SeeGoals Team Description Paper Robocup 2026 Small Size League

Vilhelm Bergqvist, Theodor Svensgård, Rasmus Wallin, Anton Östman, Aditya R Dharamshi, and Reza Saman Mahmoodi

FIA Robotics
Linköpings Universitet,
Kårallen, 581 83 Linköping, Sweden
seegoals@fiarobotics.se
https://www.fiarobotics.se/seegoals

Abstract. This paper is the first TDP by SeeGoals, which is a SSL project under the bigger robotics association FIA Robotics, Sweden. This paper presents the approach to our first robot design for the 2026 RoboCup South Korea. It explains the mechanical, electronics, software, and firmware concepts used.

1 Introduction

The SeeGoals team was founded in 2023 to compete in RoboCup SSL. The initial plan was to replicate TIGERs Manheim's 2020 open-source robot [10] to get robots up and running as quickly as possible and then focus on the software, which would be coded from scratch. This proved to be more difficult than originally intended. Two and a half years later, the robots are driving around the field, with most major parts completely redesigned from those TIGER origins. Their design philosophy lives on, along with some technical debt, where advanced parts are underutilized and overcomplicated and also driving up costs.

In 2025 SeeGoals participated in German Open. With fewer than six working robots competing was difficult. However, it was an excellent learning experience and a lot of hard work went down into the robots during the competition. Since then a lot of effort has been made to improve the design and ensure reliability. We are now ambitious to compete in the SSL Division B. Since this is our first TDP, it will mostly cover the current state of all sub-modules, the solutions developed, and our vision for the competition this summer. The paper is structured similarly to the team's internal management in four major parts: Mechanics, Electronics, Firmware, and Software.

2 Mechanical Overview

The SeeGoals robots rely heavily on 3D-printed components designed in-house in CAD. The initial design was heavily inspired by the open-source CAD files from TIGERs Mannheim, but over time, more and more components have been

redrawn from scratch. For most parts an iterative design philosophy were used, where parts are rapidly improved based on current requirements and learnings from evaluations. The robots currently consist of three layers:

- A bottom structure that holds the wheel motors, kicker solenoid, kicker PCB and dribbler assembly.
- A middle plate that holds the battery and gives space for wiring.
- The powerboard PCB with the Nucleo devboard and nRF transceiver PCB mounted on the top side, and motor drivers mounted on the bottom side.

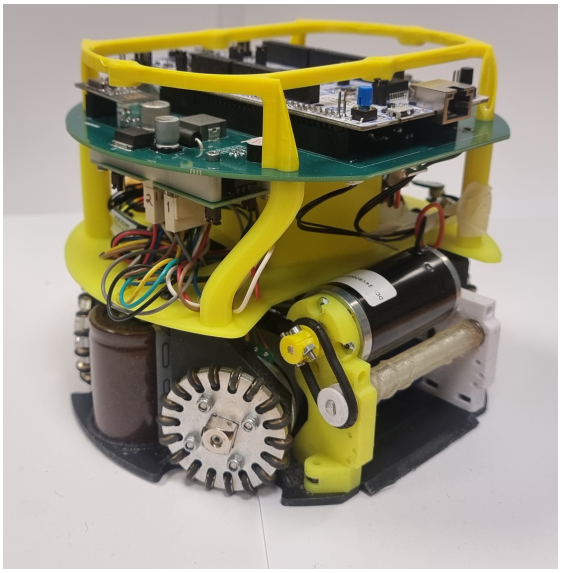

Fig. 1. An overview of the different mechanical systems of the robot.

The middle plate screws directly to the bottom structure while the powerboard PCB and the shell is attached with custom standoffs. A picture of a fully assembled robot, without the shell, is shown in Figure 1.

2.1 3D-printing as a Manufacturing Method for SSL Robots

Many teams use CNC-milled aluminium parts that are generally stronger than 3D-printed plastic, but 3D printing offers great possibilities for iterative design, which has been very helpful for us as a new team. It is also very expensive to mill large parts, thereby limiting the design. 3D printing generally offers greater geometric freedom allowing for strengthening where necessary to compensate for the material's lower strength. One practical example of this is the bottom structure, inspired by the TIGERs' design, which uses an aluminium plate with screwed-on

motor mounts also milled from a plate. In our design the plate and mounts are combined into a single part and several strengthening gussets are used to stiffen the plate and the assembly, see Figure 2. During evaluation we encountered an issue where one of the baseplates cracked when the robot drove into a wall at high speed. Since then we've added strengthening measures primarily with the gussets but also through optimised print settings. Running with five walls and 40% infill seems to be sturdy enough for this application. No baseplate or any printed part has broken since then. Another example of design freedom with 3D printing is the ability to embed captive steel nuts into printed parts. This feature has been used for mounting points at several locations on the robots.

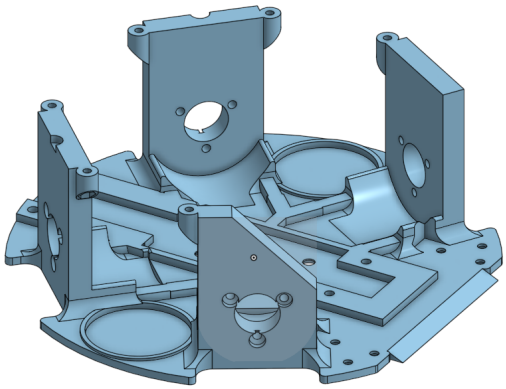

Fig. 2. Picture of the bottom plate of the robot, exported from CAD

We are hopeful about the performance of the 3D-printed parts but work will continue with stress testing and improving the robots for the competition. The team plans to upgrade some parts to metal, depending entirely on performance during tests and competitions.

The 3D-printed parts have all been produced on a BambuLab X1C printer equipped with an automatic material system (AMS), enabling multi-colour or multi-material printing. The current material choices are described in Table 1 below.

During our evaluations, if the FDM-printed parts prove too weak, the design might pivot to a more typical aluminium-based makeup. But another option is to order parts printed with selective laser sintering (SLS) from a third party. This would allow the team to continue using the 3D designs under iteration and testing, rather than starting from scratch.

2.2 Other Manufacturing Methods Used

Besides 3D-printing, the dribbler assembly uses a spinning aluminium shaft with an outer layer made of cast polyurethane resin. The casts were 3D-printed in

Table 1. Materials used with motivation

Bottom structure	PLA	Stiffness and strength
Middle plate	PLA	Stiffness and strength
Kicker PCB holder	PLA	Strong and easy to print
Power board standoffs	PLA	Strong and easy to print
Shell standoff and holder	PETG	Impact resistance and dampening
Powerboard PCB protective ring	TPU	Soft material to dampen impacts
Shell	PETG	Found to be the most appropriate material in [10]
Dribbler mounts	PETG	Impact resitance and flexibility
Dribbler Pulleys	PA12 Nylon (Addnorth Adura)	Durable and heat-resistance

two parts that encapsulate the aluminium shaft and feature pour holes along one side. The polyurethane resin used is the two-part resin 'PX60 Medium Flexible Polyurethane Resin' from Xencast. The geometry for the dribbler bar surface was inspired by Tiger's 2020 robot [10], but alternative designs from RoboTeam Twente [8] have also been considered. With some vaseline on the printed parts, the resin did not stick to the cast. Some bars have uneven casts where too much vaseline was used. A perfectly thin coating is preferred, but difficult to achieve.

Since most of the parts are bought ready-made or 3D-printed to a custom design there isn't much manual processing required. Some filing has been used to improve the fit between parts, and, of course, some work is needed just to assemble the robots. The kicker solenoids have been slightly modified to allow clearance for the wiring and the wheel motors.

2.3 Bought Components

Some of the mechanical and electromechanical components have been bought off the shelf, as listed below:

- Nanotech DF45L024048-A2 Brushless 3-phase DC motors, 24VDC, 6100RPM [2], combined with 50 mm omniwheels from GTFRobots [1], mounted directly to the motor shafts.
- Simple, cheap 24 V DC motors for the dribbler.
- 35 mm stroke solenoid rated for 12 V for the kicker solenoid.

3 Electronics Overview

Since the robot requires multiple electronic components it was good to test them individually with the microcontroller and later integrate them using PCBs. Since this is our first attempt it was decided to use a Nucleo H755ZI board[9], which handles all computations at the robot level and is mounted directly on the powerboard, rather than integrating the chip into our PCBs. This helped us

focus better on integrating the critical electronic components. All the electronic systems were initially integrated via wires and tested during the 2025 German Open. This helped us analyse the performance and make further improvements. Our 2026 design integrates the main powerboard, motor driver board, IR sensor board, and kicker board, reducing the number of wires used. Figure 3 below shows an overview of the electronic system.

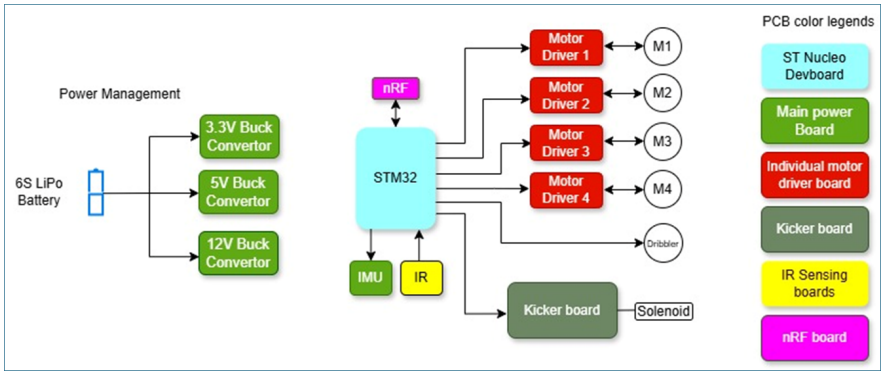

Fig. 3. Block diagram of the entire electronic system of the robot, highlighted by the colour of the PCB on which they are mounted.

3.1 Powerboard

This is the main interface board, as shown in Figure 4, which is almost as wide as our robot and acts as a plate, as previously described in the mechanical section. A 22.4V LiPo battery [7] is connected to the bottom of the board, powering three buck converters that generate 12V, 5V, and 3.3V supplies and power all the robot's electronic components.

The Nucleo board is mounted on the top side along with an nRF24 transceiver module board used for communication. An IMU LSM6DSL from STMicroelectronics [3] is mounted at the centre of this board, which provides the motion data used for calculations. Unidirectional dribbler control is achieved with an N-Mosfet and a flyback diode on the bottom side, connected to the dribbler motor cable via a 2.54 mm socket. PWM can be used to regulate speed as the current version of the dribbler has no encoders.

The kicker board is connected through a Molex picoblade cable and the motor drivers are directly connected through 2.54mm connectors. Based on the performance, some modifications will be made in the near future to ensure that the electronics are stable and more efficient.

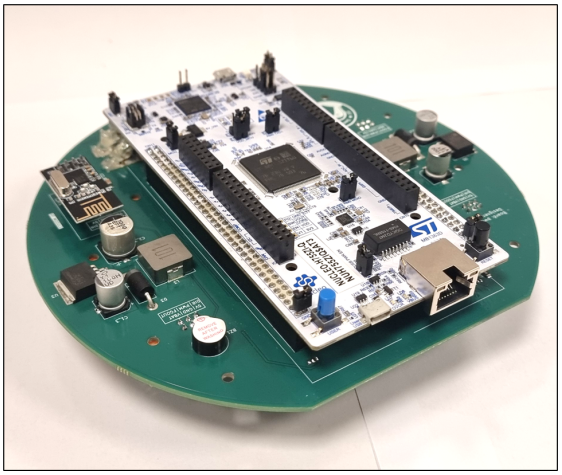

Fig. 4. Picture of the top-side of Powerboard with the STM Nucleo and nRF transceiver boards mounted.

3.2 Motor Driver Board

The motors are controlled through the MCT8316Z-Q1 chip from Texas Instruments [4]. As we are currently evaluating the performance of this chip and have observed some failures during testing we have decided to use a separate motor driver board, as shown in Figure 5, instead of integrating the chip into the main powerboard which can be easily replaced if needed.

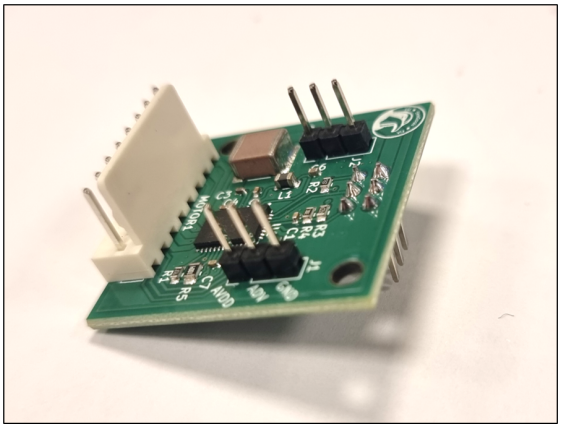

Fig. 5. Picture of the bottom side of the Motor Driver board.

3.3 IR Sensor Board

We analysed the IR ball sensor designed by TIGERs in 2016 [9], which appears to be good, and have decided to continue using the same design for our robots.

3.4 Kicker board

The initial robots were tested with the TIGERs' double-kicker board [10] which worked perfectly. Mechanical modifications and electronic layout improvements have been implemented to improve performance, as shown in Figure 6. The solenoid wires are directly soldered to the bottom of the board. The kicking solenoid is an off-the-shelf part which seemed viable at this stage, unlike the other teams who wound their own solenoids.

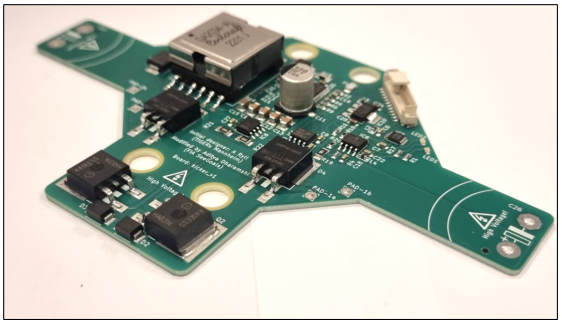

Fig. 6. The kickerboard without capacitors.

4 Firmware Overview

The firmware team's focus has been divided into two main components. One is communication from the decision-making software to the robots, the job of the base station. The second component is the robot's firmware, which receives the commands and executes them.

Since both main components are written for STM32 microprocessors, we rely heavily on the STM32 HAL library. This made getting started much easier but also increased the cost of not knowing exactly what happened within this layer. We rely on the STM32CubeMX software to configure components such as UART/SPI/I2C handlers, timers, interrupts, and more. This has both sped up development and caused problems more than once. Mostly because of the software's peculiarities and its configuration. It's built to be relied upon, but we've seen edge cases where the default configurations from CubeMX haven't worked out of the box, leaving us to create workarounds that might've been avoided had we written everything ourselves from the ground up. But doing that,

without experience with embedded software development, might've proved too big a challenge. Today, we see the HAL layer, STM32Cube, and its surrounding software as a necessary evil to reduce the complexity of getting started with the firmware.

4.1 Base station

The base station's hardware consists of an NUCLEO-H563ZI development board[6] and an nRF24 transceiver module PCB. The nRF module was in use by other teams, is cheap, and had already been battle tested.

The base station differs from the robot's development board, an NUCLEO-H755 [5], simply because it was bought first. Soon thereafter the development of the robots were to start, but there were simply no H563ZI boards to buy. Thus we diverged.

The base station mainly consists of a ThreadX core, a low-footprint RTOS forked from Azure RTOS ThreadX and tailored for STM32 (an included middleware in the CubeMX software). It enabled the use of another forked middleware, NetXDuo, which is a complete TCP/IP stack. This made it possible to receive ProtoBuf data which is packed into the IPv4 UDP packets from the controller software. Each ProtoBuf packet is unpacked and transmitted raw over the nRF in the order received on the base station's Ethernet port. There are no retransmissions, nor any acknowledgement of a received packet from the robot, by design, as packet throughput has been shown to be more important than reliability. The general flow of communication is shown in Figure 7.

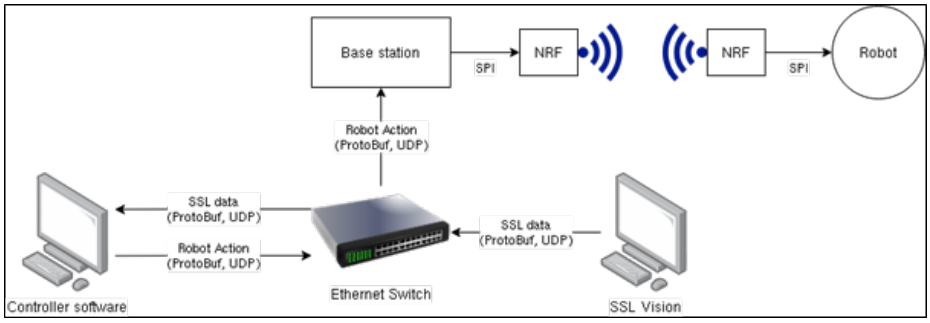

Fig. 7. Diagram of communication flow.

Because of the nRF's inherent protocol, packets are limited to 32 bytes. This could be circumvented but would introduce overhead, and a decision was made early not to add that overhead and instead try and ensure we never exceeded that size for the packets. The structure of a Robot Action packet is shown in Figure 8 below in its ProtoBuf form. This means that not the entire packet is filled with data; it depends on the Action Type sent. For example, a Move To action does not contain the same data as a Kick action.

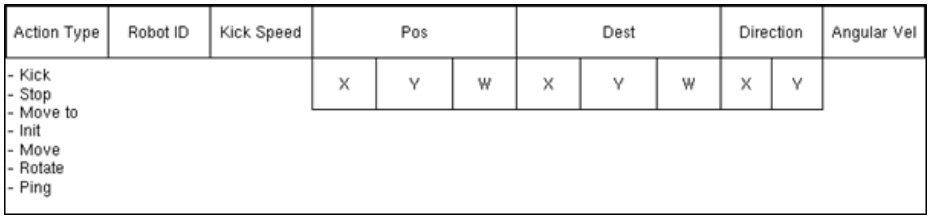

Fig. 8. Structure of the ProtoBuf message. Using ProtoBuf lets us simply ignore certain fields when sending a message to stay below the 32-byte limit on the nRF module.

The original idea was to receive both Robot Actions and SSL Vision packets and pass these on to the robots. Essentially broadcasting out the SSL Vision data to all robots and sending the specific robot commands to the target robot. Instead we include the robot's current position in each Robot Action. Since these actions are sent very frequently, they work similarly but are packed into the same structure. This introduces some delays as the packets need to pass through our software, but it reduces complexity.

5 Robot

Execution flow on the robot is mainly controlled by two mechanisms: incoming packets from the base station, and several hardware timers, both of which trigger interrupts in the system to handle the real-time requirements. The hardware timer-based interrupts are simple to implement using the STM32 HAL layer, and have proved essential to make sure our control loops run in a consistent manner.

5.1 Communication

The communication module receives raw robot commands from the base station via an nRF radio. These commands are parsed into Protocol Buffer (Protobuf) messages to simplify internal handling. The resulting packets are then forwarded to the control, and state estimation modules.

5.2 Position Control

Position control is implemented using proportional (P) control loops for both translational position and robot orientation. The robot receives target position and orientation commands from the base station, which are used in the control system.

5.3 Motor Control

The motor control uses a PI-loop and relies on encoder ticks obtained from Hall sensors on the motors, providing 48 ticks per revolution. This resolution has proven to be relatively low for accurate motor control at low speeds. This has caused us issues, and higher-resolution external encoders would be preferable; we aim to look at this in the future to improve motor control. Odometry and motor control was fully inspired by RoboTeam Twente 2020 TDP[8] in the section about wheel control, it was easy to follow and we would recommended it to other startup teams.

5.4 State Estimation

Position commands are sent to the robots, inspired by the approach used by TIGERs Mannheim. Due to a significant delay in our vision system (approximately 200 ms), this approach allows the system to partially compensate using onboard sensor feedback.

Each robot runs an Extended Kalman Filter (EKF) for state estimation. The EKF is based on the implementation used by TIGERs Mannheim, with modifications to account for differences in sensor setup and to reduce overall complexity. The filter fuses vision measurements, gyroscope data, and wheel odometry derived from motor encoder ticks. We investigated fusing accelerometer data, which is integrated into our IMU, but found it very noisy to the point where it only ruined our estimates.

5.5 Ball Sensing

An infrared (IR) sensor exists on the robot. However, the firmware to use it has not been fully developed yet. Integration of this sensor is a focus area of firmware development prior to competition, with the goal of reducing missed kicks, improving kick timing, and reliability.

This means that currently ball detection relies on vision data, and due to the vision system's latency in our lab this leads to timing issues when executing kicks.

6 Software Overview

We develop our main AI software in Go, and all components of the software are packaged as several Docker containers to simplify deployment and development. During development, SSL-Vision together with the ER force simulator is used to simulate the game. The TIGER Mannheim AutoRef system is used to obtain filtered vision data, as it was observed that the ball, robot IDs, and positions can be noisy and contain outliers.

6.1 Path Planning – RRT

Path planning is implemented using a Rapidly-exploring Random Tree (RRT) based approach. The planner is responsible for finding a collision-free path from the robot's current position to a target position.

The RRT planner operates in the two-dimensional field space and considers other robots and restricted areas as obstacles. Due to the dynamic nature of the environment, the planner runs frequently to allow replanning when obstacles move or when the target position changes. The output of the planner is a sequence of waypoints that represent a feasible path for the robot to follow.

While RRT is simple, it has issues with producing smooth paths and each iteration gives different results. Therefore, we plan to investigate using RRT-Star or a grid-based approach before the competition to improve robot control and reliability of the planner.

6.2 Trajectory Planning – Pure Pursuit

Trajectory planning is implemented using a Pure Pursuit controller. From the sequence of waypoints produced by the path planner, a collision-free waypoint is selected at a fixed distance ahead of the robot. This waypoint is used as the destination that the robot drives toward. When the last position along the path is within the lookahead distance, the destination point is chosen, if it is collision-free.

6.3 Game Planning

Game planning is divided into two main components: the planner and the activity executor. The planner assigns roles to robots, where each role consists of a set of activities that can be executed. The activity executor runs in a separate thread to ensure that time-critical components, such as control loops and path planning, can operate at high frequency.

Activities make use of a set of instruction primitives to achieve simple goals. Examples of activities include moving to a position, kicking the ball to a target position, passing the ball, and receiving the ball.

Roles combine these activities to implement behaviors such as guarding or goalkeeping. At the current stage, goalkeeper behavior, guarding, passing, and a simple set of attacking plans are implemented.

References

1. 50mm Diameter 4mm Hole Size Omni Wheel, `https://gtfrobots.com/product/50mm-diameter-4mm-hole-size-omni-wheel/`, publisher:
2. DF45S024050-A2 - Brushless DC flat motor, `https://www.nanotec.com/eu/en/products/1785-df45s024050-a2`
3. LSM6DSL IMU, `https://www.st.com/en/mems-and-sensors/lsm6dsl.html`

4. MCT8316Z - Automotive 40-V max, 8-A peak, sensored trapezoidal control 3-phase BLDC motor driver, https://www.ti.com/product/MCT8316Z-Q1
5. NUCLEO-H7536ZI, https://www.st.com/en/evaluation-tools/nucleo-h563zi.html
6. NUCLEO-H755ZI-Q, https://www.st.com/en/evaluation-tools/nucleo-h755zi-q.html
7. Tattu R-Line Version 5.0 1050mAh 6S 22.2V 150C Lipo Battery Pack with XT60 Plug, https://genstattu.com/tattu-r-line-version-5-0-1050mah-6s-22-2v-150c-lipo-battery-pack-with-xt60-plug/
8. Bos, L., Citgez, Y., Dorenbos, H., Eichler, A., van der Hulst, R., Klein Nagelvoort, L., vad der Kuil, T., Loung, J., Poort, L., Prinseberg, F., de Regt, C., Schoot Uiterkamp, L., Schreijver, R., Steernman, E., Vacariu, P., van Werven, J., van der Werff, S., Zwerver, S.: RoboTeam Twente Extended Team Description Paper 2020. Tech. rep., University of Twente (2020), https://ssl.robocup.org/wp-content/uploads/2020/03/2020_ETDP_RTT.pdf
9. Ryll, A., Geiger, M., Ommer, N., Sachtler, A., Magel, L.: TIGERs Mannheim - Extended Team Description for RoboCup 2016. Tech. rep., Cooperative State University (2016), https://download.tigers-mannheim.de/tdps/soccer_smallsize__2016__TIGERs_Mannheim__0.pdf
10. Ryll, A., Jut, S.: TIGERs Mannheim - Extended Team Description for RoboCup 2020. Tech. rep., Cooperative State University, (2020), https://download.tigers-mannheim.de/tdps/soccer_smallsize__2020__TIGERs_Mannheim__0.pdf