Multi-robot Soccer - RoboCup - PRO2 Mälardalen University

Viktor Eriksson[*], Anton Grusell[†], Mudar Ibrahim[‡], Jacob Johanssson[§], Aaiza Aziz Khan[¶], Carl Larsson[‖], Johanna Melander[**], Shruti Puthiya Kunnon[††], Pontus Svensson[‡‡], Fredrik Westerbom[x], Emil Åberg[xi]
School of Innovation, Design and Engineering, M.Sc.Eng Robotics
Mälardalens University, Västerås, Sweden
Email: {[*]Ven20002, [†]agl19003, [‡]mim20004, [§]jjn20030, [¶]akn23018, [‖]cln20001, [**]Jmr19002, [††]spn23001, [‡‡]psn19003, [x]fnl18001, [xi]eag24002}@student.mdu.se

2024-11-13

I. Project Information

This paper covers the progress report for the project Multi-robot Soccer - RoboCup for the time period from 7th of October to 8th of November. This is a collaboration project between Mälardalens Universitet (MDU), Universidad de Antioquia (UdeA) and Universidad Tecnológica de Panamá (UTP). The project is about creating a Small Size League (SSL)-RoboCup robot and developing the accompanying software. It is assumed the reader is familiar with the project, for those who are not, see the project plan available in Appendix A. Table. I outlines the team members, their roles and their assigned tasks.

Name	**Role**	**Task**
Viktor Eriksson	Software developer	Collective robot behaviour
Anton Grusell	Hardware developer	3D-CAD & Body design
Mudar Ibrahim	Team leader & Software developer	Communication
Jacob Johanssson	Software developer	Collective robot behaviour
Aaiza Aziz Khan	Software developer	Communication
Carl Larsson	Software lead & Software developer	Individual robot behaviour
Johanna Melander	Hardware developer	Mechanical design
Shruthi Puthiya Kunnon	Software developer	Communication
Pontus Svensson	Team leader & Hardware lead & Hardware developer	Power-train & Electronics
Fredrik Westerbom	Hardware developer	Sensors & Hardware communication
Emil Åberg	Software developer	Individual robot behaviour & Communication

Table I
PROJECT CONTRIBUTORS, INCLUDING THEIR ROLES AND ASSIGNED TASKS.

II. Project Progress Summary

The project has seen significant progress since it was launched. As of 2024-11-13, 3 out of 14 work packages have been completed, 8 are in progress, 1 has not started yet (upcoming), 2 is on hold and 0 have been cancelled. Note that this does not include the two Hardware-in-the-Loop (HIL) work packages which are outside of both the current and the next work period. The projects main threat is time, the project might be able to deliver the individual parts, but the integration of all the parts into a complete functioning package is unrealistic. Additionally, to keep within the limited timeline and budget, optimal and tailor-made solutions had to be sacrificed in favour of simpler ones. This in turn makes it improbable for the final product to function according to the initial demands placed by the stakeholders (capable of competing). The software side of the project is progressing well except for 1.5.5 *nav2 stack configuration* which has proven especially problematic, this still remains to be solved. This has in turn resulted in 1.5.6 *Develop supporting functions* being placed on hold. The hardware development is progressing but not at the necessary speed to deliver the product by December. Problems with 2.2.1 *Manufacture PCB* and the fact that the components have yet to arrive has caused noticeable issues for the hardware team. See Table. II for a complete overview of the project status.

III. Work Package Status

All information regarding the work packages which are apart of the 7th of October to 8th of November work period are covered in the following subsections. Note that only work packages are covered.

1.1 Simulate 12 robots

- **Planning status**: On-track
- **Completion status**: Completed
- **Accountability**: Aaiza Aziz Khan, Shruthi Puthiya Kunnon, Emil Åberg

- **Recovery plan**: N/A
- **Progress**: Completed all the deliverables and tasks.
- **Issues**: N/A
- **Next steps**: N/A

1.1.2 Simulation interface

- **Planning status**: On-track
- **Completion status**: Completed
- **Accountability**: Aaiza Aziz Khan, Shruthi Puthiya Kunnon, Emil Åberg
- **Recovery plan**: N/A
- **Progress**: Completed all the deliverables and tasks.
- **Issues**: grSim crashed suddenly and remained in an unusable state until it was fixed by following the troubleshooting instructions in the INSTALL.md file available in the grSim repository. Additionally, a way to control simulation has still not been developed.
- **Next steps**: N/A

1.2 SSL interface

- **Planning status**: On-track
- **Completion status**: Completed
- **Accountability**: Aaiza Aziz Khan, Shruthi Puthiya Kunnon, Emil Åberg
- **Recovery plan**: N/A
- **Progress**: Completed all the deliverables and tasks.
- **Issues**: There are some limitations in how well the clients for ssl-vision and ssl-game-controller can be tested automatically and independently with Google Test; they rely on User Datagram Protocol (UDP) streams which need to be generated while the clients are executing in parallel. Therefore, testing of *SSL interface* has been complemented with manual tests, where *SSL interface* is integrated with ssl-vision and ssl-game-controller.
- **Next steps**: N/A

1.3 Hardware interface

- **Planning status**: On-Track
- **Completion status**: In Progress
- **Accountability**: Mudar Ibrahim, Aaiza Aziz Khan, Shruthi Puthiya Kunnon, Fredrik Westerbom, Emil Åberg
- **Recovery plan**: N/A
- **Progress**: Currently working on 1.3.5 *Sensor interface*, specifically 1.3.5.1 *IMU*.
- **Issues**: Need admin access for installation of STM32CubeIDE to be able to write code for the STM32 board.
- **Next steps**: Sub-work package 1.3.5 *Sensor interface*.

1.3.5 Sensor interface

- **Planning status**: On-Track
- **Completion status**: In Progress
- **Accountability**: Mudar Ibrahim, Aaiza Aziz Khan, Shruthi Puthiya Kunnon, Fredrik Westerbom, Emil Åberg
- **Recovery plan**: N/A
- **Progress**: Deliverable 1.3.5.1 *IMU* is ongoing however, implementation of Serial Peripheral Interface (SPI) has been cancelled due to the support of Inter-Integrated Circuit (I^2C) for the IMU, Light Detection and Ranging (LIDAR), and RGB sensor.
- **Issues**: Unforeseen issues understanding and implementing the skeleton code generated by the graphical programming environment. Therefore the implementation of code has been delayed.
- **Next steps**: 1.3.5.1.2. *I^2C implementation*, 1.3.5.1.3. *UART implementation*. Then the rest of the Deliverables 1.3.5.2 - 1.3.5.6

1.4 Communication protocol

- **Planning status**: On-Track
- **Completion status**: On Hold
- **Accountability**: Aaiza Aziz Khan, Shruthi Puthiya Kunnon, Emil Åberg
- **Recovery plan**: N/A
- **Progress**: Work has begun on establishing communication between PC and Raspberry Pi (1.4.1).
- **Issues**: HDMI is not available.
- **Next steps**: Ability to send and receive data.

1.5 Individual robot behaviour

- **Planning status**: Off-track
- **Completion status**: In Progress
- **Accountability**: Carl Larsson
- **Recovery plan**: Enlist additional resources internally (Pontus Svensson, the only other experianced nav2 and Robot Operating System 2 (ROS2) developer) and enlist external resources (post the problem on Stack Exchange).
- **Progress**: All other deliverables and tasks except 1.5.5 *nav2 stack configuration* and the sub-work package 1.5.6 *Develop supporting functions* have been completed. 1.5.5 *nav2 stack configuration* is currently being worked on.
- **Issues**: Creating proper unit and integration tests for more advanced and interwoven parts of the code. 1.5.5 *nav2 stack configuration* is proving problematic and has taken significantly longer time to complete than expected, this has caused 1.5.6 to be placed On Hold.
- **Next steps**: 1.5.6 *Develop supporting functions*.

1.5.6 Develop supporting functions

- **Planning status**: On-track
- **Completion status**: On Hold
- **Accountability**: Carl Larsson
- **Recovery plan**: N/A
- **Progress**: Work has started on 1.5.6.1 *Run Simulation/Hardware interface*, specifically integrating the *Simulation interface* (1.5.6.1.1), no noteworthy progress has been made thus far.
- **Issues**: Waiting for 1.4 *Communication protocol* to be completed before being able to start working on 1.5.6.2 *Ability to send data*, *1.5.6.3 Ability to receive data*, *1.5.6.4 Initialize the robot*.
- **Next steps**: 1.5.6.2 *Ability to send data*, 1.5.6.3 *Ability to receive data*, 1.5.6.4 *Initialize the robot*.

1.6 Collective robot behaviour

- **Planning status**: On-Track
- **Completion status**: In Progress
- **Accountability**: Viktor Eriksson, Jacob Johansson
- **Recovery plan**: N/A
- **Progress**: Completed 1.6.1 *Research AI-algorithms*. Working on task 1.6.2.1 *Develop the AI-algorithm*, finishing the structure for the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm. 1.6.4 is currently in progress and will result in the ability to receive data from the external systems.
- **Issues**: Lack of documentation on libraries (i.e. PyTorch) has slowed down the development of the algorithm and functions.
- **Next steps**: 1.6.3 *Be able to send data*.

2.1 Base design

- **Planning status**: On-Track
- **Completion status**: In Progress
- **Accountability**: Anton Grusell, Johanna Melander, Pontus Svensson, Fredrik Westerbom
- **Recovery plan**: N/A
- **Progress**: 2.1.1 *Hardware design to meet SSL-rules* and 2.1.2 *Design modular components* completed. 2.1.3 *3D-design* and 2.1.4 *Circuit design and integration* is in progress.
- **Issues**: Components went out of stock. Restrictions on which components could be used due to collaboration with UdeA.
- **Next steps**: 2.1.3 *3D-design*, 2.1.4 *Circuit design and integration*.

2.1.3 3D-design

- **Planning status**: On-Track
- **Completion status**: In Progress
- **Accountability**: Anton Grusell, Johanna Melander, Pontus Svensson, Fredrik Westerbom
- **Recovery plan**: N/A
- **Progress**: 2.1.3.1 *Chassis design* waiting for completion of the circuits before continuing. 2.1.3.2 *Wheels design* is in the final stages, awaiting feedback from the manufacturer. 2.1.3.3, 2.1.3.4 and 2.1.3.5 are all in progress.
- **Issues**: Size requirements makes every decision take more time. As more components are finished, revisions need to be made to ensure compatibility.
- **Next steps**: Finish 2.1.3.3, 2.1.3.4, 2.1.3.5.

2.1.4 Circuit design and integration

- **Planning status**: On-Track
- **Completion status**: In Progress
- **Accountability**: Anton Grusell, Johanna Melander, Pontus Svensson, Fredrik Westerbom
- **Recovery plan**: If certain circuits are not completed in a timely manner, those functionalities will be excluded from the robot.
- **Progress**: 2.1.4.1 *Design powertrain circuit* is completed. 2.1.4.2 *Design kicker circuit* is completed. 2.1.4.3 *Design dribbler circuit* is completed. 2.1.4.4 *Design microcontroller circuit* is mostly completed, awaiting which type of connector is to be used for mainboard to kicker board connection. 2.1.4.5 *Design IC for sensors* is completed. 2.1.4.5.3 *Design break beam IC* was cancelled. 2.1.4.6 *Integrate the circuits* is in progress, the pins for the kicker control needs to be decided before the deliverable is completed. 2.1.4.7 *Design mainboard PCB* is in progress, dependent on 2.1.4.6. 2.1.4.8 *Design carrier PCB for motor driver* is completed. 2.1.4.9 *Design kicker PCB* is in progress. 2.1.4.10 *Design basestation PCB* was cancelled due to time limitations.
- **Issues**: Finding suitable board-to-board connectors is difficult and expensive, leading to a less suitable option being chosen.
- **Next steps**: 2.2.1 *Manufacture PCB*.

IV. Completed and Upcoming Activities

The project has achieved the following milestones thus far:

- *Simulation interface* done
- *SSL interface* done

The completed and the upcoming activities for the project are shown in Table. II, which is the Work Breakdown Structure (WBS) with the addition of a new column indicating the status of the activity. Note that any activity which is outside of the next work period, spanning from November 8 to December 6, is seen as N/A (-).

ID	Work Class	Name (Work entailed)	Status
-	Root	Multi-robot Soccer - RoboCup	-
1	Category	Software	In Progress
1.1	Work Package	Simulate 12 robots	Completed
1.1.1	Deliverable	Setup grSim	Completed
1.1.1.1	Task	Download grSim	Completed
1.1.1.2	Task	Run the simulation correctly	Completed
1.1.2	Work Package	Simulation interface (API)	Completed
1.1.2.1	Deliverable	Integrate Protobuf	Completed
1.1.2.2	Deliverable	Move a single robot	Completed
1.1.2.3	Deliverable	Activate kicker on a single robot	Completed
1.1.2.4	Deliverable	Activate dribbler on a single robot	Completed
1.1.2.5	Deliverable	Execute any desired control over any chosen robot in grSim	Completed
1.2	Work Package	SSL interface (API)	Completed
1.2.1	Deliverable	Integrate ssl-vision	Completed
1.2.1.1	Task	Receive single package containing positional data	Completed
1.2.1.2	Task	Receive continuous positional data	Completed
1.2.2	Deliverable	Integrate ssl-game-controller (human referee and optional auto referee)	Completed
1.2.2.1	Task	Receive a single package containing referee commands/signals	Completed
1.2.2.2	Task	Receive continuous stream of referee commands/signals	Completed
1.2.2.3	Task	Integrate AutoReferee	Completed
1.2.3	Deliverable	Develop an automatic game controller (will be used when training the AI)	Completed
1.2.3.1	Task	Research and obtain proper understanding of SSL-rules	Completed
1.2.3.2	Task	Implement the automatic game controller system	Completed
1.3	Work Package	Hardware interface (API)	In Progress
1.3.1	Deliverable	Basestation communication	Cancelled
1.3.1.1	Task	Configure basestation RF communication	Cancelled
1.3.2	Deliverable	Wheel motor interface	In Progress
1.3.2.1	Task	Establish communication using UART	In Progress
1.3.2.2	Task	Implement FOC	Upcoming
1.3.2.3	Task	Implement a velocity distributor (subscribing to cmd_vel)	Upcoming
1.3.3	Deliverable	Kicker interface	Upcoming
1.3.3.1	Task	GPIO implement activation	Upcoming
1.3.4	Deliverable	Dribbler interface	Upcoming
1.3.4.1	Task	Implement PWM control	Upcoming
1.3.5	Work Package	Sensor interface	In Progress
1.3.5.1	Deliverable	IMU	In Progress
1.3.5.1.1	Task	SPI implementation	Cancelled
1.3.5.1.2	Task	I^2C implementation	In Progress
1.3.5.1.3	Task	Communicate using UART	Upcoming
1.3.5.2	Deliverable	Wheel encoders	Upcoming
1.3.5.2.1	Task	Configure wheel encoders	Upcoming
1.3.5.3	Deliverable	Camera	Upcoming
1.3.5.4	Deliverable	LIDAR	Upcoming
1.3.5.4.1	Task	I^2C implementation	Upcoming
1.3.5.4.2	Task	Communicate using UART	Upcoming
1.3.5.5	Deliverable	RGB sensor	Upcoming
1.3.5.5.1	Task	I^2C implementation	Upcoming
1.3.5.5.2	Task	Communicate using UART	Upcoming
1.3.5.6	Deliverable	Break beam	Upcoming
1.3.6	Deliverable	Provide robot status information	Upcoming
1.3.6.1	Task	CPU temperature	Upcoming
1.3.6.2	Task	Battery charge	Upcoming

1.3.7	Deliverable	Establish communication between Raspberry Pi and Nucleo 144	In Progress
1.3.7.2	Task	Install micro-ROS on Raspberry Pi and Nucleo 144 (UART)	In Progress
1.3.7.3	Task	Implement ROS functionality	Upcoming
1.3.8	Deliverable	Establish communication between Nucleo 144 and motor drivers	Upcoming
1.3.8.1	Task	Communicating using UART	Upcoming
1.4	Work Package	Communication protocol (API)	On Hold
1.4.1	Deliverable	Establish communication between PC and Raspberry Pi	On Hold
1.4.1.1	Task	Using Wi-Fi	On Hold
1.4.2	Deliverable	Integrate Protobuf	Upcoming
1.4.3	Deliverable	Utilizing ROS2	Upcoming
1.4.4	Deliverable	Ability to switch between two carrier frequencies	Upcoming
1.5	Work Package	Individual robot behaviour (finding the best way to do the things it has been commanded to do)	In Progress
1.5.1	Deliverable	Research viable path planning algorithms	Completed
1.5.2	Deliverable	Integrate ROS2	Completed
1.5.2.1	Task	Getting sensor data	Completed
1.5.2.2	Task	Getting odometry data	Completed
1.5.3	Deliverable	Develop path planning algorithm	Completed
1.5.3.1	Task	Build using nav2	Completed
1.5.4	Deliverable	Implement robot ability to shoot	Completed
1.5.4.1	Task	Angle correctly towards target	Completed
1.5.4.2	Task	Shoot towards target	Completed
1.5.5	Deliverable	nav2 stack configuration	In Progress
1.5.5.1	Task	Create nav2 parameter files	In Progress
1.5.5.2	Task	Create nav2 launch files	In Progress
1.5.6	Work Package	Develop supporting functions	On Hold
1.5.6.1	Deliverable	Run Simulation/Hardware interface	On Hold
1.5.6.1.1	Task	Run and call on Simulation interface	On Hold
1.5.6.1.2	Task	Run and call on Hardware interface	On Hold
1.5.6.2	Deliverable	Ability to send data	On Hold
1.5.6.2.1	Task	Sending robot status (CPU temperature and battery charge)	On Hold
1.5.6.2.2	Task	Sending acknowledgements that commanded task has been completed	On Hold
1.5.6.3	Deliverable	Ability to receive data	On Hold
1.5.6.3.1	Task	Ability to receive initialization data from collective robot behaviour (ID, home playing side, starting pose)	On Hold
1.5.6.3.2	Task	Ability to receive control commands from collective robot behaviour	On Hold
1.5.6.3.3	Task	Ability to receive pose correction data from collective robot behaviour	On Hold
1.5.6.4	Deliverable	Initialize the robot	On Hold
1.5.6.4.1	Task	Getting robot ID	On Hold
1.5.6.4.2	Task	Obtaining information about which playing side of the field is friendly	On Hold
1.5.6.4.3	Task	Getting the start pose	On Hold
1.6	Work Package	Collective robot behaviour/ Centralised strategy planner (Handles what each individual robot should do to obtain a win for the team)	In Progress
1.6.1	Deliverable	Research AI-algorithms	Completed
1.6.1.1	Task	Outline actions	Completed
1.6.1.2	Task	Outline world representation (input)	Completed
1.6.2	Deliverable	Develop AI for strategy planning	In Progress
1.6.2.1	Task	Develop the AI-algorithm	In Progress
1.6.2.2	Task	Train the AI-algorithm	Upcoming
1.6.3	Deliverable	Be able to send data	Upcoming
1.6.3.1	Task	Provide initialization data to all robots	Upcoming
1.6.3.2	Task	Be able to send control commands to robots	Upcoming
1.6.3.3	Task	Be able to send pose correction data to robots	Upcoming

1.6.4	Deliverable	Be able to receive data	In Progress
1.6.4.1	Task	Be able to receive robot status information	In Progress
1.6.4.2	Task	Be able to receive referee commands and signals from SSL interface	In Progress
1.6.4.3	Task	Be able to receive positional data from SSL interface	Completed
1.6.5	Deliverable	Ability to switch playing half on demand	Upcoming
2	Category	Hardware	In Progress
2.1	Work Package	Base design	In Progress
2.1.1	Deliverable	Hardware design to meet SSL-rules	Completed
2.1.1.1	Task	Research for chassis design	Completed
2.1.1.2	Task	Chassis BOM	Completed
2.1.1.3	Task	Powertrain BOM	Completed
2.1.1.4	Task	Sensor BOM	Completed
2.1.1.5	Task	Kicker BOM	Completed
2.1.2	Deliverable	Design modular components	Completed
2.1.2.1	Task	Allow for battery swap	Completed
2.1.2.2	Task	Allow for motor swap	Completed
2.1.3	Work Package	3D-design	In Progress
2.1.3.1	Deliverable	Chassis design	Completed
2.1.3.2	Deliverable	Wheel design	Completed
2.1.3.2.1	Task	Wheel hub design	Completed
2.1.3.2.1	Task	Subwheel design	Completed
2.1.3.3	Deliverable	Dribbler design	In Progress
2.1.3.4	Deliverable	Kicker assembly design	Completed
2.1.3.5	Deliverable	Mounting design	In Progress
2.1.3.5.1	Task	Encoder mount	In Progress
2.1.3.5.2	Task	Motor mount	Completed
2.1.3.5.3	Task	Battery mount	Completed
2.1.3.5.4	Task	Dribbler mount	Completed
2.1.4	Work Package	Circuit design and integration	In Progress
2.1.4.1	Deliverable	Design powertrain circuit	Completed
2.1.4.2	Deliverable	Design kicker circuit	In Progress
2.1.4.3	Deliverable	Design dribbler circuit	Completed
2.1.4.4	Deliverable	Design microcontroller circuit	In Progress
2.1.4.5	Deliverable	Design IC for sensors	Completed
2.1.4.5.1	Task	Design IMU IC	Completed
2.1.4.5.2	Task	Design wheel encoder IC	Completed
2.1.4.5.3	Task	Design break beam IC	Cancelled
2.1.4.6	Deliverable	Integrate the circuits	In Progress
2.1.4.7	Deliverable	Design mainboard PCB	In Progress
2.1.4.8	Deliverable	Design carrier PCB for motor driver	Completed
2.1.4.9	Deliverable	Design kicker PCB	In Progress
2.1.4.10	Deliverable	Design basestation PCB	Cancelled
2.2	Work Package	Manufacturing	In Progress
2.2.1	Deliverable	Manufacture PCB	In Progress
2.2.1.1	Task	Mainboard	Upcoming
2.2.1.2	Task	Basestation	Cancelled
2.2.1.3	Task	Kicker board	Upcoming
2.2.1.4	Task	Motor driver carrier board	In progress
2.2.2	Deliverable	Manufacture 3D-designed items	Upcoming
2.2.2.1	Task	Manufacture chassis	Upcoming
2.2.2.2	Task	Manufacture wheels	Upcoming
2.2.2.3	Task	Manufacture dribbler	Upcoming
2.2.2.4	Task	Manufacture kicker assembly	Upcoming

2.2.2.5	Task	Manufacture mountings	Upcoming
2.3	Work Package	Testing	Upcoming
2.3.1	Deliverable	Test motors	Upcoming
2.3.2	Deliverable	Test kicker	Upcoming
2.3.3	Deliverable	Test dribbler	Upcoming
2.3.4	Deliverable	Test sensors	Upcoming
2.3.4.1	Task	Test IMU	Upcoming
2.3.4.2	Task	Test wheel encoders	Upcoming
2.3.4.3	Task	Test camera	Upcoming
2.3.4.4	Task	Test LIDAR	Upcoming
2.3.4.5	Task	Test RGB sensor	Upcoming
2.3.5	Deliverable	Validate the integrated circuits	Upcoming
2.3.5.1	Task	Battery works correctly	Upcoming
2.3.5.2	Task	SSL-rule compliance	Upcoming
3	Category	HIL	-
3.1	Work Package	Develop HIL	-
3.1.1	Deliverable	Design a system to integrate hardware in simulation	-
3.1.2	Deliverable	Test the HIL in simulation	-
3.1.3	Deliverable	Ensure a smooth transition from simulation to physical robot hardware	-
3.2	Work Package	Validate hardware platform	-
3.2.1	Deliverable	Conduct HIL test to validate the performance	-

Table II: Summary of completed and upcoming activities. Green for completed, orange for in progress, yellow for on hold, red for cancelled, blue for upcoming (part of the next work period which spans from November 8 to December 6) and gray for N/A (-). N/A (-) is anything outside the next work period (November 8 to December 6).

V. Issues, Blockers, and Risks Status

A number of issues were identified, and in an attempt to minimise their impact on the project they were combined with a mitigation action. The issues and their corresponding mitigation plan can be seen in Table. III.

Issue	Mitigation
1.2 & 1.6: Compilation error in centralised-ai repository after merge	Allocate additional work force to solve the problem.
1.5: nav2 stack error	Enlisting Pontus Svenssons help (the only other member with ROS2 and nav2 experience). Additionally, asking the question on Stack Exchange to allow external resources to help find the answer.
1.3.5: Difficulty implementing SPI	Reading from multiple sources online, reading manuals and data sheets. Looking at examples provided by Würth. Will look at the examples provided by STMicroelectronics, as well as other competitive teams such as TIGERs Mannheim and Delft Mercurians.
2.1.3: Size constraints for parts	Redesigning and moving already made CAD models to make more room to fit components.
1: Software modules have so far only been tested individually. There may be issues or even necessary functionality that is missing that is only apparent once integration has been attempted.	Integrate different software modules at the earliest possible moment, thus leaving time to fix any potential issues which might arise.
1.1.2: Simulation is currently limited to only being able to run in real time. Without being able to control speed of simulation, the capacity to train the AI model will be significantly reduced.	Dedicate time to try and find a way to control simulation speed. Worst case scenario, the loss of AI performance has to be accepted.
2: Components arriving late, this will delay development and testing of components.	The *Hardware interface* will be implemented until the components arrive.
2.2: No sponsor for PCB manufacturing has been found.	The project will use an external PCB supplier.

Table III
ALL IDENTIFIED ISSUES AND THE STEPS BEING TAKEN TO MITIGATE THEM.

VI. Individual Contributions

This section will provide a summary of what each team member has worked on, their challenges and their contributions to the project goals:

Viktor Eriksson

- **Work**: Did research on AI-algorithms 1.6.1, and then started to develop the AI 1.6.2.1. Will be using Multi-Agent Proximal Policy Optimization (MAPPO) algorithm for centralised strategy planing.
- **Challenges**: The amount of information found about MAPPO was very limited, the most informative paper does not provide fully correct information and contains misspellings. Despite this, MAPPO was chosen because the algorithm has proven to be efficient in multi agent reinforcement learning applications. Additional challenges include combining other team members codes and having to adapt the code to make the integration work.
- **Contributions to project goals**: Develop the strategy planing 1.6.2.

Anton Grusell

- **Work**: 2.1.1.2 *Chassis BOM*, 2.1.3 *3D-design*, 2.1.3.1 *Chassis design*, 2.1.3.2 *Wheel design*, 2.1.3.3 *Dribbler design*, 2.1.3.5 *Mounting design* 2.1.4.3 *Design dribbler circuit*, PRO1, PRO2,
- **Challenges**: Revisions to designs for compatibility, fulfilling the size constraints. The dribbler design proved problematic when trying to reduce size.
- **Contributions to project goals**: Designing the chassis and wheels in CAD, designing the dribbler system and a test rig for it. Designing encoder mounts for the wheels.

Mudar Ibrahim

- **Work**: Mostly worked on team management, Progress Report structure and keeping work packages on-track. Also worked on 1.3 *Hardware interface*, writing I^2C communication protocol. Additionally, reviewing this document on a continuous basis.
- **Challenges**: Using only one STM32 board extended the testing time, since the team had to wait for access to the board to test code. Additionally, team management sometimes took longer than expected, causing other tasks to be pushed closer to their deadlines.
- **Contributions to project goals**: 1.3 *Hardware interface* and PRO2 structure.

Jacob Johansson

- **Work**: 1.6.1 Helped research state-of-the-art algorithms used for cooperative robots. 1.6.2.1 Develop core functions of the AI algorithm.
- **Challenges**: Papers lacking the proper citations for the original sources of certain equations used. This complicated development of the MAPPO algorithm since it caused confusion about which equations was to be used.
- **Contributions to project goals**: Writing functions for the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm, which dictates the strategy behaviour of the robots.

Aaiza Aziz Khan

- **Work**: 1.1 *Simulate 12 robots*, 1.1.2 *Simulation interface*, 1.2 *SSL interface*.
- **Challenges**: Initial difficulties in familiarising with new techniques, standards and tools.
- **Contributions to project goals**: Contributed in developing ssl-vision client, automatic game controller, and some functions in 1.1.2 and 1.2. Writing test suites for ssl-vision client, ssl-game-controller client and *Simulation interface*.

Carl Larsson

- **Work**: 1.5 *Individual robot behaviour*, 1.5.6 *Develop supporting functions*, PRO1, PRO2, management and support for the team.
- **Challenges**: Time management and designing a suitable software structure so that the different software parts will be able to be integrated properly. nav2 stack error, prolonging the expected development time of 1.5.5 *nav2 stack configuration* significantly.
- **Contributions to project goals**: Writing code for the executable which will run on the robots, it is tasked with finding a suitable way to realize the commands from the centralized strategy planner. Substantial documentation efforts.

Johanna Melander

- **Work**: 2.1.1.5 *Kicker BOM*, 2.1.3.2 *Wheel design*, 2.1.4.2 *Design kicker circuit*, PRO1, PRO2.
- **Challenges**: The test solenoid did not arrive on time, so it has not been possible to experiment as much as desired.
- **Contributions to project goals**: Researched about the wheels and kicker design. Designing a simple kicker test rig and an initial version of the wheels in CAD. Designing kicker circuit in KiCAD, created Bill of Materials (BOM) for kicker circuit.

Shruthi Puthiya Kunnon

- **Work**: Work Packages 1.1 *Simulate 12 robots*, 1.1.2 *Simulation interface*, 1.2 *SSL interface*.
- **Challenges**: Initially, struggled to understand the standards and tools.
- **Contributions to project goals**: Worked on the development of 1.1.2 and 1.2. Developed test suites for *Simulation interface*, *SSL interface*, ssl-game-controller.

Pontus Svensson

- **Work**: 2.1.1.3 *Powertrain BOM*, 2.1.2 *Design modular components*, 2.1.4 *Circuit design and integration*, 2.2 *Manufacturing*, 2.2.1.4 ordered motor driver carrier board. Trying to make the collaboration work with UdeA and UTP.
- **Challenges**: Some initial difficulties regarding which components were allowed to be used, delayed the initial work packages. Size constraints in the robot leads to peculiar solutions for fitting all Electronic Speed Controllers (ESCs) and sensors resulting in almost every rule imaginable being broken regarding PCB design.
- **Contributions to project goals**: Designing the mainboard, motor carrier, and encoder PCB, reaching out to other teams for guidance and feedback.

Fredrik Westerbom

- **Work**: Deliverables 1.3.2, and 1.3.5.1, and 2.1.4.5. Tasks 2.1.1.4, and 2.3.4.1.
- **Challenges**: The task 2.3.4.1 requires a communication protocol to receive and transmit data. SPI was initially chosen (before abandoned) due to its speed and low latency. Additional pre-requisite tasks have been discovered, the duration of the task has therefore increased. These additional tasks include research into SPI, installing and getting familiarized with the recommended programming environment as well as the development board NUCLEO-H723ZG. Deliverable 1.3.2 is in the startup phase which in short includes pre-studies to familiarize with the development environment as well as acquire the necessary level of understanding to implement Field-Oriented Control (FOC).
- **Contributions to project goals**: The deliverables and tasks are identified to be crucial sub-tasks for the success of the project.

Emil Åberg

- **Work**: Work Packages 1.1, 1.1.2, organize *SSL interface* (Work package 1.2), help with deliverables 1.2.1, 1.2.2, implement collision detection and tests for the automatic game controller (1.2.3.2). Just started on *Hardware interface* (1.3).
- **Challenges**: A way to control simulation speed has not been developed yet. For the sake of training the AI, it would be best not to be limited to having it run in real time.
- **Contributions to project goals**: Develop *Simulation interface*, structure and help implement *SSL interface*.

Appendix

Project plan

Project name	Client / Sponsor	Project manager
Multi-robot Soccer - RoboCup	Mikael Ekström	Mudar Ibrahim

1 Project information

This paper covers the project plan for the the project Multi-robot Soccer - RoboCup. This project is a collaboration between Mälardalens Universitet (MDU), Universidad de Antioquia (UdeA) and Universidad Tecnológica de Panamá (UTP). This collaboration is part of the internationalization of the Robotics program at MDU and is meant to improve the students skills at working in larger multi cultured groups. The project is about developing both the hardware and the software for one Small Size League (SSL)-RoboCup division B robot. The project team consists of eleven contributors split into a hardware and a software team, see Table. 1. The software team was further divided into communication, individual robot behaviour and collective robot behaviour. The hardware team was split into four groups: powertrain & electronics, sensors & embedded systems, 3D-CAD & body design and mechanical design.

Name	**Role**	**Task**
Viktor Eriksson	Software developer	Collective robot behaviour
Anton Grusell	Hardware developer	3D-CAD & Body design
Mudar Ibrahim	Team leader & Software developer	Communication
Jacob Johanssson	Software developer	Collective robot behaviour
Aaiza Aziz Khan	Software developer	Communication
Carl Larsson	Software lead & Software developer	Individual robot behaviour
Johanna Melander	Hardware developer	Mechanical design
Shruthi Puthiya Kunnon	Software developer	Communication
Pontus Svensson	Team leader & Hardware lead & Hardware developer	Power-train & Electronics
Fredrik Westerbom	Hardware developer	Sensors & Hardware communication
Emil Åberg	Software developer	Individual robot behaviour & Communication

Table 1: Project contributors, including their roles and assigned tasks.

2 Purpose

The purpose of Multi-robot Soccer - RoboCup is to cultivate international relationships, improve students skills at working in multi culture groups, further the multi-robot system research at MDU and contribute to the SSL-RoboCup community. Collaboration with people from different cultures is becoming essential in today's increasingly globalized world, as is fostering internationalization. The importance of intelligent systems research can not be underestimated given the clear rise of Artificial Intelligence (AI), which makes for a perfect opportunity to engage with RoboCup which aims to advance the state of the art for intelligent robots [1].

3 Goal

The goal of this project is to design and create one SSL-RoboCup division B robot which complies with all SSL-RoboCup regulations. Furthermore, the necessary software to control a full SSL-RoboCup team is to be developed and run in simulation. This includes a peer-to-peer communication protocol, as well as a centralised strategy planner. These are to be completed before 2025-01-10.

4 Limitations

This project will not deliver a simulation environment, the already developed grSim will be used. Additionally, no global area network will be developed. The platform will not support any type of kicks, manoeuvres or similar that is not listed in the Requirement section 6. Finally, the project will not involve any physical playing fields for testing, benchmarking or physical test games. If these were not excluded, it would hamper early development to the point where the delivery of a complete product would be placed at risk.

5 Ethical considerations

Only robot ethical considerations with humans as the ethical subjects and the ethics of good society (as is defined in Coeckelbergh book Robot Ethics [2]) is considered in this paper. Coeckelbergh [2] raises the ethical consideration of who is responsible for a robot, and if it is ethical as a developer to just abandon the robot after it has been created, leaving all responsibilities behind. Coeckelbergh [2] goes on to discuss the ethical dilemma of what type of robots could be considered, or not considered, ethical to create. In this project the responsibility of the robots, their use, and the creation of these robots fall on the stakeholder. The developers have no choice in whether it is ethical to develop this kind of robots (the project must be done in order to pass the course). Furthermore, the developers are unable to follow along with their creation after the course ends, and can thus not be held responsible for the robot. Finally, the developers are not in charge of whether the projects findings and research is published or not. The research could result in robots competing with professional football players, possibly resulting in reducing the human football scene and professional players loosing their jobs. Entertainment, especially sport, is a prominent aspect in today's society, so should the previous consideration play out, then it could lead to a transformation of the sports entertainment. This would in turn have some consequences, most likely including economical, societal, social and more. Similar considerations are brought up by Coeckelbergh [2], however this paper specifically applied those considerations to this project. Multi agent research, and robots in general, could be applied to military and surveillance sectors. This in turn raises questions of privacy and harm to humans, and who is to be responsible should that happen. Additionally, like Coeckelbergh [2] asks the reader to ponder, in the case that the robots are employed for surveillance then what if the robot does not respect human dignity? Military applications require serious considerations as well, the disconnect that a robot creates could be seen as making it easier to kill and start conflicts since it is easier to lack sympathy for a target when one never has to interact or see them. This can be further extended to the view of some AI as "black boxes", the inability to know the exact workings of an AI and what it will do, even the developers might not fully understand them. If the robot AI harms someone or something that is should not have (it was not its intended function to do so), then who is to blame?

6 Requirement

Requirements are necessary to establish in order to ensure that a satisfactory product can be produced and tested, ensuring it meets the original demands placed on the project. It is very important that clearly defined requirements exist from the onset of the project, reducing the chance of misinterpretations by the team and ensuring that the final product aligns with what the stakeholders envisioned. Additionally, it helps benchmark the system in a measurable way. This clearly shows the necessity for stakeholders involvement in establishing requirements to provide comprehensive and accurate requirements, which in turn helps minimize the amount of rework.

6.1 Product requirements

Product requirements entails the features, functionality and specifications which the final product must have to meet the demands of the stakeholders. The product requirements will serve as a blueprint for the development and design of the product that is to be delivered.

6.1.1 Functional requirements

Functional requirements describe the functions that the final product must have. The following functional requirements were established:

1. Robot
    - 1.1. Kick the ball towards designated target
    - 1.2. Pass the ball towards designated robot
    - 1.3. Keep control of the ball using the dribbler
    - 1.4. Block ball
    - 1.5. Move to designated target
    - 1.6. Omnidirectional (holonomic) movement
    - 1.7. Not crash into other robots

- Except if it is an attempt at trying to steal the ball from an opponent or to prevent opponents from scoring a goal
- Never acceptable if it causes any permanent damage on either robot

1.8. Not crash into the arena
1.9. React accordingly to referee signals and commands (see [3] for an explanation of the signals and commands)
1.9.1. Halt
1.9.2. Stop
1.9.3. Normal start
1.9.4. Force start
1.9.5. Prepare kickoff
1.9.5.1. Yellow team
1.9.5.2. Blue team
1.9.6. Perpare penalty
1.9.6.1. Yellow team
1.9.6.2. Blue team
1.9.7. Direct free kick
1.9.7.1. Yellow team
1.9.7.2. Blue team
1.9.8. Timeout
1.9.8.1. Yellow team
1.9.8.2. Blue team
1.9.9. Ball placement
1.9.9.1. Yellow team
1.9.9.2. Blue team
1.10. Send robot status information
1.11. Hot-swap battery. The battery should be swappable by plugging in the robot to an alternative source and then changing the battery.
1.12. Continuously monitor the capacity of the battery.
1.13. Continuously monitor the temperature of the Micro Controller Unit (MCU).

2. Centralised strategic planner

2.1. AI functionality
2.2. Swap field sides on demand
2.3. Ensure actions for each robot is not taken from an individualistic point of view, but from a overarching team perspective with a strategic plan aimed at obtaining victory

3. Simulation interface

3.1. Send packages containing robot commands to grSim

4. SSL interface

4.1. Receive positional data from ssl-vision
4.2. Receive referee commands from ssl-game-controller
4.2.1. Have humanly operated referee functionality
4.2.2. Have AutoReferee functionality

5. Communication protocol

5.1. Be wireless
5.2. Be peer-to-peer
5.3. Ability to select between two carrier frequencies

6. Hardware interface

6.1. Control wheel motors
6.2. Control kicker
6.3. Control dribbler
6.4. Provide sensor data
6.5. Provide robot status information

6.1.2 Non-functional requirements

Non-functional requirements describe the specifications of the deliverables in detail. The following non-functional requirements were established:

1. The centralised strategy planner must run on a central computer.
    1.1. A latency lower than 400 ms from any point in time at which the centralised strategy planner gives a command until it starts executing on the robot.
2. Be able to simulate 12 robots.
3. Communication protocol
    3.1. Use WiFi
    3.2. Use Protobuf
    3.3. Use Robot Operating System 2 (ROS2) Humble
    3.4. 5% dataloss or less
4. Each robot must have a reverse polarity protection circuit that changes polarity when the battery is connected incorrectly.
5. Each robot must have a TVS-diode for over-voltage protection.
6. The robot must have four Swedish wheels.
7. Maximum diameter of each wheel must be 70 mm.
8. The size of each wheel motor has to be 43 mm in diameter and 30 mm in length.
9. The weight of each motor for the wheels must be less than 200 g.
10. Each motor must be capable of handling 24 V (6 S battery).
11. Each motor driver must work with an input voltage of 24 V (6 S battery).
12. The weight of each motor driver must be less than 200 g.
13. Minimum rotation per minute (RPM) for the dribbler motor is 10000 RPM.
14. Maximum length of the dribbler motor is 70 mm, excluding the mounting axles.
15. The solenoid (kicker) must be capable of launching the ball at a speed of at least 2 m/s.
16. The solenoid must be capable of handling a voltage of at least 100 V.
17. The solenoid must have a diameter of less than 30 mm and be less than 60 mm in length, excluding plunger.
18. The voltage of the battery in each robot must be 6 S (24 V nominal).
19. The robot must have battery capacity to continuously operate for 30 min.
20. For the break beam sensor, which detects if the ball is close to the dribbler, the beam cannot be smaller than 3 mm in diameter.
21. The robot must fit inside a cylinder which is 180 mm in diameter and is 150 mm tall.
22. Maximum weight of the robot is 5 kg.
23. The angle between the front wheels must be 120° and between the back wheels the angle must be 90°.
24. Minimum height of the middle plate is 80 mm.
25. Screws with a uniform diameter must be used, with a minimum size of M3.
26. The robots MCU must process input from sensors and control the motors in less than 40 ms.
27. Comply with all SSL-RoboCup rules, see [3].
28. The vision pattern on the robot must follow the SSL rules and fit on the top plate.
29. The colour markings on top of the robot must comply with SSL rules.

6.2 Project requirements

The project has locked time and cost, focus will thus be placed on scope to ensure the project does not expand unrealistically, resulting in an inability to deliver the product. The following list describes what must be done in the project to be able to deliver the product:

- Weekly meetings with both simulation and hardware teams to ensure alignment and progress tracking.
- Comprehensive documentation of all development activities, including hardware design, AI algorithms, simulation results, and testing outcomes.
- Participation in at least one international RoboCup-related event or competition during the project timeline to benchmark progress.
- Limit the project as described in Limitations, see section 4.
- Keep within budget, see Project budget, section 10.
- Facilitate clear and good communication within the team, with sponsors and with stakeholders.

6.3 Prerequisites

The project demands that the sponsors provide the hardware team with full access to the B.nr 326 workshop and the tools and machines that should be present there. If additional training is required to operate these tools and machines, then the project demands that this training be afforded to the hardware team latest 2024-10-10.

7 Situational analysis and stakeholders

The situational analysis of this RoboCup project involves evaluating both internal and external factors that can impact the project development process. Understanding these important factors gives the team the ability to make accurate and strategical decisions to ensure a successful outcome.

7.1 SWOT-analysis

The main reason for this analysis is to identify strengths, weaknesses, opportunities, and threats that stand behind the projects outcome. This analysis helps the team identify areas of potential growth, manage risks, and capitalise on resources available through partnerships and institutional support.
Strengths:

- Skilled team: the project has well defined team structure with experience in various areas in both software and hardware.
- Strong team collaboration: the team members strive to maintain great relationship with each other.
- Advanced tools and technologies: use of advanced tools and technologies including ROS2, GoogleTest and tools such as OnShape for 3D-CAD modelling.

Weaknesses:

- Time constraints: the project operates within a fixed timeline, which limits the projects scope and flexibility.
- No time for physical testing: due to the project having a fixed timeline, team members have limited opportunities to test the system in real-world conditions.
- Project budget: the project budget is limited and inhibits the teams options.
- Large group experience: the team members lack experience working on a project with a group size this large.

Opportunities:

- International collaboration: opportunity to add experienced members to the team with MDUs international collaboration with UdeA and UTP, possibly obtaining even greater results.
- Learning and practice: significant potential for students to learn and use advanced technologies.

Threats:

- Technical issues: the simulation could differ from reality, which could result in unforeseen implications when deploying the physical system.
- Demands imposed by the stakeholders and sponsors: stakeholders can and may limit the choices available to team members. This could include things like which tools and features are to be used or not used, all of which will affect the quality of project deliverables.

Conclusions and actions

- The lack of experience working in large groups can be solved by dividing the team into smaller groups, each focusing on specific areas. This approach is beneficial because the project involves multiple working areas, making this division both practical and effective.
- The time constraint can be overcome through effective time management for all team members. A well-coordinated and skilled team can minimize delays by assigning tasks to the members with the most relevant experience, ensuring the right people handle the right tasks.
- International collaboration provides valuable opportunities for early design testing, which can help identify and resolve potential issues or delays before it becomes critical. By working with partners from different regions, teams can share diverse insights and expertise, ensuring that the design process is more thorough and efficient.

7.2 Stakeholder mapping

Mapping and analysis of individuals, groups, and organisations that might affect or will be affected by the project.

- **Internal Stakeholders:** Mikael Ekström and Emil Persson.
- **Investors:** MDU has provided financial support for the project and invested in its success.
- **Partners:** UdeA and UTP working with MDU on the project.
- **Customers:** Mikael Ekström, the project stakeholder and client, will utilise the autonomous robot once it is completed.
- **Regulatory Bodies:** RoboCup and Swedish laws set the standards and rules which the project must comply with.
- **Suppliers:** MDU supplies the necessary materials and resources for the project.
- **Competitors:** Other universities working on RoboCup projects, developing related technologies and solutions.

8 Planning

The project plan is presented in this section. It will provide an overview of the work in the project, including the timeline and timeframe for all the parts of the project.

8.1 Milestone plan

A milestone plan provides an overview of the projects most important milestones [4]. See Fig. 4 in Appendix B for the milestone plan of the project.

8.2 Work breakdown structure

A Work Breakdown Structure (WBS) gives a hierarchical view of the different work parts which are part of the projects scope [4]. The WBS for this project is shown in Table. 2.

ID	Work Class	Name (Work entailed)
-	Root	Multi-robot Soccer - RoboCup
1	Category	Software
1.1	Work Package	Simulate 12 robots
1.1.1	Deliverable	Setup grSim
1.1.1.1	Task	Download grSim
1.1.1.2	Task	Run the simulation correctly
1.1.2	Work Package	Simulation interface (API)
1.1.2.1	Deliverable	Integrate Protobuf
1.1.2.2	Deliverable	Move a single robot
1.1.2.3	Deliverable	Activate kicker on a single robot
1.1.2.4	Deliverable	Activate dribbler on a single robot
1.1.2.5	Deliverable	Execute any desired control over any chosen robot in grSim

1.2	Work Package	SSL interface (API)
1.2.1	Deliverable	Integrate ssl-vision
1.2.1.1	Task	Receive single package containing positional data
1.2.1.2	Task	Receive continuous positional data
1.2.2	Deliverable	Integrate ssl-game-controller (human referee and optional auto referee)
1.2.2.1	Task	Receive a single package containing referee commands/signals
1.2.2.2	Task	Receive continuous stream of referee commands/signals
1.2.2.3	Task	Integrate AutoReferee
1.2.3	Deliverable	Develop an automatic game controller (will be used when training the AI)
1.2.3.1	Task	Research and obtain proper understanding of SSL-rules
1.2.3.2	Task	Implement the automatic game controller system
1.3	Work Package	Hardware interface (API)
1.3.1	Deliverable	Basestation communication
1.3.1.1	Task	Configure basestation RF communication
1.3.2	Deliverable	Wheel motor interface
1.3.2.1	Task	Establish communication using UART
1.3.2.2	Task	Implement FOC
1.3.2.3	Task	Implement a velocity distributor (subscribing to cmd_vel)
1.3.3	Deliverable	Kicker interface
1.3.3.1	Task	GPIO implement activation
1.3.4	Deliverable	Dribbler interface
1.3.4.1	Task	Implement PWM control
1.3.5	Work Package	Sensor interface
1.3.5.1	Deliverable	IMU
1.3.5.1.1	Task	SPI implementation
1.3.5.1.2	Task	I^2C implementation
1.3.5.1.3	Task	Communicate using UART
1.3.5.2	Deliverable	Wheel encoders
1.3.5.2.1	Task	Configure wheel encoders
1.3.5.3	Deliverable	Camera
1.3.5.4	Deliverable	LIDAR
1.3.5.4.1	Task	I^2C implementation
1.3.5.4.2	Task	Communicate using UART
1.3.5.5	Deliverable	RGB sensor
1.3.5.5.1	Task	I^2C implementation
1.3.5.5.2	Task	Communicate using UART
1.3.6	Deliverable	Provide robot status information
1.3.6.1	Task	CPU temperature
1.3.6.2	Task	Battery charge
1.3.7	Deliverable	Establish communication between Raspberry Pi and Nucleo 144

1.3.7.2	Task	Install micro-ROS on Raspberry Pi and Nucleo 144 (UART)
1.3.7.3	Task	Implement ROS functionality
1.3.8	Deliverable	Establish communication between Nucleo 144 and motor drivers
1.3.8.1	Task	Communicating using UART
1.4	Work Package	Communication protocol (API)
1.4.1	Deliverable	Establish communication between PC and Raspberry Pi
1.4.1.1	Task	Using Wi-Fi
1.4.2	Deliverable	Integrate Protobuf
1.4.3	Deliverable	Utilizing ROS2
1.4.4	Deliverable	Ability to switch between two carrier frequencies
1.5	Work Package	Individual robot behaviour (finding the best way to do the things it has been commanded to do)
1.5.1	Deliverable	Research viable path planning algorithms
1.5.2	Deliverable	Integrate ROS2
1.5.2.1	Task	Getting sensor data
1.5.2.2	Task	Getting odometry data
1.5.3	Deliverable	Develop path planning algorithm
1.5.3.1	Task	Build using nav2
1.5.4	Deliverable	Implement robot ability to shoot
1.5.4.1	Task	Angle correctly towards target
1.5.4.2	Task	Shoot towards target
1.5.5	Deliverable	nav2 stack configuration
1.5.5.1	Task	Create nav2 parameter files
1.5.5.2	Task	Create nav2 launch files
1.5.6	Work Package	Develop supporting functions
1.5.6.1	Deliverable	Run Simulation/Hardware interface
1.5.6.1.1	Task	Run and call on Simulation interface
1.5.6.1.2	Task	Run and call on Hardware interface
1.5.6.2	Deliverable	Ability to send data
1.5.6.2.1	Task	Sending robot status (CPU temperature and battery charge)
1.5.6.2.2	Task	Sending acknowledgements that commanded task has been completed
1.5.6.3	Deliverable	Ability to receive data
1.5.6.3.1	Task	Ability to receive initialization data (ID, home playing side, starting pose)
1.5.6.3.2	Task	Ability to receive control commands from collective robot behaviour
1.5.6.3.3	Task	Ability to receive pose correction data from collective robot behaviour
1.5.6.4	Deliverable	Initialize the robot
1.5.6.4.1	Task	Getting robot ID
1.5.6.4.2	Task	Obtaining information about which playing side of the field is friendly
1.5.6.4.3	Task	Getting the start pose
1.6	Work Package	Collective robot behaviour/ Centralised strategy planner (Handles what each individual robot should do to obtain a win for the team)

1.6.1	Deliverable	Research AI-algorithms
1.6.1.1	Task	Outline actions
1.6.1.2	Task	Outline world representation (input)
1.6.2	Deliverable	Develop AI for strategy planning
1.6.2.1	Task	Develop the AI-algorithm
1.6.2.2	Task	Train the AI-algorithm
1.6.3	Deliverable	Be able to send data
1.6.3.1	Task	Provide initialization data to all robots
1.6.3.2	Task	Be able to send control commands to robots
1.6.3.3	Task	Be able to send pose correction data to robots
1.6.4	Deliverable	Be able to receive data
1.6.4.1	Task	Be able to receive robot status information
1.6.4.2	Task	Be able to receive referee commands and signals from SSL interface
1.6.4.3	Task	Be able to receive positional data from SSL interface
1.6.5	Deliverable	Ability to switch playing half on demand
2	Category	Hardware
2.1	Work Package	Base design
2.1.1	Deliverable	Hardware design to meet SSL-rules
2.1.1.1	Task	Research for chassis design
2.1.1.2	Task	Chassis BOM
2.1.1.3	Task	Powertrain BOM
2.1.1.4	Task	Sensor BOM
2.1.1.5	Task	Kicker BOM
2.1.2	Deliverable	Design modular components
2.1.2.1	Task	Allow for battery swap
2.1.2.2	Task	Allow for motor swap
2.1.3	Work Package	3D-design
2.1.3.1	Deliverable	Chassis design
2.1.3.2	Deliverable	Wheel design
2.1.3.2.1	Task	Wheel hub design
2.1.3.2.1	Task	Subwheel design
2.1.3.3	Deliverable	Dribbler design
2.1.3.4	Deliverable	Kicker assembly design
2.1.3.5	Deliverable	Mounting design
2.1.3.5.1	Task	Encoder mount
2.1.3.5.2	Task	Motor mount
2.1.3.5.3	Task	Battery mount
2.1.3.5.4	Task	Dribbler mount
2.1.4	Work Package	Circuit design and integration
2.1.4.1	Deliverable	Design powertrain circuit

2.1.4.2	Deliverable	Design kicker circuit
2.1.4.3	Deliverable	Design dribbler circuit
2.1.4.4	Deliverable	Design microcontroller circuit
2.1.4.5	Deliverable	Design IC for sensors
2.1.4.5.1	Task	Design IMU IC
2.1.4.5.2	Task	Design wheel encoder IC
2.1.4.5.3	Task	Design break beam IC
2.1.4.6	Deliverable	Integrate the circuits
2.1.4.7	Deliverable	Design mainboard PCB
2.1.4.8	Deliverable	Design carrier PCB for motor driver
2.1.4.9	Deliverable	Design kicker PCB
2.1.4.10	Deliverable	Design basestation PCB
2.2	Work Package	Manufacturing
2.2.1	Deliverable	Manufacture PCB
2.2.1.1	Task	Mainboard
2.2.1.2	Task	Basestation
2.2.1.3	Task	Kicker board
2.2.1.4	Task	Motor driver carrier board
2.2.2	Deliverable	Manufacture 3D-designed items
2.2.2.1	Task	Manufacture chassis
2.2.2.2	Task	Manufacture wheels
2.2.2.3	Task	Manufacture dribbler
2.2.2.4	Task	Manufacture kicker assembly
2.2.2.5	Task	Manufacture mountings
2.3	Work Package	Testing
2.3.1	Deliverable	Test motors
2.3.2	Deliverable	Test kicker
2.3.3	Deliverable	Test dribbler
2.3.4	Deliverable	Test sensors
2.3.4.1	Task	Test IMU
2.3.4.2	Task	Test wheel encoders
2.3.4.3	Task	Test camera
2.3.4.4	Task	Test LIDAR
2.3.4.5	Task	Test RGB sensor
2.3.5	Deliverable	Validate the integrated circuits
2.3.5.1	Task	Battery works correctly
2.3.5.2	Task	SSL-rule compliance
3	Category	HIL
3.1	Work Package	Develop HIL
3.1.1	Deliverable	Design a system to integrate hardware in simulation

3.1.2	Deliverable	Test the HIL in simulation
3.1.3	Deliverable	Ensure a smooth transition from simulation to physical robot hardware
3.2	Work Package	Validate hardware platform
3.2.1	Deliverable	Conduct HIL test to validate the performance

Table 2: The WBS for the project.

8.3 Schedule

The software schedule for the project can be seen in Fig. 1 and hardware schedule in Fig. 2. These provide the timeline for the project.

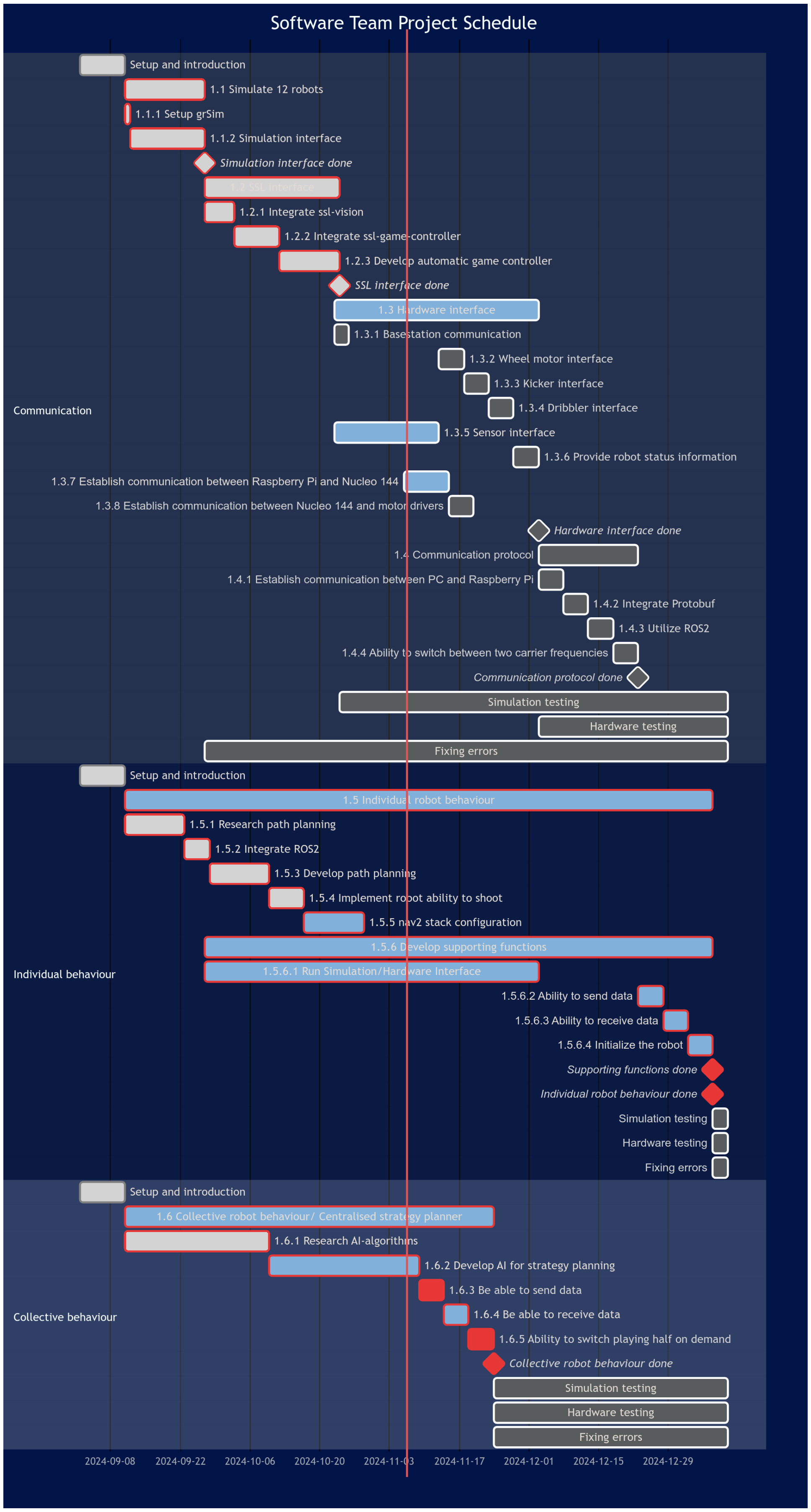


Figure 1: Software team schedule, some tasks and deliverables are excluded to reduce clutter and improve readability.

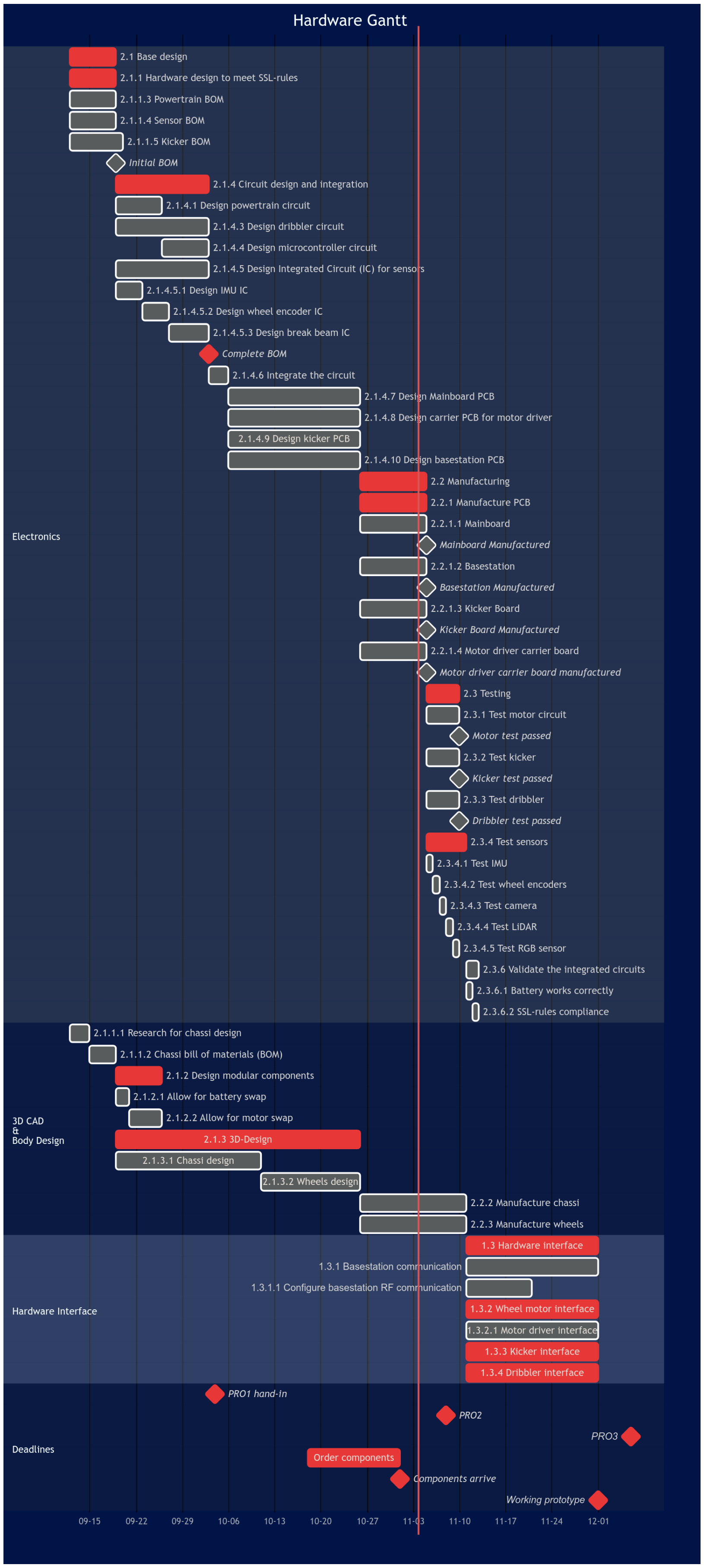


Figure 2: Hardware team schedule.

9 Staffing

The staffing plan, in terms of the staff and their roles, were arranged by external sources (sponsor), however tasks were assigned internally based on qualification, collaboration and each team members wishes. The roles and tasks of each project member can be seen in Table. 1.

9.1 Roles, responsibilities and authorities

Three authoritative roles with their own responsibilities were assigned (by the sponsor):

- Team Leader: Responsible for guiding and coordinating the group to achieve a common goal. A team leader is also responsible for making plans, assigning tasks, ensuring team collaboration, solving conflict, motivating the group and making key decision. One of the main responsibilities of a team leader is to communicate between the team and stakeholders, ensuring that the development process follows the broader project vision.
- Software lead: Organises the software team to ensure work continues smoothly and that the software requirements are met. This includes scheduling, overseeing the software teams progress, planning the software layout and thorough inspection of standards and requirements.
- Hardware lead: Manages the hardware team by overseeing the tasks, provide planning and making sure all requirements are met.

9.2 Staffing plan

The staffing plan is described in Table. 1. It describes every contributor and their task group.

10 Project budget

The budget for this project is decided by MDU. All of the passive electrical components such as capacitors, resistor and pin headers are sponsored by Würth electronics. Components such as Brushless Direct Current (BLDC) motors, MCU, sensors and additional hardware for the chassi are purchased with our budget. Due to the collaboration with UdeA and their Bill of Materials (BOM) additional sensors has been included in the budget.
Sponsors for Printed Circuit Board (PCB) manufacturing are still being considered. If no sponsors are found the cost of manufacturing will not be included in the budget due to cheap and reliable manufacturing companies do not exist. An estimated project budget is shown in Table. 3. For a more complete view of the project budget see the preliminary BOM in Table. A available in Appendix A.

Internal costs	External costs	Other costs	Summary
0	20000	0	20000

Table 3: Estimated project budget, all in SEK.

11 Development Plan

This development plan details the development process for the project, and contains important project information. It is crucial to be familiar with all of its details to be able to contribute to the project.

11.1 Development Methodology

The project does not follow any widely recognized development methodology, instead work commences according to an informal methodology. The methodology consists of dividing team members into groups (see Table. 1), afterwards the team members follow the schedule (see Section 8.3) for the current task which has been assigned to them until it is complete. This is reminiscent of the waterfall model. Communication and meetings occur whenever any project member deems it necessary. A minimum of one meeting a week is held to ensure that work is going well and that team members are working on what they should.

11.2 Team Structure and Roles

The team structure and roles are covered in Staffing (see section 9) and in Table. 1. The team structure and roles were decided by the sponsor.

11.3 Tools, Technologies, and Systems

The Operating System (OS) that are going to be used are shown in Table. 4.

Operating system	Version/Build	Purpose
Ubuntu	22.04 LTS	Used by software developers
Kubuntu	22.04.05	Used by a software developer
Arch Linux	6.11.1-arch1-1	Used by a hardware developer
Microsoft Windows	22H2 (OS Build 19045.4894)	Used by hardware developers

Table 4: The OS which will be used in the project.

Table. 5 describes the software development tools which will be used in the project.

Tool/Software	Version/Build	Purpose
Vim	8.2 and 8.2.2	Writing code
Neovim	0.10.0 and 0.10.1	Writing code
Sublime Text	4180	Writing code
Visual Studio Code	1.93.0 and 1.93.1	Writing code
CLion	2024.2.2	Writing code
CMake	20	Build tool
Git	2.34.1	Version control
GitHub	N/A (Online tool)	Online Git service
Git Bash	2,46,0,windows,1	Replacement for cmd.exe

Table 5: The software development tools which will be used in the project.

Code is to be written in the programming languages listed in Table. 6.

Programming language	Standard	Purpose
C++	20	Writing of all source code

Table 6: The programming languages which are to be used in the project.

The project will leverage the libraries, middleware and formats listed in Table. 7.

Tool/Software	Version/Build	Purpose
grSim	2.2	Simulation environment
ssl-vision	1.0	Providing positional data from SSL-RoboCup cameras.
ssl-game-controller	3.12.3	Acts as the referee and provides the game state
AutoReferee	1.4.1	Auto referee software
ROS2	Humble	Robot control and communication
Nav2	1.3.2	Path planning
Protobuf	3.12.4	Data format for communication
FreeRTOS	Undecided	Reliability and threading
GoogleTest	1.11.0-3	Software testing

Table 7: Libraries, middleware and formats which will be leveraged in the project.

The project will utilise the hardware tools listed in Table. 8.

Tool/Software	Version/Build	Purpose
OnShape	N/A (Online tool)	3D-CAD modelling
KiCAD	8.0.5	Electrical schematics design
3D printer	N/A	Print chassi, wheels, motor mounts, kicker mounts, dribbler mount
Soldering station	N/A	Solder the components on PCB
Lab bench power supply	N/A	Testing the system without a battery

Table 8: The hardware tools which will be used in the project.

The preliminary hardware list can be seen in Table. A, available in Appendix A.
The project management tools which will be used in the the project are shown in Table. 9.

Tool/Software	Version/Build	Purpose
Mermaid	N/A (Online tool)	Create gantt charts and block diagrams
EdrawMax	N/A (Online tool)	Create the WBS
draw.io	N/A (Online tool)	Create the milestone plan
Microsoft Excel	Uncertain	Create of BOM
LibreOffice	7.3.7.2	Spreadsheet editing
VisiData	2.2.1	Spreadsheet editing
Zotero	7.0.7	Reference management

Table 9: The management tools which will be used in the project.

The communication tools in Table. 10 will be used in this project.

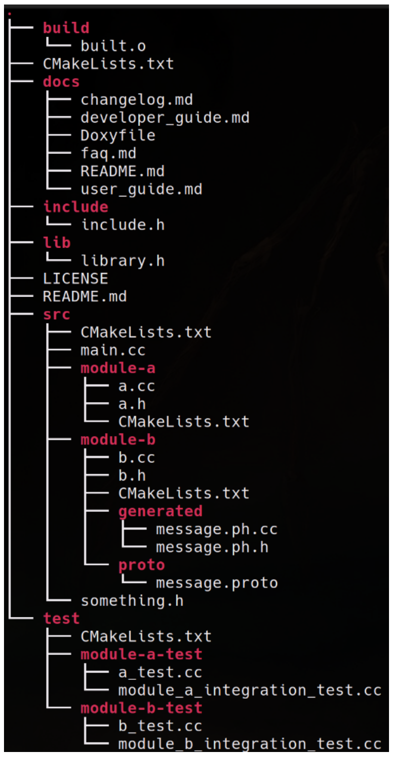
Tool/Software	Version/Build	Purpose
Discord	Windows (10), Linux (0.0.71) and Online	Team communication
Outlook	N/A (Online tool)	Communication with supervisors, stakeholders, teachers and retailers
Microsoft Teams	N/A (Online tool)	Communication with collaborators (UdeA and UTP)

Table 10: The communication tools which will be used in this project.

11.4 Coding Standards and Guidelines

The project will strictly adhere to the Google C++ Style Guide coding standard and the file structure described in Fig. 3, also available on the projects GitHub repository project-structure[1]. This was done to ensure uniformity, facilitate future development and allow for easy contribution.

[1] https://github.com/DVA490-474-Project-Course/project-structure



```
.
├── build
│   └── built.o
├── CMakeLists.txt
├── docs
│   ├── changelog.md
│   ├── developer_guide.md
│   ├── Doxyfile
│   ├── faq.md
│   ├── README.md
│   └── user_guide.md
├── include
│   └── include.h
├── lib
│   └── library.h
├── LICENSE
├── README.md
├── src
│   ├── CMakeLists.txt
│   ├── main.cc
│   ├── module-a
│   │   ├── a.cc
│   │   ├── a.h
│   │   └── CMakeLists.txt
│   ├── module-b
│   │   ├── b.cc
│   │   ├── b.h
│   │   ├── CMakeLists.txt
│   │   ├── generated
│   │   │   ├── message.ph.cc
│   │   │   └── message.ph.h
│   │   └── proto
│   │       └── message.proto
│   └── something.h
└── test
    ├── CMakeLists.txt
    ├── module-a-test
    │   ├── a_test.cc
    │   └── module_a_integration_test.cc
    └── module-b-test
        ├── b_test.cc
        └── module_b_integration_test.cc
```

Figure 3: The software file structure which will be followed in this project. Note that the file names and contents of the directories are just placeholders to give a visual representation of the structure.

11.5 Version Control

The project will use the version control tool Git to manage changes, versions, history and documentation of all files in the project. The projects GitHub[2] page will contain all the project material and is public and available to everyone.

11.6 Testing and Quality Assurance

Software testing and quality assurance will be done using GoogleTest. Unit tests will be created for every developed function. Integration tests will additionally be created for every module and sub module. All tests are to be run before committing a new feature or change to GitHub.
Standardised tests for the hardware will be used to calibrate sensors and verify that the sensors behave according to the specifications in the data sheet. These tests should be done every time the robot is used.

11.7 Integration and Deployment

The integration of the system will follow a structured approach, combining both software and hardware modules. Each software module will be developed separately and will be integrated together once it is ready and has passed its individual unit tests for all the functions it contains. Software modules will first be integrated together with other software modules and tested using integration tests. Afterwards, the hardware components will be integrated in the tests, such as the sensors, motors, and kickers. The hardware components will be integrated one at a time with a focus on ensuring accurate output and functionality when combined with the software.
It is important to always test and verify the functionality of the modules during the integration phase. Verification will be done by performing unit, integration, and Hardware-in-the-Loop (HIL) tests. The software modules will be tested in a simulation environment using grSim, while HIL tests will simulate real-world conditions by integrating hardware components.
Once the functionality of the modules has been verified and all tests have passed, the deployment phase begins. At this point full deployment tests will be done, which consist of testing the entire system in deployment settings ensuring the functionality of the complete system.

12 Communication plan

The communication plan was established to guarantee that the correct target audience receives the necessary information at the right time and through appropriate channels. The communication plan can be seen in Table. 11.

[2]https://github.com/DVA490-474-Project-Course

Target audience	**Purpose**	**Type of information**	**Timing**	**Communication channels**	**Responsible**
Project members	Everything	Everything	Any time	Discord	Project members
Project members	Keep the team updated	Project status & information	Weekly	Meeting	Team Leader
Stakeholders	Inform stakeholders of status of project, supervision & aid	Project progress summary, project plan, on leave, feedback and supervising	Weekly	Meeting	Team Leader
Colombia contact person	Collaboration	Task assignment, progress report	Weekly	Microsoft teams and mail	Team Leader
Panama contact person	Collaboration	Task assignment, progress report	Weekly	Microsoft teams and mail	Team Leader

Table 11: The communication plan for the project.

13 Risk analysis and response planning

A thorough risk analysis was done and a response plan was created to ensure that any type of damage to the project is planned for and can be dealt with appropriately. A table was created with the following attributes:

- Potential risk: short description of a potential risk. (Only those which were deemed to have a notable impact will be covered).
- Probability: score from 1-5, with 1 being very unlikely, and 5 being very likely.
- Impact: score from 1-5, with 1 having a very small impact, and 5 having a very large impact.
- Risk score: calculated according to: Probability $\times$ Impact. Score from 1-25, with 1 being a low priority risk, and 25 being a critical priority risk.
- Strategy: which strategy is to be used to deal with the risk. There are four possible: accept, transfer, ignore, mitigate.
- Mitigation action: what the mitigation action entails.

The table can be seen in Table. 12.

Potential risk	**Probability**	**Impact**	**Risk score**	**Strategy**	**Mitigation action**
Simulation differing substantially from reality	4	3	12	Accept	N/A
Reversed polarity when plugging in the battery	3	5	15	Mitigate	Reverse polarity protection circuit
Misunderstanding between hardware and software input/output values	5	5	25	Mitigate	Communicate between the hardware and software teams, and thorough design files
Ordered components not arriving on time	3	4	12	Accept	N/A
Components breaking/burning up	4	5	20	Mitigate	Order redundant components and over-dimension so that the components maximum voltage level far exceeds the expected voltage.
Robot excessively shaking	2	2	4	Mitigate	Increasing the amount of mini wheels on the wheels
Dribbler bar breaking	2	4	8	Mitigate	Use high quality material

Dribbler unable to induce enough rotation on the ball	2	3	6	Mitigate	Powerful motor and high grip material
Ball bouncing away instead of being caught in the rotation of the dribbler	4	3	12	Mitigate	Design dribbler assembly in a way that reduces bouncing
Electrocution injury	2	4	8	Mitigate	Take proper precautions, use protective gear, and rethink the action before performing it
Hardware components not being equal to those used by our collaborators	5	2	10	Mitigate	Have a continuous dialog with our collaborators as to minimise misunderstandings
Miscommunication with collaborators	5	3	15	Mitigate	Use clear communication channels, define roles and responsibilities, document key decisions and have regular check-ins.
Project duration is not enough time to complete the original plan	3	5	15	Mitigate	Plan reasonably, continuously follow up on progress made, and set realistic goals
Team members are unwell (sick)	5	2	10	Accept	N/A
Misunderstanding between team members	5	5	25	Mitigate	Use clear communication channels, define roles and responsibilities, document key decisions and have regular check-ups.
Power supply or battery failure	3	4	12	Mitigate	Implement redundant power supplies and use a reliable Battery Management System (BMS). Test battery performance under load to identify potential failures early.
Software bugs causing unexpected robot behaviour	5	5	25	Mitigate	Implement thorough software testing and continuous integration, including unit tests, HIL testing, and simulation.
Mechanical misalignment or interference between components	3	4	12	Mitigate	Use precise manufacturing techniques and perform tolerance checks during assembly. Maintain a detailed CAD model to predict issues.
Network connectivity issues between robots and central computer	3	4	12	Mitigate	Implement a robust communication protocol with redundancy and conduct extensive testing in environments with potential interference.

Difficulty in achieving desired response time due to computational limitations	4	4	16	Mitigate	Optimise code for performance, offload computationally heavy tasks to specialised hardware, and monitor performance metrics.
Software-hardware integration issues due to different update cycles	4	4	16	Mitigate	Maintain version control, design files ensuring all subsystems are compatible, and having regular integration meetings for synchronisation.
Changes in SSL rules during product development	2	5	10	Mitigate	Monitor rule updates, keep designs flexible, and allocate resources for rapid changes when required.
Exceeding the budget	1	5	5	Mitigate	Thorough budget monitoring
Affordable components lacking the performance required to achieve the goal or meet the requirements	4	5	20	Accept	N/A

Table 12: The risks, their probability, impact, risk score and the planned responses.

14 Documentation plan

Good documentation is a critical tool to help project management since it allows for reviewing the work done and observing the progress. However it also serves other very important purposes like transparency, accountability and helps others understand, use and contribute to the project. To this end, the following documentation plan was established for this project.

14.1 What to Document

The following items will be documented:

- Requirements: available in this document, see Requirement section 6.
- Specifications: available in this document, see Tools, Technologies, and Systems section 11.3.
- Design: available on the projects GitHub.
- Changes:
  - Software: available from Git commit messages and in changelog file in each repositories docs directory on the projects GitHub.
  - Hardware: each commit to the repository contains information about changes made.
- Issues/Known bugs:
  - Software: available from Git commit messages and in changelog file in each repositories docs directory on the projects GitHub.
  - Hardware: issues can be found in the issues tab for the repository.
- Testing:
  - Software: record of all test and simulation results available in each repositories wiki on the projects GitHub.
  - Hardware: data from standardised tests will be documented in the wiki.
- User manual:
  - Software: available in user_guide file in each repositories docs directory on the projects GitHub.
  - Hardware: the repositories wiki contains the user manual.

The software team will have the following documentation:

- Changelog: record major changes, new features, bugs, bug fixes and removed features in each release version.
- API: document the functions, classes and modules of the project.
- User Guide: installation instructions, how to run and use the application, configuration options.
- Developer guide: describe how to set up the project for development, code style, build instructions.
- README: a multitude of different README files will exist with different documentation purposes.
- Code comments: improve understanding of the source code.
- Git commit comments: log changes and track the status of the code and project.
- GitHub wiki:
  - Test results: create a record of the test results.
  - Simulation results: create a record of simulation results.

The hardware team will have the following documentation:

- Changelog: design changes, wiki updates, new features, bugs, bug fixes.
- Setup: A guide for installing the required dependencies, required hardware and how to configure it.
- Component database: The wiki contains information about each component mentioned in the BOM.

14.2 How to Document

All documentation is to be in British English and must follow scientific writing standards. Software documentation is additionally gonna use Doxygen, all header files are to have comments in accordance with the Doxygen tool.

14.3 When to Document

It is paramount that documentation takes place the moment any work has been done or any changes have been made. For complete clarity:

- Requirements: documentation of any changes or additions to the requirements should be done instantly.
- Specifications: if any specifications are subject to change then the accompanying documentation must instantly be altered.
- Design: whenever there is an alteration of the design, the corresponding design files and documentation must be updated accordingly as soon as possible.
- Changes: all changes must be logged.
- Issues/Known bugs: when there is an update to an existing issue or bug, or a new one is found, then these must instantly be documented.
- Testing: any changes to existing tests, addition of new tests and test results must instantly be documented.
- User manual: the user manual must only be documented in time for the final release and whenever any post release updates are done.

14.4 Who is Responsible

Team leads are responsible for overseeing their respective teams documentation as well as ensuring all the proper documentation is available and in proper order. Developers are responsible for producing proper documentation for everything the developer has done.

14.5 Where to Store Documents

All software and hardware documentation will be stored on the projects GitHub[3] page. This way of storing the documentation will ensure it is secure and accessible by everyone.

14.6 Document Review Process

The software and hardware lead will continuously review their respective teams documentation. The developers are required to review the documentation written by said developer as well.

[3]https://github.com/DVA490-474-Project-Course

14.7 Training

Members were introduced to the documentation plan during the initial project period. New members will be introduced to the documentation plan by their respective team leader at the earliest possible moment.

15 Testing Plan

Testing is required to ensure the product works as intended, for this reason the following testing plan was established. The plan should help onboard new developers and clearly show the testing practices employed in the project, helping reveal possible flaws and oversights.

15.1 Testing Methodology

The software team will make extensive use of unit and integration tests. There must be unit tests for every function ensuring it works properly. Additionally, integration tests must exist which test every modules proper workings when integrated with all its subcomponents. This verification methodology was chosen because it was expected to work the best with the teams experience and resources. The hardware team will create tests, which ensure the hardware components perform according to the specifications in their respective data sheet, the data sheets can be found on the projects github. Baseline tests will be created for the sensors which can be found on the projects wiki page on github. Response time and power consumption are example of performance indicators for sensors that will be used as baseline to ensure our sensor work as expected. The tests are designed to evaluate if the product meets the established requirements (see Requirement section 6).
Validation will be done by constant interaction with the stakeholder and client, informing them of the current state and direction of the project. Their feedback will then ensure the correct product is being developed, and that it does not turn into something else besides the desired product and keeps to the goal of the project.

15.2 Testing Team Structure and Roles

The team size for the project does not allow for a dedicated testing team. However software developers are responsible for creating and running unit tests for the code which the developer has developed, as well as integration tests for all modules and sub modules. The developer must additionally document the test results as described in Documentation plan (see section 14). The software lead is responsible for overseeing and ensuring the proper tests and documentation of the test results are available.

15.3 Testing Tools, Technologies, and Systems

The testing tools which will be used are:

- GoogleTest: GoogleTest will be used for unit and integration testing.
- Git: Version control tool.
- Github: Online Git service.

15.4 Testing Standards and Guidelines

The project follows no specific testing standards or guidelines. There must be unit tests for every function and integration tests for every module, but nothing dictates how these tests should be written or structured. However, it must not violate the coding standards followed by the project (see Section 11.4).

15.5 Version Control for Test Artifacts

This project will use the version control tool Git, see Version Control section 11.5. The online Git service GitHub will be used in this project.

15.6 Bug Reporting and Tracking

Bug reporting and documentation is kept track of using the changelog file in each repositories docs directory. Software developers are responsible for documenting the status of any bugs encountered.

15.7 Test Schedule

All tests are to be run before committing an update or change to GitHub. This includes unit tests, integration tests and all other relevant tests.

15.8 Integration and Regression Testing

Integration tests are to be done for each module and sub module. The project will run all unit and integration tests each time a new feature is added or changes are made to any code, as well as before doing a Git commit.

16 Handover plan

The handover plan outlines all necessary steps for successfully transferring the project deliverables to the stakeholders. It will include all project documentation, code and assets, ensuring the ability for maintenance and further development.

16.1 Handover Methodology

The projects handover methodology includes the following:

- Presentation: prior to the handover, a presentation will be held. This will help familiarise the client with the project and the results obtained. This presentation should ease the integration of new staff to the project but the documentation alone should be a sufficient resource for them to learn how to use the product after the handover process is complete.
- Final testing: a complete testing session where every single test will be run and all the results are logged. This is critical for a transparent description of the state of the product at the time of the handover.
- Final documentation review: a review of all the documentation, ensuring everything is available, accessible and in order. The documentation is a vital source of information, enabling for future development and helping users use the product.
- Hardware inventory check: a final comprehensive hardware inventory check will take place. Ensuring an accurate description of the final product and the process of creating it.
- Software check: a final check of the correctness and availability of all the developed code and software material on the projects GitHub. Ensuring further development can proceed smoothly and allowing for easy on boarding of new contributors with all the material which might be required gathered on the GitHub.

16.2 Handover Team Structure and Roles

The leading roles will be in charge of the handover process and will have the following responsibilities:

- **Team Leader:** Responsible for overseeing the entire handover process, ensuring that all necessary materials are included and that the product is delivered before the deadline.
- **Software Lead:** Responsible for GitHub and all software related material, including documentation and proper delivery of all the material.
- **Hardware Lead:** Responsible for all hardware assets, designs and documentation, and their delivery.

16.3 Handover Tools, Technologies, and Systems

The tools which will be used during the handover process includes:

- Canvas: Canvas will be used to deliver the project report.
- GitHub: GitHub will be used to deliver all code, documentation and all other project files.
- Overleaf: Overleaf will be used to conduct presentations and the writing of all papers.

16.4 Handover Schedule

The handover schedule consists of an initial handover on January 10th 2025. There is the possibility for a followup handover on February 14th 2025 if the product from the initial handover was found unsatisfactory.

16.5 Documentation

Comprehensive documentation regarding all the project information including user guides, design, changelog, Application Programming Interface (API), test and simulation results can all be accessed on the projects GitHub[4] page. The GitHub page is public and available to everyone.

16.6 Presentation and Demonstration

Presentations and demonstrations will be provided prior to the final hand-over to inform the sponsor of the current state of the project, as well as the results obtained. The preliminary presentation date is December 12th 2024.

16.7 Final Sign-off

A final sign-off will occur once the project sponsor has reviewed the delivery, and ensured that all project requirements and goals are met. The final sign-off will act as a formal acceptance where the stakeholders confirm their satisfaction with the project outcomes. This will mark the official conclusion of the project, after which the team will be relieved of all responsibilities concerning the product and the project.

[4]https://github.com/DVA490-474-Project-Course

16.8 Follow-up and Feedback

Any feedback is greatly appreciated and will be addressed during "Hand-in after deadline and revision" if the product is not found satisfactory. This will ensure that a satisfactory product is delivered and that all concerns are put to rest.

17 Individual contributions

The individual contributions of each team member to this project plan is outlined in Table. 13. The lead roles were assigned the main responsibility for the project plan as to not distract the developers from their assigned tasks.

Team member	**Contribution to project plan**
Viktor Eriksson	Review
Anton Grusell	Review and sections 6.1,13
Mudar Ibrahim	Review and sections 7,8,9.1,11.7,16
Jacob Johanssson	Review
Aaiza Aziz Khan	Review and sections 13
Carl Larsson	Review, sections 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17 and Appendix A,B
Johanna Melander	Review and sections 6.1,13
Shruthi Puthiya Kunnon	Review and sections 13
Pontus Svensson	Review and sections 1,6,8,9.1,10,11.3,14,15,16.2 and Appendix A
Fredrik Westerbom	Review, sections 1,6.1,10,11.3,13 and Appendix A
Emil Åberg	Review and sections 13

Table 13: Each team members contribution to this project plan.

References

[1] Small Size League | RoboCup Soccer – Information for the administration of the RoboCup Soccer Small Size League. URL `https://ssl.robocup.org/`.

[2] Mark Coeckelbergh. *Robot ethics*. The MIT Press essential knowledge series. The MIT Press, Cambridge, Massachettes, 2022. ISBN 978-0-262-37050-9 978-0-262-37049-3.

[3] Rules of the RoboCup Small Size League, May 2024. URL `https://robocup-ssl.github.io/ssl-rules/sslrules.html`.

[4] Bo Tonnquist. *Project management*. Sanoma Utbildning, Stockholm, fifth edition edition, 2021. ISBN 978-91-523-6079-8.

A Appendix: Bill of Materials

Table 14: Component Descriptions for the Project

Component	**Description**	**Purpose**	#	**á price (price per robot) SEK**
DF45L024048-A	Brushless direct current (BLDC) motor with integrated hall sensors for the wheels	Used to spin the wheels of the robot.	4	830.4 (3273.60)

Table 14: Component Descriptions for the Project

Component	**Description**	**Purpose**	#	**á price (price per robot) SEK**
Hobbywing FPV XRotor 3110 900KV	Brushelss DC motor	High revolutions per minute (RPM) motor used to control the dribbler.	1	175.20 (175.20)
B-G431B-ESC1	BLDC motor driver	Motor driver with embedded μController current sensing and hall sensing to form a closed-loop control algorithm	5	208.96 (1044.8)
NUCLEO-H723ZG	μController	Computational power and real-time processing capabilities, supports μROS	1	322.58 (322.58)
Raspberry Pi 4 Model B/8GB	Single-board computer	Processing camera input and performing local path planning	1	979 (979)
Raspberry Pi RTC	Real time clock	Sync the RPi with the MCU	1	47.61 (47.61)
TOF400C-VL53L1	Lidar	Chosen by UdeA for obstacle detection	2	176.32 (176.32)
VL6180 TOF	Lidar	Chosen by UdeA for obstacle detection	1	39.62 (39.62)
SEN0374	9DOF IMU	Determine the orientation of the robot	1	191.48
SX1280IMLTRT	Radio frequency (RF) transceiver	Used to transmit data over 2.4Ghz network	1	75.44 (75.44)
SKY66122-11	Integrated front-end-moduel (FEM)	Simplified integration with the RF circuit	1	40.48 (40.48)
6s 1300mAh -120C - GNB HV XT60	LiPo-battery	Used to power the robot	1	351.20 (351.20)
LT3750	Charging controller for the capacitors of the kicker	Charge controller for the kicker circuit	1	146.93 (146.93)
WSEN-ISDS 6 Axis IMU	6-DoF IMU	Will be used for odometry of the robot	10	N/A
Raspberry Pi Camera-module 3	Camera	Provide images in front of the robot to detect the ball and obstacles	1	369 (369)

Table 14: Component Descriptions for the Project

Component	**Description**	**Purpose**	#	**á price (price per robot) SEK**
JST 6B-PH-K-S	Connector	Hall sensor connector from the motor	4	3.85 (15.4)
JST B5P-VH	Connector	Motor connector	4	4.06 (16.24)
PHR-6	Connector	Hall sensor	8	1.07 (8.56)
VHR-5N	Conector	Motor connector	10	2.22 (22.2)
XT60PW-M	XT60 input connector	90 Degrees PCB mount XT60 connector for input power	2	11.21 (11.21)
Connectors	Passive component	Supplied by Würth	N/A	N/A
Shaft hub with clamping bracket 4mm	Coupler	Couple the wheels with the motor shaft	4	139 (556)
Bearings	Bearings	Make the roller spin (dribbler)	2	18 (36)
Resistors	Passive component	Supplied by Würth or 326	N/A	N/A
Capacitors	Passive component	Supplied by Würth or 326	N/A	N/A
Voltage regulators	DC/DC buck converters	Supplied by Würth	N/A	N/A
Solenoid	Solenoid	Supplied by MDU	1	N/A
PCB	Printed circuit board (PCB)	The students will supply any custom PCB designed	2	N/A
LM74500 -QDDFRQ1	Reverse polarity protection	Used to protect against wrong polarity connections	2	16.45 (32.9)
PSMN6R7-40MLDX	N-channel mosfet	Used with the LM74500 -QDDFRQ1	2	5.77 (11.54)
M3 16mm galvanized steel countersunk screw, 100 pack	Mounting screws	Screws for robot chassi	1	21.6 (21.6)
M3 galvanized steel locking washers, 200 pack	Washers	Chassi assembly	1	69.7 (69.7)
M3 steel locking washers, 200 pack	Locking washers	Chassi assembly	1	83.3 (83.3)
M3 16mm galvanized steel countersunk screw, 100 pack	Screws	Chassi assembly	1	21.6 (21.6)
M3 stainless steel nut, 100 pack	Screw nut	Chassi assembly	1	21.85 (21.85)

Table 14: Component Descriptions for the Project

Component	**Description**	**Purpose**	#	**á price (price per robot) SEK**
M3 threaded inserts 3mm, 50 pack	Threades inserts	Chassi assembly	1	95 (95)
Bearings with flange iglidur® M250	Bearings for mini-wheels	Wheel assembly	100	3.34 (334)
X-RING 3.63X2.62/NBR70	Fasten the mini-wheels	Wheel assembly	100	6.2 (620)
Wheel hub 4mm	Fasten wheels to motor	Wheel assembly	4	139 (556)
APDS-9960	RGB sensor	Ball detection	1	199(199)

B Appendix: Milestone plan

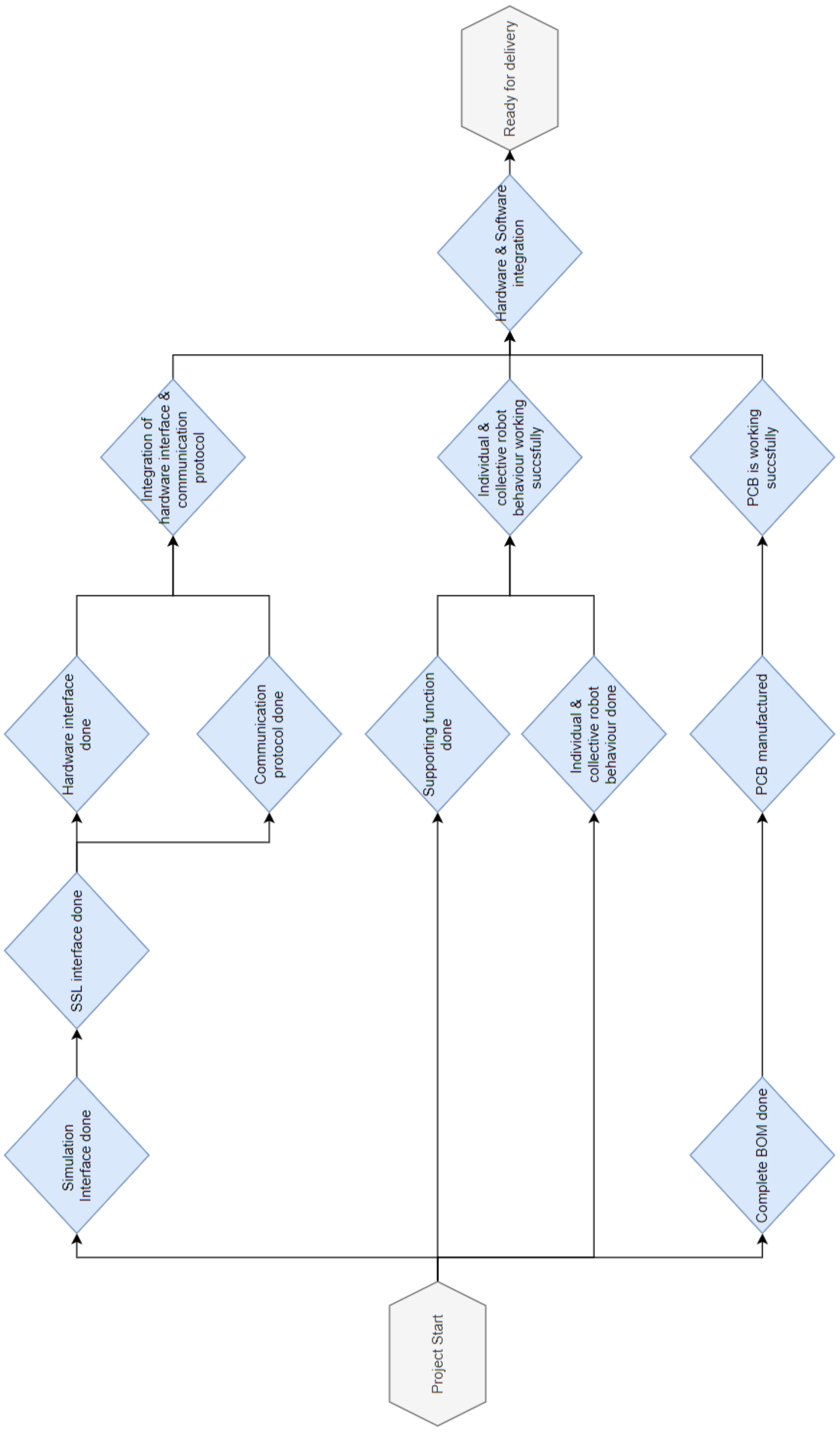

Project Start
Simulation Interface done
SSL interface done
Hardware interface done
Communication protocol done
Integration of hardware interface & communication protocol
Supporting function done
Individual & collective robot behaviour done
Individual & collective robot behaviour working succsfully
Complete BOM done
PCB manufactured
PCB is working succsfully
Hardware & Software integration
Ready for delivery

Figure 4: The milestone plan for the project.