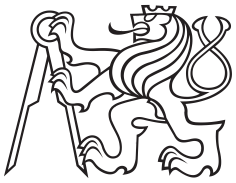
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídicí techniky

Využití robota LEGO Mindstorms EV3 - návrh robota třídícího technické díly ze stavebnic LEGO Mindstorms NXT nebo EV3 pro propagaci FEL

Vojtěch Dědek

Vedoucí: Ing. Martin Hlinovský, Ph.D.
Obor: Robotika a Kybernetika
Duben 2024

Poděkování

Děkuji ČVUT, že mi je tak dobrou *alma mater*.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 10. dubna 2024

Abstrakt

Moje práce se zabývá třízením technických dílů LEGO, cílem je vytvořit třídící linku, která bude schopná třídit díly podle tvaru a barvy. K tomu budu nejprve potřebovat separovat jednotlivé díly od sebe. K identifikaci dílů využiji robotické vidění, které bude obstarávat kamera. Závěrečné rozdělení do přihrádek bude obstarávat robotický manipulátor.

V práci využiji algoritmus z oboru strojového učení jménem k nejbližších sousedů, známý také jako Knn. Pro tento algoritmus vytvořím databázi referenčních obrázků, s pomocí kterých pak provedu samostatnou identifikaci. Algoritmus bude pracovat se všemi barvami spektra najednou aby mohl rozlišovat díly jak podle barvy tak podle tvary. Identifikované díly bude rozřazovat do jednotlivých přihrádek robotický 2D manipulátor. U manipulátoru se dostaneme do oblasti robotiky ve které projdeme přímou kinematickou úlohu, limity pracovního prostoru, inverzní kinematickou úlohu a nakonec plánování trasy pomocí interpolace. Poslední část práce se dostaneme do oblasti hardwaru, kde se podíváme na použité elektronické součástky a jejich důležité vlastnosti.

Klíčová slova: LEGO, identifikace, robotika

Vedoucí: Ing. Martin Hlinovský, Ph.D. katedra řídicí techniky FEL

Abstract

My work talks about sorting of LEGO technical parts, the goal is to create a sorting machine that will be able to sort parts according to shape and color. To achieve this, I will first need to separate the individual parts from each other. To identify the parts, I will use robotic vision, which will be provided by a camera. The final placement into compartments will be handled by a robotic manipulator.

I will use an algorithm from the field of machine learning called k-nearest neighbors, also known as Knn. For this algorithm to be able to function properly, I will create a database of reference images, which i will use to identify LEGO parts. The algorithm will work with all three colors of the spectrum at the same time to be able to distinguish parts both by color and shape. The identified parts will be sorted into individual compartments by a robotic 2D manipulator. For the manipulator, we will get robotics in which we will go through the direct kinematic task, the limits of the working space, the inverse kinematic task and finally route planning using interpolation. Lastly, we will get to the used hardware, where we will look at the used electronic components and their important properties.

Keywords: LEGO, identification, robotics

Title translation: Usage of the LEGO Mindstorms EV3 - Design of a Robot Sorting Technical Parts from LEGO Mindstorms NXT or EV3 kits for Promotion of the Faculty

Obsah

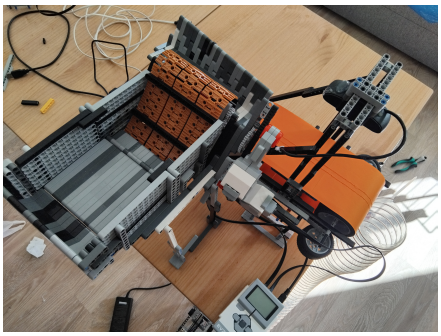
Obrázky

Tabulky

Kapitola 1

Úvod do problematiky

V této práci se budu věnovat mojí práci s cílem postavit linku na třízení technických dílů LEGO. Technické díly LEGO, které se pokusím třídit se dají odlišit tvarem či barvou a ne vždy platí že dva díly stejného tvaru jsou funkčně ekvivalentní. Avšak neplatí ani obrácené tvrzení, protože existují díly se stejným tvarem a funkcí, mají odlišnou barvu, ale my by jsme si je přáli považovat za stejné. Budeme proto muset třídit jak na základě barvy tak na základě tvaru.



Obrázek 1.1: Třídící linka

1.1 Přístup k řešení

Při stavbě stroje na třízení dílů jsem stroj rozdělil na tři hlavní části z nichž každá řeší jeden problém, tyto části jsou oddělovač jednotlivých dílů,

rozpoznávač jednotlivých dílů a stroj který uloží díly do správných přihrádek.

1.2 Oddělovač dílů

Tato část stroje na obrázku 6.2 má dvě hlavní funkce, proto se skládá ze dvou částí, první je zásobník na neroztřízené díly který slouží jako vstup do celého stroje. Zasobník podává díly po několika málo kusech do další části. Druhou částí je samotný oddělovač, skládá se z vibrující platformy která svými vibracemi odděluje díly od sebe a podává je po jednom další části.

1.3 Rozpoznávač dílů

Tato část stroje na obrázku 6.3 slouží k rozpoznávání jednotlivých dílů, skládá se z kamery a pásu po kterém jsou díly posunuty před kameru, které nejprve rozpozná že nový díl přijel po páse, poté ho klasifikuje a nakonec ho pošle dále k roztřízení.

1.4 Ukládač dílů

Poslední část stroje 3.1 má za úkol uložit identifikované díly do jednotlivých přihrádek podle předloženého klíče. Některé díly patří do stejných přihrádek, například pokud třídíme podle funkce dílů, zde jsou příkladem díly stejného tvaru s odlišnou barvou. Třídič funguje na principu dvou dimenzionálního manipulátoru který se pohybuje nad jednotlivými přihrádkami.

Kapitola 2

Rozpoznávání technických dílů LEGO

2.1 Úvod

V této kapitole se budu věnovat postupu použitém při rozpoznávání různých technických dílů LEGO. Dále se podíváme na samotný algoritmus použitý při detekci dílů a jejich následnou klasifikaci. Použil jsem upravenou verzi algoritmu knn[Fix51].

2.2 Detekce dílů

Aby jsme mohli rozpoznávat technické díly musíme je nejdříve detekovat. Poté co je detekujeme tak teprve můžeme začít rozpoznávání. Potřebujeme tedy aby každý díl byl vždy detekován ve stejný okamžik , aby při porovnání s databází dílu byly snímky konzistentní a rozpoznání bylo korektní. Ovšem různé díly mohou být detekovány v různé okamžiky, pokud se zachová předpoklad že stejný konkrétní díl bude vždy detekován na stejném místě.

2.2.1 Pás

Obrázek 2.1: Pás

Pás po kterém se díly pohybují má oranžovou barvu, protože po testování různých možných barev pásu se všechny barvy dílů dali rozlišit od barvy pásu. To je důvod proč barva pásu nemůže být libovolná, protože bílé díly by nebyli vidět na bílém pásu, šedivé na šedivém atd. Pás byl protkán oranžovou nití ve snaze zabránit kulatým dílům se odvalovat ze středu pásu kam byly umístěny kvůli detekci, až mimo pás na zem.

2.2.2 Algoritmus detekce

Kvůli šumu z kamery je potřeba udělat průměr z více hodnot. Je také potřeba dát pozor na proměnlivé množství světla v místnosti. Kamera kterou používám viz. má automatickou korekci úrovně světla. Algoritmus spočítá celkovou úroveň všech tří složek v referenční a detekční oblasti a poté spočítá jejich eukleidovskou vzdálenost. Výsledek pak porovná s detekčním prahem, který je nastaven experimentálně. Algoritmus se dá shrnout následující nerovnicí:

$$\sum_{c=0}^{2}\left[\left(\sum_{i=r_{row1}}^{r_{row2}}\sum_{j=r_{col1}}^{r_{col2}} x_{c,i,j}\right)^2-\left(\sum_{i=d_{row1}}^{d_{row2}}\sum_{j=d_{col1}}^{d_{col2}} x_{c,i,j}\right)^2\right]^2 > k_{detect} \qquad (2.1)$$

, kde r_{row1}, r_{row2} jsou řádky hranice referenčního obdélníku, r_{col1}, r_{col2} jsou sloupce hranice referenčního obdélníku, d_{row1}, d_{row2} jsou řádky hranice detekčního obdélníku, r_{col1}, r_{col2} jsou sloupce hranice detekčního obdélníku a $x_{c,i,j}$ hodnota barvy pixelu se souřadnicemi i,j a barvou c. Rozdíly barev se sčítají zvlášť ve druhé mocnině, aby nedošlo k odečtení rozdílů s opačným znaménkem a tím pádem nedošlo k detekci dílu.

2.3 K nejbližších sousedů

K nejbližších sousedů anglicky K-nearest neighbors je metoda pro porovnávání a klasifikaci. Algoritmus patří do oblasti umělé inteligence, konkrétně

strojového učení. Pro svoji správnou funkci potřebuje algoritmus připravená referenční data se kterými bude vstup porovnávat. Kvalita a množství těchto dat ovlivní schopnost tohoto algoritmu správně klasifikovat vstup. Tento algoritmus vyžaduje pro svoji funkčnost dva algoritmy a to algoritmus, který přidělí každé porovnávané položce skóre a druhý algoritmus, jenž s pomocí skóre klasifikuje položku.

2.3.1 Hodnocení obrázků

Pro přidělení skóre jednotlivým obrázkům jsem zvolil jednoduchý algoritmus, který spočítá eukleidovskou vzdálenost mezi referencí a obrázkem který je potřeba klasifikovat. Algoritmus se dá shrnout následující rovnicí:

$$s = \sum_{c=0}^{2} \sum_{i=0}^{479} \sum_{j=0}^{679} (x_{c,i,j} - r_{c,i,j})^2 \tag{2.2}$$

, kde $x_{c,i,j}$ hodnota barvy pixelu se souřadnicemi i,j a barvou c, $r_{c,i,j}$ hodnota barvy pixelu referenčního obrázku se souřadnicemi i,j a barvou c a s je skóre referenčního obrázku. Je zde vidět že všechny rozdíly se neodečtou vzájemně a tedy se projeví ve výsledném skóre.

Přetečení

Přetečení nebo anglicky overflow je událost která může nastat v počítačích pokud se do proměnné pokusíme uložit moc velké nebo malé číslo. Pokud tato událost nastane projeví se neočekávanou změnou hodnoty proměnné, například opakovaným přičítáním kladných čísel se může po přetečení hodnota proměnné změnit v zápornou, je tedy potřeba zamezit těmto případům.

Máme více možností jak omezit tyto události, v některých případech stačí vyměnit pořadí operací ve výpočtu, jindy musíme využít větší proměnné. Pro proměnné typu celé číslo anglicky intiger víme, že první bit je použit pro znaménko čísla a zbytek pro samotnou hodnotu. Nejvyšší číslo které lze do proměnné uložit je:

$$x_{max} = 2^{b-1} - 1 \tag{2.3}$$

, kde x_{max} je nejvyšší číslo a b je počet bitů. Pokud tedy nejvyšší možná hodnota našeho hodnotícího algoritmu bude menší než x_{max} k přetečení nedojde. Pro porovnání velkých čísel se dají použít logaritmy. Aby jsem

zjistili kolik bitů potřebujeme pro číslo vyžijeme logaritmu o základu 2 na rovnici 2.3 a po úpravě získáme:

$$n_{bit} = \log_2(x+1) + 1 \tag{2.4}$$

kde n_{bit} je potřebný počet bitů a x je maximální očekávaná hodnota proměnné. Dosadíme-li naše hodnoty do 2.2 získáme:

$$s_{max} = 3 \cdot 480 \cdot 640 \cdot 256^2 \tag{2.5}$$

Výsledek dosadíme do 2.4 a získáme:

$$n_{bit} = 36,81 \tag{2.6}$$

Musíme vybrat proměnnou s větším počtem bitů než vypočítaná hodnota tedy zvolíme 64-bitový intiger.

2.3.2 Klasifikace

Poté co jsme získali skóre obrázku proti všem referencím je potřeba klasifikovat obrázek. K v algoritmu K-nejbližších sousedů je proměnná a odkazuje se na kolik nejbližších sousedů se bude algoritmus dívat při svém rozhodování.

Nejbližší soused

Pro $k = 1$ se algoritmus rozhodování zjednoduší na výběr nejlépe odpovídajícího souseda, zde nebude potřeba složitá implementace rozhodujících pravidel pro případy které mohou nastat při $k > 1$.

Výběr parametru k

Výběr parametru k záleží na aplikaci a velikosti trénovacích dat. Pro různé aplikace je vhodné různé k. Velikost trénovacích dat nám shora omezuje parametr k, protože nechceme aby k bylo větší než počet referencí ke každé možné klasifikaci. Pro vyšší k je potřeba vyřešit co se stane pokud v k nejbližších sousedech jsou různé možné klasifikace.

Příklad možného algoritmu pro k=3 je nejlepší soused, tedy ten s nejnižším skóre, dostane 7 bodů. Druhý dostane 5 bodů a třetí dostane 3 body. Poté

vybereme možnost s nejvíce body a prohlásíme ji za správnou. V tomto případě pokud druhý a třetí budou stejné tak se prohlásí za správnou klasifikaci. Pro tři různé vybereme nejbližší atd.

Třízení

Může se nám stát že si budeme přát aby různé díly skončily ve stejné přihrádce toho můžeme docílit více způsoby, ale nejednodušší z těchto způsobů je klasifikovat různé díly jako stejný díl a tímto způsobem algoritmus nepozná rozdíl. Příkladem jsou například stejné díly odlišné barvy.

Důležitější je ale na tento mechanismus dát pozor v případě dílů které jsou nesymetrické tyto díly mohou být položeny na pás v různých orientacích a my rozhodně chceme, aby tyto různé obrázky byly klasifikovány stejně a v tomto případě budeme potřebovat dostatek referencí pro každou možnou polohu kterou může díl na páse zaujmout.

Kapitola 3

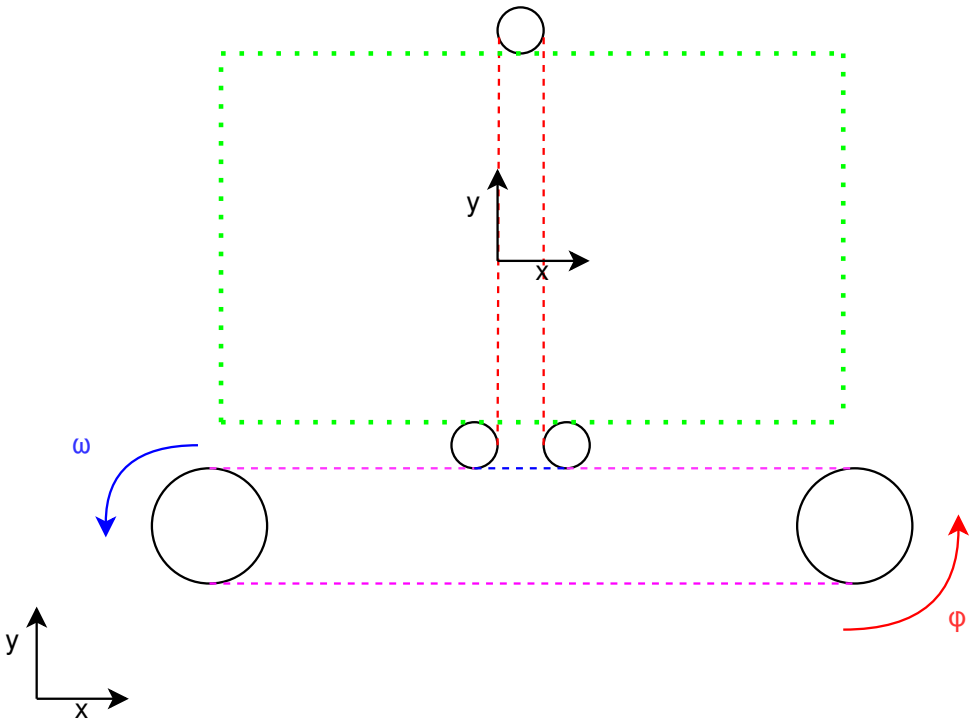
Manipulátor

3.1 úvod

V této kapitole se budeme věnovat konstrukci výstupu z třídícího zařízení. Popíšeme si teorii a ovládání manipulátoru a jeho souřadnice. Z tohoto pak získáme takzvanou inverzní kinematickou úlohu, kterou vyřešíme.

3.2 Přímá kinematická úloha

Přímá kinematická úloha se v robotice zabývá polohou a orientací konce manipulátoru vůči počátku. Přímá kinematická úloha má formu soustavy rovnic, kterou lze zapsat s pomocí matic. V robotice se využívá speciálních eukleidovských prostorů, které zahrnou všechny souřadnice rotací a posunů. Pro obecný pohyb ve dvou dimenzionální prostoru jsme schopni polohu vyjádřit pomocí tří souřadnic, protože má tři stupně volnosti, nejčastěji používáme posun x,y a rotace ϕ.



Obrázek 3.1: schéma manipulátoru

3.2.1 Omezení úlohy

V našem případě se náš manipulátor bude pohybovat v rovině, ale nebude rotovat, takže nám budou stačit pouze dvě souřadnice a to posun po x a y. Pro výslednou polohu budou tedy potřeba dvě rovnice pro posun v osách x a y.

3.2.2 Umístění počátku

Pro vytvoření přímé kinematické musí nejprve umístit počátek v reálné světě, podle teorie ho můžeme umístit libovolně, takže ho umístíme co nejlépe tak, aby se výsledná úloha počítala co nejjednodušeji. Protože náš manipulátor se může pohybovat pouze v zeleném obdélníku na obr. 3.1 a vzhledem k orientaci os v levém spodní rohu na obr. 3.1 si zvolíme za počátek levý spodní roh zeleného obdélníka. Díky tomu se bude manipulátor pohybovat pouze v kladné části os x a y.

3.2.3 Motory

Motory, které používáme patří mezi tzv. krokové motory anglicky stepper motors. Tyto motory se pohybují po tzv. krocích, diskrétních částech rozsahu. Hlavní výhodou je přesnost, proto jsou tyto motory využívány v např. 3D tiskárnách. Krok je vlastnost motoru a uvádí se v počtu kroků na celou rotaci. Nás bude zajímat o kolik se posune manipulátor při každém kroku. Což můžeme vyjádřit jako část obvodu kladky:

$$l = \frac{\pi \cdot d}{n_{step}} = \frac{\pi \cdot 20\text{mm}}{200} \doteq 0,314\text{mm} \tag{3.1}$$

,kde l je posun na krok, d je průměr kladky a n_{step} je počet kroků na otáčku. Polohu motorů budeme měřit jako počet kroků od počátku, pokud se motor otočil vícekrát projeví se to na počtu kroků, např. 400 kroků znamená že se motor otočil o dvě otáčky v kladném směru otáčení.

3.2.4 Přímá kinematická úloha

S pomocí rovnice 3.1 a obr. 3.1 získáme rovnici pro pohyb po ose x:

$$x_{man} = -l \cdot x_{steps} \tag{3.2}$$

,kde x_{man} je x-ová poloha manipulátoru, l je konstanta z 3.1 a x_{steps} je poloha x-ového motoru. Mínus je v rovnici, protože kladný směr otáčení tedy i počtu kroků je podle obr. 3.1 proti směru hodinových ručiček. Protože manipulátor se po ose y pohybuje s posunem x díky konstrukci manipulátoru musíme to zahrnout do výsledné rovnice:

$$y_{man} = -l \cdot y_{steps} + l \cdot x_{steps} \tag{3.3}$$

,kde y_{man} je y-ová poloha manipulátoru, l je konstanta z 3.1 ,y_{steps} je poloha y-ového motoru a x_{steps} je poloha x-ového motoru. Mínus je v rovnici, protože kladný směr otáčení tedy i počtu kroků je podle obr. 3.1 proti směru hodinových ručiček.

3.3 Inverzní kinematická úloha

Inverzní kinematická úloha je opak přímé kinematické úlohy zde dostaneme zadanou polohu manipulátoru a ptáme se jak nastavit polohu jednotlivých

motorů, tak abychom jsme dosáhli zadané polohy. Tato úloha se řeší různými způsoby a u manipulátorů které jsou otevřenými kinematickými řetězci jako v našem případě může úloha mít žádné až nekonečno řešení, naštěstí v našem případě, kdy manipulátor nemá žádné otočné klouby, všechny jsou posuvné existuje řešení jenom jedno. Jeden ze způsobů jak získat inverzní kinematiku je matematicky a to vyjádřením z přímé kinematické úlohy, tento postup lze prakticky použít pouze pokud jsou rovnice jednoduché, což je náš případ. Takže s využitím rovnic 3.2 a 3.3 můžeme problém vyřešit:

$$x_{man} = -l \cdot x_{steps} \tag{3.4}$$

$$y_{man} = -l \cdot y_{steps} + l \cdot x_{steps} \tag{3.5}$$

Z této soustavy rovnic musíme vyjádřit proměnné týkající se motorů:

$$x_{steps} = -\frac{x_{man}}{l} \tag{3.6}$$

$$y_{steps} = \frac{l \cdot x_{steps} - y_{man}}{l} \tag{3.7}$$

Nyní dosadíme rovnici 3.6 do 3.7 a získáme:

$$x_{steps} = -\frac{x_m an}{l} \tag{3.8}$$

$$y_{steps} = -\frac{x_{man} + y_{man}}{l} \tag{3.9}$$

3.4 Limity

Náš manipulátor má fyzické omezení které se zatím neprojevilo v rovnicích přímé a Inverzní kinematiky, ale z praktický důvodů musíme zjisti jak daleko můžeme pohybovat klouby, aby nedošlo k poškození robota. V této části si popíšeme omezení poloh kterých robot může dosáhnout.

3.4.1 Limity manipulátoru

Vzhledem k umístění počátku souřadného systému známe spodní omezení na x-ové a y-ové ose to bude:

$$x_{man} > 0 \tag{3.10}$$

$$y_{man} > 0 \tag{3.11}$$

Omezení shora uděláme s pomocí konstant, které získáme z vlastností robota:

$$x_{man} < x_{max} \tag{3.12}$$

$$y_{man} < y_{max} \tag{3.13}$$

Tedy všechny body pro které platí nerovnice 3.10,3.11,3.12 a 3.13 jsou přípustné polohy manipulátoru.

3.4.2 Plánování trasy

Aby jsme mohli manipulátorem pohybovat mezi všemi přípustnými polohami musíme naplánovat trasu tak, aby manipulátor po cestě nepřekročil limity nebo nenarazil do překážky. Existuje více algoritmů, které se používají k tomuto účelu, my si však vystačíme s interpolací. Při plánování trasy s pomocí interpolace získáme více bodů po kterými trasa prochází, pokud jsou všechny body přípustné v rámci limitů manipulátoru, tak prohlásíme trasu za platnou a můžeme se po ní pohybovat. Tyto body naleznem pomocí následujícího vztahu:

$$x_n = x_p + \frac{n}{k}(x_k - x_p); n, k \in \mathbb{N}; n < k \tag{3.14}$$

$$y_n = y_p + \frac{n}{k}(y_k - y_p); n, k \in \mathbb{N}; n < k \tag{3.15}$$

, kde x_n je x-ová souřadnice n-tého interpolovaného bodu, x_p je x-ová počáteční poloha, x_p x-ová je koncová poloha, kde y_n je y-ová souřadnice n-tého interpolovaného bodu, y_p je y-ová počáteční poloha, y_p y-ová je koncová poloha, n označuje číslo bodu a k je počet interpolovaných bodů. Zbývá pouze ověřit, že všechny body jsou limitech.

Kapitola 4

Hardware

V této části se budu věnovat všem motorům, které jsem použil pro zprovoznění celé třídící jednotky.

4.1.1 Lego Mindstorms servo motory

Tyto motory jsem použil celkem dva pro pohon prvních dvou částí třídící jednotky, oddělovače dílů a jako pohon pásu rozpoznávací jednotky, použil jsem je z více důvodů. Prvním z důvodů je jednoduchost napojení na zbytek součástí třídící jednotky. Druhý důvod je jednoduchost zapojení, kdy stačí pouze příslušný kabel a poslední důvod je jednoduchost ovládání rychlosti. Tyto motory potřebují pouze držet konstantní rychlost při provozu což není složité naprogramovat.

4.1.2 Krokové motory

Pro ovládání manipulátoru, který třídí technické díly do jednotlivých přihrádek jsem zvolil Krokové motory Nema-17 na obrázku 4.1, které mají dostatek kroutivého momentu pro plynulý pohyb manipulátorem a zároveň se dají snadno ovládat z počítače. Jsou to bipolární krokové motory, které mají 200 kroků na otáčku. Použil jsem dva krokové motory pro ovládání obou os.

4.2 Napájení motorů

Obrázek 4.1: Motor Nema-17

Zatímco v předchozí části jsme se věnovali konkrétním motorům, v této části se budeme věnovat napájení a ovládání motorů.

4.2.1 Lego Mindstorms

Tyto motory jsou napájeny a ovládány z kostky Lego Mindstroms. Pro plynulý chod je kostka připojena do svojí nabíječky, protože napětí z baterie může ovlivnit rychlost otáčení motorů. Takže předejdeme možným problémům konstantním napájením z elektrické sítě. Právě ovládání motorů je důvodem proč nejsou tyto motory použity i pro manipulátor. Je velmi obtížné tyto motory ovládat z počítače, který řídí celý stroj. Nepodařilo se mi zprovoznit žádnou formu komunikace mezi EV3 kostkou a PC, i přes to že jsou spojeny USB kabelem.

4.2.2 Krokové motory

Ovládání Krokových motorů z počítače se ukázalo jako implementovatelná alternativa Lego servo motorů. K tomuto účelu jsem použil Arduino UNO a CNC shield které společně jsou schopné ovládat krokové motory.

Arduino UNO

Využité Arduino není originální, ale klonem který poskytuje všechny funkce a porty za přístupnější cenu. Programuje se ve vlastním prostředí Arduino IDE, syntaxi má podobnou jazyku C nebo C++ s přidanými knihovnami pro ovládání jednotlivých pinů.

CNC Shield

CNC shield je jedno z dostupných rozšíření desky Arduino, stejně jako další Shieldy doplňuje funkce které nejsou na samotné desce dostupné. Tento shield byl vyroben pro ovládání CNC strojů takže má všechny potřebné funkce pro ovládání připojených krokových motorů. Aby však mohl správně fungovat je potřeba přidat drivery pro ovládání jednotlivých motorů. Tento shield má sloty pro čtyři drivery. My využijeme dva sloty pro dva drivery a-4988. Zapojíme je do slotů označených písmeny X a Y. Protože Arduino není schopné poskytovat dostatečné napájení CNC shieldu tak je potřeba připojit externí 12V napájení. Externí napájení musí být schopné dodat dostatek proudu pro funkčnost obou krokových motorů.

A-4988

A-4988 je jeden z dostupných driverů krokových motorů, který funguje společně s předem zmíněným CNC shieldem. Při použití tohoto driveru je potřeba nastavit limit proudu procházející tímto obvodem. Nastavení jsem udělal pokusem, pokud se motor přehřívá limit snížíme, pokud motor vynechává kroky tak musíme limit zvýšit limit se nastavuje šroubovákem otočením potenciometru.

DRV8825

DRV8825 je alternativa driveru A-4988, uvádím ho zde jako alternativní možnost, protože funguje ve stejném obvodu, ale je schopný řídit větší proud než driver A-4988, což může být potřeba pro některé motory.

4.3 Kamera

Výběr kamery byla jedna z částí bakalářské práce. Moje požadavky na kameru byly následující: Velikost okna alespoň 640x480 pixelů, USB konektor a alespoň 30 snímků za sekundu. Vyšší rozlišení jsem nevyžadoval, protože jsem nechtěl prodloužit výpočetní čas a zároveň 640x480 je dostatečné rozlišení. Kamera zabírá plochu o délce zhruba 30cm, což nám dá zhruba 2 pixely na 1mm. USB konektor požaduji pro jednodušší připojení k počítači. A 30 snímků za

sekundu je dostatečné pro zachycení všech dílů ve správné poloze. Kamera kterou jsem tedy zvolil byla webkamera Webkamera Eternico Webcam ET101 HD, černá Webkamera *Eternico Webcam ET101 HD*.

Kapitola 5

Software

V této části se budeme věnovat programům použitých při řešení úkolu, konkrétně to bude řídící aplikace a kód pro Arduino UNO které řídí manipulátor.

5.1 Řídící aplikace

Řídící aplikace běží na PC a řídí celý proces třízení. My jí ovládáme pomocí klávesových zkratek, které provádí jednotlivé příkazy.

Help "h"

Po zadání tohoto příkazu se nám vypíší všechny dostupné příkazy, které byli implementovány.

Kamera "c"

Po zadání tohoto příkazu se nám otevře okno ve kterém se zobrazí to co vidí kamera. Okno se dá zavřít stiskem klávesy "q".

Detekce "d"

Po zadání tohoto příkazu začne kamera snímat pás a zapne detekční algoritmus. Při detekci kostky se uloží obrázek v moment detekce do adresáře ze kterého byl program spuštěn, zároveň se obrázek zobrazí pro uživatele. Tento proces se bude opakovat dokud nebude ukončen stiskem klávesy "q".

Nový díl "n"

Po zadání tohoto příkazu se nejprve vytvoří nový adresář v databázi, do kterého se poté přesunou všechny obrázky uložené detekcí z adresáře ze kterého byl program spuštěn. Tímto vytvoříme novou položku v databázi. Poté je už jenom potřeba upravit přiložený textový soubor.

Znouvunačtení databáze "r"

Po zadání tohoto příkazu se načte znovu databáze dílů. Příkaz slouží pokud se během běhu programu přidaly nové díly. Příkaz je také automaticky spuštěn při inicializaci řídící aplikace.

start "s"

Tento příkaz spustí celou třídící linku, zobrazí to co vidí kamera. Okno se dá zavřít stiskem klávesy "q".

Zavření aplikace "q"

Po zadání tohoto příkazu dojde k zavření řídící aplikace a všech oken které aplikace otevřela.

5.2 Databáze dílů

V této části se podíváme na databázi dílů. Databáze dílů jsem vytvořil jako složku ze které se načtou všechna data po spuštění řídící aplikace. Každý díl má v databázi vlastní složku očíslovanou podle pořadí přidání dílu do databáze. V této složce jsou uloženy všechny referenční obrázky použité pro identifikaci. Zároveň je v každé složce uložen textový soubor, který obsahuje informace o dílu.

5.2.1 Uložení nového dílu do databáze

Pro uložení nového dílu do databáze musíme nejdříve vytvořit dostatek referencí. Zvolil jsem pět referenčních obrázku na jednu polohu dílu kterou díl může zaujmout. Pro vytvoření záznamu použijeme řídící aplikaci. Začneme příkazem detekce zmáčknutím klávesy "d". Pokračujeme vložením dílu do třídící linky ten se při průchodu pod kamerou načte a vytvoří se referenční obrázek. Poté co předchozí proces zopakujeme dostatečně krát, já volím referencí na polohu kterou může díl zaujmout. Provedeme příkaz nový díl zmáčknutím klávesy "n". Řídící aplikace vytvoří novou složku do které přesune všechny vytvořené záznamy a vytvoří textový soubor s informacemi o dílu. Textový soubor poté upravíme doplněním informací o dílu.

Kapitola 6

Konstrukce linky

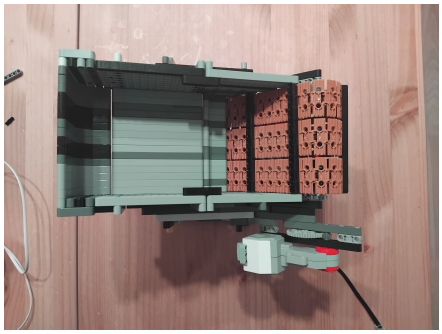
V této kapitole se budeme věnovat vlastnímu sestavení třídící linky, vlastnímu sestavení a funkčnosti.

6.1 Oddělovač dílů

Oddělovač dílů se skládá ze dvou na sebe navazujících částí, Zásobníku a vibračního oddělovače.

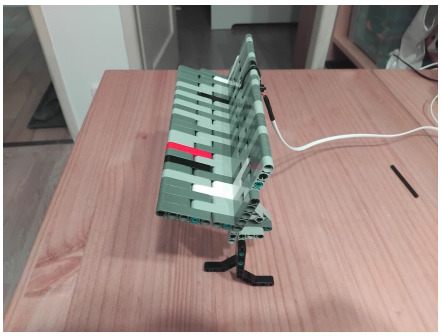
6.1.1 Zásobník

Zásobník má za úkol držet větší množství dílů a poté je postupně vydává dalším částem třídící linky. Na obr 6.1 můžeme vidět hotový zásobník. Pás na straně zásobníku slouží k postupnému vydávání dílů. Můžeme si povšimnout sklonu celého zásobníku díky kterému se díly posouvají a nakládají na pás s pomocí tíhové síly. Motor co pohání pás je napájen z kostky LEGO Mindstorms EV3 kde je mu nastavena konstantní rychlost, která je dále snížena mechanicky, pomocí převodů.

Obrázek 6.1: Zásobník dílů

6.1.2 Vibrační oddělovač

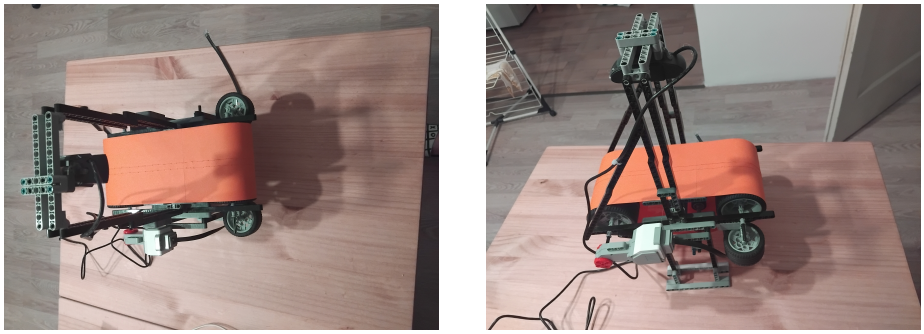


Obrázek 6.2: Vibrační oddělovač

Další díl linky je vibrační oddělovač. Skládá se z podélně nakloněné pravoúhlé plochy a vibračního motoru. Plocha je nakloněná tak aby součástky vždy zaujaly pouze jednu polohu na pásu detekční části. Kdybychom ji naklonili tak že obě pravoúhlé části mají stejný úhel k zemi tak by díly mohli zaujmout dvě polohy na pásu detekční jednotky. Vibrační motor není výrobek firmy LEGO a tak nemá standartizované připojení k ostatním částem. Pro jeho připojení jsem ho stáhl vázací páskou a přilepil horkým lepidlem, tak aby se vibrace přenášeli do celého oddělovače. Motor je napájen z USB konektoru, což si můžeme dovolit, protože motor požaduje dle výrobce pouze $1,5\text{mA}$.

6.2 Detekční jednotka

Detekční jednotka má několik hlavních částí, řadí se mezi ně pás po kterém se díly přesouvají a kamery užité pro identifikaci jednotlivých dílů.



Obrázek 6.3: Detekční jednotka

Obrázek 6.4: Detail pásu

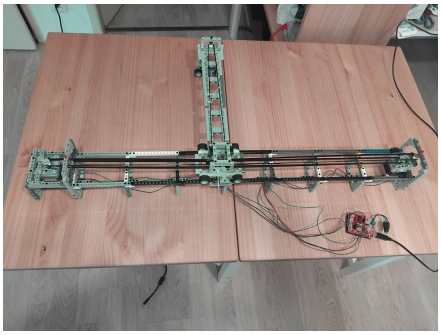
Pás byl vytvořen spojením dvou polovin oranžového barevného papíru jeho délka je tedy přibližně 60cm. Při používání jsem narazil na problém, kdy kulaté díly se odvalovali z pásu. Pro vyřešení tohoto problému jsem prošil pás oranžovou nití, která svojí výškou zastavuje odvalování kulatých dílů. Detail pásu je na obr. 6.4.

Pás je z papíru, což způsobilo problém s pohonem pásu. Pás neměl dostatek tření při pohonu, tento problém jsem vyřešil napnutím pásu pomocí napínače, který je na obrázku **??**. Napínání funguje pouze s váhou samotného napínače, kdyby to nedostačovalo tak máme dvě možnosti jak napravit tento problém. První je zvýšení váhy napínače, druhá je působí na napínač silou např. gumičkou.

Umístění kamery bylo provedeno tak aby řádky snímku byly rovnoběžné se směrem pohybu dílů po páse. Výška kamery byla zvolena tak, aby největší díl se vešel do snímku kamery a zbyly nám alespoň 4cm na obou stranách, toto místo potřebujeme protože detekce probíhá zhruba 4cm od kraje. Více v části 2.2.

6.3 Manipulátor

Poslední částí řešení je manipulátor, který umístí díly do přihrádek. Má dva stupně volnosti a je teoreticky popsaný v kapitole 3.



Obrázek 6.5: Manipulátor

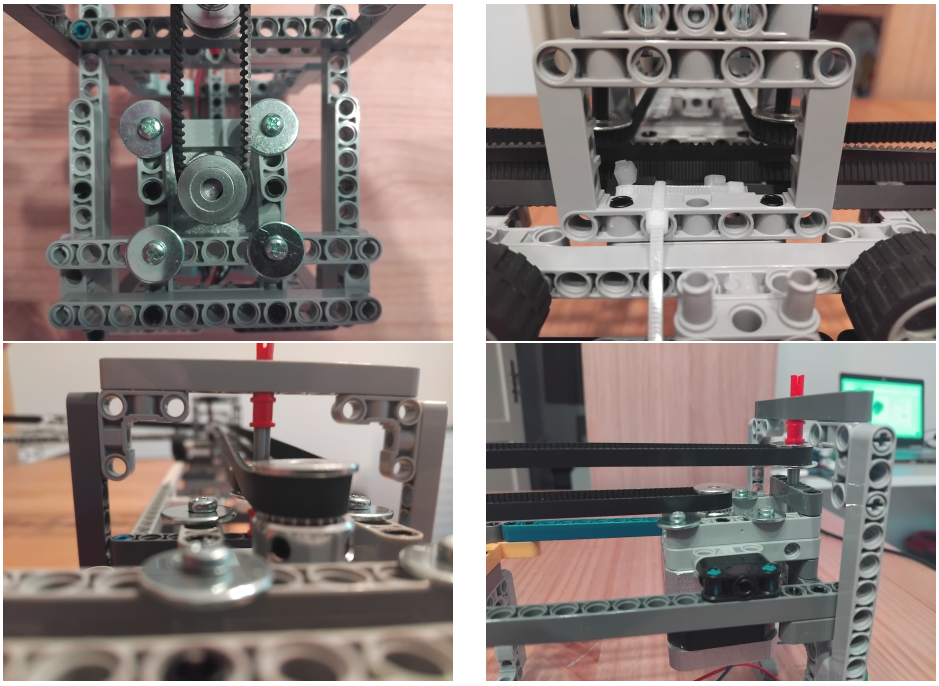
Na obrázku 6.5 můžeme vidět obě osy. Delší z nich je osa x. Uspořádání řemenů, které pohání celý manipulátor můžeme najít na obrázku 3.1, kde je naznačeno fialově společná cesta obou řemenů, ve skutečnosti jsou řemeny uloženy nad sebou červeně je potom řemen pohánějící osu y, modře pak řemen pohánějící osu x. Detail vedení pásu najdeme na obrázku 6.6.

6.3.1 Pohon os

Pohon os zajišťují krokové motory řízené Arduinem, v této části se budeme věnovat implementaci ovládání z PC. Pro funkční ovládání musíme poslat informaci Arduinu a spočítat inverzní kinematiku. Tu můžeme spočítat jak na PC tak Arduinu, protože je jednoduší opravovat kód přímo v počítači tak jsem se rozhodl spočítat co nejvíce v počítači. Počítač tedy spočítá inverzní kinematiku a z poslední známé polohy manipulátoru spočítá posun do nové polohy, jako rozdíl koncové a počáteční polohy. Tuto informaci poté předáme Arduinu, které se postará o řízení motorů a vytvoření potřebných signálů pro motory. Pro řízení směru motorů je potřeba nastavit příslušný pin Arduina do logické nuly nebo jedničky. Pro otáčení je potřeba obdélníkový signál na jiných pinech Arduina.

Uchycení motorů

Pro uchycení motorů NEMA-17 ke konstrukci ze stavebnice LEGO použijeme čtyři šrouby M3 s podložkami které se v katalogu označují jako M3 x 20 což nám označuje průměr a délku. Nepotřebujeme žádnou redukci mezi stavebnicí a motorem, protože závity v moterech pasují na otvory v lego technických dílech ve čtveci 5 x 5.



Obrázek 6.6: Detail vedení řemenů

Kladky a řemenice

Protože řemen se musí na svojí trase zatáčet, tak vyžaduje v každém ohybu kladka okolo které se otočí. Na osu motoru pasuje řemenice protože mají stejný průměr zajistíme jí na osu motoru utažením jejích vnitřních šroubů. Zajímavější je spojení ložiska kladky a technických dílů, které ji drží na svém místě. V tomto případě využijeme toho že vnitřní průměr kladky je 5mm a vnější průměr LEGO osy je 4,8mm. Vnitřek ložiska se tedy volně otáčí po LEGO technické ose, což nám v našem případě nevadí protože kladka stále plní svůj účel. Pokud by nám to vadilo bylo potřeba vyplnit prostor mezi osou a ložiskem nějakým materiálem, například by jsme mohli obalit osu několika vrstvami papíru.

Shrnutí

Začali jsme u problematiky Rozpoznávání technických dílů LEGO v kapitole 2, konkrétně detekce přicházejících dílů. Protože díly přichází do identifikátoru v náhodné časy tak jsem vymyslel algoritmus pro detekci dílů shrnutý rovnicí 2.1. Po detekci dílů musí následovat jejich identifikace, kde počítač vybere díl z databáze dílů, s nejnižším skórem popsaném rovnicí 2.2.

V další kapitole jsme se dostali do problematiky robotických manipulátorů , kde jsem nejprve zvolil vhodný souřadnicový systém. Ze schématu na obrázku 3 jsem vytvořil přímou kinematickou úlohu 3.2 . S využitím vlastností manipulátoru jsem z přímé kinematické úlohy odvodil inverzní kinematickou úlohu v části 3.3.Následovalo schéma manipulátoru na obr 3.1 Poté jsme se podívali na plánování trasy manipulátoru.

Poté následoval kapitola 4 věnující se všem využitým motorům a jejich fyzickým vlastnostem. Zároveň jsme se podívali na hardware využitý pro zprovoznění celé třídící linky. Na závěr kapitoly v části 4.3 jsem vysvětlil moje požadavky na kameru a konkrétní vybraný model

Následovala kapitola 5 věnující se ovládající aplikaci. Kapitola začal výčtem všech příkazů které řídící aplikace má.

Pak se kapitola věnovala databází dílů, jak se tvoří a jak přidat nové díly. Aktuálně se v databázi nachází přes 20 dílů, které jsem využil pro testování schopnosti identifikace linky.

V poslední kapitole jsme se věnovali vlastní konstrukci linky a hlavním myšlenkám za zvoleným způsobem konstrukce.

Literatura

[Fix51] Joseph L. Fix, Evelyn; Hodges, *Discriminatory analysis. nonparametric discrimination: Consistency properties*, 1951.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Dědek** Jméno: **Vojtěch** Osobní číslo: **474511**

Fakulta/ústav: **Fakulta elektrotechnická**

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

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Využití robota LEGO Mindstorms EV3 - návrh robota třídícího technické díly ze stavebnic LEGO Mindstorms NXT nebo EV3 pro propagaci FEL

Název bakalářské práce anglicky:

Usage of the LEGO Mindstorms EV3 - Design of a Robot Sorting Technical Parts from LEGO Mindstorms NXT or EV3 kits for Promotion of the Faculty

Pokyny pro vypracování:

1. Seznamte se s možnostmi robota LEGO Mindstorms NXT a LEGO Mindstorms EV3 (současný stav, HW a SW vybavení). Vyberte vhodnou kameru pro rozpoznávání technických dílů LEGO.
2. Proveďte a realizujte návrh robota třídícího technické díly ze stavebnic LEGO Mindstorms NXT nebo LEGO Mindstorms EV3 pro propagaci FEL (např. https://www.youtube.com/watch?v=04JkdHEX3Yk&t=5s)
3. Vytvořte databázi rozpoznávaných technických dílů pro další možné rozšíření třídicí linky
4. Vytvořte webové stránky k realizovanému projektu (popis, vysvětlení principu činnosti, vysvětlení navrženého softwaru, fotogalerii a popřípadě návod na stavbu robota).

Seznam doporučené literatury:

[1] James Floyd Kelly - LEGO MINDSTORMS NXT-G programming Guide, Second Edition
[2] Daniele Benedettelli - Programming LEGO NXT Robots using NXC
[3] https://www.youtube.com/watch?v=04JkdHEX3Yk&t=5s
[4] https://www.youtube.com/watch?v=fM9qGZCc4DY
[5] https://www.youtube.com/watch?v=nICSIJuD65A&t=93s
[6] https://www.youtube.com/watch?v=JD2jCn0Wm0Q

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Martin Hlinovský, Ph.D. katedra řídicí techniky FEL

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

Datum zadání bakalářské práce: **22.01.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

Ing. Martin Hlinovský, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. Datum převzetí zadání

Podpis studenta