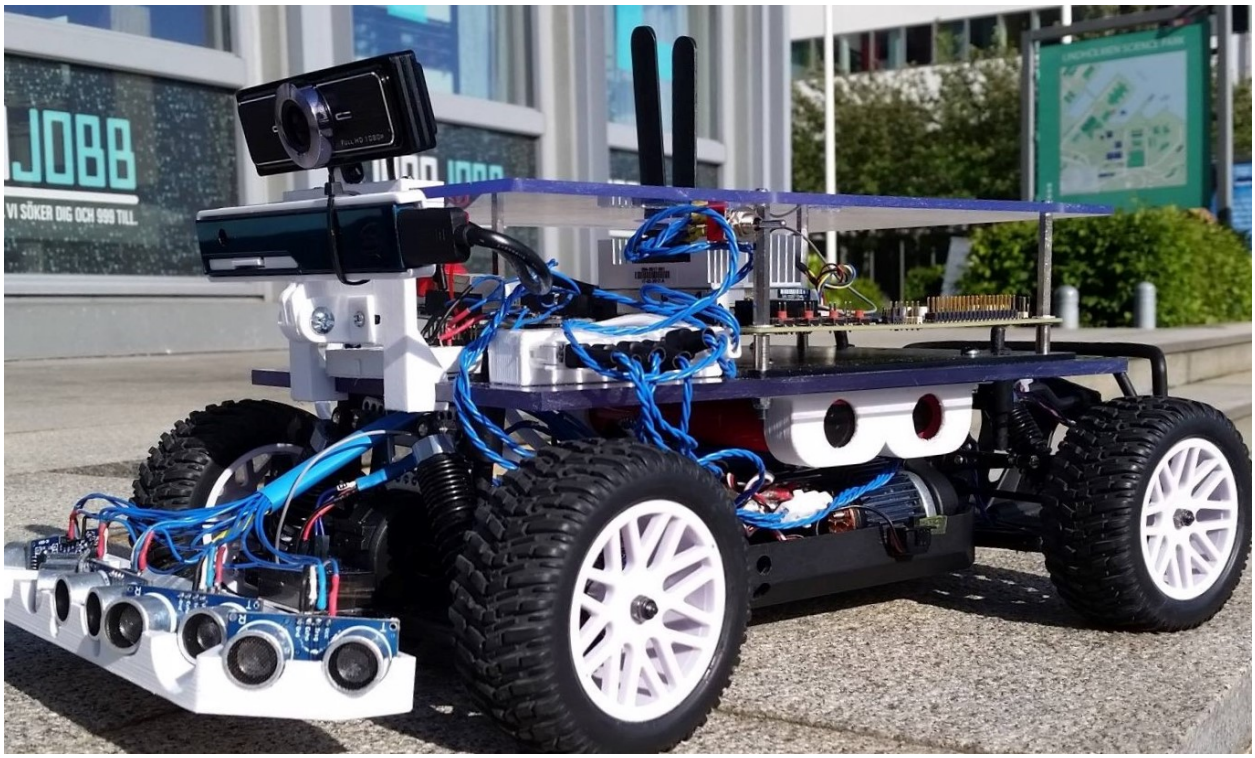
CHALMERS
UNIVERSITY OF TECHNOLOGY



Controllers and their implementation for an autonomous vehicle

Bachelor's thesis in mechatronical engineering

Andreas Jakobsson and Anders Sundkvist

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Bachelor's thesis 2017

Controllers and their implementation for an autonomous vehicle

Control algorithms, feedback systems, power supplies and hardware mounting

Andreas Jakobsson

Anders Sundkvist

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2017

Controllers and their implementation for an autonomous vehicle
Control algorithms, feedback systems, power supplies and hardware mounting
Andreas Jakobsson
Anders Sundkvist

© Andreas Jakobsson 2017.
© Anders Sundkvist 2017.

Supervisor: Alixander Ansari, Sigma Technology
Examiner: Bertil Thomas, Signals and Systems

Bachelor's Thesis 2017
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 (0)31 772 1000

Cover: The final appearance of the autonomous RC car
Gothenburg, Sweden 2017

Controllers and their implementation for an autonomous vehicle
Control algorithms, feedback systems, power supplies and hardware mounting
Andreas Jakobsson
Anders Sundkvist
Department of Signals and Systems
Chalmers University of Technology

Abstract

This project is aimed to in the end present a scale model of an autonomous car. Since the automotive industry is very rapidly making enormous progress in the field of autonomous driving, the technology companies involved also needs to keep up with the pace of the big actors on the market.

Sigma Technology is one of these companies that are very much involved in the automotive industry and the issue of autonomous driving has shown to be of much interest for them. Because of this the company has decided to give a team of students the task of investigating this subject. The team consists of students with backgrounds within product development, computer science and mechatronics. This provides the project with a wide base of knowledge and also enables the result to be more realistic.

This report will handle the issue of the controllers and their implementation for an autonomous vehicle, this includes the cruise controller and the steering controller. The cruise controller is designed by using the method of step response analysis and has resulted in an accurate and stable controller. The steering controller is designed by using a method called pure pursuit, this is a geometrical relation between the wheel angle and an point on a set distance ahead of the vehicle. This method is combined with the ackermann steering principle that the vehicle is based on and by that combination a steering controller is achieved. These controllers where then handed over to the computer science students that integrated them with their work on neural networks and computer vision.

Keywords: Autonomous system, Pure pursuit, Cruise controller, Step response, PI-controller, radio controlled (RC) car.

Abbreviations

- ESC, Electronic Speed Controller
- RC-car, Radio-controlled car
- PID controller, Proportional, Integral, Derivative controller
- PWM, Pulse Width Modulation
- DC, Direct Current
- CHEVP-algorithm, Canny Hough Estimation of Vanishing Points
- N/A Not available

Acknowledgements

We would like to appreciate Sakib Sistek, Göran Hult and Bertil Thomas for their expertise, support and advice along the timespan of this project. Sigma Technology and our supervisor Alixander Ansari also deserves a big thank you for their support and also for giving us the opportunity of writing our thesis in their facilities.

We would also like to say thank you to the rest of the project team, Vishnu Shyam, Paul Lazar, Daniel Posch, Jesper Rask, Albin Falk and David Granqvist for their part in developing the RC-car and their know-how in programming and project management.

Andreas Jakobsson and Anders Sundkvist, Gothenburg, June 2017

Contents

1
Introduction

1.1 Background

The reason for this project to be executed is the current development of different technologies for autonomous vehicles. The majority of all technology companies is in one way or another involved in this development, and the same goes for Sigma Technology that has provided us with the funding and resources for this project. Sigma Technology is a consultancy company located at Lindholmen, Gothenburg and is one of many departments in Sigma AB. The project is aiming towards modifying a radio-controlled (RC) car so that it will resemble an actual autonomous car as much as possible.

The whole project consists of eight students with different backgrounds to cover as many aspects as possible. There are two master students from the Product Development department, four bachelor students from the Computer Science department and the two authors of this report are bachelor students from Mechatronics Engineering.

1.2 Work distribution

The work within the project has been distributed so that each division handles the tasks related to their field of expertise. The Product Development students will play the role as project coordinators for the project. This means to investigate what the different stakeholders such as Sigma Technology, the industry itself, the consumers and what society are expecting from an autonomous vehicle and what part of these wishes would be possible to implement in this project. From these expectations a requirements list will be provided. This requirements list can be found in Appendix A and is what the rest of the group will work against.

The Computer Science students will play the part as the software team, this division will split in two different subdivisions. One subdivision will be "computer vision" and the other will be "neural networks". The computer vision team will construct the lane tracking and object recognition that will supply the autonomous system with the information needed to be able to stay on the road without crashing in to obstacles. The neural networks team will handle the decision making part of the autonomous system. By using all the inputs from the computer vision and other sensors as inputs for the machine learning part. The system will by itself be able to

make the most suitable decision for each given situation.

We, the Mechatronics students, will handle all issues concerning the hardware. The biggest task will be to construct the algorithms for a cruise controller that maintains a desired velocity and a steering algorithm that meets all the requirements provided. The hardware team will also mount everything on the vehicle and make sure all components are powered in the way they are supposed to and with the correct voltages.

1.3 Purpose

The purpose of this thesis is to supply the software team with a set of control algorithms and also make sure all components are powered in the correct way. The sensors of the car and their placement will also be our issue to handle. All components needs to be mounted in a safe and stable manner, this will be done according to the requirements list. The following steps will be executed in order to reach the goal:

- Choosing and implementing a suitable steering control algorithm
- Choosing and implementing a suitable cruise control algorithm
- Choosing and implementing suitable sensors and their placement
- Estimate the power consumption and from that choosing the best power supply option
- Plan the distribution of the components and mounting them in a suitable way

1.4 Limitations

In the project there has been clear requirements set, this means that the different environments that the system will be able to handle is limited. These limitations is clearly stated in the requirements list in Appendix A.
There is also a budget set from Sigma Technology at 10.000 SEK. This includes everything from buying the actual RC-car to each resistor being used. The time that will be spent on the project is also limited, since there is two teams doing their bachelor's thesis and one team doing their master's thesis the project needs to be completed in mid June 2017.

2

Theory

This chapter will describe all the theory behind this project and also list all the hardware components.

2.1 Hardware components

The following components are used to complete this autonomous driving system.

- RC-car, Maverick Strada SC Evo 1/10 Electric Short Course
- Embedded computing device, Jetson TX2
- 3D camera, Intel Realsense R200
- Wide angle camera, Genius WideCam F100
- Single-board microcontroller, Arduino Mega
- Ethernet shield, HanRun compatible with Arduino Mega
- Extra batteries, two 7.2V, 3000 mAh, NiMH batteries
- Hall effect sensor
- Ultrasound sensors, HC-SR04

2.2 Autonomous steering

In order to achieve autonomous steering that is acceptable as a replacement for a human driver the steering needs to be as smooth, and preferably smoother than the steering of an experienced driver. These terms demands that the steering controller is performing very well under a variety of conditions. The following chapter will bring up a number of steering methods that could be used for autonomous drive.

2.2.1 Ackermann steering

The vehicle in question is based on a steering method called Ackermann steering. This geometrical setup of linkages gives the outer wheel a smaller angle than the inner wheel when turning. This is to prevent the tires from slipping when the vehicle is turning. This steering design can be explained by a simple equation that can be obtained from the figure on the next page. [1]

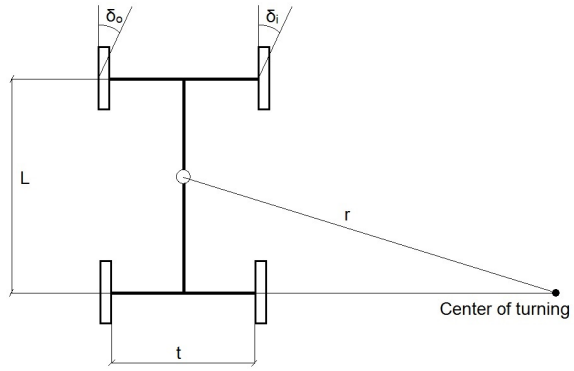


Figure 2.1: Ackermann steering geometry. r is the turning radius, δ_i and δ_o are the wheel angles of the inner and outer wheel, t is the track width and L is the wheelbase of the vehicle.

This gives the equations for the wheel angles as following:

$$\delta_i = \arctan\left(\frac{L}{r + \frac{t}{2}}\right) \tag{2.1}$$

$$\delta_o = \arctan\left(\frac{L}{r - \frac{t}{2}}\right) \tag{2.2}$$

2.2.2 Pure Pursuit

The pure pursuit method is based on a few geometrical properties that connects a point on the desired path with a point on the vehicles path. The path of the vehicle is calculated as a circular arc depending on the wheel angle of the vehicle. The point on the desired path is set to be at a certain distance ahead of the car, this distance is often called the lookahead distance. A human driver is also looking at a certain distance ahead of the car and then adjusting the steering wheel to reach the desired point. Just a moment after a new point is chosen and the procedure is repeated, this behaviour is what the pure pursuit method is mimicking.[2]

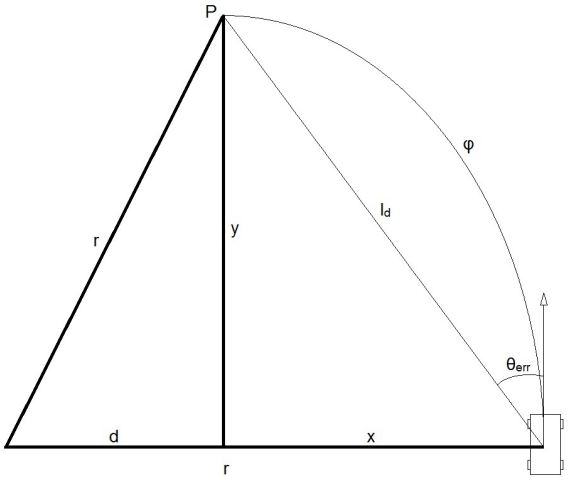


Figure 2.2: The geometrical definitions of pure pursuit. φ is the curvature of the circle that the car needs to follow, θ_e is the error angle between the desired path and current heading, r is the radius of that circle, l_d, x, y and d is auxiliary variables

By using the definitions from Figure 2.2 the circular arc needed to reach the desired point P can be calculated.[3]

$$r = x + d \tag{2.3}$$

$$sin(\theta_e) = \frac{x}{l_d} \Rightarrow x = sin(\theta_e) l_d \tag{2.4}$$

Pythagoras equation gives:

$$x^2 + y^2 = l_d^2 \tag{2.5}$$

and

$$d^2 + y^2 = r^2 \tag{2.6}$$

These equations gives:

$$(r-x)^2 + y^2 = r^2 \Rightarrow r^2 - 2rx + x^2 + y^2 = r^2 \Rightarrow x^2 + y^2 = 2rx \Rightarrow 2rx = l_d^2 \tag{2.7}$$

$$r = \frac{l_d^2}{2x} \tag{2.8}$$

From the definition of the curvature of a circle:

$$\varphi = \frac{1}{r} \tag{2.9}$$

The circular arc that the vehicle will perform at the set lookahead can be calculated:

$$\varphi = \frac{1}{r} = \frac{2x}{l_d^2} \tag{2.10}$$

From equation 2.6 we obtain:

$$\varphi = \frac{2 l_d * sin(\theta_e)}{l_d^2} \Rightarrow \varphi = \frac{2 * sin(\theta_e)}{l_d} \tag{2.11}$$

This equation gives the mathematical expression for the curvature of the circular arc that the vehicle needs to follow to reach the goal point.

2.3 Step response analysis

To determine the transfer function for a system, step response analysis is a simple and effective method to use. By exposing the system of a step formed input signal and recording the output that the system gives, a step response is obtained. By then comparing the appearance of the given output with tables from literature the type of process can be determined. The type of process is then linked to a certain type of transfer function.[4]

2.4 Feedback systems

To obtain accurate control over any system a feedback systems is absolutely essential. This chapter will describe how the feedback systems for the vehicle are constructed and what they do.

2.4.1 Cruise controller

To be able to determine the longitudinal velocity of the vehicle, a feedback system must be implemented. This system was constructed with a Hall effect sensor, a sensor that gives an electrical pulse when it detects a magnetic field in its proximity, and a number of magnets. The method of calculating the speed can be done in two ways, either by counting the number of pulses during a specific time span and then converting this accordingly:

$$Speed = \frac{Number\ of\ pulses}{Time\ span} * Distance\ between\ two\ magnets \tag{2.12}$$

This method is more suitable for higher velocities since the slower the vehicle moves the fewer pulses can be counted, and if the sensor then fails on reading a pulse the error will affect the output immensely.

The other method is instead of counting the number of pulses, count the time between two pulses. This showed to be more effective at lower velocities, this is because of at greater velocities the measured time will be very short and therefore accurate measuring devices are required.

$$Speed = \frac{Distance\ between\ two\ magnets}{Time\ between\ two\ pulses} \tag{2.13}$$

2.4.2 Steering feedback

The steering feedback is basically the vision system including the camera. By using the trajectory algorithm supplied by the software team, the offset from the trajectory can be calculated and then used as a feedback signal to the steering system. The algorithm is called CHEVP-algorithm which in short produces multiple midpoints between two parallel lines represented by the side lines on the road. Further reading about this can be found in the Bachelor's thesis written by Jesper Rask and Daniel Posch.[5]

3
Method

3.1 Initial work

To obtain the knowledge of what signals the Electronic Speed Controller (ESC) and steering servo needed to function a oscilloscope was used to probe the receiver for the hand controller. This gave us the range of the Pulse Width Modulated (PWM) signal sent from the hand controller. To test this out an Arduino Mega was used to replicate the signals, this step gave us full control of the car.

3.2 Propulsion controller

To obtain autonomous driving a cruise controller that maintains the desired speed is necessary. The chosen way of doing this was to construct a conventional PI-controller. This meant to calculate the transfer function for the system and through that dimension a suitable controller.

The first step was to mount a sensor that could record the velocity of the wheel, to do this a Hall effect sensor was used. By mounting a 3D printed plastic ring with multiple cavities in one of the back wheels. Then fitting small magnets in the cavities and counting the time between each pulse and translate this time to actual longitudinal velocity using equation 2.13 a velocity feedback system was obtained. Although the data in Figure 3.1, 3.2 and 3.3 is collected with equation 2.12 hence the very noisy output. But by executing the large number of runs and then using the mean value of these a workable curve is obtained anyway.

This feedback system enabled velocity data to be collected. By using an Arduino Mega and a memory card to record the data and then importing the files to MATLAB, a number of plots where obtained. From these plots a transfer function was calculated by using the method for step response analysis. To get reliable values a large amount of test runs was executed, as shown in the figure below. As mentioned earlier the requirements states that three different speeds will be relevant. Therefore steps where performed between these speeds, and every step was executed twenty times to get reliable data and prevent large deviations.

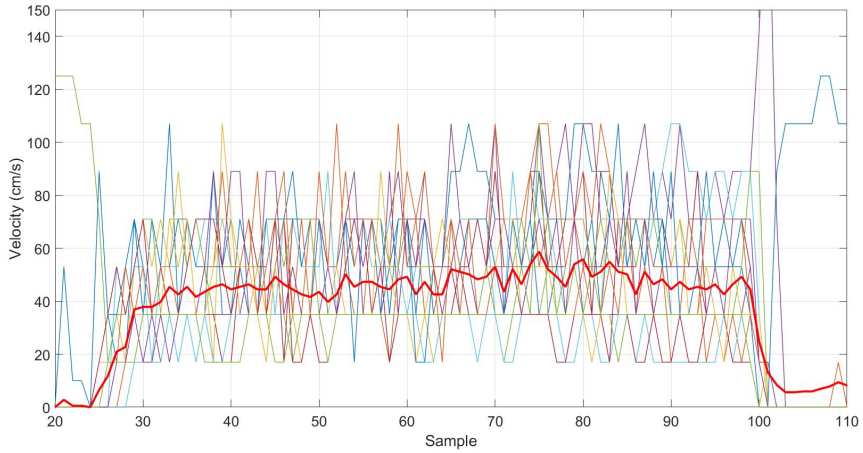


Figure 3.1: Data collecting: From standstill to parking speed. The red line represents the mean value of all curves

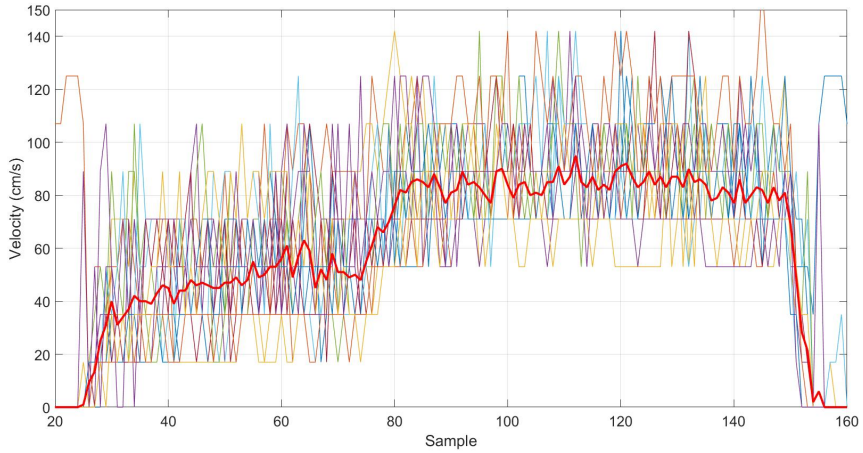


Figure 3.2: Data collecting: From parking speed to city speed. The red line represents the mean value of all curves

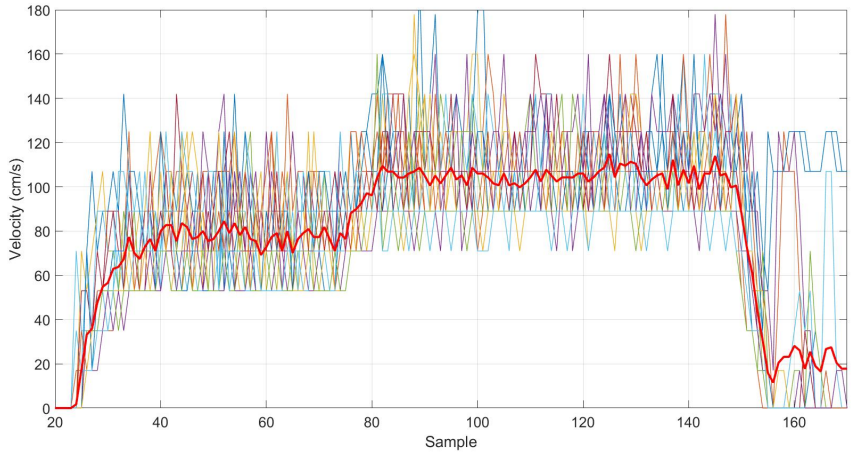


Figure 3.3: Data collecting: From city speed to highway speed. The red line represents the mean value of all curves

By plotting the mean values of the collected speed data from the steps that was performed by the system, the step responses in Figure 3.4 was achieved.

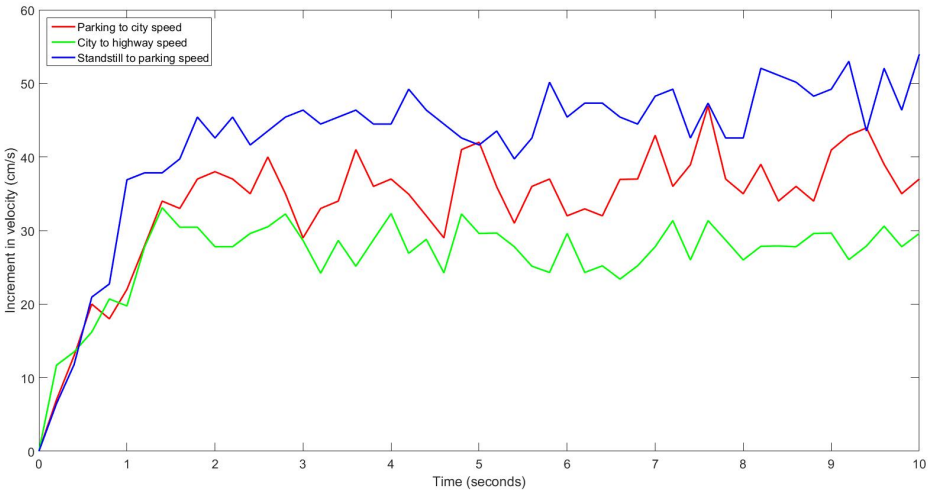


Figure 3.4: Step responses, standstill to parking speed, parking to city speed and city speed to highway speed.

From Figure 3.4 the system can be identified as a system with one time constant and therefore the transfer function for the systems is described as:

$$G(s) = \frac{K}{1 + T * s} \tag{3.1}$$

Where:

$$K = \frac{\Delta y}{\Delta u} \tag{3.2}$$

$$\Delta y = y_{final} - y_{intial} \tag{3.3}$$

$$\Delta u = u_{final} - u_{intial} \tag{3.4}$$

T is the time to reach 63% of max output value, u is the input and y is the output.

By comparing "parking speed to city speed" with "city speed to highway speed" where both steps are 20 units it is clear that the system is not linear. This means that the system behaves differently in certain condition, but since the most critical speed is from "standstill to parking speed" the transfer function is calculated based on this data.

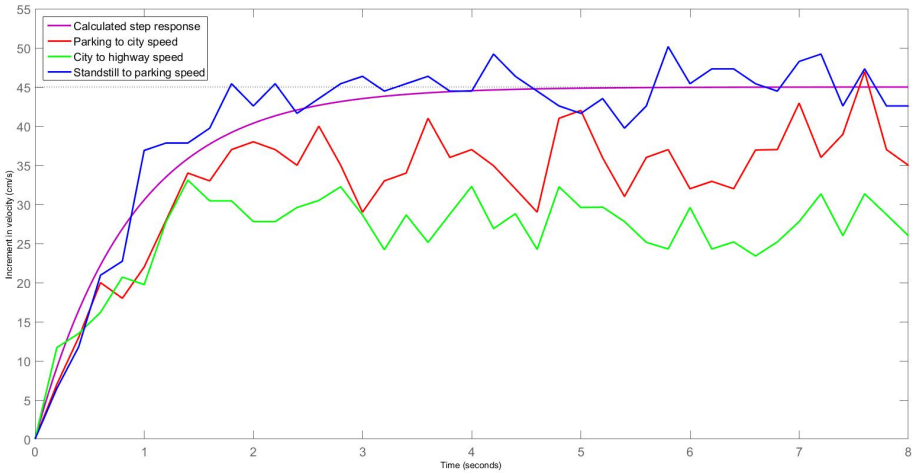


Figure 3.5: Step responses and transfer function for standstill to parking speed, G(s).

In Figure 3.5 the transfer function is shown based on Equation 3.1 and the recorded data from the test runs performed from "standstill to city speed".

To easily obtain suitable controller algorithm parameters, Simulink was used and a model of the system was constructed to simulate the real world. By adding Simulinks PID-controller block and then using the tuning option. This generated parameters for a fast and robust cruise controller.

3.3 Steering controller

It was found that there is a large number of different control algorithms for autonomous steering of a vehicle. Many of them are very mathematically complicated, such as the vector pursuit. Vector pursuit is based on the theory of screws.[6] This method was chosen to not proceed with because of its complexity. Another method that was considered but also rejected because of its complexity is Nested PID steering control.[7] This method uses the yaw rate measured by a gyroscope and the lateral offset from the desired path and the current heading. Nested PID steering control showed also to be a very complex matter, and since the time for the project is limited the method was discarded. The methods that where the most interesting was the ones based on geometrical calculations. This because of the simplicity of their nature when calculating and implementing them.

The pure pursuit method was found to be a suitable option for a controller method. By linking the wheel angle from Equation 2.1 and 2.2 with the curvature from Equation 2.11 the expression for how the wheel angle acts as input for system to reach the desired path. To simplify this even further the approximation of substituting the four wheel vehicle with a two wheeled vehicle was made. This means that Equation

2.1 and 2.2 can be merged into just:

$$\delta_o = \delta_i = \delta = \arctan\left(\frac{L}{r}\right) \tag{3.5}$$

Just as in the pure pursuit case, the curvature of the circular arc that the vehicle will perform for the given wheel angle can be calculated with the definition of the curvature of a circle:

$$\varphi = \frac{1}{r} \tag{3.6}$$

Equation 3.5 can be rewritten as:

$$r = \frac{L}{\tan(\delta)} \tag{3.7}$$

Which gives the curvature:

$$\varphi = \frac{1}{r} = \frac{\tan(\delta)}{L} \tag{3.8}$$

Now the pure pursuit can be integrated with the mechanical properties of the vehicle by taking the two curvatures and make the wheel angle the input:

$$\varphi_{pure\ pursuit} = \varphi_{ackermann} \Rightarrow \frac{2sin(\theta_e)}{l_d} = \frac{\tan(\delta)}{L} \tag{3.9}$$

Equation 3.9 then gives the wheel angle as a expression of the lookahead distance l_d, the error angle from the current heading and the desired path θ_e and the wheelbase L:

$$\delta = \arctan\left(\frac{2 * L * sin(\theta_e)}{l_d}\right) \tag{3.10}$$

Now the only input needed for a simple and smooth control algorithm is the error angle θ_e which is supplied from the CHEVP-algorithm created by the software team.

3.4 Power supply

To supply the hardware with the correct voltages, the components power consumption had to be determined. By studying the specification sheets for the Jetson TX2, the Arduino Mega, the camera, the sensors and the vehicle itself the total power need was calculated. The biggest concern was the Jetson TX2 board, since according to the specifications the board can only handle voltages between 5,5 and 19,6 volts.[8] The only information about the power consumption found for the board is from different forums online, these forums stated that at full load on the board draws 75 watts. The Arduino with an ATmega chip draws 0,2 mA and an additional 20 mA per I/O pin, this means that the Arduino needs approximately 0,1 A and runs on 5 volts which is the same as 0,5 watts.[9] With this knowledge we know that the additional components, excluding the vehicles standard servo motor and propulsion motor, draws about 75,5 watts at maximum load.

The different approaches that we found was either buying a large powerbank that could supply the components with the correct power. A project made by Jetson-Hacks called RACECAR[10] used this method, the problem with this method is our budget. The powerbank with their quality and performance would make us exceed our budget, to avoid this other methods of powering the components was investigated.

What was found was that a decent capacity could be obtained by connecting the already obtained RC-car batteries included with the car in series. By connecting two batteries in parallel the capacity is doubled, if the batteries instead are connected in series the output voltage is doubled and the capacity stays the same.[11] The method of parallelly connecting two batteries with a capacity of 7,2 volts and 3000 mAh, a capacity of 6000 mAh was achieved. This is equivalent to 75,5 watts for approximately 30 minutes. Which is an acceptable according to the requirements.

$$\text{Current needed} = \frac{\text{Power needed}}{\text{Voltage from the battery}} \tag{3.11}$$

$$\text{Duration} = \frac{\text{Current needed}}{\text{Capacity of battery}} \tag{3.12}$$

The above is theoretical run time and the absolute maximum wattage used.

3.5 Ultrasound sensors

The ultrasound sensors used as a redundancy system for the vision system is mounted in a way that the sensors cover the area specified in the requirements. Figure 3.6 represents the field of view that is given by the requirements list, from this the ultrasounds are mounted to replicate the coverage that is desired.

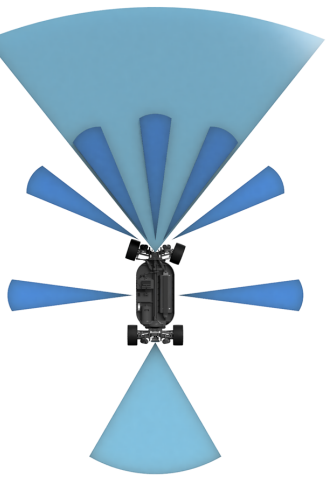


Figure 3.6: A representation of the field of view needed from the ultrasound sensors and the camera to meet the requirements

3.6 Test and verification

This chapter will describe and display the tests made during the project.

3.6.1 Cruise controller

The cruise controller was tested by giving the system various set point values and then recording the actual velocities that the vehicle reached.

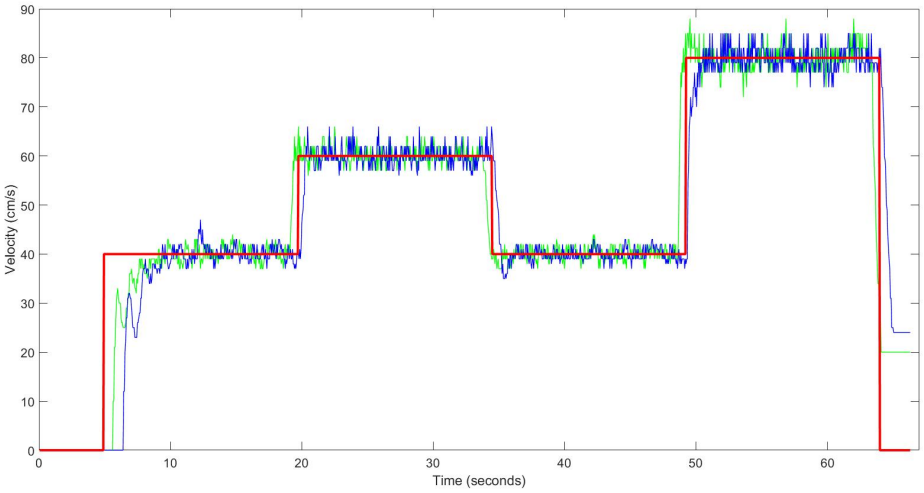


Figure 3.7: The cruise controller is given a setpoint value for 15 seconds and then the setpoint changes

The plots in Figure 3.7 is the verification of the performance of the cruise controller. It can be seen that the controller reaches the desired velocity in approximately 2 seconds when starting from a standstill. The rise time is 1 seconds when increasing and decreasing the speed when already in motion.

The controller was also tested by letting the vehicle run a carefully measured distance and measuring the time the vehicle takes to run that distance. This verifies that the controller actually maintains the desired velocity in real life and not only is calculating that it runs in the desired velocity. The table below shows the accuracy of the controller at different setpoint velocities.

Table 3.1: Verification of cruise controller

Setpoint [cm/s]	Mean time [s]	Actual velocity [cm/s]	Error
40	25,404	39,4	1,59%
60	16,994	58,8	1,93%
80	12,8528	77,8	2,74%

When comparing the finished cruise controller with the first feed back system that can be seen in Figure 3.1, 3.2 and 3.3. It clearly shows the improvement of using the method from Equation 2.13 instead of Equation 2.13 where the result is really noisy and inaccurate. The results of the finished cruise controller can be seen in Figure 3.7.

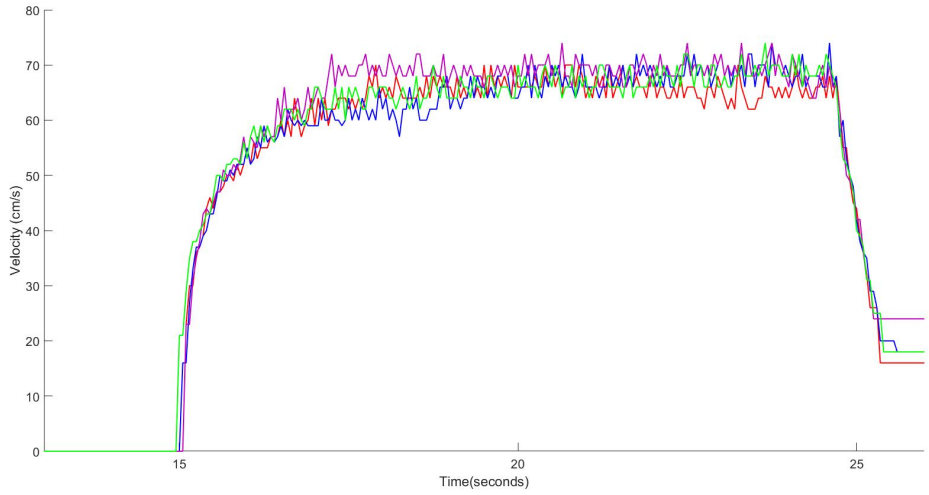


Figure 3.8: Data collecting: From standstill to parking speed.

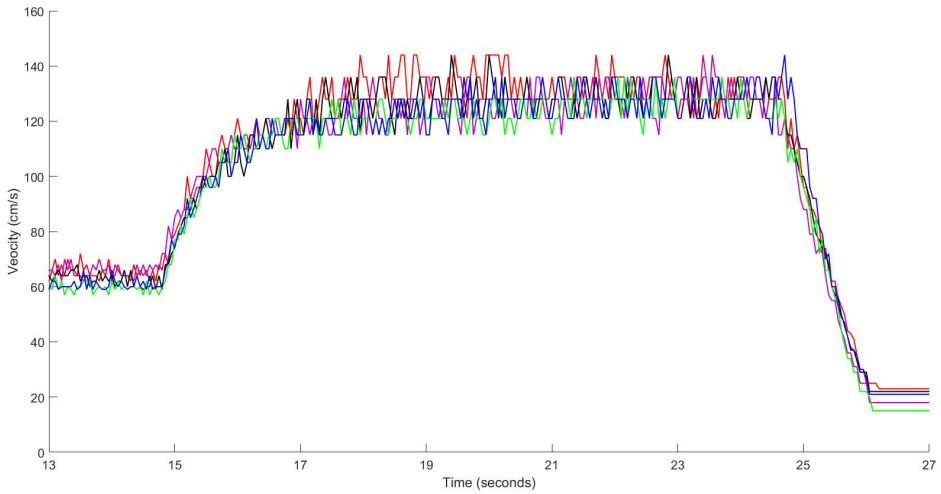


Figure 3.9: Data collecting: From parking speed to city speed.

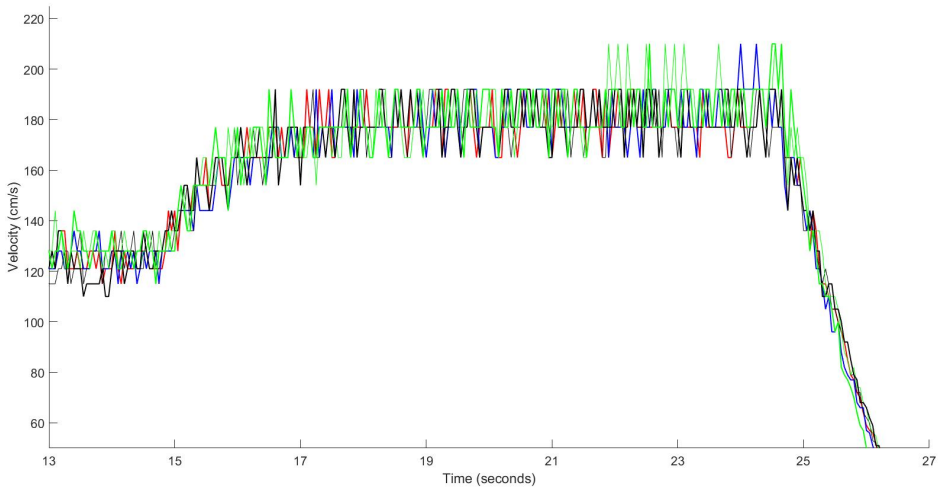


Figure 3.10: Data collecting: From city speed to highway speed.

3.6.2 Steering controller

The steering controller was tested by giving the error angle as input. Then verifying that the error angle translated to the wheel angle corresponds to the turning radius from Equation 3.9. The table below shows the result of the tests, an error angle of 30° is not applicable on the steering system. This because of the limitations of the steering servo that can just only deliver a steering angle between 25°and 30°.

Table 3.2: Verification of steering controller

Error angle [degrees]	Calculated turning radius [m]	Actual turning radius[m]	Error
5	2,868	2,535	-11,60%
10	1,4397	2,155	49,70%
15	0,9659	1,055	9,20%
20	0,73095	0,71	-2,80%
25	0,59155	0,555	-6,20%
30	0,5	N/A	N/A%

4

Results

The desired result for this project was to supply the software team with two control algorithms that would enable the vehicle to steer, accelerate and decelerate in a smooth and controlled way. When all steps that are described in the previous chapters were executed this was achieved.

When verifying the cruise controller it was found that the accuracy of the controller is less then +2% for slower speeds. At higher speeds the accuracy is a bit worse, but this is expected since the method of how the velocity is measured is adapted for low velocities. This performance proved to be acceptable to the software team and their requirements on how accurate the cruise controller needed to be.

The steering control algorithm when verified showed to be not very accurate, this due to play in the hardware components. As shown in the table in the subchapter 3.5.2 the accuracy of the controller is fluctuating very much, but at a larger error angle the accuracy improves. This improvement originates from the servos capability of moving the steering linkage, when a small wheel angle is needed the servo is set to only move the steering linkages a very small distance. At these small movements the play affects the outcome a lot. When all of this is accounted for, the steering controller is still suitable because ff the improvement in accuracy and also the simplicity and easy implementation.

The power supply for the system resulted in two 7,2 volt NiMH batteries with 3000 mAh each connected in parallel. This setup will supply the system in a stable way and for a long enough time.

The sensors utilised on the vehicle are the ultrasound sensors, the Intel Realsens R200 camera and the Genius WideCam F100. The placement of the camera was based on the software teams requirements for the CHEVP-algorithm. It was chosen so that enough of the side lines on the road i visible and also enough road ahead is visible.

The resulting placement of the ultrasound sensors is shown in the figure below.

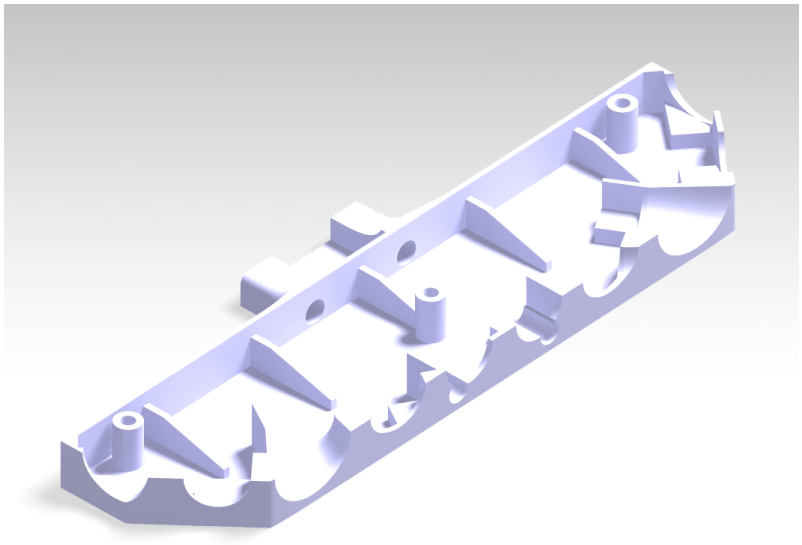


Figure 4.1: The 3D printed bracket made to hold five ultrasound sensors

Figure 4.1 is a 3D printed mount that is replacing the front bumper on the vehicle. To be able to get the appropriate coverage five ultrasound sensors were implemented and mounted on the 3D printed bracket.

5
Conclusion

5.1 Problems

In this section we will bring fourth some of our more severe and difficult problems we encountered during the project.

5.1.1 Cruise controller

Since all tests regarding the cruise controller were done using the car's on-board battery, it is possible that the results varies. This is due to the fact that when the battery is fully charged it can supply the car with more power, therefor the car has more torque in the beginning of our tests, resulting in faster acceleration and higher speed. Another factor that affects the testing is that due to lack of space different locations for testing were used. Different locations means that the surface differs and therefor traction and rolling resistance as well.

During the testing phase for the cruise controller we found that, when we applied torque to the steering servo, the car took of at full speed for no apparent reason. We discussed the problem and arrived at that the problem should lie somewhere in the signal sent to the ESC. By connecting our oscilloscope to ground and the signal wire, we found a slight deviation in the signal. The deviation is the small voltage increase right after the square signal with a amplitude of 0,2 volts and a duration of approximately 2100 μs, which corresponds with what the ESC recognises as "full throttle", see Figure 5.1. We found that when the steering servo tried to resist the applied torque it drew to much current for the Arduino to handle which was the reason for this behaviour. We solved this by connecting the servo to a separate power supply.

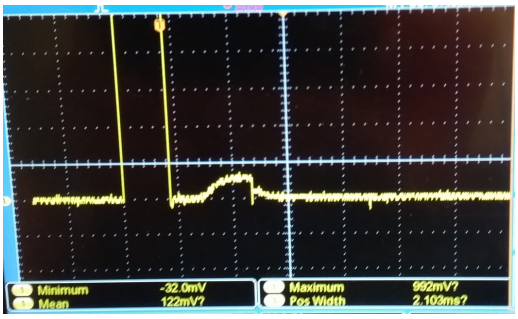


Figure 5.1: Control signal to the ESC and the recorded noise.

5.1.2 Steering controller

The steering controller is despite the very low accuracy when testing, working pretty good. When considering the earlier mentioned play in the steering system, the approximation of assuming that the four wheeled vehicle behaves as a two wheeled vehicle and the fact that the adjustment of the wheels on the RC-car is not optimised. The controller performs well enough to be used in the autonomous system.

5.1.3 Batteries and power supply

In the beginning we had big issues with getting the TX2 to run on batteries, even though the voltage of the batteries was correct when measured with a voltmeter. After some extensive troubleshooting we found that the two batteries that we were using was faulty. By measuring the internal resistance of the batteries we saw a resistance of 1 ohm, which leads to a voltage drop when current is drawn from the battery. Ohm's law gives that at 1 ampere the voltage drop is 1 volt. The TX2 wants 15 watts in idle which equals 2 amperes at 7,2 volts. When loaded the TX2 can draw up to 75 watts, so due to the voltage drop in the batteries it wont be able to do any processing with these batteries. A new set of batteries were purchased.

5.2 Further development and improvements

The results of our part in the project of constructing an autonomous system for a RC-car are pleasing, but there are a lot of improvements that can be done. Both on the cruise controller and the steering controller.

5.2.1 Cruise controller and its feedback system

Since the car does not behave equally in every speed interval a cruise controller designed to use more then one transfer function would make the controller perform better at a wider span of velocities. Also as we built our own pulse sensing device to measure the velocity of the car there are a few flaws with it in terms of accuracy. An example is that the distance is not equal between each magnet mounted inside the wheel and therefor the measured time between pulses are effected. By buying a commercial decoder that can count pulses or the time between pulses, the accuracy could have been improved.

What also would improve the cruise controller is measuring the velocity on more then one wheel. As of now it is only done on one wheel, this means that if the car is turning, that wheel will either be the inner or the outer wheel. If it is the inner wheel it will spin much slower than the outer wheel, which in turn will mean that the actual velocity of the car will increase since the controller thinks the car has slowed down. If the measuring is done on more then one wheel this problem could be avoided.

Another interesting feature that could be further developed when measuring the velocity on all four wheels is something called dead reckoning. This is a method of navigating without any external information being sent or received. By knowing the velocity of each individual wheel, the heading can be calculated and how far the vehicle has travelled from its original position.[12] This could be implemented as a navigation method for a robot that only operates in a controlled environment, for example a laboratory where samples needs to be transported and human presence could be compromising the work.

5.2.2 Steering controller and its feedback system

The steering controller has some interesting points that could be developed. Just as the cruise controller, an adaptive parameter could be utilised, the lookahead distance. This parameter could be alternating depending on the upcoming road. If the road has very narrow curves ahead, the lookahead distance could be much smaller then when driving on a straight road. If the lookahead distance is too great when entering a narrow curve, the controller will cut corners in a much higher degree then if the lookahead is small. But when driving on a straight road, a short lookahead would mean that if the vehicle drifts to one side, the controller would correct this error with a smaller turning radius than it would with a greater lookahead. This would effect the smoothness of the steering and in turn the experienced ride as a driver or passenger. Due to lack of time to explore this feature, this has not been investigated or tested further then just discussing the possibility.

Bibliography

[1] Gillespie T. D. Fundamentals of Vehicle Dynamics. Warrendale, Pennsylvania. Society of Automotive Engineers, INC. 1992.

[2] Coulter R. C. Implementation of the Pure Pursuit Path Tracking Algorithm. Pittsburgh, Pennsylvania. Robotics Institute at Carnegie Mellon University. January 1992. [cited 5 May 2017]. Available at: http://www.ri.cmu.edu/pub_files/pub3/coulter_r_craig_1992_1/coulter_r_craig_1992_1.pdf

[3] Giesbrecht J., Mackay D., Collier J., Verret S. Path Tracking for Unmanned Ground Vehicle Navigation, Implementation and Adaptation of the Pure Pursuit Algorithm. Suffield, Canada. Defence Research and Development Canada. December 2005 [cited 5 May 2017]. Available at: www.dtic.mil/get-tr-doc/pdf?AD=ADA599492

[4] Thomas B. Modern Reglerteknik. Fourth edition. Stockholm, Sweden. Liber AB. 2008.

[5] Rask J., Posch D. A model based approach to lane detection and lane positoining using OpenCV. Gothenburg, Sweden. Chalmers University of Technology. June 2017.

[6] Ringdahl O., Techniques and Algorithms for Autonomous Vehicles in Forest Environment. Umeå, Sweden. Umeå University. October 17, 2017.

[7] Marino R., Scalzi S., Netto M., Nested PID steering control for lane keeping in autonomous vehicles. Elsevier. Septemper 2011.

[8] Specification NVIDIA Jetson TX1/TX2 Developer Kit Carrier Board. NVIDIA Corporatoin. Santa Clara, California. Inital release date: May 2017. [cited 19 May 2017].

[9] Arduino, Arduino/Genuino UNO. [cited 19 May 2017]. Available at: https://www.arduino.cc/en/main/arduinoBoardUno

[10] JetsonHacks. Jetson RACECAR. April 2017. [cited 19 May 2017] http://www.jetsonhacks.com/2017/04/17/jetson-racecar-build-upper-

platform/

[11] Battery University. British Columbia, Canada. Cadex Electronics Inc. 5 May 2017. [cited 19 May 2017]. Availeble at: http://batteryuniversity.com/learn/article/serial_and_parallel_battery_configurations

[12] Cho B.S, Moon W.S, Seo W.J., Baek K.R. A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding. Busan, Korea. Department of Electronic Engineering, Pusan National University. June 27, 2011. [Cited 4 June 2017]. Avalable at: https://pdfs.semanticscholar.org/67b0/b4fe8972762bc8d86066cfeb427a9a7b3d58.pdf

[13] Lazar P., Shyam V. Agile Development of Automated Driving System: A study on process and technology. Chalmers Univeristy of Technology. June 2017.

A

Appendix

The tables in this appendix is the requirements list provided by the Product Development team consising of Paul Lazar and Vishnu Shyam. Further reading about their work is available at [13]. The rows marked green are the requirements that are relevant for this thesis.

Requirements for 1:10 vehicle for 3 ODDs (City, Park, Highway)				
Nr	Need Category	Requirement		Justification
1	**Regulations**			
		Follow reasonable etiquette		AV Policy guidance
		ADS must be able to be switched off by user		Revised Vienna Convention
		Document development process		NHTSA AV guidance
			Document process and procedures for assessment, testing and validation of system's capabilities	AV Policy guidance
			Document process for assessment, testing and validation of OEDR capabilities	AV Policy guidance
			Document process for assessment, testing and validation of behavioural competencies applicable for HAV	AV Policy guidance
			Document process for assessment, testing and validation of crash avoidence capabilities and design choices	AV Policy guidance
			Document process for transitioning to a minimal risk condition when a problem is encountered	AV Policy guidance
			Document process for assessment, testing and validation of fall back approaches	AV Policy guidance
		Collect , store and analyse event, incident and crash data, as well as positive outcomes		AV Policy guidance
		Evaluate and validate ADS against ODD		AV Policy guidance
2	**Testing**			
		Test approaches must include a combination of simulation, test track and on-road testing		AV Policy guidance
3	**Safety**			
		Identify minimum risk situation		SAE J3016
		Respond for minimal risk achievement		SAE J3016
		Detect performance-relevant system failure		SAE J3016
3	**Operational Design Domain**			
		Function on flat surface with good contrast road markings		ODD
		Function in speed range 0 to 7 km/h		ODD
		Function in indoor lighting conditions		*ODD*
4	**Functionality**			
		Control Steering		SAE j3016
		Control Acceleration/Deceleration		SAE j3016
		System must stop if distance to object <= 8cm		Minimal risk condition, own spec
		ODD bound		
			Allow for driving modes for each ODD	Own
			Allow enabling of operation only within ODD	SAE j3016
			Detect approaching ODD exit	SAE j3016
			DDT Fallback with minimal risk condition by ADS if no user action taken	SAE j3016
			Match environment to ODD	Own
		Lane positioning		SAE j3016, own
			Detect Vehicle positioning with respect to lane boundary	SAE j3016
			Apply steering to maintain appropriate lateral positioning	Own
			Change lanes	Own
			Function with discontinuous left hand lane	
		Speed control		SAE j3016 DDT definition
			Allow for steady speed maintenance	Desired functionality
			Determine gap to preceding vehicle	Desired functionality
			Maintain 25 cm gap to preceding vehicle in speeds between 1 and 7 km/h	Desired functionality

Requirements for 1:10 vehicle for 3 ODDs (City, Park, Highway)				
Nr	Category	Requirement		Justification
5	**Vehicle Structure**			
		Equip 2 turn signals in front and 2 on the rear of the vehicle		Cognitive feedback to people around vehicle
		The rear of the vehicle shall be equipped with 2 brake lights		Cognitive feedback to people around vehicle
		Front lane keep camera Horizontal FOV >=150 deg		Minimum turning radius of car 30cm. Camera FOV must not lose neither of road lines for lane keeping
		Front lane keep camera must be placed such that horizon line is visible, while falling no more than 15 cm away from vehicle front axle		B-snake algorithm requirements
		Front lane keep camera sensor must be placed in coincident plane with car centerline		B-snake algorithm requirements
		Short range coverage to vehicle proximity shall not leave more than X deg blind spots		ERTRAC
6	**Monitor the driving environment**			SAE j3016 DDT definition
		Object and event detection		SAE j3016
			Detect Objects	Own
			Determine object relative position	Own
			Detect Events	Own
			Detect road surface	Own
		Recognition		
			Identify other vehicles in and out of travel path	AV Policy guidance
			Identify pedestrians	AV Policy guidance
			Identify driving lanes	Own
			Identify driving lanes	Own
			Identify Intersections	AV Policy guidance (Normal driving)
			Identify parking spaces	AV Policy guidance (Normal driving)
			Identify parking spaces	AV Policy guidance (Normal driving)
			Identify stopped vehicles	AV Policy guidance (Normal driving)
			Identify static obstacles in path of vehicle	AV Policy guidance (Normal driving)
			Identify Moving objects in path of vehicle	Own
			Identify traffic signs	Own
			Identify road markings	Own
		Classification		
			Recognise speed limit and speed limit changes	Own
			Recognise Stop/Yield signs	
			Recognise Stop/Yield signs	AV Policy guidance (Normal driving)
			Recognise right-of-way vehicle	AV Policy guidance (Normal driving)
			Recognise Freeway entries	
			Recognise Passing and No-Passing zones	AV Policy guidance (Normal driving)
			Recognise stopped vehicles	
		Response preparation		
			Compute a steering angle and vehicle speed	Interface with Arduino and control motors requirements
		Monitor vehicle performance (for DDT performance-relevant system failures)		SAE j3016 (monitoring)
			Measure vehicle speed	Own
			Measure wheel turning angle	Own
		Object and event response execution		SAE j3016 DDT definition
			Respond to encroaching incoming vehicles	AV Policy guidance (Normal driving)
			Yield to pedestrians at intersections and crosswalks	AV Policy guidance (Normal driving)
			Respond to speed limit changes	Own
			Respond to Traffic Signals and Stop/Yield Signs	AV Policy guidance (Normal driving)
			Navigate Intersections and perform turns	AV Policy guidance (Normal driving)
			Execute parking maneuver	Own
			Perform passing maneuvers	AV Policy guidance (Normal driving)
			Respond to stopped vehicles	AV Policy guidance (Normal driving)
			Respond to static obstacles in path of vehicle	AV Policy guidance (Normal driving)
			Respond to access restrictions (One-way, no turn, ramps, etc)	AV Policy guidance (Normal driving)
			Make right of way decision	AV Policy guidance (Normal driving)
			Perform high speed freeway merge	AV Policy guidance (Normal driving)
		Plan manouvre (tactical)		SAE j3016 DDT definition
		Enhance conspicuity		SAE j3016 DDT definition
			Control rear lighting	Own
			Control signaling	Own

Requirements for 1:10 vehicle for 3 ODDs (City, Park, Highway)			
Nr	Need Category	Requirement	Justification
7	**Features**		
		High-speed freeway speed shall be 7 ± 0.4 km/h	VDA
		Urban driving speed shall be 5 ± 0.4 km/h	own
		Parking speed shall be <= 1 ± 0.4 km/h	SAE J3016 examples
8	**Performance needs**		
		Ensure smooth execution of lateral movement	Sigma
		Ensure smooth execution of longitudinal movement	Sigma
9	**HMI**		
		Issue user notification of approaching ODD exit	SAE J3016
		Allow override at user request	SAE J3016
		Provide user with ADS status	AV Policy guidance
		Display system performance status	AV Policy guidance
		Notify user when ADS is currently engaged	AV Policy guidance
		Display current DDT of ADS	AV Policy guidance
		Notify user when ADS is not available	AV Policy guidance
		Notify of ADS malfunction	AV Policy guidance
		Notify user of request for user-fallback in case of system failure	AV Policy guidance