Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Robotické stavebnice pro výuku programování na základních školách

Michael Lév

Bakalářská práce
2021

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Michael Lév**
Osobní číslo:	**I18275**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Robotické stavebnice pro výuku programování na základních školách**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem práce je sestavit sadů příkladů využitelných v oblasti robotiky bude možné využít i pří výuce programování nebo algoritmizace pro žáky základních škol. Robotické stavebnice bude mít student k dizpozici od vedoucího práce. Bakalářská práce by měla obsahovat takové příklady, které pokryjí důležité úlohy, které se v oblasti programování vyskytují (tzn. cykly, podmínky atp.).

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

TROJÁNEK, P. Využití robota LEGO MINDSTORMS při výuce. Praha: ČVUT, 2009.
TOCHÁČEK, D., LAPEŠ, J. Integration educational robotics into the training of future ICT teachers. In: ALIMISIS, D., MORO, M. Proceedings of 3rd International Workshop Teaching Robotics, Teaching with Robotics Integrating Robotics in School Curriculum. Riva del Garda: TRTWR, 2012, pp. 51-56. ISBN 978-88-95872-05-6.

Vedoucí bakalářské práce: **Ing. Jan Panuš, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2020**
Termín odevzdání bakalářské práce: **14. května 2021**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 26. února 2021

PROHLÁŠENÍ

Prohlašuji:

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

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

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

V Pardubicích dne 12. 5. 2021

Michael Lév v. r.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Janu Panušovi, Ph.D. za jeho čas, trpělivost a cenné rady, které mi velice pomohly při tvorbě této bakalářské práce.

ANOTACE

Bakalářská práce s názvem „Robotické stavebnice pro výuku programování na základních školách“ se zabývá využitím robotických stavebnic, především pak stavebnice Lego Mindstorms, při výuce programování a algoritmizace pro studenty základních škol. V práci je nastíněno současné využití robotických stavebnic při výuce na základních školách v ČR. Dále je pak podrobněji popsána stavebnice Lego Mindstorms, její hardware, vývojové prostředí a jeho využití při programování této stavebnice. V praktické části jsou pak popsány příklady zaměřené na základy programování vhodné pro studenty základních škol, které lze využít při praktické výuce.

KLÍČOVÁ SLOVA

Robot, robotika, LEGO, LEGO MINDSTORMS, programování, algoritmizace, výuka programování

TITLE

Robotics kits for teaching programming at primary schools

ANNOTATION

Bachelors thesis named “Robotics kits for teaching programming at primary schools” deals with the usage of robotics building kits, especially Lego Mindstorms, for teaching primary school students programming and algorithmization. The work describes the current use of robotics building kits in education in elementary schools in the Czech Republic. Further, the thesis describes the Lego Mindstorms kit, its hardware, the developing environment, and its usage for programming. In the practical part of work are examples focused on basics of programming suitable for primary school students. These can be used for hands-on programming teaching.

KEYWORDS

Robot, robotics, LEGO, LEGO MINDSTORMS, programming, algorithmization, programming education

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

ČR	Česká republika
FLASH	elektricky programovatelná paměť s libovolným přístupem
IDE	integrované vývojové prostředí
IR	infračervený
LCD	liquid crystal display, displej z kapalných krystalů
LED	světlo emitující dioda
MicroSD	výměnná paměťová karta odvozená od formátu Secure Digital
MIT	Massachusettský technologický institut
PC	osobní počítač
RAM	random access memory, paměť s přímým přístupem
STEAM	přístup k učení, využívající vědu, techniku, inženýrství, umění a matematiku
USB	univerzální sériová sběrnice

ÚVOD

Cílem této bakalářská práce je prozkoumat možnosti dnešních robotických stavebnic, seznámit se s jejich hardwarem, softwarem a možnostmi, jak je lze programovat. Na základě těchto znalostí poté bude sestavena sada praktických příkladů, které bude možné využít při výuce programování a algoritmizace na základních školách. Příklady budou pokrývat nejen základních seznámení se s programovými konstrukcemi a tvorbou programů, ale budou také zaměřené na rozvoj logického, analytického i informačního myšlení.

V první části bude popsán pojem robotické stavebnice, nastíněna podoba výuky informatiky na základních školách v ČR a její změny plánované od počátku školního roku 2021/2022. Následně jsou pak prezentovány možné výhody a nevýhody, které skýtá použití robotických stavebnic při výuce.

Následující část je věnována stavebnici Lego Mindstorms EV3, kde je podrobněji představen hardware, který je obsahem balení, jako je např. základní řídící kostka, různé motory nebo senzory.

Třetí část se zabývá samotným programováním robota. Je představeno vývojové prostředí Lego Mindstorms EV3 Classroom spolu s programovacím jazykem Scratch. Doplněna je také druhá oficiálně podporovaná možnost, jak vytvářet programy pro roboty, a tím je textový jazyk MicroPython.

V poslední části pak naleznete praktické příklady využití robota Lego Mindstorms EV3, které lze použít pro výuku základů programování pro žáky základních škol. Nachází se zde jednodušší i složitější příklady na procvičení základních programových konstrukcí, jako jsou podmínky nebo cykly, základní práci s různými druhy senzorů či použití motorů pro pohyb robota.

1 ROBOTICKÉ STAVEBNICE PŘÍ VÝUCE

Pojem stavebnice popisuje sadu určitých součástek, které je možné spolu libovolně kombinovat a sestavovat tak různé trojrozměrné modely. Při práci s ní se zlepšují motorické schopnosti, kreativita, fantazie nebo také prostorová představivost.

Robotická stavebnice pak v sobě kloubí stavebnici a počítač. Ten se nachází povětšinou uvnitř speciální součástky této stavebnice a umožňuje „oživit“ sestavený model pomocí předem připraveného či vlastního programu. Takovéto robotické stavebnice nabízí možnosti pro zlepšení počítačové gramotnosti a položení základů výuky programování a algoritmizace. Navíc v sobě kombinují nejen učení, ale i zábavu.

V České republice jsou k dostání různé druhy a verze robotických stavebnic od různých firem. Dle výzkumu Mgr. Jana Baťka ze Západočeská univerzity v Plzni provedeném v období poslední čtvrtiny roku 2016 [2] je při výuce na základních školách zdaleka nejčastěji využívanou stavebnicí Lego Mindstorms od dánské firmy Lego. Ta se využívá téměř v 80 % základních škol ze zapojeného vzorku. Druhou nejpoužívanější stavebnicí byla česká stavebnice Merkur. Mezi dalšími používanými se objevily stavebnice Arduino, Fischertechnik, Makeblock nebo RoboRobo. Ty se však využívaly v pouhých jednotkách procent zapojených základních škol.

1.1 Výuka programování na základních školách v ČR

V České republice do roku 2021 nebylo programování součástí rámcového vzdělávacího programu pro základní školy. Výuka informatiky byla zaměřena převážně na základní práci s počítačem a informacemi, jejich vyhledávaní a zpracování. Výuka programování byla tedy pouze na dobrovolném rozhodnutí vedení každé školy.

V lednu 2021 byl však Ministerstvem školství, mládeže a tělovýchovy schválen nový rámcový vzdělávací program pro základní vzdělávání [1], který začíná platit od začátku školního roku 2021/2022. Zásadní změnou je změna původního předmětu „informační a komunikační technologie“ za předmět „informatika“ s vyšší časovou dotací a celkově odlišným obsahem. Minimální časová dotace se zvyšuje z dosavadní povinné 1 vyučovací hodiny týdně pro oba stupně základního vzdělávaní nově na 2 vyučovací hodiny pro 1. stupeň a na 4 vyučovací hodiny pro 2. stupeň.

Nově se také zásadně mění obsah tohoto předmětu. Původně se žáci učili převážně pracovat s textovými či tabulkovými editory, prezentačními programy, programy zaměřujícími se na

zpracování počítačové grafiky či vývojem jednoduchých webových stránek. Novým obsahem tohoto předmětu bude mimo zpracování dat a jejich modelování také algoritmizace a programování, informační systémy a digitální technologie. Již žáci prvního stupně se tak budou postupně učit, jak popsat problém, analyzovat ho, najít jeho řešení a ověřit si jej ve vhodném programovém prostředí.

1.2 Využití robotických stavebnic při výuce

Používání robotických stavebnic ve výuce, může pomoct studentům rozvinout jejich logické myšlení, kreativitu, schopnost řešení problému, ale také může poskytnout základy programování, matematiky či fyziky.

Díky využívání blokově orientovaných programovacích jazyků je programování těchto stavebnic snazší a přístupnější než tradiční programovací jazyky jako jsou Java, C#, Python nebo C++. Jedním z nejznámějších a nejpoužívanějších takových jazyků je Scratch, ve kterém se programuje pomocí spojování bloků s různými barvami a tvary, které reprezentují jednotlivé programové konstrukce. Není třeba tak řešit syntaktické chyby ani nelogické konstrukce, protože Scratch nepovolí takové spojení bloků, které by nedávalo smysl. Žáci se tak mohou plně soustředit na pochopení úlohy, rozdělení jejího řešení na jednotlivé kroky a následné sestavení finálního řešení.

Další nespornou výhodou je, že roboti nejsou pouze abstraktní, ale lze si na ně sáhnout a vidět výsledek své práce v reálném světě. Žáci tak okamžitě dostávají zpětnou vazbu a vidí, zda sestavený robot funguje a plní správně požadovaný úkol.

Roboti jsou také vhodní pro žáky nižšího i vyššího stupně základních škol. Ti nejmladší se díky grafickým programovacím jazykům snadno mohou naučit základy programování pomocí jednoduchých úkolů, které jim mohou připadat spíše jako hra než učení se. Pro starší žáky pak mohou přijít složitější úlohy, kde si sami budou muset poradit při návrhu a sestavování jednotlivých robotů, tak aby co nejlépe mohli splnit zadanou úlohu.

Nevýhodou těchto robotických stavebnic může být jejich vyšší pořizovací cena. Ta se pohybuje v řádech tisíců korun, a tak většinou není možné pořídit tolik sad robotických stavebnic, aby každý žák měl svou vlastní. Pokud se na tento problém však podíváme z jiného úhlu pohledu, může jej brát také jako výhodu. Díky omezenému počtu stavebnic je možné sestavit skupiny, kde každé bude přidělena jedna stavebnice. Studenti se tak seznámí s prací v týmu, kdy budou

muset spolu komunikovat a spolupracovat, tak aby robota správně sestavili, naprogramovali a splnili tak zadání.

Komplikací při výuce může být také samotná konstrukce robotů, protože kvůli nižší hodinové dotaci je čas pro výuku omezen. V České republice se jedná nejčastěji o 1-2 hodiny týdně. Konstrukce robotů může zabrat značné množství času, a tak zkrátí dobu určenou pro programování, které by mělo být primárním cílem výuky. Je tak nutné najít určitý balanc mezi sestavováním robota a vytvářením programu. Důležité je volit úlohy s vhodnou obtížností a časovou náročností, aby žáci měli dostatek času jak na sestavování, tak i programování robotů. Jednou z možností je také využívat již dříve sestavené roboty pro tyto úlohy.

2 LEGO MINDSTORMS EV3

Stavebnice Lego Mindstorms EV3 byla představena v roce 2013 jako nástupce Lego Mindstorms NXT. Přinesla nejen vyšší výkon, ale také větší množství paměti, vlastní operační systém založený na Linuxu, lepší displej či nové vývojové prostředí.

Stavebnice je k dostání ve dvou verzích. První verze určená pro běžnou veřejnost a druhá verze určená do škol s označením Education. Tyto stavebnice se mírně liší v obsahu balení. Obě balení však obsahují hlavní řídící kostku EV3 a 3 servomotory.

Verze určená do škol nabízí oproti běžné menší možností kostek pro stavbu různých druhů robotů. Neobsahuje také infračervený senzor s ovladačem. Na rozdíl od verze určené do domácností nabízí navíc nabíjecí baterii pro EV3 kostku, dva dotykové, gyroskopický a ultrazvukový senzor.

2.1 EV3 Brick (kostka EV3)

Inteligentní, programovatelná kostka EV3 je srdcem a zároveň mozkem celé stavebnice Lego Mindstorms EV3. Uvnitř můžeme najít 32bit ARM9 procesor AM1808 Texas Instruments 300 MHz, který běží na operačním systému Linux, 64 MB paměti a 16 MB paměti FLASH. Pro propojení s počítačem, chytrým telefonem či dalšími kostkami lze využít rozhraní USB 2.0, Bluetooth nebo Wi-Fi.

V běžně dostupné Home verzi je kostka je napájena pomocí 6 tužkových baterií. Doporučuje se používat alkalické nebo nabíjecí lithium-iontové baterie. Tyto se vkládají do kostky z její spodní strany po odstranění krytu baterií. Ve verzi Education je kostka napájena pomocí znovu dobíjecí baterie s kapacitou 2050 mAh.

Obr. 1 Lego EV3 Brick [12]

Na přední straně kostky se nachází černobílý displej s rozlišením 178 x 128 pixelů. Pod tímto displejem najdeme 6 podsvícených tlačítek pro přístup do jejího rozhraní. Barva podsvícení tlačítek může svítit nebo blikat třemi různými barvami, které indikují aktuální stav kostky.

Na horní straně kostky lze najít 4 výstupní porty RJ12, označené písmeny A, B, C, D. Ty slouží pro připojení motorů ke kostce. Podporují funkci zvanou Auto ID, díky které automaticky rozpoznají o jaký druh motoru se jedná. Vedle těchto portů se nachází ještě miniUSB port, který slouží ke komunikaci s hostitelským počítačem. Podporuje USB 2.0 a dokáže přenášet data rychlostí až 480 Mbit/s.

Obr. 2 Horní strana EV3 Brick [13]

Ze přední strany kostky jsou opět čtyři RJ12 porty, tentokrát se však jedná o porty výstupní. Jsou označeny čísly od 1 do 4 a slouží k připojení senzorů.

Obr. 3 Spodní strana EV3 Brick [13]

Z pravé boční strany se nachází reproduktor, ze kterého vycházejí všechny zvuky, které EV3 kostka vydává, včetně zvukových efektů použitých při naprogramování robota.

Obr. 4 Pravá strana EV3 Brick [13]

Na levé straně se nachází slot pro MicroSD kartu, díky němuž lze paměť kostky rozšířit až o 32 GB. Vedle tohoto slotu je pak běžný USB port ve verzi USB 1.1 s rychlostí až 12 Mbit/s, který lze využít pro vzájemné propojení s další kostkou. Takto propojit lze dohromady až 4 EV3 kostky.

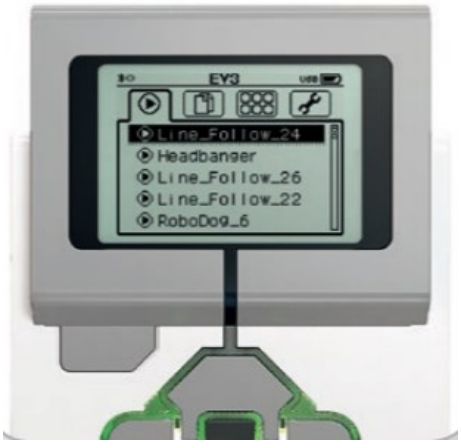
Obr. 5 Levá strana EV3 Brick [13]

2.2 Rozhraní kostky EV3

Rozhraní kostky se skládá z displeje a ovládacích tlačítek. Pomocí něj lze kostku spouštět, vypínat, nastavovat, spouštět programy, sledovat jednotlivé porty, a dokonce i vytvářet nové programy.

Ovládání probíhá pomocí dostupných tlačítek. Levé horní tlačítko plní funkci navrácení akce, zastavení programu nebo vypnutí kostky. Středové tlačítko funguje jako potvrzovací tlačítko. Další 4 tlačítka představují šipky nahoru, dolů, doleva, doprava, pomocí nichž probíhá navigace při procházení rozhraní.

Na displeji lze vidět vždy horní lištu se základními informacemi jako je stav baterie, případně stavy připojení pomocí USB, Bluetooth či Wi-Fi. Další nabídka kostky je následně rozdělena celkem do 4 záložek.



Obr. 6 Rozhraní EV3 Brick [13]

První záložka obsahuje naposledy spuštěné programy, pokud nějaké takové byly.

V druhé záložce se nachází možnost procházet dostupné soubory, a to jak v kostce, tak v připojené paměťové kartě. Jednotlivé programy jsou pak uložené ve složkách. V nich lze najít programové, zvukové a obrázkové soubory. Všechny tyto soubory je možno také přesouvat či mazat.

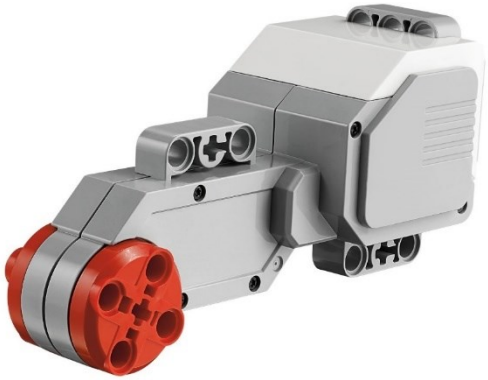
Třetí záložka obsahuje seznam aplikací, které se v kostce nacházejí, a to vlastní i ty předinstalované výrobcem. Zde se jednotlivé programy spouští. Nachází se zde 4 již předinstalované aplikace. Mezi nimi najdeme program Port View, který dokáže zobrazit přehled jednotlivých obsazených portů, doplňujících informací a hodnot z konkrétních portů. Druhou aplikací je Motor Control, díky které lze přímo ovládat připojené motory pomocí šipek. Další aplikace je IR Control s jejíž pomocí lze motory ovládat infračerveným ovladačem. Poslední aplikací je Brick Program, která nabízí možnost vytvářet vlastní programy přímo v rozhraní kostky.

Poslední záložka je určená k nastavení kostky. Zde lze nastavit např. hlasitost kostky, délku doby nečinnosti nebo správu a nastavení připojení kostky pomocí Bluetooth a Wi-Fi. Zobrazit si zde lze také informace o kostce samotné jako např. verzi firmwaru, sestavení operačního systému nebo informaci o velikosti zbývající volné paměti.

2.3 EV3 Motory

Pro stavebnici Lego Mindstorms EV3 jsou k dispozici dva druhy motorů, které lze použít k rozpohybování robotů. V balení lze najít dva velké motory a jeden střední motor. Oba tyto motory podporují funkci Auto ID, která dokáže automaticky identifikovat jaký motor je připojen k jakému portu při připojení k PC. Motory také disponují integrovaným senzorem otáčení s rozlišením 1° pro přesné ovládaní.

Velký motor je větším, těžším a silnějším z nabízených motorů. Je ale také na rozdíl od druhého pomalejší. Dokáže vykonat 160–170 otáček za minutu, má točivý moment 20 Ncm a moment zvratu 40 Ncm. Je navržen tak, aby sloužil jako hlavní hnací síla při pohybu robotů.



Obr. 7 Velký servomotor EV3 [13]

Střední motor je menší, lehčí a rychlejší než jeho větší bratr. Nedokáže vyvinout však takovou sílu, proto se používá spíše pro rotační pohon různých pohyblivých ramen apod. Dokáže vykonat 240–250 otáček za minutu, točivý moment má 8 Ncm a moment zvratu 12 Ncm.

Obr. 8 Střední servomotor EV3 [13]

Motory lze používat v různých režimech nastavení. Mohou být zapnuté po předem určenou dobu času, na předem určený počet otáček či na velikost úhlu o který se mají otočit. Lze je také spustit v neregulovaném módu, kde běží nepřetržitě do zastavení. Určovat lze také rychlost.

2.4 EV3 Senzory

Díky dodávaným senzorům roboti Lego Mindstorms EV3 dokážou přijímat vstupy ze svého okolí a následně také na ně reagovat. V běžném balení určených pro širokou veřejnost se nachází vždy po jednom kusu dotykový senzor, senzor barev a infračervený senzor spolu s infračerveným ovladačem, který je označován jako Remote Infrared Beacon v překladu vzdálený infračervený maják. V baleních určených do škol pak můžeme navíc najít jeden dotykový senzor, gyroskopický senzor a ultrazvukový senzor. Co zde chybí je infračervený senzor s ovladačem.

Mimo senzory vyráběné přímo firmou Lego, lze zakoupit také různé doplňkové senzory dostupné od oficiálních partnerů HiTechnic, Vernier a DCP Microdevelopment. Jedná se např. o kompas, zvukový senzor, teplotní senzor nebo vylepšený barevný senzor,

2.4.1 Barevný senzor

Je digitálním senzorem, který dokáže rozeznat až 8 různých barev. Lze jej také využít jako světelný senzor, který rozeznává různou intenzitu světla. Vzorkovací frekvence tohoto senzoru je 1 kHz. Podporuje 3 módy nastavení podle potřeby využití: barevný režim, režim intenzity odraženého světla a režim intenzity okolního světla.

Obr. 9 Senzor barev EV3[13]

V Color Modu (barevný režim) dokáže senzor rozpoznat 7 barev, a to bílou, černou, červenou, modrou, zelenou, žlutou, hnědou. Pokud nalezená barva neodpovídá ani jedné z výše jmenovaných barev, je výsledek uváděn jako *No Color*. Díky tomuto nastavení lze robota naprogramovat, aby reagoval určitou akcí při rozpoznání konkrétní barvy.

Druhým režimem je Reflected Light Intensity Mode (režim intenzity odraženého světla). Ten využívá červenou přisvětlovací LED diodu umístěnou na čelní straně, s jejíž pomocí měří intenzitu světla, které se odrazí nazpátek. Pro vyhodnocování se používá škála od 0 do 100,

která reprezentuje světlost odrazu od velmi tmavé po velmi světlou. Tento mód lze využít např. pokud chceme, aby se robot pohyboval pouze ve vymezeném prostoru, který je celý světlý a je ohraničen tmavou barvou.

Posledním režimem je Ambient Light Intensity Mode (režim intenzity okolního světla), který dokáže měřit sílu světla, které vstupuje do senzoru z okolního prostředí. Opět používá stupnici od 0 do 100 pro číselné ohodnocení, jak moc světla je v okolí senzoru. Tento režim lze využít v případě, že chceme, aby se robot automaticky vypnul při zhasnutí světla v místnosti.

2.4.2 Dotykový senzor

Je analogovým senzorem, který má na své čelní straně umístěné červené tlačítko. Díky němu dokáže reagovat na jeho stisknutí a uvolnění. Rozpoznávat lze 3 stavy: stisknuto, uvolněno a náraz. Stisknutí reaguje na stlačení tlačítka a platí, dokud je senzor stlačen. Uvolnění je aktivováno, jakmile je senzor uvolněn. Poslední stav je spojením přechozích dvou stavů a představuje situaci kdy je tlačítko stisknuto a následně uvolněno. Tlačítko tak musí být nejprve stisknuto, aby mohla být tento stav aktivován. Využití tohoto senzoru spočívá především ve spojení se spuštěním případně zastavením robota, pokud např. narazí do překážky nebo hrozí pád při doputování na hranu podložky.

Obr. 10 Dotykový senzor EV3 [13]

2.4.3 Infračervený senzor a ovladač

Je digitálním senzorem, který dokáže vysílat a přijímat vlny infračerveného světla, které má schopnost odrážet se od pevných povrchů. Dokáže také zaznamenat infračervený signál vyslaný z dodávaného ovladače. Podporuje tři módy: režim přiblížení (Proximity Mode) režim majáku (Beacon Mode) a vzdálený režim (Remote Mode).

Infračervený ovladač, také označován jako maják, je zařízení vysílající infračervený signál do svého okolí. Lze jej nastavit na 4 různé kanály na kterých bude signál vysílán. Je napájen dvěma

mikrotužkovými bateriemi. Lze jej používat jak samostatně, tak zakomponovaný do konstrukce robota. Na přední straně se nachází 4 tlačítka pro ovládání, tlačítko *beacon* pro zapnutí a vypnutí majáku, LED dioda pro určení aktivního stavu a přepínač kanálů.

Obr. 11 Infračervený senzor EV3 a Infračervený majáček EV3 [13]

V prvním režimu nazvaném přiblížení, senzor vysílá do okolí světelné vlny, které se odráží od okolních předmětů. Po zachycení těchto odražených vln se pak počítá přibližná vzdálenost z rozdílů časů vyslání a zachycení světla robotem. Nepočítá se však přesná vzdálenost, ale udává se hodnota z intervalu 0 (blízko) až 100 (daleko). Senzor dokáže rozeznat předměty až ve vzdálenosti 70 cm.

Pro režim majáku je nutné spustit vysílání signálu infračerveným ovladačem na stejném kanále, na který je nastavený robot. Takovýto robot pak dokáže najít vysílaný signál, pokud se nachází ve směru otočení (90° doprava/doleva) a do vzdálenosti 2 m. Vzdálenost od majáku se udává v hodnotách 0 až 100 a směr jízdy v intervalu od -25 po 25.

Posledním režimem je tzv. Remote Mode neboli vzdálený režim. Pro ten je opět potřeba infračervený ovladač, který lze použít jako ovladač pro ovládání robota. Robot umí reagovat na jednotlivá stisknutí tlačítek, případně na kombinaci dvou stisknutých tlačítek. Dohromady je tak možno nastavit až 11 různých akcí v reakci na jednu z 11 možných kombinací.

2.4.4 Ultrazvukový senzor

Je digitální senzor, který umí generovat zvukové vlny a následně zachytávat jejich odraz od ostatních objektů. Dokáže také vysílat jednu konkrétní zvukovou vlnu a fungovat jako sonar, nebo naslouchat na konkrétní zvukovou vlnu z jiného senzoru a následně na ní reagovat. Dokáže měřit vzdálenost od 1 cm do 250 cm a výsledek je opět reprezentován na stupnici od 0

do 100. Přesnost měření je udávaná na ± 1 cm. Praktické využití najde například při dodržování určité vzdálenosti robota od překážek či jiného robota.

Obr. 12 Ultrazvukový senzor EV3 [14]

2.4.5 Gyroskopický senzor

Je digitální, jednoosý senzor, měřící natočení robota a změny v jeho orientaci. Dokáže měřit jak úhel natočení, tak jeho rychlost. Úhel dokáže měřit s přesností ± 3° pro 90° otočení. Maximální rychlost natočení, kterou dokáže zaznamenat je 440° za sekundu. Vzorkovací frekvence senzoru je 1 kHz. Díky tomuto senzoru tak lze vytvářet balancující roboty na podobném principu jako funguje např. vozítko Segway.

Obr. 13 Gyroskopický senzor EV3 [14]

2.4.6 Zapojení technologie EV3

Pro fungování jednotlivých motorů a senzorů je nutné je připojit k EV3 kostce. Pro připojení se používají černé ploché kabely zakončené konektory RJ12, které jsou dostupné v balení.

Senzory se připojují ze spodní strany kostky do vstupních portů. Ty jsou označeny čísly od 1 do 4. Toto označení má svůj význam, protože pokud kostka není připojena k PC, jsou jednotlivé druhy senzorů defaultně přiřazeny jednotlivým portům.

- Port 1: dotykový senzor.
- Port 2: gyroskopický senzor.
- Port 3: barevný senzor.
- Port 4: infračervený senzor.

Pokud bychom chtěli senzory připojit jiným způsobem, je nutné kostku připojit k PC a jednotlivé porty automaticky rozpoznají jaký senzor je připojen k jakému portu.

Obr. 14 Zapojení motorů a senzorů do EV3 Brick [13]

Stejným způsobem funguje také zapojení jednotlivých motorů, pro které jsou určené porty označené písmeny A až D. Ty jsou umístěny na horní straně kostky. Výchozí nastavení opět přiřazuje jednotlivé druhy motorů daným portům a pro případnou změnu je opět nutné připojit kostku k počítači.

- Port A: střední motor.
- Port B a C: 2x velký motor.
- Port D: velký motor.

3 PROGRAMOVÁNÍ ROBOTA LEGO MINDSTORMS EV3

Pro programování stavebnice Lego Mindstorms EV3 je k dispozici hned několik dostupných možností. Jednou z možností je vytvářet programy přímo na EV3 kostce, avšak díky malému černobílému displeji a omezenému ovládání to není příliš praktické. Řešením tak je oficiálně nabízený software. Ten je k dispozici pro počítače s operačním systémem Windows nebo macOS, ale také jako mobilní aplikace dostupná na tabletech s Androidem a iOS. Pro zkušenější uživatele je nabízeno velké množství neoficiálních vývojových prostředí.

Pro počítače je k dispozici hned několik různých programů a jejich verzí. Pro veřejně dostupnou verzi stavebnice je k dostání software s názvem Lego Mindstorms EV3 Home Edition Software. Tento program však má být brzy nahrazen novým programem, který je zatím dostupný pouze na macOS 10.15. Aktualizace pro další platformy je naplánovaná na jaro 2021.

Pro výukovou verzi stavebnice je stále k dispozici původní program zvaný EV3 Lab Software. Ten byl však v letech 2019 a 2020 potupně nahrazen novým programem pojmenovaným Lego Mindstorms Education EV3 Classroom [18]. Ten se zásadně liší od původní verze. Krom vzhledu aplikací je největším rozdílem využití odlišného programovacího jazyka. Původní jazyk založený na LabVIEW, je zde nahrazen jazykem založeným na Scratchi 3.0.

Dále jsou k dispozici aplikace pro mobilní platformy. Ty jsou opět rozdělené podle určení stavebnice. Pro domácí verzi jsou k dispozici aplikace EV3 Programmer pro programování robotů a aplikace Lego Mindstorms Commander pro ovládání robota. Pro školní verzi je k dispozici aplikace EV3 Classroom Lego Education, která je shodná s desktopovou verzí programu EV3 Classroom.

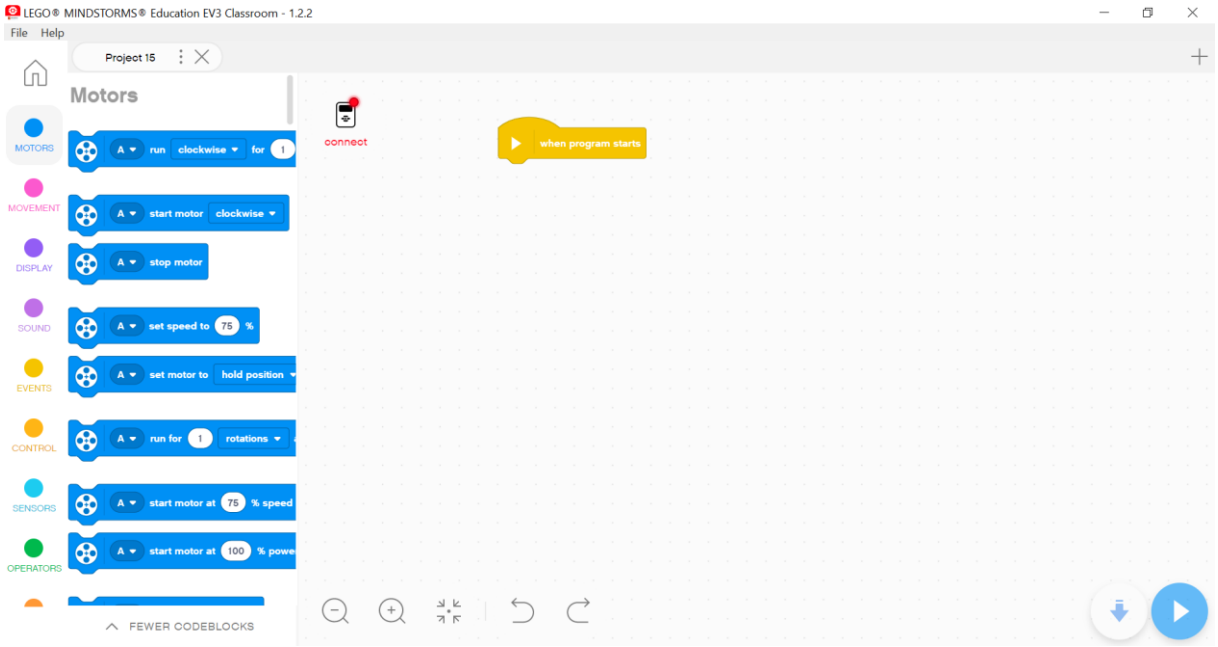
K dispozici je také celá řada neoficiálních programových prostředí podporujících různé programovací jazyky. Jedná se například o RobotC založený na jazyce C, EV3Python využívající Python, MakeCode ve kterém lze použít ke kódování buď grafické bloky či JavaScript, leJOS podporující Javu a mnoho dalších.

3.1 Lego Mindstorms Education EV3 Classroom

Tento program je určený pro programování stavebnice Lego Mindstorms Education EV3 Core Set. Byl vydán v roce 2019 pro systémy od americké firmy Apple s macOS. V roce 2020 pak vyšel i pro operační systémy Windows 10, Android a iOS. Na všech platformách nabízí jednotný vzhled. Je dostupný v 16 světových jazycích, čeština mezi nimi však chybí.

Zaměřen je především na výuku, nabízí zjednodušený a přehlednější vzhled, který je na všech platformách jednotný pro bezproblémový přechod z jednoho systému na jiný. V rámci dostupných návodů obsahuje také speciální sekci určenou jen učitelům, které je připravuje na výuku s tímto programem.

Hlavní novinkou je pak nově používaný jazyk Scratch, který nahradil původní jazyk založený na LabVIEW. Místo ikon se tak zde používají barevné bloky s přepínači, kde jednotlivé barvy určují, pro jaký účel je blok určen. Nachází se zde kategorie např. pro motory, senzory, displej, zvuky, události a další. Tyto bloky se pak spojují za sebe v směru od shora dolů.



Obr. 15 Vývojové prostředí Lego Mindstorms Education EV3 Classroom

3.2 Scratch 3.0

Jedná se o grafický programovací jazyk. Původní jazyk Scratch je blokový, určený především pro výuku, vhodný pro děti od 8 do 16 let. Jeho první verze vyšla v roce 2007 a od té doby získal již více než 69 milionů registrovaných uživatelů.

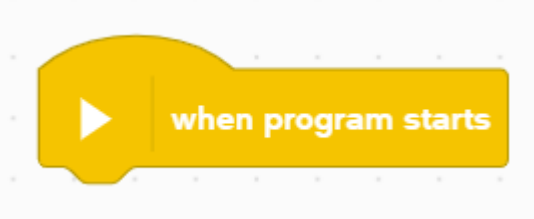
V programu EV3 Classroom je využívána speciálně upravená verze pro potřeby stavebnice Lego Mindstorms. Oproti běžnému verzi se zde nachází speciálně upravené bloky pro práci s motory, senzory, displejem a podobně. Využit je styl tzv. drag-and-drop programování, kdy jednotlivé bloky se přesouvají pomocí táhnutím myši a navzájem se do sebe horizontálně zapojují jako puzzle. Díky rozlišným tvarům a zakončením je zaručeno, že nenastanou žádné syntaktické chyby, protože editor dovoluje spojit jen takové bloky, které do sebe zapadají.

3.2.1 Tvary bloků

Jednotlivé bloky mají rozdílné tvary, kde každý tvar je určen pro jiný účel. Celkem je k dispozici 6 tvarů primitivních bloků: Hat, Stack, Boolean, Reporter, C a Cap.

Hat blok

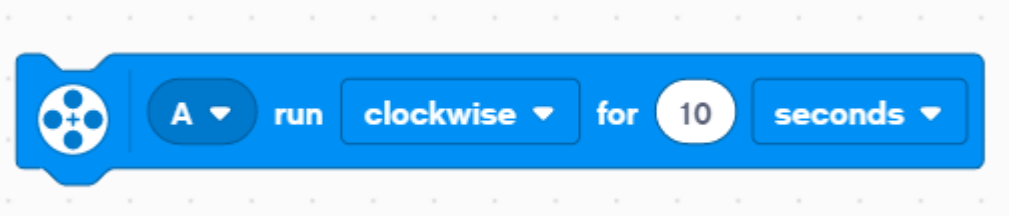
Představuje začátek všech programů. Má obdélníkový tvar. Spodní stranu se zobáčkem umožňující připojení dalších bloků a zakulacený vršek, který neumožňuje připojit před něj žádný jiný blok. Start tohoto bloku může být nastaven na prosté spuštění programu, na určitou událost, splnění podmínky, obdržení zprávy nebo vypršení časovače.



Obr. 16 Hat blok

Stack blok

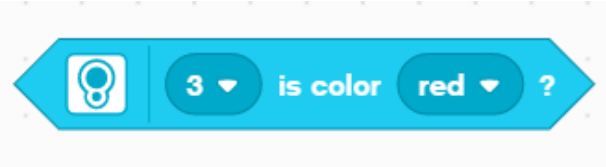
Je nejpoužívanějším blokem v tomto jazyce, protože plní funkci příkazu. Je obdélníkového tvaru se zářezem na horní straně a zobáčkem na spodní, což umožňuje použití tohoto bloku kdekoli v programu mezi jeho začátkem a koncem. Umožňuje spouštět či zastavovat motory, ovládat pohyb, upravovat obsah zobrazení displeje, přehrávat zvuky a mnoho dalších věcí. Nastavení různých hodnot probíhá pomocí textových a číselných inputů nebo pomocí kombo boxů.



Obr. 17 Stack blok

Boolean blok

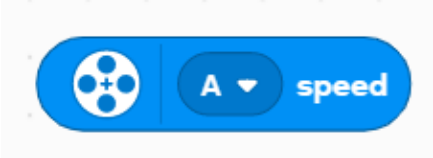
Tento blok představuje podmínku. Může nabývat pouze dvou hodnot, a to pravda nebo lež. Má tvar šestiúhelníku a umístit jej lze pouze do stejně tvarovaných otvorů uvnitř ostatních bloků. Použít je lze při vyhodnocení podmínky, pro textový výstup nebo ukládání hodnoty do proměnné.



Obr. 18 Boolean blok

Report blok

Je oválovitého tvaru a v programech se využívá pro vrácení určité hodnoty. Programové prostředí jej umožňuje umístit jen do stejně tvarovaných otvorů uvnitř dalších bloků. Využít jej lze například pro získání rychlosti motoru, zjištění barvy ze senzoru barev, nabízí také použití matematických operací nebo pro návrat hodnoty z vlastní proměnné.



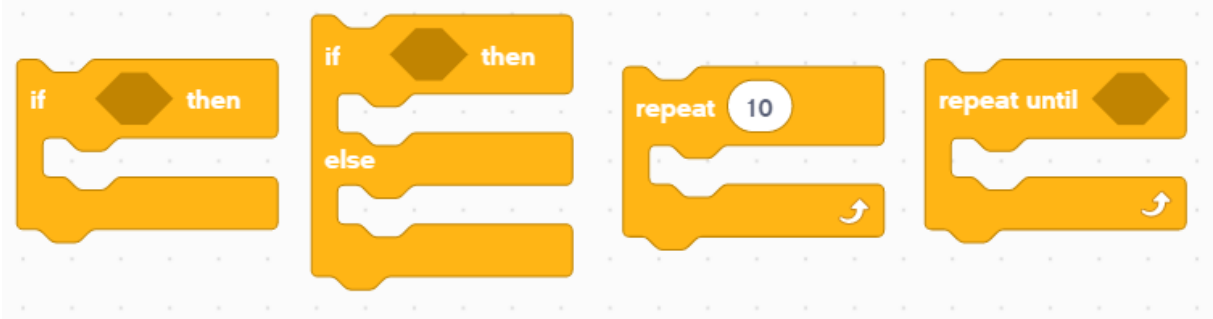
Obr. 19 Report blok

C Blok

Tento blok získal svůj název díky svému specifickému tvaru připomínající tvar písmene C. Označuje se také jako obalovací blok, protože uvnitř nabízí prostor pro jeden či více Stack bloků. Použít se dá na kterémkoli místě mezi začátkem a koncem programu. Využití se dělí na dvě hlavní funkce, a to podmínky a cykly.

Pomocí podmínek lze programy větvit a určovat tak, která část programu se provede. K dispozici je varianta *if-then*, ze které se kód provede pouze při splnění podmínky. Druhou variantou je blok *if-then-else*, díky které lze předepsat navíc také kód pro případy, kdy by podmínka splněna nebyla.

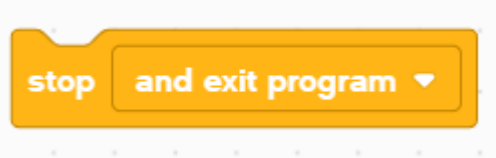
V případě cyklů, které se využívají pro opakování určité části kódu, lze smyčku provádět do splnění určené podmínky, po předem určený počet opakování nebo zvolit nekonečný cyklus.



Obr. 20 C bloky

Cap blok

Je blok obdélníkového tvaru s výřezem na jeho horní straně, které značí konec programu. Jeho tvar zajišťuje, že po jeho použití už nelze připojit žádný další blok. V jeho nastavení jsou k dispozici možnosti, zda při jeho použití má být zastaven pouze aktuální strom programu, všechny stromy programu nebo má být zastaven a vypnut celý program.



Obr. 21 Cap blok

3.2.2 Kategorie bloků

Jednotlivé kategorie bloků rozdělují jednotlivé bloky podle barev a jejich využití z pohledu robota. Jedná se např. o kategorie Motors, Movement, Sensors, Display, Sound atd.

Motors

Tato kategorie bloků je modré barvy a slouží k ovládání a získávání informací z jednotlivých motorů. Nastavovat lze rychlost, sílu, směr otáčení (po/proti směru hodinových ručiček) motoru a dobu po kterou se má akce vykonávat. Nezaměňovat však s bloky movement, které se však využívají pro odlišné účely. Použít je lze např. pro ovládání robotické paže.

Movement

Barva této kategorie je růžová. Slouží k práci s dvěma motory na jedou. Využívají se především při pohybu robotů. Synchronizovat lze pouze dva motory stejného typu, tzn. dva střední nebo dva velké motory. Nastavit je možno rychlost, dobu nebo směr jakým se má robot otáčet.

Display

Je kategorie světle fialové barvy s jejíž pomocí lze určovat obsah, který se má zobrazovat na displeji na přední straně EV3 kostky. Na výběr je z velkého množství předdefinovaných výrazů obličeje, očí, různých ikon, informací o připojených motorech a senzorech nebo vlastní text. Mimo to jde také nastavovat barvu a případné problikávání podsvícení tlačítek.

Sound

Bloky Sound mají tmavě fialovou barvu a s jejich pomocí se pracuje se zvuky. V nabídce zvuků lze najít různá krátká anglická slova či fráze, čísla od 0 do 10, různé barvy, zvuky zvířat, lidí,

mechanických zvuky a mnoho dalších. Další možností je nastavení úrovně hlasitosti nebo přehrání tónu, který si lze vybrat pomocí zobrazovaného piana.

Events

Tyto bloky jsou žluté barvy a jejich účelem je zachytávání nebo vytváření událostí. S výjimkou jednoho jsou všechny bloky tvaru Hat neboli počáteční, tudíž se musí vyskytovat na začátku programu. Najdeme mezi nimi běžný startovní blok nebo start podmínění konkrétní událostí, což může být určitý vstup na jednom ze senzorů, stisknutí tlačítka, splnění vlastní podmínky, obdržení zprávy nebo vypršení času časovače. Jediným blokem, který není zahajovacím je pak blok *broadcast*, který vysílá zadanou zprávu, na kterou mohou reagovat jiné programy.

Control

V této kategorii s barvou mezi tmavě žlutou a oranžovou, se ukrývají řídící struktury, které rozhodují o dalším průběhu programu. Pro větvení či zacyklení programu lze využít dříve zmíněné podmínky a cykly. Dále jsou k dispozici bloky *wait* pro pozdržení programu po určitou dobu nebo do splnění podmínky a blok *stop* pro úplné zastavení programu.

Sensors

Další kategorií bloků je kategorie Sensors, vyznačující se tyrkysovou barvou. Díky nim lze pracovat s jednotlivými senzory, získávat z nich informace nebo pozastavovat program, dokud nebude splněna podmínka určená vstupem ze senzoru.

Operators

Tyto bloky zelené barvy slouží pro výpočet různých matematických operací. Od těch jednoduchých jako jsou sčítání, odečítání, násobení, dělení přes zaokrouhlování, zbytek po celočíselném dělení, absolutní hodnotu, generování náhodných hodnot až po goniometrické funkce, logaritmy a další. Dále jsou zde porovnávací operátory menší než, větší než, rovná se, logické operátory *and* (logický součet), *or* (logický součin) a *not* (negace), popřípadě další bloky pro práci se stringy, jako určení jejich délky či spojení.

Variables

V těchto oranžových blocích si lze vytvářet vlastní proměnné nebo jednoduché pole proměnných. Hodnoty, které můžeme vkládat, jsou buďto celá čísla nebo text. U proměnných lze po vytvoření získávat jejich hodnotu, nastavovat novou hodnotu, případně ji zvětšovat nebo zmenšovat o určitou číselnou hodnotu, pokud se jedná o číselnou proměnnou. U jednoduchých

polí je možno přidávat prvky, nahrazovat určité prvky jinými, získávat hodnoty jednotlivých prvků, mazat obsah celého pole nebo zjišťovat počet prvků v poli.

My Blocks

Poslední kategorií bloků je My Blocks, která má červenou barvu. Pomocí ní je možno vytvářet vlastní bloky. Při definici bloku nastaví jeho jméno, až 9 různých vstupních parametrů libovolného typu a textové popisky uvnitř bloku. Následně se do definice umístí program, který chceme, aby blok zastupoval. Po vytvoření lze pak již používat nový blok na libovolném místě, v libovolném počtu i ve více programech. Zlepší se tak čitelnost kódu a odstraní se duplikace.

3.3 MicroPython

Další možností jak programovat roboty Lego Mindstorms je použití jazyku MicroPython, který je již vyšším programovacím jazykem, zapisovaný běžným způsobem pomocí textu. Jedná se o jazyk založený na Pythonu 3, který je optimalizován pro provoz na jednočipových počítačích.

Pro použití tohoto jazyka je nutné mít, krom samotné stavebnice, také MicroSD paměťovou kartu a počítač s operačním systémem Windows nebo macOS s možností čtení a zápisu na paměťové karty.

Pro zprovoznění je nejprve nutné nahrát do kostky speciální firmware, který je k dispozici ke stažení na oficiálních stránkách Lego Education. Dále je také potřeba mít nainstalován libovolný „vypalovací“ program (např. Etcher), který dokáže vytvořit „bootovací“ paměťové médium. Po stažení firmwaru je nutné takové to médium vytvořit. Po úspěšném vytvoření pak stačí vypnout EV3 kostku, vložit paměťovou kartu do kostky a znovu spustit. Pro návrat k původnímu firmwaru stačí pouze kostku vypnout a MicroSD kartu vyjmout.

Pro samotné psaní kódu je třeba mít nainstalován editor zdrojových kódů Visual Studio Code a v něm doplněk nazvaný Lego Mindstorms EV3 MicroPython. Po jeho nainstalovaní se pak objeví toto rozšíření v nabídce rozšíření a lze pomocí něj vytvářet nové či otevírat již stávající projekty.

Při vytvoření nového projektu je vytvořen adresář, který může obsahovat jednotlivé soubory s programy či zvukové nebo obrázkové soubory. Tento adresář je při spuštění vybraného programu vždy stažen do EV3 kostky a spuštěn. Každý vytvořený projekt obsahuje soubor nazvaný *main.py* s hlavním programem.

Pro samotné programování se využívají Pybrick moduly. Každý z těchto modulů obsahuje třídy pro práci s jednotlivými částmi robota. Seznam jednotlivých modulů:

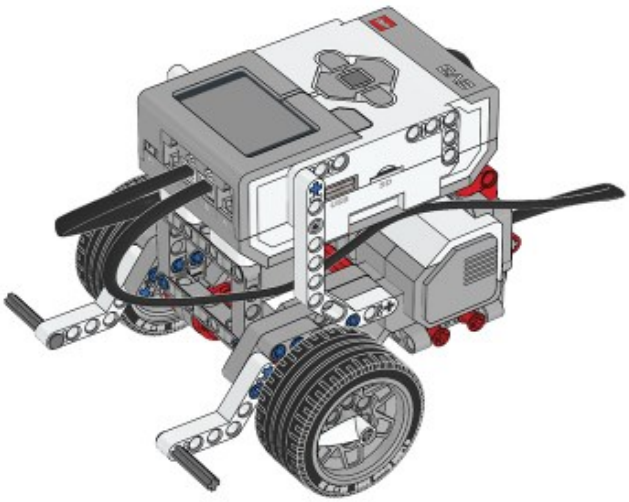
- *hubs* – obsahuje třídu EV3Brick pro práci s kostkou, jejím displejem a zvuky
- *ev3devices* – třídy pro práci s motory a senzory
- *nxtdevices* – třídy pro práci s motory a senzory z předchozí verze stavebnice Lego Mindstorms NXT
- *iodevices* – třídy pro práci s neoficiálními senzory, motory nebo vlastní elektronikou
- *parameters* – třídy s konstantami využívající se pro parametry a argumenty (názvy portů, barvy, tlačítka atd.)
- *tools* – metoda *wait*() pro pozastavení programu a třídy pro práci s časovačem a pro jednoduché logování dat
- *robotics* – třída *DriveBase* pro programování pohybu robota
- *media* – třídy pro práci s fonty, zvukovými a obrázkovými soubory
- *messaging* – třídy pro propojení a následnou komunikaci více EV3 kostek mezi sebou
- *geometry* – třídy pro využití matic, vektorů či směrových os v programu

4 ÚLOHY PRO ROBOTICKÉ STAVEBNICE

V této kapitole bude představeno postupně 12 úloh, které jsou určené pro základní seznámení se se stavebnicí Lego Mindstorms EV3 a jejím programováním. Úlohy využívají základní programové konstrukce jako jsou podmínky či cykly, vytváření vlastních bloků, práci s motory pro pohyb robota či interakcí s jeho okolím za použití různých senzorů.

Před každou úlohou je vhodné nejprve studenty seznámit s typy bloků a senzory, které budou v dané úloze potřeba použít, vysvětlit jakou zde mají funkci a jakým způsobem se s nimi pracuje. Řešení úloh jsou pouze ukázková, vždy může existovat více možných správných řešení.

Pro ukázková řešení je využita stavebnice Lego Mindstorms Education EV3 Core Set doplněná o infračervený senzor a maják. Z této stavebnice je sestaven základní robot Driving Base [21], který je vždy přizpůsoben dané úloze. K řešením lze využít také jiné roboty, ať už sestavené dle oficiálních návodů nebo vlastní výroby, je však nutné počítat, že řešení úloh se mohou mírně lišit.



Obr. 22 Robot Driving Base [21]

Řešení jsou prezentována ve dvou verzích. Jedno vytvořené v aplikaci EV3 Classroom za pomocí grafického jazyka Scratch 3.0 vhodného spíše pro mladší studenty. Druhá verze řešení je vytvořená v programu Visual Studio Code s rozšířením Lego Mindstorms EV3 MicroPython v jazyce MicroPython.

V ukázkových řešeních s použitím MicroPythonu jsou části, které se vždy opakují zobrazeny pouze v první úloze a následně jsou již vynechávány. Jedná se o importy tříd z knihoven, inicializaci EV3 kostky, motorů a řídící jednotky.

4.1 Základní pohyb robota

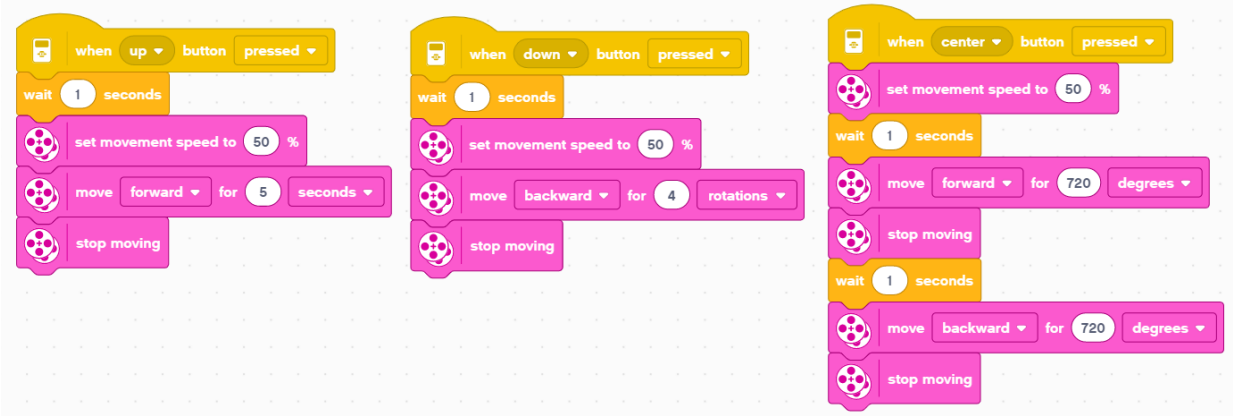
Vytvořte program, kdy po spuštění programu bude robotické vozítko vyčkávat na stisknutí tlačítka na EV3 kostce. Po stisknutí šipky nahoru provede robot po krátké prodlevě pohyb vpřed. Při stisknutí šipky dolů pak robot provede pohyb směrem vzad. Při stisknutí prostředního tlačítka obě tyto akce spojte nechte robota popojet vpřed a následně vzad. Pro řešení je možné využít různé nastavení motorů, vyzkoušejte různé varianty rychlostí a možností jak nastavit dobu po kterou robot bude popojíždět (sekundy, otáčky, stupně).

Řešení

Pro řešení úlohy je nutné kontrolovat stisknutí jednotlivých tlačítek a následně pozastavit program a spustit pohyb příslušným směrem. V jazyce Scratch lze využít bloky událostí, reagující na jednotlivá stisknutí a následně bloky *wait* a *movement* pro pozastavení a pohyb.

V MicroPythonu je nejprve nutné inicializovat EV3 kostky, použité motory a řídící jednotku pomocí *DriveBase()* s parametry proměnných motorů a hodnotami průměru kola (uváděno na pneumatikách) a rozchodu kol. Následně je třeba vytvořit cyklus, ve kterém budou neustále kontrolována stisknutá tlačítka. Pro pozastavení programu slouží metoda *wait(čas: ms)* a pro nastavení pohybu lze použít metody *straight(vzdálenost: mm)*, *drive(rychlost: mm/s, směr: stupně/s)* či *drive_time(rychlost: mm/s, směr: stupně/s, čas: ms)*. Pro pohyb směrem vzad je nutné uvést rychlost či vzdálenost v záporných hodnotách. Pro získání stisknutých tlačítek se využívá atribut a metoda EV3 kostky *buttons.pressed()*.

Řešení v jazyce Scratch



Obr. 23 Řešení úlohy 1 v jazyce Scratch

Řešení v jazyce MicroPython

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# inicializace EV3 kostky
ev3 = EV3Brick()

# inicializace motorů
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)

# průměr kola v mm a vzdálenost mezi jednotlivými koly v mm
wheel_diameter = 56
axle_track = 114

# inicializace řídící jednotky
robot = DriveBase(left_motor, right_motor, wheel_diameter, axle_track )

while True:
     while not any(ev3.buttons.pressed()):
         wait(10)

     if Button.UP in ev3.buttons.pressed():
         wait(500)
         robot.straight(1000)
         robot.stop()

     if Button.DOWN in ev3.buttons.pressed():
         wait(500)
         robot.straight(-1000)
         robot.stop()

     if Button.CENTER in ev3.buttons.pressed():
         wait(500)
         robot.straight(1000)
         robot.stop()
         wait(500)
         robot.straight(-1000)
         robot.stop()
```

4.2 Robotická minutka

Naprogramujte robota, který bude plnit funkci kuchyňské minutky, na které nastavíte čas a po jeho uplynutí robot začne vydávat zvukové a světelné signály. Pomocí šipek nahoru a dolu na EV3 kostce bude možnost přidávat a odebírat sekundy, které se bude následně čekat. Aktuální hodnota se bude vždy zobrazovat na displeji. Minutka se bude spouště pomocí prostřední tlačítka kostky a po uplynutí dané doby bude robot vydávat zvuk a blikat.

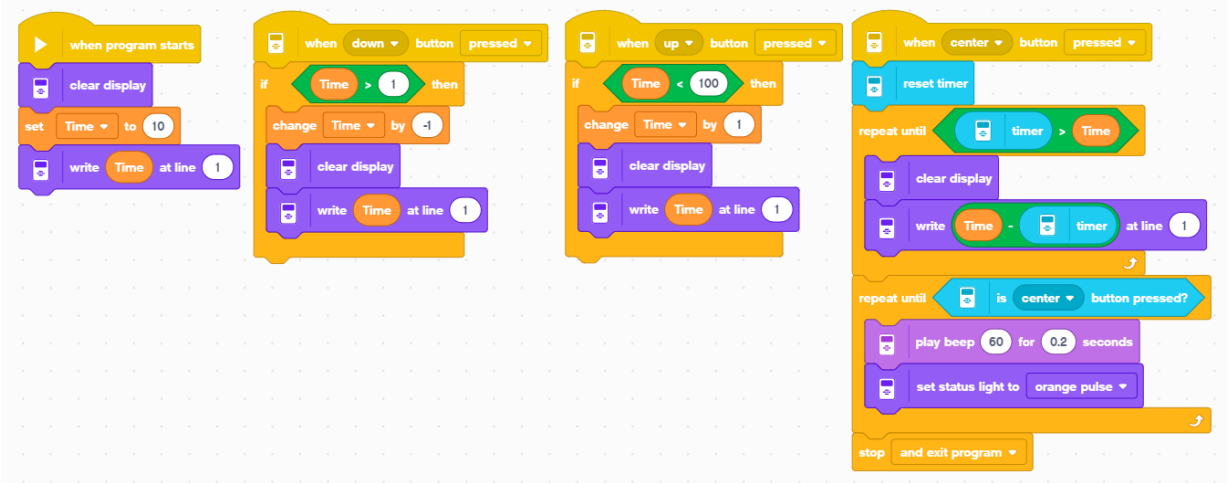
Jako modifikaci úkolu zajistěte, aby se po spuštění minutky na displeji objevoval čas, který zbývá do konce časomíry. Kontrolujte také zadané vstupy, aby se požadovaný čas nemohl dostat do mínusu.

Řešení

Pro vyřešení úlohy je třeba využití podmínek, cyklů a vlastní proměnné *time*. V jazyku Scratch použijte opět události pro reakci na stisknutí jednotlivých tlačítek. Při snižování a zvyšování času vždy kontrolujte pomocí bloků *if-else*, zda hodnota není příliš nízká respektive vysoká. Při spuštění odpočtu nejprve resetujte *timer* a spusťte cyklus do té doby, než se hodnota v časovači bude rovnat požadovanému času. V cyklu pak postupně zobrazujte čas, který zbývá dokonce odečítáním aktuálního času od cílového času. Po uplynutí času následuje další cyklus, dokud nebude stisknuto opět prostřední tlačítko, ve kterém bude přehráván zvuk a nastavena světelná signalizace.

V MicroPythonu je postup obdobný, místo událostí však použijte nekonečnou smyčku. Stisknutá tlačítka jsou k dispozici díky proměnné s inicializovanou kostkou pomocí atributu a metody *buttons.pressed()*, která vrací všechna právě stisknutá tlačítka. Dále budete z kostky využívat metody *speaker.beep(frekvence: Hz, čas: ms)* pro přehrání pípnutí, *light.on(barva)* pro světelnou signalizaci, *screen.clear()* pro vyčištění displeje a *screen.print()* pro zobrazení textu na displeji. Pro použití časovače jej nejprve musíte inicializovat pomocí *StopWacth()* a následně s ním můžete pracovat pomocí metod *reset()* pro resetování a *time()* pro získání aktuálního času v milisekundách.

Řešení v jazyce Scratch



Obr. 24 Řešení úlohy 2 v jazyce Scratch

Řešení v jazyce MicroPython

```
# inicializace časovače a základního času
watch = StopWatch()
time = 10

while True:
    ev3.screen.clear()
    ev3.screen.print(time)

    while not any(ev3.buttons.pressed()):
       wait(10)

    if Button.CENTER in ev3.buttons.pressed():
        watch.reset()
        while watch.time() / 1000 > time:
             ev3.screen.clear()
             ev3.screen.print(time - watch.time()/1000)
        while Button.CENTER in ev3.buttons.pressed():
             ev3.speaker.beep(1000, 500)
             ev3.light.on(Color.RED)
             wait(500)
        break

    if Button.UP in ev3.buttons.pressed():
        if time < 100:
            time += 1

    if Button.DOWN in ev3.buttons.pressed():
        if time > 1:
            time -= 1
```

4.3 Základní otáčení robota

Úkolem je naprogramovat robota tak, aby po spuštění programu provedl pohyb vpřed, následně se otočil o 180° a vrátil se na původní místo. Vyzkoušejte různé hodnoty otáčení a pozorujte jak se robot následně otáčí.

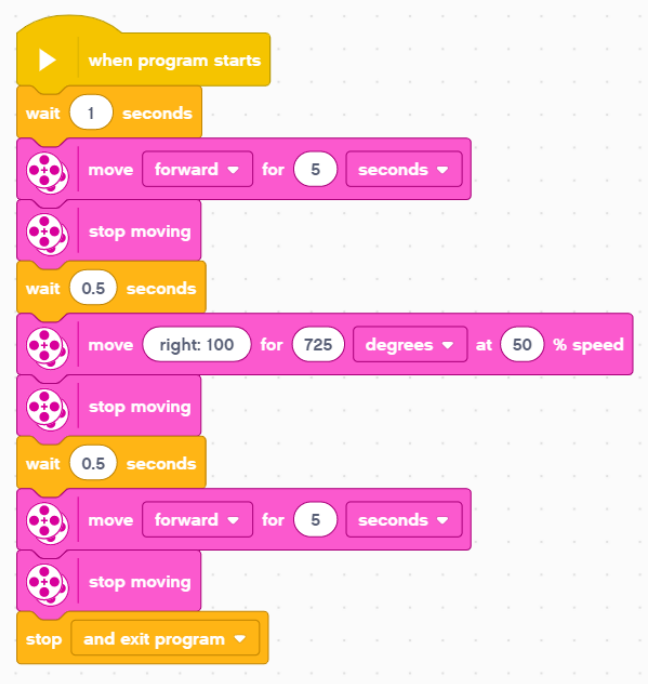
Pozor při nastavování parametrů otáčení motorů. Nastavení parametru otáčení motoru na hodnotu 180° nevede k otočení robota o 180°, jedná se o stupně otáček motoru, nikoliv robota.

Řešení

Pro jazyk Scratch nelze bez použití gyroskopického senzoru nastavit přesný úhel otočení robota, lze nastavit pouze směr otáčení v intervalu od -100 do 100. Pro nalezení správných parametrů je možné využít metody pokus-omyl a postupně upravovat dobu po kterou se robot má otáčet. Druhou možností je si nechat zobrazit na displeji port s připojeným motorem a následně provést manuálně otočení robota o 180°. Na displeji se bude průběžně zobrazovat počet otáček motoru, které byly provedeny a ty využijete při řešení. Výsledný parametr je opět nutné vyzkoušet v praxi a případně doupravit.

V jazyce MicroPython lze po inicializaci řídící jednotky využít metody *turn(úhel: stupně)*, která díky znalostem parametrů robota, dokáže otočit robota o požadovaný úhel. Pro pohyb vpřed a vzad je opět využita metoda *straight(čas: ms)*.

Řešení v jazyce Scratch



Obr. 25 Řešení úlohy 3 v jazyce Scratch

Řešení v jazyce MicroPython

```
robot.straight(2000)
wait(500)

# otočení robota o 180°
robot.turn(180)
wait(500)

robot.straight(2000)
robot.stop()
```

4.4 Ovládání robota pomocí infračerveného senzoru a ovladače

Naprogramujte robota tak, aby bylo možné jeho pohyb ovládat pomocí tlačítek na infračerveném ovladači. Reakce na stisknutí jednotlivých tlačítek nastavte na pohyb vpřed, vzad, otočení doprava a doleva. Na základě zvoleného směru také na displeji zobrazte oči robota, které se budou vždy dívat směrem kterým jede.

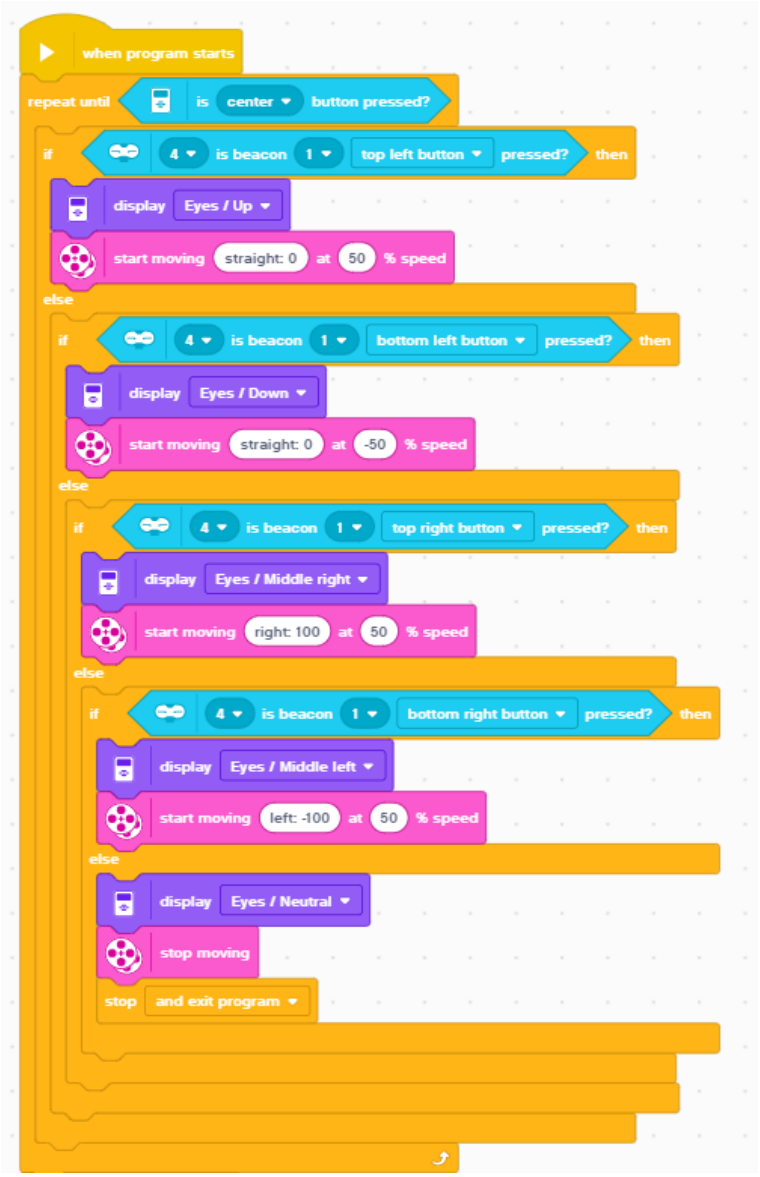
Před spuštěním programu je nutné mít nastavené infračervený senzor i ovladač na stejném kanále, přes který budou následně komunikovat.

Řešení

Pro vyřešení této úlohy využijte smyčku, která bude probíhat, dokud nebude stisknuto prostřední tlačítko na kostce. V této smyčce budete postupně kontrolovat, zda nejsou stisknuta tlačítka na infračerveném ovladači a následně bude spuštěn nebo zastaven pohyb robota a zobrazeny oči.

V jazyce MicroPython je nutné nejprve inicializovat infračervený senzor, ze kterého bude využívána metoda *buttons(kanál)*, která přijímá jako parametr vysílací kanál. Vrací seznam stisknutých tlačítek. Pro zobrazovaní obrázků se používá atribut a metoda kostky *screen.load_image(obrázek)*, kde lze jako obrázek využít jednu z konstant ze třídy *ImageFile* ukládající základní obrázky.

Řešení v jazyce Scratch



Obr. 26 Řešení úlohy 4 v jazyce Scratch

Řešení v jazyce MicroPython

```
# inicializace infračerveného senzoru
infrared_sensor = InfraredSensor(Port.S1)

while True:
     while not any(infrared_sensor.buttons(1)):
        wait(10)

    if Button.LEFT_UP in infrared_sensor.buttons(1):
        ev3.screen.load_image(ImageFile.UP)
        robot.drive(200, 0)
    elif Button.LEFT_DOWN in infrared_sensor.buttons(1):
        ev3.screen.load_image(ImageFile.DOWN)
        robot.drive(-200, 0)
    elif Button.RIGHT_UP in infrared_sensor.buttons(1):
        ev3.screen.load_image(ImageFile.RIGHT)
        robot.drive(0, 90)
    elif Button.RIGHT_DOWN in infrared_sensor.buttons(1):
        ev3.screen.load_image(ImageFile.LEFT)
        robot.drive(0, -90)
    else:
        ev3.screen.load_image(ImageFile.NEUTRAL)
        robot.stop()
```

4.5 Základní použití dotykového senzoru

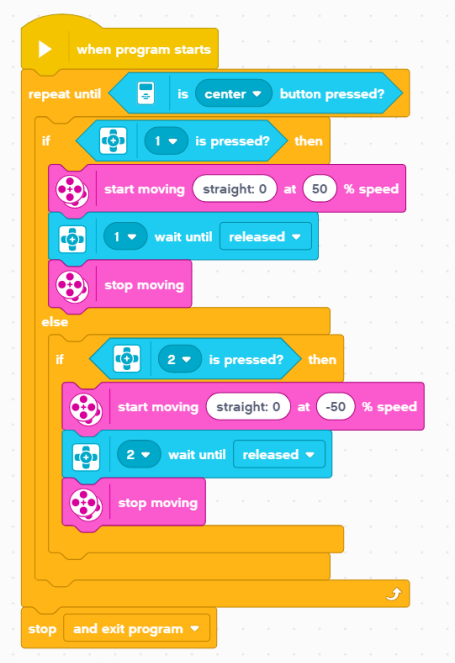
Naprogramujte robota s využitím dvou dotykových senzorů, které budou fungovat jako tlačítko pro spuštění pohybu vpřed a vzad. Robot se bude pohybovat, jen pokud je tlačítko stisknuté. Program ukončete stisknutím prostředního tlačítka na kostce.

Řešení

Pro řešení tohoto úkolu v jazyce Scratch je třeba využít bloky senzorů, ze kterých je třeba vybrat blok pro čekaní na stisknutí/uvolnění dotykového senzoru. Ten použijte tím způsobem, že budete čekat na stisknutí senzoru. Následně zahajte pohyb a čekejte, než bude tlačítko uvolněno a pohyb následně zastavte. Pro možnost opětovného použití umístěte tyto bloky do smyčky.

Pro MicroPython je nejprve nutné inicializovat dotykový senzor pomocí *TouchSensor(port)*, ze kterého se následně bude používat metoda *pressed()* při kontrole stisknutí a uvolnění tlačítka. Pro pohyb a zastavení robota budou využity metody *drive(rychlost: mm/s, směr: stupně/s)* a *stop()*.

Řešení v jazyce Scratch



Obr. 27 Řešení úlohy 5 v jazyce Scratch

Řešení v jazyce MicroPython

```
# inicializace dotykového senzoru
touch_sensor_forward = TouchSensor(Port.S1)
touch_sensor_backward = TouchSensor(Port.S2)

while True:
    if touch_sensor_forward.pressed() == True:
        robot.drive(200, 0)
    elif touch_sensor_backward.pressed() == True:
        robot.drive(-200, 0)
    else:
        robot.stop()
```

4.6 Základní použití barevného senzoru

K tomuto úkolu je třeba krom robota mít k dispozici barevnou pásku nebo papír s nakreslenou barevnou čarou. Cílem je napsat program, kdy se robot bude pohybovat, dokud nenarazí na danou barvu. Poté zastaví a řekne název barvy a zobrazí její název na displeji.

Jako rozšíření lze doplnit detekci více barev než pouze jedné.

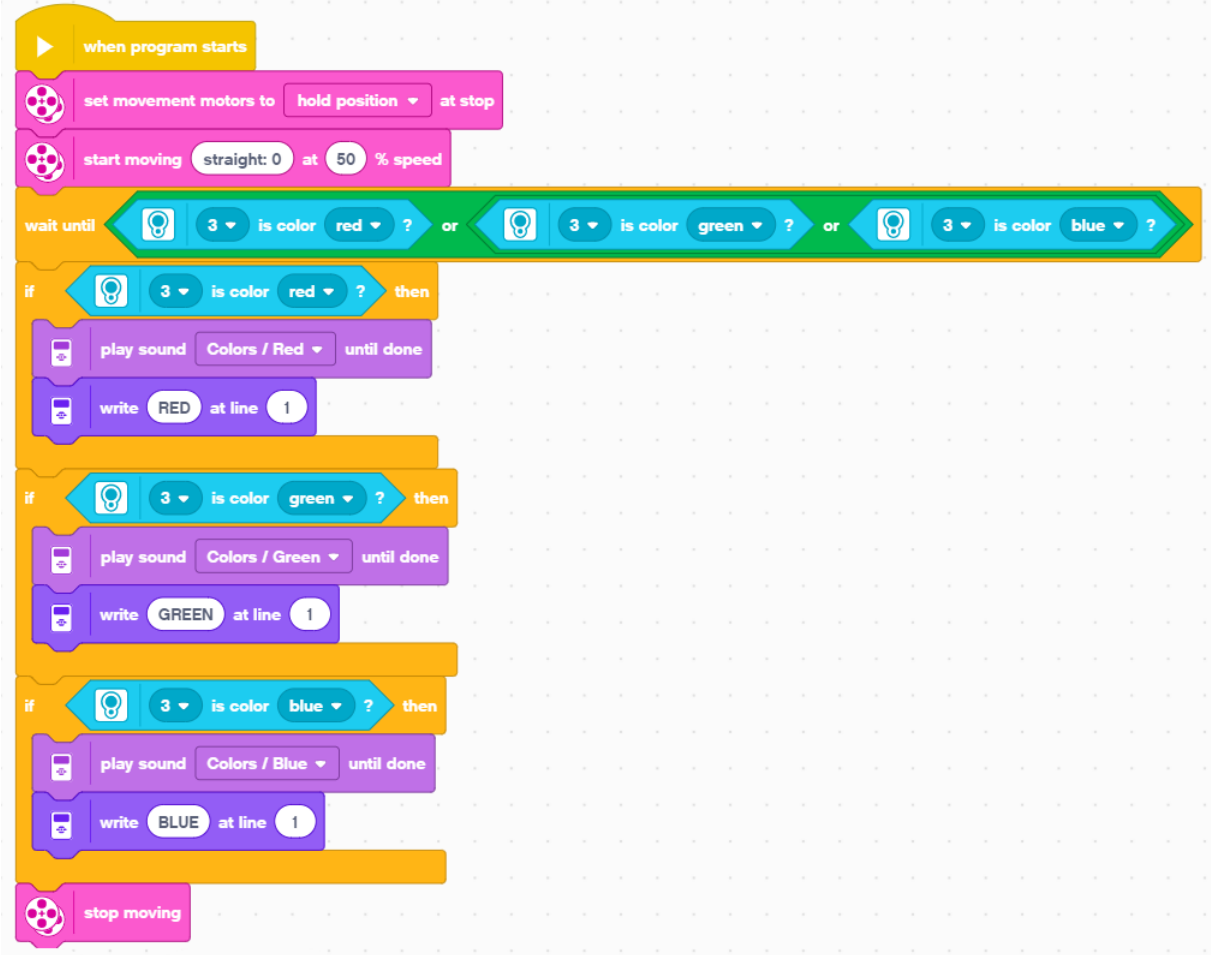
Tip: Pro okamžité zastavení robota na místě lze použít blok *set movement motors* a nastavit jej na hodnotu *hold position*. Při tomto nastavení se kola po zastavení zablokují.

Řešení

K splnění tohoto zadání v Scratchi nejprve nastavíte pro motory hodnotu *hold position*, aby se robot zastavil ihned na místě při detekci dané barvy. Následně spustíte pohyb robota a pomocí bloku *wait until* čekejte na rozpoznání dané barvy. Pro více barev je třeba využít bloky *or* a jednotlivé podmínky. Pro přehrání správného zvuku a zobrazení textu je nutné použít podmínku, aby robot zjistil na kterou barvu narazil. Následně přehrajte zvuk blokem *play sound* a zobrazte text pomocí bloku *write*.

V MicroPythonu je třeba inicializovat barevný senzor pomocí *ColorSensor(Port)*, ze kterého využijte metodu *color()*, která vrací aktuální naměřenou barvu. Tu lze porovnávat s předem definovanými barvami ze třídy *Color*.

Řešení v jazyce Scratch



Obr. 28 Řešení úlohy 6 v jazyce Scratch

Řešení v jazyce MicroPython

```
# inicializace barevného senzoru
color_sensor = ColorSensor(Port.S1)

while True:
    robot.drive(200, 0)
    if color_sensor.color() == Color.RED:
        ev3. speaker.play_file(SoundFile.RED)
        break
    if color_sensor.color() == Color.GREEN:
        ev3. speaker.play_file(SoundFile.GREEN)
        break
    if color_sensor.color() == Color.BLUE:
        ev3. speaker.play_file(SoundFile.BLUE)
        break

robot.stop()
```

4.7 Základní použití ultrasonického senzoru

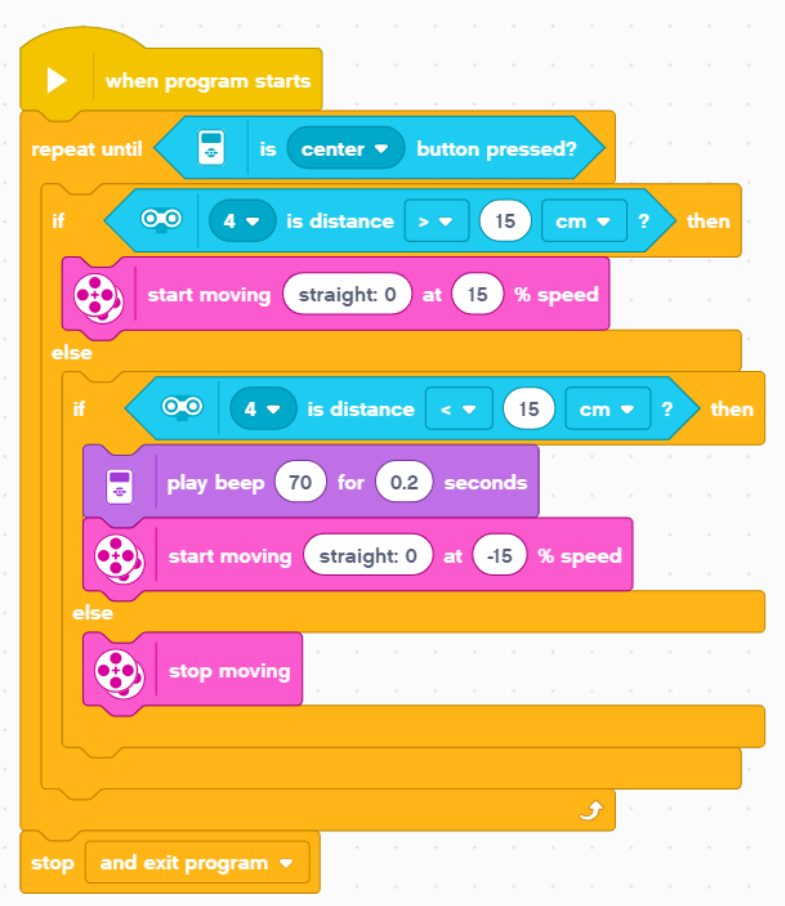
Vytvořte program, který bude využívat ultrasonického senzoru pro měření vzdálenosti od ostatních předmětů. Robot se bude pohybovat, dokud se nepřiblíží k překážce na vzdálenost 15 cm a následně zastaví. Zajistěte také, aby při umístění překážky do vzdálenosti menší než 15 cm robot vydal zvukový signál a následně couvnul nazpátek na požadovanou vzdálenost.

Řešení

Řešení úlohy v obou jazycích umístěte do cyklu pro zajištění pravidelné kontroly vzdálenosti od objektů. Následně do této smyčky umístěte rozhodovací blok *if-else*, kde jako podmínku uveďte, zda je vzdálenost od překážky větší než požadovaných 15 cm. Pokud ano, spusťte pohyb robota vpřed. V bloku *else* pak umístěte další blok *if-else*, který bude kontrolovat, zda je vzdálenost menší než 15 cm. Pokud ano, je přehrán zvuk a zahájen pohyb směrem vzad. Při nesplnění této podmínky je pohyb motorů zastaven. Při řešení je vhodné využít nižší rychlosti motorů.

V jazyce MicroPython se vzdálenost od objektů měří pomocí ultrazvukového senzoru z třídy *UltrasonicSensor(port)* a metody *distance()*, která vrací vzdálenost v milimetrech.

Řešení v jazyce Scratch



Obr. 29 Řešení úlohy 7 v jazyce Scratch

Řešení v jazyce MicroPython

```
# inicializace ultrazvukového senzoru
ultrasonic_sensor = UltrasonicSensor(Port.S1)

while True:
    if ultrasonic_sensor.disnace() > 1500:
        robot.drive(200, 0)
    elif ultrasonic_sensor.disnace() < 1500:
         # přehraje zvuk o frekvenci 500 Hz po dobu 1000 ms
        ev3.speaker.beep(500, 1000)
        robot.drive(-200, 0)
    else:
        robot.stop()
```

4.8 Pohyb robota v pravidelném mnohoúhelníku

Cílem tohoto úkolu je vytvořit program, při kterém robot svým pohybem opíše tvar pravidelného šestiúhelníku. Pro přesnější nastavení úhlu, o který se má robot otočit, využijte gyroskopický senzor, který dokáže měřit velikost provedeného úhlu.

Navíc abyste předešli zbytečnému opakování stejného kódu, vytvořte vlastní blok či metodu, která se bude dát použít vícekrát s různými vstupy a pro různé pravidelné n-úhelníky. Zajistěte také kontrolu a případné upozornění na neplatné vstupy.

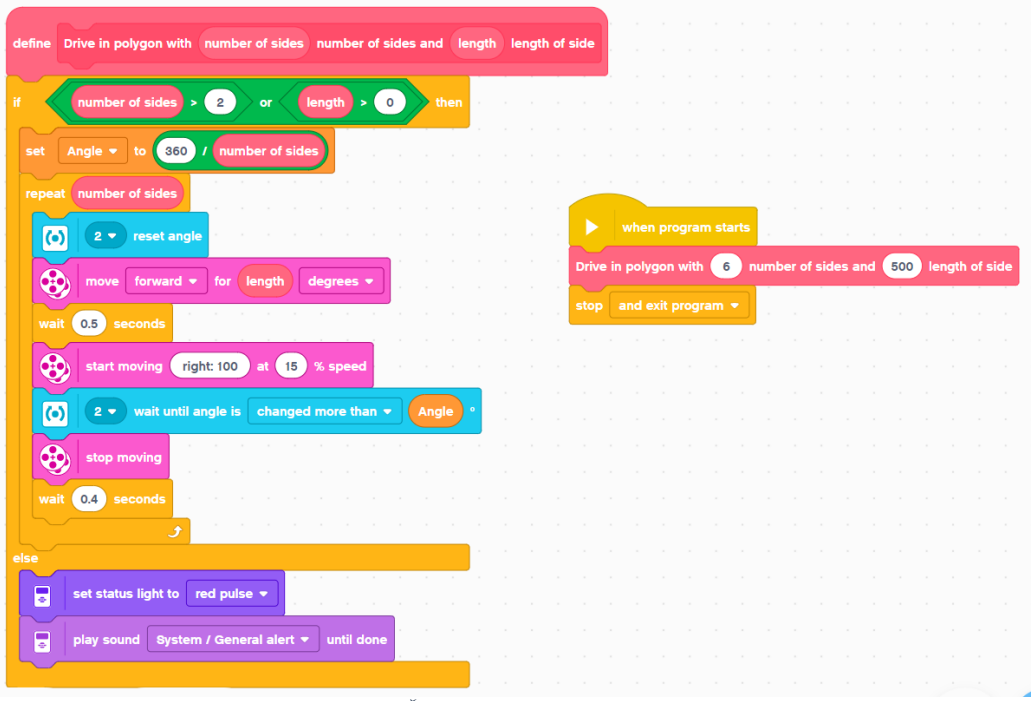
Řešení

Prvním krokem ve Scratchi je vytvoření obecného bloku pro vykonávání pohybu robota ve tvarech různých pravidelných n-úhelníků. Jako vstup bude tento blok přijímat počet stran a délku jedné strany. V tomto bloku nejprve proveďte kontrolu za pomocí podmínky, zda jsou na vstupu minimálně 3 strany a délka strany je větší než 0. Pokud tato podmínka nebude splněna, spusťte varovný zvuk a světelnou signalizaci. V případě splnění podmínky je nutné zjistit úhel o který bude muset robot u každého vrcholu zatočit. Ten lze zjistit jednoduchým výpočtem $\frac{360°}{počet\ stran}$. Pro výsledek tohoto výpočtu je vhodné vytvořit novou proměnou s názvem *angle*, do které bude výsledek přiřazen. Následovat bude smyčka s předem daným počtem opakování, které se bude rovnat počtu stran a uvnitř této smyčky bude vždy proveden pohyb robota směrem vpřed, resetování aktuálně naměřené hodnoty úhlu, spuštění otáčení robota následované blokem, který bude vyčkávat na dosažení požadovaného úhlu a posléze zastavení pohybu otáčení.

Posledním krokem bude spuštění samotného programu, ve kterém bude využit vlastní vytvořený blok pro vykonání pohybu v zadaném mnohoúhelníku, v tomto případě s počtem stran 6 a délkou pohybu např. 500 stupňů.

V MicroPythonu nejprve inicializujte gyroskopický senzor pomocí *GyroSensor(Port)*, ze kterého budete využívat metodu *angle()*, která vrací celkový uražený úhel a metodu *reset()* pro resetovaní hodnoty aktuálně uraženého úhlu. Následně si vytvořte metodu pro provedení n-úhelníku pomocí klíčového slova *def*. Zde je nutné opět kontrolovat platnost vstupů, vypočítat úhel a v cyklu s počtem opakování rovnému počtu stran provádět pohyb vpřed a otáčení se, dokud nebude naměřena gyroskopickým senzorem cílová hodnota úhlu.

Řešení v jazyce Scratch



Obr. 30 Řešení úlohy 8 v jazyce Scratch

Řešení v jazyce MicroPython

```
# inicializace gyroskopického senzoru
gyro_sensor = GyroSensor(Port.S1)

# funkce, která provede pohyb robota v pravidelném n-úhelníku
# vstupy: počet stran, délka strany v mm
def polygon(number_of_sides, length_of_side):
    if number_of_sides > 2 and length_of_side > 0:
        gyro_sensor.reset_angle(0)
        angle = 360 / number_of_sides
        for side in range(1, number_of_sides + 1):
            current_angle = side * angle
            robot.straight(length_of_side)
            wait(100)
            robot.drive(0, 50)
            while gyro_sensor.angle() < current_angle:
                wait(1)
            robot.stop()
            wait(500)
    else:
        ev3.light.on(Color.RED)
        ev3.speaker.beep(1500, 1000, 50)

polygon(6, 400)
```

4.9 Následování zdi

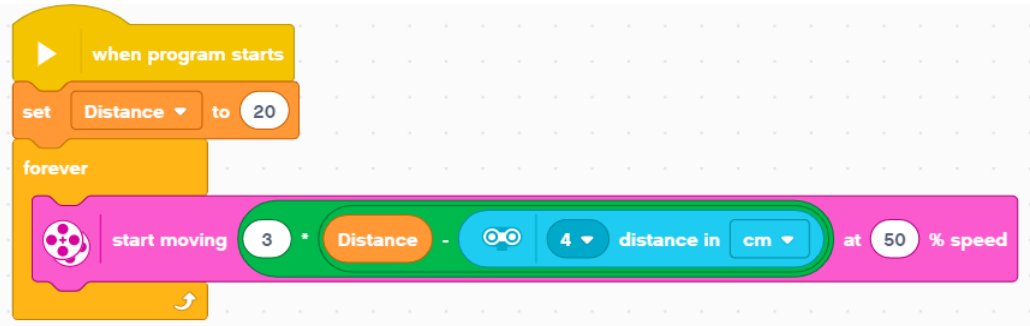
Vytvořte program, pomocí něhož bude robot schopen svou jízdou kopírovat zeď či jinou zábranu a bude od ní udržovat konstantní odstup. Pro řešení lze využít infračervený nebo ultrazvukový senzor, který umístěte přibližně v úhlu 45° mezi přední střední stranou robota a jeho bokem, tak aby robot viděl do strany před sebe a mohl reagovat na změny směru překážky.

Řešení

V programu Scratch opět využijte smyčku, do které umístěte blok *start moving* pro zahájení pohybu robota. Jako hodnotu směru, kterým se má robot ubírat, odečítejte aktuální vzdálenost od vzdálenosti, kterou chceme, aby robot udržoval diagonálně od překážky. Pokud se tyto vzdálenosti budou rovnat, bude jejich rozdíl 0 a robot tak pojede rovně. V ostatních případech bude směr jízdy upraven podle výpočtu směrem doprava nebo doleva. Toto řešení je již funkční, avšak robot špatně reaguje na ostré úhly, proto lze ještě řešení upravit, tak že již vypočtený rozdíl se vynásobí konstantou. Velikost konstanty je závislá na modelu robota, proto je nutné je individuálně přizpůsobit na základě pozorování z praxe. V ukázkovém řešení je použita velikost konstanty 3, která zajistí, že robot zvládne i 90° zatáčky.

V jazyce MicroPython je k řešení použit obdobný algoritmus. Pro měření vzdálenosti od překážky je opět využita metoda *distance()* ultrazvukového senzoru jako u předešlých úloh.

Řešení v jazyce Scratch



Obr. 31 Řešení úlohy 9 v jazyce Scratch

Řešení v jazyce MicroPython

```
# inicializace ultrazvukového senzoru
ultrasonic_sensor = UltrasonicSensor(Port.S1)
DISTANCE = 1500

while True:
    direction = 3*(DISTANCE - ultrasonic_sensor.distance())
    robot.drive(200, direction)
```

4.10 Náhodný pohyb robota kruhu

Naprogramujte robota tak aby, jezdil v oblasti velkého bílého kruhu ohraničeného černou páskou. Pokud se robot dostane na okraj kruhu tak couvne, otočí se o náhodný úhel a poté se znovu rozjede. Zajistěte, aby byla možnost, že se robot bude moct otočit jak doprava, tak doleva. Takto nechte robot se pohybovat po dobu 60 sekund.

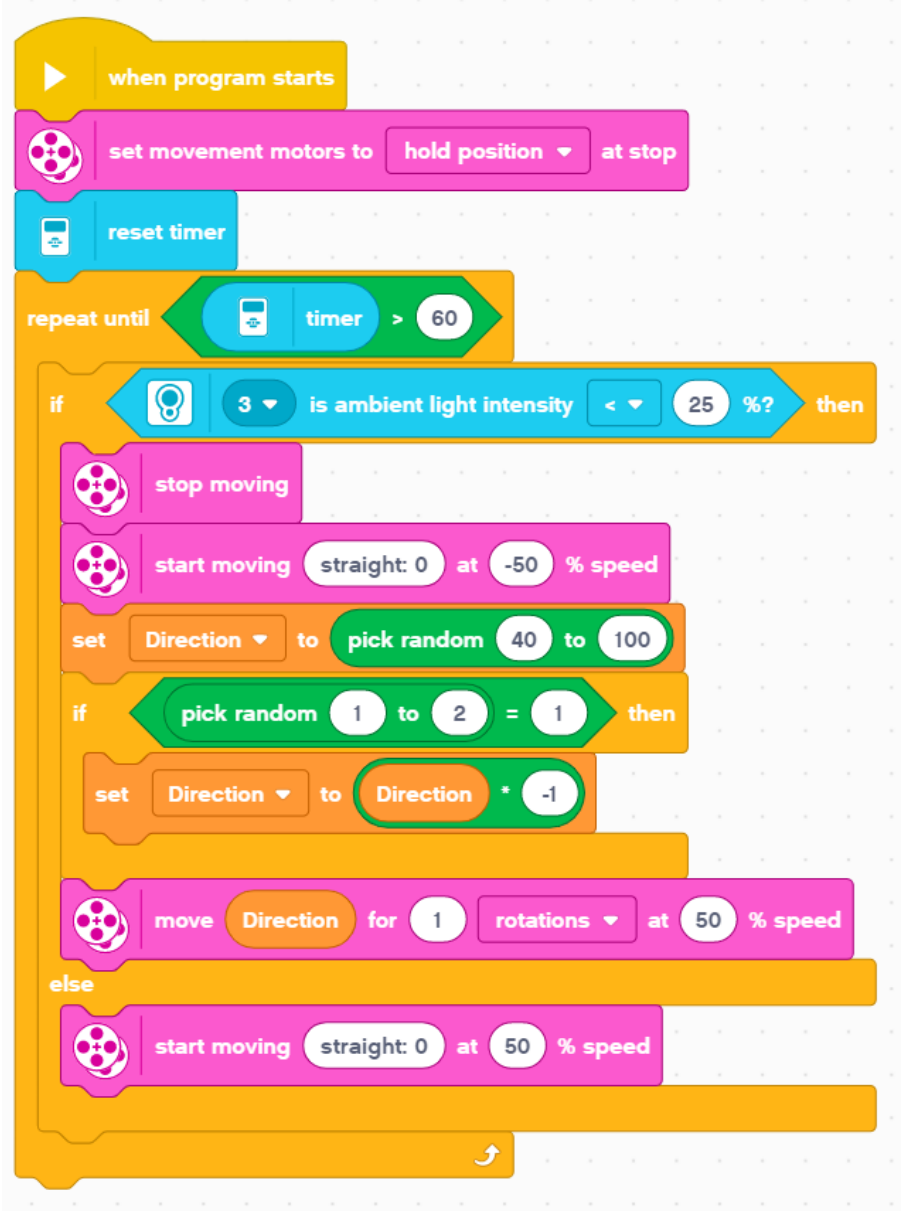
Řešení

Při řešení nejprve nastavte dobu programu na 60 sekund, a to pomocí cyklu, který bude ukončen, jakmile hodnota časovače bude větší než 60. Dále využijte barevného senzoru a jeho režimu intenzity odraženého světla, který dokáže měřit jas světla, který se do senzoru odrazí. Čím je hodnota blíže 0, tím je barva tmavší. Robot se má pohybovat po bílém kruhu ohraničeným černou páskou, proto je třeba nastavit v podmínce kontrolu, zda odražená barva je menší než hranice 25%, která značí, že robot se nachází nad hranicí kruhu. Pokud se robot nachází uvnitř kruhu, nechejte jej se pohybovat se rovně.

Pokud je na okraji, tak robota zastavte a couvněte zpět. Následně je třeba vytvořit novou proměnou *Direction* pro určení náhodného směru, do které bude přiřazena náhodná hodnota pomocí bloku *pick random*. Pro zajištění možnosti otáčení se i na druhou stranu, pro kterou musí být směr uveden v záporných hodnotách, vytvořte jednoduchou podmínku. V té se bude porovnávat hodnota 1 s náhodně generové číslo, tentokrát však pouze z intervalu dvou čísel 1 nebo 2. Pokud bude podmínka splněna, dříve vygenerovaný náhodný směr se vynásobí -1 a získá se tak opačný směr. Tento směr se následně využije pro otočení robota v náhodném směru.

V jazyce MicroPython je nutné opět nejprve inicializovat barevný senzor a časovač. Opět je použit cyklus s ukončením při dosažení času 60 sekund v časovači, který je přístupný pomocí metody *time()*. Následně kontrolujte metodou *reflection()* z barevného senzoru, zda se robot nachází nad tmavým či světlým povrchem. Při naražení na okraj je robot zastaven pomocí metody *robot.stop(Stop.HOLD)*, která zastaví robota na místě. Pro náhodné otočení je využito generování náhodných čísel pomocí *random.randint(min, max)*, kterou je nutné importovat ze knihovny *radnom*.

Řešení v jazyce Scratch



Obr. 32 Řešení úlohy 10 v jazyce Scratch

Řešení v jazyce MicroPython

```
# inicializace barevného senzoru
color_sensor = ColorSensor(Port.S2)
timer = StopWatch()

while timer.time() / 1000 > 60:
    if color_sensor.reflection() < 25:
        robot.stop(Stop.HOLD)
        robot.drive_time(-200, 0, 1000)
        if(random.randint(1,2) == 1):
            robot.turn(random.randint(60, 120))
        else:
            robot.turn(random.randint(-120, -60))
    else:
        robot.drive(200, 0)
```

4.11 Robot reagující na semafor

Vytvořte program pro robota, který zajistí, že při své jízdě bude reagovat na barvy semaforu zelenou, žlutou a červenou. Při zpozorování zelené barvy, se robot rozjede směrem vpřed se zvukem motoru. Při žluté barvě robot zpomalí a po chvíli zastaví, pokud je právě v pohybu. Jestli právě stojí, vydá zvuk nastartování motoru. Při červené barvě okamžitě zastaví a vypne motor.

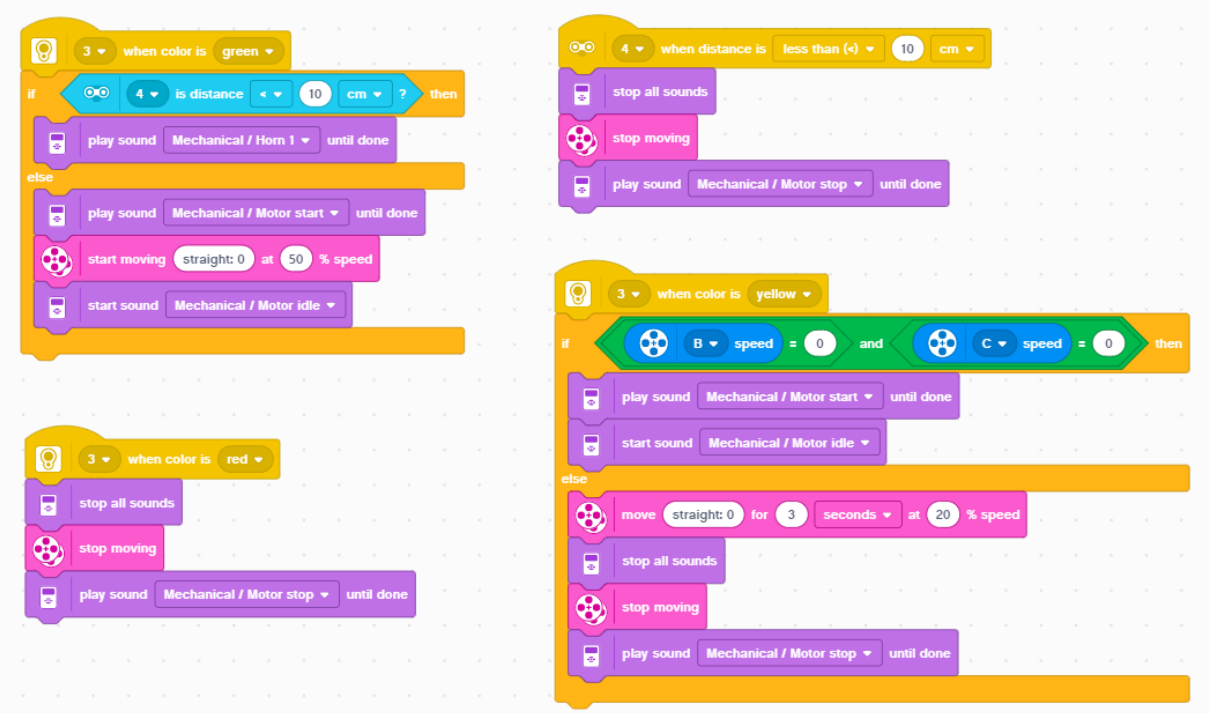
Zajistěte také, aby v případě překážky robot zastavil v bezpečné vzdálenosti. Když robot dostane zelenou a zároveň se nachází před překážkou, namísto spuštění pohybu spusťte klakson.

Řešení

V řešení pomocí jazyka Scratch jsou využity události reagující na zpozorování jednotlivých barev. V případě zelené barvy nejprve zkontrolujte, zda se robot nachází v bezpečné vzdálenosti od překážky. Pokud nikoli, robot zatroubí, jinak robot provede zvuk nastartování motoru a rozjede směrem vpřed spolu se spuštěným zvukem motoru. Dále stále také kontrolujte, zda se nachází v dostatečné vzdálenosti od překážek. Pro žlutou barvu nejprve zkontrolujte, zda je robot v pohybu či nikoli za pomocí bloků, které vrací aktuální rychlost motoru. Pokud robot stojí, jsou spuštěny pouze zvuky motoru. Pokud je v pohybu, zpomalí a za 3 sekundy následně zastaví a vypne zvuky motoru. Při červené barvě robot okamžitě zastaví motor i zvuky motoru.

V jazyce MicroPython jsou nejprve inicializované proměnné pro barevný senzor, ultrazvukový senzor a proměnná *motor_started* do níž nastavte hodnotu *false*. Následně v nekonečné smyčce nejprve kontrolujte, zda se robot nenachází v blízkosti překážky pomocí metody *distance()* a zda je v pohybu pomocí metody *speed()* u jednoho z motorů. Pokud ano, tak jej zastavte a vypněte zvuky. Následně kontrolujte, zda barevný senzor nenarazil na jednu z barev semaforu a následně na základě dané barvy robota zastavte, zpomalte a zastavte nebo opět rozjeďte a nastavte příslušně proměnou *motor_started*. Při rozjezdu zkontrolujte, zda se nenachází robot před překážkou opět pomocí ultrazvukového senzoru. Při žluté barvě kontrolujte, jestli je robot v pohybu. Jestli ano, pak jej zpomalte a zastavte, jinak pouze spusťte zvuky motoru. Na konci kontrolujte, zda je motor spuštěn pomocí proměnné *motor_started* a pokud ano, přehrajte zvuk motoru.

Řešení v jazyce Scratch



Obr. 33 Řešení úlohy 11 v jazyce Scratch

Řešení v jazyce MicroPython

```
# inicializace barveného a ultrazvukového senzoru
color_sensor = ColorSensor(Port.S1)
ultrasonic_sensor = UltrasonicSensor(Port.S2)

motor_started = False

while True:
    if ultrasonic_sensor.distance() < 50 and left_motor.speed() != 0:
        robot.stop()
        ev3.speaker.play_file(SoundFile.MOTOR_STOP)
        motor_started = False

    if colorSensor.color() == Color.YELLOW:
        if left_motor.speed() == 0 and right_motor.speed == 0:
            ev3.speaker.play_file(SoundFile.MOTOR_START)
            motor_started = True
        else:
            robot.drive_time(50, 0, 3000)
            ev3.speaker.play_file(SoundFile.MOTOR_STOP)
            motor_started = False

    if color_sensor.color() == Color.GREEN:
        if ultrasonic_sensor.distance() < 50:
            ev3.speaker.play_file(SoundFile.HORN_1)
        else:
            motor_started = True
            robot.drive(200, 0)

    if color_sensor.color() == Color.RED:
        robot.stop()
        ev3.speaker.play_file(SoundFile.MOTOR_STOP)
        motor_started = False

    if motor_started:
        ev3.speaker.play_file(SoundFile.MOTOR_IDLE)
```

4.12 Souboj sumo

Postavte a naprogramujte robota pro zápas sumo proti druhému robotovi. Cílem souboje je vytlačit soupeřova robota mimo z vytyčeného bílého kruhu ohraničeného černou páskou. Souboj začíná na středu kruhu, kde jsou umístěni oba roboti zády k sobě. Pravidlem je, že každý

robot po spuštění programu musí vyčkat 3 sekundy do zahájení souboje. Prohrává ten robot jenž je celým svým objemem mimo kruh.

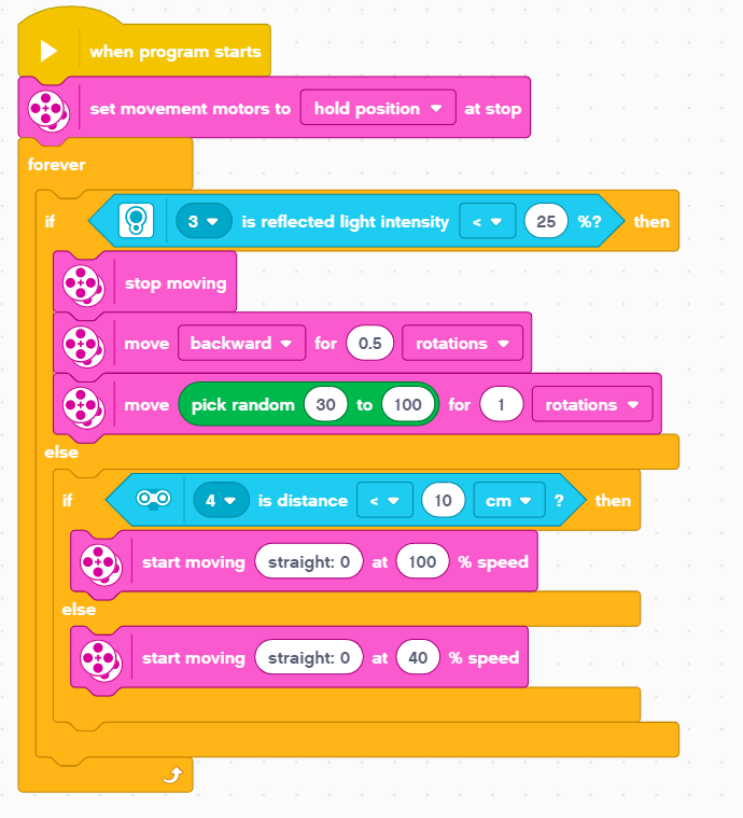
K sestavení programu využijte zkušenosti z předchozích úloh.

Řešení v jazyce Scratch

Správné řešení této úlohy neexistuje, jakékoliv z řešení může svůj souboj vyhrát, záleží na programu soupeře, modelu sestaveného robota a dalších faktorech. Každý robot by však v základu měl kontrolovat pomocí barevného senzoru, zda se nenachází na okraji kruhu a případně adekvátně reagovat. Následně je již řešení na fantazii každého. Lze např. využít ultrazvukový nebo infračervený senzor pro měření vzdáleností od soupeře, dotykový senzor pro reakci na kontakt se soupeřem nebo vlastní robotické paže, které se budou snažit vyvést soupeře z rovnováhy.

Ukázkové řešení využívá pro pohyb v kruhu kontrolu za pomocí odrazu světla podobně jako v jedné z předešlých úloh. Následně za pomocí ultrazvukového senzoru kontroluje, zda se před ním nenachází soupeř ve vzdáleností menší než 15 cm. Pokud ano, zaútočí zvýšením výkonu svých motorů. Pokud ne, pohybuje se dále běžnou rychlostí.

Řešení v jazyce Scratch



Obr. 34 Řešení úlohy 12 v jazyce Scratch

Řešení v jazyce MicroPython

```
# inicializace barveného a ultrazvukového senzoru
ultrasonic_sensor = UltrasonicSensor(Port.S1)
color_sensor = ColorSensor(Port.S2)

while True:
    if color_sensor.reflection() < 25:
        robot.stop(Stop.HOLD)
        robot.drive_time(-200, 0, 1000)
        if (random.randint(1,2) == 1):
            robot.turn(random.randint(60, 120))
        else:
            robot.turn(random.randint(-120, -60))
    else:
        if ultrasonic_sensor.distance() < 150:
         robot.drive(500, 0)
     else:
          robot.drive(200, 0)
```

ZÁVĚR

Cílem této bakalářské práce bylo seznámení se s možnostmi výuky programování a algoritmizace na základních školách za pomocí robotických stavebnic a následné vytvoření ukázkových úloh, které by mohly být při takovéto výuce využity.

V první části se práce věnovala seznámení se se pojmem robotická stavebnice a jejich využití při vyučování na základních školách. Dále byla nastíněn aktuální obsah výuky předmětu informatiky dle rámcového vzdělávacího programu a zásadní změny při této výuce pro školní rok 2021/2022, které daleko více podporují výuku programování na základních školách. Nakonec byly představeny možné výhody a nevýhody použití robotických stavebnic při výuce.

V druhé části byla práce zaměřena na robotickou stavebnici Lego Mindstorms EV3 a její hardwarové vybavení. Byly popsány jednotlivé části stavebnice jako EV3 kostka, její rozhraní, dostupné druhy motorů a senzorů s příklady jejich možného využití.

Třetí část se zaobírala možnostmi, jakými lze robota Lego Mindstorms EV3 programovat. Po obecném seznámení byl podrobněji představen program Lego Mindstorms Education EV3 Classroom spolu s grafickým programovacím jazykem Scratch, který tento program využívá. Následně byly popsány jednotlivé kategorie a tvary bloků, které tento blokový jazyk využívá a byl představen ukázkový program. Nakonec byly také popsány možnosti vytváření programů za pomocí textového jazyka MicroPython a jejich využívání pro EV3 roboty.

Poslední část práce se věnovala 12 konkrétními praktickými úlohami. Nejprve však byly představeny základní informace potřebné pro řešení úkolů. Následovaly jednotlivé úlohy, které se skládají ze zadání a řešení, a to jak v jazyce Scratch tak MicroPython. První úkoly jsou zaměřeny na seznámení se s robotem, jeho možnostmi jízdy, přehrávání zvuků, zobrazování textu či obrázků na displeji a na použití základních programových konstrukcí jako jsou podmínky či cykly. Další úlohy se soustředí na základní práci s různými senzory. Následují již komplikovanější úlohy při kterých je nutné využít a propojit jednotlivé dříve nabyté zkušenosti.

Řešení jednotlivých úloh není vždy jen jedno, proto je třeba brát ohled na kreativitu studentů. Hodnocení by mělo probíhat vždy na základě splnění či nesplnění zadání, až po prozkoumání výsledku při praktické vyzkoušení.

Pro další možné úlohy, návody na sestavení různých robotů či další doplňující informace lze navštívit webové stránky firmy Lego, kde v části Education [14] lze najít spoustu dalších informací a materiálů.

POUŽITÁ LITERATURA

[1] *Rámcový vzdělávací program pro základní vzdělávání* [online]. Praha: MŠMT, 2021 [cit. 2021-03-12]. Dostupné z: http://www.nuv.cz/file/4982.

[2] BAŤKO, J. (2018). EDUKAČNÍ ROBOTIKA VE VÝUCE NA ZÁKLADNÍCH ŠKOLÁCH V ČESKÉ REPUBLICE. *Journal of Technology and Information Education*, *10*(1), 5-16 [cit. 2021-03-12]. Dostupné z: doi: 10.5507/jtie.2018.001.

[3] SCARADOZZIA, David, Laura SORBIA, Anna PEDALEAB, Mariantonietta VALZANOC a Cinzia VERGINEC. Teaching robotics at the primary school: an innovative approach. *Procedia - Social and Behavioral Sciences* [online]. únor 2015, 174(12), 3838-3846 [cit. 2021-03-12]. ISSN 1877-0428. Dostupné z: doi:10.1016/j.sbspro.2015.01.1122.

[4] KERTB, Serhat Bahadır, Mehmet Fatih ERKOÇB a Sabiha YENI. The effect of robotics on six graders' academic achievement, computational thinking skills and conceptual knowledge levels. *Thinking Skills and Creativity* [online]. prosinec 2020, 38(4), 1-11 [cit. 2021-03-12]. ISSN 1871-1871. Dostupné z: doi:10.1016/j.tsc.2020.100714.

[5] TOCHÁČEK, Daniel a LAPEŠ, Jakub. *Integration educational robotics into the training of future ICT teachers*. In: ALIMISIS, D., MORO, M. Proceedings of 3rd International Workshop Teaching Robotics, Teaching with Robotics Integrating Robotics in School Curriculum. Riva del Garda: TRTWR, 2012, s. 51-56 [cit. 2021-03-12]. ISBN 978-88-95872-05-6.

[6] TOCHÁČEK, Daniel a LAPEŠ, Jakub. Usage of Lego NXT MINDSTROMS Kits in educational robotics. *Information and Communication Technology in Education 2011*. 2011, s 393-397 [cit. 2021-03-12]. ISBN 978-80-7368-979-7.

[7] PETR, Zdeněk. *Využití malých robotů při výuce programování na ZŠ*. Zlín, 2017. Diplomová práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky. Vedoucí práce Bronislav CHRAMCOV.

[8] TROJÁNEK, Pavel. *Využití robota Lego Mindstorms při výuce*. Praha, 2009. Bakalářská práce na Fakulta elektrotechnické ČVUT na katedře a řídící techniky. Vedoucí práce: Martin HLINOVSKÝ.

[9] PIKNER, Michal. *Využití stavebnice Lego při výuce.* Zlín, 2008. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky. Vedoucí práce: Milan ADÁMEK.

[10] LEGO MINDSTORMS – Informace. *LEGO* [online]. [cit. 2021-03-15]. Dostupné z: https://www.lego.com/cs-cz/themes/mindstorms/about.

[11] ÜÇGÜL, Memet. *History and educational potential of LEGO Mindstorms NXT*. Mersin University Journal of the Faculty of Education [online]. srpen 2013, 9(2), 127-137 [cit. 2021-03-15]. ISSN 1306-7850. Dostupné z: doi:10.17860/efd.06656.

[12] Lego Mindstorms EV3 Brick. In: *LEGO* [online]. [cit. 2021-04-17]. Dostupné z: https://www.lego.com/cdn/cs/set/assets/blte0ca03b812e48adc/45500_alt1.jpg.

[13] THE LEGO GROUP. *Mindstorms EV3 – User guide* [online]. 2013. 69 s [cit. 2021-03-15]. Dostupné z: https://www.lego.com/cdn/cs/set/assets/bltbef4d6ce0f40363c/LMSUser_Guide_LEGO_MINDSTORMS_EV3_11_Tablet_ENUS.pdf.

[14] THE LEGO GROUP. *Mindstorms EV3 education – User guide* [online]. 2013. 80 s [cit. 2021-03-15]. Dostupné z: https://education.lego.com/en-gb/product-resources/mindstorms-ev3/downloads/user-guide.

[15] Ke stažení | Mindstorms | Oficiálního LEGO® obchodu CZ. *Lego* [online]. [cit. 2021-03-15]. Dostupné z: https://www.lego.com/cs-cz/themes/mindstorms/downloads.

[16] BAŤKO, Jan, Tomáš JAKEŠ a Petr SIMBARTL. *Popis stavebnice LEGO MINDSTORMS EV3* [online]. 2015 [cit. 2021-03-15]. Dostupné z: https://dspace5.zcu.cz/bitstream/11025/29368/1/Popis%20stavebnice.pdf.

[17] BURNETT, Wayne. a Programming Languages for LEGO MINDSTORMS. *LEGO Engineering* [online]. 29. 11. 2018 [cit. 2021-04-02]. Dostupné z: http://www.legoengineering.com/alternative-programming-languages.

[18] DELHEZ, Marijn. *FIRST* [online]. 14. 7. 2020 [cit. 2021-04-02]. Dostupné z: https://hjernekraft.org/media/dokumenter/fll/2020/software/lego-mindstorms-education-ev3-classroom-app-1.pdf.

[19] SYROCKA, Ola a Dominika SKRZYPEK. EV3 Classroom or EV3 Lab? MINDSTORMS Programming Apps Comparison. *RoboCamp.eu* [online]. 24. 2. 2021 [cit. 2021-04-02]. Dostupné z: https://www.robocamp.eu/en/blog/lego-mindstorms-ev3-classroom-app.

[20] ROLLINS, Mark. *Beginning LEGO MINDSTORMS EV3*. Apress, 2014. ISBN 978-1430264361.

[21] THE LEGO GROUP. *Ev3-rem-driving-base* [online]. 2015 [cit. 2021-4-17]. Dostupné z: https://education.lego.com/v3/assets/blt293eea581807678a/bltdb0d9e7188f73df5/5ec7bfb29b2ffb61d5c8091a/ev3-rem-driving-base.pdf

[22] THE LEGO GROUP. *Getting started with LEGO® MINDSTORMS® Education EV3 MicroPython: Version 2.0.0* [online]. [cit. 2021-04-17]. Dostupné z: https://education.lego.com/v3/assets/blt293eea581807678a/bltb470b9ea6e38f8d4/5f8802fc4376310c19e33714/getting-started-with-micropython-v2_enus.pdf.

[23] THE LEGO GROUP. Robot Trainer | MINDSTORMS EV3 Unit Plan | LEGO® Education. *LEGO® Education* [online]. [cit. 2021-04-17]. Dostupné z: https://education.lego.com/en-us/lessons/ev3-robot-trainer

[24] MicroPython. *MicroPython - Python for microcontrollers* [online]. [cit. 2021-04-17]. Dostupné z: https://micropython.org/.

[25] MACEK, Tomáš. *Tvorba úloh pro předmět Programování I robotické úlohy*. Pardubice, 2015. Bakalářská práce. Univerzita Pardubice, Fakulta ekonomicko-správní. Vedoucí práce Jan Panuš.