# Reinforcement Learning Based Stabilization of Liquid Surface in Ground Vehicle payloads

Submitted in partial fulfilment of the requirements

of the degree of

Bachelor of Technology

by

**KSHITIJ BITHEL,** 18115053
**KESHAV DIXIT,** 18115050

Supervisor(s)

**Prof. Sohom Chakrabarty**

Department of Electrical Engineering

Indian Institute of Technology Roorkee

2022

## Declaration

I hereby declare that the work which is presented here, entitled **Reinforcement Learning Based Stabilization of Liquid Surface in Ground Vehicle payloads**, submitted in partial fulfilment of the requirements for the award of the Degree of **Bachelor of Technology** in the Department of Electrical Engineering, Indian Institute of Technology Roorkee. I also declare that I have been doing my work from Month Year under the supervision and guidance of **Prof. Sohom Chakrabarty, Assistant Professor, Electrical Engineering Department, Indian Institute of Technology Roorkee**. The matter presented in this dissertation report has not been submitted by me for award of any other degree of institute or any other institutes.

Date:

KSHITIJ BITHEL (18115053) KESHAV DIXIT (18115050)

## Certificate

This is to certify that the above statement made by the candidate is true to the best of my knowledge and belief.

[illegible]

Digitally signed by Sohom Chakrabarty
DN: cn=Sohom Chakrabarty, o=IIT Roorkee, ou=Electrical, email=c.sohom@gmail.com, c=IN
Date: 2022.04.28 22:14:05 +05'30'

Signature
Prof. SOHOM CHAKRABARTY
Assistant Professor
Department of Electrical Engineering
Indian Institute of Technology Roorkee

# Acknowledgement

We would like to take up this opportunity to express our sincere gratitude to our project guide Prof Sohom Chakrabarty, Assistant Professor, Department of Electrical Engineering, IIT Roorkee for his constant support and guidance throughout the course of this project, without which it would not have been possible for us to complete this project.

We also extend our gratitude  to Dr. G. N. Pillai, Professor and Head of Department, Electrical Engineering, IIT Roorkee for constantly motivating us. We would also like to thank Prof. Sharmili Das, Assistant Professor and Prof. Deep Kiran, Assistant Professor and Coordinators of B. Tech Project for letting us pursue this topic for our project.

Further we would also like to thank our department for allowing us to work in the Advance Robotics Laboratory to perform our work and build an arena for the prototype.

We would express our gratitude to our senior Mr. Ashish Shakya for his constant motivation and guidance throughout the course of this project.

Last but not the least, we would like to acknowledge the support of ARTPARK@IISc Bangalore for constantly providing us with the necessary support and equipments to carry on this project.

## Abstract

Sloshing refers to the motion of the free liquid surface inside its container. It is a complex nonlinear dynamical phenomenon that has a substantial impact on the fluid system's stability. It affects various engineering systems and processes such as liquid storage tanks, liquid rocket fuel tanks, molten metal handling in steel plants, robotic handling of liquids, etc. We aim to solve the problem of minimizing slosh in Automated Ground Vehicle (AGV) payloads, i.e, stabilize the free surface of a liquid inside a container placed as payload on an AGV, while the AGV traverses along specified paths in a 2-D plane. For this purpose, a Deep Reinforcement Learning (DRL) framework will be designed to tune a robust controller to control the prototype AGV and move it to a destination point along desired 2-D paths while minimizing the slosh of the payload liquid as well as minimizing the time taken to reach the destination.

# Table of Contents

## List of Figures

## List of Abbreviations

| | |
|---|---|
| PID | : Proportional, Integral, Derivative |
| LIDAR | : Light Detection and Ranging |
| I2C | : Inter-Integrated Circuit |
| ARM | : Advanced RISC Machines |
| RISC | : Reduced Instruction Set Computer |
| USB | : Universal Serial Bus |
| FPU | : Floating Point Unit |
| GPU | : Graphical Processing unit |
| LPDDR | : Low-Power Double Data Rate |
| PWM | : Pulse Width Modulated |
| AGV | : Automated Guided Vehicle |
| SiP | : System in Package |
| UART | : Universal Asynchronous Reciever-Transmitter |
| DRL | : Deep Reinforcement Learning |
| STC | : Super-Twisted Control |
| RL | : Reinforcement Learning |
| SMC | : Sliding Mode Control |
| SOSM | : Second Order Sliding Mode |
| UAV | : Unmanned Aerial Vehicle |
| DQN | : Deep Q-Networks |
| TRPO | : Trust Region Policy Optimization |
| DDPG | : Deep Deterministic Policy Gradient |
| PPO | : Proximal Policy Optimization |
| SAC | : Soft Actor Critic |
| NFQCA | : Neural Fitted Q Iteration with Continuous Actions |
| FFT | : Fast Fourier Transform |

# Chapter 1

# Introduction

During the translational or rotational accelerations of liquid containers, a substantial volume of liquid tends to move unrestrained in the containers. The motion of the free liquid surface inside its container in response to the force applied to the liquid directly or indirectly is called Slosh.. The motion of the liquid occurs in different forms based on the nature of the applied force, its container geometry, etc. Accordingly, there exist various sloshing phenomena [1,2], viz., lateral, rotational, swirling, or even chaotic, quasi-periodic. Sloshing is an unwanted phenomenon as it can produce additional forces and moments which affect performance. The sloshing problem frequently occurs in partially filled containers in a variety of applications, in packaging industry it can lead to improper sealing, thereby decreasing the shelf life of the product, in liquid cargo carriers it can cause dangerous overturns, in rockets and long range missiles sloshing can cause additional accelerations which have to be taken care of by guidance and control system [3]. The impact of liquid sloshing is therefore severe.

Hence, it is essential to analyze and precisely characterize the sloshing phenomenon, as well as to establish, identify, and experimentally evaluate mathematical models of slosh that may be employed to control development.

Modelling of slosh has been tried upon for a long time, a nonlinear and complicated mathematical model can be utilized to represent the sloshing dynamics [4], but such dynamics becomes too challenging for designing the controller. This necessitates the development of simpler mathematical models for slosh in order to save computational time and expense while providing controllable models. To represent the sloshing phenomenon, spring-mass damper and pendulum models are commonly used (Fig 1.1) [5]. Moving mass in these models is used to represent the sloshing mass of the liquid. We have also used the pendulum model to model our system and implement robust control techniques for controlling the slosh.

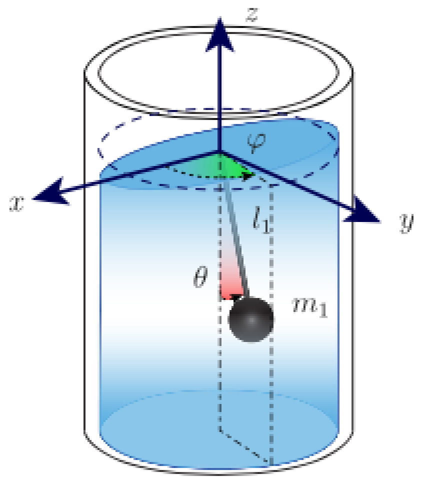


Fig1.1 Sloshing dynamics modelling using spherical pendulum

Many scientists have sought to find solutions to the difficult issues that sloshing dynamics pose. Different passive control techniques like baffles (Fig 1.2) [6,7,8] are reported to control the sloshing effects in launch vehicles specifically and in other applications alike. However, it increases the system's weight and, as a result, the cost, making it less desirable. Researchers have been increasingly interested in active control solutions for slosh suppression over the last two decades and various control techniques using approximate models have been implemented like Sliding Mode Control[3,9,10,11], PID [12], Input Shaping [13,14] and Lyapunov-based feedback control[15].

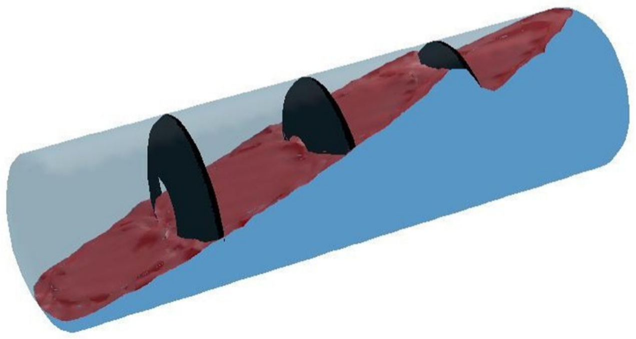


Fig 1.2 Passive Slosh Control using anti-wave Baffles

In this project, we have developed an Automated Ground Vehicle(AGV) prototype which has a holonomic drive, is capable to localise itself and navigate in a 2d Arena, has the ability to measure the slosh of the liquid placed as a payload and minimise it while navigating from one point to other using a Deep Reinforcement Learning Agent in combination with robust Sliding Mode Control. The final prototype is shown in Fig 1.3.

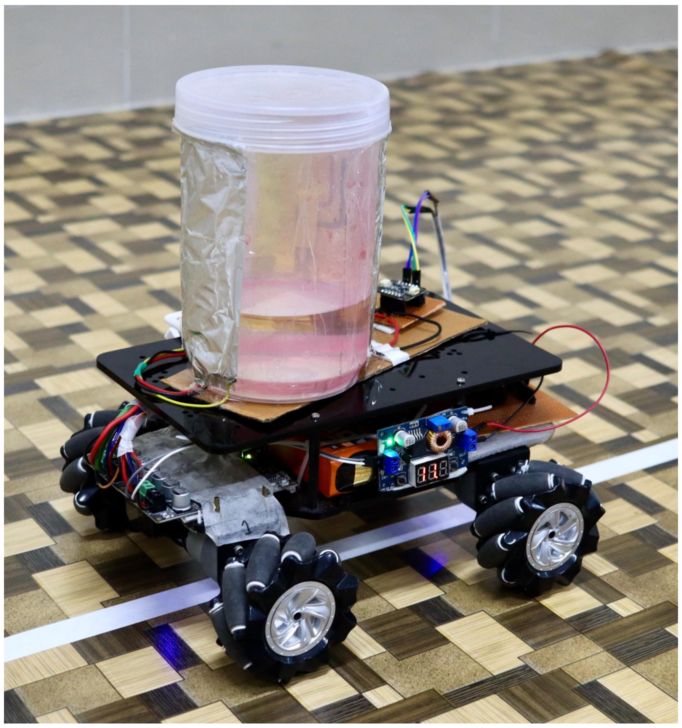


Fig 1.3 Final Prototype

# Chapter 2

# Hardware Prototype

An automated Guided Vehicle (AGV) prototype has been built which is capable to move holonomically in 2-D space and localise itself within its surroundings. It also can measure 2-D Slosh, which is needed as the feature for the Reinforcement Learning model. Different Hardware components are used, which in union provide a hardware prototype capable of deploying complex Control Algorithms and Intelligent Logic.

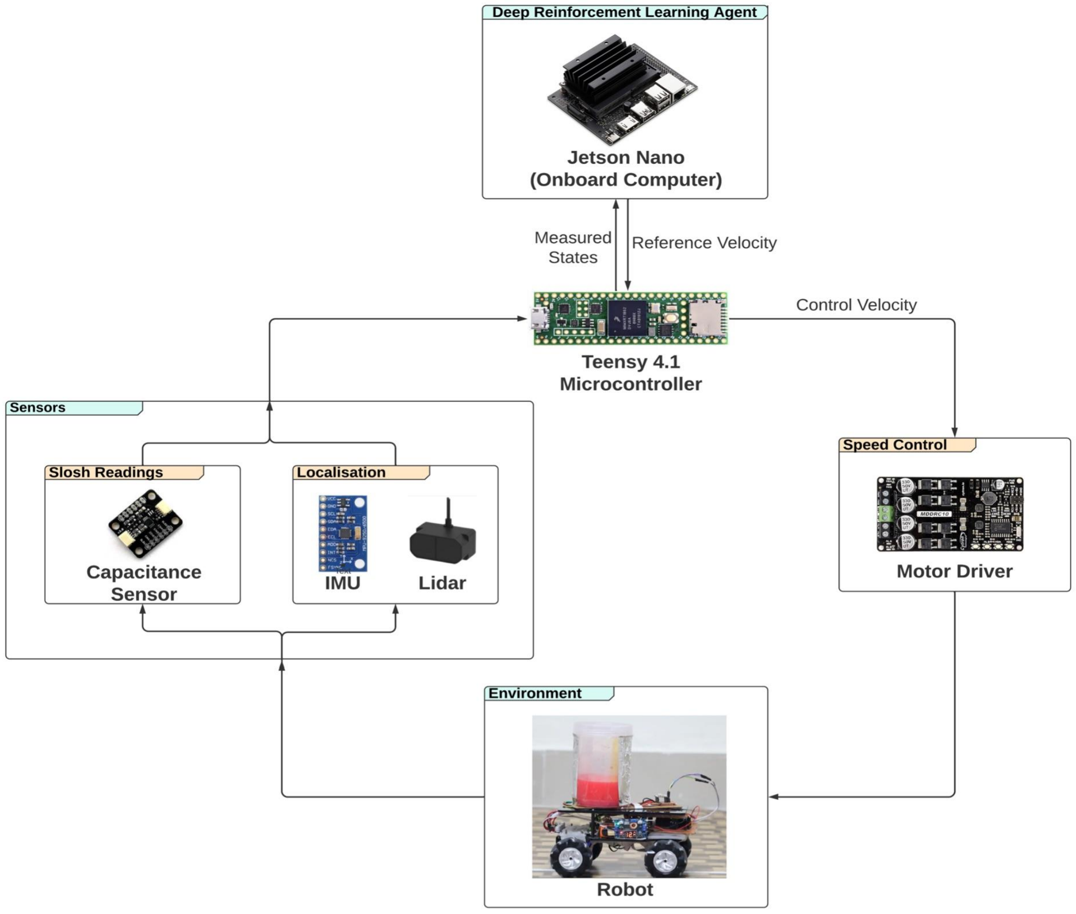


Fig 2.1 Signal flow diagram

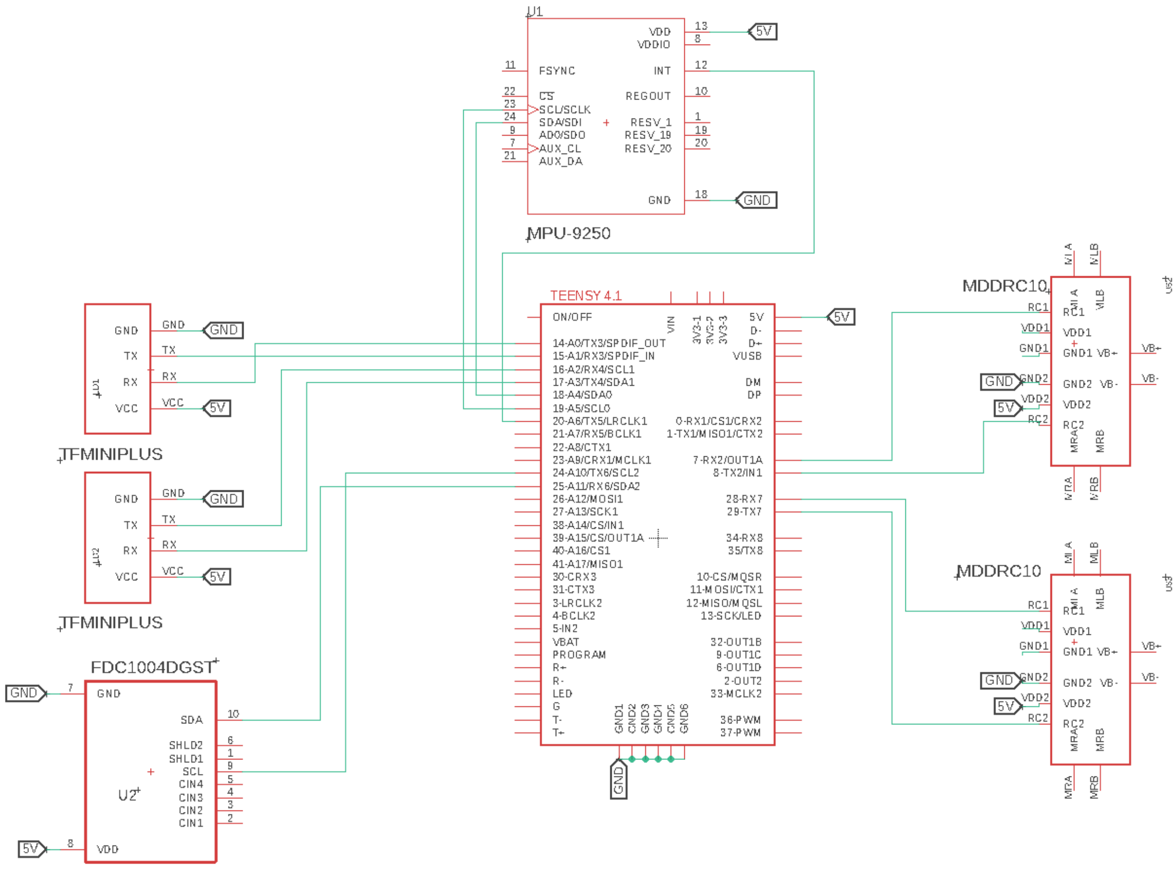


Fig 2.2 Overall schematic of the AGV

## 2.1 Mechanical Design

### 2.1.1 Chassis

The kit includes dual 5mm thick acrylic sheets. It is a 4 Wheel Drive robot chassis with Mecanum wheels which increase the maneuverability of the robot . The wheels have rollers inclined at 45º which move independently and allow the robot to move in any direction without changing its orientation or spin in place. The motors have rotary encoders so it can be used for velocity feedback in order to maintain a fixed angular velocity of the wheels.

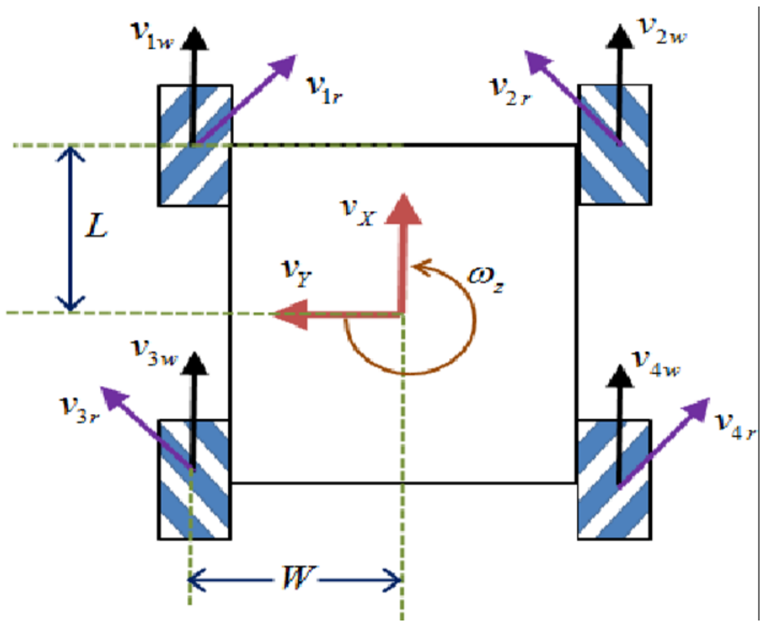


Fig 2.3 Force vectors on the Mecanum wheels

The force acting on a mecanum wheel is 45 degrees from the direction of the wheel. Hence different combinations of the rotations of the wheels results in a velocity vector at different angles. For instance if all the wheels move in the forward direction like in a case shown above, there will be two components in the force vector of each wheel. In that case the y-component of all the wheels has a counter pair thus nullifying its effect whereas the forces in the x-direction add up thus providing a movement along x-direction.

### 2.1.2 Motors

We have used four high torque geared motors (GB37-520) for the bot. The rated power of the motor is nearly equal to 7.2W, with a rated speed of 330 RPM. All four motors are fitted with a semi absolute encoder. The encoder provides maximum accuracy of 1320 CPR. In all the system provides a payload capacity of 15 Kgs.

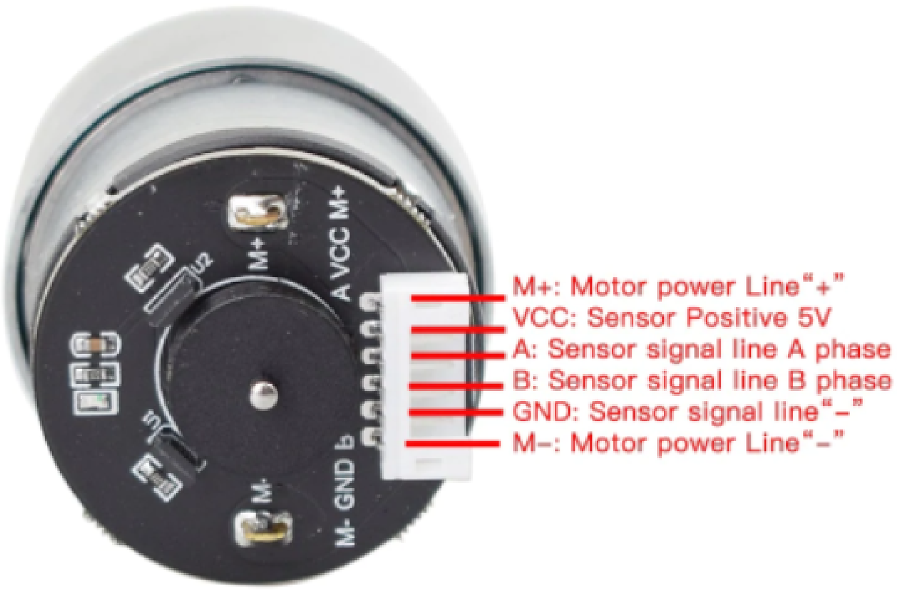
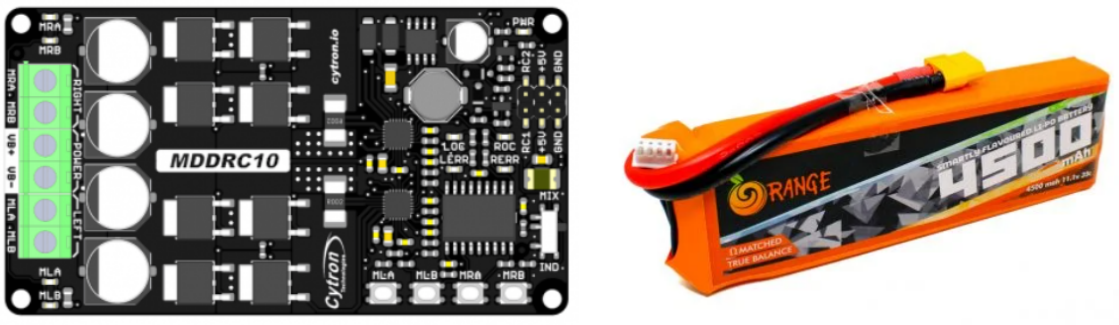

M+: Motor power Line"+"
VCC: Sensor Positive 5V
A: Sensor signal line A phase
B: Sensor signal line B phase
GND: Sensor signal line"–"
M–: Motor power Line"–"

Fig 2.4 Motor with Encoder

## 2.1.3 Power Circuit

The Motors are driven by 2 Cytron 10 Amp-30V DC motor drivers. The motor driver also offers overcurrent protection and temperature protection. The speed signal is provided as a PWM signal to the Motor Driver, generating output voltage for the motors. The range of the PWM signal is from 0 to 255, the initialization signal is treated as the zero signal, and the motor output is mapped accordingly.



(a) (b)

Fig 2.5 (a) Cytron Motor Driver, (b) Li-Po battery 4500 mah

The AGV is powered by a 12V 3S Lithium polymer battery with a capacity of 4500 mAh, and a maximum continuous current rating of 35C. We are also using the same battery to power the arduino as well by stepping it down to a regulated 5 volts DC through a Buck converter.

## 2.2 Electronics Design

### 2.2.1 Jetson Nano

The Jetson Nano developer kit (Fig 2.2) is a small powerful computer that has the power to run multiple neural networks in parallel for applications such as image classification, object recognition, segmentation, and speech recognition. It enables projects to incorporate artificial intelligence algorithms for practical applications.

Its key features are a 128-core NVIDIA Maxwell GPU. Quad-core ARM A57 CPU. 4 GB 64-bit LPDDR4. We are using it to deploy the Formulated RL agent in order to learn the parameters from the different experiences it gains throughout the learning process.

Fig 2.6 Nvidia Jetson Nano developer kit

## 2.2.2 Teensy 4.1

The Teensy 4.1 is a microcontroller offered by a popular development platform that features an ARM Cortex-M7 processor at 600MHz, with an NXP iMXRT1062 chip, 7536K of Flash Memory to store the code. The Teensy 4.1 is a 40 pin chip which comes with an excellent capability of I/O, including an Ethernet PHY, SD card socket and a USB Host port.

The processor of Teensy 4.1 includes a floating-point unit (FPU), which supports both 64-bit “double” and 34-bit “float”. This helps in hardware-accelerated calculations of the double functions like log(), sin(), cos().

The main advantages of Teensy 4.1 which motivated us to use it instead of the more popular Arduino Mega ADK are the small form factor which saves us a lot of space, the Faster clock which results in fast execution of the Instructions, and the Interrupt capability on all the digital pins compared to only 4 present on the Arduino Mega

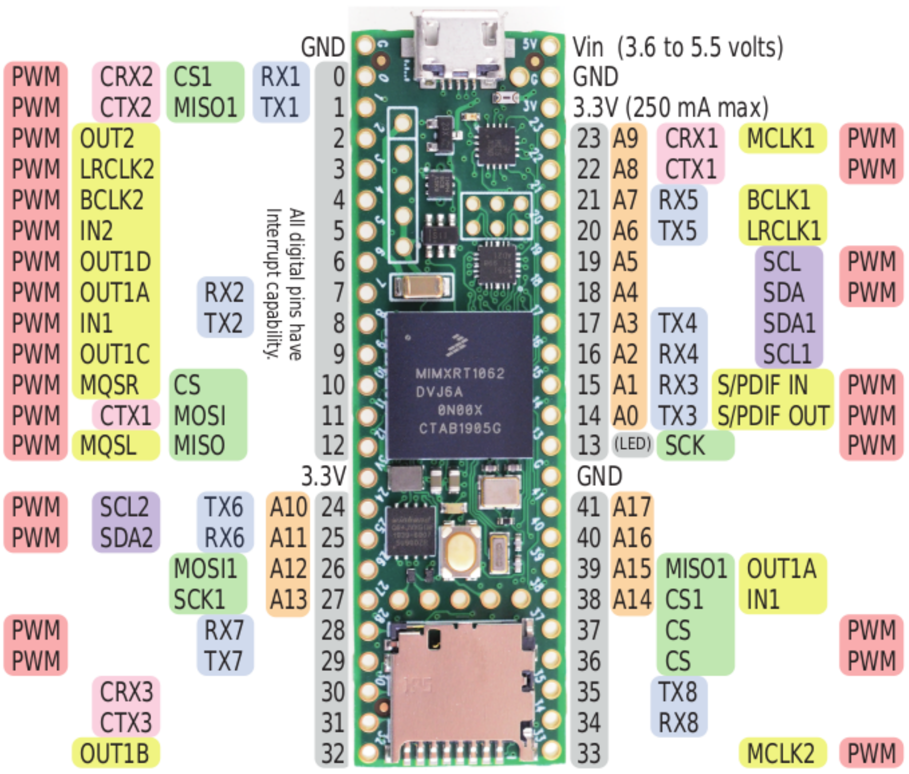
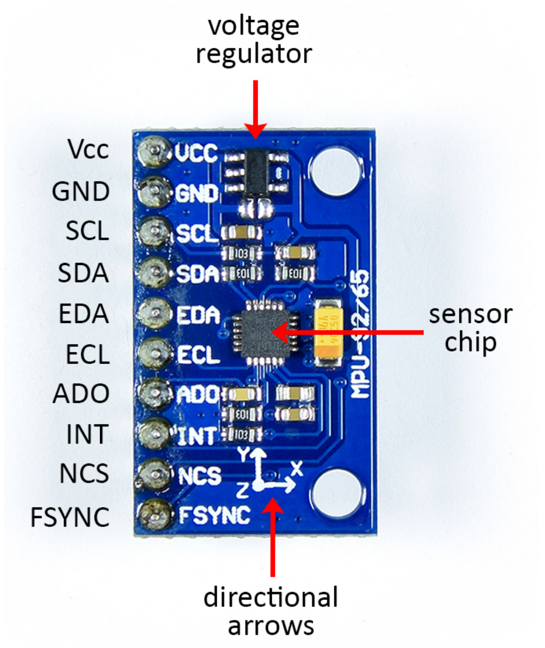


Fig 2.7 Teensy 4.1 Pinout

### 2.2.3 Localization and Orientation sensors

The MPU-9250 is a system in package (SiP) that combines two chips: the MPU-6050, which includes a 3-axis gyroscope, a 3-axis accelerometer, and an onboard Digital Motion Processor capable of processing complex Motion Fusion algorithms; and the AK8963, which is a 3-axis digital compass in a small 3x3x1mm package. It uses the I2C address, which is 0x68 by default and 0x69 if AD0 is pulled high. It also features an inbuilt Temperature Sensor, which is used to compensate for reading inaccuracies caused by temperature changes.



(a)

(b)

Fig 2.8 (a) MPU-9250 (b) TF Mini Plus LIDAR

For localization we are using two TF mini plus lidars. They are single point lidars with a range from 0.1m~12m and an accuracy of 1%, frame rate from 1Hz to 1kHz. It has been designed with an IP65 rating making it resistant to dust and water. It works on an operating voltage of 5 Volts, and communicates with the processor using UART communication protocol.

# Chapter 3

# Navigation

The AGV developed can navigate in a 2D Arena using localisation and control techniques described below. The navigation subsystem consists of wheel drive, localisation using sensor fusion, and control of the robot which are explained in detail below.

## 3.1. Mecanum Wheel Drive

Mecanum wheel, a kind of omnidirectional wheel, is widely used nowadays in mobile robotics[put recent references [16,17,18]. It is a conventional wheel with a series of rollers attached to its circumference, and these rollers have an axis of rotation at 45° to the plane of the wheel in a plane parallel to the axis of rotation of the wheel, as shown by the angle **α** in Fig 3.1.

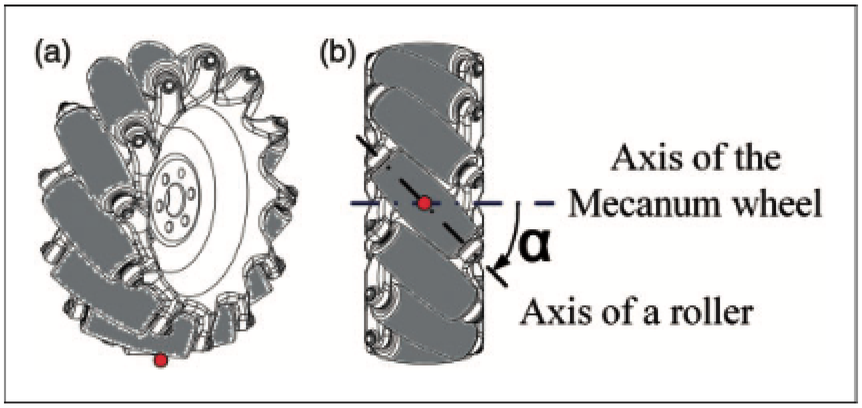


Fig 3.1 Mecanum wheel. (a) Side view and (b) bottom view [Refa]

We have used a Type-X arrangement for our AGV due to its better stiffness and dexterity over Type-O arrangement which makes it achieve more accurate and stable omni directional movement compared to type-O[19].

### 3.1.1 Kinematic Model Of 4 wheel Mecanum Drive Robot

The Kinematic model of the robot allows us to relate the individual wheel's properties to the whole robot's properties. Fig 3.2 shows the configuration of the robot with 4 mecanum wheels.

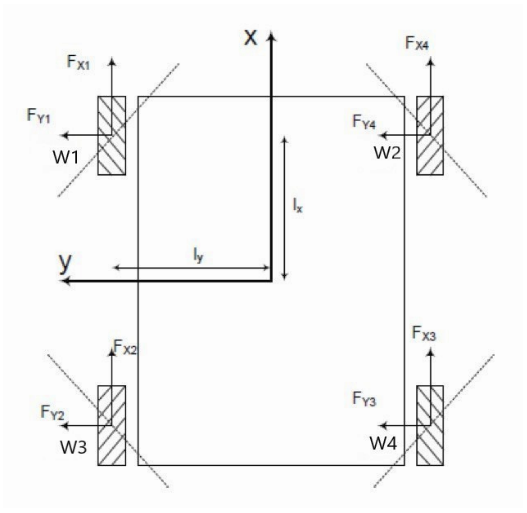


Fig 3.2 Wheels Configuration as seen from Top

The inverse kinematic equations for the robot allow us to calculate the individual wheel's angular velocity as a function of the robot's velocity. The equations are

$$\omega 1 = \frac{1}{r}(v_x - v_y - (l_x + l_y)\omega) \tag{3.1}$$

$$\omega 2 = \frac{1}{r}(v_x + v_y + (l_x + l_y)\omega) \tag{3.2}$$

$$\omega 3 = \frac{1}{r}(v_x + v_y - (l_x + l_y)\omega) \tag{3.3}$$

$$\omega 4 = \frac{1}{r}(v_x - v_y + (l_x + l_y)\omega) \tag{3.4}$$

Here, $w_i$ : angular velocity of ith wheel,

$r$ : radius of wheel,

$v_x$ ,$v_y$ : robot's linear velocity in the reference frame shown in Fig 3.2

ω : robot's angular velocity in the reference frame shown in Fig 3.2

$l_x$ : half of the distance between front and rear wheels and

$l_y$ : half of the distance between front wheels.

For detailed Kinematic analysis, refer to [20].

## 3.2. Localisation

For Localisation, a sensor fusion algorithm consisting of 2 single point Lidars and an Inertial Measurement Unit is used. Fig 3.3 shows the robot placed in the arena.

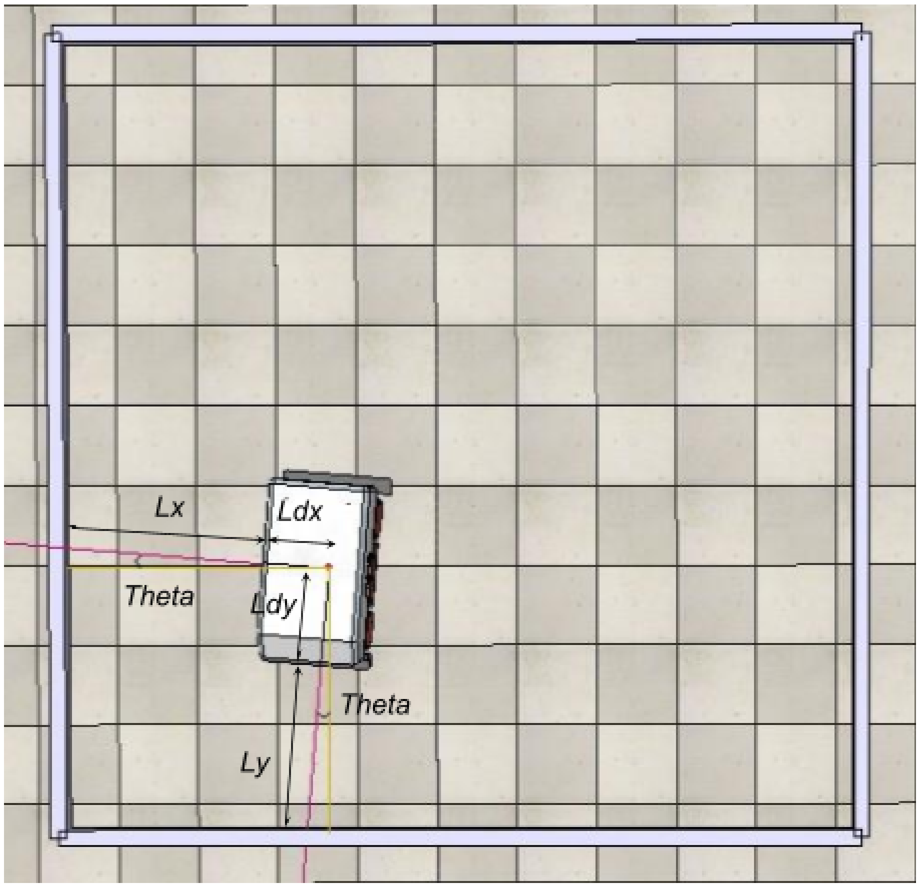


Fig 3.3 AGV localizing in an Arena

The pose of the robot is estimated by

$$x = (Ldx + Lx) * cos(\theta) \tag{3.5}$$

$$y = (Ldy + Ly) * cos(\theta) \tag{3.6}$$

where, $(x,y)$ : Coordinates of robot

$\theta$ : Yaw Angle of robot

$Lx$ : Output of Lidar in x direction

$Ly$ : Output of Lidar in y direction

$Ldx$ : distance between center of robot and lidar in x direction

$Ldy$ : distance between center of robot and lidar in y direction

## 3.3. Controller for Path Tracking

Path Tracking of the robot is done using a combination of 2 controllers : Orientation and Drift Controller.

### 3.3.1 Orientation Control

The orientation of the robot is controlled using a PD Controller whose control diagram is shown in Fig 3.4

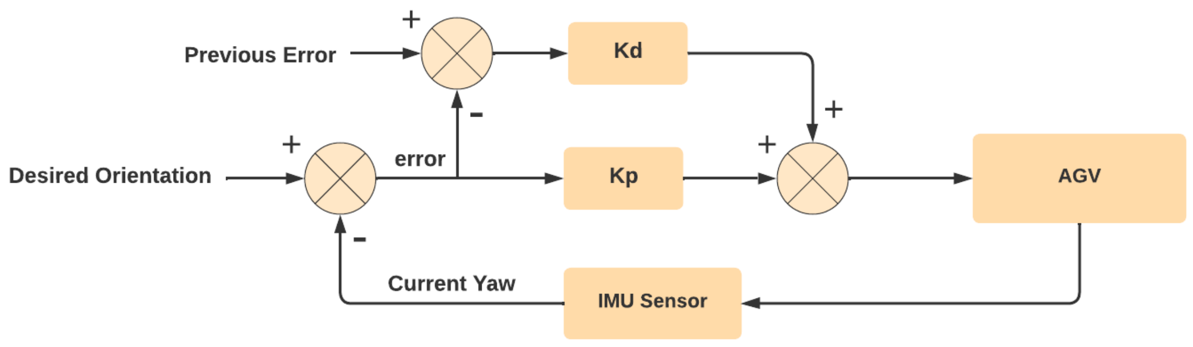


Fig 3.4 Orientation Controller

### 3.3.2 Drift Control

The drift control is applied along the x axis, thereby controlling the x coordinate of the robot. It is formed using a PD Controller whose Control Diagram is shown in Fig 3.5.

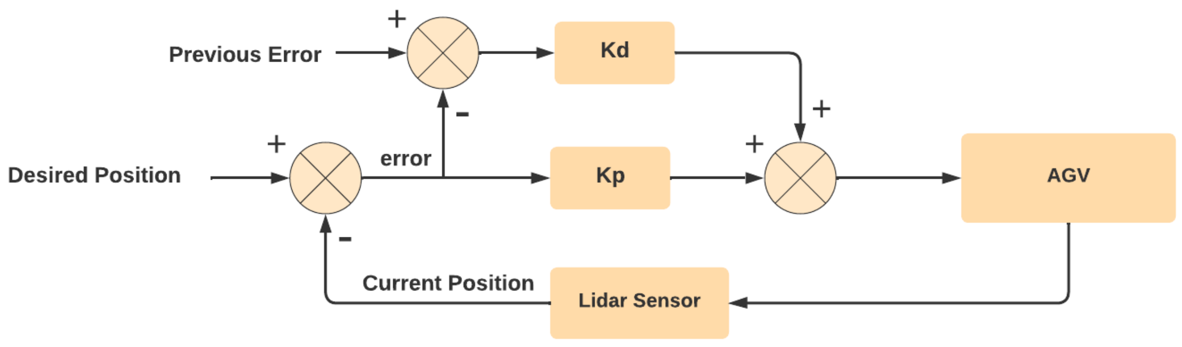
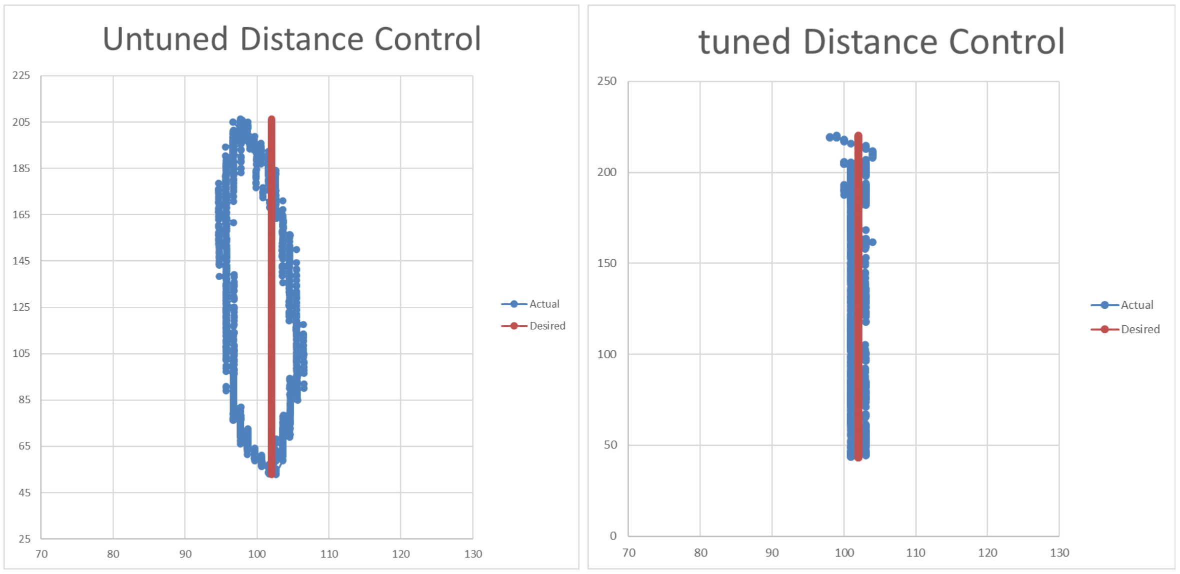


Fig 3.5 Drift Controller

By combining both the controllers, we can achieve path tracking on any predefined 2D path. We tested the controller on straight line(Fig 3.5) and sinusoidal path(Fig 3.6) and observed that the closed loop controller performs very well with mean squared error less than 1cm in straight line path and of 2.71 cm in sinusoidal trajectory.



(a) (b)

Fig 3.6 Path Tracking for Straight line (a) Untuned (b) Tuned

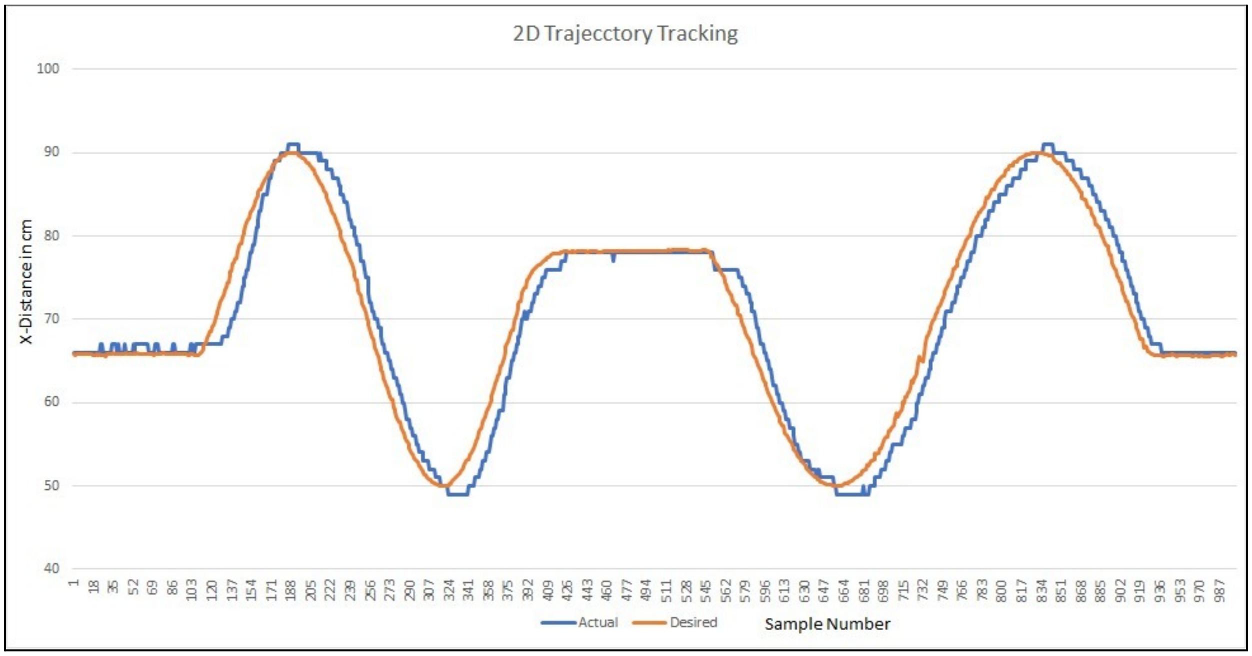


Fig 3.7 Path Tracking for sinusoidal path

# Chapter 4

# Velocity Control

We have developed an DRL based Super Twisting Controller that minimises the slosh in a given trajectory and minimises the time taken to cover that given path. We are restricting the AGV to move in a straight line for initial training, thus reducing a dimension of control. As input, the controller takes the value of slosh and returns the value of the desired velocity of the AGV. Now the AGV must enforce this velocity. For that, we have developed a velocity control loop to implement the velocity commands from the controller.

## 4.1 Controller for Velocity Control

We tried two different Algorithms for the velocity control, compared the results we got from the two, and then used the one that gave better results. The Algorithms, results and observations have been discussed below.

### 4.1.1 PID Controller

A Proportional Integral Derivative controller or PID controller is a feedback based control mechanism which has been used widely for a long time. It provides the control output based on the characteristics of error and has 3 tuning parameter.[21]. It calculates the error value e(t) using the desired setpoint and a measure of the process variable and calculates the control output based on the proportional, integral and derivative terms. The Mathematical equation for the same in the continuous time domain is given by

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \tag{4.1}$$

But we are dealing with microprocessors, so the computation is discrete. In that case the mathematical equation is given by

$$u(n) = K_p e(n) + K_i \sum_{0}^{n} e(\tau)\Delta t + K_d \frac{(e(n)-e(n-1))}{\Delta t} \tag{4.2}$$

For our controller the reference input u(t) is the desired velocity given by the DRL-STC agent. It takes the velocity feedback by differentiating the LIDAR distance values thus generating the error for the controller. The output of this controller is the PWM values to be fed to the Motor Driver.

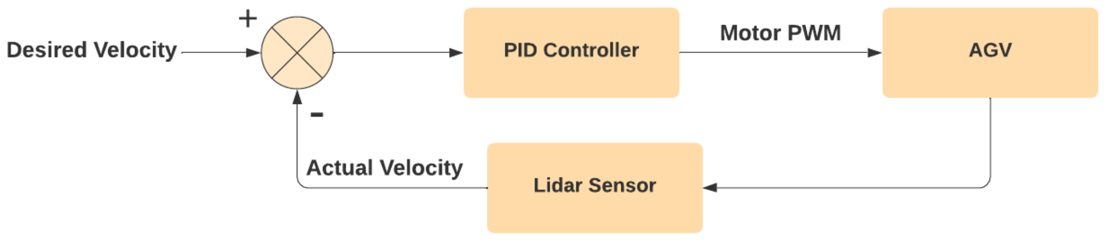


Fig 4.1 PID controller for velocity control

Even after a lot of tuning the AGV was not able to follow the desired velocity very accurately and that too with a lot of oscillations.

For a better performance we used the velocity algorithm of PID control. In this algorithm instead of using the P, I, D gains to calculate the PWM output of the controller we use the PID gains to calculate the increment in the PWM and that increment is added to the PWM which we had previously.

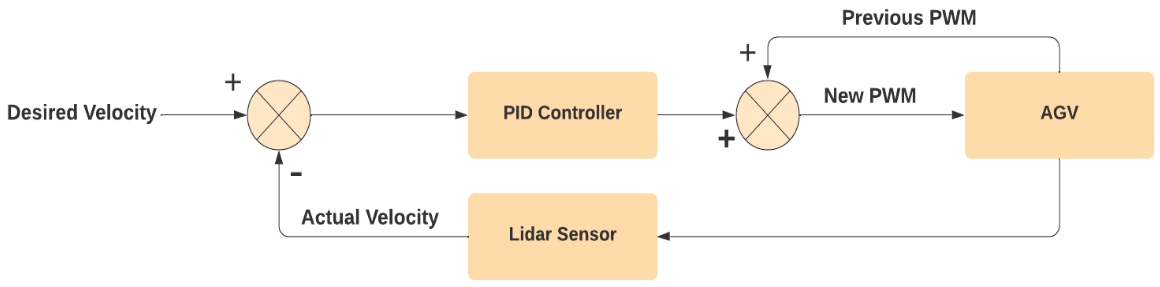


Fig 4.2 PID controller for velocity control with PWM feedback

$$u(n) = u(n-1) + K_p e(n) + K_i \sum_{0}^{n} e(\tau)\Delta t + K_d \frac{(e(n)-e(n-1))}{\Delta t} \tag{4.3}$$

Here also the e(n) denotes the error in the reference speed and the actual speed of the AGV.

This controller gave better results than before but was still away from the expected observations. This reduced the Bumpiness in the velicity and but still the actual velocity was not able to trace the desired speed curve and became unstable at times.

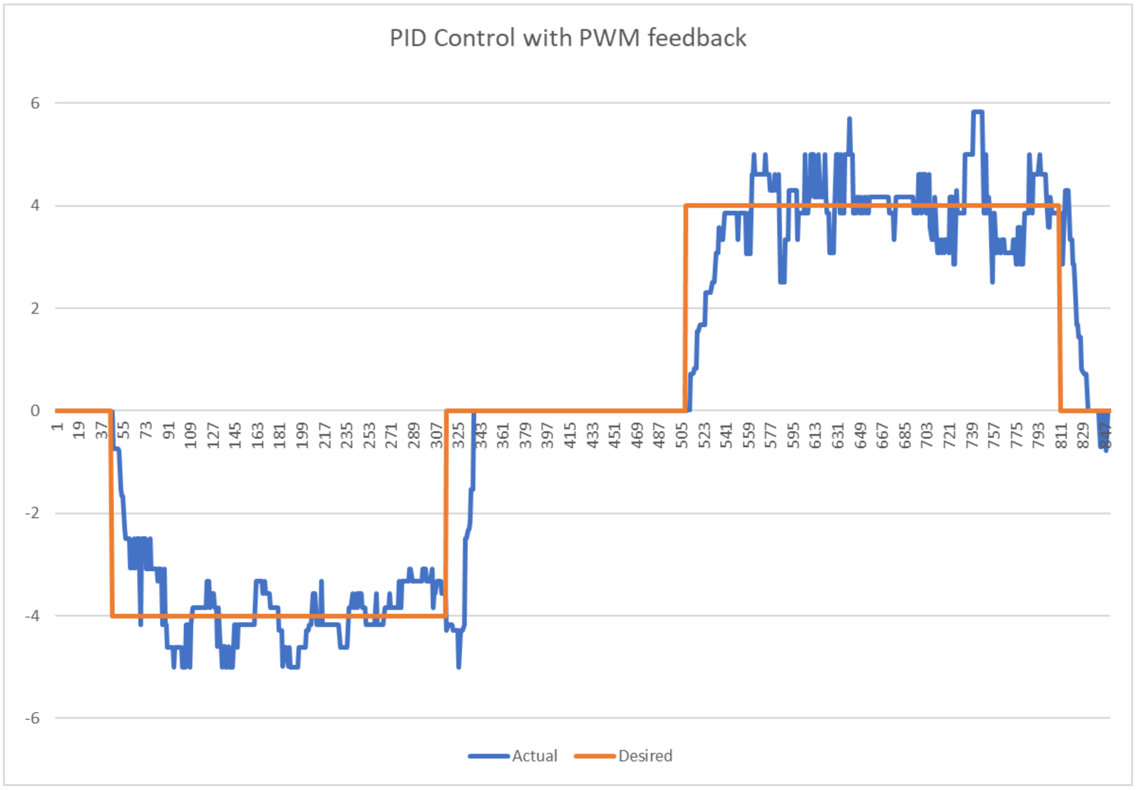


Fig 4.3 Desired velocity vs Actual Velocity curve for PID controller

## 4.1.2 Fuzzy Controller

In contrast to classical or digital logic, which operates on discrete values of 1 or 0, a fuzzy control system analyses analogue input values in terms of logical variables that take on continuous values between 0 and 1. (true or false, respectively).[22,23]

We have used a pre-made Arduino library[24] which has been developed with the purpose of providing an easy way to create fuzzy sets and implementing fuzzy logic for controlling the navigation of a four wheeled mecanum drive robot.

The library has the following properties:

1. Automatically generated logic.
2. Option to choose type of membership functions (Gaussian or Triangular).
3. Freedom to define discrete sets with varying standard deviation (or size in case of triangular) and centres.
4. Weighted average and Centroid of area techniques for defuzzification are available .
5. Centroid of the area can only be implemented for triangular sets and not for Gaussian.

After some time tuning the Fuzzy Controller parameters, we get a decent looking curve with minor delay and oscillations.

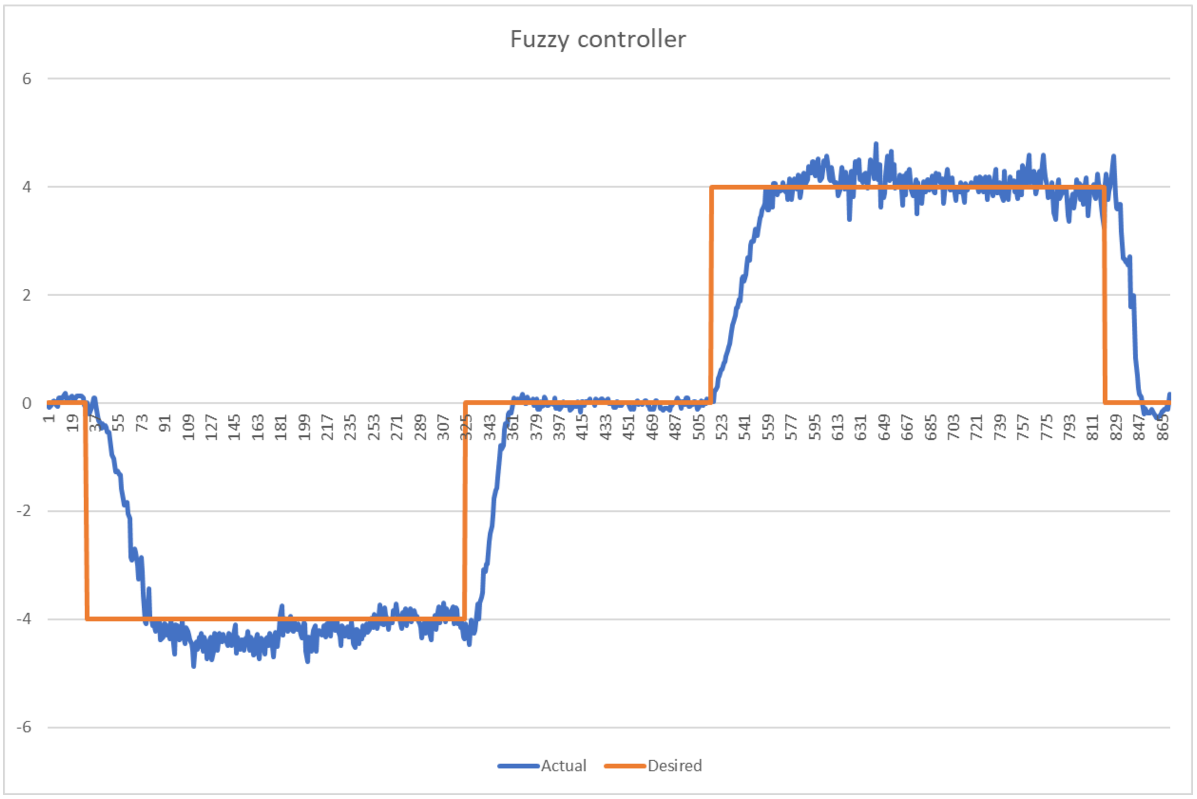


Fig 4.4 Desired Velocity vs Actual velocity curve for Fuzzy controller

# Chapter 5

# Slosh Measurement

Sloshing refers to the motion of a free liquid surface inside its container and is a complex nonlinear dynamical phenomenon which has a significant influence on the stability of the fluid system. Our aim is to develop a RL agent which reduces the slosh at the time of motion and also reduces the time of motion. For this, the first requirement is to develop a reliable slosh measurement device.

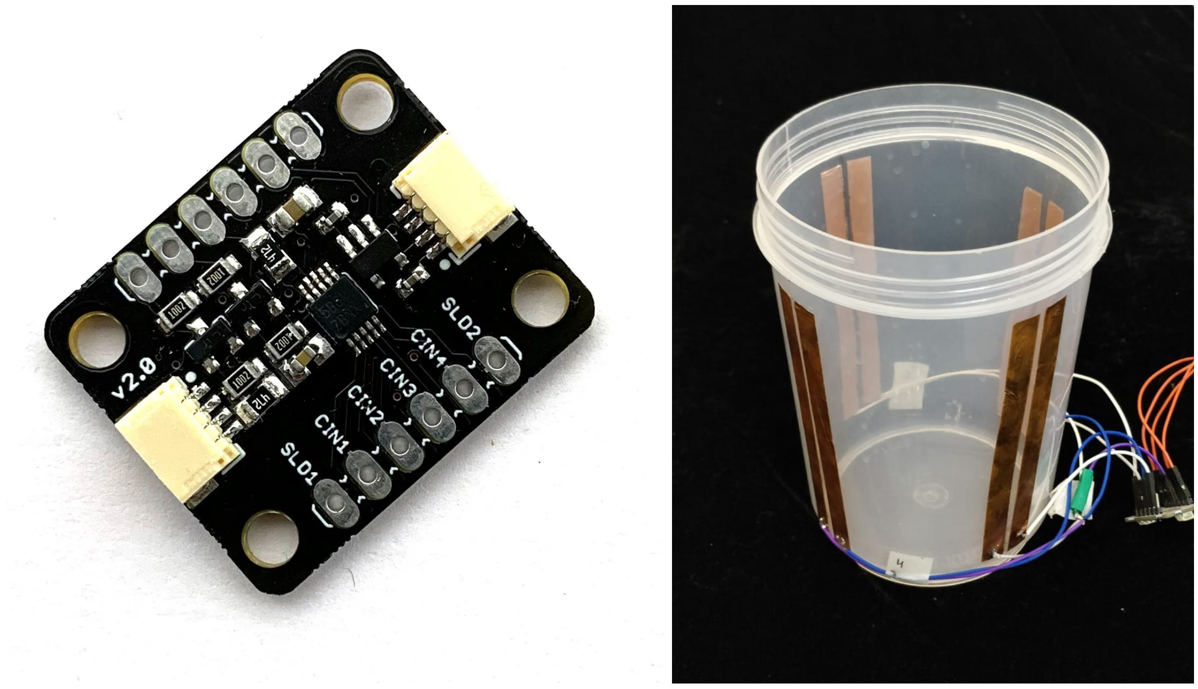
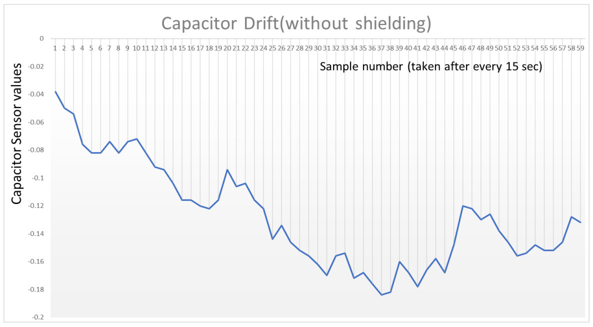


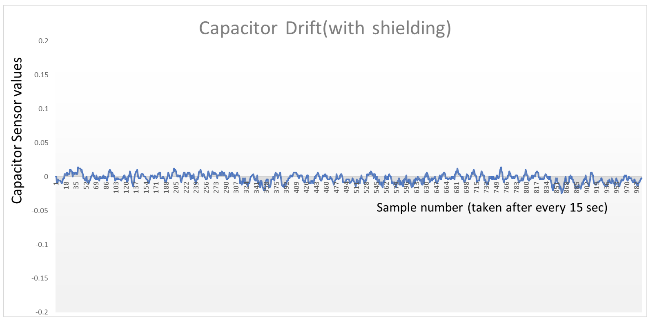
Fig 5.1 (a) Protocentral FDC1004 circuit board (b) container with capacitances

We are using Capacitive sensors to measure the height of water. We are doing this in 2 dimensions so that we know the 2-D Sloshing in the water. The capacitive sensor provides continuous data with reliable water level tracking.

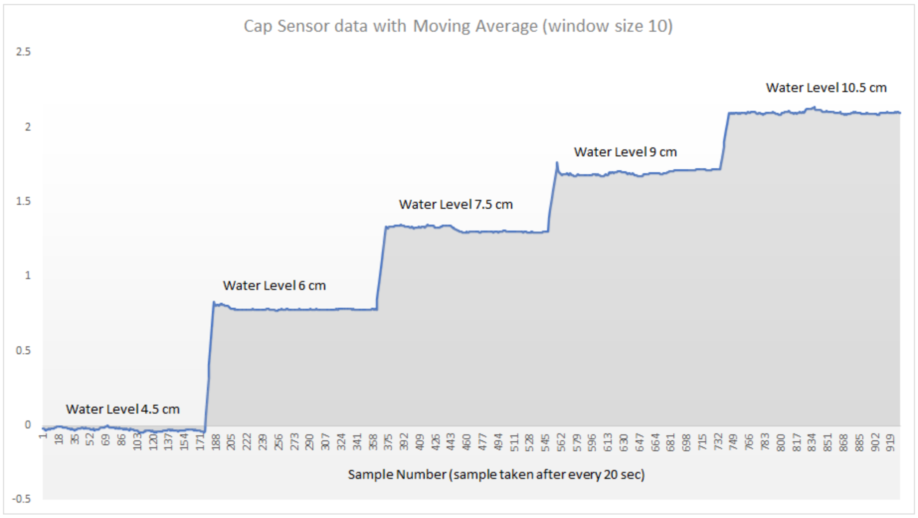
The capacitors are made from Copper Tape glued along one side of the container in pairs, so in this way they produce a fringing magnetic field. When water is filled in the container, it changes the Dielectric Medium of the capacitor thus changing the charges acquired. The reading from a capacitive sensor cannot be directly used. There is a circuit board used for this purpose to take the reading of the capacitive sensor and send the reading to the Arduino Mega board through I2C. The circuit board houses FDC1004, a chip form Texas Instruments(TI) which is capable of measuring 4 capacitance values simultaneously[ref FDC1004]. It also provides 2 shielding pins to shield the capacitors from the external noises which may be caused due to electromagnetic interference.

Initially it was found that there is a drift in the capacitance values. It resulted in the erroneous functioning of the AGV. The drift was there because of the external electromagnetic interference[ref TI guide][25]. To avoid this we used an Aluminum foil as a shield to prevent it from the external electromagnetic interference, and connected it to the shield pin. This eliminated the Drift in the capacitance values as shown in Fig 5.2.





(a) (b)



(c)

Fig 5.2 Capacitance values (a) without shielding, (b) with shielding, (c) at different water levels

In the AGV we need slosh reading in terms of the inclination of the free liquid surface. In a given moment it is calculated by

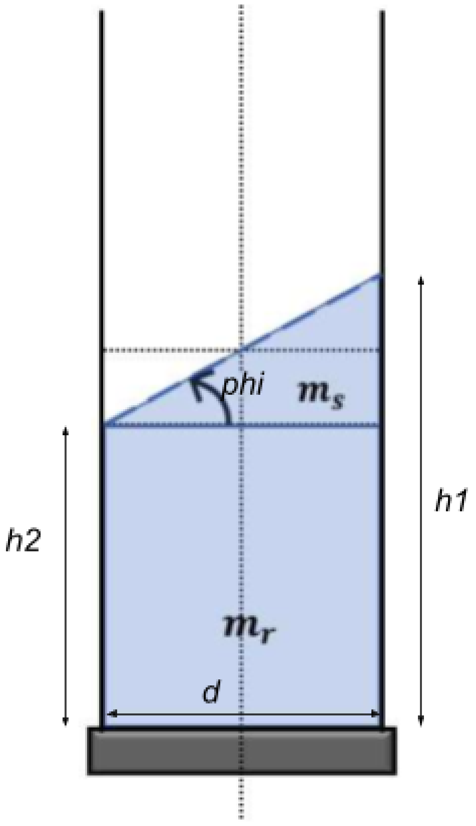


Fig 5.3 Slosh angle

$$\phi = arctan(\frac{h1-h2}{d}) \tag{5.1}$$

| | | |
|---|---|---|
| where, | $\phi$ | : Slosh Angle |
| | *h1, h2* | : height on the sides of the container |
| | *d* | : Diameter of the container |
| | $m_r$ | : mass of the resting liquid |
| | $m_s$ | : mass of the sloshing liquid |

There needs to be a conversion from capacitance readings to height values. We see that the height follows a linear relationship when plotted against the slosh values.

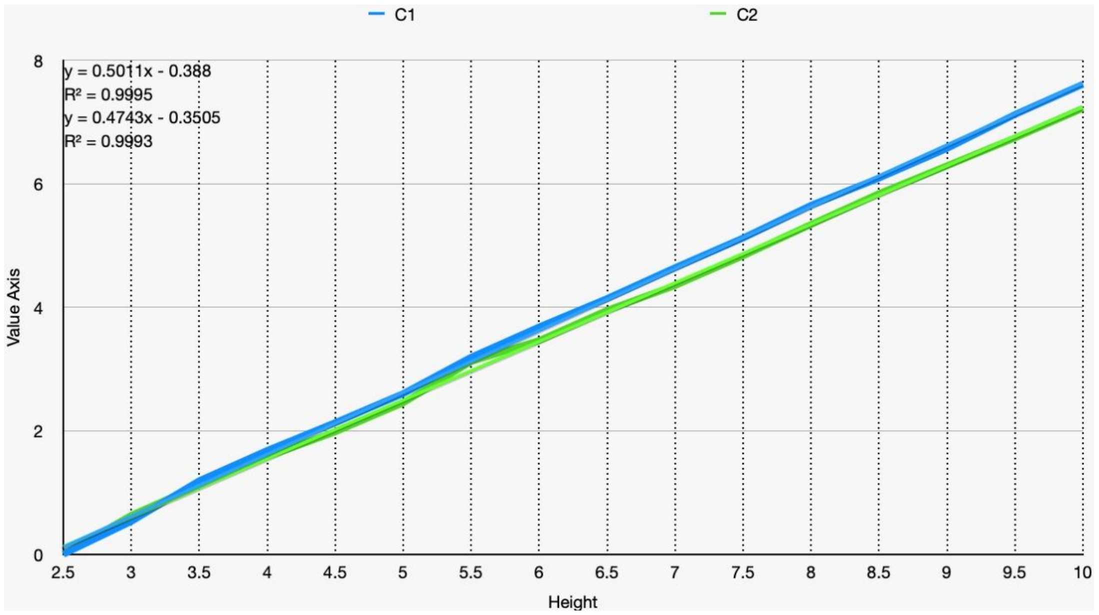


Fig 5.4 Capacitance values vs Height in cm

The Capacitor value follow a linear trend with height of water with $R^2$ ~ 0.999 and the equations for the same are

$$C_1 = 0.5011h - 0.388 \tag{5.2}$$

$$C_2 = 0.4743h - 0.3505 \tag{5.3}$$

# Chapter 6

# Super Twisting Control

Sliding mode control (SMC) is becoming a popular tool for working with UAVs and AGVs because of its resilience and speedy convergence features, making such controllers ideal for use in autonomous vehicles. This algorithm is inherently robust to the changes in parameters, non-linear models, external disturbances and uncertainty. Although the first order SMC used in the papers [26,27,28,29] yields a discontinuous control output, which cannot be applied to actuators because of the deterrent jittering effect[30] it can create in the AGV. In a slosh minimising problem this effect can give rise to more slosh. For this reason, here we have used Super-Twisting control (STC) which is based on the second-order sliding mode(SOSM)[31,32,33] for the system. This gives a continuous control signal that reduces the jitteriness when applied to the actuator.

## 6.1 Design

Here we have used a DRL based parameter tuning approach for the AGV, i.e. there is a controller which decides the velocity of the AGV based on the states it takes as feedback from the bot. The RL agent tunes the controller's parameters to achieve the desired goal in time. The state here comprises the displacement from the desired position, velocity, and sloshing angle.[59]

The dynamics of the error variables for this slosh container system can be represented as

$$\varepsilon_y = y - y_d \tag{6.1}$$

$$\varepsilon_\theta = \theta - \theta_d \tag{6.2}$$

$$\dot{\varepsilon}_y = \dot{y} - \dot{y}_d \tag{6.3}$$

$$\dot{\varepsilon}_\theta = \dot{\theta} - \dot{\theta}_d \quad \theta \tag{6.4}$$

the subscript ‘d’ denotes the desired values. For this problem, the desired position is taken as $y_d$= 1300 mm and $\theta_d$ , $\dot{\theta}_d$ will be equal to zero for slosh minimization. The error dynamics state vector can be represented as

$$[\delta_1\ \delta_2\ \delta_3\ \delta_4]^T = [\varepsilon_y\ \dot{\varepsilon}_y\ \varepsilon_\theta\ \dot{\varepsilon}_\theta]^T \tag{6.5}$$

Considering system outputs $\delta_1$ and $\delta_2$, the following is an evident linear sliding surface.

$$\rho = c_1\delta_2 + c_2\delta_1 \tag{6.6}$$

where $c_1$ and $c_2$ are sliding surface parameters.

For finite time convergence of system trajectories to a second order sliding set, the super twisting control developed in Thakar et al. (2017b)[34] is used:

$$v_l = -\, k_1|\rho|^{\frac{1}{2}} sign(\rho) - \sum_{t=0}^{t=t} sign(\rho)\, \Delta t \tag{6.7}$$

Based on the system dynamics and *v*, the designed control input in Thakar et al. (2017b) is given below:

$$u = (c_1 b_1)^{-1}(v_l - \omega_l) \tag{6.8}$$

where $b_1 = \frac{1}{M - m_s cos^2\delta_3}$ and $\omega_l = \frac{c_2}{c_1}(\rho - c_2\delta_1)$

Now after this the aim is to use the RL agent to fine tune the STC parameters $c_2$, $k_1$ and $k_2$. Without loss of generality, $c_1 = 1$ can be chosen.

# Chapter 7
# Reinforcement Learning

Reinforcement Learning is a learning approach in which an agent interacts with its surrounding environment by trial and error method and tries to learn an optimal behavioral strategy based on the reward signals received from the interactions [35]. Along with supervised and unsupervised learning, RL is one of three core Machine Learning paradigms. While traditional control approaches tend to use detailed mathematical models of the system and environment with fairly well-understood sources of uncertainty, RL methods aim to learn models and control actions directly from system data and experiments, which inherently include uncertainties in the system and disturbances acting on it. RL has recently been applied successfully on various complex scenarios like game playing [36,37,38], AGV path planning [39], legged-robot locomotion [40], autonomous helicopter flight [41] and many more. A recent work has used Deep RL along with expert demonstrations and Behaviour Cloning for control of baffles for slosh suppression in a simulated environment [42].

In RL Framework, the learner or decision maker is called the *Agent* and the thing it interacts with, comprising everything outside the agent, is called the *Environmen*t. The agent learns on the basis of its interaction with the environment to achieve a goal. More precisely , as shown in Fig 7.1, at a timestep $t$ The agent receives the state $S_t$ and reward $R_t$ and decides the action $A_t$ to be taken. After a timestep, as a consequence of its action, the agent receives the next state $S_{t+1}$ and reward $R_{t+1}$ . On the basis of this transition, the agent updates itself to perform actions that maximise the rewards received.

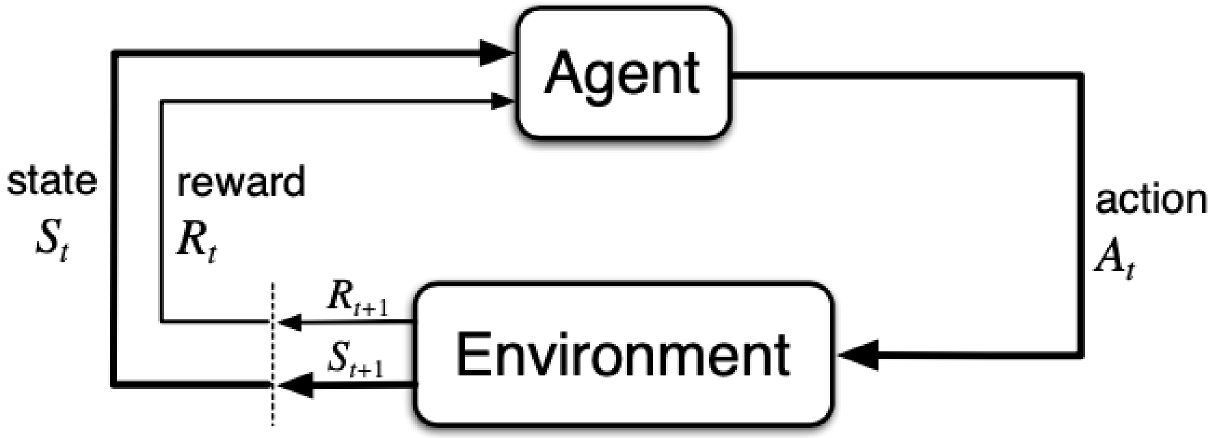


Fig 7.1 Agent - Environment Interaction

## 7.1 Definitions

The tasks which can be divided into subsequences or which have a notion of terminal state are called *Episodic Tasks*. In these tasks, the main motive of the RL Agent is to maximise the overall *Return* $G_t$ which is defined as

$$G_t = R_t + R_{t+1} + R_{t+2} + \ldots + R_T \quad (7.1)$$

where $R_t$ refers to the reward received at timestep t. In other words, the cumulative reward matters, not the immediate reward so the algorithm should learn to cope up with delayed rewards and not be greedy at every timestep.

A *Policy* π is defined as a mapping from states to probabilities of selecting each possible action[35]. Any policy π can be deterministic or stochastic and for an agent following a policy π, the probability to take action a while being in a particular state s is defined as $\pi(a|s)$.

The *Value Function* of a state s ,denoted as $v_\pi(s)$ is defined as the expected return while starting from the state s and following the policy π afterwards.

$$v_\pi(s) = E_\pi[G_t \mid S_t = s] \quad (7.2)$$

The *Action Value Function* for a state s and action a, denoted by $q_\pi(s, a)$ is defined as the expected return while starting from state s , taking action a and following policy π afterwards.

$$q_\pi(s, a) \; = \; E_\pi[G_t \mid S_t = s, \; A_t = a\,] \tag{7.3}$$

The *State Transition Probability* $p(s', r \mid s, a)$ is defined as the probability of reaching state s' and receiving a reward r after taking action a in state s. In other words, it defines the dynamics of the environment.

$$p(s', r|s, a) \; = \; Pr\{\, S_t = s', \; R_t = r \mid S_{t-1} = s, \; A_{t-1} = a\} \tag{7.4}$$

The *Bellman Equation* for value function $v_\pi(s)$ relates the value of a state to the value of the successor state. It is given by

$$v_\pi(s) \; = \; \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a)[r \, + \, \gamma v_\pi(s')] \tag{7.5}$$

The *Bellman Optimality Equation* for optimal action value $q_*$ relates the action values for an optimal policy $\pi_*$

$$q_*(s, a) \; = \; \sum_{s',r} p(s', r \mid s, a)[r \, + \, \gamma \, max_a \, q_*(s', a')] \tag{7.6}$$

The Bellman Optimality Equation is used as an update step in many algorithms[43] which seek to find the best action to be taken in a particular state using the action value function.

## 7.2 Algorithms

In Classical RL , initially the algorithms developed used Dynamic Programming like Value Iteration, Policy Iteration [44]  in which the dynamics of the environment or state transition probability is known. More recently, the trend towards making the agent  *Tabula Rasa* i.e with no previous knowledge about environment, gave rise to algorithms like Monte Carlo and Temporal Difference Learning [35] but these algorithms used tabular data structures to store state or action values which becomes infeasible as the number of  possible actions and states grow, the phenomenon is termed as the "*Curse of Dimensionality*" .

To avoid this problem, use of function approximators was introduced, these function approximators were used to approximate state or action value functions depending on the on policy data recorded. Advancing forward, after the advancements in neural networks research and due to their universal approximation property[45], neural networks began to be used as function approximators and this gave rise to the field of *Deep Reinforcement Learning.*

In Deep RL, one of the first algorithms to gain fame was DQN or *Deep Q Networks* [46], which displayed human level gameplay on Atari games. After this, many updates to DQN Algorithm came in the form of Deep Recurrent DQN[47], Dueling DQN[48], Double DQN[49], and Rainbow DQN[50] to name a few, each displaying better results in some sector of problems.

DQN was developed for discrete action spaces  and therefore was not feasible to use in tasks with continuous action spaces. This led to the development of policy gradient methods like TRPO[51], DDPG[52], PPO[53], SAC[54] etc. which use policies to map states and actions and iteratively update those policies to achieve better results.

In our case, the DRL Agent needs to tune the parameters of the Super Twisting Controller. Therefore, we have to work on continuous state and action spaces and so we choose to use the DDPG Algorithm for this task.

### 7.2.1 Deep Deterministic Policy Gradients

DDPG is a model-free, off-policy actor-critic algorithm which uses deep function approximators that can learn policies in high-dimensional, continuous action spaces [52]. The algorithm is described below followed by the description of key features.

Algorithm 1: Deep Deterministic Policy Gradients

```
Randomly initialize critic network $Q(s, a \mid \theta^{Q})$ and actor $\mu(s \mid \theta^{Q})$ with weights $\theta^{Q}$ and $\theta^{\mu}$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^{Q},\ \theta^{\mu'} \leftarrow \theta^{\mu}$
Initialize replay buffer R
for episode = 1, M do
    Initialize a random process $N$ for action exploration
    Receive initial observation state $s_1$
    for t = 1, T do
        Select action $a_t = \mu(s_t| \theta^{\mu}) + N_t$ according to the current policy and exploration noise
        Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
        Store transition $(s_t, a_t, r_t, s_{t+1})$ in R
        Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from R
        Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}| \theta^{\mu'})|\theta^{Q'})$
        Update critic by minimizing the loss : $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^{Q}))^2$
        Update the actor policy using the sampled policy gradient:
            $\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^{Q})|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu}))^2$
        Update the target networks :  $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1 - \tau)\theta^{Q'}$
                                      $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1 - \tau)\theta^{\mu'}$
    end for
end for
```

## Key Features

- **Actor Critic Approach** : To implement Q-learning on continuous action spaces, we need to optimise $a_t$ at every step to find the greedy policy, which is very slow for large unconstrained function approximators. So, DDPG uses an actor-critic approach based on the DPG Algorithm[55]. It consists of an actor function $\mu(s|\theta^{\mu})$ which deterministically maps states to specific actions and the critic function $Q(s, a)$ which estimates the action value and learns using the Bellman Equation (Eqn 7.6).
- **Batch Learning** : Introducing non-linear function approximators is essential for generalising on large state spaces but they don't guarantee convergence. Also, for making the algorithm hardware efficient, it's better to learn in batches than online. So, similar to NFQCA[56], DDPG also uses batch learning for stability, which is intractable for large networks.
- **Replay Buffer** : Most of the optimisation algorithms in Deep Learning assume that the samples are independent and identically distributed but that's not the case while learning as the samples are temporally correlated. So, like DQN, DDPG uses a replay buffer, which stores the tuples $(s_t, a_t, r_t, s_{t+1})$ and at each timestep, a mini-batch is uniformly sampled from the buffer to update the actor and critic networks.
- **Target Networks** : While computing the loss function for updating the critic, using the same critic network to provide target value can lead to divergence. Therefore, similar to DQN, DDPG uses target networks for both actor and critic whose weights are updated using "soft" target updates unlike directly copying weights [46]. The weights of target network slowly track the learned network using $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$ with $\tau \ll 1$
- **Batch Normalization** : While learning from low-dimensional features, different components may have different physical units, which can increase difficulty in learning. Using the Deep learning technique of Batch Normalization[57], each dimension of the samples in a mini-batch are normalized to have unit mean and variance which removes the need to manually ensure if the units are within a set range.
- **Exploration** : DDPG uses an additional noise process $N$ which is added to the actor policy to get the exploration policy $\mu'(s_t) = \mu(s_t| \theta_t^{\mu}) + N_t$

# Chapter 8

# RL Implementation

We have implemented DRL in combination with STC in simulation and on hardware. A simple pendulum analogy is used to model the lateral slosh dynamics for a container with a mobile base moving in a straight line [bandopadhyay] as shown in Fig 8.1. The combined system has 2 Degrees of Freedom, the displacement of the container, y and the lateral slosh angle, θ but only the displacement of the container is controllable giving rise to an underactuated system. The system's dynamical equations derived using Euler-Lagrange' formulation[] are :

$$M\ddot{y} + m_s l cos\theta\ddot{\theta} - m_s l\dot{\theta}^2 sin\theta = u + d \tag{8.1}$$

$$m_s l cos\theta\ddot{y} + m_s l^2\ddot{\theta} + c\dot{\theta} + m_s g l sin\theta = 0 \tag{8.2}$$

Where, M : Total Mass of the system

$m_s$: Mass of displaced liquid

$l$ : Length of pendulum

c : Viscous Damping Coefficient

u : Horizontal Force Applied on the container

g : Gravitational Acceleration

d : External Disturbance

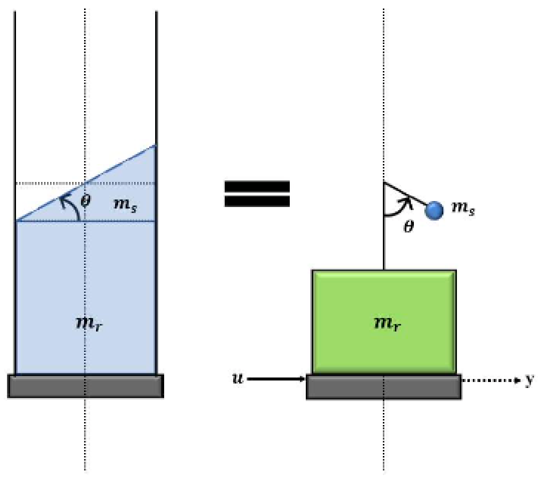


Fig 8.1 Pendulum Analogy for Lateral Slosh Control Problem

## 8.1 Parameter Estimation

To model the hardware prototype using the simple pendulum model, we need to estimate the parameters required in the dynamics[Eqn 8.1,Eqn 8.2] and controller equations.

For this, we have used the Translational Excitation and Quick Stop strategy[58]. The AGV is given a constant speed and stopped suddenly, emulating an impulsive behaviour. After getting stopped quickly, the motion of the sloshing liquid will be free under-damped oscillations. This motion is assumed to be viscously damped and is given by

$$\phi = C_1 e^{-\zeta\omega_n t} cos(\sqrt{1 - \zeta^2}\omega_n t + \psi) \tag{8.3}$$

with initial condition $\dot{\theta} = 0$ at quick stop $(t = 0)$.

The data was recorded for 34 runs, which is shown in Fig 8.2. To obtain the natural frequency and Viscous Damping Coefficient, the damped frequency and the decay of the resulting waveform were used. The damped frequency of oscillation was calculated by taking the Fast Fourier Transform of the signal. As can be seen in Fig 8.3 , the FFT of all the individual runs have peak at nearly the same frequency, the final damped frequency $\omega_d$ is the average of all the frequencies, which came out to be 2.74 Hz. The length of the pendulum can be found from the natural frequency $\omega_n$ using

$$\omega_n = \sqrt{\frac{g}{l_{pe}}} \tag{8.4}$$

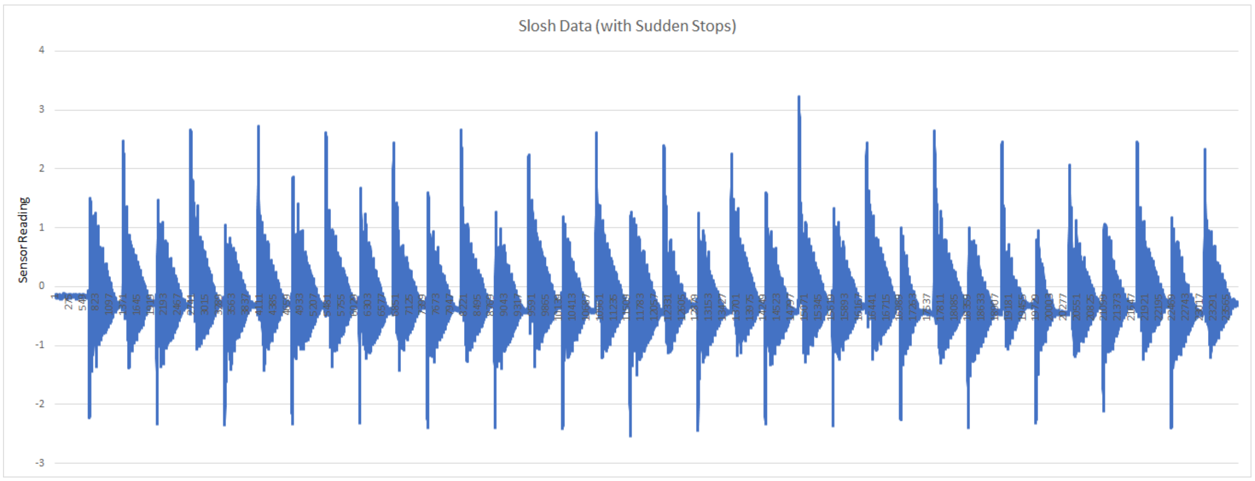


Fig 8.2 Slosh Data recorded after quick stop for 34 runs

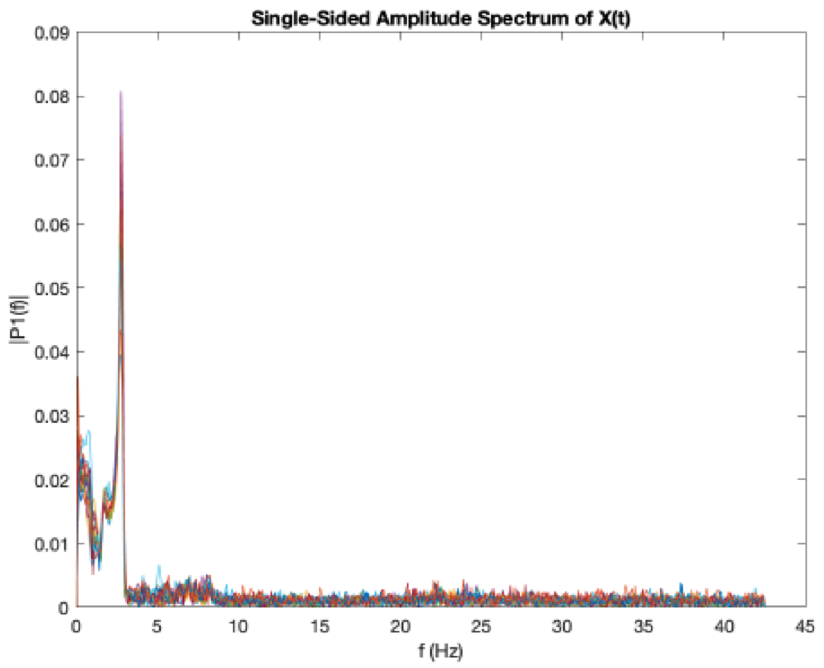


Fig 8.3 FFT of ϕ signal for all the runs

After Calculation, the final values calculated are listed in Table 8.1

| **Parameters** | **Value** |
|---|---|
| Slosh mass ($m_s$) | 0.375 kg |
| Total Mass (M) | 2.7 kg |
| Length of Pendulum ($l$) | 0.0328 m |
| Gravitational Acceleration (g) | 9.8 m/s$^2$ |
| Viscous Damping Coefficient (c) | 0.000279 kgm$^2$/sec |

Table 8.1 Estimated Parameters for AGV

Using the estimated parameters in the simulation dynamics, the DDPG Agent was trained for 10 thousand episodes using the framework shown in Fig 8.4. An NVIDIA GeForce GTX 1080 machine with 12GB RAM and 16 core CPU is used for the entire processing. The agent was subjected to 10 test runs for predicting the STC parameters with random starting states within a predefined range, the output parameters were nearly identical with the largest change being 3.94%. The predicted parameters after training that gave the best results were

$$[C1,\ K1,\ K2]\ = [2.1309,\ 1.6884,\ \ 0.9694]$$

The results for the same are shown in Fig 8.5.

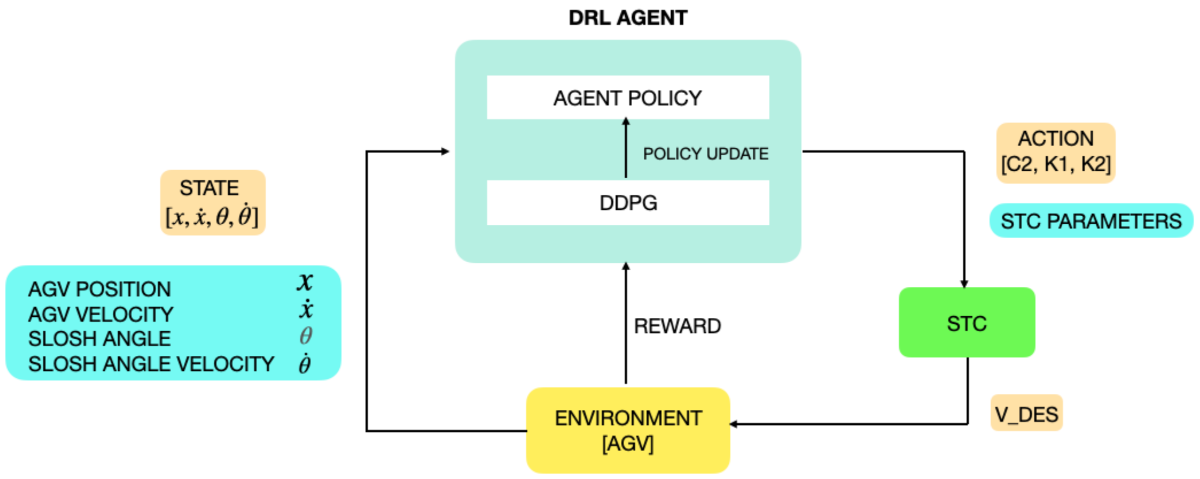
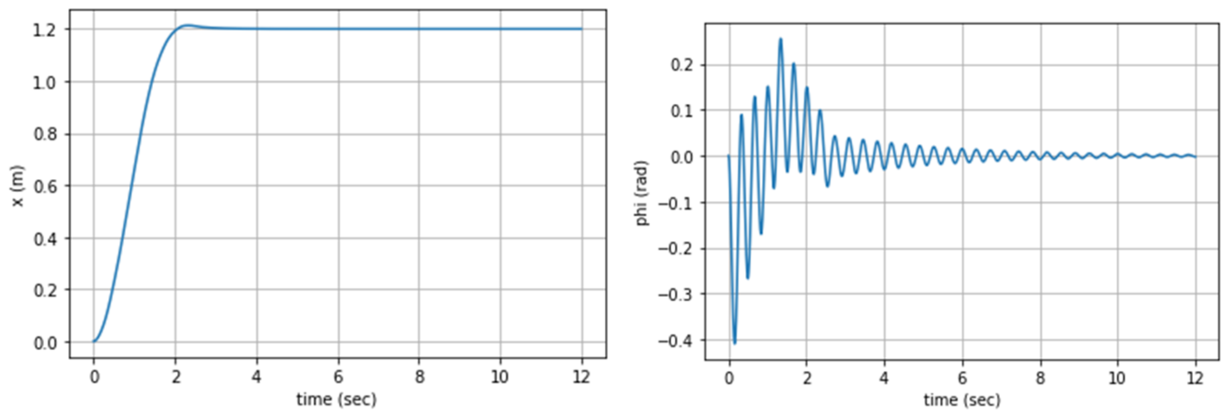
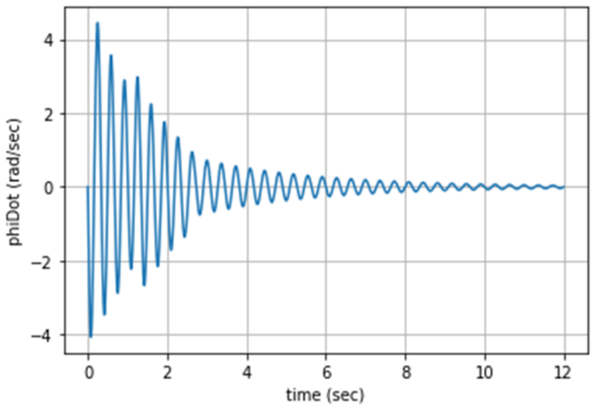


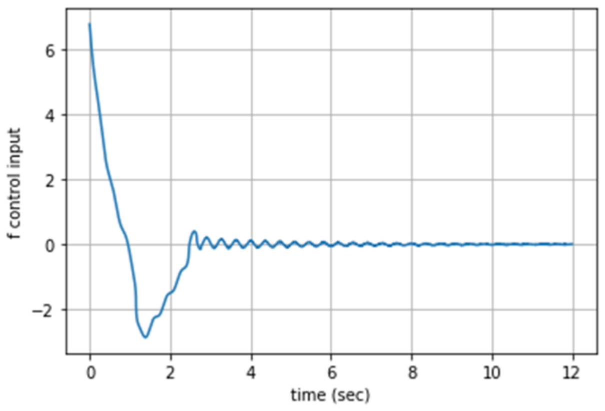
Fig 8.4 Overall DRL Framework



(a) (b)

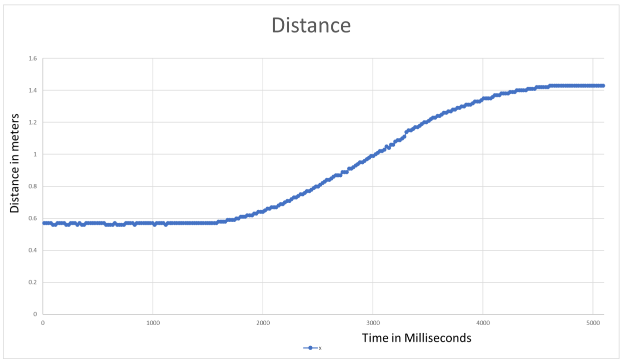
Fig 8.5 (a) Position of AGV vs time, (b) Slosh angle vs time

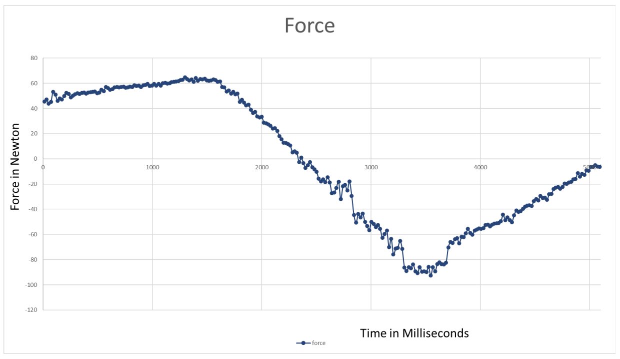




(a) (b)

Fig 8.6 (a) Rate of change of Slosh angle vs time, (b) Force output vs time





(a) (b)

Fig 8.7 Hardware implemented results (a) Position of AGV, (b) Force output of AGV

# Chapter 9

# Conclusion

An AGV prototype is built which can be used as a research testbed to implement different control algorithms for research in Slosh Control. The prototype is capable of measuring 2-D slosh, navigate in 2-D Arena, implementing velocity control, and implement compute intensive control algorithms on hardware. A slosh measurement system is developed using non-invasive capacitive water level sensing, which records the slosh in 2-D. The system is modelled using a simple pendulum based model and the model parameters are estimated on hardware using the given procedure ,which come out in the desired range. Using the estimated model, a Super Twisting Controller is implemented which provides robustness to the control system. A Deep Reinforcement Learning Agent is designed using Deep Deterministic Policy Gradient Algorithm to tune the parameters of STC which is a first of its kind implementation of DRL in the area of slosh control. The DRL Agent provides appreciable results in the simulation which when implemented on hardware results in a performance

In Future, the DRL agent can be trained directly on hardware to eradicate the errors due to modelling and different algorithms like SAC, PPO can be implemented which have shown better results on hardware tasks.

## References

[1] H. N. Abramson, “The dynamic behavior of liquids in moving containers,Technical Report, NASA, Washington, DC, USA,” 1966.

[2] R. A. Ibrahim, V. N. Pilipchuk, and T. Ikeda, “Recent Advances in Liquid Sloshing Dynamics,” Appl. Mech. Rev., vol. 54, pp. 133–199, 2001.

[3] B. Bandyopadhyay, P. S. Gandhi, and S. Kurode, “Sliding Mode Observer Based Sliding Mode Controller for Slosh-Free Motion Through PID Scheme,” IEEE Transactions on Industrial Electronics, vol. 56, no. 9, pp. 3432–3442, 2009.

[4] L. D. Peterson, E. F. Crawley, and R. J. Hansman, “Nonlinear fluid slosh coupled to the dynamics of a spacecraft,” AIAA Journal, vol. 27, no. 9, pp. 1230–1240, Sep. 1989, doi: 10.2514/3.10250.

[5] L. Moriello, L. Biagiotti, C. Melchiorri, and A. Paoli, “Manipulating liquids with robots: A sloshing-free solution,” *Control Engineering Practice*, vol. 78, pp. 129–141, Sep. 2018, doi: 10.1016/j.conengprac.2018.06.018.

[6] H. N. Abramson, J. Ransleben, and Guido E., “SOME STUDIES OF A FLOATING LID TYPE DEVICE FOR SUPPRESSION OF LIQUID SLOSHING IN RIGID CYLINDRICAL TANKS,” Defense Technical Information Center, Fort Belvoir, VA, May 1961. Accessed: Apr. 27, 2022. [Online]. Available: http://dx.doi.org/10.21236/ad0612785

[7] R. A. Ibrahim, V. N. Pilipchuk, and T. Ikeda, “Recent Advances in Liquid Sloshing Dynamics,” *Applied Mechanics Reviews*, vol. 54, no. 2, pp. 133–199, Mar. 2001, doi: 10.1115/1.3097293.

[8] T. Kandasamy, “An Analysis of Baffles Designs for Limiting Fluid Slosh in Partly Filled Tank Trucks~!2009-10-29~!2010-04-21~!2010-07-23~!,” *The Open Transportation Journal*, vol. 4, no. 1, pp. 23–32, Jul. 2010, doi: 10.2174/1874447801004010023.

[9] S. Kurode, S. Spurgeon, B. Bandyopadhyay, and P. S. Gandhi, “Sliding mode control for slosh-free motion using a nonlinear sliding surface,” IEEE/ASME Trans. Mechatronics, vol. 18, pp. 714–724, 2013.

[10] S. Kurode, B. Bandyopadhyay, and P. S. Gandhi, “Sliding mode control for slosh-free motion of a container using partial feedback linearization,”in Proc. Int. Workshop Variable Struct. Syst., Antalya, Turkey, 2008, pp. 367–372.

[11] H. Richter, “Motion control of a container with slosh: constrained sliding mode approach,” J. Dyn. Syst., Meas., Control, vol. 132, pp. 1–10, 2010.

[12] H. Sira-Ramirez, “A flatness based generalized PI control approach to liquid sloshing regulation in a moving container,” in Proc. Amer. Control Conf., Anchorage, AK, USA, 2002, pp. 2909–2914

[13] B. Pridgen, K. Bai, and W. Singhose, “Slosh suppression by robust input shaping,”. 49th IEEE Conference on Decision and Control (CDC), 2010

[14] A. Aboel-Hassan, M. Arafa, and A. Nassef, “Design and optimization of input shapers for liquid slosh suppression,” Journal of Sound and Vibration, vol. 320, no. 1–2, pp. 1–15, 2009.

[15] M. Reyhanoglu and J. R. Hervas, “Nonlinear modeling and control of slosh in liquid container transfer via a PPR robot,” Commun. Nonlinear Sci. Numer. Simul., vol. 18, no. 6, pp. 1481–1490, 2013

[16] P. S. Yadav, V. Agrawal, J. C. Mohanta, and M. D. Faiyaz Ahmed, “A robust sliding mode control of mecanum wheel-chair for trajectory tracking,” *Materials Today: Proceedings*, vol. 56, pp. 623–630, 2022, doi: 10.1016/j.matpr.2021.12.398.

[17] J. E. Mohd Salih, M. Rizon, and S. Yaacob, “Designing Omni-Directional Mobile Robot with Mecanum Wheel,” *American Journal of Applied Sciences*, vol. 3, no. 5, pp. 1831–1835, May 2006, doi: 10.3844/ajassp.2006.1831.1835.

[18] R. Xin, Z. Zou, and W. Mu, “A Design of Hull Coating Robot Based on Mecanum Wheel and Electromagnet,” Dec. 2019. Accessed: Apr. 28, 2022. [Online]. Available: http://dx.doi.org/10.1109/icicas48597.2019.00151

[19] C. He,D. Wu,K. Chen, F. Liu, N. Fan, “Analysis of the Mecanum Wheel Arrangement of an Omnidirectional Vehicle,”. Proceedings : Journal of Mechanical Engineering Science 1989-1996 (vols 203-210) , 2019, vol. 233, pp. 5329–5340

[20] H. Taheri, B. Qiao, and N. Ghaeminezhad, “Kinematic Model of a Four Mecanum Wheeled Mobile Robot,” *International Journal of Computer Applications*, vol. 113, no. 3, pp. 6–9, Mar. 2015, doi: 10.5120/19804-1586.

[21] S. Bhagwan, A. Kumar, J.K.Soni," A Review on: PID Controller," ,*International Journal on Recent Technologies in Mechanical and Electrical Engineering (IJRMEE),* vol 3, issue 2, pp. 17-22, Feb 2016.

[22] Pedrycz, Witold (1993). *Fuzzy control and fuzzy systems* (2 ed.). Research Studies Press Ltd.

[23] Hájek, Petr (1998). *Metamathematics of fuzzy logic* (4 ed.). Springer Science & Business Media.

[24] "Implementation-of-Fuzzy-Logic.pdf," *Google Docs*. https://drive.google.com/file/d/0B_0wpY02867rT3Nyd19QMVE0S2c/view?resourcekey=0-_Oqm9L4KCacRW0ReUDYIrw .

[25] David Wang, "FDC1004: Basics of capacitive sensing and its applications". Application Report by Texas Instruments. https://www.ti.com/lit/an/snoa927a/snoa927a.pdf?ts=1651139350960

[26] Bandyopadhyay, B., Gandhi, P.S., Kurode, S.: 'Sliding mode observer based sliding mode controller for slosh-free motion through PID scheme', IEEE Trans. Ind. Electron., 2009, 56, (9), pp. 3432–3442

[27] Kurode, S., Spurgeon, S., Bandyopadhyay, B., et al.: 'Sliding mode control for slosh-free motion using nonlinear sliding surface', IEEE/ASME Trans. Mechatron., 2013, 18, (2), pp. 714–724

[28] Thakar, P.S., Bandyopadhyay, B., Gandhi, P.S.: 'Sliding mode control for an underactuated slosh-container system using nonlinear model', Int. J. Adv. Mechatron. Syst., 2013, 5, (5), pp. 335–344

[29] Thakar, P.S., Bandyopadhyay, B., Gandhi, P.S.: 'Sliding Mode control for a class of underactuated systems using feedforward normal form: a sloshcontainer system'. 13th Int. Workshop Var. Struct. Syst. (VSS), Nantes, France, 2014, pp. 1–6

[30] Utkin, V.I.: 'Sliding modes in control optimization' (Springer, 1992)

[31] Levant, A.: 'Principles of 2-sliding mode design', Automatica, 2007, 43, (4), pp. 576–586

[32] Shtessel, Y., Edwards, C., Fridman, L., et al.: 'Sliding mode control and observation' (Birkhuser, 2014)

[33] Moreno, J.A., Osorio, M.: ‘Strict Lyapunov functions for the super-twisting algorithm’, IEEE Trans. Autom. Control, 2012, 57, (4), pp. 1035–1040

[34] Thakar, P. S., Bandyopadhyay, B. and Gandhi, P. (2017b). Improved output-feedback second order sliding mode control design with implementation for underactuated slosh-container system having confined track length, IET Control Theory and Applications, 11(8), 1316–1323.

[35] R. S. Sutton and A. G. Barto, Reinforcement Learning An Introduction. Cambridge, MA, USA: MIT Press, 2018

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, C. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg & D. Hassabis, “Human-level control through deep reinforcement learning,” Nature, vol. 518, pp. 529–533, 2015.

[37] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel & D. Hassabis , “Mastering the game of Go with deep neural networks and tree search,” Nature, vol. 529, pp.484–489, 2016.

[38] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel & D. Hassabis , “Mastering the game of Go without human knowledge,” Nature, vol. 550, pp. 354–359, 2017.

[39] X. Guo, Z. Ren, Z. Wu, J. Lai, D. Zeng, and S. Xie, “A Deep Reinforcement Learning Based Approach for AGVs Path Planning,” in Chinese Automation Congress (CAC), Shanghai, China, Nov. 2020.

[40] J. Yue, “Learning Locomotion For Legged Robots Based on Reinforcement Learning: A Survey,” in International Conference on Electrical Engineering and Control Technologies (CEECT), 2020.

[41] A. Coates, P. Abbeel, and A. Y. Ng, “Autonomous Helicopter Flight Using Reinforcement Learning,” in Encyclopedia of Machine Learning and Data Mining, Boston, MA: Springer US, 2017, pp. 75–85.

[42] Y. Xie and X. Zhao, “Sloshing suppression with active controlled baffles through deep reinforcement learning–expert demonstrations–behavior cloning process,” Physics of Fluids, vol. 33, no. 1, p. 017115, 2021.

[43] C. J. C. H. Watkins and P. Dayan, “Q-learning,” Machine Learning, vol. 8, no. 3–4, pp. 279–292, May 1992, doi: 10.1007/bf00992698.

[44] R. A. Howard, Dynamic programming and Markov processes. 1960.

[45] A. N. Gorban and D. C. Wunsch, "The general approximation theorem," 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227), 1998, pp. 1271-1274 vol.2, doi: 10.1109/IJCNN.1998.685957.

[46] V. Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra and Martin A. Riedmiller. “Playing Atari with Deep Reinforcement Learning.” ArXiv abs/1312.5602 (2013):

[47] M. Hausknecht ,P. Stone, “Deep Recurrent Q-Learning for Partially Observable MDPs”, 07,2015

[48] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, N. De Freitas, ‘Dueling Network Architectures for Deep Reinforcement Learning’, *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, 2016.

[49] H. van Hasselt, A. Guez, D. Silver, ‘Deep Reinforcement Learning with Double Q-Learning’, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016

[50] M. Hessel , ‘Rainbow: Combining Improvements in Deep Reinforcement Learning’, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, 2018.

[51] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, ‘Trust Region Policy Optimization’, *Proceedings of the 32nd International Conference on Machine Learning*, 07--09 Jul 2015, τ. 37, 1889–1897.

[52] T. P. Lillicrap , ‘Continuous control with deep reinforcement learning’, *CoRR*, τ. abs/1509.02971, 2016.

[53] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, ‘Proximal Policy Optimization Algorithms’, 07 2017.

[54] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, ‘Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor’, *ICML*, 2018.

[55] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, ‘Deterministic Policy Gradient Algorithms’, *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, 2014.

[56] Hafner, R., Riedmiller, M. Reinforcement learning in feedback control. *Mach Learn* 84, 137–169 (2011). https://doi.org/10.1007/s10994-011-5235-x

[57] Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[58] D. Odhekar, P. Gandhi, and K. Joshi, “Novel Methods for Slosh Parameter Estimation Using Pendulum Analogy,” Jun. 2005. Accessed: Apr. 28, 2022. [Online]. Available: http://dx.doi.org/10.2514/6.2005-5923

[59] A. K. Shakya, K. Bithel, G. N. Pillai and S. Chakrabarty, “Deep Reinforcement Learning based Super Twisting Controller for Liquid Slosh Control Problem”, in Advances in control and Optimization of Dynamical Sytems - 7th ACODS, Feb. 2022.