UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2022 **Tomáš Vladyka**

UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

LABORATORNÍ ÚLOHY PRO ROBOTICKÉ RAMENO DOBOT MAGICIAN

Tomáš Vladyka

Bakalářská práce
2022

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš Vladyka**
Osobní číslo:	**I19048**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Řízení procesů**
Téma práce:	**Laboratorní úlohy pro robotické rameno Dobot Magician**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování

Cílem práce je vytvoření sady laboratorních úloh pro výuku práce s robotickým ramenem Dobot Magician.
Teoretická část práce bude obsahovat přehled komerčně dostupných robotických ramen vhodných pro výukové účely, a to jak z pohledu technického řešení, tak i softwarové podpory. Detailněji bude popsáno robotické rameno Dobot Magician.
Praktická část bude obsahovat laboratorní úlohy včetně zadání úlohy, teoretického úvodu, schémat zapojení a postupu řešení. Nejprve se bude jednat o jednoduché seznámení se s robotem, vyzkoušení základních pohybů, případně vysvětlení základních operací v jazyce Python. V dalších úlohách budou ukázány pokročilejší funkce ovládání vstupů/výstupů a připojení rozšiřujících modulů (pásový dopravník, lineární pojezd), ovládání robotu pomocí Arduina Mega a programování v dalších programovacích jazycích, například v Matlabu/Simulinku nebo C#.

Rozsah pracovní zprávy: **50**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

KOLÍBAL, Zdeněk. Roboty a robotizované výrobní technologie. Brno: Vysoké učení technické v Brně – nakladatelství VUTIUM, 2016. ISBN 978-80-214-4828-5.
CRAIG, John J. Introduction to Robotics, Global Edition. Harlow: Pearson Education Limited, 2021. ISBN 978-1-2921-6493-9.

Vedoucí bakalářské práce: **Ing. Daniel Honc, Ph.D.**
Katedra řízení procesů

Datum zadání bakalářské práce: **17. prosince 2021**
Termín odevzdání bakalářské práce: **13. května 2022**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Daniel Honc, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 7. ledna 2022

Prohlášení

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne

Tomáš Vladyka

ANOTACE

Cílem práce je vytvoření sady laboratorních úloh pro výuku práce s robotickým ramenem Dobot Magician. Práce obsahuje zadání laboratorních úloh včetně teoretického úvodu, schémat zapojení a postupu řešení. Úlohy jsou seřazeny tak, aby uvedly do problematiky i začátečníky bez předchozích zkušeností s programováním robotů.

KLÍČOVÁ SLOVA

laboratorní úlohy, Dobot Magician, edukativní robot, Python, praktické příklady.

TITLE

LABORATORY TASKS FOR THE DOBOT MAGICIAN ROBOTIC ARM

ANNOTATION

The aim of the work is to create a set of laboratory tasks for teaching how to work with the Dobot Magician robotic arm. The work contains the assignment of laboratory tasks including theoretical introduction, wiring diagrams and solution procedure. The tasks are arranged in such a way as to introduce even beginners with no previous experience in robot programming.

KEYWORDS

Laboratory tasks, Dobot Magician, Educational robot, Python, Practical examples.

OBSAH

SEZNAM ZKRATEK A ZNAČEK

3D	třírozměrný
ABS	akrylonitril-butadien-styren
AC	střídavé napětí
ACC	zrychlení
API	rozhraní pro programování aplikací
DC	stejnosměrné napětí
DIO	digitální vstupy/výstupy
DLL	dynamicky linkovaná knihovna
GPIO	obecné vstupy/výstupy
GUI	grafické uživatelské rozhraní
I/O	vstup / výstup
I2C	dvoudrátové rozhraní
ICSP	přímé programování po sériové lince
IDE	vývojové prostředí
IT	informační technologie
JOG	běh kloubu
JST	nepájivý konektor
LCD	displej s tekutými krystaly
LED	světelná dioda
MOVJ	kloubový pohyb
MOVL	lineární pohyb
PLA	polylaktidová vlákna
PTP	pohyb z bodu do bodu
PWM	pulzně šířková modulace
RGB	červená, zelená a modrá barva
RTU	jednotka pro přesun robotu
SDK	systémový vývojový nástroj
SPI	sériové periferní rozhraní
STEM	technické obory – věda, technologie, technika, matematika
UART	univerzální asynchronní přijímač-vysílač
USB	univerzální sériová sběrnice

SEZNAM SYMBOLŮ VELIČIN A FUNKCÍ

n	počet kroků
O	obvod, m
s	vzdálenost, m
v	rychlost, $m \cdot s^{-1}$
x	souřadnice osy X, mm
y	souřadnice osy Y, mm
z	souřadnice osy Z, mm

SEZNAM ILUSTRACÍ

ÚVOD

V posledních několika letech roste ve společnosti zájem o robotiku. Učitelé a školy využívají potenciál robotiky k praktickému a poutavému způsobu výuky programování nebo elektroniky. Využití edukativních robotů je nyní díky investicím do tohoto odvětví také vnímáno jako způsob, jak studenty motivovat ke kariéře v oborech STEM – vědy, techniky, inženýrství a matematiky. Mnoho zemí po celém světě již uznalo důležitost robotiky ve třídách a vytvořilo proto různé programy pro jejich výuku, které lze začlenit do vzdělávacího systému.

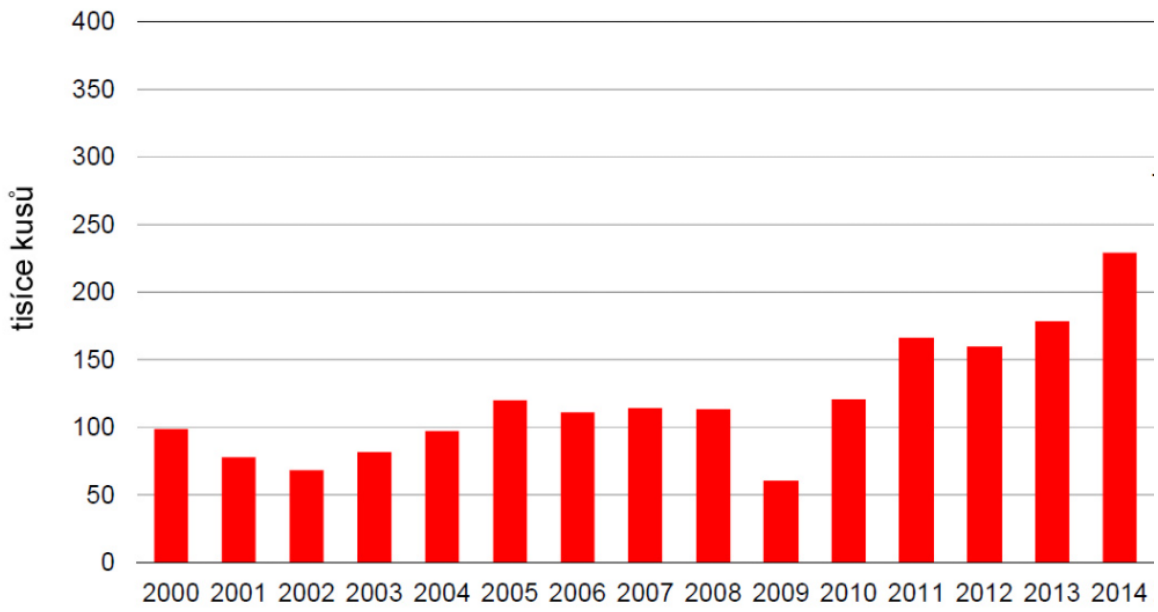
Cílem výuky STEM je pomoci studentům uspořádat informace v rámci jednotlivých oborů i napříč nimi, aby byli schopni identifikovat a zdůvodňovat podobnosti a zákonitosti informací. Při výuce robotiky mají studenti možnost spojit znalosti programování a vývoje aplikací s vědomostmi nabytými v předmětech s výukou elektroniky nebo mechaniky.

Tato bakalářská práce se zaměřuje na vytvoření laboratorních úloh, které mohou být využity při výuce. Úlohy budou zaměřené na robotické rameno Dobot Magician, které je svou univerzálností a dostupností vhodné pro výuku robotiky. Úlohy by měly provést studenty od úplných základů práce s robotem až po vývoj vlastních ovládacích aplikací. Studenti si budou moci vyzkoušet ovládat rameno pomocí vývojového prostředí DobotStudio, Visual Studio nebo Arduino IDE. K dispozici budou úlohy popisující práci s externími moduly a senzory dodávané firmou Dobot.

1 SOUČASNÁ VÝUKA ROBOTIKY

Roboty se stávají nedílnou součástí naší společnosti a mají velký potenciál pro využití jako vzdělávací technologie. Počet servisních robotů již v roce 2008 překonal počet robotů průmyslových. Roboty se pomalu začleňují do každodenního života v domácnostech a školách. Tento vliv robotiky je ještě zásadnější pro mladší generace, kde lze roboty využít pro jejich rozvoj a intelektuální růst (Mubin, 2013).

S rozvojem technologií se robotika úspěšně používá v různých odvětvích průmyslu a začleňuje tak do našeho každodenního života. Průmyslová robotická ramena nahrazují profese, které jsou pro člověka příliš namáhavé, monotónní nebo nebezpečné. U studentů robotika nabízí nový způsob výuky přírodních věd, zejména při jejím začlenění do standardních učebních osnov. Robotika může také podpořit zájem o vědecké a technické obory (Basoeki, 2013).



Obrázek 1.1 – Roční poptávka průmyslových robotů (Konference TMI, 2018)

Robotika je dnes jedním z klíčových prvků současné průmyslové a kulturní revoluce, která bude mít v blízké budoucnosti silný dopad na ekonomiku. Podle indexu digitální ekonomiky a společnosti dnes 90 % pracovních míst vyžaduje určitou úroveň digitálních dovedností (Konference TMI, 2018).

Hlavním důvodem nárůstu používání robotů je jejich klesající cena. V průběhu posledních deseti let ceny robotů klesaly, zatímco náklady na lidskou práci rostly. Roboty také nejen zlevňují, ale stávají se i efektivnějšími – rychlejšími, přesnějšími a flexibilnějšími. Pokud do čísel započteme tuto úpravy kvality, náklady na používání robotů klesají ještě rychleji než

jejich cena. S tím, jak se roboti stávají nákladově efektivnějšími při své práci, a lidská práce je stále dražší, stále více a více průmyslových pracovních míst se stává kandidáty na robotickou automatizaci (Craig, 2005).

Pro studenty vysokých škol je zásadní, aby se i nadále věnovali činnostem STEM a byli schopni uplatnit své dovednosti v reálném světě. V ideálním případě by měli rozšířit své chápání konceptů v oborech souvisejících s novodobými technologiemi prostřednictvím výzkumu.

Zejména úvodní kurzy automatizace a informatiky těží z aktivit zahrnujících roboty, protože ilustrují nejen jednotlivé aspekty strojírenství, elektrotechniky a IT, ale také to, jak spolu souvisejí v reálných aplikacích (Mead, 2005).

První ze tří rolí, které mohou roboty hrát ve výuce, je robot jako programovací projekt. Robotický systém je předmětem zadání, takzvanou černou skříňkou, kterou je třeba naprogramovat. Četné studie ukázaly, že špatné statistiky zapojení a udržení studentů v předmětech, které se zabývají úvody do programování, často pramení z neschopnosti studentů vidět, jak dovednosti, které se učí, mohou mít konkrétní dopad na fyzický svět. Fyzické projekty programování robotů mohou problémy kódování zasadit do reálného světa, čímž tyto dovednosti ovlivní jeho vnímání, a tím nabývají významu, který například výpočet Fibonacciho posloupnosti na obrazovce počítače nemůže vzbudit.

Druhá role vzdělávací robotiky spočívá v tom, že se robot zaměřuje na výuku. Roboti tak podněcují všeobecný zájem o vědu, techniku a inženýrství, a tím studentům ukazují, že se mohou aktivně podílet na utváření technologií v budoucnosti. Takto aplikačně zaměřená výuka má velký přínos v případech, kdy se žáci nejlépe učí prostřednictvím integrativní, projektově orientované výuky, a takové třídy prokázaly významné výsledky celoživotního učení v oblastech, jako je týmová práce, řešení problémů a ztotožnění se s profesemi zaměřenými na technologie.

Třetí rolí vzdělávacích robotů je role robota jako spolupracovníka při učení. V tomto případě žáci roboty nenavrhují, ale robot jim pomáhá s učením a pochopením probírané látky. Tuto roli nejčastěji zastávají roboty na základních a středních školách (Miller, 2008). Na trhu se momentálně nachází tři druhy možných řešení při nákupu robotického ramene pro edukativní účely.

Robot připravený k použití – ve většině případů se jedná o nejdražší, i když „časově úsporné“ řešení. Často jde o jednoúčelové stroje, určené jen pro daný druh práce. Umožňuje tedy jednoduché nasazení do předem vybrané aplikace, ale může být náročnější použít jej v prostředí, kde se požadavky na robot mění.

Stavebnice k sestavení robota je dobrým řešením, pokud je potřeba rozvinout představivost a manuální zručnost studentů. Sestavení robotu umožňuje studentům využít vědomosti získané teoretickou výukou, ale také zapojit dovednosti, jako je porozumění technické dokumentaci a komunikaci v týmu. Robotické stavebnice se řadí k těm levnějším způsobům, jak robotické rameno na trhu pořídit.

Obrázek 1.2 – Robotická stavebnice LAFVIN Smart robotic arm (Drátek, 2022)

Výroba robota od základu může být ekonomickým, ale časově náročným řešením. Naučit se vyrobit robota je výzva, která otestuje vědomosti i zkušenosti studentů v plné míře. Sestavení ramene zahrnuje využití celé řady dovedností – jasný postup při výběru potřebných komponent, navrhování elektronických a logických obvodů nebo kódování řídicího programu (Erasmus+ project, 2018).

2 ROBOTICKÉ RAMENO BRACCIO TINKERKIT

Braccio je stavebnice robotického ramene, které se dá složit z dodaných součástek a ovládat pomocí štítu s deskou Arduino. Rameno se skládá z celkem 6 servomotorů. Braccio lze sestavit mnoha způsoby v závislosti na požadovaném použití a na konci ramene lze také připevnit různé nástroje. Kromě klasických uchopovacích nástrojů lze s robotickou rukou Braccio použít například mobilní telefon, kameru nebo solární panel. Robotické rameno má dosah 52 cm a 6 stupňů volnosti (HW Kitchen, 2022).

Obrázek 2.1 – Robotické rameno Braccio Tinkerkit (Botland, 2019)

2.1 OBSAH BALENÍ

Robotická stavebnice Braccio Tinkerkit obsahuje modul Arduino Braccio Shield, který rozšíří vývojovou desku Arduino o dvanáct konektorů, které robot používá pro ovládání servomotorů a jiného příslušenství. Dále se v balení nachází šest servomotorů, dva typu SR 311 a čtyři typu SR 431. Samozřejmostí je samotná konstrukce robotu, která se skládá z jednadvaceti samostatných plastových částí. K sestavení robotu je dodávána i sada šroubů s podložkami a maticemi. V poslední řadě sada obsahuje 5 V zdroj napětí. Sada ovšem neobsahuje samotný vývojový modul Arduino, který je zapotřebí dokoupit samostatně. V České republice lze pořídit tuto sadu za cenu 5 899 Kč (RPishop, 2022).

Obrázek 2.2 – Obsah balení ramane Braccio Tinkerkit (Jameco, 2020)

2.2 PROGRAMOVÉ VYBAVENÍ

Jelikož je robotické rameno Braccio Tinkerkit založené na vývojové platformě Arduino, je k jeho programování využito prostředí Arduino IDE. Arduino IDE je zadarmo dostupný vývojový software. Obsahuje textový editor pro psaní kódu, textovou konzoli, panel nástrojů s tlačítky pro běžné funkce a řadu nabídek. Připojuje se k hardwaru Arduino a umožňuje nahrávat vytvořené programy.

Arduino IDE podporuje Braccio pouze s verzí 1.7.9 nebo novější. V knihovně Braccio jsou dvě hlavní funkce, a to Braccio.servoMovement a Braccio.begin. Braccio.begin inicializuje a nastavuje počáteční polohu pro rameno Braccio Arm. Všechny servomotory se nastaví do "bezpečnostní" polohy s každým úhlem následovně:

M1: Základna – 90 stupňů,

M2: rameno – 45 stupňů,

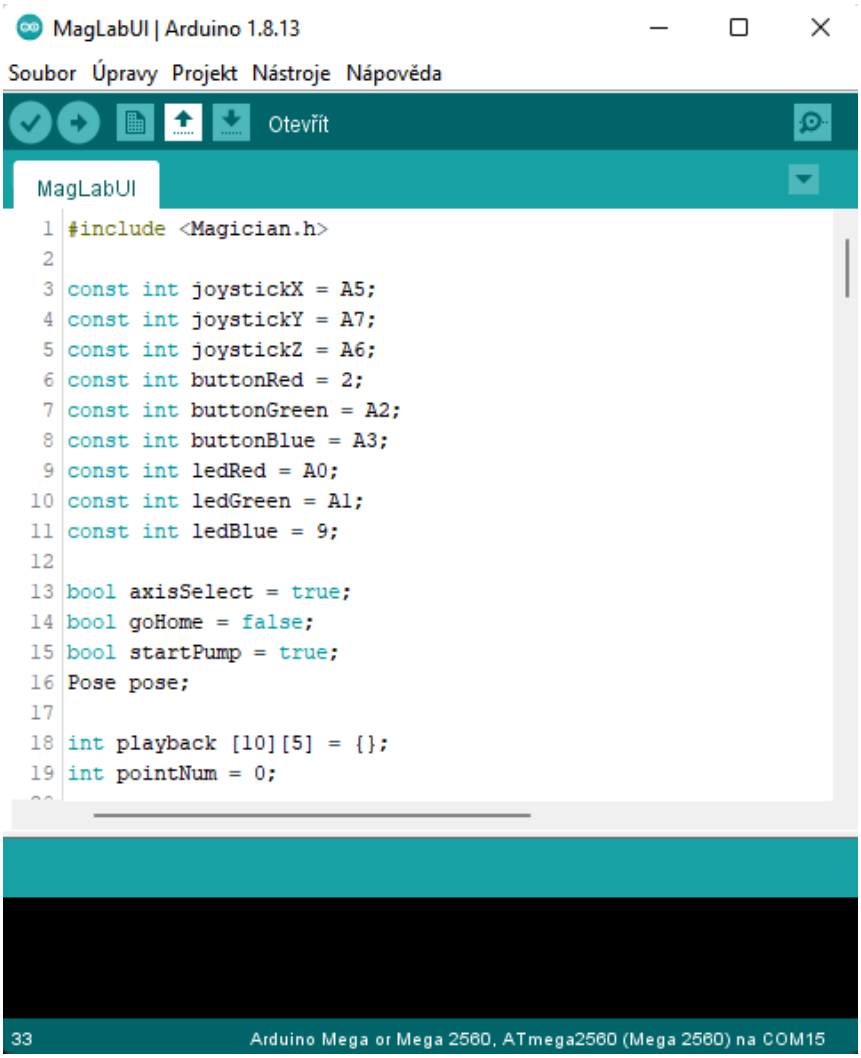
M3: Loket – 180 stupňů,

M4: Zápěstí – 180 stupňů,

M5: Rotace zápěstí – 90 stupňů,

M6: Nástroj - 10 stupňů.

Braccio.servoMovement umožňuje ovládat všechna serva Braccia pouze jedním příkazem, včetně milisekundové prodlevy mezi pohybem každého serva na začátku každého kroku a následně úhlu, o který se má každý motor pohybovat (Arduino, 2021).



Obrázek 2.3 – Programovací prostředí Arduino IDE (ArduinoIDE, 2022)

3 MYCOBOT PI

MyCobot Pi je multifunkční robotické rameno navržené a vyvinuté společností Elephant Robotics. Jedná se o šestiosý edukativní robot. Robot je založen na mikroprocesoru Raspberry Pi a vývojovém modulu M5Stack Atom. Je jedním ze základních robotů Elephant Robotics, který je široce používán ve vzdělávání v oblasti umělé inteligence. MyCobot je navržen jako modulární robot. Robot je vybaven 6 servy s rychlou odezvou, nízkou setrvačností a plynulým otáčením. Díky nim je možné dosáhnout vcelku velké přesnosti 0,5 mm.

Maximální zatížení ramene je 250 g včetně připojeného nástroje, s čímž musí uživatel počítat při návrhu aplikace. Hmotnost doporučeného uchopovače myCobot Adaptive Gripper je 88 g, a tím zmenšuje užitečné zatížení robotu na 162 g. Dosah ramene činí 280 mm od základny. Celý robot váží 850 g (Elephant Robotics, 2022).

Obrázek 3.1 – Robotické rameno myCobot (Elephant Electronics, 2022)

K připojení rozšiřujících modulů, jako jsou senzory, pohony nebo vzduchový kompresor, slouží 40 vstupně/výstupních portů, z toho 35 na základně robotu a 5 na konci šestého kloubu.

3.1 OBSAH BALENÍ

V balení robotu myCobot se nachází samotné robotické rameno, spolu s napájecím adaptérem. K robotu jsou přibalené i propojovací vodiče se zakončovacími konektory DuPont a uživatelský manuál k rychlému zprovoznění. Dodavatel za tuto sadu uvádí cenu 18 720 Kč (přepočteno z 799 $).

Obrázek 3.2 – Obsah balení ramene myCobot (myCobot User Manual, 2022)

Je na místě upozornit, že dodávaný robot neobsahuje žádné rozšiřující moduly ani uchopovače. Ty samozřejmě firma Elephant Robotics taktéž nabízí, ale je zapotřebí je dokoupit zvlášť. K dispozici jsou dva nástroje – servo gripper a vakuová přísavka. Servo gripper, nazvaný myCobot Adaptive Gripper, lze pořídit za cenu 3 000 Kč (129 $). Vakuová přísavka, spolu s kompresorem, který generuje podtlak, je k dispozici za 3 500 Kč (149 $) (Botland, 2022).

3.2 PROGRAMOVÉ VYBAVENÍ

Všechny moduly M5Stack jsou naprogramovány pomocí grafického editoru UIFlow, který je založen na blokovém programování. To samé platí pro myCobot Pi, jehož provoz zajišťuje vývojový modul M5Stack Atom a Raspberry Pi 4B. To vše dělá programování MyCobotu opravdu jednoduchým a snadným pro kohokoli, dokonce i pro začátečníky. Prostředí UIFlow je dostupné i jako webové programovací prostředí, dostupné z internetového

prohlížeče. Jednotlivé příkazy mají podobu barevných bloků kódu, které lze seskládat postupně pod sebe. Prostředí je inspirováno programem Scratch, který je často využíván pro výuku programování na základních školách.

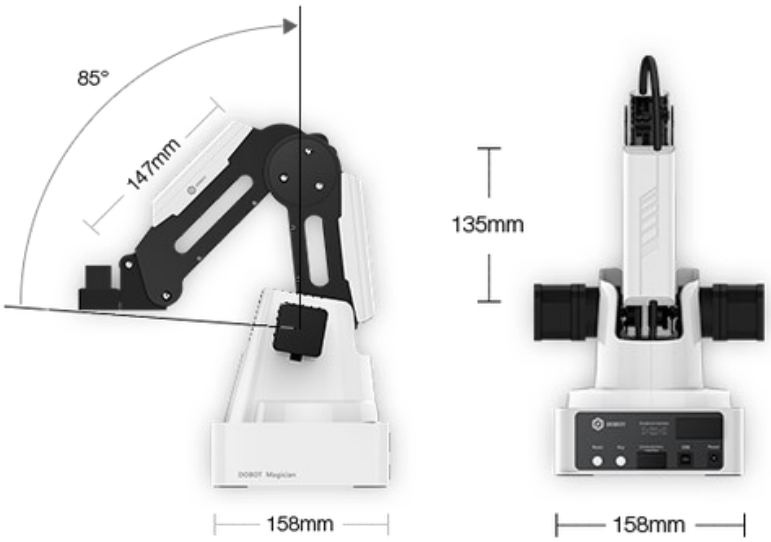
Obrázek 3.3 – Internetové vývojové prostředí UIFlow (m5stack, 2022)

Robot lze programovat i v jiných programovacích prostředích. Výrobce uvádí na svých stránkách podporu prostředí Arduino IDE, které bylo popsáno v kapitole 2.2. Kromě tohoto prostředí lze robotické rameno MyCobot programovat v textovém prostředí Python. Jednoduchá a snadno naučitelná syntaxe jazyka Python klade důraz na čitelnost, a tím usnadňuje uživateli orientaci v kódu. Python podporuje moduly a balíčky, což zvyšuje modularitu programu a zjednodušuje opakované použití kódu (m5stack, 2022).

4 DOBOT MAGICIAN

Dobot Magician je multifunkční stolní robotické rameno firmy Dobot. Jedná se o robotické rameno určené pro školy, které žákům umožňuje provádět řadu úloh s velice dobrou přesností 0,2 mm. Dobot Magician dokáže psát, kreslit, provádět 3D tisk, a manipulovat s předměty o hmotnosti až 500 g.

Na konec ramene lze jednoduchým povolením šroubu připevnit různé nástroje a nahradit tak tužku hlavou pro 3D tisk, uchopovačem nebo laserovým gravírováním. Toto robotické rameno určené pro univerzitní technologické instituty a vysoké školy má komplexní programovací prostředí s otevřeným zdrojovým kódem, mimořádně užitečné pro výuku STEM. Na stránkách výrobce lze najít odkazy ke stažení jeho SDK a aplikací, včetně nejen DobotStudia, ale také DobotBlockly, vizuálního programovacího editoru, Repetier-Host pro 3D tisk a dokonce kompletní knihovny pro sekundární vývoj pomocí jazyků Arduino, C# nebo C++ (Dobot, 2022).



Obrázek 4.1 – Rozměry robotu (Dobot, 2022)

Základní sada Dobot Magician je univerzálním řešením. Protože se jedná o robotické rameno určené pro vzdělávání, ponechává velký prostor pro fantazii a umožňuje na koncový efektor nainstalovat širokou škálu nástrojů. K dispozici je 13 programovatelných vstupně/výstupních portů pro instalaci mnoha senzorů a akčních členů. Robota navíc mohou díky rozhraní Blockly programovat i lidé, kteří s programováním teprve začínají.

K dispozici je i další příslušenství, včetně lineárního pojezdu a pásového dopravníku. K dispozici je také komplexnější výuková verze, zahrnující zejména moduly Bluetooth, WiFi a ovládání joystickem (DobotMagician, 2022).

4.1 OBSAH BALENÍ

Distribuci robotického ramene Dobot Magician v České republice zajišťuje firma ControlTech. Ta nabízí ke koupi dvě verze robotu – Basic a Educational. Tyto dvě verze se liší jak počtem doplňujícího vybavení robotu, tak cenovou nabídkou.

V balení verze Basic se kromě samotného robotického ramene nachází komunikační USB kabel spolu s napájecím adaptérem. Tento adaptér převádí klasické 230 V napětí na 12 V, které robot používá. Dále balení obsahuje sadu uchopovačů, kterými lze rameno vybavit. Prvním z těchto uchopovačů je pneumatický gripper. Ten se skládá z jednočinného pneumatického válce a čelistí, které válec otevírá a zavírá. Druhý uchopovač je vakuová přísavka. Ta se pomocí podtlaku přilne k sbíranému objektu, se kterým následně dokáže manipulovat. K těmto uchopovačům je dodáván speciální propojovací díl, který slouží k upevnění uchopovačů k robotu. Navíc obsahuje servomotor, který lze připojit ke konektorům robotu a tím přidává čtvrtou osu k ovládání natočení nástroje. Dále se v balení nachází jednoduchý držák na tužku, určený k psaní a kreslení robotem. Do tohoto držáku lze umístit celou řadu tužek, propisek nebo fixů, které jsou do držáku připevněny pomocí utahovacích šroubků. Držák navíc obsahuje pružinu, která zajišťuje správný přítlak tužky při kreslení. Poslední nástroj, který se ve verzi Basic nachází, je hlava pro 3D tisk. Ta obsahuje tiskárenský hotend, který se po připojení k robotu dá zahřát na teplotu vhodnou k tisku materiálů PLA nebo ABS. Díky tomu může robot zastat funkci klasické 3D tiskárny. Zároveň s tiskovou hlavou se v balení nachází 200 g PLA filament a podavač, který zajišťuje přísun filamentu do tiskové trysky. Balení robotu obsahuje i malý vzduchový kompresor pro ovládání pneumatického uchopovače a vakuové přísavky. Součástí balení je i sada nástrojů sloužících k upevnění nástrojů k robotu a jeho servisu. Firma ControlTech nabízí tuto variantu ramene za cenu 21 670 Kč (850 €).

Obrázek 4.2 – Obsah balení sady Educational (Dobot, 2022)

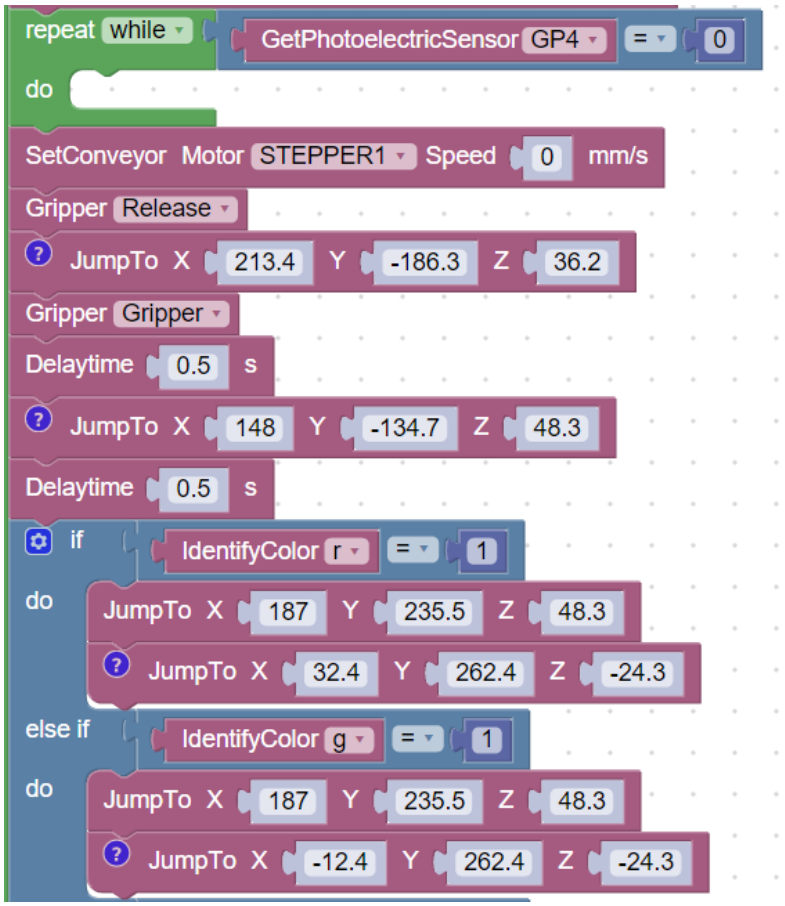
V rozšířené verzi Educational se nachází vše, co je součástí verze Basic. Navíc tato sada obsahuje hlavu pro laserové gravírování. Jedná se o 500 mW modrý laser s vlnovou délkou 405 nm. Jde tedy o slabší laser vhodný ke gravírování do materiálů, jako je dřevo, papír nebo kůže. K laseru se v balení dodávají i ochranné brýle, které slouží k filtraci vyzářeného modrého světla. Dalším vybavením v této sadě je ovladač, který lze k robotu připojit a ovládat jím jeho pohyby. Posledním přidaným příslušenstvím jsou moduly pro vzdálenou komunikaci s robotem. Jedná se o modul Bluetooth, díky kterému je možné robot ovládat přes IPhone nebo IPad, a WiFi modul, kterým lze robot připojit k počítači pomocí bezdrátové sítě. Tato sada je dražší než sada basic a od firmy ControlTech ji lze pořídit za 27 100 Kč (1 065 €) (ControlTech, 2022).

4.2 PROGRAMOVÉ VYBAVENÍ

Robotické rameno Dobot Magician lze ovládat pomocí programu DobotStudio, které je zdarma dostupné na stránkách výrobce. Prostředí DobotStudio nabízí dohromady osm různých možností ovládání ramene. Prvním způsobem ovládání je programovací prostředí Teaching & Playback. Toto prostředí nevyžaduje po uživateli žádné znalosti programování – celý program je tvořen „učením“ ramene. To může uživatel ručně přesunout do požadovaného bodu a tím vytvoří záznam v programu. Těchto bodů lze vytvořit téměř neomezené množství. Po stisku tlačítka start se robot začne přesouvat právě do takto zadaných pozic. Samozřejmostí je ovládání nástrojů, které jsou s robotem dodávány, vstupů a výstupů nebo nastavení doby pozastavení

programu. Toto prostředí slouží pro seznámení se s robotem a vytvoření jednoduchých aplikací, u složitějších programů se stává kód špatně čitelným.

Druhé programovací prostředí, které DobotStudio nabízí, je Blockly. Blockly je prostředí vyvinuté společností Google za účelem výuky programování. Uživatel nepotřebuje znát syntaxi žádného programovacího jazyka, ale jen základní logiku vytváření kódu. Jednotlivé příkazy zde mají podobu barevných bloků, které lze za sebe skládat a tím tvořit výsledný program. Na rozdíl od předešlého prostředí Teaching & Playback jsou zde zahrnuty pokročilejší funkce, jako jsou smyčky While a For, rozhodovací a porovnávací podmínky, nebo práce s textem. Nechybí zde ani příkazy pro ovládání samotného ramene, nastavení jeho parametrů nebo práce s dalším příslušenstvím. Díky tomuto prostředí je možné vytvořit komplikovanější aplikace relativně snadno. Prostředí Blockly navíc dokáže vygenerovat vytvořený kód v jazyce Python a je tedy možné program zkopírovat do jiného vývojového prostředí a následně upravit.

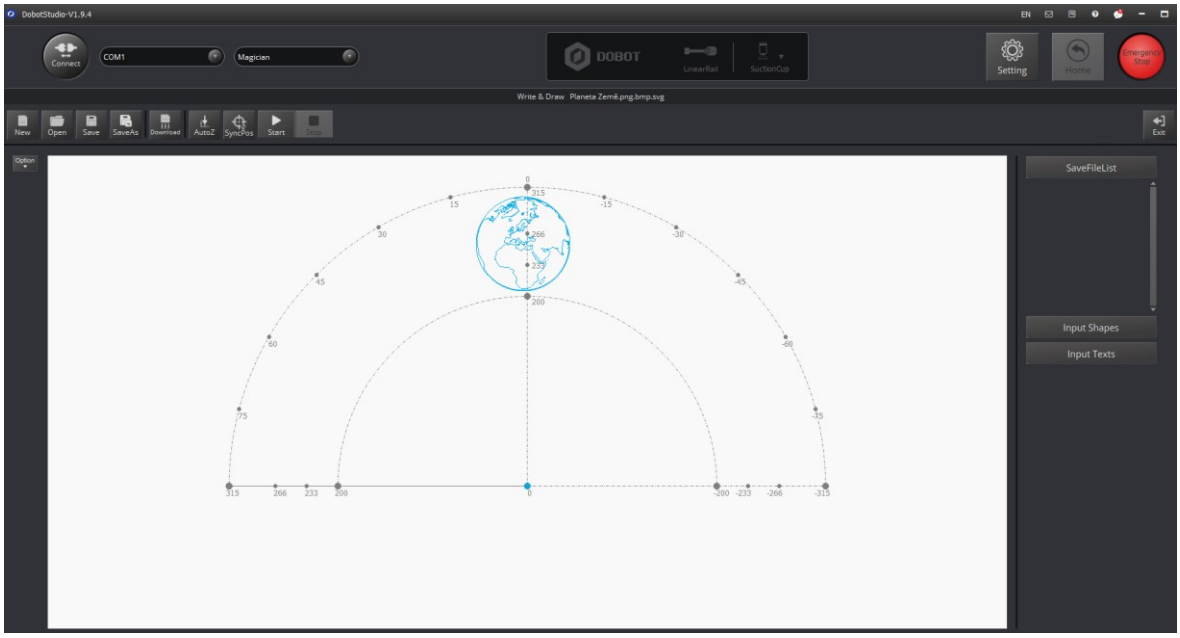


Obrázek 4.3 – Ukázka kódu v prostředí Blockly (DobotStudio, 2022)

Třetí programovací prostředí se nazývá Script. V tomto prostředí již lze kompletně využít všechny možnosti robotu. Je zde zapotřebí znát základní syntaxi jazyka Python, nicméně

veškeré příkazy pro práci s robotem jsou vysvětleny v přehledné nápovědě. Jazyk Python je vhodným programovacím jazykem i pro úplné začátečníky. Díky jednoduchým pravidlům není potřeba psát středníky na koncích příkazů nebo závorky pro vykonávání smyčky. Jazyk Python navíc podporuje funkci dynamického přetypování proměnných a díky tomu není potřeba proměnné inicializovat.

V prostředí DobotStudia se nachází také řada „neprogramovacích“ prostředí. Jedním z těchto prostředí je Write & Draw. Pokud je k robotu připevněn držák na tužku, případně nástroj pro laserové gravírování, je možné využít toto prostředí. Na pracovní ploše je znázorněn dosah robotu. Do tohoto prostoru je možné vložit obrázek z předpřipravené nabídky, případně vložit vlastní text. Možné je i použití vlastních obrázků. Podporovány jsou standardní soubory typu .jpg, .png nebo .bmp. Takto vložený obrázek je nejprve nutno převést na vektorovou grafiku, k tomu lze využít vestavěný konvertor. Jelikož při kreslení tužkou je použita pouze jedna barva, jsou barevné obrázky převedeny pouze na kontury, které jsou následně vloženy na pracovní plochu. Stiskem tlačítka Start robot automaticky spočítá trasy pohybu. Pro nakreslení obrazů nebo psaní textu tedy není zapotřebí znalostí programování.



Obrázek 4.4 – Ukázka prostředí Write & Draw (DobotStudio, 2022)

Podobné je prostředí LaserEngraving. Toto prostředí slouží pro gravírování pomocí laseru, které se v balení sady Educational nachází. Do tohoto prostředí je možné nahrát barevný obrázek, který je následně převeden do barev odstínů šedi. Díky nastavitelné intenzitě, se kterou dokáže laser pracovat, dokáže takto převedený obrázek vygravírovat do daného materiálu. Opět

zde není zapotřebí znalosti programování, po kliknutí na tlačítko Start robot automaticky začne proces vypalování (Dobot, 2022).

4.3 DOPLŇUJÍCÍ MODULY

Firma Dobot nabízí k ramenu Magician moduly, které dokáží rozšířit možnosti práce s robotem.

4.3.1 Pásový dopravník

Díky sadě pásového dopravníku je možné simulovat výrobní linku. V této sadě se nachází 600 mm dlouhý pásový dopravník, který je poháněný krokovým motorem. Motor lze připojit do základny robotu a díky tomu je možné jeho ovládání pomocí programu DobotStudio.

Obrázek 4.5 – Pásový dopravník pro rameno Dobot Magician (Dobot, 2022)

Maximální rychlost dopravníku je 120 $mm \cdot s^{-1}$, nosnost 500 g. K pásovému dopravníku jsou dodávány i senzory, díky kterým je možné vytvářet pokročilejší aplikace. Senzor barev dokáže rozpoznat barevné spektrum odraženého světla měřeného objektu. Dále je v balení infračervený senzor vzdálenosti. Tento dvoustavový senzor dokáže detekovat přítomnost objektu, který se před ním nachází. Detekční vzdálenost lze upravit šroubkem, který je na zadní straně senzoru. Oba tyto senzory jsou vybavené konektorem pro jednoduché připojení k robotu. Sada pásového

dopravníku také obsahuje čtyřicet pěnových kostek v červené, modré, zelené a žluté barvě. Firma ControlTech udává cenu za tuto sadu 7 210 Kč (RobotDobot, 2022).

4.3.2 Lineární pojezd

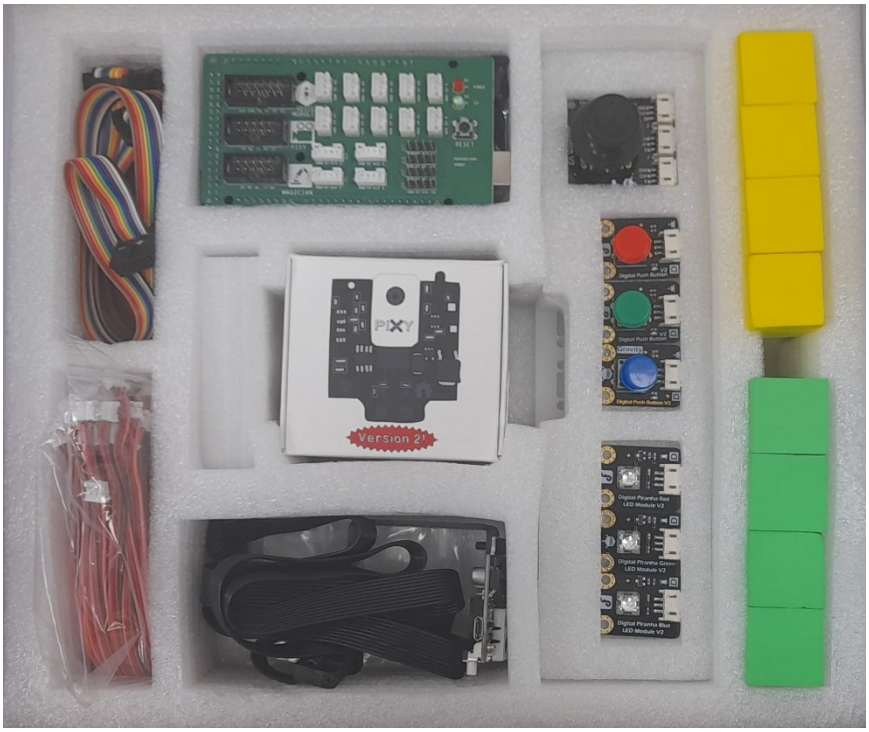
Dosah robotu lze zvětšit připojením lineárního pojezdu. Jeden metr dlouhý pojezd lze využít pro velkoformátové psaní a kreslení, laserové gravírování nebo vytvoření aplikací, kde je zapotřebí přesouvat objekty na větší vzdálenost. Lineární pojezd má maximální užitečnou nosnost 5 kg. Pojezd je poháněný krokovým motorem, díky čemuž lze dosáhnout přesnosti přesunu až 0,2 mm. Cena pojezdu je 19 890 Kč.

Obrázek 4.6 – Lineární pojezd (RobotDobot, 2022)

4.3.3 Arduino kit

Pro podporu druhotného vývoje nabízí firma Dobot sadu Arduino kit. Tato sada obsahuje klon vývojové desky Arduino Mega. Arduino Mega je deska s mikrokontrolérem založeným na ATmega2560. Má 54 digitálních vstupních/výstupních pinů (z nichž 14 lze použít jako výstupy PWM), 16 analogových vstupů, 4 konektory UART (hardwarové sériové porty), krystalový 16 MHz oscilátor, připojení USB, napájecí konektor, hlavičku ICSP a tlačítko reset. Obsahuje vše potřebné pro podporu mikrokontroléru; stačí jej připojit k počítači pomocí kabelu USB nebo napájet pomocí adaptéru AC-DC. Dále se v balení nachází vývojový modul Dobot Shield, který lze k Arduinu připojit. Tento modul seskupuje vstupně/výstupní piny do běžně používaných JST konektorů.

Kromě vývojové platformy sada obsahuje tři tlačítka, tři různobarevné LED diody – červenou, zelenou a modrou, a ovládací tříosý joystick. Tyto moduly lze jednoduše připojit k Dobot Shieldu pomocí JST konektorů (RobotDobot, 2022).



Obrázek 4.7 – Obsah sady Arduino

Dále sada obsahuje kameru Pixy pro jednoduché robotické vidění. Pixy zpracovává snímky z obrazového senzoru a vysílá do logické jednotky pouze nezbytné informace (např. při souřadnicích x = 54 a y = 103 detekce zelené kostky). Výpočet probíhá na standardní obrazové frekvenci 50 Hz. Informace jsou přenášeny pomocí rozhraní: UART, SPI, I2C, USB, digitálním nebo analogovým výstupem. Vývojové desky mohou s Pixy komunikovat snadno – nedochází k velkému zatížení procesoru, neboť většina výpočtů se provádí uvnitř samotného modulu s kamerou. Pixy používá k rozpoznání objektů filtrační algoritmus založený na barvě. Filtrační metody založené na barvě jsou populární, protože jsou rychlé, efektivní a relativně odolné. Dobře známá je kombinace tři základních barev RGB (červená, zelená a modrá). Pixy vypočítá barvu (barevný odstín) a nasycení každého pixelu RGB z obrazového senzoru a použije je jako primární filtrační parametr.

Arduino kit obsahuje i modul pro rozpoznávání řeči Grove Speech recognizer. Rozpoznávač řeči je určen pro aplikace hlasového ovládání, jako je inteligentní domácnost, inteligentní hračky nebo hlasové ovládání robotů. Deska obsahuje Nuvoton ISD9160, mikrofon, konektor SPI, Grove a konektor reproduktoru (seeed studio, 2022).

Kromě těchto rozšiřujících modulů obsahuje Arduino kit i sadu dvaceti pěnových kostek a konektory k propojení Arduina s robotem. Firma ControlTech nabízí tento produkt za cenu 5 880 Kč.

4.3.4 Robotické vidění

Robotické vidění umožňuje ovládání robotu bez potřeby zadávání přesných souřadnic. Díky tomu dokáže robot třídit objekty pomocí tvarů, barev nebo QR kódů. Obsahem sady robotického vidění je kamera AR0521, stojan a propojovací kabely. Dále se v balení nachází sada šestnácti kostek, kalibrační tabulka a objekty pro předchystané praktické příklady. K robotickému vidění je firmou Dobot vypracován obsáhlý uživatelský manuál, který do detailu popisuje způsoby programování robotu pomocí softwaru DobotVisionStudio. Díky tomuto programu lze rozpoznávat jednotlivé objekty, měřit jejich rozměry nebo porovnávat správnost objektů s předem naučenými vzory. Programování je ve formě funkčních bloků, které se vloží na pracovní plochu a podle potřeby pospojují. Tím mizí nutnost učení speciální syntaxe programovacího jazyka (DobotMagician, 2022).

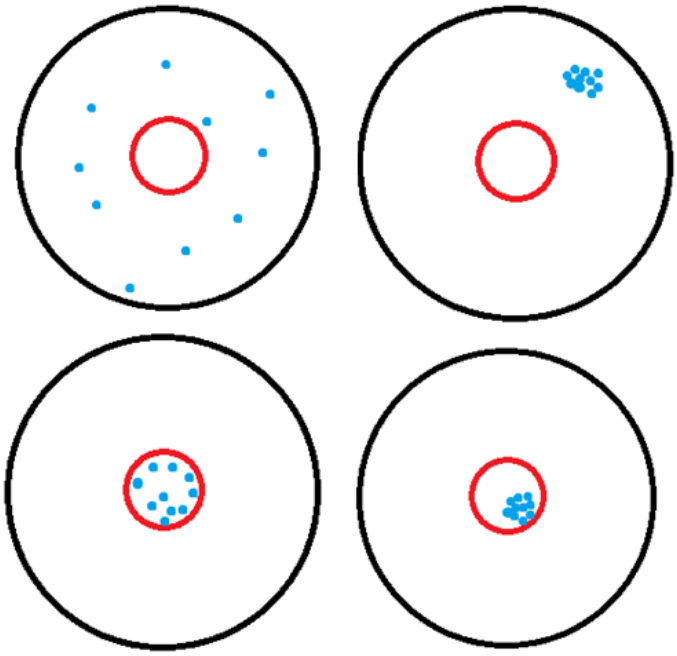
Obrázek 4.8 – Robotické vidění (DobotMagician, 2022)

5 LABORATORNÍ ÚLOHA MĚŘENÍ PŘESNOSTI

V této úloze studenti otestují přesnost a opakovatelnost robotu. Porovnají opakovatelnost robotu danou výrobcem se skutečnou při různých druzích pohybu a rychlostí.

5.1 ÚVOD

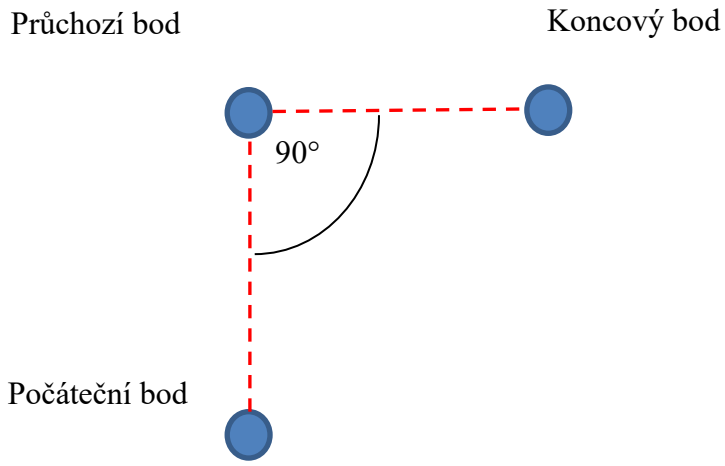
V dnešní době musí roboty splňovat řadu parametrů, aby mohly být úspěšně zařazeny do výrobního procesu. Mezi hlavní parametry při výběru správného robotu patří přesnost a opakovatelnost jeho pohybů. Pro mnoho lidí znamená přesnost a opakovatelnost jedno a to samé, nicméně se jedná o dva odlišné rysy robotu. V různých odvětvích průmyslu je kladen rozdílný důraz na tyto parametry, ale ani jednu z těchto vlastností nelze zanedbat. Přesnost určuje schopnost ramene přejít z libovolného počátečního do libovolného požadovaného bodu s požadovanou tolerancí za jakýchkoliv podmínek. Opakovatelnost označuje schopnost se opakovaně vracet ze stejného počátečního bodu do stejného koncového bodu za stejných podmínek. Pro názornost viz obrázek 5.1. Červená oblast představuje okruh tolerance přesnosti v požadovaném koncovém bodu pohybu. Modré tečky jsou koncové body pohybu ramene robotu do zadaného požadovaného koncového bodu.



Obrázek 5.1 – Druhy přesnosti a opakovatelnosti

5.2 POSTUP

K měření přesnosti a opakovatelnosti robotického ramene Dobot Magician studenti vytvoří jednoduchý program, který bude obsahovat tři body pohybu – počáteční bod, koncový bod a bod, přes který bude rameno procházet. Využito je prostředí Teaching & Playback. Počáteční a koncový bod budou mezi sebou svírat úhel 90°, který je vytvořen pomocí třetího bodu. Pohyby jsou zaznamenány v režimu MovL. Trajektorie pohybu při pohledu na robot zpředu vypadá tak, jak je znázorněno na obrázku 5.2.



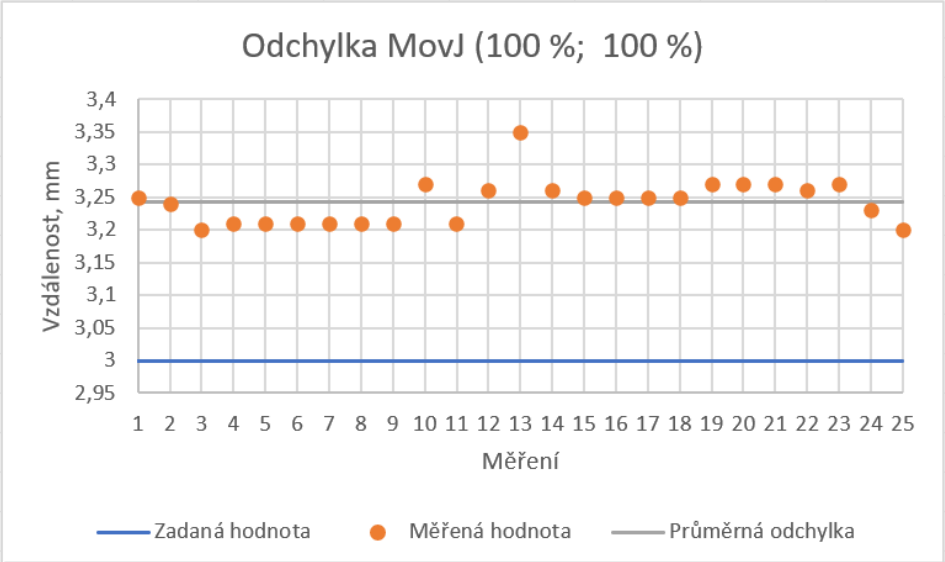
Obrázek 5.2 – Trajektorie pohybu

Pokud robot správně vykoná naprogramovanou trasu, je možné přistoupit k další části měření. K určení přesnosti a opakovatelnosti ramene bylo použito úchylkoměru. Úchylkoměr byl pevně připevněn k pracovnímu prostoru robotu tak, aby rameno tlačilo na měřicí hrot. Hodnota na ciferníku úchylkoměru určuje pozici koncového bodu, do kterého se bude rameno vracet. Rychlost i zrychlení ramene se nastaví v horní části prostředí na 100 %. Program je znovu spuštěn. Po dokončení pohybů robotu se do tabulky zaznamená aktuální hodnota na ciferníku úchylkoměru. Takto je zaznamenán dostatečný počet vzorků k určení přesnosti pohybu MovL. Stejným způsobem se provede měření pro pohyb MovJ s rychlostí a zrychlením 100 %; 100 % a 50 %; 50 % a 100 %.

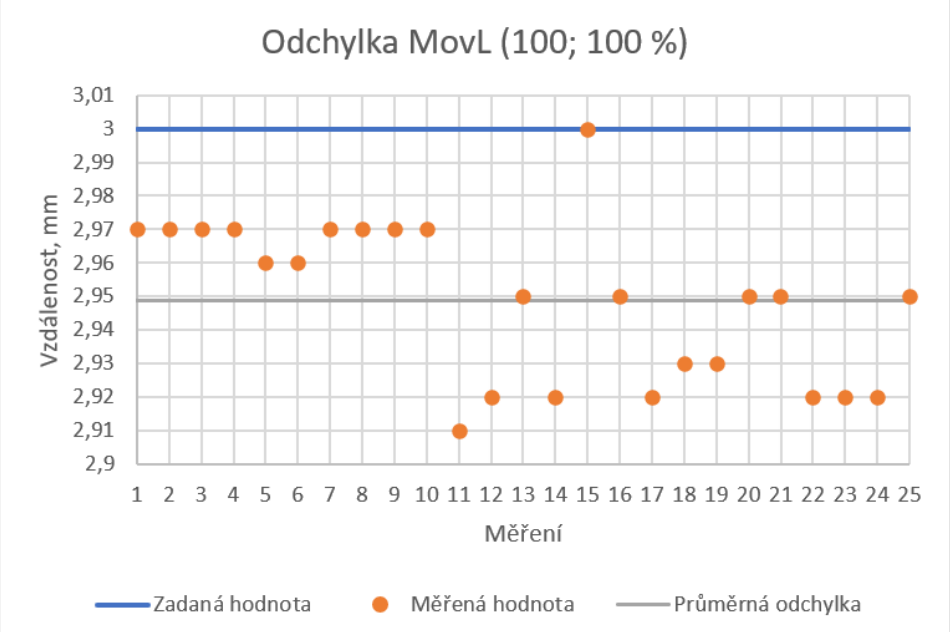
V závěru je uvedeno, zda změřená opakovatelnost odpovídá údajům uvedených výrobcem. Zhodnocena je přesnosti jednotlivých druhů pohybů a jejich rychlostí/zrychlení.

5.3 VÝSLEDKY ÚLOHY

Z výsledných hodnot vychází, že lineární pohyb ramene MovL je nejpřesnějším druhem pohybu. Přesnost tohoto pohybu je ± 0,09 mm. U pohybu typu MovJ dochází k odchylce přesnosti až 0,35 mm. V ostatních případech je opakovatelnost lepší, nejlépe vychází opět pohyb MovL, kde se opakovatelnost rovná 0,05 mm.



Obrázek 5.4 - Odchylka pohybu MovJ



Obrázek 5.3 – Odchylka pohybu MovL

6 LABORATORNÍ ÚLOHA DRUHY POHYBŮ

Tato úloha se zabývá jednotlivými druhy pohybů robotického ramene Dobot Magician. Studenti se naučí základům programovacího jazyku Python a porovnají trajektorie pohybů robotu.

6.1 ÚVOD

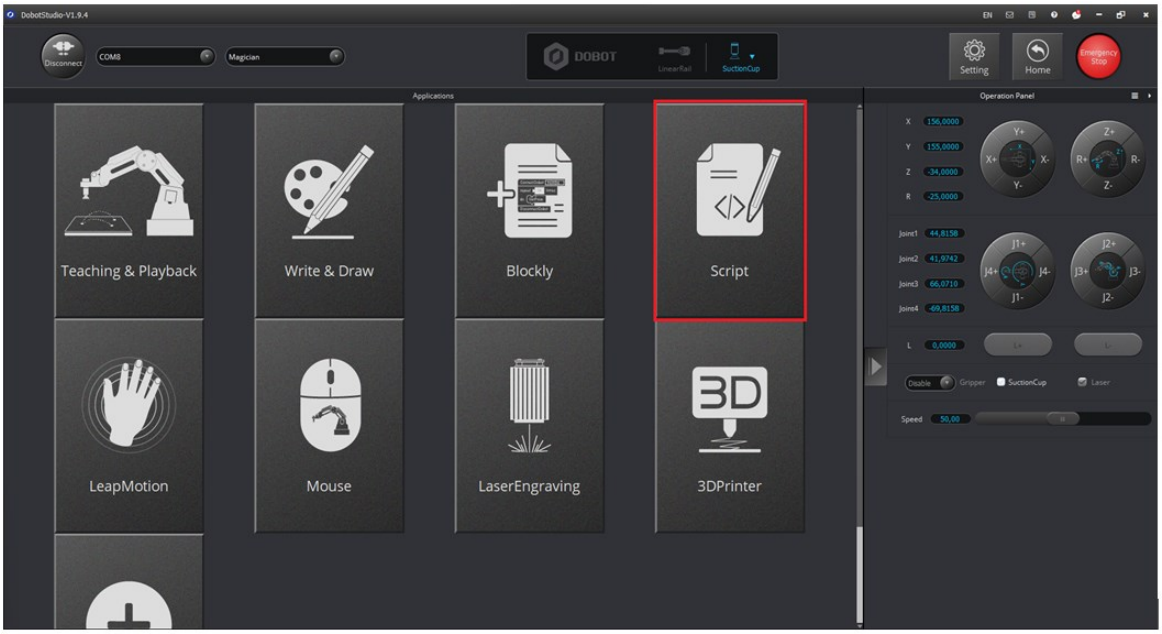
V dnešní době se z jazyku Python stává jeden z nejpřístupnějších a nejuniverzálnějších programovacích jazyků na světě. Jedná se o vysokoúrovňový programovací jazyk, který je podporován na většině běžně používaných platformách, jako je Windows, Linux, macOS nebo Android. Vývojový jazyk Python využívá řada velkých společností, mezi nejznámější patří Google, NASA nebo Facebook. Od roku 1991, kdy vyšla první veřejná verze Python 0.9.0, se neustále vyvíjí. V roce 2000 vyšla nová verze Python 2.0 a v roce 2008 verze Python 3.0. Od roku 2020 pokračuje ve vývoji pouze Python 3.0, ostatní verze již nejsou nadále podporovány.

Jednou z vlastností, která jazyk Python odlišuje od většiny ostatních moderních jazyků, je kompletní absence složených závorek a středníků na konci příkazů. Místo toho používá mezer nebo tabulátorů a odřádkování příkazů. Tím je docíleno grafického strukturování kódu a tím i lepší čitelnosti a orientaci. Nedochází tak k složitému hledání vynechaných středníků a neuzavřených závorek.

Python navíc nabízí i širokou škálu různorodých knihoven. Tyto knihovny lze zakomponovat do programu a využívat příkazy, které by se jinak musely složitě vytvářet. Mezi nejznámější a nejvíce využívané knihovny patří SciPy. Ta označuje knihovnu Scientific Python, která se dá využít pro řešení složitých výpočtů. Je často využívána vědci nebo inženýry při vývoji aplikací. PyGame poskytuje snadné rozhraní pro grafické a zvukové příkazy. Používá se pro vývoj videoher. Knihovna Matplotlib je zodpovědná za vykreslování číselných hodnot. Lze ji využít pro zobrazení dat v grafech, ať už koláčových nebo bodových, nebo histogramech.

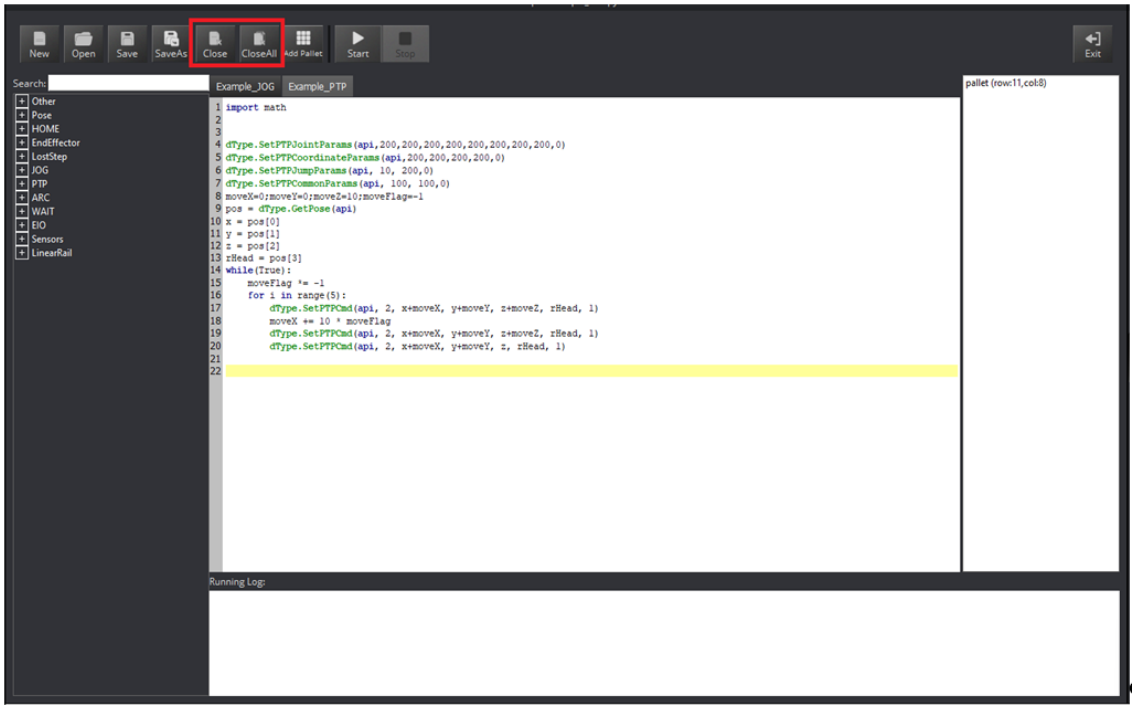
6.2 POSTUP

V ovládacím prostředí DobotStudio je otevřeno programovací prostředí Script (viz obrázek 6.1). Pro správnou funkčnost prostředí je nezbytné připojit robot pomocí tlačítka Connect v levém horním rohu programu.



Obrázek 6.1 – Spuštění programovacího prostředí Script (DobotStudio, 2022)

V prostředí jsou již otevřeny dva programy – Example_JOG a Example_PTP. Tyto programy lze zavřít tlačítky Close nebo CloseAll v levém horním rohu prostředí (viz obrázek 6.2).



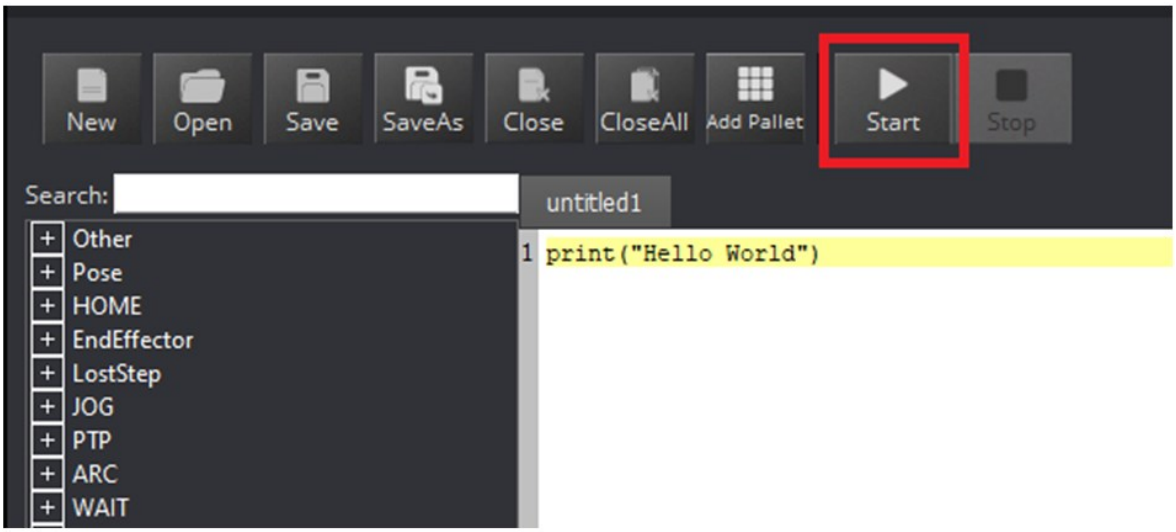
Obrázek 6.2 – Tlačítka pro zavření programů (DobotStudio, 2022)

Tlačítkem New vlevo od tlačítek pro zavření programu vytvořte nový soubor. Nyní je možné začít psát vlastní programy.

Studenti vytvoří první program: Hello World. Tento program má jedinou funkci – vytisknout do výstupní konzole text „Hello World“. K tomu byl použit příkaz print(). Pokud je do tohoto příkazu zapsán text v uvozovkách, vytiskne se do výstupní konzole. Program by tedy měl vypadat takto: print(“Hello World“).

Oproti jiným programovacím jazykům není potřeba za příkazy psát středník. Místo toho jsou jednotlivé příkazy odděleny odřádkováním – tedy lze napsat pouze jeden příkaz na řádek. Stiskem tlačítka Start v horní části prostředí dojde ke spuštění programu.

Po stisku tlačítka Start jsou slova „Hello World“ vypsána v konzoli (Running Log) v dolní části prostředí.



Obrázek 6.3 – Tlačítko Start (DobotStudio, 2022)

Další důležitou částí každého programovacího jazyku jsou proměnné. Jazyk Python disponuje dynamickým typovým systémem – proměnné není třeba deklarovat, ani jim nemusí předcházet datový typ. Python používá základní datové typy – int, float, double, char, string, ale datový typ proměnné si sám určí podle jejího obsahu. To lze vyzkoušet jednoduchou úpravou předchozího programu. Vytvořením proměnné ‚*a*‘ a přiřazením řetězce se slovy „Hello World“ je možné vytisknout obsah této proměnné do konzole.

Výsledek tohoto programu je stejný, jako programu předešlého. Symbol # se používá pro zapsání komentáře. Cokoliv se nachází za tímto znakem je programem ignorováno. Toho se dá využít pro vepsání poznámek/komentářů do programu, nebo ignorování příkazu bez nutnosti jeho smazání.

Jelikož není potřeba deklarovat datové typy proměnných, jde stejným způsobem vytvořit celočíselnou proměnnou ‚*b*‘, přiřadit k ní celé číslo a opět nechat vytisknout do konzole pomocí příkazu print().

V této úloze je potřeba znát smyčku while(). Uvnitř závorek smyčky while se nachází podmínka. Dokud je tato podmínka splněna, cokoliv, co se nachází uvnitř smyčky, se opakuje. Jakmile je podmínka nesplněna, smyčka se ukončí a program pokračuje dál od konce smyčky. Základní znaky pro podmiňovací výrazy jsou: == (rovná se), != (nerovná se), < (menší než), > (větší než), <= (menší nebo rovno), >= (větší nebo rovno). Na rozdíl od běžných programovacích jazyků není třeba vnitřek smyčky ohraničit závorkami – smyčka se odliší od zbytku programu odstavením od okraje o jednu úroveň tabulátoru. K vyzkoušení smyčky byl vytvořen program, který počítá od 1 do 5 za použití celočíselné proměnné. Čísla jsou vypsána do konzole.

Do podmínky smyčky lze napsat i podmínku složenou. Jednotlivé výrazy složené podmínky se oddělí logickými operátory *or* (nebo) a *and* (a zároveň), podle toho, zda stačí aby byla splněna jedna ze zadaných podmínek, nebo musí být splněny všechny zadané podmínky. Program by se nikdy neměl dostat do nekonečné smyčky. Pokud ale nekonečná smyčka nastane, je možné program zastavit tlačítkem Stop vpravo od tlačítka Start.

Další důležitou částí v této úloze je práce se souborem, přesněji se zapisováním hodnot do souboru. Pro otevření souboru je potřeba vytvořit proměnnou souboru, se kterou bude program dále pracovat. K této proměnné je potřeba přiřadit a otevřít daný soubor za použití příkazu open(*cesta_k_souboru, režim_otevření*). Jako parametr *cesta_k_souboru* je potřeba zapsat celou cestu souboru, jak je dána prohlížečem souborů. Ta je dána třemi částmi – cesta ke složce, ve které se daný soubor nachází, jméno souboru a jeho koncovka. Dále je nutné specifikovat, v jakém režimu se má soubor otevřít. Do parametru *režim_otevření* lze vepsat písmena ‚r', ‚w' a ‚a'. Pokud je použit režim ‚r', soubor se otevře pro čtení a lze přečíst obsah souboru. Režim ‚w' označuje otevření souboru pro zápis. Pokud soubor uvedený v parametru *cesta_k_souboru* neexistuje, bude vytvořen nový soubor s daným názvem. Pokud takový soubor již existuje, tento režim smaže celý jeho obsah a začne zapisovat data od začátku. Režim ‚a' funguje podobně jako režim ‚w', ale pokud dojde k zápisu do již existujícího souboru, tento režim nemaže jeho obsah. Místo toho připíše data na konec tohoto souboru.

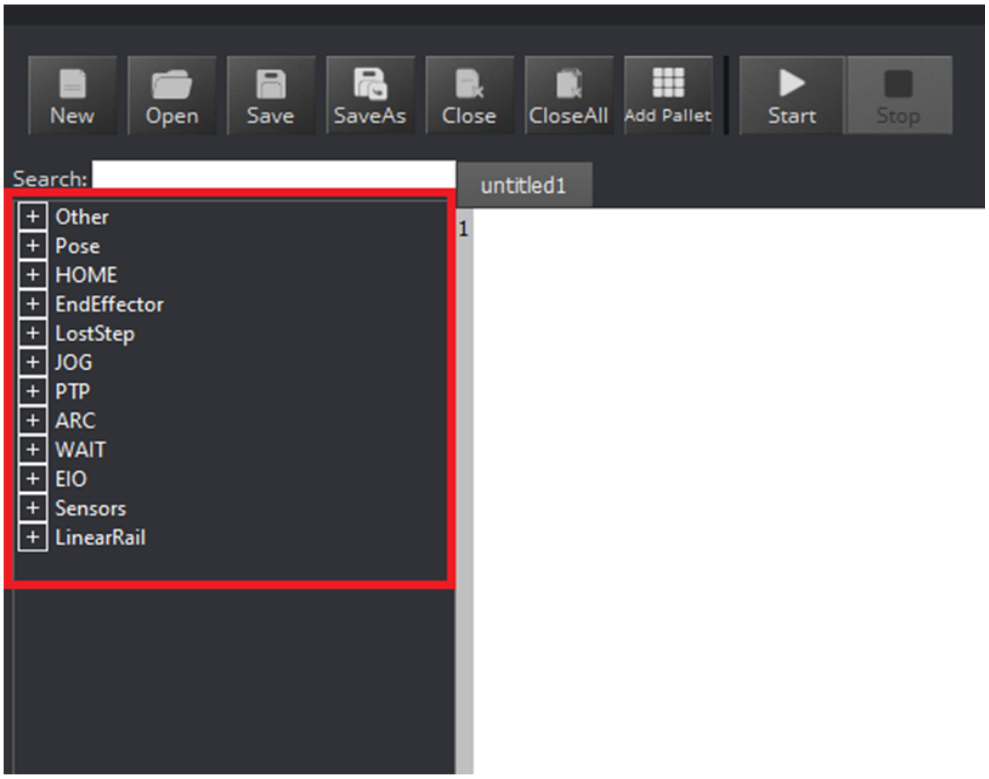
Po otevření souboru v režimu čtení (‚r'), je možné přečíst obsah souboru příkazem *proměnná*.read(). Tento příkaz vrátí celý obsah tohoto souboru. Pokud je potřeba obsah vytisknout do konzole, může se tento příkaz vepsat do příkazu print(). Při ukončení práce se souborem je potřeba vždy soubor zavřít příkazem *proměnná*.close(). Na ploše se vytvoří nový textový soubor a zapíše se do něj libovolná věta (Pozor! Není možné použít znaky s diakritikou). K vyzkoušení příkazů se napíše program, který tento soubor otevře v režimu čtení, vypíše obsah souboru do konzole a soubor zavře.

Dále je vytvořen program pro otevření souboru v režim pro psaní (‚w‘). Do souboru se programem zapíše jiná věta. K zapsání dat do souboru slouží příkaz *proměnná*.write(*text*). Za parametr *text* lze dosadit proměnou, nebo řetězec znaků v uvozovkách.

Původní věta v souboru se přepíše na větu uvedenou v programu. Jak již bylo řečeno výše, otevření souboru v režimu ‚w‘ vždy přepíše původní data. Pokud je zapotřebí data do souboru přidat tak, aby byla původní data zachována, je nutné otevřít soubor v režimu ‚a‘. K vyzkoušení práce se souborem je vytvořen program, který do souboru přidá další libovolnou větu.

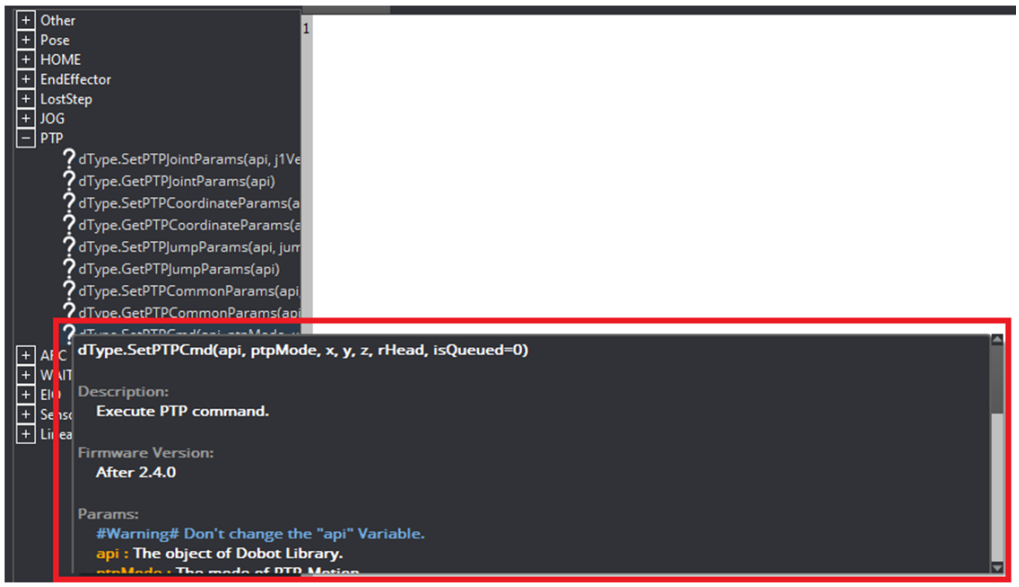
Tato věta se připíše hned za konec věty předchozí. Pokud je potřeba jednotlivé řádky oddělit, využije se znaku pro nový řádek ‚\n‘. Tento znak se připíše ke konci předchozího řádku, nebo na začátek řádku následujícího.

Poslední část úlohy se zabývá příkazy pro robota. Programování robotu v jazyce Python je nejkomplexnějším ovládacím prostředím, které DobotStudio nabízí. Na rozdíl od běžných příkazů používaných v Pythonu není zapotřebí znát jednotlivé příkazy pro operace s robotickým ramenem nazpaměť. Všechny potřebné příkazy lze najít v knihovně v pravé části prostředí DobotStudia, viz obrázek 6.4.



Obrázek 6.4 – Knihovna příkazů (DobotStudio, 2022)

Knihovna je rozdělena na dvanáct kategorií. Kliknutím na symbol ‚+' se objeví určitý počet příkazů týkajících se dané kategorie. Například rozkliknutím kategorie PTP (Point To Point, čili pohyb z bodu do bodu) se objeví příkazy týkající se nastavení a provádění různých druhů pohybu. Každému příkazu předchází znak otazníku. Po kliknutí na otazník se otevře nápověda týkající se daného příkazu, viz obrázek 6.5.



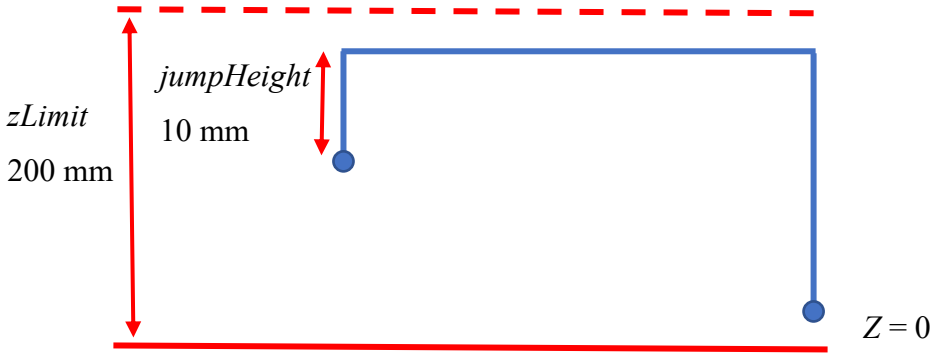
Obrázek 6.5 – Nápověda příkazu (DobotStudio, 2022)

V nápovědě je vidět celý přepis příkazu, včetně všech jeho parametrů. Pod příkazem se nachází popis, co daný příkaz provádí. Nakonec se v nápovědě nachází popis parametrů, které se dají v příkazu nastavit, spolu se seznamem výstupních proměnných. Dvojklikem na příkaz dojde k přepsání příkazu do vývojového prostředí. V této úloze jsou využity tyto příkazy:

- dType.SetPTPJumpParams(api, *jumpHeight*, *zLimit*, isQueued=0),
- dType.SetPTPCommonParams(api, *velocityRatio*, *accelerationRatio*, isQueued=0),
- dType.SetPTPCmd(api, *ptpMode*, *x*, *y*, *z*, *rHead*, isQueued=0),
- dType.GetPose(api),
- dType.dSleep(*ms*).

Příkaz dType.SetPTPJumpParams(api, *jumpHeight*, *zLimit*, isQueued=0) slouží k nastavení parametrů pro pohyb skokem. Parametr ‚api' zůstává vždy tak, jak je. Parametr ‚*jumpHeight*' určuje, jak vysoko se rameno pohne při vykonávání pohybu skokem. Výška je určena souřadnicí Z nejvyššího bodu trajektorie plus hodnotou tohoto parametru. ‚*zLimit*' určuje maximální hodnotu osy Z, do které se rameno během pohybu může dostat. Parametrem ‚isQueued' se nastaví, zda má příkaz vyčkat na vykonání předchozího příkazu ve frontě.

V programu se vykoná příkaz dType.SetPTPJumpParams(api, 10, 200, 0) tak, jak je znázorněno na obrázku 6.6.



Obrázek 6.6 – Parametry skoku

dType.SetPTPCommonParams(api, *velocityRatio, accelerationRatio,* isQueued=0) určuje rychlost a zrychlení pohybů ramene. Parametr ‚api' opět zůstává, ‚*velocityRatio*' udává maximální rychlost pohybu a ‚*accelerationRatio*' určuje maximální zrychlení ramene. U obou těchto parametrů lze nastavit hodnotu 0 až 100 procent. Pro maximální rychlost i zrychlení robotu může příkaz vypadat takto: dType.SetPTPCommonParams(api, 100, 100, 0).

Důležitým příkazem pro pohyb robotu je příkaz dType.SetPTPCmd(api, *ptpMode, x, y, z, rHead,* isQueued=0). Tento příkaz určí druh a koncovou pozici pohybu ramene. Do parametru ‚*ptpMode*' lze vepsat číslo od 0 do 9. Toto číslo určuje druh pohybu, které rameno vykoná. Možné pohyby jsou:

0 – Pohyb MOVJ se skokem v absolutních kartézských souřadnicích.

1 – Pohyb MOVJ v absolutních kartézských souřadnicích.

2 – Pohyb MOVL v absolutních kartézských souřadnicích.

3 – Pohyb MOVJ se skokem v absolutních úhlových souřadnicích kloubů.

4 – Pohyb MOVJ v absolutních úhlových souřadnicích kloubů.

5 – Pohyb MOVL v absolutních úhlových souřadnicích kloubů.

6 – Pohyb MOVJ v relativních kartézských souřadnicích.

7 – Pohyb MOVL v relativních kartézských souřadnicích.

8 – Pohyb MOVJ v relativních úhlových souřadnicích kloubů.

9 – Pohyb MOVL v relativních úhlových souřadnicích kloubů.

Parametry ‚*x*', ‚*y*', ‚*z*' a ‚*rHead*' udávají souřadnice koncového bodu, pokud je zvolen pohyb v absolutních souřadnicích, nebo o kolik se má rameno v dané ose posunout při použití souřadnic relativních. Pokud jsou využity pro pohyb kartézské souřadnice, tyto parametry

nabývají hodnot dle souřadnic X, Y, Z a R v pravé horní části DobotStudia. V opačném případě tyto parametry nabývají hodnot J1, J2, J3 a J4 podle hodnot v pravé části DobotStudia, hned pod souřadnicemi kartézskými.

Příkazem dType.GetPose(*api*) lze získat aktuální polohu ramene. Tento příkaz vrací hodnoty v poli se všemi potřebnými souřadnicemi – [x, y, z, r, J1, J2, J3, J4]. Toto pole je možné uložit do proměnné – *souradnice* = dType.GetPose(api). Následně jde přistoupit k jednotlivým hodnotám přes tuto proměnnou – *souradnice*[*hodnota*]. Parametr ‚*hodnota*' určuje, kterou souřadnici zvolit:

0 – Souřadnice x,
1 – Souřadnice y,
2 – Souřadnice z,
3 – Souřadnice r,
4 – Souřadnice J1,
5 – Souřadnice J2,
6 – Souřadnice J3,
7 – Souřadnice J4.

Poslední příkaz, který byl v této úloze zapotřebí, je dType.dSleep(*ms*). Tento příkaz pozastaví běh programu na dobu uvedenou v parametru ‚*ms*'. Zde lze zapsat dobu v milisekundách. Pozor! Pokud se příkaz dType.dSleep(*ms*) vykoná zatímco se robotické rameno pohybuje do jiného bodu, vykonávání programu se pozastaví, ale rameno bude dále pokračovat do cílového bodu. To je dáno tím, že příkaz pro pohyb do tohoto bodu byl vykonán před pozastavením programu a robot tedy již dostal informaci o pohybu do cílového bodu.

Na konci úlohy je vytvořen program pro opakované čtení pozice ramene během pohybu. Program zjistí souřadnice, kterými rameno projde během pohybů MOVJ, MOVL a JUMP. Souřadnice jsou následně zapsány do souboru .csv (comma separated values). V protokolu se sestrojí graf naměřených trajektorií. V závěru se porovnají jednotlivé dráhy pohybů.

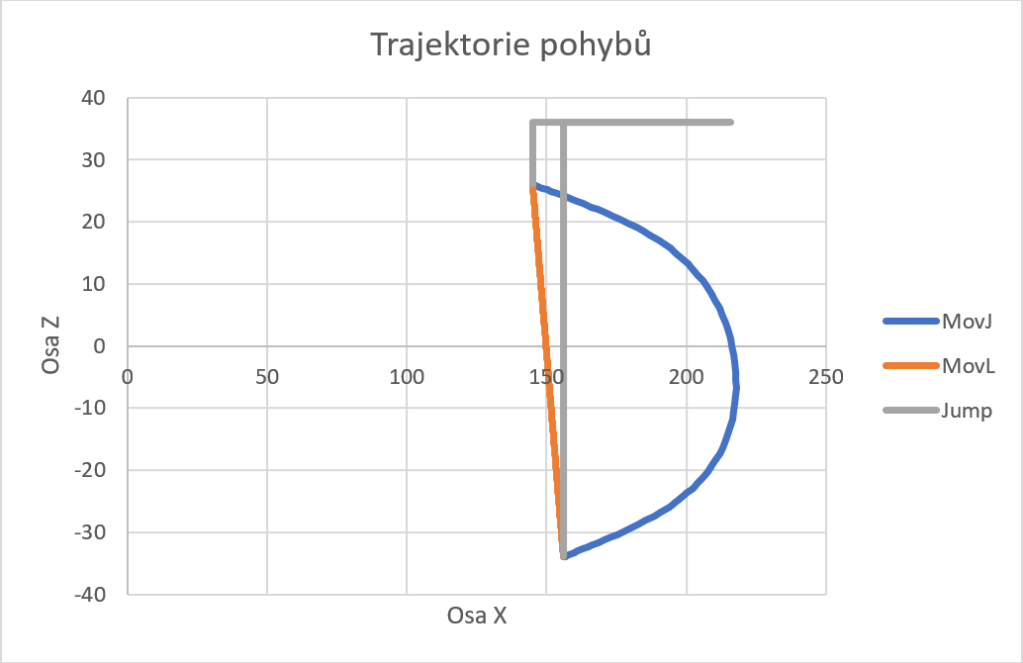
6.3 DOPORUČENÉ PŘÍKAZY

- while(*podmínka*),
- str(*číslo*),
- *soubor* = open(*cesta_k_souboru*, *režim_otevření*),
- *soubor*.close(),
- *soubor*.write(*text*),

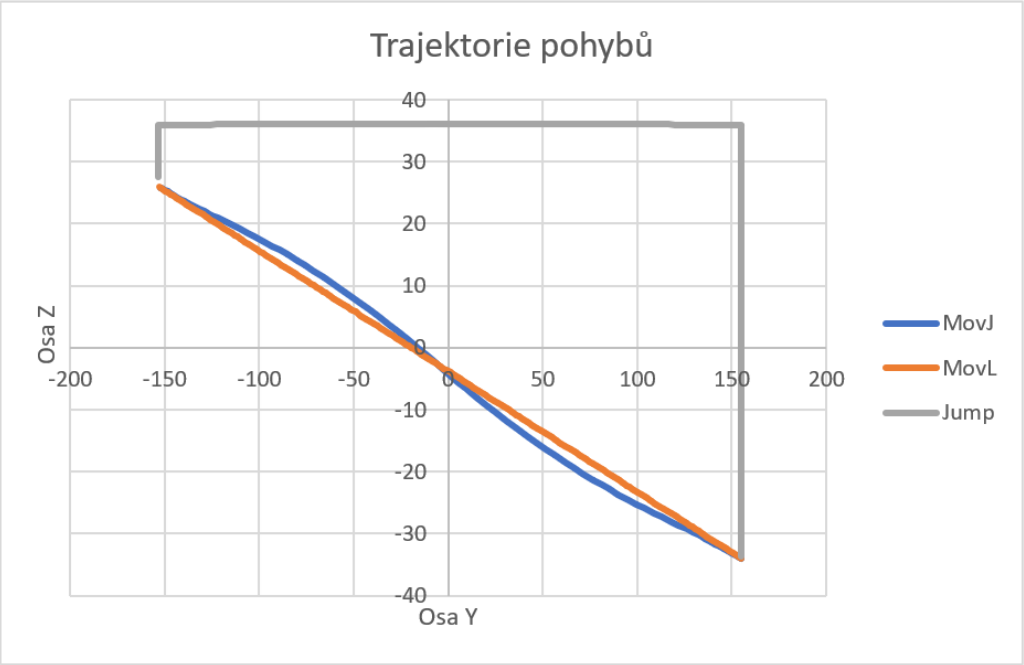
- dType.SetPTPJumpParams(api, *jumpHeight*, *zLimit*, isQueued=0),
- dType.SetPTPCommonParams(api, *velocityRatio*, *accelerationRatio*, isQueued=0),
- dType.SetPTPCmd(api, *ptpMode*, *x*, *y*, *z*, *rHead*, isQueued=0),
- dType.GetPose(api),
- dType.dSleep(*ms*).

6.4 VÝSLEDKY ÚLOHY

Měřením se zjistily rozdíly mezi pohyby MovJ, MovL a Jump. Zatímco pohyb MovL se pohybuje po lineární trase, pohyb MovJ si trasu stanovuje tak, aby byla energeticky co nejméně náročná. Při pohybu MovJ se rameno snaží mít po celou dobu pohybu zapnut co nejmenší počet motorů a tím trajektorie pohybu dostává obloukovitý tvar. To samé platí i pro pohyb Jump, zde však robot vyjede do požadované výšky a oblouk se tedy tvoří jen v osách X a Y. Pohyb MovJ je díky svému přesunu nejrychlejším pohybem. Pohyb MovL neustále přepíná jednotlivé motory, aby docílil lineární trajektorie. Kvůli tomu se jedná o nejpomalejší pohyb ze seznamu.



Obrázek 6.7 – Trajektorie pohybů při pohledu z boku



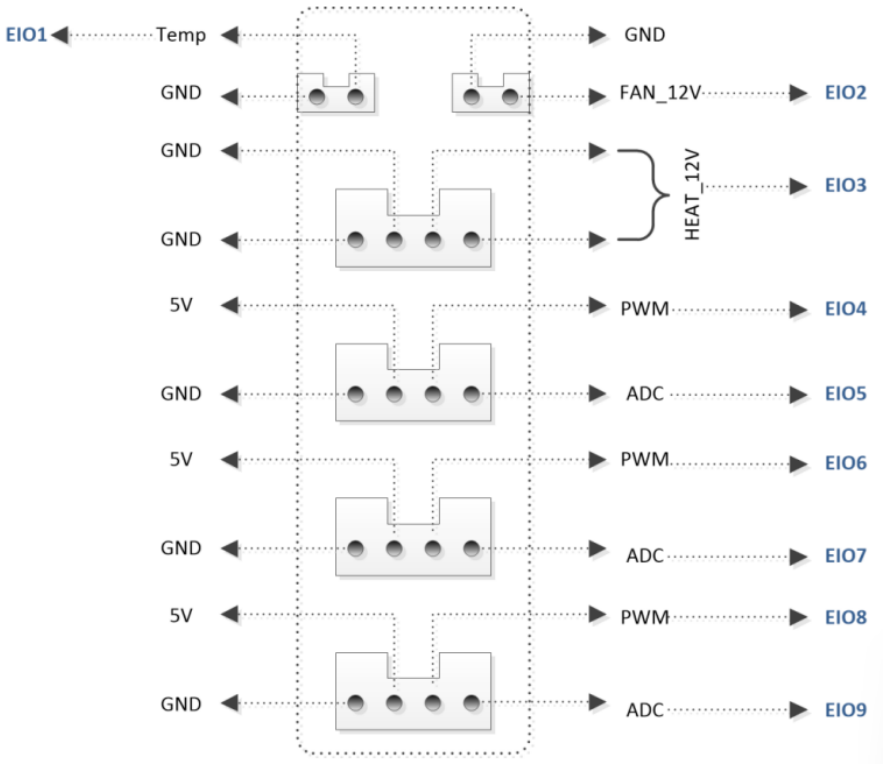
Obrázek 6.8 – Trajektorie pohybů při pohledu zpředu

7 LABORATORNÍ ÚLOHA SENZOR BAREV

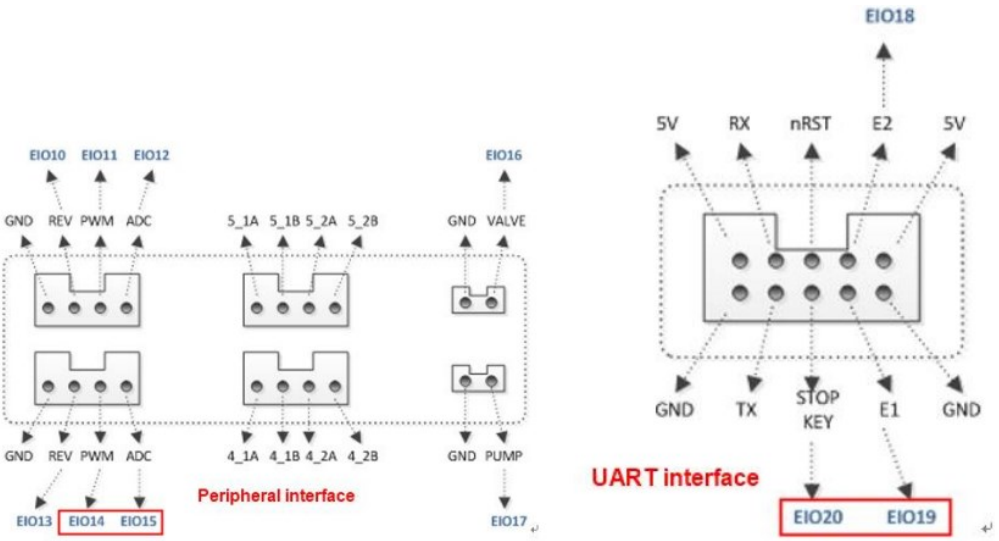
Tato úloha se zabývá ovládáním vstupů a výstupů robotického ramene Dobot Magician. Studenti připojí a ověří funkčnost senzoru barev, který bude použit v další úloze.

7.1 ÚVOD

Většina aplikací, kam se průmyslové roboty nasazují, vyžadují speciální zařízení, se kterým dokáže rameno komunikovat. Ať už se jedná o obyčejné pneumatické uchopovače nebo technologicky náročné 3D vidění, je nutné zajistit propojení s řídicí jednotkou robotu. Jedním z možných řešení je využití vstupně/výstupních pinů (také označovány jako DIO – Digital Input/Output nebo GPIO – General Purpouse Input/Output). Dobot Magician disponuje dvaceti víceúčelovými piny – devět přímo na předním rameni robotu (viz obrázek 7.1) a jedenácti na zadní straně základny (viz obrázek 7.2).

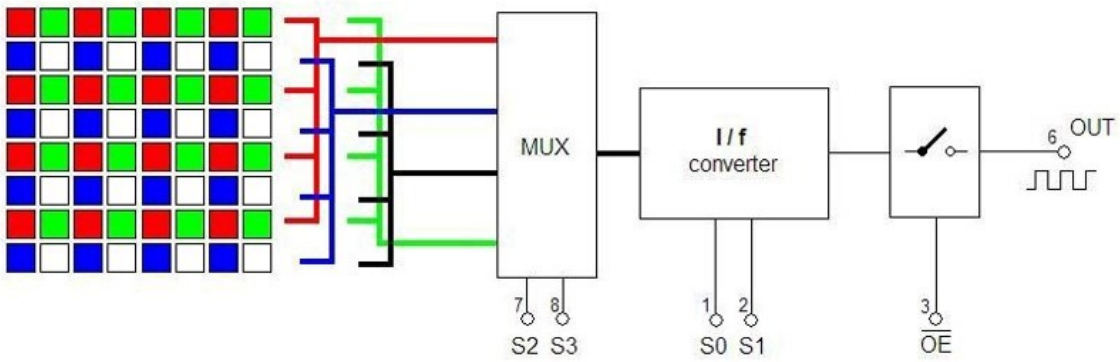


Obrázek 7.1 – I/O piny na předním rameni robotu (Dobot Magician, 2022)



Obrázek 7.2 – I/O piny na základně robotu (Dobot Magician User Guide, 2022)

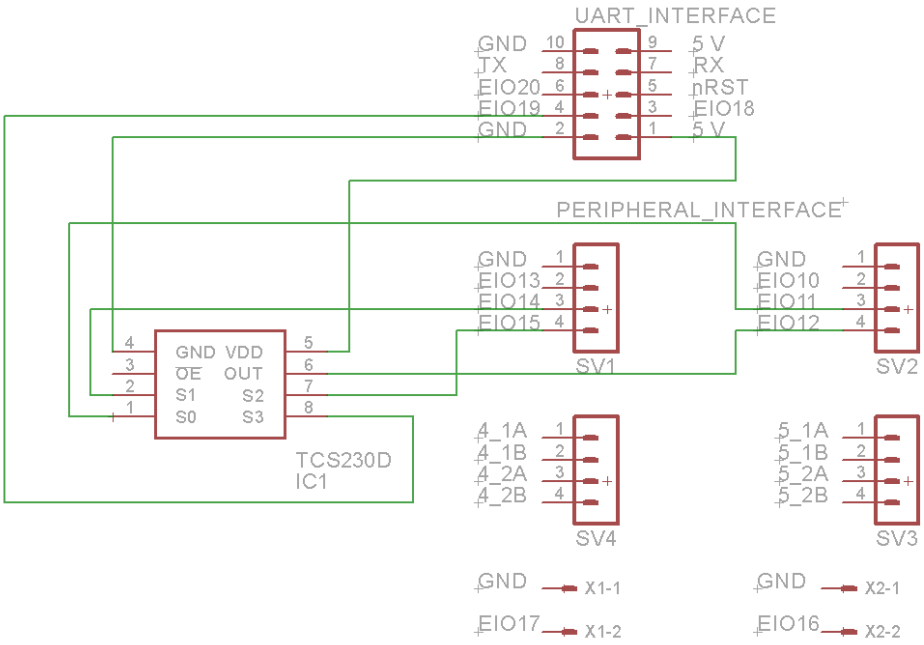
V této laboratorní úloze je využito těchto pinů pro připojení senzoru barev TCS230. Na desce senzoru se nachází čtyři bílé led k nasvícení snímaného objektu. Světlo z těchto diod se od tohoto objektu odrazí do pole 64 fotodiod umístěných uprostřed senzoru. Tyto fotodiody změří hodnotu odraženého světla a vrátí tuto informaci v podobě impulzu logické jedničky. Aby byl senzor schopen rozpoznat základní složky barevného spektra – červenou, zelenou a modrou barvu (RGB) – je každá z 64 fotodiod vybavena barevným filtrem (viz obrázek 7.3). Senzor je ovládán pěti digitálními vstupy. Pin OE slouží k aktivaci senzoru. Pro spuštění senzoru musí být OE nastaven do logické 0. Piny S0 a S1 slouží k omezení výstupní frekvence. Piny S2 a S3 vybíráme, které fotodiody měří odražené světlo. Na pinu OUT lze následně číst hodnotu ze senzoru (viz blokové schéma na obrázku 7.3).



Obrázek 7.3 – Blokové schéma senzoru TCS230 (Arduino France, 2021)

7.2 POSTUP

Senzor barev TCS230 je připojen podle zapojení na obrázku 7.4. Pro zprovoznění senzoru jsou potřebovat dva napájecí piny (5 voltů a zem), čtyři ovládací piny a jeden pin, ze kterého budou čtena data. K napájení jsou využity dva piny nacházející se na rozhraní UART. Levý vrchní je připojen k pinu označenému jako VCC a levý dolní pin se připojí na GND. Dále jsou propojeny ovládací vstupy senzoru S0 a S1 s piny EIO11 a EIO14. Piny S2 a S3 propojte s výstupy EIO15 a EIO19 robotu. Nakonec je zapojen výstup senzoru OUT na EIO12.



Obrázek 7.2 – Schéma zapojení senzoru

Aby bylo možné ovládat jednotlivé vstupy/výstupy robotu, je nutné nejdříve nastavit, jakou požadovaný pin vykonává funkci. Toho lze jednoduše docílit použitím příkazu dType.SetIOMultiplexing(api, adresa, *funkce*, 0). Do parametru *adresa* se napíše číslo pinu, který je potřeba nastavit. Parametr *funkce* udává, v jakém režimu daný pin funguje:

0 – Dummy,
1 – OUTPUT,
2 – PWM,
3 – INPUT,
4 – AD.

Pokud by bylo potřeba nastavit pin EIO20 na digitální vstup, příkaz by vypadal takto: dType.SetIOMultiplexing(api, 20, 3, 0)

Pro účely úlohy byl vytvořen nový program a nastaveny piny, ke kterým jsou připojeny vstupy senzoru S0, S1, S2 a S3 jako výstupní. Pin, ke kterému je připojen výstup detektoru barev OUT, byl nastaven jako vstup.

Ke správnému měření je zapotřebí nastavit omezení výstupní frekvence piny S0 a S1. Logická hodnota těchto pinů se zvolí s pomocí datasheetu tak, aby frekvenční omezení bylo 120 kHz. K tomu je využit příkaz dType.SetIODO(api, *adresa*, *log. úroveň*, 0). Parametr *adresa* opět představuje číslo požadovaného pinu, místo parametru *log. úroveň* se napíše 1 nebo 0 podle toho, jaký stav výstupu je potřeba.

Pro zjištění barvy využívá senzor fotodiody s barevnými filtry. Pomocí pinů S2 a S3 lze zvolit, která složka barvy (červená, modrá nebo zelená) se měří. Pro zjištění barevného spektra předmětu je zapotřebí změřit všechny tři složky barvy. Opět byl využit příkaz dType.SetIODO(api, *adresa*, *log. úroveň*, 0).

Pin OUT slouží ke čtení výstupní hodnoty senzoru. Snímač disponuje převodníkem proudu na frekvenci, který převádí údaje fotodiod na čtvercovou vlnu s frekvencí, která je úměrná intenzitě světla zvolené barvy. Pro účely tohoto měření je ovšem jednodušší měřit periodu signálu. Čtení aktuálního stavu pinu OUT zajišťuje příkaz dType.GetIODI(api, *adresa*). Parametru adresa opět značí číslo pinu, ze kterého se má logická úroveň přečíst. Příkaz vrací hodnotu 1, pokud je na vstup přivedeno napětí odpovídající logické 1, v opačném případě vrací příkaz hodnotu 0. Tuto hodnotu je možné přiřadit k vytvořené proměnné – *jmenoPromenne* = dType.GetIODI(api, *adresa*).

Jelikož je potřeba změřit periodu všech tří barevných složek, bude se určitá část v programu opakovat. K zjednodušení a zpřehlednění kódu lze využít funkci. Funkce představuje jednu nebo více řádkovou část kódu, která se může z programu opakovaně volat. Místo použití předdefinovaných funkcí je vytvořena pomocí příkazu *def* funkce vlastní. V úloze je příklad funkce s názvem MojeFunkce (viz obrázek 7.5). Nepřijímá žádné argumenty a třikrát vypíše "Test". Příkazy ve funkci se provedou pouze tehdy, když je funkce zavolána. Funkce vždy musí být definována před tím, než se v programu volá!

```
1 def MojeFunkce():      # Definice funkce
2     print("Test")      #
3     print("Test")      # Tělo funkce
4     print("Test")      #
5
6 MojeFunkce()           # Zavolání funkce
```

Obrázek 7.3 – Vytvoření a použití funkce

Pokud je potřeba ve funkci použít proměnou vytvořenou mimo tuto funkci, je nutné upravit její definici tak, jak je znázorněno na obrázku 7.6.

```
1 def VetsiNezPet(cislo):                                   # Definice funkce
2     if(cislo > 5):                                        #
3         print("Cislo " + cislo + " je vetsi nez pet.")    # Tělo funkce
4     else:                                                 #
5         print("Cislo " + cislo + " neni vetsi nez pet.")  #
6
7 a = 3
8 VetsiNezPet(a)                                            # Zavolání funkce
9 VetsiNezPet(7)                                            # Zavolání funkce
```

Obrázek 7.4 – Funkce s proměnnou

Funkce může vracet hodnoty zpět do hlavního těla programu. K tomu je potřeba do funkce zapsat příkaz return *hodnota*, který přiřadí hodnotu k proměnné v programu. Ukázkový příklad s využitím všech výše zmíněných příkazů je na obrázku 7.7.

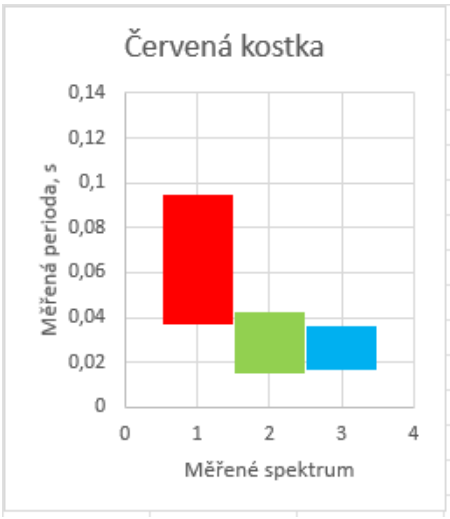
```
1  def VetsiHodnota(a, b):
2      if(a > b):
3          return a
4      else:
5          return b
6
7  prvniCislo = 5
8  druheCislo = 7
9
10 vetsiCislo = VetsiHodnota(prvniCislo, druheCislo)
11 print(vetsiCislo)
12
13 print(VetsiHodnota(12, 9))
```

Obrázek 7.5 – Funkce se vstupními i výstupními parametry

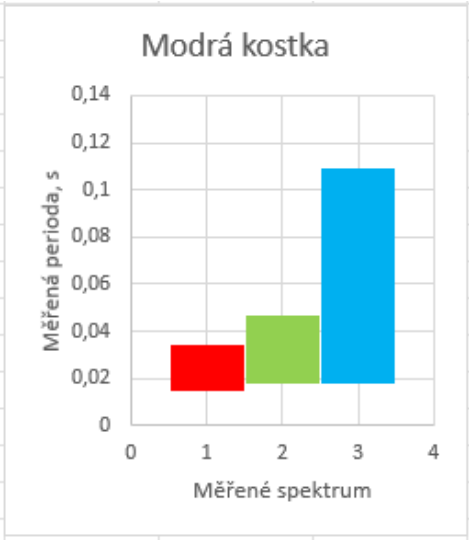
Zadáním úlohy je vytvořit program, který určí rozsahy složek RGB barevného spektra tří kostek – červené, zelené a modré. K měření je využit senzor barev TCS230 připojeného k základně robotického ramene Dobot Magician. Součástí úlohy je vytvoření funkce pro měření periody vstupního signálu. Ta je uložena do proměnné. Změřena je postupně červená, zelená a modrá složka snímaného objektu a naměřené hodnoty jsou uloženy do csv souboru. Měření se opakuje minimálně desetkrát pro každý měřený objekt.

7.3 VÝSLEDKY ÚLOHY

Z výsledných grafů je patrné, že barva s největší hodnotou odraženého světla je shodná s barvou snímaného objektu. Ostatní barevné složky jsou značně utlumené. Tento výsledek je nejvíce patrný u hodnot modré kostky. Z měření lze vypozorovat, že detekce barvy pomocí odraženého světla je značně ovlivněna vnějšími světelnými podmínkami. Ideální měření je nutné provádět v co nejtemnějším prostředí, jinak dochází k výchylkám od skutečné hodnoty. Výsledky měření také závisí na vzdálenosti objektu od senzoru. Měřený objekt by měl být vzdálený přibližně jeden centimetr. Pokud je objekt vzdálený o více než centimetr, na fotodiody dopadá více okolního osvětlení a tím přináší do měření nezanedbatelnou výchylku.



Obrázek 7.6 – Barevné složky červené kostky



Obrázek 7.7 – Barevné složky modré kostky

8 LABORATORNÍ ÚLOHA SIMULACE VÝROBNÍ LINKY

Tato lekce studenty seznámí s externím příslušenstvím robotického ramene Dobot Magician. Naučí se používat lineární pojezd a pásový dopravník s infračerveným senzorem vzdálenosti a detektorem barev.

8.1 ÚVOD

Lineární pojezdy pro polohování robotů jsou dlouhé dráhy používané ve skladech nebo v leteckém a automobilovém průmyslu, aby jeden robot mohl vykonávat více úloh. Tyto pohybové konstrukce, nazývané také robotické přenosové jednotky nebo RTU (robot-transfer units), jsou stále častěji používány pro montáž, svařování a skladování. Na rozdíl od typických sestav, v nichž je robot připevněn k podlaze, se robot s lineárním pojezdem dokáže volně přesouvat mezi pracovními stanicemi. V této úloze se ověří funkčnost ovládání lineárního pojezdu pro rameno Dobot Magician. Jeden metr dlouhý lineární pojezd významně rozšíří akční rádius robotu. To je ideální pro celou řadu aplikací – manipulace na větší vzdálenosti, psaní, kreslení nebo laserové gravírování velkých formátů. Nosnost pojezdu je 5 kg, maximální rychlost 150 $mm \cdot s^{-1}$ a přesnost opakování 0,01 mm.

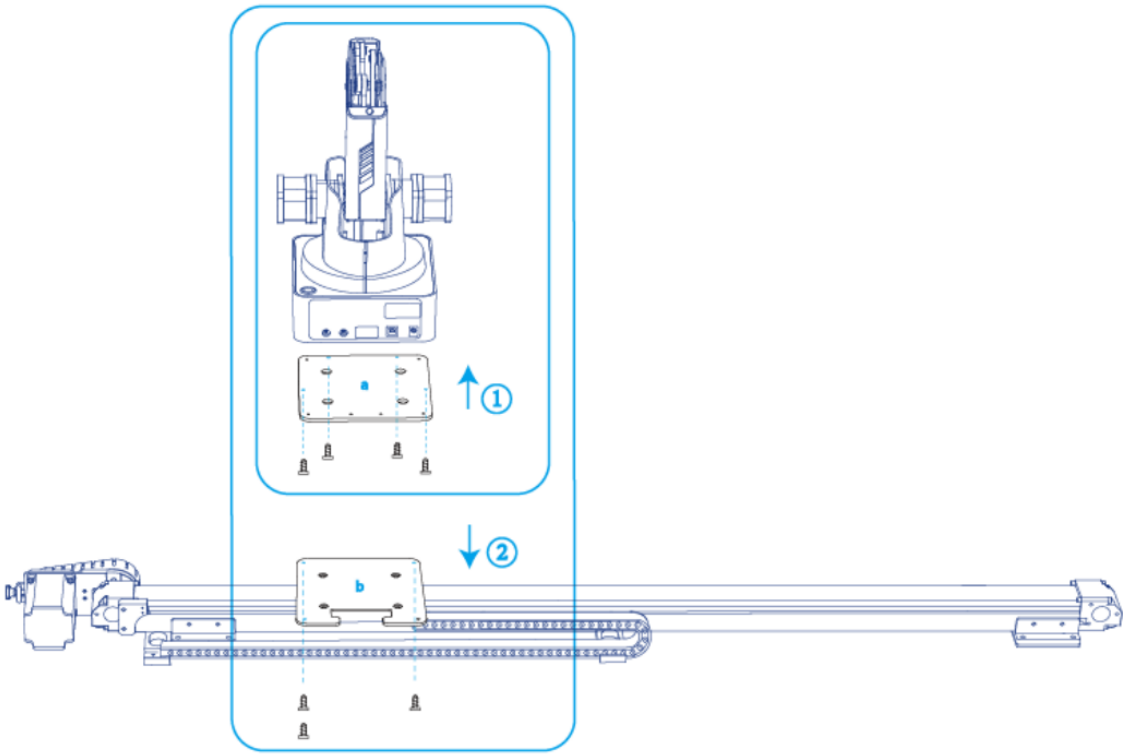
Obrázek 8.1 – Lineární pojezd (Dobot.cc, 2022)

Přestože existuje mnoho typů dopravníkových systémů, všechny slouží ke stejnému účelu – přepravě materiálů. Některé výrobky mohou vyžadovat systém bez pásu, který pro flexibilní pohyb používá pouze válečky nebo kolečka. Mnoho dopravníkových systémů se však spoléhá na rám s pásem a případné podpůrné válečky, které umožňují efektivní přepravu materiálů a výrobků. Sada dopravníku od firmy Dobot umožňuje vytvoření simulace výrobního

procesu. V sadě se kromě samotného dopravníku nachází i infračervený senzor vzdálenosti a detektor barev. Maximální rychlost dopravníku je 120 $mm \cdot s^{-1}$, nosnost 500 g.

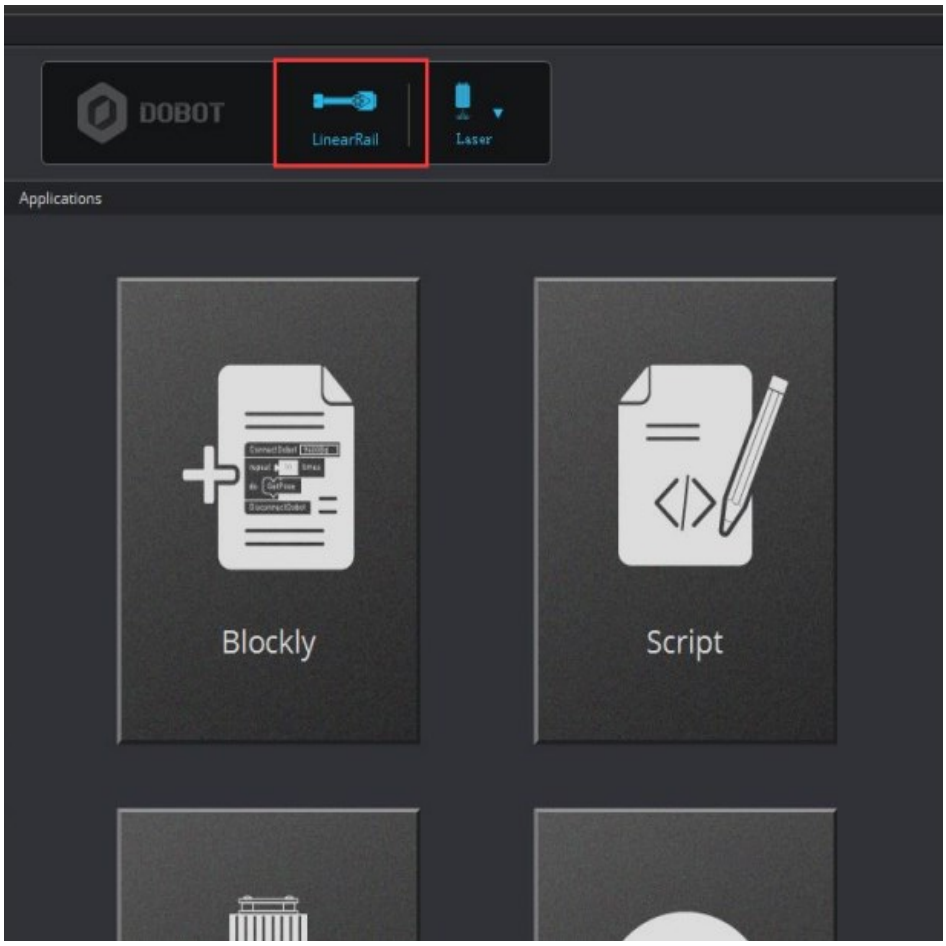
8.2 POSTUP

První část úlohy se zabývá propojením robotu s lineárním pojezdem. Robotické rameno je připevněno k pojezdu tak, jak je znázorněno na obrázku 8.2. Konektory jsou připojeny podle značení.



Obrázek 8.2 – Připevnění ramene k pojezdu (Sliding Rail User Guide, 2022)

Krokový motor pojezdu se připojí ke konektoru Stepper2 na zadní části základny robotu. Po spuštění DobotStudia lze povolit ovládání lineárního pojezdu v horní části programu (viz obrázek 8.3). Na kontrolním panelu se zpřístupní ovládání další osy – L. Tlačítka L+ a L- ovládají běh lineárního pojezdu. Po stisknutí tlačítka Home se kromě kalibrace robotu nyní provede i kalibrace pojezdu. Pojezd se přesune ke koncovému spínači na začátku svého rozsahu (souřadnice L = 0).



Obrázek 8.3 – Zapnutí ovládání lineárního pojezdu (DobotStudio, 2022)

K ovládání pojezdu v prostředí Script slouží příkazy, které se nachází v knihovně LinearRail. Pro správnou funkčnost lineárního pojezdu v programu je zapotřebí pojezd inicializovat příkazem dType.SetDeviceWithL(api, *jePojezdPripojen*, *verze*). Parametr *jePojezdPripojen* slouží k povolení pojezdu. Pokud je pojezd připojen, nastaví se parametr na hodnotu 1. V opačném případě se použije 0. Verze se nastaví podle datasheetu v balení. Podobně jako tomu bylo u nastavování parametrů robotu, i zde se doporučuje nastavit parametry rychlosti a zrychlení pojezdu. K tomu slouží příkaz dType.SetPTPLParams(api, *rychlost*, *zrychleni*, 0). Jak rychlost, tak zrychlení pojezdu je udáváno v procentech, a tedy je možné zvolit číslo od 0 do 100.

Pro pohyb robotického ramene s využitím lineárního pojezdu slouží příkaz dType.SetPTPWithLCmd(api, *ptpMode*, *x*, *y*, *z*, *rHead*, *l*, 0). Tento příkaz nahrazuje dříve používaný dType.SetPTPCmd(api, *ptpMode, x, y, z, rHead, 0*). Oproti němu přidává další osu pohybu L, kde lze nastavit souřadnici (vzdálenost), na kterou se pojezd přesune. Ostatní parametry jsou identické s předchozím příkazem.

V prostředí Script byl vytvořen program pro přesun kostek za pomoci lineárního dopravníku.

Druhá část úlohy se zaměří na sadu pásového dopravníku. Ten kromě samotného dopravníku obsahuje dvoustavový infračervený senzor vzdálenosti a detektor rozpoznání barev. Motor pásového dopravníku se připojí k portu Stepper1, který se nachází na základně robotu. Senzor barev je připojen na port GP2 a infračervený senzor vzdálenosti na přední rameno robotu, port GP4.

Pro ovládání pásového dopravníku se použije příkaz dType.SetEMotor(api, *index*, *isEnabled*, *speed*, 0). Parametr index určuje, ke kterému portu je krokový motor dopravníku připojen. Pokud je připojen ke konektoru STEPPER1, je index roven 0, pokud k STEPPER2, je roven 1. Pomocí isEnabled dojde k zapínání nebo vypínání ovládání motoru. Poslední parametr nastavuje rychlost dopravníku. Tu je však nutné nejdříve spočítat pomocí rovnice 8.1,

$$speed = v \cdot \frac{n}{o} \tag{8.1}$$

kde: *speed* je rychlost dopravníku v $mm \cdot s^{-1}$,
v je požadovaná rychlost v $mm \cdot s^{-1}$,
n je počet kroků na jednu otáčku,
O je obvod motoru v mm.

Použitý motor potřebuje 32000 kroků na jednu otočku – n = 32000. Průměr motoru je roven 36 mm, z toho vyplývá obvod motoru 113,1 mm. Pokud je tedy potřeba spustit motor dopravníku s rychlostí 50 $mm \cdot s^{-1}$, nastaví se parametr *speed* na hodnotu 14 146,77. Je-li potřeba spustit motor v opačném směru, stačí před spočtenou hodnotu připsat znaménko mínus. Pro zastavení motoru se za parametr *speed* dosadí 0.

Infračervený senzor vzdálenosti dokáže detekovat předměty, které se před ním nacházejí. Jelikož se jedná o dvoustavový senzor, vrací hodnotu 1 pokud se před senzorem nachází objekt, v opačném případě vrací hodnotu 0. Senzor je nejdříve potřeba inicializovat. K tomu slouží příkaz dType.SetInfraredSensor(api, *isEnable*, *infraredPort*, *version*). Parametr *isEnabled* určuje, zda má být senzor zapnut (hodnota 1) nebo vypnut (hodnota 0). Parametrem *infraredPort* vyberete konektor, ke kterému je senzor připojen:

- GP1 – 0,
- GP2 – 1,
- GP4 – 2,
- GP5 – 3.

K zjištění hodnoty senzoru slouží příkaz dType.GetInfraredSensor(api, *infraredPort*). Port je nastaven podle konektoru, ke kterému je senzor připojen (viz seznam výše).

Senzor barev, stejně jako tomu bylo u infračerveného senzoru, je potřeba nejdříve inicializovat. Inicializace se provede příkazem dType.SetColorSensor(api, *isEnable*, *colorPort*, *version*). Povolení senzoru se nastavuje parametrem *isEnabled*. Pro použití senzoru se nastaví tento parametr na hodnotu 1. Nastavením hodnoty 0 v tomto parametru se senzor vypne. Parametr colorPort určuje, ke kterému konektoru je senzor připojen. Hodnota je nastavena stejným způsobem, jako tomu je u infračerveného senzoru (viz seznam výše). Verzi senzoru lze zjistit z přibaleného datasheetu.

Ke čtení hodnot ze senzoru slouží příkaz dType.GetColorSensor(api). Tento příkaz vrací pole tří prvků – červené, zelené a modré složky spektra. Ukázková část kódu, jak číst požadované hodnoty, je na obrázku 8.4.

```
1 dType.SetColorSensor(api, 1, 1, 1)
2
3 spektrumBarev = dType.GetColorSensor(api)
4
5 cervenaSlozka = spektrumBarev[0]
6 zelenaSlozka = spektrumBarev[1]
7 modraSlozka = spektrumBarev[2]
```

Obrázek 8.4 – Získání složek barvy snímaného objektu

Jelikož funkce detektoru barev již byla vysvětlena v kapitole 7, je možné použít zjednodušené čtení hodnoty. Senzor pro každou složku vrací pouze hodnotu 1 nebo 0, podle toho, zda danou barevnou složku detekuje v dostatečném množství. Odpadá proto potřeba kalibrace senzoru.

Pro splnění úlohy je vytvořen program, který vezme kostku a umístí ji na začátek dopravníku. Dopravník se spustí rychlostí 50 $mm \cdot s^{-1}$ a vyčká pět sekund. Následně je nastavena rychlost dopravníku tak, aby se kostka začala vracet. Dopravník běží, dokud kostku nezaznamená infračervený senzor. Po detekování senzorem se dopravník zastaví, robot uchopí kostku a zjistí její barvu. Nakonec jsou kostky roztříděny podle barvy vedle dopravníku.

9 LABORATORNÍ ÚLOHA HARDWAROVÉ OVLÁDÁNÍ

V této úloze studenti vytvoří teach pendant pro ovládání robotického ramene. K vytvoření použijí vývojovou desku Arduino Mega spolu s tlačítky, led a ovládacím joystickem.

9.1 ÚVOD

Vývojová deska Arduino představuje malý řídicí mikroprocesor vybavený řadou vstupních a výstupních pinů. K desce je možné připojit celou řadu senzorů, od jednoduchých tlačítek a koncových spínačů, až po ultrazvukové senzory vzdálenosti nebo optické senzory kvality ovzduší. Mikroprocesor dokáže data z těchto senzorů zpracovat a díky výstupním periferiím, jako jsou světelné indikace nebo LCD displeje, informovat uživatele o jejich stavu. Platforma Arduino se stala poměrně populární mezi lidmi, kteří s elektronikou teprve začínají. Na rozdíl od většiny předchozích programovatelných obvodů nepotřebuje Arduino k nahrání nového kódu na desku samostatný kus hardwaru (tzv. programátor) - stačí použít kabel USB. Kromě toho prostředí Arduino IDE používá zjednodušenou verzi jazyka C++, což usnadňuje učení programování.

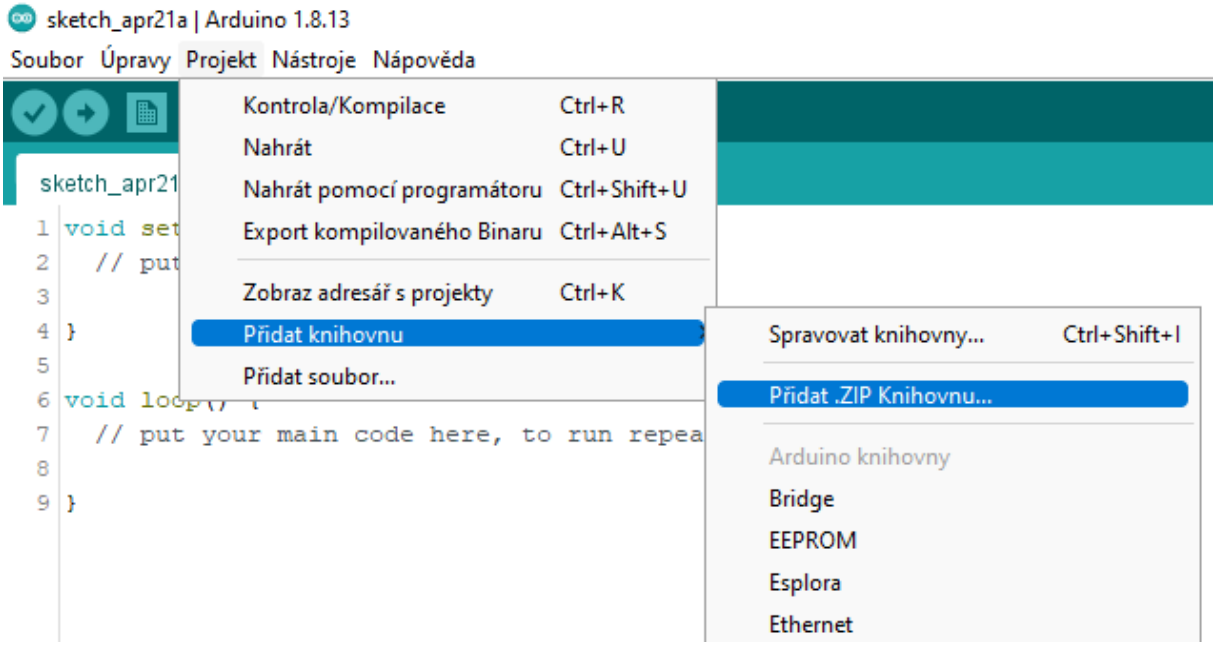
Robotické teach pendanty jsou ruční zařízení, která mohou být spojena s řídicí jednotkou robotu buď kabely, nebo bezdrátově. Teach pendanty jsou obvykle součástí řídicího systému při nákupu průmyslového robota. Obvykle obsahují několik tlačítek a přepínačů nebo jsou vybaveny dotykovým displejem, jak je tomu u novějších typů robotů. Na displeji lze procházet programy robotu a umožňuje jejich editaci. Kromě toho lze displej použít k vyvolání historie příkazů robota. Teach pendanty využívají klávesnici pro zadávání úloh a snadné programování. Kromě programování a ovládání robotů lze panely používat k testování a odstraňování případných problémů v programu. Možnost testovat robotické systémy umožňuje snadnější integraci do výrobního procesu a tím zlepšuje efektivitu robotu.

9.2 POSTUP

K ovládání ramene Dobot Magician je použita vývojová deska Arduino Mega 2560 spolu s rozšiřujícím modulem Dobot Shield. Tento modul lze jednoduše nasadit na piny Arduina. Konektor označený jako Magician slouží pro komunikaci s robotickým ramenem. Jeden konec vícežilového kabelu je připojen k Arduinu. Druhý konec je připojen ke komunikačnímu portu Magiciana, který se nachází v zadní části základny robotu. Dále se

k robotu připojí napájení. USB kabel do základny robotu se nezapojuje, namísto toho je připojen k vývojové desce Arduino.

K programování Arduina lze využít zadarmo dostupný vývojový software Arduino IDE, který je ke stažení na oficiálních stránkách výrobce. Aby bylo možné používat příkazy k ovládání ramene, je zapotřebí nejdříve přidat knihovnu Magician, která se nachází ve složce se zadáním. Knihovna se do programu přidá v horní části Arduino IDE – položka Projekt, Přidat knihovnu, Přidat .ZIP Knihovnu (viz obrázek 9.1). V otevřeném průzkumníku lze vybrat soubor Magician.h.



Obrázek 9.1 – Přidání knihovny (Arduino IDE, 2022)

K ověření funkčnosti je vytvořen jednoduchý program k navázání komunikace s robotem. Na úplný začátek programu je přidán příkaz #include <Magician.h>. Příkazy ve smyčce void Setup se provedou pouze jednou při spuštění, zatímco příkazy ve smyčce void Loop se vykonávají opakovaně až do zastavení programu. Do smyčky Setup se napíše příkaz Dobot_Init(). Tento příkaz otevře komunikaci s robotem a nastaví potřebné parametry. Program je znázorněn na obrázku 9.2. Pokud se program bez problému do Arduina nahraje, je možné pokračovat k dalšímu kroku.

Po navázání komunikace je potřeba nastavit základní parametry pohybů robotu. K tomu slouží příkaz Dobot_SetPTPCommonParams(float *rychlost*, float *zrychleni*) pro nastavení

```
#include<Magician.h>

void setup() {
  // put your setup code here, to run once:
  Dobot_Init();
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Obrázek 9.2 – Program pro otevření komunikace s robotem

vlastností pohybů typu PTP a Dobot_SetJOGCommonParams(float *rychlost*, float *zrychleni*) pro pohyby typu JOG. Parametry obou příkazů představují procentuální hodnoty maximální rychlosti a zrychlení. Pod příkazem Dobot_Init() se nastaví parametry pro pohyby PTP tak, aby rychlost byla 50 % a zrychlení 100 %. Pohybu v režimu PTP je možné docílit příkazem Dobot_SetPTPCmd(*druhPohybu*, *x*, *y*, *z*, *r*). Jako *druhPohybu* je možné zvolit jeden z deseti různých druhů pohybu, jako tomu je v jazyce Python. Možné hodnoty této proměnné jsou:

- JUMP_XYZ – Pohyb MOVJ se skokem v absolutních kartézských souřadnicích.
- MOVJ_XYZ – Pohyb MOVJ v absolutních kartézských souřadnicích.
- MOVL_XYZ – Pohyb MOVL v absolutních kartézských souřadnicích.
- JUMP_ANGLE – Pohyb MOVJ se skokem v absolutních úhlových souřadnicích kloubů.
- MOVJ_ANGLE – Pohyb MOVJ v absolutních úhlových souřadnicích kloubů.
- MOVL_ANGLE – Pohyb MOVL v absolutních úhlových souřadnicích kloubů.
- MOVJ_INC – Pohyb MOVJ v relativních úhlových souřadnicích kloubů.
- MOVL_INC – Pohyb MOVL v relativních kartézských souřadnicích.
- MOVJ_XYZ_INC – Pohyb MOVJ v relativních kartézských souřadnicích kloubů.
- JUMP_MOVL_XYZ – Pohyb MOVL se skokem v relativních kartézských souřadnicích.

Za parametry *x*, *y*, *z* a *r* je možné dosadit souřadnice požadovaného koncového bodu, pokud se robot pohybuje v kartézských souřadnicích, případně úhly jednotlivých kloubů robotu, pokud jsou použity úhlové souřadnice kloubů.

Před zahájením pohybů ramene je třeba robot zkalibrovat, aby se pokaždé vracel do stejných pozic. Toho lze docílit příkazem Dobot_SetHOMECmd(). Tento příkaz, stejně jako

tomu bylo v Dobot Studiu, přesune rameno do krajní pozice a následně se vystředí do počáteční pozice.

Vytvořený program se doplní tak, aby se robot po inicializaci nejdříve zkalibroval, poté jsou nastaveny parametry pohybů typu PTP a rameno přesunuto na souřadnice $x = 250$, $y = 0$, $z = 50$ a $r = 0$ lineárním pohybem v absolutních kartézských souřadnicích. Ukázka programu je na obrázku 9.3.

```
#include<Magician.h>

void setup() {
  // put your setup code here, to run once:
  Dobot_Init();
  Dobot_SetHOMECmd();
  Dobot_SetPTPCommonParams(50, 100);
  Dobot_SetPTPCmd(MOVL_XYZ, 250, 0, 50, 0);
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Obrázek 9.3 – Upravený program pro pohnutí ramenem

Pokud je potřeba ramenem pohybovat kontinuálně po určitou dobu, spíš než na předem dané souřadnice, využije se příkazu Dobot_SetJOGCmd(*kloub*). Za proměnnou kloub je možné dosadit číslo, které určuje směr pohybu:

- 0 – Zastavení ramene,
- 1 – pohyb ve směru +X,
- 2 – pohyb ve směru -X,
- 3 – pohyb ve směru +Y,
- 4 – pohyb ve směru -Y,
- 5 – pohyb ve směru +Z,
- 6 – pohyb ve směru -Z,
- 7 – pohyb ve směru +R,
- 8 – pohyb ve směru -R.

Dále je program upraven tak, aby se po kalibraci robot půl vteřiny pohyboval nahoru (ve směru +Z) a následně se zastavil. K nastavení doby pohybu se použije příkaz delay(*ms*), který pozastaví na určenou dobu běh programu. Výsledný program je na obrázku 9.4. Důležité je přepsat příkaz pro nastavení parametrů pohybů na pohyby typu JOG.

```
#include<Magician.h>

void setup() {
  // put your setup code here, to run once:
  Dobot_Init();
  Dobot_SetHOMECmd();
  Dobot_SetJOGCommonParams(50, 100);
  Dobot_SetJOGCmd(5);
  delay(500);
  Dobot_SetJOGCmd(0);
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Obrázek 9.4 – Program s využitím pohybu typu JOG

V některých aplikací je potřeba zjistit aktuální polohu robotu. Toho lze docílit příkazem GetPose(*&pozice*). Jako parametr je zde odkaz na adresu proměnné, která je typu Pose. Inicializace této proměnné je stejná jako je tomu například u proměnných typu int, jen se zde zamění typ int za Pose. Aktuální souřadnice ramene jsou uloženy v této proměnné a lze k nim přistoupit pomocí znaku ‚.' jako tomu bývá u tříd.

- pozice.x – aktuální hodnota souřadnice X,
- pozice.y – aktuální hodnota souřadnice Y,
- pozice.z – aktuální hodnota souřadnice Z,
- pozice.rHead – aktuální hodnota souřadnice R.

Pro ukázku je vytvořen program, který po inicializaci a kalibraci robotu pohne ramenem na pozici X = 200, Y = -100, Z = 0, R = 0. Následně se rameno bude dvě vteřiny pohybovat ve směru +Y a každou čtvrtinu sekundy vypíše aktuální souřadnice uživateli pomocí sériové komunikace. K časování pohybu se použije funkce millis().

Pro ovládání uchopení nebo puštění objektů pomocí vakuové přísavky lze použít příkaz Dobot_SetEndEffectorSuctionCup(*stav*). Parametr *stav* je datového typu bool, tedy lze nastavit na hodnotu true pro zapnutí přísavky a false pro její vypnutí. Pro správnou funkci příkazu je potřeba připojit přísavku i pumpu podle nálepek na kabelech.

Cílem úlohy je vytvořit teach pendant pro ovládání robotického ramene Dobot Magician s využitím Arduina a Dobot Shieldem, joystickem, třemi tlačítky a třemi diodami LED. Teach pendant by měl fungovat takto:

- Pohybem joysticku v ose X se robot pohne daným směrem na ose Y,
- pohybem joysticku v ose Y se robot pohne daným směrem na ose X,
- zmáčknutím joysticku (osa Z) se přepne pohyb ramene z os X a Y na osy Z a R, opětovným stiskem se přepne pohyb ramene zpět,
- pohybem joysticku v ose X se robot pohne daným směrem na ose R,
- pohybem joysticku v ose Y se robot pohne daným směrem na ose Z,
- červené tlačítko slouží k ovládání přísavky – po prvním stisku se přísavka zapne, po druhém se vypne,
- použito je dvourozměrné pole pro ukládání pozic ramene. Do pole se uloží až deset různých bodů spolu s aktuálním stavem spuštění přísavky. Pozice do pole jsou vloženy ve formátu celého čísla int,
- stiskem modrého tlačítka se uloží aktuální pozice ramene do pole tak, aby nedošlo k přepsání předchozích zapamatovaných bodů,
- podržení modrého tlačítka déle, než jedna vteřina, začne ukládání příštích bodů od začátku pole a tím dojde k vytvoření nové posloupnosti bodů,
- stiskem zeleného tlačítka dojde k zopakování dříve uložených bodů od bodu prvního až po bod poslední (poslední bod nemusí být desátý, záleží, kolik bodů uživatel do pole zapsal),
- podržením zeleného tlačítka se robot zkalibruje.

10 LABORATORNÍ ÚLOHA SOFTWAROVÉ OVLÁDÁNÍ

V této úloze studenti vytvoří jednoduchý grafický ovládací panel pro rameno Magician. Ovládací panel naprogramují v prostředí Visual Studio v jazyce C#.

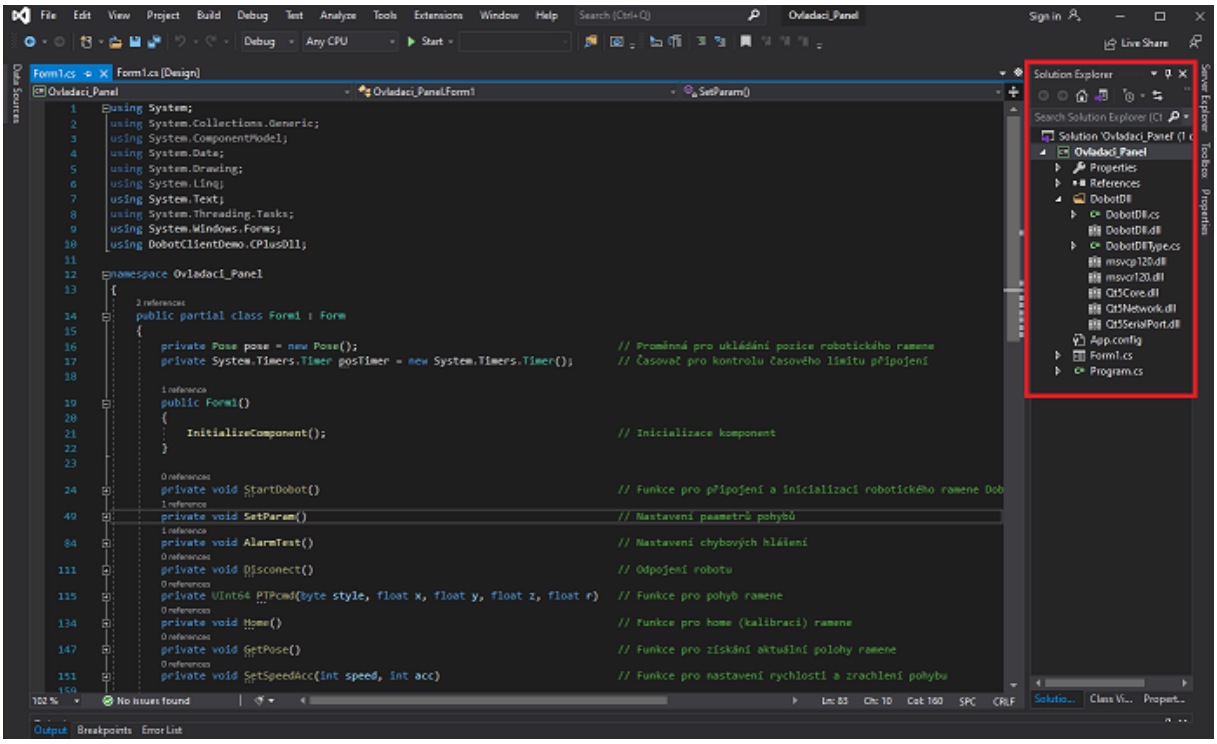
10.1 ÚVOD

Jazyk C# je univerzální, multiparadigmatický programovací jazyk. Jazyk C# navrhl Anders Hejlsberg ze společnosti Microsoft v roce 2000 a později byl v roce 2002 schválen jako mezinárodní standard organizací Ecma a v roce 2003 organizací ISO. Podle Ecma je C# jednoduchý, moderní, univerzální, objektově orientovaný programovací jazyk. Základní syntaxe jazyka C# je podobná syntaxi jiných jazyků , jako je jazyk C, C++ a zejména Java. K označení konce příkazu se používají středníky. Pro seskupování příkazů se používají množinové závorky. Příkazy se běžně seskupují do metod (funkcí), metody do tříd a třídy do jmenných prostorů. Proměnné se přiřazují pomocí znaménka rovnítka, ale porovnávají se pomocí dvou po sobě jdoucích znamének rovnítka. Hranaté závorky se používají u polí, a to jak k jejich deklaraci, tak k získání hodnoty na daném indexu v některém z nich.

Pro ovládání robotu z prostředí Visual Studia je zapotřebí využít souboru DLL (Dynamic Link Library). V operačních systémech Windows je většina funkcí operačního systému zajišťována knihovnami DLL. Navíc při spuštění programu v některém z těchto operačních systémů Windows může být velká část funkcí programu zajištěna těmito knihovnami. Například některé programy mohou obsahovat mnoho různých modulů a každý modul programu je obsažen a distribuován v knihovnách DLL. Jejich používání pomáhá podporovat modularizaci kódu, opakované použití kódu, efektivní využití paměti a snížení diskového prostoru. Operační systém a programy se tedy rychleji načítají, rychleji běží a zabírají méně místa na disku počítače.

10.2 POSTUP

V úloze je použit předpřipravený projekt Ovladaci_Panel.sln. Pro správnou funkčnost programu je nutné ověřit, zda se v průzkumníku řešení (Solution Explorer) nachází soubory DLL knihoven, viz obrázek 10.1. Tyto soubory zajišťují komunikaci s robotickým ramenem Dobot Magician. V tomto programu se odkazuje na funkce z programu DobotDll.cs, který pracuje se zmíněnými knihovnami.

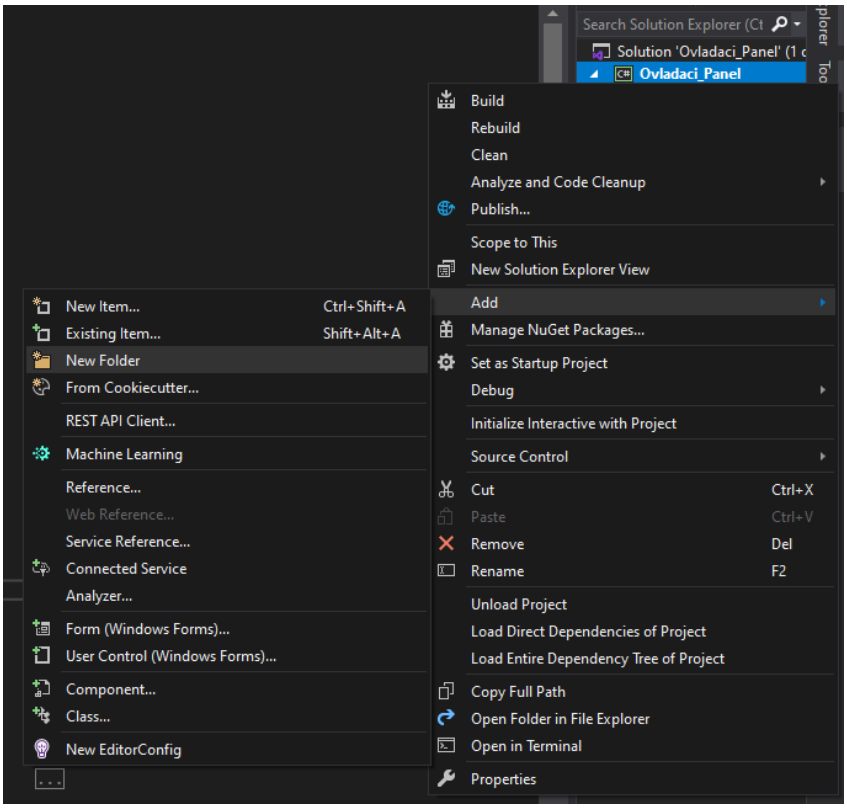


Obrázek 10.1 – Soubory DLL v průzkumníku řešení (Visual Studio 2019, 2022)

Pokud se zde soubory nenachází, je potřeba je přidat ručně. Kliknutím pravým tlačítkem myši na prvek Ovladaci_Panel v průzkumníku řešení se objeví nabídka. V té je vybrána položka Přidat (Add) a Nová složka (New Folder). Složka ponese název DobotDll.

Podobným způsobem se přidají do této složky soubory .dll. Kliknutím pravým tlačítkem myši na nově vytvořenou složku se v nabídce zvolí Přidat a Existující soubor (Existing Item). V otevřeném okně lze otevřít složku DobotDll ze zadání a vybrat všechny soubory v této složce. Výběr se potvrdí kliknutím na tlačítko Přidat (Add).

V programu se nachází osm předpřipravených funkcí. První z nich je funkce StartDobot(). Ta zajišťuje připojení k robotu a nastavení základních parametrů. Příkazy posTimer.Interval = 600 a posTimer.Start() nastavují časový limit připojení. Program automaticky najde komunikační port, na kterém se robot nachází, a připojí se k němu. Pokud je potřeba zadat komunikační port ručně, je možné ho zapsat do prázdných uvozovek v prvním příkazu. Druhý parametr v tomto příkazu udává rychlost komunikace v baudech. Podmínka hlídá, zda došlo k úspěšnému připojení. Pokud ne, funkce skončí. Příkazem DobotDll.SetCmdTimeout(3000) se nastaví časový limit příkazů. Následně se nastaví jméno zařízení a získá se jeho sériové číslo.



Obrázek 10.2 – Přidání nové složky do řešení (Visual Studio 2019, 2022)

V poslední části funkce dojde k vymazání fronty příkazů a spustí se vykonávání této fronty. Fronta se vyprázdní pomocí příkazu DobotDll.SetQueuedCmdClear() a spustí se funkcí DobotDll.SetQueuedCmdStartExec(). Nakonec se nastaví parametry pohybů díky SetParam() a odzkouší se chybová hlášení příkazem AlarmTest().

Další funkcí, která je v programu nutná, je Disconnect(). Tato jednoduchá funkce obsahuje pouze příkaz DobotDll.DisconnectDobot(), která zajistí správné odpojení robotu. Díky tomu je možné se k robotu připojit z jiného programu. Pokud se program ukončí bez odpojení, může zůstat komunikační port robotu obsazen a opětovné připojení k robotu bude možné až po jeho restartu.

Pro pohyb robotu je využita funkce PTPcmd(*style*, *x*, *y*, *z*, *r*). Podobně, jak tomu bylo s příkazem SetPTPcmd() v jazyce Python, i zde je možné nastavit jeden z deseti možných druhů pohybu. Ten se nastaví číslem 0 až 9 v parametru *style*. Dále lze nastavit bod v kartézské soustavě souřadnic do parametrů *x*, *y*, *z*, a *r*. Pokud je použit pohyb v kloubových úhlech, dosadí se do těchto parametrů dané hodnoty úhlů místo souřadnic. Pořadí v takovém případě je *J1*, *J2*, *J3*, *J4*.

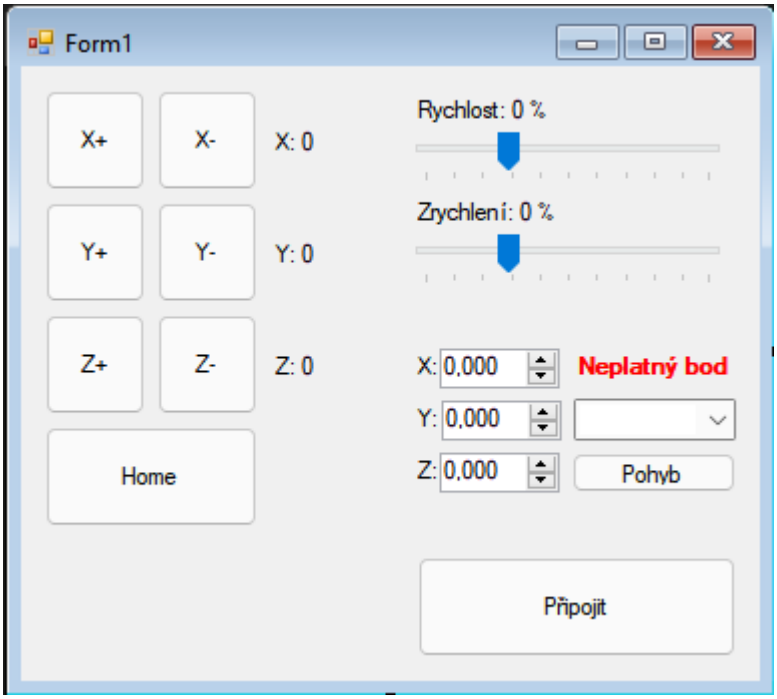
Příkazem Home() dojde ke kalibraci ramene. Stejně jako v prostředí DobotStudia po provedení příkazu pro Home se robotické rameno přesune na svůj koncový spínač a následně přejde zpět na bod uprostřed svého rozsahu.

GetPose() slouží pro získání hodnot souřadnic aktuální polohy ramene. Tato pozice se uloží do třídy pose, která je deklarovaná na začátku programu. Třída je popsána v programu DobotDllType.cs, který se nachází v průzkumníku řešení.

Pokud je zapotřebí zjistit konkrétní souřadnici bodu, zavolá se funkce GetPose(), která souřadnice uloží do této třídy, a následně lze přečíst tyto body za použití proměnné pose.*osa*. Za parametr *osa* je možné dosadit kteroukoliv osu nebo kloub tak, jak je uvedeno v její třídě.

Poslední příkaz, který se v této úloze použije k ovládání robotu, je SetSpeedAcc(*speed*, *acc*). Tento příkaz slouží k nastavení rychlosti a zrychlení pohybů robotu. Hodnoty rychlost (*speed*) a zrychlení (*acc*) se nastavují v procentech. Vlastní program je psán pod komentovaný řádek "Váš kód".

Na konci úlohy studenti vytvoří ovládací panel pro robotické rameno Dobot Magician. Ukázka panelu je na obrázku 10.3.



Obrázek 10.3 – Okno aplikace ovládacího panelu

Tlačítko Připojit slouží k připojení robotu k prostředí (funkce StartDobot()). Po připojení robotu se text na tlačítku změní na „Odpojit“. Po opětovném stisku tlačítka dojde k odpojení robotu (Disconnect())

Tlačítky X+, X-, Y+, Y-, Z+ a Z- dojde k posunutí ramene o 20 mm na dané ose a v daném směru. Použita je funkce PTPcmd v režimu MovL v relativních kartézských souřadnicích.

Texty (Label) X:, Y:, Z: vedle tlačítek slouží k informacím o aktuální poloze ramene. Aktuální polohu lze získat příkazem GetPose(). Ta bude volána v pravidelných intervalech 100 ms pomocí komponenty časovač (Timer) v nástrojích (Toolbox).

Tlačítko Home zajišťuje kalibraci ramene. Po stisku se vykoná příkaz Home().

Posuvníky (TrackBar) v pravé horní části okna se mění rychlost a zrychlení pohybů robotu. Rozsah posuvníku je 1 až 100, výchozí hodnota je 30. Hodnoty se nastaví po uvolnění tlačítka myši na posuvníku (event MouseUp). Nastavení parametrů je provedeno příkazem SetSpeedAcc().

Texty (Label) nad posuvníky informují uživatele o aktuálních hodnotách rychlosti a zrychlení. Hodnoty se aktualizují stejným způsobem, jako tomu je u polohy ramene – pomocí stejného časovače (Timer).

Numerická pole (NumericUpDown) pod posuvníky slouží k zadání bodu v kartézských souřadnicích. Číslo je zadáváno s přesností až 3 desetinná místa. Minimální hodnota je omezena na -300, maximální na 300.

Text (Label) „Neplatný bod“ slouží k informaci uživatele, zda je zadaný bod v numerických polích ramenem dosažitelný. Prostor, kam rameno dosáhne, se spočítá jako místo mezi dvěma koulemi se stejným středem v bodě X = 0, Y = 0, Z = 0 a poloměry 150 a 300 mm. Správnost bodu lze spočítat pomocí rovnice 10.1,

$$s = \sqrt{x^2 + y^2 + z^2} \tag{10.1}$$

kde: s je vzdálenost bodu v mm,
x je hodnota souřadnice X v mm,
y je hodnota souřadnice Y v mm,
z je hodnota souřadnice Z v mm.

Pokud je vzdálenost menší než poloměr větší koule a zároveň větší než poloměr koule menší, bod se nachází v dosahu robotu. Navíc je přidáno omezení X > 0 a zároveň Z > -85. Pokud bod leží v dosahu, přepíše se text na „Platný bod“ a barva se změní na zelenou. Pokud se bod v dosahu nenachází, bude obsahovat text „Neplatný bod“ v červené barvě. Kontrola bodu se provede po každé změně hodnoty v každém numerickém poli (NumericUpDown). K tomu je použit event ValueChanged v těchto polích.

Prvek ComboBox pod textem obsahuje dvě slova – MovJ a MovL, mezi kterými může uživatel vybírat.

Tlačítkem Pohyb se rameno pohne na souřadnice zadané v numerických polích. Pohyb je vykonán s rychlostí a zrychlením, které jsou dané hodnotami posuvníků. Způsob pohybu (kloubový nebo lineární) je určen podle vybraného prvku v ComboBoxu. Pohyb se provede pouze pokud je bod v dosahu ramene.

11 ZÁVĚR

Cílem této bakalářské práce bylo vytvoření sady laboratorních úloh pro robotické rameno Dobot Magician. Laboratorní úlohy byly navrženy pro studenty vysokých škol technického zaměření tak, aby vysvětlily základy programování robotických systémů. Vyzkoušeny byly na menších skupinách studentů na Univerzitě Pardubice. Délka vypracování jedné úlohy studentům zabrala v průměru hodinu a čtyřicet pět minut, což odpovídá dvěma vyučovacím hodinám.

Práce vznikala ve spolupráci s firmou Dobot a ControlTech. Tyto firmy zapůjčily vybavení potřebné ke zpracování a následnému odzkoušení úloh. Zapůjčeno bylo robotické rameno Dobot Magician, Arduino kit, lineární pojezd a pásový dopravník.

Úlohy již byly úspěšně vyzkoušeny s pomocí pana Otakara Pancnera ze střední průmyslové školy v Otrokovicích, kde byly využity k školení lektorů. Školení se skládalo ze čtyř celodenních výkladů problematiky ovládání a programování robotů. Poslední dva dny byly zaměřeny na praktické příklady, kde byly učitelům poskytnuty úlohy vytvořené v této bakalářské práci. Úlohy byly označené za „dobře napsané a podrobně vysvětlené“ a budou dále využity při přípravě výukových lekcí.

Tato práce neobsahuje postupy řešení pro ovládání robotického vidění kvůli náročnosti jeho použití. Jelikož je k programování systémů vidění nutná znalost teorie snímání obrazu a rozeznávání přítomných objektů, bylo robotické vidění firmy Dobot v této práci pouze zmíněno. Pro jeho vysvětlení by bylo zapotřebí samostatné sady laboratorních úloh, které by se zaměřily na vysvětlení základních pojmů a naučení logiky programování formou funkčních bloků v prostředí DobotVisionStudio.

POUŽITÁ LITERATURA

ARDUINO TINKERKIT BRACCIO, 2019. HW Kitchen [online]. Šenov [cit. 25. 4. 2022]. Dostupné z: https://www.hwkitchen.cz/arduino-tinkerkit-braccio-roboticka-ruka/

Balení Braccio Tinkerkit, 2020. Jameco [online]. Belmont [cit 24. 4. 2022]. Dostupné z: https://www.jameco.com/z/T050000-Arduino-TinkerKit-Braccio-Robot-Arm-Kit-Arduino-Board-not-Included-_2294071.html

BASOEKI, F. LIBERA, F. a MENEGATTI, E. 2013. Robots in Education: New Trends and Challenges from the Japanese Market. In: *Themes in Science and Technology Educatio*n vol. 6 (2013), No. 1, pp 51-62. [online]. Ioannina: The University of Ioannina [cit. 3. 4. 2022]. ISSN 1792-8788. Dostupné z: http://earthlab.uoi.gr/theste/index.php/theste/article/view/118

CONTROLTECH, 2022. *Dobot Magician – vlastnosti a ceny*. Kolín: Osobní sdělení

CRAIG, J. 2005. *Introduction to robotics: mechanics and control*. Londýn: Pearson Education. ISBN 0-13-123629-6.

Dobot Magician, 2021. *ControlTech* [online]. Kolín [cit 18. 4. 2022]. Dostupné z: http://dobotmagician.cz/magician.htm

DOBOT, 2020. *Dobot Magician User Manual v1.7.0* [online]. Šen-Čen [cit. 20. 4. 2022]. Dostupné z: https://download.dobot.cc/product-manual/dobot-magician/pdf/V1.7.0/en/Dobot-Magician-User-Guide-V1.7.0.pdf

ELEPHANT ROBOTICS, 2018. *myCobot User Manual.* [online] Šen-Čen [cit 27. 4. 2022]. Dostupné z: https://www.elephantrobotics.com/wp-content/uploads/2021/03/myCobot-User-Mannul-EN-V20210318.pdf

EMEDIA, 2019. *Educational robotics*. Projekt Erasmus+ č. 2018-1-FR01-KA201-048117.

LAFVIN Smart Robotické rameno, 2021. *Drátek* [online]. Havlíčkův Brod [cit. 25. 4. 2022]. Dostupné z: https://dratek.cz/arduino/34300-lafvin-smart-roboticke-rameno.html

MEAD, R.; THOMAS, S. a WEINBERG, J. 2012. *Robots in K-12 Education: A New Technology for Learning*. Hershey: Information Science Reference [cit. 19. 4. 2022]. ISBN 978-1-466-60184-0

MILLER, D.; NOURBAKHSH, I. a SIEGWART, R. 2008. Robots for education. In: *Springer Handbook of Robotics,* pp 1284. [online]. Springer International Publishing AG [cit. 18. 4. 2022]. ISBN 978-3-540-23957-4

MUBIN, O.; STEVENS, C. a SHADID, S. 2013. A review of the applicability of robots in education. In: *Technology for Education and Learning,* vol. 1 (2013), pp 1-7. [online]. Sydney: Western Sydney University [cit. 3. 4. 2022]. ISSN 1916-6990. Dostupné z: https://www.academia.edu/24957404/A_Review_of_the_Applicability_of_Robots_in_Education

MURPHY, R. 2019. *Introduction to AI Robotics*. Cambridge: MIT Press Ltd [cit. 23. 4. 2022]. ISBN 978-0-262-03848-5

MyCobot Pi, 2019. Botland [online]. Kunčice [cit. 27. 4. 2022]. Dostupné z: https://botland.cz/roboticke-paze/20224-mycobot-pi-6osy-robot-s-ramenem-verze-raspberry-pi-5904422350925.html

Robotický průmysl, 2018. Trade Media International [online]. Český Těšín [cit. 28. 4. 2022]. Dostupné z: https://www.konference-tmi.cz/aktuality/163-roboticky-prumysl-zaziva-silny-rust.html

SICILIAO, B.; KHATIB, O. a KRÖGER, T. 2016. *Springer handbook of robotics*. Springer International Publishing AG [cit. 16. 4. 2022]. ISBN 978-3-319-32550-7.

Tinkerkit Braccio robot, 2019. Arduino Store [online]. Turin [cit. 26. 4. 2022]. Dostupné z: https://store.arduino.cc/products/tinkerkit-braccio-robot

TinkerKit Braccio, 2020. RPishop [online]. Roudné [cit. 25. 4. 2022]. Dostupné z: https://rpishop.cz/roboti/3658-tinkerkit-braccio-robot.html

UIFlow, 2018. *M5stack* [online]. Šen-Čen [cit 27. 4. 2022]. Dostupné z: https://flow.m5stack.com

PŘÍLOHY

A – CD

Příloha A

Příloha k bakalářské práci

Laboratorní úlohy pro robotické rameno Dobot Magician

Tomáš Vladyka

CD

Obsah