SDU · UNIVERSITY OF SOUTHERN DENMARK

Alexander B. Kjærbo, Karl O. E. Soneff, Lukas H. Sørensen, Magne Jacobsen, Nikolai D. Jensen and Rasmus W. Kildeberg

Line following and object collecting with mobile robots

Deadline:
10.01.2025

Relatore:
Anders S. Sørensen
Jes H. Jepsen
Mads B. Sørensen

Group nr. 8
Robotics 1. semester 2024

ABSTRACT

In this project, a design for two autonomous mobile robots were made with distinct differences. One robot drives around in a circle marked with black tape and the other picks up nuts with an electromagnet, in addition to this it follow a path marked with blacked tape where the nuts where. Both utilise light-depending resistors (LDRs). The speed of the racecar was improved by creating a comparator and flip flop to enhance collection of information for the line of the robot. The logistics car used dead reckoning to navigate between each of the nuts, and a crane consisting of servomotors. At the end of the crane we had an electromagnet designed to pick up the nuts. The requirement for ”Pimp my ride” was fulfilled by creating 3D printed shells for both robots to look like the characters from the Disney movie ”Cars”. The racecar resembles ”Lightning Mcqueen” and the logistics car resembles ”Mater”, with the tow being replaced by the crane. On competition day the racecar managed to drive 45 laps in 5 minutes with the fastest lap being 5 seconds. The logistics car managed to score 11/22 points. Future work for the racecar would be to improve the the flip-flop and comparator and relocate the position of the LDR sensors. Future improvements for the logistics car would be to increase the accuracy and improve the crane.

CONTENTS

I. Introduction

In this project a design will be made for autonomous mobile robots capable of navigating a specified route and independently performing a series of predefined tasks. The first task is to collect the highest number of nuts within a specific time frame. The robot is allowed to take only one nut at a time before returning to PIT. To pick up the nuts, an electromagnet is used. The other task is to make the robot drive around a predefined track with the fastest and/or highest number of laps. To drive around a track, a sensor is used. The track has a white background with black lines used to indicate the direction for the robots. Therefore, light-dependent resistors (LDRs) made into sensors are used to navigate the track. The task given is defined not only by completing the objective, but also by making it as effective as possible. Since two different tasks have been given and two mobile robots have been provided, two different robots will be developed. The purpose of the design of the robots is to be unique, creative, and help support the functionality of the robots.

The Git used for the project can be found in appendix A, and the work distribution of the group can be seen in appendix C.

II. Problem Statements

Track Driving: The fastest and/or highest number of laps completed on a predetermined track within a specific time frame. The control of the mobile robot must be autonomous.

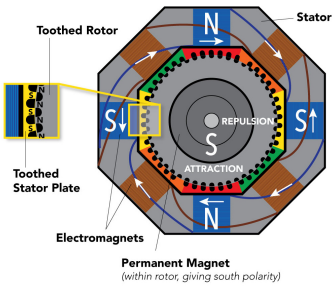
Logistics Management: Collecting the highest number of items within a specific time frame. The control of the mobile robot must be autonomous.

Pimp My Ride: Decorating your mobile robot with creative, unique, and functional designs, including costumes, lights, and sounds.

III. Robotic Fundamentals

A. *Stepper motor*

To control the robots, stepper motors were used. A stepper motor is a motor controlled with a magnet and coils around it with the possibility to run current through which will make an electrical field to apply a force to the magnet. In a stepper motor these two different wires are being used to make coils around the magnet and can be seen in fig. 1a. The magnet consists of many teeth, and when current runs through one of the coils it attracts or repels a tooth close to it, depending on the direction of the current, making a small rotation. When correct timing is used with different coils and making current run both ways through to both push and pull the magnet it is possible to convert this energy into mechanical energy and rotate the magnet. An h-bridge is used to make the current run both ways. Normally, a stepper motor has 200 steps in a full rotation and every step is of equal distance of rotation. This makes stepper motors very reliable and makes it possible to achieve the same result of distance rotated every time. Four wires could then be used to control the h-bridge which controls the stepper motors, which makes the interface for controlling the stepper motors four pins which need to be turned on and off in a specific timing. To control these wires, a type of communication called grey code is used, which when looped in a sequence will turn the stepper motors continuously in a direction. Grey code for a full step is seen on fig. 1b



(a) Cross section of a stepper motor[1]

0	1	1	0	0
1	0	1	1	0
2	0	0	1	1
3	1	0	0	1

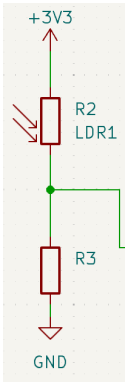
(b) Grey code for control of stepper motors

Fig. 1: Stepper motor controls

B. LDR-Sensor

With both robots having to follow a line it was necessary to make a consistent way to detect the line. When using an LDR the resistance would increase when driving over a black line because the light reflected off of the ground would decrease due to the blackness absorbing the light. The LDR was measured to have a resistance between 6-6.5 kΩ, when it was looking at white, and a resistance between 8-8.5 kΩ when it was looking at black. When this is combined with another resistor in series that has a known value, the ability to calculate the resistance of the LDR is possible by measuring the voltage between two resistors. Shielding the LDR and using a white LED the difference between detecting a black line and the white mat was increased thus making it more consistent. This can be seen in fig. 2a

(a) LDR sensor setup



(b) LDR sensor KICAD

Fig. 2: LDR

C. Differential Driver

To control the Stepper motors a program to act as differential driver was created. The program handles converting commands such as ”400 steps forward,” ”40 steps to the left,” and ”15 cm forward” into grey code for two stepper motors sitting opposite each other.

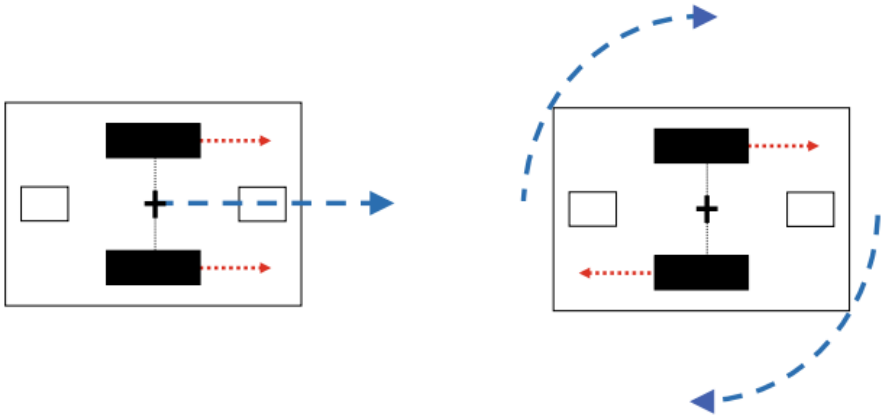


Fig. 3: Example of directional driver movement[2]

Using the differential driver 2 movements can be made. These are illustrated on fig. 3. When both motors are moving forward with the same speed, the robot drives forward. When the motors are turning in opposite directions, the robot turns in the direction of the backward motor. The differential driver is often used asynchronously, so other tasks can be done simultaneously.

IV. Racecar

A differential driver introduced earlier in the semester used the uasyncio libary in MicroPython to asynchronously take steps for stepper motors. This posed an issue when increasing the steps per second (SPS), because a limit was reached by the code's ability to switch to different points in the code at the desired times. This resulted in small delays between steps, while the Pico was letting the other code execute, and that delay took more time than the delay between steps, resulting in fewer SPS.

Since the tasks in the code were simple, it was possible to convert the code to a synchronous flow. Therefore, synchronous code flow was chosen for the racecar.

Utilising full-steps for the stepper motors also assisted in reaching a higher SPS. That is because the Pico cannot sleep for less than 1 ms using time.sleep(). (It has later been discovered that time.sleep_us() can be used to sleep for microseconds). Therefore, if the

stepper motors have more steps for the same amount of rotations (for example, by using half-steps or microsteps), it would require a delay shorter than 1 ms resulting in no delay at all, thus making the wheels have too low torque and unable to do the desired rotation for a step-cycle.

The last improvement to increase the SPS was to increase the pulse-width-modulation (PWM) to as close to 100 %, without ruining the motors, to ensure the motors have the most amount of torque possible. The longer the steppermotors are turned on, per time unit, the more root mean square (RMS) current is being sent through the coils. Increasing the current running through the motors decreases the time the motors need to turn. This is due to the stronger electrical field generated by current through the coils, which applies force to the magnet inside the stepper motor.

After the speed was sufficiently increased, the next problem was that the robot was driving off the track when using one LDR. No matter the number of steps the robot took when detecting the line with the LDR sensor, it either turned too little and drove off the track, or it turned too much, which also led to it leaving the track. Therefore, a second LDR sensor was introduced behind the first. Thereafter, the code was changed so that the robot turns slightly by running the inner wheel at a slower SPS, if the first LDR sensor detects the line. If the robot instead reads the line with the second LDR it turns sharply until the line was read by the first LDR sensor again.

This introduced another problem. When the robot was driving and detecting a line it would turn right and drive forward, but since the approach of the LDRs was to have two LDRs on one side it would not know if it oversteered resulting in driving off the track on the inside. The code was then changed to make the robot always turn a small amount left if there was no input from the LDRs (no black line). This improvement made sure the robot stayed as close to the line as possible, instead of driving straight during a corner after having turned once.

This caused a new problem, because now when the robot was driving left in order to stay close to the line, it would sometimes hit the line too direct. This, paired with driving at

high speeds, makes the time the line is under the LDRs low enough, that the Pico was not able to read the LDR sensors before the line already had been run over.

A. Comparator

Since correction of oversteering sometimes results in the robot driving sharply into the line, the transfer of information between the LDR sensors had to be improved. Instead of making the Pico read the LDR values with the ADC (Analogue-to-Digital-Converter) pins and evaluating if the value should equal an output of 0 or 1, a comparator was made to do this process electronically. The comparator uses op-amps (operational amplifiers) in order to turn the output from the LDR's to either a 1 or a 0. The two op-amps in the diagram are on the robot a single dual op-amp MCP6002 [3]

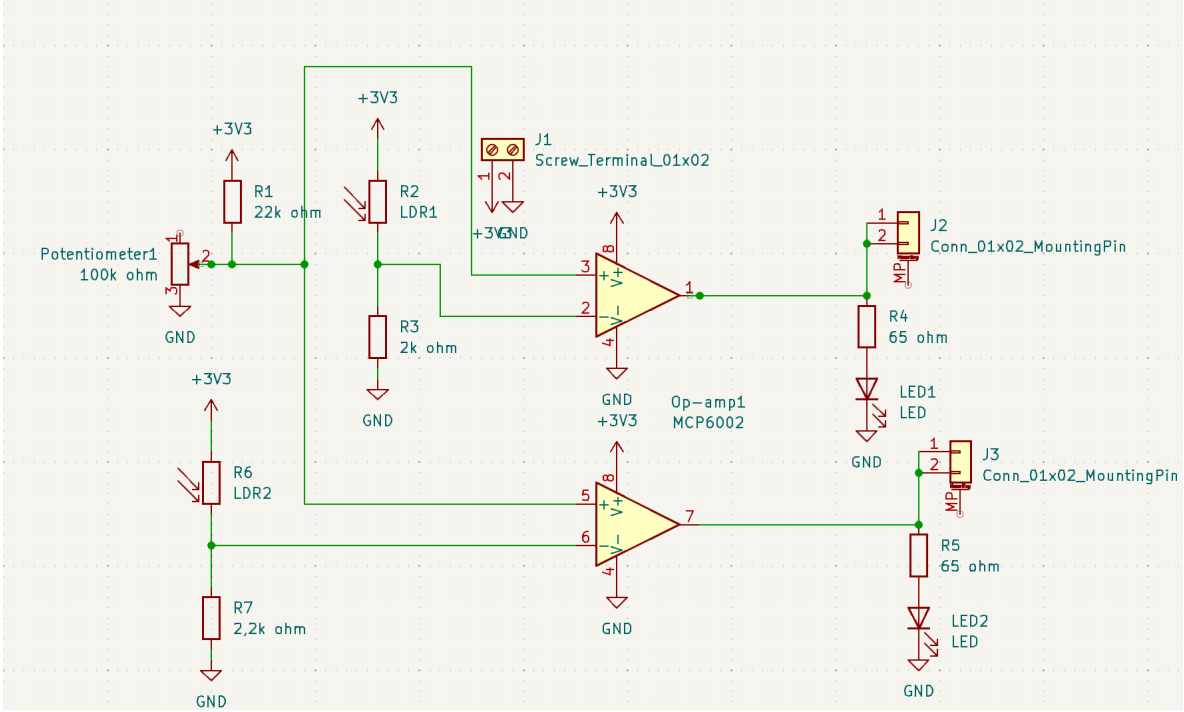


Fig. 4: Comparator

The comparator uses a potentiometer and a resistor to change the threshold for detecting the line. The analogue signal from the LDRs are connected to the negative input of the op-amps. When it detects a black line the resistance of the LDR increases and the voltage

of the negative input pin on the op-amp decreases, resulting in the output of the op-amp going high. The op-amp is rail to rail, which means the voltage of the output is 3.3 V. For testing and wiring to the Pico, two mounting pins are used to insert jumper wires. The LEDs are used to debug the robot, making it easier to see the logic the robot was doing when driving.

B. Flip-flop

Despite the improvements with the comparator making the Pico read a digital signal instead of an analogue signal, the issue with driving off the track still persists.

To fix the issue, two flip-flops were created. A flip-flop is a 1-bit memory, which saves a boolean. Flip-flops in this scenario are convenient because they can save the value of the comparator. That means if the comparator outputs a true and returns to false before the Pico reads the value, the Pico can then instead read the flip-flop which has the value true stored. Afterwards, the Pico will initiate the turning sequence and reset the flip-flop back to false. This ensures the Pico will not miss the black line because the value will be stored until the Pico reads the flip-flop and therefore does not rely on the Pico reading the comparator in the exact time interval where the comparator outputs true.

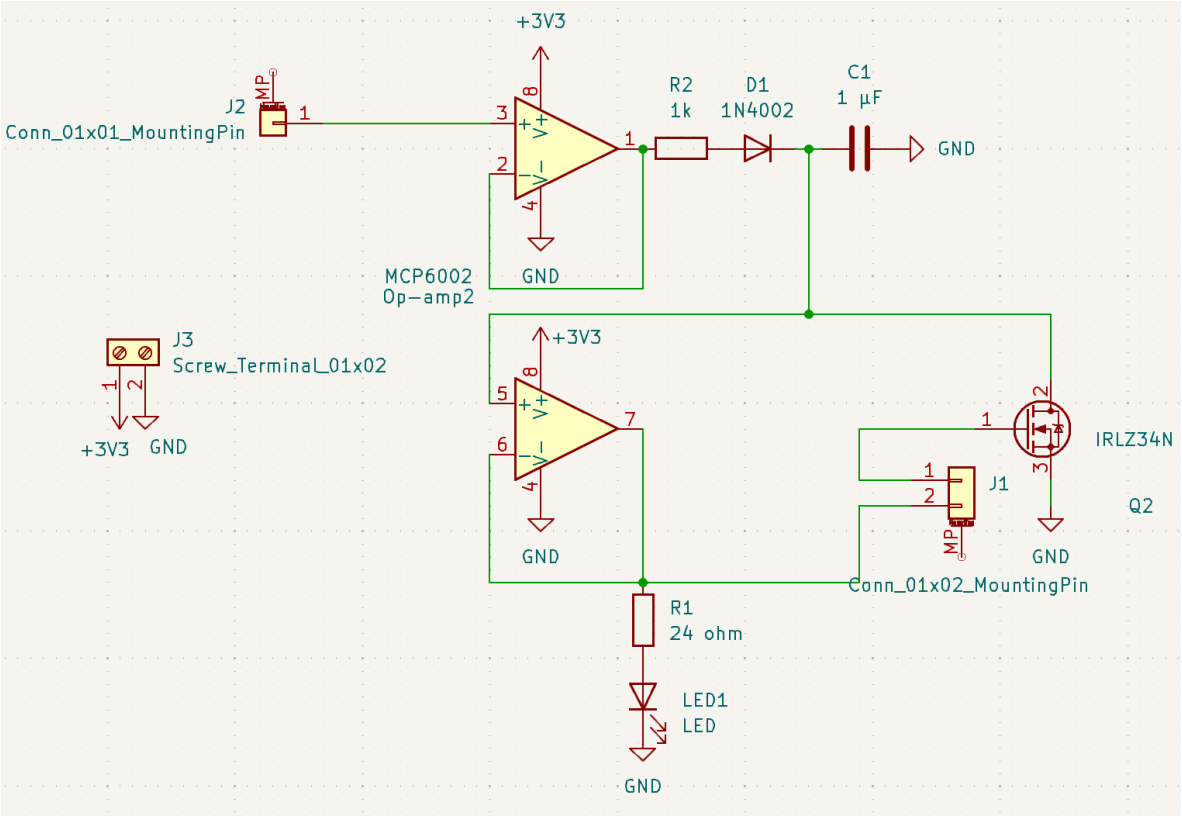


Fig. 5: Flip-flop

Using the two mounting pins, from the comparator PCB, J2 and J3 on fig. 4, a jumper wire can be used to connect to J2 on fig. 5, using an op-amp with negative feedback without any amplification the same voltage as the input was being sent to the output of the op-amp. A resistor was connected to reduce the current to a maximum of $\frac{2.2\text{V}}{1\text{k}\Omega} = 2.2\text{mA}$ due to the diode being a 1N4002 [4] and having a voltage drop of 1.1 V over the resistor is 2.2 V. The capacitance of the capacitor $C = 1\mu\text{F}$ making $\tau = 1\text{ms}$.

The Pico operates on Low Voltage Transistor-Transistor Logic (LVTTL), meaning the output was guaranteed to be registered as true when the voltage over the capacitor was 2 V. Using this formula:

$$V_c = V_s \left(1 - e^{-\frac{t}{\tau}}\right). \tag{1}$$

and isolating the time (t) in eq. (1) we get

$$t = -\tau \cdot \ln\left(1 - \frac{V_c}{V_s}\right). \tag{2}$$

inserting $\tau = 1\text{ms}$ and the voltage used to charge the capacitor $V_s = 2.2\text{V}$ and the desired voltage over the capacitor $V_c = 2\text{V}$ the time the LDR sensors need to detect the line is:

$$t = -1\text{ms} \cdot \ln\left(1 - \frac{2\text{V}}{2.2\text{V}}\right)$$

$$t = 2.3978952728\text{ms}$$

This means that as long as the output of the LDR sensors is high for more than 2.4 milliseconds, the flip-flop would charge the capacitor with enough voltage for the Pico to always detect it as high. The output of the flip-flop was also sent into another op-amp to turn on an LED. This op-amp could be replaced by a transistor, but the MCP6002 is a dual op-amp meaning that another op-amp already was there. This op-amp also had negative feedback without any amplification resulting in the highest voltage being 2.4 V, and for an LED the voltage drop is usually around 2 V or more, depending mainly on colour. This resulted in 0.4 V over a resistor, to reduce the current and to not exceed 20 mA. A resistor higher than: $\frac{0.4\text{V}}{20\text{mA}} = 20\Omega$ was needed, and a 24 Ω resistor was used.

When the capacitor had been charged, the mounting-pins J1 in fig. 5 could be used to check the digital signal with the Pico and the other pin was used to open the transistor to discharge the capacitor, which occurs quickly since there is no resistor and the capacitor was small. The only resistance is the resistance in the wire and the resistance in the transistor. In MicroPython the use of pin.value() can be called with arguments 1 and 0 subsequently with no delay between and that is enough time for the capacitor to discharge faster than the transistor can be closed by the Pico.

The combination of the comparator and the flip-flop makes a reliable way to check whether the robot is driving on a line or not. The LEDs make visual debugging possible while the robot is in motion, which can be used to test numerous different parts of the robot.

C. Driving logic

By reading the output of the flip-flop with a Pico and converting it to actual steps for the motors, an algorithm was developed by trial and error. Since the two LDR sensors was placed side by side one was used to make sure when driving along the line, small corrections would be made. The other sensor was used mainly for the turns, because a sharp turn was needed to make sure it would not drive off the track. When detecting the black line on the second sensor it would turn right until the first sensor detected the black line, and would then go back into the loop. A flowchart for the code can be seen below in fig. 6.

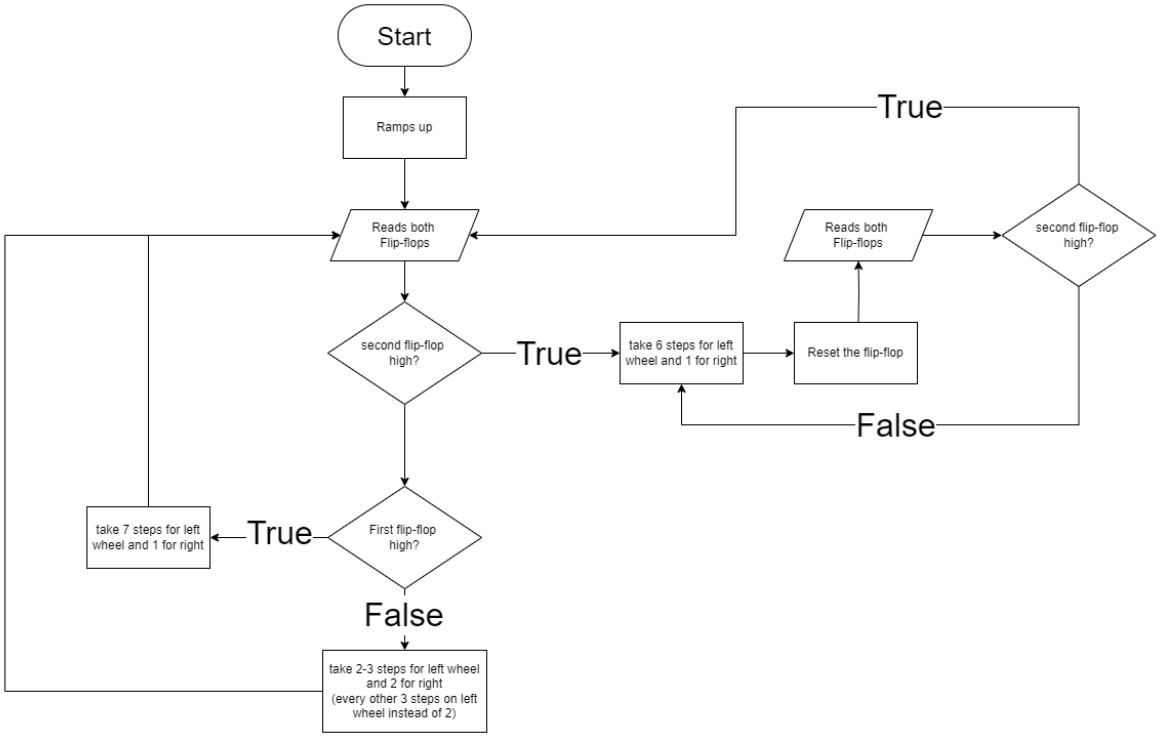


Fig. 6: Flowchart for driving logic

V. Logistics car

A. Navigation

Different modes of navigation can be used to reach the nuts on the field, for this project dead reckoning was chosen as the mode of navigation. One of the major advantages of dead

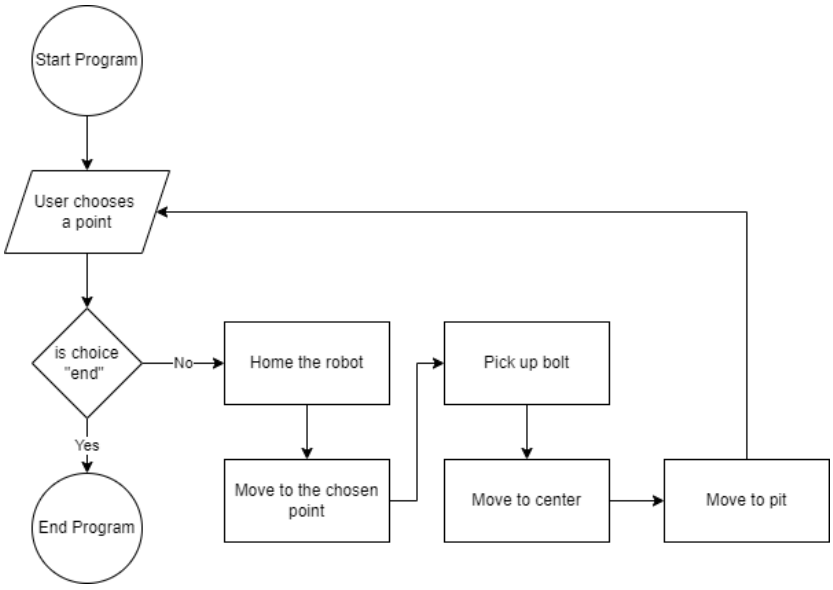


Fig. 7: Superficial flow diagram of the logistics bot navigation

reckoning is the way in which the mode of navigation impacts the difficulty of each nut. While regular line following will have a disadvantage when navigating to the nuts worth 3 and 5 points, since the paths include soft and sharp turns, dead reckonings deviation is based on the distance to the point, hence the closer nuts worth 3 and 5 points are easier than the furthest of the 1-point nuts. The flow of the navigation can be seen on fig. 7.

When navigating using dead reckoning an origin point is used, this can be represented as a position vector $\vec{O}$ with Cartesian coordinates as

$$\vec{O} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{3}$$

When an origin has been established further movement is recorded by noting the direction and distance travelled in the said direction. This can be represented as vectors with polar coordinates $\vec{v} = (r, \theta)$. By doing this, the robot can be controlled with given coordinates.

First, the current location is subtracted from the destination coordinates to determine a vector for the movement. Afterwards arctangent and Pythagorean's theorem is used for converting the Cartesian coordinates of the movement vector to polar coordinates. The

distance r can be calculated by

$$r = \sqrt{(a_1)^2 + (a_2)^2}. \tag{4}$$

while the direction θ can be calculated by

$$\theta = \arctan\left(\frac{a_2}{a_1}\right). \tag{5}$$

As the arc tangent is only defined between $\frac{\pi}{2}$ and $-\frac{\pi}{2}$, manually checking whether a_1 is positive or negative is important. Furthermore, if the angle is exactly $\frac{\pi}{2}$, it is undefined. These considerations are automatically included in the math.atan2() function of the Python math package.

By subtracting the current direction of the robot from the calculated direction, the needed rotation can be calculated.

The rotation angle is then rounded to the resolution of the stepper-motors. This resolution was measured by rotating the robot one full rotation and noting the number of steps. The robot will then rotate this rounded angle. The new direction is also noted.

When the robot has finished rotating, it will move forward. While a movement vector is needed to update the current position, it is important to account for the deviation derived from rounding the rotation. A new movement vector can then be calculated by

$$\vec{v} = \begin{pmatrix} r \cdot \cos\theta \\ r \cdot \sin\theta \end{pmatrix}. \tag{6}$$

With r calculated in eq. (4) and θ being the current direction.

As the robot does not know where the origin point is located on the field, a ”homing” procedure must be performed. It is important that both the position and the direction of the robot on the field is known before proceeding with dead reckoning, as such any deviation during the homing process must be accounted for.

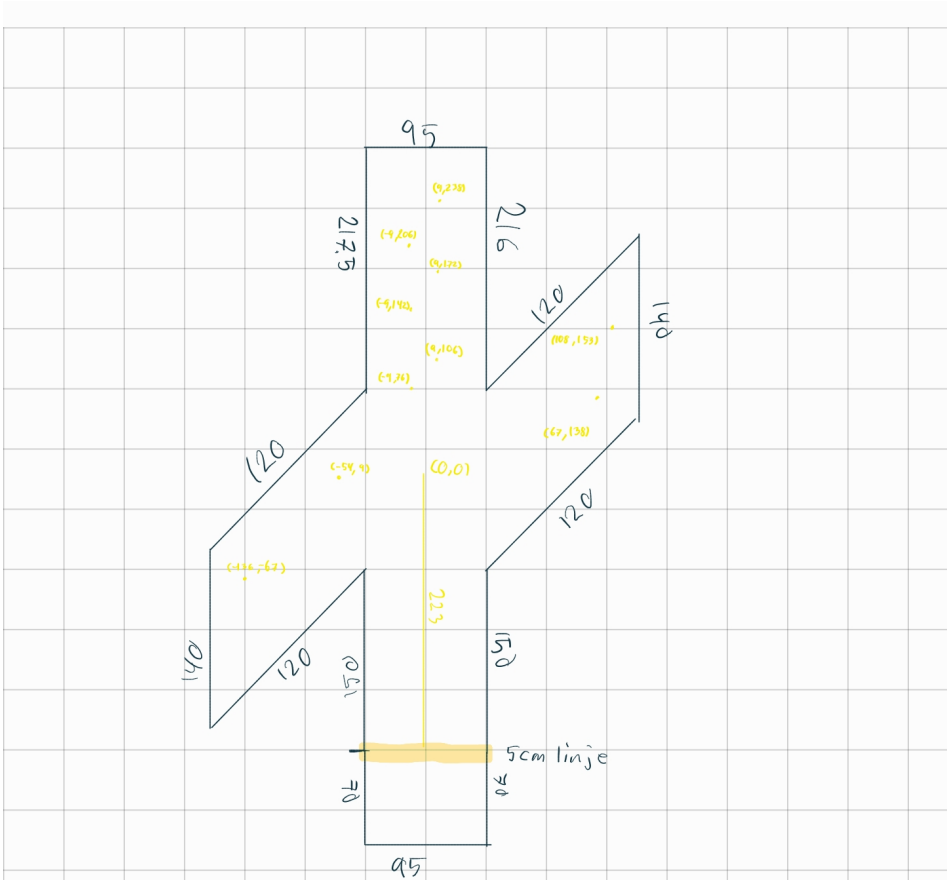


Fig. 8: logistics track mapped with origin and points

Homing is done by driving the robot down the centre line shown in fig. 8. The robot follows a regular line following behaviour during the homing procedure. To follow the line, a pair of LDR sensors spaced 7.5 cm apart are used. These are connected to ADC pins and have a threshold in code that was calibrated on the tournament day. When one of them detect the line, the robot rotates $1°$ in that direction. As the line has a width of 5 cm, it can be reasonably assumed that the positional deviation of the origin after the homing procedure can be calculated by

$$|s_{pos}| \leq |7.5cm - 5cm| = 2.5cm. \tag{7}$$

Furthermore, the directional deviation of the origin can be assumed to be $|\theta_{err}| \leq 1°$. These deviations will have to be taken into account during subsequent dead reckoning. The

Point	(67;138)	(108;153)	(-54;9)	(-136;-67)	(-9;76)	(9;106)	(-9;142)	(9;172)	(-9;206)	(9;238)
value	5	5	3	3	1	1	1	1	1	1
r	153.4	187.3	54.7	151.6	76.5	106.4	142.3	172.2	206.2	238.2
s_{rot}	2.7	3.3	1	2.6	1.3	1.9	2.5	3	3.6	4.2

TABLE I: Distance (r) and deviation (s_{rot}) for every point

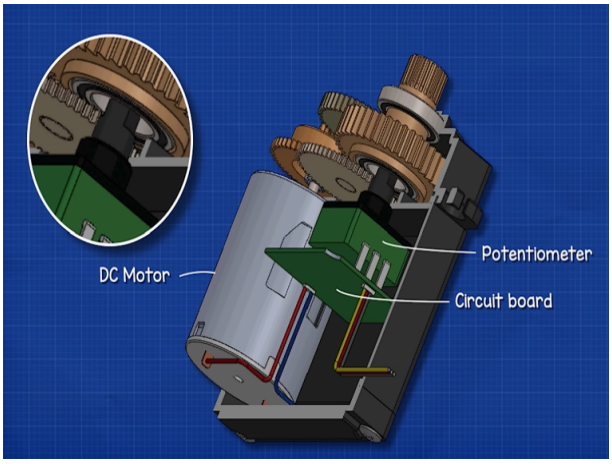
rotational deviation in cm derived at each point from the angle deviation can be calculated by

$$s_{rot} = r \cdot \tan(\theta_{err}) \tag{8}$$

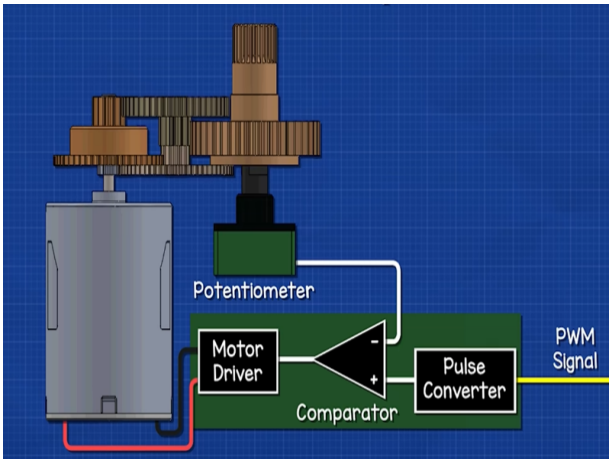
This deviation is represented in table I. The rotational deviation only affects the robot tangentially to its current direction while the positional deviation affects the robot in all directions.

B. Crane

The logistics robot is equipped with a robot arm / crane mounted on its back to address errors encountered during travel, thereby increasing it's margin for error.



(a) Inside a Servomotor



(b) Servo Electrical Circuit

fig. 9a showcases the basic build of a servomotor [5]. Within is a circuit board, a DC-motor, a potentiometer, and gears. A servomotor converts electrical energy into mechanical energy, which can be utilized in a robot arm. A servomotor is most frequently used for precise

control, which makes them a solid choice for building a crane. When a DC motor is connected to power, it rotates with a constant speed and force. A servomotor does not work in the same way. Instead, a signal is sent that can be used to control the exact number of degrees to rotate. There are two types of servomotors: closed-loop and open-loop. Closed-loop can rotate from 0 to 180 degrees, while open-loop rotates from 0 to 360 degrees. The crane utilizes closed-loop servos, which are most frequently used due to better control and precision. A servomotor has three wires: a positive, negative, and a signal wire. PWM signals are sent through the signal wire and used to control the position of the servo. Because of this signal wire, the operator of a servomotor can use a Raspberry Pi Pico to send pulsing voltage signals to the servo. PWM signals are sent every 20th millisecond, which means that it is sending with a frequency of 50 Hz. The width of the PWM signal determines the position of the servo. When a wide signal is sent, the servo rotates counterclockwise. When the signal is narrow, the servo rotates clockwise. fig. 9b showcases the electrical circuit within a servomotor. PWM signals are sent and interpreted as the desired angle. The potentiometer is connected to the final gear, meaning the actual position of the servo is read based on the voltage from the potentiometer. Based on the position of the potentiometer's arm, the input voltage is divided by the amount of resistance, and a modified value can be interpreted. The servo uses a comparator to compare the voltage levels of the actual position and the desired position. When a difference is discovered, an error signal is sent, which creates rotation of the DC-motor. The closer it gets to the desired position, the smaller the error, until finally the actual position and the desired position are in unison. At this point, there will be no difference in voltage, and the servo will stand still.

When the desired angle of the servo is 90 degrees and the actual position is 0 degrees. The actual position will behave linearly along with the error signal until the desired position is reached, at which point movement will halt almost instantaneously. For a servo there are always three cases: 1. The desired position is larger than the actual position. In this case the op-amp signal goes high, signalling the motor to turn right. 2. The desired position is smaller than the actual position, in which case the signal goes low signalling a turn in the opposite direction. 3. The actual position is equal to the desired position in which case the

signal stabilizes, which in turn creates a halt in the movement of the servomotor.

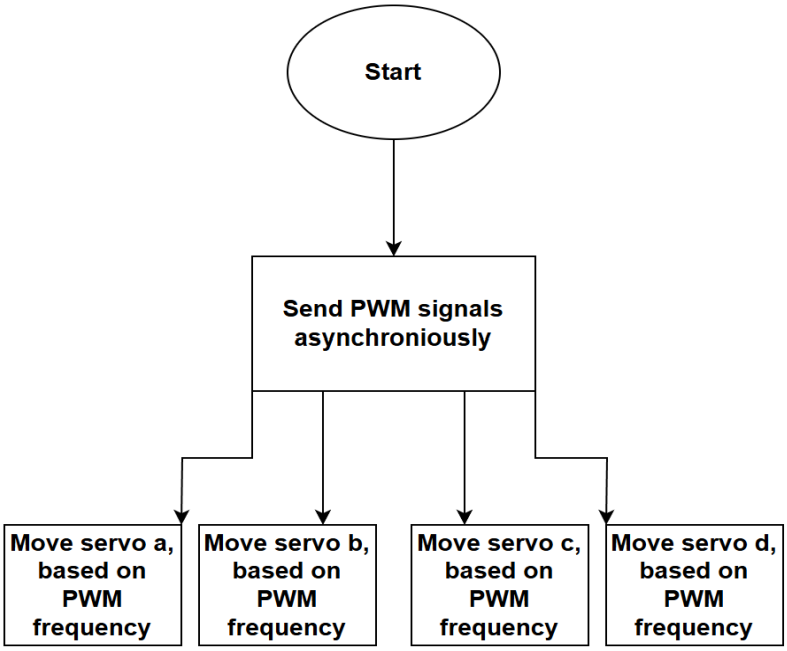


Fig. 10: Flowchart, showcasing how PWM signals are sent to servos in a robot arm.

The crane consists of four servomotors. The two middle servos are larger and have more torque, while the two others are lighter and have less torque. From a programming standpoint, a setAngle function is used and called to move the crane. This function sends a PWM signal with the desired width. By utilizing the uasyncio Python library and multiple servomotors the crane can be moved. The servomotors are moved to the desired angles by reading lists of angle data. The crane follows a certain pattern, which means that a broader search area can be established.fig. 10 is a flowchart showcasing the flow the code for the crane. All the crane's motors are connected directly to the panzaboard, where each of them is connected to a 5 Volt output, a ground pin, and a signal wire. The crane's material is 100% 3D-printed and hand-drawn in Autodesk Inventor. In order to ensure a long reach while maintaining stability and balance, the crane was fashioned to be 28 cm long.

C. Electromagnet

The goal for the electromagnet was to be able to pick up a nut.

The electromagnet seen in fig. 11 was made with a copper wire wrapped around an M6 non-stainless bolt. There are 220 windings of copper wire around the bolt. The copper wire has a diameter of 0.33 mm. The resistivity (ρ) of copper is $1.724 \cdot 10^{-8} \Omega m$ [6]. The resistance through the wire was 1.6 Ω. The length of the wire was also needed to be found. The cross-sectional area of the copper wire is calculated.

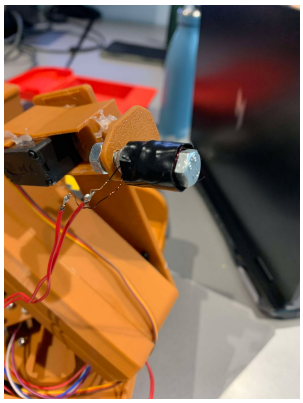


Fig. 11: Electromagnet

$$A = \frac{\pi \cdot d^2}{4} = \frac{\pi \cdot (0.33 * 10^{-3} m)^2}{4} \approx 8.552986 \cdot 10^{-8} m^2. \tag{9}$$

The length of the wire was found from equation eq. (10):

$$R = \rho \cdot \frac{L}{A}. \tag{10}$$

With the length (L) isolated in 10:

$$L = \frac{R \cdot A}{\rho} = \frac{1.6\Omega \cdot 8.552986 \cdot 10^{-8} m^2}{1.724 \cdot 10^{-8} \Omega m} = 7.937806 m. \tag{11}$$

The length of the copper wire was approximately 7.94 m.

The following formula was used to find the inductance of the Coil.

$$L = \frac{N^2}{R}. \tag{12}$$

N is the number of windings and R is the magnetic reluctance. The magnetic reluctance is found by the following formula:

$$R = \frac{L}{\mu} = \frac{L}{(\mu_0 \cdot \mu_r) \cdot A} = \frac{\pi \cdot 6 \cdot 10^{-3} m}{(4\pi \cdot 10^{-7} \cdot 1) \cdot 8.552986 \cdot 10^{-8}} \approx 1.753773 \cdot 10^{11} \frac{A}{Wb}. \tag{13}$$

Now inserted into eq. (12):

$$L = \frac{N^2}{R} = \frac{220^2}{1.753773 \cdot 10^{11}} = 2.759764 \cdot 10^{-7} H = 0.27598 \mu H. \tag{14}$$

The Strength of the magnetic field is found to evaluate the electromagnet.

$$B = \mu \cdot \frac{N \cdot I}{L} = 4\pi \cdot 10^{-7} \cdot \frac{220 \cdot 1.25}{1.5 \cdot 10^{-2}} \approx 0.02303835 Wb. \tag{15}$$

B is the magnetic flux density, μ is the permeability of the core material, I is the current, L is the length of the coil along the core and N is the number of windings.

When dealing with an electromagnet, a few issues arose. One of the issues were that the electromagnet got hot. If the electromagnet was turned on for too long with a strong current, the electromagnet would stop functioning. The software KiCad was used to design the circuit. The KiCad drawing can be seen in figure fig. 12. The battery under the robot was used as the power supply. The current from the battery moves to the electromagnet, which was activated. There was a flyback diode to ensure that the magnetic field from the electromagnet can decrease over the diode if the transistor was off. The transistor was activated by a Raspberry Pi Pico, which supplies the transistor gate with 3.3 V. The transistor used was the IRFU024PBF n-channel MOSFET transistor [7] which was turned on between 2-4 V.

The electromagnet can pick up the nut from a distance about 5 mm from the side and less when it was above because of gravity, when 50% PWM was used. This was used to collect the nut if the electromagnet was not directly on top of it or touched it, because it ensured a higher tolerance of deviation. The electromagnet was a dipole since there were positive and negative poles. The field strength of a dipole field decreases with $\frac{1}{r^3}$ [8]. Therefore, the electromagnet cannot be much further away.

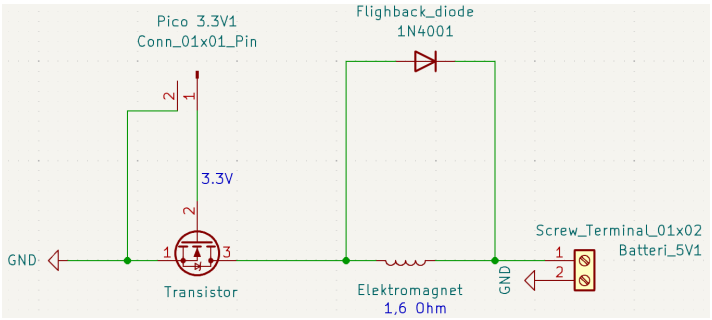


Fig. 12: KiCad Electromagnet

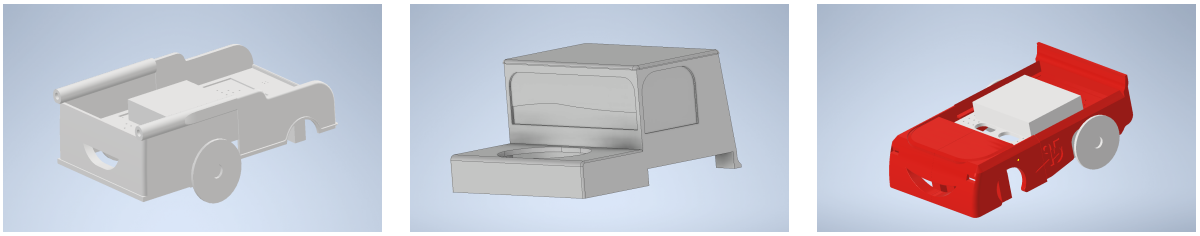
VI. Pimp my ride

For the pimp my ride part it was important to find an impressive design that should also be functional for the rest of the project.

Therefore inspiration from the Disney movie 'Cars' was taken. For the logistics car a 3D-print of the character 'Mater' was made. This design is functional with the crane at the back which is used to collect the nuts. The 3D-print is made in Inventor which can be seen in figure 13a and 13b

For the racecar the character 'Lightning Mcqueen' was taken and a 3D-print of the character was made which can be seen in figure 13c

The final design of the two cars can be seen in fig. 14



(a) 3D Logistic car bottom (b) 3D Logistic car top (c) 3D Racecar

Fig. 13: 3D print of robots

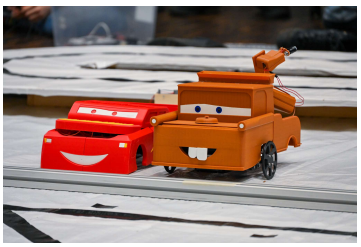


Fig. 14: Robots at the presentation day

VII. Analysis

A. Logistics car

The performance of the robot was observed during the tournament day. The purpose of the robot was to retrieve all the nuts. During the tournament, the robot managed to pick

up 3 nuts worth a collective 11 points out of 22 total. During turning, the robot did not always reach the right angles. Especially at smaller angles, the robot had a tendency to understeer without noting it. This resulted in the 1-point nuts becoming inconsistent to collect. Furthermore, this inconsistency compounded when the robot would return to the origin as well as back to pit resulting in further deviation from the directed path.

To compensate for the inaccuracies in navigation, the crane was built. Although the search pattern helped, the object was not always hit. Notably the robot once managed to pick up a nut. However, it missed the basket when dropping the nut off. This can be seen in appendix B

The electromagnet should be as strong as possible and be able to pick up the nuts. The magnet did manage to pick up the nuts during the test. Although the magnet was able to pick up the nuts, it was not very powerful. This was probably due to the fact that 50 % PWM was used on the magnet.

B. Racecar

The racecar ended up doing laps at around 5 seconds per lap consistently, although this was when it did not drive off the track. It drove 45 laps during the 5 minutes, which was because it fell off the track a few times. If the robot had not fallen of the track it would have theoretically driven $\frac{300s}{5.2s} = 57.7$ laps, because the robot drove a lap in 5.2 seconds on average. The reason the car fell off the track was because the car would slip when it had to turn sharply, since the robot would transition from a high speed to almost $\frac{1}{6}$ of the speed on the inner wheel. The slipping would make the robot drive on the edge of the corner most of the time, however due to minor inconsistencies, the slipping would sometimes be enough for the car to fall of the track.

Another improvement for the racecar would be an additional potentiometer to be able to adjust the thresholds for each of the LDR sensors in the comparator. The issue with having one potentiometer was that the comparator can read false on the line for one of the LDR's, but if the resistance was decreased until it read true, then the other LDR would read true on the red line, which was not optimal.

Although the beneficial design for pimp my ride was carefully made, the design got a second place.

VIII. Discussion

During the competition, a few inefficiencies arose.

A. Logistics car

During the tests, the robot was unable to retrieve all nuts within the time limit. So an improvement would be to increase the speed of the robot. However, when the speed increases, the wheels lose traction, resulting in larger inaccuracies. Wheels with more traction could significantly increase accuracy.

The navigation system for the logistics car was not a very accurate solution. The accuracy of the navigation method could be increased by measuring the robot's direction before driving forward. This could be done with a compass. However, this would introduce other complexities, namely the compass might get affected by the electromagnet on the crane or the magnets inside the stepper motors.

Following the line around the track could be another solution for navigation. Using this method, the inaccuracy from having to turn specific degrees would be negated. Furthermore, the robot should always stop at the correct location, as there was a piece of black tape on the floor under each nut. An LDR could be used to detect the black tape and therefore detect the location of the nuts. Returning the nut to pit can be done by turning $180°$.

There was more success with the crane, although it did have its drawbacks. The desired function was to have a larger error margin, which was achieved. In addition, it provided a larger reach. While it did miss some nuts, this can instead be attributed to the navigation. Furthermore, the crane was not time efficient, given the amount of time used for crane up-and-down movement. Alternatively, a smaller, more robust, and efficient crane could have been used, which could also have consumed less power.

During the test 50% PWM was used for the electromagnet, resulting in a decrease in the power of the magnetic field. Using 100% PWM would increase the power and therefore

the range of the electromagnet. The effect of the increase in power would also increase the heating of the electromagnet. However, this should not cause a problem since the electromagnet was only turned on for a short period of time.

B. Racecar

The whole op-amp in the flip-flop could be replaced by a single transistor since it was later discovered that the signal from the comparator could be sent through the diode and into the capacitor instead of into the second op-amp. The first board that was created did not include the diode, since it was assumed that the op-amp would not allow a current to go into the op-amp from the opposite side, but testing showed that there was a current, which led to sudden discharge of the capacitor. The op-amp the LED is using could be replaced by a transistor with the gate connected to the capacitor, which would achieve the same effect.

The diode was required in the comparator to ensure no current was running back into the op-amp when the negative feedback opened the transistor in the op-amp connecting the output to ground.

The placement of the sensors could be better to ensure that there was enough margin for error to stay on the track despite slipping outward.

IX. Conclusion

The racecar robot drove 45 laps in 5 minutes, with a fastest lap of 5 seconds. In order to achieve this, an LDR sensor was used combined with a comparator and two flip-flops to follow a line around a racetrack. A simple algorithm was created through trial and error to follow the line, where the robot sometimes drove off the track due to minor inconsistencies.

The logistics car collected 11 points out of 22. The robot would autonomously drive to the items utilising dead reckoning and a crane as opposed to regular line-following behaviour. However, this resulted in slower execution and unreliable collection of the nuts.

The robots got second place in the pimp my ride competition. The design was inspired from the Disney movie 'Cars'. The design of the crane aligned with the character 'Mater'

and played a tremendous role for collecting the nuts. The racecar aligned with the character 'Lightning McQueen'.

X. Contextualisation

Robots have increasingly become a part of the everyday life of people in society. They help people in many ways. Robots like the ones in this project can also be useful in society.

The ability to go to a place and collect objects can be useful in many ways. In a lot of industries humans go to a place to collect objects. Instead, robots such as the logistics robot can be used to make the whole process faster and more effective. If the logistics car was made bigger and stronger it is possible to see robots pick heavy objects up and spare people the risk of injury.

Autonomous cars will continue to become an even bigger part of society. The navigation used for the logistics robot can be used in society to navigate complex networks of roads, just like the track in the tournament. This in combination with the racecar which optimises for speed and efficiency can be used to make travelling faster, more efficient and safer.

APPENDIX A

STUDENT GIT

https://github.com/BigMagnus6Tips/Sememsterprojekt-grundl-ggende-styring-af-robotter

APPENDIX B

VIDEO OF ROBOT MISSING NUT

Link: https://www.youtube.com/shorts/o8UHIXkwafU

APPENDIX C

TABLE OF WORK DISTRIBUTION

Racecar	Lukas	Alexander	
Pimp my ride	Karl		
Electromagnet	Nikolai	Rasmus	
Crane	Karl	Rasmus	
Logistics car logic	Magne	Karl	Nikolai

A Bibliography

References

[1] Clippard. *How Stepper Motors Provide Precision Control*. URL: https://www.clippard.com/cms/wiki/how-stepper-motors-provide-precision-control.

[2] Mobile Robots to Autonomous Vehicles with Raspberry Pi and Arduino. *Embedded Robotics*. URL: https://link-springer-com.proxy1-bib.sdu.dk/book/10.1007/978-981-16-0804-9.

[3] *1 MHz, Low-Power Op Amp*. MCP6002. Microchip Technology Inc. URL: https://ww1.microchip.com/downloads/aemDocuments/documents/MSLD/ProductDocuments/DataSheets/MCP6001-1R-1U-2-4-1-MHz-Low-Power-Op-Amp-DS20001733L.pdf.

[4] *Vishay General Semiconductor*. 1N4002. Vishay Siliconix. Apr. 2020. URL: https://www.vishay.com/docs/88503/1n4001.pdf.

[5] The Engineering Mindset. *Servo Motors, how do they work*. Jan. 2022. URL: https://www.youtube.com/watch?v=1WnGv-DPexc&t=571s.

[6] The Engineering ToolBox. *Resistance vs. Resistivity*. URL: https://www.engineeringtoolbox.com/resistance-resisitivity-d_1382.html.

[7] *Power MOSFET*. IRFU024. Vishay Siliconix. May 2021. URL: https://4donline.ihs.com/images/VipMasterIC/IC/VISH/VISH-S-A0013572219/VISH-S-A0013857045-1.pdf?hkey=6D3A4C79FDBF58556ACFDE234799DDF0.

[8] *EM 3 Section 5: Electric Dipoles*. The university of Edinburgh. URL: https://www2.ph.ed.ac.uk/~mevans/em/lec5.pdf.

[9] *IRLZ34N HEXFET® Power MOSFET*. IRLZ34N. International Rectifier. Aug. 1997. URL: https://logosfoundation.org/instrum_gwr/tubo/irlz34n.pdf.