

Politechnika Wrocławsk

WYDZIAŁ ELEKTRONIKI

Projekt zespołowy

Sterowanie robotem mobilnym Jaguar

Prowadzący:
Dr inż. Krzysztof Arent

Studenci:
*Adam Balawender
Kamil Bogus
Dorian Janiak
Bartłomiej Kamecki
Daria Nowicka
Dawid Perdek
Mateusz Tasz*

Semestr letni 2014/2015

Spis treści

1 Opis projektu	2
1.1 Problem projektu	2
1.2 Plan pracy i rozkład w czasie	2
1.3 Diagramy Gantta i PERT	3
1.4 Doręczenie	4
1.5 Zarządzanie projektem	5
2 Zespół	5
2.1 Adam Balawender	5
2.2 Kamil Bogus	5
2.3 Dorian Jamiak	6
2.4 Bartłomiej Kamecki	6
2.5 Daria Nowicka	6
2.6 Dawid Perdek	6
2.7 Mateusz Tasz	6
3 Przegląd istniejących rozwiązań autonomicznego sterowania robotów mobilnych	7
4 Raport K1 - Instrukcja wdrożenia robota Jaguar	8
5 Raport K2 - Montaż komputera pokładowego wewnętrz robota	10
5.1 Rysunek techniczny	10
5.2 Efekt końcowy	11
5.3 Instrukcja połączenia z komputerem pokładowym	17
6 Raport K3 - Opis standardu wiadomości GPS i możliwości nawigacji robota	18
6.1 Format wiadomości NMEA	18
6.2 Opis paczek GPS i nawigacji	18

1 Opis projektu

1.1 Problem projektu

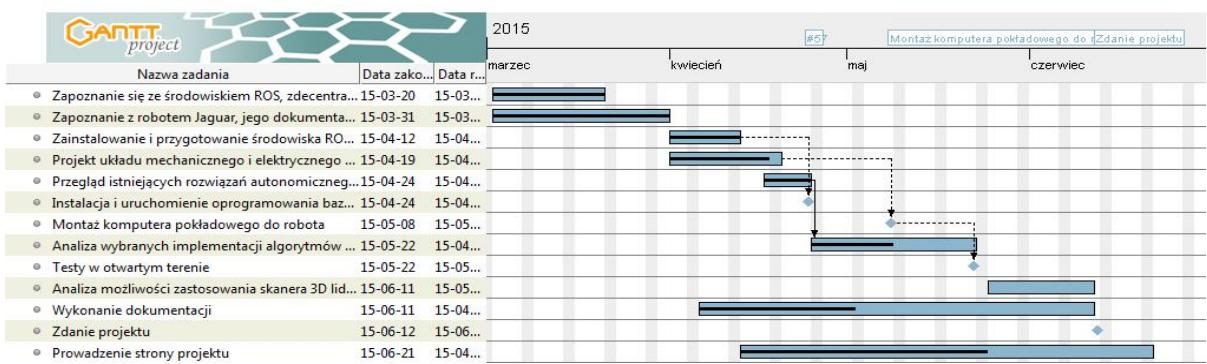
Celem realizacji projektu jest dołożenie cegiełki do rozwoju robotyki. Opracowywanym zagadnieniem będzie należący do Politechniki Wrocławskiej robot mobilny Jaguar. Robot został zakupiony przez uczelnię w 2014. roku i jeszcze nikt we Wrocławiu nie miał okazji pracy z nim. Realizujący ten projekt, będą pierwszymi którysię tego podejmą i przetrą szlaki kolejnym grupom badawczym. Obsługą i praktycznym wykorzystaniem Jaguara zainteresowana jest również firma Neurosoft. Dzięki temu praca nie musi być czysto akademicka, lecz może również uzupełnić pewną lukę w przemyśle. Dowodzi to faktu, że obiekt badań jest innowacyjny i stwarza duże pole do popisu. Celem projektu jest zrealizowanie kilku algorytmów sterowania robotem mobilnym typu Jaguar. Początkowo w wersji uproszczonej -> transport z punktu A do punktu B w warunkach laboratoryjnych. Następnym etapem jest autonomiczna jazda w otwartym terenie w oparciu o GPS. Realizacja projektu ma skutkować dostarczeniem dokładnego przeglądu możliwości robota, istniejących rozwiązań w kwestii sterowania oraz ewentualnie własnych propozycji algorytmów.

1.2 Plan pracy i rozkład w czasie

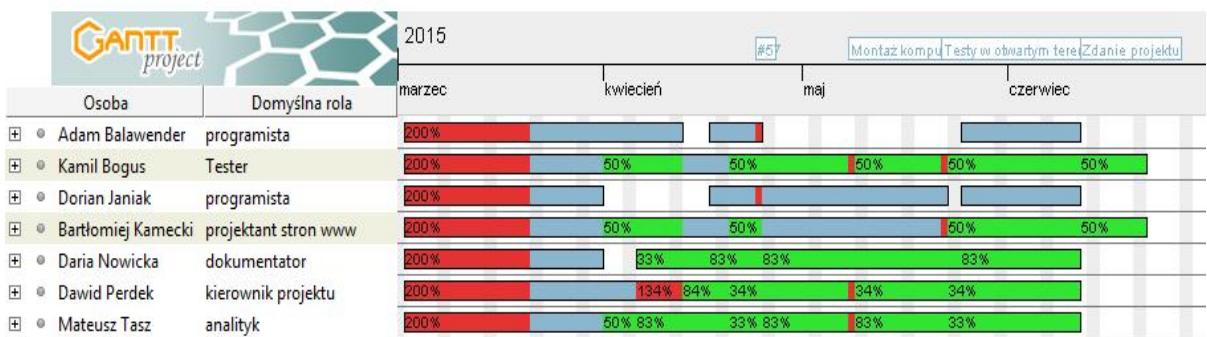
- a. Zapoznanie się ze środowiskiem ROS, zdecentralizowanym systemem kontroli wersji git oraz systemem składu tekstu LaTeX (do 20.03)
- b. Zapoznanie z robotem Jaguar, jego dokumentacją i gotowym oprogramowaniem (do 31.03)
- c. Zainstalowanie i przygotowanie środowiska ROS na komputerze pokładowym (do 12.04)
- d. Projekt układu mechanicznego i elektrycznego do integracji komputera przemysłowego z platformą Jaguar (do 19.04)
- e. Przegląd istniejących rozwiązań autonomicznego sterowania robotów mobilnych (do 24.04)
- f. Instalacja i uruchomienie oprogramowania bazowego do obsługi układu sensoryczno-wykonawczego platformy mobilnej Jaguar (K1 - do 24.04)
- g. Montaż komputera pokładowego do robota (K2 - do 08.05)
- h. Analiza wybranych implementacji algorytmów sterowania wykorzystujących GPS (do 22.05)
- i. Testy w otwartym terenie (K3 - do 22.05)
- j. Analiza możliwości zastosowania skanera 3D lidar na platformie mobilnej Jagurar i wynikające z tego konsekwencje (do 11.06)
- k. Wykonanie dokumentacji (do 11.06)
- l. Zdanie projektu (K4 - do 12.06)
- m. Prowadzenie strony projektu (do 21.06)

1.3 Diagramy Gantta i PERT

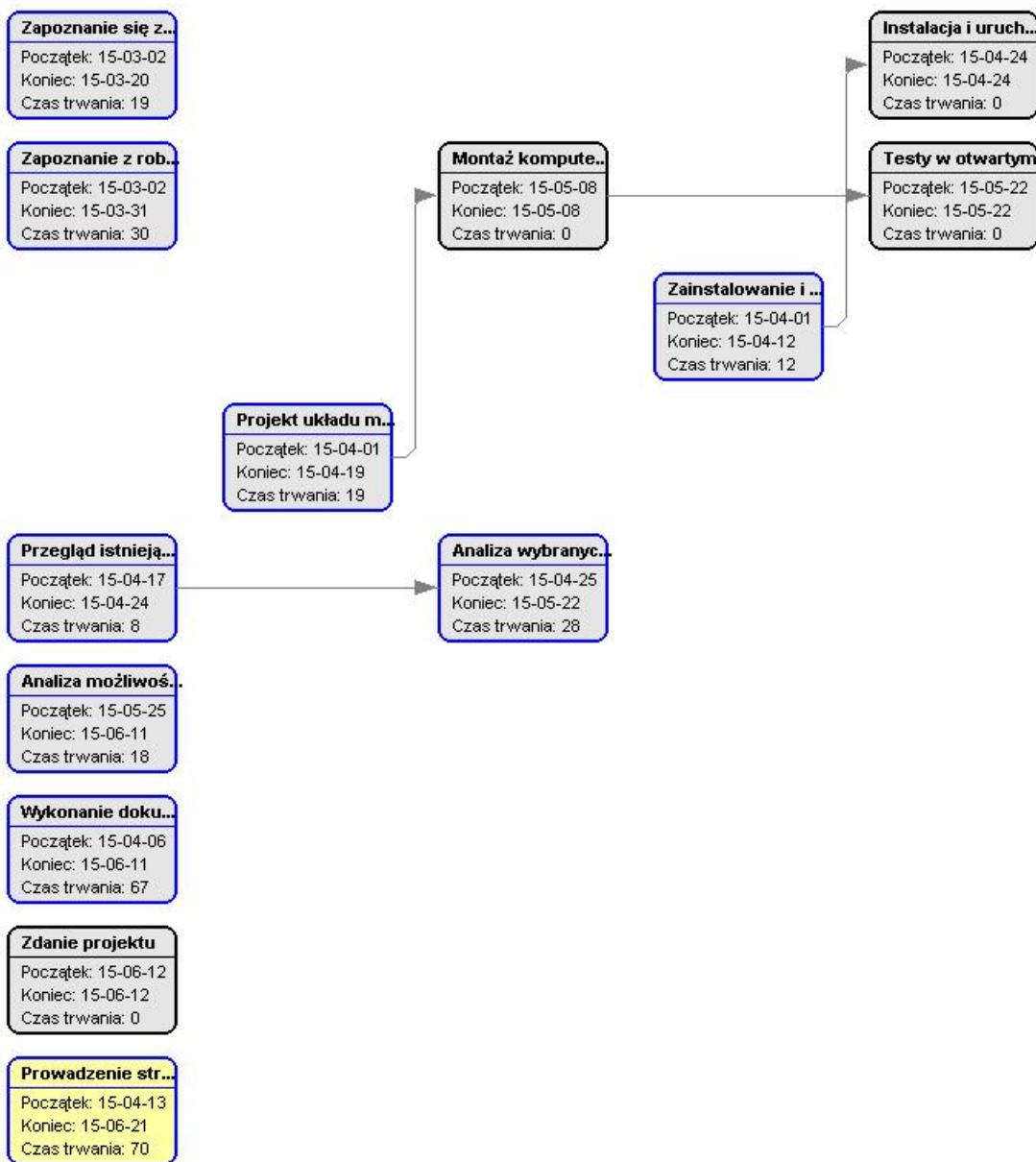
1. Diagram zadań.



2. Diagram osób.



3. Diagram PERT



1.4 Doręczenie

- K1 Raport zawierający opis sposobu komunikacji i kontroli robota oraz opis sposobu pozyskiwania odczytów z sensorów i ich formatu.
- K2 Raport złożony z rysunków technicznych, schematów elektrycznych oraz fotograficznej dokumentacji efektu końcowego.

- K3 Raport zawierający krótką analizę wybranych implementacji istniejących algorytmów sterowania oraz szczegółowy opis wykorzystanych rozwiązań i efektów ich zastosowania.
- K4 Kompletna dokumentacja przebiegu projektu. Zawierać się w niej będą wszystkie poprzednie raporty oraz analiza możliwości wykorzystania skanera 3D w platformie mobilnej Ja-guar.

1.5 Zarządzanie projektem

Spotkania odbywają się w piątki. Do każdego z zadań zostało przydzielonych od dwóch do siedmiu członków grupy, przy czym każde z zadań ma swojego lidera. W przypadku sytuacji spornych podczas realizacji poszczególnych zadań decydujący głos ma lider danego zadania. Jeśli takie podejście nie przyniesie oczekiwanej rozwiązania sprawa może zostać przedstawiona całej grupie i poddana głosowaniu. W sytuacji, gdy konieczna jest szybka decyzja istnieje możliwość rozstrzygnięcia sporu bezpośrednio przez koordynatora projektu, z pominięciem etapu głosowania. Koordynator projektu bierze udział w wyznaczonych zadaniach na zasadach takich samych jak pozostały członkowie grupy projektowej. Dodatkowo pełni on rolę osoby kontaktowej dla ludzi spoza grupy. Do obowiązków koordynatora należą również składanie raportów i koordynowanie prac całego zespołu oraz końcowa ocena członków grupy (w przypadku konfliktów również podlegająca głosowaniu). Koordynacja prac i monitorowanie ich postępów odbywać się będzie na podstawie repozytorium na GitHubie oraz grup projektu na portalu społecznościowym Facebook i w systemie ePortal, w których to na bieżąco przedstawiane i konsultowane są postępy, pytania i wątpliwości. Postępy są nanoszone przez koordynatora na diagram Gantta (lokalnie, planowane jest przeniesienie tego na platformę on-line). Każdy z członków grupy bierze udział w siedmiu spośród czternastu zadań wymienionych w harmonogramie prac, ale ma możliwość udzielania się również przy pozostałych siedmiu. W związku z takim podejściem każdy z członków grupy projektowej, który ukończy kurs „Projekt Zespołowy” z oceną pozytywną będzie miał takie same prawa własności intelektualnej do uzyskanych wyników prac.

2 Zespół

2.1 Adam Balawender

Ma doświadczenie z językami C, C++, Python oraz LaTeX. Biegły zna środowisko GNU/Linux. Oprócz tego interesuje się elektroniką oraz przetwarzaniem sygnałów i obrazów. Zna język angielski na poziomie C1. Pracuje jako programista C embedded.

2.2 Kamil Bogus

Zamiłowanie do macierzy i analitycznej analizy kinematyki robotów, potwierdzone ocenami na poziomie 5.0 z Mechaniki Analitycznej czy Robotyki 1, podobnie jeśli chodzi o umiejętności programistyczne, znajomość C, C++, C#, html oraz obsługa programów Matlab i Simulink. Znajome doświadczenie w projektowaniu i montowaniu układów elektronicznych.

2.3 Dorian Janiak

Pisze od kilku lat programy w językach C++/C (4.5 z programowania obiektowego). Podejmował się pracy z takimi językami jak Python, Matlab czy QML. Obecnie pracuje jako programista C++. Jego głównym zainteresowaniem jest grafika 3D (używa OpenGL i GLSL, zna podstawy RayTracingu). Ukończył kurs języka niemieckiego na poziomie B2 (ocena 5.0).

2.4 Bartłomiej Kamecki

Potrafi programować w języku C,C++,Matlab(Oceny z przedmiotów programistycznych w zakresie 3.5-4.5). Dobra znajomość HTML oraz oprogramowań służących tworzeniu dokumentacji takich jak MS Office oraz Latex(Ocena 5.5 z Technologii Informacyjnych oraz 4.5 SCR-Sieci operacyjne). Zagadnienia związane z robotyką takie jak kinematyka robota itp. nie sprawiają mu problemów(Mechanika Analityczna i Robotyka I zaliczone na 4.0) Dobra znajomość języka angielskiego(Angielski B2.2 zaliczony na 5.0).

2.5 Daria Nowicka

Dobrze radzi sobie z teoretycznym opisem kinematyki robota (otrzymała ocenę 5,5 z kursu Robotyka 1), lubi programować w Matlabie, nie sprawia jej również trudności obsługa toolboxa Simulink. Oczekuje, że dzięki realizacji projektu będzie miała możliwość praktycznego wykorzystania zdobytej dotychczas wiedzy.

2.6 Dawid Perdek

Studiuje również Informatykę na Wydziale Informatyki i Zarządzania. Dobrze czuje się w programowaniu, miał styczność z wieloma językami i środowiskami. Lepsze oceny otrzymywał z przedmiotów związanych z programowaniem niż z elektronicznych, ale spodobała mu się elektronika i stara się rozwijać w tym kierunku.

2.7 Mateusz Tasz

Zadania teoretyczne związane z robotyką nie sprawiają mu trudności (kurs robotyka zaliczył na 5.5). Środowisko Matlab zna w stopniu zadowalającym, a przeprowadzanie symulacji komputerowych jest dla niego przyjemnością. Potrafi programować w C/C++ (kurs zaliczony na 5,0), ale zdecydowanie bardziej podoba mu się możliwość wykorzystania tych umiejętności w programowaniu Arduino. Lubi wykonywać zadania majsterkowania.

3 Przegląd istniejących rozwiązań autonomicznego sterowania robotów mobilnych

Zagadnienie autonomicznego sterowania robotów mobilnych należy podzielić na sterowanie wewnątrz budynków oraz sterowanie w otwartej przestrzeni. W pierwszym przypadku założyć można brak poślizgów kół co ułatwia samolokalizowanie się robota oraz zapamiętywanie trasy. Tego typu rozwiązania występują na uczelni od dawna - przykładem z którym kontakt mają studenci są roboty Pioneer. Dzięki sonarom oraz informacjom z enkoderów są one w stanie poruszać się unikając zderzeń i mapować środowisko w którym operują. Analogiczne rozwiązania stosowane są w większości robotów mobilnych mających funkcjonować w pomieszczeniach. W przypadku robotów przeznaczonych do pracy poza budynkami nie można już bezgranicznie ufać odczytom z enkoderów - do pozycjonowania robota wykorzystuje się moduł GPS. Tego typu rozwiązania są rzadziej spotykane i gorzej opisane, Jaguar jest dla naszej uczelni pierwszym robotem „outdoor’owym”. Lokalizacja przy użyciu GPS ma jedną istotną wadę jaką jest mała dokładność (w porównaniu do metod wykorzystywanych w robotach „budynkowych”). Odczyt z GPS czasem wymaga konwersji na zrozumiałą dla robota informację o pozycji, w tym celu stosowany może być navsat_transform_node mający za zadanie przetworzenie danych uzyskanych od modułu GPS na ramkę przechowującą pozycję i orientację robota w zrozumiałej dla niego formie. Oprogramowanie dołączone do robota Jaguar umożliwia odczyt GPS i wskazanie lokalizacji na mapach pobranych z Google Earth. Odczyty modułu mogą być nagrywane co może posłużyć do późniejszego odtwarzania przebytej trasy. Prawdopodobnie jednak obsługa modułu GPS nie jest dostępna w udostępnionych pakietach oprogramowania i potrzebne będzie jej samodzielne napisanie.

- 1) Robot Pioneer P3-DX
- 2) Nawigacja i mapowanie robotami Pioneer
- 3) Przykład mapowania na podstawie skanerów laserowych
- 4) Tutorial navsat_transform_node
- 5) Jaguar Manual
- 6) Udostępnione pakiety oprogramowania do Jaguara

4 Raport K1 - Instrukcja wdrożenia robota Jaguar

Instalację ROS Indigo przeprowadzono na systemie operacyjnym **Ubuntu 14.04.1**, analogiczne kroki dla wersji **Ubuntu 12.04** i dystrybucji ROS Hydro również dały pożądane rezultaty.

- 1) Instalacja ROS Indigo zgodnie z instrukcją - link. Po wykonaniu wszystkich kroków należy jeszcze wprowadzić:

```
echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc
```

- 2) Przygotowanie środowiska pracy dla systemu ROS, zgodnie z instrukcją dla wersji *catkin* - link. Po wykonaniu wszystkich kroków również należy wprowadzić jeszcze:

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

- 3) Pobranie paczek z oprogramowaniem robota Jaguar do folderu *~/catkin_ws/src* - komendy:

```
cd ~/catkin_ws/src/  
git clone https://github.com/gitdrrobot/DrRobotMotionSensorDriver  
git clone https://github.com/gitdrrobot/drrobot_jaguar4x4_player
```

- 4) Kompilacja paczki *DrRobotMotionSensorDriver*:

```
cd ~/catkin_ws/src/DrRobotMotionSensorDriver/  
cmake .
```

- 5) Wstępna komplikacja paczki *drrobot_jaguar4x4_player* - aby wykonać ten krok należy edytować plik *CMakeLists.txt* z folderu paczki poprzez zakomentowanie wszystkich linii zaczynających się od *add_executable* oraz *target_link_libraries* (powinno ich być po cztery), a następnie wykonać komendy:

```
cd ~/catkin_ws/  
catkin_make
```

- 6) Ponowna komplikacja paczki *drrobot_jaguar4x4_player* - w tym celu należy przywrócić do stanu pierwotnego zakomentowane w poprzednim punkcie linie pliku *CMakeLists.txt* paczki i potem wywołać:

```
cd ~/catkin_ws/  
catkin_make
```

- 7) Pobranie i komplikacja paczki do obsługi joysticka:

```
cd ~/catkin_ws/src/  
git clone https://github.com/ros-drivers/joystick_drivers  
mv joystick_drivers/joy .  
rm -rf joystick_drivers/  
cd ..  
catkin_make
```

- 8) Pobranie węzła potrzebnego do sterowania robota za pomocą joysticka Logitech:

```
cd ~/catkin_ws/src/drrobot_jaguar4x4_player/src  
git clone https://github.com/cowiekmaupa/jaguar_joy_teleop_pwr  
mv jaguar_joy_teleop_pwr/drrobot_joy_teleop.cpp .  
rm -rf jaguar_joy_teleop_pwr
```

- 9) Dopisanie nowego węzła do pakietu *drrobot_jaguar4x4_player*. Aby to zrobić należy dokonać pewnych zmian w pliku *CMakeLists.txt* paczki - w sekcji **##Declare a cpp executable** dopisać linię:

```
add_executable(drrobot_jaguar4x4_joy_teleop_node src/drrobot_joy_teleop.cpp)
```

oraz następnie w sekcji **##Specify libraries to link a library or executable target against** dopisać:

```
target_link_libraries(drrobot_jaguar4x4_joy_teleop_node ${catkin_LIBRARIES})
```

- 10) Kompilacja paczki:

```
cd ~/catkin_ws/  
catkin_make
```

- 11) Instalacja programu potrzebnego do odczytu danych z GPS:

```
sudo apt-get install socat
```

Przed próbą uruchomienia robota należy upewnić się, że jest on włączony, komputer jest wyposażony w kartę WiFi (bądź połączony z robotem przewodem) oraz w porcie USB komputera jest poprawnie zainstalowany adapter od joysticka Logitech. Następne kroki są następujące:

- 1) Aby nawiązać połączenie należy ręcznie ustawić statyczne IP komputera (np. 192.168.0.101) i podłączyć się do sieci robota (np. DriJaguar2).

- 2) W pierwszym oknie terminala należy uruchomić węzeł nadzędny.

```
roscore
```

- 3) W drugim oknie terminala należy uruchomić główny węzeł robota, odpowiedzialny za połączenie z węzłem nadzędnym.

```
rosrun drrobot_jaguar4x4_player drrobot_jaguar4x4_player_node
```

- 4) W trzecim oknie terminala należy uruchomić węzeł odpowiedzialny za połączenie joysticka z węzłem nadzędnym.

```
rosrun joy joy_node
```

- 5) W czwartym oknie terminala należy uruchomić węzeł odpowiedzialny za sterowania robotem za pomocą joysticka.

```
rosrun drrobot_jaguar4x4_player drrobot_jaguar4x4_joy_teleop_node
```

- 6) W kolejnych oknach terminala można śledzić dowolny topic przy użyciu *rostopic* lub podglądać dane zwieracane przez GPS dzięki poleceniu:

```
socat pty,link=serial,waitslave tcp:192.168.0.61:10002
```

5 Raport K2 - Montaż komputera pokładowego wewnątrz robota

5.1 Rysunek techniczny

tutaj będą ładne rysunki

5.2 Efekt końcowy

Zastanawiając się nad sposobem montażu komputera pokładowego dla robota Jaguar rozpatrywano dwie możliwości: montaż wewnątrz oraz na zewnątrz robota. Biorąc pod uwagę, że Jaguar jest robotem przeznaczonym do jazdy w terenie, postanowiono zamontować komputer pokładowy wewnątrz jego obudowy. Jest to podyktowane względami bezpieczeństwa - dzięki takiemu rozwiązaniu, komputer jest chroniony przed wpływem warunków atmosferycznych oraz wszelkimi uszkodzeniami mechanicznymi. Ponadto takie rozwiązanie jest prostsze konstrukcyjnie, gdyż nie wymaga dużej ingerencji w obudowę robota. W przypadku zamontowania komputera na zewnątrz, konieczne byłoby wyprowadzenie przewodów z jego wnętrza, a obudowa Jaguara nie posiada zbyt wielu otworów pozwalających na taki manewr. Poza oczywistą wadą tego pomysłu, jakim jest duża inwazyjność czynności, robot mógłby stracić dużą zaletę, jaką jest szczelność obudowy.

Na rysunku 1 przedstawiono komputer pokładowy przyjmocowany do płytki pleksi.



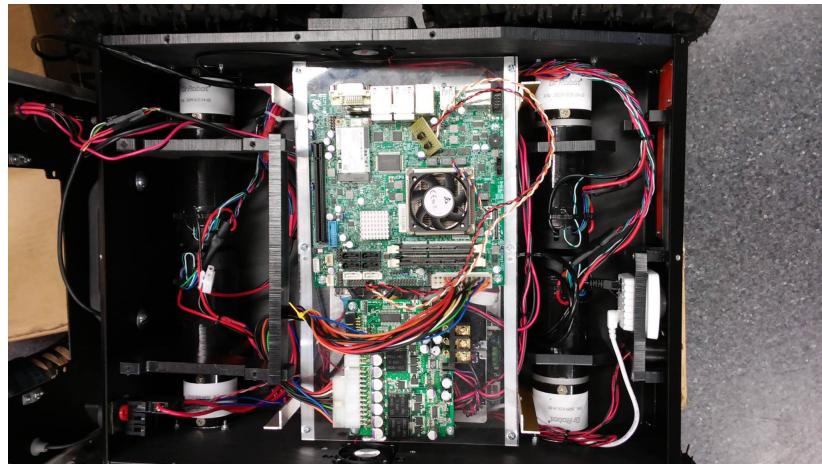
Rysunek 1: Komputer pokładowy na płytce pleksi

Na poniższym zdjęciu komputer jest już prawidłowo podłączony. Poszczególne połączenia zostaną szczegółowo opisane w dalszej części raportu.



Rysunek 2: Komputer pokładowy z prawidłowo połączonymi elementami

Kolejne zdjęcie przedstawia komputer umieszczony wewnętrz robotu (widok jest obrócony o 90° względem poprzednich).

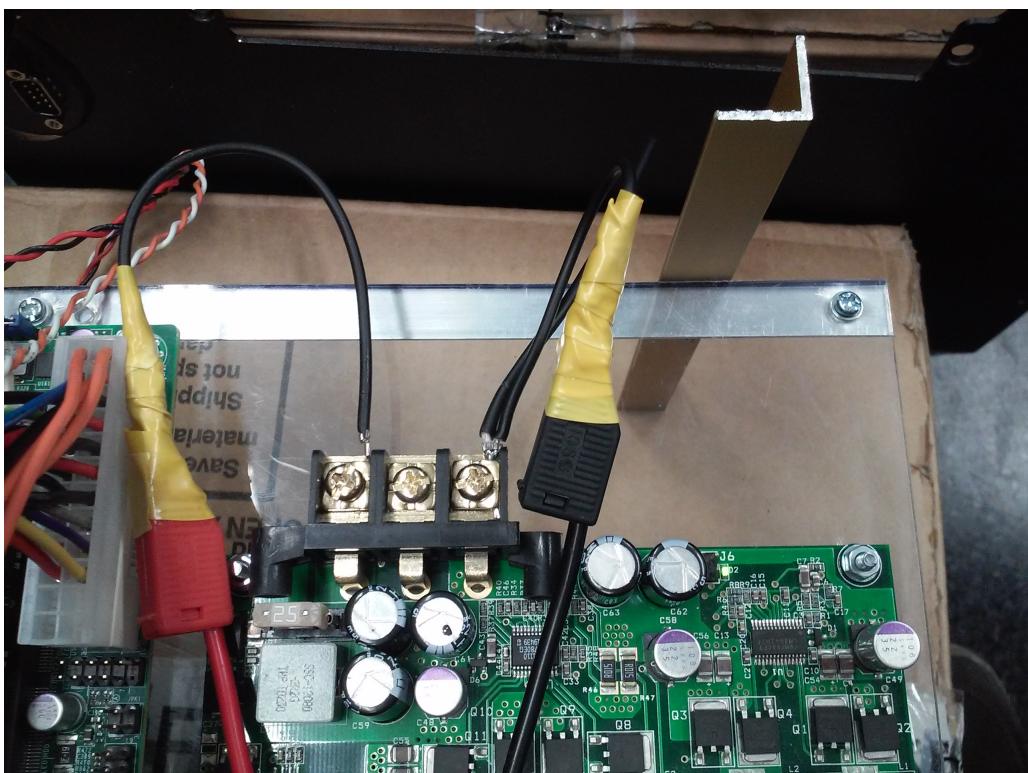


Rysunek 3: Komputer pokładowy umieszczony wewnętrz robota Jaguar

Aby płyta pleksi znajdowała się na odpowiedniej wysokości i nie kolidowała z elementami wcześniej umieszczoneymi w robocie, została przymocowana do metalowych kątowników, tworzących coś na kształt rusztowania.

Zdjęcie 4 przedstawia poprawne podłączenie zasilania przetwornicy, z której napięcie jest doprowadzane do jednostki centralnej komputera. Jest to model *M4 ATX DC DC CAR PC POWER*. Bardzo ważna jest prawidłowa polaryzacja zasilania. Na zdjęciu wyraźnie widać, że + zasilania jest z lewej strony złącza (czerwony przewód), a masa – po prawej (czarny przewód). Środkowy zacisk to tzw. zapłon (ang. ignition) - powinien pozostać niepodłączony. W przypadku jakichkolwiek wątpliwości dotyczących prawidłowego podłączenia, należy zatrzymać się na instrukcji przetwornicy, która znajduje się pod adresem
<http://resources.mini-box.com/online/PWR-M4-ATX/PWR-M4-ATX-manual.pdf>

Z tej właśnie instrukcji korzystano podczas pracy nad podłączeniem komputera.



Rysunek 4: Połączenie zasilania przetwornicy DC-DC

Kolejne dwa zdjęcia pokazują miejsce podłączenia przewodu sieciowego. Komputer pokładowy korzysta z własnej sieci robota Jaguar. Zdjęcie 5 przedstawia podłączenie przewodu od strony komputera, natomiast zdjęcie 6 – koniec przewodu wychodzący od strony robota. Ze względu na to, że wszystkie wyprowadzenia z routera zostały wykorzystane, posłużyono się przewodem, który był uprzednio wyprowadzany na zewnątrz robota (z czego korzystano podłączając się do robota z laptopa, kiedy jeszcze komputer pokładowy nie był zamontowany). Poprzednie wyprowadzenie omawianego przewodu widać na zdjęciu 7.



Rysunek 5: Podłączenie kabla internetowego od strony komputera pokładowego



Rysunek 6: Podłączenie kabla internetowego od strony robota Jaguar



Rysunek 7: Poprzednie miejsce wyprowadzenia kabla internetowego

Ostatnie dwa zdjęcia pokazują, jak należy podłączyć przyciski START oraz RESET komputera. Ich kolorystyka przedstawia się następująco

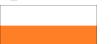
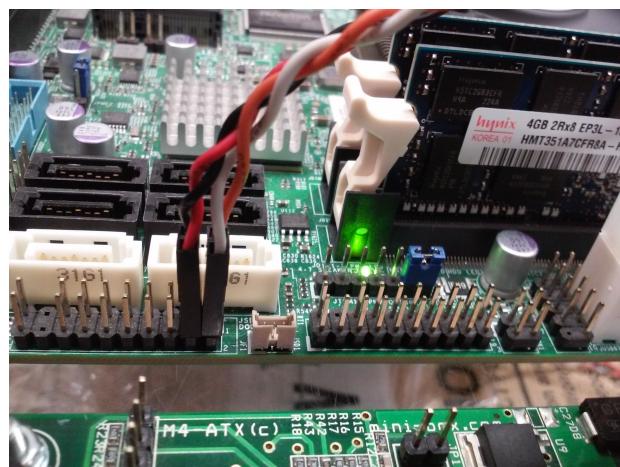
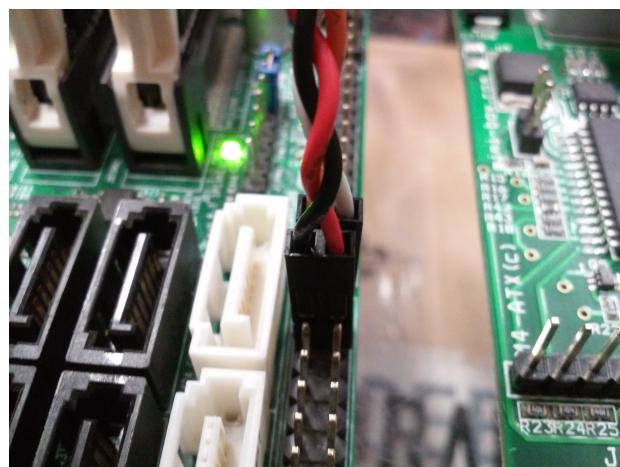
Przycisk	Kolor
START	biao – pomarańczowy 
RESET	czerwono – czarny 

Tabela 1: Oznaczenia przycisków

Ponadto, oba przewody są podpisane. Napis widnieje na czarnych wtykach. Niestety, nie jest on widoczny na zamieszczonych zdjęciach. Fizyczne podłączenie przycisków, przedstawiają rysunki 8 i 9.



Rysunek 8: Podłączenie przycisków START oraz RESET komputera pokładowego



Rysunek 9: Podłączenie przycisków START oraz RESET komputera pokładowego

5.3 Instrukcja połączenia z komputerem pokładowym

Podłączenie się do komputera pokładowego robota Jaguar nie jest trudne i nie wymaga specjalnego sprzętu. Zupełnie wystarczający jest komputer stacjonarny lub laptop z kartą WiFi. Dalsze czynności przedstawiają się następująco:

1. Po włączeniu robota należy odczekać około 2 minut. Jest to czas gwarantujący uruchomienie robota, ustabilizowanie jego sieci oraz przygotowanie komputera pokładowego do pracy.
2. Następnie należy ustawić na swoim komputerze statyczny adres IP (sieć robota nie przydziela adresów dynamicznie). Na pewno niedostępne są adresy zajęte przez robota (wymienione w jego dokumentacji) oraz adres 192.168.0.100 (adres komputera pokładowego).
3. Mając już ustawione IP należy połączyć się z siecią robota (nazwa i hasło podane w dokumentacji).
4. Po powyższych krokach możliwe jest już nawiązanie połączenia. W terminalu należy wprowadzić komendę:

```
ssh jaguar@192.168.0.100
```

Hasło: *jaguar*

Do nawiązania połączenia można wykorzystać również dowolny klient SSH.

6 Raport K3 - Opis standardu wiadomości GPS i możliwości nawigacji robota

Robot Jaguar posiada moduł pracujący i publikujący wiadomości w standardzie NMEA 0183. Komunikacja odbywa się przy użyciu protokołu RS232, aczkolwiek w robocie komunikacja ta została przeprowadzona przez TCP. W tej sytuacji, aby nasłuchała dane publikowane przez moduł, należało połączyć się z robotem (jego routerem), a następnie jak w trakcie zwykłej komunikacji szeregowej podejrzeć wysyłane dane. W trakcie testu wewnętrz budynku, niestety okazało się, że jedna z paczek dedykowanych temu typu komunikacji alarmowała o niepowodzeniu parsowania wiadomości. Dopiero wyprowadzenie robota na zewnątrz budynku poskutkowało uzupełnieniem pustych pól wiadomości. Poniżej zaprezentowany został fragment danych otrzymywanych od GPS-a robota:

```
$GPGSV,2,2,08,25,36,137,41,26,44,300,46,29,64,069,46,31,00,000,20*72  
$GPRMC,162747.6,A,5106.54375,N,01703.59396,E,000.01,122.2,290515,004.1,E*55  
$GPGGA,162747.6,5106.54375,N,01703.59396,E,1,06,1.5,231.0,M,41.8,M,,*53  
$GPGSA,A,3,05,16,20,,25,26,29,,,2.6,1.5,2.1*39  
$GPGSV,2,1,08,05,25,067,36,16,16,306,33,20,29,090,45,21,00,000,41*76  
$GPGSV,2,2,08,25,36,137,41,26,44,300,46,29,64,069,46,31,00,000,20*72
```

6.1 Format wiadomości NMEA

Powyższe komunikaty informują o następujących cechach:

- **GPGSV** - informacja o satelitach, ich ilości, położenie, moc sygnału
- **GPRMC** - informacje o poprawności pozycji, określenie położenia, wskaźnik trybu obliczeń
- **GPGSA** - informacje o obniżeniu precyzji podanej pozycji (horyzontalnie oraz wertykalnie).
- **GPGGA** - wysokość i szerokość geograficzna, informacja czy wynik jest estymowany, obniżenie precyzji pomiaru

6.2 Opis paczek GPS i nawigacji

Poniżej zostały krótko opisane paczki, które obsługują GPS lub nawigację:

- **nmea_nasat_driver** - paczka wydaje się odpowiednią do wykorzystania dla robota Jaguar, ponieważ zainstalowany moduł GPS publikuje informacje w formacie NMEA 0183. Paczkę wykorzystuje się jako interfejs, który parsuje dane wejściowe od modułu GPS i zwraca je w postaci prostych wiadomości. Może być połączona z pakietem geographic_info, który może posłużyć do obsługi nawigacji.

Uruchomiony węzeł publikuje na topikach:

`sensor_msgs/NavSatFix` - pozycja GPS
`geometry_msgs/TwistStamped` - prędkość wyznaczona przez GPS
`sensor_msgs/TimeReference` - czas z GPS-a

Pakiet udostępnia węzeł, umożliwiający odczyt danych z portu szeregowego (nmea_serial_driver).

- **navigation_stack** - paczka oferuje algorytm nawigacji bazujący na odczycie z czujnika laserowego. Sterowanie odbywa się na bazie załadowanej mapy. Informacje nt. odczytu z laseru muszą być publikowane w wiadomościach sensor_msgs/LaserScan. Dodatkowo należy publikować informacje o położeniu (odometria) przy użyciu `tf` i wiadomości `nav_msgs/Odometry`. Wynik nawigowania zostanie w takiej sytuacji wysłany przy użyciu wiadomości `geometry_msgs/Twist`. W tej sytuacji trzeba zadbać o to, aby wiadomość miała przełożenie na ruch robota. Ostatecznie mapa nie musi być stworzona i dostępna dla paczki, aczkolwiek istnieją tutoriale na stronach wiki ROS-a, które tłumaczą w jaki sposób zaimplementować obsługę wcześniej wygenerowanej mapy.
- **robot_localization** - Paczka może się przydać do wyznaczania bieżącej lokalizacji robota. Daje możliwość połączenia odczytu z wielu sensorów i na ich podstawie wyestymowania położenia i orientacji, w której znajduje się robot. Zaletą paczki jest to, że współpracuje z wieloma typami wiadomości i sensorów i można do niej podłączyć nieograniczoną liczbę czujników. W przypadku robota Jaguara ocenę położenia można uzyskać przy pomocy modułu GPS, kamery, modułu IMU przy użyciu tej paczki.