Coordinated Task Execution of Robots Using Relative Sensors

BILKENT UNIVERSITY
Electrical and Electronics Engineering Department

EEE 493/4 Industrial Design Project
Final Report
June 4, 2021

B4 Group Members

Adil Berkay Temiz (21702097), Atahan Yorgancı (21702649)
Berk Yaşar Yavuz (21600824), Tuna Alikaşifoğlu (21702125)
Yiğit Berk Üçüncü (21601607), Yüksel Arslantaş (21602915)

Academic

Dr. Mehmet Alper Kutay (Coordinator)
Asst. Prof. Aykut Koç (Mentor)
Mert Acar (TA)

Company Mentors

Dr. Bilge Kaan Görür
Serdar Kırımlıoğlu
Abdullah Alp Muhiddinoğlu

roketsan

Company Description: Roketsan, was founded on 14 June 1988 by the decision of the Defense Industries Executive Committee for the purpose of establishing a leading institution in the country for designing, developing and manufacturing rockets and missiles [1].

This project was supported by TUBITAK under the 2209-B program.

Contents

Abstract

In this project autonomous mobile robots with heterogeneous hardware configurations are developed for collective task execution without communication between the robots, or global mapping systems. Collective tasks such as converging on a predefined target will be executed with three robots with different hardware configurations. Without communication (i.e. RF, LoRa) between robots, and global localization systems (i.e. GPS), robots make use of both visual data from cameras, and distance data from LiDAR to local mapping of their environment. Visual, and distance data is used to further augmented with object identification, and localization so that robots can identify obstacles, peer robots, and objectives in the environment. Local mapping of the environment is then used by a custom obstacle avoidance algorithm to perform collective task execution. The robots' collective ability to execute common objectives will be evaluated using five different scenarios. These scenarios are "Target Identification and Transition with One Robot", "Pioneer Robot Following", "Target Identification and Locating with Three Robots", "Target Identification and Transition with Three Robots While Avoiding Obstacles" and "Hostile Target Detection and Avoiding". These scenarios test robots' ability to identify a predefined target, and converge on the said target using local mapping of the environment. More challenging problems such as path finding in an environment with obstacles, are solved using custom grid-based obstacle avoidance algorithm. Developed systems such as object identification, and path finding are required to run in parallel. For process orchestration, and lightweight message passing Redis will be used, for low memory footprint, and low communication overhead. Source code of the project is present at https://github.com/RoboCoRS/unified.

1 Motivation & Novelty

1.1 Motivation

In this project autonomous mobile robots with heterogeneous hardware configurations are developed for collective task execution without communication between the robots, or global mapping systems. These collective tasks include but are not limited to performing reconnaissance, or converging on a specified target etc. by identifying peer robots' location, and state in uncharted flat environments.

Specifically, the robot swarm should not use direct communication technologies such as RF signalling, LoRa for sharing gathered information, so that each robot can perform the objective alone if others fail. Further, robots should not have master-slave relationships while collectively executing the given task. Motivation behind this constraint is that task execution does not have a single point of failure. For instance, in master-slave configuration when a communication between the master, and the slaves is severed the slaves can fail to perform the given task. Since in our project there is no communication between agents, the proposed solution becomes nearly invulnerable to jamming or spoofing attacks which can cause miscommunication. All in all, communication constraints are expected to increase robustness in foreign, and harsh environments.

Developed solution is meant to be used in military projects, therefore, exact details about how the solution will be used, or whether the solution will be integrated into an existing product is not disclosed. However, the probable use of the end product might be

performing reconnaissance in unknown harsh environments to not risk human lives, or the developed system can be integrated into existing transport vehicles to reduce the human labor required in transporting equipment. Our project is developed such that subsystems are modular, for instance object detection subsystems can be changed independently from other subsystems such as path finding. Therefore, it is probable that the company will integrate our design into their existing systems. The project can be further improved by upgrading hardware components, improving the chassis of the robots for more resilience, and wireless reporting mechanisms for remote monitoring, and debugging.

1.2 Novelty

Since our product is intended for military use, commercial products available for the public are not suitable for feature comparison to our project. Therefore, similar products are rare, and many details of available products are not made public. So, we choose to compare our product with the previous year's project with the same company, and project details. In terms of cost, both projects are almost identical since we are using the exact same equipment and provided with the same budget. Both projects are quite novel in terms of having swarm topology; collecting, processing camera data and making decisions based on these data; and executing a task collaboratively. Different from last year's project, our project exploits the advantages of communication-free design which provides more robust coordination in harsh environments such as the battlefield. However, it also promotes some disadvantages like having to carry the necessary equipment such as LiDAR and camera in each individual robot since there is no communication between robots which is an additional constraint, so each robot uses on-vehicle sensors to collect their own data. Lastly, the common and unresolved problem of both projects is the range issue. For our project, we are limited by the range of our cameras and LiDARs due to budget constraints.

According to the research done by searching the existing patents, online literature review, it is observed that there are plenty of patents on the algorithms for robot swarms for path finding, and task allocation between swarm members. However, the field we are working on is quite new, and there are various unsolved problems. So, it is possible to improve upon existing algorithms by making them more robots, or extending their use cases. In our project, we are making use of well-known algorithms such as Kalman filters for predictive control, particle filters for processing LiDAR data, and grid-based localization, and path-finding to solve our problems. Algorithms that we use such as particle filters are widely used, but we have introduced heuristics to augment the existing filter to fit our needs. These augmented algorithms might possess the value to be patented as they solve a specific problem.

2 Design and Performance Requirements

The requirements of the overall project are discussed in two subtitles: functional and non-functional. Functional requirements, in other words, functions and capabilities of the end products are enumerated in subsection 2.1, where the non-functional requirements like environmental constraints, safety issues, etc., are presented in subsection 2.2.

2.1 Functional Requirements

1. Robots should be able to switch between and perform any of the five scenarios initially defined.
2. Platform and sensor validation must be acquired at the beginning of each scenario, e.g., validate cameras are recording, LIDARs are measuring, motors are operational.
3. Each robot must be able to detect the pre-defined target and its peers within the range of 2 meters, without internal communication or GPS information.
4. Robots should possess autonomous collective transfer ability to tend towards detected targets as a group.
5. At the beginning of each scenario, that requires a collective mission, robots shall find each other and generate a predefined formation, e.g., triangular form, single line. They should keep this formation during collective transfer while keeping the tracking distance at an optimal level between 30 centimeters to 1 meter, considering the functional range of LIDARs and cameras.
6. At least one of the robots should have a manual override option to select a pioneer, and to make other robots followers of the pioneer, in order to enable both manual and autonomous control of robot swarm.
7. Robots should be able to avoid obstacles closer than 30 centimeters during peer detection and collective transfer.
8. Robots shall possess recording and streaming ability for the data obtained from LIDARs and cameras. Recording and streaming resolution for cameras should be at least in 144p resolution at 10 frames per second.
9. The GUI of the system should display the live camera feed obtained from the cameras on the robots, as well as the captured proximity information from the LiDARs.
10. Users shall be able to configure the startup scenario for robots to accord with.
11. Robots should feature at least two heterogeneous hardware structures, e.g., their microcontrollers, LIDARs, cameras.

2.2 Non-Functional Requirements

1. In March 2021, total cost of the is determined to be approximately 3100$
2. The size and weight of the robots are flexible and they are not constrained with any specifications. Although, from a power consumption perspective, it is beneficial to keep the weight as light as possible. In this context, larger robots and smaller robots must not be heavier than 3 kilograms and 2 kilograms respectively to assure proper movements supplied by the motors.

3. Size of the robots shall be limited to the size of Dagu Wild Thumper 6WD and constructed robot structures, specified in section 4, with additional LIDARs, cameras and state indicator system on top.

4. As the power requirement, an adequate power distribution system is necessary to supply required voltage and current needs of components properly, e.g., cameras, LIDARs, motor drivers, microcontrollers, etc. All of the robots must operate with their corresponding Leopard Power 5200mAh 25C 4S 14.8V LiPo batteries provided by the company.

5. Robots shall operate in flat terrain and properly illuminated working environment.

6. Robots shall operate in a working environment where the temperature may vary between 0 °C-55 °C, and where the humidity level does not rise above 90%.

7. As safety concerns, usage of LiPo batteries issues a fire hazard when it is handled incorrectly as any other battery. Correct usage of LiPo batteries must be sustained during the project process.

8. Since the Dagu Wild Thumper 6WD can reach high speeds, theoretically it can cause physical damage to surroundings. The maximum speed of the robots shall be limited.

9. Project design is not affected by cultural and social factors. However, the no "communication" requirement is present in the functional requirements due to desired operation in environments where RF waves are distorted. This factor can be considered as a global factor.

3 Big Picture

The Figure 1 represent the big picture of the project. From the figure it can be seen that we are using 3 robots and many peripherals that help us to achieve requirements of the projects. Such as LIDARs for obtaining proximity information, ESP32 to provide live feed from robots point of view, arduino nano to drive the motors, cameras for the peer and target detection.

- *Motors:* Each robot uses the exact motors to match the drive speed and move together. One of them uses six and two of them use four motors.

- *Motor drivers:* Each robot uses two high-power 40A motor drivers to transfer energy from the battery to the motors.

- *Wheels:* Robots use big terrain wheels to move freely in any environment.

- *Raspberry 4:* Two of the robots use Raspberry 4 as the main computer, project algorithms, environmental sensing, and drive logic run on these computers.

- *Jetson Nano:* One of the robots uses Jetson Nano as the main computer, project algorithms, environmental sensing, and drive logic run on this computer.

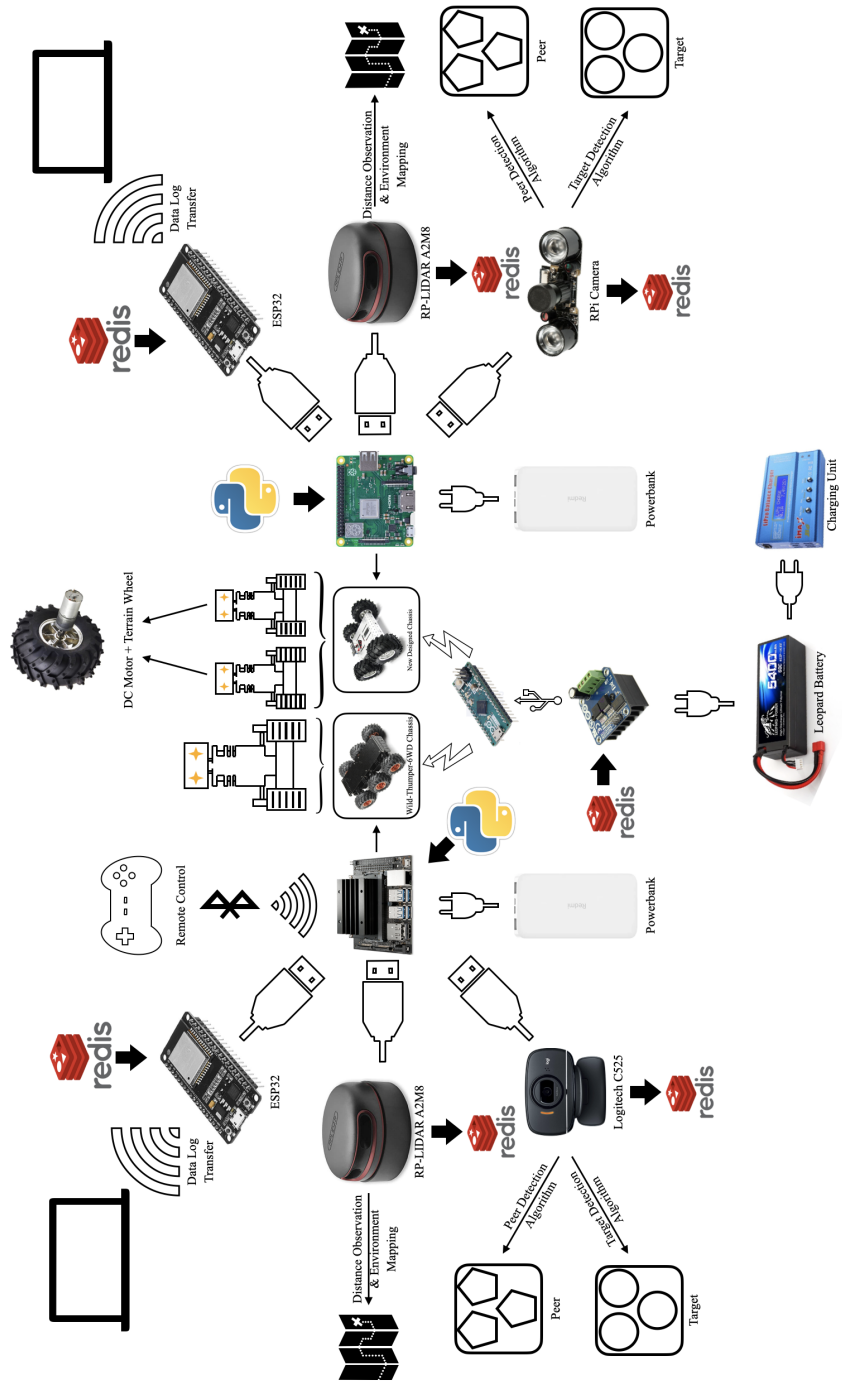


Figure 1: Big Picture

- *Arduino:* Every robot has an Arduino onboard to convert drive logic to PWM signals in order to make it suitable for motor drivers.
- *LiPo battery:* Each robot is driven by the energy supplied by the Li-Po batteries.
- *Power bank:* Each computer unit and every peripheral other than motors are powered by the energy supplied by power banks.
- *Logitech Camera:* One of our robots uses Logitech C525 camera as the image source for image processing.
- *RPi Camera:* Two of our robots use RPi Night Vision Camera as the image source for image processing.
- *Lidar:* All of our robots use a Lidar for environmental mapping and distance observation purposes.
- *ESP32:* All of our robots use an ESP32 to transfer the data generated by the onboard computers to a base computer.
- *Remote Controller:* A remote controller with 2.4GHz connectivity is used for the remote drive mode of our robots.

4 Methods & Implementation Details

4.1 Work Breakdown Structure and Project Plan

The project mainly consists of three subsystems which are object identification subsystem, environment mapping subsystem and mission subsystem. The object identification subsystem is responsible for target, obstacle and peer identification. This subsystem relies mainly on image processing algorithms, LIDARs and cameras. Second subsystem aims to map the environment in order to construct a representation of the environment for obstacle avoidance and path finding. Finally, the mission subsystem is responsible for controlling the robots via motor drivers in order to verge identified targets using two prior subsystems' output as input. The main goal with the project is to accomplish the provided scenarios (four of the scenarios are requested by the company, where one of them is designated by us as a project requirement). Thus, project progress is planned based on these scenarios. Since the scenarios are the progressive, they function as natural milestones for the project. Corresponding each scenario, five milestones are designated for five scenarios.

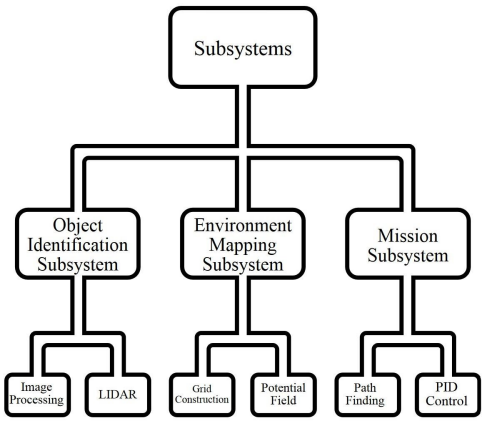


Figure 2: Subsystems

4.1.1 Scenarios

Target Identification and Transition with One Robot: In this scenario the main objective is to identify the predefined target, and arrive at the target using a single robot. In this scenario the most powerful robot (Wild Thumper with Jetson Nano) will be used. Procedure for the scenario starts with placing the robot arbitrarily into an open field without any obstacles. The robot should be able to perform reconnaissance, and identify the predefined target. After identifying the target, the robot should approach the target, upon arrival the robot should transmit collected operational data such as the robot's internal state during the mission, and collected reconnaissance data.

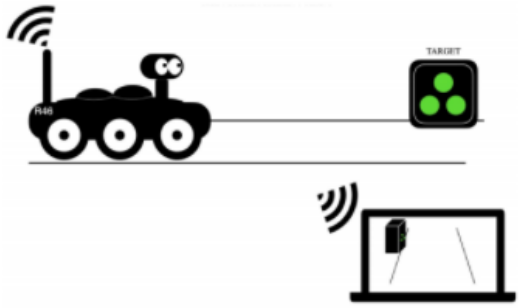


Figure 3: Representation of Scenario 1

Pioneer Robot Following: This scenario focuses on the collaborative, and swarm aspect of the project where a robot is remote controlled, and remaining robots should be able to follow the pioneer robot autonomously. Scenario starts with placing all three robots arbitrarily into an open field without obstacles. Then, a predetermined pioneer robot which is chosen to be the most powerful robot, is remote controlled to move around the field. Other two robots should be able to identify the pioneer robot from a distance, and follow the pioneer autonomously while keeping an appropriate distance. After the remote control session ends the robots should transmit the collected data.

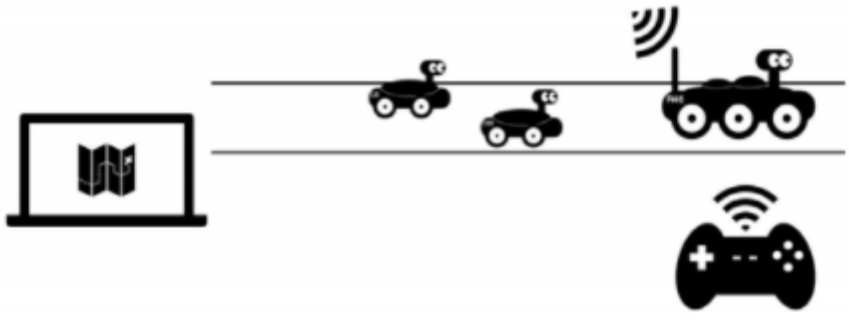


Figure 4: Representation of Scenario 2

Target Identification and Locating with Three Robots: In this scenario, the robots should be able to identify the target, and arrive at the said target together as a unit. Fundamental difference that sets this scenario apart from others is that each robot should

be able to perform target, and peer identification regardless of hardware differences, and be aware of other robot's state. Initially, all three robots are placed arbitrarily into an open field without obstacles. Then, robots should be able to identify each other, and meet together. After meeting robot's should try to identify the predefined target together, and converge on the target together after identifying it. Finally, robots should transmit collected data during the mission. The robots finding the target and forming a formation together to go to the target revealed many undefined problems that can occur during the execution of the scenarios. Some of the problems that could have been encountered before the scenario was modified are listed below:

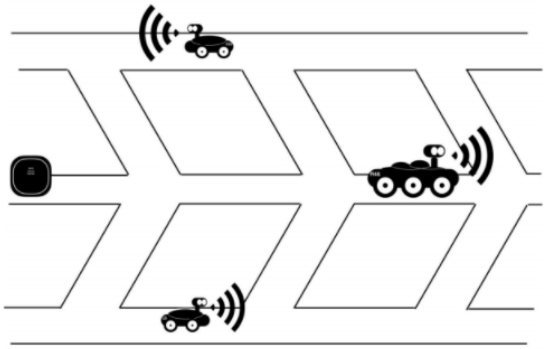


Figure 5: Representation of Scenario 3

- If all robots detect the target initially, Scenario 3 would be same as the implementation of Scenario 1 with 3 robots.
- If a robot sees the target, and waits for other robots to come near it, the pack will lose all its ability and remain in an infinite loop, when the two robots see the target.

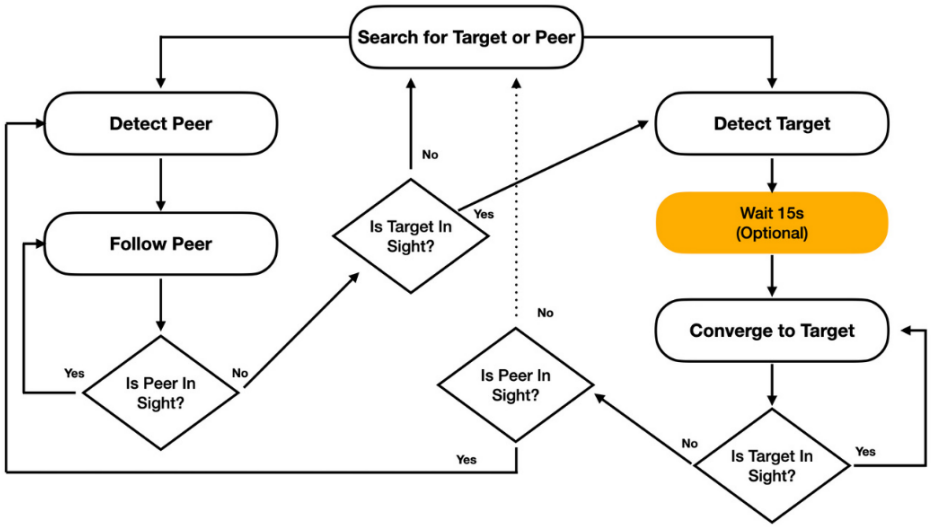


Figure 6: Algorithm for Scenario 3

Considering these problems, we discussed this issue with ROKETSAN. With the approval of the company's mentors, the scenario took its form:

- The robot that sees its peer will move towards its peer, the robot that sees the target will move towards target
- The robot that detects the target will converge to the target. In this case, if it is peer appears in sight and comes between the current robot and the target, current robot will follow its peer.

In our meeting with ROKETSAN, it was thought that situations where the robots are too far from each other might cause difficulties in the execution of the scenario 3, so the scenario should be started when the robots are in close proximity to each other. Moreover, updated scenario ensures that the loss of any robot in the swarm does not prevent task execution. The fact that the robots can recognize their peers also prevents the interference of spy robots in the swarm, as ROKETSAN requested us.

Target Identification and Transition with Three Robots While Avoiding Obstacles: Different from other scenarios, this scenario's environment includes obstacles in the environment. Even though the objective of the scenario is the same where robots should identify each other, and the target, then arrive at the target, obstacles introduce an additional layer of complexity. For instance due to obstacles vision of robots obscured, or identified objects can become invisible due to obstacles. To overcome this challenge predictive algorithms such as Kalman filters can be used. First all three robots will be placed arbitrarily into the field with obstacles. Then, robots should identify each other, and gather together. Then, robots should explore their environment, and perform reconnaissance to find the predefined target together. Lastly, robots should converge on the said target together, and transmit data collected during the mission upon arrival.

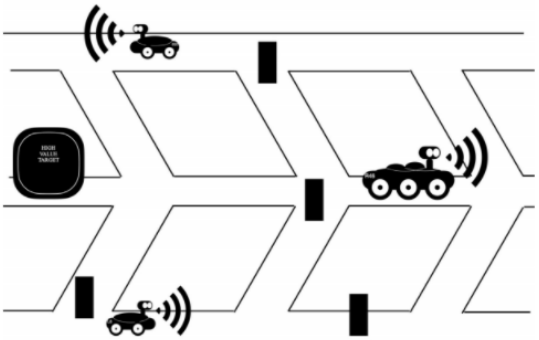


Figure 7: Representation of Scenario 4

Hostile Target Detection and Avoiding: Different from other scenarios, this scenario is designed by our team, as a request by the company. Since we have not finalized this scenario decision with our company mentors, this scenario maintains its tentative state. Since the other scenarios are based on the following theme, for the last scenario we wanted to change the scenario approach and we designed a hostile target. In this scenario, the robots should be able to identify the hostile target. The hostile target could be a moving object, a stationary object or another robot. After identifying the hostile target, the robots should increase their distance between hostile targets and themselves.

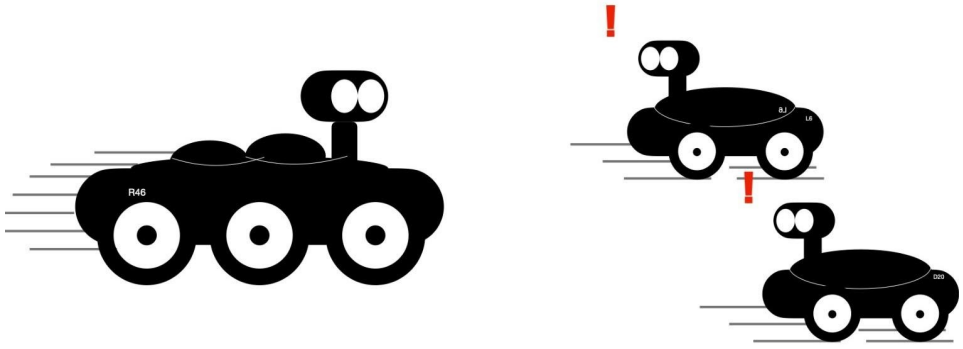
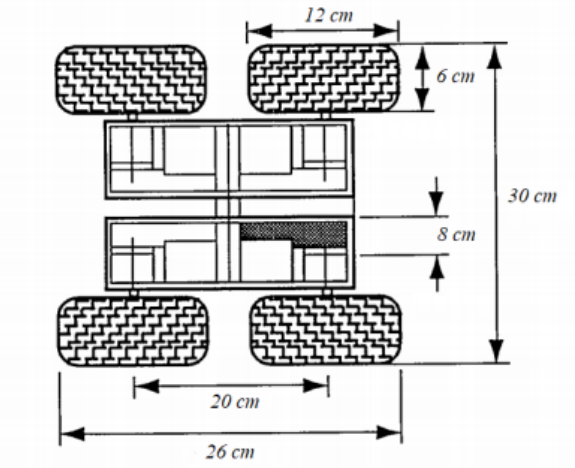


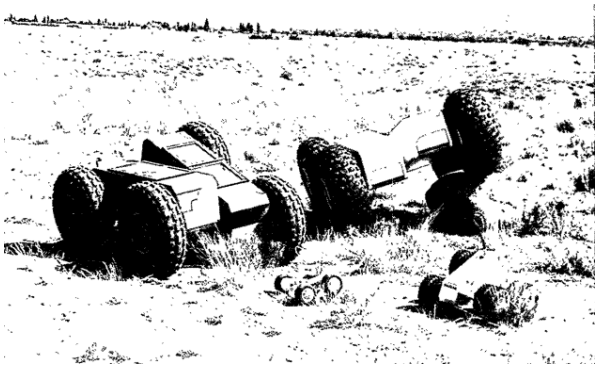
Figure 8: Representation of Scenario 5

4.1.2 Robots

ZK-4WD robots which are provided by company do not have motors that are powerful enough to carry the whole system, and make turns smoothly, since as mentioned in non-functional requirements an extra 1–1.5 kg weight is added with all equipment. That's why we built new chassis instead of ZK-4WD chassis provided to us*. In addition, smaller robots' motors and wheels will be replaced with the same model of motors and wheels in DAGU Wild Thumper in order to diminish the performance problem that might cause the loss of vision in some certain scenarios. In chassis design NASA's Robotic All-Terrain Lunar Exploration Rover (RATLER) design was taken as an example. The body is constructed of twin left and right compartments connected by a hallow central point [2]. This design perspective guarantees that the all four wheels of robot are always in contact with the ground. This feature increases the robustness of robots, and improves their performances on different terrains where scenarios are executed. The detailed schematic of body is given in Figure 9a and Figure 9b.



(a) Schematic



(b) by NASA [2]

Figure 9: RATLER Design

Furthermore, it has been observed that Raspberry Pi has a significant heating problem due to the high-power consumption during target and peer detection. Therefore, a fan

and a passive metal cooler were added to robot design. The primitive design that is used in progress and presented in P3 is shown in Figure 10. The final design of robot can be seen in Figure 11.

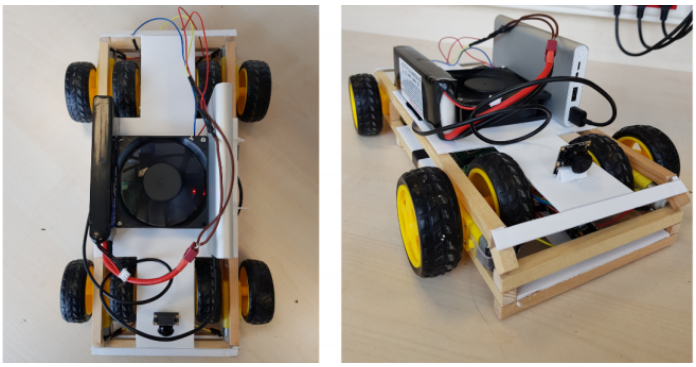
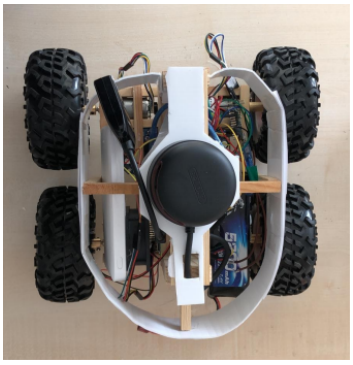


Figure 10: The Primitive Design of Small Robots

(a) Front View

(b) Side View



(c) Above View

Figure 11: Completed Robot for CM4

In the project, we use the DAGU Wild Thumper 6WD robot (see Figure 12) provided by the company besides the two robots that we constructed. This robot has been chosen as the pioneer robot whenever a scenario executed that a pioneer robot is required.

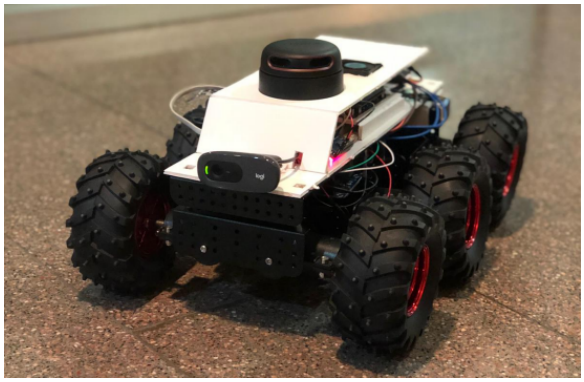


Figure 12: DAGU Wild Thumper 6WD robot

4.2 Work Breakdown Structure and Gantt Charts

As we mentioned before our project have five milestones which are corresponds to a specific scenario. Besides for the project we determine 3 main subsystems; Object Identification, Environment Mapping and Mission Subsystem. In order to complete scenario one we determine the main goal as working shape detection algorithm where it belong to the object identification subsystem of the project. For the second scenario our main aim is duplicating the scenario one, but this using Raspberry Pi instead of Jetson Nano again this is belong to the object identification subsystem of the project. For the third scenario our main aim is to have working target search algorithm which belongs to the environment mapping subsystem and mission subsystem where robots try to apply target searching algorithm while looking for target. After that for the fourth scenario we determine the goal as determining obstacles via obstacle detection algorithm where it belongs to environment mapping subsystem. Lastly for the fifth scenario we determine the main goal as identifying hostiles via hostile detection algorithm where it belongs to the object identification subsystem.

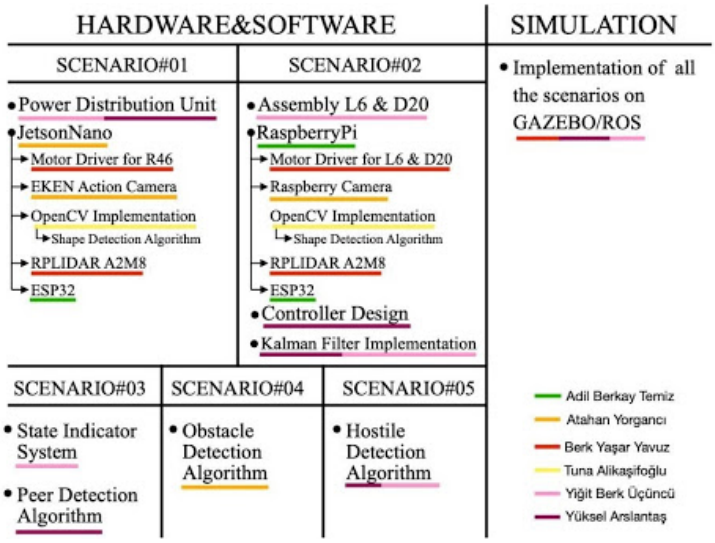


Figure 13: Work Breakdown Structure

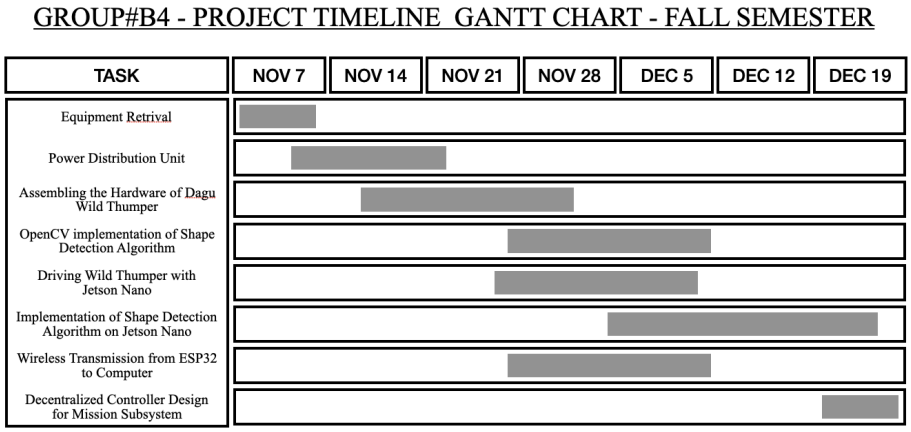


Figure 14: Gantt Chart-First Part of Fall Semester

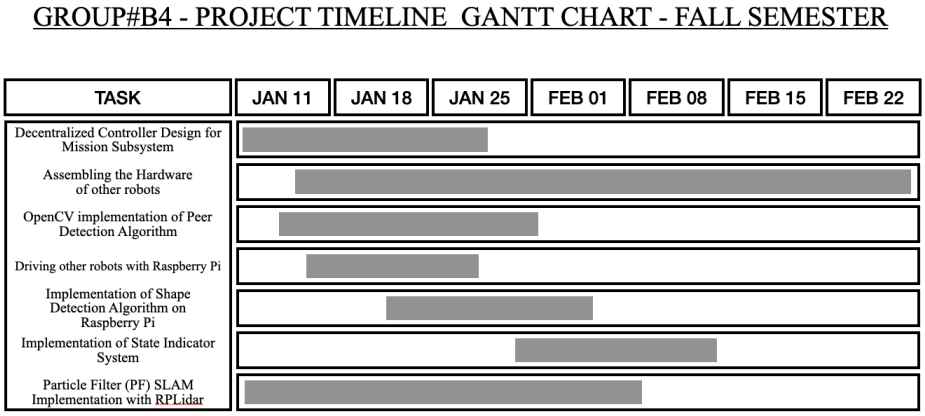


Figure 15: Gantt Chart-Second Part of Fall Semester

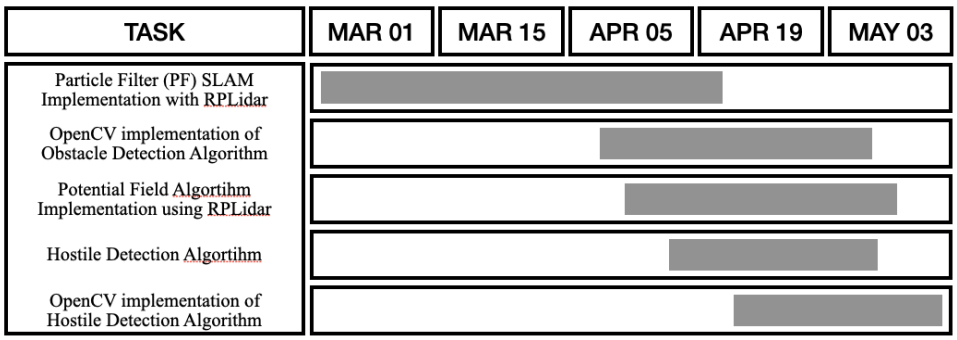


Figure 16: Gantt Chart-Spring Semester

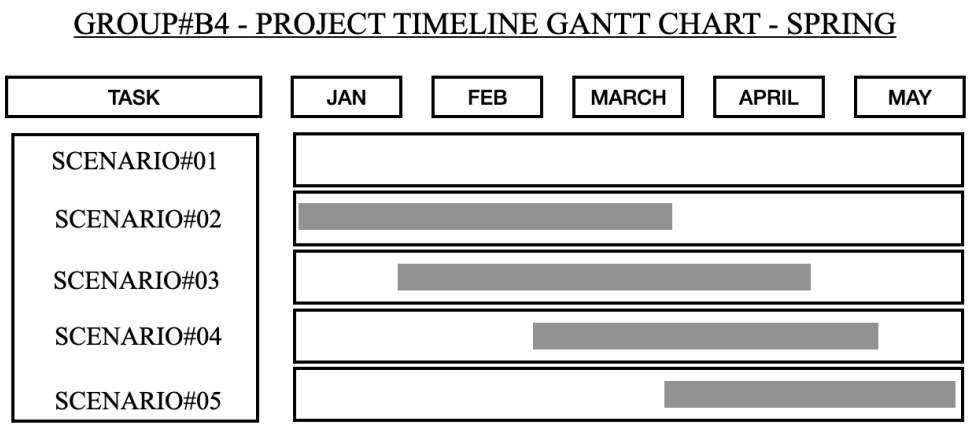


Figure 17: Gantt Chart-Scenario Based

4.3 Methods and Progress

4.3.1 Solution Strategy

The project that will be developed involves many sub-projects. The first approach is to identify these sub-projects in order to complete this project successfully. The main task of the project is to create robots which can autonomously follow each other and complete certain tasks. In order to complete this task robots' frames are needed and they should be able to be driven by controllers.

To be able to drive these robots it is needed to integrate the controllers with the robot systems. Since one of the requirements is to make the robots heterogeneous two different controller platforms are being used. These two controller platforms are Nvidia Jetson Nano and Raspberry Pi 4B. Thus, controller algorithms will be developed on these two platforms separately.

Following the verification of the integration between robots and their controllers, the next sub-project, environmental perception systems of the robots, will be developed. In order to make robots able to perceive their surroundings, a camera and a LIDAR to each robot will be implemented .

In order to make these perception data coming from LIDARs and cameras useful several different algorithms will be implemented. Using several different techniques on the data gathered from cameras and LIDARs, objects will be detected, peer robots in the system will be identified and a map of the environment will be created. OpenCV library will be used to accomplish the image processing requirements of this task. The potential field algorithm will also be implemented to use the data gathered from LIDARs to implement controller systems.

Since the requirement is to complete scenarios consecutively, the environmental mapping and object identification systems will be implemented on the pioneer robot for the first scenario. Then, these systems will be implemented on the remaining robots, a WiFi communication system on the pioneer robot will also be implemented in order to wireless control that robot for the second scenario. After the completion of the second scenario all the development will be solely based on the software of the robots except the state indicator system which will indicate the states (searching for a peer/going for the target

etc.) of the robots visually for the remaining scenarios. A visual indicator that defines the state of the robot controlled by the controller hardware will be implemented on the robots. The object detecting, environmental mapping and state indicating systems will be integrated together to implement remaining scenarios.

Every part of the development process will be implemented on the software simulation. After the verification process of the integrated systems and algorithms on different hardware which will be used in the project, developed systems into the hardware will be integrated.

4.3.2 Object Identificaiton

Target and peer identification is an important aspect of the project. Before making any moves, robots should be able decide where to go or who to follow. In this context, we possess cameras and LIDARs to function as a visual sense for robots to make them aware of their surroundings. However, obtained data from these sensors are not sufficient for robots to make decisions right away, we need to process the information that is collected by these sensors in order to make the robots "see" what they are looking at. One of the sub-components that will operate on the robots is the image processing component. It will function as the eyes of the robot and help it to detect its peers, peers' state and the target. Even though OpenCV is resourceful enough to train a machine learning model to detect the peers and the target, this will increase the complexity of the detection aspect which would undermine the remaining processes like driving the motors, operating the LIDAR, etc. Therefore, we came up with a solution that prunes the computational complexity of the detecting process which requires a set of predetermined shapes placed on both peer robots and on the target. This approach simplifies the peer and target detection problem into a shape detection and counting problem, which can be solved easily with OpenCV library.

During the first quarter of the project process, we tried to define our problem clearly, as well as its subproblems, in order to produce systematic solutions to the challenges we face. This process includes exploring the capabilities of OpenCV and obtaining an feasible approach for the target detection system, which is scalable to peer and hostile target detection. In this context, in the absence of a machine learning model, we defined our target as three close circles, which can be detected without machine learning, and started working on the implementation.

Detection: Using Hough Transformation is considered as the best practice for detecting circles. However, due to the fact that the incidence angle is not determined, usage of hough circles diminishes the accuracy of target detection, when the target is not right across the camera. Furthermore, hough circles are computationally intensive and too volatile for real-time target detection, with the limited resources, when it is compared to the proposed contour approximation method. The method approximates a contour shape to a polygon depending upon the specified precision. It is an implementation of the Douglas-Peucker algorithm. While renouncing the perfect circle detection of hough circles, robustness and fault-tolerance for incidence angle are obtained.

In order to use the polygon approximation, contours of the video frame must be obtained. For the contours, edge detection (Canny Edge Detection method is utilized) must be applied to the grayscale version of the input frame. In addition, to increase

the accuracy of the overall target detection algorithm, Gaussian blur is also applied to the gray scale image prior to edge detection to smooth the image in order to eliminate redundant edges. Then the contours are obtained emphasizing on external edges with simple chain approximation. With the detected circle contours in the preceding part, a bounding rectangle, that bounds all three circles, is defined. This bounding rectangle is necessary for initialization of well known tracking algorithms.

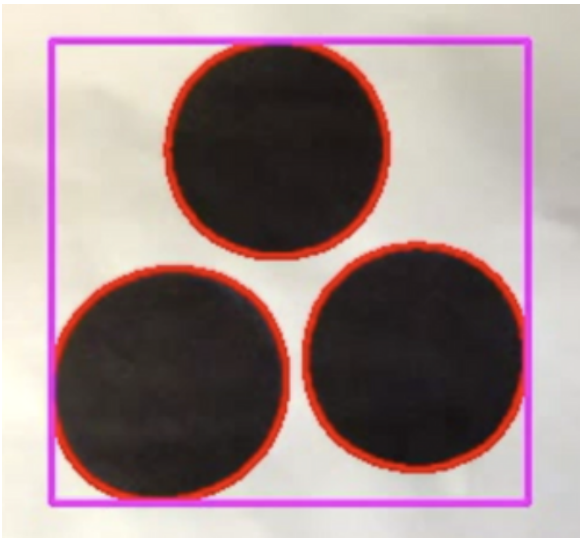


Figure 18: Detected Target

Tracking: Since usually tracking algorithms are much faster than detection algorithms, we implemented two modes to achieve target tracking: sentinel mode and tracker mode. In the sentinel mode, the system tries to detect three close circles by processing each frame captured from the camera. After obtaining the bounding rectangle as stated in the detection step, the system uses this rectangle to initiate the tracker. Currently, we implemented the OpenCV version of the Kernelized Correlation Filters (KFC) tracker. This tracker utilizes the fact that the multiple positive samples have large overlapping regions. This overlapping data leads to some nice mathematical properties that are exploited by this tracker to make tracking faster and more accurate at the same time. It also performs very well when it is time to notify the system, if the target is out of sight. Even though KFC tracker cannot recover from occlusions, it is the recommended tracker for most of the applications [3].

At this point, with an operational scenario 1 objective, our immediate goal is to implement scenario 2, which highly depends on a peer identification system for pioneer following. In this regard, our next goal is to scale the target detection system to peer detection system by defining another unique symbol for peer detection.

In addition to peer identification, improvements on the tracking system is in our current goals, we have been studying "GOTURN" tracking algorithm for some time, which is a lightweight pre-trained deep learning based approach, which can be an alternative to KFC especially in GPU accelerated environments such as Jetson Nano. Furthermore, implementation of TLD tracker, which can recover from full occlusions, is also scrutinized, however its false positives make it almost unusable. Therefore, we are planning to implement a combination of these trackers to obtain functionality from both aspects.

For peer detection we define three pentagons instead of circles. Detection and tracking implementations are similar to target case. As in the target tracking peer tracking can also be recovered from partial occlusions. When both target and peer appeared in sight system starts to track first one that can be detected. While tracking either the target or the peer if this system loses the track it immediately looks for the other one. If neither target nor peer in sight, it goes back to the sentinel mode. Occlusions between target and peers can occur during the scenarios. For example, while converging to the target another robot can come between the current robot and target. In this case even if the robot loses track of the target it can start track the robot in front.

To sum up, between two committee meetings, we have successfully implemented target detection and tracking algorithms. For the upcoming iterations we are planning to improve the performance of the current system and implement a peer identification module.

4.3.3 LiDAR

RPLIDAR A2M8 will be used to measure the distance accurately ($< 0.5\,\text{mm}$) in the horizontal plane around our robot up to 6 m of radius. Thanks to the LIDAR the surrounding objects identified by our image processing module will be detected and followed or avoided based on the case. To achieve this the LIDAR will be mounted at a location considering the height of the body of our robots and the approximate height of the obstacles which is expected to be encountered. The LIDAR is capable of taking 8000 samples and turning at 900 RPM rate, however, it is planned to be used at 4000 Hz sampling and 600 RPM rates since these are the default and recommended values by the producer. The essential part of the output of our LIDAR consists of a tuple of distance and angle. These data will be collected via UART as soon as it is sampled and be ready at the output of UART channel and will be processed in our microcontroller. Since most of the preprocessing is done on the LIDAR module itself both the processing power/time of the microcontroller and our time for implementing the preprocessing algorithms will be saved. Therefore, this tuple of data will be passed directly to the other modules of our systems and make use of it in our algorithms.

In CM3 demo, 30 cm is chosen as the safety distance, and the angle of 30° from the front seemed sufficient for the implementation of scenario 2. The data from LIDAR is processed accordingly.

4.3.4 Environment Mapping Subsystem

Potential field algorithms unlike many conventional motion planning algorithms do not require any knowledge on global coordinates instead the environment is modelled as a continuous field of values using a predefined potential function [4]. For instance using a proximity sensor such as sonars a robot can assign high potentials to points that are close to obstacles in the environment, then by minimizing this proximity field the robot can avoid obstacles.

Formally potential field algorithms can be described as follows. First, an artificial potential function that best suits the use case is chosen along with optimum range of potential for the robot. Then, potential function is used to generate the field representation of the environment. Then, optimization algorithms such as gradient descent are

used to find optimum locations in the environment. Alternatively, an iterative process can be used to search for optimum points. Lastly, the process is repeated until a point within the optimal range of potentials [4]. Potential field algorithms provide flexibility when dealing with a wide range of environments, and agents as the function can be customized to suitable for the use case. Moreover, this method can be applied for dynamic environments as well where the environment is subject to change, and potential fields calculated at different instances of time can differ.

4.3.5 Controller Design for Single Robot

In order to track the target there is a need for a controller. The input for the controller is the output of the image processing module that conveys the information where the target is located relative to our robot. Therefore, the error that we are making can be easily calculated and wheels can be driven accordingly. However, in the controller part there are a lot of different methods and algorithms that must be decided between, such as PID controller or its subsets such as P controller, PI controller etc. or bang-bang controller. Therefore, our team decided to start with the most basic one and add more complexity if it is really needed to do so. In the first trial, bang-bang controller is used, which is the most basic one, that does not take the magnitude of error into account and gives a binary output. The result of the first trial was promising so it is decided to optimize the bang- bang controller algorithm more and continue with it. Namely, a third state for the output

of the bang-bang controller is defined, which is the no-error state that is emitted when the target is in the region close to the middle of the frames taken from the camera. More specifically, if the center of the target is detected at most $+/-5\%$ away from the center in terms of the number of pixels, wheels are driven with the same speed so the robot moves straight to the target. And the other two outputs are "left" and "right" which indicates the position of the target relative to our robot. In these cases left and right motors are driven with a predefined speed so that the robot turns to right or to left with a constant rate until it catches the target in the middle. Furthermore, for the first scenario, the robot is required to stop when it reaches the target. There were a few options that were considered during the planning process. Using a lidar or an ultrasonic range sensor were discussed. However, we decided to use the data gathered from the tracking system. Using the size data of the target calculated by the tracking system was one option but being able to use different sizes of targets to control the robots is desired. Thus, the vertical position of the target is decided to be used for the reach algorithm. Due to the nature of the robot and its tracking system, the vertical position of the target goes upwards. So we calculated the necessary vertical position of the target that the robot understands it reached to target. When this condition is satisfied, a stop signal is sent to both left and right motors.

On the actuator side, Arduino Nano is used as the translator of controller output to the motor driver language which is pwm signal duration. The communication from Nano is established over UART with the Arduino and orders are sent as tuples of (actuator, value) in the format of a very lightweight protocol. Once the Arduino gets the tuple, it generates the PWM signal on the pin corresponding to the actuator with the width of the signal proportional to the value taken from the tuple. This generated signal is then

transferred to the motor controller hardware via specified pin. This motor controller adjusts the amplitude and the direction of the current which drives motors. Jetson Nano is not able to generate enough PWM signals to cover our project requirements without any complications due to its nature. Thus, this Nano-Arduino driver form is established by our team. In the following steps of this project, six different PWM signals will be required per robot for the peripherals. Lidars, state indicator systems and motor drivers should be driven by PWM signals. Even though Jetson Nano has enough pins for the current state of the robot, we implemented the Nano-Arduino system to create an adaptive framework for the later steps of the project.

In the controller design stage can be broken down to three main components which are PID controller for every agent, decentralized controller with interaction gain for multi-agent system and Kalman filter implementation to increase the robustness. The design of the controller be done with Simulink, and the design will be updated after the simulation process.

Since our multi-robot system does not have well defined leader robot that sends commands for other agents to execute, the multi-robot system can be considered a decentralized system consisting of three subsystems where each subsystem represents a single robot. Each subsystem will have a PID controller for robot's DC motors. PID controller can be used to regulate the current drawn from the battery for optimum performance. The proposed PID design is shown in Figure 19.

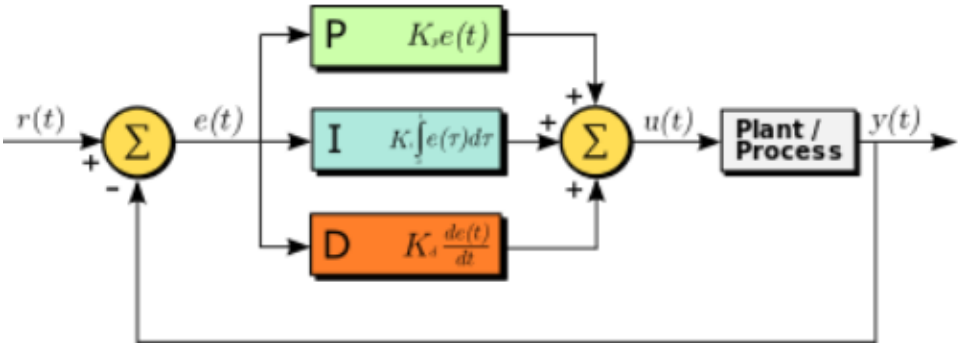


Figure 19: PID Controller [5]

Dagu Wild Thumper robot, which will be used in the first scenario, has 34:1 metal gearmotor with 280 rpm which has better performance than ZK-4WDs whose motors operating at 250 rpm. Since DC motors have different performance ratings, PID controllers has to be designed differently for Dagu Wild Thumper and ZK-4WDs. Simulink can be used to design controllers that share the structure, but vary in tuning parameters.

In a multi-robot system, main goal is making sure the overall system is connective stable which means even if some agents of swarm split up or cannot function, the overall system continues the task execution. This approach eliminates single point of failures that can stop the whole system from functioning. In each subsystem, feedback loops are added to individual controllers to achieve a connective stable system. If we denote the overall system as

$$S: \ \dot{x} = f(t, x, u) \tag{1}$$

$$y = h(t, x) \tag{2}$$

where $x(t)$ is the state variable (e.g. position, orientation, velocities of robots) of system S at time t, $u(t)$ are inputs (e.g. the commended velocities) and $y(t)$ is the output. This system can be partitioned into $N = 3$ interconnected subsystems. In that case Equation 1, and Equation 2 can be modified as follows

$$S: \ \dot{x}_l = f_i(t, x_i, u_i) + \bar{f}_l(t, x, u) \tag{3}$$

$$y_i = h_i(t, x_i) + \bar{h}_l(t, x) \tag{4}$$

where $\bar{f}_l$ and $\bar{h}_l$ represents the interaction of subsystems with the system.

In order to have a connective stable system, the input-output reachability of the overall system is essential. For instance, if the system is input reachable, all state variables can be reached from the input. Similarly, if the output can be reached from state variables, the system is output reachable. Input-output reachability guarantees that the system is controllable [6].

Since state variables and output depend on the interaction of subsystems, the interconnection matrix of S can be defined as follows

$$E_{S\times S} = \begin{bmatrix} \bar{A} & \bar{B} & 0 \\ 0 & 0 & K \\ \bar{C} & 0 & 0 \end{bmatrix} \tag{5}$$

For input-output reachability, reachability matrix can be found from interconnection matrix as seen in Equation 6 and matrix must have no zero row or column [6].

$$R = E \vee E^2 \vee \cdots \vee E^S = \begin{bmatrix} F & G & 0 \\ 0 & 0 & 0 \\ H & \theta & 0 \end{bmatrix} \tag{6}$$

Research by Feddema shows that the system shown in Figure 20, input-output reachable and controllable [6]. Therefore, this controller design will form the basis of our design. As can be seen in Figure 20, the output of one system is also the input of other systems.

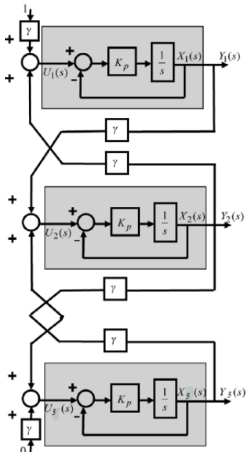


Figure 20: Controller design for multi-robot system [6]

In this decentralized case, interaction gain γ, is of great importance. This gain basically indicates the sensitivity of a subsystem to other subsystems. In order to have a connective stability, Lyapunov functions have to exist for all subsystems. In other words, Lyapunov functions must exist for all values of the interconnection matrix, E [6]. In order to check it, a test matrix W has to be designed. In the same research by Feddema following test matrix is built for a proposed controller.

$$W = \begin{bmatrix} K_p & -K_p\gamma & 0 & \dots & 0 \\ -K_p\gamma & K_p & -K_p\gamma & & \vdots \\ 0 & -K_p & K_p & & 0 \\ \vdots & & & \ddots & -K_p\gamma \\ 0 & \dots & 0 & -K_p\gamma & K_p \end{bmatrix} \tag{7}$$

Since for Lyapunov functions existence, test matrix W must be an M matrix whose all leading principle minors must be positive, and the limits of interaction gain can be found. As expected, interaction gain decreases as the number of swarm's agents increased since there are more neighbors, the become less sensitive for individuals.

In our case system has three subsystems, thus interaction gain can be 0.707 maximum. The interaction will be tuned in simulation part.

In order to increase the robustness of controllers and make reduce the noise factor Kalman filter will be added to robots. Kalman filter will act as a stochastic state observer to find the position of other robots. Kalman filter will combine the measurement from LIDAR and the prediction to find the optimal estimate of the robots' positions and any process noise and measurement noise due to LIDAR can be compensated [7]. In this design Kalman filter makes prediction and updates these predictions by changing Kalman gain. Prediction equations are in Equation 8 and Equation 9 where A is the state transition matrix which applies the effect of each system state variable at time $t-1$ on system state at t, B is control input matrix, $\hat{x}_t^-$ is the prediction of system's state, $\hat{x}_t$ is the updated prediction of system's state, u_t is input to the system at time t, P_t is the updated covariance matrix of state variable, P_t^- is the covariance matrix of state variable and Q is the process noise covariance matrix [8].

$$\hat{x}_t^- = A\hat{x}_{t-1} + Bu_t \tag{8}$$

$$P_t^- = AP_{t-1}A^T + Q \tag{9}$$

After this step, predictions will be updated according to measurements as demonstrated in (9), (10), and (11) where K_T is the Kalman gain, C is the transformation matrix used to map state variables into measurement domain, y_t is measurement vector and R is an uncertainty matrix representing the noise in measurement [8]. This process continues until the Kalman gain is tuned.

$$K_t = \frac{P_t^- C^T}{CP_t^- C^T + R} \tag{10}$$

$$\hat{x}_t = \hat{x}_t^- + K_t(y_t - C\hat{x}_t^-) \tag{11}$$

$$P_t = (I - K_t C)P_t^- \tag{12}$$

The implementation of Kalman filter will complete the controller design.

In scenario 3, robots need to meet and generate a formation after target recognition and converge to the target while keeping this formation. In addition, in scenario 4 obstacles will be added to the environment which might cause underperformance in the peer detection algorithm. We developed a suitable algorithm for scenario 4 to avoid different kinds of obstacles including very small pieces like legs of chairs as well as long obstacles such as walls. Having no knowledge about obstacle shape and the environment combined with crucial constraints such as scanning the environment exhaustively to find peers without leaving any unchecked area or retaining the formation without losing the connection between robots resulted in a quite challenging task. In order to achieve this goal, we come up with a searching algorithm which draws square spirals on the cartesian plane as shown in Figure 21 and robot stops for a second before and after each turn so that we give some time to the camera for identifying peers and the target.

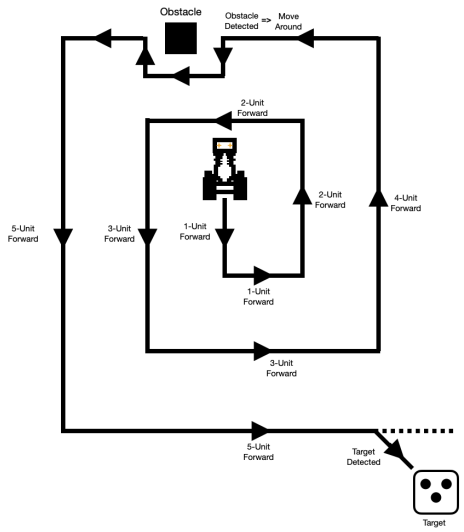


Figure 21: Search & Obstacle Avoidance algorithm illustration

For scenario 3 this algorithm works perfectly and finds peers and the target quickly, however, when there are obstacles in the environment as requested in scenario 4, an efficient handling was inevitable. We came up with the idea of avoiding obstacles by moving around it as shown in Figure 21 in order not to disturb the search algorithm and leave any unchecked area in the environment. The robot saves the current state parameters for the search algorithm and changes its state to obstacle avoidance when LiDAR detects an obstacle in the forward direction. After the obstacle is avoided and the state is changed back to search, the algorithm continues its execution from where it was interrupted so we are assured not to miss any area. While searching, the robot looks for peers and the target and increases the distance of the search path until it detects either the target or a peer. When it finds one of them it changes its state to Rush so it follows the target or the detected peer until the end of the scenario.

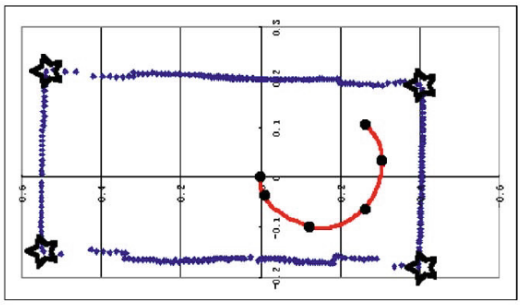


Figure 22: Point Cloud Generated by RPLIDAR [9]

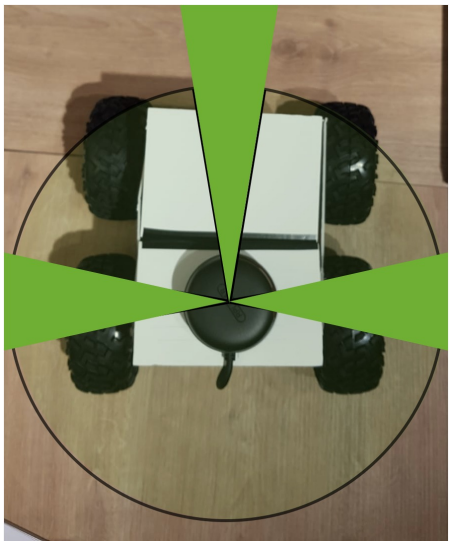


Figure 23: LiDAR Vision on Robot

4.3.6 Tools

Raspberry Pi 4 is a small computer with a powerful Quad-Core CPU and 6GB DRR4 RAM. Environmental mapping, image processing and all other control processes will run on this platform for the robots except the one running with the Jetson Nano. Raspberry Pi 4 is favorable for our application case due to it's enough computational power and GPIO pins with camera support. It also supports high speed internet connection over WiFi. Raspberry Pi has a very large community support which helps the development process, it also has community support for ROS integration.

NVIDIA Jetson Nano Developer Kit is a small, energy efficient, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. It is also a very suitable kit for on-device image processing applications. With these features, Jetson Nano becomes a very powerful tool for our application case. Thus, this computer will be used on our pioneer robot for image processing, environmental mapping and for all other control processes. However, due to lack of onboard wireless communication module on this computer, separate WiFi module will be used to control the pioneer robot in the second scenario.

Espressif ESP32 is a low-cost, low-power yet powerful SoC which is also very ver-

satile in the sense that it carries a WiFi module and supports 802.11n, Bluetooth and BLE. In addition, it has two processors and the main one has a 240MHz clock which enables us to deal with live-stream tasks. It can be configured as both Station and Access Point at the same time so that we can connect the ESP32 to the internet to access it from anywhere and log data to servers, while keeping Access Point feature so that we can easily connect to ESP32 and access the required data without needing any internet connection. We are live-streaming the camera data taken by the robots front cameras to the multiple devices via the web server we established on ESP32. Having a on-device web server enables us to access the web page without internet, it is also extremely secure since there is no connection to the rest of the network -even to local area network, and it is also fast and reliable in the sense that there is no middle-unit such as routers and there is no other data packets in the network since it is a private network established by ESP32. This also helps us in the server-side, which is ESP32, since there is no possibility to be encountered with excess number of requests thanks to keeping the network private. By using this private network, we live-stream the camera data and plan to live-stream robots status, map of the environment estimated by LIDAR data and some other necessary metadata. This is the one of the other two major change we include our project after CM2.

OpenCV is the most popular open-source computer vision library that supports C++ and Python programming languages and even has a library in ROS. It mainly aimed at real-time computer vision, and since the underlying codebase is written in C/C++, it is highly efficient and useful for generic image processing needs [3]. OpenCV's application areas include mobile robotics, motion understanding, object detection, object tracking and many more.

Redis and Multiprocessing CPython implementation of Python includes global interpreter lock (GIL) which is used to prevent race conditions in multithreaded Python code. GIL prevents multiple threads executing Python bytecode at the same time, since only one thread can hold GIL at a time there are no race conditions. However, this implementation slows down execution of parallel tasks, and we cannot make use of available computing resources using multithreading. Since our project is mainly written in Python, and we are using CPython implementation, we had to come up with an alternative solution to the computing problem. In this context, instead of using multithreading, multiprocessing approach is used with the following benefits.

- We can circumvent the effects of GIL by spawning multiple Python processes.
- Running multiple processes prevents single point of failure problems, if one process crashes others are unaffected.
- Inter-process communication through message queues (i.e. Redis, RabbitMQ) make the system more observable.

For inter-process communication Redis is used for its reliability and speed. Redis (Remote Dictionary Server) is in a memory key-value store that is optimized for speed, and reliability. Redis can be used as a cache for frequently accessed data, ephemeral key-value databases, or a message broker between processes. In our project Redis is used as a message broker between different processes. In our project we make use of publisher-subscriber capability of Redis where data producing components such as object detection

component, LIDAR component etc. "publish" data to a certain topic, and data consuming components such as motor driver "subscribe" to relevant topics. This is the other major change we include our project after CM2.

4.4 Changes & Updates

At this point of the project, we have achieved most of our objectives for the determined milestones. As we stated during our P4 presentation, after the CM3, we have delimited the constraints and the requirements of the fourth scenario which was found to be ambiguous for several cases like what happens when a robot sees both a peer and the target. With the new motors and appropriate wheels, a new design for smaller robots have been made and initial assembly has been obtained for the demonstration of CM3.

We were at the workshop stage during the CM3 demonstration, hence the robots were not suitable for on road demonstration. We were aware of this incident and asked for a demonstration in the laboratory. Even though it was not in the plan of CM3 demonstration, after the official demonstration, we willingly tried to demonstrate the second scenario with unfinished robots, and as expected we had problems with the stability of our wheels. It was a predicted complication, and we will be preventing the problem with concrete assembly for the future trials.

When it comes to the promised improvements and developments, we believe, we have successfully implemented necessary adjustments according to our design and performance requirements. First of all, the implementation of the LiDAR and its tests have been completed. During our CM3 demonstration we presented the working mechanism of our LiDAR implementation. LiDAR helps us to avoid obstacles and to prevent crashes for our robots. During our presentation, we adjusted the LiDAR to scan the 20° in front of the robot, and our controller automatically prevents any movement in the direction of an obstacle within 20cm. However the usage of its results is extended during the third and fourth scenarios with the extensive controller that tries to avoid the obstacle and continues to track the target or a peer. In peer detection and collective transfer 70 cm is chosen as the threshold, and 50 cm during converging the target. The data from a total of 30 degrees is filtered in this process.

In addition to LiDAR implementation, we also successfully implemented the live camera feed sharing via ESP32. The current frame obtained from the detection and tracking system is serially transferred to the ESP32 from either Jetson Nano or Raspberry Pi, according to the robot configurations. Then the feed is streamed from a localhost of each ESP32. It is possible to view each stream by connecting to each ESP32 individually at this point, since ESP32 behaves as a hotspot for a Wi-Fi connection. Our system also allows multiple connections per feed, in other words, multiple clients can stream the feed of a single robot simultaneously. We tried several settings to adjust the frame per second-quality trade-off. At this point, we achieved the minimum video quality that we promised in our early reports (144p), with a higher FPS count of 20. We have some ideas to increase the video quality of the feed, for better visuals even though we achieved the requirement that we promised.

Moreover, we also implemented Redis for our robots, which is an open source, in-memory data structure store, that is used as a blazing fast database since it is stored in RAM. The company asked for the ROS implementation for the communication between

several programs, however ROS creates a huge overhead especially on Raspberry Pi, when the ongoing programs like on device image processing, target tracking, driving motors as well as the LiDAR and finally the live feed communication, we do not have any tolerance for extra overhead. Therefore, instead of relying on ROS for inter-process communication, we implemented Redis based solution for this purpose. To give examples, obtaining data from LiDAR runs as an external process and the results are shared with the controller via Redis database. Another benefit of the Redis system can be proven by the camera feed aspect of the project. When the tracking frame is obtained from the tracking system, the frame is uploaded to the Redis database, where the serial process can send the frame to ESP32 for the live feed, the tracking system can work on the next frame in parallel, while the controller also adjusts to the new information obtained from the frame concurrently.

The above-mentioned search algorithm has been developed and applied for mapping and target search in environment with obstacles. Due to the curfew between April 29 and May 17, we could not do our work in the lab. In this process, we were able to execute Scenario 4 and naturally Scenario 3 on a single robot in our homes. This also prevented Scenario 5 from being tested, as at least two robots must be together. We are planning to perform swarm tests of our robots after May 17th. In this process, we will use our demo repeat right provided by our department and share the final version with you.

5 Results, Discussions, and Future Directions

5.1 Results

As discussed before, the project is designed around five different scenarios outlined by Roketsan, and we designed our work packages around these scenarios. Every scenario poses new challenges, and acts as a building block for the next one. In the first scenario, as requested, one robot should detect, and converge to the predefined target which consists of three circles within a 4 meters distance by using only a camera as an environmental sensor. In the second scenario, a robot is chosen to be the pioneer robot, and the robot is remotely controlled. Remaining two robots should follow the remotely controlled pioneer robot by using camera feed and LiDAR in a predefined formation that is single line. In the third scenario, all robots should be able to converge on the target autonomously. Each robot should detect the target, and/or their peers which are defined with three pentagons. When a robot detects a peer, it follows the peer, or converges on the target if the target is identified first. In the fourth scenario, on top of the third scenario, all robots should be able to detect and avoid obstacles using LiDAR data using our custom obstacle avoidance algorithm. In the last scenario, robots are programmed to flee from hostile units. During every scenario, every robot streams the data collected from the field such as the camera feed and distance data collected from LiDAR through WiFi in real-time. The minimum video quality is achieved as promised in our early reports (144p), with a higher FPS count of 20 rather than 10 fps. As we mentioned in the functional requirements, it is of great importance that the environment is illuminated correctly rather than well lit. Since the cameras that the company provided to us were night vision, serious performance losses were experienced in cases where sunlight came directly.

5.2 Discussions and Lessons Learned

To briefly summarize, in this project we developed autonomous mobile robots with different configurations that can collectively perform a common objective by identifying other robot's location, and condition in uncharted flat environments. In this context, we developed systems that can process sensor data gathered from LIDARs and cameras in real time to identify targets while avoiding obstacles in the surrounding environment. Due to hardware limitations auxiliary components such as Ardunios, and ESP32 microcontrollers were used to save computing power on our main microcontrollers Raspberry Pi, and Jetson Nano. In this context, ESP32 is used to stream camera data through WiFi, and Arduinos are used to drive PWM motors. Lastly, one of design constraints was to develop heterogeneous robot structures composed of different cameras and controllers.

The lack of any RF communication between robots was the biggest challenge at the beginning of the project. The fact that there is no communication between the robots but the collected data will be displayed instantly on the computer has further increased this difficulty. For this purpose, although the Beacon set was purchased the previous year, the application of this system was abandoned because the need to place the transmitter antennas at certain points for the set to work properly would limit the movement area and it would be difficult to place such equipment in the environment in a project carried out in the military field. In addition, an ESP32 microcontroller was chosen to ensure streaming security and a certain level of secure connection was achieved by installing its own web server on it. This was the first big problem we encountered in the project and that we had to solve.

In Spring semester during scenario 2 studies, we encountered the problem of low performance of ZK-4WD robots allocated to us by the company. Although robots were different from DAGU Wild Thumper, heterogeneous structure and relatively low performance were required, the provided motors prevented the robot from carrying its weight to a great extent. We had to change the robots due to the robots not being able to keep up with the performance of the DAGU and the problems of speeding up and slowing down and making a smooth turn.

In our work on Scenario 3, we realized that it is very difficult to execute the scenario execution that we propose in the first period without communication. In the main requirements of the scenario, the fact that the robots arrive at the target together was not fully defined, which left us undecided about the scope of the scenario. The fact that it was a defense industry project and the robots were able to understand the situation of other swarm members through cameras and LIDAR brought along a very important problem: what happens if some of the robots are unable to continue the mission? In this case, the herd must continue to perform the task somehow, but the lack of communication greatly limits the robots' ability to determine the situation. As the scenario was open to interpretation and improvement, we talked to our company mentors on this issue, and presented them the problems that may occur and the solutions we plan to realize. In this case, since they are clients, we came up with a solution in line with their expectations. During this process, we learned that unexpected problems may arise during the continuation of the project, and some work packages can be more complex and challenging than expected.

Furthermore, we learned how important budget management and planning is during

the project process. We had to make a new budget plan in Spring semester due to the problems we did not anticipate when planning in Fall semester. The fact that we could not procure LIDAR, which is our equipment with the highest price, from the company's budget, would cause almost one robot to be used without LIDAR and the difficulty of the project would increase one more level, although it was stated by the company that every robot should have LIDAR in the project requirements. Fortunately, we managed to complete our budget thanks to TUBITAK 2209-B support. In this process, we learned that a certain amount should always be left in budget planning and that the requirements such as material / service should be met before the project starts.

The coincidence of the project with the COVID-19 pandemic was another compelling factor. The imposition of curfews and the inability to obtain permission, especially during the period we determined as the most important process of the project and where we are planning to conduct many tests, caused a great disruption to our work. This problem, which we are experiencing due to this reason that we do not have, has made us experience the situation of working remotely brought about by the pandemic.

5.3 Future Directions

In this project we were provided with relatively low powered hardware which directly affects the performance of robots. Higher quality cameras and better LIDARs might provide better sensing capabilities and increase the accuracy of the measurements. However, increasing the number, and the quality of the measurements can incur additional computational costs. Computational resources can be improved by using microcontrollers with more high-power cores with higher clock frequencies, and CUDA enabled integrated GPUs. Moreover, better computing capabilities can enable us to run more complex algorithms faster, thus providing better performance, and reliability. By upgrading the chassis design, and solving the LIDAR's beam abduction problem with additional distance sensors can enable task execution in harsher environments.

6 Equipment List

Detailed equipment list is provided in Table 1. The "by company" obtaining method provided in Table 1 corresponds to the components that are provided by the company in the beginning of the project and utilized in the final versions of the robots. The "by purchase" obtaining method provided in Table 1 corresponds to additional purchase for the robots from the total budget of ROKETSAN and TUBITAK which are 500$ and 655$ respectively.

Table 1: Detailed Equipment List

Equipment	**Obtaining Method**	**Unit Cost**
Wild Thumper 6WD 34:1 Robot	1× from company	329$
r34:1 Metal Gear Reductor 48CPR Encoder 6V DC Motor	8× by purchase	21$
Large Terrain Wheels 125mm×58mm	8× by purchase	6.5$
BTS7960B 20 Ampere Motor Driver Card	2× from company 4× by purchase	8$
NVIDIA Jetson Nano	1× from company	170$
Raspberry Pi 4B, 4GB	1× from company 1× by purchase	126$
Arduino Nano	3× from company	24$
ESP32 WiFi + Bluetooth Dual-Mode Dev. Board	3× by purchase	6$
Leopard Power $5200mAh$ $25C$ $14.8V$ LiPo Battery Rplidar A2M8	2× from company 1× by purchase	77$
A2M8 RPLIDAR	2× from company 1× by purchase	579$
Raspberry Pi Mini Keyboard-Mouse set	1× from company	19$
Xiaomi Redmi 10000Mah-12W Powerbank 2 output	3× by purchase	10$
Logitech C525 HD Webcam	1× by purchase	68$
Raspberry Pi Original Camera Module V2	2× by purchase	75$
Everest KM 5535 Mouse + Keyboard Set	1× by purchase	9$
$60 \times 60 \times 25$mm Fan 12V 0.15A	2× by purchase	4$
Aluminum Heat Sink	2× by purchase	1$
Arduino Nano USB Power Cable	3× by purchase	4$
USB Cable Converter A-B	1× by purchase	4$
Raspberry Pi 4 Type C 5V 3A Power Cable	1× by purchase	2$
Total		3381$

References

[1] *Roketsan*. [Online]. Available: https://www.roketsan.com.tr/en/, ROKETSAN A.S. is an establishment of Turkish Armed Forces Foundation.

[2] W. A. Amai, P. R. Klarer, J. B. Pletta, J. W. Purvis, and R. P. Case, "Robotic all-terrain lunar exploration rover (ratler) fy93 program status report," Oct. 1994. DOI: 10.2172/10196260. [Online]. Available: https://www.osti.gov/biblio/10196260.

[3] O. Team. (May 2020). About, [Online]. Available: https://opencv.org/about/. [Accessed: 3. Nov, 2020].

[4] I. B.-G. Eugene Kagan Nir Shvalb, *Autonomous Mobile Robots and Multi-Robot Systems*. Dec. 2019, ISBN: 9781119212867.

[5] A. Urquizo. (2011). Pid controller overview, [Online]. Available: https://commons.wikimedia.org/wiki/File:PID_en.svg (visited on 11/17/2020).

[6] J. T. Feddema, C. Lewis, and D. A. Schoenwald, "Decentralized control of cooperative robotic vehicles: Theory and application," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 852–864, 2002. DOI: 10.1109/TRA.2002.803466.

[7] MATLAB. (2017). Understanding kalman filters, part 4: Optimal state estimator algorithm, [Online]. Available: https://www.youtube.com/watch?v=VFXf1lIZ3p8 (visited on 11/17/2020).

[8] R. Faragher, "Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 128–132, 2012. DOI: 10.1109/MSP.2012.2203621.

[9] M. Wu, H. Ma, M. Fu, and C. Yang, "Particle filter based simultaneous localization and mapping using landmarks with rplidar," in *Intelligent Robotics and Applications*, H. Liu, N. Kubota, X. Zhu, R. Dillmann, and D. Zhou, Eds., Cham: Springer International Publishing, 2015, pp. 592–603, ISBN: 978-3-319-22879-2.