Team Terra Force

Robert Flugrat

Peter Ehrlich

Richard Flugrat

Table of contents

Mobility Management	2 - 4
Power and Sense Management	5 - 7
Obstacle Management	8 - 10
Pictures - Team and vehicle	11 - 14
Engineering / Design	15 - 17
Appendix	18 - 20
Appendix - Links	21

The robot has a total of two motors, a servo motor for steering and a gear motor for driving.

Steering servo motor:

We opted for a servo motor (Figure 1) in conjunction with a gear combination to move the steering wheels. This design is very direct, has little clearance and allows for much greater steering angles than a system with a steering rod. Combined with a short wheelbase, this results in a very manoeuvrable robot.

In our design, rear-wheel steering improves the robot's responsiveness because the sensors are located in front of the steering mechanism.

A servo is standard technology and widely used. We chose the low-profile digital servo “AGF-RC B44DLM V2” as our servo motor due to its low weight, suitable dimensions, high positioning speed and extended voltage range.

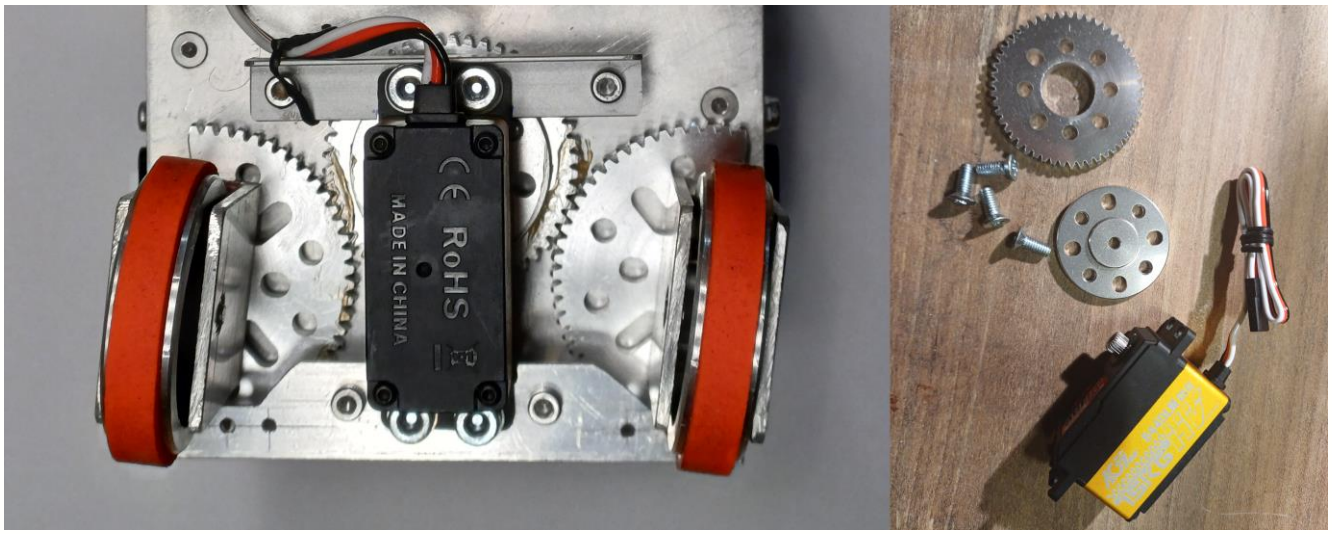


Figure 1: Steering gear assembled (left) and disassembled (right)

Servo specifications:

Operating voltage:	4.8 - 8.4 V
Acting force:	120 - 150 N/cm
Operating speed:	0.18s / 60° @ 7.4V
Weight:	44 g
Type:	digital servo
Update frequency:	50 - 333 Hz

Drive motor:

We chose a very small and lightweight high-performance gear motor.
The “INJORA 180 55T” (Figure 2) is also often used in model cars.
The motor has a voltage that matches the battery (7.2 V).

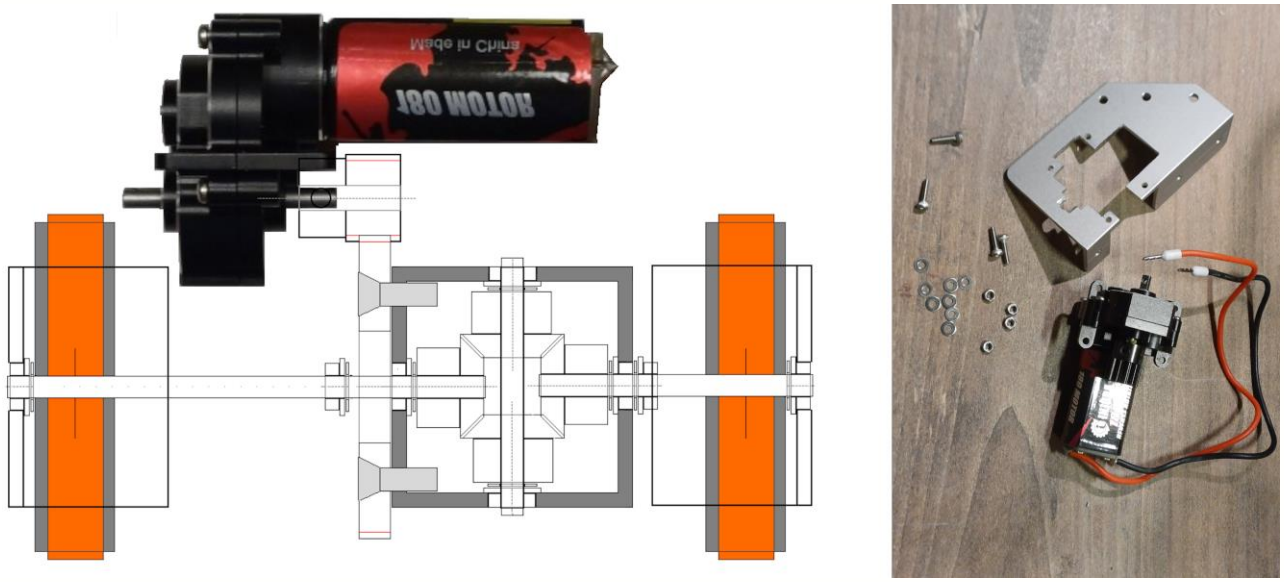


Figure 2: Schematic representation of the drive (left), motor block with holder (right)

Electric motor data:

Voltage:	7.4 V
Power:	22 W
Type:	brushed motor
Speed (without gearbox):	18500 rpm
Weight (without gearbox):	39 g

The motor is controlled by a MOSFET transistor (IRLZ44N).
This transistor is a TTL type and is controlled directly via the ESP32.
We chose this solution so that we could directly influence the PWM signal.
A conventional RC car motor controller would not allow this.
The motor direction change is implemented by a relay, which is only switched when the motor is at standstill.

The motor current can be measured via the voltage drop across a measuring resistor.
To enable the ESP32 to measure this small voltage more accurately, it is first amplified 20 times by an “INA193.”

The torque is transmitted to the wheels via a differential gear on the non-steered axle (Figure 3). This enables us to take tight corners.
The motor (top of Figure 3) is equipped with a very space-saving gearbox (24.4:1).

The torque is transmitted to the differential via another open gearbox (3.75:1).
All elements have been designed and machined to ensure that the maximum robot size of 150 x 100 mm can be maintained.

The aluminium wheels are equipped with a silicone tread, giving them more grip on the ground, which enables better cornering with little drift and powerful acceleration without visible slippage.

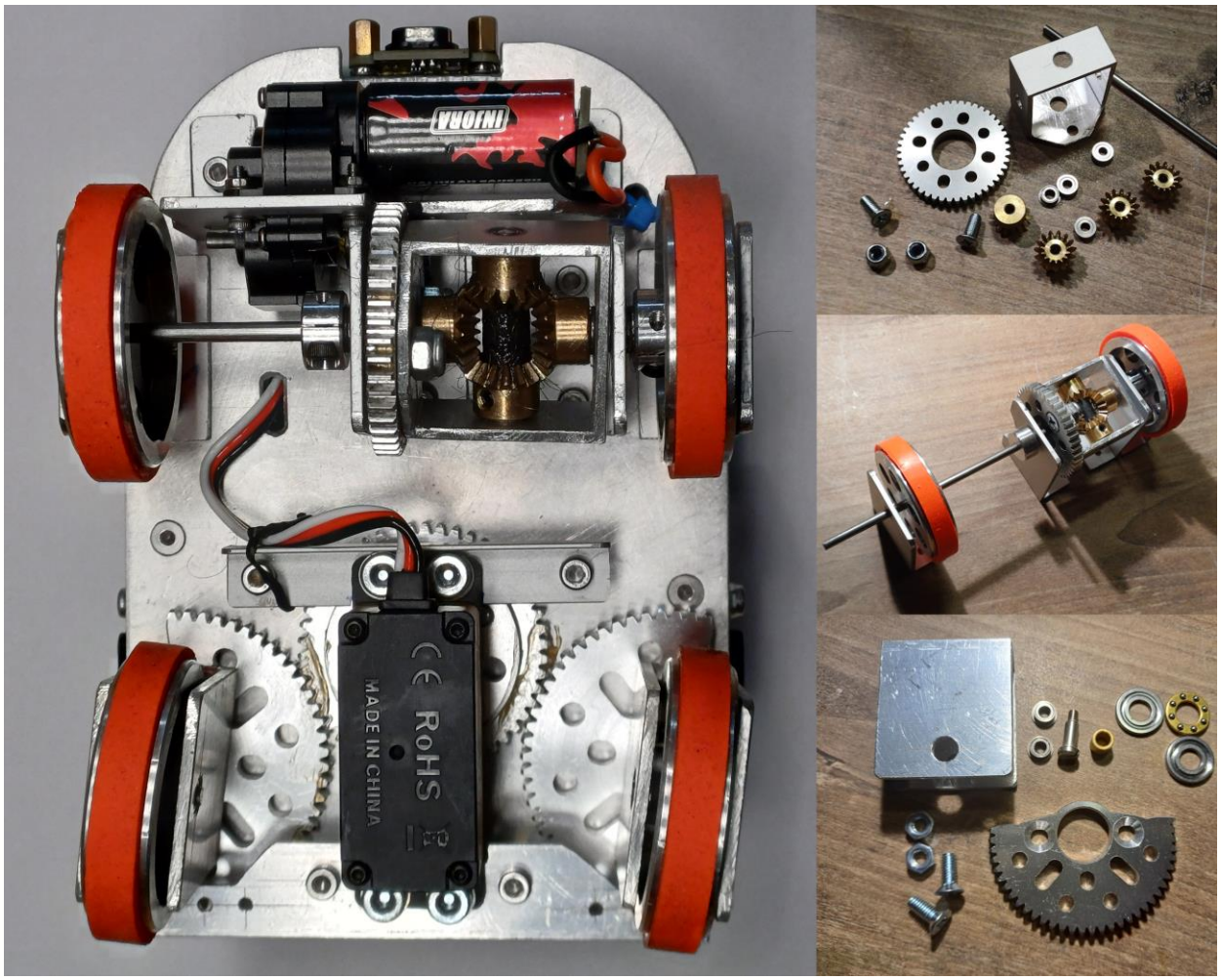


Figure 3: View from below showing the gears

The chassis is based on a self-designed aluminium plate. Aluminium was also used for most mechanical components to achieve a good compromise between stability and weight.

Energy:
We decided to use NiMH batteries as the energy source.
The charging technology for this was already available. In addition, they are safer to handle than lithium batteries and charging them in the robot is also easier to solve.

We use 6 NiMH batteries with 3400 mAh each at 1.2 V. These batteries are capable of high currents (up to 25 A).
We have assembled them into a compact battery pack using 6 vertical Sub-C cells for space reasons. This arrangement is rather rare in stores and we have also provided a T-plug connector so that we can quickly change the battery without tools.

Figure 4: Batteries individually (top) and as a battery pack (bottom)

Data from the finished battery pack:

Voltage:	7.2 V
Capacity:	3400 mAh
Weight:	362 g

The battery pack is charged at the T-plug socket when the robot is switched off.
We were looking for a high-current-resistant battery because the Raspberry Pi 5, the AI accelerator, the servo, and especially the drive motor can consume a lot of power for short periods of time. We expect a maximum short-term power consumption of up to 10 A.

The battery voltage is distributed via the “power board” after a slow-acting main fuse (10 A).
The voltage from the battery is measured and monitored via a voltage divider.
The current consumption is measured at two points using shunt current sensors (INA193). This provides us with information about the motor current and the total current consumption, which can be used for further safety functions in the future.
The steering servo is a high-voltage type and is directly connected to the battery voltage, as is the motor driver (MOSFET IRLZ44N) with motor.

→ Wiring diagrams are in the Appendix (page 18)

The 5V supply voltage for the Raspberry Pi 5 and the ESP32 is generated by a switching regulator “Pololu D36V50F5”. This supplies a maximum of 5.5 A
and generates the 5 V even when the battery is almost discharged.
Originally, a 7.5 A low-drop linear regulator (MIC29501) was planned.
However, this became very hot, so the circuit had to be changed.
The 3.3V for all sensors is provided by the ESP32.
The Raspberry Pi 5 can be switched off (via slide switch) to save energy.
The robot can also run without the Raspberry Pi 5 (but then without a camera).

The power supply board uses LEDs to indicate the presence of battery and 5 V voltages.
The logic board (upper board) has LEDs to indicate the 3.3 V voltage
and the power supply of the Raspberry Pi 5.

The LCD display shows the vehicle status, including battery voltage.
It can also be used to display sensor readings or debug information.

Sensors:
We have installed a total of 5 sensors + 1 camera for different tasks:

Rotation measurement:

gyro sensor MPU-9250
- 3-axis gyroscope
- 3-axis acceleration sensor
- 3-axis magnetic field sensor
(Earth's magnetic field/compass)

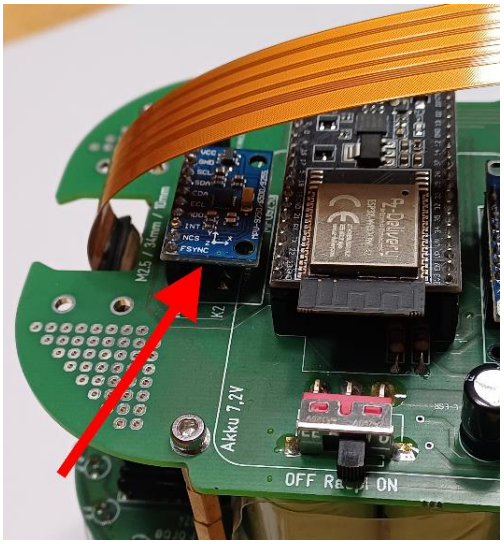


Figure 5: Position of the MPU-9250

We chose this sensor
because it is easily accessible via I2C
and contains an integrated motion processor.
It is also more accurate and reliable than the
MPU-6050, which we used in our last robot.
We mainly use it to determine the robot’s rotation.

→ Wiring diagrams are in the Appendix (page 18/19)

Obstacle detection:

Raspberry Pi Camera Module 3 Wide
- 12 megapixels Sony IMX708 sensor
- Autofocus with phase detection

We chose this camera because of its small size, high resolution, and autofocus feature.
The software is compatible with an AI acceleration module called “Raspberry Pi AI HAT+” with soldered Hailo8, which we use for image evaluation on the Raspberry Pi 5.

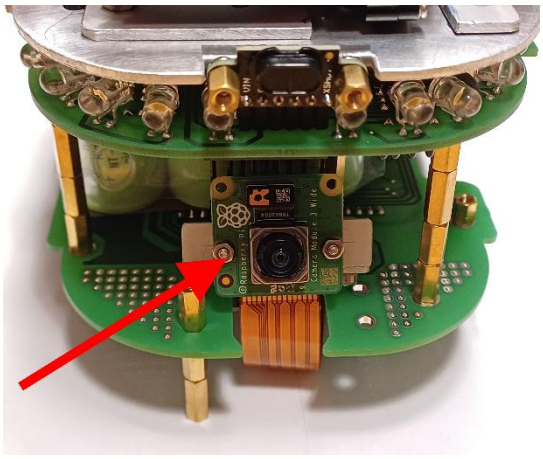


Figure 6: Position of the camera (vehicle turned upside down)

Distance measurement:

VL53L1X “Time of Flight” laser sensor
- Range: 4 to 400 cm
- Resolution: 1 mm
- Polling rate: max. 50 Hz
- Switchable distance modes (3)
- Adjustable measuring point

We chose this sensor because it is significantly more suitable than the only alternative, a VL53L0X.

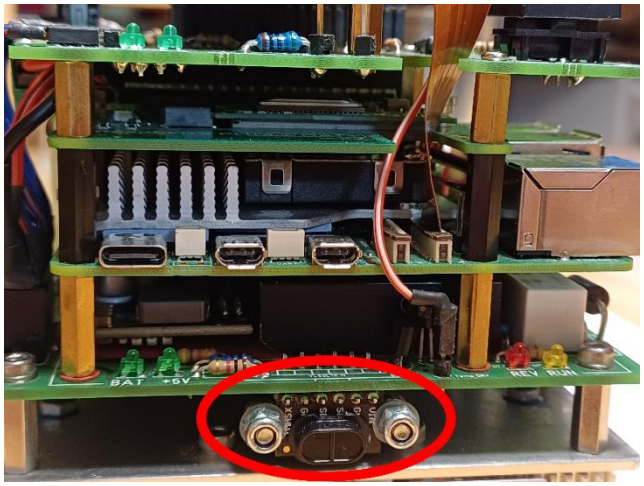


Figure 7: Position of the rear VL53L1X with mechanical protection

For last year's robot, we used ultrasonic sensors (HC-SR04). However, these sensors did not reliably detect the track boundaries if they were not aligned at right angles to the sensor.

We were considering to use a LIDAR, but the polling rate is significantly slower and multiple LIDARs would use too much space.

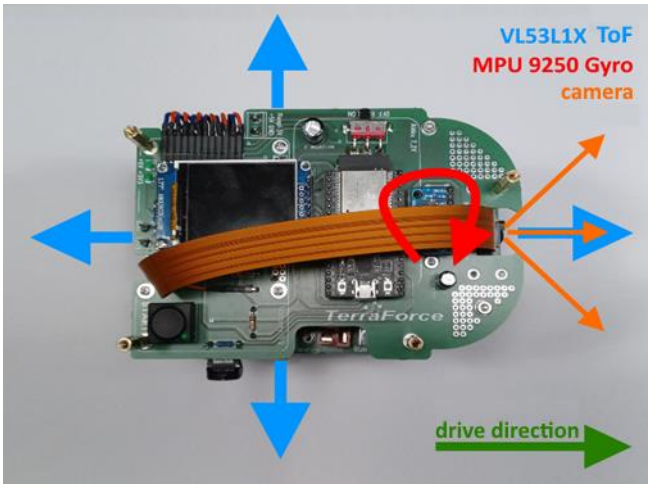


Figure 8: Arrangement of the main sensors

The various sensors and actuators are controlled by a microcontroller.
We also use a single-board computer for image analysis:

WRO ESP32 (ESP32 Dev Kit C V4):
Here we have used a powerful microcontroller from Espressif.
The two cores with a clock speed of 240 MHz are required for reading data from TOF sensors, start button and gyro sensor. At the same time, the ESP32 controls the drive motor, servo motor and LCD, and runs our driving code.
This microcontroller automatically receives information about objects on the track via UART from the Raspberry Pi 5 (when it's active).
The microcontroller generates the PWM signal for the steering servo and also the PWM signal for the main motor, which is pulse width and frequency variable.
The ESP32 measures battery voltage and current consumption via ADC inputs.

WRO RPI5 (Raspberry Pi 5):
This powerful computer also has an AI accelerator mounted on it ("Raspberry Pi Ai HAT+" with soldered Hailo8) with a connected camera ("Raspberry Pi Camera Module 3 Wide").
This computer handles image capture and object recognition.
We trained "YOLO11s", an AI training algorithm that is also suitable for the AI module, with over 2200 labelled images of the three objects in different positions and under different lighting conditions for more than 12 hours on a powerful graphics card (GeForce RTX 4060)
We then compiled the resulting AI model for the Hailo8.
It is executed on the Raspberry Pi 5 using the pre-installed rpicam-hello command:

```
rpicam-hello --framerate 50 --hflip --vflip
--post-process-file /home/terra/Documents/WRO\ RPI5/hailo_yolo11s.json
-t 0 -v -n --width 1296 --height 1296
```

Because the camera is upside down for mounting reasons, the video stream is mirrored horizontally and vertically before it is transferred to the AI module for processing.
A self-written program reads the current object positions from the output of the command, sorts them and sends them in binary form to the ESP32 via UART.

Every image requires on average 17 ms of inference time which limits us to a framerate of 50 Hz, where everything is running without issues.

There are two different types of challenges: open challenges and obstacle challenges. There are also two programs, one of each challenge, but they are located in the same file. The type of challenge is specified in the program before it is uploaded onto the ESP32.

Open Challenges:

For this challenge, the Raspberry Pi 5 is not required and switched off to save power.
After switching on the robot, the ESP32 starts immediately and initializes all sensors and actuators. The boot process takes about 7 seconds; its status is shown on the LCD display.
When the robot is ready, the green LED on the start button lights up.
When the button is pressed, the robot starts accelerating immediately.

First, the robot drives until one of the side ToF sensors detects a large space (> 1m).
At this point, the robot knows the driving direction and starts the first curve which ends when the rotation is less than 35° away from the target. This large difference is needed because of sensor latencies and servo response time. The exact target rotation is reached later using smaller servo movements. In the straightforward sections, the distance data of the side sensors is constantly monitored to avoid border crashes. When the measured distance is below 15 cm, the robot steers away from the border by simply leaving the servo on a large angle in the opposite direction for a short amount of time.

When the one of the side distance sensors detects a large space or the front distance sensor measures a distance below 65 cm, the next curve is started. Because this might happen right after a curve ended, there is a cooldown of currently 750 ms before the next curve can be started.

When the absolute value of the target distance reaches 1080° (3 rounds completed), the robot stops in the starting section and shows the total run time,
which is typically between 40 seconds and one minute, depending mainly on the amount of narrow straightforward sections.

The driving speed can be reduced before reaching a curve, but this is currently not necessary (the robot drives at maximum speed and only stops after 3 rounds).
Whenever the driving speed is changed, the robot slowly (de-)accelerates asynchronously.

Obstacle Challenge:

The Raspberry Pi 5 with AI accelerator is required for this challenge and needs to be switched on. The starting procedure is the same as for the open challenges but the boot time is around 45 seconds here because of the Raspberry Pi 5.

Our code is designed to always start from the parking space. After pressing the start button, the robot already knows the driving direction because at the ToF sensor at the side of the outside border measures a very small distance. The procedure of leaving the parking space consists of 4 small moves, 2 backward and 2 forward which are executed slowly (15% speed). After leaving the parking space, the robot starts checking for traffic signs and records them with colour and position. After passing the first sign on the right side (if existing), the robot reaches the first curve.

During the first round, when the robot detects a curve, it drives backward, turning to the outside border to have a wide view on the next straightforward section. Then, depending on the first object, the robot either directly drives forward (close to the inner border) or changes the driving side before entering the straightforward section. If there is a second traffic sign in the section with a different colour, the robot switches sides in the middle of the section using two right angle curves.

After recording all the traffic signs during the first round, the robot can drive directly around them, without the need of driving backward after a curve. The robot doesn't care on which of the two positions at the start, middle and end of a section a traffic sign is placed, it drives around both possible positions.

The most important variables in our code are simply booleans: outsideBorder and drivingSide. outsideBorder stores the information, on which side of the section the outside border is located (driving direction clockwise = left = 0, counterclockwise = right = 1).
Our robot tries to constantly drive next to a specific side of a section = drivingSide (left = 0, right = 1). It prefers driving next to the inner border but also avoids unnecessary curves.

After 3 rounds, the robot simply stops in the starting section. The robot was originally designed for non-parallel parking (driving into the parking space at a right angle), because this was allowed during national competitions. A parallel parking function was not designed yet.

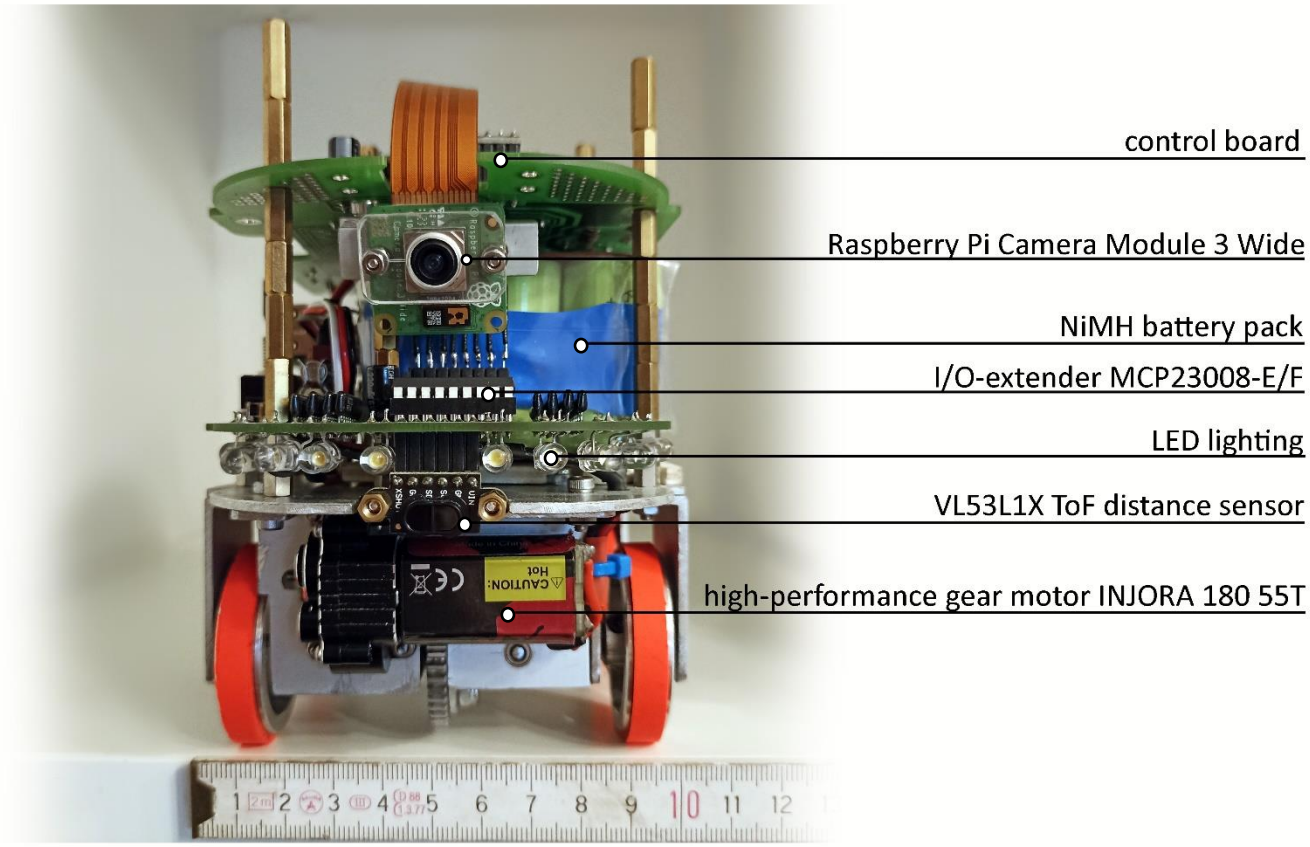


Image 9: Front view of the robot

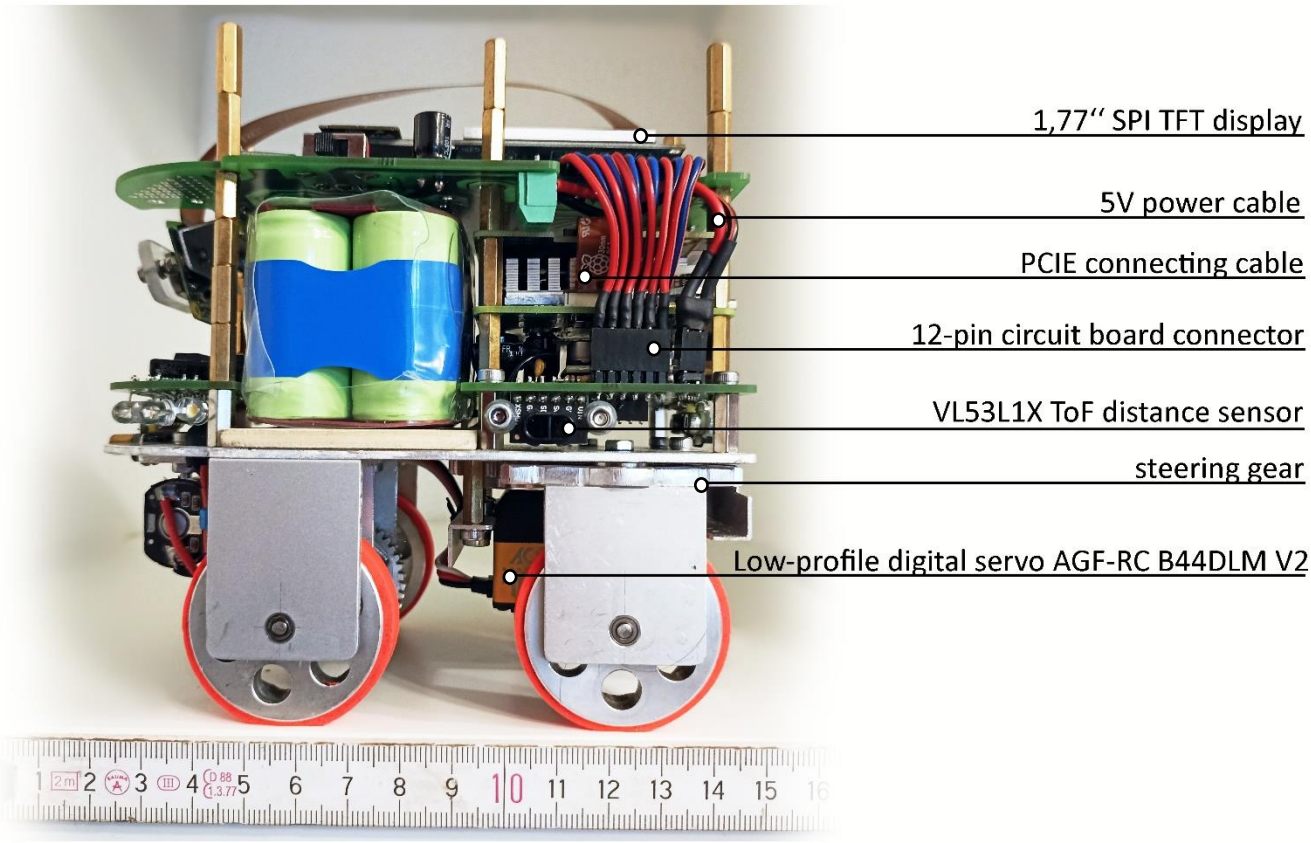


Image 10: Left view of the robot

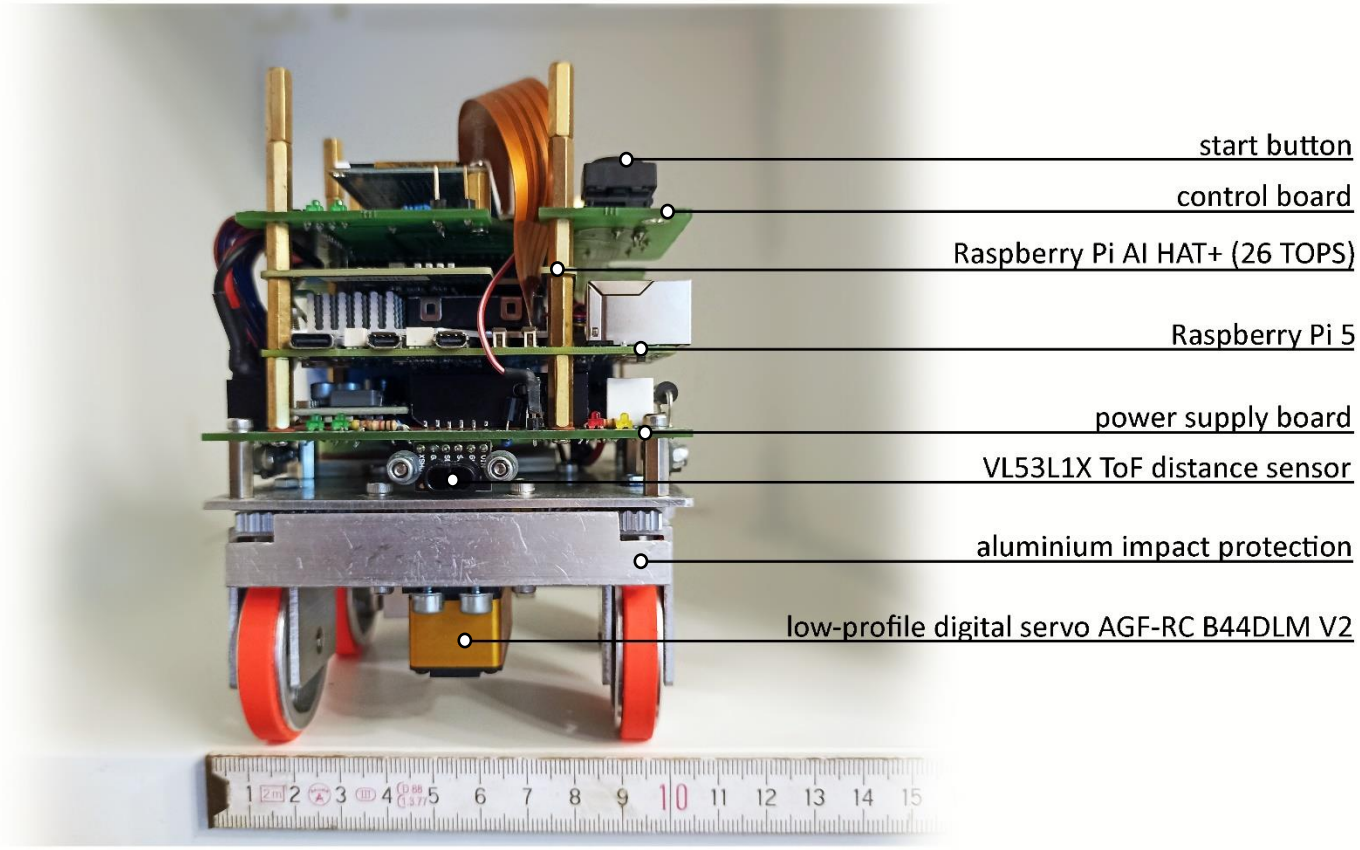


Image 11: Rear view of the robot

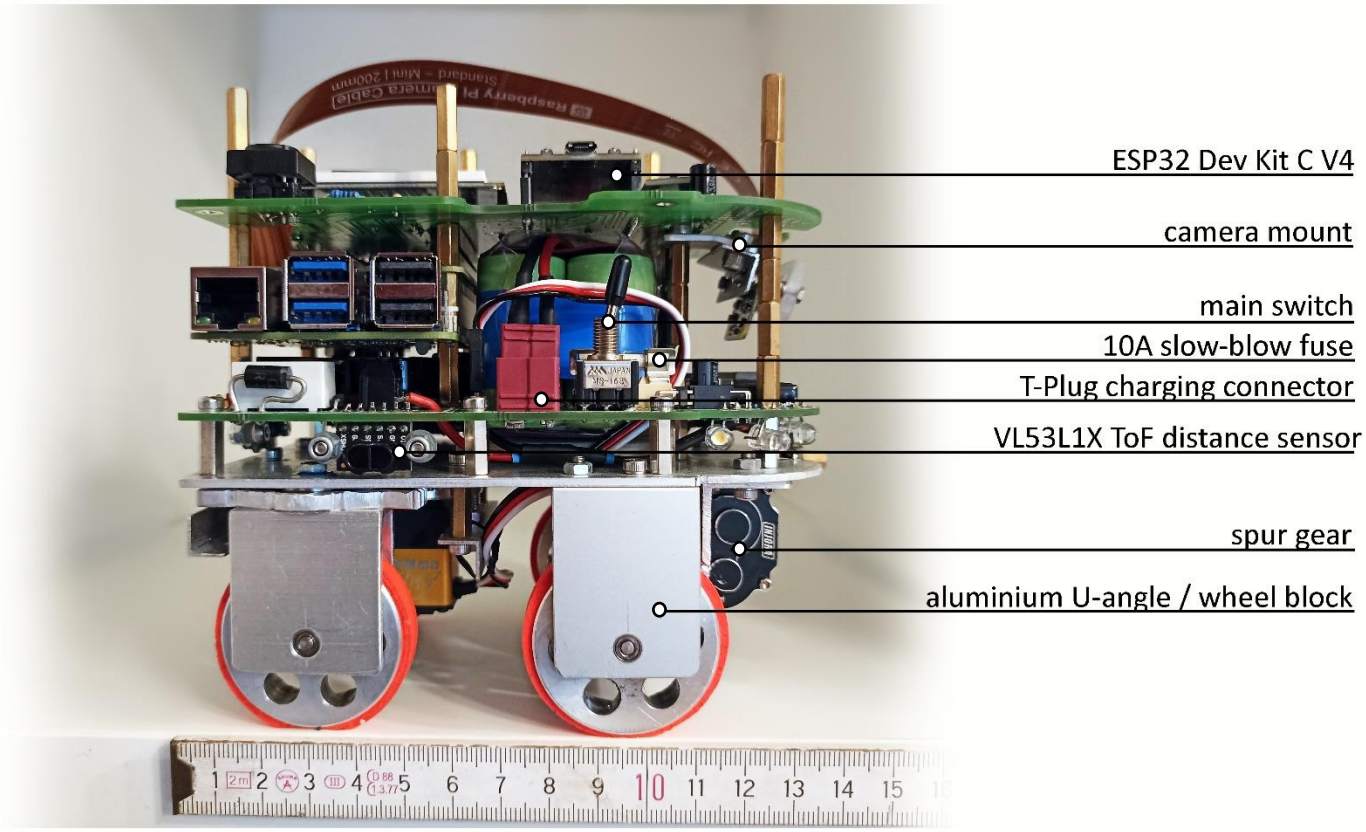


Image 12: Right view of the robot

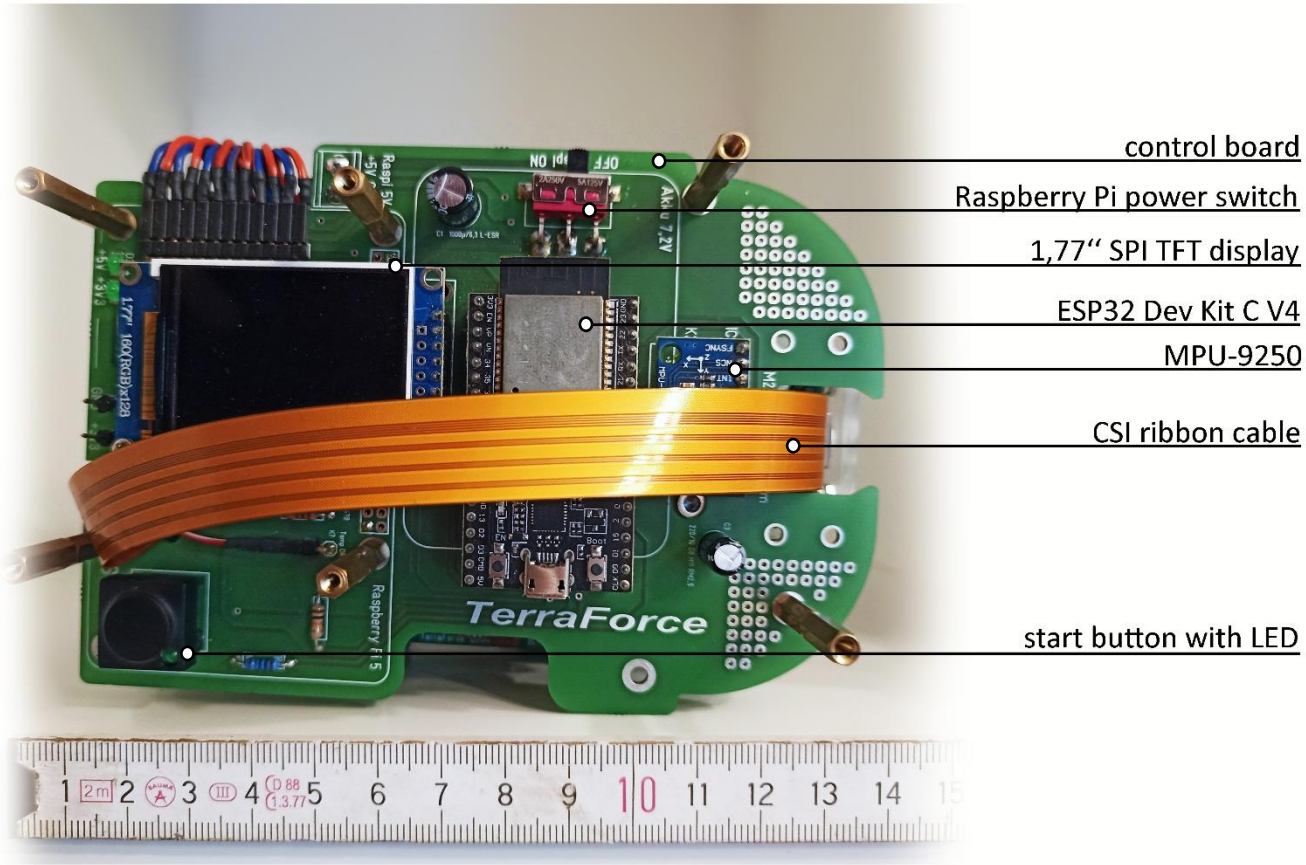


Image 13: View from above (drive direction to the right)

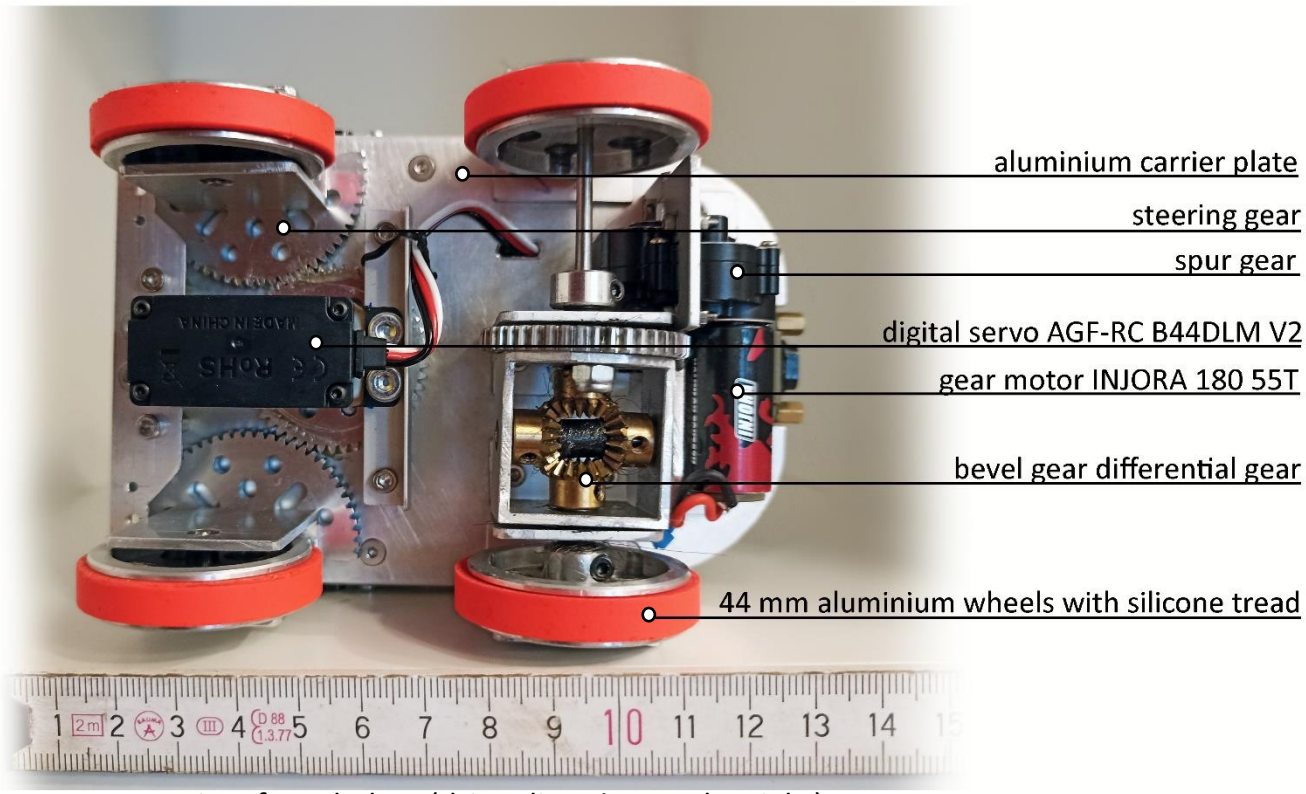


Image 14. View from below (drive direction to the right)

Pictures - Team

Review and changes:
Last year, we competed in the WRO with the robot shown on the right.

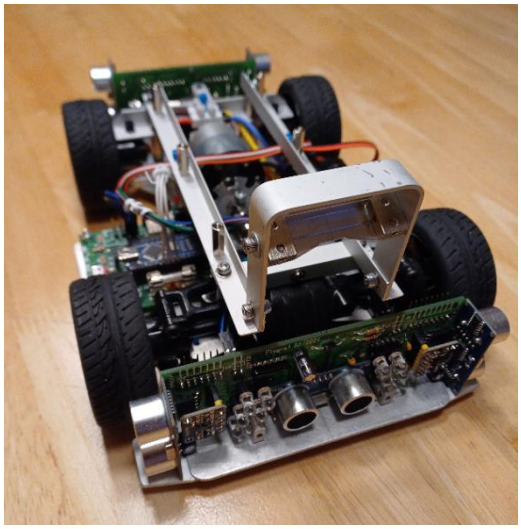


Figure 15: Previous robot

During testing and competition, several disadvantages became apparent that definitely need to be changed in the new vehicle:

The robot was too large (27.5 x 18.5 cm): it was difficult to start right next to the barrier in the open challenge. In addition, the turning circle was too large.

The motor was too weak:
the driving speed varied greatly with the voltage level of the battery.
Slow driving was not controllable.

Distance sensors: The ultrasonic sensors did not detect the track boundary when they were pointing at it at an angle far away from perpendicular.

The completely self-written object detection algorithm was very complex to program, too slow, and did not reliably detect obstacles.

Engineering:
Our new vehicle should be as compact as possible. We set ourselves a target formfactor of 15 x 10 cm. After researching online, it became clear that there were barely any suitable parts available, and never a complete chassis.
We therefore designed the chassis ourselves using available individual parts.
The differential gear (Figure 16) was also homemade, and the motor with gearbox was added using a customized bracket.
The steering was no longer implemented with a steering rod, but now with gears. This enables particularly tight steering radii (turning circle: 32 cm instead of 62 cm in the previous vehicle) and high precision.

Figure 16: Differential gear

Rear-wheel steering proved to be advantageous last year and was therefore used again.
However, very narrow wheels were now necessary for effective control;
there was only one suitable wheel set available (JSumo silicone wheel 43x11 mm).

The aluminium gears all have a module of 0.8.
The optimal gears for steering and drive were then found using different sizes (number of teeth).
All shafts are made of steel (3 mm) and mounted in ball bearings, which were glued into the bearing blocks.

Figure 17: Weight control

The motor is now quite powerful at 22 W.
To keep heat generation at the driver (MOSFET) low, we chose a motor with 55T (turns) instead of the more common 48T.
Since we are not using a speed controller (as in the last robot), we have to take care of the PWM signal for the motor driver ourselves.
After various tests, a combination of pulse width modulation and PWM frequency change proved to be optimal.
In addition, (de-) acceleration ramps had to be programmed to prevent current peaks during start-up and braking, and prevent slippage.

For the steering servo, we again used a digital servo (AGF-RC B44DLM V2), but this time an HV (high voltage) type. This allows it to be connected directly to the battery voltage.
This servo would also be externally programmable.
However, we did not make use of this option.

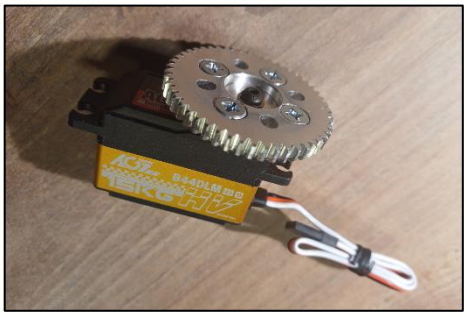


Figure 18: Steering servo + wheel

The ultrasonic sensors worked on the old robot, but only when they were positioned at right angles to the track boundaries. That's why we decided to try navigating with ToF sensors instead. We chose the top-of-the-range VL53L1X model because it offers switchable distance measurement and the active measuring point can also be changed.
In addition, there is a mechanical impact protection in front of the actual sensor.

For the gyro sensor, we chose a successor model to the MPU-6050 because it has more functions and a higher resolution.

The Raspberry Pi 5 with AI module and 12MP wide-angle camera was chosen for capturing and processing the camera images.

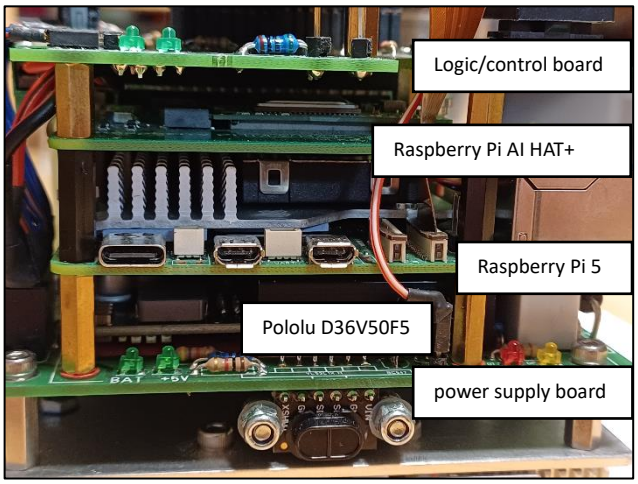


Figure 19: Circuit board overview

Electrical hardware:
Components for voltage conversion and current measurement, fuse, relay, and motor drivers are located on the bottom circuit board.

The battery goes into the large recess in the power supply board.
The LED lighting is also located on this level.

All processors, sensors, and the display are located on the top-level board.
Most of the sensors are controlled over two I2C bus system, the Raspberry Pi sends its data over UART.

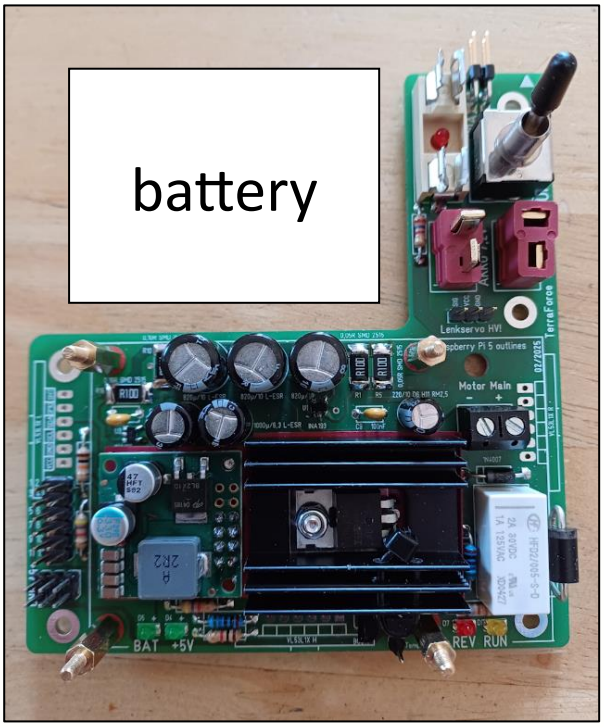


Figure 20: Power supply board

During the last WRO project, we had the unpleasant experience of discovering that a breadboard is good for development, but also costs many hours of troubleshooting because the contacts are not reliable.
For example, the main computer frequently malfunctioned due to a lack of power.
Creating a printed circuit board was therefore unavoidable.

We added an LCD display for troubleshooting and information display purposes.
This allows sensor data and the current status to be read (see appendix, page 21).
The robot is activated with the start button.
Capacitors help preventing voltage drops on the circuit board.

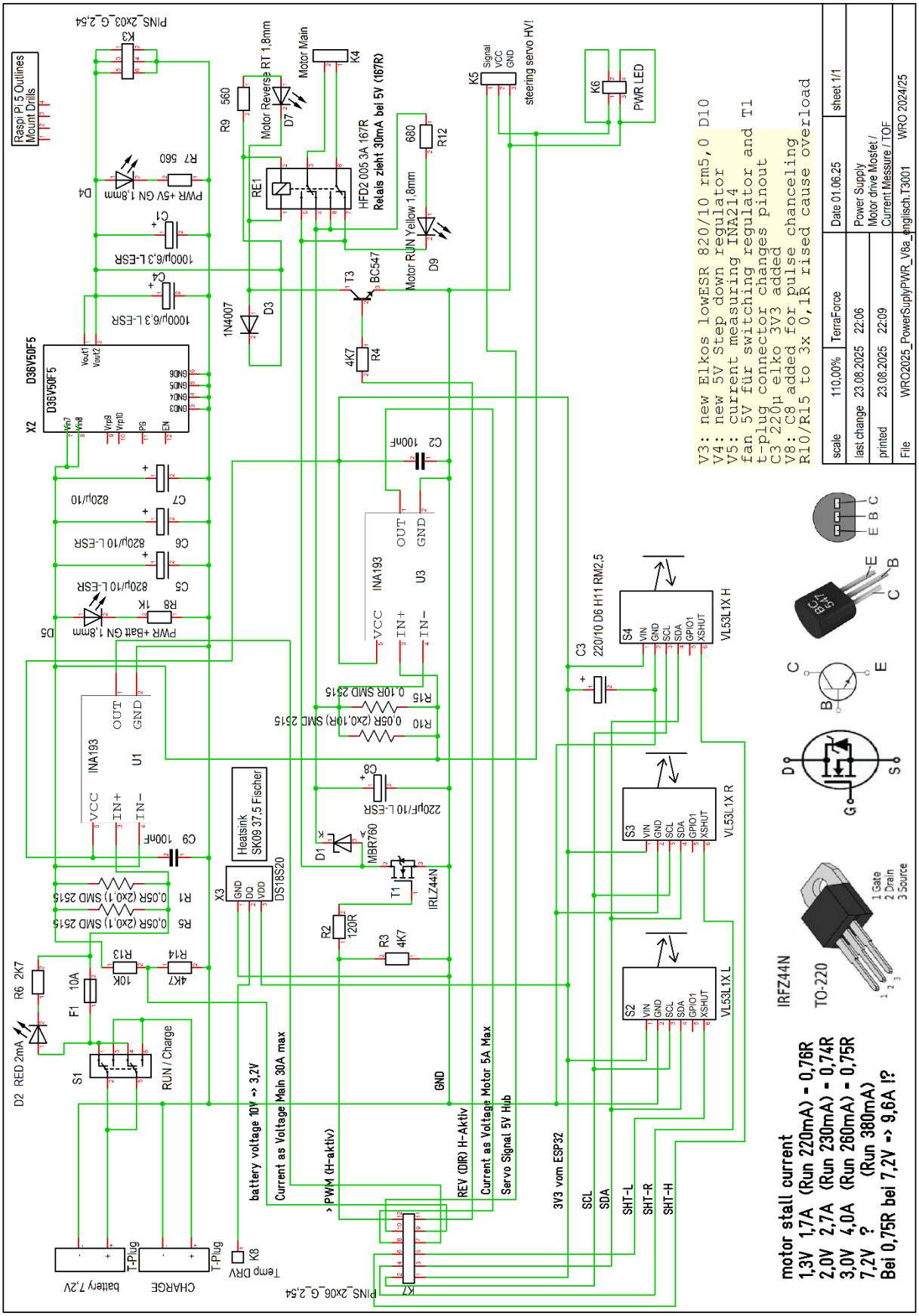


Figure 21: Circuitry of the main board

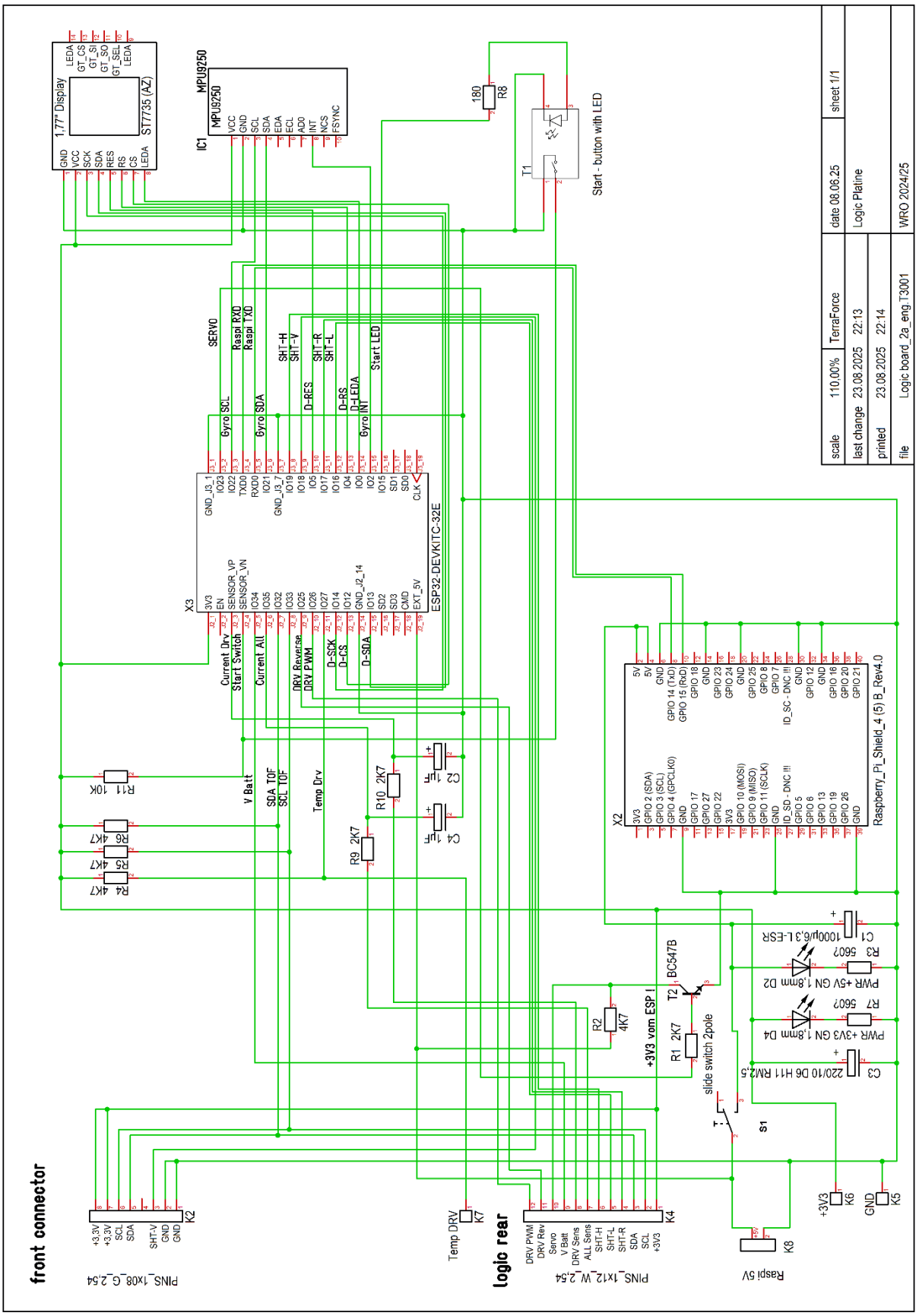


Figure 22: Circuit diagram of the logic board

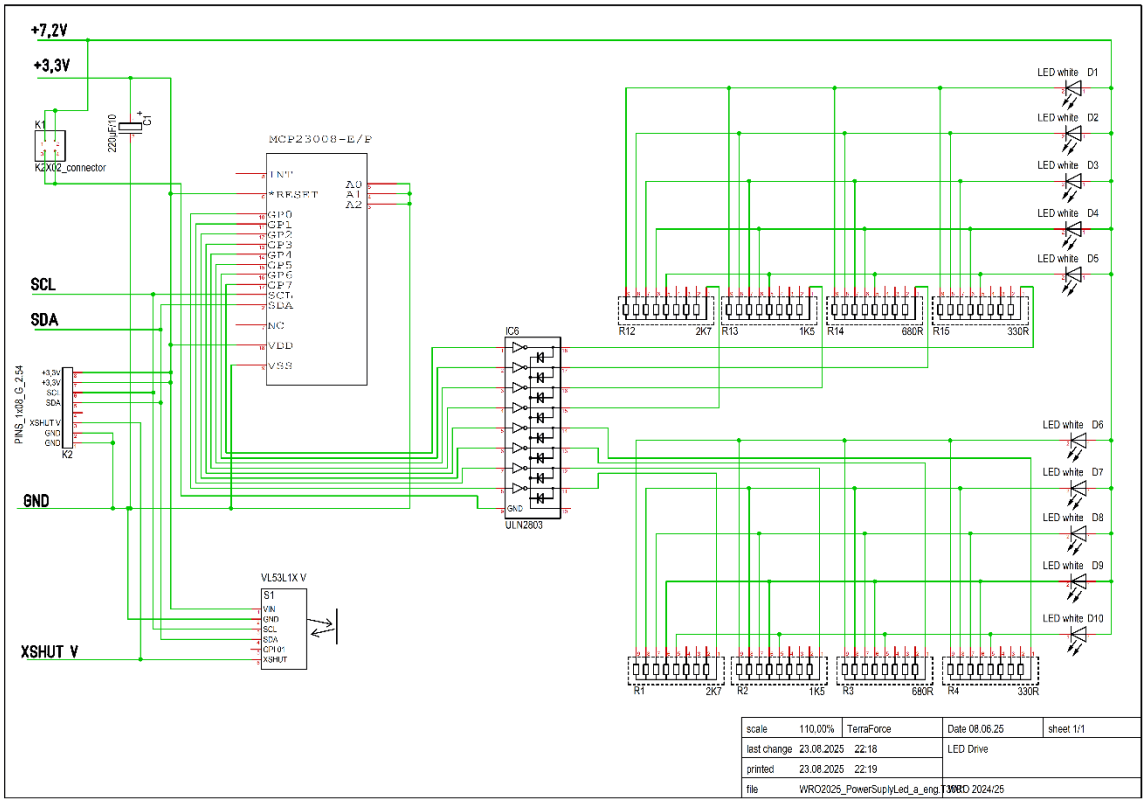


Figure 23: LED lighting circuit

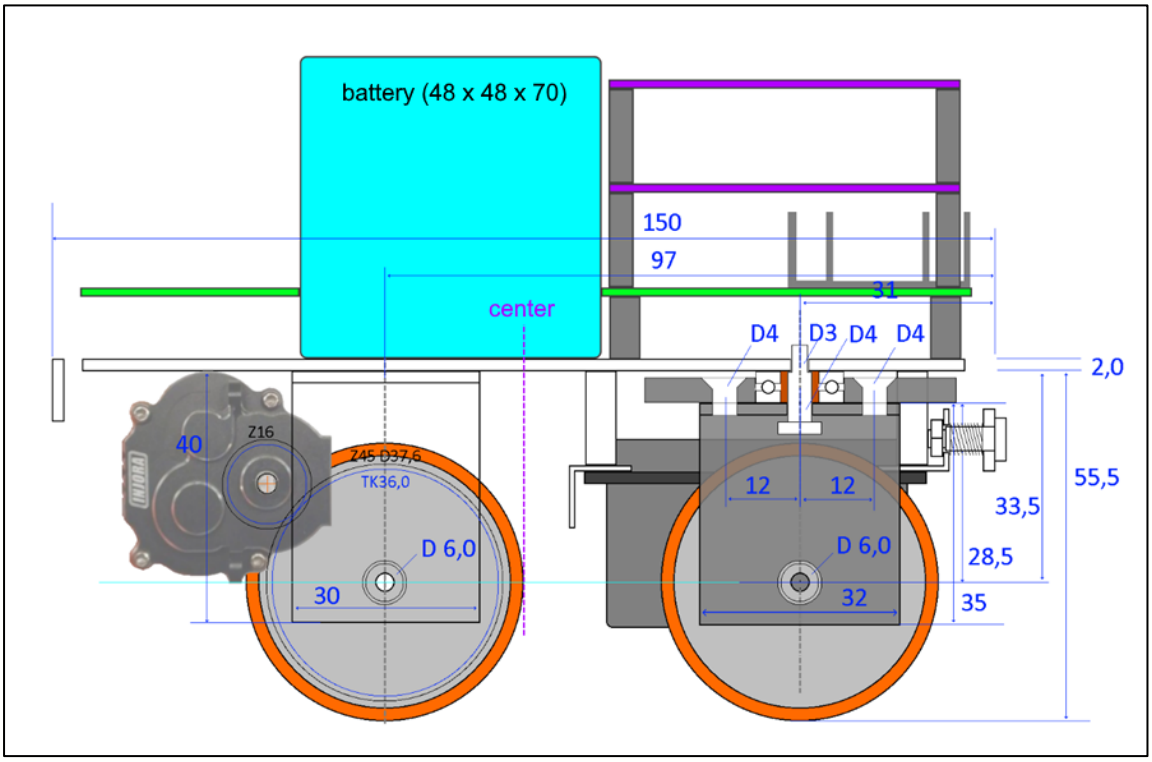


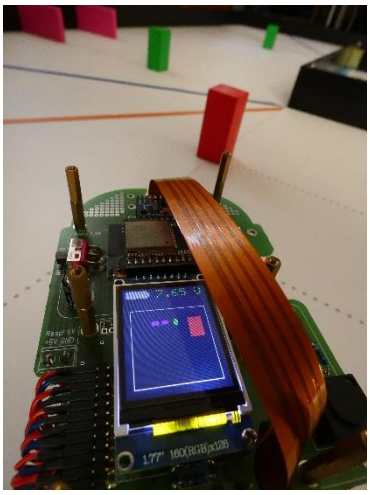
Figure 24: Sketch of the mechanical structure

Appendix – Links

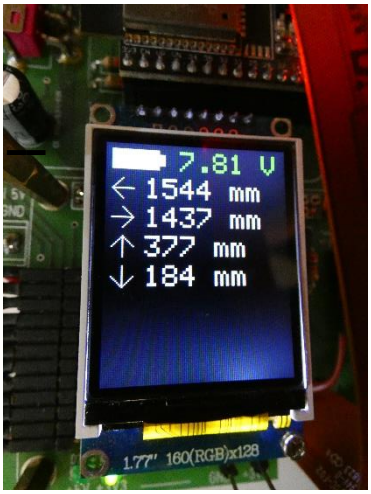
LCD Graphical User Interface:

ready message
with start time

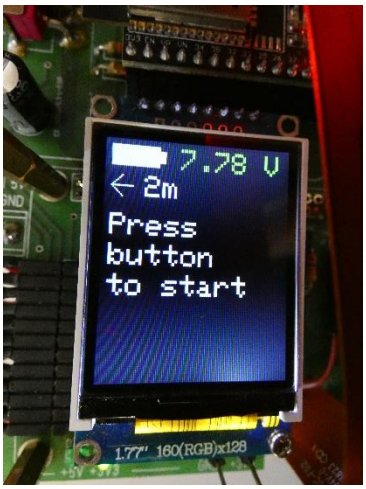
current measurements
and MOSFET temp.



detected objects live



distance sensor
measurements



distance sensor
calibration interface

distance sensor
calibration results

Video links

Open challenge: https://youtu.be/Z3k-EH-pLz8
Obstacle challenge: https://youtu.be/ie1yr02ISkE

GitHub repository

https://github.com/TerraForces/WRO2025-Future-Engineers