NTNU – Trondheim
Norwegian University of
Science and Technology

Control system for the DNV GL Fuel Fighter Prototype and the DNV GL Fuel Fighter UrbanConcept

Håkon Trømborg

Master of Science in Cybernetics and Robotics
Submission date: June 2014
Supervisor: Tor Arne Johansen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Control system for the DNV GL Fuel Fighter Prototype and the DNV GL Fuel Fighter UrbanConcept

Håkon Trømborg
Supervisor: Tor Arne Johansen

June 13, 2014

Abstract

In this report I present my work developing the main control unit in the two DNV GL Fuel Fighter cars that raced in Shell Eco-marathon 2014. The objective of the Shell Eco-marathon is to race a set distance using as little energy as possible, and my focus has been on giving the driver the tools and information to run the car in an energy efficient manner. I explored different existing driving strategies, and because of the high power consumption of the motor controller circuit I decided to use the "Pulse and Glide" strategy. This meant the car would accelerate rapidly to maximum speed and then coast with motor and its controller switched off.

Sammendrag

I denne rapporten presenterer jeg mitt arbeid med å utvikle hovedkontrollen i de to DNV GL Fuel fighter-bilene som deltok i Shell Eco-marathon 2014. Målet i Shell Eco-marathon er å kjøre en satt distanse med så lite drivstofforbruk som mulig, og mitt fokus har vært på å gi sjåføren de nødvendige verktøyene og informasjonen til å kjøre bilen på en energieffektiv måte. Jeg vurderte forskjellige kjørestrategier, og på grunn av det høye strømforbruket valgte jeg å bruke en såkalt "Pulse and Glide"-strategi. Denne innebærer å la bilen aksellerere raskt til toppfart, og så rulle fritt med motoren og motorkontrolleren avslått.

Acknowledgements

I would like to thank my supervisor Tor Arne Johansen for letting me work on this project. I also thank the DNV GL Fuel Fighter team 2014 for their hard work and great companionship. Thanks to Audun Lønmo Knudsrød, Lars Lyse Moen, Vegar Østhus and Simen Andresen for believing in me all this time. Finally, a big thanks to my brother Jørgen Trømborg for proofreading and good advice.

Project description

Design of model predictive control for the Eco-marathon car

The Shell Eco-marathon is a global competition that challenges students to apply their knowledge of engineering and design to create energy efficient cars. Studies of the previous results show that the choices the driver makes regarding speed, passing or following of other cars could be the difference between standing on top of the podium and not being on the podium at all. The NTNU car: DNV Fuel Fighter is in constant development, and we need cybernetics engineers to reach the top.

The job will be to design a model predictive control (MPC) system that integrates with the existing control system. The system should control the velocity of the car while leaving the actual steering to the driver, as required by the Shell Eco-marathon rules. This could be done based on GPS input and other data such as time, buttons pushed by the driver and sensors such as proximity sensors. From this point the imagination is the only barrier. The time left, density of cars (preventing the car from following the actual optimum driving strategy) and the track itself could all be used to constantly calculate the optimum velocity.

This project should preferably be continued as a master thesis, and then the total work should be dived as follows:

Project:

- Study of the current control system
- Literature study of MPC in vehicles.
- Problem specification.
- Identification of necessary sensors and input sources.
- Algorithm development and simulation of a simplified system..

Master thesis:

- Purchase of necessary hardware.
- Assembly and implementation of the MPC system.
- Optimisation during testing in Trondheim.
- Optimisation during testing and racing on the race course.

Abbreviations

ECU	Engine Control Unit
MCU	Main Control Unit
DNV FF	DNV Fuel Fighter
DNV GL FF	DNV GL Fuel Fighter
FPGA	Field Programmable Gate Array
DIO	Digital Input/Output
AIO	Analog Input/Output
RMC	Rio Mezzanine Card
VI	Virtual Instrument

Contents

List of Figures

1 Introduction

1.1 The competition

The Shell Eco-marathon is an annual competition for engineering students hosted by Royal Dutch Shell plc. As of 2014, the competition is arranged in Asia, the Americas and in Europe. This year's (2014) European competition was held in Rotterdam.
The objective in the Shell Eco-marathon is to build a car that can carry a driver as far as possible using as little energy as possible. Several different sources of energy are allowed as propulsion, but each car only competes against other cars using the same energy source. The energy sources allowed are

- Gasoline
- Diesel
- Alternative fuel (Ethanol, GTL[1])
- Hydrogen
- Battery electric and solar power

The competition is also split into two classes, the Prototype class and the UrbanConcept class, as explained in chapters 1.3 and 1.4.

Upon satisfying necessary safety criteria and successfully completing a thorough technical inspection, each car is allowed to partake in the race. The cars start one at a time, race for 10 laps which corresponds to 16117 meters, using a maximum of 39 minutes to complete. This time limit is set to ensure that

[1]From wikipedia: GTL - Gas to liquids is a refinery process to convert natural gas or other gaseous hydrocarbons into longer-chain hydrocarbons such as gasoline or diesel fuel.

the average speed of the vehicles is at least $25\frac{km}{h}$. Once the car crosses the finish line, the energy used is inspected by an official. Several methods for measuring energy consumption are used. In the Battery Electric category where DNV GL Fuel Fighter competed, two Joulemeters measure electric power going into the motor controller circuit, and electric power coming in from the solar cells. To limit solar cell dependency, no more than 20% of the energy used for propulsion may be covered by the solar cells.
The result in Joules used to power motor and motor controller minus Joules produced by solar cell panels (up to a maximum of 20%), is then expressed in $\frac{km}{kWh}$. The winner is the one who was able to use least energy, this year Team TERA TU Graz from Austria, reaching the equivalent of 1091.6 $\frac{km}{kWh}$, or around 23W.

1.2 The track

This year, like last, the European competition took place at the Ahoy! Arena in Rotterdam (figure 1) , a large convention centre. The track runs around the convention centre, and takes up parts of the main road next to it. Each round is 1626m, except for the slightly shorter last round, making the total distance 16,117m. The race is run clockwise, and there are five 90° turns. It is close to flat all over, with the far right corner being the lowest point of the track. It is mostly wide enough for two cars to be driving alongside each other, but usually not wide enough for three.

1.3 The Prototype

The Shell Eco-marathon is divided into two classes. The most competitive one is called the Prototype class. Here, there are few requirements to what cars must be able to do, except travelling as energy efficiently as possible

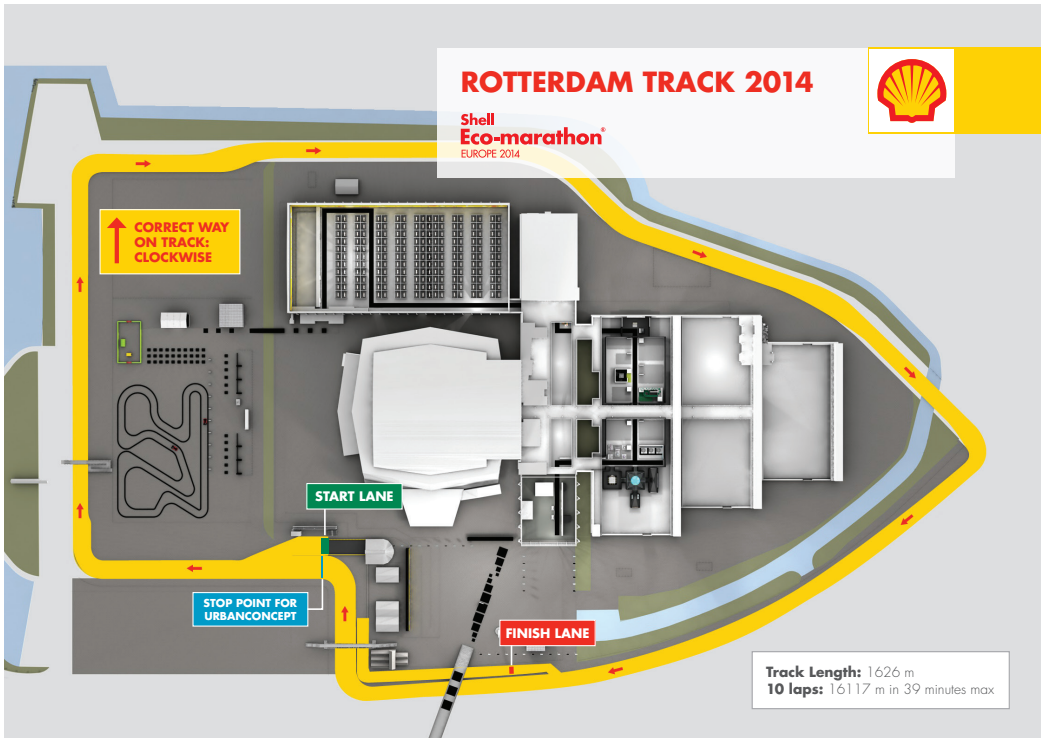

Figure 1: The Ahoy! Race Track

while keeping a certain level of safety for the driver and other participants. Over time the cars have started to move towards similar shapes. Most cars now have two wheels in the front, one wheel in the back, and are in many ways shaped like a drop (figure 2. Most are also as small and light as possible, and the driver is usually lying down while driving.

The main design criterion for the shape is to reduce air friction, but even so this is the biggest loss for most Prototype cars even at relatively low speeds such as $25\frac{km}{h}$.

Figure 2: Example of the most common shape for prototype cars, the TIMO5 from Toulouse, France. All Rights Reserved Shell Eco-marathon 2014.

1.4 The UrbanConcept

In 2003 Shell introduced a new class in the competition. The competition had been driven towards the more extreme Prototype cars, and Shell wanted to put some focus back on making solutions that could realistically be implemented in real vehicles. In the UrbanConcept class, cars must meet several roadworthiness criteria found in modern passenger vehicles. Restrictions are placed on car dimensions, number of wheels, door size and several electrical components are now compulsory. The car must have front lights, turning indicators, brake lights, braking pedal, wind-shield vipers and is required to come to a complete stop between each round. As a result, the UrbanConcept cars are not able to travel as fuel efficient as the Prototype cars, and many of the teams focus more on visual appearance and innovative design. The variations in design are much bigger in the UrbanConcept, and it's more of a fun-class than Prototype. This year, for the first time, one of the vehicles

in the competition was accepted for normal roads in the Netherlands.

Figure 3: Example of UrbanConcept cars. All Rights Reserved Shell Eco-marathon 2014.

1.5 The team

NTNU has been sending a car to the Shell Eco-marathon every year since 2008. It is mainly a project for students at NTNU, but some students from other schools in Trondheim also take part in the team. For the last years the main sponsor has been DNV GL, covering about 70% of the teams budget. This year's DNV GL Fuel Fighter team consisted of

- A project manager
- A risk manager
- Five mechanical engineers
- Three cybernetics students (my role)
- A power electronics student
- A graphical designer
- Two drivers
- A photographer
- A PR representative

1.6 The cars

For the first time ever NTNU decided to send two cars to the Shell Eco-marathon. The UrbanConcept from 2013 had done well, but due to motor problems hadn't really realised its potential. Little had to be done to design and mechanical system, and it was decided to try again this year. However, the main focus this year was to make a Prototype car from scratch.

Figure 4: The DNV GL Fuel Fighter 2014 team.

1.6.1 The UrbanConcept

The UrbanConcept had a third place from last year's competition. It also won the award for best design and was still working well when we overtook it, except from some wheel problems. The motor was replaced, and alterations were made to the electrical system to make it similar to the Prototype's electrical system.

1.6.2 The Prototype

The new DNV GL Fuel Fighter Prototype was where the mechanical engineers put most of their work. Design started in September, and building in January. The chassis was made from carbon fibre, steering wheel 3D-printed, and mostly everything was designed by NTNU students. The motor used was a small three phase brushless motor designed for RC planes. The final weight was 42 kg, and the air resistance calculated to 6% of a what a

Figure 5: The DNV GL Fuel Fighter UrbanConcept.

bicyclist experiences at $30\frac{km}{h}$.

1.7 Plan

1.7.1 Project status Januray 2014

When the work on this thesis begun in January 2014, I had recently completed my project report on creating an MPC controller for the DNV GL Fuel Fighter cars. The UrbanConcept car was mostly ready for use, except for the motor controller circuit that had to be changed. Nobody had yet been found to work on the electrical system, but we hoped we would be able to recruit someone. A sponsorship with National Instruments had been agreed

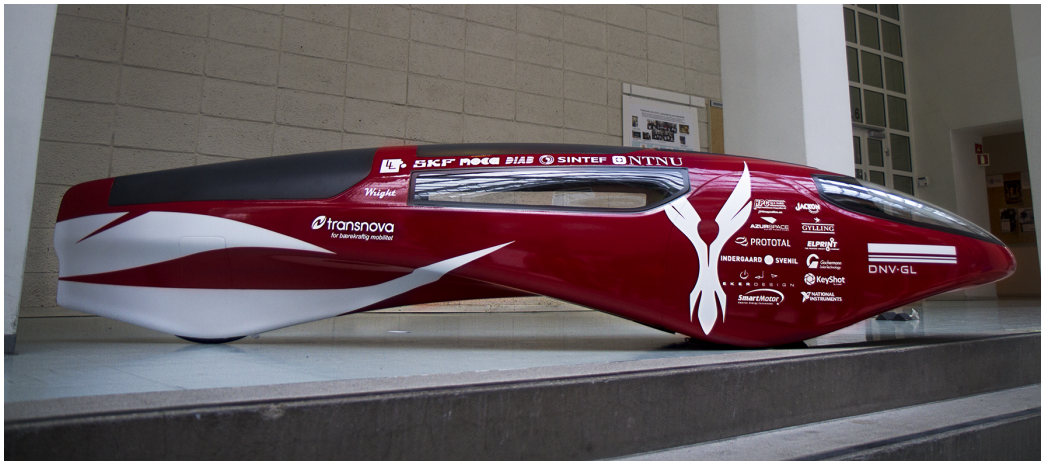

Figure 6: The DNV GL Fuel Fighter Prototype.

upon, but no hardware had yet been received. The mechanical team was done with their technical design and were ready to start building the car. We agreed to be ready for testing after Easter, on May 21.

1.7.2 Plan for the project

The plan was to recruit one or several students to work on the electrical system. If nobody were found, I would have to make the electrical systems in the two cars myself. I would work on the main control unit, and create the user interface for the driver. Once the hardware arrived, I planned to start learning the software and start prototyping. I would start running the motor in the laboratory as soon as possible, and on-track testing of the cars would begin May 21.

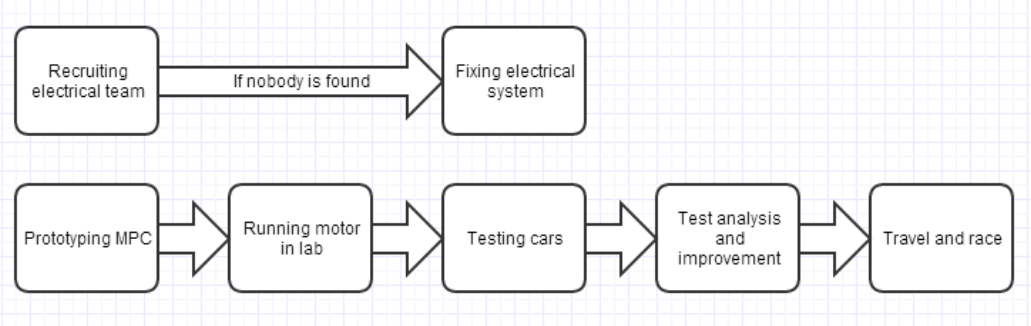

Figure 7: The initial plan for phases of the project work.

1.8 Thesis objective

The objective of this thesis is to describe in detail my role in the project and to document the considerations underlying the design choices I contributed to. My task in the project was first of all to decide on a driving strategy for the driver, and to create the necessary user interface for her to control the car in an energy efficient way. This included design decisions such as hardware and software selection, designing the input method, i.e buttons and joystick on the steering wheel and output in form of an LCD display, controlling the motor controller and gathering information from different sensors and instruments to present on the screen.

It also included some managing of the electrical team, modelling and testing the engine's and the motor controller's efficiency and reliability, logging driving data, designing a CAN bus protocol, guiding the drivers by cell phone during testing and racing, and finally analysing the results to give tips to the next year's DNV GL Fuel Fighter team.

In my project report from fall 2013, I explored the possibility of optimizing the driving strategy using and automatic MPC control algorithm. A number

of assumptions were made in this project that turned out to not hold, mainly that the standby power consumption of the motor controller was negligible, but also the simple model of engine Torque-Speed characteristics, assumption of negligible traffic and other track specific factors. As time building the car drew out, the lack of testing possibilities called for a simpler scheme than MPC. The deadline we had set for finishing the car, 21st of April, was pushed forward again and again, until we finally managed one hour of test driving the night before departure. Needless to say we opted for robustness rather than optimization of the driving strategy, and the result was what is known as a "Pulse and Glide" method[2] which will be explained in detail in chapter 2.1.1 .

2 Literature review

2.1 Fuel efficient driving strategies

2.1.1 Pulse and Glide

The Pulse and Glide strategy is widely used in competitions where fuel efficiency is key.[2] Once fuel consumption is pushed to the lower limit, the engine brake thermal efficiency becomes an increasingly important factor. The Pulse and Glide strategy consists of accelerating relatively fast to a certain speed, and from there rolling with the engine switched off. The car is then restarted after a while, and starts driving again. This method is shown to have excellent results on vehicles where thermal loss or electrical loss in the engine is a notable percentage of the cars energy consumption.

In order to fully exploit the possibilities in the Pulse and Glide strategy, it is necessary to choose a speed interval in which the car is driven. Choosing the correct positive and negative deviation from the desired average speed is essential, as well as the acceleration rate in the acceleration phase. The optimal values are decided by the car's aerodynamic characteristics, its propulsion system and mechanical frictions, as well as environmental properties such as slope. In [2] it is shown that the Pulse and Glide method is superior to keeping a constant speed when used on a Ford Focus 2007, at best more than twice the distance per unit of gasoline. The results vary however, and at speeds above 40 mph the results aren't as significant. This is explained by the quadratic nature of air friction, as running at speeds higher than necessary is increasingly energy expensive at higher speeds.

2.1.2 Optimal driving strategies for trains

Many studies have been concerned with optimizing fuel economy for a train journey. Many of these look at how to best handle driving up and down hills. In [12] a simple longitudinal train model is developed to optimize the control problem

$$J = \sum_{k=0}^{n} f_j(k+1)\tau_{k+1} \tag{1}$$

where J is total fuel consumption, $f_j(k+1)\tau_{k+1}$ is fuel consumption rate multiplied by time interval, and $k \in [0, n]$ represents the intervals needed to finish the trip.

The train's propulsion system is modelled as having three different phases: power, coasting and brake phase. The slope of the trip is assumed to be known, and there is also imposed a speed limit on the train. The remaining problem is then to find optimal switching points between the three phases. Solving the system of equations , it is shown that it is fuel optimal for a train to accelerate before an uphill, and to decelerate before a downhill as seen in figure 8.

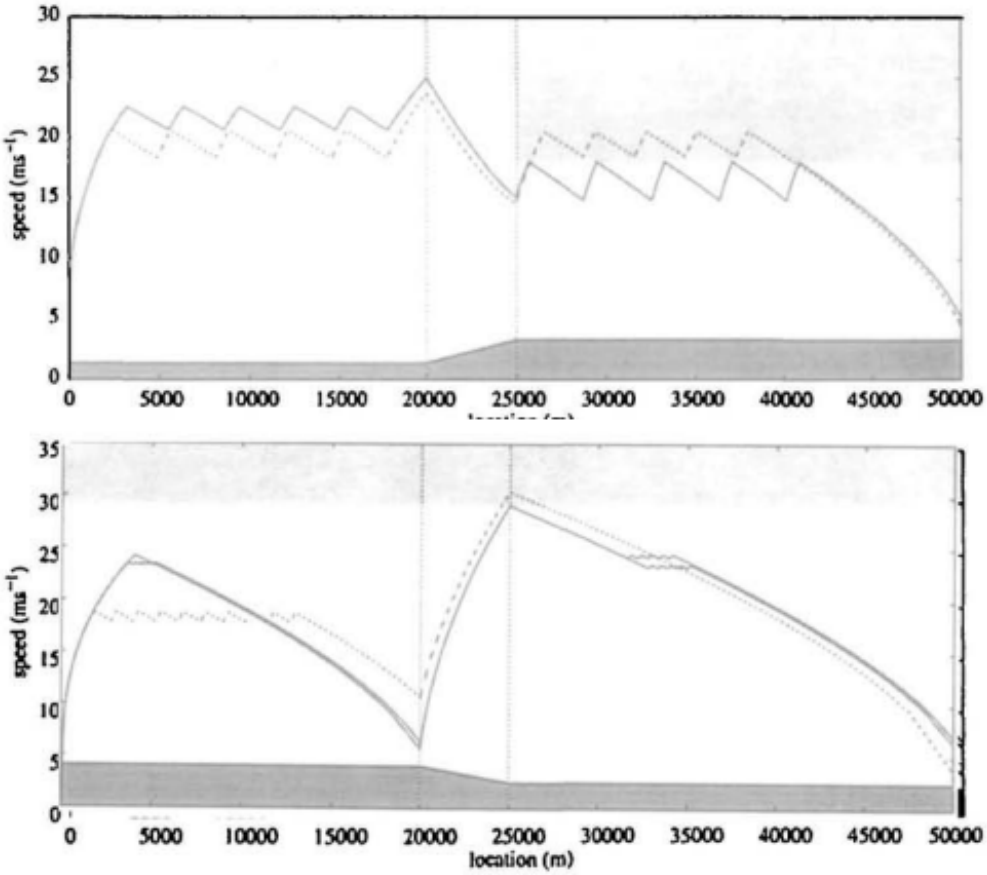

Figure 8: Simulated solutions for the fuel optimal train journey problem. Switching between coasting and accelerating on flat parts, while accelerating and then coasting uphill, or decelerating then coasting downhill.

2.1.3 Fuel economy when starting and stopping

In the paper Optimal Driving for Single-vehicle Fuel Economy[14] from 1998, J.N Hooker presents an analysis of 15 then late-model cars' fuel consumption. Focusing on statistical data from testing rather than mathematical models, he investigates optimal speeds on constant and varying slopes, acceleration rates and driving between stop signs. The idea is to be able to give general advice for fuel efficient driving for cars, and not only particular models.

The optimization problem is formulated as

$$\min_{v,T} \int_0^T f(v(t), a(t))dt \tag{2}$$

subject to

$$\begin{gathered} \dot{s}(t) = v(t) \\ a(t) = \dot{v}(t) + gsin\theta(s(t)) \\ a(t) \leq a_{max}(v(t)) \\ v(0) = v_0, s(0) = 0 \\ v(T) = v_s, s(T) = s_1(optional) \\ T = T_s(optional) \end{gathered} \tag{3}$$

Here, $f(v, a)$ is the fuel consumption at speed v and acceleration a. $\dot{s}(t) = v(t)$ is the system equation, $\theta(s)$ is the slope at position s, v_0 initial speed, v_1 terminal speed and T is the duration. The objective is to find the optimal trajectory $v(t)$ that minimizes the fuel consumption.

While the case for accelerating and maintaining cruising speed and slope handling is very dependent on gear changes and motor efficiency at different RPM's for gasoline vehicles, the result for driving between blocks is quite

interesting also in the case of DNV GL Fuel Fighter. Hooker argues that it is optimal to drive up to the stopping point and then brake as hard as possible, making the deceleration phase short as possible. Since all kinetic energy must be lost when stopping, doing so instantly instead of over time does not alter the fact that it is lost. What does however change, is that when decelerating softly one needs a higher peak speed in order to reach the stop at the same time as when braking hard at the end. Additionally it is pointed out that when time is not a hard constraint, braking hard at the end will reduce total time and so also the time where the car is idly coasting is reduced. An example can be seen in figure 9.

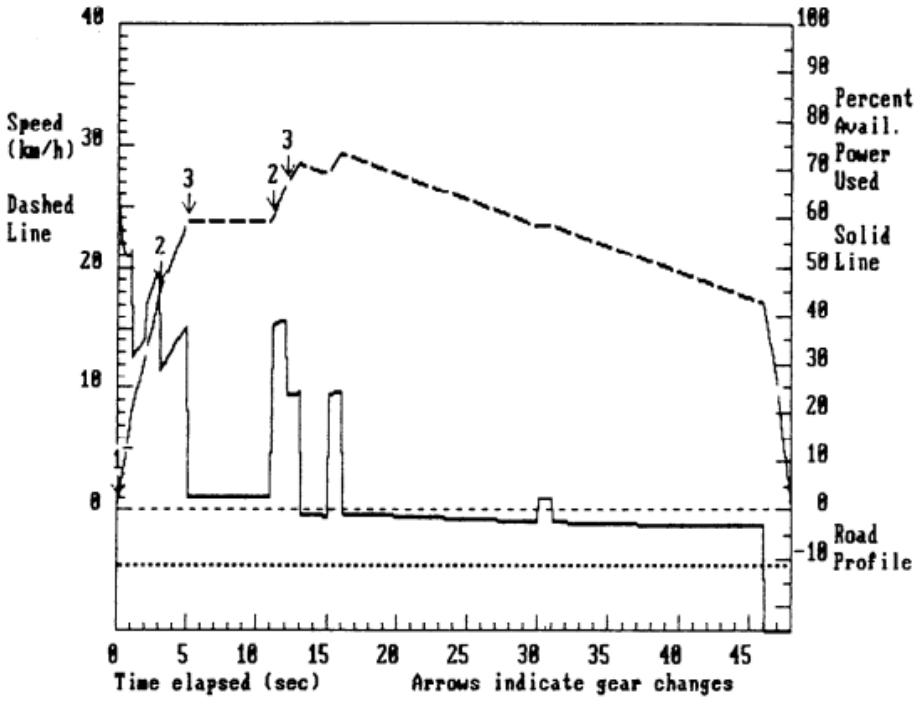

Figure 9: The dashed line shows the speed of a Toyota Corolla from start to stop over a 300 m distance. It accelerates quite rapidly to exploit the the motors most effective torques, and coasts until it brakes as hard as possible before the stop sign.

2.2 Earlier work

Many teams from NTNU before us have competed in Shell Eco-marathon. Some of them competed in different categories, and several were more concerned with having at least one complete run than optimizing driving strategy. In his thesis from the 2011 competition, Jardar Sølna Øverby recommends exploring a torque controlled motor over a speed controlled motor.

In 2012, Benjamin Gutjahr developed an algorithm he chose to call Automated Section Controlled Drive[8]. Partitioning the Ahoy! track into 4 sections, he derives a mathematically optimal torque profile. He mainly weighs a complicated track and car model, as well as a motor efficiency analysis that is not presented in the text. The motor broke this year, and the track is now run in the opposite direction. It is therefore difficult to make use of his analysis.

In last year's report[10], ambitions were to make an MPC controller for speed. Ambitions were, however, too high and much simpler strategies were adopted. All but the last attempt failed, and their final strategy was drive as fast as possible at all times.

Earlier work doesn't help much with deciding a driving strategy. If one important thing can be learned it is that testing is very important, as complicated systems such as a car are likely to fail in unanticipated ways.

3 The car system

The two cars had mostly identical electrical systems. Figure 10 is a simplified sketch of the system, but everything that was relevant for the main control unit is shown.

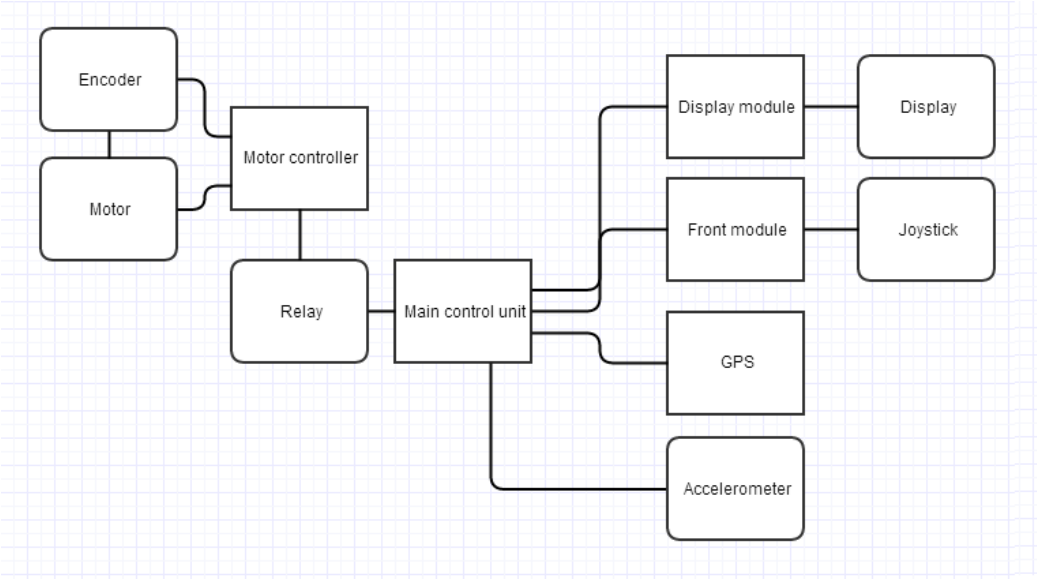

Figure 10: Sketch of the car system. Rectangular blocks represent computers and microcontrollers communicating on the CAN bus while the round edged shapes are simpler circuits.

The MCU links the driver to the actuator (motor). The driver gives input through the joystick and its buttons, the front module translates this and sends it to the MCU. Based on this input the MCU sends commands to the motor controller which creates torque from the motor. The motor controller also sends status messages back. Information sent to the MCU that is considered relevant to the driver is sent to the display module which presents this on the display. The relay shown enables the MCU to open and close the circuit leading from the battery to the motor controller circuit. This makes it possible to turn the entire motor controller circuit on and off automatically

while driving, which saved us a lot of energy.

When the DNV GL Fuel Fighter team of 2014 took over the project from last year's team, we had one more or less working car. The UrbanConcept had a lot of custom made electronics and modules, but some alterations had to be made. I decided to switch out the main module, because I wanted to use different software and hardware. Everything that would be reused also had to be made twice, as we had two cars this year. I was in charge of programming the main control unit, and the work concerning electrical system and smaller modules was done by Ole Bauck and Vebjørn Myklebust from fourth year Cybernetics. The motor and motor controller part was done by Eirik Heien Mo from Power Electronics, but I present the entire system in this chapter because it is relevant to understanding my decisions.

3.1 Electronics

The two cars had slight differences in their electronics, as the UrbanConcept class required some accessories that the Prototype did not. This consisted of lights, turn signals wind-shield vipers, while the modules that were interfaced from the main control were identical in the UrbanConcept and the Prototype. Because of this the MCU was almost identical in the two cars, except for some constants based on car properties such as wheel radius and gear ratio.

3.1.1 Display module

The display module communicated with the main control via the CAN bus. It parsed the CAN bus frames and retransmitted them to an LCD screen using UART and displayed the data on a 20x4 characters LCD screen from Sparkfun Electronics. The update rate of the LCD was programmed to 5Hz.

The display was used to show the driver

- estimated speed
- torque
- the latest status sent from the ECU
- the travelled distance from the start line
- time since start in minutes and seconds
- estimated average speed needed to finish on time
- the current driving mode
- a bar indicating centripetal acceleration, as an anti roll-over feature

The speed was estimated either by multiplying the ECU's rpm value by wheel circumference and dividing by gear ratio, or by readings from the GPS. The ECU had a vast amount of possible status messages however, and while the list was too long for the driver to memorize, it provided valuable insights to me and the rest of the electrical team when test driving or racing. How the values presented on the screen were calculated will be explained in detail in chapter 5.

3.1.2 GPS module

The GPS module contained a GPS antenna, a microcontroller and 9-axis IMU module. It had an accuracy of 10 m, and even though the start-up time was documented as 42 seconds, it was normally 10-20 seconds. The GPS module provided GPS coordinates as well as travelling direction and speed.

3.1.3 Front module

The front module's task was to handle button presses and joystick inputs from the steering wheel. Some of the functionality, such as the lights, fans and horn, was handled by the front module on its own. The button presses that were meant to reach the main control were sent via CAN bus, with a frequency of 50 Hz.

3.2 Motor controller

New for this year's Shell Eco-Marathon was that "All Battery-electric vehicles must have a self-built Motor Controller". Earlier the motor controller had been bought from commercial actors, and even then earlier teams had problems with making it work. The custom made motor controller was built, consisting of an FPGA-based Engine Control Unit (ECU) developed at SINTEF and a custom made inverter designed by Eirik Heien Mo.

3.2.1 Power Consumption

The ECU had originally been designed to operate larger engines in SINTEF labs, but could also control smaller motors. Many security mechanisms were implemented on it, making it almost impossible to destroy by accident, and accurate status messages made it easy to debug when it failed. The trade-off, however, was a relatively high standby power consumption. Originally designed for larger systems where the power consumption of the motor controller is insignificant, the ECU would draw around $15W$. This was unfortunate, because the competition rules stated that the motor controllers power consumption would count towards the final result (this was not the case for the rest of the electronics in the car).

3.2.2 Operating the motor controller

The motor controller was controlled by connecting the ECU to the CAN bus, see chapter 3.5.6. A reference for desired torque could be set by sending messages over the CAN bus. Since the ECU was designed to be able to run many different motors, one could request torque on a scale from -1000 to 1000. -1000 meant maximum torque in the negative direction, 0 meant no torque and 1000 meant maximum torque in the positive direction. Between these points, the scale was approximately linear [11].

As we had decided on switching the motor controller on and off, which it wasn't necessarily designed for, we experienced its behaviour in this situation. Switching off the circuit would cause the capacitors from the inverter to output a declining voltage. That made the system shut down over the course of one or two seconds, and in this phase it would output different error messages. The cause of this was security mechanisms in the ECU, it would notice too low values in some of its reading and issue warnings. In normal operation, it would take around 5 seconds to restart.

3.3 Motors

The motor is perhaps the most important component in a car when it comes to energy efficiency. It has to be chosen to fit the operating points of the car, if the efficiency is to be high. With two very different cars, the choices fell on two very different engines. One was an in-wheel axial flux machine, made and run by the DNV FF team of 2010. Another was a much smaller axial flux motor from AXI, designed originally for remote controlled air planes.

3.3.1 The in-wheel axial flux motor

The motor used in UrbanConcept was an in-wheel permanen magnet synchronous machine[11], see figure 12. It was built for the DNV Fuel Fighter in 2010, and improved upon in 2011. Test results from 2010 claims the motor had an efficiency of 90% under optimal conditions[**?**].

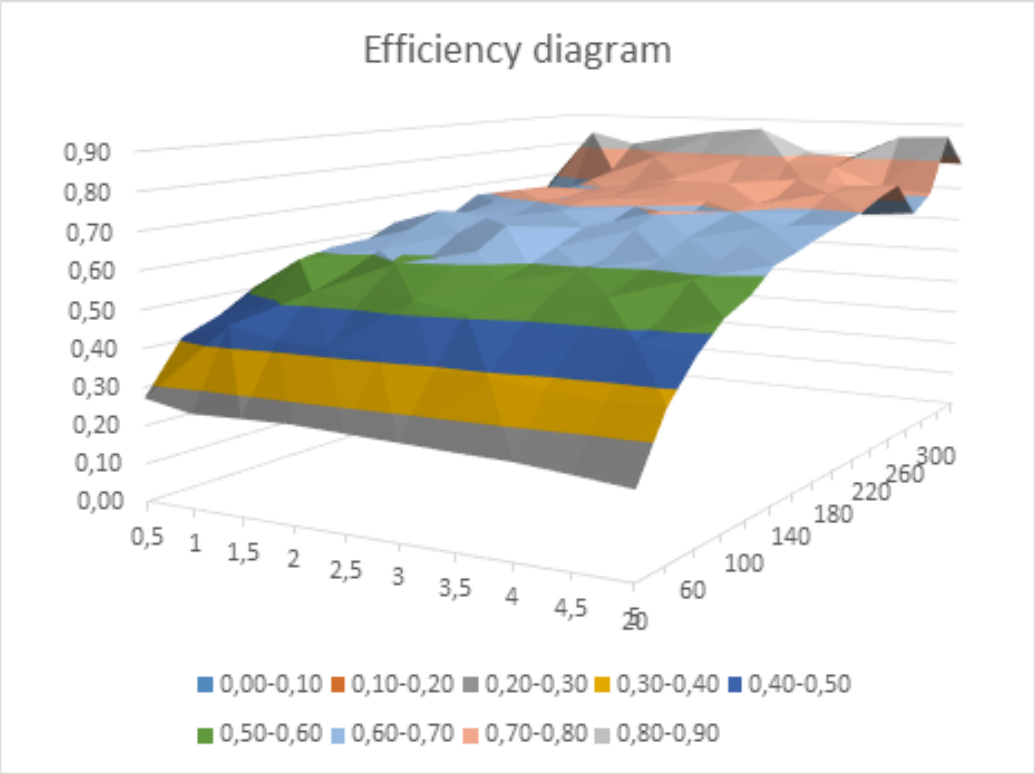

Figure 11: Testing done this year in lab showed a trend of better efficiencies at higher rotational speeds, while torque did not seem to matter as much. Nm on the x axis, RPM on the y-axis and efficiency on the z-axis. All credits for this plot go to Eirik Heien Mo[11].

Figure 12: The in-wheel motor for the UrbanConcept, photo by Eirik Heien Mo.

3.3.2 The AXi motor

The AXi 5360/20 Gold Line motor is a brushless three-phase motor designed for flying remote controller planes of up to 15 kg. The manufacturer claims the motor has an efficiency of up to 94%, and a high efficiency throughout all of its operation range. The motor was connected to the wheel with a 1:10 gear ratio, to match its most efficient ranges with the cars running speed. Testing had shown its optimal efficiency was foun around 2000-2500 RPM, while we would normally be running between 20 and 30 $\frac{km}{h}$.

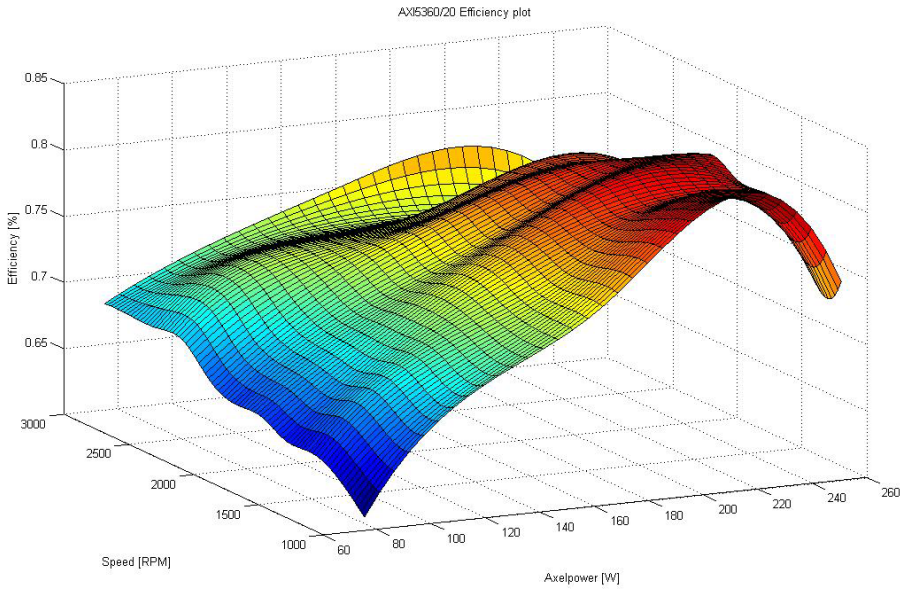

Figure 13: Torque-Speed characteristics of the AXI motor used in the Prototype car, from [7]. The efficiency is notably higher at 200-240W axelpower, which accelerates our car quite rapidly, in comparison to the axelpower needed to maintain cruising speed of $\frac{km}{h}$, which is less than 80W.

3.4 Electrical system

A very detailed explanation of the entire electrical system in both cars is outside the scope of this text, but several points were of notable importance to the main control unit.

3.4.1 The solar cell panels

The Shell Eco-Marathon rules state that in the battery electric vehicle class, up to 20% of the total energy consumed by the propulsion system may be subtracted upon finish, given that a sufficient amount of energy was produced by solar panels. Thus, solar panels are a must if one wants to succeed. A

total of 48 solar cells were installed on the prototype car, in parallel rows of 12, connected to a Maximum Power Point Tracker (MPPT[2]) as seen in figure 14. A similar system was already installed in the UrbanConcept.

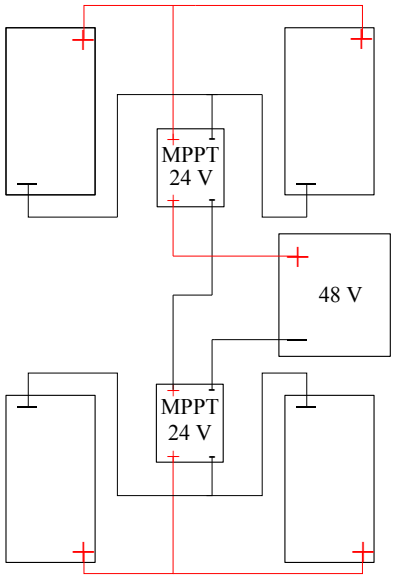

Figure 14: Block diagram of the solar cells connected to the two MPPT's, providing enough voltage to charge the 48V battery

The tuning of the MPPT's was left to Ole and Vebjørn. In order to keep the output voltage sufficiently close to the desired voltage, the MPPT's had to be tuned for each day's weather conditions, and even the sun's strength at given times of day. A voltage that was too high, would trigger security mechanisms in the BMS or the ECU, which was likely the problem in our first run.

[2]An MPPT's task is to control the load connected to the solar cell panels that gives the maximum possible power output.

3.4.2 Battery and BMS

3.5 CAN bus

3.5.1 CAN basics

The Controller Area Network is a message-based network protocol designed for automotive systems. Each node connected to the bus is allowed to send and receive messages if the bus is idle, i.e nobody is sending. If two nodes try to send at the same time, the one with the lowest ID number is allowed to send first. This gives a CAN network a hierarchy, messages with a high priority won't have to wait for less important messages. The format supports bitrates up to 1Mbit/s, in our case we used a baud rate of 500,000.

Each message contains an identifier and up to eight bytes. The sender does not define who receives the message or what it means, only its identity. Because of this, any number of nodes may choose to read messages of any identity. It also makes it easy to add and remove nodes from the network. This also ensures that a message is either accepted by all nodes in the CAN network, or no nodes in the CAN network.

Because safety often is important in automotive vehicles, the CAN bus protocol implements many safety features. Error detection is performed by the CAN bus system by performing a cyclic redundancy check, and CAN frame acknowledgement. This ensures data consistency: either all nodes accept a message, or no nodes accept the message.

3.5.2 The CAN bus in the DNV GL Fuel Fighter

The Controller Area Network bus worked both as power source for the electronics, and as a message system. Each component on the CAN bus was connected to the network via an RJ11 cable, and each end was terminated with a 120Ω resistor. Table 1 contains the different CAN frame protocols for messages sent to the main control unit. Note that the bytes are numbered from most significant to least significant, i.e the latitude should be read as

Lat1	Lat2	Lat3	Lat4

.

Sender	ID	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
GPS position	40	Lat4	Lat3	Lat2	Lat1	Lon4	Lon3	Lon2	Lon1
GPS velocity	41	Spd2	Spd1	Dir2	Dir2				
Front module	30	But1	But2	JoyX1	JoyX2	JoyY1	JoyY2		
ECU1	11	Chk	ID	Sig1	Sig2	Trq2	Trq1	RPM2	RPM1
ECU2	11	Chk	ID	Sig1		Sta2	Sta1		
Hand brake	26	Brake							
Accelerometer	45	AcX4	AcX3	AcX2	AcX1	AcY4	AcY3	AcY2	AcY1

Table 1: CAN bus protocol

3.5.3 GPS Position

The GPS position was sent in latitude and longitude, as two IEEE-754 floats[3]. The GPS would transmit frames at 1Hz, but would often send positions that were obviously wrong. These had to be discarded, and would usually appear two or three in a row. The reliability of the GPS position was therefore questionable, but it was still usable with limited accuracy as explained in chapter 5.7.1.

[3]The IEEE-754 is a 32 Bit format for representing floating point numbers. It is used by all modern CPUs

3.5.4 GPS Velocity

The speed (Spd, in knots) and direction (Dir, in degrees) read by the GPS were coded as unsigned words[4], and faced the same problems as the GPS positions they were based upon. As they were calculated from a simple distance algorithm between last and current GPS position, they were highly unreliable at low speeds because of inaccuracies in the GPS. The result between 20 and $30\frac{km}{h}$ where the GPS speed was used, was somewhat better. The deacceleration was slow, and so three or four seconds of delay was acceptable when presented in real-time to the driver. This combined with noise did however make the GPS speed measurement unsuitable for automatic decision making. Many times, the accuracy was also too bad to be used whatsoever.

3.5.5 Front module

The front module transmitted the current joystick position (JoyX and JoyY), and all button presses (But1 and But2). Note that each button byte contained binary coded buttons, however only But2 contained information relevant to the main control. But1 contained fans, lights and so on. The protocol for But1 was

Reset	Unused button	Joystick button	-	-	-	-	-

3.5.6 ECU1 and ECU2

The status messages sent from the ECU were always replies to a message sent to the ECU by the main control unit. The checksum (Chk) was a built-in security measure that was not utilized, that simply added all bytes in the rest of the frame, as validation that the contents were correct. The ID was the ECU's ID, always 24. Sig1 and Sig2 were confirmations on the

[4] 16 Bits numbers coded in two's complement

message ID's that were requested by the MCU, in the case of ECU1 the torque (Trq) and the engine RPM, while ECU2 requested to read an internal status message (Sta). These, and a more thorough explanation of the ECU CAN bus protocol can be found in [4].

3.5.7 Hand brake

The hand brake CAN frame simply implied whether or not the hand brake was currently being used. As the hand brake was an emergency option, it was safer to automatically stop the motor from providing torque if it were to be used.

3.5.8 Accelerometer

IMU measurements from an IMU located on the GPS module were also transmitted on the CAN bus, as two floats. These were meant to provide the MCU with centripetal and longitudinal accelerations, but turned out to be too noisy. Apparently this was a known problem with that particular IMU model, and another solution was used for measuring acceleration.

3.6 Steering wheel

The 3D-printed steering wheel (figure 15) contained all inputs necessary for the driver to control the vehicle, while also enabling the driver to turn the front wheels. Three buttons gave the possibility to switch fans on and off, sound the horn, and reset values such as time, position and driving mode. The joystick decided on torque (up and down) and driving mode (left and right).

Figure 15: The steering wheel in the Prototype car. Buttons at the upper left, joystick at the upper right.

4 Hardware and software choices

It was a recommendation from the DNV FF Team of 2013 that we changed main control unit this year. Jostein, who was last year's electrical and cybernetics team all on his own, recommended I used one of the boards from National Instruments' sbRIO (Single Board Reconfigurable Input Output) series. A sponsorship deal was quickly agreed upon with one of their representatives: we received two sbRIO's and all required LabVIEW software in exchange for logos on the car.

4.1 The National Instruments sbRIO-9626

The choice fell upon the sbRIO-9626, the newest and most powerful of the NI sbRIO's. It is a multi-purpose embedded control and acquisition device, custom made for real-time processes. It features a real-time processor, a user-reconfiguarable FPGA with 96 digital I/O-ports, USB, Ethernet, CAN bus, RS-232, FTP, analog IO, 512MB of storage and more. All of this made the sbRIO-9626 a good choice for a project that required a lot of prototyping, and provided flexibility when solutions were to be made on the fly.
There are two options for operating the sbRIO-9626. One can connect it to a computer running LabVIEW, through an Ethernet connection. This makes it possible to examine data flow in the program in real-time, while the actual program is running on the card and not the computer. This is particularly convenient when prototyping and testing new functionality, and is how most of my work was carried out. It's also possible to upload the code to the card as a standalone real-time application which is what was done when the car was racing.

4.1.1 FPGA

The sbRIO-9626 provides a powerful FPGA from Xilinx, that can be programmed easily through a LabVIEW interface. It enables the sbRIO to perform heavy calculations without fear of overloading the real-time processor, and to read DIO at a very high rate. It also reduces the power consumption of the system, as much of the cost is paid "up front", when the FPGA is programmed.
On the downside, however, all the DIO were required to be read by the FPGA. This meant that any changes to DIO ports and functionality would require a reprogramming of the entire FPGA, a process that took around 5

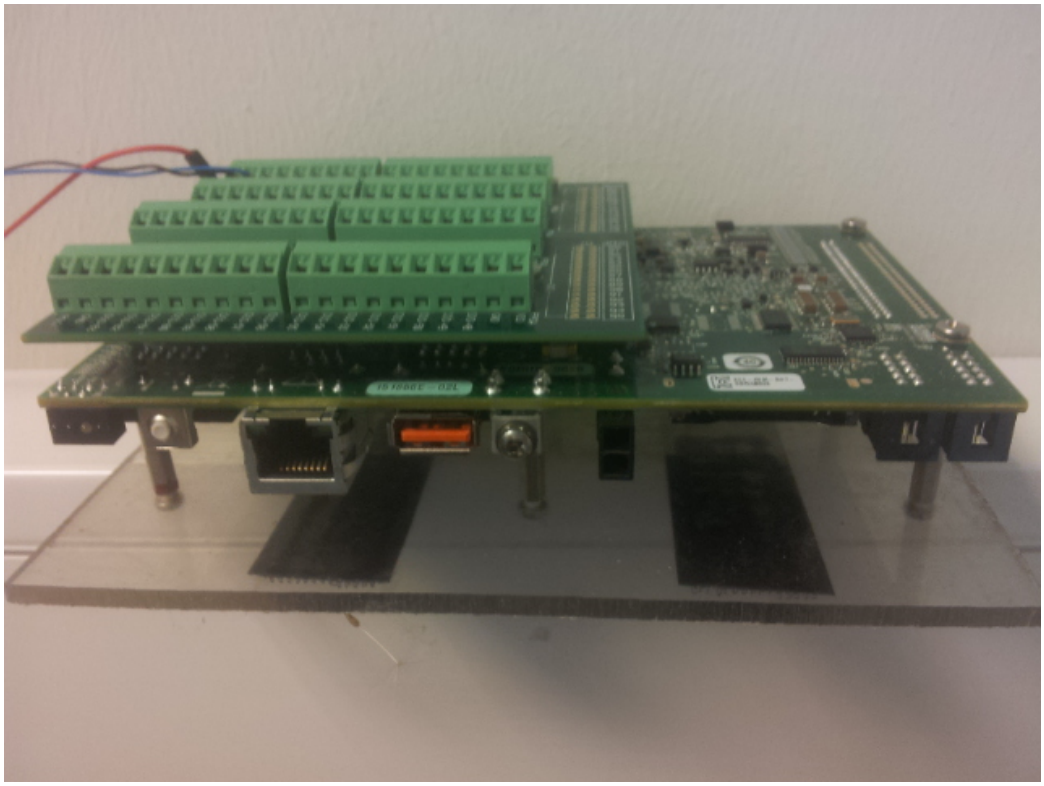

Figure 16: The NI sbRIO-9626 with Rio Mezzanine Card mounted (for DIO with the FPGA).

minutes the first times it was done, but as the FPGA had been compiled more and more times, as long as 20 minutes. 20 minutes is a long time for a minor change, when time is an issue.

4.1.2 Analog input/output

The sbRIO-9626 also provides 16 $\pm 10V$ tolerant 16 Bit AIO ports, connected directly to the real-time processor. According to the manual[3] this results in a resolution of $302\mu V$, more than enough for simple tasks the analog accelerometer would be used for.

4.1.3 CAN bus

The CAN port uses an NXP-PCA82C251T embedded CAN transceiver, supporting baud rates up to 1Mbps. It provides ground, CAN High and CAN Low pins, as well as an optional shield pin to allow connecting to shielded CAN networks. The CAN frames are transmitted directly to the real-time processor.

4.1.4 Drawbacks of the sbRIO-9626

While the sbRIO-9626 provides a lot of flexibility, it comes at a cost. The start-up time of the system depends on the software it is loaded with, but it is 50 seconds at the lowest. This eliminates restarting as a valid bug-fix during a race, as even 50 seconds is way too long a wait to be allowed to continue racing.
It is also a quite big thing, the sbRIO-9626. When we compared our electrical compartment to the ones of other high-performing Prototype cars in Rotterdam, ours stood out as big. Space and weight could definitely be saved by reducing its size.. This can be attributed mostly to the ECU and inverter, but the sbRIO is still bigger than it needs to be, considering what its tasks are. It has a lot more functionality than what is used, and 96 DIO ports are a bit of an overkill.

4.2 LabVIEW

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a development environment for the visual programming language known as G. The programming language G represents is known as a dataflow programming language[5]. The structure of the program is represented by block, connected by wires that transport data from one block to another. A block

may execute once all its data are available, and since this may be the case for several blocks at a time, parallel execution is easily achieved in G.

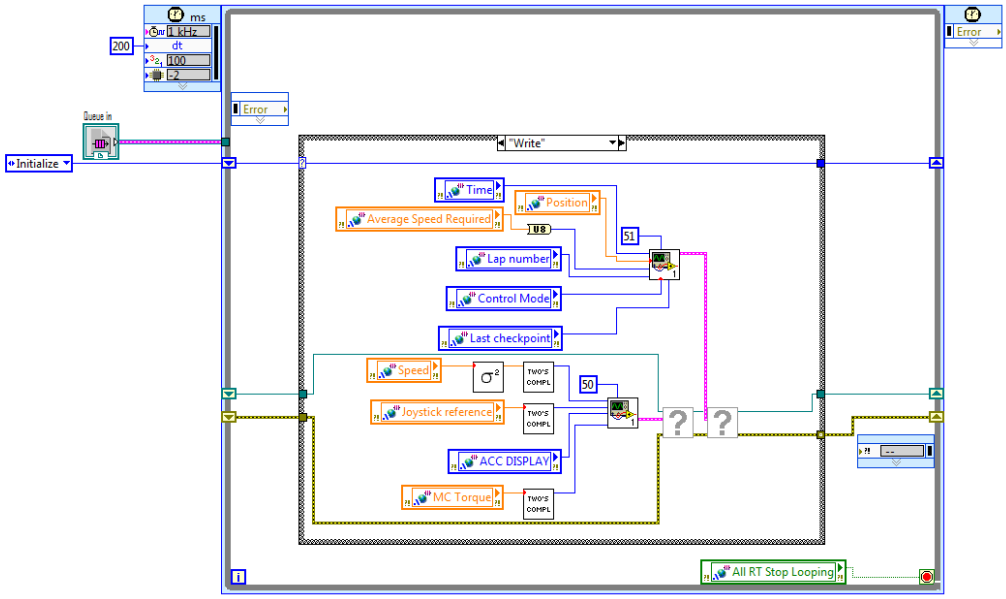

Figure 17: Example of LabVIEW block programming, here represented by the data sent to the display module.

Along with parallelism, LabVIEW also provides the user with a user interface for the program by default, as a front panel is programmed alongside the block diagrams that decide the functionality of the program. LabVIEW also has a very large library of built-in functions, which makes some tasks that usually require a lot of programming to be easily implementable with a single block. Many add-on packages also exist that implement for example real-time programming functionalities or more theoretically advanced control engineering blocks.

5 Control system

Most of my time on this project was spent on developing the main control unit. The goal was to provide the necessary functionality for the driver to operate the car, and to simplify running the vehicle in an energy optimal sense. As a side objective came the roll-over sensor implemented by the accelerometer, intended to be a safety functionality. This was made to be used as an argument in our application for the safety award, which we required to apply fro by our main sponsor DNV GL.
In this chapter I will describe how the MCU worked, why certain choices were made, and i will explain the system in such a way that it may be reused by the next team, should they wish to do so.

5.1 Driving Strategy

The most important objective was to ensure that the car would actually finish the race within given constraints, without putting the driver at increased risk. On top of this, as explained in 1.8, it was desirable to make a system that enabled the driver to drive close to optimally in terms of energy efficiency.

5.1.1 The MPC approach from the project report

In [6] I examined using an MPC based control algorithm that took into account air drag, mechanical friction and start/stop objectives, as well as a simple motor efficiency model. An MPC minimization problem was formulated as

$$\min_{\hat{u}(k),\ldots,\hat{u}(k+N_p-1)} \sum_{i=0}^{N_p-1} \hat{u}^T(k+j)\hat{u}(k+j), \tag{4}$$

$$
\begin{aligned}
s.t. \hat{x}(k+i) &= \mathbf{A}\hat{x}(k) + \mathbf{B}\hat{u}(k), && \forall i = 0, .., N_p, \\
\hat{x}(k+i) &\in X(\hat{x}(k), t, i), && \forall i = 0, .., N_p, \\
\hat{u}(k+i), &\in U(k) && \forall i = 0, .., N_p,
\end{aligned}
$$

where the objective of minimizing energy consumption has been simplified to minimizing the square sum of the torques $\hat{u}(k)$. Ensuring a finish within time is handled by the set of dynamic admissible states $X(\hat{x}(k), t, i)$, and the linearised model is represented by $\mathbf{A}$ and $\mathbf{B}$. Assuming that states are reasonably accurately acquired, that roads were empty and that the motor model was a good approximation, this would probably have been a good approach. However this was not the case.

The motor controller circuit was the only circuit required by rules to be powered by the main battery. In other words, energy consumed by this circuit could be viewed as a pure loss in terms of competition result. The standby power consumption of this particular circuit was assumed to be negligible in the project report, but in real life turned out to be around $15W$. For a car aiming to run at below $30W$ this must be considered a very important factor when deciding on driving strategy.

The simple motor model also turned out to be too simple. It assumed that by minimizing the square sum of the torques as in equation 4, the motor would operate in its most effective area. The truth was more complicated, as seen in figure 13. In fact, the lowest efficiency was found at low torque, while the highest was found at torques that would result in a relatively high acceleration for our vehicle. Both this, and the motor controller's high standby power consumption indicated that staying close to the required average speed of $25\frac{km}{h}$ was a bad idea. The traffic situation on the track also made planning

speed more difficult, as the position of other cars relative to our car had to be taken into consideration to avoid having to brake.

5.1.2 Pulse and Glide control

While the MPC approach was optimal under its assumptions, the different reality made it far from optimal in our case. It was necessary to drastically reduce the time spent the with motor controller circuit turned on. It was also desirable to avoid running the motor at its lower efficiency, in the low torque area. The obvious solution was to accelerate fast to a certain speed, and then switch the motor controller circuit off while rolling for as long as possible. The driver would then switch the system on again, through the MCU, and start accelerating when the speed was low enough. It was also convenient to give the driver more control over her speed and acceleration, as the track turned out to be more crowded than we previously thought. The resulting driving strategy can be summarized as follows

- Switch motor controller on
- Request constant torque until speed is sufficiently high
- Switch motor controller off
- Roll until speed is sufficiently low
- Switch motor controller on
- Request constant torque until speed is sufficiently high
- Repeat until finish.

While this can easily be done automatically, it was left to be done manually by the driver. This had several reasons, most importantly it gave the driver some flexibility in when to accelerate. Where to accelerate was a decision

best done by the driver, as it had to take into consideration cars nearby, turns nearby, remaining time and distance, and slight variations in slope of the track. The GPS that would measure speed while the ECU was switched off was also unreliable, and the ECU itself was not designed to be switched on and off frequently. This led to a lot of random error messages, and the ECU would sometimes have to be reset by the driver to work properly. Because of this, a simpler approach was used where the driver was the one who would switch the system on and off, based on data from the display and information from the team via cell phone.

5.2 Main

The structure of the LabVIEW program is perhaps best demonstrated by figure 18. After the block on the left has made some basic initializations, a message queue is initialized. The message queue is a feature that ensures certain events happen only once, by providing atomic execution. Then, eight different real-time loops are invoked, each seen as a colourful box connected to the purple message bus. These run in a parallel, mostly exchanging necessary data with each other through a library of shared variables. All continuously updated variables are handled locally in each loop, while events that need special handling are sent on the message queue to be handled by the main function. The main window is known as a VI, or virtual instrument, while the real-time loops are called subVI's.

5.3 Can handler

Several different modules sent messages over the CAN-bus that were directed to the MCU. While outgoing CAN frames from the MCU were handled by

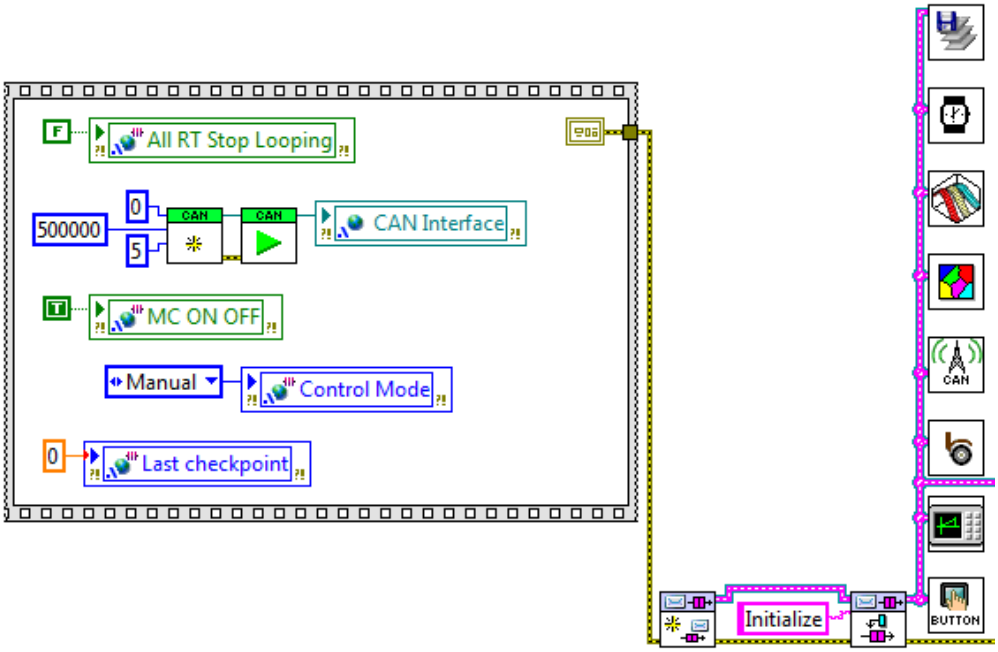

Figure 18: The main LabVIEW block diagram. After the block of code to the left has initialized the sysytem, a message queue and eight real-time loops are invoked.

different loops, all reading of CAN frames were gathered in the CAN message handler. The senders included

- ECU
- GPS
- Accelerometer (not used)
- Front module
- Hand brake

While none of these had particularly high sending rates, 100Hz at the most, their total rate of messages sent was closer to 200 per second. The timed loop that received these messages was therefore set to read the CAN bus every third millisecond, to avoid overflow in the transmit or receive queue of any of the units connected to the CAN bus.

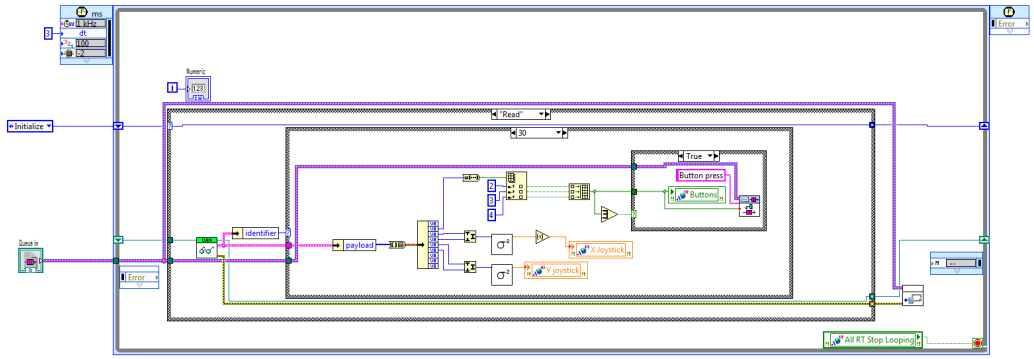

Figure 19: The CAN handler timed loop.

Each frame received by the CAN message handler would be split up into different data types. Information such as speed, GPS latitude or status messages would typically be stored in a shared variable, while button presses would be sent as a message over the message queue.

5.4 Motor control

The motor control loop handled the creation of CAN frames directed to the ECU. Each time the ECU was started anew, it was necessary to initialize some of its internal variables. The loop would therefore send these messages continuously until the ECU answered back. Upon successful initialization, it would switch from initializing to operation.
The motor controller loop sent CAN frames at 50Hz to the ECU. It would

switch between requesting a set-point for torque and requesting RPM and torque reference, or setting the engine in drive and requesting a status message.

The motor controller loop, while in operation, had three different modes. While in manual mode the driver could control the torque set-point with her joystick. In auto-drive mode, she would immediately receive maximum torque if she pushed the joystick upwards. As seen in figure 13 the engine was more effective in the higher torques, and this allowed her to avoid having to move through lower torques to get to maximum. It also allowed fast moving into auto-stop mode. Moving to auto-stop mode would switch of the entire motor controller circuit using a relay, in order to save energy while rolling.

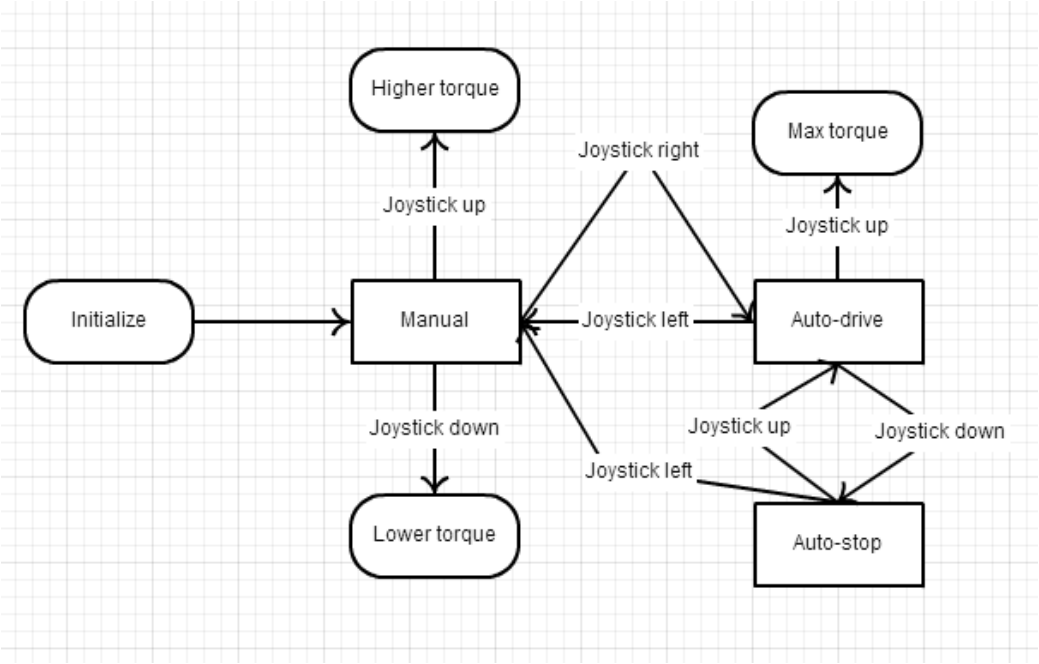

Figure 20: A state-machine representation of the driving modes. The ECU is switched on in Manual and Auto-drive, and off in Auto-stop.

5.5 User interface

The user interface block was in charge of reading the handling readings from the joystick. It would loop at 20Hz, fast enough to appear instant to the driver, slow enough to not impose an unnecessarily large load on the system. When in manual mode, it would simply add up the vertical joystick value from the front module. The joystick sent a value in the range ± 512, and the user interface loop would read these every 50 ms, and add them together. This number was scaled by a constant, to give the driver some sensitivity in how fast or slow she wanted to move the set-point of the torque. If the joystick was untouched or very close to middle, the set-point would slowly drift towards 0. This was a requirement imposed by the Shell Eco-marathon employees during technical inspection.
When in auto-stop or -drive, the functionality of the joystick was switched to being simply up, down, left or right.

5.6 FPGA

Given the LabVIEW framework for connecting real-time applications and FPGA applications, a dedicated subVI for communicating between the two layers was necessary. This subVI, through an underlying FPGA-VI, read values from an accelerometer and enabled the real-time application to switch the motor controller circuit on and off through a relay.
The accelerometer was used to measure centripetal acceleration. The mechanical engineers calculated that the car should not exceed a centripetal acceleration of $\frac{1}{2}g$, or approximately $4.9\frac{m}{s^2}$, in order to be safe from rolling over. The centripetal acceleration was read and low-pass filtered by the FPGA-VI, and scaled to fit the interval ± 128. This was sent to the display module, which would interpret the value as an acceleration to either side,

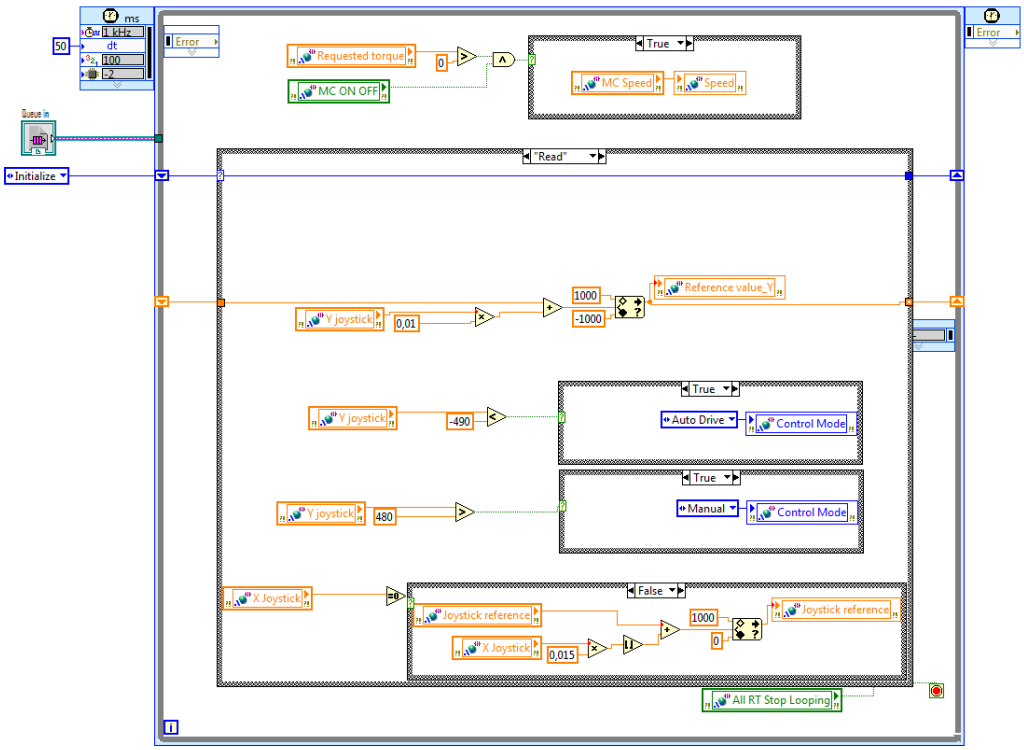

Figure 21: The User Interface subVI.

depending on the sign of the value. This was then represented on the screen as a bar, where a centripetal acceleration of $\frac{1}{2}g$ would make a full bar, indicating that the car was turning too fast.

The relay was controlled by a $3.3V$ DIO on the RMC, that was only accessible through the FPGA. The relay worked as an electric switch, which allowed power to pass through if the DIO was high, and opened the circuit when the DIO was low.

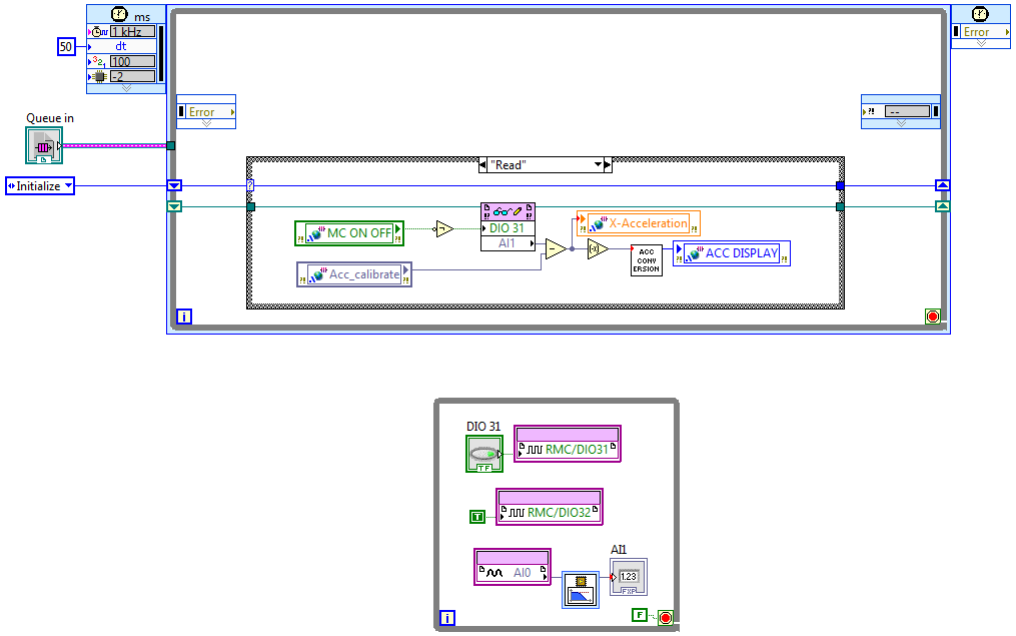

Figure 22: The FPGA subVI, and the loop running on the FPGA.

5.7 GPS

The GPS module provided two different messages, one for position and one for velocity. A checkpoint system was implemented to use the position for something meaningful, while the speed was simply stored in a shared variable.

5.7.1 The checkpoint system

In order to know the remaining distance to travel, it is necessary to keep track of the distance already travelled. Combining this with a timer, it is simple to know which average speed is required to finish the race on time. However, integrating speed to measure distance has a tendency of drifting, and combining this with the unreliability GPS measured speed, it is neces-

sary to implement a more robust speed system. As the track the car would use was known, a simple checkpoint system could be implemented. This was used to update the distance travelled six times each round, as well as keeping track of lap number. As the driver did not rely on very accurate distance measurement, this provided sufficient accuracy.

After having decided six checkpoints, and recorded these with the GPS, these positions were hard coded into the GPS subVI. The GPS would then trigger a message when the car came close to that point, and if this was a different checkpoint than last, the position would update after the formula $C_{pos} + N_r D_r$, simply the position of the checkpoint plus the lap distance times number of laps.

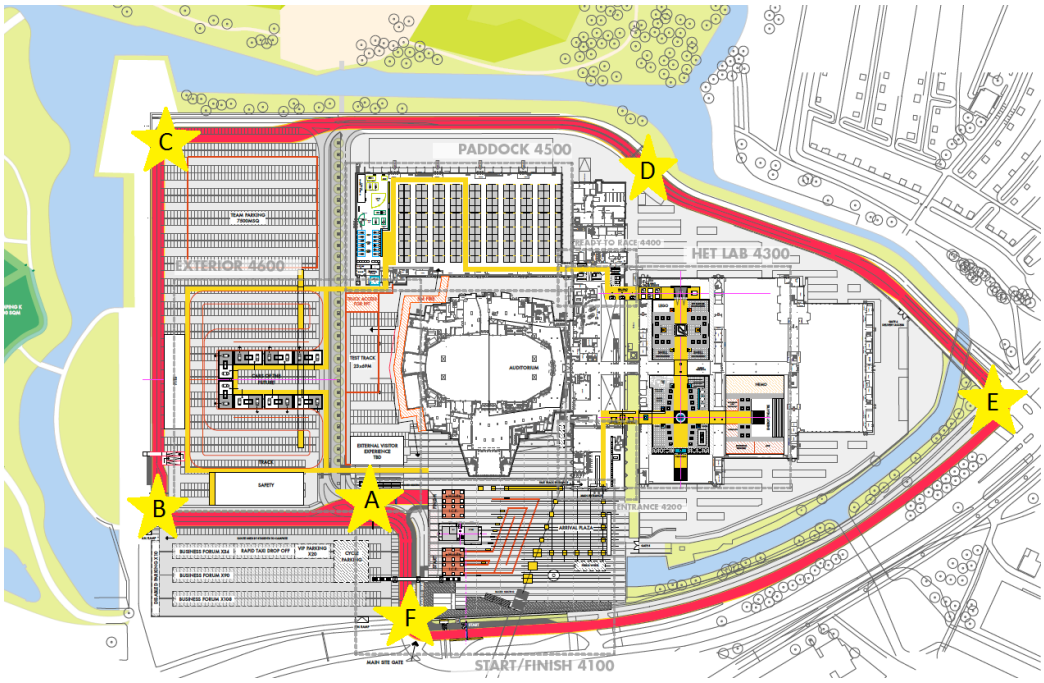

Figure 23: The six GPS checkpoints on the Ahoy! track. A marks the start line.

In table 5.7.1 the GPS coorindates are presented in degrees north and degrees

Checkpoint	**A**	**B**	**C**	**D**	**E**	**F**
Latitude	51,8836°	51,8835°	51,8813°	51,8814°	51,8832°	51,8844°
Longitude	4,48917°	4,49154°	4,49148°	4,48688°	4,48306°	4,48905°

Table 2: GPS checkpoints

east. The GPS VI would continuously check the distance between its current position and each of these six checkpoints, and trigger messages when within 50 meters of a point. For calculating distance, a simplified formula was used, that has sufficient precision over small distances (in a GPS perspective). The formula makes use of what is known as the equirectangular projection and Pythagoras' theorem.

$$\Delta\text{Position} = R_{earth}\sqrt{(\Delta\lambda cos\frac{\phi_1 + \phi_2}{2})^2 + (\Delta\phi)^2} \tag{5}$$

where R_{earth} is the Earth's radius, $\Delta\lambda$ is the difference in latitude in radians between the two points, and ϕ_1 and ϕ_2 are the two longitudes. It ignores the curvature of the Earth's surface, but includes a small correction for the Earth's elliptical shape.

5.7.2 The GPS speed

The GPS speed was, as discussed earlier, not very reliable. Due to its nature of simply calculating distance between points and dividing by time, it was not very useful when travelling at low speeds. Several of the speed measurements would simply be outliers, and had to be discarded. However, as the motor controller was turned off most of the time, the GPS speed was used in its place. The back wheel, which was where the motor was applying force, had a flywheel installed. That meant that the motor could only be assumed to be turning at the same rate as the wheel when the motor was applying torque,

or else the wheel could be moving faster. As a result, the GPS speed was presented on the screen not only whenever the motor controller was off, but whenever no torque was requested from the engine. This worked to some degree, but I recommend next year's team find a different solution.

5.8 Logging

A simple logging function logged all data that could be relevant, at a frequency of 1Hz. The data was stored as a text file in the sbRIO's memory.

5.9 Display

The Display VI sent relevant data to the display module over the CAN bus. Since the frame rate of the display was only set to 5 fps, the display module was set to send at 5Hz. 10 different pieces of data were sent to the display:

- Time
- Position
- Average speed required to finish on time
- Lap number
- Control mode
- Last checkpoint
- Speed (GPS or RPM-based)
- Torque requested
- Centripetal acceleration
- The ECU status message

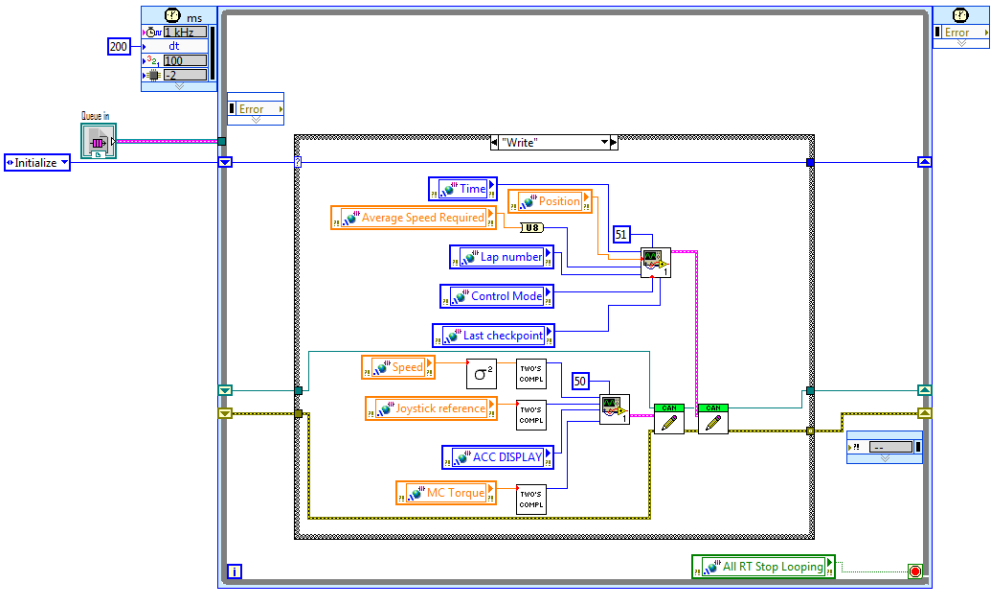

Figure 24: The Display VI.

5.10 Time and position

The Time and position VI provided a simple timing function, and calculated distance to be presented on the display. It also calculated the average speed required to finish the race on time, by dividing estimated remaining distance by remaining time.

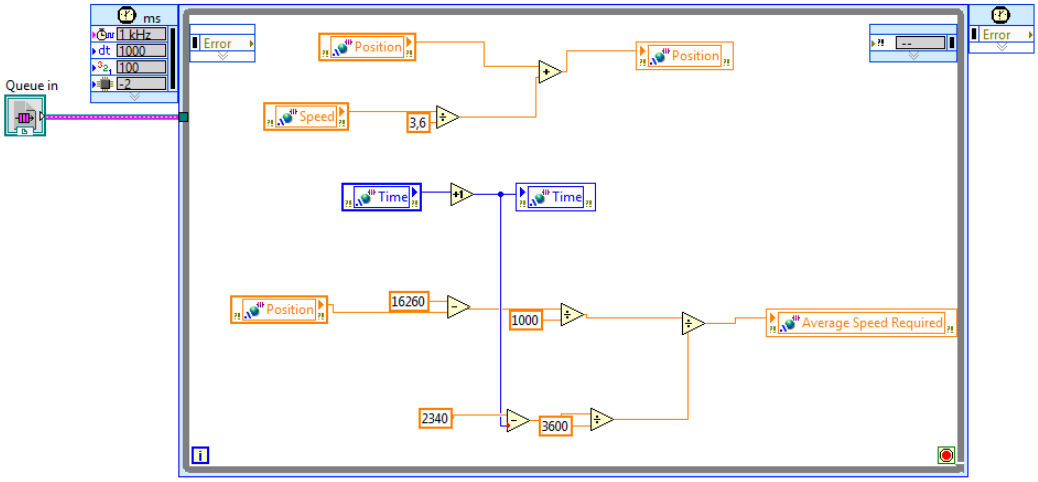

Figure 25: The Timing and position VI.

6 Competition

This year's Shell Eco-marathon in Europe hosted more than 3,000 competitors bringing over 200 self-made cars. Most of the teams competing are from Europe, but quite a few are also African. Education levels range from high school to university, and the youngest teams usually have professors participating in the work.` On May 12 most of the DNV GL Fuel Fighter team left for Rotterdam. The two cars were already on their way down, carried in two vans by the mechanical team. The Shell Eco-Marathon 2014 lasted for one week, 12th of May until 19th of May. We unpacked our things on Tuesday and started doing the final adjustments of our cars. On Wednesday, we completed the technical inspection. On Thursday we had our first test runs, and some testing was also done on Friday, while all races were run on Friday, Saturday and Sunday.

Figure 26: Most of the participants in the Shell Eco-marathon. All Rights Reserved Shell Eco-marathon 2014.

6.1 Ahoy! Arena in Rotterdam

The Ahoy! Arena is a convention center located in Rotterdam of more than $30,000m^2$. Upon arrival each team is assigned a paddock in the main hall where they can work on their cars and prepare for the technical inspection.

6.2 Testing UrbanConcept

We had very minor testing beforehand with the UrbanConcept car, but the car had raced before and the mechanical parts of the vehicle seemed fairly reliable. We were however aware that there was some trouble concerning the encoder.

Figure 27: The paddock area. All Rights Reserved Shell Eco-marathon 2014.

6.2.1 Encoder problems

The encoder's task is to keep track of the angle and rotational speed of the in-wheel motor. If the ECU assumes the wrong angle between stator and rotor, it will set up a misplaced magnetic field. This leads to a weaker, less efficient engine, and may even start turning the car backwards. In our case, loss of position mostly meant no movement whatsoever.

Many things had been tried, but since the car had been designed for another motor, the encoder solution was never really very reliable. It would slowly drift away from the ideal angle measurement, because of slipping between the shaft and the motor. It had to be tuned each time the car was moved, because of the way the dampening in the wheel changed the angle of the encoder relative to the wheel. Additionally, the wheel had to be spun around half a round upon each start-up to find position, making it necessary to keep the system running when stopping the car if one wished to start again.

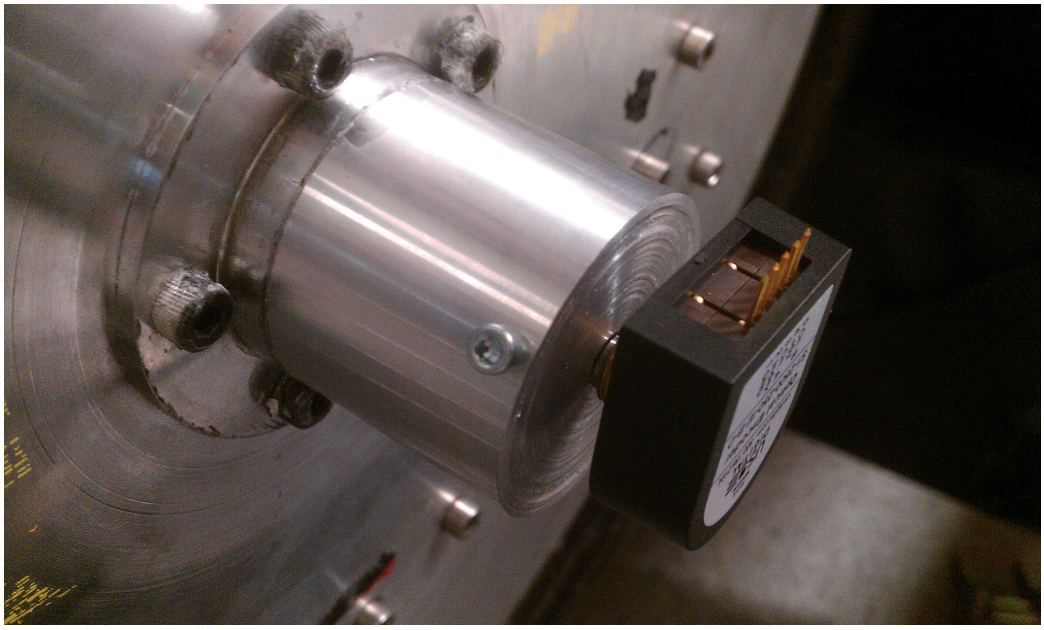

Figure 28: The encoder fastened to the wheel.

6.2.2 Test racing

Finally being able to test the car in the real world, on the actual track with the actual drivers sitting inside revealed several important points of improvement:

Driver training

The drivers needed a better understanding of the car system in order to operate it reliably. The long start-up time of the motor controller system hadn't been explained well enough, and the encoder's functionality hadn't really been understood. Also, the user interface state-machine, shown in figure 21, wasn't as intuitive as I had assumed while designing it. More off-track driver training and explanation had to be done, perhaps most importantly for them to be able to explain to me what they experienced over the phone, so that I

could provide meaningful assistance.

GPS speed

The GPS speed measurement had been finished late, and as the cars had never been run with it functioning, the reliability when driving was questionable. The speed, according to driver, seemed to be working reasonably well, but would often not update for three or four seconds.

Encoder

While the encoder problems were known beforehand, their actual race impact had yet to be tested. The driver sometimes forgot to turn on the motor controller early enough before a stop, meaning the encoder wouldn't be turned on for the final turnings of the wheel. In these occasions the car wouldn't be able to start again, which would have cost us the race if it happened on a real attempt. The slipping of the encoder shaft also seemed to be worse when asking for maximum torque, and it was decided to get a complete run with the UrbanConcept with motor controller turned on at all times, using only manual mode. While this would be suboptimal, the test runs we had did not seem promising enough when switching motor controller circuit on and off.

6.3 Testing Prototype

The Prototype had a more reliable system electrically. The main reasons for this was the different encoder solution, which was working as it should without need for tuning. While it also needed turning of the wheel to initialize

position after start-up, the Prototype wasn't required to stop once between each round. That meant the car would always be rolling whenever the encoder was restarted, and so this was not an issue with the Prototype.
The car was initially tested without solar cell panels running, as the person responsible for these would not arrive until the last day of testing. While the driver problems were more or less the same as in UrbanConcept, the motor controller and encoder seemed reliable. The start-up time of the motor controller circuit was stable at around 5 seconds.

Display

During testing it was revealed that the latest update to the display had sent the update rate down to about 1Hz. There was also problems with the timer disturbing other values once time passed 1000 seconds, and so the display module had to be reprogrammed.

Joystick

The joystick suddenly started behaving a bit differently from what we were used to in the lab, with more noise and some small offset. This was handled by Ole and Vebjørn by some filtering algorithms in the front module, but I also added a simple filter in the User Interface VI. The sensitivity of the joystick was reduced as well, on request by the drivers.

6.4 Racing Prototype

6.4.1 Race 1

After having walked in the opening ceremony, we were the first car to start the Shell Eco-marathon 2014. Unfortunately we failed the braking test that

was held at the start line. The hydraulic front brakes needed pumping in order to work, and this was not allowed. This was unexpected, as the car had been allowed to start test racing with the same brakes, and even passed the braking test on the technical inspection. This problem needed quite some work done to be fixed, and so we did not manage to get another attempt that day.

6.4.2 Race 2

After fixing the braking problem, the Prototype was ready for its first attempt. Very limited testing with the solar cell panels had been done, and we had needed to remove them from the circuit in one of our test runs to make the car run reliably. The sun was at its strongest during this race.

The test runs without solar panels had been going well, and this race also started out well. The driver accelerated to around $32\frac{km}{h}$, switched the system off and coasted, and restarted and accelerated again. She needed slightly less than 3 accelerations per round: she would still have quite a bit of speed in the beginning of each lap from previous acceleration.

The first few rounds were fine, but then the trouble started. The motor controller circuit normally used around 5 seconds to start, but this started to take more time. The motor controller began sending error messages, mostly the message 0x75, see [4]. We later learned from Kjell Ljøkelsøy that this was likely caused by some too high input voltage. As the rounds passed, the motor controller became less and less reliable. Simply waiting a longer period on start-up of the motor controller no longer worked, and it would require additional resets to start working. It would also shut off on its own accord, often just a few seconds after a successful restart.

This seemed to get worse and worse, and the last rounds the driver multiple times waited more than 30 seconds for the controller to respond, and almost coming to a complete halt. Fortunately, we had increased the average speed upon noticing problems, and we had a big enough safety margin for the last lap times to be higher. The resulting driving, as seen in figure 29, was however suboptimal. The maximum speed would be held for an increasing amount of time, and the car would also come to almost complete halts at times. The race was finished within time limit, and the result was $512\frac{km}{h}$. This placed us at a current 7th place.

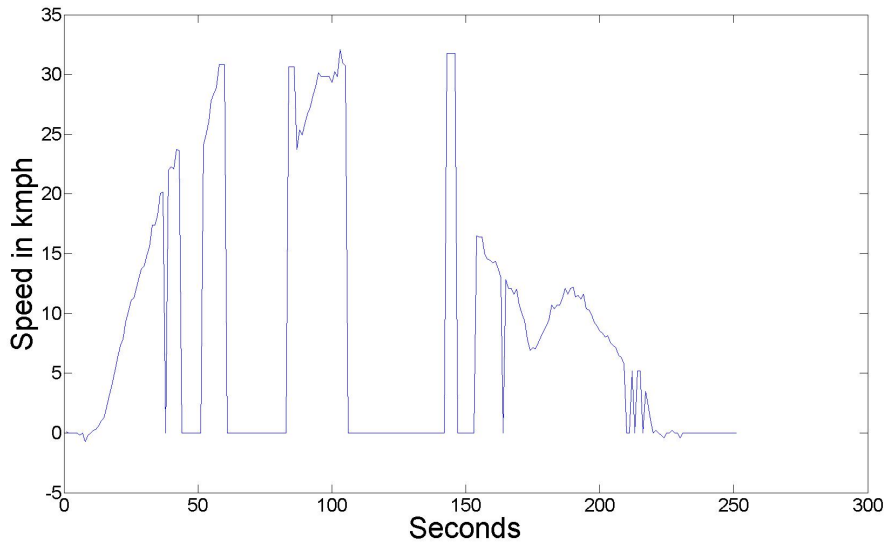

Figure 29: Speed log from run with motor controller problems. Motor controller shutting of mid acceleration, and unwanted long deceleration phase.

6.4.3 Race 3

Some safety margins on the motor controller were increased in an attempt to avoid the ECU shutting itself off due to error 0x75. The solar cell panels were tuned anew as well. I suspected solar cells to be the problem, since everything had been going well during testing, and the only new element in the system was the solar cell panels being connected.

This run was much better in terms of motor controller, but our cell phone communication failed in round 3. The driver no longer had my input on time and speed, but we moved to a different part of the track and held up signs to guide her to go faster or slower. This worked, and she finished the race with around 10 seconds left. The race was better than the first in terms of energy consumption, she reduced total consumption from 141,000 Joules to 131,000 Joules. This was, however, much earlier in the morning than last race and the solar panels didn't produce enough energy to cover the 20%. The new result was $520\frac{km}{kWh}$.

6.4.4 Race 4

The car system seemed to be working as expected in Race 3, and once the cell phones were fixed we started another race. Having spoken with the driver, we agreed she would change driving style a bit. Earlier races, she had accelerated to $32\frac{km}{h}$ and then held that speed for a few seconds. She would now switch the system of immediately at $32\frac{km}{h}$ in order to reduce the motor controller circuit's time spent running. She would also try to keep a higher speed in an area with a slight downhill, and instead go slower in areas with more turns and uphills.

The result this time surprised us all. Even though the system was working

as expected in both the third and this try, energy consumption was reduced from 131,000 Joules to 118,000 Joles just by minor adjustmens to driving strategy. This yielded 10% improvement of the result on its own, but as the solar cell panels now produced enough power, the final result was $612.8\frac{km}{kWh}$. This was an improvement of 17.6% from race 3, and ended up placing us 7th out of the 49 teams participating in the Prototype class.

6.5 Racing UrbanConcept

6.5.1 Race 1

Based on the results from testing we decided to run the UrbanConcept car in manual mode and keep the motor controller on at all times. There had been to many problems with not being able to start after stopping because of the encoder, and the team needed a morale boost after being stopped at the start line with the Prototype.

The race started out well, but it was clear that the encoder wasn't perfectly tunes. The acceleration was slower than it had been when testing with a perfectly tuned encoder, but it was acceptable. Everything seemed to be working well, until the very last part of the last round when the steering wheel broke and our driver was unable to make turns. Fortunately, she was headed straight for the finish line at this point, and we got our first successful attempt. The result was $198.7\frac{km}{kWh}$.

Figure 30 shows one round, from a complete stop to another complete stop at around 450 seconds. As shown, the driver accelerates up to $32 - 34\frac{km}{h}$ and then starts rolling. Once she starts rolling, speed measurement switches

from ECU-based to GPS-based. In this case, the GPS speed was stable at around $16\frac{km}{h}$, which was incorrect and not providing the driver with valuable information. The speed at which to start accelerating again therefore had to be decided by driver intuition and cell phone communication rather than speed measurement.

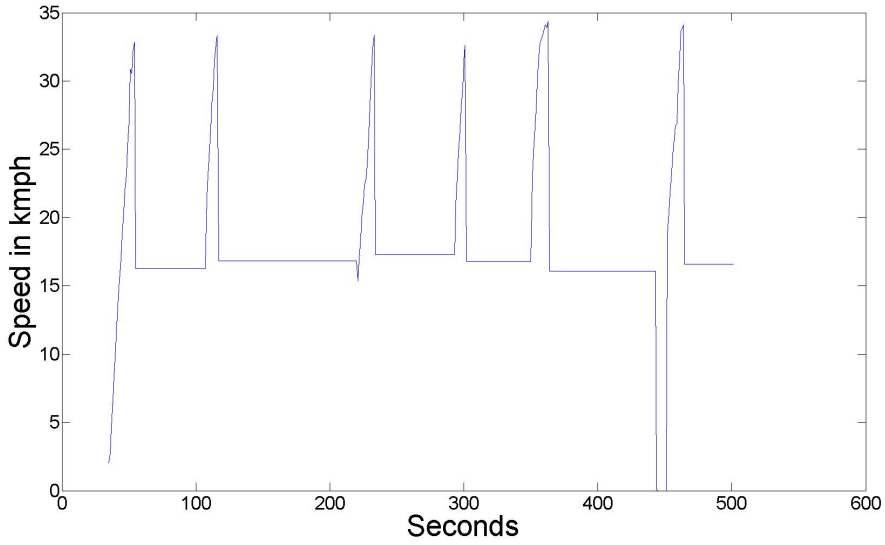

Figure 30: Speed logging from the UrbanConcept's first race.

6.5.2 Race 2

Having a successful attempt at both cars, we decided to try switching the motor controller on and off. More time was also spent tuning the encoder with the driver inside, and the race seemed to be going very well. In the fourth round however, the door flew open and the race had to be cancelled.

Figure 31: The door handle suddenly broke for unknown reasons, and it was not possible to close the door again. All Rights Reserved Shell Eco-marathon 2014.

6.5.3 Race 3

In our third attempt, there were problems with the encoder and the car never managed to build up more speed than $4-5\frac{km}{kWh}$. It was taken off track after around 50m, to be tuned again.

6.5.4 Race 4

Race attempt 4 saw the encoder problems from attempt 3 repeated, and the UrbanConcept had to cancel the race once again. The maximum number of attempts was 4, and so that marked the finish for the DNV GL Fuel Fighter UrbanConcept 2014.

7 Recommendations and conclusion

7.1 Plan versus reality

In chapter 1.7.2, I presented my initial plan for how this project would be executed. In chapter 1.8 I presented the overall goal of my project contribution. The plan was followed initially, and in the beginning of the semester I spent a lot of time recruiting the electrical team. I was fortunately successful in finding Ole Bauck and Vebjørn Myklebust for the job, I was able to focus on developing the MCU.

As seen in figure 7 I planned to start prototyping in LabVIEW early, move on to running the motor in the laboratory and then start testing the car. The prototyping and testing in the laboratory went as planned, but the testing of the car started much later than anticipated. The components needed to run the motors in the laboratory were already in place, but the printed circuit boards and custom made inverter arrived after Easter. It wasn't possible to test run the UrbanConcept outside before these parts arrived, and the Prototype was not ready to be test driven until less than 24 hours before departure for Rotterdam. Actual outside test driving before departure was therefore, unfortunately, reduced to next to nothing, and I had to test the cars standing on rigs inside.

I had also planned to implement the MPC algorithm from my project report in both the cars. As I explain in chapter 1.8 this was not implemented for a number of reasons, mainly lack of testing possibilities and that the assumptions about motor efficiency made were incorrect. This, as well as a better understanding of the traffic situation on the Ahoy!, changed my mind about the MPC controller. Not only would it be less realistic to implement because

of the more difficult optimisation problem, but it wouldn't be as optimal as my theory had shown.

Instead of implementing the MPC controller, I made a user interface that made Pulse and Glide [2] driving possible. The driver was able to switch between acceleration and coasting without much input needed, and this gave her more control over the race than an MPC would have done. This much simpler scheme also made the resulting system more reliable in contact with an unstable motor controller system, and in chapter 5.1 I argue that this approach was way more efficient than the MPC would have been.

7.2 Recommendations for the next year's DNV GL Fuel Fighter team

7.2.1 Driving strategy

From a higher level perspective, my main responsibility was that the car was driven as close to energy optimal as the physical and electrical system would allow. The decisions I made were mainly based on two factors I couldn't change: The motors' efficiencies at different torques and RPM's, as well as the high standby power consumption of the motor controller circuits. This led to a driving strategy that is suboptimal in a physical perspective. In a system where air friction is one of the main losses, it is desirable to avoid high speeds because of air friction's quadratic nature. Keeping speed constant would also reduce the amount of takeovers of or from other cars.

Obviously looking into reducing the motor controller standby power consumption is one of the most important factors, it is a pure loss. If this can

be reduced sufficiently, such as down to 1-2 W, I also recommend going for constant speed instead of Pulse and Glide.

7.2.2 Motors

If the motor controller circuit can be made to be run with a sufficiently low power consumption, a new motor should be considered for both cars. It is very important to find a motor with the right characteristics. Finding one that has its peak efficiency at the required average speed of $25\frac{km}{h}$ and the torque needed to keep this speed would definitely improve the result. It should also be considered to have an additional motor for the acceleration phase, with peak efficiency at higher torques.

7.2.3 Speed measurement

The speed measurement provided by the GPS had an inaccurate and unreliable speed measurement. It is however very easy to measure speed in other and much more accurate ways, such as installing a bike speedometer. I recommend finding another solution than using GPS.

7.2.4 Testing

One of the deciding factors for what was possible to implement this year was lack of testing. As I show in 2.2, earlier teams also report major problems that could have been avoided with more testing. If testing had started earlier, the problems with the GPS speed would have been discovered in time to find another solution. Perhaps we would also have found and fixed the solar

cell panel related motor controller problems, or the faulty door.

It is important to find time for testing, even when one has to wait for others to finish their work. I strongly recommend finding a way to test most of the system even before the rest of the team has finished their work, and to document the testing well.

7.2.5 LabVIEW and sbRIO-9626

Based on my experiences with the sbRIO-9626 and LabVIEW, I don't recommend that next year's team reuse the LabVIEW solution unless they know LabVIEW beforehand. The sbRIO-9626 is needlessly large, and the start-up time is too long (50 seconds). Most of the functionality of the card is never used, and the extra time spent developing LabVIEW programs instead of a simpler solution simply isn't worth it unless you're making a system with advanced real-time related constraints.

7.3 Conclusion

In this report I document my work on and experiences with the DNV GL Fuel Fighter team of 2014. Main control units for both DNV GL Fuel Fighter cars were developed and implemented in LabVIEW, communicating with the rest of the system on a CAN bus. The system was tested on the Ahoy! track, improved upon and successfully run in the competition.

New information about the system to be controlled made me change my approach on driving strategy. Looking at the 10% increase in distance per energy from the Prototype's third to fourth attempt, it is evident that driving strategy is a factor that can make a big impact. Based on my analysis and

experiences, I also made recommendations for next year's team concerning driving strategy, motors and speed measurement.

The Pulse and Glide approach was simple to implement, and it proved to be effective in our case. I find it to be a promising

Bibliography

[1] J. M. Harstad Bakken, *Styresystem for fremdrift av Shell-Ecomarathon-kjøretøy.* Norwegian University of Science and Technology, 2009.

[2] J. Lee, *Vehicle Inertia Impact on Fuel Consumption of Conventional and Hybrid Electric Vehicles Using Acceleration and Coast Driving Strategy.* Virginia Polytech Institute and State University, 2009.

[3] National Instruments, *OEM Operating Instructions and Specifications NI sbRIO-9605/9606 and NI sbRIO-9623/9626/9633/9636.* National Instruments, 2014.

[4] K. Ljøkelsøy, *Control system for a three-phase grid connected converter.* SINTEF Energy Research, 2014.

[5] Wikipedia, *"LabVIEW", Wikipedia, The Free Encyclopedia.* Wikimedia foundation, Inc. 16 June 2014. (http://en.wikipedia.org/wiki/LabVIEW)

[6] H. Trømborg, *Model Predictive Control For the DNV Fuel Fighter.* Norwegian University of Science and Technology, 2013

[7] B. O. Wiik, *Elektrisk Fremdriftsystem for Shell Eco-marathon PureChoice Kjrety.* Norwegian University of Science and Technology, 2008

[8] B. Gutjahr, *Energy Optimized Driving Strategy for a Shell Eco-Marathon Race Car*. Institut für Systemtheorie und Regelungstechnik Universität Stuttgart, 2012.

[9] J. S. Øverby, *Regulering og optimalisering av Shell Eco-marathon kjøretøy*. Norwegian University of Science and Technology, 2011.

[10] DNV Fuel Fighter team 2013, *Eco-marathon 2013. SEM Final Project Report*. Norwegian University of Science and Technology, 2013.

[11] E. H. Mo, *High Efficiency Electric Propulsion Systems for Shell Eco-Marathon 2014*. Norwegian University of Science and Technology, 2014.

[12] J. Cheng, Y. Davydova, P. Howlett, and P. Pudney, *Optimal driving strategies for a train journey with non-zero track gradient and speed limits*. IMA Journal of Mathematics Applied in Business & Industry, 1998.

[13] B. Asadi and A. Vahidi, *Predictive Cruise Control: Utilizing Upcoming Traffic Signal Information for Improving Fuel Economy and Reducing Trip Time*. IEEE Transactions On Control Systems Technology, 2011.

[14] J. N. Hooker, *Optimal Driving for Single-vehicle fuel economy*. Graduate School of Industrial Administration, Carnegie-Mellon University, 1998.

Appendix A

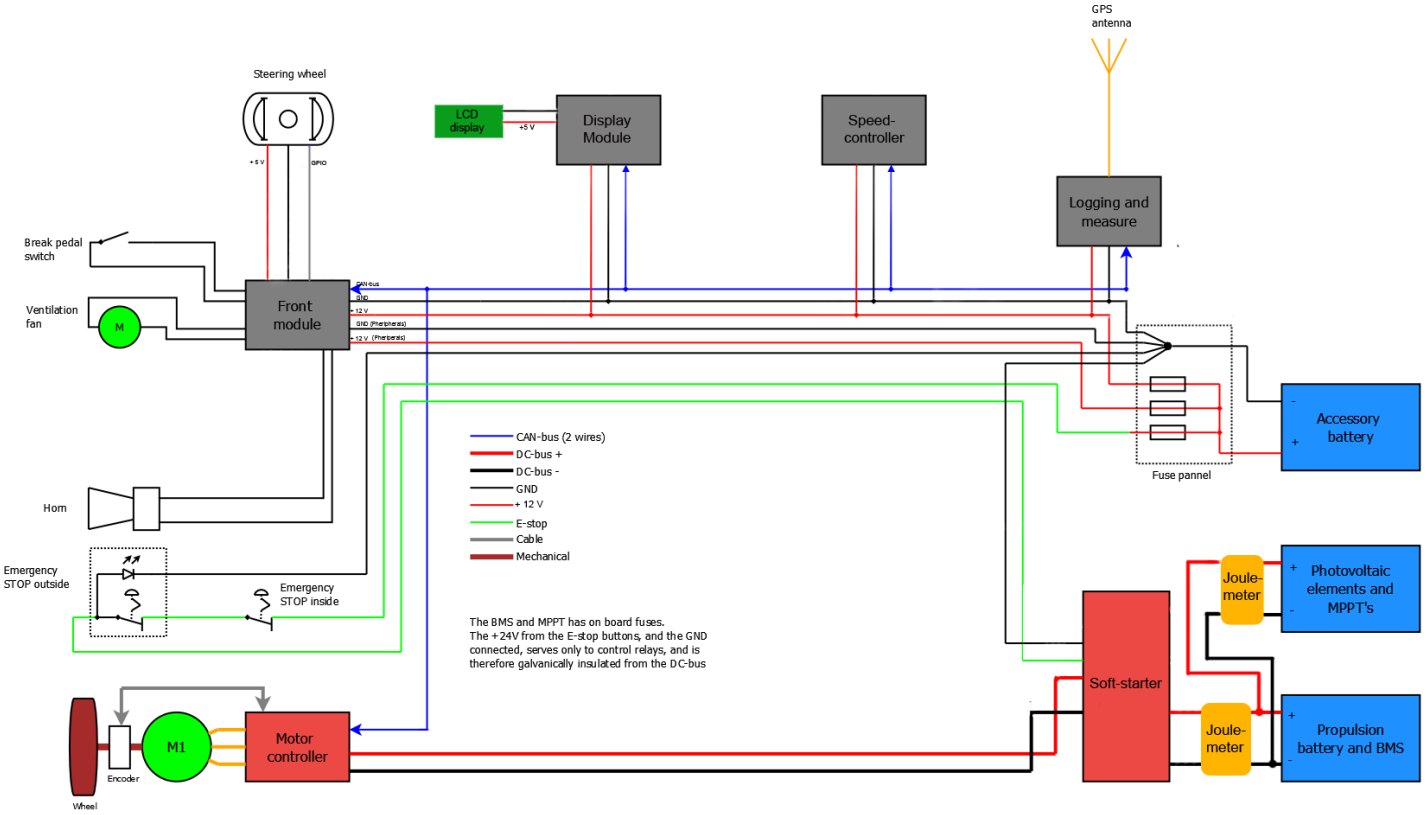

Figure 32: The overall wiring diagram of the Prototype, credits Jostein Furseth [10].

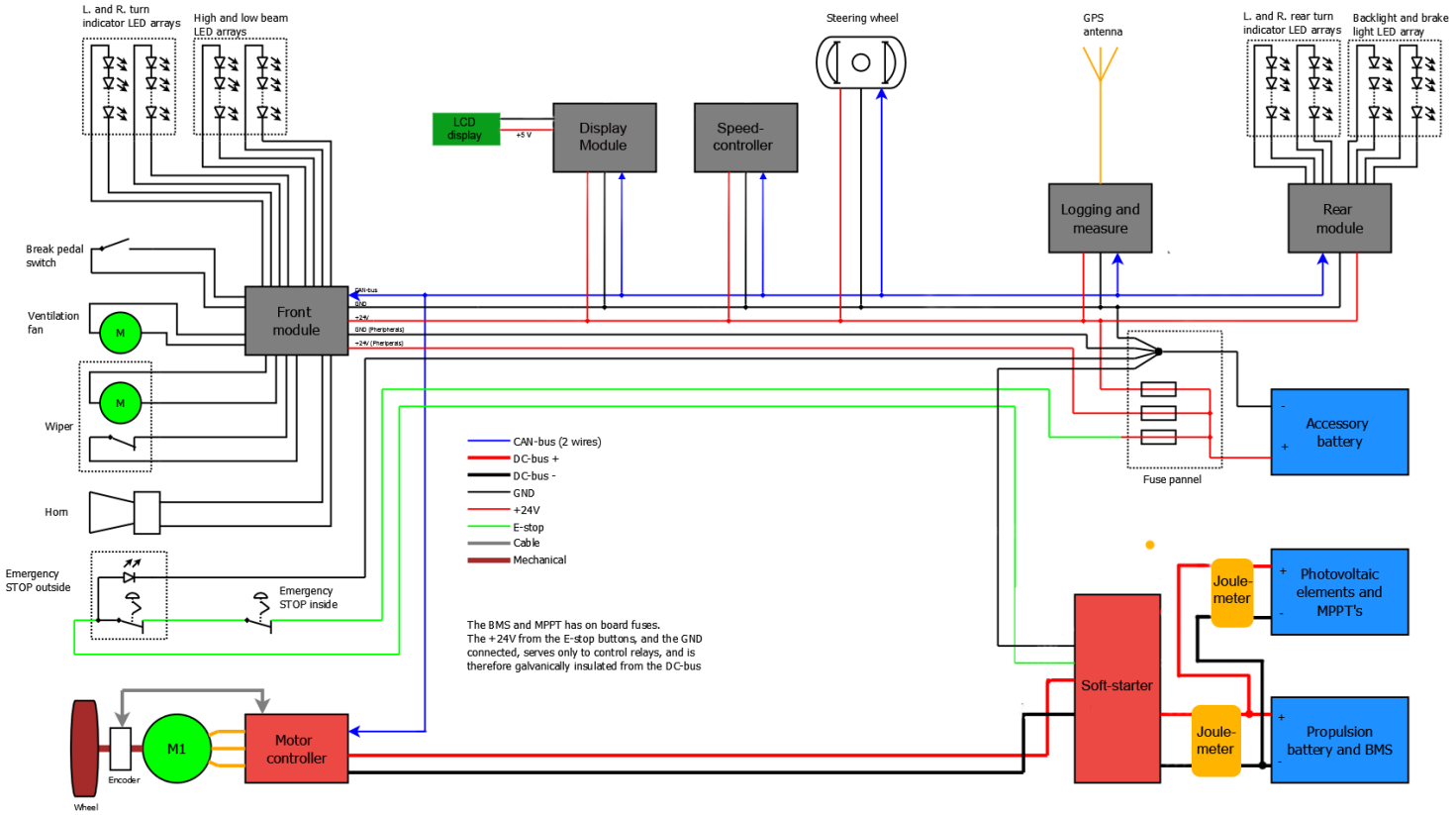

Figure 33: The overall wiring diagram of the UrbanConcept, credits Jostein Furseth [10].