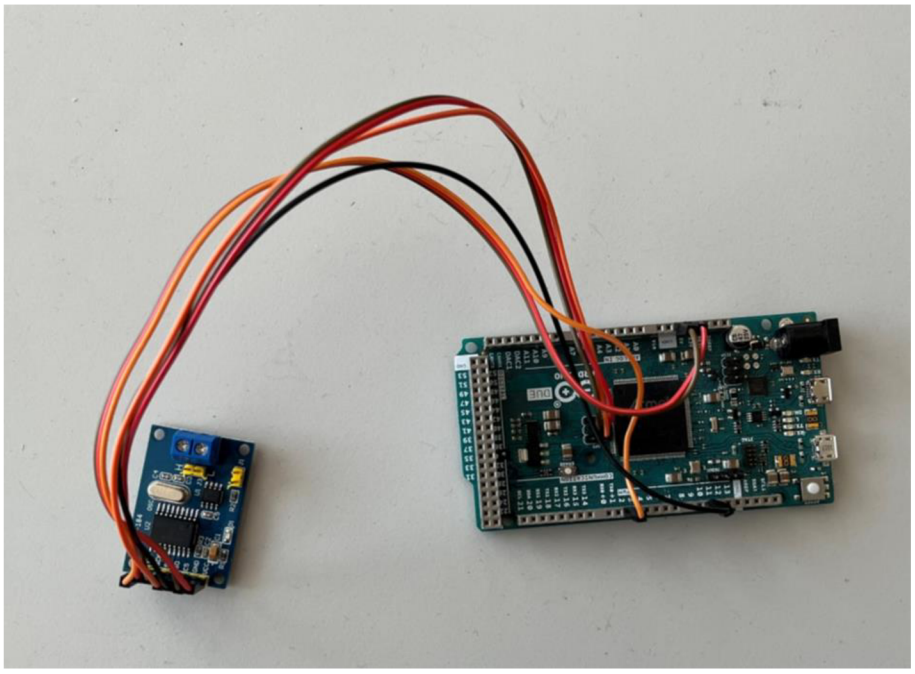
Jako zpětná vazba z motoru je využito enkodéru a proudového senzoru na 20 A. Inkrementální enkodér je 3 – kanálový. Hodnota výstupního napětí proudového snímače je 5 V, ale musela být snížena. Analogové piny Due nesnesou větší napětí, než je 3,3 V. Kdyby bylo větší mohlo by dojít k poškození desky. Pro naši aplikaci by bylo lepší použít nějaký snímač s lepším rozlišením, jelikož proudy jsou v řádech mA.

Pro přenos CAN BUS zprávy je použit převodník MCP2515, který převádí SPI na CAN. Informace z PC jsou předány po UART sběrnici do Master zařízení, kde jsou odeslány jako CAN message s náležitým ID.

Fyzické zařízení můžeme vidět na obrázku níže s výše uvedenými součástkami na už předem připravených standech.



Obrázek 14 – Slave zařízení

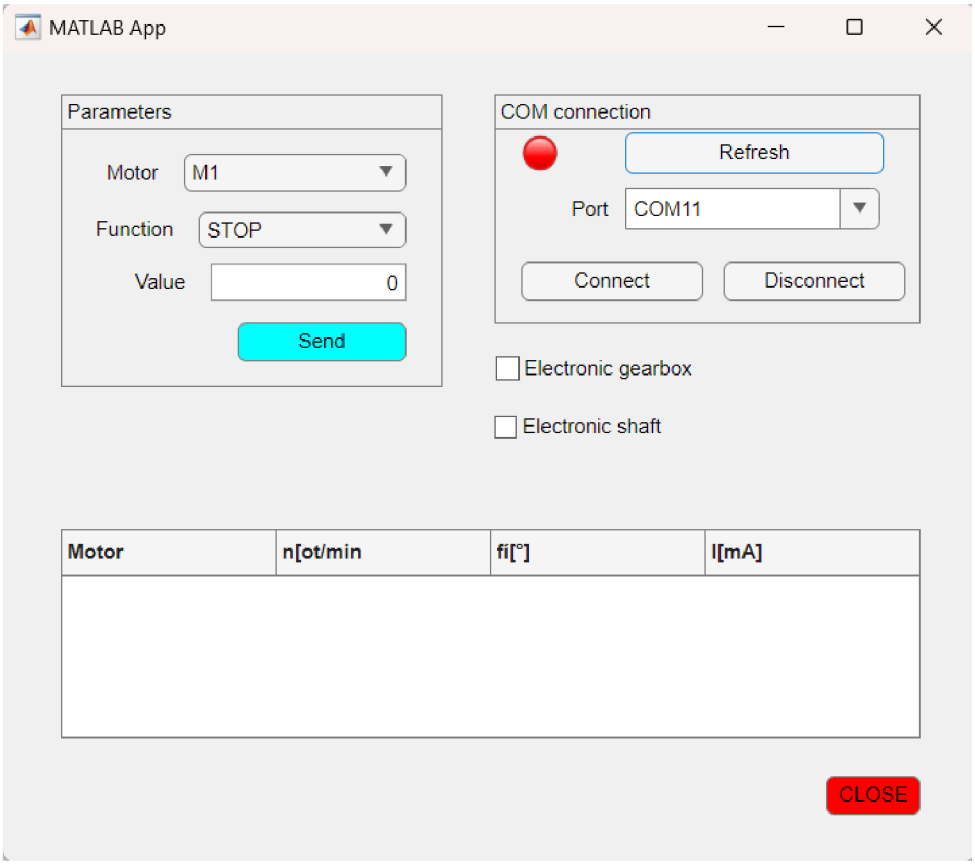


Obrázek 15 – Master zařízení

4.1.1 Aplikace

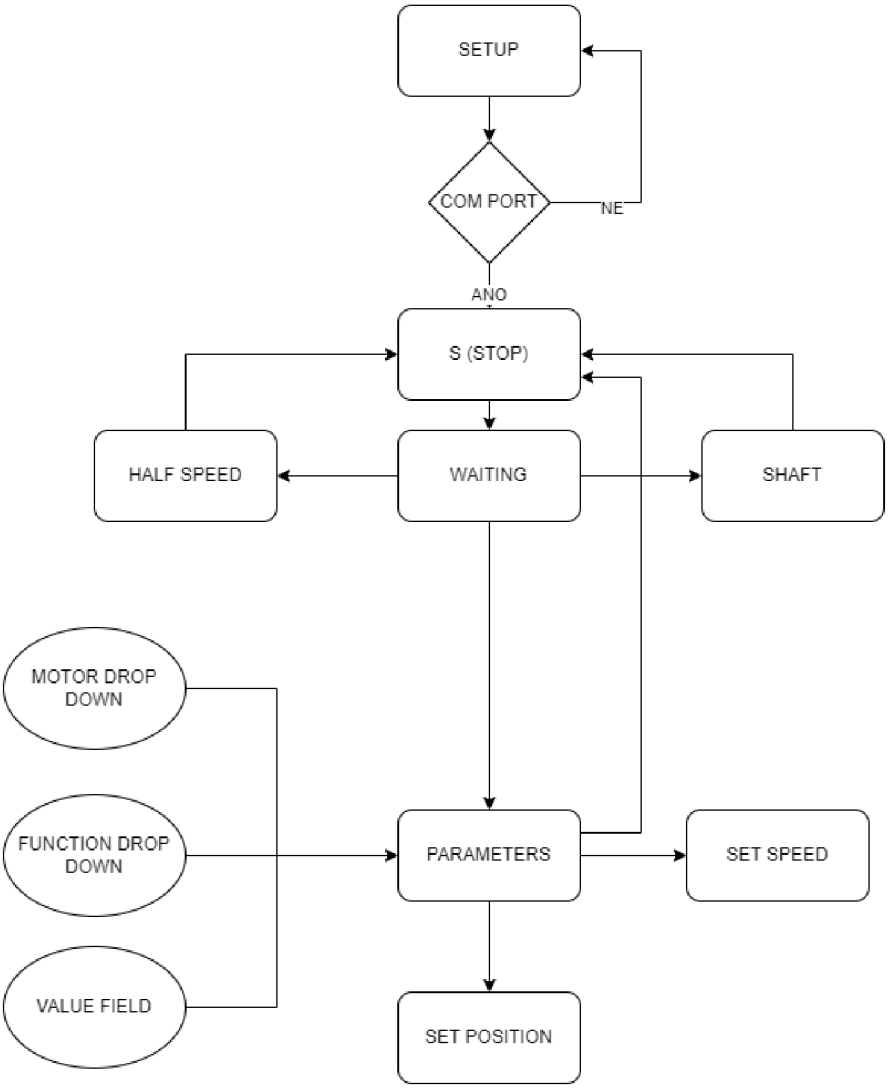
Uživatelské rozhraní bylo vytvořeno v aplikaci MATLAB App designer. Cílem bylo vytvořit nějakou jednoduchou aplikaci/GUI pro řízení soustavy motorů. Můžeme jej vidět na Obrázek 16.

Prvním hlavním krokem bylo abychom si mohli pohodlně zvolit COM port který náleží připojenému Masterovi. Kdyby nebyla možnost si změnit COM port, tak bychom to museli pokaždé složitě měnit v kódu aplikace. Celý výběr a inicializace se nachází v panelu COM connection. Je zde rozevírací nabídka, z které potřebný port můžeme vybrat. Když by nedošlo k jeho načtení můžeme použít tlačítko Refresh pro znovu načtení COM portů. To zaktualizuje aktivní COM porty připojené do PC. Pokud se nepřipojíme k danému portu tak nelze nic ovládat, odesílat nebo přijímat. Po vybrání portu a stisknutí tlačítka Connect dojde k navázání komunikace s Arduinem a otevření sběrnice UART. Sběrnice funguje obousměrně jelikož z PC budou chodit požadavky do jednotlivých Slave zařízení a do něj aktuální hodnoty parametrů jako je proud, rychlost a pozice. Tyto hodnoty budou zobrazeny v tabulce ve spodní části uživatelského rozhraní.



Obrázek 16 - Uživatelské rozhraní

Teď už můžeme začínat ovládat jednotlivá Slave zařízení, jak budeme chtít. Z obrázku můžeme využít blok Parameters, kde volíme, který motor a co za funkci má vykonat. Na výběr máme ze 3 motorů. Každému motoru můžeme poslat požadavek na rychlostní nebo polohovou regulaci nebo příkaz STOP. Když vybereme jednu z možností regulací tak v poli Value napíšeme požadovanou hodnotu otáček, která je v min^{-1} nebo požadovanou hodnotu natočení která je ve stupních. Můžeme vepisovat hodnot kladné tak záporné. Následně po zmáčknutí tlačítka Send se příkaz odešle a zpracuje jako CAN message. Když budeme chtít zastavit motor tak budeme postupovat obdobně stačí vybrat z rozevíracího seznamu příkaz STOP a následně odeslat. Když by náhodou byla nějaká hodnota v poli Value, tak dojde i tak k zastavení motoru, jelikož ta nemá vliv na STOP funkci.



Obrázek 17 - stavy uživatelské aplikace

Pokud bychom chtěli vybrat funkci elektronické převodovky, tak po vybrání volby se všechny motory uvedou do STOP stavu. Bude možné zapsat hodnotu do pole Value která se nebude přepočítávat. Do motoru č. 1 se odešle požadavek na zadanou hodnotu a motoru č. 2 se odešle požadavek na poloviční hodnotu té zadané. Mezi odesláním požadavku na oba motory je krátká pauza. Když bude vybrána tahle volba tak nebude možné vybrat jinou funkci. Musí se opět odkliknout tato, kde dojde opět ke STOP všech motorů a pak můžeme vybrat jinou funkci.

Jestliže vybereme funkci elektronické hřídele tak opět uvede, pokud nejsou všechny motory do STOP stavu. Když je vybrána tato volba tak nemůžeme vybrat jinou funkci. Následně je odeslána zpráva pro motor č.1 ať změní řídící funkci na odesílání polohy 'H'. Ta vlastně přepne řídící funkci motoru č. 2 na regulaci polohy. Pokud motor č. 1 rukou pootočíme tak se aktuální pozice odesílá do motoru č. 2, která na ni reguluje. Zpráva se odesílá jen tehdy když je nadcházející poloha jiná než předchozí. Pro vypnutí musíme opět odkliknout box a STOP požadavek se odešle do všech motorů.

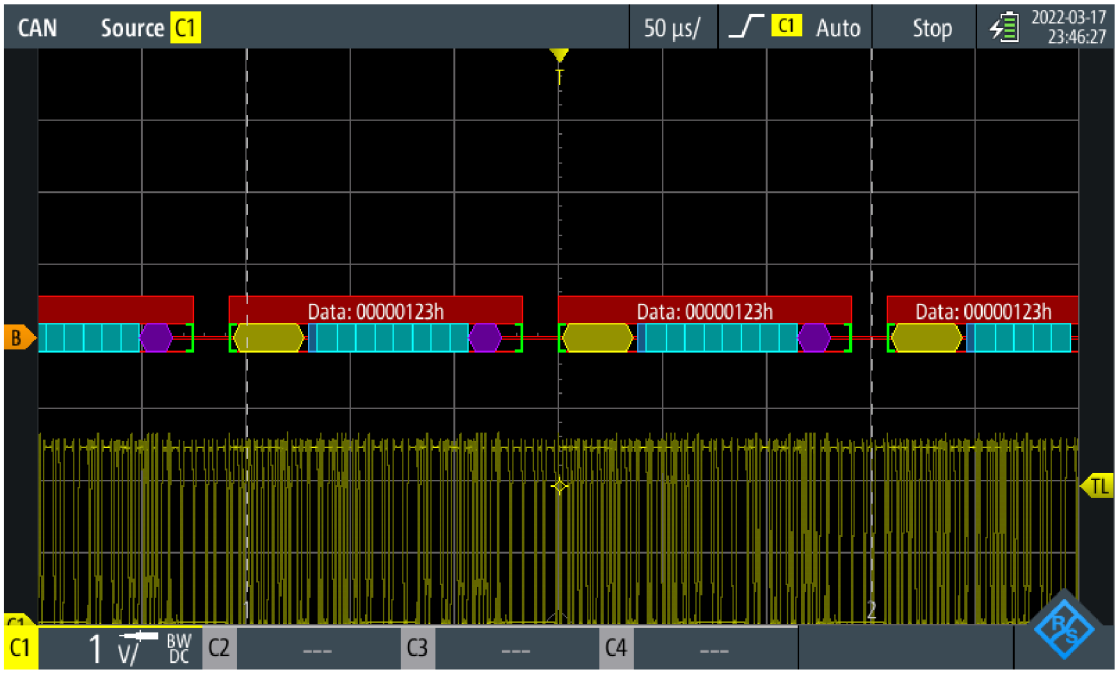
Načítání dat funguje po celou dobu připojení COM portu. Pokud zvolíme tlačítko Disconnect, tak dojde k odpojení COM portu a veškeré motory se uvedou do stavu STOP. Uzavře se sběrnice a dojde k vymazání připojeného zařízení z paměti aplikace. Pro ukončení aplikace použijeme červené tlačítko Close, které je ve spodní části aplikace. Po zavření aplikace opět dojde k ověření, jestli jsou všechny motory uvedeny ve stavu STOP a následnému smazání dat k připojenému zařízení z paměti MATLAB. Tímto je ošetřena možnost, kdyby před uzavřením uživatelské aplikace neproběhlo odpojení COM zařízení.

4.2 Master

Zařízení, které se nachází mezi motorovými standy a PC. Každé jednotlivé Arduino Due bylo programováno pomocí nástroje Arduino IDE, jelikož obsahuje široké spektrum již vytvořených knihoven. Pro CAN sběrnici je využita knihovna [6]. Na začátku proběhne inicializace CAN sběrnice, která je nastavena na *250 kbit/s*. Poté se začne vykonávat hlavní kód. V něm se nachází hlavní řídící funkce, která je vykonána, když je odeslán nějaký požadavek z PC.

Pokud dojde k příjmu zprávy, tak dojde k přečtení a uložení do String value. K tomu je využita knihovna [9]. Následně se zpráva rozparsuje do proměnných, a zní pak vybíráme požadovanou funkci pro řídící strukturu switch case. Čísla pro různé funkce jsou striktně daná, čísla motoru se zapisují na první místo ID zprávy. Tím je zaručena multifunkčnost zařízení při budoucím přidání dalších zařízení. Před odesláním zprávy se vytvoří 64 - bitové pole. Do něj se uloží jednotlivé proměnné. Pro Master je to požadovaná hodnota pro rychlost nebo polohu. Ze zařízení Slave to bude aktuální proud, rychlost a poloha. Až se data uloží, tak se vytvoří CAN zpráva s náležitým ID, délkou pole dat a proměnná kde jsou tyto data uloženy. Je možné využít i funkci Extended ID, ale v naší aplikaci tolik ID zpráv nemáme. Následně se data odešlou na CAN sběrnici. Každý motor má přiřazené své ID zpráv (1> 0x10+, 2>0x20+…).

Po celou dobu vykonávání kódu probíhá příjem zpráv ze všech Slave zařízení, které jsou odesílány každých 50 ms. Zpráva se přijme a uloží do vytvořeného bufferu. Následně se celá zpráva dekóduje, rozparsuje a odešle se na sběrnici UART do PC. CAN bus zprávu můžeme vidět na obrázku níže po použití analyzéru CAN.

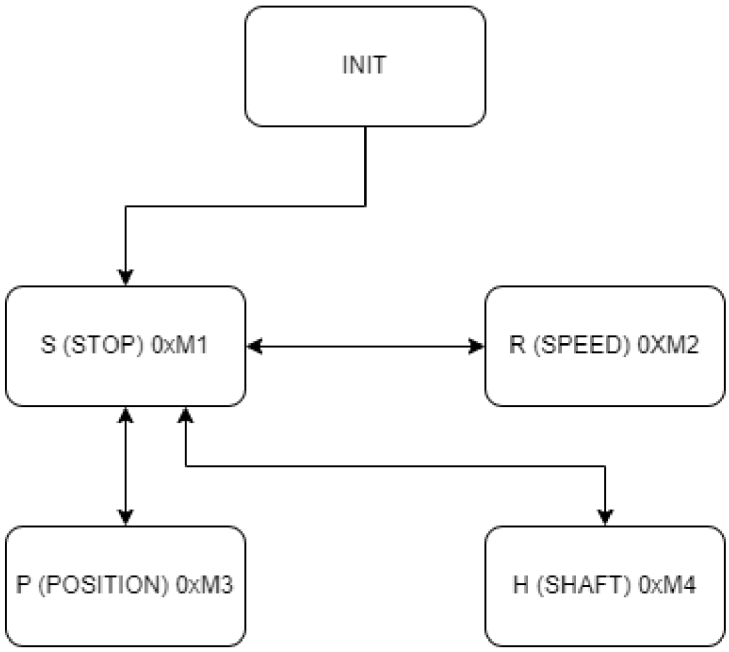


Obrázek 18 - dešifrování zprávy CAN

4.3 Slave

Na začátku opět proběhne inicializace CAN BUS na rychlost *250 kbit/s*. Při prvním spuštěné zůstává v režimu waiting dokud nepřijme nějaký požadavek. Přijmutí zprávy probíhá ve funkci dataProcess. Nejprve se ověří, jestli daná zpráva je opravdu určena pro daný motor. Poté se jednotlivé hodnoty zapíší do globálních proměnných. Po vykonání této funkce je vybrána požadovaná funkce z řídící struktury switch case. Zde je na výběr z několika funkcí jako je STOP, rychlostní regulace, regulátor polohy a odesílání polohy jako řídící signál do druhého motoru. Po celou dobu vykonávání hlavního kódu je měřen proud pomocí snímače. Každých 50 ms se odesílají aktuální hodnoty motoru do nadřazeného zařízení čímž je Master. Je na to využita funkce sendData. Kde se opět proměnné uloží do bufferu, který je pak odeslán s náležitým ID na CAN sběrnici.

Když je vybrána funkce ((motor_ID <<4) |2), tak č. 2 náleží regulátoru rychlosti. Zde využíváme jenom rychlostní regulátor, jelikož proudový je pro nás velice nepřesný. Požadavkem bylo opět pužít kaskádní regulaci 3.1 jako v předešlé simulaci. Jenže jelikož naše využití nemá nijak zatížené motory, tak pracují na hraně rozlišitelnosti proudového snímače, takže proudový regulátor by byl pro nás velice nepřesný. Pro budoucí využití, je zde vytvořený když budou proudu v řádu ampér. Nastavení konstant regulátoru opět probíhalo Ziegler – Nicholsonova metodou. Jestli budeme chtít změnit funkci na polohovou regulaci tak musím jít jen přes řídící člen STOP.



Obrázek 19 - stavový automat

Funkce pro regulaci polohy funguje obdobným způsobem, kde je vybrána č. 3 (((motor_ID <<4) |3). Je opět aplikována kaskádní regulace 3.1. Zde už je použito dvou regulátorů. Tím je rychlostní regulátor a polohový regulátor, s tím že proudový je opět zanedbán kvůli nepřesnosti pro naše řízení. Pro budoucí využití je vytvořen. Pokud přijde požadavek na rychlostní regulaci, tak je to ošetřeno podmínkou. Možné je to jenom přes STOP.

Čtvrtou funkcí motoru je elektronická hřídel. Identifikace této zprávy je č. 4 ((motor_ID <<4) |4). Zařízení nyní funguje jako Master pro druhý stand. Načítá se zde aktuální poloha pozice. Hodnota o kterou se tento motor pootočil se odešle jako požadavek do druhého motoru, aby se pootočil stejně. Požadavek na změnu polohy se neodesílá neustále, jen pokud se liší aktuální poloha od té předchozí. Z tohoto stavu se opět lze dostat jen přes STOP case. Zabránění zvolení jiné funkce je uděláno v uživatelském rozhraní. Ačkoliv je tato funkce použita na všech zařízeních, je využívána jen na standu č. 1 a č. 2 přijímá požadavky o změně.

4.3.1 Poloha

Snímání polohy využívá funkce inkrementálního enkodéru, který má *500 dílků/ot* a je 3 – kanálový. Načítání probíhá na náběžnou hranu kanálu A jako interrupt. Když si přečtu stav kanálu B tak zjistím směr otáčení, takže buď inkrementuji nebo dekrementuji absolutní pozici v dílkách. Následně je hodnota zapsána do globální proměnné.

4.3.2 Rychlost

Pro výpočet rychlosti musíme znát aktuální a předchozí čas měření. Je zapotřebí také určit aktuální hodnotu pozice enkodéru a tu předešlou. Toto je nezbytné pro výpočet následné rychlosti. Posledním krokem je přepočítání na žádané jednotky, což v našem případě bude n $[min^{-1}]$. Následně bylo nutné filtrovat hodnoty, jelikož vstupovali do regulátoru. Na to byl použit low – pass filtr [8].

4.3.3 Proudový senzor

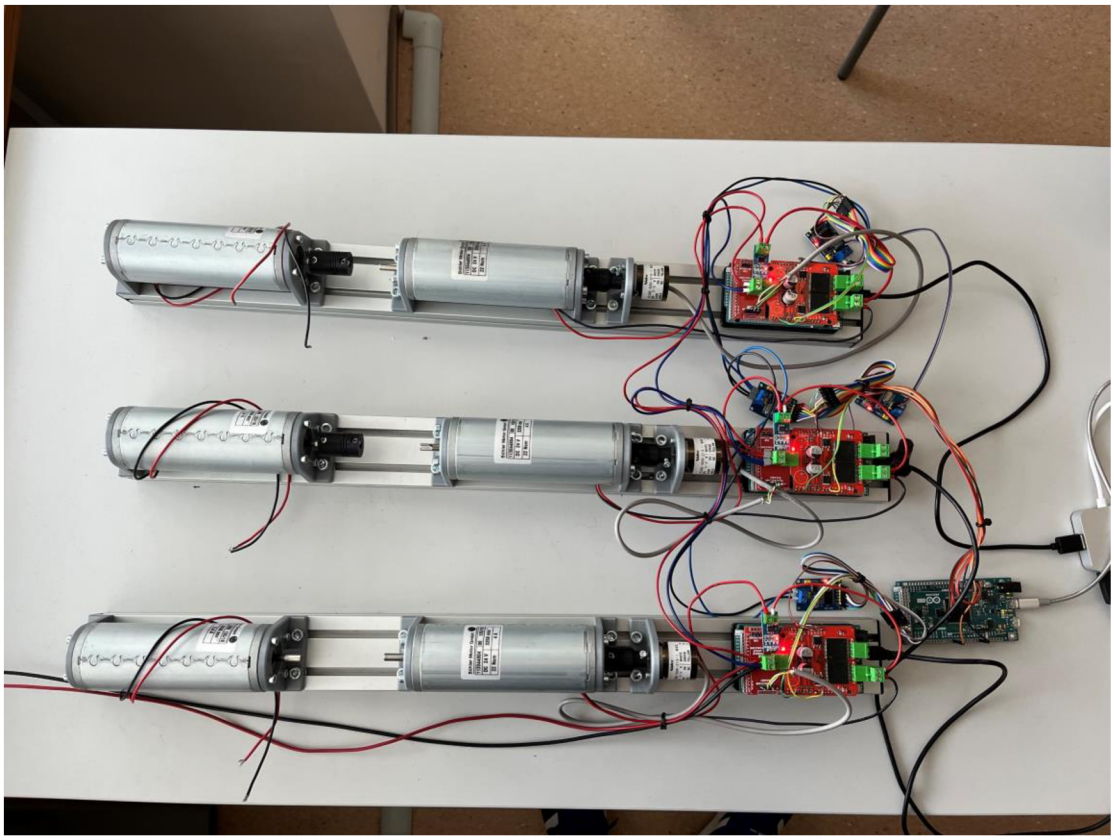
Pro výpočet proudu načítáme hodnotu analogového pinu, kterou filtrujeme použitým filtrem. Od následné hodnoty odečteme offset analogového pinu a přepočítáme na napětí přes rozlišení pinu. Arduino Due má defaultně nastavené piny na 10 – bit hodnoty. Avšak pro přesnější měření bylo přenastaveno na začátku inicializace na 12 – bit. Avšak naše hodnoty proudu byly při měření velice malé, na úrovni rozlišitelnosti a šumu, což pro využití regulátoru není žádoucí.

4.3.4 Nastavení motoru

V této funkci dochází k odeslání dat do H – můstku. Z jednotlivých řídících funkcí sem vstupuje velikost střídy a odesílá se na analogový pin. Analogová hodnota nesmí být větší než 255 a záporná. Obě varianty jsou v kódu ošetřeny. Zde dojde k rozřazení směru otáčení motoru, jestli dopředu nebo dozadu. Následně se zapíše stav na digitální piny H – můstku.

5 VYHODNOCENÍ

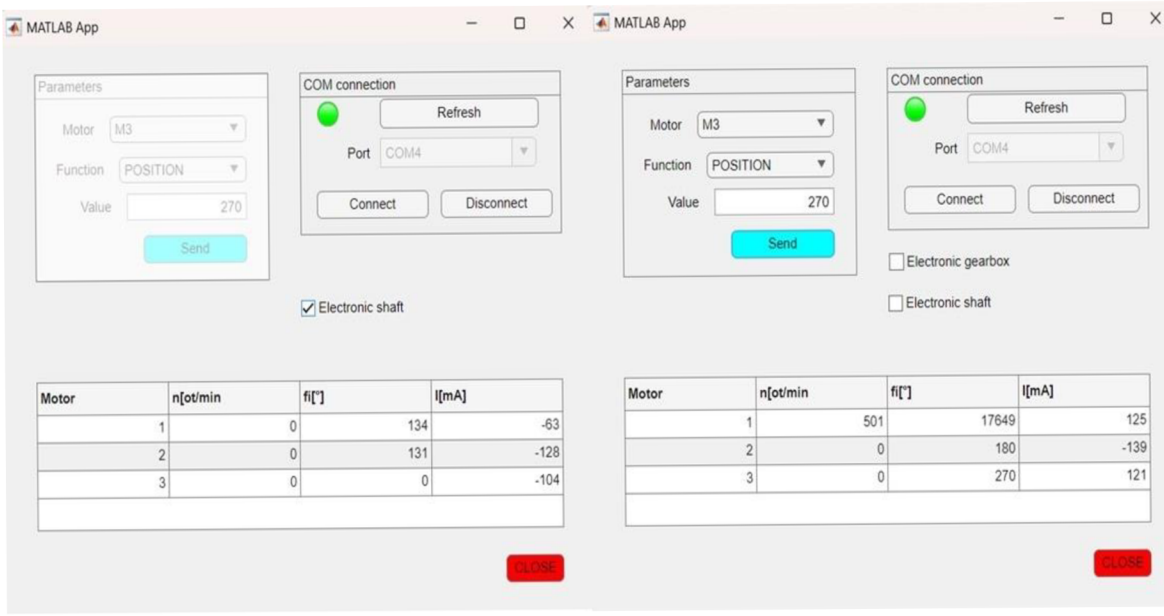
Cílem bylo řídit více stejnosměrných motorů pomocí sběrnice CAN. Nevýhodou tohoto daného řízení je že musíme využít Master jako převodník z UART na CAN bus. Nejlepší by bylo, kdyby se zprávy odesílaly přímo z uživatelského rozhraní už jako CAN. Takže by bylo ideální využít nějaký PC převodník na CAN BUS. Velikou výhodou této sběrnice je rychlost přenosu a jednoduchá implementace dalších zařízení do soustavy s využitím vytvořeného kódu. Když budeme chtít rozšířit sběrnici tak připíšeme jen ID motoru a zapojíme další zařízení.



Obrázek 20 – celková soustava použitých zařízení

Na Obrázek 21 můžeme vidět funkčnost aplikace. Byl zde požadavek na elektronickou hřídel, kde jeden motor řídí ten druhý. Je zde malá odchylka od žádané, ale to by se mohlo doladit zpřesněním konstant regulátoru. Na druhé polovině jsou spuštěny všechny tři motory, přičemž na standu č. 1 byl požadavek na rychlostní regulaci a na zbylých polohovou regulaci. Z obrázku je patrné že proudy jsou vyšší, než co je skutečný. Dalo by se to vyřešit lepším filtrováním hodnot nebo použitím jiného proudového snímače. S tím souvisí i příjem rychlosti z jednotlivých zařízení, jelikož se neustálil na konkrétní, ale byl v rozsahu jednotek okolo hodnoty žádané. Mohlo by se to vyřešit zase využitím nějakého lepšího low – pass filtru.

Funkce elektronické hřídele a převodovky jsou přiřazeny přímo motoru č. 1 a 2, což je možným nedostatkem. Zlepšením by bylo vytvoření funkce pro výběr, které motory mají být namapovány na tento příkaz.



Obrázek 21 - ukázka ovládání uživatelského rozhraní

6 ZÁVĚR

Bakalářská práce se zabývala řízením soustavy stejnosměrných motorů pomocí sběrnice CAN BUS. Následný program pro řízení soustavy motorů přes CAN BUS byl vytvořen v aplikaci Arduino IDE. Uživatelské rozhraní pro ovládání bylo nakonfigurováno v aplikaci MATLAB App Designer.

První část bakalářské práce se zabývá rešerší na téma řízení stejnosměrných motorů. Je to zpracováno stručně jen jako přiblížení problematiky.

Další část se zabývala simulací. Zde jsme si museli dopočítat nebo změřit některé konstanty motoru abychom mohli využít dynamických rovnic stejnosměrného motoru. Následná simulace byla vytvořena v prostředí MATLAB Simulink, kde byl vytvořen model DC motoru. Následovalo postupné nastavení regulátorů Ziegler – Nicholsonovou metodou.

Prvním a hlavním bodem praktické části bylo řídit motory pomocí CAN BUS sběrnice. Následně se kód rozrostl o funkci rychlosti, snímaní polohy pomocí enkodéru, snímaní proudu a ovládání motoru pomocí plného H – můstku. Tímto hardwarem už byly standy osazeny. Pro snímání proudu by bylo ale dobré využít jiný snímač, který by zvládl i velice malé proudy, jelikož jsme ho v naší aplikaci nemohli použít v regulátoru proudu. Uživatelské rozhraní bylo vytvořeno za pomoci nástroje MATLAB App Designer, který komunikuje po UART s Master. Zde jsou také načítány příchozí hodnoty z jednotlivých Slave zařízení. V rozhraní jsou dvě funkce jako je elektronická hřídel a převodovka na které by mohlo být implementována, aby si uživatel vybral, co chce, aby jaký udělal.

Soustava zařízení by mohla být vylepšena o další senzory jako je například sensor vibrací. Možným zlepšením by mohlo být také ovládání Master zařízení přes WI-FI, což by nám umožnilo nebýt závislý na připojení přes UART.

7 LITERATURA

[1] VODA, Zbyšek. Průvodce světem Arduina. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

[2] AST-CAN485 Hookup Guide. SparkFun [online]. SparkFun Electronics ®, 2018-01 [cit. 2023-05-21]. Dostupné z: https://learn.sparkfun.com/tutorials/ast-can485-hookup-guide

[3] CAN high / CAN low. Squarell Support Portal [online]. Squarell Support, 2013 [cit. 2023-05-21]. Dostupné z: https://support.squarell.com/index.php?/Knowledgebase/Article/View/94/0/can-high--can-low

[4] FRANZ, Kaitlyn. What Is an H-Bridge? Diligent Blog [online]. Digilent, 2023 [cit. 2023-05-21]. Dostupné z: https://digilent.com/blog/what-is-an-h-bridge/

[5] Analog-to-Digital Converters Basics. Arrow [online]. Arrow Electronics, 2018 [cit. 2023-05-21]. Dostupné z: https://www.arrow.com/en/research-and-events/articles/engineering-resource-basics-of-analog-to-digital-converters

[6] Seeed_Arduino_CAN. *Github* [online]. GitHub, Inc., 2023 [cit. 2023-05-24]. Dostupné z: https://github.com/Seeed-Studio/Seeed_Arduino_CAN

[7] *Kaskádní regulace* [online]. [cit. 2023-05-25]. Dostupné z: https://cw.fel.cvut.cz/old/_media/courses/drr/kaskadni_regulace.pdf

[8] *SpeedControl_NoAtomic.ino* [online]. [cit. 2023-05-26]. Dostupné z: https://github.com/curiores/ArduinoTutorials/blob/main/SpeedControl/SpeedControl_NoAtomic/SpeedControl_NoAtomic.ino

[9] *Arduino-string* [online]. [cit. 2023-05-26]. Dostupné z: https://github.com/asukiaaa/arduino-string

8 SEZNAM OBRÁZKŮ A GRAFŮ

SEZNAM PŘÍLOH

Elektronické přílohy:

- složka **2023-05-26_Arduino+App** – obsahuje jednotlivé kódy pro Master a Slave zařízení. Je v něm obsažena i uživatelská aplikace pro ovládání motorů.
- složka **2023-05-26_MATLAB_Simulink** – obsahuje modely DC motoru s následnou rychlostní nebo polohovou regulací. Main obsahuje hlavní konstanty pro modely a vykresluje grafy.