Technical University of Denmark

*Lars David Aktor - s134062*

*Jesper Martin Christensen - s130580*

# Sensor Based Navigation for Autonomous Car

Master Thesis, July 2018

DTU Electrical Engineering
Department of Electrical Engineering

**Sensor Based Navigation for Autonomous Car**
**Sensorbaseret navigation til autonom bil**

**Report written by:**
Jesper Martin Christensen
Lars David Aktor

**Advisors:**
Nils Axel Andersen
Ole Ravn
Henning Si Høj
Jesper Schramm

**DTU Electrical Engineering**
Automation and Control
Elektrovej
Building 326
Technical University of Denmark
2800 Kgs. Lyngby
Denmark
Tel : 4525 3576

studieadministration@elektro.dtu.dk

| | |
|---|---|
| Project period: | 12 February 2018 - 31 July 2018 |
| ECTS: | 30 |
| Education: | MSc |
| Field: | Electrical Engineering |
| Class: | Public |
| Remarks: | This report is submitted as partial fulfilment of the requirements for graduation in the above education at the Technical University of Denmark. |
| Copyrights: |  |

**Abstract**

In this project the goal is to develop an autonomous system for a car that based on sensor input, can calculate the optimal route and use this route as reference signal, to control steering angle, speed and brake power by simple controllers.
A series of design requirements posed to steering precision, speed, brake control and algorithm execution time have been stated and a navigational system is proposed, implemented and verified to solve these requirements.
By participating in the Shell Eco Marathon Autonomous Urban Concept competition in London 2018, the system is demonstrated to work and successfully fulfill the requirements set in this thesis.

# Contents

# Reading Guide

Here the content of the different chapters is outlined.

**1. Glossary**
An introduction to the terms used in this thesis.
**2. Introduction**
An introduction of the thesis.
**3. Problem Formulation**
Here the thesis problem is stated. **4. Software**
A brief description of the software architecture used.
**5. Hardware**
A brief description of the hardware available on the car.
**6. Navigational Algorithm**
A description of how to find the middle line from LiDAR measurements.
**7. Post-Processing**
A description of how to convert middle line into controller reference signals.
**8. Steering Control**
Here, the choice of controller used for steering and how to tune it, is explained.
**9. Program Structure**
An overview of the software architecture in the autonomous system.
**10. Testing**
An overview of how the autonomous system has been tested.
**11. Results**
A presentation of the results achieved by this system.
**12. Discussion**
Here the approach and results are critically evaluated.
**13. Future Work**
Propositions for future work are presented.
**14. Conclusion**
A conclusion is made to answer the thesis problem.
**Source code**
Source code and a kinematic model can be found on the CD included with the report, or online.

# 1 Glossary

The following is a list of the abbreviations used in this report.

**SEM** is the Shell Eco Marathon, an annual competition for student teams to bring their student made special cars to an event site and compete in different categories.

**AUC** is the Autonomous Competition, a part of SEM which runs parallel to the mileage competition.

**Wiki** The SEM AUC rules are described in the official rules, chapter 4.

**Ecocar** is DTU Roadrunners' mileage car. This year it is fitted with an autonomous system as well.

**LiDAR** is a portable laser scanner. The lidar used in this project is a 3-dimensional Velodyne puck VLP-16.

**ROS** is the Robot Operating System. Although not an actual operating system it is an open framework used in robotics applications. Programs called nodes are used to publish and receive messages.

**Publishing** is communication between nodes in ROS. The published entity is called a message.

**Gazebo** is the used simulation environment.

**Autonomous** is when a robot or machine is programmed to react to an input "by itself' as opposed to being human controlled.

**Autonomous system** is the set of programs made in this project to drive the car autonomously.

**Burn** is running the car engine.

**Speed control** is the part of the autonomous system which controls the car's speed according to the navigation. Burning can be done to accelerate the car, braking can be done to decelerate the car.

**Minspeed** and **Maxspeed** are the names for upper and lower boundaries in a speed

interval, with minspeed <= maxspeed. The speed control burns the engine if the speed of the car falls below minspeed, and stops burning when the car has accelerated above maxspeed.

**Slow** and **fast** are two separate sets of maxspeed and minspeed.

**On-line** in this report refers to something done in real time, as opposed to something pre-planned beforehand.

**Controllers** are devices which based on a system signal, can control an input to the system to achieve the desired performance. **P**, **PI** and **PID** are abbreviations which explains if the controller has proportionality, integration and/or derivative control.

**BOOST** is a C++ code-library with additional functions to use in C++.

**Brake control** in this project means a program which sends commands to the actual brake control node, which is made by other students in parallel with this project.

**Brake power** or **brake pressure** refers to the hydraulic pressure level in the brake circuit.

**Computation time** notations, that shows the time correlation between measurements and program execution time, (unitless). $O(n^2)$ means an algorithms worst-case computation time is dependent on $n^2$. i.e. if 1 observations of n result in the execution time being $1 \cdot x$ seconds, then 10 observations takes $10^2 \cdot x$ seconds. Similarly, $O(n \cdot log_2(n)$ means 10 observations would take $10 \cdot log_2(10) \cdot x$ seconds.
In this project, n is the number of LiDAR measurements.

# 2 Introduction

## Shell Eco Marathon

The Shell Eco Marathon is an annual international mileage competition for students with self-made cars. Teams compete in different categories based on one of two types of car, the *Prototype* or the *UrbanConcept*, as well as the type of fuel used. This year a new competition is introduced: the autonomous competition, which consists of a set of five challenges a car must complete autonomously without being controlled by a driver. For safety reasons a driver must still be present in the car and able to disable the autonomous system in case of accidents, however if the driver intervenes in the steering of the car during autonomous driving the attempt is invalidated.

The goal of the competition is to mimic the challenges in making real-life autonomous cars and to encourage students to develop solutions to these challenges. In the competition points are given according to performance, with 100 points to each completed challenge and an additional 50 bonus points available depending on the specific challenge.

The competition is divided into two parts: one in Paris and one in London. In order to qualify for the competition in London the car had to complete the qualification challenges in Paris. The official rulebook for the competition is the Shell Eco Marathon Autonomous Competition rulebook, also called the SEM AUC rules chapter 4.

### The five challenges

The five challenges are defined by Shell Eco Marathon to the following:

1. Autonomous lap
2. Complex track
3. Parking challenge
4. Obstacle avoidance (also known as the gate challenge)
5. Multiple parking spots

All challenges are confined to a unique, closed track walled with barriers. They differ in the specific layout of the track as well as goal of the challenge. Common for all challenges is that the car must not touch any of the barriers, or the attempt will be invalidated. From the SEM AUC rules it is known that the track will always be fenced

by barriers no less than 50cm high, and that the track width will never be smaller than 5m. The on-track barriers' dimensions are described in article 419, and the parking end-barrier as well as the parking spot dimensions are described in article 418. Track length is known beforehand for all challenges. This is all the information needed to make a car that drives autonomously in the SEM AUC. Moreover there is only allowed one car on the track at a time, so there are no moving obstacles for this competition.
Each team has two attempts at each challenge.

In the London competition only challenges 1, 3 and 4 were used and challenges 2 and 5 were discarded.

## Autonomous lap

The autonomous lap is a simple race track which the car must drive. The car must then drive to the finish line, with bonus points given to the car closest to the ideal time. In Paris the track length was 180m, and in London the track length was 970m with an ideal speed of 18kmh which equals an ideal time of 194 seconds. The maximum average speed allowed is 25 kmh, and the minimum speed is 9 kmh. If the car stops for longer than 3 seconds the attempt will end. The track contains elevation, and the width of the track varies but is no smaller than 5m. The official track layout description can be seen in figure 2.1.

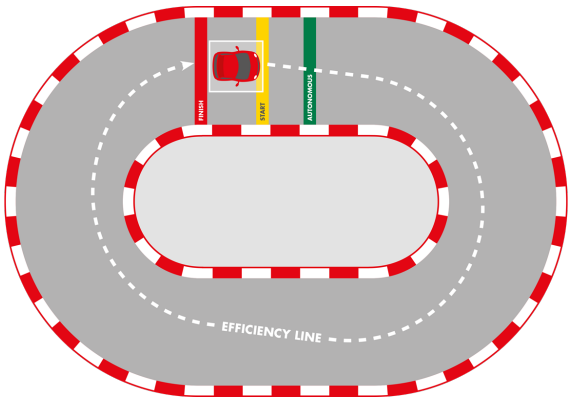



Figure 2.1: Official track description of challenge 1 (from the SEM AUC rules, article 416).

It should be noted that while the final track layout is available (see figure 2.2) the exact turn radii on the track are not. The official elevation map specifies the maximum elevation difference to be less than 2m.

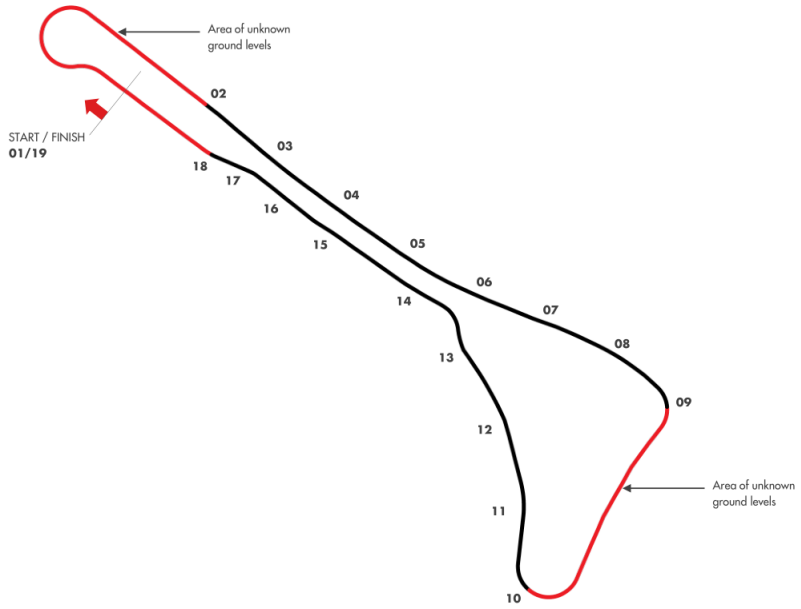



Figure 2.2: Official challenge 1 track layout, from the SEM AUC rules

## Complex track

The complex track is a stretch of track with large, square obstacles on the road. The car must then avoid touching the obstacles while driving to the finish line. Bonus points will be given to the fastest car. This challenge was completed in Paris, but was not selected to be in the final competition in London and did thus not grant points. The official layout of this challenge can be seen in figure 2.3.

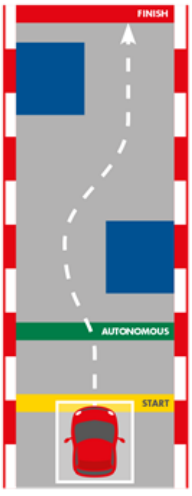



Figure 2.3: Official track description of challenge 2 (from the SEM AUC rules, article 417).

## Parking

In the parking challenge the car must drive down a straight stretch of track, detect a fixed parking spot and park within the spot and before an end barrier. The track will be

between 30 and 50m long. The parking spot is a rectangle marked with black and yellow striped tape with the inner dimensions 4m long and 3m wide. At the end of the spot is the end barrier, which is 3m wide and 50cm tall. When parked all four wheels of the car must be within the rectangle. Bonus points are given to the car that is closest to the end barrier. The official layout can be seen in figure 2.4.

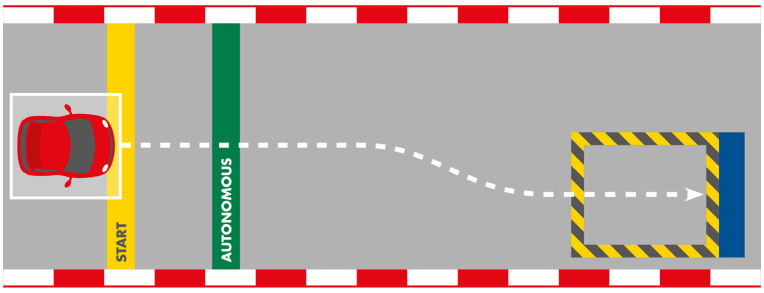



Figure 2.4: Official track description of challenge 3 (from the SEM AUC rules, article 418).

## Obstacle avoidance (the gate challenge)

In the obstacle avoidance the car must drive slalom through three fixed gates on a straight track with length between 30 and 50m. The gates are made of pairs of cylindrical barriers, each measuring 60cm in diameter and 120cm tall, with varying gate width between the cylinders. The length between the gates is 10m. The width of the first gate is 3.5m by the closest point, the width of the second gate 3.0m and the width of the last gate 2.5m.

All three gates must be cleared before crossing the finish line. Bonus points are given if the car touches certain fixed bonus markers (30x30cm) on the ground. The official layout can be seen in figure 2.5.

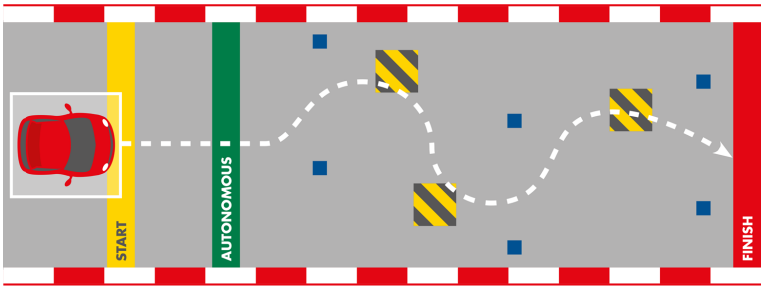



Figure 2.5: Official track description of challenge 4 (from the SEM AUC rules, article 419).

## Multiple parking spots

In this challenge the car must drive a straight stretch of track with four identical parking spot rectangles, without end barriers, only differentiated by a number written within the rectangle. The car must park within the beforehand designated spot.

This challenge was not selected to be in the final competition in London. The official layout can be seen in figure 2.6.

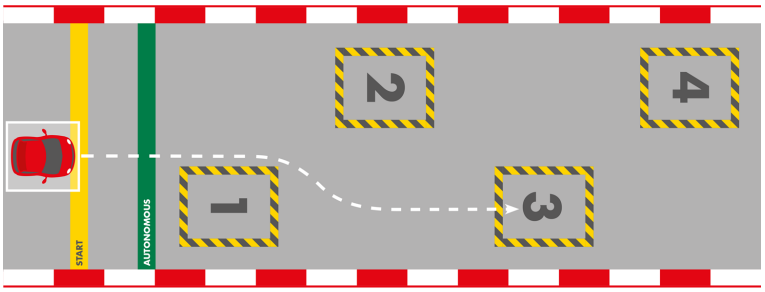



Figure 2.6: Official track description of challenge 5 (from the SEM AUC rules, article 420).

# Practical Considerations

To be eligible to participate in the autonomous category, certain official requirements must be met which include:

- There must be no external communication with the autonomous system, such as wireless communication.
- Sensors and actuators must not obstruct the driver's view nor the driver's ability to enter or exit the car
- Actuators and sensors must be isolated and shielded, so as not to risk the driver's health

Generally, teams must take care to make the car safe and the driving system must be completely autonomous with only pre-planning allowed. Furthermore, before the car is allowed to drive it has to pass a technical inspection which includes demonstrating the brakes, the engine, that the driver can exit the car in less than 10 seconds and more. The autonomous part of the inspection is summarised in the *vehicle scrutineering* sheet. The full list of requirements that the car must meet can be found in appendix B.

## Autonomous driving

In order to drive autonomously, a vehicle needs sensors to measure its surroundings map obstacles, road and other relevant inputs such as parking spots, road signs etc. The sensor measurements can be converted into a momentary map or even permanent map if the information is needed more than once. A navigation algorithm can then be made, which couples the map with the knowledge about barriers, track width, distance and desired behaviour to plan a route for the vehicle to drive. The vehicle then needs actuators for steering and a motor for propulsion, as well as a brake actuator to stop or slow down. The algorithms must be sufficiently fast in order to map, plan and control a vehicle in real-time, since reactions must be taken faster according to the speed of the vehicle.

# 3 Problem formulation

A system must be made that reliably overcomes the challenges laid out by the SEM AUC rules and win the SEM autonomous category, as described in the **Introduction** chapter. It is believed that this problem can be solved in a satisfying manner by combining control theory, robotics and software design.
Therefore, the aim of this project is to create a combination of navigation algorithm and control that can achieve the following:

- **Speed goal**: It is desired that the car completes the 970m autonomous lap as close as possible to the ideal time of 194s, which corresponds to average time of 18km/h (5m/s).
- **Parking goal**: It is desired to park the car as close as possible to the end barrier without physical contact, with all four wheels inside the parking spot rectangle
- **Gates goal**: It is desired drive slalom through every gate in the gates challenge without physical contact

## General design

Common for all these goals is that they require steering, braking and engine control. Aside from the parking goal, necessary information for navigation are found by using a LiDAR. The parking goal requires additional camera recognition to see the parking spot, and the gate goal is implemented by recognizing the shapes of the gates with a LiDAR.

## Speed control

The desired goal speed is defined as 18km/h as this is the specified ideal speed for challenge 1. This requires speed control which keeps the car moving as close as possible to 18km/h (average). Due to the car also participating in the SEM Urban Concept, the car engine is limited to run by using a "burn and coast" method, within a given speed interval. Because the engine is not suited for precise control such as can be done with a PID-controller, the speed control will be done by hysteresis. Since the track in challenge 1 is $970m$, the ideal time to complete the lap is $194s$. For the remaining challenges speed is not a goal in itself, but instead a slower speed is desired, in order to have very precise control of the car.

## Steering control

In the gates challenge the car is expected to go through gates that are 3m offset to left and right, while only having a distance of 10m in the driving direction. The steering controller must be tuned specifically to the accommodate these kinds of reference inputs, at low speeds. As such the steering controller must be able to handle a 3m offset-error orthogonal to the driving direction within 10m of driving to complete this challenge. In the parking challenge, as long as the car can do as stated above, the navigation should be achievable as the camera software implemented on the Ecocar can detect parking spot from further than 10m distances.

## Brake control

The brake must be able to reliably make a full stop when parking, before hitting the end obstacle at the parking spot, as the car can not reverse. Furthermore, the car should be able decelerate by using the brake. This is necessary to effectively control the speed of the car.

## Computation speed

Computation speed is crucial when moving a vehicle at $25km/h$, calculating the correct result in a low time frame is often as important as calculating the correct result[10]. If a vehicle drives with $25km/h$ or $7m/s$ and the road-width is $6m$, using more than even $100ms$, corresponding to a delay of $0.7m$, from measurement to reaction can result in disaster. Especially when factoring in sensor update rates, communication delay, measurement processing time, actuator step-times and -limits.
Computation time $O(n^2)$ or less is therefore a requirement, where $n$ is the amount of LiDAR measurement points received.

## Simplicity and integration

This is the first year for autonomous driving on the *DTU-Roadrunners* team and since students from the university are able to build upon the work in this project, it is a priority to build a solid foundation. A simple system is required that is easy to understand for new students and integrate with their own work.

## Middle line

Because of limitations to hardware and steering wheel speeds, there is a desire to position the car as ready as possible for new unknown input. The middle of the road is defined as the optimal position. Staying in the middle of the road maximizes the critical time available for an unknown input as well as minimizes the risk of crashing into the sides of the road.

## Specific requirements

After summarizing the system design. A list of specific requirements can be made.

The requirements for a functioning solution to this problem are:

- Make the car able to perform a 3m step to the side within 10m in the driving direction
- Perform turns with a turn radius of 8m
- Perform speed control, that can achieve an average speed of 18km/h
- Drive autonomously for 1000m
- Be able to drive in the middle of the road at all times by using LiDAR data
- Park by using camera data
- Steering control that does not oscillate, and provides a smooth experience for the driver.
- To brake and stop the car within 2m of the end barrier in the parking spot
- To have a program run time of less than 100ms

## Proposed solution

A solution is proposed, which solves the problems by making an algorithm with the following key points:

- Receive LiDAR data points and camera detected parking spots
- Determine left and right side of the road
- Calculate middle path with voronoi diagram
- Transform middle path into reference signals for speed and steering
- Implement controller for steering angle
- Implement control for speed
- Options for driving according to middle line or according to parking spot detection or gate detection.

## Timeplan

The preliminary time plan for the project can be seen in figure 3.1.

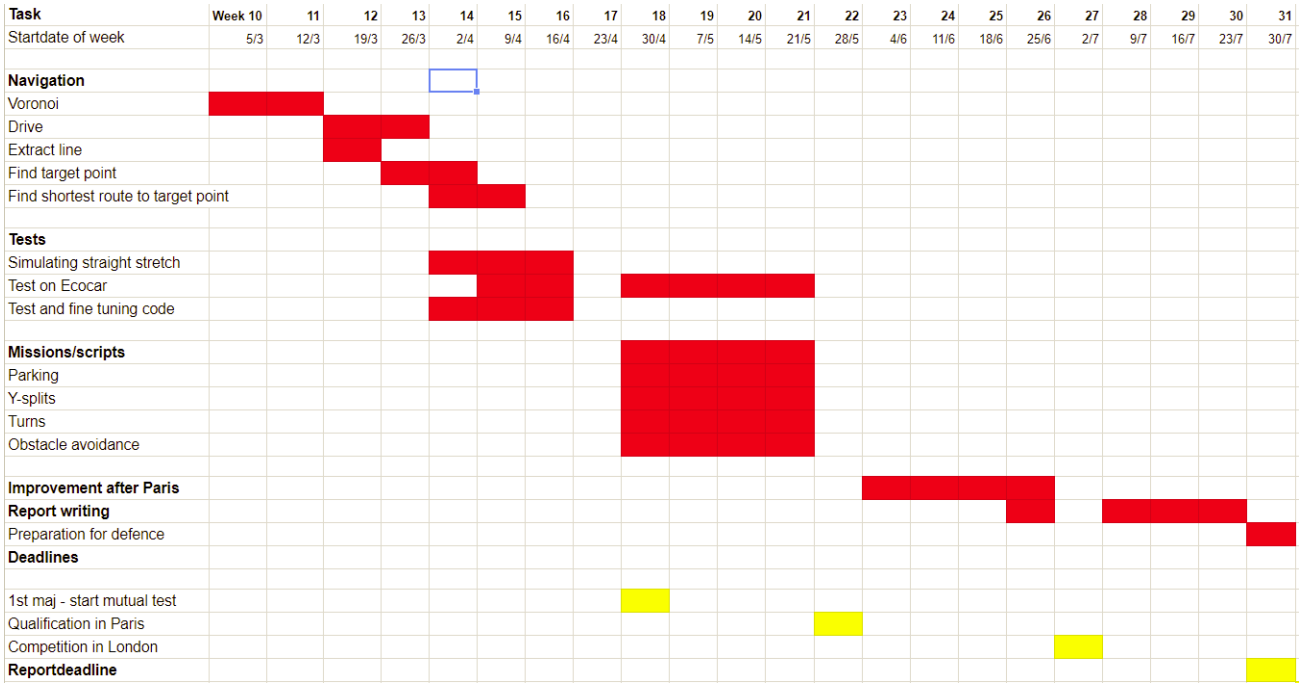



Figure 3.1: Proposed time table for the project

# 4 Software

In this chapter the software used is described.

## ROS C++

The biggest software tool used in this project is ROS C++ running on Linux. ROS is an open framework consisting of libraries and programs used to making applications for robotics [2]. It is suited for student projects and several people working in parallel as it is easy to make programs modular. Much software architecture was already implemented in ROS and as such it is decided to continue using ROS.

## How ROS works

In the ROS framework a program is called a *node*. Nodes communicate with each other by publishing and subscribing to messages on named topics with the message type and fields are defined by the user. This makes it easy to use different nodes with different purposes [3]. Each node is given a fixed update time.

## Simulating in Gazebo

A simulation environment is very useful in designing robot programs. It enables the user to easily try and verify different programs with risking harm to the robot, or in this case, car. Gazebo is a ROS-compatible simulation environment. In it it is possible to use the same aerodynamics and rolling resistance as the real car, making it a useful tool to verify the different programs, particularly in the early stages of this project. It is also possible to create tracks similar to those in the final challenge [4]. In figure 4.1 the simulation model of the car can be seen.

## Rviz

Rviz is ROS' visualization tool used to display data points in 3D. In this project it is used to display lidar data points as well as sorted points and the drive route from the

voronoi node [5].

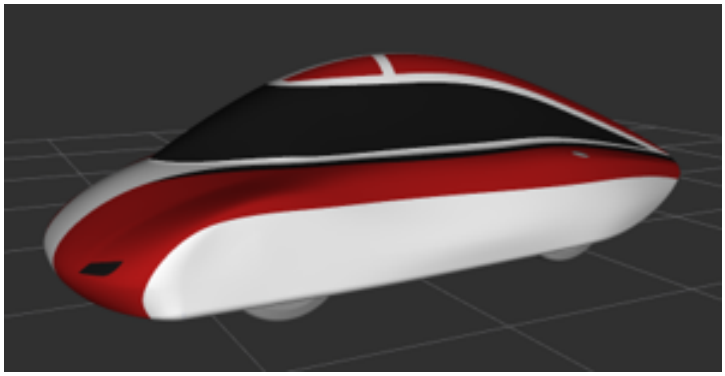


Figure 4.1: The car in Gazebo

# 5 Hardware

In this chapter, the available hardware on the car is described.
The hardware on the car consists of actuators, sensors and switches.
The dimensions of the car can be found in Appendix A.1.

The steering and brake actuators can be seen in figure 5.1. Figure 5.2 shows the actuators from the driver's seat.

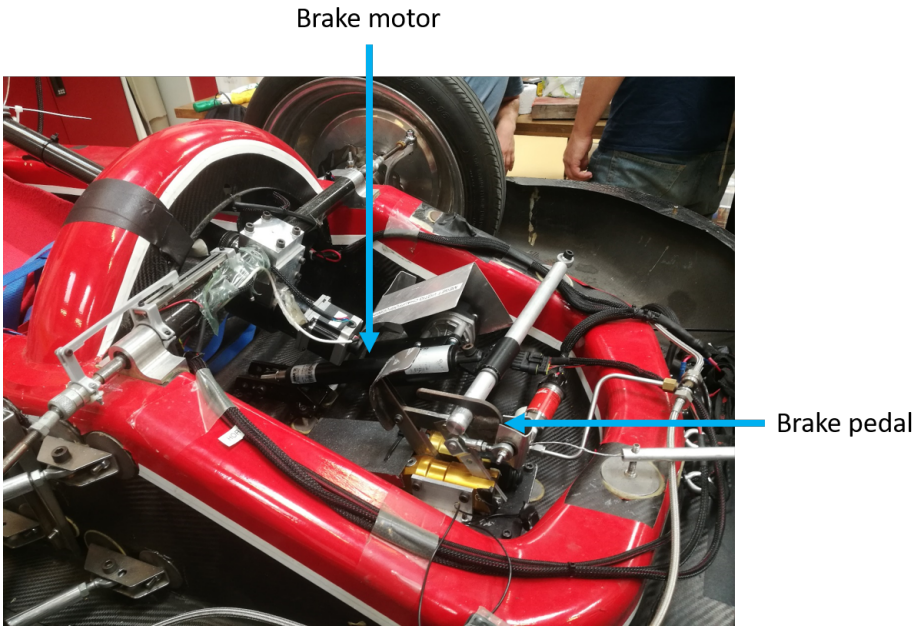



Figure 5.1: Overview of the brake system of the car


Figure 5.2: Steering wheel, auto-pilot button and emergency button

## Steering actuator

The steering actuator is a stepper motor connected to a slip gear [1]. The slip gear makes it possible for the driver to overtake control of the wheel against the stepper motor.

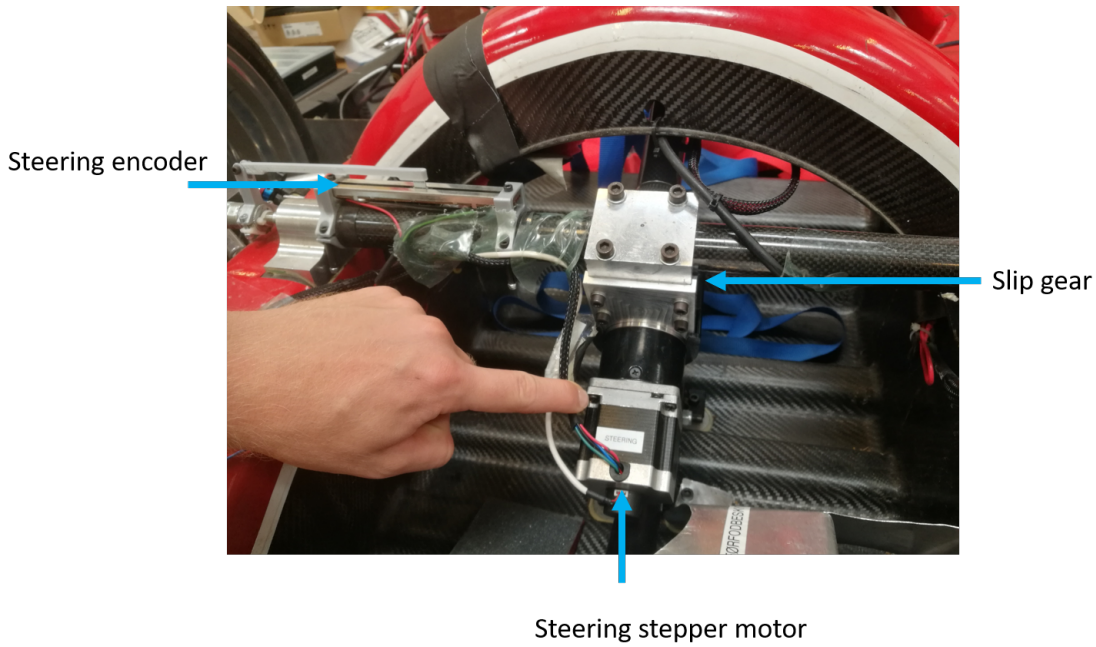



Figure 5.3: Overview of the steering stepper motor, slip gear and steering potentiometer

In figure 5.4 a step response of the stepper motor can be seen. The stepper takes 0.9s to from 0 to one extremum. Furthermore the wheels are limited to turning

within ± 15 degrees.

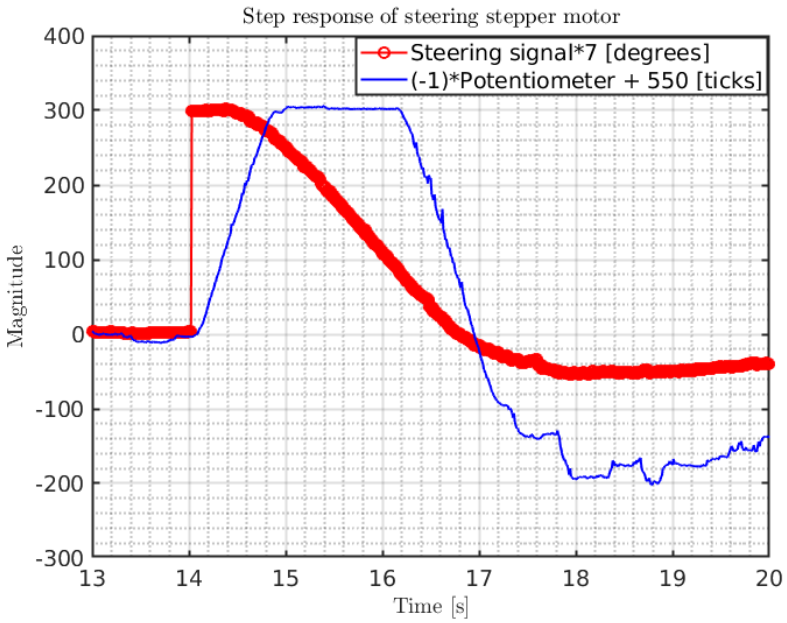
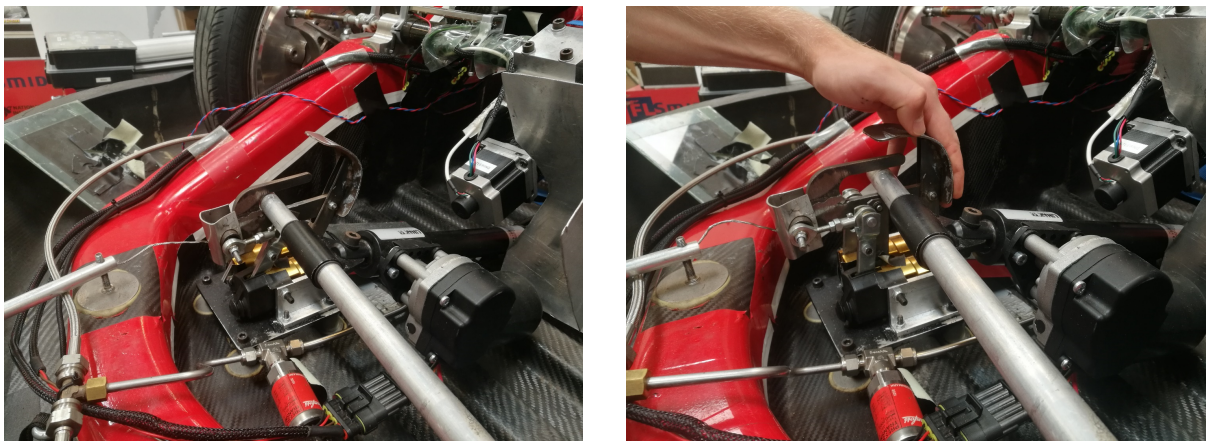



Figure 5.4: Step response of the steering stepper motor, taken from final challenge 1

## Brake actuator

The brake system consists of two hydraulic circuits, one for the front wheels and one for the rear wheels. The brake actuator used is a hydraulic prismatic motor which extends or retracts its arm along 1 dimension. On the car it is partially fixed to the brake pedal, meaning that either the driver or the actuator can press the brake, as can be seen in figure 5.5.



(a) Before activating brake

(b) Brake activated

Figure 5.5: Brake activated manually with the actuator constant at resting position

The actuator has its own black-box controller which unfortunately does not have feed-

back. It is made for elevating desktops with a manual "up/down" switch. As such it is fitted with a custom made controller which reads the feedback from the hydraulic brake circuit and commands the actuator to extend or retract, using a brake encoder sensor and hydraulic circuit sensor as input to the black box. This means that the home made controller has knowledge of the position of the actuator arm and the corresponding brake pressure level in the circuit.

The actuator itself is powerful enough to press the brake and keep it stable, however unfortunately the black box controller possesses several shortcomings.

1. It has generally been found to show irregular rise rime until the London competition where the system used was switched from closed loop to open loop

2. Furthermore, it is inaccurate. The black box controller has its own threshold of how large a step must be before the controller will activate the brake, meaning that each step must return to resting position before another step can be commanded

3. The brake also overshoots every step. Since the relationship between the physical position of the brake pedal and the actual brake pressure is non-linear the overshoot presses the brake significantly higher than commanded. This can be seen in figure 5.6, where the desired brake pressure is 5.8 bar but the brake overshoots to 11.68 bar, only to drop to 2.5 bar, as the relationship between the brake encoder (green) and the hydraulic brake pressure (red).

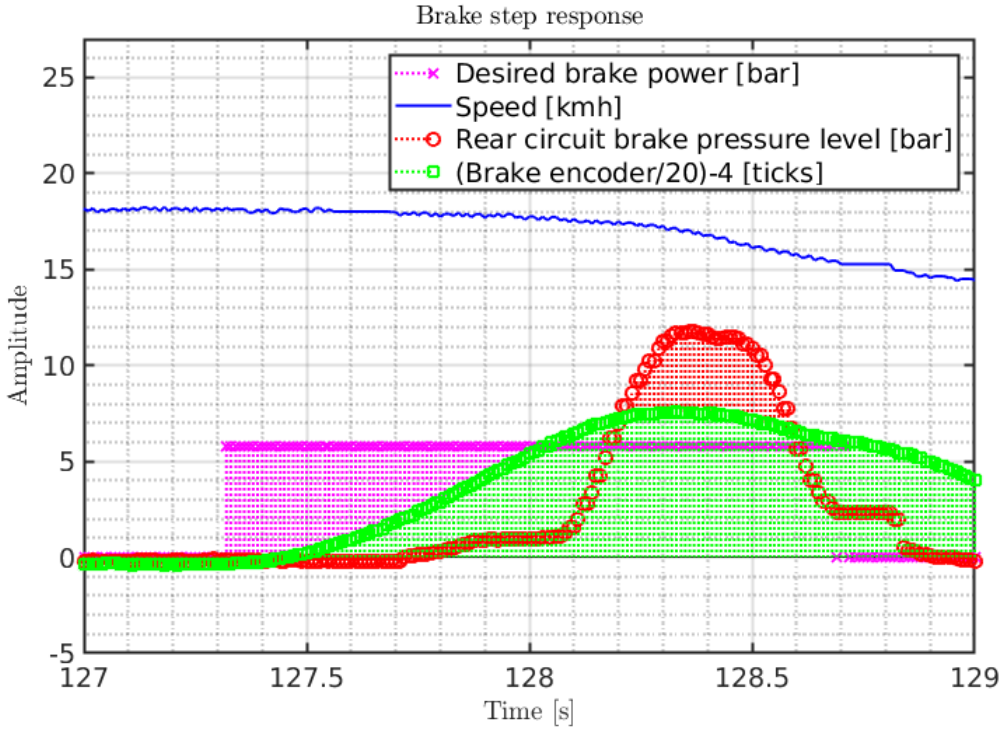



Figure 5.6: An on-line step response of the brake, taken from the final run in challenge 1 (t=127 to 129s)

Another complication is that if the relationship between hydraulic pressure in the front and rear brake circuit is changed the brake controller must be re-calibrated. As the pressure changes so too does the brake pedal position at which the braking starts and the pedal resistance becomes hard.

## Car engine

The engine is a single cylinder modified moped engine and is the propulsion of the car. On a normal car the throttle is controlled by controlling the air intake. On the ecocar the engine can only burn at a fixed air intake, and the ratio of fuel to air is pre-determined manually. This means that the engine works in one way only: coast and burn. When turned the engine "burns" to accelerate. When turned off the car coasts until burning again.

The engine uses a starter motor that connects with a bendix drive. This bendix drive is particularly sensitive. If a burn is initiated and stopped too soon the bendix drive will not necessarily connect and if the engine burned sporadically, the mechanical components can be damaged. Therefore it is not possible to PID control the speed for this engine. The engine can be seen in figure 5.7.

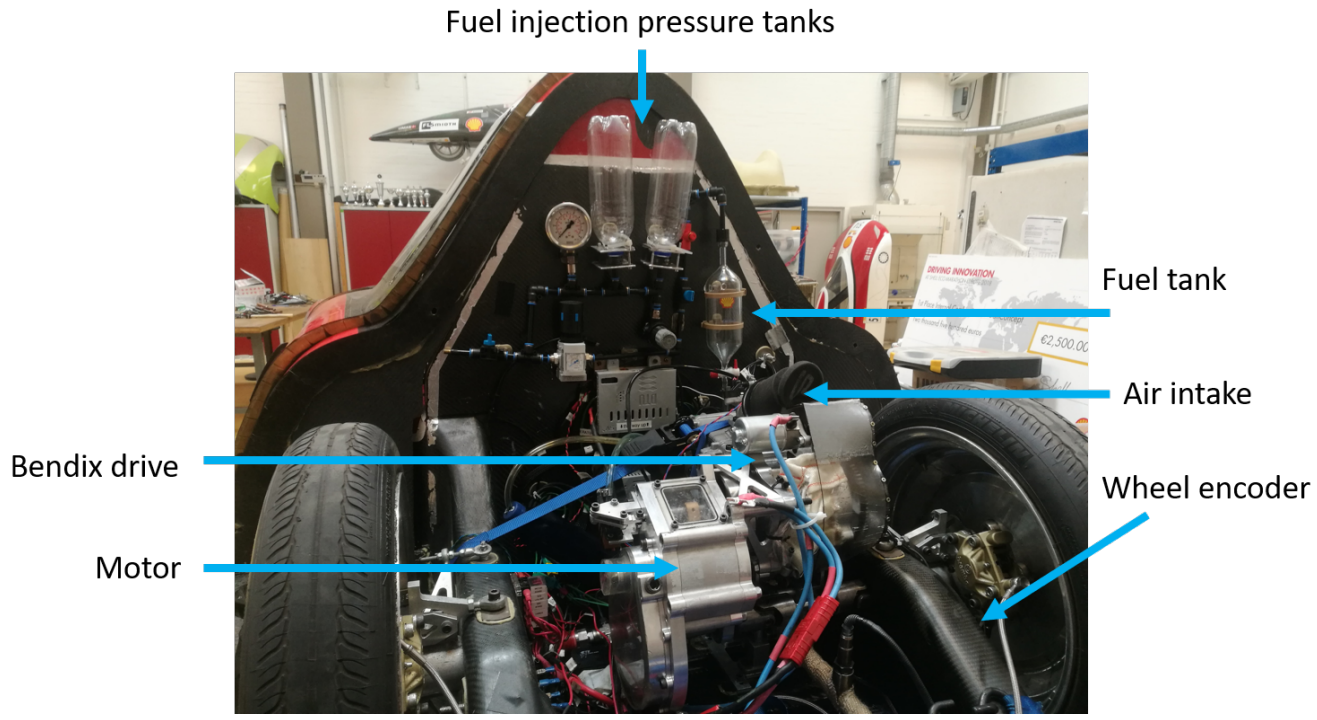



Figure 5.7: Overview of the engine

## Computer boards

The computing unit used is the Intel NUC which is a 64-bit mini PC with i7-8650U processor with a stated processing power of 1.9Ghz and 8Gb RAM. It is suited for running ROS on Linux and has ample processing power. Software run time is of great concern in this project as the entire autonomous path planning runs exclusively on software, and fast operations are needed to navigate a car in real time. This is further amplified by the desired top speed of 25kmh which is 7m/s. Even the right answer is wrong if it is found too late, and if the car is travelling at 7m/s and the track is only 6m wide it does not take long to crash the car.

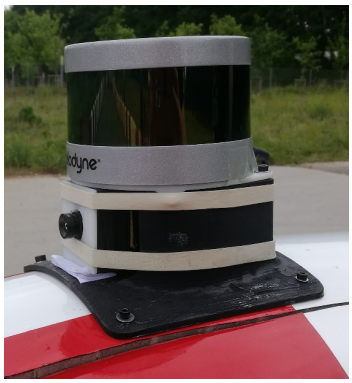
Aside from the NUC several other, smaller processing units and boards are also used.

## LiDAR and camera

The **LiDAR** is a Velodyne Puck Vlp-16, which is a 360 degree 3D LiDAR with the nominal range of 100m.[6] The vertical field of view is 30°. The rotation frequency can be set to either 5, 10 or 20Hz, however higher frequency means lower accuracy. In this project it is set to 10Hz as this is a good trade-off between speed and accuracy. Featuring a low energy consumption and low weight it is suited for this project.

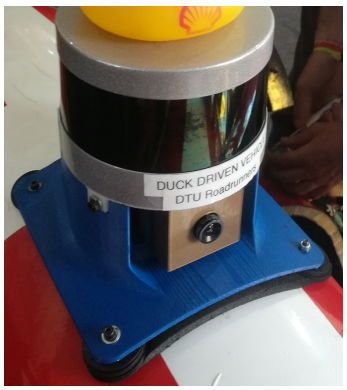
The **camera** is not covered in this thesis but the datasheet[14] can be found in Appendix A.5.

Both the LiDAR and the camera are mounted on a 3D-printed mount which can be seen in figure 5.8.



(a) Initial mount

(b) Final mount in Paris



(c) Final mount in London

Figure 5.8: The LiDAR on top of the mount, camera below

## Miscellaneous

An emergency switch which cuts off the power to the actuators is also installed, as can be seen in figure 5.9. The flip switch connects power to the entire autonomous system, which is separate from the car's normal power system. The emergency button disconnects power to the actuators only, and removes autonomous control of the engine.

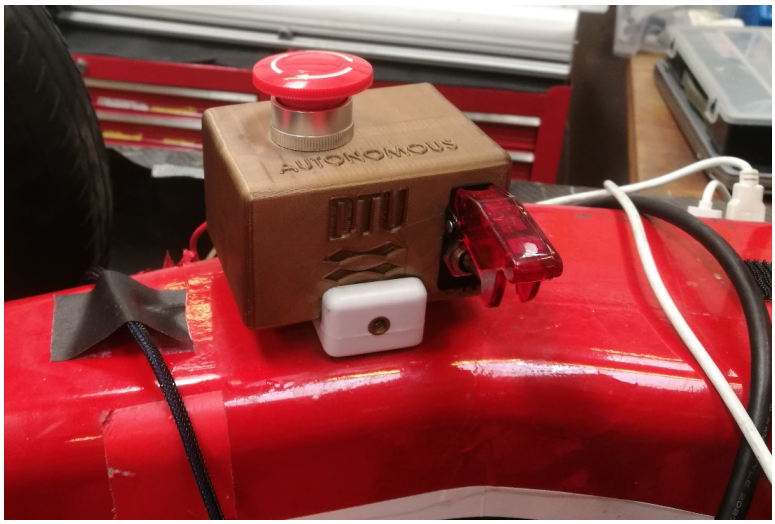



Figure 5.9: Emergency switch

The steering wheel, which can be seen in figure 5.2, posseses a number of buttons, including the **autopilot** button which activates and deactivates the autonomous program. When activated the lights on the steering wheel turn bright green and a small speaker announces activation to the driver.

# 6 Navigational Algorithm

In this chapter the navigational algorithm, how to go from LiDAR measurements to reference signals for steering and speed, is described.

## Determining Left and Right Side

Considering the features of road width and barriers that are explained in the rules from SEM, presented in the introductory chapter, determining left and right is possible and can be done simply by using the LiDAR on the Ecocar. Using the left and right boundaries of the road, should make it possible to determine the middle of the road.

In the beginning of every new mission driven on the Ecocar, a global coordinate system is established. Every data point from the LiDAR is transformed from this global coordinate system to a local coordinate system based at the middle of the rear-wheels. By doing this we can make sure, that the robot is always pointing in the x-direction. This is simply done by doing 2D-transformation as seen in equation 6.1:

$$\begin{aligned} x_{local} &= cos(-\theta_{odo}) \cdot (x_{meas} + x_{disp} - x_{odo})) - (sin(-\theta_{odo}) \cdot (y_{meas} - y_p) \\ y_{local} &= sin(-\theta_{odo}) \cdot (x_{meas} + x_{disp} - x_{odo})) + (cos(-\theta_{odo}) \cdot (y_{meas} - y_p) \end{aligned} \tag{6.1}$$

This way the LiDAR measurements, which are given in global coordinates, $(x_{meas}, y_{meas})$ transforms into $(x_{local}, y_{local})$ by using the current odometry of the robot $(x_{odo}, y_{odo}, \theta_{odo})$. Here, a $x_{disp} = 0.75$ is added to the x-coordinate because the LiDAR is displaced by $0.75m$ from the rear-wheels along the length of the car. Y-offset is negligible.
By ensuring a fixed coordinate system centered on the car, every data point with $y > 0m$ can be defined as left to the car and everything $y < 0m$ is to the right of car. If the road crosses over, like in a turn, then there is no longer clearly defined left and right. However, for all data points that are directly to the side of the car, left and right sides can be guaranteed, i.e. since the car is $2m$ from rear wheels to the furthest point on the front, it is apparent that when data also fulfill $x = [0; 2]m$ then it is *guaranteed* that this data can be defined as either to the left or right side according to the y-value. An example can be seen on figure 6.1.

After defining a fix-point for right and left side, using the fact that there were little to no gaps, $gaps < 1m$, between the barriers belonging to one side of the road, and knowing the minimum distance between the two sides of the road is at least $5m$, it is possible to classify the entire left wall and right wall into their own groups, called clusters.

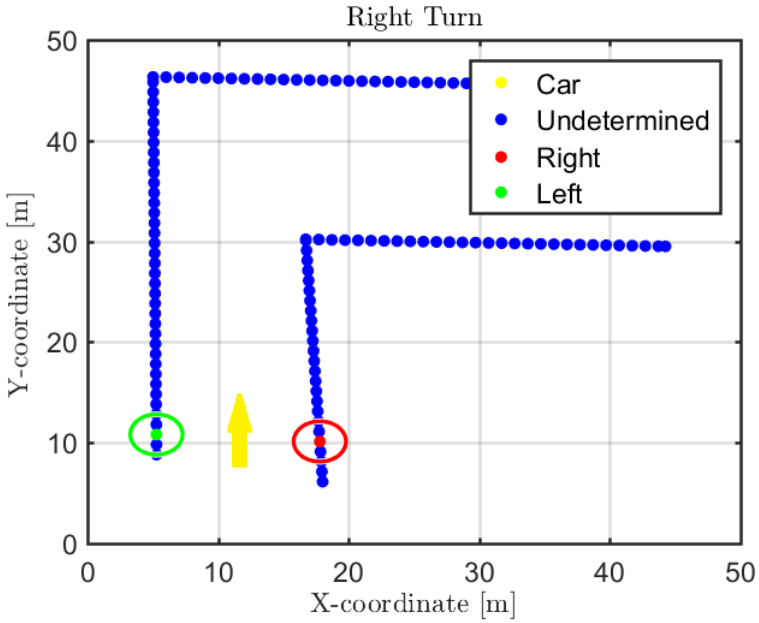



Figure 6.1: An example of determining left and right on a right turn, here it is seen that a starting point for, left and right points are found, and a circle with a radius of $2m$ is used to determine other points belong to the same cluster.

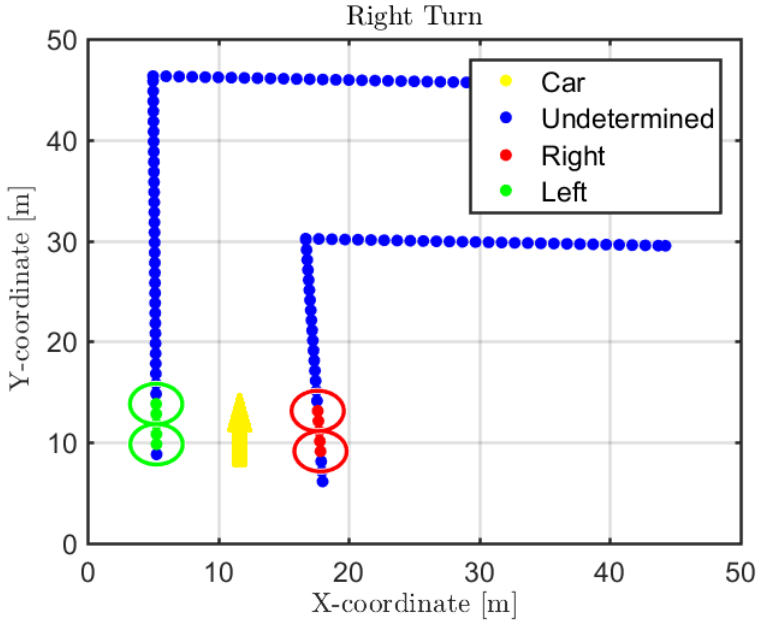



Figure 6.2: A few iterations into the left-right algorithm, the sides are getting clustered securely.

By choosing to merge any data point, with a distance less than $2m$ from an existing cluster, to said cluster, categorizing left and right data points can be done very efficiently. This means that starting at a point certain to be left or right, a circle with $radius = 2m$ is made. Any point falling under this circle is also in the same category. This is then done iteratively, until there are no more points that fall into either categories. See figure

6.1. After a few iterations the walls start to get defined, see figure 6.2, until finally the algorithm will have determined left and right wall like on figure 6.4. This way of merging barriers together, typically results in around $10m - 25m$ of left and right road, depending on elevation.

## Voronoi middle path

Another way to think about the middle path of the traversable terrain, is to think of the line that is the furthest from any obstacle. Luckily, something called a *Voronoi diagram* can be calculated, which contains exactly the needed information. This diagram partitions a 2D plane into cells by using euclidean distance between points. Looking at figure 6.3, an example can be seen of how a set of **points** is divided into their own region called a **cell**. These cells are found by calculating the euclidean distance from a point to all other neighbouring points.

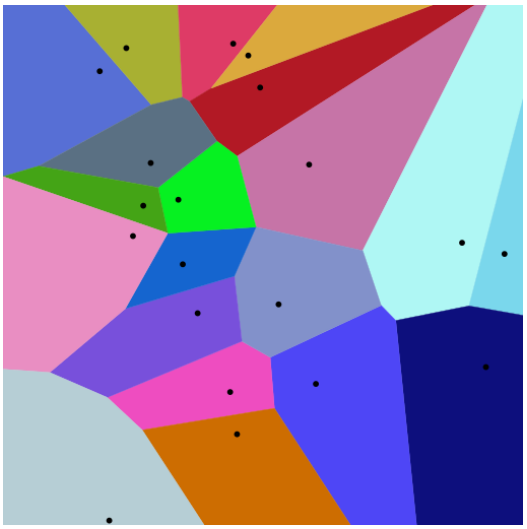


Figure 6.3: A Voronoi diagram of a random set of points. Each **point** is contained in a **cell**, distinguishable by color. The corners where 3 or more cells touch are called **vertices** and the borders between cells that spans from one vertex to another are called **edges**.(This picture is taken from *https://da.wikipedia.org/wiki/Voronoi_diagram.*)

A voronoi diagram will in our case be calculated from 2d-points that contain the position, $(x, y)$, of walls and obstacles, and when calculated, holds information about all the respective cells that belong to each point, their bounding edges, and the vertices that make out the corners of the cells.
A satisfying implementation of Voronoi for C++ is available by using the BOOST library[9]. This implementation uses a sweep line algorithm[1] to construct a Voronoi diagram in $O(n \cdot log_2(n))$ computation time[11].

To get an idea of how the voronoi diagram looks when applied to an organized set of points. Let us consider the piece of road previously seen on figure 6.1, after it has been processed with left and right it will look like figure 6.4. Here it is seen that there are two clearly separated and perfectly clustered walls. On figure 6.5, it is seen what happens

[1]Visual example of sweep line algorithm: https://www.youtube.com/watch?v=k2P9yWSMaXE

when applying MATLABs *voronoi()* function to the wall points, there are now blue lines indicating where the edges between the cells occur.

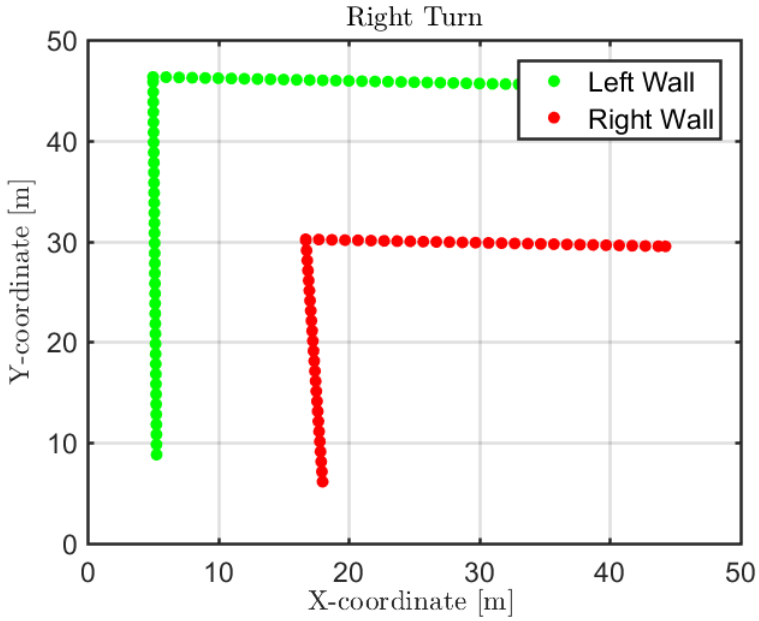



Figure 6.4: A turn is plotted with clearly separated and perfectly clustered walls

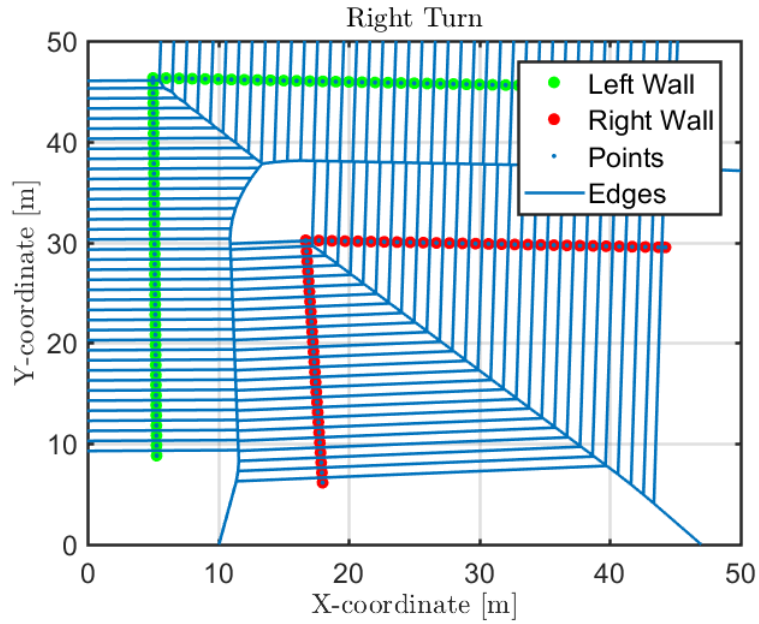



Figure 6.5: A turn is plotted with clearly separated and perfectly clustered walls. By using MATLABs *voronoi()*, the *edges* between the individual points *cells* are now are marked with blue lines. There is a middle path which can be extracted!

## Extracting middle line

There is a large amount of irrelevant information available in the voronoi diagram that can be thrown away, since this information includes all edges and vertices between all points detected by the LiDAR, as can also be understood from figure 6.5.
The voronoi diagram constructed from the BOOST algorithm is not sorted, but the diagram contains information about which points, cells, vertices, and edges that belong together. In order to extract only the middle line, which is the data of interest, the points can be segregated into clusters. By clearly defining each cluster as separate, i.e. left and right wall, every cell will belong to one cluster. Every cell, around points on the left wall, will belong to one cluster, and every cell around points on the right wall will belong to another cluster. it is possible to search through all the *edges* and save only those, that are boundary-lines between cells of two different clusters. To discretize this information, the *vertices*, (the end-points), of each *edge* is stored, corresponding to storing every (x,y) information available about the middle line. The middle line of the traversable terrain can now be extracted as seen on figure 6.6.

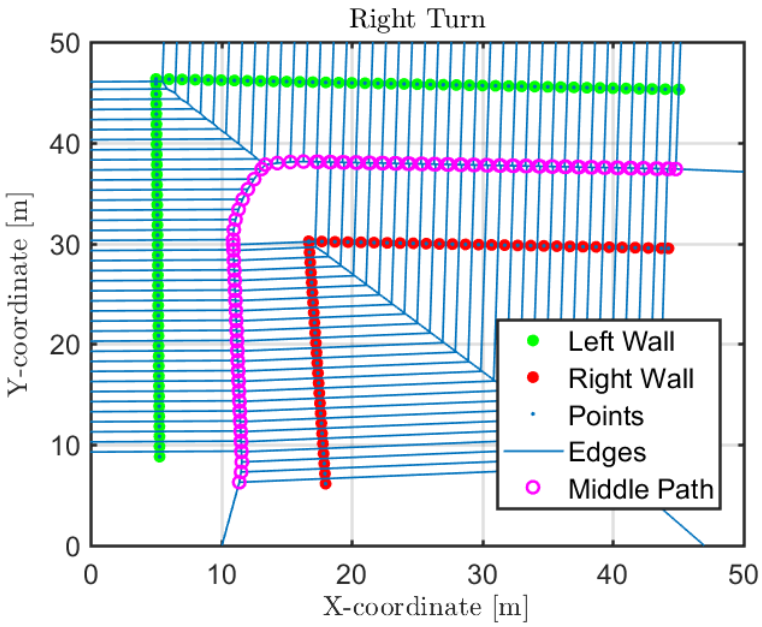



Figure 6.6: Here the voronoi data of the turn shown on figure 6.5 is now filtered by only looking for what happens at the boundary between two clusters.

Because the information about which vertices connect to each other is kept, by starting with the vertex that is closest to the car, it is possible to then go from vertex to vertex and enumerate each point making an ordered list of waypoints which the car has to follow.

A real life example of the voronoi results can be seen on figure 6.7, where a LiDAR scan from the London competition is shown, and the middle path is found. It is seen that significantly more points are selected than needed, which is fine, as the algorithm is only sensitive to too few points, and not too many, as long as the points are correctly classified as left or right.

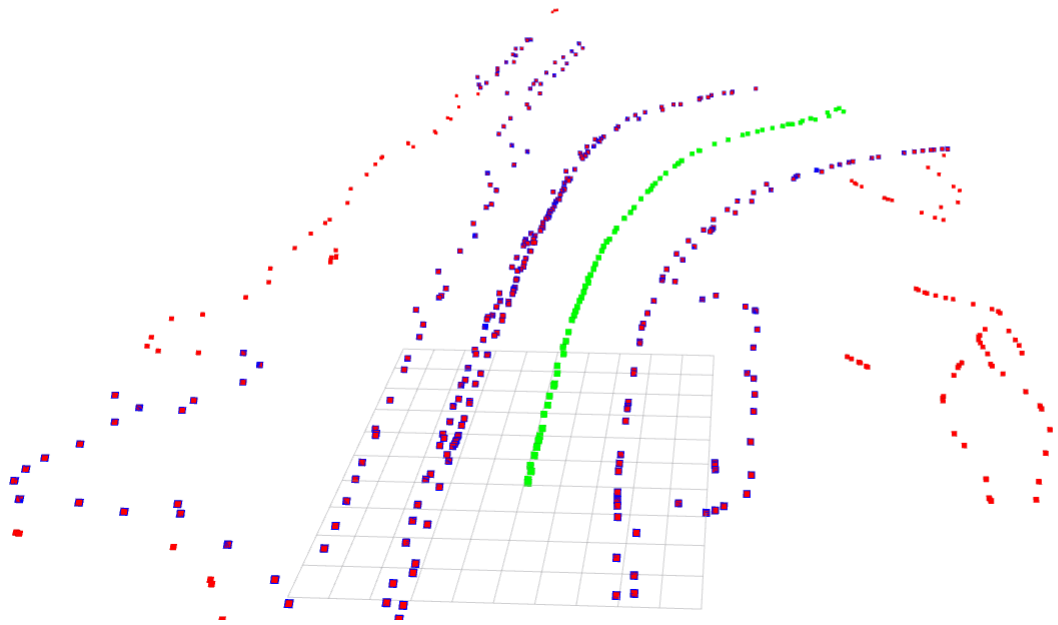


Figure 6.7: A LiDAR scan is made. The selected points for voronoi are colored purple, while the ignored points are colored red. The middle path vertices are then calculated and visualized as green points.

## Multiple paths

Since it is originally proposed from Shell, that there could be complex track layouts such as roundabouts or y-splits and similar situations on the road, it is necessary to include a solution for these type of situations where the car has to take either a leftmost or rightmost path.
The navigation described so far simply finds left and right sides, and calculates the voronoi diagram from these datapoints. This is done by having categorized left and right walls as belonging to their own clusters. Simply adding a third cluster called "obstacles" would solve the issue of having something in the middle of the road, that the voronoi diagram also needs to calculate middle paths from. Looking at figure 6.8, a roundabout is seen. Since there is already clearly defined left and right, additional relevant data points can be defined as those, that are enclosed by the left and right wall, i.e. within a certain distance of points belonging to both the left and right wall. Just like previously seen on figure 6.1, where a circle with a radius of $2m$, is used to cluster left and right wall points together. In the same way, a circle corresponding to the width of the road[2] from the left and side is used to determine obstacles. Obstacle detection is done after determining left

[2]Approximate road width should be evaluated before using this method of obstacle detection.

and right, and the results can be seen on figure 6.9.

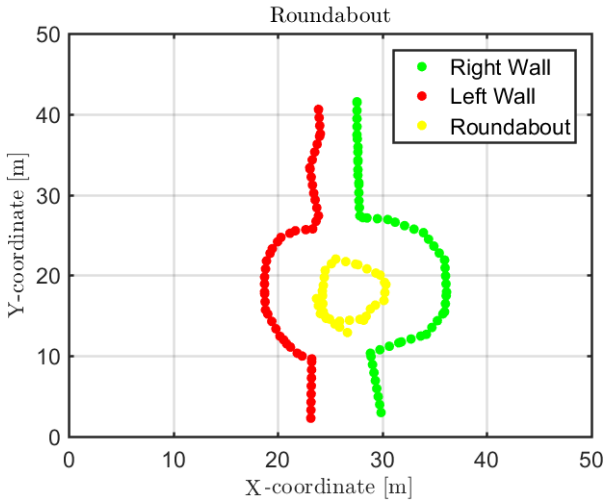



Figure 6.8: A roundabout is a more complex track layout, since there are obstacles in the middle of the road that needs to be considered.

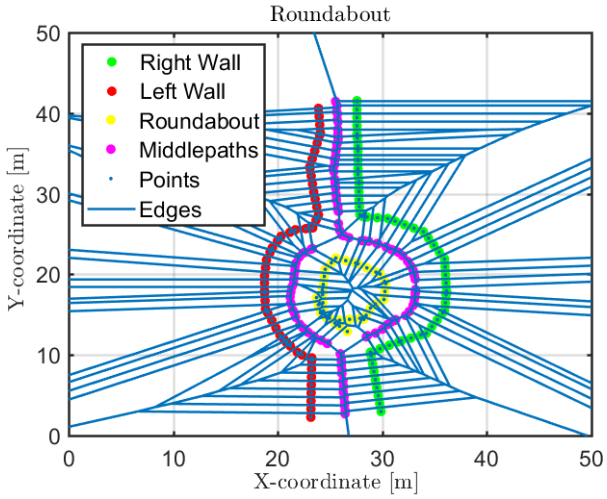



Figure 6.9: The roundabout is added to the voronoi diagram resulting in two paths.

In the program it is implemented that the car will either choose the leftmost or rightmost route. This is done by evaluating the angle of the trajectories leading out of the split by using the calculation seen in Equation 6.2. A split occurs at $(x_1, y_1)$, and the angle to the first point of each trajectory $(x_2, y_2)$ is evaluated. Since straight corresponds to an angle of 0° in the local coordinate system, and since angles are positive when moving counter-clockwise, it must be true that $\theta_{left} > \theta_{right}$.

$$Angle = atan2(y_2 - y_1, x_2 - x_1) \tag{6.2}$$

Therefore, in case of a split in the road, it is possible to either follow the leftmost or rightmost by simply choosing the highest or lowest angle at the vertex where a split occurs.

## Advantages of using Voronoi Diagrams

**Alternatives:** An intuitive and popular way to store data about a local map, is by making a 2D matrix called a **grid map**. One dimension is used for x-coords and the other for y-coords. When obstacles are observed, they are simply placed in the matrix at the corresponding position, and this information can be used to run different algorithms such as calculating heat maps or applying thinning algorithms to find the "*safest*" route, "*fastest*" route or whichever optimality criterion has been defined.
These alternative types of algorithms can also calculate the same results as voronoi, but use comparatively heavy calculations and are based on grids or graphs. Assuming such as in this case, that measurements are available in a 180° half circle at up to $50m$ distances, it would probably be desired to make a grid map that contains $30m \cdot 20m$ of grids/graph nodes. It would be possible to look $10m$ to each side and $30m$ ahead. Now assuming there is a desire for $10cm$ accuracy, $300 \cdot 200$ calculations are now required every time the whole 2D-map should be updated.
Of course optimization can be made, however if $1cm$ accuracy is suddenly required, the program execution time will increase dramatically. Trade-offs between run-time and precision should not be required, and the calculation time of an algorithm should be dependant on the amount of data available and not the predefined grid or graph.
**Why choose Voronoi:** The voronoi diagram contains the necessary information to calculate the middle path of the road, and since the optimal position has been determined to be in the middle of the road, to solve the demands for this project, the output of Voronoi matches the requirements for position perfectly.
It can be constructed in $O(n \cdot log_2(n))$ time, and it is incredibly important to keep execution times fast, or even the most perfect calculation is unusable. When compared to other algorithms, such as grid based algorithms like thinning or heat-maps, runtime is significantly better, as usually these other algorithms run in $O(n^2)$ speed and have run-times based on the precision, that he map is defined with. Voronoi diagram is based on the distance between the points, and is therefore not bound to a map based structure. For Voronoi the only requirement in order to gain excellent run-time and precision is filtering the irrelevant information away, by means of defining left and right, or choosing routes based on waypoints and algorithms such as "*dijkstra*", "*breadth first search*" or similar path choosing algorithms.

## Conclusion to the Navigation Algorithm

Entering a 90° turn with a speed of $4m/s$ on a road with a width of $6m$, means that the vehicle will crash into a wall in $1.5s$. Taking into account that the steering stepper can turn the wheels $20°/s$ to a maximum of 14°, this gives $0.6s$ less time to react. Remembering it is desired to stay in the middle of the road gives further limitation to reaction time, the authors have therefore defined a maximum execution time of $100ms$ as a conservative but realistic time limit for program execution.
By using a Voronoi diagram approach it is possible to calculate a path that is perfectly in the middle by using a set of distance measurement, that has been categorized to belong to the left or right side of the road. This can be done in $O(n^2)$ time, and most of the data processing, such as constructing the diagram, is done in $O(n \cdot log_2(n))$ running time, with $n$ being distance measurement data points. With the chosen parameters for

merging walls of left and right, the typical road visibility is around $10m - 25m$ depending on flatness of surface and the execution time is less than $50ms$ on the Intel NUC placed in the Ecocar, meaning even at top-speed of $7m/s$, the car will have a new route after a displacement of only $35cm$. It is also possible to deal with more complex track layouts such as roundabout or Y-splits, by choosing to go either left or right.

# 7 Post-processing

Driving the car autonomously consists of setting a reference angle to steer as well as setting a reference speed, as an essential problem to autonomous driving is knowing when to speed up or slow down. Calculating this decision on-line takes a certain processing time. Furthermore, the actuators take time to change their physical values, as defined by the hardware limitations. It is thus desired to know the desired trajectory as far ahead as possible. The speed of the car must be controlled according to the curvature of the route; it is desired to slow down when approaching a turn and to speed up when driving a straight stretch.

During this project the approach to regulate speed has been a simple switch between a pair of normal "slow" speeds and a pair of fast speeds. Due to the constraints of the car engine already described in the Hardware chapter. The motor control used is a simple hysteresis loop to accelerate to a given speed (maxspeed) followed by turning off the engine and coasting to a lower speed (minspeed). The slow maxspeed is defined as at the **critical speed** which is the maximum speed that car can safely complete a 90° turn with a given turn radius. In practical use the max speed is slightly below the critical speed as the engine has been found to overshoot with approximately 1kmph when burning.

The speed used in the London competition is found empirically, although it naturally depends upon the radius of the turns on the particular track. For safety and comfort the decision to switch between slow and fast pairs of speed is desired to be known as early as possible in order to brake down in time. Post-processing of the Voronoi route provides a solution to this problem.

It is desired that the car steers smoothly along a route, however after the middle path has been found the route is ragged and twisted, and driving along this route is not comfortable due to steering fluctuations. This problem can be solved by a processing of the route to make it suited. Such a processing must effectively reduce steering fluctuations.

The processing can be done in different ways, e.g. by viewing the problem as a geometric problem and using certain mathematical operations.

In this project path planning is viewed as a 2D-geometrical problem. As such, the method used is certain algorithms and formulas which geometrically soften the curvature of the route in order to decrease steering fluctuations along a given route. This has proven a good idea due to being fast and effective. In order, the process steps are:

- Statistically comparing the current route to a moving average low-pass filter

- Smoothing the route by fitting it with a 5th degree polynomial

- "Cutting" the route with a straight line. Here the speed control input is also determined

## Post-processing data

In the following the above steps are explained in detail.

## Statistically comparing current route with route bank

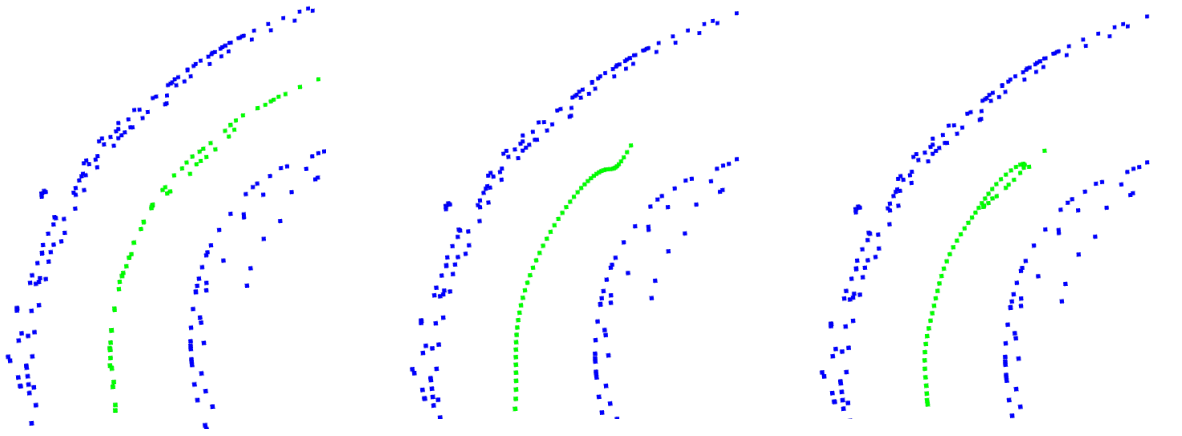
Prior to route smoothing a statistical comparison is made between the current route and a bank of routes. The bank is a low-pass filter based on measurement averages of points from the 4 previous routes. Every sample the current route is found from the Voronoi graph, and each point in the current route is added to the route bank. To sort out duplicate points the Euclidean point distance is used; if the point distance between a current point and a bank point is smaller than 0.2m the two points are merged to the summed average of the two.

The bank then contains both current and previous drive points from the last 4 samples. If a point in the bank is not merged with a current point in 4 consecutive samples then the bank point is discarded. Since the algorithm is executed every 50ms a single measurement is discarded within 250ms.

## Curve smoothing and fitting

After the bank comparison the route is still ragged and twisted. To smooth it, the points are fitted to a 5th degree polynomial using Gauss elimination[13].

This results in a smoother route while keeping the information of the curvature of the route. Any degree above 2 can be used, but a 5th degree polynomial is a good approximation as lower degrees have shown to be too inaccurate, and values higher than 7 degrees can "force" the curvature of the route into undesired shapes, and has been found to not be necessary. In figure 7.1 the difference between no smoothing, 5th degree smoothing and 11th degree smoothing can be seen. Another advantage of the polynomial fitting is that an equation of the route is acquired.



(a) No smoothing (b) 5th degree smoothing (c) 11th degree smoothing

Figure 7.1: Route with different levels of smoothing (final challenge 1 at t=29.81s)

# Finding a line to drive after

After the curve smoothing it is desired to linearly straighten the curvature of the route to further decrease steering fluctuations. To do this a linear regression is made on the points within 4m of the front of the car which can be seen in figure 7.2.

The result of the post-processing is a single $(x, y, \theta)$ value which is then sent to the linedrive node. These three values are called a *drive point*, although a better name would be a *drive line*. The point concists of two coordinates, x and y, with an angle $\theta$. Together they represent an infinite line with direction $\theta$ relative to the car's odometry.

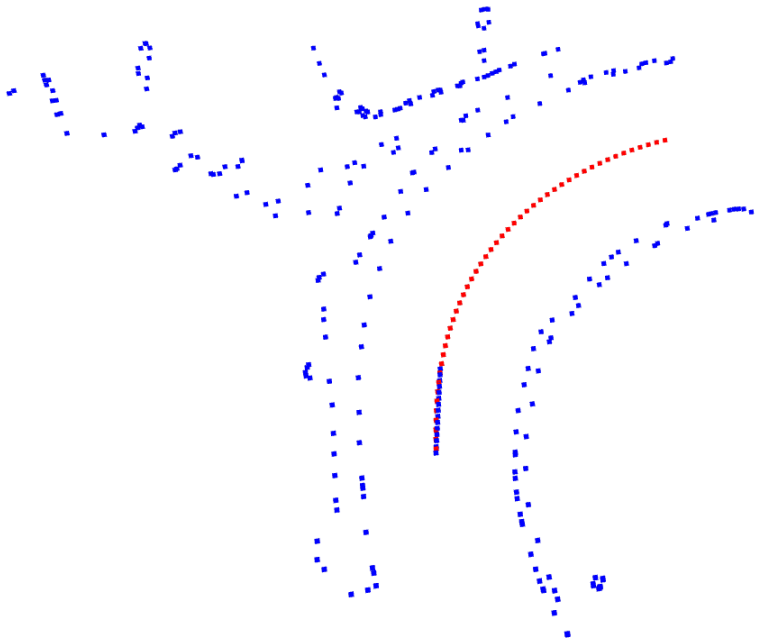


Figure 7.2: The 5th degree smoothed curve in red, with the linregged curve in blue at the start of the curve (final challenge 1 t=23.39s).

## Speed control

Aside from straightening the upcoming 4m, another linear regression is made on all available points to determine whether the route is going straight forward or if a turn is coming up. This simple calculation is the whole decision for whether fast speeds can be used or not. When switching between fast and slow a small speaker announces the new state to the driver.

## Conclusion of the post-processing

The post-processing effectively solves the problem of steering fluctuations and jagged routes. As can be seen in the video from the final challenge 1, the steering is smooth. Another advantage of the polynomial fitting is that an equation of the route is acquired, which can be used to detect turns more intelligently than the linear regression.
By post-processing the middle path, a line is constructed that can be used as reference for a steering controller.

The speed control is a simple compromise to achieve the target average speed of 18kmph while still keeping safety a priority since the slow, critical speed is the fallback speed in the program.

# 8 Steering Control

In this section it will be discussed how a controller can be implemented that dissolves a line into two error signals and uses this error signal to calculate a steering signal for the steering wheel.

## Introduction to Control-problem

The Ecocar is rear-wheel driven, front-wheel Ackermann-steered, which means that directional control is done by setting a front wheel angle, and can be done separately from propulsion (unlike vehicles with steering methods such as differential drive). Also it should be mentioned that there is no reverse driving in the car, which means there is no need to consider anti-Ackermann steering.
According to the mechanical team, there should not be slippage at the relatively slow speeds that are used autonomously, i.e. for maximum speeds of $7m/s$.
The Ecocar is mounted with a step-motor to control the steering wheel, and another student, has developed a PID-controller making it possible to set a setpoint in terms of degrees, therefore, to make the car drive according to the lines that have been calculated in the navigational algorithm, it is simply required to make a controller that can translate features of a line, into a steering signal.

## Construct Error Signals

Since the navigation result is parameterized as a line, the controller was based on the line-drive controller, which is implemented on the **SMR**, that are used for teaching purposes in the courses: *31385 - Autonomous Robot Systems*, and *31388 - Advanced Autonomous Robots* at DTU. The line-drive controller dissolves a line into two error signals, position-error and angle-error. Suppose that the line is parameterized as $(x_l, y_l, \theta_l)$, meaning that the line goes through $(x_l, y_l)$ with the angle $\theta_l$ and the car position is currently $(x_c, y_c, \theta_c)$, meaning the car is currently placed in $(x_c, y_c)$ pointing in direction $\theta_c$ then the two errors can easily be calculated. Firstly the angle error is calculated as seen in equation 8.1, by simply taking the difference between reference angle and current angle.

$$e_{angle} = \theta_l - \theta_c \tag{8.1}$$

Secondly, for the position error, the normal vector to the line is found first by using

the angle of the line to construct a normal vector, as seen in equation 8.2. Here.

$$\begin{aligned} v_x &= -sin(\theta_l) \\ v_y &= cos(\theta_l) \\ c &= -(v_x \cdot x_l + v_y \cdot y_l) \end{aligned} \tag{8.2}$$

After constructing the normal vector, it is easy to find the shortest distance from the car to the line by using equation 8.3.

$$e_{dist} = x_c \cdot v_x + y_c \cdot v_y + c \tag{8.3}$$

The controller is then implemented as a MISO (multiple input, single output) P-controller, this means the controller will use both errors with each their own gain constant $K_{angle}$ and $K_{dist}$ to output a setpoint steering angle, that can be sent to the steering wheel. This was done by as seen in equation 8.4.

$$\theta_{steer} = K_{angle} \cdot e_{angle} - K_{dist} \cdot e_{dist} \tag{8.4}$$

The signs in equation 8.4 are made such, that the controller will find an equilibrium between angle error and distance error. i.e. the car will drive unto the line if:

$$e_{dist} \cdot K_{dist} > e_{angle} \cdot K_{angle} \tag{8.5}$$

Furthermore, $e_{dist}$ was limited such, that if the reference line supplied was too far away, the car does not just start circling around. It is necessary to include this limit, because there is essentially no limit to how high values of $e_{dist}$, that can be supplied. At a certain point, $e_{dist}$ will completely dominate the controller, and the car will only drive in one direction.

# Choosing a P-controller

The reason for choosing a P-controller over a PI,PD or PID controller is primarily that a *derivative* controller is sensitive to high frequency signals, meaning that even if a satisfactory derivative, that could predict the trend of the error, could be calculated it would amplify measurement noise, and integral controllers are primarily used in situations where steady state errors are unacceptable or the system needs integration, because a too large P-controller signal would be needed to minimize steady state error.
Even if a PI-controller or an even more complex controller can do a better job, after trying a P-controller first, it was decided that it was not currently worth the effort and time to make a more complex system, since the P-controller worked satisfactory and is easily tuned and it is significantly easier for other students to build upon a simple structure with few inputs such as the line-controller, than a complex controller. It has to be remembered that, when the mechanical parts to the car is changed every year, the dynamics of the car will change, and the controller parameters will need to be tuned again.

# Kinematic model

To find the correct values for the gains, a kinematic model of the car was developed for MATLAB. By doing this, a step-solver for the controller could be used in conjunction with the kinematic model, to give an estimate of the parameters for the controller.
The Ackermann steering of a car with four wheels can be simplified to a model of a two-wheeled vehicle, such as a bicycle. If assuming constant speed, it is suddenly quite simple to make a kinematic model. Suppose the sampling time of the controller is known as $t_s$, and a speed is known as $v$, then the distance travelled in between each sample from the controller will be simply

$$d = t_s \cdot v \tag{8.6}$$

Now, according to Ackermann-steering theory[12] the vehicle will move in a circular path as can be seen in figure 8.1, depending on the angle of the front wheel $\theta_w$. The radius, $r$, of this circle given by

$$r = \frac{L}{tan(|\theta_w|)} \tag{8.7}$$

where $L$ is the distance between rear-wheel and front-wheel, which is in our case $1.516m$.

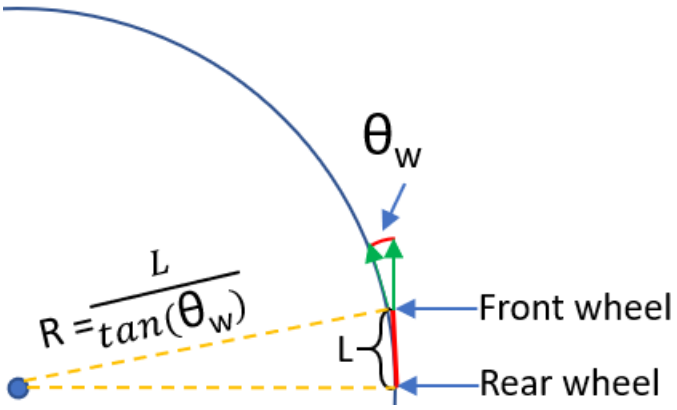



Figure 8.1: The car is modelled as a two-wheeled vehicle, and it is seen, that when the front wheel is positioned in angle $\theta_w$ $[rad]$, a circle movement with radius $r$ $[m]$ is created. This $r$ is found with knowledge of the distance between the rear and front wheels, $L$ $[m]$.

With knowledge of the radius for the circular movement, it is then easy to calculate the circumference, $cf$ as

$$cf = r \cdot 2\pi \tag{8.8}$$

and then get the amount of radians, $rad$, travelled along the circle as

$$rad = \frac{dist \cdot 2\pi}{cf} \tag{8.9}$$

Now by looking at figure 8.2, it is apparent that since the car is moving along the circle circumference, the changes to the local coordinate system are given as:

$$\begin{aligned} dx &= sin(rad) \cdot r \\ dy &= signum(\theta_w) \cdot (1 - cos(rad)) \cdot r \end{aligned} \tag{8.10}$$

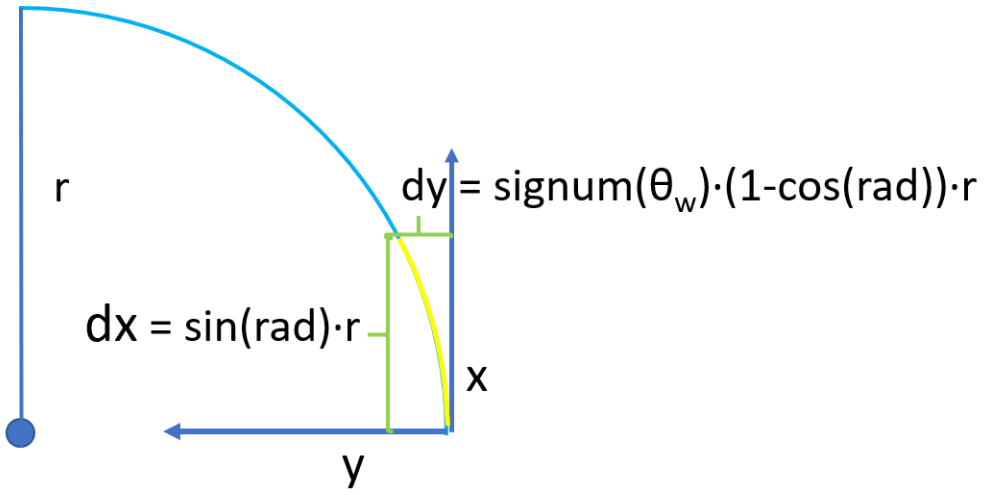



Figure 8.2: The car drives in a circle. The yellow path indicates the radians, that are driven in the time $t_s$, and the changes to x and y coordinate, can now be calculated with knowledge of $\theta_w$ and the radius, $r$.

Lastly, by transforming the local-coordinate changes $(dx, dy)$ into global-coordinate. The new position, can be calculated with knowledge of the old position $(x_{p1}, y_{p1}) \rightarrow (x_{p2}, y_{p2})$ by using equation 8.11, a simple kinematic model of the car is complete, and it is now possible to give a time, $t_s$, a speed, $v$, and a wheel angle $\theta_w$.

$$\begin{aligned} x_{p2} &= cos(\theta_{p1} + \theta_w) \cdot dx - sin(\theta_{p1} + \theta_w) \cdot dy + x_{p1} \\ y_{p2} &= sin(\theta_{p1} + \theta_w) \cdot dx + cos(\theta_{p1} + \theta_w) \cdot dy + y_{p1} \\ \theta_{p2} &= \theta_{p1} + atan2(dy, dx) \end{aligned} \tag{8.11}$$

## Tuning Controller Gains

To tune the controller to work on the ecocar, what is tuned is the ratio: $K_{ratio} = \frac{K_{dist}}{K_{angle}}$, and the absolute value of these controller gains as well. The general idea is that a high $K_{ratio}$, prioritizes to minimize the angle error to the reference line, while a low $K_{ratio}$ prioritizes staying exactly on top of the reference line. By raising the absolute value of these controller gains, the car will steer harder. Some risks to watch out for is that if the gains are set too high, because the steering wheel can not instantly respond to commands, and the calculation for reference steps are not instant, steering can become unstable, when the car starts to move at increasing speeds. Especially if the road is bumpy, which can lead to noisy measurement signals, or if similar disturbances are introduced.

A step-solver is set up as seen in figure 8.3, and the controller is saturated to only have maximal angle limits of 14° and a maximum change of $20°/s$. These values were observed in a step response also mentioned in the Hardware chapter.

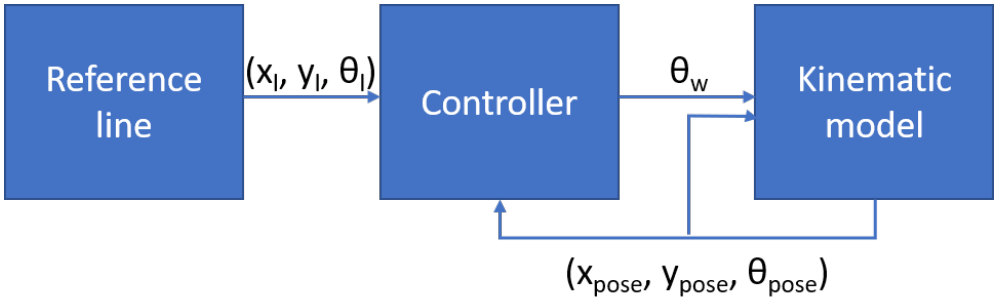



Figure 8.3: The step-solver is constructed by linking together a reference signal, the controller, and the kinematic model.

Because of the limits to the actuator change rate, it is important to tune the controller gains to the speed that is considered. E.g. if it is wanted to do precision driving, the best results are achieved at low speeds, since the actuator can be pushed to its limits. On figure 8.4, the difference can be seen between going $5m/s$ and $1.5m/s$, for equal controller gains.

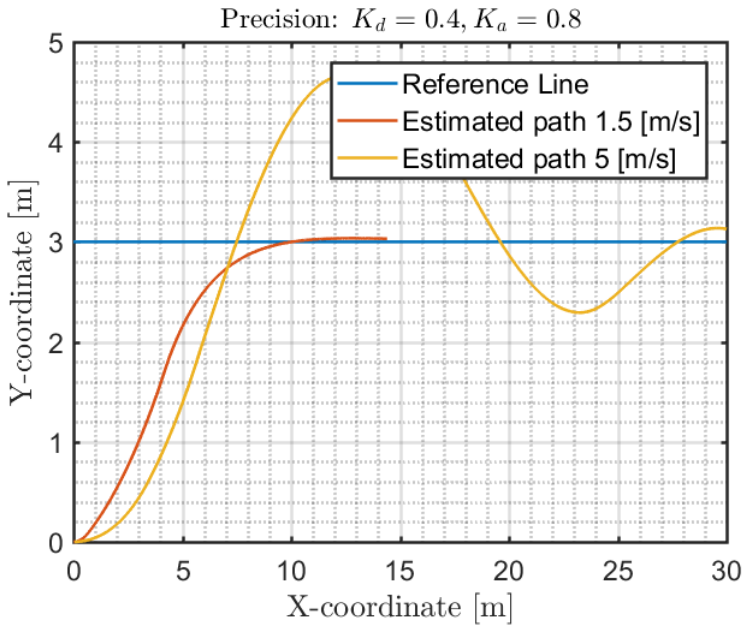
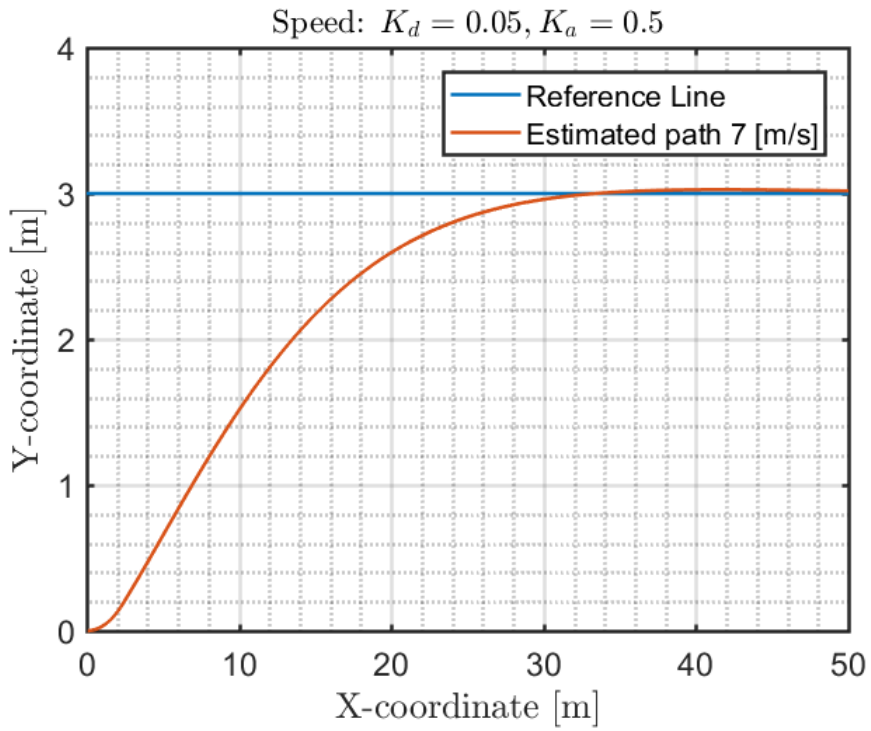
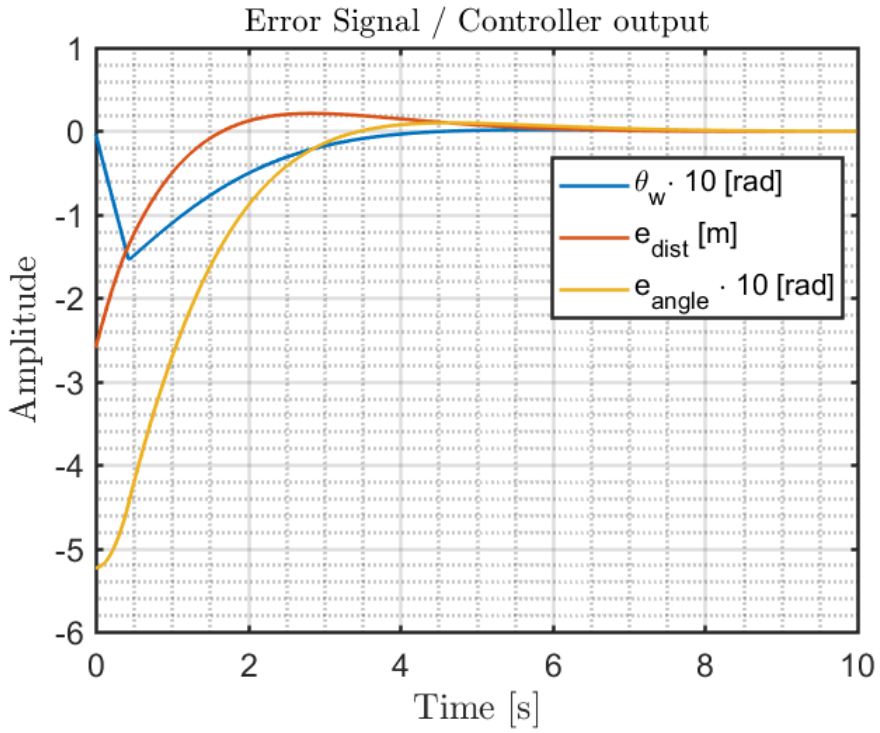



Figure 8.4: Here the same controller parameters are tried for different speeds. The system is stepped with a parallel line, that has an offset of $y = 3m$.

For higher speeds, it was then decided that the controller gains should be reduced, to allow soft control, that does not reduce error as fast, but does not overshoot or start oscillations, assuming that the road is continuous and the car has already found the middle line, this should result in robust steering. It is suggested that a tune such as seen on figure 8.5a is made, where the car can reach speeds without concern for oscillations. The speed of the car can simply be reduced if the $e_{dist}$ becomes too large. It is seen on figure 8.6a that even for greater angle changes, the controller is able to follow the reference angle.

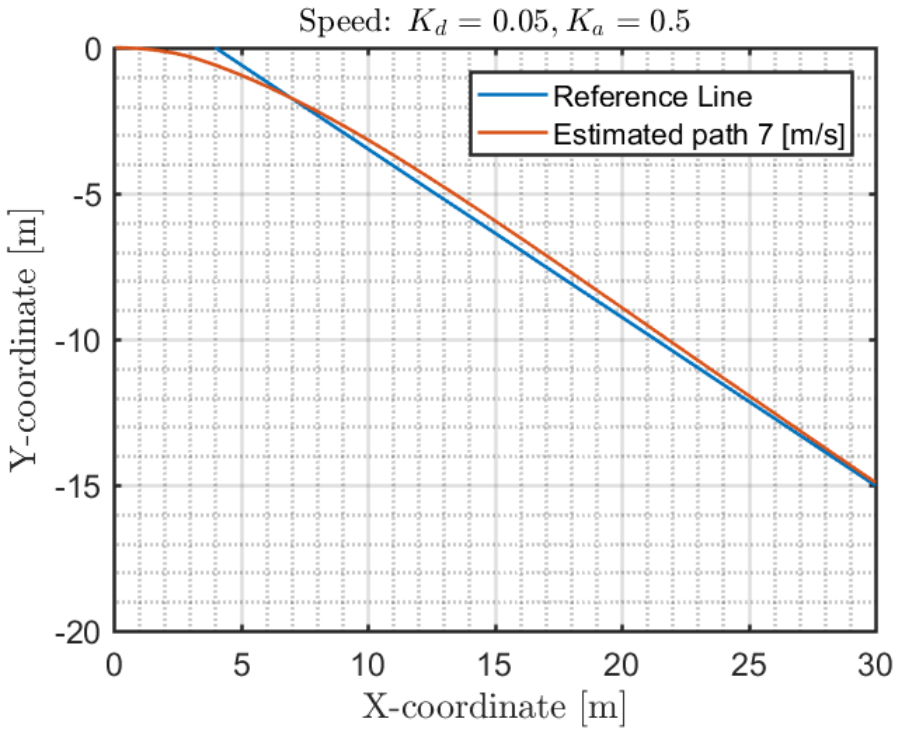



(a) For higher speeds such as $7m/s$, the controller gains should be relaxed. Here, the system is stepped with a parallel line, that has an offset of $y = 3m$. After driving $30m$ the gap is closed.

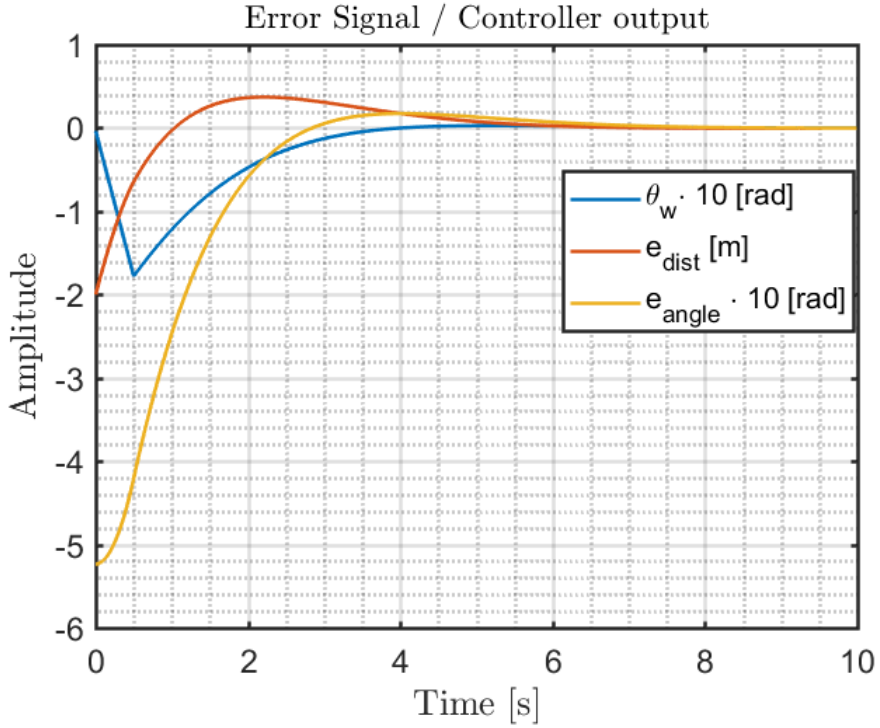



(b) The error signals, and controller output for a step with such as seen on 8.5a.

Figure 8.5: Tuning for speeds of $7m/s$




(a) Here, the system is stepped with a line starting at $(x, y) = (4, 0)$, with an angle of $\theta = -30°$.




(b) The error signals, and controller output for a step with such as seen on 8.6a.

Figure 8.6: Tuning for speeds of $7m/s$.

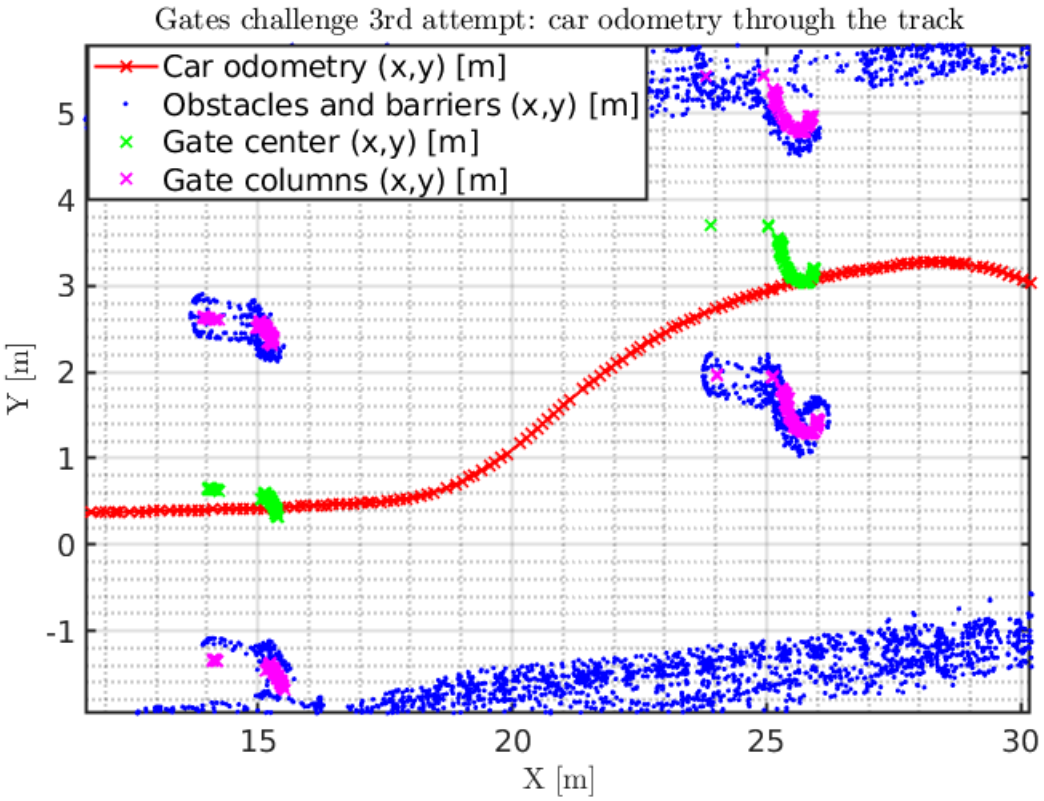
# Chosen parameters for competition

For the competition the values for controller parameters were chosen without the kinematic model, because a mistake was found in the model close to the competition. The chosen parameters can be seen on table 8.1.

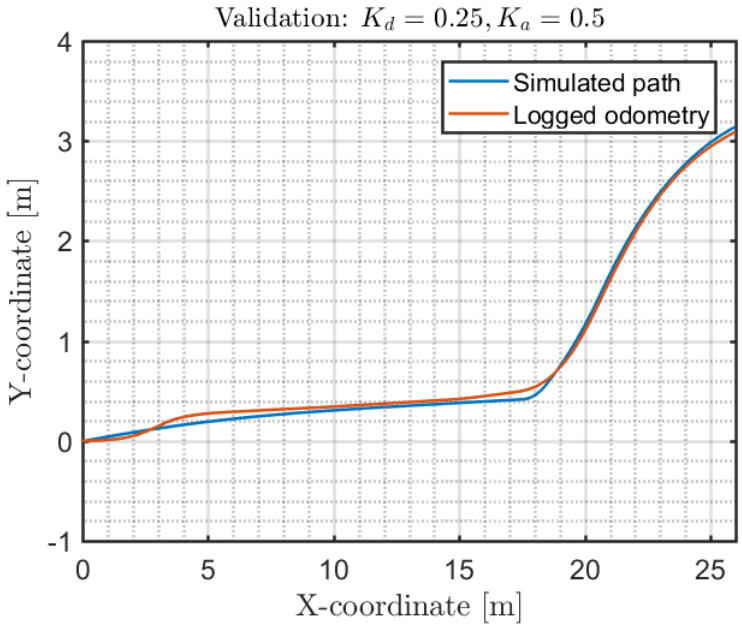
| $K_{dist}$ | $K_{Angle}$ | $Speed_{low}$ | $Speed_{high}$ |
|---|---|---|---|
| 0.2 | 0.5 | 0 | 10.5 |
| 0.025 | 0.4 | 10.5 | 14.5 |
| 0.01 | 0.2 | 14.5 | 20.5 |
| 0.005 | 0.1 | 20.5 | No Upper limit |

Table 8.1: A table containing the controller values used in on the car in London

The kinematic model has since been validated, by evaluating the log files from the competition in London and it has been possible to correctly predict the behaviour of the car when equipped with controller. On figure 8.7 the third run on the gate challenge is seen. Here, a step of $3m$ in the y-direction is seen, and the actual trajectory of the car, can be predicted with the simulation, using the same controller-gains. The references are slightly different because as can be seen on figure 8.7a, the gate center keeps changing, which means the real system has many lines with slight differences, while the simulation is only stepped with two of the given reference lines.




(a) A step of $3m$ offset in the y-direction was given during the gate challenge in London.




(b) Simulated step-response vs logged odometry. when given the same references and controller gains.

Figure 8.7: Validation of model.

# Conclusion to Steering Wheel Control

By using a simple P-controller, it is possible to set the steering angle for the Ecocar to be precise and without oscillation even for speeds up to $7m/s$ , when using lines as references. Because the controller accepts a simple input, it should be possible for any navigational method to supply the needed $(x, y, \theta)$ reference. This results in easy integration with parking spot and gate detection.
A kinematic model has been made and validated, to make the tuning of the P-controller possible by simulation. The kinematic model can be used even for other robots or if a new car is made, as long as the parameters: *steering limits, steering rate limits* or *distance between rear and front wheels* are changed in the model accordingly. If these are changed on the physical system, new controller gains should be found. The kinematic model source code for MATLAB can be found in the included source code CD.

# 9 Program structure

The following is an overview of how the software is structured. Figure 9.1 shows how the navigation and driving is divided. The Voronoi node calculates the Voronoi graph and performs the post-processing and outputs a single drive point (from the linear regression) to the *linedrive* node which in turn calculates the steering error from the car's odometry to the drive point.

### Special cases: Parking and Gates

During the gates challenge the gate classifier node publishes the center coordinates of the gates as drive points. During the parking challenge the the parking node receives the center coordinate of the parking spot rectangle from the *parking spot detection node* (which is not covered in this thesis) and publishes this coordinate as a single drive point. The Voronoi node also receives this coordinate although only using it as an indicator to stop publishing normal drive points.

The connection between these nodes can be seen in figure 9.1.

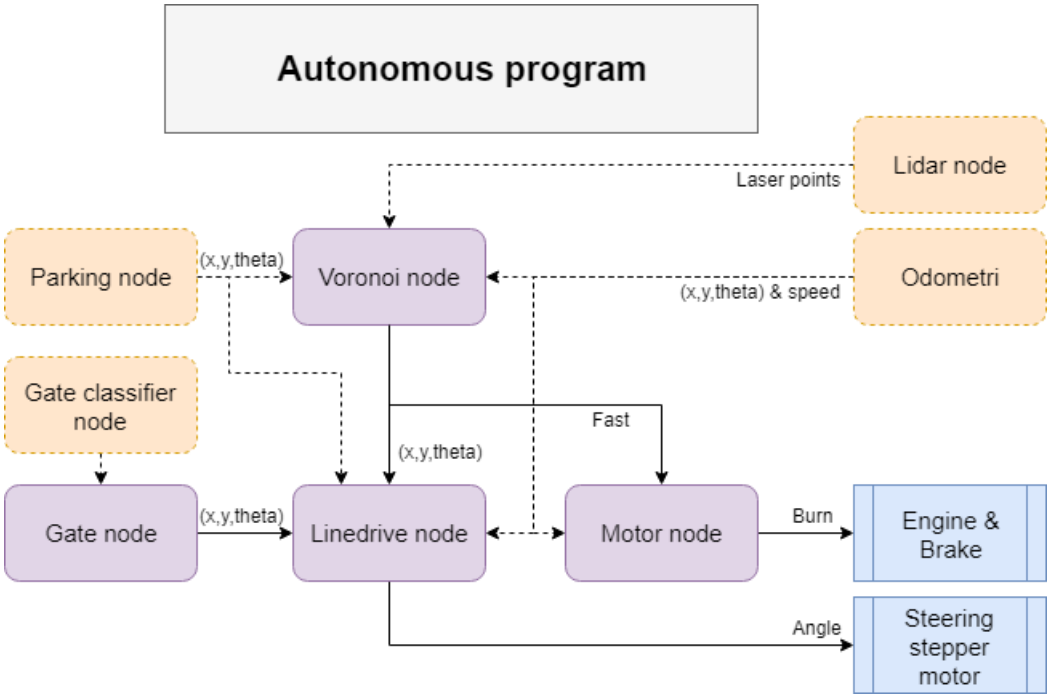



Figure 9.1: Overview of the program nodes, dotted boxes are not in the scope of this thesis, but provide inputs that are used.

The motor node performs the speed control using only the current speed of the car. If the speed is above the pre defined **maxspeed**+1.5kmph then the brake is activated until this no longer is the case. If the speed is below **minspeed** then the engine should be burned until maxspeed is reached. If none of these conditions are met the car coasts. This decision making process is illustrated in figure 9.2.

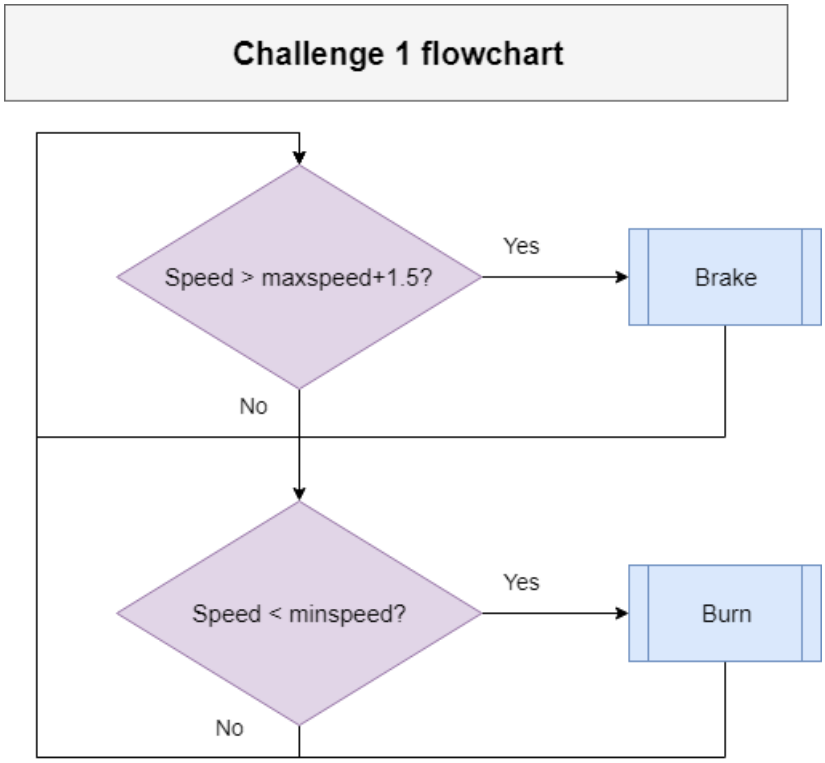



Figure 9.2: Flowchart for the speed control

# 10 Testing

Testing has been done two ways: program development by using Gazebo to validate function of algorithms, and by testing in real life on a test track to validate practical performance. This section describes how the autonomous system has been tested in real life on the car. The testing has sought to imitate the challenge track layouts as close as possible. Driving closed loop with lidar node, Voronoi node, linedrive node, brake node and the motor node is the core of the autonomous program. When testing there is an emphasis on turns and obstacles as these pose a greater challenge due to the limitations of the car's turning radius as well as the steering motor.

The following is the conclusion to the testing, followed by an in-depth analysis of each test.

## Conclusion of tests

After the tests have been completed the autonomous system has demonstrated ability to drive and steer to complete challenge 1, both with regards to distance and speed in sharp turns. The turn used in the speed test has a turn radius of 7.5m and the car completed this with 25kmph.

For the parking challenge all the different programs have been tested, and starting in the qualification in Paris the autonomous system has been tested closed loop where the autonomous system successfully parked the car 0.4m inside the rectangle from the end barrier. Furthermore, the system has before competition in London completed a 3m step in 15m.

The gates challenge has been provisionally tested, in Paris using normal Voronoi node but only briefly tested before London using the gate classifier approach.

## In-depth analysis of tests

The following is an in-depth analysis of how each element of the design specification has been tested. Please note that the Y-axis of plots is often labeled 'Amplitude': in this context amplitude is meant as 'magnitude'. Moreover, the scaling performed to make different graphs fit in the same plot is described in the legend as well as the unit for each graph. Please also note that when the unit on the legend reads "kmh" it is meant as

kmph, or km per hour.

From the SEM AUC wiki the competition barriers are known. A test track is made using two different types of barriers: plastic boxes similar to the ones being used by SEM, and a smaller, wooden fence type of barrier. As only 80m of plastic barriers was available, meaning 40m of track, the wooden barriers proved a necessary supplement. Both types be seen in figure 10.1.

(a) Plastic barrier

(b) Wooden barrier

Figure 10.1: The different types of barriers used when testing at DTU

## Initial measurements

While the Voronoi algorithm was still underway an initial lidar test was made. The goal of this test was to measure how effectively the lidar mounted on top of the car could detect the barriers. At this point the 3D-printed mount was not yet ready and the lidar was mounted on a tripod instead, as can be seen in figure 10.2.

Figure 10.2: Initial lidar measurements

## Autonomous driving

While still developing the program several tests were made on different test tracks: a straight, an S-shaped track and a jagged track. The straight and S-shaped tracks simu-

late the challenge 1 track layout. The jagged track is a stress test of the navigation and steering elements of the program.

The plastic barriers were easy to move around. This was before the wooden barriers were made; as such we were limited to 40m of stretch. In figure 10.3 the straight stretch can be seen; in figure 10.4 the S-shape can be seen, and in figure 10.5 the jagged stretch can be seen.

Figure 10.3: Testing Voronoi on a straight stretch

Figure 10.4: Testing Voronoi on a S-shaped stretch


Figure 10.5: Testing Voronoi on a jagged stretch

Challenge 1 requires the car to drive autonomously for 970m of track. While the car was found to steer stable and smoothly at high speeds (see figures 10.15 and 10.16) it was still a necessity to ensure that it would do so for at least 1,000m. It is always a good idea to ensure that a system works for as long as is required, especially if there is a chance for the software to cease working.

## Simulating

First the autonomous system was tested in Gazebo (the simulation environment). A digital track was made similar to the final track in London, which can be seen in figure 10.6.However, the Gazebo model contains an error regarding to the car's friction, making it difficult to test speeds in turns. Furthermore, the Gazebo environment is very slow and the time factor is 0.05, meaning that it takes 20s to simulate 1s.

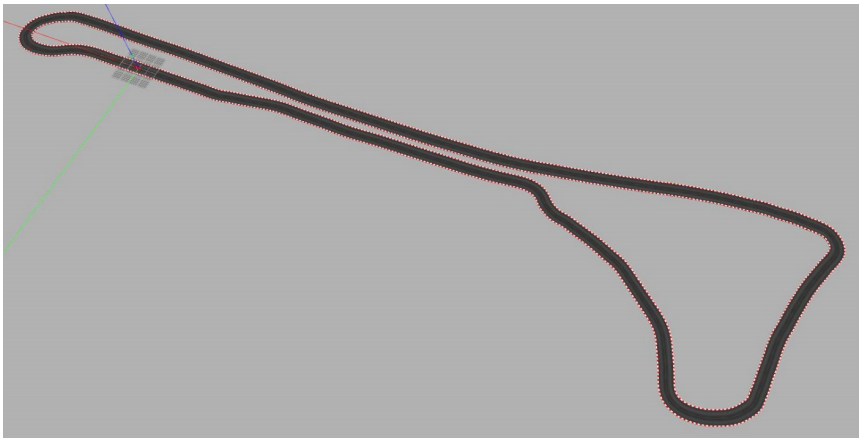


Figure 10.6: Digital representation of the 970m track in final challenge 1

# Distance test

To test this distance in real life a circular track was made prior to departing for London. For this reason 200m of wooden barriers were made, yielding over 100m of track. The track had a small elevation (0.5m) to also test this aspect of the autonomous system since the competition track has 2m height difference. While only 170m long the track served its purpose as the car simply took 6 laps. In figures 10.7, 10.8, 10.9 and 10.10 this track can be seen. In figure 10.13 the layout of the track made from laser measurements can be seen.

Figure 10.7: Circular 170m track

Figure 10.8: Circular 170m track

Figure 10.9: Circular 170m track

Figure 10.10: Circular 170m track

In figure 10.11 the speed control for the distance test can be seen. Since the track contained a paved ditch the slow speed is defined as 6-8kmph. The fast speeds are defined as 10-12 kmph, however as can be seen the engine overshoots to 13kmph. In figure 10.12 the steering output can be seen, which shows the many right turns performed.

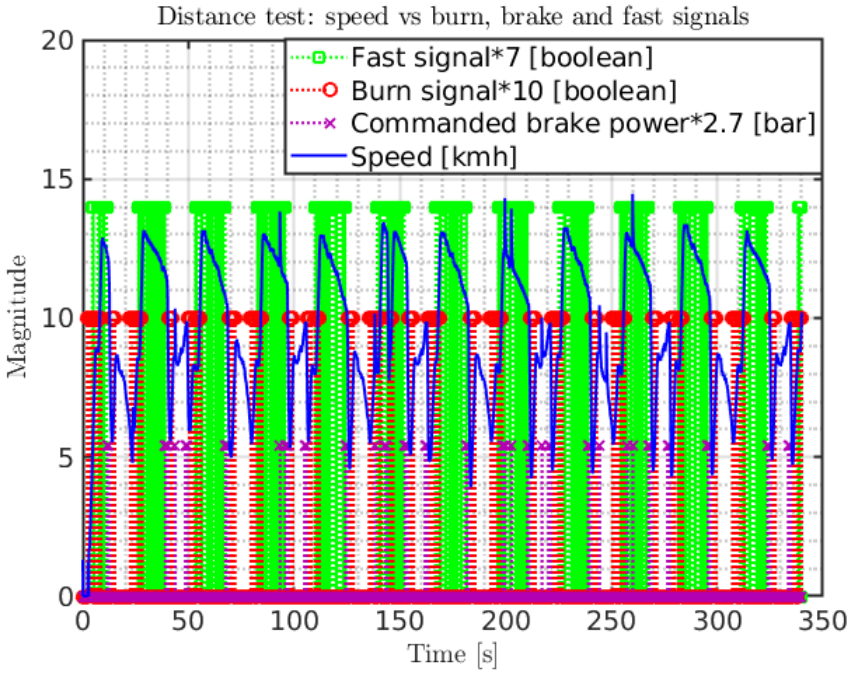



Figure 10.11: Circular 170m track

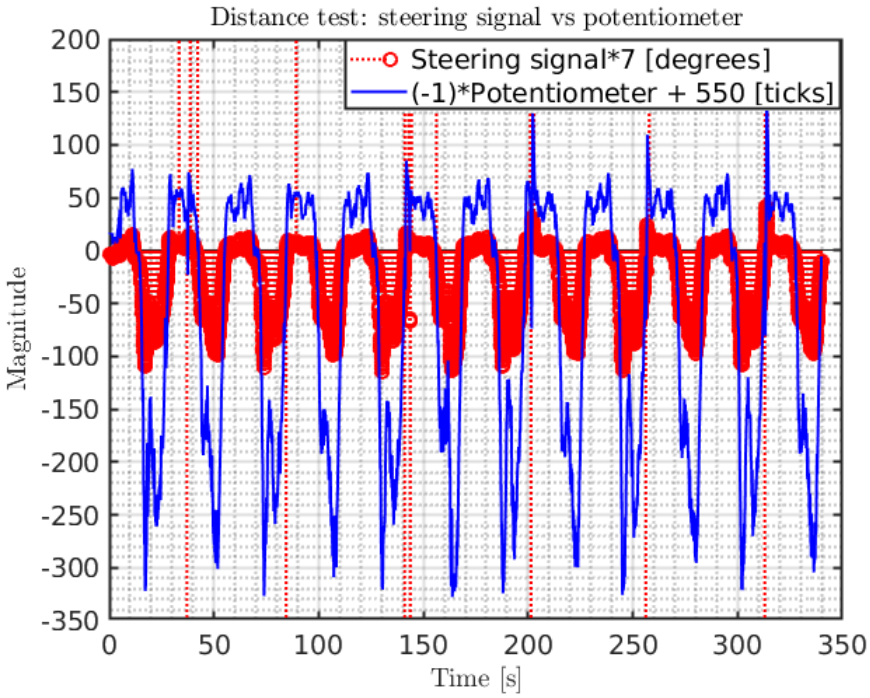



Figure 10.12: Circular 170m track

In figure 10.13 the layout of the track can be seen. Compared to figure 10.14 the odometry drifting can be seen. Since the car does not drive on odometry but on continuous laser measurements it as able to drive as many turns as needed without crashing into the barrier.

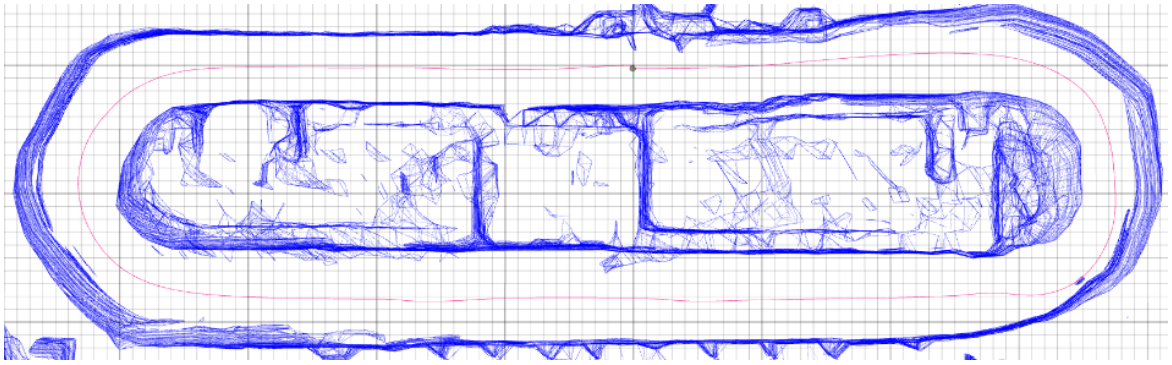


Figure 10.13: Circular 170m track representation made from lidar data. The blue points are obstacles, while the thin red line is the car's odometric path, every square is $1m$.

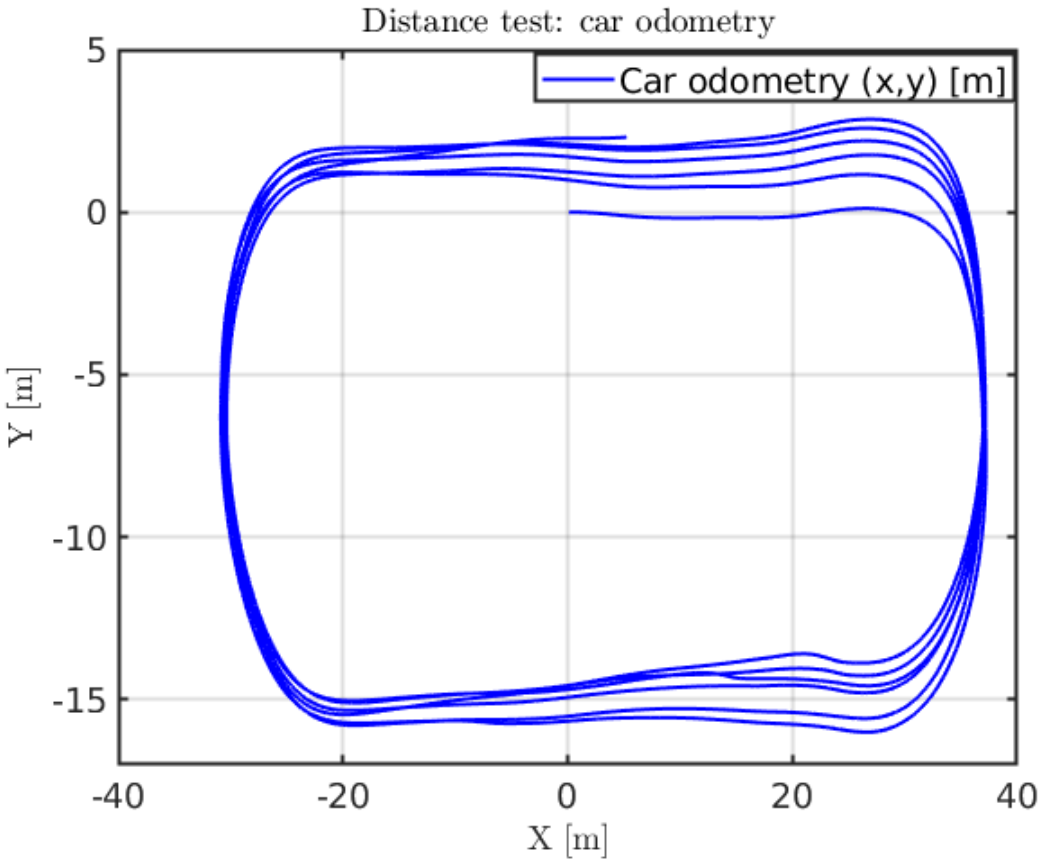



Figure 10.14: Car odometry from the distance test

## Speed test

Aside from distance, speed was also tested thoroughly before London. The critical speed of 14km/h was decided upon at the Paris qualification, since this speed was used to successfully complete challenge 1 at that time. Between Paris and London modifications were made to the *linedrive* node controller values, resulting in a smooth drive even at high speed. It was thus desired to know exactly how fast the car could safely complete a

turn.

A simple test track was made consisting of 100m of straight stretch and a 90 degree turn with a turn radius of 7.5m. For each iteration the car accelerated to a higher speed and kept the speed during the turn. A hard upper limit has been 25km/h since the beginning of the project and so it was tested if the critical speed could be this upper limit.

As can be seen in figure 10.15 the speed is between 25 and 24 kmph during the turn. The turn signal is given at t=22.8s and the smooth and stable steering output can be seen in figure 10.16, showing the stable steering output throughout the turn. From this it can be seen that it is possible for the car to clear a 90-degree turn with turn radius 7.5m at 25kmph, however the high speed pushes the car right from the middle of the road and closer to the right hand barrier. This can be a problem after the turn as the reason for driving in the middle of the road is to have as much steering margin as possible.

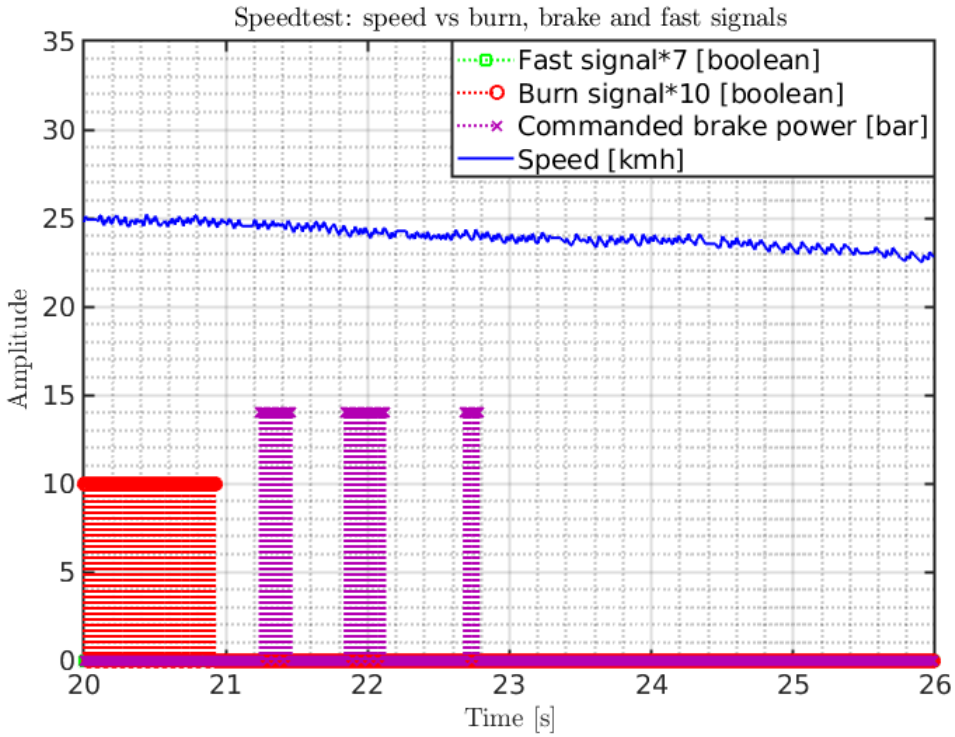



Figure 10.15: Speed test: speed control

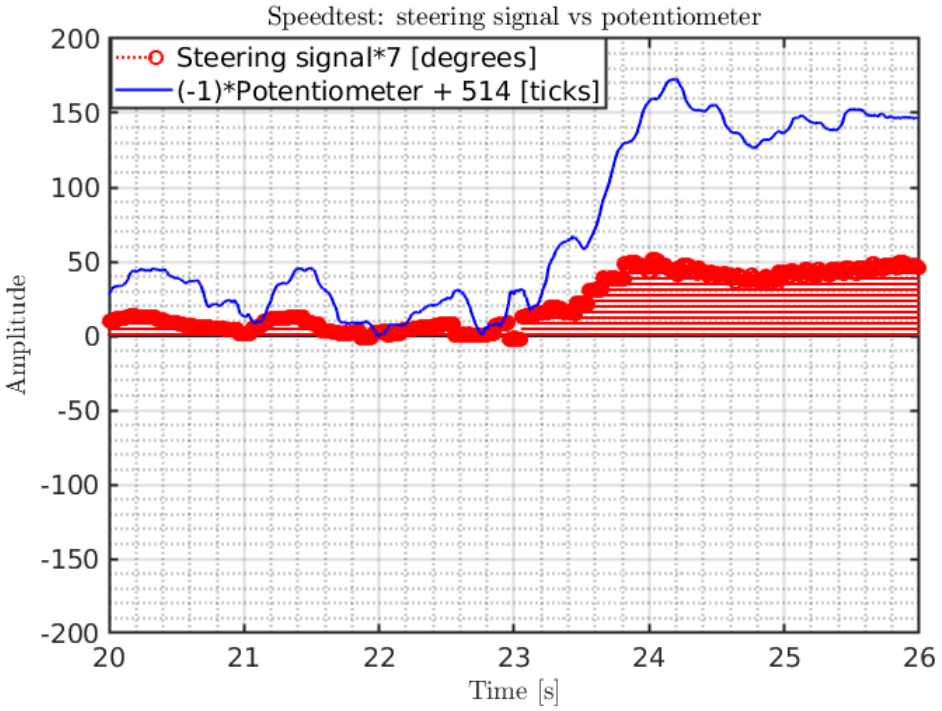



Figure 10.16: Speed test: steering

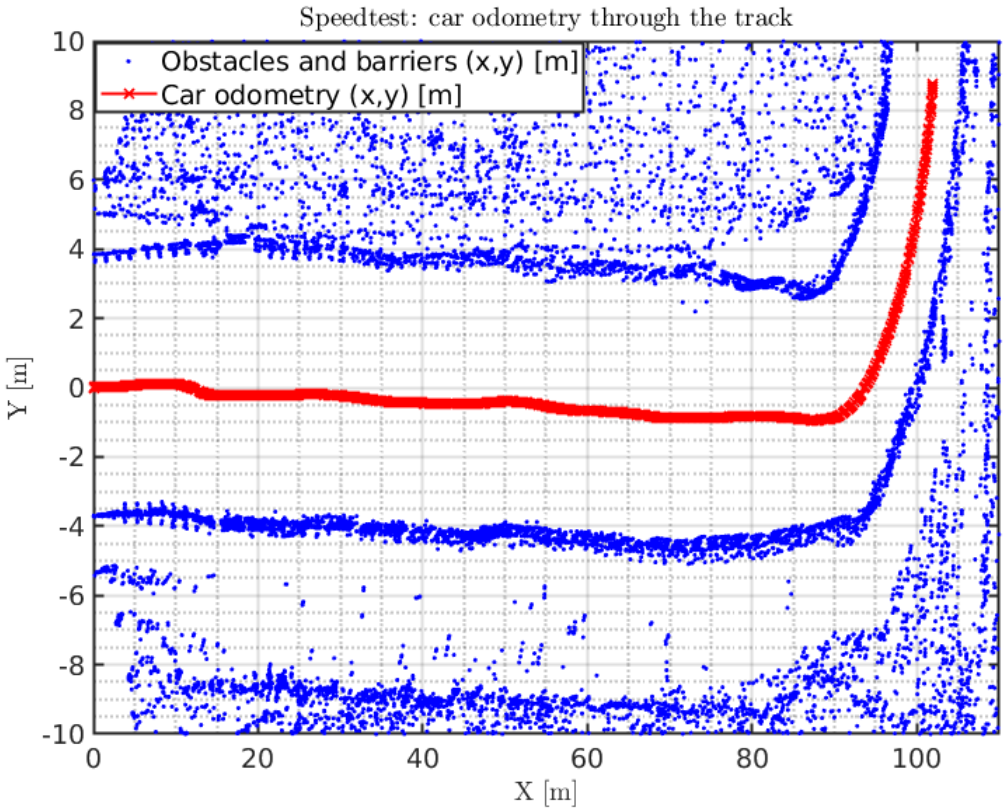



Figure 10.17: Speed test layout, as seen by the laser measurements and the car's odometry

## Challenge 2: the complex track

The complex track layout was not considered a difficult challenge due to the large size of the obstacle boxes compared to how the car drove in the jagged test track. Therefore, this challenge was only afforded a digital test track, as can be seen in figure 10.18.

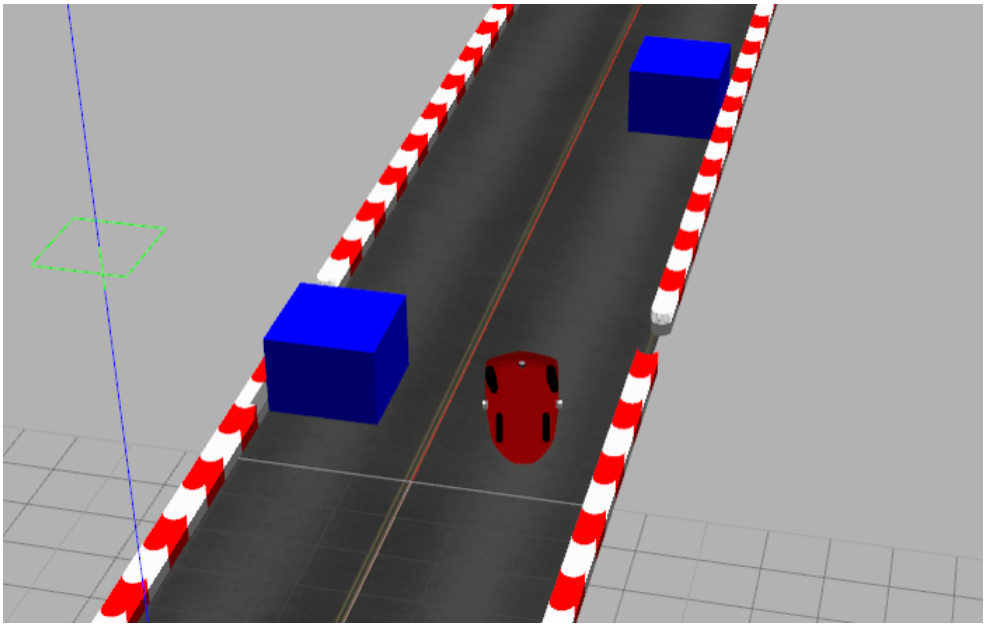


Figure 10.18: Digital representation of challenge 2

## Challenge 3: parking

For the parking challenge a parking spot with end barrier has been used as test track. At first white tape was used until black and yellow striped tape was purchased. The end barrier used was the red and white plastic barriers, as can be seen in figure 10.19.

Figure 10.19: Testing the parking code

The parking program relies on the camera detection program to detect the parking spot and to extract the spot's coordinates. The parking node then publishes this coordinate to the linedrive node which steers the car into the spot's angle. It is difficult to make the car stop at an exact distance from the end barrier as the system as a whole has some delays, and due to the speed of the brake actuator the speed of the car influences the braking distance greatly. The commanded braking pressure is currently a static value and not dependant on the speed of the car. However, after attempting parking several times it is possible to find values which effectively stop the car in front of the end barrier.

## Parking in Paris

In Paris the parking program was fine tuned to give yield the shortest possible distance to the end barrier. This was done by changing the maxspeed and minspeed, along with braking distance. In figure 10.20 the layout of the track can be seen, as seen from the autonomous system.

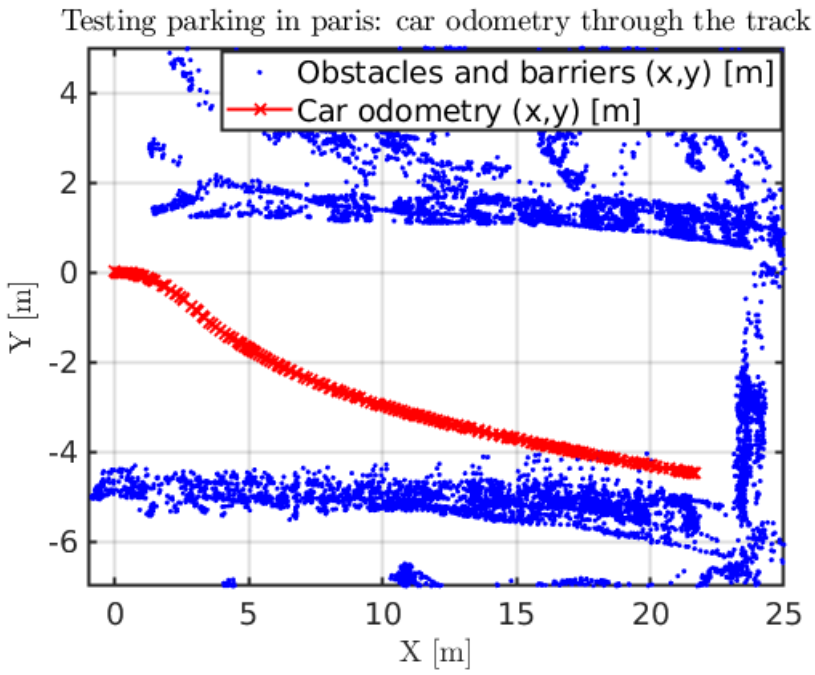



Figure 10.20: Challenge 4 in Paris: layout of the track as seen by the car's odometry and the LiDAR measureents

In figure 10.21 the speed and braking control can be seen.

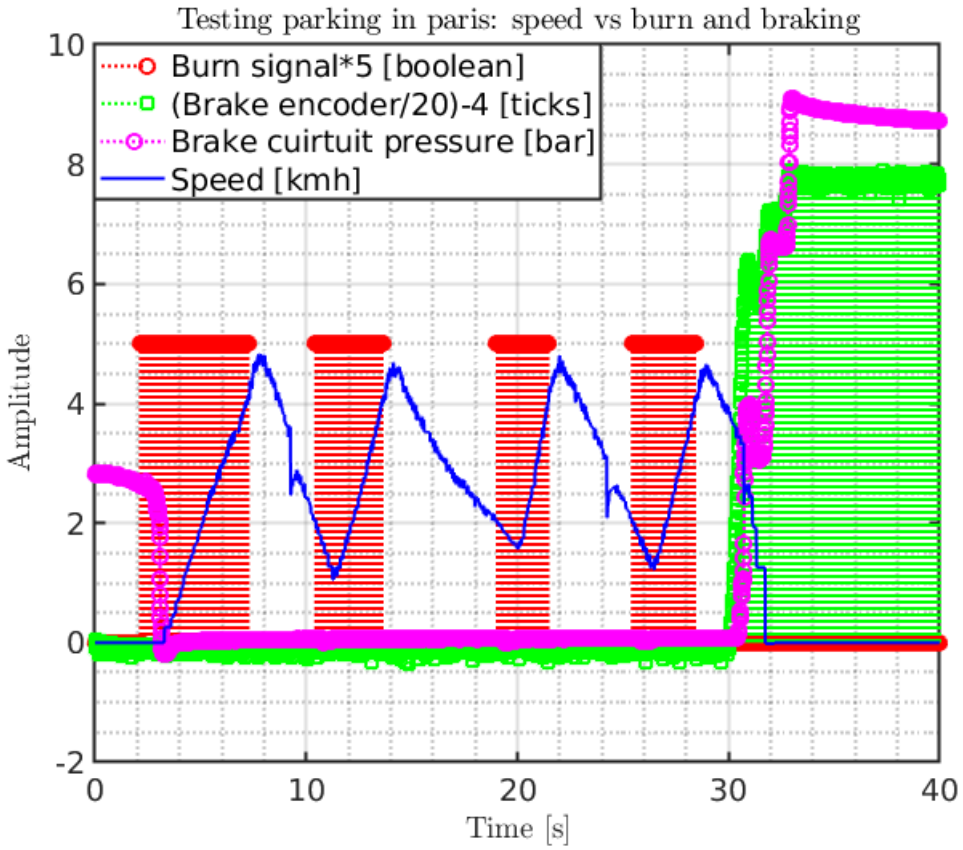



Figure 10.21: Challenge 4 in Paris: steering output

In figure 10.22 the steering output can be seen.

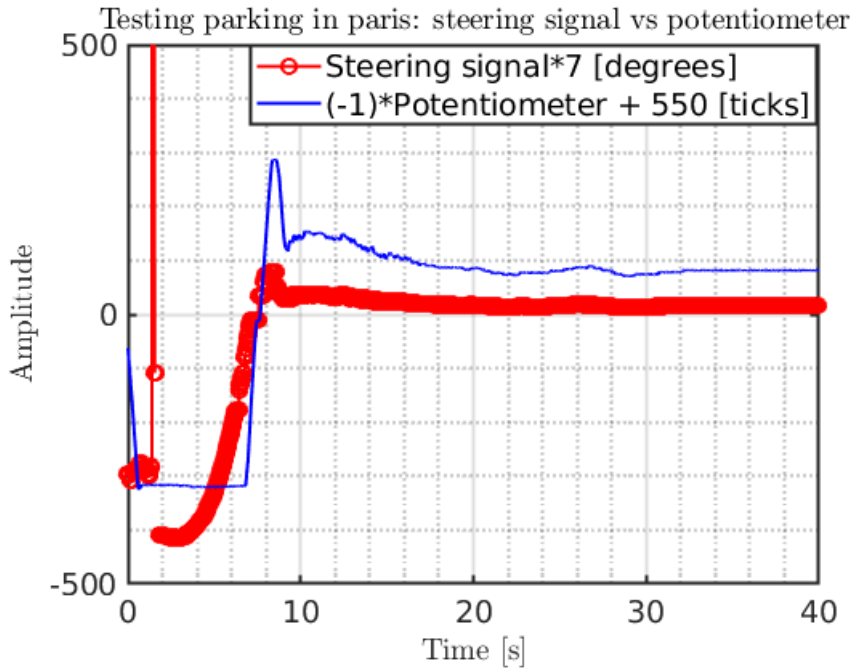



Figure 10.22: Challenge 4 in Paris: steering output

# Challenge 4: gates

In Paris the standard Voronoi algorithm was used to simply treat the gates as standard obstacles, transforming the track to three Y-splits. This approach made use of the multiple path option described in the Navigational Algorithm chapter, and the program was prepared offline to take the correct left and right turns when facing these Y-splits. In figure 10.23 the layout of the track and the splits can be seen.

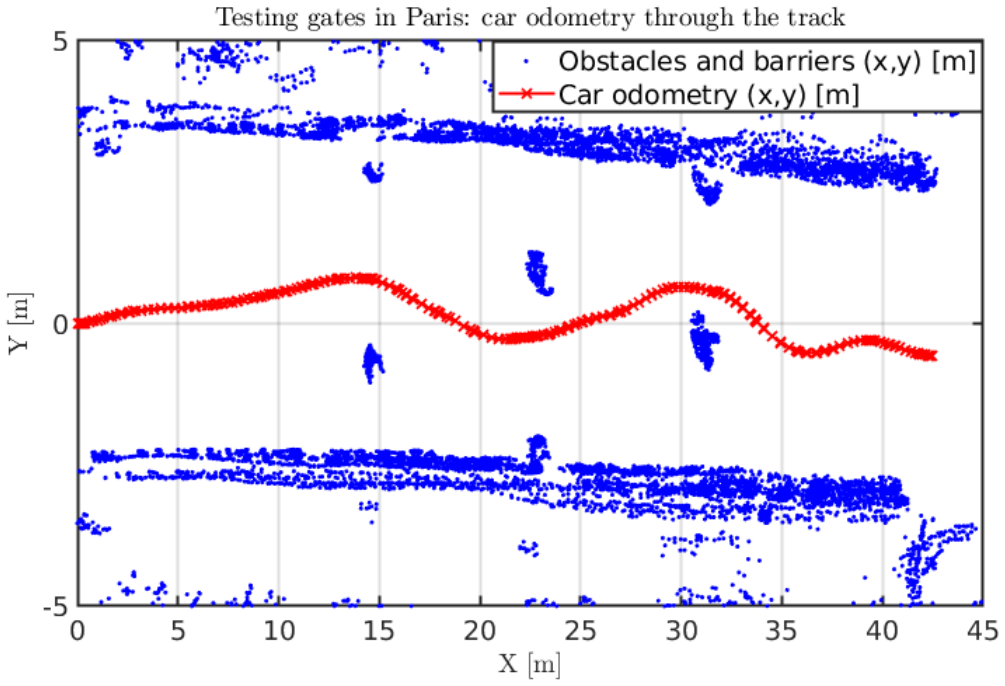



Figure 10.23: Challenge 4 in Paris: layout of the track as seen by the car's odometry and the LiDAR measurements

In figure 10.24 the steering output can be seen.

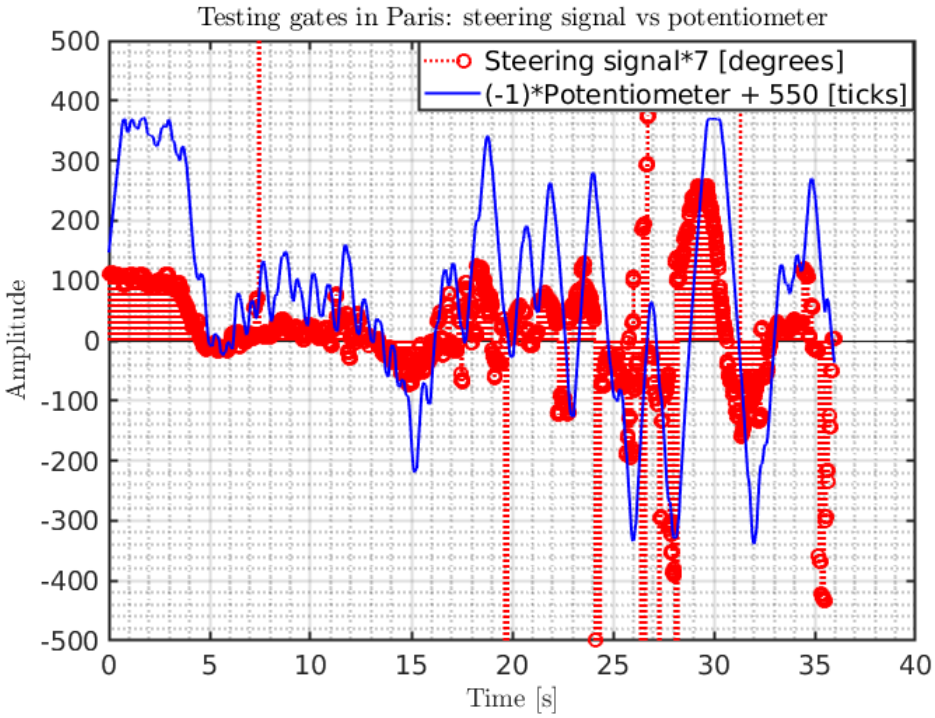
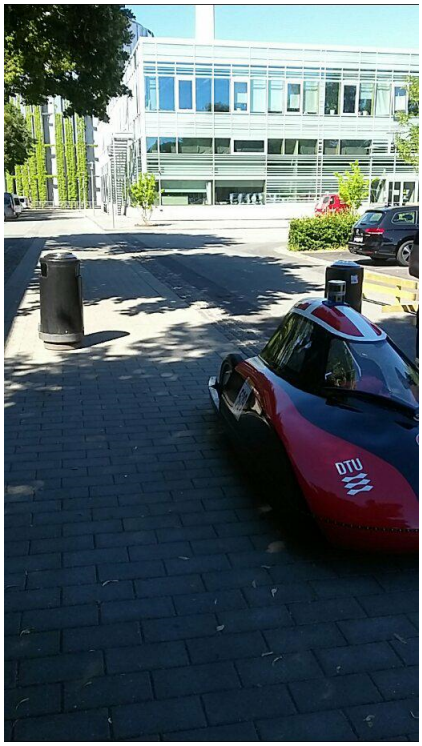



Figure 10.24: Challenge 4 in Paris: steering output

## For the final challenge in London

In the competition in London a new node called the *gate classifier* was used, which locates the center between the gates, instead of using voronoi to solve the gate problem. This gate classifier was only tested once before departing to London. Garbage bins were used in the place of gates, which is far from ideal as they were not completely identical nor cylindrical, as can be seen in figure 10.25.




(a) Using garbage bins as gates

(b) Track layout in London

Figure 10.25: Testing the gates challenge

## Brake tests

The brakes are tested with a step response: the car is accelerated to the desired top speed after which the brakes are activated with a constant pressure level until the desired minimum speed is reached. The minimum speed could be 0 (standstill) or a lower speed larger than 0, e.g. decelerating from 20 to 10 km/h. It is necessary to know how fast the brake achieves the desired pressure, if it keeps the pressure stable and how heavily this pressure decelerates the car. This test does not require barriers, only a stretch of track with the desired length.

The procedure for the brake tests is defined as follows:

1. Accelerate car to desired top speed
2. Activate brakes with desired pressure level, decelerating the car. This is $t_{start}$, starting time for the braking, and $x_{start}$, the starting x-position
3. When desired minimum speed is reached, release the brakes. This is $t_{end}$ and $x_{end}$, the end x-position.
4. Braking time is now found by $t_{end} - t_{start}$ and braking distance - or braking length - is found by $x_{end} - x_{start}$

Example: the **fast** top speed for used is 20km/h, and the **slow** top speed, which is also the critical speed, is 14km/h. Thus, it is desired to test braking from 20 to 14km/h.

The car is then accelerated to 20 after which the brakes are activated, decelerating to 14 and releasing the brakes.

The deceleration rate depends on the brake pressure level, and braking distance depends on brake pressure level as well as initial speed (the top speed right before activating the brakes).

The result from these brake tests was used to determine a deceleration brake power and a full stop brake power before going to London. This process was necessarily repeated every time the relation between brake pressure and deceleration speed was changed. As such no definite choice of brake power can be stated.

# 11 Results

In the following section the final results from the competition in London are described.

## Conclusion of the results

The autonomous system has completed the 1000m requirement and almost the 18km/h requirement in London. The 3m step within 10m is also completed in London. The parking requirement to stop within 2m of the end barrier is completed in Paris, as described in the Testing chapter.

### The competition in London

Throughout all challenges the steering worked well. During the first challenge it became evident that the fast-slow algorithm to set speed reference was not suited for the engine on the car without additional work to this algorithm, since the control was too sporadic.

During parking the brake was activated too late to effectively stop the car at the designated spot.

During the gates challenge the steering proved effective. In this challenge the biggest weakness of the system was the controller values for the linedrive node, provided that the gate classifier node is working as intended.

In conclusion, the results are considered a success as they demonstrate that it was possible to make an autonomous system that fulfills the desired targets with regards to all thesis requirements.

## In-depth analysis of results

The following is an in-depth analysis of the results from the final competition in London. Please note that the Y-axis of plots is often labeled 'Amplitude': in this context amplitude is meant as 'magnitude'. Moreover, the scaling performed to make different graphs fit in the same plot is described in the legend as well as the unit for each graph. Please also note that when the unit on the legend reads "kmh" it is meant as kmph, or km per hour.

# Challenge 1: Autonomous lap

On July 7th the autonomous lap was completed on first try, traversing the 970m in 3 minutes 29 seconds. The car started close to the left hand barrier, after which the autonomous system suffered an error which made the car drive toward the left barrier only to stabilize and drive to the middle of the track. Apart from this initial trouble the engine suffered a mechanical error at 181s. Following this the engine was unable to burn, leading to the car coasting the last 28s. The finish line was passed with only 7km/h. Aside from these two problems the run was considered a success.

In the following three aspects of the autonomous system will be shown: the speed control, the steering and the braking.

## Speed control

In figure 11.1 the speed control throughout the lap can be seen. The amplitudes of burn and brake signals are amplified for visual purposes. Braking is only done when either switching from fast to slow speeds, or if the car is coasting downhill and accelerating above the maxspeed, whether fast or slow. It should be noted that throughout the lap the speed measurements are seen to occasionally spike. These spikes take 200ms to settle. Furthermore, at t=12s a switch from 1st to 2nd gear can be seen. In 2nd gear the car accelerates significantly slower.

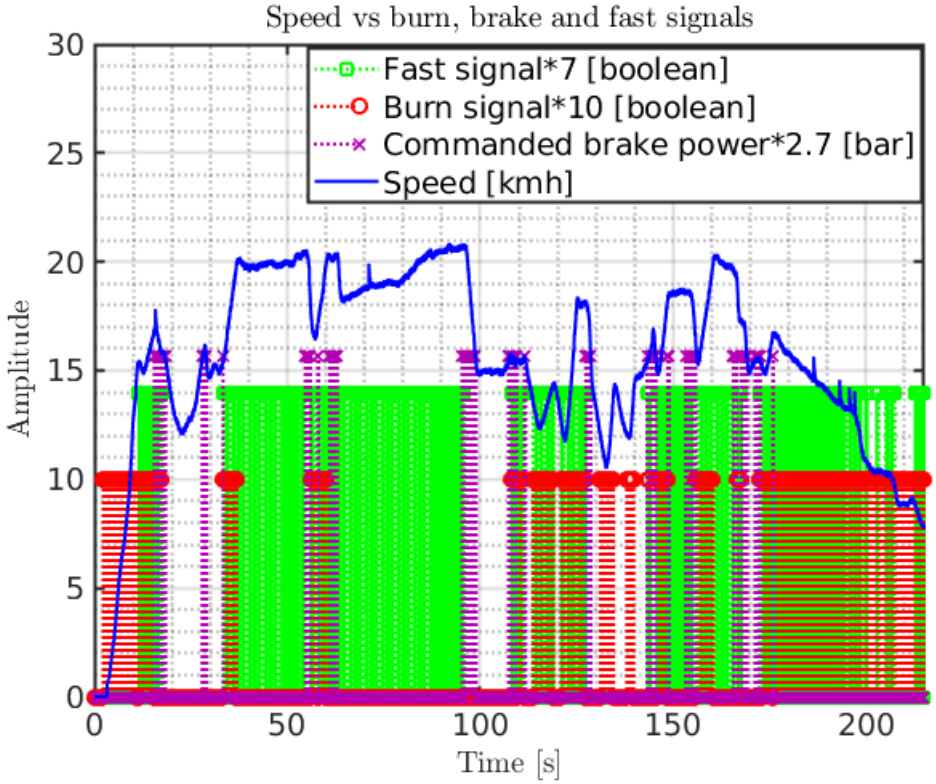



Figure 11.1: Speed control (Challenge 1)

In figure 11.2 the commanded steering angle along with the potentiometer value can be seen. The commanded signals are amplified and the potentiometer values offset for visual purposes. As can be seen, the autonomous system spends the first 10s to steer the car from the left into the middle of the track. After this the steering output is generally steady. The different turns are easily distinguished from driving straight.

An interesting phenomenon is the few spikes in extreme magnitude, found at t= 34.6s, t=76.01s and t=107.8s. This is caused by the autonomous system temporarily detecting a gap between the left hand barriers. The first instance lasts 90ms, the second 150ms and the third 100ms.

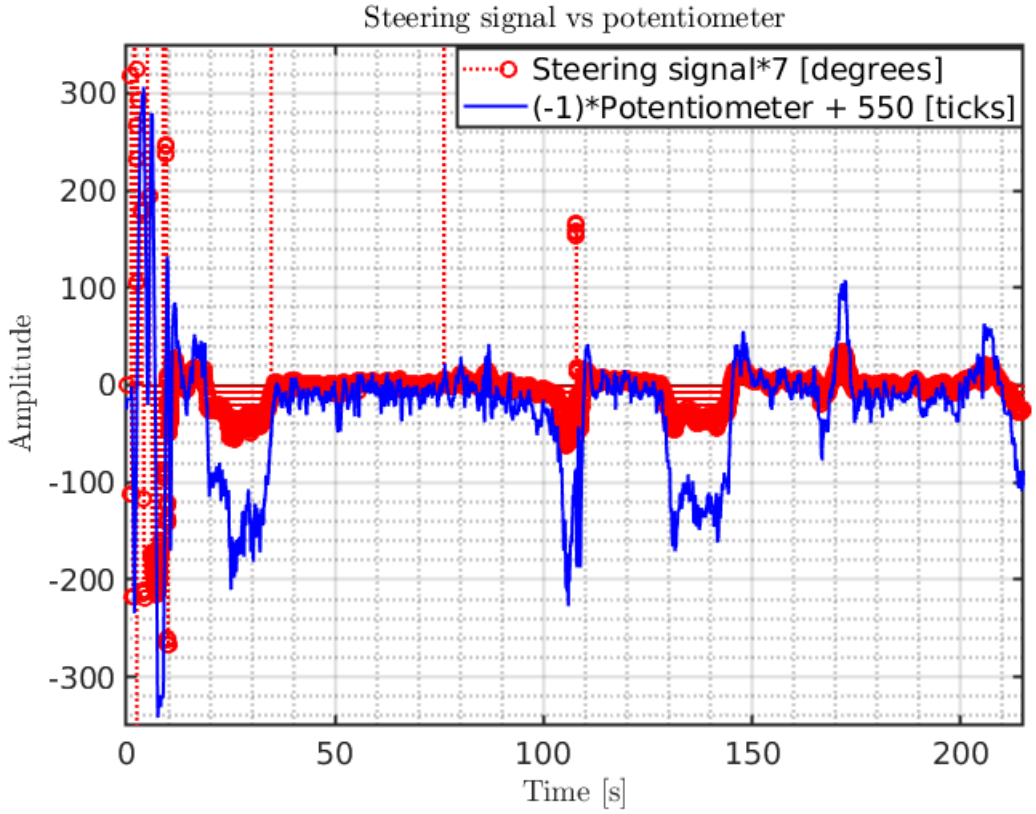



Figure 11.2: Steering control (Challenge 1)

The braking part of the speed control can be described as a step response in which the brake node is commanded to brake with a certain amplitude. As seen in figure 5.6 the brake overshoots. In figure 11.3 the brake signals of the whole run can be seen compared to the measured pressure in the brake system. Generally speaking the overshoot varies, as does rise time.

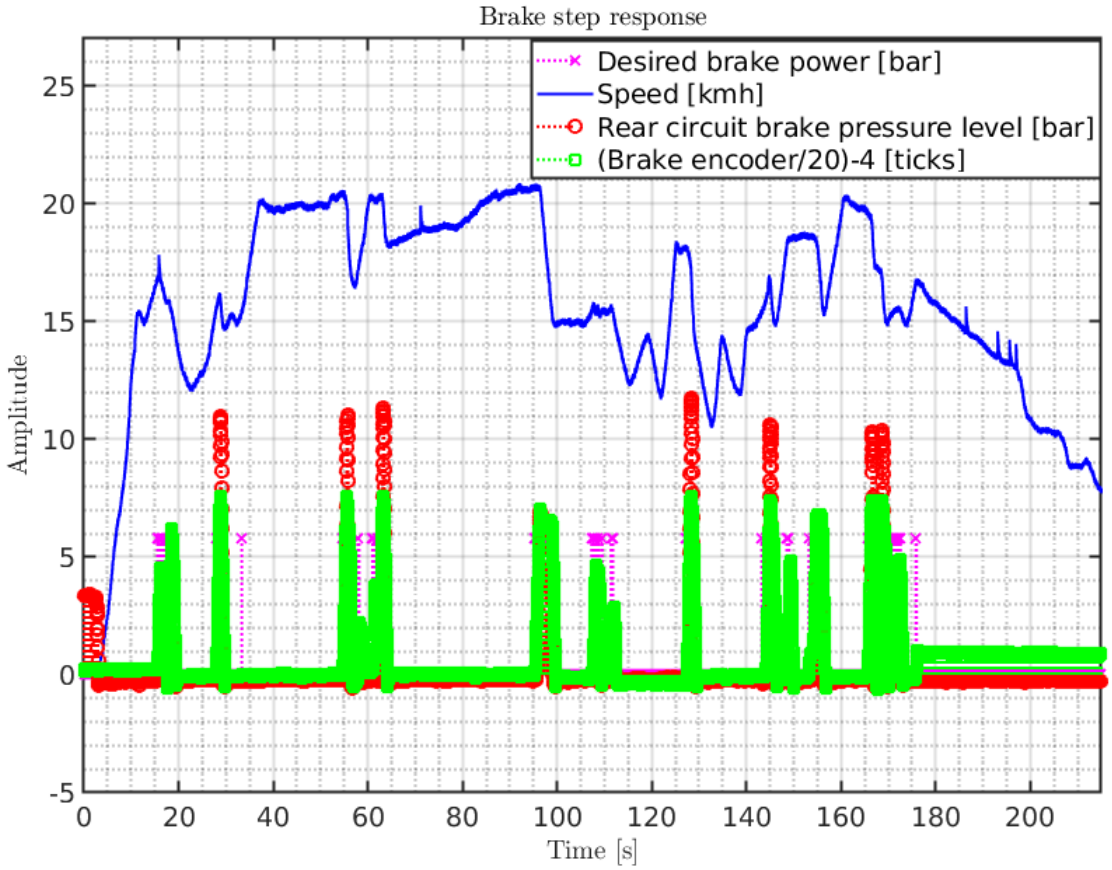



Figure 11.3: Brake control (Challenge 1)

In the following different sections of the lap will be examined closer.

## Start and the first turn

In figure 11.4 a switch between slow and fast can be seen. The autonomous system burns to above slow maxspeed and stops burning, switching to fast at t=11.5s and continues burning. In figure 11.5 the first turn can be seen. The burn to fast maxspeed from figure 11.4 ends right before t=16s followed by a series of switches between burning and braking, taking place right before the turn. During the turn the car coasts in slow speed until about t=22s after which the track goes slightly downhill, resulting in the acceleration of the car which accelerates above slow maxspeed, requiring the speed control to brake at t=28s. At t=33.5s the turn is cleared and a straight stretch detected, leaving the autonomous system to accelerate up to fast speeds again.

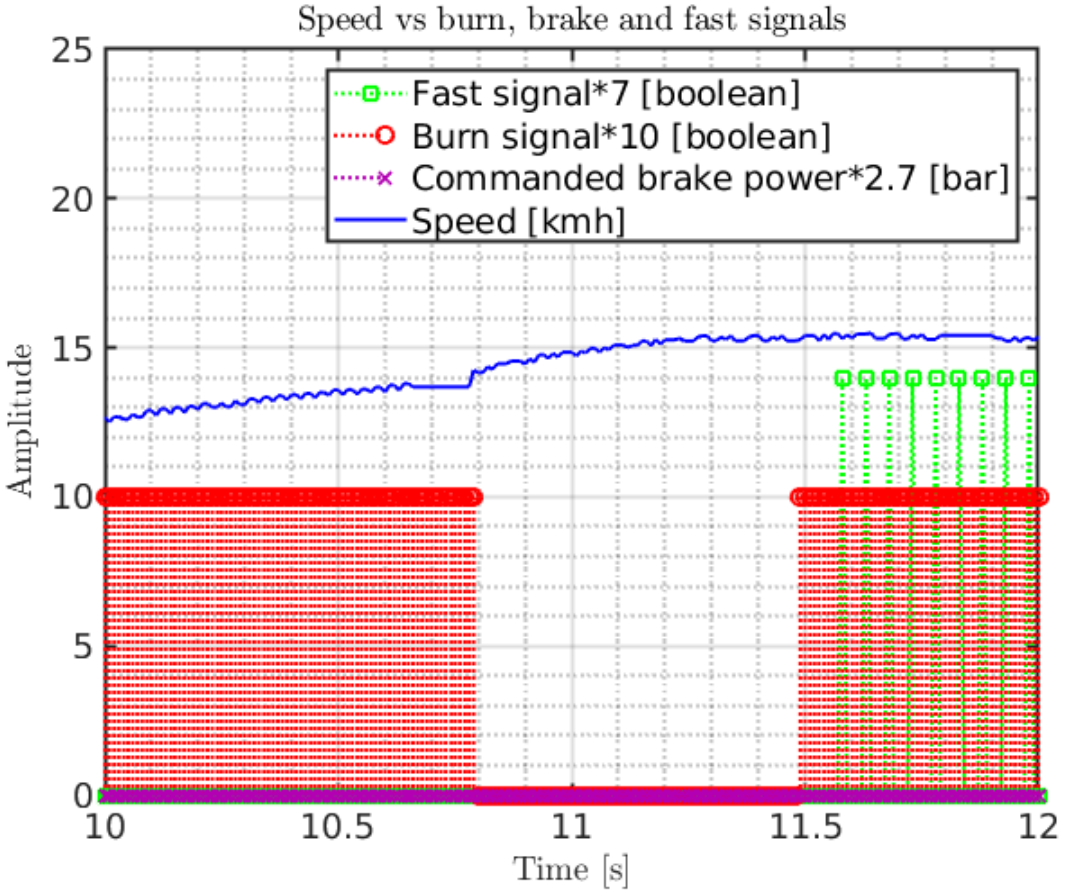



Figure 11.4: Speed control at t=10 to t=12s (Challenge 1)

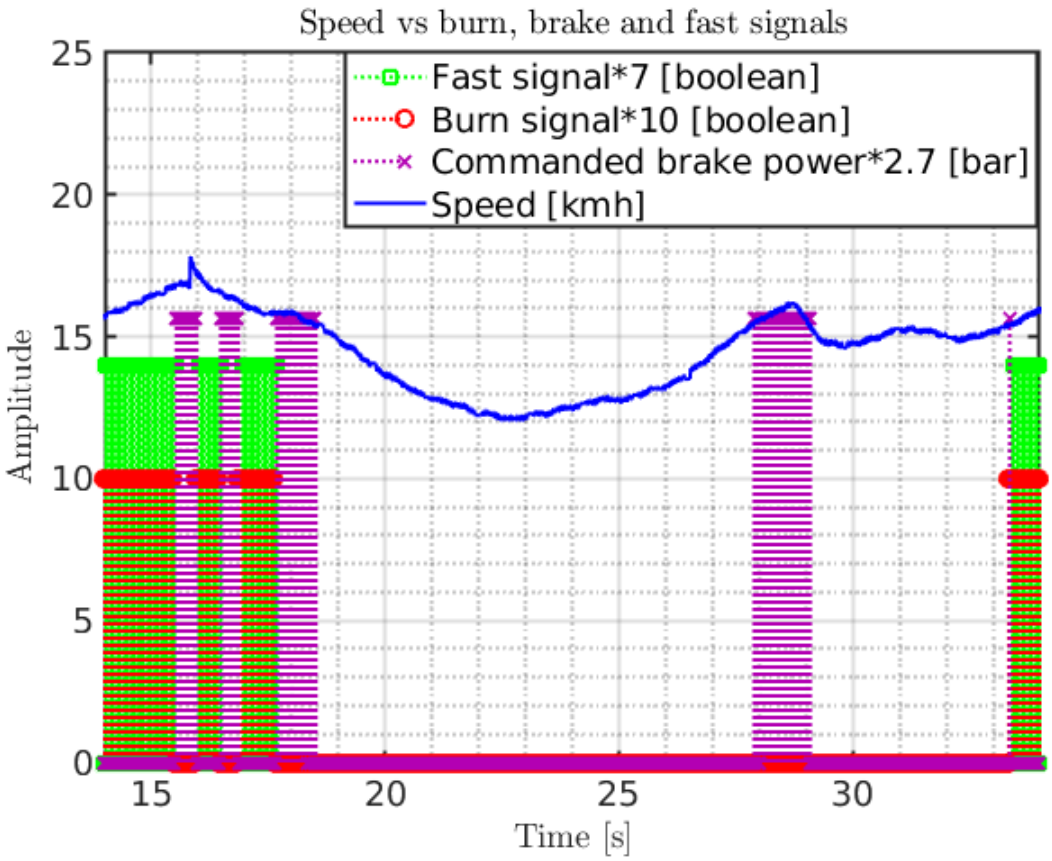



Figure 11.5: Speed control at t=14 to t=34s (Challenge 1)

In figure 11.6 the initial steering close to the left wall and then into the middle of the track can be seen. This situation arose due to two things: the car started offset to the left of the middle of the track, and subsequently had problems detecting the right side of the track due to this offset as well as the extra wide track at the starting line. The cause of this problem was later found to be a small variable in the LiDAR node. However, once the autonomous system managed to detect the right hand barriers it steered into the middle of the road.

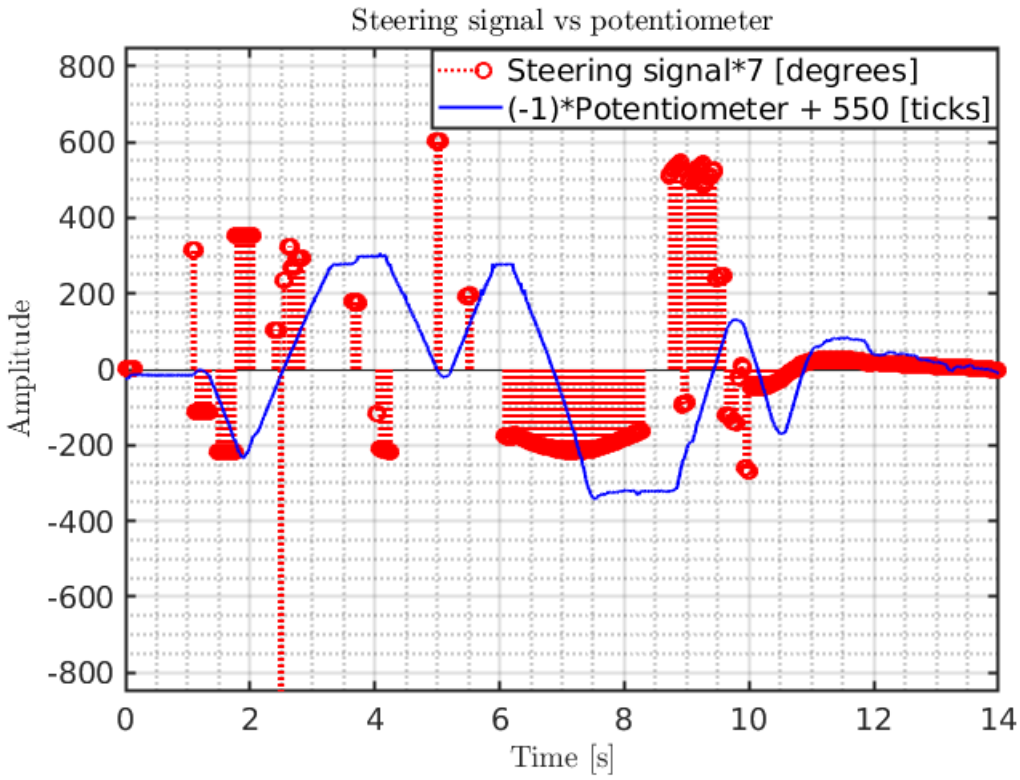



Figure 11.6: Steering control at t=0s to t=14s (Challenge 1)

Steering for the first turn can be seen in figure 11.7. As can be seen the steering is consistent except for the last spike at t=34.6s which was caused by the autonomous system briefly detecting a gap in the left hand barriers.

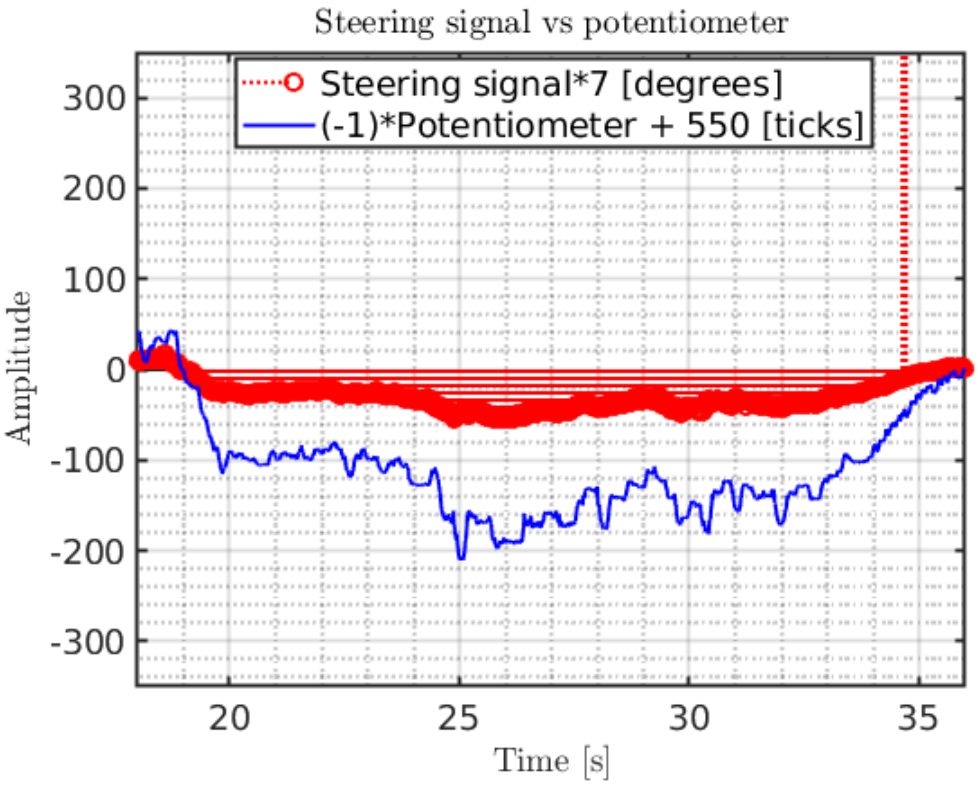



Figure 11.7: Steering control at t=18s to t=36s (Challenge 1)

## Until the second turn

Figure 11.8 shows the speed control when the car passes a small bump in the track asphalt. While travelling at high speed a series of short brakes and burns is initiated shortly within each other, illustrating the inconsistency of the slow-fast algorithm.

In figure 11.1 an interesting situation can be seen at t=65s to t=95s, immediately before the second turn. While driving in fast mode at just above minspeed (18kmph) the burn doesn't start. At this point the autonomous system expects the car to slowly coast down below 18kmph, however due to the slight decline of the track just prior to the second turn the car slowly accelerates without burning the engine. The autonomous system then detects the turn coming up and switches to slow speed, resulting in activating the brake.

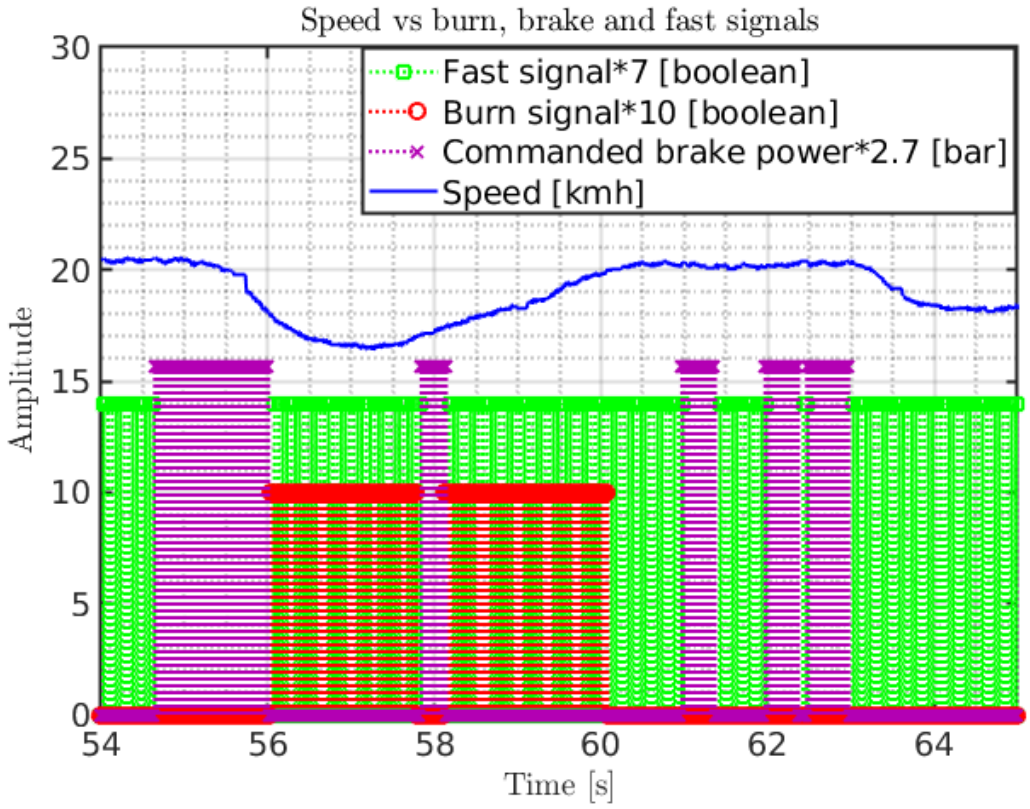



Figure 11.8: Speed control at t=54s to t=64s (Challenge 1)

In figure 11.9 the second turn can be seen. While travelling at fast maxspeed the autonomous system detects a turn and brakes down to slow speed. The inconsistent nature of the fast-slow program causes the autonomous system to cancel braking at t=96s, however following this the turn is again detected. The first brake command is given at t=95.28s and the second at t=95.91s. The car starts to decelerate at t=96.38s and stops braking at t=99.07s. This means that from the brake was activated it took the car 2.69s to decelerate 5.76kmph, from 20.73kmph to 14.97kmph. This yields a deceleration rate of 2.14kmph/s, that is 2.14 kilometers-per-hour per second. Towards the end of the turn another series of brakes and burns follow. The final burn at t=114s is initiated when below slow minspeed.

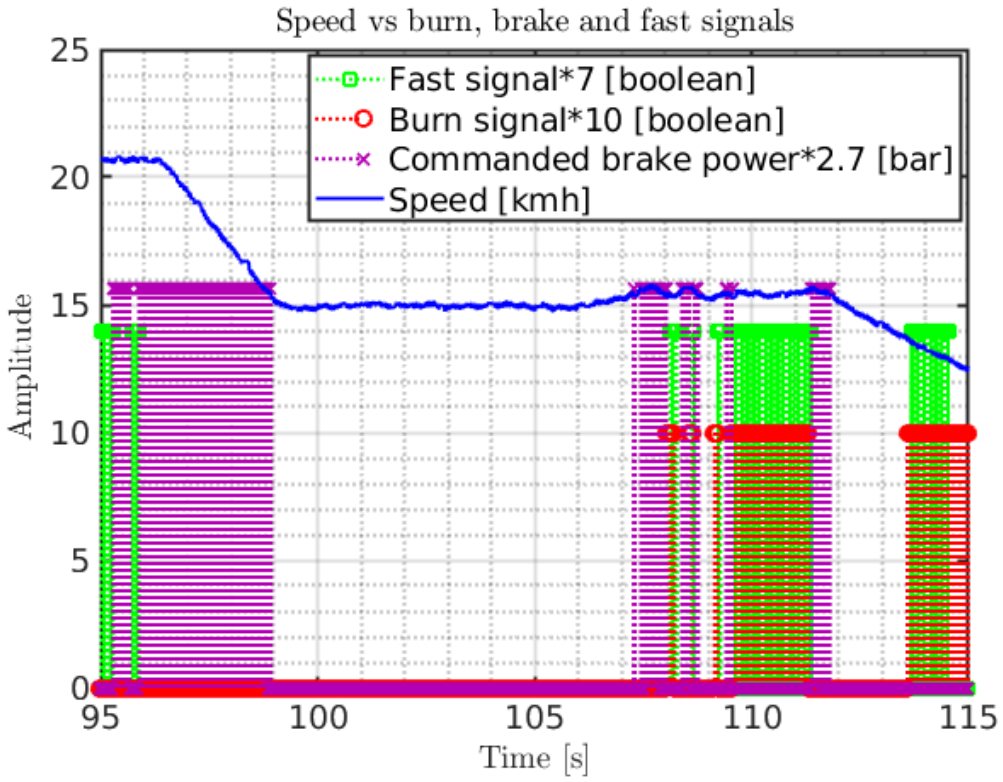



Figure 11.9: Speed control at t=95s to t=115s (Challenge 1)

Figure 11.8 illustrates how the autonomous system handles a small bump in the road. While the speed control detects a false positive and brakes the steering has no problem.

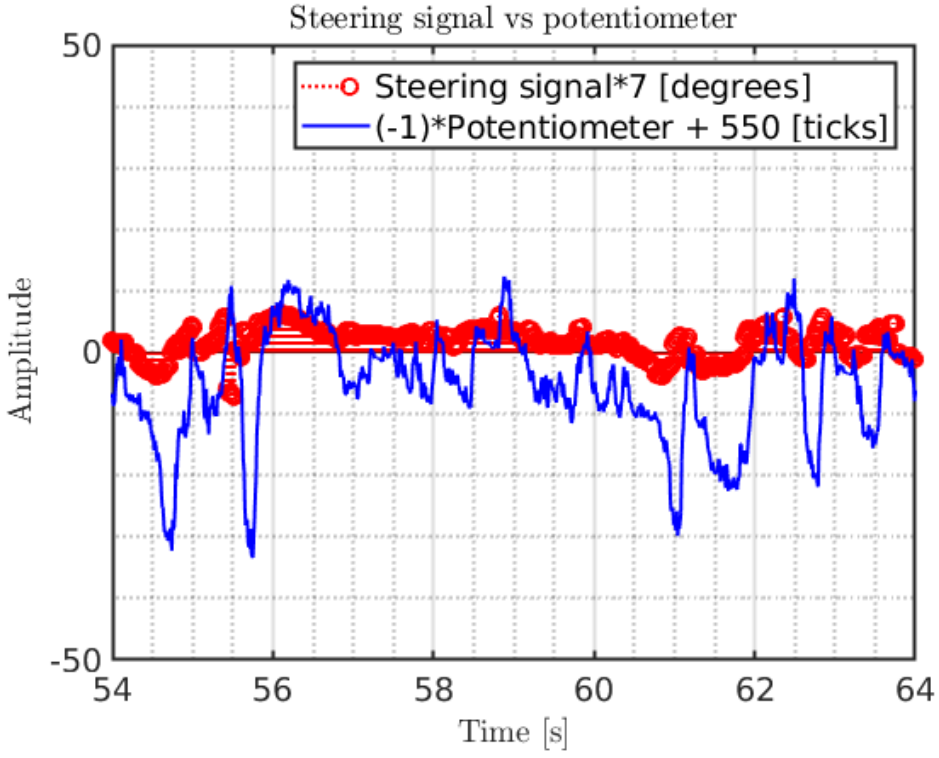



Figure 11.10: Steering control at t=54s to t=64s (Challenge 1)

In figure 11.11 the narrowing of the track prior to the second turn can be seen. The car steers smoothly and continues in the middle of the track. In figure 11.12 the second turn can be seen. Once again, the autonomous system steers smoothly to the right. The brief spike at t=107.9s is once again caused by a detected gap in the left hand barriers. The turn radius of the second turn is found to be 30m which is easily completed by the autonomous system.

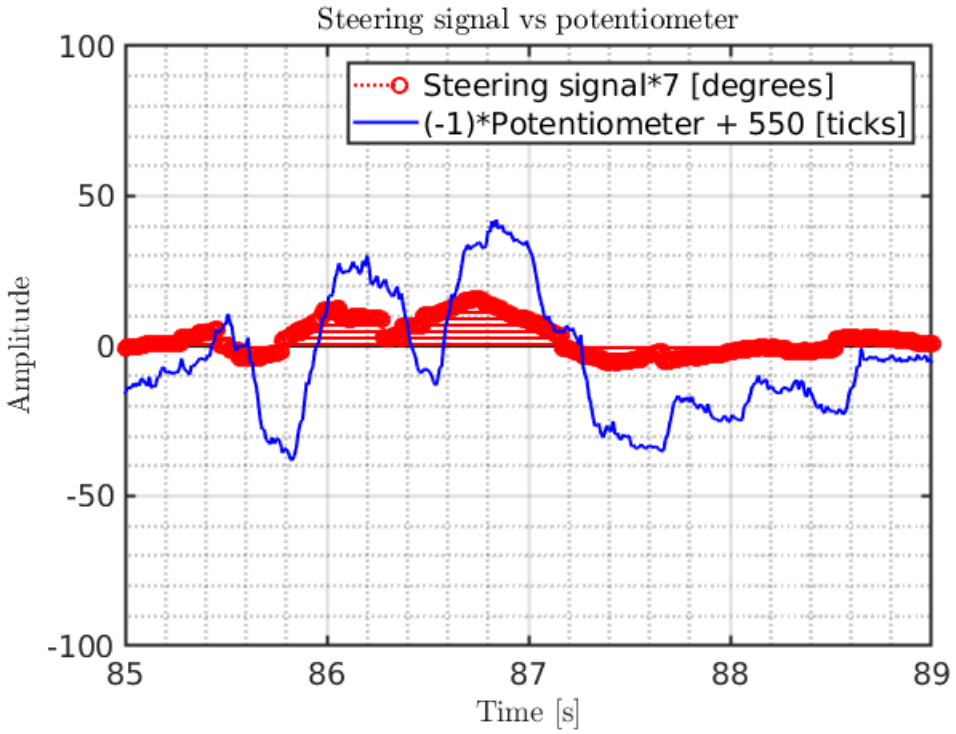



Figure 11.11: Steering control at t=85 to t=89s (Challenge 1)

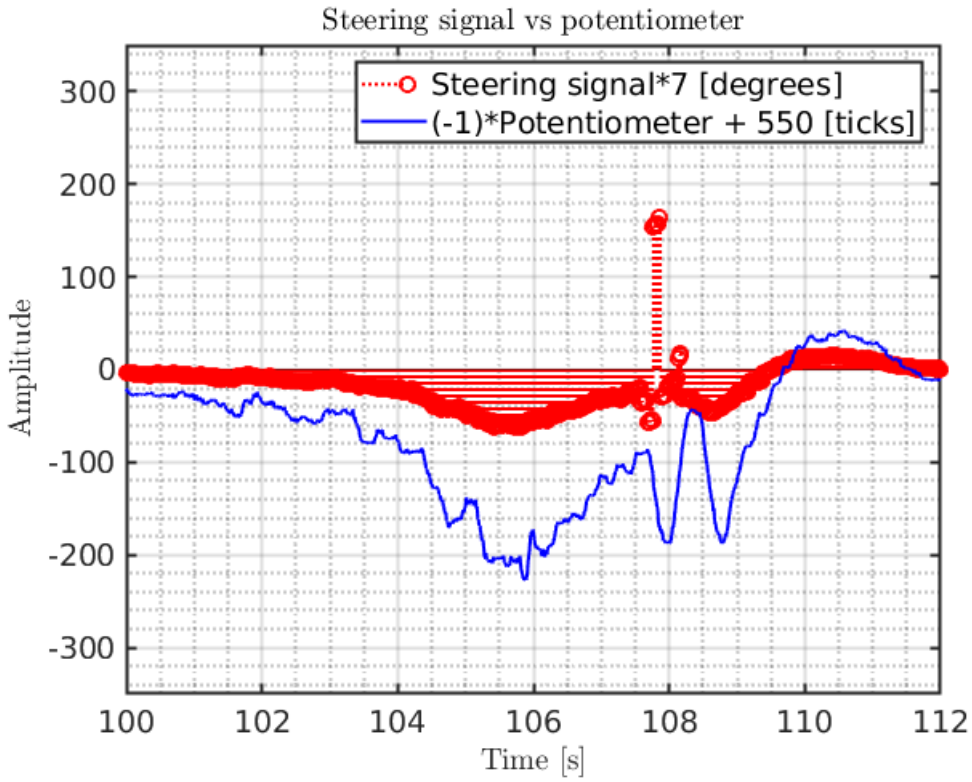



Figure 11.12: Steering control at t=100s to t=112s (Challenge 1)

In figure 11.13 the braking at the bump can be seen. The autonomous system detects a false positive and brakes at approximately t=54.6s. The braking really starts until approximately 0.5s after the command is given, however the brake is overshooting significantly in magnitude.

Figure 11.13 can be compared with 11.8 for information about burning.

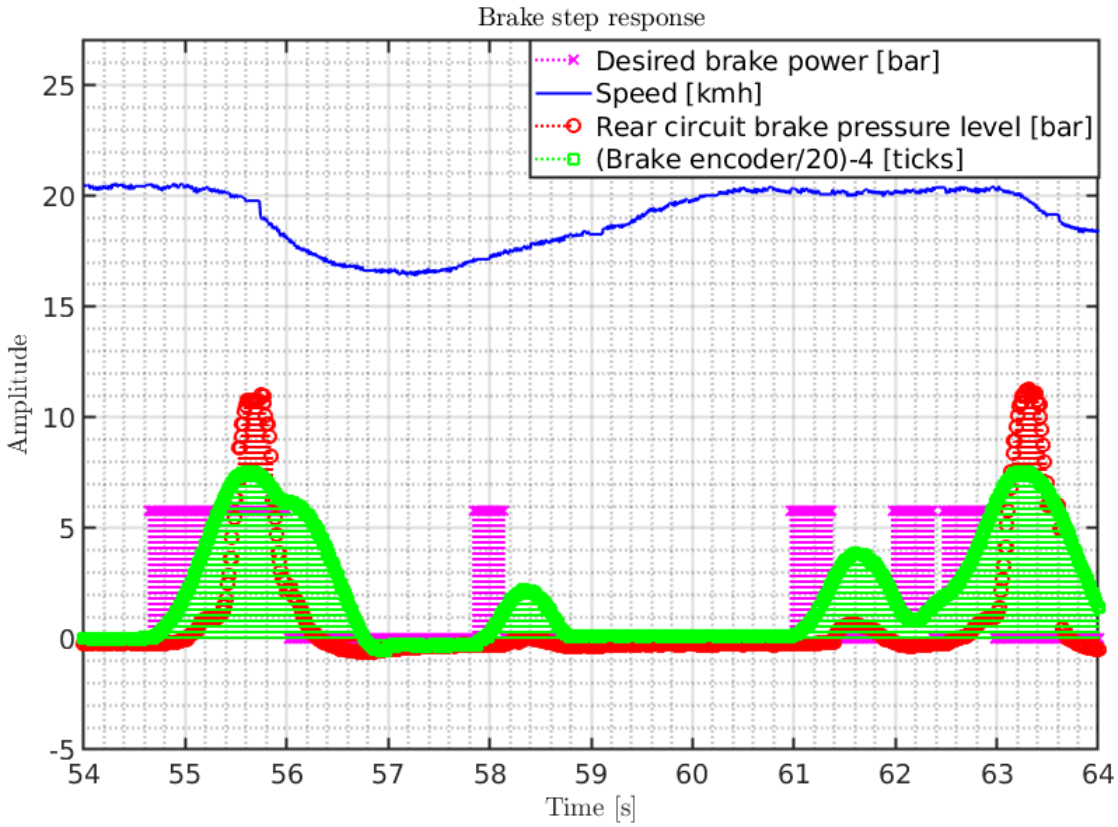



Figure 11.13: Challenge 1: brake control at t=t54s to 64s

In figure 11.14 the braking before the second turn can be seen. Due to the sporadic nature of the fast-slow algorithm the autonomous system has as a brief moment of doubt after which it keeps braking until the turn is cleared.

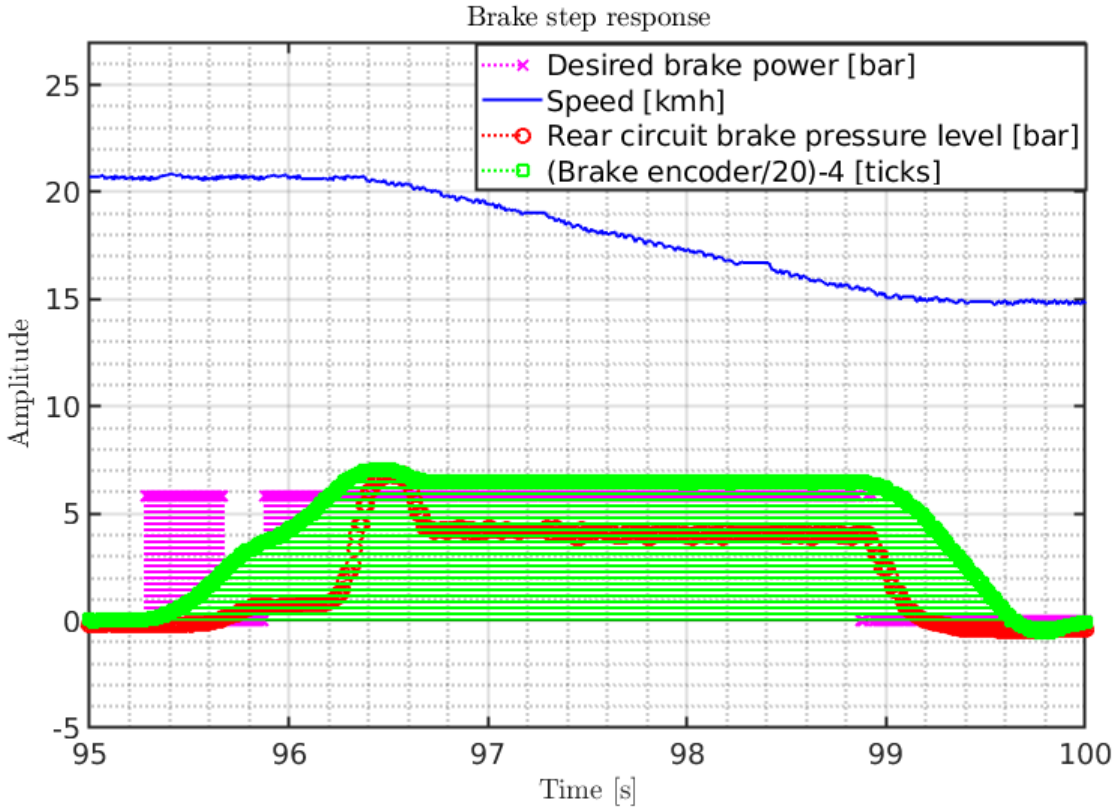



Figure 11.14: Challenge 1: brake control at t=t95s to 100s

## Uphill and the third turn

In figure 11.15 the third turn can be seen. The autonomous system detects the turn at t=127.5s and decelerates to slow speed. The brake command is sent at t=127.3s and the car starts decelerating at t=128.1s.

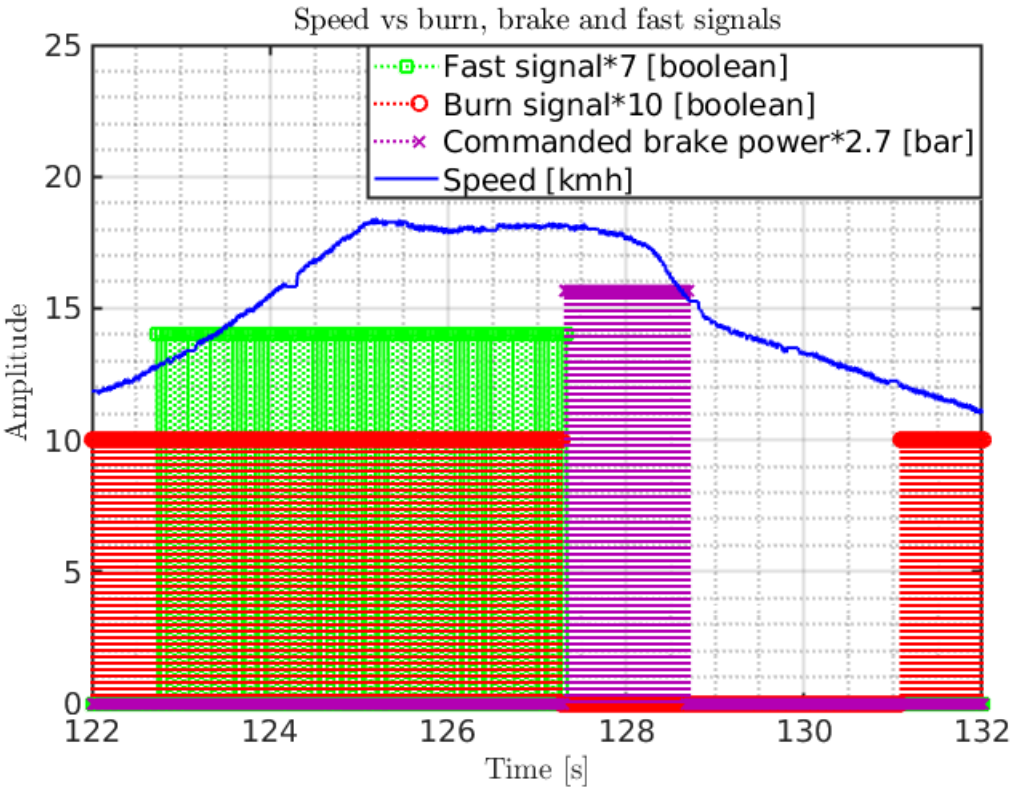



Figure 11.15: Speed control at t=122s to 132s (Challenge 1)

As can be seen in figure 11.16 the brake command is ceased at t=128.7s and the car stops decelerating at t=128.9s. The deceleration time from 17.95kmph to 14.65kmph takes 0.8s, yielding a deceleration rate of 4.12kmph/s. This is remarkably higher deceleration than during the second turn, although only measured over 0.8s it is mostly due to the uphill stretch of the track and the overshoot of the brake.

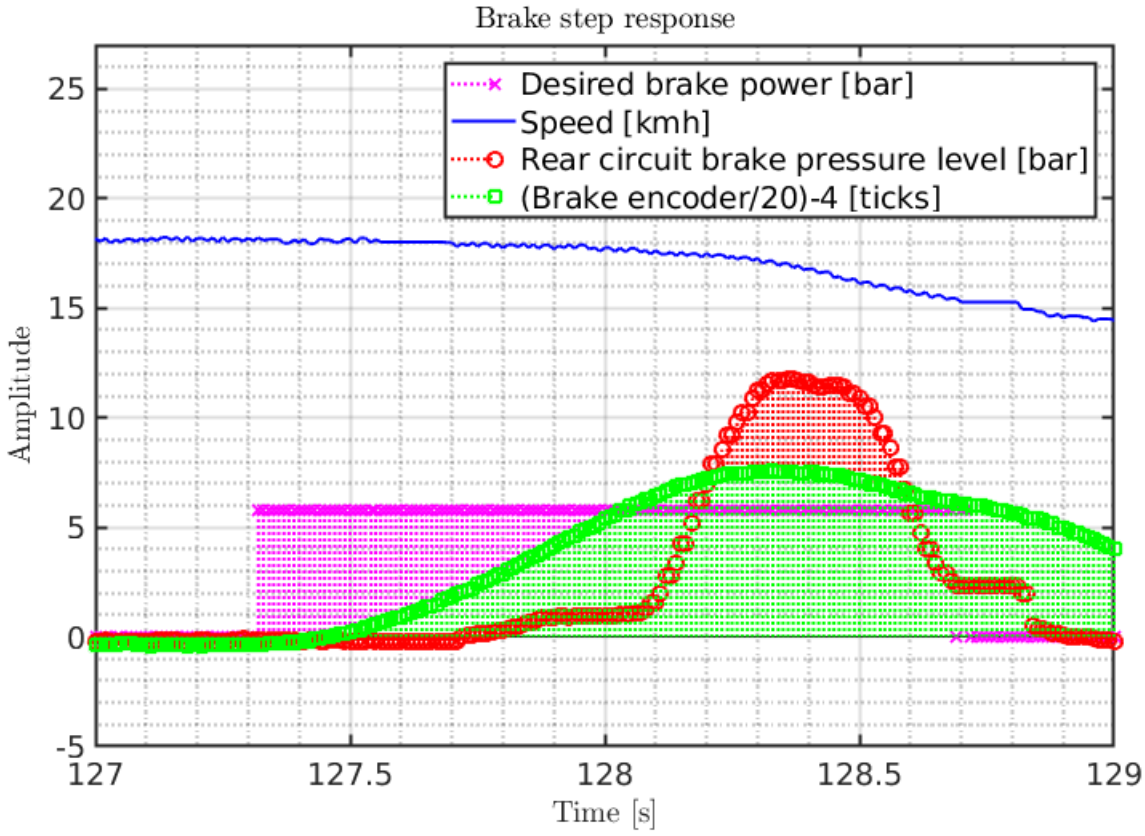



Figure 11.16: Brake control at t=122s to 132s (Challenge 1)

It should also be noted that the autonomous system only brakes the car until the speed is no longer larger than maxspeed+1.5kmph, although in practice it usually brakes further as it takes time to move the brake extender. The motor node is updating every 10ms so delay from the speed control program is minimal.

Another thing to note is that at t=125s the electronic system switches to 2nd gear while still driving uphill, resulting in the inability to accelerate up to 20kmph. In figure 11.17 the downhill stretch after the third turn can be seen. The autonomous system has problems detecting the stretch to be straight and switches between burning and braking.

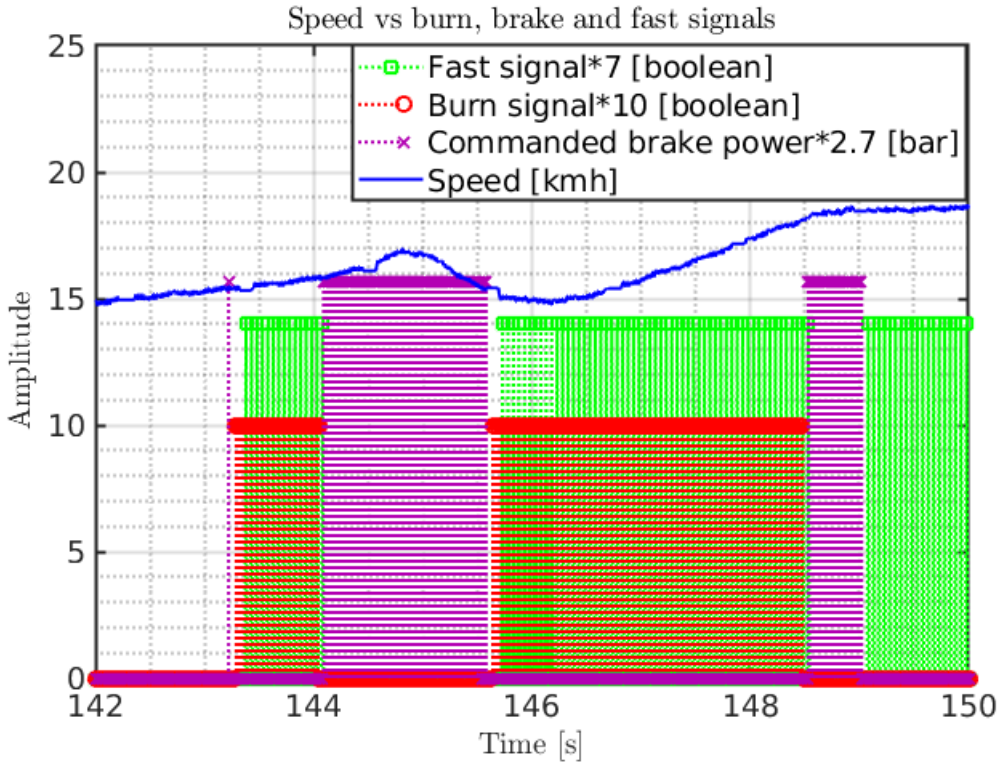



Figure 11.17: Speed control at t=142s and t=150s (Challenge 1)

In figure 11.18 steering control for the third turn can be seen. The track turns to the right followed by a small turn to the left which the autonomous system handles without problems. The radius of the third turn is found to be 15m.

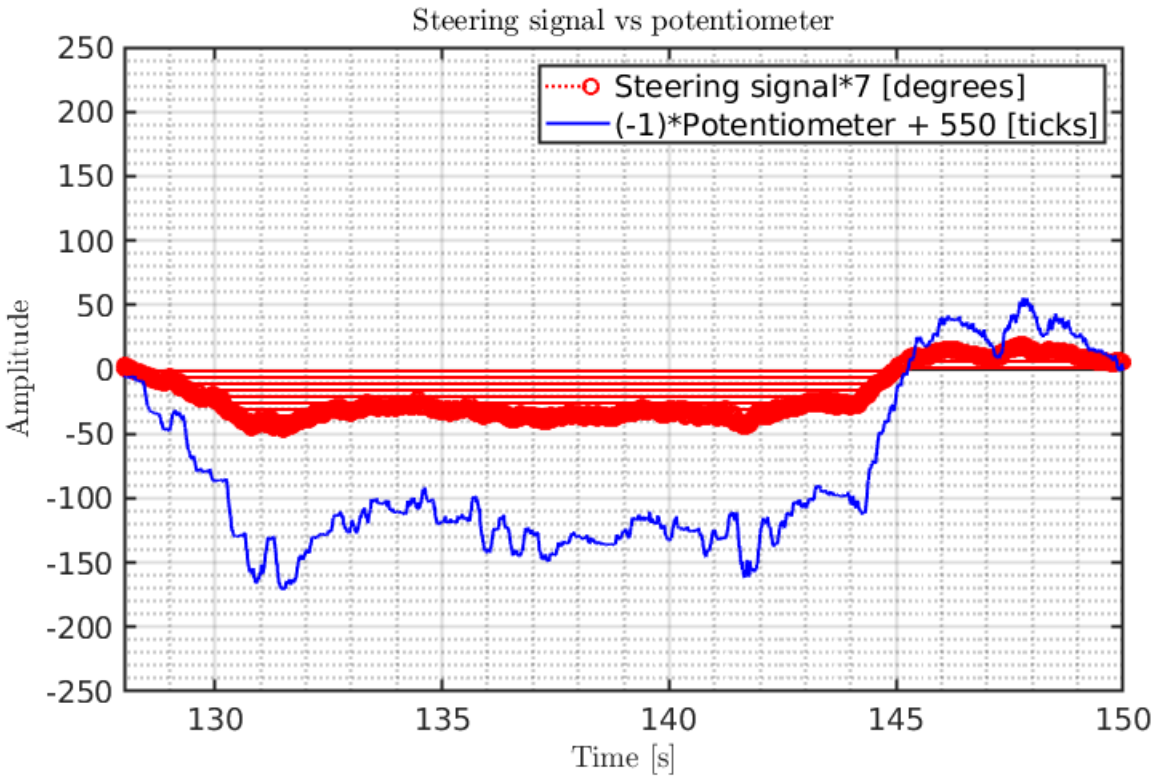



Figure 11.18: Steering control at t=128s to t=150s (Challenge 1)

## Until the finishing line

Figure 11.19 shows the speed control when the mechanical failure in the engine occurred. A long, straight stretch meant the autonomous system intended to burn up to fast speed. A switch from fast to slow and back to fast was too short, and the bendix drive in the engine locked itself in place. The autonomous system kept burning which kept the starter motor which pushes the bendix drive static, resulting in a stuck engine unable to start.

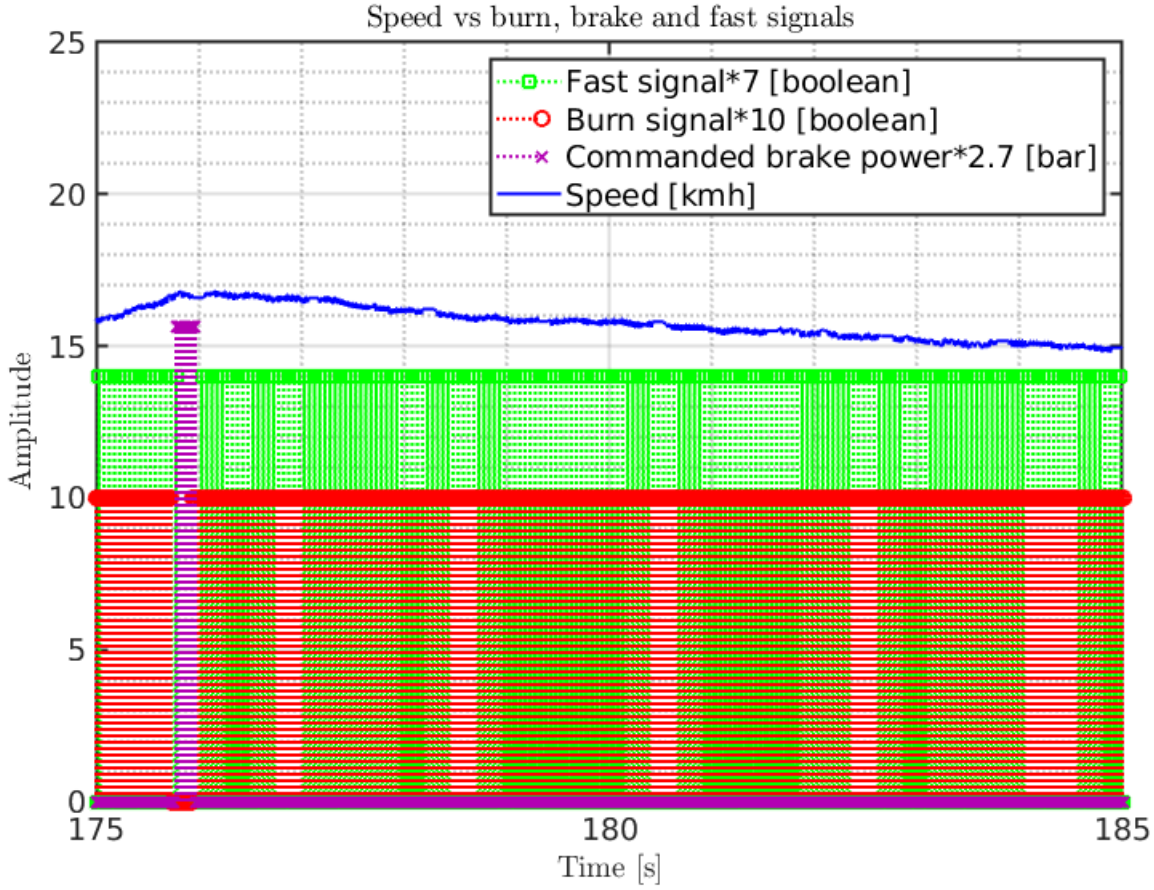



Figure 11.19: Speed control at t=175s to t=185s (Challenge 1)

The fourth turn at the bottom of the hill can be seen in figure 11.20 and the crossing of the finish line in figure 11.21. These small turns are also handled without problems, and only prove the effectiveness of the steering control.

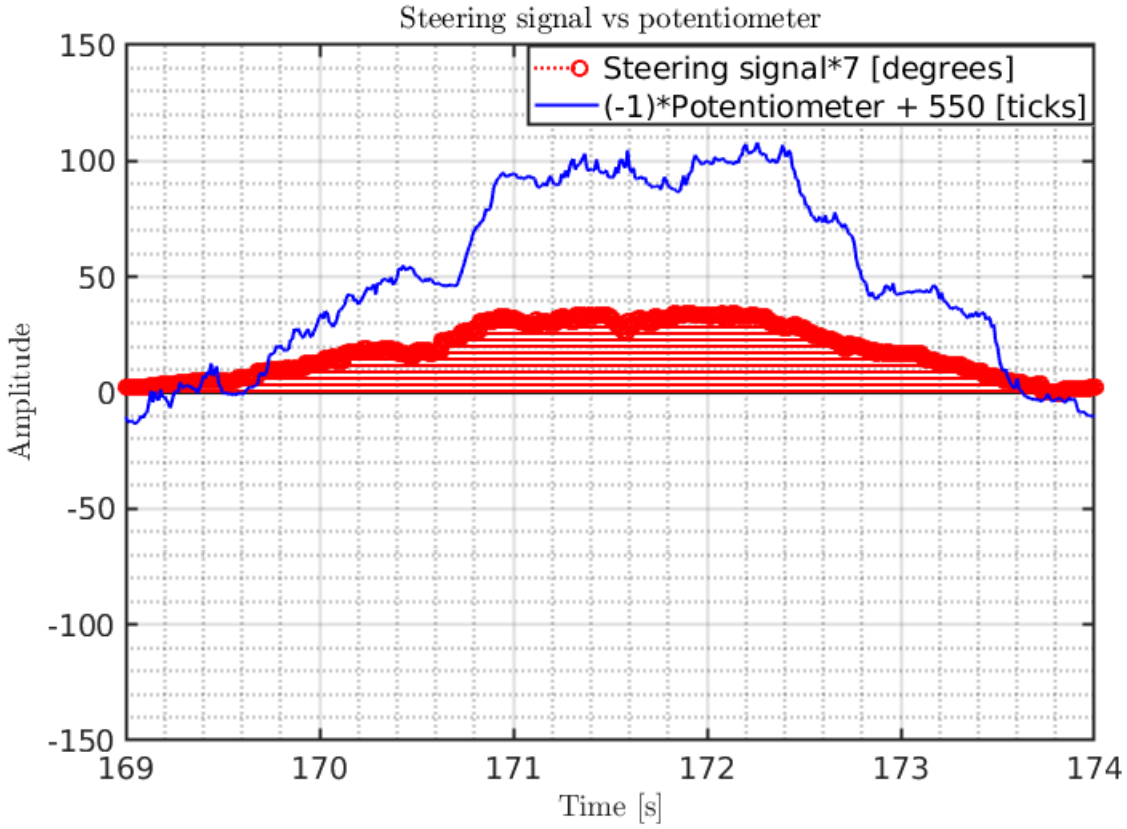



Figure 11.20: Steering control at t=169s to t=174s (Challenge 1)

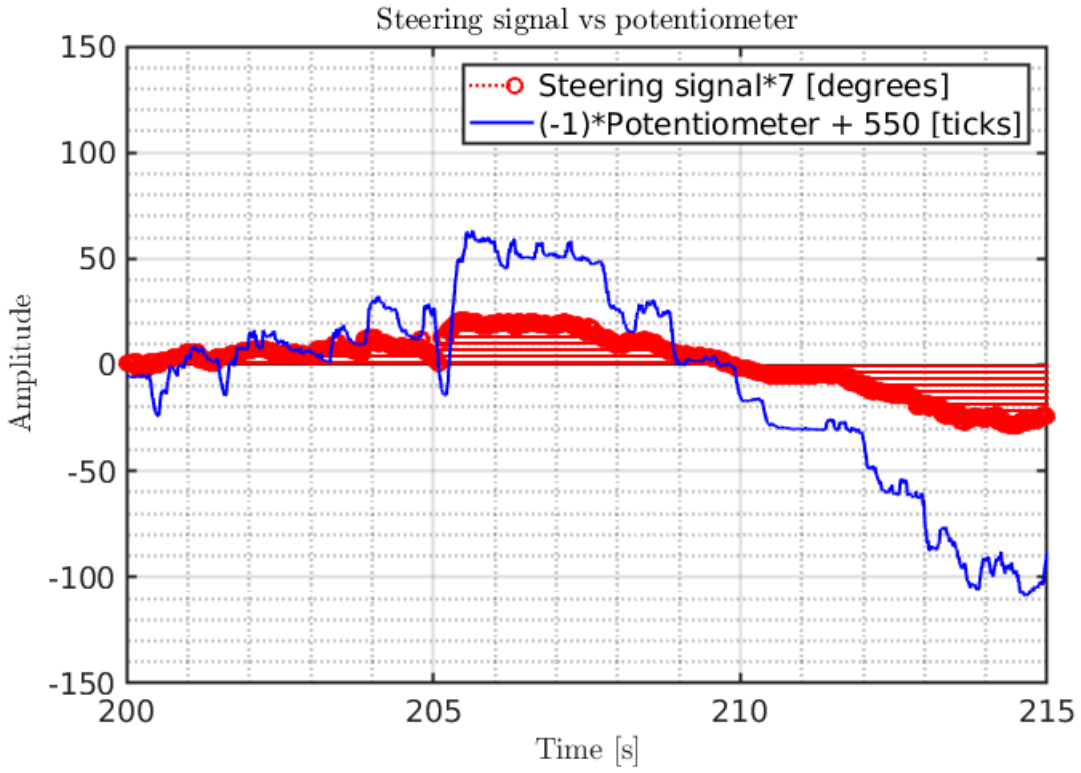



Figure 11.21: Steering control at t=200s to t=215s (Challenge 1)

As for the braking control: in figure 11.22 two consecutive brake commands can be seen. The second brake illustrates the behavior of the brake controller, complete with overshoot and a low settle value. This brake is followed by a series of commands too short for the brake to effectively activate. This is yet another example of the need for an improvement to the fast-slow algorithm.

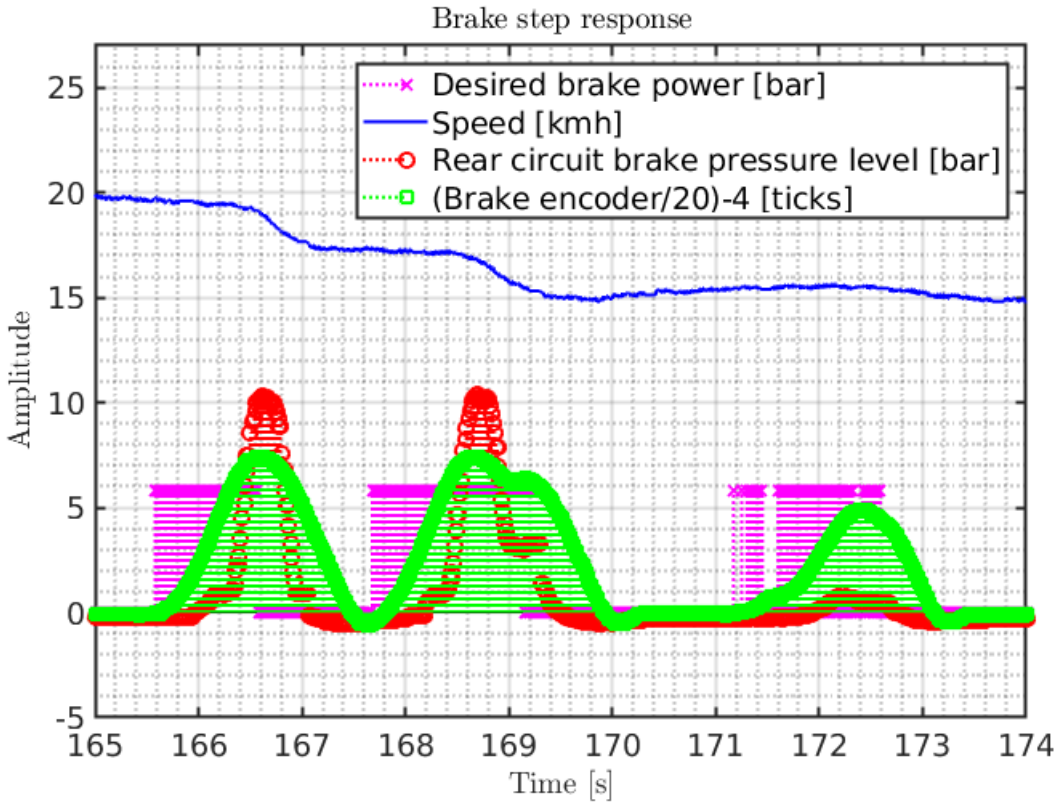



Figure 11.22: brake_t165-174

## Summarizing the autonomous system for the autonomous lap

The fast-slow algorithm was a partial success; it yielded in a faster lap time than without, however it was unstable on straight stretches of the track. When the track made small turns or when crossing small bumps in the road it switched back to slow. Moreover, the switch between slow and fast states was not always consistent as can be seen by switching back and forth in a few seconds. This problem led to the mechanical engine failure at 3:01, or 181s, when the burn was turned on and off too fast, leading to the bendix drive not connecting the starter motor to the motor. The lap time was 3:29 minutes or 209s, 15s slower than the ideal 194s.

The steering control worked satisfactory. The Voronoi program with post-processing sending points to the linedrive program proved an effective pair of steering the car through the lap, both straight stretches, turns, bumps, uphill and downhill. The weak point of this system is if the LiDAR node has fault measurements since the LiDAR is the main sensor used for navigation.

Challenge 1 was not limited to 2 attempts. Following the successful completion two more attempts were made. On both additional attempts the initial problem with path finding caused the car to steer into the left hand barrier. On these two attempts the speeds were supposed to be higher, namely 14-16kmph for slow and 20-25kmph for fast, however the car did not clear the starting line.

# Challenge 3: Parking

On July 8th the parking challenge was attempted. The challenge was limited to two attempts and both failed. On the first attempt the speed control brake signal interfered with the parking spot brake signal, which made the car crash into the end barrier. On the second attempt speed control was ignored and only the parking spot brake was used. Unfortunately the car brake activated too late and once again drove into the end barrier.

## First attempt

In the first attempt it was attempted to use the brake to both control the speed as well as to brake before the end barrier. Since both braking commands could be activated at the same time the end-brake command needed a significantly larger amplitude than the slow down-command due to the nature of the black-box brake controller, as described in the hardware chapter. However, this approach with two different brake commands proved a problem as the two nodes were not sufficiently integrated, causing the two nodes to overrule the other's commands. As can be seen in figure 11.23 this lead to the desired brake power alternating between 0 and 10 bar. Notice the high frequency fluctuations in the brake commands caused by the conflicting brake programs. Moreover, the simple brake function in the Voronoi node mistakenly identified the right hand barrier as the end barrier while steering into the parking spot trajectory as seen in figure 11.23 at t=16.2s.

The autonomous system was instructed to activate the brake 4.8m before the end barrier and the signal was sent at 4.99m before the barrier. The commands was sent at t=25.2s which is 2.25s before touching the barrier. While travelling at 4.9kmph (1.4m/s) the car must have been approximately 3.15m away from the barrier. However, due to the conflicting brake commands the brake simply did not activate at all, resulting in the car crashing into the end barrier as only the speed control could effectively activate the brake.

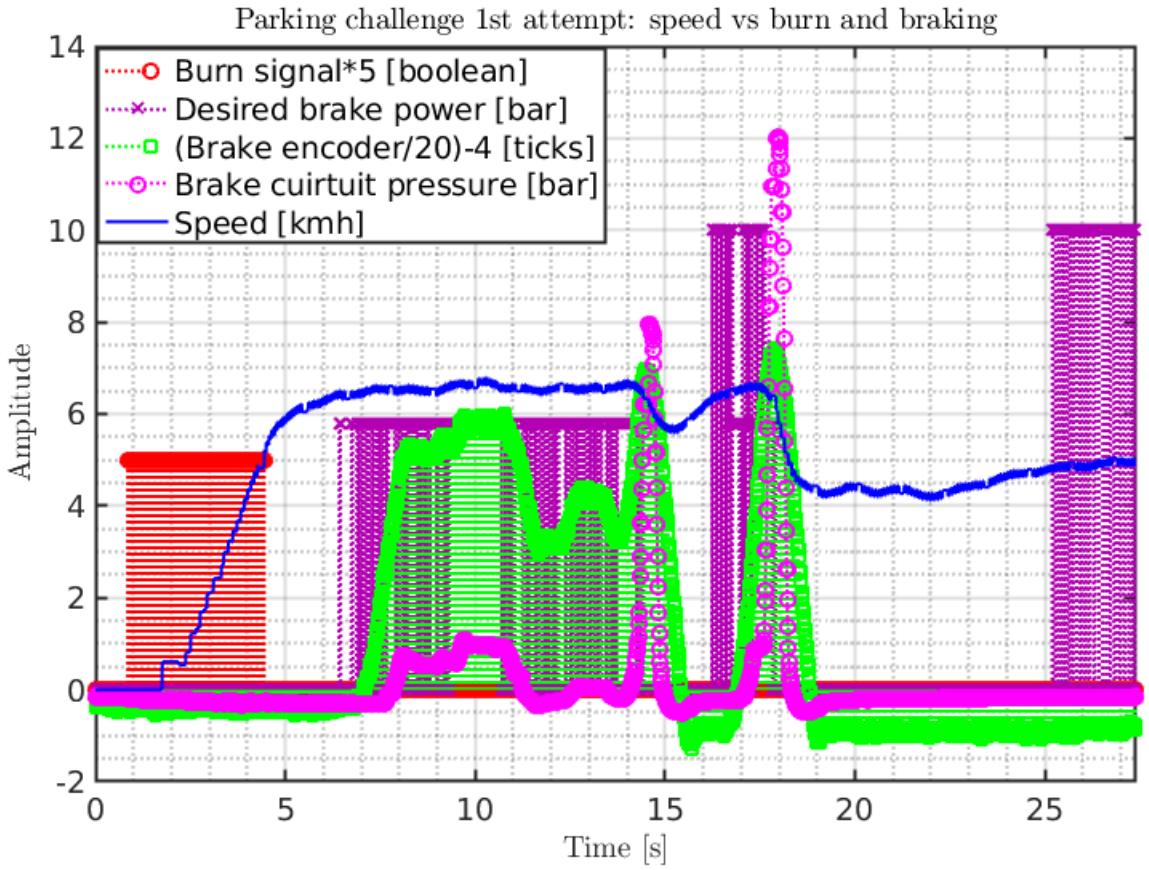



Figure 11.23: Parking challenge 1st attempt: burning and braking

In figure 11.24 the false positive detection of the right hand barrier can be seen at t=16.25s. At t=16.7s the speed has accelerated to 6.5kmph, which is the required 1.5kmph above maxspeed (5kmph) for the speed control to activate the brake. The two conflicting programs work in the sense that the commanded brake power alternates between the full stop brake power and the deceleration brake power level. As such the brake only truly activates when the speed control determines to decelerate.

This lead to the brake simply not being activated when detecting the end barrier, as can be seen in figure 11.25.

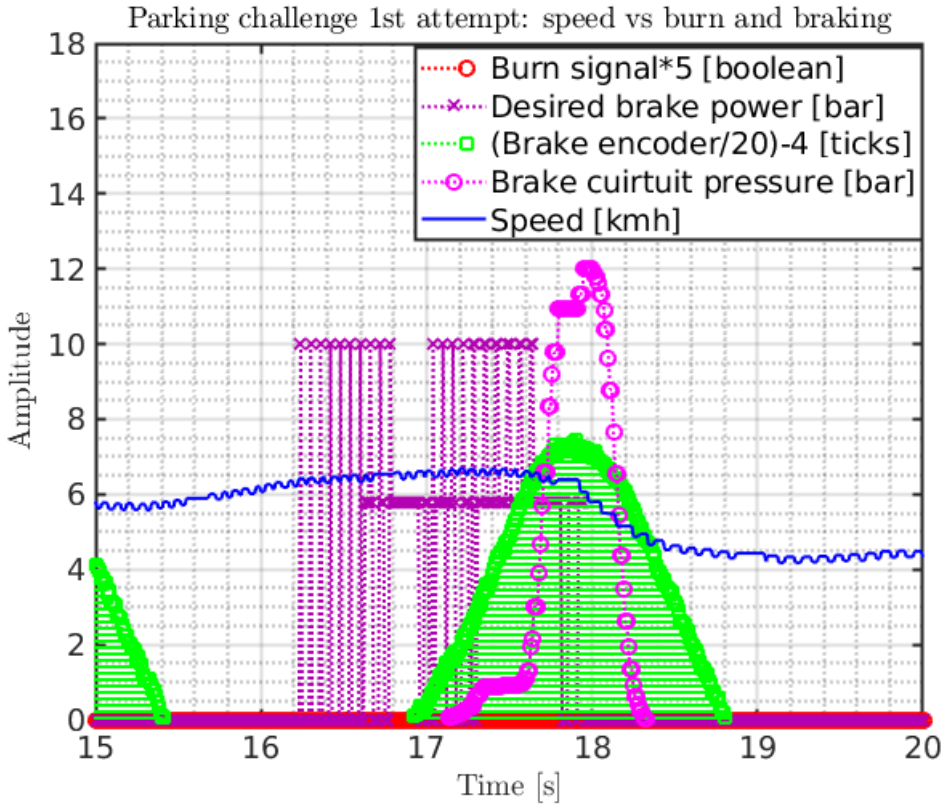



Figure 11.24: Parking challenge 1st attempt: burning and braking at t=15s to t=20s

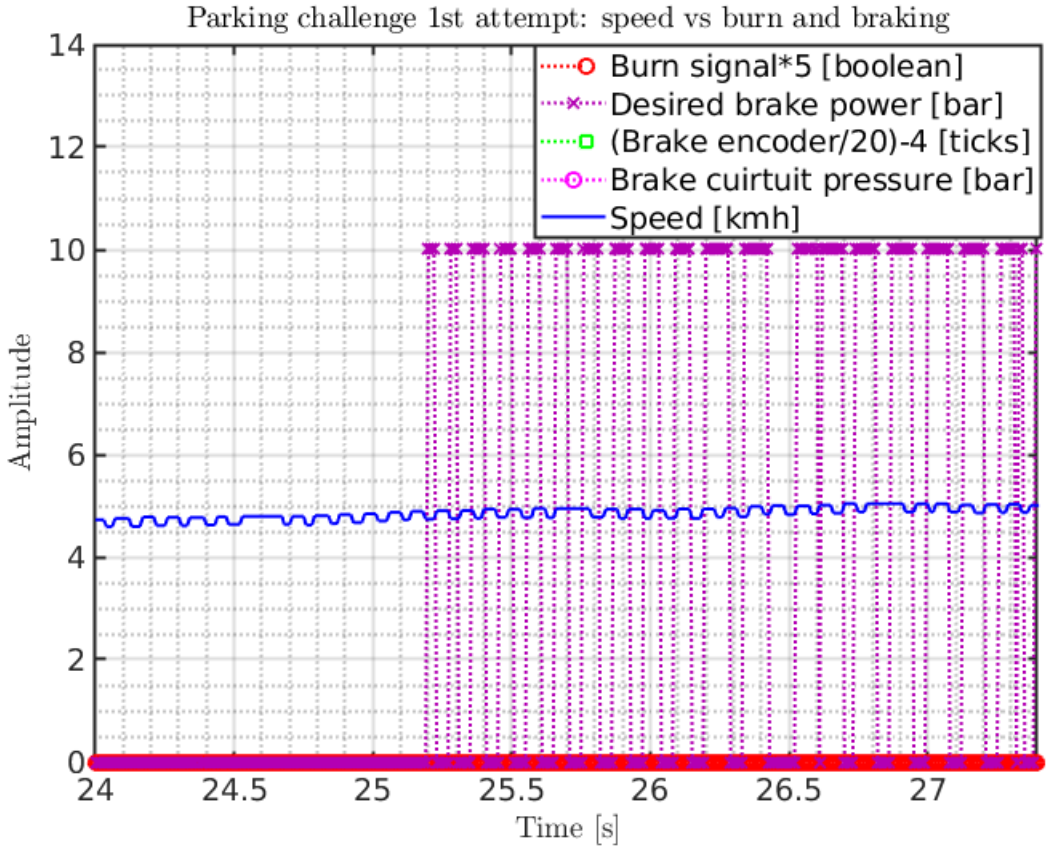



Figure 11.25: Parking challenge 1st attempt: burning and braking at t=24s to t=27s

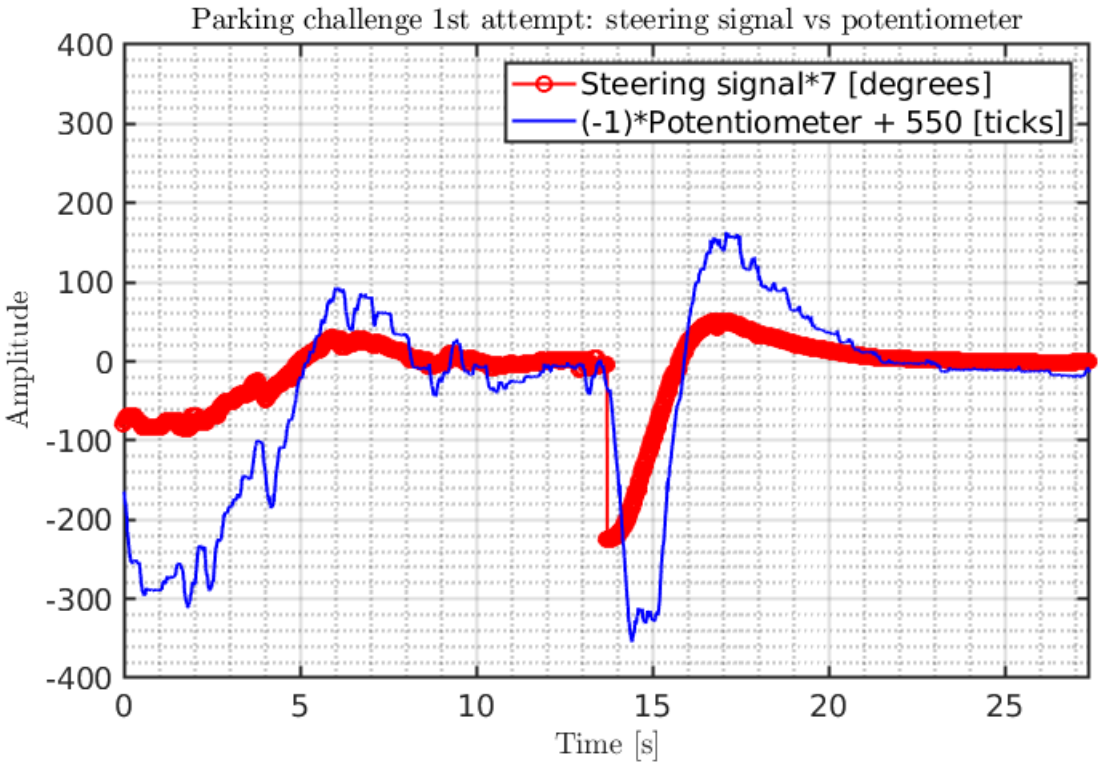



Figure 11.26: Parking challenge 1st attempt: steering signals and output

In figure 11.26 the smooth steering output can be seen as the autonomous system drives normal Voronoi until detecting the parking spot at t=13.7s after which a step to the right is commanded. Figure 11.27 shows the layout of the track as seen by the autonomous system. Comparing figures 11.26 and 11.27 shows that it takes several seconds from the autonomous system commanding a steering output until the car has actually steered.

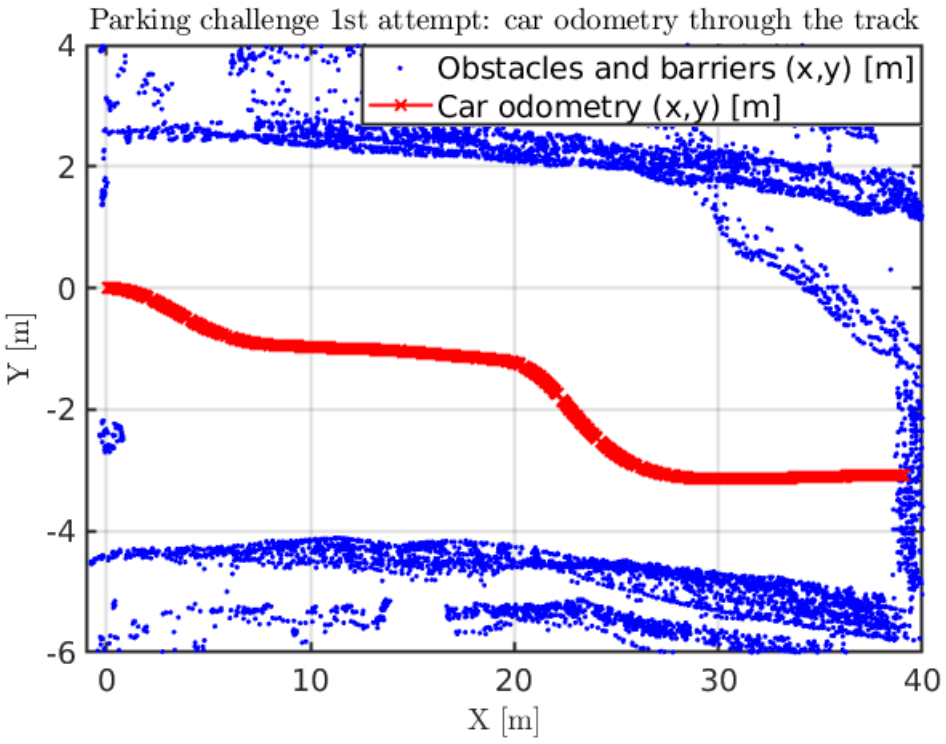



Figure 11.27: Parking challenge 1st attempt: odometry of the car in the detected track layout

## Second attempt

In the second attempt the speed control braking was disabled. This poised a problem as the track was slightly declining. As such the car would slowly accelerate above the desired maxspeed. The distance from which the Voronoi node would brake before the end barrier was also increased to 5.2m. However, this time the autonomous system did not detect to right hand barrier as the end barrier.

As can be seen in figure 11.28 and figure 11.29 the brake command was sent at t=21.5s when the car was 5.28m from the end barrier. Since the speed decreases from 8kmph to 4kmph at t=23.25s the deceleration rate is found to be 4kmph/s, resulting in the car touching the barrier with 3.5kmph (0.97m/s). However, due to the increased speed resulting from slowly accelerating downhill without speed control the car simply did not stop in time before touching the barrier.

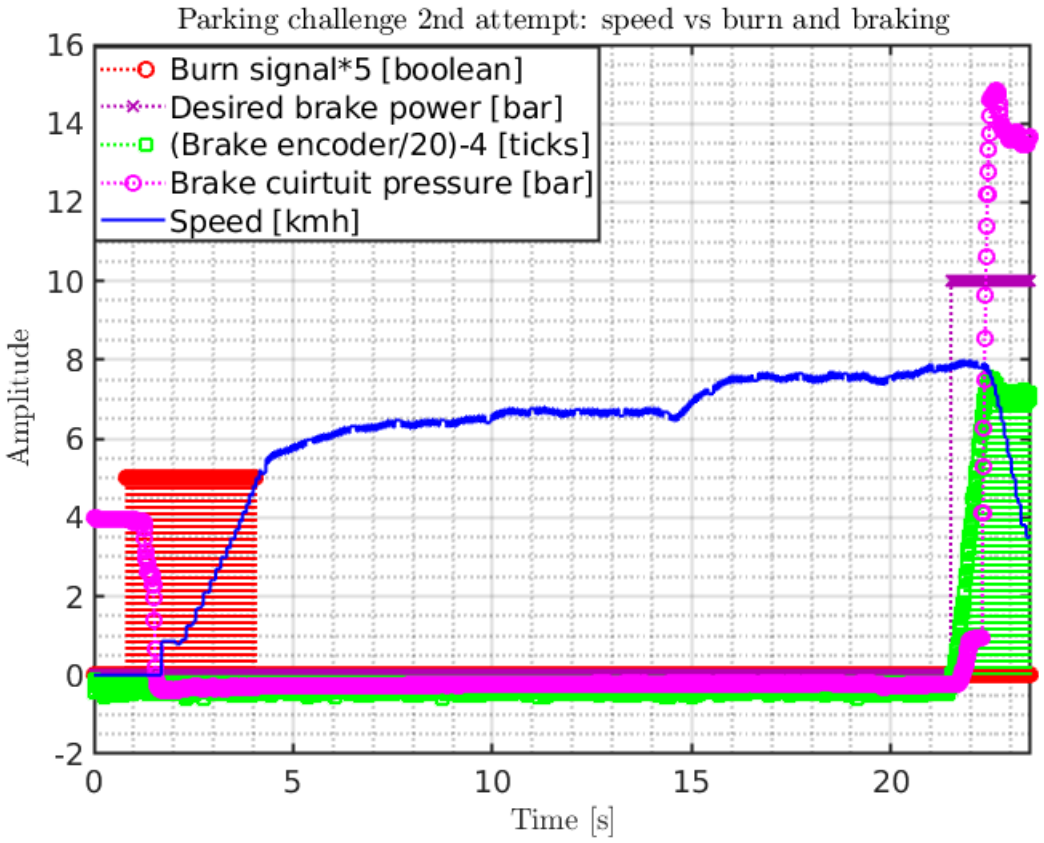



Figure 11.28: Parking challenge 2nd attempt: burning and braking

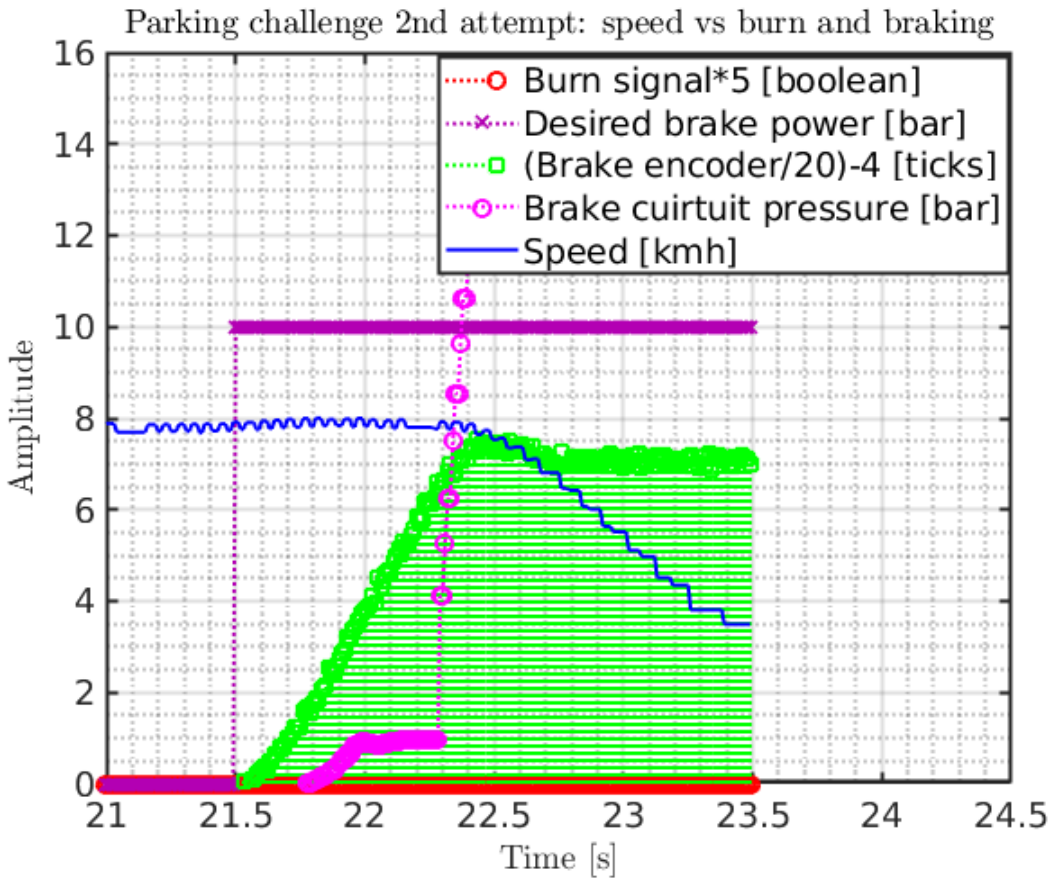



Figure 11.29: Parking challenge 2nd attempt: burning and braking at t=21s to t=23s

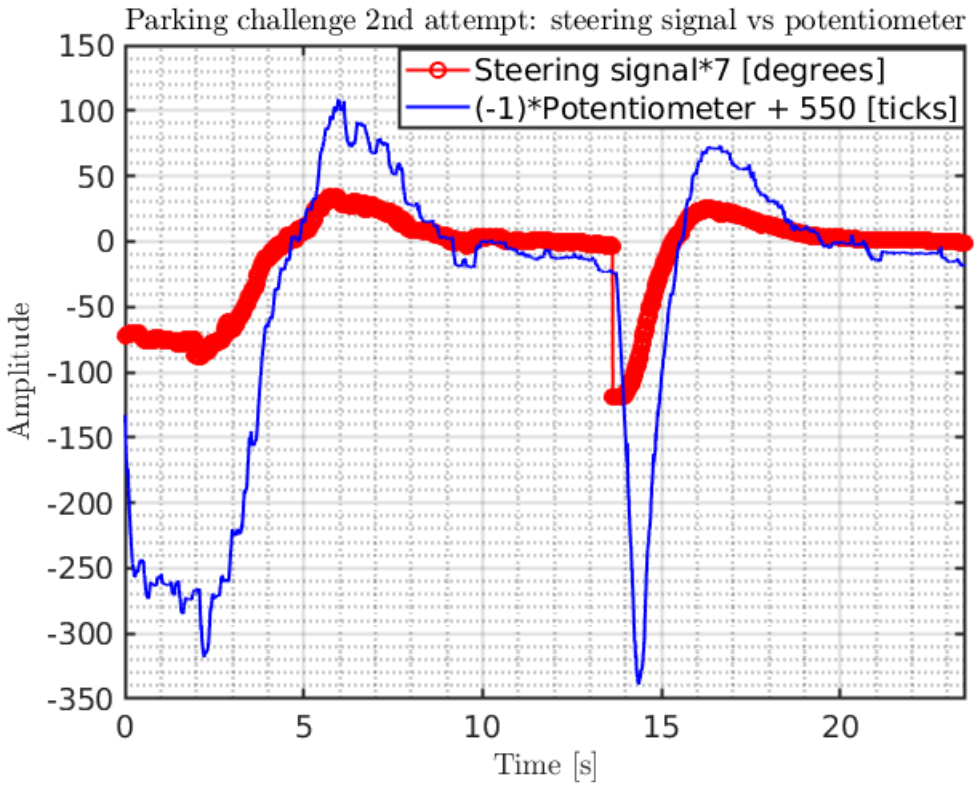



Figure 11.30: Parking challenge 2nd attempt: steering signals and output

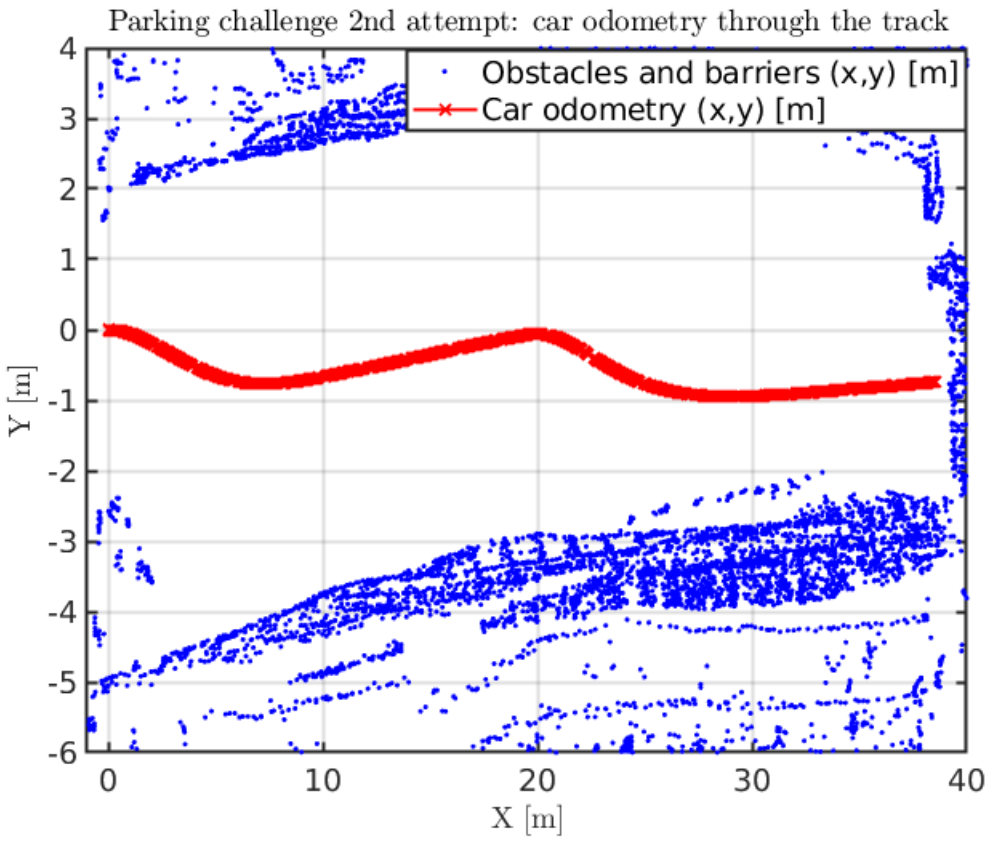



Figure 11.31: Parking challenge 2nd attempt: the car's odometry in the detected track layout

### Summarizing the autonomous system for parking

The parking challenge was not completed in London. As can be seen in figures 11.25 and 11.29 the car simply did not brake in time, partly due to the brake being activated too late, and to the relatively high speed.

However, considering that the autonomous system has previously completed this challenge during practice as well as the Paris qualification event, it must be hypothesized that it was possible for the autonomous system to work with a tweaks to break further in advance.

## Challenge 4: Obstacle avoidance (gates)

On July 8th the gates challenge was attempted and completed. The challenge was limited to two attempts only. The first two attempts did fail, the first time due to an error in the new gate classifier program, and the second time by touching the left cylinder in the last - and narrowest - gate. Fortunately, however, the dimensions of the track layout were inspected and found to be faulty - the distance between the second and third pairs of gates was only 9m and not 10m, and the third gate width was only 2.3m and not the spec-

ified 2.5m. Thus, a third attempt was granted and this time the challenge was completed.

In this challenge the Voronoi node was deactivated, leading the car to simply drive straight ahead until the gates were detected by the gate classifier node.

## First attempt

In the first attempt the new gate classifier program was tried, however due to a small error in the software the autonomous system did not publish the drive points to the linedrive node, resulting in the car simply driving straight (with a drive point of (0,0,0)) as can be seen in figure 11.32.

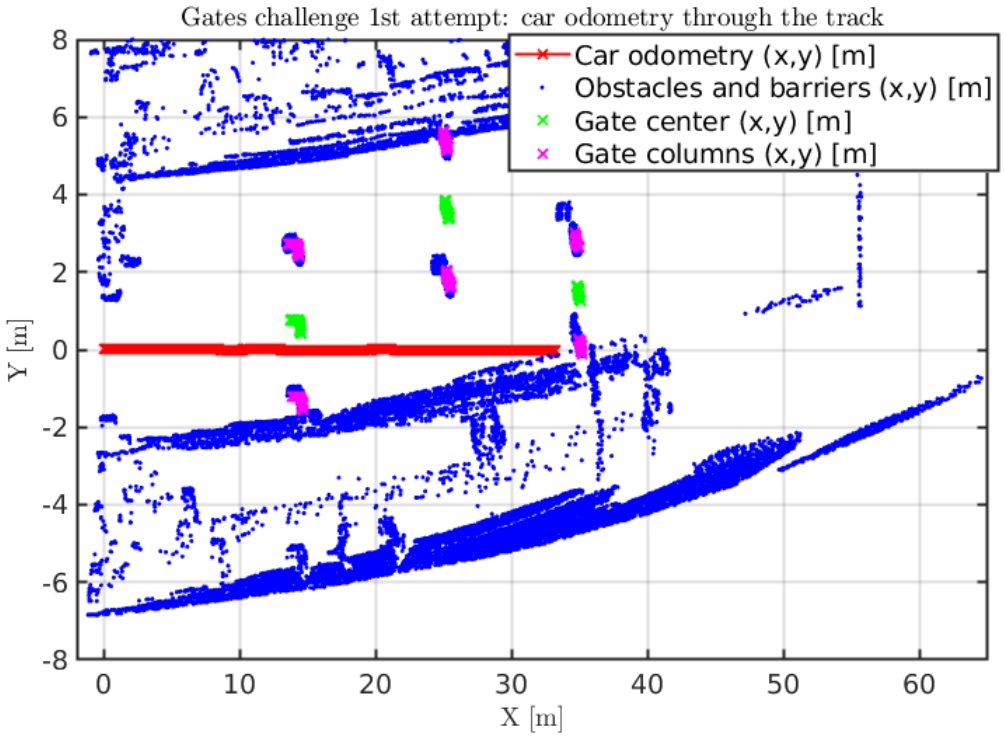



Figure 11.32: Gates challenge 1st attempt: car odometry in the detected track layout along with detected gates

## Second attempt

The cause of failure in the first attempt was localized to a small error the new gate classifier program which was fixed in the second attempt which can be seen in figure 11.33. The autonomous system effectively detected all three gates, however the steering controller values in the linedrive node did not allow the car to clear the 3rd gate without touching the left hand cylinder, however immediately after it was discovered that the dimensions of the track did not live up to the dimensions specified in the rules. The distance between gate 2 and 3 was only 9m as opposed to the specified 10m, and the width of the 3rd gate was found to be 2.3m as opposed to the specified 2.5m. As such a third attempt was

granted.

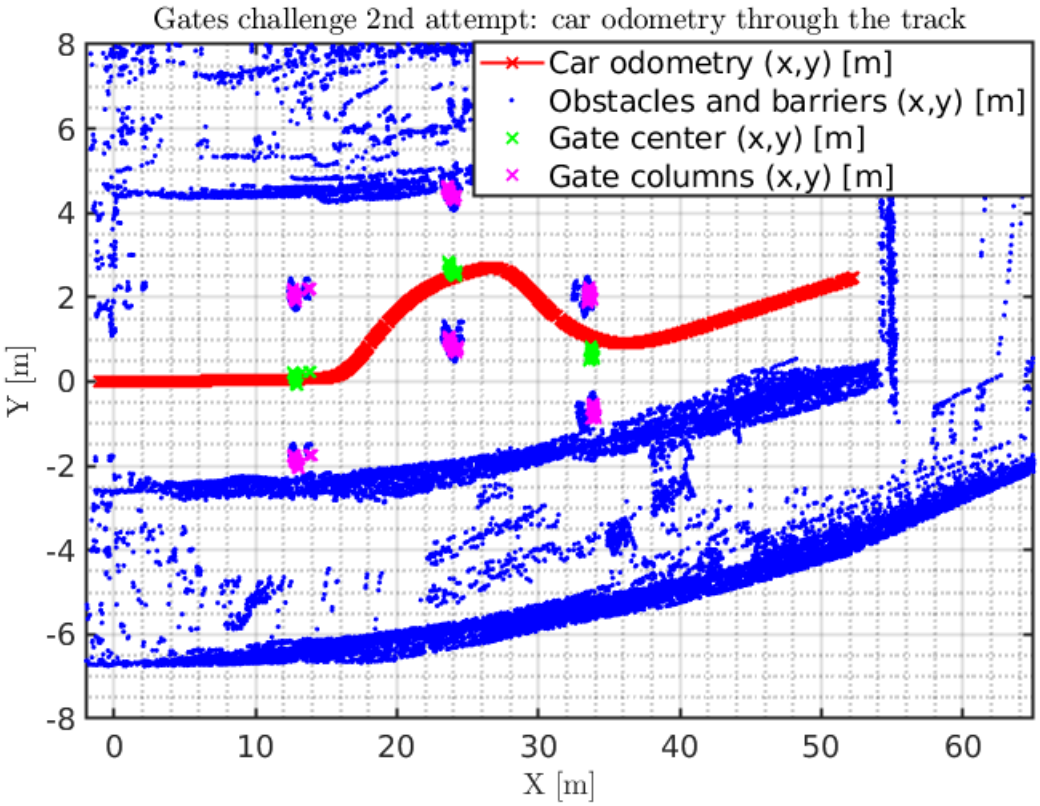



Figure 11.33: Gates challenge 2nd attempt: car odometry in the detected track layout along with detected gates

In figure 11.34 the steering output can be seen. The left step to the 2nd gate is commanded at t=13.5s and the right step to the 3rd gate at t=22.9s.

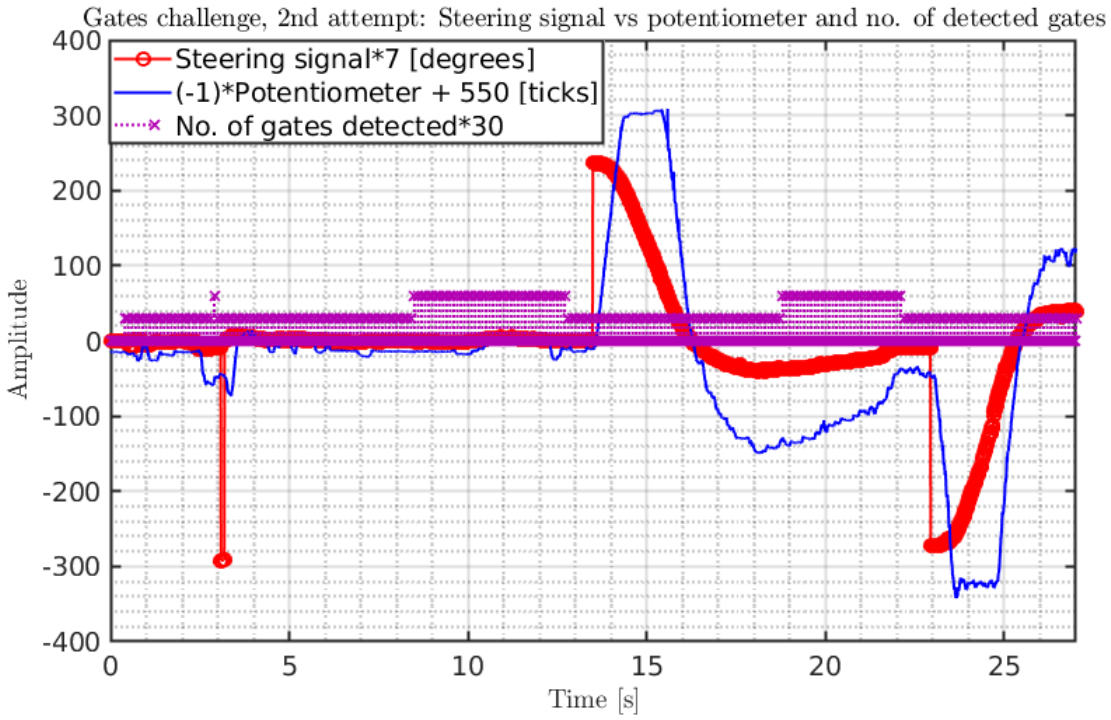



Figure 11.34: Gates challenge 2nd attempt: steering signals and output

## Third attempt

In the third attempt the car successfully cleared all three gates as can be seen in figure 11.35.

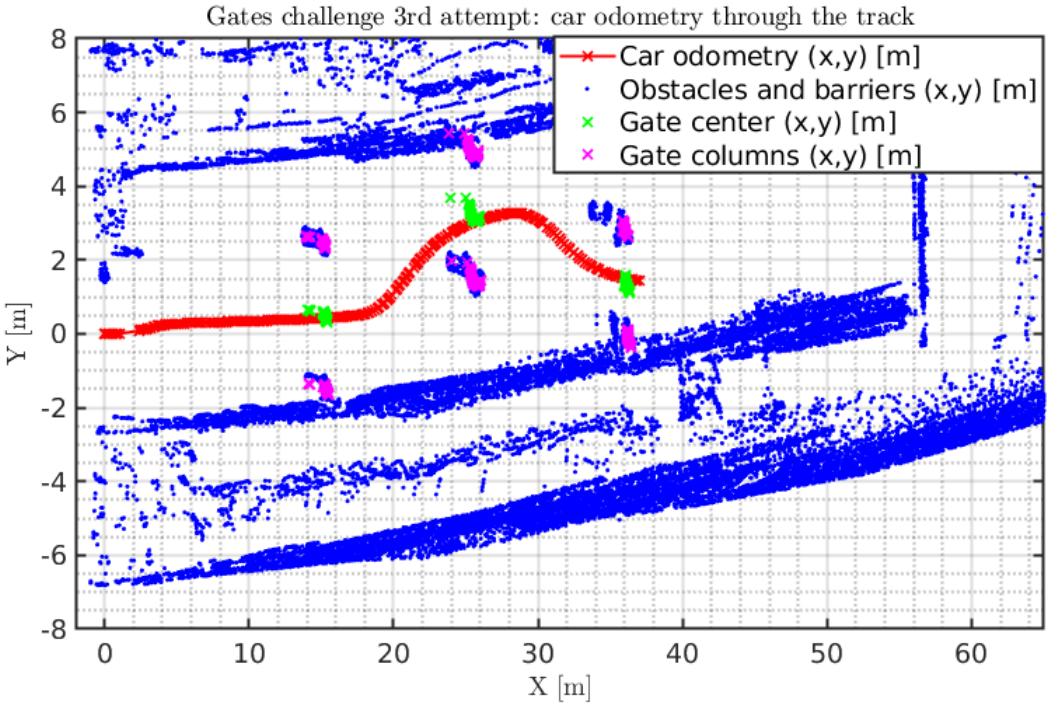



Figure 11.35: Gates challenge 3rd attempt: car odometry in the detected track layout along with detected gates

In figure 11.36 the steering output can be seen. The left step to the 2nd gate is sent at t=14s, and the right step to the 3rd gate is sent at t=22.7s.

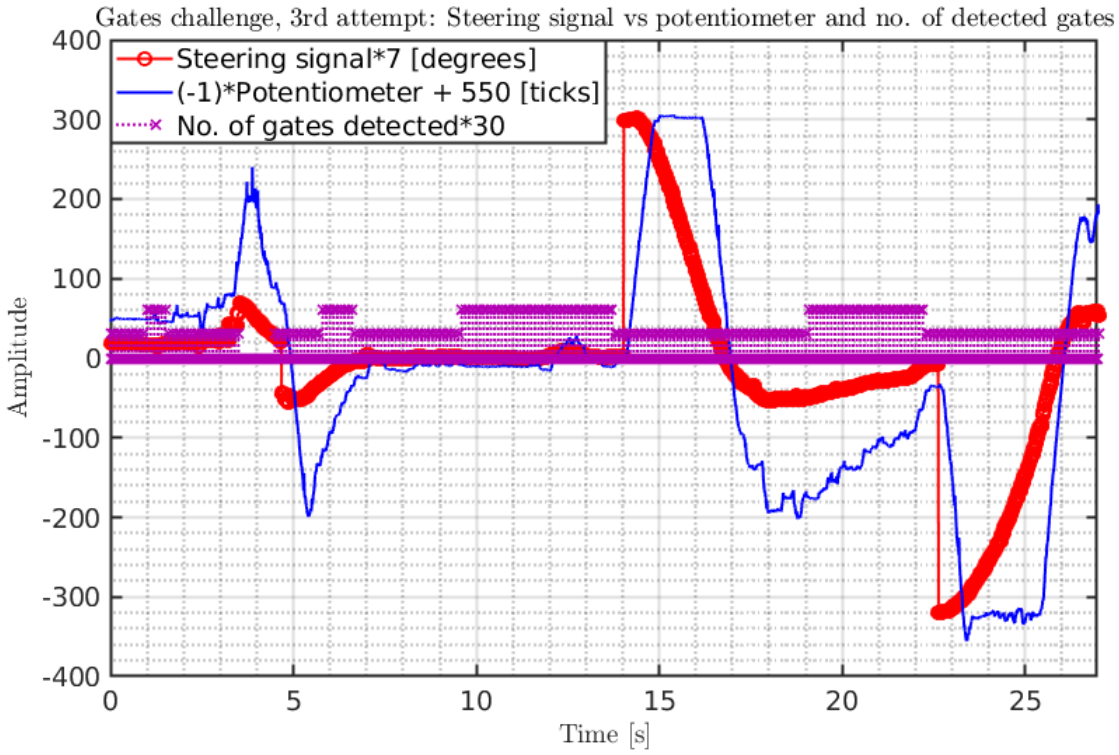



Figure 11.36: Gates challenge 3rd attempt: steering signals and output

## Summarizing the autonomous system for gates

The gate classifier method proved effective in detecting gates and their coordinates, and integration with controller was successful. The first attempt was failed due to a simple error in the software. The second attempt failed as the autonomous system did not step the 3m within 9m. In the third attempt the distances were increased sufficiently for the near-optimal parameter values to yield a success. Steering generally worked well, however this challenge highlights the importance of proper tuning parameters for the linedrive node.

It should be noted that the precision of the steering depend on the speed of the car because of the steering wheel actuators stepping time, hence why $4 - 6km/h$ was used as speed reference. Furthermore, this track did not pose any elevation which might have caused acceleration, however since no parking brake was necessary it must be hypothesized that the speed control brake could have effectively kept the speed at the desired level.

# 12 Discussion

## Approach

In this project, the goal was to develop an autonomous system that based on sensor input, can calculate the optimal route and use this route as reference signal, to control steering angle, and speed and brake power by simple controllers in order to participate and hopefully win the SEM AUC competition.
As a result of the considerable scope of project, it was proposed to solve the autonomous problem with simple and effective solutions to ensure a working system at the competition.
The system design is based around driving in the middle of the road by calculating, left side, right side and middle with use of a voronoi diagram. By doing this it is possible to feed the middle line to a controller and steer the car. Additional calculations based on the shape of the middle road are used to set speed and brake power references.

A simulation environment was used in the design process of the the project to develop and integrate the different algorithms. However, as soon as possible, testing was moved from simulation to the real car as this gave a better insight into the specific needs of the system. Furthermore, the simulation run time was exceptionally slow (factor 0.05) and so it was mostly used as a way to confirm algorithm designs before testing on the Ecocar.

## Results

The design of the system resulted in achievement of the project requirements.
The car was able to drive 1000m autonomously without any external interference, at an average speed of $16.5km/h$. The car steered to stay in the middle of the road as intended and the kinematic model can effectively predict the car steering outputs.
For precision, the car was able to redeem a $3m$ offset in side-direction within $10m$ of travelling in the direction. Furthermore, the car is able to turn with a radius of $8m$ when driving $7m/s$.
Parking and slalom type tracks were completable but braking was not sufficiently robust.
The collective system has a runtime of less than $100ms$, when used on a Intel NUC i7-8650u.
This means the project requirements were fulfilled.

# Critical evaluation

## Steering control

Calculating a middle line and using this result to steer with a simple controller proved to be a simple but effective way to solve the basic navigation problem of the challenges presented from SEM AUC.
Neither mileage nor racing efficiency lines were considered, since they are not awarded additional points at the SEM AUC competition, therefore, a safer navigation was a better solution.
We recognize that different approaches can be used to solve the problem, but as can be seen from the results, this method fulfilled the requirements.

## Speed control

Using hysteresis control on the current engine installed in the Ecocar is still considered the optimal solution since PID-control would not be compatible, as it would damage mechanical components of the engine.
The "fast-slow" algorithm that sets the desired speed reference based on the slope and offset of the steering-line reference had issues with high frequency noise. It was only used for challenge one, and resulted in a faster lap-time as was the purpose.
However, the speed reference algorithm for was not robust enough to deal with the high frequency noise. If the authors were to participate in the competition again, they would choose to omit this "fast-slow" algorithm from the system, as an issue occurred which caused a mechanical error in challenge 1. The benefit of gaining a faster time does not outweigh the risk involved. It is instead proposed to run the whole track with a singular speed optimal for the goal average speed, or develop a new and better speed reference control.
The car has managed to effectively clear a 90-degree turn with turn radius $7.5m$ while driving $25km/h$, and since the turn radii on the final track were significantly larger than what has been tested. The third turn on the track has a radius of 15m, twice the radius that was tested and completed in the speed test. Because of this it theorized that the car could have completed the track with higher speeds, achieving the average $18km/h$ goal.

## Brake control

The Ecocar had a brake implemented with a PID-controller.
As such, the brake was used by simply setting a reference signal. Generally the brake overshoots the desired brake reference signal and has an irregular step-time, making it difficult to do tasks such as precision parking as too many factors are uncontrollable.
The authors could have implemented brake control that depended on the current speed of the car.

## Algorithm Design

The system designed in this projected had a target program run time of 100ms. This was achieved by implementing the Voronoi algorithm using $O(n \cdot log(n))$ time, and the

Voronoi node using $O(n^2)$ time. Aside from this the signals between nodes were kept simple. It was managed to create a system with a simple structure which other students can add to.

# Competition

## Challenge 1: autonomous Lap

The reason for choosing the speed reference intervals as $12 - 14km/h$ and $18 - 20km/h$ was to have risk-free turning and then compensate by going faster on the straight pieces of road. $12 - 14km/h$ was defined as a critical speed for the first lap, to ensure that the driver would have time to intervene with the system, if an accident was to occur in a turn.

## Challenge 3: parking

In the first attempt the autonomous system sent the end brake command while 4.99m from the barrier although the hard-coded value in the autonomous system was 4.8m. In the second attempt the brake command was sent while 5.28m from the barrier although the hard coded value was 5.2m. This difference may have been caused by delay in the system caused by program run times.

Unfortunately the parking challenge was not completed in London. It is speculated that the challenge went wrong because it contained many unknown variables that could have been dealt with, with another brake system and more testing.

## Challenge 4: gates

The gate challenge was difficult due to the step requirements of 3m step to either side cleared within 10m. Figure 8.7 shows that the live steering closely matches the simulated steering using the same controller values. This is a good validation for the steering control model. As can also be seen in figure 8.7 it must have been possible for the autonomous system to have completed even the wrongly dimensioned track in the second attempt if the controller values chosen for the third attempt were used in the second.

Another thing to note is that in the autonomous system the car can only start stepping towards the next gate when the rear wheels (origo of the car) have passed the gate center coordinate. However, the difference in length between the rear wheels and the front tip of the car is 2m. It can be hypothesized that a possible improvement for the autonomous system is to let the car turn sooner, either 1 or 2m before the rear wheels pass the gate coordinate. With such a change it might have been possible for the autonomous system to complete the wrongly dimensioned track experienced in the second attempt even without tuning the controller.

## Additional comments

The system was designed in order to complete the kind of challenges presented at the SEM AUC competition and by placing first at competition it has been seen that the design has lived up to its premise.

# 13 Future work

In this section future work for the project is suggested.

## Design of navigation

The following is suggested improvements pertaining specifically to the navigation algorithm.

### Follow wall

When a situation occurs where the car can not detect a left and right side barrier the system could issue a simple instruction of following one wall at a pre-determined distance in order to avoid crashing into barriers on one side.

### Safety braking

From a safety perspective an algorithm could be implemented that evaluates whether the car with the current speed and angle is able to steer away from barriers, and if not, to activate an emergency brake.

### Using camera for path finding

It could be possible to use a fusion of the LiDAR and camera vision to detect the barriers along the track, which may increase robustness in the distances where the LiDAR has detected false gaps between the barriers.

## Steering

The kinematic model derived unfortunately contained a small error during the project which meant it was not used to tune the parameters for the live challenge. Instead, values determined empirically were used. However, since the London event the model has been made to work and could be used to improve the driving values. This same model can also be used to calculate the desired speed for different turn radii in challenge 1.

A more robust fast-slow algorithm could be made using the 5th degree polynomial fit instead of a linear regression. The fit contains the information of the curvature of the

route ahead. The desired speeds could become a function of the turn radius of upcoming turns coupled with a lookup table from the kinematic model. The values for speed, $k_{dist}$ and $k_{angle}$ could thus be dynamically calculated on-line based on the polynomial fit.

## Speed

The fast-slow algorithm can be made more robust to noise, e.g. by using a low pass filter and making minimum delays for how quickly the algorithm can switch between states.

## Parking

It could be possible to improve the parking program with two different but properly integrated brake pressure levels, so as to brake with one level prematurely and then brake with a higher level when approaching the end barrier. The parking brake program could generally benefit from some validation. The brake could be PID controlled.

An improvement for the program could be to increase the braking distance and include the car odometry angle relative to the parking spot angle. This would prevent false positives from the side barriers while braking the car before touching the end barrier. Alternatively, the target brake pressure could be increased although this would mean a more abrupt braking, including a higher risk of breaking the brake pedal on the car.

## Hardware changes

If the brake is changed to an electronic brake, it would be easy to make a PID-controller based on the feedback from the velocity, allowing the brake control setpoint to be a deceleration rate instead of brake pressure reference.

# 14 Conclusion

By using a Voronoi diagram approach it is possible to calculate a path that is perfectly in the middle by using a set of distance measurement, that has been categorized to belong to the left or right side of the road. This can be done in $O(n^2)$ time, and most of the data processing, such as constructing the diagram, is done in $O(n \cdot log_2(n))$ running time, with $n$ being LiDAR measurement data points.
With the chosen parameters for merging walls of left and right, the typical road visibility is around $10m - 25m$ depending on flatness of surface and the execution time is less than $50ms$ on the Intel i7-8650U processor, meaning even at top-speed of $7m/s$, the car will have a new route after a displacement of only $35cm$. It is also possible to deal with more complex track layouts such as roundabout or Y-splits, by having the option to go either left or right.
The resulting middle line from the voronoi diagram can be processed to provide a reliable reference signal for a controller and the information about the route can be represented as a polynomial fit to predict the speed which should be set on the vehicle, based on the curvature of the road.

By using a simple P-controller, it is possible to have precise control of the steering angle for the Ecocar without oscillation even for speeds up to $7m/s$ , when using lines as references. Because the controller accepts a simple input, it should be possible for any navigational method to supply the needed $(x, y, \theta)$ reference. This results in easy integration with parking spot and gate detection which was demonstrated at the SEM AUC competition.
A kinematic model has been made and validated for the car, to allow tuning of the p-controller by simulation. The kinematic model can be used even for other robots or if a new car is made, as long as the parameters: *steering wheel limits, steering wheel turn-rate limits* or *distance between rear and front wheels* are changed accordingly in the model.
The car was able to:

- Perform a 3m step to the side within 10m in the driving direction. Demonstrated at challenge 4 at SEM London
- Perform turns with a turn radius of 8m. This was completed in the speed tests where the car cleared a turn with radius 7.5m at DTU
- Drive up to 25km/h while still running in the middle of the road
- Perform speed control, that achieved an average speed of 16.5km/h at challenge 1 at SEM London 1.5km/h less than goal time

- Drive autonomously for 1000m. This was completed both at the distance test and during challenge 1 at SEM London
- Drive in the middle of the road at all times at challenges 1,3,4 in London by using LiDAR data
- Park by using camera software developed by other students. Demonstrated at DTU, in front of HRH Prince Joachim, and at SEM qualifications in France. Where braking down before obstacle collision was also demonstrated
- Control steering without oscillations, and providing a smooth experience for the driver
- Run the algorithms in less than 100ms. This was achieved as the navigational algorithm execution times were found to be no larger than 40ms when run on the intel NUC, i7-8650u

**The Ecocar team, DTU-Roadrunners were able to win 1st prize in the SEM AUC competition by using the navigational algorithm, steering controller, speed controller and brake script described in this project, by combining this project with work from the rest of the Roadrunners Team.**

# Bibliography


[1] The Phidget steering stepper motor, *Datasheet partially on the webiste, partially in mechanical drawing*,
Website: https://www.phidgets.com/?tier=3&catid=24&pcid=21&prodid=352,
Visited July 2018

[2] ROS.org, *About ROS*,
Website: https://http://www.ros.org/about-ros/,
Visited July 2018

[3] ROS.org, *ROS topics*,
Website: http://wiki.ros.org/Topics,
Visited July 2018

[4] gazebosim.org, *Gazebo website*,
Website: http://gazebosim.org/,
Visited July 2018

[5] ROS.org, *ROS rviz site*,
Website: http://wiki.ros.org/rviz,
Visited July 2018

[6] Velodyne Vlp-16 puck, *Vlp-16 datasheet*,
Website: http://velodyneLiDAR.com/vlp-16.html,
Visited July 2018

[7] Henning Si Høj, *Platform Integration for Autonomous Self-Driving Vehicles*,
Technical University of Denmark, June 2017

[8] Oliver Lynggaard Topp, *Situation Awareness for an Autonomous Vehicle*,
Technical University of Denmark, April 2018

[9] *BOOST:Voronoi*,
Website: https://www.boost.org/doc/libs/1_67_0/libs/polygon/doc/voronoi_main.htm,
Visited July 2018

[10] Andersen N.; Ravn O., *Autonomous Robot Systems Programming*, Slides, page 6,
Technical University of Denmark, 2016

[11] *Benchmark values for BOOST:Voronoi*,
Website: https://www.boost.org/doc/libs/1_53_0/libs/polygon/doc/voronoi_benchmark.htm,
Visited July 2018

[12] Nikolaus Correll, Introduction to Autonomous Robots, v1.7
*Ackermann steering*, pages 55-57
Magellan Scientific, October 6, 2016

[13] Manas Sharma, *Polynomial fit*, Website: https://www.bragitoff.com/2015/09/c-program-for-polynomial-fit-least-squares/,
Visited July 2018

[14] *Camera Datasheet*,
Website: http://www.elpcctv.com/8mp-highdefinition-usb-camera-module-usb20-sony-imx179-color-cmos-sensor-75degree-lens-p-223.html,

# Appendices

# A Datasheets

## A.1 The Ecocar

The specifications of the Ecocar can be seen in figure A.1.

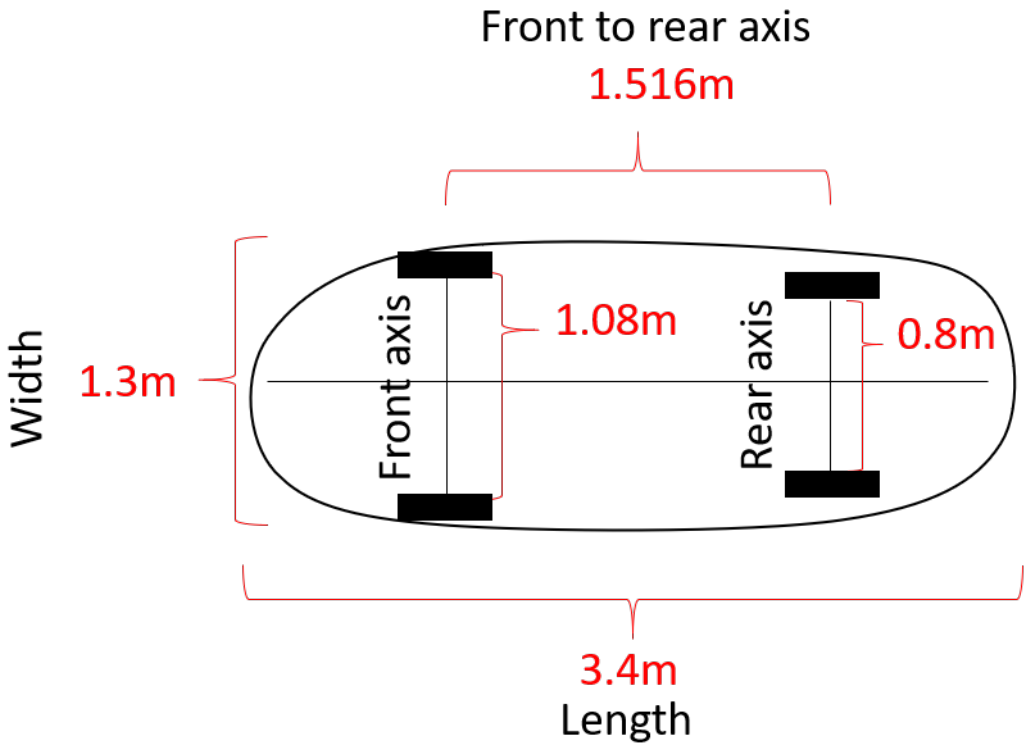



Figure A.1: The dimensions of the Ecocar

## A.2 Velodyne LiDAR

Velodyne LiDAR®

# Puck™

REAL-TIME 3D LiDAR SENSOR

Automotive | Robotics | Mapping | UAV | Security | Industrial

**VLP-16**

### Velodyne LiDAR PUCK™

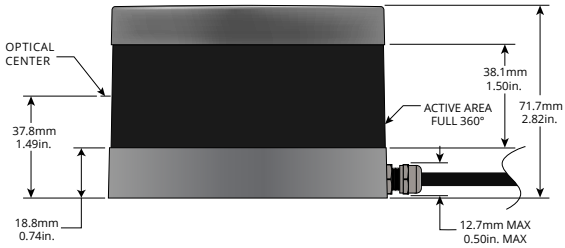
Velodyne's Puck (VLP-16) is the smallest, cost-optimized product in Velodyne's 3D LiDAR product range. Developed with mass production in mind, the Puck is far more cost-effective than comparable sensors, and it retains the key features of Velodyne's breakthroughs in LiDAR: Real-time, 360°, 3D distance and calibrated reflectivity measurements.

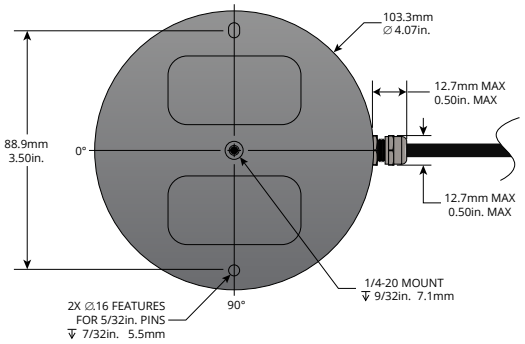

### Real-Time 3D LiDAR

The VLP-16 has a range of 100 m, and the sensor's low power consumption, light weight, compact footprint and dual return capability make it ideal not only for autonomous vehicles but also for robotics, terrestrial 3D mapping and many other applications.

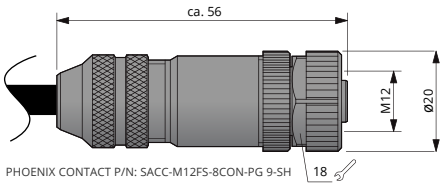
Velodyne's LiDAR Puck supports 16 channels, ~300,000 points/second, 360° horizontal field of view and a 30° vertical field of view, with ±15° up and down. The Puck does not have visible rotating parts, and is highly resilient in challenging environments while operating over a wide temperature range.

## DIMENSIONS *(Subject to change)*




## M12 CONNECTOR OPTION




For other connector options contact
**Velodyne Sales (sales@velodyne.com)**

www.velodynelidar.com

VLP-16


## Real-Time 3D LiDAR Sensor

The VLP-16 provides high definition 3-dimensional information about the surrounding environment.

| | Specifications: |
|---|---|
| **Sensor:** | • 16 Channels<br>• Measurement Range: 100 m<br>• Range Accuracy: Up to ±3 cm (Typical)[1]<br>• Field of View (Vertical): +15.0° to -15.0° (30°)<br>• Angular Resolution (Vertical): 2.0°<br>• Field of View (Horizontal): 360°<br>• Angular Resolution (Horizontal/Azimuth): 0.1° – 0.4°<br>• Rotation Rate: 5 Hz – 20 Hz<br>• Integrated Web Server for Easy Monitoring and Configuration |
| **Laser:** | • Laser Product Classification: Class 1 Eye-safe per IEC 60825-1:2007 & 2014<br>• Wavelength: 903 nm |
| **Mechanical/ Electrical/ Operational** | • Power Consumption: 8 W (Typical)[2]<br>• Operating Voltage: 9 V – 18 V (with Interface Box and Regulated Power Supply)<br>• Weight: ~830 g (without Cabling and Interface Box)<br>• Dimensions: See diagram on previous page<br>• Environmental Protection: IP67<br>• Operating Temperature: -10°C to +60°C[3]<br>• Storage Temperature: -40°C to +105°C |
| **Output:** | • 3D LiDAR Data Points Generated:<br>- Single Return Mode: ~300,000 points per second<br>- Dual Return Mode: ~600,000 points per second<br>• 100 Mbps Ethernet Connection<br>• UDP Packets Contain:<br>- Time of Flight Distance Measurement<br>- Calibrated Reflectivity Measurement<br>- Rotation Angles<br>- Synchronized Time Stamps (µs resolution)<br>• GPS: $GPRMC and $GPGGA NMEA Sentences from GPS Receiver (GPS not included) |

63-9229 Rev-H

**For more details and ordering information, contact Velodyne Sales (sales@velodyne.com)**

1. Typical accuracy refers to ambient wall test performance across most channels and may vary based on factors including but not limited to range, temperature and target reflectivity.
2. Operating power may be affected by factors including but not limited to range, reflectivity and environmental conditions.
3. Operating temperature may be affected by factors including but not limited to air flow and sun load.

CLASS 1 LASER PRODUCT



**Velodyne LiDAR, Inc.** 345 Digital Drive, Morgan Hill, CA 95037 / lidar@velodyne.com / 408.465.2800 | www.velodynelidar.com

## A.3 Brake actuator

TECHLINE MEDLINE CARELINE

# Actuator LA30

## Data sheet

**LINAK.COM**/TECHLINE
**LINAK.COM**/MEDLINE-CARELINE

LINAK
WE IMPROVE YOUR LIFE

# LA30

LA30 is a powerful actuator yet small enough to fit most applications. The actuator can be supplied with options such as built-in potentiometer for servo operation or an extra powerful motor for increased speed and strength (S-motor). In addition to industrial and agricultural applications, the actuator is also ideal for positioning satellite dishes.

**Features and options:**
- 12/24V DC permanent magnet motor
- Max. thrust up to 6000 N (LA30LK)
- Stainless steel piston rod
- Elegant and compact construction with small installation dimensions
- Protection Class: IPX0/66
- Colour: Black
- Speed max. 65 mm/s (LA30 S-motor with 12 mm pitch)
- Low noise level
- Steel construction for all bearing parts
- Acme thread spindle for optimum efficiency
- Extra powerful motor (S-motor)
- L-motor for system actuator
- IP66 (by ordering LA30 with plastic housing)
- Double-acting brake - increased self-locking ability (LA30 with 6 or 9 mm pitch + LA30 S-motor with 6 or 9 mm pitch and LA30L) which ensures that all these types are fully self-locking
- Potentiometer for positioning the actuator 0-1 K ohm, 0-5 K ohm or 0-10 K ohm
- Reed switch (only LA30 L-motor versions): 8 pulses pr. spindle revolution
- Mechanical spline function. Safety feature by using the mechanical spline, the actuator can only push
- Safety nut (only in push)
- Terminal cover (only LA30 L-motor versions)
- Ball screw (K) (only LA30 L-motor versions)
- Ball screw and safety nut (KAS) (only LA30 L-motor versions)
- Ball screw, safety nut and splines (KSM) (only LA30 L-motor versions)

**Usage:**
- Duty cycle: 10%, 2 minutes continuous use followed by 18 minutes not in use
- Ambient temperature +5° to +40°C
- Storage temperature -40° to +70°C
- Compatible with control boxes CB8, CB12,
- Should LA30 be used with a non LINAK control unit, please ask the nearest LINAK representative for further details

**LINAK.COM**/TECHLINE
**LINAK.COM**/MEDLINE-CARELINE

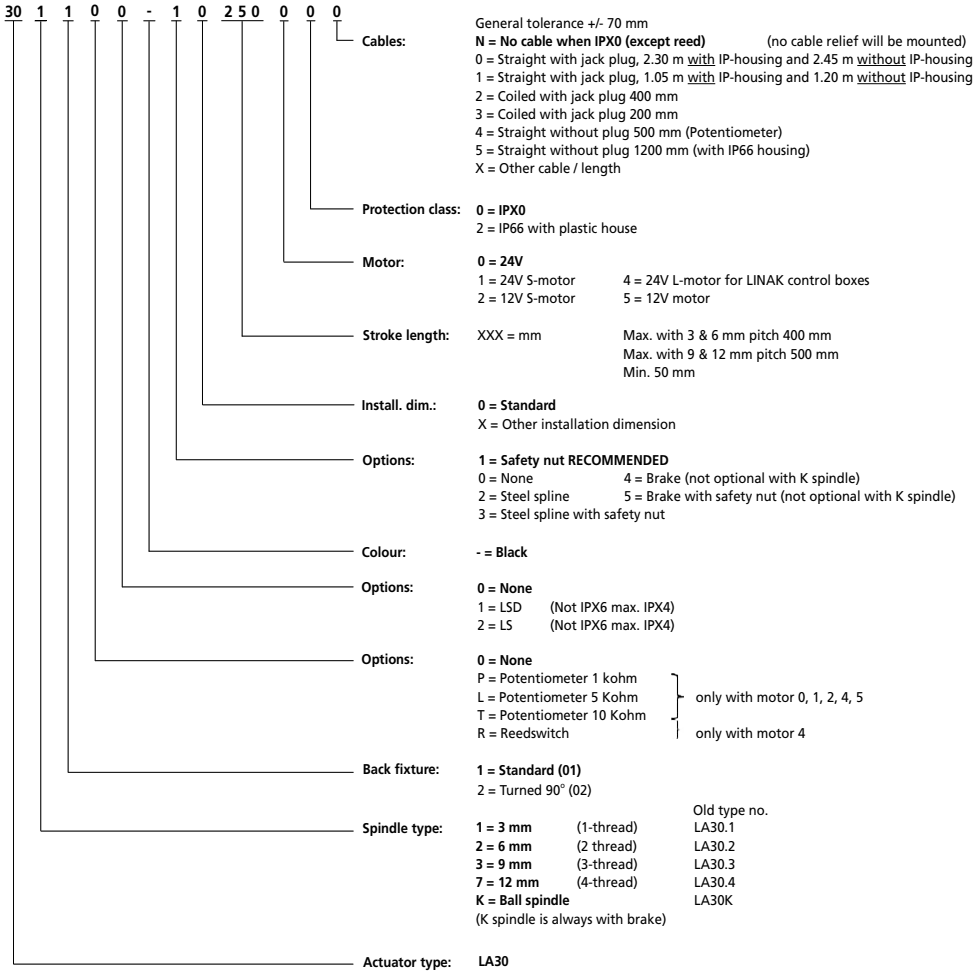
**Technical specifications:**

| New type | Spindle pitch (mm) | Thrust max. Push (N) | Thrust max. Pull (N) | *Self-lock max. With/without brake (N) | Typical speed 0/full load (mm/s) | Stroke length (mm) | | | | | | | | Typical amp. at full load 12V | 24V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 307xx0-4xxxx0/5xx | 12 | 1000 | 1000 | 1000/0 | 48/24 | 50 | 100 | 150 | 200P | 250 | 300 | 350 | 400P | 14 | 7 |
| 303xx0-4xxxx0/5xx | 9 | 1500 | 1500 | 1500/400 | 42/20 | 50 | 100 | 150P | 200 | 250 | 300P | 350 | 400 | 14 | 7 |
| 302xx0-4xxxx0/5xx | 6 | 2000 | 2000 | 2000/500 | 18.5/14 | 50 | 100P | 150 | 200P | 250 | 300 | 350 | 400P | 14 | 7 |
| 301xx0-xxxxx0/5xx | 3 | 3000 | 3000 | 3000/3000 | 16/9 | 50P | 100P | 150 | 200P | 250 | 300 | 350 | 400 | 14 | 6.4 |
| 307xx0-4xxxx1/2xx | 12 | 1000 | 1000 | 1000/0 | 65/35 | 50 | 100 | 150 | 200P | 250 | 300 | 350 | 400P | 20 | 10 |
| 303xx0-4xxxx1/2xx | 9 | 1800 | 1800 | 1800/0 | 52/25 | 50 | 100 | 150P | 200 | 250 | 300P | 350 | 400 | 20 | 10 |
| 302xx0-4xxxx2xx | 6 | 2000 | 2000 | 2000/500 | 34/20.3 | 50 | 100P | 150 | 200P | 250 | 300 | 350 | 400P | 18 | - |
| 302xx0-4xxxx1xx | 6 | 2400 | 2400 | 2400/500 | 34/20.3 | 50 | 100P | 150 | 200P | 250 | 300 | 350 | 400P | | 9 |
| 301xx0-xxxxx1/2xx | 3 | 3500 | 3000 | 3500/3500 | 17/9.5 | 50P | 100P | 150 | 200P | 250 | 300 | 350 | 400 | 18 | 9 |
| 307xx0-4xxxx4xx | 12 | 1000 | 1000 | 1000/0 | 26/20 | 50 | 100 | 150 | 200P | 250 | 300 | 350 | 400P | - | 2.5 |
| 303xx0-4xxxx4xx | 9 | 2000 | 2000 | 2000/0 | 20/13 | 50 | 100 | 150P | 200 | 250 | 300P | 350 | 400 | - | 4.4 |
| 302xx0-4xxxx4xx | 6 | 3000 | 3000 | 3000/2000 | 13.8/7 | 50 | 100P | 150 | 200P | 250 | 300 | 350 | 400P | - | 4 |
| 301xx0-xxxxx4xx | 3 | 4000 | 3000 | 4000/4000 | 7/4.5 | 50P | 100P | 150 | 200P | 250 | 300 | 350 | 400 | - | 5.5 |
| 30Kxx0-0xxxx4xx | 4 | 6000 | 3000 | 6000/n/a | 8.7/5.5 | - | - | 150 | 200 | 250 | 300 | 350 | 400 | - | 4.7 |
| 30Kxx0-1xxxx4xx | 4 | 6000 | 3000 | 6000/n/a | 8.7/5.5 | - | - | 150 | 200 | 250 | 300 | 350 | 400 | - | 4.7 |
| 30Kxxo-3xxxx4xx | 4 | 6000 | 3000 | 6000/n/a | 8.7/5.5 | - | 100 | 150 | 200 | 250 | 300 | 350 | 400 | - | 4.7 |

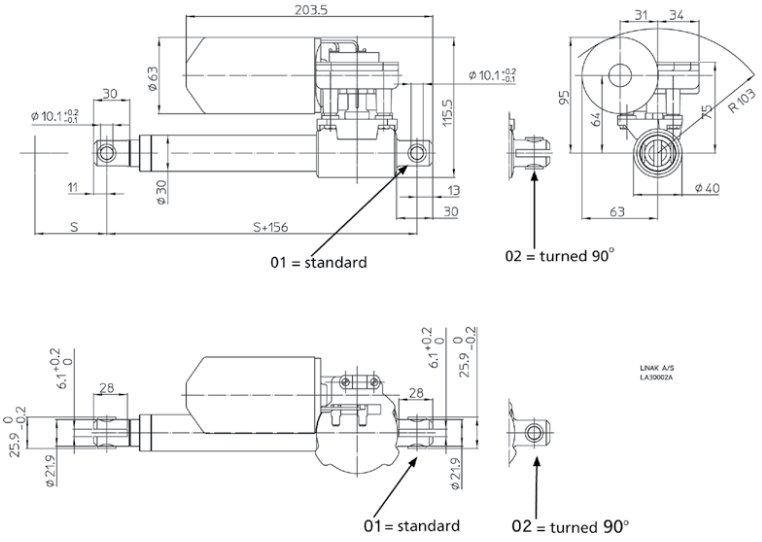
The above measurements are made with the actuators connected to a stable power supply, LA30 L-motor versions with a CB12.

S = Strong motor; L = Slow motor; K = Ball screw; KAS = Ball screw, safety nut; KSM = Ball screw, safety nut, spline.

**LA30**
**Ordering example:**



**Dimensions:**



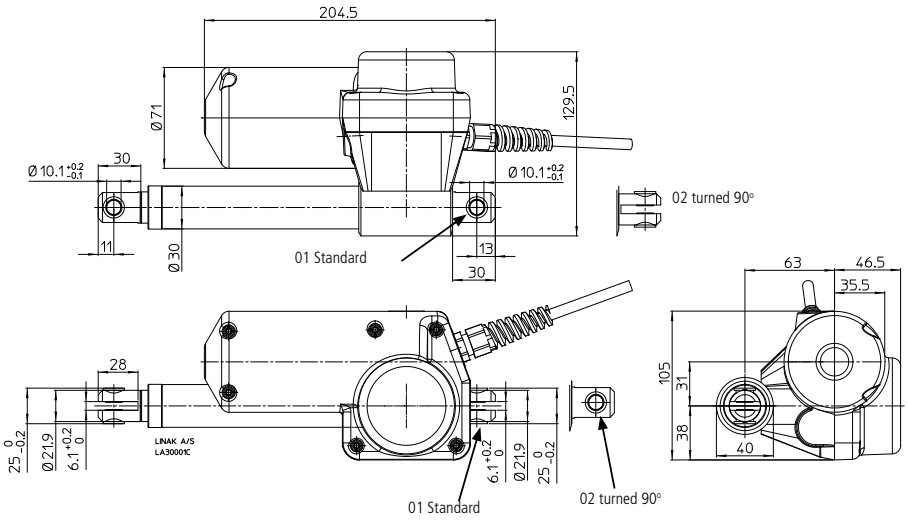

**Installation dimension:**

| | |
|---|---|
| LA30 L-motor + LA30 + LA30 S-motor with 3 or 6 mm pitch | S + 156 mm |
| LA30 with spline + LA30 S-motor with 9 mm pitch | S + 167 mm |
| LA30 with brake + LA30 S-motor with 3 or 6 mm pitch with brake | S + 189 mm |
| LA30 with ball screw and L-motor / LA30 ball screw and safety nut | S + 194 mm |
| LA30 L-motor with brake + LA30 S-motor 9 or 12 mm pitch with brake | S + 199 mm |
| LA30 L-motor with ball screw, safety nut and spline | S + 251 mm |

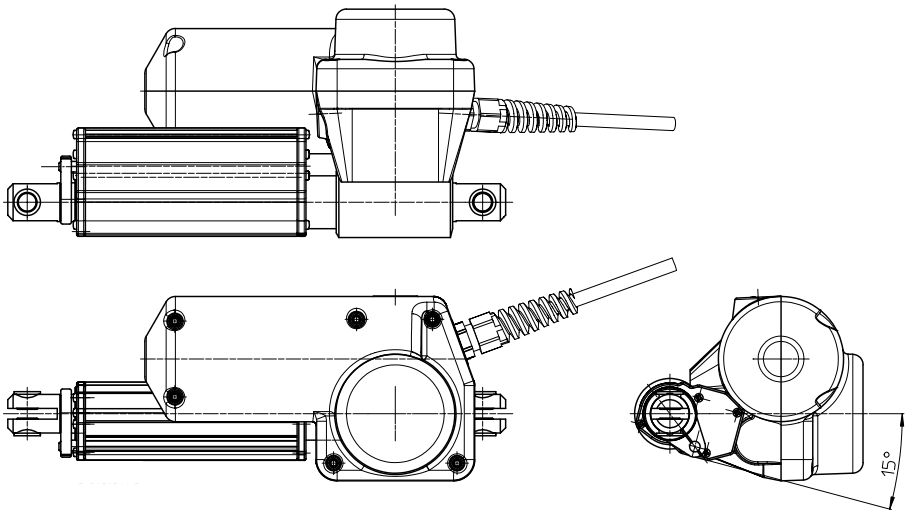
**Plastic housing for LA30:**
It is possible to order a LA30 with the protection class IP66. This option requires a plastic housing.
See drawings below.

The LA30 actuator is mounted with a 1.2 m / 0.5 m cable without plug.

**Dimensions:**

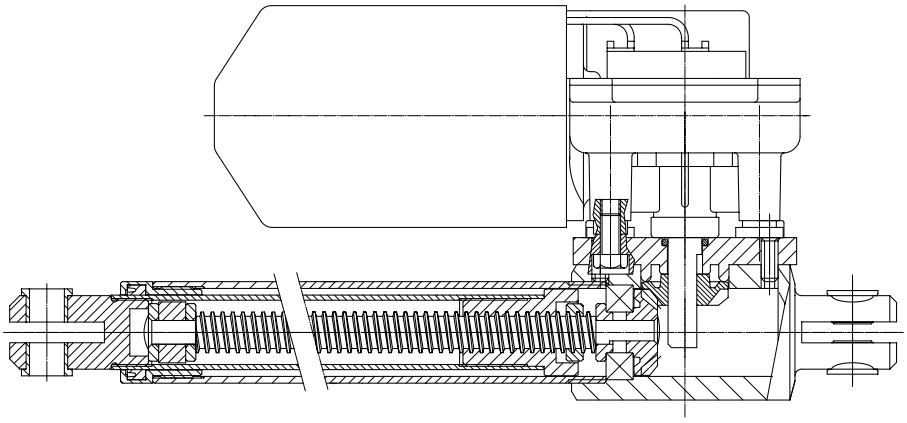



**LA30 with housing mounted with LSD (protection class IPX4):**



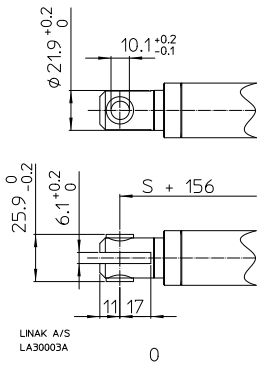

Drawing no.: LA30LSD001B

**LA30 Standard:**

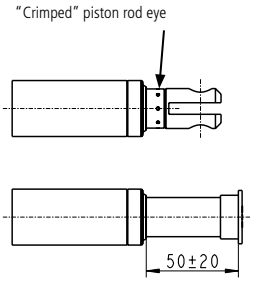


Drawing no.: LA30SNIT001B

**Piston Rod eyes:**



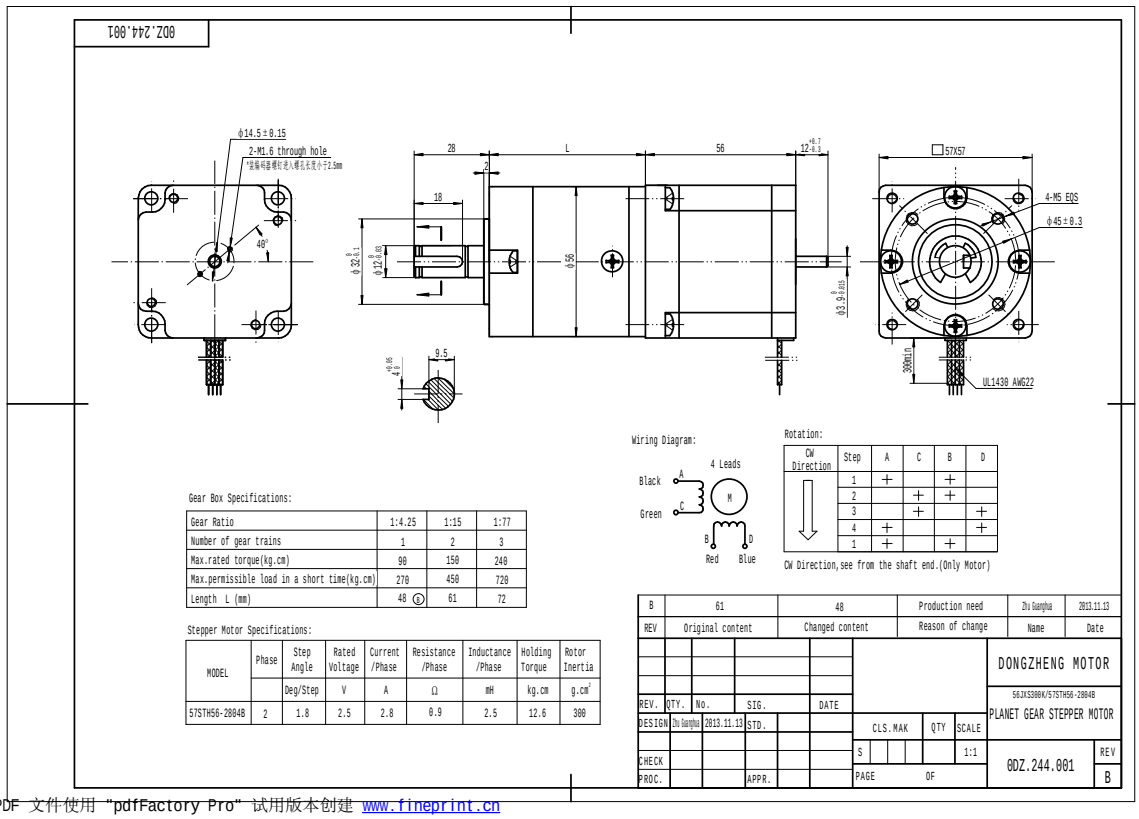

0
Standard




Piston rod without eye
(Piston rods without eyes are not tested)

## A.4 Steering actuator

A mechanical drawing. The technical specifications are found on their website ([1])



## A.5 Camera data sheet

**8MP High-Definition USB Camera Module USB2.0 SONY IMX179 Color CMOS Sensor 75Degree Lens**
**Main Specifications**

| | |
|---|---|
| Model | ELP-USB8MP02G-L75 |
| Sensor | SONY IMX179 |
| Lens Size | 1/3.2inch |
| Pixel Size | 1.4um |
| image area | 6.18mm x 5.85mm |
| Max. Resolution | 3264(H) X 2448(V) 8Megapixel |
| Compression format | MJPEG / YUV2（YUYV） |
| Resolution & frame | 3264X2448 @ 15fps / 2592X1944@ 20fps<br>2048X1536 @ 20fps / 1600X1200@ 20fps<br>1280X960 @ 20fps / 1024X768@ 30fps<br>800X600 @ 30fps / 640X480@ 30fps |
| S/N Ratio | 34dB |
| Dynamic Range | 72.5dB |
| Sensitivity | 0.65V/lux-sec@550nm |
| Mini illumination | 0.5lux |
| Shutter Type | Electronic rolling shutter / Frame exposure |
| Connecting Port type | USB2.0 High Speed |
| Free Drive Protocol | USB Video Class（UVC） |
| OTG protocol | USB2.0 OTG |
| AEC | Support |
| AEB | Support |
| AGC | Support |
| Button camera | Support |
| Adjustable parameters | Brightness, Contrast, Saturation, Hue, Sharpness, Gamma,<br>Gain, White balance, Backlight Contrast, Exposure |
| Lens Parameter | Size: 1/3.2, Iris: F3.5, Focus: 3.6mm , FOV: 75 Degree |
| LED board control interface | Support 2P-1.25mm socket |
| LED board power connector | Support 2P-2.0mm socket |
| Power supply | USB BUS POWER 4P-2.0mm socket |
| Operating Voltage | DC5V |
| Working current | 150mA~240mA |
| Working temperature | -10～70℃ |
| Storage temperature | -20～85℃ |
| Board size | 38X38mm（compatible 32X32mm） |
| Operating system request | WinXP/Vista/Win7/Win8<br>Linux with UVC（above linux-2.6.26）<br>MAC-OS X 10.4.8 or later |

| | |
|---|---|
| | Wince with UVC<br>Android 4.0 or above with UVC |

# B Vehicle Scrutineering sheet

In addition to the conventional Technical Inspection, detailled in Chapter 1 and Chapter 2 of the Official Rules, the organisers will proceed with a number of verifications on the autonomous systems and safety of vehicles.

Edit

## Technical inspection

| Autonomous system activation | Pass | Fail | Notes |
|---|---|---|---|
| Safety driver clearly understands activation of AS | | | |
| **External lighting** | | | |
| External lighting activates when AS on, not manually operated | | | |
| External lighting clearly visible | | | |
| **Brakes** | | | |
| Safety driver has complete control of brakes when AS engaged | | | |
| **Autonomous system cut off** | | | |
| Safety driver demonstrate AS cut off | | | |
| Cut off easily reachable by driver | | | |
| Cut off separate from main autonomous control switch | | | |
| Cut off isolates AS from vehicle control | | | |
| Cut off causes external lighting to go off | | | |
| **Emergency Shutdown (Driver's compartment)** | | | |
| Emergency shutdown stops all AS systems and propulsion | | | |
| Safety driver has control of steering and brakes after shutdown | | | |
| Shutdown causes external lighting to go off | | | |
| **Sensors** | | | |
| Sensors mounting secure and cannot move when vehicle in motion | | | |
| **AS Wiring** | | | |
| AS wiring tidy? | | | |
| AS wiring does not hinder movement of actuators | | | |
| Wiring not in reach of driver | | | |
| **Autonomous power supply** | | | |
| AS can be disengaged from power supply | | | |
| If AS uses dedicated battery: | | | |
| Is removable? | | | |
| See standard lithium battery checks | | | |
| **Onboard/autonomous computer** | | | |
| Securely mounted, cannot move | | | |
| Not in reach of driver | | | |
| **Dynamic testing** | | | |
| Demonstrate control of acceleration, steering and brakes | | | |
| Smooth and controlled extent of range | | | |
| Hold brake and gradually apply throttle, vehicle doesn't move | | | |
| **Wireless communication and control** | | | |
| All wireless control and communication components removed | | | |
| No microphones or controls in driver compartment | | | |
| **Actuators** | | | |
| Solenoids, servos and control actuators isolated from driver | | | |

Edit

## Notes