KLE TECH KLE Technological University
Creating Value
Leveraging Knowledge

Institutional Research Project

COURSE CODE: 17EARE491

Project Report on Humanoid Research Platform (MAYA 3.0)

DEPARTMENT OF AUTOMATION AND ROBOTICS

2023-2024

Team Members:

Chandan R Anpur	01FE20BAR011
Pruthviraj Kalal	01FE20BAR019
Chetan Kamatagi	01FE20BAR022
Nitin Mahadev Ghorpade	01FE20BAR038
Ashutosh Kangralkar	01FE21BAR410

Under the guidance of

Prof. Arun C Giriyapur

KLE TECHNOLOGICAL UNIVERSITY

DEPARTMENT OF AUTOMATION & ROBOTICS

Certificate

This is to certify that the below mentioned team has implemented the research project entitled "Humanoid Robot Research Platform (MAYA 3.0)" as a part of Institutional Research Project course in Department of Automation and Robotics, KLE Technological University, Hubballi, during 7th Semester of B.E program for the academic year 2023 – 2024. The project report fulfils the requirements prescribed.

Name	USN
1. Chandan R Anpur	01FE20BAR011
2. Pruthviraj Kalal	01FE20BAR019
3. Chetan Kamatagi	01FE20BAR022
4. Nitin Mahadev Ghorpade	01FE20BAR038
5. Ashutosh Kangralkar	01FE21BAR410

Project Guide: A.C. Giriyapur **Co-guide:**

Examiner 1:
Name:
Signature:

Examiner 2:
Name:
Signature:

ABSTRACT

MAYA 3.0 is a modular humanoid research platform developed as a sandbox for research in various fields, shifting the focus from theoretical learning to practical learning through projects. The research platform provides a mobile humanoid with a wide range of sensors and actuators. Behaviours can be implemented on the platform and can be tested and optimized.

This research aims to enable the platform to a variety of general features like visitor management system, person follower, human-machine interaction, carrying goods etc which can be adapted to different scenarios in different institutions. The physical, functional, hardware, software and cognitive architectures are developed by following the ISE-PPOOA systems engineering methodology which is helpful for software intensive systems. The robot is designed in a way that can function as two independent subsystems and also as an integrative humanoid robot. This robot is a good platform to develop specific featured robot for specific purposes without designing everything from scratch which saves lot of time, research and money. Also, this platform can be used for cognitive research purposes by students and researchers as it has high computational boards such as Jetson AGX Xavier.

The main motive behind the development of this robotic platform is to enable humanoids to read, listen and interpret user inputs and perform appropriate actions and generate responses based on the actions taken. The platform has been updated to work in a manner quite similar to smart assistants. It has personalized greetings based on the user's name and time of day. It has a database with relevant information about the institutes and can also answer any questions raised based on this information. In addition to these, it has functions that can be of great help in the form of a mobile helper robot that can carry loads from one place to another. Apart from this, it also has a few other sub-behaviors that further enhance its interactions and make it more human.

ACKNOWLEDGEMENT

The success and outcome of this project require a lot of guidance, supervision, and assistance from many people and we are extremely privileged to have gotten all this along with the completion of the project. It is now time to acknowledge our obligations and express gratitude towards all those involved in making this project a success

We respect and thank Prof. Arun Giriyapur, for providing us an opportunity to work on the project and for giving us all the resources which assisted the team in its completion. We are extremely thankful to him for providing his invaluable support, guidance, and supervision despite his busy schedule

A special thanks goes to all the staff members of the workshop and the Makerspace who have provided us the equipment and the tools that were critical in the fabrication of the final prototype along with helping us to 3D print all our components.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Robotics is a diverse field encompassing the creation, programming, and operation of robots. These artificial agents, whether physical or virtual, are designed to perform tasks either autonomously or semi-autonomously. Their functionalities range from simple, repetitive actions to intricate operations requiring decision-making and adaptability.

Robotics has elements such as mechanical structures, sensors, actuators, control systems, and programming play crucial roles. Mechanical structures provide the physical form, while sensors gather data about the robot's surroundings. Actuators enable movement or specific actions. Control systems, comprising software and hardware, process sensory input and execute commands. Programming dictates the robot's tasks, with varying levels of complexity.

Figure 1 Robotics

Robots can be operated remotely through teleoperation or function autonomously based on pre-programmed instructions or real-time decision-making. The application of robotics spans diverse fields, including manufacturing, healthcare, agriculture, space exploration, defence, and entertainment. Ongoing advancements in technology, artificial intelligence, and materials continue to shape the evolution of robotics, pushing the boundaries of capabilities and applications.

For the project team is determined in designing and fabricating a Humanoid robot, to perform hospitality and assistance tasks.

Humanoid Robots

A humanoid robot is a type of robot designed to resemble the human body in structure and, to some extent, in movement. These robots typically have a head, torso, two arms, and two legs, mimicking the general form of the human body. The goal of humanoid robot design is to create machines that can interact with the world and perform tasks in environments designed for humans.

Key features of humanoid robots include: -

Anthropomorphic Design: Humanoid robots are built with an appearance that closely resembles the human body. This design enables them to navigate and operate in environments designed for humans.

Sensory Perception: Humanoid robots are equipped with various sensors, such as cameras, microphones, and tactile sensors, to perceive and interact with their surroundings.

Artificial Intelligence (AI): Humanoid robots often incorporate advanced AI algorithms, allowing them to process information, learn from experiences, and adapt to changing circumstances.

Movement and Mobility: Humanoid robots are designed to move in ways similar to humans. They may have articulated joints, wheels, or a combination of both, enabling them to walk, run, and perform other human-like movements.

Human-Robot Interaction: Humanoid robots are developed to interact with humans in a socially acceptable and intuitive manner. This involves recognizing and responding to human gestures, facial expressions, and vocal cues.

Humanoid robots represent a fascinating intersection of technology and design, pushing the boundaries of what machines can achieve in mimicking human capabilities. Their potential applications continue to expand, offering possibilities for improved human-robot collaboration and the development of innovative solutions in various industries

Isaac Asimov, a prolific science fiction writer and biochemist, introduced the Three Laws of Robotics in his works as a framework for governing the behaviour of robots. Asimov first formulated these laws in his short story "Runaround," published in 1942.

The Three Laws of Robotics are:

1. A robot may not injure a human being, or through inaction, allow a human being to come to harm.
2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

   Asimov later added a "Zeroth Law" to precede the original three laws:
4. A robot may not harm humanity, or, by inaction, allow humanity to come to harm. The Zeroth Law elevates the welfare of humanity as a whole above the well-being of individual humans. It acknowledges a broader ethical responsibility for robots towards the entire human race.

1.1 STATEMENT OF WORK

To design and construct a modular Humanoid robot with cognitive abilities as a research platform that can be used as a general-purpose robot at institutions and can be customised depending on the need scenarios.

1.2 OBJECTIVES OF IRP

1. To extend the Al-based framework developed on the humanoid robot research platform to develop research capabilities in cognitive robotics.
2. Teams should concentrate on sophisticated human-robot interaction scenarios that include language acquisition, goal-directed behaviour, and verbal narrative presentation. The

actions might involve perception, such as face and speech recognition, visitor management, and social interaction including NLP.

3. The robot should be modular with subsystems having independent functionalities like HRI and Autonomous navigation.

1.3 PAST DEVELOPMENTS OF HRRP IN THE DEPARTMENT

1. AJIT HRRP

Localization and mapping utilising ORB2 SLAM were tried and successfully achieved using only a depth camera (realsense d435), mapping and localization were tested both indoors and outdoors. Ajit's prior development succeeded SLAM implementation with limited Navigation capabilities. Since the ORB point cloud is totally dependent on frame characteristics and does not identify any obstacles, attempting to extract the occupancy grid from the ORB point cloud was unsuccessful. An occupancy grid was created in a similar manner as Hector Mapping by reducing the depth camera output to a single plane and converting it to a laser

The use of the 2D occupancy Grid Pos Detection and Classification as well as Gesture Imitation was successful in supplying the module with target points to reach, and the Jetson Nano board delivered a detection-to-classification process running on ROS with very little latency. However, a full path-planning algorithm should be implemented in the module. The classifier model was trained in PyTorch using the Nano, which was also utilised to collect and produce gesture datasets and develop a customised ANN. Joint-to-Joint Imitation using depth information ought to be the logical next step in Pose Detection and Imitation.

2. MAYA

A robotics platform called Humanoid Robot MAYA 1.0 was created to support multidisciplinary research in the domains of robotics, AI, and cognitive robotics. The robot has a variety of sensors and actuators that were utilised for a variety of activities and behaviours, including SLAM, NLP, object detection and tracking, posture recognition, making motions, manipulating objects, locomotion, and others.

MAYA 1.0 had body with sufficient degrees of freedom (DOFs) and joint torque for full-body mobility (including grippers for manipulating the environment), as well as a three-wheeled base for locomotion. To concurrently accomplish low-level to high-level control, two Jetson Nanos are coupled to a ROS network and are both running Linux. Support for a wide range of sensors, including the stereo camera, RP Lidar, and Intel Realsense.

The platform was poorly designed since, although being sturdy, its base was too narrow to support a whole body. While manoeuvring, the robot frequently experienced jerks. The robot's power distribution system was not well-designed to support the robot's full operation. HRI used to do poorly when arms did, and vice versa. Moreover, the robot lacked the usage of suitable facial identification, voice to text, and text to speech AI techniques.

Current Robot Developed – Maya 3.0

Maya 3.0 the humanoid Robot has two major subsystems i.e., top torso and bottom AMR. Both can work independently and interactively. A pair of Nvidia boards (Jetson nano & Jetson Xavier) is connected to a ROS network, both running Linux so that low-level to high-level control is achieved simultaneously. Support for a wide array of sensors such as Intel Realsense, RP-Lidar, Motor Encoders, Re-speaker microphone and IMU. Modular architecture with behaviours modelled in a way to communicate with each other via ROS. Amr uses closed loop stepper motors in a differential drive setup for the robot's locomotion.

The following behaviors have been deployed onto the platform:

1. Human robot interaction
2. Facial expression
3. Person detection
4. Person follower

5. Visitor management system
6. Autonomous Navigation using SLAM algorithms
7. Human like aesthetics

All the above behaviors removes the shortcomings of the past projects.

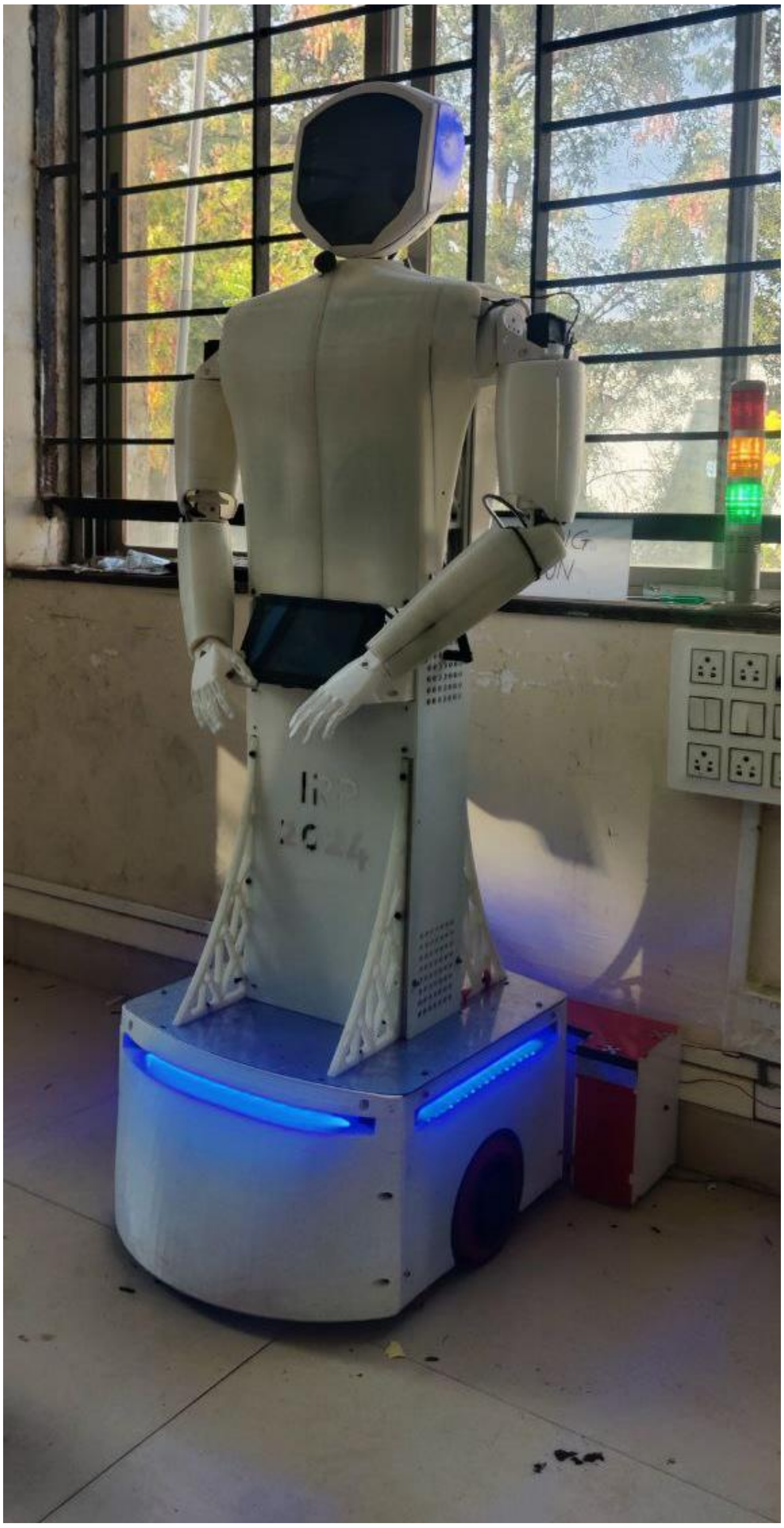

Figure 2 Maya 3.0 Humanoid Robot

CHAPTER 2

LITERATURE SURVEY

ISE PPOOA METHODOLOGY

ISE&PPOOA provides an integrated process, techniques, and technology for designing software-intensive mechatronic systems (Integrated Systems Engineering and PPOOA).

The method's ISE component covers the preliminary stages of a systems engineering process that may be applied to any form of system, not only software-intensive ones. The ISE subprocess blends traditional systems engineering best practises with MBSE. The PPOOA process element emphasises modelling concurrency as early as possible in the integrated process's software engineering phase. ISE&PPOOA presents a set of recommendations or heuristics to assist engineers in the design of a system. It simplifies architectural evaluation based on time responsiveness qualities.

The ISE&PPOOA approach for robot system level analysis allows you to monitor the allocation of different needs in the system to design decisions in the solution's architecture. The use of MBSE approaches in the development of autonomous robots, when combined with formalisms to allow capturing these design decisions and their underlying rationale, results in formal models that capture the functional relationship between the system's architecture and its mission requirements and can be exploited by the systems at runtime, allowing for new levels of self-awareness.

The ISE&PPOOA method can be thought of as a combination of three elements: mission, system, and software. Each of these has specific deliverables that are typically presented in the form of models and additional written or tabular material. The primary objective of the system aspect, as outlined in the paper, is to develop the functional and physical design of a system by identifying the subsystems and their connections. The reengineering of systems is achieved through the customization of the ISE&PPOOA process and a focus on creating functional and physical architectures for the revised system. The presentation of hierarchical arrangements of system functions and components is a key outcome of the reengineering process. This hierarchical approach is useful for the gradual improvement and release of new products.

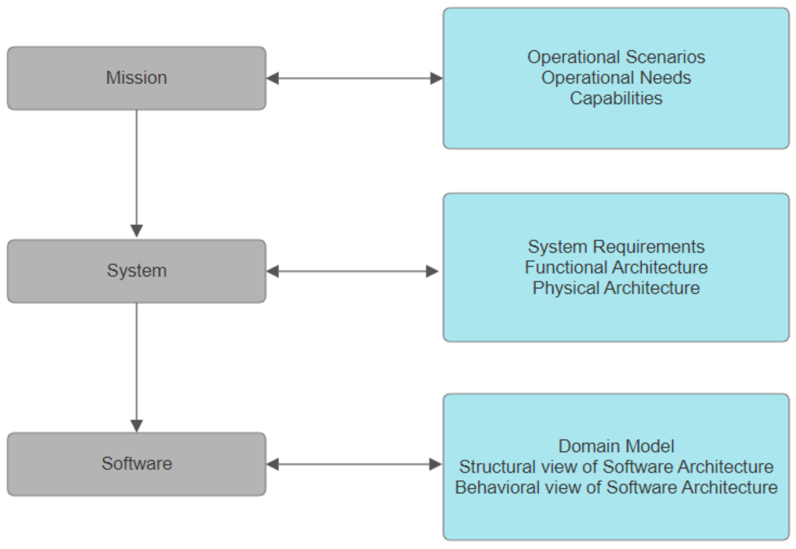

Figure 3 Flow of ISE PPOOA Methodology

The steps involved in the process are as follows:

1. Determine the system's context of use (Use cases) and outline its various modes of operation.
2. Convert the system's operational needs into a set of capabilities and high-level requirements.
3. Derive quality attributes such as efficiency, availability, and safety from the operational needs, along with any associated non-functional requirements.
4. Develop a functional architecture by breaking down the functional requirements into a hierarchy, behaviour, and interfaces.
5. Convert the functional architecture into a physical architecture for the solution. In the ISE&PPOOA process, the solution is chosen based on function grouping and design

heuristics, which are used to optimize certain quality attributes through design decisions.

The key advantages of using the ISE&PPOOA approach for robotic system control are connected to the modelling of robot capabilities and functional interfaces. This makes reuse, maintenance, and scalability easier.

SLAM

The simultaneous localization and mapping (SLAM) problem has been a major area of focus in the fields of robotics and computer vision for the past several years, especially in the context of indoor service robots. Despite the significant attention that this topic has received, there are still several challenges that need to be addressed in order to make indoor robots more widely used and practical. One of the most important of these challenges is the ability to provide real-time positioning and navigation for the robot. When addressing this problem, researchers need to focus on three key areas: accurately localizing the robot within an unknown environment, reconstructing a map of the unknown environment, and generating real-time path planning for the robot.

Accurate localization is crucial for addressing the three main challenges of real-time positioning and navigation in unknown environments. It allows the robot to determine its own location within the environment and to update this information as it moves around. There are several approaches that have been developed to achieve accurate localization, including Monte Carlo localization algorithms, which use probabilistic techniques to estimate the robot's pose based on sensory data, and simultaneous localization and mapping (SLAM), which involves building a map of the environment and using it to track the robot's position and orientation. Path planning algorithms like A* and Dijkstra can then be used to generate a safe and efficient path for the robot to follow based on this localized information.

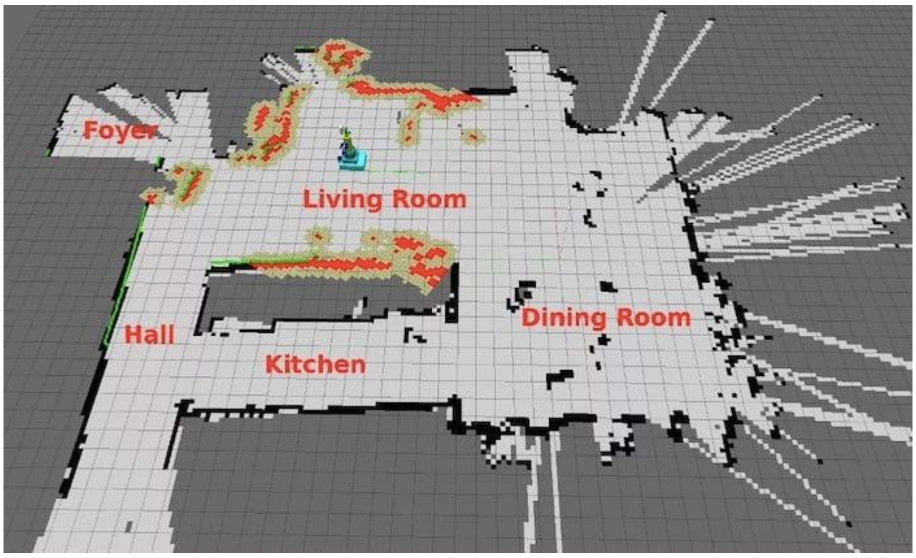

Figure 4 SLAM representation

While these methods have been successful in providing accurate localization in many cases, they also have some specific limitations that can make them less suitable for certain applications. For example, Monte Carlo localization algorithms rely on probabilistic estimates and can be computationally intensive, which may make them less suitable for real-time applications. SLAM algorithms can also be computationally demanding and may not work well in dynamic environments where the map is constantly changing. Additionally, path planning algorithms like A* and Dijkstra can be sensitive to the choice of heuristics and may not always find the shortest or most efficient path. As a result, there is still a need for further research and development in these areas to improve the performance and scalability of localization and navigation algorithms for indoor robots.

The development of hardware systems for robots has made significant progress in recent years, with a variety of processor architectures and sensor hardware modules being employed in the design of these systems. This has led to improvements in the performance of robot hardware, enabling them to handle more complex tasks and operate in more challenging environments. However, despite these advances in hardware, the reusability and portability of the software systems used by robots has often lagged behind. This has made it difficult to develop a general-purpose mobile platform that can be used for a wide range of applications.

One of the main challenges in developing software for robots is the need to support a variety of hardware platforms. Different robots may use different processor architectures, sensor configurations, and other hardware components, which can make it difficult to write software that is compatible with all of these systems. As a result, software developers often need to write custom code for each hardware platform, which can be time-consuming and can limit the reusability and portability of the software.

To address this problem, Willow Garage released the open-source Robot Operating System (ROS) in 2010. ROS is a software platform that is designed to solve software compatibility issues on different hardware platforms and to provide a foundation for the development of a universal platform for robots. By offering a standardized software interface that is compatible with a wide range of hardware systems, ROS has made it easier to develop and deploy robots for a variety of applications.

One of the key features of ROS is its modular design, which allows developers to write code for specific tasks and then reuse that code on different hardware platforms. For example, a developer could write a module for controlling a robot's arm and then use that module on multiple robots without needing to rewrite the code for each individual robot. This modular approach helps to reduce the amount of code that needs to be written and makes it easier to develop software that can be used on multiple hardware platforms.

The simultaneous localization and mapping (SLAM) technique has a long history of development, with continuous efforts to improve the efficiency and quality of the algorithms used for this purpose. The early development of SLAM can be traced back to the 1980s, when the concept of estimating spatial uncertainty was introduced at the IEEE Robotics and Automation Conference in San Francisco. This work, which was done in, became the starting point for the pre-development of the SLAM technique.

In 1991, the probabilistic approach was used to develop the first SLAM algorithm, called EKF-SLAM, based on the previous work in. This algorithm implemented the extended Kalman filter method, which has become a widely recognized approach in the field of robotics. In 2001, the use of millimetre waves (MMW) was proposed for building relative maps during the mobile robot's environment mapping process. This approach, which was introduced in, focused on the use of MMW to improve the accuracy of the mapping process.

Since the introduction of these early techniques, the field of SLAM has continued to evolve rapidly. In 2002, the FastSLAM algorithm was introduced by. This algorithm, which combines the Particle Filter and Extended Kalman Filter approaches, has become one of the most widely recognized SLAM methods, due to its high level of data accuracy. A year later, FastSLAM was modified and evolved into FastSLAM 2.0, which relied on both previous pose estimation and actual measurement of the mobile robot, as opposed to just relying on previous pose estimation in the original FastSLAM. In 2006, proposed a smoothing method called Square Root Smoothing and Mapping (SAM) to improve the efficiency of the mapping process of mobile robots. This technique used the Square Root information smoothing approach in solving SLAM problems. In 2008, introduced a new technique called UFastSLAM, which used a scale unscented transformation algorithm to improve upon the FastSLAM method presented in. In 2009, introduced a new technique called Differential Evolution, which was designed to solve SLAM problems.

ROBOT OPERATION SYSTEM (ROS)

The Robot Operating System (ROS) is an open-source operating system that is built on top of the Transmission Control Protocol/Internet Protocol (TCP/IP) network protocol. This makes it easy to communicate between different devices and systems over a network. ROS supports three main types of communication:

Figure 5 Robot Operation System (ROS) representation

1. Synchronous remote procedure calls based on process services
2. Topic-based asynchronous data flow communication
3. Data storage communication on parameter servers.

However, it should be noted that ROS is not a real-time operating system. ROS has several key features that make it a useful platform for building robot systems.

First, it uses a point-to-point design, which means that it is a multi-tasking, multi-process system rather than a single-process system. This can be more efficient in terms of hardware requirements and resource utilization, especially when multiple systems are connected over a network.

Second, ROS is nguage-agnostic, meaning it can support multiple programming languages including C++, Python, LISP, and Octave. This allows for "mixed programming," or the ability to use different languages to take advantage of their respective strengths.

Third, ROS is modular, meaning that the code for each module can be compiled separately or combined, and the process of compiling only requires the creation of corresponding CMake files.

Finally, ROS has a rich toolkit that includes visualization and debugging tools like Rviz and Rqt, which can be useful for developing and testing robot systems. Using the distributed processing capabilities of ROS, it is possible to design a high-performance mobile robot platform for indoor simultaneous localization and mapping (SLAM) and implement-autonomous navigation functions on it.

HUMAN ROBOTIC INTERACTION

The rapid advancement of robotic technology has led to the increasing use of service robots in homes, offices, and other locations for the purpose of assisting the elderly and disabled in improving their daily lives. According to the International Federation of Robotics' World Robotics 2014 Service Robots report, the global service robot market was valued at around $5.97 billion in 2014. The market for service robots is currently experiencing rapid growth worldwide. Receptionist robots are a specific type of service robot that are designed for use in office environments. These robots must be intelligent, safe, and effective in providing services to people.

Human-robot interaction is a crucial aspect of service robots, as it allows people to communicate with the robot in a natural manner. There are various methods of human-robot interaction that have been proposed, such as somatosensory interaction, voice interaction, and face recognition. The ability to localize oneself and navigate to a destination within the working environment is a vital capability for truly intelligent service robots. Gesture recognition has numerous applications, including Human Machine Interaction (HMI), Human Robot Interaction (HRI), and Social Assistive Robotics (SAR). Speech recognition has also been widely used for communication with various devices.

As a receptionist and partner, the robot must be able to perform tasks such as autonomous navigation, responding to voice commands, recognizing gestures, and following guests. From a cognitive perspective, the interaction robot should be able to detect the presence of humans and be aware when a person wants to interact with the robot.

HRI is concerned with the ways in which humans and robots interact and communicate with one another, and it involves the development of technologies and techniques that enable robots to perceive and understand the intentions and actions of humans.

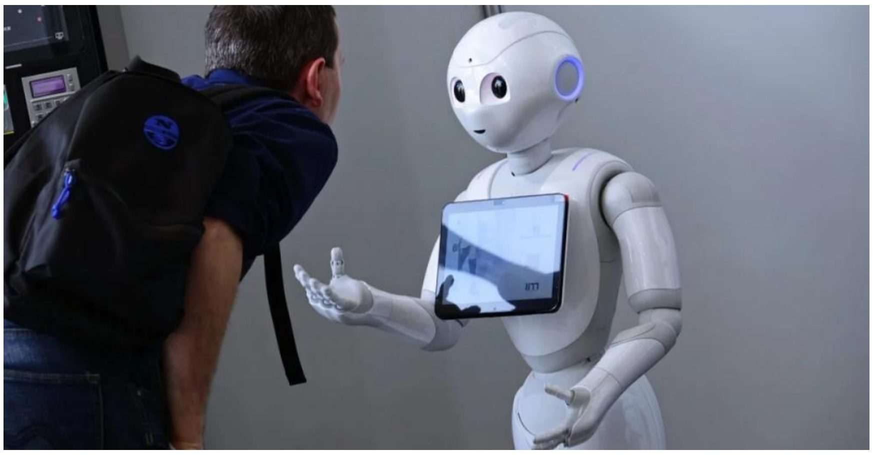

Figure 6 Human Robot Interaction representation

The paper "Human-Robot Interaction: A Survey" by Michael A Goodrich and Alan C. Schultz provides an overview of the field of human-robot interaction (HRI). The paper begins by discussing the history of HRI and the various challenges and opportunities that it presents. It then goes on to describe the different approaches that have been taken to enable effective HRI, including the use of sensors, machine learning algorithms, and interactive technologies such as natural language processing and touchscreens.

The paper also discusses the various applications of HRI, including manufacturing, healthcare, agriculture, transportation, and construction. It describes the benefits of HRI, such as increased efficiency and productivity, as well as the ethical and social issues that need to be considered when developing HRI systems.

There have been many landmark technologies developed for human-robot interaction (HRI) over the years. Some examples include:

- Natural language processing: Natural language processing (NLP) is a field of artificial intelligence (AI) that enables computers to understand and process human language. NLP has been widely used in HRI to enable robots to understand and respond to human communication, such as by recognizing and interpreting spoken or written language.
- Gesture recognition: Gesture recognition is a technology that enables computers to recognize and interpret human gestures, such as hand movements or facial expressions. This technology has been widely used in HRI to enable robots to respond to human gestures and movements, and to facilitate natural and intuitive interaction with humans.
- Facial expression recognition: Facial expression recognition is a technology that enables computers to recognize and interpret human facial expressions, such as smiles or frowns. This technology has been widely used in HRI to enable robots to respond to human emotions and to facilitate natural and intuitive interaction with humans.
- Touch-based interfaces: Touch-based interfaces are interfaces that enable humans to interact with computers or robots through touch or physical contact. This technology has been widely used in HRI to enable humans to interact with robots in a natural and intuitive way, such as by touching or manipulating objects.

MODULAR HUMANOID ROBOTS

Modular humanoid robots are robots that are composed of multiple modular components or modules, which can be easily assembled and disassembled. These modules can be customized and replaced according to the needs of the application, allowing the robot to be easily reconfigured and adapted to perform different tasks.

One of the main advantages of modular humanoid robots is their flexibility and adaptability. By using modular components, these robots can be easily reconfigured to perform different tasks or to operate in different environments. This makes them well-suited for use in a variety of applications, including manufacturing, inspection, and search and rescue.

Another advantage of modular humanoid robots is their ease of maintenance and repair. If a component fails or wears out, it can be easily replaced with a new module, reducing downtime and maintenance costs.

There are several challenges that researchers and engineers face when developing modular humanoid robots. Some of these challenges include:

- Integration of modular components: One of the main challenges of modular humanoid robots is integrating the various modular components into a cohesive system. This requires designing the modules to work together seamlessly and developing algorithms and control systems that can coordinate the movements of the different modules.
- Robustness and reliability: Another challenge is ensuring that the modular humanoid robot is robust and reliable, with minimal downtime and maintenance requirements. This requires designing the modules to be durable and resistant to wear and tear, and developing effective strategies for maintenance and repair.
- Human-robot interaction: Modular humanoid robots are often designed to interact with humans in a variety of settings, such as manufacturing or search and rescue. This requires designing the robot to be safe and easy to use, and developing algorithms and control systems that can enable the robot to understand and respond to human gestures and movements.
- Efficient power management: Modular humanoid robots typically have complex power requirements, as they often have multiple motors and sensors that need to be powered. This requires developing efficient power management strategies to ensure that the robot can operate for an extended period of time without running out of power.

There are several strategies that can be used to address the challenges of modularity in humanoid robots. Some strategies include:

- Use of standardized interfaces: Standardized interfaces can help to ensure that the modular components of the humanoid robot are compatible with each other, making it easier to assemble and disassemble the robot.
- Use of modular control systems: Modular control systems can be used to coordinate the movements of the various modular components of the humanoid robot. This can help to ensure that the robot is able to perform a variety of tasks and activities in a smooth and coordinated manner.
- Use of self-contained modules: Self-contained modules can help to reduce the complexity of the humanoid robot, as each module can be designed to perform a specific task or function without requiring a complex control system.
- Use of modular power systems: Modular power systems can be used to provide power to the various modular components of the humanoid robot. This can help to ensure that the robot is able to operate for an extended period of time without running out of power.

There are several popular modular humanoid robots that have been developed and used in a variety of settings, including manufacturing, inspection, and search and rescue. Some examples of popular modular humanoid robots include:

- Modular Robotic Vehicle (MRV): Developed by the Massachusetts Institute of Technology (MIT), the MRV is a modular humanoid robot that is designed for use in a variety of settings, including manufacturing, inspection, and search and rescue. The MRV is composed of modular components, including a torso, arms, legs, and head, which can be easily assembled and disassembled. The MRV is also equipped with a variety of sensors and cameras, which allow it to navigate and interact with its environment.

- Modular Mobile Manipulator (M3): Developed by the University of Cambridge, the M3 is a modular humanoid robot that is designed for use in manufacturing and inspection tasks. The M3 is composed of modular components, including a torso, arms, and legs, which can be easily assembled and disassembled. The M3 is also equipped with a variety of sensors and cameras, which allow it to navigate and interact with its environment.
- Modular Robotic Exoskeleton (MRE): Developed by the University of Tokyo, the MRE is a modular humanoid robot that is designed for use in search and rescue operations. The MRE is composed of modular components, including a torso, arms, and legs, which can be easily assembled and disassembled. The MRE is also equipped with a variety of sensors and cameras, which allow it to navigate and interact with its environment.

CHAPTER 3

ISE & PPOOA PROCESS & SYSTEM ARCHITECTURES

INTRODUCTION

A procedure, approach, and tool for systems engineering in software-intensive mechatronic systems is called Integrated Systems Engineering and PPOOA (ISE&PPOOA). The earliest stages of a systems engineering process, which may be applied to any type of system and is not limited to software-intensive systems, are covered by the ISE portion of the process. It blends MBSE with conventional systems engineering best practises (Model-Based Systems Engineering). The PPOOA section of the process places a focus on modelling concurrency as early as feasible in the integrated process's software engineering phase.

PPOOA is also an architecture framework for real-time software. The integration between the systems engineering sub process and the PPOOA software engineering sub process is achieved through a responsibility-driven software analysis approach supported by CRC (Class Responsibility Collaborator) cards, a technique proposed by Beck and Cuningham at OOPSLA'89

ISE&PPOOA provides guidelines or heuristics to assist engineers in the architecting of a system. One of the project deliverables is the functional architecture, which represents the functional hierarchy using a SysML block definition diagram. This diagram is supplemented with activity diagrams for the main system functional flows. The N-square chart is used as an interface diagram in the form of a matrix where the main functional interfaces are identified. A textual description of the system functions is also included as part of the deliverable. Another deliverable is the physical architecture, which represents the system decomposition into subsystems and parts using a SysML block definition diagram. This diagram is complemented with SysML internal block diagrams for each subsystem and activity and state diagrams as needed. A textual description of the system parts is also provided. The heuristics used for the specific architecture solution are identified and documented.

The architecture of a software subsystem can be described using the PPOOA method, which utilizes two views and one or more diagrams in UML notation. The first view is the static or structural view, which shows the system components and their relationships in terms of composition and usage. The second view is the dynamic or behavioral view, which illustrates the flow of actions taken by the system in response to events, using UML/SysML activity diagrams. Coordination mechanisms that act as connectors are also represented. The PPOOA method allows for the evaluation of the system architecture based on time responsiveness characteristics, and the latest version, ISE&PPOOA/Energy, is specifically designed to address energy efficiency in industrial facilities. This MBSE method supports bidirectional functional allocation between components and system responses, which can be modelled using activity diagrams and aided by tools. Several published papers in the resources section provide examples of ISE&PPOOA/Energy's application in addressing energy efficiency issues in industrial facilities.

3.1 OPERATIONAL SCENARIOS AND NEEDS

Operational scenarios can be defined as envisioned sequence of events that involves the product or service's interaction with its surroundings and consumers, as well as interaction among its product or service components. The situations in which our robot will operate after it is developed are called operational scenarios. We make assumptions about working scenarios and how the robot will behave in them in order to identify which features to include in our robot and satisfy client expectations. In this instance, we begin with the deployment process and then discuss the interaction systems, task performance, and shutdown scenarios. The below table describes about the main high-level scenarios in which our robot will operate.

Table 1 Operational scenarios for Humanoid Robot – Maya 3.0

Deployment or Adaption Phase	
S1 System configuration and calibration	The operator performs the system installation and calibrates the robot to the best suited settings according to its environment.
Operation Phase	
S2 System start	All subsystems powered up. All sensors calibrated, and actuators initialized
S3 Command action	The robot receives input command from users to perform the tasks through input devices.
S4 Perform Tasks	The robot should SENSE, PLAN, and ACT after receiving instructions from the user to do a certain task. activities such as moving around, following someone, conversing, etc.
S5 Generate response	The robot must tell users of the system's status, task completion, and what to do next after completing the tasks that have been assigned to it. Etc.
S6 Manage functionality data	Whenever any new data needs to be updated the robot must provide prompts, interfaces to update the data. The data can be anything related to navigation.

S7 Charge system	A charging mode is initiated by the operator. The system is charged when the charger is switched on, and the robot has to be attached to a power source.
S8 Shut Down	The operator can shut off the system via the user interface. As a result, the robot starts the shut-down routine, which ends all processes and deactivates actuators and sensors.
S9 Emergency Stop	Upon an anomalous behaviour, the emergency stop of the robotic system is activated, stopping any current motion, and this is notified to the operator

DESCRIPTION OF THE OPERATIONAL SCENARIO

The team incorporate some of the components in order to characterise the operational scenarios much more effectively. The preconditions, triggering event, description, and post event provide considerably more information about the robot's functions in that specific circumstance.

Individual table describes precisely about the individual system scenarios.

Table 2 Description of Operational scenarios

S1 System configuration and calibration	
Preconditions	Safe transport and export of the product.
Triggering event	Unpacking the product
Description	The test support engineer sets up and installs the system in the desired environment.

Postconditions	The test support engineer sets up installs the system in the desired environment.

S2 Start system	
Preconditions	Battery must be connected to the power distribution system.
Triggering event	Initiate operation
Description	The operator powers up all subsystems. The robot calibrates its sensors. The base actuators are initialized, connection with database established, UI functioning as per expectations.
Postconditions	Calibrated Actuators, sensors and UI.

S3 Command action	
Preconditions	Powered up system with calibrated sensors, initiated UI. Robot in stationary state.
Triggering event	User Input
Description	The robot collects the input from the user through user interface.
Postconditions	Command received

S4 Perform Tasks	
Preconditions	Powered up system calibrated actuators and sensors.

Triggering event	Command received from user through user input.
Description	The robot starts moving in case of actuation task, robot starts detection etc.
Postconditions	Tasks are performed.

S5 Generate response	
Preconditions	Completed tasks
Triggering event	Task completion, user input
Description	The robot takes in data from the user, or from the task completion. In order to notify user, it generates responses.
Postconditions	Data acquired with necessary information displayed on the screen.

S6 Manage functionality data	
Preconditions	Data submitted by the user
Triggering event	User input
Description	Every time the data is acquired from the user, it is stored in the database as the data can be anything, from crucial parameters to person's details and facial features.
Postconditions	Data updated

S7 Charge system	
Preconditions	System should be in stationary mode near charger.
Triggering event	User interface
Description	The battery connected to the charger through power distribution system. The charging mode is actuated through UI.
Postconditions	Battery charging is displayed in UI and charger.

S8 Shut Down	
Preconditions	System working in active mode
Triggering event	User input
Description	Shutdown process actuated to stop all the actuators and shutdown the system.
Postconditions	System stops

S9 Emergency Stop	
Preconditions	System working in active mode
Triggering event	Anomalous behaviour
Description	Stop any current motion, and notify the operator
Postconditions	System stops

OPERATIONAL NEEDS BASED ON SCENARIOS

Needs are critical components of any system that supports in the accomplishment of desired objectives. The components necessary for. a system to function in specific situation are referred to as operational needs. As individuals have some requirements, robots must have some in order to function in a specific context and scenario. A robot's demands might range from electrical to computational to mechanical. As a result, identifying such requirements is critical when developing a system/ frobot. The requirements for our system to perform in an individual situation are listed below.

<u>System start</u>

ON2 1: The System shall be able to run under the battery capacity of 24V, 25A and power

of 600kW.

ON2 1.2: The System shall be able to run for minimum 30 min and maximum of 2hrs.

ON2 2: The System shall be able to calibrate the motors and sensors

ON2 3: The System shall initiate all start-up applications and software.

ON2_4: The System shall be able to handle modifications in future.

<u>Command action</u>

ON3_1: System shall have a microphone in order to receive voice commands.

ON3_2: System shall have a touch display in order to receive touch commands through GUI

ON3_3: System shall be well equipped with Speech to text and text to speech algorithms to

interpret voice-based commands

ON3_4: System shall be well equipped with GUl for easy user interactions

Perform Tasks

ON4_1: The systems actuators must be calibrated when system starts.

ON4_2: The system shall perform the necessary actuations through actuators and detection related actions through sensors

ON4_ 3: System shall be well equipped with actuators with enough torque to perform

ON4_4: System shall be well equipped with camera, lidar, microphone of suitable etc

Generate Response

ON5_1: The System shall be equipped with display that displays output through Ul.

ON5_2: The system shall be equipped with speakers Go generate voice-based response.

Manage Functional Data

ON6_1: When a user enters the data, the database is updated with that data as well as other important parameters.

Charge System

ON7_1: The system shall be equipped with a independent charging port.

ON7_ 2: The system shall be equipped with the functionality of notifying charging status.

Shutdown

ON8_1: The system shall be equipped with shutdown Ul interface.

ON8_2: The System shall initiate shutdown protocol when turned off.

Emergency Stop

ON9_1: System shall have emergency stop button, so that whenever operator can stop the Actuators based on anomalous behaviour.

3.2 CAPABILITIES

The ability to carry out a given course of action or achieve specified results is referred to as a capability. Capability in human capital refers to the ability to execute or achieve certain actions/outcomes at the intersection of capacity and ability. Humanoid robot capabilities should always result in cost savings and support to people in some form. Humanoids often possess skills such as copying human movements, recognising things in order to extract characteristics from the items and their surroundings, conversing with humans in common human comprehensible language, and many more. Based on the demands and requirements of the scenarios, we have developed the following capabilities in our robots, which are listed in the table below. Team decided and designed the behaviours/functions after identifying these capabilities.

Table 3 Capabilities

C1. Easy human robot interaction	The robot must be able to communicate with people in order to do the tasks
C1.1 Voice based interaction	The robot needs to be able to communicate through voice instructions
C1.2 Touch based interaction	The robot should be able to interact communicate through display
C2. Perform tasks autonomously	The robot should be able to complete tasks without human interventions
C2.1 Identify person	The robot should be able to identify the individual by recognizing facial traits
C2.2 Follow person	The robot should be able to orient itself towards person and follow him at a safe distance.

C2.3 Mapping	The robot should be able to familiarize its environment in the form of a map.
C2.4 Navigation	The robot should be able to move to any place in the familiarized environment (map).
C2.5 Avoid obstacles	The robot should be able to avoid obstacles and go across them.
C2.6 Manage visitors	The robot should be able to store, retrieve and update the visitor's data.
C3. Quick configuration of new data	The robot should allow easy configuration of any data that is responsible for working of other functionalities.
C3.1 Manage person data	The robot should take less time to store, retrieve and update visitors/person data.
C3.2 Manage navigation data	The robot should have easy interface to store, retrieve and update visitors/person data.
C4. Harmless	The robot must be secure form its surroundings and safe for users to operate.
C4.1 Navigate safely	The robot should move freely in the map avoiding collisions and accident.
C4.2 Follow person at safe distance	The robot should follow person at a minimum safe distance, avoiding collisions and accident with users.
C4.3 Move around obstacles safely	The robot should make its way across obstacles, avoiding collisions and accident with obstacle.

C5. Resilience contingent	The robot should be able to avoid small or minor irregularities in the command or features that are required to perform task.

IDENTIFYING FUNCTIONS

Functions in a system are processes, activities, and operations that must be completed in order to use the system's resources and produce the required outcome. Each particular function of a robot is in charge of producing certain actuations or reactions. A succession of functions results in the operation of a subsystem. Most subsystems in a modular robot do not rely on the outputs of other subsystems as input. To build a modular system, functions should be efficiently structured to work on the resources and inputs available to them rather than incorporating the output of other functions as part of the inputs. Such functions are tough to recognise in a system/robot. Most fundamental functions integrate and result in independent subsystems, although there may be a few independent functions that have their own sub functions to become independent. Below are the fundamental functions framed by analysing the capabilities of the robot.

Table 4 Functionalities

Functionalities	Related Capabilities
F1. Manage the operate interface	• C1.1 Voice based interaction • C1.2 Touch based interaction • C2.6 Manage visitors • C3.1 Manage person • C3.2 Manage navigation data
F2. Classify intent of operator	• C1.1 Voice based interaction • C2.6 Manage visitors

F3. Detect features	• C2.1 Identify features • C2.2 Follow person • C2.3 Mapping • C2.5 Avoid obstacles • C2.6 Manage visitors
F4. Manage visitor’s data	• C2.1 Identify features • C2.6 Manage visitors • C3.1 Manage person data
F5. Manage Map data	• C2.3 Mapping • C2.4 Navigation • C3.2 Navigation data
F6. Calibrate robot pose	• C2.2 Follow person • C2.4 Navigation • C2.5 Avoid obstacles
F7. Map environment	• C2.3 Mapping • C2.5 Avoid obstacles • C4.3 Move around obstacles safely
F8. Move robot safely	• C2.5 Avoid obstacles • C4.1 Follow person at safe distance • C4.2 Follow person in safe distance • C4.3 Move around obstacle safely
F9. Plan the route	• C2.4 Navigation • C2.5 Avoid obstacles • C4.1 Navigate safely • Move around obstacles safely

F10. Manage errors	• C5. Resilience contingent

3.3 FUNCTIONAL ARCHITECTURE

CONTEXT DIAGRAM

A context diagram is a powerful tool that is used to provide a high-level view of a system and its relationship to other external systems or entities. It is a visual representation that shows how the system interacts with its environment and is typically created at the beginning of the systems development life cycle. The context diagram is used to understand the overall structure of a system and the interactions that occur between the system and its environment.

The diagram usually consists of a single box, known as a "system box," which represents the system of interest. Surrounding the system box are other boxes representing external entities such as users, other systems, and external data sources. These boxes are connected to the system box with arrows, which indicate the flow of data or control between the various entities.

The arrows are used to show the direction and nature of the interactions between the system and its environment. For example, an arrow pointing from an external entity to the system box might indicate that data is being sent to the system, while an arrow pointing from the system box to an external entity might indicate that the system is sending data to the external entity. Additionally, the diagram may also use different shapes, colors, or labels to represent different types of interactions or flows, such as input and output, control, or data flows.

The context diagram is a high-level representation of the system, it does not provide information about the internal structure of the system, but it does provide insight into how the system interacts with the external entities and how it fits within the overall systems architecture. By understanding the context of the system, it can be easier to identify potential issues and design solutions.

Here's a context diagram of robot showing the various sub-systems of robot and their interaction with the environment.

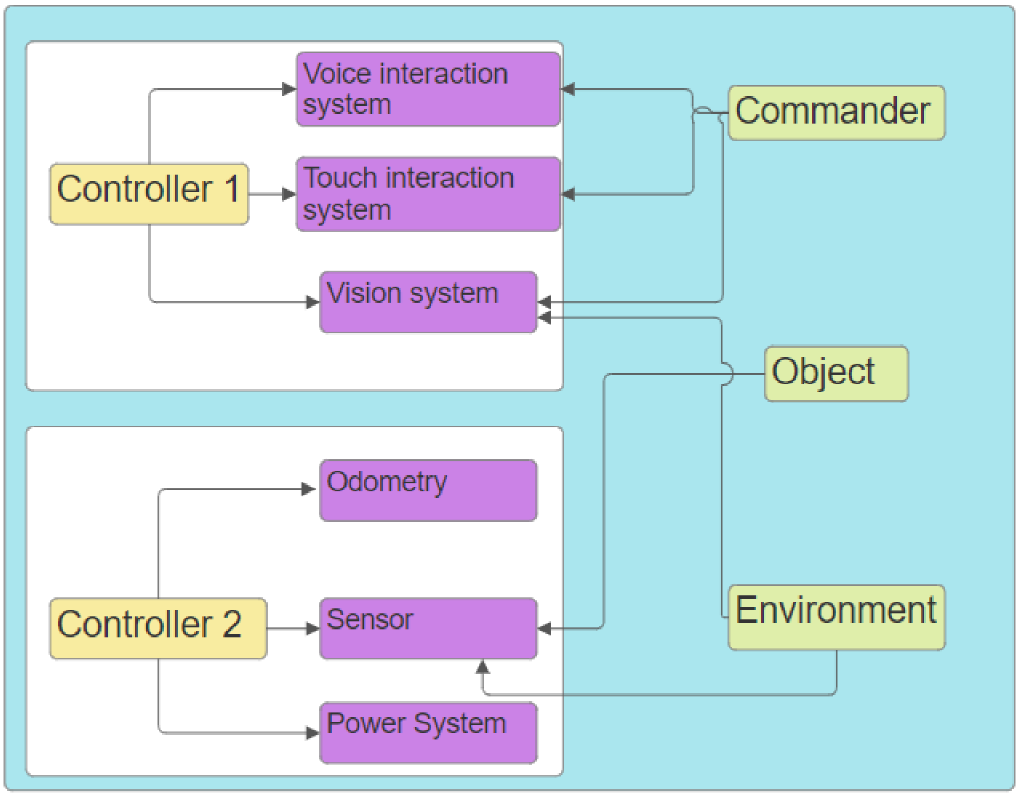

Figure 7 Context Diagram

FUNCTIONAL HIERARCHY

A functional hierarchy is a way of organizing the functions and sub-functions of a system or organization into a structured hierarchy. The term "functional hierarchy" is often used in the context of systems engineering and system design, but it can also be applied to other fields such as organizational management.

The top level of the hierarchy represents the overall function or goal of the system, and each level below it represents a more specific sub-function or sub-goal. Functions at a higher level in the hierarchy provide the context or purpose for the functions at the lower levels. The hierarchy is often represented graphically using a pyramid or tree-like diagram, with the top level at the peak and the lower levels branching out below it.

For example, in a system design, the top-level function might be "Control the temperature of a room" and sub-functions would be like "Measure the room temperature", "Determine desired temperature", "Activate heating/cooling system" and "Monitor the system" these functions arrange in a logical order, and each one of them is required for the top-level function to be achieved.

The functional hierarchy can be a powerful tool for understanding the relationships between the different functions and sub-functions of a system, and it can help identify the logical dependencies between them. Additionally, it can provide a structured approach to designing, testing, and maintaining a system, by breaking it down into manageable parts, and allow the identification of interfaces, interactions, and overall system's architecture.

Here's a figure the functional Hierarchy of Maya 3.0 showing the different functionalities of two subsystem.

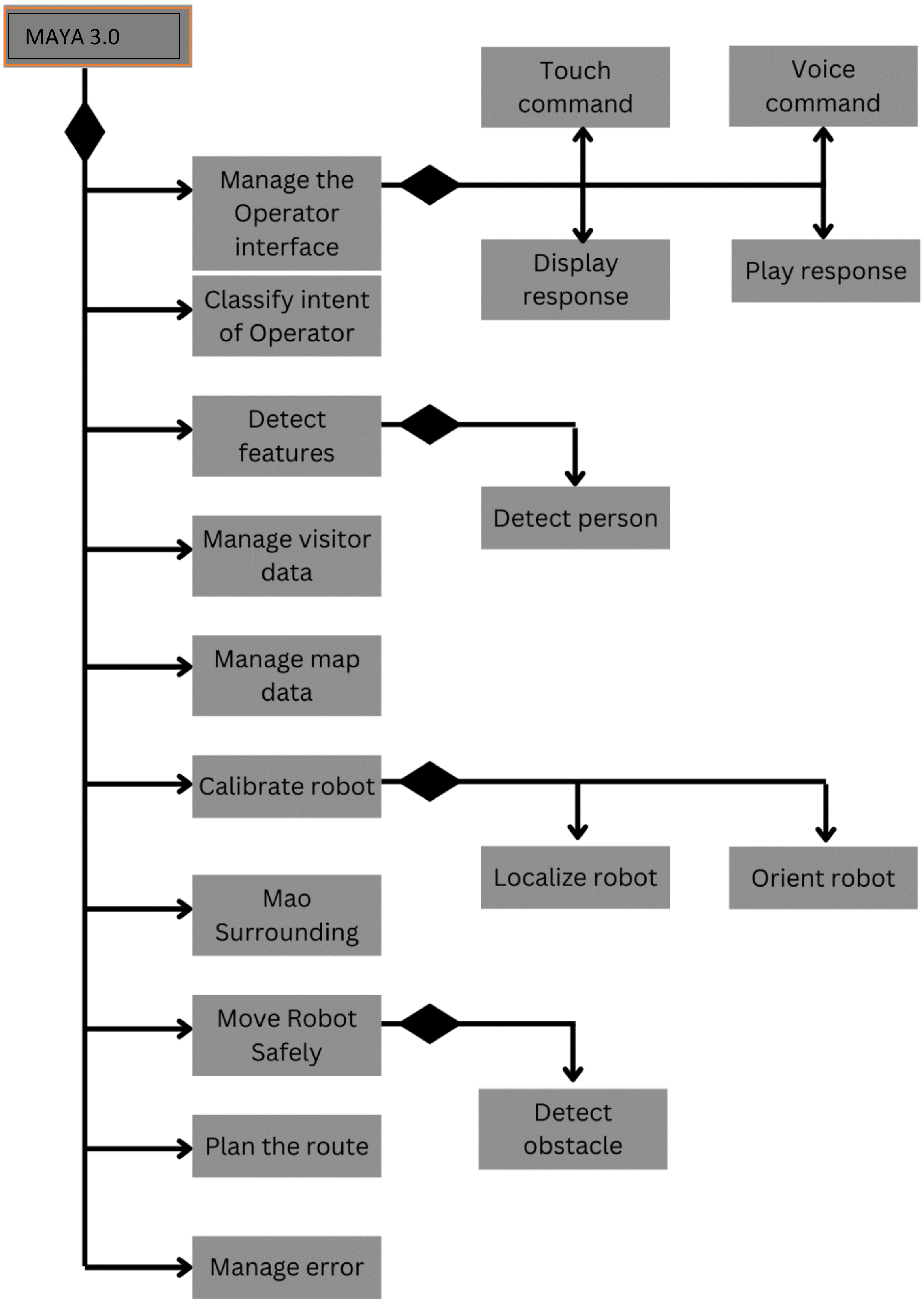

Figure 8 Functional Hierarchy

3.4 PHYSICAL ARCHITECTURE

3.4.1 MODULAR ARCHITECTURE

Modular architecture is a design approach that involves breaking a system or product into smaller, independent, and interchangeable modules or components. These modules are designed to have well-defined interfaces and can be connected to form the overall system. Modularity allows for more flexibility, maintainability, and ease of modification in the design, development, and maintenance of the system.

One of the key principles of modular architecture is that each module should have a specific, well-defined function and should be designed to be as independent as possible from the other modules. This allows for easy replacement of individual modules without affecting the functionality of the overall system. It also enables the reuse of modules in other systems, and make it easier to test and debug individual components.

Modular architecture is widely used in a variety of fields, including software development, electronics, and mechanical engineering. In software development, modular architecture is used to break down a large and complex software system into smaller, more manageable parts, each with a specific function. This makes it easier to understand and maintain the code, and it facilitates collaboration among developers working on different parts of the system.

In electronics, modular architecture is used to design electronic devices such as smartphones, computers, and TVs. By breaking down the system into smaller, independent modules, manufacturers can easily replace or upgrade individual components without having to replace the entire device.

In mechanical engineering, modular architecture can be used to design complex mechanical systems such as vehicles, machinery, and buildings. The system can be broken down into smaller, independent components that can be easily replaced or upgraded without affecting the functionality of the overall system.

In short, modular architecture is a useful approach that can make systems more flexible, maintainable, and cost-effective, by breaking them down into smaller, independent components.

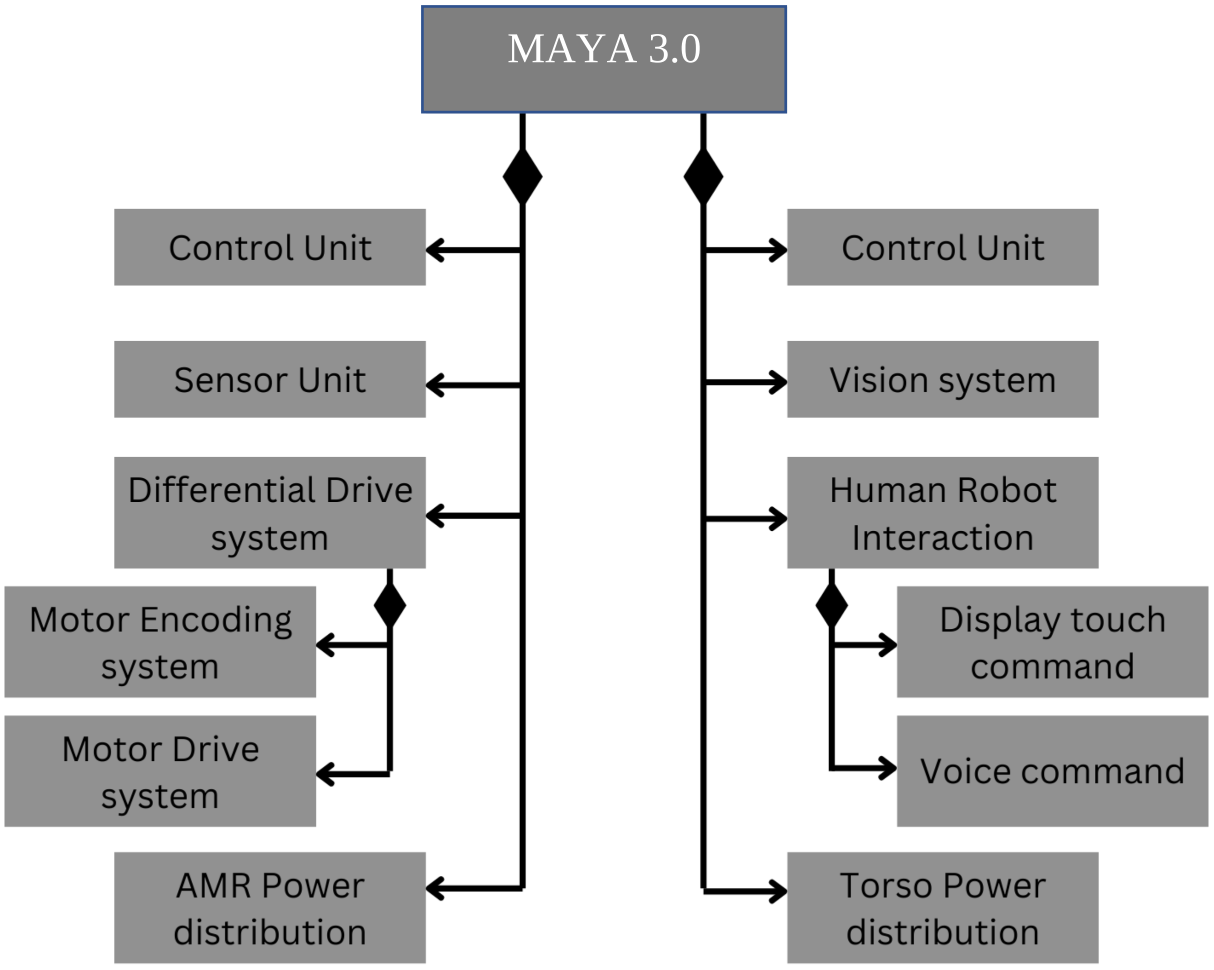

Figure 9 Modular Architecture

3.5 SOFTWARE DOMAIN MODEL

A software domain model is a representation of a specific subject area or "domain" in the form of a model, usually created using object-oriented techniques. It is used to capture the key concepts and relationships within a particular domain and represents the real-world entities, their attributes, and the relationships among them.

The main purpose of a software domain model is to provide a clear and abstract understanding of the problem domain and to capture the essential elements and relationships of the domain. This model is used to communicate the system's requirements and constraints, to guide the design of the system, and to ensure that the system's functionality and behaviour aligns with the problem domain.

A domain model typically consists of classes, interfaces, and objects, which represent the real-world entities and their relationships. These classes and objects encapsulate both the data and the behaviour that is specific to the domain, making them a valuable tool for creating a shared understanding among the stakeholders of a system.

The process of creating a software domain model involves a deep understanding of the problem domain, gathering and analysing data, identifying key concepts and entities, and creating relationships among them. Also, it may involve the use of different modelling techniques, such as use cases, class diagrams, entity-relationship diagrams, and state diagrams, among others.

A domain model is an important part of the software development process, it helps to identify the requirements, constraints and guides the design of the system, and it is a valuable tool for communication and collaboration among the stakeholders, developers, and users of the system. It is also used as a foundation for other modelling techniques, such as architectural,

data, and process models.

TORSO

Here’s The Software domain model of the Torso containing key components for each process and the software methods that work on the data sent by components.

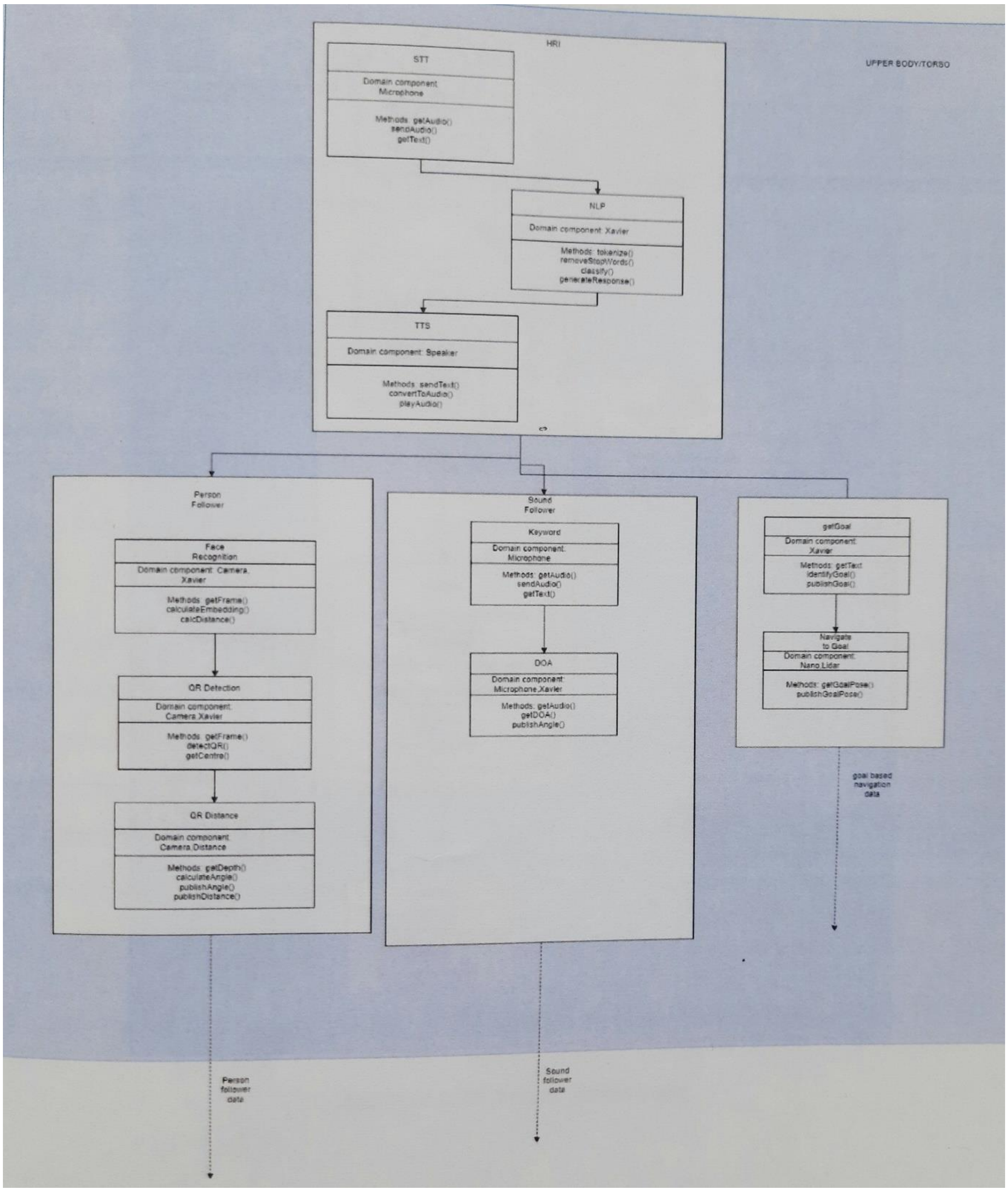
HRI
UPPER BODY/TORSO
STT
Domain component Microphone
Methods getAudio() sendAudio() getText()
NLP
Domain component Xavier
Methods tokenize() removeStopWords() classify() generateResponse()
TTS
Domain component: Speaker
Methods sendText() convertToAudio() playAudio()
Person Follower
Face Recognition
Domain component: Camera, Xavier
Methods getFrame() calculateEmbedding() calcDistance()
QR Detection
Domain component: Camera,Xavier
Methods getFrame() detectQR() getCentre()
QR Distance
Domain component: Camera,Distance
Methods getDepth() calculateAngle() publishAngle() publishDistance()
Sound Follower
Keyword
Domain component: Microphone
Methods: getAudio() sendAudio() getText()
DOA
Domain component: Microphone,Xavier
Methods: getAudio() getDOA() publishAngle()
getGoal
Domain component: Xavier
Methods: getText identifyGoal() publishGoal()
Navigate to Goal
Domain component: Nano Lidar
Methods: getGoalPose() publishGoalPose()
goal based navigation data
Person follower data
Sound follower data

Figure 10 Torso Software domain model

AMR

AMR Software domain model showing how the data from Torso can be used for actuation without disrupting normal functioning of base.

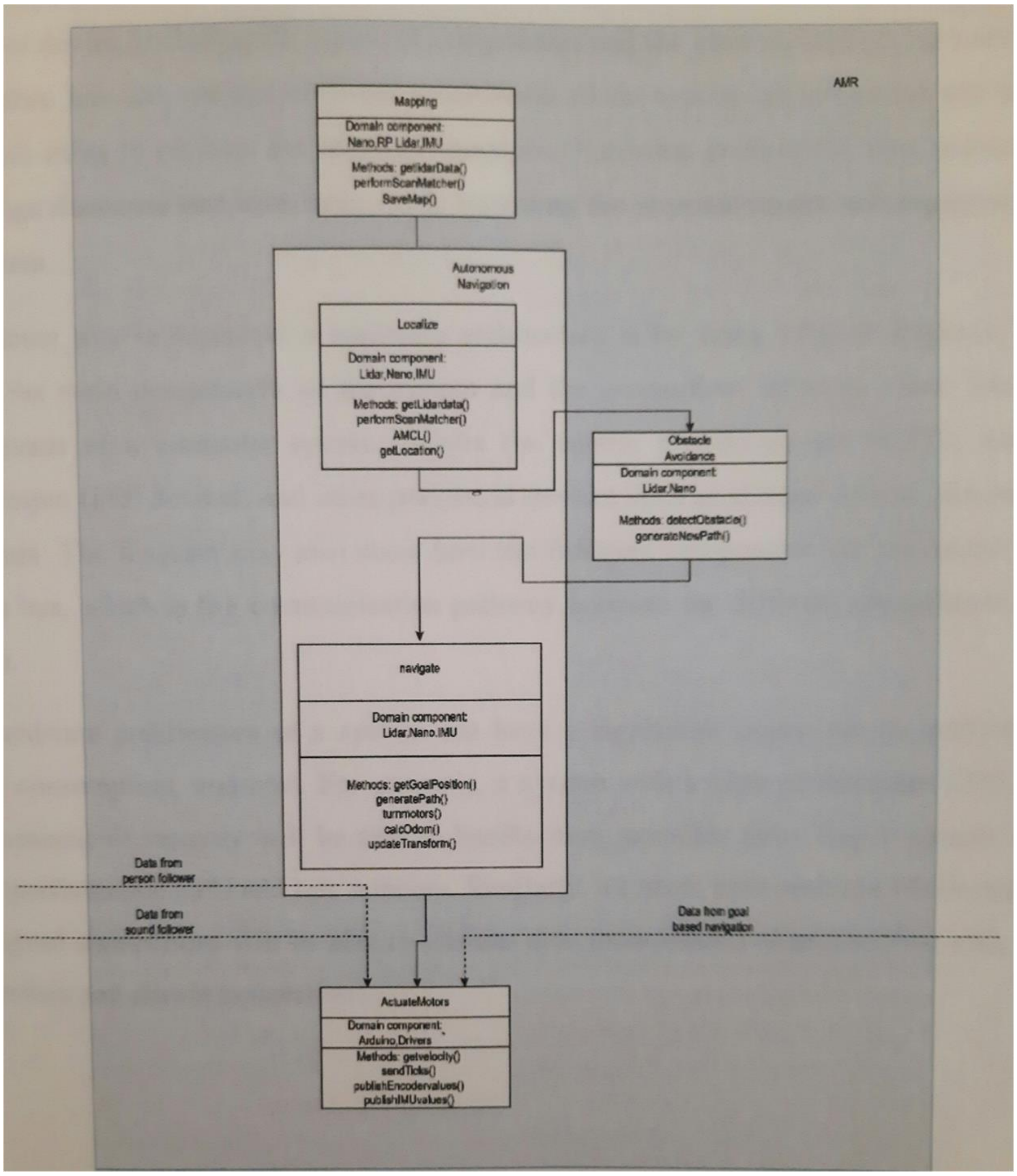
AMR
Mapping
Domain component: Nano,RP Lidar,IMU
Methods: getlidarData() performScanMatcher() SaveMap()
Autonomous Navigation
Localize
Domain component: Lidar,Nano,IMU
Methods: getLidardata() performScanMatcher() AMCL() getLocation()
Obstacle Avoidance
Domain component: Lidar,Nano
Methods: detectObstacle() generateNewPath()
navigate
Domain component: Lidar,Nano,IMU
Methods: getGoalPosition() generatePath() turnmotors() calcOdom() updateTransform()
Data from person follower
Data from sound follower
Data from goal based navigation
ActuateMotors
Domain component: Arduino,Drivers
Methods: getvelocity() sendTicks() publishEncodervalues() publishIMUvalues()

Figure 11 AMR Software domain model

3.6 HARDWARE ARCHITECTURE

Hardware architecture refers to the overall physical structure and organization of a computer system or device, including the layout of components and the interconnections between them. It describes how the various hardware components of the system are connected and interact with each other to perform the required functions. Hardware architecture also encompasses the design decisions that have been made regarding the physical layout and organization of the system.

A common way to represent a hardware architecture is by using a block diagram, which shows the main components of the system and the connections between them. The main components of a computer system include the central processing unit (CPU), memory, input/output (I/O) devices, and other peripheral devices such as storage devices and network interfaces. The diagram may also show how the different components are connected to the system bus, which is the communication pathway between the different components of the system.

The hardware architecture of a system has a significant impact on its performance, power consumption, and cost. For example, a system with a high-performance CPU and a large amount of memory will be able to handle more complex tasks than a system with a lower-performance CPU and less memory. Similarly, a system with multiple I/O devices and high-speed connections will be able to transfer data more quickly than a system with fewer I/O devices and slower connections. Here's the Hardware Architecture model for Torso and AMR containing various components used and how they are interconnected for transfer of power and data.

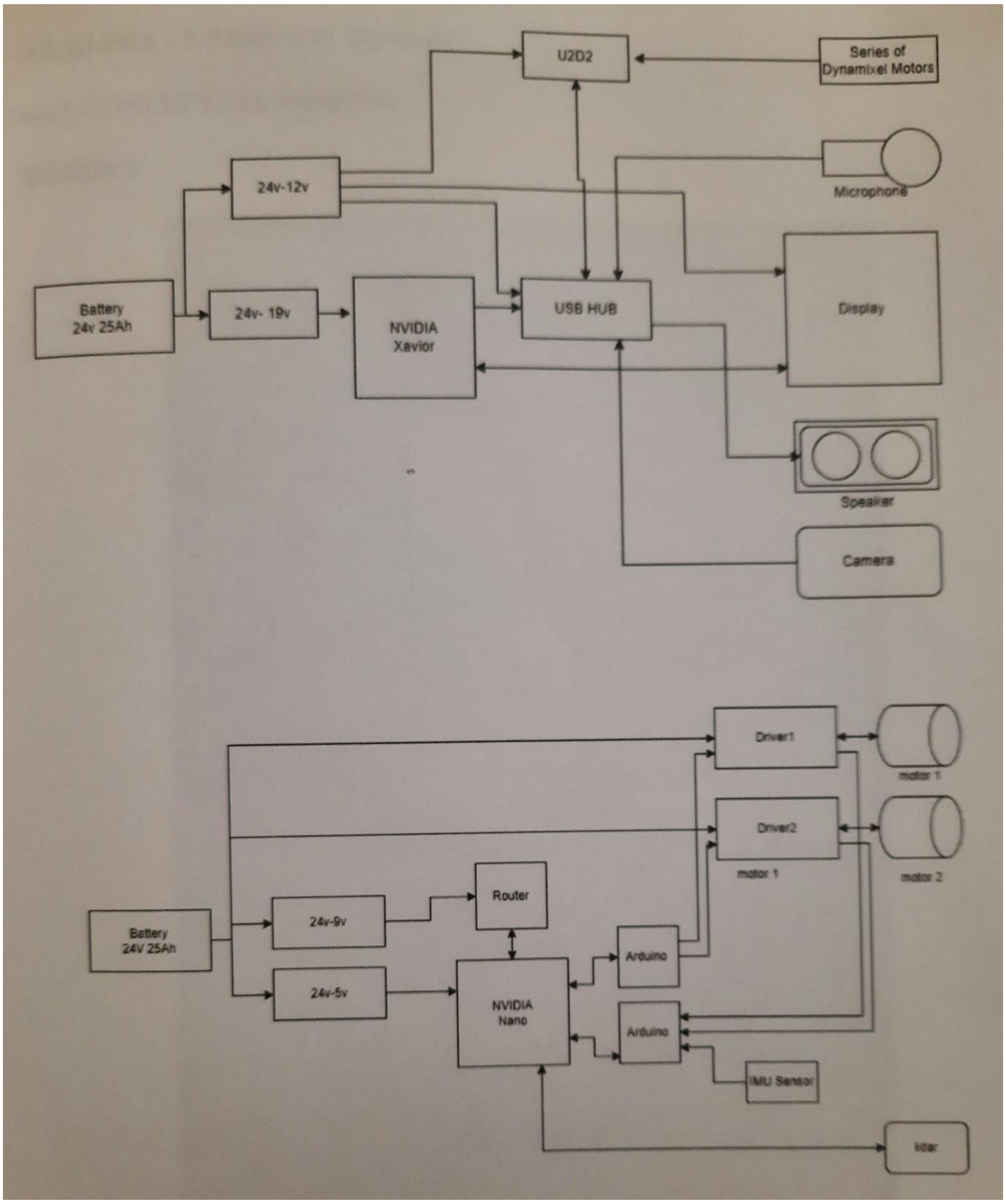

Figure 12 Hardware Architecture for Torso and AMR

CHAPTER 4

MECHANICAL DESIGN

MAYA 3.0 PRIMARY DESIGN

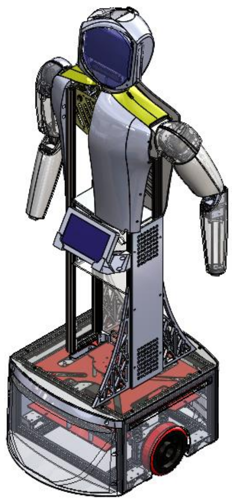

Figure 13 SOLIDWORKS Design of Maya 3.0

DETAILED DESIGN

Table 5 Input Specifications

Default Input Parameters		
Mass	15	Kg
No of Drive Motors	2	Nos
Robot Velocity	0.6	m/s
Supply Voltage	24	V
Desired Acceleration	0.2	m/s^2
Efficiency	70	%
Desired Operating Time	40	Min

Table 6 Other Specifications

	Wheel Size			
Angle of Inclination	**10cm**	**15cm**	**20cm**	**25cm**
5	0.565	0.848	1.13	1.413
10	1.019	1.529	2.04	2.549
15	1.467	2.201	2.935	3.669

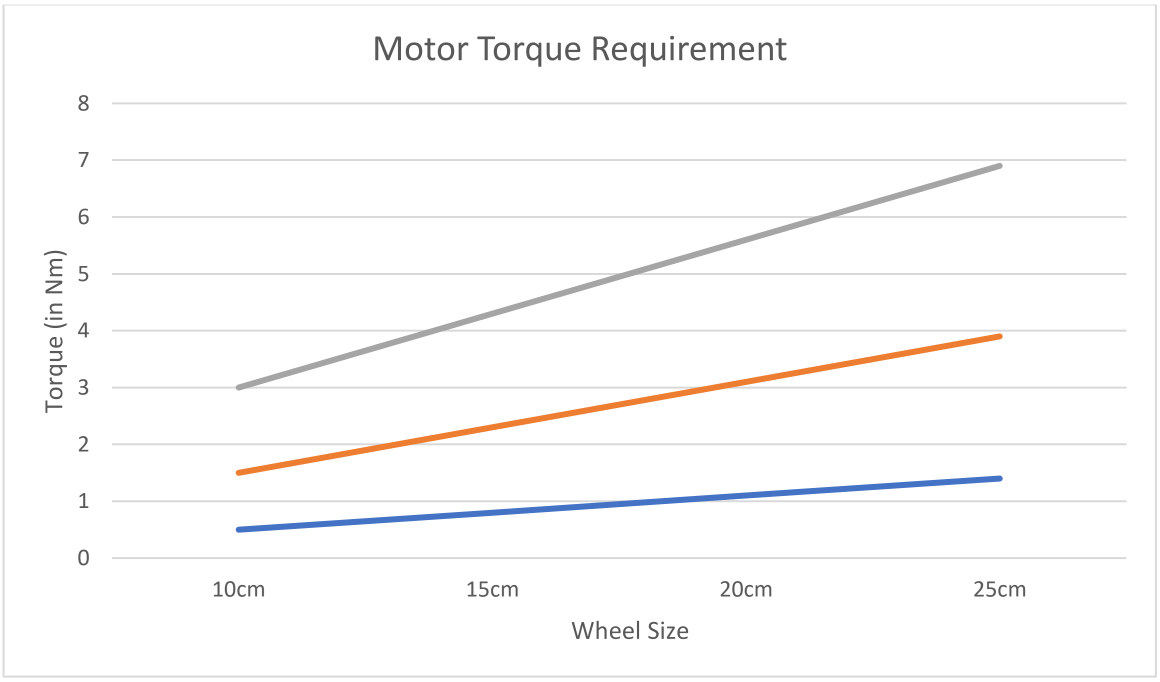

Figure 14 Plot of wheel size (cm) vs Torque required (Nm)

CHAPTER 5

BEHAVIOR DESIGN

5.1 HUMAN ROBOT INTERACTION – HRI

Human-robot interaction (HRI) is a multi-disciplinary field that encompasses a wide range of topics related to the design and evaluation of robots and other technological systems that are intended to interact with humans. The goal of HRI is to enable robots to communicate and collaborate with humans in a natural and intuitive way, and to understand and respond to the needs and preferences of individual users.

One of the main areas of focus in HRI is the design of robot hardware and software. This includes the development of sensors and other technologies that enable the robot to perceive and interpret the actions and intentions of humans, as well as the design of algorithms and other software tools that allow the robot to process and respond to this information.

Another important aspect of HRI is the development of interaction protocols and interfaces that allow humans to communicate their intentions to the robot. This includes the use of natural language processing, touch screens, and other interactive technologies that enable humans to communicate with the robot in a way that is intuitive and easy to use.

In addition to these technical aspects of HRI, there are also social and psychological aspects that need to be considered. For example, researchers in the field of HRI often study the ways in which humans perceive and interact with robots, as well as the psychological and social impacts of human-robot interaction.

One of the main goals of human-robot interaction (HRI) is to enable robots to communicate and collaborate with humans in a way that is natural and intuitive. This requires the development of interaction modalities that allow humans and robots to exchange information and interact in a way that is easy and intuitive for humans to understand.

There are several different types of interaction modalities that can be used to enable effective HRI, including speech, gesture, and touch.

Speech is a common interaction modality that is used in HRI to enable robots to communicate with humans using natural language. This can be achieved through the use of natural language processing algorithms that allow the robot to understand and respond to spoken commands and requests. Speech can also be used to enable the robot to communicate information to the user, such as status updates or alerts.

Gesture is another common interaction modality that is used in HRI to enable robots to communicate with humans using nonverbal cues. This can be achieved through the use of sensors and machine learning algorithms that allow the robot to interpret and respond to human gestures, such as hand movements or facial expressions. Gesture can be a useful interaction modality in situations where speech may not be practical or appropriate, such as in noisy environments or when the user is unable to speak.

Touch is another interaction modality that can be used in HRI to enable robots to communicate with humans using physical contact. This can be achieved through the use of touch sensors and haptic feedback devices that allow the robot to provide tactile feedback to the user. Touch can be a useful interaction modality for enabling robots to provide assistance or support to the user, or for enabling the robot to convey information or emotions through physical touch.

Using these and other interaction modalities, HRI enables robots to communicate and collaborate with humans in a way that is natural and intuitive. This can help to improve the efficiency and effectiveness of human-robot systems, and can make it easier for humans to interact with and benefit from the use of robots in various applications.

Effective human-robot interaction is crucial for the widespread adoption of robots in various fields, such as manufacturing, healthcare, and education. To enable this interaction, robots need to be able to perceive and understand the intentions and actions of humans. This can be achieved through the use of sensors and machine learning algorithms that allow the robot to interpret and respond to human cues, such as facial expressions and body language.

Sensors play a vital role in enabling robots to perceive their surroundings and gather data about the environment and the humans present in it. These sensors can be vision-based, using cameras to capture images and video of the environment, or they can be tactile, using touch or pressure sensors to gather information about the physical properties of objects. Sensors can also be used to detect sounds, smells, and other stimuli in the environment.

Once data is collected by the sensors, machine learning algorithms can be used to process and interpret this data, allowing the robot to make sense of its surroundings and the humans present in it. Machine learning algorithms can be trained on large datasets of human facial expressions, body language, and other cues to recognize patterns and understand their meanings. For example, a robot equipped with machine learning algorithms could recognize that a human is happy based on their facial expression and respond accordingly by initiating a conversation or performing a task that the human desires.

Effective human-robot interaction also requires robots to be able to communicate and interact with humans in a natural and intuitive way. This can be achieved through the use of natural language processing (NLP) algorithms, which allow robots to understand and generate human-like speech and text. NLP algorithms can be trained on large datasets of human language to recognize patterns and understand the meaning of words and phrases. With NLP, robots can understand and respond to spoken or written requests made by humans, making it easier for humans to communicate with and control the robot.

In addition to perception and communication, robots also need to be able to understand and respond to human emotions and social cues. This can be achieved through the use of emotional recognition algorithms, which can be trained on datasets of human facial expressions and other emotional cues to recognize and interpret the emotions of humans. For example, a robot equipped with emotional recognition algorithms could recognize that a human is sad based on their facial expression and respond by offering words of comfort or performing a task that may lift the human's spirits.

Designing intuitive and user-friendly interfaces for human-robot interaction (HRI) is crucial for the widespread adoption of robots in various fields. These interfaces allow humans to communicate their intentions to the robot in a natural and intuitive way, making it easier for them to control and interact with the robot. There are several approaches that can be used to design such interfaces, including the use of natural language processing, touchscreens, and other interactive technologies. Overall, the ability to perceive and understand the intentions and actions of humans is crucial for the widespread adoption of robots in various fields. By using sensors and machine learning algorithms to interpret and respond to human cues, robots can effectively communicate and interact with humans in a natural and intuitive way. As a result, robots can become valuable assistants and partners in various settings, improving the efficiency and effectiveness of tasks and ultimately enhancing the overall human experience.

5.1.1 HRI COGNITIVE ARCHITECTURE

Cognitive architecture is a branch of artificial intelligence that is concerned with the study of the structure and organization of intelligent systems and how they process information to achieve intelligent behaviour. It is an interdisciplinary field that draws on ideas and methods from psychology, neuroscience, computer science, and philosophy to understand how the human mind works and to design artificial intelligent systems that exhibit similar capabilities.

A cognitive architecture is a theoretical framework that describes the functional components, processes, and representations that make up an intelligent system. It defines the basic building blocks of the system, such as perception, attention, memory, reasoning, and decision-making, and how these building blocks interact to produce intelligent behaviour.

Cognitive architectures have various applications, such as natural language processing, robotics, virtual agents, and human-computer interaction, among others. By providing a comprehensive and unified view of intelligent systems, cognitive architectures can help to guide the design, development, and evaluation of intelligent systems, providing insights on how to improve their performance and how to make them more human-like.

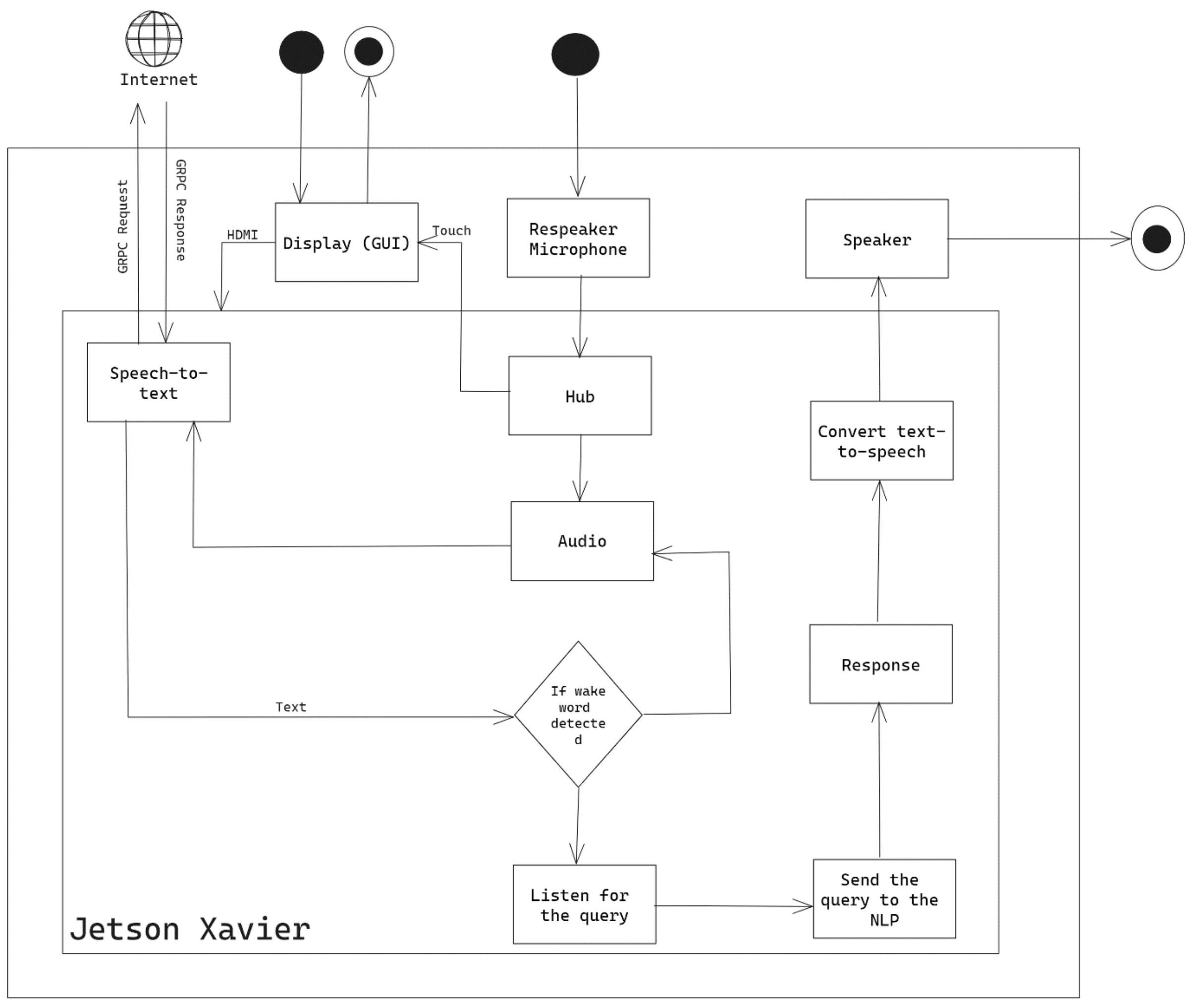

Figure 15 HRI Cognitive Architecture

5.1.2 SPEECH PROCESSING

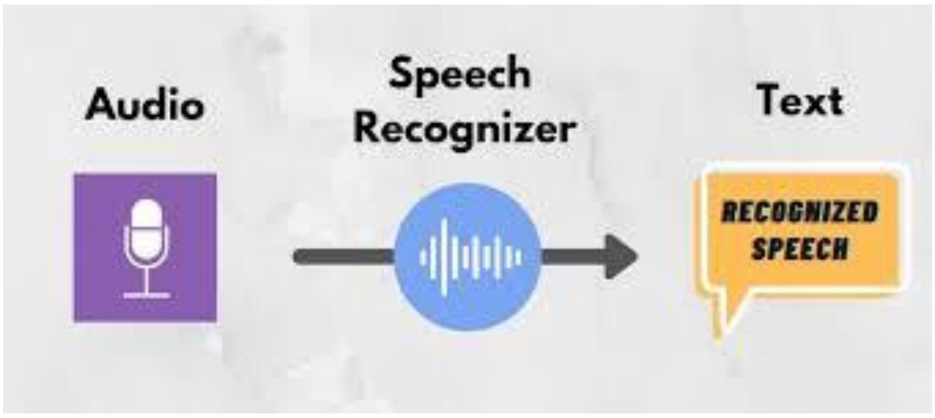

Figure 16 Speech recognition representation

In the employed sophisticated speech processing workflow, a pivotal element is the implementation of a versatile Python-based Speech Recognition library. This advanced tool excels not only in seamlessly transcribing user-spoken words into text but also in mitigating environmental challenges through robust noise cancellation features. This foundational capability sets the stage for the integration of spoken input into our system, offering users a seamless and efficient means of interaction. Beyond basic transcription, the Speech Recognition library's nuanced features contribute to a more accurate and adaptive user experience in diverse auditory settings.

The transcribed text then embarks on a transformative journey through the ChatGPT API, a cornerstone that significantly elevates the capabilities of our system. Leveraging sophisticated natural language processing, the API facilitates a worldwide search for information and generates responses that are not just accurate but also contextually rich. Noteworthy features of the ChatGPT API include multi-turn conversation handling, allowing our system to maintain context over extended interactions, and dynamic content generation, ensuring that responses are not only contextually relevant but also engaging.

Figure 17 ChatGPT API representation

Upon retrieval of the text response from the ChatGPT API, the process takes another turn with the Speech Recognition library deployed in reverse. This bidirectional capability allows for seamless transition, converting the textual information back into spoken words. Conversely, when textual information is transformed into spoken words, the same library showcases its versatility once again. The resulting speech undergoes meticulous magnification as it is channelled through speakers, ensuring a clear, resonant, and contextually appropriate voice output. This comprehensive orchestration of advanced speech recognition and natural language processing technologies, coupled with the expansive capabilities of the ChatGPT API, underscores our commitment to delivering a user-friendly and efficient interactive experience. The fusion of cutting-edge features not only ensures accurate transcription and sophisticated natural language understanding but also highlights our proficiency in bidirectional transformations between text and speech, offering users a seamless and immersive bridge between spoken and written communication realms.

5.1.3 NLP

Natural language processing (NLP) is a field of artificial intelligence and computer science that focuses on the interaction between computers and humans through the use of natural language. In the context of speech processing, NLP algorithms and systems are designed to process, analyze, and understand spoken language in order to facilitate human-computer interaction.

The goal of NLP in speech processing is to enable computers to recognize and transcribe spoken language accurately, as well as to understand and respond to spoken commands and queries. This is accomplished through a combination of speech recognition, language understanding, and natural language generation algorithms.

One of the earliest applications of NLP in speech processing was the development of speech recognition systems for dictation and transcription. These systems used hidden Markov models (HMMs) and other probabilistic models to recognize and transcribe spoken language with a high degree of accuracy.

In the late 1990s and early 2000s, the use of NLP in speech processing expanded to include the development of virtual assistants and conversational agents, such as Apple's Siri and Amazon's Alexa. These systems used a combination of speech recognition, language understanding, and natural language generation algorithms to enable users to interact with the system using spoken language.

5.2 VISITOR MANAGEMENT SYSTEM

5.2.1 FEATURES

Maya 3.0, the highly capable assistance robot, initiates its operational sequence from the convenience of its designated docking station, functioning not only as its starting point but also as the hub for recharging its efficient battery. User interaction begins with a simple turn of the robot using the user-friendly lever knob, activating Maya 3.0 and preparing it for the myriad tasks ahead. The Graphical User Interface (GUI) takes center stage, providing a comprehensive control platform where users can seamlessly configure Maya 3.0 to its predefined home position.

Upon reaching this home position, the user activates the robot by uttering the designated activation word, "Maya," establishing a fluid connection between user and machine. The activated state unlocks a spectrum of possibilities, facilitating engaging and informative interactions. The GUI, with its intuitive design, offers users a range of options, including an immersive tour feature. Maya 3.0, serving as a knowledgeable guide, takes users on a captivating journey through the LAB, introducing them to noteworthy projects such as the miniature assembly line and the remarkable 6 D.O.F Robotic Arm designed for tomato segregation by Team 08.

In addition to project introductions, users can engage in enlightening conversations about the general information pertaining to the Automation and Robotics department. Maya 3.0’s advanced cognitive abilities foster dynamic discussions, providing users with valuable insights. For a personalized touch, users can command Maya 3.0’s to follow them by simply using the term "follow me," showcasing the robot's adaptability and responsiveness in enhancing user interaction. Maya 3.0 stands as a testament to seamless human-robot collaboration, elevating the user experience in automation and robotics exploration.

5.2.2 GUI

Developing a robot with a graphical user interface (GUI) for human interaction poses both challenges and rewarding possibilities, fostering a more natural communication channel. Various approaches can be employed, each tailored to the specific goals and capabilities of the robot.

One approach involves implementing a desktop or web-based GUI, enabling control from any device with a web browser. Leveraging existing tools and frameworks for user interfaces, a desktop GUI can be created with a graphical user interface development environment (IDE), while a web-based GUI can utilize HTML, CSS, and JavaScript.

An alternative option is utilizing a touch screen or display, allowing users to directly interact with the robot through touch or gestures, offering an intuitive and natural communication method. These touch screens and displays, using technologies like resistive, capacitive, or infrared, can be integrated into the robot or connected wirelessly.

A touch screen GUI offers a versatile range of controls, such as virtual buttons, sliders, and drop-down menus, alongside real-time data and video display capabilities. Multi-touch gestures like pinch-to-zoom or swipe enhance interactions with the robot.

Another approach involves using voice or natural language processing, enabling users to communicate with the robot through spoken commands. This method, supported by software libraries and APIs, proves convenient and accessible, especially in noisy environments or for users with disabilities.

Constructing a robot with a GUI demands a blend of technical skills in software development, hardware engineering, and artificial intelligence, coupled with a deep understanding of human-robot interaction and usability principles. By carefully considering the robot's goals and capabilities and utilizing appropriate tools and technologies, an intuitive, efficient, and effective GUI can be crafted for both the robot and its users.

Libraries play a crucial role in GUI development, offering pre-built components and functions for creating diverse user interface elements. Popular choices include various frameworks for web-based GUIs that provide tools and components for building responsive and interactive web applications.

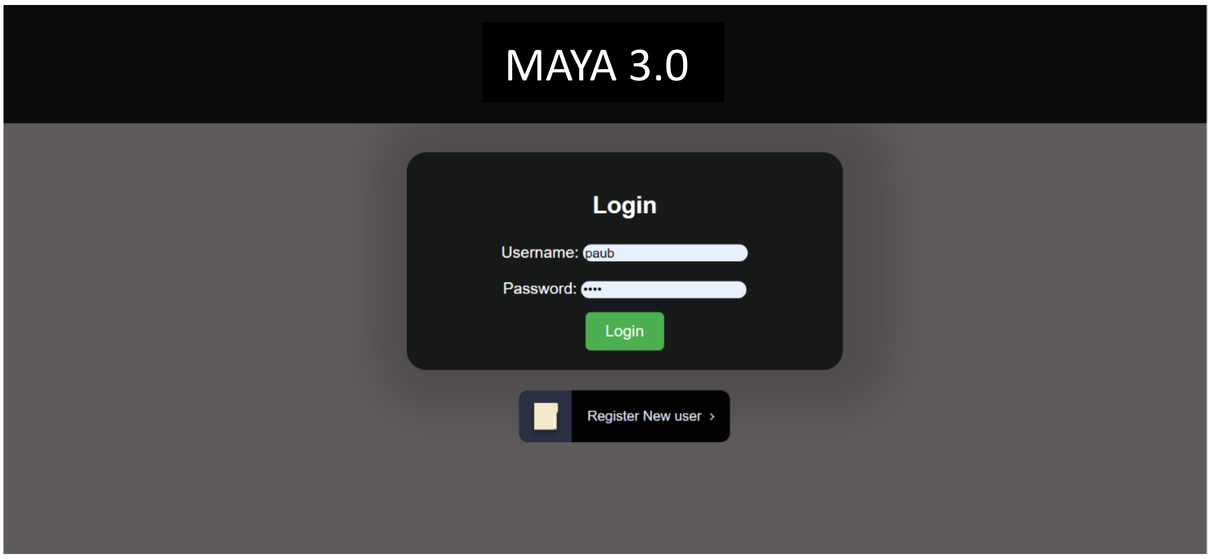

Figure 18 User login representation

For added interaction, the robot can be programmed to respond to the command "Follow me," allowing users to enjoy a dynamic and personalized experience as they move with the robot. In case of the GUI, we were able to effectively Implement a GUI Application integrating the features such as face Recognition, Speech to text, OCR, in order to develop an effective VMS (Visitor Management System). Given a good internet connection, its functionalities worked with a latency of less than 10 ms.

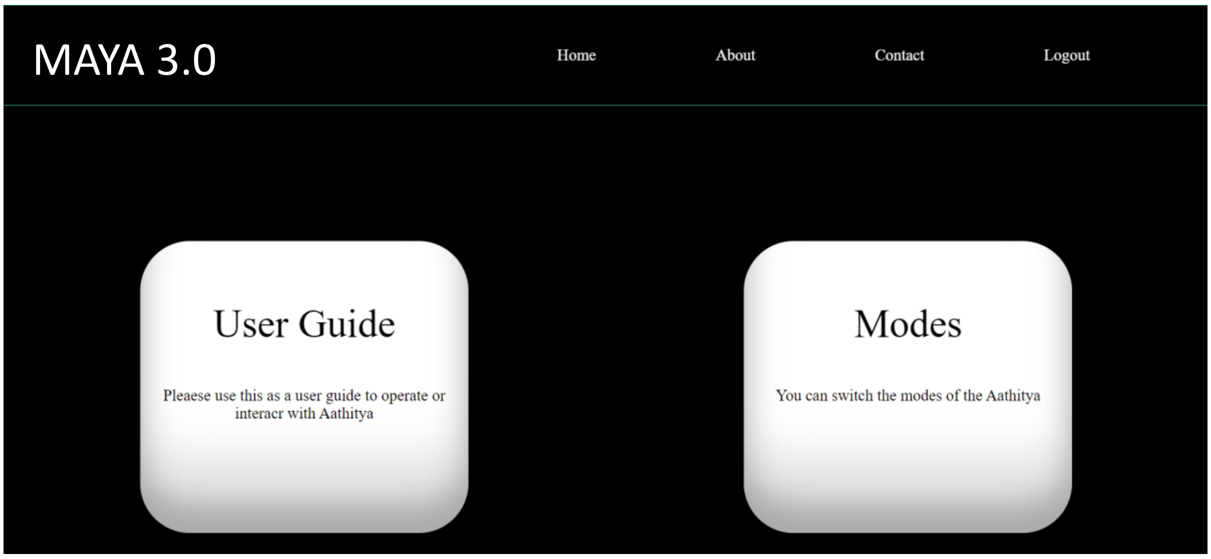
MAYA 3.0
Home
About
Contact
Logout
User Guide
Pleaese use this as a user guide to operate or interacr with Aathitya
Modes
You can switch the modes of the Aathitya

Figure 19 User Guide and Mode selection window

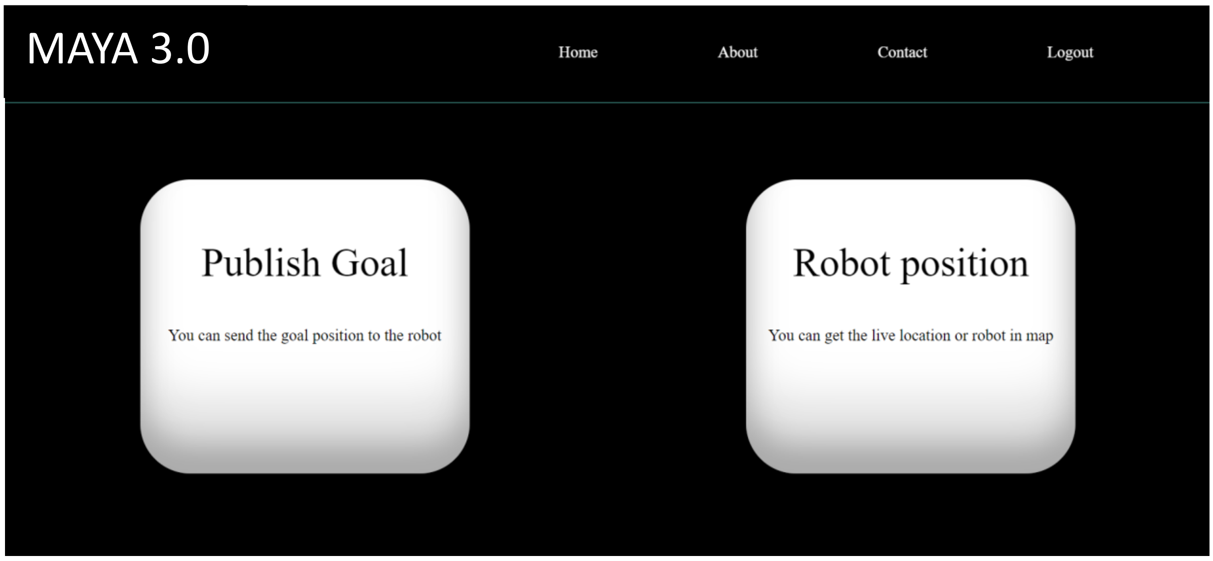
MAYA 3.0
Home
About
Contact
Logout
Publish Goal
You can send the goal position to the robot
Robot position
You can get the live location or robot in map

Figure 20 Publish Goal and Robot position window

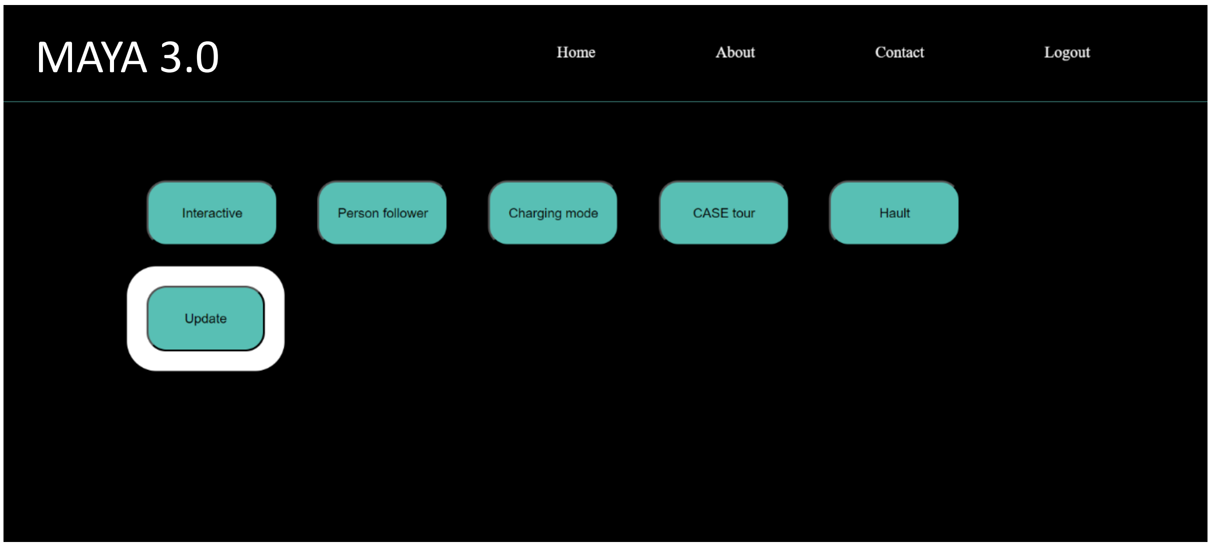

Figure 21 Tour guide window

5.3 PERSON FOLLOWER

A person follower robot, designed to autonomously track and follow an individual while maintaining a predetermined distance, relies on advanced technology for effective operation. One prominent approach involves employing a depth sensor camera, which excels at measuring distances to objects within its field of view. This innovative system finds diverse applications, including providing support to elderly or disabled individuals and serving as a reliable personal assistant or messenger.

To construct a person follower robot with a primary emphasis on the depth sensor camera, the initial step is to equip the robot with this advanced sensor technology. The depth camera becomes instrumental in continuously scanning the environment, capturing detailed depth information that goes beyond mere visual recognition.

In this context, the person follower robot is designed to utilize the depth sensor camera to precisely gauge the distance to the person it is following. By integrating this depth information with the robot's own positional data, which may include wheel odometry or an inertial measurement unit (IMU), the robot establishes a clear understanding of its proximity and orientation relative to the person. This insightful data empowers the robot to modulate its movements intelligently, ensuring it adheres to the specified distance from the person.

Various control methodologies, such as feedback control loops or path planning algorithms, can be implemented to govern the robot's movements effectively. Feedback control loops dynamically adjust the robot's velocity and direction based on real-time disparities between its current position and the desired location. On the other hand, path planning algorithms involve computing a series of intermediate locations that guide the robot toward its target position.

The incorporation of a depth sensor camera offers distinctive advantages, notably in providing a more accurate and reliable means of determining the distance to the person. This proves particularly advantageous in challenging environments where factors such as varying lighting conditions or reflections may impact traditional visual recognition systems. The depth sensor camera's ability to generate a continuous stream of precise distance measurements facilitates seamless tracking of the person's movements, ensuring a consistent and reliable following distance.

Additionally, the depth sensor camera contributes valuable environmental information, offering insights into object shapes, surface layouts, and potential obstacles. This supplementary data enhances the robot's ability to navigate its surroundings adeptly, making informed decisions for effective obstacle avoidance.

While challenges remain, such as optimizing the depth sensor camera for specific environmental conditions, the emphasis on this technology underscores its pivotal role in creating a robust and adaptive person follower robot.

5.3.1 PERSON FOLLOWER ARCHITECTURE

Here’s the figure showing the architecture for person follower with communication between Torso and AMR.

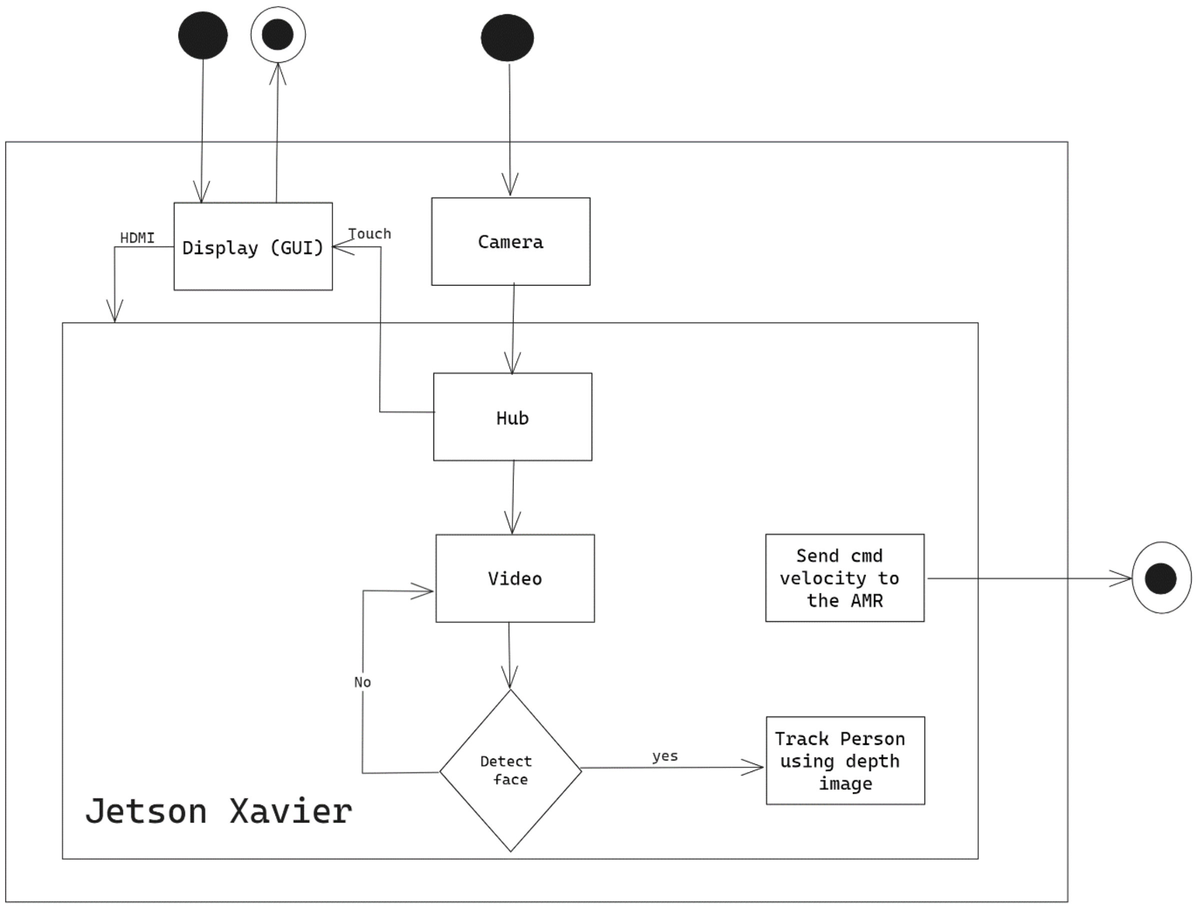

Figure 22 Person Follower Architecture

5.4 AUTONOMOUS NAVIGATION – 2D SLAM

Lidar SLAM (Simultaneous Localization and Mapping) is a method employed to empower autonomous robots and vehicles in navigating their surroundings while simultaneously creating a map of the environment using a laser ranging device, typically a 2D lidar sensor. Lidar, a form of remote sensing technology, utilizes lasers to gauge the distance to objects. Through emitting a laser beam and calculating the time it takes for the beam to return to the sensor, a lidar sensor can establish the distance to objects within its field of view.

A 2D lidar sensor excels in measuring distances in a single plane but lacks the capability to measure distances above or below that plane. Consequently, it finds optimal use in environments characterized by primarily flat surfaces, such as factory floors or roads.

For Lidar SLAM implementation, the robot or vehicle must be equipped with a lidar sensor and a means of self-localization, such as wheel odometry or an inertial measurement unit (IMU). The robot or vehicle scans its surroundings with the lidar sensor, collecting a sequence of range measurements at various angles. These measurements contribute to the creation of a map of the environment, referred to as the "map frame."

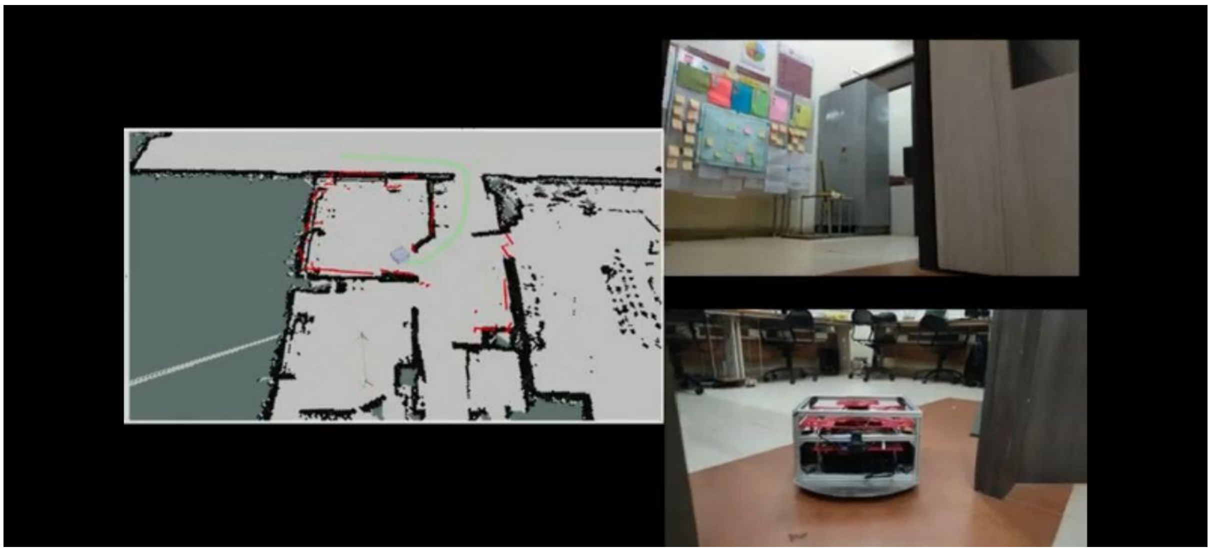

Figure 23 Autonomous Navigation using 2D SLAM representation

Simultaneously, the robot or vehicle utilizes its own motion data, including speed and direction, to ascertain its position within the map—a process known as "localization." Integrating the map of the environment with its self-determined location enables the robot or vehicle to establish its position and orientation at any given time. Several algorithms, including FastSLAM, GridSLAM, and EKF SLAM, utilize lidar sensor data to conduct 2D lidar SLAM, generating real-time maps and determining the device's position within them.

FastSLAM employs a particle filter to concurrently represent the robot's pose and the environment map, updating these using lidar range measurements and robot odometry.

GridSLAM, on the other hand, divides the map into a grid of cells and utilizes lidar range measurements to update the occupancy probability of each cell, along with using motion data for updating the robot's pose within the map.

EKF SLAM (Extended Kalman Filter SLAM) utilizes a Kalman filter to estimate the robot's pose and the environment map simultaneously, updating both using lidar range measurements and robot odometry.

Despite its merits, Lidar SLAM faces challenges associated with noise and errors in lidar sensor range measurements, arising from reflections, laser interference, or sensor noise. Lidar SLAM algorithms typically incorporate mechanisms to handle outliers and smooth the map to mitigate the impact of noise.

Another challenge lies in the computational cost of Lidar SLAM algorithms, demanding substantial processing power, which can be limiting in resource-constrained systems. Researchers have explored optimization techniques, including approximations and approximative data structures, to address this computational challenge.

5.4.1 ODOMETRY

Odometry is a method of measuring the distance travelled by a mobile robot using its internal sensors, such as wheel encoders or inertial measurement units (IMUS). It is an important localization technique that is used in autonomous mobile robots to estimate their position and orientation relative to their starting location.

There are two main types of odometry: wheel odometry and visual odometry. Wheel odometry uses the rotational movement of the robot's wheels to estimate its displacement. This is done by measuring the number of rotations of each wheel using encoders, and using this information to calculate the distance travelled.

Visual odometry, on the other hand, uses the robot's cameras to estimate its displacement. It does this by analyzing the change in the appearance of the environment as the robot moves, and using this information to estimate its movement.

Both wheel odometry and visual odometry have their own advantages and disadvantages. Wheel odometry is generally more accurate and reliable than visual odometry, but it is limited by the accuracy of the wheel encoders and the assumption that the wheels are always in contact with the ground. Visual odometry, on the other hand, is less affected by these limitations, but it is more sensitive to noise and changes in lighting conditions.

In autonomous mobile robots, odometry is often used in conjunction with other localization techniques, such as GPS or SLAM (simultaneous localization and mapping), to improve the accuracy and reliability of the localization system. However, odometry alone is often not sufficient for high-precision localization, as it is prone to drift over time due to errors in the sensor measurements.

To mitigate this drift, odometry systems often incorporate techniques such as Kalman filtering or particle filtering to fuse data from multiple sources and reduce the effect of noise and errors on the estimate of the robot's position and orientation.

In short, odometry is an important localization technique for autonomous mobile robots that allows them to estimate their position and orientation relative to their starting location based on their internal sensors. While it is prone to drift and errors, it can be used in conjunction with other techniques to improve the accuracy and reliability of the localization system.

5.4.2 MAPPING

Mapping is a crucial aspect of robotics that enables a robot to understand the layout and structure of its environment. This is necessary for the robot to be able to navigate and perform tasks within that environment. There are many different approaches to mapping, ranging from simple techniques that rely on pre-drawn maps to more sophisticated methods that involve real-time exploration and mapping of the environment.

The Robot Operating System (ROS) is a popular framework for developing robotics applications. It provides a range of tools and libraries that can be used to solve the mapping problem, including support for a variety of sensors (e.g. lidar, camera) and algorithms (e.g. simultaneous localization and mapping (SLAM)).

In ROS, mapping is typically implemented as a node in the robot's system architecture. The mapping node receives data from sensors (such as lidar scans or camera images) and processes this data to create a map of the environment. This map is then published as a message for other nodes to use. The mapping node may also communicate with other nodes, such as the localization node, to refine the map and improve the accuracy of the robot's pose estimation.

Thus, the development of effective mapping systems is an important aspect of robotics research and practice. ROS provides a range of tools and libraries to support the development of mapping systems for a variety of robotic platforms.

5.4.3 LOCALIZATION

Localization is a key problem in robotics that enables a robot to understand its position and orientation within its environment. This is necessary for the robot to be able to navigate and perform tasks within that environment. There are many different approaches to localization, ranging from simple techniques that rely on pre-drawn maps to more sophisticated methods that involve real-time exploration and mapping of the environment.

In ROS, localization is typically implemented as a node in the robot's system architecture. The localization node receives data from sensors (such as lidar scans or odometry data) and processes this data to estimate the robot's pose. This estimate is then published as a message for other nodes to use. The localization node may also communicate with other nodes, such as the mapping node, to refine the map and improve the accuracy of the robot's pose estimation.

Accurate localization is crucial for the successful operation of a robot. It allows the robot to navigate its environment and perform tasks with a high degree of precision. ROS provides a range of tools and libraries to support the development of localization systems for a variety of robotic platforms.

There are many factors that can affect the accuracy of localization, including sensor noise and errors, modelling errors, and variations in the environment. To address these issues, it is important to use sensors that are accurate and reliable, and to carefully model the sources of error in the system. In addition, it is often necessary to use a combination of different sensors and algorithms to achieve the best possible localization performance.

One common approach to localization is to use a map of the environment, which can be created in advance or in real-time using a technique such as simultaneous localization and mapping (SLAM). The map can then be used to constrain the robot's pose estimate and improve the accuracy of the localization.

Another important aspect of localization is the ability to track the robot's pose over time. This is typically achieved using techniques such as Kalman filters, which combine sensor measurements with a model of the robot's motion to produce an estimate of the robot's pose that is more accurate than any individual measurement.

Simultaneous Localization and Mapping (SLAM) is the problem of creating a map of an unknown environment while simultaneously estimating the pose (position and orientation) of a robot within that environment. SLAM is a key problem in robotics, as it enables a robot to navigate and perform tasks within an unknown environment.

In ROS, there are several software packages that can be used to solve the SLAM problem, including the following:

gmapping: This package provides an implementation of the grid-based FastSLAM algorithm, which uses laser rangefinder data to create a map of the environment and estimate the robot's pose.

Hector SLAM: This package provides an implementation of the Hector SLAM algorithm, which uses laser rangefinder data to create a map of the environment and estimate the robot's pose.

Karto SLAM: This package provides an implementation of the Karto SLAM algorithm, which uses laser rangefinder data to create a map of the environment and estimate the robot's pose.

RTAB-Map (Real-Time Appearance-Based Mapping): This package provides an implementation of the RTAB-Map SLAM algorithm, which uses RGB-D data (color and depth images) to create a map of real time environment.

5.4.4 NAVIGATION

Navigation is a fundamental problem in robotics that involves the planning and execution of paths for a robot to follow within an environment. To navigate effectively, a robot must be able to perceive its environment, estimate its position and orientation, and plan and execute paths that avoid obstacles and reach the desired destination. This requires a combination of sensors, algorithms, and software to enable the robot to perceive and understand its environment, and to plan and execute safe and efficient paths.

In ROS, navigation is typically implemented as a stack of nodes that work together to plan and execute paths for the robot to follow. The navigation stack includes support for local and global planners, which are responsible for generating paths at different scales. The local planner is responsible for generating a smooth, collision-free path for the robot to follow, while the global planner is responsible for generating a high-level path for the robot to follow that takes into account the overall layout of the environment.

Effective navigation is a crucial aspect of robotics and is closely connected to the ability of a robot to operate effectively within its environment. ROS provides a range of tools and libraries to support the development of navigation systems for a variety of robotic platforms.

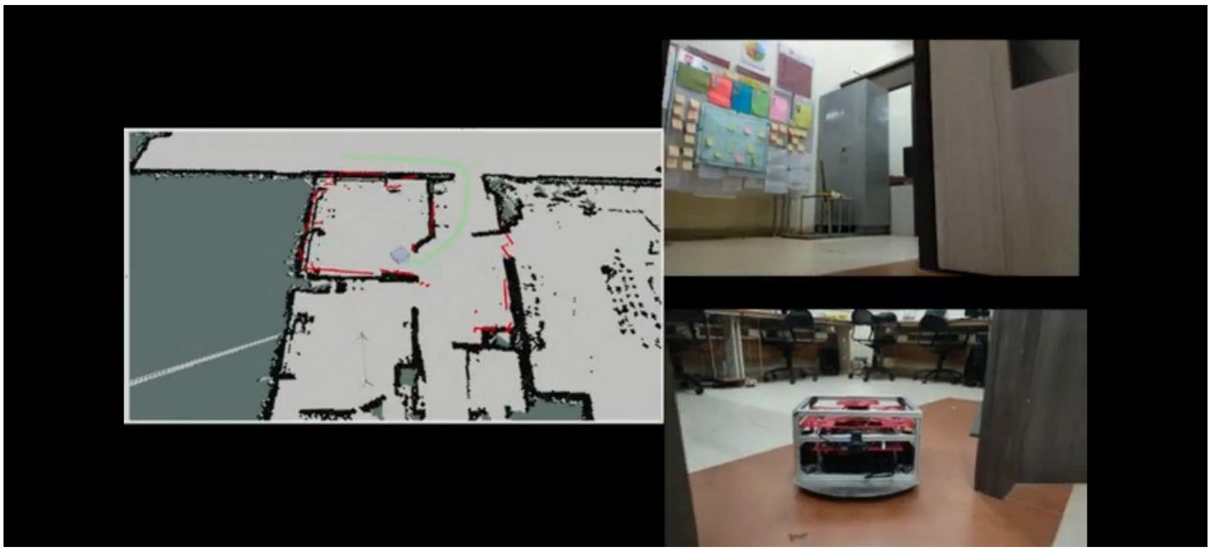

Figure 24 Navigation representation

By using the specifications of the robot such as separation between wheels, wheel diameter, the height of lidar, the dimensions of the AMR the navigation stack was customized to run for our AMR smoothly. Using this we created map and set the parameters of local costmaps, global costmaps and planner parameters to work on our robot.

The localization process is then carried out by the AMR to find its pose in the given map and receive the goal pose and input as seen in the figure below.

The robot then plans the path to be followed to reach the destination using the DWA planner and the velocities for each wheel are decided and sent to the motors by differential drive controller.

CHAPTER 6

RESULTS AND ANALYSIS

The culmination of the team’s humanoid robot project signifies a ground breaking achievement. It revolves around crafting a versatile robotic system tailored for educational institutions while holding the potential for seamless integration into hospitals and elderly care facilities with appropriate modifications. Overcoming the intricate challenges inherent in replicating the nuanced functionalities of the human brain required a strategic approach. The implementation of two distinct controllers, one for interaction and the other for navigation, emerged as a key enabler for operational efficiency and system independence.

In terms of navigation, our humanoid robot project embraced cutting-edge techniques, emphasizing precise movement and spatial awareness. Leveraging advanced navigation systems, notably Simultaneous Localization and Mapping (SLAM) technology, empowered the robot with the capability to maneuver adeptly and comprehend its surroundings effectively, fostering seamless interactions within its environment.

The mechanical modularity embedded in the system played a pivotal role in the project's success. This modular design allowed for the independent testing and operation of both interaction and navigation components, ensuring streamlined functionality and optimal performance.

Beyond navigation, the humanoid robot showcased its versatility through user assistance and collaborative functionalities. With incorporated docking capabilities resembling those of Automated Mobile Robots (AMR), the robot demonstrated the ability to autonomously dock for recharging or await user commands, underscoring its adaptability in dynamic environments.

In summary, the humanoid robot project represents a significant stride in the development of adaptable and intelligent robotic systems. The successful integration of advanced navigation techniques, coupled with modular design principles and versatile functionalities, positions the robot as a valuable asset capable of meeting the diverse needs of various institutions.

CONCLUSION

In this project, the team effectively accomplished the objectives we established before commencing it. The humanoid robot that we created successfully demonstrated all the requirements. The methodology we used in creating this robot aided us in navigating the challenging integration phase of the process and in determining the robot's behaviour with regard to its hardware and software components in the system.

All of the behaviours were independently evaluated and yielded the intended outcomes. The integration software that we created was effective in performing all of the behaviours at the same time and in the proper order.

With a strong internet connection, the human robot interaction system was able to deliver the results with a latency of 1-2 seconds, recognise the users' voices, and convert speech to text and text to speech. The NLP, or intent classification algorithm, which was created especially for institutional usage, was able to recognise the command and produce a response depending on the command's intent.

Based on the data retrieved, the robot's facial recognition system was able to distinguish between various persons. With a delay of 10 ms, detecting a human required very little time. The algorithm proved successful in distinguishing the traits of several faces.

On flat surfaces, the base component may travel without difficulty. The motors' speed and position were precisely controlled by the control system that was created for them. We were able to effectively complete autonomous navigation thanks to the odometry that was designed for navigation's minimal mistakes. In every given confined map, the AMR might go from one location to another while dynamically dodging obstacles. We were able to combine HRI with self-driving navigation to provide goal-based navigation.

The upper part of the robot was able to send the values, such as angle and speed, to the AMR for it to perform locomotion, and the AMR followed the person at a safe distance. The robot was able to detect the person, and using object tracking algorithms, it successfully calculated the depth and orientation of the object. We could achieve designing, building, assembling of the humanoid robot and we were able to deploy individual behaviours independently and in an integrative way, within the given time period, i.e. 6 Months.

FUTURE SCOPE

Autonomous mobile robots (AMRS) have the potential to revolutionize various industries by enabling the automation of tasks that are currently performed by humans. These robots can work as standalone systems or be integrated with other modular systems for various applications. In the future, the scope of AMRs is likely to expand significantly as they become more advanced and capable of performing a wider range of tasks.

One potential application of AMRS is in manufacturing, where they can be used to transport materials, parts, and finished products within a factory. AMRs can also be used for inspection and maintenance tasks, such as inspecting machinery and equipment for signs of wear or damage. In the healthcare industry, AMRs can be used to transport medical equipment and supplies, assist with patient care, and perform tasks such as taking vital signs.

AMRs can also be used in agriculture and forestry to perform tasks such as planting and harvesting crops, as well as monitoring and maintaining crops. In transportation, AMRs can be used to deliver goods and packages, as well as to transport passengers. AMRs can also be used in construction and mining to perform tasks such as excavating and moving materials.

One of the key benefits of AMRs is their ability to work in environments that may be hazardous or difficult for humans to access, such as nuclear power plants or disaster areas. In these cases, AMRs can be used to perform tasks that are too dangerous for humans, such as inspecting and maintaining equipment or cleaning up hazardous materials.

Overall, the future scope of AMRs is likely to be very broad, as they become increasingly advanced and capable of performing a wide range of tasks in various industries. As AMRs become more widely adopted, they are likely to have a significant impact on the way work is performed and will play an important role in improving efficiency and productivity in various sectors.

REFERENCES

1] C. Hernandez and J. L. Fernandez-Sanchez, "Model-based systems engineering to design collaborative robotics applications," 2017 IEEE International Systems Engineering Symposium (ISSE), 2017, pp. 1-6, doi: 10.1109/SysEng.2017.8088258

[2] Hernández Corbato, Carlos & Bharatheesha, Mukunda. (2020). A Systems Engineering Analysis of Robot Motion for Team Delft's APC Winner 2016. 10.1007/978-3-030-35679-8_7

[3] Corke, P.I.: Robotics, Vision and Control: Fundamental Algorithms in MATLAB. Springer, Berlin (2011)

[4] J. L. Fernández-Sánchez, ISE & Process Pipelines in OO Architectures (ISE&PPOOA), [online] Available: http://www.omgwiki.org/MBSE/doku.php?id=mbse:ppooa

[5] Fernandez, Jose. (2016). Reengineering the Avionics of an Unmanned Aerial Vehicle. IEEE Aerospace and Electronic Systems Magazine. 31. 6. 10.1109/TAES.2016.150048

[6] Koren Y & Borenstein J 1991 Potential field methods and their inherent limitations for mobile robot navigation. In Robotics and Automation, 1991. Proceedings. 1991 IEEE International Conference on pp. 1398-1404 IEEE

[7] Liu, Xingdong & Luo, Xiao & Zhong, Xinliang. (2018). Research on simultaneous localization and mapping of indoor mobile robot. Journal of Physics: Conference Series. 1074. 012099, 10.1088/1742-6596/1074/1/012099

[8] Hu C, Hu C, He D & Gu Q 2015 A new ROS-based hybrid architecture for heterogeneous multi-robot systems Control and Decision Conference (pp.4721-4726) IEEE

[9] Ren W, He D X & Zhao J 2012 A Component Based Hybrid Embedded Framework for Mobile Robot International Conference on Parallel and Distributed Computing, Applications and Technologies Vol.23, pp. 16-19 IEEE Computer Society

[10] Lawonn K, Mönch T & Preim B 2013 Streamlines for Illustrative Real Time Rendering. In Computer Graphics Forum Vol. 32 No. 3pt3 pp. 321-330 Blackwell Publishing Ltd.

11] Schwarz J, Mankoff J & Hudson S 2011 Monte carlo methods for managing interactive state, action and feedback under uncertainty In Proceedings of the 24th annual ACM

on User interface software and technology pp. 235-244 ACM

symposium

[12] Cornejo A, Lynch A J, Fudge E, Bilstein S, Khabbazian M & McLurkin J 2013 Scale-free coordinates for multi-robot systems with bearing-only sensors The International Journal of Robotics Research 32(12) 1459-1474

[13] Demski P, Mikulski M & Koteras R 2013 Characterization of Hokuyo UTM-30LX Laser Range Finder for an Autonomous Mobile Robot Advanced Technologies for Intelligent Systems of National Border Security. Springer Berlin Heidelberg

[14] Li T, Sattar T P & Sun S 2012 Deterministic resampling: Unbiased sampling to avoid sample impoverishment in particle filters Elsevier North-Holland, Inc

[15] Tsao C L, Sanadhya S & Sivakumar R 2015 A super-aggregation strategy for multi-homed mobile hosts with heterogeneous wireless interfaces Wireless Networks 21(2) 639-658

[16] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," The international journal of Robotics Research, vol. 5, pp. 56-68, 1986.

[17] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," Robotics and Automation, IEEE Transactions on, vol. 17, pp. 229-241, 2001.

[18] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," in In Eighteenth national conference on Artificial intelligence, Menlo Park, CA, USA, 2002, pp. 593-598

[19] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: 2.0 An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges," in In Proceedings of the international joint conference on Artificial intelligence (IJCAI03), San Francisco, CA, USA, 2003, pp. 1151-1156.

[20] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," The International Journal of Robotics Research, vol. 25, pp. 1181-1203, 2006.

[21] C. Kim, R. Sakthivel, and W. K. Chung, "Unscented FastSLAM: A Robust Algorithm for the Simultaneous Localization and Mapping Problem," presented at the IEEE international conference on robotics and automation, Roma, Italy, 2008.

[22] L. Moreno, S. Garrido, D. Blanco, and M. L. Muñoz, "Differential evolution solution to the SLAM problem," Robotics and Autonomous Systems, vol. 57, pp. 441-450, 2009.

[23] Khairuddin, A. R., Talib, M. S., & Haron, H. (2015). Review on simultaneous localization and mapping (SLAM). 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE). doi:10.1109/iccsce.2015.7482163.

[24] Zhang Zhaohui, Mei Xuesong, Bian Xu, Cai Hanghang and Ti Jian, "Development of an intelligent interaction service robot using ROS," 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2016, pp. 1738-1742, doi: 10.1109/IMCEC.2016.7867516.

[25] L. Fei, L. Na and L. Jian, "A new service composition method for service robot based on data-driven mechanism [C]", Computer Science & Education (ICCSE) 2014 9th International Conference on, pp. 1038-1043, 2014.

[26] "World robotics 2014 service robots", IFR Statistical Department, October 2014.

[27] Calderon Carlos, Antonio Acosta, Changjiu Zhou and Rajesh Elara Mohan, "Development of an autonomous service robot for social interactions", Information Communications and Signal Processing (ICICS) 2011 8th International Conference on, 2011. [28] H. M. Do, C. J. Mouser, Y. Gu et al., "An open platform telepresence robot with natural human interface[M]", 2013.

[29] K. Koay, E. Sisbot, D. Syrdal, M. Walters, K. Dautenhahn and R. Alami, "Exploratory study of a robot approaching a person in the context of handing over an object", Proc. of AAAI 2007 Spring Symposia, pp. 18-24, March 2007.

[30] Goodrich, Michael & Schultz, Alan. (2007). Human-Robot Interaction: A Survey. Foundations and Trends in Human-Computer Interaction. 1. 203-275. 10.1561/1100000005.
[31] Coleman, D.. (2015). Human-robot interactions: Principles, technologies and challenges.
[32] The SMOOTH-Robot: A Modular, Interactive Service Robot, Krüger et al., https://www.frontiersin.org/articles/10.3389/frobt.2021.645639/full.

[33] I Pavan Raju, Dr. Sunil Sikka, Mr. Ankit Garg, &amp; Mr. Manoj Pandey. (n.d.). A brief review of recent advancement - PAL Robotics. Retrieved January 7, 2023, from https://pal-robotics.com/wp-content/uploads/2020/08/A-BRIEF-REVIEW-OF-RECENT-ADVANCEMENT-IN.pdf

[34] "8 Autonomous Mobile Robot Applications - Guidance Automation." Guidance www.guidanceautomation.com/blog/a-guide-to-autonomous-mobile-robots-amrs-and-8-amr-applications. Accessed 7 Jan. 2023.

Automation,

[35] Rubio F, Valero F, Llopis-Albert C. A review of mobile robots: Concepts, methods, theoretical framework, and applications. International Journal of Advanced Robotic Systems. 2019;16(2). doi:10.1177/1729881419839596

[36] Garcia-Fidalgo, Emilio & Ortiz, Alberto. (2014). Vision-based topological mapping and localization methods: A survey. Robotics and Autonomous Systems. 64.

10.1016/j.robot.2014.11.009.

[37] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. Von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, A. Bernardino, and L. Montesano. The iCub humanoid robot: an open-systems platform for research in cognitive development. Neural Networks, 23(8-9):1125-1134, 2010

I. Bill of Materials

Table 7 Mechanical

Sl No.	Part Name	QTY.
1	Casing	1
2	Cage	1
3	Base	1
4	Castor plate	4
5	Wheel holder	4
6	Wheel pin	4
7	Castor Temp HB	16
8	Castor wheel	4
9	Castor mount plate	4
10	21 mm spacer	16
11	Spring Rod	4
12	SC8UU Linear Bearings	8
13	Drive Motor Plate	2
14	1-5 Planetary GearBox	2
15	57cme13.stp	2
16	Motor Gearbox Spacer	2
17	Wheel Fill	2
18	Wheel Mount	2
19	Wheel mount washer	2
20	Wheel mount v4	2
21	ISO 10669-4.7-N	12
22	ISO 4762 M$ x 40-20N	12
23	Cage part	1
24	Spring Base plate	2
25	Springs	4
26	Electronics base	1
27	Space Electronic base	5

28	Fusion011	2
29	Fusion010	2
30	Fusion009	2
31	Fusion008	2
32	Fusion007	2
33	Fusion006	2
34	Fusion005	2
35	15EDGK-3.81-04P-14-00AH	2
36	15EDGK-3.81-06P-14-00AH	2
37	15EDGK-3.81-03P-14-00AH	2
38	15EDGK-3.81-08P-14-00AH	2
39	DS1040-08RT v1	2
40	Component5	2
41	Cover	2
42	Lidar base	1
43	Lidar top	1
44	Spacer	4
45	LIDAR_2	1
46	LIDAR_1	1
47	Part4	4
48	Roof	1
49	Alu Prof str	2
50	Chest plate	2
51	Alu_prof_support	4
52	Shelf	2
53	Back acrylic	2
54	SKF – 61800-14, SI, NC,14 68	12
55	Bearing 6908 0	2
56	SKF 6806 – 26<SI, NC,22 68	2
57	SKF 6806 – 26<SI, NC,26 68	2

58	Support	4
59	Mirror_p1	1
60	Mirror_p2	1
61	Q4_cover_front	2
62	Q4 cover back	2
63	Q5_v3 cover_front	2
64	Q5_v3 cover_back	2
65	MX magnet mount 2	2
66	RHand	1
67	Carbon square rods	10
68	Q3_v2_Roght	1
69	Q1_p2	2
70	Nut_holder_leadscrew	2
71	MX-106R	2
72	Q2 Led motor	2
73	Q2 motor	2
74	Leadscrew	2
75	MX-64R	3
76	Q4	2
77	XM, H-430 idler	4
78	Q5	2
79	Last_v2	2
80	LHand_F	1
81	Q3 v2 Left	1
82	Neck_motor	1
83	Face_back_v3	1
84	Shield	1
85	Socket_housing	1
86	Socket_plate	1
87	terminal	5

88	26-pin-Header-Female-Double2.54mm	1
89	Tactical switch – SMD(6mmX6mmX5mm)	5
90	Zl202-26_p2-54_1330_w5-08_h8-9	1
91	LCD 7 inch_1	1
92	LCD 7 inch_0	1
93	LCD 7 inch_1	1
94	LCD 7 inch_2	1
95	LCD 7 inch_3	1
96	LCD 7 inch_4	1
97	Face_front	1
98	DS5_0390 ADHESIVE D455	1
99	DS5_0390 Foam D455	1
100	MEC060311_6	1
101	MEC060313 2	1
102	MEC060318 5	1
103	MEC060319_2	1
104	SWEEP1_1	2
105	SWEEP1_1_2_3_1	2
106	MEC060316_8	1
107	MEC060019 1	1
108	MEC060304_3	1
109	FAS000225_6	2
110	EC060308_5	1
111	AM0600115 INT REVO 1 2 SW0001 1	1
112	Neck motor shaft	1
113	Face back	1
114	Face tint	1
115	Sidebotton	4
116	Front bottom	2
117	Sides	4

118	Chest front left	2
119	Chest front right	2
120	Ch_f_f_LL	2
121	Ch_f_f_RR	2
122	Shoulder	1
123	AMR front cover	1
124	AMR front cover top	1
125	AMR back cover	1
126	AMR back cover top	1
127	AMR side 0	2
128	AMR side 2	2
129	Screen holder	1
130	Display	1
131	Screen holder 2	1
132	Q3_R cover	1
133	Q3_L cover	1

Table 8 Electrical

Sl No.	Component	Quantity	Specification
1	Nvidia Nano	1	5V 2A
2	Nvidia Xavier	1	19V 2A
3	Buck Converter	4	32V 10A
4	ATMega 328	1	5V 500mA
5	ATMega 2560	2	5V 500mA
6	Nema 23 Stepper motor	2	24V 2.8A
7	Leadshine stepper driver	1	50V 4.2A
8	U2D2 Converter	3	12V 4A
9	Dynamixel Motor	1	12V 4A
10	Intel realsense	1	5V
11	Respeaker Microphone array	1	5V 180mA

12	DC MCB	1	500V 16A
13	DC MCB	1	500V 10A
14	Red LED	2	24V 20mA
15	Green LED	2	24V 20mA
16	Orange LED	2	24V 20mA
17	Emergency Switch	2	10A
18	3 Position Rotary switch	2	10A
19	LCD Display	1	12V
20	Wifi Router	1	9V 1A
21	USB HUB	2	12V 4A
22	Batter	2	24V 25Ah
23	Speaker	2	12V
24	RP Lidar	1	5V 0.6A

II. DETAILED DRAWINGS OF PARTS

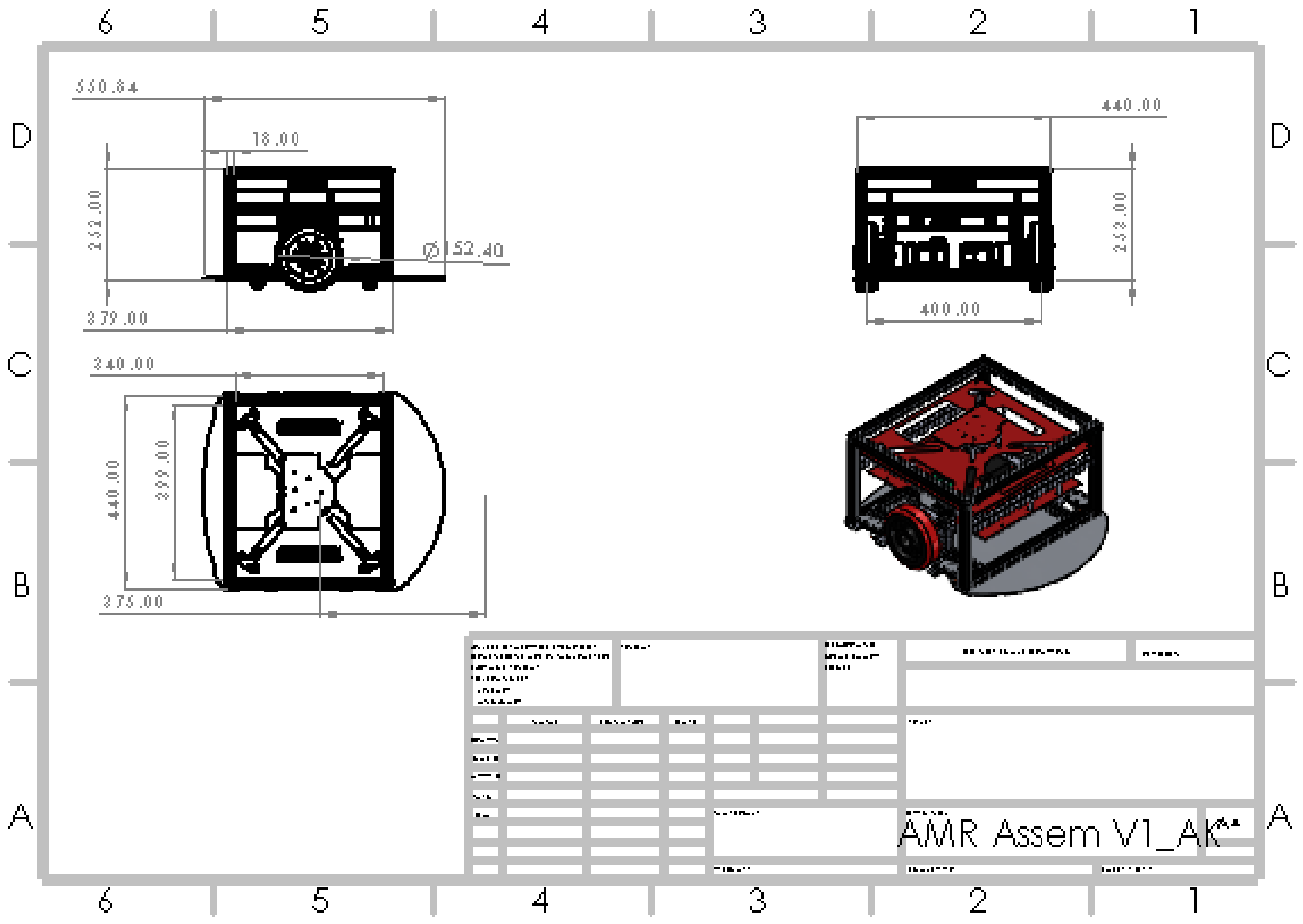

Figure 25 AMR Assembly draft

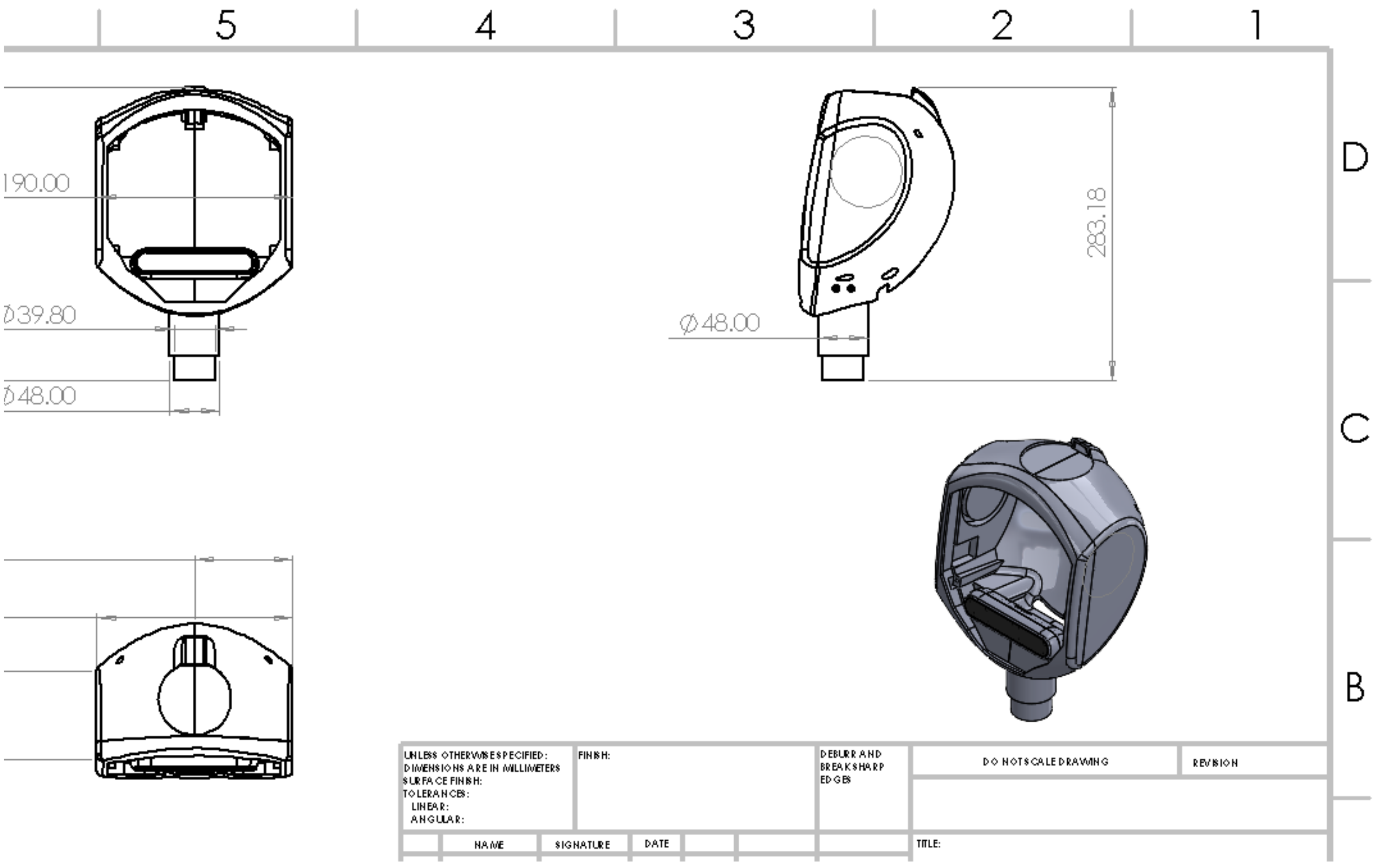

Figure 26 Head Draft

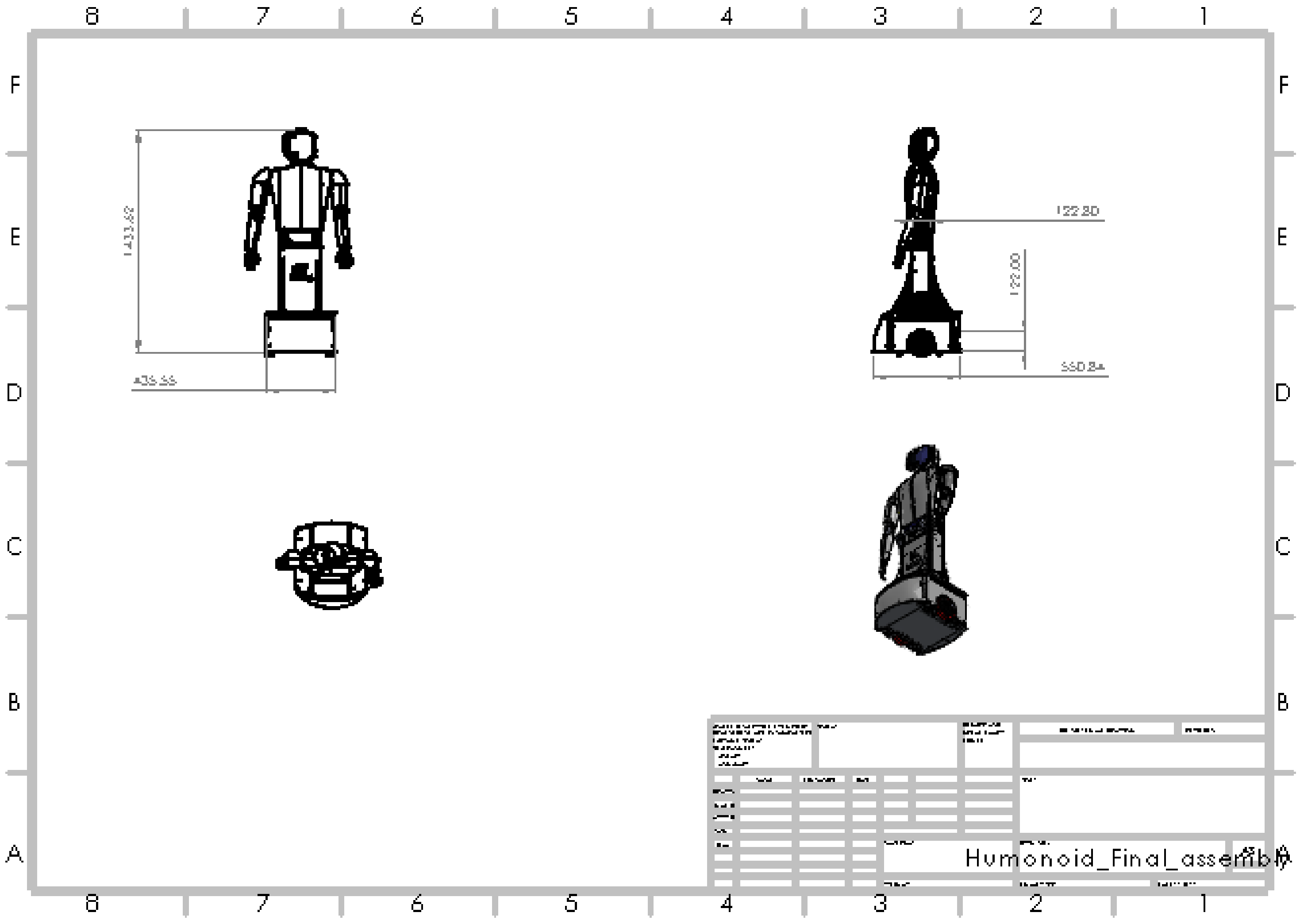

Figure 27 Humanoid robot draft

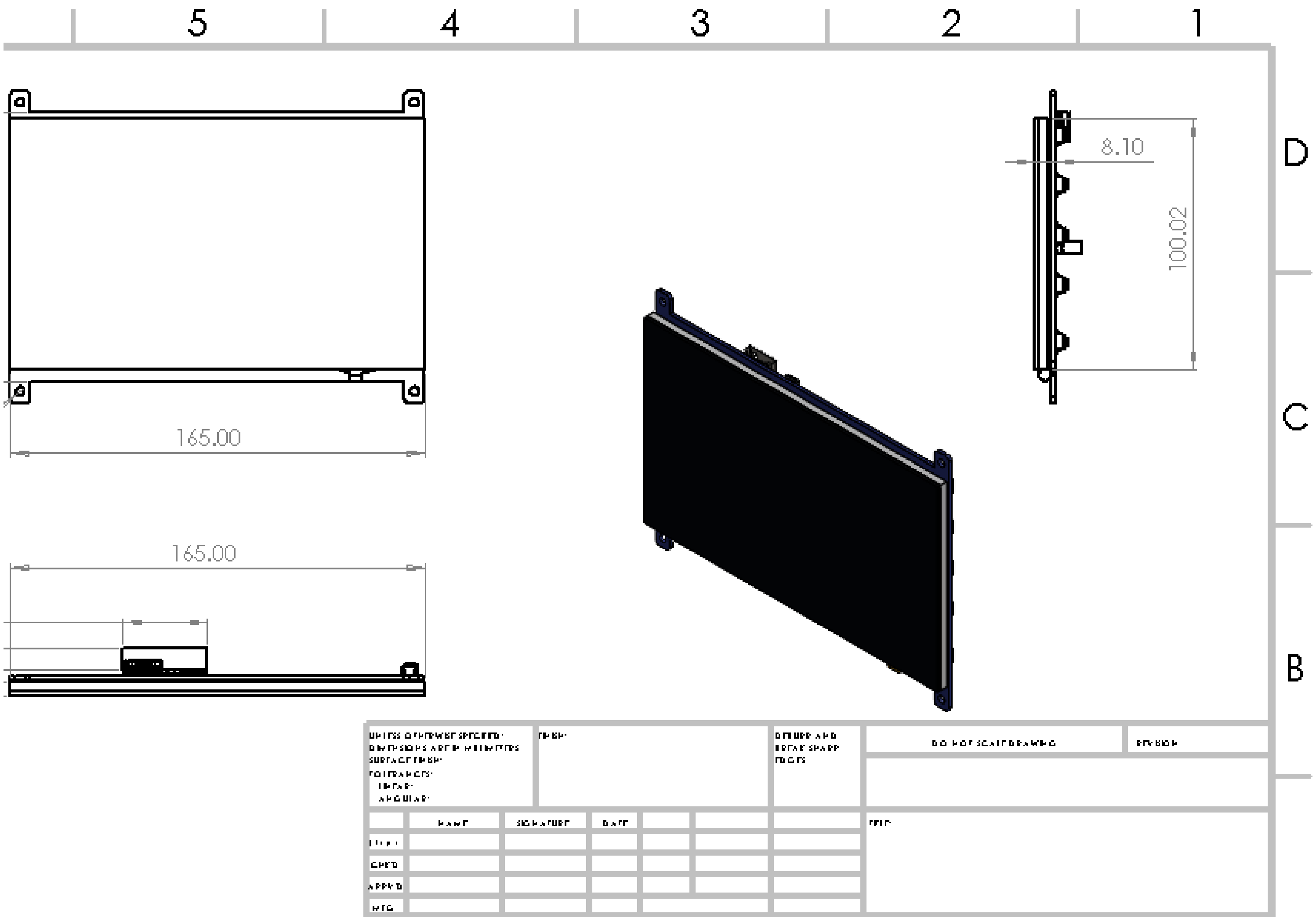

Figure 28 LCD screen plate

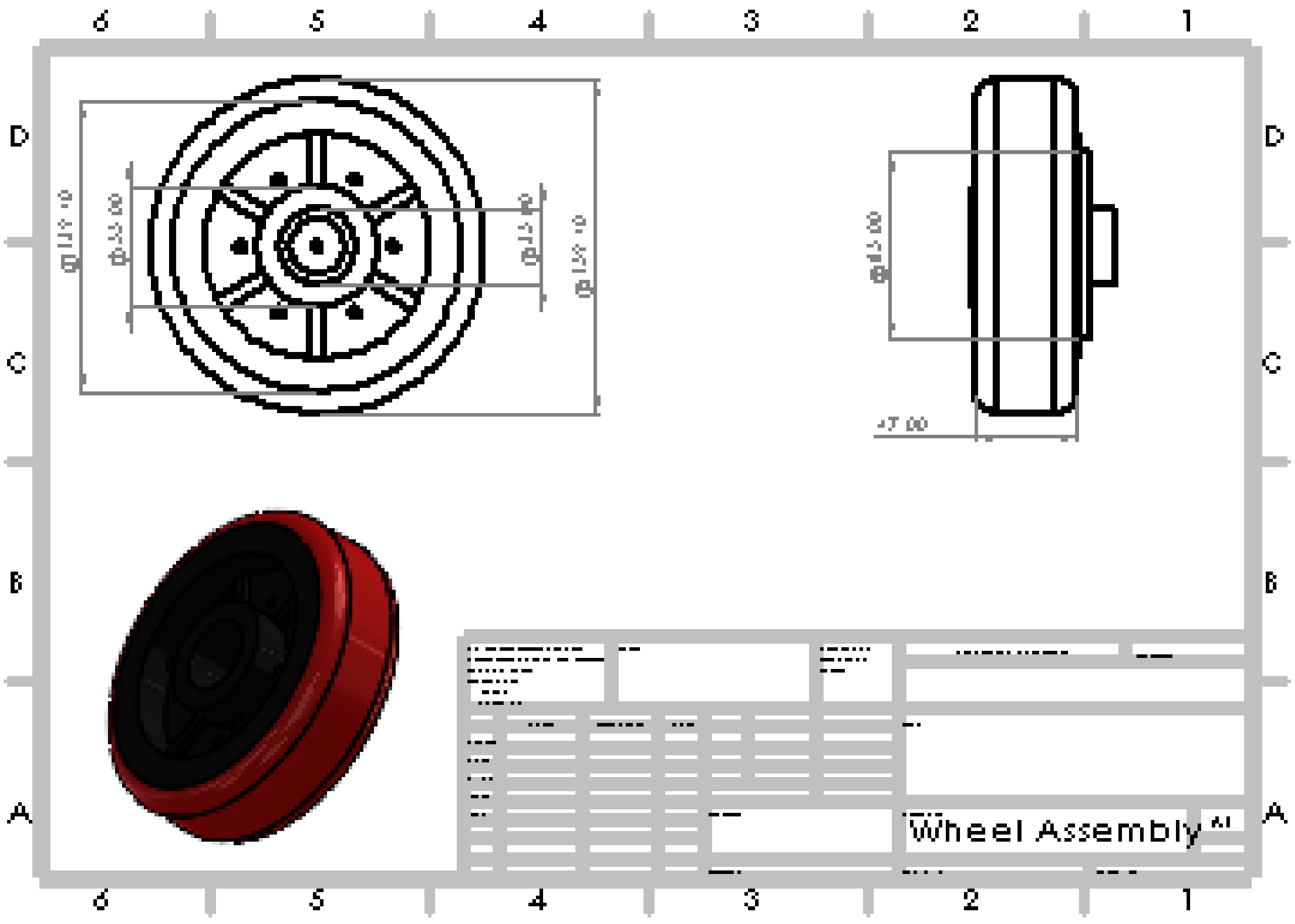

Figure 29 Castor Wheel draft

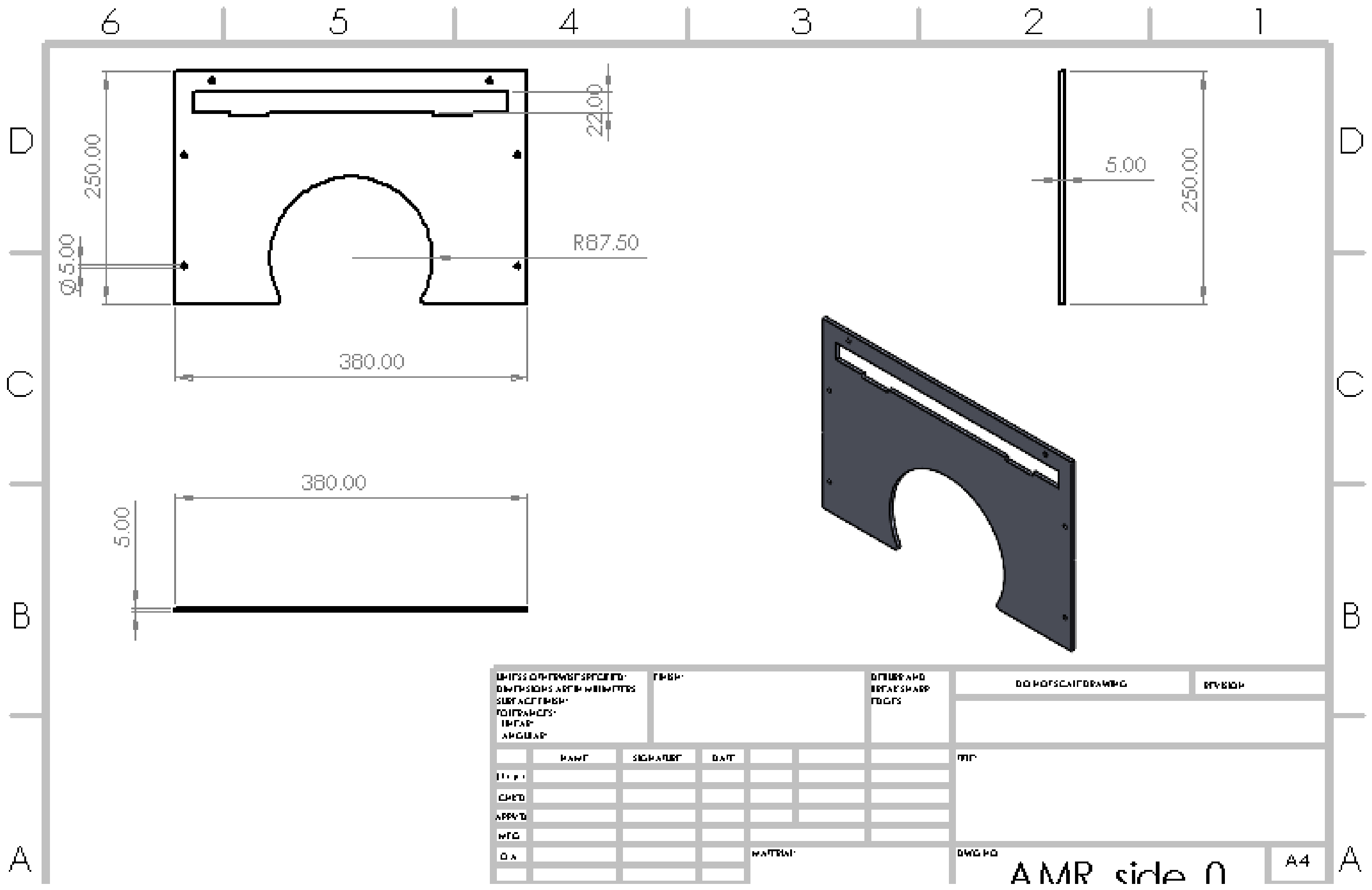

Figure 30 AMR Side plate draft

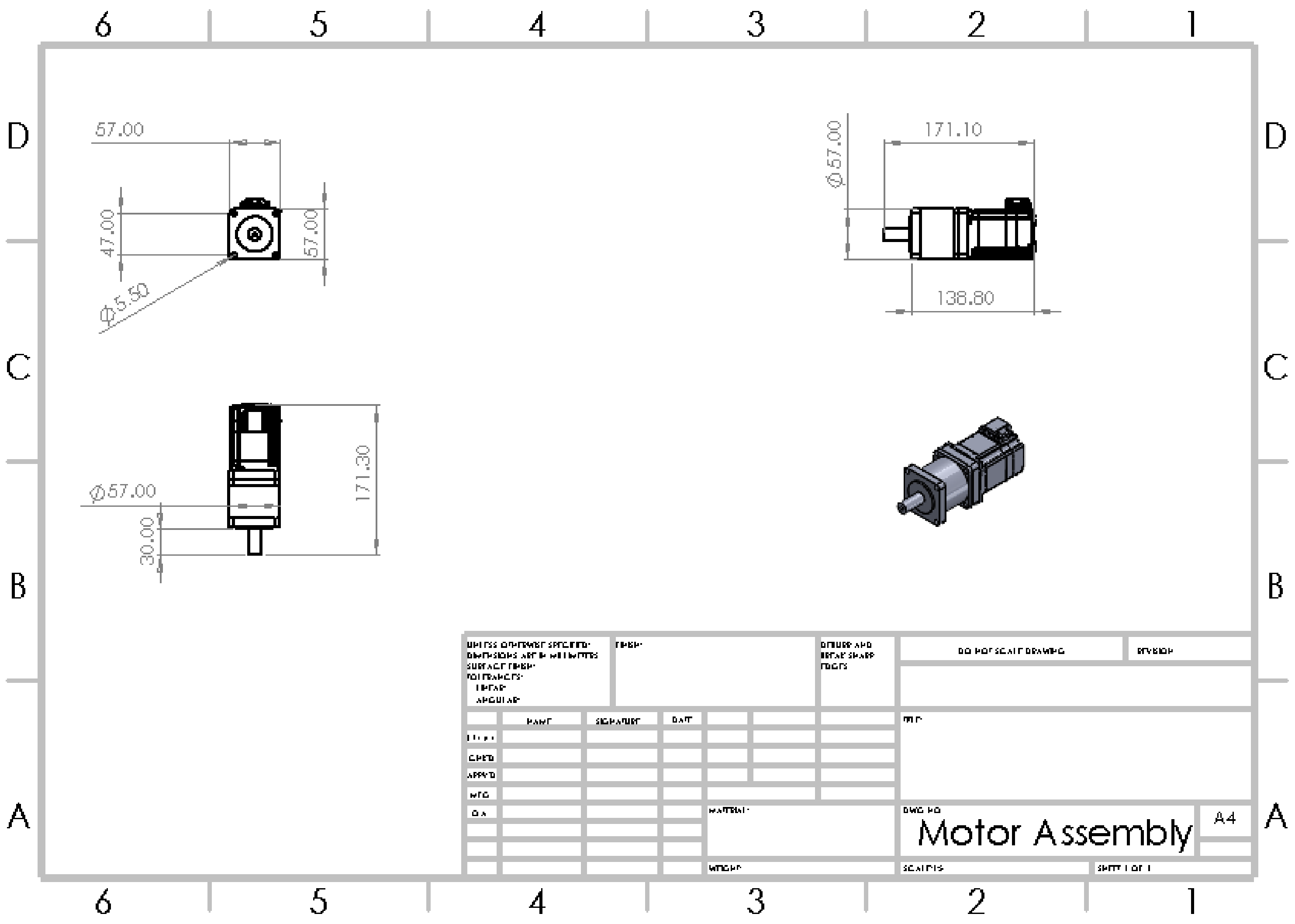

Figure 31 Motor draft