SVEUČILIŠTE U ZAGREBU
RUDARSKO-GEOLOŠKO-NAFTNI FAKULTET

Jona Brozović; Bruno Ćesić; Karlo Weiser

RAZVOJ UREĐAJA ZA PODZEMNU NAVIGACIJU PRI SPAŠAVANJU (DEVELOPMENT OF SUBTERRANEAN GUIDANCE SYSTEM - SGS)

Zagreb, 2020.

Ovaj rad izrađen je u Zavodu za rudarstvo i geotehniku, Rudarsko-geološko-naftnog fakulteta u Zagrebu, pod vodstvom izv. prof. dr. sc. Dalibora Kuhineka i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2019./20.

POPIS KRATICA

SGS – uređaj za podzemnu navigaciju pri spašavanju (*Subterranean Guidance System*)

GIS – Geografski informacijski sustav

GPIO – Ulaz/Izlaz opće namjene (*General Purpose Input/Output*)

MSHA – regulatorno tijelo Sjedinjenih Američkih Država za zaštitu na radu u industriji rudarstva (*Mine Safety and Health Administration*)

MINERS – međunarodni projekt financiran of Europskog Instituta za tehnologiju (EIT) za studente rudarstva (*Mine Emergency Response And Rescue School*)

DA – Dijkstrin algoritam

IT – Informatička tehnologija

LF – niska frekvencija (*low frequency*)

HF – visoka frekvencija (*high frequency*)

UHF – ultra visoka frekvencija (*ultra high frequency*)

MDS – višedimenzionalno skaliranje (*multidimensional scaling*)

SBL – naziv algoritma za segmentaciju podataka (*Sparse Bayesian Learning*)

GPS – globalni pozicijski sustav

CAD – dizajn s pomoću računala (*Computer Aided Design*)

VRML – jezik modeliranja virtualne stvarnosti (*Virtual Reality Modeling Language*)

NPS – Nacionalna parkovna agencija, SAD (*National Park Service*)

POPIS MJERNIH JEDINICA

g – gram

m – metar

kHz – kiloherc

MHz – megaherc

GHz – gigaherc

Mb/s – megabit po sekundi

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

1. UVOD

Podzemne prostorije koriste se od ranih početaka ljudske rase. Prvenstveno su služile kao skloništa i domovi, a kasnije i za eksploataciju mineralnih sirovina. Razvojem civilizacije rasle su potrebe za sve većim količinama mineralnih sirovina što je primoralo ljude da počnu s eksploatacijom na većim dubinama. Međutim, eksploatacija na većim dubinama nije jednostavan pothvat, te su kroz povijest mnoge tragične nesreće to i dokazale. Bila su potrebna stoljeća neuspjelih pokušaja, nesretnih slučajeva, studija i pokusa da bi se došlo do razine sigurnosti koja je poznata u 21. stoljeću. Još u drugom dijelu 20. stoljeća su se događale nesreće u podzemnim prostorijama, ponajprije rudnicima s desecima, a ponekad i stotinama smrtnih slučajeva. Upravo takve nesreće su potaknule razvoj tehnike sigurnosti s ciljem određivanja mjera za sprječavanje nesreća, te obukom osoblja u rudnicima kako bi postali članovi spasilačkih službi ukoliko sigurnosne mjere zakažu.

Tema ovoga rada je prikazati svojstva izrađenog prototipa uređaja SGS (*eng. Subterranean Guidance System – SGS)*, koji bi mogao postati dio osnovne opreme spasilačke službe u podzemnim prostorijama. Opisan je način rada, područje primjene, postupak izrade i razvoja softvera koji uređaj koristi, te ideje za daljnji razvoj.

Svrha SGS-a je jednostavna i brza navigacija kroz podzemne prostorije za sve članove spasilačke službe, neovisno o njihovom prethodnom poznavanju tlocrta podzemnog objekta i nepredvidivim uvjetima koji se događaju u takvim situacijama.

2. NESREĆE U RUDARSTVU I SPAŠAVANJE

Rudarstvo u konvencionalnom značenju riječi se bilježi još od starog Egipta, međutim tek je 1850. godine parlament Ujedinjenog Kraljevstva donio zakon o inspekcijama rudnika ugljena (eng. *Act for Inspection of Coal Mines*) u Velikoj Britaniji. Od 1855. do 1860. godine ovlasti zakona bile su proširene da obuhvaćaju i željeznu rudu. Tek iza 1870. godine zakon se proširio na sve vrste eksploatacije, a do početka 20. stoljeća u zakonu su bile obuhvaćene sve grane industrije vezane za rudarstvo, uključujući i površinsku eksploataciju (Mills, 2010). U Sjedinjenim Američkim Državama 1896. godine osnovan je Savezni zavod za rudnike (*Federal Department of Mines*) kojem je uloga bila reguliranje tematike sigurnosti u rudnicima (NPS, 2015). 1910. godine osnovan je Ured za rudnike (*eng. Bureau of Mines*) kao nova agencija čija je uloga bila istraživati i bilježiti nesreće koje su se događale u rudnicima ugljena te koristiti skupljene informacije za povećanje sigurnosti rada u rudnicima. Tek 1941. godine je Ured dobio ovlasti za provođenje inspekcija unutar rudnika, te je na temelju istraživanja i inspekcija 1969. donesen Akt o sigurnosti rudnika ugljena (Dhillon, 2010). Taj niz događanja započeo je modernu tehniku sigurnosti vezanu za rudnike i rudarsku djelatnost.

Nesreće u rudarstvu su do početka 21. stoljeća bile učestale i često smrtonosne. Najsmrtonosnija nesreća bila je 1942. godine u rudniku ugljena kod kineskog jezera Benxi, koji se nalazi u provinciji Liaoning. 26. travnja 1942. godine eksplozija plinova i ugljene prašine usmrtila je 1549 radnika (Dhillon, 2010). Nesreće tog tipa, pretežito u rudnicima ugljena bile su česta pojava. Upravo je visoka smrtnost zbog loših radnih uvjeta, nepridržavanja propisanih pravila i zakona potaknula je, drugom polovicom 20. stoljeća, da se intenzivnije počne raditi na uspostavljanju udruga i centara za obuku spasilačkih ekipa. Također, došlo je i do postroženja zakona o sigurnosti na radu i sve rigoroznijih inspekcija.

Spašavanje u rudnicima uključuje širok spektar zadataka, obzirom da pristup pri spašavanju ovisi o situaciji i razlogu intervencije, te vrsti rudnika u kojem se situacija odvija. Također, bitan faktor koji određuje pristup spasilačke grupe su i sami uvjeti u rudniku. Ovisno o situaciji i uvjetima, MSHA definirala je zadatke spasilačkih postrojbi tijekom hitnih slučajeva i nesreća u rudnicima:

- istražiti dijelove rudnika koji su pod utjecajem nesreće
- tražiti i spasiti preživjele
- pružati prvu pomoć
- procjena oštećenja
- mjerenje koncentracije plinova u zraku
- zabilježiti bitne podatke na karti rudnika
- locirati i ugasiti/spriječiti širenje požara
- izgraditi privremene ili trajne brane i/ili pregrade
- zagraditi dijelove rudnika koji su pod utjecajem požara
- pročistiti krhotine, ispumpati vodu i postaviti privremene krovne podgrade radi povećanja stabilnosti

Generalno, spasilačka postrojba sastoji se od nekoliko spasilačkih grupa (dvije do tri), čiji broj najviše ovisi o veličini rudnika i broju zaposlenih koji zadovoljavaju zdravstvene kriterije za ostvarenje članstva. Spasilačke grupe imaju od tri do šest članova, gdje je jedan član voditelj grupe. Funkcije jedne spasilačke grupe bazirane na šest članova su (izvor):

- voditelj, odnosno kapetan grupe koji donosi odluke
- osoba koja mjeri koncentraciju plinova, te je odmah iza kapetana
- osoba s kartom koja bilježi zapažanja i radnje koje je grupa odradila
- osoba koja vuče nosila
- osoba koja hoda zadnja u redu, najčešće do-kapetan koji prima zadatke i naredbe iz baze na površini, te je zadužena za komunikaciju između grupe i baze
- osoba koja se nalazi na površini u bazi za svježi zrak, te je zadužena za komunikaciju između grupe u rudniku i komandnog centra

Svi članovi moraju biti u potpunosti kvalificirani, uvježbani i opremljeni za pružanje pomoći pri spašavanju. Svi članovi su uglavnom zaposlenici tog rudnika, te su dio spasilačke grupe tek kao sekundarna funkcija (Kilinc, Monghan, Powell, 2014).

U nekim zemljama, primjerice u Austriji, spasilačke postrojbe su organizirane prema sustavu da za nekoliko obližnjih rudnika imaju zajedničke spasilačke grupe. Glavni razlozi zajedničkih postrojbi su veličina rudnika, neposredna blizina i poznavanje prostora (MINERS Handbook, 2018). Poznavanje prostora vjerojatno je najbitniji čimbenik jer ako spasioci nisu upoznati s karakteristikama rudnika ne mogu garantirati prvenstveno svoju sigurnost, a samim time ni sigurnost osoba koje spašavaju. Te karakteristike značajno variraju i gotovo nikada se ne preklapaju čak ni s obližnjim rudnicima. Iz tog razloga spasioci moraju uz obaveznu obuku biti i dobro upoznati s rudnicima za koje su zaduženi.

Oprema koja se koristi pri spašavanju je raznovrsna, te ovisi o situaciji i potrebi spašavanja. Osnovna oprema uključuje standardnu rudarsku zaštitnu kacigu, rudarsku lampu, kombinezon, izolacijski samospasioc, kartu rudnika, opremu za daljinsku komunikaciju (najčešće žična), radne rukavice, čizme sa zaštitom za prste i zglobove te štitnici za koljena (Kilinc, Monghan, Powell, 2014). Na slici 2-1 prikazani su studenti rudarskog inženjerstva s Montanuniversität Leoben i Rudarsko-Geološko-Naftnog fakulteta u Zagrebu u opremi za spašavanje u sklopu međunarodnog projekta MINERS.

Slika 2-1 Studenti rudarskog inženjerstva u opremi za spašavanje, projekt MINERS.

3. RAZVOJ UREĐAJA

Nakon razrade zamisli svojstava i funkcionalnosti te sučelja uređaja, obavljena je analiza postojećih tehnologija i algoritama koji se mogu koristiti za izradu prototipa. Odabran je optimalan algoritam te je započet rad na izradi prototipa i nakon toga niz pokušaja programiranja i testiranja uređaja dok se nije dobila stabilna i zadovoljavajuća funkcija prototipa.

3.1. Opis željenih funkcionalnosti uređaja

Subterranean guidance system (SGS) je uređaj koji bi trebao zamijeniti konvencionalne načine orijentacije i navigacije pri spašavanju u podzemnim objektima. Osnovna primjena zamišljena je za rudnike, ali nije i ograničena samo za njih.

Cilj SGS-a je omogućiti navigaciju kroz nepoznate podzemne prostore, te praćenje lokacije spasilačke grupe u stvarnom vremenu bez ovisnosti o vanjskim i unutrašnjim čimbenicima. SGS može zamijeniti karte i slične metode navigacije u potpunosti. Također, mogao bi u kasnijim stadijima razvoja ukloniti čimbenik poznavanja podzemnog objekta, te omogućiti da se zapažanja tijekom spašavanja mogu zapisati u uređaju na interaktivnoj digitalnoj karti. U konačnoj inačici SGS-a sve potrebne informacije i karakteristike moći će se pohraniti u uređaju, te prikazati u sučelju po potrebi.

3.2. Dijkstrin algoritam u SGS-u

Dijkstrin algoritam (DA) je jedan od najpopularnijih algoritama u računalnim znanostima i operativnom istraživanju. 1959. godine Edsger W. Dijkstra objavio je rad u časopisu „*Numerische Mathematik*“. U tom radu Dijsktra je predložio algoritme za dva teoretska problema grafova: *problem minimalnog razapinjajućeg stabla* i *problem najkraćeg puta*. Upravo je Dijkstrin algoritam za problem najkraćeg puta jedan od najprepoznatijih algoritama u računalnim znanostima i operativnom istraživanju. Glavni razlog popularnosti Dijkstrinog algoritma je taj da ga se smatra najbitnijim i najkorisnijim dostupnim algoritmom za generiranje

optimalnih rješenja za veliku skupinu problema najkraćeg puta. Tradicionalne skupine problema najkraćeg puta su područja računalnih znanosti, umjetne inteligencije, operativnih znanosti i upravljačkih znanosti. Navedena područja smatraju se problemima najkraćeg puta jer se svaki problem kombinatoričke optimizacije može definirati kao problem najkraćeg puta. Međutim, nove skupine problema zadnjih godina dobivaju na značaju zbog praktične povezanosti ove problematike i primjene na Geografskim Informacijskim Sustavima (GIS). Primjer takve primjene je *online* proračunavanje uputa u stvarnom vremenu tijekom vožnje.

Princip rada DA je da na zadanom grafu s poznatim točkama odredi najkraći put od zadane početne točke A do zadane konačne točke B. Poznate točke moraju biti definirane preko međusobne udaljenosti ukoliko između njih postoji put ili preko koordinata na apscisi i ordinati. Preko poznatih koordinata se može proračunati udaljenost točaka između kojih postoji put. Te udaljenosti mogu biti unaprijed proračunate ili izmjerene, a zatim i zapisane. Druga mogućnost je da se osigura način da s unosom koordinata točaka bude automatski proračunata vrijednost udaljenosti i zapisana. Algoritam prolazi sve točke na grafu, zbraja njihove međusobne udaljenosti, te potom razvrstava dobivene udaljenosti od najmanje do najveće. Kada se analiziraju sve vrijednosti, iskazuje se onaj put čija suma ima najmanji iznos (Sniedovich, 2006). Dijkstrin algoritam je prepoznat kao idealno rješenje i podloga za izradu SGS-a, obzirom da na jednostavan i pouzdan način pronalazi najkraći put, što je osnovna ideja principa rada SGS-a. Dijkstrin algoritam koji SGS koristi nalazi se u prilogu 3.

DA ima dva glavna ograničenja. Prvo ograničenje je činjenica da svaki put pri pokretanju pretražuje sve čvorove. Za manje rudnike to nije problem, s obzirom da nemaju veliki broj čvorova i hodnika. Međutim, pri većem broju čvorova i hodnika pokretanje DA je zahtjevno s gledišta procesorske snage. Drugo ograničenje je nemogućnost uključivanja negativnih vrijednosti u algoritam. Ovo ograničenje nije presudno za rad SGS-a, ali otežava eventualna poboljšanja dodavanjem negativnih vrijednosti pojedinim točkama ili hodnicima za precizniju navigaciju i/ili neku drugu funkciju samog uređaja. (Rehman i dr., 2019)

3.3. Logički princip rada uređaja

Kostur programa je Dijkstrin algoritam. Sve točke, odnosno čvorišta rudnika se moraju unaprijed unijeti u program putem koordinata u Kartezijevom koordinatnom sustavu. Oblik u kojem se koordinate točaka zapisuju je N-terac. N-terci su sljedovi vrijednosti koje mogu biti bilo koje vrste. Vrijednosti se razdvajaju, odnosno označavaju cijelim brojevima i prema tome su slični listama. N-terci se razlikuju od lista po tome što su nepromjenjivi (Downey, 2015).

Primjer N-terca koji se koristi u SGS-u:

'1' = ('78','jabuka')

Pomoću gore napisanog N-terca pod imenom *1* definirala se duljina hodnika *Jabuka*. Skup više takvih N-teraca koji definiraju točke rudnika nalaze se u rječniku koji će definirati rudnik za simulaciju u poglavlju 5. Takav rječnik koji sadrži više vrijednosti za jedan ključ naziva se multirječnik (Beazley i Jones, 2013). Svaki zapis u rječniku Pythona može se podijeliti na dva dijela:

ključ : vrijednost

Ključ je dio zapisa preko kojega se dohvaća njemu zadana vrijednost (Hruška, 2018).

Odabirom i potvrdom početne i krajnje točke na prozoru sa slike 4- program napravi nekoliko radnji:

- pohranjuje početnu točku u zasebnu varijablu za kasnije korištenje
- poziva DA da pronađe najkraći put i pohranjuje ga u novu varijablu
- uzima prvu točku i prvu slijedeću točku koja se nalazi na najkraćem putu, te poziva drugi algoritam koji generira dio karte s te dvije točke
- sprema generiranu kartu

Algoritam za generiranje karte uzima točke, pronalazi ih na x-y grafu i pridodaje im imena. Zatim algoritam uzima koordinate točaka, pridodaje im vrijednost od +0,2 ili –0,2 ovisno o poziciji u rudniku. Broj točaka koje se koriste pri generiranju ovisi o značaju za segment karte koji generira. Razlog promjene vrijednosti je preglednost generirane karte. Generirana karta se sprema u .jpg formatu, te ju onda program učitava i prikazuje u radnom prozoru.

Po završetku svih gore navedenih radnji korisniku se prikaže drugi prozor s generiranom kartom i uputama, Generirana karta prikazuje točku s koje je korisnik došao, točku na kojoj se nalazi i točku prema kojoj treba krenuti. Treba uzeti u obzir da je generirana karta sekundarni mehanizam orijentacije, a kao primarni mehanizam služi tekstualni dio ispod karte. Tekstualni dio opisuje korisniku sve informacije vidljive na karti, te mu daje uputu prema kojem hodniku treba krenuti uz naziv tog hodnika. Nazivi hodnika nisu nepoznanica u praksi, posebice ako se radi o većem rudniku gdje se to provodi radi jednostavnosti. Primjer teksta:

Udaljenost koju trebate prijeći je 384 m.
Prošla točka je 1. Nalazite se na točki 2.
Krenite hodnikom Borovnica do točke 11.

Korisnik na ovome prozoru može obavijestiti program kako:

a) je moguće proći kroz navedeni hodnik – gumb „*Hodnik je prohodan*“
b) nije moguće proći kroz navedeni hodnik – gumb „*Hodnik ima prepreku*“

Odabirom opcije a) program uzima prošlu točku, točku na kojoj se korisnik nalazi i slijedeću točku i ponovo poziva algoritam za generiranje karte. Generirana karta ovaj put prikazuje tri točke. Korisniku se prikazuje radni prozor s novom generiranom kartom, novim uputama i informacijama u tekstualnom obliku, te opet ima iste dvije opcije.

Odabirom opcije b) program ponovo poziva DA radi pronalaska novog najkraćeg puta uzimajući u obzir raniju bilješku da je put prepriječen. Novi najkraći put se pohranjuje, te se ponovo poziva algoritam za generiranje karte.

Cijeli ciklus s ove dvije opcije se ponavlja sve dok:

1) korisnik nije došao do zadane krajnje točke
2) nisu svi hodnici do zadane krajnje točke prepriječeni

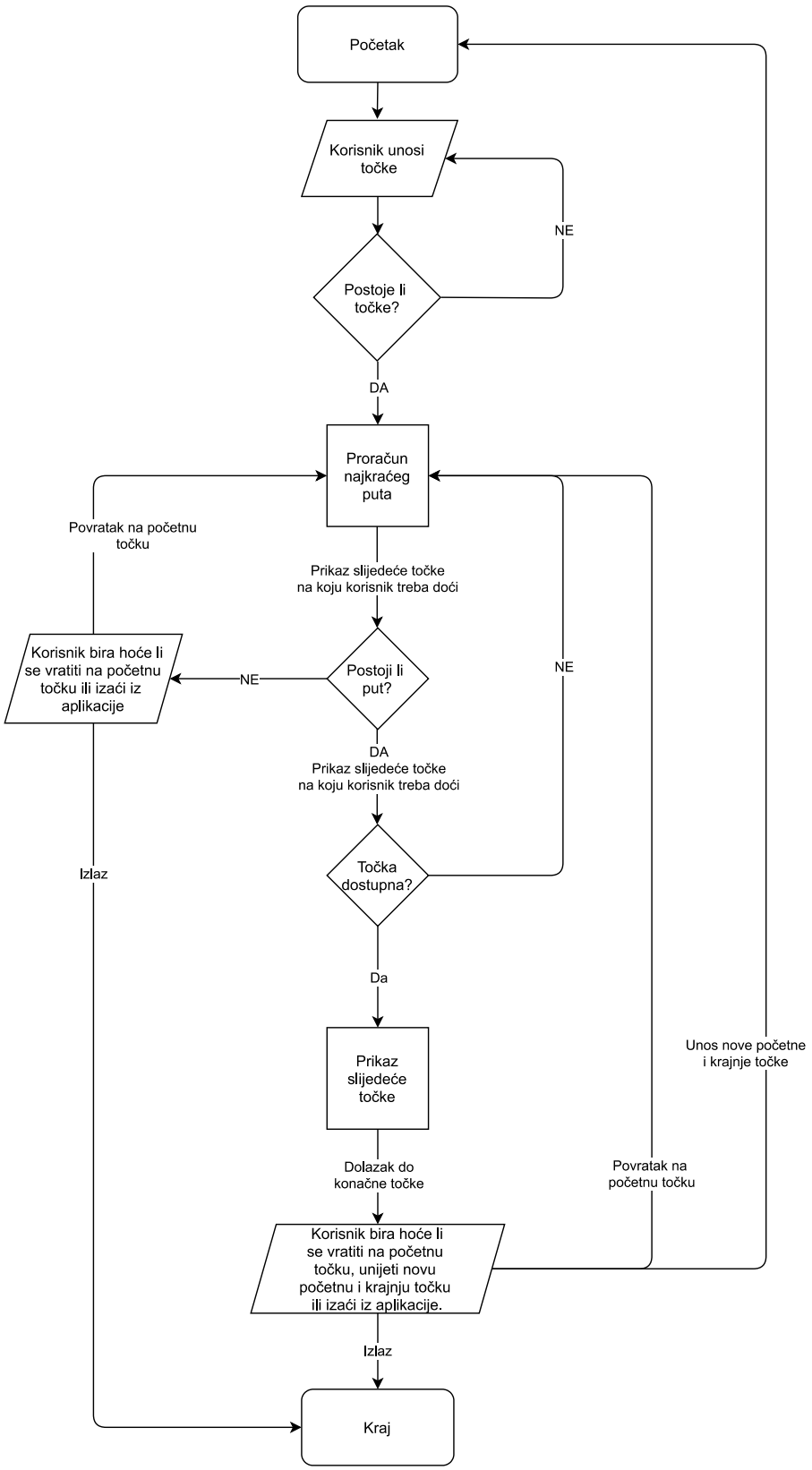
Ako je korisnik došao do zadnje krajnje točke (1), korisniku su ponuđene dvije opcije:

a) povratak na početnu točku (primjerice ulaz rudnika)
b) zatvoriti program

U slučaju opcije a) program očitava početnu točku iz varijable koja je spremljena pri početku rada programa i pamti ju kao krajnju. Točka na kojoj se korisnik nalazi uzima se kao početna, te se cijeli program pokreće ispočetka prema gore opisanom ciklusu. Povratkom na početnu točku korisnik može pokrenuti novi ciklus pritiskom na gumb „*Novi zadatak*“ ili zatvoriti program pritiskom na tipku „*Izlaz*“.

Opcija b) zatvara program i gasi uređaj.

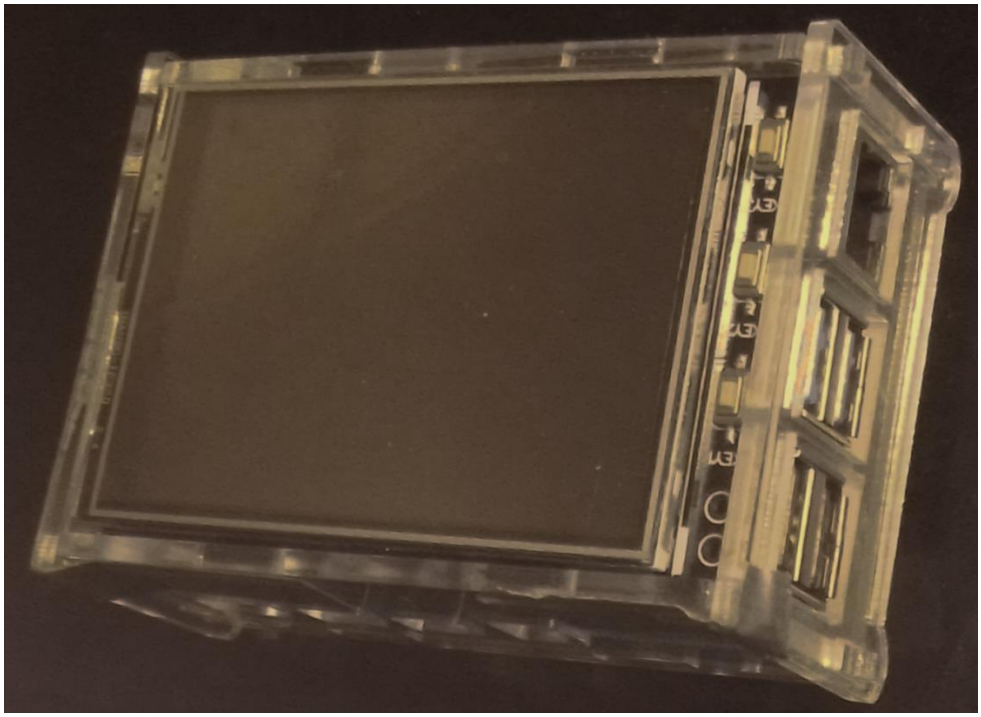
Ako su svi putevi prepriječeni (2), korisnik ima opcije vratiti se na početnu točku ili ugasiti uređaj. Početna točka tretira se kao krajnja, a trenutna lokacija (zadnja točka do koje je korisnik uspješno došao) učitava se kao početna. S tim ulaznim podatcima pokreće se ranije opisan ciklus. Slika 3-1 prikazuje dijagram tijeka rada programa SGS.



Slika 3-1 Logička shema programa SGS-a

4. IZVEDBA PROTOTIPA

Radi optimizacije troškova, lakše dostupnosti nekih komponenti te šire podrške za hardverske i softverske komponente budućeg uređaja odlučeno je koristiti mikroračunalo Raspberry Pi i programski jezik Python 3.7.0. Nabavljene su potrebne komponente i na slici 4-1 prikazan je sklopljen prototip SGS-a koji je korišten tijekom izrade programa i testiranja.

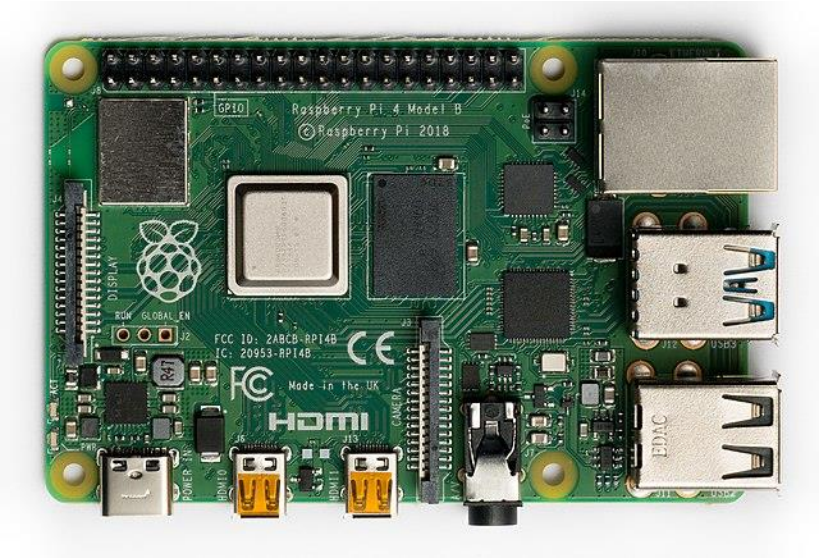


Slika 4-1 Slika hardverskih komponenti uređaja SGS.

4.1. Hardver uređaja

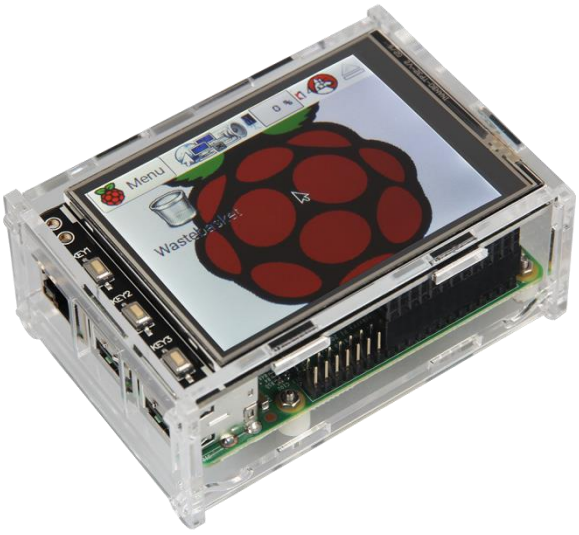
Odabrani hardware za prvu inačicu SGS-a je Raspberry Pi 4 model B koji je prikazan na slici 4-2. Raspberry Pi je serija jednopločnih računala proizvedenih u Ujedinjenom Kraljevstvu čija je primarna namjena omogućiti jeftino i pouzdano računalo za učenje osnova računalnih znanosti. Prednost korištenja Raspberry-a za prototip SGS-a je njegov operativni sustav Raspberry Pi OS (tzv. Raspbian) koji je baziran na otvorenom Linux OS-u. Raspbian dolazi s već unaprijed

instaliranim programskim jezikom Pythonom što je znatno olakšalo povezivanje programa SGS-s s Raspberry-em (Raspberry Pi Foundation, 2020).



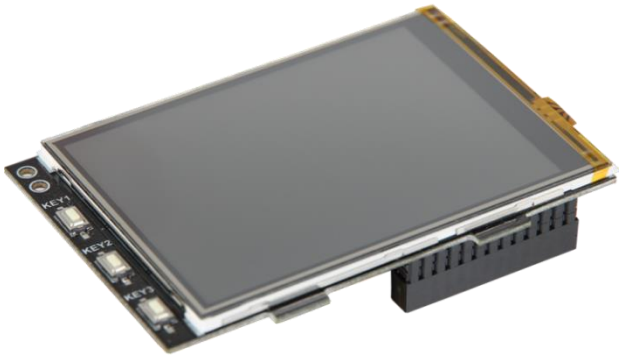
Slika 4-2 Raspberry Pi 4 B (Wikipedia, 2019)

Korišteno je akrilno kućište tvrtke Joy-IT, prikazano na slici 4-3. Tvrtka Joy-IT izradila je ovaj tip kućišta specifično za Raspberry modele B+, 2B, 3B, 3B+ i 4, te se koristi u kombinaciji s njihovim zaslonom na dodir od 3,2 inča ili 3,5 inča. Po sastavu je od transparentnog akrila visoke kvalitete. S obzirom da različiti modeli Raspberry-a imaju drugačije dimenzije, kućište se može podesiti dodatcima koji dolaze u kompletu s kućištem. Masa kućišta je 70 g (Joy-IT, 2020).



Slika 4-3 Joy-IT Akrilno kućište (Joy-IT, 2020)

Uređaj koristi zaslon na dodir prikazan na slici 4-4 koji je proizvela tvrtka Joy-IT. Dijagonala zaslona je 3,2 inča s maksimalnom rezolucijom od 320x240 pixela. Spaja ga se s Raspberry PI-em putem standardnih GPIO utora. Zaslon sadrži tri gumba koji se mogu programirati. Ovaj zaslon je kompatibilan s Raspberry Pi A, B, B+, 2B, 3B, 3B+ i 4B isto kao i kućište, što ih čini potpuno međusobno kompatibilnim za povezivanje software-a SGS-a na bilo koju drugu inačicu Raspberry Pi operativnog sustava. Masa zaslona je 52 g (Joy-IT, 2020).



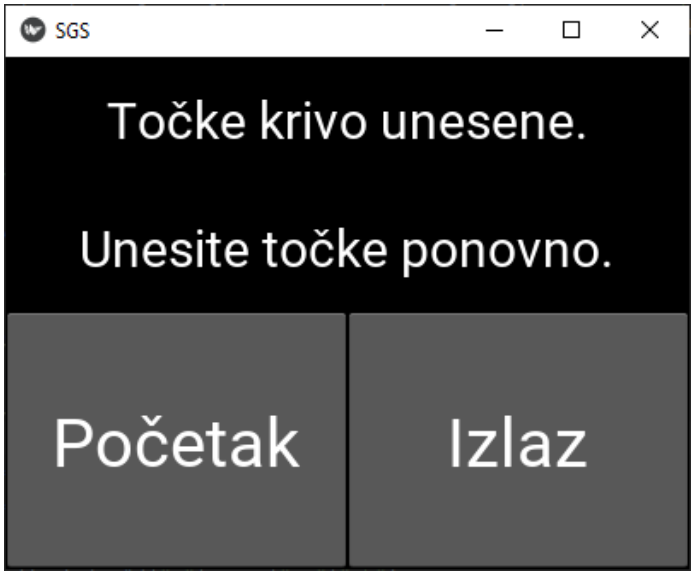
Slika 4-4 Joy-IT Zaslon na dodir (Joy-IT, 2020.)

4.2. Korisničko sučelje uređaja

Korisničko sučelje trenutne inačice SGS-a osmišljeno je tako da što jednostavnijim načinom omogući korisniku odabir željene destinacije. Izrađeno je pomoću biblioteke otvorenog koda Kivy. U okviru biblioteke Kivy postoji i Kivy jezik koji služi za odvajanje logike programa od vizualne prezentacije sučelja. Program koji pokreće SGS je pisan u programskom jeziku Python kako je ranije navedeno, a elementi grafičkog sučelja pisani su u zasebnoj datoteci s ekstenzijom „.kv“, odnosno u Kivy jeziku (Ulloa, 2013). Prilozi 1 i 4 prikazuju ispis kodova za korisničko sučelje uređaja SGS.

Odabir početne pozicije također je opcija jer ona može varirati u određenim uvjetima, primjerice ako rudnik ima više od jednog ulaza ili ako tijekom navigacije dođe do potrebe za promjenom željene destinacije. Obje opcije se biraju kroz početni prozor u sučelju. Na slici 4-5 prikazan je početni prozor SGS-a. S obzirom da je primarna navigacija kroz sučelje preko zaslona na dodir, odabir početne i krajnje točke se bira pomoću gumbi + i – , gdje + povećava broj do željene točke, a – taj broj umanjuje. Ovakav način odabira željenih točaka je odabran radi jednostavnosti i brzine odabira. Par stupaca s lijeve strane služe za određivanje početne točke, te iznad njih piše tekst „*Start*“. Brojevi za točke, odnosno čvorišta na karti su unaprijed definirani i kao takvi moraju biti unaprijed uneseni u uređaj. Kada se definiraju potrebne točke, pritiskom na gumb „*Potvrdi*“ započinje postupak proračuna najkraćeg puta, te se program prebacuje na idući prozor. Ukoliko korisnik želi izaći iz SGS-a, odnosno ugasiti uređaj mora pritisnuti gumb „*Izlaz*“. Ako odabrane točke ne postoje, otvara se prozor prikazan na slici 4-6 koji obavještava korisnika o greški, te mu nudi opcije „*Unesite točke* ponovo“ i „*Izlaz*“.

Slika 4-5 Početni prozor za unos točaka



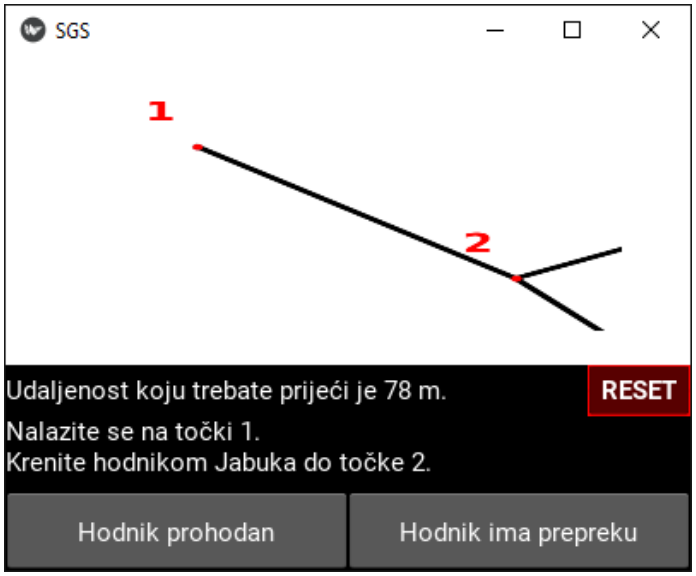
Slika 4-6 Prozor za grešku pri unosu

Nakon početnog prozora, pritiskom na „*Potvrdi*“ otvara se radni prozor i ujedno započinje prikazivanje proračunatog puta. Radni prozor prikazan je na slici 4-7. Gornji dio ovog prozora prikazuje isječak karte na kojem su prikazane točka na kojoj se korisnik nalazi i iduća točka prema kojoj korisnik ide. Ispod isječka karte je kratak tekstualni opis koji govori korisniku iz kojeg je smjera došao. Svrha tog tekstualnog opisa je lakša orijentacija. Na dnu ekrana radnog prozora nalaze se gumbi za unos stanja prohodnosti.

Trenutna inačica nudi dvije opcije:

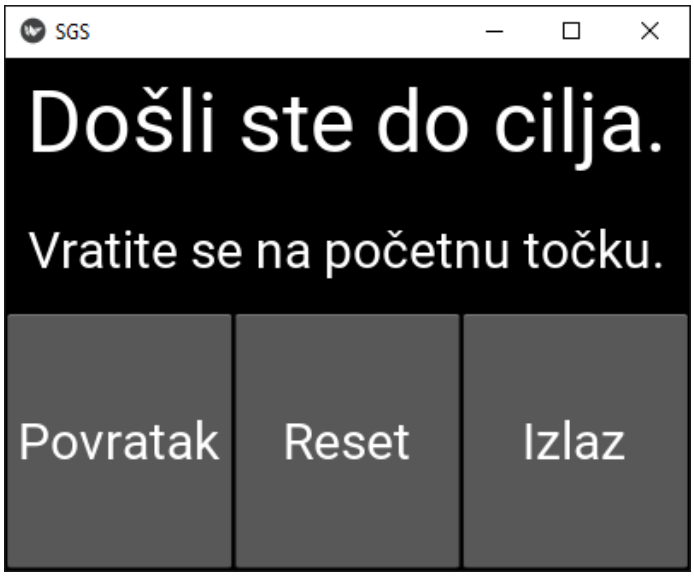
a) „*Hodnik prohodan*“
b) „*Hodnik ima prepreku*“

Ukoliko se odabere opcija a) poručuje se programu da smo prošli prikazani hodnik, te se otvara novi radni prozor s daljnjim uputama. Odabirom opcije b) poručuje se programu da trenutno proračunati najkraći put nije ostvariv. Trenutni radni prozor se zatvara, te se program vraća korak unazad i daje drugi najkraći put uzimajući u obzir da je željena konačna točka ostala ista. Gumb „*Reset*“ služi za situacije kada dođe do promjene željene krajnje točke i pritiskom na njega vraća program na početni prozor.

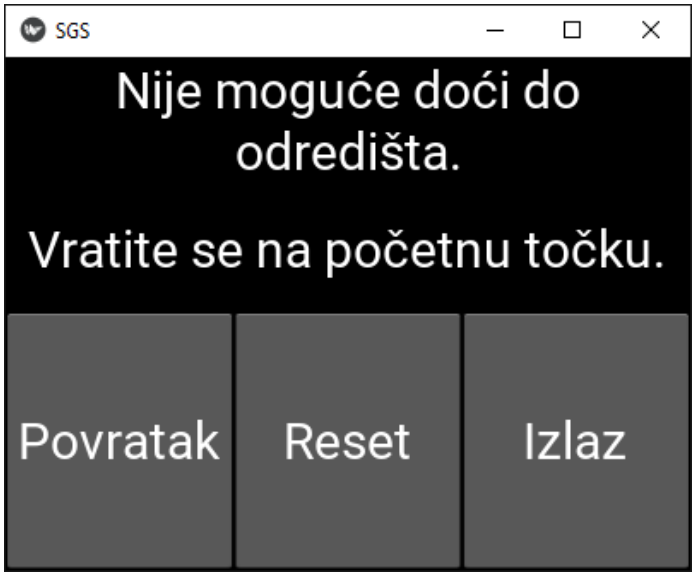


Slika 4-7 Radni prozor s uputama za navigaciju

Dolaskom na konačnu točku otvara se završni prozor koji je prikazan na slikama 4-8 i 4-9. Završni prozor obavještava korisnika da je stigao do željene konačne točke ili da nije moguće stići do konačne točke jer su svi putevi zapriječeni. Na donjoj polovici ekrana nalaze se tri gumba. Prvi po redu, gumb „*Povratak*“ nudi opciju povratka na početnu točku koja se zadala primjerice pri ulasku u rudnik. Srednji gumb „*Reset*“ služi za povratak na početni prozor, gdje se može zadati novi željeni par točaka. Desni gumb, „*Izlaz*“ služi za gašenje aplikacije.



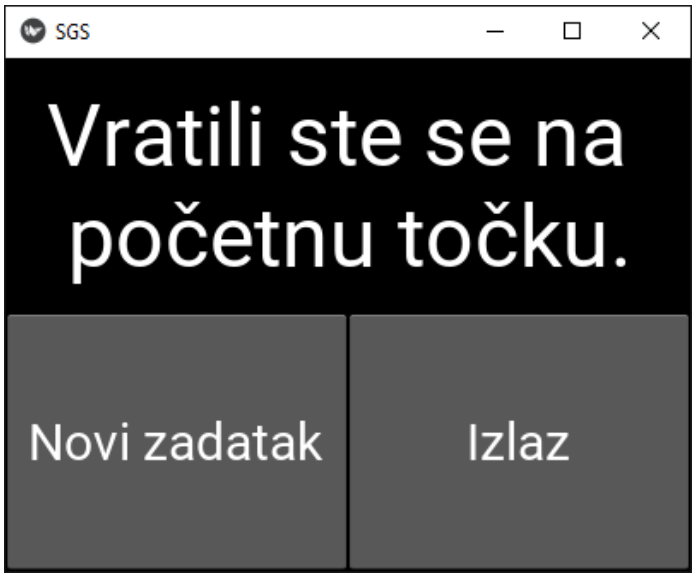
Slika 4-8 Završni prozor nakon dolaska na cilj



Slika 4-9 Završni prozor kada nije moguće doći do cilja

Ako se na završnom prozoru odabere opcija „*Povratak*“, vrti se cijeli novi ciklus radnih prozora istim putem kojim se došlo od prvotne zadane početne točke. Korisnik i dalje ima opciju zabilježiti da mu je put prepriječen čak i ako je prvi put neometano prošao tim hodnikom. Na slici 4-10 prikazan je konačni završni prozor koji se otvori dolaskom na zadanu početnu, odnosno po povratku na konačnu točku. Na donjoj polovici se korisniku nude dvije opcije. Prva

opcija je gumb „*Novi zadatak*“ koji započinje cjelokupni ciklus iz početka, a druga opcija je „*Izlaz*“ koja gasi aplikaciju.



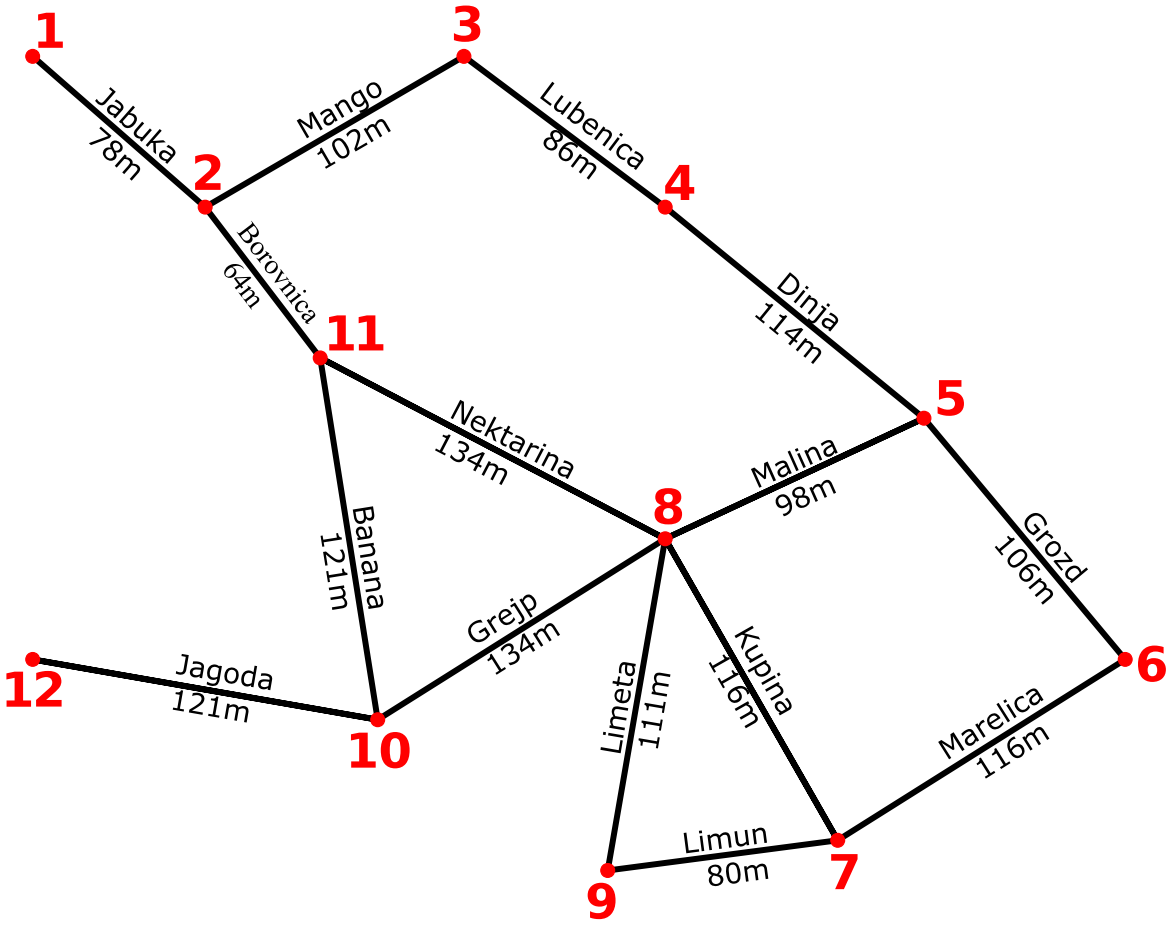
Slika 4-10 Završni prozor nakon povratka na početnu točku

Ukoliko po povratku do početne točke više niti jedan hodnik nije prohodan neovisno o ostvarenju prvobitnog cilja, otvara se konačni završni prozor koji je prikazan na slici 4-11. U tom slučaju jedina opcija je gašenje uređaja pritiskom na gumb „*Izlaz*“.

Slika 4-11 Završni prozor kada su hodnici u svim smjerovima blokirani

5. TESTIRANJE UREĐAJA NA PRIMJERU RUDNIČKE MREŽE

Simulacija nesreće, odnosno prikaz rada uređaja prikazan je korištenjem sheme rudnika koji je prikazan na slici 5-1. te se radi o shemi zamišljenog rudnika. Sastoji se od 12 točaka i 15 hodnika. Točke 1 i 12 predstavljaju ulaze u rudnik. Ukoliko hodnici rudnika, za koji se provodi instalacija SGS-a imaju nazive, oni se mogu unijeti u njega radi dodatne pomoći pri snalaženju. Za potrebe simulacije hodnicima su pridodana imena, a potpuni popis naziva hodnika i njihovih duljina, te točaka koje povezuju prikazan je u tablici 5-1. Prilog 2 prikazuje ispis koda koji služi za generiranje skice rudnika u „.png“ formatu.



Slika 5-1 Karta rudnika za simulaciju.

Tablica 5-1 Popis hodnika s imenima i duljinama.

Naziv tunela	Duljina tunela (m)	Točke koje povezuje
Banana	121	10 – 11
Borovnica	64	2 – 11
Dinja	114	4 – 5
Grejp	134	8 – 10
Grozd	106	5 – 6
Jabuka	78	1 – 2
Jagoda	121	10 – 12
Kupina	116	7 – 8
Limeta	111	8 – 9
Limun	80	7 – 9
Lubenica	86	3 – 4
Malina	98	5 – 8
Mango	102	2 – 3
Marelica	116	6 – 7
Nektarina	134	8 – 11

Scenarij simulacije je slijedeći:

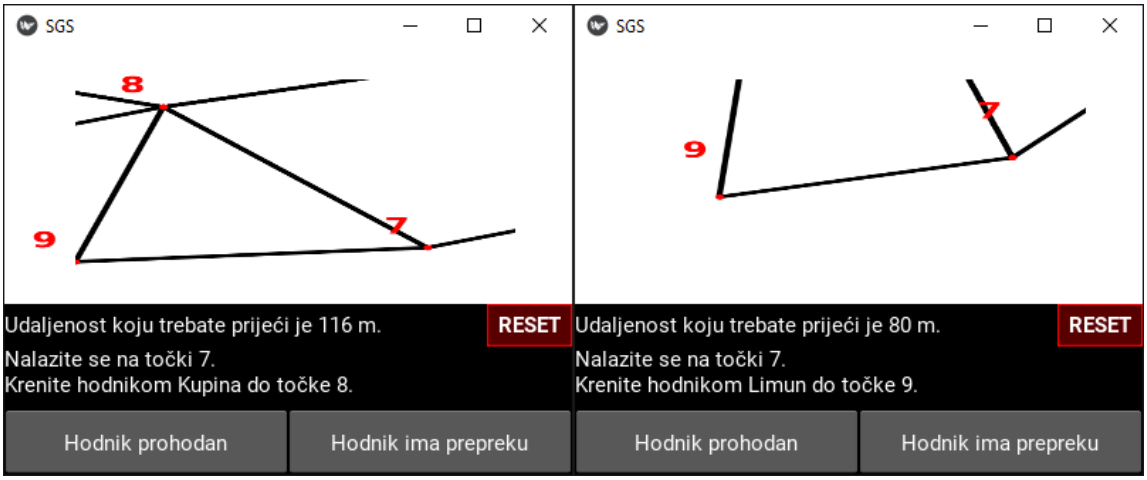
„Došlo je do eksplozije ugljene prašine na otkopnoj fronti Malina. Za vrijeme eksplozije u blizini se nalazilo troje rudara koji se još uvijek nisu vratili na površinu. Zadnja poznata lokacija nestalih rudara bila je točka 7, u blizini otkopne fronte Malina.“

Nakon proučavanja karte rudnika, zaključeno je da je točka 12 optimalan ulaz za potragu nestalih rudara. Spasioci su od ulaza tunelom Jagoda došli do točke 10 kod koje su prvo primijetili da je tunel Grejp zarušen, a zatim i da je tunel Banana zarušen. Obzirom da su svi hodnici koji idu do točke 7 prepriječeni, SGS je uputio spasioce prema izlazu. Prozori u SGS-u ovog slijeda događaja prikazani su na slici 5-2.

Slika 5-2 Ulaz i izlaz kroz točku 12

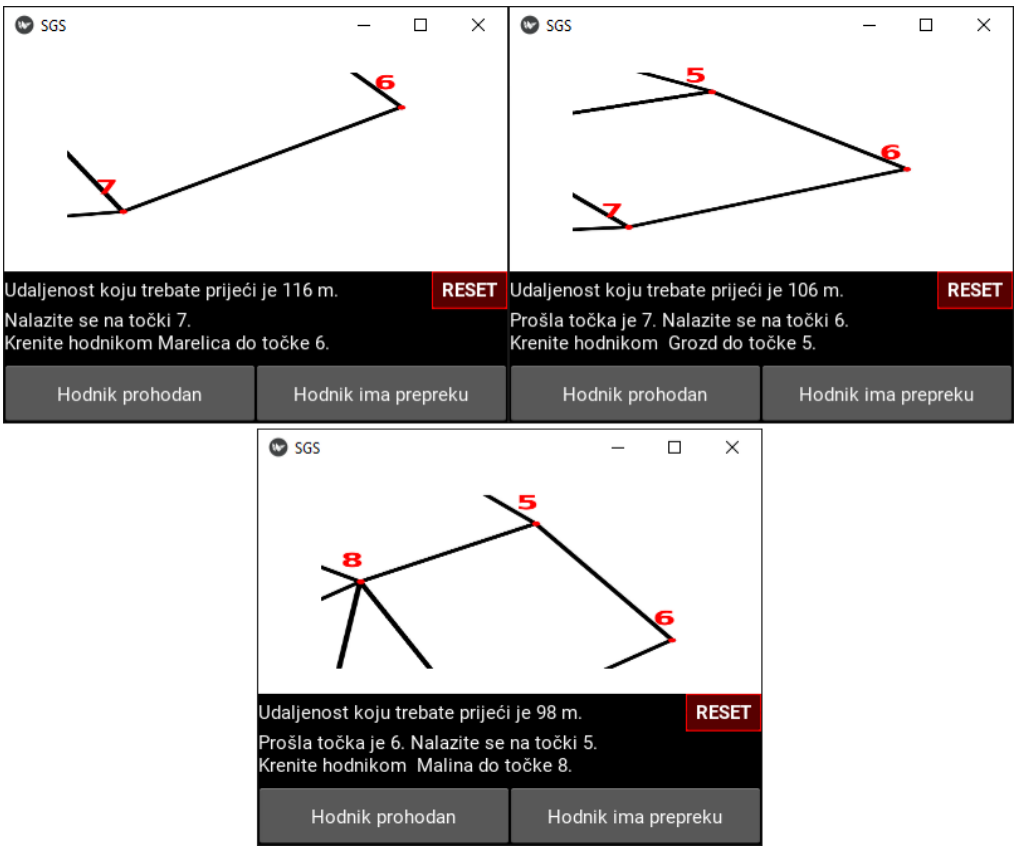
Radi činjenice da su svi putevi od točke 12 nepristupačni spasioci su primorani ući kroz ulaz na točki 1. Unijeli su točku 1 kao početnu i točku 7 kao krajnju. Spasioci su prateći upute SGS-a prikazane na slici 5-3. došli do točke 7 kroz hodnike Jabuka, Banana, Nektarina i Kupina. Na točki 7 spasioci su pronašli sva tri nestala rudara, koji su bili pri svijesti i u mogućnosti hodati. Nakon izmjerene razine plinova zaključeno je da se mogu uputiti prema izlazu na točku 1. Odabirom opcije „*Povratak*“ na sučelju SGS-a, spasioci i rudari su dobili nove upute. Međutim, došlo je do novog zarušavanja zbog kojeg su hodnici Kupina i Limun postali neprohodni. Slika 5-4 prikazuje prozore na kojima su rudari zabilježili stanje hodnika Kupina i Limun.

Slika 5-3 Ulaz kroz točku 1 i put do točke 7.



Slika 5-4 Bilježenje prepriječenih hodnika.

Spasioci i rudari su se prema uputama prikazanim na slici 5-5. uputili prema točki jedan kroz tunele Marelica i Grozd, ali kod hodnika Malina su opet stali zbog zarušenog krova i to pohranili u SGS.



Slika 5-5 Put od točke 7 do točke 5

Nakon bilježenja stanja hodnika Malina, spasioci i rudari su stigli do točke 1 kroz hodnike Dinja, Lubenica, Mango i Jabuka. Prozori ovog segmenta puta prikazani su na slici 5-6. Svi rudari uspješno su spašeni i cjelokupna misija smatra se uspješnom.

Slika 5-6 Povratak do točke 1 od točke 5

Ovom simulacijom prikazana je primjena funkcija SGS-a u jednostavnom scenariju te je dokazano da uređaj radi kao što je zamišljen. Ispis koda koji pokreće SGS nalazi se u prilogu 1.

6. PRIJEDLOZI ZA UNAPRJEĐENJE UREĐAJA

Trenutna inačica SGS-a služi kao jednostavan način navigacije u podzemnim objektima. Prijedlozi planiranih unaprjeđenja su zamišljeni tijekom izrade uređaja sa svrhom da ispune nekoliko ciljeva tj. dodatne funkcionalnosti koje bi povećale iskoristivost uređaja a time i njegovu vrijednost za korisnika. Osim praktičnosti pri korištenju predlaže se dizajnirati novo kućište i dodati pojedine funkcije kako bi se područje primjene proširilo.

Razlozi radi kojih unaprjeđenja nisu odmah implementirana su ograničeni budžet, potrebno dodatno vrijeme za razvoj i implementaciju unaprjeđenja, te potreba za stručnom pomoći od osobljem s područja IT-a.

Planovi za unaprjeđenje su:

- novi dizajn kućišta
- sustav za određivanje lokacije spasioca u stvarnom vremenu
- mogućnost bilježenja i pohranjivanja opažanja u rudniku
- prijenos zapaženih bilješki do kontrolnog centra

6.1. Dizajn kućišta

Trenutno kućište SGS-a prikazano na slici 4-3 dovoljne je funkcionalnosti i čvrstoće za prototip SGS-a, ali je daleko od optimalnog. Dimenzije novog kućišta trebaju biti prilagođene tako da stanu na ruku od korisnika. Uz dimenzije, potrebno je i smanjiti masu uređaja. Osim dimenzija i mase, novo kućište treba biti robusno, odnosno otporno na udarce, prašinu i vodonepropusno. Za otpornost na udarce se preporuča gumom obloženo kućište s čeličnim ojačanjima oko rubova zaslona. Preporuke za novi zaslon na dodir su da ima jak kontrast, dijagonale 4,5 inča do 6 inča.. Razlučivost zaslona bi trebala biti 1280x720 piksela. Ugradnjom zaslona veće rezolucije moguće je implementirati razne praktične funkcije, kao što je primjerice gumb koji može prikazati kartu podzemnog objekta bez obzira na kojem se prozoru u sučelju korisnik nalazi. Staklo zaslona treba biti visoke čvrstoće, primjerice Gorilla Glass 5 ili jače primjerice tvrtke Corning. Također, ako bi se uređaj koristio u potencijalno eksplozivnim

atmosferama bilo bi nužno projektirati ga tako da njegovi dijelovi ne mogu postati uzročnici paljenja te se moraju certificirati kao Ex uređaji (uređaji namijenjeni za rad u zonama opasnosti).

6.2. Sustav za određivanje lokacije

Tijekom razvijanja ideje za dizajn SGS-a, razmotreno je nekoliko opcija za sustav koji bi odredio lokaciju korisnika u podzemnom objektu. Jedan od prvih prijedloga za određivanje smjera kretanja bila je primjena kompasa, odnosno magnetne igle. Među prvim magnetnim instrumentima, koji se koristio u rudarstvu, bio je „švedski rudarski kompas“ koji je imao malu osjetljivost, ali je služio za otkrivanje jako magnetičnih ruda minerala magnetita (Šumanovac, 2012). Naravno, kada se razmatra funkcija određivanja smjera kretanja postalo je očito kako nije moguće primijeniti magnetni instrument s obzirom da će instrument u pojedinim podzemnim objektima reagirati na magnetične stijene i prikazivati netočan smjer sjevera.

Rad objavljen 2019. godine pod nazivom „Sustav za hitnu evakuaciju rudara pri podzemnoj eksploataciji“ (Rehman i dr., 2019) opisao je sustav koji primjenom triangulacije od raznih komunikacijskih čvorova ili RFID-a procjenjuje lokaciju rudara u rudniku. Sustav na temelju lokacije rudare navigira prema izlazu iz rudnika ako je moguće ili prema najbližoj sigurnosnoj stanici ako je izlaz zapriječen.

Sličan sustav se preporuča i za SGS, koji bi periodički slao digitalne signale do komunikacijskih čvorova i na taj način odredio približnu lokaciju spasioca u rudniku. Zatim bi SGS prikazivao tu lokaciju na zaslonu u stvarnom vremenu. Naravno, uzimajući u obzir činjenicu da nepredvidivost nesreća u rudniku SGS se nikada neće primarno oslanjati na takav sustav, ali će značajno olakšati akciju spašavanja ako signal između čvorova nije prekinut.

6.2.1. RFID u rudnicima

Osnovna podjela RFID uređaja je prema frekvenciji na kojoj uređaji rade (Hon i dr., 2015). Tablica 6-1 prikazuje radne frekvencije, njihove domete, te prednosti i nedostatke.

Tablica 6-1 Podjela RFID uređaja prema radnoj frekvenciji (Hon i dr., 2015).

Radna frekvencija	Domet	Prednosti	Nedostatci	Primjena
125 kHz do 134 kHz (LF)	< 0,5 m	- veći otpor smetnjama - mogućnost konsolidacije u blizini vode - može biti pričvršćen na metalne nosače	- nizak domet - niska brzina komunikacije - visoka antena (solenoid)	- čipovi za identifikaciju životinja - imobilizatori automobila - identifikacija proizvoda od metala
13,56 MHz (HF)	< 1,0 m	- manje dimenzije antene - viša brzina komunkacije od LF - niska cijena RFID privjeska	- metalna podloga i voda značajno smanjuju domet i stvaranju smetnje pri komunikaciji	- beskontaktno plaćanje - oznake u skladištima - snimanje i prijenos mjerenih podataka
860 MHz do 960 MHz (pasivni uređaji) (UHF)	< 6,0 m	- mogućnost daljinskog očitanja - visoka brzina - dipolna antena - jeftino za proizvodnju	- ne prodire kroz tekućine - teško za očitati ako je na metalnoj podlozi - globalno nedosljedna frekvencija	- praćenje prijevoza paleta - istovremena identifikacija više proizvoda - elektronička cestarina (ENC)
2,4 GHz	> 10 m	- veće brzine, do 2 Mb/s - male dimenzije dipolne antene	- skupa i zahtjevna proizvodnja - jake smetnje (metali, tekućine, itd..)	- elektronička cestarina (ENC) - snimanje i prijenos podataka u stvarnom vremenu

Prema podatcima iz tablice 6-1, zaključeno je da bi SGS koristio RFID uređaje od 2,4 GHz. Dijeljenje lokacije u stvarnom vremenu, kada je to moguće, je jedna od željenih funkcija kasnijih inačica SGS-a. Međutim, kako je već ranije objašnjeno ovisnost SGS-a o RFID tehnologiji nije primarna funkcija, pogotovo uzimajući u obzir navedene nedostatke. RFID tehnologija biti će u primjeni samo kada je to moguće kao dodatno korištenje dostupnih informacija za spasioce.

Druga podjela je prema načinu rada, odnosno prema snazi odašiljanja. Prema toj podjeli postoje aktivni, polu-pasivni i pasivni RFID uređaji, gdje njihova kategorija ovisi o potrebi za napajanjem za obavljanje određenih funkcija (Hon i dr., 2015).

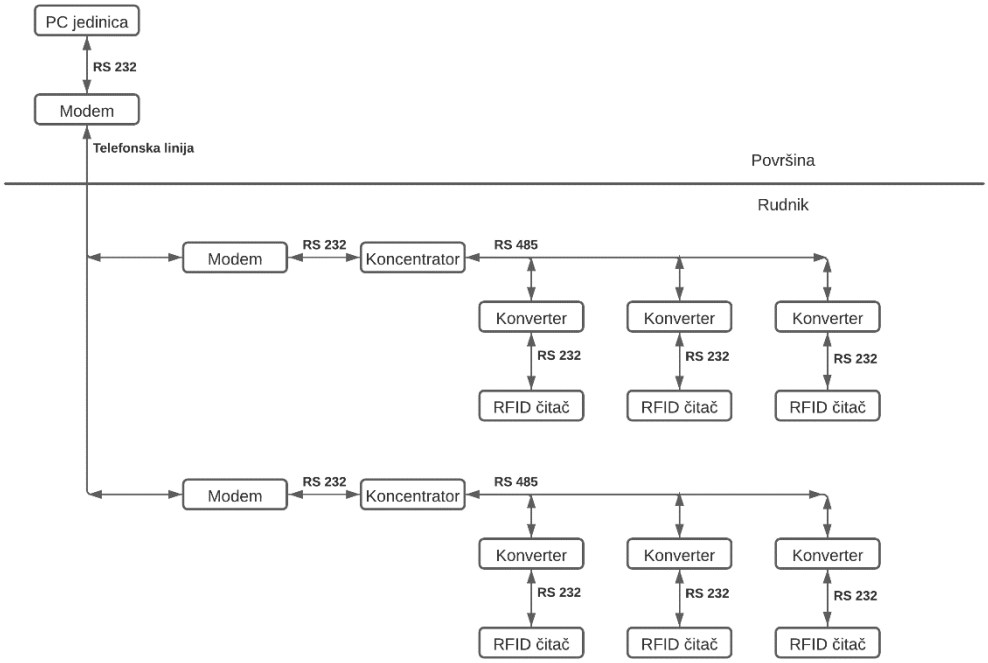
RFID lokacijski sustavi funkcioniraju na principu očitavanja RFID odašiljača koje zaposlenici, odnosno rudari nose u sklopu radne opreme. Očitavanja izvršavaju čitači koji su smješteni u karakterističnim točkama u rudniku. Te karakteristične točke mogu biti primjerice čvorišta hodnika u rudnicima (Liu i dr., 2010)

Tvrtka ZAM-SERVIS s.r.o. u suradnji s tehničkim sveučilištem u Ostravi razvila je RFID sustav za rudnike ugljena. Sustav, prikazan na slici 6-1, funkcionira preko telefonske linije, odnosno odašiljači na RFID uređajima povezuju se s računalom izvan rudnika putem te telefonske linije. Odašiljači šalju signal do konvertera signala, a konverteri šalju te podatke do koncentratora podataka preko međusklopa RS 485. Nakon koncentracije podataka, oni se šalju do podzemnog modema i zatim do površinskog modema preko međusklopa RS 232. Na taj način se s računala na površini može u svakom trenutku pratiti lokacija rudara dok se nalaze u rudniku. (Hon i dr., 2015).

Problemi koje mogu imati sustavi bazirani na bežičnom prijenosu podataka su (Pei i dr., 2009) :

- vodena para i ugljena prašina potencijalno mogu upiti bežični signal na različite načine i to može dovesti do velikih nepreciznosti pri određivanju lokacije
- kompleksan teren i nepravilna topologija podzemnih objekata, pretežito rudnika uzrok su nepravilnog rada algoritama za određivanje lokacije

Autori Pei, Deng, Xu i Xu (2009.) kao rješenje za navedene probleme predlažu ZigBee tehnologiju, koja funkcionira različito od RFID-a, te primjenu nemetričnog MDS algoritma i N-best SBL algoritma. Bitno je napomenuti kako navedeni autori ne predlažu primjenu RFID-a, već predlažu kompletnu primjenu bežičnih sustava višedimenzijskog skaliranja.



Slika 6-1 Topologija RFID sustava za rudnik ugljena (Hon i dr., 2015)

Predloženo unaprjeđenje je funkcija, odnosno mogućnost povezivanja SGS-a na postojeći RFID sustav što bi imalo višestruke prednosti:

- baza na površini, odnosno kontrolni centar ima uvid u lokaciju spasioca ukoliko su RFID stanice funkcionalne
- modifikacijom sustava omogućila bi se obostrana razmjena podataka između SGS-a i baznih stanica što bi omogućilo korisniku da u stvarnom vremenu prati svoju lokaciju unutar rudnika
- omogućiti spasiocima da na zaslonu SGS-a prate lokaciju zatočenih rudara pod uvjetom da su sve stanice operativne

Alternativa postojećem RFID sustavu je postavljanje stanica po rudniku, pretežito na čvorištima i dužim hodnicima. Takav sustav ne bi imao kontakt s površinom, već bi služio kao sredstvo za triangulaciju. Princip rada bio bi očitavanje periodičkih signala (visoko frekventnih radio valova) na SGS-u koje šalju stanice. To bi omogućilo praćenje lokacije korisnika u stvarnom vremenu. Ova alternativa svakako je skuplja od korištenja postojećeg RFID sustava, s obzirom da je kod ove opcije potrebno postavljanje novih baznih stanica koje neće nužno biti iskorištene u druge svrhe.

6.2.2. GIS u zatvorenim prostorima

Komercijalni GIS kao što je Google Maps pruža korisne informacije vezane uz lokaciju, primjerice lokacije restorana i za njih veže informacije o meniju, radnom vremenu i slično. Međutim, komercijalni GIS ne pruža informacije za zatvorene prostore, posebice za podzemne objekte. Jedan od razloga je princip na kojem GIS funkcionira, a to je pomoću koordinatnih sustava i primjene GPS-a.

Glavni izazovi za izradu GIS-a u zatvorenom prostoru su stvaranje GIS-a u zatvorenom prostoru i povezivanje tog GIS-a sa sustavom za pozicioniranje izrađenom za taj zatvoreni prostor. Za brzu izradu modela zatvorenog prostora, odnosno podzemnog objekta može se primijeniti CAD softver, kao što su AutoCAD ili CATIA. Grafički formati datoteka raznih CAD softvera su različiti, stoga se za izradu GIS-a preporuča grafički format VRML kod primjene CAD-a (Jan i dr. 2010).

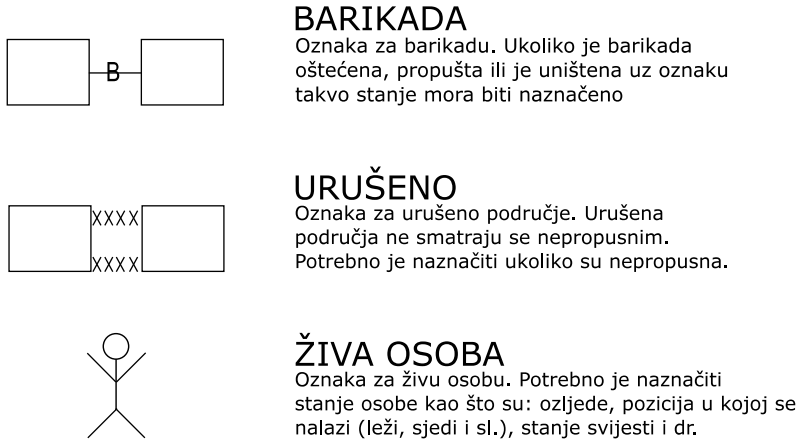
Koraci za razvoj ovakvog GIS-a su (Jan i dr., 2010):

1. postaviti cilj, odnosno funkciju GIS-a
2. prikupiti potrebne informacije, primjerice dimenzije hodnika i okana
3. nacrtati karte i/ili modele u CAD softveru
4. iskoristiti programski alat za razvoj (primjerice C++ Builder) za integraciju razvijenog GIS-a u sustav za pozicioniranje koji se mora zasebno razviti

6.3. Bilježenje i pohranjivanje

Zadatci spasioca s kartom rudnika u spasilačkoj grupi su navigacija grupe i bilježenje svakog bitnog zapaženog detalja. Primjeri simbola i njihovih opisa prikazani su na slici 6-2. Svi bitni faktori i detalji imaju unaprijed definirane i opisane simbole, te način na koji se oni bilježe na karti. Preciznost tih bilješki je značajna jer sve grupe koje ulaze naknadno koriste te bilješke radi lakše orijentacije i snalaženja. Također, ponekad prva grupa može naići na zatočene rudare, ali im ne može odmah pomoći zbog raznih mogućih faktora kao što je primjerice nedostatak potrebne opreme. U tom slučaju se na kartu bilježi lokacija unesrećene osobe i na taj način omogućava idućim grupama da lakše pronađu unesrećenu osobu.

Dodavanjem funkcije bilježenja na kartu u SGS-u značajno bi se povećala praktičnost i skratilo vrijeme bilježenja. Predlaže se izrada dodatnog sučelja koje sadržava sve potrebne simbole. Svi potrebni simboli moći će se odvući na željenu lokaciju na SGS-ovoj digitalnoj karti, a bilješke će pohraniti u uređaju. Također se predlaže izrada algoritma koji bi predlagao korisniku adekvatne lokacije za pojedine simbole kako bi se povećala preciznost i dodatno skratilo vrijeme bilježenja. Povratkom spasilačke grupe na površinu sve će se bilješke moći pohraniti na računalo preko kojeg bi se uređaji preostalih grupa ažurirali.



Slika 6-2 Primjeri simbola za kartu i njihovi opisi (MSHA, 2016)

6.4. Bežični prijenos bilješki u stvarnom vremenu

Kod akcija spašavanja, vrijeme reakcije ponekad je od presudne važnosti. Naravno, žurba ni u kojem slučaju nije preporučljiva jer spasilačke grupe moraju voditi računa prvenstveno o vlastitoj sigurnosti kako bi mogli pružiti pomoć potencijalno unesrećenim osobama u rudniku. Međutim, nastoji se skratiti vrijeme između akcija spašavanja, primjerice ulazaka i izlazaka grupa iz rudnika.

Upravo primjenom sustava kao što je RFID u kombinaciji s SGS-om može se to vrijeme značajno skratiti. Uz ranije predloženu funkciju ažuriranja digitalne karte uređaja drugih spasilačkih ekipa, pomoću ovakvih sustava može se to ažuriranje odraditi u stvarnom vremenu. SGS bi slanjem podataka preko odašiljača u bazne stanice u stvarnom vremenu ažurirao kartu na računalu na površini. Uz već postojeće oblike komunikacije između spasilačkih grupa koje su na površini i onih koje su u rudniku, ovom funkcijom se može dodatno unaprijediti brzina reakcije pri spašavanju zatočenih unesrećenih osoba.

Nedostatak ove funkcije je, kao i kod ostalih funkcija koje koriste sustave poput RFID-a, potreba da sustav tijekom havarije ne bude oštećen. Određene mjere zaštite stanica sustava se mogu poduzeti, primjerice da se njegove komponente zaštite robusnim kućištima. Međutim, niti jedna mjera zaštite ne garantira da će veza stanica s površinom biti održana.

7. ZAKLJUČAK

Ovaj rad predstavlja razvoj uređaja za podzemnu navigaciju pri spašavanju – Subterranean Guidance System. Ovaj uređaj bi mogao u potpunosti zamijeniti konvencionalne načine navigacije i bilježenja pri spašavanju i savladavanju nesreća u podzemnim objektima, prvenstveno u rudnicima.

Objašnjeni su princip rada, Dijkstrin algoritam pomoću kojega se proračunavaju najkraći putevi, primjena Dijkstrinog algoritma pri radu uređaja, komponente hardvera i softvera, te dizajn korisničkog sučelja.

Simulacija na primjeru prikazala je kako logika programa ispravno računa najkraći put do željene točke i time potvrdila tezu da SGS može zamijeniti konvencionalnu kartu pri navigaciji u podzemnim objektima.

Predložena unaprjeđenja kao što su postavljanje bilješki na digitalnu kartu i praćenje lokacije spasilaca u stvarnom vremenu uz postojeće funkcije mogla bi učiniti SGS jednim od korisnijih alata koje spasilačke službe za rudnike i ostale podzemne objekte imaju na raspolaganju.

8. LITERATURA

Članci, knjige, poglavlja iz knjiga

BEAZLEY D., JONES B. K., *Python Cookbook*, Third Edition, O'Reilly Media, Inc., Sebastopol, California, Sjedinjene Američke Države, 2013.

DHILLON, B.S., 2010, *Mine safety – a modern approach*, Ottawa, Canada, University of Ottawa, Department of Mechanical Engineering, 2010.

DOWNEY, A., *Think Python: How to Think Like a Computer Scientist*, 2nd Edition, Version 2.4.0., Green Tea Press, Needham, Massachusetts, Sjedinjene Američke Države, 2015.

HON V., DOSTAL M., SLANINA Z., *The Use of RFID Technology in Mines for Identification and Localization of Persons*, Ostrava, Czech Republic, Technical University of Ostarva, Faculty of Electrical Engineering and Computer Science, ZAM-SERVIS s.r.o., Czech Republic, 2015

HRUŠKA M., *Osnove programiranja (Python)*, Sveučilište u Zagrebu, Sveučilišni računski centar, Zagreb, 2018

JAN S-S., HSU L-T., TSAI W-M., *Development of an Indoor Location Based Service Test Bed and Geographic Information System with a Wireless Sensor Network*, Department of Aeronautics and Astronautics, National Cheng Kung University, Tainan, Taiwan, 2010.

KILINC F. S., MONAGHAN W. D., POWELL J. B., *A review of Mine Rescue Ensembles for Underground Coal Mining in the United States*, Journal of Engineer Fibers and Fabrics, Volume 9, Issue 1, 2014., str. 174-185

LIU Z., LI C., DAI W., GENG S., DING Q., *A Wireless Sensor Netowork Based Personnel Positioning Scheme in Coal Mines with Blind Areas*, Departments of Automation and Electrical Engineering, Tsinghua University, Bejing, China, 2010.

MILLS C., *Regulating Health and Safety in the British Mining Industries, 1800-1914*¸ Stirling University, Ujedinjeno Kraljevstvo, Ashgate Publishing Limited, 2010.

Mine Rescue Operations Training Handbook, priručnik o spašavanju u rudarstvu izrađen tijekom projekta MINERS, Leoben, 2018.

PEI Z., DENG Z., XU S., XU X., *Anhor-free Localization Method for Mobile Targets in Coal Mine Wireless Sensor Networks,* Sensors. 2009.

SNIEDOVICH, M., *Dijkstra's algorithm revisited: the dynamic programming connexion*, Melbourne, Australia, The University of Melbourne, Department of Mathematics and Statistics, 2006.

ŠUMANOVAC F., *Osnove Geofizičkih Istraživanja*, Sveučilište u Zagrebu, Rudarsko-Geološko-Naftni fakultet, Zagreb, 2012, str. 74

MINE SAFETY AND HEALTH ADMINISTRATION (MSHA), *Mine Rescue Contest Rules*, Sjedinjene Američke Države, 2016.

REHMAN A. U., AWUAH-OFFEI K., BAKER D.A., BRISTOW D., *Emergency evacuation guidance system for underground miners*, Conference paper, 2019., SME Annual Meeting, Denver, Colorado, Feb. 24-27, 2018.

ULLOA, R., *Kivy: Interactive Applications in Python*, Packt Publishing Ltd., Birmingham, Ujedinjeno Kraljevstvo, 2013.

www izvori

Joy-IT, 2020., URL: https://joy-it.net/en/

Raspberry Pi Foundation, 2020., URL: https://www.raspberrypi.org/

National Park Service, *History of the Buraeu of Mines Project*, 2015, URL: https://www.nps.gov/miss/learn/management/bomhist.htm

SAŽETAK

Jona Brozović; Bruno Ćesić; Karlo Weiser

RAZVOJ UREĐAJA ZA PODZEMNU NAVIGACIJU PRI SPAŠAVANJU (SGS)

Rudarstvo je jedna od najvažnijih industrijskih grana posljednjih nekoliko desetljeća. Sirovine dobivene eksploatacijom prisutne su u svakodnevnici gotovo cijele populacije. Potrebni kapaciteti za rudama uzrokovali su brojne nesreće pretežito u 20. stoljeću, posebice u manje razvijenim zemljama koje nisu imale stroge i dobro definirane zakone i propise vezane uz rudarstvo. Kako bi se smanjio broj nesreća počelo se intenzivnije raditi na sigurnosti na radu, ali i na treniranju osoblja kako bi se provele uspješne akcije spašavanja unesrećenih radnika.

Ovaj rad opisuje razvoj prototipa uređaja za podzemnu navigaciju pri spašavanju koji bi mogao značajno olakšati akcije spašavanja. Razvijen s ciljem da se olakša i skrati vrijeme potrage za unesrećenima, ali i da poveća efikasnost spasioca u teškim uvjetima, SGS proračunava najkraći put do željene lokacije u rudniku. Proračunate upute mogu se korigirati naredbama u sučelju, te ih SGS pamti dokle god korisnik ne odluči resetirati uređaj.

Predložena unaprjeđenja omogućiti će SGS-u brojne dodatne funkcije. Planirano je poboljšanje kućišta, povezivanje SGS-a na postojeće RFID mreže i/ili instalacija novih sustava za bežično prenošenje podataka na površinu i prikazivanje lokacije unesrećenih osoba u rudniku. Također, planirano je korištenje tehnologija kao što je GIS radi praćenja lokacije korisnika, odnosno spasioca u rudniku. Provesti će se i implementacija interaktivnog sučelja sa simbolima za potrebe bilježenja opažanja stanja rudnika, primjerice zarušenih hodnika ili lokacija unesrećenih osoba.

Ključne riječi: spašavanje u rudnicima, nesreće u rudarstvu, podzemna navigacija, Dijkstrin algoritam, Raspberry Pi

SUMMARY

Jona Brozović; Bruno Ćesić; Karlo Weiser

DEVELOPMENT OF SUBTERRANEAN GUIDANCE SYSTEM (SGS)

Mining has been one the most crucial industrial branches in the past few decades. Raw materials extracted by various mining methods are present in almost every part of the globe. Necessary capacities for raw materials have caused many accidents in the twentieth century, especially in less developed countries that did not have well written laws and regulations related to mining. In order to decrease the number of accidents, authorities from various countries began writing work safety regulations and started to train personnel in mine search and rescue skills to reduce the number of fatalities.

This paper covers the development of the subterranean guidance system (SGS) which could significantly facilitate mine rescue operations. Developed with the goal to simplify mine rescue operations and reduce the time necessary, SGS calculates the shortest path to any desired location inside of a mine. Calculated directions can be altered by using commands provided in the user interface. SGS also remembers changes to the original mine layout until it is reset.

Planned SGS upgrades will enable a number of additional functions such as casing improvement, RFID and SGS networking or installation of new wireless data transfer systems in order to transfer data to the surface in real time and recieve locations of miners carrying RFID tags. It is also planned to implement technologies such as GIS in order to pinpoint relative real time locations of mine rescuers. Another upgrade that shall be implemented is an interactive interface with symbols of mine rescuer observations such as a blocked passage or a location of a trapped person.

Key words: mine search and rescue, mining accidents, underground navigation, Dijkstra's algorithm, Raspberry Pi

PRILOZI

Prilog 1 - Programski kod – Korisničko sučelje

```
from kivy.app import App
from kivy.lang import Builder
from kivy.uix.screenmanager import
ScreenManager, Screen
from kivy.properties import ObjectProperty
from kivy.core.window import Window
from algoritam import test
from kivy.properties import StringProperty
from skica import zoom
from skica import koord
from kivy import Config
from kivy.cache import Cache

Config.set('graphics', 'position', 'custom')
Config.set('graphics', 'left', 0)
Config.set('graphics', 'top', 0)

t1 = "no1.txt"
t2 = "no2.txt"
t3 = "no3.txt"
t4 = "no3.txt"

class MainWindow(Screen):
    jedan = ObjectProperty(StringProperty)
    drugi = ObjectProperty(None)
    treci = ObjectProperty(None)
    cetvrti = ObjectProperty(None)

    def on_enter(self):
        with open("no1.txt", "w") as a:
            a.write("")
        with open("no2.txt", "w") as a:
            a.write("")
        with open("no3.txt", "w") as a:
            a.write("")
        with open("no4.txt", "w") as a:
            a.write("")
        with open("kno1.txt", "w") as a:
            a.write("")
        with open("kno2.txt", "w") as a:
            a.write("")
        with open("kno3.txt", "w") as a:
            a.write("")
        with open("kno4.txt", "w") as a:
            a.write("")

    def plus (self,name):
        fajl = name
        sn = ""

        def read(sn, name):
            gn = sn
            with open(name) as n:
                gn = n.read()
                gn = len(gn)
```

```
    return gn
procitan = read(sn,fajl)

def append(name,br):
    with open(name, "a") as a:
        a.write("1")
        br = br+1
        return br

def brise(name):
    with open(name, "w") as w:
        w.write("")
        gn = "0"
    return gn

if procitan < 9:
    procitan=append(fajl, procitan)
    procitan=str(procitan)
else:
    procitan = brise(fajl)

if name == "no1.txt":
    self.jedan.text = procitan
elif name == "no2.txt":
    self.drugi.text = procitan
elif name == "no3.txt":
    self.treci.text = procitan
else:
    self.cetvrti.text = procitan
return procitan
```

```
def oduzimanje (self, name):
    fajl = name
    sn = ""
    t = 0
    def read(sn, name):
        gn = sn
        with open(name) as n:
            gn = n.read()
            gn = len(gn)
        return gn

    procitan = read(sn, fajl)

    if procitan > 0:
        with open (name, "w") as b:
            b.write("")
        with open (name, "a") as a:
            i=1
            while i < procitan:
                i=i+1
                a.write("1")
        procitan = procitan - 1
        procitan = str(procitan)
    elif procitan == 0:
        procitan = "9"
        with open (name, "w") as b:
            b.write("111111111")

    if name == "no1.txt":
        self.jedan.text = procitan
    elif name == "no2.txt":
```

```
            self.drugi.text = procitan
        elif name == "no3.txt":
            self.treci.text = procitan
        else:
            self.cetvrti.text = procitan
        return procitan

    def jelpostoje(self):
        graf={}
        with open("graf.txt") as a:
            c = a.read()
            graf = eval(c)
        with open("no1.txt", "r") as a:
            prvaz = a.read()
            prvaz = len(prvaz)
            prvaz = str(prvaz)
        with open("no2.txt", "r") as a:
            drugaz = a.read()
            drugaz = len(drugaz)
            drugaz = str(drugaz)
        with open("no3.txt", "r") as a:
            trecaz = a.read()
            trecaz = len(trecaz)
            trecaz = str(trecaz)
        with open("no4.txt", "r") as a:
            cetvrtaz = a.read()
            cetvrtaz = len(cetvrtaz)
            cetvrtaz = str(cetvrtaz)

        if prvaz != "0":
            pocetna = prvaz + drugaz
        else:
            pocetna = drugaz
        if trecaz != "0":
            krajnja = trecaz + cetvrtaz
        else:
            krajnja = cetvrtaz
        if pocetna in graf.keys() and krajnja in
graf.keys():
            self.manager.current="third"
        else:
            self.manager.current="Krivet"

    def potvrda (self):

        def confirmed(self):
            with open("no1.txt", "r") as a:
                prvaz = a.read()
                prvaz = len(prvaz)
                prvaz = str(prvaz)
            with open("no2.txt", "r") as a:
                drugaz = a.read()
                drugaz = len(drugaz)
                drugaz = str(drugaz)
            with open("no3.txt", "r") as a:
                trecaz = a.read()
                trecaz = len(trecaz)
                trecaz = str(trecaz)
            with open("no4.txt", "r") as a:
                cetvrtaz = a.read()
                cetvrtaz = len(cetvrtaz)
                cetvrtaz = str(cetvrtaz)
```

```
        if prvaz != "0":
            pocetna = prvaz + drugaz
        else:
            pocetna = drugaz
        if trecaz != "0":
            krajnja = trecaz + cetvrtaz
        else:
            krajnja = cetvrtaz

        return pocetna, krajnja
    b = confirmed(self)
    if b[1] == 0:
        with open("tocke.txt", "w") as a:
            a.write("5"+" "+"1")
    else:
        with open ("tocke.txt", "w") as a:
            a.write(b[0]+" "+b[1]+" z"+"
q")
        with open ("poct.txt", "w") as a:
            a.write(b[0])
    self.jedan.text = "0"
    self.drugi.text = "0"
    self.treci.text = "0"
    self.cetvrti.text = "0"

class ThirdWindow(Screen):
    testlabel = ObjectProperty(None)
    udaljenostlabel= ObjectProperty(None)
    image_source = StringProperty()

    def on_enter(self):
        with open ("blokiran.txt") as a:
            blok = a.read()
            blok = int(blok)
        if blok < 1:
            with open ("tocke.txt") as a:
                b= a.read()
                b= b.split()
        else:
            with open ("tocke.txt") as a:
                b = a.read()
                b = b.split()
        dj = test(b[0],b[1],b[2],b[3])
        if "z" in dj:
            self.manager.current = "Krivet"
        elif "m" in dj:
            with
open("kolkoputajeblokiran.txt") as a:
                blok = a.read()
                blok = int(blok)
            blok=blok+1
            with open
("kolkoputajeblokiran.txt", "w") as a:
                a.write(str(blok))
            if blok < 2:
                self.manager.current = 'prvibl'
            else:
                self.manager.current = 'blokiran'
        else:
            udaljenost = dj [0]
            gl = []
```

```
        i=1
        z = len(dj)

        while i < z:
            gl.append(dj[i])
            i=i+1
        z1 = (z/2)
        j=0
        k=int(z1)
        tocke = []
        upute = []
        while j < z1:
            tocke.append(gl[j])
            j=j+1
        while k < (z-1):
            upute.append(gl[k])
            k=k+1

        with open ("tockegraf.txt", "w")
as a:
            a.write(str(tocke))
        with open("uputegraf.txt","w") as
a:
            a.write(str(upute))
        with open("brojac.txt","w") as a:
            a.write("1")
        m1 = koord[str(tocke[0])]
        m2 = koord[str(tocke[1])]
        zoom(m1,m1,m2)
        self.ids['my_image'].source =
'graf2.png'
        self.ids['my_image'].source =
'graf.png'
        self.testlabel.text ="Nalazite se na
točki "+tocke[0]+ "."+"\n"+"Krenite
hodnikom "+upute[0]+" do točke
"+tocke[1]+"."
        graf={}
        with open ("graf.txt")as a:
            graf=a.read()
            graf=eval(graf)

udalj=str(graf[str(tocke[0])][str(tocke[1])][0]
)
        self.udaljenostlabel.text =
"Udaljenost koju trebate prijeći je " +
udalj + " m."

    def predenatocka (self):
        with open ("tockegraf.txt") as a:
            tocke = []
            tocke = a.read()
            tocke = tocke.replace("'","")
            tocke = tocke.replace("[","")
            tocke = tocke.replace("]", "")
            tocke = tocke.replace(" ","")
            tocke = tocke.split(",")
            be = len(tocke)
        with open("uputegraf.txt") as a:
            upute = []
            upute = a.read()
            upute = upute.replace("'", "")
```

```
            upute = upute.replace("[", "")
            upute = upute.replace("]", "")
            upute = upute.split(",")
            b = len(upute)
        with open("brojac.txt") as a:
            i = len(a.read())

        i = i+1
        with open ("brojac.txt", "a") as a:
            a.write("1")
        if i <= b:
            if i < 2:
                m1 = koord[str(tocke[i-1])]
            else:
                m1 = koord[(tocke[i-2])]
            m2 = koord[tocke[i-1]]
            m3 = koord[str(tocke[i])]
            zoom(m1, m2, m3)
            Cache.remove("kv.image")
            Cache.remove("kv.texture")
            self.ids['my_image'].source =
'graf2.png'
            self.ids['my_image'].source =
'graf.png'
            self.testlabel.text = "Prošla točka je
" + tocke[i - 2] + ". Nalazite se na točki "
+ tocke[
                i - 1] + "."+"\n"+"Krenite
hodnikom "+upute[i-1]+" do točke " +
tocke[i]+"."
            graf = {}
            with open("graf.txt")as a:
                graf = a.read()
                graf = eval(graf)
            udalj = str(graf[str(tocke[i-
1])][str(tocke[i])][0])
            self.udaljenostlabel.text =
"Udaljenost koju trebate prijeći je " +
udalj + " m."
            with open ("krajt.txt","w") as a:
                a.write(tocke[i-1])
            with open("tocke.txt", "w") as a:
                a.write(str(tocke[i - 1]) + " " +
str(tocke[be-1]) + " z" + " q")
            with open("prvat.txt") as a:
                poct = a.read()
            with
open("povrataktocke.txt","w") as a:
                a.write(str(tocke[i-1])+"
"+str(poct)+" z q")

        else:
            with open ("poct.txt","r") as a:
                prvaz = a.read()
                prvaz = str(prvaz)
            with open ("tocke.txt", "w") as a:
                a.write(tocke[i-1]+" "+prvaz+"
a"+" b")
            with open("tocke.txt","r") as a:
                tocke=a.read()
            with open("pathclosed.txt", "w")
as a:
```

```
            a.write(" ")
        with open("jelprviput.txt") as a:
            kojiput = a.read()
            kojiput = int(kojiput)
            kojiput=kojiput+1
        with open("jelprviput.txt","w") as
a:
            a.write(str(kojiput))
        with open("jelprviput.txt", "r") as
a:
            kojiput=a.read()
            kojiput=int(kojiput)
        with open
("kolkoputajeblokiran.txt", "r") as a:
            kolkobl=int(a.read())
        if kojiput == 1 and kolkobl <1:
            self.manager.current = 'end'
        else:
            self.manager.current = 'Nend'



    def zblokiran(self):
        with open ("brojac.txt") as a:
            i = len(a.read())
        with open ("tockegraf.txt") as a:
            ptocke = a.read()
            ptocke = ptocke.replace("'", "")
            ptocke = ptocke.replace("[", "")
            ptocke = ptocke.replace("]", "")
            ptocke = ptocke.split(",")
        with open ("tocke.txt") as a:
            cilj = a.read().split()
        with open ("tockegraf.txt") as a:
            ktocke = a.read()
            ktocke = ktocke.replace("'", "")
            ktocke = ktocke.replace("[", "")
            ktocke = ktocke.replace("]", "")
            ktocke = ktocke.split(",")
            br = len(ktocke)
        with open ("tocke.txt", "w") as a:
            a.write(ptocke[i-1]+" "+cilj[1]+"
"+ktocke[i-1]+" "+ktocke[i])
        self.on_enter()

class EndWindow(Screen):
    def on_enter(self, *args):
        with open ("pathclosed.txt", "w") as
a:
            a.write("1")

class NewEnd(Screen):
    pass

class Blokiran(Screen):
    pass

class PrviBl(Screen):
    def on_enter(self, *args):
        with open("blokiran.txt", "w") as a:
            a.write("1")
        with open("poct.txt") as a:
            pt = int(a.read())
```

```
        with open("krajt.txt") as a:
            krajt = int(a.read())
        with open("tocke.txt", "w") as a:
            a.write(str(krajt)+" "+str(pt)+" q z")

class Krivet(Screen):
    def on_enter(self, *args):
        graph = {
            '1': {'2': (78, 'Jabuka')},
            '2': {'1': (78, 'Jabuka'), '3': (102,
'Mango'), '11': (64, 'Borovnica')},
            '3': {'2': (102, 'Mango'), '4': (86,
'Lubenica')},
            '4': {'3': (86, 'Lubenica'), '5': (114,
'Dinja')},
            '5': {'4': (114, 'Dinja'), '6': (106,
'Grozd'), '8': (98, 'Malina')},
            '6': {'5': (106, 'Grozd'), '7': (116,
'Marelica')},
            '7': {'6': (116, 'Marelica'), '9': (80,
'Limun'), '8': (116, 'Kupina')},
            '8': {'5': (98, 'Malina'), '9': (80,
'Limeta'), '7': (116, 'Kupina'), '10': (134,
'Grejp'),
                  '11': (134, 'Nektarina')},
            '9': {'7': (80, 'Limun'), '8': (111,
'Limeta')},
            '10': {'8': (134, 'Grejp'), '11': (121,
'Banana'), '12': (121, 'Jagoda')},
            '11': {'2': (64, 'Borovnica'), '8':
(134, 'Nektarina'), '10': (121, 'Banana')},
            '12': {'10': (121, 'Jagoda')},
            'z': {'q': (2, 'desno')},
            'q': {'z': (2, 'lijevo')},
            'x': {'y': (2, 'lijevo')},
            'y': {'x': (2, 'desno')},
            'a': {'b': (2, 'desno')},
            'b': {'a': (2, 'lijevo')},
            'c': {'d': (2, 'lijevo')},
            'd': {'c': (2, 'desno')},
        }
        with open("graf.txt", "w") as a:
            a.write(str(graph))
        with open("no1.txt", "w") as a:
            a.write("")
        with open("no2.txt", "w") as a:
            a.write("")
        with open("no3.txt", "w") as a:
            a.write("")
        with open("no4.txt", "w") as a:
            a.write("")
        with open("brojac.txt", "w") as a:
            a.write("1")
        with open("pathclosed.txt", "w") as
a:
            a.write("")
        with open("jelprviput.txt", "w") as a:
            a.write("0")
        with open("blokiran.txt", "w") as a:
            a.write("0")
        with open("kolkoputajeblokiran.txt",
"w") as a:
```

```
        a.write("0")

class WindowManager(ScreenManager):
    pass

sm = WindowManager()
kv = Builder.load_file("my.kv")

screens =
[MainWindow(name="main"),ThirdWindo
w(name="third"),EndWindow(name="end
"), NewEnd(name="Nend"),
Blokiran(name="blokiran"),
PrviBl(name="prvibl"),
Krivet(name="Krivet")]
for screen in screens:
    sm.add_widget(screen)

class SGSApp(App):
    def build(self):
        Window.size = (400, 300)
        return sm

    def restart(self):
        graph = {
            '1': {'2': (78, 'Jabuka')},
            '2': {'1': (78, 'Jabuka'), '3': (102,
'Mango'), '11': (64, 'Borovnica')},
            '3': {'2': (102, 'Mango'), '4': (86,
'Lubenica')},
            '4': {'3': (86, 'Lubenica'), '5': (114,
'Dinja')},
            '5': {'4': (114, 'Dinja'), '6': (106,
'Grozd'), '8': (98, 'Malina')},
            '6': {'5': (106, 'Grozd'), '7': (116,
'Marelica')},
            '7': {'6': (116, 'Marelica'), '9': (80,
'Limun'), '8': (116, 'Kupina')},
            '8': {'5': (98, 'Malina'), '9': (80,
'Limeta'), '7': (116, 'Kupina'), '10': (134,
'Grejp'),
                  '11': (134, 'Nektarina')},
            '9': {'7': (80, 'Limun'), '8': (111,
'Limeta')},
            '10': {'8': (134, 'Grejp'), '11': (121,
'Banana'), '12': (121, 'Jagoda')},
            '11': {'2': (64, 'Borovnica'), '8':
(134, 'Nektarina'), '10': (121, 'Banana')},
            '12': {'10': (121, 'Jagoda')},
            'z': {'q': (2, 'desno')},
            'q': {'z': (2, 'lijevo')},
            'x': {'y': (2, 'lijevo')},
            'y': {'x': (2, 'desno')},
            'a': {'b': (2, 'desno')},
            'b': {'a': (2, 'lijevo')},
            'c': {'d': (2, 'lijevo')},
            'd': {'c': (2, 'desno')},
        }
        with open("graf.txt", "w") as a:
            a.write(str(graph))
        with open("no1.txt", "w") as a:
            a.write("")
```

```
        with open("no2.txt", "w") as a:
            a.write("")
        with open("no3.txt", "w") as a:
            a.write("")
        with open("no4.txt", "w") as a:
            a.write("")
        with open("brojac.txt", "w") as a:
            a.write("1")
        with open("pathclosed.txt", "w") as
a:
            a.write("")
        with open("jelprviput.txt", "w") as a:
            a.write("0")
        with open("blokiran.txt", "w") as a:
            a.write("0")
        with open("kolkoputajeblokiran.txt",
"w") as a:
            a.write("0")


    def on_start(self):
        graph = {
            '1': {'2': (78, 'Jabuka')},
            '2': {'1': (78, 'Jabuka'), '3': (102,
'Mango'), '11': (64, 'Borovnica')},
            '3': {'2': (102, 'Mango'), '4': (86,
'Lubenica')},
            '4': {'3': (86, 'Lubenica'), '5': (114,
'Dinja')},
            '5': {'4': (114, 'Dinja'), '6': (106,
'Grozd'), '8': (98, 'Malina')},
            '6': {'5': (106, 'Grozd'), '7': (116,
'Marelica')},
            '7': {'6': (116, 'Marelica'), '9': (80,
'Limun'), '8': (116, 'Kupina')},
            '8': {'5': (98, 'Malina'), '9': (80,
'Limeta'), '7': (116, 'Kupina'), '10': (134,
'Grejp'),
                  '11': (134, 'Nektarina')},
            '9': {'7': (80, 'Limun'), '8': (111,
'Limeta')},
            '10': {'8': (134, 'Grejp'), '11': (121,
'Banana'), '12': (121, 'Jagoda')},
            '11': {'2': (64, 'Borovnica'), '8':
(134, 'Nektarina'), '10': (121, 'Banana')},
            '12': {'10': (121, 'Jagoda')},
            'z': {'q': (2, 'desno')},
            'q': {'z': (2, 'lijevo')},
            'x': {'y': (2, 'lijevo')},
            'y': {'x': (2, 'desno')},
            'a': {'b': (2, 'desno')},
            'b': {'a': (2, 'lijevo')},
            'c': {'d': (2, 'lijevo')},
            'd': {'c': (2, 'desno')},
        }
        with open ("graf.txt", "w") as a:
            a.write(str(graph))
        with open ("blokiran.txt", "w") as a:
            a.write("0")
        with open("prvat.txt","w") as a:
            a.write("")

    def on_stop(self):
```

```
        with open("no1.txt", "w") as a:
            a.write("")
        with open("no2.txt", "w") as a:
            a.write("")
        with open("no3.txt", "w") as a:
            a.write("")
        with open("no4.txt", "w") as a:
            a.write("")
        with open("brojac.txt", "w") as a:
            a.write("1")
        with open("pathclosed.txt", "w") as
a:
            a.write("")
        with open("jelprviput.txt", "w") as a:
            a.write("0")
        with open("blokiran.txt", "w") as a:
            a.write("0")
        with open("kolkoputajeblokiran.txt",
"w") as a:
            a.write("0")


if __name__ == "__main__":
    SGSApp().run()
```

Prilog 2 - Programski kod – Skica rudnika

```
import matplotlib.path as mpath
import matplotlib.patches as mpatches
import matplotlib.pyplot as plt

def moria(to1,to2,to3,to4):
    t1 = (3,3)
    t2 = (3.6,2.5)
    t3 = (4.5,3)
    t4 = (5.2,2.5)
    t5 = (6.1,1.8)
    t6 = (6.8,1)
    t7 = (5.8,0.4)
    t8 = (5.2,1.4)
    t9 = (5,0.3)
    t10 = (4.2,0.8)
    t11 = (4,2)
    t12 = (3,1)

    x = [3, 3.6, 4.5, 5.2, 6.1, 6.8, 5.8, 5.2, 5,
4.2, 4, 3]
    y = [3, 2.5, 3, 2.5, 1.8, 1, 0.4, 1.4, 0.3, 0.8,
2, 1]
    n = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

    fig, ax = plt.subplots()
    ax.scatter(x, y)

    for i, txt in enumerate(n):
        ax.annotate(txt, (x[i], y[i]), size = 30,
color = "red", weight = 'bold',
xytext=(x[i]-.1, y[i]+.1))

    Path = mpath.Path
    path_data = [
        (Path.MOVETO, t1),
        (Path.LINETO, t2),
        (Path.LINETO, t3),
        (Path.LINETO, t4),
        (Path.LINETO, t5),
        (Path.LINETO, t8),
        (Path.LINETO, t7),
        (Path.LINETO, t6),
        (Path.LINETO, t5),
        (Path.LINETO, t8),
        (Path.LINETO, t7),
        (Path.LINETO, t9),
        (Path.LINETO, t8),
        (Path.LINETO, t10),
        (Path.LINETO, t12),
        (Path.LINETO, t10),
        (Path.LINETO, t11),
        (Path.LINETO, t8),
        (Path.LINETO, t11),
        (Path.CLOSEPOLY, t2),
        ]
    codes, verts = zip(*path_data)
```

```
    path = mpath.Path(verts, codes)
    patch = mpatches.PathPatch(path,
facecolor='b', alpha=0)
    ax.add_patch(patch)



    x, y = zip(*path.vertices)
    line, = ax.plot(x, y,'-', color =
"black",linewidth=5)
    plt.plot(x,y,'o', color = "red")
    plt.axis('off')
    plt.axis([to1,to2,to3,to4])
    plt.savefig('graf.png')



def zoom(m1,m2,m3):
    xmax = max(m1[0],m2[0],m3[0])+.2
    xmin = min(m1[0],m2[0],m3[0])-.2
    ymax = max(m1[1],m2[1],m3[1])+.2
    ymin = min(m1[1],m2[1],m3[1])-.2


    moria (xmin, xmax,ymin,ymax)


koord = {
    "1":(3,3),
    "2":(3.6,2.5),
    "3":(4.5,3),
    "4":(5.2,2.5),
    "5":(6.1,1.8),
    "6":(6.8,1),
    "7":(5.8,0.4),
    "8":(5.2,1.4),
    "9":(5,0.3),
    "10":(4.2,0.8),
    "11":(4,2),
    "12":(3,1)
}
```

Prilog 3 - Programski kod – Dijkstra algoritam

```
def test (x,y,r,o):
    graph = {}
    with open("graf.txt") as a:
        c = a.read()
        graph = eval(c)
    def dijkstra(graph,start,goal):
        graph_dij=graph.copy()
        shortest_distance = {}
        track_predecessor = {}
        unseenNodes = graph.copy()
        infinity = 5000
        track_path = []
        izlaz = []
        for node in unseenNodes:
            shortest_distance[node] = infinity
        shortest_distance[start] = 0
        while unseenNodes:
            min_distance_node = None

            for node in unseenNodes:
                if min_distance_node is None:
                    min_distance_node = node

                elif shortest_distance[node] <
shortest_distance[min_distance_node]:
                    min_distance_node = node

            path_options =
graph[min_distance_node].items()
            for child_node, pomocna in
path_options:
                weight=pomocna[0]
                if weight +
shortest_distance[min_distance_node] <
shortest_distance[child_node]:

                    shortest_distance[child_node] =
weight +
shortest_distance[min_distance_node]

                    track_predecessor[child_node]
= min_distance_node



unseenNodes.pop(min_distance_node)
        currentNode = goal

        while currentNode != start:

            try:
                track_path.insert(0,currentNode)
                currentNode =
track_predecessor[currentNode]
            except KeyError:
                b= "m"
                return b
                break
        track_path.insert(0,start)
```

```
        putokaz=[]
        for i in range(0, len(track_path)-1):
            curr=track_path[i]
            next=track_path[i+1]
            if graph_dij.get(curr).get(next) is
None:
                b="z"
                return b
                exit()
            else:

putokaz.append(graph_dij.get(curr).get(next
)[1])
        if shortest_distance[goal] != infinity:
            i=0
            s=0

izlaz.append(str(shortest_distance[goal]))
            while i < len(track_path):
                izlaz.append(str(track_path[i]))
                i = i + 1
            while s < len(putokaz):
                izlaz.append(str(putokaz[s]))
                s = s + 1
        return izlaz

    def remove_path(graph_function,
ruin_start, ruin_end):
        if ruin_end in graph[ruin_start].keys():

graph_function.get(ruin_start).pop(ruin_end
)

graph_function.get(ruin_end).pop(ruin_start
)
            with open("graf.txt", "w") as a:
                a.write(str(graph_function))
        else:
            pass

    def checkPoints(graph_function,
ruin_start, ruin_end):

neigboursOfStartPoint=graph_function.get(r
uin_start).keys()
        if(ruin_end in neigboursOfStartPoint):
            return True
        else:
            b = "z"
            return b

    def checkPoint(graph_function, point):
        points=graph_function.keys()
        if(point in points):
            return True
        else:
            b="z"
            return b
    b=""
    start = x
    if checkPoint(graph, start) is True:
```

```
    wrongInput = False
else:
    b = "z"
    return b
    exit()

end = y
if(checkPoint(graph, end)):
    wrongInput = False
else:
    b = "z"
    return b
    exit()
```

```
    wrongInput=True
    while wrongInput:
        ruin_start = r
        ruin_end = o
        if(checkPoints(graph, r, o)):
            remove_path(graph, ruin_start,
ruin_end)
            wrongInput = False
            b = dijkstra(graph, start, end)
        else:
            b = "z"
            return b
    return b
```

Prilog 4 - Programski kod – .kv datoteka

```
WindowManager:
    MainWindow:

<MainWindow>:
    name: "main"
    jedan: prvi
    drugi: drugi
    treci: treci
    cetvrti: cetvrti
    yes:yes

    FloatLayout:
        cols:1
        canvas.before:
            Color:
                rgba: 1, 1, 1, 1
            Rectangle:
                pos: self.pos
                size: self.size
        GridLayout:
            cols:2
            size_hint: 1, 0.1
            pos_hint: {"x":0, "top":1}
            Label:
                canvas.before:
                    Color:
                        rgba: 1, 1, 1, 1
                    Rectangle:
                        pos: self.pos
                        size: self.size
                text: "Po\u010Detna to\u010Dka"
                font_size: 30
                color: (0,0,0,1)
            Label:
                canvas.before:
                    Color:
                        rgba: 1, 1, 1, 1
                    Rectangle:
                        pos: self.pos
                        size: self.size
                text: "Krajnja to\u010Dka"
                font_size: 30
                color: (0,0,0,1)
        GridLayout:
            cols: 2
            size_hint: .49, 0.68
            pos_hint: {"x":0.005, "top":0.89}

            Button:
                canvas.before:
                    Color:
                        rgba: .35, .35, .35, 1
                    Rectangle:
                        pos: self.pos
                        size: self.size
                text: "+"
                on_press: root.plus("no1.txt")
                font_size: 60
```

```
Button:
    canvas.before:
        Color:
            rgba: .35, .35, .35, 1
        Rectangle:
            pos: self.pos
            size: self.size
    text: "+"
    on_press: root.plus("no2.txt")
    font_size: 60
Label:
    canvas.before:
        Color:
            rgba: 0, 0, 0, 1
        Rectangle:
            pos: self.pos
            size: self.size
    id: prvi
    text: "0"
    font_size: 60
Label:
    canvas.before:
        Color:
            rgba: 0, 0, 0, 1
        Rectangle:
            pos: self.pos
            size: self.size
    id: drugi
    text: "0"
    font_size: 60
Button:
            canvas.before:
                Color:
                    rgba: .35, .35, .35, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            text: "-"
            on_press:
root.oduzimanje("no1.txt")
            font_size: 60
        Button:
            canvas.before:
                Color:
                    rgba: .35, .35, .35, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            text: "-"
            on_press:
root.oduzimanje("no2.txt")
            font_size: 60

    GridLayout:
        cols:2
        size_hint: .49, 0.68
        pos_hint: {"x":0.505, "top":0.89}
        Button:
            canvas.before:
                Color:
                    rgba: .35, .35, .35, 1
                Rectangle:
```

```
            pos: self.pos
            size: self.size
        text: "+"
        on_press: root.plus("no3.txt")
        font_size: 60
    Button:
        canvas.before:
            Color:
                rgba: .35, .35, .35, 1
            Rectangle:
                pos: self.pos
                size: self.size
        text: "+"
        on_press: root.plus("no4.txt")
        font_size: 60
    Label:
        canvas.before:
            Color:
                rgba: 0, 0, 0, 1
            Rectangle:
                pos: self.pos
                size: self.size
        id: treci
        text: "0"
        font_size: 60
    Label:
        canvas.before:
            Color:
                rgba: 0, 0, 0, 1
            Rectangle:
                pos: self.pos
                size: self.size
        id: cetvrti
        text: "0"
        font_size: 60
    Button:
        canvas.before:
            Color:
                rgba: .35, .35, .35, 1
            Rectangle:
                pos: self.pos
                size: self.size
        text: "-"
        on_press:
root.oduzimanje("no3.txt")
        font_size: 60
    Button:
        canvas.before:
            Color:
                rgba: .35, .35, .35, 1
            Rectangle:
                pos: self.pos
                size: self.size
        text: "-"
        on_press:
root.oduzimanje("no4.txt")
        font_size: 60

<Root>:
    GridLayout:
        cols: 2
        size_hint: 1, 0.2
        pos_hint: {"x":0, "top":0.2}
```

```
        Button:
            canvas.before:
                Color:
                    rgba: .35, .35, .35, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            text: "Izlaz"
            font_size: 50
            on_release:
                app.stop()

        Button:
            canvas.before:
                Color:
                    rgba: .35, .35, .35, 1
                Rectangle:
                    pos: self.pos
                    size: self.size

            id: yes
            text: "Potvrdi"
            font_size: 50
            on_release:
                root.potvrda()
                root.jelpostoje()

root.manager.transition.direction = "left"

<ThirdWindow>:
    testlabel: testlabel
    udaljenostlabel: udaljenost
    my_image : my_image
    name: "third"
    FloatLayout:
        GridLayout:
            cols:1
            size_hint: 0.15,0.1
            pos_hint: {"x":0.85,"top":.4}
            Button:
                canvas.before:
                    Color:
                        rgb: 1,0,0
                    Rectangle:
                        size: self.size
                        pos: self.pos
                background_color: (1, 0, 0, 1)
                bold: True
                text:"RESET"
                on_release:
                    app.restart()
                    app.root.current = "main"
        GridLayout:
            cols:1
            size_hint: 1, 0.6
            pos_hint: {"x":0, "top":1}
            Image:
                id: my_image
                size_hint: self.image_ratio, 1
                keep_ratio: False
                allow_stretch: True
                source: root.image_source
```

```
        nocache: True
        canvas.before:
            Color:
                rgb: 1,1,1
            Rectangle:
                size: self.size
                pos: self.pos
GridLayout:
    cols:1
    size_hint: 3, 0.15
    pos_hint: {"x":0, "top":0.47}
    Label:
        id: udaljenost
        text_size: self.size
        text: ""
GridLayout:
    cols:1
    size_hint: 3, 0.25
    pos_hint: {"x":0, "top":0.43}
    Label:
        id: testlabel
        text_size: self.size
        wrap: True
        text: ""
GridLayout:
    cols:2
    size_hint: 1, 0.15
    pos_hint: {"x":0, "top":0.15}
    Button:
        text: "Hodnik prohodan"
        on_release:
            root.predenatocka()
    Button:
        text: "Hodnik ima prepreku"
        on_release:
            root.zblokiran()



<EndWindow>:
    name: "end"
    FloatLayout:
        GridLayout:
            size_hint: 1, 0.5
            pos_hint: {"x":0, "top":1}
            cols: 1
            Label:
                font_size: 50
                text: "Do\u0161li ste do cilja."
            Label:
                font_size: 30
                text: "Vratite se na po\u010Detnu
to\u010Dku."
        GridLayout:
            cols: 3
            size_hint: 1, 0.5
            pos_hint: {"x":0, "top":.5}
            Button:
                font_size: 30
                text: "Povratak"
                on_release:
                    app.root.current = "third"
```

```
root.manager.transition.direction = "left"
        Button:
            font_size: 30
            text: "Reset"
            on_release:
                app.restart()
                app.root.current = "main"
        Button:
            font_size: 30
            text: "Izlaz"
            on_release: app.stop()


<NewEnd>:
    name: "Nend"
    FloatLayout:
        GridLayout:
            size_hint: 1, 0.5
            pos_hint: {"x":0, "top":1}
            cols: 1
            Label:
                font_size: 50
                text: "Vratili ste se na \n
po\u010Detnu to\u010Dku."
        GridLayout:
            cols: 2
            size_hint: 1, 0.5
            pos_hint: {"x":0, "top":.5}
            Button:
                font_size: 30
                text: "Novi zadatak"
                on_release:
                    app.restart()
                    app.root.current = "main"
            Button:
                font_size: 30
                text: "Izlaz"
                on_release: app.stop()


<Blokiran>:
    name: "blokiran"
    FloatLayout:
        GridLayout:
            size_hint: 1, 0.5
            pos_hint: {"x":0, "top":1}
            cols: 1
            Label:
                font_size: 50
                text: " Svi hodnici\nsu blokirani."
        GridLayout:
            cols: 2
            size_hint: 1, 0.5
            pos_hint: {"x":0, "top":.5}
            Button:
                font_size: 50
                text: "Izlaz"
                on_release: app.stop()


<PrviBl>:
    name: "prvibl"
    FloatLayout:
        GridLayout:
            size_hint: 1, 0.5
```

```
        pos_hint: {"x":0, "top":1}
        cols: 1
        Label:
            font_size: 30
            text: "Nije mogu\u0107e
do\u0107i do\n          odredi\u0161ta."
        Label:
            font_size: 30
            text: "Vratite se na po\u010Detnu
to\u010Dku."
    GridLayout:
        cols: 3
        size_hint: 1, 0.5
        pos_hint: {"x":0, "top":.5}
        Button:
            font_size: 30
            text: "Povratak"
            on_release:
                app.root.current = "third"

root.manager.transition.direction = "left"
        Button:
            font_size: 30
            text: "Reset"
            on_release:
                app.restart()
                app.root.current = "main"
        Button:
            font_size: 30
            text: "Izlaz"
            on_release: app.stop()
```

```
<Krivet>:
    name: "Krivet"
    FloatLayout:
        GridLayout:
            cols: 1
            size_hint: 1, 0.5
            pos_hint: {"x":0, "top":1}
            Label:
                font_size: 30
                text: "To\u010Dke krivo unesene."
            Label:
                font_size: 30
                text: "Unesite to\u010Dke
ponovno."
        GridLayout:
            cols: 2
            size_hint: 1, 0.5
            pos_hint: {"x":0, "top":.5}
            Button:
                font_size: 40
                text: "Po\u010Detak"
                on_release: app.root.current =
"main"
            Button:
                font_size: 40
                text: "Izlaz"
                on_release: app.stop()
```