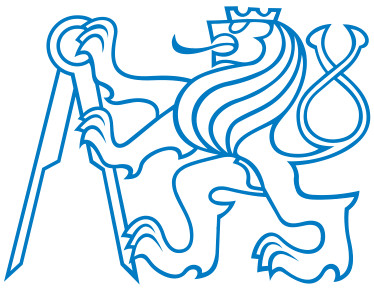


ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra telekomunikační techniky

Demonstrační model průmyslového řízení

Demonstration Model of the Industrial Control

Bakalářská práce

Studijní program: Elektronika a komunikace Vedoucí

práce: doc. Ing. Jiří Vodrážka, Ph.D.

Tomáš Tomašica

Srpen 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Tomašica** Jméno: **Tomáš** Osobní číslo: **466109**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Demonstrační model průmyslového řízení

Název bakalářské práce anglicky:

Demonstration Model of the Industrial Control

Pokyny pro vypracování:

Navrhněte a realizujte demonstrátor vybraného systému (úsek výrobní linky, zakladač, apod.). K realizaci použijte některou z polytechnických stavebnicí (např. MERKUR. LEGO) nebo vlastních vyrobených dílů (např. na 3D tiskárně). K řízení použijte platformu IPLOG společnosti METEL. Vypracujte metodiku a návod, který poslouží dalším studentům pro realizaci dalších projektů s danou platformou.

Seznam doporučené literatury:

[1] IPLOG-GAMA - průmyslové PLC s OS Linux a IO moduly. METEL. Dostupné na: https://www.metel.eu/cz/newdesign/products?itemId=225 [on-line]

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Jiří Vodrážka, Ph.D., katedra telekomunikační techniky FEL

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

Datum zadání bakalářské práce: **08.01.2020** Termín odevzdání bakalářské práce: **14.08.2020**

Platnost zadání bakalářské práce: **30.09.2021**

doc. Ing. Jiří Vodrážka, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. Datum převzetí zadání

Podpis studenta

Prohlášení

„Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.“

Praha, Srpen 2020

Abstrakt

Tato práce se zabývá návrhem a zpracováním modelu průmyslového řízení za použití vývojové platformy METEL IPLOG. Řeší se zde problematika návrhu konstrukce a její provedení. Také je zde podrobně popsáno základní nastavení vývojové platformy METEL IPLOG a popis programovacího prostředí s ukázkou kódu.

Klíčová slova: METEL, IPLOG, Průmyslové řízení, Demonstrační model Zakladač, Krokové motory,

In this thesis I will discuss design and creation of Model of the Industrial Control. We are using platform METEL IPLOG. We will discuss problems of construction and its solutions. In this thesis is also described basic settings of platform METEL IPLOG and a description of programming environment with code examples.

Key words: METEL, IPLOG, Industrial Controll, Demonstration Model, Stacker, Stepper motor

Obsah

Seznam obrázků

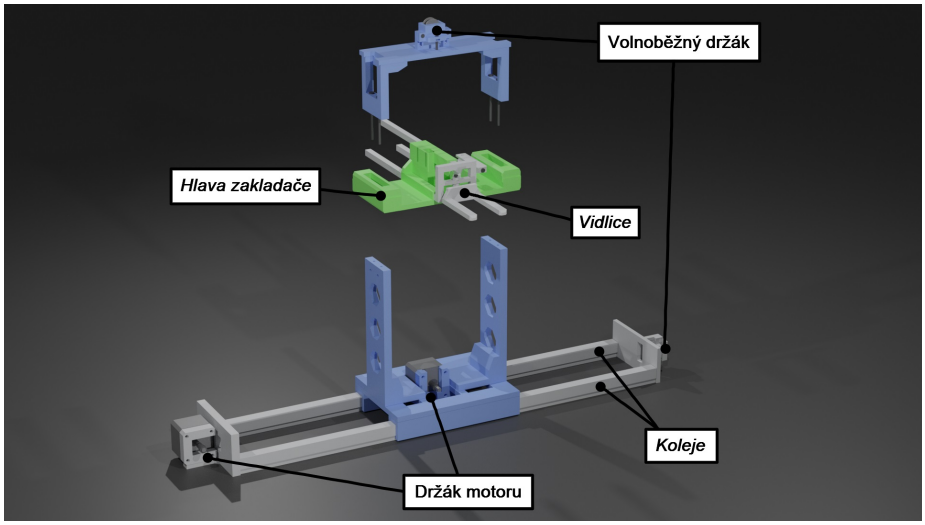
Úvod

Zadáním práce je demonstrační model průmyslového řízení. Tento demonstrační model musí využívat jako hlavní řídící platformu IPLOG of firmy METEL [1]. Vybral jsem jako demonstrační model zakladač, který přemisťuje palety na předem určené platformy. Modrá platforma je vstupní, zelená a žlutá jsou výstupní. Volba výstupní platformy je realizována pomocí dvou tlačítek.

Pro realizaci projektu je potřeba nejprve navrhnout celkovou konstrukci, které se věnuje první kapitola. Jednotlivé díly je potřeba vyrobit, čemuž se věnuje druhá kapitola. Třetí kapitola se zabívá teorií ohledně elektronických součástek. Čtvrtá a pátá kapitola popisuje desku plošných spojů, speciálně navrženou pro potřeby zakladače. Software pro arduino a jeho zapojení je popsáno v šesté kapitole. Sedmá kapitola popisuje základní nastavení platformy METEL IPLOG, s navazujívím popisem hlavního programovatelného prostředí v osmé kapitole. Devátá kapitola slouží jako ukázka vlastního provedení programu pro zakladač.

1 Návrh konstrukce

Zakladač se musí pohybovat ve dvou osách, vodorovné X a svislé Z. Pro pohyb po vodorovné ose je využit krokový motor umístěný na konci paralelních kolejí. Každá kolej se skládá ze dvou dílů slepených k sobě. Na protější straně je použita volnoběžná řemenice GT2 20 zubů a 5mm dírou. Na straně motoru je použitá řemenice GT2 12 zubů s 5mm dírou. Řemen je provléknutý svislými stěnami na konci kolejnic, kolem motoru a volnoběžné řemenice. Na pojízdné plošině je ukotven pomocí malých držáků.



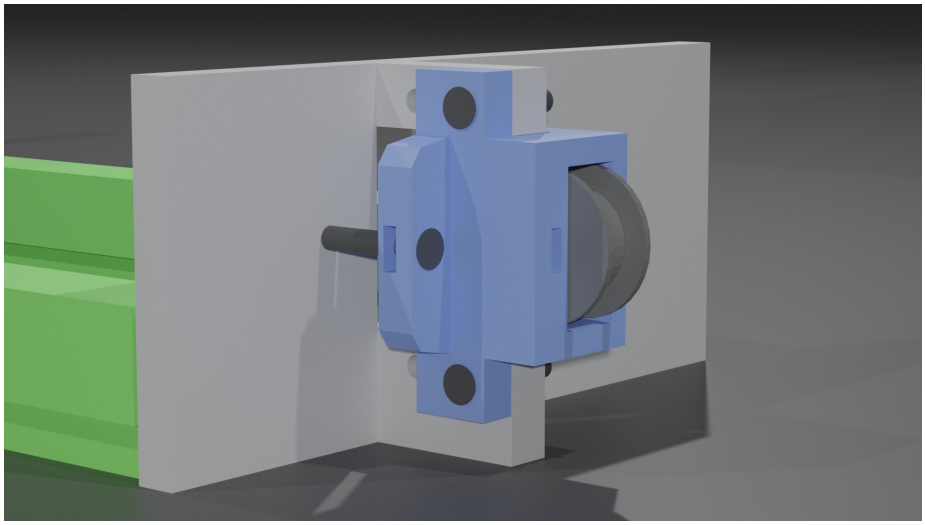
Obrázek 1: Konstrukce zakladače

Vodorovná osa je navržena podobně. Jsou použity stejné řemenice, motor i řemen. Řemen prochází skrze kanál v hlavě zakladače. Svislá osa je složena ze dvou dílů sešroubovaných pomocí čtyř šroubů, pro možnost vyjmutí a úpravu hlavy konstrukce. Základna zakladače je z jednoho dílu. Vidlice je ovládána ramenem, které je připevněno na krokový motor 28BYJ-48. Na obrázku 1 je celá konstrukce zakladače, kde je barevně odlišena nehybná část konstrukce, pojízdná plošina, hlava konstrukce a vidlice.

1.1 Volnoběžný držák

Volnoběžný držák znázorněný na obrázku 2 slouží i pro napínání řemenu. Protože jsou použity 3mm šrouby pro upevnění volnoběžné řemenice a průměr řemenice je 5mm, je zde malá redukce. Šroub, který drží řemenici nesmí být příliš utažený, protože by vytvořil brzdu a pohyb zakladače by nebyl hladký popřípadě by se nemohl vůbec pohybovat. Řemen musí být dostatečně napnutý, aby nezpůsoboval nepřesnosti v pohybu a opožděnou reakci konstrukce vůči otočení motoru. V držáku jsou vloženy dvě matice, které mohou při demontáži vypadnout. Pro

udržení držáku napnutého jsou po stranách v kolejnicích dva šrouby, které jsou na druhé straně upevněny matkou. Doporučuji při utahování těchto šroubů přidržet matku kleštěmi. Napínací šroub by neměl být dlouhodobě pod tlakem nebo pod velkým jednorázovým tlakem, hrozí zde rozbití dílu. V případě, že nejsou dva krajní šrouby dobře dotažené hrozí naklonění celého držáku a způsobení špatný pohyb řemenu.



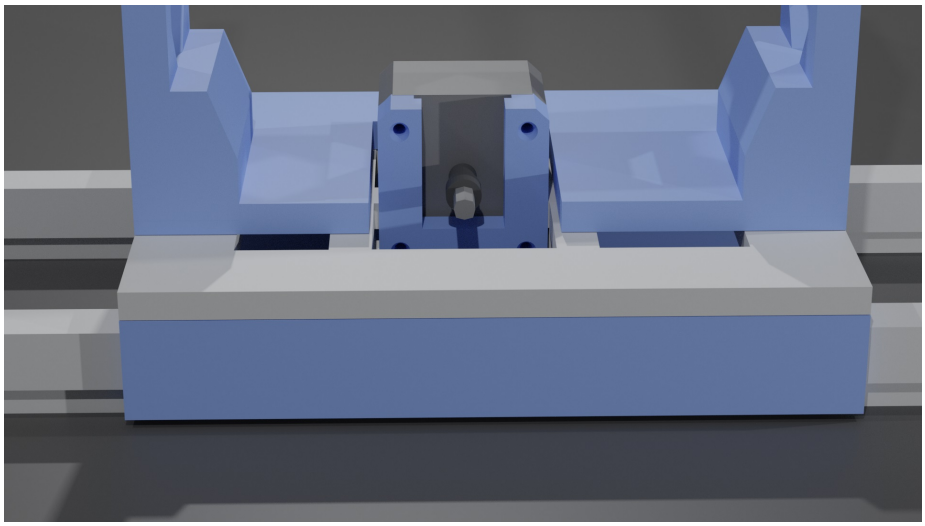
Obrázek 2: Volnoběžný držák

1.2 Držák motoru

Oba motory mají držák ve tvaru C (viz obrázek 3), který je připevněný k motoru čtyřmi šrouby a přilepený ke zbytku konstrukce. V případě, že pohyb motorů začne vytvářet hlasitý či neobvyklý zvuk, je možné že se tento držák uvolnil a je třeba přitáhnout šrouby nebo znovu přilepit ke konstrukci. Šrouby pro motor pro svislou osu jsou velice špatně přístupné, proto je upevněný pouze na třech šroubech.

1.3 Ovládání vidlice

Vidlice je umístěna v hlavním dílu hlavy ve třech čtvercových otvorech. Pohyb vidlice zajišťuje rameno složeno ze dvou dílů. Rameno má dva klouby vytvořené ze šroubu a matice, která má tendenci se používáním povolovat. Šrouby nesmí být příliš utažené, mohli by zamknout otáčení a tím znemožnit pohyb vidlice. Motor pro pohyb vidlice nemá velký výkon, je třeba dbát na to, aby neměl v pohybu překážku či jiné omezení pohybu.



Obrázek 3: Pojízdná plošina s motorem osy Y

1.4 Regály a palety

Jako náklad pro zakladač byla vytisknuta paleta s krabičkou o rozměrech 7 cm x 7cm x 6,5cm. Tato paleta váží zhruba 100g. Úložné prostory pro paletu jsou tři, modrý vstupní a Zelený se Žlutým jako výstupní. Zelený držák je vyvýšen o 10,3 cm. Tyto držáky jsou umístěny na dřevěné podložce, která je na pěti pilířích.

1.5 Ostatní díly

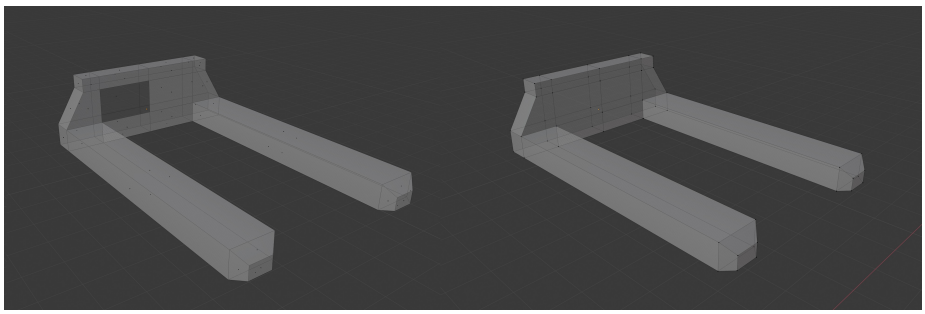
Mimo hlavní konstrukce jsou zde navrženy ještě držáky řídicí jednotky, které mají zespodu dva šrouby pro připevnění k IPlogu a jsou přilepeny na dřevěnou podložku. Dalším dílem je dutá krabička se dvěma tlačítky pro ovládání zakladače. Tlačítka jsou barevně odlišeny a toto barevné označení naznačuje, kam zakladač přemístí paletu. Arduino je přišroubováno na držák který je následně přilepen k podložce. Posledním dílem je kryt na hlavní vypínač, který zapíná proud do celého systému.

2 Výroba dílů

Díly byli vymalovány, následně zpracovány slicerem a poté vytištěny na 3D tiskárně Prusa MK3s.

2.1 Modelování dílů

Jednotlivé díly byli vytvořeny ve 3D programu Blender. Alternativou může být Fusion 360 od Autodesk, který je také zdarma ale pouze na dva roky. Použil jsem software Blender, dostupný z webových stránek https://www.blender.org/ . Aby bylo možné díly vytisknout, musí být každý díl vodotěsný, což znamená, že neobsahuje žádnou díru či jinou vadu v mříži. Na obrázku 4 je vpravo znázorněna dobře uzavřená vodotěsná mříž, která je vhodná pro další tisk. Vlevo je mříž, která obsahuje díru, tudíž jí nelze použít pro tištění. Každý díl byl konstruován aby vyžadoval co nejméně podpěr při tisku. Také je potřeba, aby každý díl měl přesné rozměry pro šrouby a všechny pohyblivé části.



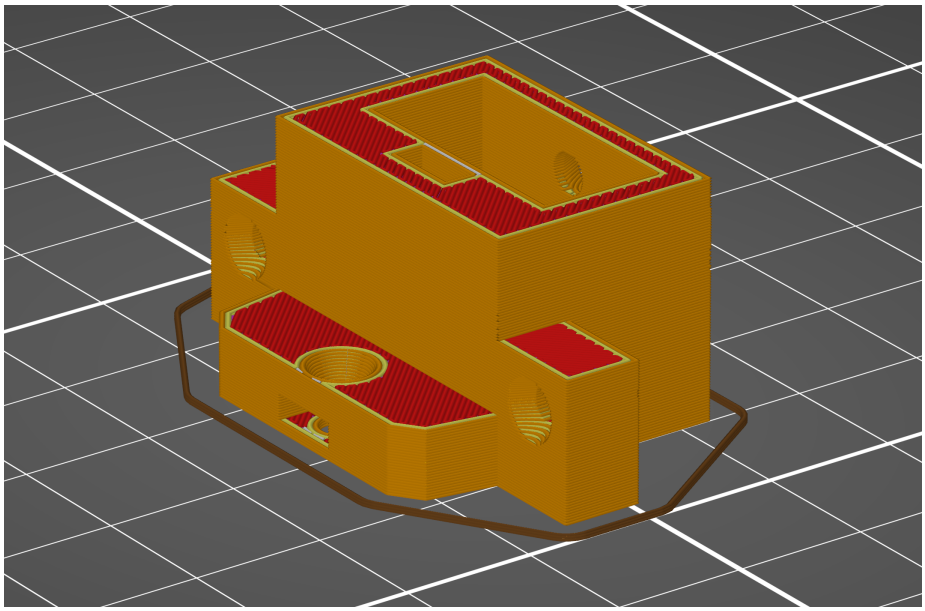
Obrázek 4: Ukázka dobře uzavřené mříže (vpravo) a mříže s dírou (vlevo)

Hlavním nástrojem pro modelování je boolean modifikátor, který nám umožní odečíst a nebo sečíst objekty. Pokud tedy potřebujeme například díru na šroub, vymodelujeme šroub a pomocí boolean operátoru ho odečteme od objektu a tím vytvoříme díru. Při používání boolean modifikátorů je třeba dávat zvýšený pozor na změnu mříže, může se nám zde vytvořit díra, nebo při špatné topologii odečítaného objektu může mít výsledný objekt nežádoucí artefakty.

Dalším velmi užitečným modifikátorem je mirror, který nám zrcadlí mříž v reálném čase v jakékoliv zvolené ose.

2.2 Tisk dílů

Všechny díly byli vytištěny na 3D tiskárně Prusa. Byl použit filament PLA. Výška vrstev se pohybuje od 0,2 mm do 0,3 mm a průměr trysky 0,4 mm. Pro převod z STL formátu do G kódu, který lze zpracovat tiskárnou, byl použit program Prusa Slicer. Slicer je program, který naplánuje každý pohyb tiskárny tak, aby výsledný produkt co nejpřesněji odpovídal vytvořenému modelu. Lze zde nastavit teplota u tisku, výška u vrstvy, rychlost, podpěry, teplota u podložky, druh materiálu a jiné. Více informací ohledně využité tiskárny je dostupných z [8]. Na obrázku 5 je zobrazen držák napínacího kolečka. červeně jsou znázorněny ploché výplně a oranžovou barvou stěny objektu. Výplň objektu je dutá se čtvercovou mřížkou. Jedním z problémů byla velikost tiskového prostoru. Díly větší než 220mm museli být tištěny po částech a následně slepeny.



Obrázek 5: Ukázka dílu po zpracování slicerem

2.3 Princip 3D tisku

Tisková hlava s horkým koncem, zvaným hotend, se velice přesně pohybuje ve třech osách. Tiskový struna zvaný filament se postupně vtlačuje do hot endu, kde díky teplotě kolem 215 stupňů Celsia se plast roztaví a nanese na tiskovou podložku. Výsledný objekt se tiskne po vrstvách o předem definované výšce. Problém při tisku nastává v případě tisku mostů nebo objektu, pod kterým je vzduch. Abychom docílili co nejefektivnějšího tisku, jednotlivé díly jsou designované aby vyžadovali co nejméně podpěr. Podpěry se nazývají věže, které tiskárna tiskne již od začátku a jsou odstraněny po dokončení tisku.

3 Elektronické komponenty

3.1 Krokové motory

K pohybu os X a Z jsou využity krokové motory SX17-1003LQEF. Krokový motor je synchroní stroj, který se otáčí v krocích. Magnetické pole v motoru je generováno vždy napájením jednotlivých pólových dvojic. Motory SX17 jsou hybridní dvoufázové motory s točivým momentem 0,3 Nm. Mají 200 kroků na jednu otočku a tím i délku kroku 1,8 stupňů. Motor má jmenovitý proud 1A.

Pro pohyb osy vidlice je použit motor 28BYJ-48. Tento motor je čtyřfázový. Součástí tohoto motoru je také driver. Motor 28BYJ-48 má točivý moment 34 mNm. Na jednu otočku má pouze 64 kroků a tím 5,625 stupně na jeden krok.

Jmenovitý proud motoru je 500mA. Více informací ohledně krokových motorů je dostupných z [6].

3.2 Drivery krokových motorů

K řízení krokových motorů a proudovému zesílení jsou použity drivery A4988 firmy Pololu. Je zde možnost nastavení režimu full step, half step, quarter step, eighth step a sixteenth step. Jednotlivý režim se nastavuje pomocí pinů MS1, MS2 a MS3.

Běžným řízením krokových motorů je počet impulsů na vstupním pinu STEP. Je pak možné přesně řídit otočení motoru. V této aplikaci ale bohužel toto řešení nemohlo být využito kvůli veliké době změny výstupu IPlogu. Nejsnažším řešením je tedy vytvoření externího zdroje těchto impulsů a ovládání zapínání a vypínání tohoto signálu. Tím pádem je i dynamické nastavování režimu kroku zbytečné a po testování je nastaveno na šestnáctinu kroku kvůli hlučnosti a vibracím. Pin DIR ovládá směr otáčení.

Více informací ohledně driverů pro motory je dostupné z [5].

3.3 Optické senzory

K měření vzdálenosti jsou použity senzory SHARP GP2Y0A41SK0F. Tento senzor je schopný měřit vzdálenost 4 cm až 30 cm. Tento senzor funguje na principu měření úhlu odrazu od objektu. Tento paprsek je realizován fotodiodou a fotosnímačem. Aby bylo zajištěno bezchybné snímání, je výstupní signál tohoto senzoru přiveden přes 820 Ohmový rezistor na zem a tím vytváří permanentní proud tímto senzorem. Více informací ohledně optických senzorů je dostupné z [4].

3.4 Tlačítka

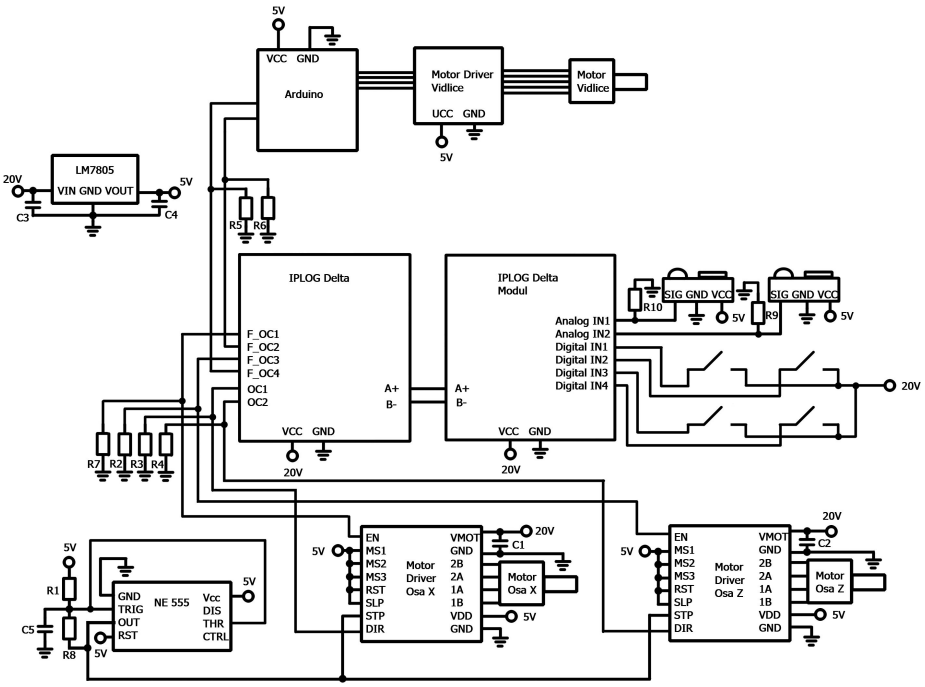
Každé tlačítko je připojeno na přímo do modulu IPLOGu. Tlačítko spíná 20V zdroj napětí přímo na vstup. Logická 0 je ošetřena na vstupu IPLOGU, takže zde není potřeba žádný pull down resistor.

4 Blokové schéma

Na obrázku 6 je znázorněno blokové schéma zapojení. Jako zdroj 20V napájení je použit síťový adaptér Lenovo 20V 2,25A. Vývojová platforma METEL IPlog má maximální příkon 5W. Stabilizátor má maximální příkon 10W. Jednotlivé motory mají příkon 8,8W. Maximální příkon zakladače je tedy 37,6W. Vzhledem k tomu že síťový adaptér může dodávat až 45W, můžeme ho využít.

Osa X a Z se skládá z motoru a driveru. Pro pohyb vidlice je využito arduino, driver pro motor 28BYJ-48 a samotný motor. Arduino je zde kvůli potřebě rychlých signálů pro řízení tohoto motoru. Každá osa je řízena dvěma signály, které jsou připojeny přes Pull down resistor. Jeden z těchto signálu je EN,

který udává, zda má být motor v pohybu nebo ne a druhý signál je DIR, který ovládá jakým směrem se má motor otáčet. Spojení mezi IPLOGem a Arduinem



Obrázek 6: Blokové schéma

je jednocestné stejného charakteru jako drivery pro osy X a Z. Jeden vodič nese informaci o zapnutí nebo vypnutí motoru a druhý nese informaci o směru otáčení motoru.

Tlačítka nepotřebují žádný resistor k jejich funkčnosti díky charakteru IPlog Vstupu. Jsou tedy přímo připojeny na napájecí napětí 20V, které nám vytvoří logickou 1 a v případě rozpojení je na vstupu IPlogu logická 0.

Oscilátor je vytvořen pomocí NE555, který generuje pulsy pro oba drivery a tím mění charakter chování motoru.

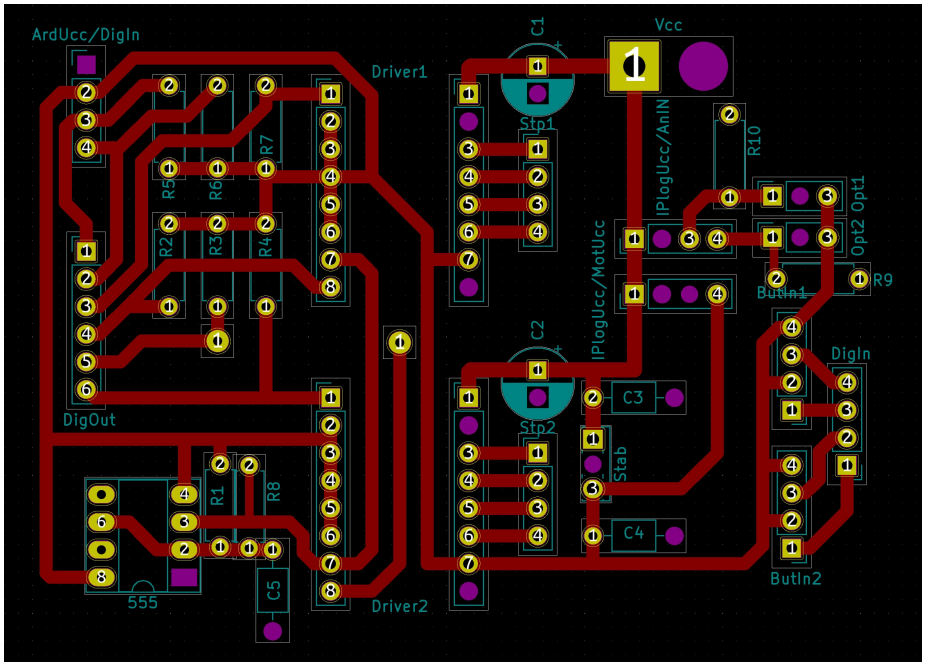
Optické senzory generují analogový výstup v rozmezí napětí 4V - 0,5V. Toto napětí je úměrné pozici zakladače.

5 Deska plošných spojů

Pro propojení všech komponent a umístění Driverů s oscilátorem byla navržena deska plošných spojů. Fialové plošky jsou připojeny na zem. Kvůli přehlednosti není na obrázku zobrazena vrstva země. Byli vyleptány pouze kanálky kolem tras a zbytek desky tvoří zem. Všechny díry byli vyvrtány 1,5mm vrtákem. Deska plošných spojů je znázorněna na obrázku 7.

Hodnoty jednotlivých součástek jsou dány tabulkou.

Součástka	Hodnota
R1	800k
R2 - R5	180k
R6, R7	12k
R8	4,7k
R9,R10	820
C1, C2	47uF
C3 - C5	100nF



Obrázek 7: Návrh desky plošných spojů

5.1 Drivery

Ve slotech Driver 1 a Driver 2 jsou osazeny drivery krokových motorů Pololu A4988. Pin Enable je ovládán z výstupu tranzistoru IPlogu, kde R7 resp R4 tvoří pull down rezistory. R4 a R7 mají menší hodnotu oproti ostatním pull down resistorům, protože driver v sepnutém stavu odebírá v tomto pinu vyšší proud než v ostatních pinech. Piny MS1, MS2, MS3 jsou napevno v logické jedničce. Tím je motor v režimu Sixteenth step (šestnáctina kroku). Tento režim byl zvolen kvůli redukci vibrací při pohybu, protože se motor pohybuje diskrétně v těchto krocích a se snižující se vzdáleností tohoto kroku se i snižuje hlučnost motoru. Nevýhodou tohoto režimu je pomalejší posun konstrukce, ale vzhledem k řízení polohy pomocí vnějšího snímání nám tento pohyb snižuje riziko chyby.

Funkčnost pinů RESET a SLEEP projekt nevyužívá. Piny jsou také invertované, proto je na jejich vstup přivedena logická 1, tím destička nepřechází do režimu sleep ani reset. Pin STEP je ovládán z oscilátoru, který permanentně dodává impulsy do driveru místo mikrokontroleru a tím otáčí motorem pokud je pin EN zapnutý (resp vypnutý, protože je pin EN invertovaný) Pin DIR je připojený přes pull down resistor na výstupní tranzistor IPlogu.

Na protější straně je pin VMOT připojen na napájecí napětí, vedle kterého je kondenzítor C1 resp. C2, který zamezuje kolísání vstupního napětí a napěťovým špičkám které by mohly poškodit driver. Pin GND je připojený na zem. Piny pro výstup motoru jsou propojeny s konektorem, do kterého se připojí daný krokový motor. Předposlední pin je pro napájení logických obvodů na driveru, tudíž je na něj připojeno napětí 5V. Poslední pin GND je připojen na zem. Pin pod rezistorem R3 a pin mezi driverem 1 a driverem 2 je přemostění.

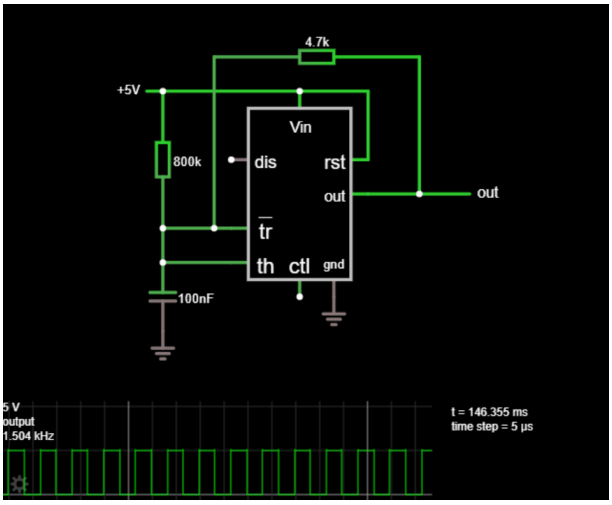
5.2 Stabilizátor

Pro napájení logiky driveru, arduina, oscilátoru a motoru vidlice byl použit 5V stabilizátor L78S05CV který je schopný dodávat 2A. Vstupní napětí pro tento stabilizátor je 20V. Kondenzátory C3 a C4 jsou zde na doporučení výrobce.

Na stabilizátor je připojeno Arduino, driver motoru vidlice, napájení logiky driverů, optické senzory a oscilátor. Arduino má proudový odběr 10mA. Driver motoru vidlice má proudový odběr 500mA. Drivery pro osy X a Z mají odběr řádově mikroampéry, počítejme tedy 1mA pro oba. Optické senzory mají maximální proudový odběr 22mA. NE555 má proudový odběr 6mA. Celkový proudový odběr všech komponent je 561mA. To vytváří výkon 2,8 W. Stabilizátor je dimenzovaný na 2A a osazený malýcm chladičem. Po přepočtu nám vychází vstupní proud stabilizátoru 140mA. Můžeme tedy tento stabilizátor bezpečně použít.

5.3 Oscilátor

Oscilátor generuje čtvercové impulsy simulující mikrokontrolér. Simulace tohoto oscilátoru je znázorněna na obrázku 8. Frekvence signálu je 1,504 kHz. Perioda signálu je 0,66 ms. Můžeme tedy vypočítat při šestnáctině kroku dobu, která udává dobu jedné otáčky a to 2,1 sekund.



Obrázek 8: Simulace oscilátoru s NE555

5.4 Konektory

Jako konektory byly použity Kolíkové lišty 2,54 mm jako samci a jako samice dutinkové lišty 2,54 mm. Vzhledem k charakteru konektorů jsou konektory barevně označeny, aby se zamezilo k opačnému připojení, což by mohlo mít za následek zničení součástek/motorů/desky.

- 4 pinový male konektor ArdUcc/DigIn 1- GND, 2- 5V Ucc, 3- En, 4- Dir. Tento konektor slouží pro připojení napájení k arduinu a pro přivedení dvou signálních vodičů pro motor vidlice, které mají stejný charakter ovládání jako motory ovládání os. Pin En slouží pro zapnutí motoru a Dir slouží pro orientaci jeho otáčení.
- 6 pinový male konektor DigOut 1-ArdEn 2-ArdDir 3-EnX 4-DirX 5-DirY 6-EnY .Konektor slouží pro připojení všech digitálních výstupu IPlogu. Pin EnX slouží pro zapnutí motoru na ose X (vodorovná). Pin EnZ slouží pro zapnutí motoru na Z ose (svislá). Pin EnV slouží pro zapnutí motoru, který ovládá vysunutí vidlice. Pin DirX, DirY, DirV ovládá směr otáčení motoru na ose X, Y nebo vidlice.
- 4 pinový male konektor Stp1 1- 2B, 2- 2A, 3- 1A, 4- 1B. Konektor slouží pro připojení motoru osy X. Piny 2B a 2A jsou připojeny na jednu z cívek

motoru, piny 1A a 1B jsou připojeny na druhou. Otočení tohoto konektoru vede k invertovnání směru otáčení.

- 4 pinový male konektor Stp2 1- 2B, 2- 2A, 3- 1A, 4- 1B. Konektor slouží pro připojení motoru osy Z. Piny 2B a 2A jsou připojeny na jednu z cívek motoru, piny 1A a 1B jsou připojeny na druhou. Otočení tohoto konektoru vede k invertovnání směru otáčení.

- 4 pinový female konektor IPlogUcc/AnIN 1- IPlogUcc, 2- IPlogGND, 3- OptDataZ, 4- OptDataX. Konektor je rozdělen do dvou částí. Levá část slouží pro připojení napájecího napětí 20V na pinu IPlogUcc k platformě IPlog nebo přídavného modulu. Pin IPlogGND slouží pro uzemnění této platformy nebo modulu. Vodič z tohoto pinu je rozvětven do všech modulů platformy IPlog. Druhá část konektoru (piny OptDataZ a OptDataX) slouží pro připojení signálu z optických senzorů os Z a X do modulu Iplogu na analogový vstup.

- 4 pinový female konektor UPlogUcc/MotUcc 1- IPlogUcc, 2- IPlogGND, 3- MotGNDV, 4- MotUccV. Konektor je rozdělen do dvou částí. Levá část slouží pro připojení napájecího napětí 20V na pinu IPlogUcc k platformě IPlog nebo přídavného modulu. Pin IPlogGND slouží pro uzemnění této platformy nebo modulu. Druhá část je vyhrazena pro motor vidlice.Pin MotGNDV slouží k uzemění řídící obvody a motoru pro ovládání vidlice. Pin MotUccV slouží pro přivedení napájecího napětí pro řídící obvody a motor vidlice. Barevné značení vodičů je opačné než je běžné (červený - zem, černý - 5V).

- 3 pinový female konektor OPt1 1- Sig, 2- GND, 3- Ucc. Konektor slouží pro připojení optického senzoru pro osu X. Pin Sig slouží pro výstupní signál senzoru. Pin GND slouží jako uzemnění. Pin Ucc je napájení senzoru 5V.

- 3 pinový female konektor OPt2 1- Sig, 2- GND, 3- Ucc. Konektor slouží pro připojení optického senzoru pro osu X. Pin Sig slouží pro výstupní signál senzoru. Pin GND slouží jako uzemnění. Pin Ucc je napájení senzoru 5V.

- 4 pinový female konektor ButIn1 1- X, 2- Limit1, 3- X, 4- Limit2. Konektor slouží pro připojení limitních tlačítek, které indikují, zada je vidlice maximálně vysunutá nebo maximálně zasunutá. Piny 1 a 3 nejsou využity, kvůli úpravě napětí tlačítek. Piny Limit1 a Limit2 slouží k přenosu signálu z jednotlivých tlačítek.

- 4 pinový female konektor ButIn2 1- X, 2- Ctr1, 3- X, 4- Ctr2. Konektor slouží pro připojení limitních tlačítek, které indikují, zada je vidlice maximálně vysunutá nebo maximálně zasunutá. Piny 1 a 3 nejsou využity, kvůli úpravě napětí tlačítek. Piny Ctr1 a Ctr2 slouží k přenosu signálu z řídících tlačítek ovládaných uživatelem.

- 4 pinový female konektor DigIn 1- Limit1, 2- Limit2, 3- Ctr1, 4- Ctr2. Konektor slouží k propojení signálů tlačítek s platformou IPlog. Piny Limit1 a Limit2 jsou pro limitní tlačítka vidlice. Piny Ctr1 a Ctr2 jsou pro tlačítka ovládané uživatelem.

6 Pohyb vidlice

Pro pohyb vidlice je použit krokový motor 28BYJ-48 s driverem. Bylo využito arduino pro simulaci stejného chování řízení jako mají motory pro osy X a Z. Arduino tedy přijímá dva signály, jeden udávající informaci o zapnutí/vypnutí motoru a druhý nesoucí informaci o orientaci otáčení. Více informací ohledně driveru 28BYJ-48 dostupné z [5].

6.1 Arduino

Pro mezistupeň byl využit klon vývojové platformy Arduino "NHduino UNO"(na tuto platformu bude stále odkazováno jako na Arduino). Tato platforma má 14 I/O pinů, z čeho může být použito 6 jako analogový vstupy a 6 je možné využít jako PWM modulace. Naše aplikace využije pouze 6 pinů a to2 jako digitální vstupy a 4 jako digitální výstup. Krystal má frekvenci 16MHz. Více informací ohledně Arduina dostupné z [7].

6.2 Popis programu

Samotný program pro arduino se skládá ze dvou komponent. Setupu, který se provede pouze jednou a loopu, který běží do nekonečna.

Na řádcích 1 až 6 je přiřazení označení pro jednotlivé piny, například pro vstup "in1"je pin 8. Integer rychlost udává zpoždění mezi jednotlivými kroky motoru v milisekundách.V části setup je samotné přiřazení pinů a nastavení, zda se jedná o vstupní nebo výstupní pin.

V části loop se nejdříve přečtou hodnoty na vstupu a uloží se do proměnných dirp a enp. V případě že je pin enable ve stavu logické nuly, program pokračuje na další If podmínku, která testuje, zda je dir 1 nebo 0 a podle toho běží program pro kroky sestupně nebo vzestupně.

Každá funkce krokX() obsahuje zápis na čtyři výstupní piny in1, in2, in3, in4 a funkci, která nám pozastaví program abychom se ujistili že funkce proběhla v pořádku. Ve funkcích kroky jsou postupně zvyšovány výstupy takovým způsobem, aby vytvořili nekonečně se posouvající vlnu. Každá změna ve výstupu je přímoúměrná jednomu kroku motoru.

6.3 Program arduina

```
const int in1 =  8;
const int in2 =  9;
const int in3 = 10;
const int in4 = 11;
const int en = 6;
const int dir = 7;
int rychlost = 5;
int enp;
int dirp;

void setup() {

pinMode(en,INPUT);
pinMode(dir,INPUT);
pinMode(in1,OUTPUT);
pinMode(in2,OUTPUT);
pinMode(in3,OUTPUT);
pinMode(in4,OUTPUT);
}

void loop() {
enp = digitalRead(en);
dirp = digitalRead(dir);

  if (!enp){
   if (dirp){
    krok1();
    krok2();
    krok3();
    krok4();
    krok5();
    krok6();
    krok7();
    krok8();
  }
  else {
   krok8();
   krok7();
   krok6();
   krok5();
   krok4();
   krok3();
   krok2();
   krok1();
  }
 }
}
void krok1(){
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  delay(rychlost);
}
void krok2(){
```

```
  digitalWrite(in1, HIGH);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  delay(rychlost);
}
void krok3(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  delay(rychlost);
}
void krok4(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  delay(rychlost);
}
void krok5(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  delay(rychlost);
}
void krok6(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, HIGH);
  delay(rychlost);
}
void krok7(){
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  delay(rychlost);
}
void krok8(){
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  delay(rychlost);
}
```

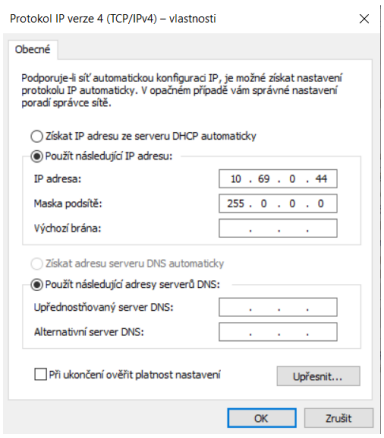
7 Základní nastavení METEL IPLOG

Jako hlavní řídící centrum je využita platforma IPLOG od firmy METEL. V tomto projektu byl využit produkt PLC IPLOG-G1-05-BI8.1-BOX a IO Modul s rozšířením AO8.1-05-BOX. Platformy musí být napájeny zvlášť napětím 10-60V DC.

7.1 Komunikace pomocí ethernet rozhraní

Pro komunikaci a programování mezi počítačem a IPLOG platformou je potřeba nastavit lokální adresu IPv4 zařízení tak, aby byla ze stejné podsítě.

Například můj modul má IP adresu 10.69.0.43. Adresa počítače může být například o jednu výš nebo níž. Použil jsem adresu 10.69.0.44 s maskou 255.0.0.0. Na obrázku 9 je zobrazeno moje nastavení IPv4 Protokolu které je využito pro komunikaci. Aby bylo možné komunikovat musí být počítač a IPLOG propojený za pomoci ethernet kabelu.



Obrázek 9: Nastavení IPv4 Protokolu

7.2 Nastavení sériového spojení

Pro správnou komunikaci modulu a platformy je potřeba nastavit sériové spojení. Nejprve je potřeba propojit moduly fyzicky vodiči. K tomuto slouží IF modul 07G, který má sériové spojení RS485. Fyzické propojení je vytvořeno třemi vodiči. Jeden vodič je z pinu A+ na pin A+, druhý vodič z pinu B- na B- a třetí vodič propojuje zem tohoto modulu.

7.3 Nastavení generování IO pro modul

Pro nastavení správného vytvoření io_variables.st které bude potřeba v dalších krocích je potřeba upravit iniciační program IPlogu. Aby tento krok fungo-

val, je potřeba aby byl správně nastaven IPv4 protokol a sériové spojení (viz předchozí sekce). Nejprve je třeba se přihlásit pomocí PUTTY. Ujistíme se že je zaškrtlé SSH spojení a vyplníme IP adresu IPLOGU (v mém případě 10.69.0.43). Poté nás IPlog zažádá o přihlašovací údaje. Pro studijní účely je zde uživatel "root"bez hesla.

Dalším krokem je zadání příkazu bez uvozovek: "vi /etc/init.metel/io-ext". Poté je třeba vstoupit do editačního režimu stisknutím klávesy "i". Řádky 42,44 a 46 je třeba odkomentovat, a to provedeme smazáním znaku #. Na řádku 46 je třeba přepsat text na "echo 1,ao8.1_if05". Tato změna funguje pro modul AO8.1-05, v případě použití jiného modulu je třeba upravit tento text pro správný modul.

```
#         START_COMMEXT1=1

#             echo ext1         > /sys/kernel/metel-io-dev/add_bus

#         echo 1,bi8.1_if05     > /sys/kernel/metel-io-dev/ext1/
    add_dev
```

Výše uvedený kód je kód před úpravou a níže je výsledný kód, jak by měl vypadat.

```
        START_COMMEXT1=1

        echo ext1         > /sys/kernel/metel-io-dev/add_bus

        echo 1,ao8.1_if05     > /sys/kernel/metel-io-dev/ext1/
    add_dev
```

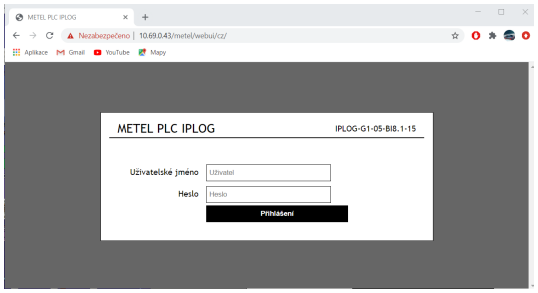
Pro dokončení této změny je třeba zmáčknout ESC, čímž se dostaneme z editačního režimu. Poté napsat znak ":"a napsat "wq". Příkaz odentrujeme a tím uložíme provedené změny.

7.4 Získání IO souboru

Pro používání vstupních a výstupních proměnných potřebujeme získat soubor, který obsahuje všechny vstupní a výstupní proměnné, jejich datový typ a jejich fyzickou adresu. Abychom mohli získat tento soubor, ujistěte se, že jsou správně provedené předchozí kroky.

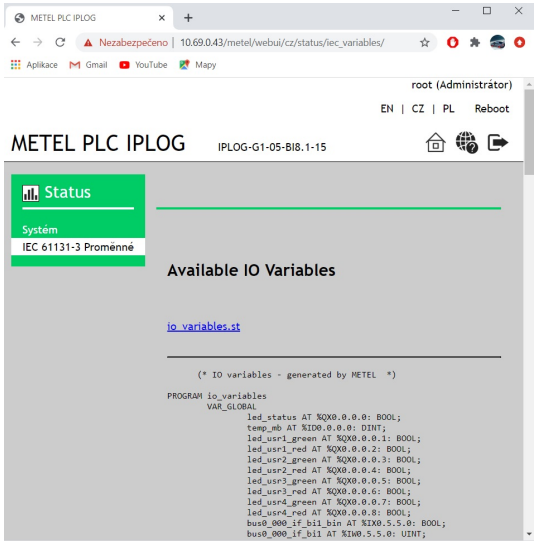
Pro získání tohoto souboru připojíme IPlog pomocí ethernet rozhraní. Do vyhledávače internetového prohlížeče zadáme IP adresu IPlogu, jak je zobrazeno na obrázku 10. IP adresu následuje "/metel/webui/cz/ takže výsledná adresa vypadá například takto: "10.69.0.43/metel/webui/cz/". Tento web je nejspíše nezabezpečený, proto je potřeba prohlížeči povolit pokračování na tuto stránku. Přihlásíme se do IPlogu. Pro studijní účely mi bylo poskytnuto přihlašovací jméno root bez hesla.

Po úspěšném přihlášení klikneme na zelené tlačítko "Status". V levém horním rohu klikneme na IEC 61131-3 Proměnné (číslo se může lišit). Zde nalezneme strukturovaný text, který obsahuje proměnné pro náš IPlog. Tyto proměnné budou potřeba v dalším kroku, proto si je pro zatím uložíme do textového souboru



Obrázek 10: Připojení k IPlogu a přihlášení pomocí prohlížeče

nebo jinou formou. Tyto data začínají řádkem "(* IO variables - generated by METEL *)"a končí řádkem "END_PROGRAM". Pokud proběhlo připojení a nakonfigurování modulu úspěšně, budou zde proměnné začínající "bus1_..."nebo "bus2_..."v případě více modulů. Proměnné začínající "bus0_..."jsou proměnné přímo na hlavní desce.



Obrázek 11: Ukázka IO strukturovaného textu v prohlížeči

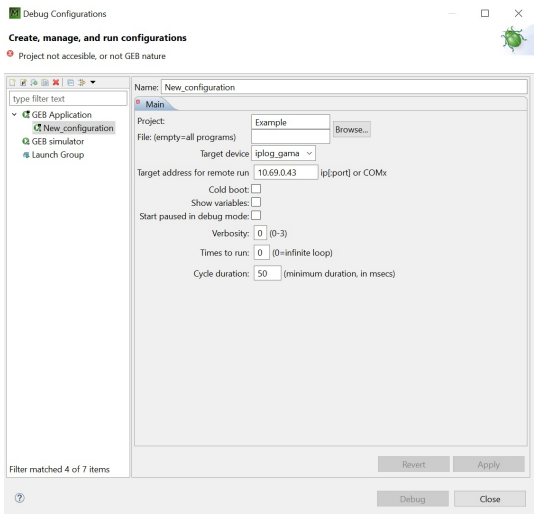
7.5 IDE pro programování

Pro samotné programování je použit software METEL IEC 61131-3 IDE. Toto prostředí je FBD (function block diagram) nebo jako strukturovaný text. Tento styl programování využívá bloků, které se propojují. Je možné vytvořit vlastní bloky a ty naprogramovat pomocí strukturovaného textu a jazyku připomínajícího

pascal. Tento software se dá stáhnout z [1] . Pro další kroky je nutné stáhnout a nainstalovat tento software.

7.6 Vytvoření debugovacího konfigurátoru

Abychom mohli program nahrát do IPlogu musíme pro něj vytvořit konfiguraci. Kliknutím na šipku u "Debug configuration"vybereme "Debug configurations...". Zobrazí se nám dialogové okno. Kliknutím pravým tlačítkem na "GEB Application"vytvoříme novou konfiguraci pomocí tlačítka "New Configuration". V této konfiguraci nastavíme jméno programu, který chceme nahrát do IPlogu (v našem případe "Example"). V položce target device vybereme "iplog_gama". Do položky Target address for remote run vyplníme IP adresu IPlogu, která je napsána na krabičce (v našem případě 10.69.0.43). Pokud chceme, aby tento program běžel v nekonečném cyklu, zvolíme v Times to run 0.



Obrázek 12: Vytvoření debugovacího konfigurátoru

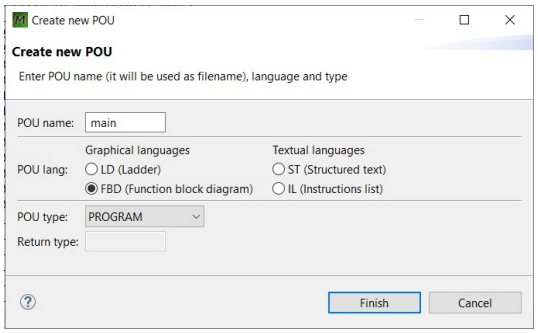
8 Programování IPlogu

Program vytvoříme kliknutím na "File» "New» "GEB 61131-3 project". Tento projekt si pojmenujeme a kliknete na tlačítko finish.

8.1 Nastavení základu programu

Nejprve si musíme vytvořit hlavní programovací prostředí. Vlevo v "Project Explorer"je složka pojmenovaná podle našeho programu. Na tuto složku kliknete pravým tlačítkem a vybereme "Create new POU". Zobrazí se nám dialogové okno. Doporučuji pojmenovat hlavní program "main". Nastavení "POU

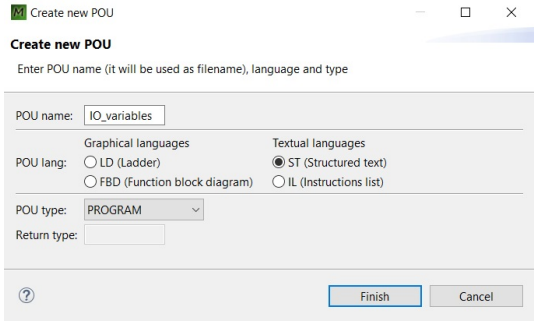
lang"udává jazyk, v jakém je toto POU tvořeno. Pro naší aplikaci si vystačíme s FBD a nebo ST. Pro hlavní program zvolíme FBD. Je důležité vybrat v POU type "PROGRAM". Toto nastavení udává, že se jedná o základní program. Dokončíme kliknutím na tlačítko Finish.



Obrázek 13: Vytvoření hlavního programu

8.2 Vytvoření IO souboru

Pro vytvoření IO souboru klikneme pravým tlačítkem na složku s programem a vybereme "Create new POU". Toto POU si pojmenujeme "IO_variables"a jako lang zvolíme Strukturovaný text (ST). POU type zvolíme program. Po vytvoření si tento soubor otevřeme a překopírujeme sem IO získané z sekce 7.4. Soubor uložíme pomocí zkratky ctrl+S nebo disketou v levém horním rohu.



Obrázek 14: Vytvoření IO strukturovaného textu

8.3 Vytvoření vlastního bloku

Vytvoření vlastního bloku je velice podobné jako u vytvoření hlavního programu. Pravým tlačítkem klikneme na složku s naším programem a vybereme "Create

new POU". Zde napíšeme jméno, jaké chceme aby měl náš blok. Jako strukturu vybereme ST nebo FBD. Důležité je změnit POU type na Function_Block. Nastavení vstupu a výstupu bloku je popsáno v kapitole 8.5.

8.4 Proměnné pro Strukturovaný text

Co se týče proměnných pro Strukturovaný text, Máme zde před generovaný program, který má předpřipravenou strukturu pro náš blok. Jsou zde tři sekce, které nám udávají, o jaký druh proměnné se jedná. Local vars jsou proměnné které existují pouze v tomto bloku, input vars jsou proměnné, do kterých zapisujeme z vyšších struktur. Output jsou proměnné, které nastavujeme v běhu programu a předávají se na výstup. Každá proměnná je zapsána ve tvaru "jmeno : INT;"kde jmeno je název proměnné a INT je datový typ, který se může měnit.

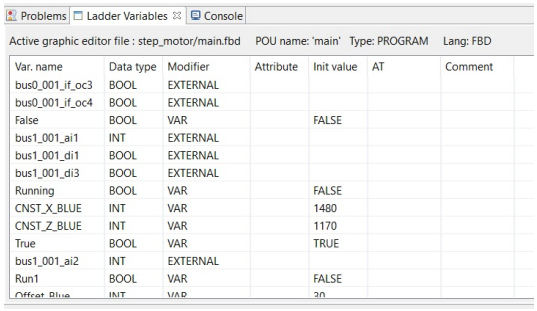
8.5 Proměnné pro FBD

Tato sekce se zabývá popisem práce s proměnnými. Tento postup funguje úplně stejně pro hlavní program i jednotlivé bloky. Pro vytvoření nové proměnné klikneme na tlačítko Add variable. Zde napíšeme jméno, datový typ (vše velkými písmeny). Initial value je počáteční hodnota, jakou proměnná má na začátku programu. Variable type udává, o jakou proměnou se jedná. Máme na výběr:

- VAR - vnitřní proměnná, která existuje pouze uvnitř tohoto bloku/programu.
- EXTERNAL - tato proměnná se používá pro řízení vstupu nebo výstupu. Aby ale tato proměnná správně fungovala, je potřeba aby měla stejné jméno jako má označení v IO souboru. Například výstup bipolárního tranzistoru, který je v IO souboru označený "bus0_000_if_bi1 AT %IX0.5.5.0: BOOL;"musí mít jméno "bus0_000_if_bi1"a datový typ BOOL. Nedoporučuji mít více výstupů v programu, mohou se mezi sebou prát a program nemusí fungovat. Příklad proměnných využitých v mém programu je zobrazen na obrázku 15.
- INPUT - tato proměnná se používá ve vlastních blocích. Slouží jako vstupní proměnná do bloku.
- OUTPUT - tato proměnná se chová jako INPUT, až na rozdíl že slouží jako výstupní proměnná v bloku. Tuto proměnnou je ale nadále používat jako Variable Ref.

8.6 Bloky

Pro přidání nového bloku slouží tlačítko v pravé nabídce ve složce Elements. Kliknutím na Blok se nám zobrazí nabídka, kde si můžeme vybrat funkční blok. Naše bloky jsou ve složce dole, která má název jako náš program. Pokud se blok nezobrazuje, je potřeba ho zkompilovat kliknutím pravým tlačítkem na blok v levé nabídce a vybrat Build Project. Dokumentace k blokům je dostupná z [2].



Obrázek 15: Příklad proměnných

8.7 Použití vstupních a výstupních proměnných

Pro práci s EXTERNAL proměnnýma, jako jsou tranzistory a analogové vstupy využíváme v nabídce elements "Variable Ref"pro výstupní proměnné a "Expression"pro vstupní proměnné.

9 Popis programu

Tato sekce popisuje program, který ovládá pohyb zakladače.

9.1 Použité proměnné

V tabulce níže je popis všech proměnných použitých v programu. Proměnné použité v jednotlivých blocích jsou popsány v jednotlivých sekcích pro určité bloky.

Jméno proměnné	Datový typ	Typ proměnné	Popis
bus0_001_oc1	BOOL	EXTERNAL	Dir osy X
bus0_001_oc2	BOOL	EXTERNAL	Dir osy Z
bus0_001_if_oc1	BOOL	EXTERNAL	En osy X
bus0_001_if_oc2	BOOL	EXTERNAL	En osy V
bus0_001_if_oc3	BOOL	EXTERNAL	En osy Z
bus0_001_if_oc4	BOOL	EXTERNAL	En osy X
False	BOOL	VAR	Konstanta udávající FALSE
bus1_001_ai1	INT	EXTERNAL	Analog in osy X
bus1_001_di1	BOOL	EXTERNAL	Zelené tlačítko
bus1_001_di3	BOOL	EXTERNAL	Žluté tlačítko
Running	BOOL	VAR	Značení běhu sekvence
CNST_X_BLUE	INT	VAR	Konstanta pozice X modré
CNST_Z_BLUE	INT	VAR	Konstanta pozice Z modré
CNST_UP	INT	VAR	Konstanta zvednutí
True	BOOL	VAR	Konstanta udávající TRUE
bus1_001_ai2	INT	EXTERNAL	Analog in osy Z
Run1	BOOL	VAR	Značení běhu 1. sekvence
Offset_Blue	INT	VAR	Konstanta offsetu pro modrou
Run2	BOOL	VAR	Značení běhu 2. sekvence
Run0	BOOL	VAR	Značení běhu 0. sekvence
bus1_001_di4	BOOL	EXTERNAL	Indikace zasunuté vidlice
Dir_oc2	BOOL	VAR	Pomocná proměnná pro OC2
bus1_001_di2	BOOL	EXTERNAL	Indikace vysunuté vidlice
Yellow_pressed	BOOL	VAR	Značení zmáčknutí žluté
Green_pressed	BOOL	VAR	Značení zmáčknutí zelené
CNST_X_YELLOW	INT	VAR	Konstanta pozice X žluté
CNST_Z_YELLOW_UP	INT	VAR	Konstanta pozice Z nad žlutou
Offset_Yellow	INT	VAR	Konstanta offsetu pro žlutou
Move1_En	BOOL	VAR	Pomocná proměnná pro 1.pohyb
Move1_Dir	BOOL	VAR	Pomocná proměnná pro 1.pohyb
CNST_X_GREEN	INT	VAR	Konstanta pozice X zelené
CNST_Z_GREEN_UP	INT	VAR	Konstanta pozice Z nad zelenou
Run3	BOOL	VAR	Značení běhu 3. sekvence
Move_place_En	BOOL	VAR	Pomocná proměnná pro pohyb
Move_place_Dir	BOOL	VAR	Pomocná proměnná pro pohyb
CNST_Z_GREEN	INT	VAR	Konstanta pozice Z zelené
CNST_Z_YELLOW	INT	VAR	Konstanta pozice Z žluté
Done	BOOL	VAR	Značení ukončení sekvence

9.2 Použité základní bloky

Tato sekce popisuje základní bloky použité v programu. Detailnější informace o blocích nebo informace o jiných blocích dostupné z [2].

- LE INT - Tento blok nastaví na výstup logickou jedničku v případě, že In1 je menší nebo roven In2.
- ADD INT - Tento blok sčítá proměnné typu INT a jejich výslednou hodnotu nastavý na výstup.
- SUB INT - Tento blok odčítá proměnné In2 od In1. Výsledek je zapsán na výstup bloku.
- OR BOOL - Tento blok slouží jako logická brána OR.
- AND BOOL - Tento blok slouží jako logická brána AND.
- NOT BOOL - Tento blok slouží jako logická inverze.

9.3 Blok BOOL Double

Tento blok je pomocný. Jeho hlavní účel je rozdvojit vstup typu BOOL na dva. Díky tomuto bloku není potřeba vytvářet přebytečné proměnné.

```
FUNCTION_BLOCK BOOL_Double
  VAR
    (* local vars *)
  END_VAR
  VAR_INPUT
    (* input vars *)
    in : BOOL;
  END_VAR
  VAR_OUTPUT
    (* output vars *)
    out1 : BOOL;
    out2 : BOOL;
  END_VAR
   (* Pou body: FUNCTION_BLOCK statements *)
   out1 := in;
   out2 := in;
END_FUNCTION_BLOCK
```

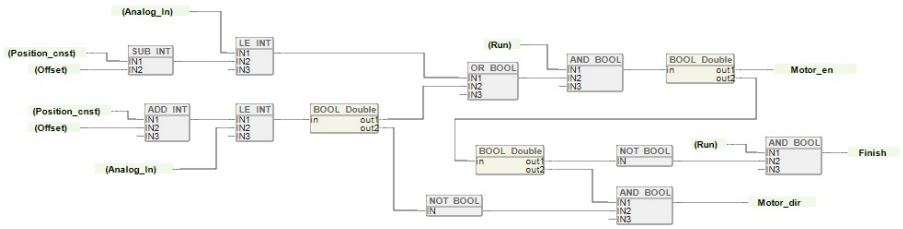
9.4 Blok Move

Tento blok je hlavním blokem pro pohyb motorů na ose X nebo Z. Skládá se ze 4 vstupních proměnných a tří výstupních. Cílem tohoto bloku je porovnat vstup ze senzoru pro X nebo Z osu s konstantou, která udává na jaké poloze se nachází platforma. Na základě tohoto porovnání poté blok rozhodne, zda zapne motor a jakým směrem se bude motor otáčet.

K porovnání dochází v bloku LE INT, který vrací logickou jedničku v případě, že In1 je menší nebo rovno In2. Do jednoho vstupu je přiveden analogový senzor,

do druhého konstanta, která je modifikována o offset, který zajišťuje dostatečný prostor pro vypnutí motorů. Toto porovnání je zde dvakrát, z důvodu určení, zda je zakladač příliš vlevo nebo vpravo. Ze spodní větve je vyveden signál, který nese informaci o směru, jakým se má motor otáčet.

Výstup z porovnávacích bloků je přes sadu logických bloků, které zajišťují, aby byl výstup sepnut pouze v případě, že jedna z porovnávacích větví je mimo určenou konstantu a zároveň je signál Run v hodnotě logické jedničky. Tento Run signál slouží pro zapínání celého bloku. Výstupní signál je rozvětven do zapínání motoru, spínání směrového signálu přes blok AND, který je vyveden z porovnávací větve a do větve, která zpracovává dokončení pohybu. Tato větev má za úkol nastavit na výstupní proměnnou Finish na logickou jedničku v případě, že Run je stále v logické ledničce a není zapnutý motor. Tato výstupní proměnná je důležitá pro správné řetězení bloků.

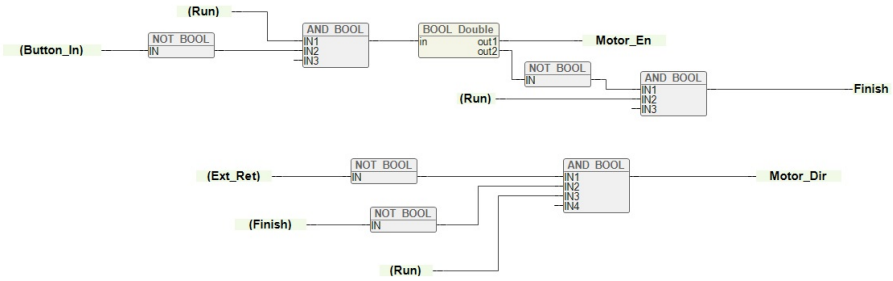


Obrázek 16: Realizace bloku Move

9.5 Blok MoveV

Tento blok slouží pro pohyb vidlice. Skládá se ze 3 vstupních a 3 výstupních proměnných. Hlavním cílem tohoto bloku je v případě zapnutí a udání informace o požadovaném stavu vidlice docílit tohoto stavu. Toho je docíleno zapnutím motoru do doby, než je sepnuto příslušné limitní tlačítko.

V případě, že není stisknuté tlačítko je potřeba na výstupní proměnnou motoru přivést logickou jedničku. Toho je docíleno pomocí bloku NOT, který invertuje vstup tlačítka. Blok AND, do kterého je přivedena proměnná Run slouží jako spínání, aby tento blok byl zapnutý jen pokud potřebujeme. Směr otáčení tohoto motoru je přiveden externě. Motor se vždy pohybuje až do svého limitu takže není třeba měnit dynamicky směr otáčení uvnitř bloku. Je ale hlídáno, aby byl směr motoru sepnutý pouze v případě, že je tento blok aktivní a ještě není dokončený pohyb. Informaci o dokončení pohybu se stará logika u výstupu Zapnutí motoru. Výstupní proměnná Finish je v logické jedničce, pokud blok dokončil svou činnost a je aktivovaný proměnnou Run.



Obrázek 17: Realizace bloku MoveV

9.6 Blok Set true

Tento blok slouží jako paměťová buňka. Při přivedení krátkého signálu na vstup nastaví blok výstupní signál na logickou jedničku, dokud nedojde k jeho resetování. Blok má 2 vstupní signály a 4 výstupní. Výstupní signál out je zdvojený na proměnnou out2 a invertovaný na proměnnou outINV. Tělo bloku je tvořeno jednou podmínkou, která řídí funkčnost celého bloku.

```
FUNCTION_BLOCK set_true
  VAR
    (* local vars *)
  END_VAR
  VAR_INPUT
    (* input vars *)
    in  : BOOL;
    rst : BOOL;
  END_VAR
  VAR_OUTPUT
    (* output vars *)
    out    : BOOL;
    out2   : BOOL;
    outINV : BOOL;
  END_VAR
  (* Pou body: FUNCTION_BLOCK statements *)
  IF in = TRUE THEN
    out    := TRUE;
    out2   := TRUE;
    outINV := FALSE;
  ELSIF rst = TRUE THEN
    out    := FALSE;
    out2   := FALSE;
    outINV := TRUE;
  END_IF;
END_FUNCTION_BLOCK
```

9.7 Blok Select

Tento Blok slouží pro nastavení konstanty pro pohyb, v případě volby. Pokud není žluté tlačítko zmáčknuté, blok pošle na výstup zelenou referenci. Pokud

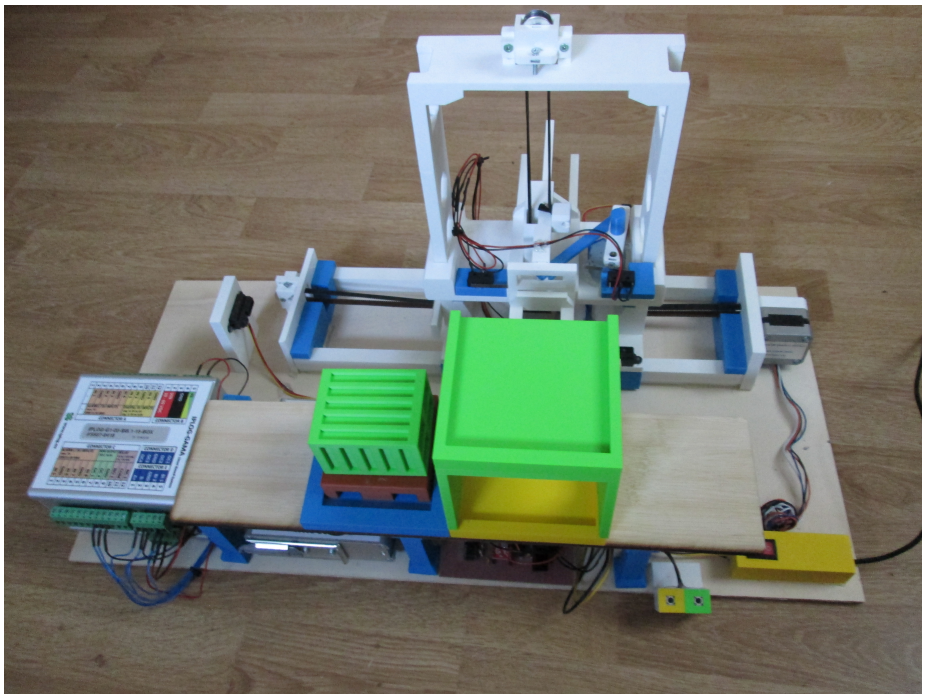
je stisknuté, pošle blok na výstup žlutou referenci. Toto chování je docíleno jednoduchou podmínkou.

```
FUNCTION_BLOCK Select
  VAR
    (* local vars *)
  END_VAR
  VAR_INPUT
    (* input vars *)
    in_ref_y : INT;
    in_ref_g : INT;
    select_y : BOOL;
  END_VAR
  VAR_OUTPUT
    (* output vars *)
    out_ref : INT;
  END_VAR

   (* Pou body: FUNCTION_BLOCK statements *)
   IF select_y = TRUE THEN
    out_ref := in_ref_y;
  ELSE
    out_ref := in_ref_g;
  END_IF;
END_FUNCTION_BLOCK
```

9.8 Hlavní program

Hlavní program zajišťuje, aby byli jednotlivé bloky aktivovány ve správném pořadí a měli všechny potřebné proměnné. Aby bylo zamezeno přepisování výstupních proměnných, jsou všechny bloky se stejnou výstupní proměnnou spojeny přes OR gate. Je tedy potřeba, aby neaktivní blok měl výstup v logické 0. Pro správné řetězení bloků jsou použity proměnné Run, nebo přímé propojení. Každý blok po dokončení operace nastaví výstup Finish na logickou 1, která se uschová v paměťové buňce "set true". Tento blok nastaví logickou jedničku na vstup "Run"dalšího bloku. Další funkcí je uchování informace o zmáčknutém tlačítku, zajištěné blokem "set true", který se resetuje po dokončení celé sekvence. Realizace tohoto programu je v příloze.



Obrázek 18: Finální konstrukce zakladače

10 Závěr

Zakladač byl navržen, vymodelován, vytisknut. Byla navržena deska plošných spojů, která byla vyleptaná a osazena. K řízení zakladače je použit IPLOG od METELU s rozšiřujícím modulem. Modul je připojen pomocí Bylo využito 6 digitálních výstupů, 4 digitální vstupy, 2 analogové vstupy Jako pomocný modul k driveru motoru vidlice bylo využito Arduino, na kterém běží program kopírující chování ostatních motorů. Pro zjišťování polohy zakladače byly využity optické senzory SHARP, které analogovně měří vzdálenost od 4cm do 30cm. Pro pohyb byly využity krokové motory SX17 společně s drivery A4988. Finální model zakladače je zobrazen na obrázku 18. Funkčnost zakladače je demonstována na videu dosupného z [9]. Podrobnější dokumentace fotografií je dostupná v příloze.

Reference

[1] *Dokumentace k výrobku METEL IPLOG*
https://www.metel.eu/cz/newdesign/products?itemId=225

[2] *Dokumentace ke knihovnám*
http://www.gebautomation.com/help/

[3] *Datasheet k driverům motorů*
https://www.pololu.com/product/1182

[4] *Datasheet k senzoru vzdálenosti*
https://arduino-shop.cz/docs/produkty/0/917/gp2y0a41sk0f.pdf

[5] *Datasheet k driveru motoru 28BYJ-48*
https://www.hwkitchen.cz/user/related_files/driver-pro-krokovy-motor-uln2003a.pdf

[6] *Datasheet ke krokovým motorům*
http://jan.kostial.sk/data/uploads/docs/cnc/microcon.cz_krokove_motory_sx_13-20.pdf

[7] *Datasheet k arduinu*
https://store.arduino.cc/arduino-uno-rev3

[8] *3D tiskárna Prusa i3 Mk3s*
https://www.prusa3d.cz/original-prusa-i3-mk3/

[9] *Demonstrace modelu*
https://youtu.be/lK4TrVKs4GQ

Přílohy

Příloha 1

Příloha 1 obsahuje hlavní program.

Příloha 2

Příloha 2 obsahuje 12 obrázků realizace zakladače

- Celá konstrukce
- Hlava s paletou
- Volnoběžný držák osy Z
- Motor osy Z, Driver motoru vidlice
- Motor osy X
- Volnoběžný držák osy X
- Senzor osy Z
- Senzor osy X
- Osazená nezapojená deska plošných spojů
- Zapojená deska plošných spojů
- Modul IPLOGU
- Řídicí tlačítka, hlavní vypínač, Arduino