STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

UNIVERZÁLNÍ ŘÍDICÍ SYSTÉM PRO OVLÁDÁNÍ VÝUKOVÝCH ROBOTICKÝCH MANIPULÁTORŮ

AUTOR	Martina Hanusová
ŠKOLA	Gymnázium a Střední průmyslová škola elektrotechniky a informatiky, Frenštát pod Radhoštěm, příspěvková organizace
ROČNÍK	Sekunda, obor osmileté gymnázium
KRAJ	Moravskoslezský
OBOR	10. Elektrotechnika, elektronika a telekomunikace

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

UNIVERZÁLNÍ ŘÍDICÍ SYSTÉM PRO OVLÁDÁNÍ VÝUKOVÝCH ROBOTICKÝCH MANIPULÁTORŮ

UNIVERSAL CONTROL SYSTEM FOR EDUCATIONAL ROBOTIC MANIPULATORS

AUTOR	Martina Hanusová
ŠKOLA	Gymnázium a Střední průmyslová škola elektrotechniky a informatiky, Frenštát pod Radhoštěm, příspěvková organizace
ROČNÍK	Sekunda, obor osmileté gymnázium
KRAJ	Moravskoslezský
ŠKOLITEL	Ing. Marcel Hanus, Ph.D.
OBOR	10. Elektrotechnika, elektronika a telekomunikace

Prohlášení

Prohlašuji, že svou práci na téma Univerzální řídicí systém pro ovládání výukových robotických manipulátorů jsem vypracovala samostatně pod vedením svého školitele a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Dále prohlašuji, že tištěná i elektronická verze práce SOČ jsou shodné a nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a změně některých zákonů (autorský zákon) v platném znění.

V Kunčicích pod Ondřejníkem dne 17. 3. 2018 Podpis:

Poděkování

Děkuji svému školiteli Ing. Marcelu Hanusovi, Ph.D. za obětavou pomoc, podnětné připomínky, nekonečnou trpělivost a za pomoc při nastavení registrů motorového driveru POWERSTEP01 a akcelerometru MPU6050. Dále děkuji svým učitelům: panu Ing. Liboru Otáhalíkovi za skvělý kroužek Arduino a za půjčení robotické ruky FRENGP a panu Mgr. Richardu Štěpánovi za skvělý kroužek robotiky. Nakonec děkuji autorům šablony SOČ Lucii Vaškeové, Jaroslavu Páralovi a Romanu Beránkovi, která mi usnadnila psaní této práce.

Anotace

Cílem této práce je vytvořit univerzální, dostupný a snadno obsluhovatelný řídicí systém pro ovládání výukových robotických manipulátorů se servomotory nebo bipolárními krokovými motory a dalšími pomocnými zařízeními, jako jsou například pásový dopravník s optickou závorou. V první kapitole jsou uvedeny příklady několika robotických manipulátorů, vytištěných na 3D tiskárně z návodů publikovaných na internetu, které je možno účelně použít pro výuku robotiky ve školách či zájmových kroužcích a které mohou být opatřeny řídicím systémem popsaným v této práci. Řídicí systém je tvořen vývojovou deskou DISCO-F746NG obsahující výkonný mikrokontrolér STM32F746NG a barevný dotykový displej s úhlopříčkou 4,3 palce. K pohonu bipolárních krokových motorů robotických manipulátorů slouží 4 motorové drivery PowerSTEP01. Všechny zbylé signály desky DISCO-F746NG jsou vyvedeny na svorkovnice, které umožňují připojení až 7 zařízení ovládaných přes PWM signály (servomotory, reproduktory, DC motory), až 128 dalších senzorů a mikrokontrolérů komunikujících přes I2C sběrnici nebo až 4 analogových senzorů. K řídicí desce je dále připojen Bluetooth modul HC-05, umožňující dálkové ovládání robotických zařízení z mobilního telefonu nebo tabletu. Práce obsahuje dokumentaci plošných spojů řídicí jednotky, její krabičky vytištěné na 3D tiskárně, software pro mikrokontrolér STM32F746NG a software pro mobilní aplikaci napsaný v programu Pocket Code pro OS Android, přizpůsobené k ovládání robotické ruky se čtyřmi bipolárními krokovými motory, používané k výuce na naší škole. Vývoj zařízení bude dále pokračovat ověřením funkce na robotických manipulátorech ovládaných servomotory, modelu vzájemné spolupráce dvou robotů a pásového dopravníku a moderních metod pokročilého senzorického ovládání manipulátorů.

Klíčová slova

robotické manipulátory, 3D tisk, mikrokontroléry, motorové drivery, mobilní aplikace

Annotation

The purpose of this project is to develop universal and available control system for educational robotic manipulators with servomotors or bipolar stepper motors and other auxiliary equipment, such as a conveyor belt with a photocell. The first chapter contains examples of selected robotic manipulators, published on the internet and printed by 3D printer, which can be very useful for teaching robotics in schools and hobby clubs. These manipulators can be equipped with control system described in this project. The control system is formed by a development board DISCO-F746NG containing a high performance microcontroller STM32F746NG and 4.3'' color touch display. Bipolar stepper motors of robotic manipulators are driven by 4 motor drivers PowerSTEP01. All spare signals of DISCO-F746NG are connected to the terminals, which allow to connect up to 7 devices controlled by PWM signals (servomotors, speakers, DC motors), up to 128 others sensors and microcontrollers communicating over I2C bus or up to 4 analog sensors. There is also a Bluetooth module HC-05 on the board, enabling robotic manipulators' remote control using a mobile phone or a tablet. This project contains documentation of the printed circuit boards, 3D printed mechanical designs of cases, software for microcontroller STM32F746NG and a mobile application, written in Pocket Code for OS Android, applied on an example of a robotic arm with 4 bipolar stepper motors, which is used for teaching robotics at our school. The development will continue with verification of the control unit with other robotic manipulators powered by servomotors, with model of two cooperating robots and a conveyor belt and with new methods of advanced sensor controls of the robotic manipulators.

Keywords

robotic manipulators, 3D print, microcontrollers, motor drivers, mobile application

Obsah

Úvod

Robotizace průmyslové výroby pokračuje rychlejším tempem, než jsme si mohli před několika lety představit. Robotické manipulátory postupně vytlačují lidskou pracovní sílu ve všech výrobních operacích, které vyžadují opakované činnosti, zejména při manipulaci s těžkými předměty, při rizikových činnostech nebo úkonech vyžadujících velkou manipulační přesnost a zvládají to mnohem rychleji, než by to byl schopen zvládnout člověk. Se zrychlujícím se tempem robotizace a snižováním výrobní ceny robotů lze předpokládat, že všechny výrobní profese provádějící opakované a jednoznačně definovatelné mechanické operace budou nahrazeny roboty.

Na druhou stranu je zde nedostatek kvalifikovaných pracovníků schopných programovat a udržovat tyto roboty a odborné školství bohužel v tomto stále zaostává za požadavky praxe. Proto je důležité, aby se studenti středních odborných škol mohli ve své výuce co nejpodrobněji seznámit s funkcemi a programováním různých typů robotických manipulátorů. Komerčně dostupné výukové manipulátory mající často pouze jednoúčelový řídicí systém jsou však velmi drahé a řadě škol chybí prostředky k jejich pořízení nebo se mohou obávat o jejich poškození při výuce a následně jejich nákladných oprav.

Proto jsem se rozhodla vyvinout vlastní jednoduchý a dostatečně univerzální řídicí systém, který je možno použít pro ovládání různých konstrukcí robotických manipulátorů, jejichž návody včetně celé výrobní dokumentace jsou k dispozici na internetu a lze je vytisknout na 3D tiskárně a sestavit s náklady několika stovek korun. Při takové cenové úrovni se školy nemusí bát nákladů z poškozených výukových modelů, protože díly jsou levné a lze je znovu vytisknout na 3D tiskárně.

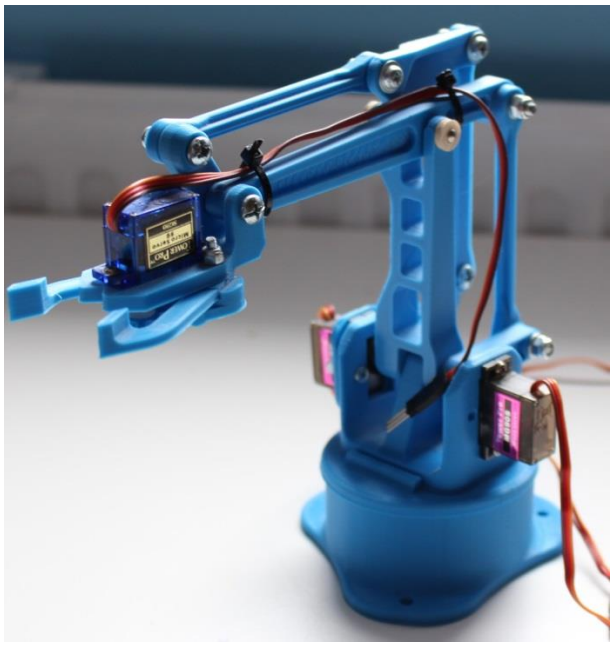
Kromě výuky robotických manipulátorů se studenti při využití tohoto návrhu budou mít možnost seznámit i s 3D tiskem, který je další moderní výrobní metodou a s programováním mikrokontrolérů ARM, které tvoří většinu všech používaných procesorů na světě. Řídicí systém by tedy měl být univerzální pro různé typy robotů, co nejlevnější, umožňující snadné uživatelské ovládání přes dotykové displeje nebo mobilní aplikace a nechat se programovat v dostatečně jednoduchém prostředí pochopitelném i středoškoláky. Veškerá dokumentace včetně schémat elektroniky, mechanické modely a software by měla být přístupná v podobě „open hardware“ a „open software“ k dispozici široké veřejnosti.

1 Příklady výukových robotických manipulátorů

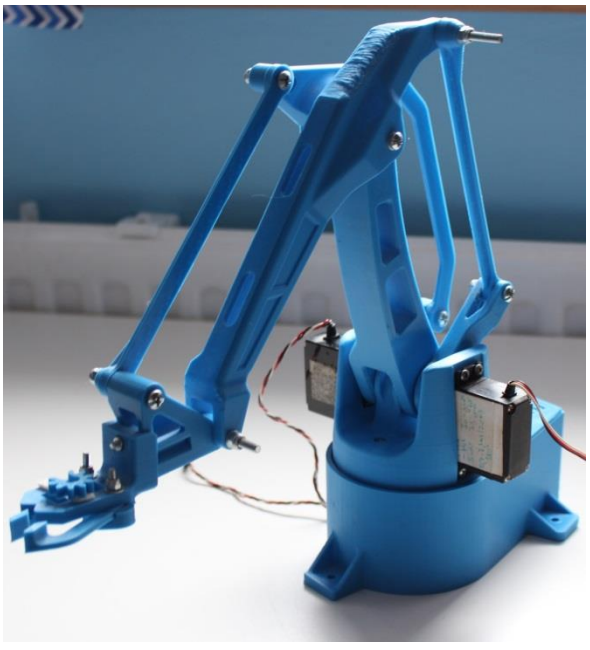
Na internetu jsou dostupné návody k výrobě stovek různých typů robotických manipulátorů pomocí 3D tisku, které je možno účelně použít k výuce robotiky na školách či v zájmových kroužcích. Mezi nejznámější stránky s použitelnými návody patří www.thingiverse.com, obsahující 3D modely pro tisk všech součástí robotů nebo server www.instructables.com, který k dokumentaci přidává podrobné návody vhodné i pro naprosté začátečníky. Jako příklad takovýchto robotických manipulátorů z internetu jsem si vybrala 4 roboty – EEZYbotARM, EEZYbotARM MK2, EEZYbotDELTA, ZORTRAX ROBOTIC ARM, které jsem vytiskla na 3D tiskárně Průša i3 MK2 z materiálu PLA a bez dalších úprav sestavila dle publikovaných návodů.

Malá robotická ruka EEZYbotARM od daGHIZmo[1][2] má otočnou základnu a dvojité rameno ovládané třemi servomotory MG90S a kleště pro manipulaci s drobnými předměty ovládané servomotorem SG90S. Hmotnost kompletního robota je pouze 192g a jeho celkové rozměry v x š x d jsou 19 cm x 9 cm x 18 cm.

Robotická ruka EEZYbotARM MK2 od stejného autora daGHIZmo[3][4] je zvětšenou verzí předchozího manipulátoru se delším dosahem a zvýšenou nosností pro manipulaci s těžšími předměty. Otáčecí základna obsahuje kuličkové ložisko pro snížení tření při otáčení. K pohonu slouží tři výkonnější servomotory MG995. K pohonu kleští se využívá menší servomotor SG90. Hmotnost robota je 581 g a rozměry v x š x d jsou 27 cm x 10 cm x 30 cm.



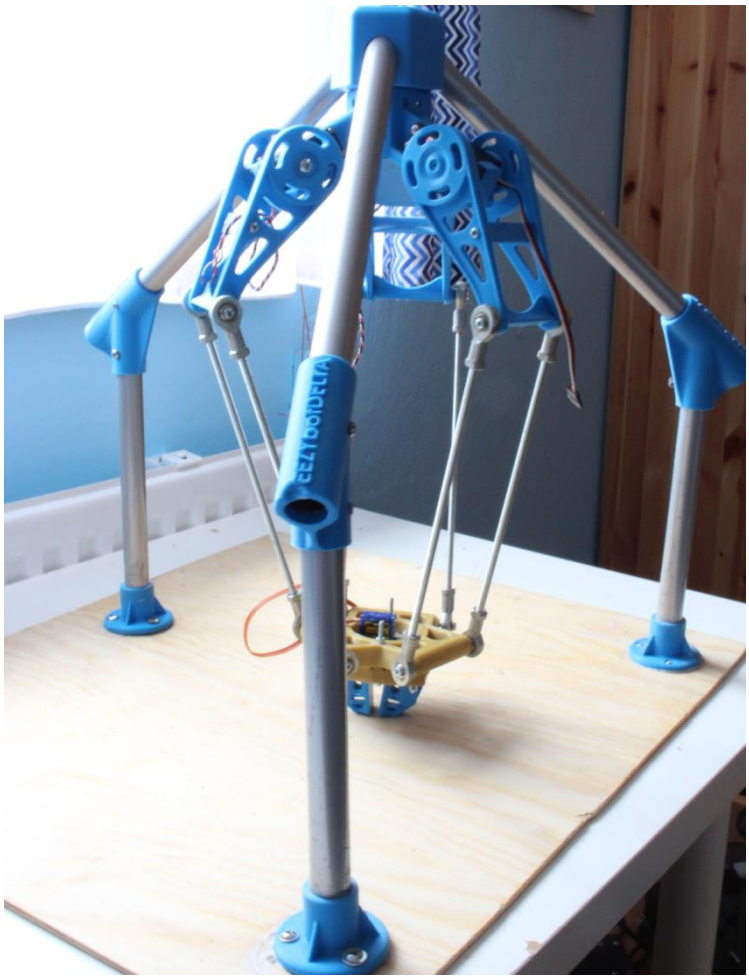
Obrázek 1: EEZYbotARM od daGHIZmo[1][2]



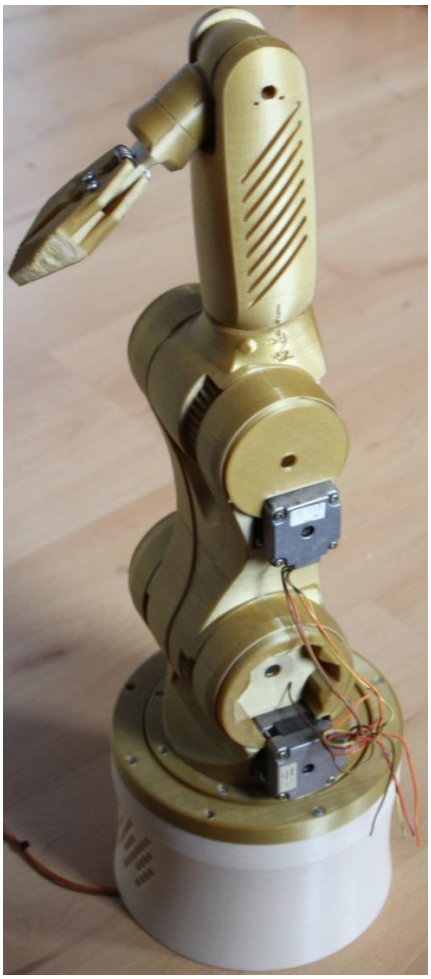
Obrázek 2: EEZYbotARM MK2 od daGHIZmo[3][4]

EEZYbotDELTA je robotická ruka tvaru delta stejného konstruktéra daGHIZmo[5][6] a představuje příklad robotického manipulátoru typu pick-and-place pro přesné polohování drobných předmětů. Manipulátor je opět poháněn třemi servomotory MG995 a k pohonu kleští slouží menší servomotor SG90. Hmotnost celého robota i s dřevotřískovou základnou je 1,8 kg a rozměry v x š x d jsou 43 cm x 33 cm x 44 cm.

Robotická ruka ZORTRAX ROBOTIC ARM[7][8] je prezentačním výrobkem výrobce 3D tiskáren ZORTRAX a využívá základnu s ložiskovými kuličkami, otočnou o 360°. K pohonu základny a dvou kloubových ramen slouží krokové motory NEMA17. Čelist robota není poháněna servomotorem, ale ovládá se mechanicky pomocí pružiny. Hmotnost sestaveného robota je 1,5 kg. Rozměry v x š x d jsou 55 cm x 15 cm x 27 cm.



Obrázek 3: EEZYbotDELTA od daGHIZmo[5][6]



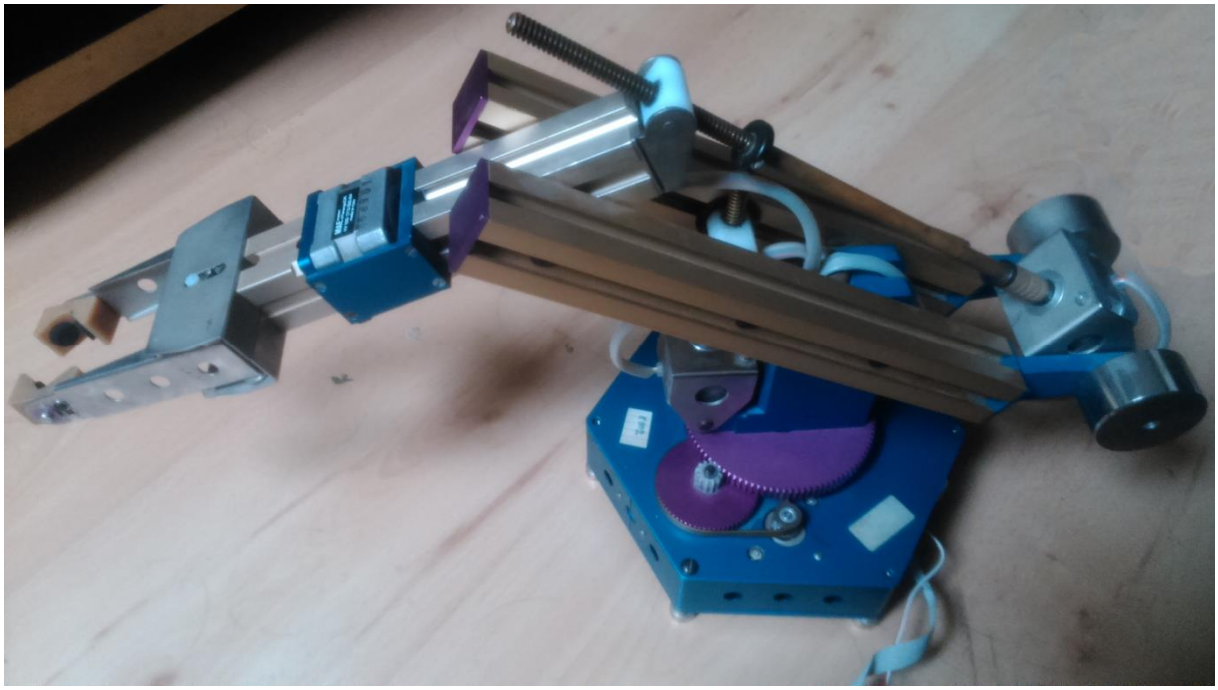
Obrázek 4: ZORTRAX ROBOTIC ARM[7][8]

Materiálové náklady na výrobu všech těchto robotických manipulátorů jsou velmi nízké, cena tiskových strun z PLA se pohybuje okolo 450 Kč/kg bez DPH, servomotory nebo malé krokové motory lze koupit od čínských internetových prodejců za cenu menší než 100 Kč za kus, celkově tak výrobní náklady většiny robotických manipulátorů s pohony nepřekračují 1000 Kč bez DPH.

K prvnímu ověření vyvíjeného univerzálního řídicího systému pro robotické manipulátory jsem si ale vybrala robotický manipulátor používaný k výuce na naší škole, který jsem si pracovně nazvala Robotická ruka FRENGP.

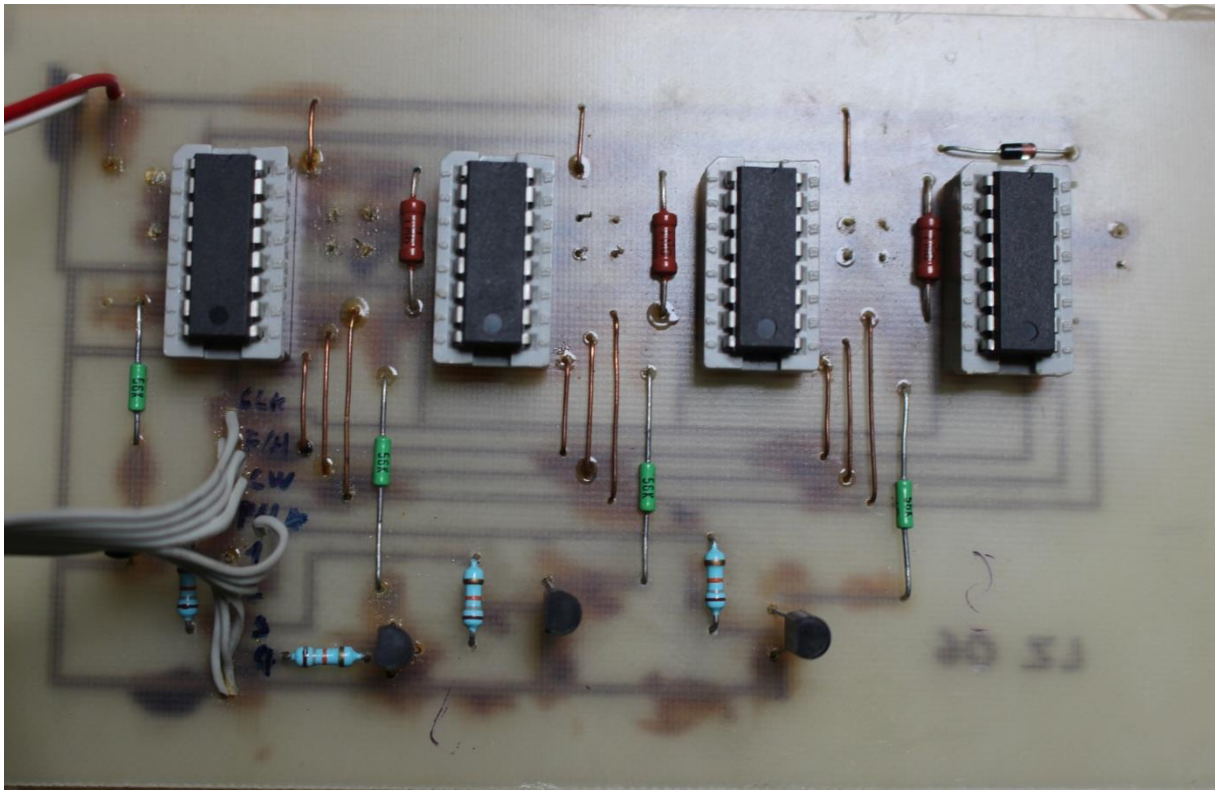
Robotická ruka FRENGP byla vyrobena nezjištěným výrobcem z hliníkových profilů a ozubených kol, s použitím 4 bipolárních krokových motorů typu HY100-1713-020-A4 italské firmy MAE motori offanengo, rozměru NEMA17, s 200 kroky na otáčku a odporem cívky 33 ohmů. Motory pohání rotaci základny robotické ruky (fialové ozubené kolo opatřené zarážkou proti překročení 360° rotace), dále páteřní dvojici hliníkových profilů, zvedenou rotací mosazné šroubovice s kyvným uložením motoru, ramenem zvedaným a sklápěným rovněž pohybem mosazné šroubovice a zakončeným kovovými kleštěmi, otevíranými a zavíranými stejným typem mechanismu.

Hmotnost robotické ruky FRENGP je 4,5 kg a její rozměry v x š x d jsou 30 cm x 18 cm x 50 cm.

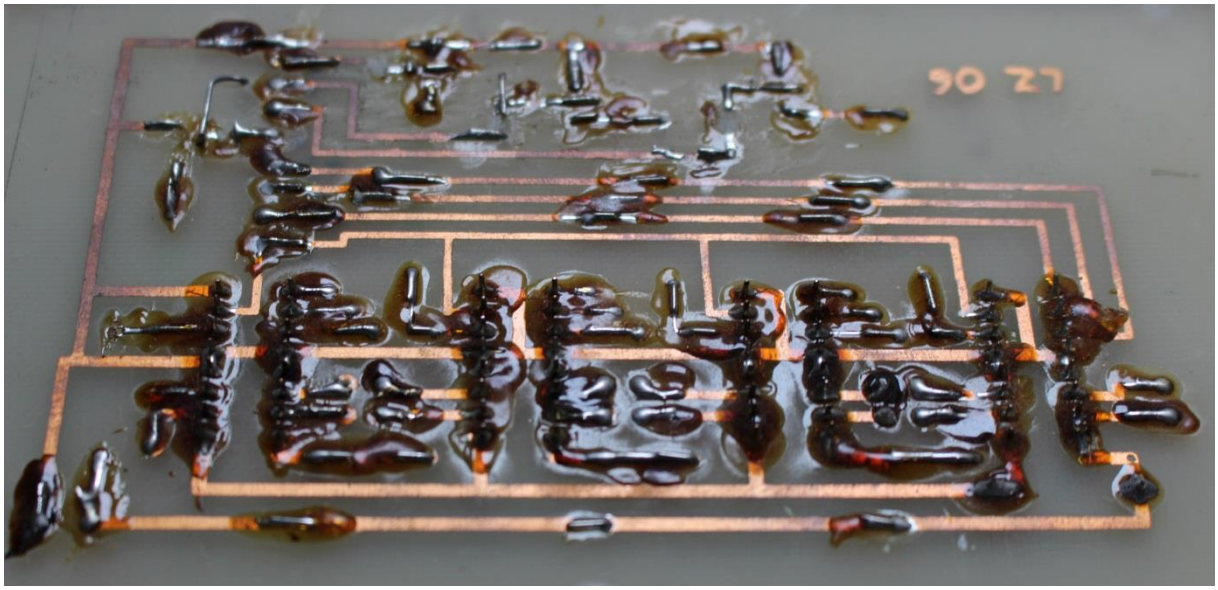


Obrázek 5: Foto robotické ruky FRENGP v původním stavu

K ovládání robotické ruky byla původně použita řídicí deska se 4 motorovými drivery MC3479 firmy ON Semiconductor [9], která byla ale kvůli přetržení slabých ovládacích drátků plochého multižilového kabelu nefunkční. Cílem řešení nového řídicího systému tak bylo i zvýšení mechanické odolnosti připojených kabelů.



Obrázek 6: Foto původního plošného spoje řízení robotické ruky FRENGP – strana součástek



Obrázek 7: Foto původního plošného spoje řízení robotické ruky FRENGP – strana spojů

2 Elektronický hardware řídicí jednotky

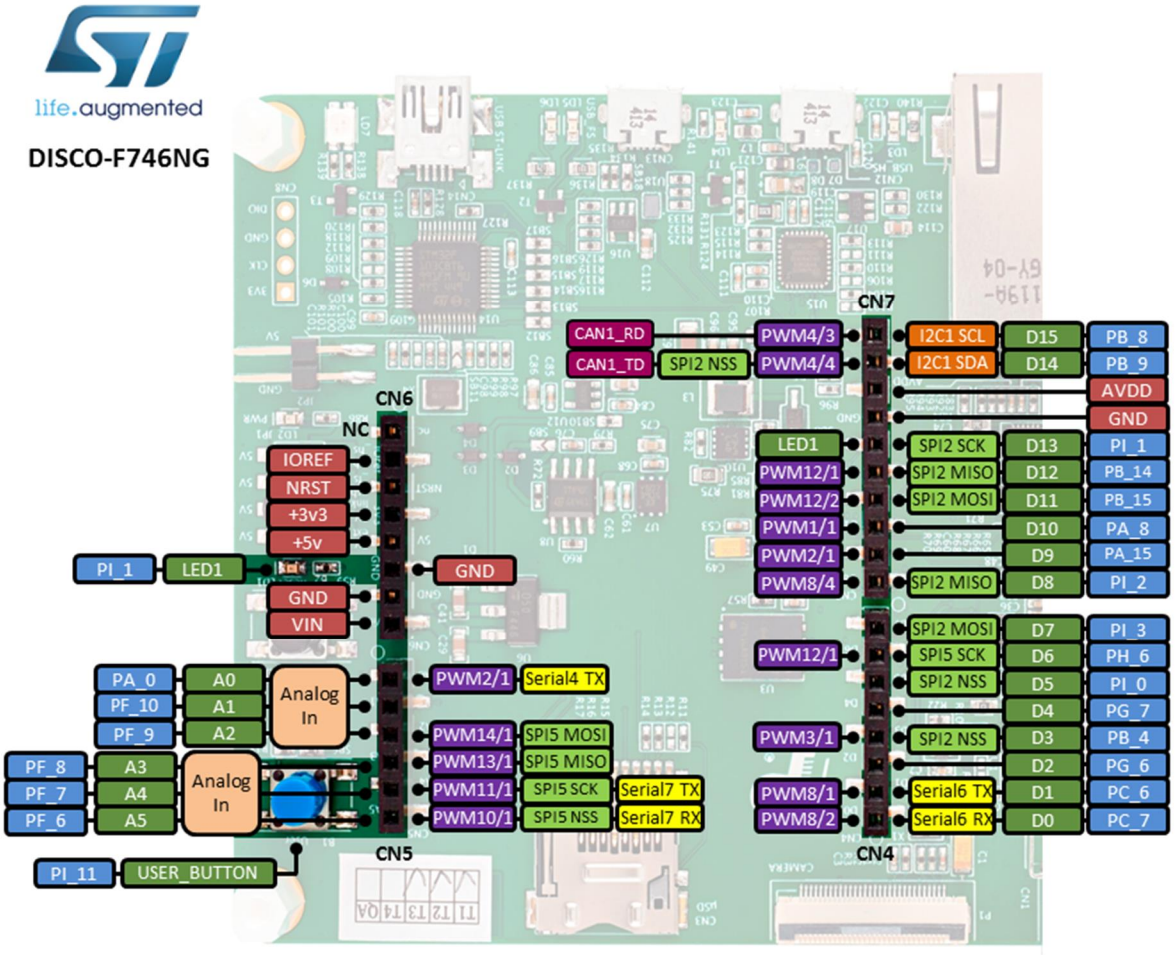
2.1 Hlavní elektronické součásti

2.1.1 DISCO F746NG[10]

Jako základ řídicího systému robotické ruky jsem se rozhodla použít vývojovou desku DISCO-F746NG od firmy STMicroelectronics a to z několika důvodů. Deska obsahuje moderní výkonný mikrokontrolér STM32F746NG, integrovaný barevný dotykový displej s úhlopříčkou 4,3 palce a na zadní straně je vyvedeno 21 signálů mikrokontroléru majících různé funkce (digitální vstup a výstup, analogový vstup, UART, SPI, I2C, PWM), jak je patrné z obrázku 9. Z hlediska připojení robotických manipulátorů je zejména důležitá komunikace SPI, která je použita při komunikaci s motorovými drivery powerSTEP01, dále komunikace UART, která se používá k připojení Bluetooth komunikátorů HC-05, I2C komunikace, která pomocí dvou drátů umožňuje připojení až 128 zařízení (počet připojených zařízení může být až několik tisíc, protože připojené mikrokontroléry se mohou dále větvit) a PWM signály potřebné k ovládání servomotorů robotických manipulátorů. Napájení desky je možné v rozsahu 7 – 12 VDC přes pin VIN.

Obrázek 8: Foto DISCO F746NG[10]

Vývojová deska DISCO F746NG obsahuje řadu dalších funkcí, které mohou být použity v následujících etapách tohoto projektu, jako jsou externí 8MB SDRAM, externí 16MB QSPI FLASH paměť, konektor pro mikroSD kartu, konektory Ethernet, USB HS, USB FS, vstupy pro mikrofony a audio kodek, digitální mikrofony a možnost připojení kamery. Tyto všechny funkce jsou dostupné díky cenové politice firmy STMicroelectronic za příznivou cenu 1228 Kč[11] bez DPH, deska je navíc snadno programovatelná ve vývojovém prostředí Mbed.



Obrázek 9: Pinout desky DISCO F746NG[10]

2.1.2 PowerSTEP01[12]

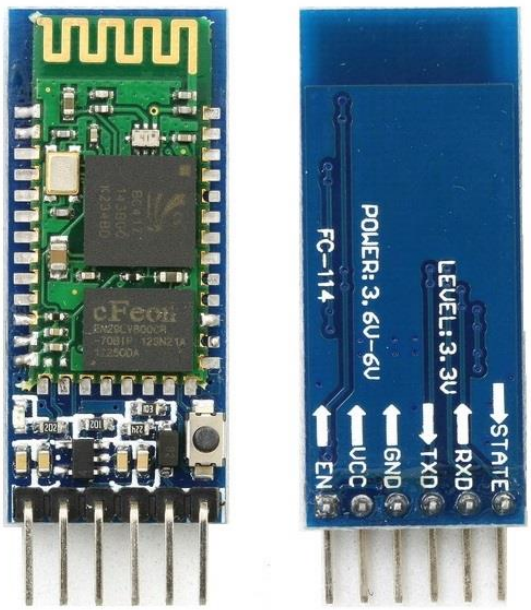
K ovládání bipolárních krokových motorů robotických manipulátorů jsem si vybrala vývojovou desku X-NUCLEO-IHM03A1 od firmy STMicroelectronics, obsahující motorový driver PowerSTEP01. PowerSTEP01 je zatím nejmodernějším a nejvýkonnějším integrovaným motorovým driverem pro bipolární krokové motory, umožňujícím připojení motorů napájených až 85 V a využívajících proud až 10 A. Hlavní výhodou těchto motorových driverů je ale komunikace s řídicím mikrokontrolérem pomocí protokolu SPI, kterým se posílá série příkazů, jež motorový driver vykonává už samostatně bez nutností neustálého řízení mikrokontrolérem, toto je hlavní rozdíl oproti klasickým motorovým driverům starším generace, například MC3479 [9] použitých v Robotické ruce FRENGP, které vyžadují neustálé odtikávání jednotlivých kroků z mikrokontroléru a tedy není principiálně možno ovládat všechny čtyři motory, aby jely současně. Driver PowerSTEP01 umožňuje ještě rozdělení jednoho kroku motoru na až 128 mikrokroků, což zajišťuje hladký pohyb manipulátorů. Velmi důležitou funkcí je vnitřní enkodér, který umožňuje zadat aktuální polohu motorů s rozlišením více než 4 000 000 kroků a velmi přesná regulace rychlosti na požadovanou hodnotu.

Ovládání motorového driveru z mikrokontroléru je potom velice snadné, stačí po SPI poslat příkazy jako GoTo(pozice) pro přesun na danou pozici, MOVE(směr, počet kroků), pro pohyb daným směrem o zadaný počet kroků nebo RUN(směr, rychlost) pro pohyb daným směrem konstantní rychlostí. Pro pohyb motoru jsem hlavně používala funkci GoTo s předchozím nastavením maximální rychlosti pohybu. Všechny tyto funkce jsou dostupné na vývojové desce X-NUCLEO-IHM03A1 za pouhých 218 Kč bez DPH [13], celkem tedy pro všechny 4 drivery 872 Kč bez DPH.

Obrázek 10: Foto vývojové desky X-NUCLEO-IHM03A s motorovým driverem POWERSTEP01[12]

2.1.3 Bluetooth modul HC-05[14]

Pro možnost ovládání robotické ruky z mobilních zařízení (telefon, tablet) jsem se rozhodla použít nejběžnější typ Bluetooth modulu HC-05, který komunikuje s mikrokontrolérem pomocí UART protokolu (stejně jako např. tisk na obrazovku) a pro ovládání je velmi jednoduchý. Nastavení modulu HC-05 jsem provedla pomocí návodu z odkazu [14]. Cena modulu se pohybuje kolem 100 Kč včetně DPH[15].



Obrázek 11: Foto Bluetooth modulu HC-05[16]

2.1.4 Akcelerometr s gyroskopem[17]

Robotické manipulátory s bipolárními krokovými motory potřebují na rozdíl od manipulátorů se servomotory vynulování své mechanické pozice (tzv. homing) před spuštěním předpřipraveného programu. K vynulování se nejčastěji používají koncové spínače, já jsem se ale rozhodla místo 4 koncových spínačů a jejich kabeláže pro všechny 4 osy použít jeden integrovaný obvod MPU6050, který v sobě obsahuje 3osý akcelerometr i 3osý gyroskop, komunikuje po I2C sběrnici (SDA, SCL) a napájí se 3,3 V z řídicí jednotky. Hodnoty měření akcelerace jsou volitelné v rozsazích ±2 g, ±4 g, ±8 g a ±16 g, rychlosti otáčení v rozsazích ±250, ±500, ±1000, ±2000 °/sec, s rozlišením 16 bitů. K nulování os manipulátoru byly použity nejmenší z uvedených rozsahů.

Nulování otáčecí základny Robotické ruky FRENGP se provádí pomocí gyroskopu v ose Z, kdy po dojetí na zarážku už gyroskop nedetekuje žádnou rychlost otáčení. Nulování polohy zvedání páteře manipulátoru se provádí podobným způsobem s použitím gyroskopu v ose X. Nulování pozice robotického ramene i kleští se provádí s použitím akcelerometru, který detekuje mechanické vibrace po dojetí šroubu na koncovou zarážku.

Cena napájeného modulu s čipem MPU6050 je 98 Kč bez DPH[18]

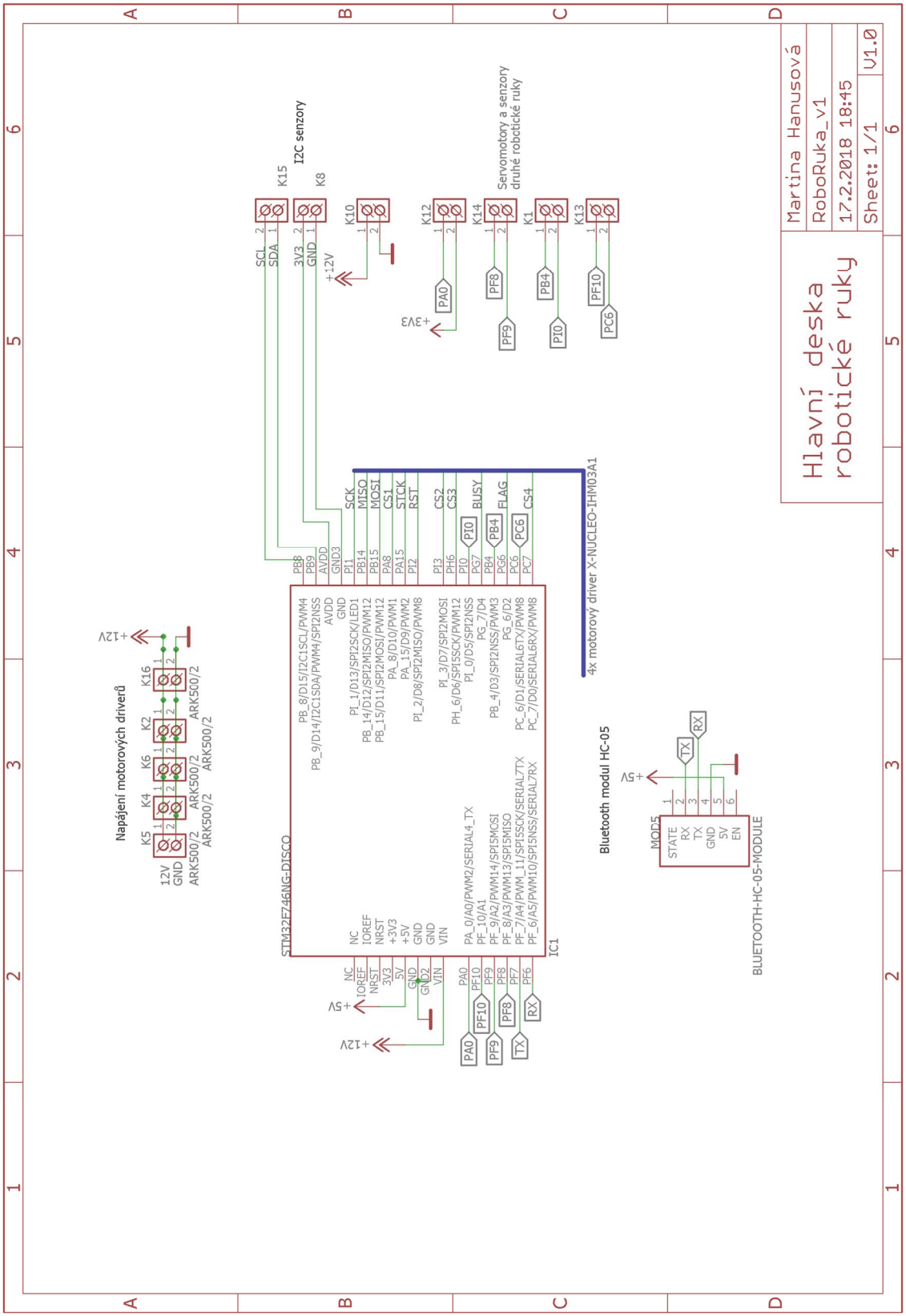
Obrázek 12: Foto modulu s MPU6050[17]

2.2 Desky plošných spojů

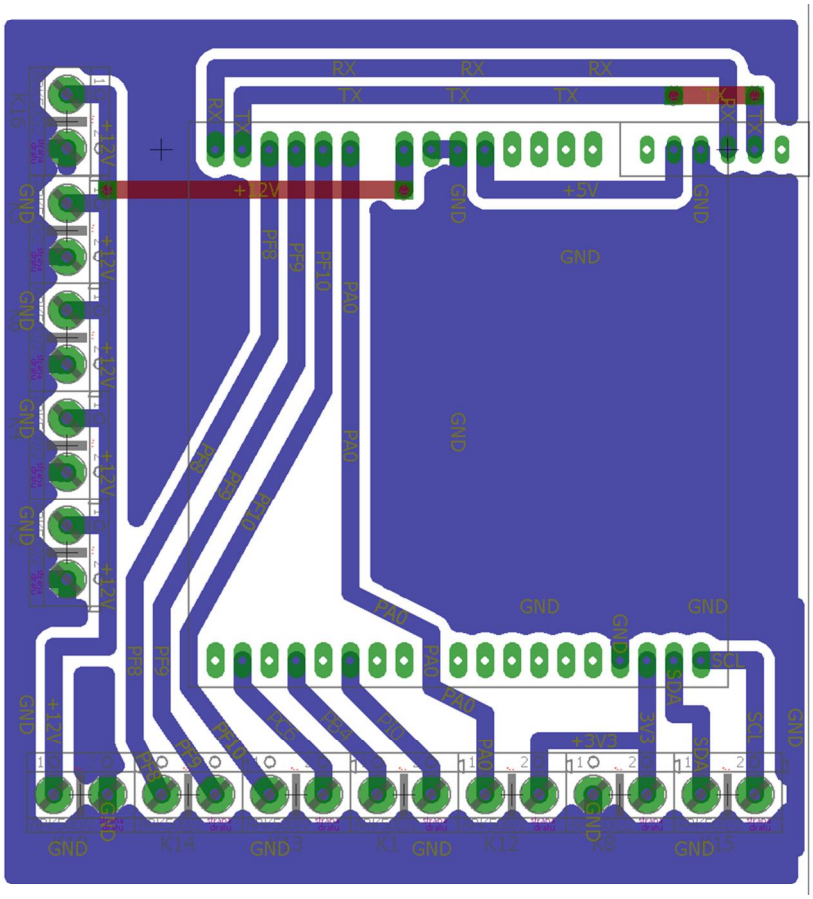
Pro napájení a rozvod signálů z vývojové desky DISCO-F746NG byly vyrobeny 2 desky jednovrstvých plošných spojů (hlavní deska a pomocná deska), klasickou cestou s osvitem fotocitlivé vrstvy UV lampou, vyvoláním v 1% roztoku hydroxidu sodného a vyleptáním v 20% roztoku persíranu sodného. Schémata plošných spojů byla nakreslena v programu Eagle CAD 7.7 a jsou obsažena v elektronické příloze této práce.

2.2.1 Hlavní deska

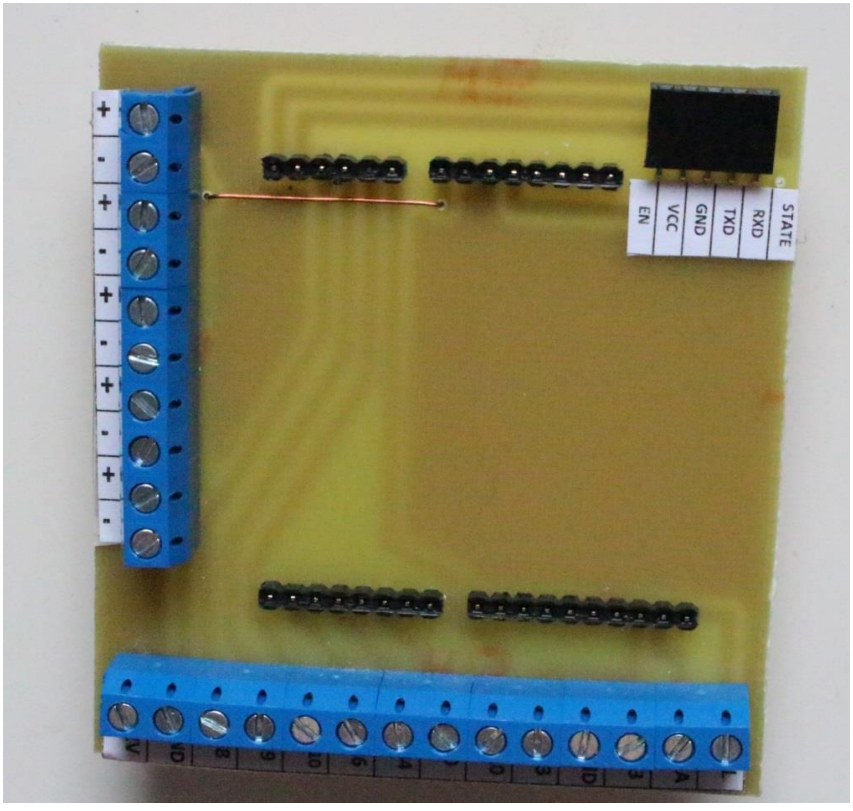
Hlavní deska se připojuje na konec sendvičové sestavy vývojové desky DISCO F746NG a 4 desek motorových driverů X-NUCLEO-IHM03A1 a zajišťuje rozvod napájení vývojových desek, připojení konektoru pro Bluetooth modul HC-05 a vyvedení všech zbylých signálů na svorkovnice.



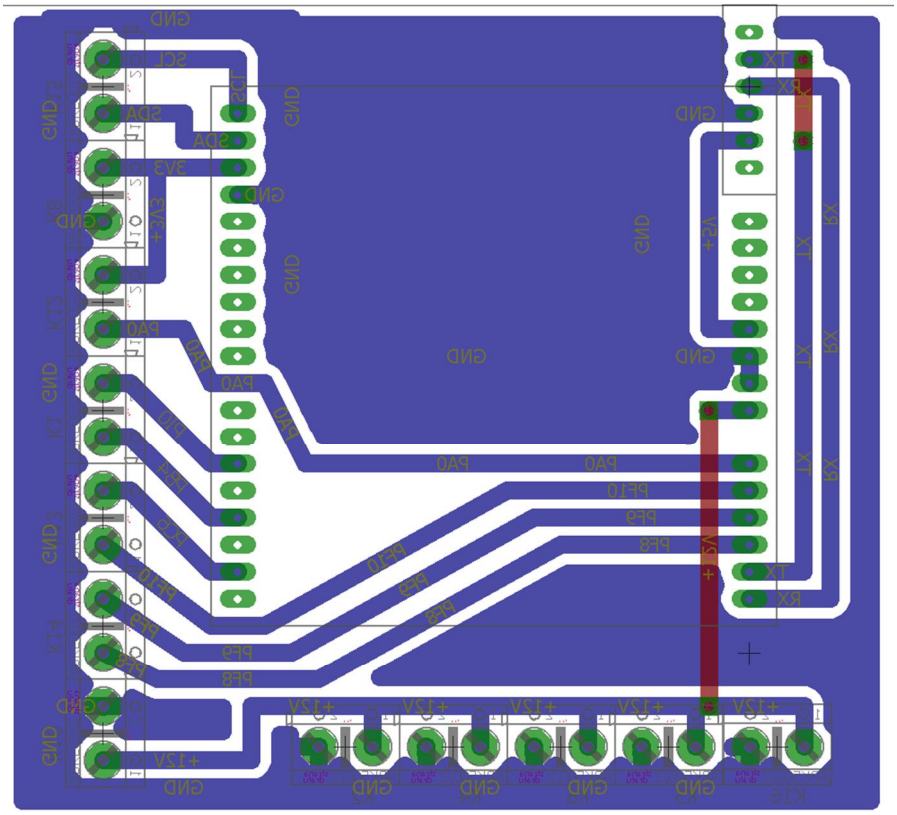
Obrázek 13: Funkční schéma hlavní desky



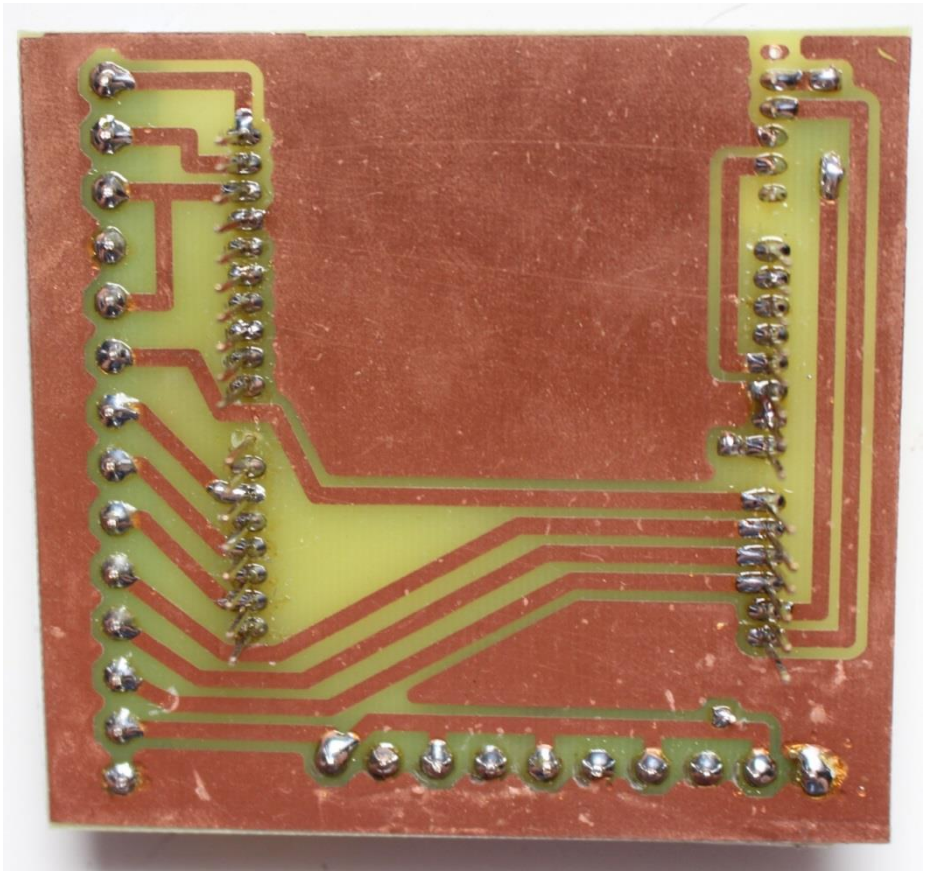
Obrázek 14: Pájecí schéma strany součástek hlavní desky



Obrázek 15: Foto strany součástek plošného spoje hlavní desky



Obrázek 16: Pájecí schéma hlavní desky – strana spojů



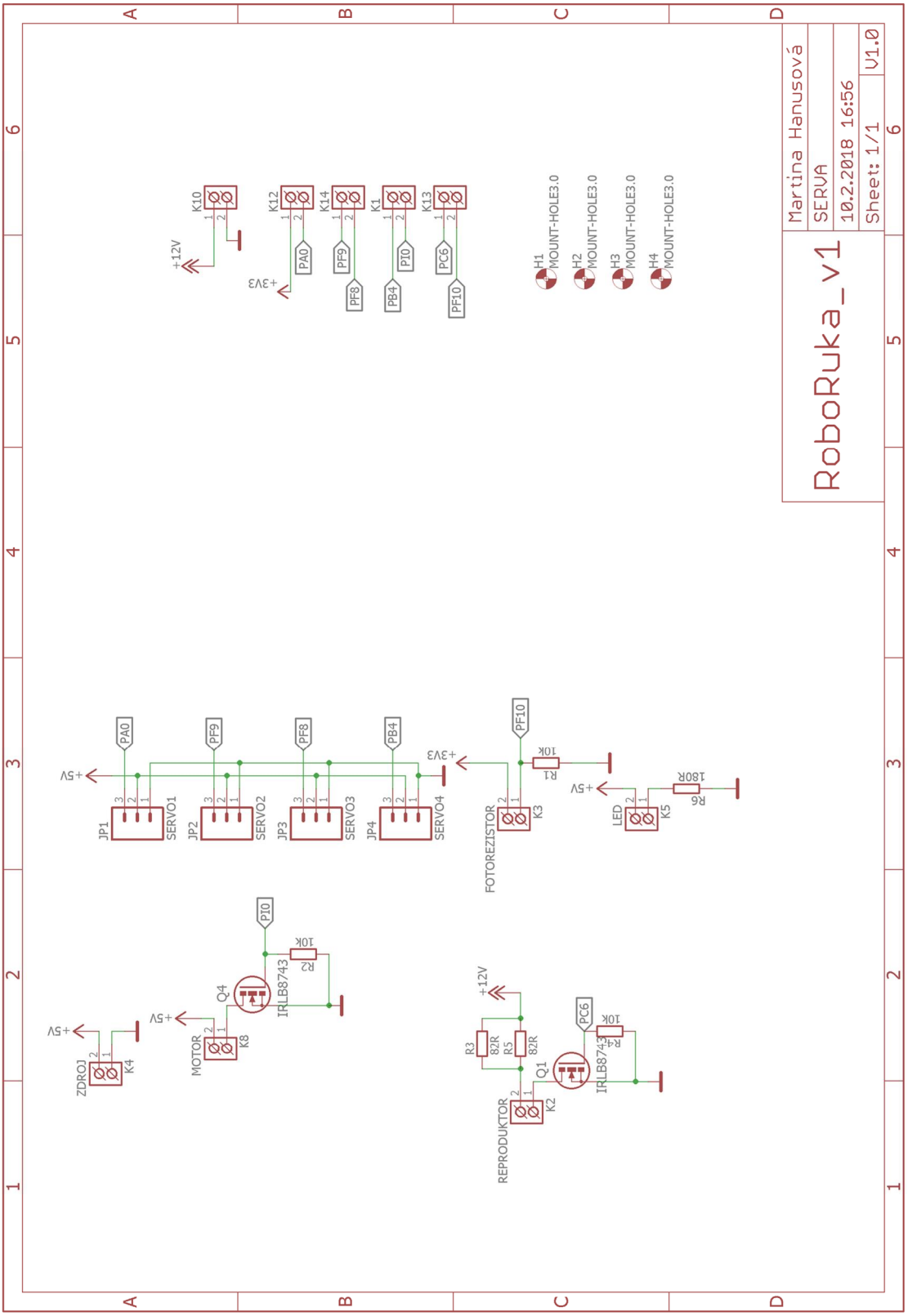
Obrázek 17: Foto strany spojů plošného spoje hlavní desky

Tabulka 1: Tabulka použitých elektronických součástek hlavní desky

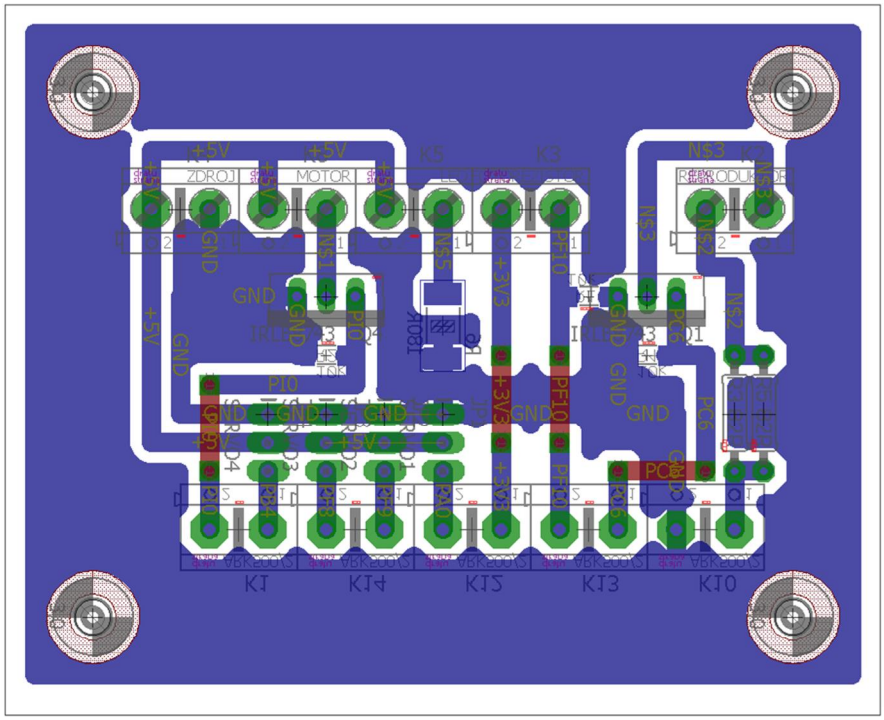
KÓD	TYP	PARAMETRY	FUNKCE	CENA (Kč)[19]
K1, K8, K10, K12, K13, K14	svorkovnice	ARK500/2	připojení pomocné desky	30,-
K2, K4, K5, K6, K16	svorkovnice	ARK500/2	připojení napájení motorových driverů	25,-
K15	svorkovnice	ARK500/2	připojení I2C senzorů	5,-
MOD5	dutinky	6 x 1	připojení Bluetooth modulu	5,-
IC1	pinové headery	6 pinů , 2x 8 pinů, 10 pinů	připojení DISCO-F746NG	5,-
	Deska plošného spoje s fotocitlivou vrstvou 160x100	FR4100x160/3500		58,-
Celková cena				**128,-**

2.2.2 Pomocná deska

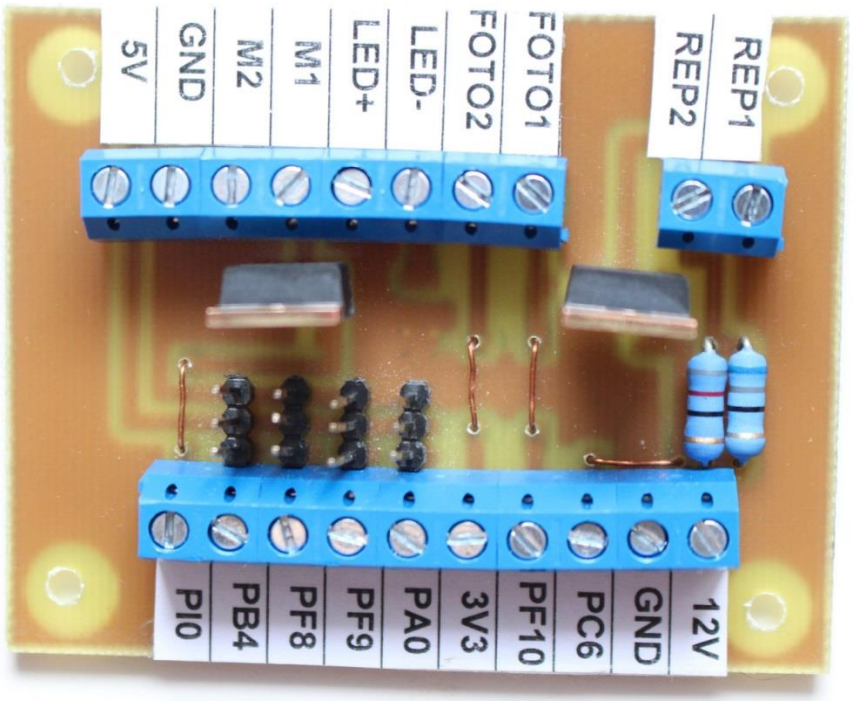
Pomocná deska umožňuje připojení volných signálů z vývojové desky DISCO, které nejsou potřeba pro ovládání robotických manipulátorů s bipolárními krokovými motory. Pomocná deska byla navržena tak, aby umožňovala připojení až 4 servomotorů druhé robotické ruky, jednoho DC motoru pásového dopravníku (přes tranzistor N-MOSFET IRLB8743), jednoho reproduktoru (přes tranzistor N-MOSFET IRLB8743) pro signalizaci chodu robotických zařízení, LED diody a fotorezistoru (přes odporový dělič) optické závory dopravníku. Pomocná deska je tak nachystána pro další etapu k možnosti spolupráce dvou robotických ruk s dávkovacím dopravníkem, v této etapě SOČ je zatím nevyužita.



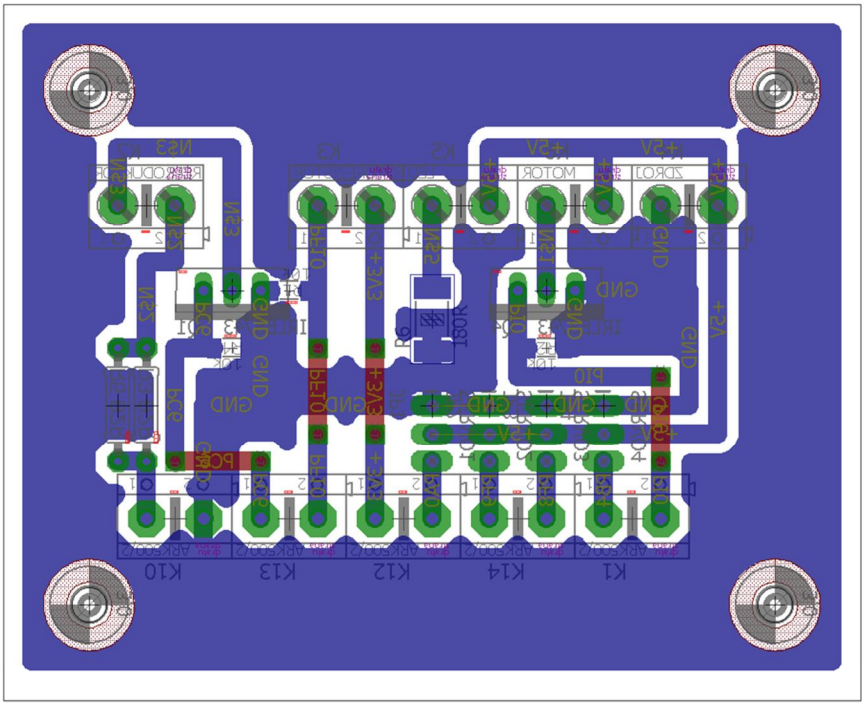
Obrázek 18: Funkční schéma pomocné desky



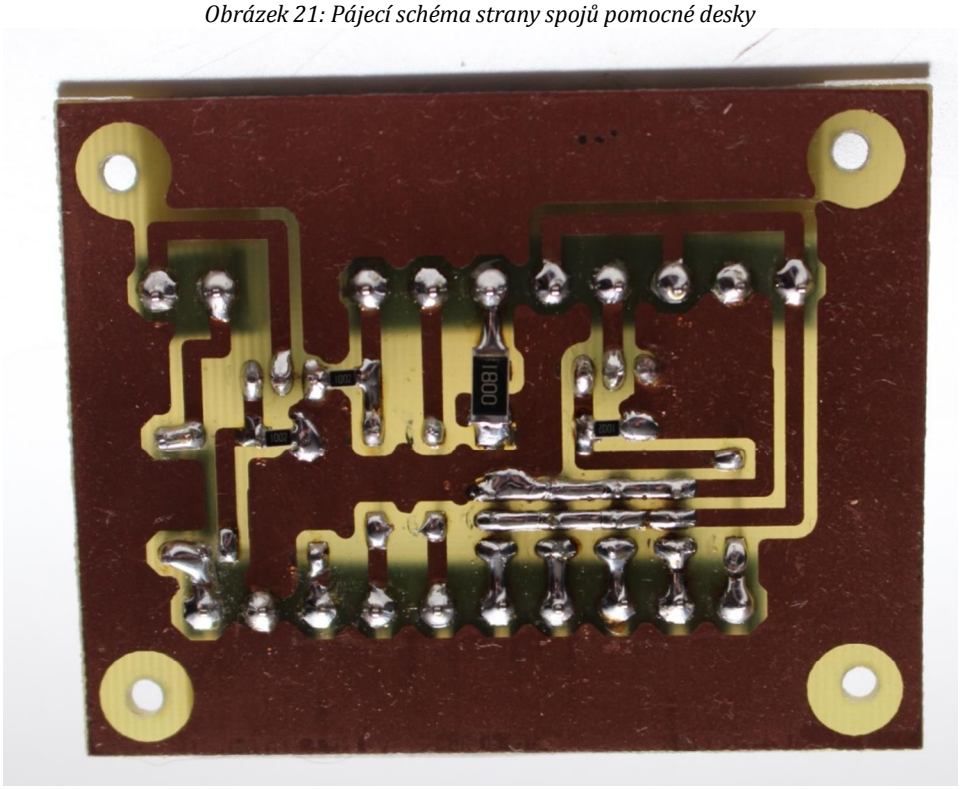
Obrázek 19: Pájecí schéma strany součástek pomocné desky



Obrázek 20: Foto strany součástek pomocné desky



Obrázek 21: Pájecí schéma strany spojů pomocné desky



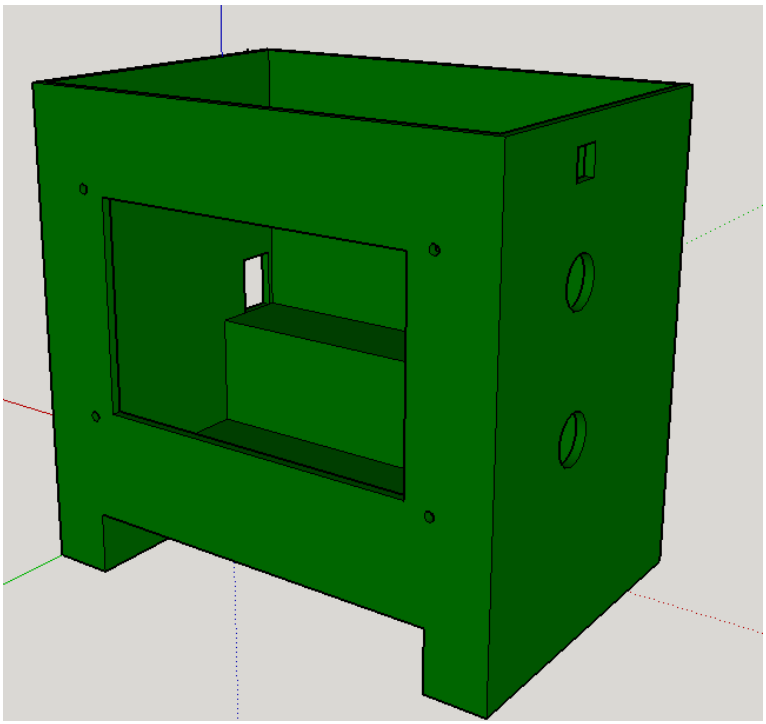
Obrázek 22: Foto strany spojů pomocné desky

Tabulka 2: Tabulka použitých elektronických součástek pomocné desky

KÓD	TYP	PARAMETRY	FUNKCE	CENA (Kč) [19]
JP1 – JP4	piny	3x1	připojení servo motorů	2,-
K1, K10, K12, K13, K14	svorkovnice	ARK500/2	připojení hlavní desky	20,-
K2	svorkovnice	ARK500/2	připojení reproduktoru	5,-
K3	svorkovnice	ARK500/2	připojení fotorezistoru	5,-
K4	svorkovnice	ARK500/2	připojení 5V zdroje	5,-
K5	svorkovnice	ARK500/2	připojení LED	5,-
K8	svorkovnice	ARK500/2	připojení motoru pro pásový dopravník	5,-
R1	rezistor	10 kΩ SMD 1206	odporový dělič	1,-
R2	rezistor	10 kΩ SMD 1206	pull-down tranzistoru	1,-
R3	rezistor	82 Ω 1W THT	proudová ochrana reproduktoru	1,-
R4	rezistor	10 kΩ SMD 1206	pull-down tranzistoru	1,-
R5	rezistor	82 Ω 1W THT	proudová ochrana reproduktoru	1,-
R6	rezistor	180 Ω SMD 2512	omezení proudu na LED	1,-
Q1	tranzistor	IRLB8743	spínání reproduktoru	18,-
Q4	tranzistor	IRLB8743	spínání motoru	18,-
Celková cena				**89,-**

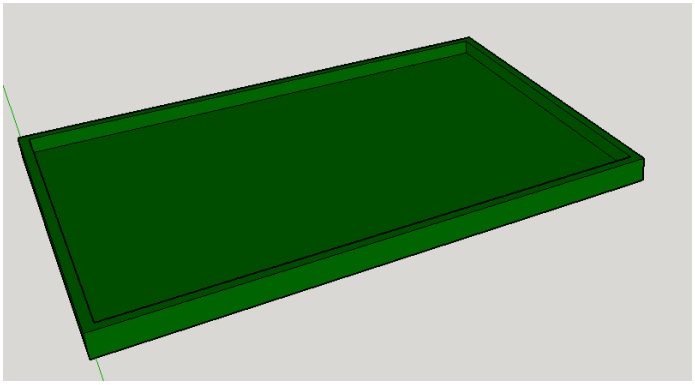
3 Mechanický hardware řídicí jednotky

Modely krabiček s víky a rámečkem displeje pro řídicí jednotku a pro pomocnou desku byly nakresleny v programu SketchUp a po exportu do formátu STL vytištěny dle standardního nastavení na 3D tiskárně Prusa i3 z materiálu PLA.

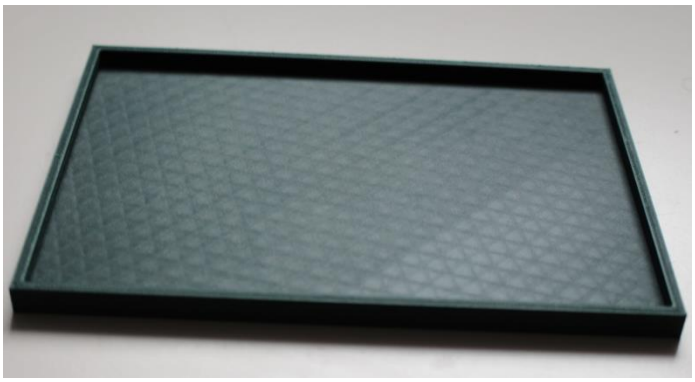


Obrázek 23: Model krabičky řídicí jednotky ve SketchUp

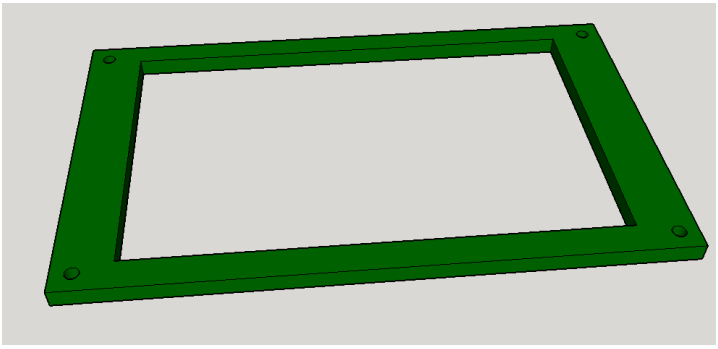
Obrázek 24: Foto vytištěné krabičky řídicí jednotky



Obrázek 25: Model víka krabičky ve SketchUp

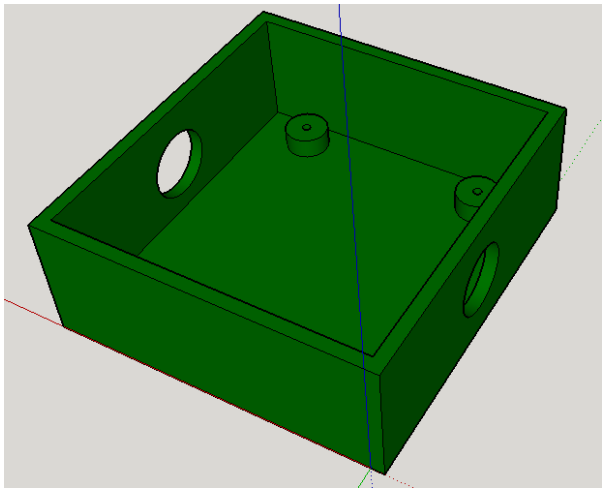


Obrázek 26: Foto vytištěného víka krabičky



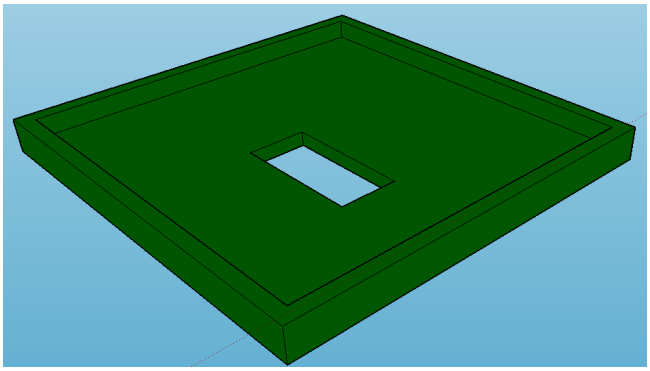
Obrázek 27: Model rámečku na displej ve SketchUp

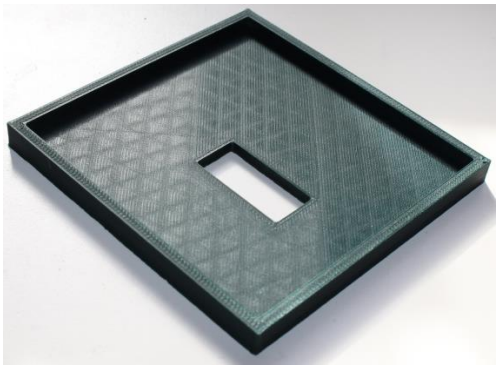
Obrázek 28: Foto vytištěného rámečku na displej



Obrázek 29: Model krabičky pomocné desky ve SketchUp

Obrázek 30: Foto vytištěné krabičky pomocné desky





Obrázek 31: Model víka krabičky pomocné desky ve SketchUp

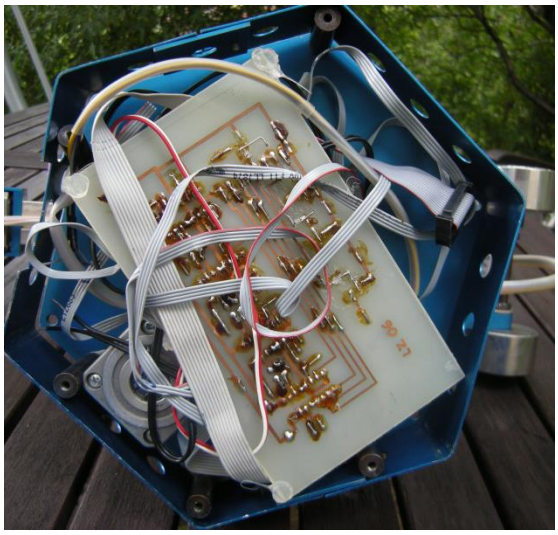
Obrázek 32: Foto vytištěného víka krabičky pomocné desky

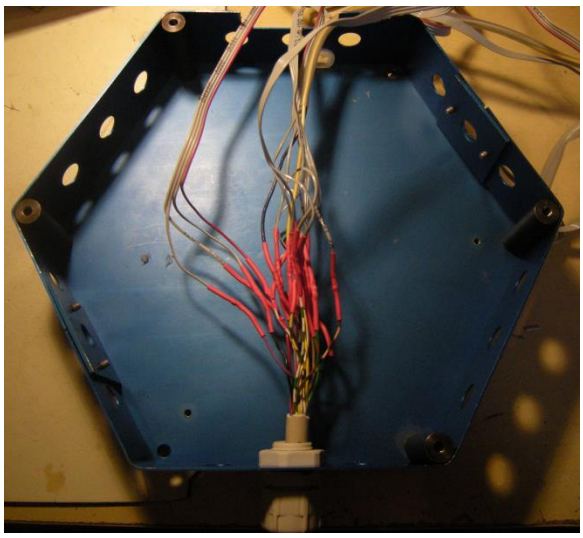
Modely ve SketchUp a vyexportované STL soubory jsou obsaženy v elektronické příloze této práce.

4 Kompletace sestavy řídicí jednotky

Připravené mechanické a elektronické díly popsané v předchozích kapitolách byly zkompletovány do řídicí jednotky Robotické ruky FRENGP následujícím postupem:

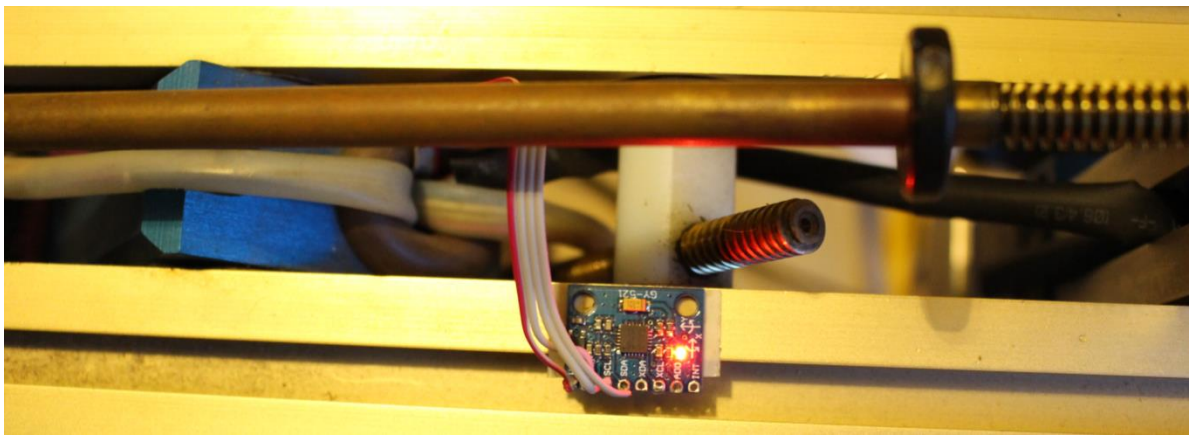
1. Původní řídicí deska motorů Robotické ruky FRENGP byla odstřižena od napájecích kabelů krokových motorů.
2. Jednotlivé krokové motory byly nově připojeny k multižilovému kabelu 20x0,14 mm^2 typu LIYY – TP – 10x2x0.14 a spoje opatřeny smršťovací bužírkou jako izolací. Modul akcelerometru MPU6050 byl připájen k plochému 4žilovému kabelu, který byl v základně rovněž připojen ke kabelu LIYY – TP – 10x2x0.14. K mechanickému upevnění kabelu v základně robota slouží průchodka PG9. Barevné značení jednotlivých kabelů je uvedeno v Tabulce 3.





Obrázek 33: Původní zapojení elektroniky Robotické ruky FRENGP v základně robota
Obrázek 34: Nové zapojení kabeláže Robotické ruky FRENGP v základně robota

3. Modul akcelerometru MPU6050 byl přilepen tenkou oboustrannou lepicí páskou k jednomu z hliníkových profilů páteře robotického manipulátoru a napájecí dráty povytaženy tak, aby nebyly namáhány pohybem manipulátoru.



Obrázek 35: Modul akcelerometru MPU6050 přilepený na hliníkovém profilu

Tabulka 3: Tabulka zapojení motorů a MPU6050 na robotické ruce FRENGP

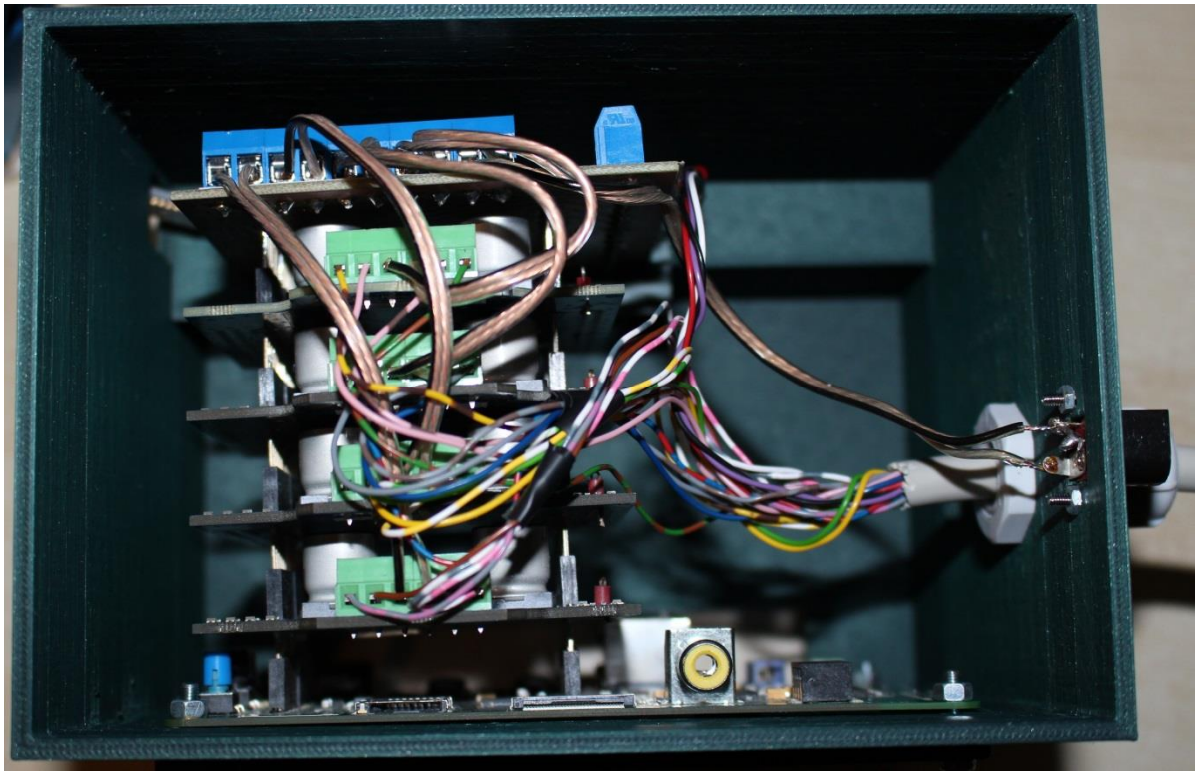
Barva kabelu	**Zapojení**	**Funkce**
BÍLORŮŽOVÁ	MOTOR_1_A-	OTÁČENÍ ZÁKLADNY
ŠEDORŮŽOVÁ	MOTOR_1_A+	
HNĚDOBÍLÁ	MOTOR_1_B-	
HNĚDÁ	MOTOR_1_B+	
ZELENOBÍLÁ	MOTOR_2_A-	PÁTEŘ
ZELENÁ	MOTOR_2_A+	
ŽLUTOBÍLÁ	MOTOR_2_B-	
ŽLUTÁ	MOTOR_2_B+	
ŠEDOHNĚDÁ	MOTOR_3_A-	KLEŠTĚ
MODROČERVENÁ	MOTOR_3_A+	
MODRÁ	MOTOR_3_B-	
ŠEDÁ	MOTOR_3_B+	
ZELENOHNĚDÁ	MOTOR_4_A-	RAMENO
ŽLUTOHNĚDÁ	MOTOR_4_A+	
RŮŽOVOHNĚDÁ	MOTOR_4_B-	
RŮŽOVÁ	MOTOR_4_B+	
BÍLÁ	SDA	MPU6050
ČERNÁ	GND	
ČERVENÁ	3V3	
FIALOVÁ	SCL	

4. Z vývojové desky DISCO-F746NG a 4 motorových driverů X-NUCLEO-IHM03A1 byl vytvořen sendvič, na který byla připojena hlavní deska popsaná v kapitole 2.1. Desky motorových driverů X-NUCLEO-IHM03A1 byly před kompletací do sendviče upraveny přepojením jejich pinů CS (chip select) tak, aby výsledné zapojení driverů odpovídalo schématu na obrázku 13.

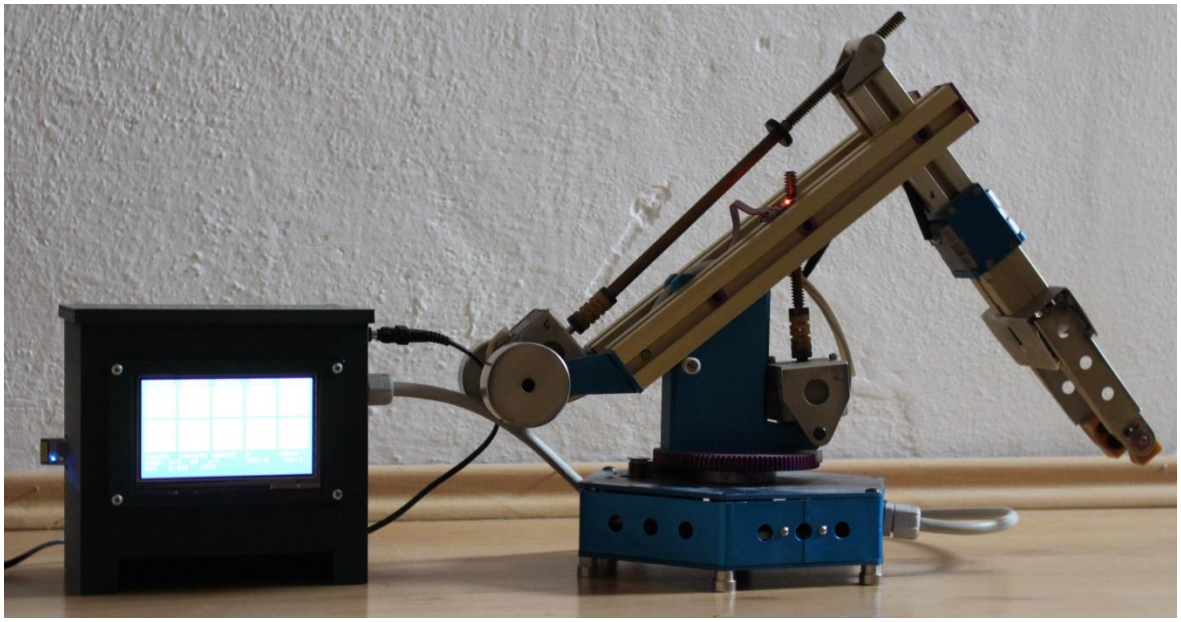
5. Multižilový kabel LIYY – TP – 10x2x0.14, dlouhý 1 m, zapojený jedním koncem do základny Robotické ruky FRENGP byl protažen krabičkou řídicí jednotky skrz průchodku PG9 ve stěně krabičky a jeho jednotlivé barevné žíly popsané v tabulce 3 byly zapojeny do svorkovnic hlavní desky dle schématu na obrázku 13. Do druhého otvoru v boční stěně krabičky byla zapojena zásuvka pro napájecí konektor 2,5/5,5 mm (7 – 12 VDC s GND v plášti), jejíž kabel byl připojený k napájecí svorkovnici hlavní desky.

6. Sendvič těchto pěti desek byl následně vložen do vytištěné krabičky řídicí jednotky a společně s rámečkem na ochranu displeje smontován 4 šroubky M3.

7. Skrz levou stěnu krabičky se do hlavní desky zasune modul Bluetooth HC-05 a tímto je montáž hardware zakončena.



Obrázek 36: Pohled na krabičku se sendvičovou sestavou řídicích desek



Obrázek 37: Foto finální sestavy

5 Software pro STM32F746NG

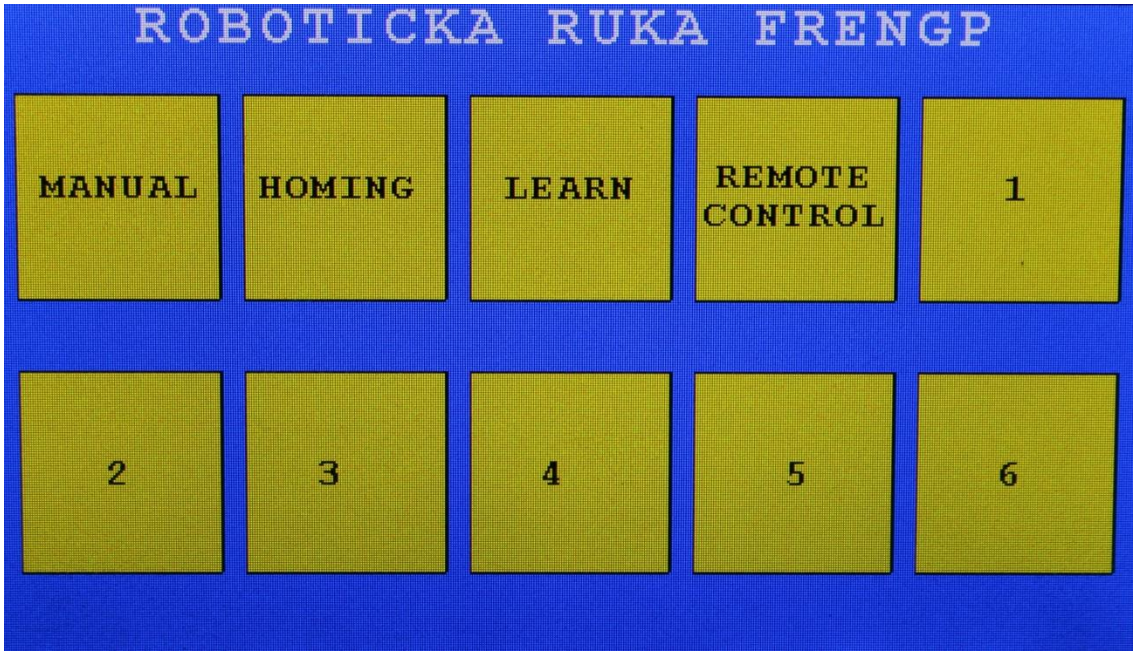
K programování mikrokontroléru STM32F746NG, který je obsažen v desce DISCO F746NG jsem použila vývojové prostředí firmy ARM Mbed OS 5 dostupné online na stránkách os.mbed.com . Vývojové prostředí Mbed má pro začínající programátory několik zásadních výhod: je zadarmo, online dostupné po internetu jen s použitím webového prohlížeče, s programováním v jazyce C s použitím předpřipravených univerzálních funkcí (seznam funkcí s příklady je uveden v referenční příručce[20]) s nimiž je programování mikrokontrolérů ARM stejně jednoduché jako například desek Arduino. Programování dotykového displeje a motorových driverů powerSTEP01 bylo provedeno s použitím knihoven firmy STMicroelectronics pro Mbed[21],[22].

Použití jednotlivých funkcí Mbed je výborně zdokumentováno a vysvětleno na praktických příkladech, takže ovládání i poměrně složitých funkcí jako je například obsluha dotykového displeje se může rychle naučit i naprostý začátečník. Velkou výhodou prostředí Mbed také je snadná přenositelnost programu mezi různými vývojovými deskami s mikrokontroléry ARM a to nejen různých typů jednoho výrobce, ale i mezi různými výrobci, programátorům se tak šetří čas s přechodem jejich programů na modernější či výkonnější platformy.

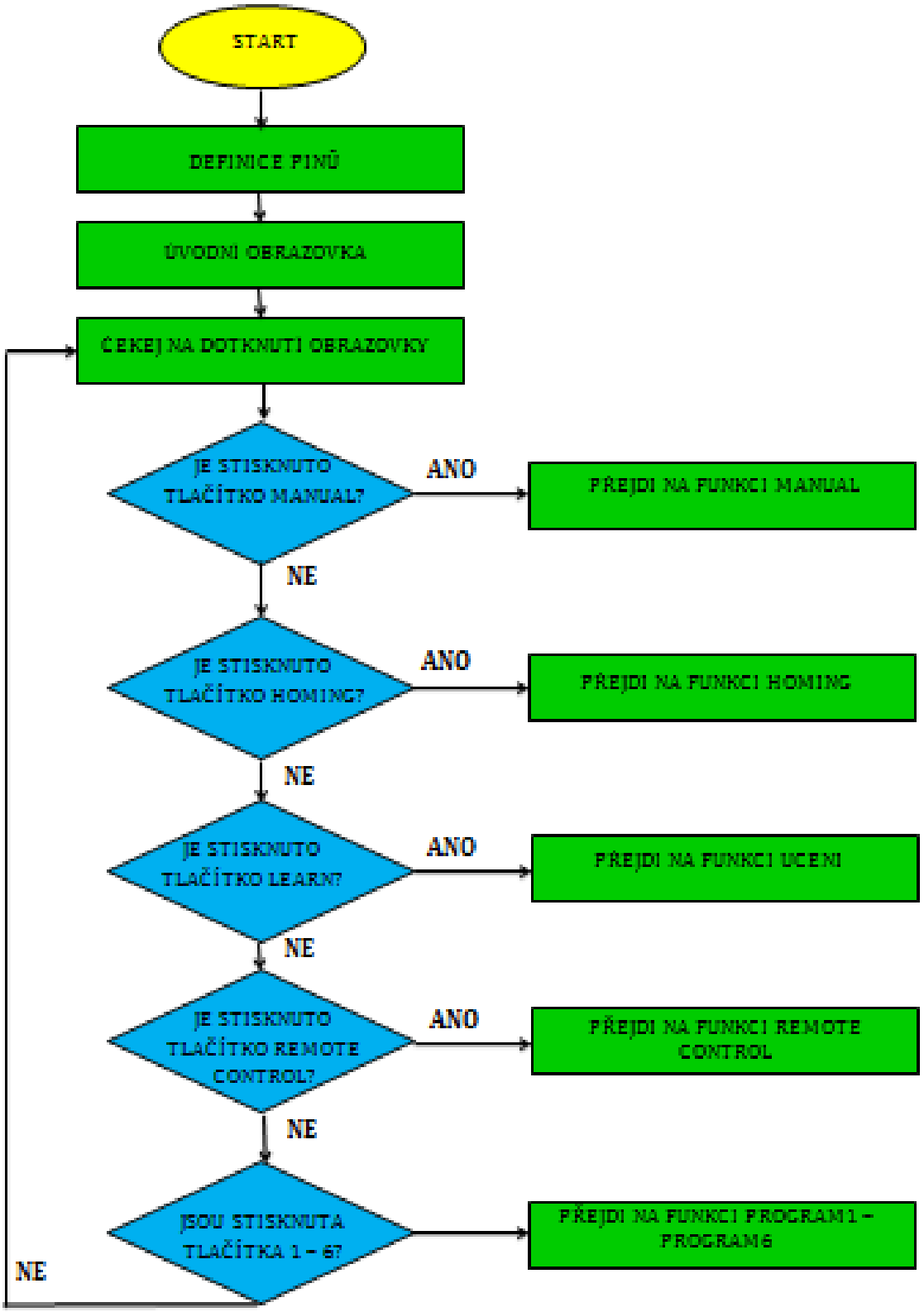
Program pro řízení robotické ruky FRENGP má několik základních funkcí:

1. Ruční ovládání jednotlivých motorů zvolenou rychlostí
2. Automatické nulování os (homing)
3. Učící režim – nahrání pohybové sekvence do paměti mikrokontroléru
4. Spouštění až 6 programů uložených v učícím režimu
5. Dálkové ovládání z mobilního telefonu nebo tabletu přes Bluetooth

Výběr z těchto funkcí se provádí na úvodní obrazovce programu ukázané na obrázku 38 s algoritmem na obrázku 39.



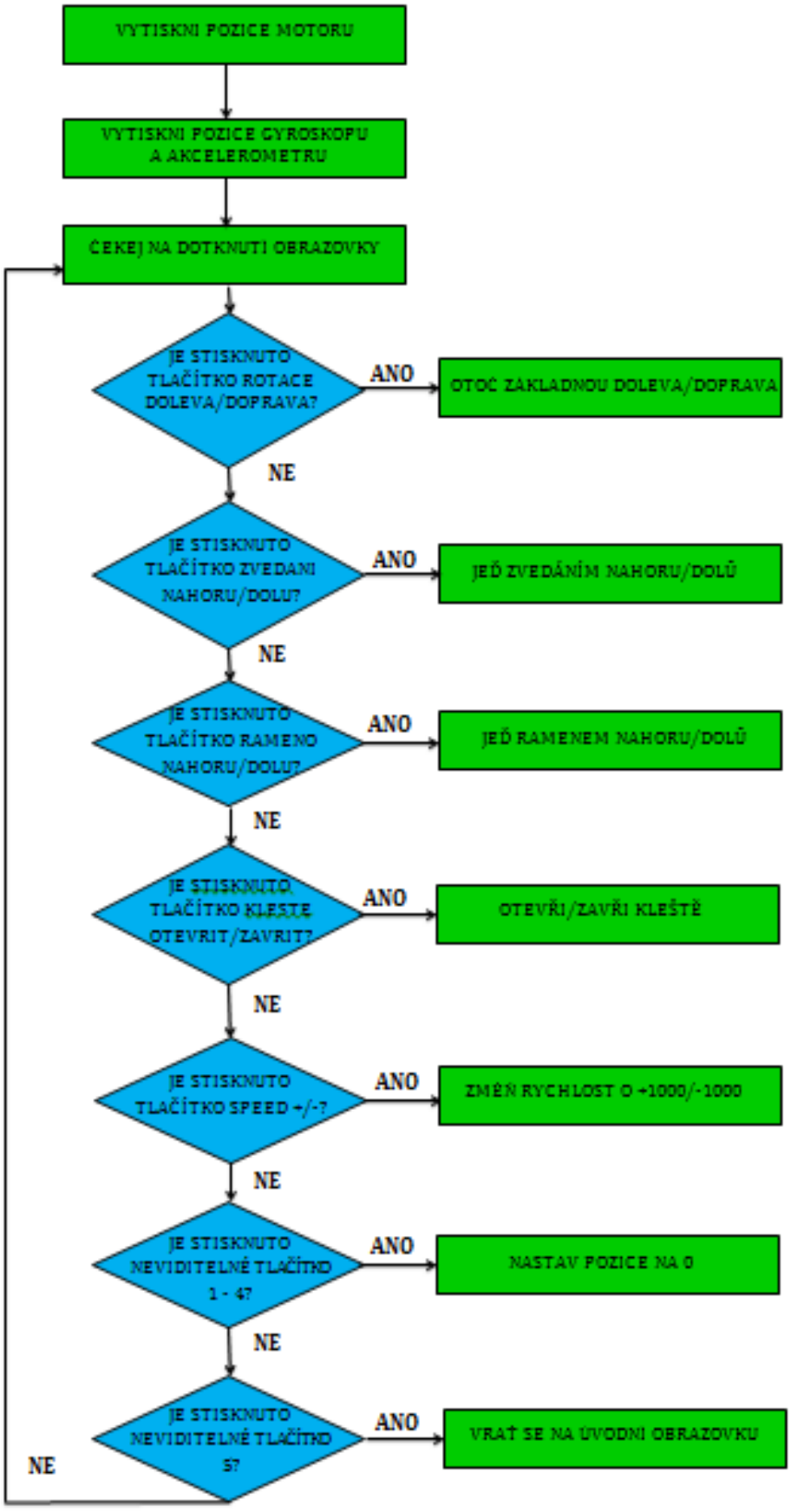
Obrázek 38: Fotografie úvodní obrazovky



Obrázek 39: Algoritmus programu pro úvodní obrazovku

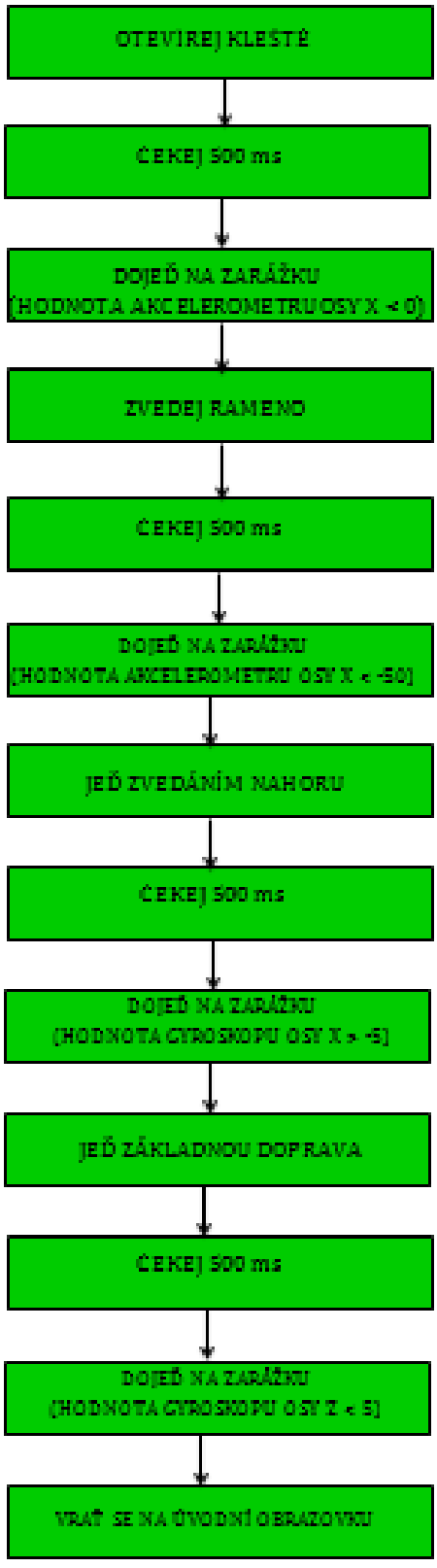
Ruční ovládání umožňuje nezávislý pohyb jednotlivých motorů oběma směry volitelnou rychlostí nastavitelnou tlačítky +/-, s průběžným zobrazováním aktuálních hodnot vnitřního enkodéru jednotlivých motorů a hodnot 3osého akcelerometru a 3osého gyroskopu MPU6050. Z hodnot akcelerometru je vypočítaná i hodnota úhlu náklonu ramene manipulátoru. Fotografie obrazovky ručního ovládání je uvedena na obrázku 40 a algoritmus programu na obrázku 41.

Obrázek 40: Fotografie ručního ovládání



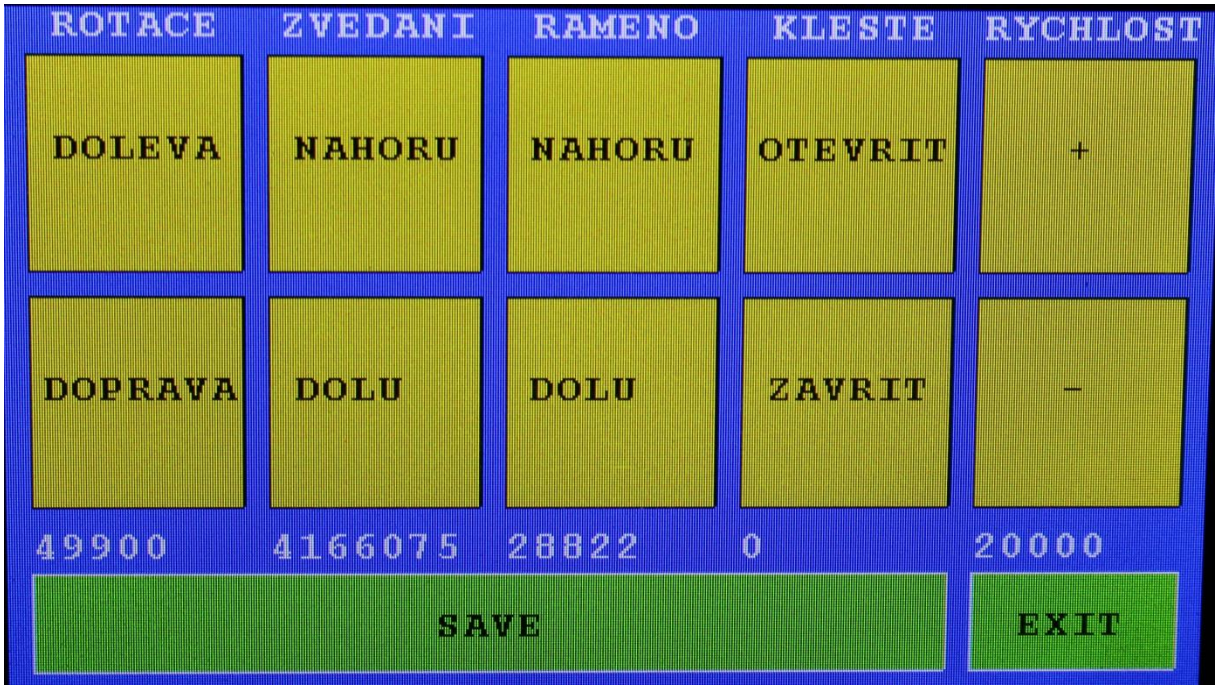
Obrázek 41: Algoritmus programu pro ruční ovládání

Funkce Homing převádí osy manipulátoru do definovaného výchozího stavu, dle algoritmu uvedeného na následujícím obrázku.



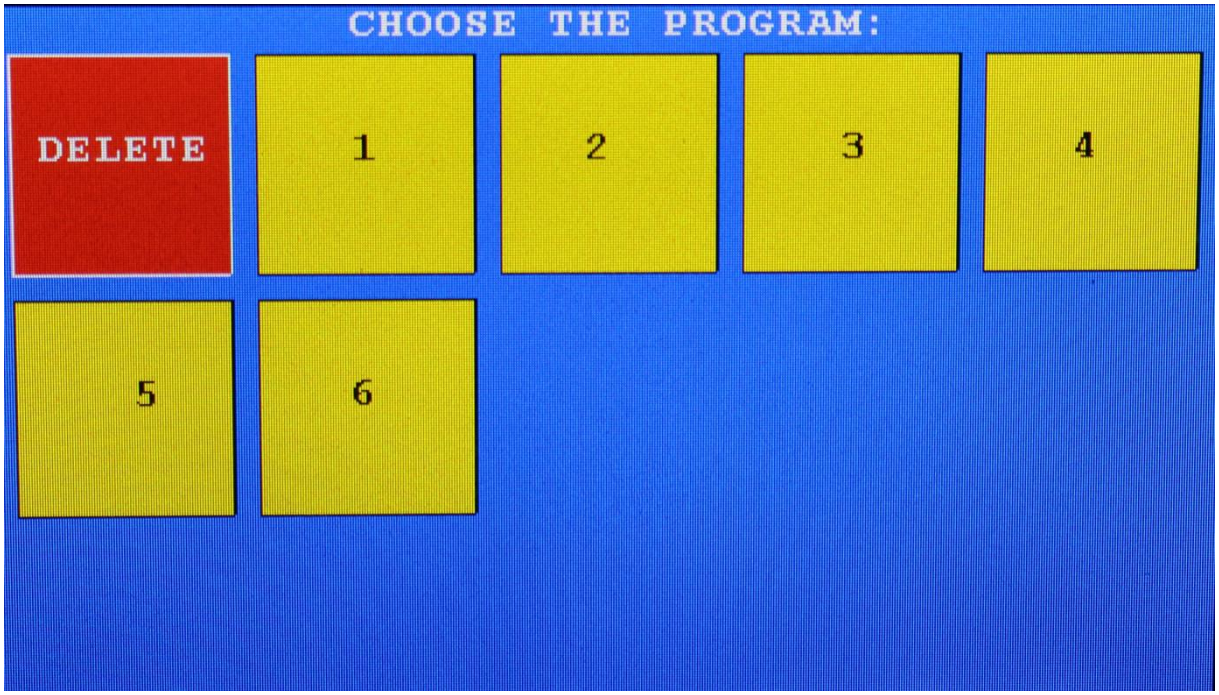
Obrázek 42: Algoritmus programu pro funkci Homing

Režim učení má stejné ovládací prvky jako funkce ručního ovládání, ale po stisknutí tlačítka SAVE uloží aktuální pozice všech 4 motorů, toto může provést až pro 100 navazujících kroků a po stisknutí tlačítka EXIT všechny uložené kroky postupně zopakuje.



Obrázek 43: Fotografie režimu učení

Po zopakování sekvence zadaných kroků program ukáže ukládací obrazovku, kde si uživatel může vybrat číslo programu k uložení sekvence nebo její smazání (DELETE).



Obrázek 44: Fotografie ukládání programů

Režim dálkového ovládání zapne UART komunikaci s Bluetooth modulem HC-05 a vyčkává na konkrétní signály z tabletu nebo mobilního telefonu odpovídající příkazům funkcí ručního ovládání nebo funkce Homing, popsaných v kapitole 6.

5.1 Výpis programu

```
//ROBORUKA NA DISCO-746NG
#include "mbed.h"
#include "PowerSTEP01.h"
#include "math.h"
#include "TS_DISCO_F746NG.h"
#include "LCD_DISCO_F746NG.h"

#define M_PI 3.14159265358979323846
#define LCD_COLOR_BLUE        ((uint32_t)0xFF0000FF)
#define LCD_COLOR_GREEN       ((uint32_t)0xFF00FF00)
#define LCD_COLOR_RED         ((uint32_t)0xFFFF0000)
#define LCD_COLOR_CYAN        ((uint32_t)0xFF00FFFF)
#define LCD_COLOR_MAGENTA     ((uint32_t)0xFFFF00FF)
#define LCD_COLOR_YELLOW      ((uint32_t)0xFFFFFF00)
#define LCD_COLOR_LIGHTBLUE   ((uint32_t)0xFF8080FF)
#define LCD_COLOR_LIGHTGREEN  ((uint32_t)0xFF80FF80)
#define LCD_COLOR_LIGHTRED    ((uint32_t)0xFFFF8080)
#define LCD_COLOR_LIGHTCYAN   ((uint32_t)0xFF80FFFF)
#define LCD_COLOR_LIGHTMAGENTA ((uint32_t)0xFFFF80FF)
#define LCD_COLOR_LIGHTYELLOW ((uint32_t)0xFFFFFF80)
#define LCD_COLOR_DARKBLUE    ((uint32_t)0xFF000080)
#define LCD_COLOR_DARKGREEN   ((uint32_t)0xFF008000)
#define LCD_COLOR_DARKRED     ((uint32_t)0xFF800000)
#define LCD_COLOR_DARKCYAN    ((uint32_t)0xFF008080)
#define LCD_COLOR_DARKMAGENTA ((uint32_t)0xFF800080)
#define LCD_COLOR_DARKYELLOW  ((uint32_t)0xFF808000)
#define LCD_COLOR_WHITE       ((uint32_t)0xFFFFFFFF)
#define LCD_COLOR_LIGHTGRAY   ((uint32_t)0xFFD3D3D3)
#define LCD_COLOR_GRAY        ((uint32_t)0xFF808080)
#define LCD_COLOR_DARKGRAY    ((uint32_t)0xFF404040)
#define LCD_COLOR_BLACK       ((uint32_t)0xFF000000)
#define LCD_COLOR_BROWN       ((uint32_t)0xFFA52A2A)
#define LCD_COLOR_ORANGE      ((uint32_t)0xFFFFA500)
#define LCD_COLOR_TRANSPARENT ((uint32_t)0xFF000000)
#define MANUAL x>10  && x<94  && y>38 && y<123
#define HOMING x>104 && x<188 && y>38 && y<123
#define UCENI  x>198 && x<282 && y>38  && y<123
#define REMOTE_CONTROL  x>292 && x<376 && y>38  && y<123
#define PROGRAM1      x>386 && x<470 && y>38  && y<123
#define PROGRAM2      x>10  && x<94  && y>153 && y<238
#define PROGRAM3      x>104 && x<188 && y>153 && y<238
#define PROGRAM4      x>198 && x<282 && y>153 && y<238
#define PROGRAM5      x>292 && x<376 && y>153 && y<238
#define PROGRAM6      x>386 && x<470 && y>153 && y<238
#define M_ROTACE_DOLEVA    x>10  && x<94  && y>113 && y<198
```

```
#define M_ROTACE_DOPRAVA    x>10  && x<94  && y>18  && y<103
#define M_ZVEDANI_NAHORU    x>104 && x<188 && y>18  && y<103
#define M_ZVEDANI_DOLU      x>104 && x<188 && y>113 && y<198
#define M_RAMENO_NAHORU     x>198 && x<282 && y>18  && y<103
#define M_RAMENO_DOLU       x>198 && x<282 && y>113 && y<198
#define M_KLESTE_OTEVRIT    x>292 && x<376 && y>18  && y<103
#define M_KLESTE_ZAVRIT     x>292 && x<376 && y>113 && y<198
#define M_RYCHLOST_PLUS     x>386 && x<470 && y>18  && y<103
#define M_RYCHLOST_MINUS    x>386 && x<470 && y>113 && y<198
#define M_POS1              x>10  && x<94  && y>198 && y<272
#define M_POS2              x>104 && x<188 && y>198 && y<272
#define M_POS3              x>198 && x<282 && y>198 && y<272
#define M_POS4              x>292 && x<376 && y>198 && y<272
#define M_HOME              x>386 && x<470 && y>198 && y<272
#define SAVE                x>10  && x<376 && y>225 && y<265
#define EXIT                x>386 && x<470 && y>225 && y<265
DigitalIn    TLA(PI_11);

DigitalIn    FLAG (D2);
DigitalIn    BUSY (D4);
DigitalIn    DRDY (D6);
DigitalOut   RST  (D8);
DigitalOut   STCK (D9);
DigitalOut   CS1  (D10);
DigitalOut   CS2  (D7);
DigitalOut   CS3  (D6);
DigitalOut   CS4  (D0);

SPI         spi(D11, D12, D13); // mosi, miso, sclk
Serial      pc(USBTX, USBRX, 115200);

Serial BT (PF_7, PF_6, 115200);
I2C i2c(PB_9, PB_8);  // SDA, SCL

LCD_DISCO_F746NG lcd;
TS_DISCO_F746NG ts;
TS_StateTypeDef TS_State;

Ticker      Sekunda;
Timer       Stopky;

void Nastav_Gyro();
void Homing();
void Manual();
void Motor_1_uceni();
void Motor_2_uceni();
void Motor_3_uceni();
void Motor_4_uceni();
void UvodniObrazovka();
void ts_check_fun();
int  NactiDotek();
int obrazovka=1;
```

```
void Button(int pozice_x,int pozice_y, int sirka, int vyska,unsigned long barva, unsigned long ramecek);
void Text(int poz_x, int poz_y,int font,unsigned long barva_textu,unsigned long barva_pozadi,char *text);
void Cislo(int poz_x,int poz_y, int font,unsigned long barva_textu,unsigned long barva_pozadi,long cislo);
void VytiskniGyro(int poz_x, int poz_y,int font,unsigned long barva_textu,unsigned long barva_pozadi);
void VytiskniPozice(int poz_x, int poz_y,int font,unsigned long barva_textu,unsigned long barva_pozadi);
void Precti_Gyro();
void RemoteControl();
void Uceni();
void Ukladani();
void Program1();
void Program2();
void Program3();
void Program4();
void Program5();
void Program6();
int pocet_doteku=0, predchozi_pocet_doteku=0, prvni_dotek=0, pastX, pastY, x, y,
krok=0,step1=0,step2=0,step3=0,step4=0,program1_step, program2_step,
program3_step, program4_step, program5_step, program6_step;
long Gyro_X, Gyro_Y, Gyro_Z, POS1, POS2, POS3, POS4,
pozice1[100],pozice2[100], pozice3[100], pozice4[100], program1[5][100],
program2[5][100],
program3[5][100],program4[5][100],program5[5][100],program6[5][100] ;
double Naklon, Rotace;
long ACC_X, ACC_Y, ACC_Z, TEMP;
unsigned long barva_tlacitka = LCD_COLOR_YELLOW, barva_textu_pozadi = LCD_COLOR_WHITE,
barva_textu_tlacitko = LCD_COLOR_BLACK, barva_pozadi = LCD_COLOR_BLUE;
double Ax, Ay, Az, Ax2, Ay2, Az2;
char MPU6050[20], I2C_cmd[10];

long spd=20000;
bool sekunda_flag=false, konec_flag=false;
int32_t celkova_pozice=0, pozice=0, minula_pozice=0, preteceni_pozice=0, enkoder=0;
int32_t vysledek2, CisloMereni=0;
double rychlost, interval;
double diag, cas;
int i;
uint8_t text[50];

unsigned char BTread[10];


void ts_check_fun()
{
  ts.GetState(&TS_State);
  pocet_doteku = TS_State.touchDetected;
}
```

```
/*****************************************************************************/
/********************** OBSLUHA DRIVERU POWERSTEP01 ***************************/
/*****************************************************************************/

 void driver_spi(uint8_t* ReadBuffer, uint8_t* WriteBuffer, DigitalOut CS)
  {
     CS = 0;
     wait_us(1);
     ReadBuffer[0] = spi.write(WriteBuffer[0]);
     CS = 1;
     wait_us(1);
  }

/*****************************************************************************************************/

uint32_t ReadDriverRegister(uint8_t RegisterCode, DigitalOut DriverCSpin)
{
   uint32_t spiRxData;
   uint8_t Readspi[10], Writespi[10];

   for(int i=0; i<10; i++) {Readspi[i] = 0; Writespi[i] = 0;}

   Writespi[0]=RegisterCode+32;

 switch (RegisterCode)
  {
    case POWERSTEP01_ABS_POS:
    case POWERSTEP01_MARK:
    case POWERSTEP01_SPEED:
        driver_spi(&Readspi[0], &Writespi[0], DriverCSpin);
        driver_spi(&Readspi[1], &Writespi[1], DriverCSpin);
        driver_spi(&Readspi[2], &Writespi[2], DriverCSpin);
        driver_spi(&Readspi[3], &Writespi[3], DriverCSpin);
        spiRxData = ((uint32_t)Readspi[1] << 16)| (Readspi[2] << 8) | (Readspi[3]);
        break;
    case POWERSTEP01_EL_POS:
    case POWERSTEP01_ACC:
    case POWERSTEP01_DEC:
    case POWERSTEP01_MAX_SPEED:
    case POWERSTEP01_MIN_SPEED:
    case POWERSTEP01_FS_SPD:
    case POWERSTEP01_INT_SPEED:
    case POWERSTEP01_CONFIG:
    case POWERSTEP01_GATECFG1:
    case POWERSTEP01_STATUS:
        driver_spi(&Readspi[0], &Writespi[0], DriverCSpin);
        driver_spi(&Readspi[1], &Writespi[1], DriverCSpin);
        driver_spi(&Readspi[2], &Writespi[2], DriverCSpin);
        spiRxData = ((uint32_t)Readspi[1] << 8)| (Readspi[2]);
        break;
   default:
        driver_spi(&Readspi[0], &Writespi[0], DriverCSpin);
```

```
        driver_spi(&Readspi[1], &Writespi[1], DriverCSpin);
        spiRxData = (uint32_t)Readspi[1];
 }
return spiRxData;
}

/**************************************************************************************************************/

void WriteDriverCommand(uint8_t CommandCode, uint32_t CommandValue, DigitalOut DriverCSpin)
{
   uint8_t Readspi[10], Writespi[10];
   for(int i=0; i<10; i++) {Readspi[i] = 0; Writespi[i] = 0;}

   Writespi[0]=CommandCode;

 switch (CommandCode)
   {
    case POWERSTEP01_ABS_POS:
    case POWERSTEP01_MARK:
    case POWERSTEP01_RUN_FWD:
    case POWERSTEP01_RUN_REV:
    case POWERSTEP01_MOVE_FWD:
    case POWERSTEP01_MOVE_REV:
    case POWERSTEP01_GO_TO:
    case POWERSTEP01_GO_TO_REV:
    case POWERSTEP01_GO_TO_FWD:
    case POWERSTEP01_GO_UNTIL_FWD_RST:
    case POWERSTEP01_GO_UNTIL_REV_RST:
    case POWERSTEP01_GO_UNTIL_FWD_MARK:
    case POWERSTEP01_GO_UNTIL_REV_MARK:
        Writespi[1] = (uint8_t)(CommandValue >> 16);
        Writespi[2] = (uint8_t)(CommandValue >> 8);
        Writespi[3] = (uint8_t) CommandValue;
        driver_spi(&Readspi[0], &Writespi[0], DriverCSpin);
        driver_spi(&Readspi[1], &Writespi[1], DriverCSpin);
        driver_spi(&Readspi[2], &Writespi[2], DriverCSpin);
        driver_spi(&Readspi[3], &Writespi[3], DriverCSpin);
        driver_spi(&Readspi[4], &Writespi[4], DriverCSpin);
        break;
    case POWERSTEP01_EL_POS:
    case POWERSTEP01_ACC:
    case POWERSTEP01_DEC:
    case POWERSTEP01_MAX_SPEED:
    case POWERSTEP01_MIN_SPEED:
    case POWERSTEP01_FS_SPD:
    case POWERSTEP01_INT_SPEED:
    case POWERSTEP01_CONFIG:
    case POWERSTEP01_GATECFG1:
        Writespi[1] = (uint8_t)(CommandValue >> 8);
        Writespi[2] = (uint8_t) CommandValue;
        driver_spi(&Readspi[0], &Writespi[0], DriverCSpin);
        driver_spi(&Readspi[1], &Writespi[1], DriverCSpin);
```

```
        driver_spi(&Readspi[2], &Writespi[2], DriverCSpin);
        driver_spi(&Readspi[3], &Writespi[3], DriverCSpin);
        break;
   case POWERSTEP01_ADC_OUT:
   case POWERSTEP01_OCD_TH:
   case POWERSTEP01_STEP_MODE:
   case POWERSTEP01_ALARM_EN:
   case POWERSTEP01_GATECFG2:
   case POWERSTEP01_KVAL_HOLD:
   case POWERSTEP01_KVAL_RUN:
   case POWERSTEP01_KVAL_ACC:
   case POWERSTEP01_KVAL_DEC:
   case POWERSTEP01_ST_SLP:
   case POWERSTEP01_FN_SLP_ACC:
   case POWERSTEP01_FN_SLP_DEC:
   case POWERSTEP01_K_THERM:
   case POWERSTEP01_STALL_TH:
        Writespi[1] = (uint8_t) CommandValue;
        driver_spi(&Readspi[0], &Writespi[0], DriverCSpin);
        driver_spi(&Readspi[1], &Writespi[1], DriverCSpin);
        driver_spi(&Readspi[2], &Writespi[2], DriverCSpin);
        break;
   default:
        driver_spi(&Readspi[0], &Writespi[0], DriverCSpin);
        driver_spi(&Readspi[1], &Writespi[1], DriverCSpin);
 }
}

/**************************************************************************************************/

uint32_t GetDriverStatus(DigitalOut DriverCSpin)
{
   uint32_t spiRxData;
   uint8_t Readspi[10], Writespi[10];

   for(int i=0; i<10; i++) {Readspi[i] = 0; Writespi[i] = 0;}
   Writespi[0]=0xD0;

   driver_spi(&Readspi[0], &Writespi[0], DriverCSpin);
   driver_spi(&Readspi[1], &Writespi[1], DriverCSpin);
   driver_spi(&Readspi[2], &Writespi[2], DriverCSpin);
   spiRxData = ((uint32_t)Readspi[1] << 8)| (Readspi[2]);

   return spiRxData;
}

/**************************************************************************************************/
void SettingDrivers()
{
   WriteDriverCommand(POWERSTEP01_ALARM_EN, 0b10000110, DRIVER1);      // Alarmy: thermal,
command error
```

```
  WriteDriverCommand(POWERSTEP01_STEP_MODE, 0b00000111, DRIVER1);   // BUSY output, voltage mode, 128 microsteps
  WriteDriverCommand(POWERSTEP01_CONFIG, 0x0000, DRIVER1);          // 19.5kHz PWM, 7.5V VCC/UVLO, nestop pri nadproudu, bez kompenzace BEMF, SW vyvola HardStop, interni oscilator 16 MHz
  WriteDriverCommand(POWERSTEP01_MIN_SPEED, 7032, DRIVER1); // Nizkorychlostni optimalizace pod 700 kroku/s
  WriteDriverCommand(POWERSTEP01_MAX_SPEED, 20, DRIVER1); // Nizkorychlostni optimalizace pod 700 kroku/s
  WriteDriverCommand(POWERSTEP01_FS_SPD, 1023, DRIVER1); // Neprepinat z mikrostep na fullstep pri prekroceni limitni rychlosti
  WriteDriverCommand(POWERSTEP01_KVAL_HOLD, 255, DRIVER1); // Amplituda napeti v zabrzdenem stavu (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_KVAL_RUN, 255, DRIVER1); // Amplituda napeti v chodu (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_KVAL_ACC, 255, DRIVER1); // Amplituda napeti pri zrychleni (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_KVAL_DEC, 255, DRIVER1); // Amplituda napeti pri zpomaleni (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_OCD_TH, 31, DRIVER1);   // Max. proud 1562 mA
  WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER1);


  WriteDriverCommand(POWERSTEP01_ALARM_EN, 0b10000110, DRIVER2);      // Alarmy: thermal, command error
  WriteDriverCommand(POWERSTEP01_STEP_MODE, 0b00000111, DRIVER2);   // BUSY output, voltage mode, 128 microsteps
  WriteDriverCommand(POWERSTEP01_CONFIG, 0x0000, DRIVER2);          // 19.5kHz PWM, 7.5V VCC/UVLO, nestop pri nadproudu, bez kompenzace BEMF, SW vyvola HardStop, interni oscilator 16 MHz
  WriteDriverCommand(POWERSTEP01_MIN_SPEED, 7032, DRIVER2); // Nizkorychlostni optimalizace pod 700 kroku/s
  WriteDriverCommand(POWERSTEP01_FS_SPD, 1023, DRIVER2); // Neprepinat z mikrostep na fullstep pri prekroceni limitni rychlosti
  WriteDriverCommand(POWERSTEP01_KVAL_HOLD, 255, DRIVER2); // Amplituda napeti v zabrzdenem stavu (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_KVAL_RUN, 255, DRIVER2); // Amplituda napeti v chodu (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_KVAL_ACC, 255, DRIVER2); // Amplituda napeti pri zrychleni (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_KVAL_DEC, 255, DRIVER2); // Amplituda napeti pri zpomaleni (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_OCD_TH, 31, DRIVER2);   // Max. proud 1562 mA
  WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER2);


  WriteDriverCommand(POWERSTEP01_ALARM_EN, 0b10000110, DRIVER3);      // Alarmy: thermal, command error
  WriteDriverCommand(POWERSTEP01_STEP_MODE, 0b00000111, DRIVER3);   // BUSY output, voltage mode, 128 microsteps
  WriteDriverCommand(POWERSTEP01_CONFIG, 0x0000, DRIVER3);          // 19.5kHz PWM, 7.5V VCC/UVLO, nestop pri nadproudu, bez kompenzace BEMF, SW vyvola HardStop, interni oscilator 16 MHz
```

```
  WriteDriverCommand(POWERSTEP01_MIN_SPEED, 7032, DRIVER3); // Nizkorychlostni optimalizace pod 700 kroku/s
  WriteDriverCommand(POWERSTEP01_FS_SPD, 1023, DRIVER3); // Neprepinat z mikrostep na fullstep pri prekroceni limitni rychlosti
  WriteDriverCommand(POWERSTEP01_KVAL_HOLD, 255, DRIVER3); // Amplituda napeti v zabrzdenem stavu (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_KVAL_RUN, 255, DRIVER3); // Amplituda napeti v chodu (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_KVAL_ACC, 255, DRIVER3); // Amplituda napeti pri zrychleni (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_KVAL_DEC, 255, DRIVER3); // Amplituda napeti pri zpomaleni (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_OCD_TH, 31, DRIVER3);   // Max. proud 1562 mA
  WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER3);

  WriteDriverCommand(POWERSTEP01_ALARM_EN, 0b10000110, DRIVER4);      // Alarmy: thermal, command error
  WriteDriverCommand(POWERSTEP01_STEP_MODE, 0b00000111, DRIVER4);   // BUSY output, voltage mode, 128 microsteps
  WriteDriverCommand(POWERSTEP01_CONFIG, 0x0000, DRIVER4);          // 19.5kHz PWM, 7.5V VCC/UVLO, nestop pri nadproudu, bez kompenzace BEMF, SW vyvola HardStop, interni oscilator 16 MHz
  WriteDriverCommand(POWERSTEP01_MIN_SPEED, 7032, DRIVER4); // Nizkorychlostni optimalizace pod 700 kroku/s
  WriteDriverCommand(POWERSTEP01_FS_SPD, 1023, DRIVER4); // Neprepinat z mikrostep na fullstep pri prekroceni limitni rychlosti
  WriteDriverCommand(POWERSTEP01_KVAL_HOLD, 255, DRIVER4); // Amplituda napeti v zabrzdenem stavu (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_KVAL_RUN, 255, DRIVER4); // Amplituda napeti v chodu (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_KVAL_ACC, 255, DRIVER4); // Amplituda napeti pri zrychleni (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_KVAL_DEC, 255, DRIVER4); // Amplituda napeti pri zpomaleni (cca 1.5A/12V napajeni)
  WriteDriverCommand(POWERSTEP01_OCD_TH, 31, DRIVER4);   // Max. proud 1562 mA
  WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER4);

}
```

```
/********************************** HLAVNI PROGRAM  ********************************************** /

int main()
{
 CS1=1; CS2=1; CS3=1; CS4=1; RST=1; STCK=0;

  pc.printf("\r\nSTART PROGRAMU\r\n");

  spi.format(8,3);  // spi PowerStep01
  spi.frequency(1000000);

  i2c.frequency(100000);


  pc.printf("\r\nSTART PROGRAMU\r\n");
  wait_ms(10);
  SettingDrivers();
   Nastav_Gyro();

   ts.Init(lcd.GetXSize(), lcd.GetYSize());
//   ts_check.attach(&ts_check_fun, 0.1);

  while(1)
  {
    switch(obrazovka)
    {
      case 1: UvodniObrazovka();
         break;
      case 2: Manual();
         break;
      case 3: Homing();
         break;
      case 4: Uceni();
         break;
      case 5: RemoteControl();
         break;
      case 6: Program1();
         break;
      case 7: Program2();
         break;
      case 8: Program3();
         break;
      case 9: Program4();
         break;
      case 10: Program5();
         break;
      case 11: Program6();
         break;
    }
  }
```

```
}

/*****************************************************************************************************/
void Button(int pozice_x, int pozice_y, int sirka, int vyska,unsigned long barva, unsigned long ramecek)
{
  lcd.SetTextColor(barva);
  lcd.FillRect(pozice_x,pozice_y,sirka,vyska);
  lcd.SetTextColor(ramecek); lcd.DrawRect(pozice_x,pozice_y,sirka,vyska);
}

/*****************************************************************************************************/
void Text(int poz_x, int poz_y,int font,unsigned long barva_textu,unsigned long barva_pozadi,char *text)
{
 lcd.SetTextColor(barva_textu);
 lcd.SetBackColor(barva_pozadi);

 if(font==12) {lcd.SetFont(&Font12);}
 else if(font==16) {lcd.SetFont(&Font16);}
 else if(font==20) {lcd.SetFont(&Font20);}
 else if(font==24) {lcd.SetFont(&Font24);}
 else          {lcd.SetFont(&Font8);}
 lcd.DisplayStringAt(poz_x,poz_y, (uint8_t *)text, LEFT_MODE);
}

/*****************************************************************************************************/
int NactiDotek()
{
  ts.GetState(&TS_State);
  if(TS_State.touchDetected > 0)
  {
     x = TS_State.touchX[0];
     y = TS_State.touchY[0];
  }
  return TS_State.touchDetected;
}

/*****************************************************************************************************/
void Cislo(int poz_x,int poz_y, int font,unsigned long barva_textu,unsigned long barva_pozadi,long cislo)
{
   lcd.SetTextColor(barva_textu);
   lcd.SetBackColor(barva_pozadi);
   sprintf((char*)text, "%i", cislo);
   if(font==12) {lcd.SetFont(&Font12);}
   else if(font==16) {lcd.SetFont(&Font16);}
   else if(font==20) {lcd.SetFont(&Font20);}
   else if(font==24) {lcd.SetFont(&Font24);}
   else          {lcd.SetFont(&Font8);}
   lcd.DisplayStringAt(poz_x,poz_y,(uint8_t *)&text, LEFT_MODE);

}
```

```
/*****************************************************************************************************/
void Nastav_Gyro()
{

  I2C_cmd[0] = 25; I2C_cmd[1] = 67; I2C_cmd[1] = 4; i2c.write(0xD0, I2C_cmd, 3);   // Registry 25 a 26: Rychlost 15 Hz, filtr bandwidth 21 Hz
  I2C_cmd[0] = 107; I2C_cmd[1] = 0; i2c.write(0xD0, I2C_cmd, 2);    // PowerManagemenent register - zapnuti mereni
  //default: rozsah gyro +/-250o/s, acc +/- 2g, no interrupts, no FIFO

}

/*****************************************************************************************************/
void Precti_Gyro()
{

  I2C_cmd[0] = 59; i2c.write(0xD0, I2C_cmd, 1);   // Nastaveni cteni na adresu 59
  i2c.read(0xD0, MPU6050, 14);


  ACC_X = (int)(MPU6050[0] * 256) + (int) MPU6050[1];
  ACC_Y = (int)(MPU6050[2] * 256) + (int) MPU6050[3];
  ACC_Z = (int)(MPU6050[4] * 256) + (int) MPU6050[5];
  TEMP  = (int)(MPU6050[6] * 256) + (int) MPU6050[7];
  Gyro_X= (int)(MPU6050[8] * 256) + (int) MPU6050[9];
  Gyro_Y= (int)(MPU6050[10] * 256) + (int) MPU6050[11];
  Gyro_Z= (int)(MPU6050[12] * 256) + (int) MPU6050[13];

  if(ACC_X > 32767) ACC_X = ACC_X - 65536;
  if(ACC_Y > 32767) ACC_Y = ACC_Y - 65536;
  if(ACC_Z > 32767) ACC_Z = ACC_Z - 65536;
  if(TEMP > 32767) TEMP = TEMP - 65536;
  if(Gyro_X > 32767) Gyro_X = Gyro_X - 65536;
  if(Gyro_Y > 32767) Gyro_Y = Gyro_Y - 65536;
  if(Gyro_Z > 32767) Gyro_Z = Gyro_Z - 65536;

  ACC_X = (ACC_X * 2000) / 32768;
  ACC_Y = (ACC_Y * 2000) / 32768;
  ACC_Z = (ACC_Z * 2000) / 32768;
  Gyro_X = (Gyro_X * 250) / 32768;
  Gyro_Y = (Gyro_Y * 250) / 32768;
  Gyro_Z = (Gyro_Z * 250) / 32768;
  TEMP = (TEMP * 1000) / 340 + 36530;

  Ax = (double)ACC_X / 1000.0; Ax2 = Ax * Ax;
  Ay = (double)ACC_Y / 1000.0; Ay2 = Ay * Ay;
  Az = (double)ACC_Z / 1000.0; Az2 = Az * Az;

  Rotace = 180.0 * atan(Ax/sqrt(Ay2 + Az2))/M_PI;
  Naklon = -180.0 * atan(Ay/sqrt(Ax2 + Az2))/M_PI;
}
```

```
/*****************************************************************************************************/
void VytiskniGyro(int poz_x, int poz_y,int font,unsigned long barva_textu,unsigned long barva_pozadi)
{

  lcd.ClearStringLine(14);

  lcd.SetTextColor(barva_textu);
  lcd.SetBackColor(barva_pozadi);
  if(font==12) {lcd.SetFont(&Font12);}
  else if(font==16) {lcd.SetFont(&Font16);}
  else if(font==20) {lcd.SetFont(&Font20);}
  else if(font==24) {lcd.SetFont(&Font24);}
  else          {lcd.SetFont(&Font8);}


  sprintf((char*)text,  "GYRO:  x%d      ",  Gyro_X);lcd.DisplayStringAt(10,LINE(14),(uint8_t  *)&text,
LEFT_MODE);
  sprintf((char*)text, "y%d  ", Gyro_Y);lcd.DisplayStringAt(135,LINE(14),(uint8_t *)&text, LEFT_MODE);
  sprintf((char*)text, "z%d  ", Gyro_Z);lcd.DisplayStringAt(185,LINE(14),(uint8_t *)&text, LEFT_MODE);
  sprintf((char*)text,  "nkl:%d       ",(long)Naklon);lcd.DisplayStringAt(295,LINE(14),(uint8_t  *)&text,
LEFT_MODE);
  sprintf((char*)text,  "rot:%d       ",(long)Rotace);lcd.DisplayStringAt(400,LINE(14),(uint8_t  *)&text,
LEFT_MODE);


    lcd.ClearStringLine(15);
  sprintf((char*)text, "x%d  y%d  z%d", ACC_X,ACC_Y,ACC_Z);
  lcd.DisplayStringAt(poz_x,LINE(15),(uint8_t *)&text, LEFT_MODE);
}

/*****************************************************************************************************/
void VytiskniPozice(int poz_x, int poz_y,int font,unsigned long barva_textu,unsigned long barva_pozadi)
{
  POS1 = ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
  POS2 = ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
  POS3 = ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
  POS4 = ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
  lcd.ClearStringLine(13);

  if(font==12) {lcd.SetFont(&Font12);}
  else if(font==16) {lcd.SetFont(&Font16);}
  else if(font==20) {lcd.SetFont(&Font20);}
  else if(font==24) {lcd.SetFont(&Font24);}
  else          {lcd.SetFont(&Font8);}
  Cislo(10,LINE(13),16,LCD_COLOR_WHITE,LCD_COLOR_BLUE, POS1);
  Cislo(104,LINE(13),16,LCD_COLOR_WHITE,LCD_COLOR_BLUE,POS2);
  Cislo(198,LINE(13),16,LCD_COLOR_WHITE,LCD_COLOR_BLUE,POS4);
  Cislo(292,LINE(13),16,LCD_COLOR_WHITE,LCD_COLOR_BLUE,POS3);
  Cislo(386,LINE(13),16,LCD_COLOR_WHITE,LCD_COLOR_BLUE,spd);
 }
```

```
/*****************************************************************************************************/
void UvodniObrazovka()
{
  lcd.Clear(barva_pozadi);
  lcd.SetBackColor(barva_pozadi);
  ts.Init(lcd.GetXSize(), lcd.GetYSize());
  Text(60, LINE(0),24,barva_textu_pozadi,barva_pozadi,"ROBOTICKA RUKA FRENGP");

  Button(10,38,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(20,70,16,barva_textu_tlacitko,barva_tlacitka,"MANUAL");
  Button(104,38,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(110,70,16,barva_textu_tlacitko,barva_tlacitka,"HOMING");
  Button(198,38,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(212,70,16,barva_textu_tlacitko,barva_tlacitka,"LEARN");
  Button(292,38,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(300,65,16,barva_textu_tlacitko,barva_tlacitka,"REMOTE");
Text(295,81,16,barva_textu_tlacitko,barva_tlacitka,"CONTROL");
  Button(386,38,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(420,70,16,barva_textu_tlacitko,barva_tlacitka,"1");

  Button(10,153,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(45,190,16,barva_textu_tlacitko, barva_tlacitka,"2");
  Button(104,153,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(134,190,16,barva_textu_tlacitko,barva_tlacitka,"3");
  Button(198,153,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(227,190,16,barva_textu_tlacitko,barva_tlacitka,"4");
  Button(292,153,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(330,190,16,barva_textu_tlacitko,barva_tlacitka,"5");
  Button(386,153,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(420,190,16,barva_textu_tlacitko,barva_tlacitka,"6");

  while(1)
  {
   if(NactiDotek()>0)
   {
     if(MANUAL)          {obrazovka=2;return;}
     else if(HOMING)     {obrazovka=3;return;}
     else if(UCENI)      {obrazovka=4;return;}
     else if(REMOTE_CONTROL){obrazovka=5;return;}
     else if(PROGRAM1)   {obrazovka=6;return;}
     else if(PROGRAM2)   {obrazovka=7;return;}
     else if(PROGRAM3)   {obrazovka=8;return;}
     else if(PROGRAM4)   {obrazovka=9;return;}
     else if(PROGRAM5)   {obrazovka=10;return;}
     else if(PROGRAM6)   {obrazovka=11;return;}
   }
  }
 }
```

```
/*****************************************************************************************************/
 void Manual()
 {
  wait_ms(1000);
  lcd.Clear(LCD_COLOR_BLUE);
  Text(20, LINE(0),16,barva_textu_pozadi,barva_pozadi,"ROTACE");
  Text(109,LINE(0),16,barva_textu_pozadi,barva_pozadi,"ZVEDANI");
  Text(208,LINE(0),16,barva_textu_pozadi,barva_pozadi,"RAMENO");
  Text(302,LINE(0),16,barva_textu_pozadi,barva_pozadi,"KLESTE");
  Text(386,LINE(0),16,barva_textu_pozadi,barva_pozadi,"RYCHLOST");

  Button(10,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(20,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"DOLEVA");
  Button(104,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(114,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"NAHORU");
  Button(198,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(207,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"NAHORU");
  Button(292,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(297,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"OTEVRIT");
  Button(386,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(420,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"+");

  Button(10,113,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(15,LINE(9),16,barva_textu_tlacitko, barva_tlacitka,"DOPRAVA");
  Button(104,113,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(114,LINE(9),16,barva_textu_tlacitko,barva_tlacitka,"DOLU");
  Button(198,113,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(207,LINE(9),16,barva_textu_tlacitko,barva_tlacitka,"DOLU");
  Button(292,113,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(302,LINE(9),16,barva_textu_tlacitko,barva_tlacitka,"ZAVRIT");
  Button(386,113,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(420,LINE(9),16,barva_textu_tlacitko,barva_tlacitka,"-");

   while(1)
   {
     Precti_Gyro();
     VytiskniPozice(10,LINE(13),16,LCD_COLOR_WHITE,LCD_COLOR_BLUE);
     VytiskniGyro(10,LINE(14),16,LCD_COLOR_WHITE, LCD_COLOR_BLUE);
     if(NactiDotek()>0)
     {
       if(M_RYCHLOST_PLUS)          {spd=spd+1000; if(spd>=999000) spd=999000;} // rychlost plus
       else if(M_RYCHLOST_MINUS)     {spd=spd-1000; if(spd<=0) spd=0;} // rychlost minus

       else  if(M_ROTACE_DOLEVA)             {WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRI-
VER1);} // jede doleva 1
       else  if(M_ROTACE_DOPRAVA)          {WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRI-
VER1);} // jede doprava 1

       else if(M_ZVEDANI_DOLU)        {WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);}
// jede dolu 2
       else  if(M_ZVEDANI_NAHORU)          {WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRI-
VER2);} // jede nahoru 2
```

```
        else  if(M_RAMENO_NAHORU)           {WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRI-
VER4);} // jede nahoru 4
        else if(M_RAMENO_DOLU)        {WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER4);}
// jede dolu 4

        else if(M_KLESTE_OTEVRIT)      {WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER3);}
// jede nahoru 4
        else if(M_KLESTE_ZAVRIT)       {WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER3);}
// jede dolu 4

        else if(M_POS1)             {WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER1);}   // vynu-
lovani pozic
        else if(M_POS2)             {WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER2);}   // vynu-
lovani pozic
        else if(M_POS3)             {WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER4);}   // vynu-
lovani pozic
        else if(M_POS4)             {WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER3);}   // vynu-
lovani pozic

        else if(x>386 && x<470 && y>198 && y<272)
        {
          obrazovka=1;return;
        }
       }

        else
        {
          WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
          WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
          WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
          WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
        }
      wait(0.1);
  }
 }

/*************************************************************************************************/
 void Homing()
 {

    lcd.ClearStringLine(1);
    lcd.ClearStringLine(2);
    lcd.ClearStringLine(13);
    lcd.ClearStringLine(14);
    lcd.ClearStringLine(15);
    lcd.Clear(LCD_COLOR_BLUE);
    Text(150,91,24,LCD_COLOR_WHITE,LCD_COLOR_BLUE,"Nulovani os...");
    Text(150,120,24,LCD_COLOR_WHITE,LCD_COLOR_BLUE,"0/4");
    WriteDriverCommand(POWERSTEP01_RUN_REV, 20000, DRIVER3);  //kleste
    wait(0.5);
    ACC_X=30;
```

```
   while(ACC_X>0)
   {
     Precti_Gyro();
     wait(0.1);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ,                0,                DRIVER3);
Text(150,120,24,LCD_COLOR_WHITE,LCD_COLOR_BLUE,"1/4");
   WriteDriverCommand(POWERSTEP01_RUN_REV, 20000, DRIVER4);  // rameno

   wait(0.5);
   ACC_X=1000;
   while(ACC_X>-50)
   {
     Precti_Gyro();
     wait(0.1);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ,                0,                DRIVER4);
Text(150,120,24,LCD_COLOR_WHITE,LCD_COLOR_BLUE,"2/4");
   WriteDriverCommand(POWERSTEP01_RUN_FWD, 20000, DRIVER2);

   wait(0.5);
   Gyro_X=-8;
   while(Gyro_X<-5)
   {
    Precti_Gyro();
    wait(0.1);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ,                0,                DRIVER2);
Text(150,120,24,LCD_COLOR_WHITE,LCD_COLOR_BLUE,"UZ TO SKORO JE");
   WriteDriverCommand(POWERSTEP01_RUN_REV, 20000, DRIVER1);
   wait(0.5);
   Gyro_Z=20;
   while(Gyro_Z>5)
   {
     Precti_Gyro();
     wait(0.1);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ,                0,                DRIVER1);
Text(150,120,24,LCD_COLOR_WHITE,LCD_COLOR_BLUE,"HOTOVO!!!       ");   wait(1);
   obrazovka=1;
}

/*****************************************************************************************************/
void RemoteControl()
{

    while(1)
  {

     if(BT.readable())    // BT module receives a signal
     {
       BTread[i] = BT.getc();  // read a character of the signal
```

```
        if(BTread[0] == 244) i++;
      }

      if(i==6)
      {
        WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
        WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
        WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
        WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
        pc.printf("%i %i %i %i %i %i\n\r", BTread[0], BTread[1], BTread[2], BTread[3], BTread[4], BTread[5]);
        if(BTread[0] == 244 && BTread[1] == 9 && BTread[2] == 3 && BTread[3] == 233 && BTread[4] == 100) {Homing();}
        if(BTread[0] == 244 && BTread[1] == 9 && BTread[2] == 3 && BTread[3] == 233 && BTread[4] == 10) {WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER1);}
        if(BTread[0] == 244 && BTread[1] == 9 && BTread[2] == 3 && BTread[3] == 233 && BTread[4] == 20) {WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER1);}
        if(BTread[0] == 244 && BTread[1] == 9 && BTread[2] == 3 && BTread[3] == 233 && BTread[4] == 30) {WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);}
        if(BTread[0] == 244 && BTread[1] == 9 && BTread[2] == 3 && BTread[3] == 233 && BTread[4] == 40) {WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER2);}
        if(BTread[0] == 244 && BTread[1] == 9 && BTread[2] == 3 && BTread[3] == 233 && BTread[4] == 50) {WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER3);}
        if(BTread[0] == 244 && BTread[1] == 9 && BTread[2] == 3 && BTread[3] == 233 && BTread[4] == 60) {WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER3);}
        if(BTread[0] == 244 && BTread[1] == 9 && BTread[2] == 3 && BTread[3] == 233 && BTread[4] == 70) {WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER4);}
        if(BTread[0] == 244 && BTread[1] == 9 && BTread[2] == 3 && BTread[3] == 233 && BTread[4] == 80) {WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER4);}
        if(BTread[0] == 244 && BTread[1] == 9 && BTread[2] == 3 && BTread[3] == 233 && BTread[4] == 0)  {WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
                                                                WriteDriver-
Command(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
                                                                WriteDriver-
Command(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
                                                                WriteDriver-
Command(POWERSTEP01_HARD_HIZ, 0, DRIVER4);}

        i = 0; // null for the next BT command
      }
    }
}

/*****************************************************************************************************/
void Uceni()
{
 Homing();
 step1=0;
 step2=0;
 step3=0;
 step4=0;
 krok=0;
```

```
 WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER1);
 WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER2);
 WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER3);
 WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER4);

 lcd.Clear(LCD_COLOR_BLUE);
  Text(20, LINE(0),16,barva_textu_pozadi,barva_pozadi,"ROTACE");
  Text(109,LINE(0),16,barva_textu_pozadi,barva_pozadi,"ZVEDANI");
  Text(208,LINE(0),16,barva_textu_pozadi,barva_pozadi,"RAMENO");
  Text(302,LINE(0),16,barva_textu_pozadi,barva_pozadi,"KLESTE");
  Text(386,LINE(0),16,barva_textu_pozadi,barva_pozadi,"RYCHLOST");

  Button(10,18,84,85,                                    barva_tlacitka,barva_textu_tlacitko);
Text(20,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"DOLEVA");
  Button(104,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(114,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"NAHORU");
  Button(198,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(207,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"NAHORU");
  Button(292,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(297,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"OTEVRIT");
  Button(386,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(420,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"+");

  Button(10,113,84,85,barva_tlacitka,barva_textu_tlacitko);      Text(15,LINE(9),16,barva_textu_tlacitko,
barva_tlacitka,"DOPRAVA");
  Button(104,113,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(114,LINE(9),16,barva_textu_tlacitko,barva_tlacitka,"DOLU");
  Button(198,113,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(207,LINE(9),16,barva_textu_tlacitko,barva_tlacitka,"DOLU");
  Button(292,113,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(302,LINE(9),16,barva_textu_tlacitko,barva_tlacitka,"ZAVRIT");
  Button(386,113,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(420,LINE(9),16,barva_textu_tlacitko,barva_tlacitka,"-");
  Button(10,225,366,40,LCD_COLOR_GREEN,LCD_COLOR_WHITE);
Text(170,LINE(15),16,barva_textu_tlacitko,LCD_COLOR_GREEN,"SAVE");
  Button(386,225,84,40,LCD_COLOR_GREEN,LCD_COLOR_WHITE);
Text(405,LINE(15),16,barva_textu_tlacitko,LCD_COLOR_GREEN,"EXIT");
 WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
        WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
        WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
        WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
  for(int k=0;k<=100;k++)
  {
    pozice1[k]=0;
    pozice2[k]=0;
    pozice3[k]=0;
    pozice4[k]=0;
  }
  while(1)
  {
    Precti_Gyro();
    VytiskniPozice(10,LINE(13),16,LCD_COLOR_WHITE,LCD_COLOR_BLUE);
```

```
    if(NactiDotek()>0)
    {

      if(M_RYCHLOST_PLUS)          {spd=spd+1000; if(spd>=999000) spd=999000;} // rychlost plus
      else if(M_RYCHLOST_MINUS)     {spd=spd-1000; if(spd<=0) spd=0;} // rychlost minus

      else  if(M_ROTACE_DOLEVA)             {WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRI-
VER1);} // jede doleva 1
      else  if(M_ROTACE_DOPRAVA)          {WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRI-
VER1);} // jede doprava 1

      else if(M_ZVEDANI_DOLU)        {WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);}
// jede dolu 2
      else  if(M_ZVEDANI_NAHORU)          {WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRI-
VER2);} // jede nahoru 2

      else  if(M_RAMENO_NAHORU)           {WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRI-
VER4);} // jede nahoru 4
      else if(M_RAMENO_DOLU)        {WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER4);}
// jede dolu 4

      else if(M_KLESTE_OTEVRIT)     {WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER3);}
// jede nahoru 4
      else if(M_KLESTE_ZAVRIT)      {WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER3);}
// jede dolu 4

      else if(SAVE)               {
                                  krok=krok+1;
                                  pozice1[krok]=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
                                  pozice2[krok]=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
                                  pozice4[krok]=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
                                  pozice3[krok]=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
                          }

      else if(EXIT)
      {
        Homing();
        lcd.Clear(LCD_COLOR_GREEN);
        Text(180,130,24,LCD_COLOR_BLACK,LCD_COLOR_GREEN,"WORKING ...");
        pozice1[0]=0;
        WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER1);
        POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
        WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER4);
        WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER3);
      for(int i=1;i<=krok;i++)
      {
        POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
        if(pozice1[i]<pozice1[i-1])
        {
          while(POS1>=pozice1[i])
          {
```

```
    POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER1);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
}
if(pozice1[i]>pozice1[i-1])
{
   while(POS1<=pozice1[i])
   {
    POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
    WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER1);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
}

POS2=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
if(pozice2[i]>pozice2[i-1] && i==1)
{
   while(POS2<=pozice2[i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
   }
   while(POS2>=pozice2[i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
   }

   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
if(pozice2[i]<pozice2[i-1])
{
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
if(pozice2[i]>pozice2[i-1] && i!=1)
{
   while(POS2<=pozice2[i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER2);
   }

   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);

POS3=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
if(pozice3[i]<pozice3[i-1])
{
   while(POS3>=pozice3[i])
```

```
            {
             POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
             WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER3);
            }
           WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
          }
          if(pozice3[i]>pozice3[i-1])
          {
           while(POS3<=pozice3[i])
           {
            POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
            WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER3);
           }
           WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
          }

          POS4=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
          if(pozice4[i]<pozice4[i-1])
          {
           while(POS4>=pozice4[i])
           {
            POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
            WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER4);
           }
           WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
          }

          if(pozice4[i]>pozice4[i-1])
          {
           POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
           while(POS4<=pozice4[i])
           {
            POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
            WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER4);
           }
           WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
          }
        }
        Ukladani();return;
        }
       }
       else
       {
        WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
        WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
        WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
        WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
       }
       wait(0.1);
  }
}
```

```
/*****************************************************************************************************/
void Ukladani()
{

   lcd.Clear(LCD_COLOR_BLUE);
   lcd.SetBackColor(LCD_COLOR_BLUE);
   ts.Init(lcd.GetXSize(), lcd.GetYSize());

  lcd.Clear(LCD_COLOR_BLUE);

   Text(140,LINE(0),16,barva_textu_pozadi,barva_pozadi,"CHOOSE THE PROGRAM: ");


   Button(10,18,84,85,                                                LCD_COLOR_RED,LCD_COLOR_WHITE);
Text(20,LINE(3),16,LCD_COLOR_WHITE,LCD_COLOR_RED,"DELETE");
   Button(104,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(140,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"1");
   Button(198,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(230,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"2");
   Button(292,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(330,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"3");
   Button(386,18,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(420,LINE(3),16,barva_textu_tlacitko,barva_tlacitka,"4");

   Button(10,113,84,85,barva_tlacitka,barva_textu_tlacitko);        Text(55,LINE(9),16,barva_textu_tlacitko,
barva_tlacitka,"5");
   Button(104,113,84,85,barva_tlacitka,barva_textu_tlacitko);
Text(139,LINE(9),16,barva_textu_tlacitko,barva_tlacitka,"6");


 while(1)
 {
  if(NactiDotek()>0)
  {
    if(M_RYCHLOST_PLUS)                                                {program4_step=krok;for(int
f=1;f<=krok;f++){program4[1][f]=pozice1[f];program4[2][f]=pozice2[f];program4[3][f]=pozice3[f];progr
am4[4][f]=pozice4[f];} obrazovka=1;return;}
    else       if(M_ROTACE_DOLEVA)                                     {program5_step=krok;for(int
f=1;f<=krok;f++){program5[1][f]=pozice1[f];program5[2][f]=pozice2[f];program5[3][f]=pozice3[f];progr
am5[4][f]=pozice4[f];} obrazovka=1;return;}

    else if(M_ROTACE_DOPRAVA)     {obrazovka=1;return;}
    else       if(M_ZVEDANI_NAHORU)                                    {program1_step=krok;for(int
f=1;f<=krok;f++){program1[1][f]=pozice1[f];program1[2][f]=pozice2[f];program1[3][f]=pozice3[f];progr
am1[4][f]=pozice4[f];} obrazovka=1;return;}
    else       if(M_RAMENO_NAHORU)                                     {program2_step=krok;for(int
f=1;f<=krok;f++){program2[1][f]=pozice1[f];program2[2][f]=pozice2[f];program2[3][f]=pozice3[f];progr
am2[4][f]=pozice4[f];} obrazovka=1;return;}
    else       if(M_KLESTE_OTEVRIT)                                    {program3_step=krok;for(int
f=1;f<=krok;f++){program3[1][f]=pozice1[f];program3[2][f]=pozice2[f];program3[3][f]=pozice3[f];progr
am3[4][f]=pozice4[f];} obrazovka=1;return;}
```

```
    else        if(M_ZVEDANI_DOLU)                                                {program6_step=krok;for(int
f=1;f<=krok;f++){program6[1][f]=pozice1[f];program6[2][f]=pozice2[f];program6[3][f]=pozice3[f];progr
am6[4][f]=pozice4[f];} obrazovka=1;return;}


  }
 }
}

/*************************************************************************************************************/
void Program1()
{
   Homing();
   lcd.Clear(LCD_COLOR_GREEN);
   Text(180,130,24,LCD_COLOR_BLACK,LCD_COLOR_GREEN,"WORKING ...");
   pozice1[0]=0;
   WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER1);
   POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
   WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER4);
   WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER3);
   for(int i=1;i<=program1_step;i++)
   {
      POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
      if(program1[1][i]<program1[1][i-1])
      {
         while(POS1>=program1[1][i])
         {
          POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
          WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER1);
         }
         WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
      }
      if(program1[1][i]>program1[1][i-1])
      {
         while(POS1<=program1[1][i])
         {
          POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
          WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER1);
         }
         WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
      }

      POS2=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
      if(program1[2][i]>program1[2][i-1] && i==1)
      {
         while(POS2<=program1[2][i])
         {
          POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
          WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
         }
         while(POS2>=program1[2][i])
         {
```

```
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
   }

   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
if(program1[2][i]<program1[2][i])
{
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
if(program1[2][i]>program1[2][i-1] && i!=1)
{
   while(POS2<=program1[2][i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER2);
   }

   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);

POS3=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
if(program1[3][i]<program1[3][i-1])
{
   while(POS3>=program1[3][i])
   {
    POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER3);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
}
if(program1[3][i]>program1[3][i-1])
{
   while(POS3<=program1[3][i-1])
   {
    POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
    WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER3);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
}

POS4=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
if(program1[4][i]<program1[4][i-1])
{
   while(POS4>=program1[4][i])
   {
    POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER4);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
```

```
    }

    if(program1[4][i]>program1[4][i-1])
    {
      POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
      while(POS4<=program1[4][i])
      {
       POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
       WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER4);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
    }
  }
  UvodniObrazovka();return;

}

/*****************************************************************************************************/
void Program2()
{
  Homing();
  lcd.Clear(LCD_COLOR_GREEN);
  Text(180,130,24,LCD_COLOR_BLACK,LCD_COLOR_GREEN,"WORKING ...");
  pozice1[0]=0;
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER1);
  POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER4);
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER3);
  for(int i=1;i<=program2_step;i++)
  {
    POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
    if(program2[1][i]<program2[1][i-1])
    {
      while(POS1>=program2[1][i])
      {
       POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
       WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER1);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
    }
    if(program2[1][i]>program2[1][i-1])
    {
      while(POS1<=program2[1][i])
      {
       POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
       WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER1);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
    }

    POS2=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    if(program2[2][i]>program2[2][i-1] && i==1)
```

```
{
  while(POS2<=program2[2][i])
  {
   POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
   WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
  }
  while(POS2>=program2[2][i])
  {
   POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
   WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
  }

  WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
if(program2[2][i]<program2[2][i])
{
  WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
if(program2[2][i]>program2[2][i-1] && i!=1)
{
  while(POS2<=program2[2][i])
  {
   POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
   WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER2);
  }

  WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);

POS3=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
if(program2[3][i]<program2[3][i-1])
{
  while(POS3>=program2[3][i])
  {
   POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
   WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER3);
  }
  WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
}
if(program2[3][i]>program2[3][i-1])
{
  while(POS3<=program2[3][i-1])
  {
   POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
   WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER3);
  }
  WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
}

POS4=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
```

```
    if(program2[4][i]<program2[4][i-1])
    {
      while(POS4>=program2[4][i])
      {
       POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
       WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER4);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
    }

    if(program2[4][i]>program2[4][i-1])
    {
      POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
      while(POS4<=program2[4][i])
      {
       POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
       WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER4);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
    }
  }
  UvodniObrazovka();return;
}

/*****************************************************************************************************/
void Program3()
{
  Homing();
  lcd.Clear(LCD_COLOR_GREEN);
  Text(180,130,24,LCD_COLOR_BLACK,LCD_COLOR_GREEN,"WORKING ...");
  pozice1[0]=0;
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER1);
  POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER4);
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER3);
  for(int i=1;i<=program3_step;i++)
  {
    POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
    if(program3[1][i]<program3[1][i-1])
    {
      while(POS1>=program3[1][i])
      {
       POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
       WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER1);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
    }
    if(program3[1][i]>program3[1][i-1])
    {
      while(POS1<=program3[1][i])
      {
       POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
```

```
    WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER1);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
}

POS2=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
if(program3[2][i]>program3[2][i-1] && i==1)
{
   while(POS2<=program3[2][i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
   }
   while(POS2>=program3[2][i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
   }

   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
if(program3[2][i]<program3[2][i])
{
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
if(program3[2][i]>program3[2][i-1] && i!=1)
{
   while(POS2<=program3[2][i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER2);
   }

   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);

POS3=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
if(program3[3][i]<program3[3][i-1])
{
   while(POS3>=program3[3][i])
   {
    POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER3);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
}
if(program3[3][i]>program3[3][i-1])
{
   while(POS3<=program3[3][i-1])
   {
```

```
        POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
        WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER3);
       }
       WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
     }

     POS4=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
     if(program3[4][i]<program3[4][i-1])
     {
       while(POS4>=program3[4][i])
       {
        POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
        WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER4);
       }
       WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
     }

     if(program3[4][i]>program3[4][i-1])
     {
       POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
       while(POS4<=program3[4][i])
       {
        POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
        WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER4);
       }
       WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
     }
   }
   UvodniObrazovka();return;
}

/*****************************************************************************************************/
void Program4()
{
   Homing();
   lcd.Clear(LCD_COLOR_GREEN);
   Text(180,130,24,LCD_COLOR_BLACK,LCD_COLOR_GREEN,"WORKING ...");
   pozice1[0]=0;
   WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER1);
   POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
   WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER4);
   WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER3);
   for(int i=1;i<=program4_step;i++)
   {
     POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
     if(program4[1][i]<program4[1][i-1])
     {
       while(POS1>=program4[1][i])
       {
        POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
        WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER1);
       }
```

```
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
}
if(program4[1][i]>program4[1][i-1])
{
   while(POS1<=program4[1][i])
   {
    POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
    WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER1);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
}

POS2=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
if(program4[2][i]>program4[2][i-1] && i==1)
{
   while(POS2<=program4[2][i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
   }
   while(POS2>=program4[2][i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
   }

   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
if(program4[2][i]<program4[2][i])
{
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
if(program4[2][i]>program4[2][i-1] && i!=1)
{
   while(POS2<=program4[2][i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER2);
   }

   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);

POS3=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
if(program4[3][i]<program4[3][i-1])
{
   while(POS3>=program4[3][i])
   {
    POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER3);
```

```
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
    }
    if(program4[3][i]>program4[3][i-1])
    {
      while(POS3<=program4[3][i-1])
      {
       POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
       WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER3);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
    }

    POS4=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
    if(program4[4][i]<program4[4][i-1])
    {
      while(POS4>=program4[4][i])
      {
       POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
       WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER4);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
    }

    if(program4[4][i]>program4[4][i-1])
    {
      POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
      while(POS4<=program4[4][i])
      {
       POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
       WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER4);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
    }
  }
  UvodniObrazovka();return;
}

/*****************************************************************************************************/
void Program5()
{
  Homing();
  lcd.Clear(LCD_COLOR_GREEN);
  Text(180,130,24,LCD_COLOR_BLACK,LCD_COLOR_GREEN,"WORKING ...");
  pozice1[0]=0;
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER1);
  POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER4);
  WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER3);
  for(int i=1;i<=program5_step;i++)
  {
    POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
```

```
if(program5[1][i]<program5[1][i-1])
{
   while(POS1>=program5[1][i])
   {
    POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER1);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
}
if(program5[1][i]>program5[1][i-1])
{
   while(POS1<=program5[1][i])
   {
    POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
    WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER1);
   }
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
}

POS2=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
if(program5[2][i]>program5[2][i-1] && i==1)
{
   while(POS2<=program5[2][i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
   }
   while(POS2>=program5[2][i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
   }

   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
if(program5[2][i]<program5[2][i])
{
   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
if(program5[2][i]>program5[2][i-1] && i!=1)
{
   while(POS2<=program5[2][i])
   {
    POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
    WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER2);
   }

   WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
}
WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
```

```
    POS3=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
    if(program5[3][i]<program5[3][i-1])
    {
      while(POS3>=program5[3][i])
      {
       POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
       WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER3);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
    }
    if(program5[3][i]>program5[3][i-1])
    {
      while(POS3<=program5[3][i-1])
      {
       POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
       WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER3);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
    }

    POS4=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
    if(program5[4][i]<program5[4][i-1])
    {
      while(POS4>=program5[4][i])
      {
       POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
       WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER4);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
    }

    if(program5[4][i]>program5[4][i-1])
    {
      POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
      while(POS4<=program5[4][i])
      {
       POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
       WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER4);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
    }
  }
  UvodniObrazovka();return;
}

/*****************************************************************************************************/
void Program6()
{
  Homing();
  lcd.Clear(LCD_COLOR_GREEN);
  Text(180,130,24,LCD_COLOR_BLACK,LCD_COLOR_GREEN,"WORKING ...");
  pozice1[0]=0;
```

```
WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER1);
POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER4);
WriteDriverCommand(POWERSTEP01_ABS_POS, 0, DRIVER3);
for(int i=1;i<=program6_step;i++)
{
  POS1=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
  if(program6[1][i]<program6[1][i-1])
  {
    while(POS1>=program6[1][i])
    {
     POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
     WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER1);
    }
    WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
  }
  if(program6[1][i]>program6[1][i-1])
  {
    while(POS1<=program6[1][i])
    {
     POS1= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER1);
     WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER1);
    }
    WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER1);
  }

  POS2=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
  if(program6[2][i]>program6[2][i-1] && i==1)
  {
    while(POS2<=program6[2][i])
    {
     POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
     WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
    }
    while(POS2>=program6[2][i])
    {
     POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
     WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER2);
    }

    WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
  }
  if(program6[2][i]<program6[2][i])
  {
    WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
  }
  WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
  if(program6[2][i]>program6[2][i-1] && i!=1)
  {
    while(POS2<=program6[2][i])
    {
     POS2= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER2);
```

```
        WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER2);
      }

      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);
    }
    WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER2);

    POS3=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
    if(program6[3][i]<program6[3][i-1])
    {
      while(POS3>=program6[3][i])
      {
       POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
       WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER3);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
    }
    if(program6[3][i]>program6[3][i-1])
    {
      while(POS3<=program6[3][i-1])
      {
       POS3= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER3);
       WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER3);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER3);
    }

    POS4=ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
    if(program6[4][i]<program6[4][i-1])
    {
      while(POS4>=program6[4][i])
      {
       POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
       WriteDriverCommand(POWERSTEP01_RUN_REV, spd, DRIVER4);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
    }

    if(program6[4][i]>program6[4][i-1])
    {
      POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
      while(POS4<=program6[4][i])
      {
       POS4= ReadDriverRegister(POWERSTEP01_ABS_POS, DRIVER4);
       WriteDriverCommand(POWERSTEP01_RUN_FWD, spd, DRIVER4);
      }
      WriteDriverCommand(POWERSTEP01_HARD_HIZ, 0, DRIVER4);
    }
  }
  UvodniObrazovka();return;
}
```

6 Software pro mobilní aplikaci

K napsání software pro mobilní aplikaci jsem si vybrala programovací prostředí Pocket Code pro OS Android[23], protože využívá velmi jednoduchý programovací jazyk, snadno pochopitelný i začátečníkům a umožňuje pomocí Bluetooth komunikace ovládat i externí elektronická zařízení, jako například Arduino, Raspberry Pi, Lego NXT nebo Lego EV3. Program Pocket Code je zdarma ke stažení na Google Play[24].

Program Pocket Code sice nemá připravené komunikační rozhraní s mikrokontroléry ARM, ale podařilo se mi pomocí desky Arduino UNO, prostředí Arduino IDE s funkcí Serial Monitor a s Bluetooth modulem HC-05 dekódovat komunikační protokol Bluetooth mezi zařízením s Pocket Codem a Arduinem a v upravené podobě ho použít k dálkovému ovládání mikrokontrolérů ARM naprogramových v prostředí Mbed.

6.1 Bluetooth komunikační protokol Pocket Code

Pro jednosměrné dálkové ovládání (pouze vysílání) Mbed kompatibilních desek z Pocket Code je potřeba:

1. Použít Bluetooth přijímač (jako třeba HC-05 nebo HC-06) a nastavit ho na přenosovou rychlost 115200 baud a datový formát 8N1.
2. Připojit Bluetooth přijímač k UART portu na Mbed desce (TXD na RX; RXD na TX) a napájet ho z Mbed desky.
3. V nastavení Pocket Code zapnout plugin pro Arduino bloky a spárovat telefon s Bluetooth přijímačem.
4. Pro komunikaci s Mbed deskou použít v Pocket Code příkazy Arduino PWM. Jako dva proměnné parametry příkazů se může použít číslo pinu (0 - 13) a hodnota PWM (0 - 16383). Umožňuje to poslat na desku kombinaci 14 příkazů a až 16384 hodnot těchto příkazů nebo až 229376 jednoúčelových příkazů (14*16384).
5. Příkaz vysílaný přes Bluetooth do mikrokontroléru bude tvořen 6 bajty:

   Byte0 = 0xF4 (fixní startovací bajt)

   Byte1 = Číslo pinu Arduina (0 - 13)

   Byte2 = 0x03 (fixní hodnota)

   Byte3 = 0xE0 + číslo pinu (např. 0xED pro pin 13)

   Byte4 = (LSB) + Byte5(MSB) = hodnota PWM vyjádřena v dvou 7bitových číslech (0 - 127). Například hodnota 10000 bude poslána jako Byte5(MSB) = 78 (=10000/128) a Byte4 (LSB) = 16 (= 10000 - 78*128).
6. V programu v Mbed použít funkce BT.readable() a BT.getc() k dekódování příkazů poslaných přes Bluetooth z Pocket Code.

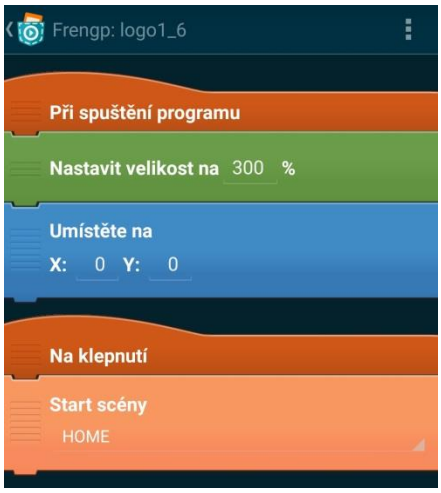
Pro ovládání Robotické ruky FRENGP byl použit Byte1 = 9 (pin Arduino číslo 9) a jednotlivé příkazy byly posílány v Byte4, dle tabulky 4:

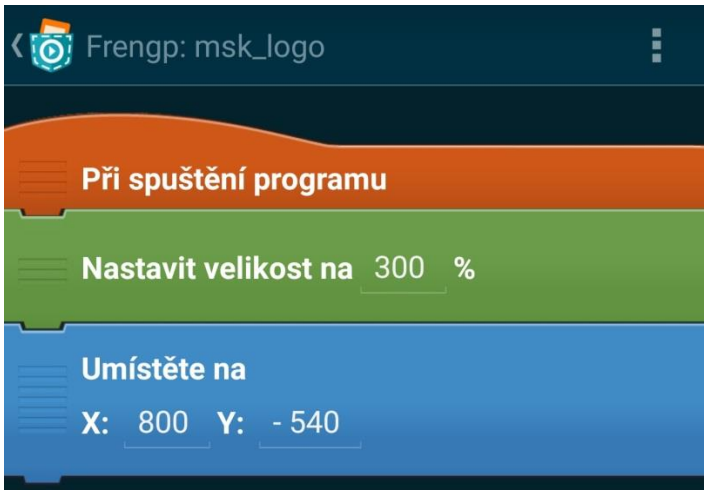
Tabulka 4: Tabulka příkazů Bluetooth komunikace

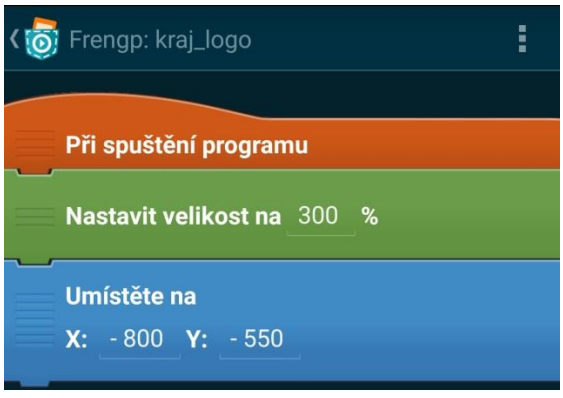
Hodnota Byte4	Příkaz
0	zastavení všech motorů
10	otáčení základnou doprava
20	otáčení základnou doleva
30	posun páteře dolů
40	posun páteře nahoru
50	otevírání kleští
60	zavírání kleští
70	posun ramene nahoru
80	posun ramene dolů
100	start funkce Homing

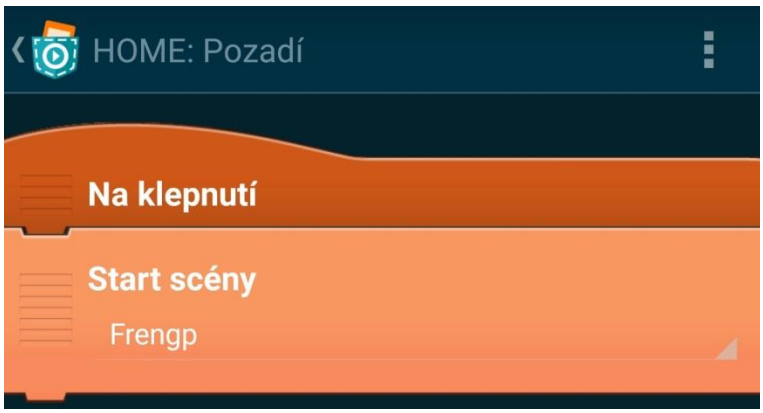
6.2 Výpis programu Pocket Code

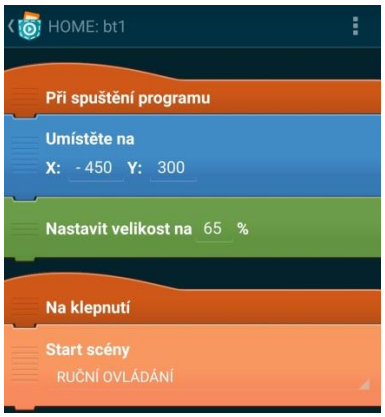
Kompletní verzi programu ovládání Robotické ruky FRENGP lze stáhnout ze zdroje[25].



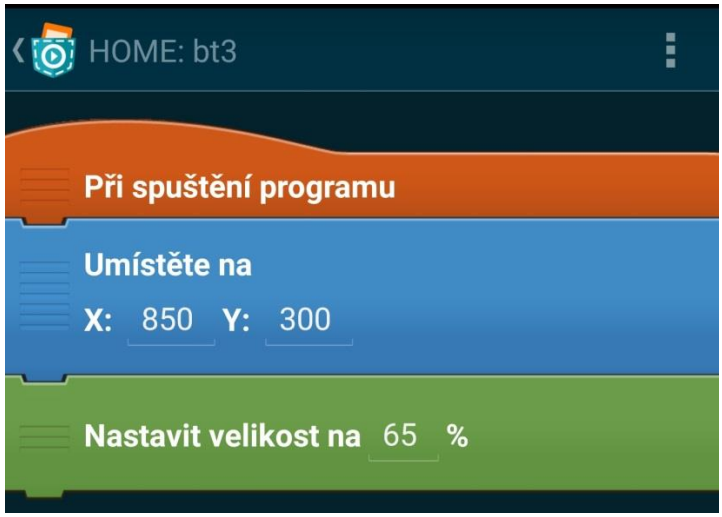


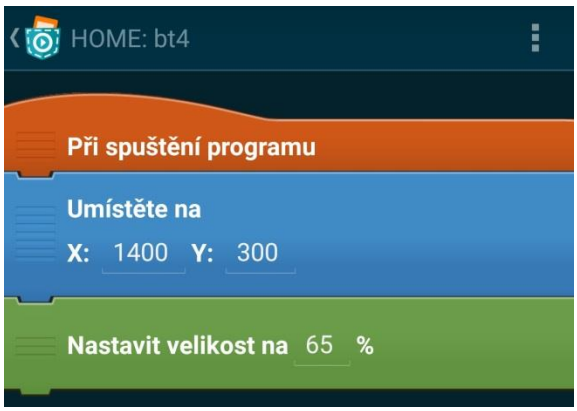



HOME: Pozadí
Na klepnutí
Start scény
Frengp

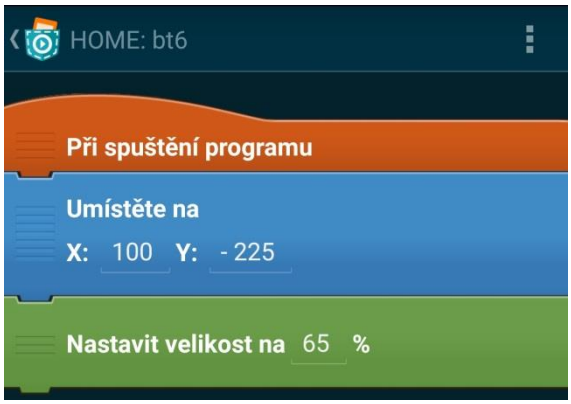

HOME: bt1
Při spuštění programu
Umístěte na
X: -450 Y: 300
Nastavit velikost na 65 %
Na klepnutí
Start scény
RUČNÍ OVLÁDÁNÍ

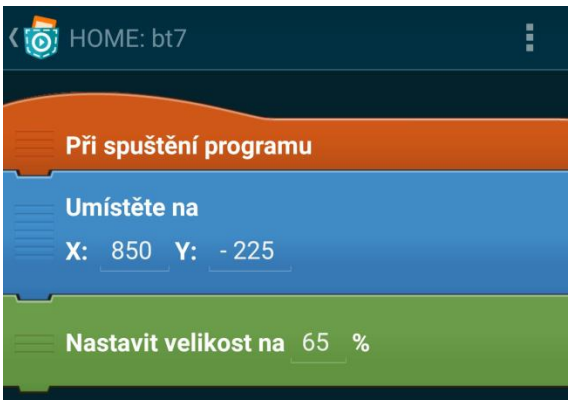

HOME: bt2
Při spuštění programu
Umístěte na
X: 100 Y: 300
Nastavit velikost na 65 %
Na klepnutí
Nastavit Arduino PWM ~ pin 9
do 100

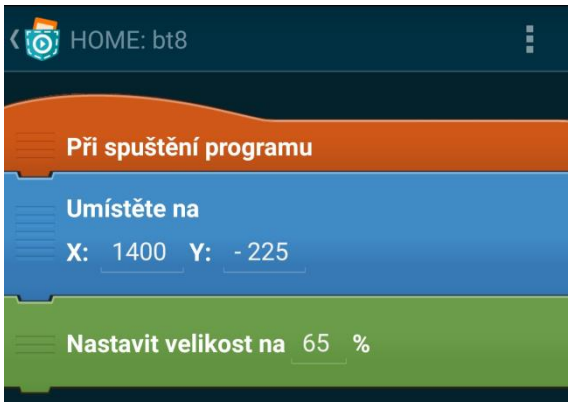

HOME: bt3
Při spuštění programu
Umístěte na
X: 850 Y: 300
Nastavit velikost na 65 %

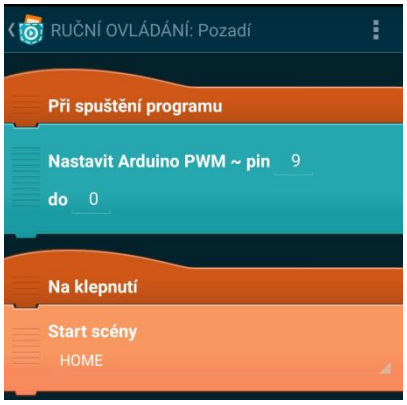

HOME: bt4
Při spuštění programu
Umístěte na
X: 1400 Y: 300
Nastavit velikost na 65 %


HOME: bt5
Při spuštění programu
Umístěte na
X: -450 Y: -225
Nastavit velikost na 65 %

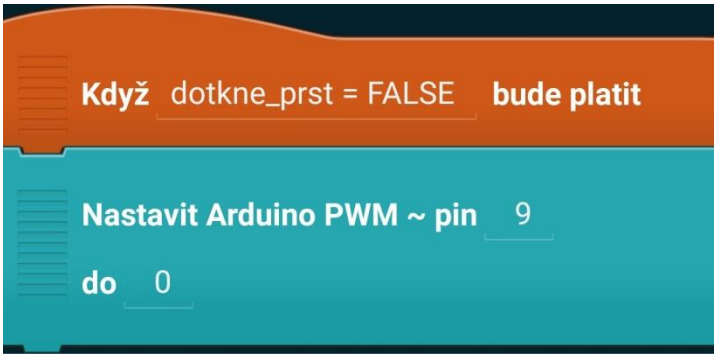

HOME: bt6
Při spuštění programu
Umístěte na
X: 100 Y: -225
Nastavit velikost na 65 %


HOME: bt7
Při spuštění programu
Umístěte na
X: 850 Y: -225
Nastavit velikost na 65 %

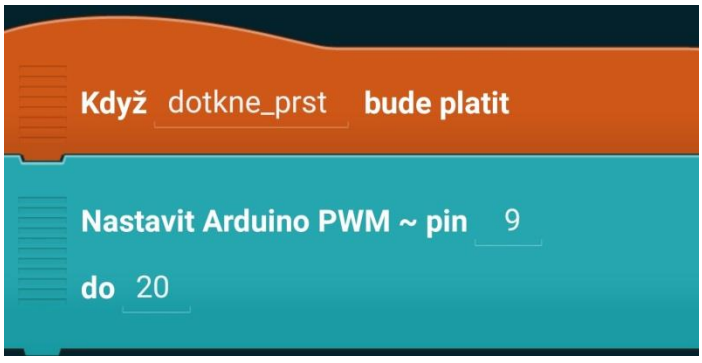

HOME: bt8
Při spuštění programu
Umístěte na
X: 1400 Y: - 225
Nastavit velikost na 65 %


RUČNÍ OVLÁDÁNÍ: Pozadí
Při spuštění programu
Nastavit Arduino PWM ~ pin 9
do 0
Na klepnutí
Start scény
HOME


RUČNÍ OVLÁDÁNÍ: bt_1
Při spuštění programu
Umístěte na
X: - 450 Y: 300
Nastavit velikost na 65 %
Nastavit Arduino PWM ~ pin 9
do 0
Když dotkne_prst bude platit
Nastavit Arduino PWM ~ pin 9
do 10


Když dotkne_prst = FALSE bude platit
Nastavit Arduino PWM ~ pin 9
do 0

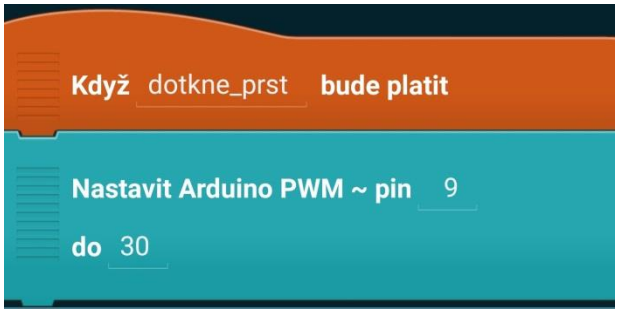

RUČNÍ OVLÁDÁNÍ: bt2
Při spuštění programu
Umístěte na
X: - 450 Y: - 250
Nastavit velikost na 65 %
Nastavit Arduino PWM ~ pin 9
do 0
Když dotkne_prst = FALSE bude platit
Nastavit Arduino PWM ~ pin 9
do 0


Když dotkne_prst bude platit
Nastavit Arduino PWM ~ pin 9
do 20

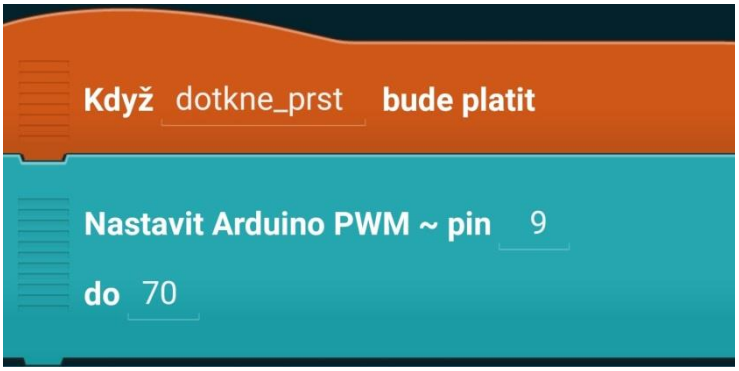

RUČNÍ OVLÁDÁNÍ: bt3
Při spuštění programu
Umístěte na
X: 160 Y: 300
Nastavit velikost na 65 %
Když dotkne_prst = FALSE bude platit
Nastavit Arduino PWM ~ pin 9
do 0


Když dotkne_prst bude platit
Nastavit Arduino PWM ~ pin 9
do 40


RUČNÍ OVLÁDÁNÍ: bt4
Při spuštění programu
Umístěte na
X: 160 Y: - 250
Nastavit velikost na 65 %
Když dotkne_prst = FALSE bude platit
Nastavit Arduino PWM ~ pin 9
do 0


Když dotkne_prst bude platit
Nastavit Arduino PWM ~ pin 9
do 30


RUČNÍ OVLÁDÁNÍ: bt5
Při spuštění programu
Umístěte na
X: 775 Y: 300
Nastavit velikost na 65 %
Když dotkne_prst = FALSE bude platit
Nastavit Arduino PWM ~ pin 9
do 0

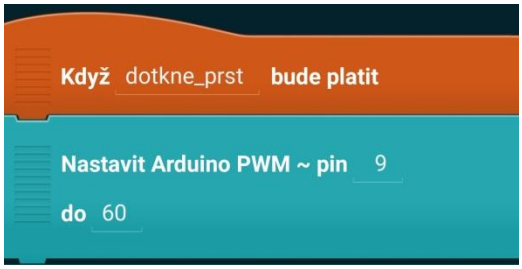

Když dotkne_prst bude platit
Nastavit Arduino PWM ~ pin 9
do 70


RUČNÍ OVLÁDÁNÍ: bt6
Při spuštění programu
Umístěte na
X: 775 Y: - 250
Nastavit velikost na 65 %
Když dotkne_prst = FALSE bude platit
Nastavit Arduino PWM ~ pin 9
do 0


Když dotkne_prst bude platit
Nastavit Arduino PWM ~ pin 9
do 80

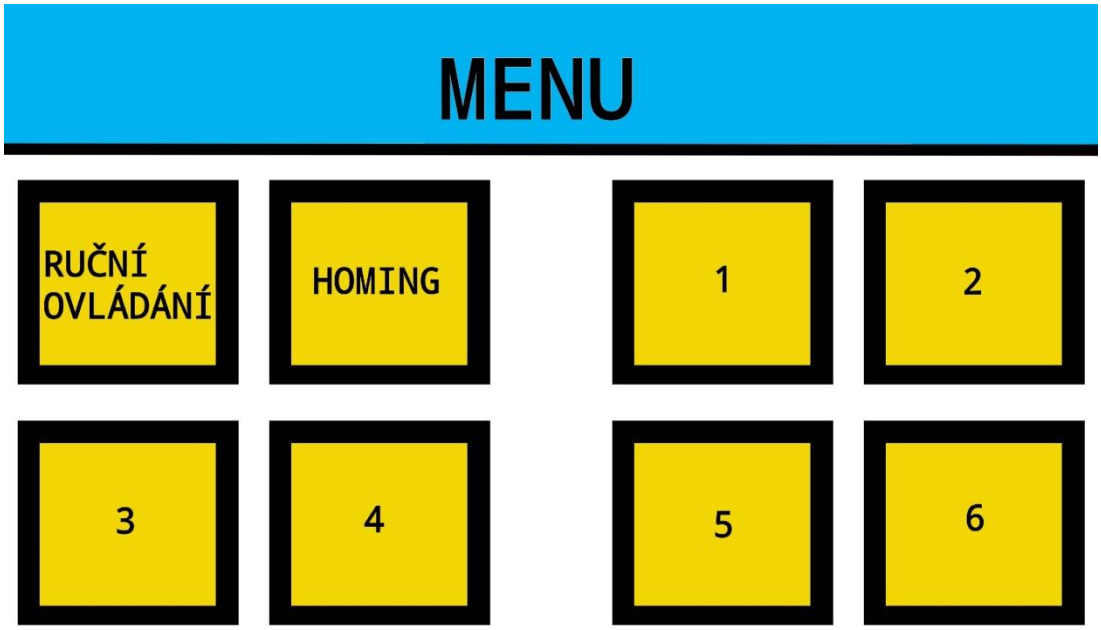

RUČNÍ OVLÁDÁNÍ: bt7
Při spuštění programu
Umístěte na
X: 1400 Y: 300
Nastavit velikost na 65 %
Když dotkne_prst = FALSE bude platit
Nastavit Arduino PWM ~ pin 9
do 0


RUČNÍ OVLÁDÁNÍ: bt8
Při spuštění programu
Umístěte na
X: 1400 Y: - 250
Nastavit velikost na 65 %
Když dotkne_prst = FALSE bude platit
Nastavit Arduino PWM ~ pin 9
do 0

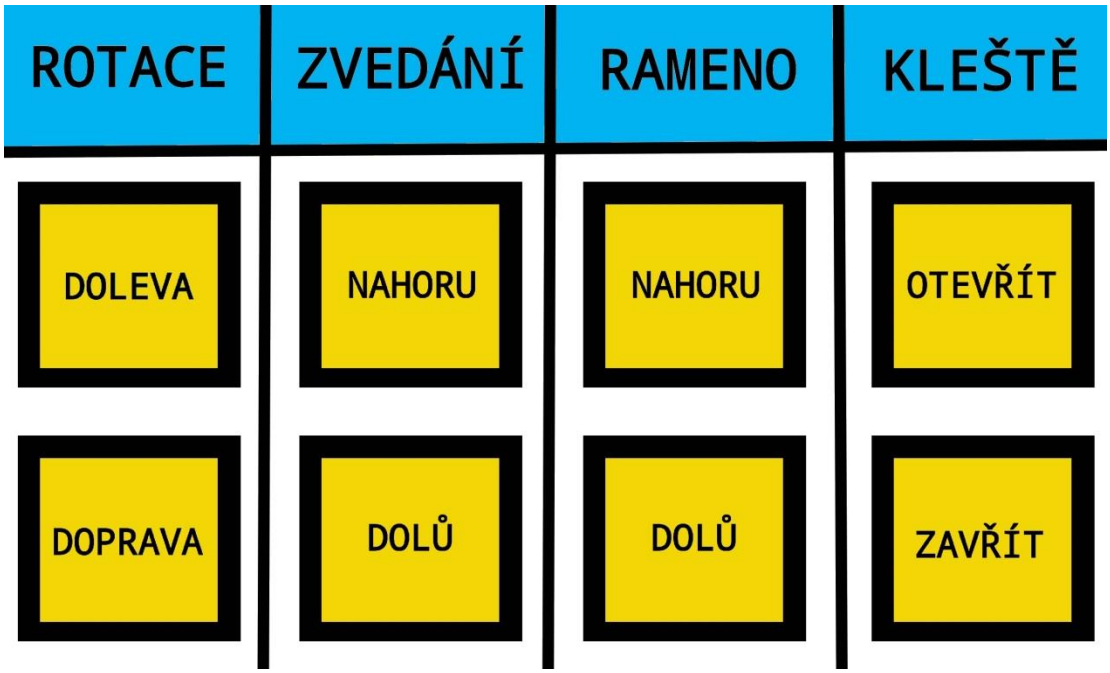

Když dotkne_prst bude platit
Nastavit Arduino PWM ~ pin 9
do 60

6.3 Obrazovky aplikace v Pocket Code

Obrázek 45: Screenshot úvodní obrazovky mobilní aplikace



Obrázek 46: Screenshot menu obrazovky mobilní aplikace



Obrázek 47: Screenshot ručního ovládání mobilní aplikace

Závěr

Cílem této práce bylo vyvinout levný, dostupný, snadno sestavitelný, ale hlavně univerzální řídicí systém pro ovládání výukových robotických manipulátorů pro školy a zájmové kroužky, zaměřený zejména na řízení robotických manipulátorů vytištěných na 3D tiskárnách z některého ze stovek různých návodů dostupných na internetu. Příklady takovýchto levných (do 1000 Kč pořizovacích nákladů), ale pro výuku robotiky vynikajících robotických manipulátorů, které se mi podařilo z návodů na internetu vytisknout a sestavit, jsou uvedeny v kapitole 1.

Domnívám se, že cíl práce se mi podařilo splnit. Sestavený řídicí systém založený na vývojové desce DISCO-F746NG s dotykovým displejem a čtyřech vývojových deskách s moderními motorovými drivery powerSTEP01 umožňuje nejen uživatelsky pohodlné ovládání jedné robotické ruky se čtyřmi bipolárními krokovými motory, ale má i dostatek volných signálů k připojení dalšího robotického manipulátoru se čtyřmi servomotory, pásového dopravníku s optickou závorou pro možnost sestavení složitější aplikace vzájemně spolupracujících robotů, reproduktoru a až několika stovek senzorů a dalších mikrokontrolérů komunikujících po I2C sběrnici, umožňujících takřka nekonečné větvení řídicího systému na další součásti robotických linek. Řídicí systém lze dále ovládat bezdrátově z mobilních zařízení, pomocí Bluetooth komunikace. Materiálové náklady na sestavení navrženého řídicího systému, včetně osazených desek plošných spojů a krabiček vytištěných na 3D tiskárně, nepřekračují 3000 Kč bez DPH. Předložená práce obsahuje kompletní dokumentaci pro výrobu plošných spojů elektroniky, mechanických dílů pro 3D tisk i zdrojových kódů určených k automatickému i manuálnímu ovládání Robotické ruky FRENGP se čtyřmi bipolárními krokovými motory a jedním akcelerometrem a gyroskopem, na níž byla vyvinutý řídicí systém úspěšně vyzkoušen.

Použitá vývojová prostředí Mbed k programování mikrokontrolérů a PocketCode k programování mobilních aplikací na OS Android jsou dostatečně jednoduchá a výborně vysvětlená v dokumentaci i příkladech a tutoriálech na webu, aby je byli schopni aplikovat i začátečníci v programování na středních školách. Navržený řídicí systém by tak měl být upravitelný i samotnými studenty a může sloužit i k výuce programování mikrokontrolérů a mobilních aplikací.

Ve vývoji řídicího systému pro robotické manipulátory bych chtěla dále pokračovat, ať už se zapojením dosud nepoužitých hardwarových funkcí vývojové desky DISCO-F746NG, jako jsou QSPI Flash paměť pro trvalé ukládání nahraných programů pohybu robotů, mikrofony s audikodekem pro analýzu zvuků či ethernet pro možnost řízení robotů po síti. Řídicí systém potřebuje programově uzpůsobit pro ovládání robotických manipulátorů poháněných servomotory, chtěla bych ověřit i složitější funkci spolupráce dvou robotických manipulátorů s pásovým dopravníkem, k čemuž je řídicí systém hardwarově připraven, ale softwarově jsem jej nestihla dokončit.

V neposlední řadě bych si na robotických manipulátorech chtěla po dokončení řídicího systému vyzkoušet některé funkce pokročilého senzorického ovládání robotů interakcí s člověkem, ať již je to ovládání pohybem těla, hlasem, mimikou, svalovými kontrakcemi a v té nejvíce high-tech variantě i mozkovými impulzy.

Veškerou dokumentaci k projektu budu po jejím dokončení publikovat jako open source na svém githubu a webu hahaharobotics.azurewebsites.net

Seznam použitých odkazů

1. **Malá robotická ruka** EEZYbotARM od daGHIZmo[1]**: Thingiverse[online]. [Cit. 14. 3. 2018]. Dostupné z URL:** https://www.thingiverse.com/thing:1015238
2. **Malá robotická ruka** EEZYbotARM od daGHIZmo[1]**: Instructables [online]. [Cit. 14. 3. 2018]. Dostupné z URL:** http://www.instructables.com/id/EEZYbotARM/
3. **Robotická ruka** EEZYbotARM MK2 od stejného autora daGHIZmo[3][4] je zvětšenou verzí předchozího manipulátoru se delším dosahem a zvýšenou nosností pro manipulaci s těžšími předměty. Otáčecí základna obsahuje kuličkové ložisko pro snížení tření při otáčení. K pohonu slouží tři výkonnější servomotory MG995. **od daGHIZmo: Thingiverse [online]. [Cit. 14. 3. 2018]. Dostupné z URL:** https://www.thingiverse.com/thing:1454048
4. **Robotická ruka** EEZYbotARM MK2 od stejného autora daGHIZmo[3][4] je zvětšenou verzí předchozího manipulátoru se delším dosahem a zvýšenou nosností pro manipulaci s těžšími předměty. Otáčecí základna obsahuje kuličkové ložisko pro snížení tření při otáčení. K pohonu slouží tři výkonnější servomotory MG995. **: Instructables [online]. [Cit. 14. 3. 2018]. Dostupné z URL:**http://www.instructables.com/id/EEZYbotARM-Mk2-3D-Printed-Robot/
5. **EEZYbotDELTA je robotická** ruka tvaru delta stejného konstruktéra daGHIZmo**: Thingiverse [online]. [Cit. 14. 3. 2018]. Dostupné z URL:**https://www.thingiverse.com/thing:1249297
6. **EEZYbotDELTA je robotická** ruka tvaru delta stejného konstruktéra daGHIZmo**: Instructables [online]. [Cit. 14. 3. 2018]. Dostupné z URL:**
http://www.instructables.com/id/EEZYbotDELTA-3Dprinted-Robot/
7. **ZORTRAX ROBOTIC ARM: Instructables [online]. [Cit. 14. 3. 2018]. Dostupné z URL:**http://www.instructables.com/id/Zortrax-Robotic-Arm/
8. **ZORTRAX ROBOTIC ARM: Zortrax [online]. [Cit. 14. 3. 2018]. Dostupné z URL:**
https://zortrax.com/stories/blog/entering-the-new-fields-of-industry-3d-printing-applied-to-robotics/
9.**MC3479: datasheet [online]. [Cit. 14. 3. 2018]. Dostupné z URL:**
https://www.onsemi.com/pub/Collateral/MC3479-D.PDF
10. **DISCO F746NG: mikrokontrolér [online]. [Cit. 14. 3. 2018]. Dostupné z URL:**https://os.mbed.com/platforms/ST-Discovery-F746NG/
11. **Eshop Mouser: obchod s elektronikou [online]. [Cit. 17. 3. 2018]. Dostupné z URL:**
https://cz.mouser.com/ProductDetail/STMicroelectronics/STM32F746G-DISCO?qs=sGAEpiMZZMsMyYRRhGMFNlt5IaEdZrcOkzygY9sKdC8%3d
12. **PowerSTEP01: motorový driver [online]. [Cit. 14. 3. 2018]. Dostupné z URL:**http://www.st.com/en/ecosystems/x-nucleo-ihm03a1.html
13. **Eshop Mouser: obchod s elektronikou [online]. [Cit. 17. 3. 2018]. Dostupné z URL:**
https://cz.mouser.com/ProductDetail/STMicroelectronics/X-NUCLEO-IHM03A1?qs=sGAEpiMZZMvNM%2fd3q5fCV2mLU9FR%252bz0orHDN218ieu0%3d
14. **Robodoupě: web o robotice[online]. [Cit. 14. 3. 2018]. Dostupné z URL:**
http://robodoupe.cz/2015/programovani-bluetooth-modulu-hc-05-a-hc-06/
15. **Aukro: obchodní platforma[online]. [Cit. 14. 3. 2018]. Dostupné z URL:**
https://aukro.cz/ard3-31-arduino-bluetooth-modul-hc-05-master-6924828170
16. **Art of Circuits: Eshop s elektronikou [online]. [Cit. 14. 3. 2018]. Dostupné z URL:**
http://artofcircuits.com/product/hc-05-bluetooth-serial-pass-through-master-slave-module
17. **MPU6050: Akcelerometr [online]. [Cit. 14. 3. 2018]. Dostupné z URL:**https://playground.arduino.cc/Main/MPU-6050

18. **Eshop Postav robota: MPU6050 [online]. [Cit. 17. 3. 2018]. Dostupné z URL:** https://www.postavrobota.cz/Digitalni-akcelerometr-gyroskop-MPU6050-modul-3-osy-d119.htm
19. **TME: Eshop s elektronikou[online]. [Cit. 17. 3. 2018]. Dostupné z URL:** https://www.postavrobota.cz/Digitalni-akcelerometr-gyroskop-MPU6050-modul-3-osy-d119.htm
20. **Referenční příručka pro funkce v Mbed OS 5[online]. [Cit. 17. 3. 2018]. Dostupné z URL:** https://os.mbed.com/docs/v5.7/reference/index.html
21. **Programová knihovna pro X-NUCLEO-IHM03A1[online]. [Cit. 17. 3. 2018]. Dostupné z URL:** https://os.mbed.com/teams/ST/code/X_NUCLEO_IHM03A1/
22. **Programová knihovna pro BSP DISCO F746NG[online]. [Cit. 17. 3. 2018]. Dostupné z URL:** https://os.mbed.com/teams/ST/code/BSP_DISCO_F746NG/
23. **Programovací prostředí Pocket Code[online]. [Cit. 17. 3. 2018]. Dostupné z URL:** https://www.catrobat.org/#pocketcode
24. **Obchod Google Play[online]. [Cit. 17. 3. 2018]. Dostupné z URL:** https://play.google.com/store/apps/details?id=org.catrobat.catroid&hl=cs
25. **Program pro ovládání robotické ruky v Pocket Code[online]. [Cit. 17. 3. 2018]. Dostupné z URL:** https://share.catrob.at/pocketcode/program/47005

Seznam obrázků

Seznam tabulek

Seznam příloh elektronické verze SOČ

Příloha 1. Schémata plošných spojů v Eagle CAD

Příloha 2. Modely krabiček programu SketchUp a vyexportované STL soubory modelů

Příloha 3. Program pro STM32F746NG v Mbed OS 5