Imię Nazwisko: Konrad Pleban
Pozycja w projekcie: Specjalista ds. obsługi technicznej

1 Założenia, literatura oraz narzędzia

1.1 Opis prac potrzebnych do wykonania projektu

Do obowiązków specjalisty ds. obsługi technicznej należało stworzenie sterowania głową robota społecznego *Samuel*, oraz podłączenie jej do platformy mobilnej *PowerBot*.

1.2 Literatura użyta przy wykonaniu projektu

Przy tworzeniu części projektu opisywanej w tym dokumencie wykorzystano następujące pomoce naukowe:

- Książka "Nauka robotyki z językiem Python"[1]
- Tutoriale ROS przeznaczone do sterowania aktuatorami *Dynamixel* [2]
- Dokumentacja aktuatora *Dynamixel* $RX-28$ [3]
- Dokumentacja konwentera $AVT-530$ [4]

1.3 Narzędzia użyte przy wykonaniu projektu

Sprzęt użyty do wykonania zadań to przetwornica Step-Down, służąca do obniżenia napięcia z 24V dostępnego w platformie mobilnej, do 18V potrzebnych do zasilenia głowy robota społecznego *Samuel*. Wykorzystane zostały również z części *Samuela* takie jak serwomechanizmy *Dynamixel* RX-28 oraz konwenter $AVT-530$. Dodatkowo przy testach aktuatorów został użyty zasilacz 12V.

[1] Wydawnictwo Packt Publishing 2015, polski przekład Helion 2016
[2] http://wiki.ros.org/dynamixel_controllers/Tutorials
[3] http://www.crustcrawler.com/motors/RX28/docs/RX28_Manual.pdf
[4] http://serwis.avt.pl/manuals/AVT530.pdf

2 Określenie prac nad elementami projektu

Wedle zasady dziel i rządź, ten element projektu został podzielony na kilka mniejszych.

2.1 Główny element projektu - Głowa Samuel

Samuel ma służyć do wykrywania, wyszukiwania i określania pozycji w przestrzeni kodów QR dzięki wbudowanej kamerze w głowie.

2.2 Sterowanie aktuatorami Dynamixel

Wykorzystane w robocie serwomechanimzy służą do poruszania robotyczną głową w przestrzeni RPY. W celu ułatwienia pracy każdy dozwolony ruch został nazwany od gestów jakie się wykonuje poruszając głową. W ten sposób obrót Yaw uzyskał nazwę - *nie*, kąt Pitch - *tak*, a kąt Roll jako, że wychodzi od bazy robota - *baza*.

2.3 Odczytywanie obrazu z kamery

Problem ten został oddelegowany do członka zespołu odpowiedzialnego za wykrywanie kodów QR.

2.4 Połączenie Samuela z PowerBotem

Do zasilenia *Samuela* wkorzystano przetwornicę Step-Down, do komunikacji natomiast użyto konwertera $AVT-530$ połączonego z portem COM3 komputera pokładowego *PowerBota*.

3 Specyfikacja rozwiązania

3.1 Nauka sterowania aktuatorami Dynamixel

3.1.1 Pozyskanie elementów

Z racji, że żaden z członków zespołu nie sterował wcześniej tego typu serwomechanizmami, pracę trzeba było zacząć od nauki ich sterowania. W tym celu z głowy robota *Samuel* zostały odłączone dwa serwomechanizmy RX-28 o numerach 8 oraz 9, które służą do wcześniej opisanego ruchu *tak*, oraz konwenter $AVT-530$.

3.1.2 Zasilenie aktuatorów

Po otrzymaniu serwomechnizmów została sprawdzona dokumentacja techniczna wymieniona w literaturze. Ważnymi elementami, które należy sprawdzić jest napięcie pracy, maksymalny prąd pobierany przez aktuator oraz sposób podłączenia. Do zasilania przy identyfikacji został użyty zasilacz sieciowy 12V o maksymalnym prądzie 2,5A. Sposób zasilania zaprezentowano na rysunku 1

3.1.3 Podłączenie serwomechanizmów

Jak wcześniej zostało napisane w dokumentacji, należy sprawdzić sposób połączenia aktuatorów z konwenterem oraz ze sobą nawzajem. Pierwszy aktuator połączono z konwenterem zgodnie z podaną specyfikacją, następnie do pierwszego motoru podłączono szeregowo kolejny skrosowanym przewodem. Sposób podłączenia został przedstawiony na rysunku 2

3.1.4 Paczka do obsługi aktuatorów

Zanim zacznie pisać się skrypt najpierw należy stworzyć paczkę z pakietami *dynamixel_controllers*, *std_msgs*, *rospy* oraz *roscpp*. Oczywiście pakiety należy wcześniej pobrać z repozytorium. Część z nich została zainstalowana podczas instalacji systemu ROS. Z dużym prawdopodobieństwem, jeżeli na komputerze nikt nie sterował serwomechanizmami nie ma zainstalowanego pakietu *dynamixel_controllers*, w celu jego pobrania w wersji ROS *Hydro* należy wpisać następujące polecenie:

```
apt-get install ros-hydro-dynamixel-controllers
```

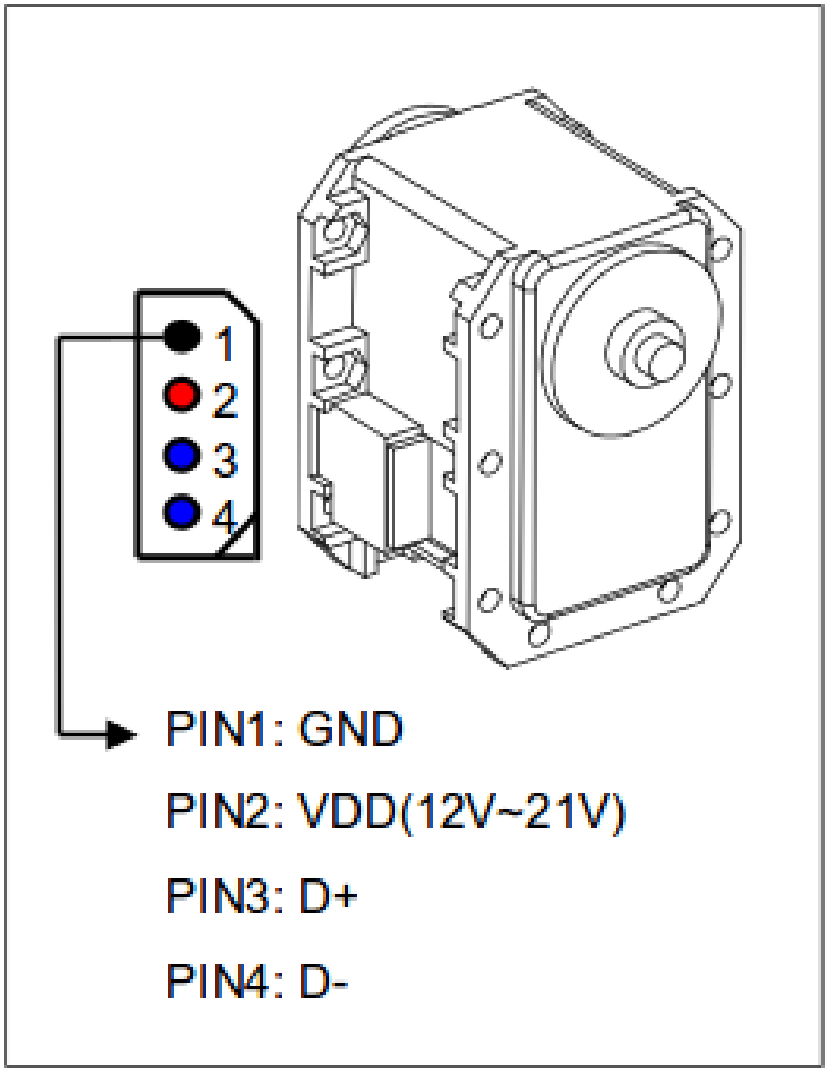
W celu zainstalowania pakietu pod inną wersją systemu ROS element *hydro*, należy zastąpić jej nazwą. Jeżeli ktoś w przyszłości chciałby zrobić coś więcej z aktuatorami dynamixel, może zainstalować wszystkie swoje pakiety. Przykładowo w wersji *Jade*, polecenie wyglądałoby następująco:

```
apt-get install ros-jade-dynamixe*
```

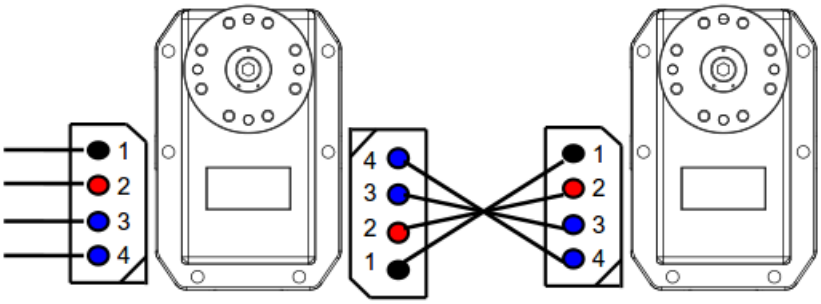
Zasady instalacji pod innymi wersjami są takie, jak we wcześniejszym przypadku.

Po zainstalowaniu niezbędnych pakietów tworzenie paczki we wszystkich wersjach ROS'a wygląda następująco:

```
cd /home/catkin_ws/src
catkin_create_pkg dynamixel_con dynamixel_controllers std_msgs rospy roscpp
```



Rysunek 1: Zasilanie serwomechanizmu



Rysunek 2: Połączenie ze sobą dwóch aktuatorów

co oznacza, że w folderze /home/catkin_ws/src zostanie utworzona paczka o nazwie dynamixel_con zawierająca pakiety *dynamixel_controllers*, *std_msgs*, *rospy* oraz *roscpp*. Oczywiście nazwę paczki można zmienić w zależności od potrzeb i upodobań.

3.1.5 Odczytywanie pozycji serwomechanizmów

Zanim zacznie się tworzyć skrypty dobrą praktyką jest uporządkowanie środowiska pracy. Podczas pracy przy projekcie zostały więc stworzone dwa foldery w katalogu /home/catkin_ws/src/dynamixel_con poleceniami:

```
mkdir launch
mkdir config
```

W pierwszym folderze znajdować się będą pliki z rozszerzeniem .launch, w kolejnym z rozszerzeniem .yaml. W tym podrozdziale należy przejść do folderu launch oraz utworzyć w nim plik controller_manager.launch. Nazwę, tak jak wcześniej, można zmienić wedle upodobań, należy jednak pamiętać o tej zmianie podczas pisania lub kopiowania skryptów. Należy również nadać uprawnienia do wykonywania pliku przy użyciu komendy chmod.

```
cd ~/catkin_ws/src/dynamixel_con/launch
touch controller_manager.launch
chmod +x controller_manager.launch
```

Następnie należy umieścić w pliku następujący kod:

```
<!-- -*- mode: XML -*- -->

<launch>
    <node name="dynamixel_manager" pkg="dynamixel_controllers"
    type="controller_manager.py" required="true" output="screen">
        <rosparam>
            namespace: dxl_manager
            serial_ports:
                dxl_tty0:
                    port_name: "/dev/ttyS0"
                    baud_rate: 57142
                    min_motor_id: 1
                    max_motor_id: 25
                    update_rate: 20
        </rosparam>
    </node>
</launch>
```

4. Linia kodu tworzy node o nazwie dynamixel_manager z pakietu dynamixel_controllers, określa jej typ i inne potrzebne rzeczy, o których można przeczytać w literaturze. Ważną rzeczą na którą trzeba zwrócić uwagę jest zmienna *port_name* - należy wpisać tutaj nazwę portu szeregowego do którego jest podłączony konwerter. Warto zwrócić uwagę, że do prawidłowego działania skryptu potrzebne są uprawnienia do zapisu udzielane przy pomocy komendy chmod.

```
chmod +w /dev/ttyS0
```

Wyżej znajduje się linia

```
dxl_tty0:
```

określa ona nazwę pod jaką zostaną zapisane dane aktuatorów z portu. Niżej w kodzie znajduje się zmienna baud_rate, wartość której można znaleźć w notach katalogowych aktuatorów.

Po zapisaniu pliku należy wywołać następujące polecenia:

```
catkin_make
roslaunch dynamixel_con controller_manager.launch
```

W terminalu powinny wyświetlić się komunikaty o znalezionych serwomechanizmach oraz o utworzonych nodach.

W nowym terminalu należy wywołać następujące polecenie:

```
rostopic list
```

Po wywołaniu tego polecenia powinna wyświetlić się lista aktywnych tematów. Wśród nich powinien znajdować się temat

```
\dxl_manager\dxl_tty0
```

Gdy włączy się nasłuchiwanie poleceniem:

```
rostopic echo \dxl_manager\dxl_tty0
```

Powinny pojawiać się stany na podłączonych serwomechanizmach oraz ich ID; przykładowo:

```
timestamp: 125345.28
 id: 1
 goal: 256
```

```
    position: 256
    error: 0
    speed: 0
    load: 0.0
    voltage: 12.2
    temperature: 41
    moving: False
  -
    timestamp: 125345.28
    id: 8
    goal: 678
    position: 650
    error: -28
    speed: 0
    load: 0.0
    voltage: 12.3
    temperature: 40
    moving: False
```

3.1.6 Zadanie momentu oraz pozycji

W tym rozdziale należy przejść do wcześniej utworzonego folderu config oraz utworzyć w nim plik z rozszerzeniem .yaml

```
cd ~\catkin_ws\src\dynamixel_con\config
touch dynamixel_joint_controllers.yaml
```

Dla tego pliku nie potrzeba nadawać dodatkowych uprawnień. Należy w nim umieścić następujący kod:

```
tak_controller:
    controller:
        package: dynamixel_controllers
        module: joint_position_controller
        type: JointPositionController
    joint_name: tak_joint
    joint_speed: 2.0
    motor:
        id: 1
        init: 0
        min: 0
        max: 1023
```

```
tak2_controller:
    controller:
        package: dynamixel_controllers
        module: joint_position_controller
        type: JointPositionController
    joint_name: tak2_joint
    joint_speed: 1.5
    motor:
        id: 8
        init: 0
        min: 0
        max: 1023
```

tak_controller, tak2_controller, tak_joint, tak2_joint, to nazwy utworzone, które można zmienić wedle potrzeb, należy jednak robić to konsekwentnie. zmienna joint_speed przyjmuje wartość z jaką ma poruszać się serwomechanizm, zmienna id oznacza odczytane we wcześniejszym rozdziale numery aktuatorów.

Następnie należy powrócić do katalogu launch i utworzyć w nim kolejny plik.

```
cd ~\catkin_ws\src\dynamixel_con\launch
touch controller_spawner.launch
chmod +x controller_spawner.launch
```

Jak we wcześniejszych przypadkach, w pliku należy umieścić następujący kod:

```
<!-- -*- mode: XML -*- -->

<launch>
    <rosparam file="$(find dynamixel_con)/config
    /dynamixel_joint_controllers.yaml" command="load"/>

    <node name="dynamixel_controller_spawner" pkg="dynamixel_controllers"
    type="controller_spawner.py"
          args="--manager=dxl_manager
                --port=dxl_tty0
                --type=simple
                tak_controller
                tak2_controller"
```

```
          output="screen"/>
</launch>
```

Ważne rzeczy, o których trzeba pamiętać to:

w 4. linii kodu ustalić ścieżkę do pliku .yaml

w 8. linii kodu ustalić nazwę portu taką samą jak w pliku controller_manager.launch,

w 10. linii wpisać nazwy aktuatorów.

Po takiej konfiguracji plików należy wywołać następujące polecenia. Jeżeli controller_manager jest uruchomiony, pierwsze można pominąć.

```
roslaunch dynamixel_con controller_manager.launch
roslaunch dynamixel_con controller_spawner.launch
```

Gdzie drugie polecenie należy wywołać w osobnym terminalu po zakończeniu uruchamiania pierwszego. Po uruchomieniu drugiego polecenia aktuatory zostaną zablokowane w aktualnej pozycji. W celu poruszenia serwomechanizmem należy opublikować wiadomość do topica komendą:

```
rostopic pub -1 /tak_controller/command std_msgs/Float64 -- 3.1415
```

Co spowoduje przesunięcie serwomechanizmu id 1. na pozycje 3,1415 czyli 180* względem pozycji 0, analogicznie steruje się drugim serwomechanizmem. Do sterowania aktuatorami są dostępne również usługi, w celu wyświetlenia ich listy należy wywołać polecenie:

```
rosservice list
```

W oknie terminala powinny wyświetlić się wszystkie dostępne usługi.

```
/restart_controller/dxl_tty0
/start_controller/dxl_tty0
/stop_controller/dxl_tty0
/tak_controller/set_compliance_margin
/tak_controller/set_compliance_punch
/tak_controller/set_compliance_slope
/tak_controller/set_speed
/tak_controller/set_torque_limit
/tak_controller/torque_enable
```

Takie same usługi są dostępne dla drugiego serwomechanizmu. Co kontrolują poszczególne usługi poleca się sprawdzić w literaturze. Warto podkreślić ostatnią dzięki której można wyłączyć blokowanie na aktuatorze.

```
rosservice call /tak/controller/torque_enable "torque_enable: false"
```

W celu zadania momentu, należy zadać mu pozycję.

3.2 Praca z głową Samuel

Po testach na "gołych" aktuatorach, głowa robota społecznego *Samuel* została złożona z powrotem w całość. Podłączono konwerter AVT do portu COM komputera, uruchomiono skrypt controller_manager.launch, odczytano nowe ID serwomechanizmów, zidentyfikowano za jakie ruchy odpowiadają i odpowiednio zmodyfikowano plik .yaml, oraz plik controller_spawner.launch. Końcowe pliki są dostępne w systemie kontroli wersji GIT [5].

Należy zwrócić uwagę, że serwomechanizm id 1.(w robocie 9.) oraz id 8. pracują w sprzężeniu i należy nimi sterować równolegle, aby uniknąć awarii.

3.3 Zasilenie głowy Samuel z PowerBota

Kolejnym etapem było odłączenie zasilacza sieciowego w głowie robota społecznego *Samuel*, zamontowanie przetwornicy Step_Down, wcześniej ustawionej na obniżenie napięcia z 24V na 18V i podłączenie jej wyjścia w miejsce, gdzie był podłączony zasilacz sieciowy, a wejście wyprowadzono z wyjścia 24V PowerBota i masy platformy. Tak zasilanego *Samuela* położono na platformie mobilnej i podłączono do jej komputera pokładowego.

[5] https://github.com/PowerBotSamuel/PowerSammy/