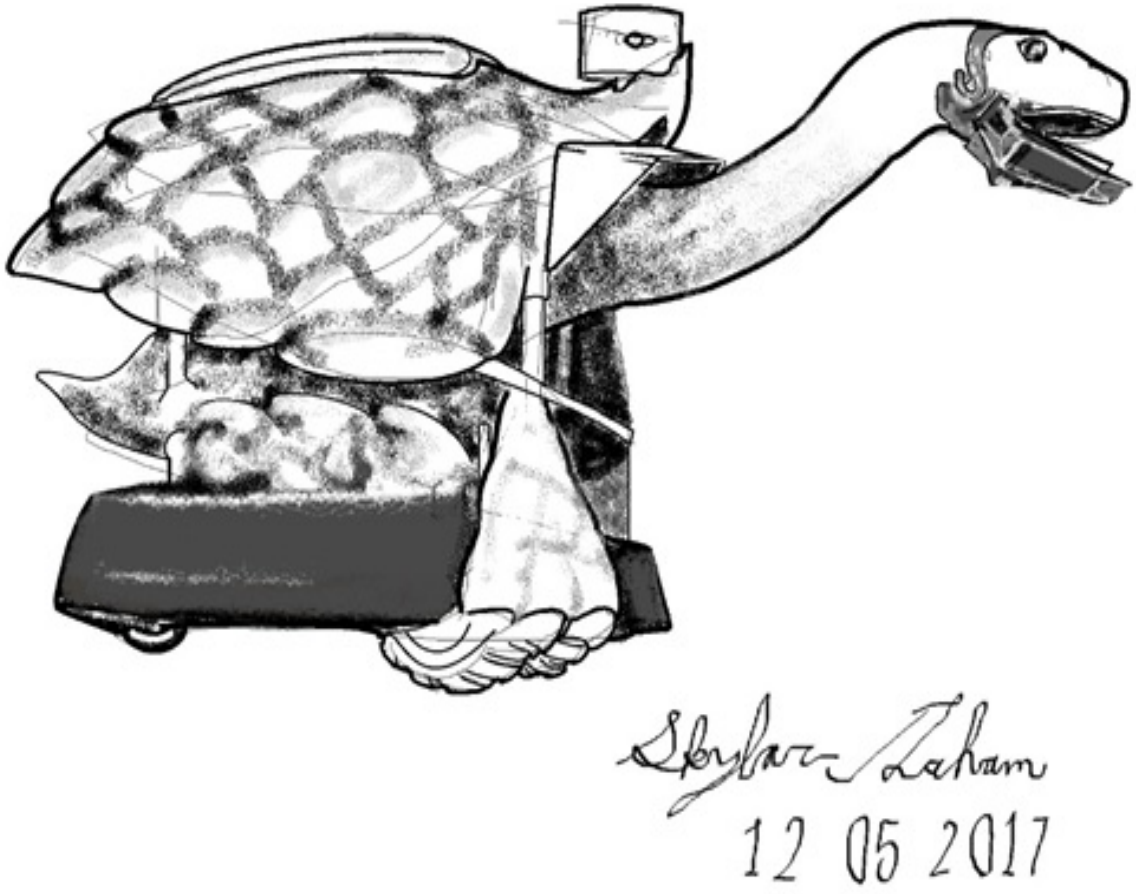
Khoi PHAM NHAT / Skylar J LAHAM / Olivier S AGBOHOUI

Robotic Engineering 2 - Technical Report

Turtlebot Implementation



From Turtle-bot to Tortoise-bot

Contents

1 Introduction

Robotics is a highly advanced and highly interactive field of engineering. Robotics has place in much of society, including agricultural, manufacturing, industry, aerospace, and military. Beyond the development of exotic robotic applications, robotics is experiencing major development in commonplace applications, such as leisurely activities such as drone piloting, economical activities such as self driving cars, and communication activities such as a fleet of mail delivering robots.

In many cases, robotic code is available for little to no monetary cost, and with entry level robotic kits with micro and nano computer boards available for approximately 100 euros, robotics has become accessible to many people. An advantage of robotic engineering is diversity; robotic engineering is able to help scientists navigate robotic submarines, and vacuums to navigate a home and clean the floor. Therefore, robotics may be thought of as a tool to aid society in many activities.

1.1 Reasons of study

As stated, Robotics is a blossoming field of engineering. As many societies of the 21st century are attracted towards smart devices, and integrated interfacing and applications, robotics becomes more dominant as a component for daily use. Whether the purpose is for passion, or job outlook, the students of the 2016 to 2017 class of Computer Vision and Robotics have chosen the highly technical and innovative field as the starting point. The computer vision and robotics program itself is developing a unique an identity each semester, as with the arrival of new students means the arrival of new components and projects for students to gain hands on work experience. Earlier semesters implemented LiDAR and Infrared sensors on a turtlebot; for the class of 2016 to 2017, the task had been to build and troubleshoot a robotic arm. Upon successful robot arm functionalities, the students have been tasked to incorporate the robot arm onto the turtlebot. While there is guidance from the professors, the students are allowed free range of arm mounting designs, etc.

The incorporation is beyond hardware systems, as the student teams are to tasked build a field of terrain for the turtlebot to navigate, and, if able, implement the robotic arm with the turtlebot in a mission scenario. As stated, the students have been allowed to develop their own mission plans and strategies. Such processes are beneficial, as the independence of the course is similar to what may be experienced in the robotic workplace.

2 System Overview

2.1 Turtlebot and PC configuration

- The turtlebot is built on a semi-autonomous vacuum base. The base is Kobuki; for the purposes of advanced robotic functionalities and simplicity of coding to middle-ware, our team of student investigators has selected to implement the robot with "Turtlebot II" commands, rather than Kobuki.
- Shelves are mounted on the vacuum base. The shelves are able to accommodate several installation positions for complementary gadgetry such as, LiDAR, Infrared sensors, network router, netbooks and board chip-sets (as arranged on the LE2i turtlebots).
- The routers allow a comparatively smaller netbook to connect with a comparatively larger work station via SSH. The routers are typically for LAN connections, whereas the standard netbook wifi is option and less than necessary.

2.2 Turtlebot Specifications

2.2.1 Included components

- Kobuki Base.
- 1x4S1P battery 2200 mAh.
- Battery charger.
- USB communication cable.

2.2.2 Functional specification

- Maximum translational velocity: 70 cm/s
- Maximum rotational velocity: 180 deg/s (>110 deg/s gyro performance will degrade)
- Payload: 5 kg (hard floor), 4 kg (carpet)
- Cliff: will not drive off a cliff with a depth greater than 5cm
- Threshold Climbing: climbs thresholds of 12 mm or lower
- Rug Climbing: climbs rugs of 12 mm or lower
- Expected Operating Time: 3/7 hours (small/large battery)
- Expected Charging Time: 1.5/2.6 hours (small/large battery)
- Docking: within a 2mx5m area in front of the docking station

2.2.3 Hardware specification

- PC Connection: USB or via RX/TX pins on the parallel port
- Motor Overload Detection: disables power on detecting high current (>3A)
- Odometry: 52 ticks/enc rev, 2578.33 ticks/wheel rev, 11.7 ticks/mm
- Gyro: factory calibrated, 1 axis (110 deg/s)
- Bumpers: left, center, right
- Cliff sensors: left, center, right
- Wheel drop sensor: left, right
- Power connectors: 5V/1A, 12V/1.5A, 12V/5A
- Expansion pins: 3.3V/1A, 5V/1A, 4 x analog in, 4 x digital in, 4 x digital out
- Audio : several programmable beep sequences
- Programmable LED: 2 x two-coloured LED
- State LED: 1 x two coloured LED [Green - high, Orange - low, Green Blinking - charging]
- Buttons: 3 x touch buttons
- Battery: Lithium-Ion, 14.8V, 2200 mAh (4S1P - small), 4400 mAh (4S2P - large)
- Firmware upgradeable: via usb
- Sensor Data Rate: 50Hz
- Recharging Adapter: Input: 100-240V AC, 50/60Hz, 1.5A max; Output: 19V DC, 3.16A
- Netbook recharging connector (only enabled when robot is recharging): 19V/2.1A DC
- Docking IR Receiver: left, centre, right
- Diameter : 351.5mm / Height : 124.8mm / Weight : 2.35kg (4S1P - small)

2.2.4 Software specification

- Arduino
- Dynamixel
- ROS
- RVIZ Visualisation: RVIZ is an important feature for ROS capabilities, including: navigation, gmapping, data monitoring.
- RQT plot: : RQT is an important graphical feature that shows the nodes and topics of the Workstation-ROS-Robot.

2.3 Components

2.3.1 Sensors and measurement devices

- Built into robotic base:
  - Two * Servo odometer (distance)
  - Gyro (axis and motion)
  - Three * binary bumpers (touch)
  - Three * Cliff sensors (terrain-floor elevation)
  - Three * Infrared sensors (docking)
- Externally mounted:
  - LiDAR (two to three dimensional mapping, depending on application and mounting)
  - Infrared camera (Kinect Infrared sensor for comparatively higher density three dimensional mapping)

2.3.2 Other components

- Installed before 2016:
  - Netbook
  - Router
- Installed by the 2016 to 2017 student researchers:
  - Robotic arm (The robotic arm allows for elevated capabilities complexities in terms of missions and scenarios, with the robot and for students).

3 Turtlebot operational stages

3.1 Utile operations

3.1.1 File transfer

```
rsync -rPz [src_path] [remote_address]:[destin_path]

ex: rsync -rPz /ros/indigo/catkin_ws/src/rbx1 turtlebot@192.168.0.100:/ros/indigo/catkin_ws/src/rbx1
```

3.1.2 ssh

```
  ssh [remote_address]
```

3.1.3 Granting Permission

```
sudo chmod a+rw /dev/ttyUSB0
```

3.1.4 Downloads / Updates

```
sudo apt-get update
sudo apt-get install [software_name]
```

3.1.5 Package installation

```
catkin_make
```

3.1.6 Teleop device Configuration

Joystick configuration

```
ref http://wiki.ros.org/joy/Tutorials/ConfiguringALinuxJoystick

$ sudo apt-get install ros-indigo-joy                  (download joystick package)
$ ls /dev/input/                                       (list all the input)
$ sudo jstest /dev/input/jsX                           (test joystick)
$ ls -l /dev/input/jsX                                 (check permission)
$ sudo chmod a+rw /dev/input/jsX                       (grant permission)
$ roslaunch turtlebot_teleop logitech.launch --screen  (launch joystick config)
```

Or use keyboard launch file

```
$ roslaunch turtlebot_teleop keyboard_teleop.launch     (launch keyboard config)
```

3.2 Turtlbot Navigation

3.2.1 G mapping

- Download: rplidar package (https://github.com/roboticslab-fr/rplidar-turtlebot2)
- ros:

```
  $ launch rplidar config file (turtle bot):
  $ roslaunch turtlebot_le2i remap_rplidar_minimal.launch
```

- launch gmapping config file (workstation):

```
$ roslaunch rbx1_nav gmapping_demo.launch   (
$ rosrun rviz rviz -d `rospack find rbx1_nav`/gmapping.rviz
```

- joystick: Use the joystick to navigate the turtlebot around to map the edge of the room.

```
$ roslaunch rbx1_nav joystick_teleop.launch
```

- record data to bag file (workstation):

```
$ roscd rbx1_nav/bag_files
$ rosbag record -O my_scan_data /scan /tf
```

- save the map (workstation):

```
$ roscd rbx1_nav/maps
$ rosrun map_server map_saver -f my_map
```

- view the map (workstation):

```
$ roscd rbx1_nav/maps
$ eog my_map.pgm
```

3.3 Navigation

- launch config file (turtle bot)

```
$ roslaunch turtlebot_le2i remap_rplidar_minimal.launch
```

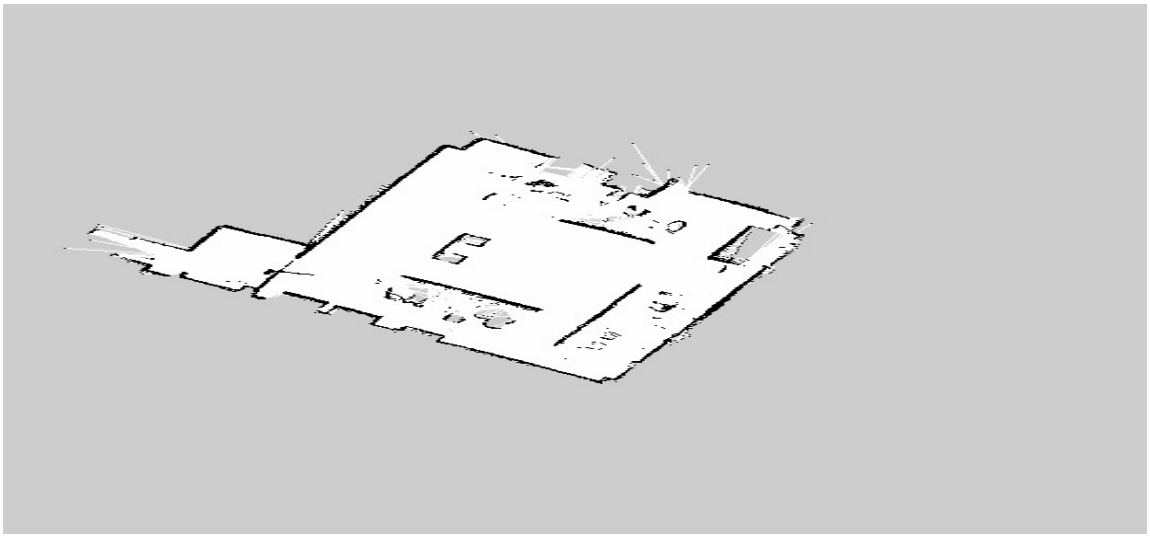
- launch amcl with map file (workstation)

```
$ roslaunch rbx1_nav tb_demo_amcl.launch map:=my_map.yaml
```

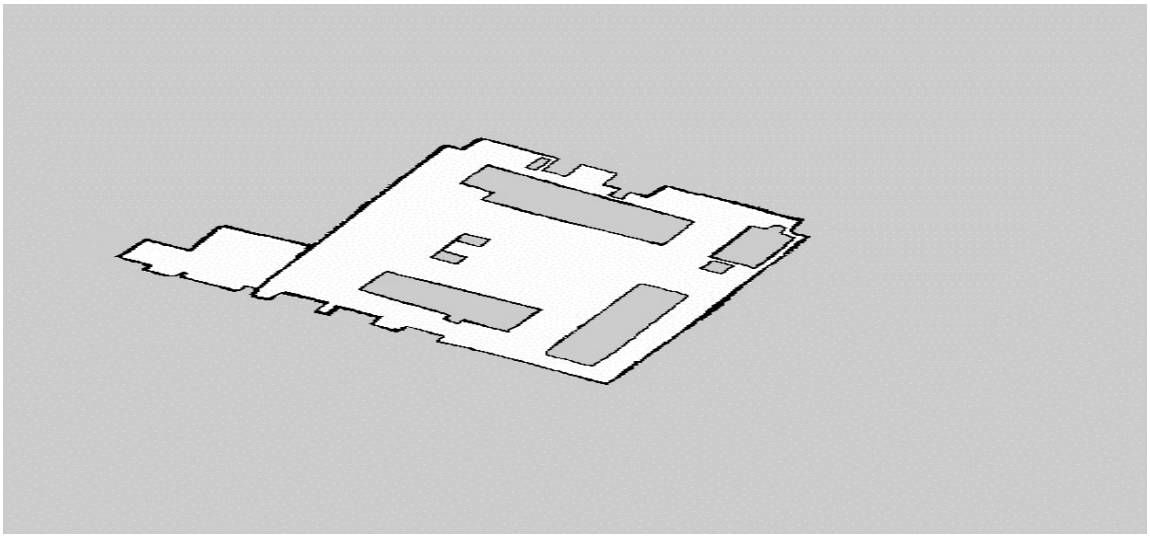
- open rviz with the saved map

```
$ rosrun rviz rviz -d `rospack find rbx1_nav`/nav_test.rviz
```

3.3.1 Map processing

- 

  Raw Map

- 

  Processed Map

3.4 Environment perception and Lidar

RPLIDAR is a low cost LIDAR sensor suitable for indoor robotic SLAM application. It provides 360 degree scan field, 5.5hz/10hz rotating frequency with guaranteed 8 meter ranger distance. By means of the high speed image processing engine designed by RoboPeak, the whole cost are reduced greatly, RPLIDAR is the ideal sensor in cost sensitive areas like robots consumer and hardware hobbyist.

3.5 Motion control

- Twist: The ROS Twist message type is for publishing motion commands, often called "cmd_vel" (command velocities). For motor rotation, the node subscribes to the cmd_vel topic and translates "twist" messages into the motor signals.
- Move base: The "move_base" is for robot movement within a specific navigation goal via mapping and odometry.
- Odometry: Odometry is a tool to measure some navigational parameters of a robot. For example: calculate the rotation of the wheel servos. Multiply the of rotation number by the circumference of the rotating robot wheel. * Robot Velocity: With the Odometry example, the speed of rotation multiplied by the circumference of the rotating wheel estimates the robot velocity.

3.6 Robotic Arm

3.6.1 Hardware Assembling

The robotic arm was built during the first half of the robotics program.

Key build notes:Key build notes:

- Thoroughly examine the instructions. By checking the manual and online sources for information of building, testing and implementing the arm, our team of student investigators knew to sand the gripper mechanism before operating the arm (the only team to do so). If the project were with an extremely high capability robotic component, abiding to the advice may save thousands of dollars and hours of labor intensive work.
- Ensure that pieces are kept in a safe place.
- Label the servos with the sticker number identifiers.
- Clean a few millimeters of material from the gripper hinge for smother functionality and superior longevity of the servo.

3.6.2 Hardware, Software, and Platforms

PhantomX Pincher Arm Software Implementation 1:

PhantomX Pincher Arm Software Implementation 1

In this image, the robotic arm has been released from the base so that the Arbotix board may be accessed without damaging components.

3.6.3 Arm Activation

ROS is a middle-ware, which is a system that is between the operating system and software. The middle-ware functionality is particularly well suitable for Ubuntu. With the assistance of the ROS tutorial guide, our team has been able to incorporate ROS with the Pincher Arm, which can be observed in the following photos and references (See source section: ROS Source Reference). Follow Gaitech turtlebot instruction:

ROS Initialization:

To allow the board to interact with the ROS technology, ROS library is loaded using Arduino as a launching tool. Upon ROS interaction, Arduino can be closed.

- Study the guide's directions. Install the download files in the Sketchbook folder.
- Access the sketchbook menu within Arduino and open the ROS implementation code and load it into the board. close Arduino.

Turtlebot ROS package activation:

Summary of the steps we have done through the Gaitech instruction:

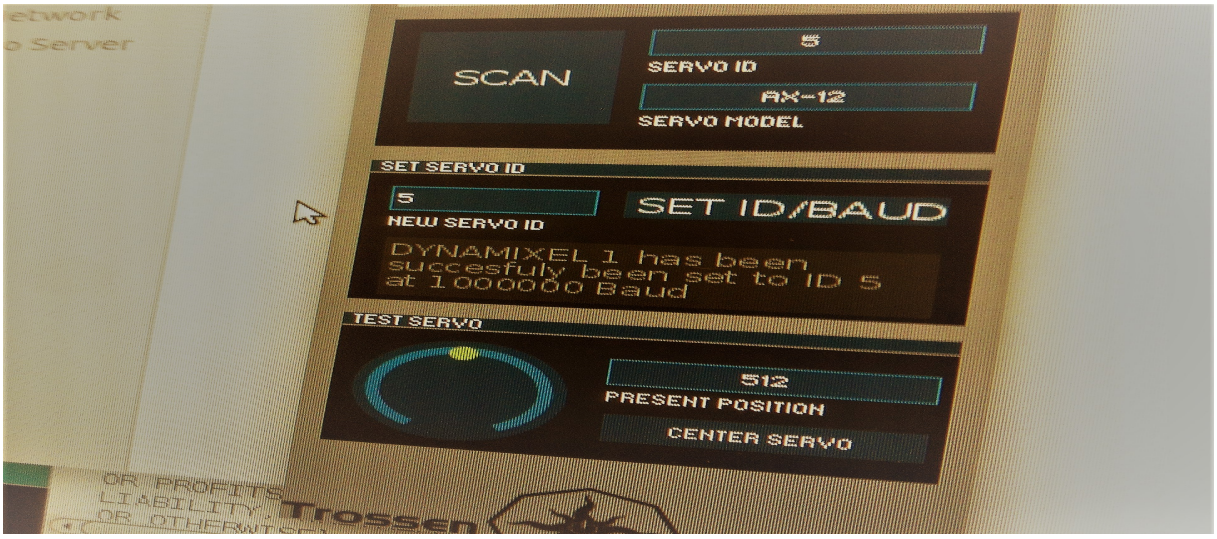
- install file to download:
  - java (for dynamixel)
  - dynamixel
  - arduino 1.0.6,
- package to download:
  - turtlebot_arm (https://github.com/turtlebot/turtlebot_arm)
  - arbotix_ros (https://github.com/vanadiumlabs/arbotix_ros)
  - sketchbook (https://github.com/trossenrobotics/arbotix/archive/master.zip)

- implementation step:
  - install java
  - install arduino
  - replace default sketchbook
  - plug the arm on computer
  - check on Arduino (Board: Arbotix, Serial port: usb0, Programmer: AVRISP MKII)
  - load ros from sketchbook on the board
  - use dynamixel set id for each servo
- ros:
  - Download package
  - Grant permission for ttyUSB0
  - Open arbotix terminal
  - List servo ids

Servo Registration

To recognize some servo motors, connect the servos via DynaManager.

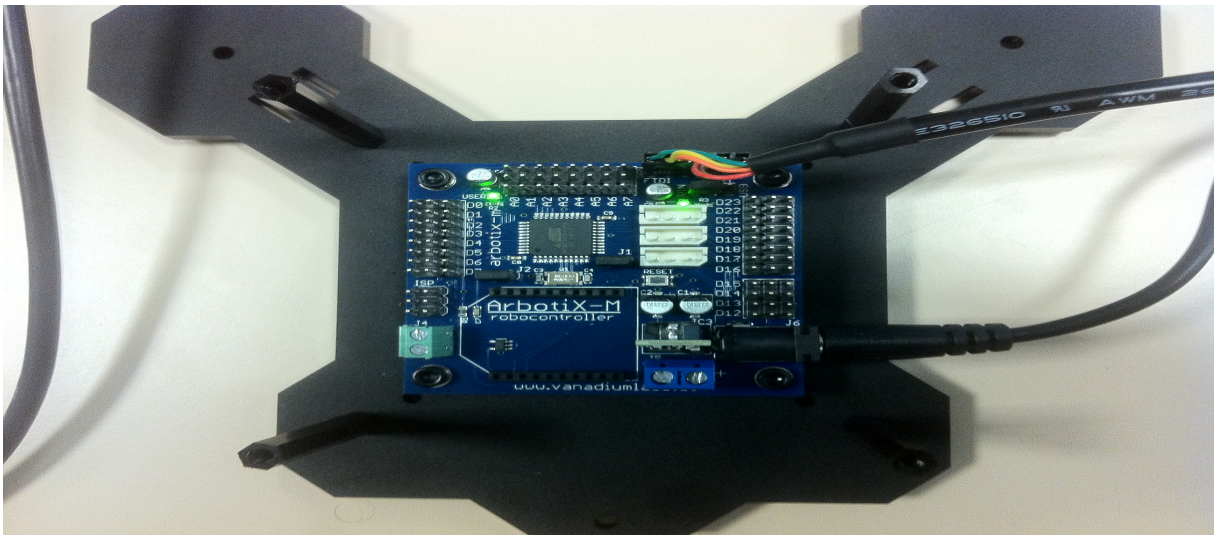
- Servo Registration.
- USB Check and Permissions
- To initiate DynaManager, download and run DynaManager.
- DynaManager accesses individual motors. Ensure that the cables are wired to an individual servo.
- Scan and register the servos one by one.
- Center all the servos using the rotary software tool.



PhantomX Pincher Arm Software Implementation 3 Board Connection

Testing

Arduino Board initial green light blinking testing (The blink test helps to make sure the board is working well. The blink is a function accessible via Arduino software, which indicates that the board is properly connected with the work station computer).



PhantomX Pincher Arm Software Implementation 2 Board Connection

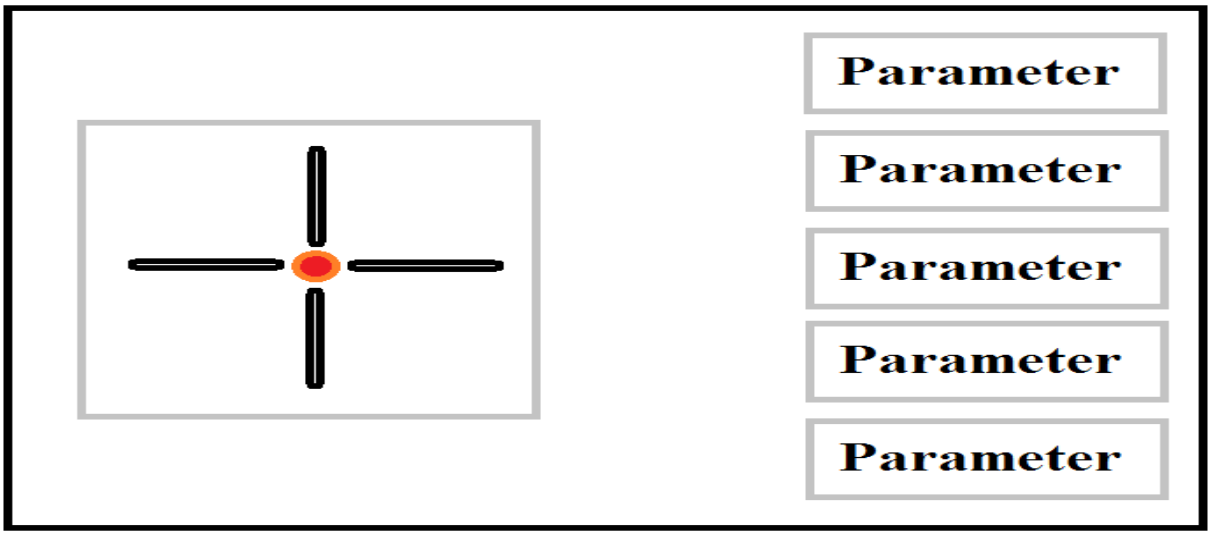
- Grant ttyUSB0 permissions
- To recognize the Arbotix board, ensure that the proper downloaded library have been placed in the sketchbook folder.
- Load the blink function with the Arduino sketchbook. A LED blinks if the actions are correct.

```
lsusb
ls /dev | gprep ttyU
ls /dev / ttyUSB0
ls -l /dev / ttyUSB0
Sudo chmod a + rw /dev / input / ttyUSB0
arbotix_terminal
ls
```

copy launch file on gaitech and modify the "yaml"

- Launch servo config file
- Open servo controller

```
$ roslaunch turtlebot_arm_bringup turtlebot_arm_bringup.launch
$ arbotix_gui
```



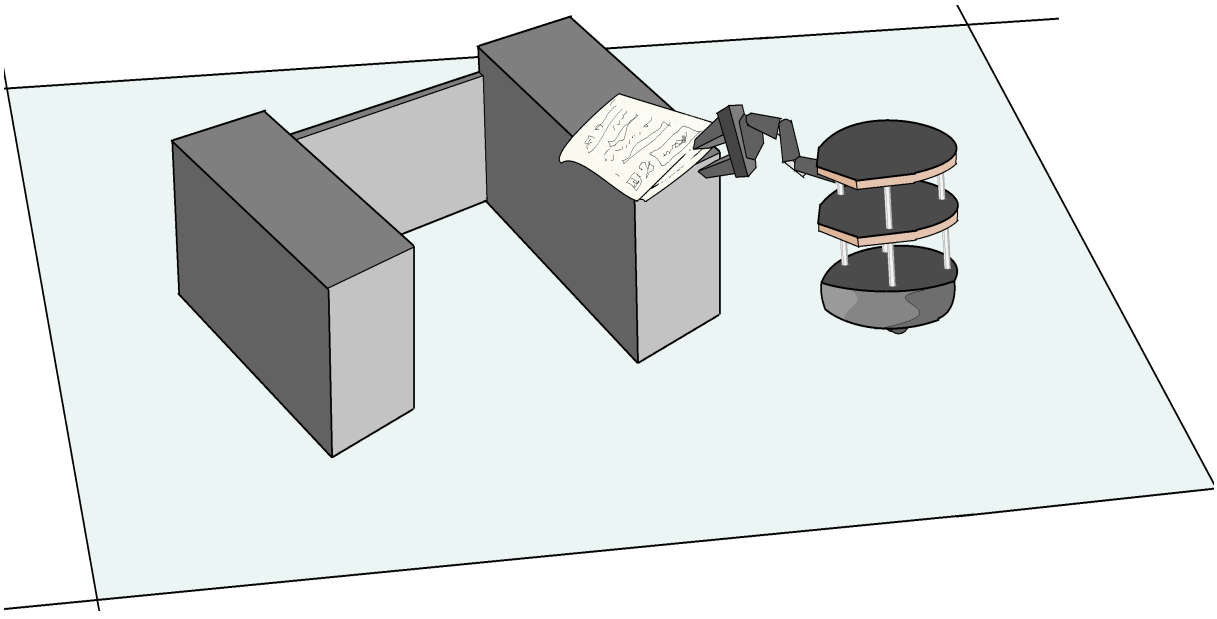
Representation of ROS servo test.

3.6.4 Flexibilities and Compatibility

The implementation of the robot arm has wide support. There are multiple software applications for motor control such as NEOBOTIX, etc. Due to the scope of our research, the team focus had been on implementing the robot arm with ROS (middleware). An initial compatibility challenge is the versions; make sure that the versions are properly accessible. For example, when downloading ROS compatible files, the files should be for the ROS Indigo version, and version of Arduino should be 1.0.5 or 1.0.6.

3.7 Scenario

3.7.1 Procedure



Students teams are tasked to create a field of terrain for the turtlebots to navigate. The terrain selection is a multi-group process, as the terrain should be for the class; therefore, as a class, placement of objects and field scenario size have been determined early on. While the GMapping may span a duration of a few minutes, the lengthier part is post processing (cleaning the map), as well as specific coordinates and self driving functionalities via python language code. As in the stated example, adjustments may incur hours of further tinkering, therefore, selecting a standard field early on in the robotics scenario project is essential for time management.

With the field terrain arranged, the Turtlebot operational stages commence. Our team of student investigators have adhered to the following procedures: sensor troubleshooting, study of map parameters (for securing an initially accurate map), map building (driving style is important for map accuracy), and map post processing (cleaning the image and arranging boundaries).

With the field and map complete, the actual mission for the scenario is created. While drafting on the large whiteboard, our team came up with multiple ideas. As part of the procedure for many successful missions, we implemented "cost and benefit analysis", whereby selecting tasks which are within the scope of the 2017 turtlebot project, while showing proof of our capabilities as robotic engineers.

With the plan complete, the final steps are: robotic arm implementation and coding. To mount the robotic arm onto the turtlebot, high quality mounting brackets have been purchased. As the mounting orientation of the robotic arm is unique (for comparatively higher range of motion, as well as placement, allowing for comparatively higher field of view in both the LiDAR line of sight and the Kinect sensing area.

3.7.2 Design

- User navigate the robot from charging station to the centre garage (consider it a book shelf), stop there for a certain amount of time (set delay time between 2 movements in nav_test.py).
- user hit play button on Arm Link software to activate the playback (which stores a series of predefined movement of the arm) to grab the newspaper.
- user navigate the robot back to charging station.

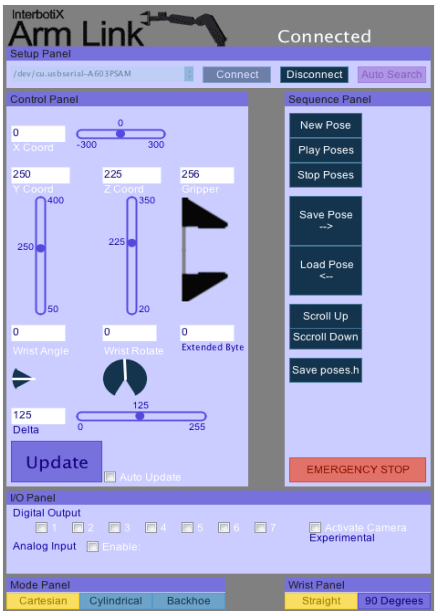
3.7.3 Implementation

- For the turtlebot:
  we use navigation config file (nav_test.py) with customized destination coordinators and delay time between two points 30 sec.
- For the Arm:
  we use Arm Link Software to create playback movements.

  Arm Link Software installation:

  - Download Arm Link Control Software (https://github.com/trossenrobotics/ArmLinkSoftware/releases/)
  - Open Arduino:
    File -> Examples -> Arm Link -> InterbotixArmLinkSerial
  - uncomment Pincher define:

    define ARMTYPE PINCHER //define REACTOR //define ARMTYPE WIDOWX
  - load the code on the board.
  - use Arm Link to set the predefined movement.
    * Hit new Pose button.
    * Use the controller and parameters in the Arm Link software to set the Pose.
    * Hit Save Pose button.
    * Redo the process with other poses.
    * Hit Save to file button to store the playback file.



Arm link software interface

- Explanation:

  After everything is set and done, in action. We open the Arm Link software, load the playback file then hit the Play poses buton during the 30 sec delay of turtlebot for it to grab the newspaper. after that the turtle bot will come back to its charging station.

- Summary:

  The final implementing has not been conducted due to the hardware malfunction we faced, we still couldn't make the arm to work again to work with Arm Link (We used to have it worked perfectly with Arm Link before, unfortunately before we finish the job the arm went unconscious). Indeed, if our equipment were fine, we should realize this scenario on time and without much effort.

4 Conclusion

4.0.1 Robotic Arm ability

The robotic arm has been mounted on the Turtlebot with an orientation that our team believes best accommodates the robot and maximizes capability and functionality (see image). By mounting the robot arm in a position that encourages infrared imaging (Kinect) and mapping (LiDAR), the engineers have the option to make low and high quality data maps. The quality of the mapping is a beneficial characteristic based upon where the data acquisition and processing is compiled (locally on robot; processed on work station).

4.1 Observed Changes

Observed throughout the Robotic Engineering project has been the growth of each group member and their capabilities surrounding Robotic Engineering. Whether the individual student investigator's strength upon entering the program had been software engineering, systems engineering, and auxiliary management, the outcome for the 2016 through 2017 has been one that:

- Provided a full scope of the Robotic Engineering process.
  As the field of robotic engineering is vast, members from our group of student laboratory investigators have been able to specialize in a niche as well a gain broad knowledge.
- Embraced teamwork formats.
  Upon completion of the initial 2016 lectures for Robotic Engineering, students were tasked to form teams by early 2017. Forming a team can be important as examined by group dynamics. A coherent group is efficient, also important is the process of group work for career enhancement.
- Encouraged troubleshooting.
  Alongside lectures and guidance, groups have been encouraged to investigate subject matter in solitude. Such processes place importance for ingenuity and can strengthen the dynamics of groups and teams.
- Intermediary pace.
  Accommodation of different student learning speeds and comprehension via lectures, group laboratory meetings, robotic exhibition and more. The professors have instructed very actively, calling attention to intermediate and advanced subject matter.

In accordance with the Robotics Engineering laboratory as stated above, students have been explorer of engineering facets and principles pertaining to Robotic Engineering and have adapted to fit the needs and betterment of the team.

4.2 Challenges

Throughout the debugging process, we realize that the hardware arbotix boards are not stable. Not so reliable due to a lot of issues have risen lately especially during 2 weeks after the holiday. We used to restart the operating system for it to start again, but now it totally goes non-response.

We tested a lot including switching laptops, cables and the arms of our classmates, and confirm it's the board which has the problem. 1 day after we received a new board from our professor, it worked! Finally we could make the arm work again, but it went down unconscious again 15 minutes later.

We wished that we could have much more time and supports to work it out; indeed the automation of the arm is the main show in our defence presentation. Honestly, time is one thing we didn't have enough to deal with these kinds of hardware issue, troubleshooting consumed way too much resources.

5 Sources

5.1 Books and Resources

- http://learn.trossenrobotics.com/arbotix/7-arbotix-quick-start-guide
- http://wiki.ros.org/Robots/TurtleBot
- http://edu.gaitech.hk/turtlebot/turtlebot-arm-pincher.html
- http://wiki.ros.org/joy/Tutorials/ConfiguringALinuxJoystick
- http://wiki.ros.org/dynamixel_motor
- http://learn.trossenrobotics.com/36-demo-code/137-interbotix-arm-link-software.html

5.2 Git link

- https://github.com/pirobot/rbx1
- https://github.com/turtlebot/turtlebot_arm
- https://github.com/vanadiumlabs/arbotix_ros
- https://github.com/roboticslab-fr/rplidar-turtlebot2
- https://github.com/arebgun/dynamixel_motor
- https://github.com/trossenrobotics/arbotix/archive/master.zip