DE-43 (MTS) UMAIR AKRAM, WASEEM IRFAN, MOHAMED NALIM, SAAD AHMAD

MODULAR SELF RECONFIGURABLE ROBOT

COLLEGE OF
ELECTRICAL AND MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY
RAWALPINDI
2025

COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING

DE-43 MTS
PROJECT REPORT

MODULAR SELF RECONFIGURABLE ROBOT

Submitted to the Department of Mechatronics Engineering
in partial fulfillment of the requirements
for the degree of
Bachelor of Engineering
in
Mechatronics
2025

Sponsoring DS: **Submitted By:**

ACKNOWLEDGMENTS

ABSTRACT

Conventional robotic systems have transformed different fields by doing outstandingly well in specialized tasks under constant conditions. They are not adaptable when exposed to different tasks or changing environments, such as in home automation or remote operations. A modular self-reconfigurable robotic system overcomes this shortcoming by providing flexibility and adaptability through reconfiguration with the same robotic modules and task-specific extensions, such as robotic gripper or arm. Our Project is proof of concept for such a modular robotic system. It comprises autonomously reconfigurable robot modules that can form user-defined 3D configurations. To provide practical application proof, the system includes a gripper extension. The robot modules consist of a cylindrical, differential-drive mobile base with a magnetic ring sandwiched between wheels. This free-locomoting and magnetic capabilities are made possible through this design. Spur-gear-like wheels not only allow robust docking, but also permit the rolling motion of a module over another module. Localization employs beacon-based UWB technology in combination with IMU sensors. There is a software-developed graphical user interface, which provides instantaneous positions of modules and permits giving commands to have specific 3D arrangements.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Latin Letters

A	acceleration
M	mass

Greek Letters

α	incidence angle
Ω	resistance

Acronyms

SOP	Standard Operating Procedures

Chapter 1 – INTRODUCTION

1.1. Definition

The term "modularity" typically describes the design approach of creating several separate, functional parts first, then combining them to form a whole system. Modular Self-Reconfigurable Robots are a group of independent joints and link modules that may be put together to create different robot configurations for a range of tasks [24]. MSRR have the ability to change their geometry according to the task requirement. These Robots are often compared to LEGO bricks, which can be combined in various ways to create different structures. However, unlike traditional LEGO bricks that require manual rearrangement, these robotic modules have the ability to autonomously rearrange themselves. Each module is a fully functional unit capable of joining with others to form different shapes or configurations, much like LEGO, but with the added intelligence and mobility to reconfigure on their own to adapt to specific tasks or environments.[1]

1.2. Motivation

The motivation stems from the transformative potential of modular robotics. Traditionally, robots are designed as specialized machines limited to perform specific tasks. In contrast, modular self-reconfigurable robots present a groundbreaking innovation, where robots can autonomously reconfigure themselves to handle a wide range of tasks across diverse domains. From space exploration to industrial automation and consumer products, these robots offer miraculous flexibility, adaptability, and scalability [2]. Moreover, modular robots address two critical needs in real world applications: a higher utilization factor and self-healing capabilities [3]. Unlike traditional robots, which often require manual intervention or replacement when components fail, modular systems can adapt by reorganizing themselves. This ability to reconfigure and self-repair makes modular robotics a more versatile and sustainable solution for the ever-growing demands of modern industries. This research, therefore, seeks to explore and advance the development of these systems to unlock their potential in addressing the challenges of future applications.

1.3. Problem Statement

Despite the growing interest in modular robotics, the field continues to face major obstacles that hinder its progress. These challenges span across platform availability, hardware limitations, and algorithmic complexity, all of which must be addressed to enable practical deployment and scalability.

1.3.1. Platform Limitation

The field of modular robotics faces a significant challenge due to the limited availability of platforms for research and development. Unlike swarm robotics, where researchers can easily access various platforms, modular robotics lacks similar options. This scarcity hinders experimentation and innovation, slowing down progress in the field. Developing accessible platforms is essential for advancing modular robotic systems and unlocking their full potential.

1.3.2. Hardware Design

The main hardware challenges in modular robotic systems can be summarized as limitations in both mechanical and electronic design. These systems need to address multiple factors to ensure optimal performance, but current designs haven't yet achieved a universal solution.

On one hand, the mechanical and electrical strength and precision of the docking interfaces between modules are critical but still limited. This affects how reliably the modules can connect and function together in different environments. On the other hand, the power and motion capabilities of individual modules are constrained, impacting how precisely and efficiently they can move, including their torque and power output. Furthermore, the overall flexibility of the robot is restricted by the dexterity of each module, which limits the robot's ability to perform complex tasks.

Thus, while various designs have been proposed, these fundamental constraints continue to limit the development of an all-encompassing, versatile modular robotic system.

1.3.3. Planning and Control Challenges

While algorithms [4] have been created to manage millions of units under ideal conditions, scalability challenges persist in both low-level control and high-level planning when addressing real-world limitations such as:

- Parallel algorithm to coordinate large scale Motion
- Efficient communication between modules
- Algorithm for handling failures
- Algorithm for efficient reconfiguration

1.4. Application Areas

Although the system holds promise for performing a wide range of tasks, identifying a definitive "killer application" remains challenging.

1.4.1. Search & Rescue

Disaster Management is one of the key applications of modular systems,[5] particularly in search and rescue operations within collapsed buildings or mines. In these scenarios, people are often trapped in hard-to-reach areas where access is severely restricted due to structural damage. And so, a need for a robotic system flexible enough to physically reach and endure the harsh environments is strongly felt.

1.4.2. Space Exploration

Another application is space exploration. Long-term space missions need a self-sustaining robotic system capable of handling unexpected situations and performing self-repairs [6]. These robots can carry out complex tasks without requiring direct involvement from astronauts.

At present, the cost of sending each kilogram of payload into orbit runs into several hundred dollars. This creates significant limitations on the availability of spare parts and the number of devices that can be launched into space. Instead of launching multiple specialized tools, modular robots can perform various tasks by rearranging themselves, maximizing functionality while minimizing payload weight and cost. This flexibility makes them ideal for long-term space missions.

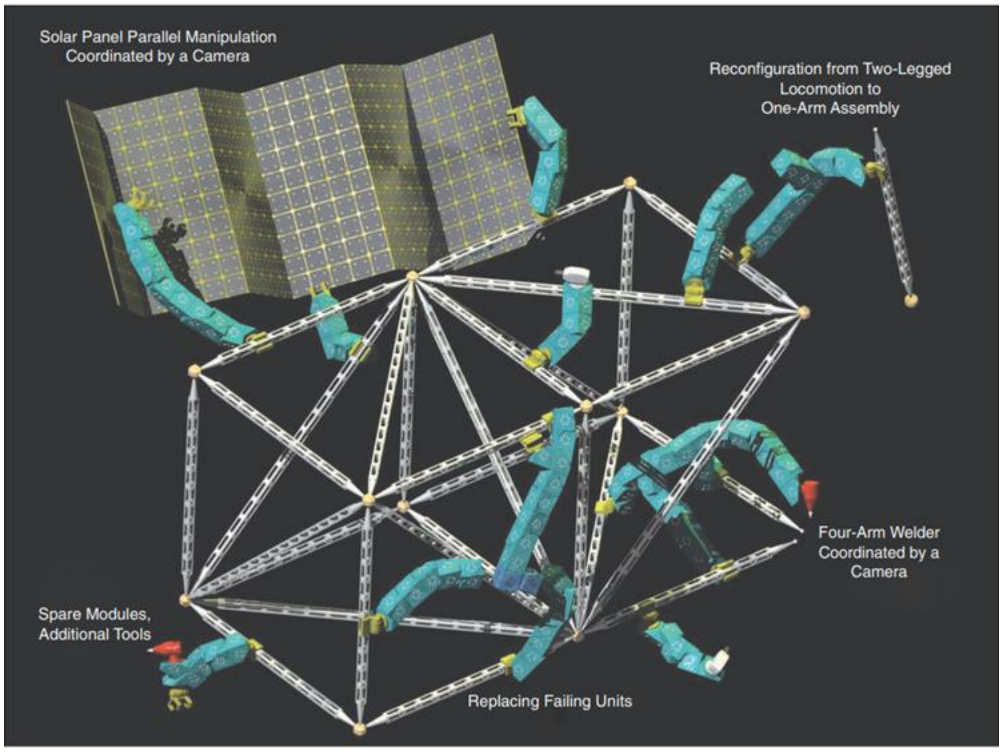

Solar Panel Parallel Manipulation Coordinated by a Camera
Reconfiguration from Two-Legged Locomotion to One-Arm Assembly
Four-Arm Welder Coordinated by a Camera
Spare Modules, Additional Tools
Replacing Failing Units

Figure 1. *Illustrates a truss-building colony of chain/tree robots made up of cubical modules. These modules are arranged into different shapes and structures to perform various tasks, such as assembling components, working together to manipulate objects, and conducting self-repair*

1.4.3. Medical Devices

Another field where the MSRR systems have great potential is medical applications. Modular systems, like the one shown in Figure 2, have significant potential in medical applications. These systems consist of smaller, interchangeable components that can be assembled or reconfigured to suit specific needs. For example, the robotic assembly in the image mimics the human spinal column and can assist with spinal movement. This type of system can replicate complex body movements such as flexion, lateral flexion, and compression, which are crucial for restoring mobility after spinal injuries [7].

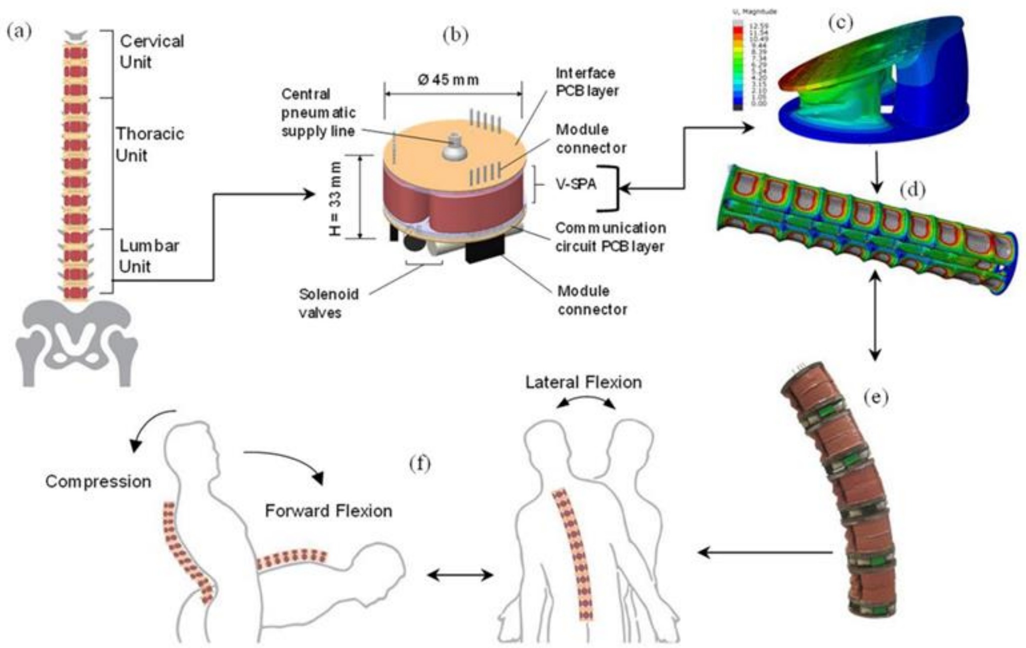


Figure 2. *(a). Human Spinal Column Representation (b). Soft Pneumatic Actuator Module (c). Numerical Simulation for V-SPA Module (d). Simulation of Module Assembly (e). Robotic Spine Assembly Bending (f). Spinal Movement Representation with Robotic Assistance*

1.5. Brief History

Modular self-reconfigurable robots (MSRRs) began as a concept in the 1950s when John von Neumann introduced the idea of a "universal automaton" that could change its structure. Although no physical robots were built from this idea, it greatly influenced future developments. Later, Lionel Penrose proposed a model for self-replicating machines which could form different structures, but research in this field slowed down for some time. It wasn't until the 1970s, with advances in electronics, that interest in reconfigurable systems re-emerged.

In the late 1980s, researchers Toshio Fukuda and Seiya Nakagawa developed the first practical system for modular robots, called the Dynamically Reconfigurable Robotic System (DRRS). This system was made up of many smaller units that could change their configuration based on the tasks they needed to perform. Following this, Fukuda and his team introduced another system known as the Cellular Robotic System (CEBOT), which laid the groundwork for more advanced modular robots. Since then, many designs have emerged, allowing robots to change their shapes and functions to suit different environments and challenges [13].

Chapter 2 – LITERATURE REVIEW

2.1. Classification of Modular Robotic systems

Each of the modular systems developed till date falls into one or more modular system categories. These categories represent a system's ability to form certain structures and exhibit certain behaviors. Therefore, it is important to understand the broad categories of modular systems, which would ultimately provide us clarity while reviewing and grasping the working of any given modular robotic system [21], [28]:

2.1.1. Chain

Modular systems capable of forming linear or branching structures. These systems have a string like morphology which allows them to make flexible configurations like snakes, quadreps, loops etc. However, they lack the ability to form lattice based 3-D structures due to lack of connection surfaces and/or module's DOF.

2.1.2. Lattice

Systems whose modules can reside together in 3-D grids. Lattice position for each module can be clearly defined. 3-D space is divided into discrete lattice regions during system modeling, which makes system analysis easier. These systems can form complex structures and has four sub-types:

- **Translational Lattice:** System in which the modules can only move in a direction perpendicular to the lattice directions since they only have translational joints. Modules are always oriented in the same way, which keeps their axes from rotating off-lattice.
- **Rotational Lattice:** Modules can be rotated between lattice places and can change orientation due to the presence of at least one rotating joint. Since all hybrid systems can create chains of modules that reside off-lattice through rotation, they all fall into this category.
- **Morphable Lattice:** Modules can be brought together without having to relocate to an adjacent grid location by allowing them to change the geometry of the lattice they would ordinarily make.
- **Fixed Lattice:** Modules need mobility or an outside force—such as a user or a stochastic process—to move them across grid positions since they lack a method for doing so.

2.1.3. Self Mobile

Modular System whose modules are individually capable of locomotion. This enhances the self-reconfigurability of a modular system since each module is capable of reaching a certain location both within the lattice and outside.

2.1.4. Hybrid

A system which combines traits from several of the aforementioned categories. The composite fusion of two or more types of features, such as lattice-type, chain-type, and mobile-type, is its main characteristic. As was previously noted, lattice-type, chain-type, and mobile-type MRRs all have unique benefits; hybrid MRRs, on the other hand, combine these features to provide flexibility, mobility, and scalability.

2.2. Relevant Modular Robotic Systems

Over the last three decades, several modular systems have been developed. However, a few of them have been found to have noticeable outcomes and hence, are mentioned below. These systems obey one or more of the following criteria:

1. The modular system's unit module must be of a common 3-D shape or its variation e.g. cube, sphere, double cube. This makes it easy to derive configurations to attain various 3-D structures.
2. The total degree of freedom of a module must allow it to make complex 3-D structures.
3. The modular system should have hybrid capabilities i.e. should fall under various modular system types.
4. The connection mechanism of the system must allow its modules to self reconfigure without any human intervention.
5. The modular system must be homogenous and may have extensions which are meant to perform a specific task.

2.2.1. RoomBots

A single Roombot (RB) module developed by Sproewitz et al. is composed of four interconnected hemispheres, which can continuously rotate around each other through the use of three motor units, as shown in Figure 3.

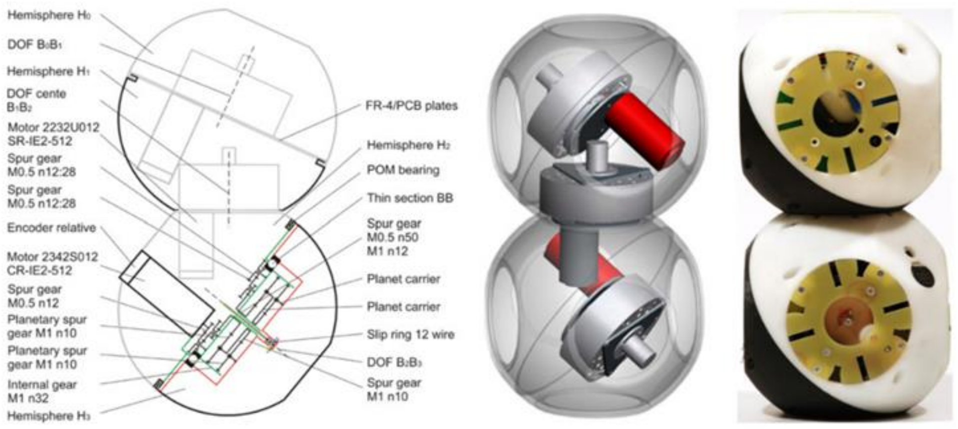


Figure 3. *Single Module of RoomBots*

Each module features 10 connection plates, two of which are equipped with actively retractable hooks, allowing them to latch onto other connection plates. This Active Connection Mechanism (ACM) enables the modules to attach to one another or to any surface fitted with connection plates, forming dynamic multi-module structures, such as adaptive furniture. Each module is fully autonomous, containing all the necessary electronics for operation, including a Bluetooth module for communication and a battery for power. Remarkably, Roombots are one of the few systems capable of 3D reconfiguration with up to 10 modules, achieving an impressive 30 degrees of freedom (DoF).[8]

The Roombots project envisions a future where furniture is no longer static. Instead, adaptive furniture systems can change their shape and functionality over time. For example, Roombots could transform a simple table into a chair or adjust the height of a surface based on the user’s needs, all while performing complex tasks in the background.

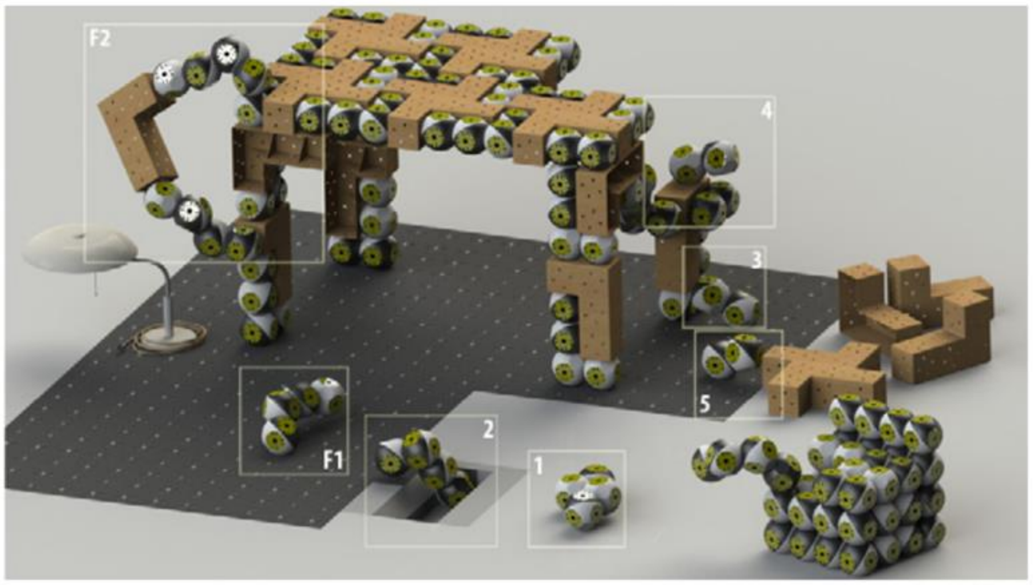


Figure 4. *Mobile Furniture [9]*

2.2.2. M-Tran III

The main characteristics of the M-TRAN system include a simple design of the module mechanism, easy connection and wide variety of possible three-dimensional configurations. An M-TRAN module [10] consists of two blocks connected by a link as shown in Figure 5. Both blocks are in the same shape, made up of half cube and half cylinder. Both blocks can rotate independently by ±90° via a motor embedded with the link around their axes.

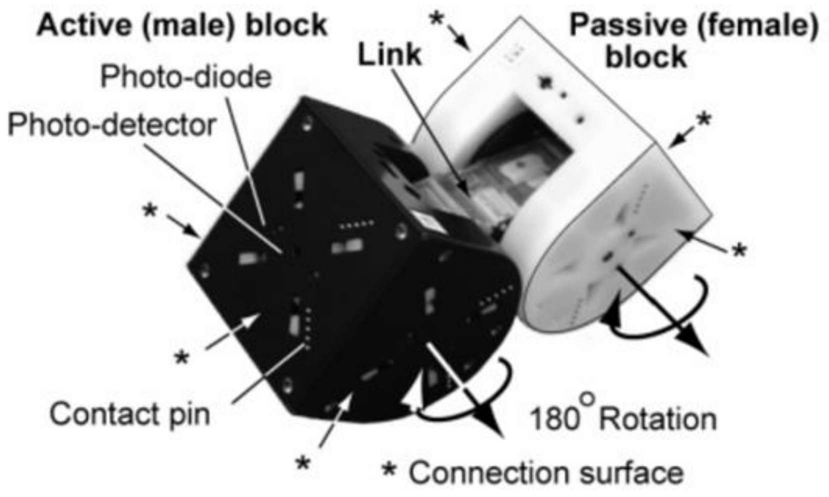


***Figure 5**. M-TRAN III module design. Each module has two actuated axes and six connection surfaces*

The two blocks in M-TRAN III are categorized as male and female. Blocks of opposite gender from two modules can connect with each other using three flat surfaces. Although these surfaces have different shapes, they are made to connect evenly in four different ways. Each module controls the connection process on its own.

In previous designs, i.e M-TRAN I , M-TRAN II, there were four permanent magnets on each of the connecting surfaces of the semi-cylindrical part for connection mechanism[11] shown in Figure 6.

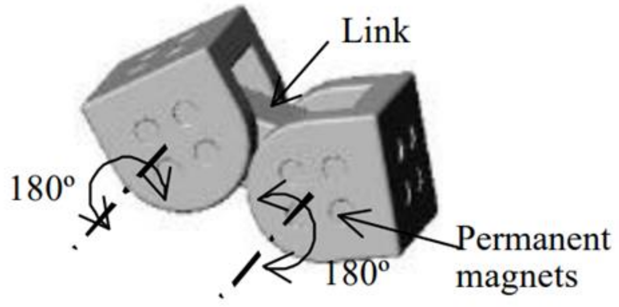
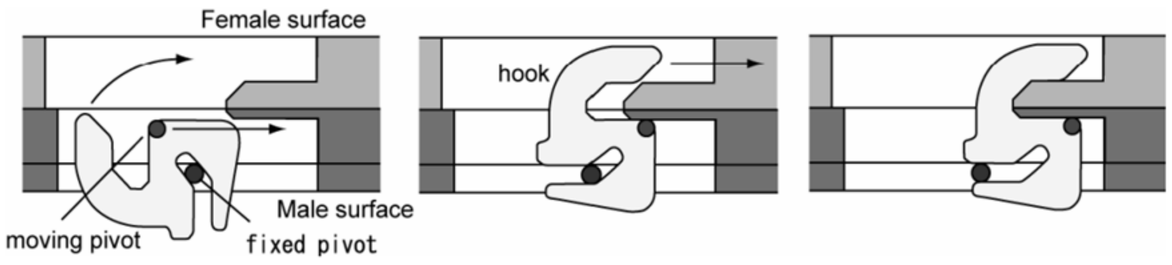


***Figure 6**. Schematic View of the M-TRAN I/II module*

The M-TRAN III connection mechanism is based on the latched connector as shown in Figure 7. This makes a faster connection as compared to the previous versions of M-TRAN. Connection and disconnection processes are completed within 5s, whereas those by the former magnetic connection often required more than a minute.



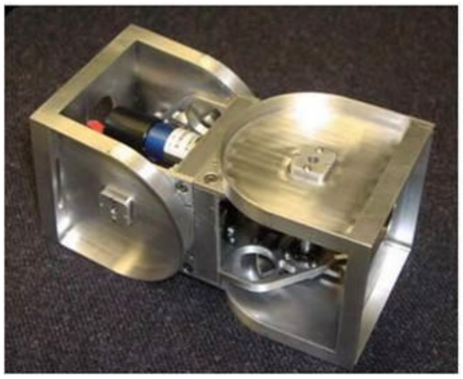
(a) (b) (c)

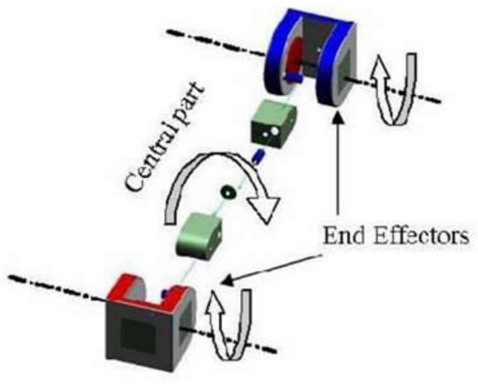
Figure 7. *Hook motion of connection mechanism in M-TRAN III: (a) retracted (b) connecting and (c) connected*

2.2.3. SuperBot:

Developed by B. Salemi et al., SuperBot module [14] has a 3 DOF double cube structure with 6 connection surfaces. The Modular system was designed to keep in view the need to perform remote tasks that require self-assembly and reconfiguration during space exploration programs. Therefore, the modules were created in such a way that they withstand harsh environments, perform locomotion and have enough torque to manipulate multiple neighboring modules.

Each SuperBot module consists of a link which joins the two cubes. Revolute joint at both ends of the link makes it possible for each cube to rotate ±90° relative to the link. Moreover, both of the rotational freedoms have their axis parallel to each other. This makes the system similar to that of the M-TRAN III. However, the SuperBot module has an additional rotational freedom that allows one cube to rotate relative to the other, with its axis being perpendicular to that of the other two rotational freedoms. This allows greater locomotion possibilities, such as using a primitive wheel or making two rotational axes perpendicular for pan and tilt operations.





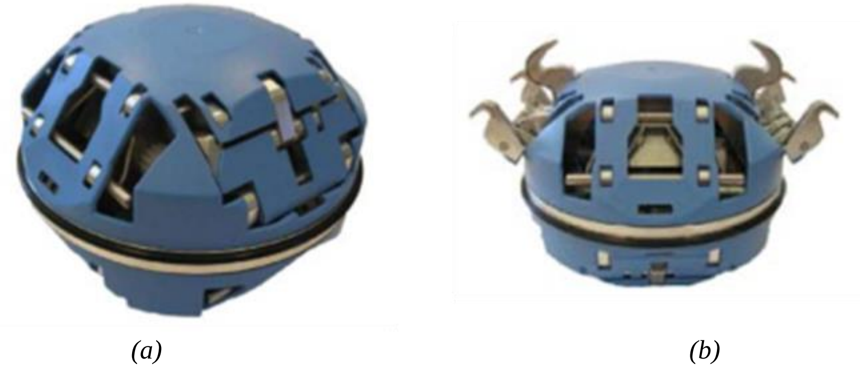
(a) (b)

Figure 8. *(a) A SuperBot module and its (b) exploded view with all 3 DOF*

To ensure effective power distribution between modules, power sharing is made possible between the connected modules. Inter-module communication is carried out via high-speed infrared LEDs. The system's reliability in real environments is yet to be established as limited self-reconfiguration experiments have been conducted on the system.

2.2.4. ATRON:

It is a rotational lattice based self-reconfiguration robotic system [15]-[17]. Each module has 1 DOF and has an approximate shape of a sphere. The sphere is divided into two hemispheres, with continuous rotational freedom between the two. Each hemisphere consists of 2 male connectors and two female connectors. In total, a single ATRON module has 8 connection points.



(a) *(b)*

Figure 9. *The ATRON module (a) in its initial state and (b) its active connectors extended*

For each hemisphere, all the 4 connectors are situated at a 45° angle to the rotational axis in a symmetrical manner, as shown in Figure 10. The male connector is made up of three hooks which grab onto the passive connector, which is made up of two stainless steel bars. The actuation time to make or break a connection between modules is 12.5, which is considered to be relatively slow. Each connector has an IR-proximity sensor and IR neighbour-to neighbour communication, and the module has a tilt-sensor. A major drawback of the system is that while it is able to complex 3-D structures, its module does not have individual mobility. This complicates the process of self-configurability.

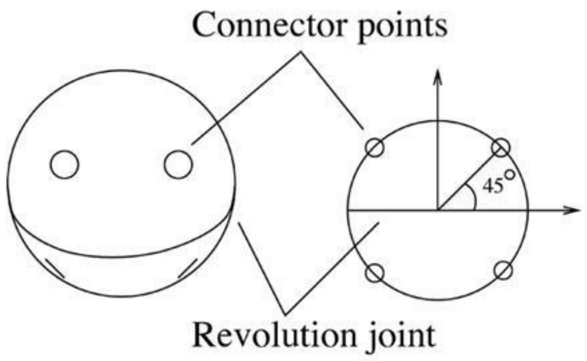


Figure 10. *Placement of the connector points relative to the revolution joint for the ATRON system*

2.2.5. 3D M-Blocks

M-Blocks are modular robots that can self-assemble to form different structures. Their internal structures allow them to completely move on their own. A unique aspect of the system's module is the presence of a single spinning mass which is completely inside the module. Momentum is created by applying a brake to an internal flywheel. This allows the module to move in any direction with a single actuator. M-Blocks can identify each other using a barcode system on each face, which allows them to communicate in order to accomplish simple tasks like following the path of an arrow. Magnets on each face give them the ability to attach to each other. Blocks can travel along other blocks and climb up or climb down their neighbors and even jump as well. Very few modular robots have the capability to jump.

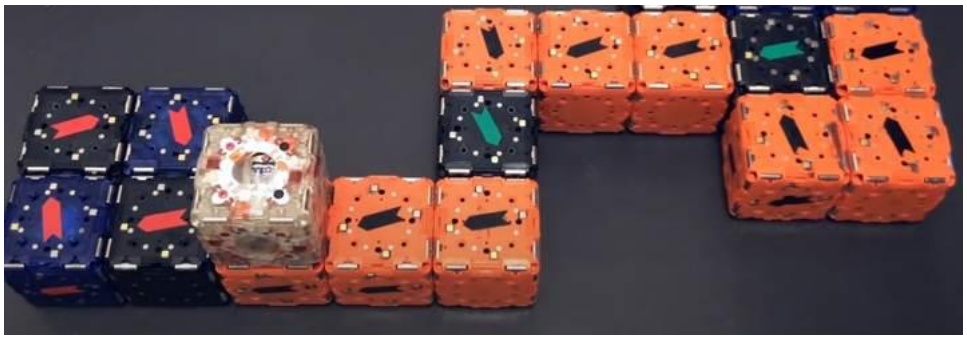


Figure 11. *M-Block following the path in the direction of arrows*

Each M-Block module is a 50 mm cubic frame milled from a single piece of 7075 aluminum, weighing 143 g. The module, including the flywheel, has a moment of inertia of 63.0E-6 kg m^2 about its center of mass. This frame [20] holds twenty-four cylindrical bonding magnets along its twelve edges, allowing the modules to connect and align with each other.

In the future, larger swarms of M-Block will be able to build bigger and more capable structures.

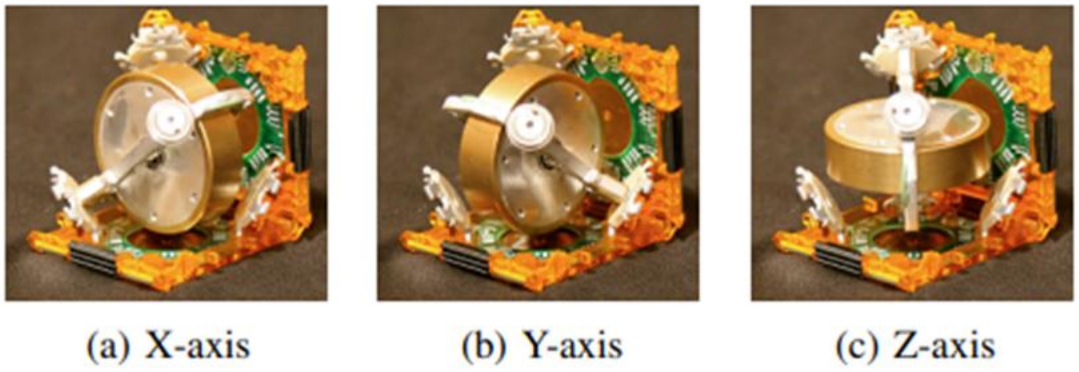


Figure 12. *Rotating Assembly of 3D M-Blocks*

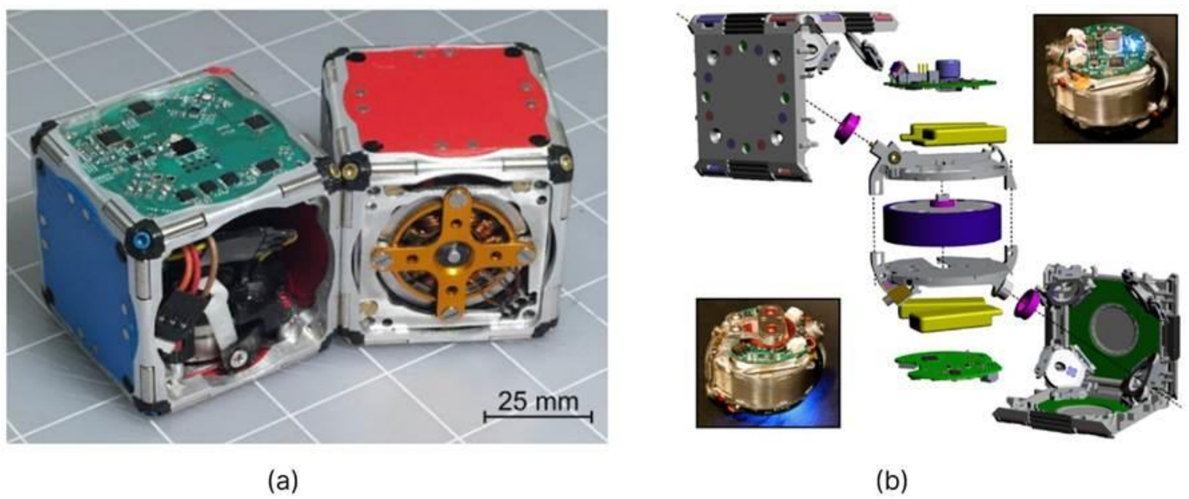


***Figure 13**. (a). Demonstrating magnetic bonding with adjacent modules to form a cubic lattice structure (b). Exploded view of a 3D M-Block showing its molded frame (gray), magnets (red/blue), batteries (yellow), circuit boards (green), and flywheel (purple). Ball bearings (pink) support rotation. Insets show the brake assembly (bottom-left) and main PCB (top-right) [12]*

2.2.6. iMobot

Developed by Ryland et al., iMobot [18] is a multifunctional modular system capable of achieving chain structures and performing individual locomotion. The Module is a double cube structure with 6 passive connection surfaces. It is a 4 DOF unit, two of which rotate each cube and the other two rotate the faceplates on the far end of each cube.

Figure 14 shows an iMobot module whose front cube is tilted upwards. The square plate with fillet edges, joined to the front cube facing upwards, is the faceplate. These plates have a continuous rotational degree of freedom and help the module locomote, provided both of them are placed perpendicular to the same surface. A cylindrical rod can be seen in between the cubes. On each end of the rod, a motor is mounted which rotates the cube relative to the rod. They can rotate the cubes 180° relative to each other. The modules have a manual connection mechanism which is a drawback since the system cannot self-reconfigure. Each of the 6 module connection faces has 4 through holes and an equal number of threaded holes. These holes have to be screwed in order for a module to connect to its counterpart.

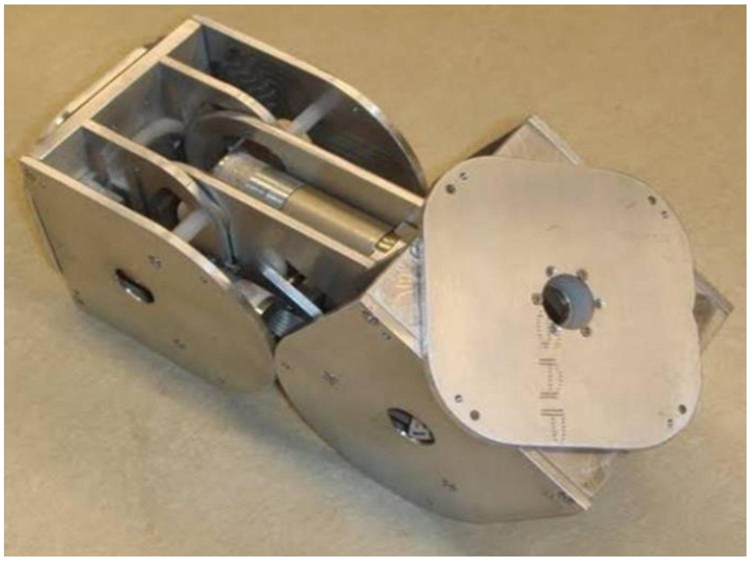


***Figure 14**. An iMobot module with the front cube tilted upwards and faceplate rotated*

Other than the locomotion provided by the faceplates, the module is capable of crawling motion with the help of inner joints. It also has the ability to attain vertical positions by taking advantage of the joint's hindrance in its rotational freedom due to lack of ground clearance. In this case, an inner joint can be seen lifting the other cube not connected to it. Similarly, the faceplate joint pans the upright body when placed flat on the ground.

2.2.7. SMORES

Self-assembling Modular Robot for Extreme Shapeshifting (SMORES) [19], developed by Davey et al., is a hybrid system with numerous capabilities. The system consists of homogeneous modules, each having 4 connection surfaces. Each module is a 4 DOF unit, two of which form a drive mechanism. Figure 15 shows the rotational freedoms each module has, one of which being used to perform tilt operations of ±90° (DOF#4).

The modules have a magneto-mechanical bi-gendered connection mechanism which allows them to connect/disconnect with other modules effectively. Each connection surface has a pair of north (red circles) and south (green circles) pole permanent magnets, as seen in the figure. The two modules' connection surfaces align in such a way that opposite magnet poles are facing each other. As a result, the surfaces attract and join each other. A square slit can be seen at the center of each connection surface. A docking key is inserted in this slit which makes the process of disconnecting the modules easier.

The modular system can make both chain and lattice structures. Unlike HyMod, no electrical connections are formed when modules connect to each other. Consequently, no power sharing and wired communication between the connected modules is possible.

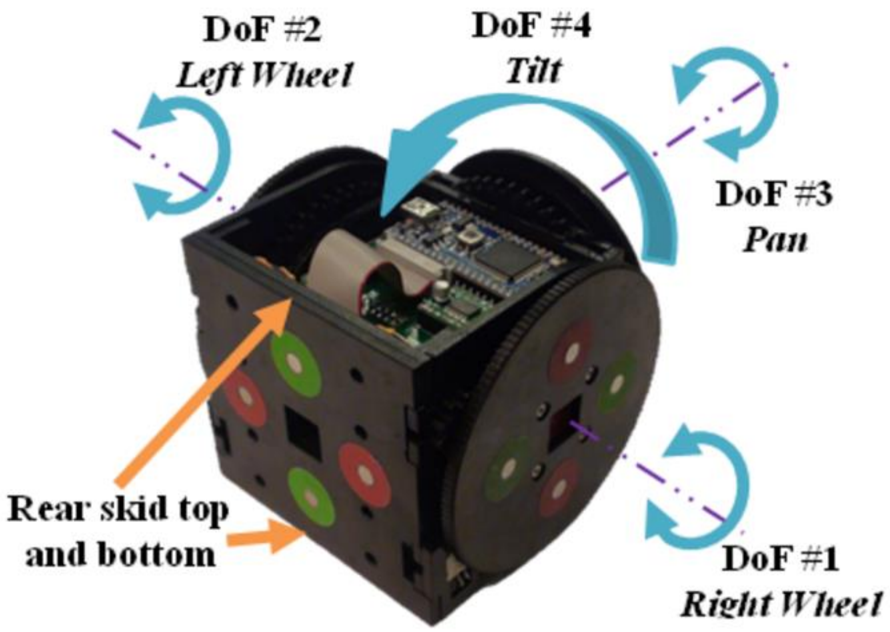


Figure 15. *Four Degrees of Freedom of a SMORES movement module*

2.2.8. HyMod

A combination where modular systems can achieve chain and lattice structures as well as modules with individual locomotion capabilities, is hard to find in the majority of recognized modular systems. Only a handful of systems such as SMORES, FreeBot, SnailBot etc. can form complex structures with modules having drive mechanisms.

The HyMod system [21], [22] by Parrott et al. covers all these features. It is a Hybrid system capable of achieving chain and Lattice structures. Each module is a 3-DOF spherical unit with 4 genderless connectors. The unit's individual locomotion comes from its dedicated drive mechanism, which consists of two wheels. Each wheel's rotational movement accounts for a degree of freedom. HiGen, a high-speed genderless connector, has been integrated as connection mechanisms between the modules.

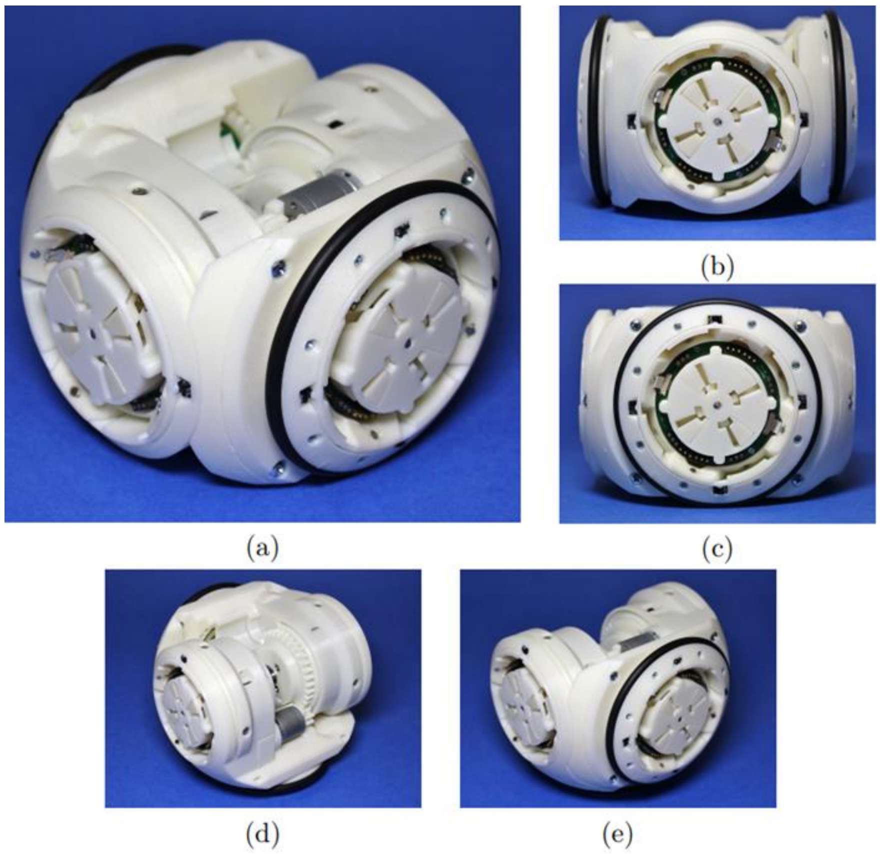


Figure 16. *HyMod unit’s (a) Isometric view (b) Front view (c) Side view. Isometric view with central rotational angle at (d)-90° (e) +90°*

The module’s central rotation, whose rotation axis is perpendicular to that of the drive mechanism, constitutes as the third degree of freedom. The physical structure of the module is divided into two parts joined together . Each part has two connectors, of which one has a wheel attached to it. The two parts, on connection, form a hinge joint which can rotate an angle of ±90°. In other words, one part can rotate ±90° relative to the other part. This gives the ability for a module to perform tilt/rotate operations with other connected modules.

HiGen connectors [23] offer two useful features to the system. Firstly, it has a genderless hook to hook connection mechanism which allows for single sided disconnect. Secondly it performs connect/disconnect operation in a very short time. The average actuation time is 0.2s, which is faster than that of ATRON, RoomBots, SMORES and M-TRAN III. The connectors also ensure electrical connectivity between modules, enabling communication and power sharing.

2.2.9. FreeBot

Developed by Liang et al., this modular system [25] is a major breakthrough in terms of the connection mechanism between the modules. Each FreeBot module is a 2 DOF sphere which connects to other modules without any dedicated connector regions. In other words, any point of a sphere’s surface can become a point of connection. This makes the FreeBot system different from the previous modular systems as it doesnt need to plan trajectories in order for modules' connectors to align properly and connect to each other.

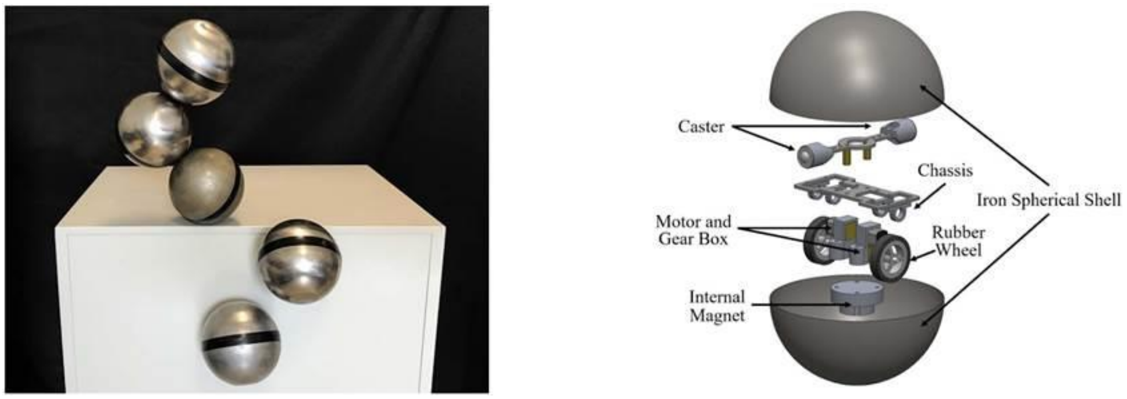


Figure 17. *(a) FreeBot Modular system (b) Exploded view of the FreeBot Module*

The module is made up of two hemispheres which are made up of Iron. Internal components include a two-wheeled internal driving vehicle, a permanent magnet and two caster wheels, as shown in Figure 17. The permanent magnet is connected at the bottom of the internal vehicle but does not have a physical connection to the spherical shell. The caster wheels touch the ends of the sphere's internal surface, and both are placed on the same strip-shaped aluminum alloy plate. They are responsible for the overall stability of the internal vehicle. The driving mechanism is responsible for the module's individual locomotion and the movement of the magnet inside the sphere.

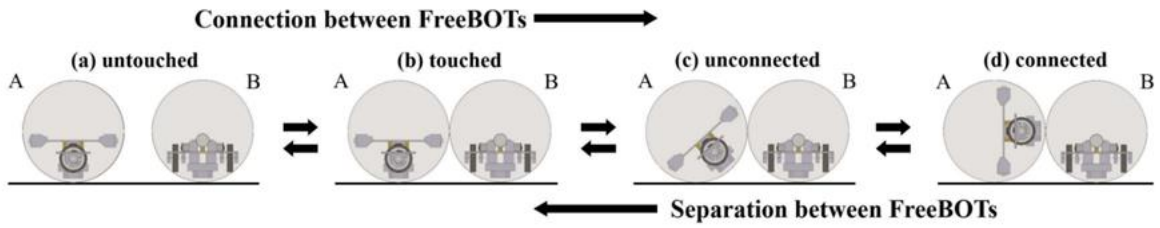


Figure 18. *Connection and separation between FreeBOTs*

Connection between two modules happens when one module's internal permanent magnet faces the other module's ferromagnetic body. In order for the magnet to face the other module, the internal vehicle's wheels move in such a way that it moves inside the sphere without causing it to locomote externally. Since the magnet is attached beneath the vehicle, it also moves and changes its location and direction in the process. The vehicle's internal locomotion is only possible if the external body is in contact with that of the other module. Disconnection of the modules is done in a similar way where the internal vehicle moves internally such that the magnet no longer faces the other module. Both processes can be visually observed in Figure 18.

Successful experiments on the system have shown that it is able to perform locomotion on walls as well as climb stairs, thanks to its genderless magnetic connection mechanism which is fault tolerant and instant.

2.2.10. SnailBot

This modular system, like FreeBot, also utilizes the Freeform connection mechanism via use of permanent magnets. Developed by Zhao et al., a SnailBot [26] module's shape resembles that of the body of a snail, with a solid ferromagnetic spherical shell joined with a six-wheel rocker chassis at the bottom, as shown in the Figure 19. Two permanent magnets are embedded in the chassis and are used for connection to other modules.

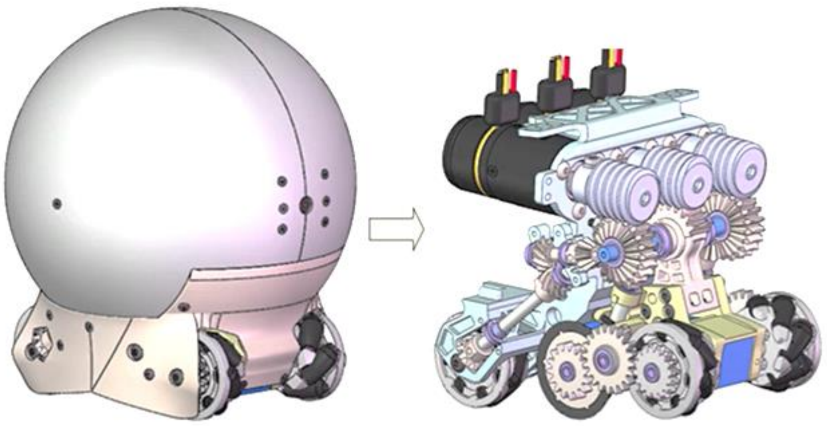


Figure 19. *CAD model of (a) A SnailBot Module and (b) its internal drive mechanism*

Figure 20 shows how a SnailBot connects to the other module, moves over its shell, eventually disconnects after getting off the shell and continues locomoting on a flat surface. The connection mechanism is somewhat similar to that of the FreeBot system. Permanent magnets residing in the chassis attract the ferromagnetic shell of the other module and thus a connection is formed between the two modules when brought together.

The rocker chassis is a parallel type coaxial type differential transmission system. It has three motors and each motor that powers the wheels can drive three wheels on one side. On each side, the first and the last wheels are mecanum wheels whereas the middle tyres are ordinary wheels. The flexibility in the rocker chassis allows the module to move flawlessly over the shell or any other uneven surface since it offers relatively more wheel to surface contact compared to a conventional chassis structure.

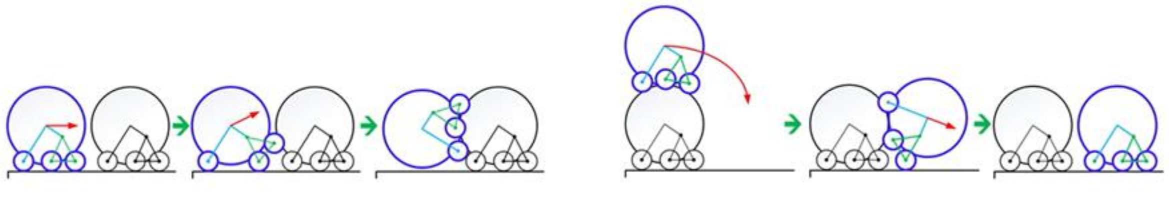


Figure20. *A step-by-Step process of connection and separation of a SnailBot module from its counterparts*

Since it is a recently developed modular system, further improvements can be made to the system which includes building a perception system for each module, improving self-reconfiguration algorithms and testing the overall system in complex environments.

2.2.11 FreeSN

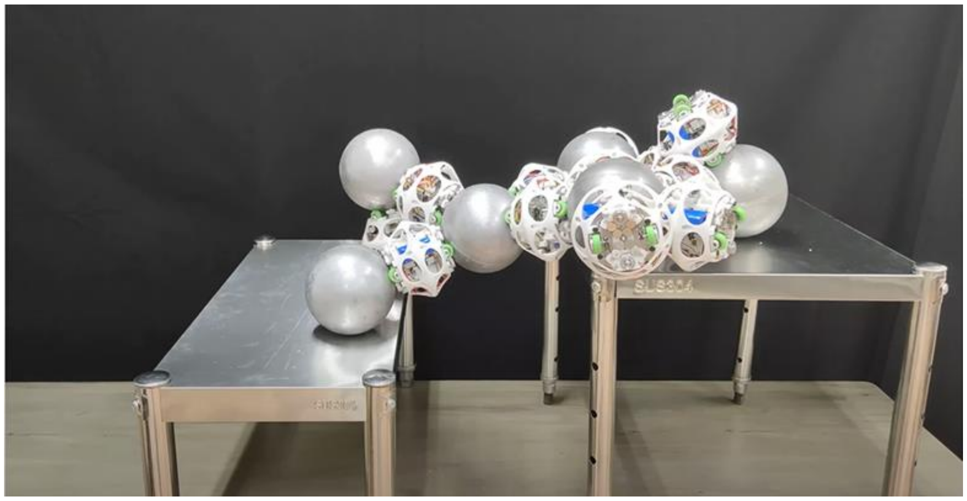


Figure 21. *A FreeSN Modular system*

It is a heterogenous self-reconfigurable modular system with FreeForm connection mechanism. The system consists of two types of modules: Strut and Node [27]. A node module is simply a low carbon steel spherical shell whereas a strut module is a mobile body which consists of two magnetic freeform connectors. Each freeform connector consists of a permanent magnet array, magnet lifting mechanism and a two wheel differential driver. The permanent magnet is responsible for the strut module's connection to a node module's shell. Magnet lifting mechanism varies the magnetic force of the magnets while the driver ensures the locomotion of the strut module. The freeform connectors are placed on top of each other, with one of them being inverted.

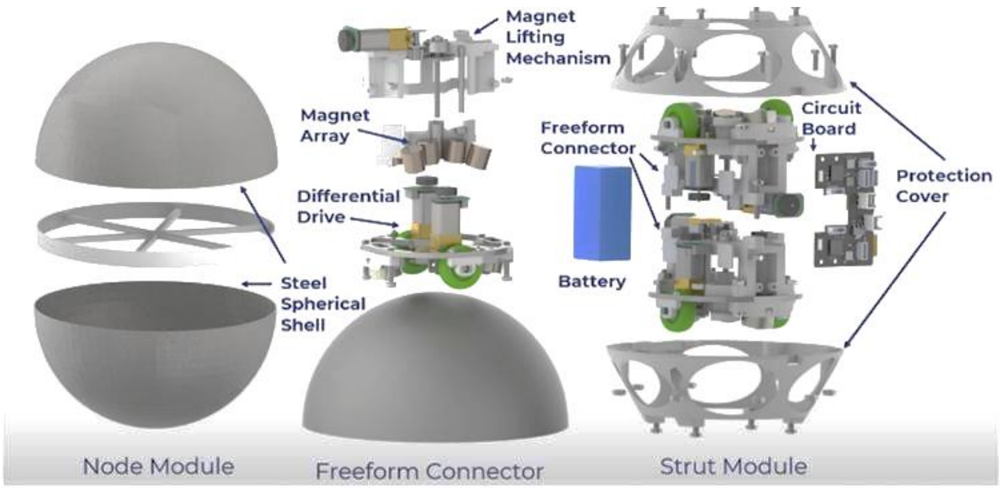


Figure 22. *Exploded view of the Node, Strut and Freeform Connector*

As a result, the strut module is able to manipulate the node module in order to attain the required system configuration. Numerous successful experiments were performed on the FreeSN system, which included assembly formation, obstacle crossing, transportation and object manipulation.

2.2.12. Datom

The Datom system is a deformable modular robot system designed for building self-reconfigurable programmable matter, as introduced by Gilpin et al. [29, 30]. Each Datom module has a cubic structure and uses electromagnets for connection, making it different from spherical systems like FreeBot and SnailBot. The modules can attach to any of their six faces, enabling them to self-assemble into complex 3D shapes and structures, which is crucial for applications such as dynamic environments and adaptive systems.

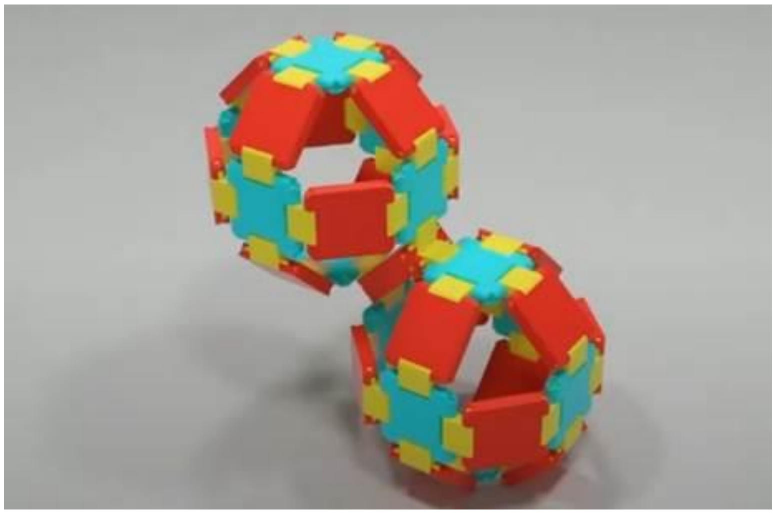


***Figure 23**. A Datom Modular system*

Each Datom contains embedded electronics that manage the control of the electromagnets, enabling precise attachment or detachment between modules. This modularity provides flexibility for tasks such as constructing bridges, or repairing structures. The system also aims to explore how programmable matter can change shape or structure autonomously, a capability that is highly relevant for various industrial and exploration applications.

However, challenges remain, particularly in improving control algorithms for efficient self-reconfiguration. Successful experiments have demonstrated the potential of Datom in dynamic structural applications, but further work is needed to address energy efficiency and enhance the system's performance in unstructured environments [29,30].

System	Dimension	Self-Reconfiguration	Chain	Lattice	Self-Mobile	Heterogenous
RoomBots	3D	Yes	Yes	Rotational	No	No

M-Tran III	3D	Yes	Yes	Rotational	No	No
SuperBot	3D	Yes	Yes	Rotational	No	No
ATRON	3D	Yes	Yes	Rotational	No	No
3-D M-Blocks	3D	Yes	Yes	Rotational	No	No
iMobot	3D	No	Yes	None	Yes	No
SMORES	3D	Yes	Yes	Rotational	Yes	No
HyMod	3D	Yes	Yes	Rotational	Yes	Yes
FreeBot	3D	Yes	Yes	Rotational	Yes	No
SnailBot	3D	Yes	Yes	Rotational	Yes	No
FreeSN	3D	Yes	Yes	Rotational	Yes	Yes
Datom	3D	Yes	Yes	-	Yes	No

Table 1. *Classifications of the various modular systems covered in this review*

Chapter 3: METHODOLOGY

3.1. System Design Approach

3.1.1. Design Motivation & Evolution

The primary objective of this project was to develop a modular self-reconfigurable robot (MSRR) that can function both independently and collaboratively, adapting its configuration to perform various tasks. The inspiration for the initial design came from FreeBOT architecture, a spherical modular robot capable of omnidirectional docking. This design concept aimed to allow the robot to connect at any point on its surface, enabling it to perform complex reconfiguration and collective behaviors in dynamic environments.

In the early conceptual phase, the robot was designed to be enclosed in a 5-inch diameter metallic spherical shell made of mild steel. The internal chassis was 3D-printed using PLA, housing motors, sensors, batteries, and control electronics. Rubber-coated wheels driven by servo motors were mounted inside, resting against the inner surface of the spherical shell. This configuration allowed the internal chassis to drive along the shell's interior, causing the entire sphere to roll externally—a mechanism inspired by ball bots and similar robotic toys.

However, through practical prototyping and testing, several limitations emerged that made the spherical design less viable:

- Material sourcing difficulty: High-precision, symmetric metallic shells with adequate strength and low weight were difficult to acquire.
- Power inefficiency: Significant torque was lost due to friction and weight of the bot.
- Heavy magnet requirements: Ensuring stable magnetic docking required very strong neodymium magnets, which increased both weight and cost.
- Design uncertainty: Mechanical tolerances and wiring complexity added risk to the project's reproducibility and reliability.

Due to these challenges, the design was re-evaluated and evolved into a cylindrical robot module. This cylindrical structure provided a stable chassis, allowed direct drive on flat surfaces, and reduced reliance on passive rolling. Unlike the spherical version, the cylindrical module could use standard differential drive principles and was better suited for modular connection, component integration, and power efficiency. The design retained modularity by allowing magnetic docking on either end of the cylinder, while simplifying the construction and control system.

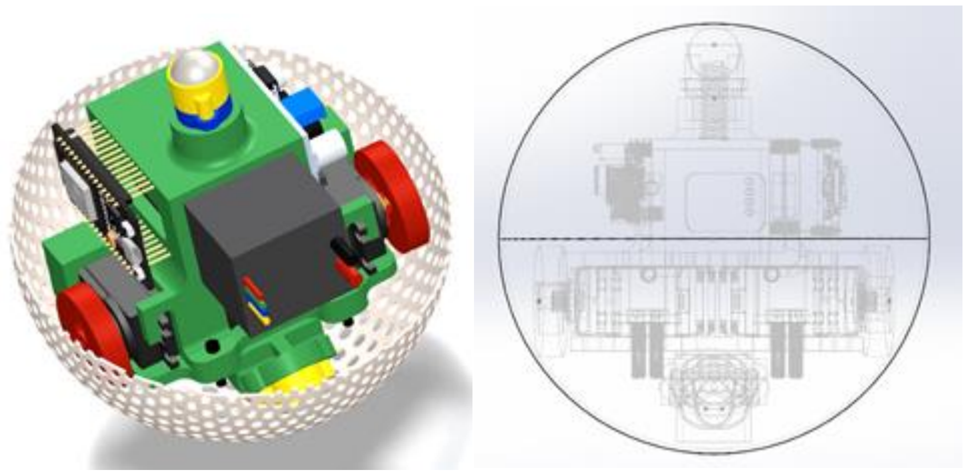


Figure 24. *Initial Spherical Design Concept Diagram*

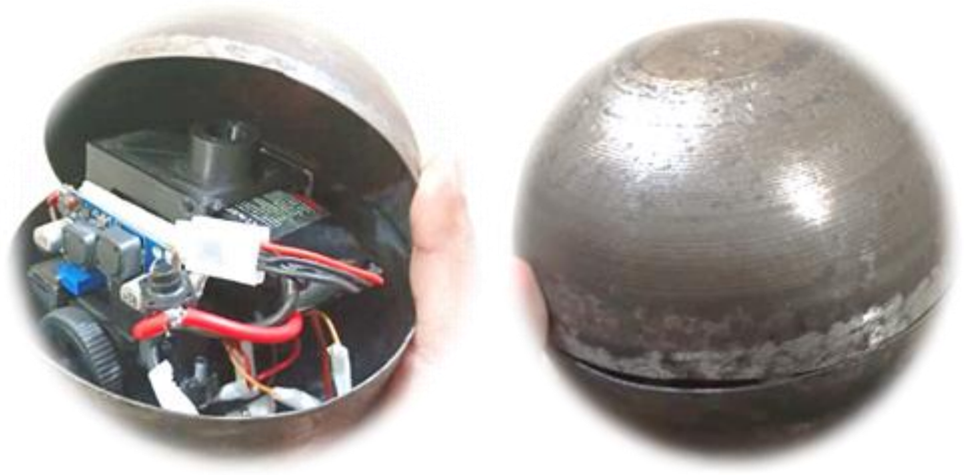


Figure 25. *Prototype of Spherical Version*

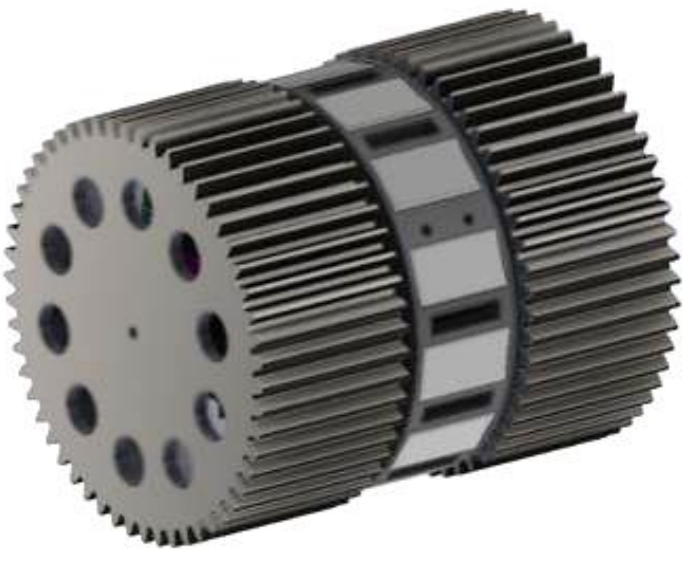


Figure 26. *Final Cylindrical Design CAD View*

Feature	Spherical Design	Cylindrical Design
Motion Control	Complex, indirect	Simple, direct
Stability	Low	Moderate
Manufacturing Complexity	High	Low
Power Efficiency	High	Moderate to High
Docking Surface	Omnidirectional	Axial

***Table 2**. Comparison Table (Spherical vs Cylindrical Design)*

3.1.2. System Requirements

The goal of this project was to design and prototype a modular self-reconfigurable robot system that could function both as an independent mobile unit and as part of a physically connected group. Unlike traditional swarm robotics, which relies on distributed control and indirect cooperation, our system aims for physical docking between modules to achieve functional reconfiguration. Each module is capable of forward and backward movement and in-place rotation, allowing it to navigate a 2D plane and align itself with other modules for task-based collaboration.

The system was designed with the intention that each module:

- Operates autonomously or through centralized control.
- Supports differential drive movement for navigation and positioning.
- Includes magnetic docking capability to form physically connected shapes.
- Performs basic reconfiguration tasks, such as forming lines, T-shapes, or transport structures.
- Can be remotely controlled through a wireless user interface, allowing non-technical users to assign tasks easily.
- Is built using affordable, readily available components to keep production costs low and enable scalability.

The target size of each robot module was limited to approximately 10 cm × 10 cm, with a target weight under 500 grams. These constraints ensured compactness, efficient power usage, and the ability to handle inter-module connection without requiring heavy structural supports or high-torque motors.

A key design motivation was the inefficiency of traditional robotics systems, where each robot is purpose-built for a single function, leading to:

- High acquisition and maintenance costs
- Limited reuse across different applications
- Increased e-waste due to early obsolescence
- Difficulty of use for non-technical personnel
- Accessibility barriers in startups or developing countries

In contrast, the proposed modular approach introduces the idea that "one robot can do many tasks". By enabling a robot to change form and function based on user requirements, the same hardware modules can be reused across different operations, industries, and environments.

Environmental and Application-Specific Requirements:
The system was intended for indoor and semi-structured environments such as:

- Manufacturing floors in small-scale industries
- Healthcare settings for mobility assistance or sensor-based monitoring
- Research and educational institutions
- Disaster-response simulations (e.g., line-following bots to locate items)
- Space-constrained environments such as space labs or defense equipment transport

Key environmental design factors included:

- Operation on smooth or mildly uneven surfaces
- Tolerance for minor collisions or misalignment during docking
- Minimal power consumption, suitable for LiPo battery operation

Summary of Core System Requirements:

Requirement Type	Description
Functionality	Forward/backward movement, rotation, magnetic docking
Modularity	Docking and undocking on a single axis; task-specific formations
Size & Weight	≤ 10 × 10 cm footprint, ≤ 500g
Power	Operates on 7.4V LiPo battery, efficient power regulation to 3.3V
User Control	Remote via mobile/web GUI; easy for non-technical users
Adaptability	Scalable for use in industry, healthcare, defense, education

***Table 3**. Summary of Core System Requirements*

3.1.3. Functional Architecture

The functional architecture of the system describes how hardware and software elements integrate to allow modularity, coordination, and autonomy across all robot modules. Each module is an independent node, capable of communication, actuation, sensing, and decision-making. Together, they form a coordinated system through structured interaction protocols.

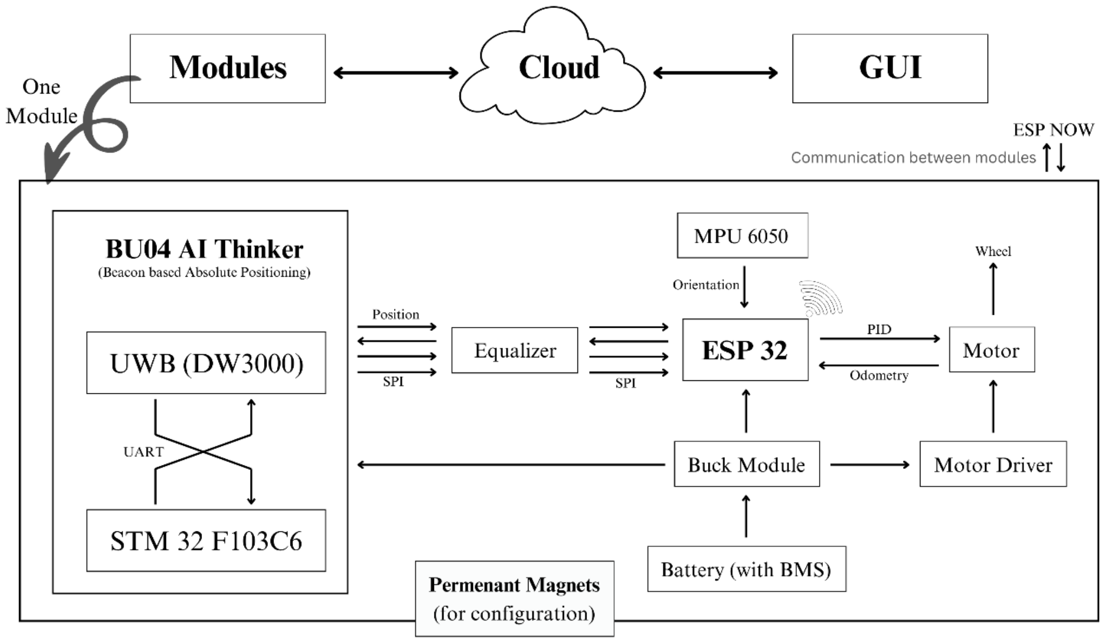


Figure 27. *Functional block diagram*

Hardware–Software Integration:

Each module is built around the ESP32 microcontroller, which runs the control logic, processes sensor inputs, and handles communication. The onboard MPU6050 IMU provides orientation data, BU04 provides positioning, while motor drivers (L9110S) are used to actuate the N20 encoder motors for movement. The software running on ESP32 handles:

- Real-time motion control (via PWM and feedback),
- Sensor data acquisition (from MPU6050),
- Communication with other modules and the GUI,

All of this is synchronized through Wi-Fi, creating a responsive and reconfigurable system architecture.

Modular Interaction Framework:

The framework enables modules to interact physically and digitally. Physical interaction is achieved via magnetic docking, while digital communication is maintained through Wi-fi signals. Once docked, data can flow between modules, making one act as the “host” and others as “extensions”.

Control commands are interpreted by individual ESP32 units, but centralized logic (in GUI) handles coordination and shape formation. This framework allows:

- Autonomous behavior, if required
- Team-based operations, where a master module assigns roles

Communication and Localization Logic:

The modules communicate using Wi-Fi-based ESP-NOW protocol for low-latency, peer-to-peer data exchange. For localization, the system integrates:

- MPU6050 for orientation and inertial tracking,
- Planned use of UWB modules (DW3000) for precise indoor positioning,

This multi-sensor fusion ensures that each module is aware of its position relative to others, enabling intelligent shape formation and coordination.

3.1.4. Technology Stack Overview

Microcontrollers, Sensors, and Actuators:

- Microcontroller: ESP32
  - Handles motion control, video streaming, communication.
- Sensors:
  - MPU6050 – 3-axis accelerometer + gyroscope for orientation and motion estimation
  - UWB Module – for indoor localization
- Actuators:
  - N20 DC encoder motors – for locomotion
  - SG90 servo motors – for extension arm movement
  - L9110S motor driver – for bidirectional motor control

Wireless Protocols:

- ESP-NOW (Wi-Fi-based peer-to-peer protocol) for fast and lightweight communication between modules without router dependency.
- Planned integration of Ultra-Wideband (UWB) for centimeter-level localization.
- Future option to use Bluetooth Low Energy (BLE) for short-range pairing or backup control.

Power Management System:

Each module is powered by a 7.4V 800mAh 2S Li-Po battery, regulated using:

- AMS1117 or MP1584 buck converters to supply stable 3.3V/5V outputs
- Current-limiting and voltage-sensing modules (for battery health)
- A compact BMS (Battery Management System) ensures safe charge-discharge cycles

3.2. Mechanical Design Methodology

3.2.1. Chassis Design

The mechanical chassis of the modular robot was carefully designed to ensure compactness, strength, stability, and manufacturability, while also supporting the unique cylindrical configuration of the robot system.

Material Selection: Entire internal chassis was modeled in SolidWorks and 3D printed using PLA

(Polylactic Acid) filament. PLA was selected primarily for its lightweight properties, good tensile strength, and ease of printing. Compared to materials like ABS or PETG, PLA provided the best balance between rigidity and print accuracy, which was essential for precisely aligning critical components such as the motor mounts, bearing housings, and magnetic docking slots.

Why PLA?

- Lightweight (helps meet the <500g weight constraint)
- High print quality (dimensional accuracy)
- Good mechanical strength for low-load applications
- Readily available and affordable
- Suitable for indoor prototypes where temperature resistance is not critical

The final weight of the fully assembled module is approximately 300g, and the chassis fits within a 10 cm (length) × 8 cm (diameter) envelope, keeping the robot compact and scalable.

3.2.1.1. Structural Layout – Cylindrical Configuration

The robot’s body is built in a cylindrical form factor, where two gear wheels are placed at either end of the chassis, driven by two N20 encoder motors in a differential drive configuration. This design ensures symmetry for better balance during docking.

One key design challenge was the absence of a third stabilizing wheel, which meant the robot could experience undesired rotational tipping (rolling sideways) during acceleration or docking. To mitigate this:

- The center of gravity (CG) of the entire system was positioned below the axis of wheel rotation. This ensures that gravitational forces contribute to the robot's self-righting behavior.
- Two high-quality bearings were mounted inside the chassis to allow rotational stability and reduce axial play.
- The internal components, including battery, motor driver, and ESP32-CAM, were arranged to maintain weight symmetry along the central axis.

This layout not only improved the physical stability of the robot during linear motion but also reduced unwanted drift or roll during turns.

3.2.1.2. 3D Modeling and Design Tools

All design components were created using SolidWorks CAD software, which allowed for:

- Parametric modeling for easy adjustment of dimensions
- Assembly simulation for checking fit and interference
- Center of gravity estimation for balance analysis

- Integration with 3D printing slicers for rapid prototyping

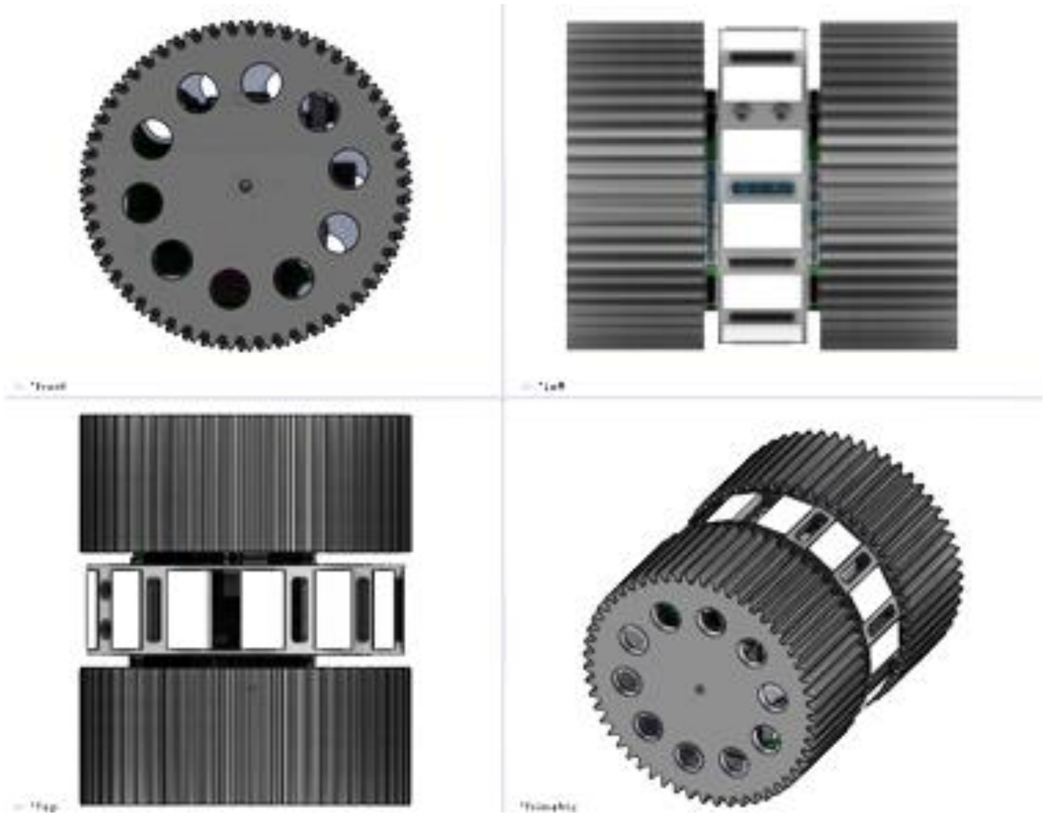


Figure 28. *SolidWorks Isometric View*

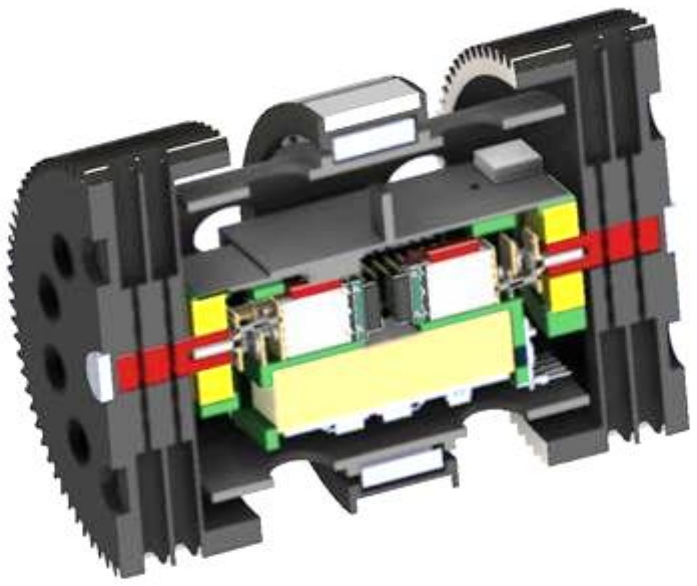


Figure 29. *Cross-Section of Robot*

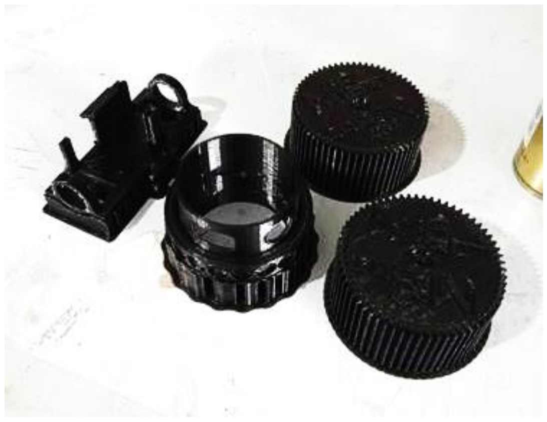


Figure 30. *Printed Chassis and wheels*

3.2.2. Mobility Mechanism

The mobility of the modular robot is achieved using a mobile differential drive system with two independently driven wheels, providing both translational and rotational motion in the 2D plane. The design is kept intentionally simple to reduce mechanical complexity and enable tighter integration within the compact cylindrical chassis.

3.2.2.1. Differential Drive System

The system uses two N20 DC gear motors with encoders mounted on either side of the cylindrical chassis. This configuration allows the robot to:

- Move forward when both wheels rotate in the same direction
- Rotate in place (yaw) when wheels rotate in opposite directions
- Execute curved paths by varying the relative wheel speeds

There is no free caster wheel in the design. Instead, the stability and balancing are passively managed by:

- Placing the center of gravity below the motor axis
- Ensuring the robot's speed remains low to minimize tipping

This design choice is deliberate to allow the robot to roll inside a spherical shell, and to keep structural symmetry.

3.2.2.2. Locomotion Geometry and Calculations

Differential Drive System:

The robot uses a basic differential drive configuration with two wheels on either side of a cylindrical chassis.

- Wheel Diameter (D) = 8.5 cm → Radius (R) = 4.25 cm
- Distance Between Wheels (L) = 10 cm
- Motor speed (both wheels) = 60 RPM (Revolutions per Minute)

Kinematic Equations of Differential Drive:
Let:

- v = Linear velocity of the robot
- ω = Angular velocity of the robot
- v_1 = Left wheel linear velocity
- v_2 = Right wheel linear velocity

The differential drive robot's motion can be described by:
1. Linear Velocity:

$$v = \frac{v_1 + v_2}{2}$$

2. Angular Velocity:

$$\omega = \frac{v_1 - v_2}{L}$$

Conversion from RPM to Linear Speed:
Each motor drives a wheel of radius R = 4.25 cm
At 60 RPM, the tangential (linear) speed of each wheel is calculated as:

$$v_{wheel} = \frac{2\pi R \times \text{RPM}}{60}$$

Substitute the values:

$$v_{wheel} = \frac{2\pi \times 4.25 \times 60}{60} = 2\pi \times 4.25 \approx 26.7 \text{ cm/s}$$

So, each wheel moves at 26.7 cm/s when the motor runs at 60 RPM.

Scenarios:
a) Straight Line Motion:

- $v_1 = v_2 = 26.7 \text{ cm/s}$
- $v = \frac{26.7+26.7}{2} = 26.7 \text{ cm/s}$
- $\omega = \frac{26.7-26.7}{10} = 0$

The robot moves straight with no rotation.

b) In-Place Rotation (Turn in place):

- $v_1 = -26.7 \text{ cm/s}, v_2 = 26.7 \text{ cm/s}$
- $v = 0$, so robot doesn't move forward
- $\omega = \frac{-26.7-26.7}{10} = -5.34 \text{ rad/s}$

This causes the robot to spin around its central vertical axis.

Interpretation:

- At 60 RPM, the robot's maximum forward speed is about 26.7 cm/s.
- The turning radius (R_{turn}) can be adjusted by varying the relative speeds of v_1 and v_2.
- The in-place rotation is achieved by running the wheels in opposite directions.

3.2.2.3. Wheel Design

Each wheel is custom-designed and 3D printed using PLA. Instead of smooth or rubber-coated wheels, we employed a geared interlock-style design. This offers:

- Increased traction and torque transfer on flat surfaces
- Simplified mechanical interlocking when modules connect
- Easy fabrication using low-cost 3D printing
- Material: PLA (chosen for availability, strength, and ease of printing)
- Design: Geared perimeter for better grip and inter-module compatibility
- Bearing: A standard bearing is embedded at the center of each wheel to ensure smooth rotation and minimize friction and rolling of body.

3.2.2.4. Support Mechanism

As noted earlier, the robot is built without a caster or passive support wheel to preserve its symmetric cylindrical form. Stability is maintained by:

- Careful weight distribution
- Use of internal bearings to support the inner frame inside the outer spherical shell (in earlier design)
- Designing the chassis such that the robot body passively aligns with the ground under gravity

While this increases reliance on precise balancing, it also reduces mechanical parts, which aligns with the design goal of making the robot compact, affordable, and mechanically minimal.

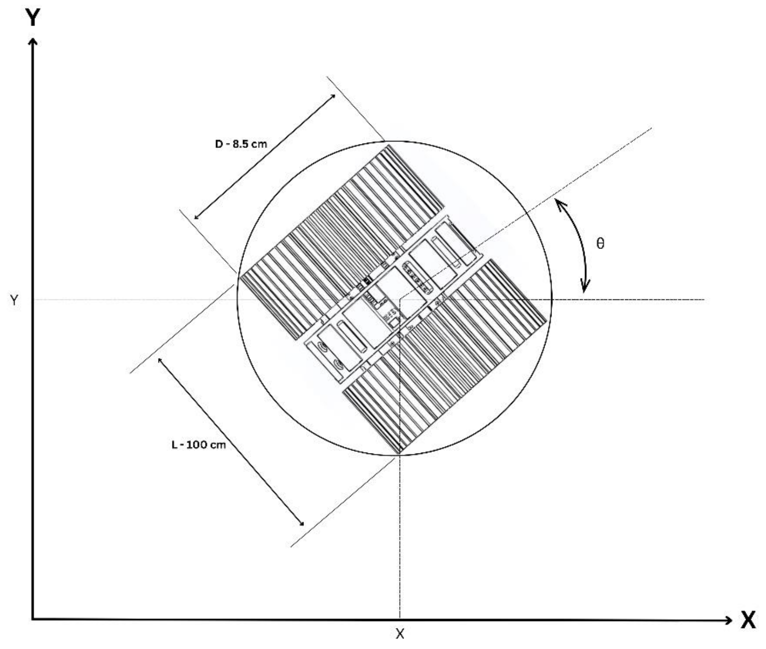


Figure 31. *Differential Drive Layout Diagram*

Parameter	Symbol	Value	Unit	Notes
Module	*m*	1.5	mm	Given
Number of Teeth	*z*	54	—	Given
Pressure Angle	*α*	20°	degrees	Given
Pitch Circle Diameter	*PCD*	81	mm	PCD=m×z
Addendum	*ha*	1.5	mm	ha=m
Dedendum	*hf*	1.875	mm	hf=1.25×m
Whole Depth	*h*	3.375	mm	h=ha+hf
Outer Diameter	*OD*	84	mm	OD=PCD+2×ha
Root Diameter	*RD*	77.25	mm	RD=PCD−2×hf
Circular Pitch	*p*	4.712	mm	p=π×m
Tooth Thickness	—	2.356	mm	Approx. p/2 at pitch circle
Hole Angular Spacing	—	45°	degrees	360°/8
Central Hole	—	Yes	—	For shaft or bearing

Table 4. *Table of geared wheel description*

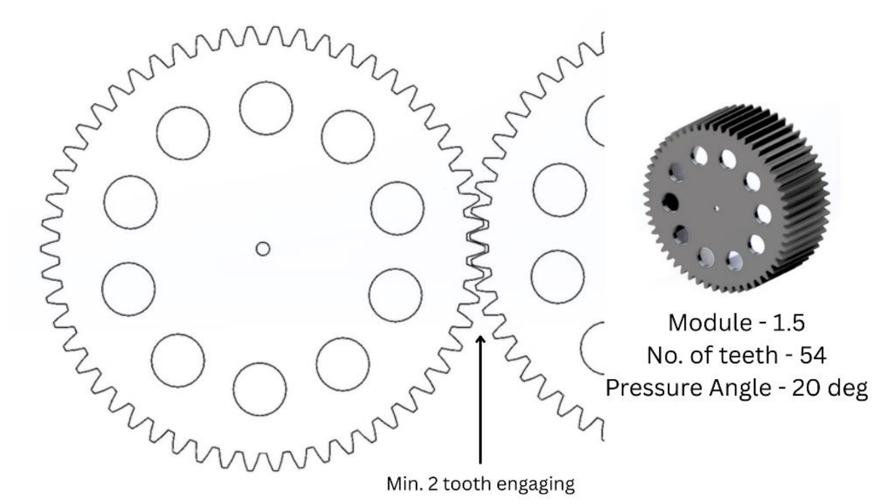


***Figure 32**. Model of Geared Wheels*

3.2.3. Docking System

A key component of the modular robot's self-reconfigurable ability is its magnetic docking system, which allows individual units to physically connect, detach, and align themselves in a structured formation. The system is designed for one-axis reconfiguration, allowing robots to attach either in a linear chain or radial layout on a 2D plane.

3.2.3.1. Magnetic Coupling Design

Each robot module is embedded with a magnetic docking strip placed at the central axis of the cylindrical chassis. This strip contains 13 neodymium N55 magnets, each of size 20 × 11 × 3 mm. These magnets are fixed in a straight configuration to ensure uniform attractive force along the docking axis.

Choice of Magnet:

- Type: N55 Neodymium (one of the strongest commercially available grades)
- Pull Force (approx):
  Each N55 magnet of 20x11x3 mm can provide around 3.5–4 kg pull force when placed on flat steel under ideal conditions. However, due to air gap, PLA cover, and alignment error, effective force is estimated at ~1.5–2 kg per magnet

This is more than sufficient to hold two or three 300g Robots together even during basic motion. No mechanical latching is used; only magnetic attraction is utilized. But geared shaped wheels give advantage to move the bot another without slipping.

Surface Chamfering for Alignment:

To facilitate passive self-alignment during docking, each chassis includes angled chamfers (around 30°–45°) on the front docking face. These chamfered surfaces help to:

- Guide the opposing module during connection
- Prevent bouncing or misalignment
- Ensure that the magnetic surfaces snap together properly even with slight offset

This funnel-like effect ensures that even imperfectly aligned robots can gradually be pulled into precise contact.

One-Axis Reconfiguration

The robot's docking system is intentionally limited to one-axis reconfiguration to simplify:

- Control algorithms
- Mechanical construction
- Energy distribution and communication in future versions

In this version, docking is performed in a linear head-to-tail manner. Lateral or vertical stacking is not implemented, as the current version lacks omnidirectional mobility or mechanical interlocks.

This design was ideal for the scope of this prototype phase and allows demonstrating:

- Modular group formation
- Task allocation among connected units
- Centralized or distributed control via master module

Torque and Motor Speed Calculations (With Magnetic Load)

Weight per robot: 300 g → ~0.3 kg

Assume 2 modules connected → Total: ~0.6 kg

Friction force (rolling, estimated):

- Assume friction coefficient (μ) ≈ 0.3
- Friction force = μ × N = 0.3 × (0.6 × 9.81) ≈ 1.77 N

Torque needed per motor:

- Wheel radius = 4.25 cm = 0.0425 m
- Torque = Force × Radius = 1.77 N × 0.0425 m ≈ 0.075 Nm

  Assume this is evenly split between two motors → each needs at least 0.038 Nm

  Motor Selection Justification:
- N20 6V gear motor (with 100:1 gearbox) can provide 0.1–0.3 Nm torque, depending on model → Sufficient for movement even with magnetic load and added friction

  Speed with load:
- If motor gives 60 RPM:

$$\text{Linear speed} = \frac{2\pi R \times \text{RPM}}{60}$$

$$= \frac{2 \times \pi \times 4.25 \times 60}{60} = \sim 26.7 \text{ cm/s}$$

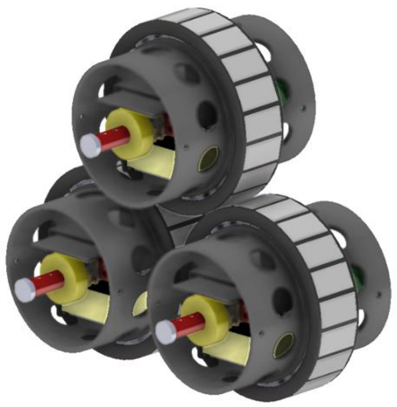


Figure 33. *Magnetic Docking Strip Design (CAD)*

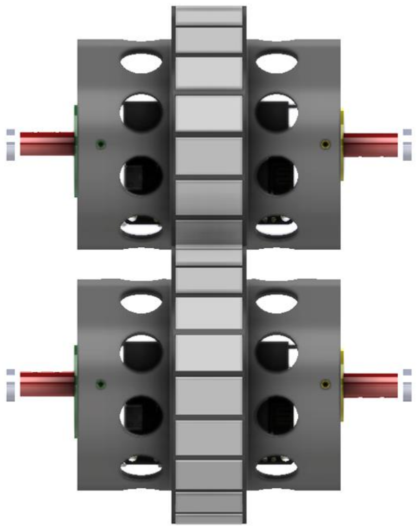


Figure 34. *Magnetic Strip Placement on Chassis*

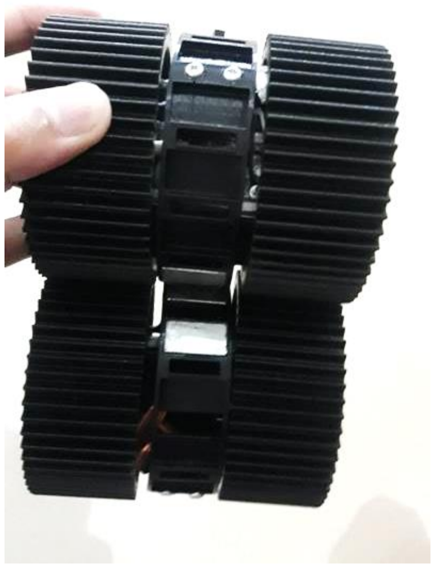


Figure 35*. Two Modules Docked Together*

3.2.4. Extension Arm Mechanism

To enhance the versatility and modular utility of the robotic system, a gear-driven scissor-type extension arm is integrated with the third module. This mechanism is magnetically docked to the module, ensuring both mechanical attachment and electrical connectivity without requiring a dedicated microcontroller, battery, or power supply. The extension arm derives power and control signals directly from the third module, reducing onboard complexity and total system weight.

The structure of the arm is designed as a scissor (zig-zag) linkage, driven by a gear assembly connected to a positional SG90 servo motor capable of 180° rotation. The servo motor drives a primary gear, which is coupled to a secondary gear to actuate the scissor links. This configuration ensures compact folding and linear extension with precise control. The arm is 3D-printed using PLA with 10% infill density, providing sufficient strength while maintaining low mass for stable deployment.

In its retracted state, the extension arm measures approximately 5 cm—half the height of the bot module. When fully extended, it reaches up to 20 cm, which is more than twice the module’s base dimension, significantly increasing the reach of the robot for specific tasks such as manipulation, inspection, or docking.

This detachable extension mechanism demonstrates the adaptability and modular expandability of the system, aligning with the project's goal of reconfigurable robotic functionality.

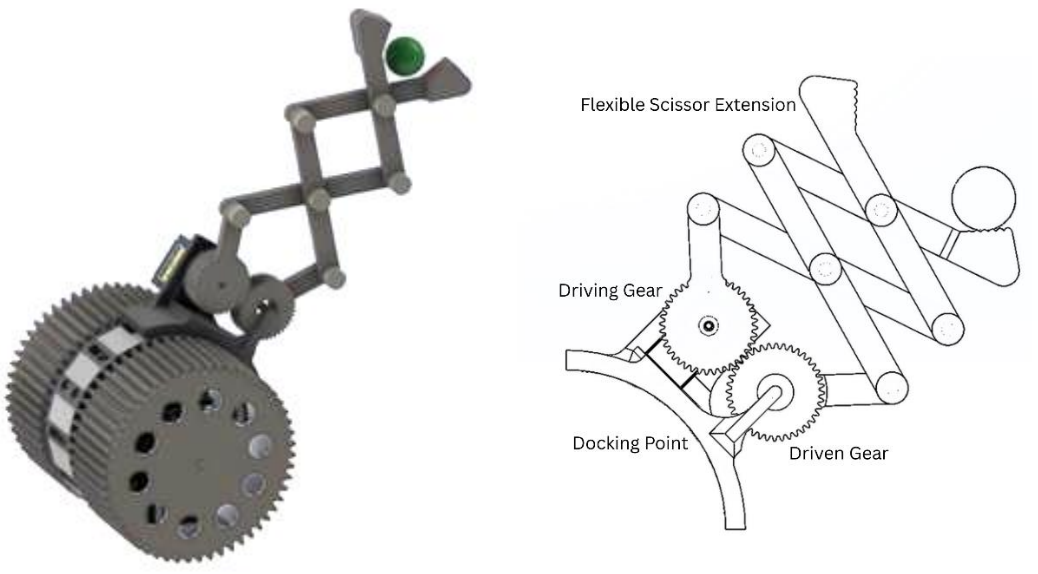


Figure 36. *Gripper design in SolidWorks*

3.3. Communication System Methodology

3.3.1. WIFI Connectivity

In the world of IoT, a stable internet connection is essential for devices to communicate with each other. For this purpose, we used the ESP32 programming model provided by Espressif, as shown below:

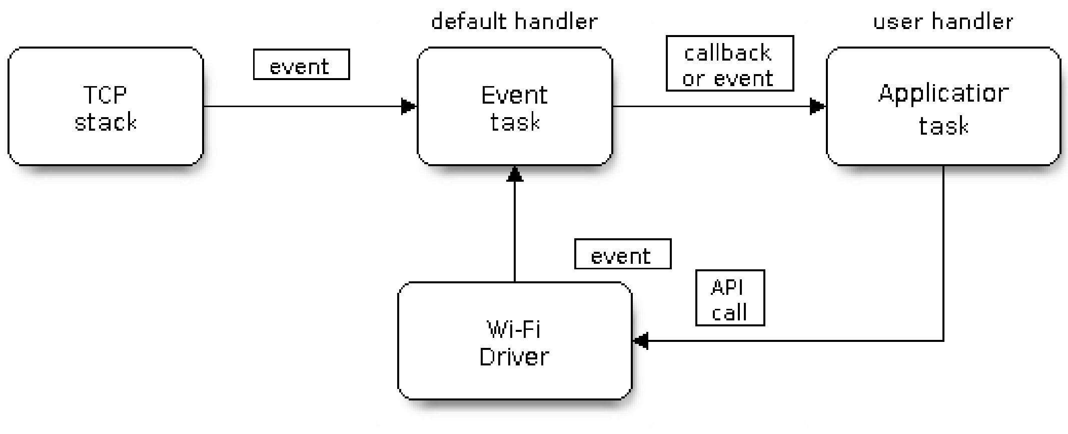
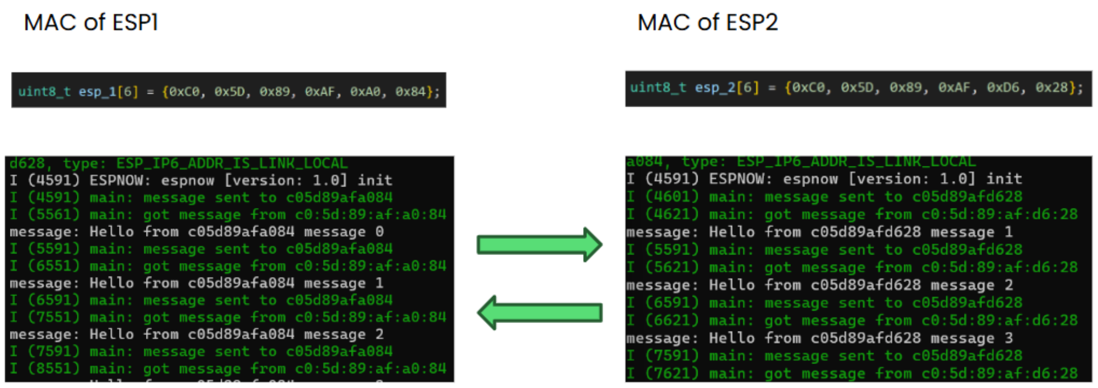


Figure 37. *ESP32 Wi-Fi Programming Model*

3.3.2. Module-to-Module Communication

Each module contains an embedded ESP32, and since every ESP32 has a unique MAC address, we used the ESP-NOW wireless communication protocol for communication between modules. Currently, we have only three units, and ESP-NOW is suitable for this setup because it offers fast response times and low power consumption. One unit can communicate with up to 20 modules within a range of approximately 100 meters. It can transmit payloads of up to 250 bytes without requiring an internet connection.

However, ESP-NOW is not ideal for scalability. As the number of modules increases, managing communication becomes complex. In such cases, a more lightweight and reliable network protocol is needed to ensure efficient wireless communication.



```
uint8_t esp_1[6] = {0xC0, 0x5D, 0x89, 0xAF, 0xA0, 0x84};
```

```
uint8_t esp_2[6] = {0xC0, 0x5D, 0x89, 0xAF, 0xD6, 0x28};
```

```
d628, type: ESP_IP6_ADDR_IS_LINK_LOCAL
I (4591) ESPNOW: espnow [version: 1.0] init
I (4591) main: message sent to c05d89afa084
I (5561) main: got message from c0:5d:89:af:a0:84
message: Hello from c05d89afa084 message 0
I (5591) main: message sent to c05d89afa084
I (6551) main: got message from c0:5d:89:af:a0:84
message: Hello from c05d89afa084 message 1
I (6591) main: message sent to c05d89afa084
I (7551) main: got message from c0:5d:89:af:a0:84
message: Hello from c05d89afa084 message 2
I (7591) main: message sent to c05d89afa084
I (8551) main: got message from c0:5d:89:af:a0:84
```

```
a084, type: ESP_IP6_ADDR_IS_LINK_LOCAL
I (4591) ESPNOW: espnow [version: 1.0] init
I (4601) main: message sent to c05d89afd628
I (4621) main: got message from c0:5d:89:af:d6:28
message: Hello from c05d89afd628 message 1
I (5591) main: message sent to c05d89afd628
I (5621) main: got message from c0:5d:89:af:d6:28
message: Hello from c05d89afd628 message 2
I (6591) main: message sent to c05d89afd628
I (6621) main: got message from c0:5d:89:af:d6:28
message: Hello from c05d89afd628 message 3
I (7591) main: message sent to c05d89afd628
I (7621) main: got message from c0:5d:89:af:d6:28
```

***Figure 38**. Peer to Peer handshake*

3.3.3. Module-to-Cloud Communication

To communicate modules to cloud we used MQTT protocol. This is efficient, light weight and easily scalable protocol. It can support millions of devices in IoT ecosystems.

Key components:

1. Publisher (ESP-32)
2. MQTT Broker (Cloud)
3. Subscriber (Web-Page)

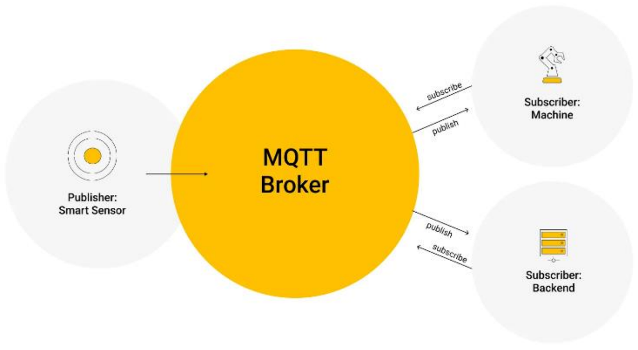


***Figure 39**. MQTT Pub-Sub Model*

Configuration and Initialization of MQTT client.

```
const esp_mqtt_client_config_t mqtt_cfg = {
.broker.address.uri = "mqtt://mqtt.eclipseprojects.io:1883"};
client = esp_mqtt_client_init(&mqtt_cfg);
```

Function to publish the payloads.

```
int mqtt_send(const char *topic, const char *payload)
{
return esp_mqtt_client_publish(client, topic, payload, strlen(payload),1, 0);
}
```

Set the QoS 1, In QoS 1 of MQTT, the focus is on ensuring message delivery at least once to the receiver. When a message is published with QoS 1, the sender keeps a copy of the message until it receives a packet from the receiver, confirming the successful receipt. If the sender doesn't receive the packet within a reasonable time frame, it re-transmits the message to ensure its delivery.

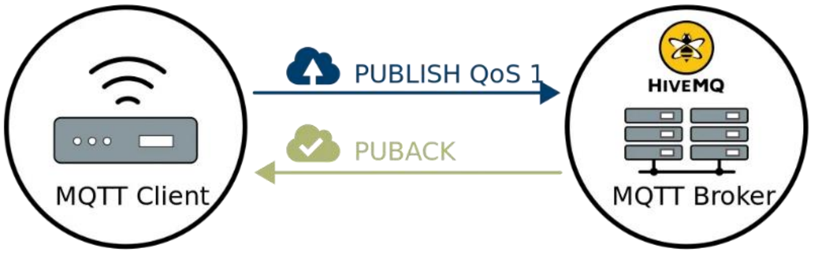


Figure 40. *Quality of Service level 1*

3.4. Electronics Hardware

1. ESP32 (Central Controller):
   - Acts as the brain of the system
   - Interfaces with sensors, motors, motor drivers, and other peripherals

2. BU04 Ai-Thinker UWB Module:
   - Used for distance measurement and communication
   - Connected to ESP32 via SPI pins
   - Powered by 3.3V from ESP32 or battery circuit

3. MPU6050 (Gyroscope + Accelerometer):
   - Provides orientation and motion sensing
   - Communicated with ESP32 through I2C protocol (SCL and SDA lines)

4. N20 Encoder Motors:
   - Two DC gear motors with built-in quadrature encoders
   - Controlled using the L9110S motor driver

5. L9110S Motor Driver:
   - Drives the motors by receiving PWM (speed) and direction signals from ESP32
   - Outputs are connected to the motor wires

6. Mini Buck Converter:
   - Converts 7.5V from the battery to 5V to safely power ESP32, motor driver, and sensors
   - Ensures stable voltage supply to components that require 5V

7. 2S BMS (Battery Management System):

- Manages charging and discharging of the 2-cell LiPo battery
- Provides overcharge, over-discharge, and short-circuit protection
- Connected directly to the battery and outputs a stable 7.5V

8. Lippo Battery:

- Powers the entire system
- Connected to BMS for safe charging and output regulation

9. ON/OFF Switch and Charging Port:

- ON/OFF switch is used to control the power supply to the whole system
- Charging port allows safe recharging of the LiPo battery via the BMS

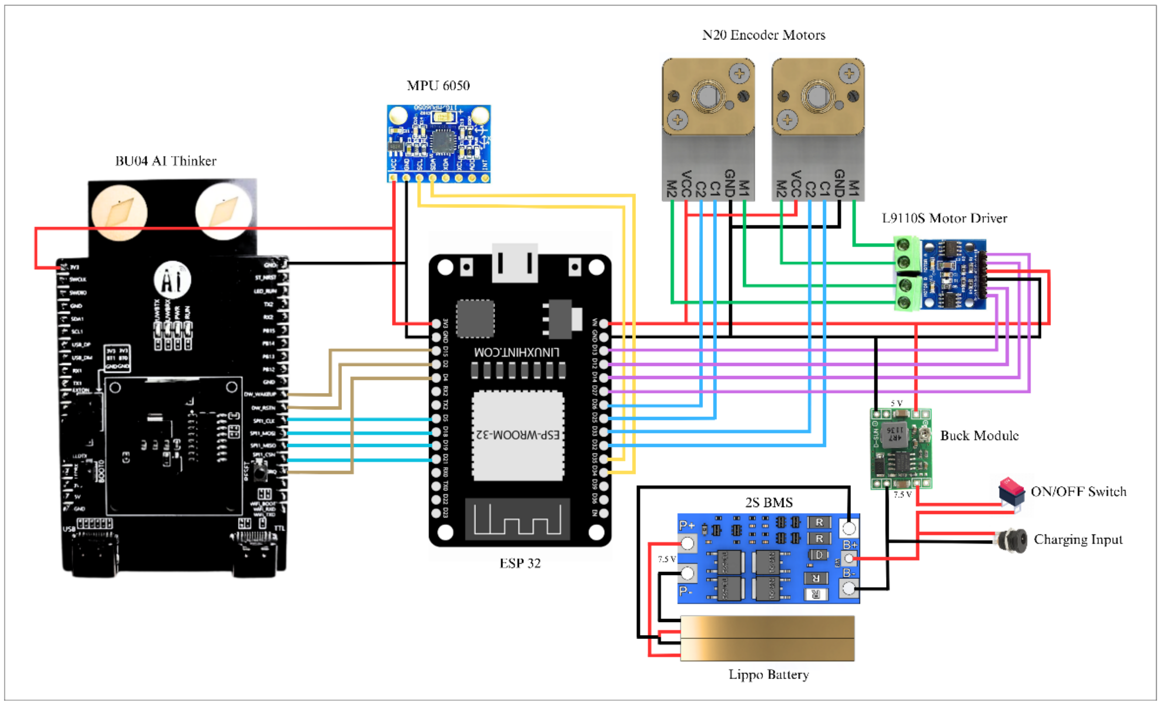
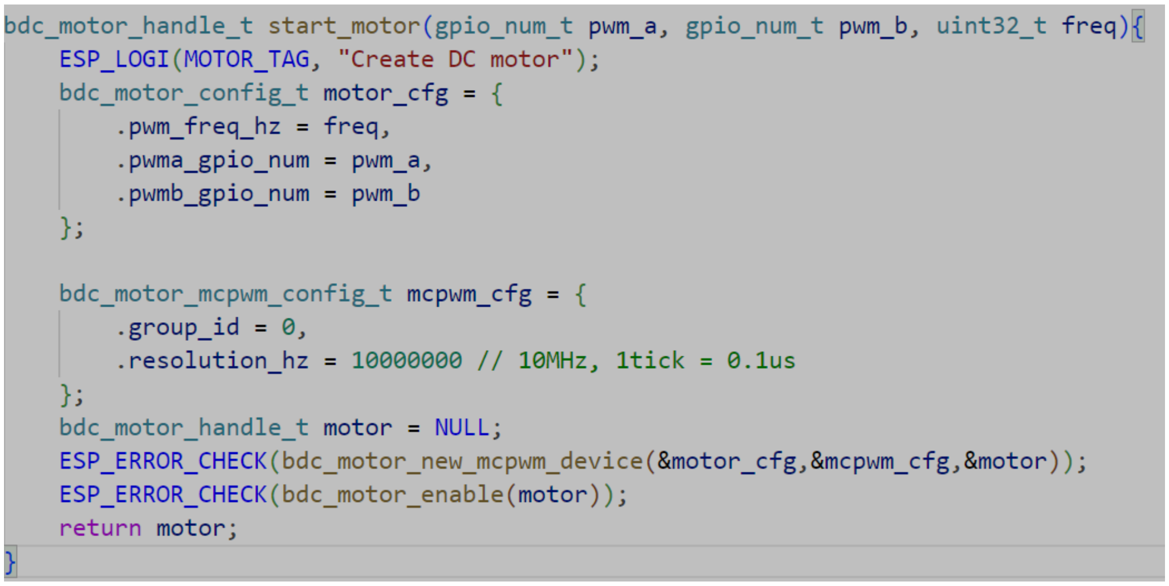


***Figure 41**. Wiring diagram*

3.5. Control Algorithm and Software Design

3.5.1. Differential Drive

We used espressif__bdc__motor esp-idf component written in C to control the dc motor. First we configured the **bdc_motor_config_t** & **bdc_motor_mcpwm_config_t** structure for motor and pwm control.



```
bdc_motor_handle_t start_motor(gpio_num_t pwm_a, gpio_num_t pwm_b, uint32_t freq){
    ESP_LOGI(MOTOR_TAG, "Create DC motor");
    bdc_motor_config_t motor_cfg = {
        .pwm_freq_hz = freq,
        .pwma_gpio_num = pwm_a,
        .pwmb_gpio_num = pwm_b
    };

    bdc_motor_mcpwm_config_t mcpwm_cfg = {
        .group_id = 0,
        .resolution_hz = 10000000 // 10MHz, 1tick = 0.1us
    };
    bdc_motor_handle_t motor = NULL;
    ESP_ERROR_CHECK(bdc_motor_new_mcpwm_device(&motor_cfg,&mcpwm_cfg,&motor));
    ESP_ERROR_CHECK(bdc_motor_enable(motor));
    return motor;
}
```

Figure 42. *Motor and PWM configuration*

To control the motor, use the following custom API:
void set_speed_direction(bdc_motor_handle_t motor, uint32_t speed, bool forward);

- motor: Pass the bdc_motor_handle_t object representing the motor instance.
- speed: Desired speed value.
- forward: Set to true for forward direction, false for reverse

Valid speed range < resolution_hz / pwm_freq

3.5.2. Odometry

Odometry is a specific form of dead reckoning that estimates a robot’s movement using data from wheel encoders. Encoders are interoceptive sensors, meaning they measure internal states of the robot—such as wheel rotations or joint angles—rather than external environmental cues. By counting the pulses generated as the wheels turn, encoders provide precise information about the robot’s displacement and velocity over time. Although they do not offer absolute position in global coordinates, encoders are fundamental to dead reckoning, enabling relative position tracking from a known starting point. This makes them particularly useful for short-term localization, and their effectiveness can be significantly enhanced when integrated with additional sensors like IMUs or UWB systems to improve long-term accuracy and compensate for drift.

$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \qquad p' = p + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$$

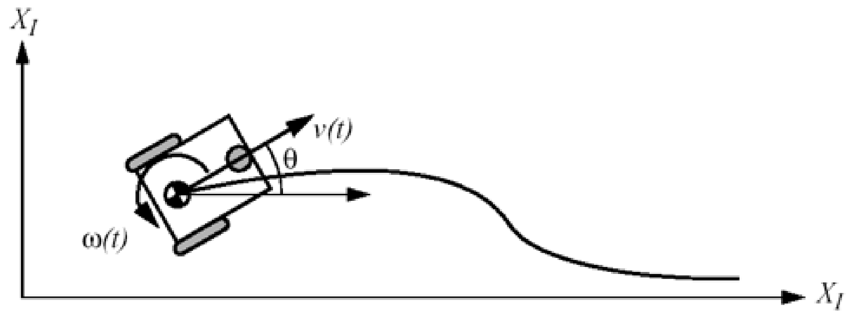


Figure 43. *Differential drive movement*

The robot's movement in a small time step is determined by:

- $\Delta s_r \rightarrow$ Distance traveled by the **right** wheel.
- $\Delta s_l \rightarrow$ Distance traveled by the **left** wheel.

Change in orientation (angle change) $\Delta\theta$:

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b}$$

$b \rightarrow$ Distance between the two wheels (wheelbase).

- If $\Delta s_r = \Delta s_l \rightarrow$ Robot moves straight.
- If $\Delta s_r > \Delta s_l \rightarrow$ Robot turns **left**.
- If $\Delta s_r < \Delta s_l \rightarrow$ Robot turns **right**.

Change in arc length traveled Δs (midpoint displacement):

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2}$$

- This is the **average** distance traveled by both wheels.

The new position is calculated using:

1. **Change in x-coordinate:**

$$\Delta x = \Delta s \cos(\theta + \frac{\Delta\theta}{2})$$

- The robot moves forward along its current direction θ.

2. **Change in y-coordinate:**

$$\Delta y = \Delta s \sin(\theta + \frac{\Delta\theta}{2})$$

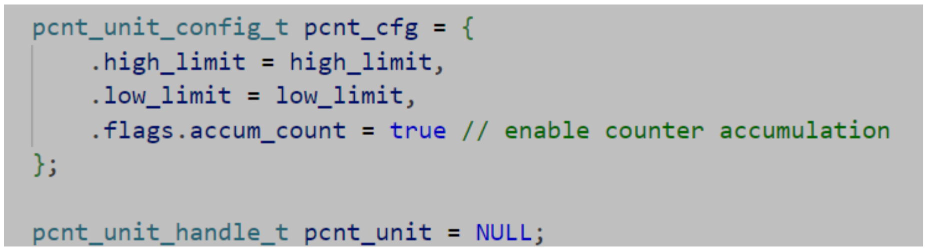
- The y-coordinate is updated similarly.

3. **New Position Vector:**

$$p' = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = p + \begin{bmatrix} \Delta s \cos(\theta + \Delta\theta/2) \\ \Delta s \sin(\theta + \Delta\theta/2) \\ \Delta\theta \end{bmatrix}$$

Encoder custom library is dependent on pulse counter component that comes up with ESP-IDF framework. In this we made setup_pcnt_encoder() API.

1. Initialization and configuration of PCNT Unit



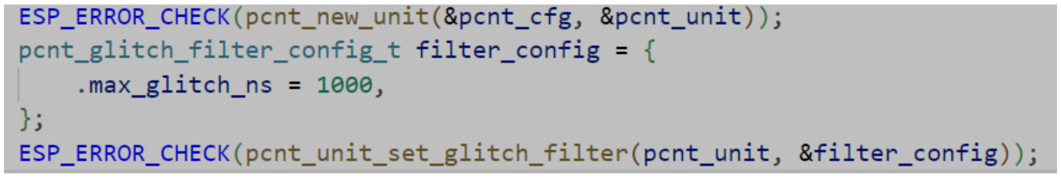
```
pcnt_unit_config_t pcnt_cfg = {
    .high_limit = high_limit,
    .low_limit = low_limit,
    .flags.accum_count = true // enable counter accumulation
};

pcnt_unit_handle_t pcnt_unit = NULL;
```

Figure 44. *Initialization of pulse counter*

2. Glitch Filter:

A glitch filter is applied to the pulse signals. It eliminates spurious noise or short pulses (under 1000 ns), ensuring that only valid encoder transitions are counted—important for signal integrity, especially in noisy environments.



```
ESP_ERROR_CHECK(pcnt_new_unit(&pcnt_cfg, &pcnt_unit));
pcnt_glitch_filter_config_t filter_config = {
    .max_glitch_ns = 1000,
};
ESP_ERROR_CHECK(pcnt_unit_set_glitch_filter(pcnt_unit, &filter_config));
```

Figure 45. *Code block of glitch filter*

3. Channel A & B Configuration:

This section configures Channel A of the pulse counter, linking one GPIO as the edge detector and another as the level reference. It sets how the counter should react to rising/falling edges and the level of the signal, enabling accurate direction and movement detection from the encoder.

Channel B is configured similarly but with inverted roles for the GPIO pins. This complementary setup allows for full quadrature decoding, which improves resolution and enables direction sensing by observing phase shifts between signals.

4. Activation and start of PCNT Unit

Finally, the configured PCNT unit is enabled, the counter is cleared to start from zero, and counting is initiated. This block ensures that the encoder is ready to begin operation and record pulse movements in real-time.

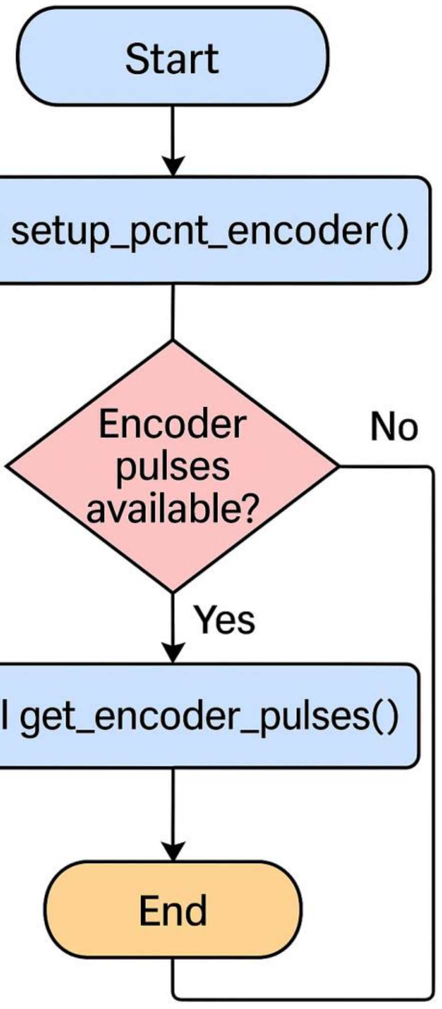
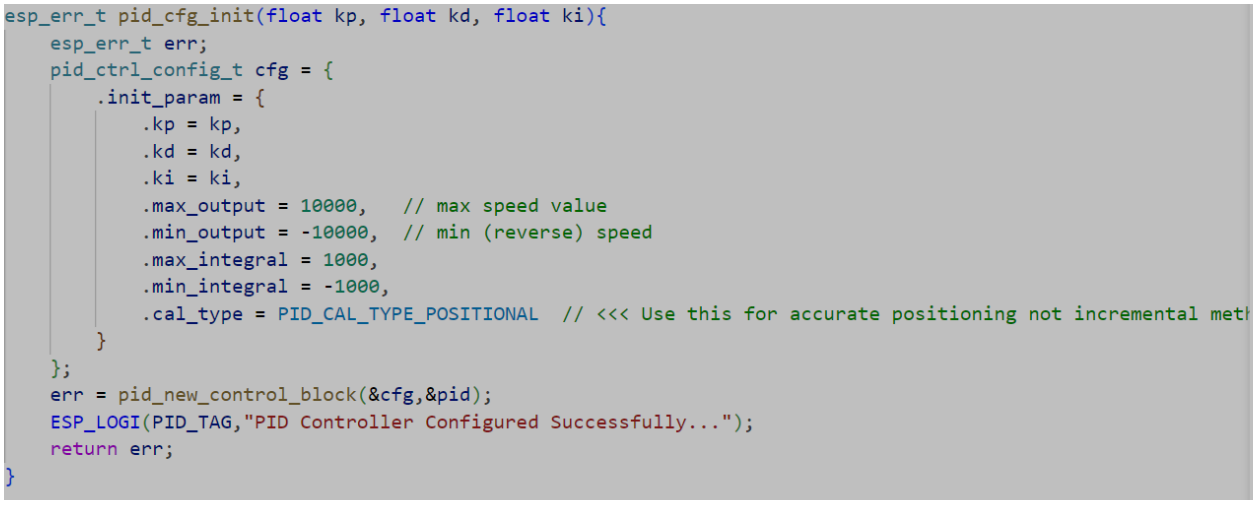

Start
Call setup_pcnt_encoder()
Encoder pulses available?
No
Yes
Call get_encoder_pulses()
End

Figure 46. *Flow chart of pulse counter*

3.5.3. PID Control

1. PID configuration & Initialization

   Set Kp, Ki and Kd by tuning.



```
esp_err_t pid_cfg_init(float kp, float kd, float ki){
    esp_err_t err;
    pid_ctrl_config_t cfg = {
        .init_param = {
            .kp = kp,
            .kd = kd,
            .ki = ki,
            .max_output = 10000,    // max speed value
            .min_output = -10000,   // min (reverse) speed
            .max_integral = 1000,
            .min_integral = -1000,
            .cal_type = PID_CAL_TYPE_POSITIONAL   // <<< Use this for accurate positioning not incremental meth
        }
    };
    err = pid_new_control_block(&cfg,&pid);
    ESP_LOGI(PID_TAG,"PID Controller Configured Successfully...");
    return err;
}
```

Figure 47. *Initialization of PID control block*

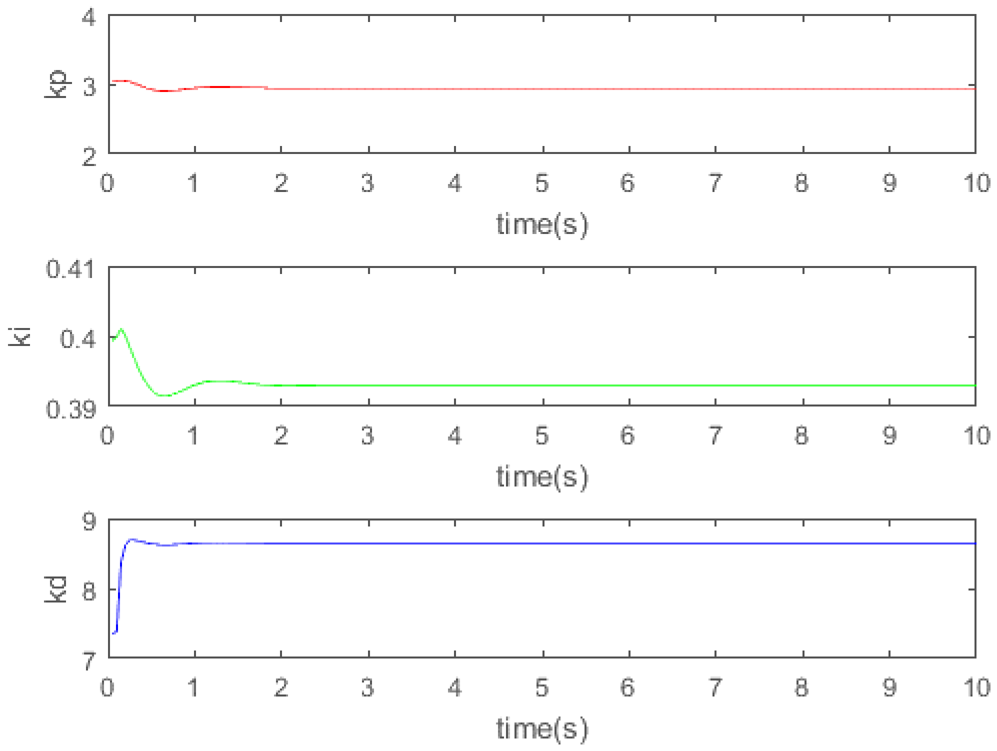


Figure 48. *Responses of kp, ki, and kd*

Here is the breakdown of PID control loop task:

1. Setup and Initialization:

   The task starts by defining the desired target position (from the macro TARGET_POSITION) and initializing a variable for control output. These will be used in the PID feedback loop to control the motor's position based on encoder feedback.

2. Reading Encoder and Error Calculation:

Inside the infinite loop, the current position of the motor is read using the encoder. The error is calculated as the difference between the target and current positions. This error is what the PID controller will try to minimize.

3. Deadband Zone:

A deadband is applied to ignore tiny errors (within ±2 pulses) to prevent the motor from continuously adjusting back and forth. If the error is within this threshold, the motor is stopped to avoid jittering, which is common when close to the target.

4. PID Computation:

If the error is outside the deadband, the PID controller computes a new control output based on the position error. This output determines the motor's speed and direction, considering proportional, integral, and derivative effects.

5. Motor Speed and Direction Control:

The sign of the PID output determines the direction (forward or reverse), and the magnitude sets the speed. The speed is clamped to stay within a safe operating range (between 100 and 400), avoiding too low or too high speeds that might be inefficient or unsafe.

6. Loop Timing:

The loop runs with a fixed interval (PID_INTERVAL_MS = 10ms), ensuring a stable control rate. It uses vTaskDelay to pause between iterations, which is important in real-time systems to give other tasks time to run.

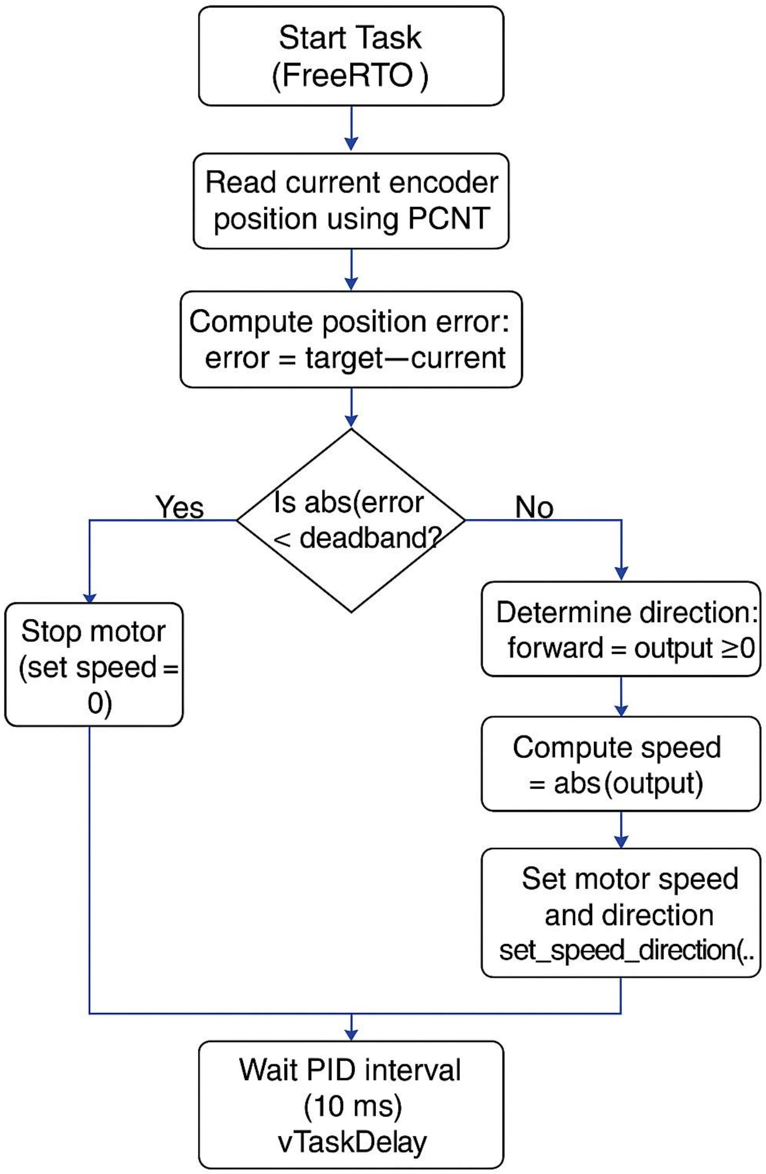

Start Task (FreeRTO)
Read current encoder position using PCNT
Compute position error: error = target—current
Is abs(error < deadband?
Yes
No
Stop motor (set speed = 0)
Determine direction: forward = output ≥0
Compute speed = abs(output)
Set motor speed and direction set_speed_direction(..
Wait PID interval (10 ms) vTaskDelay

***Figure 49**. Flow of PID control task*

3.6. Positioning System

For a robotic system to operate autonomously, it must have an accurate understanding of its position and orientation in 3D space. This requires knowledge of its x, y, z coordinates along with θ (theta), which represents its orientation. To achieve this, the system integrates data from two key components: the MPU6050 inertial sensor for orientation and a UWB (Ultra-Wideband)-based localization system for precise positioning. UWB technology provides high-accuracy distance measurements—typically within 5 cm—making it highly suitable for indoor environments. Each mobile module estimates its position by measuring its distance from multiple fixed UWB anchors and applying the principle of trilateration, enabling real-time and reliable localization.

3.6.1. Distance Measurement

Ultra-Wideband (UWB) technology is increasingly utilized for localization and tracking applications, such as asset monitoring and indoor positioning. It operates using short-range radio signals to measure the Time of Flight (ToF)—the time it takes for a signal to travel between two devices—allowing precise distance estimation using the formula:

Distance = ToF × Speed of Light.

ToF can be measured using methods like Single-Sided Two-Way Ranging (SS-TWR), Double-Sided Two-Way Ranging (DS-TWR), or Time Difference of Arrival (TDoA). In our system, we employed SS-TWR due to its simplicity and its high accuracy of approximately ±2 cm.

$$ToF\ (SS - TWR) = \frac{t,\ round1 - t, reply1}{2}$$

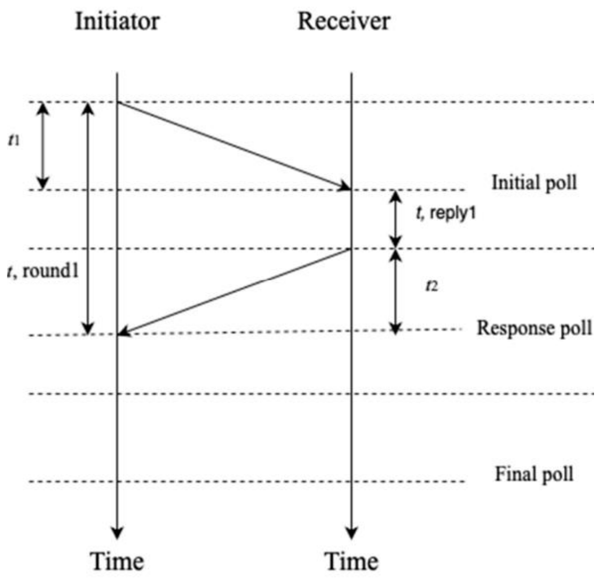


Figure 50. *Single-Sided Two-way Ranging*

3.6.2. Indoor Positioning

Trilateration is the method of determining the position of a point (called the tag) based on its measured distances to known fixed points (called anchors).

We usually start with 2D trilateration using 3 anchors, then extend it to 3D with 4 anchors.
In Figure below:

Let's assume:

- You have **three anchors** at known coordinates:
  - Anchor A1 at (x_1, y_1)
  - Anchor A2 at (x_2, y_2)
  - Anchor A3 at (x_3, y_3)
- The distances from the tag to these anchors are:
  - d_1, d_2, d_3

You want to find the tag's position: (x, y)

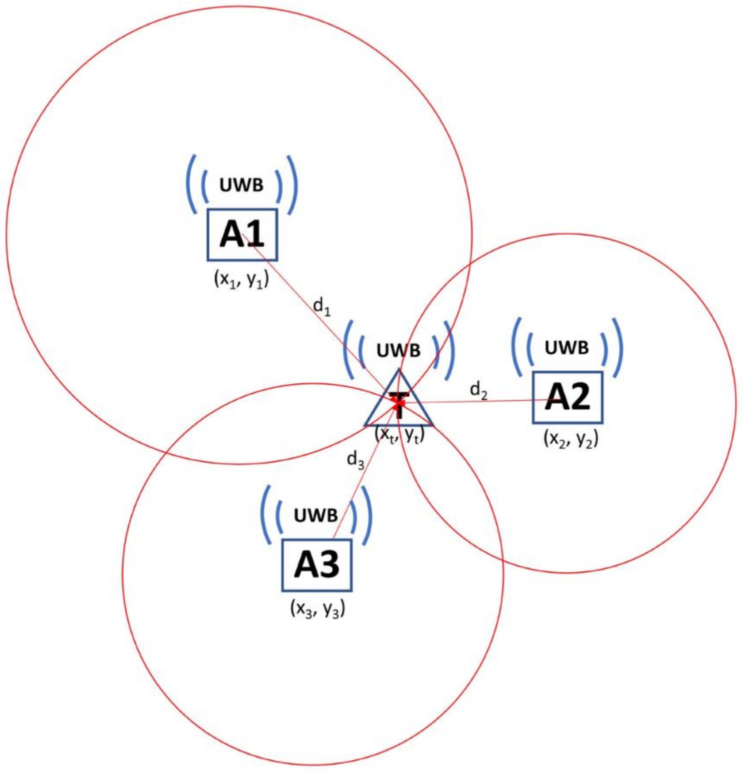


Figure 51. *TWR based Trilateration Positioning*

Using Euclidean distance formula:

$$(x - x_1)^2 + (y - y_1)^2 = d_1^2 \quad \text{(Eq 1)}$$

$$(x - x_2)^2 + (y - y_2)^2 = d_2^2 \quad \text{(Eq 2)}$$

$$(x - x_3)^2 + (y - y_3)^2 = d_3^2 \quad \text{(Eq 3)}$$

Subtract Eq2 – Eq1

This eliminates the quadratic terms:

$$(x - x_2)^2 - (x - x_1)^2 + (y - y_2)^2 - (y - y_1)^2 = d_2^2 - d_1^2$$

Simplify:

$$2(x_1 - x_2)x + 2(y_1 - y_2)y = d_1^2 - d_2^2 + x_2^2 - x_1^2 + y_2^2 - y_1^2$$

Do the same for Eq3 - Eq1:

$$2(x_1 - x_3)x + 2(y_1 - y_3)y = d_1^2 - d_3^2 + x_3^2 - x_1^2 + y_3^2 - y_1^2$$

Represent as Linear System

You now have two linear equations in two unknowns (x, y):

$$\begin{gathered} Ax + By = C \\ Dx + Ey = F \end{gathered}$$

In 3D, you use 4 anchors at (x_i, y_i, z_i) with distances d_i. You get:

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = d_i^2$$

You subtract the equations again to remove quadratic terms and end up with a **linear system of 3 equations in 3 variables (x, y, z)**, which you solve using matrix methods.

3.7. Orientation

We used the MPU6050 sensor to estimate orientation. In the mpu6050_rpy_task() function, the following operations were performed:

3.7.1. Sensor Fusion (Accelerometer + Gyroscope)

Sensor fusion combines data from multiple sensors to produce more accurate and reliable information than any single sensor alone. In this case:

- Accelerometers measure linear acceleration and can indirectly estimate orientation using gravity.
- Gyroscopes measure angular velocity (how fast the device is rotating).

By fusing them, we can get the short-term responsiveness of gyros and the long-term stability of accelerometers.

3.7.2. Complementary Filter

The complementary filter is a simple and computationally efficient sensor fusion algorithm. It works by combining the low-frequency information from the accelerometer and the high-frequency information from the gyroscope.

```
roll  = α * (roll + gyro_roll_rate * dt) + (1 - α) * accel_roll;
```

pitch = α * (pitch + gyro_pitch_rate * dt) + (1 - α) * accel_pitch;

- α is a tuning constant (0 < α < 1), e.g., 0.98.
- gyro_roll_rate * dt is the change in angle from gyroscope integration.
- accel_roll is the absolute angle estimate from the accelerometer.
- roll + gyro*dt is the predicted roll using gyroscope.

3.7.3. Mathematical Formulation:

1. Accelerometer-Based Orientation:
Derived from gravity vector projections:

- **Roll (rotation about X-axis):**

$$\text{roll}_{accel} = \arctan\left(\frac{a_y}{a_z}\right)$$

- **Pitch (rotation about Y-axis):**

$$\text{pitch}_{accel} = \arctan\left(\frac{-a_x}{\sqrt{a_y^2 + a_z^2}}\right)$$

This assumes that the only acceleration is gravity (i.e., the device is stationary or moving slowly).

2. Gyroscope Integration:
Orientation change from angular velocity:

$$\theta(t) = \theta(t - dt) + \omega \cdot dt$$

Where:

- `θ(t)` is the angle at time `t`
- `ω` is the angular velocity from the gyroscope

Gyroscopes are integrated over time to estimate angle changes.

3.8. Software Architecture

FreeRTOS on ESP32 manages resources through a combination of task scheduling, inter-task communication, and memory management.

3.8.1. Task Management

Tasks are implemented as C functions. The only thing special about them is their prototype, which must return void and take a void pointer parameter.

```
void ATaskFunction( void *pvParameters );
```

Each task is a small program in its own right. It has an entry point, will normally run forever within an infinite loop, and will not exit. Tasks are created using the FreeRTOS **xTaskCreate()** and **xTaskCreatePinnedToCore()** API function. This is probably the most complex of all the API functions, so it is unfortunate that it is the first encountered, but tasks must be mastered first as they are the most fundamental component of a multitasking system.

```
// All tasks
xTaskCreatePinnedToCore(UWB_task, "UWB Task", 1024 * 2, NULL, 5, NULL, 1);
xTaskCreatePinnedToCore(mpu6050_rpy_task, "MPU6050 Task", 1024*4, NULL, 4, NULL, 1);
xTaskCreatePinnedToCore(servo_motor_task, "Servo Task", 1024 * 2, NULL, 5, NULL, 0);
xTaskCreatePinnedToCore(test_send_messages, "MQTT Task", 1024 * 4, NULL, 3, NULL, 0);
```

The actual pattern of running two tasks on a single core

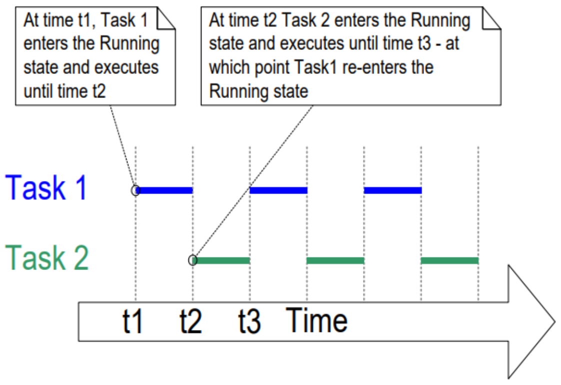
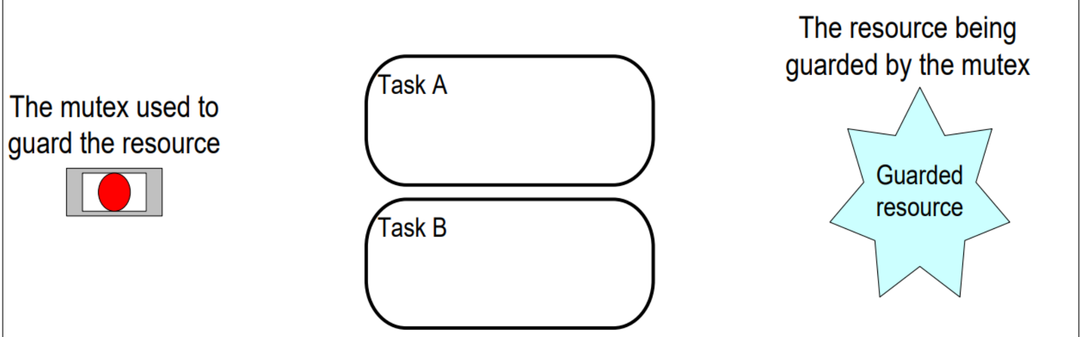

At time t1, Task 1 enters the Running state and executes until time t2
At time t2 Task 2 enters the Running state and executes until time t3 - at which point Task1 re-enters the Running state
Task 1
Task 2
t1
t2
t3
Time

Figure 52. *Two Tasks running on single core*

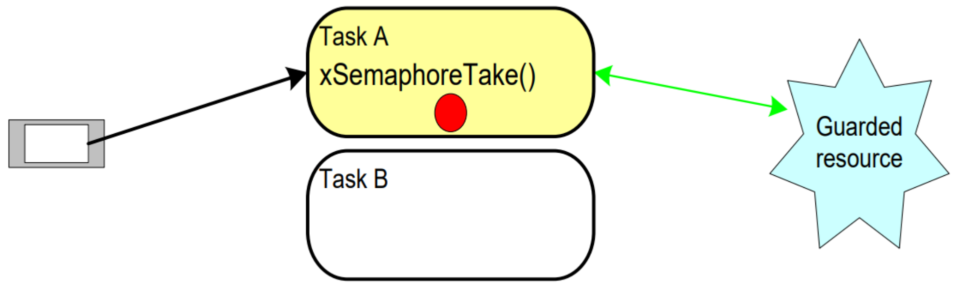
3.8.2. Mutual Exclusion

A Mutex is a special type of binary semaphore that is used to control access to a resource that is shared between two or more tasks.

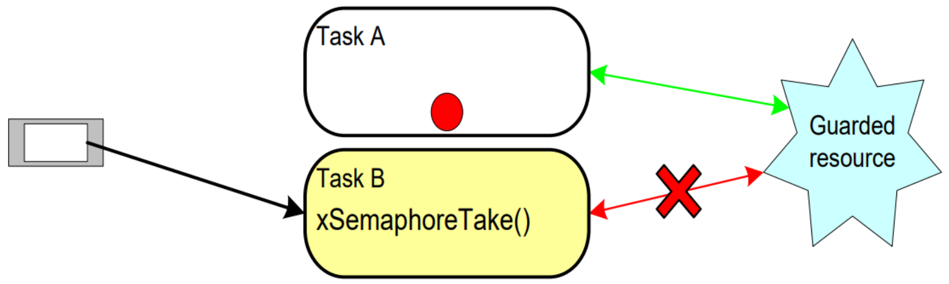
When used in a mutual exclusion scenario, the mutex can be thought of as a token that is associated with the resource being shared. For a task to access the resource legitimately, it must first successfully ‘take’ the token (be the token holder). When the token holder has finished with the resource, it must ‘give’ the token back. Only when the token has been returned can another task successfully take the token, and then safely access the same shared resource. A task is not permitted to access the shared resource unless it holds the token. This mechanism is shown in below diagrams


The resource being guarded by the mutex
The mutex used to guard the resource
Task A
Task B
Guarded resource

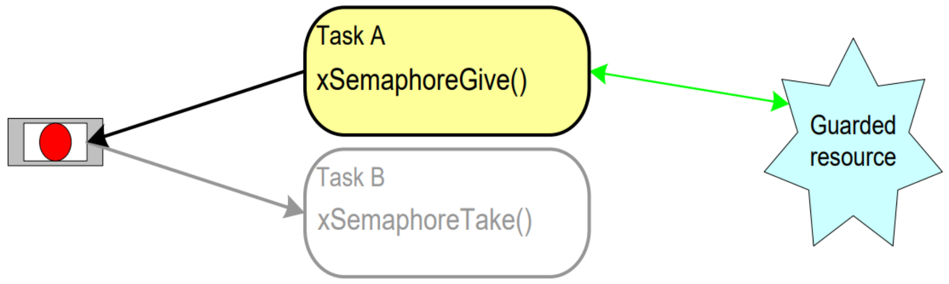
Two tasks each want to access the resource, but a task is not permitted to access the resource unless it is the mutex (token) holder.



Task A attempts to take the mutex. Because the mutex is available Task A successfully becomes the mutex holder so is permitted to access the resource.



Task B executes and attempts to take the same mutex. Task A still has the mutex so the attempt fails and Task B is not permitted to access the guarded resource.



Task B opts to enter the Blocked state to wait for the mutex - allowing Task A to run again. Task A finishes with the resource so 'gives' the mutex back.

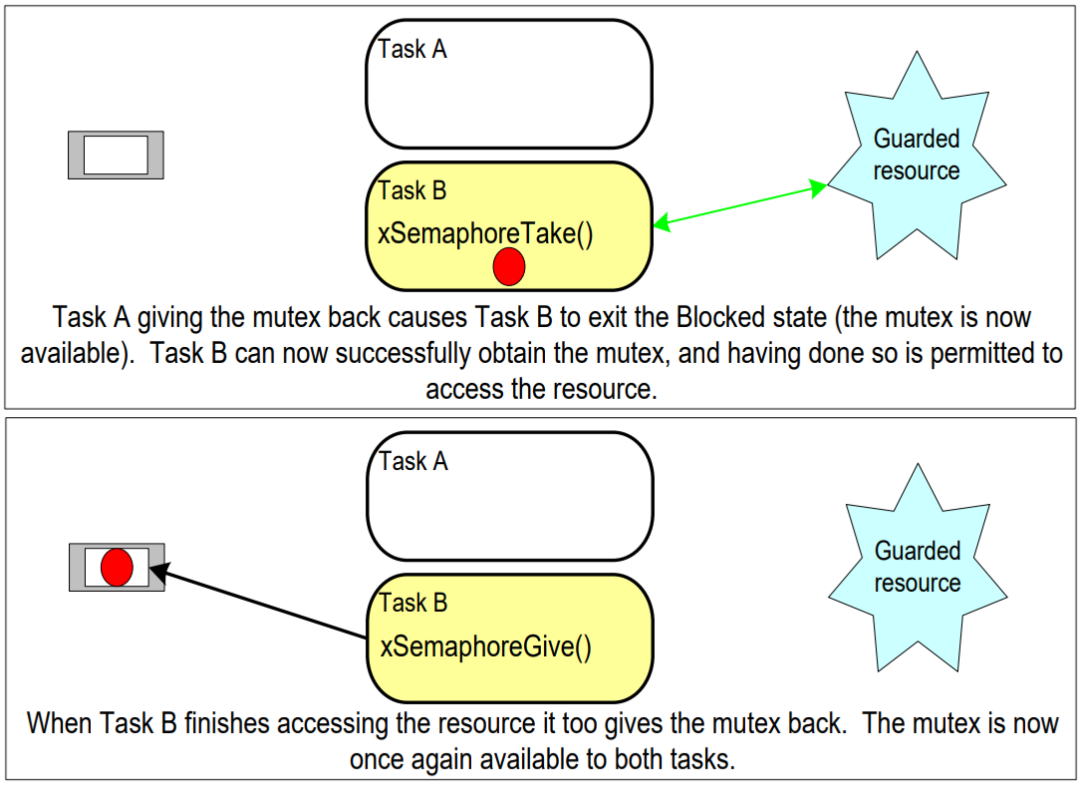

Task A
Task B
xSemaphoreTake()
Guarded resource
Task A giving the mutex back causes Task B to exit the Blocked state (the mutex is now available). Task B can now successfully obtain the mutex, and having done so is permitted to access the resource.
Task A
Task B
xSemaphoreGive()
Guarded resource
When Task B finishes accessing the resource it too gives the mutex back. The mutex is now once again available to both tasks.

Figure 53. *Mutual Exclusion implemented using mutex*

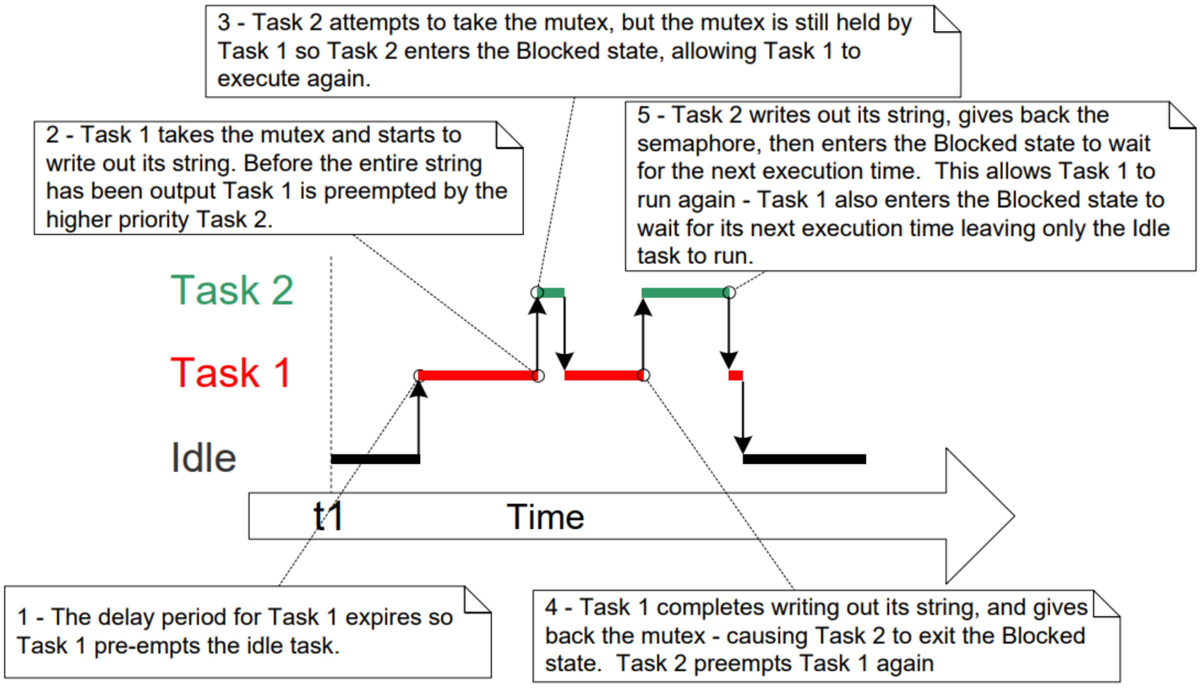
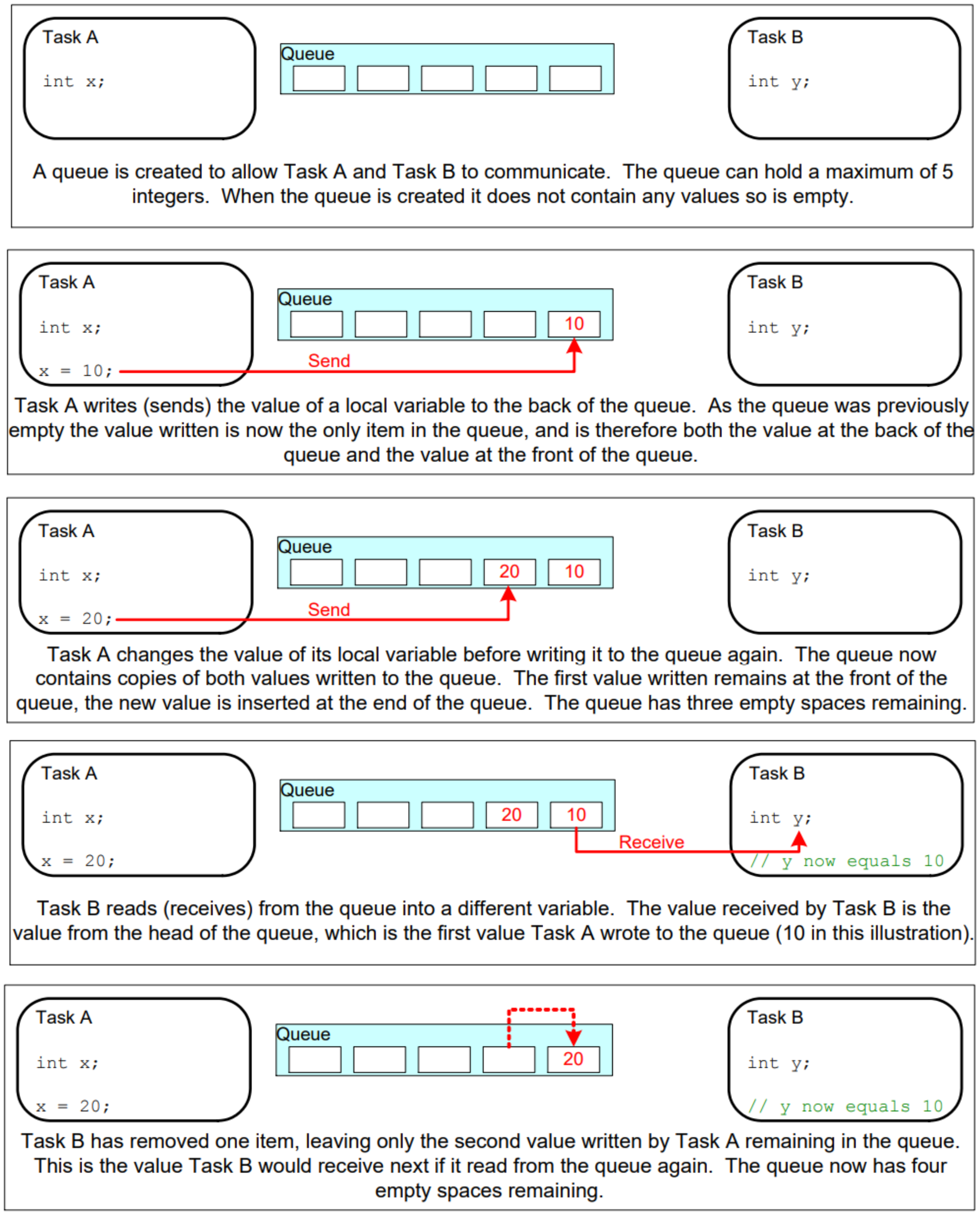

3 - Task 2 attempts to take the mutex, but the mutex is still held by Task 1 so Task 2 enters the Blocked state, allowing Task 1 to execute again.
2 - Task 1 takes the mutex and starts to write out its string. Before the entire string has been output Task 1 is preempted by the higher priority Task 2.
5 - Task 2 writes out its string, gives back the semaphore, then enters the Blocked state to wait for the next execution time. This allows Task 1 to run again - Task 1 also enters the Blocked state to wait for its next execution time leaving only the Idle task to run.
Task 2
Task 1
Idle
t1
Time
1 - The delay period for Task 1 expires so Task 1 pre-empts the idle task.
4 - Task 1 completes writing out its string, and gives back the mutex - causing Task 2 to exit the Blocked state. Task 2 preempts Task 1 again

Figure 54. *Possible sequence in which tasks execute*

3.8.3. Queue Management

'Queues' provide a task-to-task, task-to-interrupt, and interrupt-to-task communication mechanism.

A queue can hold a finite number of fixed size data items. The maximum number of items a queue can hold is called its ‘length’. Both the length and the size of each data item are set when the queue is created.



A queue is created to allow Task A and Task B to communicate. The queue can hold a maximum of 5 integers. When the queue is created it does not contain any values so is empty.

Task A writes (sends) the value of a local variable to the back of the queue. As the queue was previously empty the value written is now the only item in the queue, and is therefore both the value at the back of the queue and the value at the front of the queue.

Task A changes the value of its local variable before writing it to the queue again. The queue now contains copies of both values written to the queue. The first value written remains at the front of the queue, the new value is inserted at the end of the queue. The queue has three empty spaces remaining.

Task B reads (receives) from the queue into a different variable. The value received by Task B is the value from the head of the queue, which is the first value Task A wrote to the queue (10 in this illustration).

Task B has removed one item, leaving only the second value written by Task A remaining in the queue. This is the value Task B would receive next if it read from the queue again. The queue now has four empty spaces remaining.

***Figure 55**. Sequence of writes to, and reads from a queue*

3.9. GUI Design

A Web based GUI was designed for simultaneously sending commands and receiving information from and to the robotic system. Multiple requirements were kept in mind while creating the GUI. Various tools were utilized for the user interface's front-end and back-end.

3.9.1. GUI Requirements

To make the system user friendly, GUI needs to present detailed information regarding the extensions, the system's modules and its environment. Therefore, the following requirements were kept in mind:

- Real-time visualization of modules' position by showing their location in a 3D space along with its coordinates. The module's shape must be depicted as accurately as possible.
- The User must have the ability to create modular configurations in the 3D space. The interface must not allow invalid configurations which cannot be physically achieved by the modular system.
- There must be 3D Visualization of the extension if any in the system. The user should be able to easily control the extension.

While keeping in mind all these requirements, it is imperative that the 3D space must not violate basic rules regarding the modular system. User must only be able to command those configurations which the module's shape and mechanism allows it to achieve. The number of achievable configurations also depends on the number of modules present in the system.

3.9.2. Platform & Tools

The GUI is divided into multiple sections, each fulfilling a certain requirement. A menu has been created from where the user can select which section to visualize and interact with. For the GUI's layout, HTML and CSS were utilized. Three.js, a JavaScript library, was used to incorporate the 3D environment into the web application. The library helps the creation and visualization of modules, extensions and the environment.

Real time live position and orientation of all active modules are shown in one of the GUI's sections. Another section is dedicated for the user to assemble configuration in the 3D space and command the system to achieve the configuration. Both sections require consistent sharing of data between the GUI and the modular system. WebSocket/http is used for sharing data ensuring data is received and transmitted at all times.

3.9.3. User Interaction Flow

As mentioned before, GUI consists of many sections. The User would first decide what action to take. Based on the decided action, the section would be chosen. If the user wants to create a configuration, it will choose the section from the menu which is dedicated for that purpose. The section would open and show a 3D space consisting of modules lying on the floor. The user would be able to select any module and change its position and orientation using the keyboard keys. There is an information box on the side of the section space which shows the selected module’s coordinates and orientation angle. After creating the configuration, the user would press the “achieve configuration” button on the bottom of the section. This will prompt the modules in the system to start taking a group of coordinated actions to achieve the target configuration. Note that while the system is in the process of achieving the said configuration, the user cannot command the system to make a new configuration.

To view the real time pose of each active robot module, the user would choose the appropriate section. This section shows all active modules in the 3D space. Upon selecting any module, the user can view the 3D coordinates and orientation of its current pose. Live pose of each module is always updated and can be used to observe the system’s behavior while a certain configuration is being achieved.

To control the extensions available in the system, the user selects the related section. The section consists of several control boxes. A control box is used to control an extension by changing its control parameters. For example, a gripper extension’s control depends on its motor movement. Its control box would allow the user to view the current motor angle movement and increase/decrease the value. The section responsible for manual control of modules is also part of the GUI application. User selects the section. The module is selected for control. Users then command it to either locomote or rotate its position for a particular distance or rotation angle value, respectively.

Chapter 4: RESULTS

4.1. Prototyping

The prototyping phase involved transforming the conceptual modular design into physical working units through 3D printing, precise component integration, and extensive mechanical-electronic interfacing. A total of three cylindrical modules were fabricated, each capable of operating independently and collectively through magnetic docking. And a gear driven scissor type extension arm is fabricated to show we can attach any type of extention with the module as per requirement.

4.1.1. Module Fabrication

3D Printing & Fabrication Techniques:

All mechanical components including the chassis, wheels, motor mounts, and internal brackets were designed in SolidWorks and fabricated using FDM 3D printing technology. The chosen material was PLA (Polylactic Acid) due to its:

- Lightweight properties
- Good dimensional accuracy
- Sufficient strength for low-load applications
- Availability and ease of post-processing

Layer height was set to 0.2 mm for a balance between resolution and speed, and 10% infill density was used for non-load parts and 40% infill density was used for critical parts to reduce weight.

After printing, parts were cleaned, tolerances were adjusted where necessary (e.g., sanding docking faces), and screw holes were tapped manually.

Component Placement:

- Motors: Two N20 encoder motors were mounted symmetrically on either end of the cylinder, enabling differential drive.
- BU04 AI Thinker: Positioned at the front of the chassis for better signal transmission.
- MPU6050: Centrally placed and fixed to a vibration-isolated base inside the core chassis for accurate orientation readings.
- Motor Driver & Buck module: Mounted in line with the ESP32 to balance weight.
- LiPo Battery (7.4V, 800 mAh): Located at the base of the robot to lower the center of mass and aid in mechanical balance.

Magnet Placement for Docking:

A custom strip containing 13 N55 neodymium magnets (each of size 20×11×3 mm) was embedded at the center height of the robot’s side face. The magnet strip was fixed using a printed holding bracket that maintains magnet alignment and consistent spacing for successful docking. Chamfered edges on the docking interface assisted in smoother physical alignment.

Mechanical Observations:

- Weight: Each fully assembled module weighed approximately 300 grams, within the original design target.
- Balance: Through strategic placement of components (heavier at the base), the center of mass lies below the wheel axis, providing stability in motion and minimizing rolling.
- Shape and Fit: Modules were printed to tolerance, requiring minimal post-processing. The cylindrical design simplified rotational symmetry and docking orientation.
- Final dimensions of a single module: 10 cm (length) × 8.5 cm (diameter)
- Wheel-to-wheel distance: 10 cm, with 8.5 cm wheel diameter

4.1.2. Assembly Process

Mechanical & Electronic Integration:

The internal layout was carefully planned to minimize wiring clutter and vibration risks. All components were mounted either using screws or PLA clips built into the chassis. Motors were fixed using printed holders with dedicated motor screw mounts. The battery compartment was isolated using foam padding.

Internal Wiring and Connectors:

- All wires were soldered directly to eliminate loose connections.
- Routing channels were designed inside the chassis walls to guide wires safely and avoid entanglement with moving parts.
- Care was taken to separate power and signal lines where possible to reduce electrical noise.
- For flexibility, header pins were used for sensor-to-ESP32 connections to enable quick testing and swapping during development.

Cooling and Vibration Considerations:

- Passive cooling was sufficient for all modules as components like the ESP32 and motor driver did not overheat during testing.
- Bearings and tight-fitting chassis segments minimized internal play and reduced mechanical vibrations during movement.
- Foam and zip-tie strain reliefs were used on major connections to reduce disconnection from shocks.

A gear-driven scissor-type extension arm is integrated with the third module of the robot system. This extension mechanism is magnetically docked to the module, enabling both mechanical attachment and electrical connectivity without the need for an independent power source or microcontroller. Instead, power and control signals are supplied directly from the third module, minimizing complexity and weight. The arm is 3D-printed using PLA with a 10% infill density to ensure structural integrity while keeping the overall mass low, thereby maintaining the module’s mobility and balance.

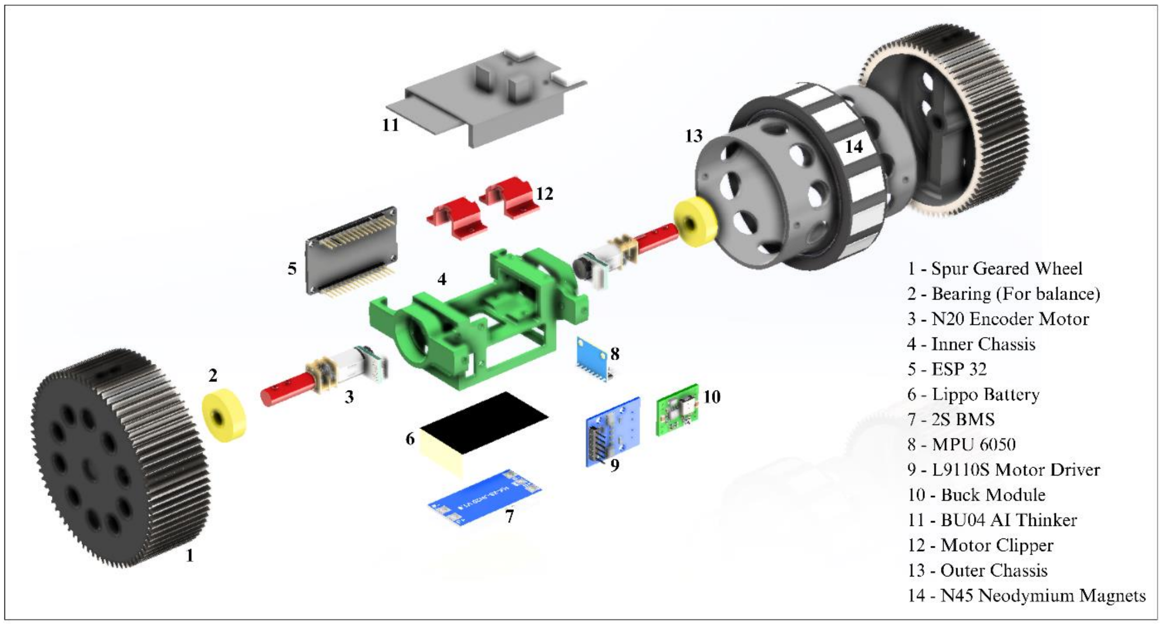


Figure 56*. Exploded View of Robot Assembly (CAD)*

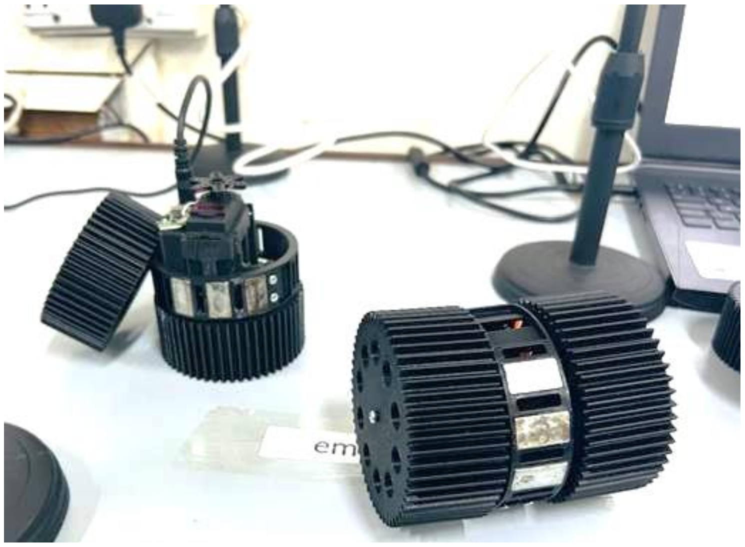


Figure 57*. Finished module*

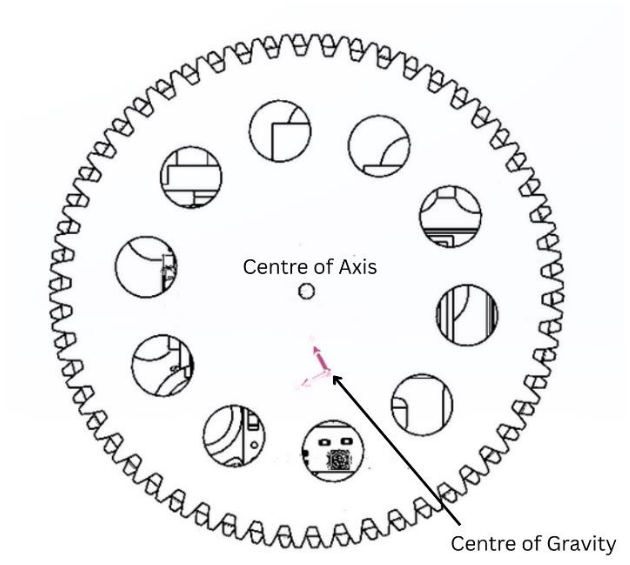


***Figure 58**. Center of Mass Illustration*

4.2. Testing

Various components of the Modular system were tested to validate the system's capabilities. These components included Robot modules, UWB chips, Gripper extension and GUI.

4.2.1. Locomotion of the Robot module

The Robot Modules had a fairly accurate forward and backward movement, with maximum deviations of 0.03-0.06 radians. It also showed good rotational stability while rotating on its axis. While a 360-degree rotation did change the positional coordinates of the module, it only resulted in an average offset of 2-3 centimeters. Since the module's wheels are actuated by servo motors, angle-based wheel movements are also accurate. Modules have shown the ability to move along an inclined surface and over other modules due to its toothed drive wheel and high torque servo motors.

4.2.2. Docking & Connectivity Test

Initially, a robot module was locomoted towards another module. The Moving module got closer to the static module and an attempt was made to dock it with the static module. Although the docking was successful, it fell back to the ground when it tried to move further up the static module. The reason for the docking failure was the lack of strength in N45 magnets which made up a ring of magnets around each module and helped attract other modules. Thereafter, the N45 magnets were replaced by N55 magnets. Upon repeating the same test with the replaced magnets, docking was successful and the locomoting module was able to climb on top of the static module without undocking. Also, if a locomoting module was made to stop at various angles while being docked, it was able to resist gravity and stay in position.

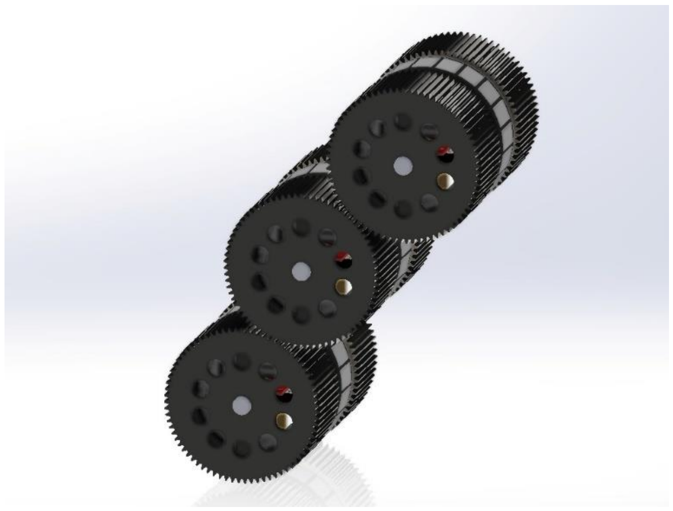


***Figure 59**. Chain configuration for docking strength testing*

To further test the docking strength between modules, both modules were picked off the ground, suspended in the air, and oriented in such a way that one of the modules was pointing downwards. Despite the module experiencing the maximum vertical gravitational force component, it was able to maintain the physical connection with the other module. Another experiment was conducted to observe how many consecutive modules, connected as a chain, a single ground-based module can host without the chain breaking. It was found that modules can host a maximum of 3 modules as a chain without any other ground-based module's assistance. This result is dependent on the magnetic strength of the N55 magnets as well as the shape and size of the module.

4.2.3. Localization Accuracy

The accuracy of UWB sensors was tested. The X, Y, and Z coordinates of the robot module fluctuated by ± 3-10 centimeters. The orientation angle obtained from the IMU sensors showed an accuracy of 1-3 degrees. After 30-40 seconds, the drift in the sensor's output value affected the accuracy of the angle's value.

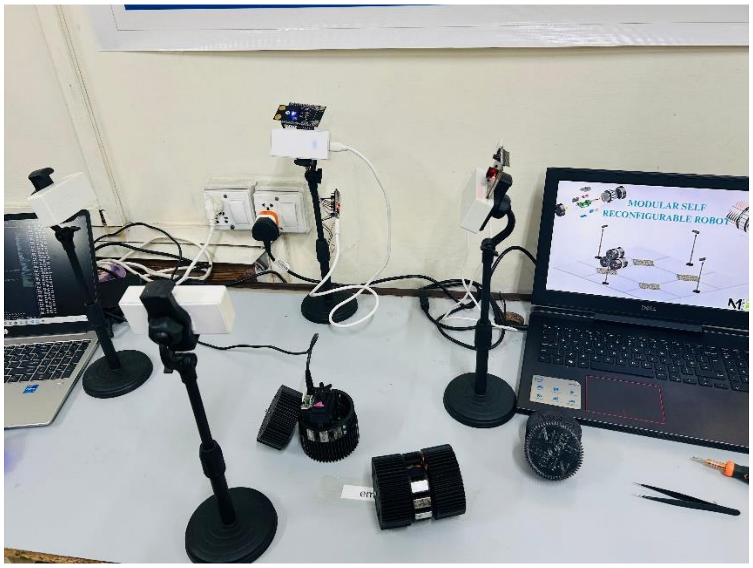


***Figure 60**. Indoor Localization setup with beacons*

4.2.4. GUI Functionality Test

Functionality of the different sections of the web-based GUI were tested. The section showing live poses of robot modules was observed. It showed accurate 3D coordinates and orientation values as fetched from the localization sensors. However, Robot modules in the 3D space showed slight fluctuations in their pose. This was due to the fluctuations in values from the localization sensors. Extension control section was tested. In our case, we tested the gripper extension by controlling the motor's angle movement. Motor was moved via command from the GUI by a certain angle value. The resulting physical angle was measured by a protractor. Both the values were compared. The experiment was repeated at different angles. It was found that there was a negligible difference in the movement angle commanded and the angle achieved. The Create Configuration section showed promising results. Modules were mostly able to configure together, but sometimes there were issues with modules finding an optimal and complete path. The manual configuration section operated as per the requirements. Although the user is needed to spend more time and energy to achieve a given configuration compared to automated configuration, the manual controls are intuitive and user-friendly.

Chapter 5: CONCLUSION AND FUTURE WORK

5.1. Conclusion

The Project successfully designed and implemented a Modular Self Reconfigurable Robot system (MSRR). The MSRR system consists of several Robot modules which can individually locomote as well as join to form three-dimensional configurations. Each robot module is a cylindrical-shaped, differential drive mobile robot equipped with a ring of magnets to dock with its counterpart. The system also consists of a scissor type gripper extension which can perform tasks and make the system beneficial. Sensors like UWB and IMU were successfully utilized to localize the Robot Modules. In order for the user to control and visualize the modular system, a web based GUI has been created which shows each module's pose as well as allows the user to control the system's configuration. Various path planning algorithms have been utilized so that the modules' automated locomotion to its target configuration is achieved. The test results show that Robot modules were able to individually locomote accurately. Furthermore, accurate localization values of the Robot module's pose were achieved. Gripper based extension was shown to have accurate angle based movement. Functionalities of all GUI sections were tested. The GUI was able to properly show incoming data and well as send commands to the modular system. The GUI reliably displayed real-time data and successfully transmitted control commands to the modular system. Overall, the developed MSRR system validated the feasibility of modular robots for reconfigurable structures, providing a strong foundation for expanding to larger module counts and more complex self-reconfiguration algorithms in future work.

5.2. Future work

While the current modular system fulfills the criteria to be considered a Self-Reconfigurable Robotic system, a lot of improvements and additions can be made to the system. The modular system must be enhanced so that it is easy to use, affordable and can be utilized in many more ways. A MSRR system relies on the capabilities of a single Robot module. Hence, making a better module improves the system to a huge extent.

5.2.1. Improvements in the Robot module

The current Robot module is a cylindrical shaped, differential drive mobile robot. While its shape and mechanism allow it to form chain-based structures with convenient docking/undocking, it also makes it hard for it to form complex 3D lattice structures. This is because only ground based modules can rotate or change their orientation, therefore the system loses the ability to form rotational lattice structures.

The docking mechanism limits the number of modules which can attach to a single module. This is due to limited space on the ring of magnets. Moreover, the magnets have limited strength and since a small magnet to magnet connection area is covered, limited force is available to keep the docked modules attracted to each other. FreeBot solves these problems as it allows a larger docking area and employs a bigger and stronger magnet. As discussed earlier, it could not be implemented due to resource constraints, but it is surely a design to be considered when designing a Robot module for a Modular Self Reconfigurable

Robot system. Based on the module shape and size, mechanical latches, electromagnetic attraction and other docking mechanisms can be explored. Hybrid approaches combining multiple docking mechanisms can also be considered.

To make the modular system deployable in the real world, it is crucial to integrate object detection mechanisms in the Robot module. Wherever the system is being used, the robot modules would most likely be in an uncontrolled environment, meaning robot modules would encounter obstacles while locomoting on the ground or switching positions in between the other modules. In those scenarios, the module has to detect those obstacles and adjust its path of movement accordingly. The current Robot modules don't have object detection mechanism and hence are meant to be only operable in controlled environments, which significantly reduces its usefulness.

5.2.2. Other improvements in the system

The number of active modules in the modular system can be increased to allow for complex 3D structures to be achieved. This would require complex algorithms such as A* algorithm for module locomotion path planning and Genetic algorithm for complex configurations in large spaces. Various types of extensions can be added to the system to increase its usability. For example, mechanical tool extensions like drill, sanders and grinders can be used in combination by a system to conduct different manufacturing processes.

To improve the user experience, significant modifications can be done to the Graphical User Interface (GUI). The current Modular System demands the user to command module configurations and extension control using mouse and keyboard keys, which requires the user's time and cognitive attention and focus. An AI-driven, prompt based input system can be developed which can make it easier for the user to command the system. It can be voice controlled as well. Current system only allows the extension's control parameters to be controlled. Like modules, visualization of extensions can be incorporated for better control from the user side.

REFERENCES

[1] J. B. Johnson, "Modular reconfigurable robots: Taking shape," *Knowable Magazine*, 2020. [Online]. Available: https://knowablemagazine.org/content/article/technology/2020/modular-reconfigurable-robots.

[2] J. Liu, "Modular Self-Reconfigurable Robots: Control and Perception for Autonomous Docking and Navigation," Ph.D. dissertation, Dept. of Comp. Sci., Univ. of Sheffield, Sheffield, 2016. [Online]. Available: https://etheses.whiterose.ac.uk/16759/.

[3] J. Lei and S. M. Sait, "Design and development of a new modular self-reconfigurable robot," *Journal of Robotics*, vol. 2017, pp. 1-12, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/epdf/10.1155/2017/5013532.

[4] P. F. Muir, "Modular robots: Capabilities, limitations, and challenges," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 363–386, 2019. [Online]. Available: https://www.annualreviews.org/content/journals/10.1146/annurev-control-053018-023834.

[5] H. Hamada, N. Watanabe, and S. Egi, "Development of a new type of modular self-reconfigurable robot," *Advances in Mechanical Engineering*, vol. 8, no. 7, 2016. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/1687814016659597.

[6] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, and H. Lipson, "Modular self-reconfigurable robot systems: Challenges and opportunities for the future," in *Proceedings of the 2007 AAAI Spring Symposium*, 2007, pp. 54–58. [Online]. Available: https://www.researchgate.net/publication/3344812_Modular_Self-Reconfigurable_Robot_Systems_Grand_Challenges_of_Robotics.

[7] M. Imagawa, M. Tokuda, T. Murooka, and M. Yamaoka, "Distributed self-reconfiguration of M-TRAN III modular robotic system," *Scientific Reports*, vol. 7, 2017. [Online]. Available: https://www.nature.com/articles/s41598-017-14220-3.

[8] Y. Liu, H. Liu, and Y. Wang, "Design of a novel modular self-reconfigurable robot system and its locomotion planning," *Robotics and Autonomous Systems*, vol. 119, pp. 1-14, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0921889019303379.

[9] H. Wang and D. Zhu, "Control and motion planning of modular self-reconfigurable robots," *Robotics and Autonomous Systems*, vol. 61, no. 11, pp. 1134-1145, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0921889013001632.

[10] Y. Suzuki, Y. Tomita, T. Fujita, and Y. Ishikawa, "Distributed control of self-reconfigurable M-TRAN III modular robots," in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 5134–5141. [Online]. Available: https://tohoku.elsevierpure.com/en/publications/distributed-self-reconfiguration-of-m-tran-iii-modular-robotic-sy.

[11] S. Murata, K. Kakomura, and H. Kurokawa, "Self-reconfigurable modular robot M-TRAN and its motion design," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, 2003, pp. 1602–1607. [Online]. Available: https://www.researchgate.net/publication/4036552_Self-reconfigurable_modular_robot_M-TRAN_and_its_motion_design.

[12] C. J. Pappas, R. U. Grier, and A. W. Klaus, "3D M-Blocks: Self-reconfiguring robots capable of locomotion via pivoting in three dimensions," in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, 2015, pp. 1932–1939. [Online]. Available: https://www.researchgate.net/publication/282918517_3D_M-Blocks_Self-reconfiguring_robots_capable_of_locomotion_via_pivoting_in_three_dimensions.

[13] S. A. Smith and J. K. Doe, "A Novel Approach to Modular Self-Reconfigurable Robots," arXiv preprint arXiv:2310.09743, 2023. [Online]. Available: https://arxiv.org/pdf/2310.09743.

[14] B. Salemi, M. Moll, and W.-M. Shen, "Superbot: A deployable, multi-functional, and modular self-reconfigurable robotic system," in *Proceedings, 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 3636–3641.

[15] E. H. Østergaard, K. Kassow, R. Beck, and H. H. Lund, "Design of the ATRON lattice-based self-reconfigurable robot," *Autonomous Robots*, vol. 21, no. 2, pp. 165–183, 2006.

[16] M. W. Jorgensen, E. H. Østergaard, and H. H. Lund, "Modular ATRON: Modules for a self-reconfigurable robot," in *Proceedings, 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 2004, pp. 2068–2073.

[17] D. J. Christensen, D. Brandt, and K. Stoy, "Towards artificial ATRON animals: scalable anatomy for self-reconfigurable robots," in *The RSS Workshop on SelfReconfigurable Modular Robots*, 2006.

[18] G. G. Ryland and H. H. Cheng, "Design of iMobot, an intelligent reconfigurable mobile robot with novel locomotion," in *Proceedings, 2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 60–65.

[19] J. Davey, N. Kwok, and M. Yim, "Emulating self-reconfigurable robots - Design of the SMORES system," in *Proceedings, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4464–4469.

[20] H. H. R. Mahmud, D. K. S. Rajapakse, and R. A. S. Al-Ani, "A Novel Approach to the Design of Modular Self-Reconfigurable Robots," in *Proc. 2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013, pp. 2494-2499. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6696971.

[21] C. Parrott, "A hybrid and extendable self-reconfigurable modular robotic system," Ph.D. dissertation, Dept. of Automatic Control and Systems Engineering,

Univ. of Sheffield, Sheffield, UK, Sep. 2016. [Online]. Available: https://sci-hub.se/10.1007/978-3-319-73008-0_28.

[22] C. Parrott, R. Groß, and T. J. Dodd, "A hybrid and extendable self-reconfigurable modular robotic system," in *Proc. 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, IL, USA, 2014, pp. 6943114. [Online]. Available: https://sci-hub.se/10.1109/iros.2014.6943114.

[23] C. Parrott, "A hybrid and extendable self-reconfigurable modular robotic system," Ph.D. dissertation, Dept. of Automatic Control and Systems Engineering, Univ. of Sheffield, Sheffield, UK, Sep. 22, 2016. [Online]. Available: https://sci-hub.se/10.1007/978-3-319-73008-0_28.

[24] R. A. H. Alami, R. M. A. M. Jaafar, and A. S. Ibrahim, *Modular Robots: Theory and Practice*, 1st ed. London, UK: Springer, 2021. [Online]. Available: https://www.google.com.pk/books/edition/Modular_Robots_Theory_and_Practice/vCdAEAAAQBAJ?hl=en&gbpv=0.

[25] G. Liang, H. Luo, M. Li, H. Qian, and T. L. Lam, "FreeBOT: A freeform modular self-reconfigurable robot with arbitrary connection point - design and implementation," in *Proc. 2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 2020, pp. 6506–6513, doi: 10.1109/IROS45743.2020.9341129.

[26] D. Zhao and T. L. Lam, "SnailBot: A continuously dockable modular self-reconfigurable robot using rocker-bogie suspension," in *Proc. 2022 Int. Conf. on Robotics and Automation (ICRA)*, Philadelphia, PA, USA, 2022, pp. 4261–4267, doi: 10.1109/ICRA46639.2022.9811779.

[27] Y. Tu, G. Liang, and T. L. Lam, "FreeSN: A freeform strut-node structured modular self-reconfigurable robot - design and implementation," in *Proc. 2022 Int. Conf. on Robotics and Automation (ICRA)*, Philadelphia, PA, USA, 2022, pp. 4239–4245, doi: 10.1109/ICRA46639.2022.9811583.

[28] G. Liang, D. Wu, Y. Tu, and T. L. Lam, "Decoding modular reconfigurable robots: A survey on mechanisms and design," *arXiv preprint arXiv:2310.09743*, Oct. 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2310.09743.

[29] K. Gilpin, et al., "Datom: A Deformable Modular Robot for Building Self-Reconfigurable Programmable Matter," *ResearchGate*. [Online]. Available: https://www.researchgate.net/publication/341231209_Datom_A_Deformable_modular_robot_for_building_self-reconfigurable_programmable_matter.

[30] K. Gilpin, et al., "Datom: A Deformable Modular Robot for Building Self-Reconfigurable Programmable Matter," *Academia*. [Online]. Available: https://www.academia.edu/82409904/Datom_A_Deformable_Modular_Robot_for_Building_Self_reconfigurable_Programmable_Matter.