NTNU – Trondheim
Norwegian University of
Science and Technology

Hardware and Software Design for the DNV GL Fuel Fighter Vehicles

Ole Bauck

Master of Science in Cybernetics and Robotics
Submission date: June 2015
Supervisor: Amund Skavhaug, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Acknowledgment

I will like to thank my supervisor Amund Skavhaug for guiding me through this project. Thanks to the DNV GL Fuel Fighter team 2015 for their hard work and great companionship. A big thanks to my beloved Marthe for helping me the final days and for who you are to me.

Abstract

Shell Eco-marathon is a competition between universities and schools to make the most energy efficient car. This year the DNV GL Fuel Fighter team participated with two cars, a Prototype car built last year and an UrbanConcept car built this year. The cars used different fuel types, battery for the Prototype car and hydrogen fuel cell for the UrbanConcept car. The work done for this thesis included design of three systems: The overall electrical system; a communication, data acquisition and logging system; and a motor control system. The work done was implemented in both cars, as the electrical systems were similar.

A design for the overall electrical system was chosen early to be able to delegate areas of responsibilities. The design relied on work done by previous teams and the competition rules, along with the desired functionality of the system.

The communication, data acquisition and logging system relied on the work done in the specialization project [2]. This system consisted of a communication module with a 3G modem, sensors in the car for data acquisition and a logging system on a server which allowed real time monitoring of the cars. The list below summarize the different parts that was made for this system.

3G module A module with a 3G modem based on a prototype board made last semester. The work included debugging the prototype board.

Power measurement module A module to measure the power of the propulsion system and solar panels. Additional functionality for turning on and off the propulsion system was included.

GPS A module for retrieving time and GPS position of the car.

Logging program A program showing graphs and text displaying the important data sent from the car. Data shown in the graphs included motor reference torque, speed of the car, current from the motor and current from the solar cells. Motor status, battery voltage, GPS position and time was displayed as text.

The logging system proved to be very useful both when testing the car and during the competition. The speed and the reference torque was especially useful in checking that the motor

control system worked and in deciding a driving strategy. The system worked both in Norway and the Netherlands.

The motor control system was made based on the work done by Simon Fuchs previous semester. The design aimed to control a permanent magnet AC motor. Prototype boards were tested and new PCB designs were made. Besides removing the errors, the new designs were made with an idea that the boards should be robust and easy to connect. Software was made based on a simulation model. The motor controller code was optimized using fix point calculations and lookup-tables to reach a desired time limit. A logging program was made on a computer in order to check that the system was driving the motor in an optimal way. Different controller code was implemented to get the desired behaviour out of the motor control system.

The Prototype car came in 11th place with a result of 482.5 km/kWh, while the UrbanConcept car was not able to finish any of the races. The problems with the UrbanConcept car were due to the fuel cell system shutting down in the middle of the race and an engine breakdown. The engine from the Prototype car was used in replacement for the destroyed engine in the UrbanConcept car. The team experienced several problems with the motor controller in the UrbanConcept car, probably due to exposed electronics that led to malfunctioning components. With much of the time used on working with the UrbanConcept car, it was less time left to optimize the Prototype car any further.

The systems that were developed in this project have been tested and found to be working. The communication, data acquisition and logging system forms a basis for further development including more sensors and better visualization and storage of data. The motor controller can be optimized further in regards to both motor efficiency and the use of standby power.

Sammendrag

Shell Eco-marathon er en konkurranse mellom universiteter og skoler for å lage den mest energivennlige bilen. DNV GL Fuel Fighter deltok i år med to biler, en Prototype bil lagd forrige år og en UrbanConcept bil lagd dette året. Bilene brukte ulike energityper, batteri på Prototype bilen og hydrogen brenselcelle på UrbanConcept bilen. Arbeidet beskrevet i denne rapporten inkluderte design av tre systemer: Det generelle elektriske systemet; Et kommunikasjon, datainnsamlings og logge system; og et motorkontroller system. Arbeidet ble implementert på begge bilene, siden de elektriske systemene var mye det samme.

Et design for det generelle elektriske systemet ble valgt tidlig for å gjøre det mulig å fordele arbeidsoppgaver. Designet baserte seg på tidliger års systemer og konkurranseregelene, samt ønsket funksjonalitet for systemet.

Kommunikasjon, datainnsamlings og loggesystemet var basert på arbeid gjort i prosjektoppgaven [2]. Systemet bestod av en kommunikasjonsmodul med et 3G modem, sensorer plassert i bilen for datainnsamling, og et loggesystem på en server som gjorde det mulig med sanntidsovervåkning av bilene. Listen under sammenfatter de ulike delene som ble lagd til systemet.

3G modul En 3G modul med et 3G modem basert på et prototypekort lagd forrige semester. Arbeidet inkluderte feilsøking av kortet.

Effektmålingsmodul En modul for å måle effekten til fremdriftssystemet og solcellepanelene. Ytterligere funksjonalitet for å skru av og på fremdriftssystemet ble lagt til.

GPS En modul for å hente tiden og GPS posisjon til bilen.

Loggeprogram Et program for å vise grafer og tekst med data sent fra bilen. Data vist i grafene omfattet motorpådrag, bilens hastighet, strømforbruk til motor og strømtilførsel fra solcellepanelene. Motorstatus, batterispenning, GPS posisjon og tid ble vist som tekst.

Loggesystemet var veldig nyttig både under testing og konkurransen. Hastighet og pådrag var spesielt nyttig for å sjekke at motorkontrollersystemet fungerte og i beslutning av kjørestrategi. Systemet fungerte både i Nederland og Norge.

Motorkontrollersystemet ble utviklet basert på arbeid gjort av Simon Fuchs forrige semester. Designet hadde som hensikt å drive en permanent magnet AC motor. Prototypekort ble testet og nye kort ble lagd. Foruten å fjerne flere feil i designet, så ble det nye designet laget med tanke på at kortene skulle være robuste og lettvint å koble opp. Programvare ble lagd basert på simuleringsmodellen. Motorkontrollerkoden ble optimalisert ved å bruke heltallsoperasjoner og oppslagstabeller. Et loggeprogram ble lagd på en datamaskin for å sjekke at systemet opererte motoren optimalt. Forskjellige kontrollerkode ble implementert får å få ønsket oppførsel til motorkontroller systemet.

Prototype bilen kom på 11te plass med resultatet 482.5 km/kWh, mens UrbanConcept bilen ikke klarte å fullføre noen av løpene. UrbanConcept bilen hadde problemer med at brenselcellesystemet slo seg av under løpet og et motorhavari. Motoren fra Prototype bilen ble brukt som erstatning for den ødelagte motoren i UrbanConcept bilen. Teamet hadde store problemer med motorkontrolleren i UrbanConcept bilen, sannsynligvis på grunn av eksponert elektronikk som førte til ødelagte komponenter. Med mye tid brukt på UrbanConcept bilen var det lite tid igjen for å optimalisere Prototype bilen mer.

Systemene som ble utviklet i dette prosjektet har blitt testet og funnet ut at virker. Kommunikasjon, datainnsamlings og loggesystemet danner et grunnlag for videre utvikling med tanke på flere sensorer og bedre visualisering og lagring av data. Motorkontrolleren kan bli optimalisert videre med tanke på motoreffekt og operasjonseffekt.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Shell Eco-marathon (SEM) is a yearly car competition between universities and schools all over the world. There are three competitions across the globe in America, Europe and Asia. This year, the competition in Europe was in Rotterdam, in the Netherlands, from the 21. to the 24. of May.

The cars drive a given distance within a specified time limit and the goal is to use the least amount of energy. The cars compete in different fuel classes: Battery electric, hydrogen fuel cell, CNG, diesel, gasoline and alternative fuels.

I addition to the different fuel classes, there are two general car type classes: UrbanConcept and Prototype. The Prototype class has few rules that restrict the design of the car, which lead to cars that are built for the sole purpose of energy efficiency. These cars are often low and long, with a horizontal driving position. The UrbanConcept class consist of more typical cars. The rules for the design are more specific for this class; the cars have to have lights like normal cars, the driver has to sit straight, there have to be room for baggage, and much more. There are in general more cars in the Prototype class than in the UrbanConcept class because it has less rules.

1.1 The team

The Norwegian team, now called DNV GL Fuel Fighter, participated for the first time in 2008. This years team consisted of 27 people (including the PR team), where six people were writing their master thesis on the cars. The team members were from many different studies, with

Figure 1.1: Shell Eco-marathon team picture.

mechanical engineering, cybernetics and power electronics as the most common.

1.2 The cars

The previous teams has generally participated in the competition with one UrbanConcept car, but last year the team built a new Prototype car and participated with two cars. Both cars last year participated in the battery electric fuel type class. This year a new UrbanConcept car was built, and the team participated with two cars including the Prototype car from last year. The fuel type for the new UrbanConcept class was hydrogen fuel cells. The main argument for changing fuel type was that not all the teams that apply are invited to join the competition. Teams with different cars in regard to both class and fuel type have much greater possibility for joining the competition. The Prototype car can be seen in picture 1.2, while the UrbanConcept car can be seen in picture 1.3.

Figure 1.2: The Fuel Fighter team with the Prototype car.

Figure 1.3: The UrbanConcept car.

1.3 Scope

At the start of the semester an overall plan for the electrical system was made. The electrical system is described in chapter 2. Diagram of the systems in both cars can be seen in figure 2.1 and figure 2.2 on page 8 and page 11.

Originally, the plan for this thesis was to continue the work on the wireless monitoring system started in the specialization project [2] and together with Vebjørn Myklebust act as the head electrical engineers, responsible for the overall electrical design on both vehicles. The person responsible for the motor system last semester had moved back to Germany. The new team members that was assigned the task of implementing the motor controller did not have the required expertise for this. Without assigning someone that could implement the motor controller before the competition, the team would probably not have working cars for the competition. The author therefore also took the responsibility of implementing the motor controller.

The work described in this report can be split into three parts:

- Design of the overall electrical system.
- Design and implementation of the monitoring system, from now on called communication, data acquisition and logging system.
- Design and implementation of the motor controller system.

As the work done on the electrical system was divide between different members of the team, this thesis should be seen in relation with the other thesis's regarding the electrical system [10] [21].

1.4 Disposition

Since most of the work has been practical, the results are presented and discussed in the design chapters. An overall discussion is included before the conclusion. Relevant theory is also presented along with the design in order to make it more easily accessible to the reader.

First, a background chapter presents some general theory that is common for most of the design. Then the design chapters present the system specifications, hardware and software de-

signs, implementation and testing of the completed work. The design chapters are divided into the three different parts for this thesis, described above. After the design, the results from the competition is accounted for and discussed.

Chapter 2

Overview of electrical systems

The overall electrical system design was chosen early to make the different areas of tasks and responsibilities possible to assign. The system for the Prototype car and UrbanConcept car will be presented in this chapter, and the different modules will briefly be described. The system is based on the work done previous years [8] [22].

2.1 The Prototype car

The overall electrical system that was chosen for the Prototype car can be seen in figure 2.1.

Propulsion battery

The Prototype car ran on battery. The rules only allowed one battery in the car, so the propulsion battery supplied both the propulsion system and the accessory system. However, only power used by the motor and motor control system, in addition to power gained by the solar panels, were included in the energy calculations. The battery included a Battery Management System (BMS) to check each cell for over/under voltage conditions and over current protection. A 20A fuse was installed on the positive side on the battery. The battery had a nominal voltage of 46.2 volt.

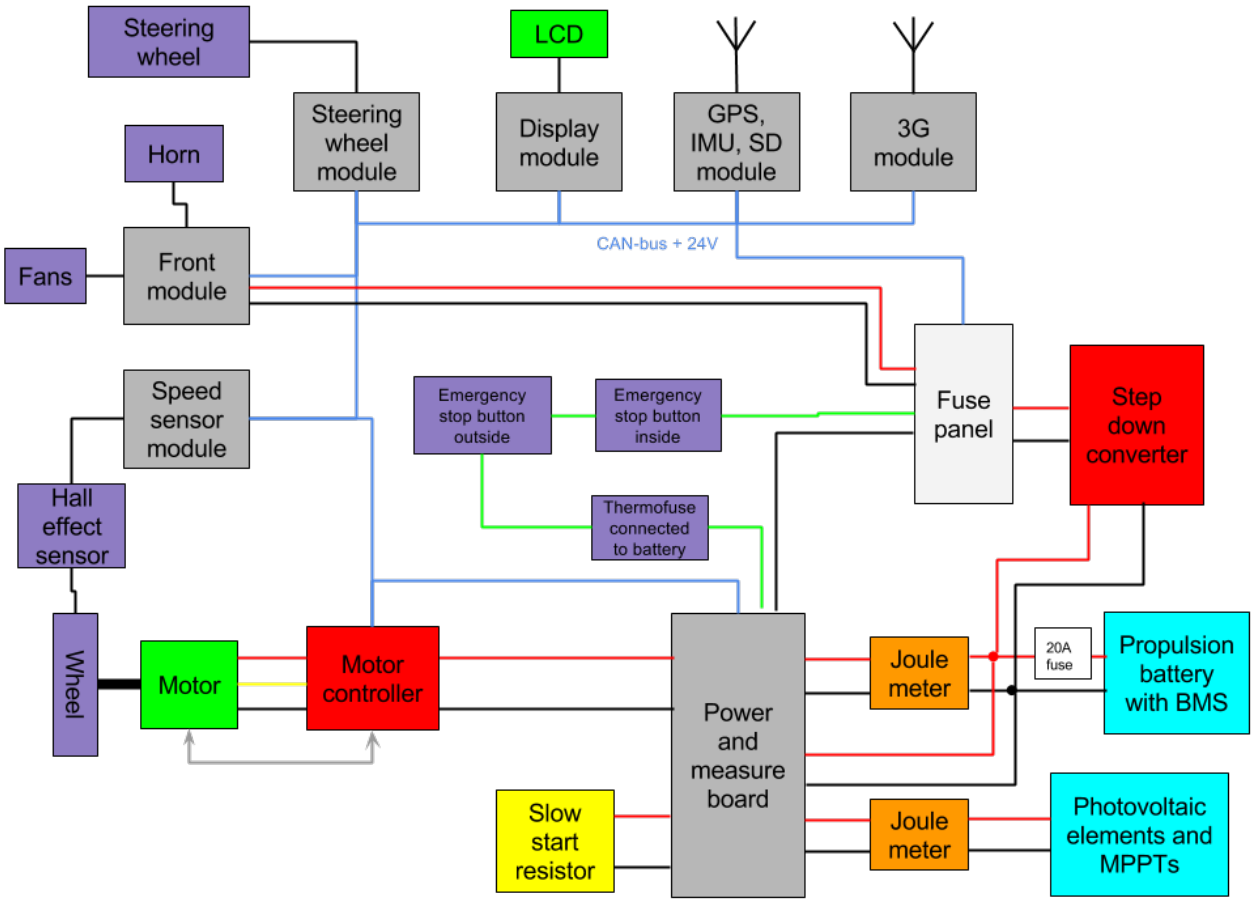


Figure 2.1: Overall electrical system Prototype car.

Photovoltaic elements and MPPT

Since the Prototype car was in the battery electric class, it was allowed to have solar panels. Maximum Power Point Tracking (MPPT) was necessary for adjusting the voltage and getting the maximum power from the photovoltaic elements.

Joule meters

The Prototype car had to joule meters for measuring the energy used by the motor and delivered by the solar panels.

Step down converter

Because the Prototype car had only one battery and the accessory system was designed for 24V, a step down converter was needed to step down the battery voltage. The converter was an isolated step down converter capable of delivering up to 100W.

Fuse panel

The fuse panel distributed the power from the step down converter. Each output had its own fuse for isolating the systems in case of an over-current condition.

Power and measure board with slow start resistor

This module measured the power consumption of the motor system and the power delivered by the solar panels. It also included a relay for disconnecting the motor system. This relay could be controlled either by CAN bus or by external switches. In the Prototype this was the emergency stop system. The slow start resistor was for limiting the amount of current to the inverter the moment the motor system was switched on.

Motor controller with motor

The motor controller with motor was responsible for the propulsion of the car. The motor was connected to the wheel with gears and a belt. The motor controller was connected to the CAN bus for retrieving driver reference and sending status signals.

Emergency stop circuit

The emergency stop circuit was for disconnecting the motor if something happened. One stop button was inside the car next to the driver and one stop button was accessible from the outside of the vehicle. A thermal fuse was also place on the battery to disconnect it if the battery temperature was too high.

3G module

The 3G module included a 3G modem for logging data over the mobile network. Data from sensors in the car was sent over the CAN bus to this module and then sent to a server for logging.

GPS, IMU and SD module

This module included a GPS, an Inertial Measurement Unit (IMU) with accelerometer, gyroscope and magnetometer, and a SD card slot for external, non-volatile storage.

Display module

The display module was for displaying useful information to the driver during the testing and the race. The display module in the Prototype car was an LCD screen.

Steering wheel module

The steering wheel module included all the buttons, knobs and a joystick on the steering wheel. Input from this module, generated by the driver, was sent over the CAN bus to other modules.

Front module

The front module was for controlling the fans and the horn in the car.

Speed sensor module

The speed sensor module measured the speed of the car. In the Prototype car this was done by measuring the speed of the back wheel. The speed was measured by magnets placed around the wheel and a hall effect sensor sensing when a magnet passed by.

Server (logging system)

The system also included a server for logging the data sent from the 3G module.

2.2 The UrbanConcept car

The overall electrical system for the UrbanConcept car can be seen in figure 2.2.

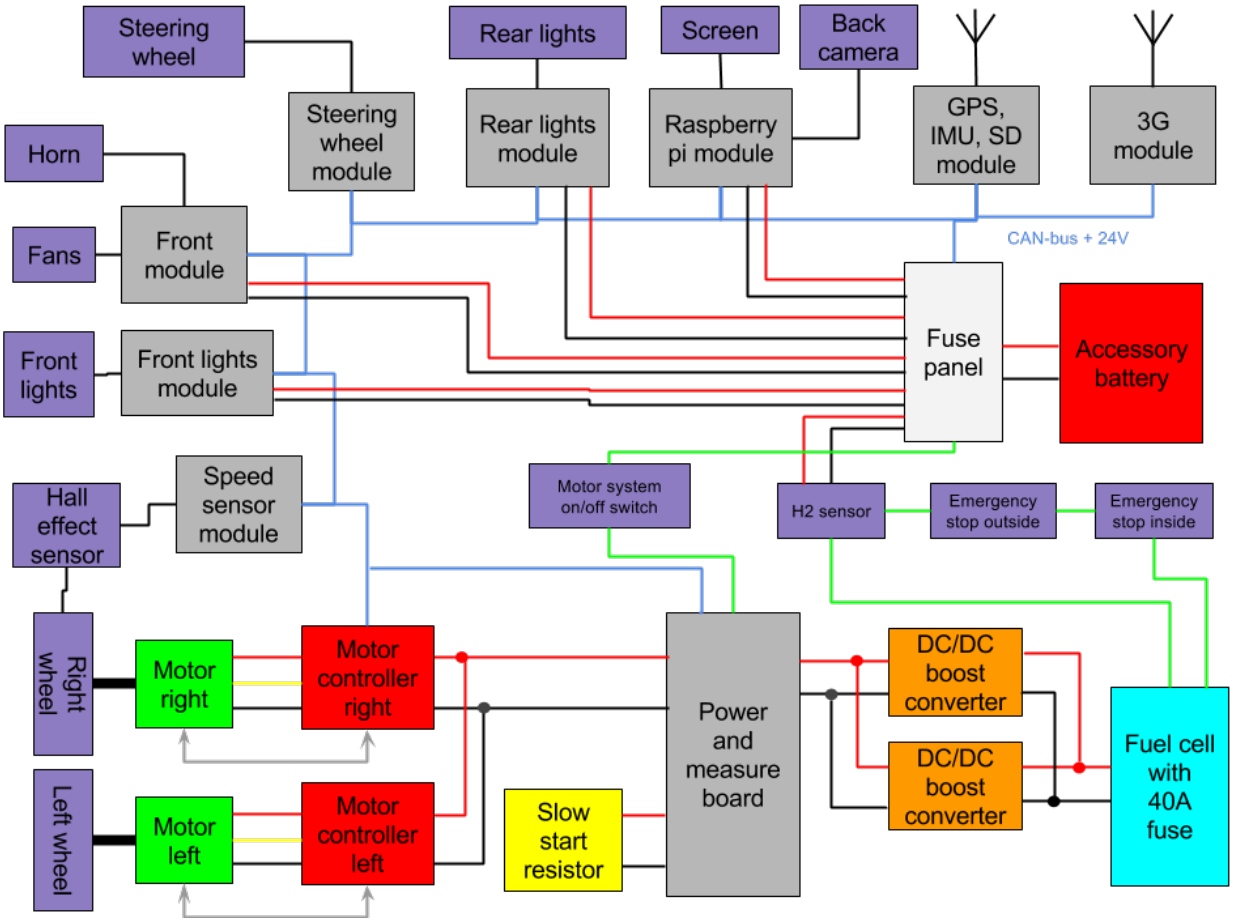


Figure 2.2: Overall electrical system UrbanConcept car.

Fuel cells

The UrbanConcept car had fuel cells as its primary energy source. The fuel cells with a controller was bought from a manufacturer. The output of the fuel cell system was a voltage that ranged from 30 to 46 volt depending on the current [12]. A 40A fuse was connected on the postive side of the fuel cell output.

DC/DC boost converters

The voltage of the fuel cell dropped with the amount of current flowing. DC/DC boost converters were used to keep the voltage at 48V independent of the current. This enabled the motor to achieve top speed. Two boost converters were used for more current capability.

Accessory battery

The UrbanConcept car had a separate battery for the accessory system. The voltage of this battery was 23.1 volt nominal.

Fuse panel

The fuse panel distributed the power from the accessory battery. Each terminal had an own fuse for better safety.

Power and measure board with slow start resistor

This module measured the power consumption of the motor systems. It also included a relay for disconnecting the motor systems. This relay was controlled either by CAN bus or by an external switch. The slow start resistor was for limiting the amount of current to the inverter from the moment the motor system was switched on.

Motor controller with motor

The motor controller with motor was responsible for the propulsion of the car. In the UrbanConcept car there were two motors, one motor connected to each back wheel, and two motor controllers. The motors were connected to the wheels with gears and a belt. The motor controllers were connected to the CAN bus for retrieving driver reference and sending status signals.

Emergency stop circuit

The emergency stop circuit was for disconnecting the fuel cells if something happened. One stop button was inside the car next to the driver and one stop button was accessible from the

outside of the vehicle. For the UrbanConcept, a hydrogen sensor also shut off the fuel cells if the hydrogen density in the air got too high.

3G module

The 3G module included a 3G modem for logging data over the mobile network. Data from sensors in the car was sent over the CAN bus to this module and then sent to a server for logging.

GPS, IMU and SD module

This module included a GPS, an Inertial Measurement Unit (IMU) with accelerometer, gyroscope and magnetometer, and a SD card slot for external, non-volatile storage.

Raspberry pi module

The UrbanConcept car had a screen that was operated by a Raspberry Pi. In addition to information shown to the driver, a back camera was streamed live to the screen.

Rear lights and front lights modules

The UrbanConcept car had to have lights according to the rules. These lights were controlled by own modules both in the back and in the front of the car.

Steering wheel module

The steering wheel module included all the buttons, knobs and a joystick on the steering wheel. Input from this module, generated by the driver, was sent over the CAN bus to other modules.

Front module

The front module was for controlling the fans and horn in the car. In the UrbanConcept, the front module also controlled the windshield wiper.

Speed sensor module

The speed sensor module measured the speed of the car. This was done by measuring the speed of the right back wheel in the UrbanConcept car. The speed was measured by magnets placed around the shaft of the wheel and a hall effect sensor sensing when a magnet passed by.

Server (logging system)

The system also included a server for logging the data sent from the 3G module.

2.3 Responsibilities

When an overview of the system was put down, the different tasks could be distributed. Table 2.1 shows who was responsible for the different tasks of the system. For a more thorough description of the modules not included in this thesis, see [21] and [10].

Task	**Responsible**
Propulsion battery	Same as last year
Photovoltaic elements and MPPT	Bjarne Kvæstad
Joule meters	Installed at race track in rotterdam
Fuel cells	Marius Hofmann with help from Ole Bauck (electrical)
DC/DC boost converters	Ole Bauck (bought)
Step down converter	Simen Hexeberg (bought)
Accessory battery	Same as last year
Fuse panel	Same as last year
Power and measure board	Ole Bauck
Motor controller with motor	Ole Bauck and Simon Fuchs
3G module	Ole Bauck
GPS, IMU and SD module	Ole Bauck
Display module	Vebjørn Myklebust
Raspberry pi module	Vebjørn Myklebust
Steering wheel module	Vebjørn Myklebust
Rear lights and front lights modules	Bjarne Kvæstad
Front module Prototype	Simen Hexeberg
Front module UrbanConcept	Vebjørn Myklebust
Speed sensor module	Bjarne Kvæstad
Server (logging system)	Ole Bauck
Produce more universal boards	Simen Hexeberg

Table 2.1: Tasks and responsibilities.

Chapter 3

Communication, data acquisition and logging system design

3.1 Previous work

The design was based on the work done in the specialization project fall of 2014 [2]. There, a module for a GSM modem and modules for RF communication were made and tested. A module with a 3G modem was also made, but did not work. The RF module was made to meet the need for a more redundant communication than just using a GSM/3G modem, but it was not used further in the work for the thesis, in order to limit the number of tasks that required attention. In addition, it was not certain that the RF modems would have enough range to be used at the race track.

To test the communication with the modem, a server program was written. A GPS module was used to acquire data.

3.1.1 Universal board

Most of the modules in the car was using the universal board, which was designed in 2010 by Anders Guldahl [8]. The universal board is also called the universal module, something that can create confusion, since the term 'module' is used on the universal board in combination with a specialized board. Therefore it will be described in this thesis as the universal board, even

though in some figures it is described as universal module. Figure 3.1 shows a block diagram of the board and figure 3.2 shows the circuit board fully assembled. As the board was an essential part of almost all the modules in the system, the different parts of the board will be described here.

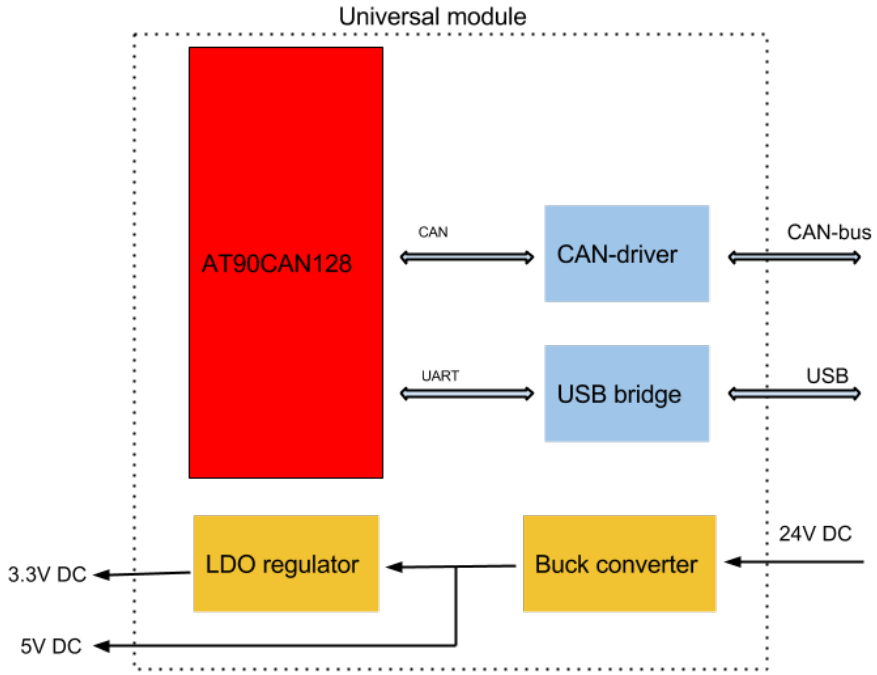


Figure 3.1: The universal board block diagram.

Microcontroller The board had an AT90CAN microcontroller which included a CAN controller.

CAN interface A CAN driver was connected between the microcontroller and two RJ11 connectors (seen in black in figure 3.2). A four wire or six wire cable with a telephone plug could be connected easily to the connectors, which made it easy to connect and disconnect modules in the system.

Power Two of the wires could be used for power and was routed to a buck converter which steps the voltage down to 5V. The input could be 5-36V. A LDO regulator reduced the voltage to 3.3V for the microcontroller and other components.

Connection with other boards A 2x10 pin 90 degree header was for connecting other boards. The header contained most of the pins that were available on the microcontroller.

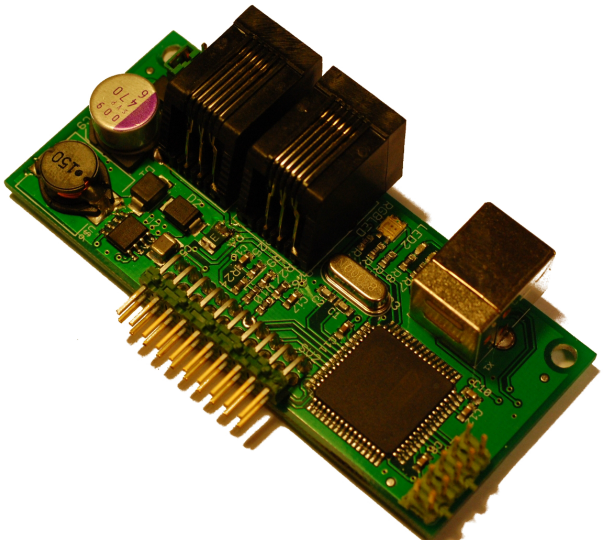


Figure 3.2: The universal board.

USB interface A USB-serial bridge allowed the microcontroller to communicate with the PC over UART. The microcontroller could also get power from the USB cable.

JTAG The microcontroller was programmed through the JTAG header.

Size and boxes The board was designed to be fit in boxes from Sparkfun [1] when connected to another board. These boxes are made to fit Eurocard size PCBs (100x80mm).

3.2 Specifications

The specifications for the system were similar to the ones proposed in the specialization project:

Latency The latency had to be fairly low to avoid lag in the system.

Update frequency The update frequency had to be high enough for the real time visualization of data to be useful. Having the possibility to change the update frequency would be helpful.

Range The range of the communication had to be good enough to cover the whole race track and when testing.

[1] https://www.sparkfun.com/products/8632

Reliable The system had to be reliable and work without any modifications.

Sensors For the system to work, there had to be sensors for gathering the data useful to the user.

Visualization The data had to be visualized in some way. As a minimum, raw data could be shown.

3.3 Overview

The system was divided in three parts:

- **Data acquisition** - Retrieving data from different sensors located in the cars.
- **Communication** - Sending the data wireless to the user.
- **Logging** - Displaying the data to the user.

An overview of the system can be seen in figure 3.3. The data from different sensors in the car was sent to a server over the mobile network. The data was then sent to clients connected to the server and displayed for the user. The data acquisition was a part of the overall electrical system described in chapter 2. Instead of a GSM modem, a 3G modem was used, since it provided more bandwidth and the team was sponsored with three 3G modems.

The modules used to gather the data was (as seen in figure 2.1 and figure 2.2 on page 8 and page 11):

- **GPS, IMU and SD module** - For GPS position and time.
- **Steering wheel module** - For driver input (throttle).
- **Speed sensor module** - For speed of the car.
- **Power measurement module** - For power measurement of the propulsion system an the solar panels.
- **Motor controller** - For motor controller status.

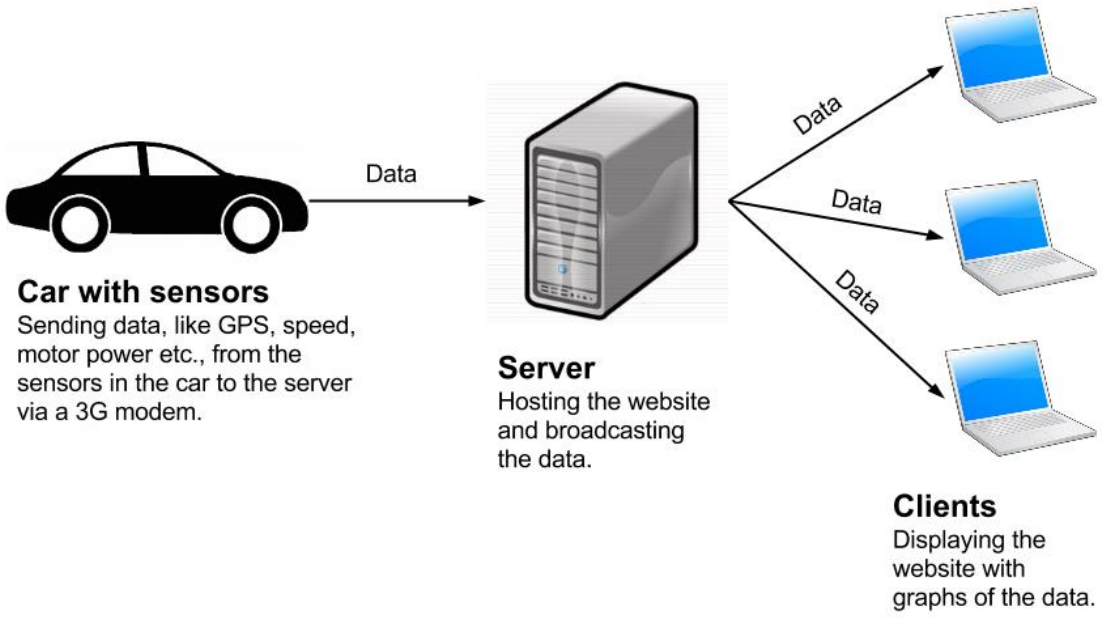


Figure 3.3: Logging system principle build up.

3.4 3G module

3.4.1 Hardware

As explained, the plan was to use 3G modems instead of GSM modems, even though a board with a GSM modem worked. This was mainly because three 3G modems were sponsored to the team. The modems were of the type UC864 from Telit [27]. The board for this modem, made in the specialization project, was unfortunately not working. A task for this project was therefore to debug the board and produce a board that worked. In case of the faults not being found, a backup solution was to use the GSM board.

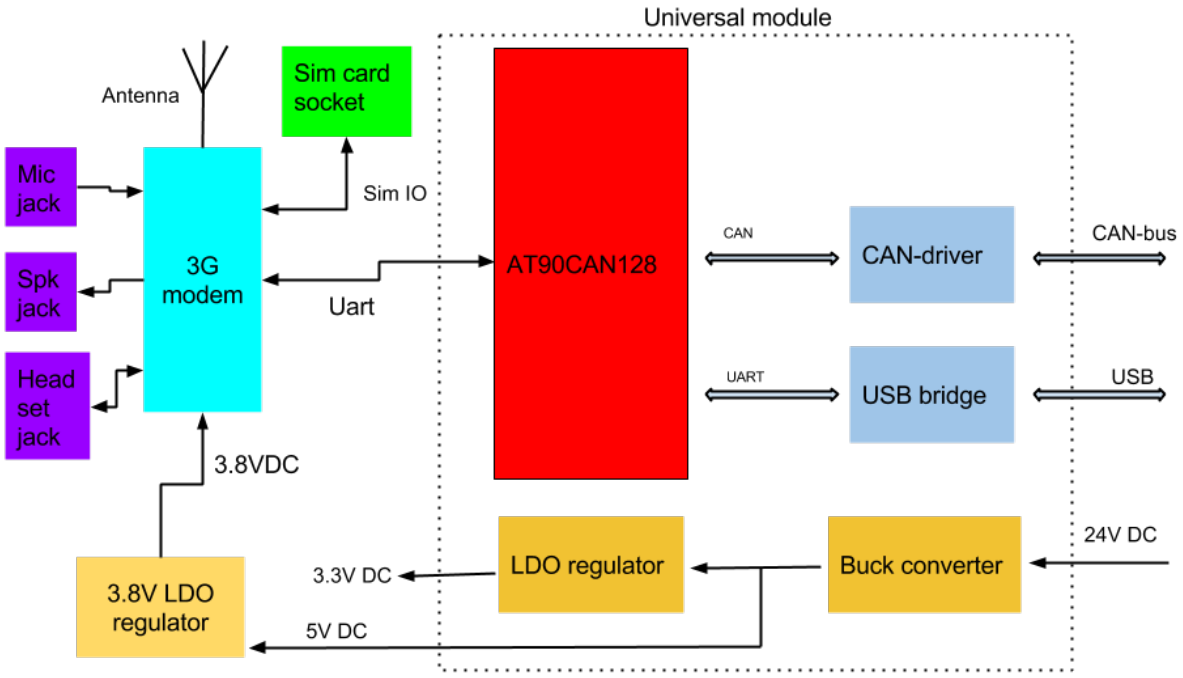


Figure 3.4: 3G module block diagram.

Debugging the 3G board

Figure 3.4 shows a block diagram of the 3G module. The microcontroller communicated with the modem over UART with AT commands [25]. When sending commands, the modem failed to respond. The default baud rate of the communication was 115200 baud. By inspecting the datasheet of the microcontroller [1], the error at this baud rate was at 8.5%, with a clock running at 8 MHz or 1MHz. This exceeded the recommended maximum error, that was ±2%. The baud rate of the modem could only be changed by certain commands sent over the UART.

A new board was made to communicate with the modem directly from a computer, rather than changing the oscillator on the universal module, which could have been another option. This was possible by using an USB-to-serial chip [6]. With this board, all the functionality of the modem could be tested in an easy way. The board was named 3G evaluation board.

The 3G evaluation board was milled with a CNC milling machine at the university. The PCB was designed with components that was already available, and was assembled in the workshop at the school. The board included a connection for a separate power supply. This was done since the modem could draw more power than a single USB port of a computer could deliver.

The UART voltage level on the modem was at 2.8V, so a voltage regulator was added to supply this voltage. The finished board can be seen in picture 3.5.

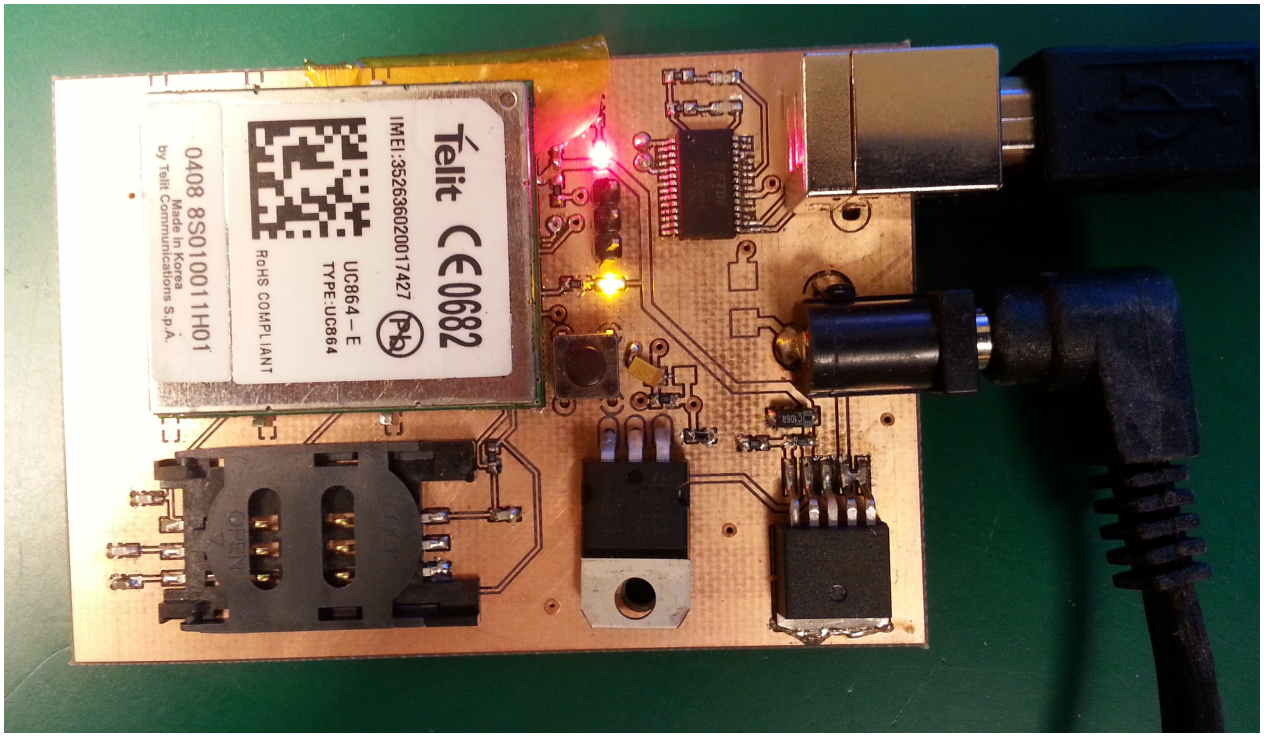


Figure 3.5: 3G evaluation board.

A terminal program was used to communicate with the modem, and the baud rate was set to 115200 baud. The highest baud rate that the microcontroller on the universal board could communicate with an error less than the recommended ±2%, was 38400. With this baud rate the error was 0.2%. By sending the command [25]

AT+IPR=38400

the baud rate was changed. To store this in the non-volatile memory of the modem, making this the default baud rate at start up, the command

AT&W0

was sent.

The modem was then placed on the 3G board made last semester. The communication with the modem was now successful. However, the modem was not able to connect to the mobile network. This was confirmed with the commands (including the response)

```
AT+CREG?
+CREG: 0,0
AT+CGREG?
+CGREG: 0,0
```

The second zero in this response meant that the modem was not registered on the GSM/GPRS network and was not searching a new operator to connect to. The reason was found to be that the modem would not recognize the SIM card, with the command (including the response)

```
AT#QSS?
0,0
```

Where the second zero in the response meant that the SIM card was not inserted. A check of all the connections and voltages were made. It was discovered that there was no voltage at any of the SIM card pins, while at the evaluation board three of the pins were at 1.84 V. No clear reason for the fault was found. However, the UART level converter used was a voltage divider with a power monitor pin used as pull-up, as seen in figure 3.6. Since the power monitor pin was not designed for this, it may have prevented the modem from working.

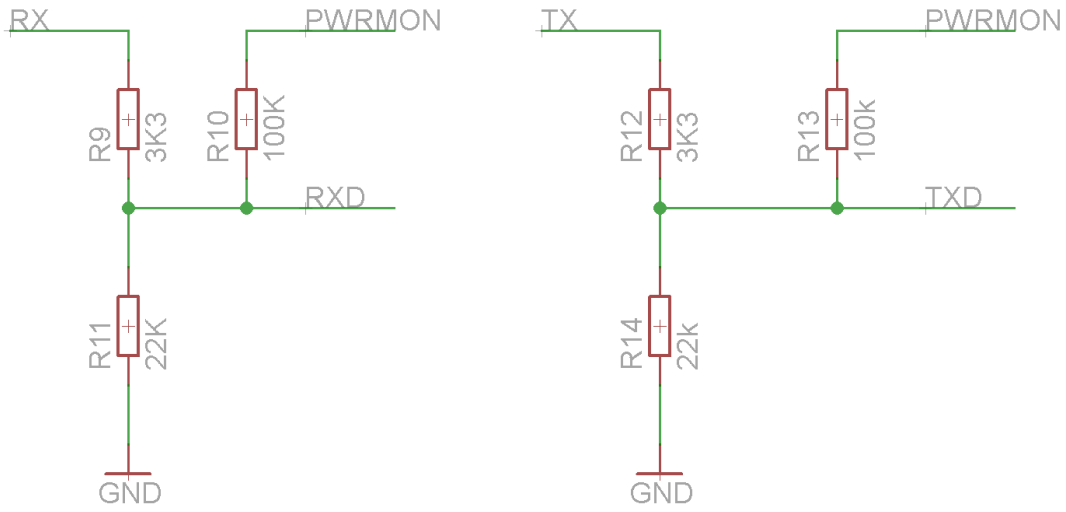


Figure 3.6: Voltage divider as level converter.

A new board was designed with a new level converter seen in figure 3.7. The level converter

was using a 2.8V refence instead of the power monitor pin.

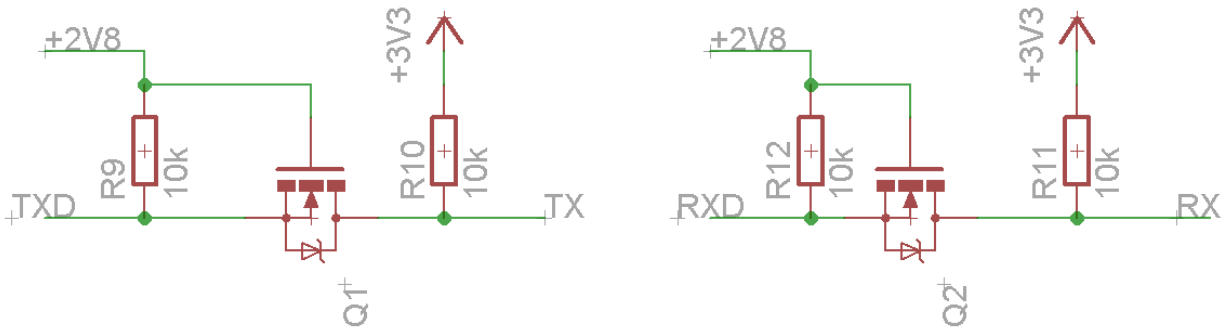


Figure 3.7: New level converter.

The board was milled out to quickly test if it worked, as this was uncertain. The board was tested and worked as expected. The new board can be seen in figure 3.8. A wire had to be added because of a poorly connected via. An antistatic bag was placed under the modem to avoid short circuits.

When it was confirmed that the new board was working, manufactured boards were ordered. The finished board can be seen in picture 3.9.

3.4.2 Software

The software was also based on the work done in specialization project. Figure 3.10 shows the state machine for the connection to the server including fault handling. The dial state was not used for reasons described in the test section.

Several CAN receiver interrupt routines was set up for the different data, and would update a shared variable containing all the data that was to be sent to the server. Once every second a packet was constructed and sent to the server with this data.

3.4.3 Test

The modem worked without major difficulties when testing in Norway. The speaker and microphone jacks did not work, even though the hardware was designed according to the datasheet

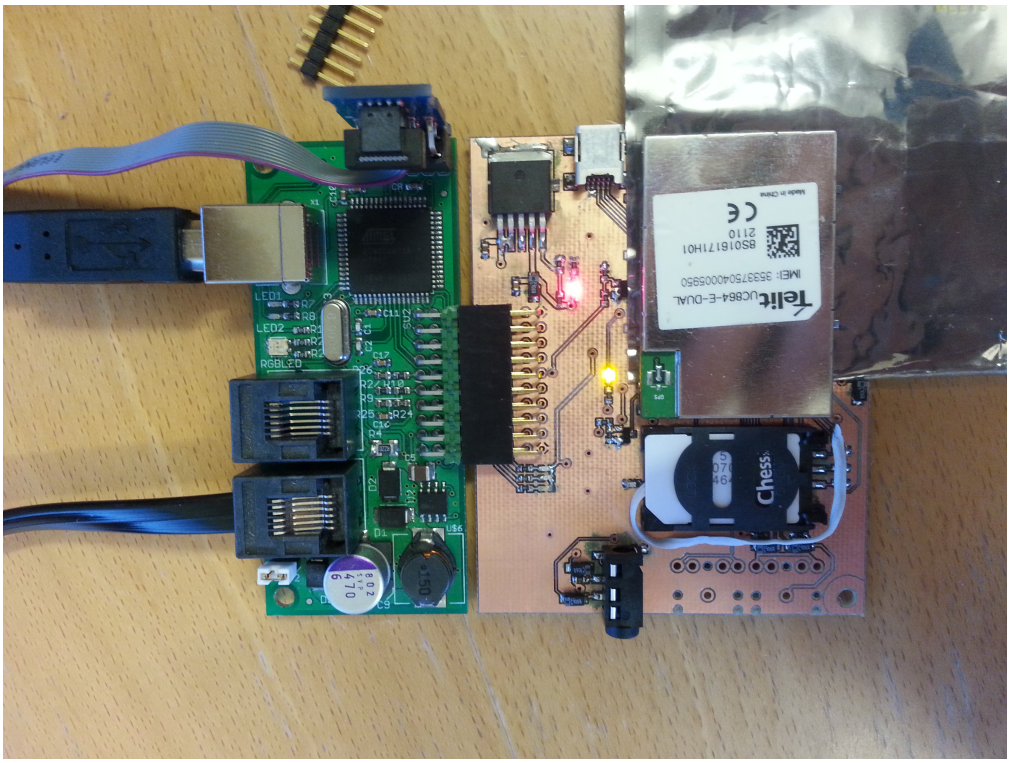


Figure 3.8: The new 3G board milled out, together with a universal board.

[26]. This was not a part of the specifications and not that important, therefore no further work was done on this part.

In the Netherlands there were some difficulties with getting the SIM cards to work with internet. The SIM cards were bought there to avoid high cost of using norwegian SIM cards. What worked was to put the SIM card in a normal phone and check the internet along with the Access Point Name (APN) that was used. After doing this and changing the APN in the code, the modems where able to connect to the internet.

One of the modules stopped working when in the Netherlands. The reason was unknown and the module could not be fixed. Because installing the modem was a bit difficult, only the Prototype car was used during the testing and the race in the Netherlands.

Figure 3.9: The final 3G module used in the car.

3.5 Power measurement module

3.5.1 Requirements

One of the requirements for the data acquisition system was to be able to measure the effect drawn by the motor and the effect delivered from the solar cells. The electrical system needed two emergency stop buttons that would isolate the battery from the propulsion system when pushed. One button was accessible from the outside of the car, and one was accessible for the driver. To avoid having the wires for the propulsion system going in front to the driver and back again through two stop buttons, a relay was used. There was also a request to be able to shut down the propulsion system from the CAN bus, to save power when the motor was not in use. It was natural to include all these parts on one board.

In addition the inverter drew a lot of current at the moment it connected to the battery, due to capacitors across the input. This would either have made the Battery Management System (BMS) isolate the battery from the rest of the system, or have blown the fuse connected to the battery. In the case of the UrbanConcept car this could have blown the fuse on the step up

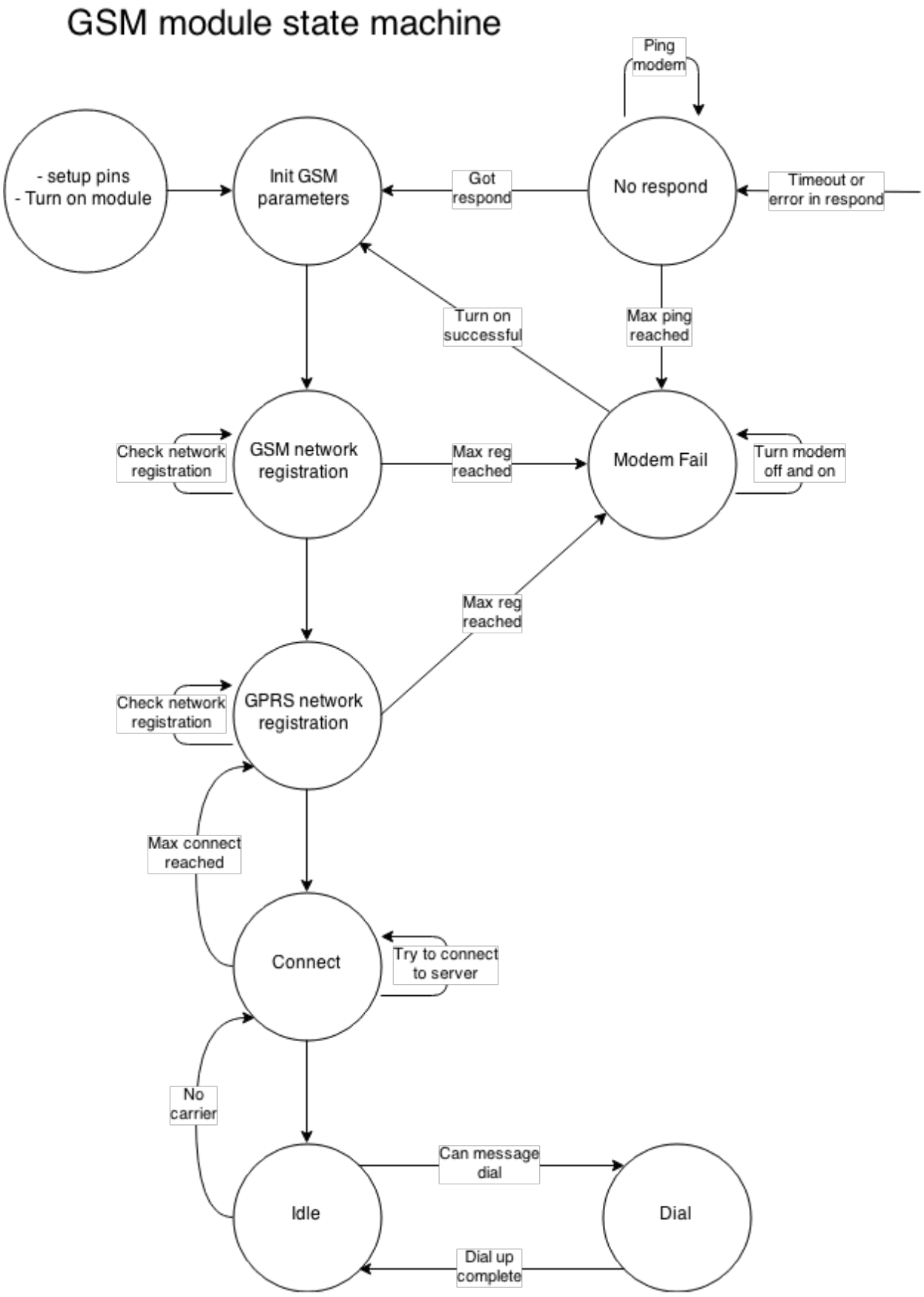


Figure 3.10: 3G module state machine.

converters. Therefore, a way of limiting the current had to be implemented either automatic or manual.

Below is a list of all the requirements that could have been joined on one board.

- **Measure propulsion system effect.** This meant measuring the current of the propulsion system and the battery voltage. The accuracy of the measurement had to be sufficiently high.
- **Measure solar panels effect.** This meant measuring the current from the solar panels and the battery voltage. The accuracy of the measurement had to be sufficiently high.
- **Emergency stop system.** A relay had to shut off the propulsion system from the battery when one of the emergency stop buttons was pushed.
- **Remote stop system.** It had to be possible to shut off the system from the CAN bus.
- **Limit current.** The current to the inverter had to be limited the moment it turned on, to avoid overloading the battery or fuel cell. Either automatically or manually.

3.5.2 Hardware

Figure 3.11 shows the power measurement board block diagram. The different hardware elements will be described here.

- **Hall effect sensor.** Measuring the current from the solar panels was done with a hall effect current sensor. The sensor had an optimized range of ±5A. The output was a voltage with sensitivity of 185 mV/A, with a zero current output voltage of VCC/2. The VCC used was 5V to give a zero current output of 2.5V. An amplifier was used before the ADC to get more accuracy over the desired range. The configuration for the amplifier and the calculations for the components is located in the appendix. Since current from the solar panels only should flow in one direction (into the battery), only the positive current range was used.
- **Current transducer.** For measuring the current to the motor system, a current transducer was used. The measuring range was ±80A and sensitivity was 25mV/A with a zero current output voltage of 2.5V. An amplifier was used to get more accuracy over the desired

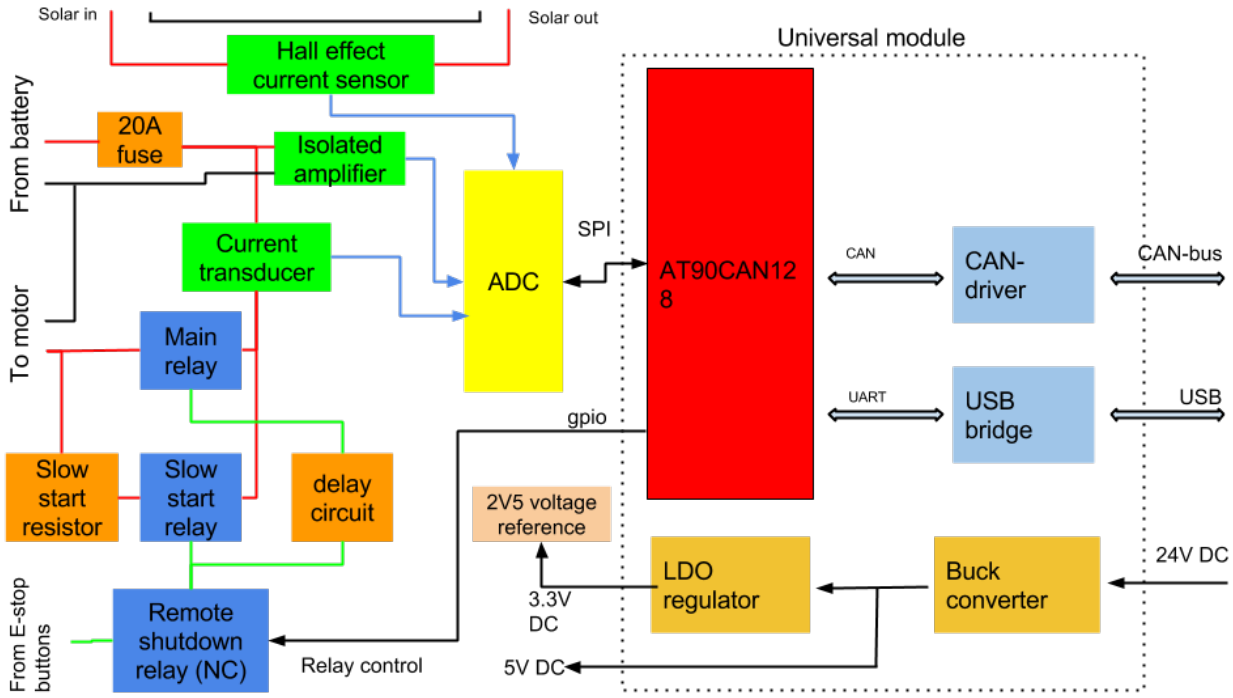


Figure 3.11: Power measurement module block diagram.

range. The desired range was set to ±20A, since the fuse was rated for 20A and the current also could flow into the battery. Configuration for the amplifier and calculations for the components is located in the appendix.

- **Isolated amplifier.** The voltage from the battery could have been measured with a voltage divider, but this would mean that the propulsion system and the accessory system would have common ground. There are a number of reasons that this should be avoided, where noise immunity and ground loops may be the most significant. A differential isolated amplifier was used to measure the voltage without a common ground. The measuring range was configured to 0-60V, as 60V was the maximum allowed voltage at any point of the vehicle specified by the SEM rules.

- **ADC.** An external ADC was used instead of the ADC input of the microcontroller. There were two main reasons for using an external ADC. The first was that the accuracy was higher, the external ADC was 12 bit while the micro controller ADC was 10 bit, giving 4 times more accuracy. The second was the possibility to have an external voltage reference.

The microcontroller can do this, but when used from the universal module this pin was not available. The micro controller has however an internal 2.56V reference.

- **2v5 voltage reference.** A 2.5V reference was necessary for the external ADC and the amplifiers to measure the desired range.
- **20A fuse.** A fuse was connected on the positive input from the battery to disconnect the battery if an over-current condition occurred. This was not necessary, as a fuse was installed directly on the battery instead. This will be described in more depth in chapter 5.
- **Main relay.** The main relay connected the battery directly to the motor system. A led was added to show if the main relay was on.
- **Delay circut.** The delay circuit was a part of the slow starter. This circuit delayed the opening of the main relay for about 1 second while the current flowed through the slow start resistor.
- **Slow start relay.** The slow start relay connected the slow start resistor between the battery and motor system, allowing the capacitors on the inverter to charge up before the main relay turned on. A led was added to show that the slow start relay was on.
- **Slow start resistor.** The slow start resistor limited the current to the motor system for about 1 second after it was turned on. The slow start resistor was not included on the board as it was too big, due to high current capability. A terminal was used to connect the external slow start resistor.
- **Remote shutdown relay (NC).** The remote shutdown relay was for turning off the motor system from the microcontroller. This allowed the system to be turned off from the CAN bus to save power when the motor was not used. The relay was normally connected (NC) such that the rest of the relay circuit could still work without the microcontroller present. All other relays on the board were normally open (NO).

The final soldered PCB can be seen in figure 3.12.

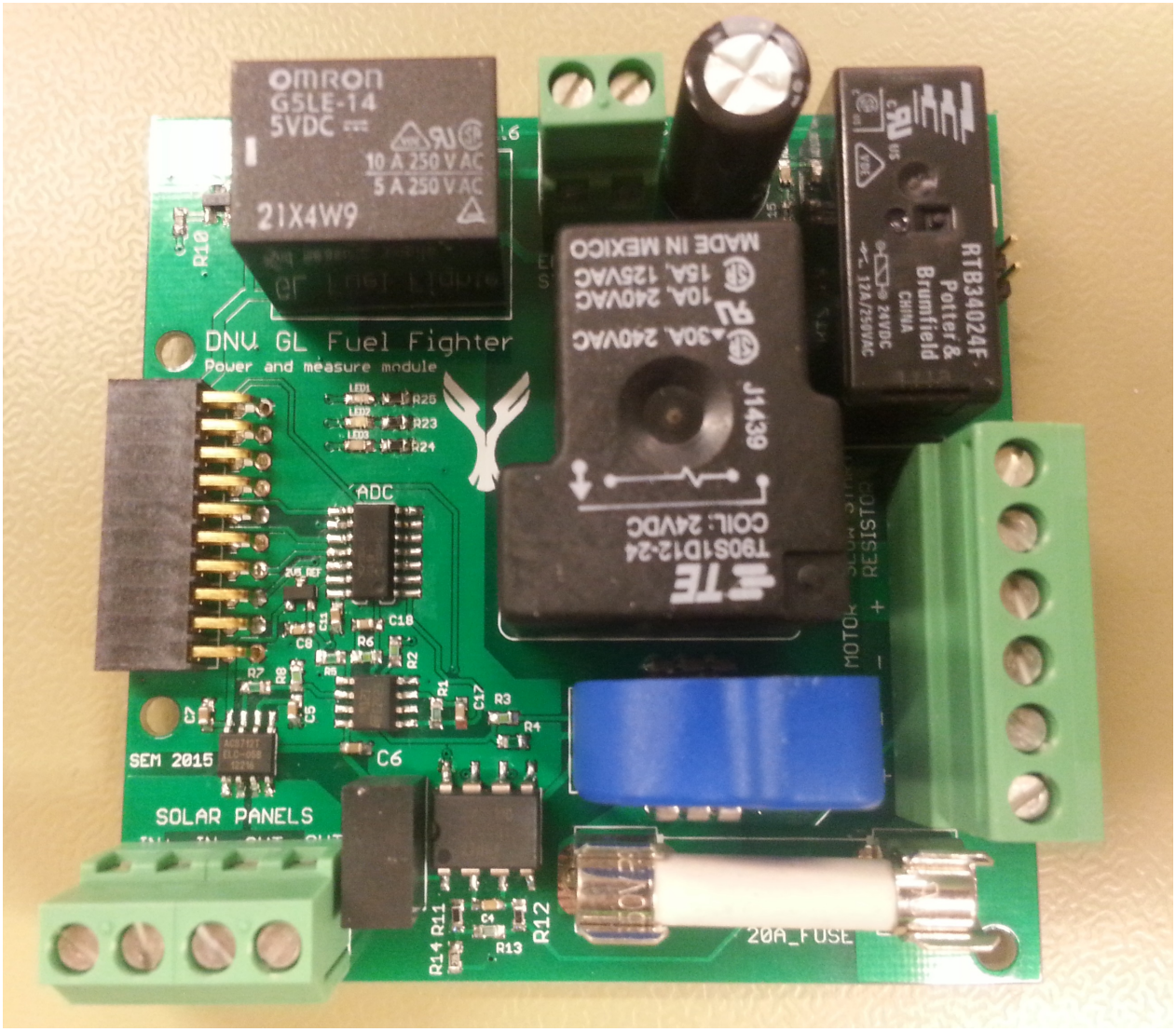


Figure 3.12: Power measurement module PCB top view.

3.5.3 Software

Since the 3G module sent packets to the server every second, a CAN packet from the power measurement module was only sent once per second. This packet contained the average measurements over the last second. The code also listened to CAN packet from the steering wheel to decide if the motor system should be turned off.

Averaging

To get the average measurements in the right format required some calculation. This would take time on the microcontroller, especially division since this is not in the instruction set [1]. If this was not done asynchronously with the ADC measurements, the microcontroller would miss some measurements and the average would not be the "true" average. To avoid this, the retrieving of ADC measurements had to happen asynchronously in an interrupt routine, while the averaging was done in the main loop every second. Since the interrupt routine had priority over the main loop, the ADC measurements could not be interrupted. The drawback was that the time from the measurements were done until the packet was sent would take some more time. With a sending frequency of 1 Hz however, this was almost not noticeable.

Interrupt based SPI

The ADC datasheet [18] explains how to interface the ADC with a micro controller using eight data bits SPI. Three bytes was sent total per ADC measurement. The two first bytes from the micro controller determined if the measurement should be single ended or differential, and which channel(s) to use. The two last returned bytes from the ADC was the measurement, MSB first and right adjusted.

Sending a byte over SPI on the AT90CAN is mostly done by the hardware. When a byte is written to the SPI data register, it will be clocked out on the MOSI (Master Out Slave In) pin. A flag is set when the transfer is done and the returned byte can be read from the same register. One can either wait for the flag to be set by pulling or by interrupt. The interrupt solution was chosen for reasons described in the previous section. In the interrupt the register was read and a new transfer was initiated. The program would cycle through the three different ADC measurements

in order.

Format

The main loop would calculate the average once every second and send the results on the CAN bus. The format of the data sent was in volt or amperes. The numbers were multiplied by 10 or 100 before the averaging to allow for one or two decimal places without using floating point numbers. All variables from the ADC measurements that was used in the averaging had to be copied to temporary variables while the global interrupt flag was turned off. This was to avoid the interrupt to change, or even corrupt, the variables while calculating the average. The global interrupt was turned on again as soon as the variables were copied.

Remote shutdown

The remote shudown was done by examining the CAN packet from the steering wheel. It was decided that the driver should pull the joystick down to turn off the propulsion system. The joystick value was a 10 bit value, but only the eight most significant bits were used, which would be the same as dividing the number by 4. Half the value (127) meant that the joystick was in zero position. The code would shut off the system if a value lower than a defined lower bound was detected, and turn it on if a value higher than a defined upper bound was detected. Between the two limits no action was taken. This was to avoid flickering around on/off. This process is also known as hysteresis.

3.5.4 Test

Before testing the board in the car, it was tested using a power supply and a multimeter. Current measurements were tested by shorting the output of the board and adjusting the current flowing. The measurements were calibrated by measuring several values.

Hall effect sensor

The hall effect sensor measuring the solar panel current was not working as it should have. The output did not change until a certain amount of current was flowing. The problem was that zero

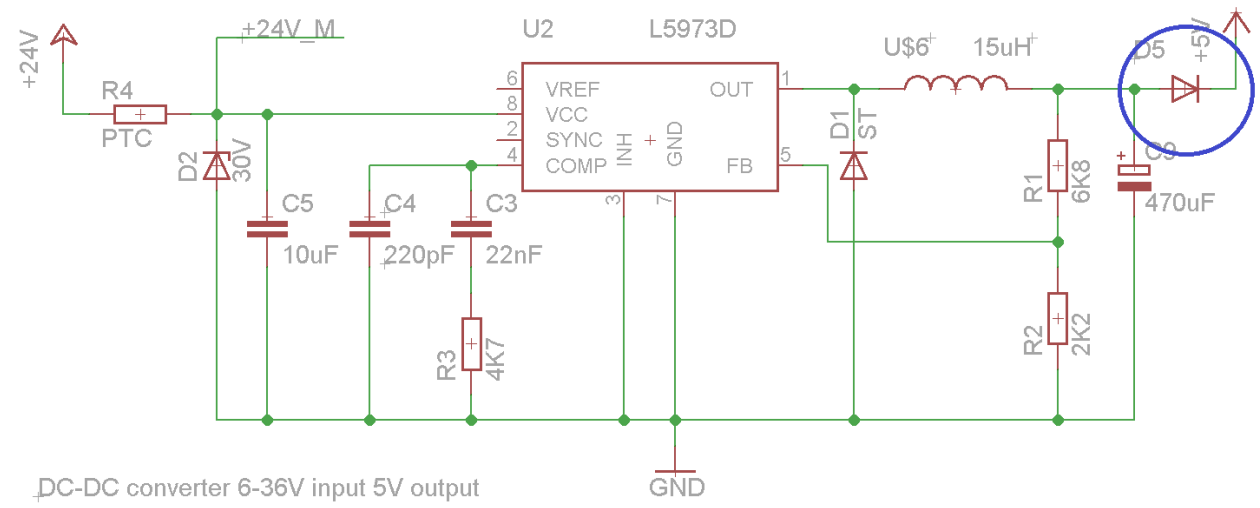


Figure 3.13: Diode (inside blue circle) at the output of the buck converter lowers voltage below 5V.

current output was 2.3V, and not 2.5V, according to the design. The reason was that the supply voltage from the universal board was 4.6V and not 5V. This was caused by a protection diode on the output of the buck converter on the universal board, as can be seen in figure 3.13.

The solution was to change the resistors on the amplifier. The resistors had to be chosen to get the desired range. The new resistor calculations can be found in the appendix. After changing the resistors the measurement was working.

Voltage measurement

Some issues were discovered when connecting the motor system and running the motor. The voltage measurement dropped from 46.2V to approximately 40V. The voltage at the input of the differential amplifier also dropped the same percentage, indicating that the problem was at the amplifier input. The design of the amplifier had been taken from the datasheet of the amplifier [13]. Different capacitors, including no capacitor at all on the input, were tried with no result. When connecting the shielding of the motor cable to the digital ground, the voltage drop was reduced to only 45V. No further work was done to remove the problem due to lack of time and low priority. Since the voltage ought not to change much, it was assumed to be constant when the motor was on. If further testing is done on the board in the future, the amplifier should be

removed to check if the problem still is present at the input.

Motor current measurement

The motor current measurement was working fine. Some of the code was changed before the competition to get more accuracy. This resulted in overflowing variables and the logged data was not of any use. An attempt to fix the overflow resulted in round off errors. These mistakes would have been easy to fix, but because of a lot of stress during the competition together with lack of testing, this was not accomplished. The data from the logging system of the motor current was therefore not useful during the competition.

3.6 GPS, IMU and SD module

The GPS, IMU and SD module was for getting GPS position, time, orientation of the car and storing data locally on an SD card.

3.6.1 Hardware

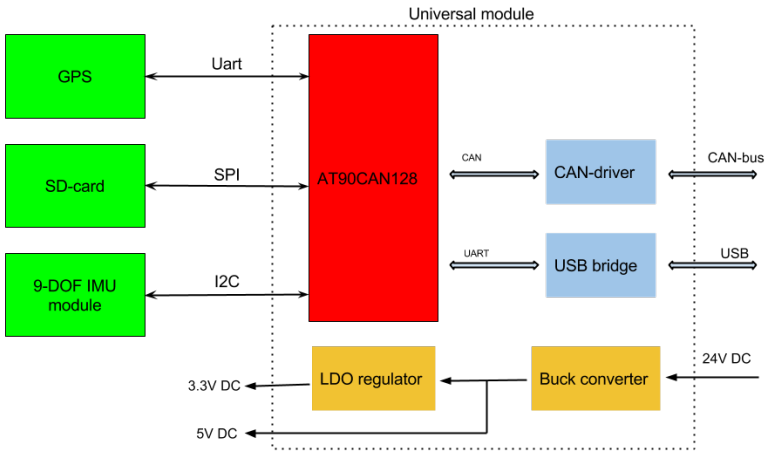


Figure 3.14: GPS, IMU and SD board block diagram.

The GPS, IMU and SD module was almost the same as one designed two years ago. The only difference was another type of GPS. Figure 3.14 shows a block diagram of the board. The IMU was a 9 degree of freedom module, meaning it had a 3-axis gyroscope, a 3-axis accelerometer

and a 3-axis magnetometer. Picture 3.15 shows the board fully mounted. Silicon paste was used to keep the GPS in place.

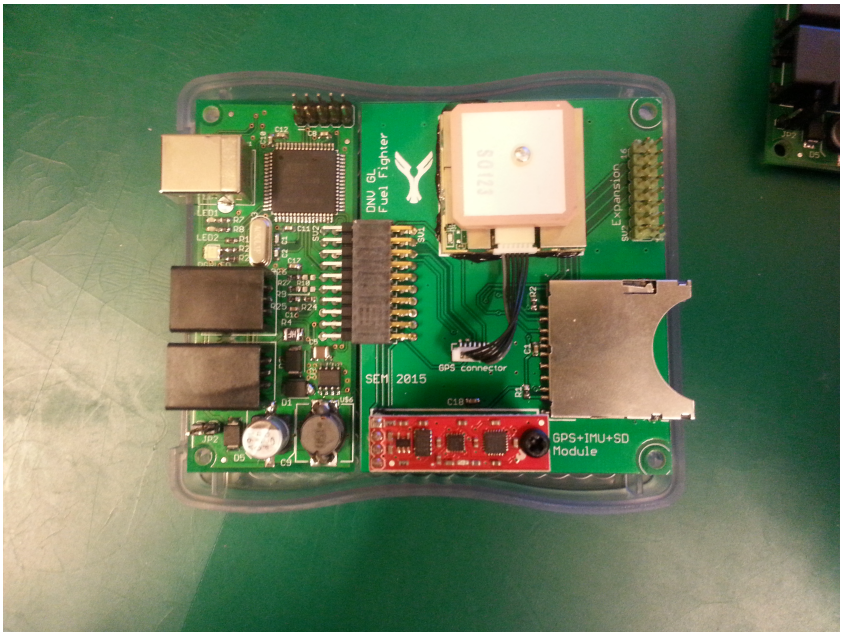


Figure 3.15: GPS, IMU and SD circuit board connected to universal module.

3.6.2 Software

Only the GPS was used for the project. When powered, the GPS sent a message every second over the UART with data divided into lines. Each line contained data organized in comma delimited format. The first line contained the information that was of interest. The datasheet of the GPS explains the meaning of the data in the message [3]. An example of a line is

```
$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M,-34.2,M,,0000*18
```

In this example line, $GPGGA is the ID and 161229.487 is 16:12:29.487 UTC time. A diagram of the code for retrieving the important data is shown in 3.16. The rest of the code constructed packets with the time, latitude and longitude and sent it to the 3G module. Milliseconds was not included since the packets was only sent once every second. The GPS position was sent to the server in a format that could easily be parsed into Google Maps.

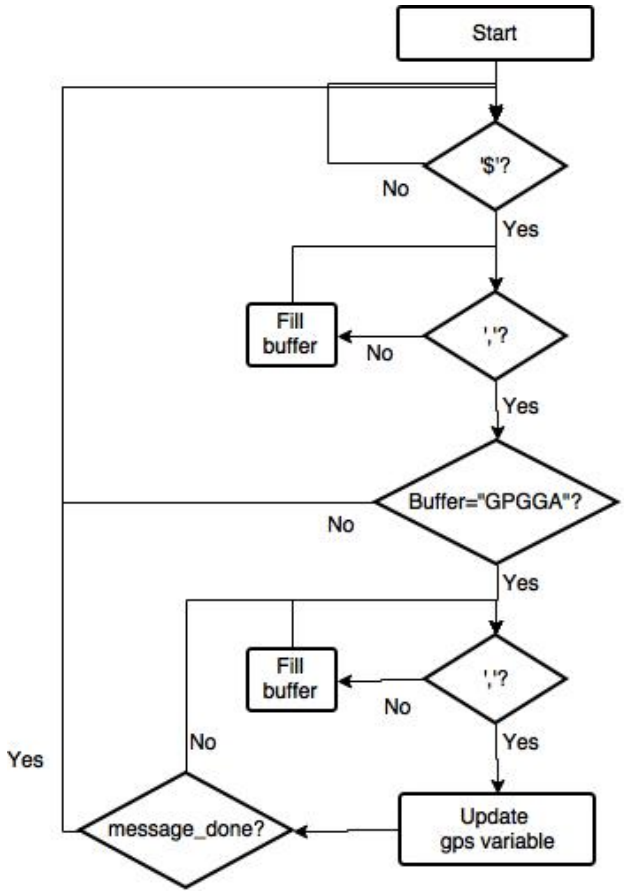


Figure 3.16: Diagram showing the retrieval of GPS data.

3.6.3 Test

The module worked without any further modifications.

3.7 Logging system

The logging system was responsible for showing the data to the user. This involved directing the data to the user and display it. The design of the system was much the same as proposed in the specialization project. An overview of the system can be seen in figure 3.3. A server was responsible for retrieving the data and sending it to the users/clients. Since the server was always up and running, the clients could connect and disconnect whenever they wanted. The server broadcasted the data to all users connected. The server was also responsible for hosting the website that displayed the data.

3.7.1 Server

The server would receive the packets from the car and broadcast them to all clients connected. The communication used sockets rather than HTTP requests, because this was better when streaming data over a longer period of time. Using sockets also meant less overhead in comparison to HTTP requests.

Because the server required to be up and running most of the time, it was best to have it on a dedicated server. Amazon Web Service was used to rent a virtual machine on one of their servers in Frankfurt. Having a virtual machine meant that installing and running programs would be much the same way as when testing on a local computer. Logging into the server was done over SSH (Secure Shell) [2] and SFTP (SSH File Transfer Protocol) [3].

3.7.2 Client GUI

Displaying the data in raw text is not very useful for the user. It was decided to display the most important data in graphs. The graphs were based on a framework called rickshaw [4], which is built on the more famous D3 (Data-Driven Documents) [5]. Buttons for starting and stopping the logging of data, as well as clearing the data already logged, were added. A terminal window at the bottom was added to display the raw packets. This was useful for debugging and for saving the data. The website with two of the graphs is seen in 3.17.

Because of other more important task in the project, not much time was put into the design of the GUI. There is a lot of potential for developing the GUI further. One of the interesting parts can be to add a map with the position of the car. This is possible since the data includes the GPS position of the vehicle. It was also no way of storing the data other than to copy it from the terminal window. Storing data in a database could be possible. This could enable the user to browse through previous races or tests.

[2] http://en.wikipedia.org/wiki/Secure_Shell
[3] http://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol
[4] http://code.shutterstock.com/rickshaw/
[5] http://d3js.org/

3.7.3 Test

The logging system was useful during testing of the car and during the race. The speed and the driver reference were especially important for optimizing the driving strategy and for testing the top speed and acceleration. Motor status data were important when testing the motor controller. Power measurements were of limited value because of trouble with the measurements and scaling, as explained earlier. Picture 3.17 shows the speed and driver reference for the first half of the second race for the Prototype car. The driver reference is from the joystick were 127 means zero position and 255 means full throttle. The speed was a bit noisy resulting in the spikes to zero.

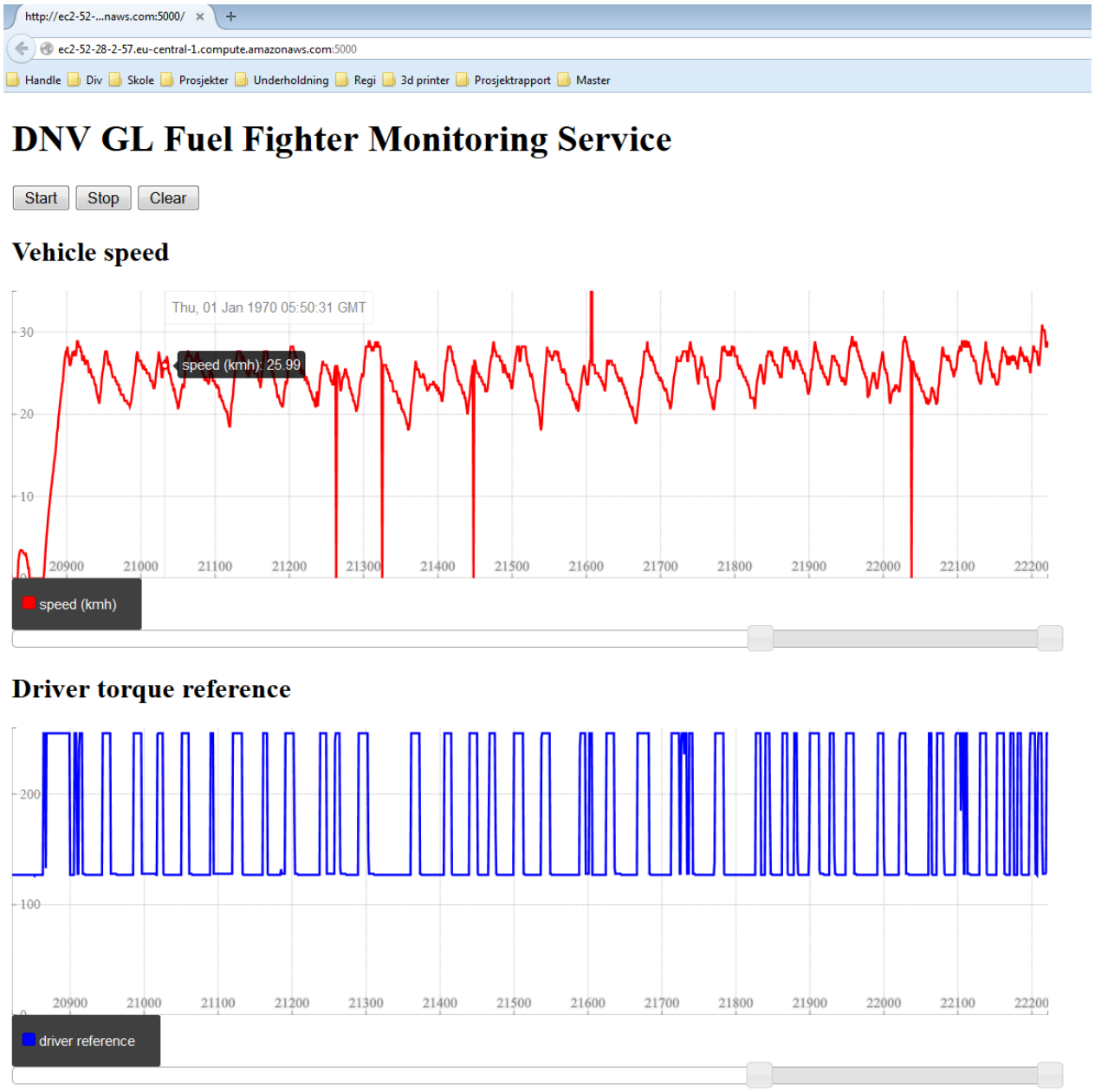


Figure 3.17: View of the logging system website (here called monitoring service), with the top two graphs.

Chapter 4

Motor controller design

The rules for SEM states that for battery electric vehicles, the motor controller have to be purpose built [5]. The motor controller for the last years team was implemented using a Field Programmable Gate Array (FPGA) from SINTEF [19]. Because the FPGA used last year was using too much power, a new motor controller was built this year. The same motor control system was used on both cars. Design of the controller in simulink and PCB design for prototype boards was done last semester [7].

This chapter is built up as follows: First, theory of motor controller operation is presented. An overview of the system is shown to see how the different parts work together. Then the developed hardware and software is described. Finally testing results and the implemented improvements is presented.

4.1 Theory

In this section some of the most important theory that is needed to understand the motor controller design is presented.

4.1.1 Motor

The motors used was of the type permanent magnet synchronous motor (PMSM), also called brushless DC motor (BLDC), and can be seen in picture 4.2. A BLDC motor has permanent magnets on the rotor and inductors on the stator, compared to brushed DC motors which has

Figure 4.1: Maxon EC60 motor.

permanent magnets on the stator. Figure 4.1 shows a basic build up of a single pole pair brushless DC motor. To get a smooth rotation of BLDC motor, the inductors have to be magnetized as the rotor turns. In general there are two ways of controlling these motors, either by block commutation or sinusoidal commutation [17]. With block commutation the current and voltage curves are block-shaped, and the only information needed is when the rotor magnetic field aligns with the stator inductors. This can be done with hall sensors or even just by measuring the back emf (electromotive force) of the motor. With sinusoidal commutation the current and voltage curves are sinusoidal. This results in a more smooth rotation, no torque ripple and more torque compared to block commutation. To apply the sinusoidal currents through the windings of the motor, a precise position of the rotor is needed. Since sinusoidal commutation should give more efficiency than using block commutation, this was the control that was chosen. Even though sinusoidal commutation requires more expensive electronics and more sophisticated control techniques.

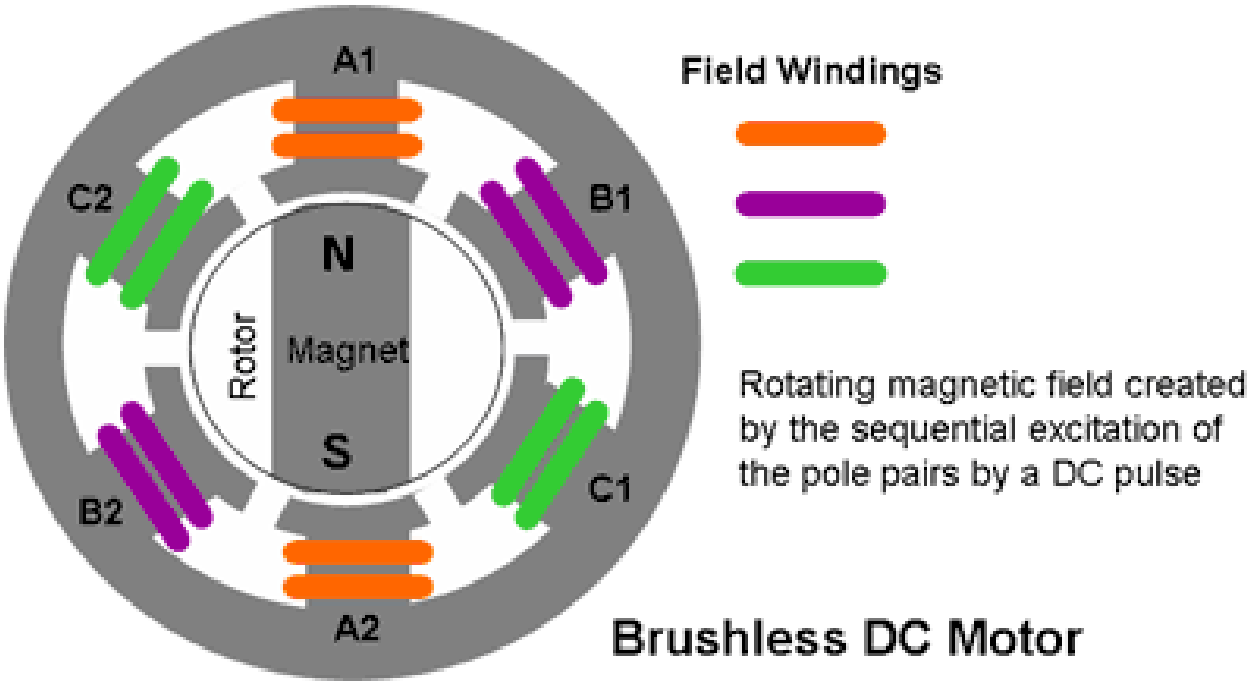


Figure 4.2: Single pole pair brushless DC motor build up.

4.1.2 Field oriented control

A standard way of controlling a BLDC with sinusoidal commutation is to transform the currents to a coordinate system that rotates with the motor. This is called field oriented control, and allows the control itself to act in a linear system. The most common way of transformation is the dq-transformation. Figure 4.3 shows a control structure from "Advanced Electric Drives" by Ned Mohan [20] based on dq-transformation.

The abc currents are transformed to a d-component and a q-component. The d-component is normally controlled to the value zero and the q-component is controlled to a reference value proportional to the torque of the motor. The current controller presented here is a PI controller, but can be of other types, as pointed out later. This block is from now referenced to as I-controller or current controller. Since the d and q components are not totally isolated, decoupling terms need to be subtracted from the controller output. The voltage output is then transformed to abc voltages. The voltages are applied to the motor phases through PWM.

Measurements of at least two of three phase currents are necessary, since the sum of all three should be equal to zero. All three currents can be measured for more robust and precise

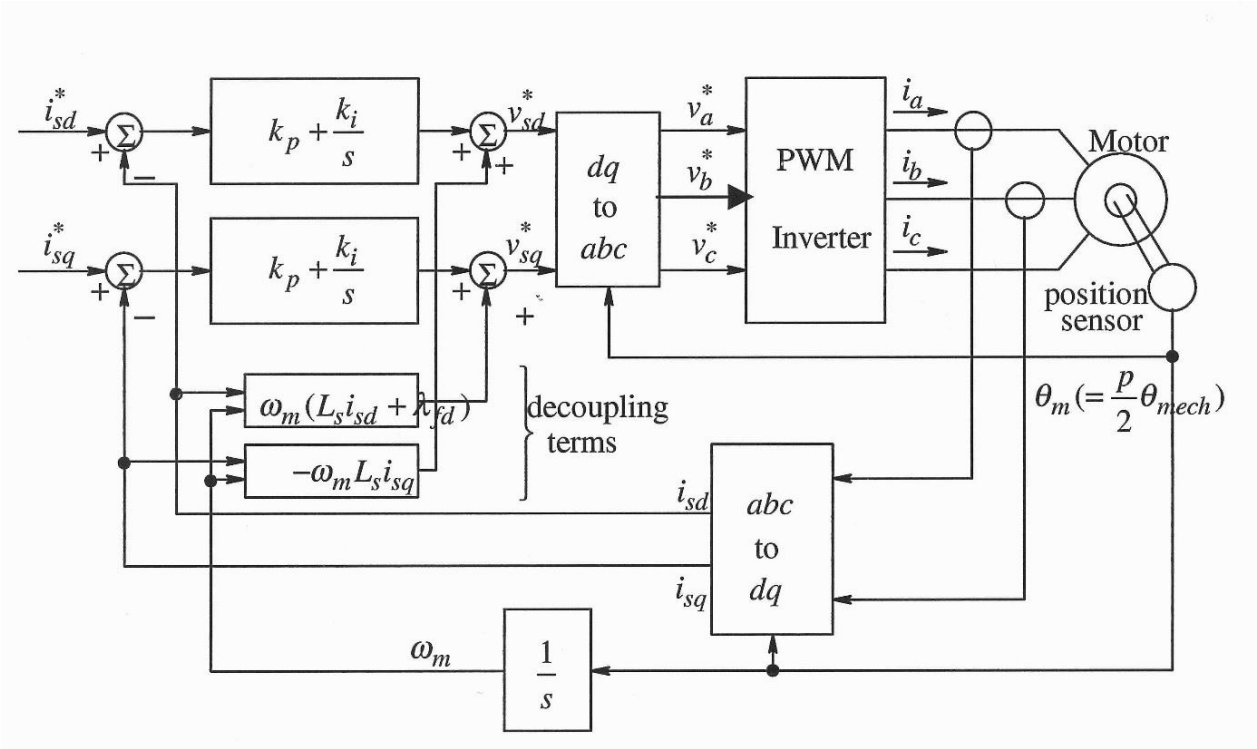


Figure 4.3: Field oriented control based on dq-transformation.

measurements.

4.1.3 Power stage

The power stage includes the power electronics for controlling the voltages to the motor phases. This is done by using three half bridges as seen in figure 4.4. The switches used are normally mosfet or IGBT since they have low switching losses. Mosfets and IGBTs are voltage controlled switches and require gate drivers to operate optimally when controlled from a low voltage device such as a micro-controller. Logic is also needed for avoiding two transistors in the same half bridge to be on at the same time, which would lead to a short circuit. Current and voltage measurements can also be considered part of the power stage.

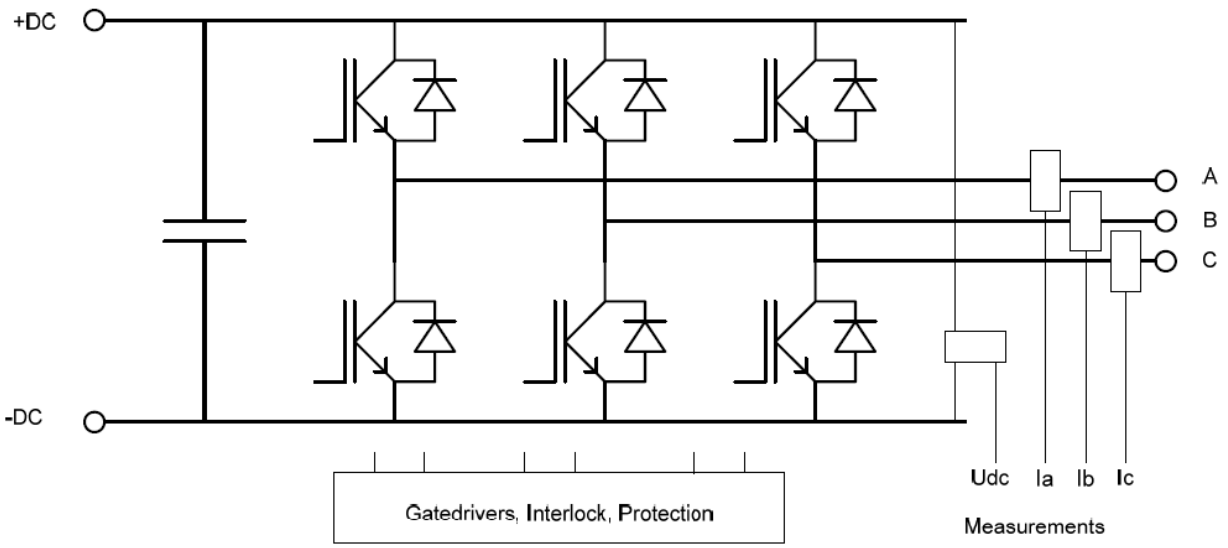


Figure 4.4: Three phase bridge.

4.2 Overview of motor controller system

Figure 4.5 shows the motor control system for one motor. For the Prototype car this was the main motor, while for the UrbanConcept car this was for each of the motors connected to the back wheels. Voltage came from the battery or fuel cell with boost converters respectively in the two cars, and was relatively constant at 46.2 V for the battery and 48V for the fuel cell plus converters configuration.

The motor was the EC60 motor from Maxon [16]. It was a brushless DC (BLDC) motor with a factory mounted resolver. The type of motor was decided and bought last semester.

The inverter contained the power electronics like the three phase half bridges, and phase currents and DC voltage measurements. The inverter was built as part of a master thesis last year [19].

The resolver to digital converter (RDC) was responsible for calculating the position and velocity of the motor from the resolver signals. The resolver was fed with an excitation signal and returned two sinusoidal signals. The excitation signal and return signals were compared to calculate the position and velocity. The position and velocity were sent to the main controller board.

The main controller board contained the microcontroller, an mbed LPC1768. This device is

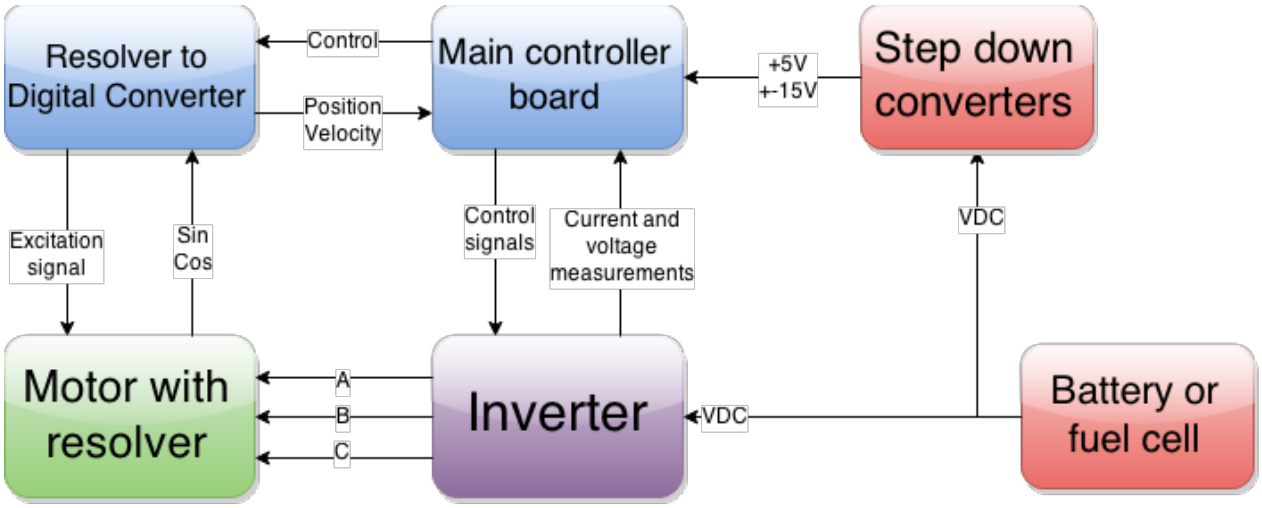


Figure 4.5: Overview of motor controller system.

explained in greater detail later. The control unit took the current, voltage, position and velocity measurements and computed the phase voltages. The phase voltages were modulated with PWM, and sent to the inverter to control the three phase half bridges. Other control signals from the inverter included on/off control of the transistors and status feedback signals.

Step down converters supplied the logic, amplifiers and sensors with the correct voltages. The step down converters was the same as used last year.

4.2.1 Microcontroller

The microcontroller used for the motor controller was the mbed LPC1768 [1], shown in picture 4.6. It is based on the NXP LPC1768, with a 32-bit ARM Cortex-M3 core running at 96MHz. Mbed is a platform for rapid prototyping, mostly against internet-connected devices. The mbed environment includes many libraries and an online code editor and compiler. This was important in the development phase, since focus could be on the code and not on the underlying functionality.

[1] http://developer.mbed.org/platforms/mbed-LPC1768/

Figure 4.6: Mbed LPC1768.

4.3 Hardware

4.3.1 Testing of prototype boards

The first designs for the *main controller board* and *resolver to digital converter board* was done by Simon Fuchs last semester. Prototype boards were milled out at the school and assembled by a company in Trondheim called Noca. Two members of the electronics team, Sacha Danjou and Milan Pesic, was responsible for this. The prototype boards can be seen in figure 4.7.

This section describes the testing of the two boards.

Resolver-to-digital board

The motor that was chosen came with a factory mounted resolver [15]. The resolver consist of one rotary winding and two windings fixed 90 degrees to each other, as seen in fig 4.8.

This type of resolver is used in the following way: A sinusoidal excitation signal is applied on the rotary winding. The resulting signal on the stationary windings S1 and S2 are then compared to the excitation signal. The position and velocity of the motor are calculated by trying to track the resulting signals.

To ease the job for the microcontroller, a specialized resolver-to-digital converter (RDC) chip

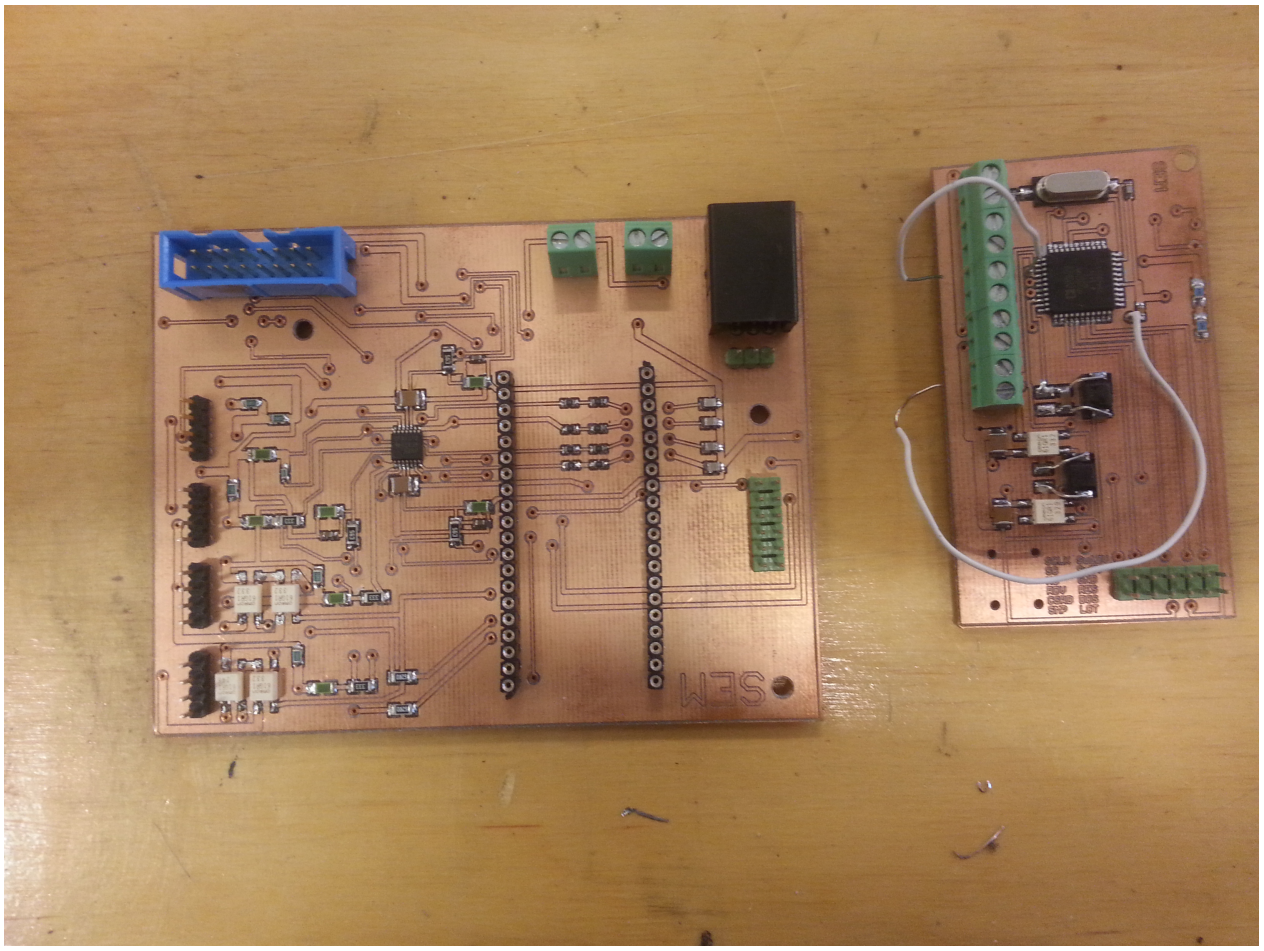


Figure 4.7: Motor controller prototype boards. Main controller board is without the microcontroller and resolver to digital board has some adjustments.

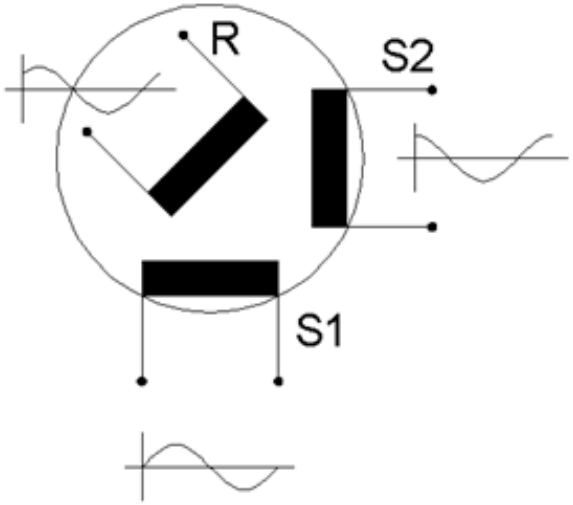


Figure 4.8: Principle of a resolver. S1 and S2 denotes the stationary windings, while R is the rotary winding.

was chosen to do this task [4]. This could send the data over a parallel or serial interface, where the latter was chosen because of less pins. The serial interface was compatible with serial peripheral interface (SPI) with clock rates of up to 25 MHz.

Resolver signals The resolver of the motor was specified for a 10V excitation signal. A buffer circuit had to amplify the signal from the RDC chip, which was 3.6 Vpp. This was done by a high current op-amp. More details about this can be found in [7].

The RDC needed a 5V power supply, while the op-amp was originally designed for ±12V. From last year there was a ready made PCB with 5V and ±15V supply. Since this was already available, it was chosen to supply the boards. The op-amp was rated for this voltage, but would dissipate more heat due to the higher voltage drop. The power supplies can be seen in picture 4.9. The testing of the RDC board was done from the main controller board.

Testing The first test of the RDC did not work. Measurements of the inputs and outputs of the resolver showed that this had the right frequency and voltage. Measuring the inputs of the RDC showed nothing. By examining the PCB design, it was found out that the board contained several unrouted tracks and a short circuit between two pins. This was difficult to spot since

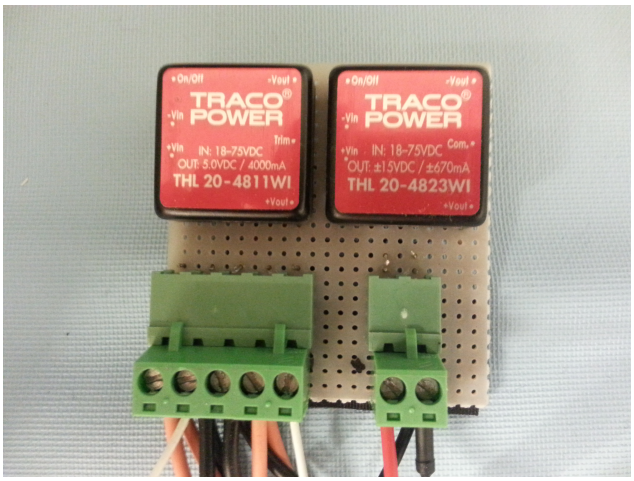


Figure 4.9: 5V and ±15V for the boards.

the unrouted layer was turned off in the design, and the design was done by someone else. In addition, the solid state relay (SSR) turning on the power to the RDC had stopped working. The solution was to remove the short by cutting away the copper and try to make a connection between the unrouted pins. The connections were made by soldering a wire in one of the vias and screwing it into the terminal block, or by adding more solder if the distance was short enough. Figure 4.10 shows the rework that was done on the board. Wire meant that a wire would be soldered in, cut meant that the track had to be cut, and short meant that some solder had to added to make a short between the pins/pads.

After fixing the mistakes on the board, the RDC was able to pick up the signals from the resolver.

Main controller board

The main control board was responsible for calculating and setting the reference voltages to the motor. Figure 4.11 shows a block-diagram of the board. Yellow blocks denotes the interface to other boards, green blocks denote converting circuits, while blue represents the power supply. The microcontroller (MCU) was the mbed NXP LPC1768, as explained earlier. The power supply was the same as the one used on the RDC board, which delivered 5V, 15V and -15V.

CAN bus interface The RDC interface was already tested working. The next part that could be tested was the CAN bus interface. The microcontroller contained a CAN driver and a ready made

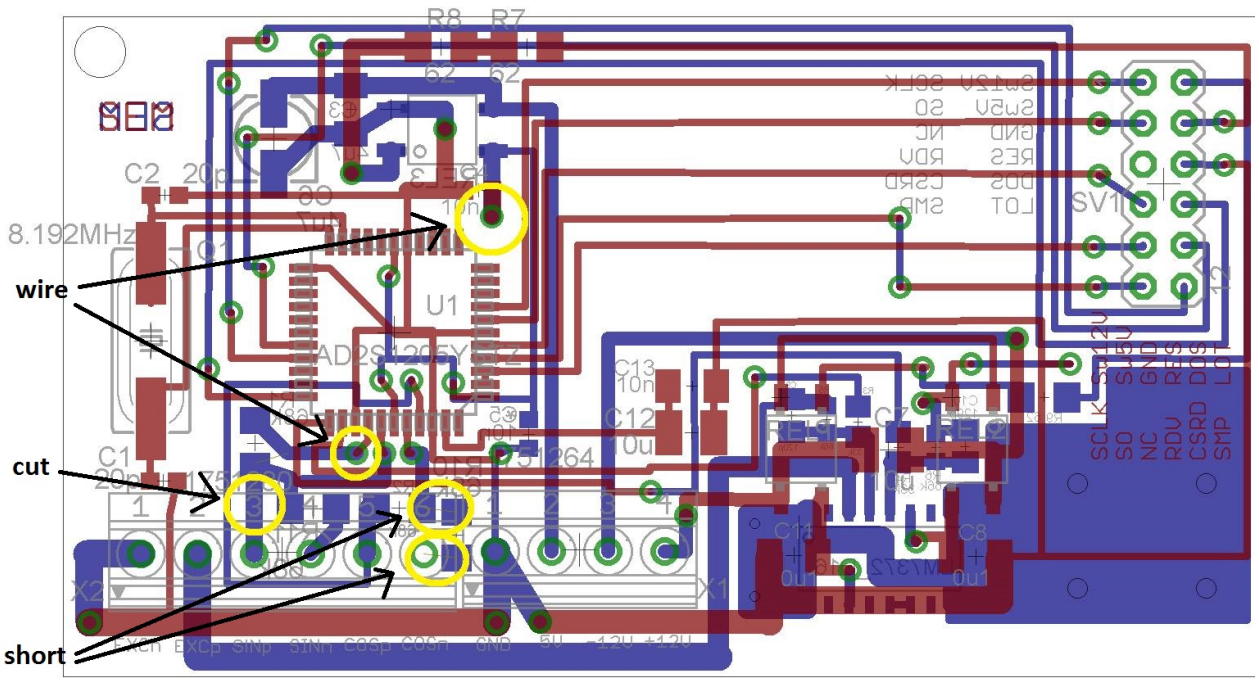


Figure 4.10: Quick solution to fix the RDC board.

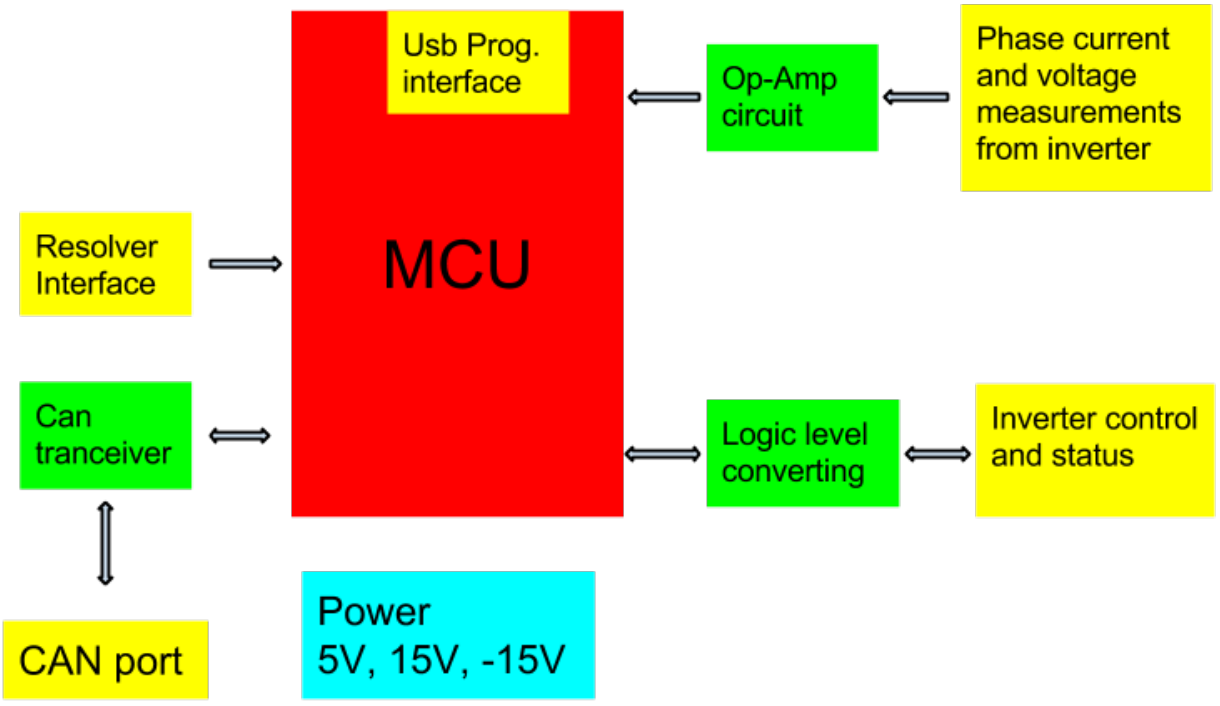


Figure 4.11: Block diagram of the main control board.

Figure 4.12: CAN adapter from PEAK.

Name	**description**
PWMA, PWMB, PWMC	Input - Pulse-width modulated (PWM) signals controlling the voltage to the different phases A, B, and C.
Transistor On	Input - Turns on the transistors that is controlled by PWMA, PWMB and PWMC.
OK	Output - High if inverter is OK, low if a fault is detected.
Status code T0 through T3	Output - Together form a status code which tells the controller what is wrong.
IA, IB IC	Current measurement of the different phases. The signal is a current source with ratio 1:1000 from the actual phase current.
UDC	Measurement of the DC input voltage. The signal is a current source with nominal output 25 mA.

Table 4.1: Inverter signals and description.

CAN library was used. To test the CAN interface a CAN adapter from PEAK was used together with software to monitor the CAN bus. The adapter can be seen in picture 4.12. The test showed that the microcontroller was able to send and retrieve CAN packages.

Inverter signals The interface to the inverter was split into two parts: Control and status signals, and the phase current and voltage measurements. The different inverter signals are described in table 4.1.

To test the control signals to the inverter, it was decided to send three sinusoidal voltages shifted 120 degree in phase. The sinusoidal signals were realized with PWM signals with dif-

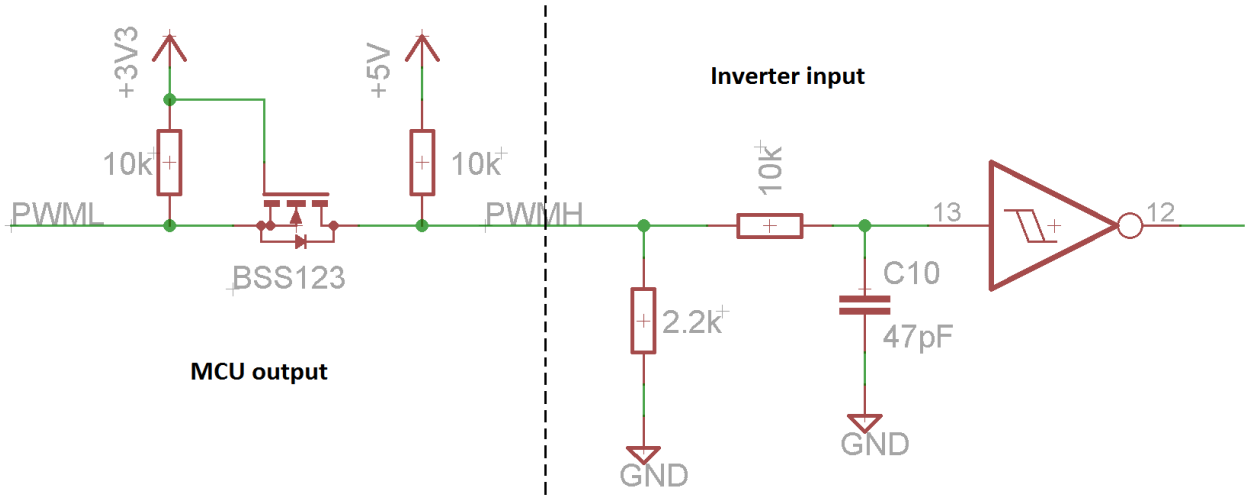


Figure 4.13: Logic level converter on microcontroller works as a voltage divider with the pull-down resistor on the inverter input.

ferent duty cycle, where full duty cycle meant the highest voltage and no duty cycle meant the lowest voltage. This was possible because the motor only "sees" the average voltage applied to it because of its inductance.

The first test did not move the motor at all. When probing the PWM output to the inverter, it was found out that the voltage level was about 2.7V and not 5V as it was supposed to be. The fault was found by consulting with Kjell Ljøkelsøy, who helped the team last year to build the inverter and also wrote the documentation for this [14]. The PWM inputs of the inverter contain pull-down resistors as a safety feature to shut off the transistors when no signal is present. The logic level converter on the main control board had a pull-up resistor to obtain 5V level when the output of the microcontroller was high (3.3V). Since the pull-down resistor was much lower than the pull-up resistor, this effectively made a voltage divider which would lower the voltage to a low enough level to never turn on the transistors on the inverter. Figure 4.13 shows the interface.

A solution was to use a non-inverting buffer which accepted TTL levels as inputs, instead of the level converter. 5V TTL levels are the same as 3.3V LVTTL levels, which is the ones used on the microcontroller. Since none of the workshops at NTNU had this buffers in stock, a temporary solution with an inverting buffer between the main control and inverter was used, as shown in figure 4.14 and picture 4.15.

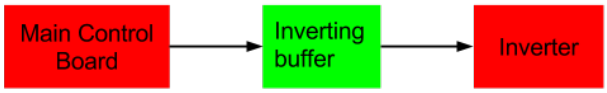


Figure 4.14: Buffer between main control board and inverter.

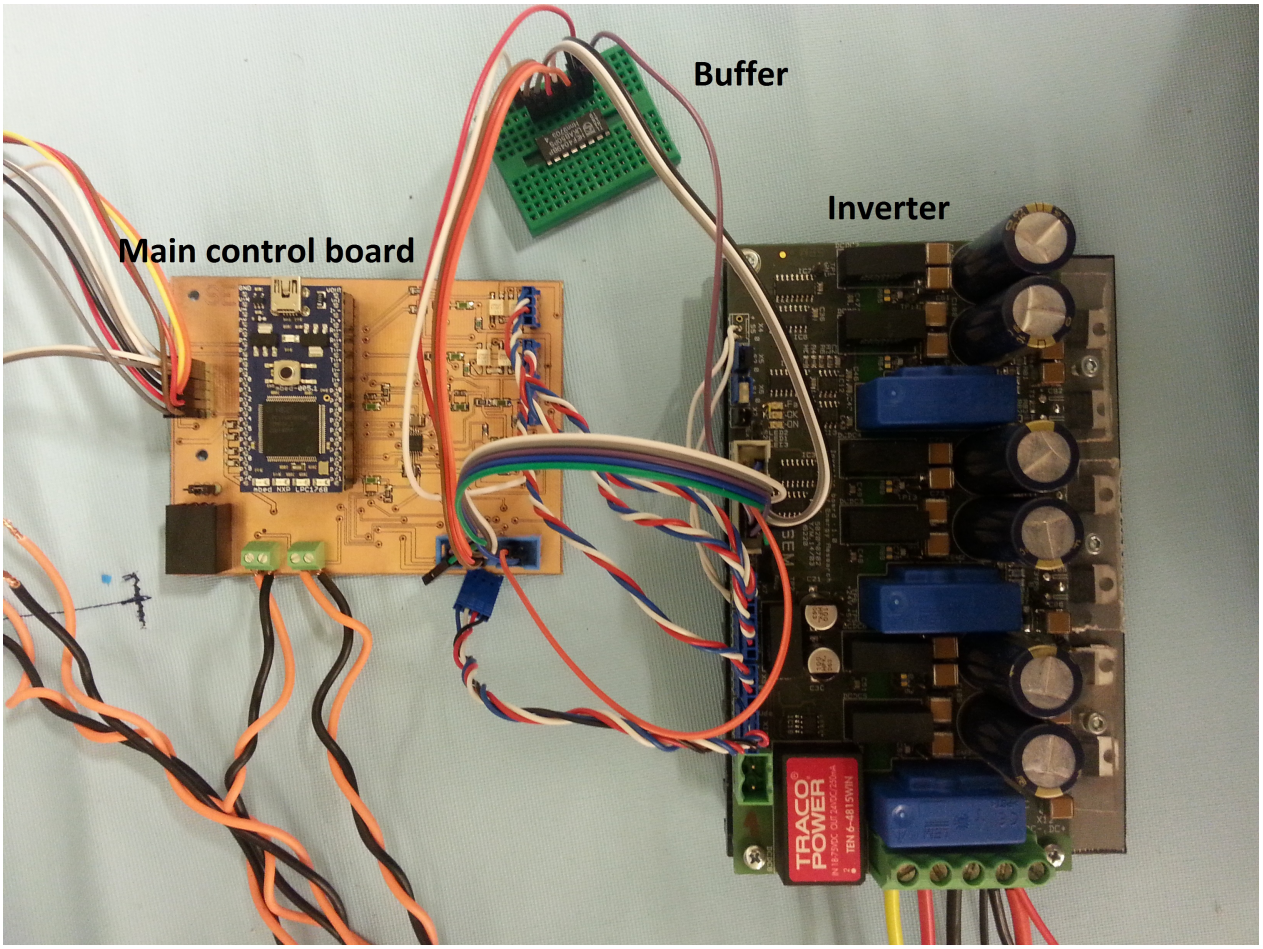


Figure 4.15: Buffer between main control board and inverter.

Appendix S: Resistor value calculation sheets

$U_{ref} = 3.3\ V$

$i_{LEM} = [-35\sqrt{2}\ mA, +35\sqrt{2}\ mA]$

$R_{Shunt} = 10\ \Omega$

$U_{IN} = 35 \cdot \sqrt{2} \cdot 10^{-3} \cdot 10 = 0.5\ V$

$\frac{R_2}{R_1} = 3.3$

$\frac{R_1+R_2}{R_1} \cdot \frac{R_u}{R_3+R_u} = \frac{1}{2} => \frac{R_3}{R_4} = \frac{56}{10} = 5.6$

$R_2 = 33\ k\Omega$
$R_1 = 10\ k\Omega$
$R_3 = 56\ k\Omega$
$R_4 = 10\ k\Omega$

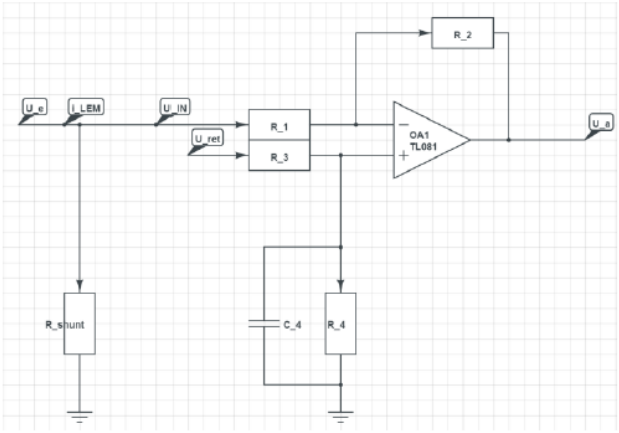


Figure 4.16: Calculation of resistors in amplifying circuit for current measurements (from [7]).

Since the buffer was inverting the signals, the output was shifted 180 degree in phase which would in practice have no effect. After applying the inverting buffer the main control board was able to spin the motor.

Voltage and current measurements The voltage and current measurement from the inverter was represented as a current source. Since the microcontroller only can measure a voltage with the Analog-to-Digital converter (ADC), the currents had to be converted to a voltage. This was done by operational amplifiers and is described in [7]. When inspecting the calculation for the different values, a mistake was found. The resistor R_3 in figure 4.16 should be $76k\Omega$ instead of $56k\Omega$. In practice this meant that the signal would be centered around $0.651 * 3.3V \approx 2.15V$ instead of $0.5 * 3.3V = 1.65V$. This would therefore not do anything for the testing since the measurements were scaled for much higher currents than the motor could handle.

Before testing with the motor running, the input of the ADC was measured. The current measurements were about 2.15V as they were supposed to be. The voltage measurement was

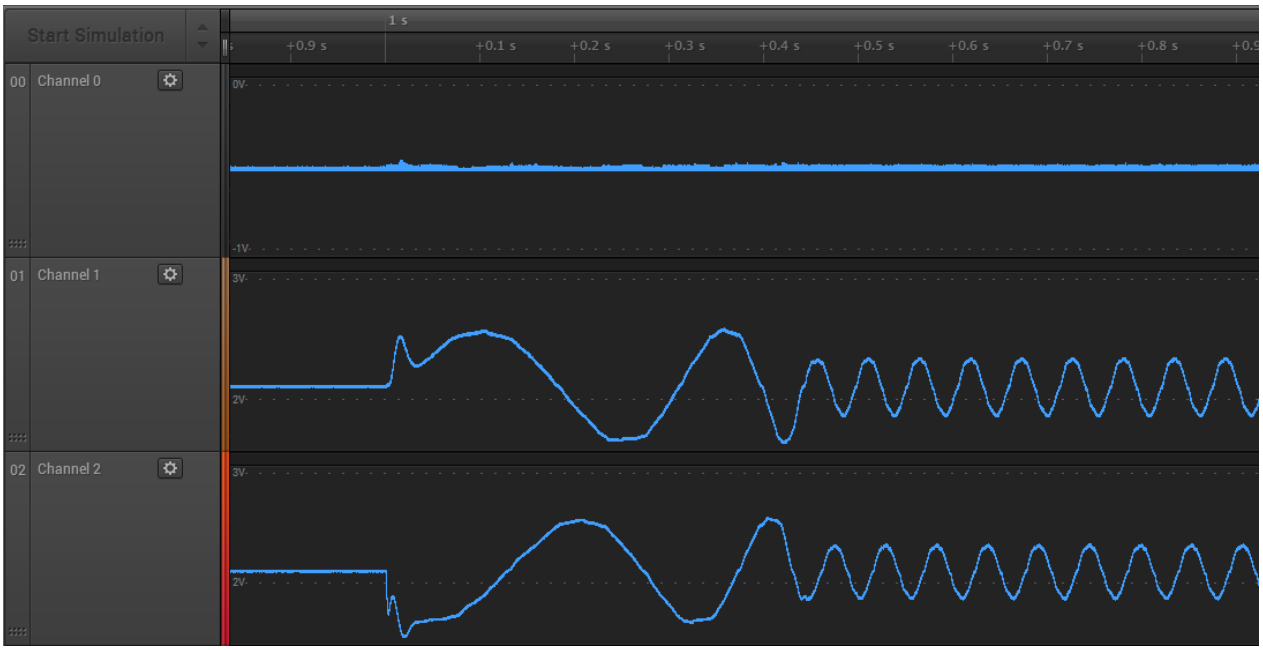


Figure 4.17: Current measurement test with sine waves input.

found to be $\frac{1}{34}$ of the actual voltage. When testing the current and voltage measurements after the motor was running, the voltage measurement and one of the current measurement was showing a constant negative voltage. The measurements of the three currents is shown in 4.17. We can see that the first current is a constant negative value of -0.5V. Measuring the value without the micro controller showed a much lower voltage, indicating that the amplifier was in saturation.

Exchanging the current measurement outputs of the inverter did not change the result, which meant that the fault was on the main controller board itself. The board did not include a ground plane which is normal when designing PCB. When milling out PCB, the tool only cuts the contours of the tracks, leaving one or more "planes" not connected to anything, even though this is not specified in the design. The board contained a more or less continuous plane. This plane was measured to 2.28V, when it shouldn't have any potential to ground. This meant that one or more components were shorted to the "plane". The conclusion was that either one of the components was damaged or that some short was corrupting the circuit. The fault was not found or removed, so no further testing could be done on the board. The assumption was that there was no fault in the design, since the measurements seemed to work at some point, showing a

constant zero level voltage.

4.3.2 New PCB designs

When all functions of the boards were tested and the faults found, it was time to design new boards to be professionally made. Here, a summary of the new elements in the design is presented. Because the new designs had many new elements and it was desirable to make the boards smaller, most of the boards were rerouted.

The old designs contained a lot of possibilities to turn off the supplies for almost everything except for the micro controller. Most of these were removed except for the possibility to turn off one of the current measurements. As described earlier only two of the three phase current measurements were needed, because the sum of all three should be equal to zero. All the other chips that was getting power from the ±15V could be turned off by turning off the power supply.

RDC board

Below is a list of the changes to the old design for the RDC board.

- **Correct the routing mistakes.** First of all the unrouted and shorted mistakes had to be fixed.
- **Smaller cable to the main controller board.** The absent need for turning off the RDC chip and the MOSI pin meant that the cable could be smaller. The new cable was a 2x5 pin and polarized, which meant that it was much easier to reconnect the board without knowing which direction the cable should be plugged.
- **New terminal blocks.** The terminal blocks were switched out for ones that were possible to plug in an out.
- **Smaller.** The board was made smaller so it would take up less space in the car. This was important because the space for the electronics in the car was quite limited.
- **Ground plane.** The prior board had no ground plane. A ground plane is important to lead all currents with less resistance to digital ground, to shield better against electrical noise and to reduce ground loops. in addition, it makes it easier to route the board.

- **Amplifier heat plane.** This was already in the design for the prior board, and was also included in the new design. The amplifier needed a heat plane to dissipate the heat generated from its operation. The heat plane could only be connected to ground in single supply application. The amplifier used dual supply and therefore an isolated heat plane was necessary. This was made as big as possible for good heat dissipation.

- **Temperature measurement.** The motor included a PTC resistor for measuring the temperature of the motor [16]. This was matched with a voltage source and a resistor to give a scaled input for the micro controllers ADC.

- **Mounting holes.** Mounting holes were added in all four corners to make it possible to mount the PCB.

- **Silk screen.** This was not on the prior board since it was milled out. The silk screen had information on the components names which makes it easier to solder. In addition the input and outputs had names to easy connect the peripherals without inspecting the design. The name SEM was also included as it was a part of the rules to include this on the motor controller purpose made boards.

Picture 4.18 shows top and bottom of the finished PCBs for the RDC board.

Main controller board

Below is a list of the changes to the old design for the main controller board.

- **Level converters.** The old level converters which used pull-up resistors were switched out for a TTL type non-inverting buffer. The transistor on signal was also included, as this was only at 3.3V level before. The feedback signals of the old design did not have level converting. Since these were only input signals, and therefore only converted down, a simple voltage divider was used for the converting.

- **Ground plane.** Ground plane was added for the same reasons as for the RDC board.

- **Added CAN port.** One more CAN port was added to be able to relay the CAN bus to other modules in the car. If the board only had one port it had to be the end node or a special

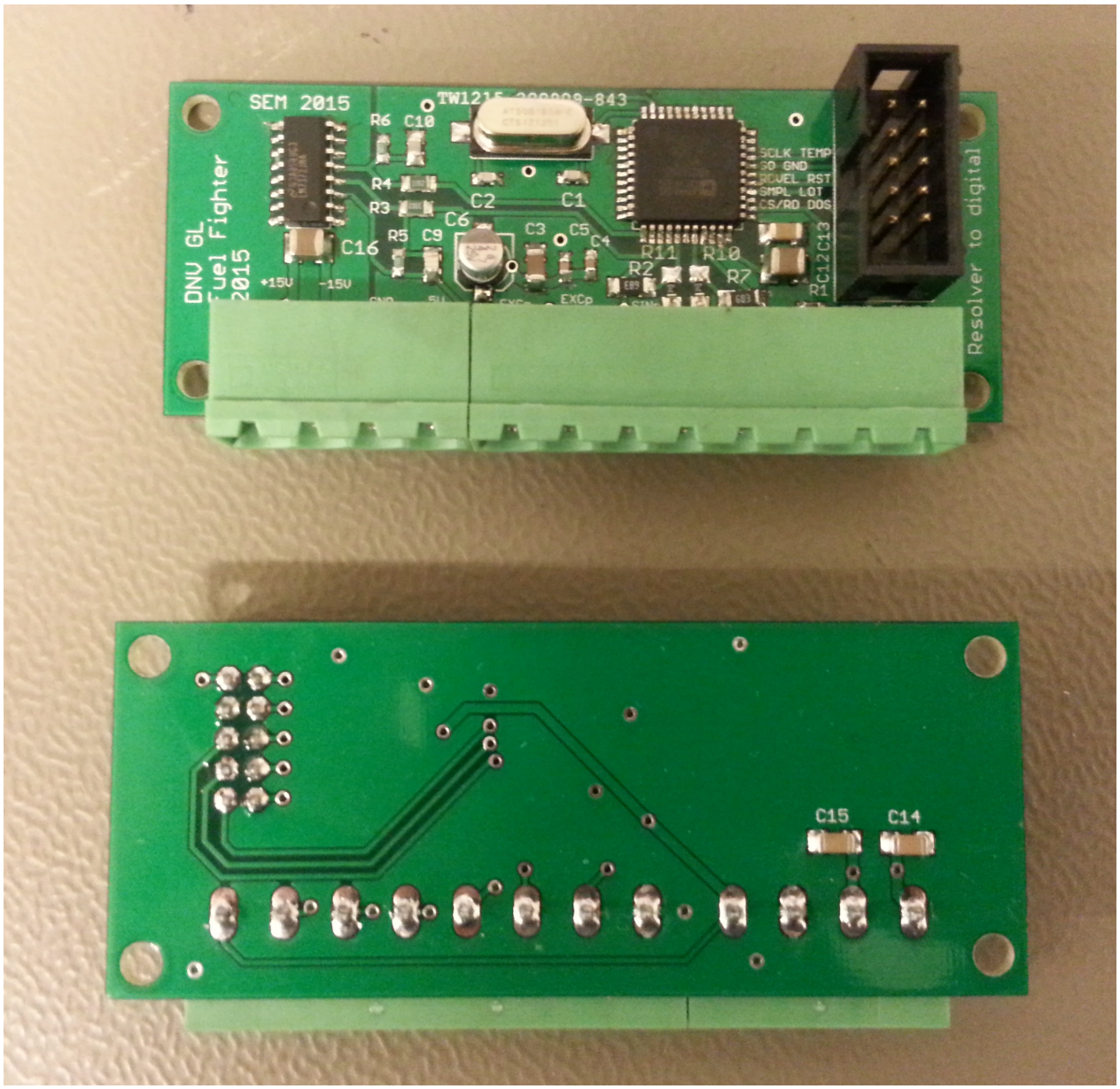


Figure 4.18: Top and bottom of the new RDC board.

cable had to do the relaying. The board could not be the last node in the UrbanConcept car because there was two sets of the boards. Also it would be bad design to fix the board to be the end node.

- **Added ethernet port for logging.** An ethernet port was added to be able to log data. Reasons for this choice will be discussed in the software section.
- **Added on/off switch for ±15V power supply.** As explained earlier, it was added a switch for turning off the ±15V power supply.
- **New terminal blocks.** As with the RDC board, the terminal blocks were switched out for ones that were possible to plug in an out.
- **New connectors for the inverter measurements.** The connectors for the current and voltage measurements from the inverter was changed to polarized ones, to avoid connecting the wires the wrong way.
- **Exclusion of USBTX and USBRX.** Some problems were experienced when using the USBTX and USBRX pins as GPIO. These were specialized pins for USB communication. Because of other unoccupied pins, these were left out of the design.
- **Smaller.** As on the RDC board, it was a desire to make the board smaller to take up less space in the car.
- **Mounting holes.** As on the RDC board, mounting holes were added to make it possible to mount the PCB.
- **Silk screen.** A silk screen was added the same way as for the RDC board.

Picture 4.19 and 4.20 shows the finished PCBs for the main controller board. One of the CAN port was not soldered. The reason for this will be described next.

Notes on the new designs. The new designs had some faults, but these were minor and could be dealt with. The faults are summarized here.

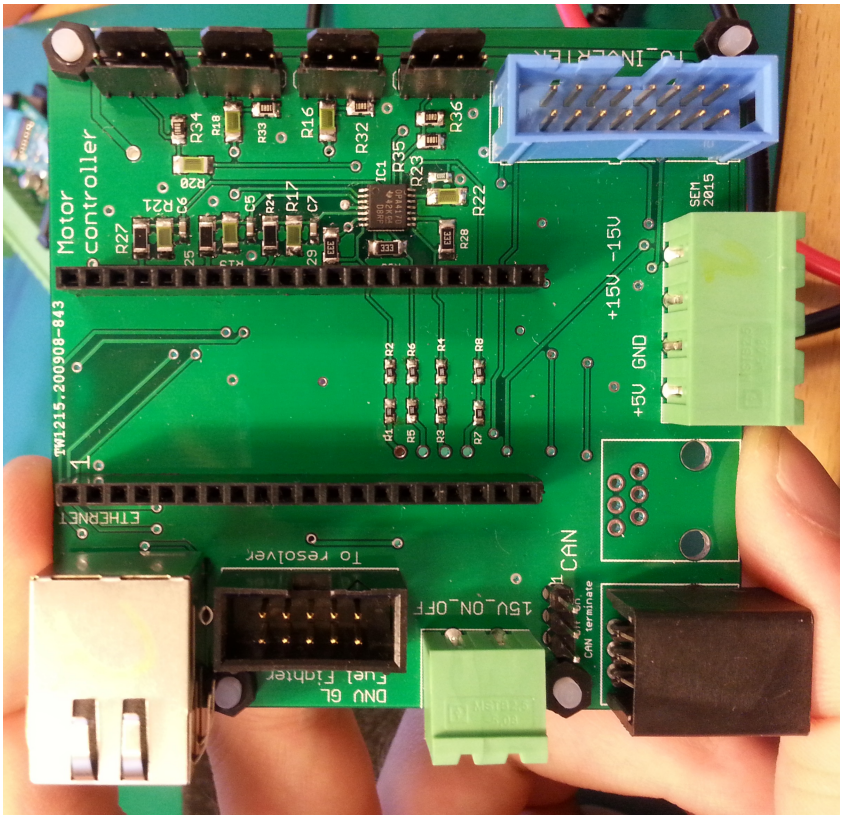


Figure 4.19: Main controller board top.

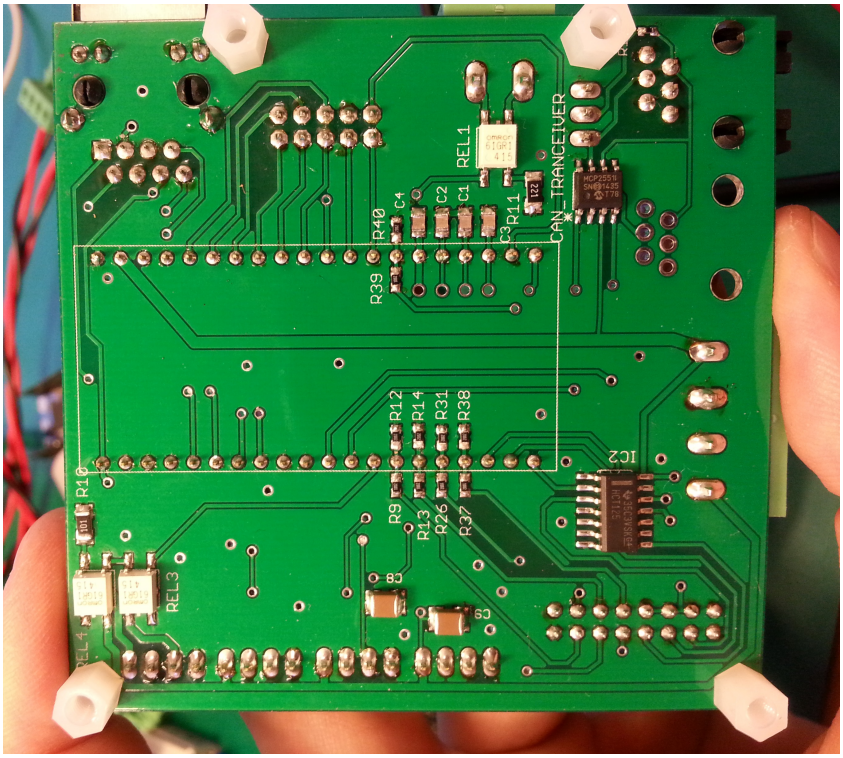


Figure 4.20: Main controller board bottom.

- **Not all pins relayed on the CAN ports.** The connector for the CAN bus consisted of six pins, of which four were used in the CAN bus cable. Two wires were used for the CAN bus TX and RX signals, while two other wires were used to power most of the modules on the bus. The main controller board had to be powered by the propulsion system as stated in the rules and therefore did not use these pins. For nodes connected after the main controller board that used these pins however, they needed to be relayed. This meant that no module that got power from the CAN bus cable could be connected after the main controller board. For the Prototype car, to avoid connecting wrong, only one port was soldered as we can see in the pictures 4.19 and 4.20.

- **No decoupling capacitors on ICs.** Decoupling capacitors are often used to filter out undesired noise from the power supply and is connected as close as possible to the positive power of an IC. Some of the ICs on the circuit did not have decoupling capacitors, which could lead to unwanted behaviour. Luckily the ICs worked as they were supposed to.

- **Current measurement amplifier inverted.** The amplifier for the current measurement was in an inverted configuration. This was simply dealt with in the code by multiplying with minus one.

4.4 Software

4.4.1 Simulink to code

Since the simulations were done in Simulink and the microcontroller was running c++ code, the Simulink blocks had to be transformed to code. Here there were two alternatives, either using a code generator in Simulink or write all the code manually. The latter was chosen because even when generating code automatically from Simulink, code for the drivers and hardware interfaces had to be implemented. To have the code generated in Simulink could also be less efficient and more time demanding than writing all the code manually [23], although in some cases it is advantageous [28]. The complete Simulink diagram can be seen in figure 4.21. The blue blocks in the diagram represents what was to be implemented on the microcontroller. Each of the blocks will be described here, including additional information on some of the blocks on

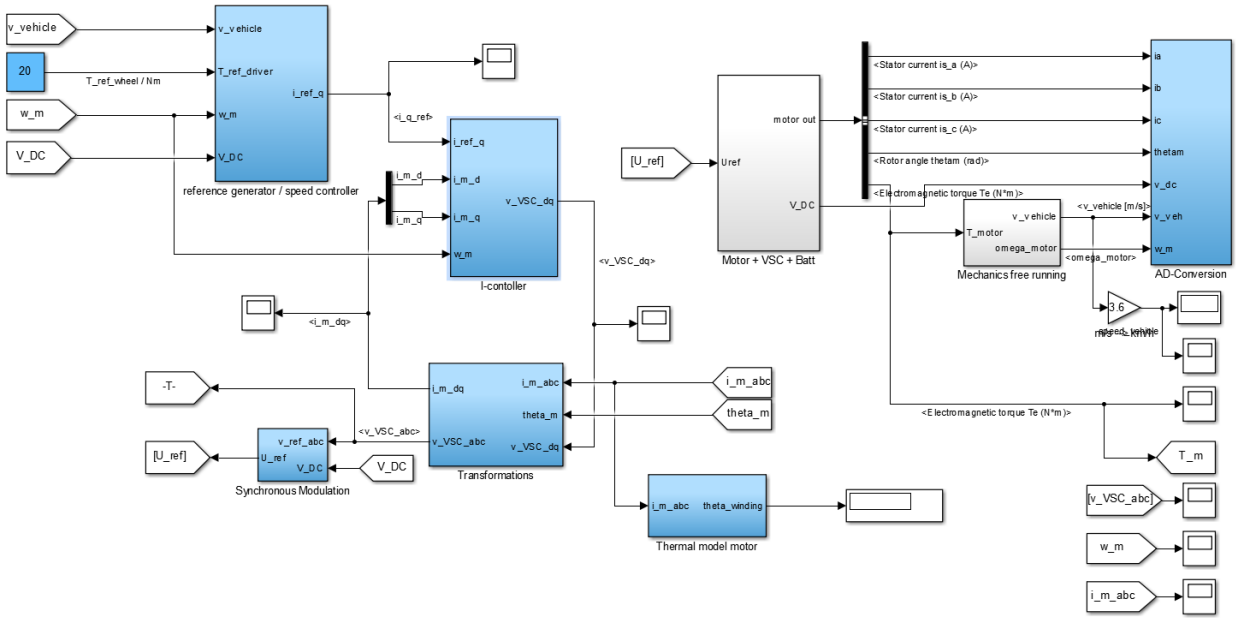


Figure 4.21: Motor controller Simulink diagram. The blue blocks had to be implemented on the microcontroller.

how they were implemented on the microcontroller. The equations implemented on the microcontroller are included in the appendix. For a more thorough understanding of the simulation, it is recommended to read [7].

I-controller with observers

The I-controller or current controller had the objective to control the dq currents to the desired values. The I-controller model can be seen in figure 4.22. Unlike the normal PI controller showed in figure 4.3, the controller used a reference tracking strategy with estimators for the real currents. The reason for the choice of controller is discussed in [7]. The decoupling terms were also included here. No reference for the d current was included since this is simply has a value of zero.

Transformations

The transformation block transformed the measured currents in a, b and c values into d and q values to be used in the I-controller, and transformed the d and q voltages from the I-controller

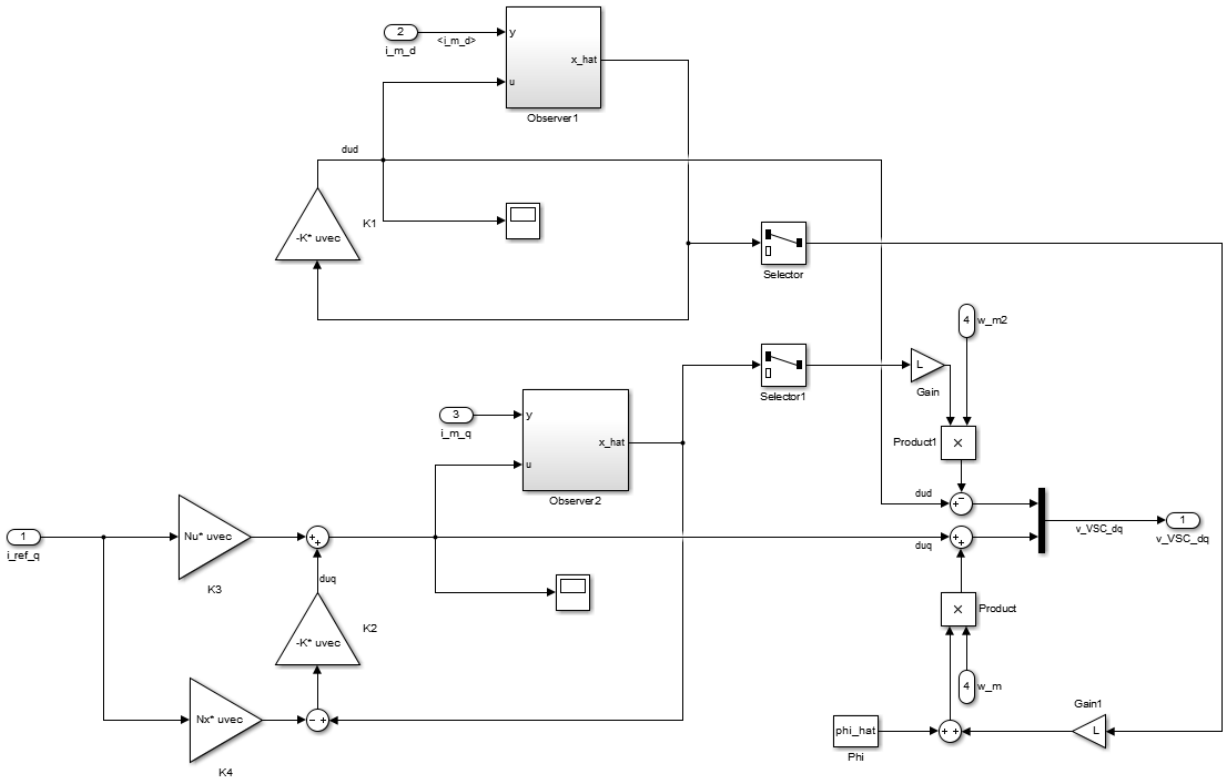


Figure 4.22: Simulink I-controller block.

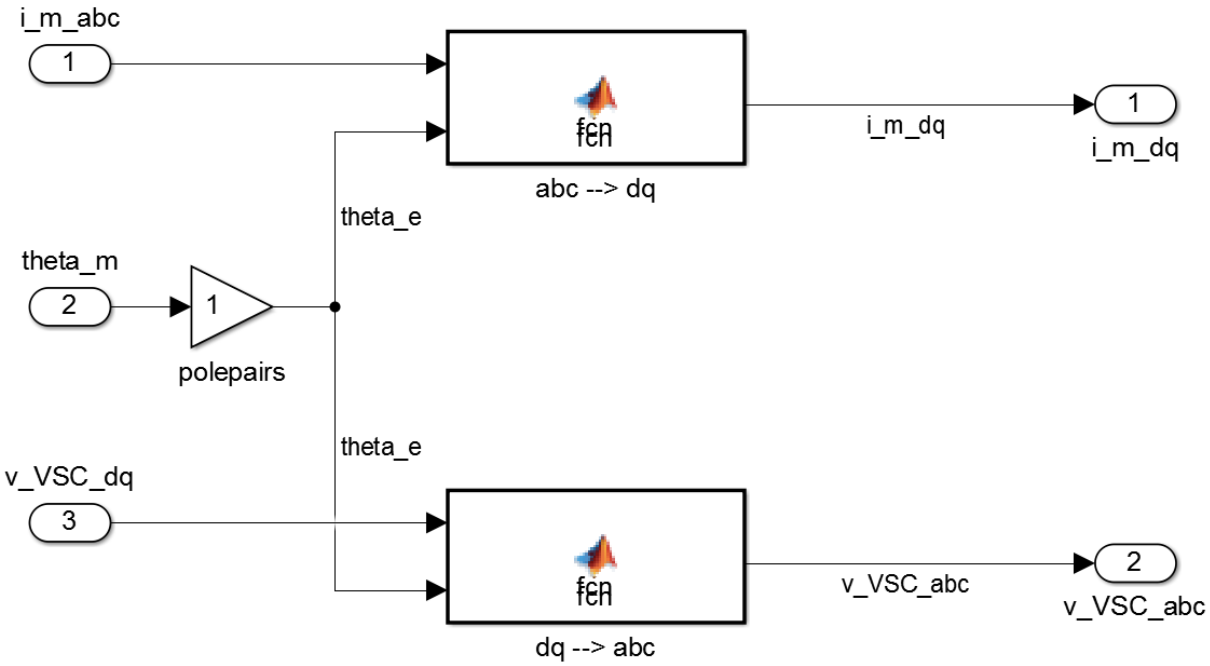


Figure 4.23: Simulink transformation block.

to a, b and c values to set the voltages for the motor. The angle of the motor, called $theta\_m$ was needed to do the transformation. The model for the transformations is shown in figure 4.23.

Synchronous modulation

The synchronous modulation block was responsible for scaling the abc voltages to the PWM values and applying symmetrical sinus modulation. For more information on symmetrical sinus modulation, see [7]. The PWM could only take a number of values, so a saturation was needed to avoid values outside the range. For testing, the function output was a value between -1 and 1 as in the simulink model, but for implementation the values corresponded to PWM with duty cycle between 0 and 100 percent. The Synchronous modulation block is seen in figure 4.24.

AD conversion and PWM

The AD conversion block was realized by an ADC library. The PWM output was realized with a PWM output library.

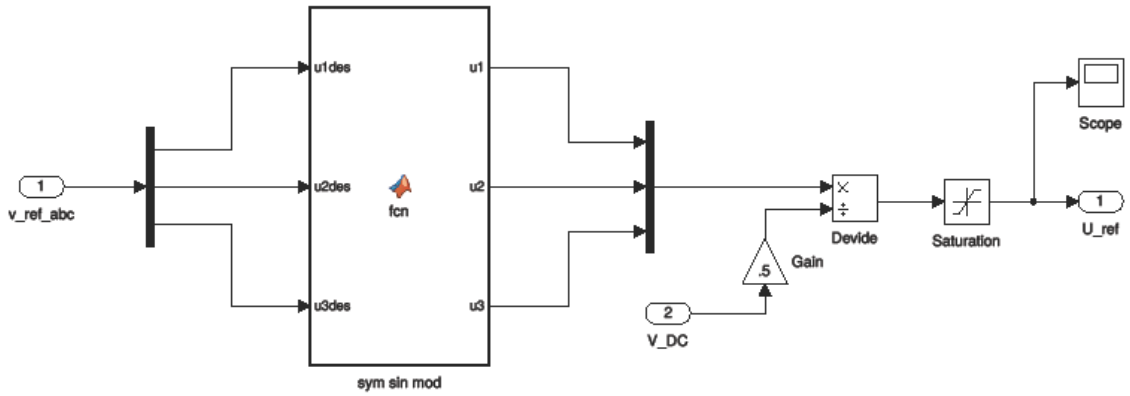


Figure 4.24: Simulink synchronous modulation block.

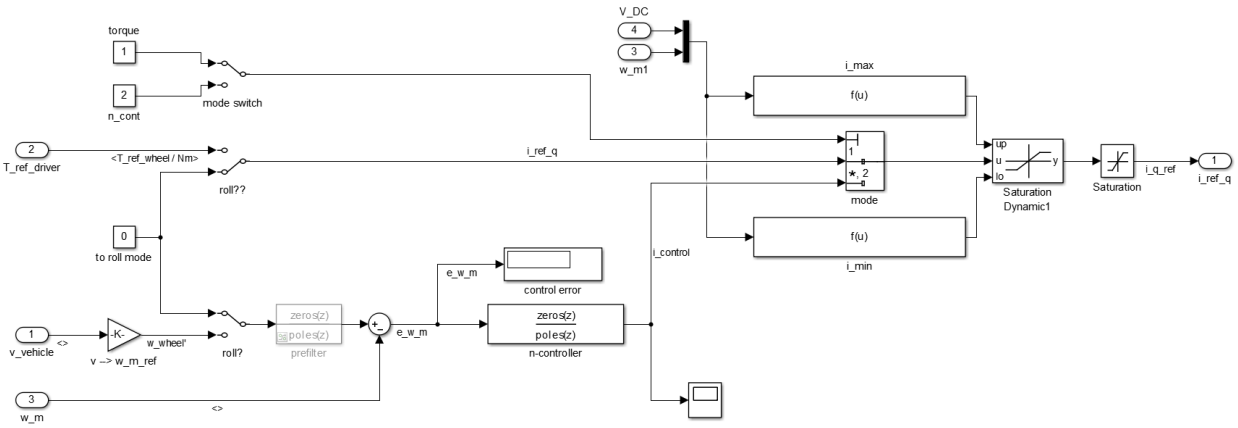


Figure 4.25: Simulink reference generator/speed controller block.

Reference generator/speed controller

This block can be seen in figure 4.25 and was responsible for generating the reference current. It was comprised of the driver reference (*T_ref_driver*), the speed controller and the dynamic saturation. The switches were for choosing whether the speed controller should be on or not.

Driver reference The reference would come from the driver of the car via the CAN bus. The reference would be the torque, which is proportional to the current reference (i_ref_q). A CAN library was used to fetch the appropriate CAN messages from the CAN bus. The CAN messages was fetched asynchronous by an interrupt routine. When in speed control mode, the reference would come from the speed controller.

Speed controller Both of the cars were using a one way bearing on the wheel, similar to those used on bicycles. This was because the motor control system was using a lot of power when running in standby. When the motor was not in use, the control system would be shut off and the motor rolling with zero energy used. When starting the motor control system again while the car was rolling, the wheel and the motor would have different speeds. The goal of the speed controller was to connect the motor and the wheel in a smooth transition. The speed of the wheel was measured by another module in the electrical system and sent to the motor controller via the CAN bus. A PI controller was used to connect the wheel and the motor, along with low

pass filtering of the reference torque to further smooth the transition.

Dynamic saturation The dynamic saturation was for limiting the reference torque if the speed was too high. The motor could only reach a certain max speed with the control used here. The controller could not reach a torque that would increase the speed of the motor when it was running at top speed. This would make the controller inefficient. Therefore it was better to limit the reference torque when the speed was too high. This was what the dynamic saturation did. The formula for the saturation was

$$i_{q,max} = \frac{a_{max} * V_{DC} - \frac{3}{2} * \hat{\Phi} * \omega_{motor}}{R + \omega_{motor} * L}$$

If the DC voltage was assumed constant, the only variable was the speed of the motor ω_{motor}. Higher speed would lead to lower $i_{q,max}$.

Thermal model motor

The thermal model of the motor was unnecessary to implement, since the motor included a thermal sensor. In addition, this was a safety feature that was not absolutely necessary for the system to work.

Test functions

To test that the embedded functions worked as the Simulink ones, it was important to make simple test functions. Test functions for the control loop including the I-controller, transformations and synchronous modulations were made.

The test functions would take in an array of simulated input values and print out the computed values. The input values was taken from values in the Simulink simulation. The output from the microcontroller was compared to the output of the Simulink simulation. The difference between Simulink output and microcontroller output for the whole control loop can be seen in figure 4.26. Simulation was done in 40 milliseconds which included a full revolution of the motor. The reason for the short simulation was the limited memory on the microcontroller. The output takes values between -1 and 1. From the figure it is clear that the error is less than

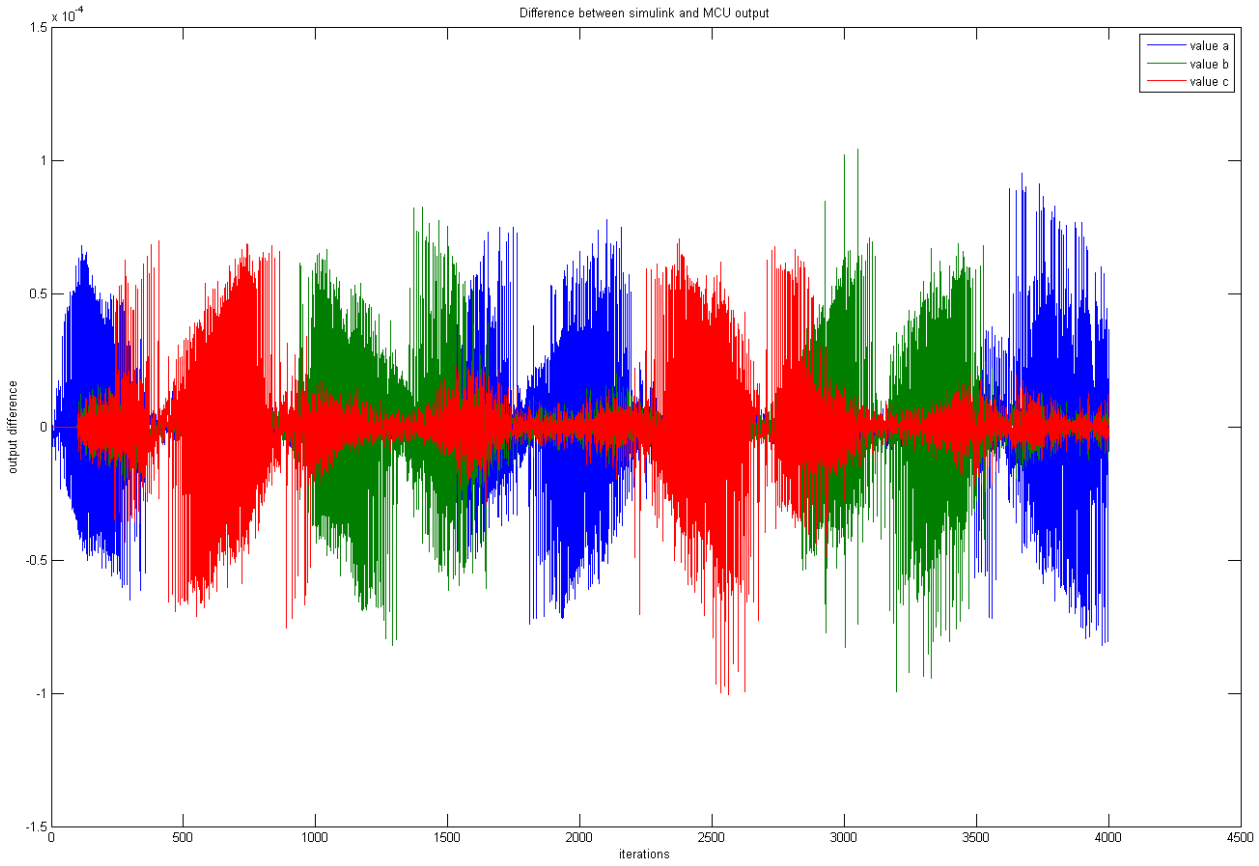


Figure 4.26: Difference between simulink output and microcontroller output using floating point calculations.

10^{-4}, which should be much less than other inaccuracies in the system (e.g. ADC, position and velocity measurement).

4.4.2 Optimizing the code

The Simulink iteration frequency was 100kHz which is equal to 10 microseconds per iteration. In the test function for the controller a timer was implemented to measure the time of the whole test. To get a realistic time, the print function was commented out during the test, since this would influence the time considerably. The time per iteration was simply the time returned from the test function divided by the number of iterations. Running the test with the input from Simulink gave the result:

```
time per iteration in us: 165
```

This was without all retrieving of data and setting of output. Adding those, the rate could be estimated to be below 5 kHz. Having a so low frequency would make the quality of the torque quite bad. Consulting with the designer of the simulink model an update frequency of about 40 kHz should be the minimum. Therefore some optimization of the code was needed.

Fixed-point calculation

The NXP LPC1768 uses a 32-bit ARM Cortex-M3 core running at 96 MHz. This processor does not have the hardware support for floating-point operations. The floating point operations must be emulated through software, which can take up a great deal of CPU cycles. If using fixed-point calculations, there is no need for the conversion, and computations can be much faster. Therefore, avoiding floating point numbers (floats and double) could reduce the calculation time considerably.

A fixed point number has a specific number of bits reserved for an integer part and a specific number of bits for a fractional part. An integer is a fixed point number with zero fractional bits. One of the normal notation is the Q notation: Qm.n, where m denotes the number of integer bits with or without a sign bit, and n denotes the number of fractional bits. Computation with fixed point numbers are faster, but they have less accuracy and/or less range than floating point numbers. However this is not a problem if the range is known and the accuracy is of less importance.

A new variable type was made with a fixed number of bits before and after the comma. This number specified both how large a number could be represented, and how accurate the numbers would be. By inspection of the Simulink model combined with some knowledge about the inputs and outputs, a safe range for the variables was assumed to be $\pm 2^{12}$. This meant using 1 sign bit, 12 integer bits and 19 decimal bits which could represent approximately ± 4096. The format will from now on be called Q19.

To be able to use the same operands when adding, subtracting, multiplying and dividing these variables, the operands had to be overloaded. This was done because the conversion from the previous floating point calculations would be much easier, in addition to creating increased readability. Addition and subtraction were done in the same way as with normal integers if one

did not check for saturation. Multiplication was implemented as such [2]:

```
// precomputed value:
#define K   (1 << (Q-1))
signed int       a, b, result;
signed long int  temp;
temp = (long int)a * (long int)b; // result type is operand's type
// Rounding; mid values are rounded up
temp += K;
// Correct by dividing by base
result = temp >> Q;
```

Division was implemented as such [3]:

```
signed int  a, b, result;
signed long int temp;
// pre-multiply by the base
temp = (long int)a << Q;
// So the result will be rounded ; mid values are rounded up.
temp += b/2;
result = temp/b;
```

New controller functions was made in order to do calculations with the new variable type. The test was run with precomputed values in Q19 format from the Simulink values. This gave the result:

```
time per iteration in us: 37
```

Which was 27kHz and between one fifth and one forth of the result using floats. This was still not good enough for the limit of 40 kHz. The difference between Simulink output and microcontroller output can be seen in figure 4.27. The result is approximately the same as with the floating point calculations.

[2] http://en.wikipedia.org/wiki/Q_(number_format)
[3] http://en.wikipedia.org/wiki/Q_(number_format)

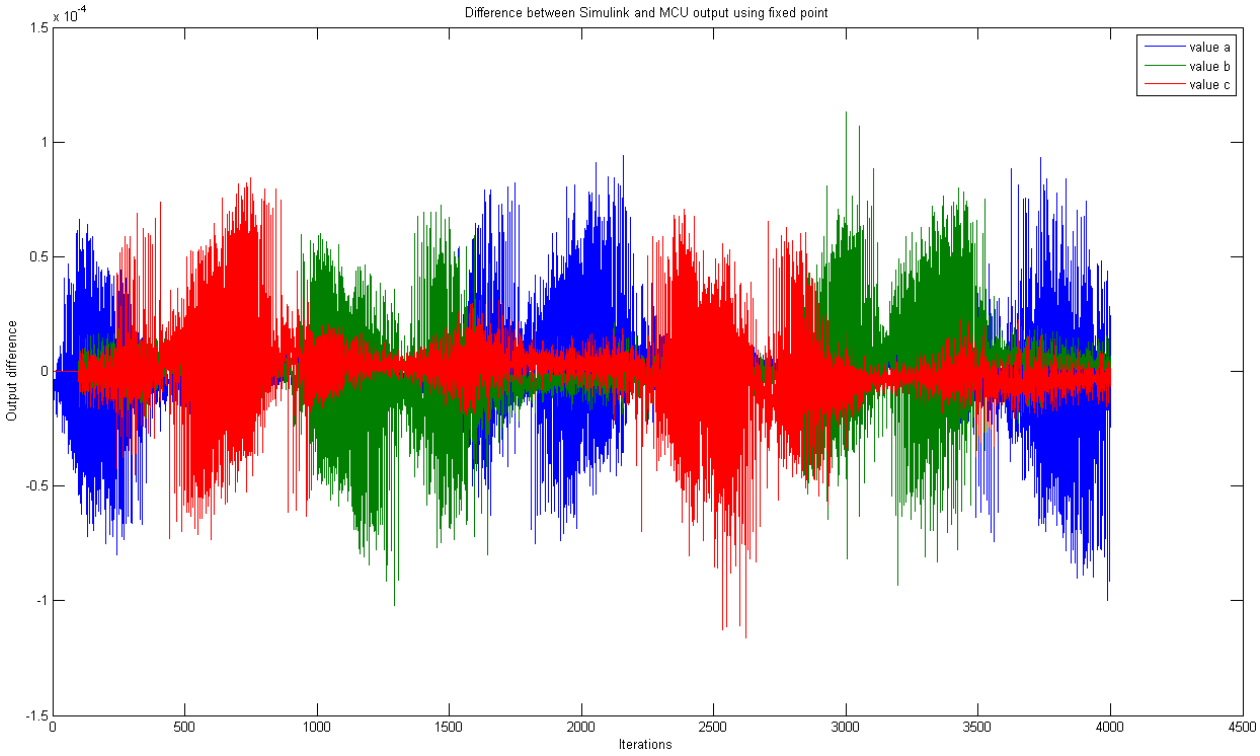


Figure 4.27: Difference between simulink output and microcontroller output using fixed point calculations.

Trigonometric lookup tables

The cosine and the sine calculations used in the transformation functions still used floating point numbers. Cosine and sine function can use a lot of CPU cycles, especially when the hardware does not support floating-point operations. When replacing the cosine and sine calculations with constants the time per iteration was just 15 microseconds. This meant that there was a huge potential for optimization here that could be utilized.

The angle of the motor would come from the resolver to digital converter, combined with the angular speed of the motor. It was represented as a 12 bit number ranging from 0 to 360 degrees, where all 0s correspond to 0 degrees and all 1s correspond to 360 degrees. The fastest way to calculate the sine and cosine values of the angle would then be to simply map the values from the resolver to the cosine and sine values. This was done by creating two lookup tables, one for the sine values and one for the cosine values. These tables was then put in the flash memory of the microcontroller by declaring them constant. The table was generated in Matlab with the equations

$$sin\_lookup\_table\_q19[i] = sin(\frac{i * 2\pi}{4095}) * 2^{19} \tag{4.1}$$

$$cos\_lookup\_table\_q19[i] = cos(\frac{i * 2\pi}{4095}) * 2^{19} \tag{4.2}$$

Running the same test as earlier gave the result:

```
time per iteration in us: 15
```

This was a value that was inside the desired update frequency. The difference between the microcontroller output and Simulink output can be seen in figure 4.28. The error was now considerably larger due to the new trigonometric lookup tables. However this was as good as it could be when the resolver put the limit on the accuracy. The output ranged from -1 to 1, so the error was up to 10% of the output range in the spikes.

The different methods and the iteration times are summarized in table 4.2.

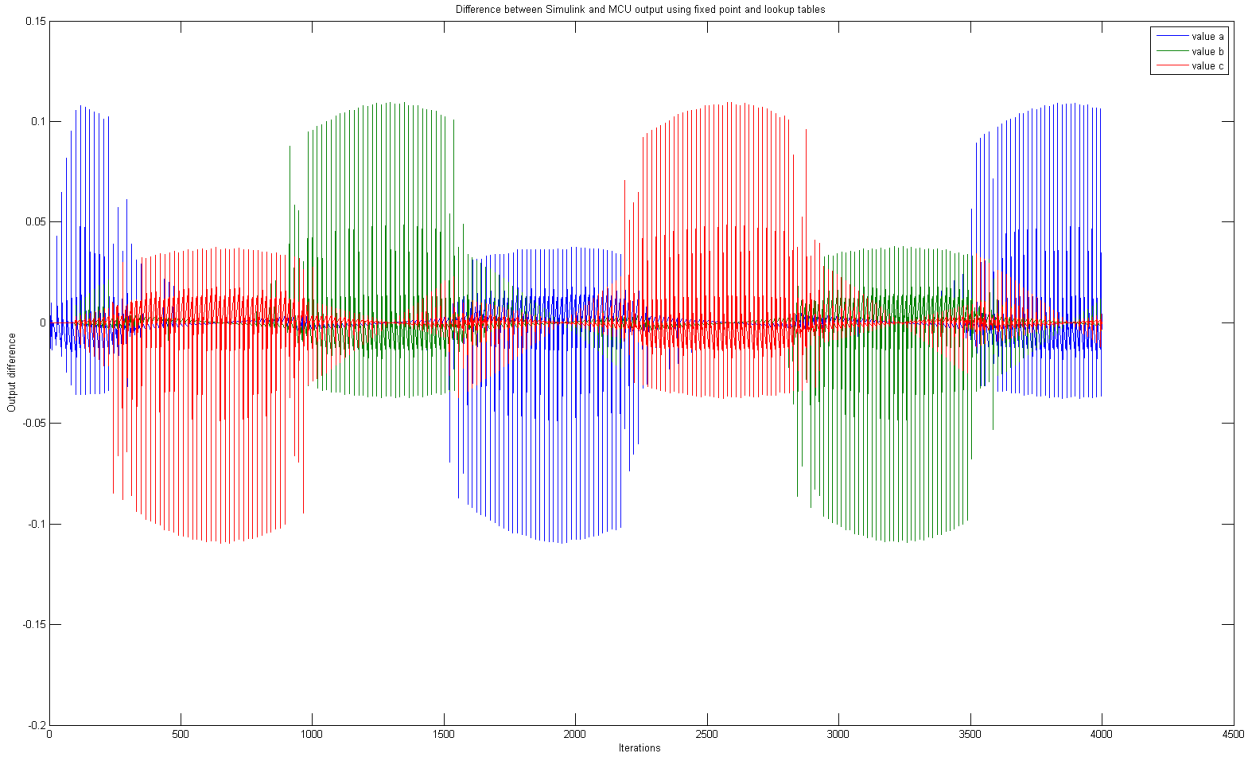


Figure 4.28: Difference between simulink output and microcontroller output with fixed point calculation and lookup tables.

Method	**time per iteration**
Floating point	165 us
Fixed point	37 us
Fixed point + lookup tables for sine and cosine	15 us

Table 4.2: Time per iteration for the different methods.

ADC, PWM, CAN and SPI

The ADC, PWM, CAN and SPI represented the interfaces of the input and output of the controller. The values from these interfaces came in 16 bit or 8 bit formats. Scaling and conversion to the fixed point format was required. To execute this as fast as possible, the scaling and conversion was done in one calculation. The equations for this can be seen in the appendix.

Measuring the time for the native ADC and PWM libraries showed that the functions used too much time. From experience with 8-bit micro controllers this should be almost as fast as writing or reading from a register. Third party ADC and PWM libraries were used instead, allowing the ADC to work in the background and directly manipulate the PWM register.

The SPI interface to the RDC board had to be as fast as possible. The clock rate was adjusted to 25 MHz which was the maximum possible clock rate for the RDC chip. A number of different pins were also used to control the acquisition of position and velocity. Care had to be taken too stay within the time limits specified in the datasheet [4]. An oscilloscope was used to check that the timing was inside the limits.

4.4.3 Logging of data

Due to the complexity of the program it was desired to have real time logging of important data. This was necessary to see that the controller was working properly.

Requirements

The requirements consisted of how much time the logging was allowed to take and how much data could be sent. These two requirements along with the overhead needed to send the data, would set the requirement for the data rate.

Time The iteration time was estimated to take around 25 us total, including retrieving input, calculation, and setting output, in addition to other features. The controller parameters were calculated based on the iteration frequency, and the controller could become unstable if the frequency was changed. A limit was set so that the logging time should not be more than 20 percent of the iteration time, which gave 5 us per iteration for sending data.

Data The d and q current, the output voltages and the position and velocity of the rotor was a minimum of data that should be sent. The phase current and voltages could be calculated from the dq values and vice versa if position of the rotor was known. Therefore, only one of the two values was needed. Velocity was important because this was also a part of the calculation. If position and velocity was the measured 16 bit values and the currents and voltages were the fixed point 32 bit values, the data size would be:

$$16 * 2 + 32 * 4 = 160\ bits = 20\ bytes$$

With the time limit of 5 us the data rate then would have to be at least (not including the overhead)

$$\frac{160\ bits}{5\ us} \approx 32Mbps = 4MB/s$$

Data viewing The logging should also have a way of displaying the data, preferably on a computer or separate screen. The transition between data acquisition and data viewing should not be made too difficult, to allow for fast analysis of the data.

Options

Below is a list of some of the different options that was considered. A number of other alternatives could be discussed, but these were the most relevant.

- **Serial interface.** The development board came with a USB connector for programming the micro controller. The same USB connector could be used to send data over the serial interface (UART). Maximum baud rate was 230.4 kbps, which was already far below the requirement.
- **SD card (SPI).** Another way was to store the data to an SD card. SD cards support SPI. An SD card could therefore be interfaced with the micro controller. However, a study has shown that the speed is quite limited [11], only 340 kB/s with pre allocation of file size.
- **Ethernet.** The LPC1768 included an ethernet controller for connecting the micro controller to the internet. The development board included the physical driver, so the only

part that was necessary for connecting to the internet was a RJ45 jack. When using the socket library the speed could be as high as 5.852 Mbps [4]. If using ethernet packets the speed should be higher, and could possibly reach the required data rate.

The different options showed that only the ethernet interface was close to the requirements. This was the reason why the RJ45 jack was included in the final design for the main controller board described in the hardware chapter.

Data viewing

Using the ethernet interface meant that a program had to be made on the computer to collect the packets and display them on the screen in some way. Because the socket library did not fulfil the data rate requirement, it was chosen to send ethernet packets. The main controller board was connected to the computer ethernet port through a networking cable. A program on the computer retrieved the packets and display the data as graphs on the screen.

The program required access to the ethernet interface of the computer and involved bit manipulation of the data. For these reasons, together with using windows operating system, the programming language chosen was C# with the .NET framework. C# is similar to the C and C++ languages.

The data rate of the logging was higher than the requirement, which allowed more data to be added while still not violate the time limit. The fixed point numbers were converted to floating point numbers before displayed on the graphs to show the "true" value.

Picture 4.29 shows a screenshot of the logging program. The capture button was for controlling when to capture the data, while the micro controller sent the data continuously. The list below summarizes the data sent and their name in the program. The listed elements corresponds with the data in the different graphs read from left to right.

- Phase currents - *I_a*, *I_b* and *I_c*.
- Output phase PWM values - *u_ref_a*, *u_ref_b* and *u_ref_c*.
- dq currents - *I_d* and *I_q*.

[4] https://developer.mbed.org/users/emilmont/notebook/networking-libraries-benchmark/

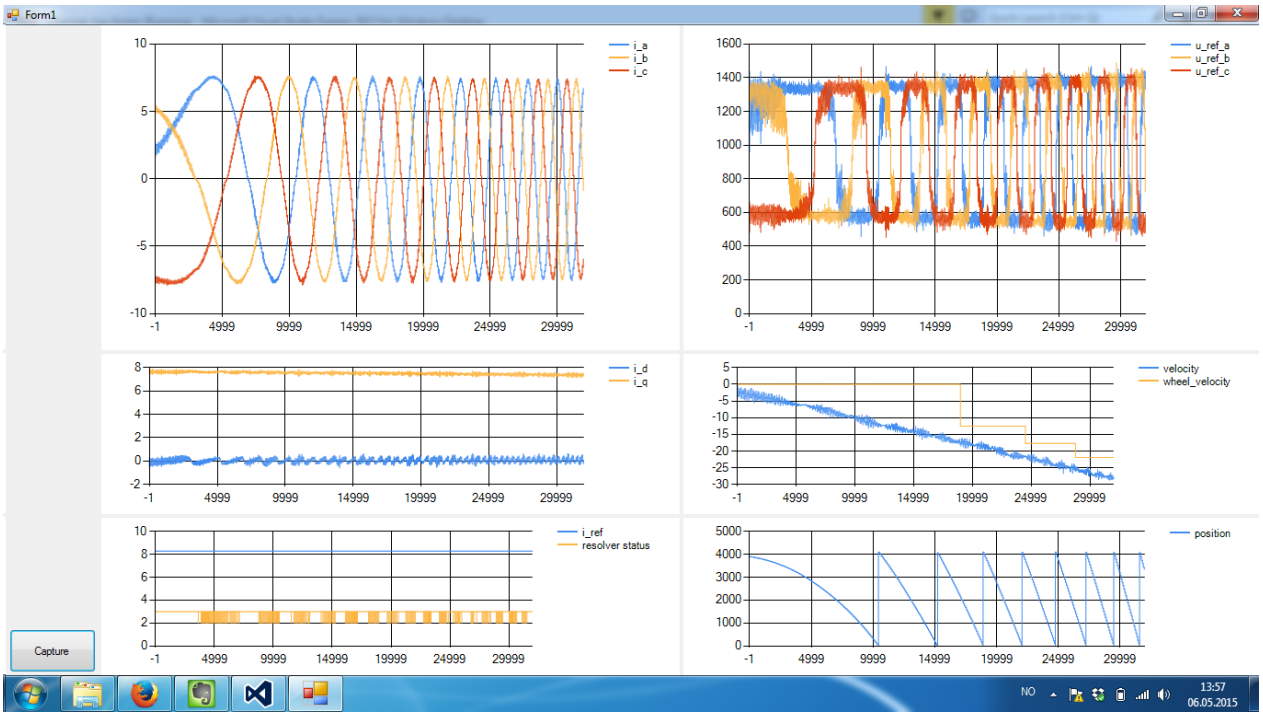


Figure 4.29: A screenshot of the logging program for the motor controller.

- Motor velocity and wheel velocity - *velocity* and *wheel_velocity*.
- Torque reference and resolver status - *i_ref* and *resolver status*.
- Motor angle - *position*.

4.4.4 Filtering

The ADC measurements on the micro controller turned out to be very noisy. In practice ADC measurements can be noisy and some filtering is required. However, these measurements were so noisy, various filtering techniques had to be used. Figure 4.30 shows the raw 16 bit ADC measurements for the currents. The sine wave of the three phase currents a, b and c in blue, yellow and red respectively, can barely be seen because of all the noise. The measurement was over approximately 5000 samples, corresponding to 5000 * 25 us = 125 milliseconds.

Most of the noise was outside the range of the phase currents, and some of the values were even the maximum value the ADC could give, $2^{16} - 1$. This was one of the aspects that could be utilized. Three techniques were used to filter out the noise.

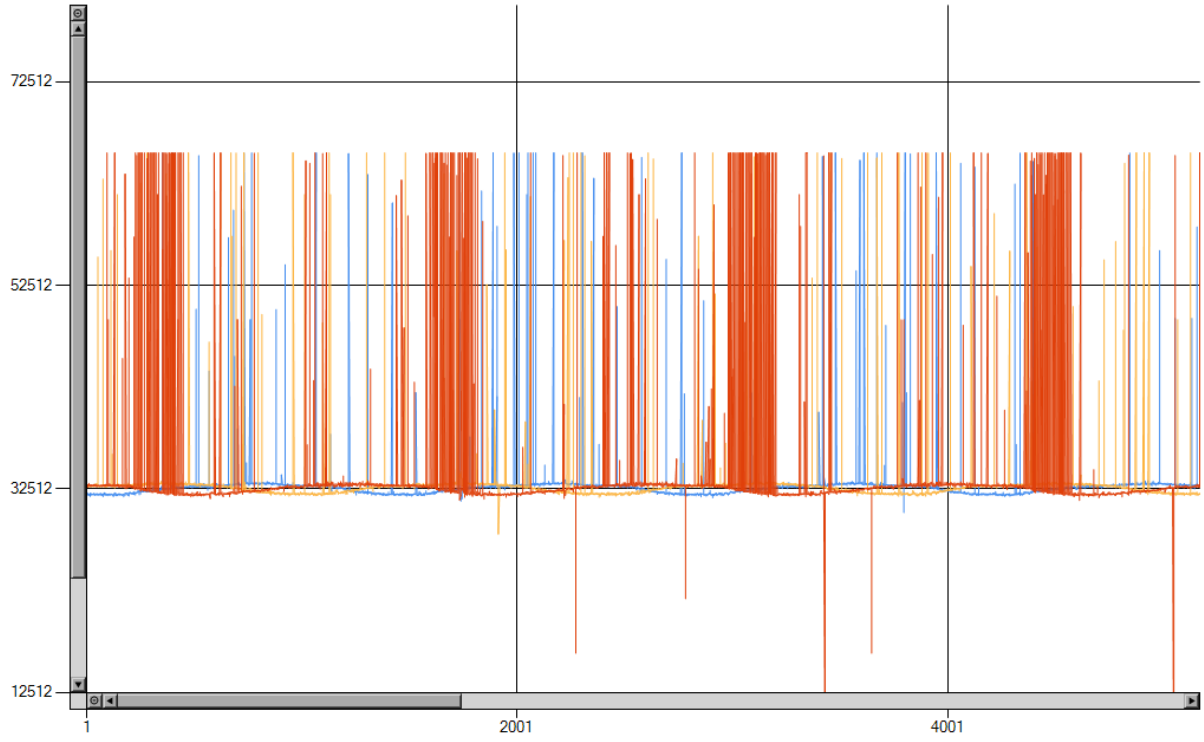


Figure 4.30: Graph showing the noise of the ADC when measuring the phase currents.

- **Remove out-of-range values.** The values that were out of range of the possible current that could flow through the motor were removed. The full range of the ADC should cover $\pm 35\sqrt{2}$A, which was the range designed from the amplifier. The limit was set at ±20A, even though these currents also were out of reach. This could be lowered if noise still was present.

- **Spike removal.** A simple spike removal algorithm was used. The goal was to remove any spikes, while still allowing large, continuous changes in values (like a step). The algorithm would discard ADC values that was too far from the last ADC value. The limit on how much change that was allowed was manually adjusted.

- **Low pass filter.** After removing out-of-range values and spikes, a low pass filter was added to smooth out the values. The filter that was used was the discrete implementation of a simple first order filter (RC filter), which is the exponential-weighted moving average [5].

$$y_i = \alpha x_i + (1-\alpha) y_{i-1}$$

Where y represents the output and x the input (ADC value). The relation between the cutoff frequency and alpha is

$$f_c = \frac{\alpha}{(1-\alpha)2\pi\Delta_t}$$

Where Δ_t is the sampling period, or iteration time in this case.

Figure 4.31 shows the ADC values for the currents before filtering on the right, and the scaled values after filtering on the left. The scaling inverted the phase current measurements, so the colors should match.

The filtering was also done on the DC voltage measurement and the velocity of the resolver. The position was not filtered because of the rapid change when going from 360 to 0 degrees. The position measurement was not noisy at all so it was no need for filtering here. Different cutoff frequencies were used on the different measurements.

[5] http://en.wikipedia.org/wiki/Low-pass_filter

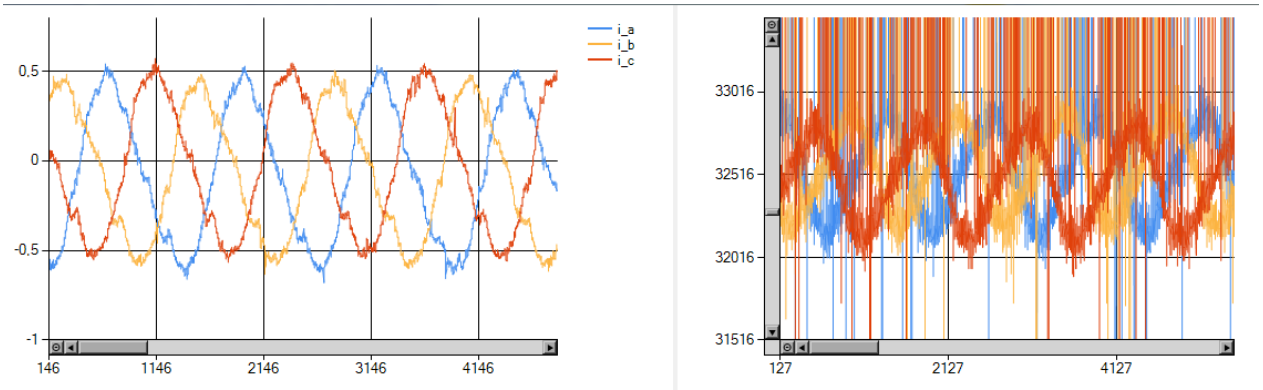


Figure 4.31: Graphs showing the ADC values after filtering and before filtering.

4.4.5 Other

This section explains various parts of the code that is of interest.

CAN timeout

A safety feature was added to disable the motor controller if a CAN message had not arrived for a given amount of iterations, which would correspond to a given time. This was done by incrementing a variable every iteration and resetting it when a CAN message from the steering wheel had arrived. The program would turn off the inverter if the variable exceeded a given limit.

The same thing was done with the speed controller. Although here the speed controller would turn off if a speed sensor CAN messages had not arrived for some time. Both the limits were set to approximately 1 second.

Speed control enabling

When the speed controller attached the motor speed to the wheel speed, the speed controller would be turned off. The speed controller was then not turned on again before the joystick was moved up from zero position. This was to allow the motor to decelerate to zero when the driver was not giving throttle.

Status packet

The motor controller logging program required a laptop to be connected and operated manually. This would be difficult, if not impossible when driving the car at high speeds. A way of monitoring the motor controller status was added. This was done by sending a CAN packet with some of the motor controller variables every second. The CAN packet could be picked up by the screen, and the 3G module to be sent to the logging system described in chapter 3.

4.4.6 Overall program overview

Figure 4.32 shows a diagram of the overall code for the motor controller. The controller was included in the code while loop to run it as fast as possible. CAN messages for the torque reference and wheel speed was retrieved asynchronously when they arrived. The ethernet initialization had to wait for about 2 seconds to establish a connection with the computer after the micro controller was turned on.

The reason for using two different i-controllers will be explained in the next section.

4.5 Testing and improvements

For the testing to be realistic, the motor should have the correct load. It was not possible to do this with the car without driving it with a person inside, and there was no time to test the motor in a lab. Therefore most of the testing with logging was done either with no load on the wheel or running the car at low speeds. Testing and logging with correct load at high speed would be very difficult because a laptop would had to be connected to the main controller.

For low speed testing of the car, the workshop was used, while for testing the car with higher speeds, the sports hall at Dragvoll was used. Picture 4.33 shows the Prototype car testing at dragvoll. The testing here was to see that the car could reach top speed and the desired acceleration. This was also to test the communication, data acquisition and logging system described chapter 3. This system could also to some degree log data from the motor controller by examining the motor controller status packets.

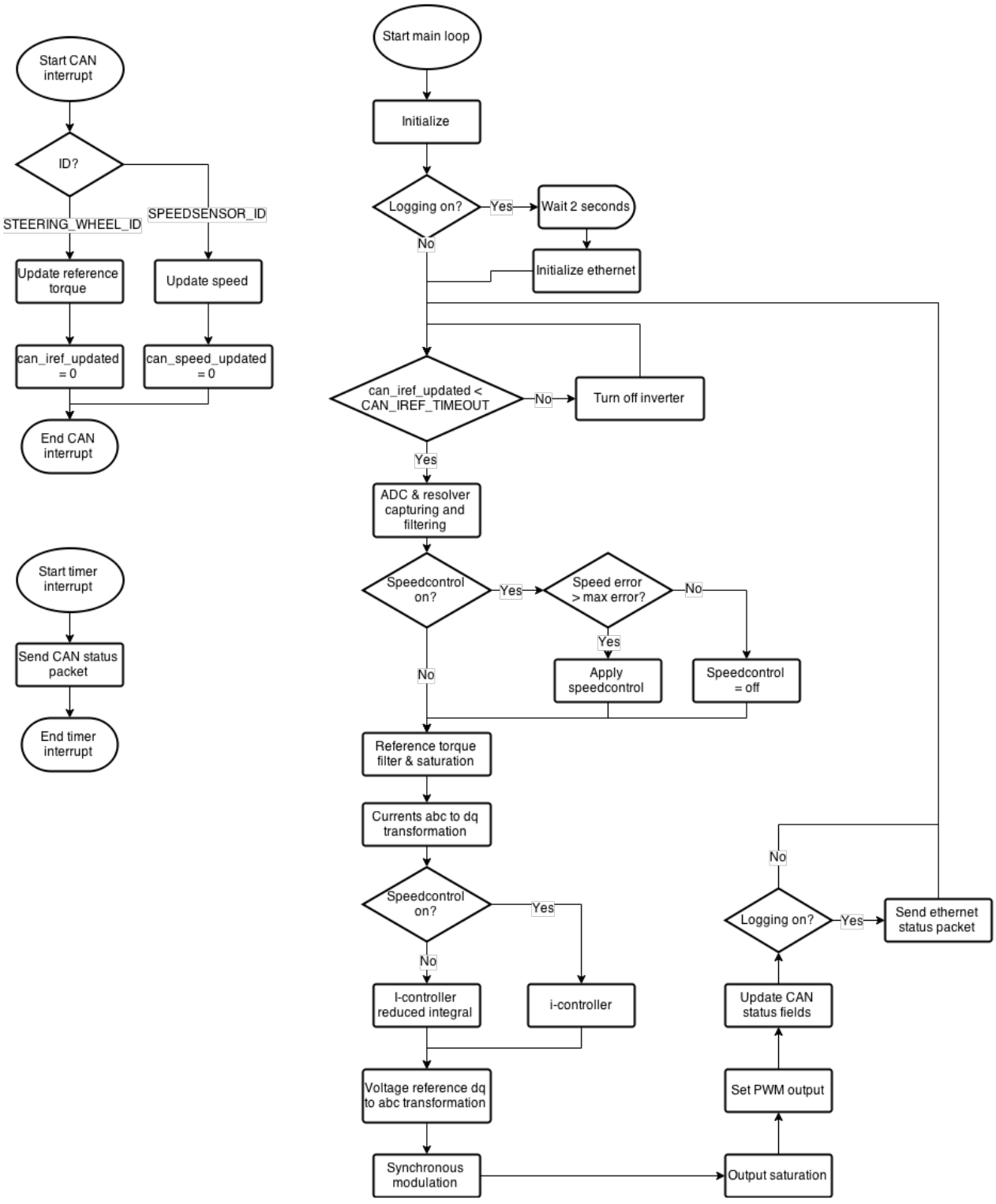


Figure 4.32: Motor controller code overview.

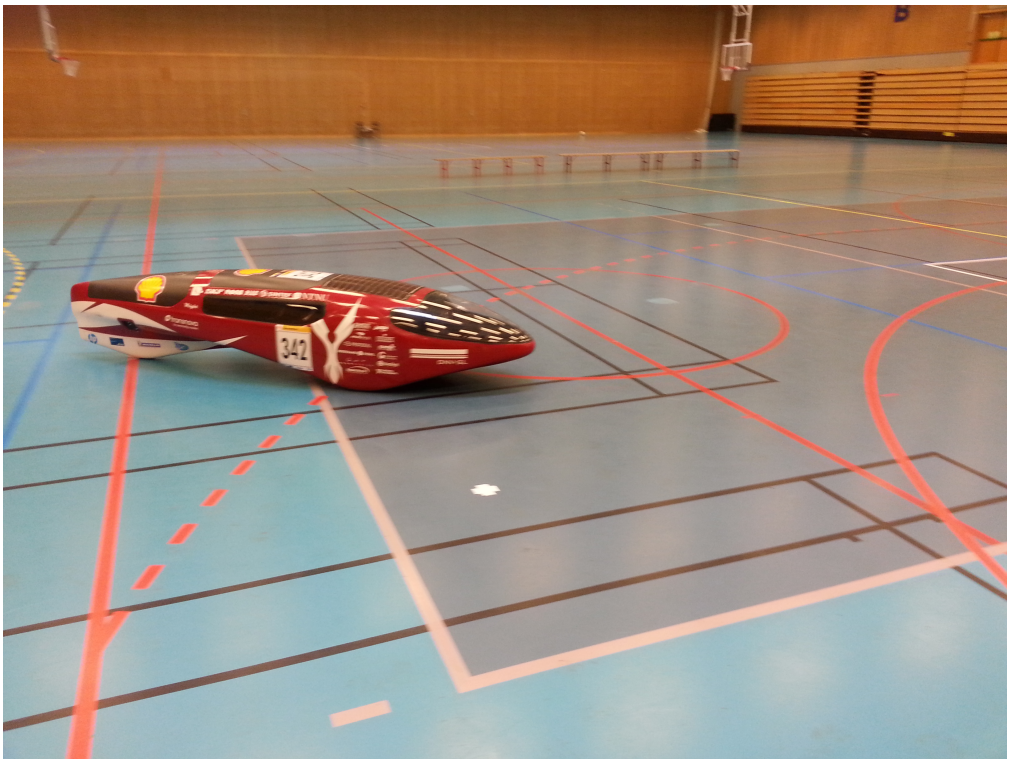
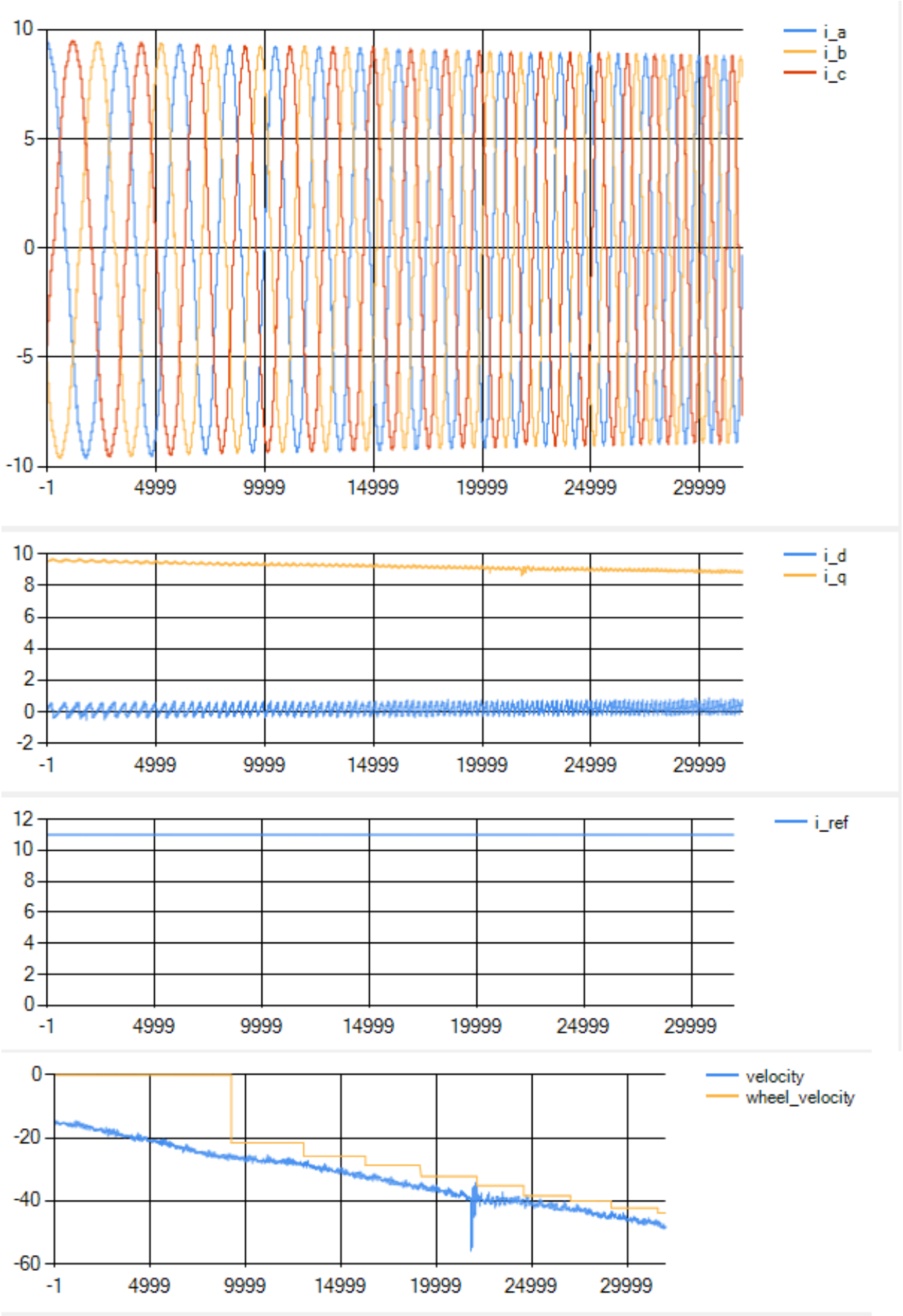


Figure 4.33: Testing of Prototype car at Dragvoll.

4.5.1 Current controller testing

The current controller was working quite well according to the logged data. Smooth sine waves was achieved for all phase currents. However, the controller was not reaching the desired reference currents. Furthermore the q current, which corresponds to the amplitude of the phase currents, seemed to decrease with higher speeds. This was further certified when testing at higher speeds. The car would only reach a top speed of around 20 km/h. Additionally, the acceleration was not as expected. By examining other variables in the controller, the reason for the decrease in q current was found to be a wrong estimation of the q current, together with a small steady state error. The estimated q current, which was the one that was regulated to the reference q current, was drifting away from the measured q current with increasing speed. Figure 4.34 shows that the q current was decreasing with speed despite the high reference q current. The current sine wave amplitudes were also decreasing as this corresponds to the q current.

Two different solutions were tried out. The first solution was adding integral action to the controller, which removed the steady state error, but the q current was still decreasing. The


10
5
0
-5
-10
-1
4999
9999
14999
19999
24999
29999
i_a
i_b
i_c
10
8
6
4
2
0
-2
i_d
i_q
12
i_ref
0
-20
-40
-60
velocity
wheel_velocity

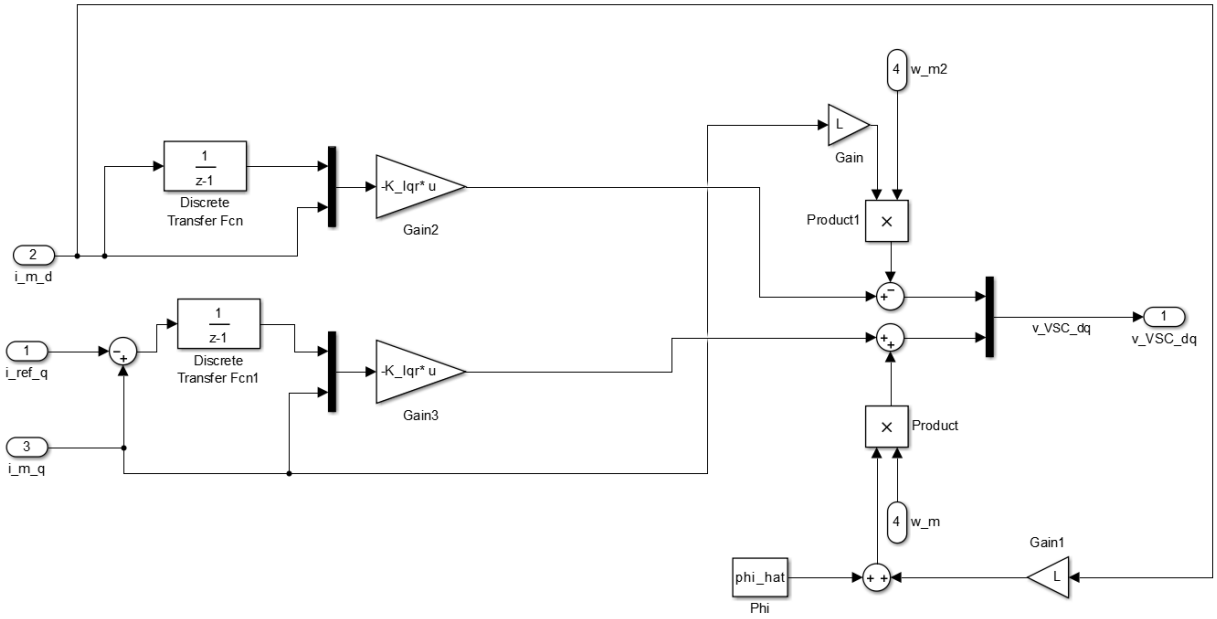


Figure 4.35: Simulink reduced I-controller with integral action.

second solution was to remove the estimators and use the measured q current directly in a PI regulator. The simulink model for this can be seen in figure 4.35. The controller gains could be calculated using various techniques. Simon Fuchs, which designed the original simulink model helped in doing this. The calculation for these gains will be included in the files accompanied with this thesis, but not discussed here.

The new controller was working as expected, and the q current was not decreasing with time. With adding the integral action a new problem emerged. The motor was making large, high frequency bumping noises at high speeds. The problem was thought to be integrator windup in the controller.

Integrator windup

All actuators have constrains. For example a valve can never be more than fully open. Windup occurs when integrator action is used and the actuator saturates. A large error from a large jump in set point can saturate the actuator. Since the saturation will decrease the error more slowly, the integrator will accumulate more because of the positive error (the integrator error will wind up). An error with an opposite sign for a period time is needed to return the integrator state

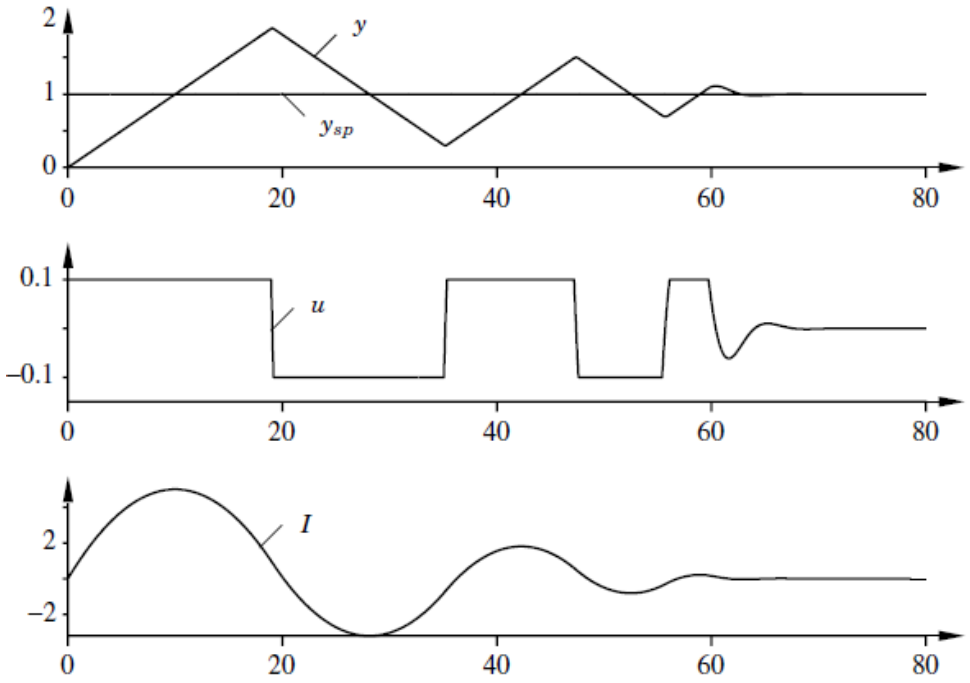


Figure 4.36: Illustration of integrator windup.

to normal. The windup will lead to more oscillations in the controller response. This can be illustrated in figure 4.36 from [24], showing the process output and setpoint (top), saturated output and integral part. The integrator part increases until the setpoint is reached, but the output is still saturated until the integrator is unwound. Then the process output is so large the actuator saturates in the other direction.

Anti-windup technique One technique for avoiding integrator windup is back-calculation. The technique is displayed in figure 4.37 from [24]. The error signal e_s is the difference between the controller output and the actuator output. This is divided by a time constant T_t and is fed to the integrator input. This resets the integrator dynamically. When the actuator output is not saturated, e_s will be zero and the system will behave as it would have without the anti-windup.

Implementation When the motor was running too fast, the output would saturate. This would lead to large oscillations in the phase currents, and the motor would make a large bumping noise with high frequency, like someone was hitting it with a hammer. It was clear that this was caused

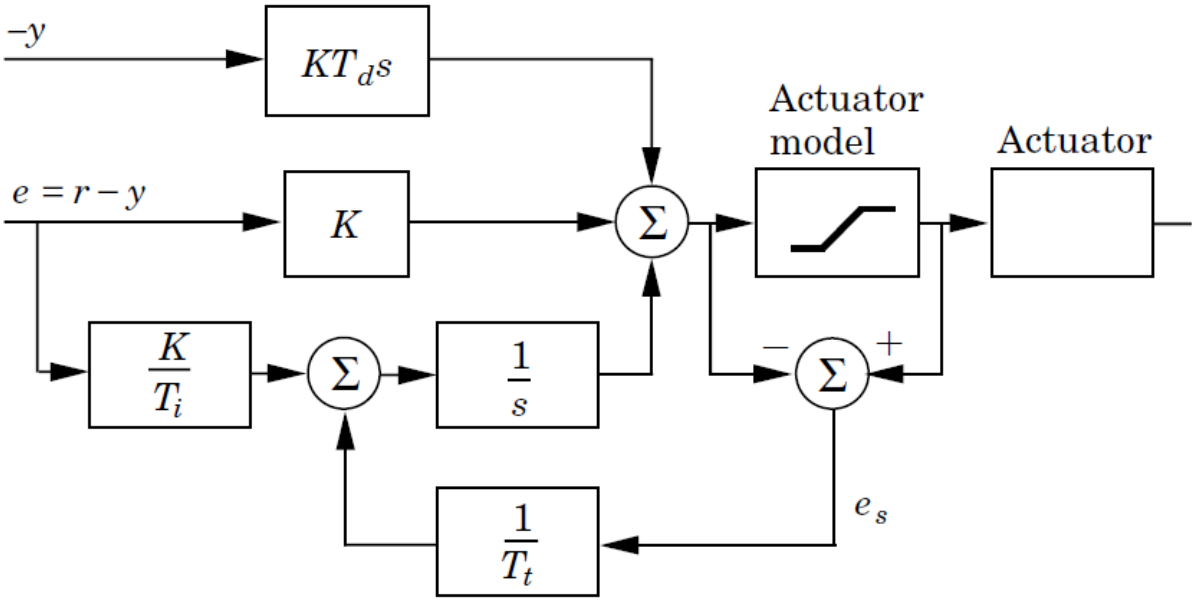


Figure 4.37: PID controller with anti-windup technique using back-calculation.

by the integrator windup.

The saturation had to be moved to the dq plane because the control was applied in this plane. The output would saturate if

$$V_d^2 + V_q^2 > V_{max}^2$$

$$V_{max}^2 = \frac{a_{max}^2 V_{DC}^2}{2}$$

Then the saturated output would be

$$Scale = \sqrt{\frac{V_d^2 + V_q^2}{V_{max}^2}}$$

$$V_{d,sat} = Scale * V_d$$

$$V_{q,sat} = Scale * V_q$$

The difference between the saturated output and the controller output was multiplied by a gain and fed back into the integrator state for anti-windup. This effectively removed the high

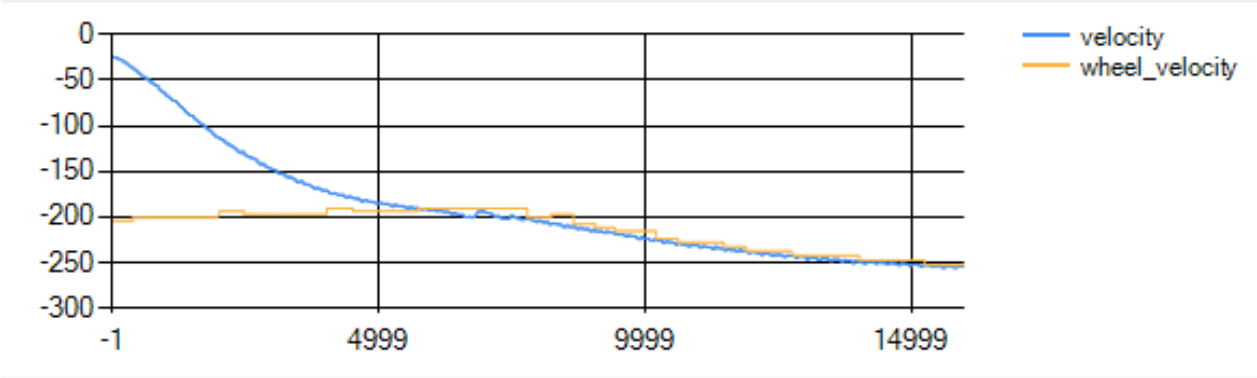


Figure 4.38: Graph showing the speed controller attaching the motor speed to the wheel speed.

frequency bumping noise at high speeds.

4.5.2 Speed controller testing

The speed controller was implemented using a PI controller. The proportional and integral gain was tuned manually. The desired attach time was under one second. This was achieved and the results can be seen in figure 4.38. In the graph only one out of four packets from the micro controller is plotted, such that the attach speed is about 7000 * 4 * 25 us = 0.7 seconds. The test was done with the wheel in free air.

Chapter 5

Competition and results

The race was at Ahoy racetrack in Rotterdam, Netherlands. The team was at the track for a week which included technical inspection, testing and three race days.

5.1 Technical inspection

No car was allowed to drive on the track before it had passed the technical inspection. Technical inspection was where they check that the cars were designed according to the rules. The inspection included checking brakes, turning radius, driver vision, electrical system and more. Passing the technical inspection was a goal in itself and many cars did not pass it at all. This section will describe the different difficulties that was encountered at technical inspection. Since the mechanical part is not relevant for this thesis, it will not be described here. For further information on the mechanical part see the overall master thesis of the car [10]. Picture 5.1 shows the UrbanConcept car at the technical inspection.

5.1.1 The Prototype car

The list below includes the most important problems encountered at technical inspection for the Prototype car.

Negative terminal of the battery connected to the car chassis The chassis of the car had the same potential as the negative terminal of the battery, which was not allowed. This was

Figure 5.1: The UrbanConcept car at the technical inspection.

not intended and the problem had to be located. The team discovered that the solar panels connections were short circuited to the chassis or monocoque of the car. Because the car was made of carbon fiber, a conductive material, the whole car was shorted to the solar panels. The solar panels was in turn connected to the negative terminal of the battery. The solution was to add isolation to the connection in form of electrical tape, which removed the short.

Thermal fuse The rules states that the Battery Management System (BMS) must automatically isolate the battery if over-temperature condition is detected. Since the BMS used in the system did not have over-temperature protection [9], a thermal fuse was used in series with the emergency stop buttons. The thermal fuse was taped to the battery pack. The cutoff temperature of the fuse had to be higher than the maximum allowed temperature of the battery cells. The team was able to document this after some search for the right datasheet of the battery cells and it was approved. However, the fuse only isolated the propulsion system if it blew. The accessory system and the solar panels would still be connected. Therefore this should not have been approved, and should be fixed for next year. The easiest solution to this would be to change to a BMS with over-temperature protection.

Fuse on positive terminal of the battery The fuse connected to the battery had to isolate the battery from the rest of the system, and had to be connected as close to the positive output of the battery as possible. The fuse that was installed isolated only the propulsion system, therefore a new fuse was installed on the positive wire from the battery.

5.1.2 The UrbanConcept car

The list below includes the most important problems encountered at technical inspection for the UrbanConcept car.

Hydrogen leakage During the inspection, hydrogen valves and tubes were checked for leakage. The maximum allowed hydrogen content in the air around the system was 100 ppm. The system was checked with the fuel cell running. The fuel cell stack releases some hydrogen

when it is running. Because of this, there was too much hydrogen in the back room of the car, making it impossible to measure if there was any leakage. When the stack have been running for a while, it releases less hydrogen because it gets warmer. Therefore the fuel cell was kept running for a while before checking the leakage again. After doing this, the test was performed successfully and approved.

Emergency stops The fuel cell came with an emergency stop button. The rules stated that two stop buttons was required, one accessible from the outside of the car and one accessible for the driver. With the assumption that the emergency stop button that was included with the fuel cell was of the Normally Connected (NC) type, a NC stop button was connected in series with this button. However, the stop button turned out to be of the Normally Open (NO) type, and the stop buttons did therefore not work unless both were pushed down at the same time. A NO stop button was borrowed from another team, and connected in parallel with the other stop button. Emergency stop buttons should generally be of the type NC because if one button is removed, the system will stop. With NO buttons, nothing happens when the button is removed. The fuel cell using NO emergency stop buttons was therefore less successful design.

5.2 Test

Before the race days, the track was open for testing. Testing before the race was important to check that everything worked and to check the fuel consumption for optimization.

5.2.1 The Prototype car

Two tests with the Prototype car were done. During the first test, 10 laps were completed and the results were

```
Motor energy: 148000 Joule
Solar energy: 38000 Joule
```

The rules stated that the total amount of solar panel energy withdrawn from the propulsion energy used could be no more than 20 percent of the propulsion energy. This meant with these

Figure 5.2: Picture of the fuel cell screen placed next to the driver.

values the result was

$$Result = \frac{16.117km}{\frac{148000J*0.8}{3600000J/kWh}} \approx 490km/kWh$$

This was about 120 km less than the Prototype car last year, so it was thought that some improvements could be made.

The second test did not finish 10 laps because the joystick fell off during the 8th lap. The results were however approximately the same for the propulsion energy and lower for the solar energy.

5.2.2 The UrbanConcept car

During the first two tests with the UrbanConcept car, the fuel cell shut off during the first lap. The thought was that the amount of hydrogen in the back was too high and triggered the hydrogen sensor, of the problems with the hydrogen leakage from the stack discovered at the technical inspection. There were no indications on why the fuel cell shut off. To better know why it turned off, a screen that could be connected to the fuel cell was placed by the driver as seen in picture 5.2. The cable to the screen had to be extended to place it there.

On the third test the car finished 9 laps with no problems. Checking the flow-meter, the hydrogen used was 140 normal liters. This was equal to a result of

$$Result = \frac{16.117km * \frac{9}{10}}{140dm^3} \approx 103.6km/m^3$$

This would have resulted in a second last place in the race, see appendix F.

5.3 Race

The cars were allowed four attempts to finish a race. The best result would be the one that counted. The car had to finish ten laps, equal to 16.117km, in 39 minutes. If the time limit was not held or less than ten laps finished, the result would not be valid. Each of the classes had three time slots of about three hours distributed across the three race days, where the teams could run their car(s). The race results is located in appendix F.

5.3.1 The Prototype car

The Prototype car was able to finish three races. Table 5.1 shows the results. The solar panels did not deliver enough power because of limited sun. The last day it was good conditions for the solar panels, but since the motor had to be moved to the UrbanConcept car (the reasons for this will be described later), only one run early on this day was made. If another run had been made later that day, the result would probably be better. The best run gave an 11th place with 482.5 km/kWh.

Motor energy	**Solar energy**	**Result**	**Comment**
142793 J	22552 J	482.5 km/kWh	
~145562 J	~0 J	398.6 km/kWh	Cloudy conditions.
136839 J	14613 J	474.7 km/kWh	Early, clear sky.

Table 5.1: Prototype race results.

Last year's result was 612.8 km/kWh. Since the team last year was able to get more power from the solar panels than the 20 percent limit, the energy used by the motor was

$$energy\ motor\ last\ year = \frac{1}{\frac{612.8km/kWh*0.8}{16.117km*3600000J/kWh}} \approx 118353Joule$$

This meant that the best result for the motor energy this year was approximately 16 percent higher than last year, even though the system used less energy in standby. Some of the reasons for this was thought to be the saturation technique used in the motor controller when the motor was at top speed. Because of much work done on the UrbanConcept car during the competition, it was no time to optimize the motor controller any further.

5.3.2 The UrbanConcept car

Table 5.2 shows the results of the UrbanConcept car. The car was not able to finish any of the races and get a valid result. The problems was due to the motor and fuel cell not working properly.

Result	**Comment**
Did not finish.	Fuse blew on the step up converters and one of the motor was destroyed.
Did not start.	Motor controllers not working properly and relay on power measurement board was always connected.
Did not finish.	Fuel cell stopped working on the 8th lap. *FC voltage low.*

Table 5.2: UrbanConcept race results.

Attempt 1 On the first race one of the motors was destroyed. The motor was unable to rotate after this incident, indicating that the bearings inside the motor were broken. The reason was unknown, but one theory was that the tension on the belt connecting the motor gear and the wheel gear was too high. The motor was rated for a certain load on the shaft [16]. The motor controller was also configured to deliver 33 percent more current to the motors to get more torque and higher acceleration, as this was requested by the driver. This could also have contributed to the engine breakdown. In addition to the breakdown, the fuses on the step up converters blew, which were rated for 20A. The motor was not possible to fix and the team could not get hold of another motor in time for the next race attempt. Therefore, the motor from the Prototype car

was used as a replacement. This meant moving the motor back and forth between the Prototype car and the UrbanConcept car, which also involved disconnection and connection of most of the motor controller electronics.

Attempt 2 On the second attempt the motor controllers did not work properly when testing them right before the race. The relay on the power measurement board had also been broken and was fixated in closed position. This, combined with a shortage of time to check what was wrong with the motor controller, made the car unable to start a second attempt. The power measurement board was switched with a replacement board. Why the motor controller was not working remained unknown, as it worked later that day when testing with a battery.

Attempt 3 On the third and last attempt the car was just able to start the race in time. There were a lot of problems with the motor controller despite the fact that it worked perfectly fine when tested with a battery the previous day. The motor controller system had been disconnected and moved to replace the motor. This may have destroyed some of the components on the boards. To fix this the motor controller from the Prototype car was used. The car ran well until the fuel cell stopped working in the 8th lap. The message on the fuel cell screen was *FC voltage low*. This denoted that the voltage from the fuel cell was too low for the fuel cell controller to work properly, and it would therefore shut the system off. It was speculated that this also was the reason the fuel cell shut off during the first test runs and not the hydrogen sensor tripping. The problem could be with the fuel cell stack, as well as the team operating the system in a wrong way.

Chapter 6

Discussion

The systems designed were tested and found to be working. However, there were some problems that were encountered. The more specific results of the designs have already been discussed to some degree in the previous chapters. This chapter will serve as a summary, and in addition discuss some of the more general results.

6.1 Motor controller

Hardware and software for the motor controller were made during this project. The motor controller was not a part of the original tasks for this thesis, but as someone was needed for the task, it was decided that it should be done by the undersigned. This resulted in less time for other tasks. The undersigned had less experience in fields regarding motor control, therefore insights were gained in the ongoing process of designing and testing.

The design was based on the prototype boards designed last semester. The microcontroller chosen was not optimal for the task, as it had limited calculating power. Other solutions may have been better, but lack of time made searching for other options not feasible. However, the microcontroller vendor supported a good software framework with libraries that made most of the programming much easier.

The prototype boards were tested and the faults were found. This process was somewhat delayed because of waiting for the soldering company to assemble the boards. The assembly of the boards were outsourced because of the difficulty of soldering the components by hand.

An attempt to solder the boards at the university could have been made, provided that the time used by the company to assemble the board had been known. Most of the new boards were soldered at the university with the same components, but in doing this manufactured boards were used, which among other things have solder mask that make them less difficult to solder. It is generally important to test prototype boards as early as possible since these often require some modification to work as expected.

When testing the motor controller with the car, problems with the controller was found and fixed. A logging program was very useful in detecting the problems and find the cause. Logging was only possible at low speeds since a laptop had to be connected to the system. This meant running beside the car when logging. Hence, it was not possible to carry out logging at speeds higher than a person could run. Some problems occurred only when driving at high speeds. Testing the car with an adjustable load without driving the car would have helped in solving these problems easier. This could have been done in a lab, but eventually the motor had to be installed in the car. A lot of teams in Rotterdam had a trolley for the car with a load attached to the wheel, allowing for easy testing of the car with appropriate load.

The results from the competition showed that the propulsion system was using more energy than the last years propulsion system with the same car, even though the system was using less energy in standby. This was probably due to the motor controller not working optimally. The saturation technique when the motor was at high speed was not delivering smooth sinus curve currents to the motor, indicating that the controller was not working well at high speeds. The last years team was using an industrial made controller from SINTEF [19]. It would be hard to match the efficiency of this controller. The main reason for switching out this controller was to lower the standby power.

There were a lot of problems with the motor controller in the UrbanConcept car at the race track. In the last race, the motor controller from the Prototype car was used as a replacement since one of the motor controllers failed. The propulsion system in both cars was the same except for the energy source, but this should not have made any difference. The main reasons for the problems was thought to be that the electronics were disconnected and moved around a lot, in addition to more people working with other systems in the back of the car where the electronics for the propulsion system were. This exposed the electronics, which may have caused

accidental damage. The propulsion system should have been built with more protection, like for example enclosures for the circuit boards.

6.2 Communication, data acquisition and logging system

A communication, data acquisition and logging system was made to be able to monitor the car in real time.

The power measurement board made was measuring the voltage of the battery, and the current to the motor and from the solar panels. Problem with the measuring of the battery emerged when the motor was running, as the measurement dropped several volts. The voltage drop became smaller when connecting the shielding of the motor cable to digital ground. Overflow and round-off errors in the motor current measurements from changing the code before the races made the logged data from these measurement less useful.

A 3G module made last year was debugged and a new module was designed. Two prototype boards were milled out at the university. One to interface the modem with a computer, and one to verify the new design. Milling out prototype boards require more work than when ordering boards from a manufacturer. In addition, this process also makes the boards more prone to errors. When the team ordered from a manufacturer, the expected delivery time was between 1 and 2 weeks. Deciding whether or not to produce prototype boards or to order from manufacturer depended on the time available and complexity of the design. It may have been beneficial to order the second prototype board from a manufacturer when looking at the time available and the work that was done making the prototype boards.

It was not much time to make a program for displaying the data to the user. The most important data were displayed in graphs and the entire packets as raw text in a terminal window. There is a huge potential for visualizing data, and storing data to go back in time and analyse. The GPS position was not used this year, but can be used for viewing the position of the car. The most important data from the car was the speed of the car and the driver reference (throttle), along with the motor status when testing the motor controller. The current used by the motor could have been useful, but due to errors in the data during the races, the values were of less importance. The current from the solar panels turned out to be of no use at all.

6.3 Fuel cell

The fuel cell system shutting down was one of the main reasons the UrbanConcept car did not finish a single race, besides the problems with the motor controller and the engine breakdown. The possibilities for testing in Norway was limited both because many components arrived later than expected and because no vendor would fill up the hydrogen tank enough to drive more than 4 minutes. When the fuel cell did not work in Rotterdam, the team did not know how to fix it. This was also because none of the team members understood the electrical system of the fuel cell completely. Assigning a person responsible for the electrical system of the fuel cell should have been done early in the process.

6.4 Future work

To test and optimize the motor controller, a trolley with a load should be made for at least one of the cars, preferably the Prototype car. The load should be adjustable or corresponding to the load on the wheel when driving the car. The inverter was also using a lot of energy and is over-dimensioned for the propulsion system in the car. This can be exchanged with a more energy efficient inverter.

The data acquisition system can be improved by fixing the power measurements and adding more sensors to the system. The logging system can include a better visualization of the data and provide storage for saving data and browsing of old data. GPS can be used to display the position of the car in real time or to make 3D graphs by using the data from other sensors in the z-axis and position of the car in the xy-plane.

The fuel cell must be tested and the cause of the problem that occurred during the race must be found. For testing longer than 4 minutes, the team must find a vendor that can fill the hydrogen tank to the rated pressure. The team should assign one person to have the sole responsibility for the electric system of the fuel cells.

Chapter 7

Conclusion

Throughout this project hardware and software have been designed for the DNV GL Fuel Fighter vehicles, two energy efficient cars that participated in the Shell Eco-marathon late May. A motor controller has been made to control the permanent magnet AC motors responsible for the propulsion of the cars. A communication, data acquisition and logging system has been made for real time monitoring of the car.

The communication, data acquisition and logging system was based on the work in the specialization project done last semester, where a prototype for a 3G module was made. This board however did not work and had to be redesigned. Two boards were milled out to configure the modem and to verify the new design before manufactured boards were ordered.

A new board was designed for measuring the power consumption of the motor and measuring the power delivered from the solar panels. This board also included a way of shutting off the motor system both from the emergency stop buttons and from the steering wheel. This function proved useful under the race, as shutting off the motor system when the motor was not in use saved energy. Problems with the battery voltage measurement appeared when running the motor. The source of the problem was not found and possible errors have been discussed. Lack of testing and minor bugs in the code made the power measurements less useful for the logging system. Power measurements can be of great value in the future for optimizing motor efficiency, if the above mentioned problems are solved.

A module with a GPS was made similar to one used by the last year's team. The GPS was utilized to send the position of the car along with the time. The original idea was to display the

position to the user on a map, but this was not done because of other more important tasks. This can be a task for next year's team.

A logging system was made to distribute the data and visualize it. The system consisted of a server which retrieved the data and broadcast it to all users connected to the server. A minimal program was made to show the most important data in graphs as well as the raw packets as text in a terminal. The terminal was a useful tool for checking that the packets were correctly formatted and for storing the data. The logging system proved that the overall system including the data acquisition and communication worked. The overall system forms a basis for other teams to develop further, especially when it comes to visualization and storing data.

Hardware and software for a motor control system were made. The system was based on the simulations and prototypes made in the last semester, as well as work done the previous year. Two prototype boards were tested and revisions were made. Among the changes was a way of logging data from the motor controller. Software based on the simulations were written. Optimization to achieve hard real time requirements were implemented by using fixed point numbers and look-up tables. A logging program was made on a computer for retrieving and displaying data from the controller. With the logging program, faults in the design were discovered. New simulation models were made and later implemented. The new controller design experienced problems when running at high speeds, which were due to output saturating and integrator windup. These problems were solved by having the saturation at another place in the controller and implementing an anti-windup technique with back calculation. Another part of the software, a speed controller, was implemented for connecting the wheel and the motor when running at different speeds. This was necessary because the wheel and the motor was connected with a one way gear.

During the competition the Prototype car finished three races and ended up in 11th place, with a score of 482.5 km/kWh, about 130 km less than last year. The reason for the result being worse than last year was thought to be that the saturation with the anti-windup technique was making the motor controller less optimal at high speeds. Another reason was that there were too little sun during the races for the solar panels to deliver enough power.

The UrbanConcept did not finish any of the races. One of the motors broke down during a race. The motor from the Prototype car was used as a replacement. During another race the

fuel cell shut down. Lack of expertise on this part of the car and lack of testing done before the competition, was the main reasons that the problems were not discovered and fixed.

The systems designed in the this project have been thoroughly tested. Except for some minor obstacles, the systems are well functioning. They form a basic platform for the next years teams to develop further. For a more detailed description of future work, see discussion chapter 6.

This project has been educational both when it comes to making a product and in working in a team. As previous years, the team had to work day and night the last weeks before the competition. The author has learned that careful planning and structured work is required to meet absolute deadlines.

Appendix A

Acronyms

AC Alternating Current

CAN Controller Area Network

DC Direct Current

FPGA Field Programmable Gate Array

GPIO General-Purpose Input/Output

GPS Global Positioning System

GSM Global System for Mobile Communications

GUI Graphical User Interface

IC Integrated Circuit

HTTP HyperText Transfer Protocol

IGBT insulated-gate bipolar transistor

IMU Inertial Measurement Unit

kbps kilobit per second

MCU Microcontroller Unit

PCB Printed Circuit Board

ppm parts per million

SD Secure Digital

SEM Shell Eco-marathon

SPI Serial Peripheral Interface

USB Universal Serial Bus

Appendix B

Motor controller calculations

B.1 Controller equations

The equations are written out to bridge the gap to code. An asterisk (*) is used to denote a multiplication, for better readability.

B.1.1 I-controller

$$duq = Nu * i\_ref\_q + (-\mathbf{K}) * (\mathbf{x\_hat} - \mathbf{Nx} * i\_ref\_q) \tag{B.1}$$

$$duq = Nu * i\_ref\_q - (\begin{bmatrix} K\_1 & K\_2 \end{bmatrix} * (\begin{bmatrix} x\_hat\_1 \\ x\_hat\_2 \end{bmatrix} - \begin{bmatrix} Nx\_1 \\ Nx\_2 \end{bmatrix} * i\_ref\_q) \tag{B.2}$$

$$\begin{aligned} duq = Nu * i\_ref\_q - K\_1 * (x\_hat\_1 - Nx\_1 * i\_ref\_q) \\ - K\_2 * (x\_hat\_2 - Nx\_2 * i\_ref\_q) \end{aligned} \tag{B.3}$$

$$dud = (-\mathbf{K}) * \mathbf{x\_hat} \tag{B.4}$$

$$dud = -\begin{bmatrix} K\_1 & K\_2 \end{bmatrix} * \begin{bmatrix} x\_hat\_1 \\ x\_hat\_2 \end{bmatrix} \tag{B.5}$$

$$dud = -K\_1 * x\_hat\_1 - K\_2 * x\_hat\_2 \tag{B.6}$$

Where K_1, K_2, Nx_1, Nx_2 and Nu are constants.

The observers are the same with different input signals.

Observer1

$$\mathbf{x\_hat} = z^{-1} * (\mathbf{Bd} * dud + \mathbf{Ad} * \mathbf{x\_hat} + \mathbf{L\_obs} * (i\_m\_d - \mathbf{Cd} * \mathbf{x\_hat})) \tag{B.7}$$

Observer2

$$\mathbf{x\_hat} = z^{-1} * (\mathbf{Bd} * duq + \mathbf{Ad} * \mathbf{x\_hat} + \mathbf{L\_obs} * (i\_m\_q - \mathbf{Cd} * \mathbf{x\_hat})) \tag{B.8}$$

Where z^{-1} denotes a unit delay, which holds and delays its input by the sample time. If we call the variable going into the $\frac{1}{z}$ block for x_hat_last, the equations will be

Observer1

$$\mathbf{x\_hat} = \mathbf{x\_hat\_last} \tag{B.9}$$

$$\mathbf{x\_hat\_last} = \mathbf{Bd} * dud + \mathbf{Ad} * \mathbf{x\_hat} + \mathbf{L\_obs} * (i\_m\_d - \mathbf{Cd} * \mathbf{x\_hat}) \tag{B.10}$$

$$\begin{bmatrix} x\_hat\_last\_1 \\ x\_hat\_last\_2 \end{bmatrix} = \begin{bmatrix} Bd\_1 \\ Bd\_2 \end{bmatrix} * dud + \begin{bmatrix} Ad\_11 & Ad\_12 \\ Ad\_21 & Ad\_22 \end{bmatrix} * \begin{bmatrix} x\_hat\_1 \\ x\_hat\_2 \end{bmatrix} + \begin{bmatrix} L\_obs\_1 \\ L\_obs\_2 \end{bmatrix} * (i\_m\_d - \begin{bmatrix} Cd\_1 & Cd\_2 \end{bmatrix} * \begin{bmatrix} x\_hat\_1 \\ x\_hat\_2 \end{bmatrix} \tag{B.11}$$

$$x\_hat\_last\_1 = Bd\_1 * dud + Ad\_11 * x\_hat\_1 + Ad\_12 * x\_hat\_2 + L\_obs\_1 * (i\_m\_d - (Cd\_1 * x\_hat\_1 + Cd\_2 * x\_hat\_2)) \tag{B.12}$$

$$x\_hat\_last\_2 = Bd\_2 * dud + Ad\_21 * x\_hat\_1 + Ad\_22 * x\_hat\_2 + L\_obs\_2 * (i\_m\_d - (Cd\_1 * x\_hat\_1 + Cd\_2 * x\_hat\_2)) \tag{B.13}$$

Observer 2

$$\mathbf{x\_hat} = \mathbf{x\_hat\_last} \tag{B.14}$$

$$\mathbf{x\_hat\_last} = \mathbf{Bd} * duq + \mathbf{Ad} * \mathbf{x\_hat} + \mathbf{L\_obs} * (i\_m\_q - \mathbf{Cd} * \mathbf{x\_hat}) \tag{B.15}$$

$$\begin{bmatrix} x\_hat\_last\_1 \\ x\_hat\_last\_2 \end{bmatrix} = \begin{bmatrix} Bd\_1 \\ Bd\_2 \end{bmatrix} * duq + \begin{bmatrix} Ad\_11 & Ad\_12 \\ Ad\_21 & Ad\_22 \end{bmatrix} * \begin{bmatrix} x\_hat\_1 \\ x\_hat\_2 \end{bmatrix} + \begin{bmatrix} L\_obs\_1 \\ L\_obs\_2 \end{bmatrix} * (i\_m\_q - \begin{bmatrix} Cd\_1 & Cd\_2 \end{bmatrix} * \begin{bmatrix} x\_hat\_1 \\ x\_hat\_2 \end{bmatrix} \tag{B.16}$$

$$x\_hat\_last\_1 = Bd\_1 * duq + Ad\_11 * x\_hat\_1 + Ad\_12 * x\_hat\_2 + L\_obs\_1 * (i\_m\_q - (Cd\_1 * x\_hat\_1 + Cd\_2 * x\_hat\_2)) \tag{B.17}$$

$$x\_hat\_last\_2 = Bd\_2 * duq + Ad\_21 * x\_hat\_1 + Ad\_22 * x\_hat\_2 + L\_obs\_2 * (i\_m\_q - (Cd\_1 * x\_hat\_1 + Cd\_2 * x\_hat\_2)) \tag{B.18}$$

Where Ad_11, Ad_12, Ad_21, Ad_22, Bd_1, Bd_1, Cd_1, Cd_1, L_obs_1 and L_obs_2 are constants.

The output of the I-controller is a vector with to values which we call $v\_VSC\_dq.d$ and $v\_VSC\_dq.q$. The final equations for the output are

$$v\_VSC\_dq.d = dud - L * x\_hat\_1 * w\_m \tag{B.19}$$

$$v\_VSC\_dq.q = duq + w\_m * (phi\_hat + L * x\_hat\_1) \tag{B.20}$$

B.1.2 I-controller reduced

$$dud = -\mathbf{K} * \begin{bmatrix} \frac{i\_m\_d}{z-1} \\ i\_m\_d \end{bmatrix} = -K\_1 * xi\_d - K\_2 * i\_m\_d \tag{B.21}$$

$$xi\_d = xi\_d + i\_m\_d \tag{B.22}$$

$$duq = -\mathbf{K} * \begin{bmatrix} \frac{i\_m\_q - i\_ref\_q}{z-1} \\ i\_m\_q \end{bmatrix} = -K\_1 * xi\_q - K\_2 * i\_m\_q \tag{B.23}$$

$$xi\_q = xi\_q + i\_m\_q - i\_ref\_q \tag{B.24}$$

$$v\_VSC\_dq.d = dud - L * i\_m\_q * w\_m \tag{B.25}$$

$$v\_VSC\_dq.q = duq + w\_m * (phi\_hat + L * i\_m\_d) \tag{B.26}$$

B.1.3 Transformations

The equations inside the abc–>dq block is

$$\mathbf{Tdq} = \begin{bmatrix} cos(theta\_m) & sin(theta\_m) \\ -sin(theta\_m) & cos(theta\_m) \end{bmatrix} \tag{B.27}$$

$$i\_m\_dq = \mathbf{Tdq} * \mathbf{Talbe} * \mathbf{i\_m\_abc} \tag{B.28}$$

Where

$$\mathbf{Talbe} = \frac{1}{3} * \begin{bmatrix} 2 & -1 & -1 \\ 0 & \sqrt{3} & -\sqrt{3} \end{bmatrix} \tag{B.29}$$

The equations inside the dq–>abc block is

$$\mathbf{Tdq\_inv} = \begin{bmatrix} cos(theta\_m) & -sin(theta\_m) \\ sin(theta\_m) & cos(theta\_m) \end{bmatrix} \tag{B.30}$$

$$v\_VSC\_abc = \mathbf{Talbe\_inv} * \mathbf{Tdq\_inv} * \mathbf{v\_VSC\_dq} \tag{B.31}$$

Where

$$\mathbf{Talbe\_inv} = \frac{1}{2} * \begin{bmatrix} 2 & 0 \\ -1 & \sqrt{3} \\ -1 & -\sqrt{3} \end{bmatrix} \tag{B.32}$$

The transformations are implemented on the MCU by writing out the matrix multiplications.

$$\begin{aligned} i\_m\_dq.d = &(Tdq_{11} * Talbe_{11} + Tdq_{12} * Talbe_{21}) * i\_m\_abc.a \\ &+ (Tdq_{11} * Talbe_{12} + Tdq_{12} * Talbe_{22}) * i\_m\_abc.b \\ &+ (Tdq_{11} * Talbe_{13} + Tdq_{12} * Talbe_{23}) * i\_m\_abc.c \end{aligned} \quad \text{(B.33)}$$

$$\begin{aligned} i\_m\_dq.q = &(Tdq_{21} * Talbe_{11} + Tdq_{22} * Talbe_{21}) * i\_m\_abc.a \\ &+ (Tdq_{21} * Talbe_{12} + Tdq_{22} * Talbe_{22}) * i\_m\_abc.b \\ &+ (Tdq_{21} * Talbe_{13} + Tdq_{22} * Talbe_{23}) * i\_m\_abc.c \end{aligned} \quad \text{(B.34)}$$

$$\begin{aligned} v\_VSC\_abc.a = &(Talbe\_inv_{11} * Tdq\_inv_{11} + Talbe\_inv_{12} * Tdq\_inv_{21}) * v\_VSC\_dq.d \\ &+ (Talbe\_inv_{11} * Tdq\_inv_{12} + Talbe\_inv_{12} * Tdq\_inv_{22}) * v\_VSC\_dq.q \end{aligned} \quad \text{(B.35)}$$

$$\begin{aligned} v\_VSC\_abc.b = &(Talbe\_inv_{21} * Tdq\_inv_{11} + Talbe\_inv_{22} * Tdq\_inv_{21}) * v\_VSC\_dq.d \\ &+ (Talbe\_inv_{21} * Tdq\_inv_{12} + Talbe\_inv_{22} * Tdq\_inv_{22}) * v\_VSC\_dq.q \end{aligned} \quad \text{(B.36)}$$

$$\begin{aligned} v\_VSC\_abc.c = &(Talbe\_inv_{31} * Tdq\_inv_{11} + Talbe\_inv_{32} * Tdq\_inv_{21}) * v\_VSC\_dq.d \\ &+ (Talbe\_inv_{31} * Tdq\_inv_{12} + Talbe\_inv_{32} * Tdq\_inv_{22}) * v\_VSC\_dq.q \end{aligned} \quad \text{(B.37)}$$

Where $i\_m\_dq.d$ indicate the d component of the vector, $v\_VSC\_abc.a$ indicate the a component of the vector and so on.

B.2 Scaling

Converting from float to Q23 format is called $f\_to\_q()$ for short.

B.2.1 PWM

- PWM frequency: 20 KHz.
- CPU frequency: 96 MHz
- PWM values: 0 to $\frac{CPU_freq}{PWM_freq} = 4800$.
- controller output values: -1 to 1.

$PWM\_out = \frac{controller\_output+1}{2} * 4800$

$PWM\_out = ((controller\_output + f\_to\_q(1.0).value) * f\_to\_q(\frac{4800}{2} * 2^{-23}).value) >> 23$

B.2.2 Current measurements

- Measuring range: -50A to 50A.
- ADC input values: 0 to $2^{16} - 1$.

$input = -(\frac{ADC\_input - \frac{2^{16}-1}{2}}{2^{16}-1} * 100)$

$input = \frac{100}{2} - ADC\_input * \frac{100}{2^{16}-1}$

$input\_q23 = f\_to\_q(\frac{100}{2}) - (ADC\_input << 23 * f\_to\_q(\frac{100}{2^{16}-1})) >> 23$

$input\_q23 = f\_to\_q(\frac{100}{2}) - ADC\_input * f\_to\_q(\frac{100}{2^{16}-1})$

B.2.3 Voltage measurement

- Measuring range: 0 to V_{max}.
- ADC input values: 0 to $2^{16} - 1$.

$input = \frac{ADC\_input}{2^{16}-1} * V_{max}$

$input\_q23 = (ADC\_input << 23 * f\_to\_q(\frac{V_{max}}{2^{16}-1})) >> 23$

$input\_q23 = ADC\_input * f\_to\_q(\frac{V_{max}}{2^{16}-1})$

B.2.4 Motor speed

- Measuring range: -1000 rps to 1000 rps.
- RDC input values: -2048 to 2047.

$$input = \frac{RDC\_input}{2048} * 1000 * 2\pi$$
$$input\_q23 = (RDC\_input << 23 * f\_to\_q(\frac{1000*2\pi}{2048})) >> 23$$
$$input\_q23 = RDC\_input * f\_to\_q(\frac{1000*2\pi}{2048})$$

Appendix C

Operational amplifier resistor calculations

C.1 Motor controller

C.1.1 Phase current measurements

- Input current range: $[-35\sqrt{2}mA, -35\sqrt{2}mA]$.
- $R_{shunt} = 10\Omega$
- $U_{min} = -0.5V$
- $U_{max} = 0.5V$
- $U_{ref} = 3.3V$

C.1.2 Input DC voltage measurement

C.2 Power measure module

$(U_{ref} - U_{out})\frac{R_2}{R_1+R_2} + U_{out} = U_{in}\frac{R_4}{R_3+R_4}$

$U_{in} = U_{max}$, $U_{out} = U_{ref}$:

$U_{ref} = U_{max}\frac{R_4}{R_3+R_4}$

$\frac{R_3+R_4}{R_4} = \frac{U_{max}}{U_{ref}}$

$\frac{R_3}{R_4} = \frac{U_{max}}{U_{ref}} - 1$

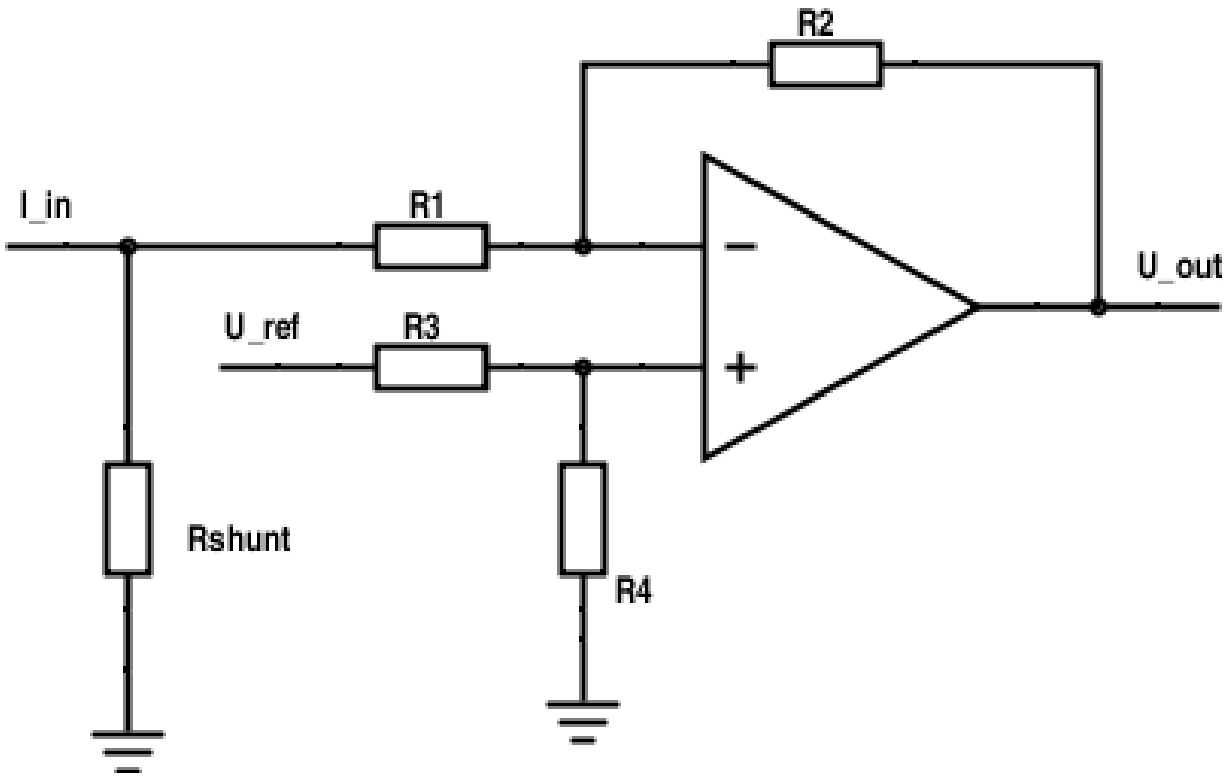


Figure C.1: Operational amplifier for inverter phase currents measurements

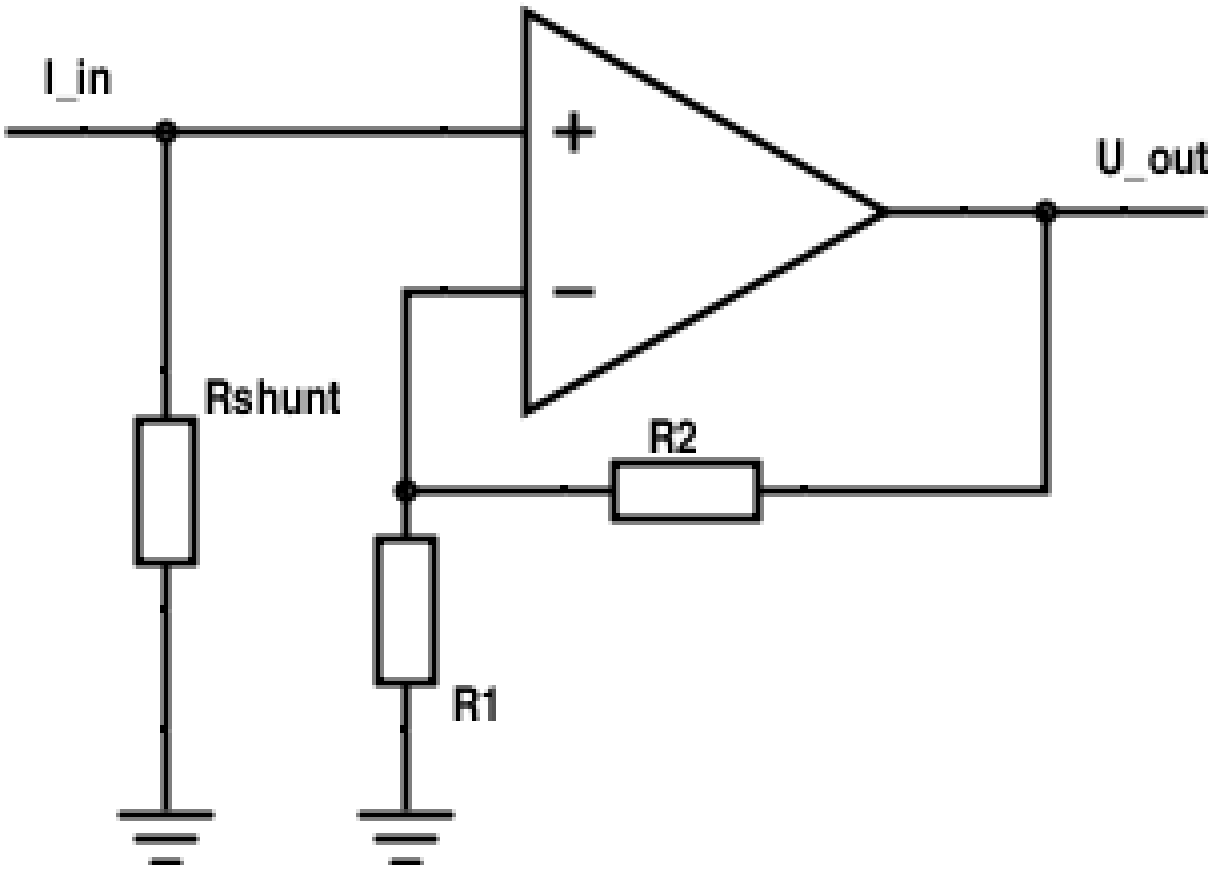


Figure C.2: Operational amplifier for inverter DC voltage measurement

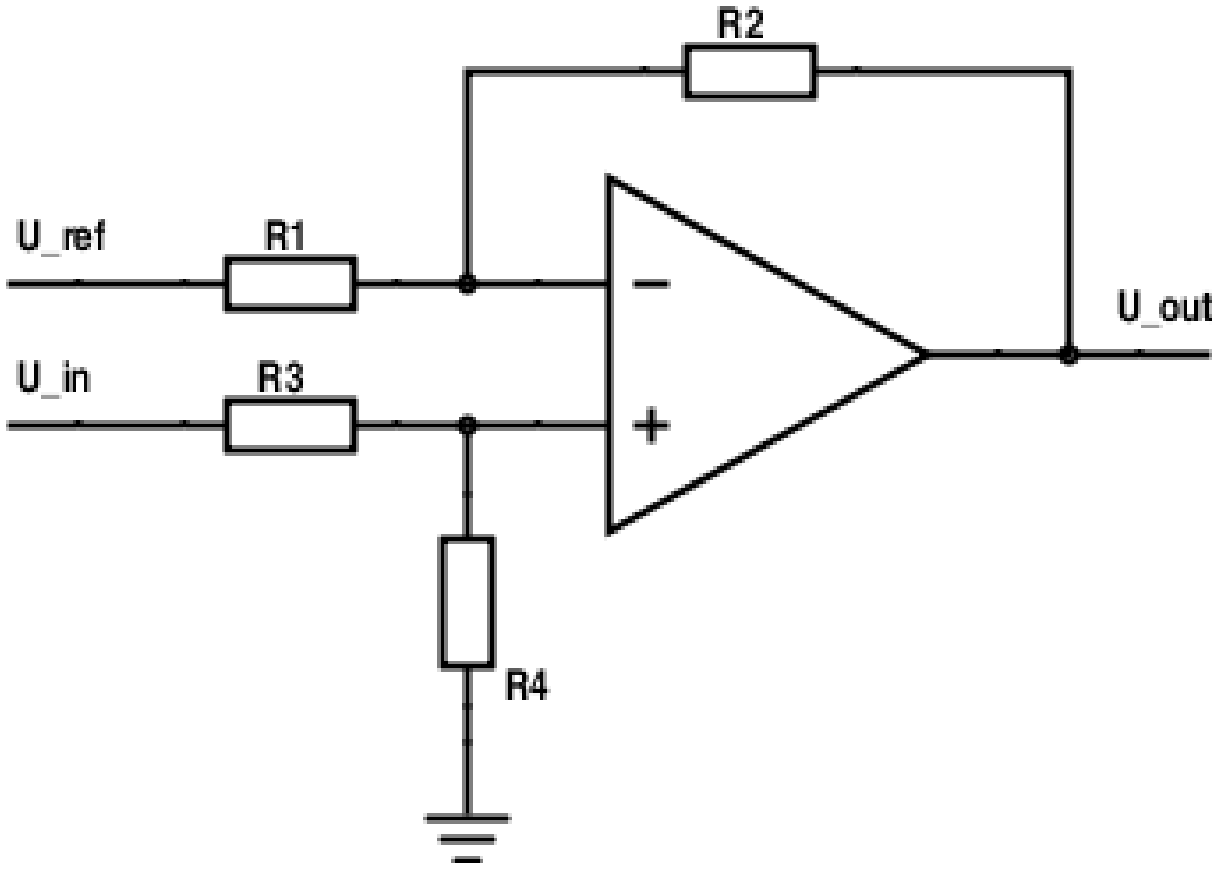


Figure C.3: Operational amplifier for motor and solar panels current measurements

$U_{in} = U_{min}, U_{out} = 0V$:

$U_{ref} \frac{R_2}{R_1+R_2} = U_{min} \frac{R_4}{R_3+R_4}$

$U_{ref} \frac{R_2}{R_1+R_2} = U_{min} \frac{U_{ref}}{U_{max}}$

$\frac{R_2}{R_1+R_2} = \frac{U_{min}}{U_{max}}$

$\frac{R_1}{R_2} = \frac{U_{max}}{U_{min}} - 1$

C.2.1 Motor current measurements

- Measuring range: -20A to 20A.
- Sensitivity: 25mV/A.
- $U_{min} = 2V$
- $U_{max} = 3V$
- $U_{ref} = 2.5V$

$\frac{R_1}{R_2} = \frac{3V}{2V} - 1 = \frac{1}{2}$

$R_1 = 10k\Omega$

$R_2 = 20k\Omega$

$\frac{R_3}{R_4} = \frac{3V}{2.5V} - 1 = \frac{1}{5}$

$R_3 = 10k\Omega$

$R_4 = 50k\Omega$

C.2.2 Solar panels current measurements

- Measuring range: 0A to 5A.
- Sensitivity: 185mV/A.
- $U_{min} = 2.5V$
- $U_{max} = 3.425V$
- $U_{ref} = 2.5V$

$\frac{R_1}{R_2} = \frac{3.425V}{2.5V} - 1 = 0.37$

$R_1 = 3.7k\Omega$

$R_2 = 10k\Omega$

$\frac{R_3}{R_4} = \frac{3.425V}{2.5V} - 1 = 0.37$

$R_3 = 3.7k\Omega$

$R_4 = 10k\Omega$

C.2.3 Solar panels current measurements revised

Calculations after discovering that U_{min} was 2.3V. The resistor values are the closest that were found without changing the 10k resistor.

- Measuring range: 0A to 5A.
- Sensitivity: 185mV/A.
- $U_{min} = 2.3V$
- $U_{max} = 3.225V$
- $U_{ref} = 2.5V$

$\frac{R_1}{R_2} = \frac{3.225V}{2.3V} - 1 \approx 0.4$

$R_1 = 4.3k\Omega$

$R_2 = 10k\Omega$

$\frac{R_3}{R_4} = \frac{3.225V}{2.5V} - 1 = 0.29$

$R_3 = 3k\Omega$

$R_4 = 10k\Omega$

Appendix D

Code

The code is included with the files accompanying the thesis.

Appendix E

PCB schematics and layouts

E.1 3G board

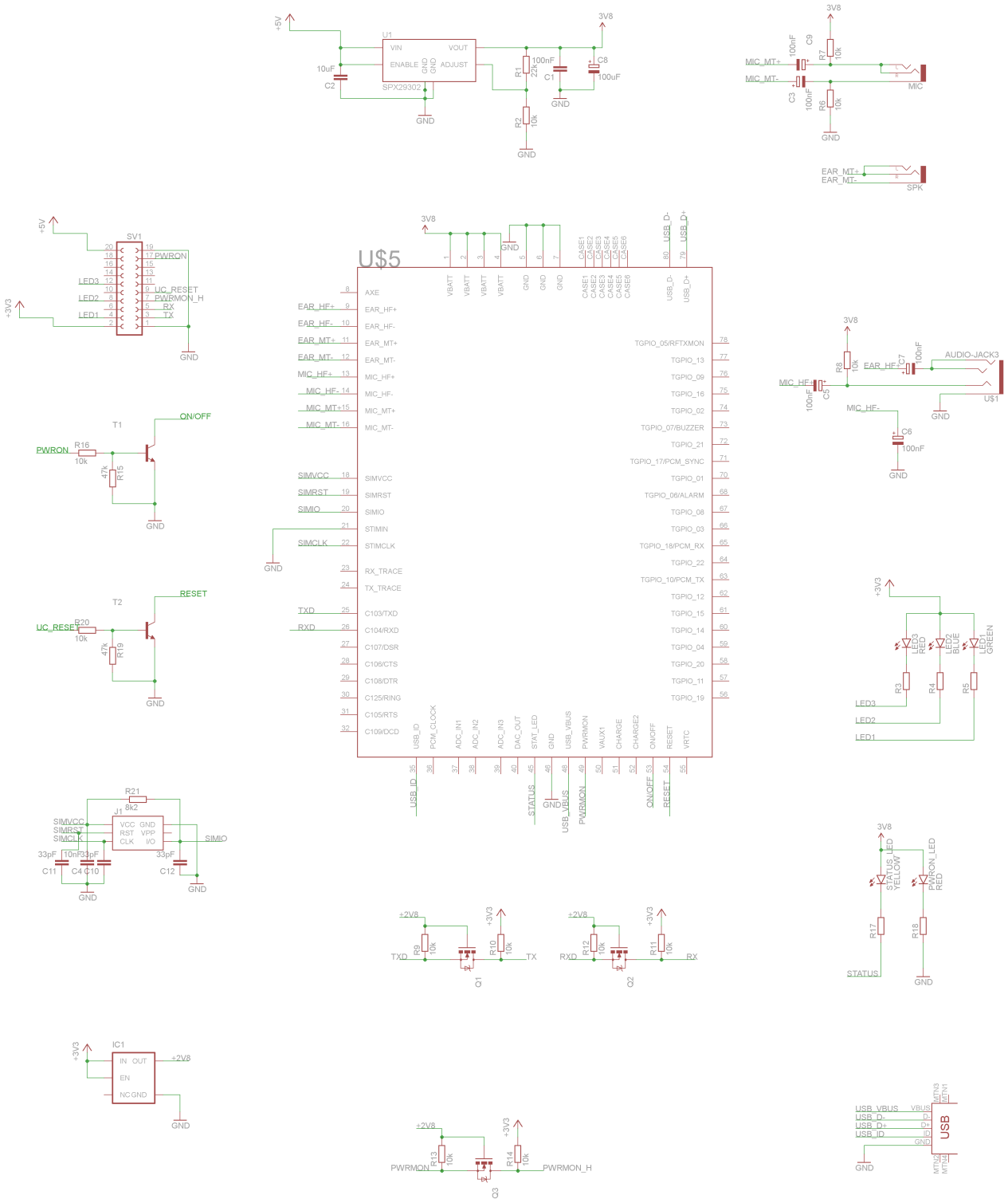


Figure E.1: 3G board schematic.

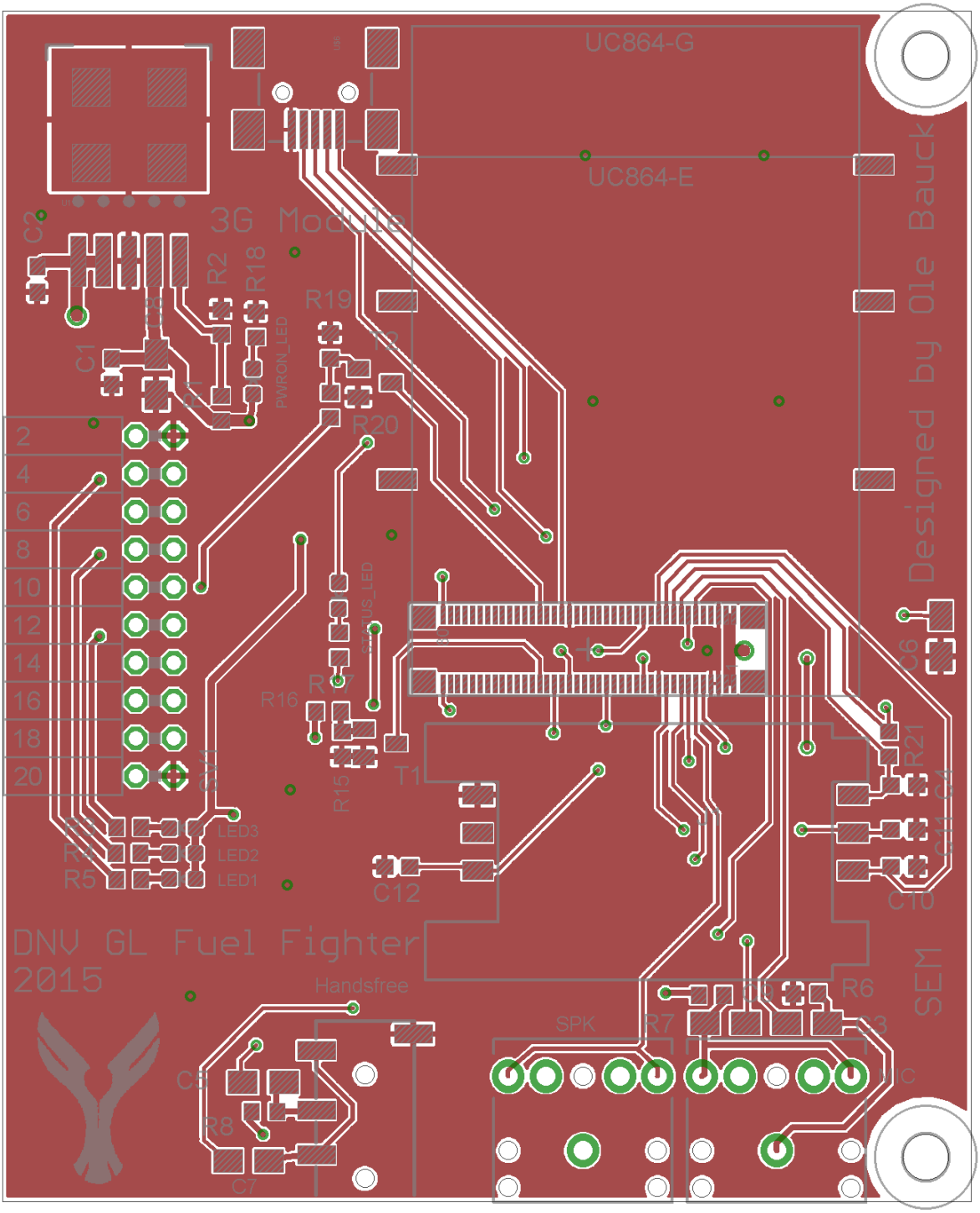


Figure E.2: 3G board layout top.

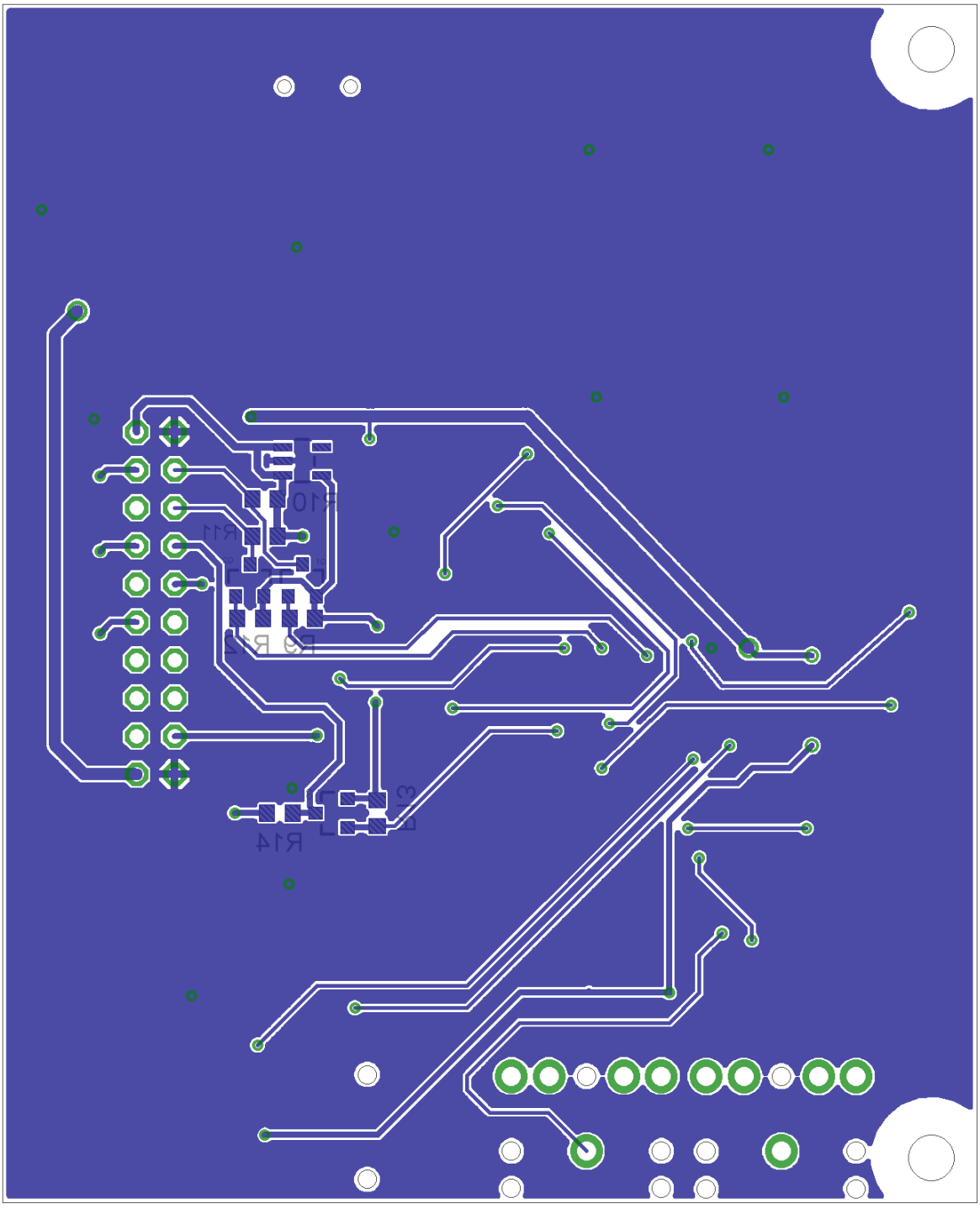


Figure E.3: 3G board layout bottom.

E.2 3G evaluation board

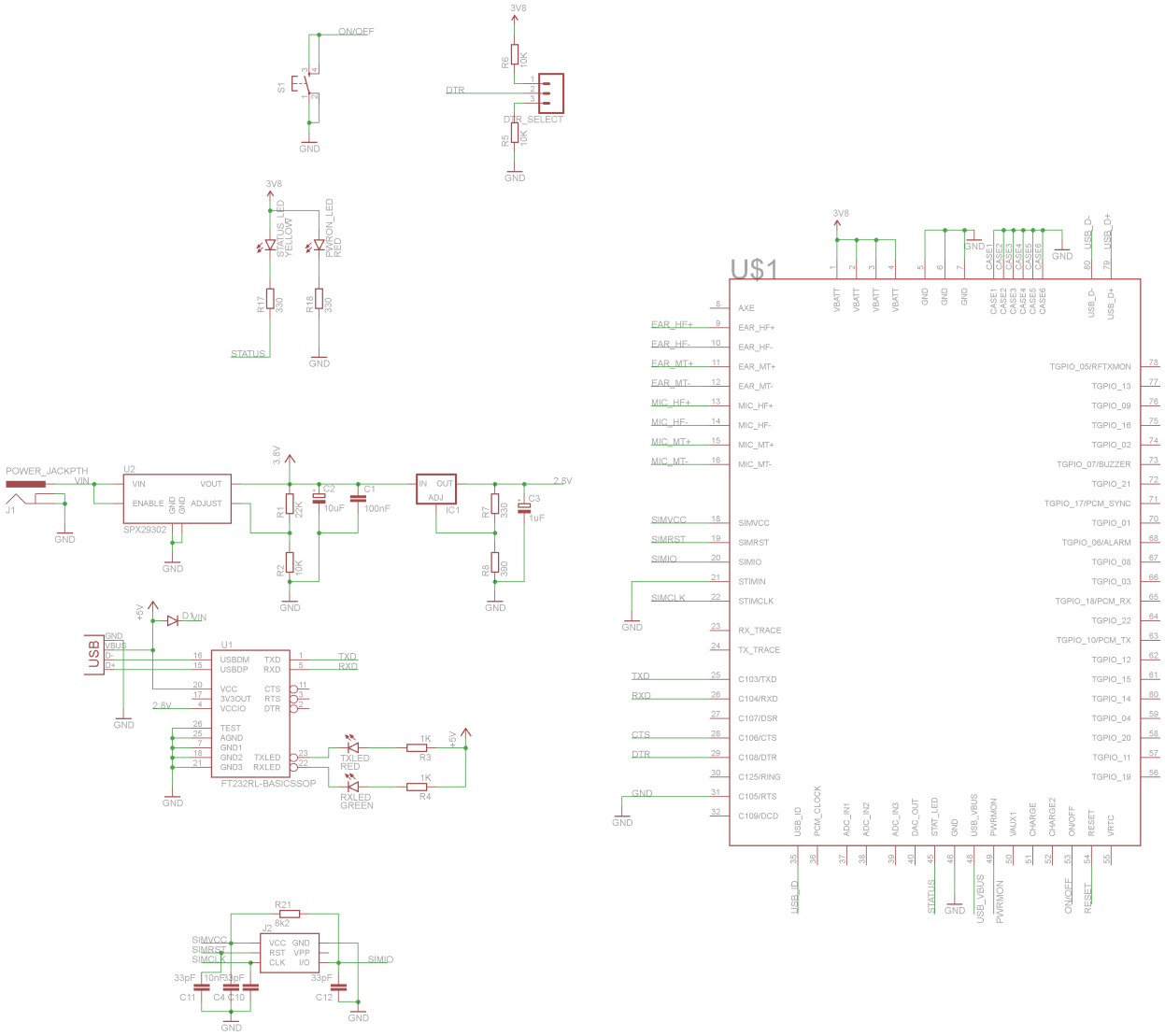


Figure E.4: 3G evaluation board schematic.

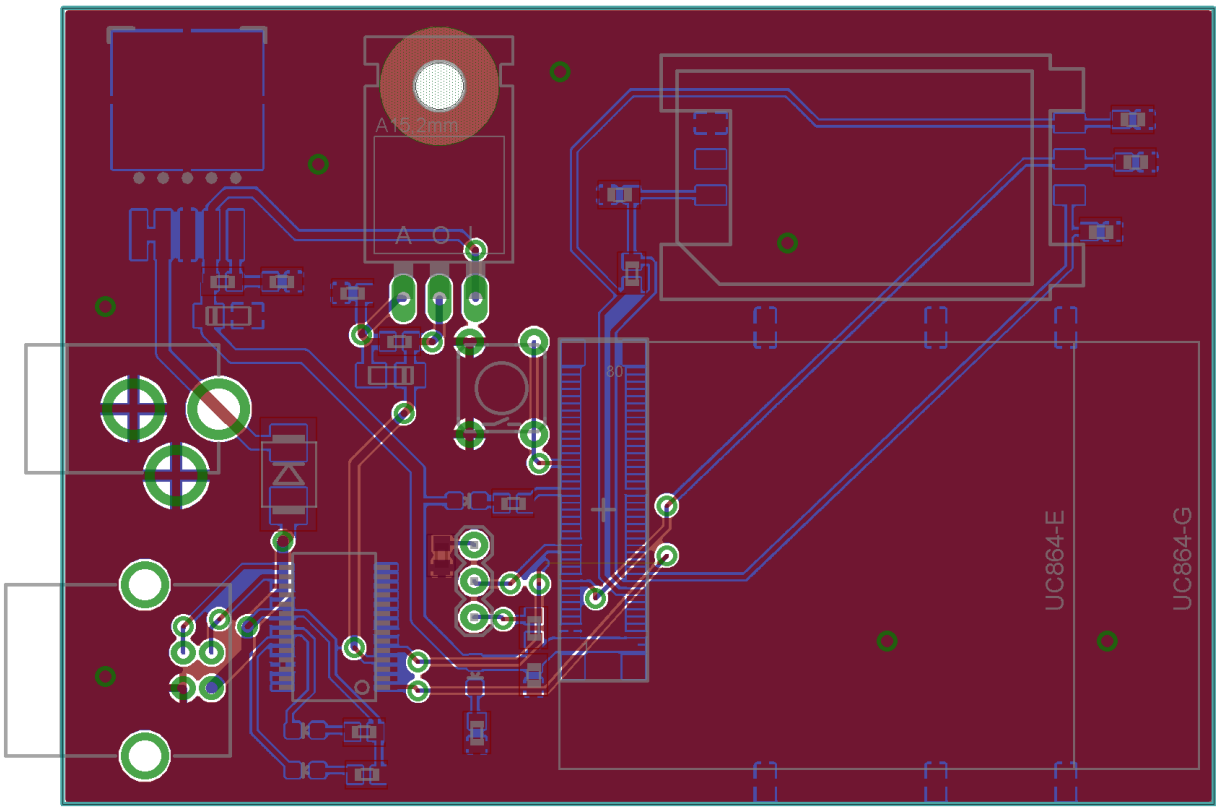


Figure E.5: 3G board layout.

E.3 Power measurement board

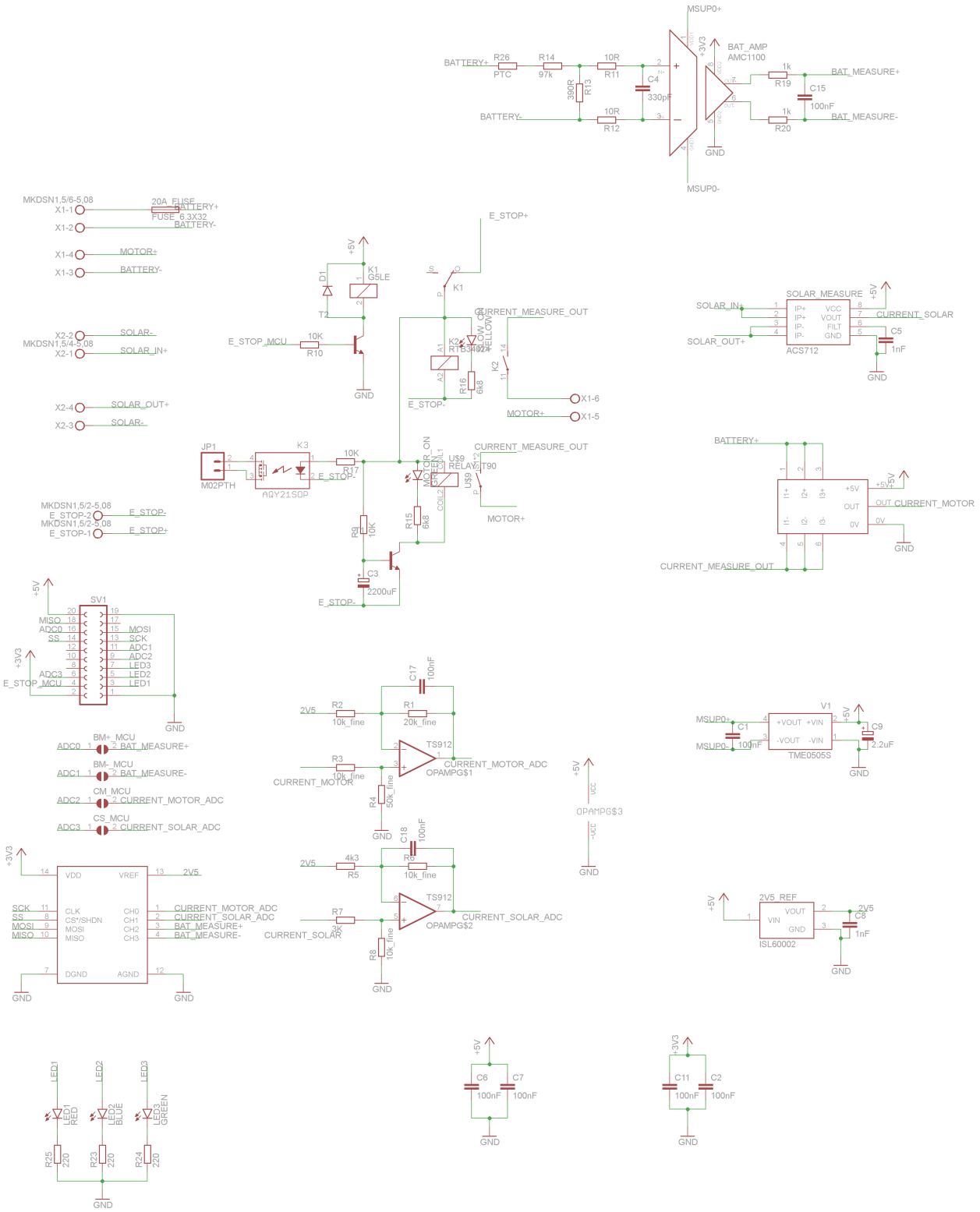


Figure E.6: Power measurement board schematic.

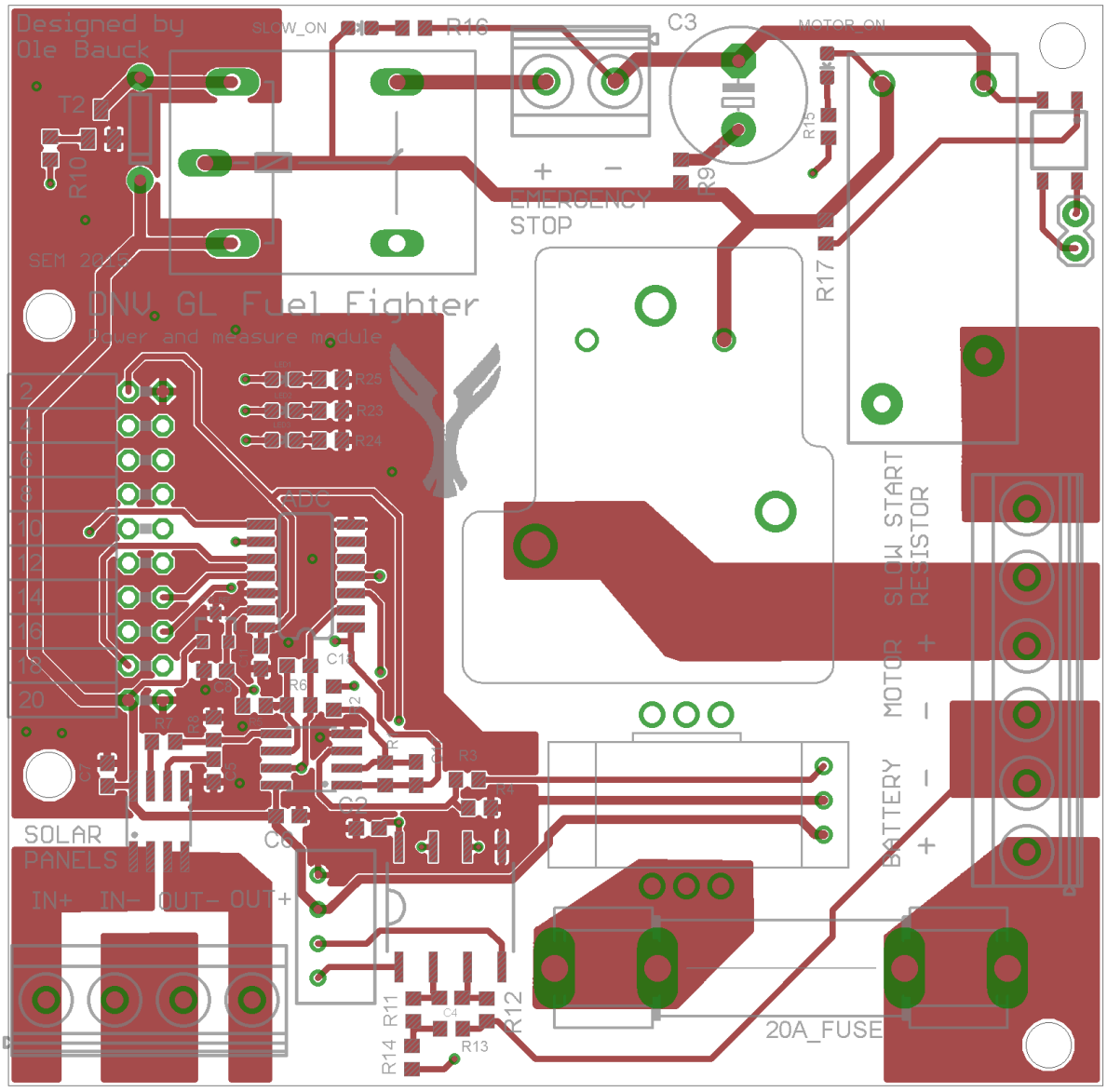


Figure E.7: Power measurement board layout top.

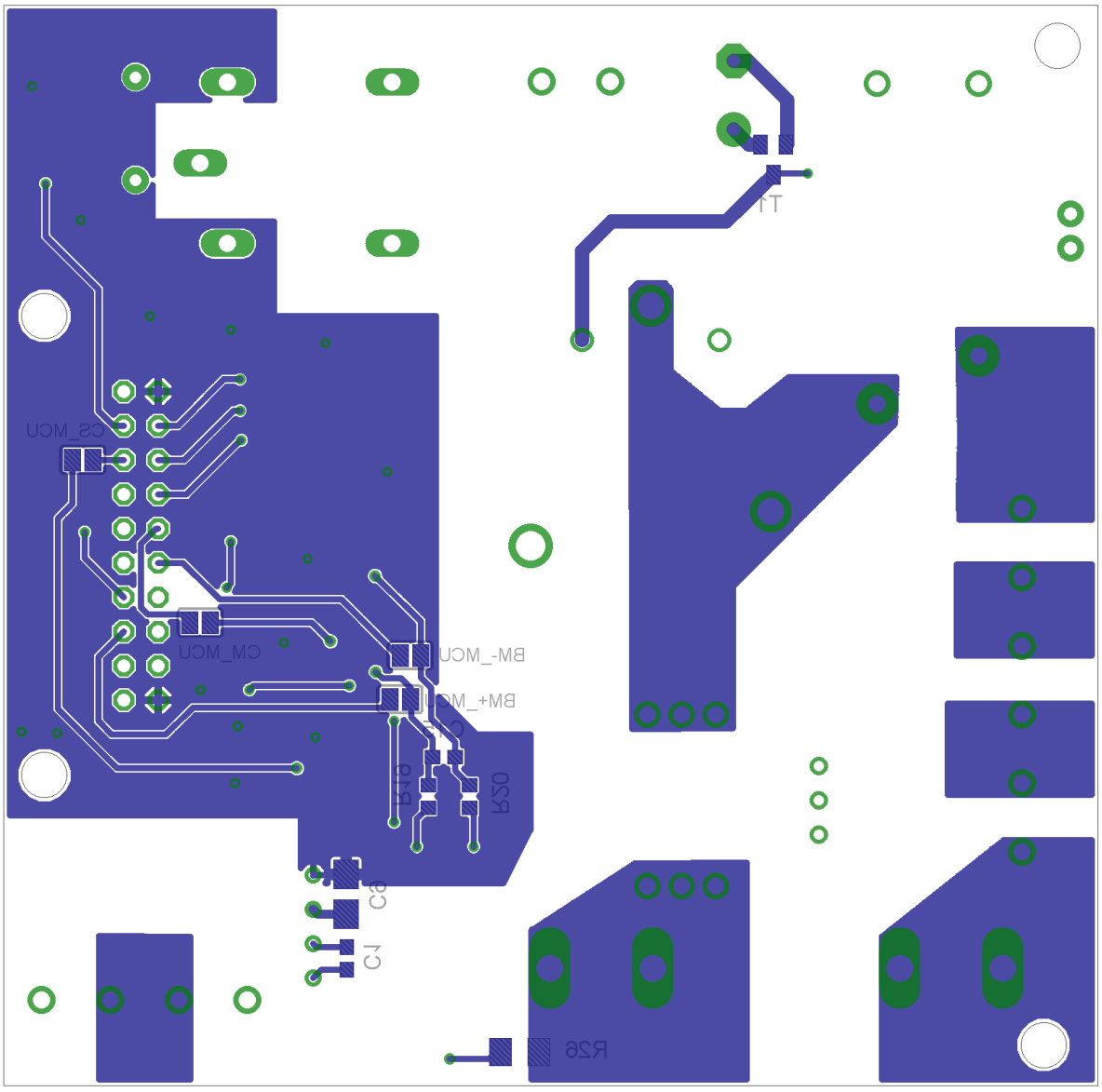


Figure E.8: Power measurement board layout bottom.

E.4 GPS, IMU and SD board

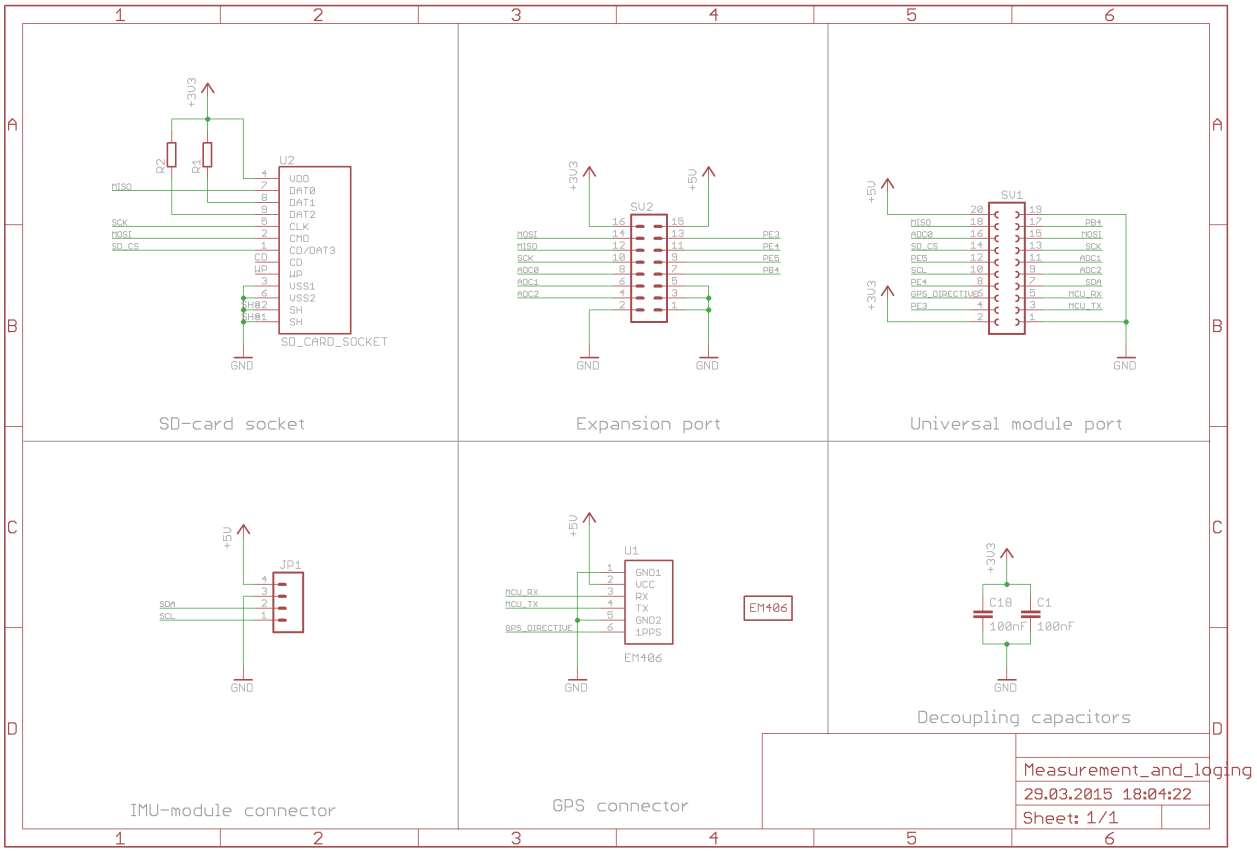


Figure E.9: GPS, IMU and SD board schematic

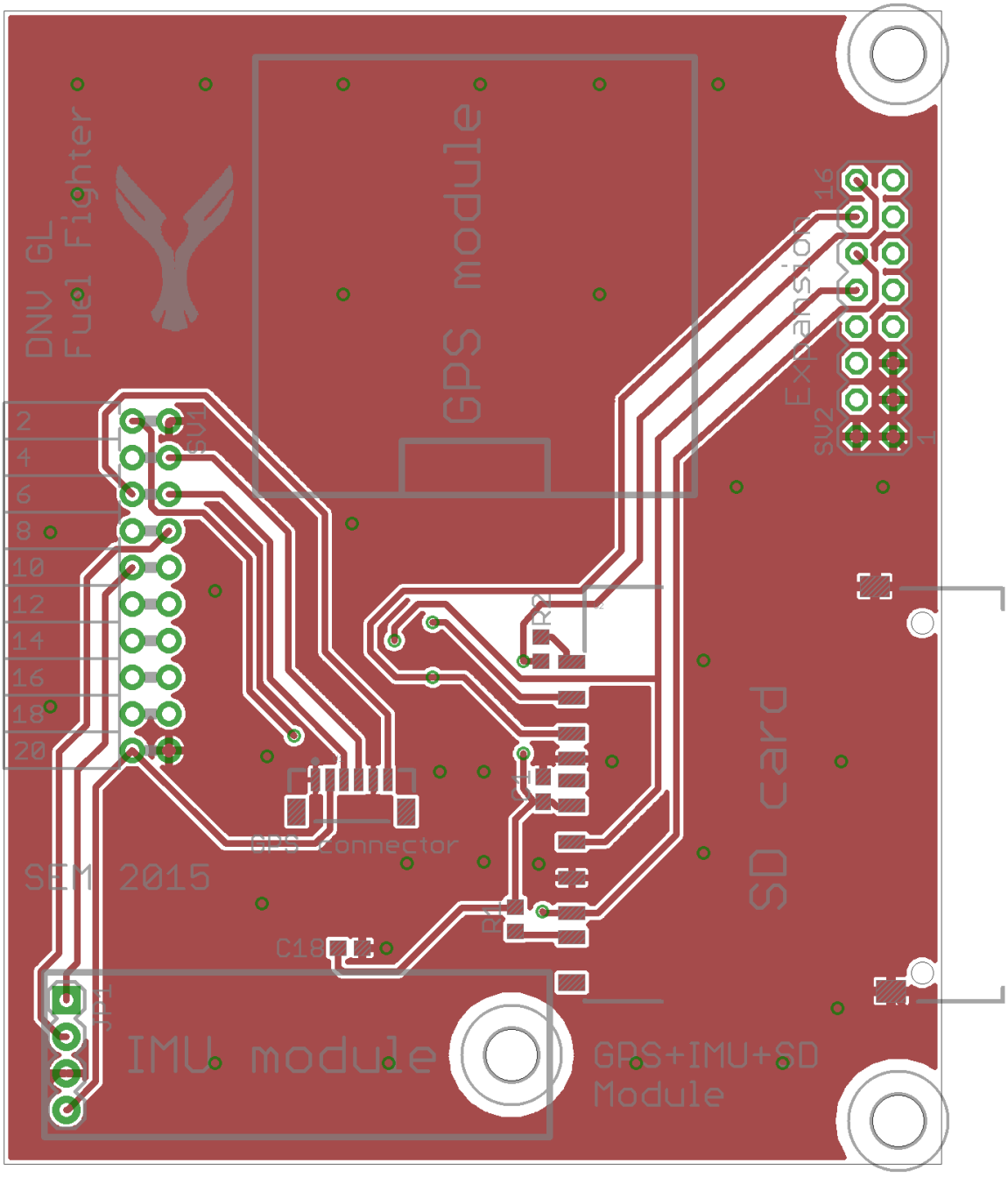


Figure E.10: GPS, IMU and SD board layout top.

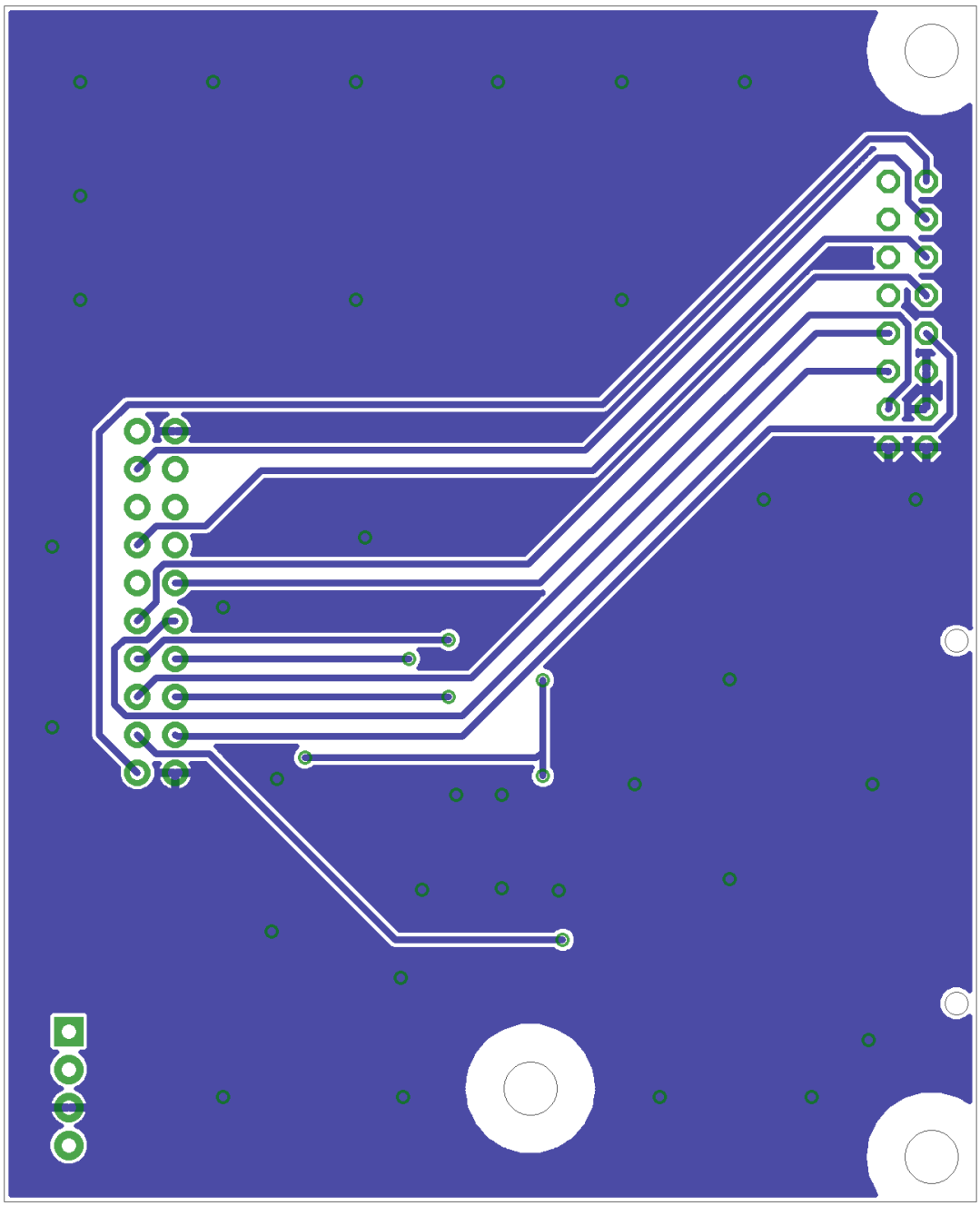


Figure E.11: GPS, IMU and SD board layout bottom.

E.5 RDC board

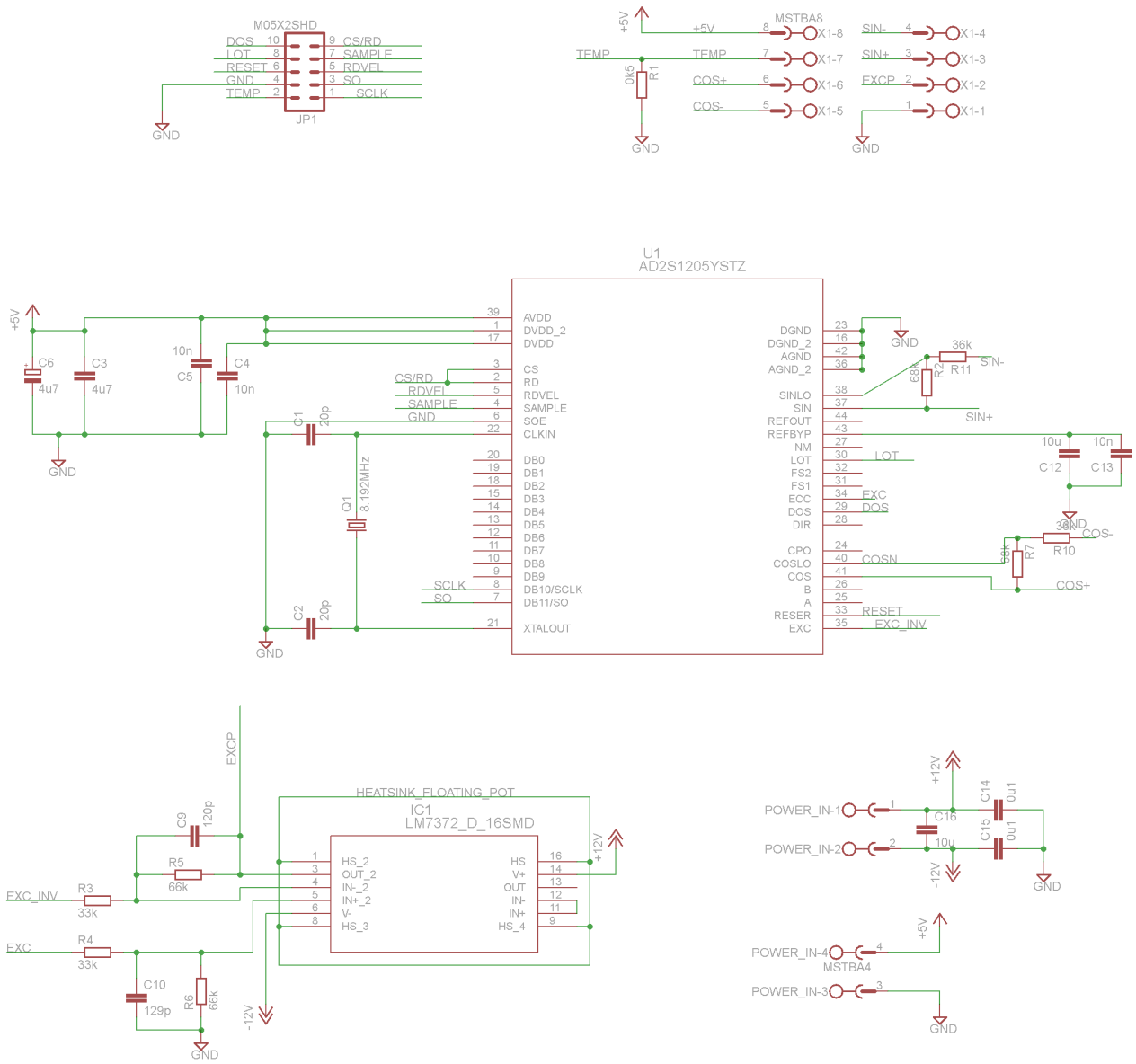


Figure E.12: RDC board schematic.

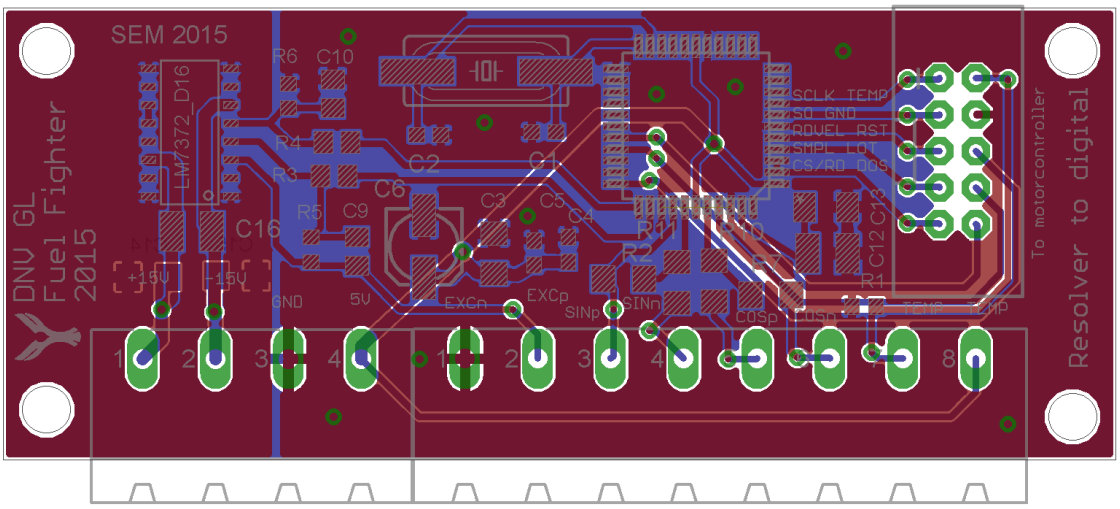


Figure E.13: RDC board layout.

E.6 Main controller board

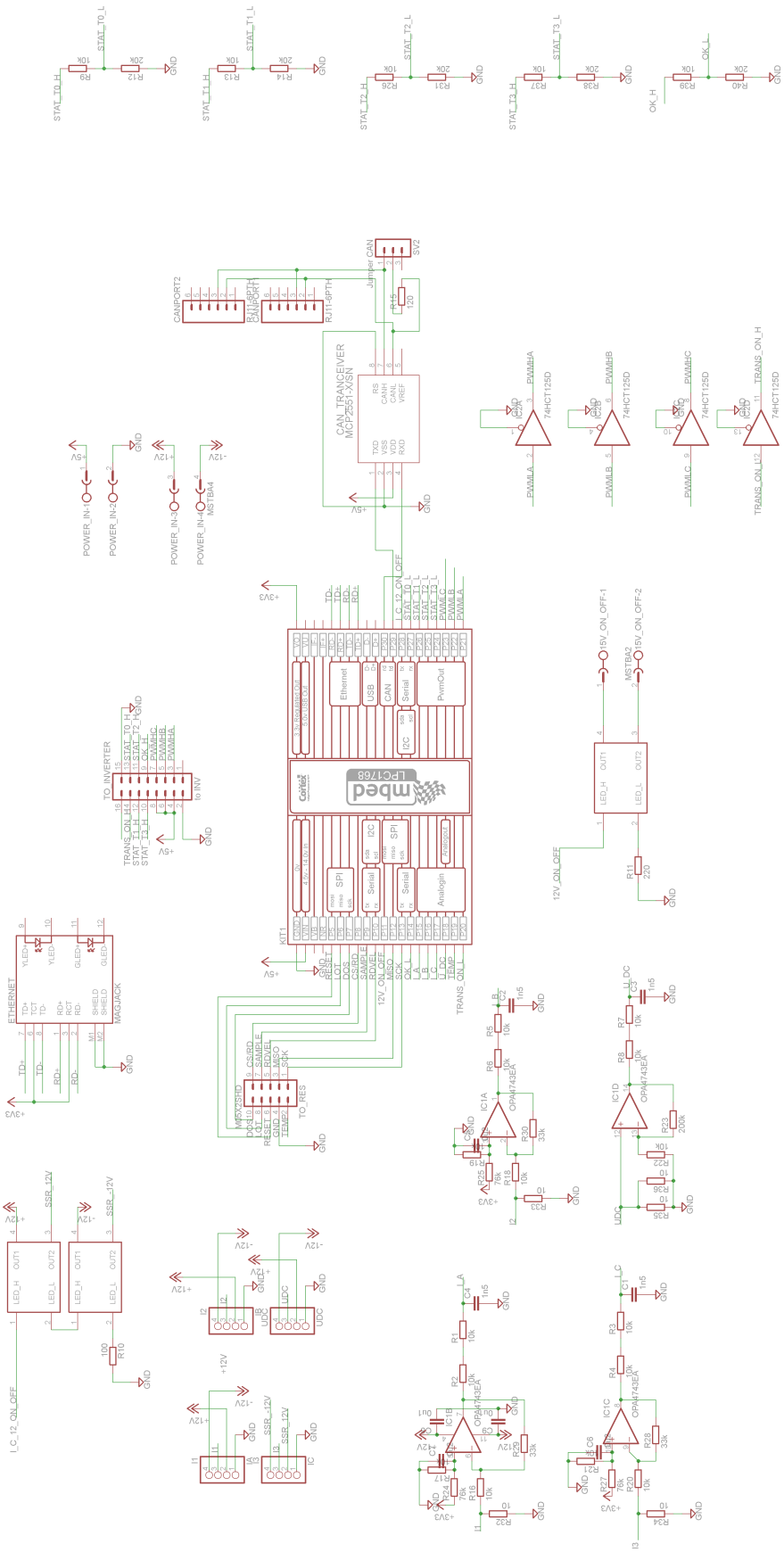


Figure E.14: Main controller board schematic.

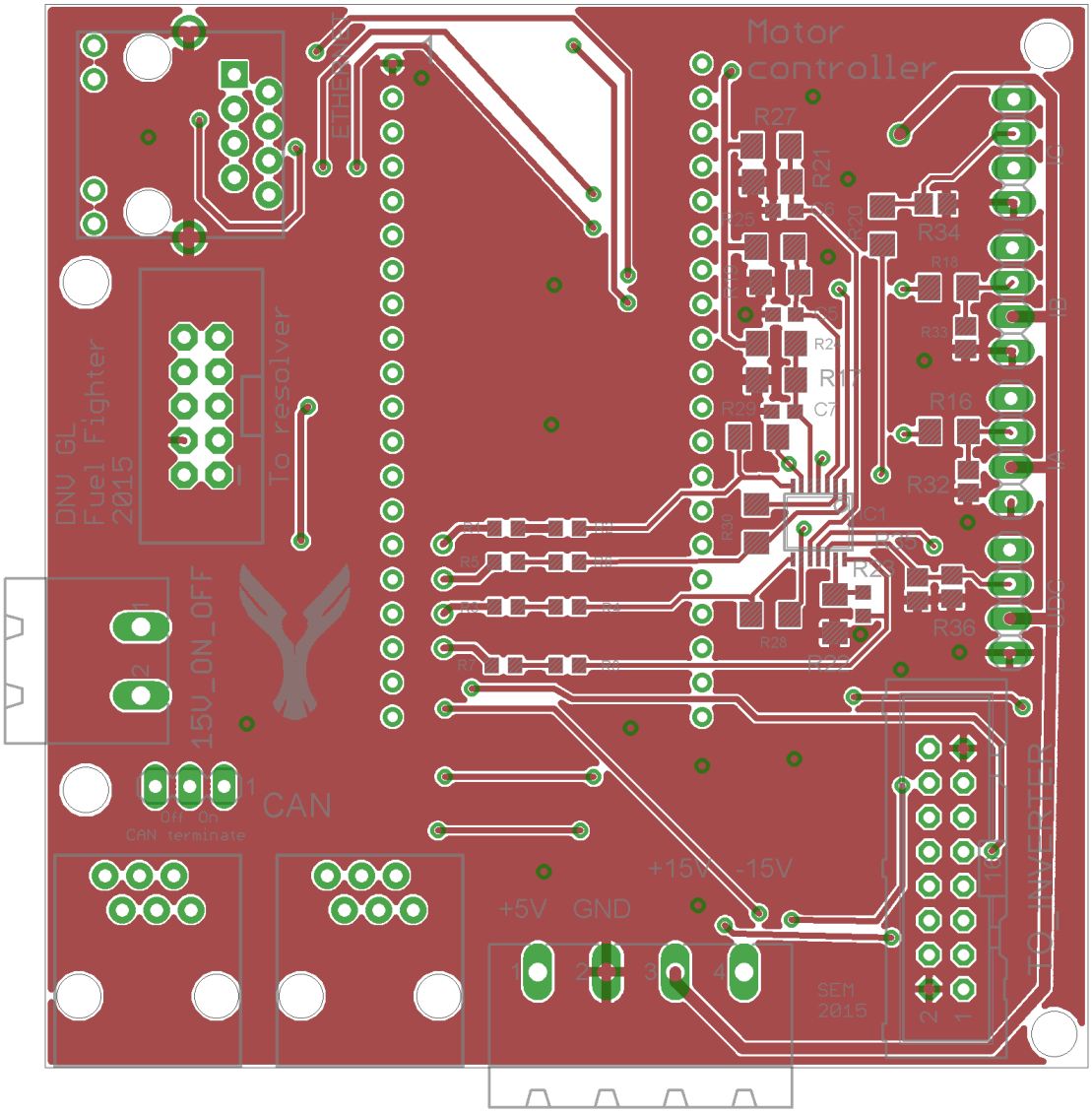


Figure E.15: Main controller board layout top.

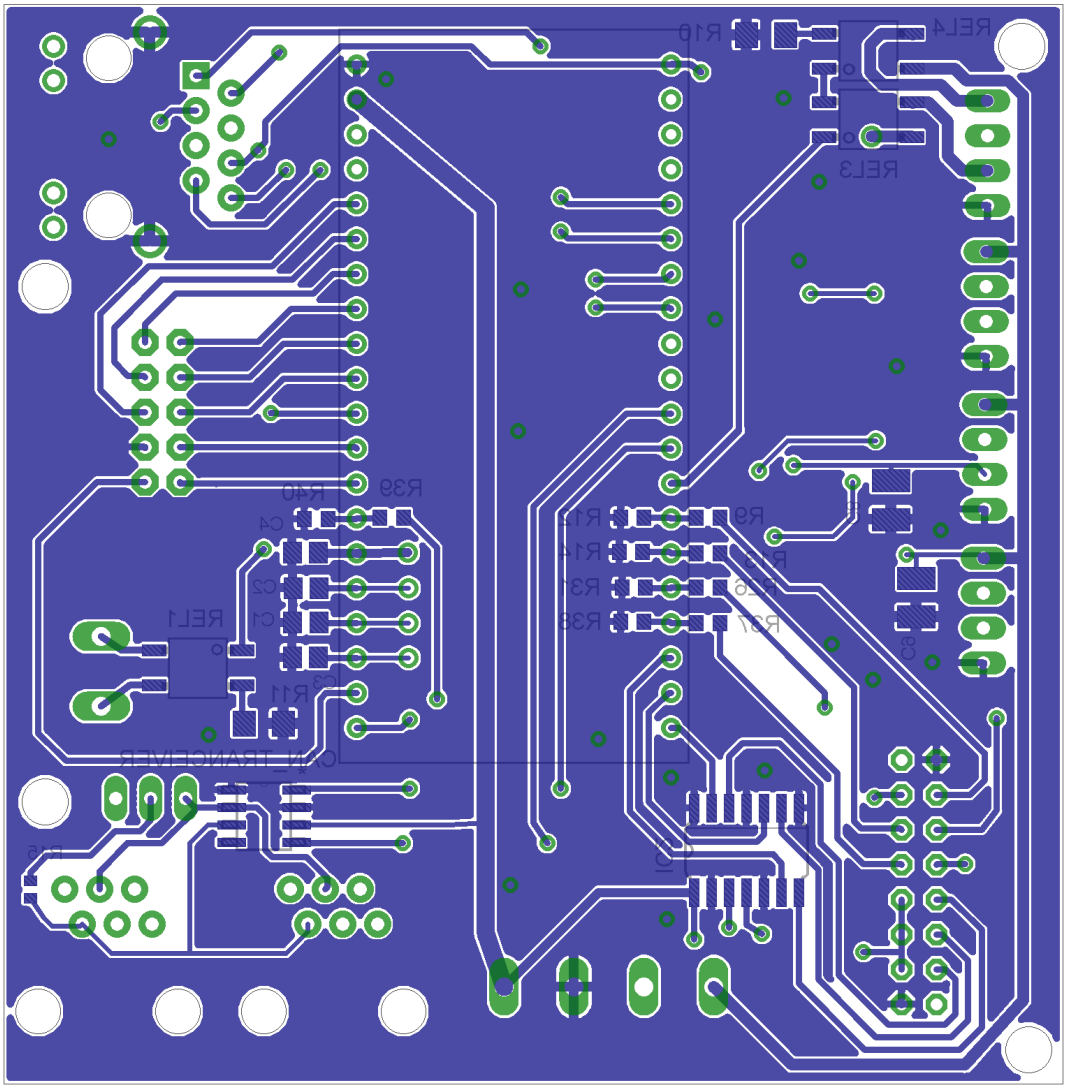


Figure E.16: Main controller board layout bottom.

Appendix F

Race results

If the text is hard to read, see the online version [1] and [2].

[1] http://s00.static-shell.com/content/dam/shell-new/local/corporate/ecomarathon/downloads/pdf/europe/2014-results/sem-europe-2014-results-prototype-battery-electric-220514.pdf

[2] http://s02.static-shell.com/content/dam/shell-new/local/corporate/ecomarathon/downloads/pdf/europe/2014-results/sem-europe-2014-results-urbanconcept-hydrogen-220514.pdf

Shell Eco-marathon

Shell Eco-marathon Europe 2015
Final results: Prototype Battery-electric

27/05/2015

Rank	Team n°	Team name	Country	Organization	Institution type	Competition category	Energy type	Best attempt (km/kWh)	Attempt 1 (km/kWh)	Attempt 2 (km/kWh)	Attempt 3 (km/kWh)	Attempt 4 (km/kWh)
1	327	TUfast Eco Team	Germany	Technische Universitaet Muenchen	University	Prototype	Battery-electric	863.1		722.1	766.5	863.1
2	304	Ruppin-Jet	Germany	Oberstufenzentrum Ostprignitz Ruppin	School	Prototype	Battery-electric	815.7	748	782.5	704.1	815.7
3	344	ECO-DIMONI	Spain	I.E.S Cotes Baixes	School	Prototype	Battery-electric	649	608.8	587.3	612.5	649
4	309	Schluckspecht	Germany	University Of Applied Sciences Offenburg	University	Prototype	Battery-electric	640.5	494.3	611.8		640.5
5	320	Augustine	France	Lycée Léonard de Vinci	School	Prototype	Battery-electric	602.2	469.2	491.6	602.2	586.6
6	308	Team Zero C	Italy	Itis Leonardo Da Vinci	School	Prototype	Battery-electric	600.8	553.7	485.8	450.6	600.8
7	318	Team Eco'Momes 31	France	College Marcel Doret	School	Prototype	Battery-electric	529.8		529.8		514.6
8	338	Smart Power	Poland	Silesian University Of Technology	University	Prototype	Battery-electric	503.7	503.7	497.6	443	
9	310	Eco Motion Team by ESSTIN	France	ESSTIN Nancy	University	Prototype	Battery-electric	495.8	489.9		483.1	495.8
10	328	Prometheus	Greece	National Technical University Of Athens	University	Prototype	Battery-electric	487.4	397.9	402.5	445.9	487.4
11	616	DNV GL Fuel Fighter 2	Norway	Norwegian University Of Science And Technology	University	Prototype	Battery-electric	482.5	482.5	398.6	474.7	
12	317	Riquet Eco Car	France	Lycée Pierre Paul Riquet	School	Prototype	Battery-electric	470.3			431	470.3
13	302	Electricar Solution	France	Lycee Louis Pasquet	School	Prototype	Battery-electric	446.7	410.1			446.7
14	326	PSTVA	France	PST Université Paris Ouest	University	Prototype	Battery-electric	432.9	412.8		405.9	432.9
15	311	elena	Austria	Fachhochschule Vorarlberg	University	Prototype	Battery-electric	430.4	362.3	374	398.3	430.4
16	330	Kandó Electric	Hungary	Kalman Kando Secondary Technical And Vocational Sc	School	Prototype	Battery-electric	405.3	392.6	380.2	405.3	384.1
17	324	Vector Ecoteam	France	MINES ParisTech & Lycées Louis Armand	University	Prototype	Battery-electric	397	295.2	332.6		397
18	345	Solar-GT	Spain	C.I.P.F.P. Benicarló	School	Prototype	Battery-electric	377.2			267.5	377.2
19	341	AERO@UBI	Portugal	Universidade Da Beira Interior	University	Prototype	Battery-electric	330.8	217.4	212.3	330.8	
20	319	Limotion	France	Universite De Limoges	University	Prototype	Battery-electric	321.5	274.4	253.4	312.8	321.5
21	331	Arpad Eco Team	Hungary	Arpad Fejedelem Gimnazium	School	Prototype	Battery-electric	300.7	296.3	300.7		
22	329	Poseidon	Greece	Piraeus University of Applied Sciences (T.E.I. of Piraeus	University	Prototype	Battery-electric	292.8		245	292.8	
23	333	National University of Ireland Galway	Ireland	National University of Ireland, Galway	University	Prototype	Battery-electric	287.1	172.6	187.7	202.1	287.1
24	356	Pteron	United Kingdom	Colchester Institute	University	Prototype	Battery-electric	221		221		
25	353	AE2 PROJECT TEAM	Turkey	Yildiz Technical University	University	Prototype	Battery-electric	217.1	185.1	184.4	200.1	217.1
26	346	UPCT SOLAR TEAM	Spain	Universidad Politécnica de Cartagena - UPCT	University	Prototype	Battery-electric	216.4	216.4			
27	347	UCAM RACING TEAM	Spain	Universidad Catolica San Antonio	University	Prototype	Battery-electric	214.7	214.7	197.5	183.3	203.1
28	314	Avtomobilist	Bulgaria	Univerity of Ruse	University	Prototype	Battery-electric	212.8			212.8	
29	348	GREENWHEEL	Spain	IES FRANCISCO DE GOYA	School	Prototype	Battery-electric	189.5	180.3	189.5	181.2	178.7
30	355	temiz ve yenile	Turkey	Erciyes Univetsity	University	Prototype	Battery-electric	170.4	134.8	136.7		170.4
31	312	blue-ev	Bulgaria	Professional High School N.Vaptsarov	School	Prototype	Battery-electric	131.8			127.2	131.8
32	340	WAT ECO TEAM	Poland	Military University of Technology Warsaw	University	Prototype	Battery-electric	121.1			73.1	121.1
33	336	The Leekburners	Netherlands	The Lindenborg	School	Prototype	Battery-electric	104		104		

Shell Eco-marathon

Shell Eco-marathon Europe 2015
Final results: UrbanConcept Hydrogen

27/05/2015

Rank	Team n°	Team name	Country	Organization	Institution type	Competition category	Energy type	Best attempt (km/m3)	Attempt 1 (km/m3)	Attempt 2 (km/m3)	Attempt 3 (km/m3)	Attempt 4 (km/m3)
1	601	La joliverie Polytech Nantes	France	Polytech Nantes	University	UrbanConcept	Hydrogen	372.6	372.6			
2	610	TUS Team	Bulgaria	Technical University Of Sofia	University	UrbanConcept	Hydrogen	215.4				215.4
3	602	Team Aalborg Energy	Denmark	Aalborg University	University	UrbanConcept	Hydrogen	204.2	191	204.2		
4	604	TUC ECO RACING	Greece	Technical University Of Crete	University	UrbanConcept	Hydrogen	192.7	192.7	187	159.1	
5	620	HIDROIST	Turkey	Istanbul University	University	UrbanConcept	Hydrogen	163.2	163.2	143.8	145	156.2
6	603	HAN Hydromotive	Netherlands	Hogeschool Van Arnhem En Nijmegen	University	UrbanConcept	Hydrogen	143.7		142.9	143.7	
7	622	Smart Power Urban	Poland	Silesian University Of Technology	University	UrbanConcept	Hydrogen	118.2	118.2			
8	605	ENSEM Eco Marathon	France	Ensem Vandoeuvre-Les-Nancy	University	UrbanConcept	Hydrogen	97.4	97.4			

Bibliography

[1] Atmel. *AT90CAN*, 2008. http://www.atmel.com/Images/doc7679.pdf.

[2] Ole Bauck. Wireless monitoring system for the DNV GL Fuel Fighter cars. 2014.

[3] Globalsat Technology Corporation. *GLOBALSAT GPS Module*, 2013. http://cdn.sparkfun.com/datasheets/GPS/EM506_um.pdf.

[4] Analog devices. *AD2S1205 12-Bit RDC with Reference Oscillator*, 2010. http://www.analog.com/media/en/technical-documentation/data-sheets/AD2S1205.pdf.

[5] Shell Eco-marathon. 2015 SEM official rules. 2014. http://s02.static-shell.com/content/dam/shell-new/local/corporate/ecomarathon/downloads/pdf/global/sem-2015-global-rules-chapter1-updated-020914.pdf.

[6] FTDI. *FT232R USB UART IC*, 2010. http://www.ftdichip.com/Documents/DataSheets/ICs/DS_FT232R.pdf.

[7] Simon Fuchs. DNV GL Fuel Fighter towards Shell Eco-marathon 2015 , pre master. pages 109–159, 2015.

[8] Anders Lier Guldahl. Styre- og overvåkningssystem for Shell Eco-marathon kjøretøy. Master's thesis, 2010.

[9] gylling. *Protection modules, Safety and cellbalancing for LiFePO4*. http://www.gylling.no/produkter/batterier/protection-module.shtml.

[10] Benjamin Halsøy and Magnus Buodd. DNV GL towards Shell Eco-marathon 2015. Master's thesis, 2015.

[11] Ragnar Ranøyen Homb. Design of a Data Acquisition System for a Formula Student Car. Master's thesis, 2012.

[12] Horizon. *User Manual H-1000XP Fuel Cell System*, 2013. http://www.udomi.de/downloads/h-1000xp.pdf.

[13] Texas instruments. *AMC1100 Fully-Differential Isolation Amplifier*, 2012. http://www.ti.com/lit/ds/symlink/amc1100.pdf.

[14] Kjell Ljøkelsøy. *Inverter board V1_0 Documentation*, 2014.

[15] Maxon. *Resolver Res 26*, 2009. http://www.maxonmotor.com/medias/sys_master/root/8816814424094/15-374-EN.pdf.

[16] Maxon. *EC60*, 2014. http://www.maxonmotor.com/medias/sys_master/root/8816804528158/15-218-EN.pdf.

[17] Maxon. *maxon EC motor*, 2014. http://www.maxonmotor.com/medias/sys_master/root/8815461662750/EC-Technology-short-and-to-the-point-14-EN-32-35.pdf?attachment=true.

[18] Microchip. *MCP3204/3208*, 2008. http://ww1.microchip.com/downloads/en/DeviceDoc/21298e.pdf.

[19] Eirik Heien Mo. High Efficiency Electric Propulsion Systems for Shell Eco-Marathon 2014. Master's thesis, 2014.

[20] Ned Mohan. *Advanced Electric Drives*. 2012.

[21] Vebjorn Myklebust. Driver interface for shell eco marathon vehicles. Master's thesis, 2015.

[22] Vebjørn Myklebust and Ole Bauck. Electrical report fuel fighter 2014, 2014.

[23] A. Skavhaug, T. Lundheim, B. r. Vik, and T. I. Fossen. A decade of rapid software development for control system experiments: Lessons learned. *Proceedings of the 15th IFAC World Congress, no. 1999*, 2002.

[24] Karl Johan Åström. Control System Design. 2002. http://www.cds.caltech.edu/~murray/courses/cds101/fa02/caltech/astrom-ch6.pdf.

[25] Telit. *UC864 E/G/WD/E-DUAL AT Commands Reference Guide*, 2011. http://www.telit.com/index.php?eID=tx_nawsecuredl&u=0&g=0&t=1434807446&hash=2d370928e4e79025ad931266a7356ef3093fa7a1&file=downloadZone/1947.pdf.

[26] Telit. *UC864 E/G/WD/E-DUAL Hardware User Guide*, 2013. http://www.telit.com/index.php?eID=tx_nawsecuredl&u=0&g=0&t=1434807446&hash=d4ebd90aee222bbabcc6274d958242151c9d4c7d&file=downloadZone/623.pdf.

[27] Telit. *UC864-E*, 2014. http://www.telit.com/index.php?eID=tx_nawsecuredl&u=0&g=0&t=1434807446&hash=dee2f48e11f17c1d2221d060831011e7534ca9fc&file=downloadZone/Telit_UC864-E_Datasheet.pdf.

[28] Øyvind Netland and Amund Skavhaug. Software Module Real-Time Target: Improving Development of Embedded Control System by Including Simulink Generated Code into Existing Code. *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on. IEEE*, 2013.