Riadenie mobilných robotov

Cvičenia

Dokumentácia k semestrálnemu zadaniu

Bc. Viktor Lučkanič

Bc. Patrik Herčút

LS 2020

Úlohy:

Úloha 1. Lokalizácia robota v prostredí

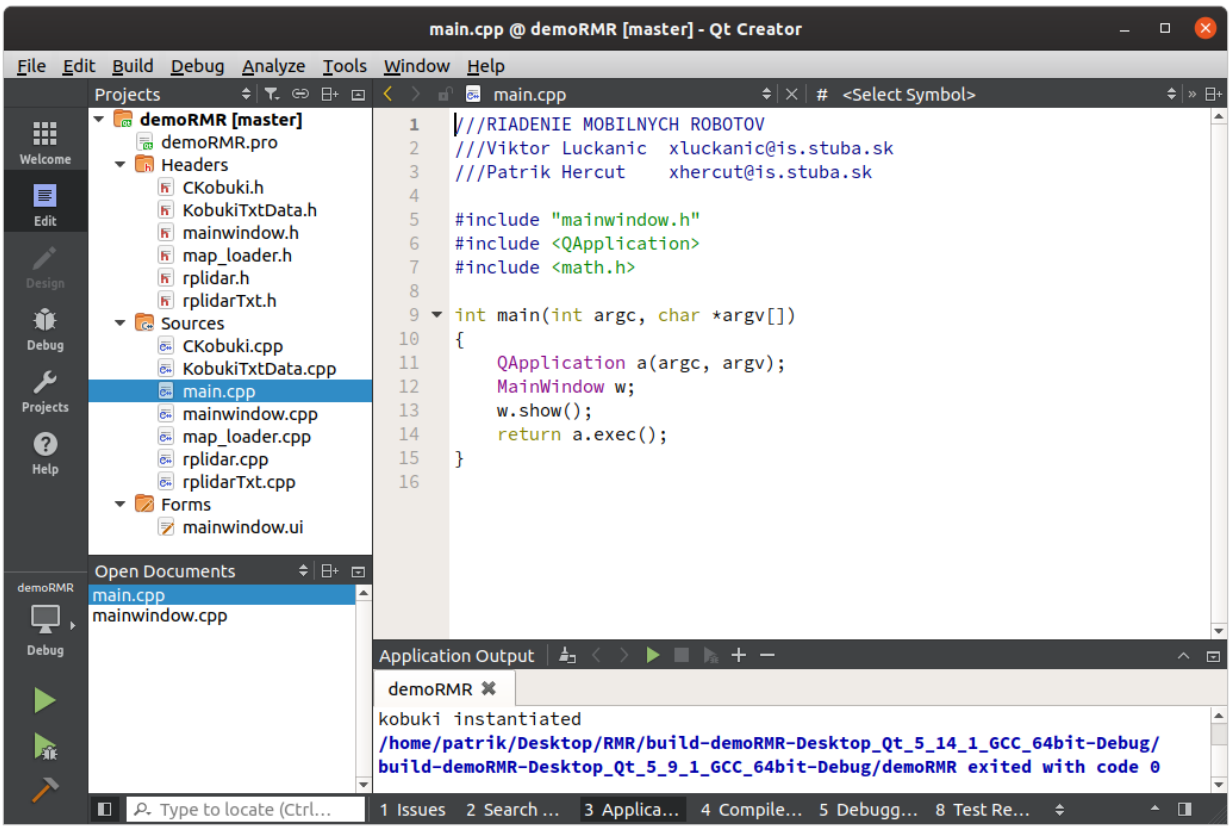
Úloha 2. Navigácia

Úloha 3. Mapovanie

Úloha 4. Plánovanie trajektórie

Program a súbory:

UBUNTU Qt Creator 5.14.1



Obr. 1 Programové okno

CKobuki

- trieda a funkcie na prácu s robotom pri bežnej UDP komunikácií

KobukiTxtData

- trieda a funkcie na čítanie dát robota z textového súboru

mainwindow

- trieda a funkcie na prácu s aplikačným oknom
- vo funkciách a callbackoch tejto triedy je implementovaná celá logika a všetky úlohy zadania

map_loader

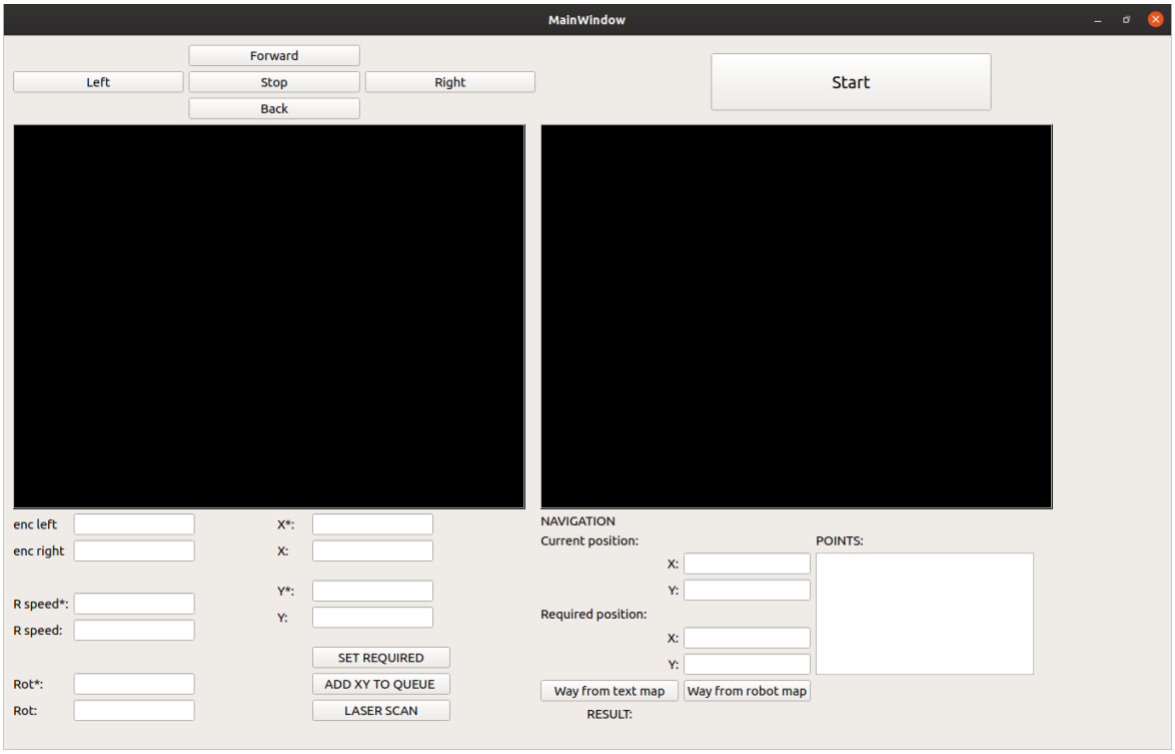
- trieda a funkcie na prácu s ideálnou mapou

rplidar

- trieda a funkcie na prácu s laserovým skenerom pri bežnej komutácií cez UDP

rplidarTxt

- trieda a funkcie na čítanie dát lidaru z textového súboru



Obr. 2 Aplikačné okno

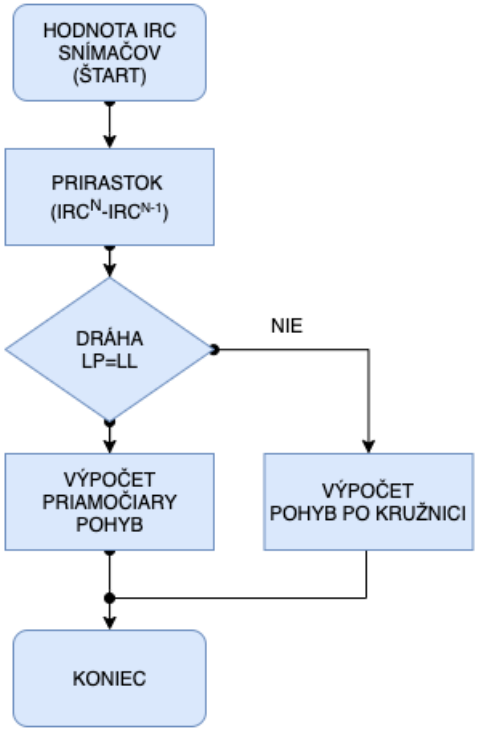
Tlačidlá *Forward, Left, Stop, Right* a *Back* - priame posielanie príkazov do robota pri UDP komunikácií.

Tlačidlo *Start*: spustenie čítania dát z textových súborov.

Ľavé okno: zobrazenie aktuálnej snímky z lidaru. Pod oknom sa zobrazujú aktuálne hodnoty enkodérov, polohy a natočenia robota (úloha1 lokalizácia). Možnosť pridania požadovaných parametrov do kolónky*, následne stačiť *SET REQUIRED.* V prípade reálneho pohybu robota, možnosť vytvorenia laserového snímku *LASER SCAN.*

Pravé okno: zobrazenie globálnej mapy (úloha 3. mapovanie). Zobrazenie polohy robota a polohy cieľa (červené bodky). Modrá bodka - smer vyhnutia sa prekážke (úloha 2. navigácia), ak sa nachádza v smere cieľa. Pod oknom je možné zadať súradnice štartu a cieľa (úloha 4. plánovanie) a vygenerovať tranzitné body do cieľa z ideálnej mapy alebo z mapy robota (mapa robota sa vytvorí až po načítaní všetkých dát z lidardata.txt). Tranzitné body sa zobrazia v okne POINTS. Mapa ohodnotená záplavovým algoritmom sa uloží do súboru mapa_robot.txt alebo mapa_ideal.txt.

Úloha 1 Lokalizácia:



Použité funkcie:

```
//výpočet odometrie
void MainWindow::processLocalization()
```

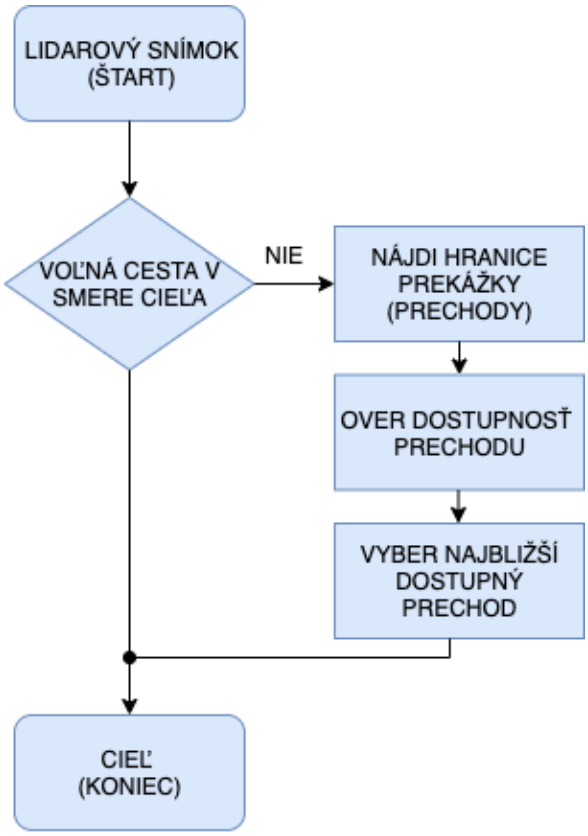
-volanie v každom cykle získania nových hodnôt z IRC snímačov

-ošetrenie pretečenia snímačov

-rozdelenie výpočtu na sériu priamočiarych pohybov alebo pohybu po kružnici

-uloženie aktualizovaných hodnôt polohy a natočenia robota do dátovej štruktúry

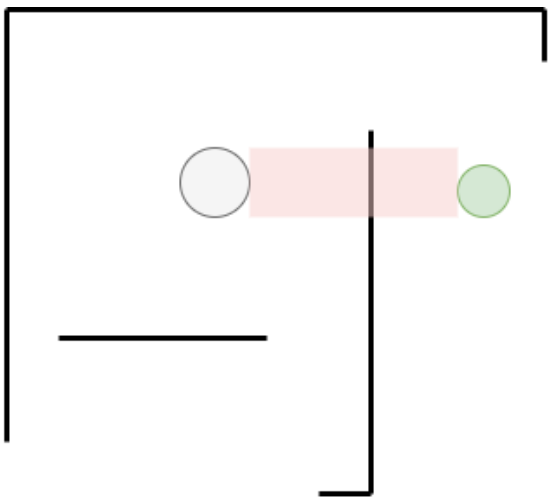
Úloha 2 Navigácia:



Použité funkcie:

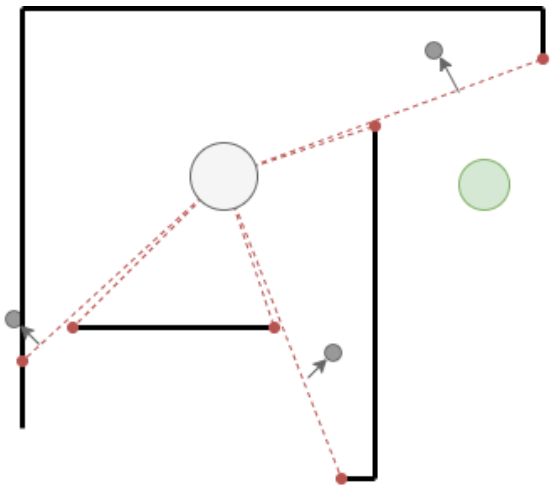
po príchode novej snímky lidaru sa vykonajú nasledovné funkcie:

```
//deteguj prekážku v smere cieľa
void MainWindow::checkWay(LaserMeasurement &laserData)
```

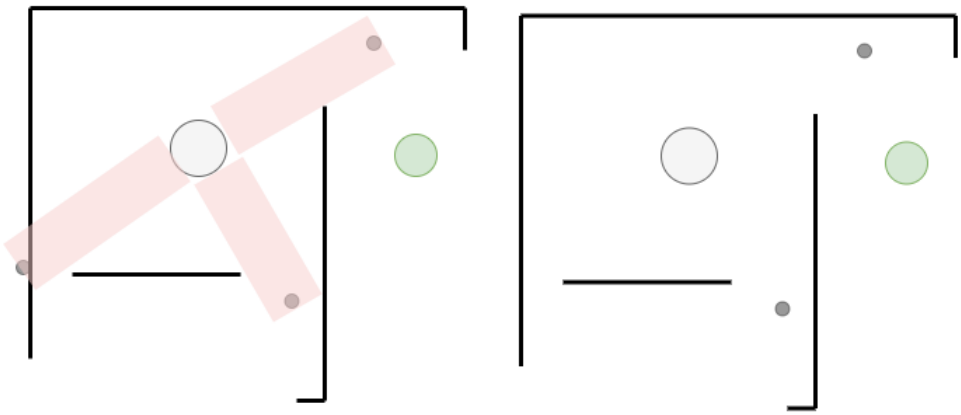


-ak sa nedetegovala kolízia, koniec, inak

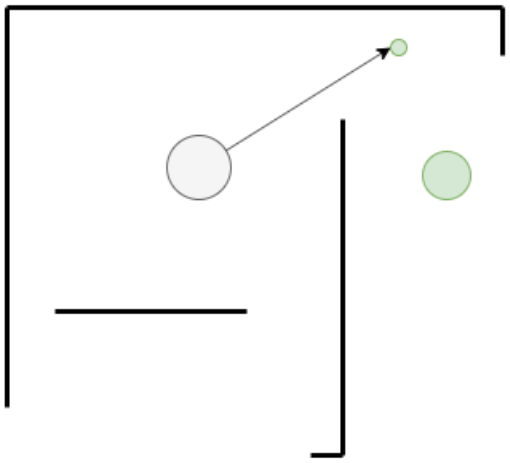
```
//nájdi všetky prechody
void MainWindow::findTransitions(LaserMeasurement &laserData)
```



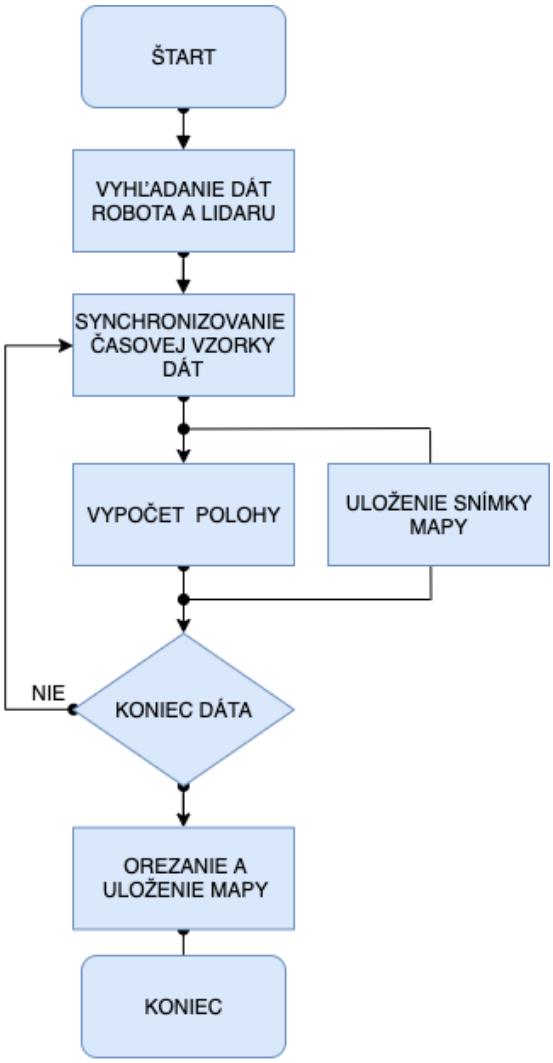
```
//over prechody dostupnosť do prechodov
void MainWindow::checkTransitions(LaserMeasurement &laserData)
```



```
//vyber najlepší prechod
void MainWindow::choseTransition()
```



Úloha 3 Mapovanie:



Použité funkcie:

Po stlačení ŠTART v okne programu sa vytvoria vlákna pre robot a lidar.

```
//synchronizácia času čítania z textu
mutex.lock();
t_offset=clock();
mutex.unlock();

//vytvorenie vlákien robota a lidaru
laserthreadID=pthread_create(&laserthreadHandle,NULL,&laserUDPVlakno,(void *)this);
robotthreadID=pthread_create(&robotthreadHandle,NULL,&robotUDPVlakno,(void *)this);
```

ďalej sa alokuje globálna mapa v rozmere 12x12 m v mierke 5 cm na bunku glob_map[240][240], keďže nevieme, kde v priestore sa robot nachádza, nastavíme pozíciu [x][y]= [6][6]

Obe vlákna sa časovo synchronizujú a vyhľadajú textové súbory. Vlákno robota - súbor robotdata.txt, vlákno lidaru - súbor lidardata.txt. (súbory by sa mali nachádzať v priečinku projektu *build-...-Debug*)

```
//čítanie dáta z robota v časovej vzorke
void MainWindow::readRobotSynch()
```

- synchronizované čítanie dát robotdata.txt
- výpočet odomerie

```
//čítanie dáta z lidaru v časovej vzorke
void MainWindow::readLidarSynch()
```

- snímka synchronizovaná s aktuálnou polohou a natočením

```
//zápis snímky lidaru do globálnej mapy
void MainWindow::writeMap(LaserMeasurement &laserData)
```

- zápis aktuálnej snímky do globálnej mapy

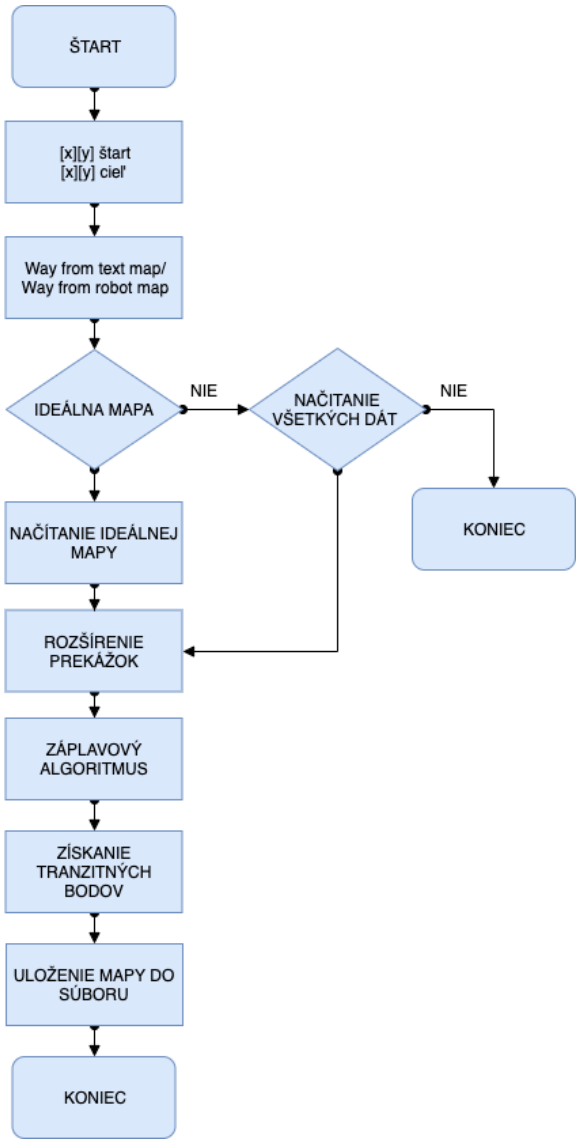
```
//zmenši mapu a ulož
void MainWindow::saveMap()
```

- po prečítaní všetkých dát z lidardata.txt
- oreže iba platné údaje mapy (alokuje sa mapa o veľkosti platných údajov v globálnej mape, približne 6x6m s mierkou 5cm na bunku, čiže mapa[120][120])

```
//zápis do textového súboru
void MainWindow::printToFile(char* file,int** map, int xSize, int ySize, bool points)
```

- uloží globálnu mapu aj orezanú platnú mapu do textových súborov mapa_cela.txt a mapa.txt

Úloha 4 Plánovanie:



Použité funkcie:

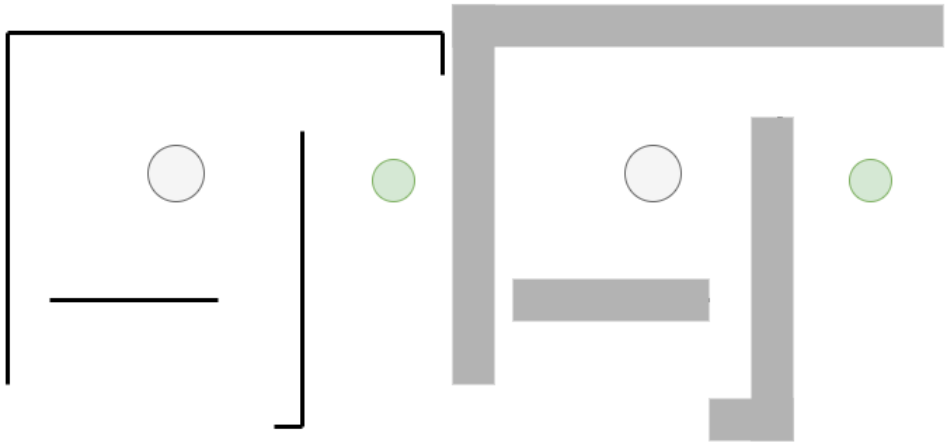
Po zadaní súradníc štartovacieho a koncového bodu (v metroch) je možné zvoliť: Way from text map/Way from robot map (Plánovanie z ideálnej mapy/ Plánovanie z vytvorenej mapy -ak už bolo dokončené čítanie mapy z lidardata.txt). Po zadaní platných súradníc štartovacieho a koncového bodu budú tranzitné body vypísané v aplikačnom okne v pravej časti (Points). Ohodnotená mapa bude uložená v súbore mapa_ideal.txt/ mapa_robot.txt.

```
//prerobí mapu z TMapArea do podoby 2D poľa
void MainWindow::fileToIdealArrayMap(char* filename)
```

- ak bola zadaná možnosť plánovanie z ideálnej mapy
- zo súboru priestor.txt načíta mapu do podoby 2D poľa, s ktorým sa ďalej pracuje (súbor by sa mal nachádzať v priečinku spolu s robotdata.txt, lidardata.txt)

//rozšírenie prekážok v mape
void MainWindow::expandObstacles(int ** map, int xSize, int ySize)

- prekážky podľa zvolenej možnosti mapy (ideálna alebo vytvorená) rozšíri o definovaný rozmer, čo zabráni kolíziám s prekážkami

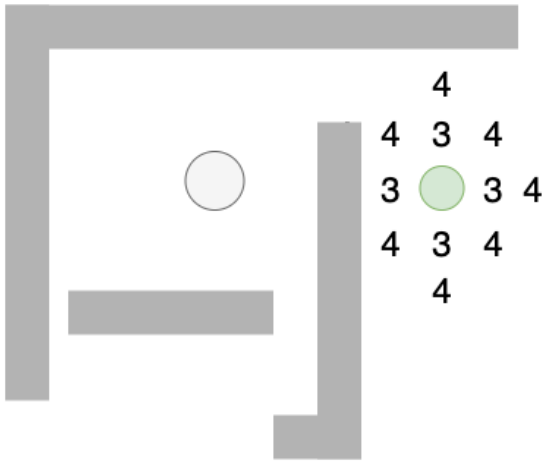


//štart > cieľ plánovanie
void MainWindow::getWayPoints(int ** map, int xSize, int ySize, double rx, double ry, double finX, double finY)

- tu sa nachádza implementovaný záplavový algoritmus

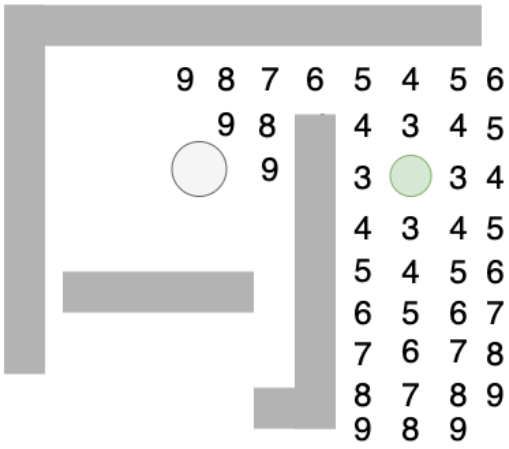
//4-susedne ohodnotenie
void MainWindow::evaluateNeighbors(int ** map, int xSize, int ySize, queue<Point>* pointsToEvaluate , int x, int y)

- v cykle prebieha 4 susedné ohodnocovanie bodov mapy až do ukončovacej podmienky



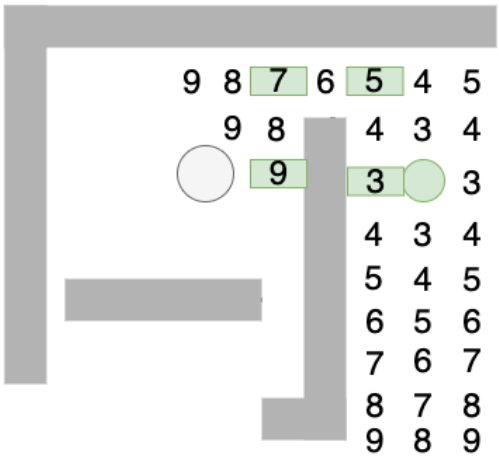
//zaplav. algoritmus - ukončovacia podmienka
bool MainWindow::foundFinish(int** map, Point p, int xSize, int ySize)

- ukončovacia podmienka záplavového algoritmu
- keď algoritmus dosiahol štartovací bod (súradnice robota) skončí sa zaplavovanie priestoru
- ak sa nesplní, algoritmus pokračuje, kým nezaplaví celý možný priestor



//určí smer ďalšieho pohybu // 1-vpravo 2-hore 3-vlavo 4-dole
int MainWindow::setDirection(int ** map, int xSize, int ySize, int x, int y, int oldDirection)

- výpočet tranzitných bodov

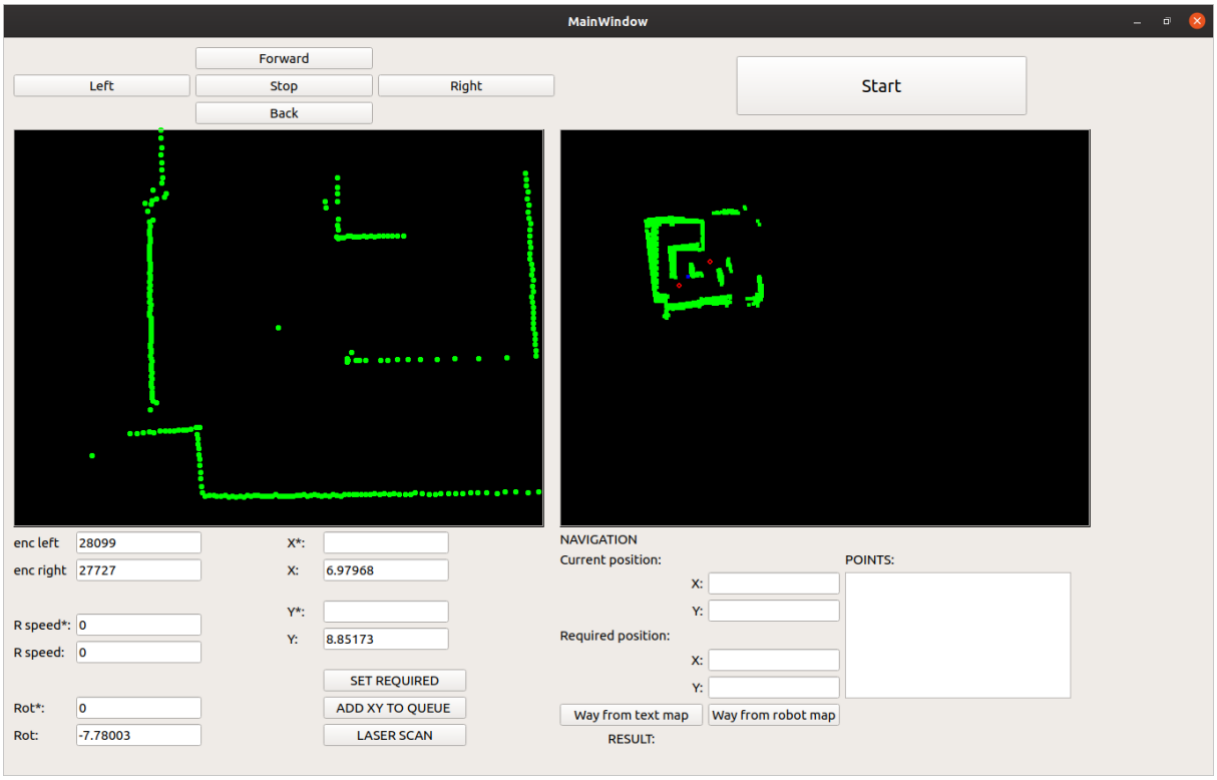


//zápis do textového súboru
void MainWindow::printToFile(char* file, int ** map, int xSize, int ySize, bool points)

- uloženie ohodnotenej zaplavenej mapy do súboru

Prílohy:

Odkaz na program:
https://github.com/Patrik-654123/RMR



Obr. 3 Aplikačné okno počas čítania textových súborov