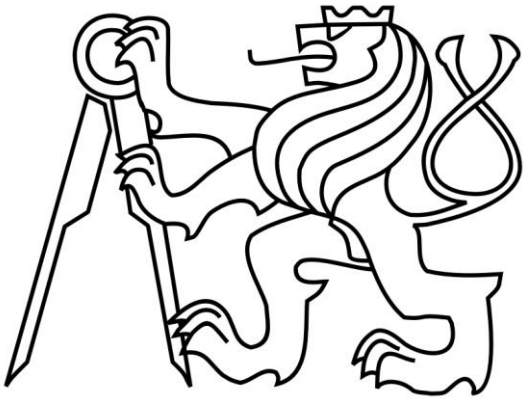
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STROJNÍ



BAKALÁŘSKÁ PRÁCE

Návrh ovládání trimovací plošky ultralehkého letadla

Autor: Karel Pelc

Vedoucí práce: Ing. Tomáš Čenský, Ph.D.

Praha, 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pelc** Jméno: **Karel** Osobní číslo: **457519**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav letadlové techniky**

Studijní program: **Strojírenství**

Studijní obor: **Konstruování podporované počítačem**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Návrh ovládání trimovací plošky UL letounu

Název bakalářské práce anglicky:

Design of the VLA airplane trim control

Pokyny pro vypracování:

Konstrukční a elektronický návrh přístroje ovládání podélného elektrického trimování ultralehkého letounu pro konfiguraci s řízením uspořádaným za sebou.
1. Proveďte rešerši v oblasti ovládání trimu letadel.
2. Popište způsoby elektrického ovládání trimovacích ploch.
3. Proveďte koncepční návrh elektronického ovládání podélného trimu, navrhněte řešení pro uspořádání pilotů za sebou.
4. Proveďte úplný konstrukční návrh elektroniky ovládání trimu.

Seznam doporučené literatury:

Dle pokynů vedoucího

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Tomáš Čenský, Ph.D., ústav letadlové techniky FS

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **30.04.2019** Termín odevzdání bakalářské práce: **02.08.2019**

Platnost zadání bakalářské práce:

Ing. Tomáš Čenský, Ph.D.	Ing. Robert Theiner, Ph.D.	prof. Ing. Michael Valášek, DrSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

30.4.2019
Datum převzetí zadání

Podpis studenta

Prohlášení

Prohlašuji, že jsem předloženou bakalářskou diplomovou práci zpracoval samostatně a souhlasím s tím, že její výsledky mohou být dále využity dle uvážení vedoucího této práce Ing. Tomáše Čenského, PhD. jako jejího spoluautora.

Datum…… Podpis……

Poděkování

Mé poděkování patří Ing. Tomášovi Čenskému, PhD. za odborné vedení, trpělivost, ochotu a zapůjčení zkušebních komponent, které daly možnost vzniku nového trimovacího systému a splnění cíle práce.

Abstrakt

Bakalářská práce se zabývá problematikou trimování neboli vyvažování ultralehkých letadel, primárně z pohledu ovládání. Z počátku se práce zaobírá trimováním na teoretické rovině, což zahrnuje principy, důvody trimování, stávající řešení ovládání, indikace polohy atp. Dále se práce zabývá vlastním návrhem elektrického trimu v kreativním a moderním kontextu pro ultralehký letoun, který má sedadla pilotů řazeny za sebou. Součástí rozboru návrhu jsou rešeršní texty o využitých segmentech systému, hlavní částí je práce zaměřena na samostatný návrh ovládání přes ovládací prvky, servo, zpětnou vazbu pro pilota, řízení pomocí programovatelné mikrokontrolérové desky a využití datové sběrnice letadla. Pro vlastní finální verzi je v práci popsán postup vývoje hardwarové desky a konstrukce serva za využití CAD prostředí.

Abstract

The bachelor thesis deals with trim control of ultralight aircraft. Theory of trim theme, principles, informations about usage, current system solution and indicator design are located at the beginning of the thesis. This primary topics are followed by a new electrical trim system designing in a modern and creative context. The designed system is determinated for ultralight aircraft which has pilot´s seats located inline. Design part of the thesis also includes reseach about used segments, the main section deals with technical solusion of control, motion servo design and trim indicator. Trim is controled by designed unique microcontroller board which is connectable to CAN. The last part of the thesis shows CAD designing process of servo and the microcontroller board, these components are used in the final version of a draft.

Obsah

1 Úvod do problematiky trimování 11
1.1 Trimování a ovládání letounu 11
1.2 Ovládání ploch 12
1.3 Princip trimování 14
1.4 Trim výškového kormidla 15
1.5 Popis ovládání 17
2 Prostředky elektronického ovládání plošky 17
2.1 Základní přehled 17
2.2 Praktický rozbor 17
2.2.1 Stejnosměrný motor 17
2.2.2 H můstek 18
2.2.3 CAN 18
2.2.4 ARINC 23
2.3 Ovládání prostřednictvím mikroprocesorů 23
2.3.1 Arduino obecně 23
2.3.2 Způsob programování 24
2.3.3 Jazyk C 25
2.3.4 Arduino UNO 25
2.3.5 Arduino DUE 26
3 Praktický návrh 26
3.1 Rešeršní verze systému 27
3.1.1 Základní informace 27
3.1.2 Řešení indikátoru polohy 27
3.1.3 Seznam použitých komponent 28
3.1.4 Zapojení 28
3.1.5 Program 30
3.2 Návrh zkušební verze systému 31
3.2.1 Základní informace 31
3.2.2 Řešení Indikátoru polohy 32
3.2.3 Seznam zkušebních komponent 34
3.2.4 Zapojení zkušebních modelů 34
3.2.5 Programy zkušebních modelů 39
4 Akční členy 40
4.1 Návrh serva 40
4.1.1 Převodovka 40
4.1.2 Motor 42

4.2 Konstrukce serva ... 44
4.2.1 Rozložený pohled ... 45
4.2.2 Obecný seznam částí ... 47
4.2.3 Popis montáže ... 47
5 Návrh elektronické desky ... 48
5.1 Úvodní přehled ... 48
5.2 Postup návrhu ... 48
5.2.1 Procesor ... 48
5.2.2 Napájení a ochrana ... 51
5.2.3 H můstek ... 51
5.2.4 CAN bus ... 52
5.3 Design desky a rozmístění komponent ... 52
6 Finální podoba ... 55
6.1 Přehled návrhu ... 55
6.2 Nástin instalace do letadla ... 56
7 Závěr ... 56
Seznam zdrojů ... 57
Zdroje samostatně převzatých obrázků z webu ... 60
Software ... 62
Seznam obrázků ... 63
Seznam tabulek ... 64
Seznam příloh ... 65

Použité zkratky a jejich význam v této práci

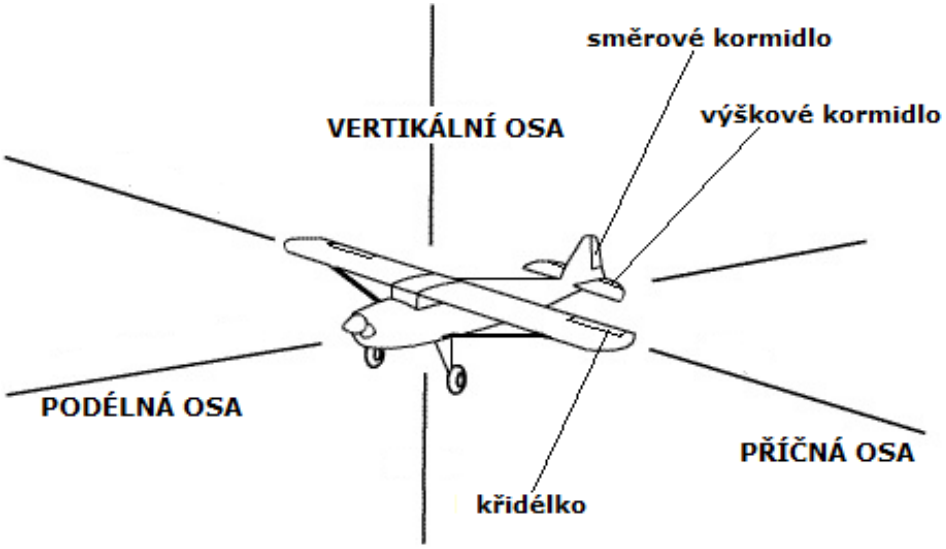
ARINC	Aeronautical Radio, Incorporated
CAN	Controller Area Network, Sériový komunikační protokol
D/A	Digitálně analogový převodník
DN	Down, Dole
GND	Ground, Země
H	High, Horní
I	Input, Vstup
ICSP	In-System Programming, Možnost programování uvnitř obvodu bez nutnosti vyjmutí
ISO	International Organization for Standardization, Mezinárodní organizace zabývající se tvorbou norem
L	Low, Spodní
LCD	Liquid crystal display, Displej z tekutých krystalů
LED	Light-Emitting Diode, Elektroluminiscenční dioda
LLC	Logical Link Control, Podvrstva řízení datového spoje
MAC	Medium Access Control, Hardwarově závislá podvrstva linkové vrstvy
O	Output, Výstup
OSI	Open Systems Interconnection Basic Reference Model, Model síťové komunikace a protokolů
PCB	Printed circuit board, Deska plošných spojů
PWM	Pulse Width Modulation, Pulzně šířková modulace
SPI	Serial Peripheral Interface, Sériové periferní rozhraní
TO	Take off, Vzlet
UART	Asynchronous Receiver and Transmitter, Asynchronní sériové rozhraní
UP	Up, Nahoře
USB	Universal Serial Bus, Univerzální sériová sběrnice

1 Úvod do problematiky trimování

1.1 Trimování a ovládání letounu

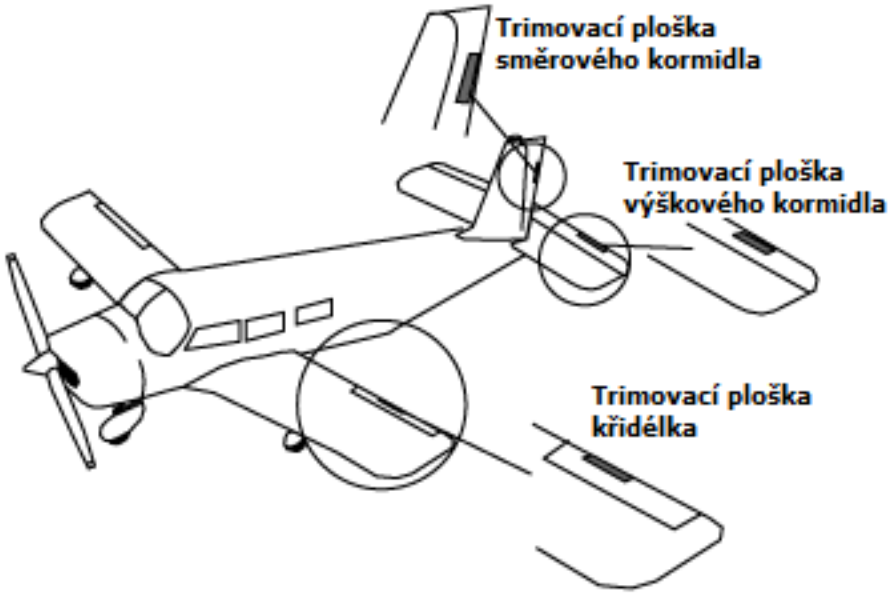
Trimováním se rozumí kalibrační vyvažování letu v osách letadla, tedy osy podélné, příčné a svislé (vertikální). Letadlo je pro rotaci kolem těchto os ovládáno pomocí příslušných řídících, případně trimovacích ploch. Vychylování těchto ploch ovlivňuje proudění vzduchu při letu, což způsobuje úpravy pohybu letadla.

Ovládaná plocha pro řízení rotace kolem vertikální osy se nazývá směrové kormidlo, nachází se na konci ocasní plochy. K řízení rotace kolem podélné osy slouží plochy křidélek, které jsou umístěné na konci křídel. Výškové kormidlo je nezbytné pro rotaci kolem příčné osy. Společný bod všech zmíněných os je těžiště letadla. Vyjmenované plochy jsou náležitostmi primárního řízení letadla.



Obr. 1 Osy letadla (Zdroj: Převzato z [1], upraveno)

Samotná funkce trimování je realizována jemným vychylováním menších ploch, které jsou k tomu určeny. Tyto menší trimovací plošky jsou obvykle umístěny přímo na hlavních řídících plochách, jak je graficky znázorněno níže. Jedná se o sekundární prvky řízení letadla.

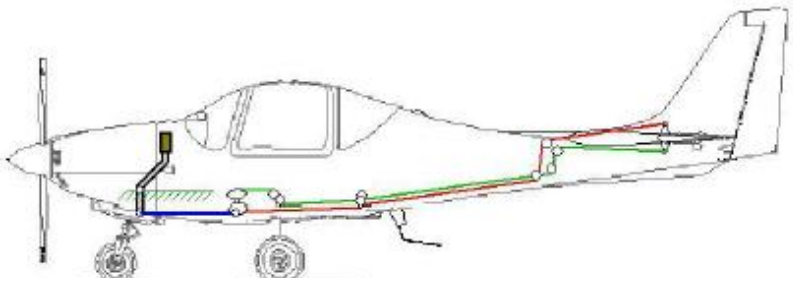


Obr. 2 Trimovací plošky (Zdroj [O2], upraveno)

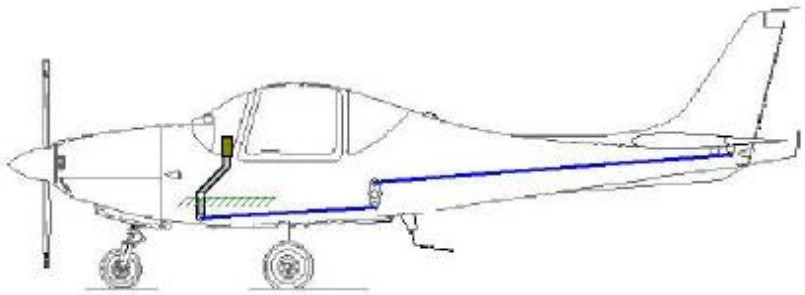
U menších, potažmo ultralehkých letadel, není často instalován trimovácí systém křidélek pro rotaci kolem podélné osy nebo směrového kormidla pro rotaci kolem osy vertikální. Taktéž může být instalován pouze trim výškového kormidla pro vyvažování rotace kolem příčné osy. [1][2]

1.2 Ovládání ploch

Hlavní řídicí plochy bývají v případě ultralehkých letadel ovládány mechanicky, neboť aerodynamické síly nedosahují příliš velkých rozměrů a letoun tak lze ovládat pouhou silou pilota. Pro přenos sil mezi ovládacími prvky v kokpitu a řídícími plochami letounu se používají dva hlavní systémy, a to systém napínacích lan a kladek, a systém táhel. Na obrázku 3 je zobrazeno ovládání výškového kormidla systémem lan a kladek, na obrázku 4 systémem táhel. Někdy je použito i kombinování obou. U větších letadel s mechanickým systémem ovládání, kde by měl pilot problém letoun ovládat pouze svými silami, nebo by ho ovládání mohlo příliš unavit, se používá systém mechanických převodů, které zvyšují sílu pilota.



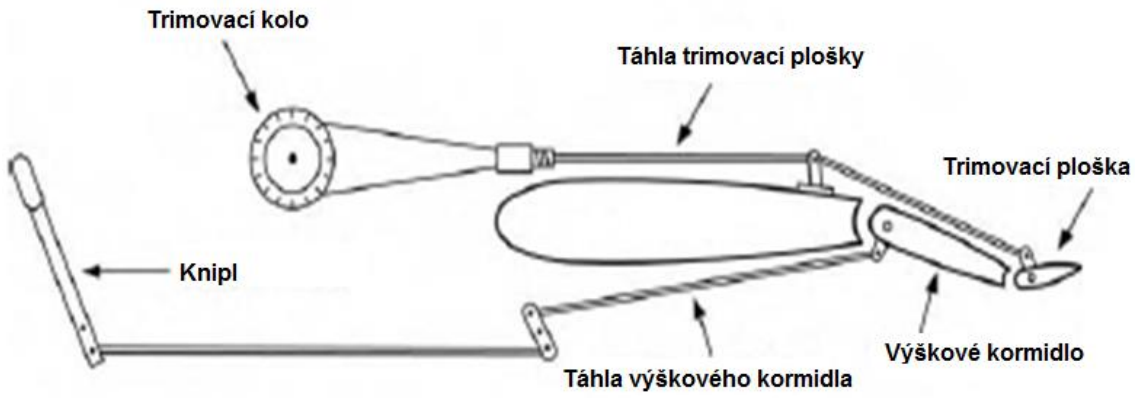
Obr. 3 Mechanický systém řízený lany (Zdroj: Převzato z [2])



Obr. 4 Mechanický systém řízení táhly (Zdroj: Převzato z [2])

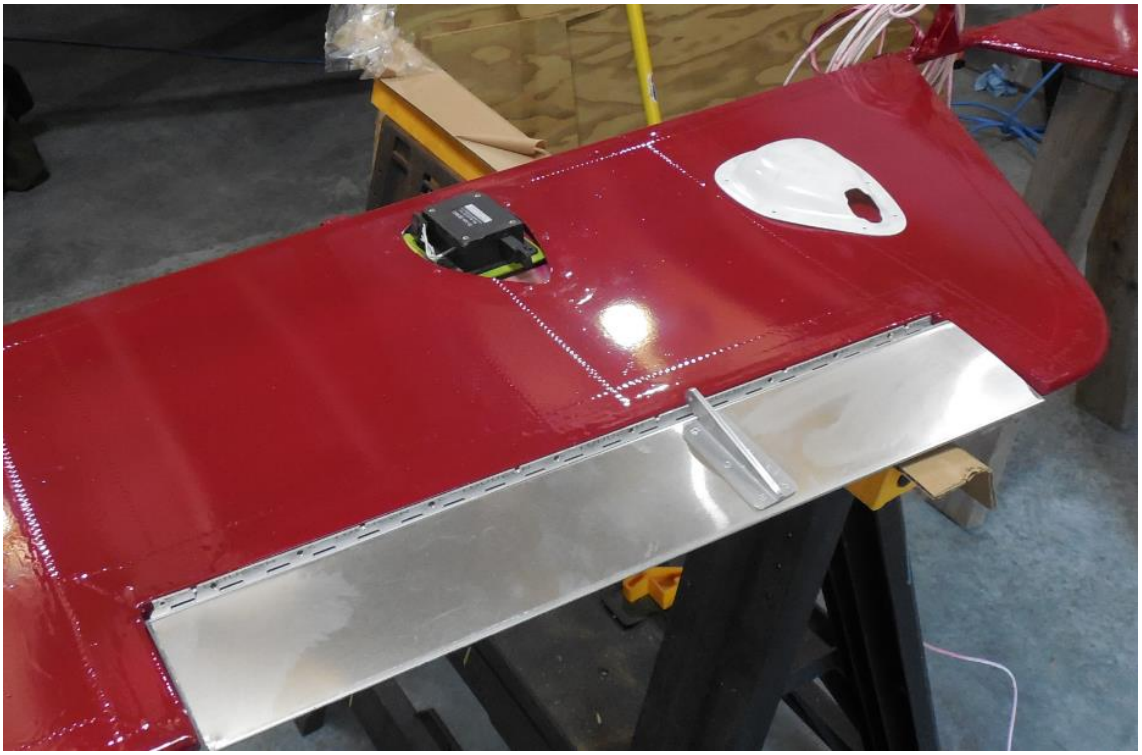
U některých mechanických systémů se také používá pomoc vyvažovacích plošek. Vyvažovací plošky jsou závislé na řídících plochách letounu a pomáhají jim k jejich optimálnímu pohybu, ke snížení působících aerodynamických sil a tím ke snížení mechanických sil potřebných k jejich ovládání. Poprvé byly využity u menších nákladních letadel.

Na obrázku je časté řešení mechanického ovládání výškového kormidla a jeho trimovací plošky, která je polohována trimovacím kolem. Pokud pilot začne s kolem otáčet, táhlo trimovací plošky se dá díky řetězovému převodu do pohybu a ploška tak pomalu začne měnit svoji polohu. Výškové kormidlo je ovládáno rovněž mechanicky pomocí táhel, ale ovládacím prvkem je knipl nebo beran, který se po opuštění vrací do středové neutrální polohy. [2]



Obr. 5 Ovládání výškového kormidla a trimovací plošky (Zdroj [O5], upraveno)

Vyvažovací plošky mohou být na ultralehkých letadlech řízeny i elektricky pomocí serva a tlačítek. V takovém případě může být servo instalováno na vhodném místě na hlavním kormidle, například jako tomu je na obrázku 6. [1][2]



Obr. 6 Instalované servo (Zdroj [O6])

1.3 Princip trimování

Trimování se provádí za účelem eliminace potřeby stálého vychýlení kniplu, beranu nebo pedálů ze středové polohy, a to kdykoli pilot uzná za vhodné.

Při trimování se tedy pilot snaží dosáhnout ideálu letu tak, že usiluje o to, aby některý z parametrů letu byl konstantní a knipl přitom byl ideálně ve středové poloze, kdy jej není potřeba vychylovat. V této situaci tedy spustí trimovací prvek v patřičném smyslu a příslušná trimovací plocha se dá do pomalého pohybu. V okamžiku, kdy letoun dosáhne požadovaného rovnovážného stavu, je pohyb trimovací plošky vypnut. Rovnovážným stavem rozumím například držení konstantní vertikální rychlosti. Tímto postupem pilot eliminuje potřebu držet beran, knipl nebo pedály v jiné než středové poloze, což značně usnadní a zpohodlní pilotáž. Pro vzlet se trimovací plošky nastavují do poloh, které jsou k tomu určené, u větších letadel je tato poloha závislá na rozložení hmotnosti a naložení letadla. [3]

Na obrázku kokpitu Cessny 182 je vyznačeno ovládání trimovacích plošek výškového kormidla a směrového kormidla. Ovládání je řešeno mechanicky pomocí trimovacích kol, indikátory poloh plošek jsou s těmito ovládacími koly spřaženy.

Obr. 7 Ovládání v kokpitu (Zdroj [O7], upraveno)

1.4 Trim výškového kormidla

Tato práce je zaměřena pro návrh ovládání trimu výškového kormidla. Ovládací prvek tedy řídí rotaci kolem příčné osy letadla, jinými slovy lze hovořit o kalibraci vertikální rychlosti letadla nebo úhlu náběhu. Na dalších fotografiích jsou různé způsoby ovládání a indikátorů pro toto trimování.

Na obrázku 9 je detail ovládání trimu z Cessny 172, po levé straně ovládacího kola je indikátor polohy, mechanicky spřažený, momentálně v poloze pro vzlet.

Pohyb kola za letu směrem nahoru zapříčiní, že se úhel náběhu letounu bude mít tendenci pomalu snižovat a naopak pohyb kola směrem dolů způsobí, že úhel náběhu letadla bude mít tendenci pomalu růst. Pohyby a polohu ovládacího serva zachycuje snímač, jehož aktuální pozice znázorňuje polohu plošky na indikátoru pro orientaci pilota.

Na obrázku 8 je tentýž systém ovládaný tlačítky. Na obrázku 10 jsou tlačítka ovládání přímo na beranu, konkrétně Boeingu 737. Filosofie ovládání a funkce na dopravních letadlech je, při manuálním letu, analogická s ultralehkými letadly.

Obr. 9 Trimovací kolo (Zdroj [O9])

Obr. 8 Trimování pomocí tlačítek (Zdroj [O8])

Obr. 10 Ovládání trimu na beranech B737 (Zdroj [O10])

Obr. 11 Indikátor trimu B737 (Zdroj [O11])

1.5 Popis ovládání

Každé letadlo disponuje rozdílným způsobem ovládání, ploška může být ovládána mechanicky kolem, tlačítky na panelu nebo tlačítky přímo na beranu či kniplu.

Pokud se nehýbe ovládací kolo nebo není zmáčknuté žádné tlačítko, trimovací ploška setrvává ve své poloze. Do pomalého pohybu na příslušnou stranu se dá po dobu, po kterou pilot drží tlačítko, otáčí kolem nebo dokud se nedostane do koncové polohy.

Indikátor polohy slouží pouze pro orientaci ohledně krajních poloh plošky a nastavení polohy pro vzlet. Za letu prakticky neexistuje možnost, jak nastavit optimální polohu trimovací plošky podle indikátoru.

2 Prostředky elektronického ovládání plošky

2.1 Základní přehled

V celé praktické části práce řeším ovládání trimovací plošky výškového kormidla. Ovládání bude realizováno pomocí dvou tlačítek na každém z kniplů obou pilotů. Sedadla pilotů jsou řazena za sebou, což je důvod, proč musí mít každý pilot vlastní ovládání trimu. Na obou pozicích pilotů bude rovněž displej s indikací polohy. Poloha plošky a data pro diagnostiku mají být odesílány do CAN sběrnice letadla.

2.2 Praktický rozbor

Dle požadovaných funkcí v zadání je zřejmé, že se jedná o mechatronickou úlohu. Je nezbytné využít jak části mechanické a elektrické, tak i programovatelné řídicí jednotky. Hlavními vstupy do systému jsou dvě tlačítka na každém kniplu pilotů a analogový signál z potenciometru, který bude součástí serva. Servo, pro pohyb plošky, bude vybaveno dvěma koncovými spínači, po jednom v každé krajní poloze. Toto zařízení, umístěné u ovládané plošky, bude poháněno přes převodovku stejnosměrným motorem. Nejoptimálnějším způsobem pro řízení tohoto typu motoru je v tomto případě za použití H můstku. Poloha zaznamenána potenciometrem bude zobrazena na displeji u obou pilotů. Pro odesílání polohy do sběrnice a diagnostiku bude použit CAN bus modul.

Následují rešeršní texty o stejnosměrném motoru, CANu, mikrokontrolerových deskách Arduino, mikroprocesorech a způsobu jejich programování.

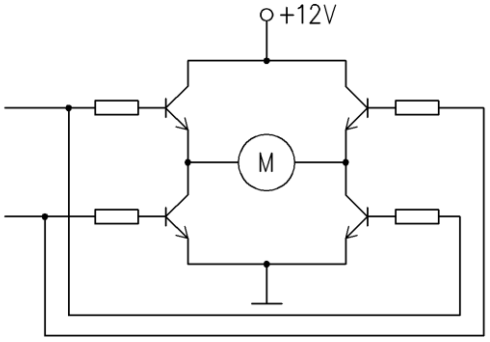
2.2.1 Stejnosměrný motor

Stejnosměrné motory jsou nejstaršími elektrickými stroji. Jejich hlavní výhodou je snadné řízení, které je realizováno změnou budícího napětí motoru. Opačného otáčení motoru se docílí pouhým otočením smyslu průtoku proudu. Toto byl jeden z hlavních důvodů zvolení stejnosměrného motoru pro řízení trimovací plošky, potřebuji pohyb v obou smyslech otáčení, který bude snadno kontrolovaný pomocí dvou tlačítek. Nevýhodou stejnosměrných strojů je jejich konstrukční náročnost.

Hlavními částmi motoru je rotor a stator. Na statoru jsou k vytvoření magnetického toku umístěny hlavní póly, které mohou být buzeny cívkami, nebo pernamentními magnety. Póly s budícím vinutím se skládají z pólového jádra a pólového nástavce. Dále zde mohou být pomocné póly komutační, umístěné mezi hlavními póly pro zlepšení komutačních vlastností stroje. [4]

2.2.2 H můstek

H můstek je elektronické zařízení, které bude využito pro řízení motoru. Skládá ze čtyř tranzistorů, které jsou zapojeny do tvaru písmene H, jak ukazuje schéma. Podle toho, které tranzistory se otevřou, je motor spuštěn v příslušném smyslu otáčení. Pomocí H můstku je možné motor roztočit v obou smyslech otáčení, což je potřebná funkce pro řízení serva, potažmo plošky.



Obr. 12 H můstek (Zdroj [O12])

Tranzistory tedy plní funkci spínacích prvků. Vhodnou kombinací řídicích signálů přiváděných na spínací prvky lze dosáhnout toho, že na svorky zátěže bude přivedeno napájecí napětí jedné nebo druhé polarity. Tím je tedy možné řídit nejen rychlost, ale i směr otáčení. Tato možnost řízení směru platí pro stejnosměrné motory s permanentními magnety. Při použití H-můstku musíme zařídit, aby nebyly v jeden okamžik sepnuty oba spínací prvky v jednom rameni můstku. Došlo by ke zkratu a procházející proud by mohl zničit spínací tranzistory. Celistvé H-můstky mívají zabudovanou vnitřní logiku, která zabraňuje, aby tato situace nastala. [6] [7]

2.2.3 CAN

2.2.3.1 Úvod

Controller Area Network (CAN) je sériový komunikační protokol, který bude využit pro odesílání polohy plošky a dat potřebných pro diagnostiku do sběrnice letadla. Data polohy ze sběrnice budou využita i pro zobrazení polohy na displeji kopilota. Systém byl původně vyvinut firmou Bosch pro nasazení v automobilech. Vzhledem k tomu, že přední výrobci integrovaných obvodů implementovali podporu protokolu CAN do svých produktů, dochází ke stále častějšímu využívání tohoto protokolu i v různých průmyslových aplikacích a letectví. Důvodem je především nízká cena, snadné nasazení, spolehlivost, vysoká přenosová rychlost, snadná rozšiřitelnost a dostupnost potřebné součástkové základny.

V současné době má protokol CAN své pevné místo mezi ostatními sběrnicemi a je definován normou ISO 11898. Ta popisuje fyzickou vrstvu protokolu a specifikaci

CAN 2.0A. Později byla ještě vytvořena specifikace CAN 2.0B, která zavádí dva pojmy - standardní a rozšířený formát zprávy (lišící se v délce identifikátoru zprávy). Tyto dokumenty definují pouze fyzickou a linkovou vrstvu protokolu podle referenčního modelu ISO/OSI. [9]

2.2.3.2 Vývoj a využití

Komplexnost využívaných funkcí implementovaných v těchto nejrůznějších systémech si vynutila potřebu vzájemné komunikace mezi těmito systémy. V konvenčních systémech je pro každý přenášený signál vyhrazena jedinečná přenosová linka, což se ale pro velký počet přenášených signálů stává z finančního hlediska neúnosné. Navíc to přináší mnohé komplikace vyplývající z takto vysokého počtu vodičů určených pro přenos dat.

Veškeré jednotky, které mají potřebu komunikovat ať už mezi sebou, či s jednotlivými senzory zajišťujícími sběr informací jsou propojeny navzájem právě pomocí sběrnice CAN. Účelem použití této sběrnice v automobilovém průmyslu je zajištění komunikace mezi jednotlivými jednotkami tak, aby nedocházelo k velkému zatížení centrálního procesoru. [9]

2.2.3.3 CAN v letectví

V roce 1998 vyvinula společnost Stock Flight Systems protokol založený na CANu, jež označuje jako CANaerospace. Tento protokol podporuje systémy, které využívají koncepci vyjímatelných položek. Jednotlivé připojené komponenty tak bývají snadno vyměňovány, ať už z důvodu údržby nebo testování nových komponentů. Interoperabilita je zajištěna sdílením dat napříč připojenými zařízeními CANu prostřednictvím fyzické vrstvy, síťových vrstev a komunikačních mechanismů. Jedná se o otevřený software, který se stále vyvíjí a standardizuje rozhraní položek na systémové úrovni.

CANaerospace slouží v řadě leteckých, avionických systémech, v leteckém výzkumu a leteckých simulátorech, kde propojuje vybavení kokpitů. [23]

2.2.3.4 Základní vlastnosti protokolu

CAN umožňuje distribuované řízení systémů v reálném čase s vysokou mírou zabezpečení proti chybám. Jedná se o protokol typu takzvaného multi-master, kde každý uzel sběrnice může být master a řídit tak chování jiných uzlů. Není tedy nutné řídit celou síť z jednoho nadřazeného uzlu, což přináší zjednodušení řízení a zvyšuje spolehlivost, při poruše jednoho uzlu může zbytek sítě pracovat dál. Pro řízení přístupu k médiu je použita sběrnice s náhodným přístupem, která řeší kolize na základě prioritního rozhodování. Po sběrnici probíhá komunikace mezi dvěma uzly pomocí zpráv (datová zpráva a žádost o data), a management sítě (signalizace chyb, pozastavení komunikace) je zajištěn pomocí dvou speciálních zpráv (chybové zprávy a zprávy o přetížení).

Zprávy vysílané po sběrnici protokolem CAN neobsahují žádnou informaci o cílovém uzlu, kterému jsou určeny, a jsou přijímány všemi ostatními uzly připojenými ke sběrnici. Každá zpráva obsahuje identifikátorem, který udává význam přenášené zprávy a její prioritu. Protokol CAN zajišťuje, aby zpráva s vyšší prioritou byla v případě kolize dvou zpráv doručena přednostně. Dále je možné na základě identifikátoru zajistit, aby uzel přijímal pouze ty zprávy, které se ho týkají.

Pro zajištění transparentnosti návrhu a flexibility implementace je sběrnice CAN rozdělena do tří rozdílných vrstev:

- CAN vrstvy objektů,
- CAN transportní vrstvy,
- fyzické vrstvy.

Vrstva objektů a transportní vrstva zahrnuje veškeré služby a funkce poskytované v rámci linkové vrstvy, tak jak je definována modelem ISO/OSI. Vrstva objektů je odpovědná za:

- nalezení zprávy, která má být vyslána,
- rozhodnutí, které přijaté zprávy od transportní vrstvy mají být použity,
- poskytování rozhraní aplikační vrstvě související s hardwarem.

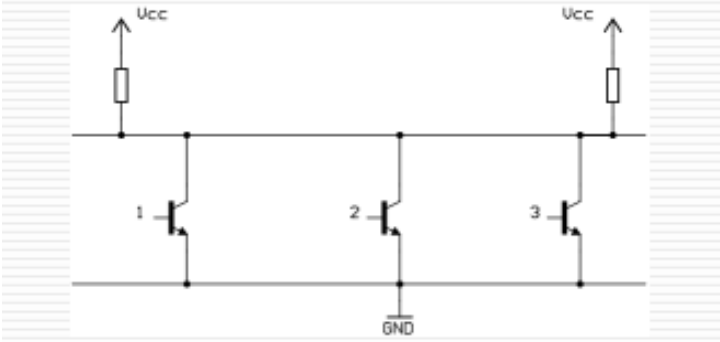
Úkolem transportní vrstvy je především přenosový protokol. Například řízení rámců, řízení, kontrola chyb, signalizace chyb. Uvnitř transportní vrstvy je rozhodnuto, zda je sběrnice volná pro nový přenos dat či naopak jejich příjem. Také několik obecných vlastností týkajících se časování bitů je svěřeno transportní vrstvě. Je možné prohlásit, že vzhledem k povaze transportní vrstvy zde není žádný prostor pro její modifikaci ze strany uživatele.

Úkolem fyzické vrstvy je vlastní přenos jednotlivých bitů mezi jednotlivými uzly s respektováním všech elektrických vlastností. Uvnitř jedné sítě má fyzická vrstva stejné parametry pro všechny uzly, nicméně je možné zvolit si její parametry tak, aby co nejlépe vyhovovaly dané aplikaci. [9]

2.2.3.5 Fyzické médium a fyzická vrstva

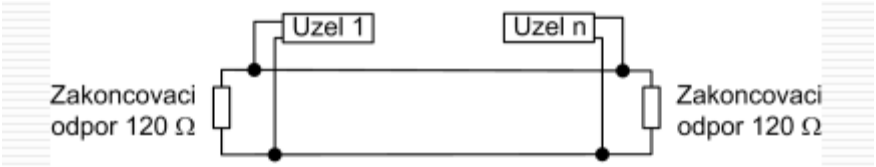
Protokol CAN definuje vlastní rozhraní k fyzickému přenosovému médiu a v tomto směru se odlišuje od modelu ISO/OSI. Na druhé straně jsou vlastnosti fyzické vrstvy velkou předností protokolu CAN. Základním požadavkem na fyzické přenosové médium protokolu CAN je, aby realizovalo funkci logického součinu. Za účelem zvýšení rychlosti a odolnosti proti rušení je účelné, aby spoj byl symetrický. Standard protokolu CAN definuje dvě vzájemně komplementární hodnoty bitů na sběrnici - dominant a recessive. Jedná se v podstatě o jakýsi zobecnělý ekvivalent logických úrovní, jejichž hodnoty nejsou určeny a skutečná reprezentace záleží na konkrétní realizaci fyzické vrstvy.

Pravidla pro stav na sběrnici jsou jednoduchá a jednoznačná. Vysílají-li všechny uzly sběrnice recessive bit, pak na sběrnici je úroveň recessive. Vysílá-li alespoň jeden uzel dominant bit, je na sběrnici úroveň dominant. Příkladem může být optické vlákno, kde stavu dominant bude odpovídat stav svítí a recessive stav nesvítí. Dalším příkladem může být sběrnice buzená hradly s otevřeným kolektorem, kde stavu dominant bude odpovídat logická nula na sběrnici a stavu recessive logická jednička. Pak, je-li jeden tranzistor sepnut, je na sběrnici úroveň logické nuly (dominant) a nezáleží již na tom, zda je či není sepnutý i nějaký jiný tranzistor. Pokud není sepnut žádný tranzistor, je na sběrnici úroveň logické jedničky (recessive).



Obr. 13 Příklad realizace fyzické vrstvy protokolu CAN (Zdroj: Převzato z [9])

Pro realizaci fyzického přenosového média se nejčastěji používá diferenciální sběrnice definovaná podle normy ISO 11898. Tato norma definuje jednak elektrické vlastnosti vysílacího budiče a přijímače tak zároveň principy časování, synchronizaci a kódování jednotlivých bitů. Sběrnici tvoří dva vodiče (označované CAN_H a CAN_L), kde dominant či recessive úroveň na sběrnici je definována rozdílovým napětím těchto dvou vodičů. Dle nominálních úrovní uvedených v normě je pro úroveň recessive velikost rozdílového napětí V_{diff} = 0 V a pro úroveň dominant V_{diff} = 2 V. Pro eliminaci odrazů na vedení je sběrnice na obou koncích přizpůsobena zakončovacími odpory o velikosti 120 Ω. Jednotlivá zařízení jsou na sběrnici připojena pomocí konektorů.



Obr. 14 Fyzické uspořádání sítě CAN podle ISO 11898 (Zdroj: Převzato z [9])

Ke sběrnici může být teoreticky připojen libovolný počet uzlů, ale prakticky s ohledem na zatížení sběrnice, je počet připojených uzlů podstatně nižší a uvádí se kolem 64 na segment. Rovněž přenosová rychlost 1 Mbit/s je dosažitelná pouze na krátké vzdálenosti do 40m a se vzdáleností prudce klesá, takže na 1,2km činí asi 70 kbitů/s. Plyne to z původního poslání sběrnice CAN, která byla určena pro malé vzdálenosti v instalaci automobilů. [9]

2.2.3.6 Linková vrstva

Tak jako v modelu ISO/OSI i v protokolu CAN je linková vrstva rozdělena na podvrstvu LLC a MAC:

- MAC (Medium Access Control) reprezentuje jádro protokolu CAN. Úkolem je provádět kódování dat, vkládat doplňkové bity do komunikace (Stuffing/Destuffing), řídit přístup všech uzlů k médiu s rozlišením priorit zpráv, detekce chyb a jejich hlášení a potvrzování správně přijatých zpráv.
- LLC (Logical Link Control) je podvrstva řízení datového spoje, což zde znamená filtrování přijatých zpráv a hlášení o přetíženích. [9]

2.2.3.7 Řešení přístupu média a kolize

Vzhledem k tomu, že se jedná o síť typu multimaster, každý z účastníků může zahájit vysílání, jakmile je připraven a síť je v klidovém stavu. Vždy vysílá pouze první vysílající. Ostatní mohou vysílat až poté, co je zpráva odvysílána. Výjimku tvoří

chybové rámce, které se dají vysílat okamžitě po identifikaci chyby kterýmkoli účastníkem.

Zahájí-li vysílání současně několik uzlů, pak přístup na sběrnici získá ten, který přenáší zprávu s vyšší prioritou (nižším identifikátorem). Identifikátor je uveden na začátku zprávy. Každý vysílač porovnává hodnotu právě vysílaného bitu s hodnotou na sběrnici a zjistí-li, že na sběrnici je jiná hodnota než vysílá, okamžitě přeruší další vysílání. Tím je zajištěno, že zpráva s vyšší prioritou bude odeslána přednostně a že nedojde k jejímu poškození, což by mělo za následek opakování zprávy a zbytečné prodloužení doby potřebné k přenosu zprávy. Uzel, který nezískal při kolizi přístup na sběrnici musí vyčkat, až bude sběrnice opět v klidovém stavu, a pak zprávu vyslat znovu. [9]

2.2.3.8 Druhy zpráv

Specifikace protokolu CAN definuje čtyři typy zpráv:

- datová zpráva
- žádost o data
- zpráva o chybě
- zpráva o přetížení

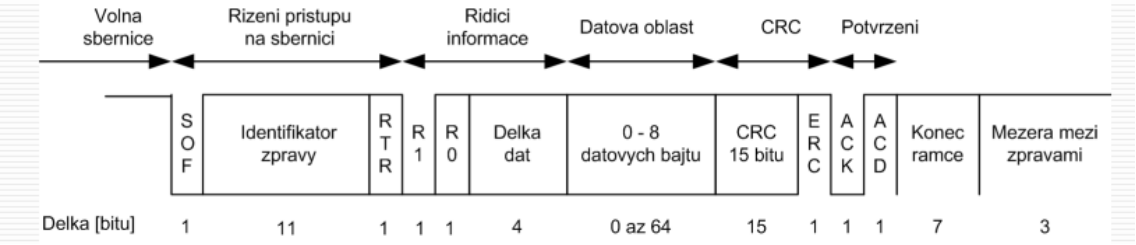
Datová zpráva a žádost o data se týkají přenosu dat. Datová zpráva tvoří základ komunikace, umožňuje zařízení vyslat zprávu dlouhou až 8 Bytů. Naopak při jednoduchých typech datových zpráv, jako jsou povely zapnout nebo vypnout a podobně není třeba posílat žádná data, tyto binární příkazy mohou být obsaženy v identifikátoru zpráv. Tím se zvyšuje rychlost přenosu v protokolu CAN. Zařízení, které tato data vlastní je vyšle na sběrnici.

Další dva typy zpráv slouží k řízení sběrnice a to k signalizaci chyby a eliminaci chybných zpráv a k signalizaci o přetížení, tedy vyžádání prodlevy v komunikaci. [9]

2.2.3.9 Datové a standardní zprávy

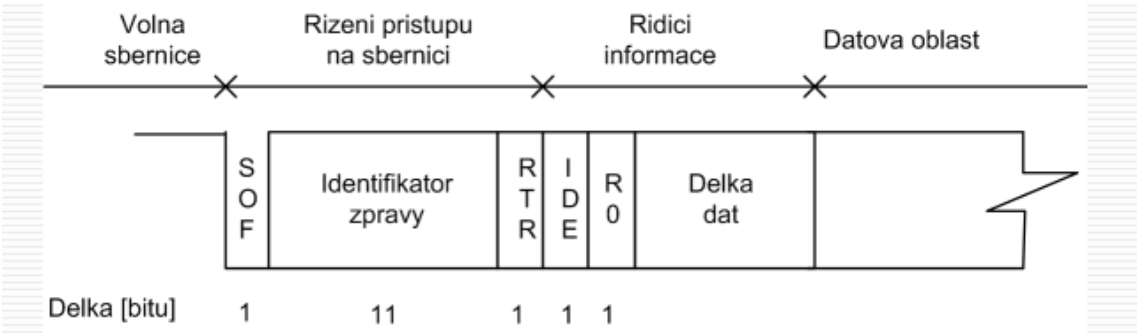
Protokol CAN používá dva typy datových zpráv. První typ je definován specifikací 2.0A a je v literatuře označován jako standardní formát zprávy, zatímco specifikace 2.0B definuje navíc takzvaný rozšířený formát zprávy. Jediný podstatný rozdíl mezi oběma formáty je v délce identifikátoru zprávy, která je 11 bitů pro standardní formát a 29 bitů pro rozšířený formát. Oba dva typy zpráv mohou být používány na jedné sběrnici, pokud je použitým řadičem podporován protokol 2.0B.

Vyslání datové zprávy je možné pouze tehdy, je-li sběrnice volná. Jakmile uzel, který má připravenu zprávu k vyslání, detekuje volnou sběrnici, začíná vysílat. Zda získá přístup na sběrnici či nikoliv, záleží na již popsaném mechanismu řízení přístupu k médiu. Strukturu datové zprávy podle specifikace 2.0A ilustruje obrázek.



Obr. 15 Datová zpráva podle specifikace CAN 2.0A (Zdroj: Převzato z [9])

Standardní zpráva je převzata ze specifikace 2.0A, má délku identifikátoru zprávy 11 bitů. Jediným rozdílem je zde využití prvního bitu na indikaci, zda se jedná o rámec standardní nebo rozšířený. Z obrázku, který zobrazuje začátek rámce je vidět, že řízení přístupu na sběrnici (priorita zprávy) je dána opět 11ti bity identifikátoru a hodnotou bitu označeného jako RTR (Remote Request). [9]



Obr. 16 Začátek datové zprávy (standardní formát) podle specifikace 2.0B (Zdroj: Převzato z [9])

2.2.4 ARINC

Společnost ARINC se zabývá poskytováním komunikačních systémů sběru a sdílení dat přímo pro aplikace v letectví. Příklady standardů této společnosti uvádím jako alternativu ke CANu. Ačkoli byl CAN vytvořen původně pro automobilový průmysl tak se používá i v řadě letadel, případně může být instalován společně s ARINCem.

ARINC 429: Nejpoužívanější komunikační standard, který vymezuje avioniku letadla. Elektrický a datový formát je definován dvouvodičovou sériovou sběrnicí s jedním vysílačem a až 20 přijímači. Sběrnice je schopna pracovat rychlostí 100 kbit/s, naproti tomu CAN s 1Mbit/s. Délka dat zprávy je 19 bitů.

ARINC 629: Počet přijímačů je až 120, rychlost je zvýšena na 2Mb/s. Tento standard je instalován například v Boeingu 777.

ARINC 664: Využívá komunikaci přes ethernet, byl vyvinut pro fly-by-wire konfiguraci Airbusu A380. [24]

2.3 Ovládání prostřednictvím mikroprocesorů

Pro řízení akčních členů této úlohy je vhodné použít programovatelné mikroprocesory. Při správném zapojení a naprogramování dokážou řídit stejnosměrný motor přes H můstek, zobrazovat požadované informace na displejích a spolupracovat s CAN.

Pro testovací verze jsou všechny funkce demonstrovány na deskách Arduino Uno a Arduino Due. Pro finální verzi, potenciálně praxi, bude vyvinuta deska přímo pro tento trimovací systém s procesorem ATmega2560.

2.3.1 Arduino obecně

Arduino je otevřená platforma s grafickým vývojovým prostředím, které využívá rozsáhlých knihoven Wiring, který se díky jejím komplexnostem často považuje za samostatný jazyk. Základem těchto knihoven je však jazyk C a Processing, prostředí

pro výuku programování. Může být použito k vytváření samostatných interaktivních zapojení nebo může být připojeno k softwaru na počítači. Arduino UNO i DUE jsou již zkompletované desky připravené k zapojení, nahrání programu a použití.

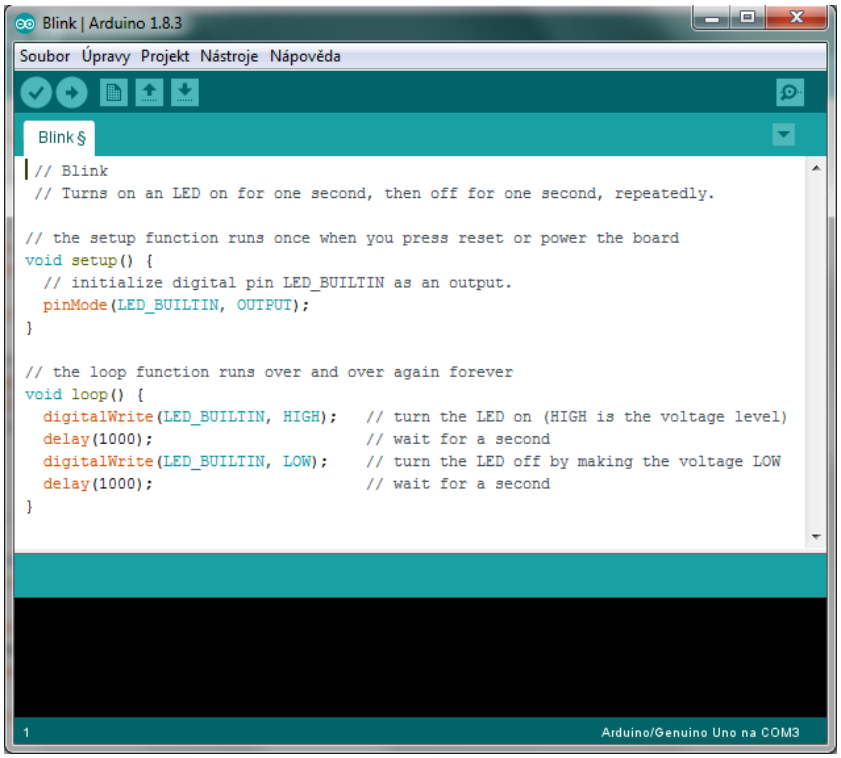
Arduino není plnohodnotný stolní počítač. Řídící program je vyvíjen zvlášť na stolním počítači a do Arduina je posléze nahrán prostřednictvím propojovacího kabelu a spuštěn. Uvnitř Arduina je pak spuštěn jen tento program, který typicky obsahuje smyčku, která se neustále opakuje. Arduino neustále zjišťuje stav svého okolí a na změny reaguje, má nízkou spotřebu a mimo jiné se používá pro řízení dronů, robotů, průmyslových procesů, nejrůznějších dálkově ovládaných vehiklů atp. [5]

2.3.2 Způsob programování

Na obrázku 17 je vývojové prostředí Arduina, ve kterém jsem vytvořil veškeré programy pro řízení. Využíval jsem komplexní knihovny Wiring a základní znalosti programovacího jazyka C. Jeho výhodou je jednoduchost praktického použití, které jsem aplikovat i bez hlubších, specifických znalostí z oblasti hardwaru a mikroelektroniky.

Pro program (též sketch) jsou charakteristické vždy dvě části. První se označuje „setup“, která proběhne pouze jednou. Zaznamenáváme do ní prvotní nastavení, počáteční stavy, vstupy a výstupy. Druhá nese označení „loop“ a běží neustále dokola, zadávají se do ní podmínky řízení a působení na akční členy. Na obrázku je pro příklad program, který vykoná následující kroky: [5]

- V části „setup“ Nastaví vestavěnou LED elektronické desky jako výstup, akční člen
- Poté tuto diodu po jedné sekundě nebo tisíci milisekundách zapíná a vypíná a to stále dokola neboť jsou tyto příkazy umístěné v části „loop“



Obr. 17 Vývojové prostředí Arduina [S1]

2.3.3 Jazyk C

Jazyk C je jedním z nejvýznamnějších imperativních programovacích jazyků, který zásadně ovlivnil vývoj celého odvětví IT. Jazyk C je jednoduchý, expresivní a univerzální, vyznačuje se úsporností a určitým minimalizmem. Není specializovaný pro žádnou určitou oblast a lze jej využít jak k programování mikroprocesorů, tak i k implementaci informačních systémů. Byl poprvé představen v roce 1972 a jeho autory jsou Brian W. Kernighan a Dennis M. Ritchie. Jazyk C se symbioticky vyvíjel společně s operačním systémem Unixe a tato skutečnost je patrná dodnes, jádro i většina programů je napsána právě v C. V průběhu času byl několikrát vylepšen a vzniklo i několik standardů. Jejich cílem bylo vytvořit bezespornou a strojově nezávislou definici jazyka.

Ideově vychází z jazyků BCPL (Martin Richards) a B (Ken Thompson), ke kterým přidává typy a strojově nezávislou adresovou aritmetiku, realizovanou ukazateli. Každý program v jazyce C se skládá z proměnných a funkcí (lépe řečeno podprogramů). Vstupním bodem programů je speciální funkce main. [10]

2.3.4 Arduino UNO

Arduino Uno je mikrokontrolérová vývojová deska založená na ATmega328. Deska obsahuje 14 digitálních vstupních / výstupních pinů (z toho může být 6 použito jako výstupy PWM), 6 analogových vstupů, 16 MHz krystal, připojení pomocí USB, napájecí konektor, ICSP rozhraní a resetovací tlačítko. Obsahuje vše potřebné k provozu mikrokontroléru, připojení k počítači pomocí USB kabelu. Pracovní napětí je 5V.

Tuto desku jsem použil pro zprovoznění ovládání motoru serva přes H můstek pomocí tlačítek a spínačů koncových poloh. Rovněž jsem s její pomocí posílal data pro sběrnici CAN. UNO bylo využito také pro vytvoření první testovací verze (kapitola 3.1) za účelem vyzkoušení funkčnosti komponent a vytvoření rešerše pro regulérní řešení. Úložné místo pro program je 32256 bytů. [11]

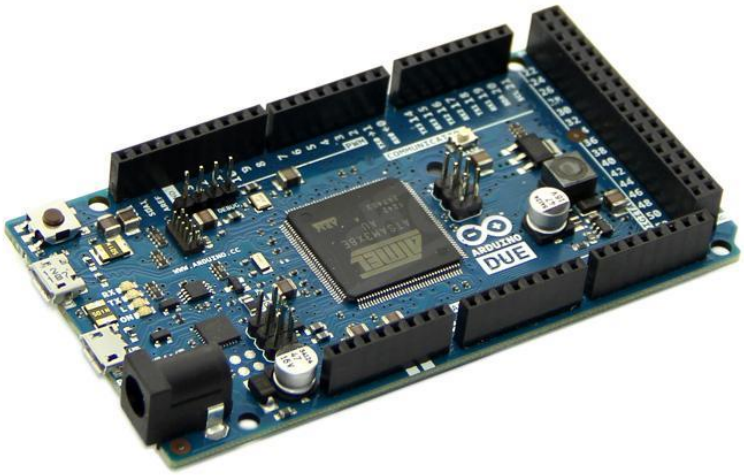
Obr. 18 Arduino UNO (Zdroj [O18])

2.3.5 Arduino DUE

Arduino Due je mikrokontrolérová deska založená na procesoru Atmel SAM3X8E ARM Cortex-M3. Jedná se o první Arduino desku založenou na 32-bit jádru. Má 54 digitálních vstupních / výstupních pinů (z toho 12 z nich lze použít jako výstupy PWM), 12 analogových vstupů, 4 UART (hardware sériové porty), 84 MHz, USB, 2 D/A (digitálně analogový převodník), napájecí konektor, SPI, tlačítko reset a tlačítko mazání.

Na rozdíl od většiny Arduino desek Arduino Due deska běží na 3.3V. Maximální napětí, které I / O piny tolerují je 3.3V. Napětí vyšší než 3,3V na jakémkoli I / O pinu může poškodit vývojovou desku.

Na desce jsem zprovoznil displej pro zobrazení polohy jezdce potenciometru. Výhoda desky je silnější procesor a 3V logika, která se hodí přímo pro zvolený typ displeje, nemusel jsem tak využívat převodníků logických úrovní ani napěťových děličů pro zajištění správného napájení displeje. Úložné místo pro program je 524288 bytů. [11]



Obr. 19 Arduino DUE (Zdroj [O19])

3 Praktický návrh

Tato kapitola obsahuje návrhy verzí, které označuji jako:

- Rešeršní verze (kapitola 3.1)
- Zkušební verze (kapitola 3.2)

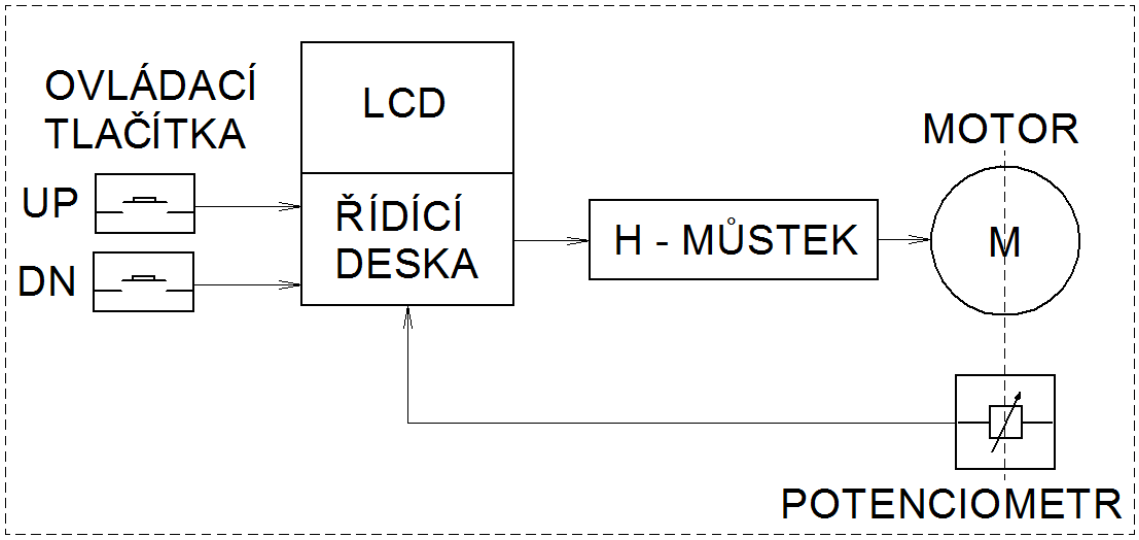
Finální verze této práce je obsahem následujících kapitol 4,5,6 a je silně inspirována verzí zkušební, bude v ní však použit specifický hardware a komponenty.

3.1 Rešeršní verze systému

3.1.1 Základní informace

Jedná se o rešeršní, zjednodušenou verzi, kterou jsem vytvořil s cílem ověření principů fungování komponent a jejich ovládání tak, aby je bylo možné využít ve zkušební a finální verzi. Pro tento účel jsem použil desku Arduino UNO (kapitola 2.3.4), ke které jsem připojil shield s LCD, ovládací tlačítka, potenciometr a stejnosměrný motor řízený přes H můstek. Funkce této verze byly následující:

- Točit s hřídelí motoru v obou smyslech otáčení pomocí stisku a držení příslušného tlačítka
- Hlídat spouštění motoru v krajních polohách potenciometru
- Vypnout motor pokud se jezdec potenciometru dostal do blízkosti některé z krajních poloh
- Na LCD graficky zobrazit polohu jezdce potenciometru
- Díky podmínkám programu chránit před potencionálním zkratem v můstku (v případě stisku obou tlačítek ve stejný okamžik)



Obr. 20 Schéma rešeršní verze [S2]

3.1.2 Řešení indikátoru polohy

Poloha byla graficky zobrazována na LCD prostřednictvím vykreslení jednoho až šestnácti znaků „X“, dle aktuální pozice zaznamenané potenciometrem. Na displeji je možné, ve spodní řádce, vidět označení krajních poloh UP, DN a polohu TO. V poloze UP by byla trimovací ploška v takové poloze, že by úhel náběhu letadla maximálně zvyšovala, naopak v poloze DN snižovala. TO konkretizovaná znakem „|”, je poloha určená pro nastavení na zemi před vzletem.

Obr. 21 LED indikátor polohy rešeršní verze

3.1.3 Seznam použitých komponent

- Arduino UNO
- Arduino LCD shield
- Stejnosměrný motor s napájením na 12V
- H můstek L298N
- Potenciometr TP160
- Dvě tlačítka bez aretace polohy
- Dva odpory 360 Ω
- Zdroj 12V
- Nepájivé kontaktní pole
- Potřebné vodiče

3.1.4 Zapojení

LCD shield je komponenta, kterou není třeba připojovat k desce UNO pomocí, kabelových vodičů, stačí ji nasadit přímo na desku, jednotlivé kontakty tak přijdou přesně do využitých pinů. Tento LCD je schopen zobrazit 16 znaků ve dvou řádcích. Tento typ displeje byl využit výhradně pro tuto rešeršní verzi.

H můstek L298N je schopen řídit jeden krokový motor nebo dva stejnosměrné, proto bude půl funkce tohoto můstku nevyužita a určité piny zůstanou volné.

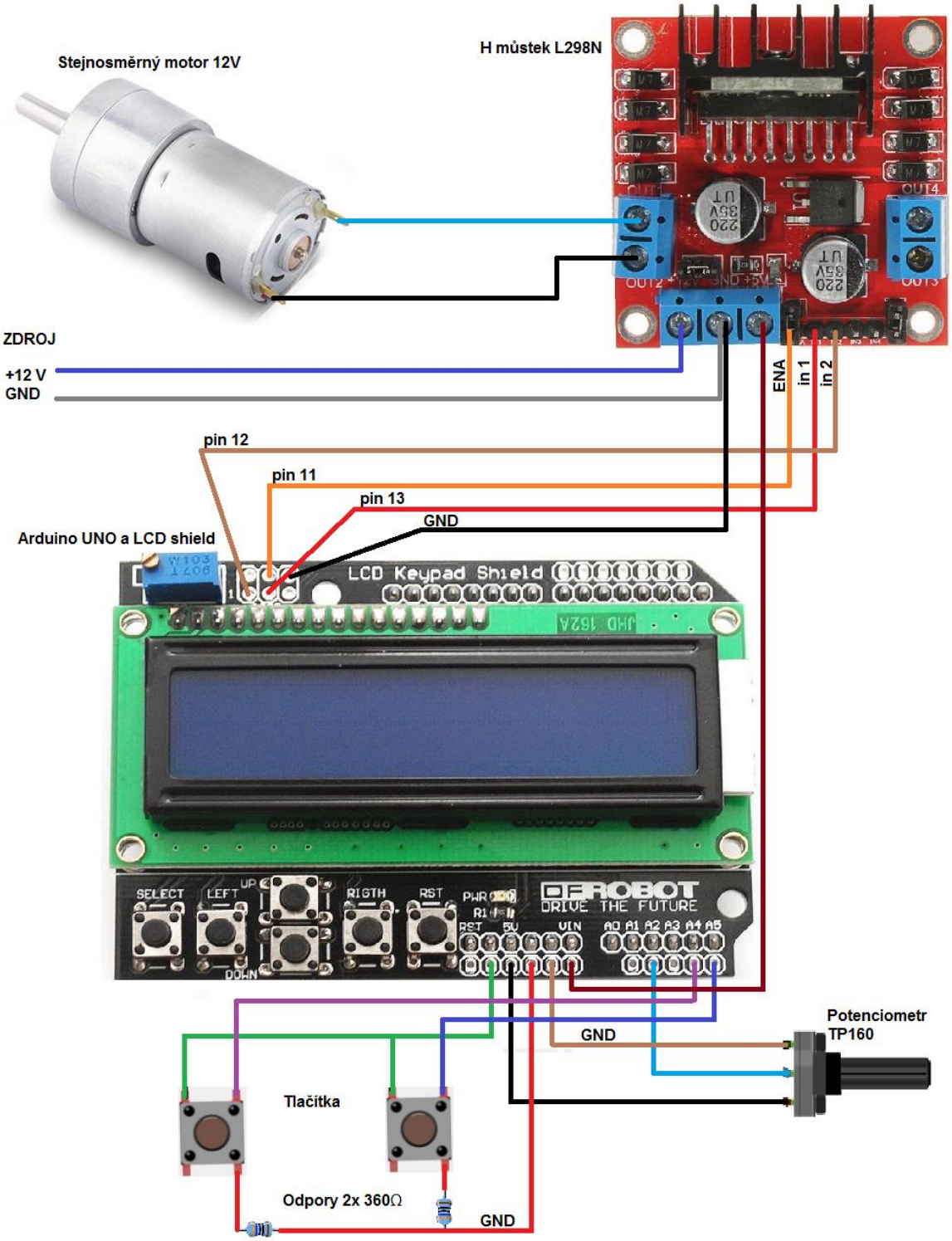
Provizorní potenciometr TP160 s rotační drahou je využit jako snímač polohy, jeho výstupem je analogový signál, který vstupuje do řídící desky. Poloha jezdce se začne měnit s rotací hřídele potenciometru. V dalších verzích i v praxi bude použit potenciometr s lineární drahou umístěný v servu.

Tlačítka jsou zapojena tak, aby poskytovala digitální signály a informovala tak řídicí desku zda je v jistý okamžik některé tlačítko stisknuto či nikoli. Piny výstupů Arduina není radno přetěžovat, v krajním případě, proudem větším než je 40 mA, proto byly využity odpory, neboť zde platí Ohmův zákon.

12V zdroj je k můstku připojen až po nahrání programu, aby mělo celé zařízení dostatečné napětí pro spouštění motoru. Po nahrání programu, odpojení řídicí desky

od počítače a před připojením ke zdroji se připojuje napětí z napájeného můstku do pinu s označením VIN na LCD shield pro napájení jeho a desky.

Arduino UNO při pohledu shora není vidět, protože ho překrývá nasazený LCD shield, který i vyvádí jeho nevyužité piny. [17][19][25][26][8]



Obr. 22 Zapojení rešeršní veze [S3] (Zdroj [O22], upraveno a zkompilováno)

Obr. 23 Zapojení rešeršní verze ve skutečnosti

3.1.5 Program

Kompletní program se nachází v příloze 1, veškeré komentáře a vysvětlení jsou v programu umístěny za značkami „//“, aby nezasahovaly do funkční oblasti programu. Zde se omezím pouze na stručný popis a princip naprogramování.

Na začátku programu jsou definovány vstupy a výstupy. Digitální vstupní signály vcházejí do pinů A4 a A5, jedná se o indikaci stisku některého z tlačítka. Analogový signál potenciometru je připojen k pinu A2. Výstup je na pinech 12 a 13, což jsou signály ovládání můstku k řízení motoru. Motor je spuštěn, když je jeden z pinů 12 a 13 v logické nule a druhý v logické jedničce, tak je motor spuštěn buď vpřed, nebo vzad.

Je zařízeno, aby oba tyto signály nemohli být v logické jedničce, kvůli možnému zkratu, který by tato situace mohla vyvolat. Pro řízení rotace motoru v obou smyslech otáčení jsou použity podmínky „jestliže“, které motor spustí právě tehdy, když je stisknuto tlačítko a potenciometr se nenachází v blízkosti krajní polohy na té straně, kam se má motor začít točit.

Na displej se poloha přepíše výhradně tehdy, jestliže se od posledního čtení programu změnila nebo byl program právě spuštěn. Analogový signál z pinu A2, který nabývá hodnot od 0 do 1023, byl fragmentován na 16 částí. Pomocí cyklu se na displeji zobrazí právě tolik znaků „X“, kolik fragmentovaných částí v danou chvíli analogický signál, svou velikostí, zaplňuje. Znaky „X“ tímto způsobem, graficky, zobrazí velikost analogového signálu v rozlišení, které displej poskytuje.

Tento program na desce UNO zabírá 11% úložného místa pro program (3582 bytů), globální proměnné zabírají 12% dynamické paměti (264 bytů).

3.2 Návrh zkušební verze systému

3.2.1 Základní informace

Po vyzkoušení komponent a principů v rešeršní verzi jsem pokračoval tvorbou verze zkušební, jejímž cílem byl návrh autentického systému trimu za požití desek Arduino. Verze finální se liší využitým hardwarem, kde bude využito desky, která bude navržena přímo pro tento systém.

Trimovací plošku výškového kormidla je možné ovládat ze dvou stanovišť. Na předním stanovišti, u pilota, bude instalována část systému označená jako Master Trim, na stanovišti zadním, u kopilota, Slave Trim.

Master Trim slouží pro samotné řízení stejnosměrného motoru přes H můstek prostřednictvím tlačítek obou pilotů, která jsou zapojena pararelně. Dále odesílá data do sběrnice a zobrazuje polohu potenciometru, potažmo plošky, na displeji. Do podmínek řízení motoru vstupují ještě digitální signály od koncových spínačů serva, aby v koncových polohách došlo k zastavení motoru a nemohlo dojít ke kolizi. Slave Trim přijímá data polohy ze sběrnice a zobrazuje je na displeji kopilota.

Funkce této verze jsem testoval a programoval samostatně na jednotlivých modelech, jejichž popis je níže, vše bude sjednoceno ve finální verzi na elektronické desce, která bude vyvinuta přesně pro tento účel.

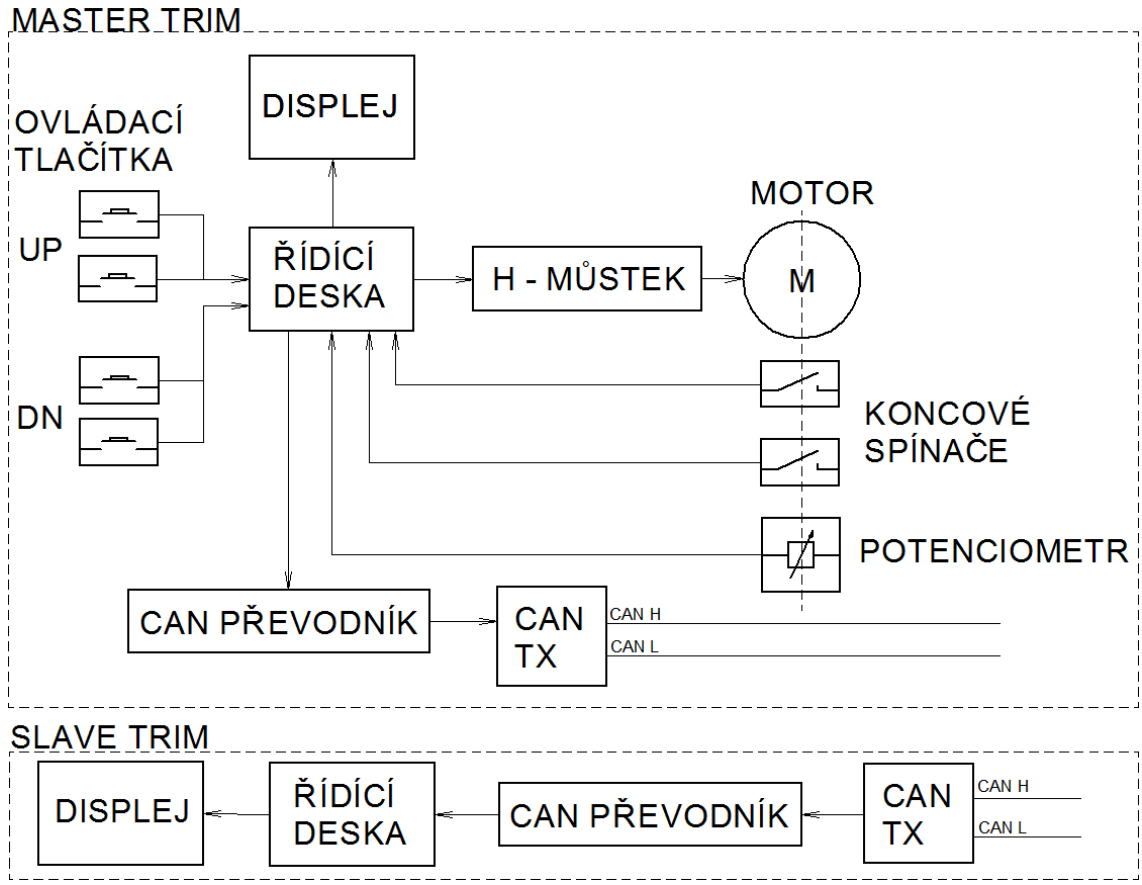
Ovládání motoru je principiálně stejné jako v rešeršní verzi. Tato verze je doplněna o CAN, koncové spínače a displej, který bude použit i ve finální verzi. Také je zde navíc řešen systém pro kopilota. Pro dílčí funkce jsem vytvořil zkušební modely označené jako:

a) model řízení motoru

b) model řízení displeje

c) model CAN části „Master Trim“

d) model CAN části „Slave Trim“

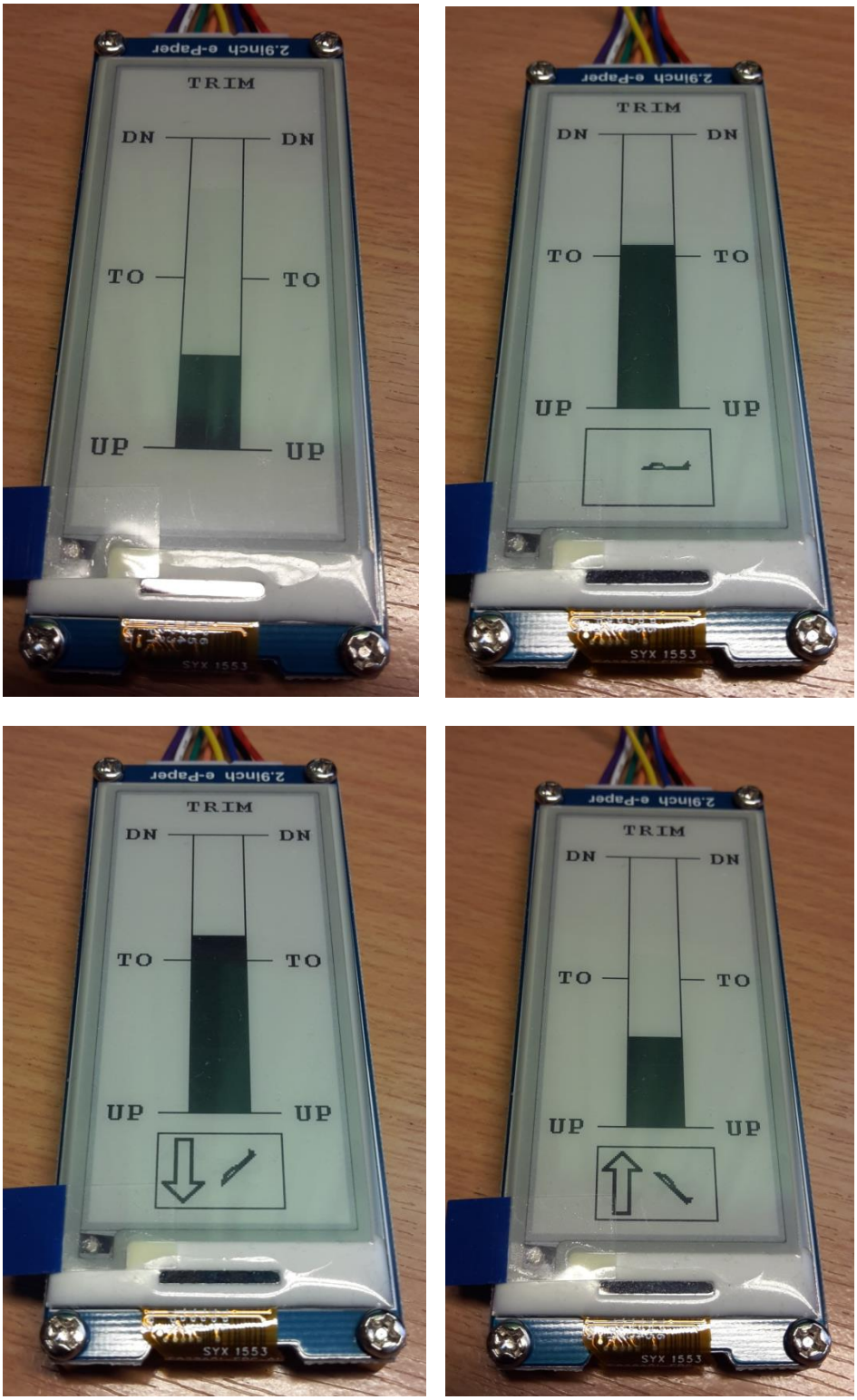


Obr. 24 Schéma zkušební verze [S2]

3.2.2 Řešení Indikátoru polohy

Jako indikátor polohy byl zvolen: Waveshare 2.9 Inch E-Paper Display, jedná se o displej s technologií elektronického inkoustu, který nemá problém s viditelností zobrazovaných dat na slunci a snadno zvládá jednoduché animace, pro využití v kokpitu se tedy, pro takovéto účely, hodí.

Pro displej jsem vytvořil dvě varianty zobrazení. Obě naprogramované varianty ukazují polohu jezdce potenciometru pomocí černého obdélníka, který se posouvá a mění svoji velikost. Druhá verze navíc, po dobu pohybu plošky (stisku tlačítka), vykresluje obrázek aktuálního chování letadla. Tato funkce si klade za cíl vyjasnit aktuální situaci o trimu potenciálně vystresovanému pilotovy. Tyto dočasně zobrazené symboly ukazují, co trim v daném okamžiku s letadlem dělá, zda má jeho úhel náběhu snahu zvyšovat či snižovat.



Obr. 25 Indikace polohy zkušební verze, dvě alternativy

V levém horním rohu je verze programu bez znázornění tendencí chování letadla, na ostatních fotografiích je jeho verze se znázorněním ve všech případech, které mohou nastat.

3.2.3 Seznam zkušebních komponent

Pro model a) řízení motoru

- Arduino UNO
- Stejnosměrný motor s napájením na 12V
- H můstek L298N
- Čtyři tlačítka bez aretace polohy
- Čtyři odpory 360 Ω
- Zdroj 12V
- Nepájivé kontaktní pole
- Potřebné vodiče

Pro model b) řízení displeje

- Arduino DUE
- Potenciometr TP160
- Waveshare 2.9 Inch E-Paper Display
- Nepájivé kontaktní pole
- Potřebné vodiče
- USB propojovací kabel

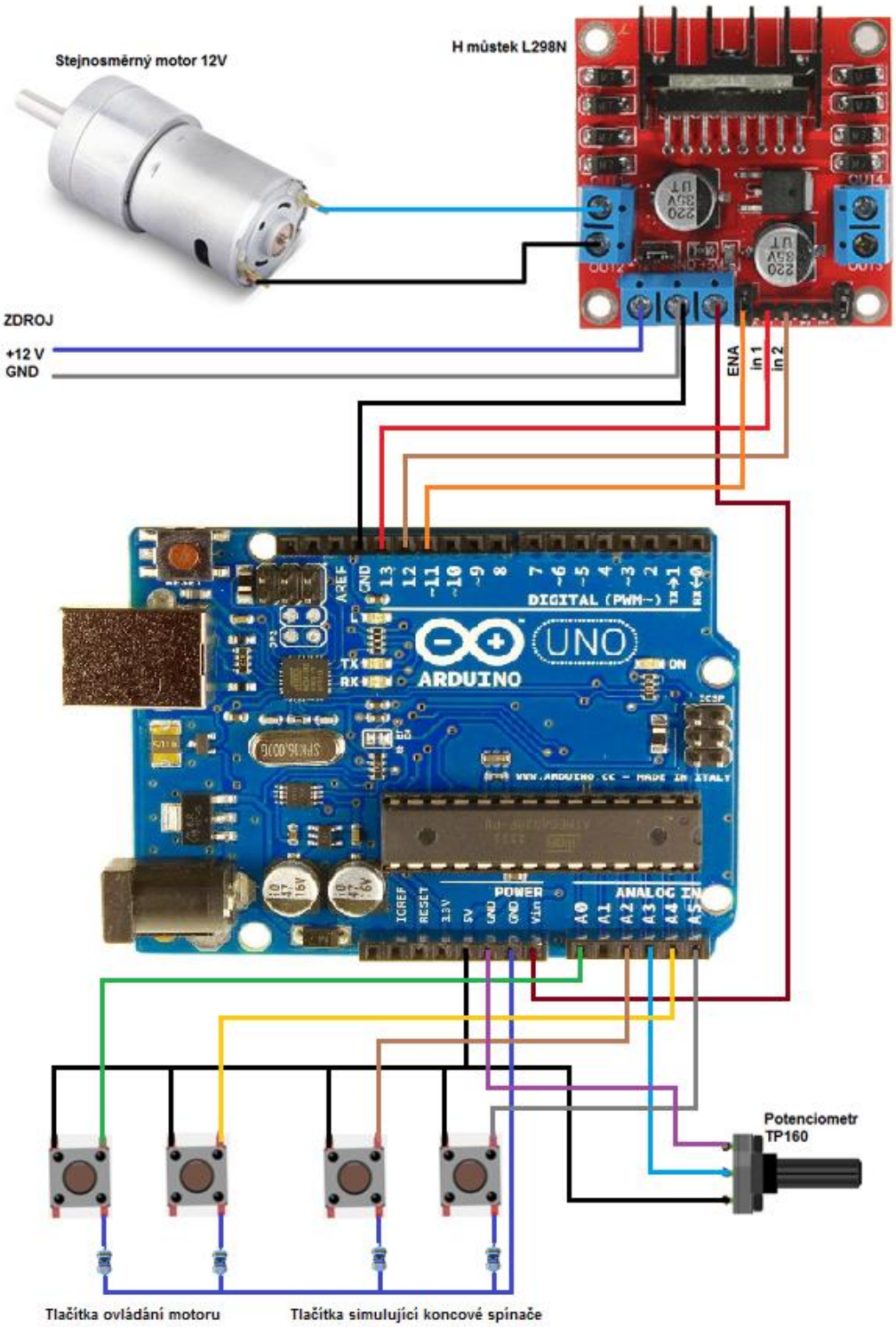
Pro každý z modelů c) a d) pro CAN

- Arduino UNO
- CAN bus MCP2515
- Potřebné vodiče
- USB propojovací kabel

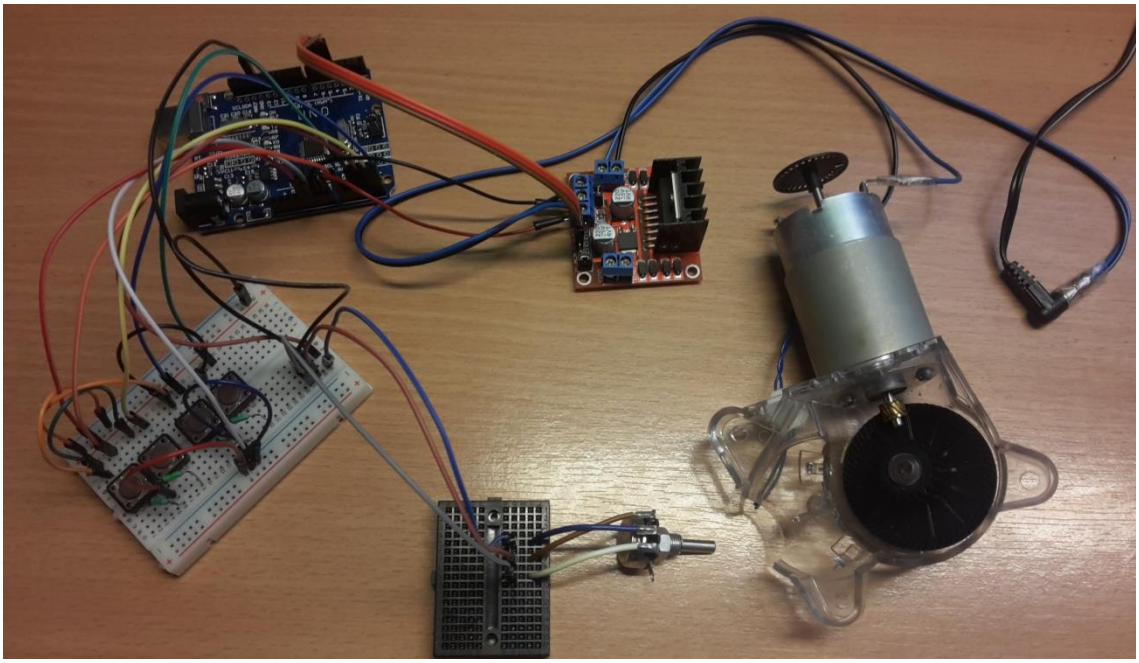
3.2.4 Zapojení zkušebních modelů

Zapojení modelu a) řízení motoru

Model řízení motoru je zapojen obdobně jako v rešeršní verzi (kapitola 3.1). Navíc jsou zde pouze spínače koncových poloh. Tlačítka ovládání motoru jsou připojena k pinům A0 a A4, tlačítka, která simulují funkci koncových spínačů k pinům A2 a A5. Analogový signál potenciometru je připojen k A3, potenciometr byl na tomto modelu připojen pouze pro experimentální účely, na rozdíl od rešeršní verze zde však nemá reálný dopad na řízení motoru a nemusel by zde být zapojený vůbec. Opět je využito odporů pro snížení proudu a zamezení přetěžování pinů. Můstek je připojen ke stejným pinům z předchozí verze (11, 12, 13). [17][19][8]



Obr. 26 Zapojení modelu a) zkušební verze [S3] (Zdroje [O26], upraveno a zkompilováno)

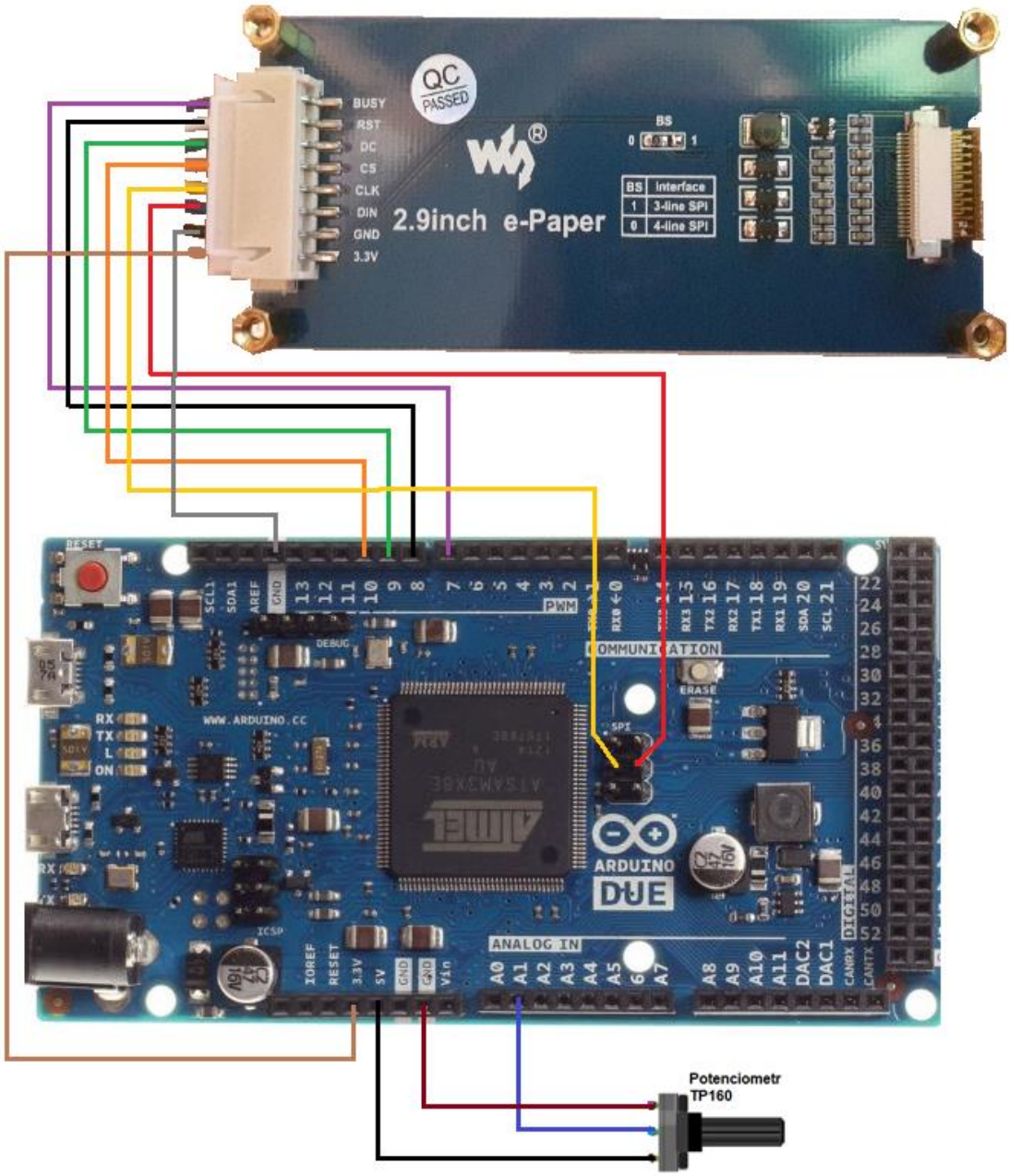


Obr. 27 Zapojení modelu b) zkušební verze ve skutečnosti

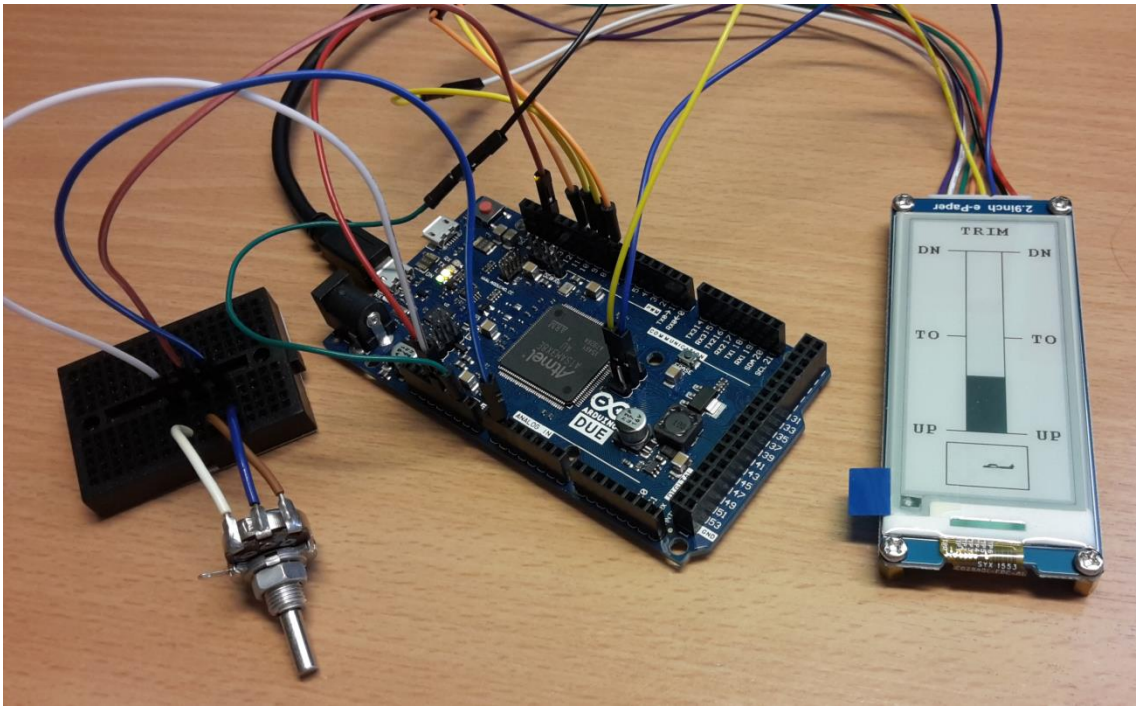
Zapojení modelu b) displej

Model b) byl realizován připojení displeje k desce DUE viz schéma na následující stránce. Potenciometr, coby snímač polohy, jehož polohu, potažmo i aktivitu pohybu, bude displej zobrazovat, je připojen stejným způsobem jako na předchozím modelu.

[22][20][26]



Obr. 28 Zapojení modelu b) zkušební verze [S3](Zdroje [O28], upraveno a zkompilováno)

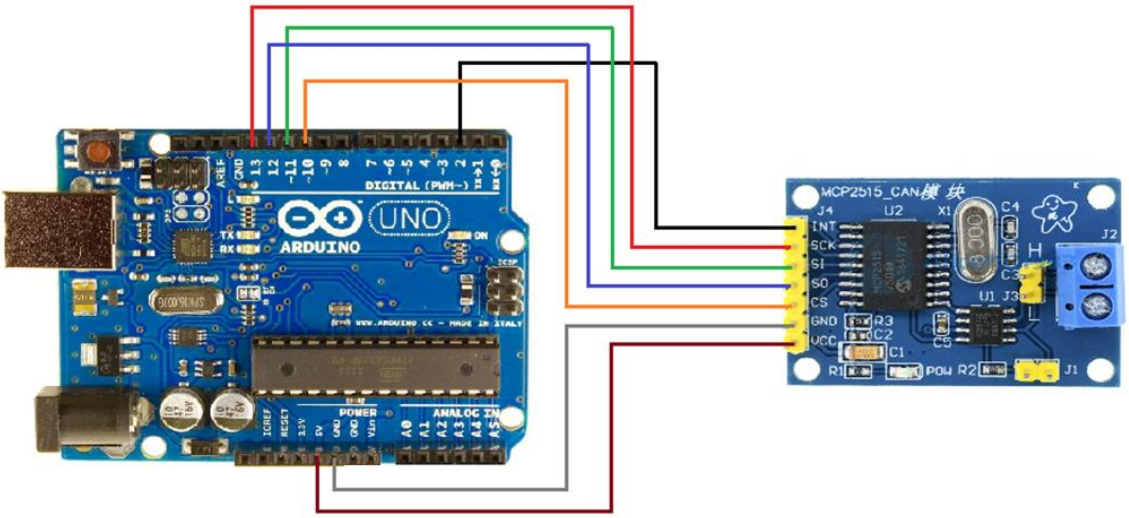


Obr. 29 Zapojení modelu b) zkušební verze ve skutečnosti

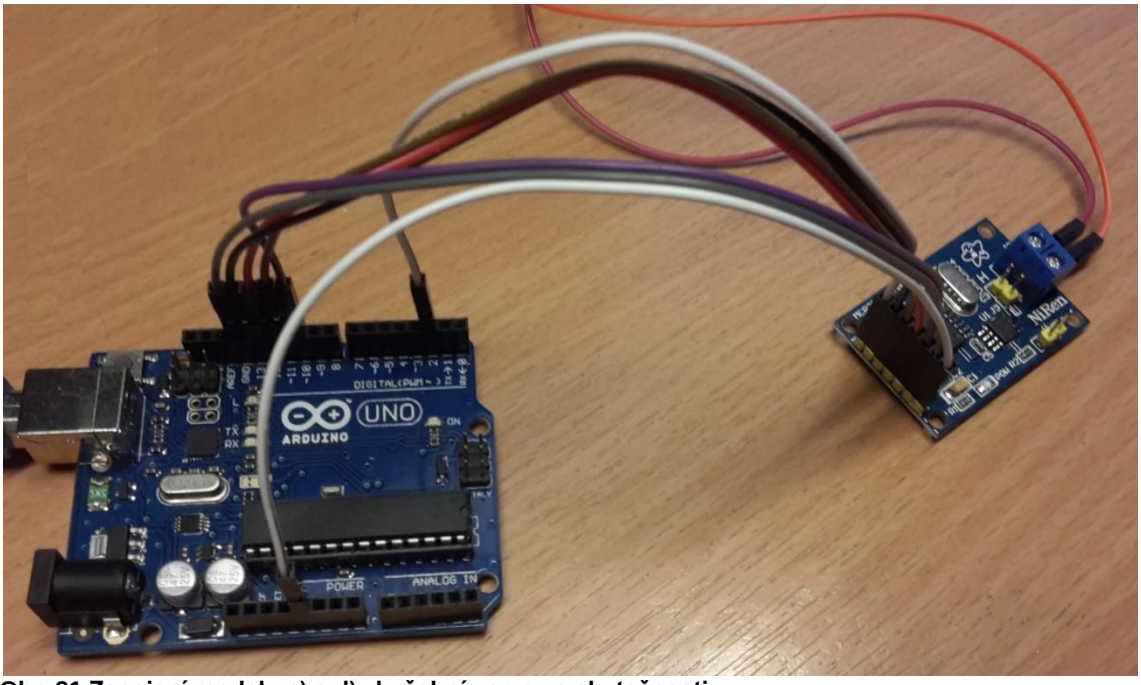
Zapojení modelu c) a d) pro CAN bus MCP2515

Zapojení, pro komunikaci CANu, je pro obě varianty realizováno mezi deskou UNO a MCP 2515.

[18]



Obr. 30 Zapojení modelu c) a d) zkušební verze [S3] (Zdroje [O30], upraveno a zkompilováno)



Obr. 31 Zapojení modelu c) a d) zkušební verze ve skutečnosti

3.2.5 Programy zkušebních modelů

Samotné programy jsou opět obsahem příloh. Zde jsou pouze průvodní slova.

Program pro model a) řízení motoru, je principiálně stejný s rešeršní verzí. Navíc jsou zde vstupy koncových spínačů, které jsou zahrnuty do podmínek spouštění motoru. Tento program na desce UNO zabírá 8% úložného místa pro program (2640 bytů).

Program pro model b) displej, zobrazuje polohu jezdce připojeného potenciometru. Displej disponuje dvěma úrovněmi paměti, do spodní části paměti se uloží a zobrazí rám a vyznačení poloh indikátoru, což se stane v části programu „setup“. V části „loop“ se poté vykresluje aktuální poloha jezdce pomocí černého obdélníku, který mění svoji velikost v závislosti na velikosti analogového signálu. Analogový signál je proto fragmentován na takový počet částí, který odpovídá výšce indikátoru. Přídavnou funkcí další verze stejného programu je grafické zobrazení chování letadla při trimování. Jedná se o tři druhy obrázků, které se zobrazují v situacích, kdy se černý obdélní „zvětšuje“, „zmenšuje“ nebo je stálý. V této verzi je zobrazení obrázků řešeno zdánlivě neproduktivní metodou navrženého obrazce na mapě po pixelech. Do budoucna vývoje systému však doporučuji zobrazovat obrázky jako piktogramy vytvořené pro rozšíření fontové knihovny, případně využít software dostupný z https://www.waveshare.com/wiki/File:Image2Lcd.7z, který požadované obrázky generuje automaticky. Tento program na desce DUE zabírá 6 nebo 7% úložného místa pro program (32092 nebo 39628 bytů) v závislosti na použití nebo nepoužití zmíněné přídavné funkce.

Program pro model c) obsahuje náležitosti pro odeslání informace o poloze polohy do CAN. Jako poloha poslouží velikost černého obdélníku, který se zobrazuje na displeji. Tato hodnota bude využita jak pro displej Slave Trimu, tak pro možnost diagnostiky. Hlavním důvodem tohoto řešení je skutečnost, že se odesílaná zpráva skládá z bytů. Pro zkoušku byla odeslána největší možná hodnota, tedy maximální výška obdélníku,

která nastane v dolní krajní poloze. V programu modelu d) jsou údaje pro přijímání odeslané zprávy, z ní je dekódován odeslaný údaj, ten bude použit pro zobrazení polohy na displeji Slave Trimu. Každý z programů pro modely c) a d) na deskách UNO nezabírá více než 11% úložného místa pro program (3794 bytů). V programu pro přijetí zprávy modelu d) je chyba knihovny, která komplikuje příjem, předpokládám, že novější verze knihovny budou fungovat bez problému.

4 Akční členy

Akční členy slouží k využití zpracovaných vstupních signálů mechanickým způsobem. Na základě vstupů, které se zaslouží o zprostředkování výstupního signálu, je tento signál následně využit mechanickým způsobem, v mém případě k řízení motoru serva. Toto servo je komponenta, která je součástí finálního trimovacího systému této práce. Bude instalováno na výškovém kormidle, zřejmě obdobně jako na obrázku 6.

4.1 Návrh serva

Podkladem pro návrh serva bylo zařízení Ray Allen T2-10A. Jedná se o servo jehož chod je dlouhý 26 milimetrů, doba vysunutí z koncové polohy je 16 sekund. Ovládací síla je přibližně 180 N.

Systémy vyvážení Ray Allen byly navrženy speciálně pro ultralehká letadla. Serva Ray Allen jsou dostatečně malá, takže se vejdou do řídících ploch nebo mohou pomocí pružin zavádět síly přímo do mechanismu řízení. Snímač polohy je umístěn uvnitř serva. Motor serva a snímač polohy jsou od sebe z důvodu ochrany proti elektrické poruše izolovány.

Obr. 32 Servo Ray Allen T2-10A (Zdroj [O32])

Po principiální a konstrukční stránce je můj návrh tímto servem silně inspirován. V mém návrhu bylo zvoleno ozubení bez korekcí působící podřezání, přidány byly i další optimalizační konstrukční úpravy. [12]

4.1.1 Převodovka

Převodovka je navržena reverzně na základě již běžně provozované převodovky ze serva Ray Allen T2-10A, veškerá funkčnost je již ověřena praxí. Kola ze zvolených

materiálů snesou vysoká zatížení i při horších vnějších podmínkách, díky těmto skutečnostem a faktu, že výkon motoru je 2W neprovádím pevnostní výpočty.

Vstup do převodovky je hřídel motoru s pastorkem, výstupem je poslední kolo s nábojem, v němž je pohybový šroub zajištěn čepem. Převodovka má čtyři soukolí, kola na stejné hřídeli jsou vždy spojena a tvoří tak jediný celek, krouticí moment je tak zbytečné přenášet na předlohové hřídele. Modul je u všech kol volen stejný 0,4 mm. Všechna kola jsou čelní s přímými zuby bez korekcí.

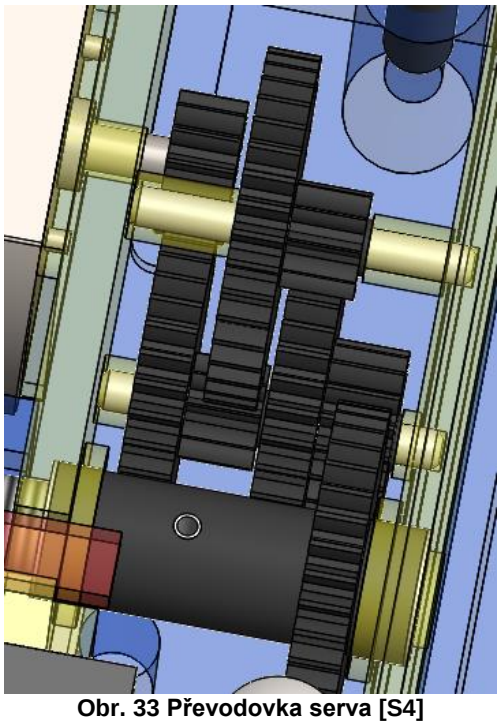
Jednotlivé údaje navržených kol:

Název	Počet zubů z	Roztečný průměr d [mm]	Šířka ozubení bw [mm]	Osová vzdálenost a [mm]	Převodový poměr i
Kolo 1 (Vstupní)	14	5,6	3	10,2	2,6429
Kolo 2	37	14,8	2,3		
Kolo 3	10	4	3,25	9,4	3,7
Kolo 4	37	14,8	2,3		
Kolo 5	10	4	3,25	9,4	3,7
Kolo 6	37	14,8	2,3		
Kolo 7	18	7,2	3,25	11	2,0556
Kolo 8 (Výstupní)	37	14,8	2,3		

Tab. 1 Parametry navržených kol převodovky

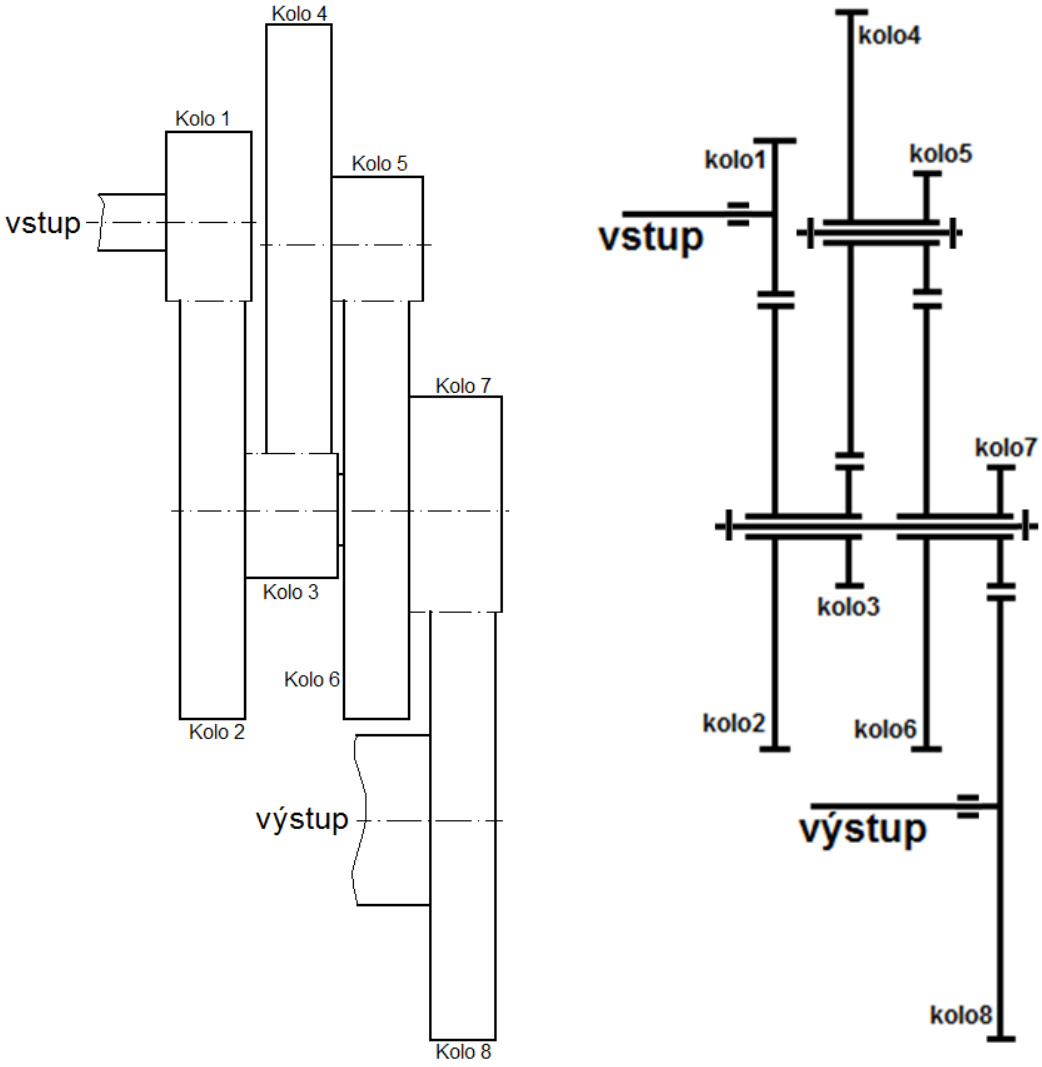
$i_c = i_{12} \cdot i_{34} \cdot i_{56} \cdot i_{78} = 2{,}6429 \cdot 3{,}7 \cdot 3{,}7 \cdot 2{,}0556 = 74{,}3744$

Celkový převodový poměr i_c je součinem všech dílčích převodových čísel, tedy 74,3744.



Obr. 33 Převodovka serva [S4]

Koncepční schéma převodovky



Obr. 34 Koncepční schéma převodovky serva [S2][S3]

4.1.2 Motor

Požadované otáčky motoru jsem zjistil následujícím způsobem. Jestliže je dráha pohybu serva dlouhá 26 mm a plné vysunutí má z koncové polohy zabrat 16 sekund, pak je rychlost vysouvání 1,625 mm/s. Pohybový šroub serva má závit M4, jemuž náleží rozteč P rovna 0,7 mm. Při jedné otáčce za sekundu posledního kola se servo vysune právě o 0,7 mm. V otáčkách výstupního kola a rychlostí pohybu serva platí přímá úměrnost, aby byla docílena požadovaná rychlost je tedy třeba 2,3214 ot/s na výstupním kole.

Celkový převodový poměr převodovky i_c je 74,3744. Jedná se tedy o převod, který bude 74,3744 krát snižovat otáčky motoru pro správný pohyb serva. Jestliže jsou požadované otáčky na výstupu 2,3214 na vstupu musí být tolikrát vyšší, jaký je celkový převodový poměr, a sice 172,6527 ot/s, neboli 10395 ot/min.

Pro výpočet potřebného výkonu znám přibližně maximální požadovanou ovládací sílu F=180N, která odpovídá osové síle v pohybovém šroubu. Dle vztahu jsem určil potřebný krouticí moment na výstupním kole, aby tento požadavek na sílu byl splněn.

$$M_{k8} = \frac{F \cdot d_2}{2} \cdot \mathrm{tg}(\varphi' + \gamma)$$

kde d_2 je střední průměr závitu M4 s roztečí P=0,7 mm. φ´ je třecí úhel, který při třecím součiniteli f=0,1 odpovídá 5,7°. γ je úhel stoupání šroubovice vypočten ze vztahu:

$$\gamma = \mathrm{arctg}\,\frac{P}{\pi \cdot d_2} = \mathrm{arctg}\,\frac{0{,}7}{\pi \cdot 3{,}545} = 3{,}6°$$

$$M_{k8} = \frac{180 \cdot 3{,}545}{2} \cdot \mathrm{tg}(5{,}7 + 3{,}6) = 52{,}3\ \mathrm{Nmm}$$

Potřebný krouticí moment na hřídeli motoru jsem zjistil z rovnice

$$\frac{M_{k8}}{M_{k1}} = i_c$$

$$M_{k1} = \frac{M_{k8}}{i_c} = \frac{52{,}3}{74{,}3744} = 0{,}7\ Nmm$$

Potřebný ideální výkon je z krouticího momentu snadno dopočítán.

$$P = M_{k1} \cdot 2\pi \cdot n = 0{,}7E-3 \cdot 2\pi \cdot 10000/60 = 0{,}73W$$

Tento ideální výkon je třeba zvýšit o ztráty vlivem účinnosti závitu a převodovky

$$\eta_z = \frac{tg\,\gamma}{tg(\gamma + \varphi')} = \frac{tg\,3{,}6}{tg(3{,}6 + 5{,}7)} = 0{,}384$$

$$\eta_p = \eta_{12}\,\eta_{34}\,\eta_{56}\,\eta_{78} = 0{,}98^4 = 0{,}922$$

$$\eta_c = \eta_p\,\eta_z = 0{,}384 \cdot 0{,}922 = 0{,}353$$

Celkový požadovaný výkon motoru tedy je:

$$P_p = P \cdot \frac{1}{\eta_c} = 0{,}73 \cdot \frac{1}{0{,}353} = 2\ W$$

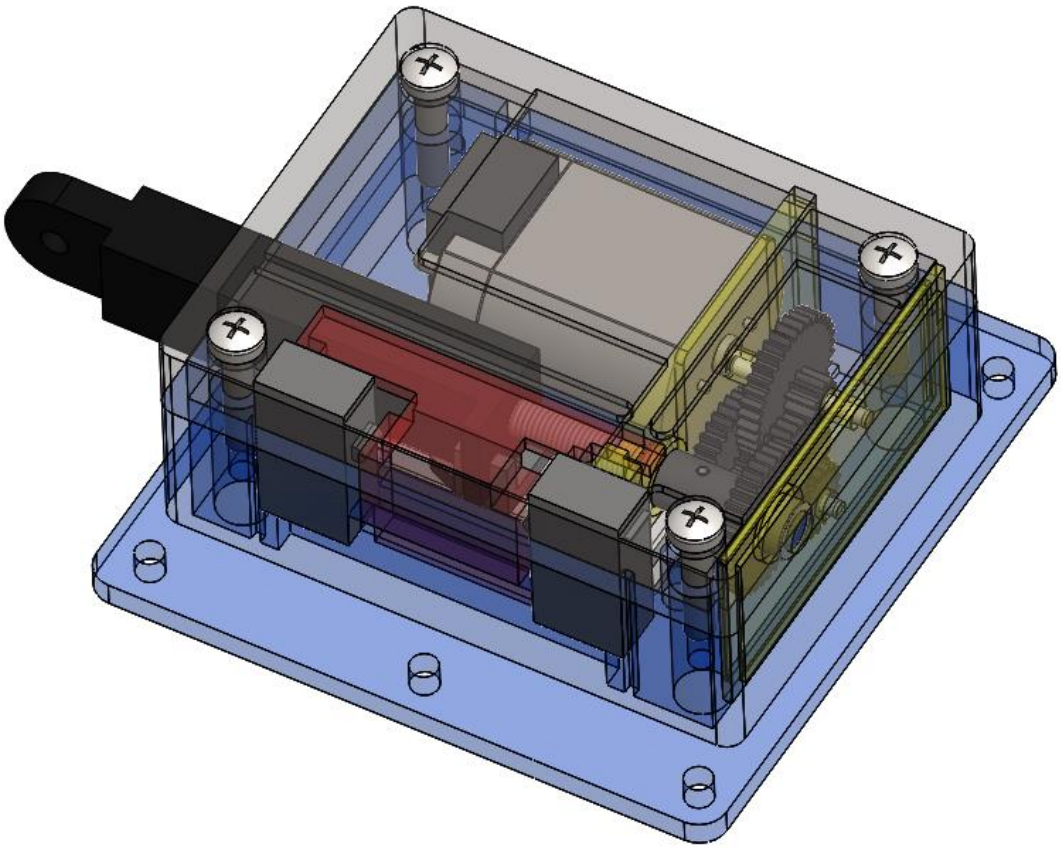
Hlavní požadované parametry pro stejnosměrný motor jsou, kromě rozměrů, napájení na 12V, otáčky 10000 ot/min, výkon 2 W.

Těmto požadavkům vyhovuje například motor FK-280 8000RPM 10000RPM Small Electric 6V 12V DC Motor Dostupný z [13]. [13][14]

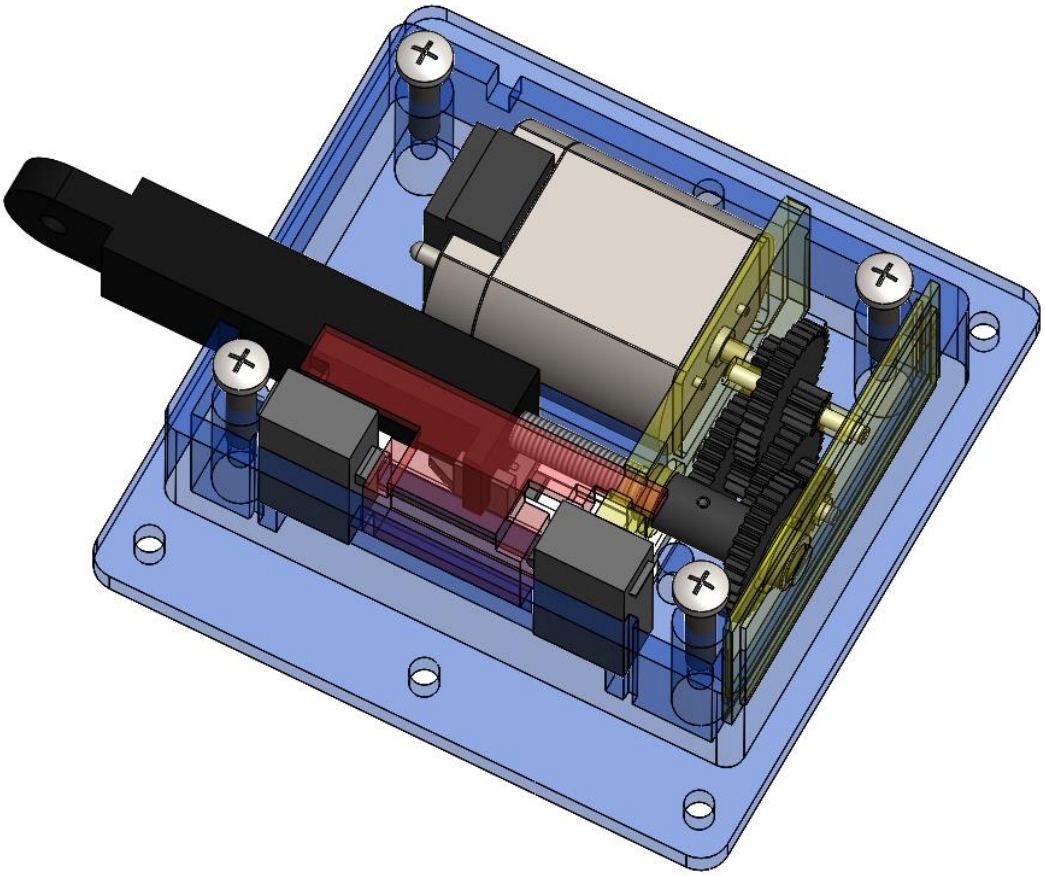
4.2 Konstrukce serva

Pro výstup reverzního návrhu byl vytvořen CAD model na základě Ray Allen T2-10A. Jako vývojové prostředí jsem zvolil konstrukční program SolidWorks 2017. Kromě modelu jsem vytvořil rozložený pohled využitelný pro případnou montáž a následující popis jednotlivých částí. Součástí CAD modelu není žádná kabeláž, model však počítá s její instalací do míst k tomu určených a to od motoru, potenciometru a koncových spínačů.

Navržené servo se skládá ze tří hlavních částí. První část tvoří uložení stejnosměrného motoru, který je ukotven mezi spodní díl a víko pomocí zajišťovací součástky a přepážku převodovky. Druhá část tvoří převodovka s celkem čtyřmi hřídeli, která redukuje otáčky motoru pro správný pohyb serva. Předlohové hřídele jsou pevné, kroutící moment zde přenáší pouze sdružená kola. Výstupní kolo převodovky je spojeno s nábojem, který udává do pohybu šroub spojený se samotným ramenem serva (konečným pohybovým výstupem serva). Do třetí části serva, kromě zmíněného výstupního ramene, spadá odměřování jeho polohy (polohy plošky) pomocí potenciometru, koncové spínače a jejich ovladač. Spodní a horní díl vnějšku je spojen pomocí čtyř šroubů.

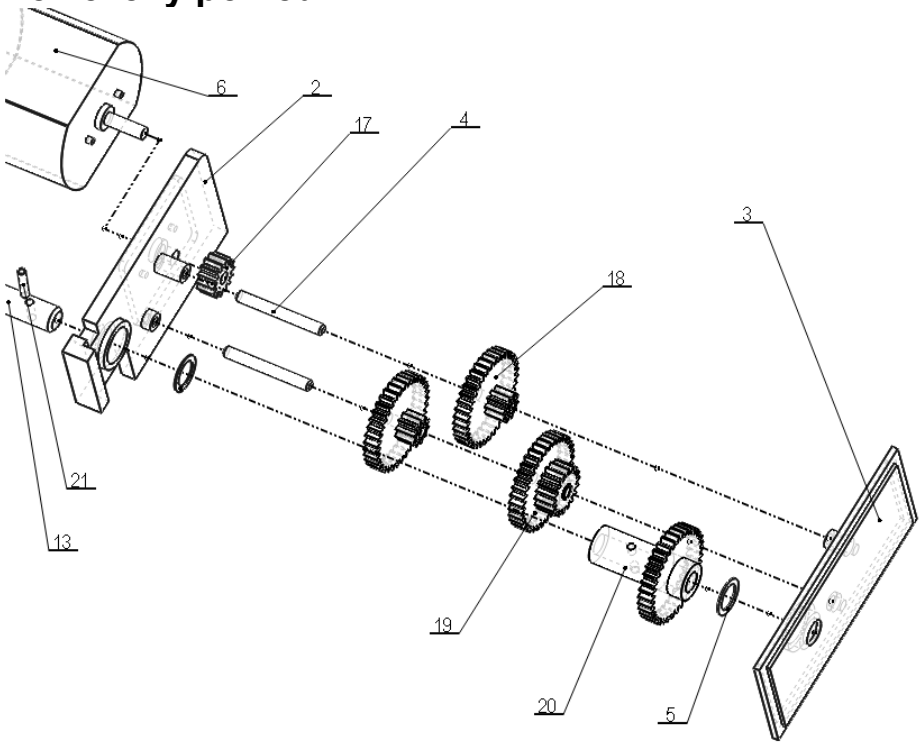


Obr. 35 Navržené servo v CAD prostředí [S4]

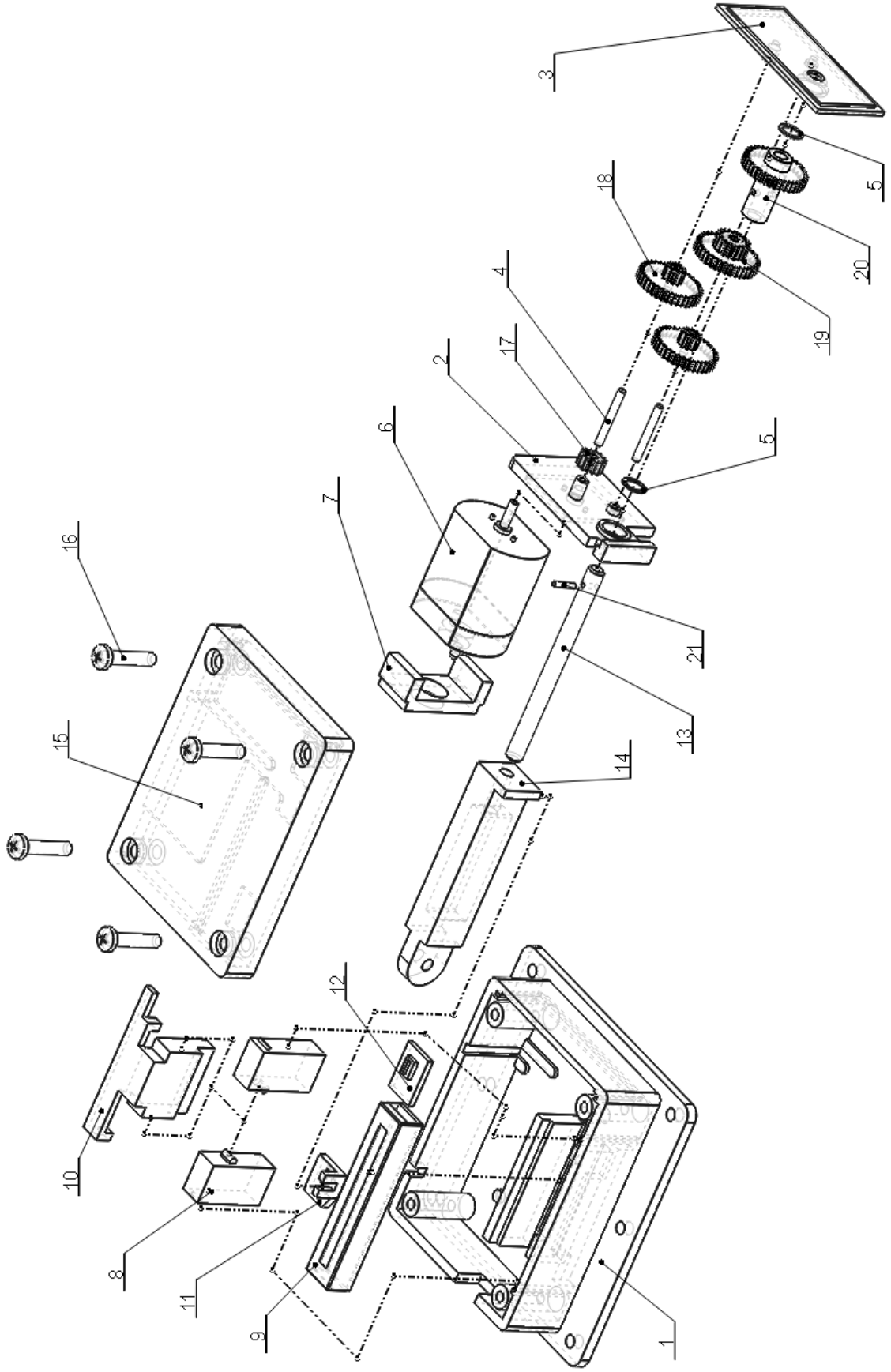


Obr. 36 Navržené servo v CAD prostředí bez víka [S4]

4.2.1 Rozložený pohled



Obr. 37 Rozložený pohled serva - detail převodovky [S4]



Obr. 38 Rozložený pohled serva [S4]

4.2.2 Obecný seznam částí

Pozice	Název části	Navržený materiál
1	Spodní díl	PA 6 MG
2	Přepážka	PA 6 MG
3	Stěna	PA 6 MG
4	Hřídel	11500 (E295)
5	Kroužek	11500 (E295)
6	Stejnosměrný motor	
7	Zajištění motoru	PA 6 MG
8	Koncový spínač	PA 6 MG
9	Potenciometr	
10	Ovladač koncových spínačů	PA 6 MG
11	Horní část jezdce	PA 6 MG
12	Spodní část jezdce	PA 6 MG
13	Pohybový šroub	PA 6 MG
14	Výstup serva	PA 6 MG
15	Víko	PA 6 MG
16	Spojovací materiál	
17	Kolo A	PA 6 GF30
18	Kolo B	PA 6 GF30
19	Kolo C	PA 6 GF30
20	Kolo D	PA 6 GF30
21	Čep	11500 (E295)

Tab. 2 Seznam částí serva

Pro ozubená kola jsem navrhl plast PA 6 GF30, jedná se o polyamid 6 zesílený přidáním 30% skleněných vláken a grafitu, vysoce odolný vůči abrazi, tlaku a ohybu, odolný vůči stárnutí a klimatickým podmínkám. Pro ostatní plastové komponenty jsem navrhl materiál PA 6 MG, jedná se o polyamid s přídavkem disulfidu molybdenu a grafitu, odolný proti stárnutí a povětrnostním podmínkám. [15]

4.2.3 Popis montáže

Při montáži postupujeme tak, že si připravíme potenciometr (1) zakomponujeme do něj jezdec (11)(12). Do spodního dílu (1) umístíme zkompletovaný potenciometr a koncové spínače (8), které jsou přilepeny na svá místa k výstupkům na dílu. Dále si připravíme převodovku, skrze přepážku (2) opatříme hřídel motoru (6) ozubeným kolem A (17). Přepážka má dva delší náboje, do nich jsou uloženy dva hřídele (4). Do otvoru s největším průměrem přijde kroužek (5) a na, zatím, letmé hřídele kolo B (18), pro hřídel blíže motoru a kola B (18) následně C (19), pro hřídele dále od motoru. Na pohybový šroub (13) našroubujeme výstup serva (14). Dále je do největšího otvoru stěny nasazeno kolo D (20), které opatříme pohybovým šroubem a zajistíme čepem (21). Do otvoru stěny (3) je připraven kroužek (5), tato stěna se poté nasadí tak na letmé hřídele a kolo D. Na motor osadíme přípravek pro jeho správné uložení (7) a celou převodovku i s motorem uložíme do spodního dílu tak, aby byl výstup serva spojen s jezdcem. Mezi koncové spínače je nasazen jejich ovladač (10). Na celý spodní díl je nyní usazeno víko (15), zajištěno čtyřmi šrouby (16).

5 Návrh elektronické desky

5.1 Úvodní přehled

Po úspěšném otestování principů, programů zkušebního provedení a konstrukci serva pokračuji vývojem řídícího hardwaru pro finální verzi trimu. Navržený hardware bude ovládán modifikovanými a sloučenými programy ze zkušební verze (kapitola 3.2).

Pro finální trimovací systém, který má být uveden do provozu, je třeba vyvinout elektronickou desku přímo pro tento účel. Pro její konstrukci bude použit procesor z desky Arduino MEGA, veškeré nepotřebné vývody původní desky budou odebrány, zůstanou pouze ty využité. Deska musí mít odpovídající ochranné prvky při napájení a její součástí bude CAN bus a H můstek.

Použitý procesor ATmega2560 jsem zvolil v souladu se skutečností, že předpokládaná velikost finálního programu je vyšší než úložné místo pro program na desce UNO, tento závěr vyplívá z již navržené zkušební verze. Deska UNO rovněž nemá potřebné rozhraní pro všechny nezbytné komponenty. Procesor z desky DUE nebyl použit, protože se pro vývoj jiných desek obecně nedoporučuje a software KiCad který jsem pro návrh použil, neobsahuje knihovnu tohoto procesoru.

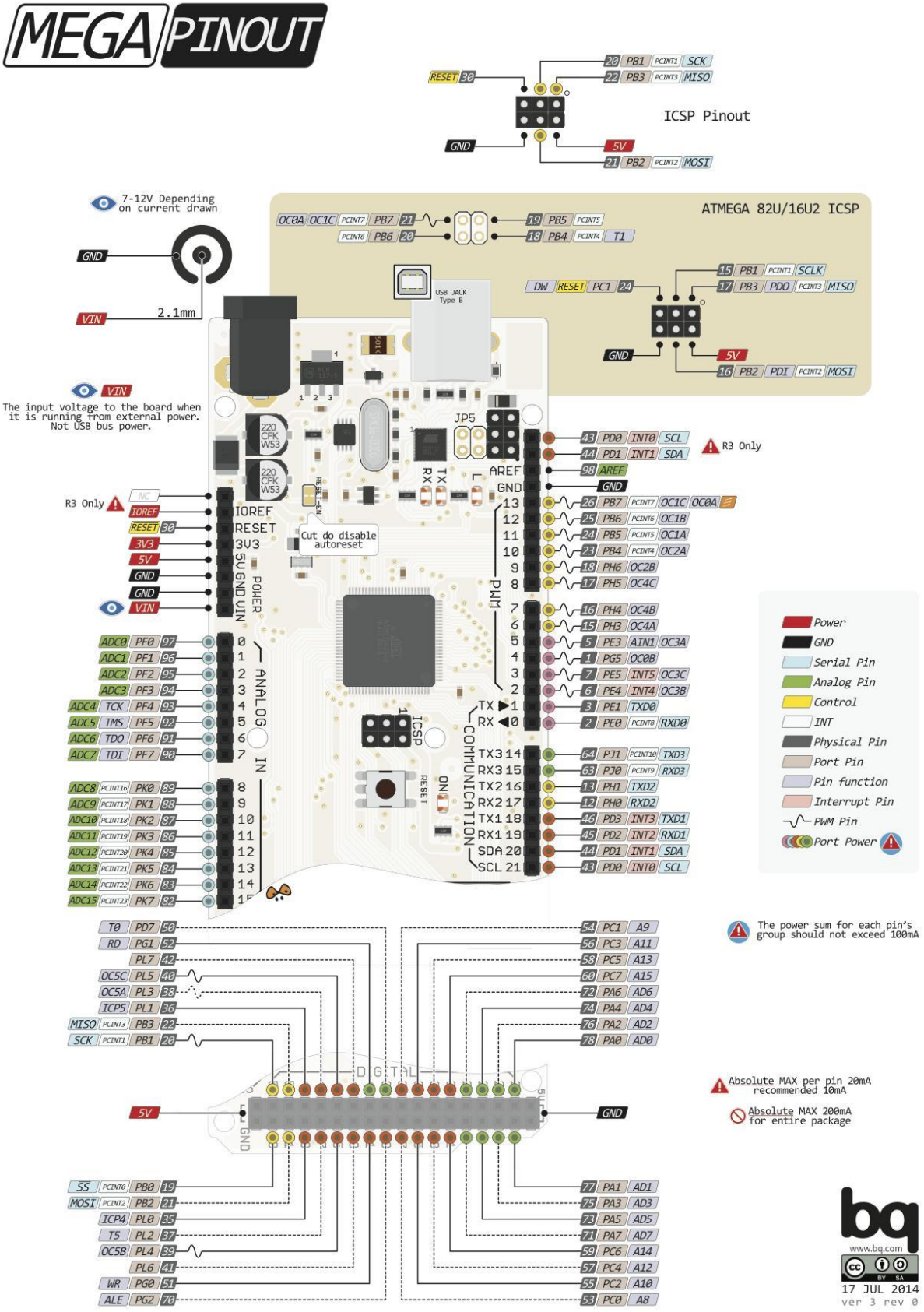
Výkres s kompletním návrhem je obsahem přílohy 3.

5.2 Postup návrhu

5.2.1 Procesor

Vlastní návrh založený na ATmega2560 jsem započal zvolením použitých vývodů procesoru, které budou použity pro připojení displeje, potenciometru, tlačítek, koncových spínačů, CAN busu a H můstku. Pro výběr jsem použil následující schéma a znalosti nabyté při tvorbě zkušební verze (kapitola 3.2).

Závěr volby jednotlivých vývodů procesoru zobrazuje tabulka 3.



Obr. 39 MEGA Pinout (Zdroj [O39])

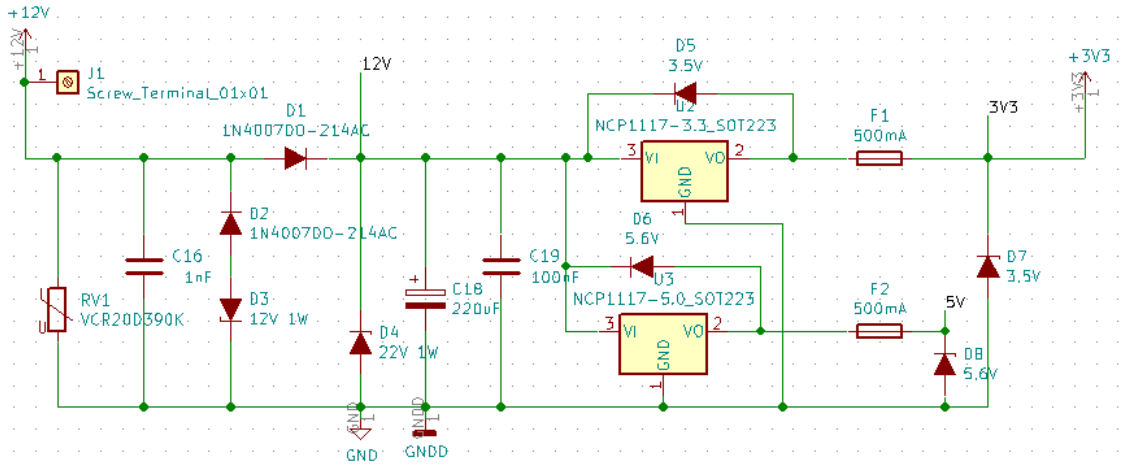
Displej		
Název pinu komponenty	Pin na desce MEGA	Odpovídající vývod CPU
BUSY	7	PH7
RST	8	PH5
DC	9	PH6
CS (1)	10	PB4
DIN	11 MOSI	PB5
CLK	13 SCK	PB7
GND	GND	
3,3V	3,3V	
Potenciometr		
Název pinu komponenty	Pin na desce MEGA	Odpovídající vývod CPU
Analogový výstup	A1	PF1
GND	GND	
3V3	3V3	
CAN bus		
Název pinu komponenty	Pin na desce MEGA	Odpovídající vývod CPU
INT	2	PE4
SCK	52	PG1
MOSI	51	PG0
MISO	50	PD7
CS (2)	53	PC0
GND	GND	
VCC	5V	
H můstek		
Název pinu komponenty	Pin na desce MEGA	Odpovídající vývod CPU
in1	3	PE5
in2	5	PE3
GND	GND	
VREF	12V	
Koncové spínače		
Název pinu komponenty	Pin na desce MEGA	Odpovídající vývod CPU
Digitální vstup 1	56	PC3
Digitální vstup 2	57	PC4
GND	GND	
3V3	3V3	
Tlačítka		
Název pinu komponenty	Pin na desce MEGA	Odpovídající vývod CPU
Digitální vstup 3	54	PC1
Digitální vstup 4	55	PC2
GND	GND	
3V3	3V3	

Tab. 3 Volba vývodů procesoru

5.2.2 Napájení a ochrana

Deska bude napájena ze sítě letadla, která je standardně 12V. S ohledem na použité komponenty a jejich logiku je nezbytné, aby napájecí část desky obsahovala dva regulátory napětí pro snížení napětí hlavního napájení. Tyto regulátory zajišťují převod původních 12V na 3,3V a 5V. 12V je využito pouze pro pohon motoru, 5V pouze pro CAN moduly a 3,3V pro logiku, napájení procesoru, potenciometru, tlačítek a koncových spínačů.

Ochranu proti přetížení, přepólovaní, špičkám a poklesům v síti letadla zajišťují vhodně umístěné kondenzátory, diody a varistor. Pro ochranu při kritickém poškození zařízení jsou přítomny pojistky.

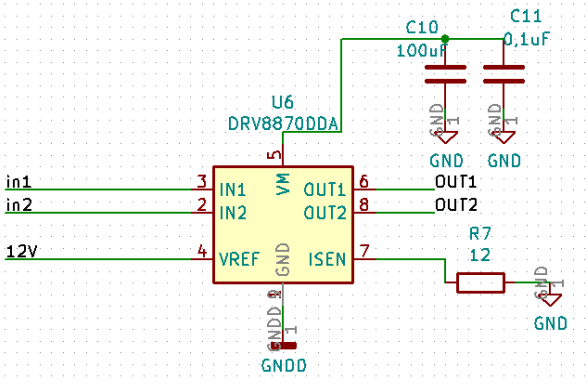


Obr. 40 Napájení a ochrana [S5]

5.2.3 H můstek

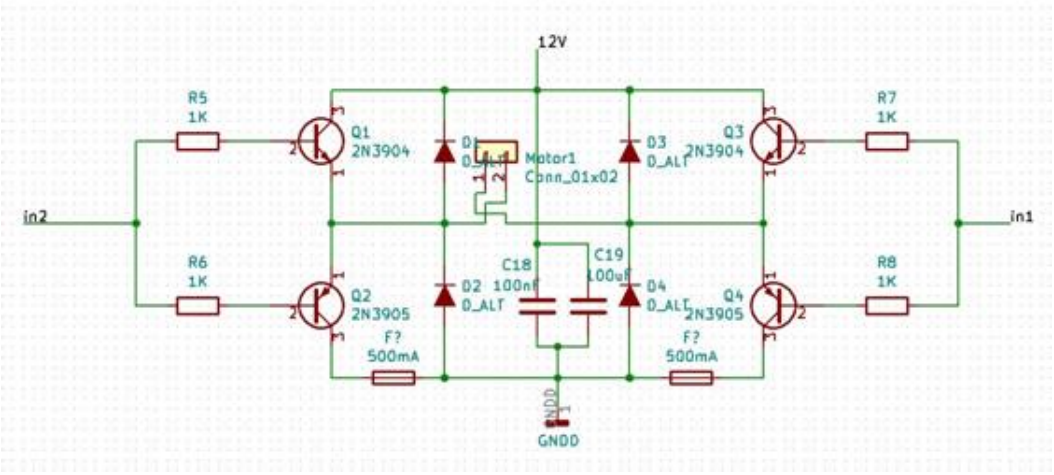
H můstek je na desku instalován jako již hotový celek, který disponuje vlastní ochranou logikou. Tato logika zafunguje v případě, kdyby se jakýmkoli nestandardním způsobem stalo, že by oba vstupy do můstku byly jedničkové. Program tuto situaci sám o sobě nepřipouští, je však třeba uvažovat možnost poškození CPU apod.

Odpor označený jako R7 určuje proud pro motor, určí se dle informace z datasheetu: $R7 = \frac{Vref}{10 \cdot I}$ [16]



Obr. 41 H-můstek [S5]

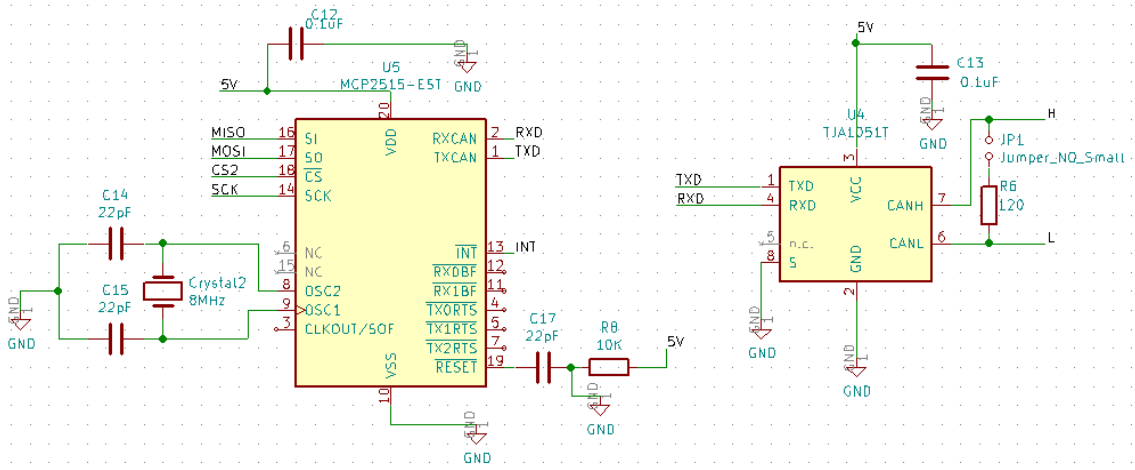
Zde je alternativní řešení můstku pro desku, které nebylo použito. Ochranu v situaci obou jedničkových vstupů zde zajišťují pouze pojistky, což se pro mou aplikaci nehodí, neboť po jejich zafungování by bylo ovládání plošky definitivně vyřazeno z činnosti.



Obr. 42 H-můstek alternativní [S5]

5.2.4 CAN bus

Funkci CANu zajišťují dva celistvé moduly, CAN bus a jeho rozhraní, viz obrázek 43. Mezi CANH a CANL je možné pomocí jumperu připojit standardní závěrný odpor pro CAN. Zda bude tento odpor skutečně použit, není zřejmé, pokud neznám jakým způsobem je CAN zařízen v celém letadle. [18]



Obr. 43 CAN bus [S5]

5.3 Design desky a rozmístění komponent

Kompletní výkres návrhu elektroniky je obsahem přílohy 3. Níže je navržené rozložení částí a jejich propojení na desce. Při návrhu jsem využil datasheetů všech komponent.

Při designování jsem postupoval tak, aby krystaly a jejich rezonanční kondenzátory byly co nejblíže svým komponentům, stejně jako kondenzátory, které chrání při poklesech v napájení. Co nejblíže procesoru jsem umístil i CAN moduly. H můstek a

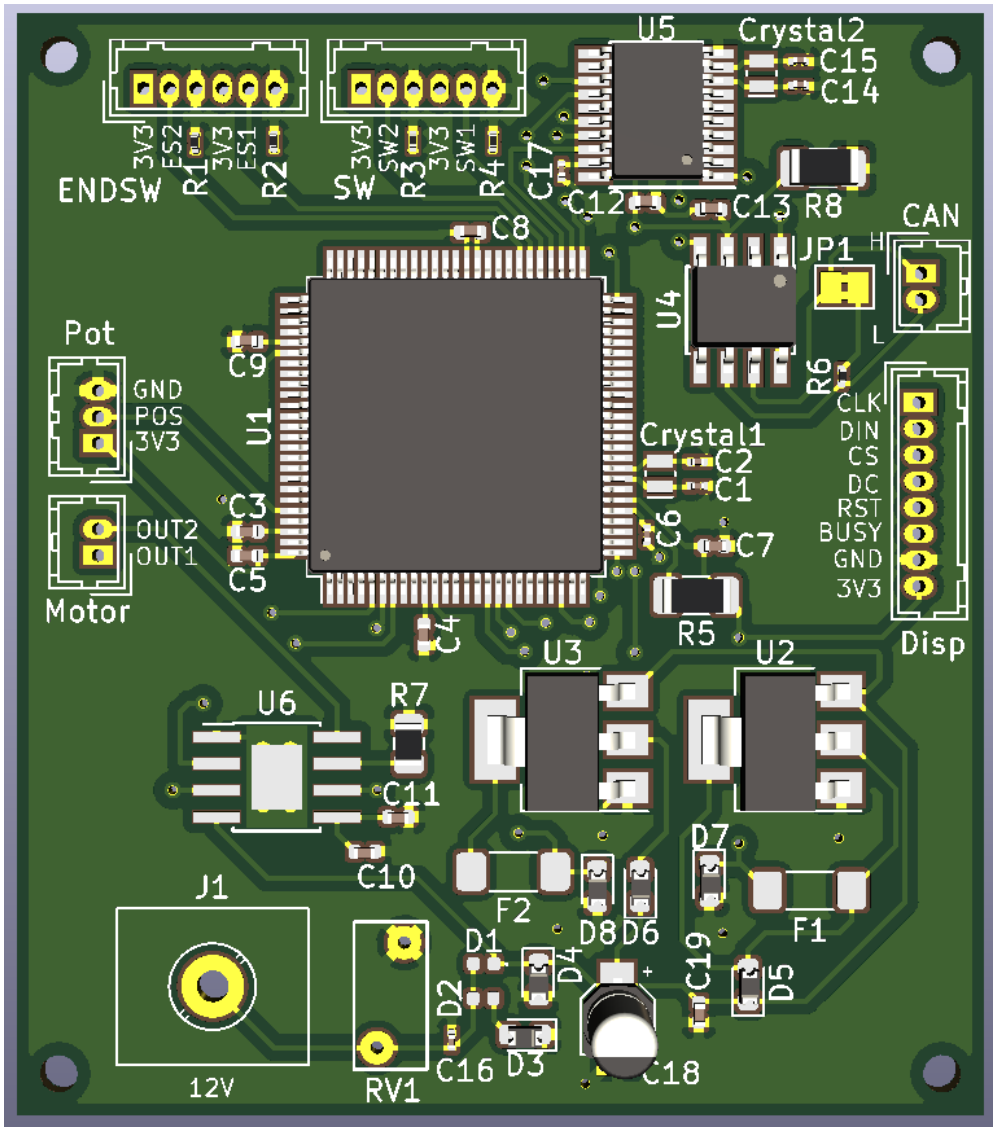
napájení jsou na vhodných místech o něco dále, přičemž můstek je uzemněn co nejblíže zemi u 12V. Piny pro zapojení jsou vždy blízko svých příslušných částí.

Deska je řešená jako oboustranná, neboť je zde mnoho míst, kde je nezbytné křížit propojení. Napájení vychází z jedné části do všech potřebných míst a vývodů z procesoru, na různých místech, je příliš mnoho, aby bylo možné dosáhnout optimálních rozměrů využitím pouze jedné strany desky.

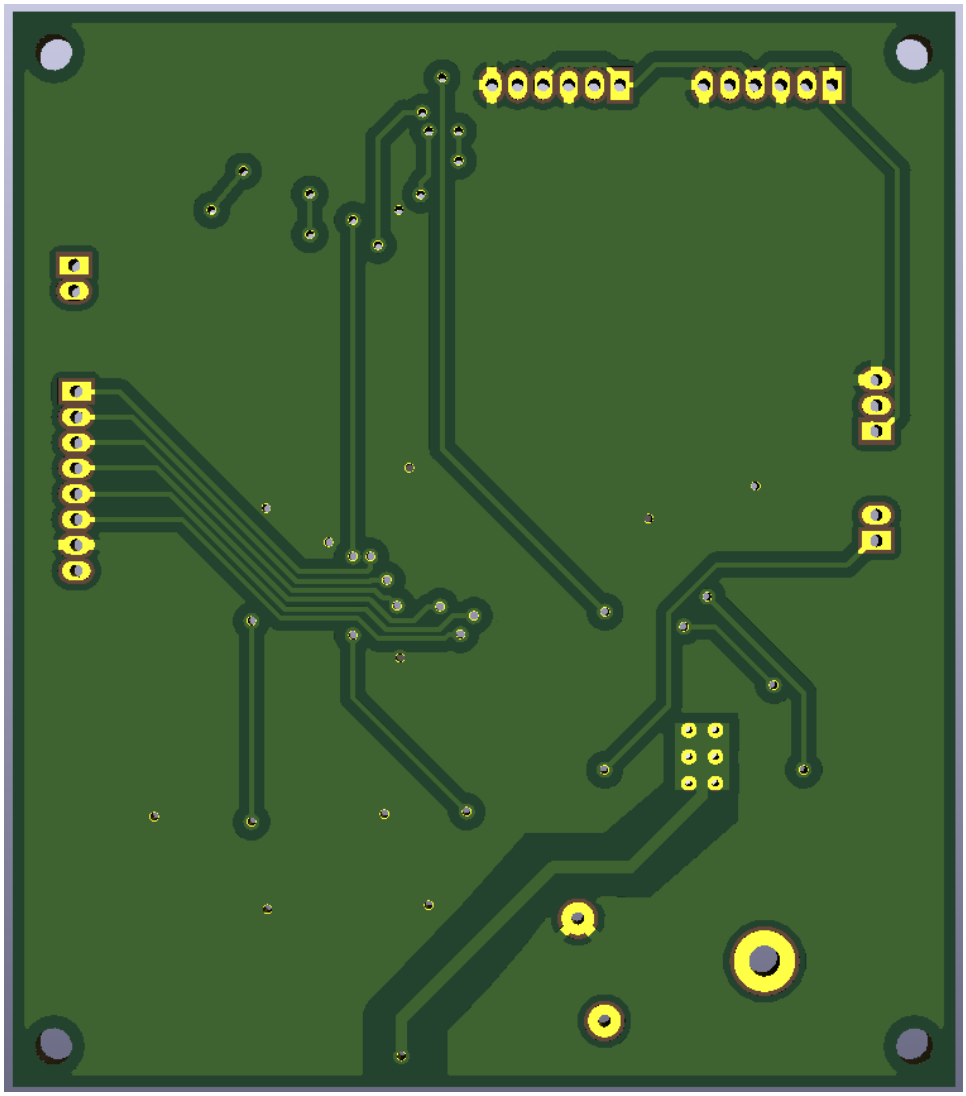
Země jsou propojeny rozlitou mědí (tzv. ground plane), na obou stranách desky.

Rozměry této navržené desky jsou 52,5 x 46 mm, pro instalaci do přístrojové schránky jsem zvolil čtyři otvory o průměru 1,8 mm.

Pro instalaci desky je u jednotlivých pinů napsáno kam se k danému vnějšímu prvku připojují. Jsou zde rozdělené skupiny pinů pro koncové spínače (ENDSW), pararelně zapojená ovládací tlačítka pilotů (SW), CAN, potenciometr (Pot), motor a displej (Disp). Své popisy mají i všechny komponenty osazené na desce, které odkazují na výkres v příloze 3. [16][18][21][22][27]



Obr. 44 Navržená deska - přední strana [S5]

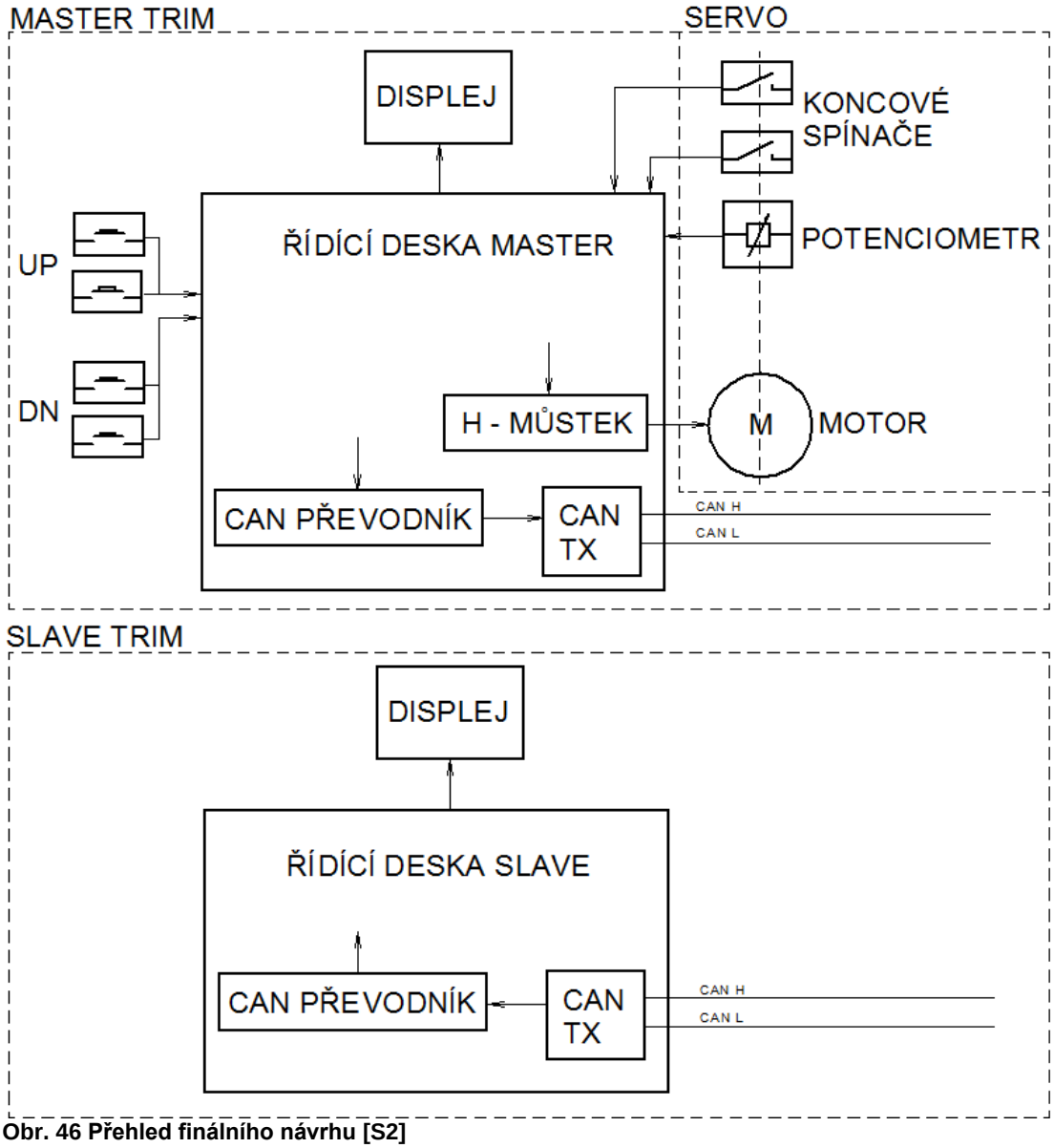


Obr. 45 Navržená deska - zadní strana [S5]

6 Finální podoba

Návrh finální podoby se má v této práci maximálně přiblížit reálnému řešení v letadle. Principy a programy jsou převzaty ze zkušební verze (kapitola 3.2), byl využit hardware vyvinutý v předchozí kapitole přesně pro tuto aplikaci a servo z kapitoly 4.

6.1 Přehled návrhu



Obr. 46 Přehled finálního návrhu [S2]

Navržená deska je určena primárně pro master trim u pilota, pro kopilota může být použita buď identická deska, nebo její modifikace. Bude-li použita deska identická, pak na ní nebudou připojena tlačítka (tlačítka kopilota jsou připojena paralelně k tlačítkům pilota), koncové spínače, potenciometr a H-můstek. Na případně použité modifikované desce nebudou přítomny konektory, ani veškeré komponenty pro funkci tlačítek, koncových spínačů, potenciometru a H můstku.

K desce Master Trimu jsou připojena paralelně tlačítka pilotů, koncové spínače, potenciometr, H můstek, CAN a displej. Na desce Master bude nahrán sjednocený a upravený program z modelů zkušební verze:

a) model řízení motoru - Příloha 2.a

b) model řízení displeje - Příloha 2.b

c) model CAN části „Master Trim“ - Příloha 2.c

K desce Slave Trimu je připojen pouze CAN a displej. CAN bus zde funguje jako přijímač polohy jezdce potenciometru, který odesílá CAN bus desky Master trimu. Na desce Slave bude nahrán sjednocený a upravený program z modelů zkušební verze:

b) model řízení displeje - Příloha 2.b

d) model CAN části „Slave Trim“ - Příloha 2.d

Přehled finální verze zobrazuje příloha 4.

6.2 Nástin instalace do letadla

Elektronické desky obou pilotů budou instalovány v zadní části příslušné schránky, jejíž součástí bude v obou případech i displej. Schránky pro Master a Slave Trim budou z konstrukčního hlediska stejné. Deska bude uchycena v prostoru schránky pomocí tyčí za čtyři otvory v rozích pomocí distančních válečků. Tyče budou vetknuty ve vnitřním povrchu schránky. Na přední straně schránky bude z její vnitřní strany přišroubovaný displej. Servo trimovací plošky bude instalováno na výškovém kormidle, jako na obrázku 6 a bude spojeno příslušnými vodiči od potenciometru, koncových spínačů a ovládání motoru s deskou Master. Tlačítka pilotů budou zapojena paralelně a připojena k desce Master Trim. Obě desky budou připojeny ke CANu letadla a zdroji 12V.

7 Závěr

V rešeršní části byl představen trim a náležitosti, které jsem využil ve svém návrhu nového trimu. Ve stati práce byla popsána sekvence vývoje nového řešení trimovacího systému pro dané uplatnění v ultralehkých letadlech, které má sedadla pilotů řazeny za sebou. V této práci byl vývoj trimu završen navržením finální verze, finální podoby v kapitole 6. Finální verze a další výstupy práce jako programy ze zkušební verze, návrh desky a serva atp. mohou posloužit jako rešeršní podklady pro další vývoj tohoto systému. Práci mohou shledat užitečnou rovněž zájemci o programování Arduina, díky podrobnému zpracování několika úloh.

Seznam zdrojů

[1] FUNDAMENTALS OF AERODYNAMICS: FUNCTIONS OF THE CONTROLS. Learn to Fly at North Las Vegas Airport [online]. Las Vegas: L. Scott Brooksby [cit. 2019-03-04]. Dostupné z: https://www.learntoflylv.com/aerodynamicspage1.htm

[2] SOKOL, FILIP. VÝVOJ PALUBNÍCH SOUSTAV DOPRAVNÍCH LETADEL [online]. Brno, 2012 [cit. 2019-03-04]. Dostupné z: https://dspace.vutbr.cz/bitstream/handle/11012/7678/Vyvoj%20palubnich%20soustav%20dopravnich%20letadel.pdf?sequence=2&isAllowed=y. Bakalářská práce. VUT FSI. Vedoucí práce Ing. Helena Trefilová.

[3] Trim tab. Wikipedia: the free encyclopedia [online]. City and County San Francisco, 2019 [cit. 2019-03-04]. Dostupné z: https://en.wikipedia.org/wiki/Trim_tab

[4] VRÁNA, Václav a Stanislav KOCMAN. Stejnosměrné stroje [online]. Ostrava, 2004 [cit. 2018-08-12]. FEI VŠB-TU Katedra obecné elektrotechniky. Dostupné z: http://fei1.vsb.cz/kat420/vyuka/Bakalarske_FS/prednasky/sylab_stejnosmerne%20stroje_bc%20FS.pdf

[5] Arduino. Wikipedia: the free encyclopedia [online]. City and County San Francisco, 2018 [cit. 2019-03-04]. Dostupné z: https://cs.wikipedia.org/wiki/Arduino

[6] H – můstky – řízení stejnosměrných motorů. Robotický tým Gymšpit[online]. 2011 [cit. 2019-03-04]. Dostupné z: http://robot.gymspit.cz/new/cz/eurobot-2010/motor-controll-2010/hbridge/

[7] PATYŠTYKA, Jan. DC motory řízené pomocí H můstku [online]. Praha, 2017 [cit. 2018-08-12]. Dostupné z: https://embedded.fel.cvut.cz/sites/default/files/kurzy/lpe/hbridge/H_Bridge.pdf. ČVUT FEL Katedra měření.

[8] DUNDÁČEK, M. Mikroprocesorový modul řízení SS motoru se zpětnou vazbou. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 69 s. [cit. 2018-04-03]. Dostupné z: https://dspace.vutbr.cz/bitstream/handle/11012/2600/DP_dokumentace.pdf?sequence=3&isAllowed=y Vedoucí diplomové práce Ing. Zdeněk Bradáč, Ph.D.

[9] POLÁK, Karel. Sběrnice CAN [online]. Brno, 2003 [cit. 2018-08-12]. Dostupné z: http://www.elektrorevue.cz/clanky/03021/index.html. Elektrorevue. VUT FEKT Ústav telekomunikací.

[10] Jazyk C (ANSI). Voho [online]. Vojta Hordějčuk [cit. 2019-03-04]. Dostupné z: http://voho.eu/wiki/c/

[11] Originál Arduino Uno Rev3. Arduino-shop [online]. Havlíčkův Brod [cit. 2019-03-04]. Dostupné z: https://arduino-shop.cz/arduino/1511-original-arduino-uno-rev3.html

Arduino DUE R3 SAM3X8E 32-bit ARM Cortex-M3. Arduino-shop[online]. Havlíčkův Brod [cit. 2019-03-04]. Dostupné z: https://arduino-shop.cz/arduino/1129-arduino-due-r3-sam3x8e-32-bit-arm-cortex-m3.html

[12] SERVOS. The Ray Allen Company [online]. Vista (USA) [cit. 2019-03-04]. Dostupné z: http://www.rayallencompany.com/products/servos.html

[13] FK-280 8000RPM 10000RPM Small Electric 6V 12V DC Motor. Alibaba[online]. Anhui Sheng Da Electronic Technology Co. [cit. 2019-03-04]. Dostupné z: https://www.alibaba.com/product-detail/FK-280-8000RPM-10000RPM-Small-Electric_60774264962.html?spm=a2700.7724857.normalList.14.3cd250935jYoHO&s=p

[14] ZELENÝ, Jiří. Stavba strojů - strojní součásti. 2. Brno: Computer Press, 2007. ISBN 80-7226-311-0.

[15] Technické plasty. Tenart [online]. Příbram [cit. 2019-03-04]. Dostupné z: http://tenart.cz/technicke-plasty/produkty/polyamid-pa-6-pa-66/

[16] DRV88703.6-A Brushed DC Motor Driver (PWM Control). Datasheet. Texas Instruments [online]. Dallas, 2019 [cit. 2019-03-04]. Dostupné z: http://www.ti.com/lit/ds/symlink/drv8870.pdf

[17] L298: Datasheet. Sparkfun [online]. 2000 [cit. 2019-03-04]. Dostupné z: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

[18] MCP2515. Datasheet. Microchip [online]. Chandler (USA): Microchip Technology, 2016 [cit. 2019-03-04]. Dostupné z: http://ww1.microchip.com/downloads/en/devicedoc/20001801h.pdf

[19] Arduino Uno R3: Datasheet. Fecegypt [online]. [cit. 2019-03-04]. Dostupné z: http://www.fecegypt.com/uploads/dataSheet/1522237550_arduino%20uno%20r3.pdf

[20] Arduino Due. Components101 [online]. 2018 [cit. 2019-03-04]. Dostupné z: https://components101.com/microcontrollers/arduino-due

[21] Arduino MEGA 2560: Pinout. Arduino [online]. [cit. 2019-03-04]. Dostupné z: https://www.arduino.cc/en/uploads/Main/arduino-mega2560_R3-sch.pdf

[22] 2.9inch e-Paper Module. Waveshare website [online]. 2018 [cit. 2019-03-04]. Dostupné z: https://www.waveshare.com/wiki/2.9inch_e-Paper_Module

[23] CANaerospace. Wikipedia: the free encyclopedia [online]. City and County San Francisco, 2018 [cit. 2019-03-04]. Dostupné z: https://en.wikipedia.org/wiki/CANaerospace

[24] ARINC. Wikipedia: the free encyclopedia [online]. City and County San Francisco, 2018 [cit. 2019-03-04]. Dostupné z: https://en.wikipedia.org/wiki/ARINC

[25] Arduino LCD KeyPad Shield (SKU: DFR0009). DFRobot [online]. 2017 [cit. 2019-03-04]. Dostupné z: https://www.dfrobot.com/wiki/index.php/Arduino_LCD_KeyPad_Shield_(SKU:_DFR0009)#Pin_Allocation

[26] HOW TO: control DC Motors with Arduino + L298N. *Youtube* [online]. LessonStudio, 7.12.2014 [cit. 2019-06-18]. Dostupné z: https://www.youtube.com/watch?v=kv-9mxVaVzE

[27] AddOhms: Electronics Tutorials For Everyone. Youtube [online]. AddOhms, 2014-2019 [cit. 2019-06-18]. Dostupné z: https://www.youtube.com/channel/UCi4UZoZM0Iw9_tTeRjZd_bA

Zdroje samostatně převzatých obrázků z webu

[O2] Trim Tabs. In: WattFlyer [online]. Florence, SC: FlyingBrick50, 2010 [cit. 2019-03-05]. Dostupné z: https://www.wattflyer.com/forums/showthread.php?t=62090, upraveno

[O5] Trim tab working principles. In: Zoombd24 [online]. [cit. 2019-03-05]. Dostupné z: http://www.zoombd24.com//wp-content/uploads/2015/03/Trim-tab-working-principles.png, upraveno

[O6] DSCN3443.JPG.4df92636d6622f58b88434e25b17e2fd.JPG. In: Avid Fox Flyers Forums [online]. 2015 [cit. 2019-03-05]. Dostupné z: http://www.avidfoxflyers.com/uploads/monthly_2017_05/DSCN3443.JPG.4df92636d6622f58b88434e25b17e2fd.JPG

[O7] Cessna 182 Skylane: Avionics for the Cessna 182. In: WIPAIRE, INC.[online]. South St. Paul, MN, 2019 [cit. 2019-03-05]. Dostupné z: https://www.wipaire.com/aircraft_page/cessna-182-skylane/, upraveno

[O8] KIT TRIM ÉLECTRIQUE RAY ALLEN T2-10A. In: GYROS EVASION[online]. [cit. 2019-03-05]. Dostupné z: https://www.gyros-evasion.com/boutiqueaero/trim-electrique/795-kit-trim-electrique-ray-allen-t2-10a.html, upraveno

[O9] Trim tab wheel. In: Steven Pam: Photoshelter [online]. [cit. 2019-03-05]. Dostupné z: https://steven-pam.photoshelter.com/image/I000041w50j7MVmwú

[O10] ACE B737 Yoke. In: FLAPS 2 APPROACH [online]. 2011 [cit. 2019-03-05]. Dostupné z: http://www.flaps2approach.com/journal/2011/12/14/ace-b737-yoke-column-review.html

[O11] Stab Trim Indicator Needle. In: FLAPS 2 APPROACH [online]. 2019 [cit. 2019-03-05]. Dostupné z: http://www.flaps2approach.com/journal/2011/12/14/ace-b737-yoke-column-review.html

[O12] H-mustek_4x_NPN. In: Robert Kocián [online]. [cit. 2019-03-05]. Dostupné z: http://www.kocian.info/kap/obrazky/H-mustek_4x_NPN.png

[O18] Originál Arduino Uno Rev3. In: Arduino-shop [online]. [cit. 2019-03-05]. Dostupné z: https://arduino-shop.cz/arduino/1511-original-arduino-uno-rev3.html

[O19] Arduino DUE. In: PTshop [online]. Meziboří [cit. 2019-03-05]. Dostupné z: https://www.ptshop.cz/Arduino-DUE-d325.htm

[O22] L298N. In: TD Egypt [online]. Egypt [cit. 2019-03-05]. Dostupné z: https://www.tdegypt.com/ar/product/%D9%85%D8%B4%D8%BA%D9%84-%D9%85%D9%88%D8%A7%D8%AA%D9%8A%D8%B1-%D8%AA%D9%8A%D8%A7%D8%B1-%D9%85%D8%B3%D8%AA%D9%85%D8%B1-%D8%AB%D9%86%D8%A7%D8%A6%D9%8A-%D8%AD%D8%AA%D9%89-2-%D8%A3%D9%85%D8%A8%D9%8A%D8%B1/

LCD Keypad Shield. In: TINYTRONICS [online]. Holansko [cit. 2019-03-05]. Dostupné z: https://www.tinytronics.nl/shop/nl/arduino/shields/lcd-keypad-shield

12V DC engine gearbox motor for garden tool KM-32A360/365. In: Shenzhen Kinmore Motor Co. Ltd [online]. [cit. 2019-03-05]. Dostupné z: https://kinmoremotor.manufacturer.globalsources.com/si/6008836310204/pdtl/Geared-motor/1155337188/12V-DC-engine-gearbox-motor.htm

Schématická značka a obrázek tlačítka. In: ARDUINO.CZ [online]. ZBYŠEK VODA, 2016 [cit. 2019-03-05]. Dostupné z: https://arduino.cz/proc-je-u-tlacitka-rezistor/

Zapojení potenciometru. In: ARDUINO.CZ [online]. Kamila Ježková, 2014 [cit. 2019-03-05]. Dostupné z: https://arduino.cz/arduino-zaklady-3-analog-read-serial/

360Ω Resistor. In: I am technical [online]. [cit. 2019-03-05]. Dostupné z: http://iamtechnical.com/360-ohm-resistor-color-code

Vše upraveno a zkompilováno

[O26] Viz zdroj [O22]

[O28] Arduino DUE R3 32 Bit ARM Development Board with USB Cable. In: Tag Detacher Co. [online]. USA, 2015 [cit. 2019-03-05]. Dostupné z: https://tagdetacher.com/product/arduino-due-r3-32-bit-arm-development-board-usb-cable/

Zapojení potenciometru. In: ARDUINO.CZ [online]. Kamila Ježková, 2014 [cit. 2019-03-05]. Dostupné z: https://arduino.cz/arduino-zaklady-3-analog-read-serial/

Vše upraveno a zkompilováno

[O30] Arduino MCP2515 CAN Bus Module TJA1050 Receiver SPI Module UK. In: Ebay [online]. 2019 [cit. 2019-03-05]. Dostupné z: https://www.ebay.co.uk/itm/Arduino-MCP2515-CAN-Bus-Module-TJA1050-Receiver-SPI-Module-UK-/201706913054

Kit Arduino Uno. In: Dificildeencontrar [online]. [cit. 2019-03-05]. Dostupné z: https://www.dificildeencontrar.com.br/arduino/kits-arduino/kit-arduino-uno/

Vše upraveno a zkompilováno

[O32] Ray Allen T2-10A. In: AIR TEAM [online]. 2019 [cit. 2019-03-05]. Dostupné z: https://www.airteam.eu/cz/p/ray-allen-t2-10a

[O39] MEGA Pinout. In: Stack Exchange [online]. Tom van der Zanden, 2018 [cit. 2019-03-05]. Dostupné z: https://3dprinting.stackexchange.com/questions/5221/how-do-i-use-mega-pin-number-designations-rather-than-ramps-pin-numbers-in-marli

Software

[S1] Vývojové prostředí Arduino 1.8.3

[S2] CAD SolidEdge v20

[S3] Microsoft Malování

[S4] CAD SolidWorks 2017

[S5] CAD KiCad 5.0.0

Seznam obrázků

Obr. 1 Osy letadla (Zdroj: Převzato z [1], upraveno) 11
Obr. 2 Trimovací plošky (Zdroj [O2], upraveno) 12
Obr. 3 Mechanický systém řízený lany (Zdroj: Převzato z [2]) 12
Obr. 5 Ovládání výškového kormidla a trimovací plošky (Zdroj [O5], upraveno) 13
Obr. 4 Mechanický systém řízení táhly (Zdroj: Převzato z [2]) 13
Obr. 6 Instalované servo (Zdroj [O6]) 14
Obr. 7 Ovládání v kokpitu (Zdroj [O7], upraveno) 15
Obr. 9 Trimovací kolo (Zdroj [O9]) 16
Obr. 10 Ovládání trimu na beranech B737 (Zdroj [O10]) 16
Obr. 11 Indikátor trimu B737 (Zdroj [O11]) 16
Obr. 8 Trimování pomocí tlačítek (Zdroj [O8]) 16
Obr. 12 H můstek (Zdroj [O12]) 18
Obr. 13 Příklad realizace fyzické vrstvy protokolu CAN (Zdroj: Převzato z [9]) 21
Obr. 14 Fyzické uspořádání sítě CAN podle ISO 11898 (Zdroj: Převzato z [9]) 21
Obr. 15 Datová zpráva podle specifikace CAN 2.0A (Zdroj: Převzato z [9]) 22
Obr. 16 Začátek datové zprávy (standardní formát) podle specifikace 2.0B (Zdroj: Převzato z [9]) 23
Obr. 17 Vývojové prostředí Arduina [S1] 24
Obr. 18 Arduino UNO (Zdroj [O18]) 25
Obr. 19 Arduino DUE (Zdroj [O19]) 26
Obr. 20 Schéma rešeršní verze [S2] 27
Obr. 21 LED indikátor polohy rešeršní verze 28
Obr. 22 Zapojení rešeršní veze [S3] (Zdroj [O22], upraveno a zkompilováno) 29
Obr. 23 Zapojení rešeršní verze ve skutečnosti 30
Obr. 24 Schéma zkušební verze [S2] 32
Obr. 25 Indikace polohy zkušební verze, dvě alternativy 33
Obr. 26 Zapojení modelu a) zkušební verze [S3] (Zdroje [O26], upraveno a zkompilováno) 35
Obr. 27 Zapojení modelu b) zkušební verze ve skutečnosti 36
Obr. 28 Zapojení modelu b) zkušební verze [S3](Zdroje [O28], upraveno a zkompilováno) 37
Obr. 29 Zapojení modelu b) zkušební verze ve skutečnosti 38
Obr. 30 Zapojení modelu c) a d) zkušební verze [S3] (Zdroje [O30], upraveno a zkompilováno) 38
Obr. 31 Zapojení modelu c) a d) zkušební verze ve skutečnosti 39
Obr. 32 Servo Ray Allen T2-10A (Zdroj [O32]) 40
Obr. 33 Převodovka serva [S4] 41
Obr. 34 Koncepční schéma převodovky serva [S2][S3] 42
Obr. 35 Navržené servo v CAD prostředí [S4] 44
Obr. 36 Navržené servo v CAD prostředí bez víka [S4] 45
Obr. 38 Rozložený pohled serva - detail převodovky [S4] 45
Obr. 39 Rozložený pohled serva [S4] 46
Obr. 39 MEGA Pinout (Zdroj [O39]) 49
Obr. 40 Napájení a ochrana [S5] 51
Obr. 41 H-můstek [S5] 51
Obr. 42 H-můstek alternativní [S5] 52
Obr. 43 CAN bus [S5] 52
Obr. 44 Navržená deska - přední strana [S5] 53
Obr. 45 Navržená deska - zadní strana [S5] 54
Obr. 46 Přehled finálního návrhu [S2] 55

Seznam tabulek

Tab. 1 Parametry navržených kol převodovky 41
Tab. 2 Seznam částí serva 47
Tab. 3 Volba vývodů procesoru 50

Seznam příloh

Příloha 1 – Program rešeršní verze

Příloha 2.a – Program zkušební verze model a)

Příloha 2.b – Program zkušební verze model b)

Příloha 2.c – Program zkušební verze model c)

Příloha 2.d – Program zkušební verze model d)

Příloha 3 – Výkres elektronického návrhu

Příloha 4 – Přehled finální verze

Příloha 1 – Program rešeršní verze

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

const int  buttonPin1 = A4;    //tlacitka ovladani motoru (jako digitalni)
const int  buttonPin2 = A5;

int buttonState1 = 0; //inicializace tlacitek
int buttonState2 = 0;

int spd = 90;  //rychlost motoru

int in1 = 13;   //H mustek
int in2 = 12;
int ENA = 11;  //-pin musi byt s PWM!

              //display
  int flag = 1; //konstanta vyuzita pro zobrazeni prvni polohy pri zapnuti, inicializace
  int dif = 0; //diference mezi lastind a ind (rozdil indikovanych poloh v case)
  int lastind = 17; //posledni indikovana poloha potenciometru (inicializace)

//analogovy vystup potenciometru zapojen do A2

void setup() {

  pinMode(buttonPin1, INPUT);  //ovladani motoru
  pinMode(buttonPin2, INPUT);

  pinMode(in1, OUTPUT);  //H mustek
  pinMode(in2, OUTPUT);
  pinMode(ENA, OUTPUT);

  digitalWrite(in1,LOW);
  digitalWrite(in2,LOW);
  analogWrite(ENA,spd);

  lcd.begin(16, 2); //uvod na lcd
  lcd.setCursor(3, 0);
  lcd.print("RUNNING...");
  delay(1200);
  lcd.setCursor(3, 2);
  lcd.print("TRIM V1.0");
  delay(3300);
  lcd.clear();

  Serial.begin(9600); //pri vyvoji programu bylo pouzito pro zobrazeni udaju do serial monitoru
}
void loop() {

  int pos = analogRead(A2); //cte potenciometr
  //Serial.println(pos);

   //tlacitka ovladani motoru, ochrana proti pretoceni v krajnich pololohach potenciometru
   buttonState1 = digitalRead(buttonPin1);
       if (buttonState1 == HIGH && buttonState2 == LOW && pos < 992) {
          //Serial.println("on1 ");

            digitalWrite(in1,HIGH);

       }
       else {
```

```
            //Serial.println("off1 ");

            digitalWrite(in1,LOW);
        }

        delay(1);
   buttonState2 = digitalRead(buttonPin2);
        if (buttonState2 == HIGH && buttonState1 == LOW && pos > 32) {
            //Serial.println("on2 ");

            digitalWrite(in2,HIGH);

        }
        else {
            //Serial.println("off2 ");

            digitalWrite(in2,LOW);
        }
        delay(1);

  lcd.setCursor(0, 1); //indikovana pos potenciometru je na lcd jako 1-16 znaků "X"
  lcd.print("UP");
  lcd.setCursor(14, 1);
  lcd.print("DN");
  lcd.setCursor(7, 1);
  lcd.print("|TO");
  lcd.setCursor(0, 0);
  lcd.print("X");

  int ind = pos/64;
  //Serial.println(ind);

 dif = ind - lastind; //poloha se na displeji preklesli pouze kdyz byla zmenena
//minimalizace problikavani LCD displeje

if (dif != 0 || flag == 1)
{
  lcd.clear();
  flag = 0;
  int i;
  lcd.setCursor(1, 0);
  for (i = 0; i < ind; i++)
   {lcd.leftToRight();
   lcd.print("X");}

  delay(1);
  lastind = ind;

}

}
```

Příloha 2.a – Program zkušební verze model a)

```
const int  buttonPin1 = A4;    //tlacitka ovladani motoru (jako digitalni)
const int  buttonPin2 = A0;

const int  endSwPin1 = A2;    //koncove spinace (jako digitalni)
const int  endSwPin2 = A5;

int buttonState1 = 0; //inicializace  tlacitek
int buttonState2 = 0;

int endSwState1 = 0; //inicializace koncovych spinacu
int endSwState2 = 0;

int spd = 90;  //rychlost motoru

int in1 = 5;   //H mustek
int in2 = 4;
int ENA = 6;  //-musi byt s PWM!

              //display
  int flag = 0; //konstanta vyuzita pro zobrazeni prvni polohy pri zapnuti

//analogovy vystup potenciometru zapojen do A3

void setup() {

  pinMode(buttonPin1, INPUT);  //ovladani motoru
  pinMode(buttonPin2, INPUT);

  pinMode(endSwPin1, INPUT);  //koncove spinace
  pinMode(endSwPin2, INPUT);

  pinMode(in1, OUTPUT);  //H mustek
  pinMode(in2, OUTPUT);
  pinMode(ENA, OUTPUT);

  digitalWrite(in1,LOW);
  digitalWrite(in2,LOW);
  analogWrite(ENA,spd);

  Serial.begin(9600); //pri vyvoji programu bylo pouzito pro zobrazeni udaju do serial monitoru
}
void loop() {

  int pos = analogRead(A3); //cte potenciometr
     buttonState1 = digitalRead(buttonPin1); //tlacitko pro pohyb do horni polohy
     buttonState2 = digitalRead(buttonPin2); //tlacitko pro pohyb do dolni polohy
     endSwState1 = digitalRead(endSwPin1); //koncovy spinac v horni poloze
     endSwState2 = digitalRead(endSwPin2); //koncovy spínac v dolni poloze

  //Serial.println(pos);
        if (buttonState2 == HIGH && buttonState1 == LOW && endSwState1 == LOW) {
          Serial.println("pohyb k dolni poloze LOW/HIGH");

          digitalWrite(in1,LOW);
          digitalWrite(in2,HIGH);

       }
       else {
           Serial.println(" /LOW");

            digitalWrite(in2,LOW);
```

```
      }


      if (buttonState1 == HIGH && buttonState2 == LOW && endSwState2 == LOW) {
          Serial.println("pohyb k horni poloze HIGH/LOW");

           digitalWrite(in2,LOW);
           digitalWrite(in1,HIGH);

      }
      else {
         Serial.println("LOW/ ");


           digitalWrite(in1,LOW);

      }


      delay(100);

}
```

Příloha 2.b – Program zkušební verze model b) bez přídavné funkce

```
#include <SPI.h>
#include <epd2in9.h>
#include <epdpaint.h>

#define COLORED     0
#define UNCOLORED   1

unsigned char image[1024];
Paint paint(image, 0, 0);    // sirka obrazku z knihovny by mela byt nasobky osmi
Epd epd;
unsigned long time_start_ms;
unsigned long time_now_s;

void setup() {
  Serial.begin(9600); //pri vyvoji programu bylo pouzito pro zobrazeni udaju do serial monitoru
  if (epd.Init(lut_full_update) != 0) {
      Serial.print("e-Paper init failed");
      return;
  }

  epd.ClearFrameMemory(0xFF);   // bit set = bila, bit reset = cerna
  epd.DisplayFrame();
  epd.ClearFrameMemory(0xFF);   // bit set = bila, bit reset = cerna
  epd.DisplayFrame();

  paint.SetRotate(ROTATE_0);

  //

  paint.SetWidth(128);
  paint.SetHeight(24);

paint.Clear(UNCOLORED);
  paint.DrawStringAt(43, 4, "TRIM", &Font16, COLORED);
```

```
  epd.SetFrameMemory(paint.GetImage(), 0, 0, paint.GetWidth(), paint.GetHeight());

  paint.Clear(UNCOLORED);
  paint.DrawStringAt(4, 4, "DN       DN", &Font16, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 5, 45, paint.GetWidth(), paint.GetHeight());

  paint.Clear(UNCOLORED);
  paint.DrawStringAt(4, 4, "TO       TO", &Font16, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 5, 145, paint.GetWidth(), paint.GetHeight());

  paint.Clear(UNCOLORED);
  paint.DrawStringAt(4, 4, "UP       UP", &Font16, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 5, 245, paint.GetWidth(), paint.GetHeight());

  paint.SetWidth(33);
  paint.SetHeight(201);

  paint.Clear(UNCOLORED);
  paint.DrawRectangle(0, 0, 32, 200, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 48, 55, paint.GetWidth(), paint.GetHeight());


 paint.Clear(UNCOLORED);
  paint.SetWidth(65);
  paint.SetHeight(1);
  paint.DrawHorizontalLine(0, 0, 65, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 39, 55, paint.GetWidth(), paint.GetHeight());

  paint.DrawHorizontalLine(0, 0, 65, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 39, 255, paint.GetWidth(), paint.GetHeight());

  paint.SetWidth(14);
  paint.SetHeight(1);

    paint.DrawHorizontalLine(0, 0, 14, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 39, 155, paint.GetWidth(), paint.GetHeight());

    paint.DrawHorizontalLine(0, 0, 14, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 87, 155, paint.GetWidth(), paint.GetHeight());
  epd.DisplayFrame();
//

  paint.SetWidth(128);
  paint.SetHeight(24);

 paint.Clear(UNCOLORED);
  paint.DrawStringAt(43, 4, "TRIM", &Font16, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 0, 0, paint.GetWidth(), paint.GetHeight());

  paint.Clear(UNCOLORED);
  paint.DrawStringAt(4, 4, "DN       DN", &Font16, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 5, 45, paint.GetWidth(), paint.GetHeight());

  paint.Clear(UNCOLORED);
  paint.DrawStringAt(4, 4, "TO       TO", &Font16, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 5, 145, paint.GetWidth(), paint.GetHeight());

  paint.Clear(UNCOLORED);
  paint.DrawStringAt(4, 4, "UP       UP", &Font16, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 5, 245, paint.GetWidth(), paint.GetHeight());

  paint.SetWidth(33);
  paint.SetHeight(201);
```

```
  paint.Clear(UNCOLORED);
  paint.DrawRectangle(0, 0, 32, 200, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 48, 55, paint.GetWidth(), paint.GetHeight());


 paint.Clear(UNCOLORED);
  paint.SetWidth(65);
  paint.SetHeight(1);
  paint.DrawHorizontalLine(0, 0, 65, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 39, 55, paint.GetWidth(), paint.GetHeight());

  paint.DrawHorizontalLine(0, 0, 65, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 39, 255, paint.GetWidth(), paint.GetHeight());

  paint.SetWidth(14);
  paint.SetHeight(1);

    paint.DrawHorizontalLine(0, 0, 14, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 39, 155, paint.GetWidth(), paint.GetHeight());

    paint.DrawHorizontalLine(0, 0, 14, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 87, 155, paint.GetWidth(), paint.GetHeight());
  epd.DisplayFrame();


  if (epd.Init(lut_partial_update) != 0) {
      Serial.print("e-Paper init failed");
      return;
  }


 // time_start_ms = millis();
}

void loop() {


  int pos = analogRead(A1); //cte potenciometr
  int vel; //velikost cerneho obdelnika
  int pol; //...jeho umístění
  vel = pos/5.141; //=1023/199 = pos/rozsah
  pol = 255 - vel;

if (pos >= 1022)
{vel = 200;
pol = 55;}

if (pos <= 2)
{vel = 1;
pol = 254;}

Serial.println(pos);


 // paint.SetRotate(ROTATE_90);
  paint.SetWidth(32);
  paint.SetHeight(199);

  paint.Clear(UNCOLORED);
  //paint.DrawRectangle(0, 0, 32, 50, COLORED);
  paint.DrawLine(0, 0, 1, 200, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 48, 56, paint.GetWidth(), paint.GetHeight());

    paint.SetWidth(32);
```

```
  paint.SetHeight(99);
  paint.DrawLine(0, 0, 1, 100, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 48, 156, paint.GetWidth(), paint.GetHeight());
 // epd.DisplayFrame();


 paint.SetWidth(32);
  paint.SetHeight(vel);

  paint.Clear(UNCOLORED);
  paint.DrawFilledRectangle(0, 0, 32, vel, COLORED);
  epd.SetFrameMemory(paint.GetImage(), 48, pol, paint.GetWidth(), paint.GetHeight());
  epd.DisplayFrame();

  paint.SetWidth(32);
  paint.SetHeight(50);


  delay(150);
}
```

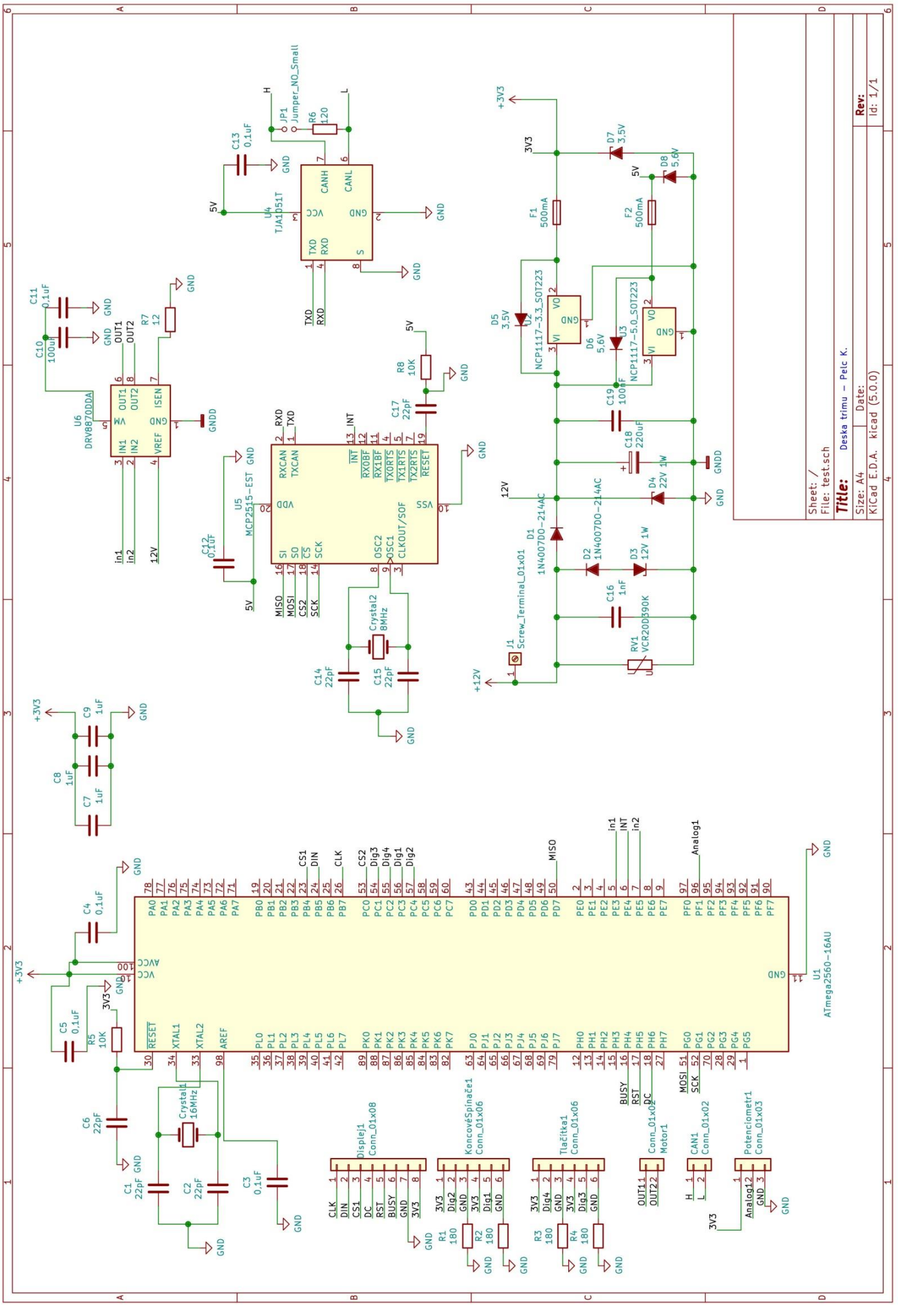
Příloha 2.c – Program zkušební verze model c)

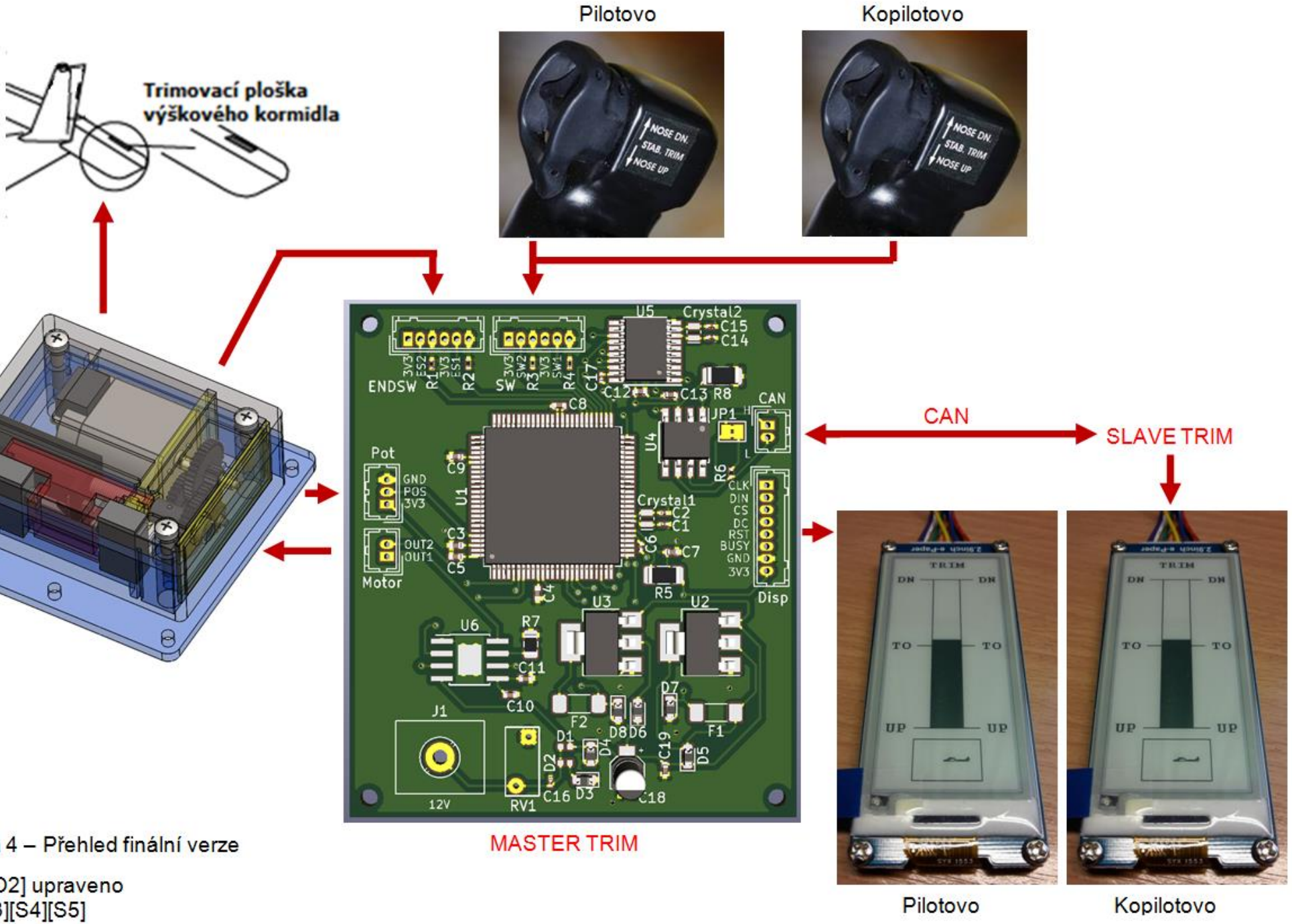
```
#include <mcp_can.h>
#include <SPI.h>

const int SPI_CS_PIN = 10;
//const int ledHIGH    = 1;
//const int ledLOW     = 0;

MCP_CAN CAN(SPI_CS_PIN);                                    // nastavit CS pin

void setup()
{
    Serial.begin(115200); //pri vyvoji programu bylo pouzito pro zobrazeni udaju do serial monitoru

    while (CAN_OK != CAN.begin(CAN_500KBPS))              // init can bus : baudrate = 500k
    {
        Serial.println("CAN BUS Shield init fail");
        Serial.println(" Init CAN BUS Shield again");
        delay(100);
    }
    Serial.println("CAN BUS Shield init ok!");
}

unsigned char stmp[8] = {199, 1, 2, 3, 4, 5, 6, 7}; //odeslani maximamalni velikosti indikace jako priklad

void loop()
{   Serial.println("In loop");
    // send data:  id = 0x70, standard frame, data len = 8, stmp: data buf
    CAN.sendMsgBuf(0x70, 0, 8, stmp);
    delay(1000);                       // send data once per second
}
```

Příloha 2.d – Program zkušební verze model d)

```
#include <SPI.h>
#include "mcp_can.h"

const int SPI_CS_PIN = 10;
int vel = 1;
//const int LED        = 8;
//boolean ledON        = 1;

MCP_CAN CAN(SPI_CS_PIN);                                    // nastavit CS pin

void setup()
{
    Serial.begin(115200); //pri vyvoji programu bylo pouzito pro zobrazeni udaju do serial monitoru
   // pinMode(LED,OUTPUT);

    while (CAN_OK != CAN.begin(CAN_500KBPS))              // init can bus : baudrate = 500k
    {
        Serial.println("CAN BUS Shield init fail");
        Serial.println("Init CAN BUS Shield again");
        delay(100);
    }
    Serial.println("CAN BUS Shield init ok!");
}


void loop()
{
    unsigned char len = 0;
    unsigned char buf[8];

    if(CAN_MSGAVAIL == CAN.checkReceive())
    {
        CAN.readMsgBuf(&len, buf);    // cte prijata data

        unsigned long canId = CAN.getCanId();

        Serial.println("-----------------------------");
        Serial.println("get data from ID: 0x");
        Serial.println(canId, HEX);

        for(int i = 0; i<len; i++)    // print the data
        {
            Serial.print(buf[i]);
            Serial.print("\t");
       vel = buf[0];
                    }
        Serial.println(vel);
    }
    delay(1000);
}
```

Příloha 3 – Výkres elektronického návrhu





Příloha 4 – Přehled finální verze

[O10][O2] upraveno
[S1][S3][S4][S5]