# RoboCupJunior Rescue Line 2024

## Engineering Journal

## Team BitFlip

## Introduction

This Engineering Journal details the development of our robot for the RoboCupJunior Rescue Line competition in 2024. We have developed many robots, so we are reusing many components, including software from previous robots. This Journal only details the work done on this years' robot.

Our development is very irregular because we all finished school last year and rarely find the time to meet up and continue developing, sometimes there are weeks where none of us did any work on the robot. And only every few months did we meet up in person. Nevertheless we still managed to improve on last years' robot and are even more looking forward to the competition for our last year!

## Major Milestones

| Date | Milestone | Page |
|---|---|---|
| 03.09.2023 | First Design Concept | 5 |
| 12.09.2023 | Main PCB schematic developed | 7 |
| 28.09.2023 | First main PCB design | 10 |
| 30.11.2023 | Robot design finished in CAD | 12 |
| 28.12.2023 | Robot assembled, first fully integrated tests | 22 |
| 24.02.2024 | First competition in Hamburg | 30 |

## April & June 2023

**Tasks completed**

- Problems with robot:
  - Unreliably electronics (power supply), possibly software and communication as well
  - Victim pick up does not work near walls (claws cannot close)
  - Victim pick up inaccurate (robot misses victims)
  - Victim neural network unreliable
  - Did not use victim storage tank: Repairs were difficult
- Other possible improvements:
  - Try PD controller for line following (to reduce overshooting)
  - Did not use victim storage tank
- What went well:
  - Raspberry Pi
  - Arduino as motor and sensor controller
  - victim collection from front
  - four-wheel drive: Same motors, same wheels

**Next steps**

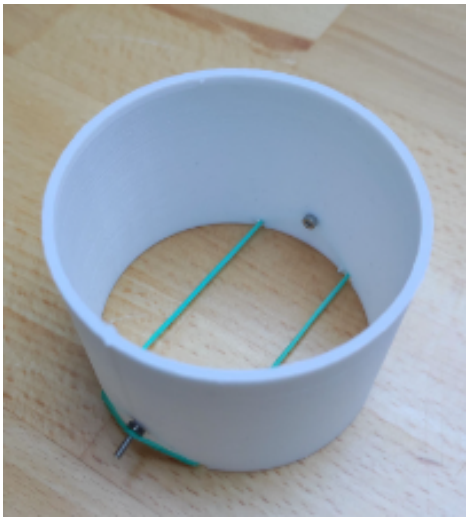
- Develop an alternative victim collection mechanism. Ideas:
  - Cup (see 2021 robot, not good for rescue kit, works very well with victims)
  - Large claws (see 2022 robot, very large, may not clear the camera)

# July 2023

## 17.07.2023

**Tasks completed**

- Prototyped new victim pick up mechanism (similar to 2021):

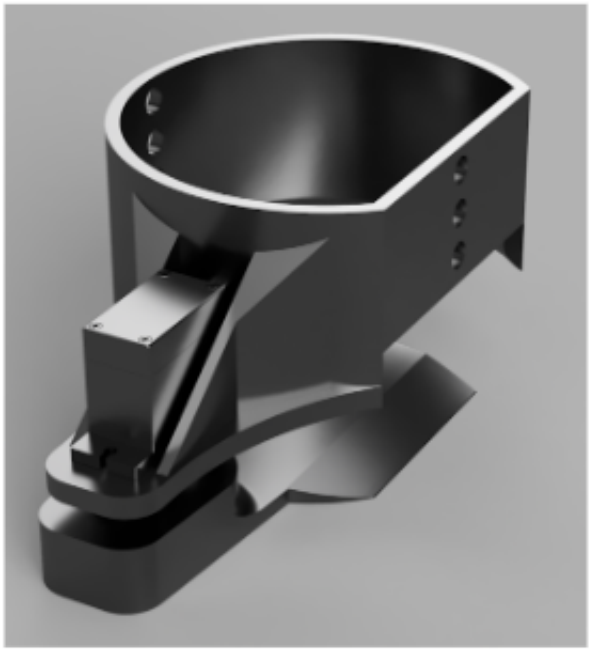


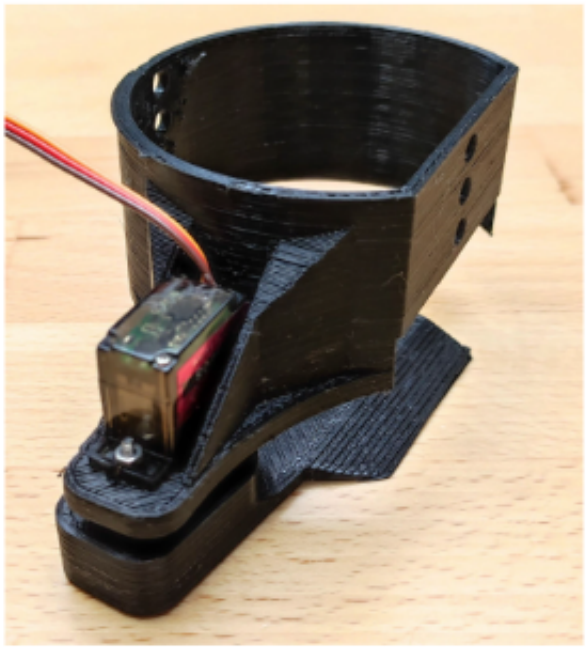
Problem: Picking up rescue kit does not work reliably

## 21.07.2023

**Tasks completed**

- New Solution: A narrow wedge is slid underneath the victims using a small servo motor





After some tweaking, this approach is also able to pick up the rescue kit reliably, as the wedge causes the kit to rotate. Gravity then does all the work for us, tipping over the kit onto the wedge

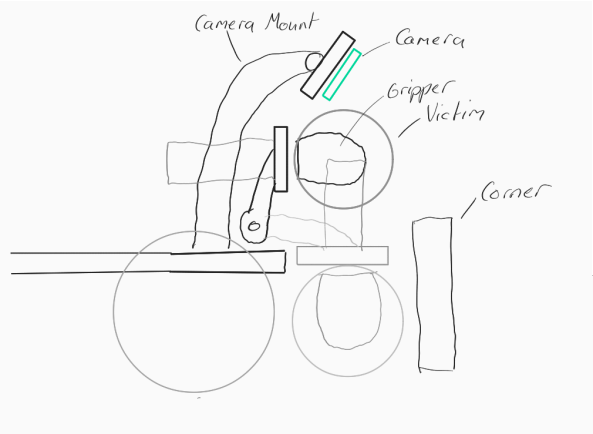
# September 2023

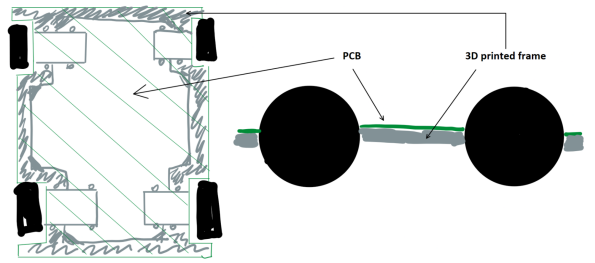
## 03.09.2023

First major design meeting

**Decisions**

- Main structure: Larger PCB with structure around the outside (less 3D-printing, less cables, straight-forward access to electronics)
- 14.8V nominal battery voltage, 4x 18650 Li-Ion batteries mounted directly to PCB from the bottom
- Elimination of voltage boost modules, 12V motors are powered by 14.8V using PWM
- Lengthening of wheel base by a few millimeters to accommodate batteries
- Wheel gauge kept the same
- Elimination of tank (one victim at a time, time savings would most likely be minimal)
- Same motors
- Same wheels and motor mounts
- Same camera
- Electronics:
  - Raspberry Pi
  - Arduino Nano
  - 2x 5V step down Module (Stable net for controllers and sensors, Unstable net for Servos)
- Gripper Concept (old one was too large to fit underneath the camera and could not pick up victims when there was a bumper underneath). Trying out a claw design (as in 2022).



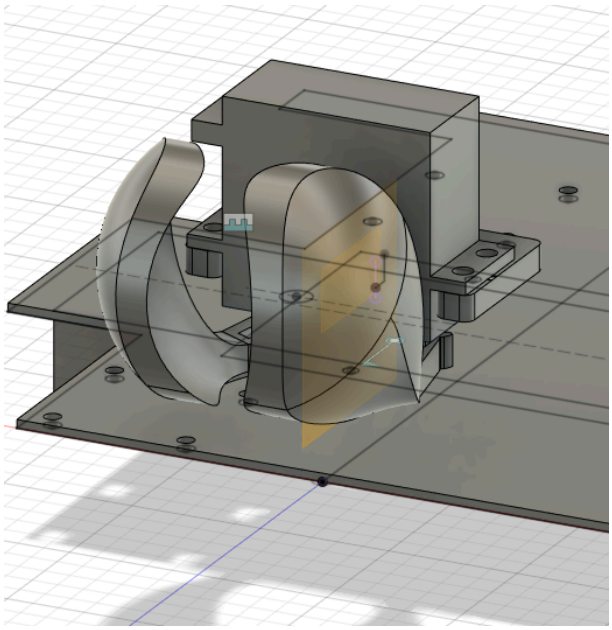




- Camera height may have to be adjusted to more than 130mm.
- Battery Configuration: four in rectangular formation, between wheels
- Handle: centrally along length of robot, mounted to camera mount and rear mount

## 04.09.2023

Started design of gripper/arm. Decided on using a large servo to move one half and move the other using gears.

Gripper design:



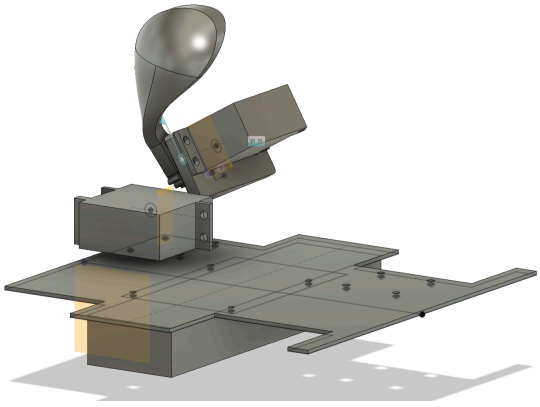
## 10.09.2023

Tried out a USB camera for detecting victims. Problem: works perfectly on Windows (using OpenCV, Google Meet and the default camera app) and Ubuntu (using OpenCV, default camera app, Google Meet and VLC) horribly slow on Raspberry Pi (no matter which program)

Tried out Arducam RasPi Camera Multiplexer to be able to use two RasPi cameras instead of the USB camera.
Problem: as the I2C pins on all our Pis are broken, we would need to control the multiplexer via an Arduino which uses 5V instead of 3V3.

-> While it would certainly be possible to control the multiplexer via an Arduino (e.g. by using a level shifter), we decided on using a servo motor to tilt the camera upwards when entering the rescue area. This solution might not be optimal, but it worked decently enough in 2023 and saves a lot of time that can be better spend on other parts of the robot (e.g. CAD or PCB design)

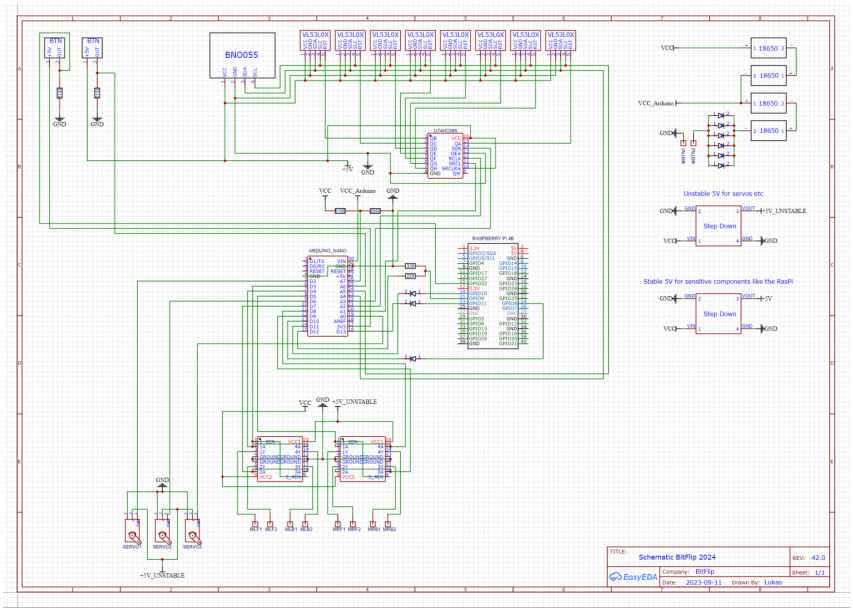
Current design:



Problem: as the gripping mechanism is quite high (causing the overall camera height to increase as well), we decided to move the whole mechanism a bit to one side:
This way we can roughly keep the camera position from last year (around 125mm from floor)

## 12.09.2023

First schematic for PCB:



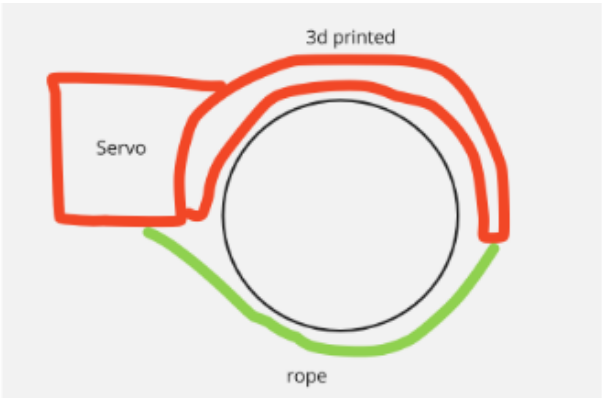

## 13.09.2023

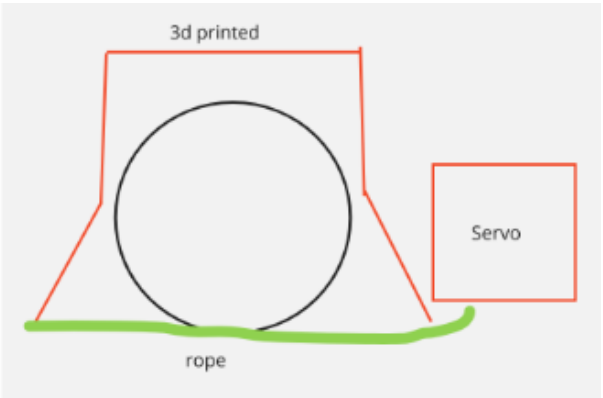
schematic review and evaluation, no mistakes found (replaced standard RasPi with custom one as we only need five GPIO pins)

## 14.09.2023

Discussion about current gripping mechanism:

- Victim collection positional tolerance is small which would require precise victim detection
- 3D printed gears are not that strong and can wear down over time
- there are small gaps between the teeth of the gears due to manufacturing limitations using a cheap 3D printer, resulting in a wobbling structure overall and therefore making it harder to pick up the victim (or requiring more torque which our servos can't deliver)
- mechanism uses a lot of space, therefore increasing camera height
- Potential alternative mechanisms:



## 15.09.2023

Finished chassis outline based on last year's dimensions, which we are keeping roughly the same:

## 19.09.2023

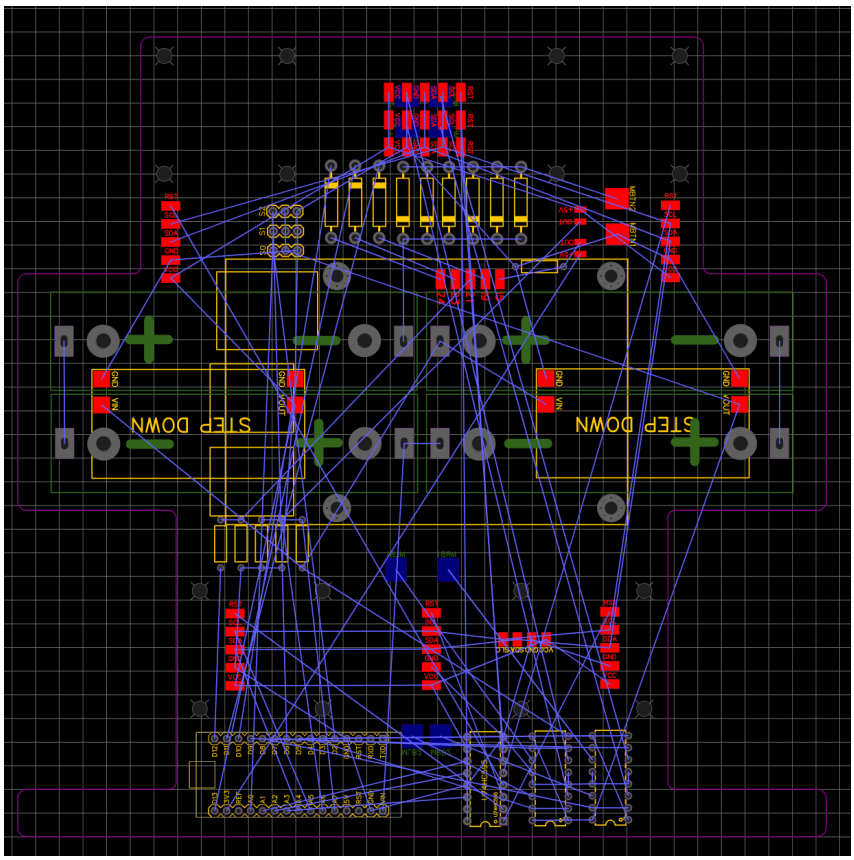
Draft 1 of the PCB design (excluding routing):

Pros:

- efficient placement of batteries and motors -> low center of gravity
- Soldering pads for sensors and motors are roughly placed where the components themselves will be located allowing for short cable connection

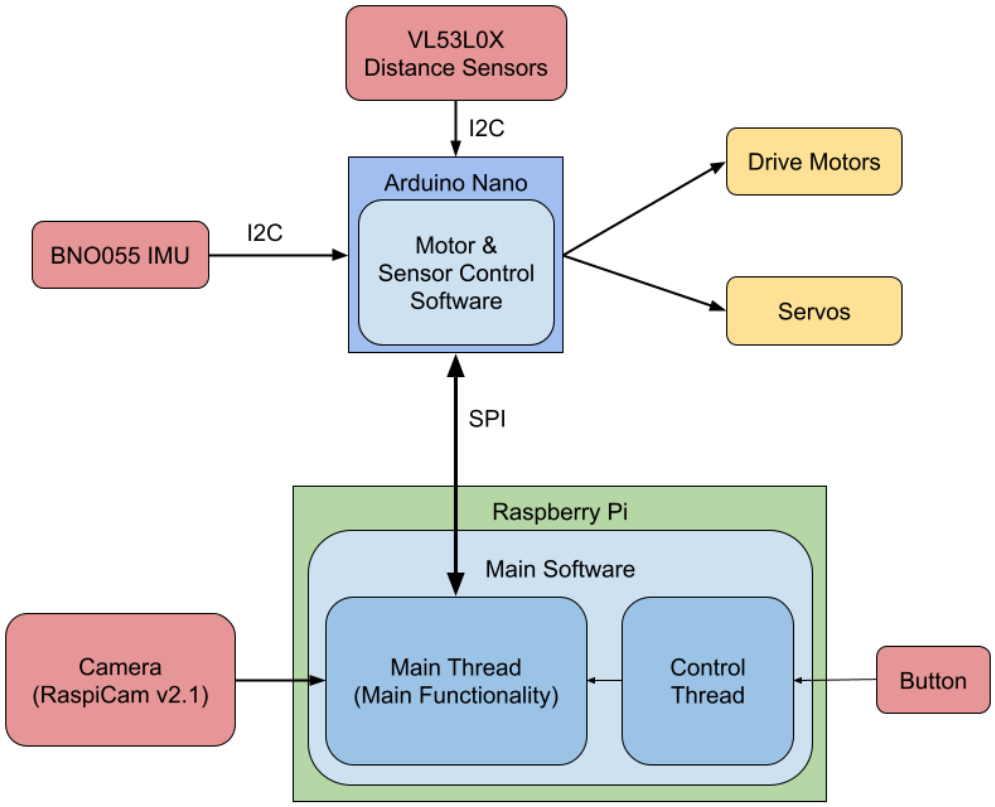
Cons:

- Awkward placement of Arduino (no actual alternative for current placement as batteries occupy center and the four motors occupy the front and back center)
- long traces from H-Bridges to motor pads could cause problems when routing traces
- long traces for SPI connection between RasPi and Arduino limit max transfer speed
- long traces for I2C lines can damage signal integrity



## 23.09.2023

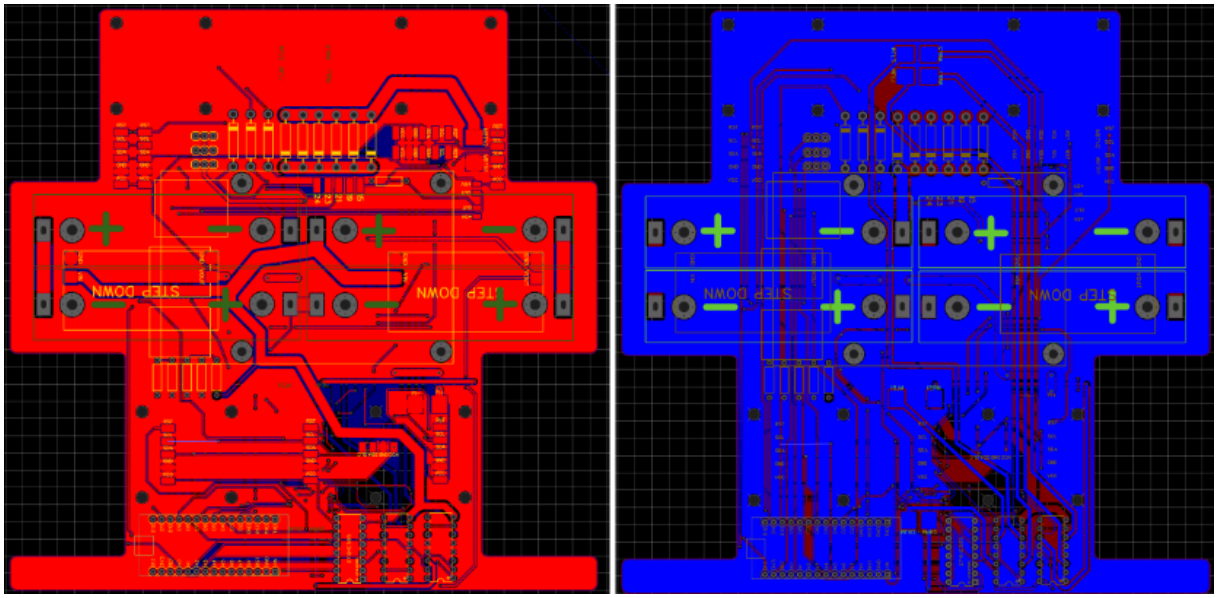
Prepared basic software and interface diagram:




Differences to last year: SPI instead of UART over USB. Main software split into two threads (maybe more threads for obstacle, silver, etc. later). Control Thread now continuously monitors the button, this way the robot can be stopped at any time.

## 28.09.2023

Finished first version of PCB routing:

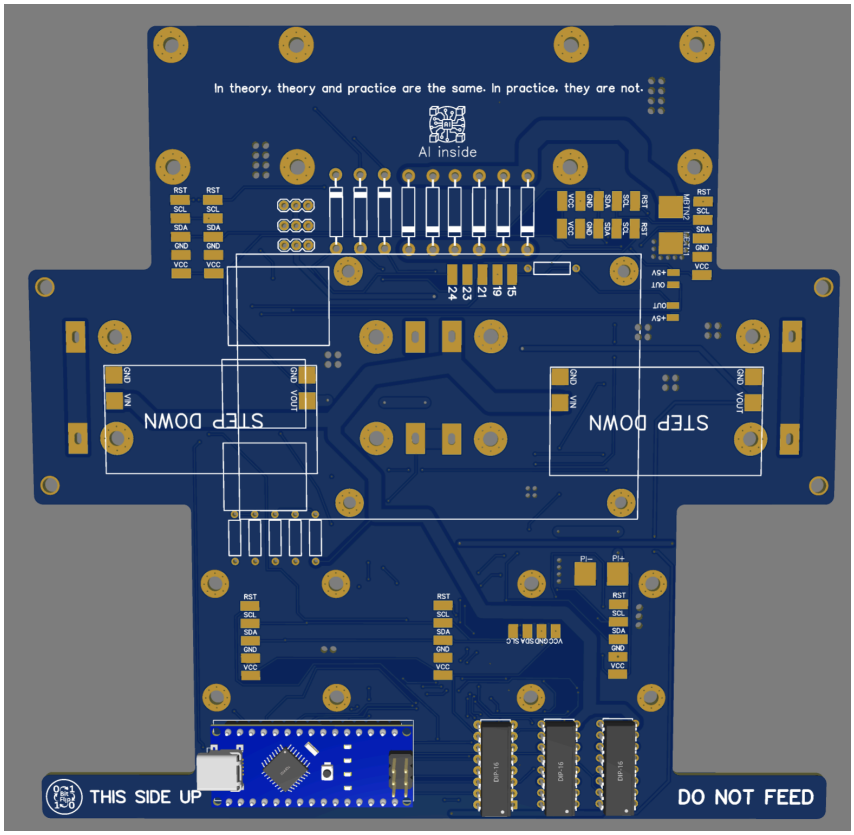


**ToDo**

- Check if GND copper area ensures sufficient GND connection everywhere
- Review

## 29.09.2023

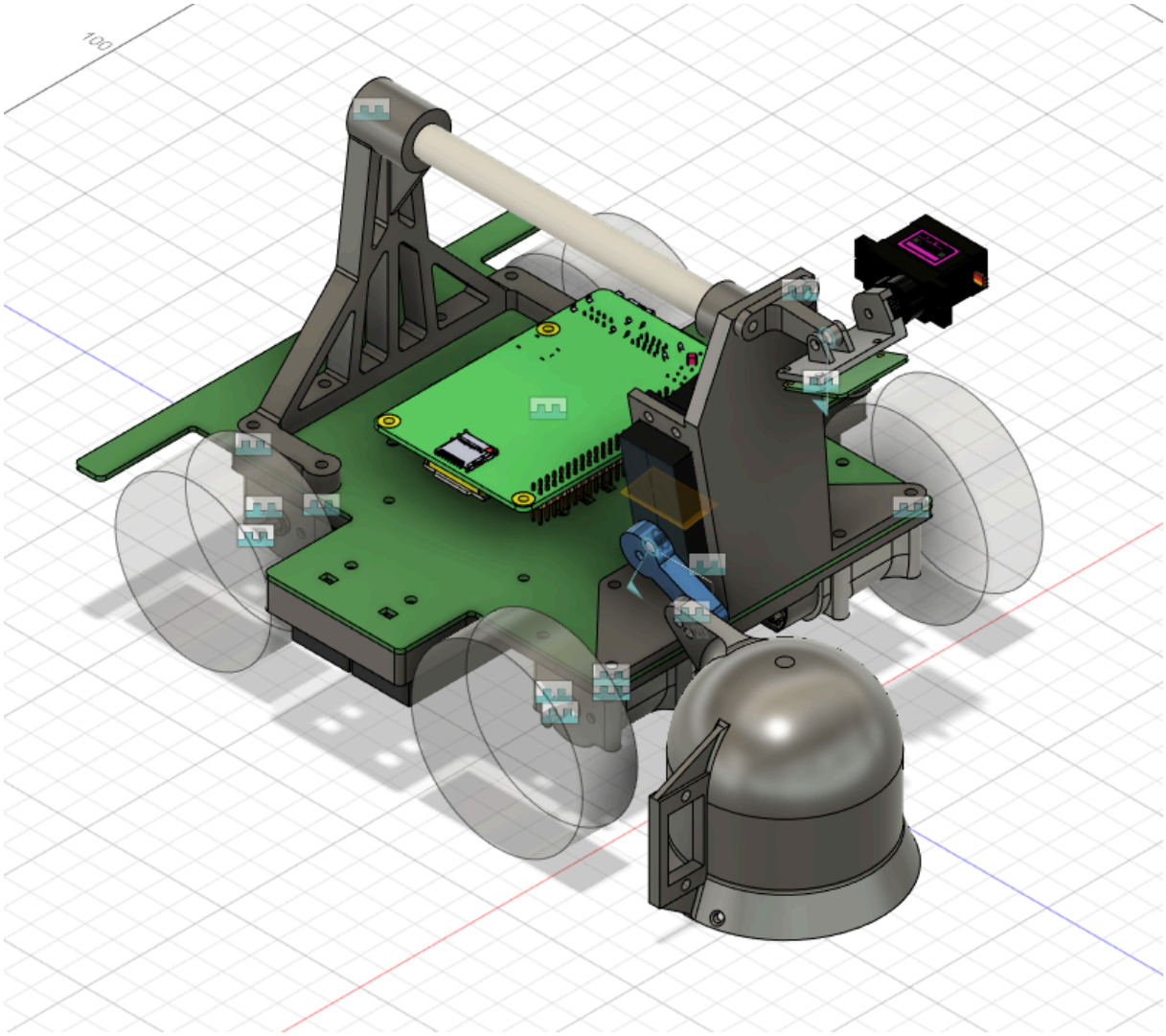
Improved GND copper areas, added texts, added vias around mounting holes to increase stability:



## 12.10.2023

Designed the following parts:

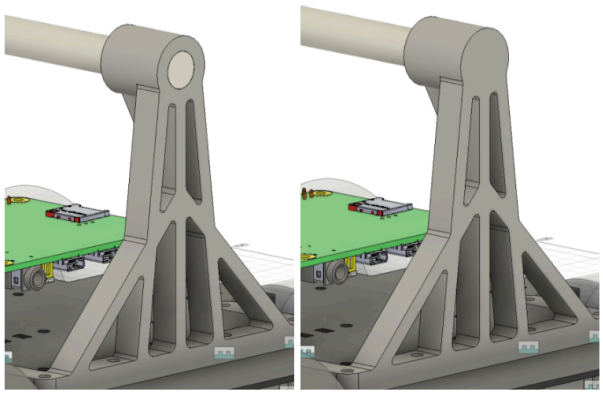
- Chassis
- Rear mount
- Front mount (rough draft)
- Camera Mount and pivot mechanism (rough draft)
- Arm



## 13.10.2023

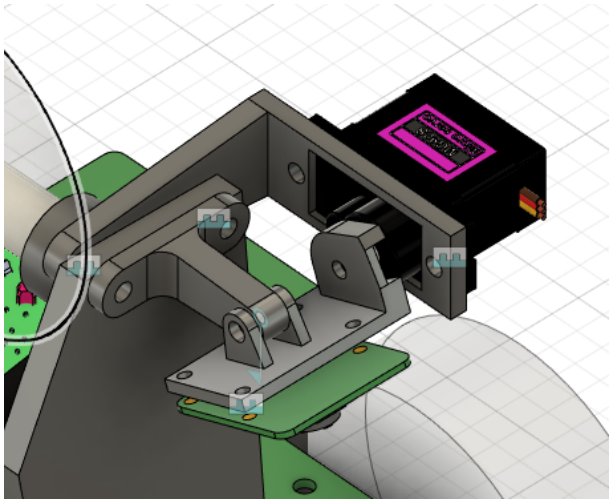
Design review

- Improved stability of carrying handle

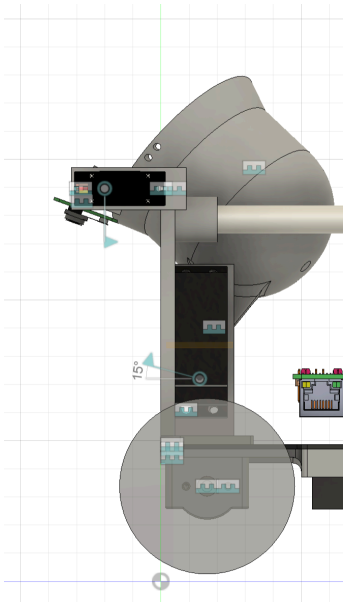


**ToDo**

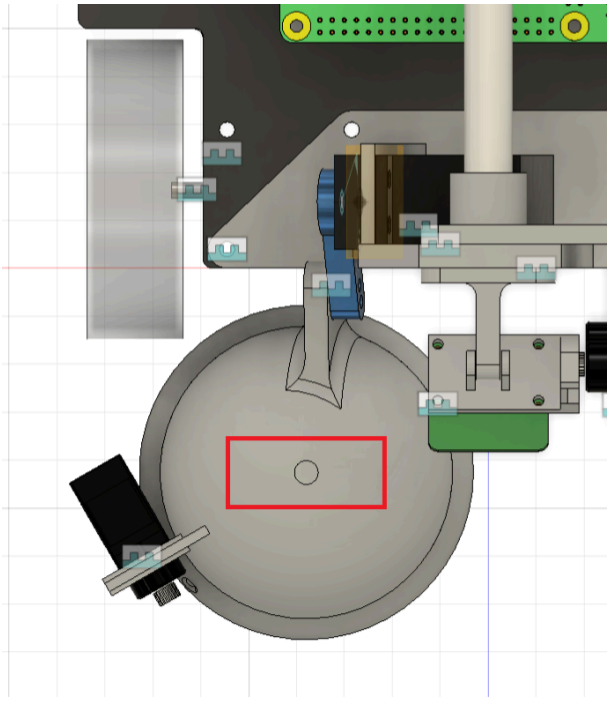
- increase overall strength of camera mount:



- add bracket in front of camera to prevent damage when driving against walls:



- place servo on top of container (to prevent damage):



## Development of testing schedule:

### Testing

**Electronics**

1. Check PCB before assembly (continuity testing & visual inspection)
2. Check component connections during assembly (continuity testing & short circuits)
   Assembly instructions:
   - 2.1. Main switch, 1 “power” diode, battery (check VIN of Arduino, check motors VIN)
   - 2.2. Arduino (Blink sketch)
   - 2.3. Right Step down module (check Servo Power supply = 5V, check 5V VIN of H-Bridges = 5V)
   - 2.4. Left Step down module (check PI+ pad = 5V, check VL53L0X VIN pads = 5V, check reset button pad = 5V, check VIN mux = 5V)
   - 2.5. All other diodes, RasPi (start up test)
   - 2.6. Gyro, VL53L0X
3. Full power up (all electronics installed and ready)
4. Load test (Raspberry Pi, Raspberry Pi with all actuators simultaneously)

**Integrated Tests (Low-level and mechanical)**

1. Read out button from Pi
2. Read out distance sensors from Arduino
3. Test motors from Arduino
4. Test servos from Arduino
5. Test RasPi <-> Arduino Protocol
   - 5.1. Test all possible commands and responses (normal operation)
   - 5.2. Test fail-safe mechanisms
   - 5.3. Test during full load operation
6. Test victim collection from Arduino
7. Test ramps, speed bumps

**Fully Integrated Tests (High-level software and mechanical)**

1. Identify problems from Bordeaux 2023
2. Line-following: Line, 270° corners
3. Intersections (T-intersection without markers!), T-intersections, circles (rectangles), dead-ends
4. Obstacle ☹
5. Silver (no false-positives)
6. Red (with smaller-than-allowed tapes)
7. Rescue:
   - 7.1. Silver (no false-negatives)
   - 7.2. Corner finding and unloading
   - 7.3. Victim identification and collection
   - 7.4. Exit
   - 7.5. Test with all tasks (strategy)
8. Test on (parts of) competition fields: RoboCupJunior CMS
   Difficulty level during testing >= Bordeaux 2023!
   - 8.1. Clearly note down problems and possible solutions

**Message**

| | | | | |
|---|---|---|---|---|
| **Raspberry Pi** | Command | Data | - | Parity bit |
| **Arduino** | - | - | Return data | Parity bit |

**Commands**

| Function | ID | Data | | Return | Length | Comments |
|---|---|---|---|---|---|---|
| drive | 0x01 | Left speed | Right speed | | 16 | |
| | | -128 to 127 | -128 to 127 | | | |
| | | int8 | int8 | | | |
| servo | 0x03 | Servo position [degrees] | Stall | | 9 | Servo position = 0: No servo movement |
| | | 0 to 255 | True or false | | | |
| | | uint8 | boolean | | | |
| sensor | 0x04 | Sensor id | | Sensor value | | See Sensors section |
| | | 1 to 11 | | 0 to 65.535 | | |
| | | uint8 | | uint16 | | |
| turn | 0x05 | Angle [mrad] | | Finished | | |
| | | −6.283 to 6.283 | | True or false | | |
| | | int16 | | boolean | | |

Boolean size is one bit, true is 1, false is 0.

**Sensors**

| ID | Sensor | |
|---|---|---|
| 1 | *Reserved for touch sensor* | |
| 2 | Front | Distance |
| 3 | Left front | |
| 4 | Left rear | |
| 5 | Rear left | |
| 6 | Rear right | |

## 19.10.2023

Implemented camera video capture and image show (camera.c and vision.c). Tested on Ubuntu VM. Functions implemented:

- Image resizing
- Image cropping
- Saving image as PNG
- Color to grayscale
- Bresenham's line drawing algorithm
- Start camera capture (starts thread)
- Stop camera capture
- grab and copy frame
- Create window for image display (starts thread)
- Show image in window
- Destroy window

Problems which don't necessarily need to be fixed:

- Only one window at a time
- No free image resizing
- Line drawing only with 1px width

## 20.10.2023

Changes to Protocol (communication is in full bytes, so parity bit requires some more thought. No parity check for now) Started implementing robot.c. Started implementing Arduino firmware. SPI communication should be fully functional.

**All untested!**

Current plan is to have the robot fully assembled and partially tested after christmas. For details see Testing Schedule.

## 22.10.2023

Trained mobilenet_v2 model on victim data (instead of our own custom model). Seems to be very accurate and fast with very little training and data ( 1600 images for now). TODO: Try inference with this model on the Pi (or at least on Ubuntu in C).

## 26.10.2023

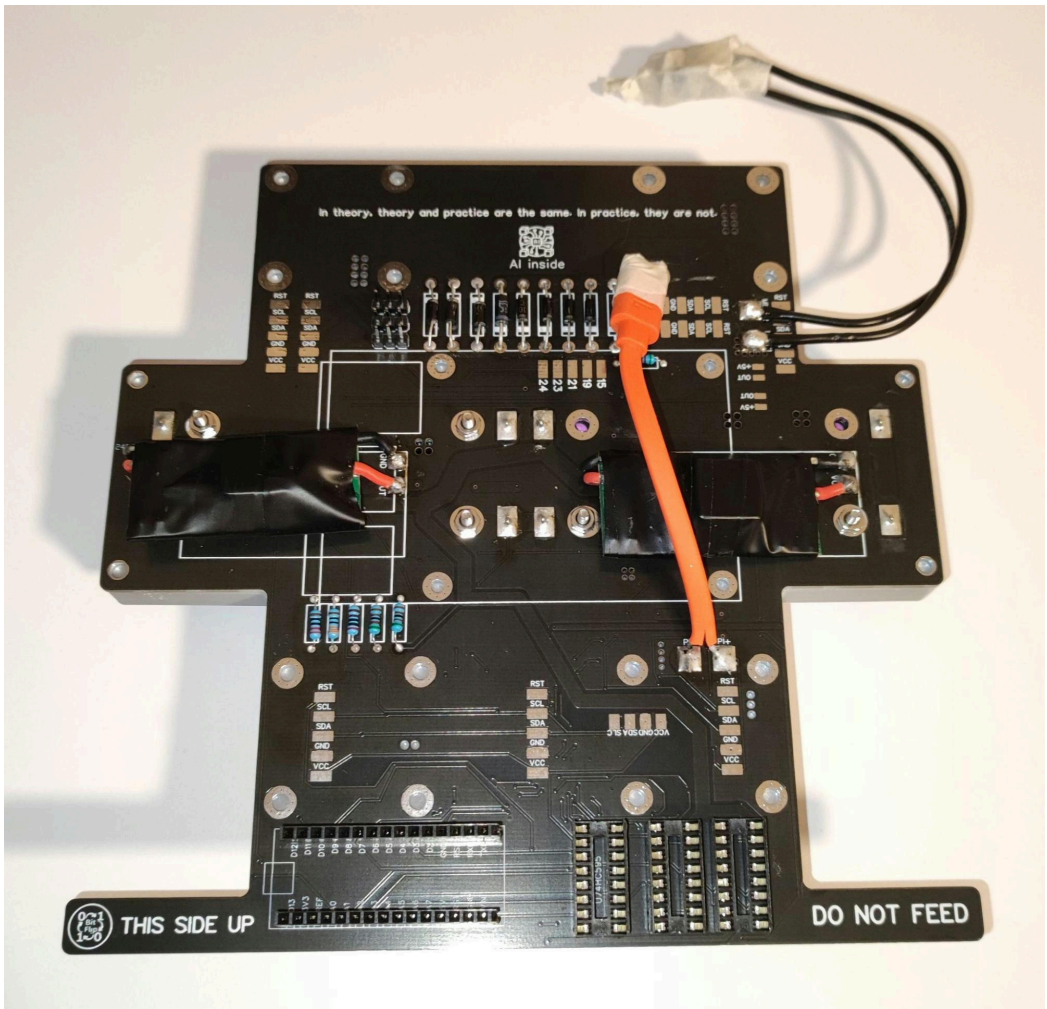
Finalized chassis design. Chassis is now ready for printing and motor mounts can be reused from last year. After assembly, we can start testing the electronics and start driving.

## 28.10.2023

Printed chassis, started assembling PCB

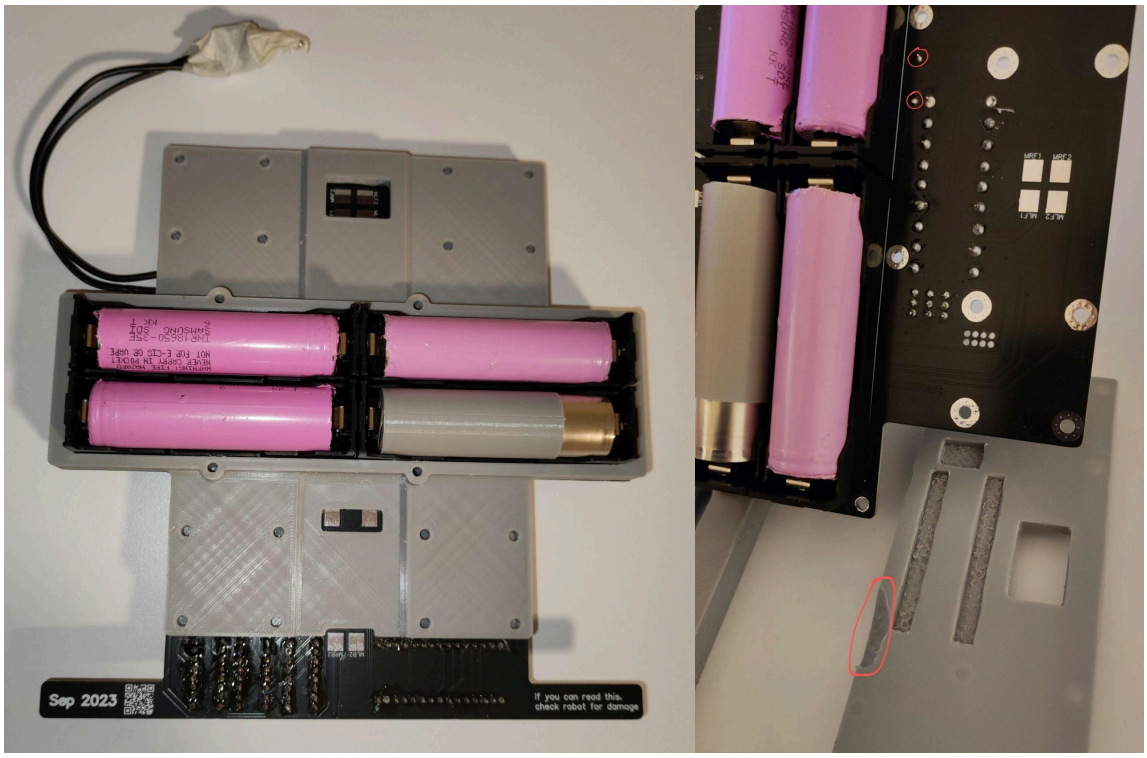
## 29.10.2023

Continued PCB assembly, most important part done. Ready for subsystem testing



## 30.10.2023

Assembled chassis to PCB, looks good so far (Fig. 1), but we forgot to add two small holes for resistor legs. Fixed using soldering iron (Fig. 2)




"Smoke test" was successful, following the test procedure. Everything works as expected, starting with rudimentary firmware development.

## 31.10.2023

First working function of new firmware:

```
--- DEBUGGING WINDOW ---
Battery Voltage: 15.50V
Battery Voltage: 15.48V
Battery Voltage: 15.50V
Battery Voltage: 15.50V
Battery Voltage: 15.50V
```

Found hardware Bug: when Arduino is inserted into PCB and main switch is off, the Step Down Modules still receive an input voltage of around 7V and therefore output 5V.
Same behavior can be achieved by inserting a resistor (tested with 330 Ohm) between VIN and GND of the Arduino Current suspect: Voltage divider for measuring battery voltage with the Arduino as second resistor provides a direct path to the VIN pin of the Step Down Modules.

Edit: Removing the according resistor from the PCB entirely did not resolve the issue. Emergency meeting with Sven tomorrow

## 01.11.2023

The problem is caused by the fact that we power the Arduino via 2x 18650 cells out of our 4x 18650 in series. To power off the entire robot, we cut off the GND connection between the common GND on the PCB and the battery pack using a standard switch. The problem with this procedure is that we still apply 2x 3.7V to the VIN of the Arduino and 4x 3.7V to the VIN of the Step Down Modules (compared to the absolute GND of the batteries). When we now insert an Arduino, the Arduino connects the VIN pin and the GND pin on the according header row together (with a high resistance though) which means the GND plane of the PCB now has the same voltage potential as 2x 18650 cells (again compared to the absolute GND of the batteries). But since there are still 16.8 absolute Volts being applied to the Step Down Modules, there is a relative voltage potential of around 7.4V between the GND plane of the PCB and the VIN of the Modules causing the Modules to "work properly".

Possible solutions to fix the problem:

1. Adding diodes (no idea if this could work, requires lots of time)
2. Cutting the VIN trace of the Arduino and…

a) power the Arduino from 5V (output of the Step Downs) via the

   i) VIN pin (not recommended by datasheet, but should work according to some forums. Only problem is that the reference voltage would be less than 5V, therefore making analogRead inaccurate and digitalWrite would output less than 5V)
   ii) 5V pin (bypasses protection circuit of the Arduino, if you plug in USB simultaneously you destroy the Arduino)

b) power the Arduino via a USB cable from the RasPi (inconvenient as you have to plug in and out whenever you want to upload new code from the computer. Subsystem testing becomes difficult as Arduino would always require the RasPi would not work without him)
c) power the Arduino with the voltage of all four 18650 (not recommended, Arduino might get hot and lifespan decreases significantly)
d) Add another switch (would work flawlessly, but requires the user to toggle two switches to power on and off the robot. Not very convenient and increases chances of human error)

## 02.11.2023

Settled with option 2ai, analogRead for battery voltage could be corrected with a factor, digitalWrite now outputs 2V which is far too low.
We now opt for solution 2aii and currently await a response from the Arduino support as some Nano boards have built in protection to not fry your board when plugged into your PC and powered via the 5V pin simultaneously.

The support reached back to us and clarified that the Nano could indeed be powered via the 5V and USB at the same time. First tests were successful (meaning the Arduino survived), but the Step Down Modules produce an extremely high oscillating sound, so we will try to power off the main PCB before uploading new code to the Arduino just to be on the safe side.

New problem: when adding the H-Bridges to the PCB and letting the Arduino execute a simple motor test script, they get extremely hot, so there might be yet another hardware bug needing to be resolved.

## 04.11.2023

Retrained TensorFlow Lite victim detection model with 500 randomly selected images.

Tested new model from C with four images (under tests/tflite_c_api_test. Not yet integrated into the main program). The model is very accurate with detecting victims and finding their positions (just a few pixels off the real center). Of the 6 victims in the (very old) test images one dead was misidentified as alive.

Model takes 320x320, trained and tested with resized 160x120 images.

Maybe switch to 320x240 images to improve detection of victims that are further away (new training data required for this to have any effect). Requires no software or model changes (In theory, the model can even be trained with images of mixed resolution).

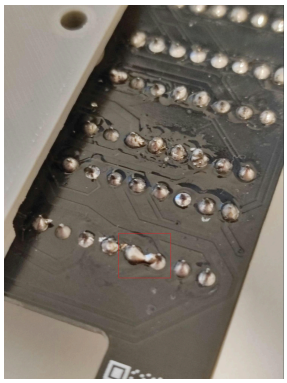
Current plan is to keep the other two models as they were, without retraining. TODO: Invoke Silver and Corner models from C. Implement in the main program.

## 07.11.2023

Motors not working, needs further investigation. Voltages on IC sockets seem okay

## 08.11.2023

Finally found (obvious) problem with motors after one blown-up H-Bridge:



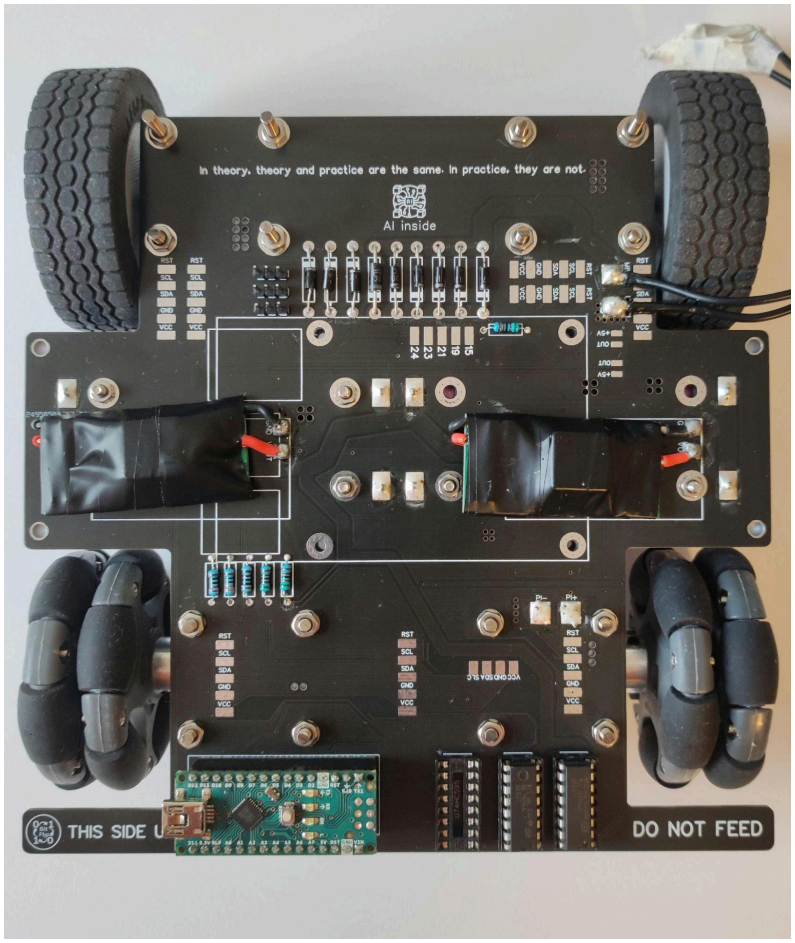
The output of the motor was directly connected to GND causing a massive current flow. New problem: motors receive more than 12V, probably problem with duty cycle calculation

## 10.11.2023

Fixed max motor voltage by adding a correction factor. TODO: Test with different battery voltages.

## 12.11.2023

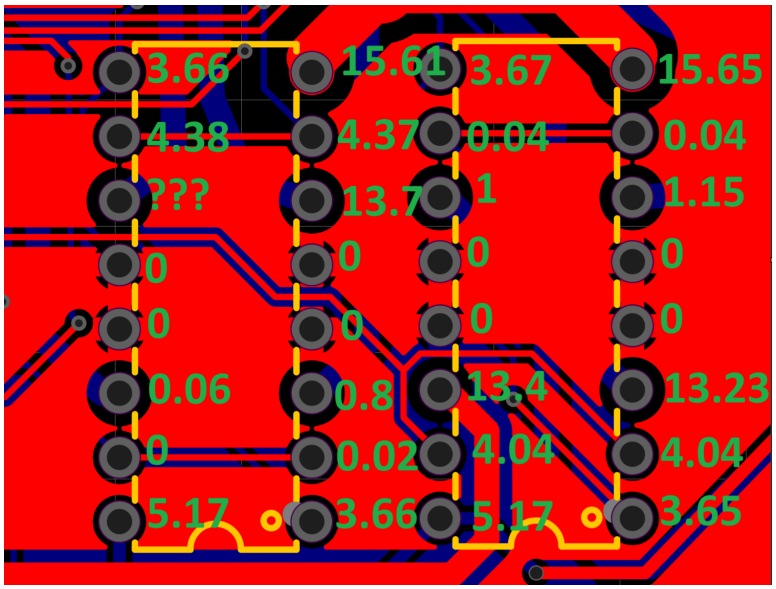
Added motors to the PCB:




Working so far, but the right side turns slower than the left side. Voltage left: 12V, voltage right: 11.6V

## 14.11.2023

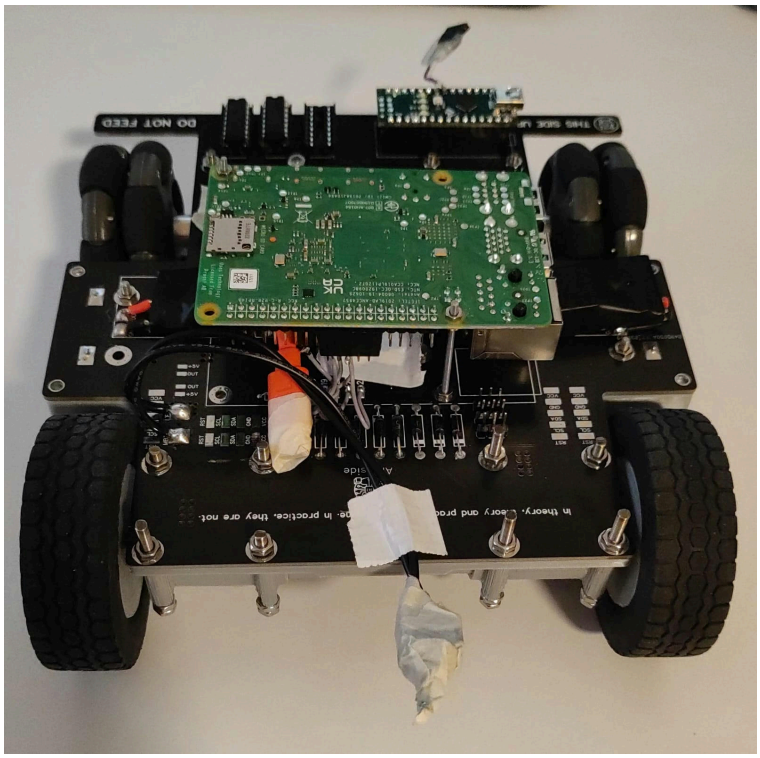
Measured voltage when turning motors forward at supposedly full speed:



Fried an Arduino because I shortened the ??? pin with the neighboring digital pin of the Arduino. The voltage there should be around 13.7V
Note to future self: Before checking anything else, try using another Arduino. Motors working properly now (one side still marginal faster, but negligible for now), overall voltage of motors is > 12.5V
Servos also working as expected

## 19.11.2023

RasPi is ready for testing:

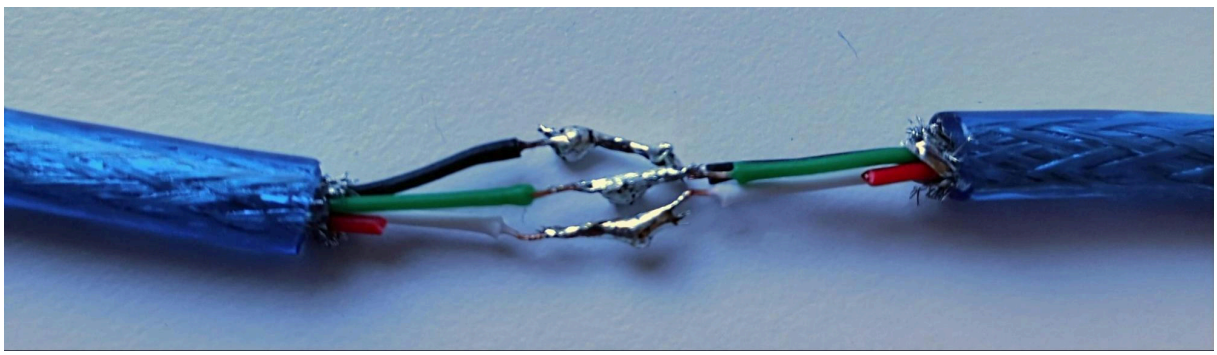


## 20.11.2023

SPI test, not successful. Problem: ???

## 25.11.2023

Cut VCC line of USB cable so we can connect Arduino to PC safely while powering Arduino via PCB



## 26.11.2023

Further SPI testing, somehow Arduino receives weird stuff instead of actual data (once a bit is set to 1 it stays on one forever):

```
0  = 00000000 (0)
2  = 00000010 (1 - 00000001)
6  = 00000110 (3 - 00000011 : 4 - 00000100)
E  = 00001110 (5 - 00000101 : 7 - 00000111)
1E = 00011110 (8 - 00001000 : 15 - 00001111)
3E = 00111110
7E = 01111110

0
4  - 00000100
C  - 00001100
1C - 00011100
3C - 00111100
```

Hardware check okay, all pins of RasPi working properly

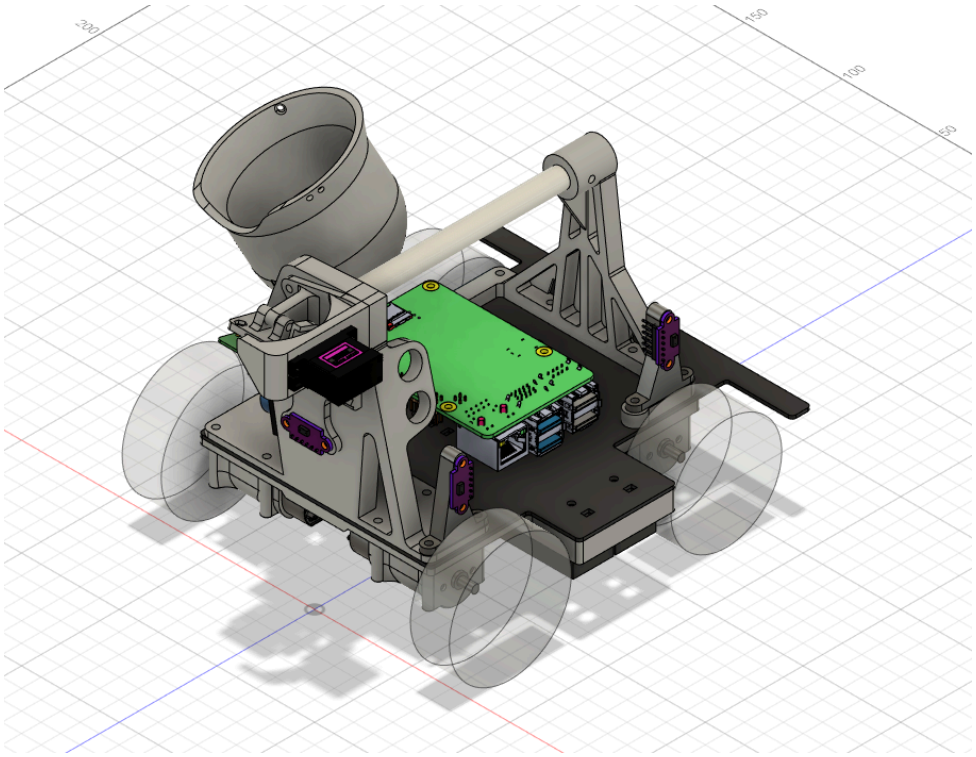
Since SPI PCB traces are roughly 15cm long, there seems to be quite a high capacitance so when RasPi sends SPI data, the line stays high for  6ms. Decided to switch to USB, motor control via RasPi working

## 30.11.2023

Finished robot design in Fusion.

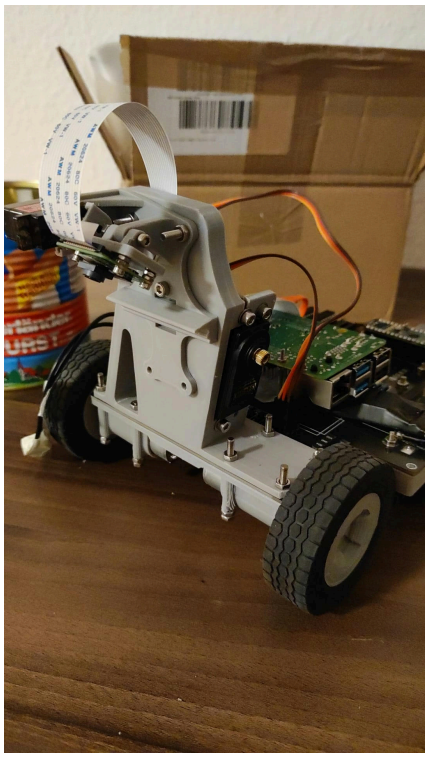
- Improved camera mount (increased strength and added bracket for protection)
- Added VL53L0X mounts around robot
- Front mount with VL53L0X, light and camera servo
- Reset button and main power button in the front
- Cut out part of the victim collection cup
- Added holes for handle fastening (if necessary)
- Increased victim collection cup clearance

Ready for printing, the handle is not required for now.



## 05.12.2023

Printed and assembled camera mount. Flipped design for facilitated cable management and better ergonomics (if you're not a lefty, the home position of the gripper is quite annoying when carrying the robot)



## 06.12.2023

Implemented Lukas' requested changes in CAD:

- Small cutout in front mount for PCB-mounted diodes

- Camera servo mount too fragile, increased width and depth
- Mounted camera servo has different dimensions than anticipated, moved servo further out
- Arm servo mount too fragile, increased width and depth

Final aluminum handle dimensions are unclear. Have to measure available sizes. Settled on 10mm diameter.

## 07.12.2023

Using fixed-size static images instead of heap allocated images. Implemented main.c, line following and green.

## 08.12.2023

First main program build. Test image from RasPi camera:

First test of line following. Hard to tell how well it works without proper light installed. Line angle algorithm and green detection work.

Next steps for proper testing: Fix motor issues and install light.

Changed handle dimensions in CAD to 12mm. Found possible place for IMU below the arm at the front.

## 09.12.2023

Robot still drives to the right side when trying to drive forward. Debugging notes:

Visual control: no abnormalities, motors are all equally difficult to turn by hand
With pwm 127, motors on the left get 11.5V, but motors on the right only 10.5V. This explains why the robot is driving to one side.
Voltage pwm pin left: 2.53V Voltage pwm pin right: 2.53V (so both are correct, as pwm 127 is half of 5V) -> Pwm pins seem to be OK

Possible other source of error: H bridges get different VCC for the motors → both get 15.5V, so this cannot be the cause

Possible other source of error: The motor direction pins do not go completely LOW/HIGH
HIGH: 5.1V, 5.1V, 5.1V, 5.1V
LOW: 0.00V, 0.00V, 0.00V, 0.00V
→ Pins seem okay

Possible other source of error: Resistance of one motor cable is very high, so that 1V drops via cable. Resistance from output H bridge to motor input < 0.6 Ohm for all 8 cables (for 1V to drop

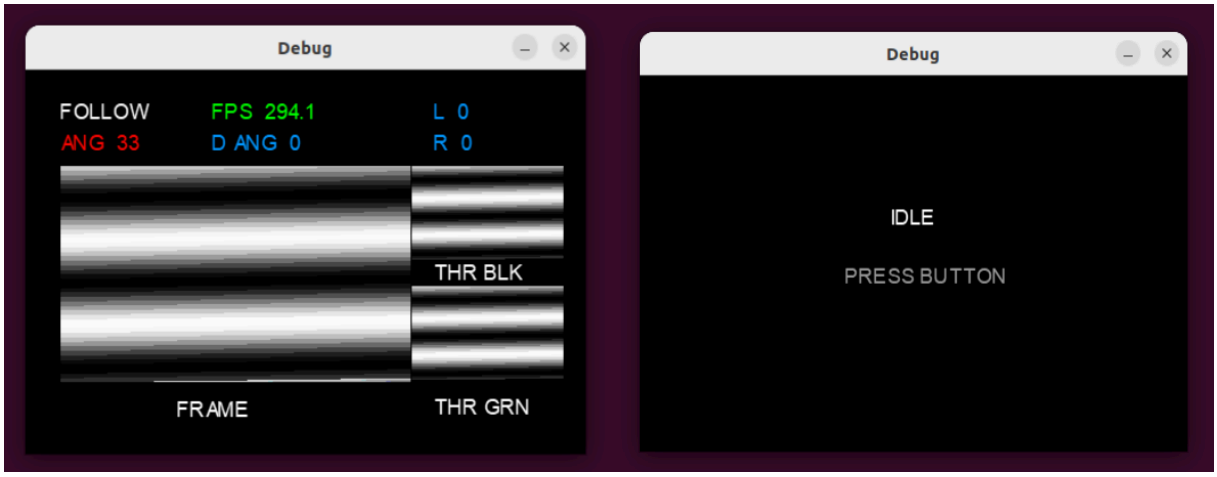
via cable, approx. 20A would have to be drawn at 0.6 Ohm). Trace has almost no resistance in all cases. The 0.6 ohms seem to come from the solder connection

Possible other source of error: broken H bridge After swapping the two H bridges, the robot still drives askew. And in the same direction as before → H Bridges can't be the cause

Advice from Sven: soldering right motors to left motor pads and vice versa so we can say for certain whether the motors or the electronics create the problem

## 10.12.2023

Created debug UI system using SDL2 and tested on Ubuntu virtual machine:




Implemented functions:

- Window creation (fixed size)
- Window close upon pressing 'q' key (essential to stop program when in fullscreen)
- Text rendering (bare metal, one font and one font size)
- Image rendering (pass image pointer and specify coordinates and it gets drawn inside the window)
- Modes implemented (see screenshots)
- Idle mode
- Line follow mode
- Passing numbers to display (see fps and angle in screenshot)

Up next:

- Line Obstacle mode (shows obstacle cutout and maybe number of pixels)
- Rescue debug mode (shows captured images and NN outputs, as well as some other variables such as captured victims etc.)
- Draw lines
- Draw circles or boxes (can be made out of lines)
- Integrate into main program

## 11.12.2023

Added line and rectangle drawing functions. Added line obstacle and rescue screens.

Integrated into main program

## 15.12.2023

Tested Sensor command and answer over serial. Tested Servo over serial.
Tested display on Pi.
Motors are wired incorrectly and light is missing, so no Line testing, but software works.

TODO: add epic screenshot from debug UI system

## 16.12.2023

Added capture.py script to capture images for NN training.

Increased strength of a few parts in CAD.

Implemented algorithm to find "real" line angle in python (normal line angle algorithm will point towards a vertical line if it is not in the center of the image, even though the real line angle of a vertical line is zero).
For use after gaps: When a gap is detected, we can drive backwards a few cm and then turn this gap line angle to align the robot with the gap

## 17.12.2023

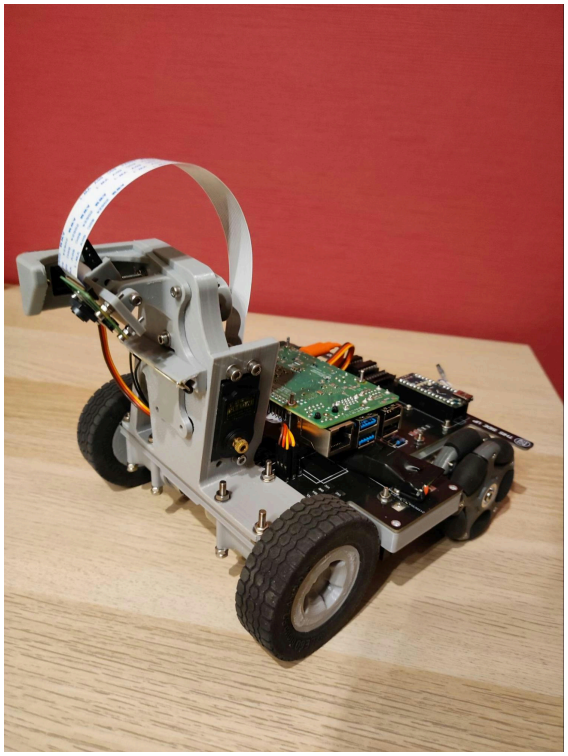
Tried to hardware debug why one side of the robot turns significantly slower. Swapped right and left motors, now the slower side is on the left (previously it was on right). So there shouldn't be a problem with the PCB, but rather with the motors themselves. Removed motors altogether, to resolder everything from scratch

## 23.12.2023

Started printing new components Sven redesigned

## 26.12.2023

Debugged motors further, so far no progress despite new wires and solder connections.
Assembled new parts:



## 27.12.2023

Meeting at Lukas'. Printed VL53L0X mounts.

Fixed motor issues: Pin 3 and 5 have different PWM frequencies ( 400Hz,  900Hz). Due to our PCB design and its poor high-frequency properties it caused the motors to receive different voltages. Fix: Move motors from Pin 3 to Pin 6, which has the same frequency as Pin 5.

Tested line following: Tuned parameters. Much less oscillation and closer line following than 2023. Due to higher control gains T-intersection did not work. Fix: extra weight ("center weight") which gives straight lines more weight (Weight function: 0.5*exp(-angle^2)+0.5).

Tested green: Movement values may need to be adjusted (almost no proper approaching before turning). Camera exposure too high: Changed auto_exposure_bias from 12 to 10. Green color values may need adjustment.

Turning without gyro for now. Captured  200 images for victims NN

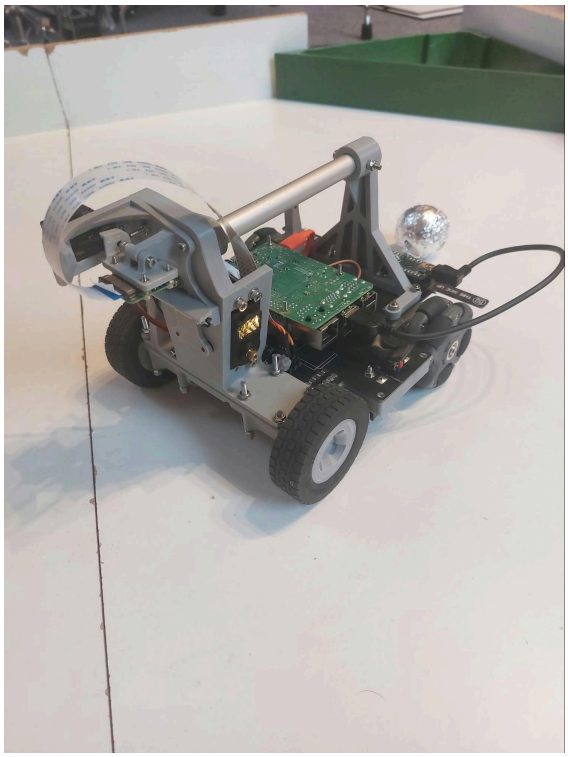
TODO list for the near future:

- Silver NN
- Wiggle
- Capture images for rescue
- Rescue

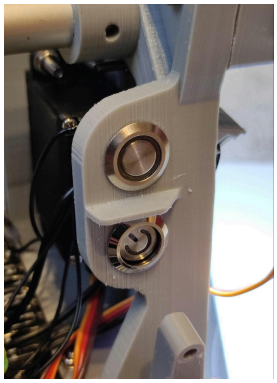
## 28.12.2023

Tested cup printed overnight: forgot to mirror
Minor improvements for cup design:

- servo dimensions
- rope now divides 2D circle area exactly in half which should facilitate victim pick up



Added Power button and start button:

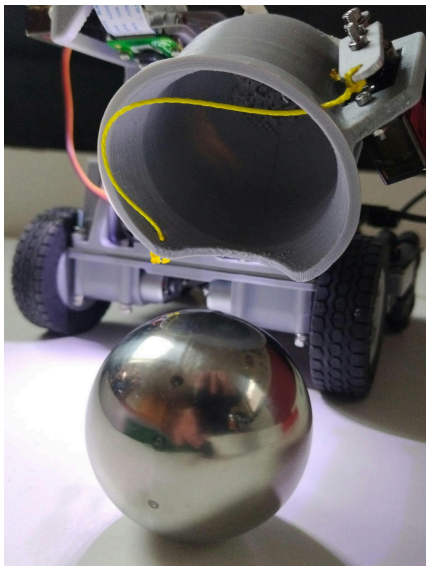


Designed epic idle screen for RasPi:


Line working ludicrously good (extreme speed)

## 30.12.2023

Added cup for victim pickup:



Obstacle working, minor improvements to line (improved intersection algorithm)

Released season 2024 Trailer on YouTube

Isolated old 18650 batteries to enhance safety:

## 09.01.2024

Silver troubleshooting: Images of silver that I took with the capture.py script contain some of the robot, which probably confuses the NN. This is not the case for line following, because the camera is in high frame rate mode and crops the image. I have cropped the images to remove the robot and have re-trained the model.

The NN performed poorly on the test images from last year (however, the bad model performed very well, so this is probably not an indication for real performance). Need to test deployed on the Pi with Lukas.

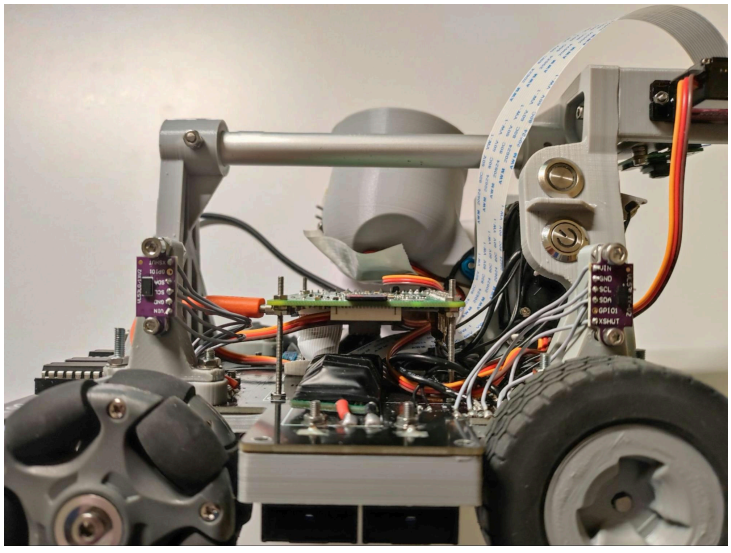
## 10.01.2024

Fixed Silver issues: Silver NN was fed only blue channel instead of grayscale frame. Works very well now. Made better capture.py script (now with different resolutions!).

Fixed issue with distance sensor: Wrong I2C address.

## 13.01.2024

Added side distance sensors:



## 14.01.2024

Distance sensor multiplexing works

```
Front: 125              Right front: 81         Right back: 213
Front: 129              Right front: 79         Right back: 211
Front: 128              Right front: 79         Right back: 209
Front: 129              Right front: 81         Right back: 210
Front: 127              Right front: 79         Right back: 212
Front: 128              Right front: 81         Right back: 214
Front: 128              Right front: 81         Right back: 213
Front: 126              Right front: 80         Right back: 218
Front: 130              Right front: 79         Right back: 211
Front: 129              Right front: 80         Right back: 213
Front: 128              Right front: 79         Right back: 209
```

## [UNIVERSITY EXAMS, NO DEVELOPMENT]

## 19.02.2024

Implemented victim delivery. We can re-use the corner NN from last season.
Implemented turning away from walls when entering the evac zone.

## 22.02.2024

- Fixed distance sensors: The Pi always read the previous instead of the current value
- Improved obstacle motor and distance values
- Added obstacle alignment with front distance sensor
- Implemented victim collection and delivery
- Corner detection using simple color detection instead of neural net (We have to test how robust this approach is, as we chose an NN previously because of reliability problems)

There are problems with the camera exposure. Quick fix is stopping and restarting camera capture, but does not work all the time. Auto exposure should be enabled, but exposure is not adjusted during capture. In previous years and now this caused problems detecting victims and corners, and even during line following (see entry from 27.12.2023).

There are also some issues with detecting the entrance to the evac zone (around one in ten detections does not work, there are also some false positives at the beginning of gaps and with Lukas' wooden bumpers). Possible causes:

- The NN is too slow for the speed we are driving (also may be caused by the distance sensor readings being slow)
- The NN lacks data (the entrance we tested with is not very representative of a competition entrance, so this may not turn out to be a problem)

## 23.02.2024

Improved corner and victim pick up. Last minute fixes before the Qualification tournament in Hamburg. Arduino broke (PWM pin), replaced by Nano with CH340.

Problem with Arm servo: The position changes without changes to the program. Did not observe this problem with Cam or String servo. Arm Servo does not feel very good when moved by hand, replace soon.

## 24.02.2024

First day of competition.
Distance sensor multiplexer broke. Replacement immediately broke as well → Cause in PCB, not identified. Using one sensor without multiplexing instead.

Still there is a problem with the distance sensor. Tried continuous mode: Does not help.

Implemented find_center and improved various things in Rescue.

Runs went well, currently on first place.

## 25.02.2024

Temporarily fixed distance issues by resetting the distance sensor each time the program starts and also restarting the entire program when the button is pressed.

find_center now turns 4s right and 4s left instead of just right (when standing right next to a wall, we can easily get stuck by just turning to one side).

Basic find_exit implemented by using front distance sensor.

Last runs also went well, finished first place and qualified for German Open.

Problems identified during competition:
- NN living/dead and false positives at wall and from shadows
- Corner approach not very precise
- Fix distance sensors
- Entrance detection only worked on the third try when going down the ramp
- Arm servo changes position without changes to code
- General observation: robot works flawlessy at home, but not on competition day. Probably caused by the fact that we dont't test enough at home!

Possible performance improvements we are looking at now:
- Use two cameras with Raspberry Pi 5: One looking down for line following and final victim approach. One looking forward for victims and corners
- Improve victim approach. Very slow (not cool enough for German Opens)
- Train NN for longer (should help with false-positives and far-away victims)

## 26.02.2024

Cut videos from Hamburg and uploaded to YouTube

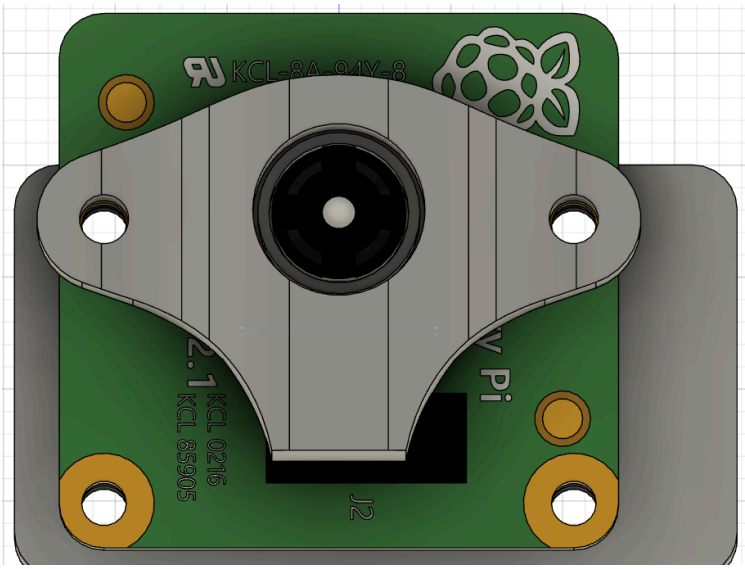
We also did further performance evaluations based on the video material (no more issues found)

## 05.03.2024

Worked on exit, general logic done. Waiting for Sven to finish CV tasks (detecting corner and exit)

## 06.03.2024

Designed camera bracket to keep camera lens and sunny connector in place:




## 07.03.2024

Sven wrote most parts of the TDP. Lukas has to write the section about electronics.

## 08.03.2024

Modified and printed camera bracket (several times…)
Nearly finished exit. Todo: last fixes with Sven

## 12.03.2024

Started to convert Engineering Journal on Google Docs (simple spreadsheet in endless paper format) to proper document using typst.

## 16.03.2024

Shot images for poster:

### 20.03.2024
Discussed exit strategy, minor exit improvements (actually checking for silver/black instead of guessing)
Corner approach: removed incrementally decreasing turning angle

### 24.03.2024
Improved search for exit: Improved Line Follower (use side rear in addition to side front distance sensor to measure angle to wall). Using number of black pixels to distinguish between entrance and exit: Small difference in number of pixels, but appears to be enough. Robot detects black pixels even while driving inside evac zone (debris can also cause this) → Use Silver NN (TODO)

### 17.04.2024 to 21.04.2024
German Open in Kassel. Fixed minor bugs:
- confusing deadend for normal intersection
- radius around obstacle too small
- alternate between left/right durning on deadend and obstacle
- "improved" exit (didn't work a single time…)

### 23.04.2024
Fixed broken hardware from German Opens

### 03.05.2024
Printed more replacement parts, hardware finished for Hanover

### 16.05.2024
Brainstormed new NN approaches to decrease computation time

### 18.05.2024
Used purely graphical Edge Empulse Studio over custom Google Colab trainings script to facilitate rapid prototyping and validating new NN architechtures.

### 22.05.2024
Found new NN approach that seems to work faster for detecting victims: Using heatmap based model like previous season

### 24.05.2024
Trained proper model using old victim data, first tests are approx. 30x faster (30ms instead of 900ms) than old model running on the same hardware

### 28.05.2024
Further testing revealed that while speed has been increased significantly, the accuracy is insufficent

### 07.06.2024
Finished docs, uploaded for competition. Printed poster. Final testing