



Universidad de Valladolid



ESCUELA DE INGENIERÍAS  
INDUSTRIALES

Máster en Electrónica Industrial y Automática

# **MASTER EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA**

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

**Master Thesis**

## **AUTONOMOUS TELEPRESENCE MOBILE ROBOT (ATEMR)**

---

### **DISEÑO DE UN ROBOT MÓVIL AUTÓNOMO DE TELEPRESENCIA**

*Author:*

Ephson-Effa Guakro Asare

*Supervisors:*

Dr. Eduardo Zalama Casanova

Dr. Jaime Gómez García-Bermejo

Valladolid, February 2022



Universidad de Valladolid



ESCUELA DE INGENIERÍAS  
INDUSTRIALES

Máster en Electrónica Industrial y Automática

# MASTER EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

**Master Thesis**

## **AUTONOMOUS TELEPRESENCE MOBILE ROBOT (ATEMR)**

---

## **DISEÑO DE UN ROBOT MÓVIL AUTÓNOMO DE TELEPRESENCIA**

*Author:*

Ephson-Effa Guakro Asare

*Supervisors:*

Dr. Eduardo Zalama Casanova

Dr. Jaime Gómez García-Bermejo

Valladolid, February 2022

## *Abstract*

The recent rise in tele-operated autonomous mobile vehicles calls for a seamless control architecture that reduces the learning curve when the platform is functioning autonomously (without active supervisory control), as well as when tele-operated. Conventional robot platforms usually solve one of two problems. This work develops a mobile base using the Robot Operating System (ROS) middleware for teleoperation at low cost. The three-layer architecture introduced adds or removes operator complexity. The lowest layer provides mobility and robot awareness; the second layer provides usability; the upper layer provides interactivity. A novel interactive control that combines operator intelligence/ skill with robot/ autonomous intelligence enabling the mobile base to respond to expected events and actively react to unexpected events is presented. The experiments conducted in the robot laboratory summarises the advantages of using such a system.

El reciente auge de los vehículos móviles autónomos teleoperados exige una arquitectura de control sin fisuras que reduzca la curva de aprendizaje cuando la plataforma funciona de forma autónoma (sin control de supervisión activo), así como cuando es teleoperada. Las plataformas robóticas convencionales suelen resolver uno de los dos problemas. Este trabajo desarrolla una base móvil que utiliza el middleware Robot Operating System (ROS) para la teleoperación a bajo coste. La arquitectura de tres capas introducida añade o elimina la complejidad del operador. La capa más baja proporciona movilidad y conciencia robótica; la segunda capa proporciona usabilidad; la capa superior proporciona interactividad. Se presenta un novedoso control interactivo que combina la inteligencia/habilidades del operador con la inteligencia autónoma del robot, lo que permite que la base móvil responda a los eventos esperados y reaccione activamente a los eventos inesperados. Los experimentos realizados en el laboratorio robótica resumen las ventajas de utilizar un sistema de este tipo.

## *Acknowledgements*

Glory to the most high God for His guidance and provision. I would like to express my gratitude to Dr. Eduardo Zalama Casanova and Dr. Jaime Gómez García-Bermejo for their time and experience that enabled this work to reach completion. Ivan, Mykhaylo and Andrei, I am grateful for every minute we spent together sharing ideas that made this project a success. It has been a long run, and an adventurous one as such. Last but not least, I would like to thank my mother, Mrs. Margaret EFFA ASARE and uncle, Paul ASARE WIREDU (may his soul rest in peace) for their continual support and encouragement without which none of this would have been possible. Thank you.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>ii</b>
<b>1 GENERAL INTRODUCTION AND OBJECTIVES</b>	<b>1</b>
1.1 INTRODUCTION . . . . .	1
1.2 PROBLEM STATEMENT . . . . .	1
1.3 RELATED WORK . . . . .	2
1.3.1 Robotic Telepresence . . . . .	2
1.3.2 ROS Based Mobile Robots . . . . .	2
1.3.3 Fuzzy Inference Systems in Mobile Robots . . . . .	3
1.4 SCOPE AND OBJECTIVE . . . . .	4
1.5 PROJECT STRUCTURE AND ORGANIZATION . . . . .	4
<b>2 LITERATURE REVIEW</b>	<b>6</b>
2.1 INTRODUCTION . . . . .	6
2.2 DEFINITION OF TERMS . . . . .	6
2.3 WHAT IS A ROBOT? . . . . .	6
2.3.1 WHEELED MOBILE ROBOT (WMR) KINEMATICS . . . . .	9
2.3.2 DIFFERENTIAL DRIVE MOBILE BASE . . . . .	10
2.4 AUTONOMOUS MOBILE ROBOT . . . . .	12
2.4.1 ENCODER . . . . .	13
2.4.2 INERTIAL MEASUREMENT UNIT (IMU) . . . . .	13
2.4.3 GYROSCOPE . . . . .	14
2.4.4 LIGHT DETECTION AND RANGING (LIDAR) SENSOR . . . . .	14
2.4.5 ULTRASONIC SENSOR . . . . .	15
2.4.6 INFRARED SENSOR . . . . .	16
2.4.7 MAPPING . . . . .	16
KALMAN FILTER . . . . .	16
PARTICLE FILTER . . . . .	18

2.4.8	LOCALIZATION . . . . .	18
2.4.9	NAVIGATION . . . . .	20
	PATH PLANNING . . . . .	20
<b>3</b>	<b>MOBILE ROBOT ANALYSIS AND DESIGN</b>	<b>23</b>
3.1	INTRODUCTION . . . . .	23
3.1.1	Identification of requirements . . . . .	23
3.1.2	Definition of requirements . . . . .	24
3.2	THE MOBILE BASE HARDWARE . . . . .	24
3.2.1	DESIGN AND CONSTRUCTION OF THE MOBILE BASE . . . . .	25
3.2.2	Robot Base Specifications . . . . .	37
	Components of the Mobile Base . . . . .	38
3.3	Robot Base Core Software . . . . .	40
3.4	ATEMR Layout and Sub-Systems . . . . .	41
3.4.1	Low-Level: SpeedRamp . . . . .	41
3.4.2	Low-Level: The HEXAGONAL PATTERN (HEX-PATTERN) Generator . . . . .	43
3.4.3	Low-Level: The Reactive Base Fuzzy-Inference Engine . . . . .	44
3.5	The Primary Core . . . . .	46
3.5.1	ATEMR ROS Packages . . . . .	47
3.5.2	Agent Statemachine . . . . .	52
3.5.3	Web User Interface (UI) . . . . .	52
	General Overview . . . . .	54
	The Authentication Page . . . . .	55
	The Control Page . . . . .	57
	The Mapping Page . . . . .	59
	The Setting Page . . . . .	60
3.6	The Secondary Core . . . . .	62
3.6.1	BYTE-WISE CAN data frame . . . . .	64
3.6.2	BIT-WISE CAN data frame . . . . .	66
3.6.3	Secondary Core: Internal Workflow . . . . .	67
	Control-Power Interface . . . . .	71
	Display Interface . . . . .	72
	Captive Portal & IP Address Interface . . . . .	76
	Sensing & Actuating Interface . . . . .	78
3.7	DESIGN CYCLE . . . . .	78
<b>4</b>	<b>ATEMR BUILD AND IMPLEMENTATION</b>	<b>79</b>

4.1	INTRODUCTION . . . . .	79
4.2	ROBOT STARTUP . . . . .	79
4.3	ROBOT LOCALIZATION AND NAVIGATION . . . . .	80
4.4	ROBOT MAPPING . . . . .	84
4.5	NAVIGATION TESTING . . . . .	84
<b>5</b>	<b>CONCLUSIONS AND RECOMMENDATIONS</b>	<b>87</b>
5.1	Summary . . . . .	87
5.2	The Reactive Base Fuzzy-Inference Engine - Testing . . . . .	88
5.3	Conclusion . . . . .	88
5.4	Recommendations . . . . .	89
	<b>References</b>	<b>90</b>

# List of Figures

1.1	Fuzzy Inference System (Source)	3
2.1	Robot Classification by Operating Environment (Ben-Ari and Mondada (2018))	7
2.2	Pick and Place foundry automation (Bachmann, KUKA Roboter GmbH (2021))	7
2.3	Pick and Place foundry automation (Ben-Ari and Mondada (2018))	8
2.4	Differential Drive kinematics (from Dudek and Jenkin (2010))	11
2.5	General organization of the functions of an autonomous mobile robot (Siegwart et al. (2011))	12
2.6	And Incremental Rotary Encoder (Lambtron (2018))	13
2.7	Apollo IMU stable member (Reinhold (2019))	14
2.8	A digital gyroscope module connected to an Arduino Uno board (from Rahat (2014))	15
2.9	Time of flight principle (from RCraig09 (2020))	15
2.10	Kalman Filter algorithm (from Petteri Aimonen (2011))	17
2.11	General schematic for mobile robot localization (from Siegwart et al. (2011))	19
2.12	Localization by Trilateration (from GIS Geography (2021))	19
2.13	Localization by Triangulation (from GIS Geography (2021))	19
2.14	Configuration space of a point-sized robot. White = $C_{free}$ , gray = $C_{obs}$ (from Simeon87 (2008))	20
3.1	ATEMR: Mobile Base Design - Raw sketch 1 FRONT VIEW	25
3.2	ATEMR: Mobile Base Design - Raw sketch 1 RIGHT VIEW	25
3.3	ATEMR: Mobile Base Design - Raw sketch 1 TOP VIEW	26
3.4	ATEMR: Mobile Base Design - Raw sketch RIGHT VIEW	26
3.5	ATEMR: Mobile Base Design - 3D design v1 FRONT VIEW	27
3.6	ATEMR: Mobile Base Design - 3D design v1 LEFT VIEW	27
3.7	ATEMR: Mobile Base Design - 3D design v1 ISOMETRIC VIEW	27
3.8	ATEMR: Mobile Base Design - 3D design v2 FRONT VIEW	28
3.9	ATEMR: Mobile Base Design - 3D design v2 LEFT VIEW	28
3.10	ATEMR: Mobile Base Design - 3D design v2 BOTTOM VIEW	29



3.11 ATEMR: Mobile Base Design - 3D design v2 EXPLODED VIEW . . . . .	29
3.12 ATEMR: Secondary Core Casing Design - Raw sketch . . . . .	30
3.13 ATEMR: LIDAR and IMU Support Structure - 3D . . . . .	30
3.14 ATEMR: Spring-Loaded Lock Design - 3D . . . . .	31
3.15 ATEMR: Overall electrical diagram . . . . .	31
3.16 ATEMR: Mobile Base Build - FRONT VIEW . . . . .	32
3.17 ATEMR: Mobile Base Build - TOP VIEW . . . . .	33
3.18 ATEMR: Mobile Base Build - REAR VIEW . . . . .	34
3.19 ATEMR: Mobile Base Build (without wheels)- PERSPECTIVE VIEW . . . . .	35
3.20 ATEMR: Mobile Base Completed Build - FRONT VIEW . . . . .	36
3.21 ATEMR: Mobile Base Completed Build - LEFT VIEW . . . . .	37
3.22 ATEMR: Core Software Organization . . . . .	40
3.23 ATEMR: Speed Ramp visual . . . . .	41
3.24 ATEMR: Project layout . . . . .	42
3.25 ATEMR: Project architecture . . . . .	43
3.26 ATEMR: HEX-PATTERN . . . . .	44
3.27 ATEMR: HEX-PATTERN-90 (90°) . . . . .	44
3.28 ATEMR: IDLE Inference Engine setup . . . . .	45
3.29 ATEMR: MANUAL Inference Engine setup . . . . .	46
3.30 ATEMR: Project sub-divisions . . . . .	46
3.31 ATEMR: Package structure and organization . . . . .	47
3.32 ATEMR: Gazebo Simulated World . . . . .	48
3.33 ROS Navigation stack setup (from Marder-Eppstein (2020a)) . . . . .	48
3.34 Dynamic Window Approach (from Marder-Eppstein and Perko (2019)) . . . . .	51
3.35 The Web-UI Inter-Connection diagram . . . . .	53
3.36 The Web-UI Authentication Page . . . . .	56
3.37 The Web-UI Create User Page . . . . .	56
3.38 The Web-UI No-Access Page . . . . .	57
3.39 The Web-UI Control Dashboard . . . . .	57
3.40 The Web-UI Mapping Dashboard . . . . .	59
3.41 The Web-UI Setting Page: Current-User Page . . . . .	61
3.42 The Web-UI Setting Page: Manage-User Page . . . . .	61
3.43 The Web-UI Media and Map Panel . . . . .	62
3.44 Secondary Core . . . . .	63
3.45 Secondary Core connection diagram . . . . .	64
3.46 Secondary Core internal workflow - Flowchart . . . . .	68

3.47 Secondary Core internal workflow - Flowchart cont'd . . . . .	68
3.48 Secondary Core internal workflow - Sequence diagram . . . . .	69
3.49 Secondary Core wiring - Breadboard connection . . . . .	70
3.50 Secondary Core wiring - Schematic diagram . . . . .	71
3.51 Secondary Core Display- BOOT SCREEN . . . . .	73
3.52 Secondary Core Display- POWER-ON SCREEN . . . . .	73
3.53 Secondary Core Display- OPS SCREEN . . . . .	75
3.54 Secondary Core Display- CONFIG SCREEN . . . . .	76
3.55 Secondary Core Captive Portal- JOIN NETWORK . . . . .	77
3.56 Secondary Core Captive Portal- CONFIGURATION page . . . . .	77
3.57 Secondary Core Captive Portal- IP ADDRESS page . . . . .	77
3.58 ATEMR: Project Design Cycle . . . . .	78
4.1 ATEMR: Secondary core ready (on startup) . . . . .	80
4.2 ATEMR: Default map . . . . .	81
4.3 ATEMR: Default velocity limits . . . . .	81
4.4 ATEMR: Nodes run status . . . . .	82
4.5 ATEMR: Setting initial robot pose . . . . .	82
4.6 ATEMR: Setting initial robot pose (initial run : rviz) . . . . .	82
4.7 ATEMR: Setting initial robot pose (pose convergence : rviz) . . . . .	83
4.8 ATEMR: Sending robot goal . . . . .	84
4.9 ATEMR: Sending robot goal (rviz) . . . . .	84
4.10 ATEMR: Map save dialog . . . . .	84
4.11 ATEMR: AMCL Pose Covariance . . . . .	85

# List of Tables

2.1	Wheeled Mobile Robot configurations . . . . .	9
2.1	Wheeled Mobile Robot configurations . . . . .	10
3.1	AEMR: Design specifications . . . . .	23
3.2	AEMR: Design specification metrics . . . . .	24
3.3	AEMR: Electronic components . . . . .	39
3.4	<b>CAN</b> communication protocols . . . . .	65
3.5	POWER ON/OFF (0x150) . . . . .	65
3.6	<b>PRIMARY</b> Core Statuses (0x153) . . . . .	67
5.1	AEMR: Design specifications vs Obtained Objectives . . . . .	88

# List of Abbreviations

<b>ATEMR</b>	<b>A</b> utonomous <b>T</b> elepresence <b>M</b> obile <b>R</b> obot
<b>AMCL</b>	<b>A</b> daptive <b>M</b> onte- <b>C</b> arlo <b>L</b> ocalization
<b>WMR</b>	<b>W</b> heeled <b>M</b> obile <b>R</b> obot
<b>ROS</b>	<b>R</b> obot <b>O</b> perating <b>S</b> ystem
<b>RViz</b>	<b>R</b> obot <b>V</b> isualization
<b>CAN</b>	<b>C</b> ontroller <b>A</b> rea <b>N</b> etwork
<b>LIDAR</b>	<b>L</b> ight <b>D</b> etection <b>A</b> nd <b>R</b> anging

# List of Symbols

$a$	distance	m
$P$	power	W ( $\text{J s}^{-1}$ )
$\omega$	angular frequency	rad
FR	FrontRight	Front Right Obstacle
FL	FrontLeft	Front Left Obstacle
LC	LeftCenter	Left Center Obstacle
RL	RearLeft	Rear Left Obstacle
RR	RearRight	Rear Right Obstacle
RC	RearCenter	Right Center Obstacle



# Chapter 1

## GENERAL INTRODUCTION AND OBJECTIVES

### 1.1 INTRODUCTION

This chapter presents the problem statement, the previous work, scope and objective of this project and finally, describes how this document is structured in general.

### 1.2 PROBLEM STATEMENT

Telepresence is a medium in which transducers, such as video cameras and microphones Adalgeirsson and Breazeal (2010) and Tsui et al. (2012), substitute for the corresponding senses of the participant, Sherman and Craig (2019). That is to say, the user can interact and or affect the remote environment via any action permitted by the medium used. The slightly similar but different terminology to consider for the purposes of this work is teleoperation. According to Sherman and Craig (2019), teleoperation refers to cases where the operator uses a remote device to interact with the environment. While telepresence enables the operator to interact with the remote environment just as they if they were physically present, teleoperation is achieves the same objective but just from a second person point of view. A robotic telepresence offers the benefits of traditional telepresence with the added value of moving and actuating in that location Kristoffersson et al. (2011). Telepresence mobile robotic platforms are being used almost in every sector of life ranging from industrial floors, offices, malls, supermarkets, hospitals and aged/residential homes. These platforms which are sometimes very expensive, provide movement, actuation, video, audio, and touch etc. capabilities depending on its application to the operator. It is necessary however that, the learning curve for integrating and using these mobile platforms be as

seamless as possible when moving from one system to another without sacrificing usability nor robot autonomy. Most platforms either provide telepresence usability or robot autonomy but seldom both on a singular system. Hence, the need for an Autonomous Telepresence Mobile Robot (**ATEMR**) that is low cost, provides novel interactive control to the operator while maintaining complete autonomy, for use in residential homes to provide assistive care and monitoring.

## 1.3 RELATED WORK

### 1.3.1 Robotic Telepresence

The use of mobile robots and robot autonomy for defense/security Kenyon et al. (2005), hazardous environment assessment (reconnaissance), and most recently, for interplanetary exploration (National Aeronautics and Space Administration (NASA)) and as assistive technologies Kristoffersson et al. (2011), Marafa et al. (2020), Michaud et al. (2007), Shiarlis et al. (2015), Tsai et al. (2007), and Tsui et al. (2012) is on the rise mostly due to advancements in artificial intelligence and semi-conductor prices, hence electronic components. The Mars perseverance rover, NASA Science (2020), is a clear application of mobile telepresence. The first robotic telepresence was the Personal Roving Presence (PRoP) Paulos and Canny (1998a) and Paulos and Canny (1998b). Paulos and Canny (1998a) highlights the importance of the human component in this ecosystem. What these systems have in common is that, they can be operated remotely, provide video and or audio feedback to the operator and possess some sort of navigational autonomy.

### 1.3.2 ROS Based Mobile Robots

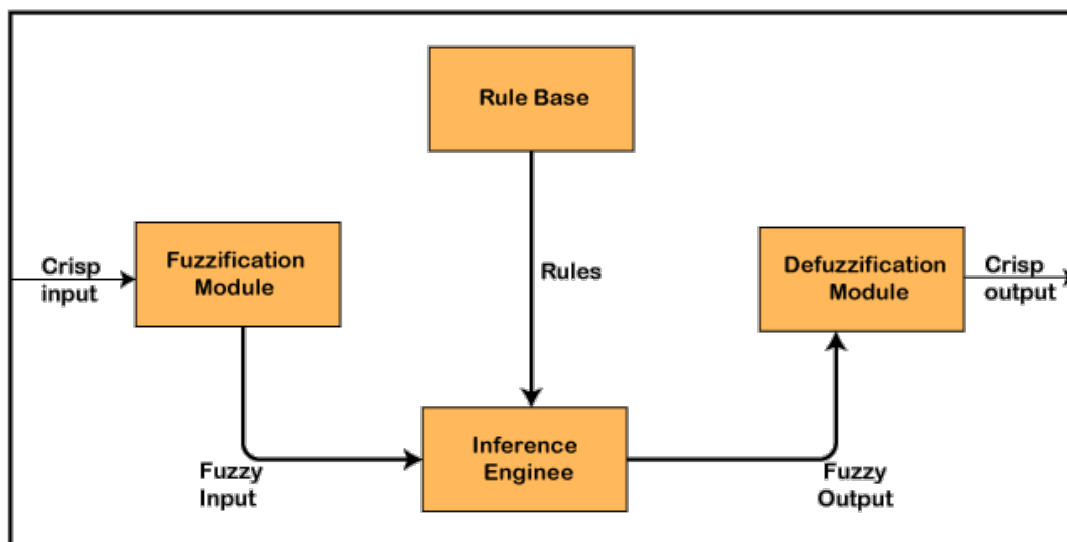
The Robot Operating System (ROS) is rapidly becoming the defacto middleware for robotic projects (Dragičević and Jovanović (n.d.)) mainly to due its comprehensible application programming interface (API), multi-language, peer-to-peer architecture (uses the publisher-subscriber approach), abundant tutorials/forums and its many open source contributors. ROS provides a structured communications layer above the host operating systems of a heterogenous comput cluster, Quigley et al. (2009). In An et al. (2016) a tracked mobile robot was developed that uses the Adaptive monte carlo localization algorithm for robot pose estimation. The mobile base was designed with a dual core 1.8 GHz CPU with 1 GB RAM using ROS-fuerte. The study conducted in Afanasyev et al. (2015) uses the Rao-Blackwellized particle filters and laser data to locate the robot in unknown environment and build a map. The implementation in Ruiz et al. (2013) uses a kinect sensor mounted on top of the platform to



compute velocity vectors for controlling the mobile base. The omni-directional self driving robot designed in Do Quang et al. (2020) was built with a Jetson TX2, an Astra depth camera and an RPLidar sensor and navigates its environment using the data generated from the 3D Astera camera and RPLidar. The follow person robot developed in Priyandoko et al. (2018) uses an Arduino Mega as the base controller and mounts a Kinect camera for person detection and following.

### 1.3.3 Fuzzy Inference Systems in Mobile Robots

A Fuzzy Inference is a method that interprets the values in the input vector and, based on some sets of rules, assigns values to the output vector (Figure 1.1), Kalogirou (2013). Many systems such as washing machines, rice cookers and refrigerators make use of fuzzy logic for their operation.



**Figure 1.1:** Fuzzy Inference System (Source)

The Fuzzy logic techniques implemented in Pradhan et al. (2009), Reignier (1994), and Thongchai et al. (2000) enabled the robot platform to navigate and avoid obstacles successfully with an array of ultrasonic sensors. The Fuzzy Logic Controller (FLC) having a Gaussian membership function is found to be most efficient for mobile robots navigation Pradhan et al. (2009) having tested a trapezoidal and triangular FLC (three membership engine) and a triangular FLC (five membership engine). The rules used were “obtained heuristically using common sense”, Pradhan et al. (2009). The wheeled mobile robot control studied in Rashid et al. (2010) developed a two input two output FLC for controlling the wheel velocities and concluded that a five membership function provides better performance compared to that

of a seven membership function controller mainly due to the time taken to reach the target position.

## 1.4 SCOPE AND OBJECTIVE

The scope of this project is to design a low-cost robotic mobile base capable of autonomous navigation, to aid in assistive care taking of elderly persons in a residential home.

At the end of this study, the following objectives should be achieved:

- (i) A differential drive mobile base running on ROS should be designed and constructed.
- (ii) The mobile base should be capable of navigating autonomously.
- (iii) A Web Interface for robotic telepresence and teleoperation should be created for controlling the mobile base.
- (iv) The mobile base should have an FLC awareness engine for reactive response to unexpected stimuli when both when idle as well as, when being operated manually.
- (v) A “SMACH” based state machine should be created acting as the middle engine between the higher-level web-operated interface, and the lower-level robot motor control interface.

## 1.5 PROJECT STRUCTURE AND ORGANIZATION

This project proposes a structure in five chapters where all the information and workdone to reach the final project objective will be discussed.

Chapter 1 introduces the work to be done, the problem statement, discusses previous related work done in a similar direction and specifies the scope and project objective and finally, outlines the structure and organization of the project.

Chapter 2 shows a general overview of the chosen field and discusses the state of art of robotics systems with respect to autonomous mobile robots, mapping, localization and navigation.

In Chapter 3, the study analyzes the project objectives and identifies the necessary requirements to attain these objectives. It also introduces and describes the implementation of the proposed system and sub-systems.

The prototype implemented previously is test and the obtained results are presented in Chapter 4.

Finally, Chapter 5 compares the obtained results with the project requirement metrics and discusses conclusions and recommendations for future work.

## Chapter 2

# LITERATURE REVIEW

### 2.1 INTRODUCTION

The goal of this chapter is to establish what is considered a robotic system. This chapter will describe the various types of robots and compare their kinematics. The remainder of the chapter will focus on the current state of autonomous mobile robot, mapping, localization and navigation.

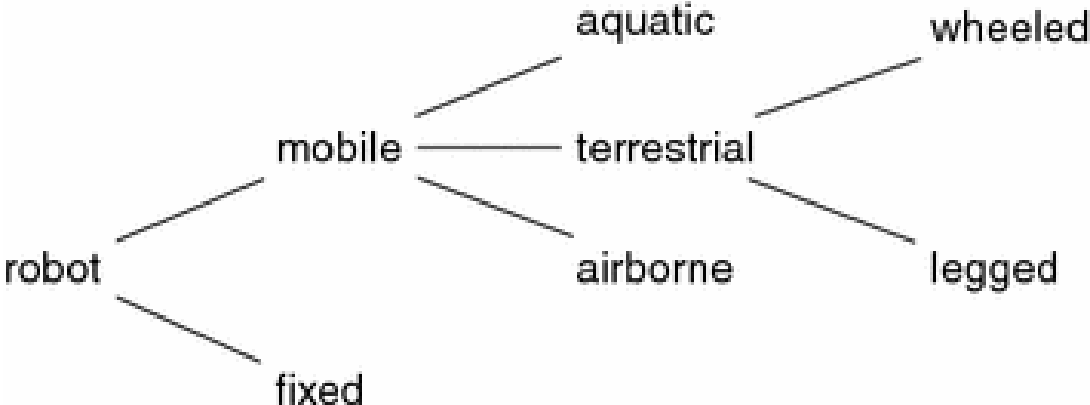
### 2.2 DEFINITION OF TERMS

The mobile base is developed under the project Autonomous Telepresence Mobile Robot (ATEMR) and dubbed the name “KAY”. Anywhere **ATEMR** or **KAY** is used throughout this project, it will refer to the mobile base being developed. SLAM in the context of this work will refer to Simultaneous Localization and Mapping.

### 2.3 WHAT IS A ROBOT?

A robot can be thought of as “a machine - especially one programmable by a computer - capable of carrying out a complex series of actions automatically”, Dictionary (2016). The word robot was first used in a fictional play Rossumovi Univerzální Roboti – Rossum’s Universal Robots (R.U.R) by Karel Čapek in 1920 to mean “forced labour”. Robot can be classified based many categories ranging from the environment in which they operate (Figure 2.1) and their intended application and tasks.

Two (2) most common distinction that stands in classification by the environment in which they operate are fixed and mobile robots. Fixed robots tend to be industrial grade robotic manipulators (Figure 2.2) stationed in a work cell, working alone and or in unison with other



**Figure 2.1:** Robot Classification by Operating Environment (Ben-Ari and Mondada (2018))

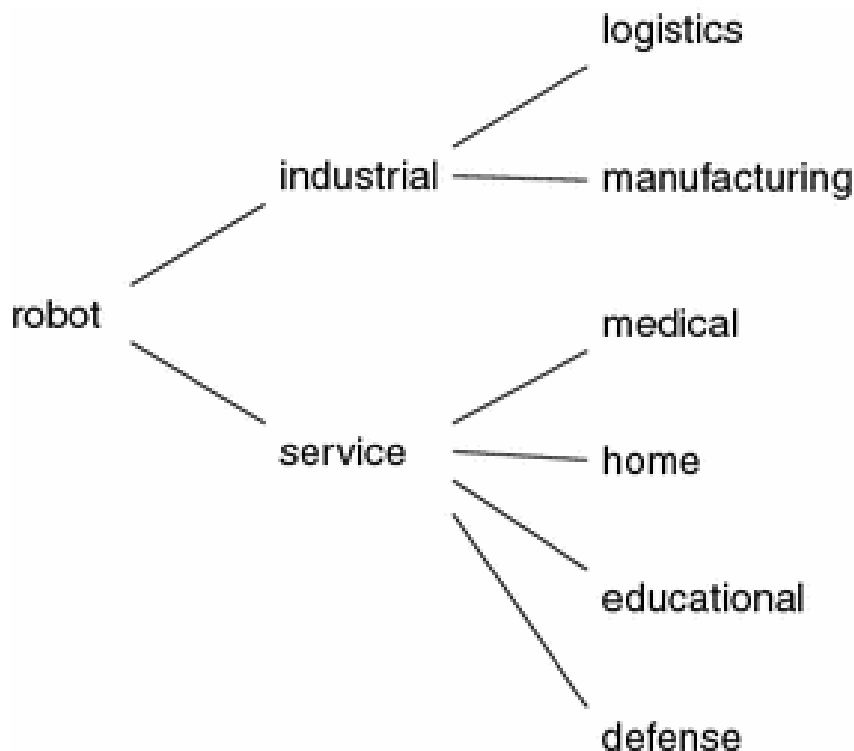
robots or robot operators. This type of robots are mainly used for heavy machinery handling (e.g pick and place in a vehicle manufacturing plant), repetitive and precision work such as painting, sorting, vehicle loading and unloading and welding.



**Figure 2.2:** Pick and Place foundry automation (Bachmann, KUKA Roboter GmbH (2021))

Mobile robots on the other hand, “are expected to move around and perform tasks in large,

ill-defined and uncertain environments that are not specifically for robots” (Ben-Ari and Mondada (2018)). Referring back to the diagram 2.1, mobile robots can be further categorised into aquatic, airborne and terrestrial types. The terrestrial type mobile robots branches into legged and wheeled robots mainly due to ease of locomotion in the chosen operating environment. The next category of robot classification is by the intended application and task performed, Figure 2.3.



**Figure 2.3:** Pick and Place foundry automation (Ben-Ari and Mondada (2018))

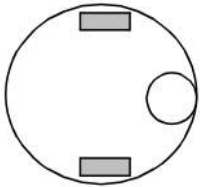
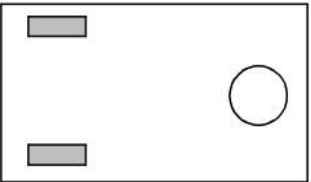
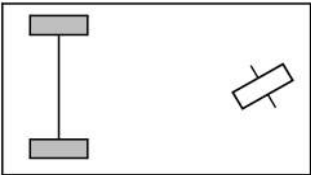
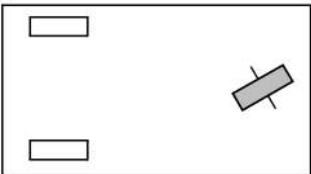
Industrial robots like has been already discussed, are designed for a specific task in a well defined environment requiring little to no human supervision. Service robots on the other hand, perform tasks that are usually in sync with human collaboration. These include house hold chores by vacuum cleaners, assistive care (Adalgeirsson and Breazeal (2010), Marafa et al. (2020), Shiarlis et al. (2015), Tsai et al. (2007), and Tsui et al. (2012)), surgical robots, rehabilitation, defense (Kenyon et al. (2005)) and training etc.

This study will delve into further detail on wheeled mobile robots and their kinematic models.

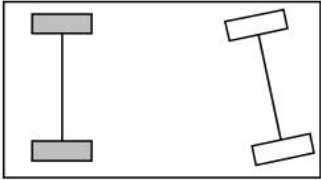
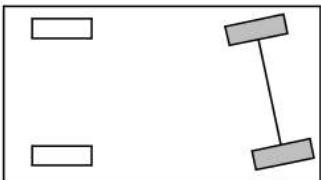
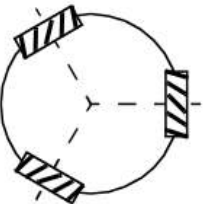
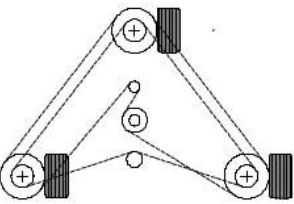
### 2.3.1 WHEELED MOBILE ROBOT (WMR) KINEMATICS

Mobile robots come in many different configurations depending on the application. They are mainly configured as differential drive, tricycle, ackermann steering and multiwheel omnidirectional and synchro drives (Siegwart et al. (2011) and Tzafestas (2013)). The table 2.1 presents the various configurations and their description.

**Table 2.1:** Wheeled Mobile Robot configurations

Configuration	Description	Example
	Two-wheel centered differential drive with a third point of contact or Two independently driven wheels in the rear/front, 1 unpowered omnidirectional wheel in the front/rear	Nomad Scout, smartRob EPFL, ATEMR
		
	Two connected traction wheels (differential) in rear, 1 steered free wheel in front or Two free wheels in rear, 1 steered traction wheel in front	Piaggio minitrucks, Neptune (Carnegie Mellon University), Hero-1
		

**Table 2.1:** Wheeled Mobile Robot configurations

Configuration	Description	Example
 	<p>Two motorized wheels in the rear, 2 steered wheels in the front; steering has to be different for the 2 wheels to avoid slipping/skidding or, Two motorized and steered wheels in the front, 2 free wheels in the rear; steering has to be different for the 2 wheels to avoid slipping/skidding.</p>	<p>Car with rear-wheel drive, Car with front-wheel drive</p>
	<p>Three motorized Swedish or spherical wheels arranged in a triangle; omnidirectional movement is possible</p>	<p>Stanford wheel Tribolo EPFL, Palm Pilot Robot Kit (CMU)</p>
	<p>Three synchronously motorized "Synchro drive" and steered wheels; the orientation is not controllable</p>	<p>Denning MRV-2, Georgia Institute of Technology, I-Robot B24, Nomad 200</p>

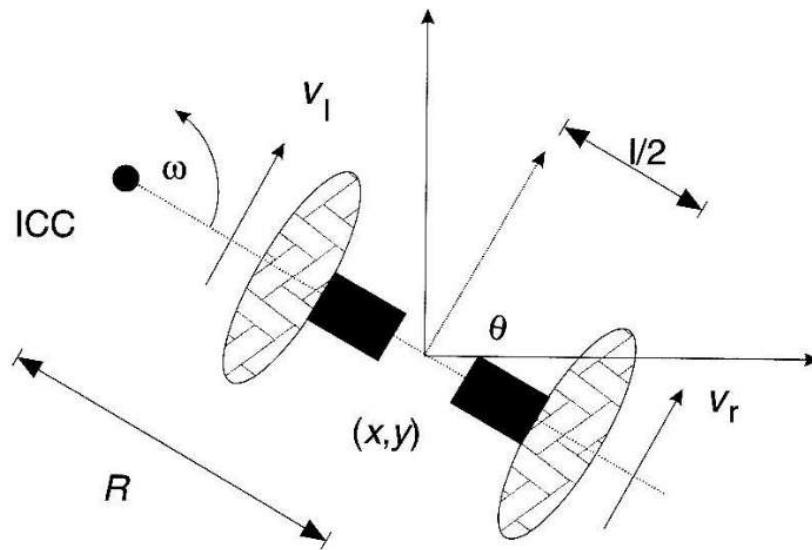
NOTE: The table 2.1 is taken from Siegwart et al. (2011).

The ATEMR project implements the differential drive configuration due to its simplicity and having constraints that lie within the limits of the chosen operational environment

### 2.3.2 DIFFERENTIAL DRIVE MOBILE BASE

The differential drive configuration basically consists of two (2) driven wheels mounted on a common axis, and each wheel can independently be driven either forward or backward, Dudek and Jenkin (2010). For the robot to perform a rolling motion, it must rotate about a point that lies along their common left and right wheel axis known as the *Instantaneous Center of Curvature (ICC)* (refer to Figure 2.4).





**Figure 2.4:** Differential Drive kinematics (from Dudek and Jenkin (2010))

Let:

- $\omega$ : Rate of rotation about the ICC
- $R$ : The the signed distance from the ICC to the midpoint between the wheels
- 

$$V_l$$

and

$$V_r$$

: Represent the left and right wheel velocities respectively

- $(x, y)$ : Be some assumed robot position
- $\theta$ : Assumed robot heading

Then, the **forward kinematics** of the robot can be given by equation 2.1:

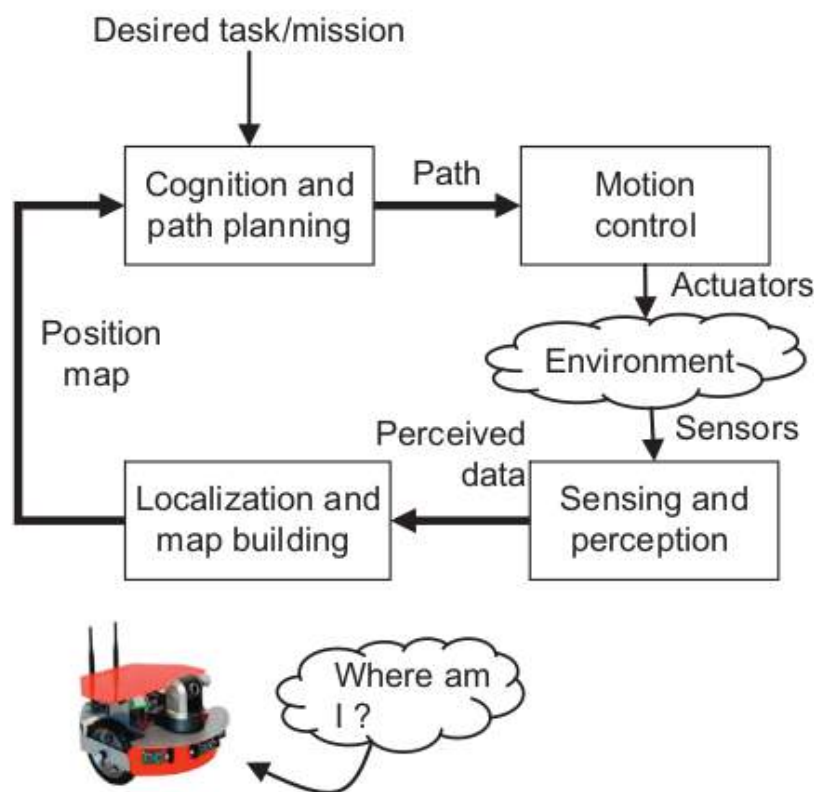
$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ 0 \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{bmatrix} \quad (2.1)$$

Equation 2.1 simply describes the motion of a robot rotating a distance  $R$  about its ICC with an angular velocity of  $\omega$ , robotkinematics (p.41). Thus figuring out the robot's pose given its dimensions and wheel velocities. The opposite and much related question is, how can the

robot base be controlled to reach a given set of coordinates. This is known as the **inverse kinematics** problem is employed by autonomous navigation algorithms to plan paths and translate the robot base.

## 2.4 AUTONOMOUS MOBILE ROBOT

Figure 2.5 is an illustration of the interrelations and functions of the fundamental capabilities of an integrated robotic system.



**Figure 2.5:** General organization of the functions of an autonomous mobile robot (Siegwart et al. (2011))

Basically, an autonomous must be able to perceive the environment via its sensors in order to create the proper assumption of where it is in the world: localization, determine how to get to a given target objective (path planning) and traverse the terrain from its current position to said objective while (or not) avoiding any obstacles in its path: navigation and obstacle avoidance.

Before discussing the fundamentals components however, autonomous robots come with a plethora of sensors based on the application and operating environment that allow it

to take input from the external environment for its internal computations. The next section discusses some of the most common sensors found in an autonomous mobile robot system.

Robotic sensors are generally distinguished into two main types; *Analog sensors* provide analog output signals and requires an analog-to-digital (A/D) conversion. *Digital sensors* however are more accurate when compared to their analog counterparts and can have different forms of outputs: serial or parallel form. According to Tzafestas (2013), the desired sensors features are high resolution, wide operation range, fast response, easy calibration, high reliability and low cost.

### 2.4.1 ENCODER

Encoders convert motion (linear or rotary) to an electrical signal that can be read by some type of control device in a motion control system, Company (2021). They mainly distinguished into absolute and incremental types. An Incremental encoder does not measure the specific position of the device but rather outputs a steady stream of high and low signals and are used to monitor repetitive movement in the same direction, Figure 2.6. An Absolute encoder however, provides the true position of the device with high resolution.



**Figure 2.6:** An Incremental Rotary Encoder (Lambtron (2018))

### 2.4.2 INERTIAL MEASUREMENT UNIT (IMU)

For mobile robots such as the focus for this study, the reported odometry feedback isn't always as accurate as desired due to slippage, uneven floors etc. and hence the need for an inertial measurement unit (IMU) to help better gauge the mobile base movement in its environment. An IMU can be thought of as an electronic device that measures and reports a body's specific force, angular rate, and sometimes the orientation of the body, using a combination of accelerometers, gyroscopes, and sometimes magnetometers (Figure 2.7) and

are often incorporated into Inertial Navigation Systems which utilize the raw IMU measurements to calculate attitude, angular rates, linear velocity and position relative to a global reference frame. Johnson (2011). High performance IMUs, or IMUs designed to operate under harsh conditions, are very often suspended by shock absorbers.



**Figure 2.7:** Apollo IMU stable member (Reinhold (2019))

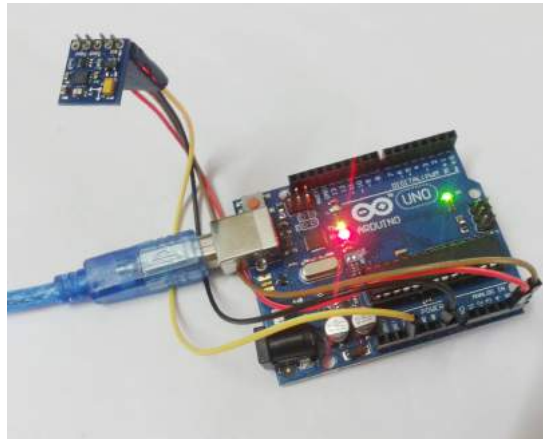
### 2.4.3 GYROSCOPE

A gyroscope is a device used for measuring or maintaining orientation and angular velocity. A classic gyroscope is a spinning wheel or disc in which the axis of rotation (spin axis) is free to assume any orientation by itself. When rotating, the orientation of this axis is unaffected by tilting or rotation of the mounting, according to the conservation of angular momentum, Oxford Dictionary (2022).

Gyroscopes come in different forms and have other operating principles such as the microchip-packaged MEMS gyroscopes found in electronic devices (sometimes called gyrometers, Figure 2.8), solid-state ring lasers, fibre optic gyroscopes, and the extremely sensitive quantum gyroscope, (Wheatstone (1879)).

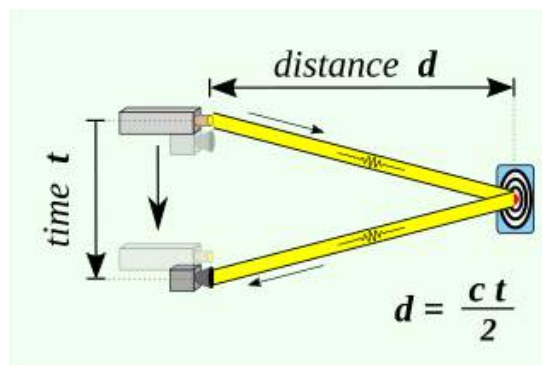
### 2.4.4 LIGHT DETECTION AND RANGING (LIDAR) SENSOR

Lidar is a method for determining ranges (variable distance) by targeting an object with a laser and measuring the time for the reflected light to return to the receiver (Pfeifer and Brieser (2007)), see Figure 2.9 where 'c' is the speed of light .Lidar can also be used to make digital 3-D representations of areas on the earth's surface and ocean bottom, due to differences in laser return times, and by varying laser wavelengths. It has terrestrial, airborne, and mobile applications. Lidar uses ultraviolet, visible, or near infrared light to



**Figure 2.8:** A digital gyroscope module connected to an Arduino Uno board (from Rahat (2014))

image objects, Willing (2021). These systems normally are composed of MEMS sensors, Photodetector and receiver electronics, Phased arrays and 600-1000nm lasers.



**Figure 2.9:** Time of flight principle (from RCraig09 (2020))

The proposed LIDAR sensor unit for this work is the RPLIDAR A2M8 with a resolution of 0.9° and a usable range of 8 meters.

## 2.4.5 ULTRASONIC SENSOR

An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity, MaxBotix (2022). Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. They are mainly used in the detection of presence, level, position and distance.

## 2.4.6 INFRARED SENSOR

An infrared sensor (IR sensor) is a radiation-sensitive optoelectronic component with a spectral sensitivity in the infrared wavelength range 780 nm ... 50  $\mu\text{m}$ , CRISP (2001) and INFRADEC (2022). Infrared radiation is used in industrial, scientific, military, commercial, and medical applications. Night-vision devices using active near-infrared illumination allow people or animals to be observed without the observer being detected. Military and civilian applications include target acquisition, surveillance, night vision, homing, and tracking (Alexander Chilton (2014)).

The fundamental components (mapping, localization and navigation) of autonomous mobile robots will be discussed in the following.

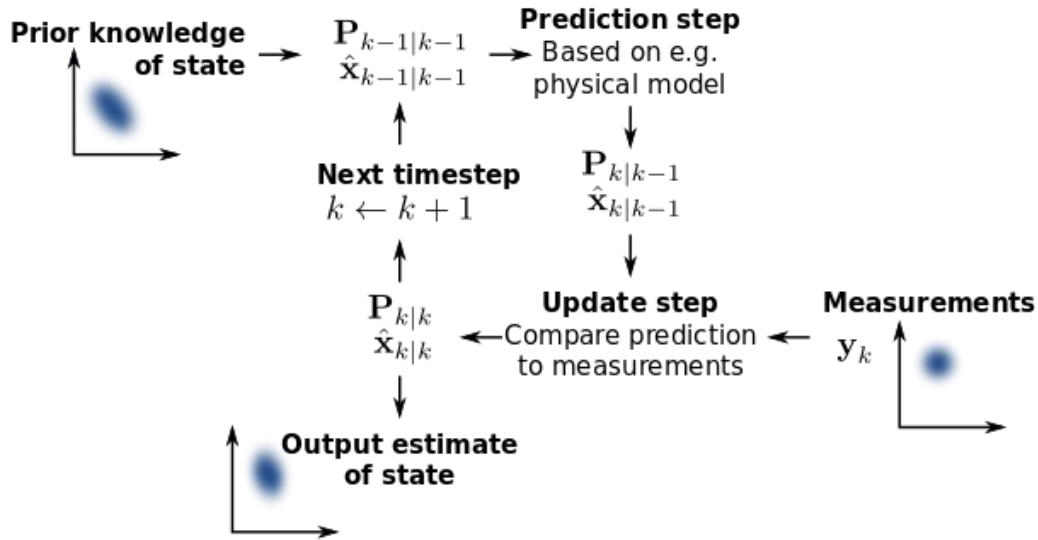
## 2.4.7 MAPPING

A robot's ability to localize itself based on information about the position of obstacles or other information on the environment (Ben-Ari and Mondada (2018)) is made possible by maps. Robotic maps use arrays of discretized cells to represent a topological world, also known as grid maps. The necessary information required for localization is collected and neatly packaged through a process known as *Mapping*. To build a map however requires the robot to know where it is in the environment; localization, and for the robot to localize itself, it requires a map to do this. This cyclic dependency issue is solved by Simultaneous Localization and Mapping (SLAM) algorithms (Bailey and Durrant-Whyte (2006)). SLAM algorithms are used to construct a map of an unknown environment while simultaneously keeping track of the robot's location within it. Some algorithms that use this technique are the particle filter, extended Kalman filter (Willner et al. (1976)), covariance intersection and GraphSLAM (Grisetti et al. (2010)).

## KALMAN FILTER

The Kalman Filter named after Rudolf E. Kálmán uses a series of measurements observed over time, including statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe, Chui, Chen, et al. (2017) and Grewal et al. (2020). The sensor input for ATEMR like the Inertial Unit has low precision and hence the data coming from this unit will have to be processed by a Kalman Filter node before it can be used by the rest of the system. The

algorithm shown in Figure 2.10 has found applications in guidance, navigation, and control of vehicles.



**Figure 2.10:** Kalman Filter algorithm (from Petteri Aimonen (2011))

Kalman filtering uses a system's dynamic model (e.g., physical laws of motion), known control inputs to that system, and multiple sequential measurements (such as from sensors) to form an estimate of the system's varying quantities (its state) that is better than the estimate obtained by using only one measurement alone. As such, it is a common sensor fusion and data fusion algorithm.

$$x_k = f(x_{k-1}, u_k) + w_k \quad (2.2)$$

$$z_k = h(x_k) + v_k \quad (2.3)$$

Where;

- $x_k$ : Current state of the model at time, k
- $x_{k-1}$ : Previous state of the model
- $h_{x_k}$ : Observation of the current state at time, k
- $h_{x_{k-1}}$ : Observation of the previous state
- $z_k$ : Current measurement at time, k
- $v_k$ : Measurement noise at time, k

In the extended Kalman filter (EKF), the state transition equation 2.2 and observation models equation 2.3 need not be linear functions of the state but may instead be nonlinear functions.

## PARTICLE FILTER

Particle filtering uses a set of particles (also called samples) to represent the posterior distribution of some stochastic process given noisy and/or partial observations. The state-space model can be nonlinear and the initial state and noise distributions can take any form required. Particle filters update their prediction in an approximate (statistical) manner. The samples from the distribution are represented by a set of particles; each particle has a likelihood weight assigned to it that represents the probability of that particle being sampled from the probability density function. Weight disparity leading to weight collapse is a common issue encountered in these filtering algorithms; however it can be mitigated by including a resampling step before the weights become too uneven. Several adaptive resampling criteria can be used, including the variance of the weights and the relative entropy with respect to the uniform distribution. In the resampling step, the particles with negligible weights are replaced by new particles in the proximity of the particles with higher weights, Del Moral et al. (2011). Refer to Gustafsson (2010) for further details. The AMCL (Gerkey (2020)) ROS package that will be used for localization for the purposes of this project, implements given a map and an initial state estimate.

### 2.4.8 LOCALIZATION

Localization is the process that allows the robot to determine its position relative to an external reference called a *landmark*, Ben-Ari and Mondada (2018). Landmarks can be lines on the ground, doors in corridor, text/ image on a wall or a beacon etc. The process involved can be visualized in Figure 2.11. Depending on the type of landmark used, they can be classified as active artificial, passive artificial and natural landmarks.

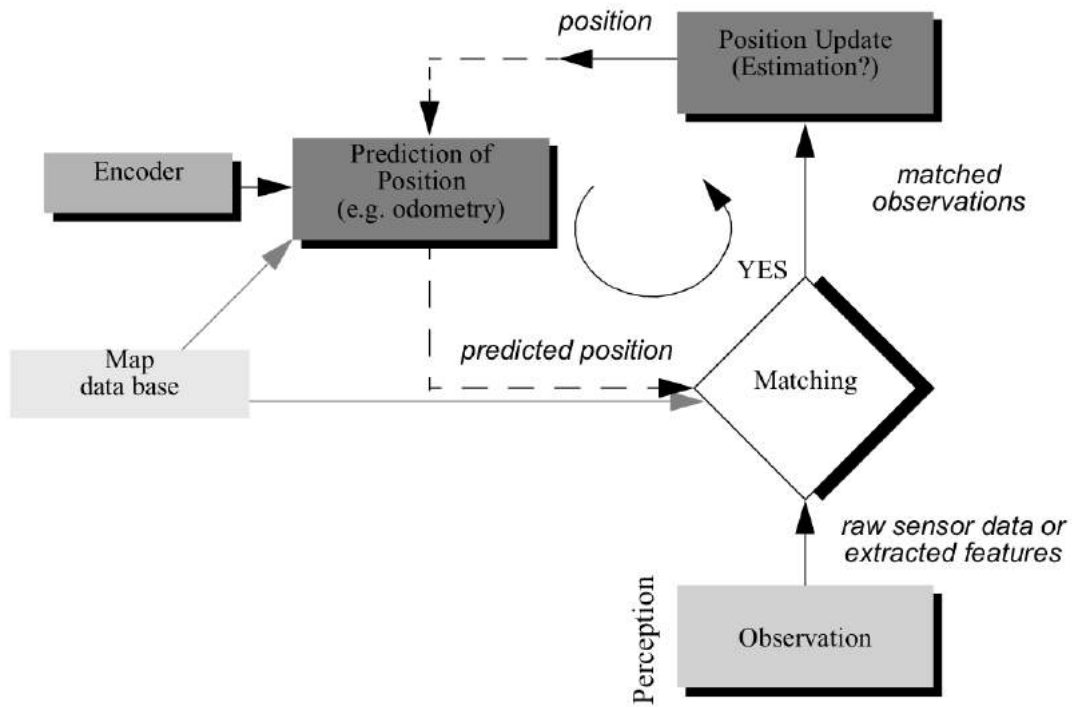
“An active landmark emits some kind of signal to enable easy detection. In general, the detection of a passive landmark needs more of the sensor. To simplify the detection of passive landmarks, we may use an active sensor together with a passive landmark that responds to this activeness. For example, use of bar codes on walls and visual light to detect them for building topological maps.”

(Tzafestas (2013)).

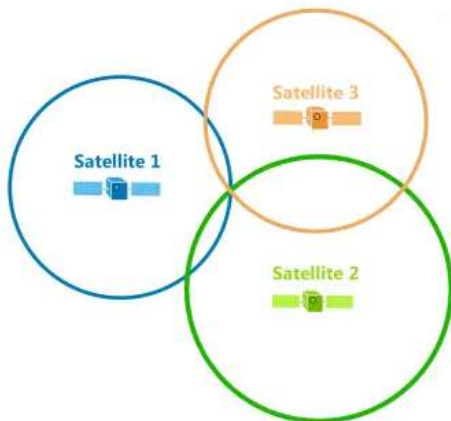
Localization using active landmarks is distinguished into trilateration Figure 2.12 and triangulation Figure 2.13 , (Tzafestas as cited in Borenstein et al. (1997)).

- *Trilateration* involves measuring distances. Using three distances, trilateration can pinpoint a precise location. In the case of a *Global Positioning System (GPS)* each

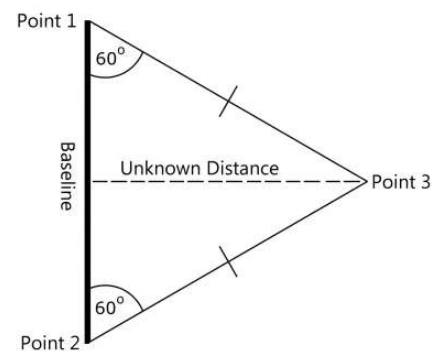




**Figure 2.11:** General schematic for mobile robot localization (from Siegwart et al. (2011))



**Figure 2.12:** Localization by Trilateration (from GIS Geography (2021))



**Figure 2.13:** Localization by Triangulation (from GIS Geography (2021))

satellite is at the center of a sphere and where they all intersect is the position of the GPS receiver.

- *Triangulation* involves measuring angles.

Furthermore, even though localization by GPS is of no essence to this work, it is best to touch on it since location determination in recent years has been made easier and more accurate

with the introduction of the *Global Positioning System (GPS)*. This kind of localization is based on satellites orbiting the Earth and commercial systems are accurate to roughly 10m. The accuracy provided by GPS systems maybe enough depending on the robotic application but is not reliable when used in an indoor environment. Robots can estimate their relative position in the environment based on the robot's motion given a known starting position through a process known as probabilistic localization.

## 2.4.9 NAVIGATION

Navigation in the context of mobile robot is mainly based on path planning or trajectory generation. According to Siegwart et al. (2011), the purposeful decision-making and execution that a system utilizes to achieve its highest-order goals directly linked to robust mobility is *navigation competence*. Navigation is the fusion of mapping and localization previously discussed, enabling the robot to act based on available data to reach a given goal or target position efficiently.

### PATH PLANNING

Robot path planning usually occurs in a representation called *configuration space*. A configuration describes the pose of the robot, and the configuration space  $C$  is the set of all possible configurations, LaValle (2006).



**Figure 2.14:** Configuration space of a point-sized robot. White =  $C_{free}$ , gray =  $C_{obs}$   
(from Simeon87 (2008))

With reference to Figure 2.14, the three terminologies used in path planning are:

- *Free Space* The set of configurations that avoids collision with obstacles is called the free space  $C_{free}$ . The complement of  $C_{free}$  in  $C$  is called the obstacle or forbidden region.
- *Target Space* Target space is a subspace of free space which denotes where we want the robot to move to.
- *Target Space* is a space that the robot can not move to,  $C_{obs}$ .

Some algorithms designed for robot path planning include:

- *Grid-based search* Grid-based approaches overlay a grid on configuration space and assume each configuration is identified with a grid point. At each grid point, the robot is allowed to move to adjacent grid points as long as the line between them is completely contained within  $C_{free}$  (this is tested with collision detection). This discretizes the set of actions, and search algorithms (like A\*) are used to find a path from the start to the goal.
- *Interval-based search* These approaches are similar to grid-based search approaches except that they generate a paving covering entirely the configuration space instead of a grid. The paving is decomposed into two subpavings  $X, X^+$  made with boxes such that  $X \subset C_{free} \subset X^+$ .
- *Geometric algorithms* With these types of algorithms, given a bundle of rays around the current position attributed with their length hitting a wall, the robot moves into the direction of the longest ray unless a door is identified. Such an algorithm was used for modeling emergency egress from buildings.
- *Artificial potential fields* One approach is to treat the robot's configuration as a point in a potential field that combines attraction to the goal, and repulsion from obstacles. The resulting trajectory is output as the path.

$$U(q) = U_{att}(q) + U_{rep}(q) \quad (2.4)$$

Where  $U(q)$  is the potential field,  $U_{att}(q)$  attractive potential field of target and  $U_{rep}(q)$ , repulsive potential field of obstacles, Siegwart et al. (2011).

- *Sampling-based algorithms* Sampling-based algorithms represent the configuration space with a roadmap of sampled configurations. A basic algorithm samples  $N$  configurations in  $C$ , and retains those in  $C_{free}$  to use as milestones. A roadmap is then constructed that connects two milestones  $P$  and  $Q$  if the line segment  $PQ$  is completely in  $C_{free}$ . Again, collision detection is used to test inclusion in  $C_{free}$ . To find a path that connects  $S$  and  $G$ , they are added to the roadmap. If a path in the roadmap links  $S$

and  $G$ , the planner succeeds, and returns that path. If not, the reason is not definitive: either there is no path in  $C_{free}$ , or the planner did not sample enough milestones. An example is the global planner implemented in To et al. (2022).

- *A\* search algorithm* is an informed search algorithm, or a best-first search, meaning that it is formulated in terms of weighted graphs: starting from a specific starting node of a graph, it aims to find a path to the given goal node having the smallest cost (least distance travelled, shortest time, etc.). It does this 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\* selects the path that minimizes the cost of the path. In Piskorskii et al. (2020), the optimized A\* algorithm allowed the robot to take into consideration additional information, estimate the path length and computational time.

$$f(n) = g(n) + h(n) \quad (2.5)$$

- *Dijkstra's algorithm* For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used as demonstrated in Wang et al. (2011), for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined.

NOTE: Sourced from LaValle, 2006.

Two main types of planners are needed for a successful path planning and following. The local and global planners. The global planner is concerned with plotting a path from the robot's current position to a specified goal position with reference to a static map. The local planner on the other hand plans mini trajectories that follow the global planner output closely but also take into account immediate obstacles that did not exist in the static map. In the case of the proposed prototype, this means, two paths will be generated for the robot when a target is specified. The former being the global path and the latter, the local path. While the global path may change less or not at all depending on the environment changes after a plan has been generated, the local path is always being modified to account for obstacles in the robot's path as it approaches the target destination. In Darweesh et al. (2017) the authors developed a planner called OpenPlanner that incorporated global and local planners for navigation and obstacle avoidance but also added a behavior state generator for path tracking, object following, emergency stopping etc.

## Chapter 3

# MOBILE ROBOT ANALYSIS AND DESIGN

### 3.1 INTRODUCTION

This chapter will identify the requirements of the mobile robot base and operator necessities. Specifications to the requirements and necessities will be outlined. The mobile base design will be split into subsystems and discussed into detail and finally, the subsystems discussed previously will be put together to form a single mega system.

#### 3.1.1 Identification of requirements

Based on the scope (refer to SCOPE AND OBJECTIVE) of this work, the technical requirements of the system can be specified in Table 3.1.

**Table 3.1:** ATEMR: Design specifications

<b>Sequence</b>	<b>Requirement/ Specification</b>
1	Move to the specified target in an indoor environment
2	Avoid obstacles when navigating to goal
3	Provide visualization from robot's front camera
4	A Web-based operator-centric graphical interface
5	A low-level self-awareness control feature
6	A telescopic mast for holding the Human Machine Interface
7	Autonomous navigation and remote control

### 3.1.2 Definition of requirements

Once the requirements have been identified, the system parameters will be outlined in this section. The definition of requirements represented in Table 3.2 will serve as the metric against which success will be measured.

**Table 3.2:** ATEMR: Design specification metrics

Sequence	Metric	Magnitude	Unit
1, 7	Ground clearance	48	mm
1, 2, 7	Average Maximum Linear velocity	1.6	m/s
1, 2, 5, 7	Average Slowdown Linear velocity	0.35	m/s
1, 2, 5, 7	Obstacle Slowdown Distance	0.56	m
1, 2, 5, 7	Obstacle Stop Distance	0.28	m
1, 2, 5	Idle transition time	8	s
4, 7	Number of tabs on web interface	3	-
3, 4, 7	Number of visualization cameras	2	-
4, 7	Operator access privileges	YES	-
6	Mast capable of housing a 10-inch tablet	YES	-
3, 4	Minimum stream resolution	640x480	pixels
1, 2, 5	Number of staircase detection sensors	3	-
3, 4	Image/ Video capture and recording	YES	-
3, 4	Image, Video and Map management	YES	-
1, 2, 3, 4, 5, 6, 7	Operating voltage	29.4	V

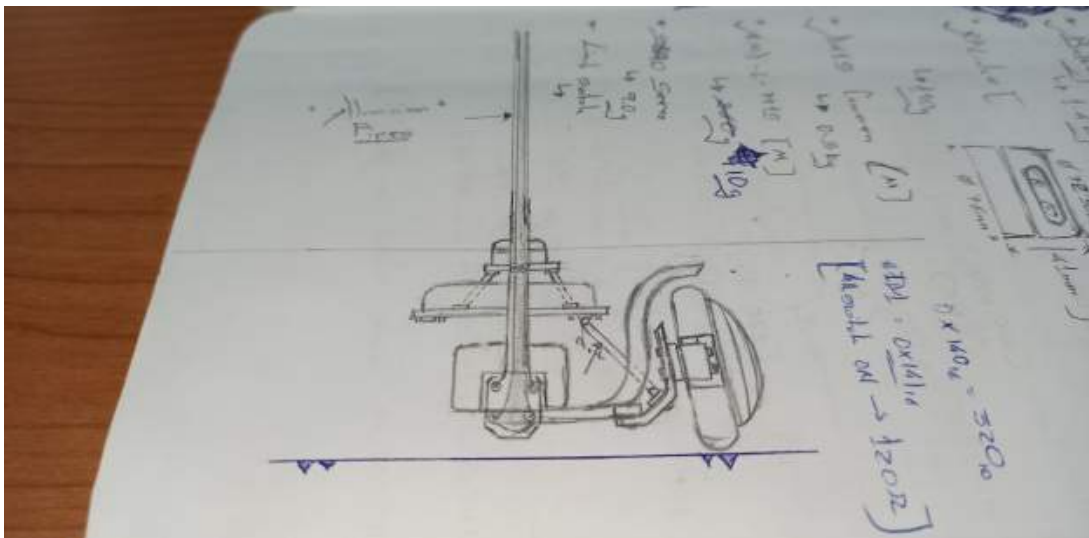
## 3.2 THE MOBILE BASE HARDWARE

In order to build a fully telepresence mobile robot for this purposes of the realization of this project, the robot should be able to move from point A to point B either on its own (autonomously) when commanded to do so or, by responding to manual controls coming from an operator (manual). The robot base is fitted with a specialized payload. An interactive tablet to serve as a Human Machine Interface (HMI), weighing approximately (450-500)g will be fitted at an about one (1) metre height on top of the mobile base. Gandhinathan and Joseph, 2019 states that the robot base should ensure it is electromechanically stable so that is has enough torque to pull its own load, along with the rated payload, and move smoothly with fewer jerks and with marginal pose error (p.87).

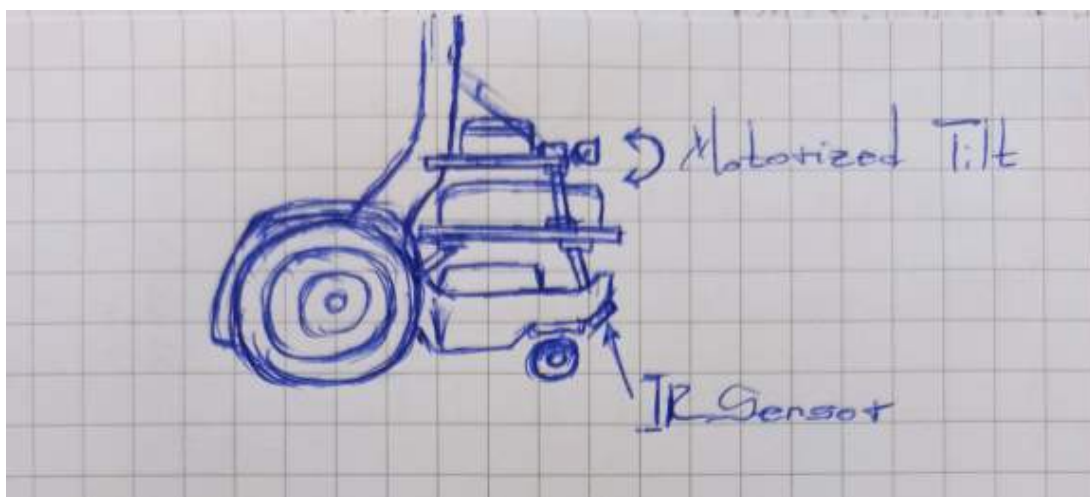
This next section will discuss the construction of the mobile base from conception of the ideas right through to the final accepted design.

### 3.2.1 DESIGN AND CONSTRUCTION OF THE MOBILE BASE

Having outlined the necessary requirements for the prototype, the next step is to come up with the actual idea fitting for the intended application, assistive care. The robot should be small enough to fit through standard doors which are normally 2040 x 826 mm wide in Europe. And have enough torque to carry a payload of up to 5kg.

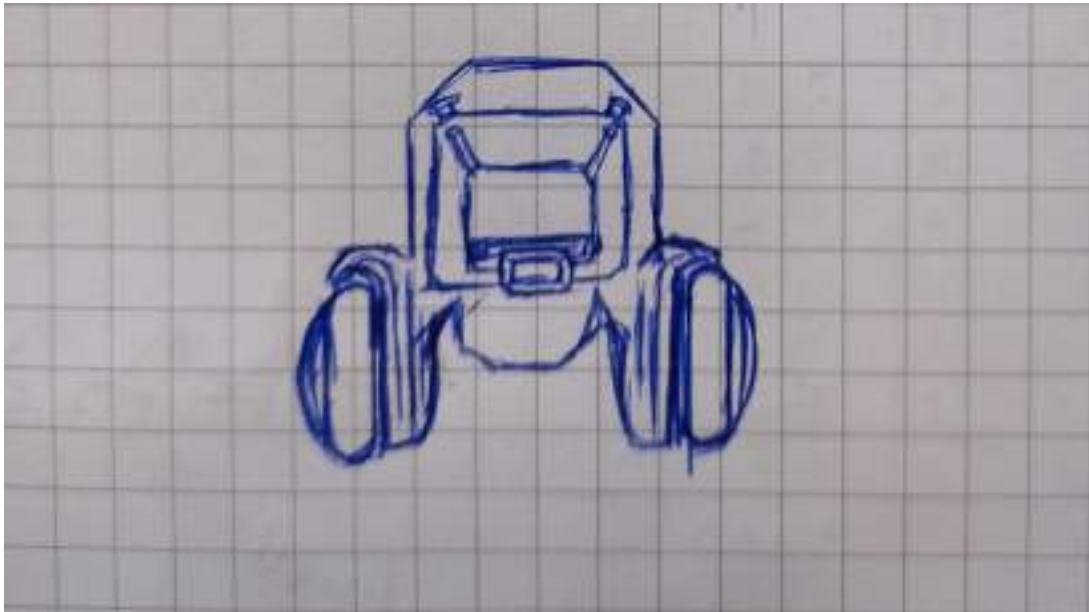


**Figure 3.1:** ATEMR: Mobile Base Design - Raw sketch 1 FRONT VIEW



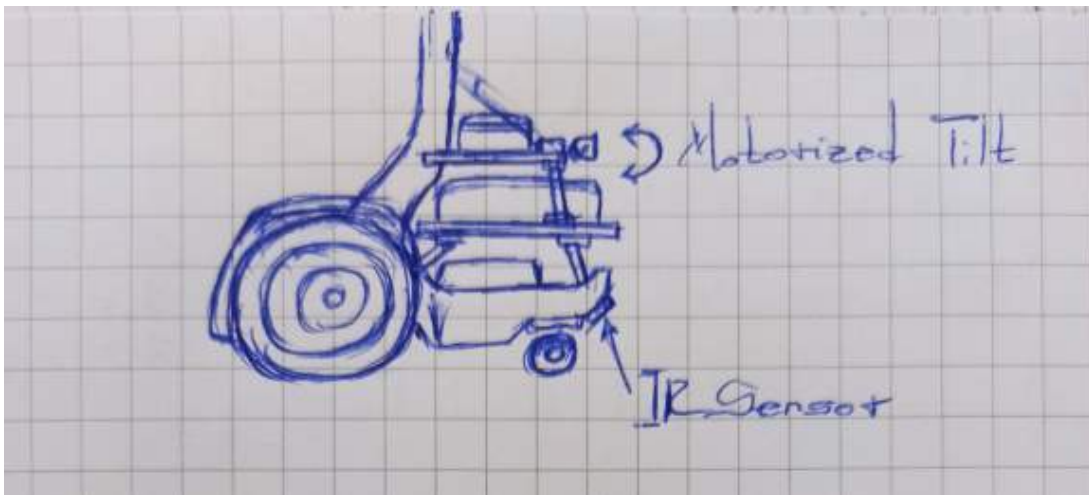
**Figure 3.2:** ATEMR: Mobile Base Design - Raw sketch 1 RIGHT VIEW

The first proposed design Figures 3.1, 3.2 and 3.3 had a small enough width to fit through doors but left all other electronic systems exposed. The motorized wheels were positioned in



**Figure 3.3:** ATEMR: Mobile Base Design - Raw sketch 1 TOP VIEW

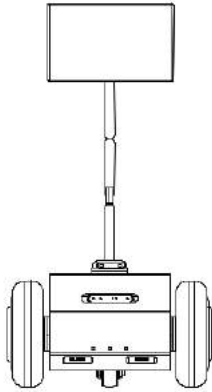
the rear and the caster wheel in front of the mobile base, adding uncertainty to the stability of the base when navigating and a lower ground clearance.



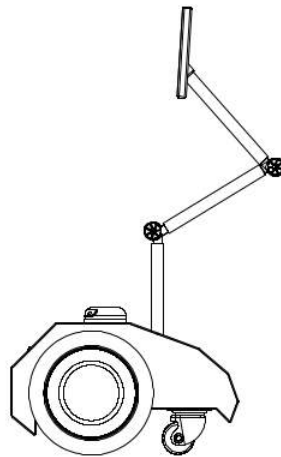
**Figure 3.4:** ATEMR: Mobile Base Design - Raw sketch RIGHT VIEW

The subsequent sketch iteration provided Figure 3.4 upon which the 3D design will be based. This design solved the issue of exposed electronics, uncertainty with regards to positioning of the motorized wheels (front) and had a much higher ground clearance. The next step in the design process is the 3D design using Solidworks. Figures 3.5, 3.6 and 3.7 shows the front, left and isometric views of version 1 of the mobile base inspired by the approved sketch in Figure 3.4.





**Figure 3.5:**  
ATEMR: Mobile  
Base Design - 3D  
design v1 FRONT  
VIEW



**Figure 3.6:**  
ATEMR: Mobile  
Base Design - 3D  
design v1 LEFT  
VIEW

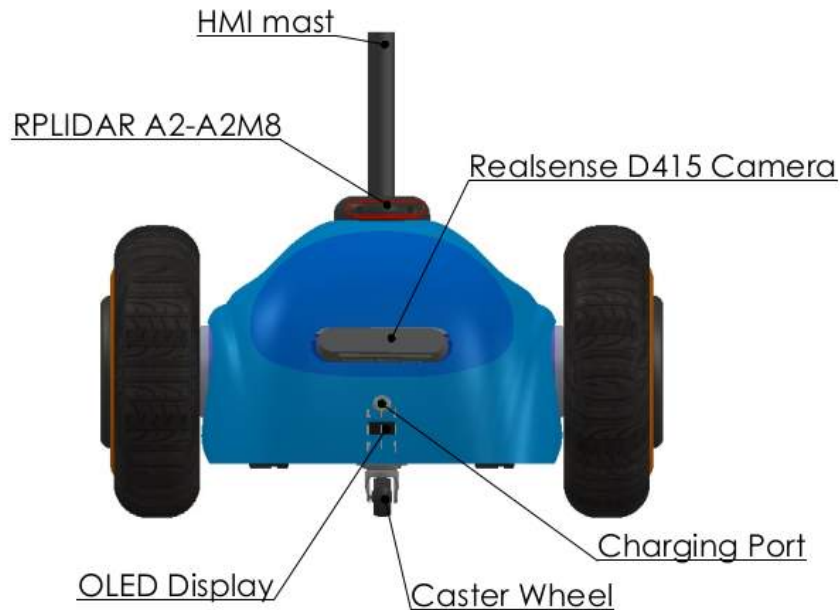


**Figure 3.7:**  
ATEMR: Mobile  
Base Design -  
3D design v1  
ISOMETRIC VIEW

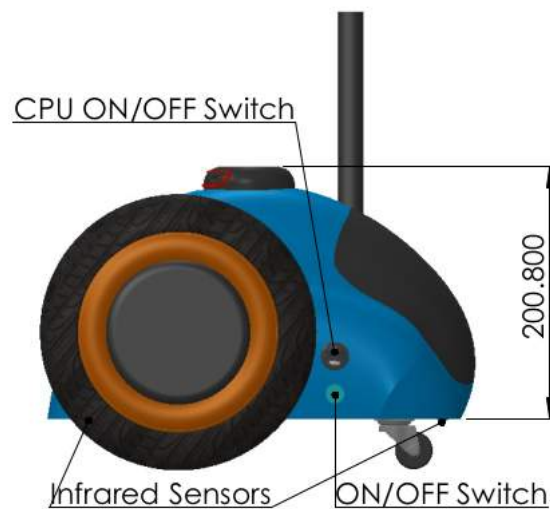
Upon subsequent analysis of this model however, three (3) things stood out and needed to be changed. The first is the positioning of the infrared sensor. With the current design, even though the sensor's angle placement allows it to take measurement a certain distance ahead and behind the mobile base, it is also easily exposed to sun rays that may come in through open windows. This can mess with the data output. The second issue has to do with the bends at the front and rear of the mobile base. The introduction of these bends lowers the overall ground clearance drastically and might have made it not suitable for environments with thicker carpets on the floor. The third issue with this design is the chassis coupling mechanism. There seem to be no clear way of taking this design apart one piece at a time in order to access a particular area on the inside like the battery and front camera.

Figures 3.8, 3.9 and 3.10 shows a much improved version of the previous design. Aesthetics aside, this design has a flat base (higher ground clearance), the infrared positioning issue due to the angle of incidence of incoming sun rays is solved. The infrared sensors are positioned underneath the mobile base and tilted at angle for a wider range sensing. This final design is modular in the sense that, the front camera, battery and LIDAR sensor can be reached separately without necessarily taking apart the whole chassis.

Furthermore, a look at the exploded assembly in Figure 3.11 shows an internal skeletal system (green colored) that adds robustness to the chassis while also serving as an anchor point for the external chassis.



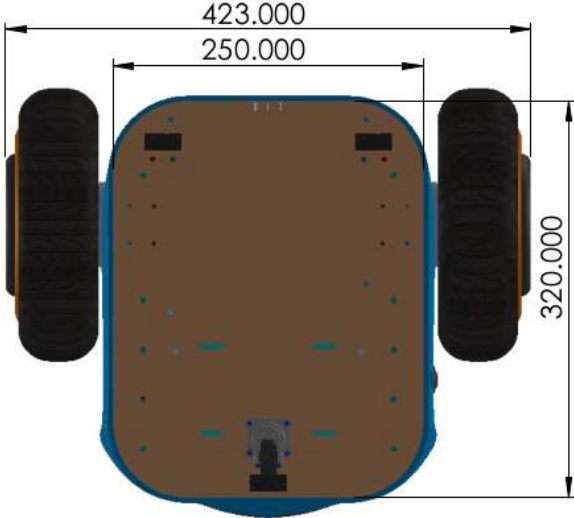
**Figure 3.8:** ATEMR: Mobile Base Design - 3D design v2 FRONT VIEW



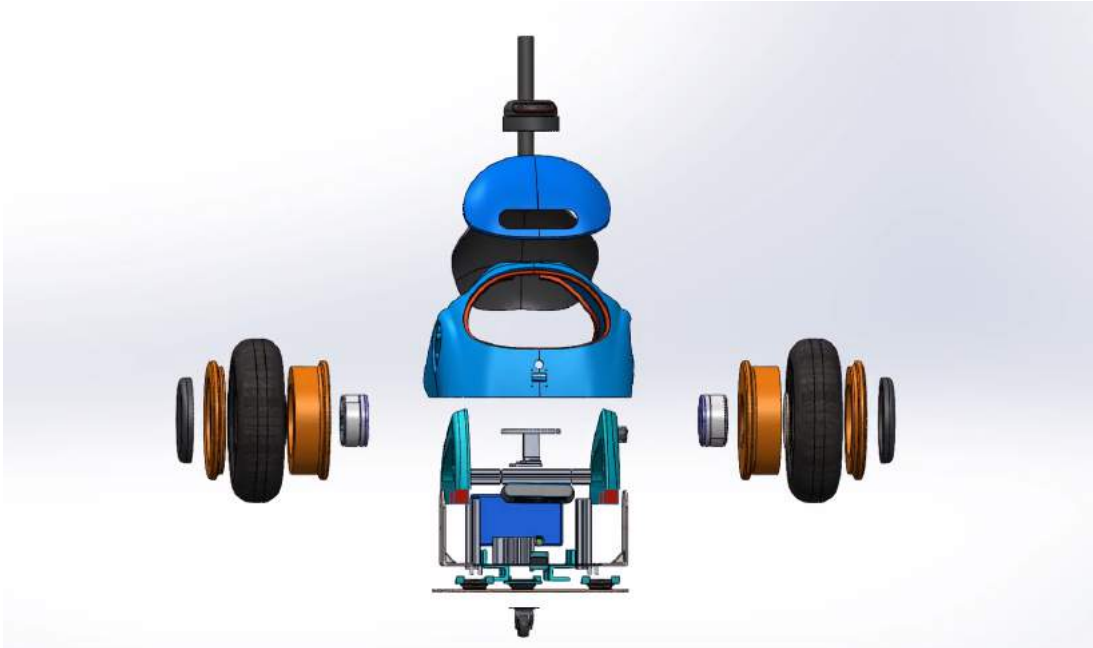
**Figure 3.9:** ATEMR: Mobile Base Design - 3D design v2 LEFT VIEW

Some design components of worthy mention are the design of the secondary core (The Secondary Core) casing Figure 3.12, the LIDAR and IMU support structure Figure 3.13 and the custom spring loaded locks that were designed for the ATEMR project Figure 3.14.

The mobile base design for this project as has been already stated has two large motorized wheels at the front with a caster wheel in the rear (Figure 3.8) and has an approximate 4.5cm ground clearance (refer to Figure 3.9) making it suitable for most indoor applications. The base is also coupled with three (3) infrared(IR) sharp sensors for staircase and extreme elevation change detection (see Figure 3.10): two (2) IR sensors in the front, placed on the

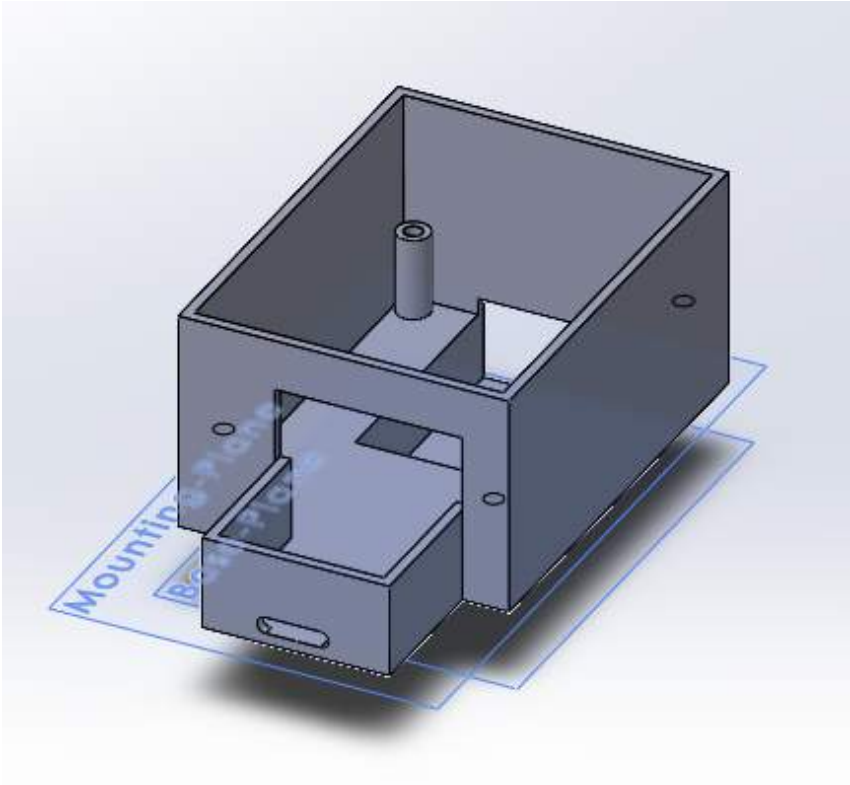


**Figure 3.10:** ATEMR: Mobile Base Design - 3D design v2 BOTTOM VIEW

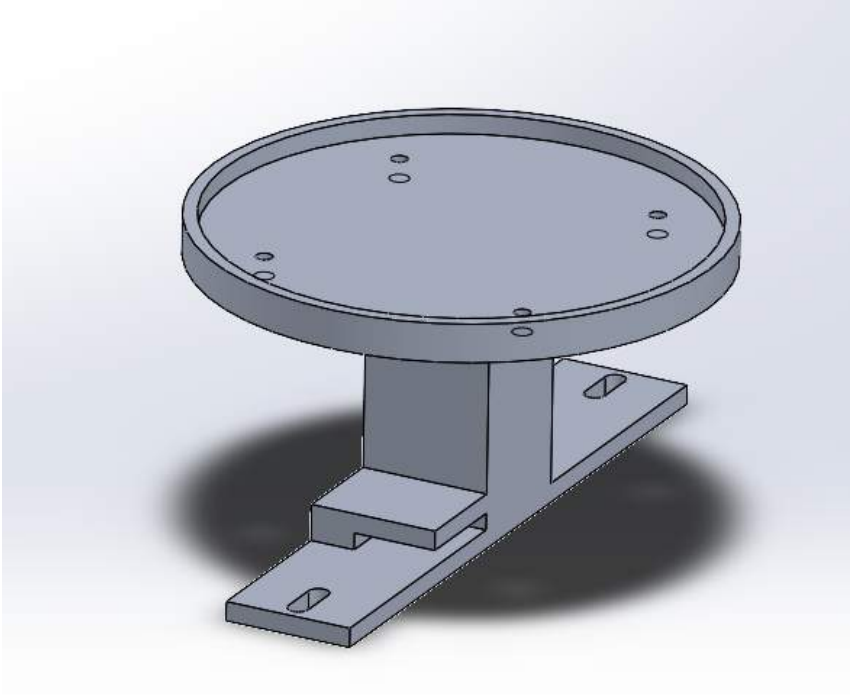


**Figure 3.11:** ATEMR: Mobile Base Design - 3D design v2 EXPLODED VIEW

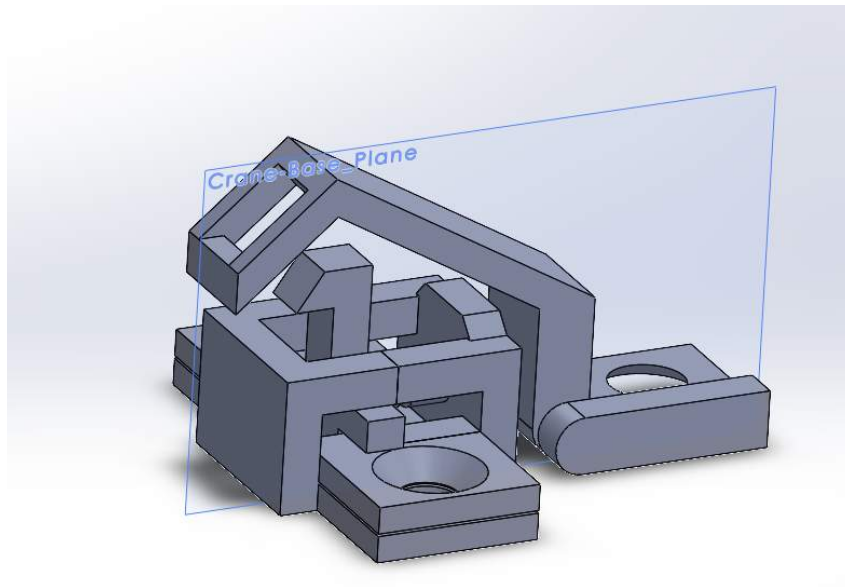
left and right, and one (1) IR sensor centrally positioned in the rear.



**Figure 3.12:** ATEMR: Secondary Core Casing Design - Raw sketch

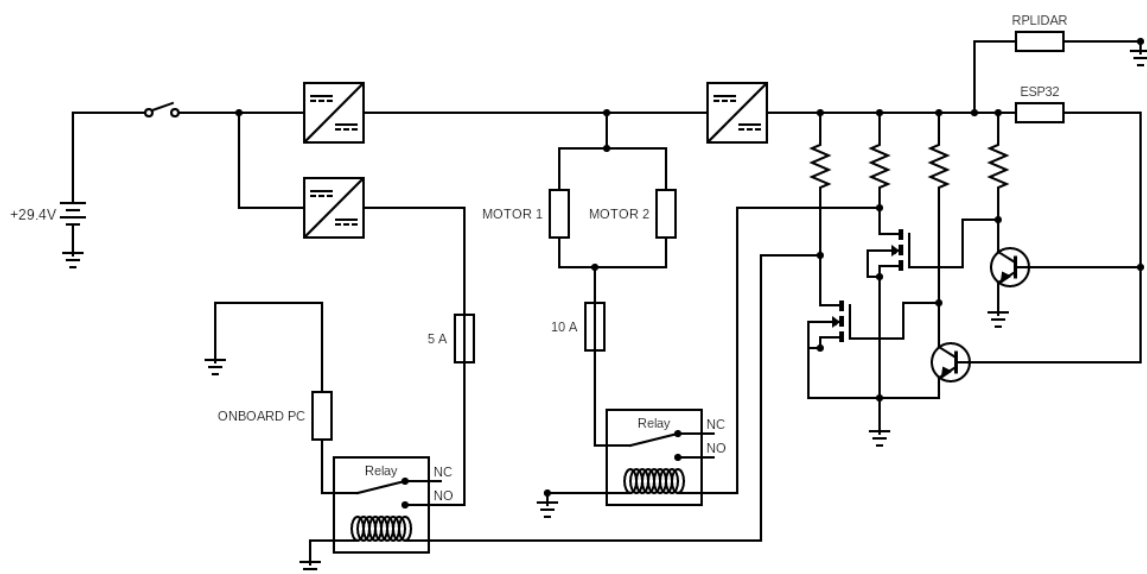


**Figure 3.13:** ATEMR: LIDAR and IMU Support Structure - 3D



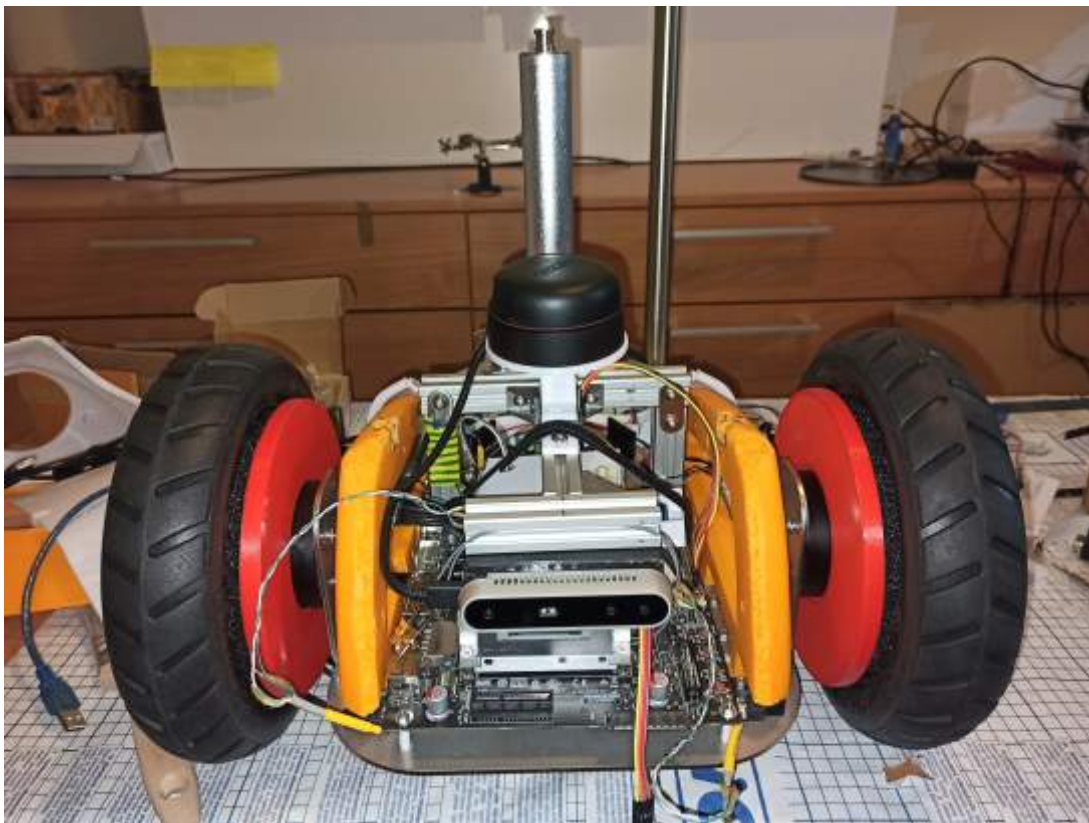
**Figure 3.14:** ATEMR: Spring-Loaded Lock Design - 3D

The approved and final version of the mobile base is printed out after making additional changes to fit the models in the volume allowed by the print bed to be 3D printed. After all required parts and electronics assembled, ATEMR is put together following the electrical diagram presented in Figure 3.15. There are three (3) DC DC Converters: one to supply voltage to the motors, another to supply voltage to the onboard pc and the third, to supply voltage to the secondary core and LIDAR unit. The DC DC Converters are used to step down the battery supply voltage from 29.4V to 24V, 19V and 5V respectively.

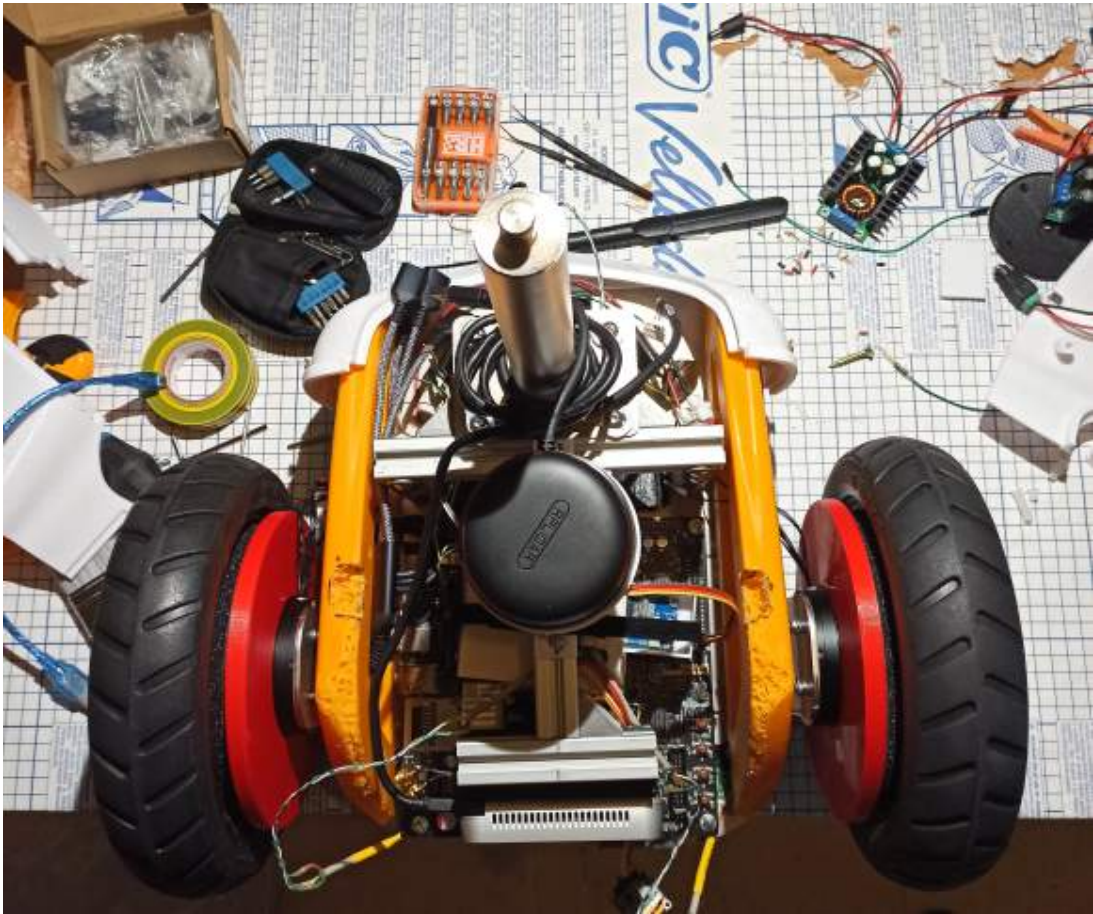


**Figure 3.15:** ATEMR: Overall electrical diagram

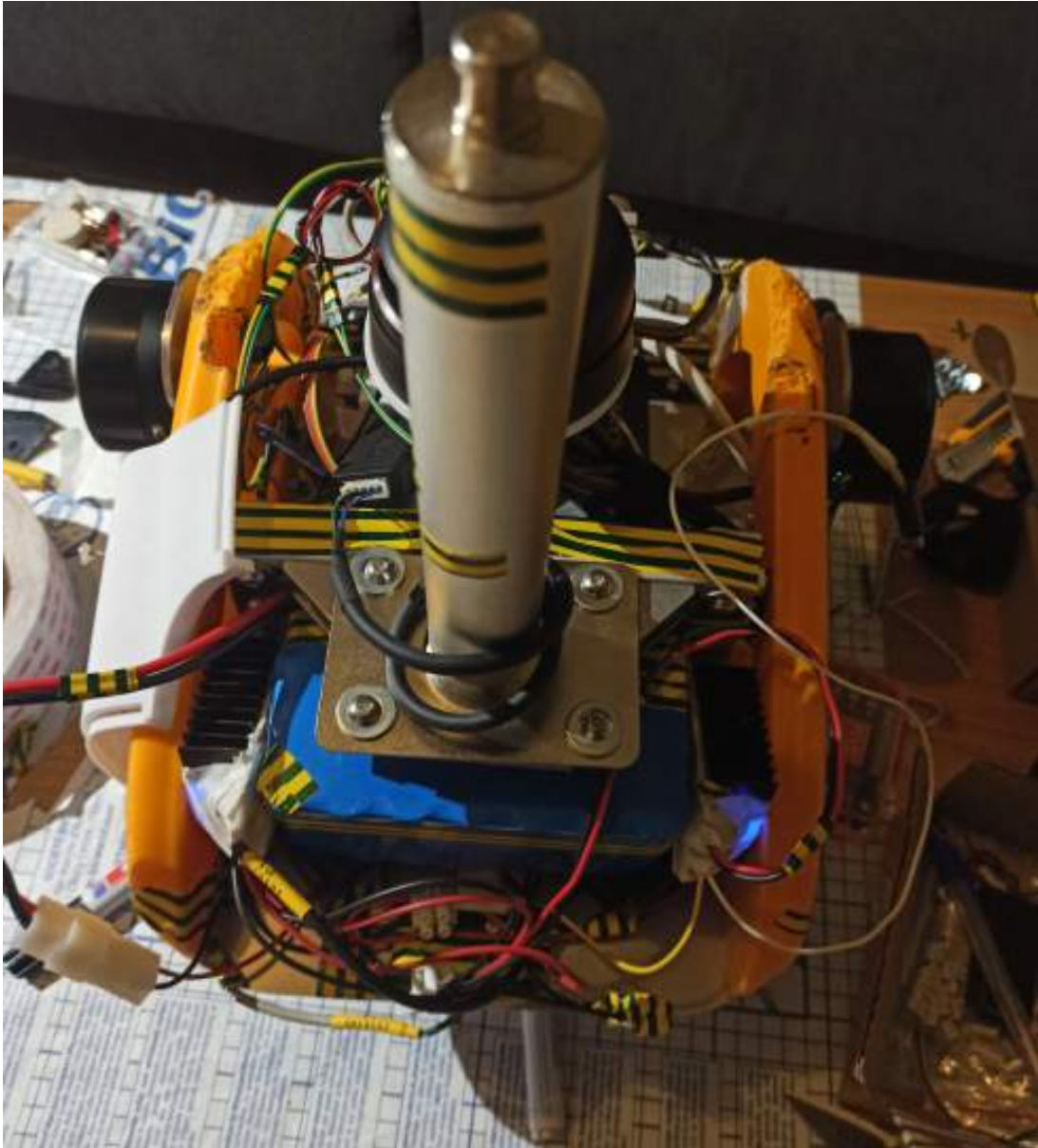
The build and wiring process took several attempts to get right. The project suffered some setbacks on the first two attempts due to exposed wires and human error in putting together the designed circuitry. Figures 3.16, 3.17, 3.18 and 3.19 shows the build process steps. First, the electronic system and wheels were assembled without the external chassis and tested for correct operation, Figures 3.16, 3.17, 3.18 and 3.19. Following the success achieved in the previous step, the wheels were taken apart to allow for the external chassis to be added Figure 3.19.



**Figure 3.16:** ATEMR: Mobile Base Build - FRONT VIEW



**Figure 3.17:** ATEMR: Mobile Base Build - TOP VIEW



**Figure 3.18:** ATEMR: Mobile Base Build - REAR VIEW





**Figure 3.19:** ATEMR: Mobile Base Build (without wheels)- PERSPECTIVE VIEW

The completed build shown in Figures 3.20 and 3.21 has the designed spring-loaded locks added and is painted for better aesthetics.

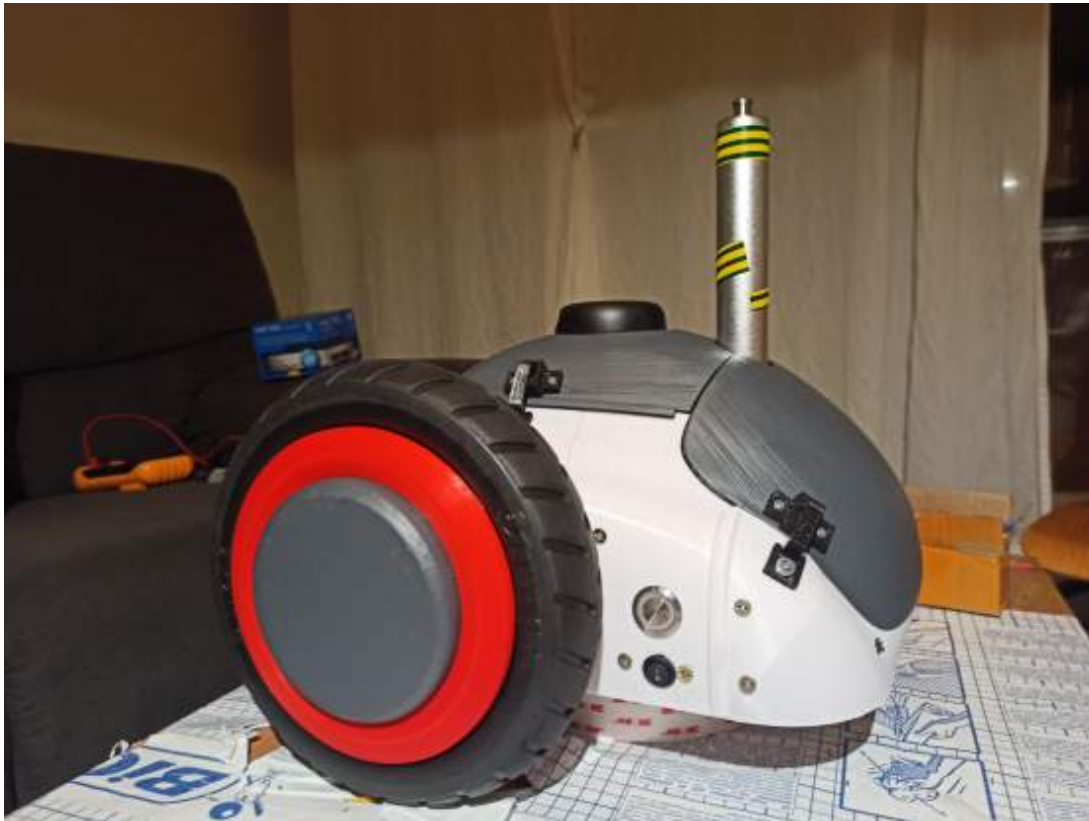


**Figure 3.20:** ATEMR: Mobile Base Completed Build - FRONT VIEW

The motorized wheels are driven via the **Controller Area Network (CAN)** bus. All CAN communications originate from two (2) main sources:

- (i) The **Type 1 or Safety and Emergency Velocity Controls**: These kind of controls mainly come from the *Secondary Core* when an undesired *Secondary core* state is entered and at robot startup before the *Primary Core* comes online.
- (ii) The **Type 2 or Operational Velocity Controls**: The *Type 2* Velocity controls are a super-set of the *Type 1* Velocity controls and also allows for the normal operation of the robot base. Normal operation implies both manual teleoperation of the robot base as well as autonomous operation/ navigation of the robot base.

It should be noted however that under certain circumstances for example when the heart-beat sync (2 seconds) fails, both primary and secondary core will publish emergency commands to immobilize the robot base.



**Figure 3.21:** ATEMR: Mobile Base Completed Build - LEFT VIEW

### 3.2.2 Robot Base Specifications

The robot base is designed with the following specifications:

#### *Speed and Performance*

- **Base plate dimensions:** 320 x 250 (L X B, in mm)
- **Drive Type:** Two-wheel differential drive
- **Speed:** Up to 1.8 m/s
- **Ground Clearance:** 48 mm
- **Robot Weight:** Approximately 20 kg
- **Payload:** Approximately 5kg (Including top mount. Payload is reference only while robot moving on flat surface)
- **Working Environment:** Indoor
- **Operating Ambient Temperature:** -10 to +40 degrees Centigrade

#### *System*

- **Operating System:** Ubuntu 20.4 running ROS Noetic Ninjemys
- **Battery:** 29.4V DC power lithium Ion Battery with 8Ah capacity

- **On-board Computer:** Gigabyte Brix GB-BSi5-6200 Core i5 16 GB Ram, M.2 SSD 120 GB
- **Control System:** ROS
- **Charge Time:** 3 Hours

#### *Communication*

- **WiFi:** YES
- **Ethernet:** YES

#### *Sensors*

- **2D-Lidar:** RPLidar A2-A2M8
- **3D-RGBD:** Realsense D415
- **Infrared Distance Sensor:** GP2Y0A41SK0F

#### *Indicators*

- **OLED:** SSD1306 128×64 Oled i2c display Module
- **LED Indicator:** CPU Power
- **Sound Indicator:** Power On, Power Off, Display Screen Change

The prototype should be able to map, localize and navigate in an indoor environment. In order to be able to control the mobile base remotely, send navigation goals, make maps, capture images and record videos with the front camera, the robot will mount a local python based web server that can be accessed by valid users/ operators.

## **Components of the Mobile Base**

The various physical and electronic components found in the prototype are listed in Table 3.3.

Since the robot prototype is ROS based, the main packages used to accomplish the project's objectives are as follows:

- **amcl:** amcl is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track the pose of a robot against a known map, Gerkey (2020).
- **rplidar\_ros:** This package provides basic device handling for 2D Laser Scanner RPLIDAR A1/A2 and A3, Slamtec ROS (2020).

**Table 3.3:** ATEMR: Electronic components

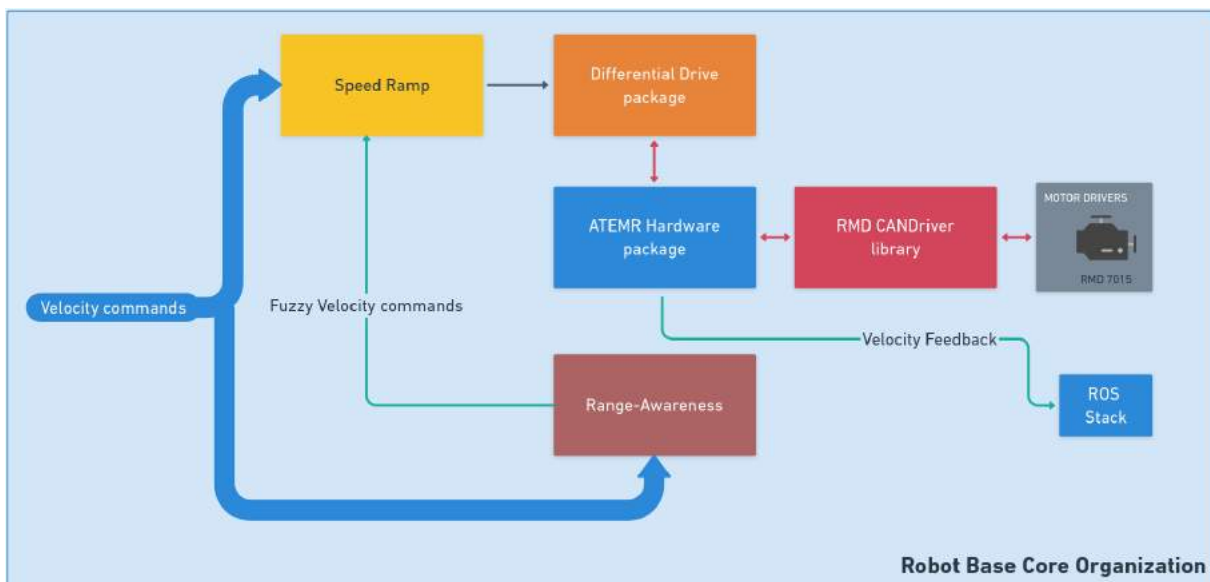
Item	Part	Quantity
1	ESP32 Micro-controller	1
2	Gigabyte Brix GB-BSi5-6200 Core i5 Mini PC	1
3	Infrared Distance Sensors	3
4	RPLidar A2-A2M8	1
5	Realsense D415	1
6	DC-DC Voltage converter	3
7	RMD-L-7015-45T motor with 14-bit magnetic encoder	2
8	WT901 Inertial measurement unit	1
9	29.4VDC 8A Rechargeable Battery	1
10	WT901 Inertial measurement unit	1

- **ROS Navigation Stack:** The Navigation Stack is fairly simple on a conceptual level. It takes in information from odometry and sensor streams and outputs velocity commands to send to a mobile base, Marder-Eppstein (2020b).
- **ROS Control: diff\_drive:** Controller for differential drive wheel systems. Control is in the form of a velocity command, that is split then sent on the two wheels of a differential drive wheel base. Odometry is computed from the feedback from the hardware, and published, Magyar (2021).
- **move\_base:** The move\_base package provides an implementation of an action that, given a goal in the world, will attempt to reach it with a mobile base. The move\_base node links together a global and local planner to accomplish its global navigation task, Marder-Eppstein (2020a).
- **roslibjs:** Is the core JavaScript library for interacting with ROS from the browser, Open Robotics (2019b).
- **rosbridge\_server:** Rosbridge server is part of the rosbridge\_suite of packages, providing a WebSocket transport layer, Brandon (2013).
- **ros2djs:** Is the standard JavaScript 2D visualization manager for ROS, Open Robotics (2019a).
- **twist\_mux:** When there are more than one source of twist input, it is important to multiplex all those input sources into a single one that goes to the controller. The twist multiplexer node uses a priority selection mechanism determine which input get published. The mobile base has two main input sources, velocity commands from operator input (keyboard, joystick, reactive control) and velocity commands from the

move\_base node (navigation stack) and needs to be operated exclusively. Hence, when in manual mode, any input from the navigation stack will be ignored and operator inputs prioritized. And vice-versa when in auto mode.

### 3.3 Robot Base Core Software

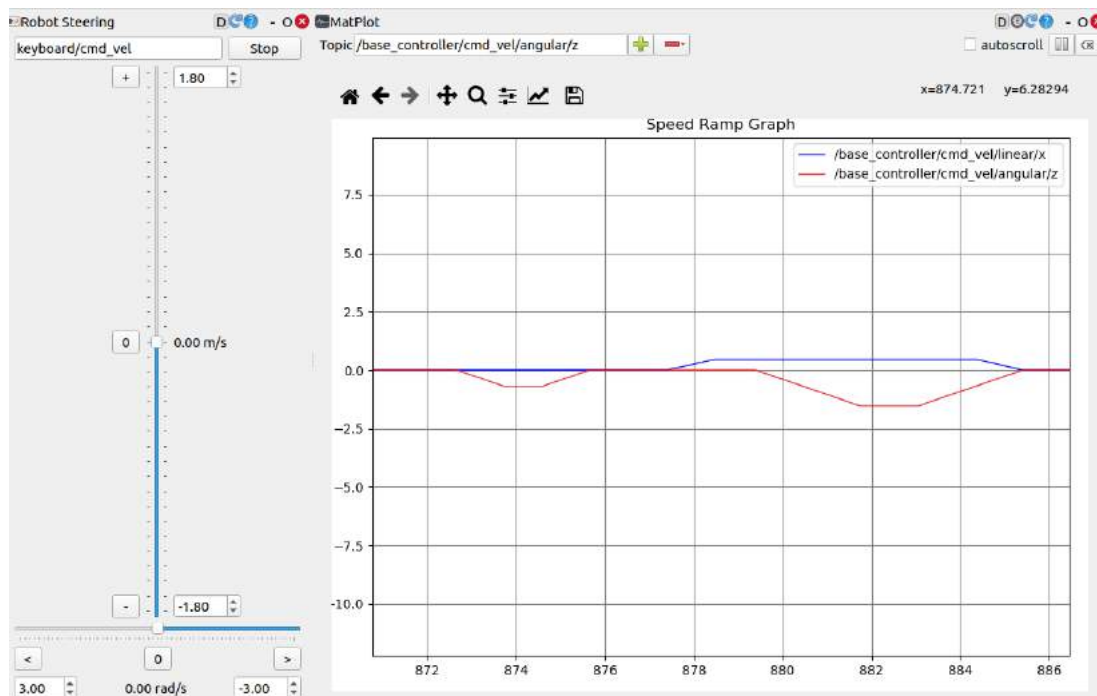
The following defines the core base software and its structure. The robot base uses two (2) RMD-L-7015 45T(CAN) brushless motors controlled via CAN @1000Kbs baudrate. Three (3) main packages come together to make control of these motors and hence the mobile base possible. Looking at Figure 3.22, they consist of the 'Differential Drive ROS package', the 'ATEMR Hardware package' and the 'RMD CAN Driver library'.



**Figure 3.22:** ATEMR: Core Software Organization

Referring to the core software organization in Figure 3.22, incoming control velocity is first sent to the *Speed Ramp* node where it gets pre-processed and or smoothed (ramped up or down, 3.23) and then, passed on, to the *Differential Drive* package. Inside the *Differential Drive* package, the received ROS twist message is converted to raw left and right motor velocities that can be used at the lower level to control the motors.

The *ATEMR Hardware* package takes the output from the *Differential Drive* package and using the *RMD CAN Driver* library, transmits motor commands to the motorized wheels. The process also goes the opposite direction when motor feedback is read and made available to the entire ROS stack.



**Figure 3.23:** ATEMR: Speed Ramp visual

Velocity Input from the *Range-Awareness/ Reactive* node is selected for the robot base control if the ‘awareness’ flag is set by the operator. By default, the ‘awareness’ flag is set to FALSE.

## 3.4 ATEMR Layout and Sub-Systems

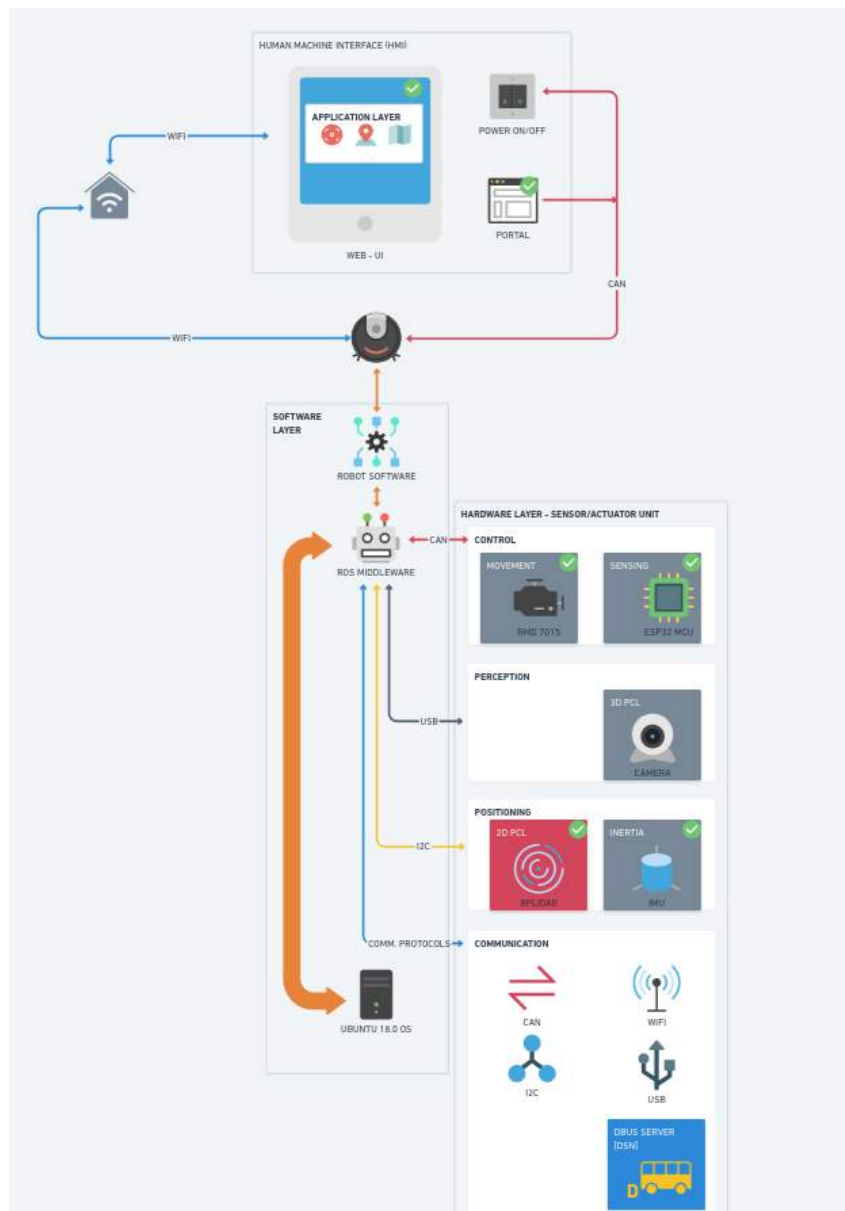
Figure 3.24 gives the general layout overview of the internal system of ATEMR.

The project is structured as a three (3) layer architecture shown in Figure 3.25 consisting of low-level, mid-level and high-level layers.

The operator interacts with *high-level* layer which is mainly the web interface. Operations and or computations such as navigation, obstacle avoidance, mapping and localization happen at the *mid-level* layer. The *low-level* layer is in charge of robot awareness (fuzzy inference engine) capabilities and manual/automatic velocity control (speed ramp). The low-level is also responsible for slowdown and emergency activations and de-activations.

### 3.4.1 Low-Level: SpeedRamp

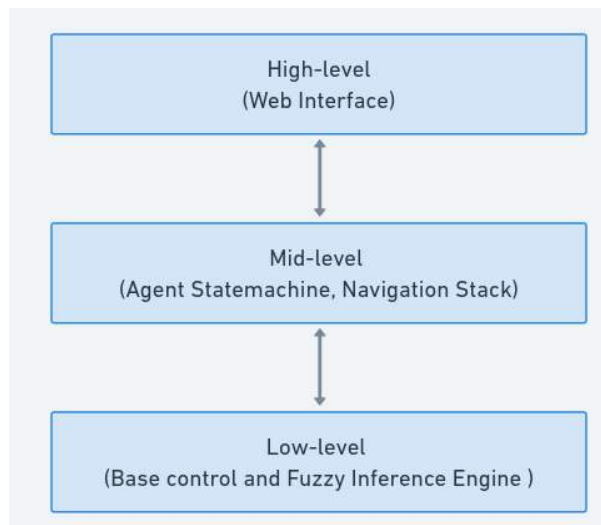
The SpeedRamp node’s function is to ramp up or down the velocity commands that are being sent to the base controller in order to avoid jerky motions when operating the mobile



**Figure 3.24:** ATEMR: Project layout

base. It uses linear time dependent ramp to increase or decrease the output velocity command gradually over a period of time until the desired velocity is achieved. It also subscribes to the HEX-PATTERN (see Low-Level: The HEXAGONAL PATTERN (HEX-PATTERN) Generator) and performs obstacle detection and mobile base state transitions based on this data. For obstacle detection, two main functionalities are implemented. If the obstacle detected is further out than the “obstacle stop distance (28cm)” but within the “obstacle slow distance (56cm)”, the mobile base velocity is limited to 0.35m/s in linear x-axis and 1.6 rad/s in the angular z-axis. The second implementation activates the mobile base obstacle emergency





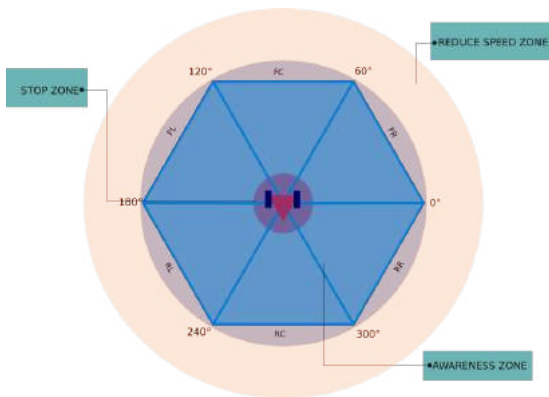
**Figure 3.25:** ATEMR: Project architecture

mode if there is an obstacle within the “obstacle stop distance” and resumes normal operation mode two (2) seconds after the obstacle has moved out of range. Furthermore, the speedramp node has a state detector for detecting when the mobile base is idle after a certain period of time, eight (8) seconds. If no state transition command is received after 8 seconds, the sets the mobile base to idle state. This state is further used by the reactive Fuzzy Inference Engine (FIE) and hardware nodes to adjust their operational limits and rate.

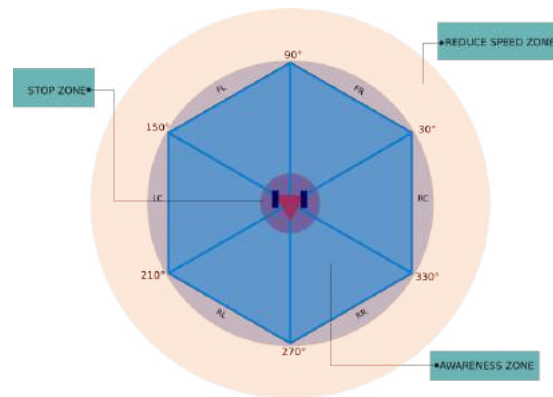
In the absence of this node, any velocity command sent to control the mobile base will not be registered and hence the base cannot be operated.

### 3.4.2 Low-Level: The HEXAGONAL PATTERN (HEX-PATTERN) Generator

Since the robot software runs on a mini PC having limited processing resources, it is necessary to simplify some of processing done on sensor input in order to approximate some form of instantaneous behaviour and real-time response. The HEX-PATTERN generator residing in the ‘scan processor node’ accomplishes this purpose by first splitting the raw laser scan input into various sections based on the angle of detection and averaging. Other parts of the robot stack that require obstacle input for their proper operation take as input, the resulting pattern and make behavioural decisions based on it. This has the benefit of taking up less resources during processing and faster decision making since the final resolution is extremely reduced. The laser scanner used has a resolution of  $0.9^\circ$  approximating about 360 laser samples which when passed through the pattern generator is reduced to only six (6) averaged laser samples for further processing.



**Figure 3.26:** ATEMR: HEX-PATTERN



**Figure 3.27:** ATEMR: HEX-PATTERN-90 (90°)

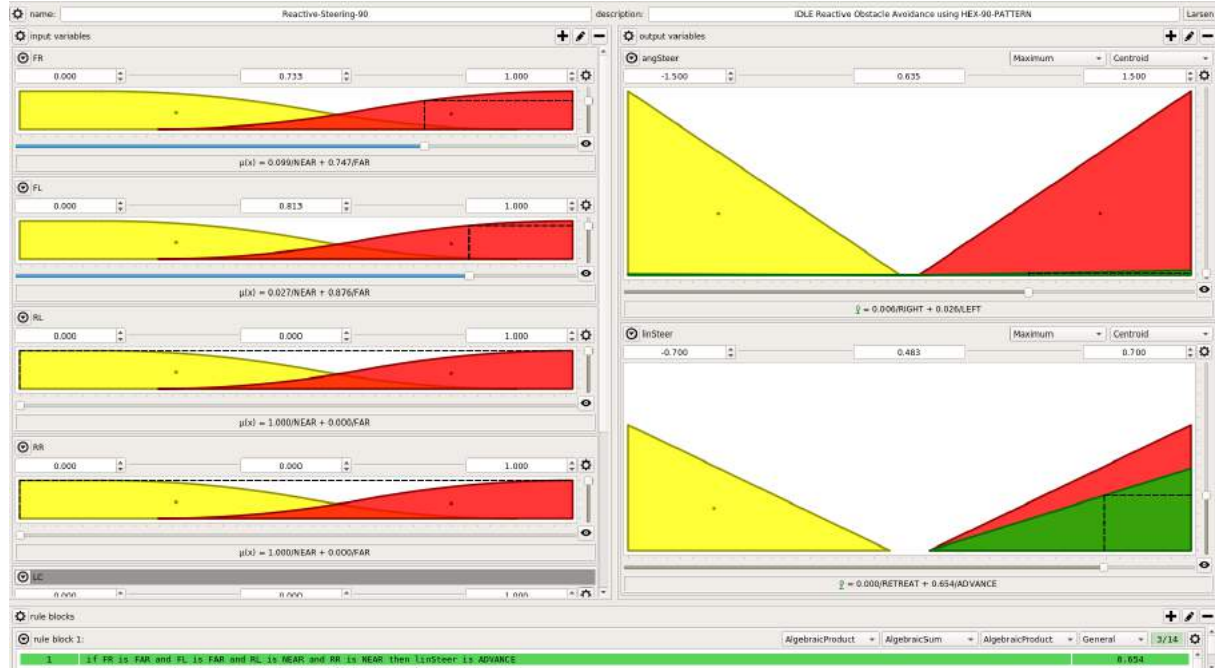
When input from the laser scan arrives, it is first transformed to the base frame and then converted into a point cloud for further processing. The next stage involves slicing and filtering the point cloud by radius into Front-Right(FR), Front-Left(FL), Left-Center(LC), Rear-Left(RL), Rear-Right(RR) and Right-Center(RC) sections as demonstrated in Figures 3.26 and 3.27. The two (2) main types of patterns resulting from this stage are the HEX-PATTERN (Figure 3.26) and the HEX-PATTERN-90 (Figure 3.27). The later is a 90 degree rotated version of the former and is the pattern type around which the Fuzzy Inference Engine (subsection Low-Level: The Reactive Base Fuzzy-Inference Engine) is built.

### 3.4.3 Low-Level: The Reactive Base Fuzzy-Inference Engine

The basis for the 'reactive-base-node' is to introduce an obstacle-aware collision avoidance and self preservation during the operation of the robot. The reactive base node uses two (2) fuzzy logic inference engines operating sequentially (both when the mobile base is being operated manually, and when idle) to achieve this. In Bayar et al., 2014, three (3) fuzzy inference systems were used to realize goal reaching, obstacle avoidance and a controller for combined behavior selection. In this study however, the two (2) inference engines will be used mainly for obstacle/ collision avoidance and uses the fuzzylite library implemented in Rada-Vilela, 2018 at the core of the reactive base node. The library implements many controllers including: Mamdani, Takagi-Sugeno, Larsen etc. among others, has five (5) linguistic hedges and exports to C++, Java and other languages. Hence, the reason it was chosen for this project. It is also open source.

The IDLE inference system (3.28) uses the Larsen controller and has six (6) inputs: Front-Right(FR), Front-Left(FL), Left-Center(LC), Rear-Left(RL), Rear-Right(RR), Right-Center(RC).

These are the result of the HEX-PATTERN discussed in Low-Level: The HEXAGONAL PATTERN (HEX-PATTERN) Generator subsection, each described by the linguistic term NEAR and FAR. And two (2) outputs: linSteer (linear steering) and angSteer (angular steering) for controlling the mobile base. This makes for 66 combinations which have been further reduced to 14 rules. The IDLE inference system has total control of the mobile base when it runs.



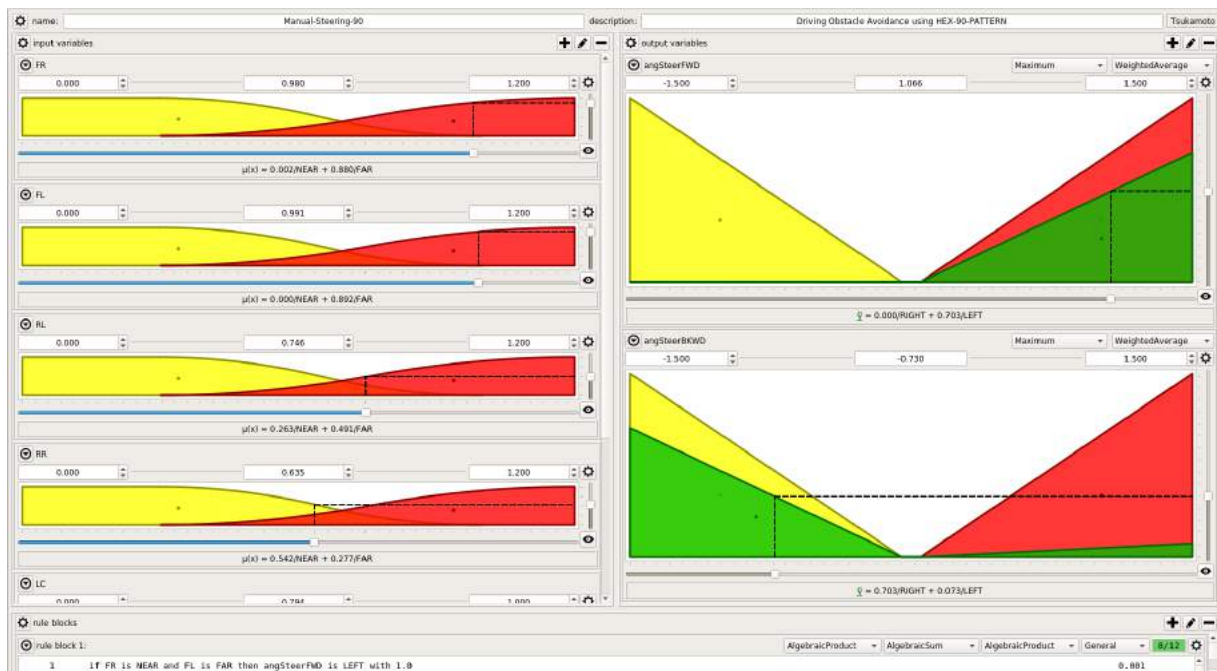
**Figure 3.28:** ATEMR: IDLE Inference Engine setup

The MANUAL inference system is setup similar to the IDLE inference system having six inputs (FL, FR, LC, RL, RR and RC) and two (2) outputs (forward angular steering and reverse angular steering) and uses the Tsukamoto controller. It has 13 rules out of the 66 possible combinations.

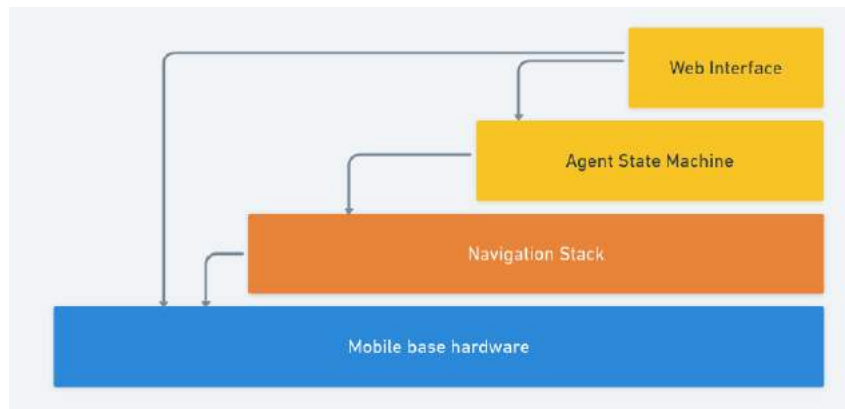
$$W_{applied} = (W_{ME} * \tau) + (W_{IO} * (1 - \tau)) \quad (3.1)$$

Since this system works in tandem with the operator inputs, the engine's output will have to be combined with the operator's velocity commands using the equation 3.1. The equation is inspired by Bayar et al., 2014 as cited in Abdallah and Dan, 2013.

From a different perspective however, it can be said that, ATEMR has four (4) main divisions as shown in Figure 3.30.



**Figure 3.29:** ATEMR: MANUAL Inference Engine setup



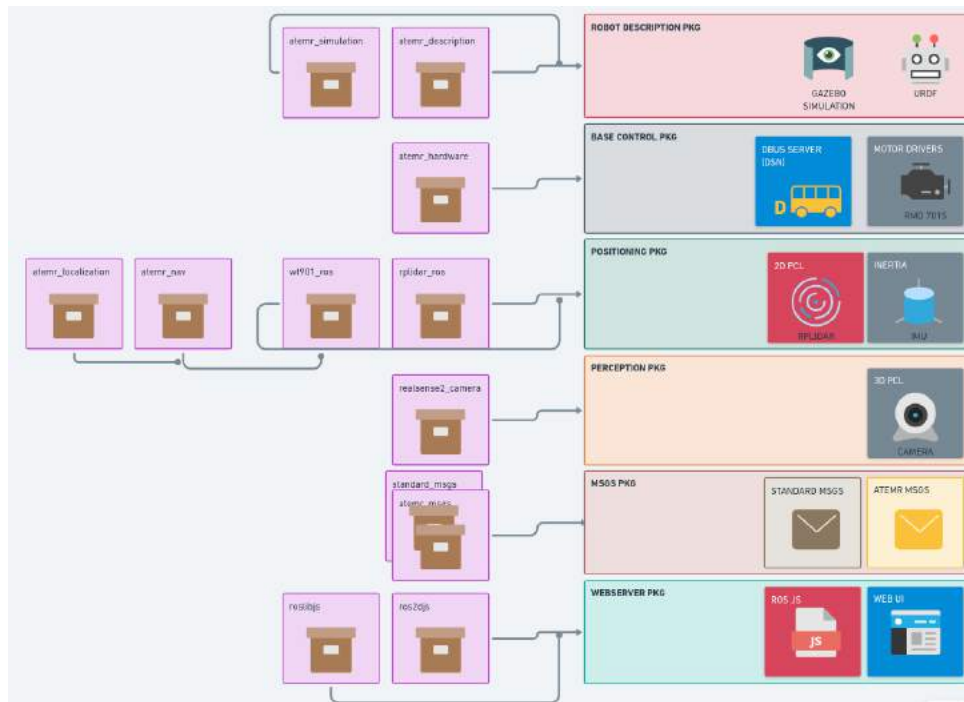
**Figure 3.30:** ATEMR: Project sub-divisions

The project can be further divided into two (2) main parts based on the functionality. These are the primary and secondary cores each of which may be composed of a single or multiple sub-systems. The cores and their sub-systems are discussed next.

## 3.5 The Primary Core

The Primary Core consists of all sub-systems in the on-board pc. These include the ATEMR ROS packages, agent statemachine and operator web interface.

### 3.5.1 ATEMR ROS Packages



**Figure 3.31:** ATEMR: Package structure and organization

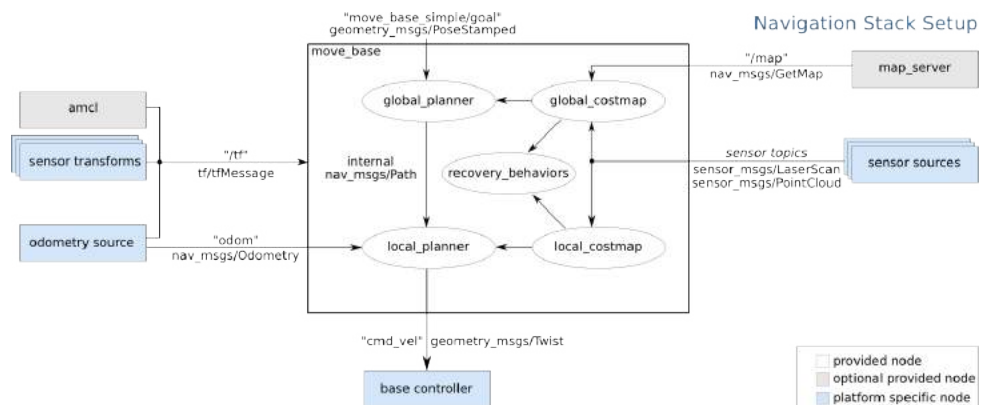
The ROS packages descriptions are as follows:

- **atemr\_description:** This package contains the main universal robot description file (urdf) consisting of the visuals, collision matrices, inertias, joints and their inter-connection, as well as 3d meshes that make up for the internal description of the robot.
- **atemr\_simulation:** The robot model is also simulated in Gazebo (ROS compatible robot simulation platform, Figure 3.32) to speed up algorithm development and testing. This ROS package contains one simulation world consisting of a single studio apartment with a sofa, dining table, a small balcony and a bed. The simulation starts up with the robot at the center of the living room. There are two main light sources in the scene: a sun light and a point light. The camera is static and positioned in a birds-eye view looking down on the simulated world. The infrared sensors are visualized in the simulated world. The walls of the apartment have been made transparent to aid visualization and keep the robot in view without having to displace the camera. This package also has a config file “sim\_control.yaml” that contains the PID gains for the simulated robot.
- **atemr\_nav & atemr\_localization:** Autonomous navigation is achieved using the ROS navigation stack (Figure 3.33) and is setup in these packages. The



**Figure 3.32:** ATEMR: Gazebo Simulated World

move\_base node provides a ROS interface for configuring, running, and interacting with the navigation stack on a robot. A high-level view of the move\_base node and its interaction with other components is shown above. The blue vary based on the robot platform, the gray are optional but are provided for all systems, and the white nodes are required but also provided for all systems. The navigation stack which consists of multiple packages spanning the localization, control and navigation packages was setup in parallel.



**Figure 3.33:** ROS Navigation stack setup (from Marder-Eppstein (2020a))

Moving from left to right (see Figure 3.33), the ATEMR mobile base's operating odometry is fusion of three(3) odometry sources using the extended kalman filter (EKF) (Moore and Stouch, 2014) provided by the robot localization ROS package. The first input source (base controller odometry) is the raw motor velocities returned by the base hardware as feedback to the differential drive controller. From this input, the

algorithm uses the x and y coordinates, yaw, and z angular velocity. The second odometry input is the robot pose output published by the hector mapping node which provides a fast online learning of occupancy grid maps requiring low computational resources Kohlbrecher et al., 2011, in the robot's odometry frame. The third input source is the pose outputted by the scan matcher node, Dryanovski et al., 2019. The the point line iterative closest point (PLICP), a variant of the iterative closest point (ICP) used by the scan matcher is described in Censi, 2008.

The Advanced Monte-Carlo Localization (AMCL), Gerkey, 2020 node that provides the mobile base's pose with respect the world (map) is fused with the odometry output from the local EKF node in a global EKF node to provide localization for the robot.

The move base package links together a global (GlobalPlanner) and local (TrajectoryPlannerROS) planner to accomplish its global navigation task Marder-Eppstein, 2020a. It also has a plethora of recovery behaviors that can be used to unstuck the robot should the need arise. The global and local planners are setup with four (4) main 2D/ 3D occupancy grid layers, Marder-Eppstein et al., 2018.

- (i) The Static layer: The static layer mainly refers to the current map loaded in the map server.
- (ii) The Laser Obstacle layer: This layer's input is the live laser scan data stream and is based in the x-y dimension.
- (iii) The Voxel Obstacle layer: This layer takes input from the point cloud data generated by the 3D Camera on-board and allows the planner to take 3D obstacles into account when plotting a trajectory for the mobile base.
- (iv) The Inflation layer: The inflation layer is mainly a safety layer and adds a cost area around obstacles and walls that the mobile base should try to stay away from.

The global costmap constituting the static layer (map) is used by the global planner for plotting robot trajectories. The local costmap used by the local planner however, is made up of the laser obstacle layer, voxel obstacle layer and inflation layer for navigation and obstacle avoidance.

The map server node Gerkey and Pratkanis, 2020, on the extreme right in diagram 3.33 provides the map used by the AMCL node for localization and move base node, for navigation.

The steps followed in setting up the navigation stack is as follows:

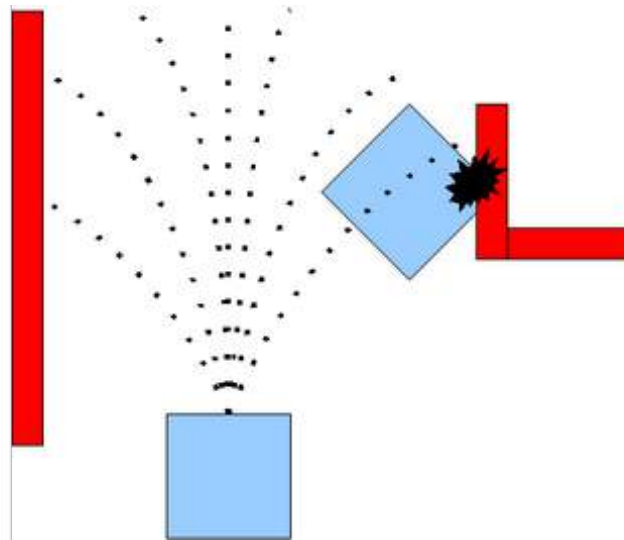
1. Two (2) main nodes are configured to provide the robot odometry in the fixed space/ world "odometry/filtered\_world" and local/odom space "odometry/filtered\_odom". For this project, the inputs to the "odometry/filtered\_odom" EKF

node consists of continuous data sources; the madgewick filtered IMU data, Hector mapping node and robot base odometry. Also, the “world frame” is set to the same frame as the odometry frame. A voxel layer that uses the camera’s point cloud as input is configured for the local costmap enabling the robot to perceive obstacles in 3D.

2. Once the robot sensor data is fused together, the next step is to configure the “slam\_gmapping” package for ATEMR. This is setup in the `atemr_localization` package. This configuration process consists of topic remapping, parameter setup and finally the creation of a launch file (`slam_gmapping.launch`) to start the `slam_gmapping` node. The input the `gmapping` node are fixed laser scan and robot odometry data and the output, a 2D occupancy grid representation of the environment and robot pose estimate in the created map.
3. As a parallel process to the previous step, the `map_server` package is also configured and a launch file created (`map_server.launch`) for serving and saving maps during mapping and navigation. The inputs to the “odometry/filtered\_world” consists of the output from the “odometry/filtered\_odom” ekf node and the output from the AMCL node.
4. Since ATEMR is an indoor robot, the AMCL package will be used as the preferred localization method. For that all parameters necessary for smooth operation is setup in the AMCL config file, as well the creation of a launch file (`atemr_amcl.launch`) to run the node. AMCL takes in a laser-based map, laser scans, and transform messages, and outputs pose estimates. On startup, AMCL initializes its particle filter according to the parameters provided. Note that, because of the defaults, if no parameters are set, the initial filter state will be a moderately sized particle cloud centered about (0,0,0).
5. The last step is to configure the local and global planners needed by `move_base` to move the robot towards a set goal while avoiding obstacles in the process. For the global planner, the standard ROS GlobalPlanner is chosen in comparison to the Navfn global planner. The GlobalPlanner used both the djikstra path finding and the A-Star path finding algorithms based on the configuration. Once the global plan is created, a local planner is required to move the robot base and in so doing, also avoid obstacles in its path. The both DWAPlaner and TrajectoryPlanner were compared. This work will use the TrajectoryPlanner for local navigation. The `base_local_planner` package provides a controller that drives a mobile base in the plane. This controller serves to connect the path planner to the robot. Using a map, the planner creates a kinematic trajectory for the robot



to get from a start to a goal location. Along the way, the planner creates, at least locally around the robot, a value function, represented as a grid map. This value function encodes the costs of traversing through the grid cells. The controller's job is to use this value function to determine  $dx$ ,  $dy$ ,  $d\theta$  velocities to send to the robot as depicted in Figure 3.34.



**Figure 3.34:** Dynamic Window Approach (from Marder-Eppstein and Perko (2019))

- **atemr\_hardware** package is the most important package in the entire ATEMROS Stack. The hardware package contains the custom motor driver scripts for sending velocity commands to the motors and for reading motor feedback. This package implements the ROS Velocity Interface and works in tandem with the differential drive package to ensure motor commands reach the motors and motor feedback, reaches the differential drive controller.
- **atemr\_control** package is the next most important package after the **atemr\_hardware** package and as its name suggests, is responsible for control of the mobile base. The speed ramp node without which velocity inputs do not reach the base controller lives in this package. The mobile base odometry precision as compared to its real world ground state is mainly dependent on the sensor input and configuration.
- **atemr\_dbus** package is a wrapper for the Desktop Bus (DBus) linux programming interface and enables high-level operators to send system level messages such as to connect to a different network, get the SSID of the current network, shutdown and restart, to the core operating system.

### 3.5.2 Agent Statemachine

The mobile base statemachine agent consists of seven (7) principal states in total. Six (6) of these states run sequentially while the seventh state runs parallel to the other six states. The principal states are discussed as follows:

1. **StartUP State:** This is the first state that runs when the agent starts. It is in charge of starting up all other robot nodes and checking for their correct operation before continuing. After the StartUP State exits, if no critical error occurred, it transitions to the idle state. Else, transitions to the shutdown state.
2. **Idle State:** Just as the name suggests, this state is where the agent runs when it has no target goal and when running in manual mode (default on start up). When a goal is received, it gets processed here and if valid, transitions to the exec state. Else, transitions to the error state.
3. **Exec State:** The task is simple, this state monitors the goal execution and reports feedback to the operator and also listens to operator inputs such as goal cancelling and system shutdowns. If the goal is reached successfully, it returns to the idle state to await the next goal. Else, transitions to the error state.
4. **Error State:** This is where all statemachine errors get processed. If the error is not critical, the agent returns to the idle state afterwards. Else, a transition is made to the shutdown state.
5. **Shutdown State:** The shutdown state has two purposes: one to restart the entire statemachine stack and two, to shutdown the agent.
6. **Monitor State:** This is the seventh state that runs in parallel to all other states. It serves as the main communication hub between all nodes outside the agent and all states inside the agent statemachine. It receives service calls and reports statemachine state to the web interface. It also accepts input from the hardware node, pre-processes it and forwards it to the web-interface and vice-versa.

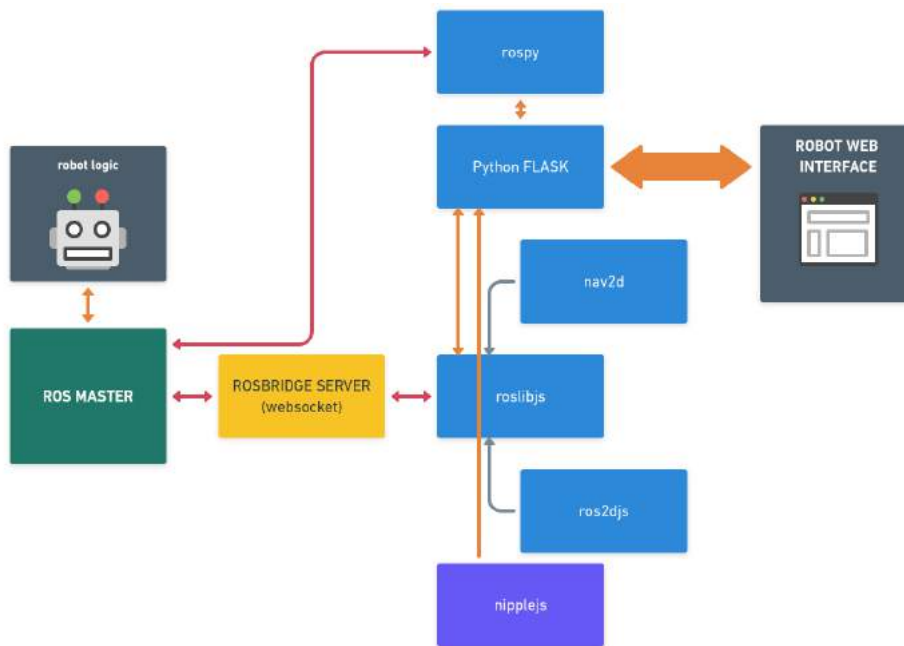
### 3.5.3 Web User Interface (UI)

This is the part of the project that interfaces **operators** with the robot and or completes the circuit that is the **tele-presence** involved in the use of the robot. The dynamic web page serving as an operational interface is build upon the **Python FLASK** module and uses javascript libraries in the background to communicate with the Robot Operating System(**ROS**) master node and on-board PC.

The following is a discussion of relevant libraries and packages that were leveraged to put together the web interface for the robot and their interconnections as shown in Figure 3.35.

- (i) **rosbridge\_server**: According to Brandon (2013), creates a WebSocket connection and passes any JSON messages from the WebSocket to `rosbridge_library`, so `rosbridge` library can convert the JSON strings into ROS calls. The reverse also happens, with `rosbridge` library converting any ROS responses into JSON, then passing it to `rosbridge` server to send over the WebSocket connection.

It serves as the main backbone linking the web page and the ROS master node (hence, to all other nodes running on the on-board PC). The `rosbridge` server runs on port '9090' and listens on '0.0.0.0' (all incoming ip addresses).



**Figure 3.35:** The Web-UI Inter-Connection diagram

(ii) **rospy:**

The rospy client API enables Python programmers to quickly interface with ROS Topics, Services, and Parameters, Conley and Thomas (2017)

The robot web page leverages this Application Programming Interface (API) to interact with nodes running on the system. Particularly, it used to check the ROS master state, get absolute url to package and file locations and also to read and edit parameters in the ROS parameter server.

- (iii) **roslibjs:** This library is actually an implementation of the rospy API in javascript and is the core library for interacting with ROS from the browser (Open Robotics (2019b)). For this implementation, this library plays a major role in report robot velocity feedback to the browser, commanding the robot base, initial robot positioning and goal sending among many others.
- (iv) **ros2djs:** In order to provide accurate visualization of the robot and its surrounding, the ros2djs library is used to draw the current map being used or created by the robot. It is built on top of roslibjs (Open Robotics (2019a)).
- (v) **nav2d:** This is a functional library built on top of the ros2djs library and provides an easy to use interface to send goals to the robot, monitor goal status and goal feedback.
- (vi) **nipplejs:** This library is used to draw a static joystick with which the robot operator can freely control the movement of the robot base. It is coupled with and depends on the roslibjs core library for sending velocity commands to the robot base.

In the upcoming sub-sections; General Overview will describe the overall operational behaviour of the web page and how everything is interconnected. Going further, each of the main pages Authentication, Control, Mapping, Setting will be discussed.

## General Overview

The best depiction of the interconnection existing in the web page is shown in Figure 3.35. On startup (it is assumed that the ROS master node and other nodes are running), the ros-bridge server is launched and listens for client connection from all ip addresses on port '9090'. The Python *FLASK* is configured to use either 'eventlet' or 'gevent' to serve pages based on which of the two is readily available. The FLASK server listens for incoming connections on the default port '5000'. It should be noted also that the FLASK based web page uses its own internal WebSocket used to communicate with the Python processes running on the on-board PC and it is always connected and running as long as the Python server is alive. This socket is mainly used to update the current memory, CPU and disk usage on the

web page at 1Hz. It also used to check if the ROS master core node is alive and appropriate actions taken. Thus, it will attempt to connect to the rosbridge WebSocket if the master is available and it is not connected and, disconnect from the WebSocket if the ROS master dies. Other operations that require live update of the internal file content status of the robot like the listing, download and deletion of maps, images and videos also happens through the internal WebSocket.

On a successful startup, if the operator is not currently logged in, the authentication page is the only page that will be served. Upon successful login, the operator is redirected to the Control page which serves as the main dashboard for operating the robot. The other two (2) main pages are the Mapping and Setting Page, and will be discussed at length continuing.

During correct operation of the web page, should there be any disconnection to the rosbridge server socket, the roslibjs library connection re-attempt is stopped after five(5) attempts and connection restarted. The FLASK web page makes use of the loaded libraries including but not limited to the roslibjs, ros2djs and nav2d to communicate with the robot **agent** or logic. Simply put, any action direct or indirect that changes the status of the robot, performed by the operator, passes through the afore mentioned libraries.

Finally, video stream coming from the robot base is published on port '9091'.

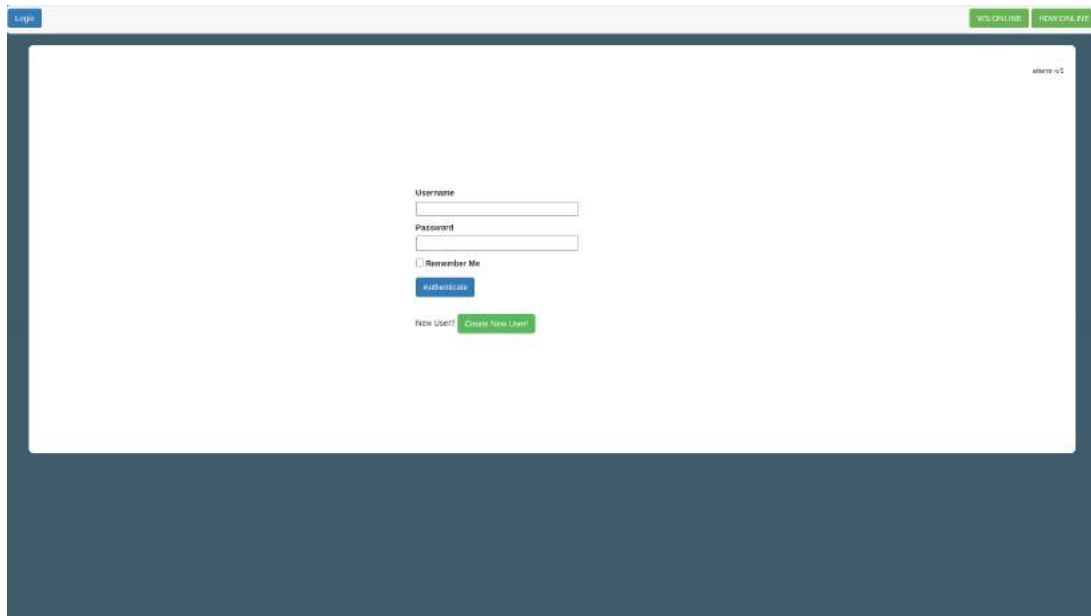
## The Authentication Page

As mentioned already in the previous section, when an operator first visits the web interface, if not already logged in, the Authentication or *Auth* page is served as seen in Figure 3.36.

This page has a status bar at the top consisting of a Login/ Logout button and two indicators showing the rosbridge socket referred to as 'WS:ONLINE/WS:OFFLINE' and internal socket referred to as 'HDW:ONLINE/HDW:OFFLINE' connections. There exist a third text option 'WS:ERROR' and 'HDW:ERROR' displayed if an error is encountered. The top status bar is present on all other pages and also shows the login name of the current operator upon successful authentication.

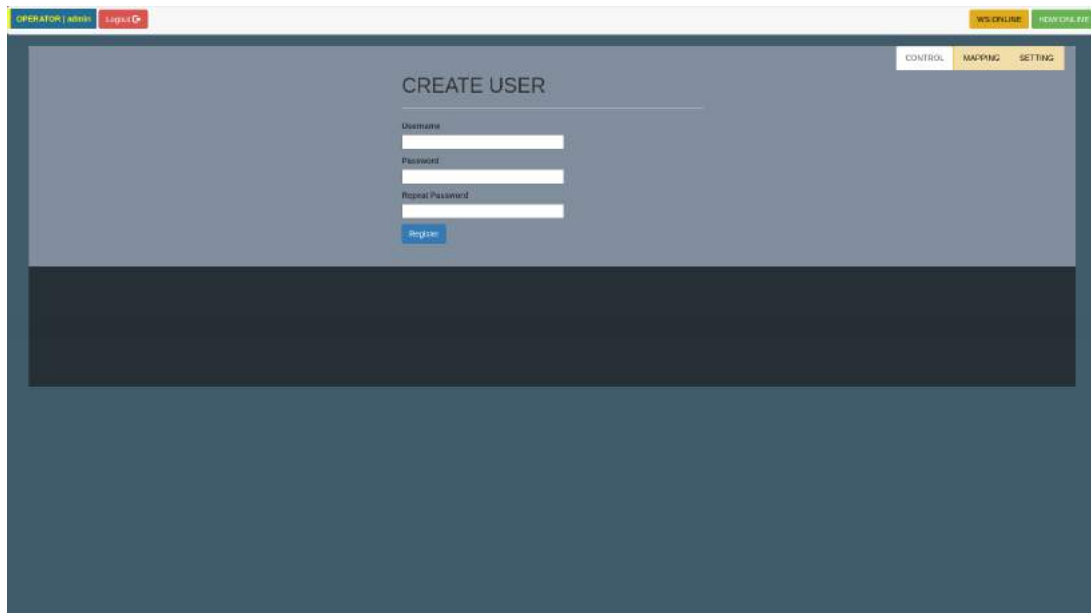
The main body of the Auth Page contains two inputs for 'Username' and 'Password', a checkbox to persist the username data and two buttons to authenticate or create a new user. The top-right corner of the body also shows the current version of the web page.

Once the correct credentials are provided and successfully authenticated, the operator is redirected to the Control Page. On the other hand, clicking the 'Create New User!' button



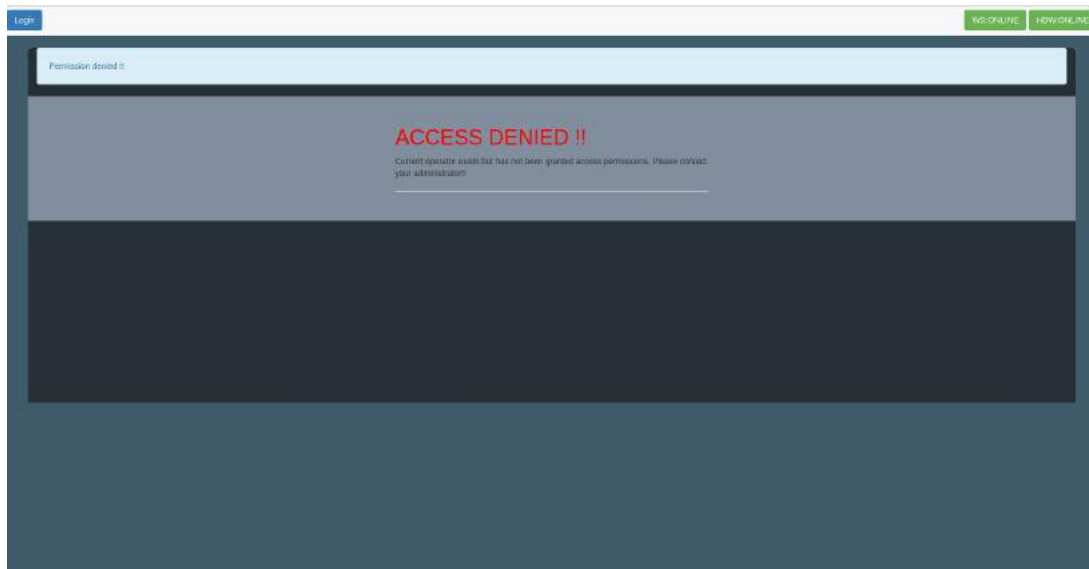
**Figure 3.36:** The Web-UI Authentication Page

will open another page or mini-page, Figure 3.37 where a new operator maybe added to the robot database.



**Figure 3.37:** The Web-UI Create User Page

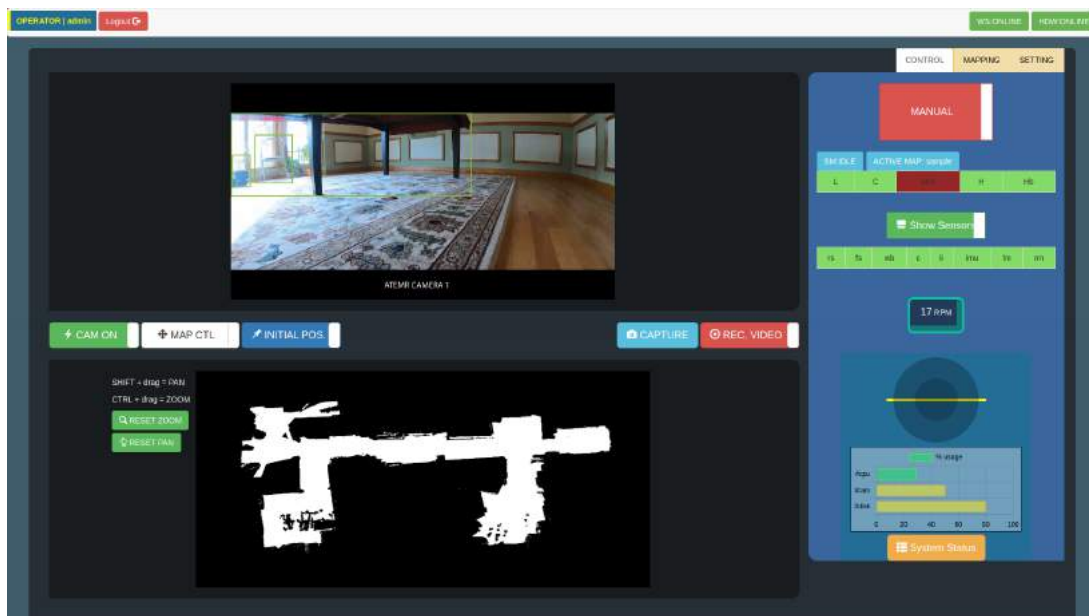
One thing to take notice of is that an operator can exist but may not have the correct permissions in order to operate the robot. When an operator without the correct access permissions tries to log on, the no-access page shown in Figure 3.38 is served and log-in is denied.



**Figure 3.38:** The Web-UI No-Access Page

## The Control Page

The main robot operation page also known as the *Control Dashboard* is the default go to page upon a successful login, Figure 3.39.



**Figure 3.39:** The Web-UI Control Dashboard

The main body of this page is split into two main panels: the Camera and Map visualization panel (on the left), and the Status and Control panel (on the right). The *Camera and Map visualization panel* is further split horizontally into two halves with the top half containing

the Camera stream and the bottom half, the Map visuals. A horizontal interaction bar is sandwiched by both halves and contains buttons for (going from left to right);

- (a) *Camera Feed Toggle*: It is by default in the 'ON' position (CAM-ON) and displays the live camera feed from the robot base. When toggled to the 'OFF' position (CAM-OFF), the feed is terminated. This can be used in instances when there is a tight budget on bandwidth.
- (b) *MAP control Type 1*: This is one of two buttons/ toggles for interacting with the Map visualization window. When toggled 'ON' (MAP CTL) by default, the operator can perform manouevres such as pan and zoom on the Map canvas and when toggled 'OFF' (SEND CMD), any event registered in the Map area will be interpreted as a robot command based on the state of MAP control Type 2.
- (c) *MAP control Type 2*: By default also in the 'ON' position (INITIAL POS.), this means that any commands sent in this mode will cause the robot to attempt to re-position itself. On the contrary, when toggled to the 'OFF' position (SEND GOAL), all registered events will be interpreted as goal commands and the robot will attempt to navigate to the given operator coordinates. Note however in order to enter into this mode, the robot needs to be in *AUTO MODE* else, it will fail to toggle.
- (d) *CAPTURE*: This button captures and saves images on click. It will only do so provided the Camera feed is in the 'ON' position.
- (e) *RECORD VIDEO*: This is the last toggle on the far right of the interaction bar and is used to record the Camera stream for future referencing and or analysis. It will only work when the Camera feed is in the 'ON' position.

The Map visuals on the bottom half of the Camera and Map visualization panel has two buttons for resetting the map *zoom* and *pan* operations.

The *Status and Control panel* on the right-most edge of the page however, allows for very different set of operations and interactivity. Going from top down;

- (a) *Interface Tab*: There three(3) main tabs. These tabs enables the operator to switch between the three(3) main operation pages: Control, Mapping and Setting.
- (b) *Manual - Auto mode Toggle*: This toggle allows the operator to switch between AUTO and MANUAL MODES. For autonomous navigation and planning, the operator is required to first switch to AUTO MODE and for this to happen, it is preferred for all *Level 1 Robot States* with the exception of the man-auto 'M/A' to be in a valid state. Else, the operation fails and is switched back to the MANUAL MODE.
- (c) *State Machine and Active MAP Indicators*: These two indicators show the current state of the robot agent state-machine and also, the current map loaded by the map server



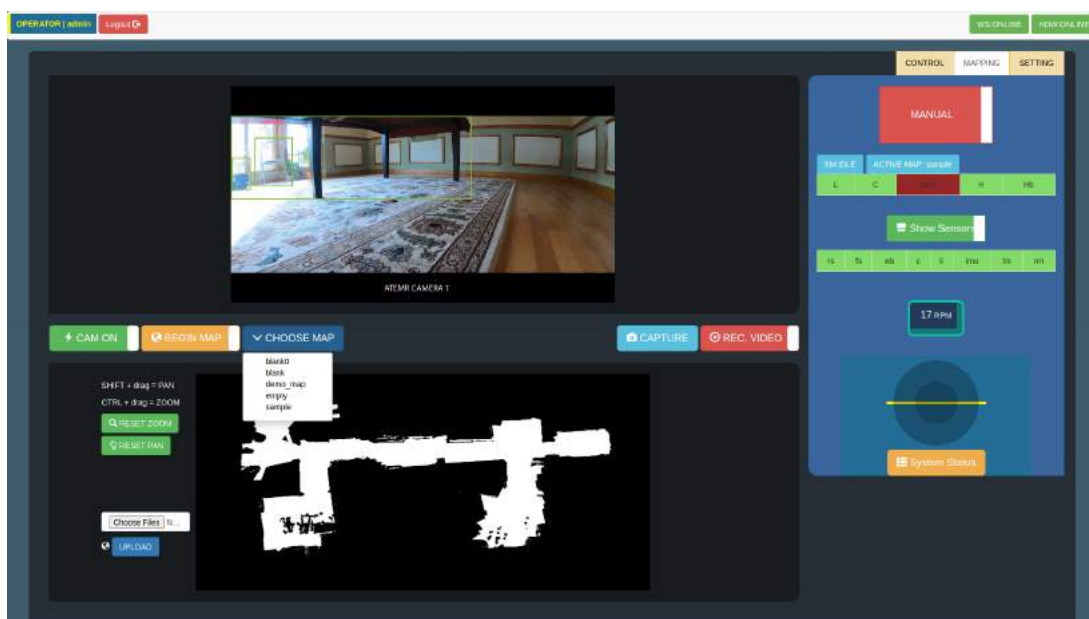
of the robot.

- (d) *Level 1 Robot State Display*: Shows the high-level sensor state of the robot
- (e) *Level 2 Sensor Display Toggle*: Toggles on/off the Level 2 Robot State Display
- (f) *Level 2 Robot State Display*: Shows the low-level sensor state of the robot.
- (g) *Velocity Feedback and Emergency Indicators*: This reports the current wheel velocities of the robot and also indicates if the robot base is in emergency mode (low level brakes applied) or not.
- (h) *Robot Joystick*: This allows the operator to send velocity commands to the robot base and hence control the robot when running in MANUAL MODE.
- (i) *Robot Hardware Status Graph*: The data displayed in this graph is updated at a frequency of 1Hz and contains the available memory, CPU and disk usage.
- (j) *Robot Hardware Status Button*: This button toggles on and off the Robot Hardware Status Graph.

Also, the *Status and Control panel* is persistent and functions the same on the Mapping Page when triggered from the Interface Tab.

## The Mapping Page

The Mapping Page also known as the *Mapping Dashboard* is organized and works just like the *Control Page*: previously discussed with a few minor modifications, Figure 3.40.



**Figure 3.40:** The Web-UI Mapping Dashboard

The major changes on this page are; on the interaction bar:

- (a) *Map Creation Toggle*: This toggle by default shows (BEGIN MAP) and triggers the map making/ creation procedures of the robot. The robot must be in MANUAL MODE during the MAP creation process.
- (b) *Map Selection Dropdown*: This Dropdown lists all the maps existing on the robot and selecting any one of them will trigger the MAP change procedures of the robot.

And on the bottom half: in the *Camera and Map visualization panel*, two new features have been added that allows the operator to upload new maps from other robots or sources.

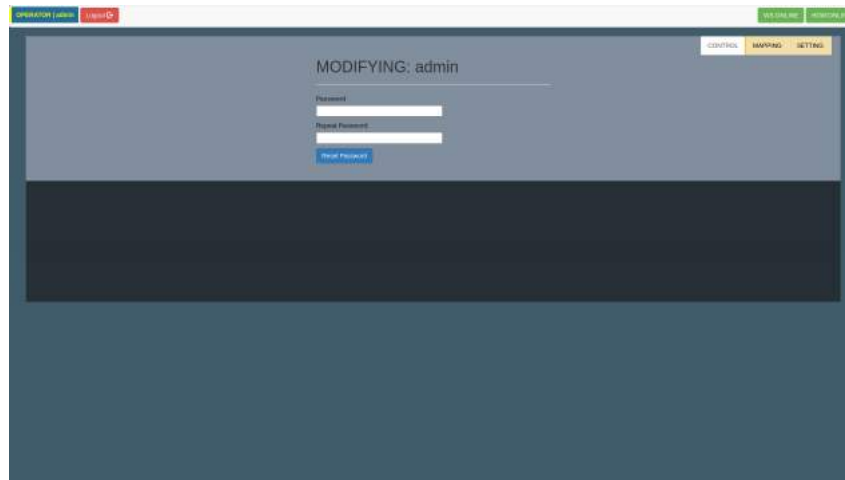
Seemingly, the Setting Page which is discussed next, can be triggered from the Interface Tab.

## The Setting Page

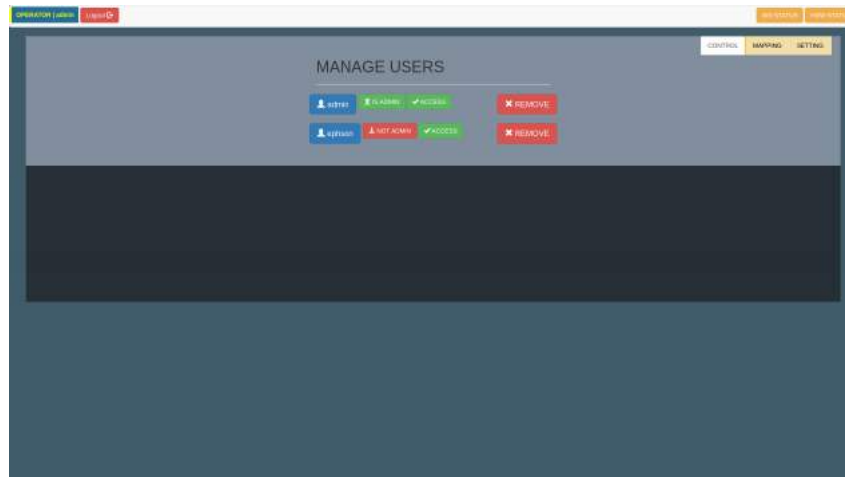
The Setting Page just like the Control and Mapping pages, has a similar organization. It has two main panels: the Parameter and Management panel (on the left) and the Media and Map panel (on the right).

The *Parameter and Management panel* has two main sections:

- (a) *Robot Parameters Section*: This section allows the operator to switch the robot to a different network by introducing the SSID and password in the corresponding input area. In addition, the linear and angular velocities can be changed here without needing to restart the robot stack. The 'UPDATE' button applies the inputted configuration and velocities to the robot base.
- (b) *Auth Section*: The content of this section is dynamic and varies according to the current user/ operator. If the operator has administrator level privileges, then the Current User, Manage Users and Add User buttons will be shown. Else, only the Current User button is shown.
  - (i) The *Current User Button*: When clicked opens a new page that allows the operator to modify/ change the current password, Figure 3.41.
  - (ii) *Manage Users Button*: This button when clicked opens another mini-page where all existing users and user roles can be modified and or removed. This page allows the administrator to grant or remove access privileges and also to grant or remove administrator privileges to normal users and other administrator level users alike as can be seen in Figure 3.42.  
Note however, the root administrator 'admin' cannot be removed neither can its access privileges nor system level access.



**Figure 3.41:** The Web-UI Setting Page: Current-User Page

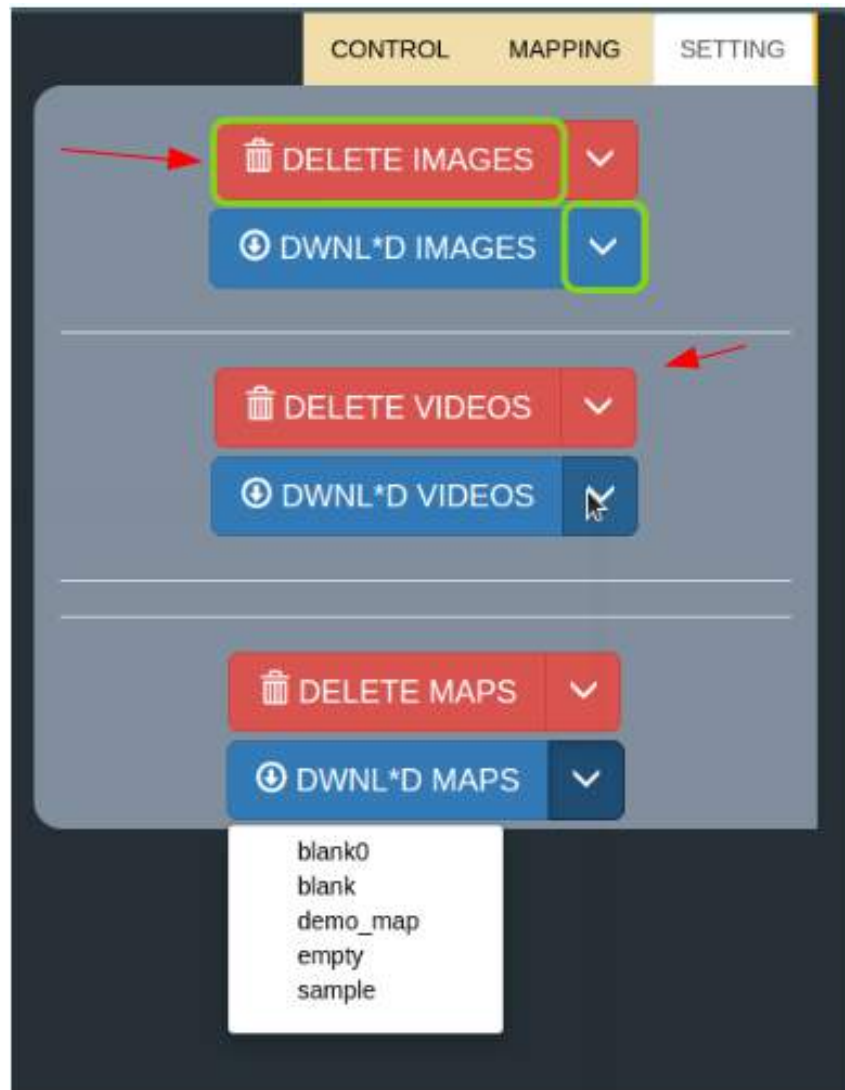


**Figure 3.42:** The Web-UI Setting Page: Manage-User Page

- (iii) *Add User* Button: This button allows for a new user/operator to be added to the robot database. When triggered, the a new page will be opened as shown in Figure 3.37 that facilitates the creation of a new user.

However, the *Media and Map panel* unlike the *Status and Control panel* also, introduces a different but much desired context that allows the operator to have access to captured images, recorded videos and uploaded/created maps existing on the robot, Figure 3.43.

The design used here is a split drop-down button style. Clicking on the main button (left side) causes the operation to be applied to all corresponding files whilst using the drop-down button (right side) lists the top ten(10) items in the corresponding section. Clicking on any item in the list will cause the operation to be applied *ONLY* to the selected file.



**Figure 3.43:** The Web-UI Media and Map Panel

## 3.6 The Secondary Core

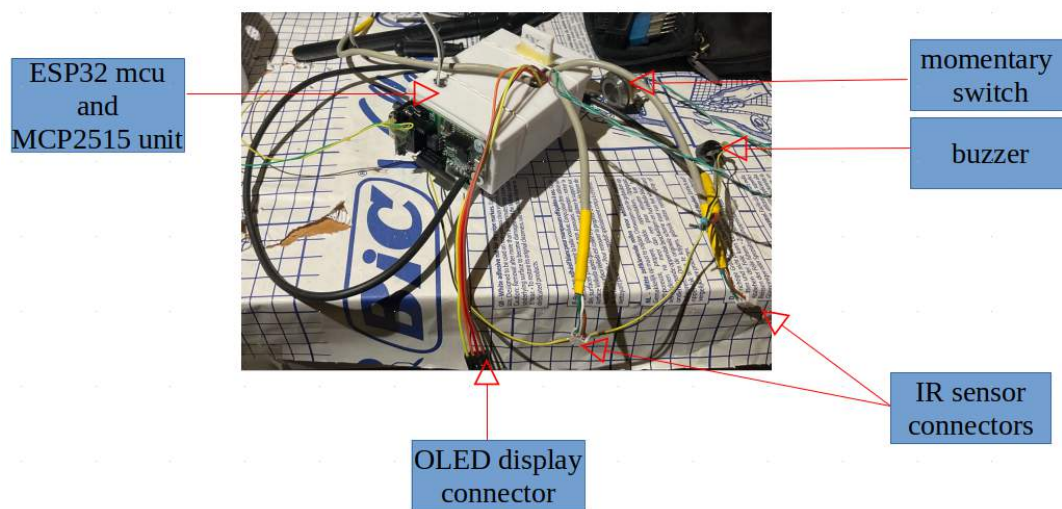
The secondary core as its name suggests operates in the background and is vital for the overall operation of the robot. Considering the myriad functions of the primary core consisting of but not limited to navigation, agent logic and environment mapping, the **secondary core** is provided to ensure vital functionalities needed for the operation of the **primary core** to successfully do its job are present and available.

The core comprises of a buzzer unit, a momentary switch with led, an MCP2515 CAN controller, three(3) Sharp GP2Y0A41SK0F Analog distance sensor units and an az-delivery ESP32 Micro-controller with the following specifications;

- Clock rate of up to 240 MHz

- 4MB Flash
- 520 KB SRAM
- Capable of PWM, I2C, SPI, UART, 1-wire, 1 analogue pin
- Wireless connectivity capabilities
- 3.3 V Operated, 15 mA output current per GPIO pin, 80 mA average working current

The core is powered by a 24v to 5v DC step-down converter connected to the robot's battery via a single-pole-single-throw (SPST) switch.



**Figure 3.44:** Secondary Core

The vital functionalities provided by the **secondary core** to the **primary core** as depicted in figure 3.45 are as follows:

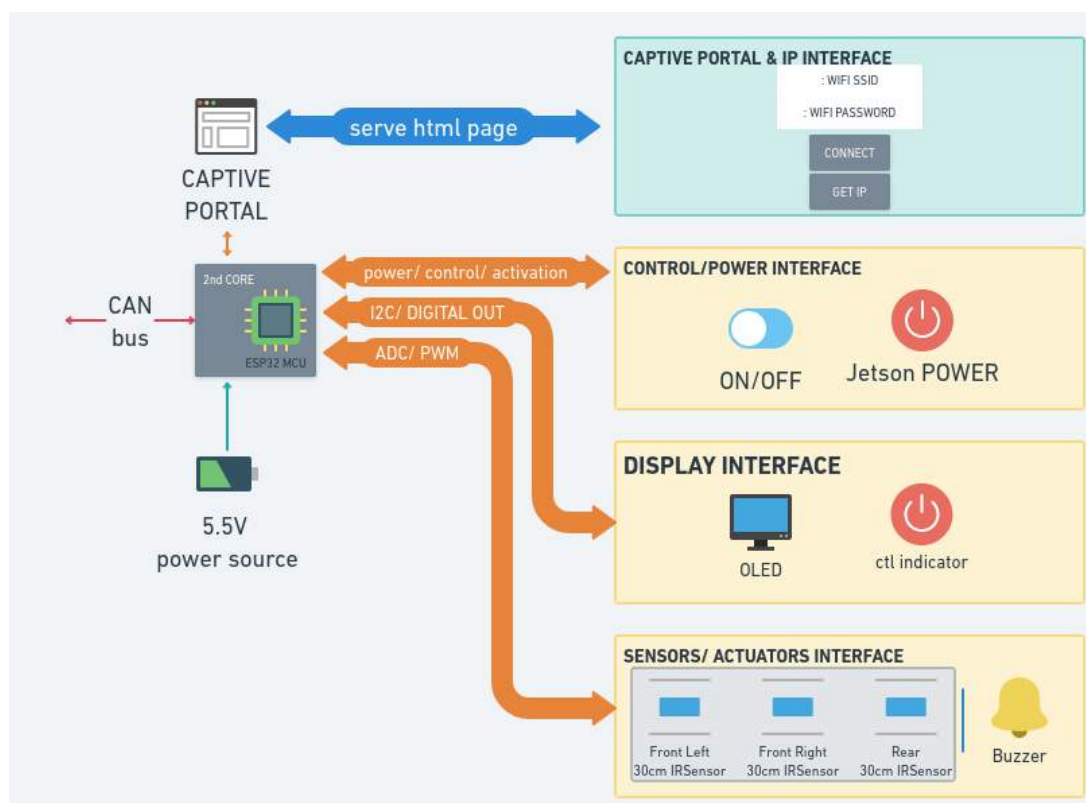
- Control-Power Interface
- Display Interface
- Captive Portal and IP address Interface
- Sensing and Actuating Interface

The main channel of communication between the **primary** and **secondary** cores is via the **CAN** bus. The communication channel uses a custom protocol defined for the sole purpose of this project centered around the **CAN** ID and data frame.

*NOTE* however that some messages are marked with an asterisk (“\*”), meaning they possess an internal significance to the robot’s code. These category of messages are said to be **extended**. The two main classifications are *byte-wise* breakdown and *bit-wise* breakdown of the dataframe.

### 3.6.1 BYTE-WISE CAN data frame

With reference to table 3.4, the messages that fall into this category are:



**Figure 3.45:** Secondary Core connection diagram

**Table 3.4:** CAN communication protocols

Direction	Command	CAN ID	Msg size (bytes)
Bi-directional	Power ON/OFF	0x150	*2 byte(s)
	IP Address	0x156	4 byte(s)
	Heartbeat	0x157	1 byte(s)
Incoming	Primary Core Status	0x153	*3 byte(s)
Outgoing	IRSensor front	0x151	8 byte(s)
	IRSensor rear	0x152	4 byte(s)
	WIFI SSID	0x154	*8 byte(s)
	WIFI PASSWORD	0x155	*8 byte(s)

(i) *POWER ON/OFF* makes use of bytes at positions 0 and 1.

**Table 3.5:** POWER ON/OFF (0x150)

Byte	Value	Function
[0]	<b>0</b>	Primary core not started or initializing
	<b>1</b>	Primary core running
	<b>2</b>	POWER OFF acknowledged
[1]	<b>0</b>	Normal operation
	<b>1</b>	POWER OFF request

- Byte 0 is only modified by the **primary** core to indicate POWER ON and also to acknowledge POWER OFF state of the robot
  - Byte 1 is only modified by the **secondary** core to trigger POWER OFF/ SHUT-DOWN process.
- (ii) The data will be transmitted in sequential bursts consisting of 7 data bytes and 1 control/ counter byte. The **WIFI SSID and WIFI PASSWORD** data is variable and almost always bigger than can be transmitted in a single burst. Hence, the number of sequential transmissions in this case is variable and dependent on the size of data to be transmitted.

### 3.6.2 BIT-WISE CAN data frame

With reference to table 3.4, the messages that fall into this category are:

- (i) **PRIMARY** *Core Statuses* makes use of bytes at positions 0, 1 and 2.
  - *Byte 0* -> AGENT STATE
  - *Byte 1* -> HARDWARE STATUS
  - *Byte 2* -> STATEMACHINE STATE



**Table 3.6: PRIMARY** Core Statuses (0x153)

Byte	Bit-Value	Function
[0]	<b>0</b>	Heartbeat
	<b>1</b>	Hardware
	<b>2</b>	Manual/ Auto
	<b>3</b>	Connectivity
	<b>4</b>	Localization
	<b>5</b>	*unused*
	<b>6</b>	*unused*
	<b>7</b>	*unused*
[1]	<b>0</b>	Left motor
	<b>1</b>	Right motor
	<b>2</b>	IMU
	<b>3</b>	Lidar
	<b>4</b>	Camera
	<b>5</b>	Webserver
	<b>6</b>	IR Sensor front state
	<b>7</b>	IR Sensor rear state

Byte - wise Interpreted

Byte	Value	Function
[2]	<b>0x00</b>	UNDEFINED
	<b>0x01</b>	STARTUP
	<b>0x02</b>	IDLE
	<b>0x03</b>	EXEC
	<b>0x04</b>	MAPPING
	<b>0x05</b>	SHUTDOWN

### 3.6.3 Secondary Core: Internal Workflow

The internal workflows of the **secondary** core ranging from the minute the core is powered on till it gets powered off is presented in the flowcharts 3.46 and 3.47 and detailed in the sequence diagram 3.48.

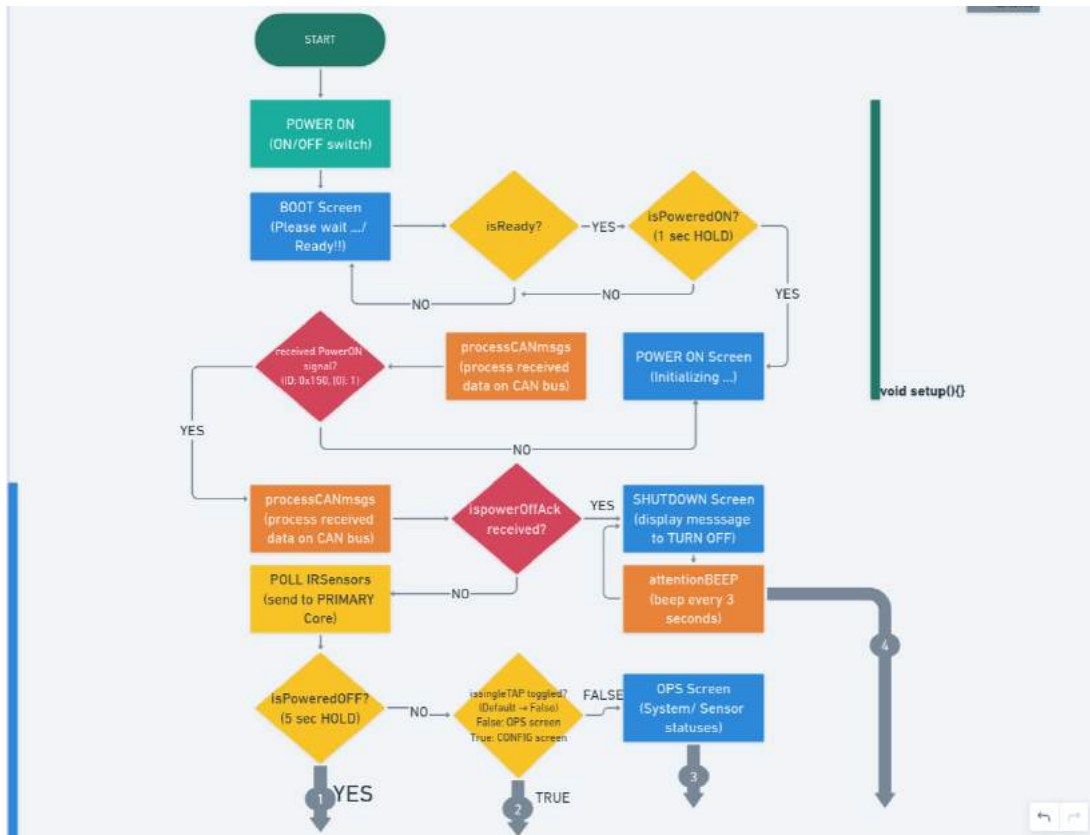


Figure 3.46: Secondary Core internal workflow - Flowchart

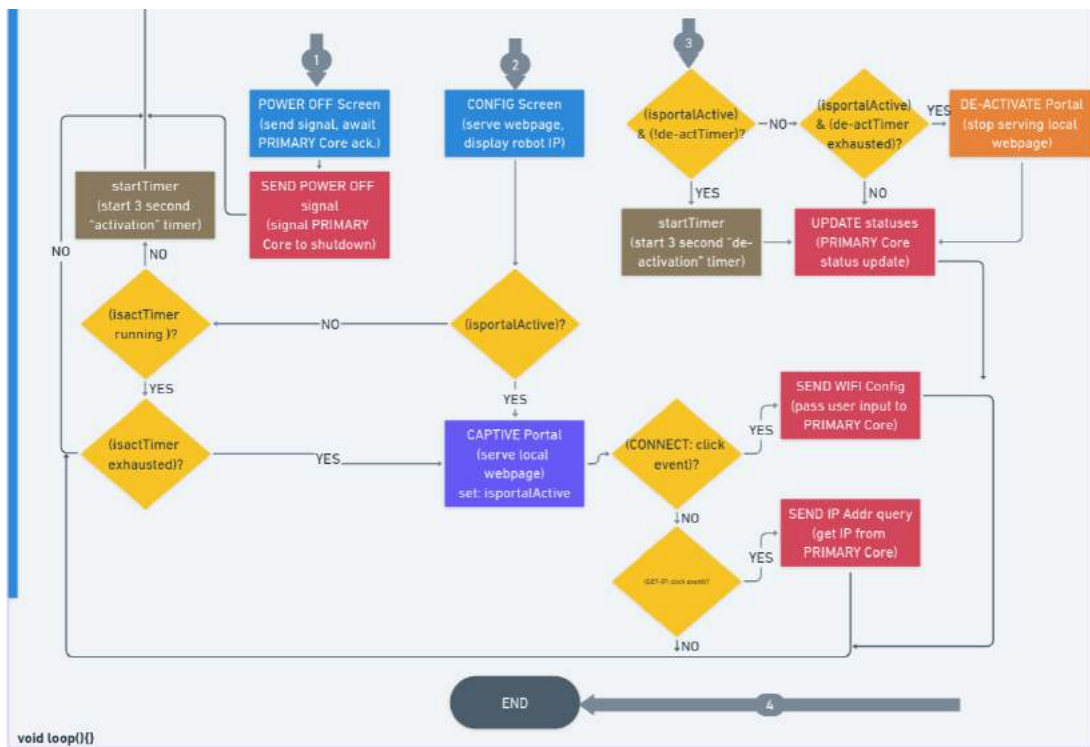
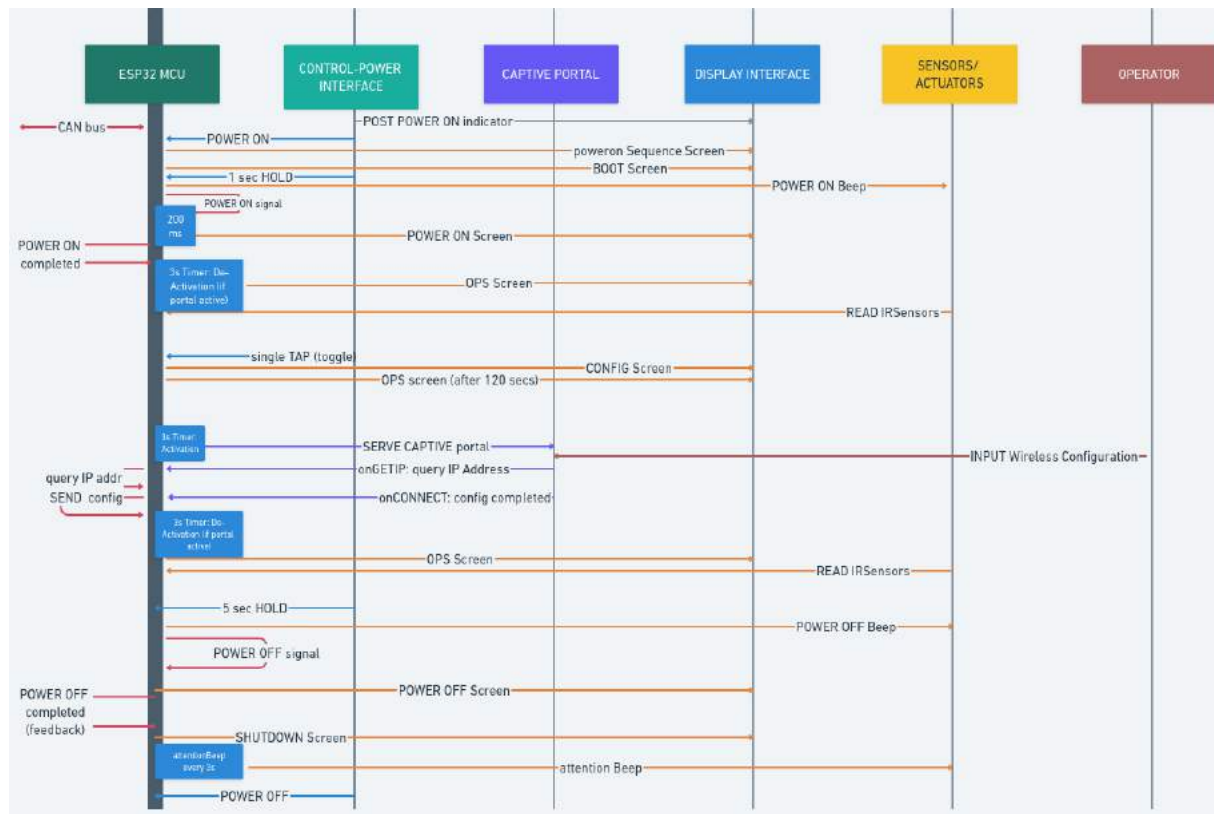


Figure 3.47: Secondary Core internal workflow - Flowchart cont'd



**Figure 3.48:** Secondary Core internal workflow - Sequence diagram

The two(2) main workflow sequences of interest are the *HEARTBEAT LOGIC - 0x157* and *POWER OFF LOGIC 0x150*. The robot uses byte 0 of the heartbeat data frame for sequencing the *\*isAlive\** checks between the **primary** and **secondary** cores. The primary core writes *0x01* in this position and expects to receive *0x00* in the same position. On the other hand, the secondary core expects to receive *0x01* in this byte position and writes *0x00* to this position afterwards. This process is activated the moment the secondary core receives the *\*ispoweredON\** signal from the primary core on startup.

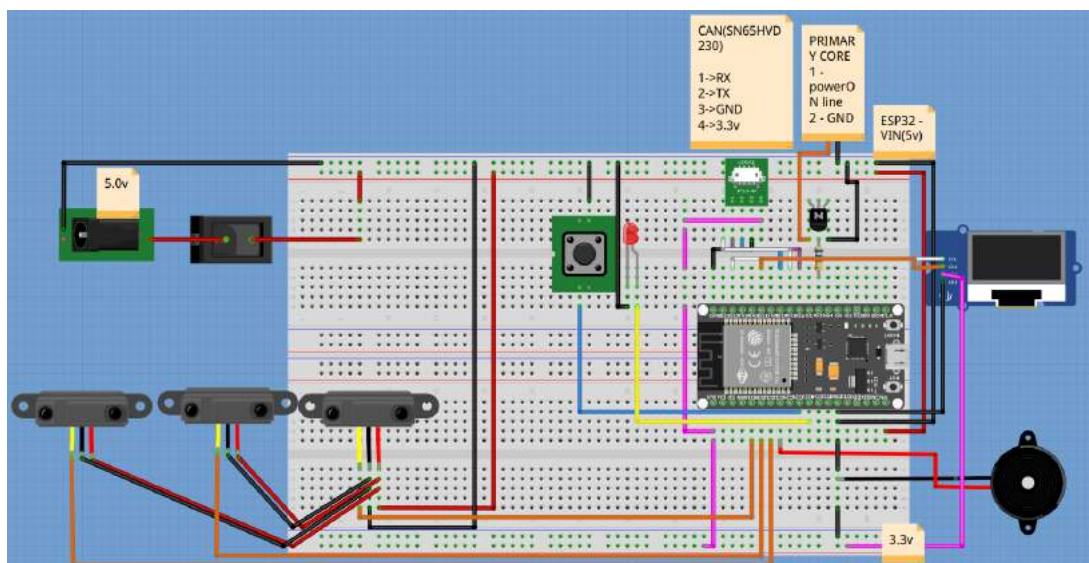
The *POWER OFF LOGIC* is a bit more complex since it involves two data frame bytes. The Power off sequence can be triggered in two ways,

- Press and Hold the power button for 5 seconds
- Clicking shutdown from the robot operator's web page

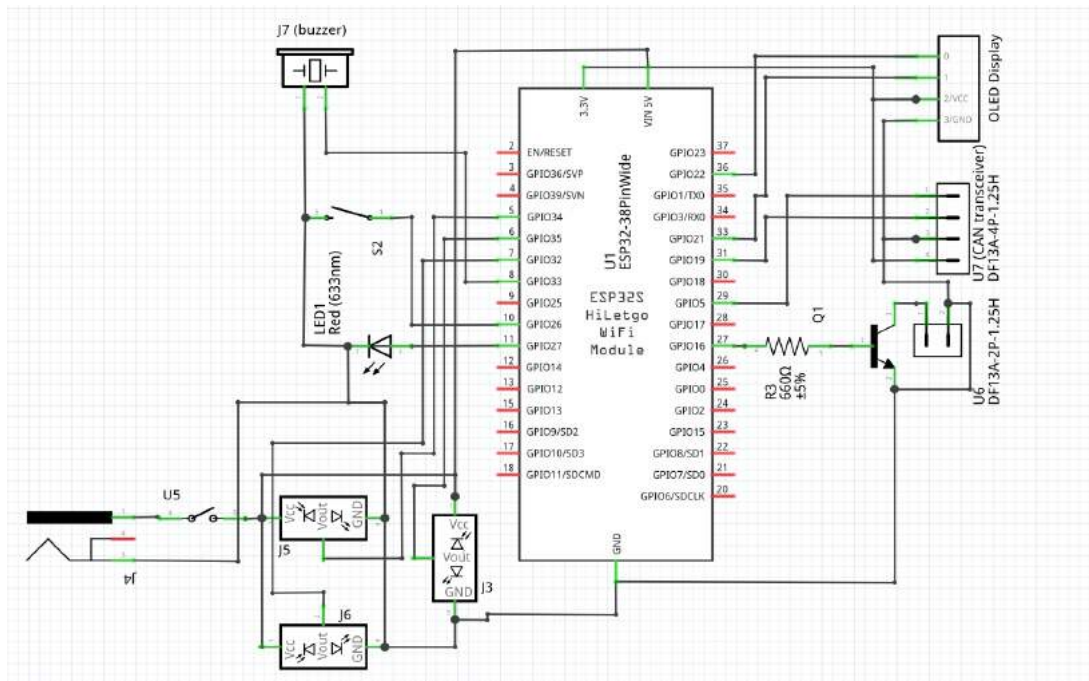
If triggered from the power button, the **secondary** core sends a power off request by setting the byte at position 1 to 1 and switches to the Power Off screen while it awaits confirmation from the **primary** core. The primary core upon receipt of the shutdown/ power off request will proceed to complete all current processes if it is currently busy and then

transitions to the *SHUTDOWN STATE* - 0x05 of the statemachine. It then sends the acknowledgement signal to the secondary core by setting the byte at position 0 to 2 and proceeds to shutdown the system completely. The secondary core waits for 5 seconds on receipt of the acknowledgement signal and then transitions to the *SHUTDOWN SCREEN* and prompts the operator to manually turn off the power switch.

The breadboard circuit diagram and circuit schematic showing the how secondary core is wired is shown in figures 3.49 and 3.50.



**Figure 3.49:** Secondary Core wiring - Breadboard connection



**Figure 3.50:** Secondary Core wiring - Schematic diagram

Details about the internal operations of the vital sub-module workflows are provided in subsequent sections.

## Control-Power Interface

The CONTROL/POWER interface sub-module facilitates basic operator interaction with robot. The five(5) main interactions are permitted via this interface to allow the operator to interface with the **primary** core are:

**POWER ON - Type 1** : This involves flipping the on/ off switch on the side of the robot in order to turn on the **secondary** core *ONLY*. The secondary core takes up to about 10 seconds to finish booting up. The secondary core startup sequence is completed when the display is showing the *BOOT/READY SCREEN*.

**POWER ON - Type 2** : To turn on the primary core, press and hold the power button for 1 second/ beep and then release. After that, the power on alert sound will play, the secondary core display will transition to the *POWER ON SCREEN* and the power button will remain illuminated in green - fixed. Note however that this should only be done once the secondary core has completed starting up and is showing the *BOOT/READY SCREEN*.

<sup>1</sup>NOTE however that **Type 1** operations only affect the secondary core whilst **Type 2** operations only affect the primary core.

**POWER OFF - Type 2** : To turn off the primary core, press and hold the power button for 5 seconds/ beeps and release. Once the power off sequence is triggered, the power off alert will play, the primary core will be alerted via CAN and the secondary core display will transition to the *POWER OFF SCREEN* awaiting shutdown confirmation from the primary core.

**POWER OFF - Type 1** : After the shutdown sequence has been triggered and confirmed, the secondary core display will transition to the *SHUTDOWN SCREEN* prompting the operator to power off the module by flipping the on/ off switch on the side of the robot to the *OFF* position.

**OPS/ CONFIG TOGGLE - Type 1** : To transition between the normal operation screen and the configuration screen of the robot, press and release the POWER button to toggle between the screens.

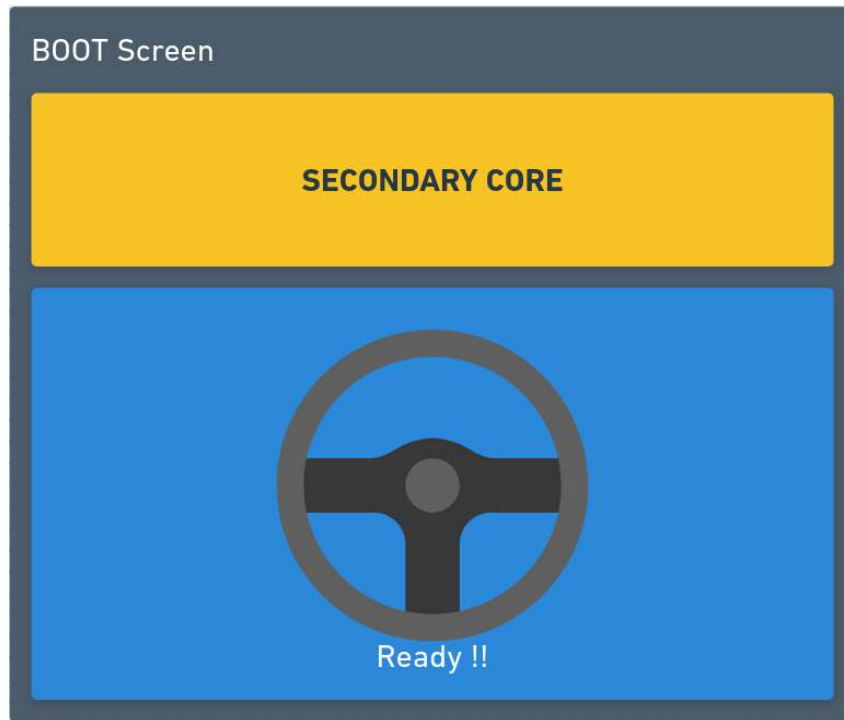
## Display Interface

The OLED display interface serves to make visible to the operator, the internal operations of the secondary core. The display is divided into two(2) parts. The top part with black text on a yellow background is the title section and the bottom part occupying the major part of the display interface with cyan coloured text on black background constitutes the body section.

The various *screens* available consists of:

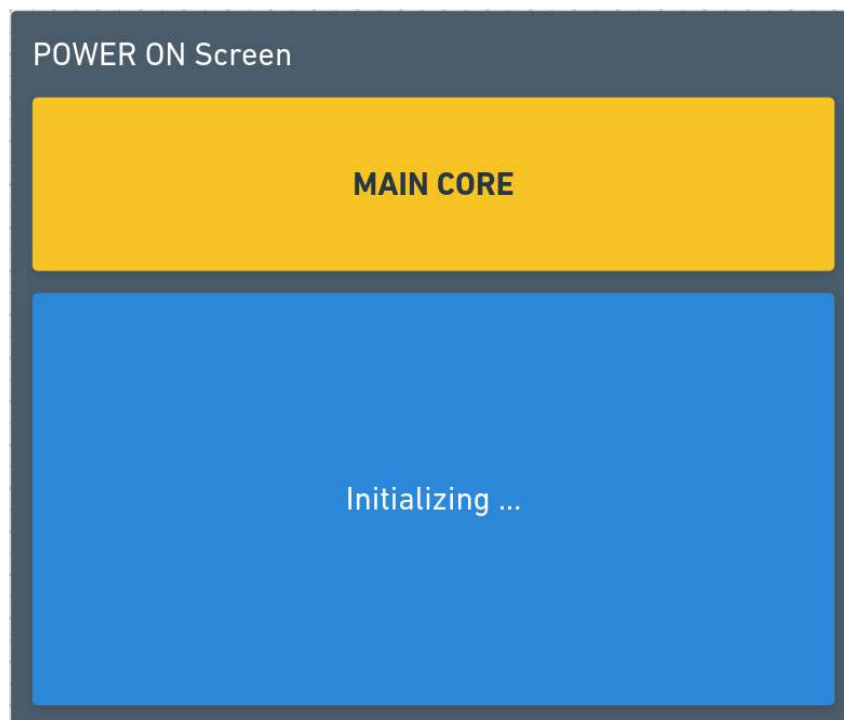
**POWER-ON SEQUENCE SCREEN** : This is the first screen that is shown on the display upon powering up the module. Aside the introductory ring animations, it also shows the name and author of the project and prompts the operator to wait as all other sub-modules are being loaded up. **Soft emergency** protocol is also activated at this stage and causes the secondary core to send braking signals to two motors (left and right) in order to prevent any spontaneous movements during the startup phase where the primary core is not yet active. Note however the module during this phase is receptive to any inputs whatsoever either coming from the operator via button presses or via the CAN communication channel.

**BOOT/ READY SCREEN** : The core transitions to this screen figure 3.51, after successful initialization of all sub-modules and emergency protocols and is ready to receive operator inputs. This screen consists of the title text “SC CORE”, a steering wheel (representing the primary and secondary cores) logo and the text “Ready !!” in the body section.



**Figure 3.51:** Secondary Core Display- BOOT SCREEN

**POWER-ON SCREEN** : When the module receives the external signal to turn on the



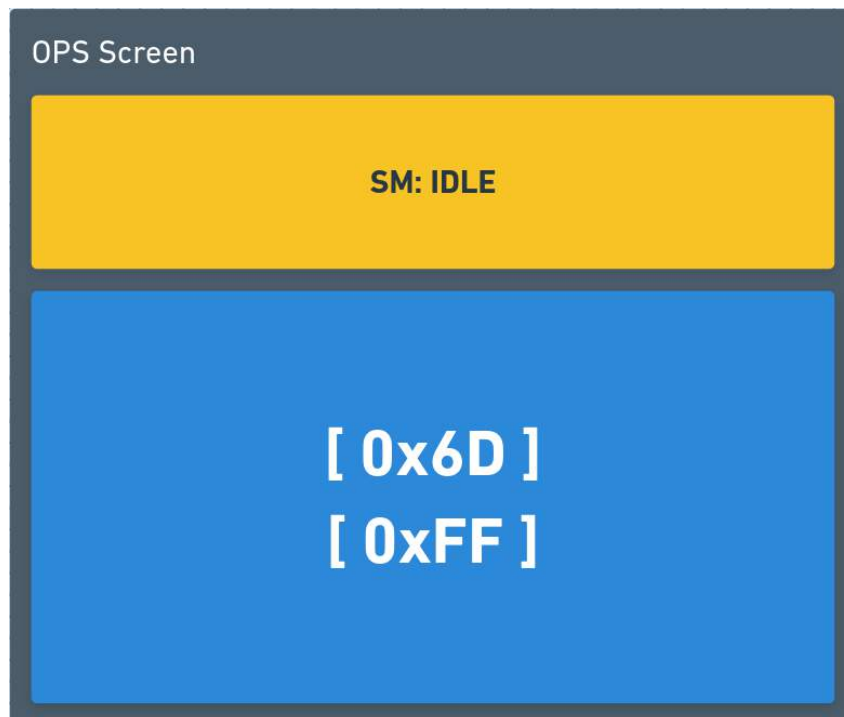
**Figure 3.52:** Secondary Core Display- POWER-ON SCREEN

primary core, it transitions to show this screen figure 3.52, and remains visible until the

power-on acknowledgement signal is received from the primary core. This screen displays a text informing the operator that the input was received the primary core is being powered on. It also has a progress animated bar that indicates to the operator the status of the primary core boot up process.

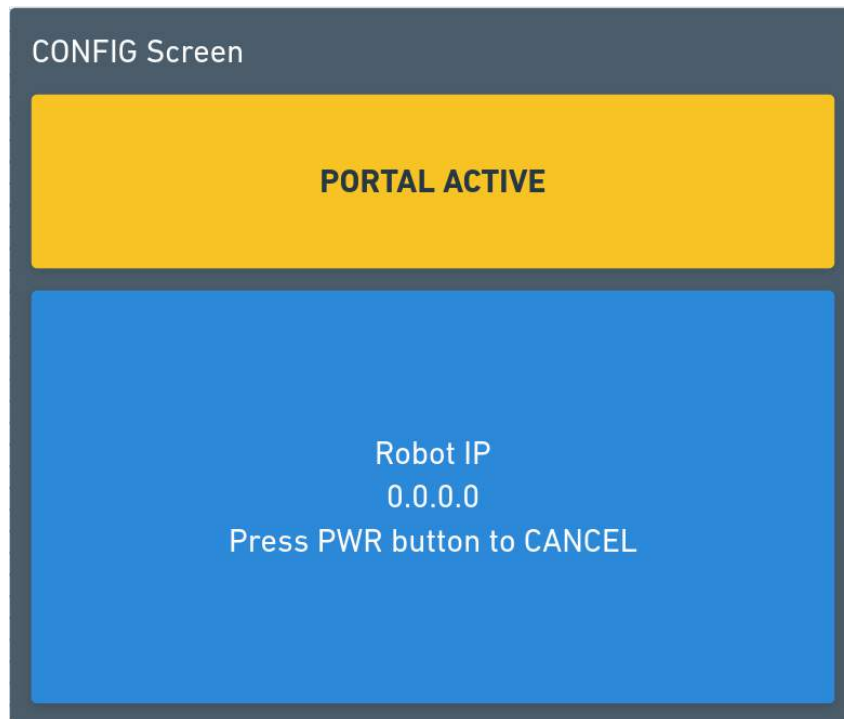


**OPS SCREEN** : Once the primary core is boot and startup sequences are completed, the module transitions to the Operations screen or Ops screen shown in figure 3.53. The main function of this screen is to display in detail, the internal status of the entire robot. This screen uses a special binary protocol and text to indicate the behavior agent state, the overall primary core sub-module statuses and the sensor(s) or hardware statuses. The behavior agent's state is indicated in text in the title section at the top of the screen and in black against a yellow background. The remaining two statuses are represented in binary format in cyan against a black background in the body section of the screen. The primary core sub-modules statuses are indicated against the text "AGENT" while the later are indicated against the text "HDW". Interpretation of the binary protocol used and agent state can be found at 3.6. Another event of important notice is that, upon transitioning to this screen, the captive portal server will be deactivated after three(3) seconds if active.



**Figure 3.53:** Secondary Core Display- OPS SCREEN

**CONFIG SCREEN** : The Configurations or Config screen can only be transitioned to after



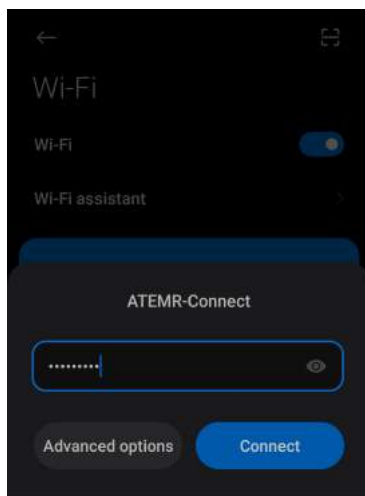
**Figure 3.54:** Secondary Core Display- CONFIG SCREEN

the primary and secondary core startup procedures are successfully completed via press and release of the power button. On transitioning to this screen depicted in figure 3.54, the captive portal is activated and the activation timer is refreshed. This happens every time irrespective of whether the portal is already active or not. The portal will remain active and this screen shown for a period of 120 seconds after which the portal is automatically deactivated and the module transitions back to the ops screen. On the configuration screen, the title text displays the captive portal status while the body text shows the current IP address of the robot if any, else, shows “0.0.0.0”.

### **Captive Portal & IP Address Interface**

The Captive Portal and IP address interface/ sub-module takes care of serving the local web page of the robot and responds any request sent from said web page. The main problem solved by this sub-module is that, it allows to easily configure the network to which robot should connect to. A feature which comes in handy especially at first deployments in new environment. Also, without resorting to any network administration wizardry and whatnot,

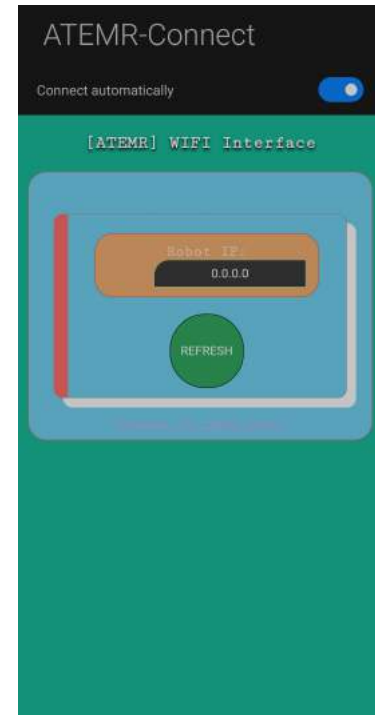
it displays the assigned IP address of the robot on the current network.



**Figure 3.55:**  
Secondary Core  
Captive Portal-  
JOIN NETWORK



**Figure 3.56:**  
Secondary Core  
Captive Portal-  
CONFIGURATION  
page



**Figure 3.57:**  
Secondary Core  
Captive Portal- IP  
ADDRESS page

The captive portal page can only be seen after successfully connecting to local network "ATEMR-Connect" with password "123456789" figure 3.55. The main page contains two buttons, "CONNECT" and "VIEW IP" buttons as can be seen in figure 3.56. The former upon inputting the desired WIFI SSID and password to which the robot should connect to, send the inputted data to the primary core after it is clicked. The secondary core responds with a sent success page containing a hyperlink to go back to the main page. The later opens the IP address page, figure 3.57 where the current IP address of the robot can be queried by clicking on the "REFRESH" button. The page also provides a hyperlink to the main page. Note however that if no IP address is available on entering the page for the first time, the refresh button would have to be clicked twice in order to get the IP address shown on the page. First click triggers the secondary core to query the primary core for its current IP address. The robot IP variable is updated when a response is received and the second click updates the page with the content of the robot IP variable and also queries the primary core

again.

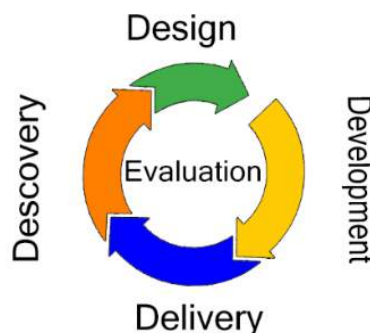
### Sensing & Actuating Interface

Three (3) sharp infrared distance sensors with a range of 4 - 30 cm, a buzzer, and momentary switch LED constitutes the Sensing and Actuating Interface sub-module. The IR sensors are configured with two (2) placed left and right at the lower front side of the robot with one (1) sensor placed at the center at the lower rear of the robot. The buzzer is located on the inner of the robot and is used to notify the power on/ off signal triggering , the power on/ off sequences and also to sound alerts at the secondary core level. The final component making up this sub-module, the momentary switch LED, is used to visually indicate all button interactions with the exception of toggling operations.

## 3.7 DESIGN CYCLE

Engineers, designers and developers have created a standard method to take projects from start to finish and solve the problem, Figure 3.58. This method is called the design or engineering cycle and it is used in all types of industries. It help problems get solved in an efficient manner.

The ATEMR project is developed incrementally using the agile approach. The early increments are identified but the development of later increments depends on prior successes of the individual modules and new discoveries.



**Figure 3.58:** ATEMR: Project Design Cycle

## Chapter 4

# ATEMR BUILD AND IMPLEMENTATION

### 4.1 INTRODUCTION

This chapter entails environment setup and preparations made during test runs. It reports on various images captured and their descriptions as the robot starts up, maps and navigates in the operating environment.

### 4.2 ROBOT STARTUP

When the main ON/OFF switch is toggled on, the Secondary Core runs. This initializes the OLED display which transitions through the power on sequence, displays the ready screen, Figure 4.1 and activates the motors when the primary core is ready to be activated.

On powering on the primary core, the two (2) main startup files that run are the web user interface server and agent statemachine scripts. The Web UI python server serves the welcome screen allowing users to be authenticated and have access to other system functionalities.

The agent statemachine on the other hand takes care of running all other system nodes and launch files, tests for their valid operation and transitions to the idle state where it awaits user input/ control. The map server starts up with an empty map loaded by default (see Figure 4.2). The velocity controls are also limited within safe operating parameters, Figure 4.3 on startup.



**Figure 4.1:** ATEMR: Secondary core ready (on startup)

### 4.3 ROBOT LOCALIZATION AND NAVIGATION

Now that all systems are running and status displayed as okay see Figure 4.4, the operator can proceed to set initial robot pose in the Control Tab, Figure 4.5 and 4.6. Sometimes the operator is required to drive around after setting the initial pose to aid the AMCL node achieve a faster convergence, Figure 4.7.



Figure 4.2: ATEMR: Default map

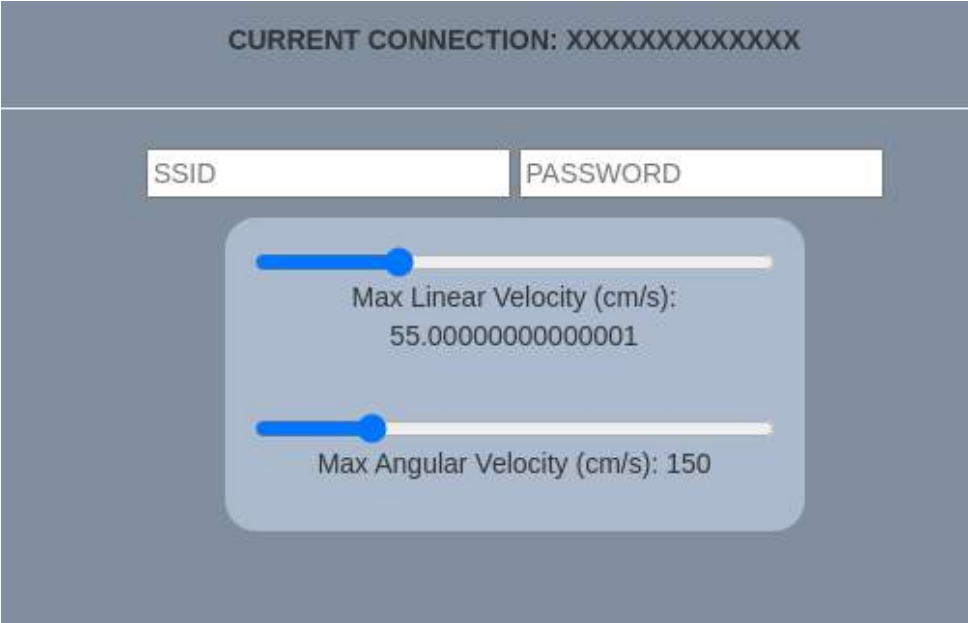
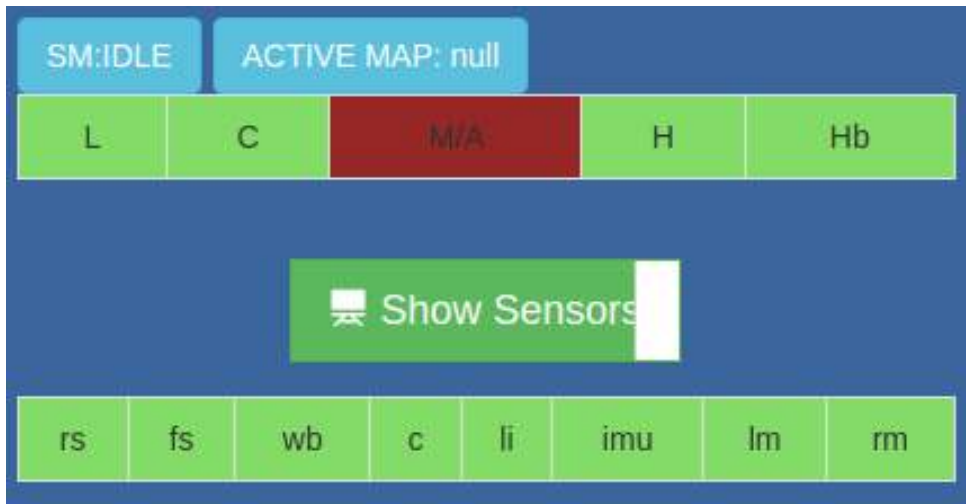
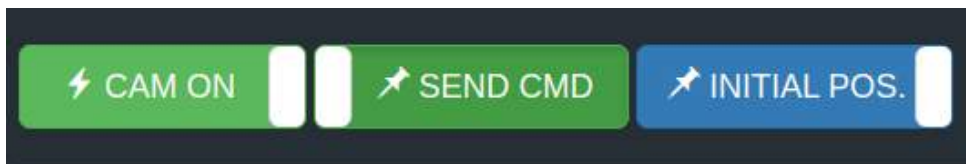


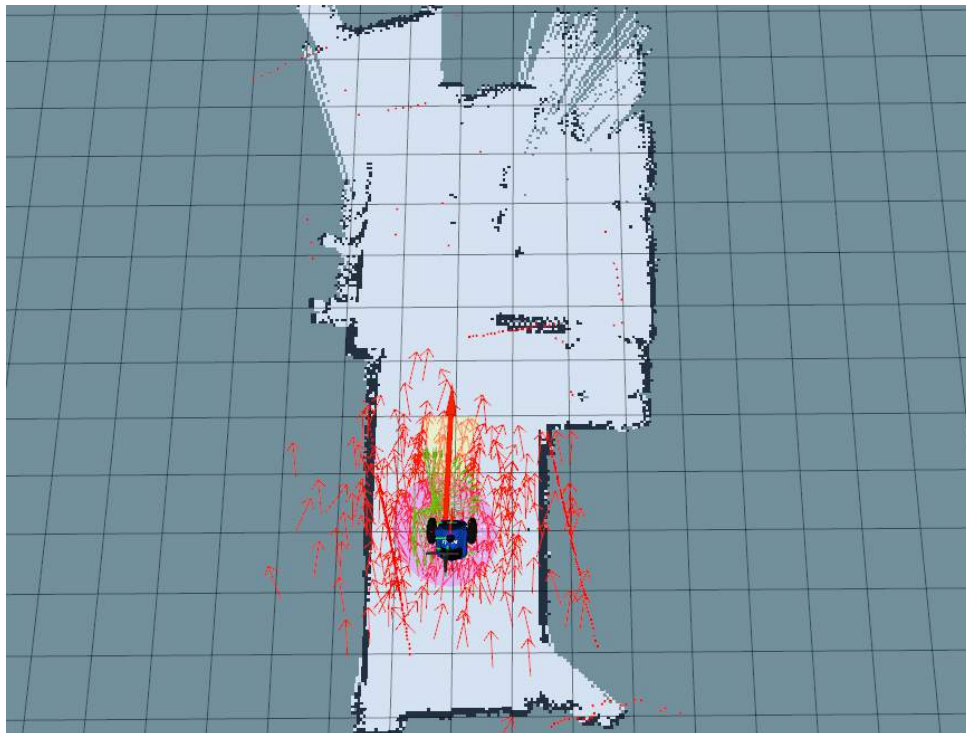
Figure 4.3: ATEMR: Default velocity limits



**Figure 4.4:** ATEMR: Nodes run status

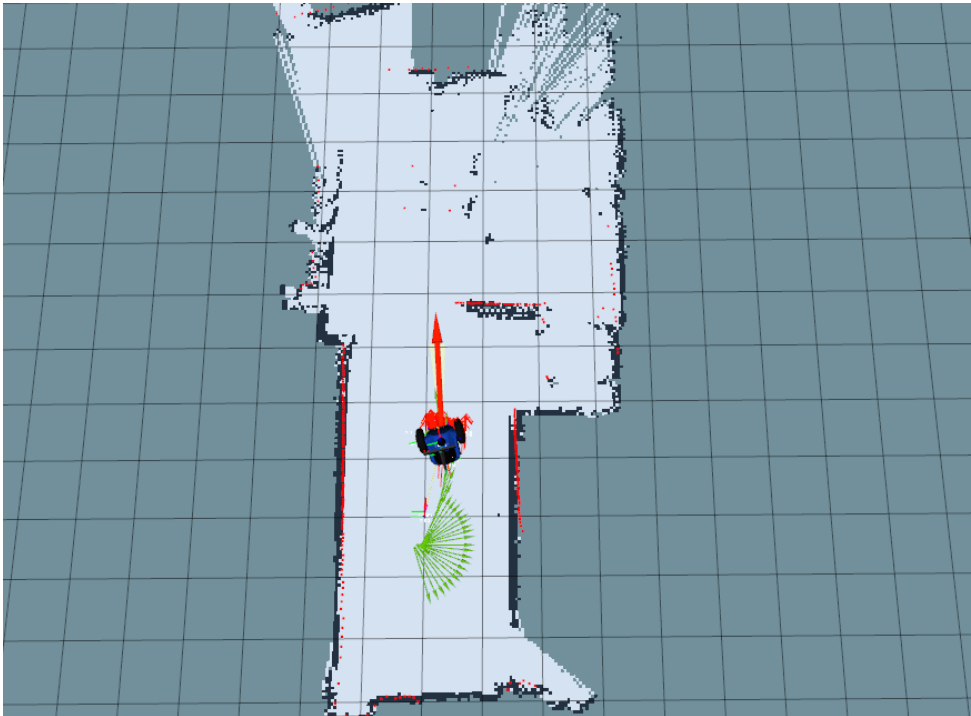


**Figure 4.5:** ATEMR: Setting initial robot pose



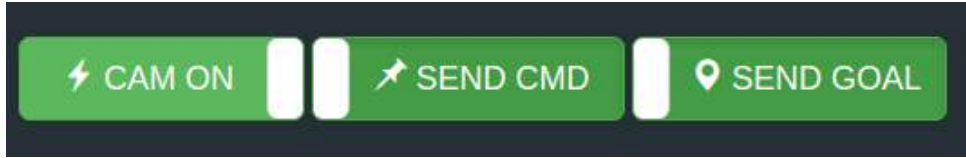
**Figure 4.6:** ATEMR: Setting initial robot pose (initial run : rviz)





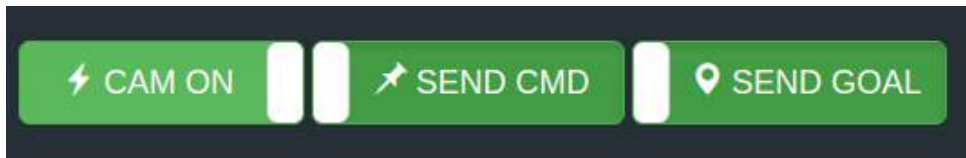
**Figure 4.7:** ATEMR: Setting initial robot pose (pose convergence : rviz)

Once the robot is localized, the operator can proceed to send goals by toggling the send goal button, Figure 4.8 and choosing a target point on the selected map.



**Figure 4.8:** ATEMR: Sending robot goal

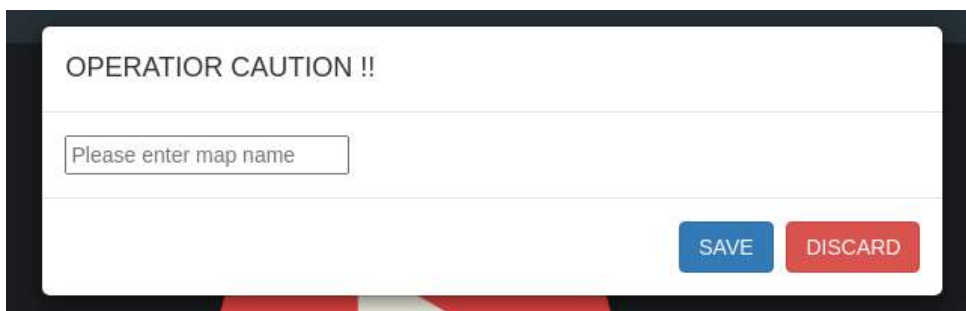
Figure 4.9 shows the mobile moving to a set goal point within the map.



**Figure 4.9:** ATEMR: Sending robot goal (rviz)

## 4.4 ROBOT MAPPING

The interface for mapping is straight forward. First the operator switches to the Mapping Tab and toggles the begin map slider. The operator can then proceed to drive around until the entire area is fully mapped at which point he will be asked if the generated map should be saved or discarded, Figure 4.10.



**Figure 4.10:** ATEMR: Map save dialog

## 4.5 NAVIGATION TESTING

In order to verify the prototype, the robot was stationed at one end of the room with a runway area of about 3 x 8 meters and made to navigate autonomous from one point to another.

This limitation in test area is mainly because that is the available workshop area when the test was done and is in no way a limitation to the robots ability to map and navigate in larger spaces.

The robot was made to repeat this trajectory 8 times continuously after setting the initial pose and localizing the mobile base. The move base node is configured with a tolerance of 25cm in the x-y plane and 0.13 radians for the goal yaw tolerance. It was observed after the eighth run that the covariance matrix, Figure 4.11 of the AMCL node starts to grow due accumulated error in the robot's local odometry. The AMCL node became lost when the robot was unable to reach its set target and rotated in place near an obstacle.

```
#AMCL-POSE LOCALIZED  
covariance: [0.018049881132575774, 0.0007398704789688126, 0.0, 0.0,  
0.0, 0.0, 0.0007398704789688126, 0.0023917028767439646, 0.0, 0.0, 0.0,  
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.004427198067955303]  
  
#AMCL-POSE WHEN LOST  
(the base rotated in place close to an obstacle )  
covariance: [0.5710807084972225, -0.07875365263898937, 0.0, 0.0, 0.0,  
0.0, -0.07875365263898937, 0.40108738636949975, 0.0, 0.0, 0.0, 0.0,  
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.26030546933469106]
```

**Figure 4.11:** ATEMR: AMCL Pose Covariance

Six (6) out of eight (8) times, it was able to reach the set destination. Below is a series of links to videos that were recorded during the testing phase.

- (i) **LOCAL ODOMETRY TESTING**): In this test the robot is driven around in a circular motion and the odometry covariance progression is monitored.
- (ii) **MOVEBASE SIMULATION TEST**: After setting up the Navigation stack, it is first tested and tuned on the simulation platform, Gazebo. This video shows how the mobile base behaved during the testing process.
- (iii) **NAVIGATION TEST**: One of the principal objectives of this work is for the mobile base to be able to move from one point to another autonomously. This video shows the autonomous navigation test conducted.
- (iv) **REACTIVE ENGINE SIMULATED IDLE TEST**: As mentioned in chapter 3, the low-level layer contains a self awareness feature or reactive engine for staying clear of obstacles. This video is simulated test of the operation of the engine when idle.
- (v) **REACTIVE ENGINE SIMULATED MANUAL TEST**: As mentioned in chapter 3, the low-level layer contains a self awareness feature or reactive engine for staying clear of obstacles. This video is simulated test of the operation of the engine when being driven by the

operator. In this test only forward (x linear) velocity was being sent. All mobile base turn velocities were generated by the reactive engine.

## **Chapter 5**

# **CONCLUSIONS AND RECOMMENDATIONS**

### **5.1 Summary**

At the start of this study, the project objective and scope was established. Having reviewed the start of art of an autonomous mobile robot, the requirements needed to reach the stated objectives for the project were outlined and metrics against which to measure such requirements were set.

The design and development of the proposed robot prototype was made possible by the use of major software like:

- (i) Solidworks
- (ii) Blender 3D
- (iii) ROS
- (iv) Gazebo Simulator
- (v) Robot Visualization (RViz)
- (vi) Fritzing

The steps for designing and building the proposed mobile base prototype and sub-systems like the primary core, secondary core and web interface has been discussed into detail in the previous chapters and the mobile base has been tested within correct operating parameters.

## 5.2 The Reactive Base Fuzzy-Inference Engine - Testing

The engine performs best at low speeds : maximum linear velocities of 0.26m/s and hence has an internal velocity limiting system implemented. This limitation is mainly due to the distance at which obstacles in HEX-PATTERN are being considered by the engine. In this study, the engine is configured to work with obstacles distances of 1.2m.

## 5.3 Conclusion

Referring back to Table 3.2, the designed prototype checks all the boxes successfully. The Table 5.1 compares the identified design requirements at the start of the project versus the obtained results after testing the prototype.

**Table 5.1:** ATEMR: Design specifications vs Obtained Objectives

Sequence	Requirement/ Specification	Objective Obtained?
1	Move to the specified target in an indoor environment	YES
2	Avoid obstacles when navigating to goal	YES
3	Provide visualization from robot's front camera	YES
4	A Web-based operator-centric graphical interface	YES
5	A low-level self-awareness control feature	YES
6	A telescopic mast for holding the Human Machine Interface	YES
7	Autonomous navigation and remote control	YES

A mobile robot that can be operated remotely either manually (MANUAL mode) or autonomously (AUTOMATIC mode), is able to avoid obstacles in its path when moving from one point to another via commands received from the web interface running on the onboard pc has been designed and implemented. A low-level fuzzy logic controller that can work on its own (self awareness feature) or run in unison with manual operator velocity commands has been designed and tuned for the purposes of this project.

It can therefore be concluded that, the design of an AUTONOMOUS TELEPRESENCE MOBILE ROBOT (ATEMR) for use in residential homes as a tool for medical and assistive care personnel is a success.

## 5.4 Recommendations

It can be recommended for future generations who might want to build further on this work, to:

- (i) Change the ESP32 CAN transceiver used since it worked poorly at higher baudrates.
- (ii) Research integrating a more industrial and robust Inertial Measurement Unit
- (iii) It was found during the mobile base autonomous navigation tests that, the odometry output gets corrupted after especially after the robot goes around corners. This causes the AMCL node also depends on the odometry data to become lost and needs to be reset. Future iterations that build on this work should research on writing an odometry monitor to automatically trigger local and global odometry resets when the covariance grows past a certain limit.
- (iv) Improve upon the Web Interface making it scale on smaller end devices like mobile phones and tablets.
- (v) Include a voltage sensor to feed remaining battery voltage levels to the on-board computer.

# References

- Abdallah, K., & Dan, Z. (2013). A fuzzy logic behavior architecture controller for a mobile robot path planning in multi-obstacles environment. *Research Journal of Applied Sciences, Engineering and Technology*, 5(14), 3835–3842.
- Adalgeirsson, S. O., & Breazeal, C. (2010). Mebot: A robotic platform for socially embodied telepresence. *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 15–22. <https://doi.org/10.1109/HRI.2010.5453272>
- Afanasyev, I., Sagitov, A., & Magid, E. (2015). Ros-based slam for a gazebo-simulated mobile robot in image-based 3d model of indoor environment. In S. Battiato, J. Blanc-Talon, G. Gallo, W. Philips, D. Popescu, & P. Scheunders (Eds.), *Advanced concepts for intelligent vision systems* (pp. 273–283). Springer International Publishing.
- Alexander Chilton. (2014). The working principle and key applications of infrared sensors [[Online; accessed 11-February-2022]]. <https://www.azosensors.com/article.aspx?ArticleID=339>
- An, Z., Hao, L., Liu, Y., & Dai, L. (2016). Development of mobile robot slam based on ros. *International Journal of Mechanical Engineering and Robotics Research*, 5(1), 47–51.
- Bachmann, KUKA Roboter GmbH. (2021). File:automation of foundry with robot.jpg – wikipedia commons, the free media repository [[Online; accessed 11-February-2022]]. <https://www.kuka.com/%7D%5Curl%7Bhttps://www.kuka.com/%7D>
- Bailey, T., & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics Automation Magazine*, 13(3), 108–117. <https://doi.org/10.1109/MRA.2006.1678144>
- Bayar, V., Akar, B., Yayan, U., Yavuz, H., & Yazici, A. (2014). Fuzzy logic based design of classical behaviors for mobile robots in ros middleware. *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*, 162–169. <https://doi.org/10.1109/INISTA.2014.6873613>
- Ben-Ari, M., & Mondada, F. (2018). Robots and their applications. *Elements of robotics* (pp. 1–20, 141–163). Springer International Publishing. [https://doi.org/10.1007/978-3-319-62533-1\\_1](https://doi.org/10.1007/978-3-319-62533-1_1)



- Borenstein, J., Everett, H. R., Feng, L., & Wehe, D. (1997). Mobile robot positioning: Sensors and techniques. *Journal of robotic systems*, 14(4), 231–249.
- Brandon, A. (2013). Rosbridge\_server [[Online; accessed 12-January-2022]]. [http://wiki.ros.org/rosbridge\\_server](http://wiki.ros.org/rosbridge_server)
- Censi, A. (2008). An icp variant using a point-to-line metric. *2008 IEEE International Conference on Robotics and Automation*, 19–25. <https://doi.org/10.1109/ROBOT.2008.4543181>
- Chui, C. K., Chen, G. et al. (2017). *Kalman filtering*. Springer.
- Company, E. P. (2021). What is an encoder [[Online; accessed 6-February-2022]]. <https://www.encoder.com/article-what-is-an-encoder>
- Conley, K., & Thomas, D. (2017). Rospy [[Online; accessed 12-January-2022]]. <http://wiki.ros.org/rospy>
- CRISP. (2001). Electromagnetic waves [[Online; accessed 11-February-2022]]. <https://crisp.nus.edu.sg/~research/tutorial/em.htm>
- Darweesh, H., Takeuchi, E., Takeda, K., Ninomiya, Y., Sujiwo, A., Morales, L. Y., Akai, N., Tomizawa, T., & Kato, S. (2017). Open source integrated planner for autonomous navigation in highly dynamic environments. *Journal of Robotics and Mechatronics*, 29(4), 668–684.
- Del Moral, P., Doucet, A., & Jasra, A. (2011). On adaptive resampling procedures for sequential monte carlo methods. hal-inria rr-6700-2008. *Bernoulli*, 2496–2534.
- Dictionary, O. E. (2016). Definition of 'robot'.
- Do Quang, H., Manh, T. N., Manh, C. N., Tien, D. P., Van, M. T., Tien, K. N., & Duc, D. N. (2020). An approach to design navigation system for omnidirectional mobile robot based on ros. *International Journal of Mechanical Engineering and Robotics Research*, 9(11), 1502–1508.
- Dragičević, S., & Jovanović, M. D. (n.d.). Robot operating system and its implementation on pc architecture.
- Dryanovski, I., Morris, W., & Censi, A. (2019). Laser scan matcher [[Online; accessed 3-February-2022]]. [http://wiki.ros.org/laser\\_scan\\_matcher](http://wiki.ros.org/laser_scan_matcher)
- Dudek, G., & Jenkin, M. (2010). *Computational principles of mobile robotics* (2ed.). Cambridge University Press.
- Gandhinathan, R., & Joseph, L. (2019). *Ros robotics projects* (Second Edition). Packt Publishing.
- Gerkey, B. (2020). Amcl [[Online; accessed 3-February-2022]]. <http://wiki.ros.org/amcl>
- Gerkey, B., & Pratkanis, T. (2020). Map server [[Online; accessed 3-February-2022]]. [http://wiki.ros.org/map\\_server](http://wiki.ros.org/map_server)

- GIS Geography. (2021). How gps receivers work – trilateration vs triangulation [[Online; accessed 6-February-2022]]. <https://gisgeography.com/trilateration-triangulation-gps/>
- Grewal, M. S., Andrews, A. P., & Bartone, C. G. (2020). Kalman filtering. *Global navigation satellite systems, inertial navigation, and integration* (pp. 355–417). <https://doi.org/10.1002/9781119547860.ch10>
- Grisetti, G., Kümmerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4), 31–43.
- Gustafsson, F. (2010). Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7), 53–82.
- INFRATEC. (2022). Infrared sensor - ir sensor [[Online; accessed 6-February-2022]]. <https://www.infratec.eu/sensor-division/service-support/glossary/infrared-sensor/>
- Johnson, R. C. (2011). Gps system with imus tracks first responders [[Online; accessed 11-February-2022]]. <https://web.archive.org/web/20121003161057/http://www.eetimes.com/electronics-news/4216978/GPS-system-with-IMUs-tracks-first-responders>
- Kalogirou, S. A. (2013). *Solar energy engineering: Processes and systems*. Academic press.
- Kenyon, S. H., Creary, D., Thi, D., & Maynard, J. (2005). A small, cheap, and portable reconnaissance robot. *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IV*, 5778, 434–443.
- Kohlbrecher, S., Meyer, J., von Stryk, O., & Klingauf, U. (2011). A flexible and scalable slam system with full 3d motion estimation. *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*.
- Kristoffersson, A., Coradeschi, S., Severinson Eklundh, K., & Loutfi, A. (2011). Sense of presence in a robotic telepresence domain. In C. Stephanidis (Ed.), *Universal access in human-computer interaction. users diversity* (pp. 479–487). Springer Berlin Heidelberg.
- Lambtron. (2018). File:rotaryincrementalencoder.jpg [[Online; accessed 6-February-2022]]. <https://commons.wikimedia.org/w/index.php?title=File:RotaryIncrementalEncoder.jpg&oldid=505843145>
- LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.
- Magyar, B. (2021). Differential drive controller [[Online; accessed 7-February-2022]]. [http://wiki.ros.org/diff\\_drive\\_controller](http://wiki.ros.org/diff_drive_controller)

- Marafa, N. A., de Britto Vidal Filho, W., & Llanos, C. H. (2020). A design methodology and development of a mobile telepresence robot for paraplegics. *Product: Management and Development*, 18(2), 181–195.
- Marder-Eppstein, E. (2020a). Move base [[Online; accessed 3-February-2022]]. [http://wiki.ros.org/move\\_base](http://wiki.ros.org/move_base)
- Marder-Eppstein, E. (2020b). Navigation [[Online; accessed 7-February-2022]]. <http://wiki.ros.org/navigation>
- Marder-Eppstein, E., & Perko, E. (2019). Trajectoryplannerros [[Online; accessed 7-February-2022]]. [http://wiki.ros.org/base\\_local\\_planner#TrajectoryPlannerROS](http://wiki.ros.org/base_local_planner#TrajectoryPlannerROS)
- Marder-Eppstein, E., V. Lu, D., & Hershberger, D. (2018). Costmap 2d [[Online; accessed 1-February-2022]]. [http://wiki.ros.org/costmap\\_2d](http://wiki.ros.org/costmap_2d)
- MaxBotix. (2022). Understanding how ultrasonic sensors work [[Online; accessed 6-February-2022]]. <https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm>
- Michaud, F., Boissy, P., Labonte, D., Corriveau, H., Grant, A., Lauria, M., Cloutier, R., Roux, M.-A., Iannuzzi, D., & Royer, M.-P. (2007). Telepresence robot for home care assistance. *AAAI spring symposium: multidisciplinary collaboration for socially assistive robotics*, 50–55.
- Moore, T., & Stouch, D. (2014). A generalized extended kalman filter implementation for the robot operating system. *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*.
- NASA Science. (2020). Perseverance rover [Last accessed 4 February 2022]. <https://mars.nasa.gov/mars2020/>
- Open Robotics. (2019a). 2d visualization library for use with the ros javascript libraries [[Online; accessed 12-January-2022]]. <http://wiki.ros.org/ros2djs>
- Open Robotics. (2019b). Roslibjs [[Online; accessed 12-January-2022]]. <http://wiki.ros.org/roslibjs>
- Oxford Dictionary. (2022). Gyroscope [[Online; accessed 11-February-2022]]. <https://www.lexico.com/definition/gyroscope>
- Paulos, E., & Canny, J. (1998a). Designing personal tele-embodiment. *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, 4, 3173–3178 vol.4. <https://doi.org/10.1109/ROBOT.1998.680913>
- Paulos, E., & Canny, J. (1998b). Prop: Personal roving presence. *Proceedings of the SIGCHI conference on Human factors in computing systems*, 296–303.
- Petteri Aimonen. (2011). Basic concept of kalman filtering [[Online; accessed 6-February-2022]]. [https://en.wikipedia.org/wiki/File:Basic\\_concept\\_of\\_Kalman\\_filtering.svg](https://en.wikipedia.org/wiki/File:Basic_concept_of_Kalman_filtering.svg)

- Pfeifer, N., & Briese, C. (2007). Laser scanning–principles and applications. *GeoSiberia 2007-International Exhibition and Scientific Congress*, ср–59.
- Piskorskii, D., Abdullin, F., & Nikolaeva, A. (2020). Optimization of a-star search algorithm. *Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника*, 20(1), 154–160.
- Pradhan, S. K., Parhi, D. R., & Panda, A. K. (2009). Fuzzy logic techniques for navigation of several mobile robots. *Applied soft computing*, 9(1), 290–304.
- Priyandoko, G., Wei, C. K., & Achmad, M. S. H. (2018). Human following on ros framework a mobile robot. *Sinergi*, 22(2), 77–82.
- Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., et al. (2009). Ros: An open-source robot operating system. *ICRA workshop on open source software*, 3(3.2), 5.
- Rada-Vilela, J. (2018). The fuzzylite libraries for fuzzy logic control. <https://fuzzylite.com/>
- Rahat. (2014). File:digital compass sensor and arduino uno.jpg – wikimedia commons, the free media repository [[Online; accessed 6-February-2022]]. [https://commons.wikimedia.org/w/index.php?title=File:Digital\\_Compass\\_sensor\\_and\\_Arduino\\_Uno.jpg&oldid=472866801](https://commons.wikimedia.org/w/index.php?title=File:Digital_Compass_sensor_and_Arduino_Uno.jpg&oldid=472866801)
- Rashid, R., Elamvazuthi, I., Begam, M., & Arrofiq, M. (2010). Differential drive wheeled mobile robot (wmr) control using fuzzy logic techniques. *2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation*, 51–55.
- RCraig09. (2020). 20200501 time of flight.svg [[Online; accessed 6-February-2022]]. [https://en.wikipedia.org/wiki/File:20200501\\_Time\\_of\\_flight.svg](https://en.wikipedia.org/wiki/File:20200501_Time_of_flight.svg)
- Reignier, P. (1994). Fuzzy logic techniques for mobile robot obstacle avoidance. *Robotics and Autonomous Systems*, 12(3-4), 143–153.
- Reinhold, A. (2019). File:apollo imu at draper hack the moon exhibit detail 1.agr.jpg [[Online; accessed 6-February-2022]]. [https://commons.wikimedia.org/w/index.php?title=File:Apollo\\_IMU\\_at\\_Draper\\_Hack\\_the\\_Moon\\_exhibit\\_detail\\_1.agr.jpg&oldid=619078577](https://commons.wikimedia.org/w/index.php?title=File:Apollo_IMU_at_Draper_Hack_the_Moon_exhibit_detail_1.agr.jpg&oldid=619078577)
- Ruiz, E., Acuña, R., Certad, N., Terrones, A., & Cabrera, M. E. (2013). Development of a control platform for the mobile robot roomba using ros and a kinect sensor. *2013 Latin American Robotics Symposium and Competition*, 55–60. <https://doi.org/10.1109/LARS.2013.57>
- Sherman, W. R., & Craig, A. B. (2019). Dedication. *Understanding virtual reality (second edition)* (Second Edition, p. v). Morgan Kaufmann. <https://doi.org/https://doi.org/10.1016/B978-0-12-800965-9.03001-4>

- Shiarlis, K., Messias, J., van Someren, M., Whiteson, S., Kim, J., Vroon, J., Englebienne, G., Truong, K., Evers, V., Pérez-Higueras, N., et al. (2015). Teresa: A socially intelligent semi-autonomous telepresence system. *Workshop on machine learning for social robotics at ICRA-2015 in Seattle*.
- Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.
- Simeon87. (2008). Motion planning workspace 1 configuration space 1.svg [[Online; accessed 6-February-2022]]. [https://en.wikipedia.org/wiki/File:Motion\\_planning\\_workspace\\_1\\_configuration\\_space\\_1.svg](https://en.wikipedia.org/wiki/File:Motion_planning_workspace_1_configuration_space_1.svg)
- Slamtec ROS. (2020). Rplidar ros [[Online; accessed 7-February-2022]]. <http://wiki.ros.org/rplidar>
- Thongchai, S., Suksakulchai, S., Wilkes, D. M., & Sarkar, N. (2000). Sonar behavior-based fuzzy control for a mobile robot. *Smc 2000 conference proceedings. 2000 IEEE international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no. 0, 5, 3532–3537)*.
- To, D. et al. (2022). Custom global planner for robots in retail environment.
- Tsai, T.-C., Hsu, Y.-L., Ma, A.-I., King, T., & Wu, C.-H. (2007). Developing a telepresence robot for interpersonal communication with the elderly in a home environment. *Telemedicine and e-Health*, 13(4), 407–424.
- Tsui, K. M., Von Rump, S., Ishiguro, H., Takayama, L., & Vickers, P. N. (2012). Robots in the loop: Telepresence robots in everyday life. *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, 317–318. <https://doi.org/10.1145/2157689.2157802>
- Tzafestas, S. G. (2013). *Introduction to mobile robot control*. Elsevier.
- Wang, H., Yu, Y., & Yuan, Q. (2011). Application of dijkstra algorithm in robot path-planning. *2011 second international conference on mechanic automation and control engineering*, 1067–1069.
- Wheatstone, C. (1879). *The scientific papers of sir charles wheatstone*. Taylor & Francis.
- Willing, N. (2021). Adoption of gallium-based lidar sensors gathers pace [[Online; accessed 11-February-2022]]. <https://www.argusmedia.com/en/news/2229445-adoption-of-galliumbased-lidar-sensors-gathers-pace>
- Willner, D., Chang, C., & Dunn, K. (1976). Kalman filter algorithms for a multi-sensor system. *1976 IEEE conference on decision and control including the 15th symposium on adaptive processes*, 570–574.