UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Uporaba robota pri učenju programiranja

(The use of a robot to learn programming)

Ime in priimek: Ana Đorđević
Študijski program: Računalništvo in informatika
Mentor: doc. dr. Branko Kavšek

Koper, september 2016

Ključna dokumentacijska informacija

Ime in PRIIMEK: Ana ĐORĐEVIĆ

Naslov zaključne naloge: Uporaba robota pri učenju programiranja

Kraj: Koper

Leto: 2016

Število listov: 46 Število slik: 2 Število tabel: 2

Število prilog: 3 Število strani prilog: 3 Število referenc: 38

Mentor: doc. dr. Branko Kavšek

Ključne besede: robot, robotika, učenje programiranja, poučevanje programiranja, motivacijski pristop, programer začetnik, vizualizacijska programska orodja

Izvleček:
Glavni cilj zaključne naloge je preučiti motivacijski vidik uporabe robota pri učenju programiranja osnovnošolskih otrok. V teoretičnem delu naloge je predstavljena zgodovina poučevanja programiranja, pedagoški vidiki učenja programiranja in učenje programiranja skozi stopnje izobraževanja ter motivacijski pristopi učenja za začetnike, ki smo jih uporabili pri izvedbi delavnice učenja programiranja za osnovnošolske otroke. V eksperimentalnem delu je podan opis delavnice in njene izvedbe. Kot glavni motivacijski pristop k učenju programiranja na delavnici smo uporabili robota TiddlyBot, enostavnega robota, ki lahko riše, sledi črtam, zaznava ovire, se oglaša in ima tri lučke, ki jih lahko prižiga in ugaša, upravljamo pa ga lahko v vizualnem programskem okolju. Na delavnici so učenci skušali naučiti robota prevoziti poligon, ki je zasnovan tako, da je potrebno uporabiti vse robotove osnovne funkcije in upoštevati vnaprej določena navodila.

Key words documentation

Name and SURNAME: Ana ĐORĐEVIĆ

Title of final project paper: The use of a robot to learn programming

Place: Koper

Year: 2016

Number of pages: 46 Number of figures: 2 Number of tables: 2

Number of appendices: 3 Number of appendix pages: 3 Number of references: 38

Mentor: Assist. Prof. Branko Kavšek, PhD

Keywords: robot, robotics, teaching programming, learning programming, motivational approach, novice programmer, visualization programming tools

Abstract:
The main goal of the thesis is to study the motivational aspect of the use of a robot to learn programming in elementary school children. The theoretical part of the thesis presents the history of teaching programming, the pedagogical aspects of learning programming and learning programming through the process of education as well as motivational approaches of learning programming for beginners, which we used in preparing and executing a programming workshop for elementary school children. In the experimental part the description of the programming workshop for elementary school children and its execution is described. At the workshop we used the TiddlyBot robot, a simple robot that can draw and follow lines, perceive barriers, make sounds, has three lights that can be switched on and off and can be programmed through a visual programming environment, as the main motivational approach to learn programming. At the workshop children tried to teach the robot to drive through a polygon, which was designed in a way for them to use all the basic functions of the robot and take into account all predetermined instructions.

Zahvala

Rada bi se zahvalila mentorju zaključne naloge doc. dr. Branku Kavšku za ves trud in tehnično podporo, ki mi jo je nudil med pisanjem zaključne naloge ter ostalim pedagoškim delavcem na UP FAMNIT, ki so mi omogočili kakovostno izobraževanje.

Zahvalila bi se rada še sestri, staršem in vsem, ki so od samega začetka verjeli vame, za vso njihovo podporo in potrpežljivost v času mojega študija.

Hvala vam.

Kazalo vsebine

Seznam tabel

Seznam slik

Seznam prilog

Seznam kratic

OŠ	Osnovna šola
HTML	Označevalni jezik za izdelavo spletnih strani (HyperText Markup Language)
IKT	Informacijsko-komunikacijske tehnologije
LED	Svetleča dioda (Light-Emitting Diode)
RS	Republika Slovenija
IP	Številka, ki natančno določa računalnik v omrežju Internet (Internet Protocol)
URL	Enolični krajevnik vira - naslov spletnih strani v svetovnem spletu (Uniform Resource Locator)
IR	Infrardeče območje spektra elektromagnetnega valovanja

1 UVOD

Programiranje je eno pomembnejših področij računalništva, vendar se številnim učencem učenje programiranja ne zdi enostavno in se pri učenju spopadajo s številnimi težavami. Zato je pri učenju programiranja ključni dejavnik motivacija, saj le motivirani učenci učenje začnejo, se učijo (sprašujejo, poslušajo, sodelujejo, preizkušajo, berejo, razmišljajo, primerjajo, doživljajo, vrednotijo, ustvarjajo ...) in pri učenju vztrajajo, vse dokler ne končajo učnih nalog ali ne dosežejo zastavljenih učnih ciljev [12]. Na motivacijo učencev lahko vpliva učitelj, z uporabo razlčnih pristopov poučevanja. Glavni cilj zaključne naloge je preučiti motivacijski vidik uporabe robota pri učenju programiranja osnovnošolskih otrok.

V 2. poglavju zaključne naloge naprej predstavimo problematiko. Na kratko povzamemo splošno zgodovino programiranja in zgodovino učenja programiranja v Sloveniji ter primerjamo tradicionalni pristop učenja programiranja in poučevanje programiranja z uporabo različnih oblik poučevanja, ki so opisane v nadaljevanju naloge. Sledi primerjava kognitivizma in konstruktivizma - teorij, na katerih je osnovano problemsko učenje, opišemo algoritmično razmišljanje, problemsko učenje ter motivacijske pristope. Na koncu predstavitve problematike opišemo učenje programiranja skozi stopnje izobraževanja, v osnovni šoli, srednji šoli in na fakulteti.

V naslednjem, 3. poglavju, sledi opis motivacijskega pristopa učenja programiranja za začetnike, ki smo ga uporabili pri izvedbi delavnice učenja programiranja. Najprej opredelimo, kdo so začetniki, predstavimo izbiro programskega jezika ter uporabo robota, kot motivacijski pristop, nato pa predstavimo Raspeberry Pi [37] robota Tiddlybot [33] in njegove senzorje.

Četrto poglavje predstavi eksperimentalni del zaključne naloge, kjer opišemo zasnovo in izvedbo delavnice učenja programiranja za osnovnošolske otroke s pomočjo robota Tiddlybot in rezultate anketiranja. V 5. poglavju je podana interpretacija rezultatov. Zadnje, 6. poglavje, predstavi zaključek ter možnosti nadaljnjega dela.

2 PREDSTAVITEV PROBLEMATIKE

2.1 Zgodovina učenja programiranja

Programiranje ne velja za preprosto sposobnost, saj povezuje več sposobnosti, kot so analiza, specifikacija, oblikovanje algoritmov in konstrukcija programske kode. Cilj poučevanja programiranja je, da se učenci naučijo sistematičnega pristopa oblikovanja računalniških programov. V nadaljevanju je podan kratek pregled splošne zgodovine učenja programiranja in zgodovine programiranja v Sloveniji.

2.1.1 Splošna zgodovina učenja programiranja

Začetki poučevanja programiranja segajo v 60-ta leta, ko so na univerzah začeli s poučevanjem višjih programskih jezikov. Večji napredek nato sledi v 80-ih letih, ko so v šole uvedli uporabo računalnikov in je programiranje postalo del učnega načrta. Leta 1986 je v Združenih državah Amerike računalnik uporabljalo 25% srednjih šol [30]. Primarni način učenja programiranja je bilo tradicionalno poučevanje, kjer so imeli učenci pasivno vlogo [29]. Tradicionalni pristop poučevanja programiranja zajema učenje sintakse programskega jezika ter načrtovanje in oblikovanje programa, vendar je potrebno k temu dodati še druge vidike, kot na primer spreminjanje in testiranje spremenljivk v programu.

V tabeli 2.1 na koncu tega poglavja je podana primerjava različnih metod učenja. Tradicionalno poučevanje poteka skozi učne materiale, kot so učbeniki, zapiski, tabla, večpredstavnostne predstavitve ipd., ki so uporabni za predstavitev že končanega programa, ne pa tudi za predstavitev samega procesa programiranja - ustvarjanja programa [13].

Winslow je pojasnil, da učencem ne predstavlja problem sintaksa in semantika izjav v programu, pač pa sestava delujočega programa [1]. Učitelji se osredotočajo na sintakso programskega jezika [13] prav tako učbeniki pomagajo učencem razumeti programski jezik in struktutro programov, vendar ne pokažejo načina kako programirati ali samega procesa programiranja [4].

Učenci se v šoli učijo programiranja v programskem jeziku, ki je določen v kuriku-

lumu. Na začetku je poudarek na sintaksi, kasneje pa se v učenje vključi še semantika programskega jezika [5]. Torej, učenje se začne z učenjem lastnosti in sintakse programskega jezika ter pisanjem enostavnih programov, ki se postopoma nadgrajuje s kombiniranjem več ukazov skupaj. Pri učenju programiranja z učnimi materiali tradicionalnega poučevanja programiranja lahko učencem predstavimo idealne rešitve, kar pa lahko povzroči težave, saj oni svoje rešitve oblikujejo na podoben način in če ne zmorejo oblikovati pravilne rešitve, obtožujejo sebe in se počutijo nesposobne, posledično pa niso več samozavestni in motivirani za učenje programiranja [4].

Tradicionalni pristop učenja programiranja pa ni ravno idealen za konstrukcijo mentalnih modelov konceptov programiranja [21], zato se pri poučevanju programiranja predlaga uporaba različnih tehnik poučevanja, da bi učence naučili abstraktnega mišljenja in spretnosti, ki so poglavitne pri programiranju [14]. Pri poučevanju programiranja se za motiviranje učencev in učinkovitejše učenje uporabljajo nove metode poučevanja, kot so problemsko učenje, projektno delo, sodelovalno učenje, učenje z didaktičnimi igrami, uporaba vizualizacije ipd. Te metode poučevanja so se razvile na osnovi teorij kognitivizma in konstruktivizma [13], kjer ima učenec aktivno vlogo pri učenju. Na te oblike poučevanja in na kurikularne spremembe je leta 1980 vplival Papert, ki je zagovarjal programiranje, kot pomembno izobraževalno področje [22]. Raziskave kažejo, da so študenti, ki obiskujejo tečaje tradicionalnega poučevanja programiranja, pri opravljanju izpita 1,5-krat bolj neuspešni kot pa študenti, ki imajo pri učenju aktivno vlogo [29].

Čeprav ima učenec aktivno vlogo pri učenju, je pri poučevanju ključna učiteljeva vloga. Njegova naloga je, da ustvari spodbudno učno okolje, v katerem lahko učenci odkrivajo, gradijo in oblikujejo svoje znanje ter razvijajo kritičnost in odgovornost. Poučevanje naj bi temeljilo na tem, da učenci aktivno osvajajo nova znanja, vendar se mora upoštevati predznanje vsakega učenca. Učitelj mora poskrbeti tudi za ustvarjanje okoliščin, ki učence motivirajo za učenje ter spodbujajo skrb za pridobivanje znanja [15].

2.1.2 Učenje programiranja pri nas

Začetki poučevanja računalništva na slovenskih srednjih šolah segajo v leto 1971. Med pionirkse države vpeljevanja IKT v vzgojno-izobraževalni proces je Slovenijo popeljal takratni svetovalec Zavoda za šolstvo Branko Roblek, s projektom Uvajanje računalniške pismenosti v srednje šole, v katerem je prvo leto sodelovalo 200 učencev, do šolskega leta 1974/75 pa se je to število poveečalo na 2500 [16].

S takratnim stanjem strojne in programske opreme je bil pouk v večini teoretičen, s poudarkom na algoritmih in programskih jezikih. Po učnem načrtu so se učenci sezna-

nili z osnovnimi pojmi, kot so računalnik, algoritem, podatek, informacija, programski jezik, binarni, oktalni in heksadecimalni številski sistem, razvojem računalnikov, uporabo in zgradbo računalnikov ter diagrami poteka. Druga polovica predmeta pa je bila namenjena programiranju.

Organizirana informatizacija v osnovnih šolah se je pričela leta 1985, ko je delovna skupina za računalništvo Zavoda SRS za šolstvo in šport organizirala 150 urno izobraževanje za učitelje računalniških interesnih dejavnosti in prvo tekmovanje iz računalništva za osnovnošolce [16]. Leta 1986 so na Institutu Jožef Stefan med svoje tečaje iz računalništva za mladino vključili tudi tečaj programskega jezika Logo, ki je bil kasneje vključen tudi v dopolnilno izobraževanje iz računalništva za osnovnošolske učitelje na Oddelku za matematiko Univerze v Ljubljani in na Pedagoški fakulteti v Mariboru [16].

Leta 1994 se je v Sloveniji začel šestletni program Računalniško opismenjevanje [16], katerega namen je bil dvigniti raven informatizacije slovenskega šolstva in s tem pomembno prispevati k učinkovitejši, sodobnejši, ustvarjalnejši in prijaznejši vlogi vzgojno izobraževalnih zavodov. Naslednja prenova učnega načrta je sledila leta 1995, kjer je bil večji poudarek na obdelavi besedil in izdelavi daljše seminarske naloge z uporabo sodobnega urejevalnika besedil [16]. Kurikularna prenova v letu 1998, katere cilj je bilo razvijanje didaktike računalniškega poučevanja je v osnovne šole pričela uvajati uporabo računikov pri pouku slovenščine, likovne in tehnične vzgoje. S tem so bile v šole vpeljane novosti, kot so timsko poučevanje, sodelovalno učenje, razvojne skupine ipd [16].

Z razvojem računalništva in informatike ter razširjenostjo uporabe IKT se sorazmerno razvijajo tudi učni načrti predmetov. Vsi učenci morajo pridobiti temeljna teoretična znanja, vendar je poudarek na učenju uporabe računalnika.

Po uvedbi devetletke so lahko v osnovni šoli kot izbirni predmet učenci računalništvo naprej izbrali v 7., 8. in/ali 9. razredu, od šolskega leta 2014/15 pa se predmet izvaja za učence 4., 5. in/ali 6. razreda [15]. Pri tem predmetu učenci pridobijo temeljna znanja računalniško-informacijske pismenosti, ki so potrebna v življenju in nadaljnem izobraževanju, spoznajo in razumejo osnovne pojme in zakonitosti ter se spoznajo z delom z računalniki. V vseh treh letih obiskovanja predmeta se učenci seznanijo z urejanjem besedil ter pridobijo znanje, potrebno za razumevanje in temeljno uporabo računalnika ter se seznanijo z računalniškimi omrežji in multimedijo. Skozi celoten učni proces se učenci seznanjajo z osnovnimi oblikami dela, kot so: skupinsko delo, problemsko učenje, izbiranje vsebin glede na zanimanje in sposobnost učencev, individualizacija, vključevanje raznih socialnih aktivnosti, povezovanje predmeta z drugimi predmeti, sodelovanje z zunajimi strokovnjaki ter razvijanje različnih strategij mišljenja. Pri tem učenci razvijajo sposobnost ustvarjalnega in kritičnega mišljenja ter presojanja in si s

tem zagotovijo razumno in samozavestno odločanje v novih, nepredvidljivih situacijah.

V srednji šoli se učenci srečajo s predmetom informatika [11], ki predstavlja nadgradnjo osnovnošolskega predmeta računalništvo [16]. Ker je računalništvo v osnovni šoli izbirni predmet, je pomembna naloga učiteljev informatike, da pri svojem načinu poučevanja upoštevajo predznanje svojih dijakov. Predmet informatika je obvezen predmet, katerega osnovni cilj je organizirana in sistematična uporaba računalnika in informacijskih sistemov v vsaskdanjem življenju in nadaljnjem šolanju. Kot je določeno z učnim načrtom dijaki spoznajo osnovne pojme informatike, razvoj, zgradbo in delovanje računalnika, področje njegove uporabe ter osnovne vrste podatkov, ki so lahko računalniško obdelani, poudarek pa je na pisni predstavitvi informacij. Dijaki večino teoretičnega znanja utrdijo z izdelavo projektnih nalog, ki vključujejo zbiranje podatkov, njihovo ovrednotenje in obdelavo ter uporabo podatkov za predstavitev ostalim dijakom.

Predmet informatika je v višjih letnikih gimnazije na voljo tudi kot izbirni predmet [19]. Tu se dijaki seznanijo z računalniškimi omrežji in računalniškimi predstavitvami, poglobijo znanje o računalniški obdelavi slik, zvoka in videa ter teoretično spoznajo različne storitve Interneta. Svoje znanje nadgradijo tudi z izdelavo podatkovnih baz, odločitvenih modelov in preprostega računalniškega programa v izbranem programskem jeziku.

2.2 Pedagoški vidiki učenja programiranja

Kognitivizem [13] in kostruktivizem [13] sta teoriji, na katerih je osnovano problemsko učenje. V nadaljevanju sledi njuna razlaga in primerjava ter predstavitev računalniškega in algoritmičnega razmišljanja, problemskega učenja in različnih motivacijskih pristopov k poučevanju programiranja.

2.2.1 Kognitivizem, konstuktivizem – podobnosti in razlike

Kognitivizem je izobraževalna smer, ki namesto pasivne absorbcije znanja iz knjig in učbenikov, zagovarja pomen aktivne udeležbe učencev v procesu učenja. Začetnik in utemeljitelj kognitivizma je švicarski psiholog Jean Piaget [35], katerega teorija temelji na tem, da morajo učenci sami pridobiti znanje.

Učitelji z opazovanjem učencev med učnim procesom pridobijo vpogled v njihove zmožnosti in učne navade. Način učenčevega razmišljanja, znanje, pričakovanja, občutki in interakcije z okoljem so pomembni dejavniki, ki vplivajo na način učenja in izbor vsebin, saj na podlagi teh dejavnikov učitelji pripravijo naloge, ki učencem predstavljajo izziv. Pri reševanju problemov se poveča aktivnost učencev in posledično tudi

produktivnost učnega procesa.

Pomemben vidik kognitivizma je prav tako ponotranjenje znanja na podlagi lastnih izkušenj. Psihološke struktrue nam omogočajo, da lažje razumemo okolico, odnose v njej in reakcije nanjo. Te strukture je Piaget poimenoval sheme in v njegovi teoriji kognitivizma predstavljajo osnovni element mišljenja [28].

Kognitivisti so mnenja, da morajo učenci v procesu učenja ne samo odkrivati obstoječa dejstva ampak tudi graditi miselne strukture. Kognitivistični učitelj ima zahtevnejšo vlogo kot učitelj pri klasičnem poučevanju, saj mora razumeti vsakega učenca, opredeliti fazo kognitivnega razvoja v kateri se učenec nahaja in na podlagi tega prilagoditi učni proces [3].

Piaget je definiral štiri faze kognitivnega razvoja, ki so razdeljene glede na starost posameznika in njegovo sposobnost mišljenja [28]:

- senzomotorična faza: prva faza v človekovem razvoju, v kateri se otrok spoznava s svetom in okolico ter začne uporabljati spomin in mišljenje. V to fazo vključujemo gledanje, poslušanje, gibanje, dostikanje in okušanje.

- predoperacionalna faza: začetek postopne uporabe jezika, pri komunikaciji uporablja razne simbole, besede, znake in podobno. Otrok okolico dojema predvsem iz svojega zornega kota, njegovo razmišljanje je omejeno in postopoma razvija sposobnost mišljenja v simbolni obliki.

- konkretne operacije: faza, v kateri je posameznik sposoben rešiti konkretne primere na logičen način.

- formalne operacije: faza, v kateri posameznik razume abstraktne pojme in ima zelo dobro razvite predstave. Sposoben je analizirati in reševati probleme ter uporablja ločigno mišljenje.

Prehajanje ljudi med temi fazami je različno, vendar je zaporedje faz enako.

Kognitivizem se je na začetku uveljavil na področju matematike in naravoslovnih znanosti, kasneje pa tudi na področju računalništva.

Konstuktivizem je teorija učenja, ki zagovarja in poudarja, da je učenje aktiven proces, ki temelji na učenčevih izkušnjah. Teorijo konstruktivizma je razvil ameriški matematik, računalničar in pedagog Seymour Papert, ki meni, da je ključ učenja izražanje notranjih občutkov in idej učenca. Emotivno izražanje posamezniku omogoča lažjo vključitev v družbo in lažje komuniciranje, povezovanje in širjenje svojih obzorij [2]. Ključnega pomena je prav tako spodbudno okolje, ki posameznika motivira za učenje ter postavljanje izvirnih in inovativnih vprašanj. Ker gre za pridobivanje znanja preko izkušenj, je potrebno učence spodbujati k ustvarjanju in eksperimentiranju, prav

tako pa mora biti učenje relevantno in v kontekstu z učno snovjo. Učenci lahko pri raziskovanju svojih interesov uporabljajo tudi različne tehnologije. Z raziskovanjem učenci razvijajo kognitivne, problemske in logične veščine in omogočajo, da so ideje posameznikov inovativne, njihovo znanje pa dolgoročno [20].

Piagetova teorija kognitivizma in Paparetova teorija konstruktivizma združujeta isti pogled na učenje in sicer gradnjo miselnih struktur z aktivnim učenjem, pri konstukrivizmu je poudarek še na učenčevem zavestnem sodelovanju pri pridobivanju znanja. Obe teoriji zagovarjata, da morajo biti posametniki aktivni v procesu učenja in da morajo svoje znanje graditi na kreativen način. Glavna razlika med teorijama je, da je pri kognitivizmu poudarek na gradnji notranjih struktur pri posamezniku, pri konstruktivizmu pa je poudarek na spremembah v okolju, ki vplivajo na znanje posameznika. Bolj podrobno je razlike med teorijama predstavila Ackermann [2]:

- Teorija kognitivizma temelji na posameznikovih notranjih strukturah in organizaciji znanja na različnih stopnjah razvoja. Poudarek je na postopnem raziskovanju in razumevanju sveta. Na začetku posamezniki operirajo s konkretnimi stvarmi, skozi razvoj pa pridobijo sposobnost manipulacije z navideznimi objekti in simboli znotraj hipotetičnega sveta.

- Teorija konstruktivizma poudarja, da se mora znanje definirati in preučevati na samem kraju. Za Paperta inteligentnost pomeni občutljivost, odbornost in prilagodljivost na spremembe v okolju. V nasprotju s kognitivizmom je tu poudarek na dejtvu, da se mora posameznik poglobiti v situacijo in je ne le opazovati iz razdalje.

2.2.2 Algoritmično razmišljanje

Računalniško razmišljanje lahko opredelimo kot miselni proces, ki se odraža kot sposobnost dekompozicije in reševanja problemov, abstraktnega in algoritmičnega razmimšljanja ter evalvacije in posploševanja [7]. Pomemben gradnik računalniškega razmišljanja je algoritmično razmišljanje, ki je bistvenega pomena pri programiranju, saj nam pomaga pri reševanju računalniških problemov. Algoritmični način razmišljanja je eden ključnih elementov funkcionalne pismenosti, ki nam koristi pri branju navodil, sledenju postopkov, predstavitvi problema, boljšemu razumevnaju tehnoloških naprav, ustvarjanju računalniških iger in drugih aplikacij [27].

Bistvo algoritmičnega razmišljanja je, da dan problem rešujemo postopoma. Algoritmično razmišljanje zajema sledeče aktivnosti [9]:

- analizo danega problema,

- natančno specifikacijo problema,

- iskanje osnovnih korakov za rešitev problema,
- priprava algoritma na podlagi osnovnih korakov, ki bo rešil dani problem,
- razmišljanje o različnih možnostih izvedbe algoritma ter robnih pogojih,
- izboljšanje učinkovitosti algoritma.

Tehnike, ki jih učenci uporabljajo pri izvajanju aktivnosti algortimičnega razmišljanja so sledeče [7]:

- pisanje navodil, ki si sledijo v danem vrstnem redu, za dosego željenega rezultata,
- pisanje navodil, ki vsebujejo aritmetične in logične operacije, za dosego željenega rezultata,
- pisanje navodil za manipulacijo s podatki, za dosego željenega rezultata,
- pisanje navodil za omogočanje ponavljanja določenih lastnosti, za dosego željenega rezultata,
- pisanje zaporedij navodil, za dosego željenega rezultata,
- uporaba standardne notacije za predstavitev vseh zgodaj opisanih lastnosti,
- kreiranje algoritmov za testiranje hipotez,
- kreiranje algoritmov, zasnovanih na resničnem svetu, za boljše razumevanje.

Algoritmično razmišljanje je odvisno od abstraktnega in logičnega razmišljanja, razmišljanja v strukturah, kreativnosti ter kompetenc reševanja problemov. Tak način razmišljanja velja za kompleksen in nenaraven način razmišljanja, ta kompleksnost pa lahko povzroči težave in nerazumevanje pri začetnikih [9]. Ta stereotip pa izvira predvsem iz njegovega nepoznavanja, saj se s tem načinom razmišljanja srečujemo v vsakdanjem življenju ter ga uporabljamo pri reševanju problemov in opravljanju vsakodnevnih opravil, kot na primer pot v službo, kuhanje kave, gradnja hiše ipd.

2.2.3 Problemsko učenje

Problemsko zasnovano učenje je učenje z reševanjem realističnih problemov iz življenja, ki so ga razvili na zdravstveni šoli McMaster University v Kanadi leta 1960 [36]. Najprej je potrebno razumevanje samega problema, nato njegova razlaga in šele nato razrešitev [17]. V večini primerov je možnih več pravilnih rešitev in načinov reševanja [28].

Problemsko zasnovan pouk poteka v sedmih korakih [17]:

1. Predstavitev problema
2. Identifikacija problema
3. Vihrarjenje možganov
4. Pregled oziroma skica obstoječega problema
5. Postavitev učnih ciljev
6. Samostojno doseganje zastavljenih ciljev
7. Elaboracija.

Pomembno vlogo ima tukaj učitelj, saj mora dobro zastaviti problemsko situacijo, ki jo učenci nato poskušajo rešiti. Problem mora ustrezati učni vsebini, biti mora razumljiv ter prilagojen dosedanjemu znanju in sposobnostim učencev, hkrati pa mora v učencih vzbuditi zanimanje, saj je učni proces bolj uspešen, če učenca problemska situacija pritegne. Problemsko zasnovan pouk spodbuja skupinsko učenje in učence uri za delo v skupini, vodenje skupine ter jim omogoča razvoj psihosocialnih kompetenc [17].

Kompleksne probleme učenci rešujejo tako, da se problem razdeli na več manjših korakov. Vsak korak je potrebno opredeliti tako, da je za učence rešljiv, saj lahko naslednji korak začnejo reševati šele, ko je predhodni uspešno zaključen [23].

Učenci si s problemskim učenjem zapomnijo več informacij, za učenje so bolj motivirani, odkrivajo in razvijajo nova znanja ter razvijajo veščine kritičnega vrednotenja in informacijske pismenosti [17].

2.2.4 Motivacijski pristop

Psihološko gledano, se učna motivacija razume kot posebna vrsta motivacije, ki jo učenec izraža v okviru šolskega učenja. Motivacija učni proces sprva aktivira, nato pa ga usmerja do zaključka učne naloge oziroma učnega cilja [6]. Motivacijo lahko učitelji v učencu z različnimi motivacijskimi spodbudami negujejo, ozaveščajo, krepijo in spodbujajo ali pa lahko dosežejo nasproten učinek.

Motivacijske spodbude so lahko didaktične, kot na primer organizacija učnega okolja in učenja, učne metode, izbira nalog in didaktičnega materiala ali pa psihološke, kot na primer vodenje učenca skozi proces učenja s povratnimi informacijami o njegovem znanju in dosežkih, omogočanje učne podpore in usmerjanje pri učenju ter ravnanje učitelja kot vzor, katerega pristop k učnim nalogam, učenju in razlagi rezultatov učenci posnemajo. Pomembni elementi, ki motivirajo učenca so logika, spomin, vizualnizacija in reševanje problemov [8].

Eden izmed problemov, s katerimi se spopadajo učitelji programiranja je, da v učencih težko vzbudijo zanimanje in jih motivirajo za programiranje [4]. Učinkovit in motivacijski pristop je učenje programiranja z vizualizacijskimi programskimi orodji.

Eden izmed motivacijskih pristopov, ki ga uporabljajo učitelji programiranja je učenje z didaktičnimi igrami, saj ta način zajema elemente notranje motivacije, kot so izziv, radovednost in domišljija [10] ter poživi učno okolje. Didaktična igra je igra, ki ustreza vnaprej določenim izobraževalnim ciljem, ki jih dosežemo z aktivno vlogo udeležencev [24] ter povezuje igranje in učenje.

Pristop, ki lahko poveča samozavest in motivacijo učencev je tudi sodelovalno učenje oziroma natančneje programiranje v paru [13]. Učenje mora biti podprto z dobrim programskim okoljem, zato k motivaciji učenja programiranja prispevajo tudi izobraževalna vizualizacijska orodja in programski jeziki, ki so oblikovani prav za učenje programiranja.

Prav tako lahko kot motivacijski pristop uporabimo robotiko, ki velja za edukacijski način za učinkovito razumevanje programiranja in inženirskih načel. Uporablja se v osnovni šoli, srednji šoli in fakulteti. Učencem robotika predstavlja spodbudo in motivacijo pri učenju programiranja, saj jih vodi misel "Naučil bom robota, da naredi to kar hočem!" [25].

2.3 Učenje programiranja skozi stopnje izobraževanja

V nadaljevanju sledi opis vsebine izbirnega predmeta Računalništvo v osnovni šoli, obveznega ali izbirnega predmeta Informatika ter izbirnega predmeta Računalništvo v srednji šoli in predstavitev učenja programiranja na fakulteti.

2.3.1 Osnovnošolsko programiranje

Računalništvo je v osnovni šoli neobvezni izbirni predmet, ki ga lahko obiskujejo učenci četrtega, petega ali šestega razreda in pridobijo znanja iz različnih področij računalništva. Pri predmetu se učenci seznanijo s temeljnimi računalniškimi koncepti in procesi, tehnikami in metodami reševanja problemov, razvijajo algoritmičen način razmišljanja in spoznavajo omejitve računalnikov. Pri predmetu se s pristopi poučevanja spodbuja ustvarjalnost, sodelovanje in poseben način razmišljanja ter delovanja učencev [15].

2.3.2 Srednješolsko programiranje

Na srednjih šolah se Informatika izvaja kot obvezni predmet v prvem in drugem letniku. Informatika je splošno-izobraževalni predmet, ki daje dijakom osnovna znanja informacijske pismenosti, ki so nujno potrebna pri nadaljnjem izobraževalnem in osebnostnem razvoju. Na gimnaziji se Informatika poučuje tudi kot izbirni predmet za maturo in sicer v četrtem letniku, kjer se lahko dijaki naučijo tudi programiranja. Pri predmetu utrdijo znanje iz prvega in drugega letnika, pridobijo pa nova znanja o infomacijskih tehnologijah, podatkovnih bazah ter algoritmih in programiranju [11]. Na tehniški gimnaziji lahko dijaki kot izbirni predmet izberejo Računalništvo. Predmet vključuje tako teoretična kot praktična znanja s področja strojne in programske opreme računalnika, algoritmov in programskih jezikov, objektnega programiranja in implementacije algoritmov v izbranem programskem jeziku, računalniških omrežij, metodologij in tehnik iskanja in zbiranja podatkov, informacijskih sistemov in podatkovnih baz. Predmet je zasnovan tako, da dijaki dobijo vpogled v pet temeljnih računalniških discipin, ki poleg temeljnih znanj računalništva vključujejo še računalniško inženirstvo, razvoj programske opreme, informacijske tehnologije in informacijske sisteme in tako pridobijo dobro osnovo za nadaljnje izobraževanje na tehniških, naravoslovnih in tudi družboslovnih fakultetah. Predmet prinaša dijakom možnost sistematičnega razvijanja digitalnih kompetenc in hkrati nadgrajuje tudi druge pomembne kompetence, kot so sporazumevanje v maternem in tujem jeziku, osnovno znanje na področju znanosti in tehnologije, socialne kompetence, učenje učenja, samoiniciativnost in podjetnost, ki jim bodo omogočile uspešen študij na univerzi ter uspešno poklicno in življenjsko pot.

Cilji učenja programiranja so, da dijaki ob koncu šolanja [19]:

- poznajo pomen načrtovanja in sistematične gradnje programa,
- poznajo pojem algoritma in opredelijo njegove lastnosti (razumljivost, končnost, enoumnost, razčlenjenost),
- znajo opisati načine zapisa algoritma in jih prikazati na primeru,
- znajo izdelati algoritem za lažji problem,
- znajo opredeliti pojem programskega jezika in razložiti njegovo funkcijo,
- poznajo različne vrste programskih jezikov in jih razvrstijo po namenu in uporabi,
- poznajo temeljne gradnike postopkovnega programskega jezika, razložijo njihove funkcije in razlago ponazorijo s primeri (zaporedje, vejitev, iteracija),
- znajo opredeliti strukturirano in objektno programiranje,

- ločijo prevajanje in tolmačenje ter znajo razliko razložiti,
- poznajo osnovno razvojno okolje,
- poznajo pojme deklaracija, inicializacija, postopek, konstanta, spremenljivka, rezervirana beseda, operator, prioriteta ...
- ločijo med pojmi izvorna koda, izvršljiva koda in vmesna (byte) koda, pridobljeno znanje o tipih in algoritmih prenesejo v programski jezik in samostojno rešujejo probleme s pomočjo višjega programskega jezika,
- poznajo in uporabijo osnovne in sestavljene tipe podatkov,
- znajo uporabiti pogojni stavek,
- iteracijo realizirajo z zankami,
- uporabijo tokove podatkov,
- pripravijo testne podatke, testirajo delovanje programa in beležijo rezultate testiranj, uporabijo razhroščevalnik,
- napišejo sled programa,
- znajo izdelati dokumentacijo programa,
- znajo uporabiti metode za delo z objekti razredov,
- znajo napisati definicijo razreda, lastnosti, metode ...,
- deklarirajo in uporabijo objekte,
- poznajo vlogo in način izvajanja konstruktorja,
- uporabijo enkapsulacijo, dedovanje in polimorfizem,
- poznajo načine za prestrezanje in obravnavo izjem,
- znajo napisati programe za enostavne probleme iz okolja,
- znajo analizirati program in ovrednotiti rezultate, dobljene s programsko rešitvijo (različni algoritmi urejanja podatkov, različni načini zapisovanja podatkov v datoteke ...),
- znajo predstaviti delovanje programa,
- znajo analizirati in kritično vrednotiti rešitve,

- poznajo zahtevnejše tehnike programiranja,
- znajo napisati program z uporabo zahtevnejših tehnik programiranja.

2.3.3 Programiranje na fakulteti

Študenti na fakulteti najprej opravijo obvezne predmete, ki jim dajo osnovna matematična, teoretična in strokovna znanja, nato pa opravljajo izbirne predmete iz različnih področij računalništva, kot so spletne tehnologije, programska oprema, strojna oprema, informacijski sistemi ipd. [32].

Učenje programiranja študenta najprej uvede v programersko razmišljanje nato pa študenti razširijo in poglobijo svoje programeske sposobnosti. Spoznajo različne programske jezike in pojme, ki se nanašajo na konkretne programske konstrukte (podatkovni tipi, spremenljivke, izrazi, operatorji, pomnilnik ...) ter različne programerske pristope, ki se uporabljajo v praksi in načine reševanja problemov. Naučijo se dela z datotekami, izjemami in podatkovnimi zbirkami ter izdelave programa z vsemi elementi grafičnega uporabniškega vmesnika.

Učenje programiranja je večinoma praktično naravnano oziroma je osnovano na izdelavi projektov. Za vsak predstavljen teoretični koncept ali programski konstrukt je na predavanjih prikazana tudi njegova uporaba v konkretnih primerih. Preko manjših nalog, ki jih študentje rešujejo na vajah in doma kot domače naloge ter večjih projektov, pa imajo priložnost, da pridobljeno znanje utrjujejo in poglabljajo [32].

Učna metoda	Prednosti	Slabosti
• Programiranje na tabli	• Učenci lahko sledijo • Interakcija med učiteljem in učencem	• Na tablo lahko pišemo le manjše programe
• Dokončani programi na prosojnicah	• Predstavimo večje in kompleksnejše programe	• Razlaga je lahko prehitra in učenec ne sodeluje v procesu nastajanja programa
• Programiranje v živo (z uporabo računalnika in projektorja)	• Programe lahko zaženemo in testiramo s pomočjo orodij	• Omejen čas v razredu • Predstavitev ni shranjena in kasnejši ogled ni možen
• Proces snemanja (z uporabo računalnika, mikrofona in programa za snemanje)	• Vzamemo si čas, ki ga potrebujemo za predstavitev kompleksnega problema • Predstavitev je shranjena in si jo lahko naknadno večkrat ogledamo	• Ni prikazana uporaba različnih orodij • Razlage na posnetkih so navadno v naprej pripravljene in ne prikazujejo napak ter možnih rešitev

Tabela 2.1: Primerjava različnih metod učenja

3 MOTIVACIJSKI PRISTOP UČENJA PROGRAMIRANJA ZA ZAČETNIKE

3.1 Kdo so začetniki

Učenje programiranja ni enostavno, zato se začetniki spopadajo z različnimi težavami, imajo omejeno znanje, pomanjkljive mentalne modele in svojega znanja ne uporabljajo ustrezno.

V zaključni nalogi, kot začetnike opredelimo učence druge triade osnovne šole, ki obiskujejo neobvezni izbirni predmet Računalništvo. Cilj predmeta je, da se učenci seznanijo s temeljnimi računalniškimi koncepti in procesi, tehnikami in metodami reševanja problemov, razvijajo algoritmičen način razmišljanja ter spoznavajo omejitve računalnikov.

3.2 Izbira programskega jezika

Kateri je najprimernejši programski jezik za poučevanje programiranja? Ali je za začetno učenje programiranja bolj ustrezen proceduralni ali objektno-orientirani programski jezik, še ni znano. Ne glede na to, s katerim pristopom bi se morali učenci najprej seznaniti, je Lahtinen mnenja, da bi se učenci morali najprej naučiti osnovnih struktur programskega jezika [18]. Ni pomembno, kateri programski jezik se učenci najprej naučijo, poudarek je na pristopu učenja [8].

Pri učenju programiranja bomo kot motivacijski pristop uporabili uporabo robota Tiddlybot [33], katerega upravljanje in programiranje poteka prek vizualnega urejevalnika Blockly [34]. Scratch oziroma sorodno programsko okolje Blockly deluje po principu “povleci in spusti” grafičnih ploščič oziroma blokov, ki ustvarjajo program. Čas, ki ga učenec potrebuje, da osvoji znanje za uporabo vizualnega programskega okolja je kratek in tako lahko učenci porabijo večino časa za razvijanje algoritmov, oblikovanje programov ter razumevanje načel programiranja, prav tako pa ne more priti do napak v sintaksi.

Na voljo so bloki iz različnih področij:

- logika,
- zanke,
- matematika,
- spremenljivke,
- funkcije,
- vhod/izhod,
- čas,
- motorji,

kodo programa pa lahko izvozimo v programski jezik Python.

3.3 Motivacija = uporaba robota

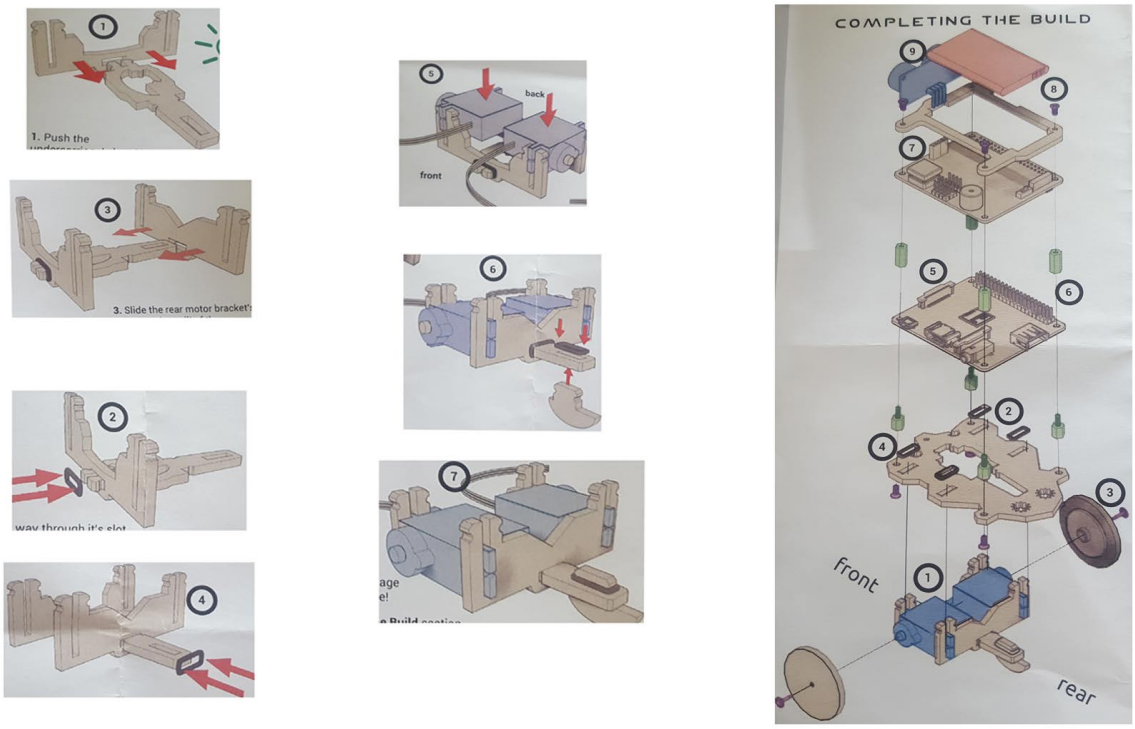
Robotika je intuitivnen, enostaven in učinkovit način za uvod v programiranje. Učenec spozna programiranje s preučevanjem, kako nadzorovati robota, da izvaja različne ukaze. Razvojni psihologi poudarjajo pomembnost uporabe fizičnih predmetov v otroštvu za uspešno razvijanje kognitivnih sposobnosti [26]. Z uporabo robotike pri učenju programiranja lahko učenci identificirajo in razumejo načela, koncepte ter prakso, ki so osnova programiranja in inženirstva, z reševanjem problemov pa osvojijo znanja, ki so se jim na začetku zdela zahtevna in nedostopna. Robotika ima na področju učenja posebno vlogo, saj je multidisciplinirana in zajema sinteze različnih tehničnih tem, vključno z algebro, dizajnom, inovacijami, elektroniko, programiranjem, silami in zakoni gibanja, materiali ter fizičinmi procesi [25]. Robotika velja za aktiven in problemsko zasnovan način učenja, ki se osredotoča na skupinsko delo. Učenci se učijo z aktvinim delom, sami pridobivajo znanje in razvijajo razumevanje na osnovi svojih aktivnosti, ki so vodene s strani učitelja. Robotika ponuja priložnost za učenje tehnik reševanja problemov, zajema različne teme, predstavlja omejitve in težave iz realnega sveta ter spodbuja kreativnost. Robotika ni odgovor ali rešitev za vsak problem, ampak daje vpogled, kako lahko "prava" tehnologija na področju problemsko zasnovanega učenja spodbuja učence k učenju osnovnih načel.

3.3.1 Robot in njegovi senzorji

Raspberry Pi [37] robot Tiddlybot [33] je nizkocenoven in učinkovit način za učenje otrok o tehnologiji in enostavnem programiranju, kjer je ključnega pomena enostavnost programske in strojne opreme.

Slika 3.1 prikazuje sestavne dele robota Tiddlybot, postopek sestave robota pa je prikazan na sliki 3.2. Sestava robota je enostavna in zabavna, saj robota sestavimo po jasnih, vizualnih navodilih, sami sestavni deli pa so enostavni.

Slika 3.1: Sestavni deli robota Tiddlybot



Slika 3.2: Postopek sestave robota Tiddlybot

Robot lahko opravlja različne funkcije. Gibanje robota omogočata dva rotacijska motorja koles, za več barvno osvetlitev ima robot 3 LED lučke, v modri, zeleni in rdeči barvi, zvok lahko robot ustvarja s pomočjo brenčala, ovire zazanava s pomočjo ultrazvočnega senzorja, črte lahko riše s pisali, ki jih vstavimo v luknje na podvozju, sledimo pa jim s pomočjo infrardečega senzorja. Robot ima tudi kamero, ki jo s pomočjo rotacijskega motorja lahko obrača, sliko kamere pa lahko spremljamo na napravi, prek katere upravljamo robota. Robot in njegovi senzorji so prikazani v prilogi C: Robot Tiddlybot. Te funkcije so nam lahko v pomoč pri učenju programiranja kot tudi pri igranju iger, v šoli ali doma.

Programska oprema omogoča oddaljeno upravljanje in programiranje robota v prilagojenem Blockly vmesniku, prek kateregakoli pametnega telefona, tablice ali osebnega računalnika, ki podpira HTML.

Postopek vzpostavitve povezave z robotom je sledeč:

- Z gumbom za vklop vključimo robota in počakamo, da lučka neha utripati,
- Na mobilni napravi se povežemo na robotovo brezžično povezavo in počakamo, da se povezava vzpostavi,
- V spletnem brskalniku z vnosnom IP naslova odpremo spletni vmesnik, prek katerega lahko upravljamo našega robota,

- V meniju aplikacije so na voljo že napisani programi za delovanje osnovnih funkcij robota, z navigacijskimi gumbi pa lahko robota poljubno vodimo po prostoru in na zaslonu naprave se prikazuje slika kamere,
- V vizualnem Blockly urejevalniku lahko napišemo svoj program za upravljanje robota in ga shranimo kot .xml datoteko.

4 EKSPERIMENTALNI DEL

4.1 Delavnica za učence osnovnih šol

Za raziskavo prednosti in učinkov uporabe robota pri učenju programiranja, smo izvedli delavnico učenja programiranja za osnovnošolske otroke s pomočjo Raspberry Pi robota - TiddlyBot. TiddlyBot smo uporabili izključno kot pomoč pri učenju programiranja. Delavnica "Kako robota nekaj naučiti?" je potekala v sklopu računalniške delavnice Orada [31] na Fakulteti za matematiko, naravoslovje in informacijske tehnologije Koper.

4.1.1 Zasnova delavnice

Delavnica je bila zasnovana tako, da so otroci poskušali robota naučiti prevoziti poligon, ob tem pa so uporabili vse osnovne funkcije robota in upoštevali vnaprej določena navodila.

Cilji delavnice so bili, da učenci:

- Razumejo robotove osnovne funkcije;
- Razumejo koncepte reševanja problemov in algoritmičnega razmišljanja;
- Spoznajo programiranje v programskem okolju Scratch [38] oz. sorodnem programskem okolju Blockly, ki ga uporablja TiddlyBot;

Delavnico smo izvedli dvakrat. Na prvi izvedbi delavnice je vsak udeleženec nalogo reševal samostojno, pri drugi izvedbi pa smo kot obliko poučevanja uporabili sodelovalno učenje in je delo potekalo v parih. Pri obeh izvedbah smo kot glavni motivacijski pristop uporabili robota, katerega so učenci skozi igro učili prevoziti poligon ob upoštevanju danih omejitev. Kot motivacijski pristop smo uporabili tudi programiranje v vizualnem programskem okolju.

4.1.2 Izvedba delavnice

V prvem delu delavnice so se učenci seznanili z robotom in njegovimi osnovnimi funkcijami ter z vizualnim programskim okoljem Blockly, prek katerega lahko upravljajo

robota. Nato smo testirali programe za vsako funckijo, ki so na voljo v robotovem uporabniškem vmesniku. Po testiranju vsake funkcije, so učenci po naših navodilih skušali napisati program, ki se je nekoliko razlikoval od programa, ki so ga pred tem preizuksili, da smo preverili njihovo razumevanje kode in utrdili njigovo znanje uporabe programskega okolja Blockly.

V drugem delu delavnice so učenci dobili skico poligona, ki je v prilogi B: Skica poligona in navodila, kako mora robot prevoziti poligon ter katere omejitve morajo upoštevati. Po pregledu poligona in navodil so začeli razmišljati, kako bodo rešili dan problem, nato pa začeli s pisanjem programa, ki robota pripelje od začetka do konca poligona po vnaprej določenih navodilih.

Poligon sestavljajo sledeči elementi: znak za obvezno smer, prehod za pešce, ovira na cestišču, semafor, tri parkirišča. Vsak element predstavlja problem, ki ga morajo učenci rešiti, da bi robot lahko uspešno prevozil poligon. V tabeli 4.1 so prikazane robotove funkcije, katere morajo učenci robota "naučiti" pri določenem elementu.

Navodila, ki jih morajo učenci upoštevati so sledeča:
(100 enot = 1 s)

1. Po pritisku na gumb robot začne z vožnjo.

2. Upoštevati mora znak "obvezna smer desno" in biti mora previden saj je 200 enot po znaku prehod za pešče.

3. Pred prehodom se mora ustaviti in počakati 3 sekunde, nato pa lahko nadaljuje z vožnjo.

4. Če je na cestišču ovira, se ji mora izogniti po levem pasu in nadaljevati z vožnjo po desnem pasu še 200 enot do "parking semaforja". Pri izogibanju ovire mora nakazati smer zavijanja z vklopom leve oziroma desne luči.

5. Pred semaforjem se mora ustaviti in počakati pritisk na gumb, s pritiskom na gumb pa naključno izbrati eno izmed treh parkirišč;

   - Če izbere modro parkirišče (1), prižge modro luč in nadaljuje z vožnjo v levo še 200 enot.
   - Če izbere rdeče parkirišče (2), prižge rdečo luč in nadaljuje z vožnjo v desno še 200 enot.
   - Če izbere zeleno parkirišče (3), prižge zeleno luč, se z vzvratno vožnjo vrne 200 enot nazaj in nato nadaljuje z vožnjo v levo še 150 enot.

   Ko konča s parkiranjem ugasne luč, ki je trenutno prižgana in z zvočnim signalom, ki traja 1 sekundo obvesti, da je uspešno prevozil poligon.

4.1.3 Rezultati anketiranja

Po končani delavnici so učenci dobili vprašalnik, ki je v prilogi A: Anketa. Vprašalnik vsebuje nekaj osnovnih vprašanj o učencu, nato pa nekaj trditev za evalvacijo pridobljenega znanja.

V prvem delu vprašalnika je učenec podal naslednje informacije: spol, starost, šola, ki jo trenutno obiskuje, ali je že kdaj prej programiral robota ter ali je že kdaj prej uporabljal vizualni urejevalnik Blockly.

Na obeh delavnicah skupaj so sodelovali štirje učenci, vsi so bili moškega spola, stari pa so bili od 11 do 14 let (učenci od 5. do 9. razreda). Trije učenci (75%) so že programirali robota in so že uporabljali vizualno programsko okolje Blockly, en učenec (25%) pa je robota prvič programiral na naši delavnici in prav tako se je prvič spoznal s programskim okoljem Blockly.

Sledi analiza trditev:

- Sam bi znal sestaviti robota po priloženih navodilih.

  V prvem delu delavnice, smo predstavili sestavne dele robota in pokazali predstavitveni video sestave robota. Dva učenca (50%) bi znala robota sestaviti sama, dva (50%) pa mogoče.

- Zapomnil sem si, kaj robot TiddlyBot vse zmore.

  V predstavitvi smo predstavili robota in njegove funkcije prikazali tudi z video vsebino, nato pa so učenci njegove funkcije uporabili pri reševanju naloge. Vsi učenci (100%) so si zapomnili, kaj TiddlyBot vse zmore.

- Znam se povezati na brezžično povezavo TiddlyBot in v brskalniku odpreti uporabniški vmesnik, prek katerega lahko robota nekaj naučimo.

  Pred uporabo robota smo robota skupaj povezali z brezžično povezavo, postopek pa smo prav tako opisali v uvodni predstaviti. Trije učenci (75%) bi se znali povezati na brezžično povezavo, en učenec (25%) pa mogoče.

- Znam uporabljati vizualni urejevalnik Blockly.

  V predstavitvenem delu smo predstavili tudi programsko okolje Blockly in vsak učenec ga je uporabljal pri programiranju robota. Vsi učenci (100%) so si bili enotni in vsi znajo uporabljati programsko okolje Blockly.

- Razumel sem navodila in vedel sem kaj moram robota "naučiti".

- Pred začetkom programiranja sem si predstavljal, kako mora robot prevoziti poligon.

Vsi učenci (100%) so na podlagi priložene slike poligona in navodil točno vedeli, kako mora robot prevoziti poligon in kakšne omejitve morajo upoštevati in si to tudi predstavljali.

- Robota sem znal naučiti vsak korak, ki je bil opisan v navodilih.

  Dva učenca (50%) sta znala robota naučiti vsak korak, en učenec (25%) je znal robota naučiti skoraj vsak korak, en učenec (25%) pa pravi da tega ni znal. Učenca, ki sta na to trditev odgovorila z "da", sta programirala v paru.

- Vsak korak, ki je bil opisan v navodilih sem robota uspešno naučil že v prvem poizkusu.

  Dva učenca (50%) sta robota uspešno naučila skoraj vsak korak že v prvem poizkusu, dva učenca (50%) pa ne. Učenca, ki sta na to trditev odgovorila z "da", sta programirala v paru.

- Na koncu delavnice sem robota naučil prevoziti poligon po priloženih navodilih.

  Dva učenca (50%) sta robota naučila prevoziti poligon, enemu učencu (25%) je skoraj uspelo, en učenec (25%) pa tega ni znal. Učenca, ki sta na to trditev odgovorila z "da", sta programirala v paru.

Element	Uporabljene funkcije
• Znak za obvezno smet	• Zavijanje
• Prehod za pešče	• Ustavljanje • Čakanje
• Ovira na cestišču	• Zaznavanje ovir • Vklop in izklop LED lučk • Zavijanje
• Semafor	• Pritisk na gumb • Naključna izbira
• Parkirišča	• Zavijanje • Vzvratna vožnja • Vklop in izklop LED lučk • Ustvarjanje zvoka

Tabela 4.1: Funkcije, katere morajo učenci robota "naučiti" pri določenem elementu

5 INTERPRETACIJA REZULTATOV

Robot TiddlyBot je enostaven in bi ga učenci po priloženih navodilih znali sestaviti sami, večina učencev pa si je zapomnila tudi vse njegove funkcije. Dovolj enostaven je tudi postopek povezave z robotovim brezžičnim omrežjem ter uporaba programskega okolja Blockly, prek katerega robota lahko upravljamo, saj večini učencev to ni predstavljalo težav.

Učenci so dobili dovolj jasna in natančna navodila ter skico poligona, saj so razumeli, kako mora robot prevoziti poligon in kakšne omejitve morajo upoštevati ter si vožnjo pred začetkom programiranja tudi vizualno predstavljali.

Polovica učencev je znala robota naučiti vsak korak, vendar noben učenec ni uspešno rešil korak že v prvem poizkusu. Večina učencev je na koncu uspela oziroma skoraj uspela robota naučiti prevoziti poligon.

Učenca, ki sta programirala v paru, sta bila bolj uspešna kot ostala dva, ki sta programirala samostojno.

6 ZAKLJUČEK IN NADALJNJE DELO

Teoriji konstruktivizma in kognitivizma poudarjata, da je znanje najučitkovitejše in dolgotrajno, če ga učenci pridobijo sami s svojimi aktivnostmi. V procesu učenja pa ima pomembno vlogo učitelj, ki z različnimi motivacijskimi spodbudami pripomore k kvalitetnejšemu učenju programiranja in motiviranju učencev. Pristopi, ki so se pri izvedbi delavnice učenja programiranja za osnovnošolske otroke izkazali kot motivacijski so uporaba vizualizacijskih programskih orodij, učenje z didaktičnimi igrami, sodelovalno učenje oziroma programiranje v paru in robotika. Učenje je potekalo skozi igro in z uporabo robota Tiddlybot, ki ga upravljamo v vizualnem programskem okolju Blockly. Delavnico smo izvedli dvakrat in pri drugi izvedbi kot motivacijski pristop dodali programiranje v paru. Izkazalo se je, da sta bila učenca, ki sta programirala v paru uspešnejša kot učenca, ki sta programirala samostojno.

Delavnico smo izvedli le za manjšo skupino učencev, za natančnejše in zanesljivejše ugotovitve pa bi bila potrebna izvedba delavnice za večjo skupino učencev. Zato v prihodnje načrtujemo izvedbo delavnice za učence druge triade v sklopu neobveznega izbirnega predmeta Računalništvo na eni izmed osnovnih šol.

Literatura

[1] K. Ala-Mutka. Problems in learning and teaching programming. (2004). *Edge: Development Group for Programming Education.* Dosegljivo: `www.cs.tut.fi/~edge/literature_study.pdf`. [Dostopano: 02.06.2016]. *(Citirano na strani 2.)*

[2] E. Ackermann. Piaget's Constructivism, Papert's Constructionism: What's the difference? . (2013). *Mit Media Lab: Massachusetts Institute of technology.* Dosegljivo: `http://learning.media.mit.edu/content/publications/EA.Piaget%20_%20Papert.pdf`. [Dostopano: 04.06.2016]. *(Citirano na straneh 6 in 7.)*

[3] M. Ben-Ari. Constructivism in Computer Science Education. (2001). *Uppsala University Sweden.* Dosegljivo: `https://it.uu.se/edu/course/homepage/cosulearning/st12/reading/Moti-Ben-Ari-jcmst.pdf`. [Dostopano: 04.06.2016]. *(Citirano na strani 6.)*

[4] J. Bennedse, M. Caspersen in M. Kölling, *Reflections on the Teaching of Programming: Methods and Implementations*, Berlin: Springer, 2008. *(Citirano na straneh 2, 3 in 10.)*

[5] S. Booth, *Learning to program: a phenomenographic perspective*, Acta Universitatis Gothoburgensis, 1992. *(Citirano na strani 3.)*

[6] D. Blaznik. Učiteljevi pristopi pri poučevanju matematike in njihov vpliv na učno motivacijo. (2013). *PeFprints: institucionalni repozitorij znanstvenoraziskovalnih, umetniških in visokošolskih del Pedagoške fakultete Univerze v Ljubljani.* Dosegljivo: `http://pefprints.pef.uni-lj.si/1525/1/diploma_blaznik_dunja.pdf`. [Dostopano: 07.06.2016]. *(Citirano na strani 9.)*

[7] P. Curzon et. al. Developing computational thinking in the classroom: a framework. (2014). *University of Southampton Institutional Research Repository ePrints Soton.* Dosegljivo: `http://eprints.soton.ac.uk/369594/10/DevelopingComputationalThinkingInTheClassroomaFramework.pdf`. [Dostopano: 06.06.2016]. *(Citirano na straneh 7 in 8.)*

[8] W. C. Chang in Y. M. Chou, "Introductory C Programming Language Learning with Game-Based Digital Learning" v: *Advances in Web Based Learning - ICWL 2008*, 7th International Conference, Jinhua, China, Frederick J. et al. Berlin: Springer, 2008, str: 221-231. *(Citirano na straneh 9 in 15.)*

[9] G. Futschek in J. Moschitz. Developing Algorithmic Thinking by Inventing and Playing Algorithms. Developing Algorithmic Thinking by Inventing and Playing Algorithms. *Publikationsdatenbank der Technischen Universität Wien.* Dosegljivo: `http://publik.tuwien.ac.at/files/PubDat_187461.pdf`. [Dostopano: 09.06.2016]. *(Citirano na straneh 7 in 8.)*

[10] R. Garris, R. Ahlers in J. E. Driskell. Games, Motivation, and Learning: A Research and Practice Model. (2002). *SAGE Publications.* Dosegljivo: `http://sag.sagepub.com/content/33/4/441.full.pdf+html`. [Dostopano: 03.06.2016]. *(Citirano na strani 10.)*

[11] Gimnjazija Nova Gorica. INFORMATIKA: Predvideni načini ocenjevanja znanja, kriteriji in minimalni standard znanja. (2015). *Gimnazija Nova Gorica.* Dosegljivo: `http://www.gimng.si/files/2015/11/INF-2015.pdf`. [Dostopano: 20.08.2016]. *(Citirano na straneh 5 in 11.)*

[12] M. Juriševič. Motiviranje učencev v šoli. (2012). *PeFprints: institucionalni repozitorij znanstvenoraziskovalnih, umetniških in visokošolskih del Pedagoške fakultete Univerze v Ljubljani.* Dosegljivo: `https://www.pef.uni-lj.si/fileadmin/Datoteke/CRSN/branje/Motiviranje_u%C4%8Dencev_v_%C5%A1oli_u%C4%8Dbenik__2012_.pdf`. [Dostopano: 07.06.2016]. *(Citirano na strani 1.)*

[13] K. Kolar. Novi pristopi pri poučevanju programiranja. (2012). *PeFprints: institucionalni repozitorij znanstvenoraziskovalnih, umetniških in visokošolskih del Pedagoške fakultete Univerze v Ljubljani.* Dosegljivo: `http://pefprints.pef.uni-lj.si/997/1/KKOLAR_diploma.pdf`. [Dostopano: 07.06.2016]. *(Citirano na straneh 2, 3, 5 in 10.)*

[14] J. Kramer. Is abstraction the key to computing?. (2007). *UCI: The Donald Bren School of Information and Computer Sciences.* Dosegljivo: `http://www.ics.uci.edu/~andre/informatics223s2009/kramer.pdf`. [Dostopano: 08.06.2016]. *(Citirano na strani 3.)*

[15] R. Krajnc et. al. Učni načrt. Program osnovna šola. Računalništvo: neobvezni izbirni predmet. (2013). *Republika Slovenija: Ministrstvo za izobraževanje, znanost in šport.* Dosegljivo: `http://www.mizs.gov.si/fileadmin/`

mizs.gov.si/pageuploads/podrocje/os/devetletka/program_razsirjeni/Racunalnistvo_izbirni_neobvezni.pdf. [Dostopano: 06.04.2016]. *(Citirano na straneh 3, 4 in 10.)*

[16] A. Krapez et. al. Razvoj predmeta računalništvo in informatika v osnovni in srednji šoli. (2009). *Slovensko društvo Informatika*. Dosegljivo: http://www.drustvo-informatika.si/fileadmin/dsi2001/sekcija_e/krapez_rajkovic_batagelj_wechtersbach.doc. [Dostopano: 06.04.2016]. *(Citirano na straneh 3, 4 in 5.)*

[17] P. Križman. Projekti z Raspberry Pi za zgodnje učenje programiranja. (2015). *PeFprints: institucionalni repozitorij znanstvenoraziskovalnih, umetniških in visokošolskih del Pedagoške fakultete Univerze v Ljubljani*. Dosegljivo: http://pefprints.pef.uni-lj.si/3089/1/DiplomskoDelo_Krizman_Peter.pdf. [Dostopano: 10.06.2016]. *(Citirano na straneh 8 in 9.)*

[18] E. Lahtinen, K. Ala-Mutka in H. M. Järvinen. A study of the difficulties of novice programmers. (2005). *ACM Digital Library*. Dosegljivo: https://dl.acm.org/purchase.cfm?id=1067453&CFID=828827370&CFTOKEN=24150029. [Dostopano: 09.06.2016]. *(Citirano na strani 15.)*

[19] T. Lončarič. Učni načrt. Računalništvo: gimnazija tehniška gimnazija, izbirni strokovni maturitetni predmet. (2010). *Republika Slovenija: Ministrstvo za izobraževanje, znanost in šport*. Dosegljivo: http://eportal.mss.edus.si/msswww/programi2013/programi/media/pdf/un_gimnazija/tehniska-gimnazija/UN_Racunalnistvo.pdf. [Dostopano: 10.07.2016]. *(Citirano na straneh 5 in 11.)*

[20] A. Luštek. Priprava problemsko oblikovanih učnih gradiv za pouk pri predmetu Računalništvo v drugem triletju. (2015). *PeFprints: institucionalni repozitorij znanstvenoraziskovalnih, umetniških in visokošolskih del Pedagoške fakultete Univerze v Ljubljani*. Dosegljivo: http://pefprints.pef.uni-lj.si/3056/1/Diploma_AnjaLustek_CD.pdf. [Dostopano: 10.06.2016]. *(Citirano na strani 7.)*

[21] L. Ma et. al. Improving the mental models held by novice programmers using cognitive conflict and jeliot visualisations. (2009). *ACM Digital Library*. Dosegljivo: https://dl.acm.org/purchase.cfm?id=1562931&CFID=828827370&CFTOKEN=24150029. [Dostopano: 06.08.2016]. *(Citirano na strani 3.)*

[22] R. Mayer. (2013). *Teaching and Learning Computer Programming: Multiple Research Perspectives [Online]*. Routledge. Dosegljivo: Google, https://books.google.si/books?hl=sl&lr=&id=tsNcAgAAQBAJ&oi=

fnd&pg=PP1&dq=programming+teaching+in+school&ots=RpMzgXlwZ-&sig=UnVqhD2E25zsiizb8ssn93Tek5A&redir_esc=y#v=onepage&q=history&f=false. [Dostopano 18. 7. 2016]. *(Citirano na strani 3.)*

[23] M. Močnik. Učenje programiranja: Lov na zaklad. (2015). *PeFprints: institucionalni repozitorij znanstvenoraziskovalnih, umetniških in visokošolskih del Pedagoške fakultete Univerze v Ljubljani*. Dosegljivo: http://pefprints.pef.uni-lj.si/3058/1/diploma_kon%C4%8Dna.pdf. [Dostopano: 14.08.2016]. *(Citirano na strani 9.)*

[24] I. Mrak-Merhar et. al. Didaktične igre in druge dinamične metode. (2013). *Republika Slovenija: Ministrstvo za izobraževanje, znanost in šport*. Dosegljivo: http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/odrasli/Gradiva_ESS/NIO/NIO_2DMC_Didakticne.pdf. [Dostopano: 08.08.2016]. *(Citirano na strani 10.)*

[25] M. Petre in B. Price. Using robotics to motivate "back door"learning. Using robotics to motivate "back door"learning. *The Open University: Faculty of Science, Technology, Engineering and Mathematics*. Dosegljivo: http://mcs.open.ac.uk/bp5/papers/2004-eit/2004-EIT-Robotics-Backdoor.pdf. [Dostopano: 09.08.2016]. *(Citirano na straneh 10 in 16.)*

[26] K. Stoeckelmayr, M. Tesar in A. Hofmann. Kindergarten Children Programming Robots: A First Attempt. (2011). *Robotchallenge ORG*. Dosegljivo: http://www.robotchallenge.org/fileadmin/user_upload/_temp_/RiE/Proceedings/51.pdf. [Dostopano: 25.07.2016]. *(Citirano na strani 16.)*

[27] B. Stopar. Aktivnosti za razvoj algoritmičnega mišljenja med osnovnošolci. (2013). *PeFprints: institucionalni repozitorij znanstvenoraziskovalnih, umetniških in visokošolskih del Pedagoške fakultete Univerze v Ljubljani*. Dosegljivo: http://pefprints.pef.uni-lj.si/1793/1/Diplomsko_delo_StoparB_pdf.pdf. [Dostopano: 25.07.2016]. *(Citirano na strani 7.)*

[28] A. Woolfolk, *Pedagoška psihologija*, Educy, 2002. *(Citirano na straneh 6 in 8.)*

Ostali viri

[29] Bookes Oxford University[Online] Dosegljivo: http://bejlt.brookes.ac.uk/paper/reflections-on-the-evolution-of-the-teaching-of-programming-to-unde [Dostopano: 23.08.2016] *(Citirano na straneh 2 in 3.)*

[30] California State University[Online] Dosegljivo: http://web.csulb.edu/~murdock/histofcs.html [Dostopano: 23.08.2016] *(Citirano na strani 2.)*

[31] Fakulteta za matematiko, naravoslovje in informacijske tehnologije, Univerza na Primorskem[Online] Dosegljivo: http://orada.famnit.upr.si/sl/ [Dostopano: 06.06.2016] *(Citirano na strani 20.)*

[32] Fakulteta za računalništvo in informatiko, Univerza v Ljubljani[Online] Dosegljivo: http://fri.uni-lj.si/si/izobrazevanje/ [Dostopano: 06.08.2016] *(Citirano na strani 13.)*

[33] TiddlyBot[Online] Dosegljivo: http://www.pibot.org/tiddlybot/ [Dostopano: 20.04.2016] *(Citirano na straneh 1, 15 in 17.)*

[34] Wikipedia: the free encyclopedia[Online] Dosegljivo: http://en.wikipedia.org/wiki/Blockly/ [Dostopano: 08.04.2016] *(Citirano na strani 15.)*

[35] Wikipedia: the free encyclopedia[Online] Dosegljivo: https://sl.wikipedia.org/wiki/Jean_Piaget [Dostopano: 08.08.2016] *(Citirano na strani 5.)*

[36] Wikipedia: the free encyclopedia[Online] Dosegljivo: https://sl.wikipedia.org/wiki/Problem-based_learning [Dostopano: 08.08.2016] *(Citirano na strani 8.)*

[37] Wikipedia: the free encyclopedia[Online] Dosegljivo: https://en.wikipedia.org/wiki/Raspberry_Pi [Dostopano: 06.08.2016] *(Citirano na straneh 1 in 17.)*

[38] Wikipedia: the free encyclopedia[Online] Dosegljivo: `http://en.wikipedia.org/wiki/Scratch_(programming_language)/` [Dostopano: 06.05.2016] *(Citirano na strani 20.)*

Priloge

Priloga A: Anketa

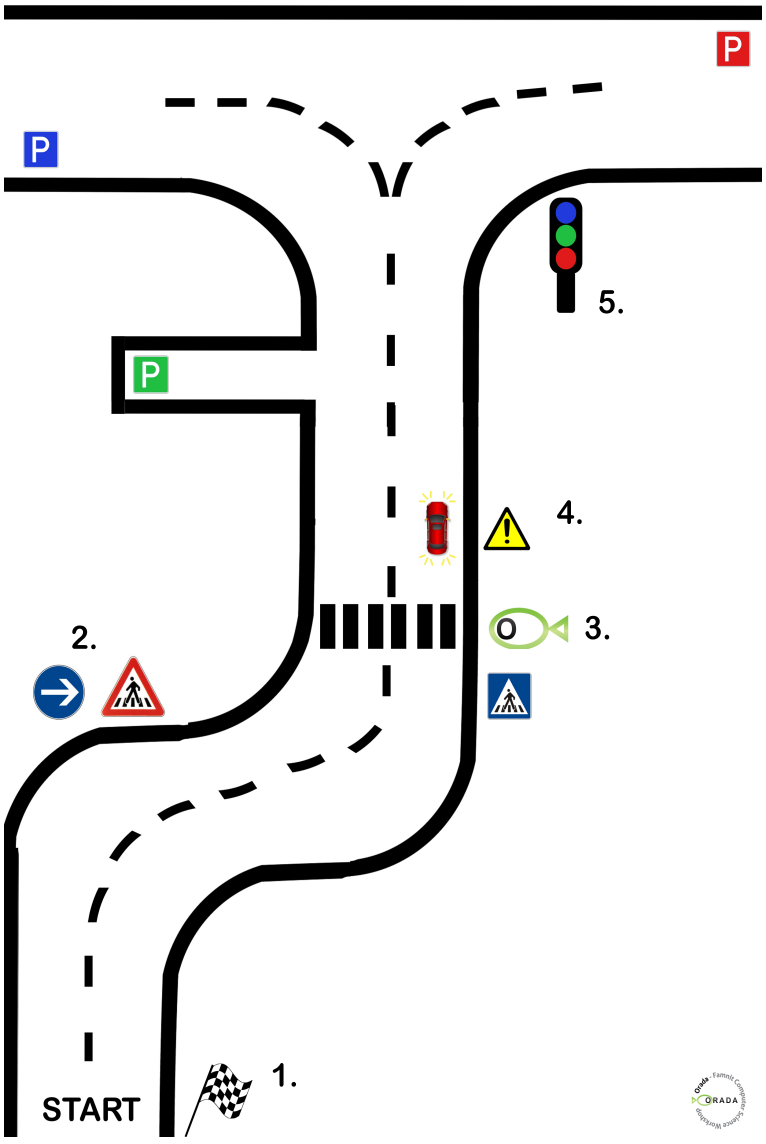
1. Spol: M Ž
2. Starost:
3. Šola, ki jo trenutno obiskuješ:
4. Ali si že kdaj programiral robota? DA NE
5. Ali si že kdaj uporabljal vizualni urejevalnik Blockly? DA NE

TRDITVE:

☺ = DA, 😐 = MOGOČE/SKORAJ, ☹ = NE;

Sam bi znal sestaviti robota po priloženih navodilih.	☺	😐	☹
Zapomnil sem si, kaj robot TiddlyBot vse zmore.	☺	😐	☹
Znam se povezati na brezžično povezavo TiddlyBot in v brskalniku odpreti uporabniški vmesnik, prek katerega lahko robota nekaj naučimo.	☺	😐	☹
Znam uporabljati vizualni urejevalnik Blockly.	☺	😐	☹
Razumel sem navodila in vedel sem kaj moram robota "naučiti".	☺	😐	☹
Pred začetkom programiranja sem si predstavljal, kako mora robot prevoziti poligon.	☺	😐	☹
Robota sem znal naučiti vsak korak, ki je bil opisan v navodilih.	☺	😐	☹
Vsak korak, ki je bil opisan v navodilih sem robota uspešno naučil že v prvem poizkusu.	☺	😐	☹
Na koncu delavnice sem robota naučil prevoziti poligon po priloženih navodilih.	☺	😐	☹

Priloga B: Skica poligona



Priloga C: Robot Tiddlybot