# LOCALIZATION AND NAVIGATION CONTROL OF A DIFFERENTIAL-DRIVE AUTONOMOUS INTELLIGENT VEHICLE (AIV) WITH CART-SEARCHING AND AUTOMATIC PICKUP

by

Pognakvorn Meth

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Mechatronics

| | |
|---|---|
| Examination Committee: | Prof. Manukid Parnichkun (Chairperson) |
| | Dr. Mongkol Ekpanyapong |
| | Mr. Chaiya Thongrattana (External Expert) |

| | |
|---|---|
| Nationality: | Cambodian |
| Previous Degree: | Bachelor of Engineering in Electronics |
| | and Telecommunication Engineering |
| | Rajamangala University of Technology |
| | Thanyaburi Pattumthani Campus, Thailand |
| Scholarship Donor: | Thai Pipe Scholarship – AIT Fellowship |

Asian Institute of Technology
School of Engineering and Technology
Thailand
May 2021

# AUTHOR'S DECLARATION

I, Pognakvorn Meth, declare that the research work carried out for this thesis was in accordance with the regulations of the Asian Institute of Technology. The work presented in it are my own and has been generated by me as the result of my own original research, and if external sources were used, such sources have been cited. It is original and has not been submitted to any other institution to obtain another degree or qualification. This is a true copy of the thesis, including final revisions.

Date:

Name: Pognakvorn Meth

Signature:

# ACKNOWLEDGMENTS

I would like to express my gratitude to Prof. Manukid Parnichkun, my academic advisor for the support and guidance I received throughout my master's program. Without him this thesis would not have been able to accomplish.

I would like to thank to Asian Institute of Technology and my partial scholarship donor Dr. Yanyong Phataralaoha and Thai Pipe Industry Company for giving the scholarship for this master degree. Further, I would like to express my sincere thank my supervisor Mr. Chaiya Thongrattana to give me such an opportunity and immense support to conduct my thesis in such a prestigious company as Western Digital, Thailand. Moreover, I would like to give special thanks to Mr. Angkhan Thongsuksai and Mr. Panuwat Rodchom who are my mentors in Western Digital, Thailand for your overwhelming support to my thesis.

My utmost gratitude goes to my parents and family for their time, unconditional support, and guidance I received throughout my life. This journey would not have been possible without them. They are the most important people in my life.

And finally, I would like to thank all my friends, all the people in Advance Manufacturing Engineering (AME) Department and HR Department in Western Digital, Thailand, who helped me throughout this time period.

# ABSTRACT

This thesis presents the development of Differential-drive Autonomous Intelligent Vehicle (AIV) with autonomous cart-searching and picks up. An AIV is a mobile robot/vehicle used to transport materials in indoor manufacturing environments or a warehouse and designed to receive and distribute material or finished goods. AIV can self-navigate through a real indoor environment and be able to avoid dynamic obstacle and find an alternating route in case the pre-planned route is blocked. Autonomous Cart-Searching and Picks Up feature is developed to allow the AIV to find and get the specifically requested cart.

Two Laser Scanner unit are using to create a map in the environment. Extended Kalman Filter uses for the sensor fusion of IMU unit and wheel odometry. Then, Adaptive Monte Carlo Algorithm provides the localization of the robot on the map. Moreover, several local and global planner performances have tested. Then the camera detects and decode April Tag on the cart while searching. April Tag is used to localize the cart then proceed to engage to the cart to AIV.

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

**Figures** **Page**

# LIST OF ABBREVIATIONS

| | |
|---|---|
| AGV | = Automated Guided Vehicle |
| AIT | = Asian Institute of Technology |
| AIV | = Autonomous Intelligent Vehicle |
| AMCL | = Adaptive Monte Carlo |
| DWA | = Dynamic Window Approach |
| EKF | = Extended Kalman Filter |
| IMU | = Inertia Measurement Unit |
| PID | = Proportional Integral Derivative Controller |
| ROS | = Robot Operating System |
| SLAM | = Simultaneous Localization and Mapping |
| UAV | = Unmanned Aerial Vehicle |
| WD | = Western Digital Company |

# CHAPTER 1
# INTRODUCTION

## 1.1 Background

This thesis is discussing on the development of control system for the autonomous intelligent vehicle (AIV) that is based on Unicycle Model, a non-holonomic system, which a robot has some forward velocity but no instantaneous lateral motion. These systems are widely used for research in automation.

## 1.2 Introduction

The introduction of the Industry 4.0, the use of advance control system with cutting edge embedded software is increasing implemented. (Huang et al., 2018). It enables new method of production, value, and system optimization. This advancement is pushing the step toward the smart factories. In industry 4.0, Robot implementation is essential elements.

AIV, also known as Autonomous Mobile Robot, is a robot that runs without needing to draw a line but uses maps to determine the route. It is easy to create a map of the working area by taking the AIV to work in the direction that is required. Its body has sensors to scan and collect information of the surrounding area and then use the data to create a map for the working area. AIV is special due to the ability to work in dynamic environment without the need of guidelines and can be modified to fit various task in different industry.

## 1.3 Statement of the Problems

This thesis is focusing on the development of control system, navigation system, and algorithm for the differential-drive AIV robot operate in clean-room in manufacturing plant of Western Digital Company (WD. The current commercial AIV operated in WD can handle the payload of 60kg. The commercially costs much more than the in-house-designed designed AIV.

## 1.4 Objectives

The objective of the thesis is the development of the controlling system, navigation system, and algorithm for searching and engaging to the desired cart for the differential-drive AIV robot at Western Digital, Thailand. The mechanical structure is designed by Western Digital Engineer while the electrical system and control system are designed and discussed in this thesis.

Overall objective is to:

- Localization and Navigation Control of a Differential-drive Autonomous Intelligent Vehicle (AIV) with Cart-Searching and Automatic Pickup.

The proposed AIV will be able to:

1. Create the map of the working area by manually control the robot to teach it.
2. Monitor man scanning and operation externally.
3. Localization of the robot is done using SLAM based localization with 360-degree scan.
4. Navigate from known point to given point in the known working area.
5. Avoid static and dynamic obstacle, and able to find alternative route in case of the blocking of initially planned route.
6. Identify the cart identity using a camera and Quick Response code (QR code).
7. Search for a specifically identified cart in a specific area.
8. Localize the Cart using AprilTags.
9. Engage the cart to be moved.

## 1.5 Scope

The proposed AIV is working under the following limitations:

- The experiments will be tested in lab environment with consistent lighting.
- The floor surface should be evenly leveled and not slippery.
- The obstacles height should be more than the height of the laser scanner on AIV measured from the ground.
- Glass obstacles will not be detected or avoided.
- Overhang obstacle will not be detected or avoided.
- The opening of the cart must be no less than 25 cm wider than the width of the AIV.

## 1.6 Main Contributions

The main contribution to the development of this AIV are listed below:

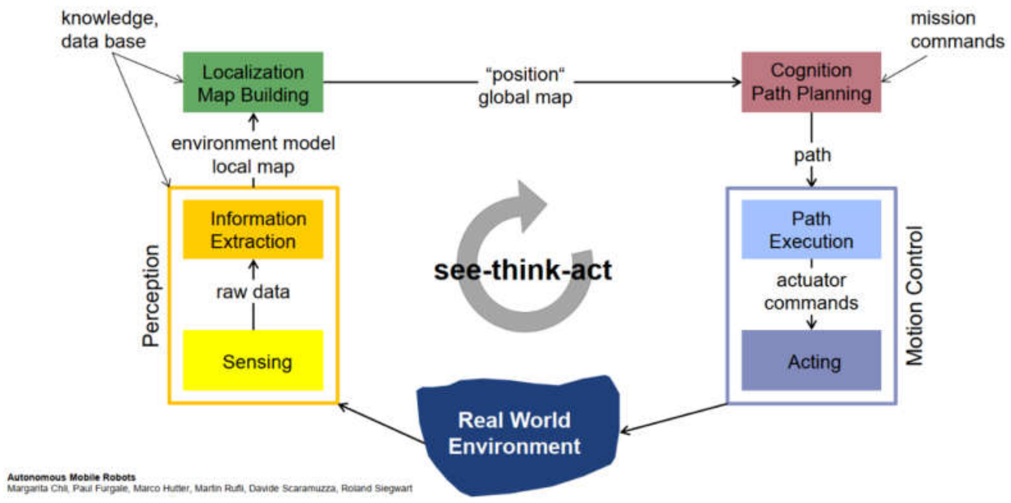
- Creating the math model of the robot.
- Design the controller to control the motion of the robot.
- Design and evaluate the algorithm for cart searching and pick-up.

# CHAPTER 2
# LITTERATURE REVIEW

The challenging problem of the AIV is the navigation in dynamic environment, a mobile robot must be aware of 3 main concepts. To begin with, it must know where it is (Localization). The second is where it is going. Finally, it is regarding to the way to gets there (Path Planning). The robot must have a model of the environment (Mapping). It could be provided or generate along the way. It must recognize and compare to the existing map to figure out its position in the map. The behavior of the mobile robot, see-think-act cycle can be expressed as below.

**Figure 2.1**

*See-Think-Act*



## 2.1 Warehouse Transportation Robot

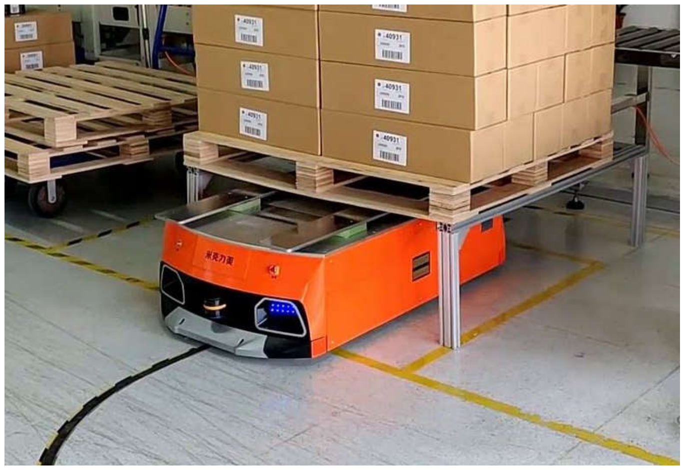
Many logistic companies are increasingly implementing the robot in their facility. This make the value of the robot market increase and is estimated to reach 6 billion USD by 2022.(Allen, 2020). Logistic robot includes automatic inventory storing moving or sorting.

### *2.1.1 Automated Guided Vehicles (AGV)*

AGV is the inventory moving robot in the storage facility or manufacturing facility. Its operation depends on the dictated guidance marked or embossed on the floor.

**Figure 2.2**

*Automated Guided Vehicles (AGV)*

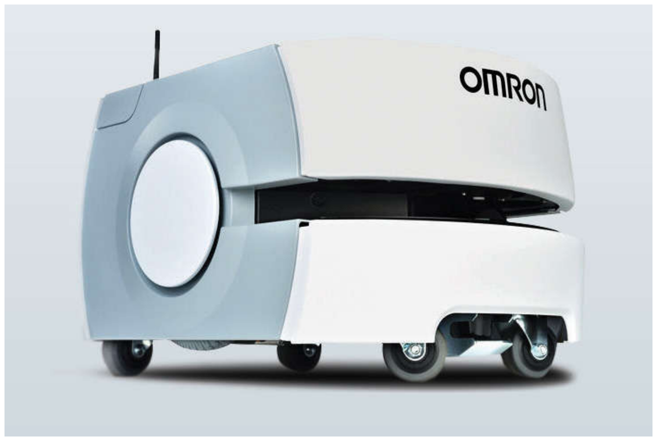


### *2.1.2 Autonomous Intelligent Vehicles (AIV)*

Autonomous Intelligent Vehicles function similarly to AGV, yet it does not depend on the dictated guidance. It depends on the map and sensors to acknowledge the environment and moving in the map without colliding into any obstacles.

**Figure 2.3**

*Autonomous Intelligent Vehicles (AIV)*

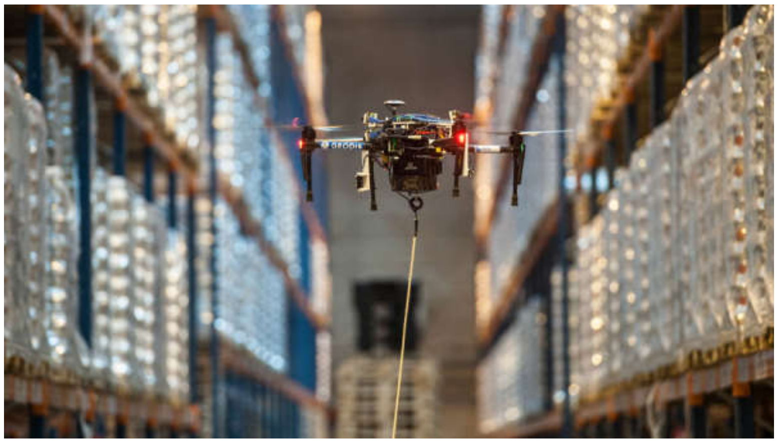


### *2.1.3 Unmanned Aerial Vehicles (UAV)*

Unmanned aerial vehicles function as real-time inventory monitor through the on-board video feed.

**Figure 2.4**

*Unmanned Aerial Vehicles (UAV)*



### *2.1.4 Warehouse Robots Navigation*

Depending on the types of robots, the navigation approaches can be implemented in different ways such as:

- Rail navigation
- Magnetic tape-based navigation
- Label-based navigation
- Laser-based navigation
- Vision-based navigation
- Geo-guidance

## 2.2 Kinematic of the Robot

It is important to understand the kinematics of the robot, there for the behavior of the robot can be predicted when the input is applied of the disturbance is encounter.

## 2.3 Mobile Robot Wheels Arrangement

The number of degrees of freedom and constraint of the robot is directly corresponding to the wheel’s type selection and placement.

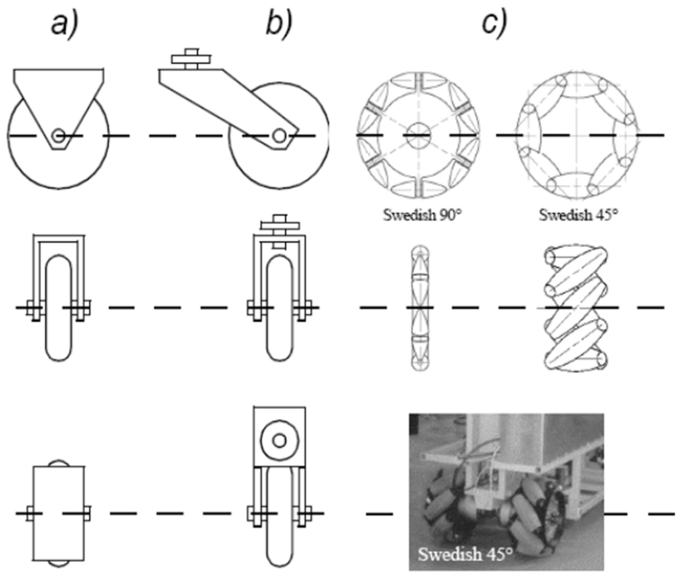
### *2.3.1 Wheel Types*

The wheels can be classified according to its dynamic as follow:

- Standard wheel: 1 degree of freedom as shown in Figure 2.5 (a)
- Castor wheel: 3DoF as shown in Figure 2.5 (b).
- Swedish wheel: as shown in Figure 2.5 (c).

**Figure 2.5**

*Types of Wheel*



### *2.3.2 Characteristics of Wheeled Robots*

Depending on the robot-wheel configuration, it affects the stability of the robot. It requires 3 wheels arrange in triangle shape where the center of mass in triangle to ensure the natural stability.

Wheel size has impact on the mobility of the robot. Bigger size wheel performs better on terrain such as grass, gravel, while small size is good for even and smooth terrain.

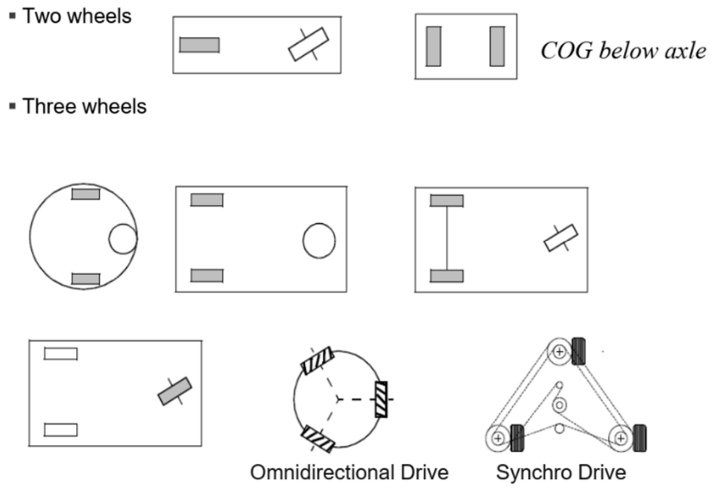
There are some wheels configurations as listed below:

- Synchro Drive: the wheel rotating at the same speed at the same direction.
- Differential Drive: it depends on the difference of the speed of right and left wheel to control the robot movement.

- Omnidirectional Drive: the difference speed of the wheels enabled the robot to move in every direction on 2D plane.

**Figure 2.6**

*Wheel Configurations*



## 2.4 Model of Differential Drive Robot

Supposed that reference frame is $\{x_i, y_i\}$, robot frame is $\{x, y\}$, robot position $P\ [x\ y\ \emptyset]$ is in cartesian co-ordinate system of reference frame. The transformation between the 2 fames can be express using this homogenous transformation matrix.

$$P_i = AP$$

Where,

$$A = \begin{bmatrix} \cos\emptyset & \sin\emptyset & t_x \\ -\sin\emptyset & \cos\emptyset & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

The movement of Differential Drive Robot can be control by varying the velocity and the direction of each wheel, $v_r$, velocity of right wheel, $v_l$. Then the position and the heading of the robot can be determined by taking the radius of the wheel, $R$ and the width of the robot, $L$ into consideration. The relation between the input $v_r$, $v_l$, the position and be derived as below.

$$\dot{x} = \frac{R}{2}(v_r + v_l)\cos\emptyset$$

$$\dot{y} = \frac{R}{2}(v_r + v_l)\sin\emptyset$$

$$\dot{\emptyset} = \frac{R}{L}(v_r - v_l)$$

For easy conceptualize it can be substituted by Unicycle model, which consider only translational and angular velocity. Thus the input of the system are $v$, translational velocity, and $\omega$, angular velocity. (Malu & Majumdar, 2014)

$$\dot{x} = v\cos\emptyset$$

$$\dot{y} = v\sin\emptyset$$

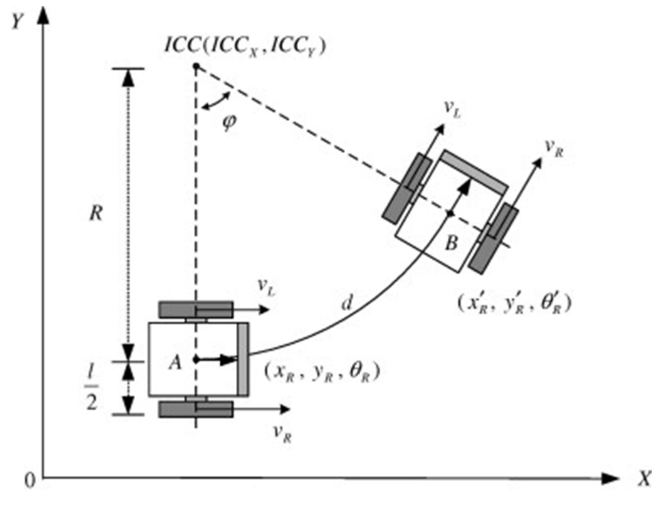
$$\dot{\emptyset} = \omega$$

$$v_r = \frac{2v + \omega L}{2R}$$

$$v_l = \frac{2v - \omega L}{2R}$$

**Figure 2.7**

*Differential Drive*

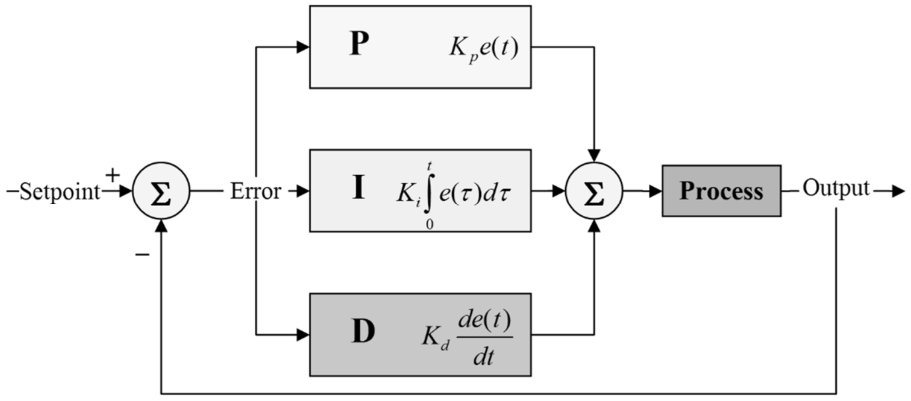


## 2.5 PID Controller

PID is a proportional, integral, derivative controller. The output is the summation of the 3-control terms. PID is a closed loop feedback controller, where the P I D term are calculated according to error of variable set-point to process variable. The controller behavior and system performance can be tuned by tuning the P I D gain.

**Figure 2.8**

*PID Controller*



## 2.6 2D Laser Rangefinder

2D laser rangefinder or laser scanner uses the laser time-of-flight to determine the distance to the object. It measures the time laser took to travel to and reflect with high precision. The laser scans the area in a 2D plane around itself. The advantage of laser is the measurement accuracy, high sampling rate, high angular resolution.

There are some limitations of Laser rangefinder which are:

- Transmitted energy in Nature. The Laser rangefinders are not capable of detecting optically transparent obstacles. Also, reflection from small dust particle should make the measurement invalid.
- The measuring distance.

Beside the limitation above, laser scanner provides accurate measurement. Many literatures implement it in the robot.

## 2.7 Localization and Mapping

Localization is the process of locating the robot position in the map. The map is required to do the localization.

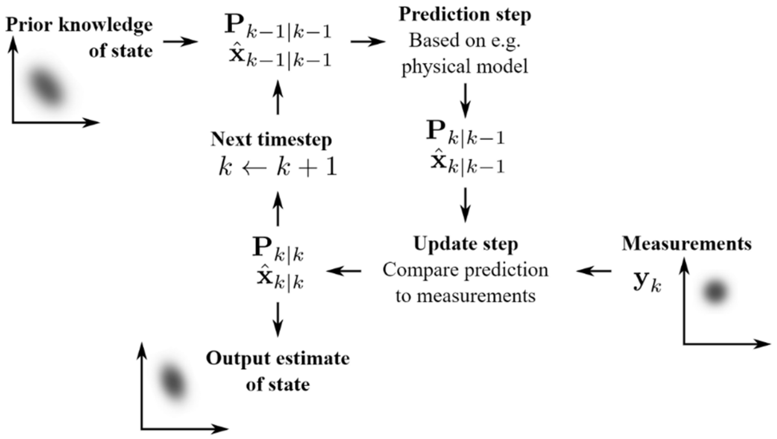
### *2.7.1 Kalman Filter Localization*

Kalman Filtering is a method applying to the comprehend the data with random noise. It can merge the uncertainties of the current state described as Gaussian probability distribution.

The algorithm is recursive of prediction and update procedure. The algorithm estimates the current state with the uncertainties. By observing the next measurement, the estimation is updated with the average of the weight.

**Figure 2.9**

*Kalman Filter Process*



### *2.7.2 Simultaneous Localization and Mapping (SLAM)*

SLAM is a method of localizing, generating and update the map at the same time. It is a popular method applied for robot application. The process comprises of many steps. SLAM aims to update the current location of the robot by using the combination of different sensor such as scanner data, wheel odometry, visual odometry.

**Table 2.1**

*Various SLAM Algorithm*

| Name | Loop Closure | Algorithm |
|---|---|---|
| Gmapping | Yes | Particle filter |
| Hector SLAM | No | Scan Matching + EKF |
| Get SLAM | Yes | Graph based SLAM |
| Google Cartographer | Yes | Graph based SLAM |
| Ezthasl icp mapping | Yes | ICP based SLAM |

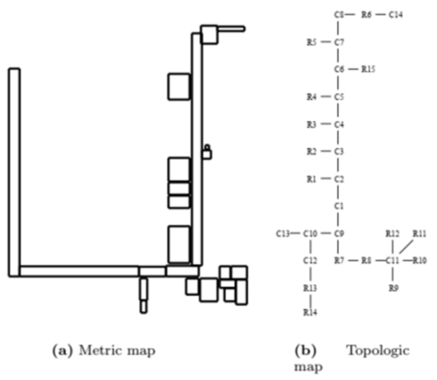
### *2.7.3 Map Representations*

The environment can be represented in different way such as:

- Topological Maps: as shown in Figure 2.10

**Figure 2.10**

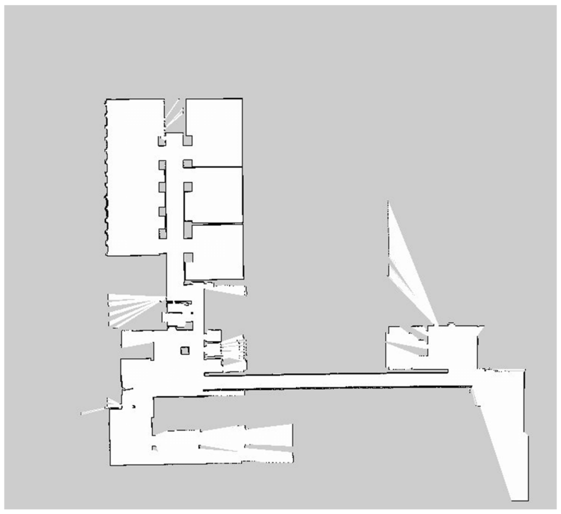
*Metric Map vs Topologic Map*



- Feature maps: map that represent the point, line, corner, or edge
- Grid maps: The maps representation to a specific feature

**Figure 2.11**

*Grid Map*



## 2.8 Path Planning and Navigation

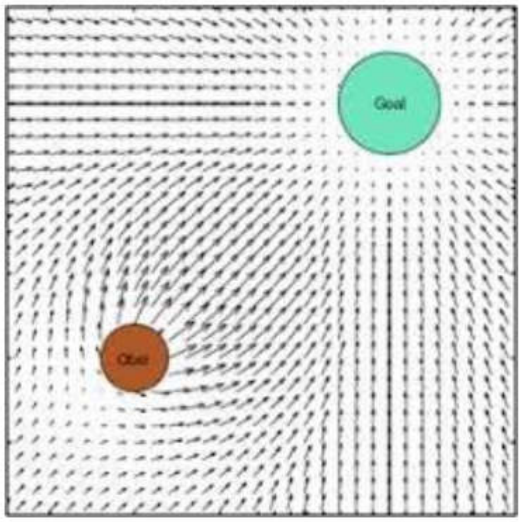
Path planning and navigation are the method of guiding the robot to move to the goal position successfully without colliding into any obstacle.

### *2.8.1 Potential Field Methods*

To regulate the robot in the space, the potential field algorithm uses the artificial field. The simple form of potential motion planning is to let the robot follow the potential gradian to the goal.

**Figure 2.12**

*Potential Field in 2D Space*



### *2.8.2 Planning*

Planning is the process of finding the optimum path for the robot to the goal according to the robot current position, and the condition of the environment.

### *2.8.3 Obstacle Avoidance*

The obstacle avoidance prevents the robot to collide into any object. It predicts the trajectory of the robot. Below are some methods:

- Bug algorithm: it works by moving the robot toward the goal until there is obstruction, then it follows the edge of the obstruction until it found the goal.
- Dynamic Window Approach (DWA): it is a sampling-based algorithm. Velocity-command pairs are simulating for a set period and the best velocity-command pair is chosen.
- Velocity Obstacles (VO): it deals with moving obstacle.

## 2.9 Robot Operating System

ROS is a publicly available firmware that allow easy the integration of sensors and machine for a complex robot operation. ROS functions as the meta-operating system that provide hardware preoccupation, low level control, inter process and package management. It can be run in different computers while share and utilizing the same resource. The shared data is handled by ROS and allocated according to the topic. The data in the topic is called message. A node is a program or an execution. The node can subscribe and publishes many topics.

**Figure 2.13**

*Robot Operating System (ROS)*

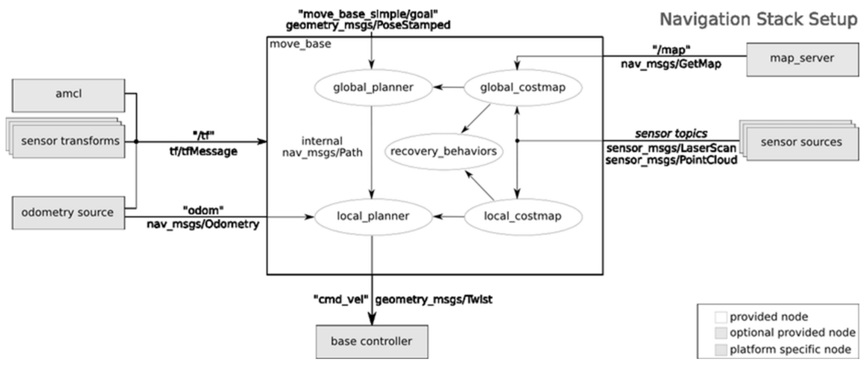
ROS.org

### *2.9.1 ROS Navigation Stack*

ROS navigation stack is the core of the robot navigation. It combines the map, localization, path planning. It utilizes the data and command the velocity-command to the robot to move.

**Figure 2.14**

*Navigation Stack*

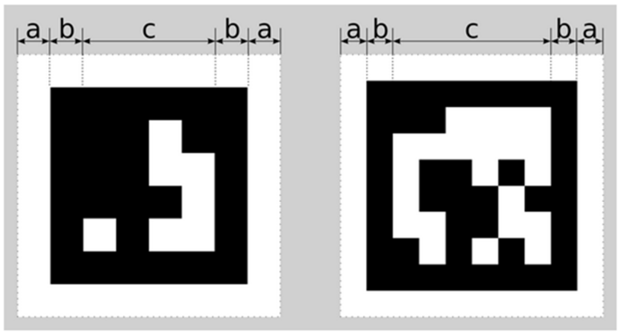


## 2.10 AprilTag

Apriltag is used as visual fiducial mark. The tag is detected from a camera while the position and orientation of the tag can be also obtained. The number of tag, data storing and decoding is different according to the tag family.

**Figure 2.15**

*Apriltag*



*A 16h5 tag (left) and a 36h11 tag (right). AprilTags consist of a mandatory white (a) and black (b) border and a variable amount of data bits (c).*

# CHAPTER 3
# METHODOLOGY

This chapter is primarily covered the performance of planning that was developed by doing alternative literature review. The plan of study will be strictly aligned to the objective that was mentioned earlier in the report. This thesis will be carried out in the following way. First is Hardware Setup. It includes the main structure assembly of the robot, motors, drivers, wheels, various sensors, circuit, power distribution, and control boards. Second is the design and implementation of controller running on ROS.

## 3.1 Mechanical Structure

The proposed design of the robot is a cylindrical shape robot. The diameter of this robot is 450 mm, and the height is 445 mm when the engaging mechanism (triangular extrusion) is fully extended. This robot sits on 2 differential driven wheels at the middle of the robot and 2 caster wheels placing one at the front and one at the back as can be seen in figure21. It is powered by a 24V 50AH Li-ion battery. There are 2 Laser Range Finder placed at the front and the back of robot for mapping, localization, and obstacle avoidance. A camera is placed the upper front of the robot for QR detection and tag localization. The engaging mechanism is a triangular shape block that raise from the top of the robot to engage the cart to the robot.

**Figure 3.1**

*View of WD AIV*

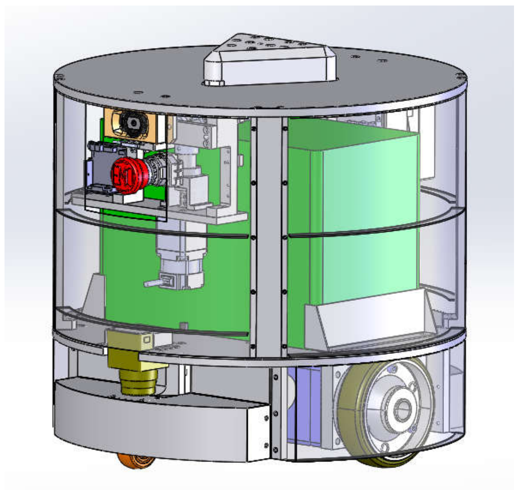
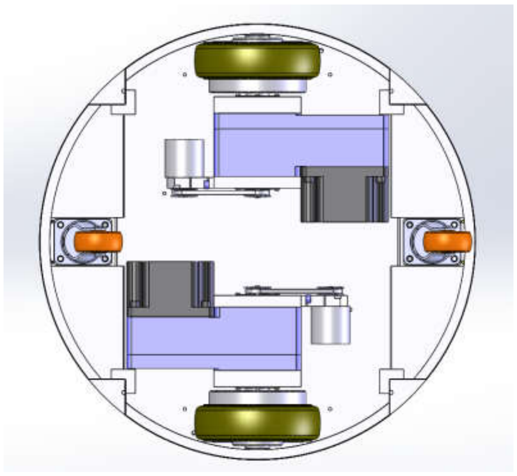


**Figure 3.2**

*Bottom View of WD AIV*

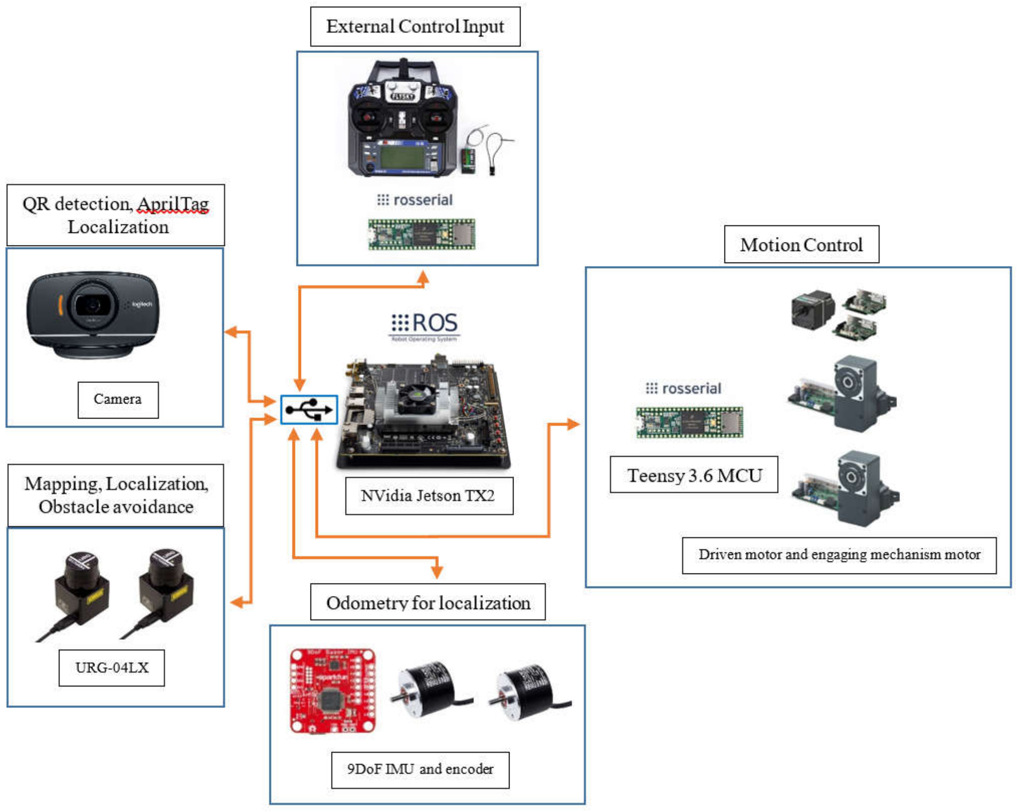


## 3.2 System Design

The design of the system can be divided into 5 main parts: Central Processor, Motion Control, External Control Input, Perception sensors, and Power distribution.

**Figure 3.3**

*System Design*


External Control Input
rosserial
QR detection, AprilTag Localization
Camera
ROS
NVidia Jetson TX2
Motion Control
rosserial
Teensy 3.6 MCU
Driven motor and engaging mechanism motor
Mapping, Localization, Obstacle avoidance
URG-04LX
Odometry for localization
9DoF IMU and encoder

### *3.2.1 Central Processor*

The main system of the AIV is the Central Processor which all the programs to operate the AIV are running on. The Nvidia Jetson TX2 is chosen for this job. Jetson TX2 is built using an NVIDIA Pascal™-family GPU with an 8GB, 59.7GB/s of memory bandwidth memory card (NVIDIA, 2021). Considering the efficient power consumption with the high performance, it is suitable to be used on the mobile robot which requires high computation power.

**Figure 3.4**

*NVIDIA Jetson TX2 Development Board*

The board is equipped with onboard Ethernet, Wi-Fi, and Bluetooth modules makes it convenient for this application. There are only one USB 3.0 port for external connection thus USB3.0 hubs are used to extent connectivity to external devices and sensors.

The operating system operates on this board is Ubuntu 18.04 LTS. On top of that, Robotic Operating System Melodic (ROS-Melodic) is use manage, control, and operate the AIV.

### *3.2.2 Motion Control*

In order to be able to control the movement of various moving parts of the AIV, a good controller is crucial and need to be well designed. The main moving parts on this AIV consists of 2 main elements including wheels to for AIV movement and lifer for AIV to cart engagement.

***3.2.2.1*** **Drive motor.** The drive motors are responsible for AIV movement. This AIV is a differential drive robot, thus 2 driver motor are needed to drive right and left wheel. The motor that is being used is brushless Oriental Motor BHL series, BLH 5100K, which is rated at 100W power. This particular motor is used in combination with 20:1 gear reduction head providing good speed and tremendous amount of torque. The motor is running on 24V system. The Oriental motor driver is used in this application.

**Figure 3.5**

*Oriental Motor BLH5100K with the Motor Driver*

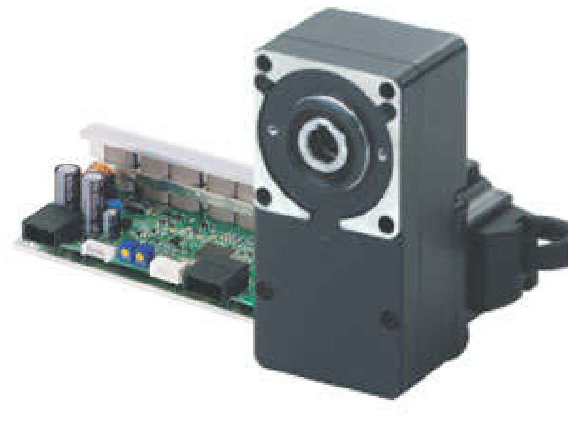


**Table 3.1**

*Motor Specification*

| Technical Specification | |
|---|---|
| Shaft/Gear Type | Combination Type Parallel Shaft Gearhead |
| Rated Output Power (Continuous) | 100W |
| Power Supply Input Rated Voltage | 24V |
| Power Supply Input Permissible Voltage Range | ±10% |
| Power Supply Input Rated Input Current | 6.0A |
| Power Supply Input Maximum Input Current | 9.8A |
| Rated Torque (Motor Shaft) | 0.4N・m |
| Starting Torque (Motor Shaft) | 0.5N・m |
| Rated Speed (Motor Shaft) | 2500r/min |
| Speed Control Range | 10-300r/min |
| Acceleration Time / Deceleration Time | 0.5~10 sec |
| Speed Setting Method | ・Internal potentiometer.<br>・External potentiometer: 20 kΩ, 1/4 W<br>・External DC voltage: 0~5 VDC, 1 mA min. (Input Impedance: 47 kΩ). |
| Gear Ratio | 20:1 |

***3.2.2.2 Encoder.*** The incremental rotary encoder is connected to motor output shaft via a timing belt to measure the output shaft rotation speed and sense the position of the robot. The Omron E6B2 is used for this AIV. This encoder provides a good resolution of 1/2000 of rotation which is beneficial for accurate speed and position feedback.

**Figure 3.6**

*Omron Encoder*

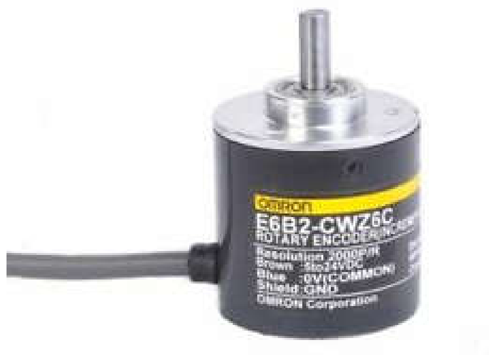


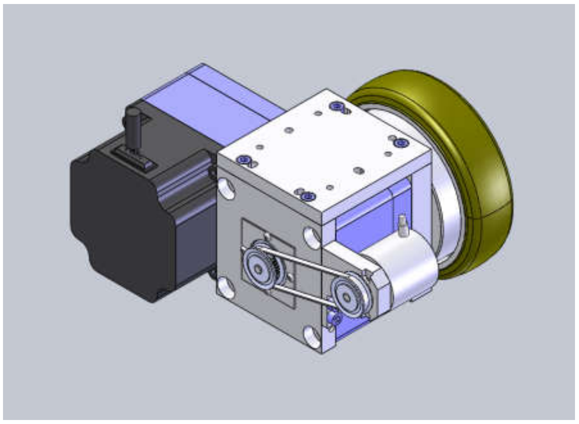
**Table 3.2**

*Encoder Specification*

| Technical Specification | |
|---|---|
| Power supply voltage | 5-24V |
| Resolution | 2000 |
| Output Configuration | NPN open-collector output |
| Maximum response frequency | 100 kHz |
| Maximum permissible speed | 6,000 rpm |

**Figure 3.7**

*Motor, Encoder, and Wheel Assembly*



***3.2.2.3 Lift Motor.*** The AIV is designed with a raise and lower able engaging triangular block function as the engaging point between the AIV and the cart. The motor responsible for raising and lowering the block is BLH015K brushless Oriental Motor with 20:1 gear reduction head.

**Figure 3.8**

*Oriental Motor BLH015K*

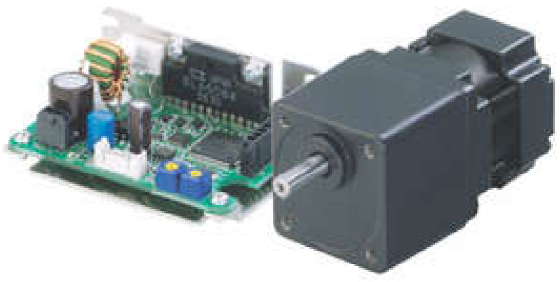
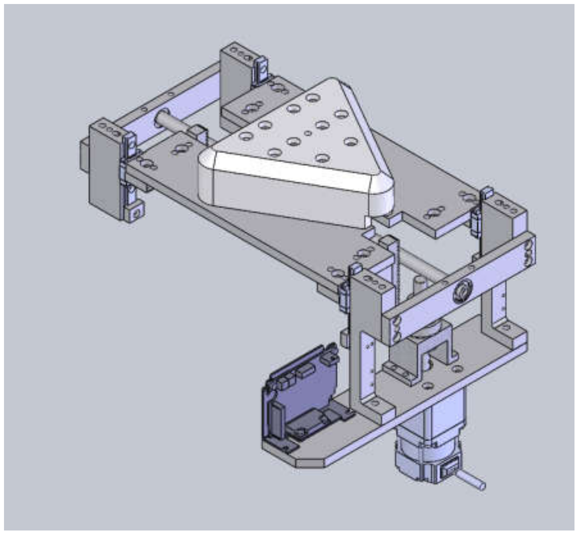


**Figure 3.9**

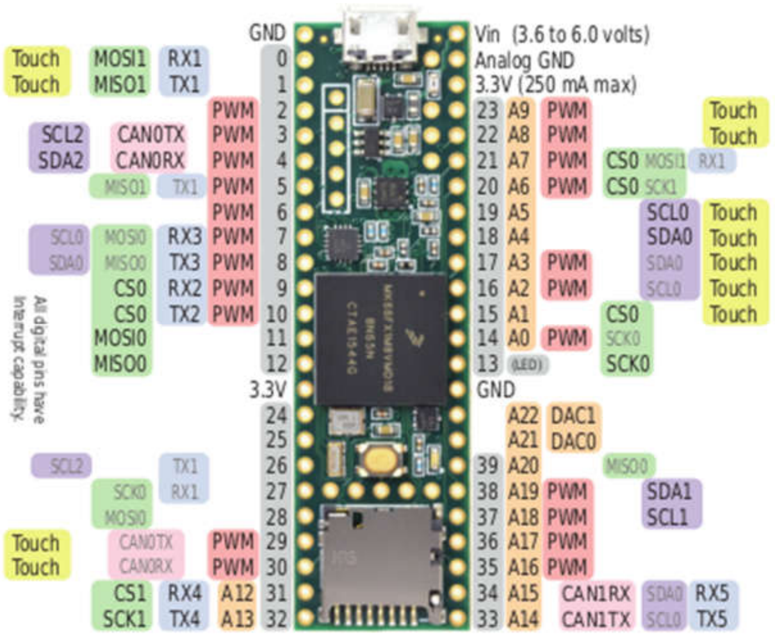
*Engaging Block of Lift Assembly*



***3.2.2.4 Microcontroller.*** The microcontroller is used to take care of the control of wheel speed, motion control of AIV, lift mechanism control and communicate with ROS. For this application, teensy 3.6 is selected for this function. It is a powerful is a 3.3V ARM Cortex-M4 board running at 180 MHz is a small form factor board.

**Figure 3.10**

*Teensy 3.6*



***3.2.2.5 Motion Controller Design.*** ROS operation, the ROS navigation stack publishes a set of velocity command to a controller node to make the AIV move called geometry_msgs/Twist.msgs. The Twist message consists of 2 vector sets, linear and angular velocity in x, y z axis respectively of the robot. The controller node is responsible for translating the input command to motor speed command to achieve robot motion accordingly.

**Figure 3.11**

*Microcontroller Connection*

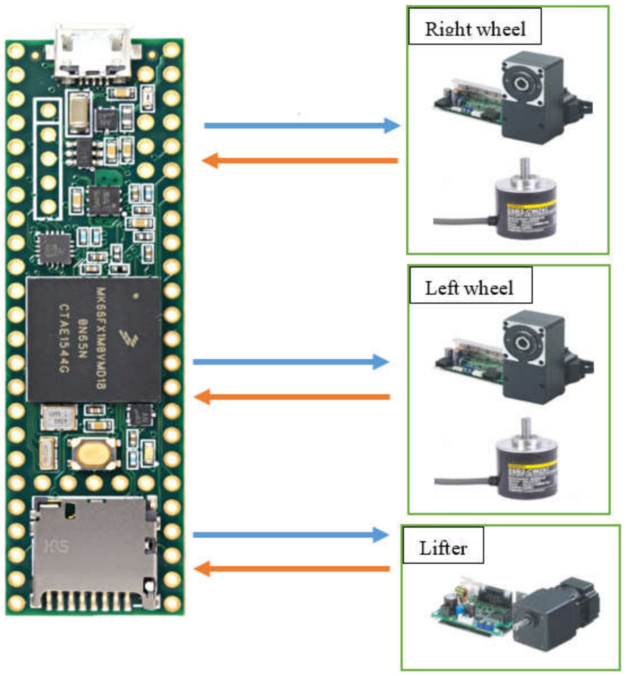
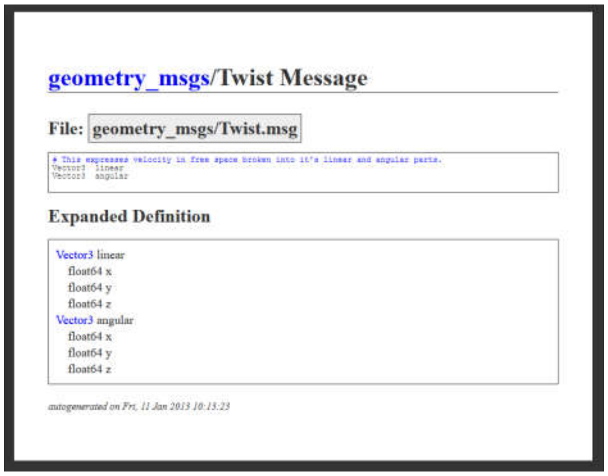


**Figure 3.12**

*Twist Message*



geometry_msgs/Twist Message

File: geometry_msgs/Twist.msg

```
# This expresses velocity in free space broken into it's linear and angular parts.
Vector3  linear
Vector3  angular
```

Expanded Definition

```
Vector3 linear
  float64 x
  float64 y
  float64 z
Vector3 angular
  float64 x
  float64 y
  float64 z
```

*autogenerated on Fri, 11 Jan 2013 10:13:23*

For this particular AIV, the linear velocity in x axis command moves the robot forward or backward, and angular velocity in z axis rotate the AIV clockwise or counterclockwise while other are not necessary. The command speed for right and left can be calculated as follow:

$$V_r = \dot{x} + \frac{\dot{\theta}\, W}{2}$$

$$V_l = \dot{x} - \frac{\dot{\theta}\, W}{2}$$

Where,

$V_r$ , $V_l$ are velocity of right and left wheels.

$\dot{x}$ , $\dot{\theta}$ are linear velocity in x axis and angular velocity in z axis

$W$ is the distance between right and left wheel

The measured angular velocity of each wheel by the encoder can be calculated as follow:

$$V_r = \frac{encoder\ count\ right \times 2\pi \times wheel\ radius \times 0.0005}{time\ interval}$$

$$V_l = \frac{encoder\ count\ left \times 2\pi \times wheel\ radius \times 0.0005}{time\ interval}$$

### *3.2.3 External Control Input*

External Control Input is required to control the AIV remotely. There are many methods for this, yet for reliable connection over the distance, the commercial remote controller is use for this application.

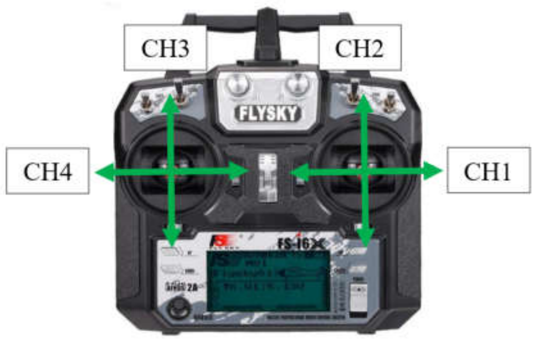
**Figure 3.13**

*FLYSKY Radio Transmitter and Receiver*

Two control inputs are required to control the AIV. Channel 2 (CH2) and Channel 4 (CH4) are uses for velocity in X axis and angular velocity in Z axis, respectively. Then control inputs are transmitted to and received by the Receiver.

**Figure 3.14**

*Input Channel Mapping*



At receiver side, the input command receives for the transmitter is sent microcontroller Arduino MEGA 2560. The Arduino Mega maps the command data to geometry.msgs/Twist.msgs. Then Arduino functions as a node name /serial_node_teensy published it to ROS on /remote_cmd_vel topic.

**Figure 3.15**

*Receiver and Arduino MEGA Connection*

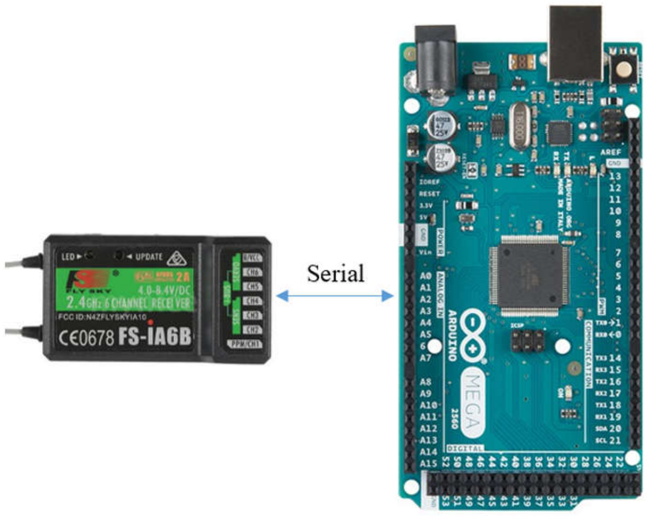
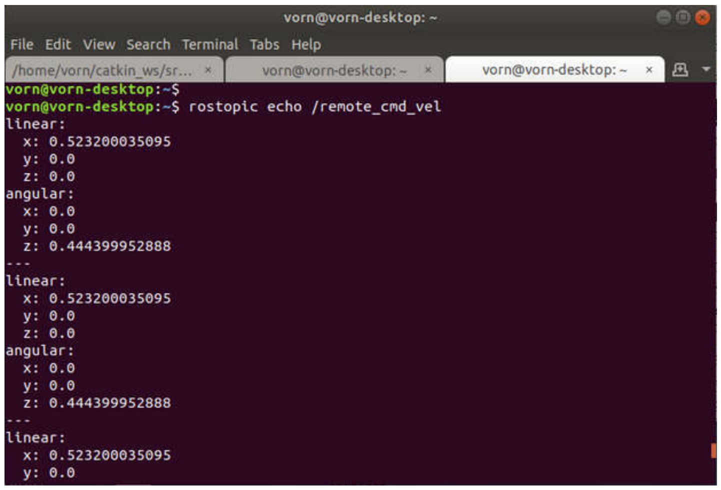


**Figure 3.16**

*Remote_cmd_vel Monitor*


vorn@vorn-desktop: ~
File Edit View Search Terminal Tabs Help
/home/vorn/catkin_ws/sr...
vorn@vorn-desktop: ~
vorn@vorn-desktop: ~

```
vorn@vorn-desktop:~$
vorn@vorn-desktop:~$ rostopic echo /remote_cmd_vel
linear:
  x: 0.523200035095
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.444399952888
---
linear:
  x: 0.523200035095
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.444399952888
---
linear:
  x: 0.523200035095
  y: 0.0
```

### *3.2.4 Perception Sensors*

The AIV is equipped with various sensors to perceive the environment around it. The data from those sensors will be used and manipulate to enable robot to work autonomously.

***3.2.4.1 Hokuyo 2D Laser Rangefinder.*** The Hokuyo URG-04LX-UG01 Scanning Laser Rangefinder is selected to use in this project. Considering the compact size, the detectable range, resolutions, and less energy usage makes it the good choice for this project

**Figure 3.17**

*Hokuyo URG-04LX-UG01*

Below are the specifications of the Hokuyo URG-04LX-UG01

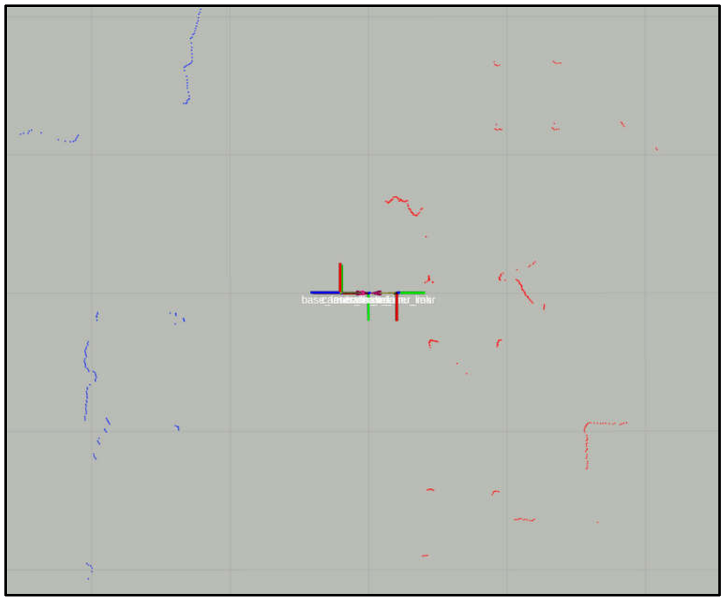
- Power source: 5VDC±5 (USB Bus power)
- Light source: Semiconductor laser diode(785nm), Laser safety class 1
- Measuring area: 20 to 5600mm (white paper with 70mm×70mm), 240°
- Accuracy: 60 to 1,000mm: ±30mm, 1,000 to 4,095mm: ±3 of measurement
- Angular resolution: Step angle: Approx. 0.36° (360°/1,024 steps)
- Scanning time: 100ms/scan
- Noise: 25dB or less
- Interface: USB2.0/1.1[Mini B] (Full Speed)
- Command System: SCIP Ver.2.0

- Ambient illuminance*1: Halogen/mercury lamp: 10,000Lux or less, Florescent: 6000Lux (Max)
- Ambient temperature/humidity: -10 to +50 degrees C, 85% or less (Not condensing, not icing)
- Vibration resistance: 10 to 55Hz, double amplitude 1.5mm each 2 hour in X, Y and Z directions
- Impact resistance: 196m/s2, Each 10 time in X, Y and Z directions
- Weight: Approx. 160g

There are 2 Hokuyo URG-04LX-UG01 onboard the AIV, one is for front scanning and another one for rear scanning. Combining the scanning data from the 2 sensors, the 360 full coverage around the AIV is achieved.

**Figure 3.18**

*Front Scan in Blue Color and Rear Scan in Red*



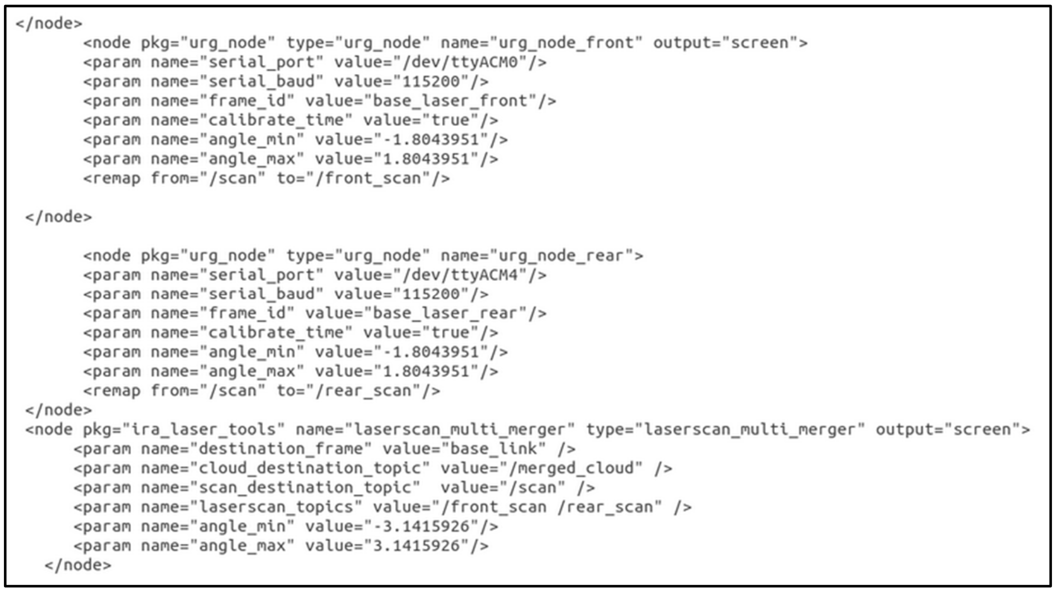
Urg_node packaged is utilized to get the scan data from each laser scanners and publish it to its respected topics such /base_laser_front and /base_laser_rear. The lunch file defines the serial port and serial baud rate and other parameters.

Then the ira_laser_tools package is used to combind the 2 scan data into 1 scan data. The node subscribes to /base_laser_front and /base_laser_rear then publishes the merged data to topic name /scan.

The below figure shows the lauch file that aquires data from 2 laser scanners and merge them into topic.

**Figure 3.19**

*Lunch File for the Scanner*



```
</node>
      <node pkg="urg_node" type="urg_node" name="urg_node_front" output="screen">
      <param name="serial_port" value="/dev/ttyACM0"/>
      <param name="serial_baud" value="115200"/>
      <param name="frame_id" value="base_laser_front"/>
      <param name="calibrate_time" value="true"/>
      <param name="angle_min" value="-1.8043951"/>
      <param name="angle_max" value="1.8043951"/>
      <remap from="/scan" to="/front_scan"/>

 </node>

      <node pkg="urg_node" type="urg_node" name="urg_node_rear">
      <param name="serial_port" value="/dev/ttyACM4"/>
      <param name="serial_baud" value="115200"/>
      <param name="frame_id" value="base_laser_rear"/>
      <param name="calibrate_time" value="true"/>
      <param name="angle_min" value="-1.8043951"/>
      <param name="angle_max" value="1.8043951"/>
      <remap from="/scan" to="/rear_scan"/>
 </node>
 <node pkg="ira_laser_tools" name="laserscan_multi_merger" type="laserscan_multi_merger" output="screen">
     <param name="destination_frame" value="base_link" />
     <param name="cloud_destination_topic" value="/merged_cloud" />
     <param name="scan_destination_topic"  value="/scan" />
     <param name="laserscan_topics" value="/front_scan /rear_scan" />
     <param name="angle_min" value="-3.1415926"/>
     <param name="angle_max" value="3.1415926"/>
  </node>
```

***3.2.4.2 Inertial Measurement Unit (IMU).*** The 9DoF Razor IMU from Sparkfun is installed on the AIV. Before the IMU can be used it needs to be properly calibrated since the other magnetic components and meatal structure effect the performance for the IMU.

**Figure 3.20**

*9DoF Razor IMU*

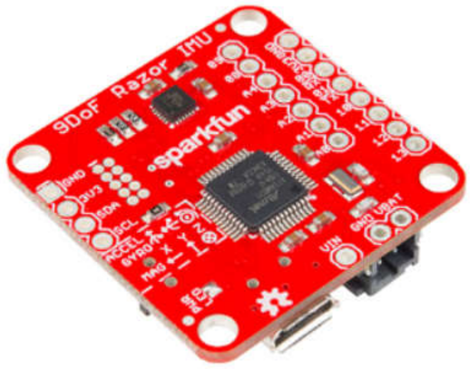
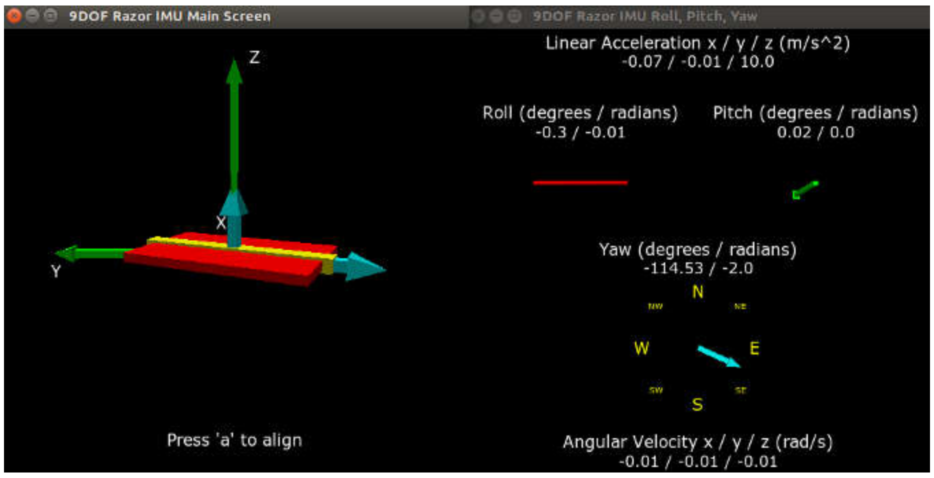


**Figure 3.21**

*IMU Visualization*



Linear Acceleration x / y / z (m/s^2)
-0.07 / -0.01 / 10.0

Roll (degrees / radians)
-0.3 / -0.01

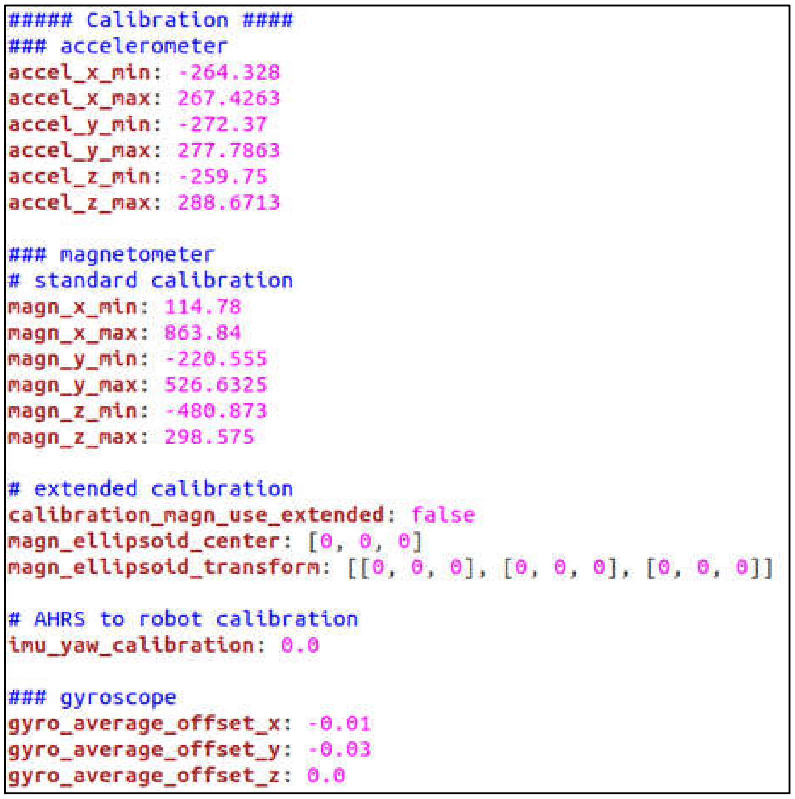
Pitch (degrees / radians)
0.02 / 0.0

Yaw (degrees / radians)
-114.53 / -2.0

Angular Velocity x / y / z (rad/s)
-0.01 / -0.01 / -0.01

**Figure 3.22**

*Calibrated Parameters*



```
##### Calibration ####
### accelerometer
accel_x_min: -264.328
accel_x_max: 267.4263
accel_y_min: -272.37
accel_y_max: 277.7863
accel_z_min: -259.75
accel_z_max: 288.6713

### magnetometer
# standard calibration
magn_x_min: 114.78
magn_x_max: 863.84
magn_y_min: -220.555
magn_y_max: 526.6325
magn_z_min: -480.873
magn_z_max: 298.575

# extended calibration
calibration_magn_use_extended: false
magn_ellipsoid_center: [0, 0, 0]
magn_ellipsoid_transform: [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

# AHRS to robot calibration
imu_yaw_calibration: 0.0

### gyroscope
gyro_average_offset_x: -0.01
gyro_average_offset_y: -0.03
gyro_average_offset_z: 0.0
```

***3.2.4.3 Camera.*** Logitech B525 HD camera is use for April tag scanning and localizing. The camera is a 1080p Full HD at 30 frames per second with the field of view of 69 degree.

**Figure 3.23**

*Logitech B525*

### *3.2.5 Power Distribution*

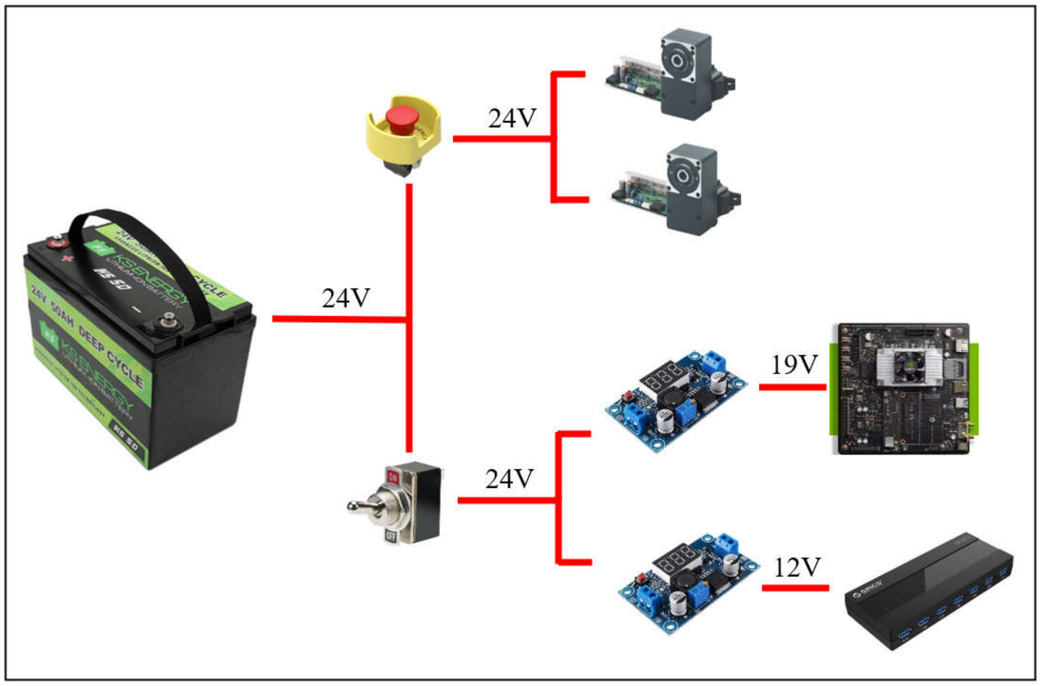
The AIV is powered by a 50 Ah 24V Li-ion battery. The regulators are mandatory to regulate the voltage to suit the electrical components used onboard AIV. The power distribution switches as well as the main voltage emergency cut-off is equipped.

**Figure 3.24**

*24V Li-ion Battery*

**Figure 3.25**

*Power Distribution*



## 3.3 Robotic Operating System

The Robot Operating System is a framework to write the software of the robot. There are numbers of tools, libraries, and methods that are helpful for the creation of the complex tasks and reliable the behavior of the robot across the platform variation.

**Figure 3.26**

*ROS Framework*

ROS allows user to group the programs into packages. There can be multiple packages, each handles a specific task in the robot operation. Separation of the programs into packages reduces code complexity and makes it easy to scale the application and debug.

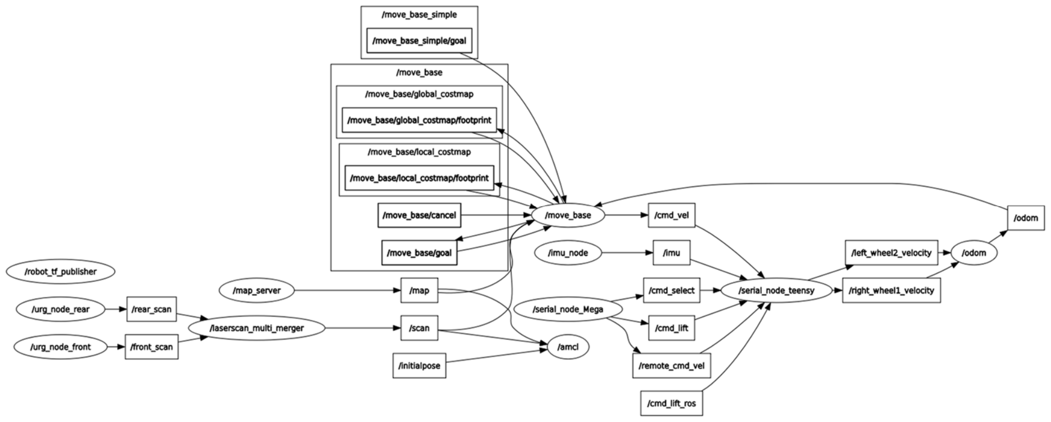
It is possible to have more than one program in a package. A node in ROS is define as each program. A node is an executable program or a process that performs computation inside the application. Each node is launched separately. ROS communication functionalities, for example topic, services, actions, etc. are used for the communication of each node between each other.

ROS Topics are labeled buses that nodes interchange messages. Topic is anonymously publishers/subscriber denoted. Topics are unidirectional streaming communication. Multiple nodes can publish and subscribe to a topic while a node can publish and subscribe to multiple topics. Nodes subscribe to a relevant topic interested and publish generated data to the relevant topic.

In this thesis, the programs for the AIV are implemented the ROS melodic running on Ubuntu 18.04 LTS.

**Figure 3.27**

*ROS Nodes and Topics*



## 3.4 ROS Tf

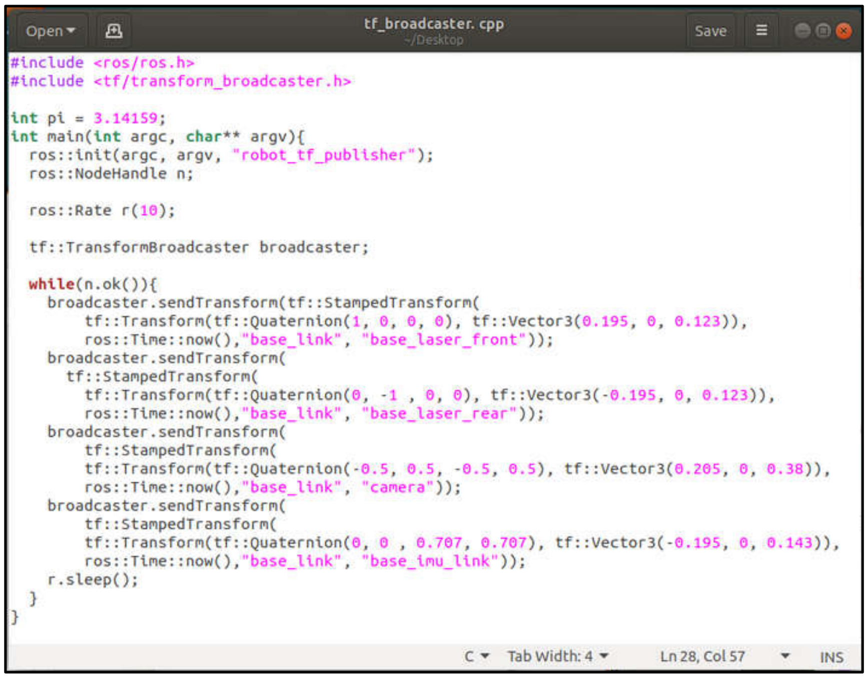
Tf is a package that users can keep track of the coordinate frame. Users can transform points, vectors, between coordinate frames at any preference at any given time.tf also carry on the transformation between the 2 frames is recorded

There are 2 types of transformation between coordinate frames, static transformation, and dynamic transformation. Static transform is a constant transformation between 2 frames, whereas dynamic transform is subject to changes overtime. In this application, the static transformation is the transformation of the sensors frames to the base of the robot. In is AIV the base of the robot is called “base_link”. Then the frame of other sensors is referred to this frame. Each sensor has their own frame. Therefore, the data refers to the sensors can be transformed to another frame. The dynamic transformation on the other hand is the transform between the “base_link” to the “odometry” frame as an example. As the AIV move, the transformation between these 2 frames is updated.

The tf_broadcaster node written in C++ is responsible for publishing the static transformations. While the dynamic transformation is taken care by another package that will be discussed later in this thesis report.

**Figure 3.28**

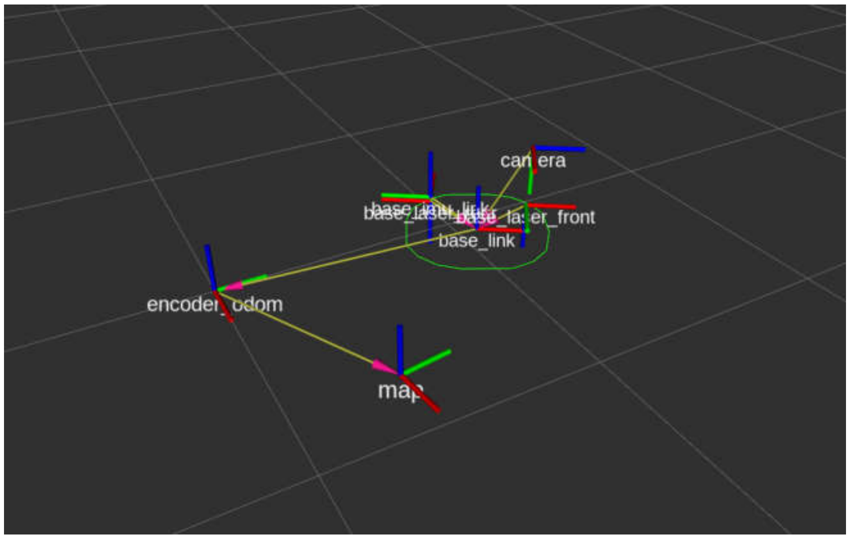
*Static TF Broadcaster Node*



```
#include <ros/ros.h>
#include <tf/transform_broadcaster.h>

int pi = 3.14159;
int main(int argc, char** argv){
  ros::init(argc, argv, "robot_tf_publisher");
  ros::NodeHandle n;

  ros::Rate r(10);

  tf::TransformBroadcaster broadcaster;

  while(n.ok()){
    broadcaster.sendTransform(tf::StampedTransform(
        tf::Transform(tf::Quaternion(1, 0, 0, 0), tf::Vector3(0.195, 0, 0.123)),
        ros::Time::now(),"base_link", "base_laser_front"));
    broadcaster.sendTransform(
      tf::StampedTransform(
        tf::Transform(tf::Quaternion(0, -1 , 0, 0), tf::Vector3(-0.195, 0, 0.123)),
        ros::Time::now(),"base_link", "base_laser_rear"));
    broadcaster.sendTransform(
        tf::StampedTransform(
        tf::Transform(tf::Quaternion(-0.5, 0.5, -0.5, 0.5), tf::Vector3(0.205, 0, 0.38)),
        ros::Time::now(),"base_link", "camera"));
    broadcaster.sendTransform(
        tf::StampedTransform(
        tf::Transform(tf::Quaternion(0, 0 , 0.707, 0.707), tf::Vector3(-0.195, 0, 0.143)),
        ros::Time::now(),"base_link", "base_imu_link"));
    r.sleep();
  }
}
```

**Figure 3.29**

*TF Visualization on Rviz*

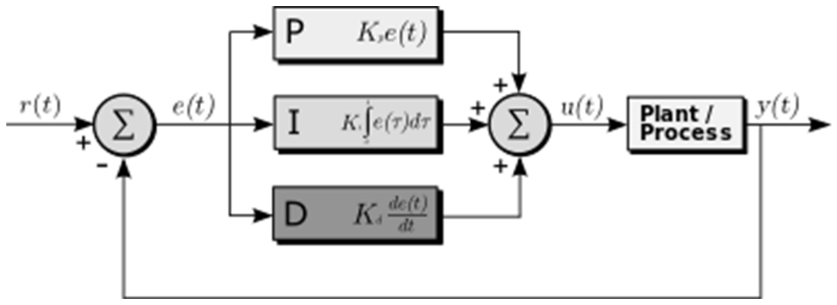


## 3.5 Motion Control Software Implementation

The motor controller is used for controlling the speed of the motor to exact command speed. Proportional Integral Derivative (PID) is a motor controller which is applied in this function.

***Figure 3.30***

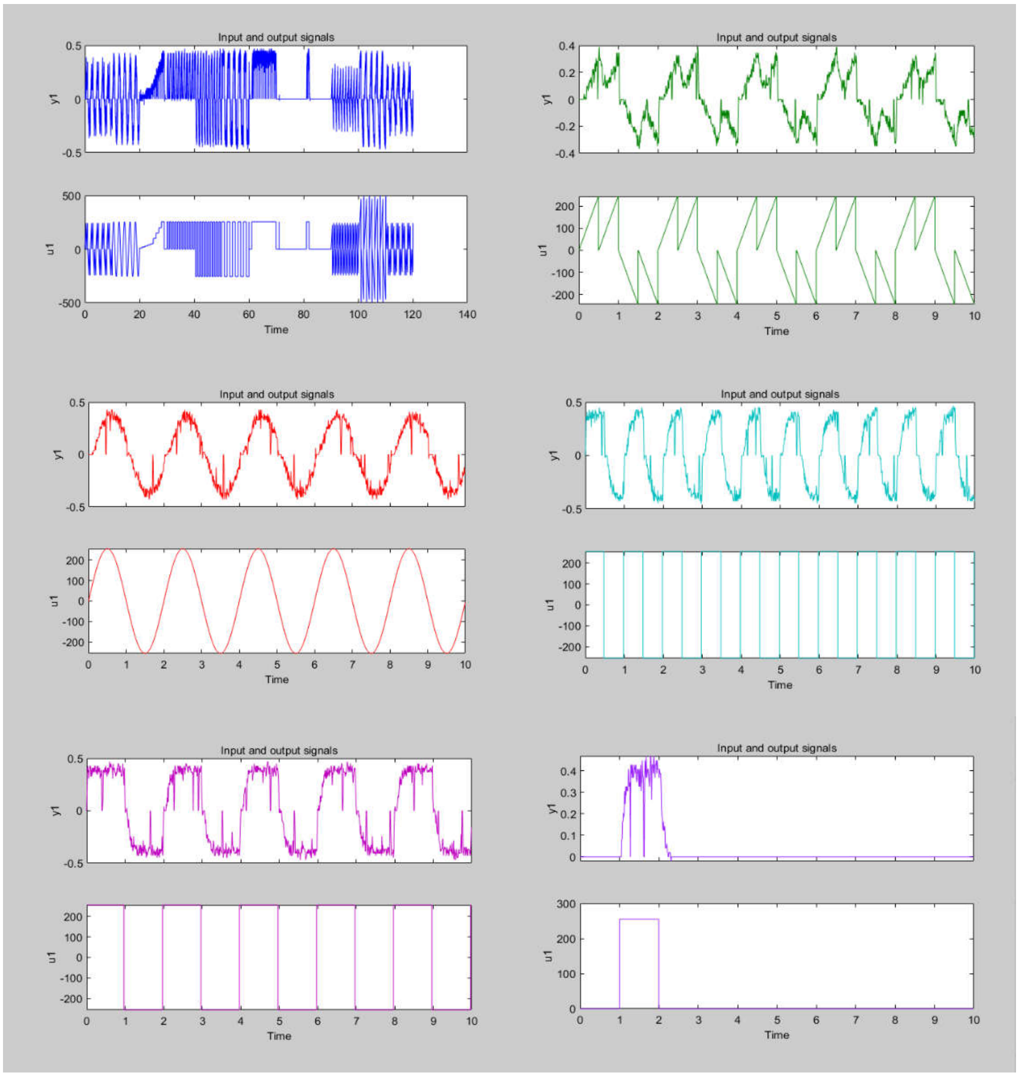
*PID Controller*



The transfer function of the plant which is the motor and gear reduction assembly from micro controller PWM output to wheel velocity is required. The transfer function can be defined using system identification method. The system identification method can be used to determine the transfer function of an unknown system which is considered as a black box. The input is feed into the system then the output is measured. From the relation of the input to the output, the transfer function can be estimated. In our approach, the time series PWM signal is feed to the motor driver and the time series output wheel velocity is logged to the log file. The PWM signal consists of square signal, step signal, sinusoidal signal, saw tooth signal with different frequency are used the output are logged into their respectable signal. Then MATLAB is utilized to estimate the transferring function by using system identification toolbox.

**Figure 3.31**

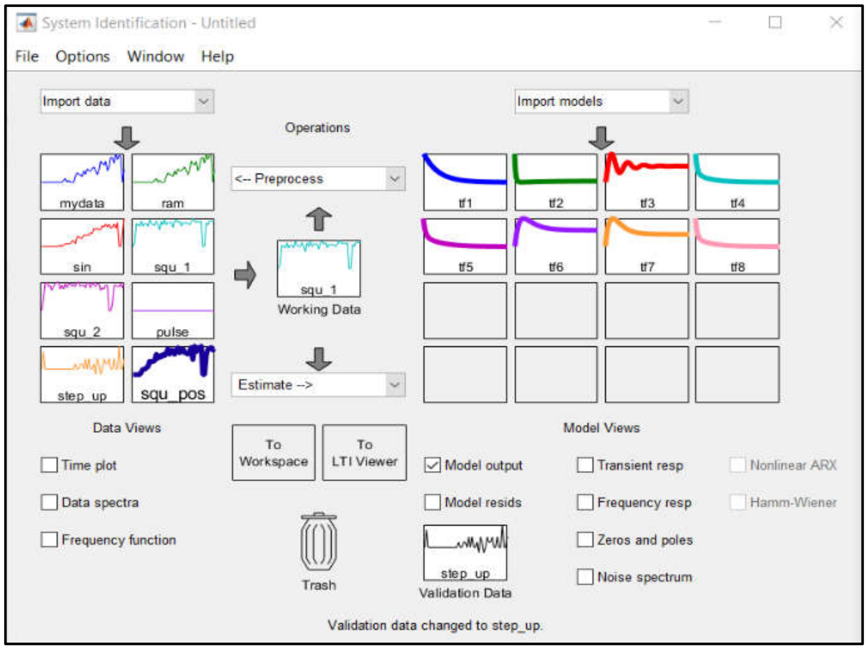
*Input Signal U1 and Output Signal Y1 Data for System Identification*



In MATLAB System Identification Toolbox, the sets of input and output signal are imported. Each data set is used to estimate a transfer function as can be seen in figure below.

**Figure 3.32**

*MATLAB System Identification Toolbox Window*



Validation data changed to step_up.

After doing testing and validating it can be seen that the transfer function that best fit the validating date is t1.

**Figure 3.33**

*Transfer Function Validation*

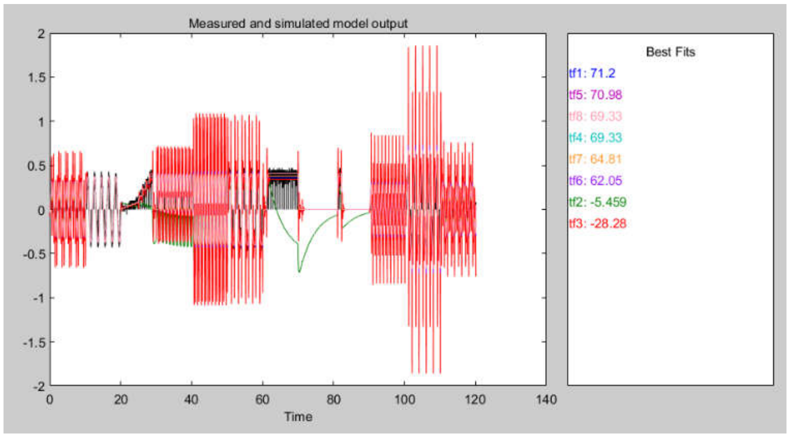
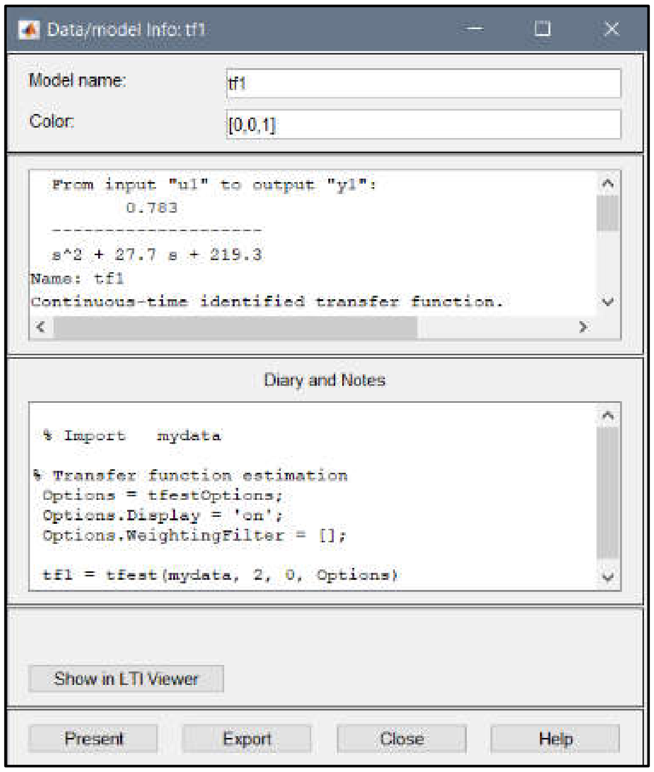


**Figure 3.34**

*Transfer Function TF1*



Model name: tf1

Color: [0,0,1]

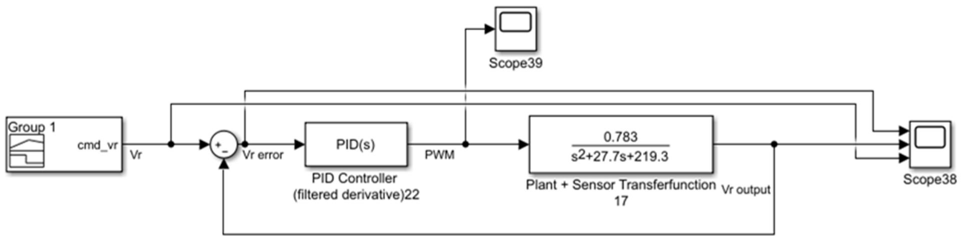
```
 From input "u1" to output "y1":
        0.783
 -------------------
 s^2 + 27.7 s + 219.3
Name: tf1
Continuous-time identified transfer function.
```

```
 % Import   mydata

% Transfer function estimation
 Options = tfestOptions;
 Options.Display = 'on';
 Options.WeightingFilter = [];

 tf1 = tfest(mydata, 2, 0, Options)
```

After the transfer function is obtained the PID controller can be design. MATLAB Simulink is used to design and simulate result. Trial and error approach is used to tune the controller while focusing on minimizing overshot and control effort and considerable response time. The PID Transfer function based PID tuner is use for this job.

**Figure 3.35**

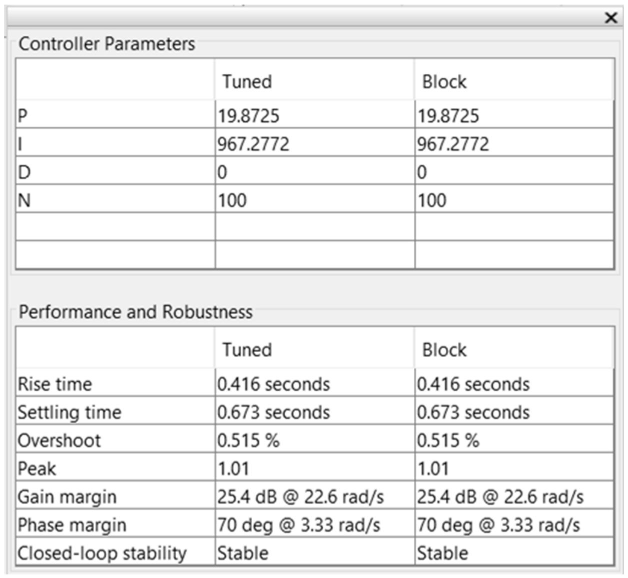
*Wheel Speed PID Controller Loop*



The **Figure 3.36** shows the P, I, and D parameter gain after tuning. The loop has 0.515% overshot and settle time of 0.673 second.

**Figure 3.36**

*Parameter form PID Tuning*



Controller Parameters

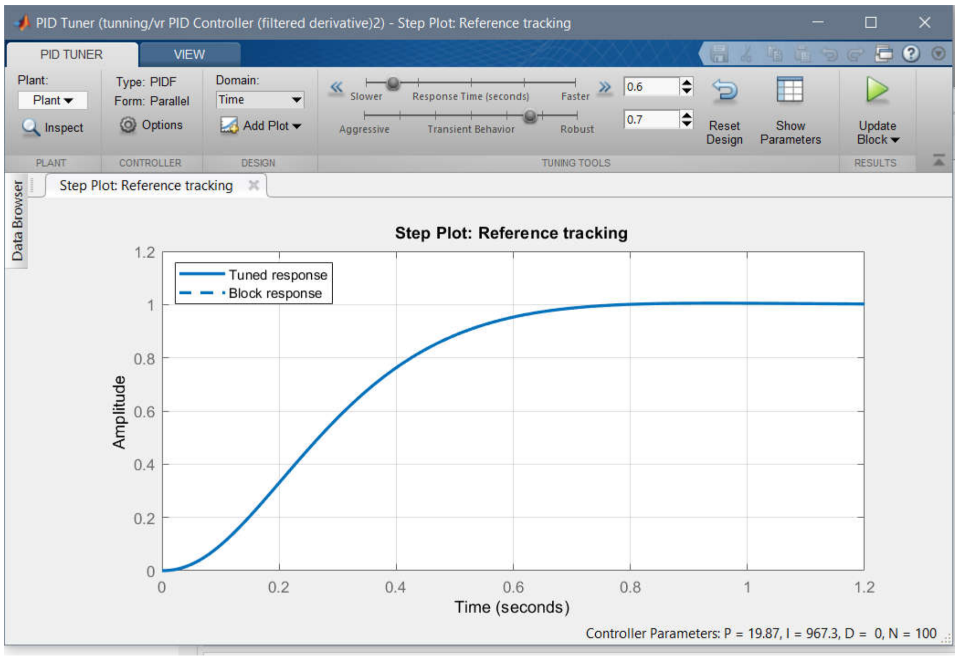
| | Tuned | Block |
|---|---|---|
| P | 19.8725 | 19.8725 |
| I | 967.2772 | 967.2772 |
| D | 0 | 0 |
| N | 100 | 100 |
| | | |
| | | |

Performance and Robustness

| | Tuned | Block |
|---|---|---|
| Rise time | 0.416 seconds | 0.416 seconds |
| Settling time | 0.673 seconds | 0.673 seconds |
| Overshoot | 0.515 % | 0.515 % |
| Peak | 1.01 | 1.01 |
| Gain margin | 25.4 dB @ 22.6 rad/s | 25.4 dB @ 22.6 rad/s |
| Phase margin | 70 deg @ 3.33 rad/s | 70 deg @ 3.33 rad/s |
| Closed-loop stability | Stable | Stable |

**Figure 3.37**

*Vr Step Response*



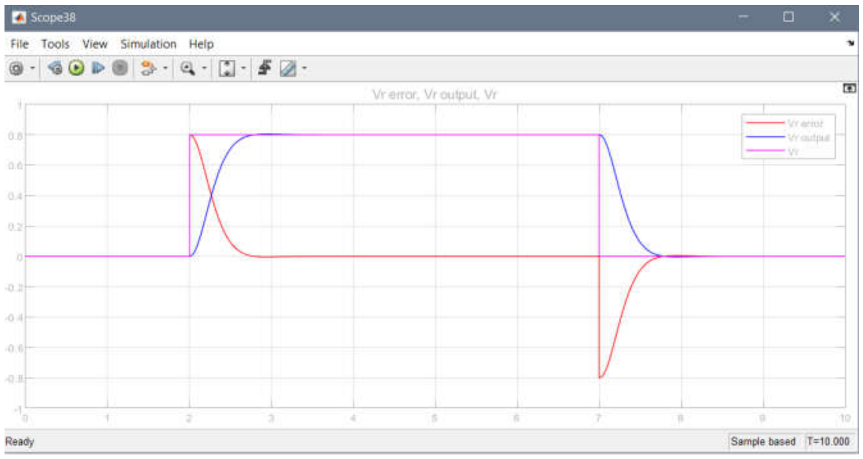
PID Tuner (tunning/vr PID Controller (filtered derivative)2) - Step Plot: Reference tracking

Type: PIDF
Form: Parallel
Domain: Time
Response Time (seconds): 0.6
Transient Behavior: 0.7

Controller Parameters: P = 19.87, I = 967.3, D = 0, N = 100

**Figure 3.38**

*Simulation of Control Loop*



The PID controller transfer function of the loop can be calculated using the formula below:

$$PID_{(S)} = P + I\left(\frac{1}{s}\right) + D\frac{N}{1 + N\frac{1}{s}}$$

By plug in the parameter obtain from tuning.

$$P = 19.8725$$

$$I = 967.272$$

$$D = 0$$

Then

$$PID_{(s)} = \frac{19.87\,s + 967.3}{s}$$

Then the transfer function can be discretize using Zero Order Hold (ZOH) at time interval of 0.01 second corresponds to 100 Hz looping

$$PID_{(z)} = \frac{19.87\,z - 10.2}{z - 1}$$

The difference equation can be obtained:

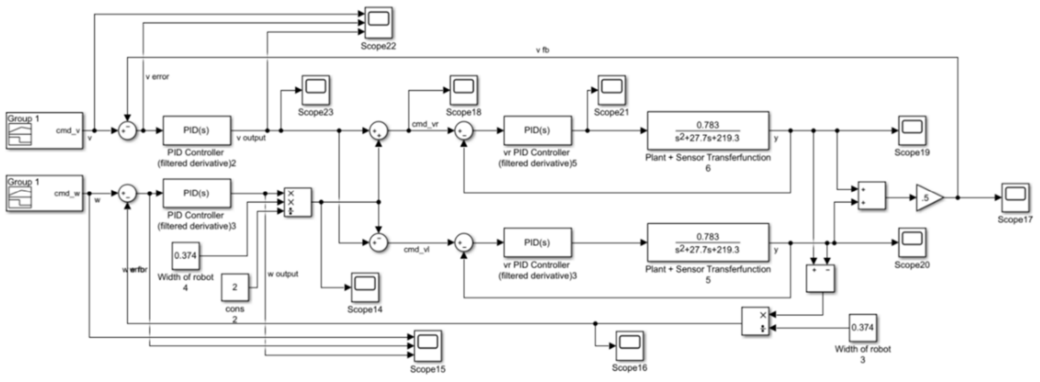
$$Y_{[k]} = 19.87\, U_{[k]} - 10.2\, U_{[k-1]} + Y_{[k-1]}$$

The obtained difference equation can then be used to implement in microcontroller.

Then 2 more PID control loops need to be designed to control AIV velocity in x direction (cmd_v) and the angular velocity in z direction (cmd_w). The same approach to above is implemented for these controllers.

**Figure 3.39**

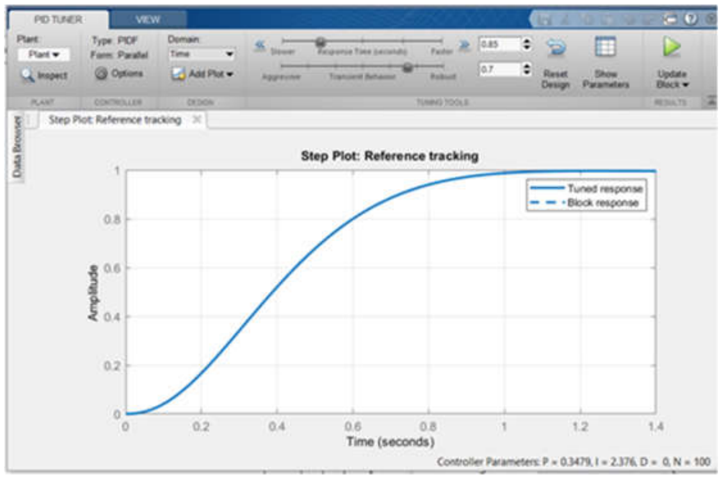
*AIV Motion PID Controller Loop*



The controller loop for linear velocity in x direction (cmd_v) can be turned in PID Transfer function based PID tuner. The figures bellow shows the output parameter and performance of controller.

**Figure 3.40**

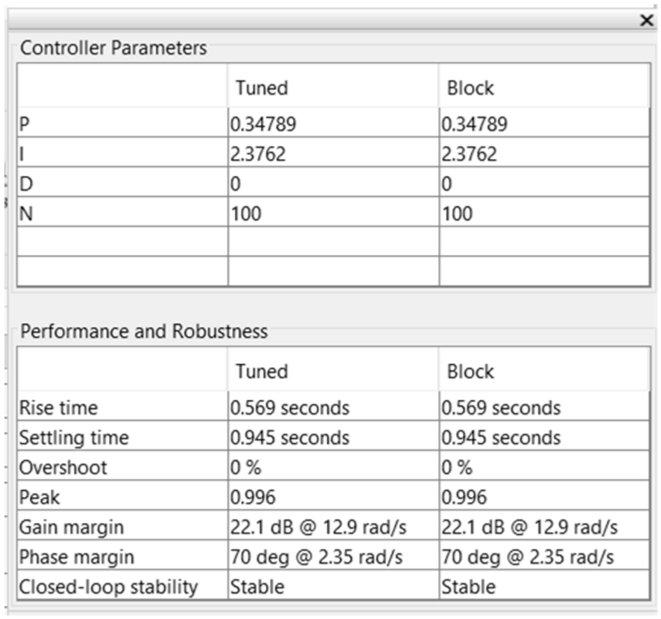
*Cmd_v Step Response*



Controller Parameters: P = 0.3479, I = 2.376, D = 0, N = 100

**Figure 3.41**

*Cmd_v Controller Parameter and Performance*



Controller Parameters

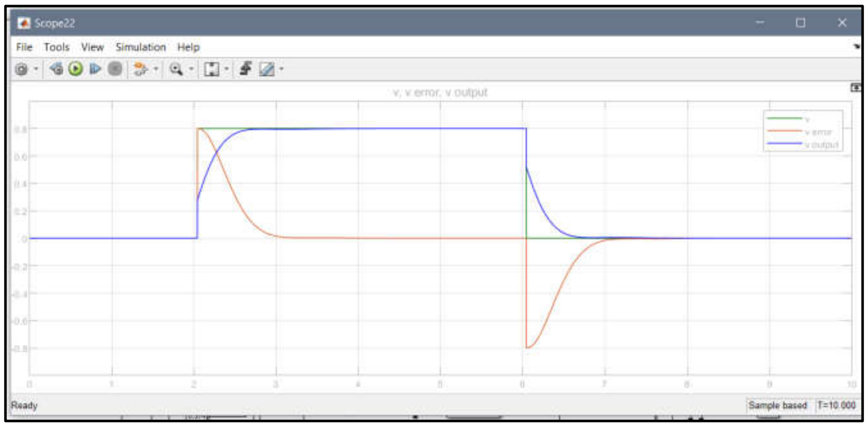
| | Tuned | Block |
|---|---|---|
| P | 0.34789 | 0.34789 |
| I | 2.3762 | 2.3762 |
| D | 0 | 0 |
| N | 100 | 100 |
| | | |
| | | |

Performance and Robustness

| | Tuned | Block |
|---|---|---|
| Rise time | 0.569 seconds | 0.569 seconds |
| Settling time | 0.945 seconds | 0.945 seconds |
| Overshoot | 0 % | 0 % |
| Peak | 0.996 | 0.996 |
| Gain margin | 22.1 dB @ 12.9 rad/s | 22.1 dB @ 12.9 rad/s |
| Phase margin | 70 deg @ 2.35 rad/s | 70 deg @ 2.35 rad/s |
| Closed-loop stability | Stable | Stable |

**Figure 3.42**

*Simulation of Cmd_v Control Loop*



The PID controller transfer function of the loop can be calculated using the formula below:

$$PID_{(S)} = P + I\left(\frac{1}{s}\right) + D\frac{N}{1 + N\frac{1}{s}}$$

By plug in the parameter obtain from tuning.

$$P = 0.3478$$

$$I = 2.362$$

$$D = 0$$

Then

$$PID_{(s)} = \frac{0.3479\, s + 2.376}{s}$$

Then the transfer function can be discretize using Zero Order Hold (ZOH) at time interval of 0.01 second corresponds to 100 Hz looping

$$PID_{(Z)} = \frac{0.3479\, z - 0.3241}{z - 1}$$

The difference equation can be obtained and implemented in microcontroller

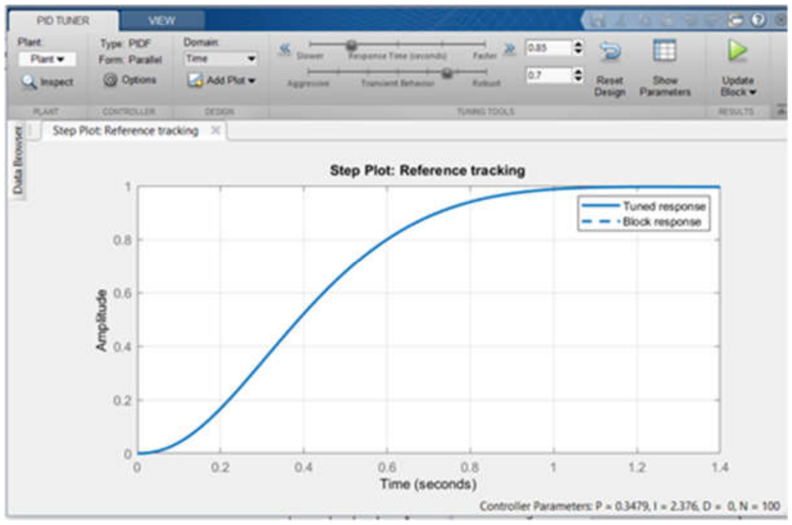
$$Y_{[k]} = 0.3479\, U_{[k]} - 0.3241\, U_{[k-1]} + Y_{[k-1]}$$

The obtained difference equation can then be used to implement in microcontroller.

Likewise, the controller loop for angular velocity in z direction (cmd_w) can be turned in PID Transfer function based PID tuner. The figures bellow shows the output parameter and performance of controller.

**Figure 3.43**

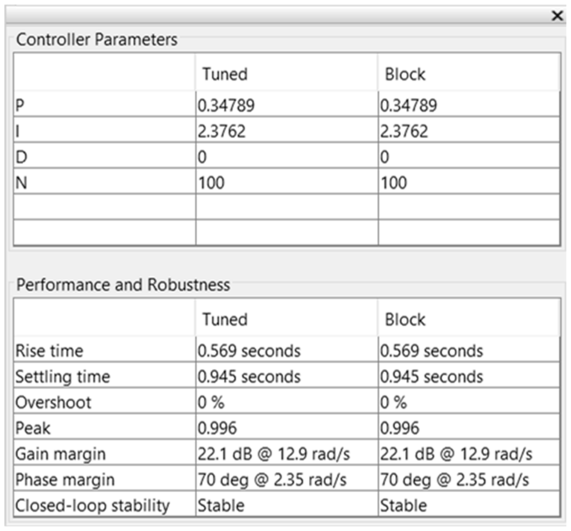
*Cmd_w Step Response*



Controller Parameters: P = 0.3479, I = 2.376, D = 0, N = 100

**Figure 3.44**

*Cmd_w Controller Parameter and Performance*



Controller Parameters

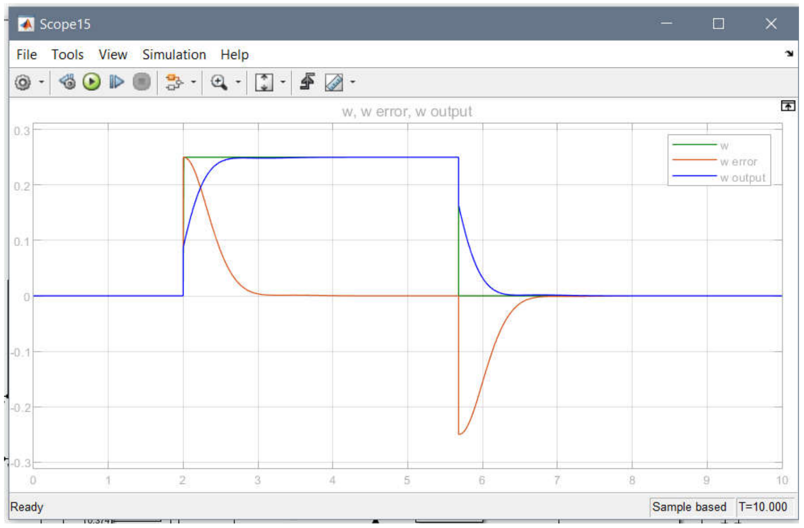
| | Tuned | Block |
|---|---|---|
| P | 0.34789 | 0.34789 |
| I | 2.3762 | 2.3762 |
| D | 0 | 0 |
| N | 100 | 100 |
| | | |
| | | |

Performance and Robustness

| | Tuned | Block |
|---|---|---|
| Rise time | 0.569 seconds | 0.569 seconds |
| Settling time | 0.945 seconds | 0.945 seconds |
| Overshoot | 0 % | 0 % |
| Peak | 0.996 | 0.996 |
| Gain margin | 22.1 dB @ 12.9 rad/s | 22.1 dB @ 12.9 rad/s |
| Phase margin | 70 deg @ 2.35 rad/s | 70 deg @ 2.35 rad/s |
| Closed-loop stability | Stable | Stable |

**Figure 3.45**

*Simulation of Cmd_w Control Loop*



By plug in the parameter obtain from tuning,

$$P = 0.3479$$

$$I = 0.3241$$

$$D = 0$$

Then

$$PID_{(s)} = \frac{0.3479\, s + 2.376}{s}$$

Then the transfer function can be discretize using Zero Order Hold (ZOH) at time interval of 0.01 second corresponds to 100 Hz looping.

$$PID_{(Z)} = \frac{0.3479\, z - 0.3241}{z - 1}$$

The difference equation can be obtained and implemented in microcontroller

$$Y_{[k]} = 0.3479\, U_{[k]} - 0.3241\, U_{[k-1]} + Y_{[k-1]}$$

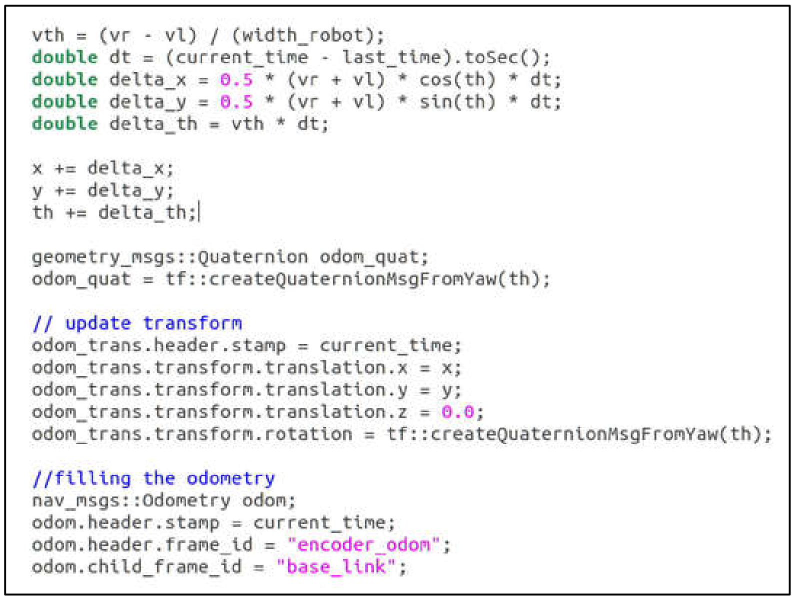
## 3.6 Odometry

In robotic, Odometry is defined some legged or wheeled to estimate the position related from starting to ending location. The sensitive point in this method is the errors of the integration of velocity measurement over time that is used to allocate and estimate the location. The effectively used of Odometry in most cases include rapid and accurate data collection, instrument calibration, and the processing.

Encoder_dometry node written in C++ is responsible for this task. It keeps transformation between "base_link" to "odom" frame. The node subscribes to "/right_wheel1_velocity" and "/left_wheel2_velocity" which are the wheels velocity of the AIV's right and left wheel respectively. The publisher is responsible for publishing these topics is serial_node_teensy which reads, calculates, and publishes the topics once in 0.02 second or 50Hz. Then the data is calculated and publishes the odometry data in topic "/odom"

**Figure 3.46**

*Odometry Calculation and Update*



```
vth = (vr - vl) / (width_robot);
double dt = (current_time - last_time).toSec();
double delta_x = 0.5 * (vr + vl) * cos(th) * dt;
double delta_y = 0.5 * (vr + vl) * sin(th) * dt;
double delta_th = vth * dt;

x += delta_x;
y += delta_y;
th += delta_th;

geometry_msgs::Quaternion odom_quat;
odom_quat = tf::createQuaternionMsgFromYaw(th);

// update transform
odom_trans.header.stamp = current_time;
odom_trans.transform.translation.x = x;
odom_trans.transform.translation.y = y;
odom_trans.transform.translation.z = 0.0;
odom_trans.transform.rotation = tf::createQuaternionMsgFromYaw(th);

//filling the odometry
nav_msgs::Odometry odom;
odom.header.stamp = current_time;
odom.header.frame_id = "encoder_odom";
odom.child_frame_id = "base_link";
```

## 3.7 Mapping

SLAM Gmapping is implemented for map generating. ROS node called slam_gmapping. 2-D occupancy grid map, like a building floorplan, can be created using slam_gmapping from the laser and pose data collected by AIV.

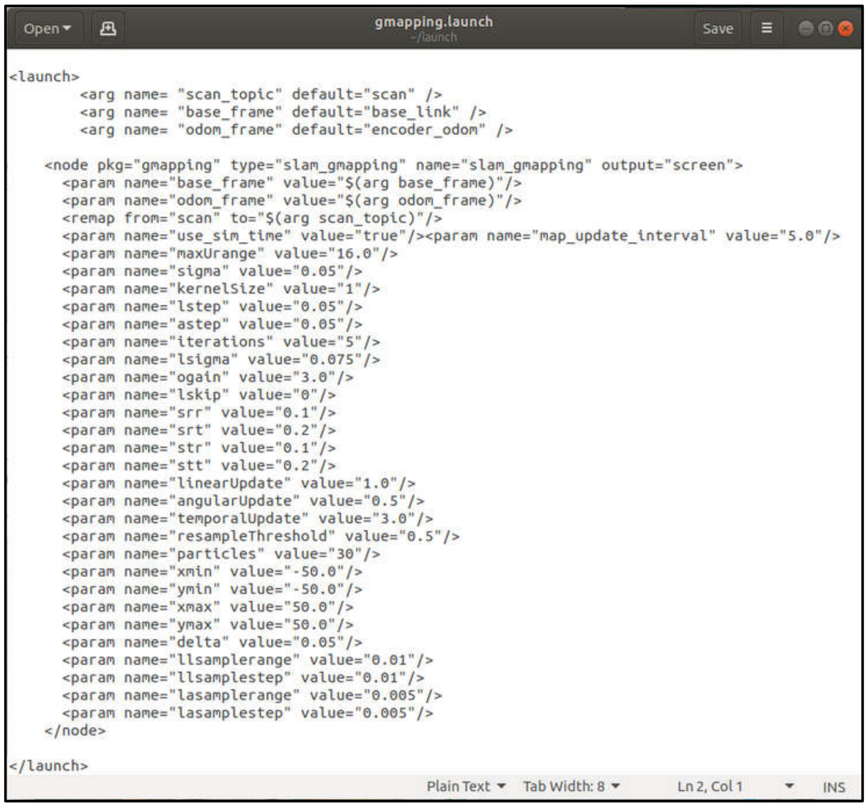
To do the mapping, it requires 2 topics:

- /scan: Laser scan data
- /odom: Odometry from wheel encoder.

The important parameter can be set in the launch file, such as the topic name of the Laser scan data, base frame, the odom frame and other parameters. Below figure show the launch file utilizing the gmapping package and the configured parameter for map scanning.

**Figure 3.47**

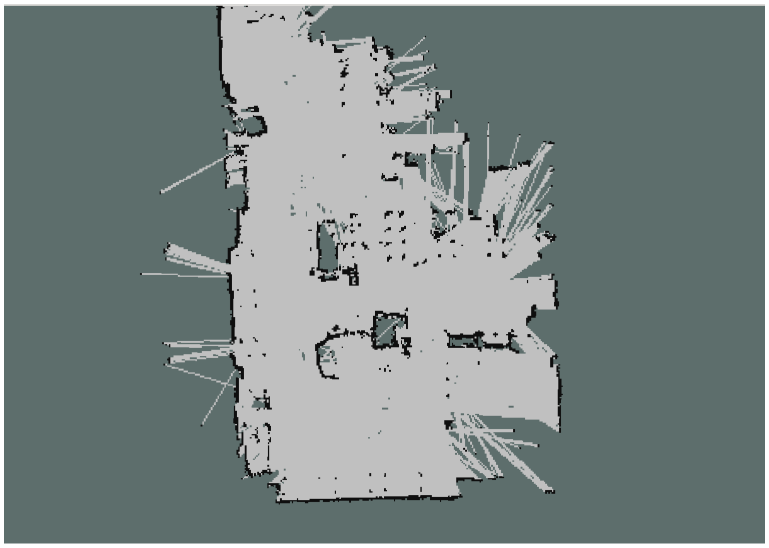
*Gmapping Launch File*



```
<launch>
        <arg name= "scan_topic" default="scan" />
        <arg name= "base_frame" default="base_link" />
        <arg name= "odom_frame" default="encoder_odom" />

    <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">
      <param name="base_frame" value="$(arg base_frame)"/>
      <param name="odom_frame" value="$(arg odom_frame)"/>
      <remap from="scan" to="$(arg scan_topic)"/>
      <param name="use_sim_time" value="true"/><param name="map_update_interval" value="5.0"/>
      <param name="maxUrange" value="16.0"/>
      <param name="sigma" value="0.05"/>
      <param name="kernelSize" value="1"/>
      <param name="lstep" value="0.05"/>
      <param name="astep" value="0.05"/>
      <param name="iterations" value="5"/>
      <param name="lsigma" value="0.075"/>
      <param name="ogain" value="3.0"/>
      <param name="lskip" value="0"/>
      <param name="srr" value="0.1"/>
      <param name="srt" value="0.2"/>
      <param name="str" value="0.1"/>
      <param name="stt" value="0.2"/>
      <param name="linearUpdate" value="1.0"/>
      <param name="angularUpdate" value="0.5"/>
      <param name="temporalUpdate" value="3.0"/>
      <param name="resampleThreshold" value="0.5"/>
      <param name="particles" value="30"/>
      <param name="xmin" value="-50.0"/>
      <param name="ymin" value="-50.0"/>
      <param name="xmax" value="50.0"/>
      <param name="ymax" value="50.0"/>
      <param name="delta" value="0.05"/>
      <param name="llsamplerange" value="0.01"/>
      <param name="llsamplestep" value="0.01"/>
      <param name="lasamplerange" value="0.005"/>
      <param name="lasamplestep" value="0.005"/>
    </node>

</launch>
```

**Figure 3.48**

*Grid Map Obtained from Gmapping*



## 3.8 Localization and Navigation

Robot navigation means the ability define its own position in the reference frame. Robot navigation be able to plan the pathway to its goal location. Robot and other mobility device need to representation in order to navigate in the environment.

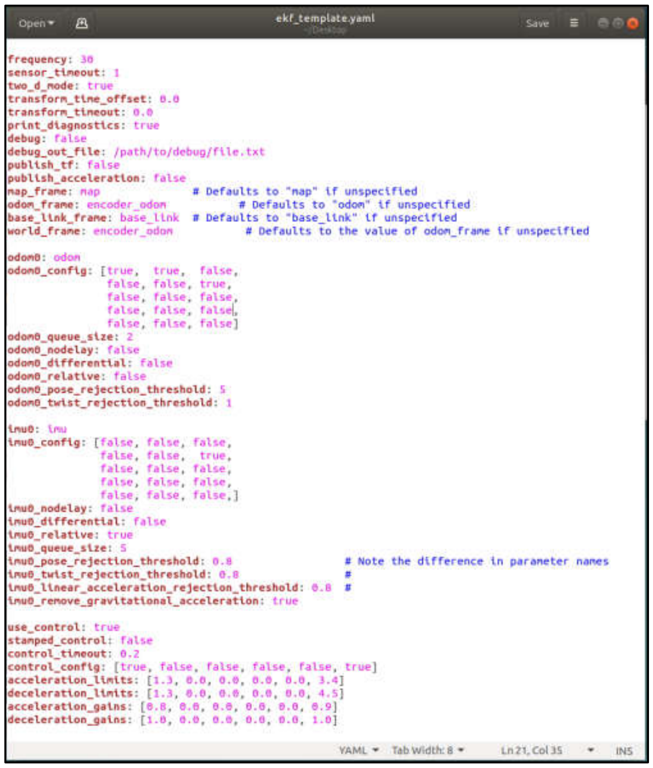
### *3.8.1 Sensors Fusion*

In this AIV, the sensor fusion is implemented using robot localization package. There are 3 sensors be fused such as odometry (/odom), IMU sensor, and command velocity (/cmd_vel). The package results the fused odometry (/odom_fusioned).

The figure below shows the implemented configuration of each sensor. Odometry sensor position x, y, yaw, IMU yaw, cmd_vel x and v_yaw are used.

**Figure 3.49**

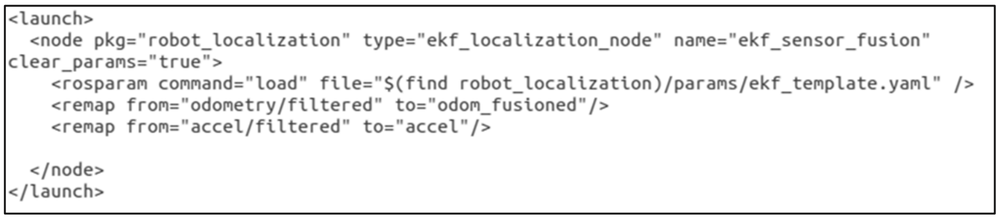
*EKF Parameter Setup*



```yaml
frequency: 30
sensor_timeout: 1
two_d_mode: true
transform_time_offset: 0.0
transform_timeout: 0.0
print_diagnostics: true
debug: false
debug_out_file: /path/to/debug/file.txt
publish_tf: false
publish_acceleration: false
map_frame: map              # Defaults to "map" if unspecified
odom_frame: encoder_odom            # Defaults to "odom" if unspecified
base_link_frame: base_link  # Defaults to "base_link" if unspecified
world_frame: encoder_odom            # Defaults to the value of odom_frame if unspecified

odom0: odom
odom0_config: [true,  true,  false,
               false, false, true,
               false, false, false,
               false, false, false,
               false, false, false]
odom0_queue_size: 2
odom0_nodelay: false
odom0_differential: false
odom0_relative: false
odom0_pose_rejection_threshold: 5
odom0_twist_rejection_threshold: 1

imu0: imu
imu0_config: [false, false, false,
              false, false,  true,
              false, false, false,
              false, false, false,
              false, false, false,]
imu0_nodelay: false
imu0_differential: false
imu0_relative: true
imu0_queue_size: 5
imu0_pose_rejection_threshold: 0.8                 # Note the difference in parameter names
imu0_twist_rejection_threshold: 0.8                #
imu0_linear_acceleration_rejection_threshold: 0.8  #
imu0_remove_gravitational_acceleration: true

use_control: true
stamped_control: false
control_timeout: 0.2
control_config: [true, false, false, false, false, true]
acceleration_limits: [1.3, 0.0, 0.0, 0.0, 0.0, 3.4]
deceleration_limits: [1.3, 0.0, 0.0, 0.0, 0.0, 4.5]
acceleration_gains: [0.8, 0.0, 0.0, 0.0, 0.0, 0.9]
deceleration_gains: [1.0, 0.0, 0.0, 0.0, 0.0, 1.0]
```

**Figure 3.50**

*Robot Localization Launch File*



```
<launch>
  <node pkg="robot_localization" type="ekf_localization_node" name="ekf_sensor_fusion"
clear_params="true">
    <rosparam command="load" file="$(find robot_localization)/params/ekf_template.yaml" />
    <remap from="odometry/filtered" to="odom_fusioned"/>
    <remap from="accel/filtered" to="accel"/>

  </node>
</launch>
```

### *3.8.2 Particle Filter (AMCL)*

AMCL algorithm is used for robot localization in the given map. The AMCL algorithm is a probabilistic localization technique that uses a particle filter to track the pose of a robot in a known map. The odometry keeps track of the robot position relative to the odometry map. However, the odometry data is drifting overtime, the accumulated error makes it unusable for a long period of time. Therefore, the odometry data is only use in a short time to give the idea where the robot was and where it should be after a period of time continuously. The laser scan data is used in combination with the odometry data is used to compare to the existing map data to estimate the robot’s position and update the translation from odom frame to map frame. Therefore, the translation between base_link frame to odom frame is a continuous translation while the translation between base_link frame to odom frame is a discrete translation.

**Figure 3.51**

*AMCL Map Localization*

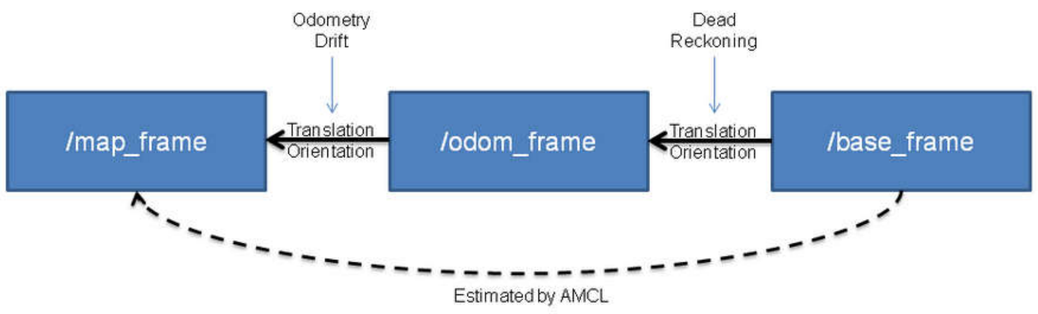
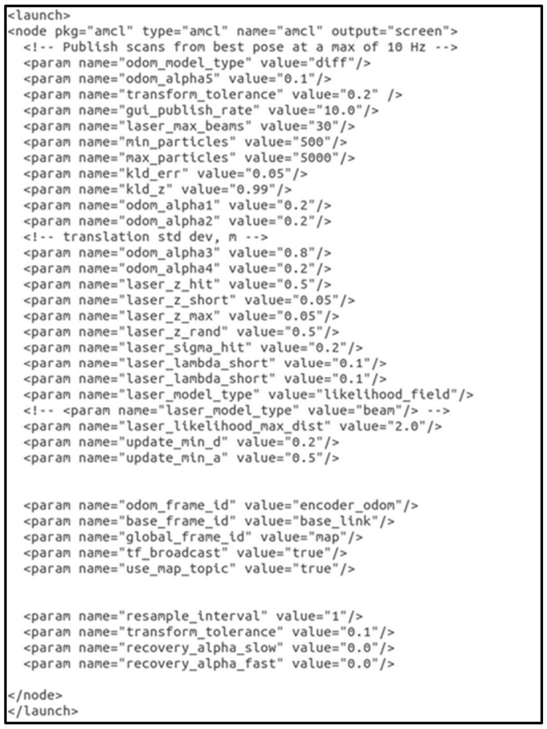


**Figure 3.52**

*Launch Files for AMCL Node and Configured Parameter*



```
<launch>
<node pkg="amcl" type="amcl" name="amcl" output="screen">
  <!-- Publish scans from best pose at a max of 10 Hz -->
  <param name="odom_model_type" value="diff"/>
  <param name="odom_alpha5" value="0.1"/>
  <param name="transform_tolerance" value="0.2" />
  <param name="gui_publish_rate" value="10.0"/>
  <param name="laser_max_beams" value="30"/>
  <param name="min_particles" value="500"/>
  <param name="max_particles" value="5000"/>
  <param name="kld_err" value="0.05"/>
  <param name="kld_z" value="0.99"/>
  <param name="odom_alpha1" value="0.2"/>
  <param name="odom_alpha2" value="0.2"/>
  <!-- translation std dev, m -->
  <param name="odom_alpha3" value="0.8"/>
  <param name="odom_alpha4" value="0.2"/>
  <param name="laser_z_hit" value="0.5"/>
  <param name="laser_z_short" value="0.05"/>
  <param name="laser_z_max" value="0.05"/>
  <param name="laser_z_rand" value="0.5"/>
  <param name="laser_sigma_hit" value="0.2"/>
  <param name="laser_lambda_short" value="0.1"/>
  <param name="laser_lambda_short" value="0.1"/>
  <param name="laser_model_type" value="likelihood_field"/>
  <!-- <param name="laser_model_type" value="beam"/> -->
  <param name="laser_likelihood_max_dist" value="2.0"/>
  <param name="update_min_d" value="0.2"/>
  <param name="update_min_a" value="0.5"/>


  <param name="odom_frame_id" value="encoder_odom"/>
  <param name="base_frame_id" value="base_link"/>
  <param name="global_frame_id" value="map"/>
  <param name="tf_broadcast" value="true"/>
  <param name="use_map_topic" value="true"/>


  <param name="resample_interval" value="1"/>
  <param name="transform_tolerance" value="0.1"/>
  <param name="recovery_alpha_slow" value="0.0"/>
  <param name="recovery_alpha_fast" value="0.0"/>

</node>
</launch>
```

### *3.8.3 Navigation Stack*

This package in ROS is helpful to navigate using SLAM. The path generating with navigation function is also handled by this package.

The list below indicates the parameter to be include on move_base launch file.

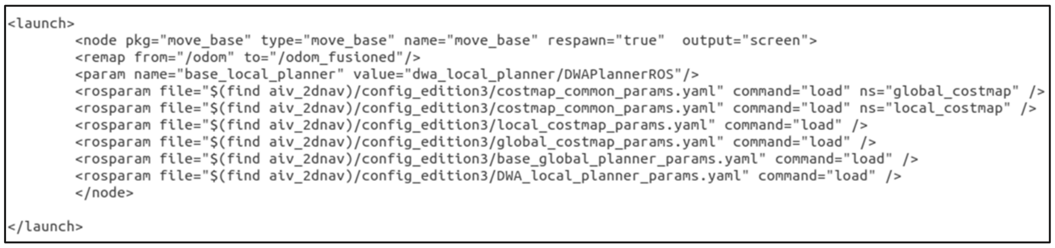
- Costmap common param
- Global costmap params
- Local costmap params
- Move base params
- Global planner params
- Local planner params

Move_base subscribe to the topic bellow,

- Goal pose
- map
- scan data

**Figure 3.53**

*Move_base Launch File*



```
<launch>
        <node pkg="move_base" type="move_base" name="move_base" respawn="true"  output="screen">
        <remap from="/odom" to="/odom_fusioned"/>
        <param name="base_local_planner" value="dwa_local_planner/DWAPlannerROS"/>
        <rosparam file="$(find aiv_2dnav)/config_edition3/costmap_common_params.yaml" command="load" ns="global_costmap" />
        <rosparam file="$(find aiv_2dnav)/config_edition3/costmap_common_params.yaml" command="load" ns="local_costmap" />
        <rosparam file="$(find aiv_2dnav)/config_edition3/local_costmap_params.yaml" command="load" />
        <rosparam file="$(find aiv_2dnav)/config_edition3/global_costmap_params.yaml" command="load" />
        <rosparam file="$(find aiv_2dnav)/config_edition3/base_global_planner_params.yaml" command="load" />
        <rosparam file="$(find aiv_2dnav)/config_edition3/DWA_local_planner_params.yaml" command="load" />
        </node>

</launch>
```

### *3.8.4 Cost Map*

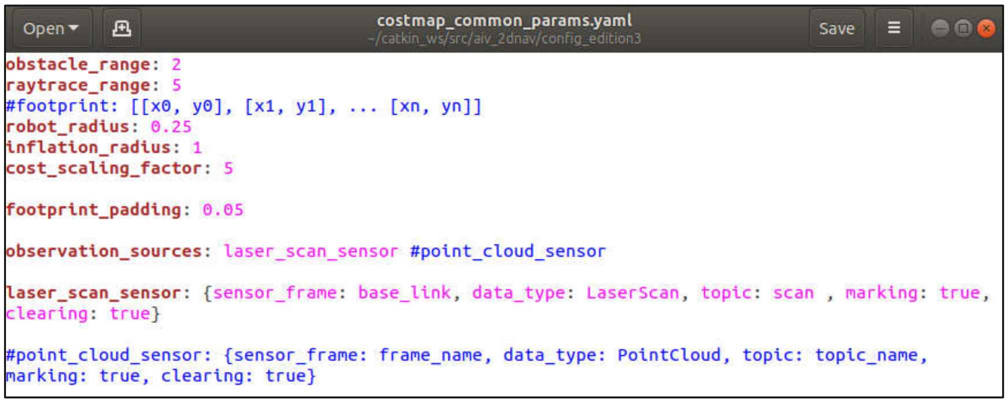
A cost map is a fundamental concept in autonomous robotics. It represents the cost (difficulty) of traversing different areas of the map. In the case of a ground robot working on rough ground the cost map could be a 2D map with lower values where the ground it flat and higher where the ground is rough/sloping. The values held in a cost map are usually abstract and do not directly represent any measurement of the world, they are simply used to guide a route planning algorithm to find efficient and safe routes across the ground.

The grid occupancy is defined as bellow:

- Black as occupied
- White as free
- Gray as unknown

**Figure 3.54**

*Cost Map Common Parameter*


Open
costmap_common_params.yaml
~/catkin_ws/src/aiv_2dnav/config_edition3
Save

```
obstacle_range: 2
raytrace_range: 5
#footprint: [[x0, y0], [x1, y1], ... [xn, yn]]
robot_radius: 0.25
inflation_radius: 1
cost_scaling_factor: 5

footprint_padding: 0.05

observation_sources: laser_scan_sensor #point_cloud_sensor

laser_scan_sensor: {sensor_frame: base_link, data_type: LaserScan, topic: scan , marking: true,
clearing: true}

#point_cloud_sensor: {sensor_frame: frame_name, data_type: PointCloud, topic: topic_name,
marking: true, clearing: true}
```

***3.8.4.1 Map Inflation.*** Inflation is the process of generate cost values out from occupied cells that decrease with distance.

Cost map defining step:

- Lethal
- Inscribed
- Possibly circumscribed
- Free space
- Unknown

**Figure 3.55**

*Cost Map Defining Step*

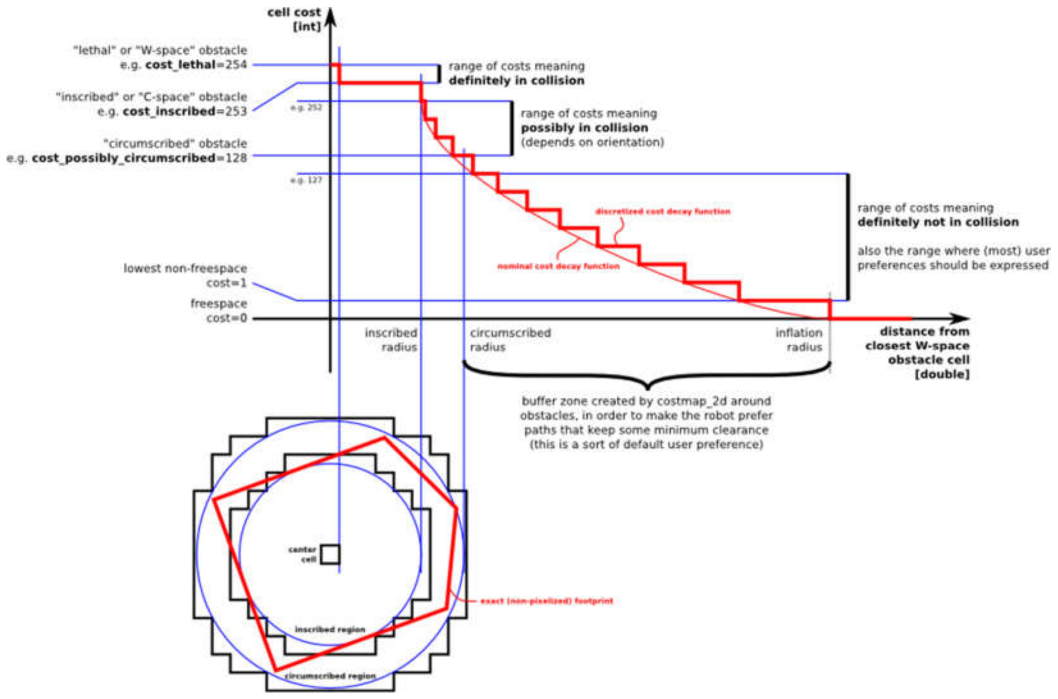
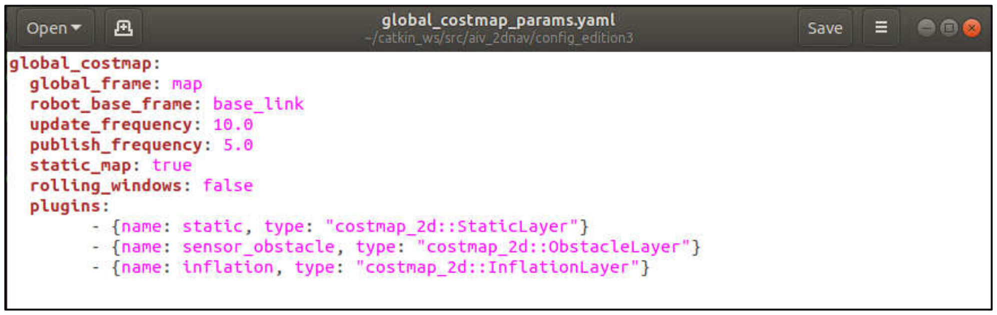

cell cost [int]
"lethal" or "W-space" obstacle e.g. cost_lethal=254
"inscribed" or "C-space" obstacle e.g. cost_inscribed=253
"circumscribed" obstacle e.g. cost_possibly_circumscribed=128
e.g. 252
e.g. 127
lowest non-freespace cost=1
freespace cost=0
range of costs meaning definitely in collision
range of costs meaning possibly in collision (depends on orientation)
range of costs meaning definitely not in collision
also the range where (most) user preferences should be expressed
discretized cost decay function
nominal cost decay function
inscribed radius
circumscribed radius
inflation radius
distance from closest W-space obstacle cell [double]
buffer zone created by costmap_2d around obstacles, in order to make the robot prefer paths that keep some minimum clearance (this is a sort of default user preference)
center cell
inscribed region
circumscribed region
exact (non-pixelized) footprint

**Figure 3.56**

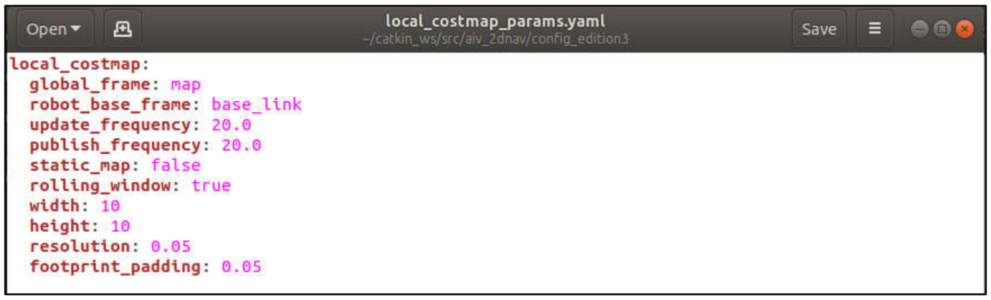
*Global Cost Map Parameter*


Open
global_costmap_params.yaml
~/catkin_ws/src/aiv_2dnav/config_edition3
Save

```
global_costmap:
  global_frame: map
  robot_base_frame: base_link
  update_frequency: 10.0
  publish_frequency: 5.0
  static_map: true
  rolling_windows: false
  plugins:
        - {name: static, type: "costmap_2d::StaticLayer"}
        - {name: sensor_obstacle, type: "costmap_2d::ObstacleLayer"}
        - {name: inflation, type: "costmap_2d::InflationLayer"}
```

**Figure 3.57**

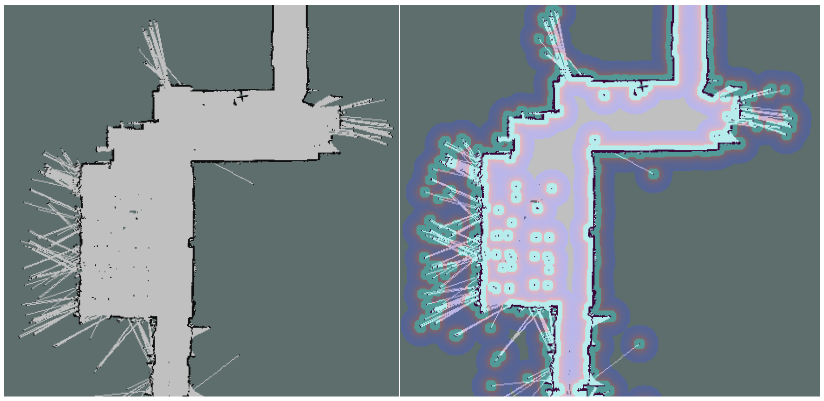
*Local Cost Map Parameter*


Open
local_costmap_params.yaml
~/catkin_ws/src/aiv_2dnav/config_edition3
Save

```
local_costmap:
  global_frame: map
  robot_base_frame: base_link
  update_frequency: 20.0
  publish_frequency: 20.0
  static_map: false
  rolling_window: true
  width: 10
  height: 10
  resolution: 0.05
  footprint_padding: 0.05
```

**Figure 3.58**

*Grid Map vs Inflated Cost Map*



### *3.8.5 Global Path Planner*

Path Planning is a process of calculating a sequence of valid configurations in order to send the object of the current position to destination. Path planning process considers the current known environment (map), the location where the object is currently placed (localization) and destination. Then the algorithm calculates the optimum path, which is considering the path length, feasibility of the object for moving in the environment, the energy consumed, and other variation factors the mark the path.

Therefore, the Path Planning requires,

- A known stating-position
- The desired destination
- Robot Geometry (footprint)
- Geometric description of the world such as map, local costmap, global costmap.

***3.8.5.1 A* Algorithm.*** A* is a search algorithm that based on the term of weight graphs. It focuses on the allocation of the path to goal node with the smallest cost. The tree path from the start point is conserved during the calculation until the best path is determined.

It performs this process by maintaining a tree of paths originating at the start node and extending those paths one edge at a time until its termination criterion is satisfied.

A* determine which paths to explore at each iteration main loop. The performance is based on the path cost. The estimation of the cost required to explore the path all the way to the goal. A* result path the ensure the least cost path to the goal.

$$f(n) = g(n) + h(n)$$

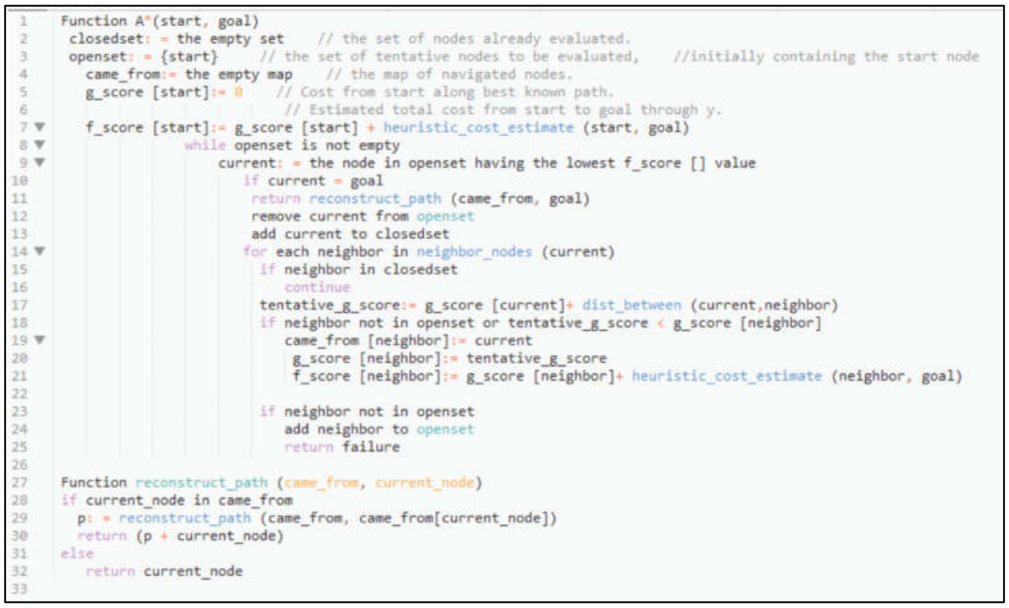
n: node

g(n): path cost from the start

h(n): heuristic function of the path from goal from n

**Figure 3.59**

*A* Algorithm*



```
Function A*(start, goal)
 closedset: = the empty set    // the set of nodes already evaluated.
 openset: = {start}     // the set of tentative nodes to be evaluated,    //initially containing the start node
   came_from:= the empty map    // the map of navigated nodes.
   g_score [start]:= 0    // Cost from start along best known path.
                           // Estimated total cost from start to goal through y.
   f_score [start]:= g_score [start] + heuristic_cost_estimate (start, goal)
               while openset is not empty
                   current: = the node in openset having the lowest f_score [] value
                      if current = goal
                       return reconstruct_path (came_from, goal)
                       remove current from openset
                       add current to closedset
                      for each neighbor in neighbor_nodes (current)
                        if neighbor in closedset
                           continue
                        tentative_g_score:= g_score [current]+ dist_between (current,neighbor)
                        if neighbor not in openset or tentative_g_score < g_score [neighbor]
                           came_from [neighbor]:= current
                            g_score [neighbor]:= tentative_g_score
                            f_score [neighbor]:= g_score [neighbor]+ heuristic_cost_estimate (neighbor, goal)

                        if neighbor not in openset
                           add neighbor to openset
                           return failure

Function reconstruct_path (came_from, current_node)
if current_node in came_from
  p: = reconstruct_path (came_from, came_from[current_node])
  return (p + current_node)
else
   return current_node
```

### *3.8.6 Local Path Planner*

Similar to Global path planner, the Local path planner plan the short path to navigate according to the global path while focusing avoiding the obstacles or the changes in the environment. The local path planner can be considered as obstacle avoidant planner.

***3.8.6.1 Dynamic Window Approach (DWA).*** Dynamic Window Approach choose the best trajectory that the ensure the robot to navigate safely from any obstacle. The trajectory is planned by selecting a velocity command pair (v, ω), linear velocity and angular velocity.

Dynamic Window Approach (DWA) algorithm function as follow:

- Discretizing the space.
- Forward simulation according to the robot station
- Command velocity pair the result the highest score trajectory is chosen
- clear and repeat

The cost function score of the best trajectory is evaluated using this formular:

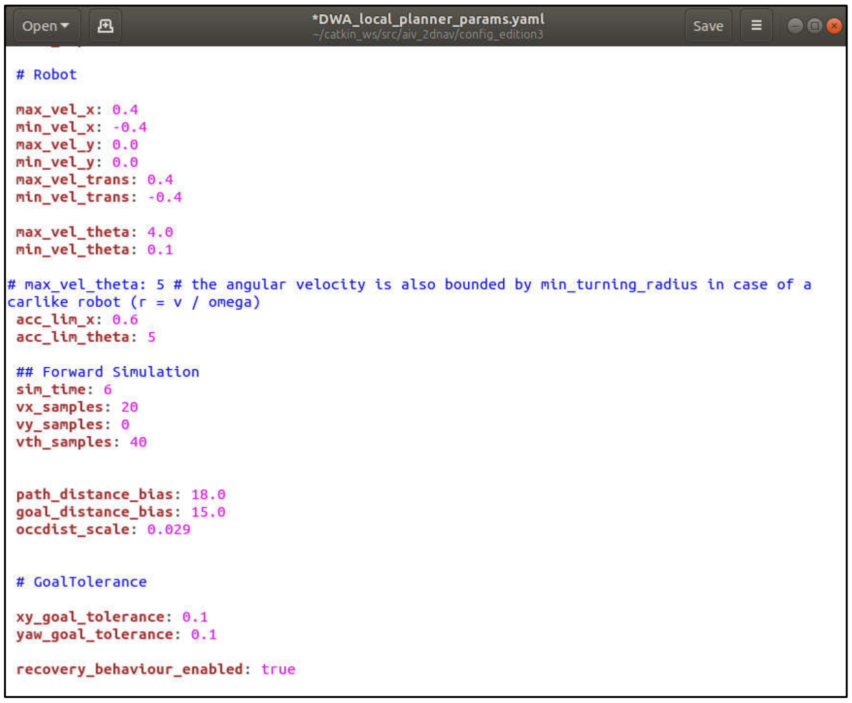
$$\begin{aligned} &cost \ = \\ &path\_distance\_bias \ * \ (distance\ to\ path\ from\ the\ endpoint\ of\ the\ trajectory\ in\ meters) \\ &+ goal\_distance\_bias \\ &\qquad * \ (distance\ to\ local\ goal\ from\ the\ endpoint\ of\ the\ trajectory\ in\ meters) \\ &+ occdist\_scale \ * \ (maximum\ obstacle\ cost\ along\ the\ trajectory\ in\ obstacle\ cost\ (0 - 254)) \end{aligned}$$

where,

- Path distance bias: the weight defines how close the DWA path to the global path.
- Goal distance bias: the weight defines the controller attempt to get to the goal also controls speed
- Occdist scale: the weight defines the controller attempt to avoid object.

**Figure 3.60**

*DWA Local Planner Parameter*


Open
*DWA_local_planner_params.yaml
~/catkin_ws/src/aiv_2dnav/config_edition3
Save

```
 # Robot

 max_vel_x: 0.4
 min_vel_x: -0.4
 max_vel_y: 0.0
 min_vel_y: 0.0
 max_vel_trans: 0.4
 min_vel_trans: -0.4

 max_vel_theta: 4.0
 min_vel_theta: 0.1

# max_vel_theta: 5 # the angular velocity is also bounded by min_turning_radius in case of a
carlike robot (r = v / omega)
 acc_lim_x: 0.6
 acc_lim_theta: 5

 ## Forward Simulation
 sim_time: 6
 vx_samples: 20
 vy_samples: 0
 vth_samples: 40


 path_distance_bias: 18.0
 goal_distance_bias: 15.0
 occdist_scale: 0.029


 # GoalTolerance

 xy_goal_tolerance: 0.1
 yaw_goal_tolerance: 0.1

 recovery_behaviour_enabled: true
```

## 3.9 AprilTag Detection and Localization

ROS April tag is utilized to detect, decode, and localize the tag. The detection supported AprilTag family tag36h11, tag36h10, tag25h9, tag25h7, and tag16h5. However, it cannot detect multiple AprilTag family at the same time. In this project tag36h11 family will be tested.

The node subscribes to

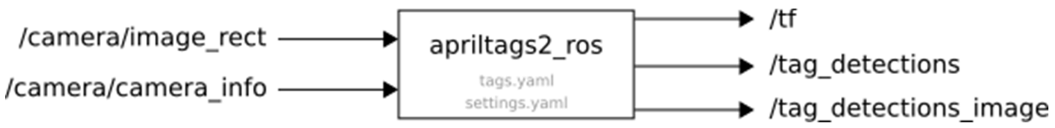
- /camera/camera_info, where the details of the camera using is stored and the calibration metrices.
- /camera/image_rect, which contains the image from the camera.

The node publishes to

- /tf topic the relative pose between each detected tag to the camera frame.
- /tag_detections, the information that include the tag id, size, and pose.
- /tag_detections_image, the image with the detected tags ID overlayed.

**Figure 3.61**

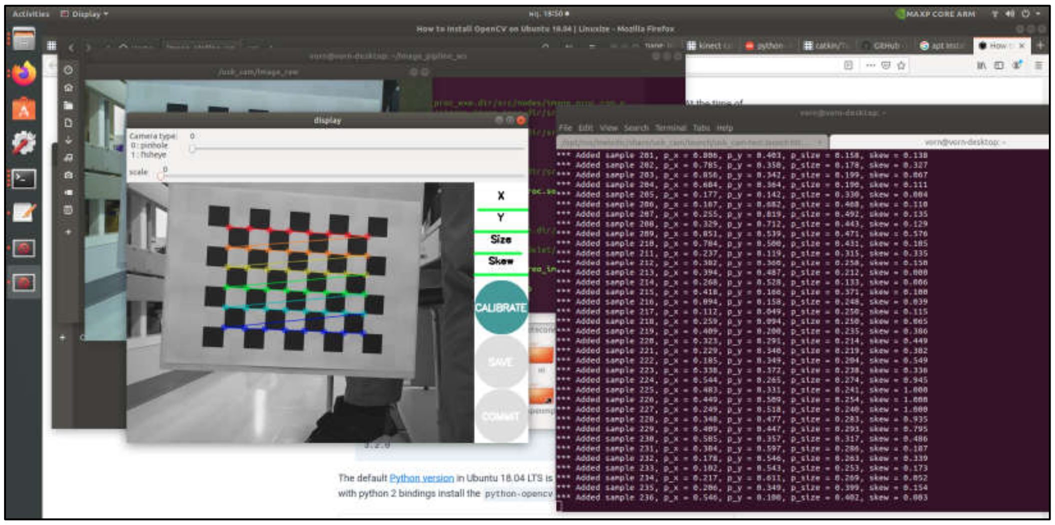
*Apriltag ROS Node*



However, the camera must be calibrated before using. The figure below is calibration the camera. The 25 mm square chessboard grid is use as templet for this calibration. After calibration, the calibrated parameters are saved to the package and utilized one the node is launched.

**Figure 3.62**

*Camera Calibration*

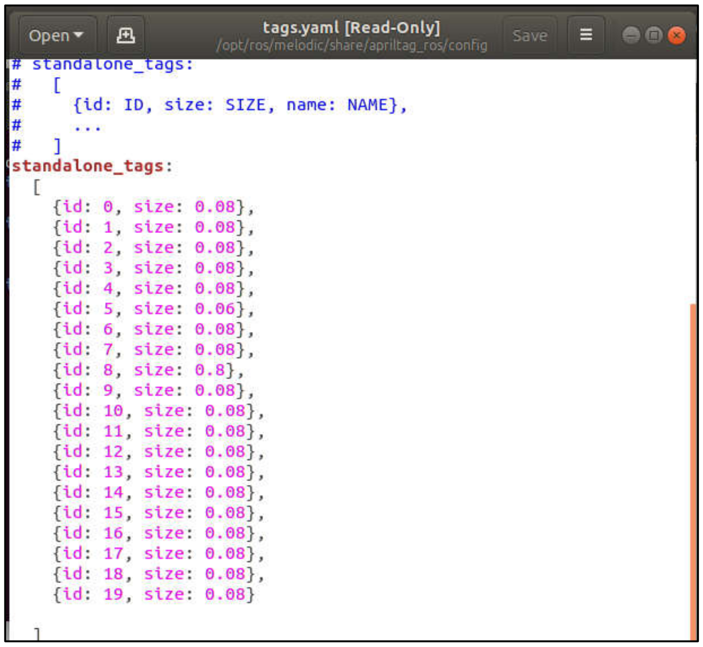


```
*** Added sample 201, p_x = 0.806, p_y = 0.403, p_size = 0.158, skew = 0.138
*** Added sample 202, p_x = 0.785, p_y = 0.358, p_size = 0.178, skew = 0.327
*** Added sample 203, p_x = 0.856, p_y = 0.342, p_size = 0.199, skew = 0.067
*** Added sample 204, p_x = 0.604, p_y = 0.364, p_size = 0.190, skew = 0.111
*** Added sample 205, p_x = 0.177, p_y = 0.142, p_size = 0.330, skew = 0.004
*** Added sample 206, p_x = 0.107, p_y = 0.682, p_size = 0.408, skew = 0.110
*** Added sample 207, p_x = 0.255, p_y = 0.819, p_size = 0.492, skew = 0.135
*** Added sample 208, p_x = 0.329, p_y = 0.712, p_size = 0.443, skew = 0.129
*** Added sample 209, p_x = 0.851, p_y = 0.539, p_size = 0.471, skew = 0.576
*** Added sample 210, p_x = 0.704, p_y = 0.500, p_size = 0.431, skew = 0.185
*** Added sample 211, p_x = 0.237, p_y = 0.119, p_size = 0.315, skew = 0.335
*** Added sample 212, p_x = 0.302, p_y = 0.300, p_size = 0.258, skew = 0.158
*** Added sample 213, p_x = 0.394, p_y = 0.487, p_size = 0.212, skew = 0.008
*** Added sample 214, p_x = 0.268, p_y = 0.528, p_size = 0.133, skew = 0.066
*** Added sample 215, p_x = 0.418, p_y = 0.166, p_size = 0.371, skew = 0.108
*** Added sample 216, p_x = 0.094, p_y = 0.158, p_size = 0.248, skew = 0.039
*** Added sample 217, p_x = 0.112, p_y = 0.049, p_size = 0.250, skew = 0.115
*** Added sample 218, p_x = 0.259, p_y = 0.094, p_size = 0.250, skew = 0.065
*** Added sample 219, p_x = 0.409, p_y = 0.200, p_size = 0.235, skew = 0.386
*** Added sample 220, p_x = 0.323, p_y = 0.291, p_size = 0.214, skew = 0.449
*** Added sample 221, p_x = 0.229, p_y = 0.340, p_size = 0.219, skew = 0.382
*** Added sample 222, p_x = 0.185, p_y = 0.349, p_size = 0.204, skew = 0.549
*** Added sample 223, p_x = 0.338, p_y = 0.372, p_size = 0.238, skew = 0.338
*** Added sample 224, p_x = 0.544, p_y = 0.265, p_size = 0.274, skew = 0.945
*** Added sample 225, p_x = 0.483, p_y = 0.331, p_size = 0.241, skew = 1.000
*** Added sample 226, p_x = 0.449, p_y = 0.509, p_size = 0.254, skew = 1.000
*** Added sample 227, p_x = 0.249, p_y = 0.518, p_size = 0.240, skew = 1.000
*** Added sample 228, p_x = 0.348, p_y = 0.477, p_size = 0.283, skew = 0.935
*** Added sample 229, p_x = 0.409, p_y = 0.447, p_size = 0.293, skew = 0.795
*** Added sample 230, p_x = 0.585, p_y = 0.357, p_size = 0.317, skew = 0.486
*** Added sample 231, p_x = 0.304, p_y = 0.597, p_size = 0.286, skew = 0.107
*** Added sample 232, p_x = 0.178, p_y = 0.546, p_size = 0.263, skew = 0.339
*** Added sample 233, p_x = 0.102, p_y = 0.543, p_size = 0.253, skew = 0.173
*** Added sample 234, p_x = 0.217, p_y = 0.611, p_size = 0.269, skew = 0.052
*** Added sample 235, p_x = 0.206, p_y = 0.349, p_size = 0.399, skew = 0.154
*** Added sample 236, p_x = 0.546, p_y = 0.100, p_size = 0.402, skew = 0.003
```

Similarly, the tag ID, size, tag bundle must be declared. The figure below is the tag declaration yaml file.

**Figure 3.63**

*Tag ID Declaration*



```
# standalone_tags:
#   [
#     {id: ID, size: SIZE, name: NAME},
#     ...
#   ]
standalone_tags:
  [
    {id: 0, size: 0.08},
    {id: 1, size: 0.08},
    {id: 2, size: 0.08},
    {id: 3, size: 0.08},
    {id: 4, size: 0.08},
    {id: 5, size: 0.06},
    {id: 6, size: 0.08},
    {id: 7, size: 0.08},
    {id: 8, size: 0.8},
    {id: 9, size: 0.08},
    {id: 10, size: 0.08},
    {id: 11, size: 0.08},
    {id: 12, size: 0.08},
    {id: 13, size: 0.08},
    {id: 14, size: 0.08},
    {id: 15, size: 0.08},
    {id: 16, size: 0.08},
    {id: 17, size: 0.08},
    {id: 18, size: 0.08},
    {id: 19, size: 0.08}

  ]
```

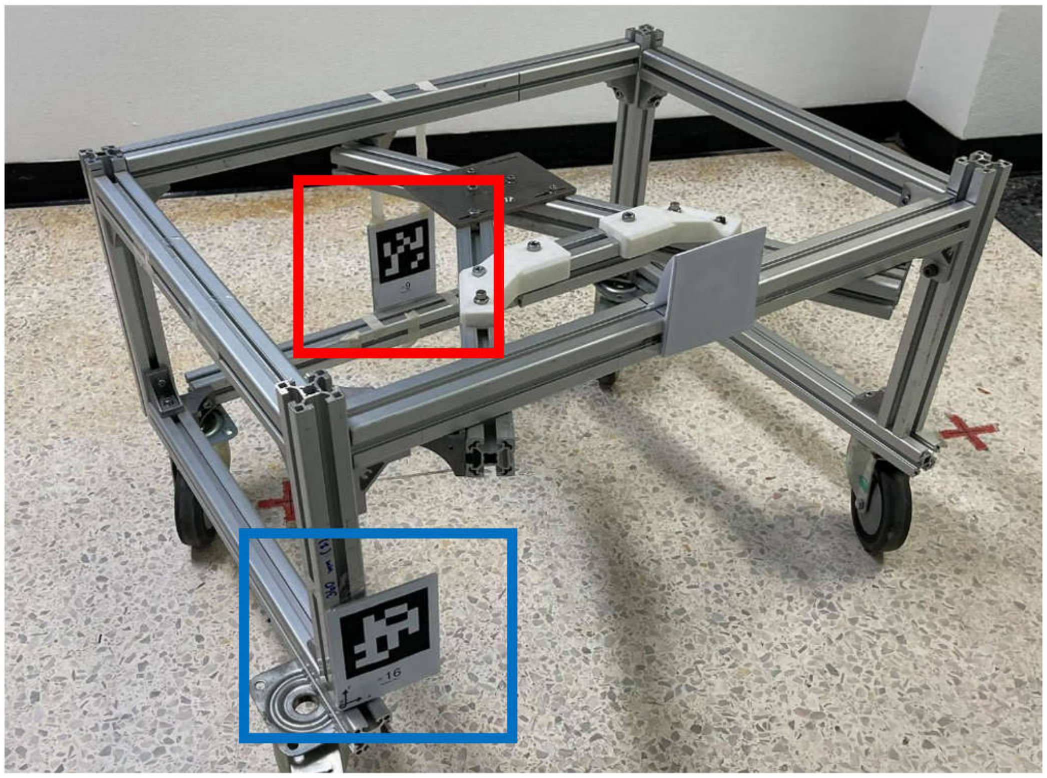
## 3.10 Cart Finding and Engaging

Until now the AIV can be able to build the map and navigate in the map from place to place autonomously. It can also identify the tag and localize the position of the tag to itself. The cart finding and engaging is implemented to enhance the ability of the AVI to perform a task autonomously.

There are 2 tags attached to each identical cart. One tag is used as cart ID tag and another is used as guiding tag. The tag is attached at predefined position.

**Figure 3.64**

*Cart ID Tag in Blue, Guiding Tag in Red*

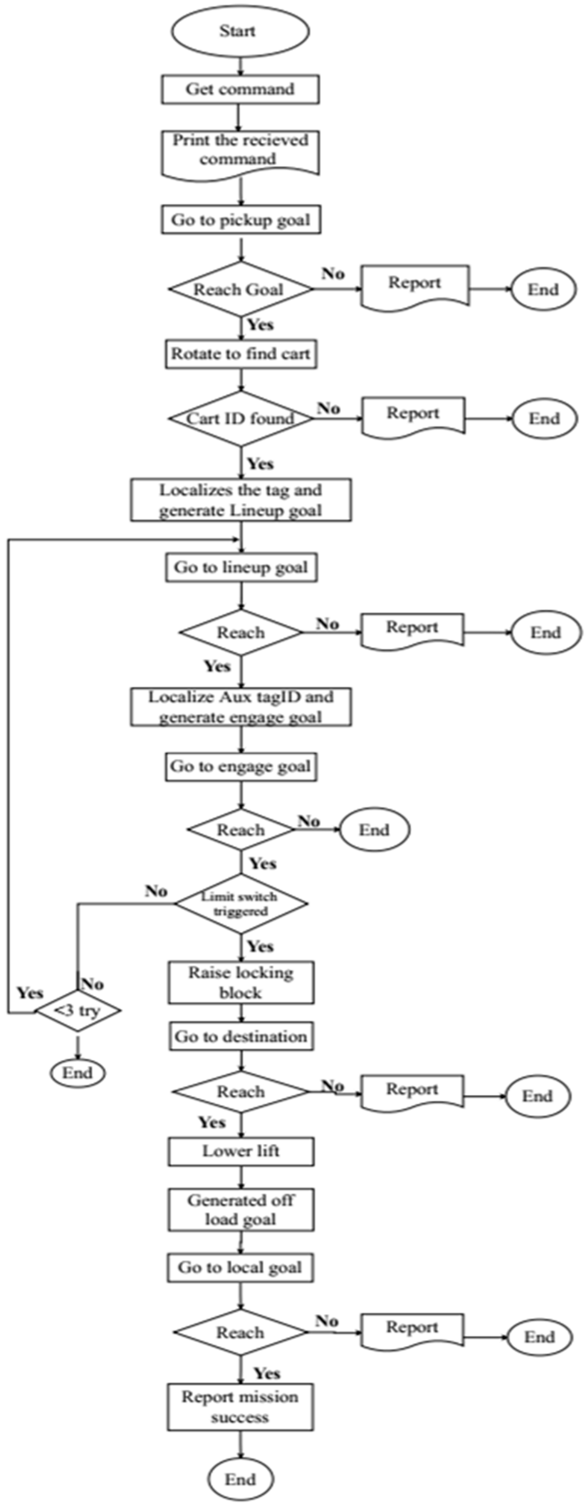


The feature of finding, engaging the cart and move is written in python script. It is the implemented using ROS Actionlib. The function requires the coordinate of cart picking area, the cart ID tag, guiding tag and coordinate of cart destination.

The python script is working as in the flowchart bellow.

**Figure 3.65**

*Cart Finding and Engaging Flowchart*



# CHAPTER 4
# RESULTS AND DISCUSSION

## 4.1 Motion Control

The PID controller is implemented at the frequency of 100Hz for each loop. It can be noticed that for the inner loop, the loop controlling the velocity of the right wheel and the velocity of the left wheel, and the outer loop, the loop controlling the linear velocity and the angular velocity, the parameter tuned by MATLAB results as 0 for Derivative gain. It means that the derivative part is not requires to make the velocity follow the set velocity. Similarly, since there is no derivative part the filter N is not necessary as well. Therefore, from the parameter tune by MATLAB, the controller is a PI controller.

The figure bellow shows the performance of the inner loop controlling the right wheel (cmd_vr).

**Figure 4.1**

*Cmd_vr Performance*

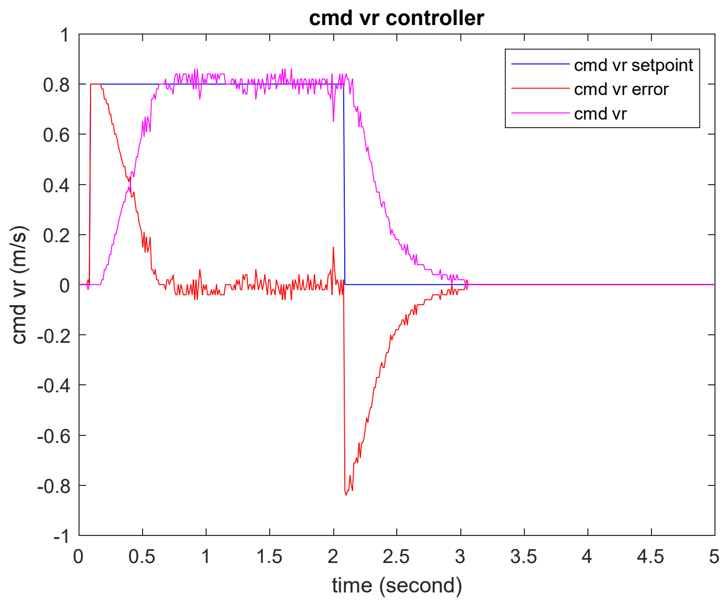
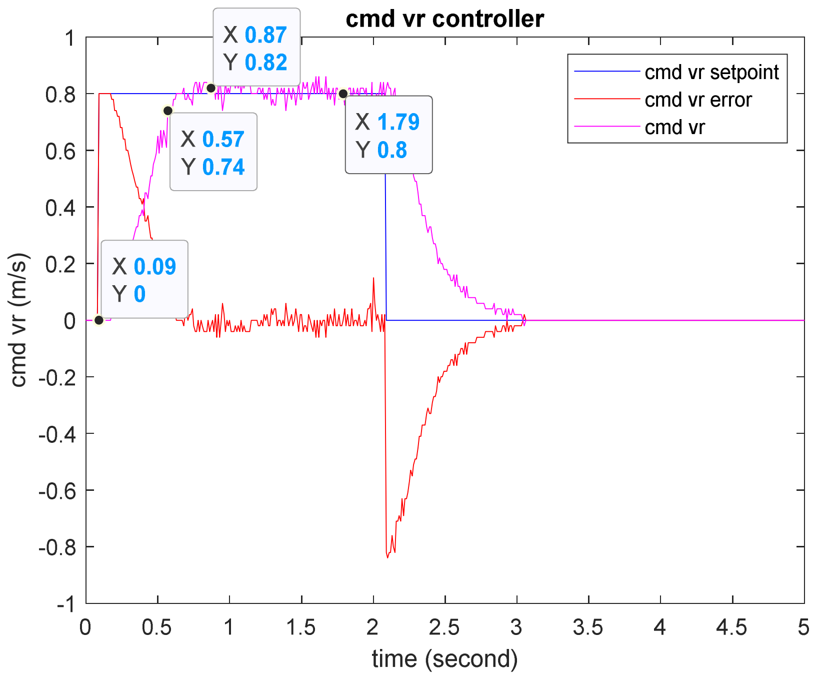


**Figure 4.2**

*Cmd_vr Performance with Marking*



From the graph above, it can be seen that

- The rise time: 0.48 second
- Peak: 0.82 m/s

The figure bellow shows the performance of the outer loop controlling the forward velocity (cmd_v).

**Figure 4.3**

*Cmd_v Performance*

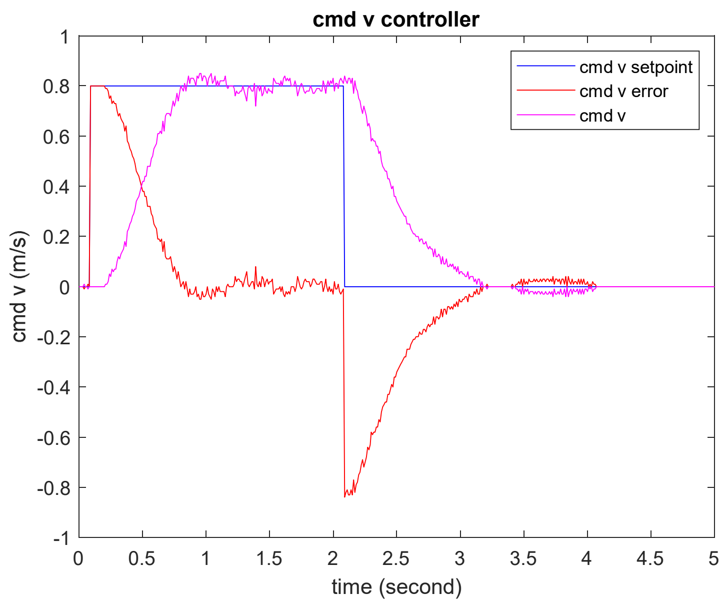
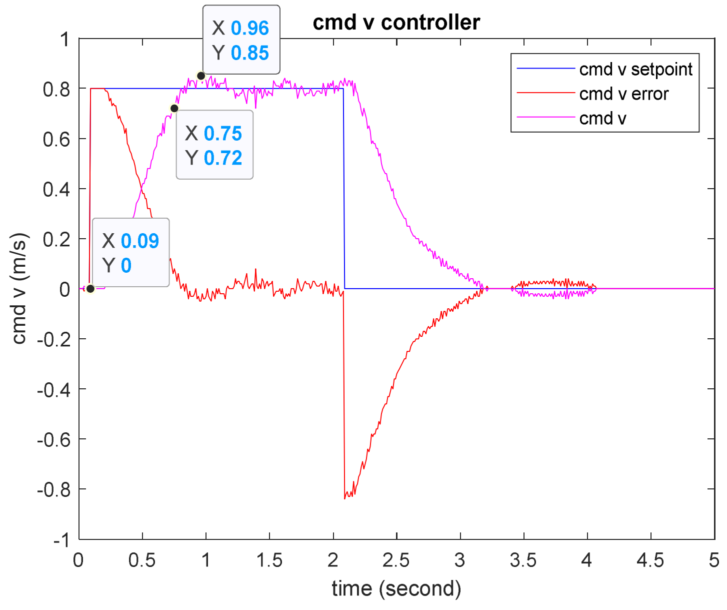


**Figure 4.4**

*Cmd_v Performance with Marking*



From the graph above, it can be seen that

- The rise time: 066 second
- Peak: 0.85 m/s

The figure bellow shows the performance of the outer loop controlling the angular velocity (cmd_w).

**Figure 4.5**

*Cmd_w Performance*

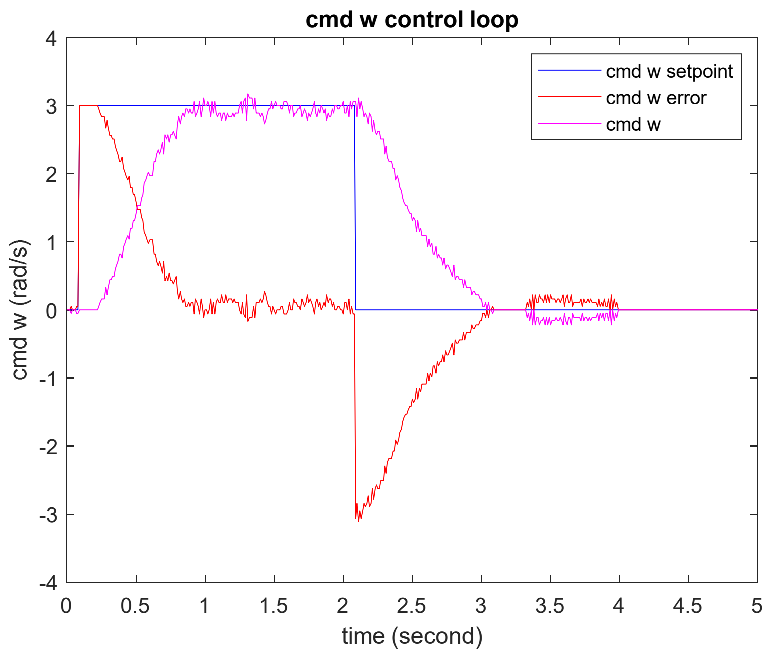
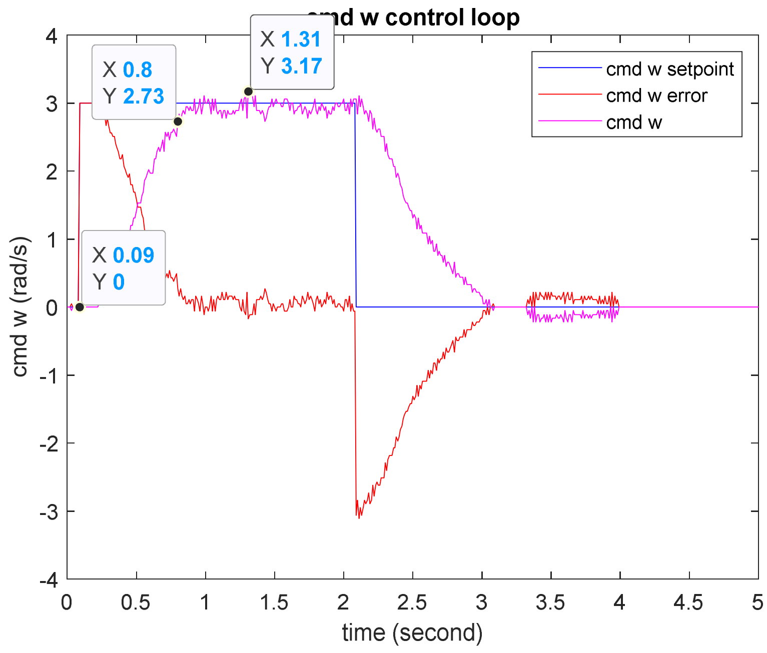


**Figure 4.6**

*Cmd_w Performance with Marking*



From the graph above, it can be seen that

- The rise time: 0.71 second
- Peak: 3.17 rad/s

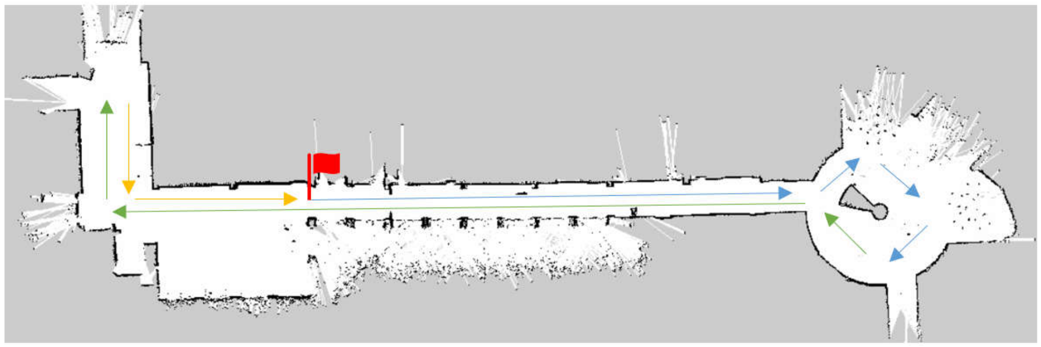
The controller performance is close to what designed and tuned in MATLAB. The controller drives the velocity to the setpoint with minimal oscillation.

## 4.2 Mapping

Gmapping SLAM is evaluated in here according to loop closure and performance. The map scanning is experimented in ISE building ails. The AIV is manually maneuvered along the ails of the building for one round as arrow-indicated figure below. Starting form position 🚩 follows the blue arrow, the green arrow, and the yellow arrow.

**Figure 4.7**

*AIV Path for Map Scanning*



To start mapping, the node that taking care of merging the laser scan data, odometry must be launched in advance. After checking that the require topic is available, the gmapping can be launched.

The mapping experiment is going to be tested as shown in **Table 4.1 Table 4.1**.

**Table 4.1**

*Mapping Experiment*

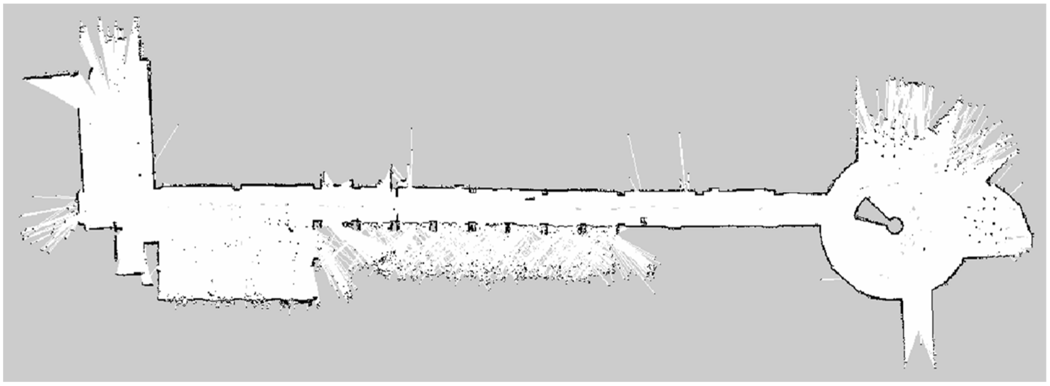
| Laser | Particle Value | Scan Round | Resolution |
|---|---|---|---|
| Dual Laser 360-degree scan | 30 | 1 | 0.05 m/pixel |
| Dual Laser 360-degree scan | 30 | 2 | 0.05 m/pixel |
| Dual Laser 360-degree scan | 60 | 1 | 0.05 m/pixel |
| Dual Laser 360-degree scan | 60 | 2 | 0.05 m/pixel |

The first experiment is started with particle value of 30 and AIV is maneuvered around the building one round. The parameter below are some key parameters used for this test.

- Iterations = 5
- map_update_interval = 5.0
- maxUrange = 16.0
- linearUpdate = 1.0
- particlesvalue = 30

**Figure 4.8**

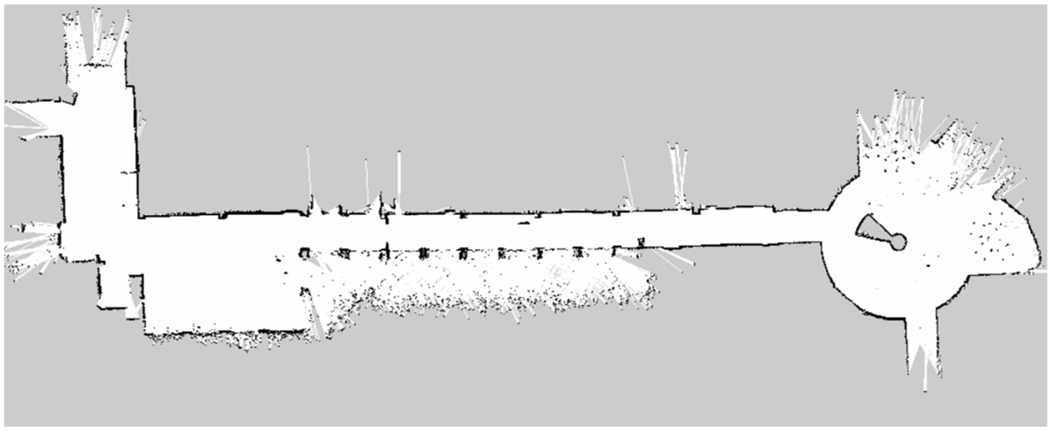
*ISE Map Particle Value 30 0.05m/pixel One Round*



The second experiment is started with particle value of 30 and AIV is maneuvered around the building for 2 rounds. The parameter is kept the same as the first experiment.

**Figure 4.9**

*ISE Map Particle Value 30 0.05m/pixel 2 Round*

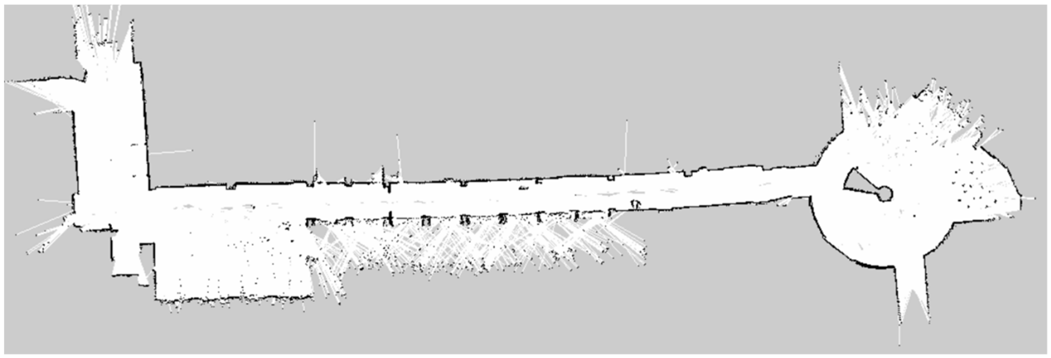


The third experiment is started with particle value of 60 and AIV is maneuvered around the building one round. The parameter below are some key parameters used for this test.

- Iterations = 5
- map_update_interval = 5.0
- maxUrange = 16.0
- linearUpdate = 1.0
- particlesvalue = 60

**Figure 4.10**

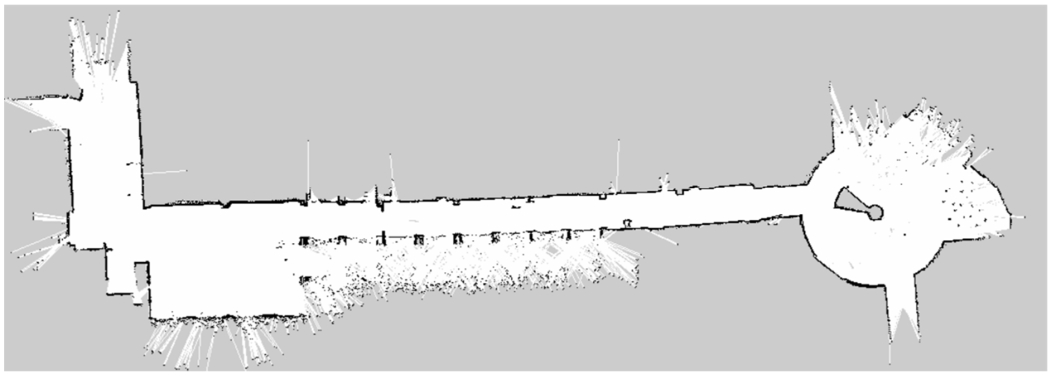
*ISE Map Particle Value 60 0.05m/pixel One Round*



The fourth experiment is started with particle value of 60 and AIV is maneuvered around the building for 2 rounds. The parameter is kept the same as the third experiment.

**Figure 4.11**

*ISE Map Particle Value 60 0.05m/pixel 2 Round*



Comparing the map from experiment 1 it can be seen that there are many spots along the ails where the map is registered as unknow or fault positive spot as indicated in the

**Figure 4.12**

*ISE Map Particle Value 60 0.05m/pixel 2 Round Analysis*

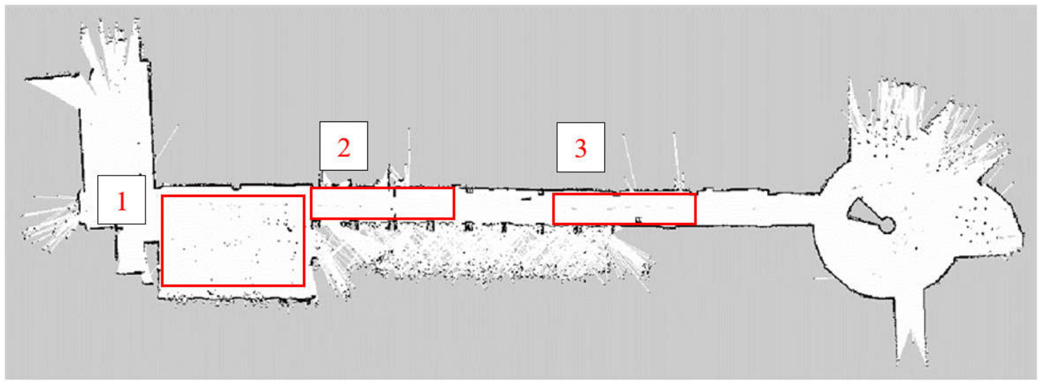
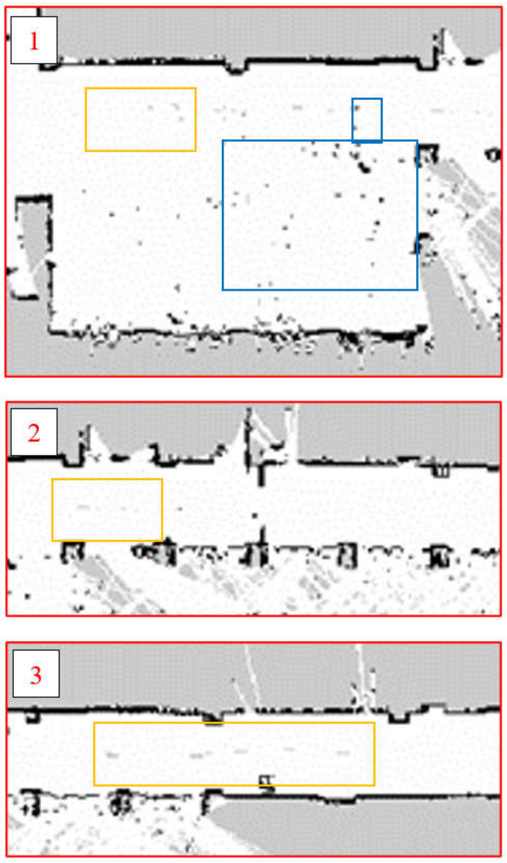


**Figure 4.13**

*Map Defection*



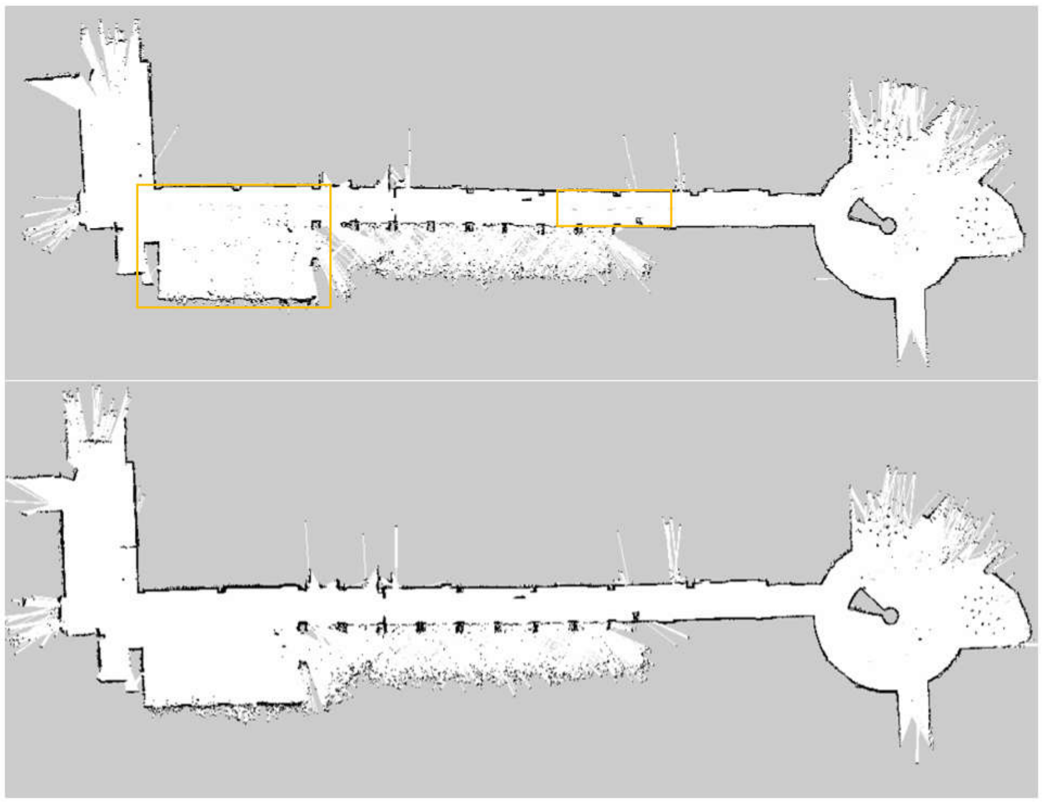
Unknow registered map is shown in the yellow rectangles and the fault positive registered map in blue rectangles. It happens at the place where the laser scanner reaches its maximum measurable value which is not reliable and neglected.

However, when comparing the first experiment to with the second where the AVI scans the area for 2 rounds, the problems mentioned above are no longer existed. Scanning the area for 2 rounds give more data for the algorithm to work with resulting the more refined map while the map-feature-capturing is also improved.

The figure below shows the comparison of the first experiment map and the second experiment map side by side.

**Figure 4.14**)

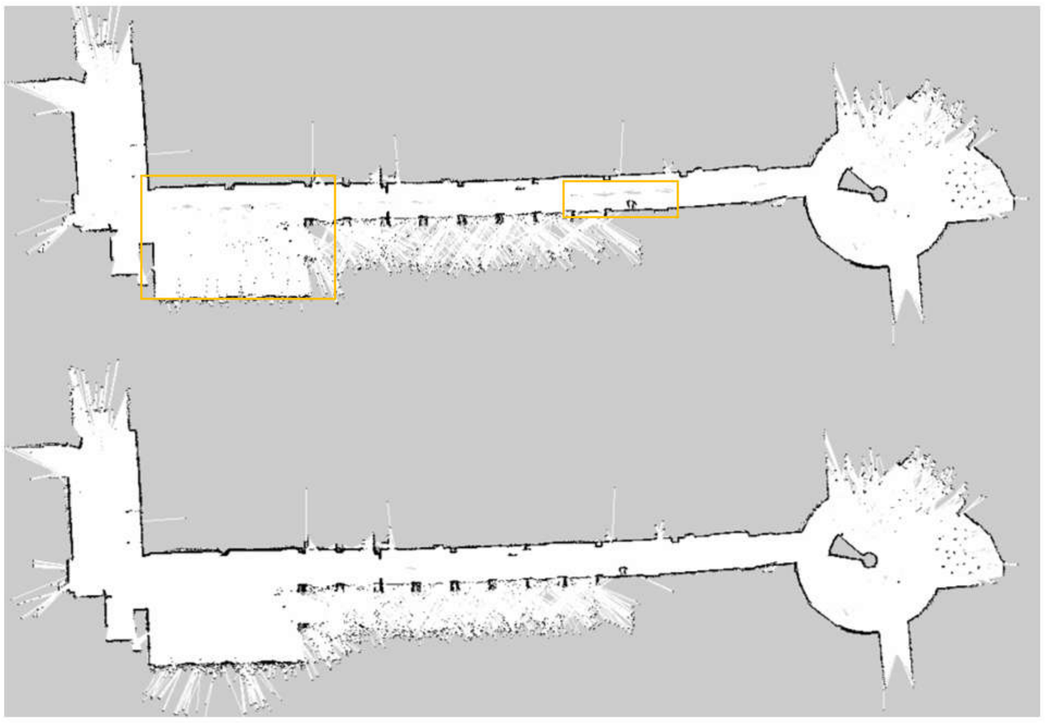
*Map Side by Side Comparison Experiment 1 (above) and 2 (below*



Similarly, comparing experiment 3 and 4, map refinement improves when the area is scanned for 2 rounds.

**Figure 4.15**

*Map Side by Side Comparison Experiment 3 (above) and 4 (below)*



Beside the number of scan round, the difference between the experiment 1,2 and 3,4 is the number of particles. Increasing the number of particles increases the calculation. As tested, the peak CPU usage of the NVIDIA increase from about 80 percent when using 30 particles to about 95 percent when using 60 particles.

However, the result of the mapping is not significantly different when increasing the particle number from 30 to 60 due to the reason that the map is not very complex.

## 4.3 AMCL

The Localization of the AIV is handle by the AMCL. The Algorithm initially starts by distribute the particle throughout the map. The particles converge to the real location of the AIV as it moves.

**Figure 4.16**

*Particle Distribution at Time 0*

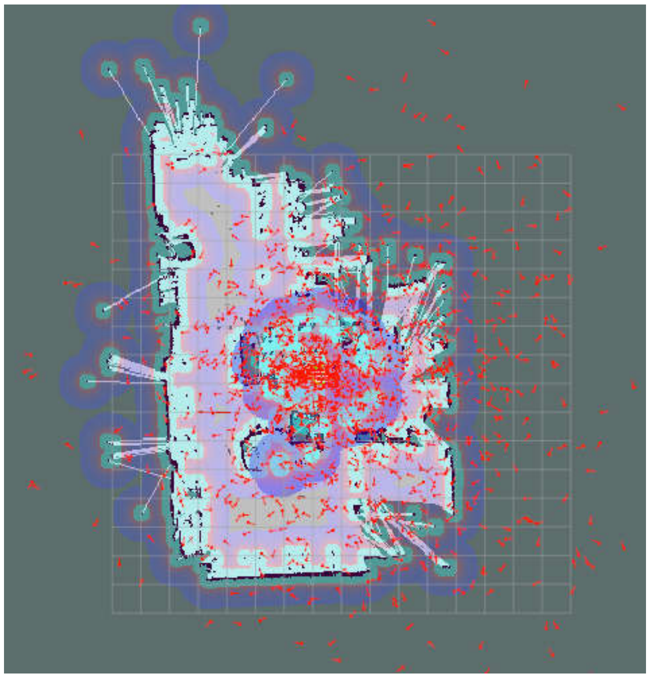


**Figure 4.17**

*Particle Distribution at Time 1*

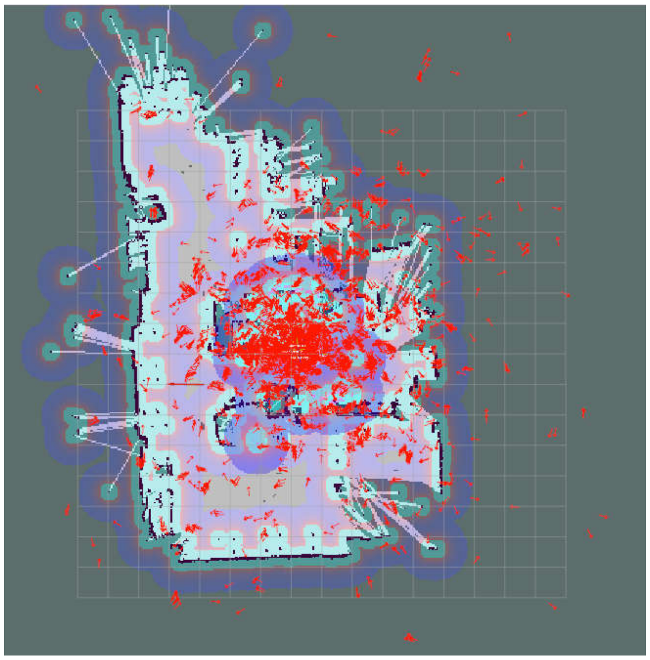


**Figure 4.18**

*Particle Distribution at Time 2*

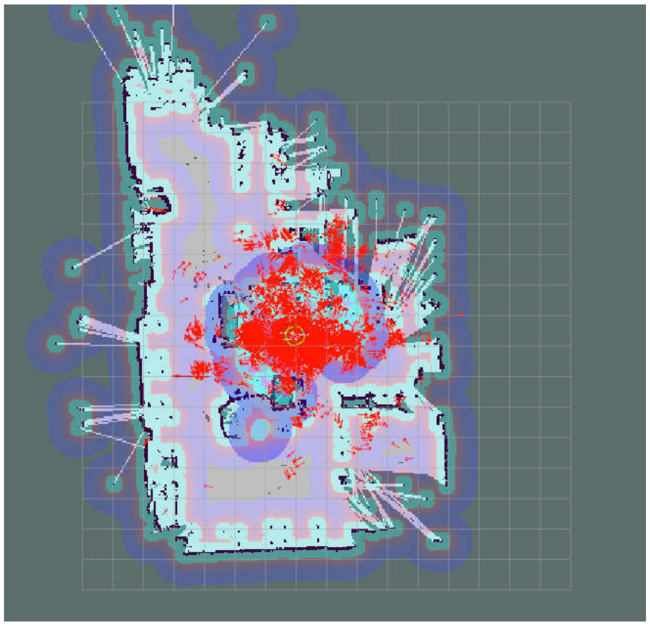
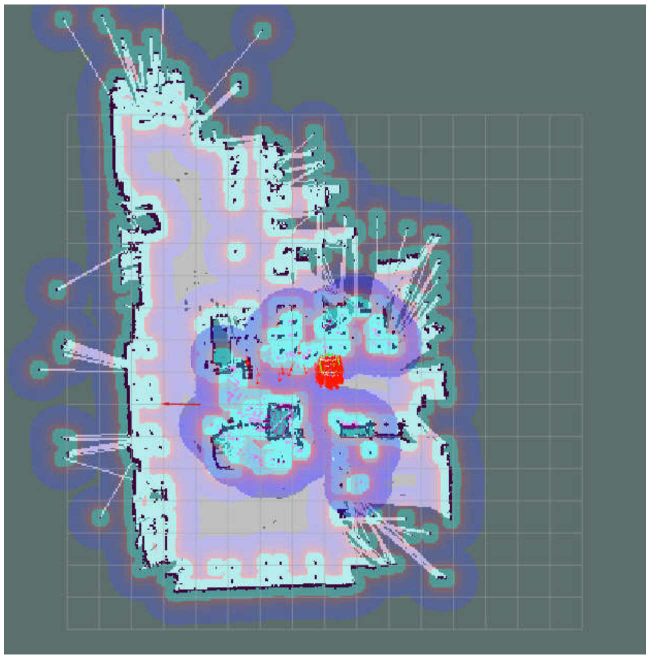


**Figure 4.19**

*Particle Distribution at Time 3*



## 4.4 Navigation

### *4.4.1 Global Path Planner*

Both A* and Dijkstra algorithm as can be seen in Figure 4.20 and Figure 4.21 provides the optimum path.

**Figure 4.20**

*A* Path*

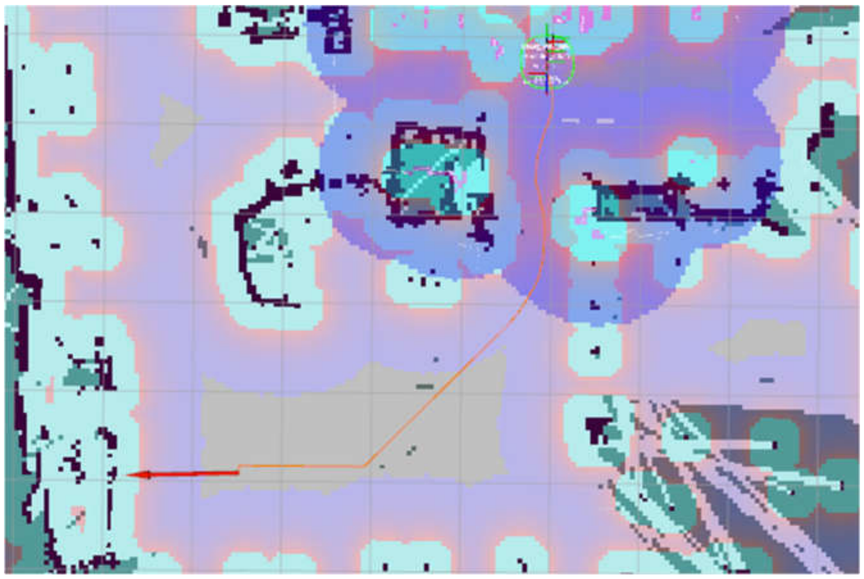
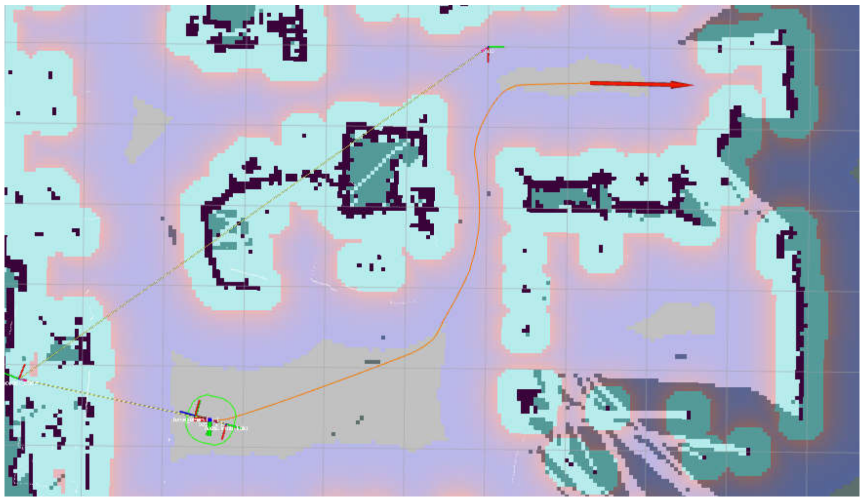


**Figure 4.21**

*Dijkstra Path*



### *4.4.2 DWA Local Path Planner*

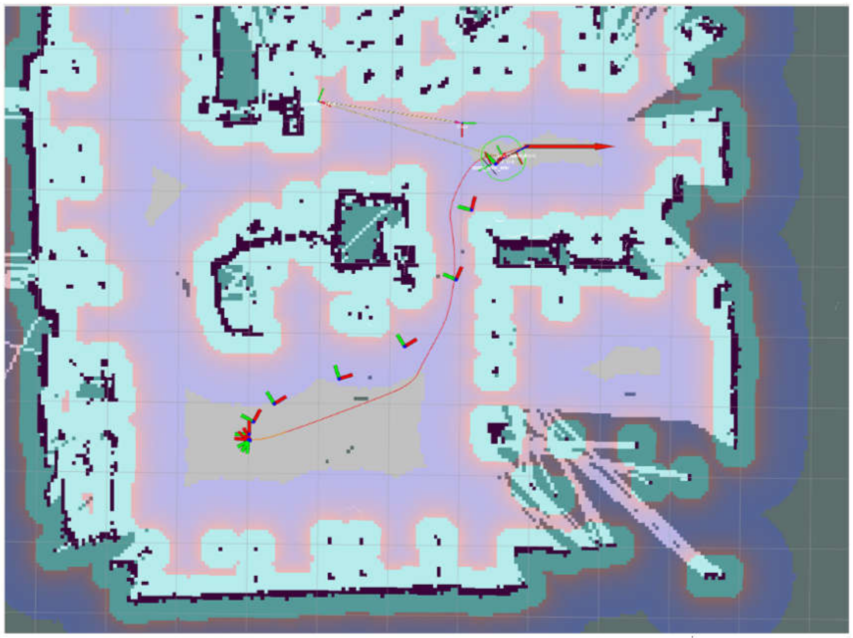
The DWA simulate the trajectory according to the situation at that time. The simulation time determine the approximate position if the velocity command pairs (cmd_v and cmd_w) are apply. The longer the simulation time results in the further the prediction, so the robot can predict well ahead. Below figure shows the different in simulation time.

The trajectory is chosen according to the score of cost function at that particular time, which consist of goal bias, path bias, and occdist scale. 2 of them greatly affect the path selection of the DWA. As already discussed, the local planner DWA follows the global path planned by the global path planner to the goal. By varying the 2 biases, the path produced by the DWA deviates or follows the global path. The higher the path bias, the trajectory tend to follow the global path while the higher the goal bias the trajectory tend to deviate from the global path to toward the goal. The behavior of the path is evaluated below.

The test is to move from point A (on the right-hand side) to point B (on the left-hand side) in the map as shown in the figure below with various path bias and goal bias value.

**Figure 4.22**

*DWA Path Bias 18 Goal Bias 15*



The values of the parameters are:

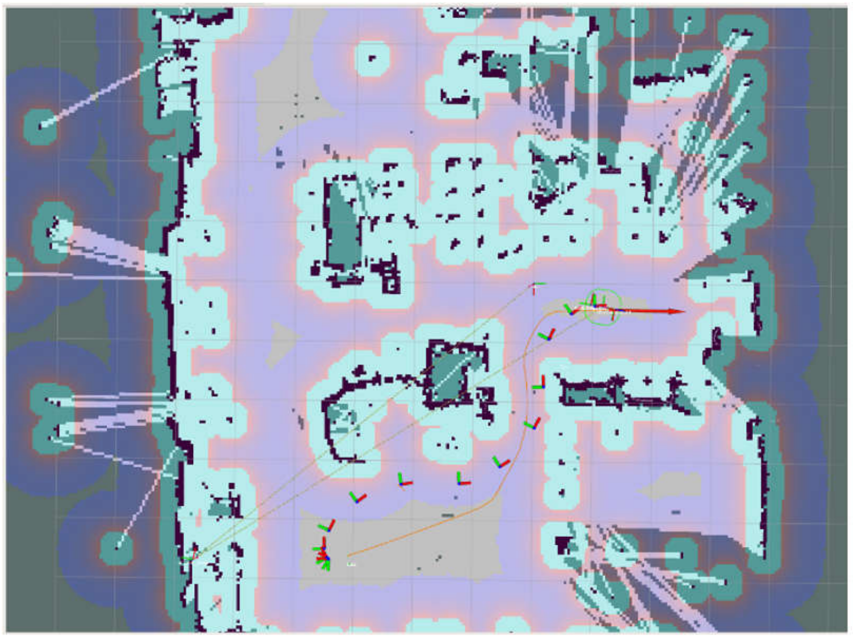
- path bias 18
- goal bias 15

From the path above, rms error is calculated.

- RMS path deviation = 0.3127 meter

**Figure 4.23**

*DWA Path Bias 18 Goal Bias 30*



The values of the parameters are:

- path bias 18
- goal bias 30

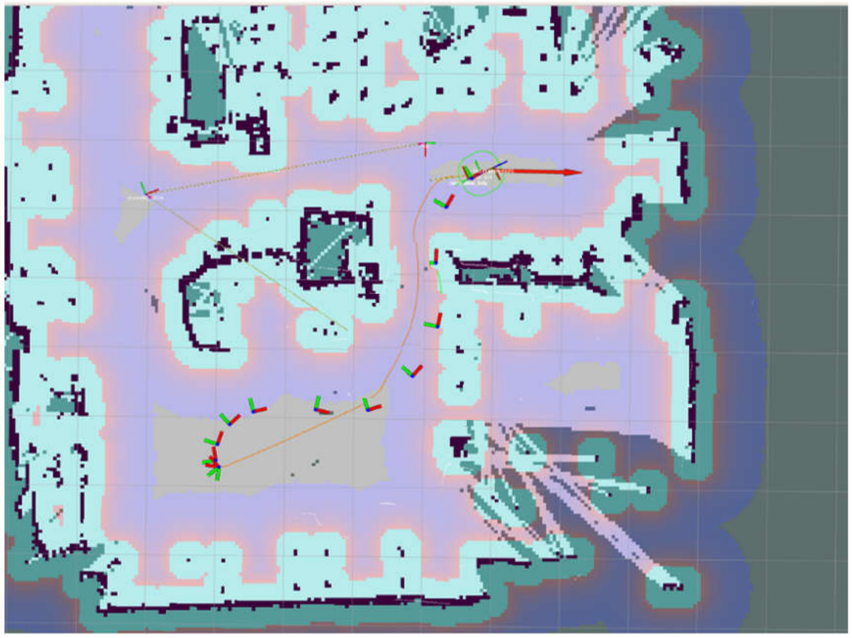
From the path above, rms error is calculated.

- RMS path deviation = 0.4479 meter

It can be noticed that the DWA path deviates global path more comparing to the previous test while it tends to lean toward the goal as the parameter suggests. The path also tends to go near the wall where it is close to the goal.

**Figure 4.24**

*DWA Path Bias 36 Goal Bias 15*



The values of the parameters are:

- path bias 36
- goal bias 15

From the path above, rms error is calculated.

- RMS path deviation = 0.3283 meter

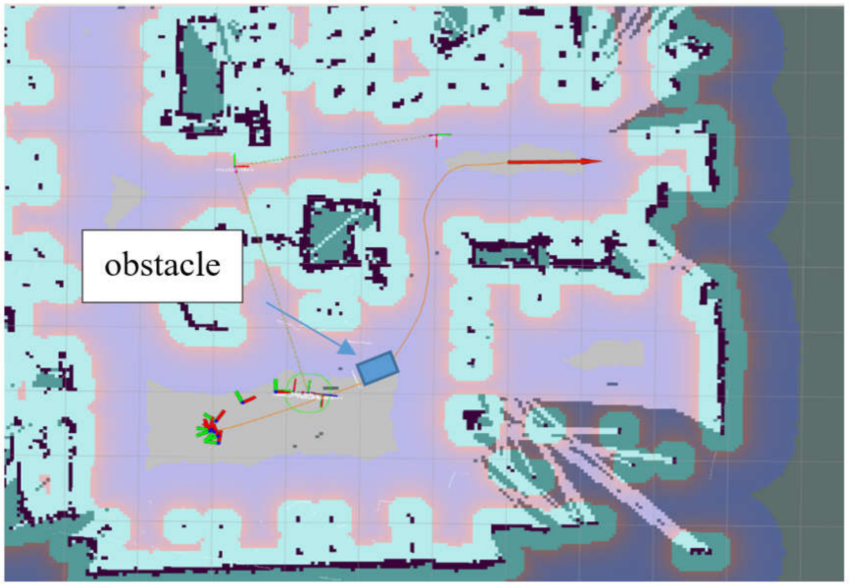
Comparing the RMS path deviation to the previous path, it is closed to the first test. However, it is noticeable that the DWA path tend to follow the global path more than the first test.

In case of there is obstacle on the path, different parameter values behave noticeably different.

In the case below with the path bias and goal bias equals to 36 and 15 respectively, the AIV follows the path, however when it encounters the obstacle the AIV just stop and does not move forward anymore.

**Figure 4.25**

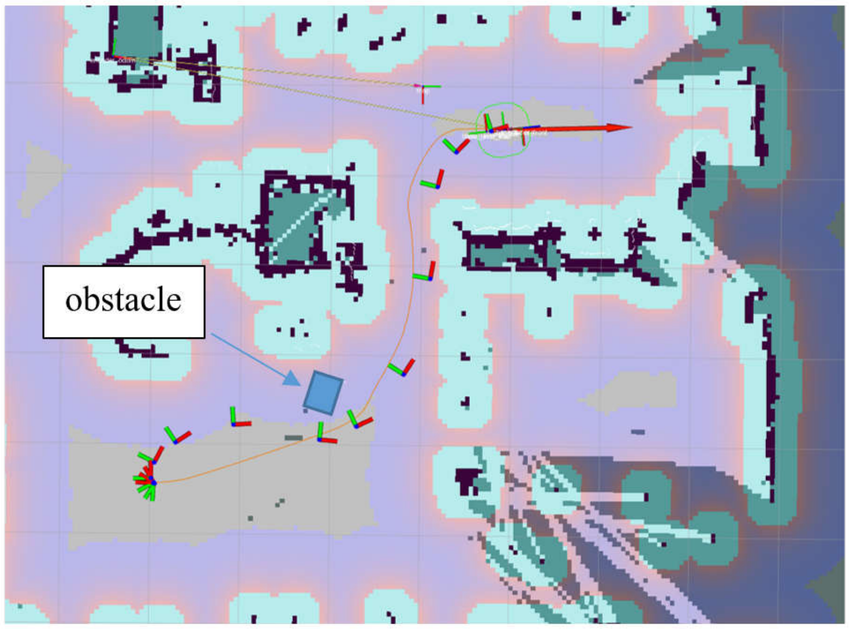
*DWA Path Bias 36 Goal Bias 15 Stuck in Front of Obstacle*



In the same situation, changing the path bias and goal bias to 18 and 15 respectively, the AIV navigate to goal without any problem.

**Figure 4.26**

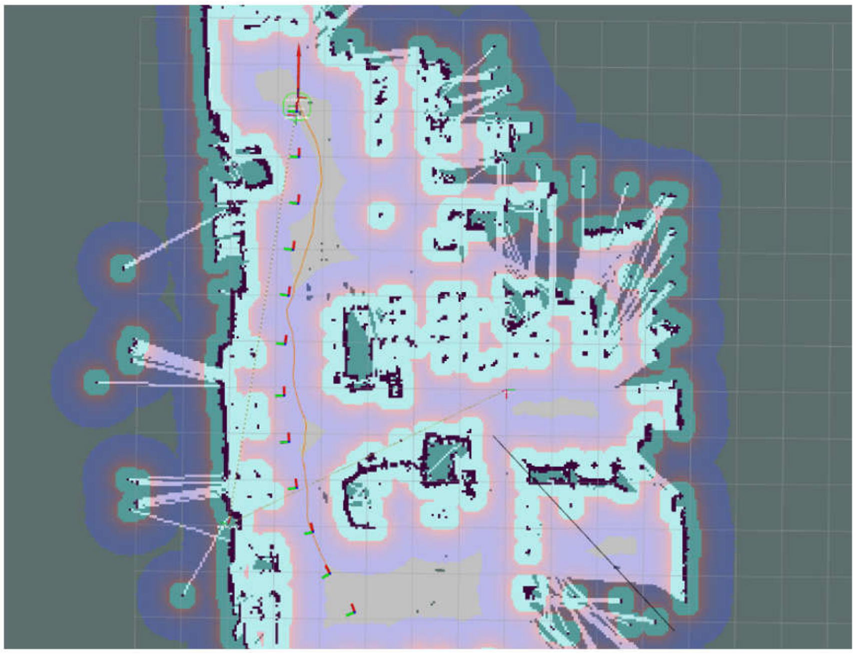
*DWA Path Bias 18 Goal Bias 15 Avoided Obstacle on the Path*



The path bias and goal bias to 18 and 15 respectively give better performance. Below is the testing of these parameter value in a straight path. The RMS path deviation equals to 0.1935 meter.

**Figure 4.27**

*DWA Path Bias 18 Goal Bias 15 Straight Path*



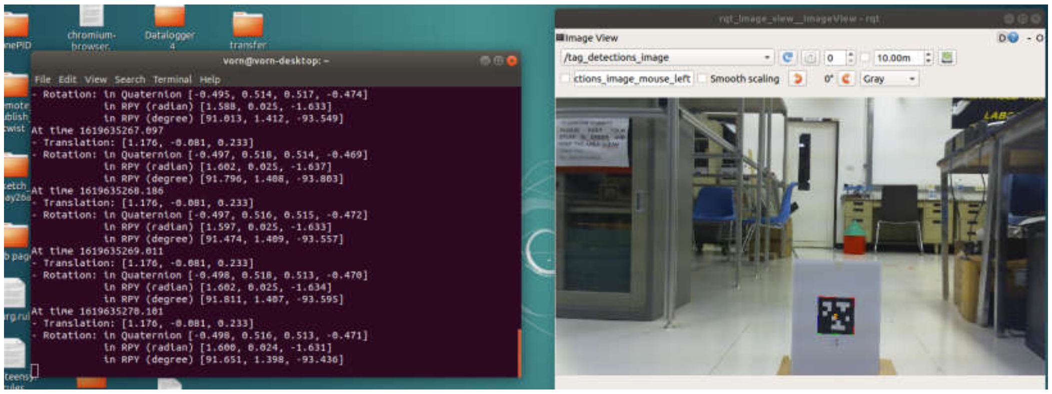
## 4.5 Tag Detection and Localization

The tag detection and localization, it is important to check whether the required topics for the detection are available. Then the continuous detection node can be launched. The 36h11 family tags are used and tested.

After launching the node if there are tags placed in the visible range of the camera, the tag ID will be detected and the transformation of the tags to camera, and image with detected tag coordinate frame overlay are published. The transformation from a known frame to the tag can be obtain using command for example, rosrun tf tf_echo /base_link /tag_2 and the result can be monitored on terminal reporting the translation and rotation. The frame of the tag can also be visualized on Rviz.

**Figure 4.28**

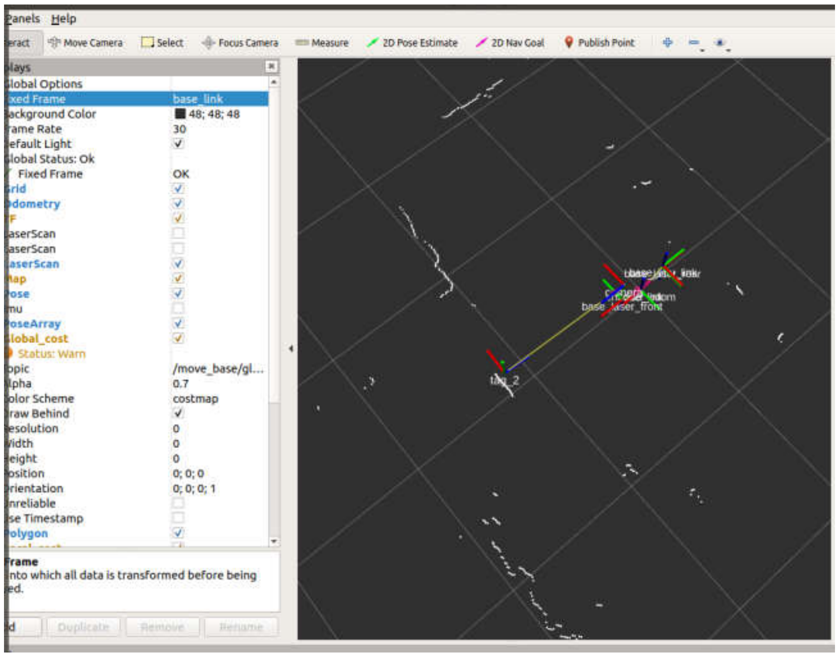
*Detected Tag_2 and the Transformation from Base_link to Tag_2*



```
- Rotation: in Quaternion [-0.495, 0.514, 0.517, -0.474]
            in RPY (radian) [1.588, 0.025, -1.633]
            in RPY (degree) [91.013, 1.412, -93.549]
At time 1619635267.097
- Translation: [1.176, -0.081, 0.233]
- Rotation: in Quaternion [-0.497, 0.518, 0.514, -0.469]
            in RPY (radian) [1.602, 0.025, -1.637]
            in RPY (degree) [91.796, 1.408, -93.803]
At time 1619635268.186
- Translation: [1.176, -0.081, 0.233]
- Rotation: in Quaternion [-0.497, 0.516, 0.515, -0.472]
            in RPY (radian) [1.597, 0.025, -1.633]
            in RPY (degree) [91.474, 1.409, -93.557]
At time 1619635269.011
- Translation: [1.176, -0.081, 0.233]
- Rotation: in Quaternion [-0.498, 0.518, 0.513, -0.470]
            in RPY (radian) [1.602, 0.025, -1.634]
            in RPY (degree) [91.811, 1.407, -93.595]
At time 1619635270.101
- Translation: [1.176, -0.081, 0.233]
- Rotation: in Quaternion [-0.498, 0.516, 0.513, -0.471]
            in RPY (radian) [1.600, 0.024, -1.631]
            in RPY (degree) [91.651, 1.398, -93.436]
```

**Figure 4.29**

*Tag_2 Visualization on Rviz*



| | |
|---|---|
| lobal Options | |
| xed Frame | base_link |
| ackground Color | 48; 48; 48 |
| rame Rate | 30 |
| efault Light | ✓ |
| lobal Status: Ok | |
| Fixed Frame | OK |
| rid | ✓ |
| dometry | ✓ |
| F | ✓ |
| aserScan | |
| aserScan | |
| aserScan | ✓ |
| lap | ✓ |
| ose | ✓ |
| mu | |
| oseArray | ✓ |
| lobal_cost | ✓ |
| Status: Warn | |
| opic | /move_base/gl... |
| lpha | 0.7 |
| olor Scheme | costmap |
| raw Behind | ✓ |
| esolution | 0 |
| idth | 0 |
| eight | 0 |
| osition | 0; 0; 0 |
| rientation | 0; 0; 0; 1 |
| nreliable | |
| se Timestamp | |
| olygon | ✓ |

rame
nto which all data is transformed before being ed.

The range of detection is tested with 3 36h11-family-tags, 60mm x 60mm, 80mm x 80mm, and 130mm x 130mm, respectively.

As the result, the 60mm x 60mm can be detected up to 6.6 meters from the camera, whiles 80mm x 80, and 130mm x 130mm is detectable up to 9 meters and 14 meters, respectively.

The tag localization is tested using the same tag size. The graphs below show the accuracy of the tag localization respective to the distance of the tag to the camera.

**Figure 4.30**

*Tag Localization Error on X Axis*

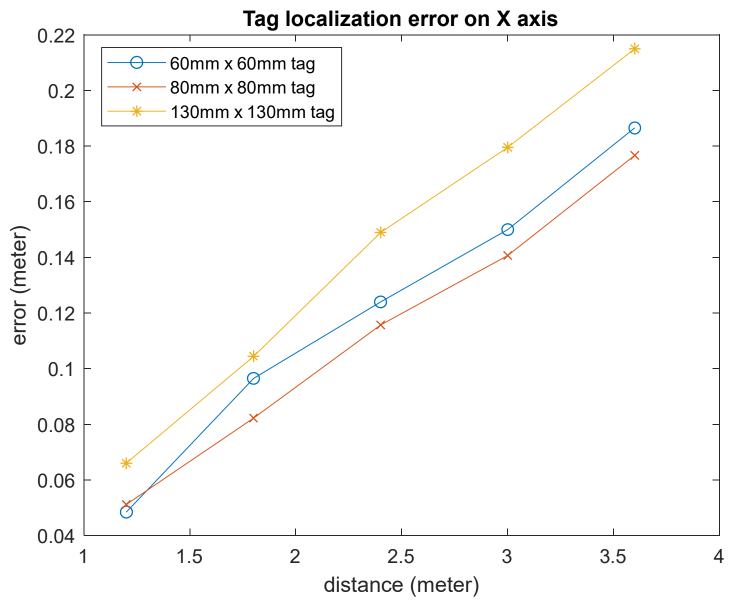


**Figure 4.31**

*Tag Localization Error on Y Axis*

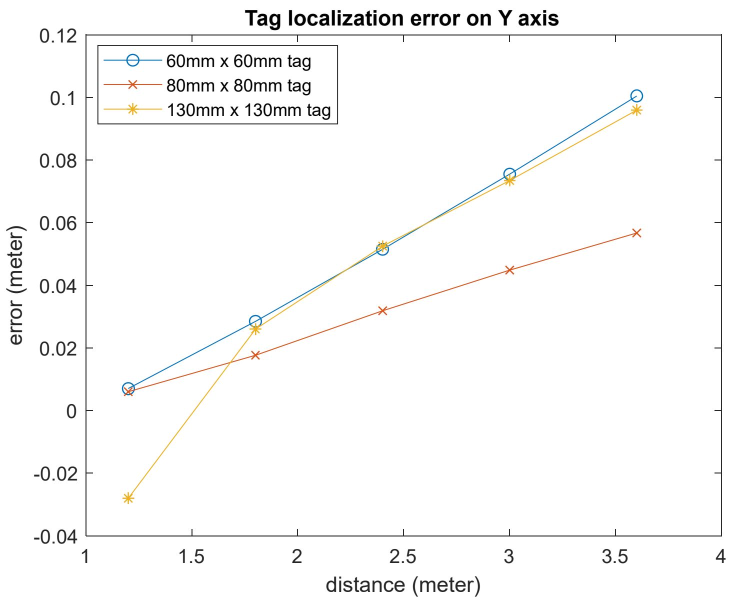
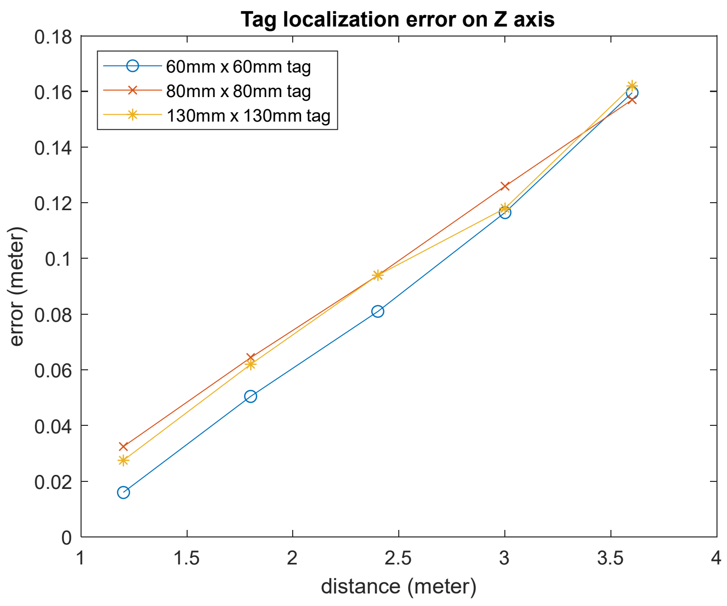


**Figure 4.32**

*Tag Localization Error on Z Axis*



From the graph, the accuracy of localizing each tag size is similar in each axis. It can be seen that the error tends to diverge almost linearly relative to the further the distance in the tested range. The rate of error is calculated to be about 6% regarding to the distance.

Therefore, the further the distance of tags to the camera, the bigger the error. In the range of about 1 to 2 meters apart, the error is between 0.02 meter to 0.06 meter which is acceptable comparing to other error in the system.

## 4.6 Cart Engaging

In the process of moving the cart, there are 3 main factor involves. The first one is the AIV need to be able to navigate to the pick-up place, which it is already tested and proof that it has ability to do so. Second, the ability to identify and locate the cart that have been discussed in previous chapter. The third one is the algorithm to engage the cart to AIV.

The mission is considered as not successful if any of the circumstances below does not meet.

1. Navigate to pick-up goal
2. Find and locate the cart
3. Engage the cart to AIV (Fail if AIV could not engage after 3 tries)
4. Move the cart to the destination
5. Disengage the cart from the AIV.

The experiment is tested for 10 rounds. The cart is placed randomly within the 2 meters radius of the pick-up goal facing into the goal with no wheel's lock.

**Table 4.2**

*First Experiment Testing Result*

| Round | Circumstance | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| 1 | ✓ | ✓ | ✗ | | |
| 2 | ✓ | ✓ | ✗ | | |
| 3 | ✓ | ✓ | ✓ | ✓ | ✓ |
| 4 | ✓ | ✓ | ✗ | | |
| 5 | ✓ | ✓ | ✓ | ✓ | ✓ |
| 6 | ✓ | ✓ | ✗ | | |
| 7 | ✓ | ✓ | ✗ | | |
| 8 | ✓ | ✓ | ✗ | | |
| 9 | ✓ | ✓ | ✓ | ✓ | ✓ |
| 10 | ✓ | ✓ | ✗ | | |

From the result above, majority of the round fails to engage the AIV to the cart. Success rate is 3/10.

On the cart, there is a V shape guide rail to help align the triangle locking block to get the limit switches triggered and raise the locking block. The observed factor that made it fails is the cart move away while the AVI hits and tries to follow the rail.

To solve the problem, the cart's wheels will be locked while the AIV tries to engage to the cart. The design of the cart can then be proposed for further improvement.

Thus, the experiment is conducted again with the cart's wheels locked.

**Table 4.3**

*Second Experiment Testing Result*

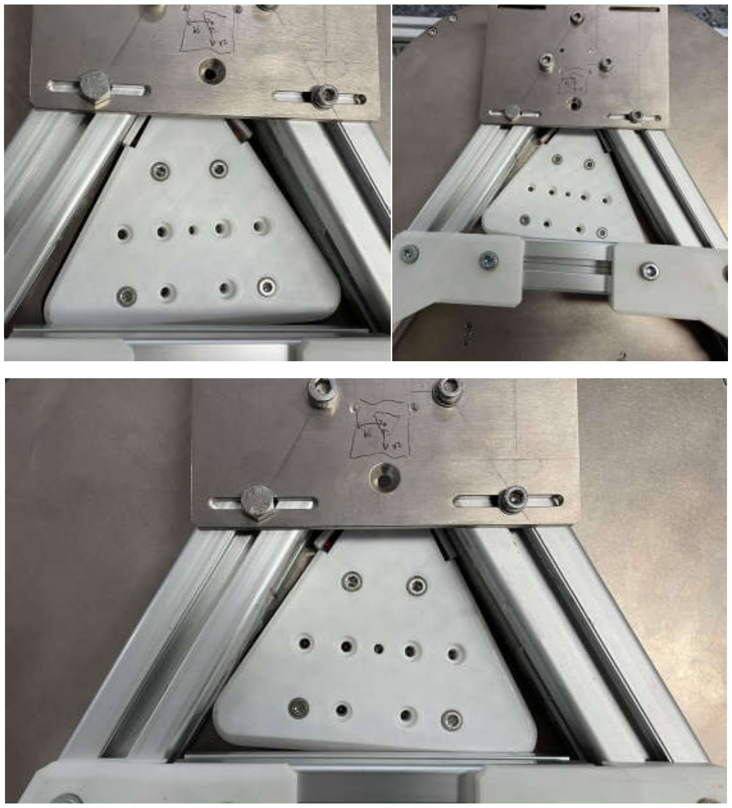
| Round | Circumstance | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| 1 | ✓ | ✓ | ✗ | | |
| 2 | ✓ | ✓ | ✓ | ✓ | ✓ |
| 3 | ✓ | ✓ | ✓ | ✓ | ✓ |
| 4 | ✓ | ✓ | ✗ | | |
| 5 | ✓ | ✓ | ✓ | ✓ | ✓ |
| 6 | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7 | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8 | ✓ | ✓ | ✓ | ✓ | ✓ |
| 9 | ✓ | ✓ | ✗ | | |
| 10 | ✓ | ✓ | ✗ | | |

It can be noticed that the success rate improves from 3/10 to 6/10. However, the 4/10 still fails at cart engaging stage.

The position of the limit switch also has a major impact on the engaging part. Below are some figures show the orientation of the triangle locking block fails to get the limit switch triggered.

**Figure 4.33**

*Orientation That the Limit Switch Fails to Trigger*



# CHAPTER 5
# CONCLUSION AND RECOMMENDATIONS

## 5.1 Conclusion

The control software for the AIV is developed and implemented. The velocity-control motion controller for the AIV is design by first, determine the approximation of the transfer function of the AIV using MATLAB system identification toolbox, then use MALAB Simulink to design and test the PID controller, after that utilize the PID tuner to tune the PID loops. The AIV forward kinematic model is created and be able to control from the remote controller. The odometry of the robot is produced using the wheels' encoders and then fused with IMU data and command velocity input using Extended Kalman Filter. The 2 Laser scanner data is merged to produce a 360-degree scan. The map then can be created using odometry and laser scanner data using SLAM technic called Gmapping. The AIV is localized in the map using Adaptive Monte Carlo algorithm. The ROS navigation stack is implemented enabled the AIV to navigate in the map. ROS navigation stack utilizes global planner (A* and Dijkstra) and local planner Dynamic Window Approach (DWA). The A* algorithm and Dijkstra planned the optimal path that is followed by DWA as local obstacle avoidance.

Moreover, the April tag detection and localization are implemented enabled the AIV to identify and locate the cart. The custom algorithm is designed to get the AVI engaged to the cart and move the cart to its designated goal. The AIV is engaged to the cart by a triangle locking block which use the limit switch as the trigger to raise the locking-block. The position of the trigger-switch affects the engaging ability of the AIV. The design of the cart's wheel affects the engaging ability drastically.

The Table 5.1 shows all the expense of this project.

**Table 5.1**

*Project Expense*

| No | Description | Amount | Unit Price | Price (THB) |
|---|---|---|---|---|
| 1 | Aluminum fabricated part for main structure | 1 set | 75,000 | 75,000 |
| 2 | Motor, gearbox, driver, and wheel | 2 set | 14,000 | 28,000 |
| 3 | Lift motor, gearbox, and driver | 1 set | 14,000 | 14,000 |
| 4 | Nvidia Jetson TX2 | 1 | 25,000 | 25,000 |
| 5 | Hokuyo URG-04LX | 2 | 45,000 | 90,000 |
| 6 | Rotary Encoder | 2 | 1,500 | 3,000 |
| 7 | Sparkfun Razor 9 DoF IMU | 1 | 2,000 | 2,000 |
| 8 | Logotech Camera | 1 | 2,000 | 2,000 |
| 9 | 24V 50Ah Li-ion Battery | 1 | 35,000 | 35,000 |
| 10 | Other | 1 | 25,000 | 25,000 |
|  |  |  | Total | 299,000 |

## 5.2 Recommendations

From the design the robot wheel, in some situation where the floor is quite uneven, the robot's drive wheels is not contacting the floor resulting robot to stuck. The suspension system is recommended, so that the drive wheel is contacting the floor all the time.

The DWA gives good performance in locally planning the path. However, in some situation, the robot stuck while the cost function reaches the local minima. Therefore, it is recommended that there should be an algorithm designed to detect when the DWA cost function stuck in the local minima, so that the planner parameters can be dynamical changed according to the current situation or the planner can be switched to another planner to free the robot.

The camera used for tag detection and localization should be upgraded to a higher resolution to helps improve the accuracy of localizing the tags. The cart's wheels design change is recommended. The wheels should be able to lock itself and unlock automatically when the locking block is raised. Also, the position of the limit switch needs to be placed at place that ensure the maximum contact while robot is engaging. It is going to improve the engaging success rate which also improves the system overall.

Considering the implementation of the robot where there are many robots operating, the fleet management, and logistics management need to be implemented to share the work and managing the rout of the robots.

# REFERENCES

Allen, W. (2020). *Guide to warehouse robots: types of warehouse robots, uses & more - 6 River Systems*. https://6river.com/guide-to-warehouse-robots/

Bahrin, M. A. K., Othman, M. F., Azli, M. F., & Talib, M. F. (2016). *INDUSTRY 4.0: A REVIEW ON INDUSTRIAL AUTOMATION AND ROBOTIC*.

Burgard, W., Bennewitz, M., Tipaldi, D., & Spinello, L. (2014). *Introduction to Mobile Robotics SLAM : Simultaneous Localization and Mapping*.

Chan, S. H., Wu, P. T., & Fu, L. C. (2019). Robust 2D Indoor Localization Through Laser SLAM and Visual SLAM Fusion. *Proceedings - 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018*, 1263–1268. https://doi.org/10.1109/SMC.2018.00221

Cheng, Z., & Wang, G. (2018). Real-Time RGB-D SLAM with Points and Lines. *Proceedings of 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, IMCEC 2018*, *Imcec*, 119–122. https://doi.org/10.1109/IMCEC.2018.8469424

Fiorini, P., & Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, *17*(7), 760–772. https://doi.org/10.1177/027836499801700706

Huang, J. T., Hu, J. Y., Lo, J. W., & Cai, Q. D. (2018). Development and design of AIV using hub motor embedded in mecanum wheel. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, *2018-July*, 658–663. https://doi.org/10.1109/AIM.2018.8452381

Jensfelt, P. (2001). Approches to Mobile robot localization in indoor navigation. In *Royal Institute of Technology* (Vol. 23). http://nyx-www.informatik.uni-bremen.de/347/1/jensfelt_thesis_01.pdf

Krogius, M., Haggenmiller, A., & Olson, E. (2019). Flexible Layouts for Fiducial Tags. *IEEE International Conference on Intelligent Robots and Systems*, 1898–1903. https://doi.org/10.1109/IROS40897.2019.8967787

Li, M., & Zhang, C. (2018). A spatial pose measurement scheme by using IMU and AprilTag Technologies. *2018 IEEE International Conference on Information and Automation, ICIA 2018*, *August*, 1048–1052. https://doi.org/10.1109/ICInfA.2018.8812438

Malu, S. K., & Majumdar, J. (2014). Kinematics, Localization and Control of Differential Drive Mobile Robot. *Global Journal of Researches in Engineering*, *14*(1).

Olson, E. (2011). AprilTag: A robust and flexible visual fiducial system. *Proceedings - IEEE International Conference on Robotics and Automation*, 3400–3407. https://doi.org/10.1109/ICRA.2011.5979561

omega.co.uk. (2020). *PID Controller: Types, What It Is & How It Works | Omega*. https://www.omega.co.uk/prodinfo/pid-controllers.html

ROSWiki, & Brian, G. (2020a). *amcl - ROS Wiki*. http://wiki.ros.org/amcl

ROSWiki, & Brian, G. (2020b). *Gmapping - ROS Wiki*. http://wiki.ros.org/gmapping

ROSWiki, & David V., L. (2020). *global_planner - ROS Wiki.* http://wiki.ros.org/global_planner

Shashika, D. C. (2018). *Localization and Navigation Control of an Autonomous Intelligent Vehicle ( AIV )* (Issue May 2018). Asian Institute of Technology.

Siegwart, R., Nourbakhsh, I. ., & Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots* (2nd ed.). Intelligent robotics and autonomous agents.

Zhang, F., Li, S., Yuan, S., Sun, E., & Zhao, L. (2018). Algorithms analysis of mobile robot SLAM based on Kalman and particle filter. *Proceedings of 2017 9th International Conference On Modelling, Identification and Control, ICMIC 2017*, *2018-March*(Icmic), 1050–1055. https://doi.org/10.1109/ICMIC.2017.8321612