CHALMERS
UNIVERSITY OF TECHNOLOGY

Advanced vehicle control systems

A multi-vehicle experimental platform for automotive algorithms

Bachelor's thesis in Control and Mechatronics at the Department of Signals and Systems

Johan Ek
Johannes Gunnarsson
Viktor Hellaeus
Viktor Insgård
Mattias Kuosku
William Norrblom

Supervisor:
Sébastien Gros

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

SSYX02-15-36

Advanced vehicle control systems:

A multi-vehicle experimental platform for automotive algorithms

JOHAN EK
JOHANNES GUNNARSSON
VIKTOR HELLAEUS
VIKTOR INSGÅRD
MATTIAS KUOSKU
WILLIAM NORRBLOM

Technical report no SSYX02-15-36
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover:
A picture of the two RC-cars used in this project.

Photographer:
Fannie Runneberger, www.fannie.se
(Figures: 2, 3, 4, 9, 23a, 23b, 24a, 25 and cover)

Typeset in LaTeX
Göteborg, Sweden 2015

Abstract

This report presents the results of a project that is a further development of a multi-vehicle experimental platform (MVP) for automotive algorithms at Chalmers University of Technology. The project has resulted in an upgraded version of the platform. The aim of the project was to create a platform that enables testing of different race tracks and traffic situations with different automotive systems at high speeds.

The MVP can be run with different race tracks created by the user and different automotive algorithms executed by RC vehicles. The platform has been tested handling two Kyosho dNaNo RC cars at the same time but is prepared to handle up to four vehicles simultaneously. The platform is controlled and autonomously driven by a computer, which recognizes the vehicles by an IR light camera based recognition system.

The project group is satisfied with the results of the project and a lot of features have been upgraded compared to the previous platform. The car is now able to drive at 200 cm/s and still follow the reference track compared to the top speed of 30 cm/s in previous projects. The new camera operates at 150 FPS compared to the camera used in the previous project, which operated at 30 FPS. This improvement enables driving the cars at higher speeds. The recognition system change from a glyph system to an IR system has decreased the size of the markers with 98.6 %. The speed of the image processing meets the update frequency of the camera and executes in less than 4 ms, which make it possible to analyze each image from the camera.

Two automotive algorithms have been implemented in the system, one for collision avoidance and overtaking and one for platooning. The algorithms for collision avoidance and overtaking are able to detect and to overtake both stationary and moving obstacles. The algorithm is not complete and there are still scenarios that the system can not handle and fails with the overtaking. The algorithm for platooning is able to have a vehicle following a leader from a constant distance, but there is still a large uncorrected control error and there have not been any tests with more than two vehicles.

Even if the project has accomplished a lot there is still much room for improvement on the MVP in upcoming projects. But the members of the project group believes that the current platform is good starting point with great potential for future use. Besides implementing new automotive algorithms the project group recommends further development of today's automotive algorithms and more reliable speed measurement and position recognition.

Keywords: automotive algorithms, active automotive systems, experimental platform, control systems, RC-car, automation, vehicle, collision avoidance, overtaking, platooning.

Sammanfattning

Den här rapporten presenterar resultatet av ett projekt som är en vidareutveckling av en experimentell plattform för att köra flera fordon samtidigt på Chalmers tekniska högskola. Den möjliggör test av fordonståg och omkörningsmanövrar. Projektet har resulterat i en uppgraderad plattform kallad MVP – Multi Vehicle Platform – i den här rapporten. Målet med projektet var att skapa en plattform som gör det möjligt att testa olika bilbanor och trafiksituationer med olika aktiva bilsystem i höga hastigheter.

Plattformen kan köra radiostyrda bilar på olika racingbanor och med olika självkörande algoritmer skapade av användaren. Plattformen har testat att köra två bilar samtidigt men är förberedd på att hantera upp till fyra bilar samtidigt. Plattformen är reglerad och självkörd av en dator som känner igen bilarna med hjälp av ett kamera baserat IR-system.

Projektgruppen är nöjd med resultatet och utfallet av projektet. Många delar har utvecklats och uppgraderats jämfört med tidigare projekt. Bilen har nu möjlighet att köra upp till 200 cm/s och kan fortfarande följa efter referensbanan, jämfört med tidigare projekt då bilen endast klarade av att köra i 30 cm/s. Den nya kameran tar bilder i 150 Hz, jämfört med den gamla kamerans 30 Hz. Den här uppgraderingen möjliggör att köra fordonen i högre hastigheter. Ändringen från ett glyf-baserat till ett IR-baserat system har reducerat storleken på de nödvändiga markörerna med 98,6 %. Bildhanteringens exekveringstid på 4 ms är kortare än tiden mellan bilderna från kameran. Detta möjliggör att varje bild från kameran kan analyseras.

Två självkörande algoritmer är implementerade i systemet, en för kollisionsundvikning och omkörningar och en för fordonståg. Algoritmen för kollisionsundvikning och omkörningar kan upptäcka och köra om både stillastående hinder och hinder som rör på sig. Algoritmen är dock inte helt färdig utan det finns fortfarande många situationer som systemet inte klarar av. Fodonstågsalgoritmen klarar av att hålla ett fordon som ledare och ett annat fordon på ett konstant avstånd bakom ledaren. Avståndet mellan bilarna är fortfarande för stort, relativt mot referenssignalen, och systemet har inte testats med mer än två bilar.

Även om projekt har utvecklats mycket finns det fortfarande plats för ytterligare förbättringar av MVP:n i framtida projekt. Projektmedlemmarna ser stor potential för framtida projekt och tror att det är en bra startpunkt för fortsatt utveckling. Förutom att utveckla nya självgående algoritmer rekommenderar projektgruppen en vidareutveckling av befintliga självkörande algoritmer och pålitligare hastighetsmätning och positions uppskattning.

Nyckelord: självkörande algoritmer, aktiva självkörande system, experimentell plattform, reglersystem, radiostyrda bilar, fordon, kollisionsundvikning, omkörning, fordonståg.

Acknowledgements

This report describes a bachelor's thesis project carried out during the spring of 2015 at the department of Signals and Systems at Chalmers University of Technology. The project group consisted of Johan Ek, a student at the program for Engineering Physics, and Johannes Gunnarsson, Viktor Hellaeus, Viktor Insgård, Mattias Kuosku and William Norrblom, students at the program for Automation and Mechatronics. The project group would like to thank the following persons and departments for their support during the project:

Sébastien Gros, Assistant Professor in the Automatic control research group, Department of Signals and Systems at Chalmers University of Technology, for his cooperation and advice.

Nikolce Murgovski, Post doctoral researcher in the Mechatronics research group, Department of Signals and Systems at Chalmers University of Technology, for his feedback and support during the project.

Göran Stigler, Research Engineer at Chalmers University of Technology, for his advice regarding 3D printing.

Fannie Runneberger, Photographer, for taking photos to the project.

The department of Signals and Systems, Chalmers University of Technology for buying the expensive camera outside of the project budget.

Becky Bergman, University lecturer at the Division for Language and Communication at University of Gothenburg, for her feedback on the project reports.

The bachelor's thesis project group Smartlift - voice activated elevator for their feedback on the midterm report.

Pontus Bokinge, Chemical Engineering with Physics student at Chalmers University of Technology, for his feedback on the final report.

List of Abbreviations

Abbreviation	**Definition**
AZ	Activation Zone
c.g.	Centre of gravity
DAC	Digital to Analog Converter
DZ	Dead Zone
FPS	Frames Per Second
GRATF	Glyph Recognition and Tracking Framework
GPS	Global Positioning System
GUI	Graphical User Interface
I/O	Input/Output
IR	Infrared
LED	Light Emitting Diode
LP	Low Pass
LPS	Local Positioning System
MVP	Multi-Vehicle Platform
MPC	Model Predictive Control
PCB	Printed Circuit Board
PID	Proportional-Integral-Derivative
PWM	Pulse Width Modulation
QR	Quick Response
RC	Radio Controlled
RGB	Red, Green and Blue representation of a color
SARTRE	Safe Road Trains for the Environment
SPI	Serial Peripheral Interface
USB	Universal Serial Bus

Contents

1 Introduction

One of the biggest threats in the traffic is the human factor (Ljung, Fagerlind, Lövsund, & Sandin, 2007). If the human factor could be eliminated with the help of active systems that assist humans in the traffic, the number of accidents could be reduced. The recent advancements in sensor technology in the vehicle industry has led to a significant progress in active safety. Many believe that the future of personal transportation lies in autonomous driving. With autonomous vehicles the only thing humans have to decide is the destination. Control strategies such as platooning (a convoy of vehicles where a lead vehicle leads a number of closely following vehicles) and overtaking will be milestones in this area. These milestones will act as a bridging technology for a future where human driving might be unnecessary.

Platooning and road trains could be the future of automotive transportation. Funded by the European Commission, Volvo Cars was part of a big project about platooning called SARTRE (Safe Road Trains for the Environment). With the same hardware found in new Volvo cars (except a communication unit) the SARTRE project managed to let cars connect to a truck that controlled a road train by taking over the steering of the cars in the road train. Driving close to another vehicle saves fuel by reducing drag. In a report on fuel consumption when platooning by Davila, 2013 it is showed that the fuel consumption could be reduced between 7-16% depending on the type of vehicle used and distance between them. These results make it interesting to continue research in these areas.

Almost every new car nowadays has systems for collision avoidance such as forward collision warning and adaptive cruise control. Also, systems for automatic overtaking can be found in cars today like the Tesla Autopilot system in Tesla Motors car Model S (Musk, 2014). In order to develop these systems further, safe test environments could be helpful. Technical failures will not have significant consequences, concerning safety or economical aspects in these environments. One way to create a safe environment is to scale down the system and use RC vehicles instead. The solution of today's and tomorrow's automotive problems is not in RC vehicles controlled to follow a reference signal. But the algorithms developed during this project for collision avoidance, overtaking and platooning are of great interest for the automotive industry and could be a starting point for testing and verifying these kinds of algorithms in a smaller scale.

Chalmers University of Technology has developed a multi-vehicle experimental platform with RC vehicles that could be used to test different vehicle control methods. The platform includes a camera system that can locate and track the motion of vehicles travelling on a black painted surface. The first version of the platform was developed as a bachelor thesis in the spring of 2013. The platform was further developed during a eight week project in the autumn of 2013 by master students. At that moment the project's focus was positioning and control strategies for swarm-robots (multiple robots with cooperative behaviour). During spring of 2014 a bachelor thesis refined and added more functions to the swarm-bots. At this time the project used m3pi-bots (Pololu, 2012). The m3pi-bots were relatively big and slow robots (see the "Platform for Robot Swarm Behaviour" Hagebring, Hansson, and Waldemarsson, 2014 for further information). During a design project in autumn of 2014 a group of master students decided to change the vehicles used in the platform and make it more connected to reality by using more realistic cars (Sjödin, Sampath, Diakou, & Johansson, 2014). This was achieved with fast RC cars from Kyosho called dNaNo, 1:43 the size of a real car (see Section 2.1 for a further information). Compared to the m3pi-bots used in previous projects, which used tank steering, these cars are controlled by regulating the front wheel steering angle and rear wheel throttle. In this way more realistic car dynamics can be seen on the platform when implementing different controller methods.

1.1 Problem description

The previous project, Sjödin et al., 2014, focused on setting up a platform that could follow a trajectory and avoid a stationary object by overtaking it. This required a system capable of estimating several states of a vehicle on a race track. The system can be divided into several parts as seen in the picture below.

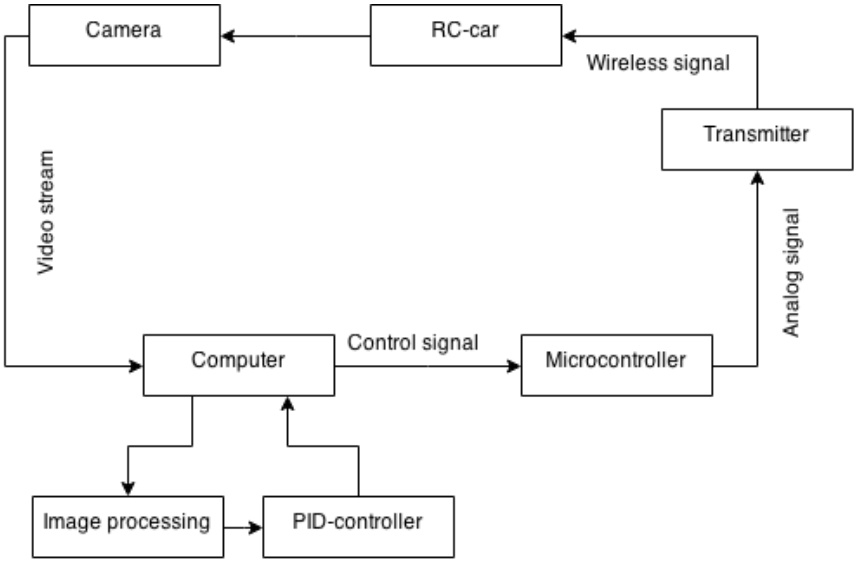


Figure 1: General description of the previous platform.

A camera provides a live stream over the race track. With real time image processing vehicles can be identified by the software of the platform, which estimates the vehicles' states. This information is passed to a controller that calculates control signals for the vehicles and sends these signals to the transmitters via a micro controller. The transmitters interpret the signals and sends them wirelessly to the RC vehicles, which actuate the signals. The platform constructed by Sjödin et al., 2014 was created with parts of the platform used in the swarm-bots project which did not have the dNaNo cars in mind. This resulted in a platform with several issues. The major problem was that the platform had a low update frequency due to hardware limitations in the camera. This limited the speed of the cars to around $30\,\mathrm{cm/s}$, in order to be detected properly by the system. The image processing used to detect a car was time consuming and had impact on the cars' performance because it required that each vehicle should be fitted with an $6 \times 7\,\mathrm{cm}^2$ piece of paper with a QR code. The communication between the micro controller and the transmitter was an issue as well. The transmitter required an analog signal and the micro controller used did not have any analog outputs. Instead PWM signals with second order passive LP filters were used to get a continuous signal. The mapping between intended input voltage and received input voltage was done manually, which was time consuming, not accurate and resulted in uncertainties in the control loop. Because the previous projects that developed most of the software to the platform did not have other hardware in mind than the one used at the time, the program structure was not so modular. Many functions were implemented in a way that made them hard to distinguish from each other.

1.2 Purpose and goals

The purpose of this project is to create an improved multi-vehicle experimental platform that can be used in later projects. The focus is on increasing the speed while following a track and on having a modular structure of the software.

The goals of the project are divided into three parts, where the first one is the most important. The algorithms for platooning, collision avoidance and overtaking are unusable if there is no platform to implement them on.

Platform

- The platform should be able to control RC cars to follow a trajectory at 200 cm/s.
- The positioning system should have no effects on the performance of the vehicles.
- The communication between the computer and transmitter should be reliable and accurate.
- The code should be easy to develop further and new automotive algorithms should be easy to implement.

Platooning

- The platform should have a functioning system for platooning with at least two vehicles.
- The distance between the vehicles should be at an average of 13 cm.

Collision avoidance and overtaking

- The platform should have a functioning system for collision avoidance and overtaking.
- The vehicles on the platform should be able to detect objects in its path and make maneuvers to avoid a collision.

1.3 Delimitations

The platform will only use PID-controllers when control theory is applicable. By doing this more time can be spent on optimizing the inner workings of the platform instead of optimizing the control methods, which will result in better code. Since the purpose of the project is to achieve a modular structure it is wise to limit the different automotive algorithms, which could be time consuming to develop and resulting in unstructured code. The project will use two Kyosho dNaNo RC cars to test and verify the functions of the platform. The platform will function as intended when not exposed to direct sunlight. The location where the platform should be built and stored has capabilities to be shielded from sunlight therefore it is no need to put effort on the problem that could occur if the platform is exposed to sunlight.

1.4 Outline of the report

The report is divided into eight sections. After the introduction the second section explains the technical details of the hardware used in the project. It also briefly explains how the platform is intended to work. The hardware presented here is the one that the platform is using at the time this report is written.

In Section 3 theory about the car models used in the simulations of the platform are presented. It consists of two sections where the first gives a kinematic model of a vehicle's movement in a plane and then a dynamic model that takes the vehicle's physical properties into account.

In Section 4 the different tools used to build the platform in the project are presented. A brief summary of the tools is attached together with an explanation why and where they are used.

Section 5 is divided into five parts, which cover the whole working process from the beginning to the end. At first the preparations of the project is explained in detail, after this the assembling of the platform is covered. This is followed by a section regarding how the algorithms for following a race track is done. Then the simulation and development of the control strategies collision avoidance, overtaking and platooning is explained. At last a section on the user interface of the platform is presented.

In Section 6 the results of the platform is presented. It follows the structure in section five and presents the relevant results for each section in the same order. The results are compared to the goals set at the start of the project.

In Section 7 a discussion about the results and the overall outcome of the project are presented along with recommendations on improvements for later projects. This section also includes a section about the future of automotive systems and how it could affect the environment.

In Section 8 a conclusion for this report is presented.

2 System description

The MVP enables high speed real world autonomous driving with RC cars in the size of 1:43 to a real world car. In this section the most important hardware components of the platform are introduced and described to easily overview the system.

Table 1: The key hardware components used in the platform.

Component	**Model**
RC car	1:43 scale Kyosho dNaNo
Transmitter	Kyosho Perfex KT-18
Microcontroller	Arduino Uno
Digtal to analog converter	Texas Instruments TLC5620CN
Camera	Point Grey Flea3 1.3 MP Mono USB3 Vision
Lens	GOYO GM24514MCN
IR lamp	ILS 4 IR LEDs 850nm
IR markers (tape)	ifm electronic Reflective Tape
IR filter	760 nm, High pass filter
Computer	PC with Intel i7 running at 1.9 GHz, 10 GB RAM and USB 3.0

The motivations for the choice of hardware are found in Section 5 and a full list of all hardware used in the project can be found in Appendix A.

As stated in the introduction the platform needs to be able to estimate the states of the vehicles in the race area. This is done similarly to how it was done in the previous platform. Some components have been changed but the general concept is the same. In Figure 2 the connections between the hardware are shown and how they communicate with each other.

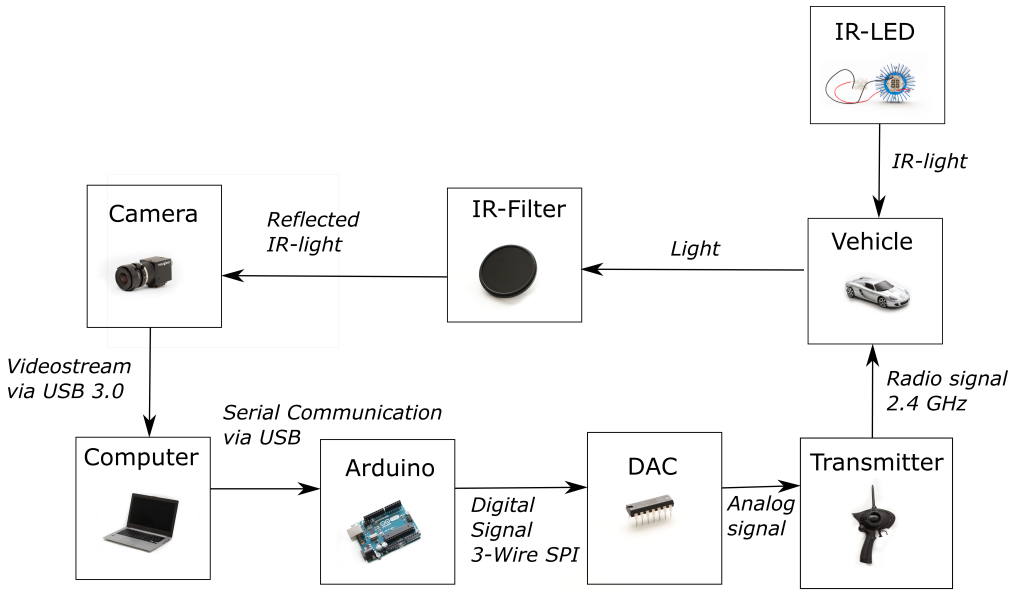


Figure 2: Overview of the platform.

2.1 RC car and transmitter

The vehicles used in this project are two Kyosho dNaNo Porsche Carrera GT and are from a series with 1:43 scaled RC cars. The top speed is around 500 cm/s and the cars have a steering angle around 30 ° from the middle to an outermost position. Each car is 10.65 cm long and 4.45 cm wide. The dNaNo is rear wheel driven and is steered by the front wheels. It carries a one cell LiPo battery that delivers about 30 min run time on one charge. At the front is a small servo motor which sets the steering angle and at the back a PWM controlled DC motor is located in a H-bridge. It can be controlled by a Perfex KT-18 transmitter operating at the 2.4 GHz frequency band. It allows up to 40 cars to be used at the same place by splitting up the frequency band if multiple transmitters are used simultaneously. The unmodified transmitter receives analog signals in the range of 0-3 V, which is controlled with two separate potentiometers in the transmitter. These signals are converted and sent via radio waves to the car that interprets how to behave depending on the voltages. The transmitter also has functions that allows for precision adjustments of the car it is controlling which allows to compensate for any differences between the neutral value (i.e. zero throttle or steering gain) at the transmitter and the car. It can also reverse the signals meaning that full throttle can be at either zero or three volt. In this project the transmitter's potentiometers have been bypassed. Instead the analog signals are being directly sent to the transmitter's circuit board from a digital to analog converter. In Figure 3 the RC car and transmitter are shown.

Figure 3: The dNaNo Porsche Carrera GT and Perfex KT-18 transmitter

2.2 Microcontroller and digital to analog converter

The mircocontroller used in the platform is an Arduino Uno. It is equipped with an Atmel ATmega328 running at a clock speed of 16 MHz, which allows for fairly fast serial communication. By using the Arduino a lot of open source libraries for communication between different devices can be used. It is also easy and fast to reprogram the ATmega, which is helpful when testing and debugging implementations. In the platform it is located between the computer running the core components of the platform software and a digital to analog converter (DAC) as described in Section 5.2.2. The DAC is a Texas Instruments TLC5620CN which is an 8 bit DAC communicating with three wire Serial Peripheral Interface (SPI) at a maximum clock speed of 1 MHz.

2.3 Camera, lens, IR lamp and race area

The vehicle position recognition is done with the help of a camera and an IR system. An IR lamp is mounted together with the camera emitting IR light that is reflected by markers on the vehicles. The camera is a monochrome camera from the company Point Grey called Flea3 (FL3-U3-13Y3M-C) PTGrey, 2014. It shoots in 5:4 formats with a maximum resolution of 1280×1024 pixels. On the camera a c-mount lens from Goyo is fitted with a focus length of 4.5 mm and an angle of view of $79\,^\circ \times 59.4\,^\circ$. The camera and the lens are shown in Figure 4. On the lens there are two knobs for setting the aperture and focus. In front of the lens is an IR filter which cuts off light with lower wavelength than 760 nm. The IR lamp used is an OSRAM 4 PowerStar IR that has four high output IR LEDs with a peak wavelength of 850 nm. Its angle of half intensity is $\pm 45\,^\circ$. By using IR light instead of visible light the image processing can be simplified because there will be less disturbances from the surroundings as long as there is no IR rich light available. The race area is a black rubber carpet on the floor of size $2.8 \times 3\,\mathrm{m}^2$. The tests could take place directly on the floor, but this would expose the vehicles to more dust, which could jam their gears. The dust would also decrease the overall performance of the vehicles by decreasing the friction between the wheels and the surface.

Figure 4: The camera fitted with lens.

3 Theory of a mathematical 2-point model of a vehicle

The subject presented in this section is a theoretical base for how a vehicle can be represented as a 2-point model. The 2-point model used in this project is developed by Rajamani, 2012 and it is presented in two parts. The first part is a kinematic model of the vehicle, which does not take forces into consideration and the second part is a dynamic model based on the kinematic model.

3.1 Kinematic model of a vehicle's lateral motion

The model is implemented according to a kinematic bicycle model of a vehicle created by Rajamani, 2012 and is visualized in Figure 5. It is a mathematical model of a vehicle where the two front wheels are represented with a single wheel located in the front of the vehicle in the centre of its lateral body. The rear wheels are represented with one wheel using the same methodology. The angle between the vehicle's front wheel compared to the vehicle's centre line is called δ_f. The angle between the rear wheel compared to the vehicle's centre line is in this case equal to zero, because the rear wheels of the vehicles used are fixed and do not steer.

The vehicle's centre of gravity (c.g.) is in the point C. The longitudinal distance from C to the rear wheel is called ℓ_r and the distance from C to the front wheel is called ℓ_f. Since the experiments will take place on a floor the vehicle is assumed to have a planar motion. The coordinates of the vehicle's c.g. are X and Y. The vehicle's heading angle Ψ describes the orientation of the vehicle compared to the x-axis.

The vehicle's body slip angle β is the angle of the velocity vector V compared to the centre line. For small vehicles the slip angle can be assumed to be zero at lower speeds, i.e., speeds lower than 500 cm/s. The instantaneous rolling centre point of the vehicle is O and it is located at the radius R from the vehicles c.g.

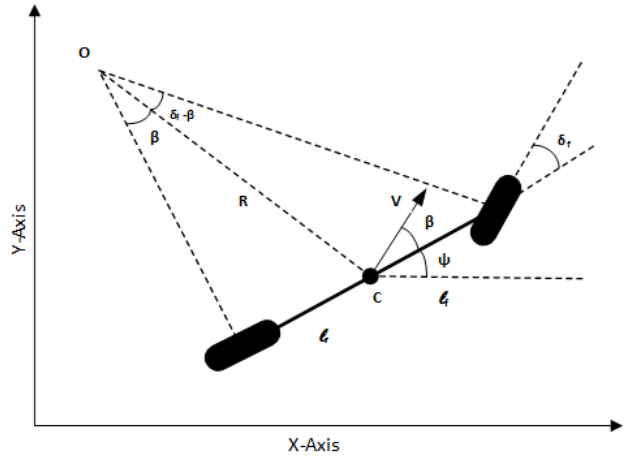


Figure 5: 2-point model used for a vehicle.

Due to the low velocity of the vehicle the radius R is assumed to change slowly. The angular velocity of the vehicle can then be calculated as the velocity V divided by the radius R which is described by

$$\dot{\Psi} = \frac{V}{R}. \tag{1}$$

Geometrical connections in Figure 5 result in four relations:

$$\dot{X} = V\cos(\Psi + \beta), \tag{2}$$

$$\dot{Y} = V\sin(\Psi + \beta), \tag{3}$$

$$\dot{\Psi} = \frac{V\cos(\beta)}{\ell_\mathrm{f} + \ell_\mathrm{r}}\tan(\delta_\mathrm{f}), \tag{4}$$

$$\beta = \tan^{-1}\left(\frac{\ell_\mathrm{r}\tan(\delta_\mathrm{f})}{\ell_\mathrm{f} + \ell_\mathrm{r}}\right). \tag{5}$$

3.2 Dynamic model of a vehicle's lateral motion

A state model of the vehicle is implemented in order to create a simulation where the controller parameters of a vehicle can be calculated. The dynamic model of the vehicle has two degrees of freedom, the lateral position y and the yaw angle Ψ, visualized in Figure 6.

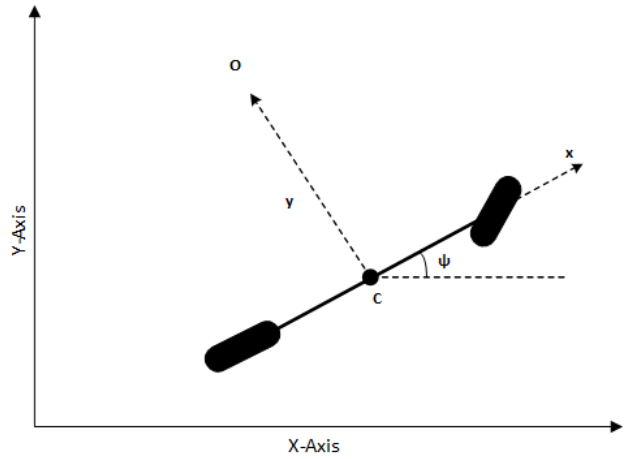


Figure 6: A vehicle's position parameters.

The first matter to take into account is the force of the vehicle. This is calculated according to Newton's second law of motion, being used along the y-axis as

$$ma_\mathrm{y} = F_\mathrm{yr} + F_\mathrm{yf} \tag{6}$$

with the forces F_yr and F_yf affecting each wheel.

The terms contributing to the vehicle's inertial acceleration along the y-axis a_y is calculated as the acceleration along the y-axis $\ddot{y}$ and the centripetal acceleration $V_x\dot{\Psi}$ as

$$m(\ddot{y} + V_\mathrm{x}\dot{\Psi}) = F_\mathrm{yr} + F_\mathrm{yf}. \tag{7}$$

To determine the yaw dynamics of the vehicle the moment balance with respect to the y-axis is calculated. The forces affecting the moment balance is the positive forces affecting the front wheel and the negative forces affecting the rear wheel. This is described as

$$I_z\ddot{\Psi} = \ell_\mathrm{f} F_\mathrm{yf} - \ell_\mathrm{r} F_\mathrm{yr} \tag{8}$$

where $I_z\ddot{\Psi}$ is the angular momentum, ℓ_f and ℓ_r is the length between the vehicle's c.g. and its front and rear wheel respectively.

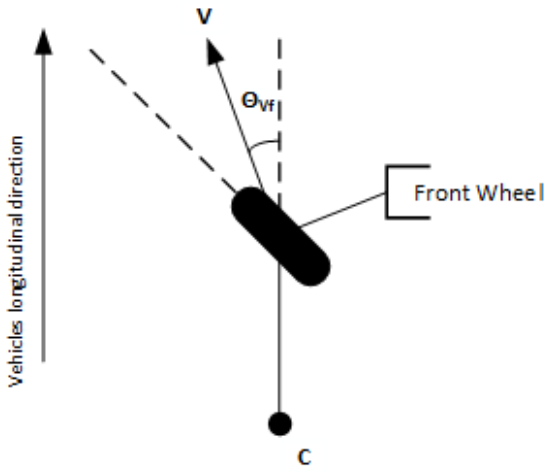


Figure 7: Slip angle of the front tire of the 2-point model.

Figure 7 describes the front wheel of the model in Figure 5. The angle between the front wheel's velocity vector V to the vehicle's longitudinal body is called θ_Vf. The slip angle α_f of the front wheel can be calculated as the steering angle δ_f subtracted with the wheels velocity vector angle θ_Vf. The slip angle α_r of the rear wheel can be calculated using the same formula. Resulting in

$$\alpha_\mathrm{f} = \delta_\mathrm{f} - \theta_\mathrm{Vf}, \tag{9}$$

$$\alpha_\mathrm{r} = -\theta_\mathrm{Vr} \tag{10}$$

where δ_r is equal to zero in this case.

According to Rajamani, 2012 experimental results have shown that the lateral tire force F_y is proportional to the slip angle α, when working with small slip angles. Multiplying the slip angle α of a wheel with the cornering stiffness C_α, which is the proportionality constant of each tire, result in a lateral tire force F_y. This force affects each wheel and since there are two front and rear wheels respectively, this force has to be multiplied by two resulting in

$$F_{\mathrm{yf}} = 2C_{\alpha\mathrm{f}}(\delta_{\mathrm{f}} - \theta_{\mathrm{Vf}}), \tag{11}$$

$$F_{\mathrm{yr}} = 2C_{\alpha\mathrm{r}}(-\theta_{\mathrm{Vr}}). \tag{12}$$

The wheels velocity angle θ_V can be calculated as the vehicle's inertial velocity along the y-axis divided by the vehicles velocity along the x-axis V_x. Small angle approximations can be used to eliminate tangent terms, which results in

$$\theta_{\mathrm{Vf}} = \frac{\dot{y} + \ell_{\mathrm{f}}\dot{\Psi}}{V_{\mathrm{x}}}, \tag{13}$$

$$\theta_{\mathrm{Vr}} = \frac{\dot{y} - \ell_{\mathrm{r}}\dot{\Psi}}{V_{\mathrm{x}}}, \tag{14}$$

when the elements contributing to the inertial velocity along the y-axis is the the velocity $\dot{y}$ and the angular velocity $\ell_{\mathrm{f}}\dot{\Psi}$.

The acceleration along the y-axis $\ddot{y}$ of the vehicle and the change of the yaw angle $\dot{\Psi}$ is used to create a state model of the vehicle. These values can be calculated by the relationships of the vehicle. The functions are calculated as

$$\ddot{y} = -\frac{2C_{\alpha\mathrm{f}} + 2C_{\alpha\mathrm{r}}}{mV_{\mathrm{x}}}\dot{y} - (V_{\mathrm{x}} + \frac{2C_{\alpha\mathrm{f}}\ell_f + 2C_{\alpha\mathrm{r}}\ell_{\mathrm{r}}}{mV_{\mathrm{x}}})\dot{\Psi} + \frac{2C_{\alpha\mathrm{f}}}{m}\delta_{\mathrm{f}}, \tag{15}$$

$$\ddot{\Psi} = -\frac{2\ell_{\mathrm{f}}C_{\alpha\mathrm{f}} + 2\ell_{\mathrm{r}}C_{\alpha\mathrm{r}}}{V_{\mathrm{x}}I_{\mathrm{z}}}\dot{y} - \frac{2\ell_{\mathrm{f}}^2C_{\alpha\mathrm{f}} + 2\ell_{\mathrm{r}}^2C_{\alpha\mathrm{r}}}{V_{\mathrm{x}}I_{\mathrm{z}}}\dot{\Psi} + \frac{2\ell_{\mathrm{f}}C_{\alpha\mathrm{f}}}{I_{\mathrm{z}}}\delta_{\mathrm{f}}, \tag{16}$$

which lead to the state model

$$\begin{bmatrix}\dot{y}\\ \ddot{y}\\ \dot{\psi}\\ \ddot{\psi}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & -\frac{2C_{\mathrm{af}}+2C_{\alpha\mathrm{r}}}{mV_{\mathrm{x}}} & 0 & -V_{\mathrm{x}} - \frac{2C_{\alpha\mathrm{f}}\ell_{\mathrm{f}}+2C_{\alpha\mathrm{r}}\ell_{\mathrm{r}}}{mV_{\mathrm{x}}}\\ 0 & 0 & 0 & 1\\ 0 & -\frac{2C_{\mathrm{af}}\ell_{\mathrm{f}}+2C_{\alpha\mathrm{r}}\ell_{\mathrm{r}}}{I_{\mathrm{z}}V_{\mathrm{x}}} & 0 & -\frac{2C_{\alpha\mathrm{f}}\ell_{\mathrm{f}}^2+2C_{\alpha\mathrm{r}}\ell_{\mathrm{r}}^2}{I_{\mathrm{z}}V_{\mathrm{x}}}\end{bmatrix}\begin{bmatrix}y\\ \dot{y}\\ \psi\\ \dot{\psi}\end{bmatrix} + \begin{bmatrix}0\\ \frac{2C_{\alpha\mathrm{f}}}{m}\\ 0\\ \frac{2C_{\alpha\mathrm{f}}\ell_{\mathrm{f}}}{I_{\mathrm{z}}}\end{bmatrix}\delta_f. \tag{17}$$

4 Method

An important part of developing the platform has been the use of different software tools. In this section these tools are introduced. The project management is also presented as it has been an essential part of developing the platform.

4.1 Project management

Due to the many subject areas present in this project, varying from dynamical models for the cars to construction of physical components to the platform, project management has been important. Initially, the recent reports about the platform along with reports and literature about related subjects were read to get a greater understanding of the topic and possible tweaks on the existing platform. After reading these reports some problems were identified in the current platform.

Function analysis has been used to take a step back and identify the problems that needs to be solved. Brainstorming was used to get different solutions to the problems. With a requirement specification the solutions to the problems could be examined and evaluated in a general way.

A Gantt chart has been used to dispose the parts of the project to the project members and set deadlines. Using a Gantt chart makes it easy to see which process of the project has to be completed before the project can proceed. If one part is delayed more focus can be put to that part in order to meet the deadlines. The Windows based program *Project 2013* was used to create the Gantt chart and schedule the events of the project.

Common files such as data sheets, text documents and log files has been shared through Google Drive. The code of the project was shared through the web based repository service Github, which is an easy way to work with the code in the project without interfering with each other's parts.

4.2 Softwares

The project has used a set of software and programming libraries. A short description of each program and library is presented below.

Simulation

MATLAB
MATLAB is a tool for calculating large quantities of data, presenting graphs and results of simulations. It has been used to solve mathematical problems and visualize results of simulations during the project.

Simulink
Simulink is a drag and drop program to simulate solutions of problems. Simulink has been used to simulate, verify and present graphical results of the control strategies. It provides a clear graphical overview of the dynamic models and feedback loops in the platform.

Software and programming language

C#
The previous project used the programming language C# (Sjödin et al., 2014). To save time and effort the project group choose to use the same programming language. The old code could therefore be used as a base for the upgraded program.

Visual Studio

The program is written in Visual Studio. Visual Studio is an integrated development environment (IDE) from Microsoft. The project group choose Visual Studio's developing environment because it contains powerful tools for sharing code through Github as well as profiling. Profiling can be use to analyze the execution time of different program parts, to help optimizing a program.

Arduino programming language

In order to program the Arduino Uno the Arduino IDE has been used. The Arduino programs are written in C and uses the Wiring framework to make the I/O operations easier.

Software libraries

FlyCapture SDK

FlyCapture SDK is a library for the Flea camera from PTGrey, 2015. It enables the C# code to acquire images from the camera.

AForge.Net

AForge is an open source library with useful methods and classes in the fields of Artificial Intelligence and Computer Vision. The project has used the AForge libraries to make the image processing, especially the class `BlobCounter`.

Product developing tools

CATIA

CATIA is a computer aided designing tool for making sketches, drawings and 3-D models in the computer. CATIA has been used to make drawings of the LED mount adapter on the lens and the unit that holds the communication devices. The drawing has been extracted to STL files and then 3-D printed.

CS EAGLE

CS-EAGLE is a computer aided program to draw printed circuit boards (PCB) for manufacturing. It has been used to construct the PCB, which simplifies the connection to the DAC:s.

5 Project execution

This section explains the implementation and development of the platform. It is divided into five parts starting with the preparations to upgrade the platform. This is followed by the sections on how the upgrade was made. After this, the development of the automotive algorithms are presented and lastly there is a section about how the user interface on the platform was made.

5.1 Preparations and decisions for the upgraded platform

The project started with research on the subject and similar projects. Thereafter a function analysis was done as seen in Figure 8. The starting point for the function analysis was how to follow a trajectory.

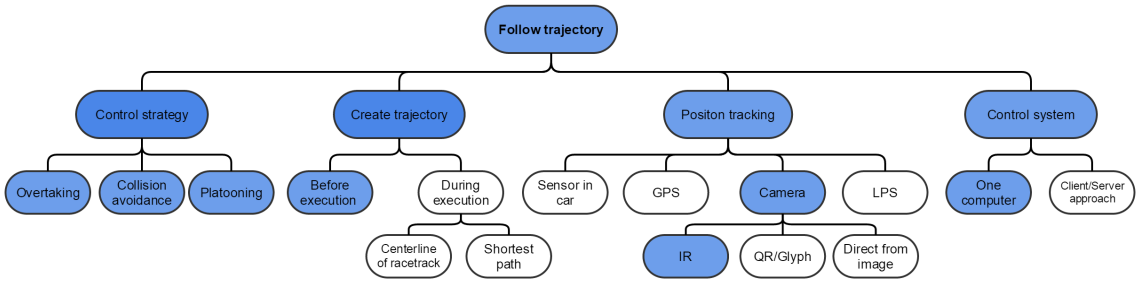


Figure 8: Flowchart over possible solutions to have a functioning platform. The selected solutions are showed in blue.

The function analysis together with the findings in the literature review formed the basis of the approach to the different sub problems that should be solved. In the literature review two projects were found that had developed similar platforms. The ORCA project at Eidgenössische Technische Hochschule (ETH), Zürich ORCA, 2015 and CARS at Uppsala University Nilsson, Björsell, Öhlund, Eriksson, and Bäck, 2014. These projects have showed results that fits the goals of this project. Another factor that was considered when choosing the approach was the current state of the platform. To be able to use preceding components in the platform only moderate changes could be made on the basic principles of the platform. When the solutions to the problems were selected it was clear that the platform would need some new hardware.

The project was divided into two parts. Part one consisted in upgrading the existing platform to a platform that enables vehicles to follow a racetrack at higher speed than before. Part two consisted in implementing automotive algorithms to the upgraded system.

5.2 Upgrading the platform

The requirement of the physical platform is that it enables driving vehicles at high speed while following a trajectory. It should also be able to handle different racetracks, i.e., different trajectories. Lastly the platform needs to be able to use different automotive algorithms for collision avoidance, overtaking and platooning with no tweaks to the hardware when changing the automotive algorithms. The frame rate of the camera may set the limit of the platform, regarding possible velocities of the vehicle and is one of the updated hardware components in the platform. The camera, a Flea3 from Point Grey, uses the USB3 Vision interface. This allows a high bandwidth to utilize the high frame rate of the camera. The maximum frame rate of the camera is 150 FPS and it has a good quantum efficiency on higher wavelengths, compared to many other cameras, which is needed to register the IR light that will be reflected from the vehicles.

The previous platform, Sjödin et al., 2014, used a system to recognize position, identification and direction called *Glyph Recognition And Tracking Framework* (GRATF). The idea was to have a black and white pattern on paper sheets. The size of the paper sheet was bigger than the actual car. A glyph is shown in Figure 9.

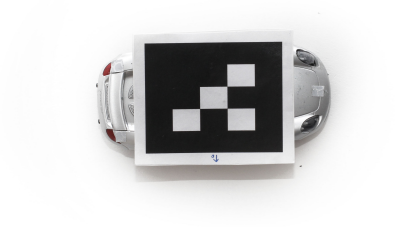


Figure 9: A glyph pattern used by GRATF.

The previous recognition system with GRATF effected both the design and performance of the car. Its execution time was slow and the system was not accurate enough when the vehicles were travelling at speeds greater than 30 cm/s. The GRATF system was replaced with an IR light system. The IR light system is based on an IR lamp that illuminates the race area. The vehicles are equipped with pieces of reflective tape which reflects the light to the camera. In order for the computer to recognize the vehicles, each vehicle is marked with a triangle pattern with the tape, one dot in the front and two in the back. Using these triangles the image processing can estimate both the position and orientation of the vehicle. It is important that the IR lamp is mounted near the camera to get decent amount of reflected light to the camera. With the help of an IR filter that cuts off light with lower wavelengths than 760 nm, an IR lamp and IR reflective tape there are good preconditions for accurate position tracking. The tape has the advantage of reflecting light to where it originates from, resulting in a better robustness from disturbing light.

Since the IR filter is of a standard 67 mm size and the lens is smaller, an adapter was made. The part was constructed in CATIA and 3D printed at Chalmers Prototype Laboratory. A 3D picture of the model is shown in Figure 10 and the drawing can be found in Appendix C.

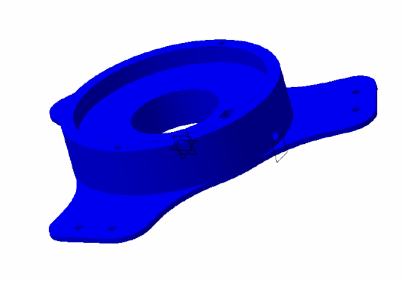


Figure 10: 3D view of the adapter.

The previous platform, Sjödin et al., 2014, used a table with a black painted fiberboard with a size of $2200 \times 1200\,\mathrm{mm}^2$ as the race area. To allow bigger and more complex race tracks it was replaced by a rubber carpet laying on the floor. The size of the new race area is $3973 \times 2749\,\mathrm{mm}^2$ and each pixel of the camera corresponds to $2.6 \times 2.6\,\mathrm{mm}^2$. The measurement of the pixels was made with a stick that was 1 m long with markers attached to the ends. Figure 11 shows the differences between the camera's field of view, depending whether the table or the floor is used.

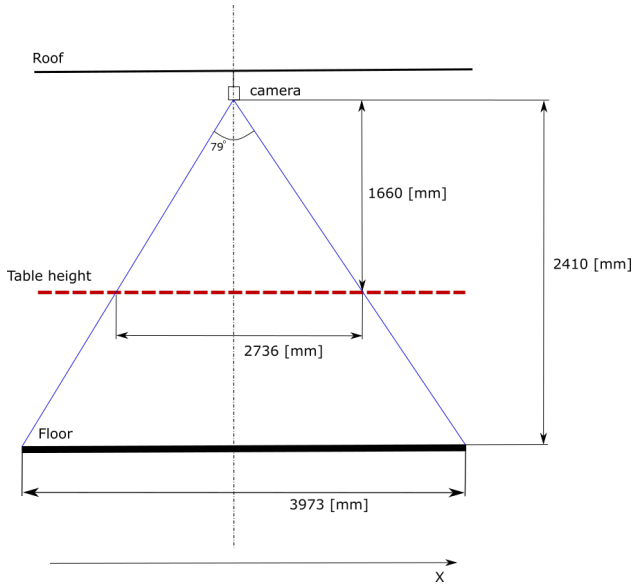
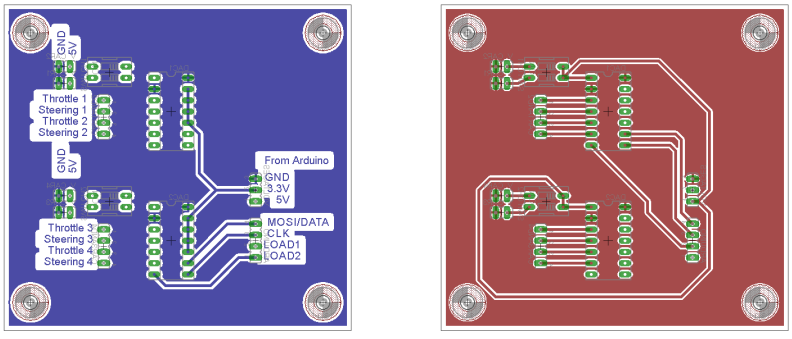


Figure 11: An overview of how the race area will increase with an increasing distance to the camera.

To be able to handle different race tracks and traffic situations the race track is only virtual, created and stored in the computer. By having a virtual race track it only takes seconds to change it, compared to changing between stationary real world race tracks. One drawback of this approach is that the track cannot be seen on the race area.

In order to make the setup of the platform easier and more reliable a PCB to simplify connections between Arduino and computer was made. The PCB has space for two integrated circuits (IC:s) containing DACs, which makes it possible to connect four vehicles to the platform. It also has power buttons for the transmitters. The resulting PCB is showed in Figure 12. For schematics see Appendix B.



(a) The top of the PCB. *(b) The bottom of the PCB.*

Figure 12: The PCB.

To have a single unit to work with a circuit board mount was constructed. The mount contains holes for the Arduino and the new PCB that lays flat on the mount. The circuit boards for the modified transmitters are mounted vertically in order to allow the buttons on them to be used. The mount has eight vertical stands, two for each transmitter. This allows four transmitters to be connected to the platform. The mount is visualized in Figure 13 and drawings can be found in Appendix C. The mount was designed in CATIA and extracted as a STL file. The part was 3D printed at Chalmers Prototype Laboratory.

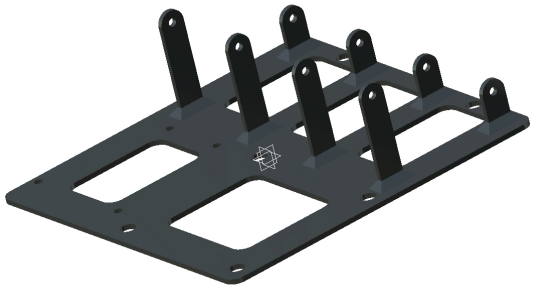
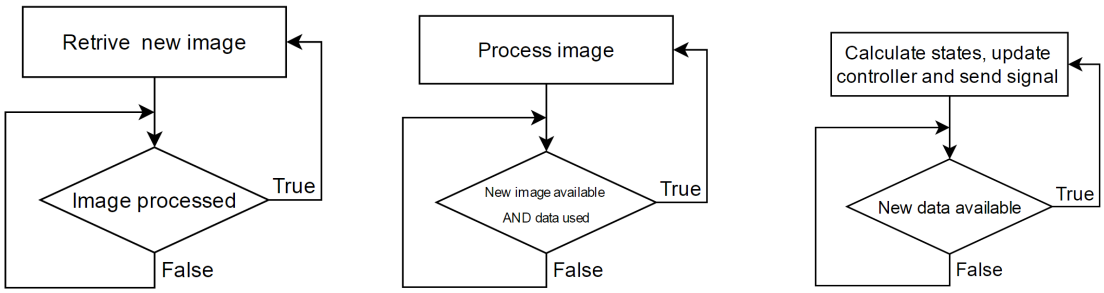


Figure 13: 3D view of the circuit board mount.

5.2.1 Main program

The main program is written in C# and is the part of the platform where all calculations are made. It also receives the video stream from the camera and sends output based on the calculated values to the Arduino. Because one goal of the platform is to achieve high speed it is important that the main program executes fast in order to make the lag between input and output as small as possible. To be able to do this the program is divided into three different main parts, each with its own thread. These parts are the camera thread, image processing thread and the controller thread. By having these threads the execution time will be lower because it allows for parallel computing. As described in Section 1.1 a video stream is sent from the camera to the computer. Each of these images needs to be processed to get information about the current states of the vehicles in the platform. When this is done calculations can be done based on the information from the image processing. These calculations end up with a control signal that is passed to the Arduino.

Timing is important in the main program. Each image from the camera should only be processed once and the platform should only update the state of the vehicles once between each image. Otherwise the platform's performance will be reduced and the state update could give inaccurate results. For example, if one image is processed twice the position of a vehicle between two state updates will not be changed resulting in a zero speed reading when the vehicle may not be standing still. To make sure that this does not happen the threads always wait until the other threads has signaled that they have used the current data before a new image is captured or a new calculation is made, as seen in Figure 14.



(a) The camera thread. *(b) The image processing thread.* *(c) The controller thread.*

Figure 14: The three main threads.

There is also a graphical user interface (GUI) thread in the program. This is used to control the platform and has the functionality to make a user able to start, stop and set different settings on the platform. It can also plot and collect different data, such as the vehicles' speed and reference signals, which can be used to examine the performance of the platform.

Another goal of the platform was to achieve a modular structure to help further development and to simplify changes in the main program. Because of this the program is divided into classes where each class represents a piece of the program. A simplified overview of the classes and how they communicate is shown in Figure 15.

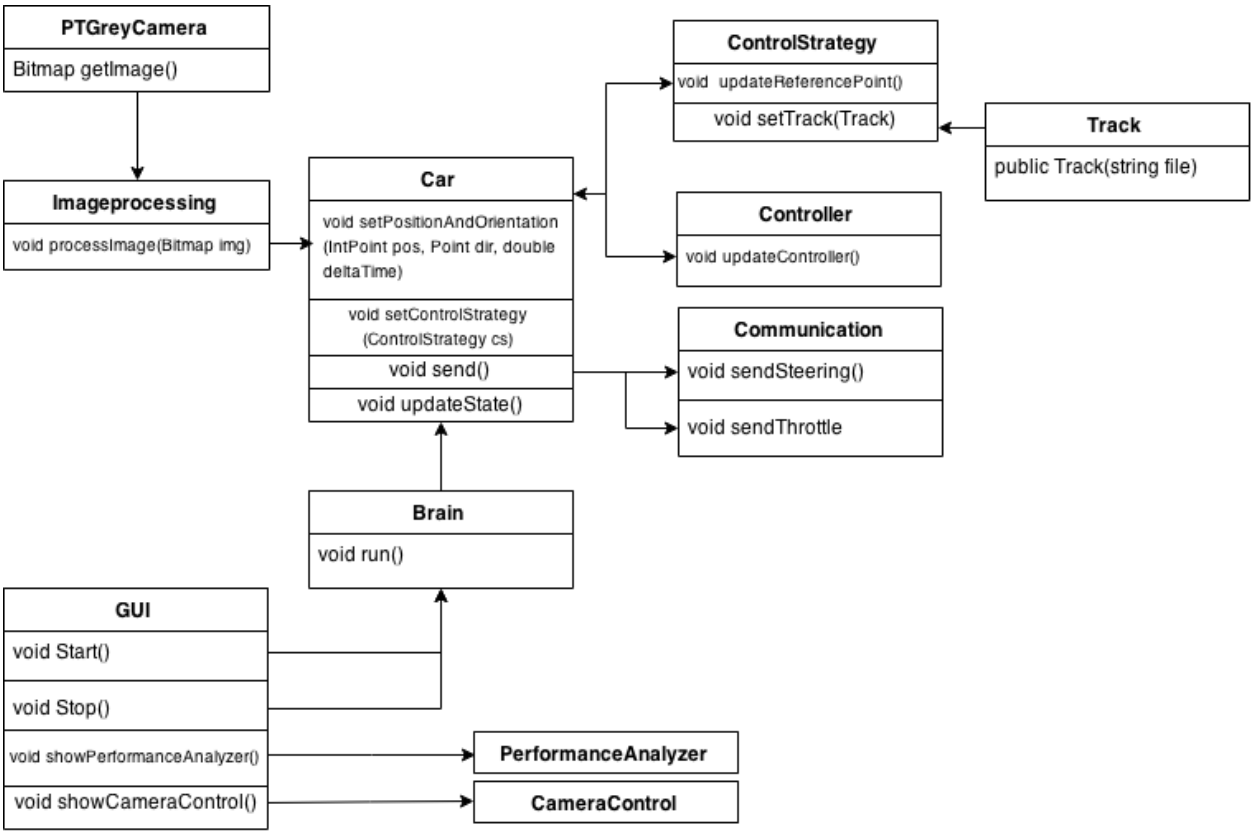


Figure 15: Overview of the different classes in the main program.

The `PTGreyCamera` and `Imageprocessing` classes are running in their own threads as stated above. The controller thread is running the `Brain` class, which holds the information of all active vehicles in the platform. These vehicles have their own control strategy, controller, track and the ability to send data to the Arduino.

5.2.2 Communication between computer and transmitter

For the system to work properly the communication between the components is essential. As stated in Section 1.2 the platform should be able to handle cars driving at speeds around 200 cm/s without losing the race track. The communication should be fast enough to make it possible to achieve this requirement, i.e., keep the lag to a minimum as stated in Section 5.2.1. With the new hardware used in the platform one bottleneck could be the camera that has an update frequency of 150 FPS. This will be true as long as the computer used to run the platform executes the program at 150 Hz or faster. With that in mind the communication should follow the following rules:

$$t_{\text{cam}} = \frac{1}{f_{\text{cam}}}, \tag{18}$$

$$t_{\text{com}} \leq t_{\text{cam}}, \tag{19}$$

where f_{cam} is the camera update frequency of 150 Hz, t_{cam} is the camera cycle time and t_{com} is the transmission time in the communication. Using equation (18) and (19) the information needs to be sent from the computer and received to the vehicle in 6.6 ms or less. The communication also needs to be reliable and it should have some restrictions in order to protect the electronics in the transmitter from voltages that could damage it. The components used in the communication is a computer running the C# program, an Arduino Uno, a DAC and a Kyosho Perfex KT-18 transmitter. The C# program sends updated steering and throttle values to the Arduino using serial communication with the baud rate 115200 bit/s.

To achieve a more accurate output signal the upgraded platform uses TLC5620CN DACs from Texas Instruments instead of the LP filtered PWM signal that was used before. The TLC5620CN contains four 8 bit DACs. When the Arduino receives data, containing an address and a value, from the computer it is passed to the TLC5620. Depending on the data, it will update the corresponding DAC with a new output voltage. The transmitter has been modified to get voltages from the DACs instead of the potentiometers, which was the original design. When the voltage change on the throttle or steering pins on the transmitter it sends these values to the vehicle which then actuates it by changing the speed of the motor or steering angle of the front wheels. The circuit is shown in Figure 16.

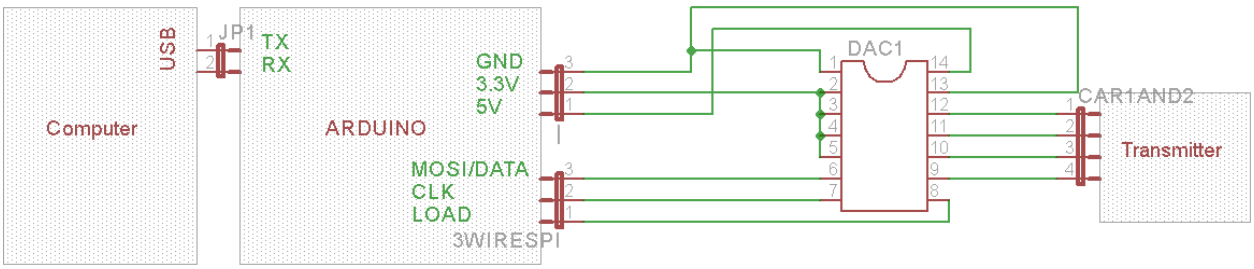


Figure 16: Schematic of the communication circuit.

To update a vehicle's steering and throttle values a 4 byte message is sent. Two of these bytes are needed to choose which DAC to use on the TLC5620 and the other two is a value between 0 and 255 that correspond to the output voltage of the DAC. The output voltage is then set according the formula

$$V_{\mathrm{o}} = V_{\mathrm{ref}} \cdot \frac{\mathrm{value}}{256} \tag{20}$$

from the TLC5620 data sheet, TexasInstruments, 2001, where V_{ref} is the reference voltage that is applied to the TLC5620 and V_{o} is the output voltage.

5.2.3 Vehicle position and direction recognition

The problems of determining position and direction of an object on the race area are split into the following sub problems:

- Extracting information from the image. In this case the positions of IR markers in the race area.
- Determine how the found markers in the area represents vehicle positions and directions.
- Distinguish which set of markers belongs to which vehicle.

The input to the system is a grey scale image with white spots, where the white spots are the reflective markers on the vehicles. The image is not perfect and will have distortions and unwanted white spots. The image is analyzed with `BlobCounter` from AForge which will use a threshold value to determine if a certain pixel is considered being a part of a white spot or not. The threshold is represented as a RGB value and can be adjusted depending on the brightness of the environment. The `BlobCounter` filter out shapes that are too large or too small from the set of found spots. Each remaining spot have their center of gravity assigned to 2D points. The `BlobCounter` returns this set of points.

To reduce the number of calculations in the program it is desirable to represent the position and direction of a vehicle with as few points as possible. Positions are easy to estimate, one point is enough, but it is impossible to calculate the orientation of the object. It is important to notice that the orientation of the object could be different from the direction of travel. With two points it is possible to draw a line between them and therefore limit the possible orientations to two ways. In other terms it will be impossible to determine what is the front and the rear of the object. Three points is the least number of the points needed to obtain both position and orientation of an object. The implementation uses an isosceles triangle to estimate the vehicle's state. Figure 17 is an illustration of the chosen pattern.

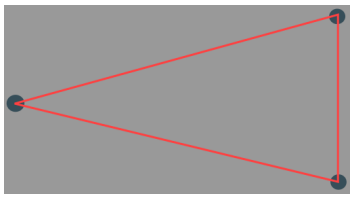


Figure 17: Image of the direction and position pattern. The grey rectangle represent the physical vehicle and the dots are the reflective markers.

The points extracted from the `BlobCounter` are organized into a set of triangles. This set contains all triangles that could be formed by the points without any doublets. In other terms if you have 4 points the program will generate 4 triangles. The example is shown in Figure 18.

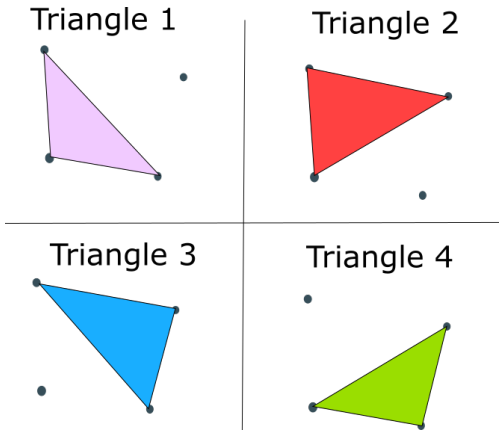


Figure 18: Triangles constructed by 4 points.

The number of triangles are binomially growing and can be calculated with the factorial function:

$$T(n) = \begin{cases} \frac{n!}{3!(n-3)!} & \text{if } n \geq 3 \\ 0 & \text{if } n < 3. \end{cases} . \tag{21}$$

The size of the problem will therefore grow factorial and is causing problems with the execution time with larger number of points to analyze. This problem will be solved later in this section.

A lot of the found triangles will not be vehicle triangles. All of the triangles in the set have to be evaluated. The evaluation of a triangle is made by a comparison of another triangle. A quality value for how well a triangle fits into another triangle can be calculated by the following equation:

$$\begin{aligned} q = & e^{||p_1 - p_{1comp}||} + e^{||p_2 - p_{2comp}||} + e^{||p_3 - p_{3comp}||} + \\ & + e^{|l_1 - l_{1comp}|} + e^{|l_2 - l_{2comp}|} + e^{|l_3 - l_{3comp}|} + e^{|h - h_{comp}|}, \end{aligned} \tag{22}$$

where the p:s correspond to points in the triangle. The l:s corresponds to the lengths of the sides and h is the height of the triangles. An illustration of the comparison is shown in Figure 19.

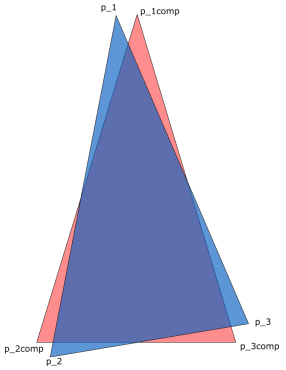


Figure 19: An illustration of two triangles that are compared with each other.

By using e as a base for the error, the value q is increasing faster with larger errors. Therefore the solution is more stable and it is easy to achieve a good approximation of which triangle has the biggest correlation to the ideal triangle.

The algorithm to estimate the current position and orientation of the vehicles relies on previous vehicles states. Therefore, it is essential to initiate the system before starting the simulation. When the user press the initiate button the program will execute the following steps:

- Sort the triangles after their quality value based on the static ideal triangle for a vehicle.
- Select the triangles that has a quality value good enough to be classified as a vehicle triangle.
- Search for identification markers and assign an identification number to the vehicle if it finds any identification markers. If no markers are found the program will discard the triangle.
- Present the found vehicles to the user.

It is important that the vehicles are close to the center of the race area because the distortion of the wide angle lens scrambles the triangles close to the edge of the image.

The identity of a vehicle triangle is represented as the number of markers on the vehicles. By using the direction, height and base points of the isosceles vehicle triangle the program calculates a rectangular search zone for the identification markers on the vehicle. The rectangle is constructed by four points were the base points of the isosceles triangle are two of them. The other two are calculated by the following equation:

$$p = \text{base} + \text{direction} \cdot \text{height}. \tag{23}$$

Where p is the new point, base is one of the base-points, height is the height of the triangle and the *direction* is a normalized vector in the direction of the triangle's sharp corner. A dot is considered to be an identification dot if it is within the search zone of the vehicle and it is not part of the triangle.

The following algorithm is used to check if a marker is within the search zone.

```
Calculate the area of the four triangles between the marker
    and each side of the rectangular search zone.
Summarize the area of the four triangles.
Compare the sum of the area to the rectangular search zone's area.
IF the area of the triangles is greater then the search zone's then
    The marker is not considered to be inside the rectangle.
ENDIF
```

Figure 20 illustrates the concept of the algorithm. The red regions is the excess area of a marker outside of the search zone.

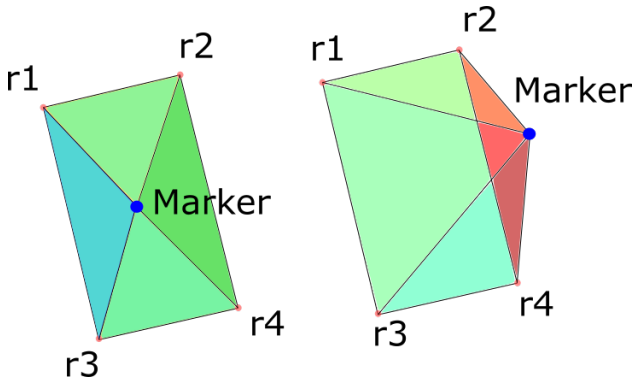


Figure 20: A dot inside and outside of the search zone and the triangles it makes.

The concept of finding and estimating direction and position of a vehicle is similar to the algorithm used to initialize the system. But it has to be more flexible and faster because it has to execute in 6.6 ms, which is

the update time of the camera, and be able to detect the triangle even if they are scrambled by the distortion of the camera lens. To meet these requirements two problems needs to be solved:

1. The `BlobCounter` has to check each pixel in the picture, even though the vehicles only occupy small areas of the image.
2. The number of possible triangles is growing factorially with the number of dots.

In order to solve these problems the algorithm searches for one vehicle at a time and divides the image into smaller sections around the vehicles. This limits the search area for `BlobCounter` and saves time when it only searches for white spots close to the last known position of the vehicle. By reducing the size of the image it also avoids unwanted markers outside of the search area and reduces the number of triangles the program has to search through. The program discards markers that it knows belongs to other vehicles with the same algorithm used for the identification markers. By doing this it will prevent the program from confound the vehicles if an identification marker is unrecognizable.

As previously stated it is hard to estimate with a static reference triangle if a triangle that is located close to the border of the image is a vehicle triangle or not. The solution therefore includes a dynamic comparator. The triangles are sorted by their quality value, which is affected by the distortion caused by the lens. Therefore the program saves the previous triangles and compares them with the found triangles. Instead of using a static standard triangle as comparison triangle, the previous triangle is adjusted along with the vehicles movement over the image. If it is located close to the edge the comparison triangle will be almost as distorted as the new triangle found.

Pseudocode for the algorithm is presented below:

```
FOR EACH vehicle
    IF the vehicle is lost THEN
        Use the whole image.
    ELSE
        Use a smaller image centered where the vehicle was last time.
    ENDIF
    Extract dots from the image.
    Sort away dots that belongs to other vehicles.
    Create triangles.
    Sort triangle according to the quality value based on the triangle
        found last iteration.
    IF the triangle with the best quality value is good enough THEN
        Use that triangle to determine position and direction
            and assign the new position and direction to the vehicle.
    ELSE
        Set the vehicle lost.
    ENDIF
ENDFOR
```

5.3 Follow race track

For a vehicle to follow a race track some problems needs to be solved. A track must be defined in a mathematical way that enables a steering angle and throttle value to be calculated. Simulations verify the algorithms before implementing them in the real world platform.

5.3.1 Creating the race track

The race track in the platform is a virtual track. The race track is created in MATLAB and saved as a .txt file that the user loads in the program before starting the platform. The .txt file contains a $3 \times n$ dimensional matrix representing the track. Each row contains a x, y and v value, where x and y are the position in the race area and v is the desired speed at that position. Since the camera is of a $5:4$ format with resolution of 1280×1024 pixels this also gives the axis for the track matrix. In order to get the positions in pixels only integers are used.

The racetrack is created by points given by the user with the help of `ginput()` in MATLAB. The `spline()` function then interpolates a curve between these points. Since the `spline()` function is not designed for closed loops the interpolation is made over three laps but only the mid lap is saved in the track matrix. The track matrix now contains a two dimensional matrix of size n, where n is the number of rows in the matrix, i.e., the number of positions in the track.

The interpolation gives a discrete curve with different lengths between the points. The distance between two points is determined by how sharp the curve is turning in that area. In a sharp turn there is a higher concentration of points. With this information a vector is created that describes how the track is changing relative to its easiest point (straight line) by calculating the length between every point. The vector is then normalized and gives a vector of size n with values between $0-1$, where 1 is the easiest part of the track and values toward 0 the hardest. The vector tends to become a bit noisy and this is solved by smoothing it with the MATLAB function `smooth()`. This vector is then added to the track matrix. The vector is multiplied by the maximum speed, chosen by the user in the program before starting the platform. The track matrix is now $3 \times n$ dimensional with positions and speeds and is saved as a .txt file.

5.3.2 Control strategies

When the track is created the next step is for the vehicle to determine how to drive along it. Because the vehicle can follow the track in a variety of ways a few different algorithms have been created to do this. These algorithms are called *control strategies*. They all have in common that they choose a coordinate for the vehicle to drive to and the speed of which to do so.

The most basic *control strategy* is called `JustFollow`. This chooses between the points in the track and selects the most suitable point for the vehicle to drive towards at the speed corresponding to that point. To find the most suitable point three different factors are taken in to consideration; the distance from the vehicle to the point with a set minimum, the angle from the vehicles heading to the point and the difference in track matrix index between the point and the previously chosen point. The minimum distance is used to smooth out the steering reference. Because when the vehicle gets close to a point and is not pointing exactly at it the angle gets quite big until the next point is selected. Selecting points further away also gives the vehicle some time to react to whats happening ahead. The index difference makes it prefer a point that is closer to the previous point. This enables different parts of the track to be very close to each other and even cross each other without the vehicle confusing which part it is following. All these factors have an adjustable weight associated with them and the track point with the lowest sum is selected.

The more advanced *control strategies* `Overtaking` and `Platooning` adds their own points along the track (see Section 5.4).

5.3.3 Controlling the vehicle

PID controllers are used to make the vehicle drive towards the point selected by the *control strategies* with the desired speed and steering angle. The steering and the throttle have a PID controller each. The error in

steering is the angle between the heading of the vehicle and the angle to the point. The error in throttle is the difference between reference speed in the track matrix and the actual speed of the vehicle at the moment.

There is also an option to use a manual controller where the operator can control a vehicle via the arrow keys on the keyboard.

5.3.4 Simulation of following race track

The simulation is made of a mathematical model implemented in MATLAB. The simulation is structured as a main class, which is running the process and handling the visualization, a class creating a reference signal, a mathematical model of a vehicle, a controller handling the regulation and a control strategy which consist of algorithms making the vehicle follow the reference track.

The platform is able to handle a simulation of multiple vehicles at the same time. In the first step of the execution the user choose the number of vehicles and a speed for each vehicle. Then the user creates a track using a function called `setTrack()`, which is the same function used when creating a track for the real platform. Thereafter the simulation starts. To simplify the calculations they are run one vehicle at a time. In order to act as the real world platform the simulation iterate with the same time steps (1/150 s) as the real world platform.

Three functions are executed in the main loop. First the function `Reference()` chooses which control strategy to use (see Section 5.3.2). The `Controller()` calculates the throttle and steering values according to 5.3.3. These values for steering and velocity are sent to the function `Update()`, which simulates the vehicle's behavior using a simplified version of the mathematical 2-point model described in Section 3 and sets a limit for the vehicle's maximum outputs of steering and acceleration.

The vehicle's values for position, steering and speed are saved in a lists which is used to display the results of the simulation. There are two different ways the program can show the results of the simulation. The first is to make a real time playback of the simulation where the vehicles are visualized as dots which are moving along a track. The second is plotting graphs of reference and actual values for speed and heading for each vehicle, as well as a plot of the reference track and the vehicle path.

5.4 Automotive algorithms

A part of this project is to develop algorithms for collision avoidance, overtaking and platooning. These algorithms are referred to as "automotive algorithms". The implementation of these algorithms in the MVP is described in the two following subsections.

5.4.1 Collision avoidance and overtaking

In this project the system for collision avoidance is not solved by breaking in front of an obstacle. Instead the vehicle drives around the obstacle, in other words performing an overtaking. The difference when referring to collision avoidance or overtaking is that collision avoidance refers to stationary objects and overtaking to moving objects. The chosen approach assumes that all obstacles can be overtaken, i.e., a wall would cause the vehicle to crash.

Figure 21: The dead zone (DZ) and activation zone (AZ) around an obstacle.

The algorithms for collision avoidance and overtaking are made with the help of two zones (shown in Figure 21), the first zone is the AZ (activation zone). When a vehicle reaches the AZ of an object or another vehicle the system starts to recalculate the reference track. With the help of the second zone, the DZ (dead zone), the algorithm recalculates the reference track around the DZ of an obstacle. In order to know when a vehicle is inside the AZ or the DZ of an obstacle the distance between the vehicle and the obstacle is measured. The recalculation of the reference track is made by placing new points around the DZ. Three points are placed parallel to the track at the point of the obstacle, just outside of the DZ. One point is placed between these three points and a point just outside of the activation zone on both sides of the obstacle.

The reference track is recalculated and modified in every iteration while a vehicle is inside an AZ. If the object to be avoided moves, a new trajectory that prevents the vehicle from colliding will be calculated. This means that the algorithm can handle both overtaking and collision avoidance situations.

5.4.2 Platooning

The platooning algorithm is implemented as two different phases. A vehicle's current phase depends on whether it is close enough to the leader to start following or not. The platooning behavior is initiated when the user assigns the platooning control strategy to a vehicle in the GUI. The user is responsible for choosing which vehicle should be followed. At the start the platooning vehicle is in the first phase, the *searching phase*. In this phase, the vehicle will continue to follow its assigned track. In each program iteration a check will be made to see if the platooning vehicle is close enough to the leader to change to the second phase, the *following phase*. The required distance is equal to two times the desired platooning distance, which can be changed using a control in the GUI. When the *following phase* is reached, the vehicle will stop using its own track to choose the next point to travel to, instead it will use a vector containing the leader's previous positions. If the following vehicle falls too far behind the leader it will leave the *following phase* and fall back into the *searching phase*.

The distance between two vehicles is calculated as the distance between the following vehicle's position and a line perpendicular to the heading of the leader vehicle, visualized in Figure 22. Measuring the distance this way, as opposed to measuring the euclidean distance, means that a following vehicle will strive to stay behind the leader.

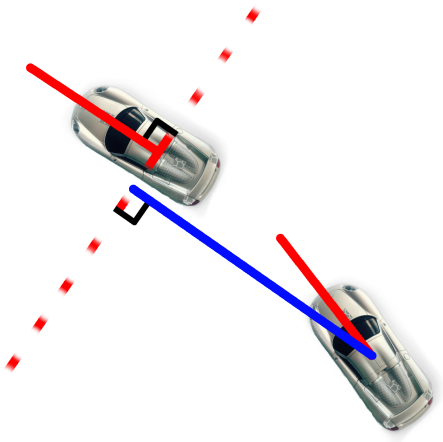


Figure 22: The distance between the vehicles is measured along the blue line.

When the following vehicle has reached the *following phase* a PID controller is used to keep the desired distance between the vehicles. The output of the system is the distance between the vehicles. The PID controller calculates a reference speed that is then passed to the vehicle's own throttle controller. The controller parameters used in the platooning can be changed in the GUI.

In order to be able to test the controller parameters in the platooning algorithms a basic simulation is implemented in Simulink. The simulation of platooning is made to test the velocity regulation since this is seen as the main problem when working with platooning. It is implemented as a transfer function based on the behavior of a step response of the platform using the vehicles standard regulator used for regulating the velocity. To this basic vehicle model the platooning controller is added to regulate the output signal in order to keep the desired platooning distance.

5.5 Interaction between the user and the system

The user interaction with the system is made through a graphical user interface (GUI). The focus of the GUI is to let the user change system parameters in an easy fashion and to display useful data about the system's performance.

As the MVP is meant to be an experimental platform for vehicle control and control strategies, the GUI needs to be flexible in what settings it enables the user to change while running. Making some parts of the GUI change dynamically solves this. Each controller or control strategy type has a specific control panel. When the user selects a certain controller or control strategy for a vehicle the corresponding control panel is displayed in the GUI as well.

When developing controllers or control strategies for vehicle control it is important that the system is easy to troubleshoot. To assist in this task the GUI lets the user open two different windows. The first troubleshooting window is the *camera control* window. The *camera control* window displays the image captured by the camera and enables a wide variety of overlays to be painted on top of the image, such as the current race track, vehicle position history, detected spots and current point to travel to. The other troubleshooting window is the *performance analyzer* window. This window lets the user select desired data and plot this data in a 2D graph. Some examples of available data to plot are: vehicle velocity, vehicle control signal, vehicle speed and heading reference signals, platooning error and main loop execution time. The main window GUI also contains a data grid, which displays the current values of some variables, such as vehicles' x and y positions and control signals.

6 Results

This section presents the results of the project. The performance of the platform is compared to the project goals and to the previous platform when applicable. The first section presents the results regarding the hardware and recognition system of the platform. The second section is about how well the platform controls the vehicles. The third section presents the results regarding the automotive algorithms. Finally, there is a section about the user interface of the platform.

6.1 Upgrades to the platform

The upgrade is implemented to the meet the purpose and goals of the project. Because of the hardware changes in the platform and because the old code did not have a very modular structure, almost all code had to be rewritten.

6.1.1 New components for the physical platform

The new camera is 5 times faster the the old camera. Therefore the camera is no longer a bottleneck in the platform. To achieve the project goal of driving a car at 200 cm/s, the old camera would cause the car to travel 6.7 cm between each update. With the new camera this distance is reduced to 1.3 cm. The new camera has a lower resolution then the one in the previous platform (Hagebring et al., 2014). The resolution of the previous camera was 1920×1080 pixels, the new one has a resolution of 1280×1024 pixels, which means that the old camera had 58 % more pixels. However, with the new recognition system and the new camera's higher update frequency the accuracy has increased, see equation (24).

The old GRATF system used $6 \times 7\,\mathrm{cm}^2$ markers, which was wider than the actual car. The markers of the new IR system occupies $0.5 \times 0.5\,\mathrm{cm}^2$ of the car's roof which leads to a negligible effect on the performance of the car. The appearance of an old and a new marked car is shown in Figure 23.

(a) Car with a glyph used by the old GRATF system.

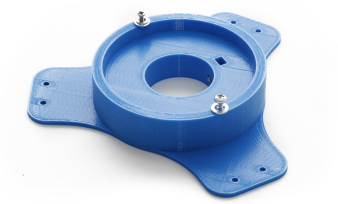
(b) Car with IR markers used by the new IR system.

Figure 23: Comparison between the old and the new looks of the cars.

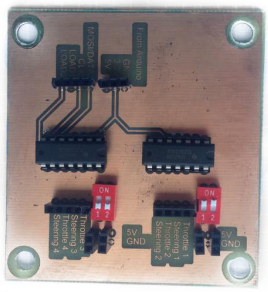
The threads in the main program have been observed with Visual Studio's concurrency visualizer and profiling tool. This shows that the program is doing parallel computing in some parts but not in all parts. However, it does not affect the platforms performance because the average throughput time, the time from a new image acquired from the camera to the calculated output values has been send to the Arduino, is faster than the capture rate of the camera most of the time. There are some occasional slow downs when more than four variables are plotted in the performance window, as stated in Section 6.4. It also occurs if there is a big reflective object (a person with a bright shirt for example) in the camera's view. This results in many

computations for the GUI part of the image processing. But during normal circumstances the program is running as fast as the camera's update frequency.

An adapter was 3D printed in order to mount the IR filter to the lens. The adapter also features three mounts for LED lamps and their heat sinks. The adapter is shown in the Figure 24a, drawings can found in Appendix D. The PCB for the DACs was etched and soldered at Chalmers Robot Förening and is shown in Figure 24b.



(a) The 3D printed lens adapter.



(b) The PCB for the DACs.

Figure 24: Constructed components to the platform

The mount for the PCBs was 3D printed. The result of the PCBs mount equipped with one Arduino Uno, one DAC PCB and two controllers is shown i Figure 25.

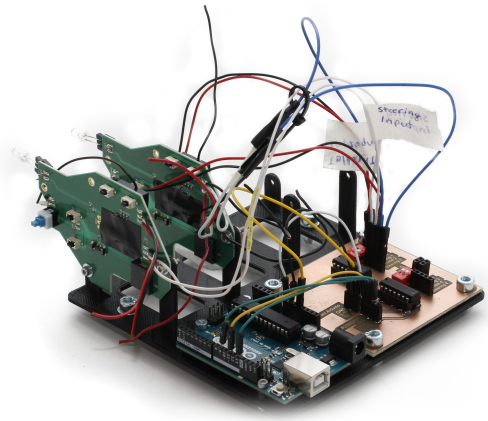


Figure 25: The circuit board mount with one Arduino Uno, one DAC PCB and two controllers mounted.

6.1.2 Communication between the computer and RC vehicles

To examine the speed requirement of the communication in the platform a time analysis of the different subsystems has been done. Firstly the communication between the program and the Arduino is limited by

the baud rate. The baud rate that is used is 115200 which means that 14400 bytes can be sent every second. In the current system two vehicles can be used, which means 8 bytes (4 bytes bytes per vehicle as mentioned in Section 5.2.2) of information needs to be sent every time the values needs to be updated. This means that each vehicle can be updated 900 times per second, which far exceeds the update frequency of 150 Hz that the camera runs at. Between the Arduino and the DAC a 3-wire Serial Peripheral Interface (SPI) is used with a clock speed of 1 MHz that matches the DACs recommended clock speed. To update all vehicles a total of 64 clock cycles are required, 8 bits for each of the 8 bytes. Thus this part can update the vehicle around 15000 times per second. As soon as the DAC is updated the output voltages change. Because the wires between the DAC and the transmitter is very short there is no need to calculate the update speed between these components, it could be said to change instantaneous. Thereafter the data is sent to the vehicle. This is done in the same way as when the transmitter is unmodified and therefore no investigations of transmission speed between the transmitter and the vehicle have been made. The transmission is made on the 2.4 GHz frequency band and should give very high transfer speeds. Having concluded that the transmission speed between the components is sufficient the components themselves also needs to be investigated. The results of this is shown in Table 2.

Table 2: Time analysis for the hardware components when using two vehicles.

Component	**Time** (ms)	**Tool used to measure**
Computer	0.2200	Visual studio profiler
Arduino	0.0980	`micros()` in the Arduino sketch
DAC	0.0002	Datasheet
Total time	0.32	-

The communication passes the requirements. Up to twelve vehicles could be used simultaneously without slowing down the system. Above that point, the bottleneck would be the communication speed between the computer and the Arduino.

6.1.3 Vehicle position and direction recognition

During normal circumstances the position and direction recognition systems finds the vehicles. In specific circumstances it is harder for the recognition system to find the vehicles and they can get mixed up. These circumstances are when the vehicles are in each others search zone (see Section 5.2.3) or when the image processing does not recognize a vehicle due to low IR reflection near the edges of the race area and therefore searches the whole image. As shown in Figure 26 the system can most of the time successfully estimate the position of two vehicles with overlapping search zones.

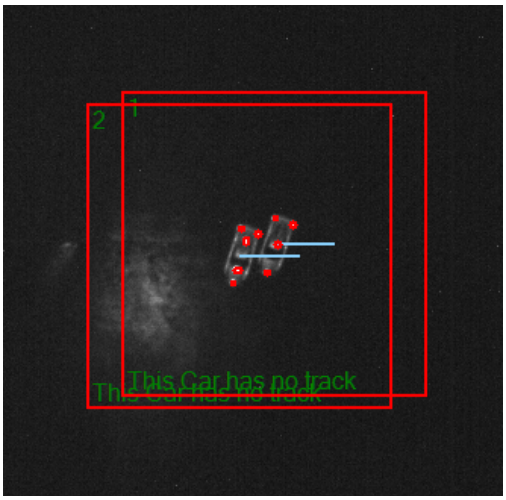


Figure 26: Two cars standing close to each other with search zones shown as red rectangles.

When a vehicle is close to the edge of the race area the system can estimate the vehicle's position even if the vehicle triangle is affected by distortion. The result is shown i Figure 27. The figure shows two vehicles where one vehicle has been moved along the race area to a position near the edge of the race area.

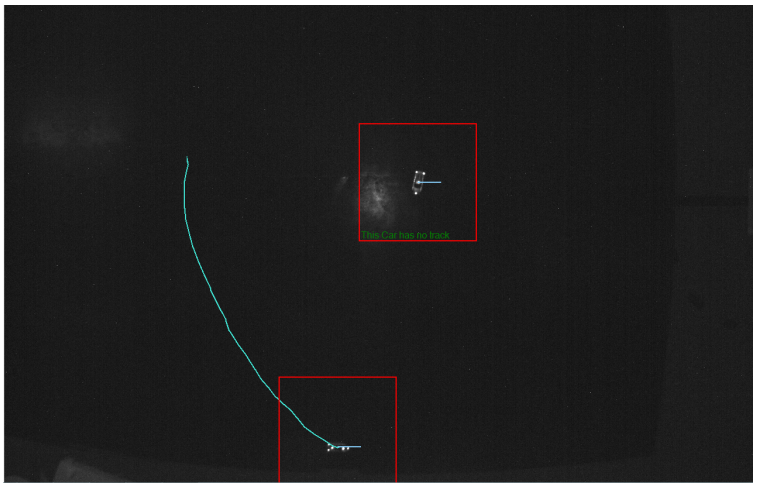


Figure 27: A car close to the edge of the race area.

With multiple markers the number of triangles found in the image increases binomially with the number of markers as stated in 5.2.3. Figure 28 shows that the program can handle a multiple markers and filters out the unwanted triangles when finding the vehicles.

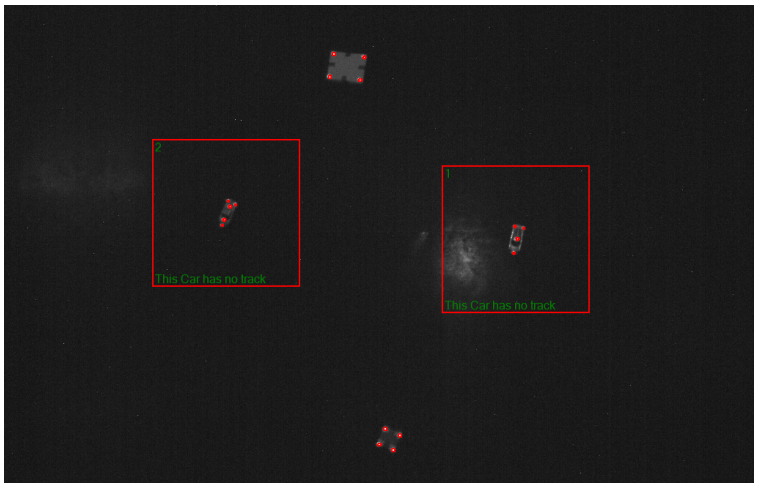


Figure 28: Distortion with unwanted markers in the race area.

The execution time of the image processing presented in Figure 28 is shown in Table 3. The image processing is done in less than 6.6 ms, which is the update time of the camera. The start time is before the first execution of the image processing and the stop time is after the last execution.

Table 3: Presentation of the execution time of the image processing.

Iteration	**Start time** (ms)	**End time** (ms)	**Execution time** (ms)
1	689	694	5
2	694	697	3
3	746	749	3
4	756	760	4
5	884	888	4
6	924	927	3
7	928	930	2

For position estimation the worst case scenario is when the centre of the vehicle is located between 4 pixels. Because of the brightness is unstable the image will change between each iteration and the center of gravity for the blobs may flicker. Therefore the position may flicker between 4 pixels. This leads to an accuracy of the recognition system of ± 1 pixel, which corresponds to ± 2.6 mm (see Section 5.2). This is only true if the vehicle is stationary. Compared to the previous platform with an accuracy of ± 1.86 mm, Hagebring et al., 2014, for stationary vehicles, this is a setback because of the lower resolution of the new camera. However, this is only considering when the vehicle is stationary. When the vehicle is in motion at 200 cm/s the accuracy of the recognition systems is

$$\pm \left(2.6\,\text{mm} + 200\,\text{cm/s} \cdot \frac{1}{150}\,\text{s}\right) = 13.3\,\text{mm}. \tag{24}$$

The error margin is the accuracy of the recognition system and the distance the car can move during one iteration, which is 200 cm/s multiplied with the sample time $1/150$ s. This results in an error margin of ± 13.3 mm. This is five times smaller compared to the ± 67.1 mm of the previous platform at a speed of 200 cm/s. This is due to the five times higher frame rate of the new camera.

6.2 Follow race track

The car is able to follow race tracks as long as there are no turns sharper than the car's minimum turning radius. The vehicle finds a reference points in the track just as good in the simulation as in the real world. But the car struggles with tracks where there are close distance between different sections and choose the wrong reference point. The controller parameters calculated in the simulation does not work well with the real car probably because there are additional dynamics in the real car.

6.2.1 Creating the race track

The verification of the results from creating the Track matrix is made in MATLAB. In order to make sure that the race track's x and y coordinate vector is correct the track is plotted. In order to verify that the velocity vector is correct the time in every point is calculated using

$$t = \frac{v}{s}. \tag{25}$$

Then a dot is plotted on the track staying the calculated time in every point of the track. This results in a dot moving around the track in different velocities depending on the value of the velocity a that point. This helps the user to observe how the velocity changes around the track. The result is a high velocity on straights and low in sharp turns. A result of a created track is shown in Figure 29.

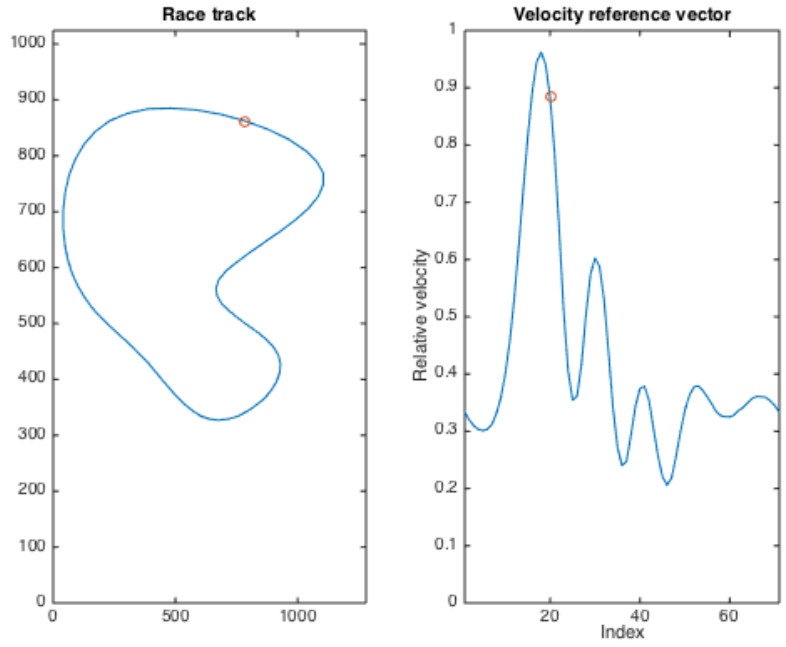


Figure 29: The plot to the left shows in blue a created race track and to the right in blue its velocity reference vector. The red dot to the left shows a position on the track and to the right its corresponding position in the velocity reference vector.

6.2.2 Simulation of following a race track

The simulation of the platform is implemented to have a behavior, which is corresponding to the behavior of the real platform. In Figure 30 the output of a simulation is shown, showing a car's speed, steering and position compared to a reference signal.

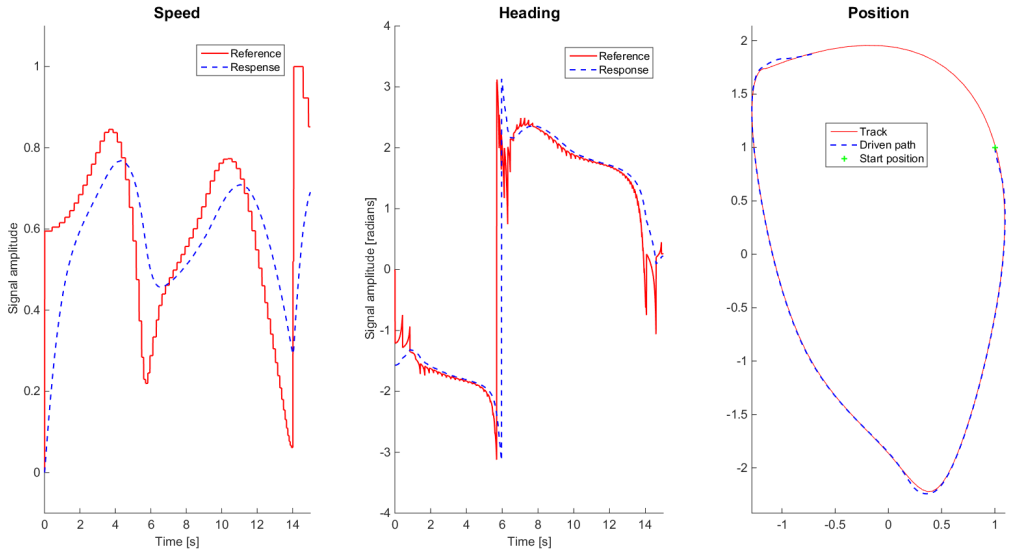


Figure 30: The graphs display the result from a simulation. The reference signal is displayed in red and the system response is displayed as a blue pointed line.

From the speed graph it can be deduced that the car's velocity does follow the reference pattern. However, the controller do not change the velocity at the same pace as the reference signal and therefore there is an offset when the required reference signal is changing quickly. From the heading and position graphs, it can be deduced that the car's steering output corresponds well to the reference signal in that there are only offsets from the reference when the required reference are changing faster than the car has the capability to turn. It can also be seen that the heading reference is noisy.

6.2.3 Following race track in real world

The platform has been tested on tracks with different level of difficulty and with a maximum of two cars driving simultaneously. How well the cars are able to follow the track depends on the speed and the difficulty of the track. On a circle with a radius of 75 cm with a constant speed reference, an average velocity of 220 cm/s has been observed while the car is following the reference in a stable way. At greater speeds the car loses traction from the carpet and spins around. Figure 31 shows position data from a test run when the car is following a circle at 140 cm/s which is considered an easy track.

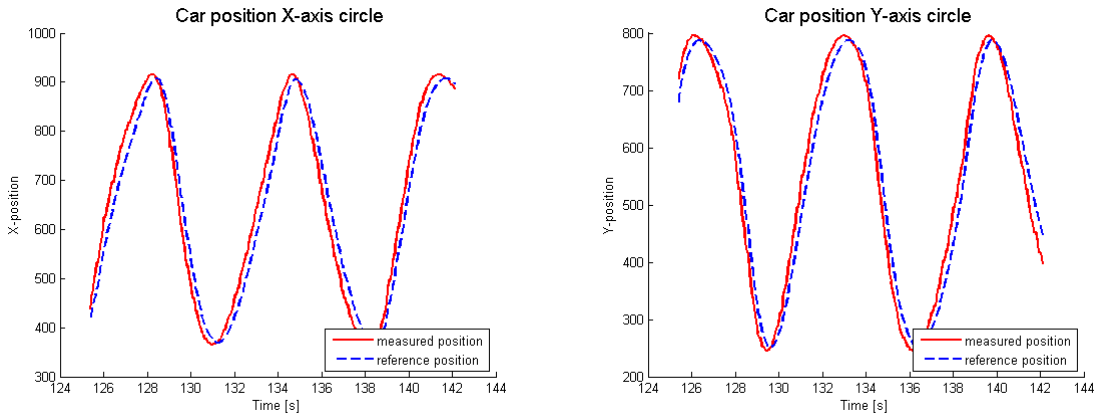


Figure 31: Positioning data from a car while following a circle. The reference signal is displayed in red and the car's position is displayed as a blue pointed line.

As seen in the figure the car is following the reference well. The observed delay in the graphs comes from the fact that the car is always trying to get to positions in front of it. By shifting either the reference or measured position a little bit it is showed that there is no constant positioning error while following an easy track. It can also be seen that the following algorithm is working well. The reference position shows a smooth sine and cosine wave, as expected when following a circle.

The noise in the steering reference that could be seen in the simulations of the car is also present on the real platform. The amplitude of the signal is small and it is not a problem for the car while following a track. In Figure 32 data from the steering control signal when the car is driving in a circle is presented. The maximum output the controller can give is 1 which results in a steering angle of 30 °.

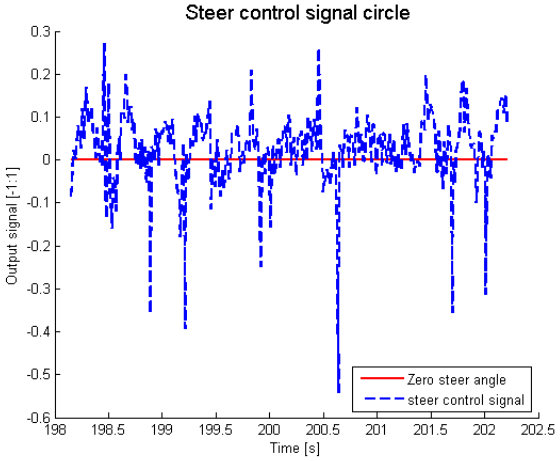


Figure 32: The steering control signal when travelling in a circle.

The results of the velocity measurements is not following the reference in the same way as it does in the simulations. Even when the reference speed is constant the platform is not able to maintain a constant velocity. The model used in the simulation lack some of the dynamics the real RC-cars has and because of this the theoretical parameters of the PID controller could not be used. Attempts with online tuning methods

like the Ziegler-Nichols method has been tested but have not been successful either. Instead experimental parameters have been used which have been tested and revised during the development of the platform. As seen in Figure 33 the velocity measurements oscillates around the reference, which results in loss of traction when the speed gets too high.

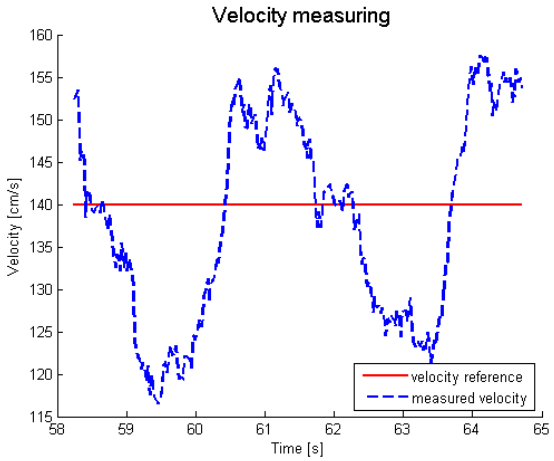


Figure 33: A car trying to hold a speed of 140 cm/s.

A harder track is called "Nürburgring" which uses almost all of the racing area and has sharp turns and a reference speed that varies over the different parts of the track. On the harder tracks the car is not able to follow the reference position as good as on the easy tracks. When driving slow and by trimming the cars the result can be very good. But this is not repeatable and depends on a lot of different parameters such as the battery charge, maximum speed and trimming settings. Figure 34 is a result from a test run where no specific tweaks have been made.

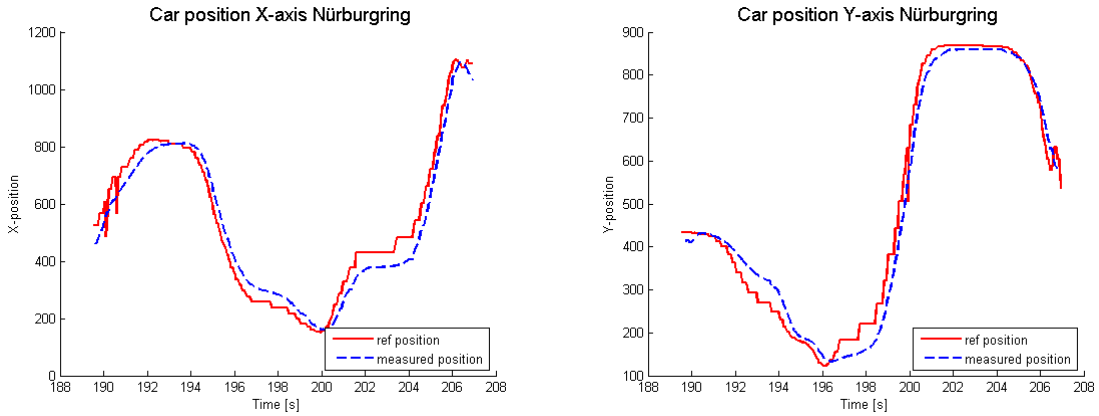


Figure 34: Positioning data from a car while following the "Nürburgring" track.

In the graph it is shown that sometimes the car struggles to find the right reference point. This results in a flickering between points which is not desired, this behaviour can be seen in the beginning of the left graph in Figure 34. However, it still manages to follow the track but not as closely as on easier tracks. A problem on a track like this is that the vehicles sometimes get lost by the image processing, which leads to problems updating the state of the vehicles.

Since the cars are not able to follow the reference speed while it is fixed they do not follow the reference speed very well when it is varying either. However, it will accelerate in fast sections and slow down in sharp corners but the response from the controller is too slow and it does not match the velocity reference vector. A result from this is shown in Figure 35.

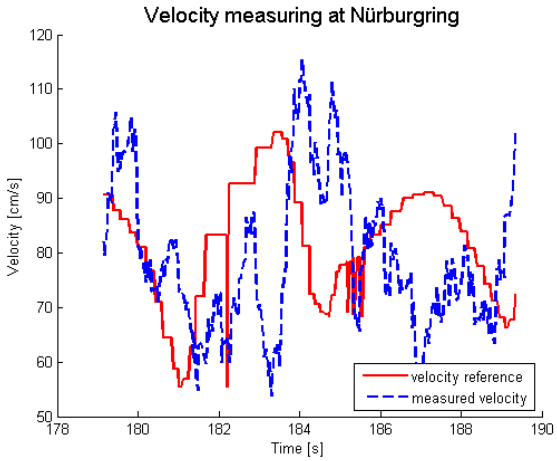


Figure 35: A car trying to follow the reference speed at the "Nürburgring".

6.3 Automotive algorithms

Very basic algorithms are implemented for both overtaking and platooning. They are both able to perform what they should under certain conditions, but they are not very reliable.

6.3.1 System for collision avoidance and overtaking

The performance of the system for collision avoidance and overtaking meets the goals in Section 1.2. The system is able to successfully overtake both stationary and moving obstacles on easy tracks, such as a circle, but struggles on harder tracks.

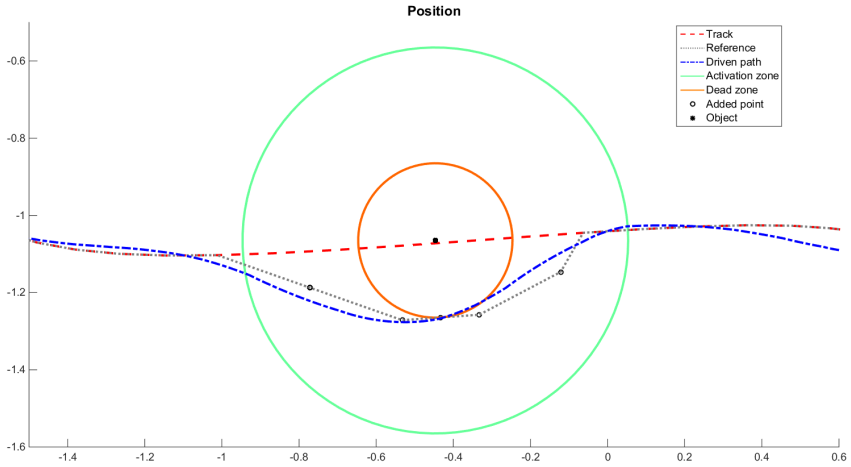


Figure 36: The AZ and DZ around a obstacle and the recalculated reference track around it. The car is driving from the right to the left.

The algorithms for the collision avoidance and overtaking systems have been simulated in MATLAB with good results. The vehicle can find the obstacle and recalculate a new way around it and then continue on its normal reference track, see Figure 36. The simulation of the overtaking looks realistic and natural around the object. The simulations have only tested overtaking of stationary objects but it is assumed that the results would be the same for moving objects since the vehicle is recalculating a new reference track each iteration.

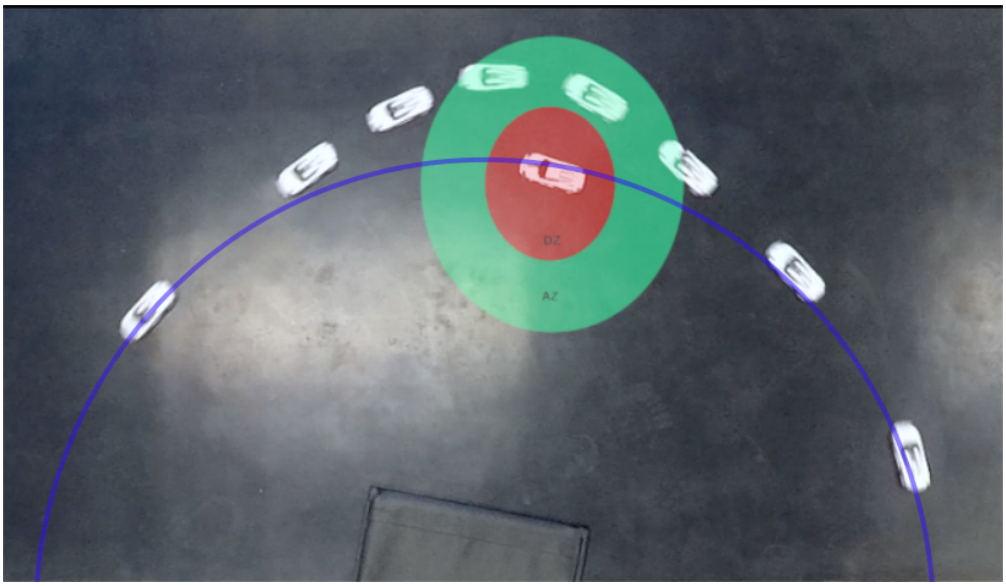


Figure 37: Action sequence photo of an overtaking of a stationary vehicle with graphical overlay showing the AZ, DZ and normal reference track.

The actual system can detect when an obstacle is in a vehicle's track and calculate the AZ and DZ of the obstacle, see Figure 37, and then recalculate a new reference track around the obstacle. Since this is done every iteration this works well for both moving and stationary obstacles. If the obstacle is on the left side of the track, the overtaking will be performed on the right side of the obstacle and vice versa. But this is not

always the optimal side though, considering where the obstacle in front of the vehicle is heading (if moving) and its own current position on the race track. There are some scenarios that the system can not handle and the overtaking fails, causing the vehicle to crash into the obstacle in front or losing its own race track.

The system has also been tested to handle frontal collision scenarios when two vehicles are driving in the opposite directions on the same track. But the result of this is varying and rely mostly on luck to succeed.

6.3.2 System for platooning

The used platooning algorithm has shown some promising initial results, but there is a lot of room for improvement. A vehicle can follow another vehicle on a somewhat constant distance when the leading vehicle is following an easy track, such as a circle. However, there is quite a large uncorrected control error, often several times larger than the reference signal, which seems to increase as the speeds of the vehicles are increased. There has not been any proper testing on a more advanced track.

Aside from the speed regulation, the platooning algorithm seems to be working. The vehicles chooses points to travel to correctly and they change phases as intended (see Section 5.4.2).

6.4 Interaction between the user and the system

The final result of the implemented GUI is shown in Figure 38. The GUI lets the user select different settings for different vehicles. The possible settings include: choosing controller type and controller parameters, control strategy and control strategy specific settings and reference track. It also aids the user when evaluating the system performance through a plotting window and a camera control window. The plotting window lets the user display desired data as graphs. The camera control window shows the camera image, along with some different overlays, such as dots detected by the camera, vehicle track and vehicle directions.

As it stands, it is unclear how much the system is slowed down by the GUI in the default state. Settings may be changed during run time without the system slowing down noticeably. The camera control and the performance analyzer windows also seems to be running without slowing down the system. However, if the user tries to plot a lot of data at the same time the system may slow down a bit.

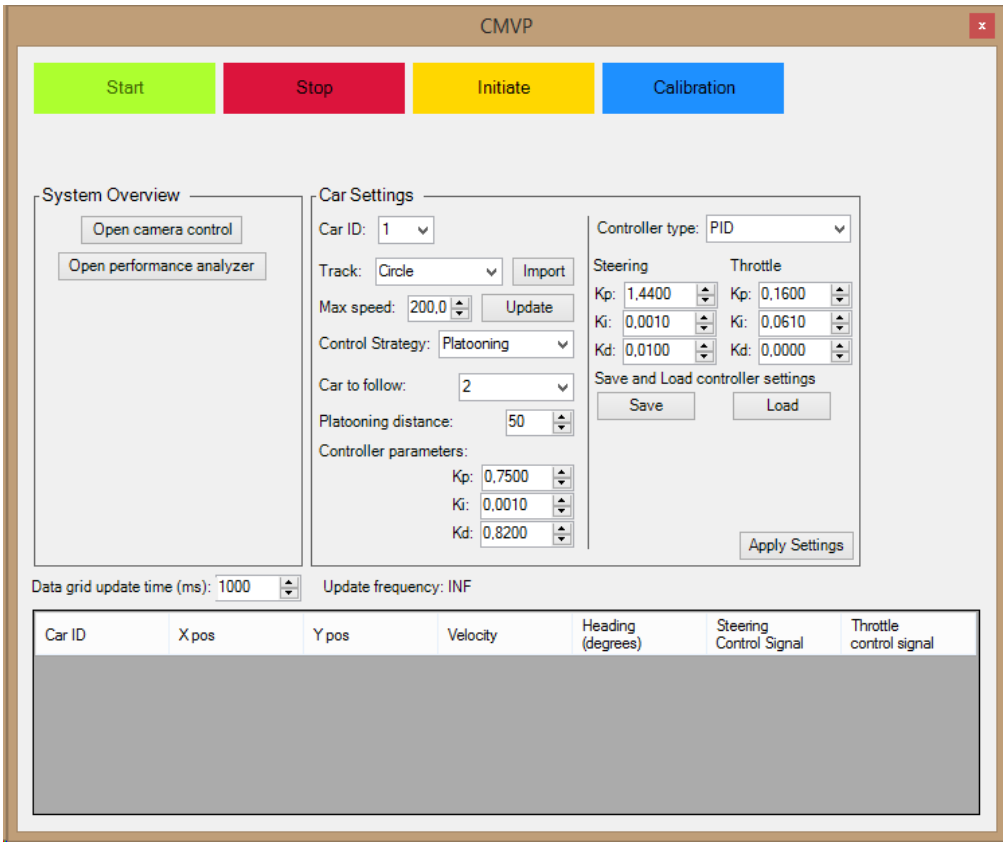


Figure 38: Screen capture of the user interface. The areas bellow "Control Strategy" and "Controller type" changes dynamically based on what is selected in the corresponding drop down menus.

7 Discussion

The project group is satisfied with the result of the final product and there has been a huge progression compared to the previous platform. However, the MVP is not flawless and has some problems. In this section the result of the project is discussed and compared with the goals and theories that has been used during the project. The section also include sections about the environmental aspects of active automotive systems as well as suggestions for further development of the platform.

7.1 Result of the final platform

The project became more comprehensive and time consuming then first expected, therefore some parts of the project were down-prioritized. However, most of the goals that were set before the project started have been achieved. As stated in Section 1.2 the platform is now able to control RC cars to follow a trajectory at 200 cm/s. The new high frame rate camera enables the platform to drive in high speed and the new IR positioning system with markers has no effects on the performance of the vehicles. The communication is accurate and the new code has more modular structure than the previous one. Since several parts of the platform were harder to develop than anticipated and less parts of the previous platform could be used than expected, achieving the goals has taken more time than it was first planned to do. For example merging the different parts of the program code faced many problems with how the different threads should communicate concurrently.

The automotive algorithms has been developed but they are not complete. The platooning algorithms is functioning in some conditions with two vehicles but it does have problems with a constant error distance as well as problems with loosing contact with the leading vehicle. The collision avoidance and overtaking system can detect an object (if it is IR marked) in its path and is able to avoid a collision by overtaking the object in many situations. Both of the automotive algorithms are on an early stage of their development and there is still room for further improvements. However, the platform offers a good environment for testing automotive algorithms even if the platform is in need of some improvements.

7.1.1 Main program

Achieving parallel computing has been time consuming. Even though the program works well during normal circumstances there are still some things that could be improved. The main issues are that the threads are blocking each other for a long time if there is a big reflective object under the camera or many variables are plotted in the performance analyzer at the same time. These problems do not occur during normal circumstances but can happen when debugging new automotive algorithms. For example: when debugging and testing new algorithms the platform could behave in unexpected ways. Someone might need to enter the race area. Or sometimes it could be useful to display several variables in the performance analyzer, which would clog the system. It can be quite tedious to debug the platform if it is crashing all the time.

To reduce the risk of threads blocking each other data structures that handle concurrency better could be used. The biggest blocking issue comes from the fact that the acquired image from the camera is saved in a `Bitmap` that is used in many parts of the program. To create copies of the image `Clone()` is used. This results in a shallow copy that will have the same references to the properties of the objects in the `Bitmap`. If the image processing thread and the GUI thread tries to use the `Bitmap` at the same time one of them will have to wait, which results in bad performance of the program. The code should always choose the image processing thread over the GUI but sometimes this does not work. One easy way to solve this would be to create a deep copy, where each `Bitmap` object will have its own properties, but this takes too much time and cannot be used if the execution time should be lower than the camera's update rate. Another solution to this is to save the results of the image processing in variables in order to make sure that the GUI does not need to use the image at all. However, the `Bitmap` is still needed to have a live stream in the GUI so it needs

access to it somehow. One could also change the `Bitmap` from a RGB one to a greyscale one which could be fast enough to make deep copies in time but the panel that draws the live stream does not support greyscale `Bitmaps` and there has not been enough time to investigate this further.

7.1.2 Vehicle position and direction recognition

The project group is satisfied of how well the image processing manages to follow the vehicle in high speeds. The accuracy of the recognition system has made huge improvements compared to the previous platform. Frame rate was prioritized above resolution when choosing a camera. Therefore, the recognition system has become much more reliable for vehicles in motion. For vehicles moving at 200 cm/s the new system has improved the accuracy by 400 % compared to the old system. However, for stationary objects the new system is 39 % less accurate. Since our error margin for stationary recognition is rather small, ± 2.6 mm, and the focus of the project is to create a high speed platform the project group does not believe this is a relevant problem.

The position and direction recognition is not perfect. Sometimes it will lose a vehicle due to undetected markers. When this occurs, the program searches through the whole image again and usually finds the lost vehicle. This is not optimal because the vehicle will still be close to its previous position and to search the whole image takes a long time. The program could instead make a linear approximation of where the vehicle ought to be, using the previous speed and direction. The linear approximation could be done for at least one or two iterations before it give up and set the vehicle's `lost` value to true. This would also reduce the risk of mixing up the vehicles with each other when the whole image is searched. If the vehicles are somehow mixed up it will cause the lost vehicle to be controlled as if it was at the same position as the found vehicle.

Low brightness cause problems when trying to find vehicles close to the edge of the image. This results in undetected IR markers by the `BlobCounter`. This could be solved by using a more even distributed light source that gives a higher light intensity near the edge of the race area. A solution could be to install two more LED lamps to the lens adapter. However, if a new brighter light source is installed it might lead to reflections from the carpet. The camera could register these as markers, which would lead to a reduced performance from the image processing. To some extent, this problem is already present in the platform. In the center of the race area the light intensity is to high resulting in a reflective spot in the middle. This could be solved by reducing the aperture which will result in a darker image. However, this would also reduce reflections from markers near the edges, which is not desired. A temporary solution has been to put a black non reflective object over the light polluted area, but this constrains the race area.

7.1.3 Follow reference

The car is in general following the reference quite well but there are a lot of parts that can be improved. When driving at speeds above 220 cm/s the tendency is that the car starts to steer in sinusoidal waves along the race track. This is one of the biggest obstacle for increasing the speed further. The problem is caused by overshoots in steering leading to an unstable system in these high speeds. It should be possible to reduce the sinusoidal behavior by tweaking the controller parameters or implement more advanced control algorithms. Perhaps model predictive control (MPC) would be more suitable to the platform. The project has not made any research on how well it could solve the problem.

The velocity measurement of the vehicles used as feedback in the controller tends to be noisy. Using the mean value of the measured velocities during the last half second currently solves this problem. The reasons for the noisy measurements is unclear but the project group has noticed some factors that could cause this behavior. One of the reasons could be that the high frame rate of the camera while measuring low velocities of a vehicle. The vehicle needs to drive at least 40 cm/s to guarantee that the vehicle moves one pixel between each iteration. Adding the error margin of ± 1 pixel the estimated position and therefore the velocity could give inaccurate values.

In theory, when a vehicle is moving at less than 40 cm/s the system might not notice the cars movement. If the error margin is of its maximum value the system might interpret it as if the vehicle is standing 1 pixel behind its previous position. When its actual position would be more accurate to 1 pixel in front of its previous point. This would give the vehicle a speed of 40 cm/s backwards when its actually driving with a speed of up to 40 cm/s forward. However, this does probably not happen too often.

The program analyses one picture at a time. But the time stamp used for each iteration is taken from the system environment. Therefore it is a small delay between when the image is taken and when the time stamp is received. The camera has capabilities to mark timestamps in the image. Therefore a stronger connection to the time and each image is obtained and a more accurate speed estimation could be done. However, the project group also believes that the problem with the noisy velocity measurement could be solved by using a Kalman filter for the measurements, instead of further investigating the cause of the noisy behavior.

The problem with slow reactions in corners may be solved in a few different ways but they all have side effects. The derivative parameter for the steering could be increased but that causes problems along other parts of the track and can cause the vehicle to lose traction when turning faster at high speeds. The minimum distance to a reference point may also be increased to give the car more time to react but that will cause the vehicle to cut corners. There is also a physical limit for how tight the vehicle can take a turn. The measured steering angle of 30 ° could sometimes be smaller when the car is not trimmed and also varies a bit between the two cars.

Going to fast in the corners can be solved by lowering the speed reference earlier in the track but that has not been done yet. This would probably just solve the problem in one specific speed. The main problem is the inability to achieve and hold a reference speed.

The PID controller that is used is slow and has a big integral part. This results in a slow response on quick changes as in sharp corners. It has been tested to use a bigger derivative part to try to catch these quick changes in order to lower the control signal quick. However, the car will not drive smooth on straights then because of to much reaction due to the noise in the velocity measurements.

Having the speed as a part of the track might seem as an odd solution. Another way of doing it would be to continuously calculate the optimal speed. But that would require something like a MPC that in turn requires a better mathematical model of the car. The cars performance is sensitive to dust. Both the tires and the carpet needs to be cleaned often to get best performance out of the platform.

The platform considers the vehicle as a black box where the velocity is set by changing the voltage on the transmitter with a linear function. If the estimated speed is lower then the desired speed it will result in a higher voltage and vice versa. However, this is a simplification. Firstly, the voltage that is applied to the transmitter is decoded in an unknown way. This decoded data is sent wirelessly by radio waves to the vehicle. The cars interprets them as different duty cycles in a PWM signal, which will result in different angular velocity on the DC motor of the car. A change of the angular velocity will result in an acceleration of the car. Which after some time should result in the desired velocity for the controller. There is obviously some dynamics between the voltages of the transmitter and the velocity of the vehicle that is not taken in to account. This should be investigated further in order to get better results out of the controller.

In order to verify that the speed measurements of the vehicle give accurate data in the platform, the result from following this circle (shown in Figure 31) has been analyzed. The analysis was done by measuring the time it took for the vehicle to travel one lap. From the analysis it has been noticed that the velocity measured in the platform is not accurate to the real world. However, the velocity measured seems to have the right dynamics corresponding to the real velocity. No investigation has been done to determine if the platforms data is correct or if the results from the investigation are wrong due to lack of time.

7.1.4 Automotive algorithms

The algorithm for overtaking is simple. It just redraws the track around the other vehicle or object. A better and more advanced way, in the case of two vehicles, would be to have communication between the cars and together decide how the overtaking should take place. It would also be good to make sure that the track is suitable for overtaking so that it is not executed in sharp corners or cause the vehicles to drive too far away from the center line of the track. Just like in real world when you catch up with a vehicle in front you can not directly overtake the vehicle. You obviously need to look ahead and see how the road changes before you can make the overtaking.

The platooning algorithm is not complete. The biggest problem in the current implementation is the constant distance error. This problem could be solved by increasing the integral gain in the PID controller that controls the distance between the vehicles. Considering the size of the error this approach might cause other problems in forms of overshoot that results in a crash. This approach has been tried in simulations without good results. The calculation of the distance between the vehicles might not be the best since it creates unwanted errors in a track with sharp turns.

The system with Control Strategies should probably be modified to allow vehicles to have more then one. Now the overtaking strategy has its own `JustFollow` algorithm that is executed after the track is redrawn. This could be done in a more modular structure. For example, if the platooning algorithm should be able to avoid collisions then it would have to contain `Platooning`, `CollisionAvoidance` and `JustFollow` algorithms in it. With separate strategies all these behaviors could then be added to a vehicle separately.

7.1.5 Differences between simulation and reality

The initial simulations of the system where very promising, but as it turned out, the model did not represent the real world very well and the obtained control parameters had to be tuned. The cause of this is probably that the simulation is using a point mass model and the fact that no real system identification were made. The simulations were not entirely in vain though, as it showed that the chosen path following algorithm worked.

7.2 The future and environmental effects of automotive systems

The future of driving is in active automotive systems. Cars become smarter as sensors are improved, more computing power is made available and more automotive systems get implemented. The cars do not only become safer, they also become more energy efficient, and therefore more environmentally friendly. This is especially true when it comes to autonomous driving systems such as platooning, which reduces fuel consumption by reducing drag.

Even if the algorithms of this project or future projects will not reach the automotive industry it help students understand how important automotive algorithms are for the automotive industry's impact on the environment. Future projects could also develop algorithms that calculates the most efficient and energy saving way to drive the vehicles.

The environmental impact of this project is negligible. The electronics used in the project consume very little in the ways of power and most of the components are bought from Chalmer's suppliers, which hopefully have environmentally sound policies.

7.3 Further development

Even if the project has accomplished a lot there is room for more improvements of the platform. There are some areas that the project group recognise would need some extra attention. These areas are, in order of importance:

Speed and position measurements
The measured vehicle speed used in the feedback for the controller is a bit noisy and could be more stable with the help of, for example, a Kalman filter. The measurements could also be improved by using time stamps from the camera images. Right now the platform uses the time measured by the main program, which does not necessarily correspond to the time the picture was captured.

Acquiring a more realistic vehicle model
Having a more realistic model of the used vehicles would enable more advanced controls, such as model predictive controls. It would also make it easier to calculate controller parameter values and do simulations in MATLAB and Simulink.

Finding markers
Right now there is not enough light from the LEDs near the edge of the race area for the camera to detect the IR markers. In the middle of the race area it is too bright, and the reflections in the carpet interferes with the vehicle recognition. A more evenly distribution of light would solve most of these problems. Also, there is no correction for the lens distortion, which should be implemented to further improve the position measurements.

Control strategies
The MVP is created to simulate automotive algorithms and traffic situations. The implemented control strategies need more work to function properly. Additional automotive algorithms, such as intersection handling and speed optimization, could also be implemented.

8 Conclusions

The platform has been improved in almost every aspect and it should now be easy to continue its development. The IR markers and filter, the new camera and the new image processing forms the new positioning system. Together they make it possible to drive at higher velocities than before. The vehicles are now driving on a carpet on the floor, which is significantly larger than the table used in the previous project (Sjödin et al., 2014). This enables larger race tracks and makes it easier to test automotive algorithms on the race area. Signals can now be sent faster to the vehicles via the new DAC:s. All hardware for the communication between the computer to the vehicles are assembled on a mount for ease of use.

A control strategy to follow a trajectory has been developed, which is used by the vehicles to follow the race tracks in the platform. The strategy is working very well as long as the race track is not too complex. It is also used as a fundamental base for the automotive algorithms.

The expanded user interface lets the user tweak controllers and control strategies while the platform is running. Without these features, one would need to spend much more time waiting for the main program to compile for each change of parameter value. The development of automotive algorithms is also easier, since tweaking a parameter have an immediate visual effect on the performance of the vehicles.

Along with the development of the platform two automotive algorithms have been implemented, collision avoidance & overtaking and platooning. The collision avoidance & overtaking algorithm is unreliable, especially when faced with moving obstacles. It works as a proof of concept, but further development is necessary to get more stable results. The platooning needs a lot of work, the following vehicle does follow the leading vehicle but with an error margin and sometimes it loses contact to the leading vehicle. The Simulink model of the platooning does not represent the real system very well and therefore testing different control parameters or algorithms can be cumbersome.

There is a lot of potential for the platform even if there is still room for improvements. The project group believes that the platform provides a good starting point for future projects and hopefully upcoming projects can continue the development and solve the issues of the platform. Our recommendations are to further develop the velocity and position measurements, create a more realistic mathematical model of the cars and further develop the automotive algorithms.

References

Davila, A. (2013, January). Report on fuel consumption. Retrieved April 24, 2015, from http://www.sartre-project.eu/en/publications/Documents/SARTRE_4_003_PU.pdf

Hagebring, F., Hansson, K., & Waldemarsson, O. (2014). *Platform for robot swarm behaviour* (Bachelor Thesis, Department of Signals and Systems, Chalmers University of Technology).

Ljung, M., Fagerlind, H., Lövsund, P., & Sandin, J. (2007). Accident investigations for active safety at chalmers - new demands require new methodologies. *Vehicle System Dynamics*, *45*, 881–894.

Musk, E. (2014, October). Dual motor model s and autopilot. Retrieved May 19, 2015, from http://www.teslamotors.com/sv_SE/blog/dual-motor-model-s-and-autopilot

Nilsson, A., Björsell, J., Öhlund, L., Eriksson, M., & Bäck, V. (2014). Cars - camera-based autonomous racing system. Retrieved May 19, 2015, from http://cars.it.uu.se/projects/project-1/

ORCA. (2015, February). Retrieved May 19, 2015, from https://sites.google.com/site/orcaracer/home

Pololu. (2012). Pololu m3pi users guide. Retrieved from https://www.pololu.com/docs/pdf/0J48/m3pi.pdf

PTGrey. (2014, September). Flea3 usb3 datasheet. Retrieved February 27, 2015, from http://www.ptgrey.com/support/downloads/10136w?usp=sharing

PTGrey. (2015, May). Flycapture sdk. Retrieved February 27, 2015, from http://www.ptgrey.com/flycapture-sdk/

Rajamani, R. (2012). *Vehicle dynamics and control, second edition*. 233 Spring Street, New York: Springer.

Sjödin, C., Sampath, C., Diakou, G.-E., & Johansson, I. (2014). *Cooperative multi-robot systems: experimental platform for advanced vehicle control applications*. Department of Signals and Systems, Chalmers University of Technology.

TexasInstruments. (2001, November). Texasinstruments tlc5620 datasheet. Retrieved February 27, 2015, from http://www.ti.com/lit/ds/symlink/tlc5620.pdf

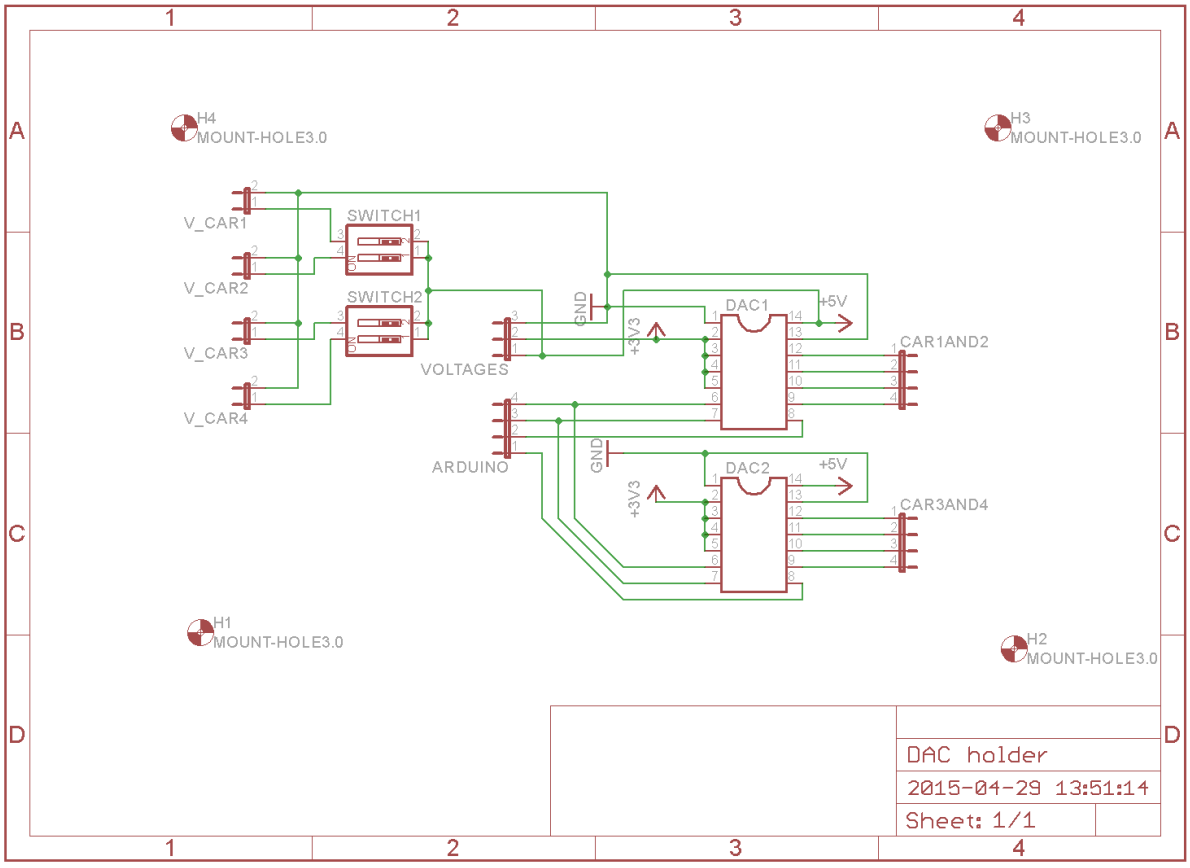
Appendices

A Inventory List

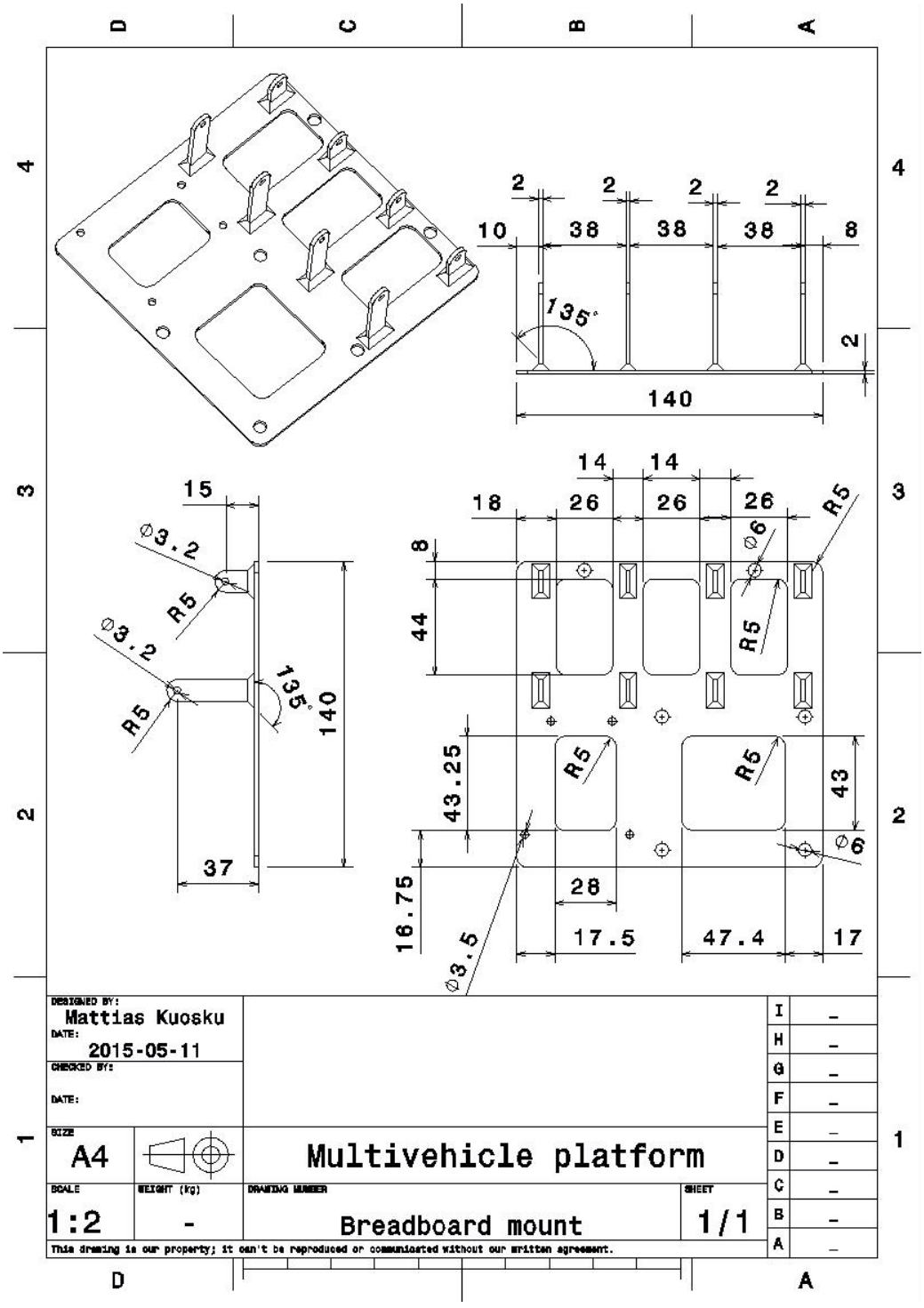
Table 4: The hardware used in the platform

Hardware	**Model**	**Quantity**
RC-car	1:43 scale Kyosho dNaNo	2
RC-chassis	Kyosho dNaNo Porsche Carrera GT	2
Li-Po-batteries	Kyosho dNaNo	4
Transmitter	Kyosho Perfex KT-18	2
Battery charger for Li-Po	-	2
Micro controller	Arduino Uno	1
DC-cable	-	1
USB-cable	-	1
IR filter	760nm, High pass filter	1
IR lamp	ILS 4 IR LEDs 850nm	1
Heatsink	LED Star Heatsink Kit 50x20mm	1
Constant Current LED Driver	ILS IZC035-017F-0067A-SA	1
IR markers (tape)	ifm electronic Reflective Tape	
Digtal to analog converter	Texas Instruments TLC5620CN	2
Camera	Point Grey Flea3 1.3 MP Mono USB3 Vision	1
Camera mount to the roof	DELTACO Stand For Surv. Cameras 13CM 1 Ball Join	3
Lens	GOYO GM24514MCN	1
Lens adapter for camera	3D-printed from CAD blueprints	1
Circuit board for DAC:s	PCB for simplifying the connections	1
Cicruit board holder	3D-printed from CAD blueprints	1
Rubber carpet	Smooth Rubber mat from Gerdman #12-768574 $1.4 \times 3.5\,\mathrm{m}^2$	2

B Schematic for PCB holding the DACs



C Drawing of the breadboard mount



D Drawing of the adapter