VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

NÁVRH 3D JOYSTICKU SE ŠESTI STUPNI VOLNOSTI
DESIGN OF 3D JOYSTICK WITH 6DOF

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Ladislav Magyerka
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Jiří Krejsa, Ph.D.
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Bc. Ladislav Magyerka**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**doc. Ing. Jiří Krejsa, Ph.D.**
Akademický rok:	2017/18

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

Návrh 3D joysticku se šesti stupni volnosti

Stručná charakteristika problematiky úkolu:

V některých aplikacích, například počítačových simulátorech, by bylo vhodné zvýšit rozsah pohybu klasické 3D myši z milimetrů na centimetry. Podstatou diplomové práce je návrh takového zařízení, tedy komplexní návrh ovladače se šesti stupni volnosti, konkrétně návrh snímačů, návrh mechanické konstrukce, návrh obslužné elektroniky a softwaru pro komunikaci s nadřazeným zařízením (PC).

Cíle diplomové práce:

1. Proveďte stručnou rešerši 3D ovládacích zařízení se shrnutím jejich parametrů, výhod a nevýhod.
2. Vyberte snímače a navrhněte mechanickou konstrukci joysticku
3. Navrhněte elektroniku pro obsluhu snímačů
4. Navrhněte obslužný SW pro komunikaci zařízení s PC

Seznam doporučené literatury:

PAWLAK, M. Andrzej: Sensors and Actuators in Mechatronics: Design and Applications, CRC Press, 2006

FRADEN Jacob: Handbook of Modern Sensors, Springer, 2010

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2017/18.

V Brně, dne 26. 10. 2017

L. S.

prof. Ing. Jindřich Petruška, CSc.
ředitel ústavu

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

ABSTRAKT

Cieľom diplomovej práce bolo navrhnúť kompletné ovládacie zariadenie so šiestimi stupňami voľnosti na použitie v počítačových simulátoroch. Po prieskume súčasných dostupných zariadení sa ďalej vytvoril model mechanizmu určený na výrobu pomocou 3D tlačiarne. Následne sa navrhla elektronika a obslužný softvér na spracovanie prijatých hodnôt zo snímačov. Nakoniec sa hotové zariadenie odskúšalo v niekoľkých aplikáciách.

Kľúčové slová

3D, joystick, šesť stupňov voľnosti, Arduino, SPI

ABSTRACT

The aim of the diploma thesis was to design a complete controller device with six degrees of freedom for use in computer simulators. After the analysis of current available devices, a model of mechanism was created for production using a 3D printer. Subsequently, electronics and utility software were designed to process received sensor readings. Finally, the finished device was tested in several applications.

Key words

3D, joystick, 6DOF, Arduino, SPI

BIBLIOGRAFICKÁ CITÁCIA

MAGYERKA, L. *Návrh 3D joysticku se šesti stupni volnosti.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2018. 68 s. Vedoucí diplomové práce doc. Ing. Jiří Krejsa, Ph.D..

ČESTNÉ PREHLÁSENIE

Prehlasujem, že som diplomovú prácu na tému **Návrh 3D joysticku so šiestimi stupňami voľnosti** vypracoval samostatne s použitím odbornej literatúry a prameňov, uvedených na zozname, ktorý tvorí prílohu tejto práce.

Dátum	Ladislav Magyerka

POĎAKOVANIE

Rád by som poďakoval vedúcemu práce doc. Ing. Jiřímu Krejsovi, Ph.D. za cenné rady a pripomienky pri vypracovaní tejto diplomovej práce. Ďalej by som poďakoval Ing. Vojtěchovi Kolomazníkovi za pomoc s elektronikou a tiež firme AMV Slovakia s.r.o. za výrobu pružín.

OBSAH

ÚVOD

Klasické 3D myši boli už zo začiatku navrhované primárne na uchopenie medzi prstami. Z toho dôvodu museli mať snímače a aj celá konštrukcia menšie rozmery a ich rozsahy pohybov v jednotlivých osiach boli iba v jednotkách milimetrov. Navyše snímače prvých zariadení už z princípu neumožňovali vôbec žiadne pohyby. Jeden z často prehliadnutých problémov je nevhodnosť použitia niektorých ovládacích zariadení na určité úlohy. Na manipuláciu virtuálnych objektov v modelovacom programe postačí aj malý ovládač, ale na pilotovanie vesmírnej lode so šiestimi stupňami voľnosti v počítačových simulátoroch je už potrebné niečo intuitívnejšie a omnoho väčších rozmerov. Značne obmedzený výber vhodných ovládacích zariadení je dostatočne silnou motiváciou na vývoj 3D joysticku na takéto účely.

Hlavnou požiadavkou v tejto práci bolo navrhnúť špeciálne ovládacie zariadenie vhodné najmä na vesmírne a letecké simulátory, v ktorých sa vyžadujú pohyby po šiestich osiach. Názov 3D joystick sa zvolil nie kvôli tvaru, ale kvôli veľkosti a účelu. Zväčšenie rozsahov natočení a posunutí na jednotky centimetrov by určite prispievalo precíznejšiemu ovládaniu a celkove aj oveľa lepšiemu zážitku.

V dnešnej dobe si človek môže vybrať z niekoľkých rôznych typov 3D ovládacích zariadení. Tie so šiestimi stupňami voľnosti sú v podstate vždy príliš malé na pohodlné uchopenie. Ďalšie, s veľkosťou porovnateľnou počítačovým myšiam, majú o jeden alebo dva stupne voľnosti menej. Po získaní prehľadu súčasného stavu na technologickom trhu sa rozhodlo vytvoriť úplne nový typ ovládacieho zariadenia. Za snímače polohy sa zvolili bezdotykové magnetické enkodéry kvôli dlhodobej životnosti elektroniky. Mechanická konštrukcia bola založená na správnom umiestnení týchto snímačov, aby ich výstupné hodnoty mohli byť veľmi rýchlo spracované. Komunikácia s počítačom sa potom vyriešila pomocou zaužívaného USB rozhrania z dôvodu potreby jednoduchého a nenáročného používania.

Počas celého návrhu sa brala do úvahy aj možnosť výroby tohto zariadenia a jej následné využitie vo zvolených aplikáciách. Model mechanizmu sa vytvoril v programe Autodesk Inventor Professional 2018 a jednotlivé súčiastky sa po ich úprave v Cura 3.0.4 vyrobili z plastu na 3D tlačiarni. Návrh dosiek plošných spojov sa realizoval pomocou programu Autodesk EAGLE 8.3.1 a za mikrokontrolér sa vybrala vývojová doska Arduino Leonardo, ktorého kód bol vypracovaný v Arduino IDE 1.8.5 v jazyku C++.

Priebeh práce na mechanizme, elektronike a programovaní sa často striedal, ale aj napriek tomu bol ich popis rozdelený na tri časti. Bez chronologického postupu popisu týchto častí nemusia niektoré zvolené metódy návrhu vždy hneď na začiatku dávať zmysel a miešanie rôznych kategórií v rozsiahlej práci by tiež nebolo veľmi prehľadné. Je to ale len malá bezvýznamná nevýhoda mechatronického postupu.

1 PREHĽAD 3D OVLÁDACÍCH ZARIADENÍ

Už niekoľko desaťročí sa ľudia snažili vytvoriť zariadenia na ovládanie priemyslových robotov a strojov, ktoré umožňujú pohyby v šiestich stupňoch voľnosti. V osemdesiatych rokoch minulého storočia bolo možné ovládať jednoduché 3D modely na počítačoch špeciálnou krabičkou s otočnými tlačidlami. Každé tlačidlo umožňovalo pohyb v jednom stupni voľnosti. Koncom sedemdesiatych rokov vedci z Nemeckého aerokozmického strediska (DLR) začali vyvíjať zariadenia na ovládanie robotických manipulátorov v trojrozmernom priestore. V roku 1981 vyvinuli šesťosový silovo-momentový snímač založený na tenzometroch, ktorý po vložení do malej dutej plastovej gule dokázal zaznamenávať posunutia a natočenia vyvolané ľudskou rukou. Toto zariadenie, ani napriek vynikajúcej aplikovateľnosti v 3D grafických systémoch a pri ovládaní robotov, sa kvôli vysokej cene (približne $8,000 za kus) neujalo na trhu. V roku 1985 sa vedcom z DLR podarilo vyvinúť lacnejší optický merací systém, ktorý využíval šesť jednorozmerných polohových snímačov. Táto technológia je aj v súčasnosti základom mnohých ovládacích zariadení so šiestimi stupňami voľnosti [1].

1.1 Typy 3D ovládacích zariadení

Vstupné zariadenia, ktoré počítaču poskytujú informácie o pohybe vo viac ako dvoch osiach, sa nazývajú 3D ovládacie zariadenia. To znamená, že klasické počítačové myši, klávesnice a gamepady do tejto kategórie nepatria, ani napriek možnosti využitia svojich tlačidiel na zjednodušené ovládanie v ďalších osiach. Príkladom je nastavenie rýchlosti pohybu vpred v leteckých simulátoroch pomocou bočných tlačidiel alebo kolieska na myši. Pri ovládaní objektov v spojitom priestore sa primárne vyžaduje analógový vstup kvôli rýchlemu získaniu požadovanej hodnoty. V tejto dobe sú skoro všetky analógové signály zo snímačov digitalizované. Práve preto sa v mnohých zariadeniach kladie veľký dôraz na rozlíšenie pri prevode analógového signálu na diskrétne hodnoty. Pri dostatočne veľkom rozlíšení sa vstup považuje za spojitý.

Snímače sú najvýznamnejšou časťou ovládacích zariadení. Pri návrhu viacosových zariadení je možné zvoliť jeden z troch typov snímačov pre každý stupeň voľnosti. Izotonické snímače merajú dráhu pohybu a musí na ne byť vynaložená konštantná sila, ktorá ale môže byť veľmi malá. Príkladom je počítačová myš alebo rotačný potenciometer. Elastické snímače dovolia obmedzenú odchýlku od základnej polohy a pôsobia silou proti pohybu podobne ako pružiny. Tretím typom sú izometrické snímače, ktoré zaznamenávajú sily a momenty, najčastejšie z tenzometrov. Tie sa následne prepočítavajú na posuny a natočenia využiteľné v programoch. Niektoré snímače poskytujú hodnoty z viacerých stupňov voľnosti. Jedným príkladom je optický snímač na myši, ktorý udáva zmenu polohy na dvoch osiach. Pri návrhu ovládacieho zariadenia sa tieto typy často rôzne kombinujú. Týmto spôsobom je ich možné vytvoriť na špecifické účely [2].

Dôležitým cieľom pri návrhu konštrukcie je správna voľba oddeliteľnosti a celistvosti stupňov voľnosti. Zariadenia, ktoré disponujú len jedným miestom uchopenia, majú s veľkou pravdepodobnosťou všetky stupne voľnosti zjednotené. To so sebou prináša vedľajšie účinky v podobe miernych nechcených translačných pohybov počas zámerných rotačných a naopak. Tento problém je možné čiastočne vyriešiť implementáciou takzvanej mŕtvej zóny v okolí základnej polohy, čím sa ale zníži užitočné rozlíšenie snímačov a zvýši minimálna potrebná odchýlka na zaznamenanie pohybu. Druhým spôsobom je oddelenie rotačnej a translačnej časti tak, aby sa navzájom neovplyvňovali [3].

V ďalších častiach sa popíšu jednotlivé typy 3D ovládacích zariadení. Niektoré z nich ale budú vynechané, pretože ich princíp snímania polohy je od základov úplne odlišný od predpokladaného výstupu tejto práce. Medzi tieto zariadenia (pozri Obr. 1.1) patria rôzne príslušenstvá pre systémy virtuálnej reality, ako napríklad ovládače pre Oculus Rift [4] alebo HTC Vive [5], ktoré využívajú sledovanie polohy pomocou niekoľkých kamier alebo infračervených LED svetiel.

Obr. 1.1: Oculus Touch [4] a HTC Vive Controller [5].

1.2 Joysticky

Väčšinu joystickov sa dá ovládať len na dvoch osiach, a to naklonením páky do štyroch smerov, čo umožňuje dvojica potenciometrov, ako je vidno na Obr. 1.2. V kvalitnejších modeloch sa využívajú Hallové snímače. Joysticky poskytujú intuitívne ovládanie nielen v leteckých simulátoroch, ale aj v civilných a vojenských lietadlách a helikoptérach. Pohyby s pákou napodobňujú požadované pohyby ovládaného stroja, čo mnoho ľudí uprednostňuje pred použitím myši a klávesnice. Herné joysticky takmer vždy obsahujú aj množstvo tlačidiel a často aj viacsmerový prepínač (hat switch) používaný väčšinou v simulátoroch na zmenu uhlu pohľadu v kabíne.

Obr. 1.2: Dvojosový [6] a trojosový joystick [7].

Existujú však aj joysticky, ktorých otočná páka umožňuje pohyb v troch osiach. Jedným z najznámejších je Extreme 3D Pro Joystick [8], znázornený na Obr. 1.3. Obsahuje až 12 programovateľných tlačidiel a osemsmerový prepínač. Najdôležitejším parametrom ale je už zmienený tretí stupeň voľnosti, ktorý analógovým vstupom nahradí dve tlačidlá

na klávesnici (Q a E), a tým ich sprístupní pre iné príkazy. Vo vesmírnych simulátoroch je tento dodatočný rotačný pohyb nesmierne výhodný, keďže zatáčanie doprava a doľava je oveľa častejšie využívaný manéver ako rotácia okolo osi v smere pohybu vpred.

Obr. 1.3: Extreme 3D Pro Joystick [8].

Omnoho sofistikovanejšie joysticky, ako napríklad X56 H.O.T.A.S. [9] (pozri Obr. 1.4), sa skladajú z dvoch častí a disponujú oveľa vyšším počtom programovateľných tlačidiel a prepínačov. Navyše vďaka množstvu miniatúrnych analógových pák pridávajú ďalšie ovládateľné stupne voľnosti. Takéto zariadenia sú vyvinuté predovšetkým na letecké a vesmírne simulátory, ale ich cena je značne vyššia oproti klasickým joystickom.

Obr. 1.4: X56 H.O.T.A.S. [9].

1.3 3D myši

Do tejto podkapitoly sú zaradené len tie zariadenia, ktoré výrobca nazval ako 3D myš, keďže existujú aj iné ovládače s podobnými funkciami a vlastnosťami vyvinuté na špeciálne účely. Nižšie je popísaných len šesť z mnohých rôznych modelov.

Prvá komerčne úspešná 3D myš bola v uvedená do predaja v roku 1988 pod menom SpaceBall 1003 [10], ktorú je vidno na Obr. 1.5. Jej pôvodná cena sa pohybovala okolo $3,200 za kus. Obsahuje 9 tlačidiel a jej silovo-momentové snímače sú vstavané v dutej plastovej guli. Komunikácia s počítačom sa realizuje pomocou sériového portu.

Obr. 1.5: SpaceBall 1003 [10].

Jedna z prvých 3D myší navrhnutá na hry bola SpaceOrb 360 [11] zobrazená na Obr. 1.6. Vydaná v roku 1996 za cenu približne $100, táto myš má 6 programovateľných tlačidiel, ale ešte stále používa rovnaký typ snímačov, ako jej predchodcovia. Sériová komunikácia je však mierne odlišná od ostatných modelov.

Obr. 1.6: SpaceOrb 360 [11].

V roku 2003 bola vydaná prvá 3D myš s optickými snímačmi namiesto silovo-momentových s názvom SpaceBall 5000 [12] (pozri Obr. 1.7) za pôvodnú cenu okolo $500. Obsahuje 12 programovateľných tlačidiel a oddeliteľnú opierku zápästia, ktorú je možné nasadiť na pravú alebo ľavú stranu podľa preferencie užívateľa. Novšie modely podporujú už aj komunikáciu s počítačom cez USB.

Obr. 1.7: SpaceBall 5000 [12].

V súčasnosti je najznámejšou 3D myšou SpaceMouse Compact [13] zobrazenou na Obr. 1.8, ktorá nahradila starší model SpaceNavigator z roku 2006. Od svojho predchodcu sa líši hlavne upraveným dizajnom a lepšou distribúciou váhy. Jej cena sa pohybuje okolo $129. Obsahuje optické snímače a 2 programovateľné tlačidlá umiestnené na bokoch. Navrhnutá bola primárne na manipuláciu 3D modelov v CAD aplikáciách, použiteľná je ale aj v rôznych simulátoroch a hrách. Pokročilejšie verzie tohto zariadenia s vyšším počtom tlačidiel a operadlom na zápästie sú tiež dostupné, niektoré z nich dokonca aj v bezdrôtovom prevedení.

Obr. 1.8: SpaceMouse Compact [13].

SpaceController [14] je ďalšia 3D myš založená na rovnakej technológii ako zariadenia zo série SpaceMouse. Hlavným rozdielom je možnosť výberu prevedenia rukoväte medzi pukom a guľou. Obsahuje 17 programovateľných tlačidiel a LCD displej, ktorým je možné meniť nastavenia. Vydaná bola v roku 2009 za pôvodnú cenu $319. Novšie modely od roku 2011 majú pod rukoväťou dodatočné ovládacie koliesko na zmenu citlivosti.

Obr. 1.9: SpaceController [14].

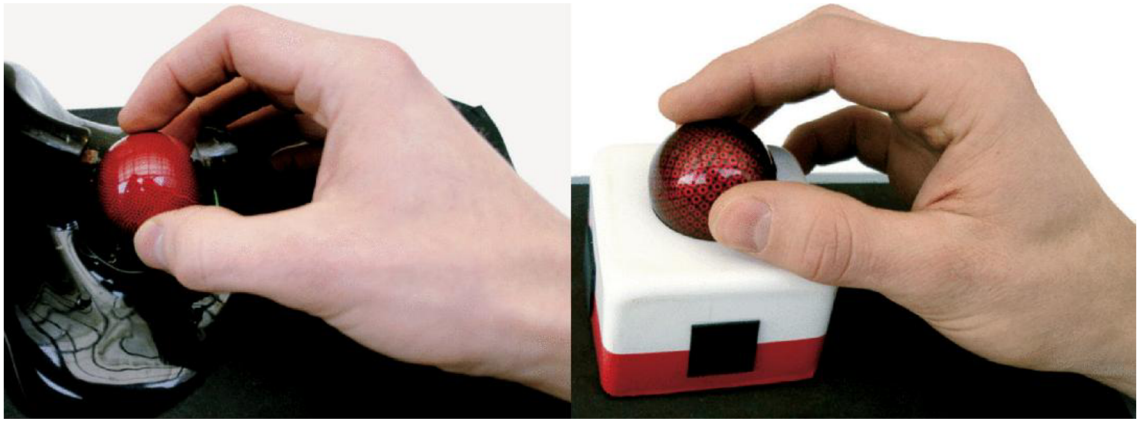
Lexip 3D Pro [15] (pozri Obr. 1.10) je zariadenie vyvinuté predovšetkým na hry, ktoré zlučuje hernú počítačovú myš a dva miniatúrne dvojosové analógové joysticky. Vonkajší joystick sa nachádza na ľavej strane pod palcom a vnútorný sa ovláda naklonením krytu myši dopredu, dozadu a do bokov, čím sa zaistí pohyb v šiestich stupňoch voľnosti. Navyše všetkých 7 programovateľných tlačidiel je dostupných bez nutnosti pustenia myši a následnej straty kontroly, ako je to pri ostatných zariadeniach.

Obr. 1.10: Lexip 3D Pro [16].

1.4 Viacosové ovládacie zariadenia

V tejto podkapitole sú popísané rôzne viacosové ovládacie zariadenia, ktoré výrobcovia neklasifikujú ako 3D myši. Väčšinou majú buď netradičné tvary, alebo nie sú príliš vhodné na náhradu myši. Zaradené sú sem aj prototypy a experimentálne zariadenia vyrobené na vedecké účely.

Pri práci s ovládacími zariadeniami v CAD aplikáciách sa translačné a rotačné pohyby najčastejšie vykonávajú sekvenčne, nie súčasne. 3D model sa najprv natočí do správneho uhlu pohľadu, potom sa až priblíži a vycentruje. Preto je veľmi výhodné použiť trackball s tromi stupňami voľnosti, ktorý zaisťuje rotáciu, pripevnený k elastickému rámu, ktorý povoľuje len translačný pohyb. Týmto spôsobom sa dá vyhnúť mnohým ťažkostiam pri manipulácii s modelmi. Nechcené translačné pohyby počas rotácie nenastávajú. Nesmie sa však použiť nadmerná sila na trackball, keďže jeho natočenie spravidla vyžaduje veľmi malú silu. Dva takéto prototypy, GlobeFish a GlobeMouse na Obr. 1.11, boli porovnané s komerčne dostupnou 3D myšou SpaceMouse pri práci v CAD aplikácii. Väčšina užívateľov uprednostňovala tieto prototypy, pravdepodobne kvôli oddeleným trojiciam stupňov voľnosti [2].



Obr. 1.11: GlobeFish a GlobeMouse [2].

Jedno podobné zariadenie, ktoré využíva trojosový trackball a elastický rám, bolo síce vyvinuté, ale nikdy sa nedostalo do masovej výroby. 3D-Spheric-Mouse [17], ktorý je zobrazený na Obr. 1.12, zachytáva rotáciu optickými snímačmi a posunutie magnetickými. Navrhnutý bol primárne pre digitálnych umelcov pracujúcich s 3D modelmi.

Obr. 1.12: 3D-Spheric-Mouse [17].

Tesseract [18] (pozri Obr. 1.13) je 3D ovládacie zariadenie schopné pohybu v piatich osiach s rozsahom ± 5 mm od stredu. Nadvihnutie a zatlačenie v osi Z síce neumožňuje, ale koliesko myši sa môže považovať za zjednodušenú náhradu. Obsahuje až 8 programovateľných tlačidiel a pomocou ovládacieho panela je možné zmeniť funkciu každej osi z emulácie joysticku na klávesové skratky stlačené so zvolenou frekvenciou. Navrhnutý bol najmä na počítačové hry, a to hlavne simulátory, ale veľmi efektívne zvláda aj závodné a FPS hry. Vhodný je aj na prácu v CAD aplikáciách a na malé diaľkovo ovládané helikoptéry. Toto zariadenie sa do masovej výroby nikdy nedostalo.

Obr. 1.13: Tesseract [18].

Zariadenie Wing [19], ktoré je zobrazené na Obr. 1.14, bolo vyvinuté špeciálne na riadenie malých diaľkovo ovládaných helikoptér a lietadiel. Citlivý laserový snímač na spodnej strane spolu s vnútorným štvorosovým joystickom umožňuje pohyb v šiestich stupňoch voľnosti. Jedným nedostatkom je ale to, že pohyb v osi Z je len jednosmerný. To znamená, že celú vrchnú časť sa síce dá zatlačiť smerom dole a po pustení ju pružina vráti naspäť, ale nadvihnutie tejto časti zo základnej polohy je už nemožné. Preto sa táto os najčastejšie využíva na prepínanie režimov kolieska medzi normálnym rolovaním a ťahom motorov ovládaného vozidla. Výroba zariadenia je len na objednávku za cenu £825.

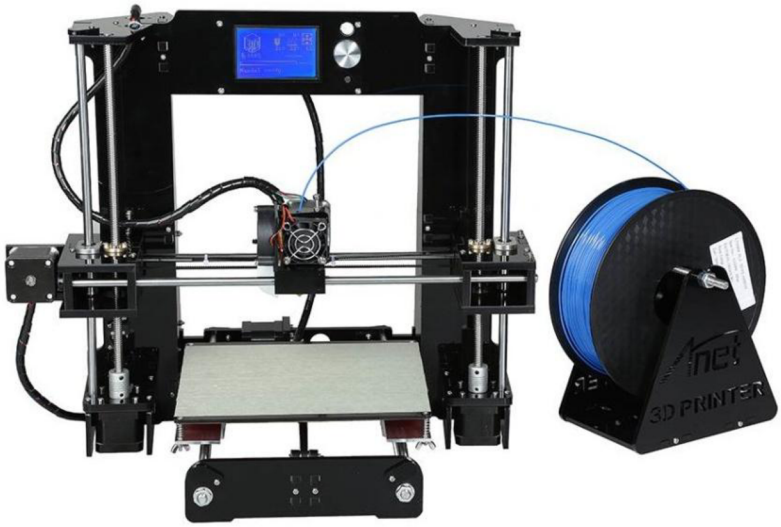
Obr. 1.14: Wing [20].

2 NÁVRH MECHANICKEJ KONŠTRUKCIE

Na začiatku bolo v pláne navrhnúť model mechanizmu schopného pohybu v šiestich stupňoch voľnosti v rozsahu niekoľkých centimetrov. To sa však po mnohých skúškach s plastovými lineárnymi ložiskami vyrobenými na 3D tlačiarni javilo príliš problematické. Pri veľkej vzdialenosti klzného ložiska od pôsobiska sily, napríklad rukoväte, sa v mieste pohyblivého kontaktu dvoch kusov plastu vytvárajú nadmerné trecie sily, ktoré spôsobujú zasekávanie mechanizmu. Zvýšením rozsahu pohybov sa taktiež zmení veľkosť celého zariadenia v oveľa väčšom pomere a použitie niektorých komponentov môže byť veľmi nepraktické a v niektorých prípadoch aj nemožné. Navyše pri väčších rozmeroch rukoväte by každý pohyb ruky značne ovplyvňoval ako rotačné, tak aj posuvné osi zariadenia. Pôvodný plán sa preto musel trochu zmeniť a model mechanizmu sa rozdelil na dve časti. Rotačná časť ostala prevažne bez zmeny, pridala sa len samostatná podstava. Lineárna časť bola upravená tak, aby sa čo najviac zmenšilo trenie v klzných ložiskách kvôli príliš veľkej vzdialenosti pôsobiska sily od kontaktnej plochy. Výhodou tejto zmeny je okrem minimalizácie počtu nechcených pohybov aj možnosť oddelenej manipulácie s dvomi rozdielnymi druhmi pohybu. Jedinou závažnou nevýhodou je potreba využitia obidvoch rúk na ovládanie v šiestich stupňoch voľnosti, keďže v niektorých aplikáciách sa tiež vyžaduje používanie klasickej počítačovej myši alebo klávesnice.

2.1 Spôsob výroby mechanizmu

Na výrobu všetkých plastových častí mechanizmu sa používala jednoduchá 3D tlačiareň nižšej kategórie Anet A6 [21], znázornenú na Obr. 2.1, s priemerom trysky 0,4 mm. Je to v podstate modifikovaná verzia veľmi populárnej 3D tlačiarne Prusa i3. Za materiály sa používali dva typy plastu, a to svetlomodré PLA a čierne ABS. S plastom PLA sa pracuje omnoho lepšie, pretože má nižšiu teplotu topenia ako ABS. Navyše PLA zvyčajne nemá problémy s prilepením na pracovnú plošinu ani pri izbovej teplote. ABS potrebuje mať plošinu vyhriatu ideálne na 105 °C, čo je pri otvorených 3D tlačiarňach, ako aj pri Anet A6, často nedosiahnuteľné. Pri výrobe mechanizmu sa teploty plošiny nastavili na 60 °C pre PLA a 80 °C pre ABS. Súčiastky z PLA sú pevnejšie, ale krehkejšie ako z ABS, ktoré zase zvláda väčšie ohyby pred lomom [22].



Obr. 2.1: Anet A6 [21].

ABS plasty je tiež možné naleptať v rôznych chemikáliách. Najpoužívanejšou je z nich acetón, ktorý pri vyparovaní v uzavretej nádobe vyhladí povrch plastu, čím sa uzatvoria mikroskopické póry a vcelku sa zlepší vzhľad materiálu. Najväčším problémom s ABS je ten, že počas chladnutia po výrobe sa všetky súčiastky značne deformujú najmä na okrajoch, čomu nie je také jednoduché zamedziť. Znížiť mieru tejto deformácie je možné zaistením kvalitnej adhézie plastu o povrch maskovacej pásky na pracovnej plošine a zvýšením jej teploty, čím sa spomalí celý proces chladnutia [23,24].

Keďže sa jedná o veľmi jednoduchú cenovo dostupnú 3D tlačiareň, kvalita jej súčiastok je nízka. To so sebou prináša mnohé problémy, ktoré sa dajú len čiastočne vyriešiť. Málo ohybné plastové hnacie remene v sebe majú určitú vôľu a zapríčiňujú príliš veľké výchylky pri prudkej zmene smeru pohybu. Na vyrobených predmetoch sa to potom prejavuje v lepšom prípade len zvlnením na okrajoch a v horšom prípade odchýlkou rozmerov až niekoľko desatín milimetra na veľmi krátkych úsekoch. Tepelná deformácia taktiež vedie k nepresným rozmerom v hotových súčiastkach. Tie však je možné niekedy aj predpovedať a veľkosti pred výrobou upraviť, ale vo väčšine prípadov meniť netreba nič. Výnimkou sú funkčné rozmery, kde jednu súčiastku treba vložiť do druhej, alebo kde dochádza k vzájomnému pohybu. Najčastejšie sa ale upravujú rozmery dier pre skrutky, kde sa odporúča k priemeru pridať približne 0,3 až 0,4 milimetrov pri použití trysky 3D tlačiarne s priemerom 0,4 mm, aby nedošlo k poškodeniu súčiastky alebo k oddeleniu vrstiev. Ak sa jedná o pohyblivý kontakt dvoch alebo viacerých častí s čo najmenšou vôľou medzi nimi, je takmer vždy očakávané, že sa budú musieť vyrobené kusy premerať a rozmery následne modifikovať. Tento proces sa potom niekoľkokrát môže opakovať, čím sa spotrebuje veľké množstvo plastu.

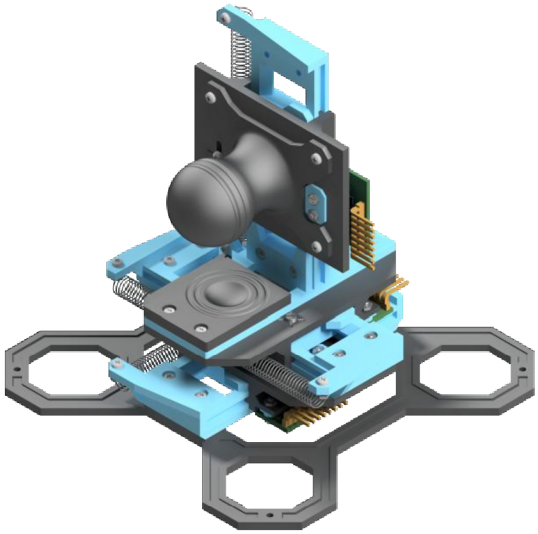
Jedným spôsobom, akým sa dá zaistiť presnú vzdialenosť medzi dvomi nepresnými súčiastkami pomocou skrutiek, je náhrada okrúhlych dier drážkami s dostatočnou dĺžkou. Tým sa umožní ručné nastavenie vzájomnej polohy bez nutnosti výroby nového kusu alebo zdĺhavého a deštruktívneho opracovania plastu, často aj na ťažko dostupných miestach. V ďalších dvoch podkapitolách bude ukázaná výhoda tohto spôsobu na konkrétnych súčiastkach.

V tejto práci bolo vynechané počítanie napätí v materiáli z dvoch dôvodov. Prvým je nenáročné využitie hotového zariadenia, kde sily pôsobiace na mechanizmus sú neporovnateľne nižšie oproti maximálnemu povolenému zaťaženiu daného materiálu. Druhým dôvodom je to, že parametre prakticky každej vyrobenej súčiastky, aj keď sú to len kópie, sú vždy odlišné kvôli už zmieneným nepresnostiam. Navyše pri prevedení modelu na inštrukcie potrebné na výrobu pre 3D tlačiareň sa rozmery automaticky upravia a niektoré malé časti kompletne vynechajú. Najväčší vplyv na to má hlavne výplň materiálu, ktorú sa dá zmeniť na požadovaný typ a hustotu alebo rovno všetko vynechať, čím sa hmotnosti modelu a hotovej súčiastky môžu výrazne líšiť. Z toľkého množstva neznámych parametrov by nebolo možné dostatočne presne zistiť akékoľvek užitočné vlastnosti ktoréhokoľvek vyrobeného kusu. Pri návrhu modelu preto treba použiť vždy viac materiálu, aby sa zaistila dostatočná pevnosť a bezpečnosť aj napriek nepresnostiam a častým poruchám v štruktúre vyrobených kusov. Jediným efektívnym spôsobom na odhad vhodnosti súčiastky na danú funkciu je nadmerné ručné zaťažovanie a následná ostrá skúška. Z toho plynie výhoda použitia 3D tlačiarne na prototypovanie. Je veľmi jednoduché na modeli vykonať mierne úpravy a nechať nový kus vyrobiť v priebehu niekoľkých minút až hodín.

2.2 Lineárna časť

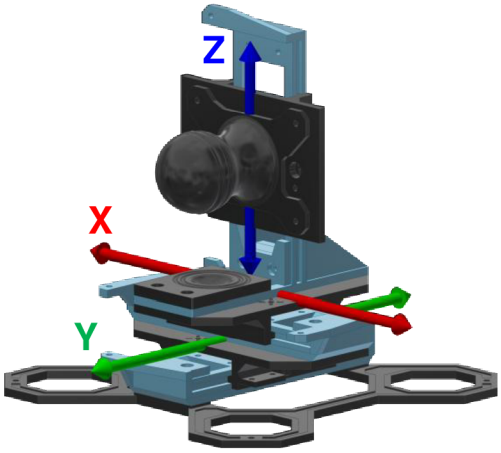
Pred návrhom mechanizmu sa najprv museli zvoliť snímače a vhodná riadiaca jednotka. Bližšie budú popísané v nasledujúcej kapitole. Keďže zvolený mikrokontrolér nedisponuje špeciálnym hardvérom na prácu so zlomkovými číslami, použitie goniometrických funkcií na výpočet polohy pri rôznych kĺbových mechanizmoch by bolo veľmi neefektívne. Zlomkový počet je mnohonásobne pomalší ako celočíselný. V goniometrických funkciách navyše prebieha až niekoľko zlomkových výpočtov, čo môže viesť ku stovkám až tisíckam inštrukčných cyklov na získanie polohy len v jednej osi. Existujú aj alternatívne metódy na zvýšenie rýchlosti, ale skoro všetky sú na úkor presnosti. Z toho plynie, že najvhodnejším spôsobom zachytenia posuvného pohybu je pomocou troch klzných lineárnych ložísk, kde je možné získať polohu na jednej osi pomocou jediného snímača bez akéhokoľvek prepočtu hodnôt. Tým sa zaistí veľmi rýchla odozva, ktorou sa ale presnosť hodnôt nijak nezníži.

Na Obr. 2.2 je zobrazený hotový model lineárnej časti mechanizmu bez kabeláže. Príloha č. 2 obsahuje túto zostavu v rozloženom pohľade. Podstava má na okrajoch štyri diery na jednoduchú montáž pomocou skrutiek M2 kvôli veľmi malej celkovej váhe. So zariadením sa manipuluje pomocou hornej veľkej rukoväte. Prstami sa chytí guľatá časť s priemerom 3 cm, ktorou sa potom voľne posúva v troch stupňoch voľnosti. Rozsah pohybu každej osi je ± 2 cm a ich základnú polohu udržiavajú dvojice malých veľmi slabých ťažných pružín. Ak v niektorých aplikáciách, napríklad v závodných hrách, nie je nutné využitie osi Z, v strednej časti mechanizmu sa taktiež nachádza malá druhotná rukoväť, ktorá slúži na pohyb len v osiach X a Y iba pomocou jediného prstu. Obidve rukoväte na sebe majú malé zárezy, ktoré pomáhajú s ich udržiavaním a zabraňujú nadmernému šmýkaniu.



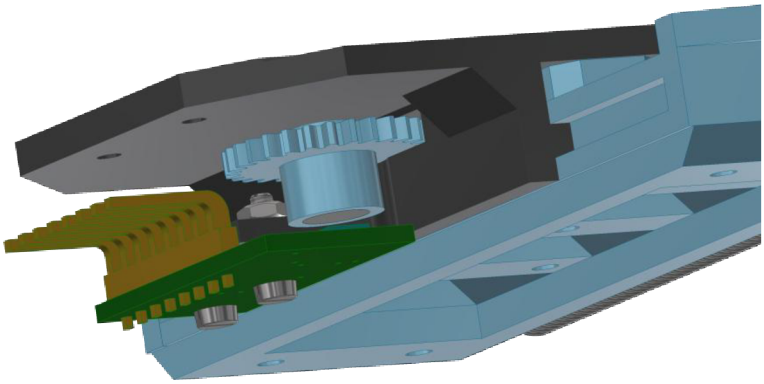
Obr. 2.2: Model lineárnej časti mechanizmu.

Priamo nad podstavou sú umiestnené tri klzné lineárne ložiská prepojené v sérii od osi Y cez X po Z, ako je znázornené na Obr. 2.3. Ich smery sa zvolili podľa rozloženia na 3D tlačiarňach. Jedná sa teda o karteziánsku súradnicovú sústavu, kde jednotlivé osi sú na seba vzájomne kolmé. Podstava je pevne spojená s vedením ložiska Y, na ktorom sa pohybuje príslušný vozík. Na tomto vozíku je pripevnené vedenie ložiska X kolmo a horizontálne na Y. Ich usporiadanie je v podstate rovnaké. Vozík X má na sebe umiestnené vedenie ložiska Z kolmo na X a aj Y, čiže vertikálne.



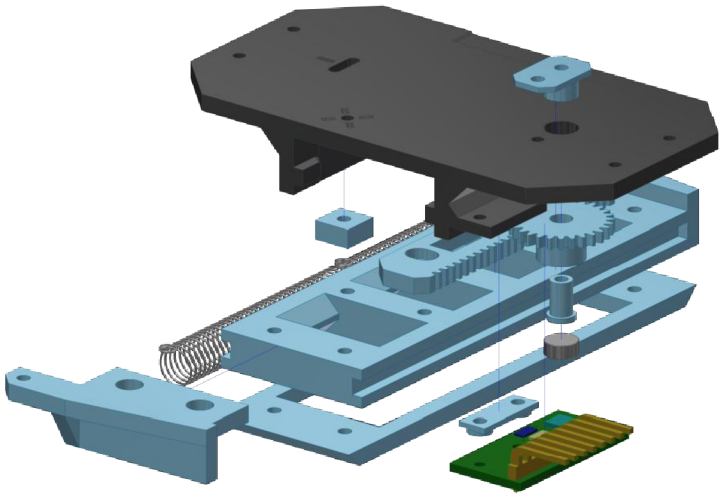
Obr. 2.3: Osi posuvného pohybu.

Na Obr. 2.4 je zobrazené jedno izolované ložisko spolu so snímačom a magnetom. Na modrom vedení z plastu PLA sa posúva čierny vozík z ABS. Z predbežných pokusov sa zistilo, že použitie rozdielnych materiálov pomáha zaistiť plynulejší pohyb. Ich kontaktné plochy boli pred montážou nanesené suchým teflonovým mazivom kvôli zmenšeniu trecích síl. Keďže tieto ložiská nie sú uzavreté, prašné prostredie by mohlo negatívne ovplyvniť akékoľvek vlhké mazivo. Na spodnej strane vedenia sa nachádza obdĺžniková príruba, ktorá má za úlohu oddeliť jednotlivé ložiská. Keby bolo nutné zmeniť vzdialenosť dvoch ložísk, stačí upraviť výšku príruby bez potreby zdĺhavej výroby celého nového vedenia. DPS snímača je pripevnená k pravej spodnej strane vozíka cez podložku, ktorej výšku je možné jednoducho upraviť odobraním malého množstva materiálu. Tesne nad snímačom je prichytené ozubené koleso s magnetom.



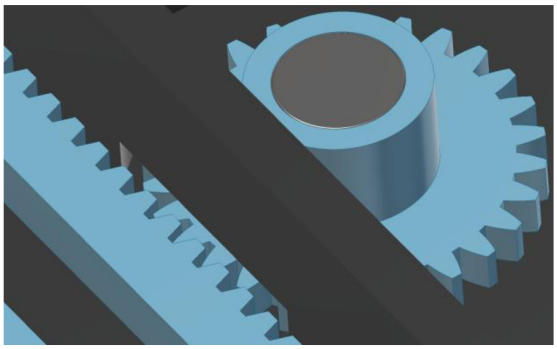
Obr. 2.4: Umiestnenie snímača na lineárnej časti.

Celá zostava ložiska so snímačom je v rozloženom pohľade zobrazená na Obr. 2.5. Pri montáži ložiska sa vozík nasunie do drážky vedenia, na ktorého konci sa pripevnením plastovej dosky zabráni vypadnutiu tohto vozíka. Táto malá doska taktiež umožňuje uchytenie dvoch pružín na ľavej strane vedenia. Na pravej strane vozíka je miesto na pripevnenie držiaka ozubeného kolesa. Spolu s ozubeným hrebeňom na pravej strane vedenia, lineárny pohyb vozíka zapríčiňuje rotačný pohyb ozubeného kolesa, a teda aj magnetu, čím sa získava hodnota posunutia na danej osi.



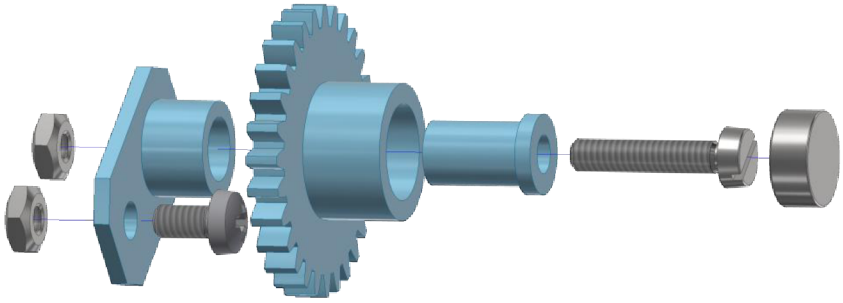
Obr. 2.5: Zostava ložiska na lineárnej časti v rozloženom pohľade.

Ozubený hrebeň sa kolesa dotýka cez otvor v strede na pravej strane vozíka, ako je vidno na Obr. 2.6. Samotný hrebeň v sebe má namiesto jednoduchých dier na skrutky dve pozdĺžne drážky, ktoré uvoľnením skrutiek, posunutím a ich následným pritiahnutím umožňujú meniť osovú vzdialenosť. Drážky sú tu nevyhnutné kvôli nadmernej deformácii plastu na zuboch.



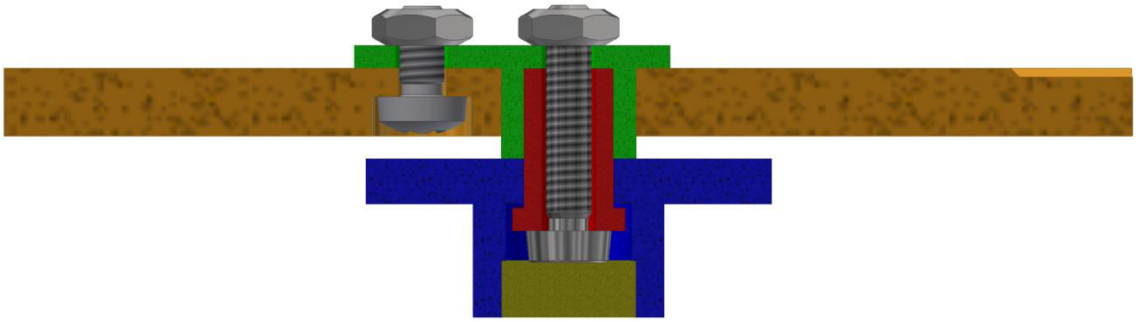
Obr. 2.6: Ozubené súkolesie.

Z Obr. 2.7 je možné pozorovať zostavu držiaka magnetu zloženú z jednotlivých súčiastok. Do ozubeného kolesa sa vloží dutý valcový kolík, ktorý sa zasunie do jeho držiaka namontovaného na vozíku. Potom sa skrutkou M2 tieto dve časti pevne spoja na druhej strane. Táto skrutka musela mať na rozdiel od ostatných valcovú hlavu s plochým vrchom, pretože zaberala menej miesta pod magnetom, ktorý má byť tesne nad ňou zasunutý do diery na hornej časti ozubeného kolesa. Zúženie vnútorného priemeru o štyri desatiny milimetra stačí na usadenie magnetu v diere na doraz.



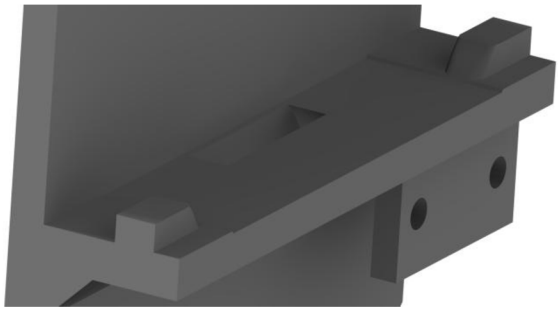
Obr. 2.7: Zostava držiaka magnetu v rozloženom pohľade.

Na Obr. 2.8 je znázornený prierez už zloženej zostavy z predošlého obrázku. Kvôli lepšej viditeľnosti sú súčiastky odlišne zafarbené. Niektoré kritické rozmery boli ešte pred výrobou upravené, aby sa zaistila správna funkčnosť zostavy, takže pretínanie magnetu a skrutky na obrázku v skutočnosti nenastalo. Navyše po zatiahnutí skrutky, ktorej hlava mala o niečo menšie rozmery ako v modeli, sa plast tiež mierne stiahol. Oranžová časť na obrázku je vozík, modrá je ozubené koleso, zelená jeho držiak, červená je kolík a žltá magnet.



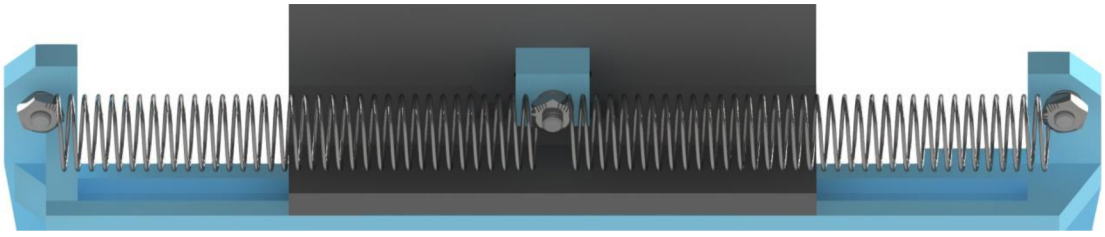
Obr. 2.8: Prierez držiaka magnetu.

Aby sa vozík mohol v drážke pohybovať s malým trením, profil jeho kontaktnej plochy s vedením sa musel modifikovať podľa Obr. 2.9. Z celej plochy sa okrem piatich milimetrov od okrajov ubral materiál hrúbky štyroch desatín milimetra spolu aj s lištami na tomto mieste. Ostali len dve kratšie lišty na každej strane, čím sa nesmierne uľahčil pohyb ložiska. Počas výroby plastových kusov na 3D tlačiarni sa veľmi často stáva, že sa na stenách v niektorých vrstvách vytvorí prírastok materiálu kvôli rozdielnym trasám trysky a následnej odchýlky požadovanej polohy pred nábehom na daný rovný úsek. Môže to byť spôsobené vôľou v hnacích remeňoch, ako už bolo popísané v predošlej podkapitole.



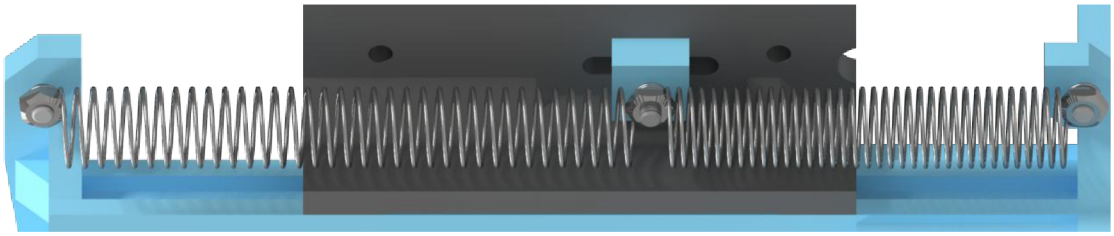
Obr. 2.9: Profil vozíka na mieste kontaktu s vedením.

Na ľavej strane ložiska na osiach X a Y sa nachádzajú dve identické ťažné pružiny (pozri Obr. 2.10) pripevnené k vedeniu. Vozík sa k nim pripája cez drážku, ktorá umožňuje posunutie skrutky po osi pružín v rozsahu 5 mm, čím sa dá ručne nastaviť základná poloha v prípade menších rozdielov v parametroch pružín.



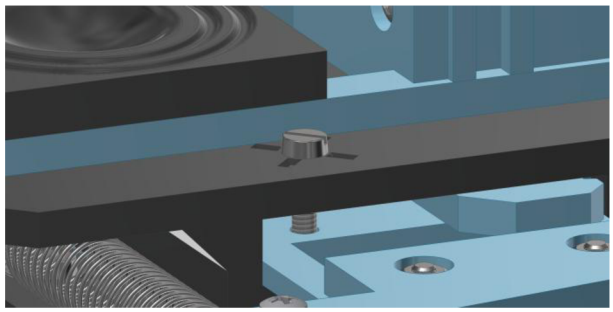
Obr. 2.10: Ťažné pružiny na osiach X a Y.

Pružiny v osi Z sú rovnaké ako v osiach X a Y, akurát je miesto ich pripevnenia na vozíku posunuté nižšie, aby sa sila hornej pružiny vyrovnala s gravitačnou silou vozíka a rukoväte. Drážka má v tomto prípade rozsah 10 mm a je posunutá o 6,4 mm smerom dole, čo na Obr. 2.11 znamená vpravo.



Obr. 2.11: Ťažné pružiny na osi Z.

Keby nastal prípad, v ktorom pohyby v niektorých osiach nie sú potrebné, vložením skrutky M2 do diery označenej písmenom X v základnej polohe je možné celé ložisko zablokovať. Skrutka, ktorá je znázornená uprostred Obr. 2.12, sa týmto spôsobom využíva viac-menej len ako blokovací kolík. Vyrobiť špeciálnu súčiastku z plastu vôbec nebolo nutné.



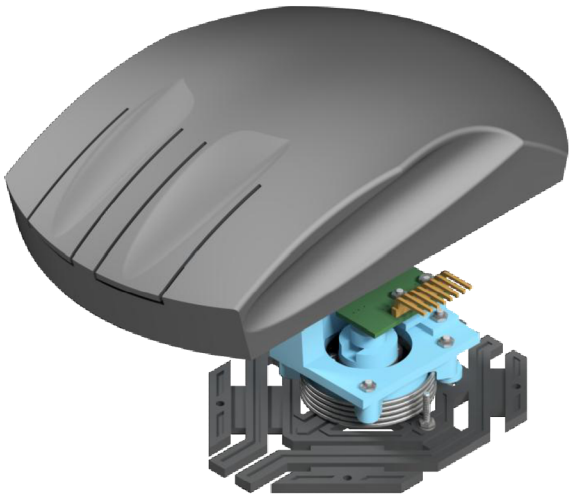
Obr. 2.12: Blokovací kolík na lineárnej časti.

2.3 Rotačná časť

Za inšpiráciu pri návrhu rotačnej časti sa zvolil mechanizmus založený na princípe gyroskopu s tromi stupňami voľnosti, kde každá os otáčania je kolmá na ostatné. V tomto prípade ale usporiadanie týchto troch osí bolo odlišné. Kým v gyroskope každé ďalšie ložisko udržiava zvyšné súčiastky v smere do vnútra zariadenia, v 3D joysticku musela podstava a aj rukoväť ostať na vonkajšej strane. Týmto spôsobom sa zabráni viacnásobnej rotácii okolo dvoch osí, čo ale vôbec nie je problém pri tejto konkrétnej aplikácii. Typ použitých snímačov je rovnaký ako na lineárnej časti, akurát sa zvolil kvalitnejší model s vyšším rozlíšením výstupných hodnôt. Dôvodom je oveľa menší rozsah natočenia, ktorý tu môže byť snímačmi zachytený.

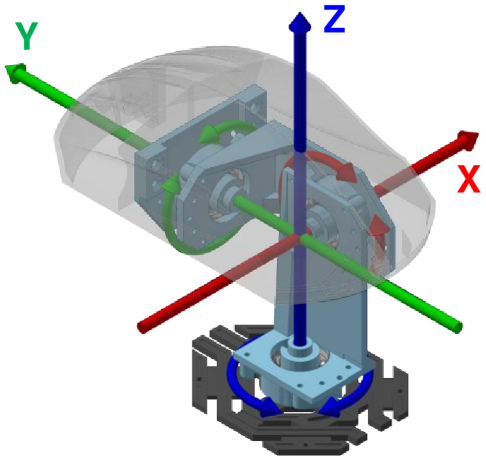
Na Obr. 2.13 je zobrazený hotový model rotačnej časti mechanizmu bez kabeláže. Príloha č. 3 obsahuje detailnejšiu zostavu v rozloženom pohľade. Podobne ako je to na lineárnej časti, aj tu má podstava štyri malé diery pre skrutky M2. Jej nezvyčajný štýl má čisto estetickú funkciu, keďže na udržanie zariadenia by stačil aj jednoduchý rám. Rukoväť sa podobá skôr veľkým počítačovým myšiam, než ako klasickým joystickom. Vzhľadom na to, aké časté je nevhodné pomenovanie niektorých nových zariadení, sa názov joystick zvolil podľa jeho plánovaného využitia. Tvar rukoväte je len čiastočne ergonomický kvôli možnosti uchopenia pravou aj ľavou rukou. Na bočných stranách sa nachádzajú dve veľké mierne naklonené drážky, ktoré umožňujú pevné uchopenie a jednoduchú manipuláciu. Na

tlačidlách sú tiež dve malé priehlbiny na ich jednoduché vyhmatanie. Rozmery rukoväte sú nasledujúce: šírka 90 mm, dĺžka 162 mm a výška 45 mm. Z toho vyplýva, že ovládanie je pohodlné aj s veľkými rukami.



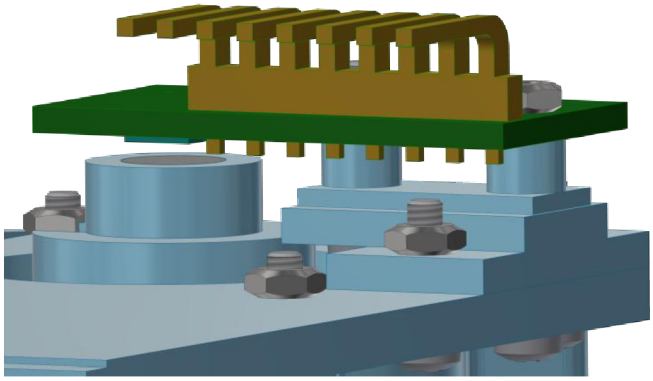
Obr. 2.13: Model rotačnej časti mechanizmu.

Všetky tri osi otáčania (pozri Obr. 2.14) sú prepojené v sérii. Podstava má na svojej hriadeli uchytené guľôčkové ložisko, ktoré je pripevnené na rám v tvare L rotujúci len okolo osi Z. Ďalšie ložisko sa na kolmú stenu taktiež namontuje rovnakým spôsobom, a tým umožňuje druhému rámu natočenie v osi X. Tretí stupeň voľnosti sa získa pridaním ešte jedného ložiska na kolmú stenu predošlého rámu. Rukoväť je potom napevno spojená s doskou rotujúcou okolo osi Y. Základná poloha sa udržiava pomocou torzných pružín na vonkajšej strane držiakov ložísk. Ich podrobnejší popis sa nachádza nižšie v texte. Každé ložisko umožňuje natočenie rukoväte okolo svojej osi v rozsahu ± 35°. Zaujímavosťou tejto konštrukcie je, že všetky tri osi sa pretínajú v jednom bode, ktorý sa nachádza tesne pod rukoväťou v jej zadnej časti. Týmto umiestnením kĺbu sa počas používania zariadenia drasticky zmenšia nutné výkyvy zápästia a lakeť môže pohodlne ostať na stole. Zaistenie prekrižovania všetkých troch osí priamo v strede zápästia v tomto prípade nebolo možné. Na to by sa muselo vytvoriť nové omnoho väčšie zariadenie, ktoré ložiskami obkolesuje požadované miesto kĺbu.



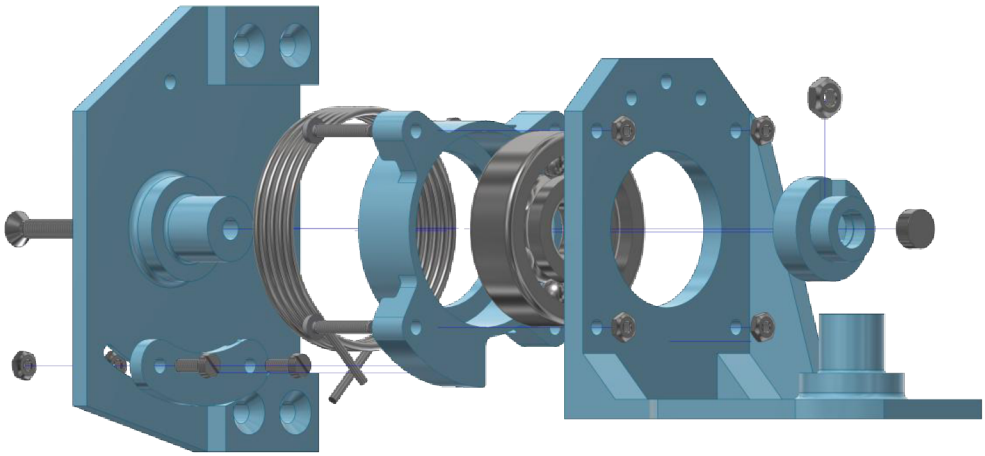
Obr. 2.14: Osi rotačného pohybu.

Na Obr. 2.15 je znázornené uchytenie snímača podobne ako na lineárnej časti. DPS sa pripevní k rámu cez plochý držiak a malú podložku, ktorej výšku je nutné s pilníkom zmenšiť, až kým nie je magnet od snímača v potrebnej vzdialenosti. Namiesto použitia ozubených prevodov na zvýšenie rozsahu natočenia magnetu sa radšej zvolil snímač so štvornásobným rozlíšením, keďže prevody vyrobené na 3D tlačiarni sú veľmi často deformované a do systému by priviedli dodatočné nepresnosti. Rozmery zariadenia by týmto tiež boli nepriaznivo ovplyvnené.



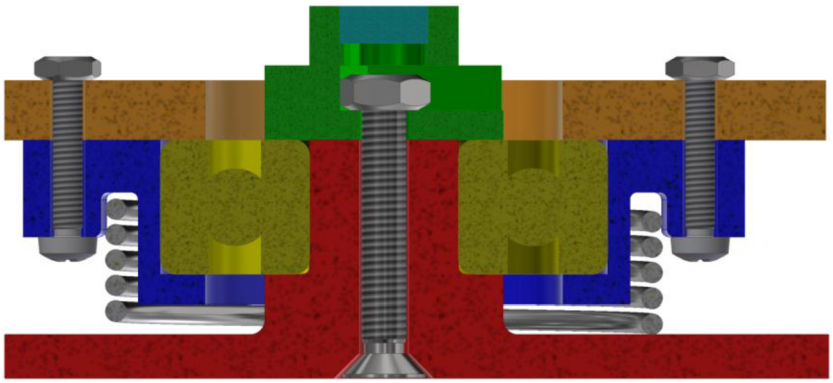
Obr. 2.15: Umiestnenie snímača na rotačnej časti.

Zostava súčiastok v okolí osi rotácie Y je zobrazená na Obr. 2.16. Na hriadeľ dosky vľavo sa nasunie torzná pružina a ložisko vo svojom puzdre, ktoré sa štyrmi skrutkami M2 pripevní k pravému rámu. Matica M3 sa vloží z bočnej strany do držiaka magnetu, ten sa potom natočí na skrutku M3 vychádzajúcu zo stredu hriadeľa, čím sa k doske prichytí aj vnútorný krúžok ložiska. Rotácia vo všetkých troch osiach sa realizuje práve takýmto spôsobom. Magnety sa do držiaka vložia až pred montážou DPS, aby smer magnetického poľa bol vzhľadom na snímače približne rovnaký a počas práce nedochádzalo k prechodu cez nulovú hodnotu, čo by si vyžadovalo zbytočné a zdĺhavé ošetrenie v kóde programu.



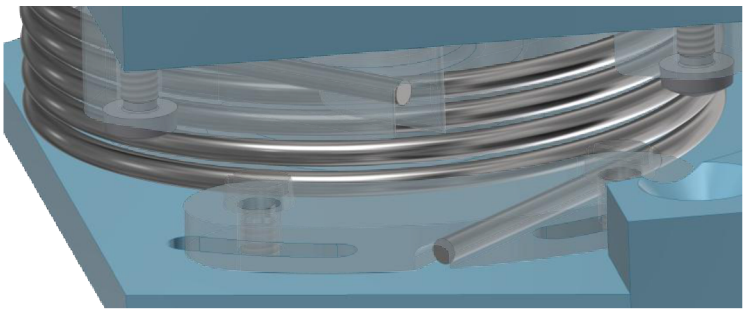
Obr. 2.16: Zostava osi rotácie v rozloženom pohľade.

Detailnejšie zobrazenie uloženia ložiska je na Obr. 2.17. Oranžová časť na obrázku je rám, červená doska s hriadeľom, svetlomodrá a zelená sú magnet a jeho držiak, žltá a tmavomodrá zase ložisko a jeho puzdro, okolo ktorého je namotaná torzná pružina. Prierez bol vykonaný diagonálne, aby sa zachytilo čo najviac detailov.



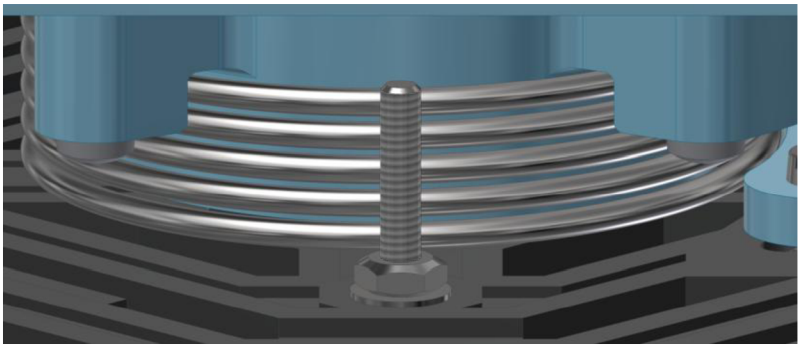
Obr. 2.17: Prierez osi rotácie.

Pružina sa k doske alebo k rámu pripevní pomocou zakriveného plochého držiaka s drážkou na spodnej strane, ako je znázornené na Obr. 2.18. Podobná, ale hlbšia drážka je aj na puzdre ložiska. Do nich sa vložia ramená torznej pružiny, čo v tejto zostave vytvára silu pri vychýlení zo základnej polohy. Puzdro je síce s rámom pevne spojené, ale s malým držiakom na opačnej strane je možné po uvoľnení skrutiek posúvať po kružnici okolo osi, a tým nastaviť požadovanú základnú polohu. Kvôli väčšej váhe prednej časti mechanizmu je nutné presunúť miesto uchytenia pružiny na osi X tak, aby sa moment od dodatočnej deformácie vyrovnal s momentom od gravitačnej sily. Taktiež treba zväčšiť priemer drôtu tejto pružiny na zachovanie dostatočne veľkej tuhosti a bezpečnosti.



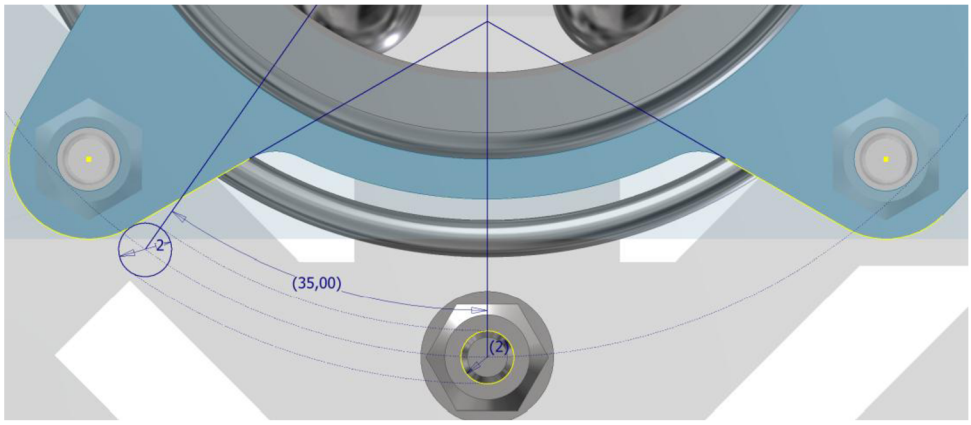
Obr. 2.18: Umiestnenie torznej pružiny.

Obmedzenie uhlu natočenia na určitú požadovanú hodnotu sa zaistí blokovacím kolíkom (pozri Obr. 2.19) umiestneným na kružnici medzi dvoma výbežkami ložiskového puzdra. Tie pri maximálnom natočení narazia na daný kolík, čím sa znemožní ďalší pohyb. Kvôli malým rozmerom puzdra sa na to musela použiť skrutka M2, keďže podobne tenká plastová súčiastka by bola pravdepodobne príliš slabá a pri zvolených uhloch by väčšie časti už nemali dosť miesta na kružnici.



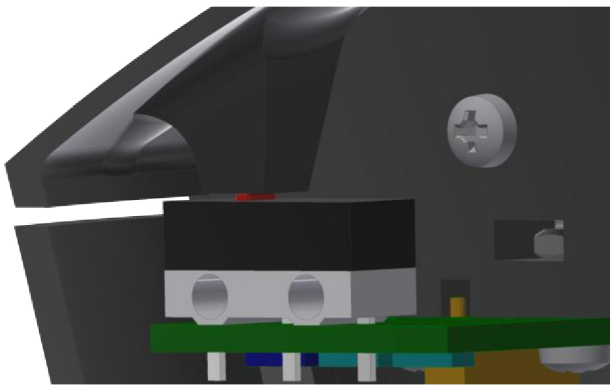
Obr. 2.19: Blokovací kolík na rotačnej časti.

Presná poloha blokovacieho kolíku sa zistila premietnutím okrajov ložiskového puzdra a ich následným spojením s kružnicou skrutky posunutou o požadovaný uhol. Ako je z náčrtu na Obr. 2.20 vidno, pri maximálnom natočení puzdra o uhol 35° by sa skrutka dotýkala výbežku tesne na jeho okraji. Na obrázku je znázornené len výsledné umiestnenie kolíku, kde uhol je poháňaným rozmerom. Pri návrhu to ale bola vzdialenosť skrutky od stredu osi otáčania. Týmto spôsobom sa dá vyhnúť komplikovaným výpočtom a predať tak prácu softvéru.



Obr. 2.20: Umiestnenie blokovacieho kolíku na rotačnej časti.

Typ konštrukcie tlačidiel bol inšpirovaný klasickými počítačovými myšami. V nich sa prstom mierne ohne tenká ohybná doska, na ktorej spodnej strane sa nachádza malý naklonený stĺpik. Ten sa v základnej polohe len dotýka vrchnej časti mikrospínača na DPS. Deformáciou dosky sa ale tlačidlo na mikrospínači zatlačí, čím sa vytvorí požadovaný signál. Na Obr. 2.21 je zobrazené umiestnenie DPS s tlačidlami v prednej časti rukoväte, kde jediným vhodným spôsobom montáže bolo vloženie matice do malého výrezu v plaste a následné pripevnenie skrutkou zo spodku. Na to sa vzala inšpirácia z 3D tlačiarne, ktorej akrylátové dosky majú na okrajoch výrezy v tvare znaku plus.



Obr. 2.21: Umiestnenie tlačidiel na rotačnej časti.

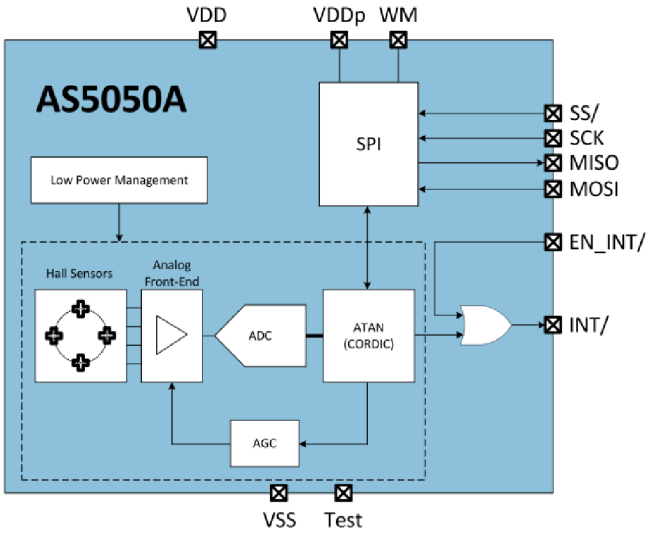
3 NÁVRH ELEKTRONIKY

Požiadavkou tejto časti práce bol výber vhodných snímačov, mikrokontroléru a návrh dosiek plošných spojov, čo sa v tomto poradí aj vykonalo. Riadiaca jednotka sa mohla zvoliť len s dostatočnou znalosťou parametrov snímačov, a DPS na ich obsluhu navrhnúť až po odskúšaní periférií a komunikácie. Predbežné skúšky prebiehali väčšinou len na kontaktnom poli (breadboard) prepojeným s mikrokontrolérom, kvôli možnosti rýchleho prototypovania. Po dokončení návrhu boli všetky DPS zhotovené firmou Gatema, ale ich elektronické súčiastky sa museli osadiť ručne. Veľká časť kabeláže sa tiež manuálne vyrobila z jednotlivých dielov.

3.1 Voľba snímačov

Precízne snímanie uhlu natočenia sa môže dosiahnuť rôznymi spôsobmi. Najjednoduchším z nich je za pomoci potenciometrov. Tie sú však náchylné na akékoľvek rušenie v obvode a vyžadujú veľmi stabilný zdroj napätia, aby sa zaistili presné výstupné hodnoty. Ďalším spôsobom sú absolútne optické enkodéry, ktorých vysoká cena ale nie je vyhovujúca pri potrebe až šiestich snímačov. Relatívne enkodéry zas musia byť resetované pri každom zapnutí, a to natočením do krajnej polohy.

Najlepšou voľbou sa stali otočné snímače polohy AS5050A [25] a AS5055A [26]. Sú to absolútne magnetické enkodéry na snímanie uhlu natočenia pri jednej celej otáčke. Na Obr. 3.1 je znázornený blokový diagram AS5050A, ktorý má 10-bitové rozlíšenie. Pokročilejší model AS5055A sa odlišuje len v rozlíšení, v tomto prípade 12-bitovým. Nevyhnutnou súčasťou týchto snímačov je valcový neodýmový magnet, ktorý musí byť umiestnený nad alebo pod ich Hallovými sondami v strede čipu. Signály z týchto sond sú najprv zosilnené a konvertované AD prevodníkom, potom sa prepočítajú na výsledný uhol. V lineárnej časti mechanizmu sa kvôli nutnosti ozubených prevodov použili jednoduchšie snímače AS5050A, keďže sa tu mohol nastaviť prevodový pomer tak, aby sa využila väčšina rozsahu 10-bitového rozlíšenia. Rotačná časť mechanizmu s 12-bitovým modelom AS5055A dosahuje na menšom rozsahu približne rovnaký počet hodnôt ako AS5050A.

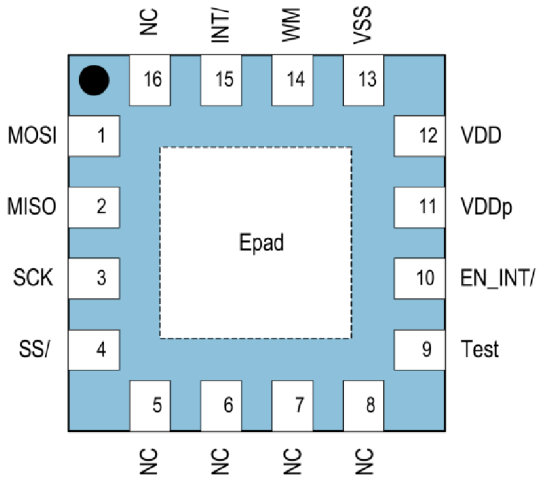


Obr. 3.1: Blokový diagram AS5050A [25].

Pri odporúčanom napájacom napätí 3,3V je spotreba prúdu 8,5 mA počas čítania hodnoty a komunikácie, ale len 33 μA v režime nízkej spotreby. Vhodným nastavením v registri sa dá tento prúd znížiť až na 3 μA.

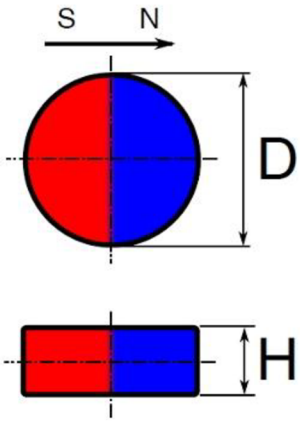
Dôležitou vlastnosťou týchto snímačov je podpora 16-bitového komunikačného rozhrania SPI, čo predstavuje veľmi rýchlu a efektívnu metódu na získavanie hodnôt uhlu natočenia zo všetkých šiestich snímačov. SPI podporuje obojsmernú súčasnú komunikáciu pri frekvenciách až do 10 MHz a využitím prerušenia na čipe dokonca umožňuje zapojenie veľkého množstva snímačov v konfigurácii Daisy Chain len pomocou piatich vodičov vychádzajúcich z mikrokontroléru. V tomto projekte sa ale kvôli lepšej ovládateľnosti zvolila klasická konfigurácia s tromi vodičmi a jedným dodatočným pre každý jednotlivý snímač.

Typ puzdra s označením QFN-16 je rovnaký pre AS5050A aj AS5055A. Rozmery obidvoch snímačov sú teda 4x4x0,85 mm. Ich pinový diagram je zobrazený na Obr. 3.2. Toto puzdro má všetkých 16 pinov umiestnených len na spodnej strane, takže ich osadenie na DPS si vyžaduje teplovzdušnú pec alebo špeciálnu výhrevnú dosku.



Obr. 3.2: Pinový Diagram AS5050A [25].

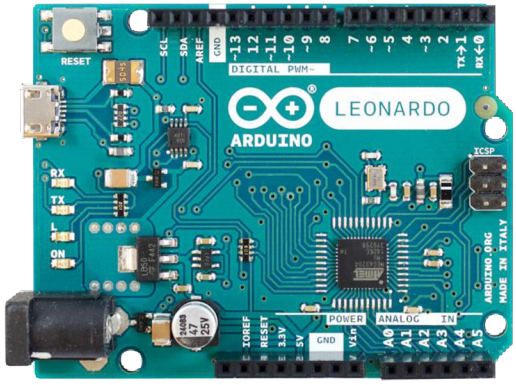
Jeden z odporúčaných valcových neodýmových magnetov k týmto snímačom má označenie AS5000-MD6H-3 [27]. Je magnetizovaný diametrálne (pozri Obr. 3.3), čiže jeho severný a južný pól sa nachádzajú na opačnej strane kružnice. Hallové sondy zisťujú smer magnetického poľa, ktorý sa mení natočením, práve týmto spôsobom. Priemer tohto magnetu je 6 mm a hrúbka 2,5 mm. Povrch je upravený niklovaním kvôli ochrane pred koróziou.



Obr. 3.3: Diametrálny magnet [28].

3.2 Voľba mikrokontroléru

Signály zo zvolených snímačov museli byť spracované riadiacou jednotkou s podporou SPI komunikácie. Vývojová doska Arduino Leonardo [29], znázornená na Obr. 3.4, bola na túto úlohu ideálna. Je založená na mikrokontroléri ATmega32u4 [30], obsahuje 16 MHz oscilátor, micro USB konektor, napájací konektor, ICSP konektor, tlačidlo na resetovanie a až 20 digitálnych vstupno-výstupných pinov vyvedených na konektory, z čoho sa 7 môže použiť na PWM výstup a 12 na analógový vstup. Na doske sa taktiež nachádza napäťový regulátor na 5V a druhý na 3,3V. Napájanie sa môže realizovať cez vstavaný micro USB konektor alebo externým zdrojom, napríklad batériou alebo adaptérom s jednosmerným prúdom. Keďže väčšina vyvedených pinov má podobu dutinkovej lišty, ich zapojenie do testovacích obvodov poskladaných na kontaktnom poli je nesmierne jednoduché, veľmi rýchle a bez potreby akéhokoľvek spájkovania.



Obr. 3.4: Arduino Leonardo [31].

Čisto nový kus tejto vývojovej dosky je zvyčajne uložený v priehľadnej plastovej podložke kvôli ochrane spodnej vrstvy a dvoch dodatočných dier na uchytenie pri montáži. Zhrnutie niektorých jej parametrov je dostupné v Tab. 3.1.

Tab. 3.1: Parametre Arduino Leonardo [29].

Microcontroller	**ATmega32u4**
Operating Voltage	5V
Input Voltage (Recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega32u4) of which 4 KB used by bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Clock Speed	16 MHz
Lenght	68.6 mm
Width	53.3 mm
Weight	20 g

Najdôležitejšou časťou dosky Arduino Leonardo je riadiaca jednotka ATmega32u4. Hodnoty zo snímačov získané pomocou SPI rozhrania sa môžu ďalej poslať do počítača. Tento čip má vstavaný USB modul, takže nie je nutné použiť druhý procesor na prevod komunikácie medzi rozhraniami USART a USB. To tiež znamená, že sa v počítači môže javiť ako myš alebo klávesnica. Po určitých úpravách v kóde sa s ním dajú emulovať aj iné zariadenia, ako je joystick a gamepad, dokonca aj súčasne viac kusov a rôznych druhov popri sériovej komunikácii cez ten istý kábel.

Nahrávanie nového kódu do mikrokontroléru je mimoriadne zjednodušené využitím programu bootloader, ktorý síce zaberá až 4 KB z pamäte. Týmto spôsobom nie je nutné použitie externého programátora. Na celý proces postačí jeden USB kábel, ktorý slúži na komunikáciu s počítačom a tiež napájanie čipu. Čiže po naprogramovaní nie je potrebné prepojovať vôbec nič a môže sa s Arduinom hneď začať pracovať. Nový kód sa jediným kliknutím do niekoľkých sekúnd nahrá do pamäte a automaticky sa spustí. Z toho dôvodu je vývoj na tomto modeli Arduina veľmi efektívny. Alternatívna metóda na programovanie je dostupná prostredníctvom ICSP konektoru, ktorý taktiež slúži aj na komunikáciu cez SPI. Zhrnutie parametrov mikrokontroléru ATmega32u4 je v Tab. 3.2.

Tab. 3.2: Parametre ATmega32u4 [30].

Name	Value
Program Memory Type	Flash
Program Memory Size (KB)	32
CPU Speed (MIPS/DMIPS)	16
SRAM Bytes	2,560
Data EEPROM/HEF (bytes)	1024
Digital Communication Peripherals	1-UART, 2-SPI, 1-I2C
Capture/Compare/PWM Peripherals	2 Input Capture, 2 CCP, 12PWM
Timers	2 x 8-bit, 2 x 16-bit
Number of Comparators	1
Number of USB Modules	1, Full Speed
Temperature Range (C)	-40 to 85
Operating Voltage Range (V)	2.7 to 5.5
Pin Count	44

Zariadenia od Arduina sú navrhnuté hlavne na rýchle prototypovanie a nenáročné skúšanie rôznych elektrických obvodov. Ich vývojové prostredie užívateľom sprostredkuje veľké množstvo funkcií určených na jednoduchú prácu s mikrokontrolérom a perifériami. Väčšina týchto funkcií má spoločný zápis pre všetky modely Arduina, ale v ich registroch sú jednotlivé vstupy a výstupy často odlišné. Aby ich používanie bolo oveľa prehľadnejšie, zaviedla sa u nich substitúcia názvov, ktorá navyše ošetruje aj niektoré chyby spôsobené užívateľom, napríklad pri pokuse o čítanie analógovej hodnoty na nesprávnom pine. To ale veľakrát vedie k zvýšenému počtu potrebných inštrukčných cyklov a k spomaleniu behu programu.

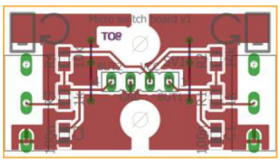
3.3 DPS pre tlačidlá na rotačnej časti

Manipulácia s rukoväťou na rotačnej časti si vyžaduje len tri prsty na pravej ruke. Zvyšné dva preto môžu byť využité na iné účely. Podobne ako pri bežných počítačových myšiach,

aj tu sa pridali dve veľké hlavné tlačidlá pod ukazovák a prostredník, ale koliesko s tretím tlačidlom sa už na rukoväť nezmestilo.

Najdôležitejšími prvkami na tejto DPS sú mikrospínače, ktorých použitie v dnešnej dobe je v podstate štandardom v ktorejkoľvek modernej počítačovej myši. Dva takéto kusy boli z jednej starej nepotrebnej myši odstránené a použité v tejto DPS. Mikrospínače sú síce navrhnuté na rýchle spínanie, ale aj napriek tomu je nutné ich signály pred ďalším spracovaním ešte filtrovať. To je spôsobené tým, že po stlačení sa vodivá pružina viackrát odrazí od kontaktnej plochy na opačnej strane, čím sa strieda uzavretý a otvorený obvod. Celý proces trvá niekoľko milisekúnd a končí ustálením pružiny v požadovanej polohe. Filtrovať takéto signály je čiastočne možné priamo na úrovni hardvéru pridaním vhodného kondenzátora a rezistora na správne miesto v obvode. V niektorých prípadoch sa tiež používa aj čip s hysteréznymi vlastnosťami na ošetrenie okolia prechodu napäťovej úrovne z logickej nuly na logickú jednotku. Bez osciloskopu nebolo možné dostatočne presne zmerať dobu trvania týchto odrazov. Po mnohých testoch tlačidla s rôznou kombináciou hodnôt súčiastok sa zvolila jednoduchšia metóda filtrácie len s použitím kondenzátora a rezistora. Úplná filtrácia signálu sa nakoniec dosiahla softvérovo, čiže v programe mikrokontroléru, čo je detailnejšie popísané v ďalšej kapitole [32].

Príloha č. 4 obsahuje schému zapojenia a zväčšený model tejto DPS. Jej napájacie napätie má 5V a je privedené spolu so signálovými vodičmi do dutinkovej lišty v strede dosky priamo z konektorov Arduina. Kondenzátor s kapacitou 100nF sa nabíja až cez dva rezistory. Jeden z nich s odporom 10kΩ sa nachádza na doske a druhý je vstavaný (internal pull-up resistor) v čipe ATmega32u4 s odporom v rozmedzí 20kΩ až 50kΩ. Stlačením sa plne nabitý kondenzátor vybije cez tlačidlo s 1kΩ rezistorom. Na Obr. 3.5 je zobrazená táto DPS v mierke 1:1, kvôli lepšej predstave jej reálnej veľkosti. Výškou 20 mm a šírkou 36 mm sa dosku dá veľmi ľahko nasadiť na prednú časť rukoväte.

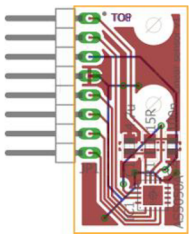


Obr. 3.5: DPS pre tlačidlá na rotačnej časti (skutočná veľkosť).

3.4 DPS pre snímače na lineárnej časti

Zapojenie obidvoch typov snímačov je úplne rovnaké, líšia sa akurát v rozložení súčiastok. Poloha snímača musela byť od stredu dosky posunutá o tri milimetre doprava, pretože na lineárnej časti mechanizmu sa už nemohol odobrať žiadny materiál z okraja vozíka. Jedna strana čipu našťastie nemusí byť vôbec zapojená, takže nebol problém ho celý natočiť tým smerom ku kraju dosky. Podľa datasheetu snímača sa k napájacím pinom pridali filtračné kondenzátory s kapacitami 100nF a 4,7μF a rezistor do série s odporom 15Ω. Na ľavej strane sa nachádza 8-miestny konektor s obojstrannými kolíkmi zahnutými o 90°, aby sa zachoval tenký profil dosky. Prvý pin zdola je slave select (SS/) a slúži na voľbu snímača počas komunikácie a druhý pri tom prijíma frekvenčné pulzy (SCK). Tretí a štvrtý pin sú výstupná (MISO) a vstupná (MOSI) zbernica z pohľadu tohto čipu. Ďalej nasleduje zem (GND), potom spínač prerušenia (EN_INT/), ktorým po aktivácii môže siedmy pin (INT/) hlásiť dostupnosť novej hodnoty uhlu, a nakoniec posledným ôsmym pinom je napájacie napätie (VDD), ktoré je v tomto prípade spoločné aj pre komunikačnú periférium SPI [25].

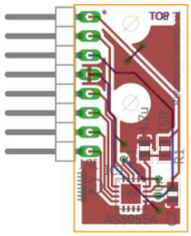
Na tejto doske bol použitý snímač AS5050A s 10-bitovým rozlíšením. Príloha č. 5 obsahuje schému zapojenia a zväčšený model DPS z obrázka Obr. 3.6. Výšku má 30 mm a šírku zas 15 mm.



Obr. 3.6: DPS pre snímače na lineárnej časti (skutočná veľkosť).

3.5 DPS pre snímače na rotačnej časti

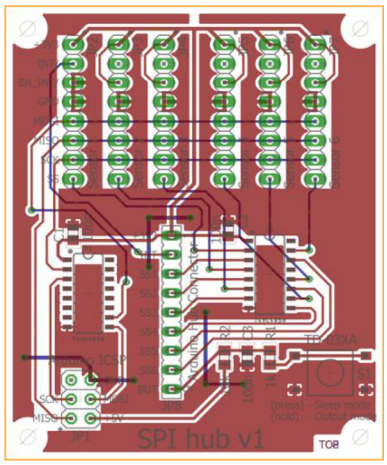
Najpodstatnejším rozdielom tohto typu DPS je použitie 12-bitového snímača AS5055A a jeho zarovnanie na strednú dolnú časť. Konektor je tiež zapojený k rovnakým pinom, ale v tomto prípade zo spodnej strany dosky. Ich poradie ostalo nezmenené. Zvyšné súčiastky boli akurát len premiestnené. Príloha č. 6 zahŕňa schému zapojenia a model DPS v mierke 4:1, ktorý je zobrazený na Obr. 3.7 v skutočných rozmeroch. Výška a šírka dosky ostali bez akejkoľvek zmeny, 30 mm a 15 mm.



Obr. 3.7: DPS pre snímače na rotačnej časti (skutočná veľkosť).

3.6 DPS rozbočovača pre SPI

Rozbočovač pre SPI je nevyhnutný na komunikáciu so snímačmi už len kvôli dvom kusom prekladačov napäťových úrovní 74HC4050D [33], ktoré menia 5V signály z Arduina na 3,3V. Keby boli snímače dlhodobo prevádzkované na plných 5V, mohlo by ľahko dôjsť k ich nenávratnému poškodeniu. Opačným smerom už úroveň napätia zvyšovať netreba, pretože mikrokontrolér ATmega32u4 na Arduine považuje hodnoty nad 1,9V za logické jednotky. Napájanie 5V sa so zemou a tromi hlavnými vodičmi na komunikáciu cez SPI (SCK, MISO, MOSI) privádza z ICSP konektora z Arduina na jeho zrkadlovo prevrátenú formu na tejto DPS. Do pravého dolného rohu sa pridalo jedno tlačidlo, ktoré slúži hlavne na okamžité zapnutie spánkového režimu a na prepínanie režimu spracovania výstupných hodnôt po dlhšom stlačení. V strednej dolnej časti sa nachádza 9-miestna dutinková lišta, ktorá z prvého pinu zhora zásobuje snímače a prevodníky napätím 3,3V. Druhý pin (INT) posiela signál prerušenia do Arduina len z prvého snímača. Nasledujúcich šesť slave select pinov (SS1-SS6) je zapojených cez prevodník úrovní do každého snímača. Posledný pin je pripojený k tlačidlu podobným spôsobom ako už bolo v podkapitole 3.3 popísané. Celá horná polovica dosky je osadená obojstrannými kolíkmi, na ktoré sa pripojí všetkých šesť snímačov. V každom zo štyroch rohov sa taktiež nachádza diera s priemerom 2 mm kvôli jednoduchej montáži na rovný povrch. Na Obr. 3.8 je znázornená táto DPS v skutočnej veľkosti, čiže výšky 60 mm a šírky 50 mm. Príloha č. 7 obsahuje schému zapojenia a Príloha č. 8 zas DPS pri trojnásobnom zväčšení.

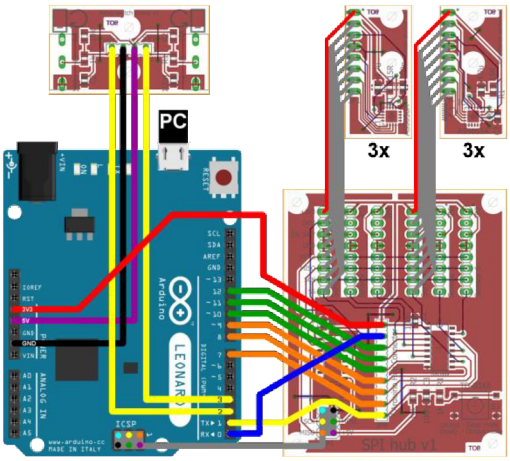


Obr. 3.8: DPS rozbočovača pre SPI (skutočná veľkosť).

Miernou zmenou v návrhu zapojenia konektorov na snímače by sa dala realizovať komunikácia metódou „daisy chain“, v ktorej sú jednotlivé snímače prepojené v sérii jeden za druhým a posielajú si medzi sebou celé reťazce prijatých povelov, ale aj odoslaných hodnôt uhlov. Výhodou je, že na celú komunikáciu postačí len päť vodičov vychádzajúcich z Arduina. Veľmi nepríjemnou nevýhodou je to, že akákoľvek porucha čipu alebo chyba s SPI môže znehodnotiť prijaté informácie a na získanie jedinej chýbajúcej hodnoty sa musí znova poslať povel cez všetkých šesť snímačov. V tomto projekte sa metóda daisy chain nepoužila kvôli nedostatku predošlých skúseností s magnetickými enkodérmi a nepresným plastovým súčiastkam vyrobeným na 3D tlačiarni. DPS snímačov ale boli navrhnuté tak, aby fungovali pri komunikácii obidvomi metódami. Záleží len na vhodnom zapojení na rozbočovači pre SPI.

3.7 Prepojenia medzi DPS

V Tab. 3.3 sú zhrnuté prepojenia vychádzajúce z Arduina spolu s popisom ich funkcie. Vynechané z nej ostali len DPS snímačov, ktoré sú najprv zapojené cez rozbočovač pre SPI. Príloha č. 9 obsahuje nákres všetkých prepojení medzi DPS. Ten istý nákres je na Obr. 3.9 znázornený v zmenšenej podobe kvôli rýchlemu odkazovaniu z tabuľky. Farby vodičov v hotovom zariadení zodpovedajú obrázku aj tabuľke až na fialové 5V napätie a jednotlivé piny na ICSP konektore, ktoré sú tam len kvôli prehľadnosti. Zobrazené sú tu len dva snímače, jeden z každého typu, pretože ich zapojenie je identické.



Obr. 3.9: Zmenšený nákres prepojení medzi DPS [34].

Tab. 3.3: Prepojenia z Arduina.

Arduino Leonardo			**DPS**		
PIN	**PORT**	**Funkcia**	**Pripojené na**	**PIN**	**Farba**
D0	PD2	prerušenie snímača 0	SPI_hub	INT	
D1	PD3	tlačidlo spánku a módu	SPI_hub	BUT	
D2	PD1	pravé tlačidlo	Micro_Switch_Board	BUT2	
D3	PD0	ľavé tlačidlo	Micro_Switch_Board	BUT1	
D4	PD4	voľné	-	-	-
D5	PC6	nezapojené (SPI ERROR FLAG)	-	-	-
D6	PD7	voľné	-	-	-
D7	PE6	slave select snímača 5	SPI_hub	SS6	
D8	PB4	slave select snímača 4	SPI_hub	SS5	
D9	PB5	slave select snímača 3	SPI_hub	SS4	
D10	PB6	slave select snímača 2	SPI_hub	SS3	
D11	PB7	slave select snímača 1	SPI_hub	SS2	
D12	PD6	slave select snímača 0	SPI_hub	SS1	
D13	PC7	nezapojené (LED_BUILTIN)	-	-	-
ICSP konektor					
1	PB3	MISO	SPI_hub	MISO	
2	5V	napájanie tlačidla 5V	SPI_hub	+5V	
3	PB1	SCK	SPI_hub	SCK	
4	PB2	MOSI	SPI_hub	MOSI	
5	RST	nezapojené (RESET)	SPI_hub	-	
6	GND	zem snímačov	SPI_hub	GND	
Napájanie					-
-	3V3	napájanie snímačov 3,3V	SPI_hub	+3V3	
-	5V	napájanie tlačidiel 5V	Micro_Switch_Board	VCC	
-	GND	zem tlačidiel	Micro_Switch_Board	GND	

4 NÁVRH OBSLUŽNÉHO SOFTVÉRU

V tomto projekte sa predpokladalo využitie 3D joysticku v prvom rade pri počítačových simulátoroch. Vďaka generickým USB ovládačom, ktoré dokážu spracovať prichádzajúce údaje na kvázi analógové signály požadované týmito aplikáciami, stačilo naprogramovať len vývojovú dosku Arduino Leonardo. Okrem získavania hodnôt natočení zo snímačov a obsluhy tlačidiel sa tiež muselo zaistiť, aby tieto informácie boli správne naformátované a odoslané do počítača cez USB. Na ten účel už našťastie existuje knižnica Arduino Joystick Library [35], ktorá dokáže emulovať jeden alebo viac joystickov. V počítači sa preto javia ako generické HID zariadenia pripravené na okamžité používanie, ktoré už nie je potrebné nijak kalibrovať. Bez nutnosti inštalácie vlastných ovládačov a s podporou automatického rozoznávania sa z tohto 3D joysticku prakticky stáva „plug and play“ zariadenie.

4.1 Štýl programovania

Každé kvalitné ovládacie zariadenie by muselo poskytovať hodnoty s dostatočne vysokou frekvenciou, aby užívateľ pociťoval čo najnižšiu odozvu. Toho sa dá dosiahnuť rýchlym a efektívnym programom, ktorý nevykonáva príliš veľa zbytočných inštrukcií. Preto je veľmi dôležité rozumne rozhodnúť medzi prehľadnosťou a účinnosťou kódu. Ak má nejaký program byť modulárny a využiteľný aj v iných aplikáciách, musí sa napísať určitými konvenciami a musí sa v ňom dať jednoducho orientovať. Keďže toto zariadenie bolo navrhnuté hlavne na osobné účely, vysoká účinnosť kódu je dôležitejšia ako jej dobrá prehľadnosť.

Existuje mnoho spôsobov, ako zrýchliť beh programu, alebo ho aspoň nezaťažovať zbytočnými inštrukciami. V tomto projekte sa často používali `inline` funkcie, ktorých výhodou je ušetrenie veľkého počtu inštrukčných cyklov na úkor zvýšených požiadaviek na pamäť. Funkcia `inline`, namiesto odkazu na svoje zavolanie, priamo skopíruje svoj obsah na miesto použitia. Jej implementácia ale v niektorých prípadoch nie je možná. Kompilátor navyše berie toto kľúčové slovo len ako návrh a nie ako pokyn. V niektorých prípadoch je preto jediným riešením použiť makro funkcie. S nimi treba narábať veľmi opatrne, keďže sa jedná o surové nahradenie textu ešte pred samotnou kompiláciou kódu a pri nepozornosti môžu nastať rôzne problémy, ktoré sa len veľmi ťažko hľadajú. Nie je to síce odporúčaný štýl programovania, ale po praktickej stránke dokážu makro funkcie ušetriť nesmierne veľký počet inštrukčných cyklov. Nižšie v podkapitole 4.3 sú popísané konkrétne príklady makier.

Použitie vhodného typu premennej môže tiež výrazne ovplyvniť výkon. Opakované čítanie tej istej hodnoty z pamäte nie je veľmi efektívne. Oveľa lepším spôsobom je túto premennú pred jej spracovaním najprv zapísať do lokálnej premennej, ktorú potom môže kompilátor dočasne prideliť do niektorého voľného registra. To je výhodné hlavne preto, lebo všetky aritmetické operácie prebiehajú prostredníctvom hodnôt v registroch. Navyše globálne premenné nikdy nie sú alokované do týchto registrov, aby boli dostupné aj pri prerušeniach. Ak sa určité hodnoty nebudú meniť v celom programe, pridaním kľúčového `const` dokáže kompilátor ušetriť miesto v pamäti a v strojovom kóde vo výpočtoch potom použiť obyčajné čísla namiesto odkazu na premenné [36].

V tejto práci sa dodatočné zvýšenie výkonu programu získalo rozvinutím krátkych iteračných cyklov. Jedná sa o vynechanie podmienky spolu s inkrementáciou počítadla a rozpísanie jednotlivých iterácií za sebou. Touto metódou sa môže extrémne zrýchliť beh

programu na úkor mnohonásobne vyššej spotreby pamäte pre daný úsek kódu. Príkladom je nasledujúci iteračný cyklus:

```
for (byte i = 0;  i < 3;  i++)
{
    funkcia(i);
}
```

Ten sa po rozvinutí zapíše nasledovne:

```
funkcia(0);
funkcia(1);
funkcia(2);
```

Použitých snímačov bolo len šesť, takže rozvinutie cyklov si vôbec nevyžadovalo veľké úsilie. Druhou veľmi významnou výhodou bola jej synergia s kompilátorom. Zadávaním konštantnej hodnoty identifikátorov snímačov do argumentov funkcií sa kompilátoru tak umožnilo zapisovanie priamych odkazov príslušných premenných namiesto ich zdĺhavého vyhľadávania počas behu programu [36].

Na zistenie presného počtu inštrukcií sa skompilovaný kód v súbore s príponou .elf musí spustiť v aplikácii avr-objdump.exe s príponou -S. Tým sa vygeneruje textový súbor, ktorý obsahuje celý kód zmiešaný v jazykoch C++ a assembler. Význam každej inštrukcie je možné nájsť v datasheete pre mikrokontroléry AVR [37]. Týmto spôsobom sa dajú odhaliť mnohé chyby, hlavne ak kompilátor vymaže nepotrebné úseky kódu, a miesta, ktoré by bolo vhodné upraviť a zrýchliť. Tiež sa tu jednoducho zistí, či je na určitú volanú funkciu odkaz alebo sa jej celý obsah skopíroval na požadované miesto ako pri napísaní kľúčového slova **`inline`**. V podstate sa takto môže kontrolovať účinnosť kódu a overiť, či sa nejaké zmeny, napríklad kvôli zvýšeniu rýchlosti, vôbec uplatnili.

4.2 Podmienená kompilácia

Počas programovania a testovania funkcií bolo často nutné dočasne vylúčiť niektoré časti kódu. Komentovať celé riadky alebo prepísať hodnoty na viacerých miestach nie je príliš výhodné a pri nepozornosti môže spôsobiť nepredvídané problémy. Jednoduchým riešením je využitie podmienenej kompilácie, kde sa preprocesoru prikáže, aby rozhodol, či určité kusy kódu budú skompilované alebo nie. V tomto projekte je možné zmenou jediného čísla vypnúť a zapnúť celé sady funkcií bez akýchkoľvek negatívnych následkov pre zvyšok programu. Do pamäte sa z nich teda nič nezapíše a nebudú hlásené žiadne chyby. Použitý kompilátor avr-gcc síce dokáže odstrániť bloky kódu, ktoré nemajú šancu byť vykonané, aj pri jednoduchých podmienkach v príkazoch **`if`** s konštantnými premennými. Musí pri nich ale ostať zachovaná správna syntax [37].

Problém však nastáva vtedy, keď má byť vynechaná aj definícia alebo deklarácia premennej. Nasledujúci kód hlási chybu deklarácie v danom rozsahu za druhým príkazom **`if`** pri **`hodnota1`**:

```
int podmienka1 = 0;
if (podmienka1 == 1) int hodnota1 = 1;
if (podmienka1 == 1) hodnota1 = 0; // 'hodnota1' was not declared in this scope
```

Riešenie spočíva v podmienenej kompilácii, kde preprocesor podľa makra **`podmienka2`** vyradí všetky príslušné bloky textu z kompilácie. V týchto blokoch sa navyše nekontroluje ani syntax. Zmenou makra **`podmienka2`** na hodnotu 1 ale bude daný text plnohodnotnou súčasťou kódu na kompiláciu, akoby tam príkazy preprocesora ani neboli.

```
#define podmienka2 (0)
#if podmienka2
  int hodnota2 = 1;
#endif
#if podmienka2
  hodnota2 = 0;
#endif
```

Podobným spôsobom bolo v zdrojovom kóde tohto projektu použitých šesť makier:

```
#define USE_JOYSTICK (1)
#define SERIAL_FEEDBACK (0)
#define SERIAL_PRINT_ALL_ANGLES (0)
#define RANGE_CALIBRATION (0)
#define RESET_ON_ERROR (1)
#define SENSITIVITY (0)
```

Najdôležitejším makrom podmienenej kompilácie bol **USE_JOYSTICK**, ktorým sa dal kompletne vylúčiť modul na používanie joysticku a odosielanie hodnôt do počítača. Nasledujúca definícia triedy **Joystick** popisuje vlastnosti zariadenia, ktoré sa nastavili na šesť osí pohybu s dvomi tlačidlami.

```
#if USE_JOYSTICK
  // Create the Joystick
  Joystick_ Joystick(JOYSTICK_DEFAULT_REPORT_ID,  // hidReportId,
    JOYSTICK_TYPE_JOYSTICK, 2, 0, // joystickType, buttonCount, hatSwitchCount
    true, true, true, true, true, true,
    // includeXAxis, Y, Z, includeRxAxis, Ry, Rz
    false, false, false, false, false);
    // Rudder, Throttle, Accelerator, Accelerator, Brake, Steering
#endif
```

Ďalšie dve makrá súvisia so sériovou komunikáciou, ktorou sa emuluje rozhranie USART na vypisovanie požadovaných hodnôt z mikrokontroléru na počítač, hlavne kvôli ladeniu programu. Makro **SERIAL_FEEDBACK** zapína modul a **SERIAL_PRINT_ALL_ANGLES** zas spúšťa automatické zobrazovanie nových hodnôt zo snímačov. Nastavenie ich rozsahu sa zapne pomocou **RANGE_CALIBRATION**, čím sa okrajové hodnoty postupne aktualizujú pohybom zariadenia. Výsledky sa potom musia prečítať pomocou sériovej komunikácie. Makrom **RESET_ON_ERROR** sa sprístupní automatické resetovanie snímačov pri akejkoľvek závažnej chybe, čo je absolútne nevyhnutné na plynulé používanie joysticku. Zmenou čísla pri **SENSITIVITY** sa zvolí požadovaná citlivosť zariadenia z troch možných úrovní. Jedná sa o prepísanie okrajových hodnôt pri zachovaní aktuálnej polohy. Prvá úroveň (1) rozšíri rozsah každej osi na dvojnásobok, čím sa zníži citlivosť. Na počítači je potom maximálne natočenie zariadenia zobrazené len do 50%. Opakom je druhá úroveň (2), kde sa rozsah zúži na polovicu, takže natočenie zariadenia na 50% odpovedá maximálnej hodnote na počítači. Makro funkcia **CutOff()** sa stará o zotrvanie v tomto rozmedzí. Tretia úroveň (0) nevykonáva žiadne zmeny, čím sa dosiahne normálna citlivosť.

```
#if SENSITIVITY == 1
  #define SENSITIVITY_FIX(sensor) (axisSize[sensor] / 2)
  #define SENS_MIN_MOD (4)
  #define SENS_MAX_MOD (2)
  #define CutOff(sensor)     \
  ({                         \
    /*EMPTY*/                \
  })
```

```
#elif SENSITIVITY == 2
  #define SENSITIVITY_FIX(sensor) ((-1) * (axisSize[sensor] / 4))
  #define SENS_MIN_MOD (3)
  #define SENS_MAX_MOD (1)
  #define CutOff(sensor)                                        \
    if (angleValue < SENSITIVITY_FIX(sensor))                   \
    {                                                           \
      angleValue = SENSITIVITY_FIX(sensor);                     \
    }                                                           \
    else if (angleValue > (-1) * SENSITIVITY_FIX(sensor))       \
    {                                                           \
      angleValue = (-1) * SENSITIVITY_FIX(sensor);              \
    }                                                           \
  })
#else
  #define SENSITIVITY_FIX(sensor) (0)
  #define SENS_MIN_MOD (0)
  #define SENS_MAX_MOD (0)
  #define CutOff(sensor)     \
  ({                         \
    /*EMPTY*/                \
  })
#endif
```

Keďže preprocesor neprijme príkazy podmienenej kompilácie **#if** vnútri makro funkcií, muselo sa toto obmedzenie obísť iným spôsobom, a to definovaním viacerých makier s rovnakým názvom ale rôznymi hodnotami, ktoré už bolo možné podmienkou vynechať z procesu kompilácie. Tieto makrá potom stačí vložiť na ich správne miesto. Nasledujúce dva bloky textu prijímajú hodnoty zvolené podmienkou z predošlého bloku. Ako je z nich vidno, rozsah natočenia sa zmení len v prípade, ak makrá **SENSITIVITY_FIX**, **SENS_MIN_MOD** a **SENS_MAX_MOD** nie sú nulové. Podobne na tom je aj funkcia **CutOff()**, ktorá keď je vyžadovaná prázdna, tak po zavolaní nemá žiadny vplyv na beh programu.

```
#define rangeChangeOutputMode_0(sensor) /*normal, zero-centered range*/  \
({                                                                       \
  axisMinActual[sensor] = axisMin[sensor] - axisMean[sensor];            \
  axisMaxActual[sensor] = axisMax[sensor] - axisMean[sensor];            \
  axisMinActual[sensor] -= SENSITIVITY_FIX(sensor);                      \
  axisMaxActual[sensor] += SENSITIVITY_FIX(sensor);                      \
})
```

Predošlý blok kódu sa používa na zmenu okrajových hodnôt počas normálneho režimu operácie s centrovanou základnou polohou v nule. Nasledujúci blok bol vytvorený len kvôli jednej počítačovej hre, v ktorej sa z nejakého dôvodu využíva len horná polovica celého rozsahu vstupu. To znamená, že pri symetrických limitoch maximum ostáva a stred sa zobrazuje ako minimum. Riešením je nesymetrické natiahnutie rozsahu dvomi makrami **SENS_MIN_MOD** a **SENS_MAX_MOD** a dodatočné posunutie minimálnej hodnoty.

```
#define rangeChangeOutputMode_1(sensor)  /*upper half of range is used*/      \
({                                                                            \
  axisMinActual[sensor] = axisMin[sensor]-axisMean[sensor]-axisSize[sensor];  \
  axisMaxActual[sensor] = axisMax[sensor]-axisMean[sensor];                   \
  axisMinActual[sensor] -= SENS_MIN_MOD * SENSITIVITY_FIX(sensor);            \
  axisMaxActual[sensor] += SENS_MAX_MOD * SENSITIVITY_FIX(sensor);            \
})
```

Jediným dôvodom, prečo sa citlivosť nastavuje pri kompilácii je to, že táto funkcia zariadenia sa navrhla až po vyrobení mechanizmu a všetkých DPS. Nebolo počítané s tým, že bude potrebné aj druhé ovládacie tlačidlo. Nastavenie citlivosti je našťastie len vedľajší prídavok a nie dôležitá súčasť zariadenia.

Ďalšími významnými makro funkciami závislými na podmienenej kompilácii sú **`JOYSTICK_SET_RANGE()`** a **`JOYSTICK_SET_AXIS()`**. Prvá nastaví okrajové hodnoty a druhá zas aktuálne natočenia. Ak sa používanie joysticku nevyžaduje, jadro týchto funkcií ostane prázdne, aby sa po ich volaní nič nestalo. Presnejšie povedané, na miesto volania v kóde sa nakopíruje komentár **`/*EMPTY*/`**, ktorého znaky preprocesor zamení za medzery a vynechá z kompilácie. Nie je sem síce nutné napísať vôbec nič, ale pomáha to lepšej prehľadnosti.

4.3 Makro funkcie

Makro je kus kódu, ktorý je nejako pomenovaný. Použitím tohto mena sa na dané miesto nakopíruje jeho obsah ešte pred samotnou kompiláciou kódu. Existujú dva typy makier. Prvý typ pripomína dátové objekty a používa sa na symbolické pomenovanie číselných konštánt. Druhým typom sú makro funkcie, ktoré sa správajú podobne ako pravé funkcie, čiže dokážu prijať aj argumenty. Pri nesprávnom zaobchádzaní s nimi môžu v kóde nastať zvláštne problémy, ale výhody práce s preprocesorom sa niekedy nedajú ignorovať. Kým na inline funkcie často ostanú odkazy, makro ho na seba v zásade nevytvorí. Zlučovanie textových reťazcov na vytvorenie názvu funkcie, ktorá sa hneď aj zavolá, je tu tiež možné a v tejto práci sa tým ušetrí veľký počet inštrukčných cyklov. Ak je nutné makro zapísať na viac riadkov, musí sa na ich koniec vložiť znak spätnej lomky „\“ [37].

Nasledujúce tri makro funkcie umožňujú najrýchlejšie spínanie pinov a registrov na Arduine. Po kompilácii sa v kóde v assembleri vytvorí len jediná inštrukcia s rovnaným názvom, ktorá trvá len dva cykly. Platí so samozrejme len vtedy, keď sa do argumentov zapíšu konštanty, a to názov portu a jeho číslo. Môžu síce byť uložené v poli, ale záleží len na preprocesore, či sa budú musieť tieto hodnoty v poli vyhľadávať. Ak je na úseku kódu vyžadovaná vyššia rýchlosť, nemali by sa používať v iteračných alebo v podmienených cykloch, kde sa požadovaný pin nepozná už pri kompilácii. Takto by bolo nutné dodatočné vyhľadávanie identifikátorov všetkých pinov z premenných v poli.

```
#ifndef sbi  // set bit
  #define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif
#ifndef cbi  // clear bit
  #define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef toggle  // toggle bit
  #define toggle(sfr, bit) (_SFR_BYTE(sfr) ^= _BV(bit))
#endif
```

Na zabránenie vloženia odkazu na funkciu **`SPI.transfer16()`** z knižnice Arduina pri každom pokuse o komunikáciu cez SPI sa musel celý jej kód nakopírovať do makra **`SPI_transfer16()`** a mierne upraviť. Ten funguje úplne rovnako a jej celý text je vždy bez odkazu vložený na miesto volania. Našťastie bolo v čipe ATmega32u4 na túto úpravu dostatok voľnej pamäte, aj keď sa daná makro funkcia volá veľmi často aj na viacerých miestach pre všetkých šesť snímačov. Ušetrilo sa ale veľké množstvo inštrukčných cyklov.

```
#define SPI_transfer16(data)                                                    \
({                                                                              \
  union { uint16_t val; struct { uint8_t lsb; uint8_t msb; }; } in, out;  \
  in.val = data;                                                                \
  SPDR = in.msb;                                                                \
  asm volatile ("nop");                                                         \
  while (!(SPSR & _BV(SPIF))) ;                                                 \
  out.msb = SPDR;                                                               \
  SPDR = in.lsb;                                                                \
  asm volatile ("nop");                                                         \
  while (!(SPSR & _BV(SPIF))) ;                                                 \
  out.lsb = SPDR;                                                               \
  out.val;                                                                      \
})
```

Po získaní novej hodnoty zo snímačov sa vždy skontroluje, či sa nachádza vnútri rozsahu danej osi. Keď je všetko v poriadku, makro funkcia **checkRange()** nespraví nič. Ak sa ale náhodou niektorá hodnota dostane mimo limitov, musí sa hranica **axisMin[]** alebo **axisMax[]** aktualizovať aj s veľkosťou rozsahu **axisSize[]** a základnou polohou **axisMean[]**. Tieto premenné priamo zodpovedajú hodnotám získaným zo snímačov. Čísla zobrazené na počítači majú rozsah centrovaný na nulu a ich limity **axisMinActual[]** a **axisMaxActual[]** sú prepočítané podľa režimu výstupu a citlivosti. Hodnota natočenia zo snímača **receivedValue[]** sa okrem posunutia na stred odčítaním základnej polohy **axisMean[]** nijak neupravuje, aby sa zachovala čo najvyššia presnosť.

Na zistenie poruchy alebo iného problému so snímačmi sa prijaté údaje ešte pred ďalším spracovaním musia overiť, či niektoré ich bity nehlásia nejaké chyby. Na to sa použije krátke makro **CHECK_ERROR()**, kde sa do argumentu zapíše prijaté 16-bitové číslo. Jeho výstup predstavuje logickú hodnotu, ktorá rozhoduje medzi ošetrením chyby alebo ďalšej práci s číslom.

```
#define CHECK_ERROR(x) (((x) & SPI_REC_ALARM_LO) | ((x) & SPI_REC_ALARM_HI) |
((x) & SPI_REC_ERRFLG))
```

Najdôležitejšia makro funkcia na komunikáciu so snímačmi je **SPI_send_loop()** a používa sa výhradne len v hlavnom cykle programu. Číslo v argumente presne identifikuje snímač, aby preprocesor dokázal ešte pred kompiláciou zvoliť všetky potrené premenné z polí. Ďalej sa zlúčia dva reťazce textu do jedného na vytvorenie názvu jednej zo šiestich **inline** funkcií, ktoré majú za úlohu získať hodnoty zo snímačov. Prvý reťazec **SPI_send_** sa pomocou operátora **##** spojí s číslom z argumentu a s druhým reťazcom **(SPIcommand)**. V podkapitole 4.5.1 je presnejší popis týchto funkcií. Po otestovaní chybových bitov sa pokračuje buď ošetrením poruchy alebo ďalším spracovaním hodnoty. Bitovým posunom a filtrom sa zo 16-bitového čísla získa výsledné natočenie, ktoré sa hneď aj skontroluje, či sa nachádza v požadovanom rozsahu. Potom sa ešte musí aktualizovať čas poslednej zmeny hodnoty ktoréhokoľvek snímača kvôli automatickému prechodu do spánkového režimu mikrokontroléru. V dôsledku prítomnosti šumu sa tu pridalo aj pásmo necitlivosti (dead-zone) v okolí priemernej hodnoty **receivedValueMean[]**. Ďalej nasledujú už len mierne úpravy podľa zvolených režimov výstupu a zapísanie čísla do premennej v triede Joystick.

Veľmi podobnou makro funkciou je aj **SPI_send_setup()**, ktorá sa spustí iba na začiatku programu po zapojení do počítača alebo resetovaní tlačidlom na Arduine. Má za úlohu nastaviť uložené okrajové hodnoty a v prípade poruchy sa vrátiť na začiatok funkcie, kde sa musí znova pripraviť prvý uhol na čítanie v hlavnom cykle. Jedine na tomto mieste

sa použila menej odporúčaná a niekedy priam nebezpečná funkcia **goto**. V tomto prípade ale žiadne problémy nenastali, ani keď ich identifikátory vznikli zlúčením troch textových reťazcov.

Kódy predošlých dvoch makro funkcií **SPI_send_loop()** a **SPI_send_setup()** sú príliš rozsiahle, takže sa spolu aj s mnohými inými z textu práce vynechali. Je však možné ich nájsť v digitálnej podobe na CD v prílohách.

4.4 Nastavenie Arduina

V tomto programe boli okrem štandardných knižníc Arduina použité aj dve dodatočné na sprístupnenie funkcie spánku pre mikrokontroléry triedy AVR, a to **sleep.h** a **power.h**. Na zaistenie komunikácie cez SPI sa sem zahrnula aj **SPI.h** a na emuláciu joysticku cez USB zas **Joystick.h**.

```
#include <avr/sleep.h>
#include <avr/power.h>
#include <SPI.h>
#include "Joystick.h"
```

Preprocesor sa niekedy rozhodne ignorovať navrhnuté kľúčové slovo **inline** pred definíciou funkcie, preto treba v týchto prípadoch na začiatok kódu pridať ich deklarácie spolu s vlastnosťou **__attribute__((always_inline))** za názvom. Pri funkciách, ktoré sprostredkujú komunikáciu, je tento postup nevyhnutný. Ak sa ešte ani po tomto pokuse o optimalizáciu kódu na miesto volania zapíše odkaz, musí sa upraviť jadro danej funkcie alebo zvoliť úplne iný postup, napríklad už zmienené makrá.

```
inline unsigned int SPI_send_0(unsigned int) __attribute__((always_inline));
inline unsigned int SPI_send_1(unsigned int) __attribute__((always_inline));
inline unsigned int SPI_send_2(unsigned int) __attribute__((always_inline));
inline unsigned int SPI_send_3(unsigned int) __attribute__((always_inline));
inline unsigned int SPI_send_4(unsigned int) __attribute__((always_inline));
inline unsigned int SPI_send_5(unsigned int) __attribute__((always_inline));
inline void sleepNow() __attribute__((always_inline));
```

V prípade potreby vypisovania polohy zariadenia alebo iných hodnôt na ladenie programu sa nastaví sériová komunikácia, ktorá v programovacom prostredí napodobňuje USART. Zvolená rýchlosť sa musí zhodovať v mikrokontroléri a aj na počítači. Pre istotu sa program zdrží na pol sekundy, aby tento modul mal dostatok času na správnu aktiváciu.

```
#if SERIAL_FEEDBACK
    Serial.begin(9600);
    delay(500);
#endif
```

Z knižnice **SPI.h** sa použila len jedna funkcia na zjednodušené nastavenie režimu SPI komunikácie. Ďalej sa musia v registroch správne nastaviť vstupné a výstupné piny, najlepšie makrami **sbi()** a **cbi()**, inak tento modul nebude fungovať, čo sa vo väčšine návodov zabudne spomenúť.

```
SPI.beginTransaction(SPISettings(500000, MSBFIRST, SPI_MODE1));
sbi(SPCR,4);  // set as Master
cbi(DDRB,3);  // set MISO as input
sbi(DDRB,2);  // set MOSI as output
sbi(DDRB,1);  // set SCLK as output
```

Trieda **Joystick** z knižnice **Joystick.h** obsahuje veľký počet premenných, ktoré v sebe udržiavajú rôzne vlastnosti emulovaného zariadenia. Taktiež sú tu mnohé funkcie na zjednodušenie ich spracovania. Najprv sa nastavia ich požadované hodnoty, potom sa funkciou **Joystick.sendState()** odošlú do počítača cez USB. Prvá inicializácia tejto komunikácie volaním **Joystick.begin()** rozhoduje o tom, či sa budú údaje odosielať manuálne alebo automaticky. Zapísaním **true** do argumentu sa do počítača dostanú nové hodnoty po každej aktualizácii ktorejkoľvek premennej. V tomto programe je komunikácia spustená vždy manuálne, aby sa zbytočne neplytvalo inštrukčnými cyklami, takže sa sem zapísalo **false**.

Existuje veľa rôznych metód na zistenie stavu tlačidiel. Správne by sa to ale malo riešiť pomocou externých prerušení, ktorých má Arduino Leonardo dokonca až päť. Na tomto zariadení sa nachádzajú len tri tlačidlá, takže nie je nutné pridávať žiadne dodatočné čipy. Vhodnou zmenou bitov v registri **EICRA** sa nastavilo spustenie prerušení **INT0**, **INT1** a **INT3** po zaregistrovaní klesajúcej hrany na pinoch **PD0**, **PD1** a **PD3**, spôsobené vybitím kondenzátora cez tlačidlo na DPS. Ďalej sa pre istotu vymažú indikátory prerušenia z **EIFR** registra a nakoniec sa v **EIMSK** tieto moduly zapnú.

```
sbi(EICRA, ISC01);  // falling edge generates an INT0 interrupt request
sbi(EICRA, ISC11);  // falling edge generates an INT1 interrupt request
sbi(EICRA, ISC31);  // falling edge generates an INT3 interrupt request
sbi(EIFR, INTF0);  // clear INT0 interrupt flag
sbi(EIFR, INTF1);  // clear INT1 interrupt flag
sbi(EIFR, INTF3);  // clear INT3 interrupt flag
sbi(EIMSK, INT0);  // enable INT0 interrupt
sbi(EIMSK, INT1);  // enable INT1 interrupt
sbi(EIMSK, INT3);  // enable INT3 interrupt
```

Arduino vždy po svojom spustení automaticky zapne 8-bitový časovač na počítanie uplynulého času v milisekundách, ktorého hodnota sa získa funkciou **millis()**. Návratový typ tejto premennej je **unsigned long**, ktorý pretečie približne až o 50 dní. Správnym použitím pri logickom porovnávaní toto problémy robiť nebude. Treba sa len uistiť, aby odčítanie predošlej hodnoty času od novej bolo umiestnené na rovnakej strane logického operátora. Týmto spôsobom nezáleží na pretečení bezznamienkových dátových typov v podobných výpočtoch, aké sa ukážu v ďalších podkapitolách.

4.5 Hlavný cyklus programu

Po nastavení všetkých premenných a inicializácii potrebných periférií sa program dostane do nekonečného cyklu **loop()**, v ktorom sa vykonáva komunikácia cez SPI, spracovanie signálov z tlačidiel a odosielanie ich hodnôt do počítača. Ďalej sa tu sleduje čas poslednej aktivity zariadenia a v prípade jeho vypršania sa prejde do režimu spánku. Na podobnom princípe sa na konci hlavného cyklu zaisťuje periodické spínanie vstavaného LED svetla každú pol sekundu kvôli monitorovaniu behu programu.

4.5.1 Komunikácia cez SPI

So snímačmi sa komunikuje pomocou 16-bitových príkazov odoslaných z mikrokontroléru cez SPI a súčasne s nimi sú prijaté aj hodnoty natočenia. Frekvencia tejto činnosti bola nastavená na 100 Hz, čím sa dosiahne dostatočne rýchla odozva. Maximálna frekvencia sa nájsť neskúšala, pretože zariadenie už zvládalo komunikáciu s podobnou odozvou akú majú súčasné ovládacie zariadenia a vyššie hodnoty vyžadujú oveľa viac výpočtovej sily na počítači.

Nasledujúci blok kódu z hlavného cyklu má za úlohu získať všetkých šesť hodnôt natočenia zo snímačov. Na začiatku sa skontroluje, či už prešlo dosť času od poslednej komunikácie, potom sa zvolí príkaz na čítanie uhlu. Ak pin **PD2** ukazuje logickú nulu, čo predstavuje prvý snímač pripravený na prijatie ďalšieho príkazu, môžu sa začať preberať nové hodnoty. Tento pin zodpovedá pinu prerušeniu **INT/** iba na prvom snímači, pretože sa v každom cykle získavajú uhly zo všetkých šiestich do radu za sebou, takže nie je potrebné sledovať aj zvyšných päť. Podľa čísla v makre **ACTIVE_SENSOR_NUM**, čo predstavuje počet použitých snímačov, vie kompilátor nakopírovať všetky potrebné bloky kódu obsiahnuté v predošlých makro funkciách s veľmi malým množstvom odkazov. Väčšina riadkov tohto programu sa nachádza práve tu. V prípade, že sa vyžaduje vypisovanie prijatých hodnôt cez emulovaný USART, sa bezprostredne po ich získaní odošlú do počítača.

```
static const unsigned long spiInterval = 10;  // fetch interval (milliseconds)
static unsigned long spiPreviousMillis = 0;   // last time data was fetched
unsigned long spiCurrentMillis = millis();
if (spiCurrentMillis - spiPreviousMillis >= spiInterval)
{
  spiPreviousMillis = spiCurrentMillis;
  usbNewData = 1;
   // SPI Communication
  SPIcommand = SPI_CMD_ANGLE;  // ANGULAR DATA command
  if ((PIND & _BV(PD2)) == 0)  // INT/ is low ... (digitalRead(INTpin) == LOW)
  {
    if (ACTIVE_SENSOR_NUM >= 1) SPI_send_loop(0);
    if (ACTIVE_SENSOR_NUM >= 2) SPI_send_loop(1);
    if (ACTIVE_SENSOR_NUM >= 3) SPI_send_loop(2);
    if (ACTIVE_SENSOR_NUM >= 4) SPI_send_loop(3);
    if (ACTIVE_SENSOR_NUM >= 5) SPI_send_loop(4);
    if (ACTIVE_SENSOR_NUM >= 6) SPI_send_loop(5);

    #if SERIAL_FEEDBACK
      #if SERIAL_PRINT_ALL_ANGLES
        Serial.print("\n");
        Serial.println(receivedValue[0]);
        Serial.println(receivedValue[1]);
        Serial.println(receivedValue[2]);
        Serial.println(receivedValue[3]);
        Serial.println(receivedValue[4]);
        Serial.println(receivedValue[5]);
      #endif
    #endif
  }
}
```

Jedna zo šiestich **inline** funkcií na získanie 16-bitovej hodnoty zo snímačov je **SPI_send_0()**. Líšia sa len v odkazoch na premenné v poliach so slave select pinmi. Tento krátky kus kódu by sa síce mohol priamo zapísať aj do makro funkcií, ale týmto spôsobom sa v prípade potreby dajú jednoducho vykonať malé úpravy na jednotlivých snímačoch. Nie je vôbec problém kombinovať viac typov programovacích štýlov a navyše sa takto aj odskúšajú rôzne metódy prístupu k premenným. V nasledujúcej funkcii sa ešte počas kompilácie zvolí slave select pin z polí **SSport[]** a **SSnum[]**. Ich indexom je číslo od nuly do päť, čím sa pri kompilácii zaistí priamy odkaz na položku v poli bez hľadania počas behu programu. Zvolený pin sa nastaví na logickú nulu, čo signalizuje začiatok prenosu dát niektorého snímača. Ďalej sa do lokálnej premennej **rec_val** zapíše hodnota

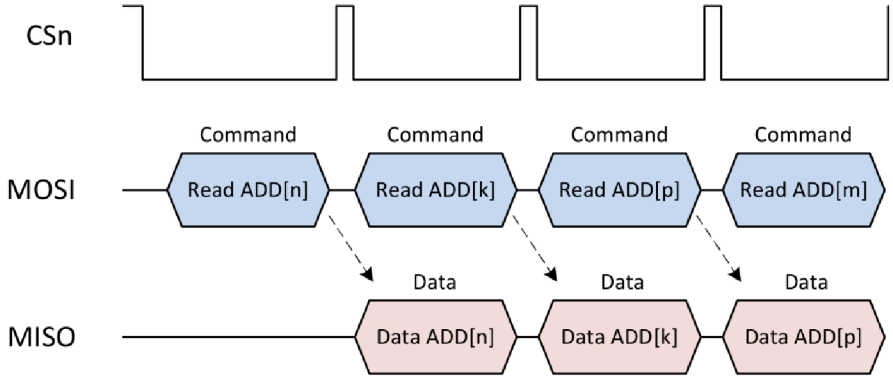
získaná pomocou makra **SPI_transfer16()**, slave select pin sa vráti späť na logickú jednotku a výstupom funkcie sa stáva **rec_val**.

```
inline unsigned int SPI_send_0(unsigned int SPIcommand)
{
  cbi(*SSport[0], SSnum[0]);
  unsigned int rec_val = SPI_transfer16(SPIcommand);
  sbi(*SSport[0], SSnum[0]);
  return rec_val;
}
```

Keby sa v prijatej hodnote objavili chybové indikátory na niektorom z troch bitov, zavolá sa funkcia **SPI_error_handle()** na zistenie príčiny. Argumenty sú: číslo snímača, získané údaje o chybe a smerník na funkciu, ktorá sa má použiť na komunikáciu. Najprv sa otestujú dva bity signalizujúce intenzitu magnetického poľa, potom sa v prípade logickej jednotky tretieho indikátora vyžiada kompletné chybové hlásenie. Ak sa v kóde nastavilo automatické resetovanie snímačov, na konci funkcie sa odošle príkaz Master Reset.

```
void SPI_error_handle(byte sensor, unsigned int recVal, unsigned int
(*f_send)(unsigned int))
```

Princíp čítania uhla je ilustrovaný na Obr. 4.1. Prvým príkazom sa najprv pošle žiadosť o hodnotu, ktorá sa potom prijme počas odosielania druhého. Čiže vyžiadané informácie sú vždy získané až v nasledujúcom komunikačnom cykle. Na obrázku skratka CSn znamená to isté ako slave select niektorého snímača. Na mierne zníženie odozvy zariadenia by sa namiesto čítania všetkých šiestich hodnôt do radu až v nasledujúcom cykle mohli odosielať dvojice povelov hneď za sebou. Druhým príkazom by bol NOP, ktorý nevykoná žiadnu činnosť, ale pritom sprostredkuje údaje z predošlej komunikácie. Takto by sa v podstate zdvojnásobilo využitie rozhrania SPI. Po prvých testoch s hotovým zariadením ale nebolo nutné znížiť odozvu.



Obr. 4.1: Čítanie uhla cez SPI [25].

4.5.2 Obsluha tlačidiel

Pri návrhu spracovania signálov z tlačidiel sa rozhodlo, že doba reakcie na stlačenie musí byť čo najkratšia. To sa dá zaistiť jedine prerušeniami pri zmene stavu na určitých pinoch. Nasledujúca funkcia sa zavolá iba po stlačení pravého tlačidla na rukoväti rotačnej časti mechanizmu. Na začiatku sa vypne toto konkrétne prerušenie, aby sa zbytočne nevolalo znova niekoľko krát, kým sa signál na pine neustáli. Premenná **buttonRightState** potom zaznamená stlačenie, aby sa v hlavnom cykle mohli vykonať potrebné procesy na obsluhu. Keďže sa v tomto programe nachádza aj režim spánku, musí sa pomocou **awakened** zistiť, či mikrokontrolér spal tesne pred vstupom do tejto funkcie. Ak áno, **awakenedByLR** dá

vedieť hlavnému cyklu, že sa nemá vykonať nič okrem ošetrenia odrazov v signáli. Inak sa aktualizuje stav tlačidla v triede **Joystick** a premennou **usbNewData** sa naznačí potreba odoslania nových údajov do počítača cez USB. Na konci funkcie sa uloží čas posledného odrazu, a teda aj stlačenia, do 32-bitovéhej premennej **lastBounceTimeRight**.

```
ISR(INT1_vect)
{
  cbi(EIMSK, INT1);  // disable INT1 interrupt
  buttonRightState = 1;  // Right Button was pressed
  if (awakened == 0)  // CPU was NOT sleeping
  {
    usbNewData = 1;
    #if USE_JOYSTICK
      Joystick.pressButton(jButtonRight);
    #endif
  }
  else
  {
    awakenedByLR = 1;
  }
  lastBounceTimeRight = millis();
}
```

Všetky časovo náročné príkazy na obsluhu sa nachádzajú v hlavnom cykle. Najprv sa z premennej **buttonRightState** musí zistiť, či bolo tlačidlo vôbec stlačené. Ďalej sa sleduje aktuálny stav pinu **PD1**, až kým nebude toto tlačidlo pustené. Inak sa ešte naďalej predpokladá jeho pridržiavanie alebo prítomnosť odrazov a **lastBounceTimeRight** je preto aktualizované. Po pustení tlačidla musí prejsť dostatočne dlhá doba od posledného odrazu, presnejšie doba **debounceDelay**, aby sa konečne mohla začať vykonávať samotná obsluha. Premenná **buttonRightState** sa vynuluje a v prípade, že mikrokontrolér pred stlačením tlačidla nespal, v triede **Joystick** sa aktualizuje súčasný stav a **usbNewData** zas signalizuje potrebu komunikácie s počítačom. Obsluha končí vymazaním indikátora tohto prerušenia a jeho opätovným zapnutím. Až po týchto dvoch inštrukciách je možné znova zaregistrovať ďalšie stlačenie tlačidla. Hodnota **debounceDelay** sa nastavila na 10 ms.

```
if (buttonRightState == 1)  // Right Button was pressed
{
  unsigned long currentMillisRight = millis();
  noActivityPreviousMillis = currentMillisRight;  // refresh activity timeout
  if ((PIND & _BV(PD1)) != 0)  // Right Button is released
  {
    if ((currentMillisRight - lastBounceTimeRight) > debounceDelay)
    // debounced
    {
      buttonRightState = 0;
      if (awakenedByLR == 1)
      // CPU was sleeping and was awakened by Left or Right Button
      {
        awakenedByLR = 0;
      }
      else
      {
        usbNewData = 1;
```

```
        #if USE_JOYSTICK
          Joystick.releaseButton(jButtonRight);
        #endif
      }
      sbi(EIFR, INTF1);  // clear INT1 interrupt flag
      sbi(EIMSK, INT1);  // enable INT1 interrupt
    }
  }
  else
  {
    lastBounceTimeRight = currentMillisRight;  // reset lastBounceTime
  }
}
```

Pravé a ľavé tlačidlo majú veľmi podobnú štruktúru obsluhy, ktorá sa od tretieho líši hlavne možnosťou uspania mikrokontroléru a prepínania režimov výstupu. Krátkym kliknutím sa zavolá funkcia **sleepNow()** na spánok a pridržaním na 500 milisekúnd sa zmení režim výstupných hodnôt z centrovaného na hornú polovicu rozsahu.

4.5.3 Ostatné

Prepnutie mikrokontroléru do režimu spánku sa vykonáva **inline** funkciou **sleepNow()**. So zapnutým modulom joysticku sa na začiatku vycentrujú limity a natočenia tak, aby sa na počítači zobrazili všetky osi v presnom strede. Ďalej sa vypne vstavané LED svetlo na Arduine, časovač na počítanie milisekúnd a taktiež prerušenia od USB komunikácie, kde sa ešte musí uložiť vyrovnávacia pamäť. Tesne pred spánkom sa znova zapne možnosť prerušenia na treťom tlačidle, aby sa jej stlačením mohol čip zobudiť. Zo spánkového režimu sa mikrokontrolér dostane kliknutím na ktorékoľvek tlačidlo. Potom sa všetky nastavenia zmenené pred spánkom vrátia späť do pôvodného stavu, až na predošlú hodnotu natočenia, ktorá už pravdepodobne nie je aktuálna. Tá sa získa v ďalšom komunikačnom cykle SPI.

Odoslanie nových údajov do počítača sa rieši v podobe žiadostí o komunikáciu. Ak sa nastaví premenná **usbNewData** na logickú jednotku, v hlavnom cykle sa zavolá funkcia **Joystick.sendState()**, ktorá odošle všetky hodnoty používaných osí a tlačidiel. Pri získaní nových natočení zo snímačov sa automaticky požiada o komunikáciu v rovnakých časových intervaloch. Zmenou stavu tlačidiel sa tiež podá žiadosť, takže netreba čakať na snímače.

```
#if USE_JOYSTICK
  if (usbNewData == 1)
  {
    usbNewData = 0;
    Joystick.sendState();  // Send Axis Values, Ranges and Button States
  }
#endif
```

Ešte pred koncom cyklu sa kontroluje čas od poslednej aktivity zariadenia. Keď sa za posledných 5 minút nestlačilo žiadne tlačidlo alebo sa nepohlo ani s jednou rukoväťou, prejde sa do režimu spánku kvôli šetreniu energie. Premenná **buttonModeState** sa nastaví na logickú jednotku preto, aby sa napodobnil spôsob prechodu do režimu spánku stlačením tretieho tlačidla, ktorého obsluha v hlavnom cykle dokáže spracovať premennú **awakened** vytvorenú funkciou spánku **sleepNow()**. V podstate tu ani nezáleží na tom, čo spôsobilo prechod do tohto režimu. Koncový efekt je v obidvoch prípadoch rovnaký.

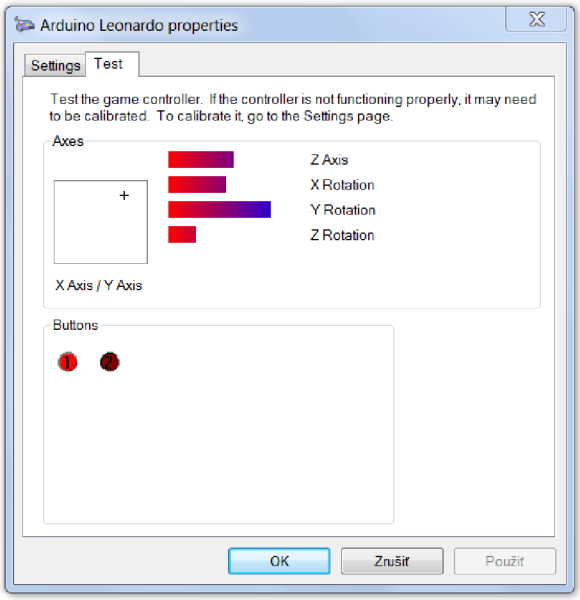
```
static const unsigned long noActivityInterval = 5*60*1000;
// time until sleep (milliseconds)
unsigned long noActivityCurrentMillis = millis();
if (noActivityCurrentMillis - noActivityPreviousMillis >= noActivityInterval)
{
  noActivityPreviousMillis = noActivityCurrentMillis;
  buttonModeState = 1;  // as if Mode Button was pressed (from sleep mode)
  sleepNow();
}
```

Posledným blokom kódu v hlavnom cykle sa periodicky bliká s LED svetlom na Arduine, aby sa vedelo, kedy mikrokontrolér pracuje, kedy spí a či sa náhodou nezasekol. Spínanie svetla nastáva presne každých 500 milisekúnd a počet potrebných inštrukcií je minimálny.

```
static const unsigned long ledInterval = 500;  // blink interval (milliseconds)
static unsigned long ledPreviousMillis = 0;  // last time LED was updated
unsigned long ledCurrentMillis = millis();
if (ledCurrentMillis - ledPreviousMillis >= ledInterval)
{
  ledPreviousMillis = ledCurrentMillis;
  PORTC ^= _BV(PC7);  // toggle LED_BUILTIN
}
```

5 SKÚŠKA FUNKČNOSTI HOTOVÉHO ZARIADENIA

Zasunutím USB kábla do počítača sa automaticky nainštalujú potrebné ovládače tohto plug and play zariadenia, čím sa rozpozná, že sa jedná o joystick so šiestimi osami a dvomi tlačidlami. V ovládacom paneli na operačnom systéme Windows 7 sa medzi ostatnými ovládacími zariadeniami zobrazí aj Arduino Leonardo a po preklikaní k nastaveniam sa dá dostať do okna s testovaním jeho funkcií, ako je vidno na Obr. 5.1. Ľavé tlačidlo s číslom 1 je na tomto obrázku práve stlačené. Názvy osí zodpovedajú označeniam na mechanizme na Obr. 2.3 a Obr. 2.14. Tento postup bol zopakovaný aj na druhom počítači, kde aj napriek zlyhaniu inštalácie ovládača sa dal joystick plne používať. Problém nebol so zariadením, ale so stratou povolení k prístupu k zložke so všetkými ovládačmi operačného systému.



Obr. 5.1: Testovanie vlastností zariadenia.

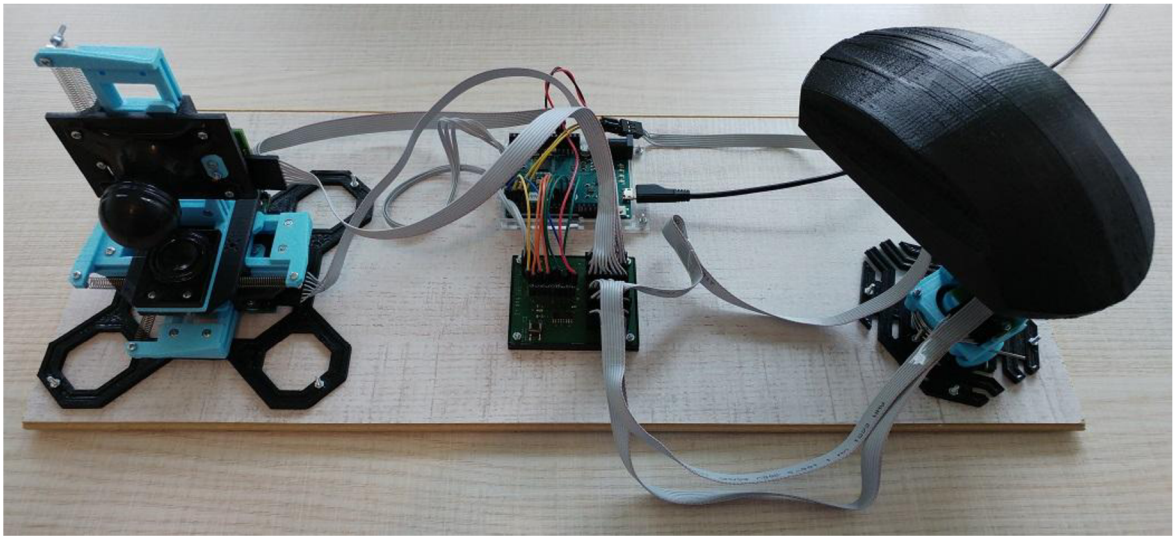
3D joystick bol odskúšaný v štyroch počítačových hrách. Prvou bol simulátor vesmírneho programu Kerbal Space Program s možnosťou priameho ovládania vesmírnej lode v šiestich stupňoch voľnosti. Práve tu sa musí nevyhnutne zmeniť režim výstupných hodnôt na využitie hornej polovice rozsahu. Keďže všetky pohyby v hre sa berú ako miery zrýchlenia, ovládanie ľahkých plavidiel s týmto zariadením nie je príliš užitočné. Každý malý pohyb s rukou spôsobí prudké otáčanie vesmírnej lode, ktorá sa za krátku chvíľu stane úplne nekontrolovateľnou. Keby miera natočenia rukoväte odpovedala rýchlosti a nie zrýchleniu, tak by 3D joystick bol vhodným ovládacím zariadením k tejto hre. Simulátor Take On Mars, ktorý sa odskúšal ako druhý, má podobný štýl ovládania zrýchlenia ako predošlá hra a v podstate rovnaké nedostatky.

V tretej hre X Rebirth, kde sa riadi len jedna malá vesmírna loď, je ovládanie už riešené pomocou voliteľnej rýchlosti rotácie v závislosti od natočenia rukoväte. Podobne to funguje aj s posuvným pohybom. V tomto simulátore je ovládanie 3D joystickom veľmi príjemné, a to hlavne v okolí vesmírnych staníc, kde sa jemnými pohybmi dá vynikajúco manévrovať medzi rôznymi výbežkami bez kolízií. Taktiež tu je možné presne prispôsobiť svoju rýchlosť k inému plavidlu a byť pritom k nemu vždy nasmerovaný, čo je s použitím tlačidiel veľmi ťažké. V podstate toto zariadenie robí z obyčajného poletovania pomerne zaujímavú činnosť.

Štvrtým simulátorom bola jedna staršia letecká hra Echelon z roku 2001. Podporuje síce iba ovládanie pre dva analógové vstupy, ktoré aplikácia zvolí z prvých dvoch osí posunutia X a Y a nie z natočení, ale aj tu bolo zariadenie použiteľné. Výmenou týchto dvoch osí by sa dosiahlo podobného efektu ako od klasických joystickov. Keby sa do programu mikrokontroléru navyše integrovala aj knižnica na emuláciu klávesnice, mohli by sa nastaviť zvyšné osi na signály zvolených tlačidiel.

Aplikácia Google Earth tiež podporuje tento 3D joystick na navigáciu. Musela sa ale použiť jej staršia verzia 7.1.8.3036, pretože v novšej sa z nejakého dôvodu nerozoznalo pripojené ovládacie zariadenie. Taktiež tu nie je implementované žiadne pásmo necitlivosti vstupných hodnôt, čím sa pohyb vo všetkých piatich osiach nikdy nezastaví. Aj napriek týmto malým nedostatkom sa tu dá kvalitne odreagovať pri ľahkom poletovaní po planéte.

Fotku hotového zariadenia je možné pozorovať na Obr. 5.2. Kvôli veľmi nízkej váhe ich podstáv sa obidve časti mechanizmu museli pripevniť o ťažkú základňu, v tomto prípade o kus drevenej parkety dĺžky približne pol metra a šírky 20 cm. Na spodnú stranu sa ďalej nalepili štyri malé protišmykové podložky. Profil parkety sa navyše veľmi podobá povrchu stola, na ktorom bude používaný. Lineárna časť je uložená na ľavo kvôli zvyku ovládania posuvných pohybov tlačidlami na tej istej strane klávesnice. Rotačná časť sa musela umiestniť mierne natočená smerom na stred, aby ju bolo možné pohodlne držať. Na uchytenie lineárnej časti stačia len prsty, takže smer montáže sa ostal priamy, čo pomáha intuitívnemu ovládaniu. V strednej časti sa nachádza Arduino Leonardo a pod ním zas DPS rozbočovača pre SPI. Obidve dosky sú pevne spojené s parketou. Už len z takého veľkého množstva kabeláže je zrejmé, že využitie parkety pomáha s udržiavaním celého zariadenia pohromade.



Obr. 5.2: Hotové zariadenie.

ZÁVER

Hlavný cieľ tejto práce, číže návrh 3D joysticku so šiestimi stupňami voľnosti, bol úspešne splnený. Navrhol sa virtuálny model mechanizmu, elektronika na obsluhu snímačov a riadiaci program mikrokontroléru na komunikáciu s počítačom. Dokonca sa podarilo toto zariadenie aj vyrobiť a odskúšať v reálnych aplikáciách, čo už bolo nad rámec požiadaviek. Výsledkom je hotový a funkčný 3D joystick v štádiu vývoja laboratórnej vzorky schopný nahradenia rôznych iných ovládacích zariadení. Jeho použitie môže byť na niektoré účely ešte aj vhodnejšie. Najväčší úžitok z neho majú počítačové simulátory vesmírnych lodí, kde vstupné hodnoty odpovedajú rýchlostiam pohybov.

Na začiatku sa plánovalo navrhnúť mechanizmus v jednom celku, čo sa ale zdalo byť nerealizovateľné kvôli metóde výroby súčiastok. Domáca 3D tlačiareň nedokázala vyrobiť dostatočne presné kusy plastu, čo sa prejavilo hlavne v zasekávaní lineárnej časti. Preto sa celý mechanizmus rozdelil na dva kusy, každý schopný pohybu v troch stupňoch voľnosti, kde jeden z nich zachytával iba natočenia a druhý len posunutia. Týmto riešením sa tiež čiastočne predišlo väčšine problémov s nežiaducimi pohybmi v ostatných smeroch pri bežnom používaní. Ovládanie objektov na šiestich osiach jedinou rukou je síce lákavé, ale v praxi to nemusí viesť k príliš dobrým výsledkom. Môžu nastať problémy v prepojení pohybov ruky s vnímaným objektom z dôvodu zmyslového preťaženia v mozgu.

Možné vylepšenie mechanizmu spočíva v nahradení plastového posuvného ložiska precíznym kovový alebo kolieskovým, ktorý má omnoho nižšie trenie a kde nedochádza k častému zasekávaniu. Na rotačnej časti by sa taktiež mohli vymeniť súčasné guľôčkové ložiská za kvalitnejšie a menšie kusy, aby sa v nich nevyskytovala nadmerne veľká vôľa. Navrhnuté pružiny na lineárnej časti sa nepodarilo vyrobiť, ale v katalógu sa našiel veľmi podobný, aj keď o niečo slabší model, ktorý síce sprostredkuje silovú odozvu na pohyby, ale kvôli treniu nedokáže presne nastaviť rukoväť do základnej polohy. Rozmery torzných pružín na rotačnej časti sa mierne líšia kvôli vysokej náročnosti ich výroby.

Za použité snímače sa zvolili magnetické absolútne enkodéry, ktorých hodnoty sa získavali pomocou SPI rozhrania. Dohromady ich bolo až šesť, čiže jeden na každú os, a všetky sa zapojili na ten istý rozbočovač. Na rukoväť rotačnej časti boli tiež pridané aj dve tlačidlá podobné klasickým počítačovým myšiam. Navrhli sa až štyri typy dosiek plošných spojov, a to jeden na rozbočovač, jeden na tlačidlá a dva na snímače.

Tieto DPS sú potom zapojené na vývojovú dosku Arduino Leonardo, ktorý dokáže komunikovať s počítačom priamo cez rozhranie USB. Tam sa 3D joystick zobrazí ako herné ovládacie zariadenie so šiestimi osami a dvomi tlačidlami. Dokonca nie je potrebná ani inštalácia špeciálnych ovládačov alebo iného softvéru. Odozva na pohyby je dostatočne rýchla na plynulé používanie pri frekvencii 100 prerušení za sekundu. Jedinou nevýhodou je malý počet tlačidiel. Potreba pridania ďalších sa zistila až počas testovania programu po výrobe mechanizmu a DPS. Z praktického hľadiska by nebol problém pridať ľubovoľný počet nových tlačidiel, ale maximum bez použitia dodatočných logických čipov je len 7 kvôli nedostatku voľných pinov na Arduine.

V budúcnosti by sa dalo zakladať na tejto práci pri návrhu nového ovládacieho zariadenia so šiestimi stupňami voľnosti. Mechanizmus by sa znova skladal z dvoch častí, teraz už ale vyrobený bez 3D tlačiarne. Snímače by sa mohli zmenšiť, kabeláž zjednodušiť a mikrokontrolér by sa zvolil primerane na daný účel. Druhou zaujímavou možnosťou je navrhnúť takéto zariadenie za čo najnižšiu cenu, keďže dnešné dostupné modely sú drahé.

ZOZNAM POUŽITÝCH ZDROJOV

[1] The DLR SpaceMouse (1981–1993). *DLR - Institute of Robotics and Mechatronics* [online]. [cit. 2018-03-18]. Dostupné z: <http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-9467/16255_read-8998/>.

[2] FRÖHLICH, B. et al. On 3D input devices. *IEEE Computer Graphics and Applications* . 2006. Vol. 26, no. 2, s. 15–19.

[3] FROEHLICH, B. The Quest for Intuitive 3D-Input Devices. *ResearchGate* . 2005. .

[4] Accessories | Oculus. *Oculus* [online]. [cit. 2018-03-18]. Dostupné z: <https://www.oculus.com/accessories/>.

[5] Vive | Controller. *Vive* [online]. [cit. 2018-03-18]. Dostupné z: <https://www.vive.com/us/accessory/controller/>.

[6] 2 Axis Joystick. *RobotShop* [online]. [cit. 2018-03-19]. Dostupné z: <https://www.robotshop.com/en/2-axis-joystick.html>.

[7] 3 Axis Joystick w/ Button. *RobotShop* [online]. [cit. 2018-03-19]. Dostupné z: <https://www.robotshop.com/en/3-axis-joystick-w-button.html>.

[8] Extreme 3D Pro Joystick. *Logitech* [online]. [cit. 2018-03-19]. Dostupné z: <https://www.logitechg.com/en-us/product/extreme-3d-pro-joystick>.

[9] X56 H.O.T.A.S. *Logitech* [online]. [cit. 2018-03-20]. Dostupné z: <https://www.logitechg.com/en-us/product/x56-space-flight-vr-simulator-controller>.

[10] SpaceBall 1003. *Spacemice* [online]. [cit. 2018-03-20]. Dostupné z: <http://spacemice.org/index.php?title=Spaceball_1003>.

[11] SpaceOrb 360. *Spacemice* [online]. [cit. 2018-03-20]. Dostupné z: <http://spacemice.org/index.php?title=SpaceOrb360>.

[12] SpaceBall 5000. *Spacemice* [online]. [cit. 2018-03-20]. Dostupné z: <http://spacemice.org/index.php?title=Spaceball_5000>.

[13] SpaceMouse Compact. *3Dconnexion* [online]. [cit. 2018-03-22]. Dostupné z: <https://www.3dconnexion.com/spacemouse_compact/en/>.

[14] SpaceController. *SpaceControl* [online]. [cit. 2018-03-22]. Dostupné z: <https://spacecontrol.de/produkte/>.

[15] Lexip 3D Pro. *Indiegogo* [online]. [cit. 2018-03-22]. Dostupné z: <https://www.indiegogo.com/projects/revolutionary-gaming-mouse-2-internal-joysticks-technology#/>.

[16] Lexip 3D Pro. *Spacemice* [online]. [cit. 2018-03-22]. Dostupné z: <http://spacemice.org/index.php?title=Lexip3d>.

[17] RIDDEN, P. 3D-Spheric-Mouse. *New Atlas* [online]. [cit. 2018-03-22]. Dostupné z: <https://newatlas.com/axsotic-3d-spheric-mouse-precision-digital-object-manipulation/16832/>.

[18] Tesseract. *BackTesseract* [online]. [cit. 2018-03-23]. Dostupné z: <http://backtesseract.com/>.

[19] Wing. *Worthington Sharpe* [online]. [cit. 2018-03-23]. Dostupné z: <http://www.worthingtonsharpe.com/index.html>.

[20] Call for loan of one Wing Ground Control Station (GCS). *euRobotics* [online]. [cit. 2018-03-23]. Dostupné z: <https://www.eu-robotics.net/robotics_league/erl-emergency/about/call-for-loan-of-one-wing-ground-control-station-gcs.html>.

[21] Anet A6 3D Printer. *Shenzhen Anet Technology Co.,Ltd* [online]. [cit. 2018-03-28]. Dostupné z: <http://www.anet3d.com/English/3D_Printer/105.html>.

[22] GIANG, K. PLA vs. ABS. *3D Hubs* [online]. [cit. 2018-03-28]. Dostupné z: <https://www.3dhubs.com/knowledge-base/pla-vs-abs-whats-difference>.

[23] TAKAGISHI, K. - UMEZU, S. Development of the Improving Process for the 3D Printed Structure. *Scientific Reports* . 2017. Vol. 7.

[24] ABS Bed Adhesion Tips & Tricks. *MatterHackers* [online]. [cit. 2018-03-28]. Dostupné z: <https://www.matterhackers.com/articles/printing-tips-&-tricks:-abs-bed-adhesion>.

[25] AS5050A Rotary Sensor. *ams AG* [online]. [cit. 2018-03-17]. Dostupné z: <http://ams.com/eng/Products/Magnetic-Position-Sensors/Angle-Position-On-Axis/AS5050A>.

[26] AS5055A Position Sensor. *ams AG* [online]. [cit. 2018-03-17]. Dostupné z: <http://ams.com/eng/Products/Magnetic-Position-Sensors/Angle-Position-On-Axis/AS5055A>.

[27] AS5000-MD6H-3 Magnet. *ams AG* [online]. [cit. 2018-04-09]. Dostupné z: <http://ams.com/eng/Products/Magnetic-Position-Sensors/Magnets/AS5000-MD6H-3>.

[28] Neodym disk, diametrálna magnetizácia. *SELOS* [online]. [cit. 2018-04-09]. Dostupné z: <https://www.magnety.sk/neodymy/neodym-disk-diametralna-magnetizacia/>.

[29] Arduino Leonardo with Headers. *Arduino* [online]. 2017. [cit. 2018-03-17]. Dostupné z: <https://store.arduino.cc/arduino-leonardo-with-headers>.

[30] ATmega32U4. *Microchip* [online]. [cit. 2018-04-10]. Dostupné z: <https://www.microchip.com/wwwproducts/en/atmega32u4>.

[31] Arduino Leonardo ATmega32u4 with headers. *Pi Supply* [online]. [cit. 2018-04-01]. Dostupné z: <https://uk.pi-supply.com/products/arduino-leonardo-atmega32u4-headers>.

[32] WILLIAMS, E. Embed with Elliot: Debounce your Noisy Buttons, Part I. *Hackaday* [online]. [cit. 2018-04-11]. Dostupné z: <https://hackaday.com/2015/12/09/embed-with-elliot-debounce-your-noisy-buttons-part-i/>.

[33] 74HC4050D datasheet. *NXP Semiconductors* [online]. [cit. 2018-04-12]. Dostupné z: <https://www.nxp.com/docs/en/data-sheet/74HC4050.pdf>.

[34] Kerbal Kontrol Board pt. 4: Hello Arduino. *Tech-Taught* [online]. [cit. 2018-04-13]. Dostupné z: <https://techtaught.wordpress.com/2014/09/12/kerbal-kontrol-board-pt-4-hello-arduino/>.

[35] Arduino Joystick Library. *GitHub* [online]. [cit. 2018-04-16]. Dostupné z: <https://github.com/MHeironimus/ArduinoJoystickLibrary>.

[36] GHOSH, K. Writing Efficient C and C Code Optimization. *CodeProject* [online]. [cit. 2018-04-17]. Dostupné z: <https://www.codeproject.com/Articles/6154/Writing-Efficient-C-and-C-Code-Optimization>.

[37] STALLMAN, R.M. - WEINBERG, Z. The C Preprocessor. *Free Software Foundation, Inc.* [online]. [cit. 2018-04-18]. Dostupné z: <https://gcc.gnu.org/onlinedocs/cpp.pdf>.

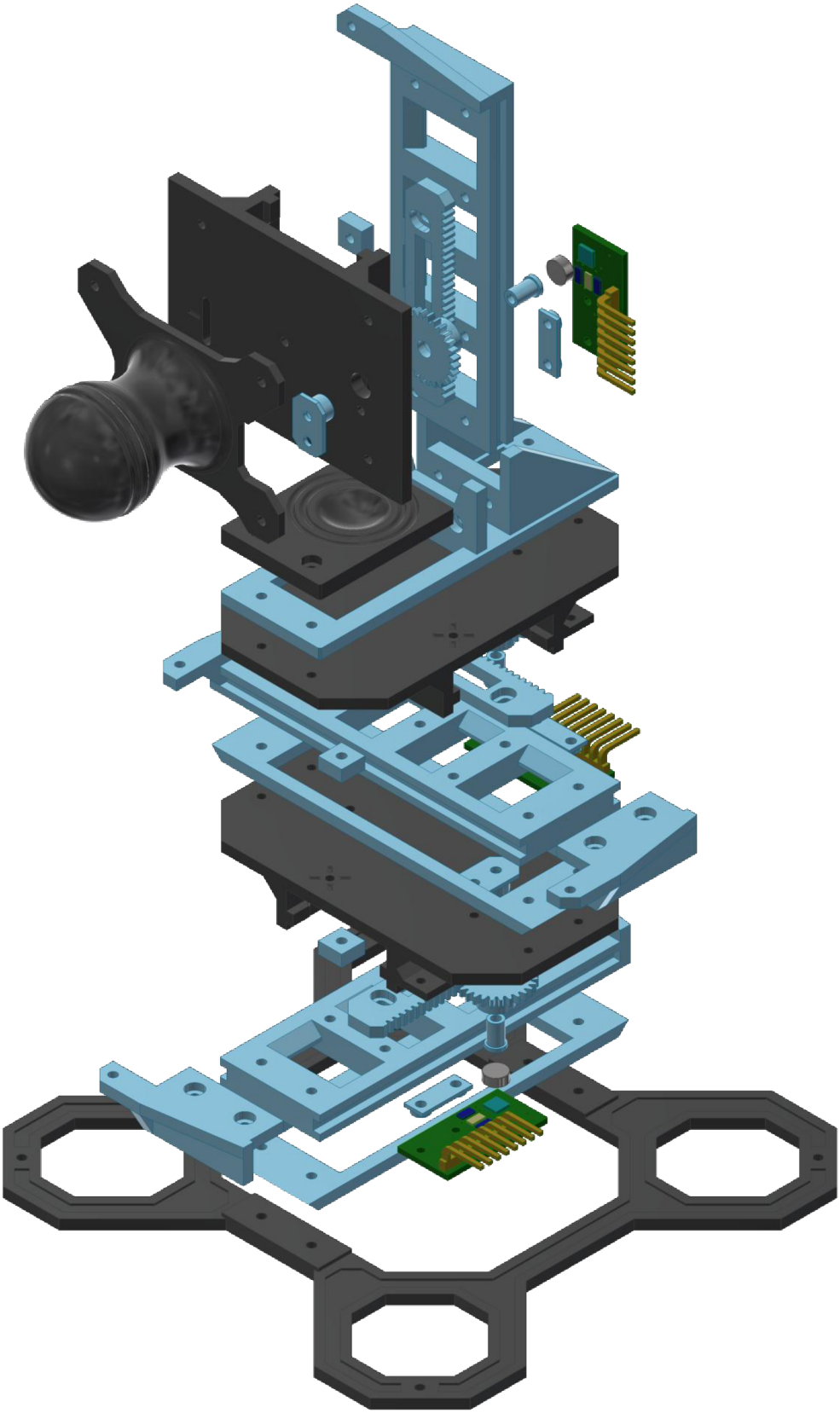
ZOZNAM POUŽITÝCH SKRATIEK

Skratka	Popis
ABS	Acrylonitrile Butadiene Styrene
AD	Analog-to-Digital
CAD	Computer-Aided Design
CSn	Chip Select Number
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DPS	Doska Plošných Spojov
EN_INT	Enable Interrupt
FPS	First-Person Shooter
H.O.T.A.S.	Hands-On Throttle-And-Stick
HID	Human Interface Device
ICSP	In Circuit Serial Programming
INT	Interrupt
LCD	Liquid-Crystal Display
LED	light-Emitting Diode
MISO	Master Input Slave Output
MOSI	Master Output Slave Input
NOP	No Operation
PLA	Polylactic Acid
PWM	Pulse-Width Modulation
SCK	Serial Clock
SPI	Serial Peripheral Interface
SS	Slave Select
USART	Universal Synchronous/Asynchronous Receiver/Transmitter

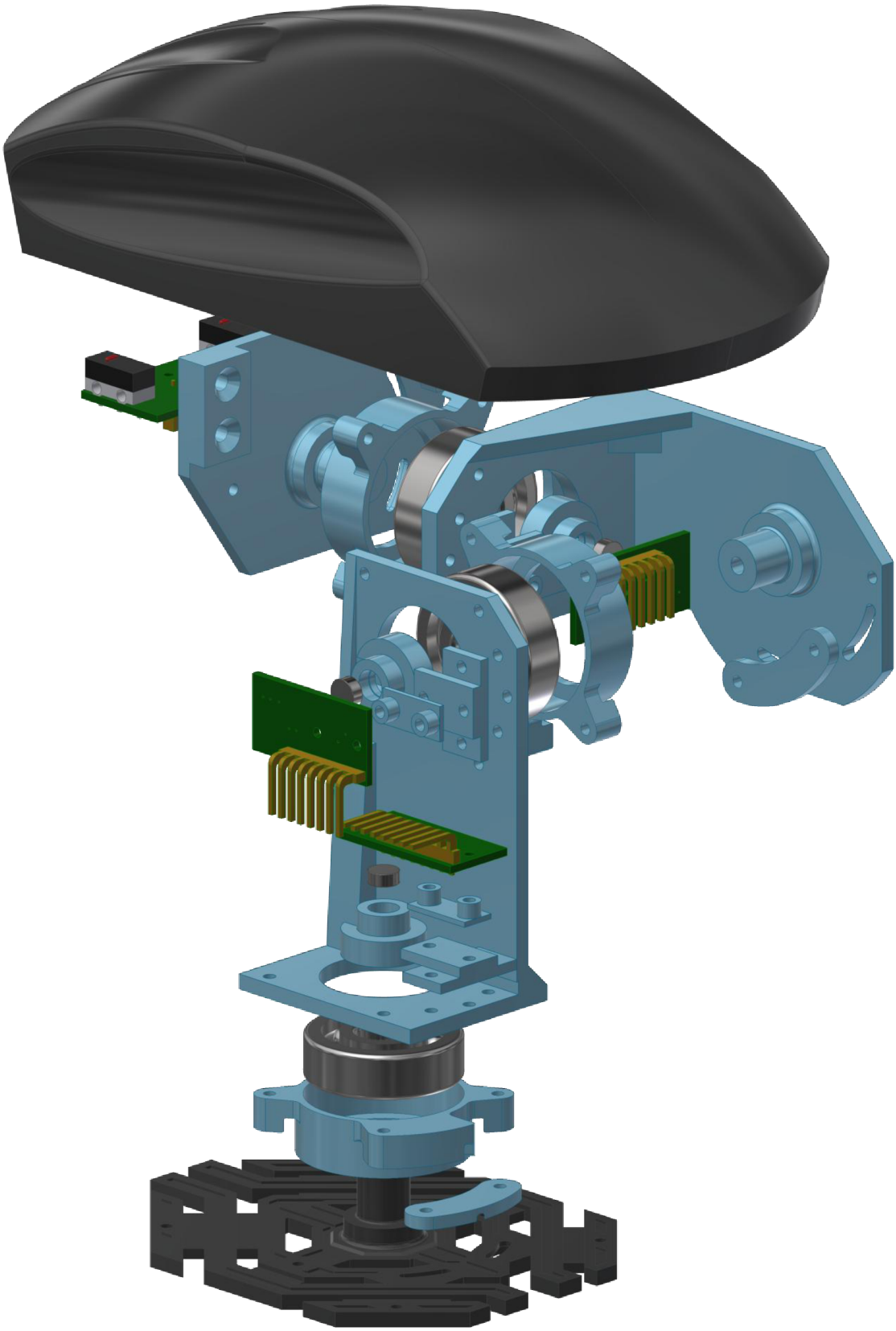
ZOZNAM OBRÁZKOV A TABULIEK

ZOZNAM PRÍLOH

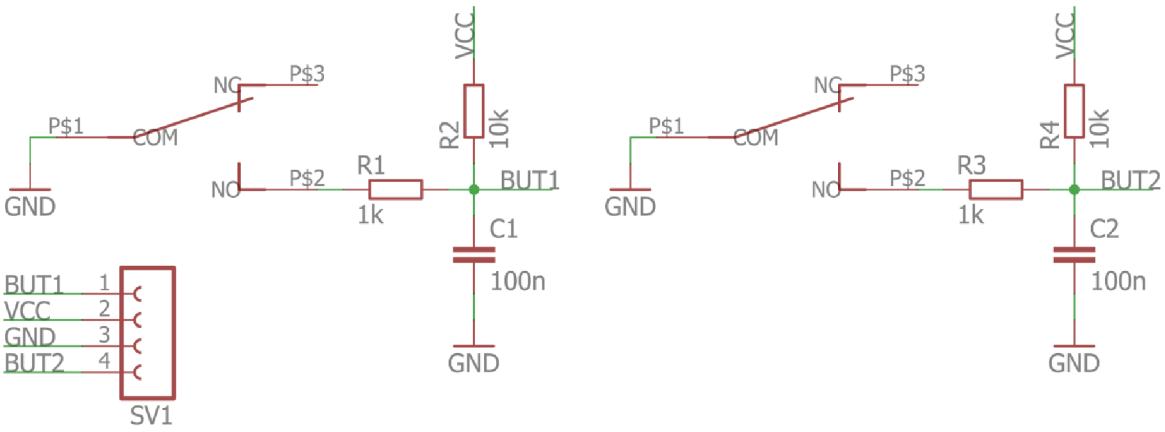
PRÍLOHA Č. 2 - ROZLOŽENÁ ZOSTAVA LINEÁRNEJ ČASTI MECHANIZMU

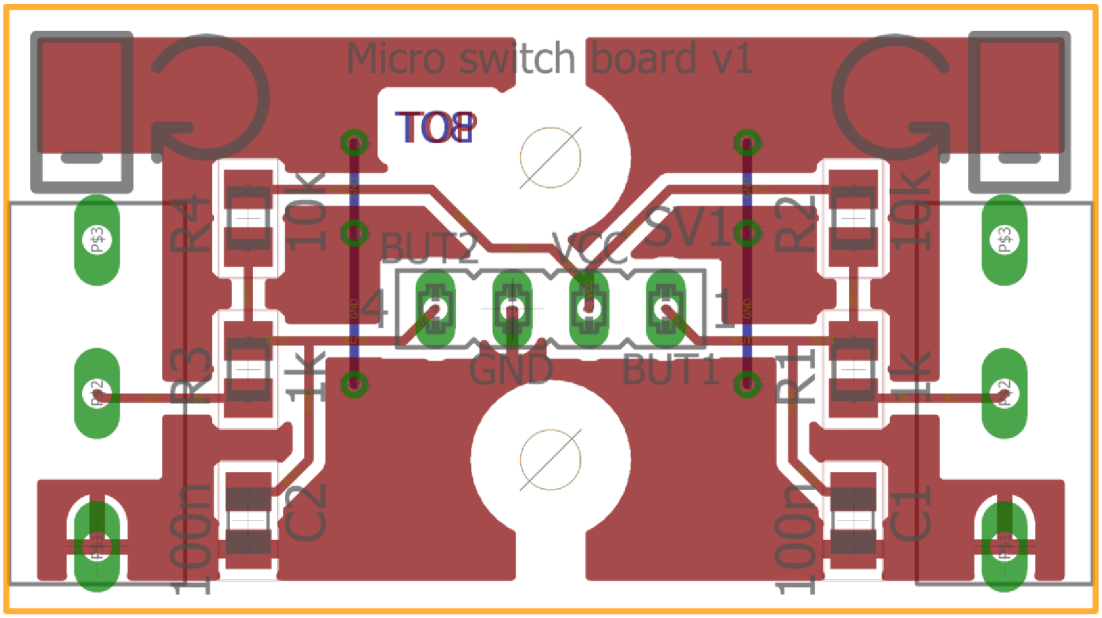


PRÍLOHA Č. 3 - ROZLOŽENÁ ZOSTAVA ROTAČNEJ ČASTI MECHANIZMU

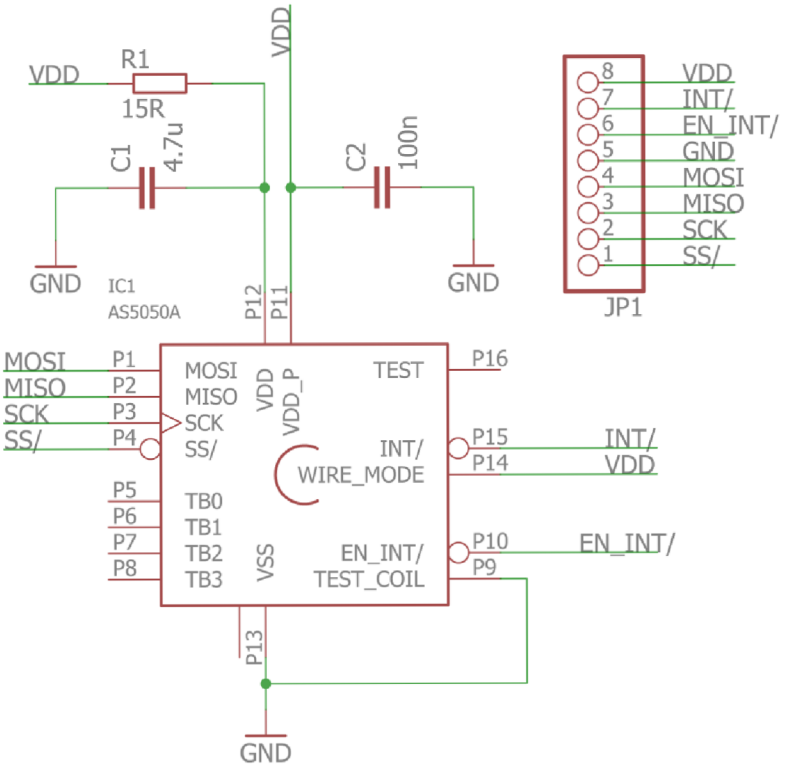


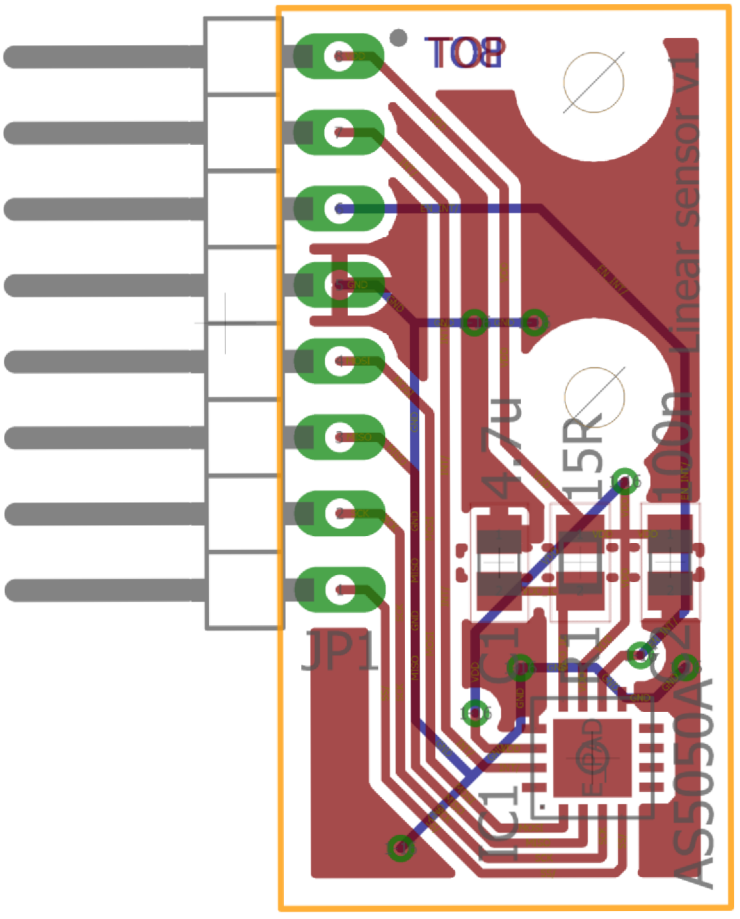
PRÍLOHA Č. 4 - SCHÉMA ZAPOJENIA A DPS PRE TLAČIDLÁ NA ROTAČNEJ ČASTI (VEĽKOSŤ 400%)



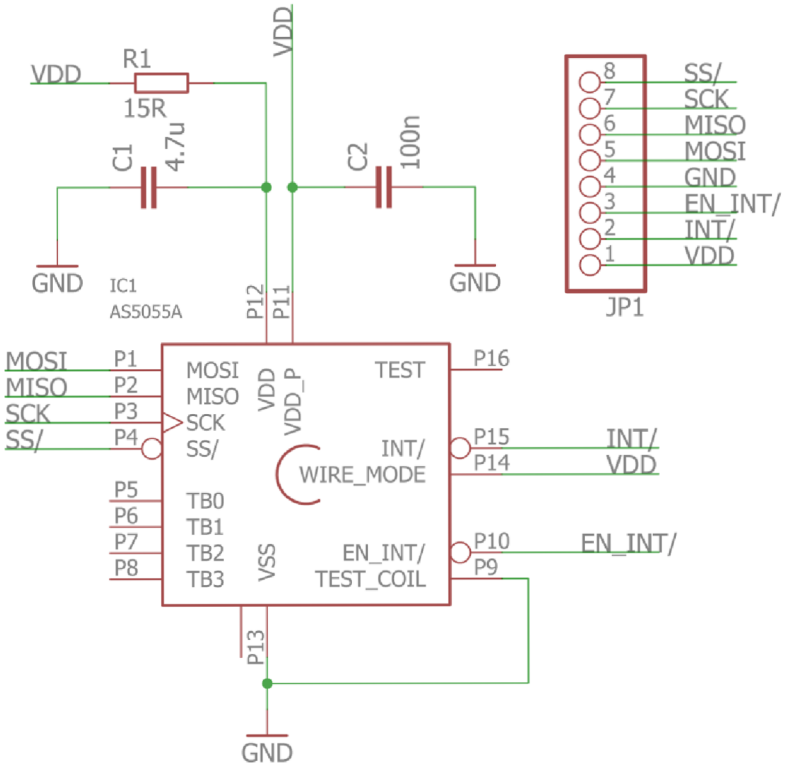


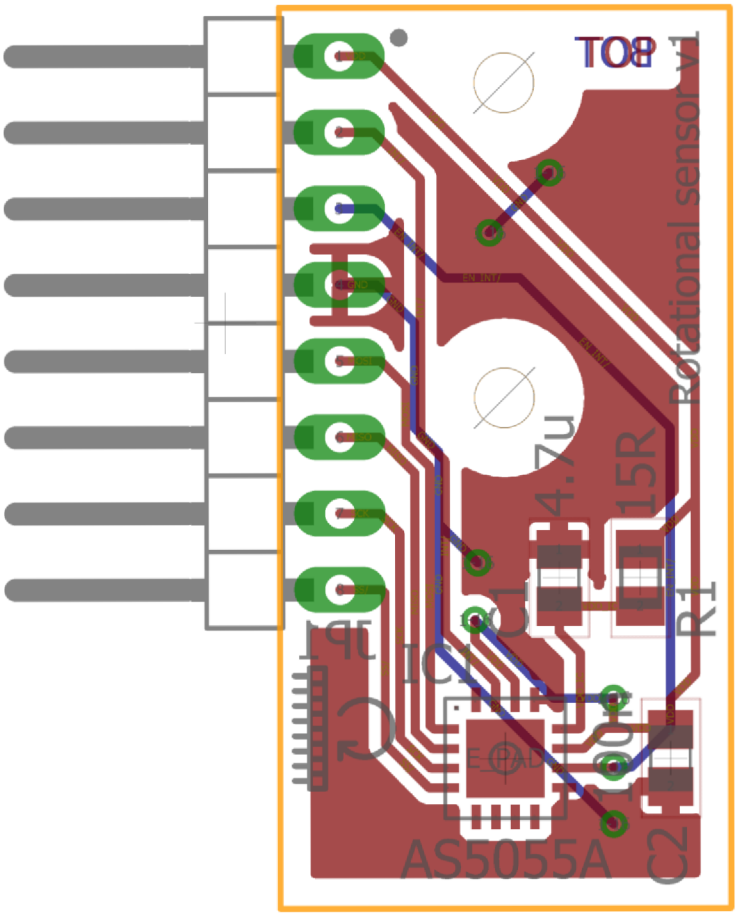
PRÍLOHA Č. 5 - SCHÉMA ZAPOJENIA A DPS PRE SNÍMAČE NA LINEÁRNEJ ČASTI (VEĽKOSŤ 400%)



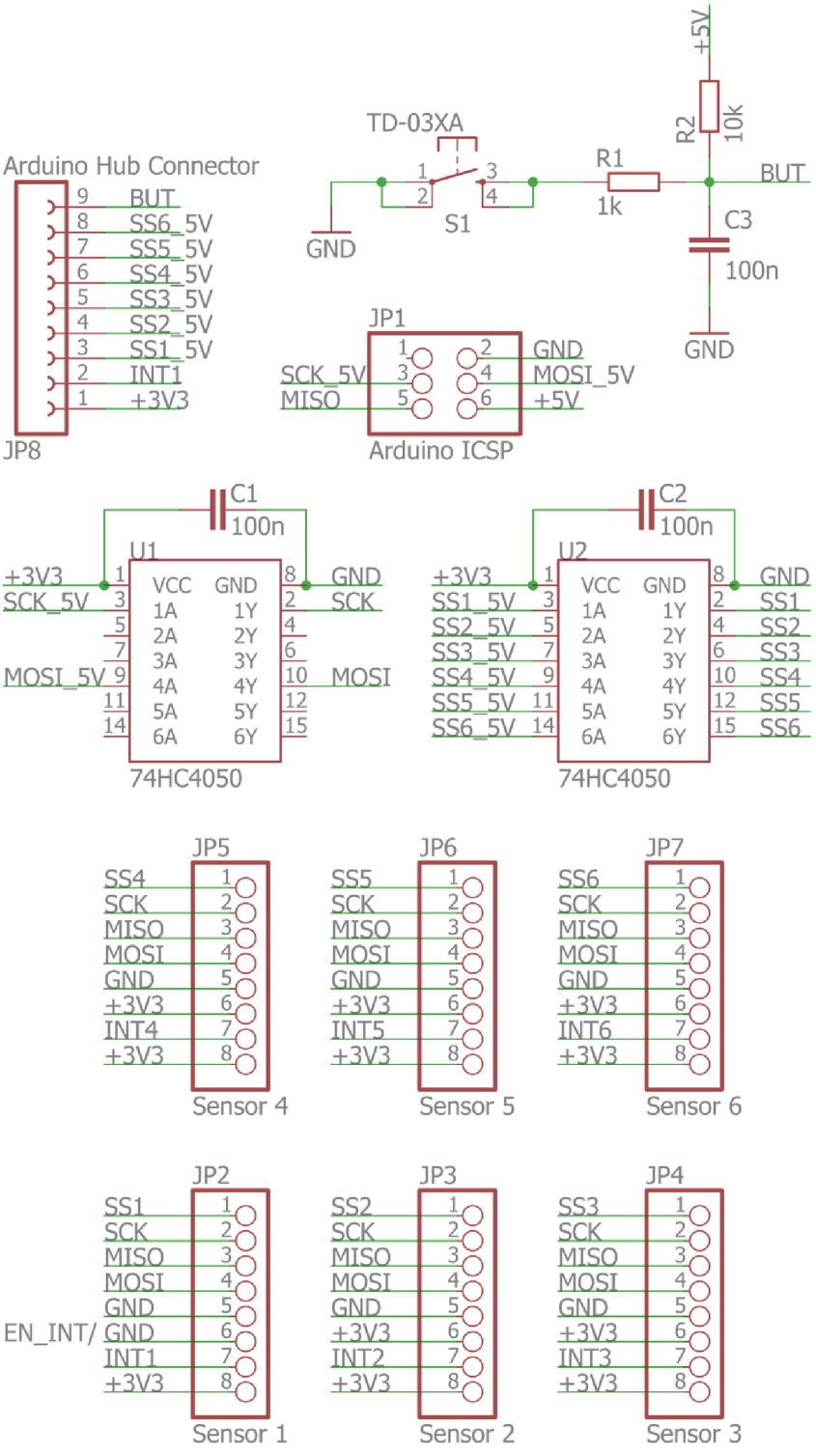


PRÍLOHA Č. 6 - SCHÉMA ZAPOJENIA A DPS PRE SNÍMAČE NA ROTAČNEJ ČASTI (VEĽKOSŤ 400%)

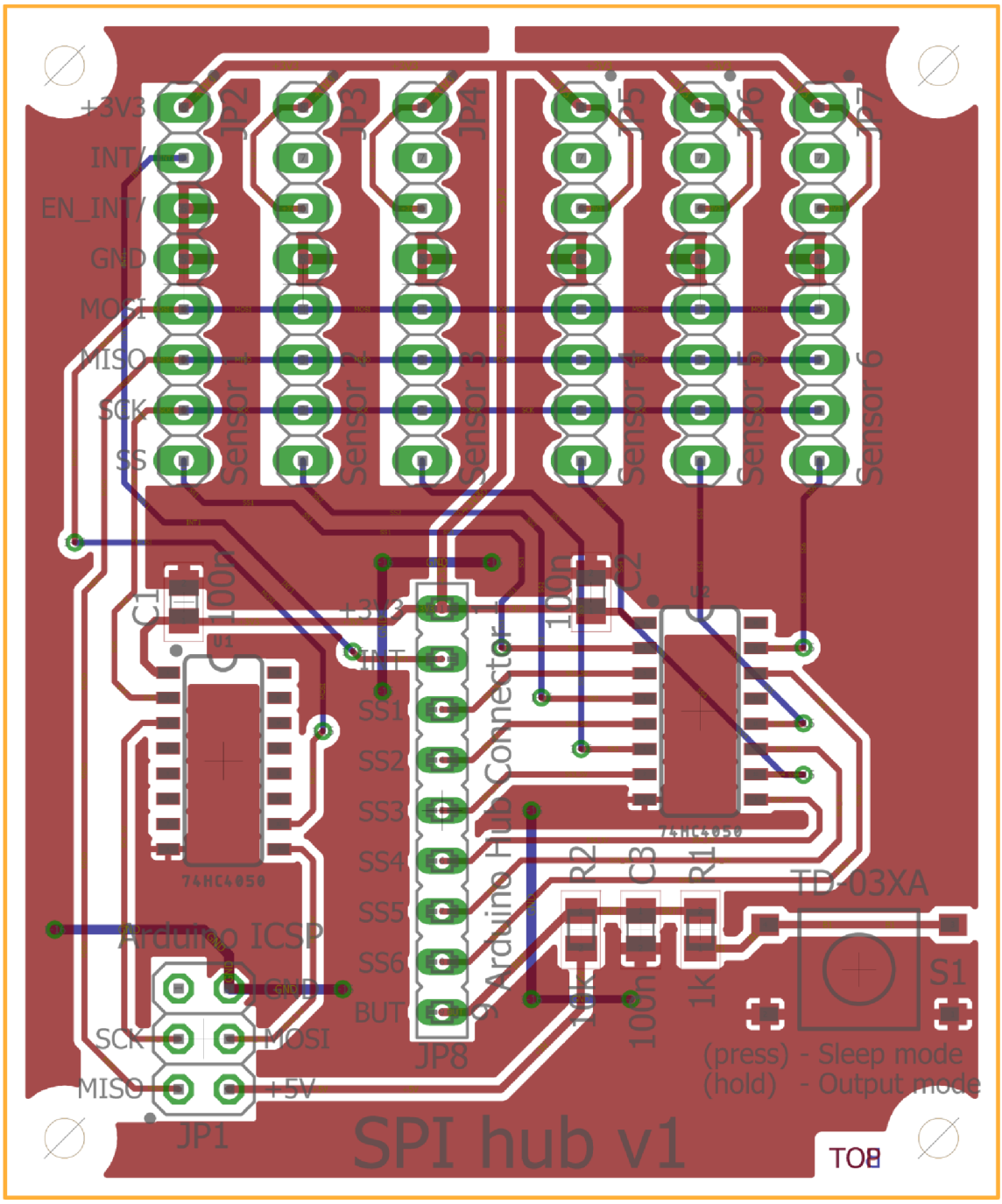




PRÍLOHA Č. 7 - SCHÉMA ZAPOJENIA ROZBOČOVAČA PRE SPI



PRÍLOHA Č. 8 - DPS ROZBOČOVAČA PRE SPI (VEĽKOSŤ 300%)



PRÍLOHA Č. 9 - NÁKRES PREPOJENÍ MEDZI DPS