Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Programovanie robotov pomocou stavových automatov

Bakalárska práca

2024
Tomáš Vikiszály

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Programovanie robotov pomocou stavových automatov

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Mgr. Pavel Petrovič, PhD.

Bratislava, 2024
Tomáš Vikiszály

96368802

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Tomáš Vikiszály
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Programovanie robotov pomocou stavových automatov
Programming Robots Using Finite State Machines

Anotácia: Robotické modely zo stavebnice Spike Prime sa programujú v troch rôznych jazykoch: IconBlocks – určený najmä pre prvý stupeň ZŠ, WordBlocks – určený najmä pre druhý stupneň ZŠ, Python – určený najmä pre pokročilých používateľov. Problém je, že ani jeden nie je ideálny na modelovanie správania robota: IconBlocks: neobsahuje ani podmienky, WordBlocks: obsahuje podmienky, cykly, a (bohužiaľ iba globálne) udalosti, Python: texový jazyk a preto neprehľadný a nevhodný na modelovanie správania. Najprirodzenejším modelom správania je stavový automat (state machine), pretože robot počas riešenia úlohy prechádza cez rozličné stavy/fázy – napr. stav1: je na čiare, stav2: hľadá pokračovanie čiary po jej prerušení, stav3: prechádza cez tunel, stav4: obchádza prekážku, ale aj jednotlivé stavy môžu o úroveň abstrakcie nižšie obsahovať stavové automaty, napr. kým je na čiare, tak sa mení stav medzi dvoma stavmi: hranavpravo: nachádza sa naľavo od hrany, hranavľavo: nachádza sa napravo od hrany. Čiže každý stav môže byť vnútri znova stavový automat a naopak, celkové správanie automatu možno považovať za jeden makrostav v stavovom automate na vyššej úrovni abstrakcie. Na každom stavovom prechode je určená udalosť, kedy k nemu dochádza, pričom virtuálne udalosti môže automat generovať aj sám. V jednotlivých stavoch a na stavových prechodoch možno štartovať aj tradične zapísaný procedurálny kód (task), pričom tasky môžu interagovať, bežať aj v pozadí stavového automatu alebo byť automaticky ukončené pri zmene stavu.

Cieľ: Úlohou študenta bude naprogramovať a na netriviálnych ukážkach otestovať nový grafický programovací jazyk pre Spike Prime založený na stavových automatoch.

Literatúra:
R. A. Brooks, "A robot that walks; emergent behaviors from a carefully evolved network," in Proceedings, 1989 International Conference on Robotics and Automation, pp. 692-4+2 vol.2, 1989.
R. Balogh, D. Obdržálek, “Using Finite State Machines in Introductory Robotics”, in: Robotics in Education. RiE 2018. Advances in Intelligent Systems and Computing, vol 829. Springer, 2019.
R. Ghzouli et al. "Behavior Trees and State Machines in Robotics Applications," in IEEE Transactions on Software Engineering 49 (9), Sept. 2023.

Kľúčové slová: konečný automat, programovanie robotov, robotika vo vzdelávaní

Poďakovanie: Cítim obrovskú vďačnosť voči môjmu školiteľovi, ktorým bol Mgr. Pavel Petrovič, PhD, za všetku pomoc, podporu a hodiny so mnou strávene v učebni. Vďačny som aj svojej rodine za podporu a povzbudenia, ktoré som počas svojho celého štúdia od nich dostával.

Abstrakt

Kľúčové slová: Konečný automat, programovanie robotov, robotika vo vzdelávaní

Abstract

Keywords: Finite state machines, robot programming, robotics in education

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

...

Kapitola 1

Východiská

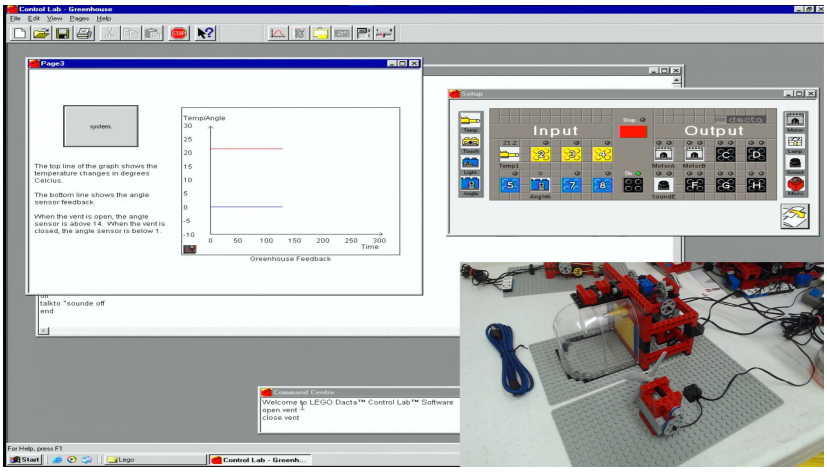
V tejto kapitole sa budem venovať teoretickej časti mojej bakalárskej práce. Budem v nej popisovať stavebnice LEGO Education, ich históriu a postupný vývoj, možnosti programovania týchto stavebníc, výhody a problémy takéhoto programovania a programátorské prostredia. Popíšem ďalšie možnosti, ako programovať tieto stavebnice, kde spomeniem konečné stavové automaty, aktuálny firmvér, tvorbu grafickej aplikácie v jazyku C#.

1.1 Stavebnice LEGO Education

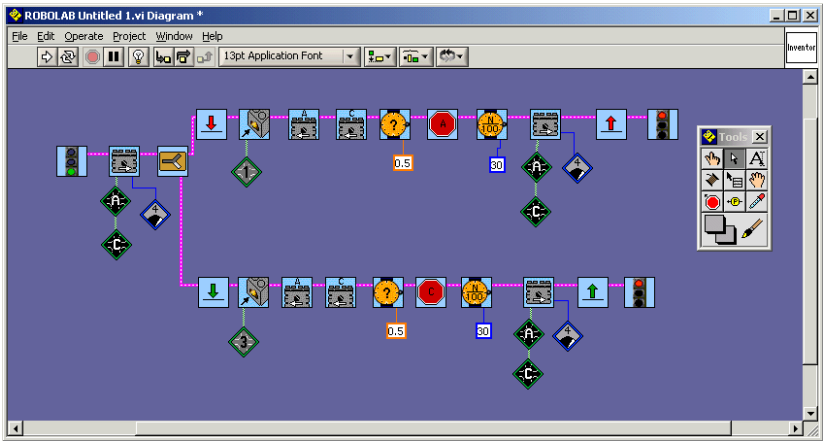
Stavebnice od spoločnosti LEGO Education vznikli v osemdesiatých rokoch minulého storočia a sú určené pre žiakov základných a stredných škôl i navzdory tomu, že prvé stavebnice boli vyrábané ako bojové roboty so základnými súčiastkami pre chlapcov. Dôvodom toho boli ceny hárdveru, ktorými stavebnice disponovali. S tým, ako išla veda a výskum dopredu, súčiastky boli lacnejšie, z čoho plynulo to, že stavebnice už neboli kupované iba základnými a strednými školami ale i domácnosťami, a to s plným vybavením. Okrem jednoduchých servomotorov začali stavebnice obsahovať rôznymi senzormi na farby a vzdialenosť, gyroskopom, displejom a ďalšími komponentami. Školy ich využívali na vzdelávanie informatiky, robotiky a programovania pomocou hry, kreativity a objavovania, k čomu dostávali od LEGO Education štúdijne materiály.

1.1.1 Prvá generácia

Prvou stavebnicou bola stavebnica LEGO MINDSTORMS. Táto generácia sa programovala na počítačoch pomocou softvéru LEGO Dacta. V tomto softvéri sa vytvoril kompletný program aj s jeho vizualizáciou, napríklad grafovou, a bol spúšťaný priamo v počítači. Stavebnica neobsahovala žiadnu pamäť ani batériu, len hárdver pozostávajúci zo servomotorov a senzorov. Nevýhodou bolo to, že robot musel byť neustále pripojený k počítaču, čo obmedzovalo majteľa počas programovania k nejakému zložitejšiemu



Obr. 1.1: Na obrázku je možné vidieť softvér LEGO DACTA a stavebnicu LEGO MINDSTORMS.Zdroj obrázka je snímok obrazovky z videa [2]



Obr. 1.2: Ilustračný obrázok programu v LEGO RoboLab. Zdroj je [3]

pohybu čo ale zároveň bolo výhodou to, že všetky dáta si mohol uživateľ nechať vykresľovať alebo vypisovať na obrazovku. Nejaké ukážky môžeme nájsť na tejto stránke [1].

Alternatívou k softvéru LEGO Dacta bol vytvorený softvér založený na jazyku LabView s názvom LEGO RoboLab. Narozdiel od textového kódu v LEGO Dacta tento softvér bol grafický postavený na takzvanom toku dát. Môžeme si to predstaviť ako súbor nezávyslích firiem, ktoré spracujú material okamžite po tom, čo k ním príde a hneď ho posielajú ďalej, do ďalšej firmy. Tým je zabezpečená rýchlosť behu programu a preto je tento jazyk veľmi populárny medzi fyzikmi programujucími senzori.

1.1.2 Druhá generácia

Keďže LEGO Education malo so stavebnicou úspech, tak pokračovali vo výskume a na trh priniesli novú generáciu stavebnice čo popisuje aj jej názov, LEGO MINDSTORMS

Obr. 1.3: Robot LEGO MINDSTORMS NXT-G. Zdroj je [4]

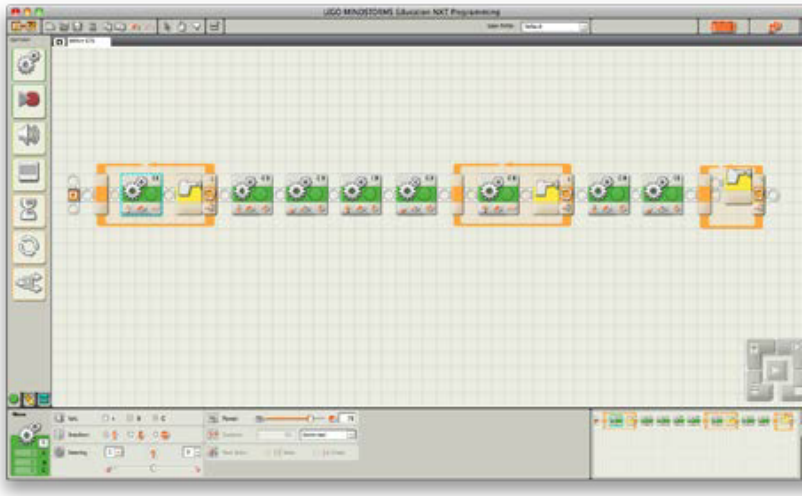
Next Gereration, skrátene LEGO NXT-G. Rozdielov bolo niekoľko. Niektoré boli malé a iné opäť veľké. Medzi tie veľké vieme zaradiť to, že riadiaca jednotka, ktoré sa doteraz muselo pre programovanie a spúšťanie programov pripájať k počítaču, malo v sebe základný hárdver pre spustenie operáčneho systému linux. To bolo veľkou výhodou, keďže sme už nepotrebovali mať stavebnicu neustále pripojenú k počítačú. Prinieslo to aj negatívny dopad, a to ten, že stavebnica už nekomunikovala s počítačom a tak sme stratili vedomosť a všetkých hodnotách, s ktorými robot narábal.

Robota bolo už možné programovať dátovým káblom alebo na ďiaľku pomocou Bluetooth. Na programovanie slúžil nový grafický softvér, ktorý ale stále používal jazyk LabView alebo aj samotný LabView pre pokročilích. Tento softvér mal viac prehľadnejšie ikony jednotlivých príkazov a po rozkliknutí nejakého bloku sa otvorila paleta, kde sme si mohli navoliť parametre, s ktorými bude robot pracovať. Keď ale chcel uživateľ zdieľať svoj program, tak to bolo veľmi nepraktické, lebo musel odfotiť každý blok samostatne, lebo bez parametrov by druhý uživatel nemal žiadaný výsledok.

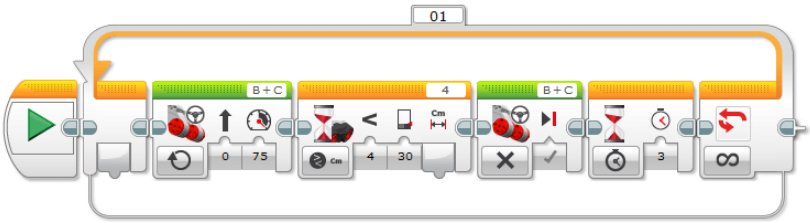
Počas prudukcie tejto generácie sa začal usporudavúvať program First LEGO League, ktorý je pre deti vo veku od 4 až 16 rokov formou súťaže, ktorá má u účastnikov podporovať kritické a tvorivé myslenie pri riešení reálnych problémov z nášho sveta.

1.1.3 Tretia generácia

Tretia generácia nesie meno LEGO MINDSTORMS Evolution, alebo skrátene LEGO EV3. Táto verzia robota dovoľovala takzvané logovanie informácii, programovanie cez dátový kábel a Bluetooth, či za použitia iPad alebo Android zariadenia. Bola programovaná vo vynovenom grafickom systéme, ktorý používala druhá generácia ale všetky informácie a nastavenia jednotlivých blokov programu boli viditeľné bez ďalšieho kli-



Obr. 1.4: Program pre LEGO MINDSTORMS NXT-G. Zdroj je [3]



Obr. 1.5: Program pre LEGO MINDSTORMS EV3. Zdroj je [3]

kania na ne. Disponovala aj modulom pre obsluhu wifi, ktorý neskôr pre program First LEGO League vypli na úrovni softvéru.

Počas existencie tejto generácie vznikol a bol spopularizovaný jazyk z MIT, ktorý nesie meno Scratch. Je populárny pre svoju nenáročnosť a jednoduchosť, čo prijmelo LEGO Education zakomponovať možnosť programovať roboty pomocou jazyka Scratch. Na popularite medzi žiakmi narástol aj jazyk Python, ktorý bol tiež zakomponovaný medzi možnosti, ako programovať LEGO Education robotov.

1.1.4 Štvrtá generácia

Táto generácia nesie meno LEGO Spike Prime. Líši sa tým, že nemá displej, porty sú univerzálne, čiže už nie sú vyhradené iba pre motor alebo senzor, má 5x5 LED maticu a bezchybný gyroskop v troch osiach. Na programovanie slúžia už len jazyky Scratch a Python. Robot je programovateľný cez dátový kábel alebo použitím Bluetooth a nejakú dobu disponoval aj modulom pre wifi ako EV3, ktorý nakoniec tiež vypli.

1.2 Jazyky na programovanie robotov

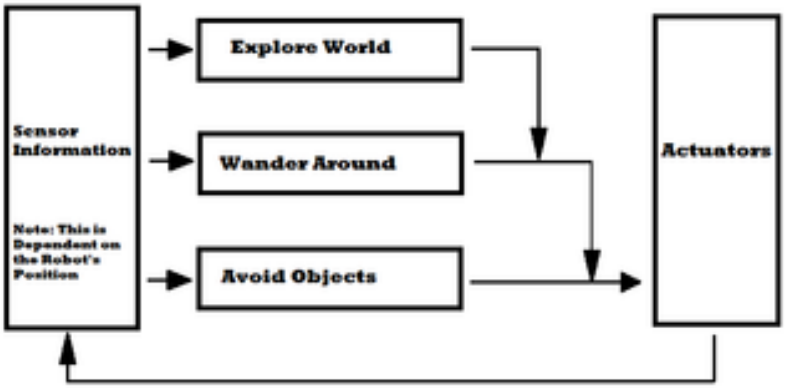
LEGO Education nám ponúka momentálne dva programovacie jazyky, ktorými môžeme proramovať ich stavebnice a nimi sú Python a Scratch. Tieto jazyky sú vhodné pre začiatočníkov ako úvod do programovania ale sú veľmi nevhodné pre programovanie robotov, kde nám záleží na každej milisekunde. Každá zbytočná operácia, ktorú musí riadiaca jednotka urobiť navyše nás len vzdiaľuje od toho, čo sme v programe napísali. Týmto trpí jazyk Python. Keďže je veľmi dynamicky, alokuje a dealokuje si pamäť kedykoľvek to on uzná za vhodné a vtedy je v rýchlosti veľmi nekonzistentný. Prvýkrát mu nejaká operácia trvá určitý počet milisekúnd a keď tú istú operáciu zavoláme o chvíľu neskôr, tak to môže trvať aj niekoľko násobbne dlhšie, lebo práve v tej chvíli spustil vnútorný garbage collector. Ďalšou nevýhodou je, že programátor musí ovládať celú radu príkazov z knižnice, ktorá mu sprístupňuje ovládanie hárdveru. Jazyk Scratch môže byť veľmi náročný pre programovanie zložitejších robotických úloh a taktiež ho obmedzuje rýchlosť ako Python. Pri programovani robotov sa snažíme naprogramovať rozhodovanie pre rôzne situácie v ktorých sa bude robot nachádzať. To pri lineárnom programovaní nemusí byť hneď jasné a celkovo to môže čitateľa a programátora pri oprave miasť.

1.2.1 Konečné stavové automaty

Koncept stavových automatov existuje od roku 1936 počnúc profesorom Alanom Turingom. Takýto stavový automat krásne popisuje naprogramované správanie robotov. S touto myšlienkou prišiel Rodney Brooks, ktorý sa považuje za otca "behaviour-based"robotiky a tento systém sa používa už vyše dvadsať rokov. S tým je prepojený systém programovania s názvom "Subsumption architecture", kedy staviame správanie robota do vrstiev a až keď máme správne uchopené nižšie správanie prograpujeme správanie na vyššej vrstve, ktoré využíva prvky nižších. Pre príklad uvediem prípad, kedy chceme robota naučiť behať cez prekážky. V prvom kroku ho naučíme stáť, v ďalšom kráčať a skákať, kde využikeme správanie, ktoré sme naprogramovali v státi. Ďalej ho naučíme behať kde opäť využijeme kráčanie a napokon môžeme programovať beh cez prekážky, kde budeme využívať beh a skákanie.

Dôvody, prečo používať konečné stavové automaty pre programovanie robotov dobre zhrnul autor v publikácii [7], kde hovorí, že stavové automaty sú ľahko pochopiteľné, často používane medzi programátormi ako komunikácia a často už same o sebe tvoria dokumentáciu. Je jednoduché sledovať v ktorom stave sa daný robot nachádza a ľahko sa tak hľadajú chyby v programe.

[8] A finite automaton (FA) consists of a finite set of states and a set of transitions from state to state that occur on input symbols chosen from an alphabet X. For each



Obr. 1.6: SubSumption Architecture z knihy [6]

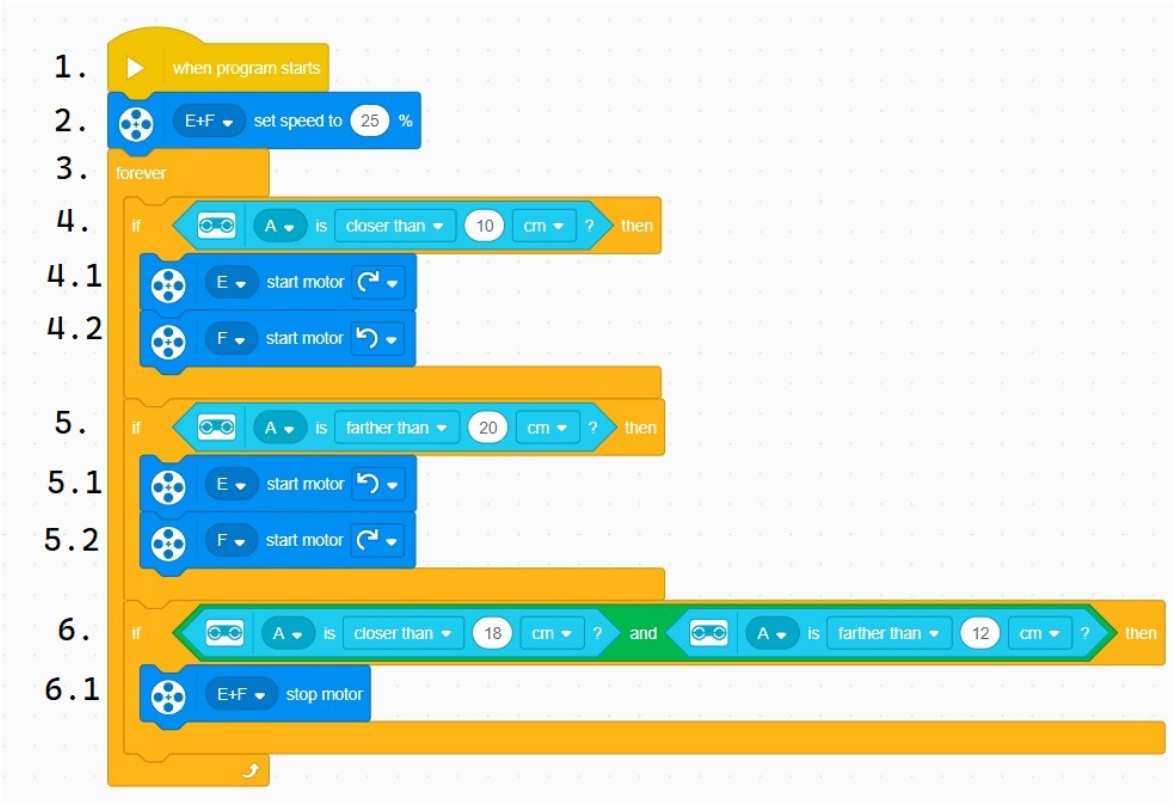
input symbol there is exactly one transition out of each state (possibly back to the state itself). One state, usually denoted g0 , is the initial state, in which the automaton starts. Some states are designated as final or accepting states. A directed graph, called a transition diagram, is associated with an FA as follows. The vertices of the graph correspond to the states of the FA. If there is a transition from state q to state p on input a, then there is an arc labeled a from state q to state p in the transition diagram. The FA accepts a string x if the sequence of transitions corresponding to the symbols of x leads from the start state to an accepting

Ak ale chceme, aby nám robot dával nejakú spätnú väzbu, potrebuje miesto, kam to zapísať. Tomuto popisu vyhovuje konečný stavový transducer. Tento automat má možnosť zapisovať do pamäťi a túto pamäť môžeme dať opäť najekému inému automatu, ktorý ju bude ďalej spracuvávať.

Definition 1. A weighted finite-state transducer T over (R, +, ·, 0, 1) is an 8-tuple T = (, , Q, I, F, E, ,) where is the finite input alphabet of the transducer, is the finite output alphabet, Q is a finite set of states, I Q the set of initial states, F Q the set of final states, E Q×(ϱ)×(ϱ)×R×Q a finite set of transitions, : I → R the initial weight function, and : F → R the final weight function mapping F to R.

1.3 LEGO Spike Prime

Tak ako som už písal v 1.1.4, tohto robota je možné programovať pomocou jazykov Python a Scratch. Hárdver riadiacej kocky pozostáva z procesora STM32F413, 32MB internej pamäte, šiestimi portami pre vstupno-výstupné zariadenia z čoho sú dva vysokorýchlostné a štyri obyčajné, micro USB káblom, ktorý okrem programovania slúži aj na inštaláciu firmvérov, Bluetooth a Bluetooth Low Energy, gyroskopom pre tri osi, acelerometra a zariadením pre tvorbu zvuku. Všetky tieto informácie spísal majiteľ

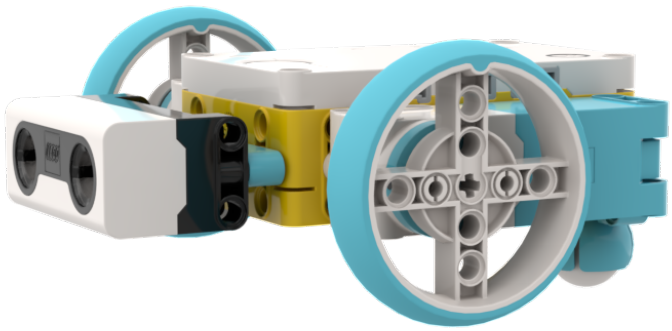


Obr. 1.7: Na obrázku je program, ktorým robot prenasleduje objekt, ktorý je pred ním.

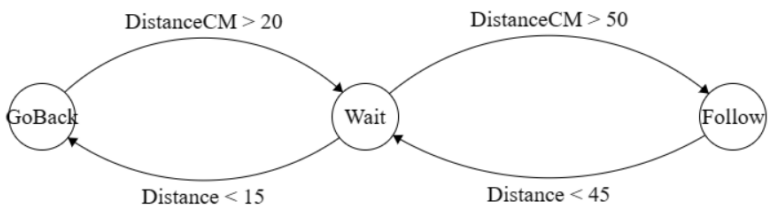
github repozitára [9]. Okrem vstavaného hárdveru stavebnica obsahuje servomotory, ultrazvuk, dotykový senzor a farebný senzor.

Na ilustráciu práce s touto stavebnicou použijem úlohu, kde robot má prenasledovať objekt, ktorý je pred ním, ale ak sa objekt priblíži príliš blízko tak robot začne cúvať. Pre túto úlohu si zostrojíme robota s ultrazvukom, aký je vyobrazený na obrázku 1.8. Je to zjednodušená verzia príkladu z príručky [10] úloha 16. My ale použijeme iba jeden senzor. Obrázok 1.7 je fotografia funkčného programu a tu je jeho vysvetlenie:

- 1. začiatok programu, ktorý sa spustí hneď po prijatí programu do riadiacej jednotky alebo po opätovnom stlačení ovladacieho tlačidla
- 2. nastavenie rýchlosti robota na 25% výkonu
- 3. nekonečný cyklus programu
- 4. podmienka, v ktorej sa pýtame, či objekt pred nami je bližšie ako 10 centimetrov. Ak je táto podmienka splnená, tak:
  - 4.1 a 4.2 zapnutie motoru E v smere hodinových ručičiek a motoru F v opačnom smere
- 5. podmienka, v ktorej sa pýtame, či objekt pred nami je od nás ďalej ako 20 centimetrov. Ak je táto podmienka splnená, tak:



Obr. 1.8: Na obrázku je robot s ultrazvukom Obrázok bol vytvorený pomocou stud.io aplikácie.



Obr. 1.9: Robot pohybujúci sa závisle od vzdialenosti od objektu podľa stavového automatu vytvoreného na [11].

  - 5.1 a 5.2 zapnutie motorov E a F v opačnom smere ako v bodoch 4.1 a 4.2

- 6. podmienka, v ktorej sa pýtame, či objekt pred nami je ďalej ako 12 centimetrov a zároveň bližšie ako 18 centimetrov. Ak je táto podmienka splnená, tak:

  - 6.1 vypneme oba motory

Takýto program by vyzeral v podaní stavových automatov ako na ukážke 1.9. Automat pozostáva z troch stavov, "Wait", "GoBackä "Follow", medzi ktorými prechádza závisle od toho, v akej vzdialenosti je od objektu. Pod prechodmi si môžme ešte predstaviť príkazy pre ovládanie pohybu motorov, ktoré tento online dizajnér neponúka.

1.3.1 Programátorské prostredia

Pre programovanie týchto stavebníc existuje softvér priamo od LEGO Education, ktorý je v online podobe aj ako aplikácia. V nich si vie programátor zvoliť medzi jazykom

Python, Ïcon blocks", ktorý je jednoduchý Scratch jazyk podobájuci sa softvéru pre programovanie EV3 stavebnice, ktorý bol postavený na jazyku LabView a "World blocks", ktorý je Scratch, ktorého ukážku môžeme vidieť na obrázku 1.7. Okrem tohto softvéru vieme používať aj softvér s názvom Pybricks [12], ktorý má online aj offline verziu. Ponúka programovanie v jazyku Python a spoplatnenú verziu jazyku Scratch, inštaláciu firmweru a ukážky časti kódov, ktoré pracujú s knižnicou na správu hárdveru v jazyku Python.

Pybricks a firmvér

Celý softvér Pybricks je open source na githube [13]. V repozitári "pybricks-code"sú zdrojové súbory aplikácie, ktorá je písana v jazyku TypeScript nad frameworkom React. "pybricksdev"je základné nastavenie pre pokročilejších programátorov, ktorý chcú programovať v prostredí Visual Studio Code. Samotný firmvér je v repozitári "pybricks-micropython". Tento repozitár má v stromovej štruktúre niekoľko pre nás zaujímavých častí. Prvá časť je priečinok "bricks", ktorý obsajuje pravidla, podľa ktorých sa vytvorí firmvér a v ktorých potom aj tento firmvér nájdeme (/bricks/primehub/build/firmware.zip). Druhou dôležitou časťou tohto repozitára je priečinok /lib/pbio/sys/, v ktorom sú príkazy na ovládanie hárdveru. V main.c sa nachádza hlavná funkcia, v ktorej robot čaká na vypnutie alebo validný kód, ktorý následne vykoná. Ďalšou potrebnou časťou je súbor /bricks/_common/source.mk, v ktorom sú vypísané všetky súbory, ktoré sa pri vytváraní firmvéru použijú. Pybricks aplikácia je rozdelená na dve časti. V ľavej časti máme dve ikony, list a ozubené koleso, á navigáciu. Ozubené koleso symbolizuje nastavenia. Tu si vieme meniť farbu pozadia editora, inštalovať firmver, nájdeme tu veľa užitočných linkov, napríklad na úvod do tohto systému, vzorových projektov alebo miesot, kde vieme hlásiť akékoľvek problémy s týmto softvérom. Druhá možnosť, ktorú predstavuje prázdny list, slúži na správu našich programov. Jednotlivé programy vieme nahrávať, sťahovať a vytvárať nové. Pri vytváraní sa nás aplikácia opýta na jazyk, v ktorom budeme nášho robota programovať a názov programu. V pravej časti hore nájdeme štyri tlačidla. Prvým sa pripájame na našu riadiacu jednotku pomocou Bluetooth, druhým spúšťame program, tretím ho vieme vypnúť a štvrtým spustíme konzolu, v ktorej vieme posielať robotovi príkazy postupne. Pri tvorbe firmvéru je potrebný cross-compiller a kód písaní v jazyku C. Pre vytvorenie firmvéru pre stavebnicu LEGO Spike Prime potrebujeme byť v najvýššom priečinku repozitára a v operačnom systému Linux zavolať príkaz "make primehub". Na úspešnú inštaláciu potrebujeme mať aktualizované ovládače, na čo nás upozorní aj aplikácia pre neúspešnej inštalácii. Po úspešnom vytvorení firmvéru v aplikácii urobíme následujúci zoznam úkonov v danom poradí:

1. v ľavej časti hlavného menu otvoríme ozubené koleso, čo predstavuje nastavenia

2. v navigácii v časti "Firmware"zvolíme možnosť Ïnstall Pybricks Firmware"
3. vyskakovacie okno v dolnej časti v strede má možnosť Ädvanced", klikneme na ňu
4. do zobrazeného rámčeka presunieme náš vytvorený firmver alebo ak klikneme doň, tak nám to otvorí prieskumnika, v ktorom si náš firmver vyhľadáme
5. po vložení firmveru by ho aplikácia mala rozoznať, čo sa prejaví tým, že nám v hornej časti už neukazuje možnosti jednotlivých stavebníc
6. potvrdíme tlačidlom "Next"
7. odsúhlasíme licenciu v zaškrtávacom boxe a klikneme na "Next"
8. pomenujeme si nášu riadiacu jednotku, ktorej názov sa nám bude zobrazovať pri pripájaní na ňu a potvrdíme tlačidlom "Next"
9. postupujeme podľa návodu:
   (a) odpojíme riadiacu jednotku a vypneme ju
   (b) za stáleho držania tlačidla Bluetooth pripojíme kocku dátovým káblom k počítaču
   (c) tlačidlo Bluetooth pustíme potom, čo začne blikať na rúžovo-zeleno-modrú farbu
10. klikneme na tlačidlo Ïnstall"
11. aplikácia nám ponúkne nové vyskakovacie okno v ktorom zvolíme našu kocku a potvrdíme tlačidlom "Connect"
12. úspešná inštalácia je signalizovaná rozsvietením led metrixu na riadiacej jednotke alebo zmiznutím ukazovateľa priebehu inštalácie v aplikácii

1.4 Programovanie grafickej aplikácie v C#

Programovanie aplikácii v jazyku C# je jednoduché a efektívne vďaka používaniu softvérov na to určených. Jazyk je objektovo orientovaný a o správu pamäte sa nemusíme starať, keďže ju spravuje mechanizmus garbage collector. Aplikácie sú plné rôznych predprogramovaných komponentov, užitočných a efektívnych knižníc a okolo týchto aplikácii je široká komunita a podrobná dokumentácia. Aplikácie využívaju framework .NET. Pri vytvárani novej aplikácie vytvoríme nový projekt, zvolíme možnosť "Windows Forms App"s ".NET Framework"v záložke "Deskop". Po potvrdení tlačidlom

Algoritmus 1.1: Automatický generovaný kód po vytvorení novej aplikácie C#

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

"Create"nám softvér vytvorí prázdnu aplikáciu, ktorá po spustení otvorí našu novovytvorenú aplikáciu. Spustiť ju vieme z editora a poslednú spustenú z editora vieme spustiť v priečinku, kde je náš projekt /nazovProjektu/nazovProjektu/bin/Debug/nazovProjektu.exe. Automaticky vygenerovaný kód novej aplikácie môže vyzerať ako v ukážke 1.1. Príkaz Üsing"znamená to isté, ako v iných jazykoch Ïnclude". "namespace"je totožný s príkazom, ktorý sa používa v jazyku C++, čiže namiesto volania "WindowsFormsApp1.Form1()"nám postačí "Form1()". Funkcia "public Form1()..."je konštruktor našej triedy, ktorý inicializuje všetky grafické komponenty. Neodporúča sa do tejto funkcie čokoľvek písať, lebo ak to interaguje s nejakým grafickým komponentom, ktorý ešte nebol inicializovaný, tak to môže spôsobiť spadnutie našej aplikácie. Ak máme nejaký kód, ktorý chceme spustiť hneď po otvorení aplikácie, je potrebné na to použiť funkciu "public Form1_Load()...", ktorá slúži presne pre tieto účely. Vytvoriť ju je potrebné cez záložku "Design"v spodnej časti softvéru.

V softvéri vieme všetko grafické ručne programovať, alebo si v záložke zvolíme namiesto "Code"záložku "Design", čo nám otvorí celú grafickú paletu nástrojov. Komponenty vieme pridávať do našej aplikácie, odoberať, škálovať, meniť rôzne atribúty a po dvojkliku na ne, ak sú interaktívne, sa nám vygeneruje kód 1.2, v ktorom vieme programovať správanie. Ak chceme takýto kód neskôr odstrániť, je nutné to robiť opäť cez "Design", alebo túto funkciu nájsť v súbore "Form1.Designer.csä vymazať ju i odtiaľto.

Algoritmus 1.2: Automatický generovaný kód pre grafické komponenty v C#

```
private void button1_Click(object sender, EventArgs e)
{
    throw new System.NotImplementedException();
}
```

1.4.1 Serializácia v aplikáciach C#

Serializácia je ukladanie objektov v pamäti do formátu, ktorý sa dá prenášať medzi počítačmi a neskôr pri čítani opäť vytvoriť pôvodne objekty. C# nemá vstavané fukncie serializácie ale potrebné knižnice sa dajú veľmi ľahko a rýchlo nainštalovať. Vhodná knižnica je "Newtonsoft.Json", ktorú nainštalujeme v časti "Tools» "NuGet Package Manager» "Manage NuGet Packages for Solution", kde túto knižnicu vyhľadáme a nainštalujeme. Potom stači pridať "string jsonString = JsonConvert.SerializeObject(Object, settings);", kde jsonString je textový reťazec vo formáte JSON. Ten následne môžeme zapísať do súboru. Knižnica serializuje všetko, čo je v triede Öbject"prístupne metódami " get; set; ä pri tom používa nastavenia, ktore sú v argumente "settings". Vieme tam nastaviť počet tabulátorov, ignorovanie rekurzívnych závislosti a mnoho ďalších parametrov. Pri ignorovaní závislosti si treba dať pozor, lebo kničnica si neurobí nejakú tabuľku podľa ktorej tieto závislosti pri deserializacii naspäť nastaví, ale všetky tieto parametre serialiuje ako null.

Kapitola 2

Špecifikácia a ciele práce

Kapitola 3

Návrh

Kapitola 4

Implementácia

Kapitola 5

Výsledky

Záver

...

Literatúra

[1] Webová lokalita s LEGO MINDSTORMS prvej generácie, ktorá obsahuje vzoroví program a robota <https://www.eurobricks.com/forum/index.php?/forums/topic/190311-wip-in-depth-lego-dacta-retrospective-video-series/> (naposledy navštívené 17.05.2024)

[2] Video so stavebnicou a programom prvej generácie LEGO Education <https://youtu.be/0pHY7IJ5T-o?si=Y-19RfcTDB0d5R6Y> (naposledy navštívené 17.05.2024)

[3] DE MEDEIROS, LUCIANO F AND CUCH, LUIZ, 2017, Robótica Educacional como Recurso Pedagógico para Alunos de Baixo Rendimento: Relato de Experiência <https://www.researchgate.net/publication/> (naposledy navštívené 17.05.2024)

[4] Oficiálna stránka LEGO Education <https://education.lego.com/> (naposledy navštívené 18.05.2024)

[5] Stránka venujúca sa robotom, ktorej správcnom je Pavel Petrovič <https://wiki.robotika.sk/robowiki/index.php?title=RCX_and_NXT> (naposledy navštívené 18.05.2024)

[6] BROOKS R., 1999, Cambrian Intelligence: The Early History of the New AI, Massachusetts : Cambridge, 1999, ISBN 978-0-262-02468-6

[7] Balogh, Richard & Obdržálek, David. (2019). Using Finite State Machines in Introductory Robotics: Methods and Applications for Teaching and Learning. 10.1007/978-3-319-97085-1_9.

[8] HOPCROFT JOHN E., ULLMAN JEFFREY D., Introduction to automata theory, languages and computation, 1939, ISBN: 0-201-02988-X

[9] Github repozitár zhŕňujúci všetký základné informácie o LEGO Spike Prime stavebnici <https://github.com/gpdaniels/spike-prime> (naposledy navštívené 19.05.2024)

[10] PETROVIČ P., Stavebnice lego mindstorms education EV3, materiály ku školeniam, robotika.sk/events/18Skolenia/priruckaEV3.pdf

[11] Stránka na tvorbu konečných stavových automatov <https://www.madebyevan.com/fsm/> (naposledy navštívené 19.05.2024)

[12] Alternatívny softvér Pybricks pre programovanie Spike Prime stavebníc <https://pybricks.com/> (naposledy navštívené 19.05.2024)

[13] Zdrojové súbory softvéru Pybricks <https://github.com/pybricks> (naposledy navštívené 19.05.2024)

Príloha A: obsah elektronickej prílohy

...

Príloha B: Používateľská príručka

...