Razvoj programske podrške za Raspberry Pi mobilnu platformu namijenjenu za prikaz agenta u Wumpus svijetu

Petrović, Ilija

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

RAZVOJ PROGRAMSKE PODRŠKE ZA RASPBERRY PI MOBILNU PLATFORMU NAMIJENJENU ZA PRIKAZ AGENTA U WUMPUS SVIJETU

Završni rad

Ilija Petrović

Osijek,2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 12.09.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Ilija Petrović
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4263, 26.07.2018.
OIB studenta:	03181015610
Mentor:	Izv. prof. dr. sc. Damir Blažević
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Razvoj programske podrške za Raspberry Pi mobilnu platformu namijenjenu za prikaz agenta u Wumpus svijetu
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	12.09.2021.
Datum potvrde ocjene Odbora:	22.09.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 23.09.2021.

Ime i prezime studenta:	Ilija Petrović
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4263, 26.07.2018.
Turnitin podudaranje [%]:	6

Ovom izjavom izjavljujem da je rad pod nazivom**: Razvoj programske podrške za Raspberry Pi mobilnu platformu namijenjenu za prikaz agenta u Wumpus svijetu**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Damir Blažević

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1.UVOD 1
1.1 Zadatak Završnog rada 1
2. WUMPUS SVIJET 2
2.1 Zašto Wumpus svijet? 3
2.2 Razrada algoritma 3
3. MOBILNA PLATFORMA 15
3.1 Raspberry Pi 16
3.1 Značajke AlphaBot2 mobilne platforme 16
3.2 Spajanje na AlphaBot2 mobilnu platformu 16
3.3 Python 17
3.4 Demonstracija algoritma na robotskoj platformi 17
3.5 Primjer rada algoritma 23
LITERATURA 26
SAŽETAK 27
ABSTRACT 28
ŽIVOTOPIS 29
PRILOZI 30

1.UVOD

Moderni svijet se sve više automatizira. Od industrije do medicine, pametni roboti zamjenjuju ljude koji su nepouzdani i umaraju se tokom svog posla samim time smanjujući kvalitetu zadatka koga obavljaju. Dok ljudi posjeduju nekakav radni kapacitet, stroj ne poznaje granice doli granica programera koji ga upravlja. Samim time, roboti maksimiziraju produktivnost.

U ovom radu opisana je Raspberry Pi mobilna platforma i izrada aplikacije koja rješava problem Wumpus svijeta originalno inspiriran video igricom Gregory Yoba iz 1973. Rad će biti prikazan u tri poglavlja. U prvom poglavlju napravljen je uvod u robotiku i njenu svrhu u današnjem svijetu. Također je pojašnjeno što će se koristiti u ovom završnom radu te s kojim ciljem. U drugom poglavlju isprva će se pojasniti kako funkcioniraju elementi Wumpus svijeta. Nakon toga bit će razrađen i pojašnjen algoritam za rješavanje problema Wumpus svijeta. U trećem poglavlju bit će opisan programski jezik koji se koristi za programiranje AlphaBot2 mobilne robotske platforme te specifikacije same platforme. Nadalje u istom poglavlju implementirat će se algoritam iz drugog poglavlja na spomenutu platformu.

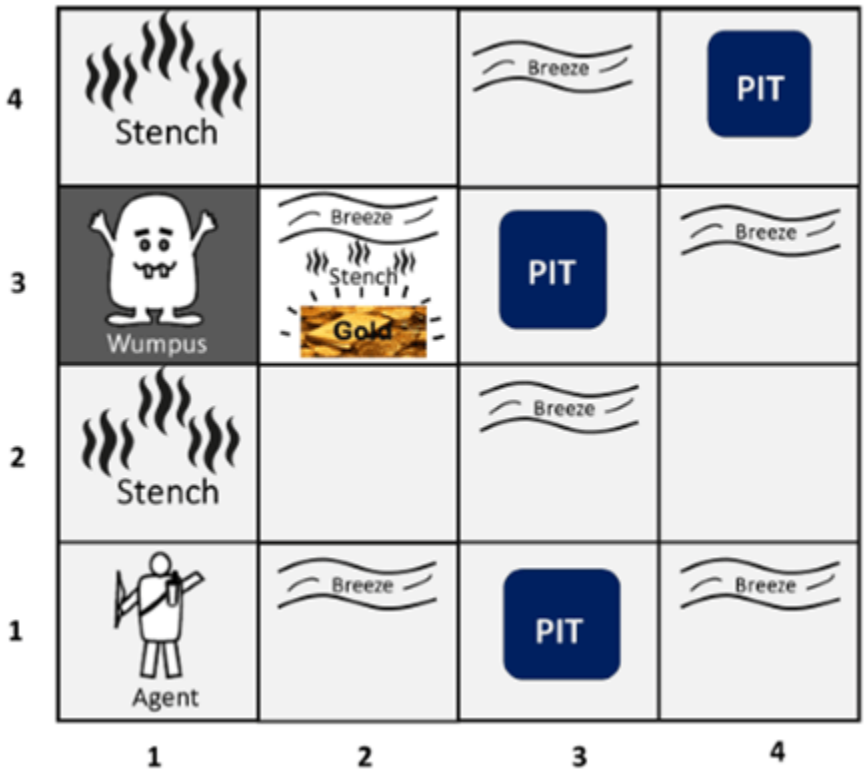
1.1 Zadatak Završnog rada

U ovom radu opisana je AlphaBot2 mobilna platforma, svijet inteligentnog agenta u Wumpus okolišu te deomnstracija algoritma za rješavanje Wumpus problema na AlphaBot2 platformu. Opisat će se razlozi proučavanja Wumpus svijeta te potrebni alati za izradu programa. Koristit će se infracrveni senzori koji će registrirati različitim bojama predstavljene elemente Wumpus svijeta. Maketa će se kretati kao agent kroz Wumpus okoliš koristeći algoritam za rješavanje Wumpus svijeta.

2. WUMPUS SVIJET

Wumpusov svijet je primjer okoliša inteligentnog agenta koji se često pojavljuje u literaturi iz područja umjetne inteligencije i inteligentnih sustava koji ilustrira vrijednost agenta temeljenog na znanju i predstavlja tumačenje stečenih informacija. Inspirirana je videoigrom Hunt the Wumpus autora Gregoryja Yoba 1973. godine.

Wumpus svijet je zapravo omeđen prostor koja ima 16 soba raspoređene u 4 reda s 4 stupca. Agent se pokušava kretati kroz svijet i pronaći zlato te ga iznijeti iz Wumpus svijeta. U tome mora izbjeći zvijer koja se nalazi u jednoj od soba, a zove se Wumpus. Također, u Wumpus svijetu postoji određeni broj prepreka u obliku jame, odnosno sobe bez dna u koje agent može upasti. Bitno je napomenuti da nije svaka generirana mapa uspješno rješiva.



Slika 2.1 *Primjer Wumpus svijeta; Izvor [2]*

Postoje neke stvari koje pomažu agentu u kretanju Wumpus svijetom:

- u prostorijama uz jamu osjeti se kretanje zraka, odnosno propuh

- u prostorijama uz Wumpusa osjeti se neugodan miris
- u sobi za zlatom vidi se sjaj zlata (agent zna ako ga je pronašao)
- agent se ne može kretati dijagonalno niti može saznati informacije o sobama Wumpus svijeta koje su mu dijagonalno

Sobe su rasporeda 4x4, agent kreće s polja [1,1] okrenut prema desnoj strani. Wumpus, jame i zlato su na nasumičnim mjestima osim početnog.

2.1 Zašto Wumpus svijet?

Wumpus svijet pruža jednostavan virtualni okoliš s ograničenim brojem mogućih poteza, kakav se može pronaći i u stvarnom svijetu. Na osnovu percipiranih stvari agent izvodi zaključke o svom okolišu. Možemo pokušati zamisliti stvarni svijet kao svijet u kom imamo ograničen broj ishoda, odnosno poteza. Inteligentni agent bi mogao u takvim uvjetima savršeno funkcionirati, u teoriji, naravno. Svi znamo da stvarni svijet nema tu osobinu determinizma, odnosno ograničenosti u broju ishoda. Često se događaju naizgled nasumične serije događaja. Jedino što možemo učiniti je predstaviti jedan mali podskup događaja iz stvarnog svijeta koga bi savršen inteligentni agent mogao razlučiti i inteligentno se kroz njega kretati. Wumpus svijet koristi se kao demonstracija primjenjivanja umjetne inteligencije u svrhu odlučivanja, zaobilaženja prepreka te pronalaska cilja (u Wumpus svijetu, zlata). Svakim potezom agent zna sve više i može mapirati svijet. Kako agent ima više informacija, može bolje rasuđivati. Problem Wumpus svijeta nije uvijek rješiv jer postoji nepotpuna percepcija svijeta. Ponekad je potrebno uraditi nasumičan potez koji ponekad može odvesti u uspješno, a ponekad u neuspješno rješenje problema.

2.2 Razrada algoritma

U ovoj relizaciji inteligentni agent bit će implementiran u obliku AlphaBot platforme, odnosno samohodne robotske platforme. Ona koristi infracrvene senzore za raspoznavanje

```
class Cave:
      pit = False
      breeze = False
      stench = False
      Wumpus = False
      pitPossibility = True
```

elemenata Wumpus svijeta, te kotače kao aktuatore u ovom slučaju. Wumpus svijet bit će realiziran kao lista listi objekata klase Cave. Potrebno je razraditi takav algoritam da sukladno infracrvenim senzorima na AlphaBot 2 platformi raspoznaje 4 boje te pomoću njih može navigirati kroz Wumpus svijet. Za početak potrebno je predstaviti jedno polje, odnosno jednu sobu Wumpus svijeta programski. Rješenje sam kreirao kao klasu „Cave“.

```
    WumpusPossibility = True
    hasGold = False
    isVisited = False
```

Slika 2.2 *Definicija klase Cave*

Klasa Cave posjeduje 8 atributa, 7 koji opisuju osobine polja vezane uz Wumpus svijet te zastavicu isVisited koja označava posjećeno polje.Wumpus svijet je kreiran kao lista listi objekata „Cave“ s obzirom na veličinu svijeta koja je ovdje 4 x 4.

```
size = 4
cave = [[Cave() for i in range(size)] for j in range(size)]
```

Slika 2.3 *Kreiranje Wumpus svijeta u kodu*

Također je bilo potrebno kreirati metode za dodavanje jama, Wumpusa te zlata, odnosno, dati tome polju i sudjednim poljima ukoliko je potrebno značajke koje omogućavaju rješavanje problema.

```
def addPit(row, column):
    row = size - row
    column = column - 1
    cave[row][column].pit = True
    if row >= 1:
        cave[row - 1][column].breeze = True
    if row <= (size - 2):
        cave[row + 1][column].breeze = True
    if column >= 1:
        cave[row][column - 1].breeze = True
    if column <= (size - 2):
        cave[row][column + 1].breeze = True
```

Slika 2.4 *Dodavanje jame u Wumpus svijet*

```
def addWumpus(row, column):
    row = size - row
    column = column - 1
    cave[row][column].Wumpus = True
    if row >= 1:
        cave[row - 1][column].stench = True
    if row <= (size - 2):
        cave[row + 1][column].stench = True
    if column >= 1:
        cave[row][column - 1].stench = True
    if column <= (size - 2):
        cave[row][column + 1].stench = True
```

Slika 2.5 *Dodavanje Wumpusa*

```
def addGold(row, column):
    row = size - row
    column = column - 1
    cave[row][column].hasGold = True
```

Slika 2.6 *Dodavanje zlata*

Također je potrebno napraviti funkciju koja ispisuje Wumpus svijet kako bi se mogao provjeriti

```
def printCave(row, column):
    for i in range(size):
        for j in range(size):
            toPrint = "-"
            if row == i and column == j:
                toPrint = "*"
            elif cave[i][j].pit:
                toPrint = "O"
            elif cave[i][j].Wumpus:
                toPrint = "X"
            elif cave[i][j].hasGold:
                toPrint = "$"
            print(toPrint + "\t", end=" ")
        print()
```

ispravan rad algoritma.

Slika 2.7 *Funkcija za debugiranje*

Nakon toga bilo je potrebno dodati jame, Wumpusa te zlato na pozicije u Wumpus svijetu koje želimo. Postavljamo početnu poziciju agenta u donji lijevi kut, postavljamo varijable koje će pratiti

```
addPit(1, 3)
addPit(3, 3)
addPit(4, 4)
addWumpus(3, 1)
addGold(3, 2)
row = 3
column = 0
rowPrevious = -1
columnPrevious = -1
```

prethodni korak agenta te brojača za broj koraka. Nadalje ispisujemo na ekran početno stanje Wumpus svijeta.

```
moves = 0
print("State at the beginning")
printCave(row, column)
```

Slika 2.8 *Početni uvjeti i stanje*

Nadalje dolazi glavna petlja algoritma. Ona je zamišljena kao while petlja koja se vrti dok agent nije pronašao zlato. Stupanjem na polje se atributi sobe označavaju kao posjećen te se mogućnost Wumpusa te jame postavlja na False.

```
while not cave[row][column].hasGold:
    cave[row][column].isVisited = True
    cave[row][column].pitPossibility = False
    cave[row][column].WumpusPossibility = False
```

Slika 2.9 *Početak glavne petlje*

Shodno tomu provjerava se sadrži li polje na koje je agent stupio propuh ili neugodan miris. Ukoliko ne, mijenjaju se sukladni atributi susjednih soba.

```
    if not cave[row][column].breeze:
        if row >= 1 and cave[row - 1][column].pitPossibility is True:
            cave[row - 1][column].pitPossibility = False
        if row <= (size - 2) and cave[row + 1][column].pitPossibility is True:
            cave[row + 1][column].pitPossibility = False
        if column >= 1 and cave[row][column - 1].pitPossibility is True:
            cave[row][column - 1].pitPossibility = False
        if column <= (size - 2) and cave[row][column + 1].pitPossibility is True:
            cave[row][column + 1].pitPossibility = False
```

Slika 2.10 *Provjera sadržavanja propuha*

```
  if not cave[row][column].stench:
        if row >= 1 and cave[row - 1][column].WumpusPossibility is True:
            cave[row - 1][column].WumpusPossibility = False
        if row <= (size - 2) and cave[row + 1][column].WumpusPossibility is True:
            cave[row + 1][column].WumpusPossibility = False
        if column >= 1 and cave[row][column - 1].WumpusPossibility is True:
            cave[row][column - 1].WumpusPossibility = False
        if column <= (size - 2) and cave[row][column + 1].WumpusPossibility is True:
            cave[row][column + 1].WumpusPossibility = False
```

Slika 2.11 *Provjera sadržavanja neugodnog mirisa*

Nakon toga uvodi se zastavica newPathFound koja se na početku postavlja na False te se traži novi put.

```
    newPathFound = False


    if row >= 1 and not ((row - 1) == rowPrevious and column == columnPrevious)
and \
            cave[row - 1][column].isVisited is False and \
            cave[row - 1][column].pitPossibility is False \
            and cave[row - 1][column].WumpusPossibility is False:
        rowPrevious = row
        columnPrevious = column
        row -= 1
        newPathFound = True
```

Slika 2.12 *Početak pronalaska novog puta*

Slično ovome provjeravaju se i ostale okolne sobe ukoliko ova soba nije pogodna za nastavak.

U slučaju da nije pronađen pogodan put za nastavak, izvršava se povratak na prethodno polje te ponavljanje postupka opisanog sljedećim kodom.

```
  if not newPathFound:
        rowPrevious, row = row, rowPrevious
        columnPrevious, column = column, columnPrevious
```

Slika 2.13 *Što se događa u slučaju da novi put nije pronađen*

Na kraju petlje dodaje se jedan na brojač poteza, ispisuje trenutno stanje Wumpus svijeta te pozicija agenta i postavlja se uvjet za prekid petlje uvjetovan brojem poteza ukoliko algoritam ne uspijeva pronaći siguran put do zlata.

```
moves += 1


    print("\n\nMove ", moves, ":")
    printCave(row, column)


    if moves > size * size:
        print("\nSolution not found.")
        break
```

Slika 2.14 *Kraj petlje*

Na kraju samog programa ukoliko je algoritam uspješan ispisuje se na ekran u koliko poteza je algoritam završio,

```
if moves <= size * size:
    print("\nGold found in ", moves, " moves.")
```

Slika 2.15 *Ispis broja poteza*

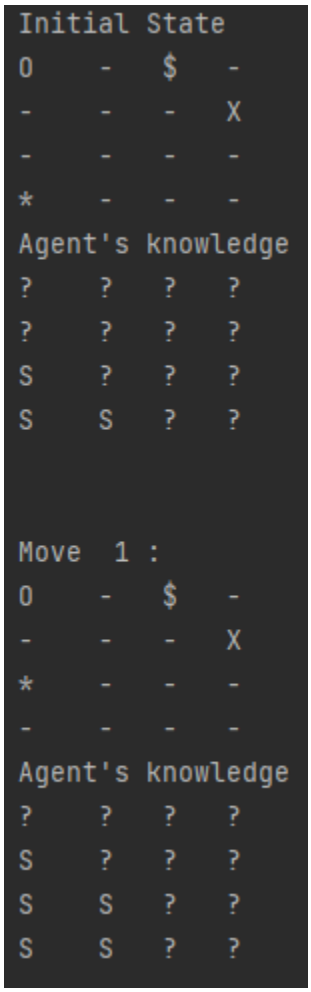
te pokreće se dio koda za izlazak iz Wumpus svijeta.

```
while not (row == 3 and column == 0):
    print()
    if row < 3:
        if row < 3 and cave[row + 1][column].isVisited is True:
            rowPrevious = row
            row = row + 1
            printCave(row, column)
            continue
    if column > 0:
        if column > 0 and cave[row][column - 1].isVisited is True:
            columnPrevious = column
            column = column - 1
            printCave(row, column)
            continue
    if row > 0:
        if cave[row - 1][column].isVisited is True:
            rowPrevious = row
            row = row - 1
            printCave(row, column)
            continue
    else:
        columnPrevious = column
        column = column + 1
        printCave(row, column)
```

***Slika 2.16** Izlazak*

Za izlazak iz Wumpus svijeta korišten je jedan od algoritama za rješavanje problema izlaska iz labirinta, „Wall follower“ algoritam. Kreće se samo po poljima koja su posjećena te radi savršeno za ovaj primjer zbog malog broja dostupnih polja.

Kako bi se algoritam mogao lakše razumjeti i pratiti, napravljena je funkcija koja ispisuje dosadašnja saznanja agenta u obliku matrice. Polja označena s „S“ označavaju sigurna polja, a polja označena s „?“ nesigurna.



Slika 2.17 *Primjer ispisa poteza i svijesti agenta*

```
def printAgentKnowledge(row, column):
    print("Agent's knowledge")
    for i in range(size):
        for j in range(size):
            toPrint = "?"
            if cave[i][j].pitPossibility or cave[i][j].wumpusPossibility:
                toPrint = "?"
            else:
                toPrint = "S"
            print(toPrint + "\t", end=" ")
        print()
```

Slika 2.18 *Kod za ispis agentovih saznanja*

Da bi se provjerila uspješnost algoritma, testiran je u for petlji od 100 iteracija s brojačem uspješnih pokušaja. Postavljeni su jama, wumpus te zlato na nasumična mjesta u pećini.

```
    while True:
        randomX = random.randint(1, 4)
        randomY = random.randint(1, 4)
        print(randomX, randomY)
        if not ((randomX == 1 and randomY == 2) or (randomX == 2 and
randomY == 1)):
            addPit(randomX, randomY)
            break
```

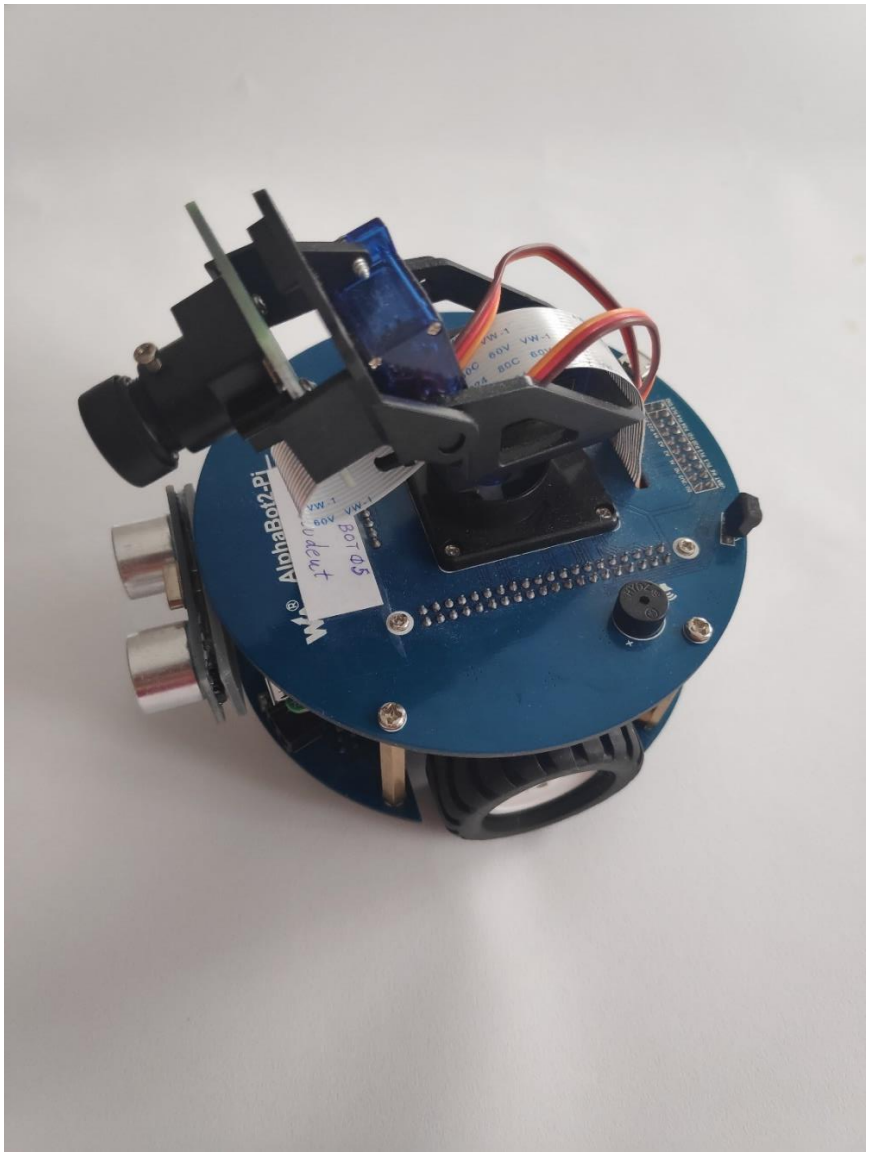
```
  while True:
        randomX = random.randint(1, 4)
        randomY = random.randint(1, 4)
        print(randomX, randomY)
        if not cave[size - randomX][randomY - 1].pit and not ((randomX == 1
and randomY == 2) or (randomX == 2 and randomY == 1)):
            addWumpus(randomX, randomY)
            break
    while True:
        randomX = random.randint(1, 4)
        randomY = random.randint(1, 4)
        print(randomX, randomY)
        if not cave[size - randomX][randomY - 1].pit and not cave[size -
randomX][randomY - 1].wumpus and not (randomX == 1 and randomY == 1):
            addGold(randomX, randomY)
            break
```

Slika 2.19 *Nasumično postavljanje elemenata*

Od 100 iteracija napisani algoritam postiže 68% uspješnosti. S obzirom na to da je određeni broj mapa teoretski nerješiv, ovaj broj je zadovoljavajući.

3. MOBILNA PLATFORMA

AlphaBot2 mobilna robotska platforma dizajnirana je za upotrebu s Raspberry Pi 3 Model B. Sadrži veliki niz uobičajenih funkcija robota, uključujući praćenje linija, izbjegavanje prepreka, Bluetooth / infracrveni / WiFi daljinski upravljač, video nadzor itd. AlphaBot2 koristi dvoslojnu strukturu koja pruža izvrsnu stabilnost i kompatibilnost : AlphaBot2-Base, donja strana, odnosno baza, te AlphaBot2-Pi, gornja ploča koja je adapter za kontroler.



Slika 3.1 *AlphaBot2*

3.1 Raspberry Pi

Raspberry Pi je relativno jeftino računalo malih dimenzija. Sastoji se iz serije malih jednopločnih računala razvijenih u Ujedinjenom Kraljevstvu od strane Raspberry Pi Foundation-a i Broadcoma. Projekt je izvorno bio namijenjen promoviranju učenja računarstva u školama i zemljama u razvoju. Originalni model bio je popularniji nego je očekivano zato što je prodavan i izvan ciljnog tržišta, kao npr. za robotiku. Koristi se u mnogim područjima, robotici, edukaciji, automatizaciji, praćenju vremena itd. zbog niske cijene, visoke modularnosti i otvorenog dizajna. Raspberry Pi jedan je od najprodavanijih britanskih računala. Do svibnja 2021. godine prodano je preko 40 milijuna računala. Većina ih je proizvedena u Sony-evoj tvornici u Pencoed-u u Walesu, dok se ostatak proizvodi u Kini i Japanu.

3.1 Značajke AlphaBot2 mobilne platforme

Posjeduje quad-core procesor s frekvencijom 1.2 GHz te 1 GB RAM. Od portova ima ethernet, HDMI, četiri utora za USB, slot za micro SD karticu te 3.5 mm za audio i video ulaz. Također ima mogućnost spajanja na WiFi te povezivanje putem Bluetooth tehnologije. Od senzora posjeduje ultrazvučni, reflektivni fotoelektrični infracrveni senzor za zaobilaženje prepreka, te reflektivni fotoelektrični infracrveni senzor za praćenje crte koji će bit korišten u ovom radu. Posjeduje LED žaruljicu koja služi kao indikator prepreke koristeći ultrazvučne senzore. Također na maketi se nalaze LED žaruljice koje je moguće upravljati programski. AlphaBot2 koristi baterije 14500 koje se nalaze u držaču za baterije u donjem dijelu makete. Na gornjem dijelu makete nalazi se RGB kamera koju je moguće pokretati horizontalno i vertikalno.

3.2 Spajanje na AlphaBot2 mobilnu platformu

Maketu je potrebno HDMI kabelom povezati na monitor. Da bi to bilo izvedivo potrebno je odvijačem odvrnuti 4 vijka na gornjem dijelu makete kako bi se došlo do HDMI porta. Također je potrebno privremeno skinuti ultrazvučne senzore s prednje strane kako bi se došlo do USB portova u svrhu povezivanja tipkovnice. Nakon boot-anja robota potrebno se ulogirati odgovarajućim podacima. Nadalje, potrebno je koristiti komandu *sudo raspi-config* kako bi se robot povezao na WiFi mrežu. Nakon toga potrebno je komandom hostname -I saznati ip adresu makete. Nakon toga na maketu se može povezati bežično preko računala programom za povezivanje, kao npr. Putty koristeći maločas saznanu ip adresu. Za prebacivanje datoteka potreban je nekakav program za File Transfer, kao npr. WinSCP.

3.3 Python

Python je programski jezik koji je moguće koristiti za vrlo široku lepezu primjena. Interpretirani je jezik, te spada u jezike visoke razine. Razvijen je 1990. godine od strane Guido von Rossuma. Ime je dobio po televizijskoj seriji „Monty Python's Flying Circus“. Python se koristi za web aplikacije, operativne sustave, umjetnu inteligenciju, strojno učenje, razvoj mobilnih aplikacija i video igre. Python ima automatsku memorijsku alokaciju te je po tome sličan programskim jezicima kao što su Perl, Ruby i Smalltalk. Pythonova filozofija dizajna uvelike naglašava čitljivost koda i upečatljivo uvlačenje koda nasuprot korištenja zagrada. Upravo iz tih razloga često se preporuča ljudima koji se tek upoznaju s programiranjem jer je samo poznavanje engleskog jezika već dovoljno da bi se određeni dijelovi Python koda razumjeli. Osim toga izuzetno je fleksibilan što se tiče stila pisanja, pa je tako moguće koristiti objektno orijentirano, strukturno i aspektno orijentirano programiranje. S vremenom postaje sve popularniji. Najviše se koristi na Linuxu, na kome je upravo temeljen Raspbian operacijski sustav.

3.4 Demonstracija algoritma na robotskoj platformi

Robotska mobilna platforma se treba kretati na papiru određene veličine obojenog na takav način da vizualno predstavlja Wumpus svijet te demonstrira algoritam koji je na njega implementiran. Koristeći pokretnu mobilnu platformu upravljajući preko gpio pinova robot navigira na komadu papira izbjegavajući jame, Wumpusa te pronalazeći zlato. U tu svrhu koristit će se datoteke TRSensors.py za upravljanje infracrvenim senzorom ITR20001/T za praćenje crte te AlphaBot2.py koja je datoteka za upravljanje kotačima na maketi. Ove datoteke dostupne su na

```
import time
import AlphaBot2
import TRSensors
```

izvoru [5] u literaturi. Da bi se koristile potrebno ih je uvesti u projekt sljedećim kodom.

Slika 3.2 *Uvoz potrebnih datoteka*

Također je potrebno napraviti instancu klasa potrebnih za rad.

```
TR = TRSensors.TRSensor()
Ab2 = AlphaBot2.AlphaBot2()
```

Slika 3.3 *Instanciranje potrebnih klasa*

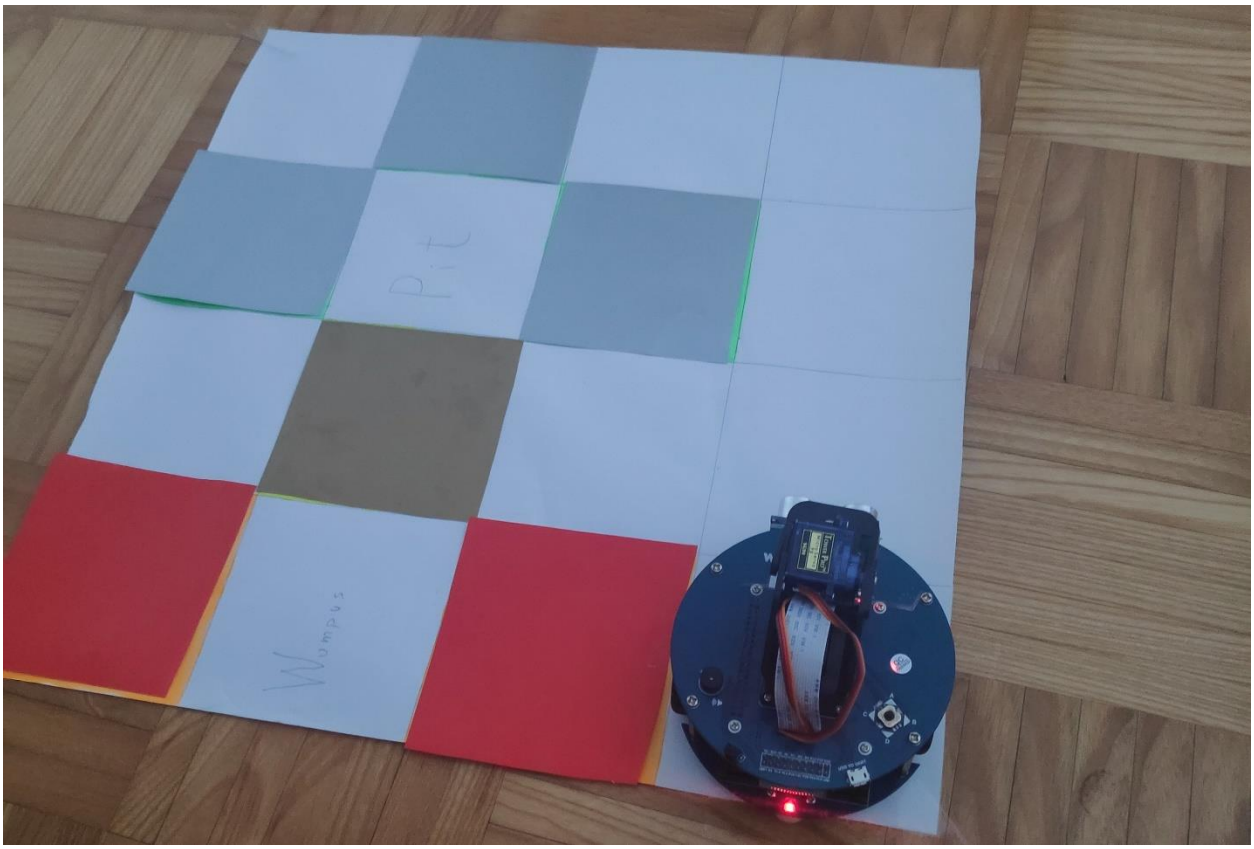
Algoritam za rješavanje Wumpus svijeta ostaje isti, samo je potrebno iskoristiti infracrvene senzore za prepoznavanje dijelova Wumpus svijeta te kotače za kretanje. Iako je senzor ITR20001/T namijenjen za praćenje crte, on zapravo vraća vrijednosti u rasponu [0, 1024] koje predstavljaju razine refleksije onoga što se nalazi ispod robota (jer su senzori s donje strane makete). U teoriji bi npr. vrijednosti 800-900 trebala davati bijela KT ploča, dok bi crna trebala davati vrijednosti u rasponu [100, 300], a praznina [0, 100]. Prvobitna zamisao bila je da je potrebno prepoznavati 4 boje; propuh od jame, miris Wumpusa, slobodno polje te zlato. U tu svrhu ideja je bila koristiti:

1. 0% crna (bijela) 0
2. 33% crna
3. 66% crna
4. 100% crna 1024

Problem osvjetljenja (različita osvjetljenja utječu na razinu refleksije i neke nepreciznosti zamišljeno je riješiti na ovaj način; podijeliti vrijednosti senzora u intervale; 0% do 20% tada je to 1, 21% do 45% tada je 2, 46% do 80% tada je 3, 81% do 100% tada je 4.
U praksi se pokazalo da bijeli papir daje vrijednosti 450-500, crni oko 350, a sivi čak oko 950. Bilo je potrebno izmjeriti vrijednosti senzora na oko desetak boja kako bi se pronašle 3 boje uz bijelu (jer je podloga bijela) koje su dovoljno udaljene po vrijednosti da bi se mogle razlikovati. Zanimljivo je da je oko polovice boja davalo vrijednosti [630, 650]. Za mjerenje korištena je metoda TRSensors.AnalogRead() koja vraća listu od 5 vrijednosti koje odgovaraju 5 senzora koji se nalaze s donje strane. Definirao sam metodu averageSensorValue() koja vraća prosjek tih vrijednosti.

```
def averageSensorValue():
    return sum(TR.AnalogRead()) / len(TR.AnalogRead())
```

Slika 3.4 *Metoda za prosječnu vrijednost senzora*



Slika 3.5 *Primjer izgleda demonstracije*

Nadalje, bilo je potrebno iskoristiti metode iz AlphaBot2.py; move.forward(), move.backward(), move.left(), move.right(), move.stop() kako bi se robot kretao određenu udaljenost, te okretao na mjestu za 90 ili 180 stupnjeva. Iskorištena je funkcija time.sleep() iz datoteke time kako bi se napisale sljedeće funkcije.

```
def turnRight():
    Ab2.right()
    time.sleep(0.36)
    Ab2.stop()
def goForward():
    Ab2.forward()
    time.sleep(0.25)
    Ab2.stop()
def turn180():
    Ab2.right()
    time.sleep(0.5)
    Ab2.stop()
def turnLeft():
    Ab2.left()
    time.sleep(0.36)
    Ab2.stop()
```

Slika 3.6 *Funkcije za kretanje makete*

Vrijednosti predane funkciji time.sleep() dobijene su empirijski, odnosno eksperimentalno.

Promijenjen je uvjet za while petlju, te način određivanja propuha, odnosno neugodnog mirisa.

```
def hasGold():
    if averageSensorValue() > 700 and averageSensorValue() < 900:
        return True
    return False
```

Slika 3.7 *Metoda za prepoznavanje zlata*

Potrebno je u svakoj iteraciji petlje znati orijentaciju robota, a to je urađeno na sljedeći način.

```
if rowPrevious - row == 1:
      orientation = 1
elif row - rowPrevious == 1:
      orientation = 3
elif columnPrevious - column == 1:
      orientation = 2
    elif column - columnPrevious == 1:
        orientation = 0
```

Slika 3.8 *Određivanje orijentacije robota*

```
if 750 > averageSensorValue() > 550:
      cave[row][column].stench = True
elif averageSensorValue() > 900:
      cave[row][column].breeze = True
```

Slika 3.9 *Interpretacija dobivenih vrijednosti*

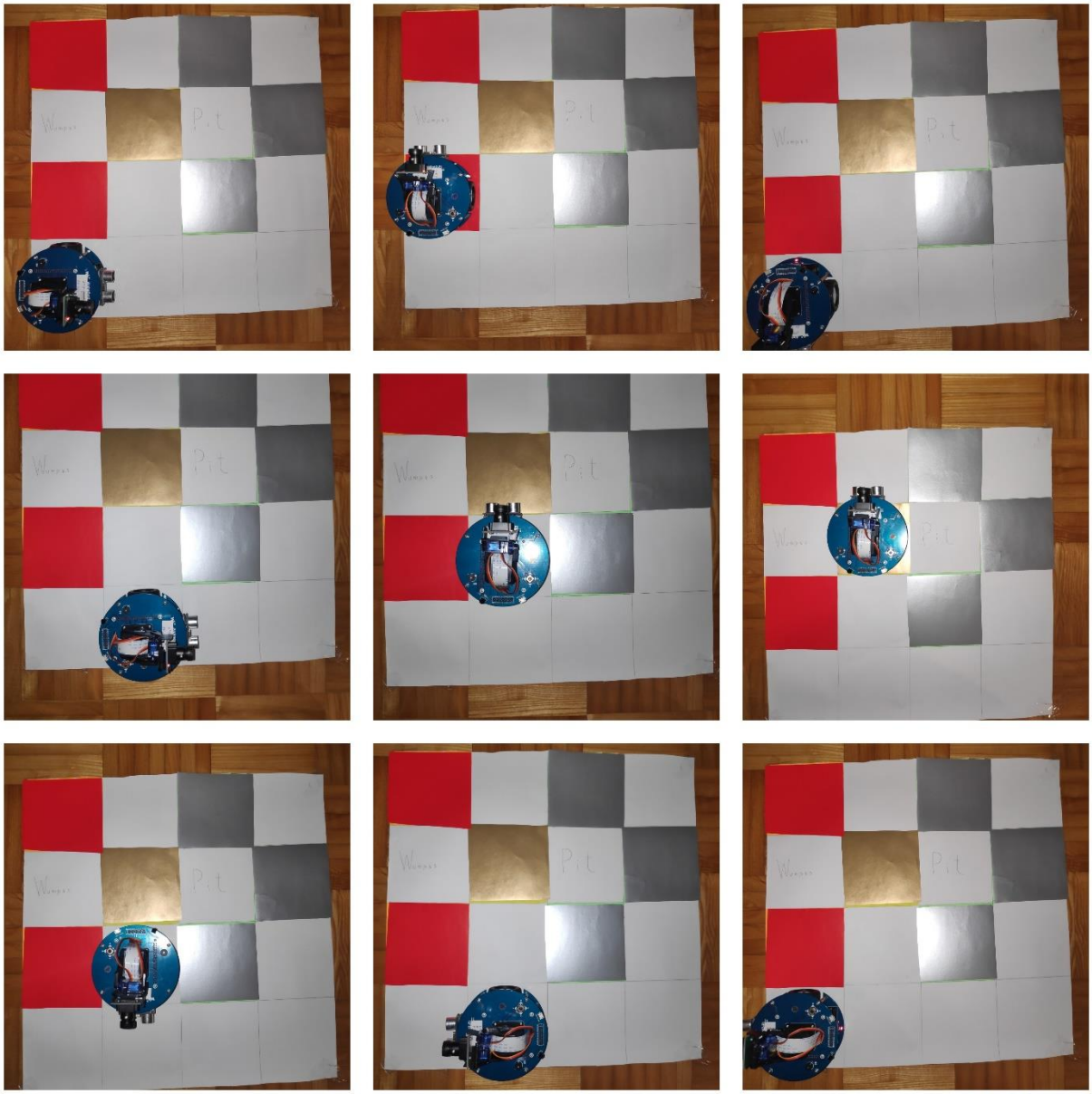
Daljnji tok algoritma ostaje isti kao u prethodnom poglavlju, do dijela gdje je potrebno na maketi prikazati potez agenta, a to je urađeno na sljedeći način.

```
    time.sleep(1)

    if rowPrevious - row == 1 and orientation == 0 or columnPrevious -
column == 1 and orientation == 1 or row - rowPrevious == 1 and orientation
== 2 or column - columnPrevious == 1 and orientation == 3:

        print("turn left, move forward")

        turnLeft()

        goForward()

    elif column - columnPrevious == 1 and orientation == 0 or rowPrevious -
row == 1 and orientation == 1 or columnPrevious - column == 1 and
orientation == 2 or row - rowPrevious == 1 and orientation == 3:

        print("go forward")

        goForward()

    elif row - rowPrevious == 1 and orientation == 0 or column -
columnPrevious == 1 and orientation == 1 or rowPrevious - row == 1 and
orientation == 2 or columnPrevious - column == 1 and orientation == 3:

        print("turn right, go forward")

        turnRight()

        goForward()

    elif columnPrevious - column == 1 and orientation == 0 or row -
rowPrevious == 1 and orientation == 1 or column - columnPrevious == 1 and
orientation == 2 or rowPrevious - row == 1 and orientation == 3:

        print("turn 180 degrees, go forward")

        turn180()

        goForward()
```

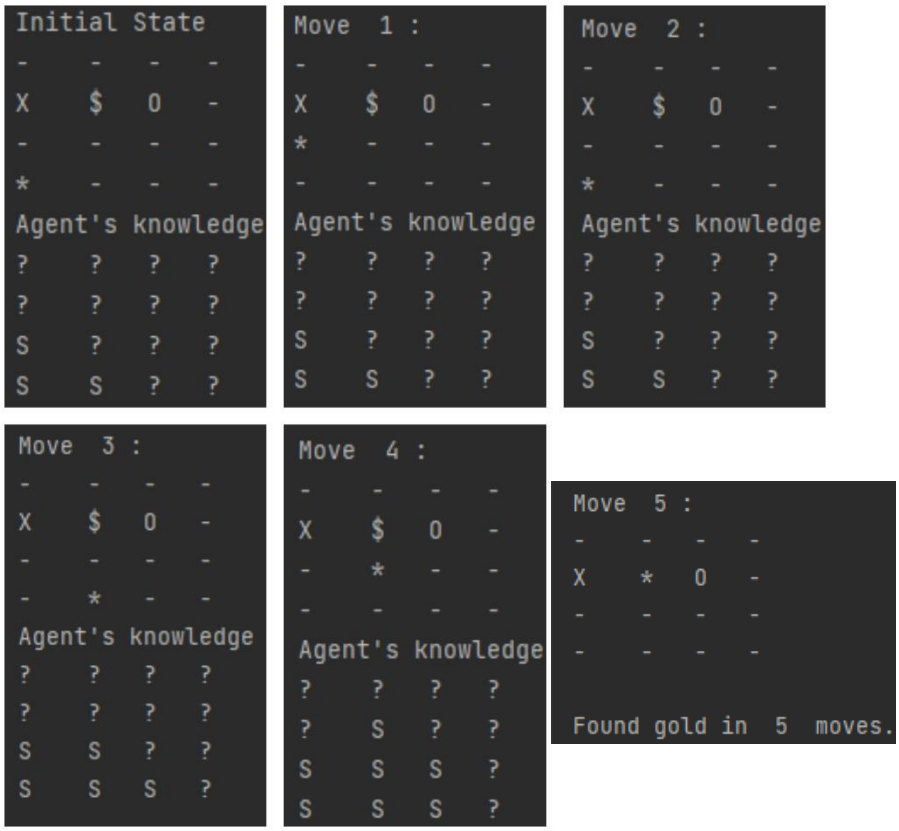
Slika 3.10 *Pokretanje makete*

3.5 Primjer rada algoritma

Demonstrirat će se rad algoritma na primjeru sa slike 2.17. Crvena boja predstavlja neugodan miris Wumpusa, siva boja predstavlja propuh od jame, a zlatna boja zlato.



Slika 3.11 *Kretanje makete*



Slika 3.12 *Ispis stanja na ploči i agentovog znanja*

Isprva agent označava polja 2,1 i 1,2 sigurnim. Kreće se na polje 2,1 gdje detektira vrijednost koja odgovara boji koja predstavlja neugodan miris Wumpusa. Vraća se na početno polje i nastavlja u drugom smjeru u nastojanju da izbjegne Wumpusa. Odabire polje 2, 2 gdje detektira polje kao sigurno te označava okolna polja sigurnim za prolazak. Nadalje stupa na 3,2 gdje pronalazi zlato, ispisuje koliko poteza mu je trebalo te se pokreće algoritam za izlazak iz špilje koristeći samo polja koja su već posjećena. Na svakom potezu se također ispisuje što agent zna o Wumpus svijetu.

4. ZAKLJUČAK

AlphaBot2 je izvrsna platforma s velikim brojem mogućnosti savršena za uvod u procesno računarstvo. Činjenica je da će u budućnosti sve se više koristiti roboti, pa čak i u svakodnevnom životu. Roboti kao AlphaBot2 nam dopuštaju da spoznamo moć koju je moguće imati s nekoliko senzora i pokretnim dijelovima. Maketa je dovoljno dobra da neke stvari radi i bolje od ljudi, kao npr. određivanje udaljenosti pomoću ultrazvučnih senzora, dok funkcije koje su napisane za kretanje ne određuju kretanje robota jednoznačno zbog prostora za grešku na motorima koji upravljaju kotačima, proklizavanja na podlozi i sl. Mogućnosti rastu eksponencijalno s mogućnostima robota. U ovom radu testirano je rasuđivanje mobilne robotske platforme s obzirom na prijem i interpretaciju informacija iz vanjskog okoliša.

LITERATURA

[1] AlphaBot2 robot building kit for Raspberry Pi 3 Model B

Dostupno na : https://www.waveshare.com/alphabot2-pi.htm [12.07.2021.]

[2] The Wumpus World in Artificial intelligence

Dostupno na : https://www.javatpoint.com/the-Wumpus-world-in-artificial-intelligence [13.07.2021.]

[3] Designing an Artificial Intelligence Agent to Navigate a World

Dostupno na: http://www.primaryobjects.com/2020/10/26/designing-an-artificial-intelligence-agent-to-navigate-a-world/ [16.07.2021.]

[4] Raspberry Pi - Wikipedia

Dostupno na: https://en.wikipedia.org/wiki/Raspberry_Pi [16.08.2021]

[5] File:AlphaBot2-Demo.7z

Dostupno na: https://www.waveshare.com/wiki/File:AlphaBot2-Demo.7z [18.08.2021.]

[6] Python (programming language)

Dostupno na: https://en.wikipedia.org/wiki/Python_(programming_language) [29.08.2021]

[7] Stuart J. Russell, Peter Norvig; Artificial Intelligence: A Modern Approach, 4th edition, 2004.god.

SAŽETAK

U ovom radu napisan je algoritam za rješavanje Wumpus svijeta te je napisan kod za demonstraciju istog na AlphaBot2 mobilnoj robotskoj platformi. Algoritam za Wumpus svijet napravljen je u skladu s praktičnim mogućnostima infracrvenih senzora. Testiran je isprva na matrici koja predstavlja Wumpus svijet nakon čega je algoritam implementiran na AlphaBot2 mobilnu robotsku platformu. Pojašnjeno je povezivanje na maketu te nakon proučavanja datoteka za korištenje eksperimentalno je pronađen način za kretanje robota određene udaljenosti i kuta. Empirijski su određene boje koje daju određene razine refleksije kako bi AlphaBot2 mogao prepoznavati elemente Wumpus svijeta na papiru.

ABSTRACT

In this final thesis, an algorithm was written for solving the Wumpus world problem and code was written to demonstrate it on the AlphaBot2 mobile robotic platform. The algorithm for the Wumpus world is made in accordance with the practical capabilities of infrared sensors. It was first tested on a matrix representing a cave after which the algorithm was implemented on an AlphaBot2 mobile robotic platform. Connecting to the model was explained, and after studying the files for use, a way was found to experimentally move the robot at a certain distance and angle. Empirically, certain colors were determined that give certain levels of reflection so that AlphaBot2 can recognize the elements of the cave on paper.

ŽIVOTOPIS

Ilija Petrović rođen je 1999. godine u Orašju. Pohađao je Osnovnu školu Braće Radića u Domaljevcu. Srednjoškolsko obrazovanje nastavlja u Školskom centru fra Martina Nedića u Orašju upisujući smjer opću gimnaziju. Završava ju 2018. godine te se odlučuje upisati preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Trenutno završava treću godinu studija.

PRILOZI

Prilog 1: CD medij sa source kodom aplikacije