Arhitektura sistema za izvedbo tekmovanja v mobilni robotiki

Nejc Ilc, Jakob Maležič, Matija Rezar, Davor Sluga

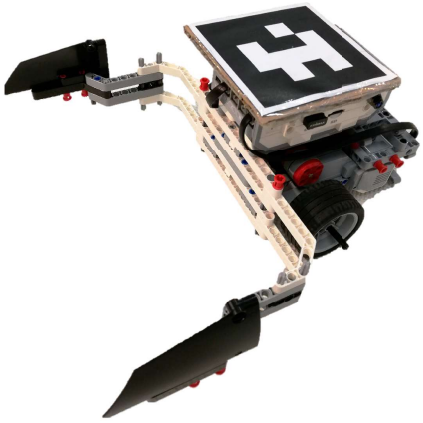
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko
E-pošta: nejc.ilc@fri.uni-lj.si

System architecture for running mobile robotics competition

Robo liga FRI is a competition in mobile robotics based on the LEGO Mindstorms kit. It is organised by the Faculty of Computer and Information Science at the University of Ljubljana. In the 15 years of its existence, the competition has proven to be a valuable and fun way to build professional and social skills. It offers students the opportunity to acquire new knowledge in programming, process control, computer communication, and machine learning. It also encourages the development of collaboration and communication skills through teamwork. In this article, we briefly review past challenges and describe the most recent one in more detail. We then present an architecture of a new system for running the competition that includes efficient object tracking, a simple interface for communicating with the robots, and a web application for game management. The presented system enables the design of more complex and, thus, more exciting competition tasks while making it easier for newcomers. Moreover, the developed software is free and open source. We have used it successfully in the last two seasons of the Robo liga FRI.

1 Uvod

Tekmovanja v robotiki predstavljajo koristen vzvod za vzpodbujanje učenja na vseh ravneh izobraževanja in izboljšanje kognitivnih ter socialnih sposobnosti [1]. Poučevanje skozi robotiko je postalo zelo dostopno, predvsem z izdajo kompletov LEGO iz serije Mindstorms, ki omogočajo enostavno gradnjo robotov, nudijo širok nabor merilnih in izvršilnih členov, podporo za različne programske jezike in zmogljiva razvojna okolja [2]. Primer mobilnega robota, ki je sestavljen iz takega kompleta, vidimo na sliki 1. Zaradi tega se je tako v Sloveniji kot tudi po svetu uveljavilo kar nekaj robotskih tekmovanj, ki temeljijo na LEGO Mindstorms in ciljajo na osnovnošolce, dijake in študente. Med najbolj znanimi mednarodnimi sta First Lego League [3] in World Robot Olympiad [4]. Med aktualnimi tekmovanji v Sloveniji sta ROBObum [5], ki ga za osnovnošolce in dijake organizira Fakulteta za elektrotehniko in računalništvo Univerze v Mariboru, in Lego Masters [6], ki ga za dijake in študente organizira Fakulteta za elektrotehniko Univerze v Ljubljani.



Slika 1: Tekmovalni robot, sestavljen iz kompleta kock *LEGO Mindstorms Education EV3 Core*. Nanj je pritrjena značka ArUco.

Tekmovanja v mobilni robotiki organizira tudi Fakulteta za računalništvo in informatiko Univerze v Ljubljani v okviru Robo Lige FRI [7]. Tekmovanje je primarno osredotočeno na študente Univerze v Ljubljani, z nekaj izvedbami, ki so naslavljale tudi srednješolce. Začetki Robo Lige FRI segajo v leto 2007, ko se je odvilo prvo tekmovanje. Dober odziv študentov je botroval organizacij nadaljnjih tekmovanj, ki so se osredotočala na reševanje raznoraznih robotskih izzivov.

Do sedaj je bilo izvedenih 9 tekmovanj Robo Lige FRI, ki se jih je udeležilo skupno 387 tekmovalcev v 142 ekipah iz različnih fakultet in tudi nekaj srednjih šol. Vendar pa precej ekip, okoli 40 %, tekmovanja ni zaključilo, saj je tekmovalcem zaradi drugih študijskih obveznosti pogosto zmanjkalo časa za razvoj programske opreme robota. Na nekaj tekmovanjih je bil uporabljen sistem s kamero za določanje položaja robotov, ki je omogočal bolj napredne izzive, vendar je deloval nezanesljivo in je tekmovalcem povzročal precej preglavic.

Z razvojem popolnoma novega zmogljivega sistema za določanje položaja robotov preko kamere in učinkovitim vmesnikom za komunikacijo z roboti na poligonu želimo zmanjšati količino časa, ki jo morajo vložiti udeleženci, da razvijejo svojo rešitev izziva. To pomembno vpliva na samo uspešnost ekip in jim hkrati omogoči, da več časa posvetijo zanimivejšim problemom krmiljenja

robotov brez da bi se ukvarjali s zapletenimi tehničnimi podrobnostmi.

V nadaljevanju najprej v poglavju 2 predstavimo zadnji tekmovalni izziv „Čebelnjak”, v katerem so se študenti pomerili leta 2020. V poglavju 3 opišemo razvit sistem za izvedbo tekmovanj. V poglavju 4 pa so zbrane zaključne ugotovitve in predstavljena izhodišča za nadaljnje delo.

2 Tekmovalni izziv „Čebelnjak”

Tekmovalne ekipe so morale sestaviti svojega avtonomnega robota – čebelarja. Roboti so se premikali po poligonu, ki je predstavljal sadovnjake, na njem pa so se nahajale lesene kocke, ki so ponazarjale zdrave ali okužene panje. Na poligonu sta se nahajali tudi dve skladišči, kamor so morali roboti pripeljati panje. Šlo je torej za dvoboj med dvema robotoma, ki sta želela v omejenem času v svoje skladišče pripeljati čim več zdravih panjev in poskrbeti, da je v njem čim manj okuženih. Robota sta za navigacijo uporabljala podatke, ki sta jih je preko brezžičnega omrežja pridobila s strežnika. Strežnik je s pomočjo kamere spremljal položaj robotov in panjev.

2.1 Tekmovalni poligon

Poligon, prikazan na sliki 2, je bil sestavljen iz penastih plošč, ki jih je obdajala ograja, znotraj katere so se lahko premikali roboti. Poligon je bil razdeljen v več različnih con: dve coni, ki sta predstavljali skladišči in tri cone, ki so predstavljale sadovnjake z različno bogato pašo.

Na levi strani sta se nahajali dve modri plošči, ki sta predstavljali skladišče modre ekipe. Skladišče je obdajal eksotičen (glej poglavje 2.5) sadovnjak rdeče ekipe, ki je bil hkrati tudi domači sadovnjak modre ekipe. Na sredini je bil sadovnjak skupen obema ekipama. Podobno sta se na desni strani nahajali dve rdeči plošči, ki sta predstavljali skladišče rdeče ekipe. Skladišče je obdajal eksotični sadovnjak modre ekipe, ki pa je bil hkrati tudi domači sadovnjak rdeče ekipe.

2.2 Značke ArUco

Za hitro in robustno določanje položaja vseh objektov na poligonu preko kamere smo uporabili značke ArUco [8]; primer take značke je viden na zgornji strani robota na sliki 1. Značke imajo črno obrobo in črno-bel vzorec na sredini, po katerem značke ločimo med sabo ter določimo njihovo smer.

2.3 Panji

Na poligonu so bile postavljene lesene kocke velikosti 10 cm $\times$ 10 cm $\times$ 8 cm z značko ArUco na vrhu. Kocke so predstavljale čebelje panje, ki so jih roboti morali spraviti v svoje skladišče. Z dvema barvama smo označili vrsto panja: zelena za „zdrave” panje, in rjava za „okužene” panje. Zdravi panji, ki so jih roboti uspešno pripeljali v svoje skladišče, so ekipi prinesli točke. Ti panji so bili odstranjeni s poligona, ko je sistem zabeležil točke. Rjavi panji, ki so se nahajali v skladišču ob koncu igre, pa so ekipi odvzeli točke. Rjavi panji niso bili odstranjeni s poligona, kar je omogočalo njihovo premikanje do konca tekme in so jih roboti tako lahko odstranili iz svojega skladišča.

2.4 Robot

Robot je moral biti v celoti sestavljen iz kock kompleta LEGO Mindstorms EV3 in je moral na začetku tekme meriti največ 30 cm $\times$ 30 cm $\times$ 30 cm. Na vrhu robota je morala biti jasno vidno značka ArUco, preko katere je sistem beležil položaj robota.

2.5 Tekma in točkovanje

Cilj robotov je bil, da naberejo čim več točk in s tem prinesejo zmago svoji ekipi. Vsak dvoboj je trajal največ 3 minute; lahko se je zaključil prej, če se robota nista več premikala.

Vsak zdrav panj, ki ga je robot uspešno pripeljal v svoje skladišče, je ekipi prinesel točke. Število točk pa je bilo odvisno od tega, kje se je panj nahajal, oziroma, kje je med tekmo potoval. Panj, ki se je nahajal v domačem sadovnjaku, je bil vreden 1 točko. Če se je nahajal v skupnem sadovnjaku, je bil vreden 2 točki. Če se je nahajal v eksotičnem sadovnjaku, pa je bil vreden 3 točke. Robot je lahko panj tudi odpeljal v bolj oddaljen sadovnjak in ga s tem obogatil, kar mu je prineslo več točk. Za vsak okuženi panj, ki se je ob koncu tekme nahajal v njihovem skladišču, so tekmovalci izgubili 2 točki, ne glede na to, kje se je panj pred tem nahajal.

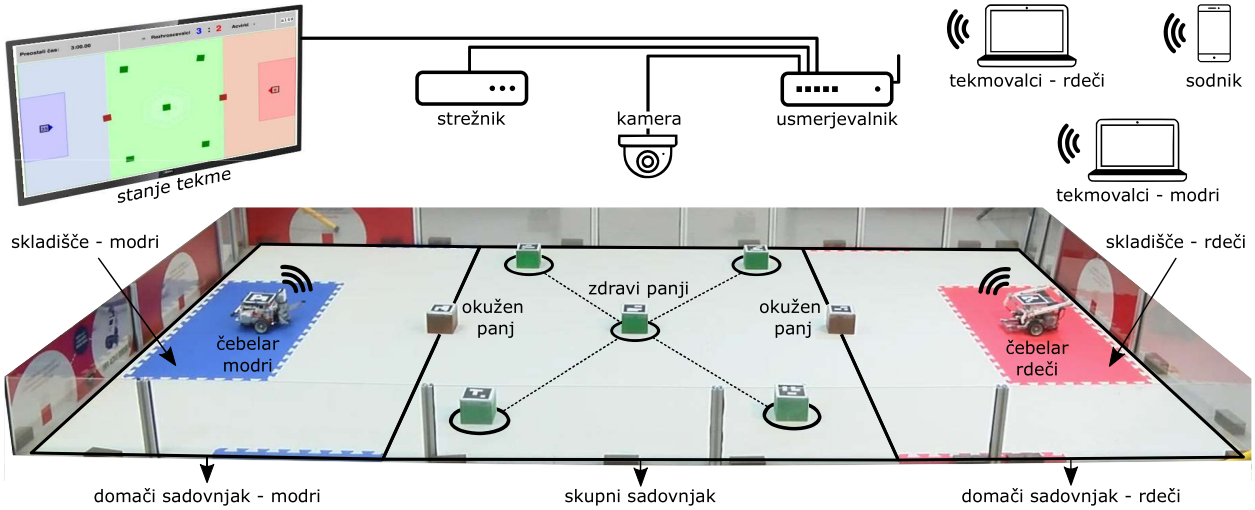
3 Arhitektura sistema za izvedbo tekmovanja

Sistem za izvedbo tekmovanja smo razvili v programskem jeziku Python in je odprtokoden ter prosto dostopen. Sestavljen je iz sledilnika, igralnega strežnika in programa za prikaz in nadzor tekme [9].

3.1 Sledilnik

Sledilnik je prvi del zalednega sistema. Njegova naloga je sledenje objektom na poligonu. Za to uporablja kamero, ki je nameščena nad poligonom, in programsko knjižnico OpenCV [10] v kombinaciji z značkami ArUco. Nastavitve sledilnika pripravimo s posebnim programom. Z njim označimo polja, ki jih za tekmo potrebujemo. Polja, ki jih želimo označiti, vnaprej naštejemo v nastavitveni datoteki.

Program najprej prebere podatke iz nastavitvene datoteke, ki jo je ustvaril program za pripravo sledilnika. Potem kreira sporočilno vrsto, v katero zapisuje položaj objektov, ki jih vidi kamera. Program nato začne svojo glavno zanko, kjer pridobi barvno sliko in jo za lažje prepoznavo značk ArUco pretvori v sivine. Slika je zaradi leče popačena, zato jo izravna in s pomočjo značk ArUco zazna položaj objektov. V primeru, da kateri od objektov ni več viden, z uporabo Kalmanovega filtra [11] predvidi njegovo trajektorijo in trenutni položaj. Če objekt ni viden dalj časa, program privzame, da objekta ni več na poligonu. Na koncu zanke program vse podatke o stanju na poligonu shrani v skupen objekt in ga zapiše v vrsto. Tako so podatki časovno urejeni in pripravljeni za branje ter obdelovanje.



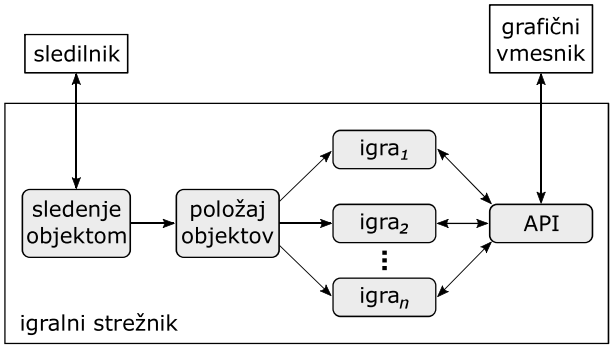
Slika 2: Arhitektura sistema za tekmovanje in prikaz glavnih elementov tekmovalnega izziva „Čebelnjak”.

3.2 Igralni strežnik

Igralni strežnik skrbi za potek celotne tekme. Pridobiva podatke iz sledilnika, jih obdeluje, dodeljuje tekmovalne točke in izpostavlja končne točke za upravljanje ter pridobivanje podatkov o tekmi.

Za asinhrono izvajanje programa smo uporabili knjižnico `Gevent` [12]. Ta nam omogoča, da se deli programa izvajajo sočasno, vendar ne vzporedno. To si lahko predstavljamo kot vrsto z moduli, ki se pričnejo izvajati ob sprožitvi dogodka. `Gevent` vedno omogoči izvajanje prvemu modulu v vrsti, ki je že pripravljen oziroma ne čaka več na sprožitev dogodka.

Igralni strežnik za svoje delovanje uporablja tri asinhrone module: *sledenje objektom*, *položaj objektov* in *igra*. Njihovo logično povezanost prikazuje slika 3. Vsak posamezen modul vsebuje atribut, ki predstavlja dogodek. Dogodek deluje kot zastavica: ko modul izvaja svojo nalogo, zastavico odstrani; ko pa modul opravi svojo nalogo, zastavico nastavi.



Slika 3: Arhitektura igralnega strežnika.

Najprej se izvede modul *sledenje objektom*. Ta skrbi, da proces sledilnika, ki je opisan poglavju 3.1, ves čas teče in ga v primeru nenadne zaustavitve ponovno požene. Na igralnem strežniku obstaja samo en primerek modula *sledenje objektom*. Ko konča, nastavi svojo zastavico in za kratek čas zaspi.

To zastavico spremlja modul *položaj objektov*. Takoj za tem, ko je zastavica sledilnika nastavljena, modul *položaj objektov* odstrani svojo zastavico, prebere podatke, ki jih je pridobil modul *sledenje objektom*, jih pretvori v podatkovno strukturo, ki predstavlja položaj značk ArUco in nastavi svojo zastavico ter zaspi za kratek čas. Na igralnem strežniku obstaja prav tako samo en primerek modula *položaj objektov*.

Modul *igra* prebere pripravljen objekt in ga uporabi za izračun rezultata in za nudenje podatkov preko končnih točk programskega vmesnika (ang. *application programming interface* - API). Obstaja lahko več primerkov modula *igra*, kar omogoča izvajanje več različnih tekem hkrati. To olajša testiranje, ko je na poligonu več robotov naenkrat, saj si ekipi lahko razdelita poligon na pol in razvijata svojo rešitev neodvisno druga od druge. Ker noben modul ne čaka na modul igre, se lahko ponovno izvede celoten krog.

Kot zadnji se zažene programski vmesnik, ki izpostavlja končne točke za branje podatkov in upravljanje tekme. Za definicijo končnih točk smo uporabili programski arhitekturni slog REST (ang. *REpresentational State Transfer*, prenos predstavitve stanja). Seznam razpoložljivih končnih točk je prikazan v tabeli 1.

3.3 Prikaz in nadzor

Za namen prikazovanja in nadzor tekmovanja smo ustvarili `monitor` – spletno aplikacijo napisano v jeziku Dart. Aplikacija uporablja osnovne elemente HTMLja za prikaz podatkov o tekmi, kot je rezultat in preostali čas, ter element `<canvas>` za izris igrišča.

Ker strežnik na končni točki `/game/{id}` ponuja celostne in merodajne podatke o igri, hkrati pa so ti podatki preprosti, je zasnova monitorja enostavna. Ob prejemu podatkov se celotno podatkovno strukturo igre zamenja z objekti izzaporedjenimi (ang. *deserialized*) iz prejete JSON strukture. Nato se izvede izris na novo prejetega stanja.

Zahtevki GET

/game	Vrne seznam identifikacijskih številk iger.
/game/{id}	Vrne stanje igre z identifikacijsko številko *id*.
/teams	Vrne podatke o ekipah.

Zahtevki POST

/game/{id}/score	Nastavi rezultat igre *id*.
/game/{id}/teams	Nastavi ekipi, podani v zahtevku, za igro *id*.
/game/{id}/time	Nastavi čas igre *id*.

Zahtevki PUT

/game	Ustvari novo igro za ekipi, ki sta podani v zahtevku.
/game/{id}/start	Zažene igro *id*.
/game/{id}/stop	Zaključi igro *id*.
/game/{id}/pause	Začasno zaustavi igro *id*.

Tabela 1: Seznam končnih točk, ki jih izpostavlja programski vmesnik in omogočajo spremljanje in nastavljanje stanja tekme.

Za izris aplikacija uporablja princip grafa scene (ang. *scene graph*). S pomočjo knjižnice `json_annotation` JSON predstavitev stanja igre pretvorimo v enakovredno strukturo objektov, kjer vsak tak objekt implementira metodo `_draw()` po principu račjega tipiziranja (ang. *duck typing*). Ob klicu za izris se objekti rekurzivno izrišejo. Pri tem si podajajo kontekst izrisa, s čimer zagotovijo, da se transformacije, torej premiki in rotacije starševskih elementov, ohranjajo. To predvsem omogoča, da se orientacija prikaza lahko hitro in enostavno usklajuje s fizičnim igriščem; glede na to, kje je prikazovalnik (televizor ali projektor) relativno na igrišče.

Glavni del aplikacije predstavljata dve asinhroni funkciji, oziroma korutini: `fetch()` in `refresh()`. Prva s strežnika zahteva podatke o igri, druga podatke prebere in izvede izris. Ker je monitor navadno na isti mreži ali celo na istem računalniku kot strežnik, je zamik za pridobivanje podatkov zanemarljiv. Prav tako je zanemarljivo hiter izris. To pomeni, da lahko izrisujemo hitreje kot s frekvenco zaslona, kar je potratno. V ta namen smo v `fetch()` pred izvedbo klica na strežnik dodali še časovnik, ki privzeto vrine 10 ms postanka. To nam da teoretično frekvenco 100 Hz, kar je več kot dovolj za večino zaslonov.

4 Zaključek

Tekmovanje v mobilni robotiki Robo liga FRI se je skozi 15 let svojega obstoja razvilo v prepoznavno in zabavno obogatitev študijskega procesa. Študentom ponuja pridobitev novih inženirskih znanj s področja programiranja, vodenja in regulacije procesov, računalniških komunikacij in strojnega učenja. Prav tako s skupinskim načinom dela močno spodbuja razvijanje veščin sodelovanja in sporazumevanja.

V članku smo predstavili kratek pregled tekmovanja Robo Liga FRI in podrobneje opisali zadnje tekmovanje z naslovom „Čebelnjak". Da bi tekmovalcem olajšali začetno stopnjo reševanja izziva in tako preprečili prevelik osip pred zaključkom tekmovanja, smo razvili nov sistem za učinkovito sledenje objektom na poligonu in enostavno komunikacijo z roboti. Ta skupaj z upraviteljem in prikazovalnikom tekem tvori prilagodljivo celoto, ki omogoča snovanje kompleksnejših in s tem zanimivejših tekmovalnih nalog.

Razvit sistem za izvedbo tekmovanja smo uspešno uporabili za izvedbo zadnjih dveh sezon Robo lige FRI, v teku pa so tudi že priprave novih tekmovalnih izzivov. V bodoče se bomo osredotočili na izboljšave pedagoškega dela s tekmovalci – predvsem želimo zagotoviti več podpore v obliki vodenih tečajev in delavnic za hitrejše premagovanje začetniških težav.

5 Zahvale

Iskreno se zahvaljujemo so-organizatorjem Robo lige FRI v preteklih letih: Urošu Lotriču, Tomu Vodopivcu in Ratku Pilipoviću. Hvala vodstvu Fakultete za računalništvo in informatiko Univerze v Ljubljani za finančno in strokovno pomoč pri izvedbi tekmovanj. Predvsem pa velika zahvala vsem tekmovalcem – brez vas ni Robo lige FRI.

Literatura

[1] S. Evripidou, K. Georgiou, L. Doitsidis, A. A. Amanatiadis, Z. Zinonos in S. A. Chatzichristofis, "Educational Robotics: Platforms, Competitions and Expected Learning Outcomes", *IEEE Access*, let. 8, str. 219 534–219 562, 2020. DOI: 10.1109/ACCESS.2020.3042555.

[2] R. Grandi, R. Falconi in C. Melchiorri, "Robotic Competitions: Teaching Robotics and Real-Time Programming with LEGO Mindstorms", *IFAC Proceedings Volumes*, let. 47, št. 3, str. 10 598–10 603, 2014. DOI: 10.3182/20140824-6-ZA-1003.00222.

[3] *First Lego League*, https://www.firstlegoleague.org/.

[4] *World Robot Olympiad*, https://wro-association.org/.

[5] *ROBObum*, https://robobum.um.si/.

[6] *Lego Masters*, https://lego-masters.si/.

[7] *Robo Liga FRI*, https://www.fri.uni-lj.si/sl/robo-liga-fri.

[8] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas in M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion", *Pattern Recognition*, let. 47, št. 6, str. 2280–2292, 2014. DOI: 10.1016/j.patcog.2014.01.005.

[9] *Robo liga FRI - repozitorij kode*, https://github.com/RoboLiga.

[10] G. Bradski, "The OpenCV Library", *Dr. Dobb's Journal of Software Tools*, 2000.

[11] G. Welch in G. Bishop, "An Introduction to the Kalman Filter", *In Practice*, let. 7, št. 1, str. 1–16, 2006.

[12] *Gevent*, http://www.gevent.org/.