Design of Electric Power Steering System for Autonomous Driving

Jens D. K. Modvig - s173955, Lucas Lyck - s183685, Mads W.-Kruse - s183683 & Mickael M. Jiang - s180351
Department of Electrical Engineering, Technical University of Denmark
31015 Introductory Project - Electrotechnology

Abstract*—This paper presents a possible solution for an automated steering system to be used by the DTU Formula Student team Vermilion Racing. The solution features an electric power steering system to aid the driver in turning the wheels as well as support for commands from an external source, allowing the car to be steered autonomously. The reader is presented with a short introduction followed by an overview of the system, where covered issues include hardware requirements, choice of components and PCB design with supplementary software. In order to test the system a simplified version of the original design is used as a prototype followed by a discussion regarding possible improvements. Ultimately, it is concluded that while the prototype only succeeded as a proof of concept, the originally presented design should serve as a solid foundation for any future work.

I. Introduction

Electric vehicles are rapidly becoming the norm as more and more car manufacturers roll out their latest electric model. In these environmentally conscious times electric vehicles are very popular, and even though modern gasoline cars are able to drive 10s of kilometers per liter, the complete lack of any kind of emission or pollution makes the electric vehicles superior. It is therefore no surprise that electric cars have made their way into the racing world.

Formula Student is an engineering competition where student teams compete in many different racing categories, including electric vehicles. DTU's Vermilion Racing team currently competes in this category, but would in the future like to be able to compete in the driverless vehicles category as well. For this to become a reality the team will need some kind of way to control the movement of the vehicle without the presence of a driver. This can be done through a remotely controlled power steering-system.

Power steering has become a staple in all modern cars and serves to make driving a smooth and effortless experience. This is done through either a hydraulic system, an electric system, or a mixture of the two. When the driver turns the steering wheel, all three systems amplify the drivers movement, thus greatly minimising the amount of physical effort required to turn the steering wheel - especially when the car is not in motion, or moving very slowly. In electric power steering (EPS)-systems this is done with an electric motor.

A. Problem formulation

An EPS-system must be designed for the Vermilion Racing team's racing car for the Formula Student race. Ultimately the EPS-system will be implemented such that it can be controlled by both a driver, as well as a computer, allowing the car to be entirely autonomous, so that it in the future may compete in the driverless vehicles category.

- How is an EPS-system designed, what kinds of EPS-systems exist, and what are their advantages?
- How do you design a safe system for manual and automated steering?
- How can it be designed within the rules of the competition?
- How can it be designed to have an acceptable range of precision?

B. Problem limitation

Due to time constraints we will only be designing the framework for autonomous steering. The actual implementation of any kind of artificial intelligence and/or autonomy will be done by a different team in the future.

II. System Design

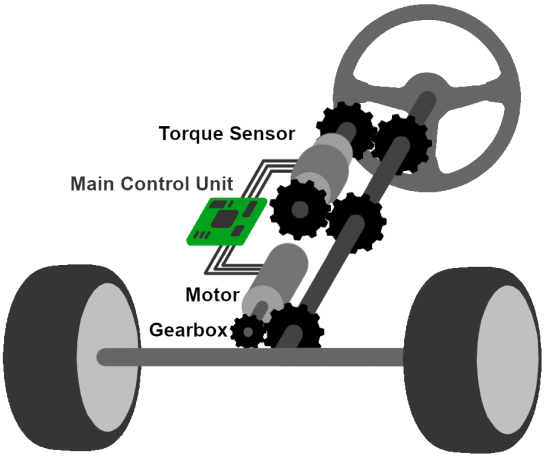

Fig. 1. Overwiev of the EPS-system

A. Measuring driver input

In order to design an EPS-system it must first be possible to measure inputs from the driver. This can effectively be done in two different ways: either measuring the displacement of the steering wheel, or measuring the torque which the driver exerts on the steering wheel.

Measuring the displacement is quite straightforward as this can be done with an optical encoder consisting of a disk with an alternating black and white pattern and a photoelectric sensor. This does however pose a challenge, since the motor, which moves the wheels, will also cause a displacement of the

steering wheel. The system has no immediate way to discern the source of the displacement, and a displacement caused by the motor would be seen the same way as a displacement caused by the driver, resulting in incorrect measurements and thus incorrect movements of the vehicle. A solution to this would be to create a small window of time in which the motor moves the wheels, and where the displacement of the steering wheel isn't measured. This would however severely impact the responsiveness of the system, and would effectively prevent the driver from controlling the vehicle for short periods of time, which is obviously not desirable.

It is instead preferable to measure the torque exerted on the steering wheel. Using a rotating torque sensor it is possible to measure the torque exerted on the steering wheel relative to the steering shaft completely independently of the shafts position, thus eliminating the need for any kind of downtime and keeping the system completely responsive to the drivers inputs at all times. When working with a closed-loop system, such as this one, where the motor is mechanically connected to the torque sensor, it is important to avoid feed-back loops. This makes torque sensing superior as the only measured torque τ, as a result of the motor, is a function of the angular acceleration α of the motor and the rotational inertia I of the steering wheel and coupling ($\tau = I \cdot \alpha$). The threshold for torque should then be set accordingly. Utilizing this, makes it possible to measure very small torques, when using very light steering wheels.

Per section T2.6.2 of the rules of the competition, the driver must always have a mechanical connection to the wheels [1]. It is therefore not allowed to have the torque sensor in-line with the steering shaft, since a mechanical overload (breakage) of the torque sensor could cause a mechanical disconnect, and the driver could lose control of the vehicle. This can be solved by placing the torque sensor on a parallel shaft as illustrated in fig. 1 together with a torque limiting coupling, which begins slipping in the event of torques above a certain threshold. Torque sensors are however often quite bulky and heavy, and a smaller more lightweight solution might be preferable.

Alternatively the torque could be measured using strain gauges in a torsion full-bridge configuration angled 45 degrees from the axis of stress [2]. The strain gauges would be attached directly to the vehicles steering shaft and would thus avoid the usage of a parallel shaft, thus complying with the rules of the competition since the mechanical connection is maintained. When torque is exerted on the steering shaft the strain gauge deforms, resulting in a change in the electrical resistance of the strain gauges which can be measured. Strain gauges are a more robust solution compared to the torque sensor, and only risk breakage at mechanical loads that would deform the steering shaft. In the event of damage or malfunction, they are also very cheap to replace.

Strain gauges are a quite simple and very lightweight solution, however the changes in electrical resistance are quite small, and the measured outputs equally so, meaning it can be difficult to get accurate and precise measurements.

In order to determine the best solution the system will be able to measure torque from both a torque sensor and a strain gauge.

The average driver can exert upwards of 88 Nm of torque [3]. Rounding up, the torque sensor must then able to withstand at least 100 Nm of torque. Adding a safety margin of 100% the torque sensor should preferably be able to withstand 200 Nm of torque.

For comfortable steering the torque sensor must be able to measure torque in the range of ± 10 Nm [4]. This, combined with the required overload torque of 200 Nm, poses a challenge, since very few sensors meet these requirements. Using a torque limiting coupling to protect from mechanical overload, as discussed in the previous section, would allow for a much more realistic overload torque.

For safety reasons and for increased responsiveness, the torque sensor will be sampled 100 times per second [7].

B. Choosing the correct motor

The next step is to choose the motor which will aid the driver in turning the wheels.

Formula Student cars typically need up to 11 Nm of steering effort at standstill [3]. The motor must also be able to turn the steering wheel all the way from left to right (120° to either side for a total of 240°) in less than a second [5]. Thus a motor that can deliver at least 11 Nm of torque with a speed of at least 40 RPM is required [5].

The two primary candidates are a stepper or servo motor. A key difference between the stepper and servo motor is the ability to keep a constant torque at different levels of speed. It is only past 1000 RPM that the servo motor can maintain a higher torque than the stepper motor [6], thus the stepper motor is favorable at the required speed of 40 RPM. Stepper motors generally also have a high accuracy, allowing very precise control of the vehicle, making the stepper motor the ideal choice.

Stepper motors however run the risk of missing steps, and thereby offsetting the steering wheel angle. To avoid this a rotary encoder, or better yet, a motor with a build-in rotary encoder and a compatible stepper motor driver can be used.

To achieve 11 Nm at 40 RPM the power rating of the motor should be 46 W, and adding a 100% safety-factor for both speed and torque, the motor must then have a power rating of 184 W [5]. This is however an ideal case, and it is more realistic to simply have a 100% safety-factor for the power, thus giving the motor a required power-rating of 92 W.

In reality a motor with these exact specifications might not exist, however given that torque and speed are dependent the desired speed and/or torque can be achieved with appropriate gearing.

A possible motor and driver combo would be AM34HD1404-E1000D and CL86T, which can deliver around 2.2 Nm at 360 RPM. Using a 6:1 gearing it is then possible to achieve 13.2 Nm at 60 RPM. This results in a power-rating of 83 W, which almost meets the desired power-rating.

C. PCB Design

The PCB design for the EPS-system consists of two PCBs: the Strain-Gauge-PCB and the Main Control Unit-PCB (MCU-PCB). The signal from the strain gauge is very small, making it prone to noise. In order to mitigate this, an amplifier must be placed as close as possible to the strain gauge. For this reason, a separate PCB is dedicated to amplifying the strain gauge signal. The MCU-PCB is used to handle incoming signals and is in charge of controlling the entire system.

The car battery will supply the PCBs from either a 12V or a 24V power line. Therefore, the PCBs feature an LM43600PWPR buck converter to power the PCBs at 5V. They also include reverse polarity protection [9] and a fuse, to protect the boards against any accidental mishaps.

Both boards feature a microcontroller - an Arduino Pro Mini - responsible for all the calculations and managing the other components on the board. The Arduino is based on the ATmega328 chip which is a requirement from Vermillion Racing.

Both boards support CAN-communication (Controller Area Network communication). CAN is a very stable serial communication bus designed for industrial and automotive applications, and is the standard communication type in most modern vehicles. CAN allows microcontrollers and other devices in the vehicle to communicate independently without a host computer. The hardware is implemented using an MCP2515-I/P (controller) and an MCP2551-I/P (transceiver).

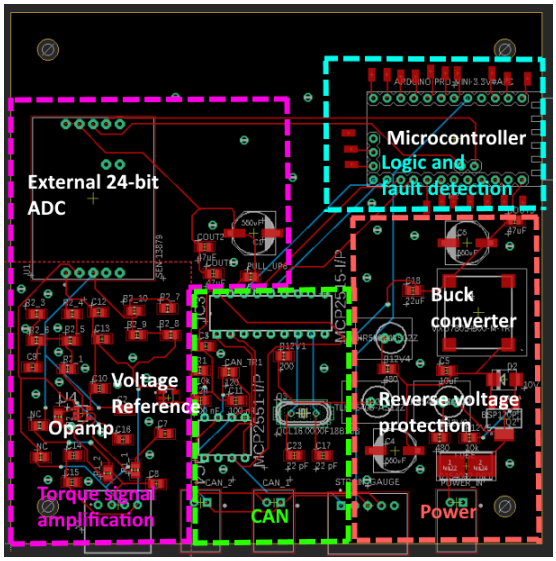

Fig. 2. Strain gauge PCB.
Magenta: Torque signal amplification, Cyan: Logic and Fault detection, Green: CAN communication, Rosé red: Power.

1) Strain-Gauge-PCB: An overview of the PCB can be seen in fig. 2. As discussed the strain gauge-signal is very prone to noise. To reduce noise further the strain gauge signal passes through short EM-shielded-cables, and is then fed into a low-noise operational-amplifier, with a separated analog ground plane and source, which amplifies the signal. The analog data is then converted by an external analog to digital converter (ADC) to achieve the desired resolution.

Torsion Wheatstone bridges typically have a sensitivity of about 1mV/V [2]. Using our 5V analog supply the output will be in the range of $[-5\text{mV}; 5\text{mV}]$. We then need a gain of 500 and an offset of +2.5V to get the output within the range of [0, 5V] so that the ADC takes full advantage of its range. This can be achieved using a differential amplifier configuration with offset (fig. 3).

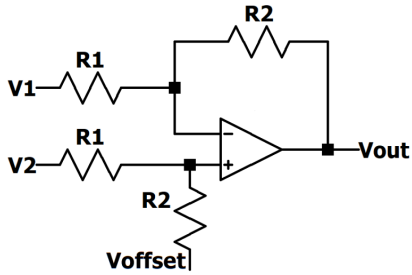

Fig. 3. Differential amplifier configuration with offset.

$$V_{out} = V_{offset} + \frac{R_2}{R_1}\,(V_2 - V_1)$$

If a torque sensor with a known range, eg. 20 Nm, is connected, we can calculate the theoretical resolution by taking the full range of the torque sensor and dividing by the resolution of the ADC

$$\frac{20 \cdot 2\ \text{Nm}}{2^{24}} = 2.38418 \cdot 10^{-6}\ \text{Nm} \tag{1}$$

This is however assuming no noise and perfect amplification, and is of course an ideal case. To ensure the preferred resolution of 0.015 Nm [8] is achieved from the strain gauges the ADC resolution has been overcompensated. Finally the output from the external ADC is then read by the Arduino and the measurement is transferred to the MCU via CAN.

2) MCU-PCB: An overview of the PCB can be seen in fig. 4. The entire EPS-system will be controlled by the Main Control Unit (MCU). The MCU primarily consists of a microcontroller - in this case an Arduino Pro Mini.

A passive-divider was added to the MCU, allowing the direct usage of a torque sensor with a built-in amplifier together with the internal ADC of the Arduino. The analog signal from the torque sensor will be a signal in the range $[-5\text{V}; 5\text{V}]$. The passive divider then converts the voltage from $[-5\text{V}; 5\text{V}]$ to $[0\text{V}; 5\text{V}]$ which the Arduino can read.

As discussed the MCU will also receive measurements from the strain gauge-PCB. Both the torque measured by the strain gauges, as well as the torque measured by the torque sensor, will be read by the MCU's Arduino and converted to a set of commands which instructs the motor to move the wheels of the vehicle in accordance with the drivers input.

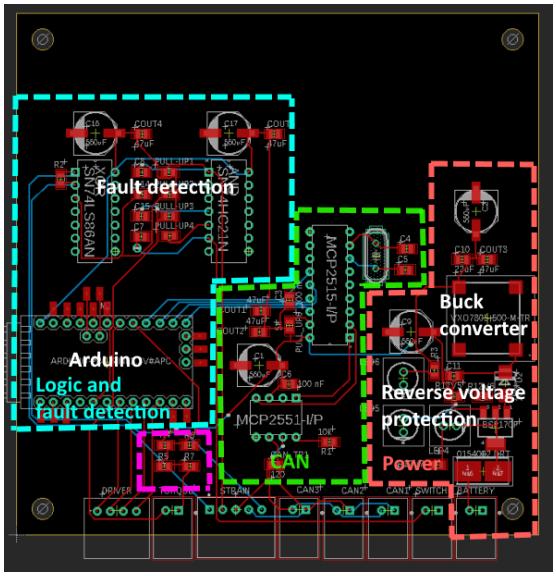

Fig. 4. MCU PCB.
Magenta: Passive divider for torque signal, Cyan: Logic and Fault detection, Green: CAN communication, Rosé red: Power.

While the system has support for both torque sensors and strain gauges it is not intended to use both forms of measurement at the same time. It is only in order to be able to test the two measurement solutions and choose the most optimal.

The MCU also has a fault detection section. The fault detection utilizes active-low logic, meaning the signals are high by default. Thus failures or disconnects of the alert signals themselves will also signal an error. In case of any detected fault the power to the motor driver will be cut off, thus effectively turning off the EPS-system and allowing the driver to take manual control of the vehicle. The fault detection checks the following:

1) Disconnect between the strain gauge-PCB and the MCU-PCB. VCC and GND from the strain gauge-PCB are sent alongside the other signals to the MCU. By XOR-ing VCC and GND it is possible to check if the two boards are disconnected.
2) Torque measurement from the strain gauge-PCB is out of range. The range is configured on the external ADC on the strain gauge-PCB.
3) The motor drivers alert signal, which goes high if the motor driver detects that the motor isn't moving when instructed to, e.g. the motor is stuck.
4) Software alert set by the Arduino on the MCU, e.g. startup problems, failed calibration of the torque sensor/strain gauge etc.
5) A manual shut-off switch placed inside the driver compartment.

D. Programming

The microcontrollers were chosen because of their popularity and availability, and as such it is possible to make use of a wide variety of libraries and examples. The programming of the ATmega328p is nearly always done using the Arduino IDE in Arduino language, which is an extension of the popular language C++.

Each microcontroller on each PCB, will have their own software, each with their own specifications. The microcontroller on the torque-measuring-PCB will be requesting a reading from the external ADC using Serial Peripheral Interface (SPI), convert it to Nm and transmit it over CAN all with an update-rate of 100 Hz [7]. For this the Arduino "SPI" and "mcp2515" libraries could be used. The MCU microcontroller will be reading incoming CAN-messages at 100 Hz from the external controller address and the strain-gauge-microcontroller address, giving priority to the strain-gauge-microcontroller address. The MCU keeps track of the steering wheel angle at all times and changes the target angle if a torque is detected. It is only when no torque is detected that the external controller is considered. As such the driver always has priority over the eventual autonomous system.

To communicate with the MCU any external controller must send a 1 byte CAN-message to the MCU's CAN address. The byte is interpreted at 100 Hz as a two's complement 8-bit integer representing the requested steering wheel angle in degrees. Negative and positive values represent clockwise and counter-clockwise respectively and are defined in the range of $[-120; 120]$. In case of any measurements in the ranges $[-128; -121]$ or $[121; 127]$ nothing currently happens, however these could be treated as a software error for the fault detection system.

III. Prototype

A. Overview

Due to the COVID-19 lockdown we have not been able to access the laboratory, and by extension the vehicle. As a consequence we have been unable to develop a proper mechanical solution. For this reason the prototype is a bare-bones version of the original design which doesn't quite meet the desired expectations. The goal of the project has therefore been reduced to a proof of concept and is not implemented on the actual vehicle.

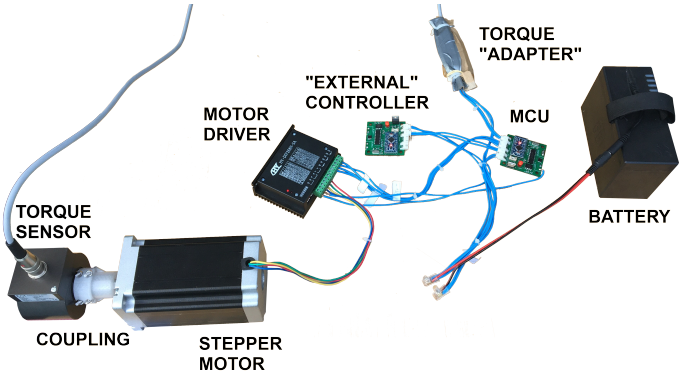

Fig. 5. Overview of the prototype system. Note that the "external" controller is simply a copy of the MCU, whose only purpose is to send commands to the MCU over CAN in order to simulate an external controller.

Therefore, several of the key components have been replaced with components already available in the Vermilion Racing workshop. This includes the torque sensor, stepper driver and stepper motor. An overview of the design can be seen on fig. 5.

The torque sensor DRBK-II doesn't measure in the desired range of ± 10 Nm and instead measures ± 50 Nm. Additionally the resolution is also lower than the set standard of 0.015 Nm [8].

The stepper motor is a Nema 34 HS5435C-37B2 and the stepper driver is a TB6600 Stepper Driver. Unfortunately these don't live up to the set expectations. As a consequence, there is no gearing of the motor as in the original design. The stepper driver can only supply 5 A compared to the desired driver which could deliver 9-10 A. Additionally, there is no encoder feedback support and therefore no way to regulate the motors position in case of skipped steps. The motor driver can also only deliver 60 W compared to the desired 92 W discussed in section II.B.

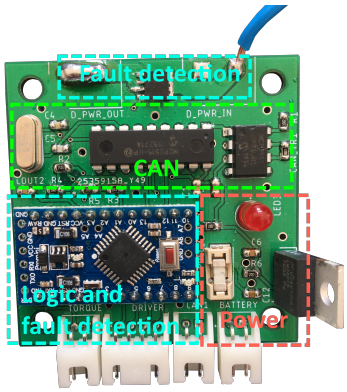

Fig. 6. Prototype PCB. Note that the MOSFET at the top of board was initially designed to switch the motor driver's supply, but has since been repurposed to switch the motor driver's ground. It was discovered that switching the high-side of the motor driver wasn't possible with an N-channel MOSFET. Instead a PMOS-transistor could be used. This, however, turns the driver on by default which is undesirable as it could be dangerous and potentially turn on the motor when the MCU is un-powered.

The prototype only uses one PCB-design, which can be seen on fig. 6. The PCB is designed to read the output from the torque sensor, and does not have support for strain gauges. The rest of the design follows the MCU-PCB described in section II.C.2) with a few exceptions. The PCB is powered by a linear voltage regulator LM7805CT/NOPB instead of the mentioned buck converter. In order to keep the prototype simple there is no reverse polarity protection and the fault detection system has been reduced to the software alert.

B. Implementation

The implementation was done in several steps. First the components were tested, to ensure that they are working as intended and to help isolate any hardware problems, since troubleshooting the entire system at once can be tedious and nigh impossible. Later the construction of the final product could take place followed by the programming of the micro controller.

First the torque sensor is tested, since the entire system won't work if no driver input can be measured. To do this a small demo setup is built on a breadboard, to see if it is possible to convert the torque sensors output from [-5V, 5V] to [0, 5V]. Afterwards an Arduino is connected to see if it is possible to interpret the voltages as realistic torques.

Another crucial part of the system is the motor and motor driver. Again a small demo is built on a breadboard, where the motor is sent a series of commands from an Arduino.

Once the torque sensor and motor both work correctly, they can then be connected to the final PCB. Once connected to the PCB the same tests are performed once again to ensure that everything still works as intended.

To test the entire system as an EPS-system a 3D-printed coupling is designed to connect the torque sensor with the stepper motor, which can also be seen on fig. 5.

C. Programming

The programming of the ATmega328p was done using the Arduino IDE as planned. In accordance with the discussed design is section II.C a 100 Hz interrupt was set up, to ensure a stable update-rate. The software was set up to measure the torque and read the CAN-input 100 times a second to step the motor to the desired angle. If the torque reading exceeds a threshold the steering wheel overrules the external computer and lets the driver take control of the EPS-System. It was quickly observed that without an FPU (floating-point unit) the 8-bit micro-controller wasn't able to handle all the floating point arithmetics to keep up with the data streams. As such the code was changed to use direct reading/writing to registers as well as 18.14 fixed point format to improve the performance. This is however not optimal on an 8-bit machine.

To read the torque it was necessary to create a function that converts the analog signal read by the Arduino back to the corresponding torque. The torque exerted on the torque sensor T is first converted into an analog signal A between -5 and 5 volts by the sensor itself. The analog signal then passes a passive divider to change the output to B in the range of 0 to 5 volts, which the ADC on the Arduino can read and further converts this into a digital number C. The relationship between the torque and the output, and the relationship between the voltage on the ADC and the read number is

$$A = \frac{T}{T_{range}} \cdot 5V, \qquad C = \frac{B}{5V} \cdot 1024$$

Using Kirchoff's current law on the passive divider (fig. 4) circuit we get the following equation

$$\frac{A-B}{1510\Omega} + \frac{5V-B}{1500\Omega} + \frac{-B}{100 \cdot 10^3\Omega} = 0$$

Solving for the torque using the three equations we get the following expression

$$T(C) = \frac{(60653\ C - 30924800)\ \text{Nm}}{614400}$$

which was implemented in software. The measured torque could then be converted into a set of speed and direction commands for the motor driver.

D. Results

Despite the inadequate hardware it was still possible to implement a working prototype which could successfully measure a torque and convert this into a motor speed. Additionally the prototype also has fully functioning CAN-communication.

This video shows a short demonstration of the system. In the video it can be seen that the motor receives a set of commands over CAN to alternate between turning left and right with a couple of seconds between each turn. It also shows how a person is able to manually overrule the received commands and take control of the system. A link to the projects GitHub is provided for future work.

IV. Discussion

A. Microcontroller

The microcontroller barely satisfies the performance requirements, thus restricting any improvements to the resolution of the measurements, as well as the amount of serial communications and debugging possible, without the program running slower and violating the update-rate requirement. A possible alternative is the popular Teensy 4.0 which consists of a 32-bit processor and an in-build FPU allowing for floating point-arithmetics and therefore easier-to-understand code as well as further improvements in the future. The downside of the Teensy 4.0 is that it does not follow the ATmega328p board design, yet the coding of the microcontroller is very similar.

A big improvement of the entire system is to use an external ADC to read the torque output. Using the build in Arduino 10-bit ADC, is not sufficient for precise readings. Using equation (1) we only get a resolution of 0.09 Nm, instead we can get the desired resolution using a 12-bit external ADC or utilizing the build in 13-bit ADC of the Teensy 4.0. This will results in more precise readings in future projects if the torque sensor is used.

B. Motor/Driver

As discussed the motor and motor driver used for the prototype are already inadequate compared to the intended design, however throughout the testing we encountered additional issues with the motor driver. It is advertised as being able to deliver between 0.5 and 4.0 A when supplied with 9-42 V, however when powering the driver from our battery (12.8 V) it was only able to deliver 0.27 A, thus severely limiting the speed and torque of the motor. Replacing these components with earlier listed or similar components is critical for a proper EPS-system.

V. Conclusion

With all of the inconveniences due to COVID-19, the prototype is still successful as a proof of concept. The PCB is able to communicate with the torque sensor and the stepper motor/driver, to produce a direction and speed according to the force exerted on the torque sensor. It can also receive commands from an external source over CAN-communication, while still allowing the user to overrule the external commands and take manual control of the system.

Safety features on the prototype have been restricted to a software alert. Implementing the earlier listed fault detection, safety margin and mechanical solution is critical for a safe EPS-system. The mechanical solution discussed in section II will also allow the EPS-system to follow the rules of the competition. In the prototype the torque sensor and motor are coupled directly, making the torque sensor effectively in-line, which as discussed is against the rules of the competition.

The precision of the system is critical, however due to the lack of encoder and driver feedback, as well as overall sub-par quality of the motor, the prototype is quite inaccurate. Replacing the motor, motor driver, torque sensor and adding an external ADC, with the earlier listed requirements will result in the desired precision.

Unfortunately the prototype did not end up as the solution for the Vermilion Racing team, however the design described in section II should nevertheless serve as a solid foundation for any future work.

References

[1] Formula Student Germany. ”Formula Student Rules 2020, Version: 1.0, Rev-1fcc153”. https://www.formulastudent.de/fileadmin/user_upload/all/2020/rules/FS-Rules_2020_V1.0.pdf, p. 26 Visited 16/03/2020

[2] HBM. The Wheatstone Bridge Circuit. https://www.hbm.com/en/7163/wheatstone-bridge-circuit/ Visited 18/06/2020.

[3] Fox, Steve. (2010, July 10). ”Cockpit Control Forces – or – How Robust Do Driver Controls Really Need To Be?” *Formula Student Germany*. https://www.formulastudent.de/pr/news/details/article/steves-box-of-tricks Visited 05/03/2020.

[4] Roger Hazelden - ”Automotive power assisted steering using optical torque monitoring” https://findit.dtu.dk/en/catalog/2233871449 Visited 05/03/2020.

[5] Seligmann, Albert Simon & Højlund, Frederik Victor. Supervised by Mijatovic, Nenad. (2019, August 31). ”Design and implementation of sensor system for autonomous driving.” *Technical University of Denmark*, p. 14. https://findit.dtu.dk/en/catalog/2463631980 Visited 05/03/2020.

[6] RealPars. ”Servo Motor Advantages And Disadvantages” https://realpars.com/servo-motors-advantages/ Visited 06/03/2020

[7] Johansson, R. et al. (2003). ”On Communication Requirements for Control-by-Wire Applications”. *Semantic Scholar*. https://pdfs.semanticscholar.org/2dc0/3068c8c7a85399518f7ff7536b0fef62f84e.pdf Visited 23/03/2020.

[8] BOSCH. Servolectric® electric power steering system - Torque sensor. https://www.bosch-mobility-solutions.com/media/global/products-and-services/passenger-cars-and-light-commercial-vehicles/steering-systems/torque-sensor-steering/torque-sensor_product-data-sheet.pdf Visited 16/03/2020.

[9] Gustav Ohlendorff Brønd, Anders Juhl Jørgensen & Alex Fihl Hedegaard Nielsen. ”Design og implementering af en robust og strømsparende løsning til den integrerede elektronik til Økobilsrattet.” *Technical University of Denmark*, p. 2.

DTU Electro	Spring 2020	Group ID: 02
Course 31015	Title	Group members
Introductory Project Electrotechnology	Automation of steering CEE01	s183683 Mads Wedendahl-Kruse s173955 Jens Dieter Kjær Modvig s180351 Mickael Mathiias Jiang s183685 Lucas Lyck
Document	Problem formulation	Pages: 1
Version/Status	Version 2	

Problem formulation

An EPS-system must be designed for the Vermilion Racing team's racing car for the Formula Student race. Ultimately the EPS-system will be implemented such that it can be controlled by both a driver, as well as a computer, allowing the car to be entirely autonomous, so that it in the future may compete in the driverless vehicles category.

- How is an EPS-system designed, what kinds of EPS-systems exist, and what are their advantages?
- How do you design a safe system for manual and automated steering?
- How can it be designed within the rules of the competition?
- How can it be designed to have an acceptable range of precision?