UNIVERZITET U BEOGRADU

FAKULTET ORGANIZACIONIH NAUKA

ZAVRŠNI RAD

RAZVOJ KONKURENTNOG PAMETNOG SISTEMA ZA PODRŠKU DRUŠTVENIM IGRAMA

Mentor

Prof. dr Božidar Radenković

Redovni profesor

Student

Danilo Paunović, 2017/0166

Beograd, 2023. godina

Apstrakt

Društvene igare su u poslednjih par godina doživele veliki porast u popularnosti što se može videti porastom vrednosti industrije koja sada vredi oko 2.2 milijarde dolara. Porast tog tržišta, kao i veća pristupačnost IoT-a u poslednjih par godina daju mogućnost društvenim igrama da nastave svoj razvoj primenom pametnih uređaja. Cilj ovog rada je kreiranje konkurentnog pametnog sistema koji će upotpuniti igračko iskustvo za igru Asocijacije, kao osnova za primenu pametnih uređaja za društvene igre. Korišćena metodologija za izradu rada je pregled literature o pametnim sistemima, konkurentnom programiranju kao i njihovoj primeni u razvoju društvenih igara; i razvoj sistem korišćenjem tehnologija Python, Rpi i Flask, sa izdradom dokumentacije. Glavni doprinos rada je kreirani sistem za podršku igri Asocijacije korišćenjem tehnologija pametnih sistema, koji daje mogućnosti boljeg korisničkog iskustva.

SADRŽAJ

1. UVOD

Društvene igre su jedna od osnovnih načina zabave kod ljudi. Svaka kultura poseduje svoje igre koje su se igrale od davnina. Postoji određen osećaj sreće kod ljudi kada znaju da je današnji dan namenjen za igranje društvenih igara sa prijateljima i/ili porodicom. Privlačnost društvenih igara se krije u njihovoj raznovrsnosti , bile one strateške, edukativne, igre izdajnika, RPG (*eng. Role playing game*), postoji za svakog po nešto. Predviđa se da će vrednost industrije društvenih igara do kraja 2027. godine iznositi oko 3.51 milijardi dolara. (Statista, 2023)

Sve je veća upotreba inteligentnih uređaja u svakodnevnom životu, počevši od mobilnih telefona. Procenjuje se da je prosečno vreme provedeno na telefonu svakog dana oko 3,5 sati. (Exploding Topics, 2023) Prednosti dobijene primenom pametnih uređaja u prikupljanju i deljenju informacija se ne smeju zanemariti. Inteligentni uređaji stalno pronalaze načine da se uključe u naš svakodnevni život pružajući neku vrstu poboljšanja u vidu energetske efikansosti, daljinskog praćenja, konotrole fizičkih sredstava i produktivnosti kroz veliki broj različitih aplikacija. (Anuradha & Tripathy, 2018)

Odatle se javlja mogućnost i potreba za primenom inteligentnih uređaja u razvoju pametnih sistema za podršku društvenim igrama. Poboljšavanje i upotpunjavanje igračkog iskustva kao i olakšavanje igranja društvenih igara su jedne od osnovnih prednosti njihove integracije sa pametnim uređajima. Pored toga što je moguće projektovati sisteme za postojeće igre, upotreba IoT-a otvara vrata za kreiranje novih igara u čijoj se srži nalazi pametni sistemi. Dobija se mogućnost da se društvene igre kao koncept razvijaju ponovo i da se ponovo zamisli šta je to i šta to može da bude društvena igra.

U 2. poglavlju opisani su neki od osnovnih primera pametnih sistema (pametni gradovi i pamentne kuće), kao i šta to čini jedan sistem pametnim. Na koji način su inteligentni uređaji našli svoje mesto u društvenim igrama prikazano je u 3. poglavlju. Tu su dati i neki osnovni primeri igara koje su uspešno implementirale pametne sisteme. Pre nego što se opiše sam projektni rad prikazane su tehnologije koje su korišćene za razvoj pametnog sistema, a rečeno je nešto više i o konkurentnom programiranju u poglavlju 4. Za kraj definisani su korisnički zahtevi, slučajevi korišćenja, a opisana je arhitektura i implementacija sistema u 5. poglavlju.

2. PAMETNA OKRUŽENJA

Kako Internet nastavlja svoj razvoj povezanost se ne završava sa ljudima. Dolazimo do mogućnosti da povezujemo uređaje koji dele informacije i preduzimaju određene akcije. Ovaj koncept se naziva Internet inteligentnih uređaja (eng. Internet of things - IoT) i predstavlja osnovu sledećeg Interneta „Web 3.0“ . (Mavromoustakis, Mastorakis, & Batalla, 2016)

Upotreba IoT uređaja u svokodnevnom životu donosi veliki broj poboljšanja u vidu energetske efikasnosti, daljinskog praćenja, konotrole fizičkih sredstava i produktivnosti kroz veliki broj različitih aplikacija. Usled sve veće digitalizacije i povezanosti uređaja uspostavljaju se mreže između mašina, ljudi i Interneta. Time dolazimo do novih sistema koji omogućavaju veću produktivnost, energetsku efikasnost i veću profitabilnost. (Anuradha & Tripathy, 2018)

Pametni uređaji prikupljaju informacije iz svoje okoline i donose odluke zasnovane na datim informacijama bez ikakve intervencije ljudi. (Anuradha & Tripathy, 2018). Uređaji koji su međusobno povezani čine sistem u kome primenom senzora prikupljaju podatke i međusobno komuniciraju. (Posey, 2022)

Postoje različite primene *IoT-a* u svrhu kreiranja pametnih okruženja: pametni gradovi, pametni saobraćaj, pametne kuće, pametna industrija, pametna poljoprivreda, pametna elektroenergetska mreža (*eng. Smart Grid*).

2.1 PAMETNI GRADOVI

Jedna od najpopularnijih sfera istraživanja primene *IoT-a* je upravo pronalaženje adekvatnog načina upotrebe pametnih uređaja kako bi se poboljšao kvalitet života u gradovima. (Hassan, Khan, & Madani, 2018)

Postoji veliki broj definicija pametnih gradova (Radenković, i drugi, 2017):

- Pametni gradovi su gradovi koji fizičku infrastrukturu povezuju sa IT infrastrukturom i poslovnom infrastrukturom, a sve u cilju korišćenja kolektivne inteligencije grada;

- Grad koji za cilj ima uticaja na ekonomski rast i visok kvalitet života, s pametnim upravljanjem prirodnim resursima ulaganjem u ljudski i društveni kapital i tradicionalnu i modernu komunikacionu infrastrukturu;
- Grad čija zajednica je spremna da uči, da se prilagođava i uvodi inovacije;
- Grad u kome gradske vlasti, preduzeća i građani koriste informacione tehnologije da osmisle i ojačaju ulogu zajednice u novoj ekonomiji usluga, da stvaraju nova radna mesta i poboljšaju kvalitet života.

Pametni gradovi se istorijski mogu opisati kroz njihovu urbanu zamisao da prioritizuju održavanje reda i efikasnosti, kao i podsticanje ekonomskog rasta i konkurentnosti kako na globalnom tako i na regionalnom tržištu i to kroz tehniloški i naučni razvoj. Globalizacija, rapidna urbanizacija, kao i sve veća konkurencija između gradova kako bi privukli što veći finansijski kao i ljudski kapital izaziva konstantan pritisak na gradove da urade što više korišćenjem što manje resursa. Odatle se opitimizacija i održivost ističu kao ciljevi koje tehnološki sistemi ostvaruju restrukturiranjem gradova ostvarujući njihovu isprogramiranost da konstantno prikupljaju, analiziraju i reaguju na informacije iz realnog vremena. (Halegoua, 2020)

Početkom 2000-ih godina IBM, kao vodeća tehnološka kompanija koristi pametne gradove kako bi opisala poboljšanja javnih usluga i infrastrukture primenom informacionih tehnologija i analize podataka. Godine 2010. IBM u saradnji sa opštinskom vladom Rio de Žaneira osniva *Operation Center – OC* i *Inegrated Center of Command and Control – ICCC, big data* centre, koji su kontinuirano prikupljali i interpretirali podatke od oko 30 javnih servisa u okviru grada Ria. *ICCC* služi kao obezbeđenje i operativni štab dok *OC* prikuplja informacije od strane određenog broja javnih poslova uključujući sakupljanje smeća, transpotr i javna bezbednost sa vizuelizacijama i upozorenjima o saobraćajnim nezgodama, vremenskim uslovima i nestancima struje. Kada su se 2012. godine srušile tri susedne zgrade u centru Ria javni zvaničnici su hvalili *OC* i njenu momentalnu reakciju. Zaposleni u *OC* su uputili kola hitne pomoći i vatrogasce na lice mesta, isključili su gas i struju, zatvorili obližnje stanice metroa, a zatim su tvitovali upozorenja kako bi ljudi izbegavali tu oblast grada. Na taj način poboljšan je sistem za reagovanje na katastrofe i hitne slučajeve u Riju, a sve to je pripisano *OC-u.* (Halegoua, 2020)

Još neki od primera pametnih gradova su Songdo u Južnoj Koreji i Santender u Španiji. U ovim gradovima postoje sistemi uličnih rasveta koji se uključuju i isključuju u zavisnosti od doba dana, sistemi za merenje vlažnosti zemljišta koji omogućavaju automatsko zalivanje parkova, kao i alarmni sistemi u kontejnerima za smeće koji upozoravaju kada se oni napune. Svi podaci koji su na ovaj način prikupljeni prosleđuju se u centralni sistem za nadgledanje i upravljanje pametnim gradom. (Radenković, i drugi, Internet inteligentnih uređaja=Internet of things, 2017)

Iako većina projekata vezanih za pametne gradove podrazumeva to da postojeći gradovi postanu pametni, Songdo je jedini pametan grad na svetu koji je izgrađen od nule. Opisan je kao prvi u svetu *greenfield* pametan grad. Songdo je jedan od najambiciozniji projekat pametnog grada. (Kshetri, Alcantara, & Park, 2014)

Usled povećanja broja stanovništva u gradovima dolazimo do velikog broja problema koji nastaju u društvenoj i ogranizacionoj sferi. *IoT* rešenja su tu da bi se njihovom primenom rešili problemi upravljanja gradskih struktura, manjka resursa, otežanog odvijanja saobraćaja kao i ugrožavanja životne sredine. Time se ostvaruje pametna infrastruktura za oblasti kao što su, saobraćaj, električna energija, vodosnabdevanje, stambena izgradnja i javni servisi. (Radenković, i drugi, Internet inteligentnih uređaja=Internet of things, 2017)

2.2 PAMETNE KUĆE

Osnovni zadatak pametnih kuća je da svojim stanarima pruža komfor i kontrolu, tako što se zahvaljujući ugrađenim uređajima prilagođavaju njihovim aktivnostima, raspoloženju, navikama i životnom stilu. Stanarima se omogućava da na jednostavan način upravljaju osvetljenjem, multimedijalnim uređajima, klimatizacijom i slično putem daljinskog upravljača, veb aplikacije ili mobilnog telefona. (Radenković, i drugi, Internet inteligentnih uređaja=Internet of things, 2017)

Ako pitamo nekoga ko je u zdravstvenom sektoru, pametna kuća se definiše kao prebivalište koje omogućava prevenciju bolesti, praćenje zdravlja kao i pružanje pomoći za probleme koji su izazvani bolešću kako bi poboljšali kvalitet života. U građevinskom sektoru pametna kuća

se posmatra kao životni prostor koji poseduje tehnologije koje omogućavaju uređajima i sistemima da budu kontrolisani automatski. Energetski sektor povezuje pametne kuće sa efikasnim obezbeđivanjem, proizvodnjom i potrošnjom energije. U skladu sa konceptom *IoT,* informaciono-komunikacioni sektor se primarno fokusira na inovativna rešenja koja su dizajnirana da poboljšaju povezanost ljudi i uređaja, dok se takođe bavi zabavom i radom na daljinu. (Solaimani, Keijzer-Broers , & Bouwman , 2013)

Osnovne karakteristike pametnih kuća su (Radenković, i drugi, Internet inteligentnih uređaja=Internet of things, 2017):

- Komfor, toplotna, zvučna i fizička izolacija i zaštita;
- Inteligencija koja se ogleda u prilagodljivosti, štedljivosti i racionalnosti u izboru i korišćenju energije;
- Automatizacija velikog broja uređaja i sistema s ciljem podizanja kvaliteta življenja.

3. PRIMENA IOT TEHNOLOGIJA U IGRAMA

Primena IoT-a u industriji video igara postaje sve zastupljenija, omogućavajući igračima i developerima da kreiraju svoje virtualne svetove sa sveobuhvatnijim iskustvima i sadržajem. (How the Internet of Things Impacts Game Developmant, 2023)

U sličaju video igara primena IoT uređaja u velikoj meri zavisi od samog tipa igre. Danas postoji veliki broj odevne opreme sa senzorima koji omogućavaju da se karakteri u igrici kreću kao igrači u stvarnosti. Moguće je uspostaviti komunikaciju između igrača, primenom različitih konzola, čime se daje mogućnost igračima da zajedno igraju. Bežični upravljači, gejming oružja su još jedan način da se primenom senzora, pokreti u stvarnom životu odraze na igru. (Muskan, 2021)

IoT je u velikoj meri i na raličite načine uticao na razvoj industrije video igara (How the Internet of Things Impacts Game Developmant, 2023) :

- Povećana angažovanost korisnika;
- Poboljšan razvoj igara;
- Bolja povezanost;
- Analitika u realnom vremenu;
- Prožimjuća iskustva.

3.1 PROŽIMAJUĆE IGRE I IOT

Godine 2016. pojavila se igrica pod nazvom „Pokemon Go“, koja je za jako kratak vremenski period opčinila čitav svet. Ova igrica proširene stvarnosti je svoje korisnike navodila da istražuju stvarni svet u potrazi za „pokemonima“ korišćenjem pametnih telefona koji su služili kao medijum između stvarnih i vurtuelnih elemenata.

Ovo je tipičan primer prožimajućih igara koje za cilj imaju da igračko iskustvo prošire na realan svet; ulice grada, prirodu, pa čak i dnevnu sobu. Korišćenjem mobilnih telefona i senzora koje oni poseduju prikupljaju se informacije o korisniku kao što su njegova lokacija, šta radi, pa čak

i kako se oseća i na osnovu toga se kreira jedinstveno igračko iskustvo. (Benford, Magerkurth, & Ljungstrand, 2005)

Neki od primera prožimajućih igara su:

- Negone;
- The Killer;
- The Beast;
- Shelby Logan's Run;
- Ingress;
- Can You See Me Now?.

Sve ove prožimajuće igre se zasnivaju na tri ključne tehnologije (Benford, Magerkurth, & Ljungstrand, 2005):

- Displej – koji digitalni sadržaj čini dostuplim igračima dok se oni kreću kroz fizički svet, uključujući mobilne telefone, ručne računare, slušalice, odevne reačunare, kao i interaktivne projekcije i opipljive interfejse ugrađene u okruženje;
- Bežične komunikacije – koje omogućavaju igračima komunikaciju sa udaljenim serverima i drugim igračima, uključujući mobilnu telefoniju (3G, GPRS i GMS) i *Bluetooth* ;
- Senzore – koji prikupljaju informacije o igraču uključujući GPS lociranje, kamere, mikrofone, čak potencijalno i psihološke senzore.

3.2 DRUŠTVENE IGRE I IOT

Već smo se upoznali sa prožimajućim igrama koje se fokusiraju na to da klasične video igre predstave u stvarnom svetu. Drugi tip se fokusira na socialnu interakciju. (Benford, Magerkurth, & Ljungstrand, 2005)

Primeri ovih igara su (Benford, Magerkurth, & Ljungstrand, 2005):

- *Pirates!* – fantazijska igra trgovine i ratovanje na moru;

- *Can You See Me Now?*;
- *STARS* – platforma za proširene društvene igre koje zadržavaju visok nivo socijalne interakcije koji je specifičan za tradicionalne društvene igre.

Industrija video igara je u stalnoj potrazi za načinima da privuku nove korisnike koji ne pripadaju grupi strastvenih *„gejmera"*, u pokušaju da i od njih stvore verne igrače i korisnike. Uspeh *Nitendo's Wii* kontrolora se u velikoj meri pripisuje tome što su pokreti koji se koriste u njegovim igrama jako intuitivni, što mu je omogućilo pristup tržištu koji se ne podrazumeva kao „gejmersko". *Nitendo's Wii* kontrolor je uspeo da privuče roditelje koji ostaju kod kuće, staračke domove kao i poslovne objekte. Prednost ovog kotrolora je ta što omogućava specifičnu socialnu interakciju kada igra više ljudi (*eng. multi-player mod*). (Haller, Forlines, Koeffel, Leitner, & Shen, 2010)

3.2.1 Can You See Me Now?

„Can you see me now?" je specifična igra zbog toga što spada u jednu od prvih igara zasnovanih na lokaciji igrača. Za potrebe igre kreirana je 3D virtuelna mapa prostora na kome se igra odvija. Jedan deo igrača je preko kopijutera povezanog na Internet upravljala svojim avatarima i kretala se po virtuelnoj mapi, dok je drugi deo igrača bio na ulicama iste mape u stvarnom svetu (tragači). (Can You See Me Now?, 2013)

Zadatak tragača je bio da pronađu online igrače korišćenjem mobilnih uređaja kako bi pratili njihovu lokaciju, dok je u isto vreme njihova lokacija bila predstavljena na 3D mapi u stvarnom vremenu. Zadatak online igrača je jednostavano bio da istražuju virtuelni svet dok u isto vreme izbegavaju tragače, u suprotnom su bili izbačeni iz igre. (Can You See Me Now?, 2013)

Slika 1. Can You See Me Now? (Can You See Me Now?, 2013)

3.2.2 STARS platforma

Kako bi sačuvali socijalni aspekat tradicionalnih društvenih igara pružajući interfejs sa opipljivim elementima kreirana je STARS platforma. Virtualni elementi igara kreiranih za STARS platformu pružaju veliku slobodu u kreiranju novih i uzbudljivih igara koje je nemoguće ostvariti primenom tradicionalnih media. (Magerkurth, Memisogul, Engelke, & Streitz, 2004)

Slika 2. STARS platform setup

Ideja je kreirati sledeću generaciju društvenih igara koja će u potpunosti iskoristiti prednosti informaconih tehnologija, a u isto vreme sačuvati taj socijalni aspekt koji poseduju tradicionalne igre. Osnovna komponenta STARS platforme je sto za igru koji omogućava uvođenje različitih uređaja kako bi se ostvarili dodatni benefiti u cilju da se poboljša igračko iskustvo. (Magerkurth, Memisogul, Engelke, & Streitz, 2004)

Slika 3. STARS platform table

Interaktivni sto za igru STARS platforme je horizontalni ekran osetljiv na dodir na kome su prikazane table odgovarajćih društvenih igara. Uključene su i fizičke figurice koje su specifične

za tradicionalne igre. Kamerama se prate pozicije figurica i igrača. Primenom RFID tehnologije omogućeno je praćenje različitih fizičkih token postavljenih na sto, koji po samoj logici igre mogu predstavljati različite elemente. (Magerkurth, Cheok, Mandryk, & Nilsen, 2005)

Neke od igara implementiraih na STARS platformi su (Magerkurth, Cheok, Mandryk, & Nilsen, 2005):

- KnightMage;
- Riziko;
- Monopol.

3.3. PRIMERI MODERNIH DRUŠTVENIH IGARA I IOT

3.3.1 Asocijacije

Asocijacije ili papirići (*eng. Charades*) je jako jednostavna društvena igra koja ima veliki broj varijacija. Jedna od varijacija se igra na mobilnom telefonu putem mobilne aplikacije.

Igra započinje tako što birate kategoriju (špil) karata koju želite da pogađate. Telefon stavljate na čelo okrenut ka vašim prijateljim čiji je zadatak da vam pamtomimom, pevanjem ili

Slika 4. Igra Asocijacije

jednostavnim opisivanjem u što kraćem vremenskom roku objasne reč prikazanu na ekranu telefona.

Ako pogodite o kojoj se reči radi okrećete ekran telefona na dole, a ako želite da preskočite zadatu reč ekran telefona okrećete na gore.

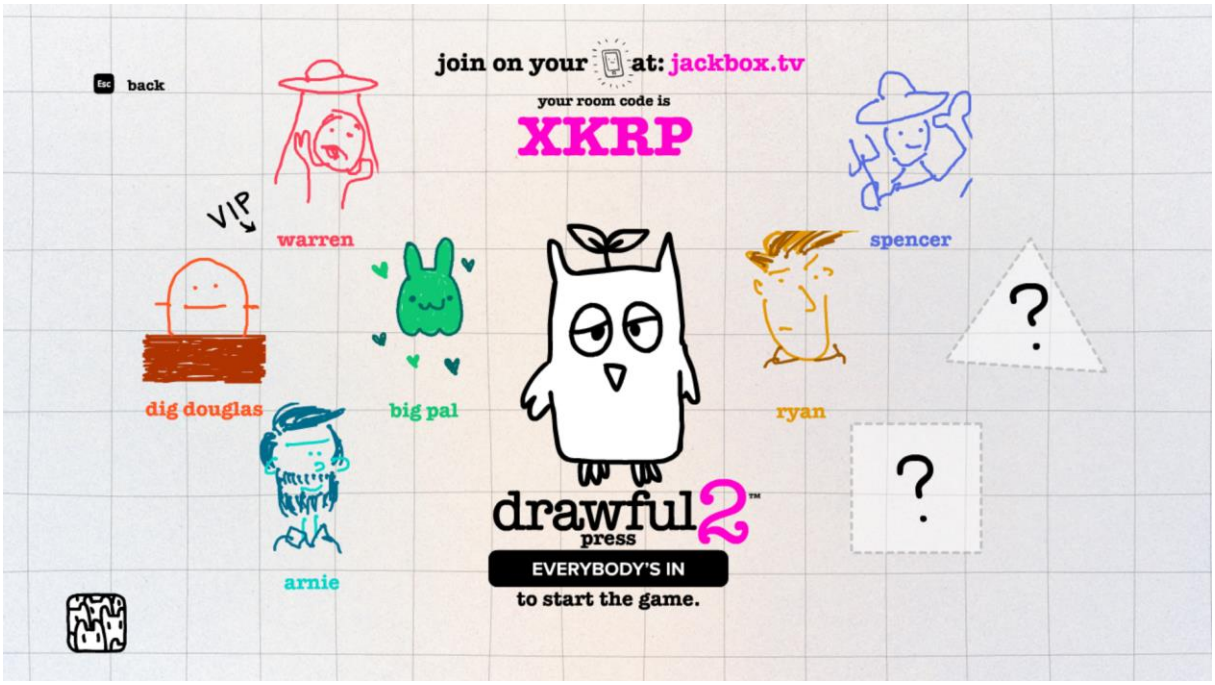
Cilj igre je pogoditi što više reči u zadatom vremenskom periodu.

Mobilni uređaji kao takvi spadaju u pametne uređaje. U ovoj igri reči su prikazane na ekranu uređaja, dok se informacije o položaju ekrana prikupljaju odgovarajućim senzorima koji se nalaze u telefonu. Senzori koji se najčešće koriste za određivanje položaja telefona su akcelerometar (*eng. accelerometer*) i žiroskop (*eng. gyroscope*).

3.3.2. Jackbox Games

The Jackbox Party Pack je društvena igra namenjena za žurke u kojoj igrači mogu igrati iz iste prostorije i online. Prednost ove igre je veliki broj igrača. Jedan igrač je host koji započinje igru i kreira „sobu za čekanje“ kojoj se igrači priključuju putem svojih mobilnih telefona ili kompjutera, koji služe kao kontrolori za interakciju sa igricom, odlaskom na Jackbox web aplikaciju. (How to Play Jackbox Games, 2023)

Kada su svi igrači tu igra može da počne. Jackbox do sada ima deset kompleta igara. Svaki komplet poseduje različite igre. Jedna od ovih igara je Drawful. U ovoj igri igrači dobijaju zadatak koji treba da nacrtaju što bolje mogu. Kada svi igrači završe sa crtanjem na ekranu se prikazuje jedan od crteža. U tom trenutku zadatak igrača je da naprave što bolji opis crteža. Kada svi završe sa unosom opisi se prikazuju na ekranu, dok je jedan od tih opisa onaj koji je dobila osoba čiji je crtež. Igrači glasaju po kom opisu je crtež napravljen.



Slika 5. Drawful 2 - a jackbox game

3.3.3 Unlock! Escape Adventures

Unlock! Escape Adventures je *escape room* društvena igra koja se igra za stolom. Ova igra se sastoji iz dva dela, špila karata i aplikacije. Grupa ljudi prelazi igru tako što prolaze kroz definisane scenarije i rešavaju zagonetke kako bi dobili nove informacije. (BoardGameGeek, 2023)

Trenutno postoji 3 scenarija u kojima igrači mogu da uživaju (Gariepy, 2023):

- The Formula
- Squeek and Sausage
- The Island of Doctor Goorse

Svaki scenario se sastoji iz špila od 60 karata koje su obeležene sa zadnje strane brojem ili slovom. Karte imaju nekoliko obeležja na sebi koja određuju na koji način se sa njima interaguje. Postoje karte koje služe kao objekti koji se mogu koristiti, lokacije koje se mogu obilaziti, „mašine“ sa kojima se najčešće interaguje korišćenjem aplikacije. (Gariepy, 2023).

Slika 6. Unlock! Escape Adventure karte

Sama aplikacija pre svega služi kao tajmer, ali ima i drugih namena. U slučaju da igrači ne mogu da reše neku zagonetku aplikacija im nudi pomoć, u slučaju da igrači pogrešno reše neku zagonetku igrači plaćaju penale u vidu dodatog vremena na tajmer igrice. Ako je u pitanju karta „mašina“ igrači koriste tu mašinu preko aplikacije. Interakcije karata i aplikacije su raznovrsne, od jednostavnog unosa koda do promene realnosti posmatranjem karte kroz kameru telefona (Gariepy, 2023).

Slika 7. Unlock! Escape Adventure aplikacija

3.3.4. Operation

Operation je jedna od omiljenih porodičnih društvenih igara u Americi, u kojoj je cilj igrača da izvrše operaciju nad pacijentom tako da mu ne nanesu bol. Igra se sastoji od table na kojoj je prikazan pacijent sa svojim organima, pincete - koja je povezana na tablu i služi kako bi se organi izvadili i kartica koje definišu koji je sledeći organ nad kojim treba operisati (BoardGameGeek, 2023).

Sa tehničke strane imamo tablu koja je priključena na baterije. U slučaju da igrač pincetom dotakne metalni okvir rupice u kojoj se nalazi organ koji on treba da izvadi nos pacijenta zasvetli i oglašava se zvono, što znači da je red na sledećeg igrača.

Svaka uspešna operacija donosi određeni broj poena, pobednik je igrač sa najviše poena kada su svi organi izvađeni.

Slika 8. Društvena igra Operation

4. PREGLED TEHNOLOGIJA

4.1. PYTHON

U dokumentaciji, Python se definiše kao interpretirani, objektno orjentisan programski jezik visokog nivoa koji poseduje dinamičku semantiku. Primamljiv zbog svojih struktura podataka visokog nivoa zajedno sa dinamičkim kucanjem i povezivanjem za brzi razvoj aplikacija, upotrebu kao skriptni jezik i jezik za povezivanje postojećih komponenti. (Python Software Foundation, n.d.)

Guido van Rossum, kreator Python-a, je osamdesetih došao na ideju da kreira programski jezik koji će nadoknaditi nedostatke koje je imao ABC programski jezik, projekat na kome je Rossum radio u tom periodu. ABC je u velikoj meri inspirisao Rossuma da njegov program podržava proširivanje i bude dosta fleksibilniji. (Venners, 2003) Prva verzija Python-a objavljena Februara 1991. je verzija 0.9.1 .

Neke od prednosti upotrebe Python-a su (Python Software Foundation, n.d.):

- Njegova jednostavna sintaksa koja je veoma laka za naučiti, sa ciljem povećanja čitljivosti koda;
- Podržava module i pakete;
- Python akcenat stavlja na jednostavnost sintakse koju koristi i na čitljivost programa;
- Povećana produktivnost zbog jednostavnosti u isprvljanju grešaka.

Python se korist u velikom broju različitih domena, kao što su:

- Analiza podataka i mašinsko učenje;
- Razvoj veb aplikacija;
- Pisanje skripti za automatizaciju;
- Razvoj i testiranje softvera;
- Razvoj pametnih sistema;

Za razvoj veb aplikacija Python nudi različite okvire (eng. frameworks) i mikrookvire kao što su (Python Software Foundation, n.d.):

- Django;
- Pyramid;
- Flask;
- Bottle.

Standardna biblioteka Python-a podržava veliki broj Internet protokola: obrada e-pošte, podrška za FTP, IMAP i druge protokole kao i jednostavan soket (*eng. Socket*) interfejs. Unutar PyPI (*eng. Python Package Index*) može se naći još dosta korisnih biblioteka. (Python Software Foundation, n.d.)

Postoji dosta razloga zašto koristiti baš Python u kreiranju pametnih okruženja (David, 2023):

- Python je jezik koji je univerzalan za više platformi (Windows, Linux, macOS), ovo otklanja probleme kompitabilnosti zbog mogućnosti da kod bude pokrenut na različitim IoT uređajima i sistemima;
- Skalabilnost Pythona je jedan od važnih faktora pri izboru pošto se može koristiti i za razvoj malih a i velikih projekata;
- Python olakšava kreiranje aplikacija koje lako mogu da komuniciraju sa drugim uređajima i platformama tako što pruža veliku podršku za različite IoT protokole (HTTP, MQTT, CoAP) ;
- Za razvoj IoT aplikacija u kojima su podaci osnovna komponenta, Python je odličan alat za analizu i vizuelizaciju podataka pomoću biblioteka kao što su Pandas, NumPy, Matplotlib.

4.1.1 Konkurentno programiranje u Python

4.1.1.1 Osnovni pojmovi

Schneider (1997) navodi da veliki broj softvera danas zavisi od konkurentnog programiranja i da je za svaku pohvalu uspeh koji je postignut od strane programera koji koriste jednostavne i primitivne alate za obavljanje svog posla.

Da bi smo sagledali u kakvim sistemima se javlja potreba za konkurentnim programirnjem potrebno je objasniti reaktivne i transformacione sisteme. U transformacionim sistemima implementira se funkcija od ulaza do izlaza, dok se ispravnost sistema ogleda u njegovom početnom i končnom stanju. Sa druge strane imamo reaktivne sisteme čiji rezultati mogu uticati na okruženje u kojem se nalaze, na taj način utiču na ulaze u budućnosti. Iz navedenih razloga potreba za konkurentnim programima se najčešće javlja kod reaktivnih sistema. Za razliku od transformacionih sistema, ispravnost reaktivnih se odnosi na njegovo početno stanje, konačno stanje i međustanje. (Schneider, 1997)

Sledeći zadatak je napraviti razliku između sekvencijalnog, paralelnog i konkurentnog izvršavanja. Sekvencijalno izvršavanje podrazumeva da se sledeća računarska operacija izvršava tek nakon završetka prethodne i to u unapred definisanom redosledu. Kod paralelenog izvršavanja istovremeno se izvršava veći broj računarskih operacija, sekvenci operacija, delova programa ili programa. Konkurentno izvršavanje je izvršavanje više programskih tokova jednog programa s tim što se oni ne moraju izvršavati istovremeno. (Ikodinović & Jovanović, 2004)

Operativni sistem je jedan od najstarijih primera konkurentnog programa, no ne treba ga vezivati isključivo za sistemsko programiranje, postoje veliki benefiti konkurentsnosti programa kod kog je neophodan bilo koji vid paralelizma. Za postizanje paralelnog izvršavanja procesa, kao i mogućnosti da jedan procesa utiče na izvršavanje drugog, komunikacija između njih je neophodna. Najčešći vid komunikacije procesa je slanje poruka, upisivanje i čitanje iz deljenih promenljivih (Raynal, 2012)

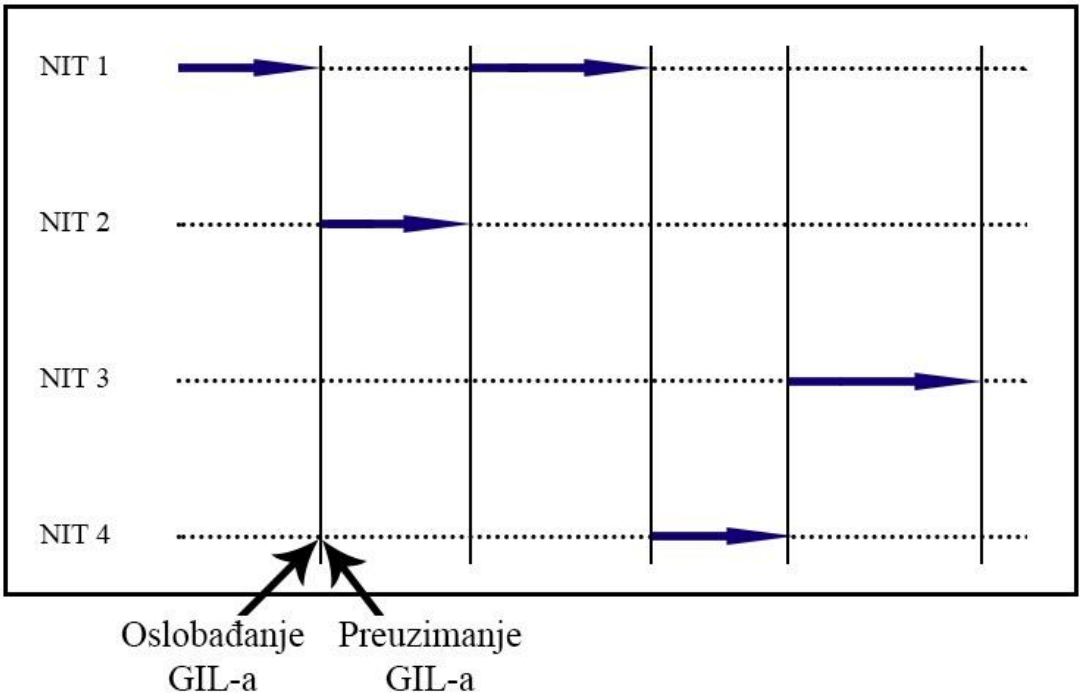
Pre nego što nastavimo sa temom o konkurentnom programiranju u Pythonu neophodno je napraviti još jednu distinkciju, i to između distribuiranog i konkurentnog programiranja. Dok se kod konkurentnog programiranja podrazumeva da postoji više tokova izvršavanja koji dele memoriju u okviru istog sistema, kod distribuiranog programiranja tokovi izvršavanja se nalaze na različitim mašinama pa samim tim imaju i odvojene memorije. (Isailović, Stanković, Vasiljević, & Ristić, 2021)

4.1.1.2. Python moduli

Konkurentno programiranje u Python-u nije ništa drugo no izvršavanje dva ili više zadataka istovremeno. Python pruža tri osnovna tipa konkurentnog programiranja, svaki u zasebnom modulu i fokusiran na različite jedinice konkurentnosti (Brownlee, 2022):

- *Multiprocessing* – konkurentnost zasnovana na procesima;
- *Threading* – konkurentnost zasnovana na nitima;
- *Asyncio* – konkurentnost zasnovana na kooperativnim tokovima

Međutim, Python zbog posedovanja Globalnog interpreterskog katanca (*eng. GIL – Global interpreter lock*) koji se nalazi u srži izvršavanja bilo kakvog Python koda, nije predviđen za paralelizam. GIL je katanac za međusobno isključivanje, to je poseban resurs koji nit mora da „drži“ kod sebe kako bi se izvršavala, što podrazumeva da u datom momentu samo jedna nit ga može posedovati. Na koji način funkcioniše GIL može se videti na sledećoj slici. Bitno je naglasiti da pojedine Python biblioteke u potpunosti mogu zaobići GIL time što će u pozadini raditi na nekom drugom jeziku. (Forbes, 2017)



Slika 9. Rad GIL-a

Modul *threading* omogućava rad sa nitima. Nit nije program, već deo programa i predstavlja najmanju izvodljivu jedinicu unutar operativnog sistema. U odnosu na procese niti koriste manje resursa, dok je jedna nit blokirana i u stanju je čekanja, druga može da se izvršava, jedna nit može da čita upisuje i menja podatke druge niti. (Concurrency in Python - Threads, 2023)

Postoji nekoliko stanja u kojima se može nalaziti nit, razlikujemo sledeća (Palach, 2014):

- Kreirana – Stanje u kojem se nit nalazi kada je kreirana od strane glavnog procesa, nekon čega je prosleđuje u red za čekanje niti;
- U izvršavanju – Kada je nit u stanju izvršavanja ona koristi procesorske resurse;
- Spremna/Čekanje – Stanje u kojoj se nit nalazi kada je u redu čekanja i može preći u stanje izvršavanja čim joj računar dodeli procesorske resurse;
- Blokirana – U ovom stanju nit ne koristi procesorkse resurse pošto čeka i ne može da nastavi sa svojim izvršavanjem;
- Završena – Nit po završetku svog toka izvršavanja oslobađa resurse, čime se životni cikljus niti završava.

Threading unutar Python-a se postiže pomoću klase *Thread*, instanciranjem ove klase dobija se nit za koju se definiše funkcija koju ona poziva. Nit se započinje pozivanjem funkcije start(). (Brownlee, 2022)

Neke od metoda koje klase *Thread* su:

- run() – metoda koja služi kao ulaz niti;
- start() – metoda koja služi za pokretanje niti pozivanjem funkcije run();
- join() – metoda koja ne dozvoljava dalje izvršavanje koda dok se nit ne završi
- isAlive() – metoda koja suži za utvrđivanje da li se nit izvršava ili ne;
- getName() – metoda koja vraća naziv niti;
- setName() – metoda koja postavlja naziv niti;

Sada treba razrešiti pitanje koje nastaje postojanjem GIL-a i kako on interaguje sa nitima, modula threading. Pošto GIL isključuje postojanje paralelnog izvršavanja niti, momenti kada

je poželjno koristiti niti su kada imamo operacije koje veliki deo vremena provode u stanju čekanja. Treba naglasiti da se ovime postiže samo prividna paralelnost. (Palach, 2014)

Kako bi se započeo rad sa nitima prvo je neophodno importovati klasu *Thread* modula *threading*.

```
from threading import Thread
```

Slika 10. Importovanje klase Thread

Inicijalizacijom objekta klase *Thread* kreiramo novu nit kojoj prosleđujemo funkciju koju ona treba da izvrši kao i argumente funkcije.

```
newThread = Thread(target=None,args = [])
```

Slika 11. Kreiranje nove niti

Pozivanjem metode *start()* pokrećemo nit, dok pozivanjem metode *join()* program čeka da se data nit izvrši.

```
newThread.start()
newThread.join()
```

Slika 12. Pokretanje niti

Pod procesima se podrazumevaju kompjuterski programi. Svaki Python program poseduje jednu nit koja se naziva *main.* Pomoću klase *Process* dobijamo mogućnost multiprocesiranja, odnosno izvođenja drugih procesa pored naseg Python programa. (Brownlee, 2022) Klasa *Process* poseduje iste funkcije kao i klasa Thread.

Kooperativni tokovi se, za razliku od niti i procesa nad kojima se opertivni sistem pita kada će biti zaustavljeni, ponovo pokrenuti i izvršeni, sami pitaju kada se zaustavljaju i nastavljaju. Kooperativni tokovi se definišu korišćenjem *async/await* sintakse. (Brownlee, 2022)

4.1.2 Flask

Flask je mikro okvir kreiran 2004. godine, napisao ga je Armin Ronacher u programskom jeziku Python. Kreiran je sa namerom da olakša u ubrza proces krairanja novih veb aplikacija kao i da poboljša složene aplikacije. Vremenom Flask je postao jedan od najpopularnijih Python okvira. Zasnovan je na dve tehnologije Werkzeug i Jinja (Mufid, Basofi, M., Rochimansyah, & Rokhim, 2019). Werkzeug je koristan za dobijanje i slanje zahteva, dok Jinja2 omogućava prenos Python variabli unutar HTML kodova.

Kako bi započeli rad sa Flask okvirom prvo je nephodno kreirati Python virtuelno okruženje, koje se izvršava pozivanjem komande *venv*. Potreba za instalacijom virtualnog okruženja se javlja u tome što verzije Python-a kao i Python biblioteka instaliranih za potrebe jednog projekta neće uticati na druge. (Pallets, 2023)

```
> mkdir myproject
> cd myproject
> py -3 -m venv .venv
```

Slika 13. Kreiranje virtuelnog okruženja (Pallets, 2023)

Sledeći korak je aktivacija okruženja u kome instaliramo naš Flask okvir. Flask aplikaciju može činiti samo jedan dokument, što je u redu za manje projekte, dok se širenjem projekta kodovi raspoređuju u pakete. (Pallets, 2023)

```
> .venv\Scripts\activate
```

Slika 14. Aktivacija virtuelnog okruženja (Pallets, 2023)

```
$ pip install Flask
```

Slika 15. Instalacija Flask okvira (Pallets, 2023)

Svaka Flask aplikacija zapravo predstavlja instancu klase Flask, unutar koje se pamte konfiguracije i putanje (URLs) vezane za aplikaciju. Flask aplikacija može biti kreirana na dva načina:

- Kreiranjem globalne Flask instance na početku koda;

- Kreiranjem Flask instance korišćenjem funkcije pod nazivom *application factory*, koja će nakon izvršenih konfiguracija, registracija i ostalih podešavanja kao povratnu vrednost dati aplikaciju.

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'Dobrodosli u moju Flask aplikaciju'
```

Slika 16. Prva Flask aplikacija

Kada smo završili sa kreiranjem aplikacije njeno pokretanje se vrši iz terminala pozivanjem komande *flask*. Kada u pretraživač unesemo definisanu rutu (u ovo slučaju http://127.0.0.1:5000/) videćemo našu Flask veb aplikaciju.

```
$ flask --app flaskr run --debug
```

Slika 17. Pokretanje Flask aplikacije

Ne samo da je Flask pogodan za kreiranja malih projekata kao što su sopstvene blog aplikacije, socijalne mreže, Rest API aplikacije, kreiranje modela za mašinsko učenje, vremenske aplikacije; Flask je svoje mesto našao i kod velikih firmi kao što su (Joy, 2023):

- Pinterest;
- Twilio;
- LinkedIn;
- Uber;
- Netflix;
- Airbnb i dr.

4.2 RASPBERRY PI

Raspberry pi (*RPI*) je mikroračunar, mala prenosiva računarska ploča, koj se može povezati na monitor računara, tastaturu, miš itd. Na RPI-u programeri mogu pisati skripte i programe u programskom jeziku Python koji je ujedno i osnovni jezik jezgra operativnog sistema RPI-a. (Zhao, Jegatheesan, & Loon, 2015)

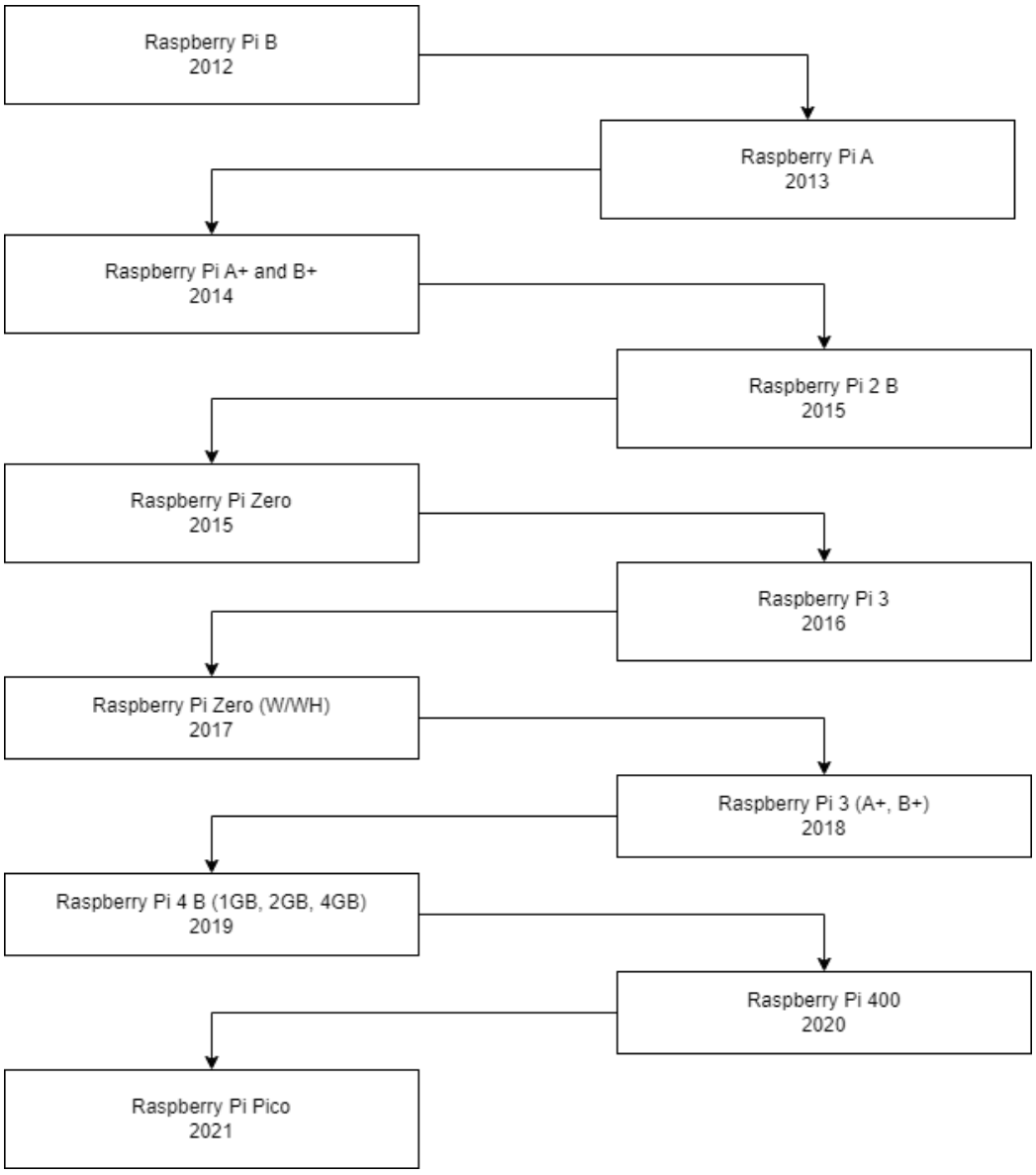
Kada se sve standardne komponente računara kao što su operativna memorija, procesor, memorija za skladištenje podataka, prikjučci kao što su USB i HDMI i mrežni interfejs, nalaze na jednoj štampanoj ploči, taj računar se naziva mikroračunar**.** Ono što ih najčešće razlikuje od klasičnih računara je procesorska arhitektura. Dok klasični računari koriste x86 i x64 arhitekture, mikroračunari koriste ARM i RISC procesorsku arhitekturu. Prednost koju mikroračunari imaju nad klasičnim računarima se najčešće ogleda njihovoj energetskoj efikasnosti, pogodnosti za rad napajanjem baterijom, što omogućava da budu postavljeni na inače teško pristupačnim mestima. Mikroračunari najčešće rade na nekoj od distribucija operativnog sistema Linux kao što su: Raspbian, Pidora i OpenELECT. Kako bi se povezivali sa ostalim uređajima kao što su senzori i aktuatori, na mikroračunarima se može naći GPIO interfejs. (Radenković, i drugi, 2017)

U izboru na kojoj platformi će se razvijati projekat, da li na mikrokontroleru Arduion ili na mikroračunaru RPI utiče više faktora. Richardson navodi primer da u slučaju kada želimo da kreiramo jednostavan termostat verovatno je bolje koristiti Arduino Uno ili neki slični mikrokontroler, ali ako naš projekat zahteva daljinski pristup termostatu preko veba kako bi nešto promenili ili prikupili određene podatke dobijene od strane termostata, prednost dajemo RPI-u. (Richardson & Wallace, 2013)

RPI je kreiran 2012. godine u Velikoj Britaniji. Na razvoju ovog mikroračunara učestvovali su Eben Upton, Rob Mullins, Jack Lang i Alan Mycroft. Od nastanka do danas RPI je prošao kroz nekoliko faza, svaka je sa sobom donosila nešto novo. Razvoj RPI-a prikazan je slikom 18. (DevicePlus, 2023)

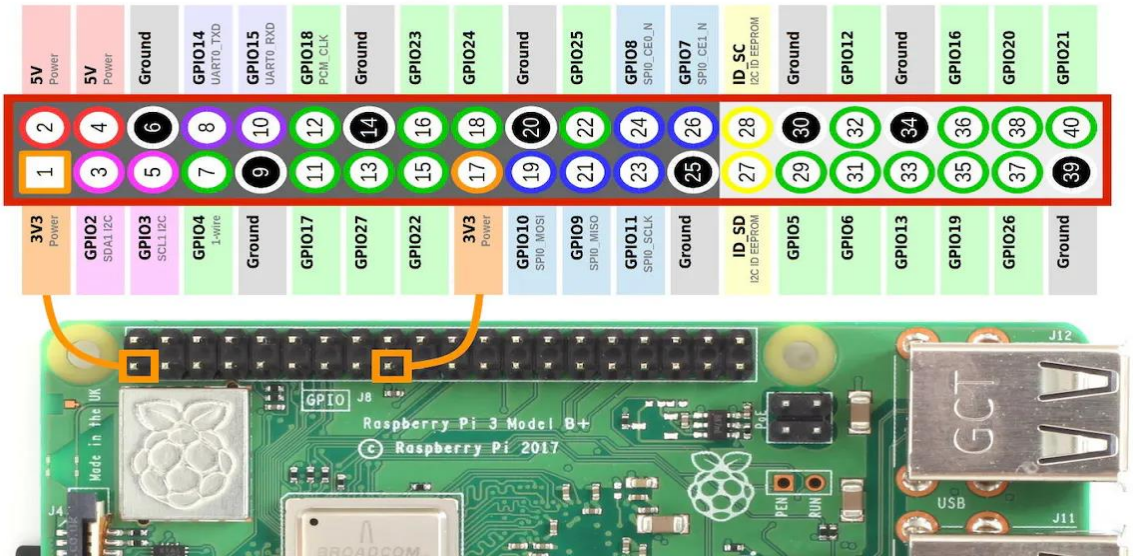
Mogućnosti primene RPI-a u razvoju pametnih okruženja je raznovrsna. Od kreiranja jednostavnog termostata kojim se daljinski upravlja, do kreiranje kompleksnih i povezanih senzora, aktuatora, koji nadgledaju i upravljaju fizičkim uređajima na udaljenim lokacija

primenom bežičnih tehnologija. Sam RPI u kombinaciji sa IoT tehnologijama (RFID tagovi, RFID čitači, barkod skeneri, GPS prijemnici i drugi pametni uređaji) omogućava to nadgledanje i upravljanje uređajima na daljinu, kreiranje automatizacionih sistema kao i prikupljanje i razmenu podataka (Nilay, 2023).



Slika 18. Evolucija Raspberry pi-a

GPIO (*eng. General Purpose Input Output*), koji služe kao elektronski interfejs između RPI ploče i različitih eksternih elektronskih komponenti, je jedan od najbitnijih karakteristika koju RPI nudi. Njihova jedinstvenost se krije u tome što omogućavaju direktan pristup ulazno/izlaznom kolu računara. (Nilay, 2023)



Slika 19. GPIO layout (Raspberry Pi Spy, 2023)

GPIO zaglavlje Raspberry Pi 4 mikroračunara se sastoji od 40 pinova koje možemo podeliti u sledeće grupe (The Robotics Back-End, 2023):

- Pinovi uzemljenja (Ground pins) – 8 pinova pripada ovoj grupi, a označeni su kao GND;
- Naponski pinovi (Power pins) – postoje po 2 ovakva pina za jačinu od 3.3V i 5V, a služe kao izvor napajanje za eksterne komponente kao sto su senzori i aktuatori;
- Rezervisani pinovi (Reserved pins) – dva pina koja se najčešće koriste za I2C (eng. Inter-Integrated Circuit) komunikaciju sa EEPROM (eng. Electrically Erasable Programmable Read-Only Memory) memorijama;
- GPIOs – preostalih 26 pinova služi za upisivanje podataka (izlaz) u eksterne komponente ili čitanje podataka iz eksternih komponenti (ulaz).

5. STUDIJSKI PRIMER – DRUŠTVENE IGRE ASOCIJACIJE

Potrebno je projektovati i implementirati pametno okruženje koja će olakšati i upotpuniti korisnicima društvenu igru Asocijacije. U okviru ove aplikacije igrači se dele u timove, a svaki igrač za sebe popunjava papiriće pojmovima po želji. Svi papirići se stavljaju na jedno mesto i igra može da počne.

Cilj svakog tima je da u zadatom vremenskom periodu pogodi što više pojmova. Sama igra se sastoji iz tri kruga koji se igraju sve dok se svi papirići ne iskoriste:

- Prvi krug – bilo koji način objašnjavanja je dozvoljen;
- Drugi krug – potrebno je objasniti pojam korišćenjem samo jedne reči;
- Treći krug – potrebno je objasniti pojam korišćenjem pamtomime.

Pobednik je tim koji je ukupno kroz sva tri kruga pogodio najviše pojmova.

Tim započinje sa pogađanjem pritiskom na dugme START. Otkucavanje vremena prikazano je na displeju. Ako igra teče bez ikakvih problema, tim pogađa sve dok se tajmer na displeju ne završi čime se oglašava i zvono (*eng. Buzzer*). To znači da je red na sledeći tim da pogađa. Pritiskom na dugme RESET se vreme na displeju ponovo postavlja i čeka se pritisak dugmeta START.

Često je zbog prirode igre nepohodno da se vreme pauzira, bez obzira na to da li igrač ne može da pročita reč, ne razume o kojoj se reči radi ili je nephodno prokomentarisati da li je način objašnjavanja ispravan. Zbog toga dugmetu START dodaje se opcija za pauziranje i ponovno pokretanje tajmera na ekranu. Za ovu funkcionalnost neophodno je korišćenje konkurentnog programiranja, konkretno niti.

Za potrebe projekta kreirana je i veb aplikacija sa kojom komuniciraju korisnik i naše pametno okruženje. Korisnik za početak unosi broj timova, naziv timova i započinje igru. Korisnik na aplikaciji ima uvid u sva dešavanja i promene koje nastaju u interakciji sa okruženjem (preostalo vreme, da li je vreme zaustavljeno ili pokrenuto, resetovanje vremena). Pored toga kroz aplikaciju je omogućeno korisniku da unosi ostvarene poene svakog tima nakon odigranih rundi. Na kraju se kroz aplikaciju proglašava pobednički tim.

Akcenat ovog projekta je na pametno okruženje pa se u skladu sa tim razvijaju korisnički zahtevi i slučajevi korišćenja.

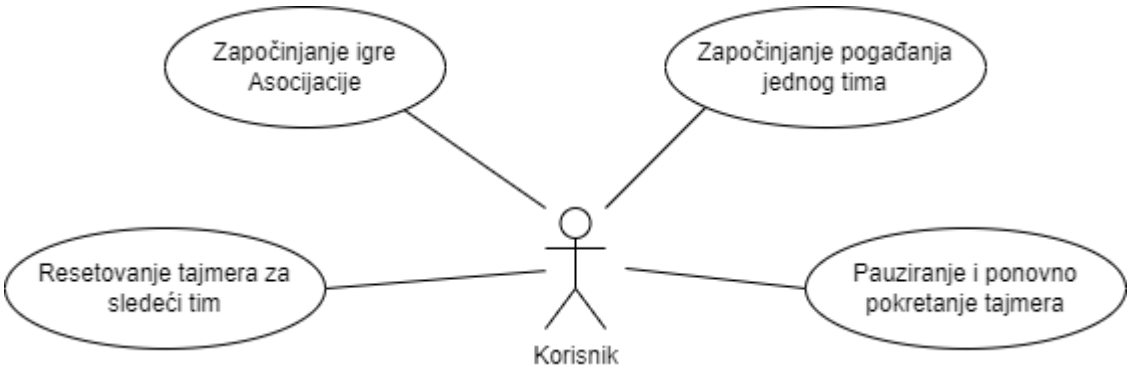
5.1. KORISNIČKI ZAHTEVI

Potrebno je projektovati i implementirati konkurentno pametno okruženje koje će upotpuniti i pružiti podršku korisnicima u igranju društvene igre Asocijacije. Korisnicima treba omogućiti da započnu svoju igru pritiskom na dugme START. Po pritiskanju dugmeta START na displeju počinje odbrojavanje tajmera, a korisnici u tom vremenskom periodu treba da pogode što više reči.

Ako vreme samo po sebi istekne (tajmer na displeju dođe do nule) i oglasi se zvono, korisnici u veb aplikaciju unose ostvarene poene. Pritiskom na dugme RESET se tajmer vraća na početak i čeka se da sledeći tim započne sa svojim pogađanjem. Na displeju je ponovo prikazano dogovoreno vreme trajanja jednog kruga pogađanja.

U slučaju da u toku pokrenute igre, igrač/korisnik nije u mogućnosti da pročita pojam koji se nalazi na papiriću ili ne razume o kom se pojmu radi, korisnik ima mogućnost da pauzira vreme ponovnim pritiskom na dugme START. Tajmer na displeju više ne otkucava i čeka se da igrač koji je zaustavio igru reši svoje nedoumice. Igra se nastavlja u trenutku kada korisnik ponovo pritisne na dugme START, a tajmer na displeju ponovo počinje da otkucava od trenutka kada je bio zaustavljen.

5.2. SLUČAJEVI KORIŠĆENJA



Slika 20. UML dijagram slučajeva korišćenja

Karakteristični slučajevi korišćenja za korisnika su:

1. Započinjanje igre Asocijacije;
2. Započinjanje pogađanja jednog tima;
3. Resetovanje tajmera za sledeći tim;
4. Pauziranje i ponovno pokretanje tajmera.

6.2.1. Započinjanje igre Asocijacije

Slika 21. Dijagram slučajeva korišćenja - Započinjanje igre Asocijacije

Slučaj korišćenja – Započinjanje igre Asocijacije

Naziv SK: Započinjanje društvene igre Asocijacije

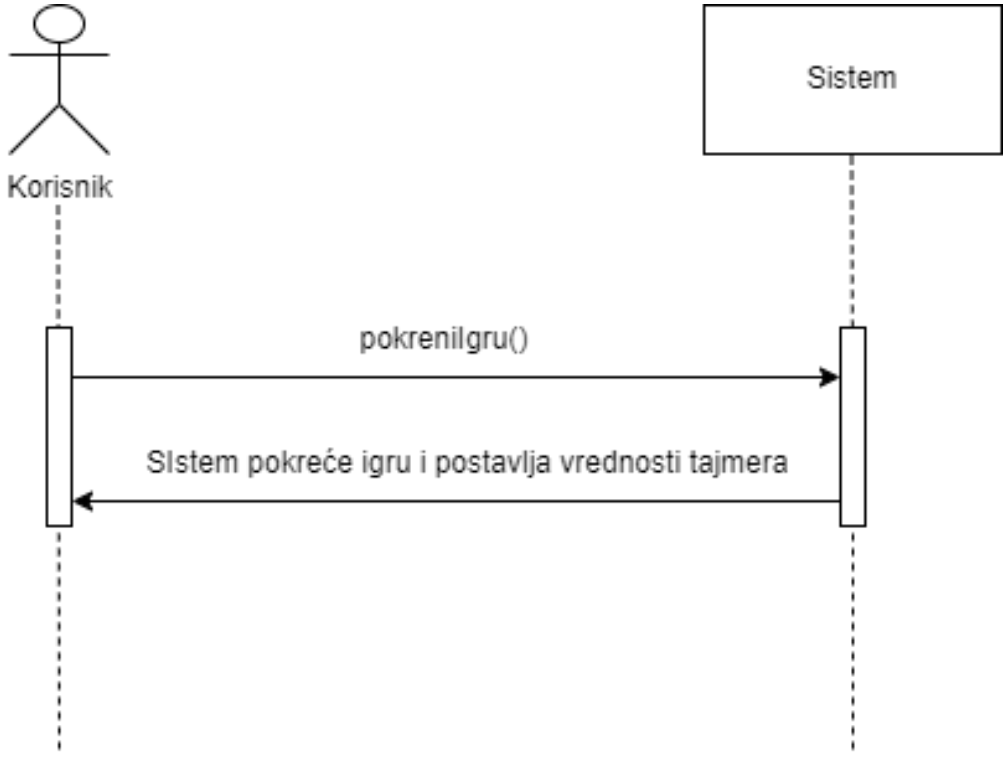
Aktori SK: Korisnik

Učesnici SK: Korisnik i sistem

Preduslov: Sistem je pokrenut i korisniku je na veb aplikaciji prikazano dugme za započinjanje društvene igre Asocijacije

Osnovni scenario:

1. Korisnik poziva sistem da pokrene društvenu igru Asocijacije
2. Sistem inicijalno postvlja vrednosti tajmera i prikazuje ih na displeju



Slika 22. UML dijagram - Započinjanje igre Asocijacije (osnovni scenario)

5.2.2. Započinjanje pogađanja jednog tima

Slika 23. Dijagram slučajeva korišćenja - Započinjanje pogađanja jednog tima

Slučaj korišćenja – Započinjanje pogađanja jednog tima

Naziv SK: Započinjanje pogađanja jednog tima pokretanjem tajmera

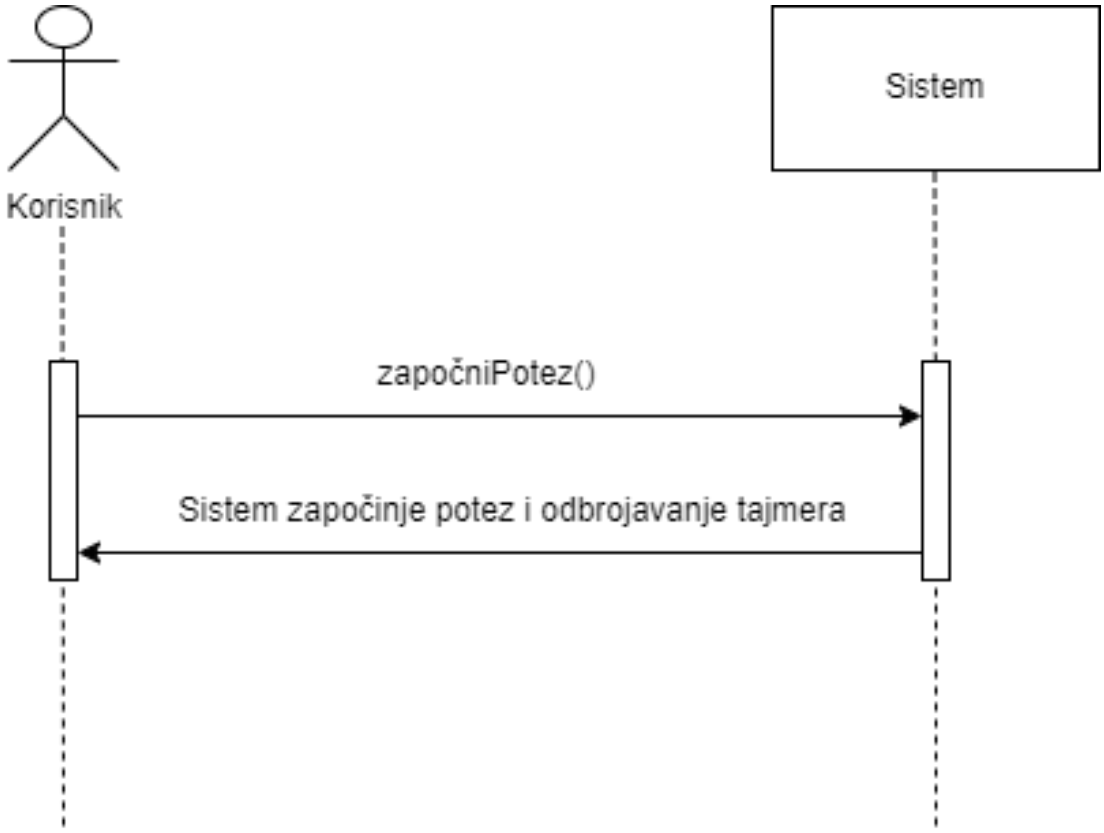
Aktori SK: Korisnik

Učesnici SK: Korisnik i sistem

Preduslov: Sistem je pokrenut i korisniku je prikazana strana Asocijacije

Osnovni scenario:

1. Korisnik poziva sistem da pokrene potez tima i odbrojavanje tajmera
2. Sistem pokreće potez i započinje odbrojavanje tajmera



Slika 24.UML dijagram – Započinjanje pogađanja jednog tima (osnovni scenario)

5.2.3. Resetovanje tajmera za sledeći tim

Slika 25. Dijagram slučajeva korišćenja – Resetovanje tajmera za sledeći tim

Slučaj korišćenja – Resetovanje tajmera za sledeći tim

Naziv SK: Priprema sistema za pogađanje sledećeg tima

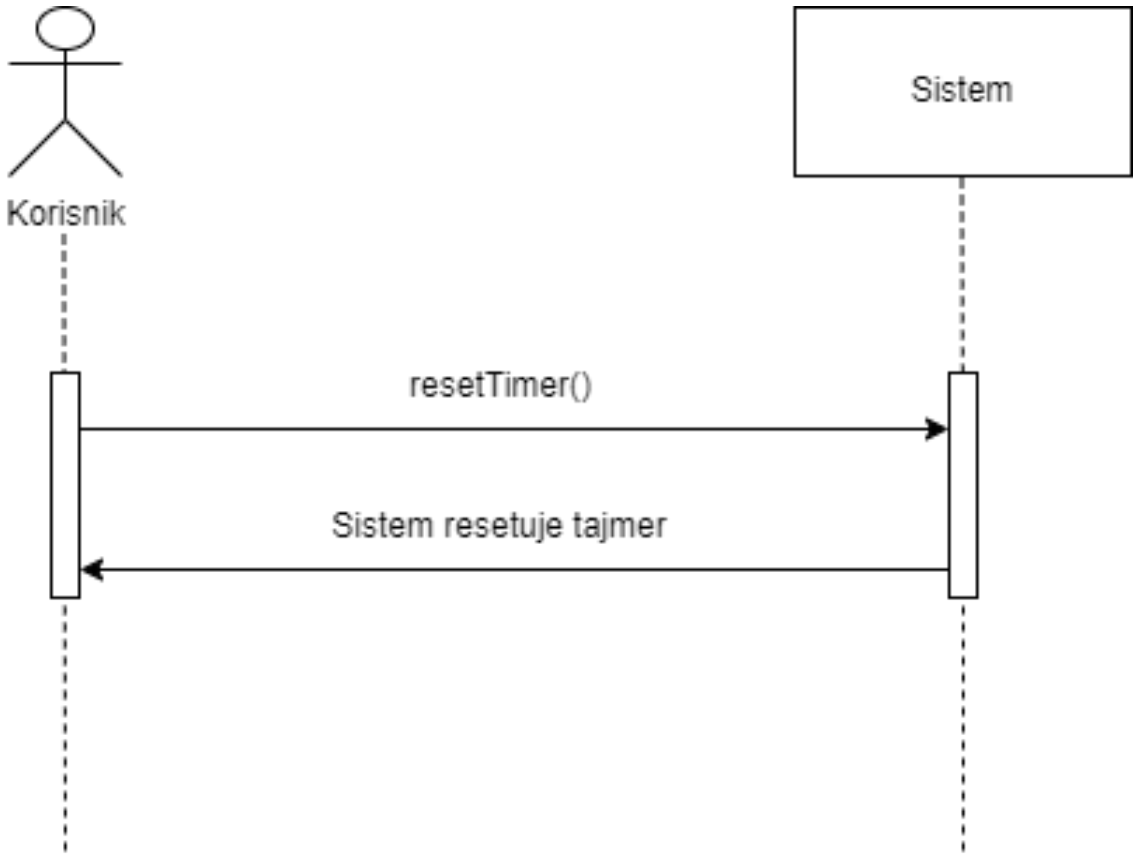
Aktori SK: Korisnik

Učesnici SK: Korisnik i sistem

Preduslov: Sistem je pokrenut i korisniku je prikazana strana Asocijacije

Osnovni scenario:

1. Korisnik poziva sistem da resetuje tajmer za sledeći tim
2. Sistem resetuje tajmer za sledeći tim



Slika 26. UML dijagram – Resetovanje tajmera za sledeći tim (osnovni scenario)

5.2.4. Pauziranje i ponovno pokretanje tajmera

Slika 27. Dijagram slučajeva korišćenja – Pauziranje i ponovno pokretanje tajmera

Slučaj korišćenja – Pauziranje i ponovno pokretanje tajmera

Naziv SK: Pauziranje i ponovno pokretanje tajmera

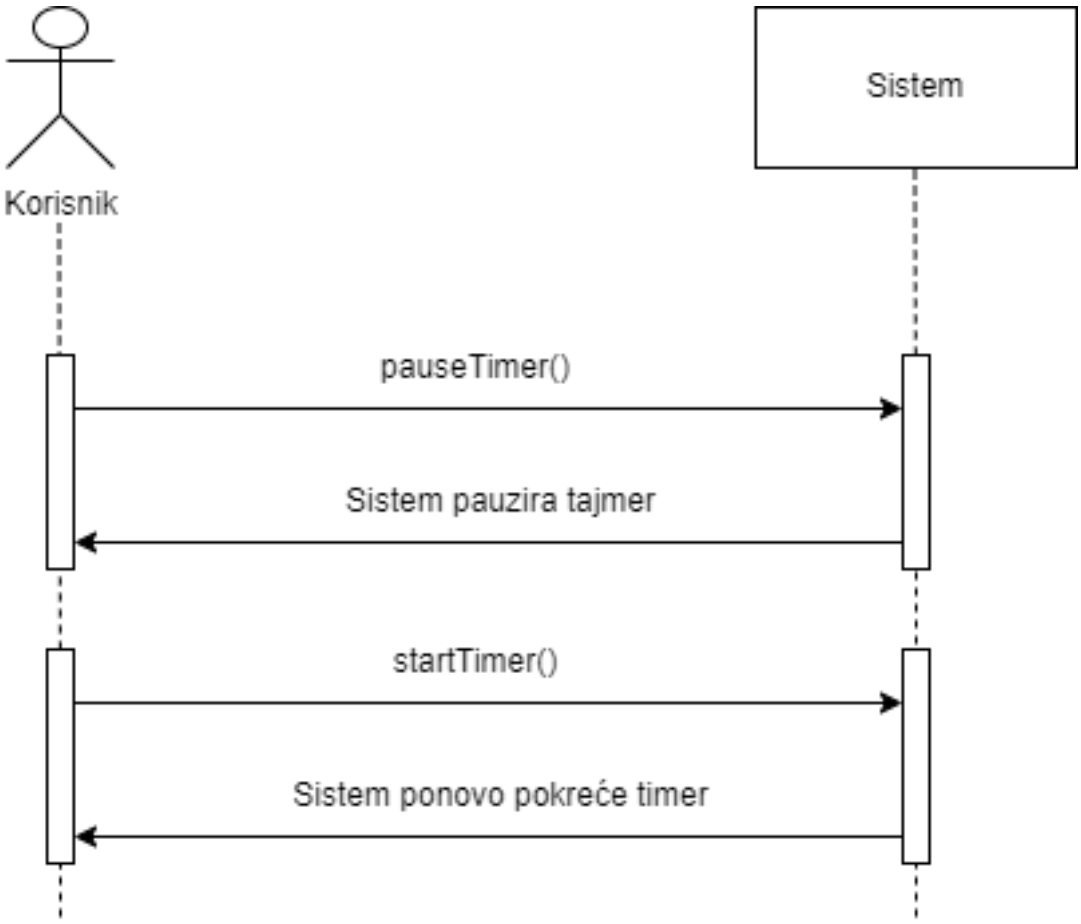
Aktori SK: Korisnik

Učesnici SK: Korisnik i sistem

Preduslov: Sistem je pokrenut i tajmer za odbrojava preostalo vreme

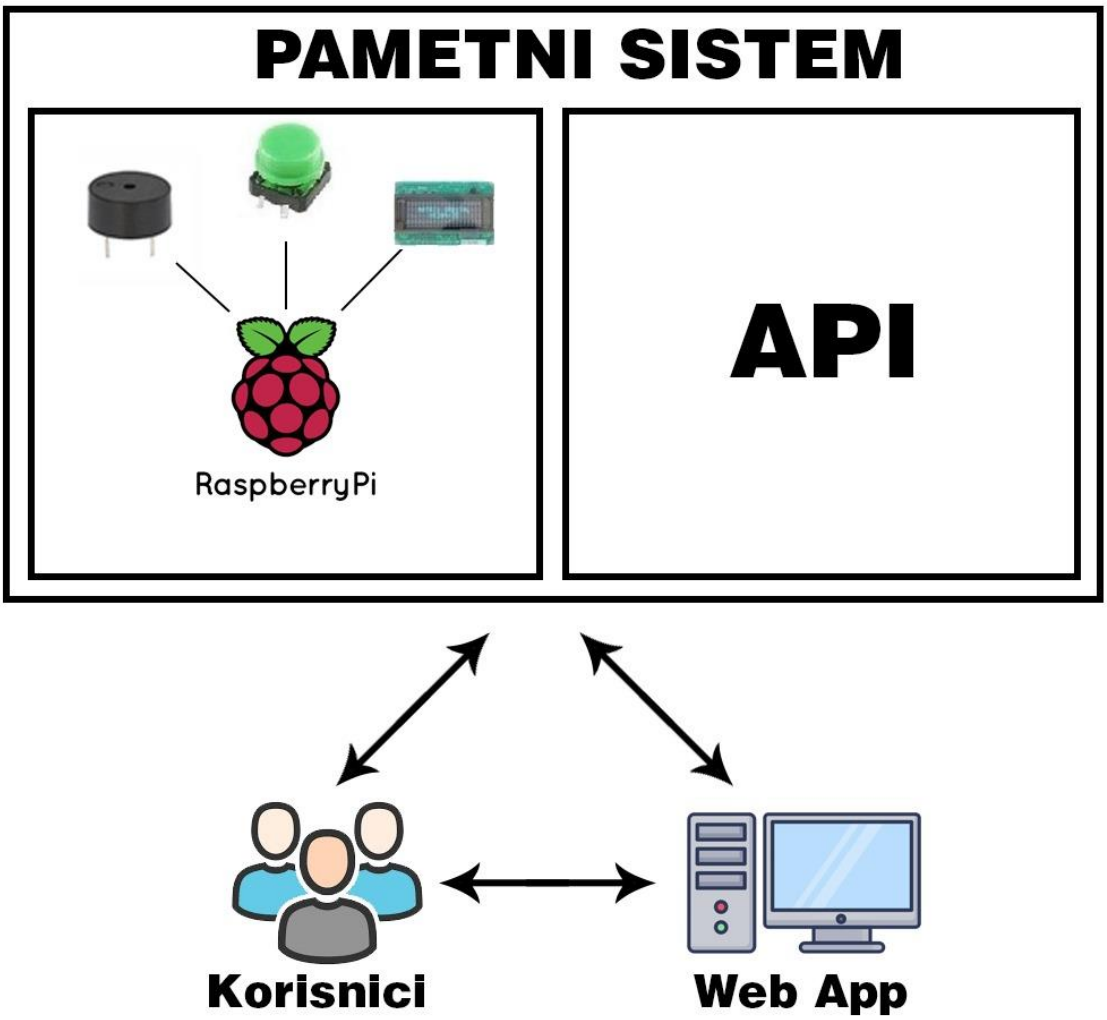
Osnovni scenario:

1. Korisnik poziva sistem da pauzira tajmer
2. Sistem pauzira tajmer
3. Korisnik poziva sistem da ponovo pokrene tajmer
4. Sistem ponovo pokreće tajmer



Slika 28. UML dijagram – Pauziranje i ponovno pokretanje tajmera (osnovni scenario)

5.3. ARHITEKTURA



Slika 29. Arhitektura sistema

Arhitektura sistema se može logički podeliti u dva dela:

1. Veb aplikacija
2. Pametni sistem – koji je dalje podeljen na hardverske i softverske komponente.

Korisnici interaguju kako sa pametnim uređajima korišćenjem aktuatora za pokretanje određenih akcija, tako i sa veb aplikacijom koja upravlja procesom igranja društvene igre. Aktuatori u pametnom okruženju su dugmići, dok reaktori koji se koriste su zvono i LED displej. Pritiskanjem dugmeta START/PAUSE – započinje tajmer ili ga pauzira, dok dugme

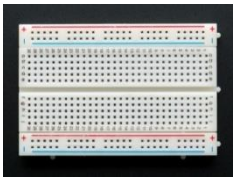
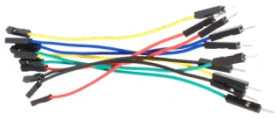
RESET resetuje tajmer na početak. Prikaz informacija o procesu igre se vidi na LED displeju. Kroz veb aplikaciju korisnici sve promene i sva dešavanja u sistemu mogu sagledati, kao i da kroz aplikaciju izvršavaju funkcije startovanja igre.

Veb aplikacija komunicira sa Raspberry Pi-em pomoću kreiranog APIa. API prima zahteve od veb aplikacije i prosleđuje ih RPIu, Takođe i šalje podatke i odgovore sa uređaja do veb aplikacije. API je implementiran primenom Flask okvira.

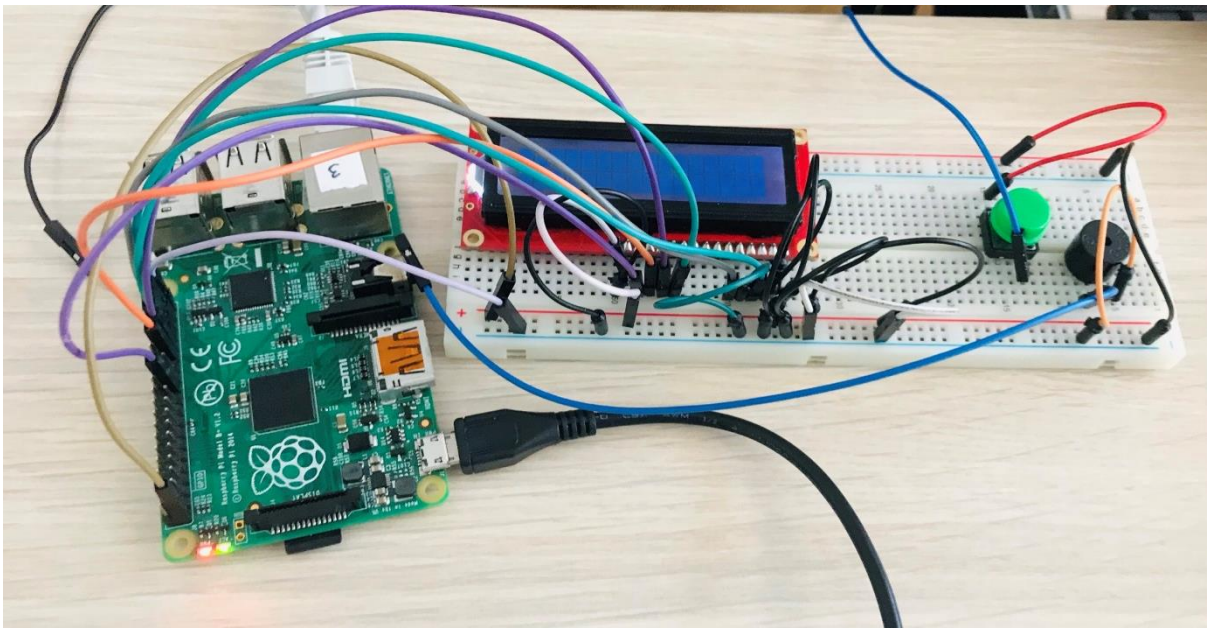
5.4. IMPLEMENTACIJA

5.4.1. Implementacija hardverskog dela projekta

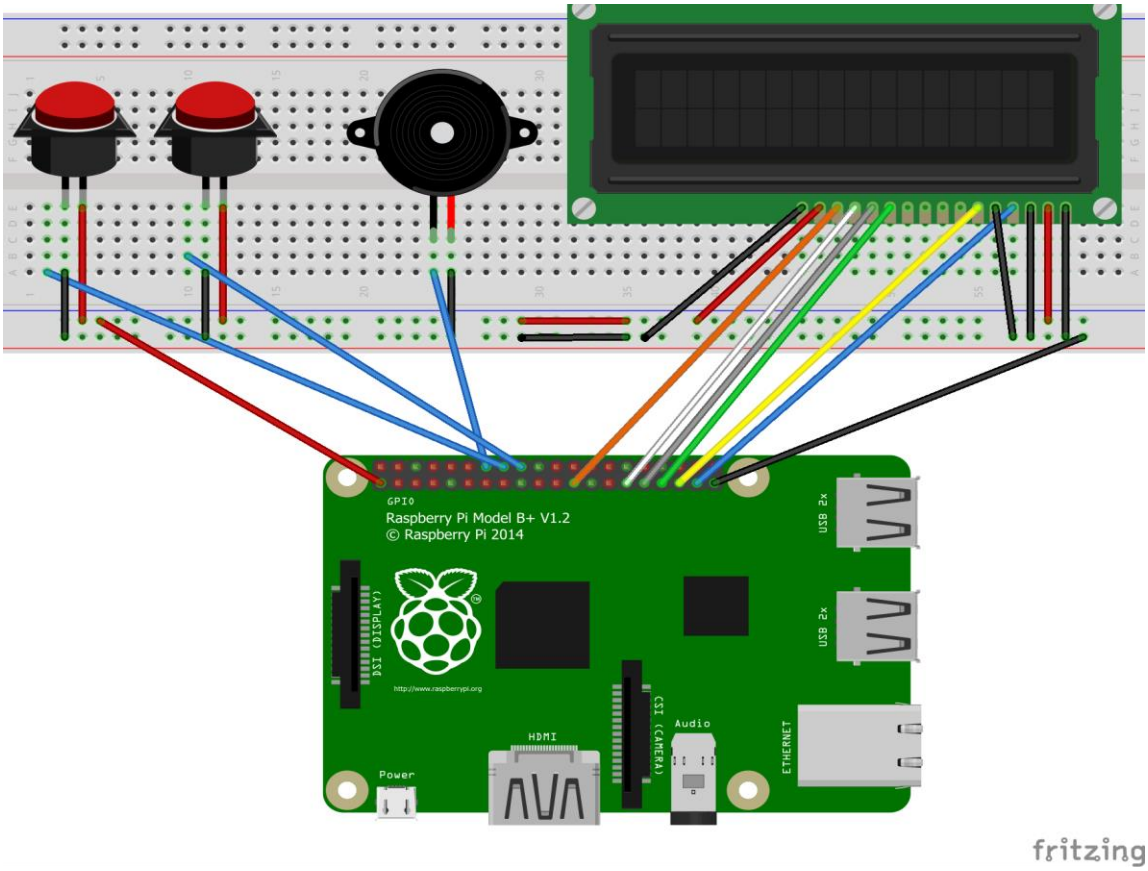
Pametni uređaji korišćeni za razvoj projekta su prikazani u sledećoj tabeli:

Naziv komponente	Izgled komponente	Opis	Količina
Raspberry Pi mikroračunar		Model 4	**1**
Dugme			**2**
LED display		Sparkfun 1602k	**1**
Buzzer			**1**
Proto ploča			**1**
Kablovi		Male - Male	**7**
Kablovi		Male - Female	**10**

Fizičko povezivanje pametnih uređaja prikazano je sledećim slikama:



Slika 30. Fizičko povezivanje pametnih uređaja



Slika 31. Fizičko povezivanje pametnih uređaja (fritzing)

5.4.2 Implementacija softverskog dela projekta

Implementacija softverskog dela projekta započije kreiranjem okruženja u kome će se projekat razvijati. Za početak kreiran je folder Project u kome je dalje razvijana naša Flask aplikacija.

Unutar foldera Project kreirano je virtelno okruženje izvršavanjem sledećih komandi u terminalu.

```
PS D:\FAX\Diplomski\Project> py -3 -m venv .venv
```

Slika 32. Kreiranje virtuelnog okruženja za projekat

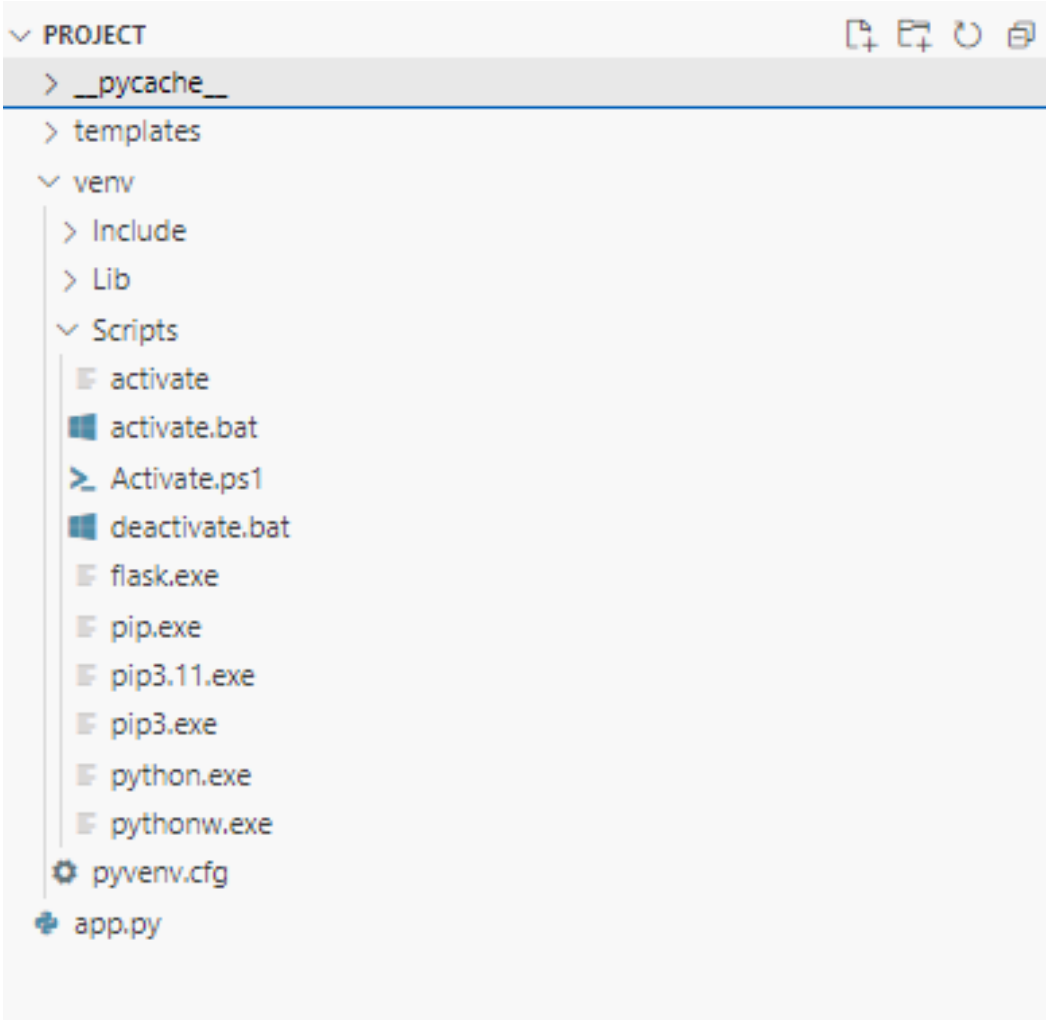
Sledeći korak je aktivacija ovog virtuelnog okruženja, kao i instalacija Flask okvira.

```
PS D:\FAX\Diplomski\Project> .venv\Scripts\activate
```

Slika 33. Aktivacija virtuelnog okruženja

```
$ pip install flask
```

Slika 34. Instalacija Flaska za projekat



Slika 33. Pregled repozitorijuma Project

U fajlu app.py definisana je naša Flask aplikacija sa svim promenljivim, funkcijama i rutama. Na početku su importovani klasa *Flask*, modul *threading* i modul *time*, kao i definisane promenljive:

- is_timer_running – definiše nam da li je tajmer pokrenut I može imati vrednosti True I False;
- start_time – definiše nam koje je je vreme određeno za tajmer;
- time_passed – definiše nam proteklo vreme
- timer_thread – nit koja se kreira pri pokretanju tajmera

app.py > ...

```python
from flask import Flask, render_template, request
import threading
import time

app = Flask(__name__)

#Globalne promenljive

is_timer_running = False
start_time = None
time_passed = 0
timer_thread = None
```

Slika 34. Kod (slika 1)

Funkcija start_timer() je zadužena za pokretanje tajmera i njegove niti. Prvo funkcija proverava da li je tajmer pokrenut, ako nije pokreće ga. Preračunava se početno vreme i menja se vrednost promenljive is_timer_running na True. Unutar ove funkcije definisana je funkcija update_timer() koje će se izvršavati sve dok se tajmer ne zaustavi. Ona će svake sekunde povećavati proteklo vreme (time_passed).

```python
def start_timer():
    global is_timer_running, start_time, timer_thread

    if not is_timer_running:
        start_time = time.time() - time_passed
        is_timer_running = True

        def update_timer():
            global time_passed, is_timer_running

            while is_timer_running:
                current_time = time.time()
                time_passed = current_time - start_time
                time.sleep(1)
                print(time_passed)

        timer_thread = threading.Thread(target=update_timer)
        timer_thread.daemon = True
        timer_thread.start()

def stop_timer():
```

Slika.35. Kod funkcije start_timer()

Na kraju kreira se nova nit korišćenjem modula threading i dodeljuje joj se funkcija update_timer. Ova nit je zaslužna za logiku tajmera koja se vrši u pozadini. Na kraju pokrećemo nit linijom timer_thread.start().

Definisane su i funkcije *start_timer(), stop_timer()* i *format_time(time).* Funkcija *stop_timer()* menja vrednost globalne promenljive *is_timer_running* na *False* u slučaju ako je ona bila *True*. Dok funkcija *format_time(time)* služi za formatiranje vremena u „mm:ss“.

```
def stop_timer():
    global is_timer_running
    if is_timer_running:
        is_timer_running = False
```

Slika 36. Kod funkcije stop_timer()

```
def format_time(time):
    minutes = int(time // 60)
    seconds = int(time % 60)

    return f"{minutes:02}:{seconds:02}"
```

Slika 37. Kod funkcije format_time(time)

Implementacija APIa se bazira na sledećim putanjama:

- „/“ – definiše se putanja početne strane naše aplikacije;
- „/*toggle_timer*“ – definiše se putanja koja odgovara samo na POST zahteve, i pozvia funkcije *start_timer()* i *stop_timer()* u zavisnosti od toga da li je isti pokrenut;
- „/*reset_timer*“ – definiše se putanja koja odgovara samo na POST zahtev i poziva funkciju *reset_timer()*;
- *„/get_time_passed“* – definiše se putanja koja vraća proteklo vreme.

```python
@app.route('/')
def index():
    return render_template('index.html', time_passed=format_time(time_passed), is_timer_running=is_timer_running)

@app.route('/get_time_passed')
def get_time_passed():
    global time_passed
    return str(format_time(time_passed))
```

Slika 38. Ruta "/"

```python
@app.route('/toggle_timer', methods=['POST'])
def toggle_timer():
    if request.method == 'POST' :
        if is_timer_running:
            stop_timer()
        else:
            start_timer()

    return '', 204
```

Slika 39. Ruta "/toggle_timer"

```python
@app.route('/reset_timer', methods=['POST'])
def reset_timer():
    global time_passed
    time_passed = 0
    stop_timer()
    return '', 204
```

Slika 40. Ruta "/reset_timer"

Pritiskom na dugme START poziva se ruta „*/toggle_timer*“ koja pokreće ili zaustavlja tajmer. Pritiskom na dugme RESET okida se ruta „*/reset_timer*“ koja resetuje vrednost tajmera na početnu.

ZAKLJUČAK

Na osnovu pregleda literature možemo uvideti da vremenom dobijamo sve veći broj sfera primene interneta inteligentnih uređaja u razvoju pametnih okruženja. Razvoj društvenih igara se kreće baš u tom smeru. Sve je više hibridnih društvenih igara koje pokušavaju da zadrže pozitivne efekte tradicionalnih igara, dok u isto vreme modernizuju načine igre i upotpunjuju iskustvo igrača primenom inteligentnih uređaja.

Kroz rad smo prikazali jedan od načina upotpunjavanja naizgled jednostavne društvene igre kao što su Asocijacije kreiranjem pametnog okruženja za istu. Ovo pametno okruženje olakšava igračima da sa svojim prijateljima uživaju u igri koju vole bez dodatnih opterećenja oko toga da li su sva pravila igre ispoštovana, za to se brine naš sistem.

Postoji veliki broj tradicionalnih društvenih igara za čije bolje iskustvo igranja je gotovo pa neophodan jedan ovakav sistem. Igre koje imaju veliki broj različitih karaktera sa svojim različitim vrednostima napada, odbrane i sl., probleme pamćenja mogu rešiti jednostavnom aplikacijom i displejom. Na osnovu pametnog okruženja razvijenog u ovom radu dobijamo mogućnost proširenja i dodavanjem novih funkcionalnosti koje mogu biti i prelazak sa klasičnih 2D tabli i svetova kreiranih unutar igre na 3D table primenom holograma i VR tehnologija.

Ovaj rad ima svoje limitacije, rešenje predstavljeno u radu prikazuje mogućnost primene pametnih sistema na primeru samo jedne igre. Za dalje istraživanje planirano je posmatranje korisnika i njihovih potreba pri korišćenju ovog rešenja, a u cilju razvijanja sistema koji bi mogao da se primeni na ostale igre sličnog tipa. Za to rešenje planirano je korišćenje najnovijih tehnologija IoT-a, dok su za ovo testno rešenje korišćene malo jednostavnije tehnologije. Takođe, ovim radom su obuhvaćeni samo osnovni scenariji korišćenja što bi isto trebalo proširiti.

REFERENCE

Anuradha, J., & Tripathy, K. (2018). *Internet of things (IoT) : technologies, applications, challenges and solutions.* Boca Raton, Florida, USA: CRC Press;Taylor & Francis.

Benford, S., Magerkurth, C., & Ljungstrand, P. (2005). Bridging the physical and digital in pervasive gaming. *Communications of the ACM, 48*(3), 54.

BoardGameGeek. (2023). *Operation*. Preuzeto sa BoardGameGeek: https://boardgamegeek.com/boardgame/3737/operation

BoardGameGeek. (2023). *Unlock! Escape Adventures*. Preuzeto sa BoardGameGeek: https://boardgamegeek.com/boardgame/213460/unlock-escape-adventures

Brownlee, J. (2022). *Concurrent programming in Python*. Preuzeto sa SuperFastPython: https://superfastpython.com/concurrent-programming/

Can You See Me Now? (2013). Preuzeto sa Blast Theory: https://www.blasttheory.co.uk/projects/can-you-see-me-now/

Concurrency in Python - Threads. (2023). Preuzeto sa TutorialsPoint: https://www.tutorialspoint.com/concurrency_in_python/concurrency_in_python_threads.htm

David. (2023). *Why Using IoT with Python is a Great Idea*. Preuzeto sa Deep Sea Developments: https://www.deepseadev.com/en/blog/iot-with-python-reasons-to-use-it/#:~:text=The%20last%20reason%20to%20use,from%20IoT%20sensors%20and%20devices.

DevicePlus. (2023). *The History of Raspberry Pi*. Preuzeto sa DevicePlus: https://www.deviceplus.com/raspberry-pi/the-history-of-raspberry-pi/

Exploding Topics. (2023). *Time Spent Using Smartphones (2023 Statistics)*. Preuzeto sa Exploding Topics: https://explodingtopics.com/blog/smartphone-usage-stats

Forbes, E. (2017). *Learning Concurrency in Python.* Packt Publishing Ltd.

Gariepy, A. (2023). *Unlock! Escape Adventures Game Review*. Preuzeto sa meeple mountain: https://www.meeplemountain.com/reviews/unlock-escape-adventures/

Halegoua, G. R. (2020). *Smart Cities.* London, England: The MIT Press.

Haller, M., Forlines, C., Koeffel, C., Leitner, J., & Shen, C. (2010). Tabletop Games: Platforms, Experimental Games, and Design Recommendations. U A. D. Cheok, *Art and Technology of Entertainment Computing and Communication* (str. 271-297). London: Springer-Verlag.

Hassan, Q. F., Khan, A. u., & Madani, S. A. (2018). *Internet of Things: Challenges, Advances, and Applications.* Chapman and Hall/CRC;Taylor & Francis.

How the Internet of Things Impacts Game Developmant. (2023). Preuzeto sa Juegostudioe Web site: https://www.juegostudio.com/blog/how-the-internet-of-things-impacts-game-development

How to Play Jackbox Games. (2023). Preuzeto sa Jackbox Games: https://www.jackboxgames.com/how-to-play/

Ikodinović, I., & Jovanović, Z. (2004). Konkurentno programiranje. Beograd: Akademska misao.

Isailović, S., Stanković, M., Vasiljević, D., & Ristić, L. (2021). *Osnove distribuiranih sistema i distribuiranog programiranja.*

Joy, A. (2023). *What Can You Do With Flask? (Applications & Examples)*. Preuzeto sa Pythonista Planet: https://pythonistaplanet.com/what-can-you-do-with-flask/

Kshetri, N., Alcantara, L. L., & Park, Y. (2014). Development of a Smart City and its Adoption and Acceptance: The Case of New Songd. *COMMUNICATIONS & STRATEGIES*(96), 113. Preuzeto sa https://ssrn.com/abstract=2636412

Magerkurth, C., Cheok, A. D., Mandryk, R. L., & Nilsen, T. (2005). Pervasive Games: Bringing Computer Netertainment Back to the Real World. *Computers in Entertainment, 3*(3).

Magerkurth, C., Memisogul, M., Engelke, T., & Streitz, N. (2004). Towards the Next Generation of Tabletop Gaming Experiences. *Graphics Interface* (str. 73-80). London: AK Peters.

Mavromoustakis, C. X., Mastorakis, G., & Batalla, J. M. (2016). *Internet of Things (IoT) in 5G Mobile Technologies* (T. 8). Springer International Publishing.

Mufid, M. R., Basofi, A., M., H. U., Rochimansyah, I. F., & Rokhim, A. (2019). Design an MVC Model using Python for Flask Framework Development. *International Electronics Symposium.*

Muskan. (2021). *4 Applications of IoT in Gaminng Industry*. Preuzeto sa Analytics Steps: https://www.analyticssteps.com/blogs/4-applications-iot-gaming-industry

Nilay, D. (2023). *Developing IoT Software Solution Using Raspberry Pi: What It Takes*. Preuzeto sa Intuz: https://www.intuz.com/blog/raspberrypi-for-iot-software-development

Palach, J. (2014). *Parallel programming in Python.* Packt Publishing Ltd.

Pallets. (2023). *Flask*. Preuzeto sa Installation: https://flask.palletsprojects.com/en/3.0.x/installation/

Pallets. (2023). *Project Layout*. Preuzeto sa Flask: https://flask.palletsprojects.com/en/3.0.x/tutorial/layout/

Posey, B. (2022). *IoT devices (internet of things devices)*. Preuzeto sa TechTarget Network: https://www.techtarget.com/iotagenda/definition/IoT-device

Python Software Foundation. (n.d.). *Applications for Python*. Preuzeto sa Python: https://www.python.org/about/apps/

Python Software Foundation. (n.d.). *Python*. Preuzeto sa threading: https://docs.python.org/3/library/threading.html

Python Software Foundation. (n.d.). *What is Python? Executive Summary*. Preuzeto sa Python: https://www.python.org/doc/essays/blurb/

Radenković, B., Despotović-Zrakić, M., Bogdanović, Z., Barać, D., Labus, A., & Bojović, Ž. (2017). *Internet Inteligentnih uređaja.* Beograd: Fakultet organizacionih nauka.

Radenković, B., Despotović-Zrakić, M., Bogdanović, Z., Barać, D., Labus, A., & Bojović, Ž. (2017). *Internet inteligentnih uređaja=Internet of things.* Beograd: Fakultet organizacionih nauka.

Raspberry Pi Spy. (2023). *Simple Guide to the Raspberry Pi GPIO Header*. Preuzeto sa Raspberry Pi Spy: https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/

Raynal, M. (2012). *Concurrent Programming: Algorithms, Principles and Foundations.* Springer Science & Business Media.

Richardson, M., & Wallace, S. (2013). *Getting Started with Raspberry Pi.* Maker Media.

Schneider, F. B. (1997). On Concurrent Programming. New York: Springer.

Shafique, K., Khawaja, B. A., Sabir, F., Qazi, S., & Mustaqim, M. (2020). Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios. *IEEE Access, 8*, 23022-23040.

Solaimani, S., Keijzer-Broers , W., & Bouwman , H. (2013). What we do – and don't – know about the Smart Home: an analysis of the Smart Home literature. *Indoor and Built Environment*, 1-14.

Somani, A. K., Shekhawat, R. S., Mundra, A., Srivastava, S., & Verma, V. K. (2020). *Smart Systems and IoT: Innovations in Computing: Proceeding of SSIC 2019* (1st izd., T. 141). Sydney: Springer Singapore.

Statista. (2023). *Board Games - Worldwide*. Preuzeto sa Statista: https://www.statista.com/outlook/dmo/app/games/board-games/worldwide

The Robotics Back-End. (2023). *Raspberry Pi 4 Pins – Complete Practical Guide*. Preuzeto sa The Robotics Back-End: https://roboticsbackend.com/raspberry-pi-3-pins/

Venners, B. (2003). *The Making of Python: A Conversation with Guido van Rossum, Part I*. Preuzeto sa Artima: https://www.artima.com/articles/the-making-of-python

Zhao, C. W., Jegatheesan, J., & Loon, S. C. (2015). Exploring IOT Application Using Raspberry Pi. *International Journal of Computer Networks and Applications, 2*(1).