Imię Nazwisko: Tobiasz Jakubowski
Pozycja w projekcie: Programista ROS'a

1 Narzędzia użyte przy tworzeniu projektu

1.1 ROS

ROS [1] - Platforma programistyczna do tworzenia oprogramowania sterowania robotów. Pozwala na sterowanie niskopoziomowe, komunikację międzywątkową oraz zarządzanie pakietami.

1.2 RosAria

RosAria [2] - otwarte biblioteki zapewniające interfejs dla sprzętu robota. *ROS*

1.3 SickLMS

SickLMS [3] - toolbox zapewniający obsługę lasera *SICK* umożliwiający skanowanie otoczenia przed platformą mobilną *Powerbot*.

1.4 VideoStreamOpenCV

VideostreamOpenCV [4] - zestaw oprogramowania zawierający funkcje umożliwiające przesyłanie strumienia obrazów z kamery.

1.5 VispAutoTracker

VispAutoTracker [5] - zestaw oprogramowania zawierający funkcje umożliwiające czytanie, śledzenie oraz interpretację *QR Code*.

[1] https://ros.org/
[2] https://wiki.ros.org./ROSARIA
[3] https://wiki.ros.org/sicktoolbox
[4] http://wiki.ros.org/video_stream_opencv
[5] http://wiki.ros.org/visp_auto_tracker/Fuerte

2 Założenia projektowe:

Głównym źródłem obrazu jest kamera znajdująca się w głowie (prawe oko), ma ona służyć za medium do odczytywania kodu QR.

W QR kodzie zadawane są: początkowy obrót, odległość ruchu po obrocie oraz obrót na koniec sekwencji.

Dostępne są dwa warianty programu: poruszajacy się według sekwencji z kodów QR (qr_node) oraz drugi, dodatkowo wykorzystujący dalmierz laserowy do dokładniejszego pozycjonowania (qr_lms_node).
Drugi wariant wymaga, aby wszystkie kody QR były umieszczone na ścianie, najlepiej daleko od rogów, itd.

Program działa, przełączając się między kolejnymi statusami/krokami wykonując sekwencyjnie odpowiednie działania.

2.1 Kolejne kroki obsługi:

Krok 0:
: Czekaj aż pojawi się kod QR posiadajacy dane w pożądanym formacie (trwa obrót głową), jeśli tak się stanie, to przejdź dalej.

Krok 0.5:
: Jeśli korzystamy z node'a qr_lms_node, ustaw robota prostopadle do kodu QR (patrz !!! DODAĆ LINK DO MAIN.TEX ALGORYTM GABRYSIA !!!).

Krok 1:
: Zapisz dane o aktualnym położeniu, wylicz pożądaną pozycje do następnego ruchu, przejdź dalej.

Krok 2:
: Obracaj się w odpowiednim kierunku, aż do pożądanego kąta, przejdź dalej.

Krok 2.5:
: Jeśli obrót (krok 2) zajmuje więcej niż X czasu, przerwij działanie, wróć do kroku 0.

Krok 3:
Jedź do przodu, (do tyłu, jeśli przestrzeli) aż do zadanej odległości.

Krok 3.5:
J.w. ograniczenie maksymalnego czasu jazdy.

Krok 4:
Obracaj się ponownie o zadany kąt końcowy.

Krok 4.5:
J.w. ograniczenie czasowe.

Krok 5:
Wykonano zadanie, wróc do kroku 0.

2.2 Konfiguracja konieczna do prawidłowego działania programu

Na maszynie operatora

Niezbędne jest połączenie się w tej samej sieci lokalnej, do której podłączony jest robot, jak również ROS zainstalowany w kompatybilnej z robotem wersji. Dodatkowo konieczna jest prawidłowa konfiguracja zmiennych ROS'a poprzez każdorazowe wpisanie w nowym terminalu LUB załączenie na stałe w pliku bash.rc następujących komend:

```
export ROS_MASTER_URI=http://192.168.1.108:11311/
export ROS_IP=$(hostname -I)
```

Jeśli nadal nie ma prawidłowego połączenia należy sprawdzić czy IP robota nie uległo zmianie poprzez użycie komendy hostname -I w konsoli komputera robota.
Sprawdzenie połączenia pomiędzy systemem ROS uruchomionym na komputerze znajdującym się w robocie i systemem ROS działającym na komputerze operatora odbywa się poprzez uruchomienie żoscoreńa komputerze robota, a następnie wpisanie komendy żostopic -list"w konsoli na komputerze operatora. Jeśli nie zostaje wyświetlony komunikat o błędzie połączenia z masterem, połączenie jest prawidłowe i powinno

być stabilne.

Na komputerze robota

Do prawidłowego działania należy, podobnie jak na komputerze operatora, skonfigurować zmienne ROSa. Ponieważ system ROS działający na komputerze robota jest masterem, nie jest konieczne eksportowanie mastera i wystarczy użyć (tak jak wcześniej - wpisując w konsolę lub załączając na stałe w pliku bash.rc) komendy:

```
export ROS_IP=$(hostname -I)
```

Ponadto konieczne jest zainstalowanie paczki (np. którejś z listy "użyte narzędzia") w zależności od potrzeb.

Kolejne kroki konieczne do uruchomienia programu:

1. Podłączyć sprzęt (kamera, port kom. silników), włączyć PowerBota.
2. Zalogować się zdalnie na PowerBota:

   ```
   ssh powerbot@powerbot
   ```

3. Sklonować repozytorium z kodem:

   ```
   git clone https://github.com/PowerBotSamuel/PowerSammy
   cd PowerSammy
   ```

4. Uruchomić roscore, Rosarię (obsługa ruchu) oraz SickLMS (obsługa lasera):

   ```
   roslaunch launch/powerbot.launch
   ```

   Jeśli pojawiają się błędy, należy zakończyć proces (Ctrl-C) i próbować do skutku.

5. Uruchomić obsługę kamery:
   (a) Uruchomić streaming danych z kamery:
   `roslaunch launch/labcam.launch`
   (b) Można przetestować streaming wpisując na którejś z podłączonych do sieci maszyn:
   `rosrun image_view image_view image:=/webcam/image_raw compressed`
6. Uruchomić obsługę kodów QR:

(a) Uruchomić paczkę wykrywającą i dekodującą kody QR:
`roslaunch launch/visp.launch`

(b) Można przetestować streaming wpisując na którejś z podłączonych do sieci maszyn:
`rostopic echo /visp_auto_tracker/code_message`
I podstawiając pod kamerę kod QR.

7. Uruchomić program interpretujący kody QR
   (a) Stworzyć paczkę zgodnie z dowolnym tutorialem ROS'a (np. http://wiki.ros.org/ROS/Tutorials/CreatingPackage)
   (b) Skopiować kod źródłowy (src/qr.cpp, src/qr_lms.cpp) do źródeł paczki
8. Podstawić kod QR pod kamerę lub podjechać do kodu.