



**Autonomous Collaborative Ground Traversal of
Discontinuous Terrain**

Final Report

TEAM A

Dhanvi Sreenivasan | Hari Shankar Ravisankaran | Sandhya
Vidyashankar | Sankalp Chopkar | Sudhansh Yelishetty

Sponsors: Prof. Zeynep Temel | Prof. Katia Sycara | Sha Yi

Date: Dec 13, 2023

**Master of Science
Robotic Systems Development**

**Carnegie Mellon University
The Robotics Institute**

Abstract

Robot swarms enhance individual robot capabilities through collaborative efforts. The introduction of physical coupling between robots further extends their effectiveness in navigating unstructured environments. This is particularly advantageous in complex environments such as mines, where inspections pose challenges. Rather than allocating valuable human hours for on-site inspections in mines, deploying a collaborative swarm of robots proves to be a more efficient and effective solution.

In this project, we design a robot swarm capable of crossing gaps in the terrain through a process of autonomous coupling. The swarm is controlled by a centralized server, which orchestrates the system's behavior in the coverage, coupling and crossing phases. The platform of choice is the Khepera IV robot made by K-Team, graciously provided to us by our sponsor Prof Katia Sycara

The key subsystems are the coupling mechanism and supporting electronics, the planning stack which includes a high level behavior planner, a task allocation module and a multi-agent path planning module. To execute these plans and to perform highly accurate autonomous coupling, we employ finely tuned PID controllers. Finally, to make all of this possible, we have a communication pipeline to interface the agents with our central server. In this report you will find the high level overview of each of the subsystems and the process of managing our progress

By the end of this project, we demonstrated coverage of points of interest without collisions, along with fully autonomous alignment and coupling of agents. The agents then cross the gap, decouple on their own and continue their coverage tasks.



Table of Contents:

Abstract	2
1. Project Description	4
2. Use Case	4
3. System-Level Requirements	5
3.1 Mandatory Requirements	5
3.1.1 Functional Requirements	5
3.1.2 Performance Requirements	6
3.1.3 Non-Functional Requirements	6
3.2 Desirable Requirements	7
4. Functional Architecture	7
4.1 System Architecture	7
4.2 Server functional architecture	8
4.3 Agent functional architecture	9
5. Trade Studies	10
6. Cyber-physical architecture	11
6.1 Server cyber-physical architecture	11
6.2 Agent cyber physical architecture	12
7. System Description and Evaluation	13
7.1 FVD Performance	13
7.2 Overall system depiction	14
7.3 Subsystem descriptions	15
7.3.1 Controllers	15
7.3.2 Coupling Mechanism	16
7.3.3 Mapping and Localization	17
7.3.4 Preprocessing	17
7.3.5 Task allocation	17
7.3.6 High-level Planner	18
7.3.7 Multi-robot planning	19
7.3.7 Autonomous Coupling	20
7.3.8 Fleet management system	21
7.4 Modeling, Analysis and Testing	21
7.5 SVD performance evaluation	22
7.6 FVD performance evaluation	23
7.6 Strong and Weak Points	23
8. Project Management	25
8.1 Work Breakdown Structure	25
8.2 Project Management: Schedule Status	26
8.3 Project Management: Fall Test Plans	27
8.4 Project management: Parts list and Budget	27
8.5 Risk Management	29
9. Conclusions	31
9.1 Lessons Learnt	31
10. References	31

1. Project Description

Robot swarms have been shown to improve the ability of individual robots by inter-robot collaboration. Introducing physical coupling between robots can further extend their ability in an unstructured environment. This is particularly beneficial in inspection scenarios, especially in mines, where the environment is complicated and thus challenging. Instead of having the crew spend precious day hours inspecting several sites in a mine in person, they can deploy a swarm of robots to do the same in a more efficient and collaborative manner.

Rather than sending in a team of people for inspection at each site, it is more efficient to send a swarm of robots to collect more information at each point of interest to the crew. Each agent in the system will inspect a set of allocated points of interest, and once every point is done, they will return all the information to the end users, using which further plans can be made.

Previously, PuzzleBots [1][2] was proposed for the same. It is a robotics swarm system where the robots can dynamically couple with each other to form bridges and decouple to perform individual tasks but on a small scale. This system can either be used to build bridges for the transportation of other materials, or to get the robots from one side of a gap to the other. The aim of this project is to investigate the physical coupling of robot swarms on a larger scale, as well as implement and test existing control and planning algorithms on a larger system.

2. Use Case

Consider a mine with several hotspots, where a hotspot is a location with high concentrations of mineral deposits. These minerals are of critical importance and the mining crew is on a time crunch. That being said, each hotspot in the mine must be inspected in detail before approaching the next steps.

Mines, in general, can be narrow, in caves, and can have cracks or fissures on the ground or other surfaces. Due to rugged terrains in such unstructured environments, traversing these conditions can be potentially risky for humans, e.g., collapse of overhead rock structures. The mining crew needs to inspect each hotspot but will lose out on time and need to find a better solution. An autonomous distributed solution.

Bring in PuzzleBots. The crew inputs a blueprint of the map of the mine. This would typically be an occupancy grid of the mine with hotspots marked in them. All agents in the system start from the base station at the entrance of the mine. Each agent gets allocated hotspots optimally, and collision-free optimal trajectories are planned. The agents branch off from the swarm to inspect individual hotspots. In case of emergencies involving imminent collisions, each agent is equipped with a collision avoidance system. They spend a predetermined fixed amount of time at the hotspots before being allocated new ones.

It is very likely that hotspots are located across fissures in the mine, which will be referred to as gaps. Should there be any hotspots on either side of any gaps, the system determines the feasibility of getting there. If feasible, the system plans for a group of agents to physically couple with each other and cross the gap. After that, they split up, and follow their respective planned trajectories to inspect individual hotspots. After inspecting the entire map, the agents return to the base station.

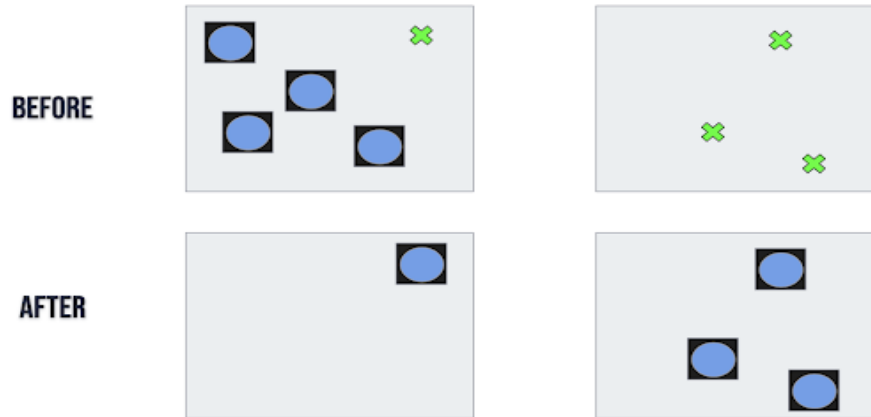


Figure 1. Use Case Illustration

3. System-Level Requirements

Our system-level requirements were elicited from the use case, discussions with our sponsors, and internal team meetings based on the time and budget constraints to the project timeline. All the requirements are shown in Table 1 to Table 3.

3.1 Mandatory Requirements

3.1.1 Functional Requirements

Table 1. Functional Requirements

	Requirement
MF1	The system shall localize agents in a given map.
MF2	The system shall route agents and avoid collisions.
MF3	The system shall determine feasible gaps.
MF4	The system shall determine and achieve coupled configurations.
MF5	The system shall cross gaps.
MF6	The system shall reach given regions of interest.

3.1.2 Performance Requirements

Table 2. Performance Requirements

	Requirement
MP1	The system will cross gaps up to 1.3 agent lengths.
MP2	The system will have 0 unplanned collisions between agents.
MP3	The system will achieve formations with at least 3 robots.
MP4	The system will cross feasible gaps 75% of the time.
MP5	The system will reach all POIs 75% of the time.
MP6	The coupling mechanism will bear the weight of one agent.
MP7	The coupling mechanism will self-align against heading errors of 2.5 degrees and position error of 3.5 mm.
MP8	The system shall handle gaps with varying widths between 9 and 19 cms
MP9	The system shall determine optimum crossing configurations to minimize overall system time

- MP1 was modified from before so as to not damage the robots’ hardware over time. The robots rely on momentum to cross gaps. With larger gaps, the robots hit the edges of the gaps at a larger impact. We observed that the caster wheels of the robots pop out of their sockets from time to time when trying to test this out. In order to not damage the robots further, we rescoped this requirement.
- MP8 and MP9 were added as part of our push to demonstrate smart task allocation strategies and the ability of the system to dynamically reach configurations depending on the task layout. These were primarily reach goals which were achieved in the encore

3.1.3 Non-Functional Requirements

Table 3. Non-Functional Requirements

	Requirement
MN1	The weight of an agent shall be minimal.
MN2	The coupling mechanism shall consume a low amount of energy.
MN3	The system shall be scalable.

MN4	The system shall be easily maintainable.
MN5	The team shall maximize learning and fun throughout the project through flexible work plans across subsystems.

3.2 Desirable Requirements

We have chosen not to go with any desirable requirements because the project is highly complex.

4. Functional Architecture

4.1 System Architecture

The system is divided into two blocks: the server and the agents. A brief overview of the functions of agents and the server is shown below:

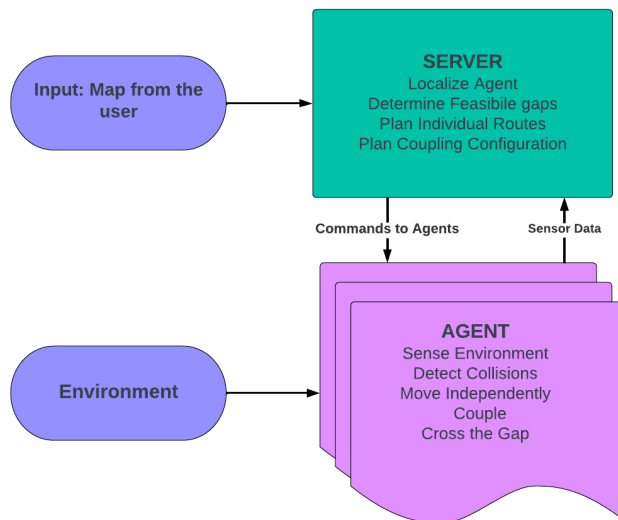


Figure 2. Functional Architecture of the System

4.2 Server functional architecture

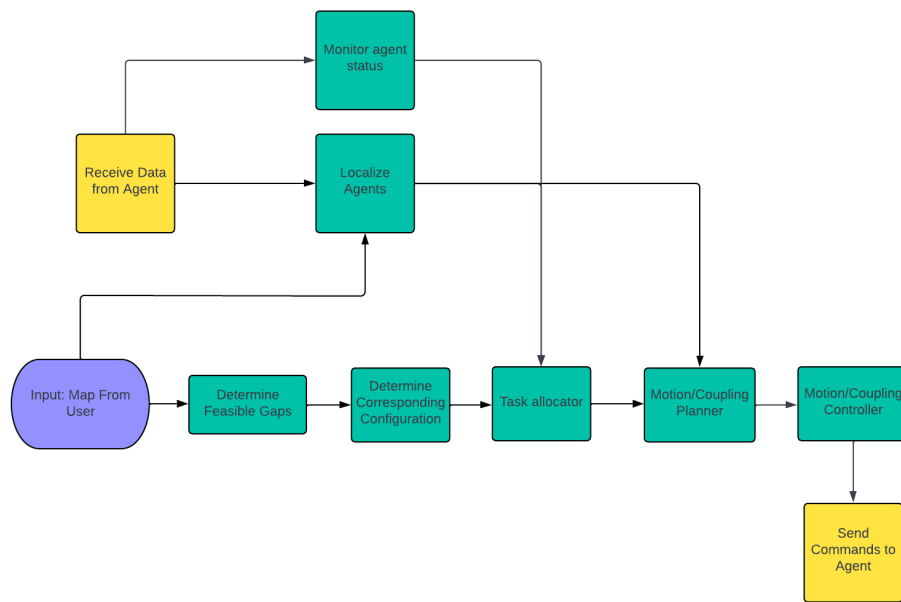


Figure 3. Functional Architecture of the Server

The user provides an initial map of the environment to the server which includes information about the gaps, the points of interest, and the initial agent poses. The server receives sensor data from the agents. The server monitors the agent's status and localizes each agent within the environment. Based on the input map, the server determines the feasible gaps and also the configuration required to cross the gaps. The task allocator takes in this information and allocates specific tasks to each individual agent.

The tasks assigned are then sent to the planner block which generates either collision-free paths for agents to cover the points of interest or coupling and gap-crossing plans if the task is to cross the gaps. To execute the planned paths, the controller block computes the error between the desired and current agent pose to generate linear and angular velocity commands for movement and coupling commands to enable coupling.

4.3 Agent functional architecture

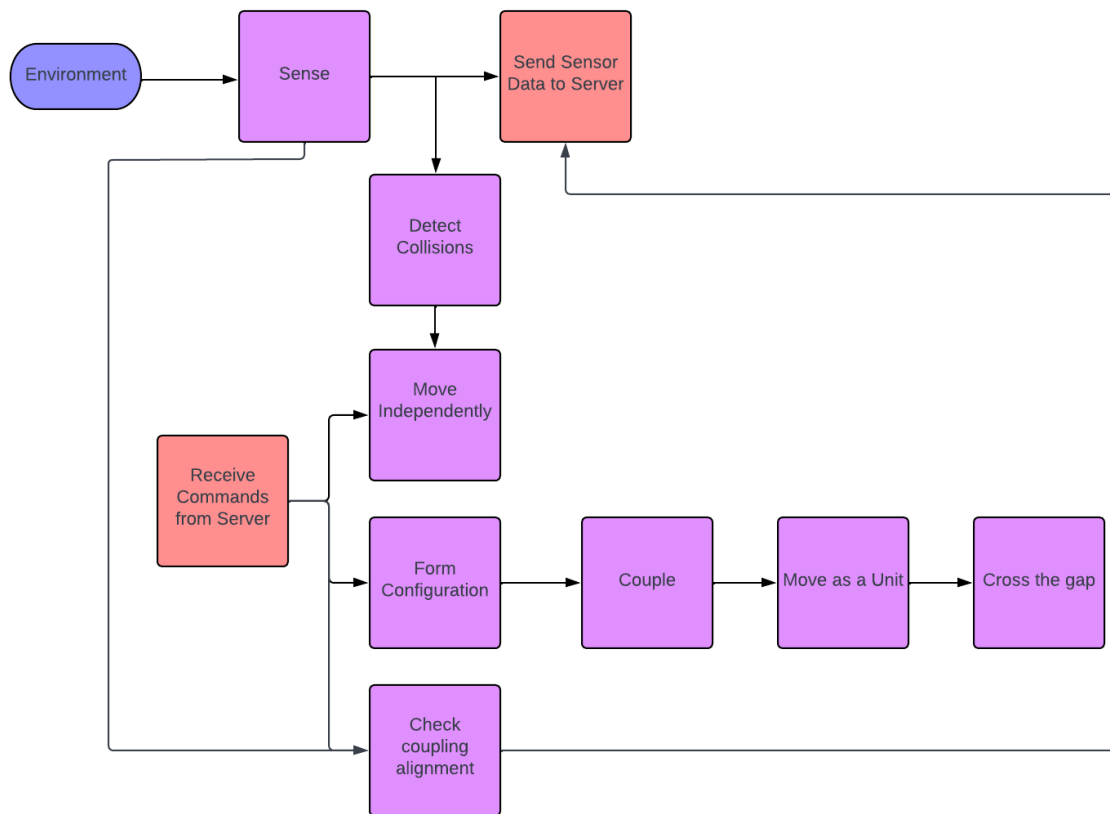



Figure 4. Functional Architecture of the System

The agent receives linear, angular velocities, and coupling commands from the server. The linear and angular velocities are used to generate the motor commands that enable the agent to move independently to cover points of interest, form configuration, and move as a unit depending upon the state of the agent. When the agent moves, the sensor readings are continuously transmitted to the server. The sensor data is also used to detect imminent collisions and avoid them. The coupling commands are used to actuate the coupling motors to enable successful coupling between the agents. To refine localization estimates for coupling, the agent also performs coupling alignment checks using the sensor readings which are then sent to the server.

5. Trade Studies



Solution	Variable Length	Jumping	Drones	Coupled Swarm
Favourable	Simple Controls	Fast No constraints on the gap	Fastest No constraint on gap Self sufficient	No constraints on gap LOW work volume Variable configurations
Unfavourable	Weight Distribution Maintaining Balance Life Cycle Constraints on gap	Vertical Clearance Difficult to control Mechanical wear and tear Payload constraints	Vertical clearance Susceptible to weather Inefficient for ground tasks	Slow Difficult to coordinate

Figure 5. System-level trade studies

For our system-level trade studies, we followed a bottom-up approach. We started with the use case of gap-crossing, and identified high-level approaches to designing a solution to solve the core problem. We identified 4 approaches to do the same -

- **Variable-Length robots** - robots that can extend/contract accordingly to cover the gap width. These are tricky to control since the weight distribution changes with time. Further, the heavy reliance on mechanical parts means a lower life cycle
- **Jumping robots** - robots that can jump to clear gaps. These are difficult to control, require large vertical clearances and have payload constraints. Further, repeated jumping and landing forces will cause wear and tear on the joints
- **Drones** - though the quickest solution, drones require vertical clearances which make them difficult to deploy in tightly constrained environments like caves or mines.
- **Coupled swarms** - though slow and difficult to coordinate, a swarm has many advantages, primarily scalability and low work volume. A swarm can also have different configurations depending on the hardware which makes it more flexible to unpredictable terrain

Based on the trade studies, we are using Khepera 4 robots for the agents. The platform was primarily designed for testing multi-agent systems, so they are suitable for our purpose. The sensors include wheel encoders, a 3-axis gyroscope, a 3-axis accelerometer, infrared sensors, ultrasonic sensors, laser range finders, RGB camera. All of these sensor data are packaged and transmitted to the server. The agent runs a collision detection service on its own to detect

imminent collisions and avoid them. It also monitors its own status based on the server commands and sends messages to the server in case of any failure. The velocity and coupling commands from the server are passed to the respective controllers which convert them to actuator velocities. For refinement of localization estimates during coupling, the agent also runs an alignment-checking algorithm based on Aruco markers and sends the processed data to the server.

6. Cyber-physical architecture

6.1 Server cyber-physical architecture

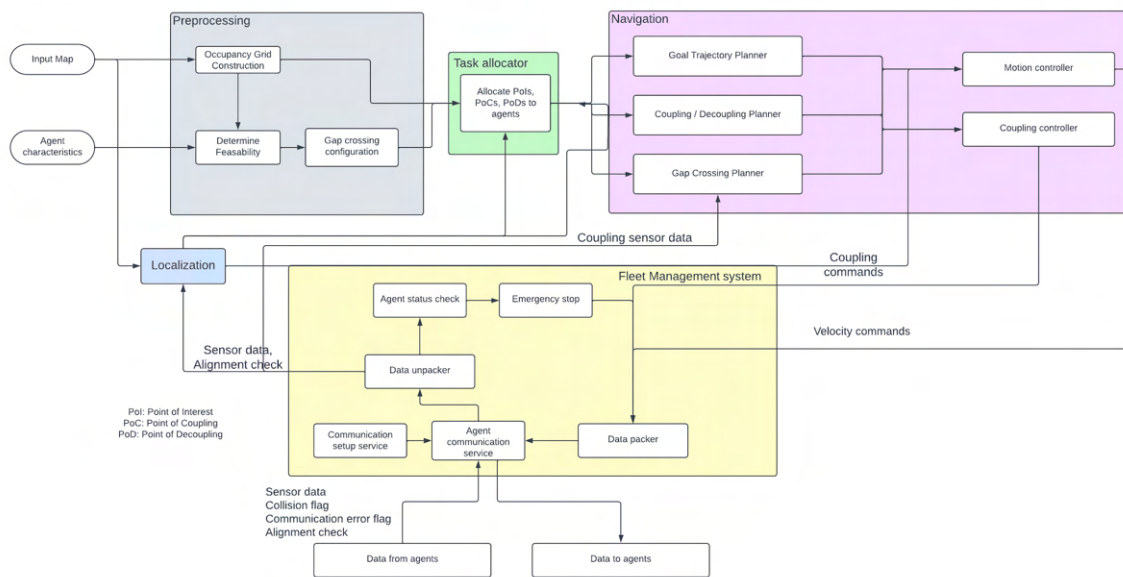


Figure 6. Server Cyber-Physical Architecture

The server acts as the brain of the system and controls the behavior of the individual agents. It takes an initial map of the environment and a config file from the user. The map contains information about the gaps and points of interest. The config file contains information about all the available agents. The FMS block uses the config file to set up communication service between the agents and the server. FMS handles the packing of data that needs to be sent to the agents and the unpacking of data from the agents to feed them into the corresponding subsystems. The map and the sensor information are used by the localization subsystem to determine the poses of each agent. The FMS also monitors the individual agents to check for communication failures.

The map provided by the user is preprocessed into Voronoi diagrams/connected grids to feed into the task allocation block. The preprocessing block also determines the feasible gaps in the map and the corresponding configuration required to cross those gaps. Based on these, the task

allocator assigns tasks to each individual agent. The tasks can be covering points of interest or reach points of coupling and decoupling. Allocation is formulated as a modified Multiple Traveling Salesman problem to account for the gaps and coupling behavior.

The assigned tasks are then passed to the navigation subsystem. The navigation subsystem includes two major blocks: the trajectory planner and the controller. Based on the assigned tasks, for each agent, the goal trajectory planner will be used to generate collision-free paths to cover points of interest, the coupling planner will be used to generate coupling paths and commands, the gap crossing planner will be used to generate paths for the coupled robots to cross the gaps. The controller takes in the generated paths and the actual agent pose to compute the desired linear, angular velocities for agent movement and coupling commands for enabling coupling. These commands are then passed to the FMS to send to the agents.

6.2 Agent cyber physical architecture

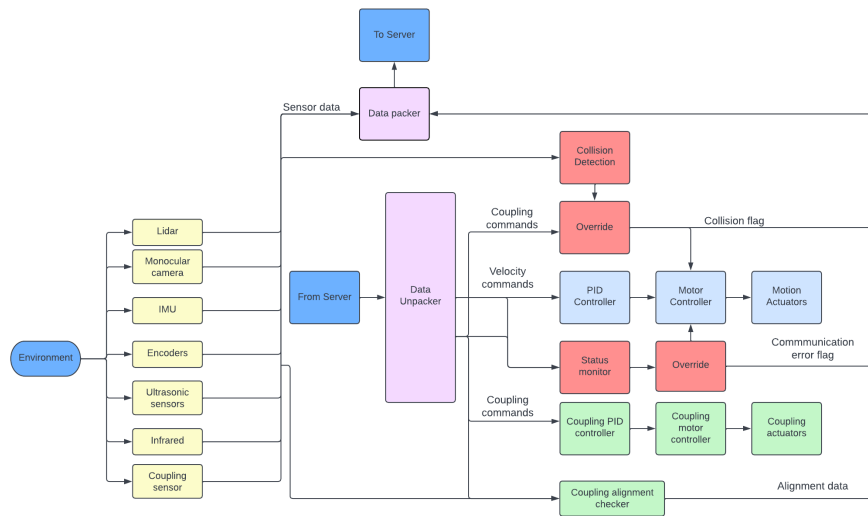


Figure 7. Agent Cyber-Physical Architecture

7. System Description and Evaluation

7.1 FVD Performance

The following functional requirements were demonstrated in the Fall Validation Demo

Table 4. Fall semester Functional Performance Requirements

	Requirement
MP1	The system will cross gaps up to 1.3 agent lengths.
MP2	The system will have 0 unplanned collisions between agents.
MP3	The system will achieve formations with at least 3 robots.
MP4	The system will cross feasible gaps 75% of the time.
MP5	The system will reach all POIs 75% of the time.
MP6	The coupling mechanism will bear the weight of one agent.
MP7	The coupling mechanism will self-align against heading errors of 2.5 degrees and position error of 3.5 mm.
MP8	The system shall handle gaps with varying widths between 9 and 19 cms
MP9	The system shall determine optimum crossing configurations to minimize overall system time

The functional performance requirements covered all subsystems including task allocation, localization, planning, control and mechanical subsystems. We were successfully able to demonstrate all the above requirements during FVD and FVD encore.

7.2 Overall system depiction

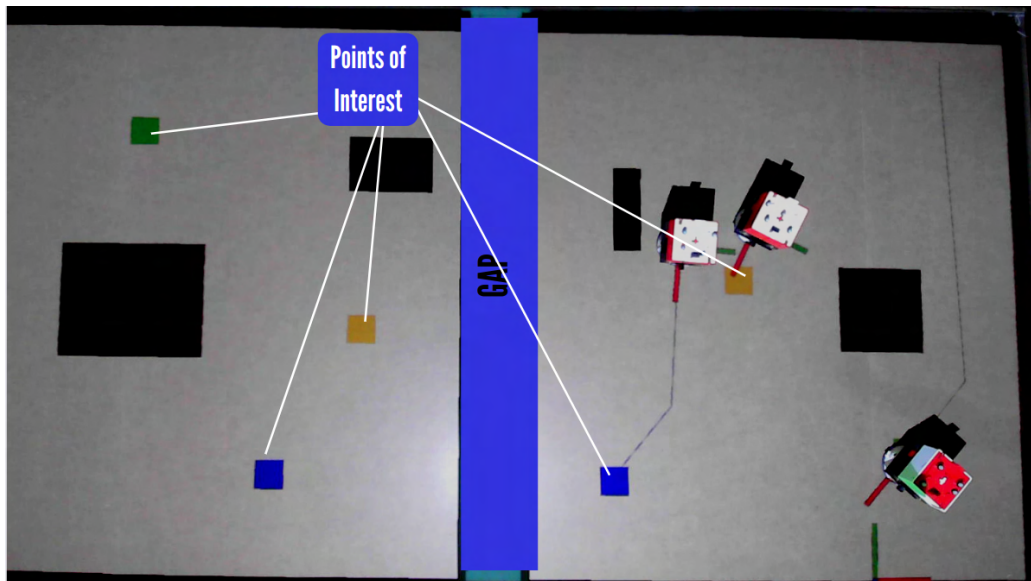


Figure 8. Overall System depiction

Figure 8 shows the overall system depiction. There is a map which is clearly divided into two regions, separated by a gap of upto 19 cm in length (can vary). The agents all start from one side of the gap at arbitrary start locations, and are given tasks or PoIs on either side of the gap.

The system first splits the tasks into two groups, and uses the MTSP paradigm to create task queues for each agent on either side of the gap. The CBS planner is then used to plan routes from the start locations to each task in the agent's queue in a collision-free manner.

Once the agents have completed all their tasks on one side of the gap, a set of agents is optimally chosen to cross the gap to minimize overall system time. The agents then head to their crossing positions from where the autonomous coupling routine is called

The autonomous coupling routine brings agents into position using a PID controller with hierarchical refinement. It couples all the agents in a pair-wise fashion to form a chain of n agents (in our case up to 3) to cross the gap. Once the last agent has coupled, the system issues a cross command, and uses the position feedback from our localization system to identify when the agents have safely reached the other side

Finally, the decoupling service is called and the agents are then fed their remaining tasks via the CBS planner. All this is underpinned by communications via Python scripts from the on-board RPis to the central server. The packets are sent in the UDP format and a timestamp check is in place to ensure only the latest data is used by the server

7.3 Subsystem descriptions

7.3.1 Controllers

The goal of the controller subsystem is to execute the collision free paths given by the planner reliably and ensure time synchronization. It also needs to communicate with the planning subsystem to reinitiate the planning process whenever one of the agents reach their goal location. We tried two different approaches using PID and pure pursuit controller. The pure pursuit method did not give good results due to the fact that the use of lookahead distance does not guarantee collision avoidance. The current implementation for SVD used a dual PID control to control the linear and angular velocity independently. The switching between the two controllers takes place based on an error threshold for the linear and angular component. Additionally, time synchronization was implemented to ensure that the agents avoid collisions between themselves.

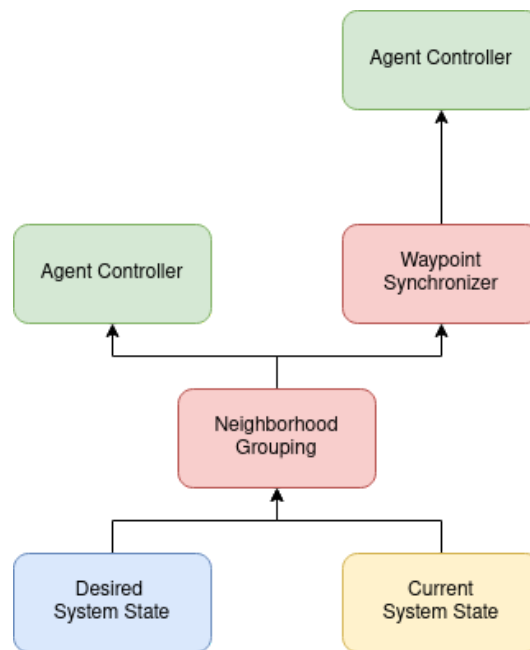


Figure 9. Dynamic Grouping Control Logic

New to the fall semester is our dynamic grouping logic for improving the performance of our controllers. The key insight from our testing during the spring was that agents only needed tight-knit coordination when they are close to each other. Considering this, we constantly check the pairwise distance between agents. If two agents fall within the threshold, the waypoints are discretized so that they are closer together, ensuring coordinated behavior.

However, if an agent has no other agents close by, it will follow a sparse discretization enabling it to move much quicker to its next waypoint. This dynamic grouping maintains all the guarantees of collision-free paths from the CBS planner, but is faster to execute.

7.3.2 Coupling Mechanism

This subsystem is a hardware addition to the robot mobility platform Khepera IV. It is responsible for facilitating the physical coupling between robot agents, allowing them to collaborate and traverse discontinuous terrain. The coupling mechanism is inspired by how train bogies are connected, using a pin-and-hole mechanism for initial latching, with a set of linear bearings for final coupling. This enables the system to couple reliably, and allows one agent to fully bear the weight of another through the precisely machined linear bearings from Igus

The electromechanical system consists of ESP8266- Node MCU (Microcontroller with WIFI module), L298N (motor driver for linear actuator), Raspberry Pi Zero v2, Actuonix micro-linear servo and a NEMA 11 linear actuator. The power for the microcontrollers is through a USB power bank, while the actuators use a 11.1v LiPo battery with appropriate relays.

The main architectural change is the introduction of the RPi and the micro-linear servo into the circuit. The RPi is now the main communication interface between the Kheperas and the central server, through a dedicated python script for each agent. The RPi is then connected via serial to the ESP module, sending it commands to actuate the various actuators and read the limit switch states, which is packaged and sent to the server as a single state packet.

With the present coupling mechanism, 3 agent 1 dimensional coupling configuration can be achieved post pre-alignment and can successfully cross 19 cm gaps.

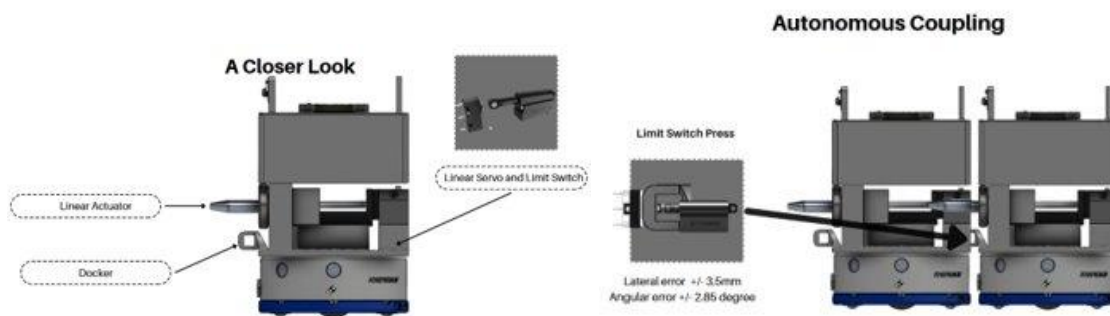


Figure 10. Coupling Mechanism Overview

7.3.3 Mapping and Localization

The pose of every agent on the map needs to be determined precisely to enable coupling between the robots and cover the points of interest efficiently. The user provides a map and the initial positions of the agents. The sensor readings from the agent will be used for getting the pose estimate. The wheel encoders provide an initial first estimate of the robot's pose. Then the data from the laser range finder is used to refine the estimate of the change in the pose by matching scans of consecutive frames. The wheel encoder provides the odometry and the laser range finder (LRF) and the odometry together help in building the map, which is fed downstream to the localization, task allocation, and planning subsystems. We also explored OpenCV-based map building methods to enable quick variability of maps.

The laser scans are also matched with the provided map to provide localization information, coupled with the wheel odometry. This process is done separately for each agent using the map provided by the user. As of now, since we require very accurate localization for multi-agent motion planning and coupling, we use the VICON tracker system by placing trackable patterns as markers on the robots. During the coupling process, the localization precision should be high. Therefore, when the robots are sufficiently close to each other during coupling, additional redundant methods like Aruco marker-based pose estimation will also be used along with the previously mentioned method.

7.3.4 Preprocessing

The preprocessing block involves the processing of the map into Voronoi diagrams/connected grids or any other format required by the planning block. Additionally, the map also contains information about the gaps in the system. The preprocessing block determines all the feasible gaps in the map based on the number of agents available and also determines the configuration required by the agents to cross those feasible gaps. The infeasible gaps will be considered obstacles in the map. The outputs from the system include the processed map, feasible gaps, and their corresponding configuration.

7.3.5 Task allocation

The task allocator assigns uncompleted tasks to each agent based on the inputs from the preprocessing block while minimizing the overall time taken. The overall task allocation pipeline can be broken down into the following steps

- Ingesting start locations, goal locations and the map from a ROS service call
- Processing the map in OpenCV to determine the gap location and size. This is used to then split the tasks into two groups on either side of the gap. A key assumption is that all agents start on one side of the gap since we use the centroid of the agents to determine the grouping of the tasks

- Determining free space for coupling and crossing - we discretize the map in units of couple agent lengths to determine points where coupling can take place. A potential crossing point is when we find two such points on either side of the gap, the vector between which is parallel to the gap orientation.
- For each potential crossing configuration, we determine the cost of the system to use that point by solving a set of MTSP instances and adding a fixed cost for coupling. The cost metric used is the makespan of the system. After iterating through all possibilities, we obtain the number of agents and the location of crossing
- We use this to create pseudo-tasks for coupling, crossing and decoupling. These are added to the respective task queues with different IDs to let the planner know these are not part of the CBS tasks and require different behavior
- The final task queues for each agent are then sent via a ROS service to the high level planner for coordinating behavior

7.3.6 High-level Planner

The high-level planner or the coordinator sits atop the task allocator and the CBS planner, coordinating the overall system behavior through a state-machine. The behavior can be broken down as follows

- Initially, the HLP calls the task allocator and requests task queues for agents
- On receiving task queues, it splits the queues for each agent into the first half and second half for the CBS planner. For agents that are not crossing, the second half task queue is just its final location.
- Based on the list of crossing agents, the HLP also determines the order of crossing agents based on the distance from the gap centroid of the crossing tasks. This is used by the coupling node to determine order of operations
- The HLP then calls the first set of tasks through the CBS planner service, which returns a flag once all agents complete their set of tasks
- On receiving the flag, the HLP calls the CBS planner to get the crossing agents into position for coupling
- Once the crossing agents have reached, the HLP makes subsequent service calls to couple the pairs of agents according to the ordered agent list
- Once all the agents have coupled successfully, a velocity command is published to the crossing agents at max speed to cross the gap, controlled through position feedback of the middle agent
- Once the middle agent reaches the center of the decoupling zone, a stop command is sent, and decoupling is initiated
- Once decoupling is done, the CBS service is called with the second set of task queues and the system proceeds to completion

7.3.7 Multi-robot planning

The planning subsystem takes as input the map and the task queues determined by the task allocator for each agent and formulates collision-free paths for each agent. This is done in a sequential manner, making the entire process piecewise-optimal. The planner is based on Conflict Based Search, a high level decision tree to resolve conflicts between agents and re-route them if required. Individual agent plans are found using Theta*, an extension of A* search that generates smoother, shorter paths at the cost of increased computation. This is achieved using line of sight checks with static and dynamic obstacles.

Since the spring semester, the planner has undergone two major improvements -

1. Parallelization - since the planner was taking quite a while to compute collision-free paths between to goals due to slow traversal of the high-level CBS search tree, we implemented a multi-threaded version of the breadth-first search algorithm to quickly find feasible solutions. We would then pick the best solutions from the ones found. This means that the algorithm is no longer guaranteed to be the optimal solution, but practically almost always gives a very close solution quite quickly
2. Focal search for Theta Star - We implemented a focal list along with the traditional open and closed lists for the low-level search, greatly improving the speed of node exploration in the absence of a biased heuristic. This means the solution is bounded-suboptimal, but in practice since the controllers take a large amount of time to execute a path anyway, we felt this trade-off was worth it for our system

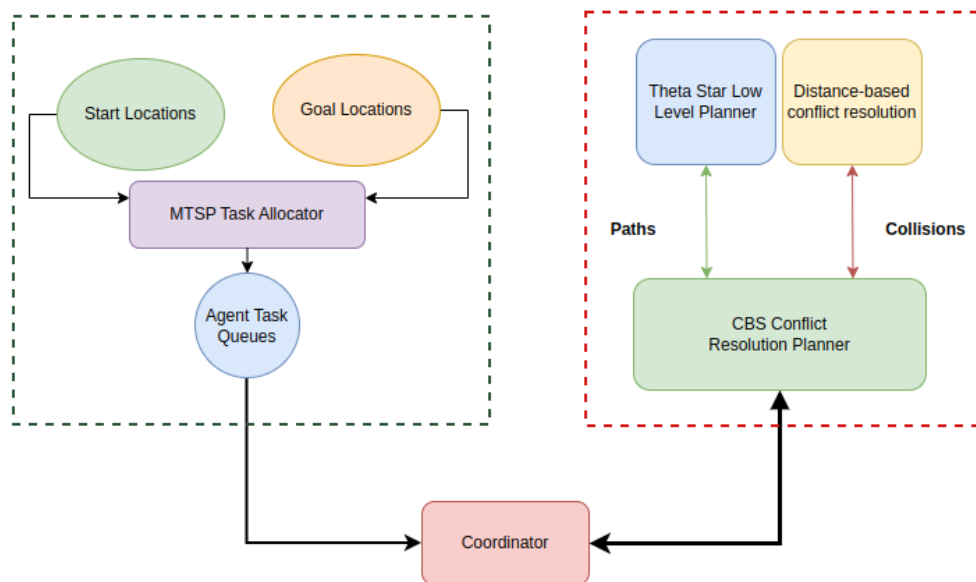


Figure 11. Planning and Task Allocation Overview

7.3.7 Autonomous Coupling

The autonomous coupling node operates in a step-by-step fashion, with 4 key steps:

1. Search phase - the coupling robot searches for the front robot through a swivel behavior. Though this is necessary when using Aruco tags, in the final implementation with Vicon this was dropped
2. Navigating to approach point - An approach point is determined by intersecting the current line of motion and the line of motion of the front agent. This ensures all large angular corrections are done in the start since the Khepera controllers are not reliable in controlling yaw
3. Feedback-controlled approach - Once in the approach point with the poses aligned, the back agent slowly approaches the pose of the front agent through fine angular and lateral adjustments.
4. Fine-grained approach for docking - Once very close to the front robot, the back agent yaw control is dropped and a pure forward velocity is given till the limit switch on the front agent is hit

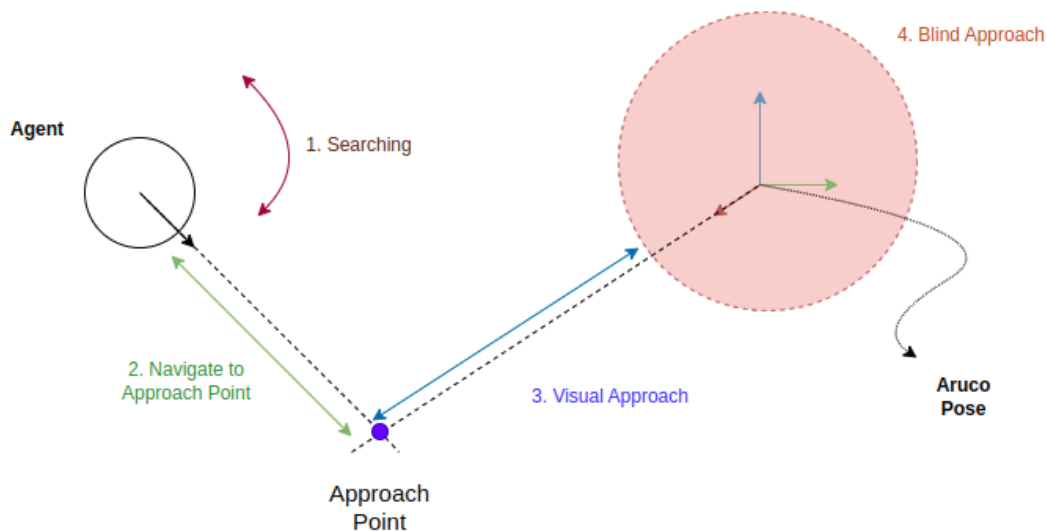


Figure 12. Autonomous Coupling Overview

Due to the narrow tolerances of our system, often two agents would be misaligned though the approach is complete. In such cases, there is a final check of lateral error before the electromechanical system is engaged. If the error is above 5 mm, the back agent reverses and retries the entire process. This addition dramatically reduces our failure rates and ensures that when the system does couple, it does so without causing damage to our mechanical components

7.3.8 Fleet management system

The fleet management system serves as the communication bridge between all the agents and the server. The user provides a config file with information about all the agents in the fleet to set up communication channels for transmission and reception for all the agents. It also handles packing, unpacking, and logging of data and monitors the agents during operation. It relays information back to all the other subsystems in case of any changes in the fleet. The commands to the individual agents from each subsystem also go through the FMS. Since everything is integrated through ROS, the FMS becomes the direct interface with the robots. The mechanism has also been integrated with the FMS. This enables communication between the host computer and the NodeMCU microcontrollers in the robots' mechanisms.

7.4 Modeling, Analysis and Testing

Table 5. Project unit test plans

Test No.	Objective	Procedure	Requirement
1	Build a map of the test arena and localize agents within the map	Run SLAM Toolbox which takes in the robot's wheel odometry and LRF scan messages to build a map, localize agents within the map using VICON.	MF1
2	Plan collision-free trajectories for agents as per their assigned goal locations	Given the map of the environment and localization using the VICON tracker, run the planner to assign collision-free paths for each robot in the swarm.	MF2, MF6
3	Allocate robots respective PoIs optimally and allocate robots for the gap-crossing task whenever required.	Given the map and localization information, run the task allocator to allocate goal points to respective robots.	MF2, MF6
4	Couple three robots and make them cross gaps	Place three robots with the mechanism close to each other. Manually align them to allow for smooth coupling. Cue the coupling routine and gap-crossing routine.	MF5
5	Ensure lossless communication	Turn on the robots and connect	-

	between server and agents	them to the server through the FMS, observe packet information and rostopic rates.	
6	Robustness of Coupling Mechanism	Manually couple 2 and 3 agents, attempt crossing at various lengths with varying run-up distances	MF5
7	Task Allocation with Gap Consideration	Create multiple maps with varying gap widths and manually inspect solution for feasibility	MF3, MF4, MP9
8	Autonomous Alignment of Agents	Attempt just the autonomous alignment and coupling from various different start positions for 2 and 3 agents and record success rate	MP8
9	Electromechanical Coupling	Pre-align agents and issue terminal commands to initiate entire coupling sequence, test by tele-operating robots over gap	All
10	Integrated Tests	Testing all subsystems integrated and documenting edge cases	All

7.5 SVD performance evaluation

The system performed very well and met all the requirements set for the spring semester at both SVD and SVD encore. The following table summarizes our test results for different requirements

Table 6. SVD Performance Summary

Success Criteria	Requirements	Unit tests	SVD	Encore
Crossing gaps of length 19cm	MP1, MP3, MP6	10/15	2/2	1/1
80% reliability for coupling mechanism	MP3, MP4	14/15	2/2	1/1
100% POIs reached	MP2, MP5	24/30	3/3	3/4

100% collisions avoided	MP2	29/30	2/3	3/4
POI exploration less than 5 minutes per run	MP2	22/30	3/3	3/4

7.6 FVD performance evaluation

Table 7. FVD Performance Summary

Success Criteria	Requirements	Unit tests	SVD	Encore
Crossing gaps of length 19cm	MP1, MP3, MP6	8/10	2/2	0/2
80% reliability for coupling mechanism	MP3, MP4	14/15	2/2	0/2
100% POIs reached	MP2, MP5	30/30	2/2	2/2
100% collisions avoided	MP2	27/30	2/2	2/2
80% success rate for autonomous coupling	MP8	17/20	2/2	0/2
Total system run time less than 20 mins	MP2	19/30	2/2	0/2

7.6 Strong and Weak Points

Strengths

- POI coverage is sufficiently reliable
- The system successfully crosses the gaps of desired length
- The system can now handle variable gap lengths
- Planning is sped up considerably due to multithreaded exploration of search trees
- There are no unplanned collisions between agents and/or obstacles
- The electro-mechanical system is much more stable due to the inclusion of an RPi

Weaknesses

- Mechanism is prone to wear and tear, increasing chances of failure
- Localization still relies on the Vicon system
- Task allocation is compute and time intensive due to exhaustive search across all potential crossing configurations

- The executing controller, though accurate, is slow and visually unappealing. This is in part due to not having time to tune the PID, and in part due to the unreliability of the Khepera IV to execute motor commands
- Communications are quite intensive and the system can run in to slow down due to network congestion under certain conditions

8. Project Management

8.1 Work Breakdown Structure

We chose to split the project into 5 key verticals - Hardware, Fleet Management System, Planning, Localization and Management. This is derived from the major blocks in our cyber physical architectures explained above (section ref.).

For Hardware, We have completed the design and development of the coupling mechanism and enclosure with few iterations. The coupling mechanism has been tested thoroughly but has scope of improvement in terms of robustness. The electromechanical system has been integrated and tested with the mechanism. Upcoming semester plans include design improvement and integration with the entire system

For FMS, all the mentioned tasks have been completed, but will need revisiting in the upcoming semester once we add feedback and perception to the communication pipeline

For Planning, task allocator and goal trajectory planner have been developed and tested. configuration determination, coupling/decoupling planner and gap crossing planner are the tasks for the upcoming semester.

For Localization, all the tasks are completed. We will be revisiting all the tasks again, since we will be using different methods of localization for upcoming semester

For all the verticals some tasks will be revisited, in order to address the issues incurred due to entire system integration.

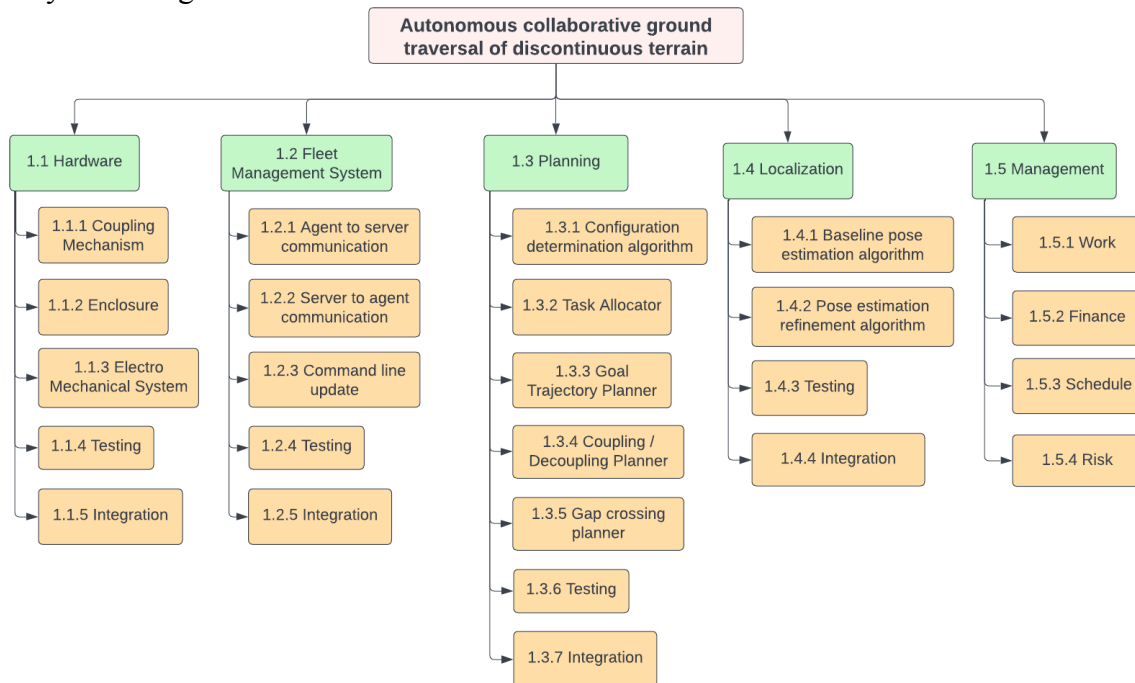


Fig 13: WBS Schematic

8.2 Project Management: Schedule Status

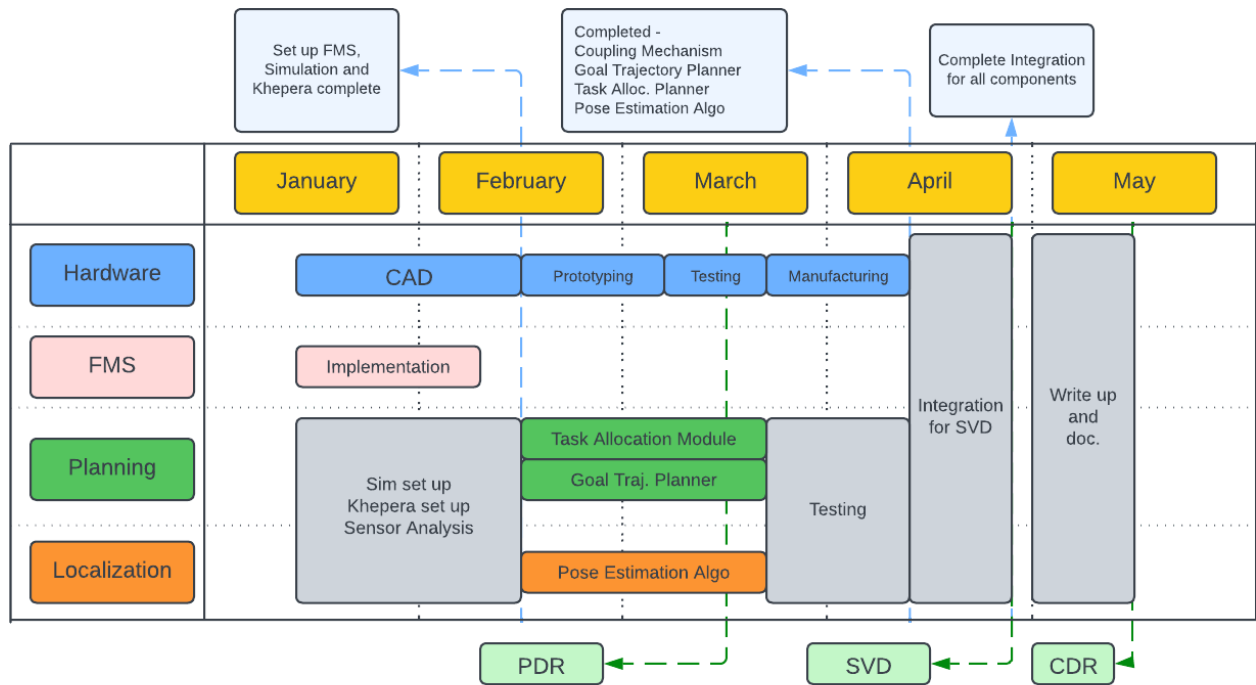


Fig 14: Spring Schedule

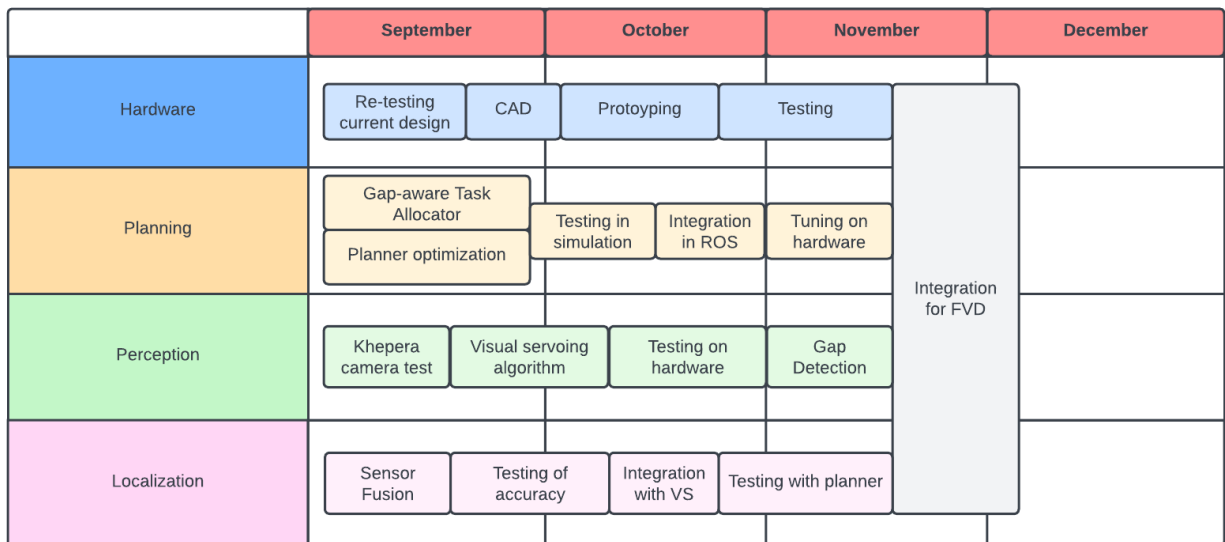


Fig 15: Fall Schedule

8.3 Project Management: Fall Test Plans

Table 8. Fall Semester Test Plans

Date	PR	Capability Milestones	Subsystem
13th Sept	6	<ul style="list-style-type: none"> ● Robust Coupling Mechanism with improved reliability ● MPC controller testing ● Khepera Camera Testing with Aruco Tags ● LRF + encoders vs Vicon 	All Subsystems
27th Sept	7	<ul style="list-style-type: none"> ● First prototype of redesigned mechanism ● Visual Servoing first iteration ● Gap-aware task allocation in simulation ● Planner integrated with new localization 	All Subsystems
11th Oct	8	<ul style="list-style-type: none"> ● Visual servoing final testing ● Integration of task allocator, planner and controller in simulation ● Manufacturing of mechanisms 	All Subsystems
1st Nov	9	<ul style="list-style-type: none"> ● Integrating perception pipeline with the navigation stack on physical robots ● Finalize demonstration concepts 	All Subsystems
15th Nov	10	<ul style="list-style-type: none"> ● Full system testing on various configurations ● Edge case documentation ● Variable gap-width test 	All Subsystems

8.4 Project management: Parts list and Budget

Table 9: BOM Budget Requirement

S No.	Part	Item Name	Link
1	Linear Stage Actuator	Linear Rail 50mm / 100mm / 150mm/ 200mm Linear Stage Actuator with Square Linear Rails Mini Slide Table + NEMA 11 Stepper Motor for DIY CNC Router Milling Machine (50mm)	https://a.co/d/b0CcMrV
2	Solenoid Lock	uxcell DC 5V 1A 30g 3mm Mini Electromagnetic Solenoid Lock Pull Type for Electric Lock Cabinet Door Lock	https://a.co/d/hdm5zyP

3	Rail	drylin® T, miniature profile rail	https://www.igus.com/product/730?artNr=TS-04-09
4	Carriage	drylin® T Miniature Carriage	https://www.igus.com/product/930?artNr=TW-04-09-LLY
5	Rollers	Donepart Small Bearings 4mm x10mm x4mm MR104 2RS Miniature Ball Bearings Double Metal Shielded for Mini Motors, Fidget Spinners, Industrial Machinery, etc (10 Pack)	https://a.co/d/egq43P0
6	Roller head	Breezliy 590Pcs 1-8mm Metric Precision 304 Stainless Steel Assorted Loose Bicycle Bearing Steel Ball Assortment Kit (12 Sizes)	https://a.co/d/aMCcrPI
7	ESP 8266, Node MCU	HiLetgo 3pcs ESP8266 NodeMCU CP2102 ESP-12E Development Board Open Source Serial Module Works Great for Arduino IDE/Micropython (Large)	https://a.co/d/jgQUhG0
8	DC-DC Step down buck converter	Valefod 3 Pack LM2596 DC to DC Voltage Regulator 4-40V to 1.5-35V Buck Converter with LED Display	https://a.co/d/1HutURo
9	DC-DC Step-up buck converter	DFRobot DFR0123	https://www.digikey.com/short/fnrvjf7
10	3V Single channel relay	3v Relay Board Raspberry Pi Arduino Relay Module 1 Channel Opto-Isolate High Level Trigger for IOT ESP8266 Microcontrollers Development Board	https://a.co/d/e5UaWQy
11	11.1 V Battery	Zeee 3S Lipo Battery 5200mAh 50C 11.1V RC Batteries with XT60 Connector Soft Case for RC Airplane Helicopter Plane Quadcopter RC Car Truck Boat	https://a.co/d/djVCLGE
12	Test surface	LAGKAPTENTabletop, white,	https://www.ikea.com/us/e

		63×31 1/2 “	n/p/lagkaptent-tabletop-white-10508220/
13	Raspberry Pi Zero	dryLin® R pillow block RJUM-05	https://www.igus.com/product?artNr=RJUM-05-12
14	Pillow Block Bearing	dryLin® R pillow block RJUM-05	https://www.igus.com/product?artNr=RJUM-05-12
15	Bearing Flange	dryLin® R pillow block FJUMT-01	https://www.igus.com/product/1211?artNr=FJUMT-01-10
16	Shaft	drylin® R hard-anodized, aluminum shaft, AWM	https://www.igus.com/product?artNr=AWM-12

8.5 Risk Management

Table 10: Key Risks and Mitigation Plans

Sr No	Risk Description	Likelihood	Consequence	Mitigation
1.	ESP + NodeMCU is unreliable in communications and controlling linear actuators	4	5	<ul style="list-style-type: none"> We replaced the stack with an RPi and ESP connected via serial. The RPi provided superior WiFi reliability
2.	Enclosure and pins break easily during testing	4	5	<ul style="list-style-type: none"> Redesigned the mechanism using standard Iigus aluminum linear bearings which made the entire mechanism more robust Negotiated with our sponsors to use their 3D printing facilities when required for fast prototyping of new designs
3.	Autonomous Coupling failing due to incorrectly calibrated agents	5	5	<ul style="list-style-type: none"> We followed an extensive SoP, by recreating an agent’s Vicon tracking model every time it underwent a change
	Components not arriving or arriving late	4	4	<ul style="list-style-type: none"> Frequent communications with stakeholders in case a critical part does not arrive

				<ul style="list-style-type: none"> • Backup orders for parts that have analogues • Space out ordering (by a week) to make sure that we catch stock refills on scarce components like RPis and Igus bearings
4.	Kheperas get stuck on small bumps and edges during the demo	3	3	<ul style="list-style-type: none"> • We procured acrylic sheets which smoothed out the surface
5.	Unavailability of testing area	2	4	<ul style="list-style-type: none"> • Let our stakeholders know early on that we plan on using the lab space to avoid any issues
6.	WiFi failing	2	4	<ul style="list-style-type: none"> • Ask audience to put their phones on airplane mode • Restart the router and the agents to flush any sockets • Restart the central server and the docker container
7.	Components overheating/burning when under extended use	2	5	<ul style="list-style-type: none"> • Have strict timings for testing to prevent overheating • Attach heatsinks to the RPis and the ESP • Take small breaks between runs to give the circuits • Perform a sniff test after every run to ensure nothing is burning out
7.	Team Member(s) fall sick	1	2	<ul style="list-style-type: none"> • Continuous documentation using the website • Every task had at least two people involved allowing for one to take over if the other was unavailable

9. Conclusions

9.1 Lessons Learnt

Through the course of the semester working on our system, we had quite a few challenges that we needed to overcome and some valuable learnings, both in the technical aspects as well as in managing project progress, stakeholders and conducting a smooth demonstration.

1. Test early, fail early - our integration went quite smoothly because we had tested each subsystem early in development and documented the known issues well.
2. Clear communication with stakeholders - there were multiple times where we were facing issues in hitting our performance requirements due to unforeseen technical issues in the platforms. Keeping a clear line of communication with our sponsors made sure that all decision were transparent
3. Too much testing can backfire - since we had a significant electromechanical component in our project, the wear and tear due to repeated testing meant that our system lost reliability closer to the demo, necessitating last minute problems in integration
4. Controlled conditions are best - for our encore, we encountered issues with the WiFi setup we had never seen before. We learnt later that it was due to our router being used by another PC, adding a strain to the network that wasn't there during testing.

10. References

1. Sha Yi, Zeynep Temel, and Katia Sycara. "PuzzleBots: Physical Coupling of Robot Swarm." IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021.
2. Sha Yi, Zeynep Temel, and Katia Sycara. "Configuration Control for Physical Coupling of Heterogeneous Robot Swarms." IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2022.
3. Saab, W., Racioppo, P., & Ben-Tzvi, P. (2019). A review of coupling mechanism designs for modular reconfigurable robots. doi:10.1017/S0263574718001066
4. Khepera IV manual and datasheet <https://www.k-team.com/khepera-iv>
5. Project Website: <https://mrsdprojects.ri.cmu.edu/2023teama/>