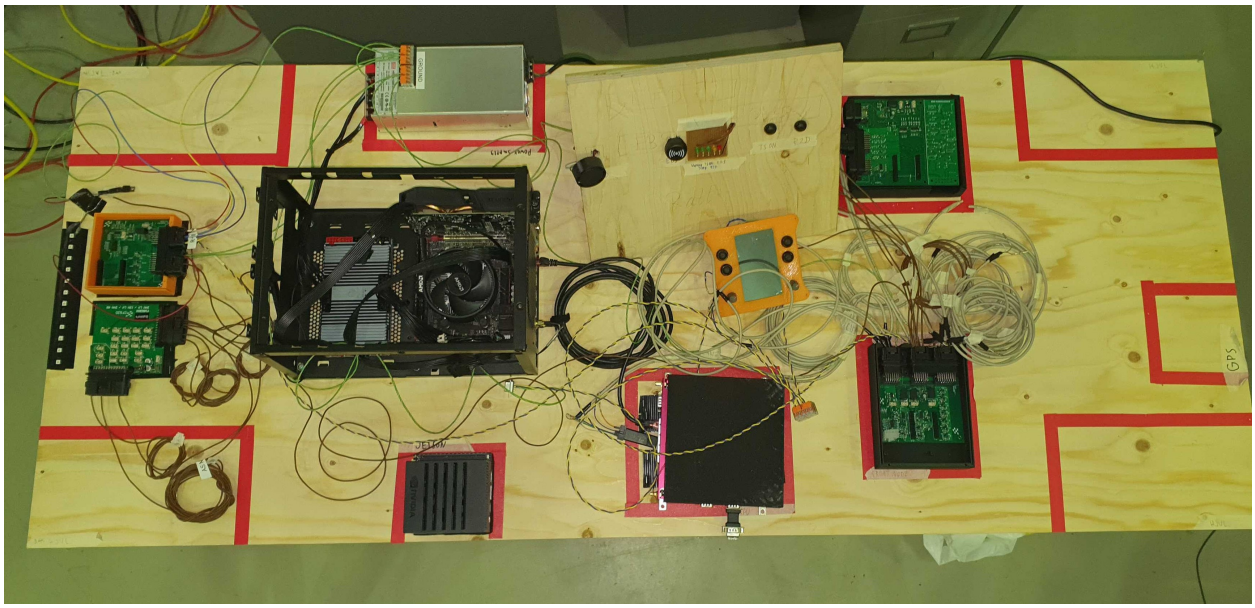
CHALMERS



A hardware-in-the-loop rig for hardware-agnostic simulation of digital twins

The development of a hardware-in-the-loop rig for an autonomous electric race car of Chalmers formula student driverless

MMSX20-21-5

Alan Ali Doosti, Gabriel Ingemarsson, Elmer Lingeståal, Martin Losman, Viktor Wennergren and André Örjas

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Bachelor thesis 2021

A hardware-in-the-loop rig for hardware-agnostic simulation of digital twins

The development of a hardware-in-the-loop rig for an autonomous electric race car of Chalmers formula student driverless

Alan Ali Doosti
Gabriel Ingemarsson
Elmer Lingestål
Martin Losman
Viktor Wennergren
André Örjas

CHALMERS

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2021

A hardware-in-the-loop rig for hardware-agnostic simulation of digital twins

The development of a hardware-in-the-loop rig for an autonomous electric race car of Chalmers formula student driverless

Alan Ali Doosti
Gabriel Ingemarsson
Elmer Lingeståal
Martin Losman
Viktor Wennergren
André Örjas

Supervisor: Christian Berger, Associate professor, Software Engineering division, Department of Computer Science and Engineering.
Examiner: Ola Benderius, Associate Professor at Mechanics and Maritime Sciences, Division of Vehicle Engineering and Autonomous Systems

Bachelor Thesis 2021:14
Department of Mechanic and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The *hardware-in-the-loop* test rig at its most complete state.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

A hardware-in-the-loop rig for hardware-agnostic simulation of digital twins

The development of a hardware-in-the-loop rig for an autonomous electric race car of Chalmers formula student driverless

ALAN ALI DOOSTI
GABRIEL INGEMARSSON
ELMER LINGESTÅL
MARTIN LOSMAN
VIKTOR WENNERGREN
ANDRÉ ÖRJAS
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

Modern cars require a lot of resources to develop, and future fully autonomous cars will likely add further complexity. In order to limit the resources spent on development and testing with a real car on a real track, a technique called hardware-in-the-loop (HIL) can be used. This means testing new hardware and software on a separate build consisting of the same hardware as in a real car together with digital representations of hardware components not physically present, a *digital twin*. Chalmers formula student driverless (CFSD) is a project where master students from Chalmers University of Technology build their own autonomous race car and they could benefit from the concept of HIL for the same reasons that the industry does. By accessing the database of CFSD and getting involved in their project, while taking ethics into account, it should be possible for a group of bachelor students to provide them with a rig utilising the concept of HIL. This goal was not fully reached in the time scope of the bachelor thesis described in the report. Nevertheless, a foundation to build upon has been created, as well as hardware working as intended and successfully using the same software as the car.

Keywords: HIL, hardware-in-the-loop, CFSD, Chalmers formula student driverless, digital twin, autonomous race car, bachelor thesis

Acknowledgements

Many thanks to the team of Chalmers formula student driverless 2021 with their providing of great technical expertise as well as material. Thanks also goes to Mikael von Redlich and Fredrik von Corswant at the Chalmers Revere facility for providing access to their laboratory facility where the project has been carried out, as well as their meeting rooms in SAFER, Vehicle and traffic safety centre at Chalmers University of Technology. Christian Berger, who during the project has been at leave, was kind enough to still provide support and shine light at some crucial topics. Also thanks to Robert Östberg, a member of Chalmers Formula Student, who has given us a lot of great advice on wiring and other electrical systems. A special thanks also goes to Fredrik Ljungdahl at Volvo Car Corporation for sharing his knowledge about how HIL-rigs are implemented in the industry. Last but not least goes a great thanks to Ola Benderius for supervising the project from start to finish and helping whenever a problem has risen.

Nomenclature

AI	Artificial Intelligence
AIR	Accumulator Isolation Relay
AMS	Accumulator Management System
APU	Advanced Processing Unit
ASSI	Autonomous System Status Indicator
CAD	Computer Aided Design
CAN	Controller Area Network
CFS	Chalmers Formula Student
CFSD	Chalmers Formula Student Driverless
CPU	Central Processing Unit
C++	An object oriented programming language
DBC	Database Container
DRAM	Dynamic Random Access Memory
Docker	Virtual Container, runs on many operating systems
EBS	Emergency Brake System
FEM	Finite Element Method
FSG	Formula Student Germany
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HIL	Hardware-In-the-Loop
IP	Internet Protocol
LXC	Linux Containers, similarities with Docker
NAND	NOT-AND logical gate
Odvd	Open DLV, a software ecosystem
OS	Operating System
OTA	Over-The-Air
PCB	Printed Circuit Board
PE-box	Power Electronics Box
PMIC	Power Management Integrated Circuit
PXE	Preboot Execution Environment
RS232	Recommended Standard 232
SOM	System On Module
SPI	Serial Peripheral Interface
STL-file	Stereolithography-file

Contents

List of Figures

List of Tables

1
Introduction

In recent years cars have become more and more complex. Driver aids have evolved from cruise control to adaptive cruise control with assisted lane keeping to autonomous driving now being a reality. Because of this trend, development of new cars require great amounts of resources as drivers, to a greater extent, rely on various implemented driving aids. As a result, the vehicle industry has developed methods to test and build new system layouts in easier and faster ways than building a full car prototype. These methods can allow automation of large amounts of testing, which previously would have had to be done by car prototypes.

Digital twins are an important part in modern vehicle development. A digital twin is a digital representation of a physical system, often consisting of 3D-models as well as mathematical models of the dynamics and electronics of the system. The purpose of a digital twin is to limit the testing needed in the real world by simulating and testing different scenarios in a digital environment. In this project a digital twin is integrated with hardware components, thus becoming a hardware-in-the-loop (HIL) rig.

Using HIL-rigs is essential when designing and developing modern vehicles. HIL-rigs are primarily built with the same hardware components as the real car has, stacked inside a cabinet-like compartment. This concept is used in order to efficiently test the behaviour of certain systems by performing simulations with varying conditions in controlled environments. HIL-rigs are most commonly used in the initial stages of product development to ensure high quality standards and to decrease the time it takes to discover faults in the system. Building a prototype and performing tests, which is the alternative, requires a lot more resources to both plan and execute. Additionally a HIL-rig will operate in a much more controlled environment than a prototype, which can reduce the risk of unexpected costs and delays.

The purpose of the project described in this report is to develop and build a fully functioning HIL-rig. The rig will be a digital twin of a subsystem in a physical car with as similar hardware components as possible. When the rig is finalised the goal is to be able to seamlessly mix pure simulation with the hardware in the rig. For example log-files from when the car is running should be possible to use in the rig. This, so that both the behaviour of the autonomous system can be studied, and also to test the system and its components during longer periods of time. Another purpose of the HIL-rig is to make it possible to test new electrical components or software without having to disassemble the car or test unstable software in the real

world. This will save time during future development of the car and increase safety, since a HIL-rig allows both software and hardware components to be verified before being installed in the real car.

1.1 Chalmers formula student driverless

Chalmers formula student driverless (CFSD) is a student project carried out annually at Chalmers University of Technology with the primary goals of designing, developing, and manufacturing an autonomous electric race car. The car is based on an electrical powertrain and computer solutions for handling the driving of the vehicle. Besides the technical skills required for designing a race car, the teams are also expected to demonstrate skills in marketing, such as finding sponsorship and creating business plans. All these aforementioned activities are then being assessed during the final stage of the project, which is to, hopefully, compete in an international competition for autonomous race cars.

Each year a team is assembled of master students from many countries with different engineering backgrounds and technical experience. In addition to building a race car, the project enables students within the CFSD-team to carry out their masters thesis. Every year the new team is initially supervised by a few team members from the previous year. Their task is to introduce new members to the project and to act as mentors during the initial phases of the project, making sure that the project works smoothly from the start, and to mediate technical knowledge and experience.

The project of CFSD is based on frameworks settled by the international organisation Formula Student Germany (FSG). This organisation holds several Formula Student competitions in different classes around Europe during summer. The classes are *internal combustion engine vehicles*, *electric vehicles* and *driverless vehicles*, of which the team of CFSD is qualifying for the latest of the three. However, in order to participate in any of the competitions, the team is expected to follow certain rules and to keep a detailed technical documentation of their cars. Admission to, and participation in Formula Student competitions are performed in several steps.

Initially, to gain entrance to a competition, a team has to qualify via a quiz which is provided within a limited time with emphasis on mechanical and electrical engineering problem solving. Once a team manages to qualify for one of the limited number of places in a competition, the next step is to send detailed documentation of different subsystems in the car to the competition management.

When a team finally gets qualified to a competition, the fully functioning vehicle is transported to a competition site where several stages of scrutinising and performance tests are conducted before the competition can start. Assessments are then being made in static events where points are given to teams based on how well their business plan was developed, the cost of the production and how the design and the manufacturing was carried out. There are also assessments of dynamic events where points are given to teams based on how well the race car performed on the track.

At the end of a competition, the team with most points win.

Figure 1.1: The autonomous electric race car built by the CFSD-team in 2018, driving in a circuit made of cones.

The main task of the autonomous race car, during the dynamic events, is to manage to drive as fast as possible on different tracks marked by traffic cones, see Fig. 1.1. A benefit of such a contest is that engineering students are given an opportunity to sharpen their skills in mechanical and electrical engineering. It is also a chance for the engineers to learn and implement technical innovations in order to attract potential employers after finishing their education.

1.2 Delimitations

This project was limited to physically building the low voltage system of the CFSD race car, powered by 24 V or less, as well as implementing the associated software. This was decided due to the risks of working with higher voltages and larger currents, and since it would also be more cost-effective. The rest of the subsystems of the car was part of the digital twin, and was consequently simulated. This included a vehicle dynamics model and a simulation of all relevant signals needed to accurately represent the real car.

2
System architecture

The following section explains the architecture of the electronics in the race car of CFSD. The car consists of a high voltage system and a low voltage system. The high voltage system will not be covered in great depths since it is not a part of this project. The low voltage system mostly consists of control circuits, sensors and actuators which sends signals to each other in order to control the car. Specifically, some of the most important components of the low voltage system, are *the autonomous system node*, *front node*, *rear node*, *APU*, *PC rotor temperature sensors*, *damper position sensors*, *steering angle sensor*, *brake actuator*, *steering wheel actuator*, *steering wheel* and *the dashboard*. In Fig. 2.1 a general layout of the low voltage system for both the race car and the HIL-rig is presented. The components are 1) an autonomous system node, 2) a front node, 3) a rear node, 4) an APU, 5) the main computer, 6) the accumulator, 7) electric motors, 8) the IMU, 9) camera and GPS, 10) various sensors for each wheel.

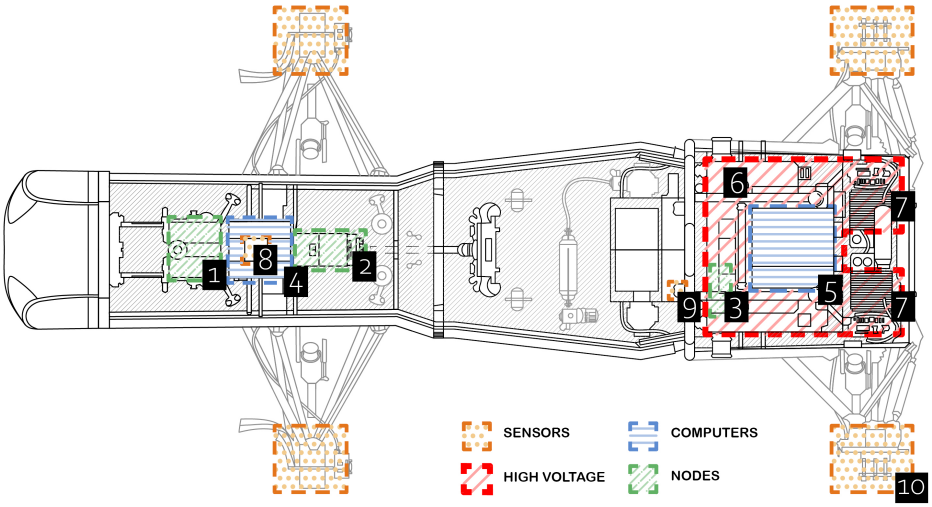


Figure 2.1: Overview of the various components of the race car.

The blue areas of the car in Fig. 2.1 represents the various computers in the car, while the red area shows the high voltage system, the green areas represents the nodes, and the orange areas represents the various sensors. The sensors mainly consists of an IMU in the front of the car, a camera and GPU in the back, and various sensors at the wheels and actuators. However, there are other sensors spread through out the car that are not shown in Fig. 2.1. Each of these sensors are connected to one of the nodes, which interpret signals from the sensor. The entire high voltage system is concentrated to the back of the car and consists of the accumulator and the electric motors, as well as supporting components like the DC-AC converter and safety systems. There are also two computers in the car; a PC in the back as well as an APU in the front.

The car of CFSD is based on several different technical principles. To help turn electricity into movement, power electronics, such as inverters to change direct current into alternate current, is used. The software is largely based on *microservices* which is the practise of dividing a software application into small, well designed and easily understood parts in a characteristic architecture [3]. Actuators are mechanical devices which move when needed. Control systems are built into the software and use feedback loops to stabilise and control the car. Furthermore, sensors are an essential part of the system as they measure different values within or outside the system to make this data available for the computers.

A *digital twin* is a digital copy of a physical system. *Digital twins* are very useful for simulating the expected behaviour of a system in an environment, calculating lifespan or service intervals. Another use case is testing new software or evaluating new components before changing anything in the physical real world system.

CAD-models can also be a part of *digital twins*, in which case the twin can be more of a physical representation and then also be used for packaging, or a so called finite element method (FEM) strength analysis. *Digital twins* are also often made of mathematical models of systems, like for example a vehicle dynamics model, or a model of an electronic system [7].

A *digital twin* may be complemented with hardware components by putting them in a HIL-rig. Conducting simulations using a HIL-rig is, essentially, to use a digital twin with certain components that are physical hardware instead of simulated. This will most likely make the twin behave more accurately, like the physical system. This also allows for swapping hardware to evaluate new hardware, as well as making it possible to diagnose malfunctioning hardware components in a simulated environment within a specific sequence of events. This is very valuable for the development process, since testing hardware in a simulation is both more cost and time effective than testing in the real world. In many cases it is also safer, especially in vehicle development [8].

The *digital twin server* used in this project is a computer with the purpose of replaying data logs recorded from the real car to the HIL-rig. Therefore, the server

had to be connected to the CAN bus and the APU, and the digital twin server was set to feed video to the HIL-rig.

2.1 High- and low- voltage circuits

The high voltage system of the race car of CFSD is composed of the accumulator, or battery of 580 V to supply power to the tractive system and the low voltage system via a DC-DC converter. The accumulator is built up by several accumulator cells which delivers direct current.

The tractive system of the car is defined as the high voltage system connected to the motors and the accumulator. From the relay box, the direct current has to be converted to an alternating current in order to power the motors. This conversion of voltage is executed by a subsystem known as the inverters, one for each motor. The system has two motors, to enable torque vectoring. Torque vectoring is when a car can individually control the torque of a given wheel or axle in order to improve performance, handling and stability.

High voltage is in this case defined as voltage above 24 V. High voltage is often needed for certain actuators such as electric motors. However, high voltage is not needed for all the systems in the car. In order to supply appropriate voltage and current to each electronic part, some necessary steps are needed to be performed. Firstly, a relay box is a subsystem needed, closely assembled to the accumulator which converts from high voltage to low voltage and provides a safety mechanisms according to the rule book of FSG. Also, two safety systems built in to the relay box is the accumulator management system (AMS), with the task to monitor and send signals to indicate certain critical states occurring, and the accumulator isolation relay (AIR) which isolates the accumulator from the rest of the system.

Low voltage is, in this case, 24 V or less. The low voltage circuit used here is synonymous with a control circuit. Sensors, actuators, micro controllers and computers are powered by low voltage and are a part of the control circuit. The low voltage system includes more complicated processors and graphics processors but also more simple integrated circuits such as NAND-gates and everything in between. The control circuit or low voltage circuit is used to control the car using either low voltage actuators, such as the steering actuator, or communicating with the high voltage tractive system which controls the motors.

2.2 Communication

All the different components in the car needs to communicate with each other. This section describes different protocols used for communication. It also includes how the HIL-rig communicates with the car and how this can be tested.
One of the communication protocols is the controller area network (CAN) bus. The

CAN bus is a network standard mainly used in vehicles. It consists of two wires which are shared between all the devices. The two wires are CAN-high and CAN-low and they send the same signal but mirrored in voltage. The cables are twisted to reduce noise. The different components are connected in parallel and send their signals through the same CAN bus. This means that the CAN bus handles the signal from a single component at any time. In this manner the CAN bus monitors the system and keeps it from sending out two signals at the same time. CAN buses are used in vehicles, mainly because of the convenience of only having one wire, but also since it makes the network robust. Each sent message has a priority level which helps to decide which message comes first. Usually the transmitters themselves do not have a priority, but all messages from a certain transmitter can of course be set to a certain priority.

In the car there are three different CAN networks CAN0, CAN1 and CAN2. CAN0 is connected to the APU and components in the rear of the car. These components are the rear node and the motors. CAN1 is connected to many of the main components of the car such as front node, rear node and the APU. CAN2 is connected between the APU and the motor controller for the steering actuator. The APU works as a CAN interface between the high level components, the PC and the camera, and the low level CAN-network.

Serial peripheral interface (SPI) is a widely used serial bus. One of the big benefits that come with using SPI is that data can be transferred without interruptions. Devices that communicates via SPI have a master and one or more slaves. The master is the controlling device and the slave takes instructions from the master. SPI is a three or four wire system with different ports. These are MOSI, MISO, SCLK and SS/CI. The MOSI stands for Master output Slave input and is used so the master can send data to the slave. MISO stands for Master input Slave output and is used so that the slave can send signals to the master. SCLK is a clock which is used for the clock signal. SS/CS stands for slave select/chip select and it tells the master which slave to send the data to. Because of this, SPIs that only have one slave do not need the SS/CS and are therefore a three wire system.

The *recommended standard 232* (RS-232) is one of the oldest, yet widely used communication protocols. It uses serial communication which means that the data will be sent bit by bit. The downside with serial communication is that it is slower than parallel communication but is used for longer data transmissions due to its lower cost compared to parallel communication.

Ethernet is used to connect network devices with each other. The purpose of Ethernet is that computers and other devices such as APU can share files, data and information with each other and with the camera. The most common Ethernet cable is the twisted pair cable which consists of multiple pairs of wires twisted together. Each signal is transmitted in a twisted pair of wires, one positive signal and one mirrored negative, this minimises ripple and noise. Ethernet also uses duplex communication which allows for communication in both directions at the same time.

These two properties allow fast and reliable communication.

Fig. 2.2 displays a schematic figure of the system, Ethernet and the CAN bus. The goal is for the data received from the database to automatically be played to the rig. In principle this is to be done by first letting the car do a test drive or a race. Then the collected *req.* file is automatically uploaded to the database in the Chalmers Revere facility. Lastly, the digital twin server is automatically triggered to run on the HIL-rig and the results of the drive are saved.

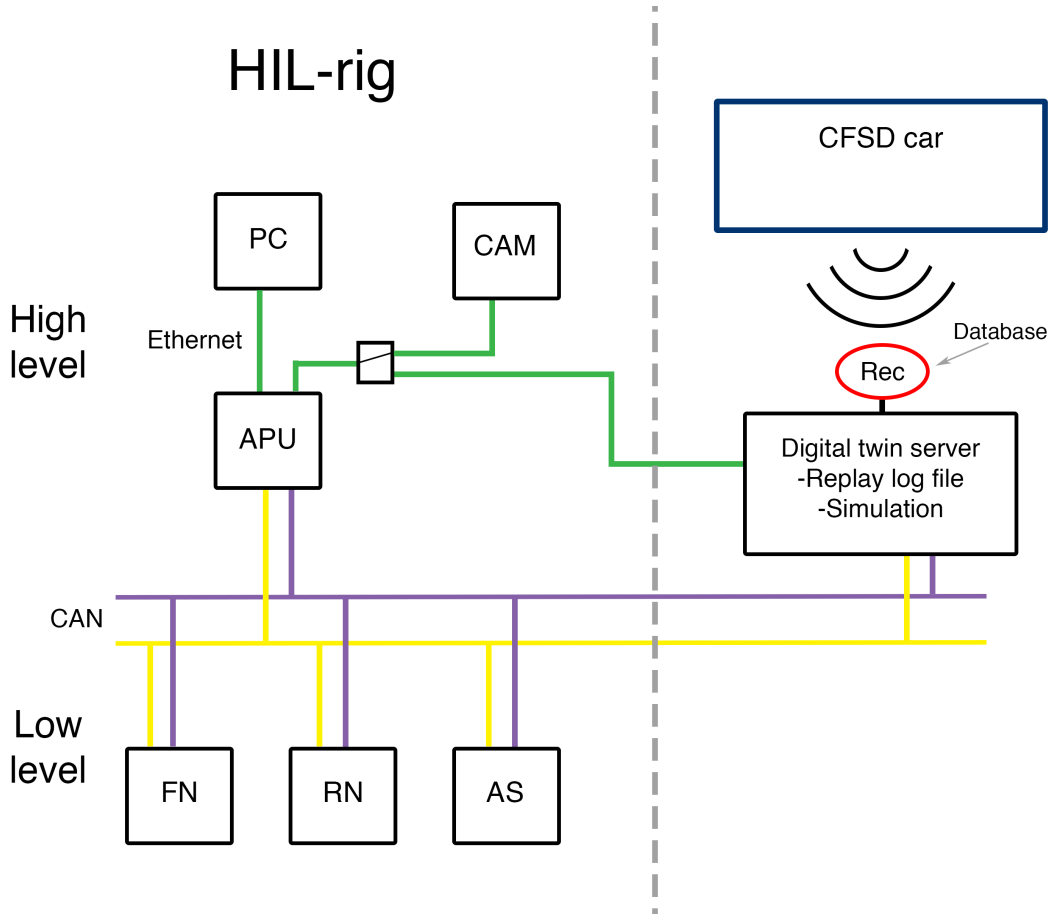


Figure 2.2: Schematic figure of the system.

The system can be tested to make sure that the communication between the car and the HIL-rig works. This can be either done with simulation, data replay, or a combination of them both. The simulation, as seen in Fig. 2.3, is achieved by injecting artificial CAN signals generated by a vehicle dynamics model, and an artificial camera feed generated by a 3D engine. In addition, the digital twin concept can also be realised by using recorded log files directly from the actual racing car. In these log files, also referred to as *rec files*, all high-level signals from the actual car are present, including the video feed, as shown in Fig. 2.4. By injecting these recorded signals into the HIL-rig the exact low-level behaviour of the actual car can be replayed.

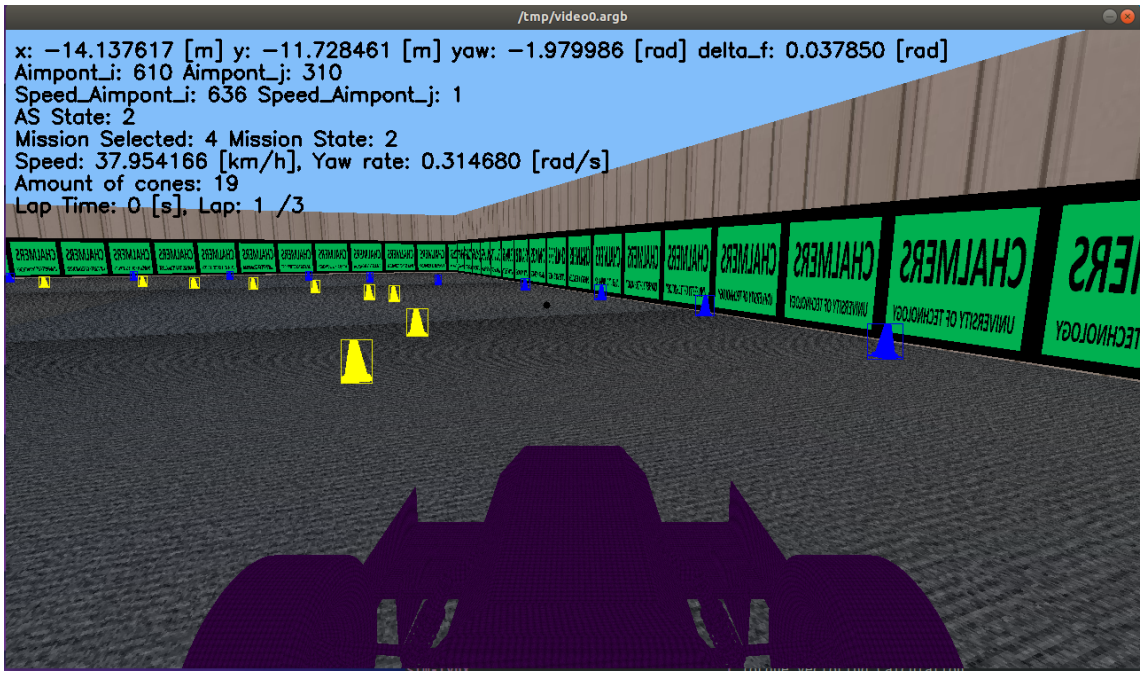


Figure 2.3: A simulated vehicle dynamic model.

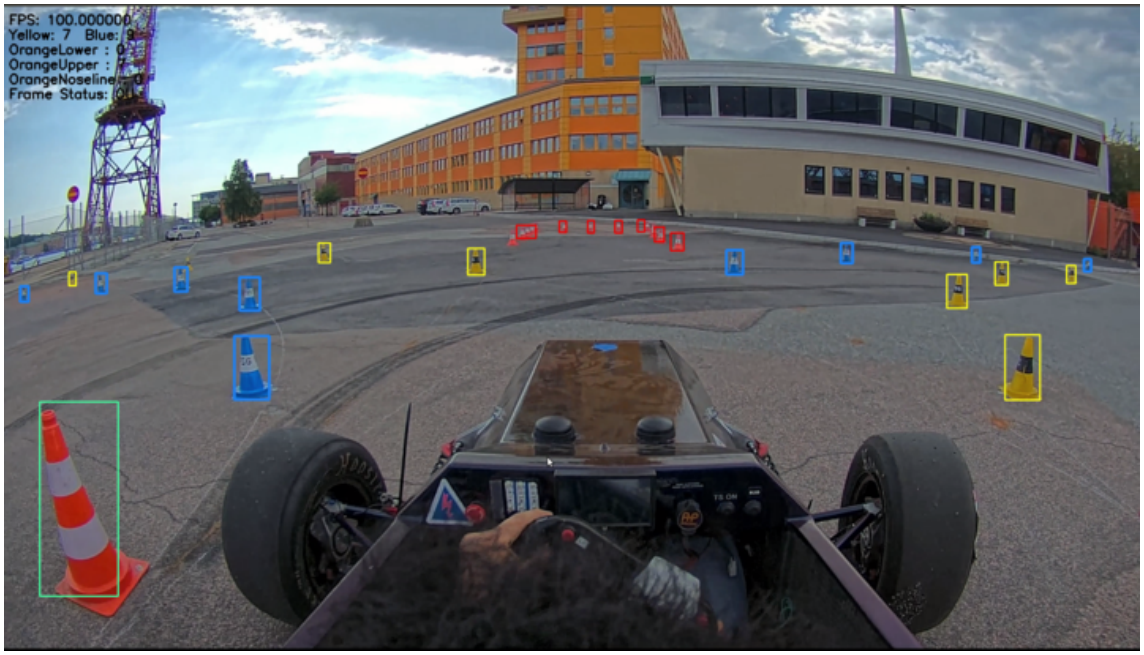


Figure 2.4: A picture of the video feed of the race car on a test track.

2.3 Nodes

The low voltage system uses three nodes to bundle together sensors, actuators and micro controllers. The nodes are also connected to the CAN bus to communicate with each other and the APU. The front node is connected to components located in the front part of the car, and the rear node connects to components in the rear. The third node is the autonomous system node, which connects to components

specifically used for the autonomous system. The nodes are used primarily for three reasons; firstly, to avoid running all cables through the entirety of the car; secondly, to convert all the different types of signals from the sensors to CAN-messages, and finally, to assist in troubleshooting. When it comes to troubleshooting, it is beneficial to have the possibility to localise the problem to the system and connecting it to a particular node before looking for the problem in specific components. It is easier to find problems with a particular node and then the specific component, rather than if everything would be connected to a single component.

Fig. 2.5 illustrates the front node and the components connected to it. For example, the front node collects temperature data from the front tires and brakes, as well as position data from sensors by the dampers. Information about break pressure and steering angle is also obtained at the front node. Furthermore, the front node sends these signals to the dashboard and to the APU through the CAN bus.

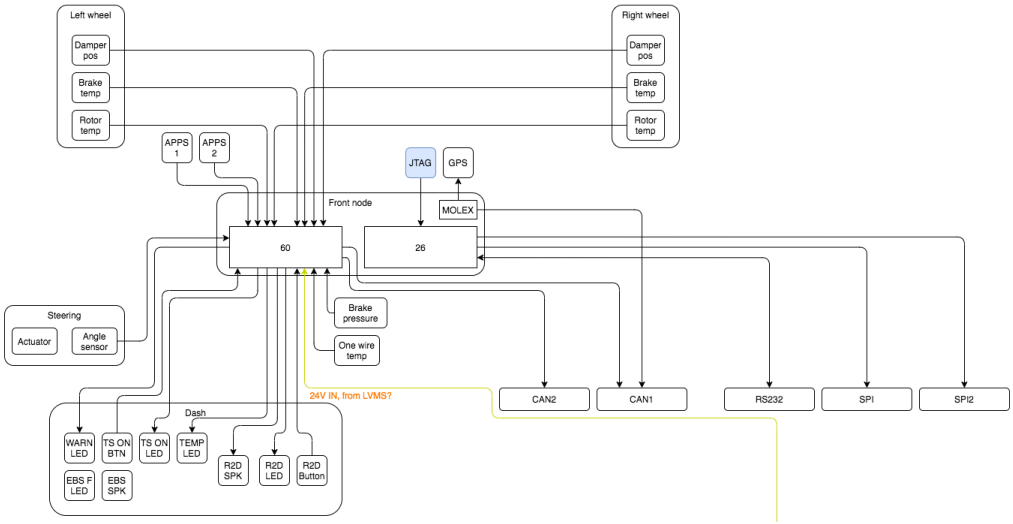


Figure 2.5: A schematic figure of the front node and its connections.

As previously mentioned, the front node communicates with the components in the front of the car. Among other things, these components include sensors by the wheels, the dashboard and the steering actuator. As shown in Fig. 2.5 the front node collects signals from the sensors which are then sent to the front node. In order to make it possible to drive the car manually, the dashboard has a ready-to-drive button and a tractive system on button. The signals from the buttons and the sensors are processed and then sent to LEDs in the dashboard to a display in the steering wheel through the SPI. The signals are also sent to the APU via the CAN bus.

The rear node works much in the same way as the front node. Fig. 2.6 displays the different connections of the rear nodes. The rear node collects sensor signals of the rear damper positions and temperature data from the rear tires and their brakes. Furthermore, it controls the brake light, radiator fan and is connected to the PE-box that controls the motors.

Fig. 2.6 illustrates the rear node and the components connected to it. The rear node collects temperature data from the rear tires and brakes, as well as position data from the dampers. It also controls the brake light and is connected to the PE-box that controls the motors.

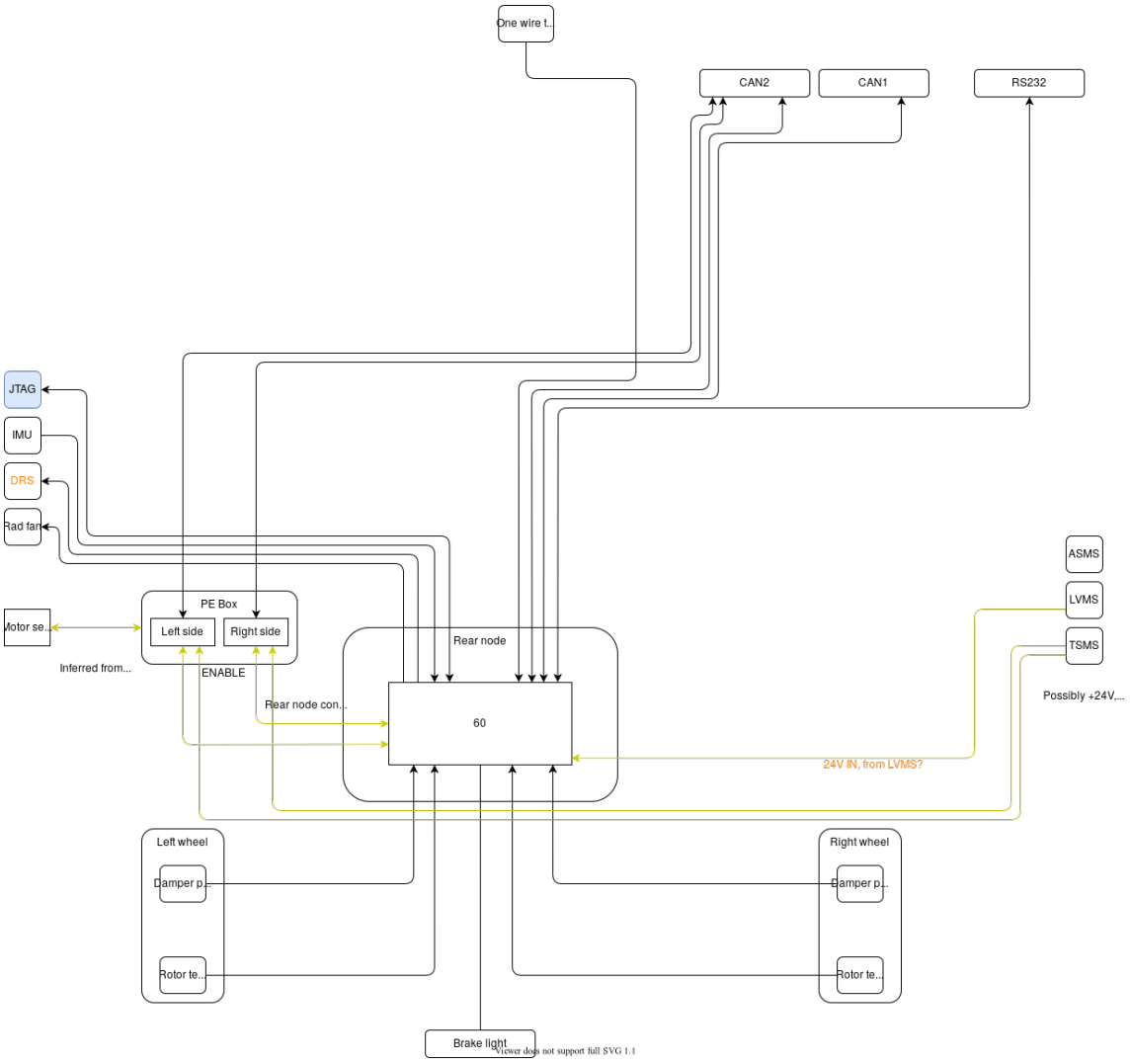


Figure 2.6: A schematic figure of the rear node and its connections.

The autonomous system node collects data from sensors which measures wheel speed, steering angle position, pneumatic pressures in the break system and position for the break actuator. With this information, the node then controls the steering system through a motor controller and motor for an actuator. It also controls the brake system with a pneumatic compressor and an electric pressure regulator. This is done through hardlogic on the same PCB as the autonomous node, but completely separated from each other due to regulations from FSG. The autonomous system node also controls the lighting process of the *autonomous system status indicator* (ASSI) and the *emergency brake system* (EBS) failure LED.

2.4 Actuators

While driving autonomously the car uses actuators to replace the physical inputs of a human driver to the car. These actuators are used for braking and steering, while the other systems are controlled digitally. The actuators are responsible for physically pushing down the brake pedal, as well as turning the front wheels for steering. The reason why the brake and steering use actuators is because both of them require physical input. Neither the steering system nor the hydraulic breaking system work with electrical signals. This is also the reason why there are no other actuators in the car; every other system is compatible with electrical signals, which are more flexible and scalable.

2.5 Autonomous system

The purpose of the autonomous system is to replace the human driver's senses and decision making, and its core part is the APU and the PC. The senses are replaced with a camera and the decision making is done with artificial intelligence on a PC that has a powerful GPU for handling graphic inputs from the camera. The PC is the main calculating unit of the car. It is connected to the APU and the camera through Ethernet. The main purpose of the PC is to process the pictures taken by the camera and then change the direction of the car accordingly.

One of the great challenges of designing a driverless race car, is the determination of its position at every moment in time, with a limited camera frame rate and with limited processing power. The reason for this is that when the car moves at high speeds, the distance it travels between the captured frames and during the image processing, is non-negligible. In order to reduce this problem, an algorithm that uses Global Positioning System (GPS) will be implemented in the car in the future. The GPS tracks the position and the direction of the car by triangulating it with satellites. By combining these two sources of position data, the position of the car can be determined with a higher accuracy, than if only one of these sources of data would be used. More specifically, this is done by projecting the cones that the camera detects onto the ground, and then rotate them in an absolute frame of reference that was employed when the car started. The GPS data is used to locate the car and determine its direction.
The APU is a small computer that does several things. One of these things is to act as a CAN gateway between the low- and high-level systems, see Sect. 2.2. It converts the low level CAN messages into OpenDLV (ODVD), and vice versa. The OpenDLV message can then be sent out on the high-level ethernet network to be further processed by the various OpenDLV microservices.

2.5.1 Safety system

In order to minimise the risk of accidents, several safety systems were integrated into the design. These include the emergency brake system (EBS) and the remote

emergency system (RES). The RES consists of a remote controller and a receiver connected to the EBS. The EBS is activated either automatically, whenever the system detects an error that requires the car to stop, or manually through the RES. When activated, the EBS shuts the tractive system off and brakes the car with the aforementioned braking actuator. Design of the EBS was made by CFSD in accordance with FSG 2021 EBS reference design and a copy of CFSD's EBS could therefore be implemented in the HIL-rig.

2.5.2 Sensors

In order to enable proper control of the car by the PC, several data collecting sensors are required. For this reason there are multiple sensors on the car, monitoring tire temperatures, brake temperatures, hydraulic pressure in the brake system, damper positions in the suspension, steering angle and pedal positions. These sensors provide the computer with the necessary information of the status of the car.

One of the most important sensors is the camera since it delivers a lot of information about the environment. Since the camera has no depth perception, it is difficult, and takes a lot of time for the AI when it comes to calculating the distance to the objects. This is especially troublesome in the real world where there are a lot of different shapes, making it quite difficult to teach the computer all of the edge cases. However, considering that the track where the CFSD car will compete is relatively predictable, this weakness of the camera is reduced. The cones visualising the track-limits make it easier to compute the path. The system also takes other information into account when calculating the driving path, like for example the wheel speed, steering angle, et cetera.

2.5.3 Nvidia Jetson AGX Xavier

The Jetson AGX Xavier, which can be seen in Fig. 2.7, is an artificial intelligence computer created by NVIDIA intended for autonomous robotic applications. It focuses on deep learning and machine learning tasks. The needs of the CFSD car fits the intended purpose of the NVDIDA Jetson, and was therefore implemented in the HIL-rig to evaluate the possibilities of using it in the car. The NVIDIA Jetson uses a complete system on module (SOM), with CPU, GPU, PMIC, DRAM and flash storage. The system provides the performance and power efficiency needed to run autonomous machines software faster and with less power. One advantage with NVIDIA Jetson is that it supports CAN which is an important communication protocol in the car.

Earlier CFSD cars have had both a PC and APU. Since the APU does not have enough processing power to handle image processing, it have been necessary to use a PC with a GPU to handle the image processing and AI. This means that the APU is mainly used for its CAN interface capabilities and 4G Internet connectivity for the purpose of remote monitoring and over-the-air (OTA) software updates.

One of the main purposes of the HIL-rig is to explore the possibility of merging the APU and PC to one NVIDA Jetson Xavier unit, handling everything from the CAN signals to the image processing. This would bring a lot of benefits. The NVIDIA Jetson is only using a fraction of the power that the PC uses, around 30 W compared to the PC that can draw around 200 W around peak loads. With this significantly lower power draw, the NVIDIA Jetson also manages to outperform the PC in certain types of image processing. And it would lead to a much smaller and lighter construction, since it would replace both the PC and the APU.

Figure 2.7: An NVIDIA Jetson AGX Xavier chip to the right and a chip with a large mounted heat sink to the left.

2.5.4 Software automation

In this project all the components of the HIL-rig and car runs the same software. Every time the system starts it looks for, and downloads any updates. By introducing this level of automation we make sure everything is up to date and avoid errors. In order to achieve this it was necessary to introduce an operating-system-level virtualisation, over-the-air (OTA) update and remote system monitoring, as seen in Fig. 2.8.

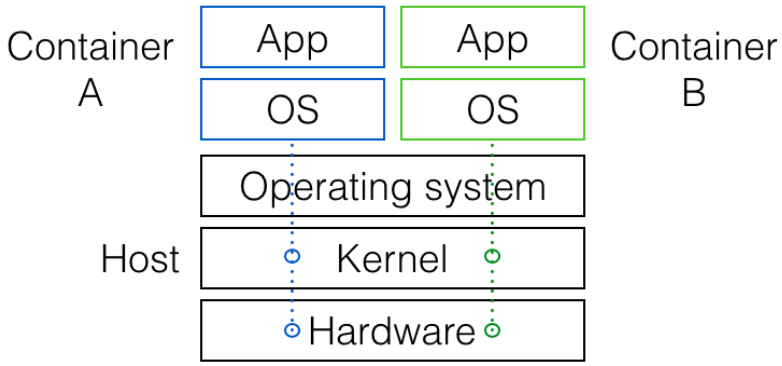


Figure 2.8: An operating-system-level virtualisation.

Operating-system-level virtualisation uses a host and containers. The host system

includes a Hardware Kernel and an operating system. The containers include a small version of the operating system and an application. The containers use the same Kernel and hardware as the host. The big advantage with operating-system-level virtualisation is that no kernel and hardware needs to be simulated. This means that the system will not receive any performance penalties. By default the containers are isolated from each other. This way there will be no interference between the containers. Common implementations used in this context are for example Docker and LXC. In this project Docker was used which is mainly a software development platform that makes it easy to develop and deploy apps inside virtual containerised environments. Meaning apps will run the same no matter which machine they are running on.
OTA updates are primarily for products in the wild, meaning products that have been sold, or products that are unavailable. However, it can also be helpful during development. It allows each developer to deploy their binary in a seamless way. This way the developer does not need to go to the workshop and manually put in the binary. The OTA updates therefore enables the engineer to push the newest software into the product.

Remote system monitoring is a way of assessing system performance of the product so that the software can be refined. This is useful because it can measure the performance between different updates. So if a change is made to the software the developer will know if it is better than the last one.

By combining the steps above, a full automation chain can be created. The automation chain includes the following steps. Developer commits changes to the code repository. The Continuous integration infrastructure checks the code, and returns to the developer if something is wrong with the code. A tag is made to the new release on the accepted new binary. The products get the new software entities. Then the products can send back monitoring data to the developer and the process can be repeated.

The benefits from having such an automation chain is that the products are always up to date. Engineers can always check the in-field performance, and the engineers working on the software will have a better understanding of the products.

3
Method

In order to assemble the HIL-rig, knowledge about the low voltage system had to be gained by collecting information from sources provided by the CFSD team. These sources were primarily schematics and other technical reports written by previous CFSD students. The current CFSD team members were also consulted to compliment the documentation. With this information an outline of the low voltage system was created and the components were ordered.

While awaiting the components, work on various parts of the project began. These parts included the wooden test bench, the node containers and the wires, as well as the software.

3.1 Assembly of the HIL-rig

The design of the HIL-rig consists of a constructed wooden structure with space for mounting various hardware components. The approach was to, primarily, mount components by the use of Velcro tape and otherwise to use screws to attach heavy components. The overall purpose of such a test bed is to make it possible to implement new solutions quickly with regards to existing software infrastructure and to be able to put the new hardware efficiently into work. Initially, the hardware installed on the HIL-rig consists of a front node, rear node, autonomous system node, PC, APU, fusebox, power supply, sensors, brake system, steering system, steering wheel, dashboard and a Nvidia Jetson AGX Xavier.

The wooden structure provides a surface to attach various hardware components to. This surface needs to be comfortable and ergonomic to work with, and give an easy overview of the system when building and troubleshooting. To achieve this the rig is designed with a plywood top board which can be locked in two different angles. One of the angles is horizontal to use when placing and attaching components. The other is a display angle, where the top board can be tipped 73 degrees to allow an easy overview, for example when planning or showcasing.

A number of different designs were evaluated. A detailed production of different concepts was deemed unnecessary since time was of the essence. With a sketch ready, wood, screws and fittings were acquired from a hardware store. The Chalmers Revere facility, where most of the project was conducted, has a well-equipped workshop with all the tools necessary to cut, measure, drill and attach the parts to the wooden

structure. Some issues concerning fastening methods and rigidity had to be solved during the building phase. In the end, however, no additional material than planned had to be used.

In order to acquire the correct components, the documentation done by the CFSD team of the real car was thoroughly studied. Since the documentation was not entirely complete it was also necessary to gather information directly from the people involved in building the car. The biggest obstacles with this kind of work method was identifying who has been responsible for a certain function and then clearly communicate what is needed from them.

There already was a PCB and some components for the fuse box in the Chalmers Revere facility. In order to create a copy of CFSD's fuse box, the PCB and its components were soldered together. In order to make the process easier, CFSD's fuse box and the electronic design was used as a template, as seen in Fig. 3.1. By looking at the electronic design the purpose of each fuse could be discerned.

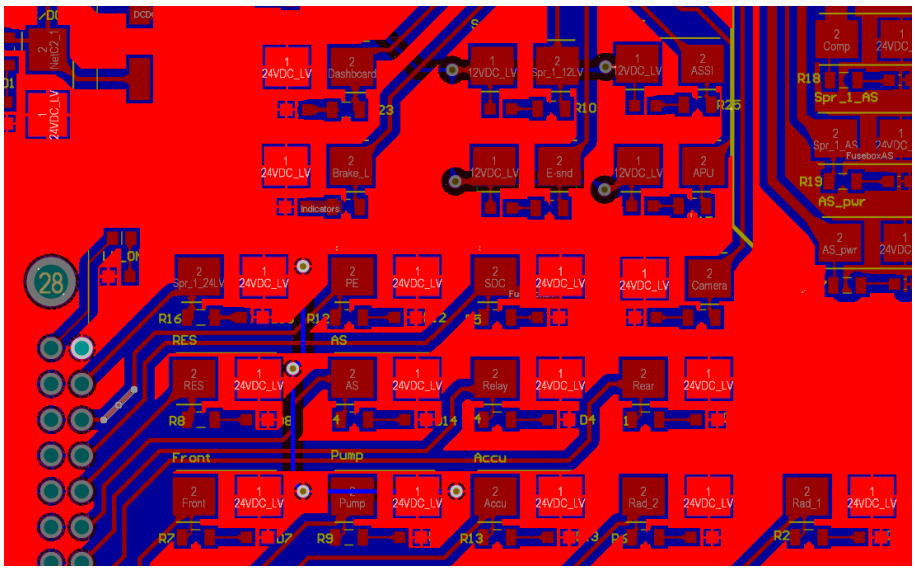


Figure 3.1: The circuitry for the fuse box design generated in the software Altium.

After the fuse box was soldered, testing was required. This included checking for continuity and checking that the diodes lit up properly. Continuity was checked by measuring over input and output and then over each fuse holder. The diodes were checked by applying a small voltage over each diode to see if they lit up.

The containers for the front and rear node, steering wheel, relay box and autonomous system node were all manufactured using a 3D printer. The complete design was saved as a *.stl* file, and Simplify3D was used to prepare the 3D-model for the printer. The containers were then mounted to the rig and the PCB's installed in the containers.

3.1.1 Creating the cable harness

To organise the designing of the cable harness the group received an Excel document from Chalmers formula student (CFS) with all their cable lengths and pin-numbers for the different connectors. The Excel document was organised with one tab for each component of the car. Since the CFSD car has some components that the CFS car does not have, these tabs were added by the group. The document was organised so that all of the data that the group collected, or verified as correct was marked green. This way the CFS information could be kept for reference until new data was collected.

The designing of the cable harness started by measuring all cable lengths in the car. This was done by placing out the different components in the physical car and then using measuring tape to measure the distance between all components that would need some kind of cable between them. Each measured length was then put in the Excel document received from CFS.

To get all the pin-numbers, the group studied documentation of the circuit boards of the car. From these files the pin number could be found and was then put in the Excel document. Finally a standard for the colours of the cables was decided so that the cables would be easy to separate, depending on what type of cable it was, see Tab. 3.1.

CAN high	CAN low	Ground	Power	Sensors
Purple	Yellow	Yellow/Green	Brown	Gray

Table 3.1: The decided colour standard for the cables.

The grey cable which goes to the sensors includes three cables. These are 5 V which is a brown cable, a ground cable which is green and a white cable for sending signals. For CAN the CAN-High and CAN-Low, the cables needs to be twisted. To do this a vise and a screwdriver was used. The ends of the cables were taped together, then one end was put in the vise, and the other one in the screwdriver. The screwdriver was then used to twist the cables.

There was little to no experience of wiring in the group when the project started. The experience in CFSD was also very limited, which led to the whole wiring process taking a lot of time. The group had to figure everything out by trial and error. During this process some help was received from CFS when one of the group members visited them to discuss wiring and how it should be done. After that an instruction, which can be found in App. A was written by one of the group members, including pictures of how the wiring should and should not be done. This enabled more people from the group to learn the process which quickened the work rate.

To control the brake light the wiring needed to be adapted, since the LED-strip that was used was a pull down LED-strip. This means that it is powered by a constant of

12 V and the colour, or RGB colour combination that should light up, is grounded. To control this a TIP120 transistor was needed. The LED-strip was connected to 12 V, the R-contact (red) was connected to the collector of the transistor, and the emitter of the transistor was then connected to ground. Finally, the base of the transistor was connected to the brake light, to enable signal from the rear node. All these connections can be seen in Fig. 3.2. By doing this the transistor closes the circuit when the rear node sends a brake light enable signal, turning on the brake light.

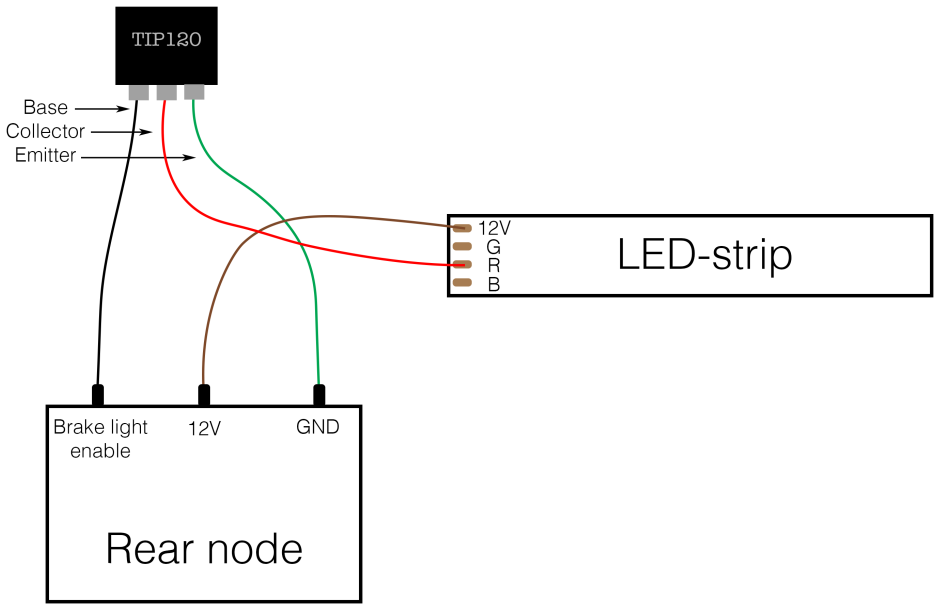


Figure 3.2: Electrical schematic between the rear node, a transistor and the brake light

3.2 Software

The APU and the PC are essential components of the HIL-rig, and it is vital that the software is both as up to date as possible and not bloated by unnecessary programs. It is also important that there is an easy way back to a steady state. To achieve this, employees at the Chalmers Revere facility developed a set of scripts to wipe a devices' hard drive clean and install a fresh Linux environment with device-specific settings. To deploy these scripts and initiate them on the device system, a *preboot execution environment* (PXE) is used. PXE is a client-server environment which allows a device to boot from a network. When the device boots from the PXE server a minimal Linux environment called *Tiny Core Linux* boots up and engages the installation scripts to wipe the hard drives and subsequently installs the *Arch*

Linux environment on the device. Alongside the installation a pre-made *docker-compose.yml* file is included and when it's automatically started the Docker images, that are the building blocks of the microservice architecture, are downloaded.

This process makes it easy to ensure that all parts of the system are up to date with each other. Only a change in the version number for a specific microservice in the *docker-compose.yml* file is all that is required for a new version to be downloaded automatically on the next system reboot. For purposes such as our HIL-rig it is even more useful to make sure that the HIL-rig, and the car it is based on, are running the same software. A simple change in the *docker-compose.yml* file will make all copies of a device update their software to the new version the next time they reboot.

For the two components on the HIL-rig, where this method isff used (the PC and the APU), the process of booting differs slightly in detail although both follow the same basic idea of booting from a PXE server.

The APU automatically boots from its memory and not a network, so some manual setups were required before the installation process could begin for the first time. As the APU has no way of connecting to a monitor of its own, a connection between a PC and the APU was established through the serial port on the APU. The necessary configurations for the APU to connect to the PXE was then be applied through a shell window on the PC. The APU could then be booted from the PXE. This installed all necessary software and the APU was then ready to boot from its own memory to be used in the HIL-rig system.

The process for booting the PC was very similar to the one for the APU but instead of connecting to a separate computer, a screen was connected to the PC to monitor the progress of the installation. The necessary settings for the PC to boot from a PXE server were applied and the PC was connected to the internet. The PC could then be booted from the PXE server which was running on a separate computer and the process of installation followed the same path as described previously.

4

Results

During the span of the project, a wooden structure was constructed, the node containers were printed, various PCBs were soldered, the PC and APU were booted and a significant amount of cables were wired. Despite the fact that the HIL-rig was not completed during this time, it can still be used for various tests and simulations. The modular nature of the design makes continued work on the rig straightforward, easy to build upon and perform modifications. An extra shelf for the brake system was retrofitted.

The rig has a working power supply which has been tested with the PC. Most components have been mounted on the rig, either with Velcro or secured with screws, according to Fig. 4.1. All CAN cables have been connected, as well as most of the ground- and power cables. The dashboard has been built and most of its lights and buttons have been connected to their corresponding components.

With the initation the APU and PC, the system now connects automatically to the PXE server and download the latest update everytime it is started. Furthermore, both the APU and PC works as intended, at least in isolation.

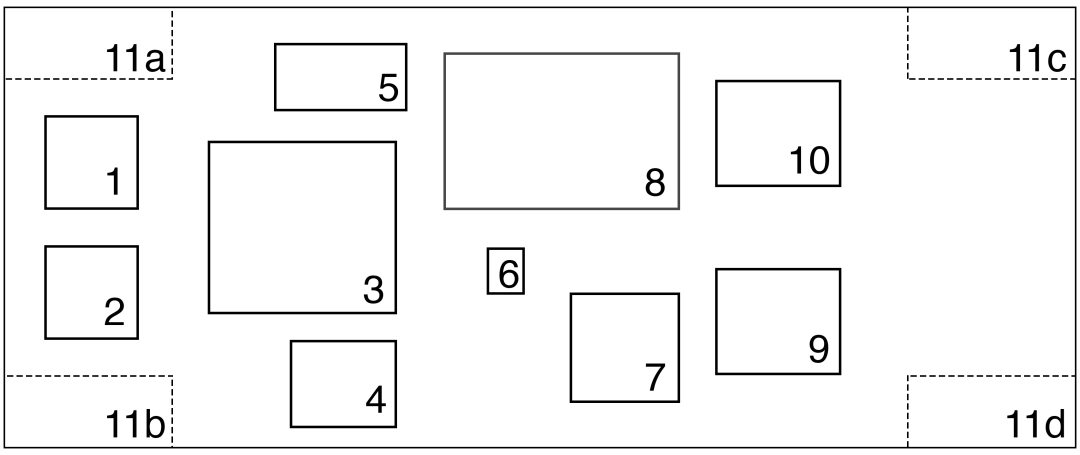


Figure 4.1: The layout of the rig at the end of the project.

The HIL-rig in Fig. 4.1 illustrates the layout of the components mounted in the rig and the names of the numbered components are listed in Tab. 4.1

Table 4.1: Description of the numbers in Fig. 4.1

1	Rear node
2	Fuse box
3	PC
4	NVIDIA Jetson
5	Power supply
6	CAN hub
7	APU
8	Dashboard & steering wheel
9	Front node
10	AS-node
11	Space reserved for sensors

5
Discussion and conclusion

In this section the ethics concerning HIL-rigs, in general, will be discussed. There is also a discussion of the progress during the project, and the effects of external factors. There are also recommendations for further development of the HIL-rig.

5.1 Ethics

In order to simulate and to have a good understanding of the digital twin, it is required to collect and extensive amount of data. In a larger context, it is a fact that more and more activities generate data that is collected and used in ways the public cannot see, nor control. Whilst data can be used for research, in order to improve, things like traffic behaviour, it can also be used in ways that violates privacy. For example, data may be shared with third parties for targeted advertisement and any recordings of sound or video are always sensitive information.

Data collection for commercial purposes, such as targeted advertisement, can be tempting, since such information can be very valuable. However this is a good example of the kind of data collection that could violate an individuals privacy and harm their interests.

The second viewpoint is the lack of transparency of how the sensitive data is handled and processed. In addition, technology develops much faster than policy. There have also occurred discrepancies in regards to data management, leading to data leaks. This has, of course, caused a mistrust between consumers and the digital commercial [6].

However, there are ways to solve problems like these, and different approaches may have different privacy concerns. Monitoring alertness, for example, can be sensitive depending on how it is done. Many car manufacturers use infrared cameras to track the drivers eyes and head movements, usually this is done in a closed loop in the car and no data is recorded or transmitted. Tesla uses sensors mounted in the steering wheel to check if the driver is holding the wheel. This method is not as reliable as using IR-sensors. In fact, it is quite easy to have one hand on the steering wheel and still not pay attention to the road. In Tesla's vehicles there is a camera they call the "cabin camera". The camera records and transmits video footage from inside the cabin. The camera is located in the rear view mirror and both the driver and passengers can be seen in the footage. The camera is turned off

by default but can be turned on if the owner chooses to. This camera is, according to Tesla, used to capture video in the event of a crash or if the automatic emergency braking system is activated. Tesla uses the footage to develop further safety features and make improvements to their software. The question here is also, if they have a camera in the car, why do they not use it to improve the driver alertness monitoring? This camera rises privacy concerns and is not really of any use for the driver.

There are always privacy concerns when one person, or multiple persons are being directly recorded by a company and their faces are visible. The driver, usually the owner of the car, has agreed to share the video but any passengers in the car might not have consented to being recorded. The concern here is that other companies or government authorities may obtain the footage, and someone with bad intentions may also want to obtain video from inside peoples cars as well. Tesla might also use the footage for other purposes than safety development, for example targeted advertising [5].

When developing cars and collecting information for development, it is important to consider in what ways the data is collected. If the alertness of a driver can be determined by IR-sensors in a closed loop system you should not collect video footage from inside the entire cabin. It is also important to consider other cameras mounted on the outside of the vehicle. These cameras record everything around the car and people near the car can not give consent for being recorded. Probably, they are not even aware that they are being recorded.

While user data is not directly collected and saved in this project, it is still important to keep this aspect in mind. This project might be expanded and evolved in a direction where this group has no longer control over it. Therefore, it is important to take ethics into account, already from the beginning.

5.2 Analysis on degree of completion of the project and further development

The accomplishments of this project were not always easily achieved and the group had to deal with various obstacles, originating both from the complexity of the project, as well as from factors during the, as of spring 2021, still ongoing COVID-19 pandemic.

The base of this project has been the Chalmers Revere facility at Lindholmen in Gothenburg, Sweden. This is a space that is shared between a number of different projects. Because of a restriction on the maximum number of people allowed in the lab, only a fraction of the group could perform practical work on the HIL-rig at once. This could largely be compensated for by careful work planning and flexibility in terms of the work site. For example, a team member might borrow a small amount of equipment from the lab to work with somewhere else. However, it proved

challenging to make up for the value of quickly and easily being able to see every member in person. Digital tools like the conference platform *Zoom* and the texting platform *Slack* were helpful, even if they came with a delay of feedback and a slight distortion of communication. In general, he group adapted well to the special prerequisites.

Building the rig involved ordering a lot of different hardware components from around the world. The delivery times were often much longer than anticipated. The project coincided with a global shortage of semiconductors and the blockage of the Suez channel, but it is only speculation that those events might have had something to do with extended delivery times. To cope with this, some components had to be adapted to work in the system. For example, some of the connectors that were used were of the wrong type. Also, since CFSD already possess a lot of hardware components, it was possible to avoid buying expensive PC parts like GPU and CPU. As of writing, these components are still in shortage and would otherwise likely have been a source of delay.

Apart from external factors, some examples of things which required a lot of work is the wiring, setting up the software on the APU, finding and compiling knowledge from the CFSD database and, generally, getting well acquainted with the project during the start up phase. Two things learned from this are that documentation and communication are very important practises. If a part of the group gain a skill, this must be passed on to the rest of the group in order to avoid spending unnecessary work time. When this is achieved, productivity can increase.

The biggest obstacles regarding wiring were the large amount of cables in the car to replicate, to build and add connectors, and estimate the length of these without a finished car to compare with. Because of communication delay in the cables, it is important that they are of similar length to the real car for the HIL-rig to mimic it as closely as possible. To produce cables of high enough quality also proved to require some dexterous skills.

Overall, there is still more work to be done before a complete HIL-rig can be presented. The details on what is left to develop is presented next.

Even though active development of the rig was halted for this group, the hardware components that were ordered will probably continue to arrive for some time to come. Therefore a lot of work can likely be done as soon as development is resumed, either by CFSD, or a future group of bachelor students. As of the time of writing there are primarily a lot of nodes and sensors which have yet to arrive.

The first step after getting familiar with the project is to organize components and attach them to the wooden rig. This should be a rather fast process and the biggest obstacle will probably be for the new group to identify each of the components. It is likely that it will be very important to test every component before continuing in order to save troubleshooting time in case of problems with the hardware. Finishing

the wiring is presumably the following challenge and will require some additional knowledge about the components and what they communicate. It is also necessary to be able to build new cables of decent quality. A guide on this matter has been produced by this group and can be used in the future development.

After the hardware is in place, or at least along side the fitting of the cables, the software for APU and PC needs to work with the whole system. Most software is already written but in order to implement it, some basic Linux knowledge is important. A suggestion on what to test first is if communication can be established with an arbitrary node and that communication over the CAN-network is executed correctly.

Soon after this the rig will probably be in a state where it can help its user to generate new knowledge. This could be done by performing various tests on performance over time, by switching out components and running experimental software. When the physical rig is completed, the last step is to implement the digital twin computer and server, where parts of the car which are not physically present in the rig are simulated, and where data logs from the real car are received.

Some of the components, which have not yet arrived, are different kinds of actuators. Actuators are not strictly needed in order to test other hardware, as they do not affect how the rest of the system works. Instead, they receive instructions from the system and move accordingly. As such was it not obvious to use so many actuators in the system, since they're quite expensive. However, using them in the system would give visual confirmation that they work in the way intended. As work continues to progress more actuators should probably be installed, but it is not assessed to be something which has to be prioritised.

As previously mentioned, it seems like testing a NVIDIA Jetson unit, and evaluating whether it is a good idea to use such a unit in the car, could be one of the first tasks for the rig. However, the future use of the rig depends on the needs of CFSD and can only be a subject of speculation. An attempt on this follows in the next section.

5.3 Future usage of the HIL-rig and conclusion of the project

With HIL, real time simulations can be performed and evaluated immediately rather than waiting for the car to be shut down and then having to download the data from its PC. By acquiring data in this manner, the capacity to interpret and process it increases. One way to simplify this is to develop a GUI for certain parameters.

The rig will probably come to its best use, and save its user a lot of time, when all the complete digital logs from testing of the real car is automatically processed and saved by the rig. This will give deeper insight to the performance of the car. If something is not working as intended, the software can be updated and the same data log can be played again to immediately see what will happen by using the new

code.

By having similar components to the actual car, the HIL-rig can provide a much faster way of evaluating technical solutions. This makes it a quick way for designers to try out and verify different software solutions, encouraging an agile development. This may also result in a decrease of the need to perform tests with actual vehicles and instead put more time and effort into software implementations.

An important outcome of this project is that both the PC and the APU has successfully been set up in the system to automatically install the latest software at start up, exactly like the real car. This is of great benefit for the CFSD-team as it is a big step towards routinely taking advantage of all the pros associated with the concept of HIL.

By making sure that the car and the rig uses exactly the same software, the testing process is eased. For example, if the rig was not using the exact same software, a frequent comment on the test track could be “But it worked on the rig yesterday”. If everything is as similar as possible, differences in run time can be avoided.

Also, knowledge gained in the process of developing the rig can in many cases be applicable to the CFSD-car and contribute to getting it ready for competitions. This is especially true in the case of wiring and setting up software in the system.

The project has occasionally challenged the group with difficulties and setbacks. These have been overcome by cooperation, communication and by trying to match the problem with a suitable person to target it, often based on the different backgrounds and the knowledge possessed in the group. While the final goal of a fully working HIL-rig is yet to be reached, the work has come a long way. The results achieved will most likely be of great help to CFSD as well as provide the team with knowledge and experience on working with HIL and digital twins.

Bibliography

[1] HIL-rig Simulation control lab
https://hil-simulation.com Accessed 2021-05-10

[2] NVIDIA Developer, Jetson AGX Xavier Developer Kit, 2021
https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit.
Accessed: 2021-04-15.

[3] Berger, C., Nguyen, B. and Benderius, O. (2017) Containerized Development and Microservices for Self-Driving Vehicles: Experiences & Best Practices, *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, 7-12, doi: 10.1109/ICSAW.2017.56.

[4] VirtualBox - About
https://www.virtualbox.org/wiki/VirtualBox Accessed: 2021-04-27.

[5] Keith Barry, Tesla's In-Car Cameras Raise Privacy Concerns(2021-03-22)
https://www.consumerreports.org/privacy/teslas-in-car-cameras-raise-privacy-con

[6] Internetsociety, Policy Brief: Principles for Responsible Data Handling(2019-06-02)
https://www.internetsociety.org/policybriefs/responsible-data-handling/

[7] IBM, What is a digital twin?, Accessed 2021-05-11
https://www.ibm.com/topics/what-is-a-digital-twin

[8] NI, What Is Hardware-in-the-Loop?, 2020-12-17
WhatIsHardware-in-the-Loop?

[9] Examining the agile cost of change curve., Accessed 2021-05-01
http://www.agilemodeling.com/essays/costOfChange.htm

A

Wiring instructions

Cable instructions

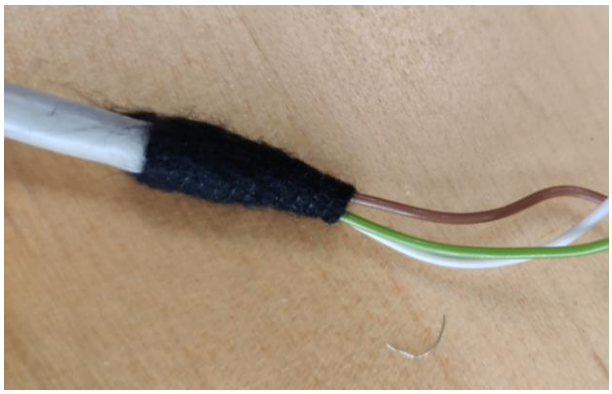
Basics

When cutting a cable you should check the “Kabeldragning” document, cut the cable to the length in the green-marked box. Mark each end of the cable with tape that says component and pin-number.

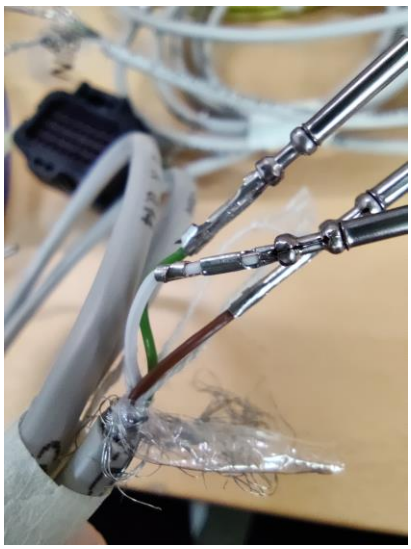
RN 24

Remove the isolation from approximately 7 cm of the cable.

If the cable is shielded, cut of excess shield, strings and plastic and use the black cable-tape to make a nice end.



What not to do



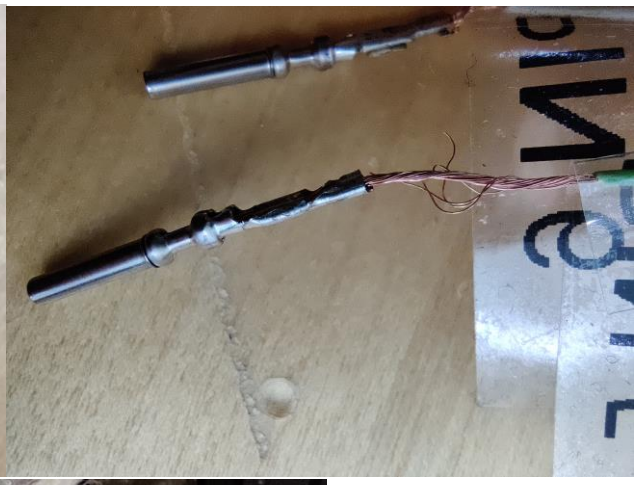
As you can see in this picture the shielding, strings and plastic is just left which makes the cable messy and could possibly create problems for other cables.

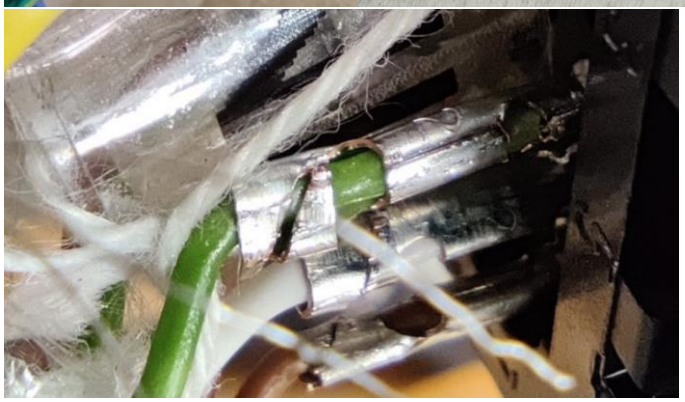
Attaching crimps

The crimp needs to be attached properly. All wires must be covered by the crimp and the crimp should end up round, not flat. The isolation also needs to go all the way to the crimp so that no wires are left exposed. A special tool should be used to attach the crimps, see picture. If possible, depending on cable thickness, the outer “flaps” on the crimp should be around the isolation.

The “flaps” should be on the side of the tool with the number. For most of our cables 1.8 is a good size. For the thin signal cables (the grey with smaller cables inside) 1.4 needs to be used. In this case you will need to push it a bit with a bigger size first so it fits in the 1.4.

What not to do

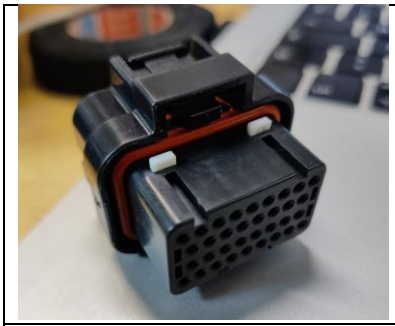




Plug in to superseal connector

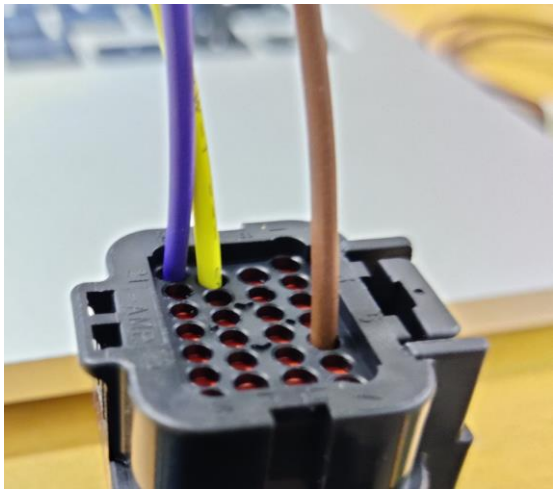
Open and closing the superseal connector

To plug in to the connector you first need to make sure that the connector is open, this is done by pressing the white part on the bottom of the connector. You can use a flat, small screwdriver. When the connector is open the two white pieces on the top is up, when it is closed they are at level with the connector. To close the connector you press the white parts on the top. If it is not possible to close it, there are (at least) one pin that is not properly inserted.

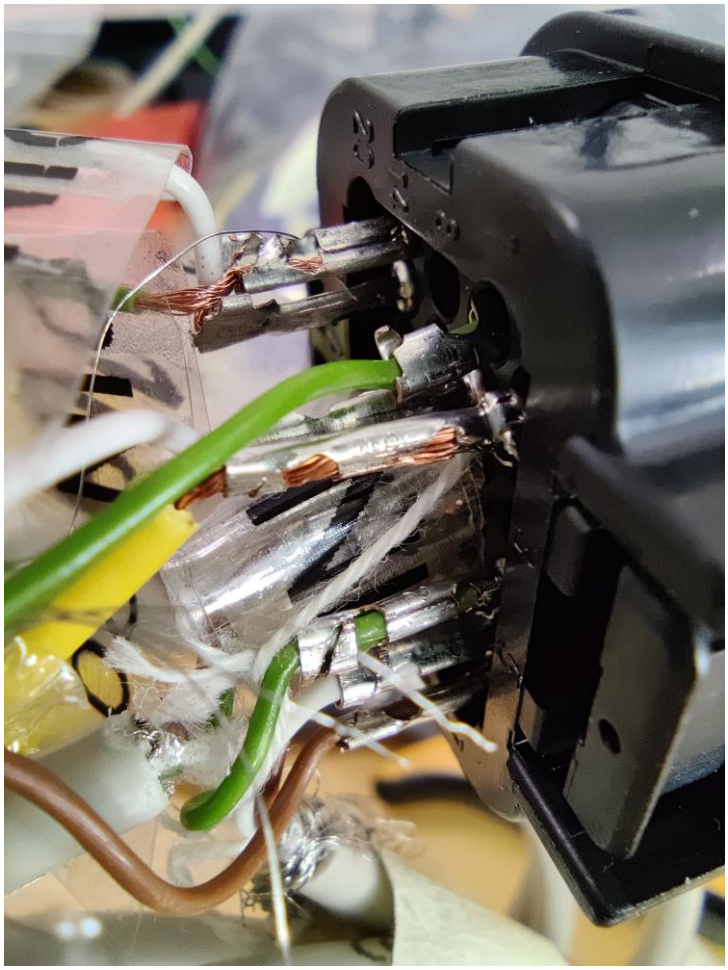
		
Open	Closed	Press this to open

Connect cables

You then need to check the “Electronic nodes and wiring” document to see the placement of the pin-number in the connector. The crimp should go so far in that the connector “clicks” when the pin goes in.



What not to do



The crimps are not inserted properly, there is a lot of excess copper, shielding is left hanging all over the place.

You should never pull out a cable without making sure that the connector is open (by pushing the white part. If you do this, there is a big risk that the crimp will get stuck in the connector.

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS