Středoškolská technika 2025 Setkání a prezentace prací středoškolských studentů na ČVUT

NÁVRH A REALIZACE ROBOTICKÉHO RAMENE

Saša Kocúrek

SPŠ VOŠ Sokolská

Sokolská 1, Brno

Poděkování

Chtěl bych tímto poděkovat svému vedoucímu práce, panu Ing. Jaroslavu Nesvadbovi, a také panu Horňákovi za cenné rady a poskytnuté informace. Také Matyáši Krčkovi a mým přátelům za nedocenitelnou pomoc a zpětnou vazbu.

Obsah

1. Zadání 6
1.1 Specifické požadavky 6
2. Úvod 7
3. Definice robotického ramene 8
3.1 Co to je robot 8
3.2 Co je to robotické rameno 8
4. Hardware 9
4.1 Krokový motor 9
4.1.1 Fyzická stavba krokových motorů 10
4.2 Driver krokových motorů 10
4.2.1 Piny Driverů 11
4.3 CNC Shield 12
4.3.1 Zapojení CNC shieldu 13
4.4 Arduino MEGA 13
4.5 Servo motor 14
4.5.1 Servo MG995 14
4.6 Konstrukční prvky 15
4.6.1 Base 15
4.6.2 Ramenní kloub 15
4.6.2.1 Bázový kryt 15
4.6.2.2 Převod 16:1 15
4.6.2.3 Ramenná hřídel 16
4.6.2.4 Ramenný kryt (rameno) 16
4.6.3 Loketní kloub 16
4.6.3.1 Ramenní kryt (loket) 16
4.6.3.2 Loketní hřídel 16

4.6.3.3 Řemenový převod 17
4.6.3.4 Loketní kryt (loket) 17
4.6.3.5 Zápěstní krokový motor 17
4.6.4 Zápěstní kloub 17
4.6.4.1 Zápěstní kryt 18
4.6.4.2 Zápěstní hřídel 18
4.6.4.3 Držák servo motoru 18
5. Software 18
5.1 Arduino IDE 2 18
5.2 Struktura Arduino kódu 19
5.2.1 Další funkce void() 20
5.3 Kód ovládání robota 20
5.3.1 Hlavní funkce programu 20
5.3.2 Popis kódu 20
5.3.2.1 Prvních 70 řádků 20
5.3.2.2 Void setup() 21
5.3.2.3 Void loop() 21
5.3.2.4 Void calculatePath@() 21
5.3.2.5 Void moveAllMotors() 22
5.3.2.6 Void changeActiveAngle() 22
5.3.2.7 Void writeEncoderValue() 22
5.3.2.8 Void handleStoreEncoder() 23
5.3.2.9 Void saveCurrentPosition() 23
5.3.2.10 Void savePosition () 23
5.3.2.11 Void resetCurrentPosition() 23
5.3.2.12 PositionData loadPosition() 24
5.3.2.13 Void updateLCD() 24

5.3.2.14 Void toggleDisplay() 24
5.3.2.15 Void homing() 24
5.3.2.16 Void cyklus() 25
6. Plány do budoucna 26
7. Závěr 26
8. Bibliografie 27
9. Seznam obrázků a tabulek 28

1. Zadání

Cílem této středoškolské odborné činnosti je vytvoření jednoduchého robotického ramene za využití krokových motorů a mikropočítače Arduino. Následně je součástí zadání i program v rozhraní ArduinoIDE, který umožní fyzické nastavení pozic robota a jejich následné přehrávání.

1.1 Specifické požadavky

- Robotické rameno má alespoň 3 osy volnosti
- Robotické rameno využívá primárně krokové motory na ohyb kloubů
- Program na ovládání ramene je vytvořen v ArduinoIDE
- Veškeré zpracování a ukládání dat probíhá v mikropočítači Arduino

2. Úvod

Nápad vytvořit si vlastní robotické rameno mě napadl už v roce 2022, kdy jsem navštěvoval druhý ročník na SPŠ a VOŠ Sokolské. Tehdy jsem si představoval, že to bude jednoduché. Říkal jsem si: „Prostě do každého kloubu dám krokový motor, propojím je s řídicí jednotkou, naprogramuji pár základních pohybů a mám hotovo.“ Byl jsem přesvědčený, že jde o přímočarý projekt, který zvládnu během pár týdnů.

To jsem ale tehdy vůbec netušil, jak moc jsem se mýlil. Postupem času jsem začal zjišťovat, že konstrukce funkčního robotického ramene je mnohem komplexnější, než jsem si původně myslel. Nešlo jen o to připojit pár motorů a roztočit je. Každý kloub vyžaduje přesné řízení, správné načasování pohybů, a hlavně stabilní konstrukci, která unese váhu jednotlivých částí a zvládne i dynamické zatížení při práci. Navíc jsem musel řešit i otázky napájení, elektronického řízení a programování.

Toto téma jsem si zvolil nejen proto, že mě oblast robotiky odjakživa fascinovala, ale také proto, abych se konečně ‘dokopal‘ k realizaci mého dlouhodobého snu – postavit vlastní robotické rameno od nuly. Věděl jsem, že mě tento projekt donutí čelit novým výzvám, prohloubí mé technické znalosti a umožní mi získat cenné zkušenosti, které využiji i v budoucnu. Chtěl jsem si vyzkoušet celý proces – od prvotního návrhu přes konstrukci až po samotné programování a ladění.

3. Definice robotického ramene

3.1 Co to je robot

Robot je stroj navržený k vykonávání opakovaných nebo složitých úkolů, často s cílem nahradit nebo usnadnit lidskou práci. Bývají mechaniko-elektronický, ovšem v posledních letech i biologicky inspirovaný. Řízení probíhá buď pomocí předem naprogramovaných příkazů, nebo prostřednictvím umělé inteligence. Moderní roboti se používají v průmyslu, medicíně, vědě, vojenství i domácnosti, kde pomáhají automatizovat procesy a pracovat v nebezpečných podmínkách, které jsou pro člověka nevhodné nebo riskantní.

Pojem ROBOT sahá až k světově proslavené divadelní hře Karla Čapka R.U.R. (Rossum's Universal Robots). V této hře byl Robot biologická bytost vytvořená člověkem k usnadnění lidského života.

V závislosti na své konstrukci mohou být roboti mobilní nebo stacionární. Některé druhy robotů také zvládají vykonávat úkoly, které vyžadují určitou míru inteligence, jako je rozpoznávání objektů, učení nebo přizpůsobování se novým situacím.

Obrázek 1 Reprezentace mobilního robota [7]

3.2 Co je to robotické rameno

Robotické rameno je mechanické zařízení, které napodobuje pohyb lidské ruky a slouží k manipulaci s objekty v různých prostředích. Skládá se z několika kloubů a segmentů, které umožňují pohyb v mnoha osách. Pohyb je možno řídit elektromotory, hydraulikou nebo pneumatickými systémy a často je koordinován pomocí mikrokontrolerů, mikropočítačů nebo průmyslových řídicích systémů. Robotická ramena mohou pracovat autonomně podle předem naprogramovaných instrukcí nebo být ovládána manuálně pomocí dálkového ovládání či umělé inteligence.

Oproti robotům obecně se robotické rameno soustředí pouze na mechanickou manipulaci a není autonomní v širším smyslu. Dá se tedy říct, že každé robotické rameno je robot, ale ne každý robot je robotické rameno.

4. Hardware

Robotické rameno se skládá z fyzické a programové části. Fyzická neboli hardware část je složena z aktivních a konstrukčních prvků.

Aktivní prvky zahrnují pohonné jednotky, jako jsou krokové motory, servomotory nebo pneumatické válce, které umožňují pohyb jednotlivých segmentů ramene. Dále sem patří senzory, například enkodéry pro zpětnou vazbu pohybu, tlakové senzory pro detekci uchopení objektu nebo kamerové systémy pro vizuální navigaci.

Konstrukční prvky tvoří rám a kloubové spoje, které zajišťují mechanickou stabilitu a určují pracovní dosah ramene. Materiály mohou být různé – od hliníku a oceli u průmyslových ramen až po plasty a plastové kompozity u lehčích aplikací, například v domácí automatizaci nebo edukativní robotice. Mezi doma postavenými robotickými rameny je často populární si konstrukční prvky vytisknout na 3D tiskárně. Je to levný a efektivní způsob, jak vyrobit složité tvary, využívané v robotice.

4.1 Krokový motor

Ve svém robotické rameni využívám krokové motory NEMA 17. Označení NEMA je zkratka pro National Electrical Manufacturers Association(česky: národní asociace výrobců elektrotechniky). Nejedná se tedy o konkrétní krokový motor, nýbrž o velikost příruby použitého krokového motoru.

Obrázek 2 Krokový motor NEMA 17 [6]

Motory NEMA 17 mají přírubu o rozměrech 42 × 42 mm a běžně se využívají v 3D tiskárnách, CNC strojích a robotických aplikacích. Jejich výhodou je kompaktní velikost, dobrý poměr výkonu k spotřebě a přesnost pohybu, což je klíčové pro plynulé a kontrolované otáčení jednotlivých segmentů robotického ramene.

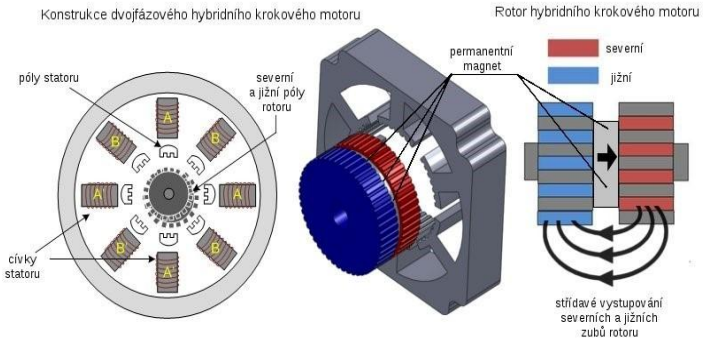
Krokové motory obecně fungují na principu diskrétních kroků, přičemž nejčastěji používané modely mají 200 kroků na otáčku (1,8° na krok). Přesnost lze dále zvýšit pomocí mikrokrokování. Mikrokrokování je technika řízení krokových motorů, která umožňuje přesnější a plynulejší pohyb oproti standardnímu krokovému režimu. Zatímco běžné krokové motory se pohybují v pevných krocích, mikrokrokování rozděluje každý krok na menší úseky, čímž zvyšuje rozlišení pohybu. Typické hodnoty mikrokrokování zahrnují poloviční krok

(1/2), čtvrtinový krok (1/4), osminový krok (1/8) až šestnáctinový nebo vyšší (1/16, 1/32, 1/64). To se provádí řízeným střídáním napětí na jednotlivých cívkách motoru, čímž se rotor drží v mezipozicích mezi plnými kroky.

Výhody mikrokrokování zahrnují nižší vibrace, tišší chod motoru a vyšší přesnost, což je zásadní pro aplikace jako robotická ramena, 3D tiskárny nebo CNC stroje. K implementaci mikrokrokování se používají drivery, které výrazně zjednodušují používání krokových motorů. Také umožňují nastavit požadovanou úroveň mikrokrokování podle potřeby dané aplikace.

4.1.1 Fyzická stavba krokových motorů

Fyzická konstrukce krokového motoru se skládá ze dvou hlavních částí: statoru a rotoru. Stator je pevná část motoru obsahující elektromagnetické cívky, které při průchodu elektrického proudu vytvářejí magnetické pole. Tyto cívky jsou rozmístěny kruhově kolem středu motoru a zapojené tak, aby při jejich postupném spínání docházelo k plynulému otáčení rotoru. Rotor je pohyblivá část motoru, která reaguje na změny magnetického pole ve statoru a přeskakuje mezi jednotlivými kroky. U hybridních krokových motorů, které jsou nejběžnější, je rotor tvořen ozubeným feromagnetickým jádrem, které zvyšuje přesnost pohybu a zlepšuje točivý moment motoru. Celý systém je uzavřen v kovovém krytu, který chrání vnitřní součásti a zajišťuje stabilitu motoru při provozu. Na zadní straně motoru může být připevněn enkodér pro zpětnou vazbu polohy, zatímco přední část obsahuje hnací hřídel, která přenáší rotační pohyb na mechanickou soustavu, například na řemeny, ozubená kola nebo přímo na konstrukci robotického ramene.



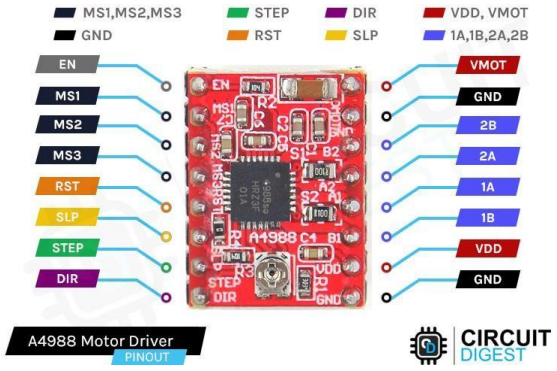
Obrázek 3 Schéma rotoru a statoru krokového motoru [1]

4.2 Driver krokových motorů

Drivery jsou elektronická zařízení, která řídí průchod elektrického proudu cívkami krokového motoru a tím určují rychlost a směr pohybu. Krokové motory nelze připojit přímo k napájecímu zdroji, protože potřebují přesně řízené pulzy elektrického signálu. Drivery tedy zajišťují nejen spínání jednotlivých fází, ale také nastavování proudu a napětí, což ovlivňuje točivý moment a plynulost pohybu motoru. Drivery jsou často obohaceny o možnost

mikrokrokovat krokový motor. Pokročilé drivery obsahují funkce pro tichý chod, teplotní ochranu nebo detekci ztracených kroků.

Drivery se ovládají pomocí digitálních signálů z mikrokontrolérů a mikropočítačů (např. Arduino, ESP32 nebo Raspberry Pi), které určují směr otáčení a počet kroků. Připojení obvykle zahrnuje signálové piny jako STEP (krok), DIR (směr) a EN (aktivace motoru). Kromě základního řízení mohou drivery obsahovat i regulaci proudu, která omezuje zahřívání motoru a zvyšuje jeho efektivitu. Tato regulace na driverech vypadá jako malý potenciometr s křížovým V některých aplikacích, jako jsou robotická ramena, CNC stroje nebo 3D tiskárny, se drivery kombinují s enkodéry a zpětnovazebními systémy, což umožňuje ještě přesnější polohování a kompenzaci mechanických nepřesností. Na obrázku je driver A4988, ale v robotickém rameni je použitý driver DRV8825. Oba tyto drivery mají stejný pinout, ale driver DRV8825 zvládá přenášet větší elektrický proud. [1] [2]



Obrázek 4 Pinout driveru krokového motoru [1]

4.2.1 Piny Driverů

EN – pin na zapnutí driveru, číslicový signál HIGH -> driver zapnut

RST – reset pin

SLP – sleep pin

STEP – krokový pin, každý HIGH pulz posune motor o krok či mikrokrok, čím větší frekvence tím rychleji se krokový motor otáčí

DIR – určuje směr otáčení, číslicový signál HIGH otáčí motor po směru hodinových ručiček, LOW protisměru

VMOT – pin určený k napájení krokového motoru, v závislosti na driveru vyžaduje 8-35 V stejnosměrného napětí

GND – pin určený k uzemnění krokového motoru nebo driveru

1A,1B,2A,2B – piny připojeny přímo k cívkám krokového motoru

VDD – pin určený k přívodu +5 V elektrického napětí za účelem napájení driveru

MS1,MS2,MS3 – určují mikrokrokování krokového motoru, kombinace vysvětleny v tabulce

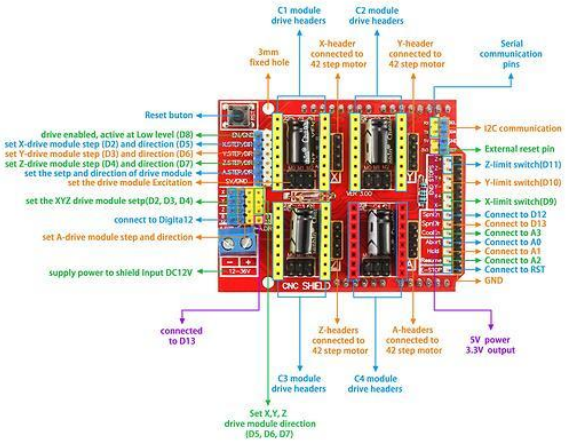
Tabulka 1 Mikrokrokování driveru krokového motoru

MS1	MS2	MS3	Mikrokrokování
Low	Low	Low	Normální kroky
High	Low	Low	1/2 kroky
Low	High	Low	1/4 kroky
High	High	Low	1/8 kroky
Low	Low	High	1/16 kroky
High	Low	High	1/32 kroky
Low	High	High	1/32 kroky
High	High	High	1/32 kroky

4.3 CNC Shield

Arduino CNC Shield V3 je rozšiřující deska určená pro řízení 3D tiskáren a CNC strojů, pomocí Arduino Uno. Tento shield umožňuje snadné připojení až čtyř krokových motorů prostřednictvím slotů pro drivery. Každá osa (X, Y, Z a volitelně A) má vyhrazené piny pro STEP a DIR signály, což umožňuje přesné řízení pohybu motorů. Shield také obsahuje konektory pro koncové spínače (endstopy), které pomáhají detekovat krajní polohy os a zabraňují mechanickému poškození. Dále disponuje výstupem pro zahřívání topné podložky 3D tiskáren.

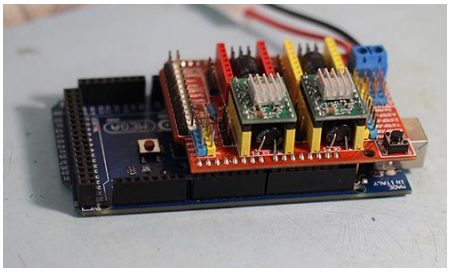
Napájení desky je řešeno prostřednictvím externího zdroje 12–36 V, což umožňuje dostatečný výkon pro krokové motory. Pro komunikaci a řízení se často využívá GRBL firmware, což je open-source software optimalizovaný pro CNC aplikace běžící na Arduinu. Tento firmware interpretuje G-kód a převádí jej na signály pro krokové motory, čímž umožňuje přesné řízení pohybu nástroje.



Obrázek 5 Pinout CNC shieldu V3 [6]

Arduino CNC Shield V3 je oblíbený díky své jednoduché instalaci, široké kompatibilitě a nízké ceně.

4.3.1 Zapojení CNC shieldu



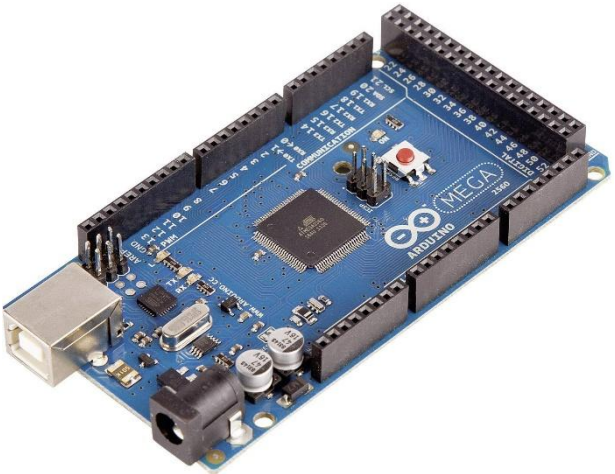
Obrázek 6 Arduino MEGA se zapojeným CNC shieldem a drivery

Zapojení a konfigurace Arduino CNC Shieldu V3 začíná jeho nasazením na Arduino UNO, čímž se propojí všechny potřebné signálové piny. Do slotů pro drivery krokových motorů je nutné správně vložit drivery s ohledem na jejich orientaci – nesprávné zapojení může způsobit jejich trvalé poškození. Pod sloty driverů jsou propojitelné piny, které propojují driver piny MS1,MS2,MS3 a tím nastavují mikrokrokování. K napájení motorů se využívá externí zdroj 12–36 V, který se připojuje ke svorkám +V a GND. V levém horním rohu desky se nachází reset tlačítko. To vypne a znovu zapne nejen CNC shield ale i připojené Arduino Na pravé straně se nachází piny pro koncové spínače (endstopy) pro osy X, Y a Z, což umožňuje přesné určení referenční polohy ramene. Pod endstopy se nachází CNC shieldem nepoužívané analogové vstupové piny. Vedle analogových pinů a endstopů se nachází řada GND pinů.

I když je Arduino CNC Shield V3 určen k instalaci na Arduino UNO, je možné jej nainstalovat i na Arduino MEGA. Tím získáme jednoduché ovládání CNC shieldu a navýšený počet pinů Arduina MEGA.

4.4 Arduino MEGA

Arduino MEGA 2560 je výkonná mikrokontrolérová deska založená na čipu ATmega2560, která je navržena pro složitější projekty vyžadující více vstupů, výstupů a větší výpočetní kapacitu. Deska obsahuje 54 digitálních pinů, z nichž 15 podporuje PWM, a 16 analogových vstupů, což ji činí ideální pro aplikace, jako jsou robotické systémy, CNC stroje, automatizace nebo datová sběrná zařízení. Arduino MEGA je kompatibilní s prostředím Arduino IDE a podporuje sériovou komunikaci pomocí UART (4×), I2C a SPI. Díky větší paměti (8 KB SRAM, 4 KB EEPROM a 256 KB Flash) zvládá zpracovávat složitější programy a zároveň



Obrázek 7 Ilustrace Arduino MEGA [6]

umožňuje práci s více knihovnami najednou.

Ve srovnání s Arduino UNO, které je vybaveno ATmega328P a nabízí pouze 14 digitálních pinů a 6 analogových vstupů, je Arduino MEGA výrazně rozšířenější a vhodné pro projekty s vyššími nároky na počet připojených zařízení. Zatímco Arduino Uno disponuje pouze jedním sériovým portem (UART), MEGA má čtyři, což je výhodné při komunikaci s více moduly, například s LCD displeji, GPS nebo bezdrátovými moduly. Další výhodou Mega je větší operační paměť, která umožňuje práci s náročnějšími programy, zatímco Uno se svými 2 KB SRAM může u složitějších aplikací narazit na limity. Nevýhodou Mega je však větší rozměr, což může být problematické při montáži do prostorově omezencých zařízení.

Díky své open-source povaze je Arduino obecně ideální pro širokou komunitu vývojářů, inženýrů a hobbyistů. Tento otevřený přístup umožňuje každému přizpůsobit a modifikovat hardware i software deskek podle konkrétních potřeb, což podporuje inovace a spolupráci na různých projektech. K dispozici je spousta knihoven, příkladů a dokumentace, které usnadňují práci na aplikacích v robotice, automatizaci a dalších oblastech. Arduino IDE a Arduino IDE2, které jsou zdarma a také open-source, poskytují jednoduché prostředí pro programování, což umožňuje rychlý vývoj i pro začátečníky. Tento ekosystém otevřených nástrojů a projektů dává možnost připojovat a využívat moduly, senzory a shieldy od různých výrobců, čímž vzniká široká kompatibilita a rozšiřitelnost.

4.5 Servo motor

Servo motor je nejvzdálenější článek od báze, pomineme-li čelisti. Nachází se na konci zápěstního kloubu. A zajišťuje náklon čelistí, přímo připojených k servu.

4.5.1 Servo MG995

Servo motor MG995 je výkonné digitální servo s kovovými převody, které poskytují vyšší odolnost a přesnost oproti plastovým variantám. Díky točivému momentu až 13 kg·cm při 6V je vhodné pro širokou škálu aplikací, včetně RC modelů, robotiky, mechanických mechanismů a automatizace. MG995 pracuje na PWM signálu, což umožňuje přesné řízení polohy v rozsahu

Obrázek 8 Servo MG995 s příslušenstvím [6]

přibližně 120°. Je oblíbené díky své spolehlivosti, dobrému výkonu a dostupné ceně.

4.6 Konstrukční prvky

Konstrukční prvky by se daly rozdělit na prvky přenosné a strukturní. Jak již název napovídá, přenosné prvky přenáší kroutící moment z jedné části robotického ramene na druhý. Prvky strukturní slouží jako pevná struktura, držící veškeré ostatní prvky pohromadě. [3]

4.6.1 Base

Base (česky Báze) je strukturní prvek držící celé robotické rameno. Obsahuje jedno axiální ložisko pro hladké otáčení ramene kolem své osy. Dále obsahuje jedno radiální ložisko k zabránění nežádoucího točivého momentu. Báze také obsahuje jeden krokový motor, napojen na ozubený převod, který otáčí celým robotickým ramenem kolem své osy.

4.6.2 Ramenní kloub

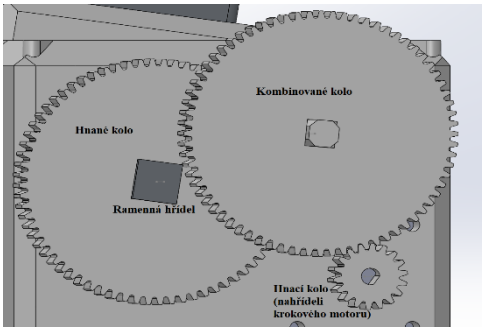
Ramenní kloub se skládá z bázového krytu a ramenného krytu. Tyto dva kryty jsou 3D tištěné a spojeny jsou ramennou hřídelí. Ramenný kryt pak dále pokračuje loketnímu kloubu.

4.6.2.1 Bázový kryt

Bázový kryt je 3D tištěný tvar připomínající dutý kvádr. Uvnitř se nachází 2 radiální ložiska, která zajišťují plynulé otáčení ramenné hřídele. Dále se uvnitř bázového krytu nachází krokový motor, který po ozubeném převodu 16:1 (vstup : výstup) otáčí ramenným kloubem. Krokový motor je k bázovému krytu přidělám čtyřmi M3 šroubky.

4.6.2.2 Převod 16:1

Ozubený převod 16:1 je složen ze tří ozubených kol, přičemž využívá dvoustupňovou redukci. První stupeň převodu má poměr 4:1, kde malé hnací kolo pohání kombinované kolo, které obsahuje dvě ozubená kola s různým průměrem – jedno pro vstup a druhé pro výstup. Druhý stupeň převodu opět využívá poměr 4:1, kde menší část kombinovaného kola přenáší pohyb na velké hnané kolo. To je napojeno na ramennou hřídel.



Obrázek 9 Model převodu 16:1 v SolidWorks

4.6.2.3 Ramenná hřídel

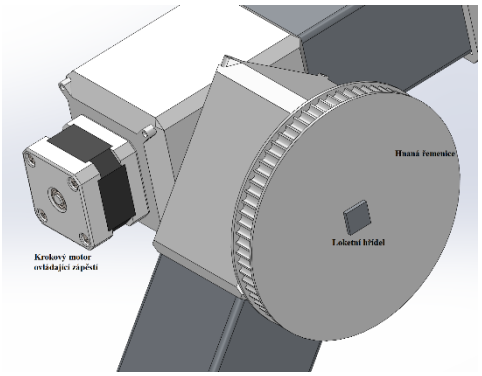
Ramenná hřídel se skládá z hliníkového profilu a 3D tištěného obalu. Jádro hřídele tvoří hliníková tyčovina čtvercového profilu (12x12mm). Ta přenáší veškerý kroutící moment. Hřídel by mohla být celá 3D tištěná, ale 3D tištěné součástky špatně snášejí namáhání na krut. Je dobré podotknout že 3D tištěný obal má přibližně poloviční délku v porovnání s hliníkovým jádrem. To je z důvodu, že 3D tištěný obal hřídele slouží k hladkému uložení do radiálních ložisek bázového krytu. Samotné jádro je tvarově spojeno s ramenným krytem.

4.6.2.4 Ramenný kryt (rameno)

Ramenný kryt je velice podobný krytu bázovému. Je ovšem méně dutý. Má v sobě jen 2 dutiny. První slouží k spojení s jádrem hřídele. Druhý pak obsahuje krokový motor, který řemenovým převodem pak otáčí loketním kloubem. To je za účelem posunu těžiště ramene blíž k bázi. To snižuje kroutící moment potřebný otáčení robotického ramene. Krokový motor je upevněn pomocí čtyř M3 šroubků. Jeden konec ramenného krytu je připojen k hliníkovému profilu (60x60mm), který slouží jako struktura mezi kloubem ramenním a loketním.

4.6.3 Loketní kloub

Loketní kloub je podobný tomu ramennímu, ale jsou mezi nimi rozdíly. Skládá se z ramenního krytu, loketní hřídele, řemenového převodu, loketního krytu (loket) a zápěstního krokového motoru.



Obrázek 10 Loketní kloub, řemen je zneviditelněný

4.6.3.1 Ramenní kryt (loket)

Jedná se o 3D tištěný dutý kvádr, který v sobě obsahuje 2 radiální ložiska. Z dolní strany ramenného krytu (loket) je připevněn hliníkový profil (60x60mm), ten stejný jako k ramennému krytu (rameno). V ložiscích je uložena loketní hřídel.

4.6.3.2 Loketní hřídel

Loketní hřídel velice připomíná hřídel ramenní. Jediným rozdílem je velikost. Loketní hřídel je složena z hliníkového profilu (12x12mm) a 3D tištěného obalu. 3D tištěný obal opět slouží jen k uložení jádra do ložisek. Jádro je z jedné strany připojeno na řemenici a z druhé na loketní kryt (loket). Jádro je opět nositelem veškerého kroutícího momentu.

4.6.3.3 Řemenový převod

Řemenový převod se skládá ze tří řemenic a jednoho řemene. Řemenový převod má převodový poměr 4:1. Řemenice hnací je malá ocelová řemenice koupená na internetu. Řemenice napínací je stejná jako řemenice hnací. Řemenice hnaná je 3D vytištěná řemenice s čtvercovou dírou uprostřed, pro připojení jádra loketní hřídele. Řemen je 720mm dlouhý. Osová vzdálenost mezi krokový motorem loktu a loktem samotným je menší, proto bylo potřeba přidat napínací řemenici.

4.6.3.4 Loketní kryt (loket)

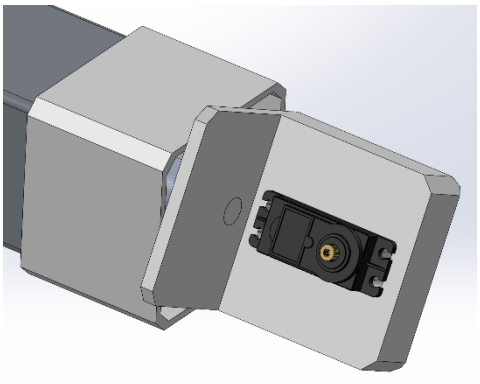
Loketní kryt (loket) se velice podobá ramennímu krytu (loket). Ze spodní strany je připojen na předloktní hliníkový profil (60x60mm), který slouží jako struktura mezi loktem a zápěstím. Dále obsahuje čtvercovou díru pro upevnění jádra loketní hřídele. Na opačné straně upevnění strukturního předloktního hliníkového profilu se nachází zápěstní krokový motor, který je upevněn pomocí čtyř M3 šroubků.

4.6.3.5 Zápěstní krokový motor

Zápěstní krokový motor je upevněn u loktového kloubu právě za účelem posunu těžiště co nejblíže bázi. Zápěstní krokový motor je připojen k zápěstní hřídeli, která přenáší kroutící moment uvnitř předloktního hliníkového profilu až k samotnému zápěstí. Motor je dost silný na otáčení zápěstí, proto nepotřebuje žádné ozubené převody.

4.6.4 Zápěstní kloub

Zápěstní kloub je nejvzdálenější kloub od báze. Také je nejunikátnější, protože je zapotřebí jím otáčet a zároveň ohýbat. První pohyb je rotační pohyb kolem osy zápěstí, který zajišťuje zápěstní krokový motor. Druhý pohyb je naklánění zápěstí „nahoru a dolů“, který je realizován pomocí servo motoru. Skládá se ze zápěstního krytu, zápěstní hřídele, držáku servo motoru, servo motoru a radiálního ložiska.



Obrázek 11 Zápěstní kloub se servo motorem

4.6.4.1 Zápěstní kryt

Zápěstní kryt lehce připomíná kryty použité v loktu. Zápěstní kryt je ale menší. Na své spodní straně se do nasouvá předloktní hliníkový profil (60x60mm). Uprostřed ve stejné ose jako předloktní hliníkový profil se nachází radiální ložisko, které drží zápěstní hřídel. Zápěstní kryt je opět 3D tištěn, z důvodu komplexnosti jeho tvaru.

4.6.4.2 Zápěstní hřídel

Zápěstní hřídel je hliníková kruhová tyčovina vysoustružená na průměr 20mm. Přenáší kroutící moment od zápěstního krokového motoru na držák servo motoru. Kroutící moment je dostatečně malý, aby se zde dala použít 3D tištěná hřídel, ale v praxi je jednoduší za pár minut vysoustružit hřídel než ji hodiny tisknout na 3D tiskárně. Na obou koncích hřídele se nachází osazení s průměrem 8mm. Na tyto osazení se pak nasadí pružné spojky, které hřídel spojují s zápěstním krokovým motorem a držákem servo motoru.

4.6.4.3 Držák servo motoru

Držák servo motoru je 3D tištěný díl připomínající písmeno L. Na spodní straně podstavy je plocha s výstupkem, který se připojuje k pružné spojce. Na opačné straně podstavy se nachází jedna stěna. Tato stěna má v sobě díru ve tvaru profilu servo motoru a čtyř šroubků M3 pro uchycení.

5. Software

Veškeré ovládání mého robotického ramene je naprogramováno v arduino kódu, v prostředí Arduino IDE 2. Na internetu je snadné vyhledat podobná robotická ramena, která používají arduino kód jen jako prostředníka mezi softwarem, který vypočítává polohy a dráhy ramene, a hardwarem. V mém robotickém rameni veškeré výpočty probíhají uvnitř arduina. Hlavní výhodou tohoto řešení, je možnost funkce robotického ramene i bez připojení počítače.

5.1 Arduino IDE 2

Arduino IDE 2 je modernizovaná verze původního vývojového prostředí Arduino IDE pro programování mikrokontrolérů Arduino. Přináší vylepšené uživatelské

Obrázek 12 Logo Arduino IDE 2

rozhraní, rychlejší odezvu a rozšířenou funkcionalitu, která usnadňuje vývoj a ladění kódu. Arduino IDE 2 běží jako nativní aplikace pro Windows, macOS i Linux, přičemž si zachovává jednoduchost a intuitivnost původní verze.

Grafické rozhraní je přehledné a obsahuje postranní panel, který umožňuje snadný přístup k připojeným deskám, dostupným knihovnám a sériovému monitoru. Sériový monitor a plotter jsou nyní integrovány přímo do hlavního okna, což umožňuje rychlejší sledování výstupů z Arduina. Významným vylepšením je také podpora více záložek a projektového přístupu, což umožňuje organizovanější práci na rozsáhlejších kódech. Arduino IDE 2.0 také usnadňuje instalaci knihoven a správu rozšíření díky vylepšenému Library Manageru a Boards Manageru, které umožňují rychle přidávat podporu pro nové desky a periferie.

Obrázek 13 Programovací prostředí Arduino IDE 2

5.2 Struktura Arduino kódu

V programování pro mikrokontroléry Arduino jsou funkce void setup() a void loop() základními stavebními bloky každého programu. Funkce void setup() se spustí pouze jednou po zapnutí nebo restartu Arduina a slouží k nastavení režimu pinů, konfiguraci sériové komunikace nebo inicializaci senzorů a periferií. Naproti tomu funkce void loop() běží opakovaně v nekonečné smyčce, dokud je Arduino zapnuté. Právě zde se nachází hlavní logika programu – zpracování vstupů, výstupů, výpočtů nebo řízení motorů. Díky tomuto cyklickému běhu může Arduino nepřetržitě reagovat na změny prostředí, například číst hodnoty ze senzorů a upravovat chování zařízení.

5.2.1 Další funkce void()

Kromě standardních funkcí void setup() a void loop() může uživatel v Arduinu definovat vlastní void funkce pro lepší organizaci kódu a opakovaně používané operace. Funkce deklarované jako void nevracejí žádnou hodnotu a obvykle slouží k provádění specifických úkonů, například ovládání motoru, zpracování senzorových dat nebo řízení LED diod. Ve funkcích void lze volat jiné void funkce, což umožňuje rozdělit kód na menší a přehlednější části. Díky tomu lze vytvářet modulární programy, kde každá funkce plní specifický úkol a může být snadno znovu použita v různých částech programu. Například hlavní smyčka void loop() může volat funkci void updateDisplay(), čímž se zjednoduší čitelnost a údržba kódu. Funkce mohou být také vnořené, což znamená, že jedna funkce void může volat jinou funkci void, například funkce void loop() může obsahovat volání funkcí void moveAllMotors() a void updateLCD(). Tato struktura pomáhá udržet kód organizovaný a efektivní.

5.3 Kód ovládání robota

Kód určený k ovládání robotického ramene je přiložen jako příloha k této práci. Obsahuje veškeré potřebné programové sekvence a funkce, které zajišťují správný chod celého systému – od základního řízení jednotlivých motorů až práci s uživatelským vstupem. V příloze je zahrnut kompletní zdrojový kód s komentáři, které vysvětlují jednotlivé části programu a jejich funkce. Díky tomu je možné snadné se v kódu orientovat a v případě potřeby kód dále upravovat nebo rozšiřovat o nové funkce.

5.3.1 Hlavní funkce programu

Kód na ovládání robotického ramene by se dal rozdělit na čtyři hlavní části. Nastavování hodnot, zpracování hodnot, ukládání hodnot a pohyb ramene. Jedná se jen o hrubé rozdělení, jednotlivé části se navzájem překrývají.

5.3.2 Popis kódu

5.3.2.1 Prvních 70 řádků

V prvních 70 řádcích jsou nadefinovány použité knihovny a proměnné využívané v celém kódu. Dále se zde označují piny pro enkodéry, krokové motory a koncové snímače. Jsou zde také nastavené rozměry LCD displeje. Poslední je zde nastavení struktury ‘PositionData‘. Tato struktura obsahuje proměnné, které se následně jako celek ukládají jako jednotlivé pozice.

5.3.2.2 Void setup()

Tato funkce je přibližně 35 řádků dlouhá a zprovozňují se v ní veškeré součásti robotického ramene. Jako první se nastaví frekvence sériového monitoru (pro debugging). Následně jsou nastaveny vstupní a výstupní piny enkodérů, koncových snímačů a servo motoru. Následuje zapnutí LCD displeje. Pak se určí kolika-početní kliknutí enkodérů co dělá. Jako poslední se volají funkce homing() a updateLCD().

5.3.2.3 Void loop()

Tato funkce slouží k řízení enkodéru, tlačítek a krokových motorů v systému. Nejprve aktualizuje stav rotačního enkodéru a jeho tlačítka. Poté sleduje změny polohy enkodéru a v případě detekce pohybu upraví hodnotu encoderValue (nastavovaná hodnota) a aktualizuje LCD displej. Pokud jsou motory v pohybu (motorsMoving == true), voláním run() posouvá jednotlivé osy, dokud všechny motory nedokončí svůj pohyb. Po zastavení motorů se uloží nové úhly jako současná poloha. Před koncem se zkontroluje, jestli je zmáčknuto tlačítko cyklování. Jestli je, spustí se funkce void cyklus(). Na konec probíhá zpracování vstupů enkodéru na ukládání poloh. Je zde volána funkce hadleStoreEnkoder() která případné změny na enkodéru zpracovává.

5.3.2.4 Void calculatePath@()

Funkce calculatePath@() neexistuje. Jedná se pouze o označení čtyř téměř identických funkcí, calculatePathX(), calculatePathY(), calculatePathZ() a calculatePathA(). Dále v textu je @ zástupce za písmena X, Y, Z a A.

Každá z funkcí calculatePath@() je 38 řádků dlouhá. Slouží k výpočtu optimální dráhy mezi současnou a nastavenou polohou s ohledem na tzv. zakázanou oblast. Nejprve normalizuje úhly currentAngle@ (současná poloha, jako úhel) a targetAngle@ (cílová poloha, jako úhel) tak, aby se pohybovaly v rozsahu 0 až limit@. Poté definuje dva pomocné body – jeden pro pohyb ve směru hodinových ručiček a druhý pro protisměrný pohyb. Vypočítá rozdíly mezi současným a cílovým úhlem v obou směrech a určí, která trasa je kratší. Pak zkontroluje, zda trajektorie k cíli prochází zakázanou oblastí. Výstupem funkce jsou dvě hodnoty – firstLeg@ a secondLeg@, které určují rozdělení pohybu do dvou segmentů.

Oblast, ve které se motor otáčí si můžeme představit jako kružnici. Na této kružnici jsou dva body, současná a cílová poloha, plus zakázaná oblast. Funkce spočítá nejkratší cestu. Jestliže nejkratší cesta prochází zakázanou oblastí, vypočítá se na delší cestě pomocný bod.

Ten leží přesně na půli cesty. Pak až se motory začnou pohybovat, motor nejdřív pojede na pomocný bod a pak následně na cílový. Tím se zabrání cestování zakázanou oblastí.

Zakázaná oblast je definována z konstrukčních důvodů. Bez definování a zohlednění zakázané oblasti by se mohlo robotické rameno protáčet, narážet do koncových bodů a následně si zpřetrhat vodiče.

5.3.2.5 Void moveAllMotors()

Funkce moveAllMotors() je dlouhá 45 řádků. Zajišťuje řídí souběžný pohyb všech os robotického ramene. Nejprve ověří, zda se motory již nepohybují, a pokud ne, nastaví stav proměnné motorsMoving (indikátor, zda se motory pohybují) na pravda. Poté vypočítá optimální trajektorii pro každou osu voláním funkcí calculatePath@(), které rozdělí pohyb do dvou fází. Následně pošle první sadu pohybových instrukcí (firstLeg@) krokovým motorům a čeká, dokud všechny motory nedokončí svou činnost. Po dokončení prvního úseku se provede druhá část pohybu (secondLeg@). Nakonec se aktualizují hodnoty currentAngle@, aby odpovídaly novým pozicím motorů, a proměnná motorsMoving bude nastavena zpět na falše, čímž signalizuje ukončení pohybu. [4]

5.3.2.6 Void changeActiveAngle()

Funkce changeActiveAngle() je dlouhá 23 řádků. Má za úkol cyklicky přepínat mezi pěti různými osami robotického ramene (X,Y,Z,A,W). Nejprve inkrementuje proměnnou activeAngle, přičemž zajistí, že po přesažení hodnoty 4 se vrátí zpět na 0. Poté pomocí switch podmínky nastaví currentLimit@ podle aktivní osy – tedy přiřadí maximální povolenou hodnotu úhlu odpovídající dané ose. Nakonec zavolá updateLCD(), čímž aktualizuje displej, aby zobrazoval změněnou aktivní osu. Funkce umožňuje uživateli jednoduše přepínat mezi nastavovanými osami.

5.3.2.7 Void writeEncoderValue()

Funkce writeEncoderValue() je tvořena třinácti řádky. Přiřazuje aktuální nastavovanou hodnotu enkodéru (encoderValue) k jedné z pěti os robotického ramene podle hodnoty activeAngle. Pomocí série if-else podmínek určí, která osa je právě aktivní, a uloží do ní aktuální nastavovanou hodnotu enkodéru. Pokud je aktivní úhel 0, aktualizuje se angleX, pokud 1, nastaví se angleY, a analogicky až po angleWrist při hodnotě 4. Nakonec funkce zavolá updateLCD(), aby zobrazila nové hodnoty na LCD displeji. Díky této funkci uživatel interaktivně nastavuje úhly jednotlivých os pomocí enkodéru a sleduje změny v reálném čase.

5.3.2.8 Void handleStoreEncoder()

Funkce handleStoreEncoder() je 12 řádků dlouhá a stará se o změnu a načítání uložených pozic pomocí enkodéru. Nejprve zjistí aktuální pozici enkodéru a porovná ji s předchozí hodnotou. Pokud se pozice změnila, spočítá rozdíl (delta) a aktualizuje proměnnou currentPosition (současná pozice), přičemž se postará o to, aby zůstala v platném rozsahu pomocí modulo operace. Pak načte odpovídající uloženou polohu voláním funkce loadPosition(currentPosition) a aktualizuje LCD displej. Funkce tak umožňuje enkodérem přepínat mezi uloženými pozicemi ramene z paměti EEPROM.

5.3.2.9 Void saveCurrentPosition()

Funkce saveCurrentPosition() slouží k uložení aktuální pozice robotického ramene. Nejprve vytvoří strukturu currentPositionData, do které uloží aktuální hodnoty všech klíčových parametrů – úhly motorů X, Y, Z, A, úhel zápěstí W, stav čelistí JAW a platnost pozice validPosition (při ukládání vždy true). Tato data se pak uloží na konkrétní místo v paměti pomocí funkce savePosition(), přičemž index pozice je určen proměnnou currentPositionIndex. Pro debugging funkce vypíše do sériového monitoru zprávu "Hodnoty uloženy do pozice [index]". Nakonec zavolá updateLCD(), aby se změny okamžitě promítly na displeji. Navíc tato funkce je jednoduchá a přehledná, obsahuje pouze 4 řádky kódu.

5.3.2.10 Void savePosition ()

Funkce savePosition() slouží k uložení dat o pozici do paměti EEPROM, která uchovává data i po vypnutí zařízení. Nejprve vypočítá konkrétní adresu v paměti pomocí vzorce index * sizeof(PositionData), čímž zajistí, že každá uložená pozice má svůj vlastní vyhrazený blok paměti a nedojde k přepsání jiných dat. Poté pomocí funkce EEPROM.put() uloží strukturu data na vypočítanou adresu. Tento přístup umožňuje snadné ukládání a pozdější načítání předem definovaných pozic robotického ramene. Navíc je efektivní, obsahuje jen 3 řádky kódu.

5.3.2.11 Void resetCurrentPosition()

Funkce resetCurrentPosition() slouží k resetování aktuálně vybrané pozice robotického ramene na výchozí hodnoty. Nejprve vytvoří strukturu resetData, kde jsou všechny úhly nastaveny na 0 a stav čelisti na nepravda (tedy zavřené). Také je zde proměnná validPosition nastavena na nepravda. Ta je kritická ve funkci void cyklus(). Poté tuto "nulovou" pozici uloží do EEPROM pomocí funkce savePosition(), přičemž využívá aktuální index pozice

(currentPositionIndex) k určení správného místa v paměti. Následně aktualizuje currentPositionData tak, aby odpovídala resetovaným hodnotám, a na závěr zavolá updateLCD() pro zobrazení změn na displeji. Tím uživatel okamžitě vidí, že pozice byla úspěšně resetována.

5.3.2.12 PositionData loadPosition()

Funkce loadPosition() slouží k načtení uložené pozice robotického ramene z paměti EEPROM. Nejprve vypočítá adresu v EEPROM pomocí vzorce index * sizeof(PositionData), což zajistí, že každá uložená pozice má v paměti svůj specifický prostor podle velikosti datové struktury PositionData. Poté vytvoří proměnnou loadedPositionData, do které načte uložená data pomocí funkce EEPROM.get() z vypočítané adresy. Nakonec tuto načtenou strukturu vrátí jako výsledek funkce.

5.3.2.13 Void updateLCD()

Tato funkce obsahuje 38 řádků kódu a její úlohou je vizuální zpětná vazba na LCD displeji. Používané LCD má 4 řádky a každý řádek má místo pro 20 znaků. V prvním řádku se zobrazují informace o aktuálně vybraném motoru, jeho současné a budoucí poloze. Druhý řádek zobrazuje informace o enkodéru, který nastavuje úhly motorů a současné nastavované hodnotě. Tato hodnota se pak zapíše jako budoucí poloha motoru po dvojitém zmáčknutí enkodéru. Třetí řádek obsahuje informace o aktuální pozici ukládání do paměti EEPROM. Poslední řádek pak ukazuje současně uložené hodnoty ze současně aktivní pozice. Jelikož je uložených hodnot hodně a všechny by se na jeden řádek nevešly, jsou informace rozděleny na polovinu. Přepínání mezi zobrazenými polovinami zajišťuje funkce void toggleDisplay().

5.3.2.14 Void toggleDisplay()

Tato funkce je velmi krátká a efektivní, obsahuje jen čtyři řádky kódu. Funkce toggleDisplay() slouží k přepínání mezi dvěma sadami zobrazovaných hodnot na LCD displeji. Používá logickou negaci showFirstSet = !showFirstSet;, čímž přepíná stav této proměnné mezi pravda a nepravda. Následně vypíše zprávu "Zobrazeni prepnut" do sériového monitoru pomocí Serial.println(), což slouží k debuggingu. Nakonec zavolá funkci updateLCD(), která aktualizuje LCD displej tak, aby zobrazoval nově přepnutou sadu hodnot.

5.3.2.15 Void homing()

Funkce homing() má celkem 65 řádků kódu a slouží ke kalibraci výchozí polohy všech čtyř os robotického ramene (X, Y, Z, A) pomocí koncových spínačů (limit switches). Nejprve

nastaví nízkou maximální rychlost a akceleraci pro každou osu, aby byl pohyb při hledání referenční polohy bezpečný a přesný. Poté instruuje motory, aby se pohybovaly směrem k záporným hodnotám s cílem narazit na koncové spínače.

V cyklu while funkce kontroluje stav jednotlivých koncových spínačů. Jakmile motor některé osy spustí spínač (digitalRead(limitSwitch@) == LOW), daná osa se zastaví, pozice se nastaví na 0, motor se mírně odtáhne zpět a při změně (digitalRead(limitSwitch@) == HIGH), opět nastaví pozici na 0 pro přesnější kalibraci. Tento proces se opakuje pro všechny osy. Po dokončení kalibrace funkce vrátí původní (vyšší) rychlosti a akcelerace (1000) pro běžný provoz. [5]

5.3.2.16 Void cyklus()

Funkce cyklus() slouží k postupnému načítání a vykonávání předdefinovaných pozic pro ovládání motorů. Na začátku se proměnná currentPositionIndex (současná uložená pozice) nastaví na hodnotu -1, aby se při prvním volání funkce loadPosition() inkrementovala na 0. První “pravá“ pozice je 0. Dále se inicializuje struktura previousPositionData s výchozími hodnotami, slouží k debuggingu. Funkce následně vstoupí do nekonečné smyčky while (true), kde pomocí funkce loadPosition() načítá jednotlivé pozice. Index se před každým načtením zvyšuje o jedničku. Pokud načtená pozice (positionData) není platná, smyčka se ukončí příkazem break. Platnost pozice určuje proměnná validPosition. V případě platné pozice se hodnoty úhlů (angleX, angleY atd.…) nastaví podle aktuální pozice a zavolá se funkce moveAllMotors(), která posune motory do nové pozice. Po pohybu se aktuální úhly (currentAngleX, currentAngleY atd....) aktualizují podle provedeného kroku. To je potřebné, jinak by se každá další pozice inkrementálně přičítala k té předchozí. Funkce tak zajišťuje sekvenční vykonávání sady pozic, dokud nejsou všechny platné pozice zpracovány.

6. Plány do budoucna

Ač projekt dopadl lépe, než jsem mohl doufat, stále jsou věci, které bych na něm vylepšil. Mezi hlavní, patří čelisti a jejich modifikace. Zvažuji možnost navrhnout čelisti jako modulární, což by umožnilo snadnou výměnu pouhým odšroubováním a připevněním alternativních čelistí.

Další věc, co by šla vylepšit je uživatelské rozhraní. Současné není vůbec špatné, ovšem jsou chvíle, kdy se člověk překlikne a tím si zmaří svoji práci.

7. Závěr

Cíl, navrhnout a sestrojit robotické rameno, byl úspěšně splněn. Veškeré elektrické a hliníkové komponenty jsem si na internetu vyhledal a nakoupil sám. Zbylé součásti jsem nechal vytisknout na 3D tiskárnách svých přátel. Za tyto služby jsem jim zaplatil a tím podporoval lokální ekonomiku. V drtivé většině případů, dohledatelných na internetu, stavby arduino robota slouží tento mikropočítač jako prostředník, ovládaný počítačem. Pomocí tohoto projektu jsem demonstroval, že arduino zvládne pracovat jako samostatná jednotka.

Po splnění minimálního zadání jsem pokračoval v projektu. Fyzickou stránku jsem stylizoval to hranatého designu. Software část jsem zjednodušil, zpřehlednil a zefektivnil abych maximalizoval místo na ukládání pozic.

Jako hlavní výhodu tohoto projektu beru zkušenosti, které jsem během práce na tomto projektu nasbíral. Znalost elektrotechniky, CAD designu a programování se mi zcela jistě budou v budoucnosti hodit. Ač nastala všelijaká úskalí, všechny problémy jsem dříve či později vyřešil.

8. Bibliografie

[1] D. DEBASHIS, „Stepper Motor controlled with A4988 stepper motor driver and arduino UNO,“ 8 břena 2023. [Online]. Available: https://circuitdigest.com/microcontroller-projects/interface-a4988-stepper-motor-driver-with-arduino.

[2] DEJAN, „Stepper Motors and Arduino,“ 15 května 2022. [Online]. Available: https://www.youtube.com/watch?v=7spK_BkMJys.

[3] T. PALOVAARA, „3D Printed 6DoF Arduino Robot arm,“ 2 listopad 2019. [Online]. Available: https://www.youtube.com/watch?v=f1F-kCW1iYs&list=PLun7tMRgC8_ZF3ugJWF9CB2DSWxfFAtst.

[4] WASPINATOR, „AccleStepper,“ 22 května 2022. [Online]. Available: https://github.com/waspinator/AccelStepper.

[5] IVAN, „Homing stepper motors using accelstepper library,“ 24 října 2020. [Online]. Available: https://www.brainy-bits.com/post/homing-stepper-motors-using-the-accelstepper-library.

[6] K. Láska, „LaskaKit,“ [Online]. Available: https://www.laskakit.cz/. [Přístup získán 3 3 2025].

[7] I. d. a. thing, „Youtube / I did a thing,“ 6 8 2022. [Online]. Available: https://www.youtube.com/watch?v=0rliFQ0qyAM. [Přístup získán 3 3 2025].

9. Seznam obrázků a tabulek

Obrázek 1 Reprezentace mobilního robota [7] 8
Obrázek 2 Krokový motor NEMA 17 [6] 9
Obrázek 3 Schéma rotoru a statoru krokového motoru [1] 10
Obrázek 4 Pinout driveru krokového motoru [1] 11
Obrázek 5 Pinout CNC shieldu V3 [6] 12
Obrázek 6 Arduino MEGA se zapojeným CNC shieldem a drivery 13
Obrázek 7 Ilustrace Arduino MEGA [6] 13
Obrázek 8 Servo MG995 s příslušenstvím [6] 14
Obrázek 9 Model převodu 16:1 v SolidWorks 15
Obrázek 10 Loketní kloub, řemen je zneviditelněný 16
Obrázek 11 Zápěstní kloub se servo motorem 17
Obrázek 12 Logo Arduino IDE 2 18
Obrázek 13 Programovací prostředí Arduino IDE 2 19

Tabulka 1 Mikrokrokování driveru krokového motoru 12