Implementation & optimization of steering- and light-boards on Ecocar

31015 Introductory project - Electrotechnology 2018 GRP_01.20_MEK-1_Ecocar

Pelle Schwartz (s147170)†, Jacob Frederiksen (s163911)†,
Jakob Thomsen (s163917)†, Jonas Christoffersen (s163944)†
†Department of Electrical Engineering
Technical University of Denmark

***Abstract*— This project examines the work on a concept Ecocar – an interdisciplinary DTU student project. The Ecocar has to compete in a competition every year, in which it will be tested for mileage and new to this year; autonomous driving. This paper is about making and implementing robust control PCB's for a steering wheel and autonomous lights. The boards are built around a Teensy 3.6 micro-controller unit, that is programmed to transmit and/or receive signals, and control equipment on the Ecocar using the custom circuits on the boards. An introduction to CAN bus, the communications system between boards, is included. The code is rewritten, to make the readability and expandability easier for a smoother transition for next years team. In the end it succeeded us in making a steering board, with new and object oriented code that can control all the desired signals needed. A multifunctional lighting system was also implemented.**

Keywords*—*PCB Design, CAN bus, Micro-controllers

I. INTRODUCTION

The DTU Road Runners Electronics Project is a continuation from previous generations, where the goal is to participate in Shell Ecomarathon with a "Ecocar" in the category "UrbanConcept" race with a 1-person car. The race has certain rules[1] defined by the Shell Eco-Marathon team which the car has to comply. The project is a collaboration between a team of electrical, production and construction and mechanical engineering students, who work together on making improvements on the new generation of the Ecocar. The goal of the Ecomarathon is to drive as efficiently as possible considering the car's fuel consumption. New to this year the team is also competing in a newly established autonomous competition where the car is self driving and has to complete certain obstacle courses on the track.

A. Problem definition

How to produce a steering wheel board and implement new code for controlling, the car horn, wipers, cooling fan, lights and steering-display, with the communication form "Controller Area Network"(CAN bus), which sends and receives data from different boards in the car. Furthermore to find a solution to the noise- and display-problem, and install a visual warning system for when the car is in autonomous mode.

- How to design a PCB for the steering wheel, with buttons and a display, which is able to control different functions of the car, and display important information to the driver.
- How to interact and control CAN bus, to communicate between the boards on the car.
    - What is required for CAN bus to function between the different boards in the car.
- How to protect the electrical system in the car from shutting down, when major electrical disturbances occur in the power network.
- Which method should be used to indicate that the car is driving in autonomous mode?

II. PREREQUISITES & DELIMITATIONS

As this is a project focused on building a car together, every team working together must carefully focus on each area of the car. This team in particular is responsible for the assembly of electronics in the steering wheel. Fitting the electronics in the new steering wheel is done in cooperation with another part of the team, who 3D printed the shell with room for the electronics and correctly placed buttons. This team also works with the light board and controlling the LED-strip mounted around the car. In other words the delimitation is around the electronics and programming of the steering wheel with it's tft-screen and light board with LEDs. A natural time limit of the project was the competition at Shell Eco-Marathon which started at 2nd of July.

There are two groups of electronic engineers working on the Ecocar. Besides the main focus of the group there are also other minor electrical concerns to be taken care of, e.g. the servo to control the wiper of the car. This project differs from other projects, in the case that there are a lot of cooperation between other groups and other student programs. Also testing on the car, is not always possible since the car is often taken by other students. Time management is then very crucial for a project like this.

To be able to work with PCB design in this course, a preliminary course of everyone designing their own micro controller on a PCB has been done by each participant of the group. Through this every member is now adequately able to work with Cadsoft EAGLE[2] and use information from this program to understand how components are connected on the actual board.

III. Communication between boards

Controller Area Network (CAN) is a vehicle network for controlling the communication between multiple networks[3]. The communication between all of the boards in the car is solely handled via the CAN-bus where measurements and control signals are transmitted at roughly every 100 ms. The steering wheel does not send any measurements and therefore only transmits a 16 bit signal where each bit represents if either button is pressed or not. This is transmitted along with an ID so that any other board can verify that this signal comes from this particular board. The light board for instance only transmits the front break pressure, and is then purely controlled by signals from the Steering board. Two termination resistors are required at two endpoints at any time for the CAN-bus to function. These are located on the steering board and the ECU(Engine Control Unit). On the contrary, the ECU transmits all measurements and receives control signals from the steering board.

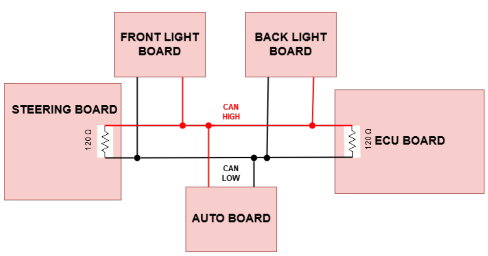

Fig. 1. Visualization of the CAN bus network in the car

IV. Board design

A. Steering board

The steering board is a PCB with a Teensy 3.6 microprocessor on the back and a 3.5” FeatherWing TFT screen[4] with resistive touch (an upgrade from last generation which was without touch), 10 onboard buttons (2 on the back and 8 on the front) and several LEDs. The board is powered through a Molex 2x2 which contains power, ground, CAN-HIGH and CAN-LOW signals, where the latter two are used for CAN-bus communication. It also contains a CAN-bus chip for converting signals to and from the CAN-bus. The main physical layout of the steering board was not changed much from the last generation - following are modifications made to this generation. Some changes to the shape of the PCB was made and 4 excess buttons along with their corresponding LEDs were removed. The CAN-bus signals on the steering board are fitted with 2x60 Ω termination resistors. A CAN-bus needs a termination resistor at two ends to function properly. On earlier versions of the car, these resistors were located on the light board and the motor board. This caused problems during testing whenever the front top shell of the car was off, due to the termination resistor being on the light board which is mounted the front shell. The steering wheel also features two minor PCBs named *X-boards* which each contains two buttons and two LEDs. The X-boards puts frequently used buttons closer to the driver’s thumbs. The X-boards are connected to the main steering board using small Molex row connectors. The main 12V power supply from the battery is treated with a low-pass LC filter to avoid the microprocessor restarting during usage of the horn, which creates noise on the 12V power supply line. We ended up rewiring the horn directly to the battery with a switch, to isolate it from the other lines.

1) Robust electronics: In the previous years the DC/DC converter was suspected to not be so reliable, hence this year we have changed it to a different more expensive one, that should be more stable under the conditions. As mentioned above, there were some problems with noise on the 12V power supply line, that we would like to eliminate on the boards. Therefore a low pass Chebyshev PI LC filter has been placed before the DC/DC converter. This filter only purpose is to attenuate higher frequency noise from the supply line.

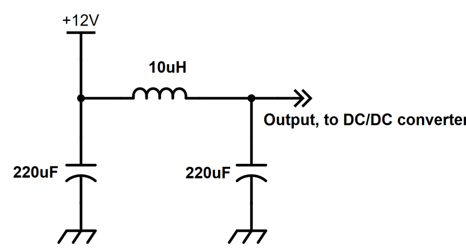

Fig. 2. Chebyshev PI filter circuit used on the Steering Board

2) Programming: The code for the steering board has been completely rewritten for much better readability and understanding both on the user end and programming end. To make effective functions, a structure called *CarParameters* is now implemented to contain all the values read from the CAN-bus. Whenever a function needs to make use of multiple values only a pointer to this structure is passed to keep a low stack size. All functions used are properly stored in meaningful files with headers. The files used are Buttons.cpp, CarParameters.cpp, Display.cpp, LEDs.cpp and logos.h. The names of these files clearly represent which functions they contain. This convention of obvious names has been diligently used throughout the code for variable and function names, this is to ensure an easily modifiable code for future generations. .

3) Visuals: To make testing vs driving easier, three different presets for the TFT display are now available. One for starting up and early testing named *startupScreen* and one focused on displaying only vital information to the driver called the *drivingScreen*. The startupScreen shows a list of all values present on the CAN-bus. It is possible to change the values shown by only editing a few lines in the code. In this mode, where applicable, icons are used to indicate what value is shown. The background turns red in case the oil or water temperature reaches a certain preset value to warn the user to take action. Lastly there is a *potentiometerScreen* where it is possible to change the potentiometer value using the paddles, this was a feature requested to improve fuel consumption mid-race.

For optimal driving, reading the speed and RPM is very important as well as the time elapsed. The gear is also essential in case the automatic gearing fails. Therefore the drivingScreen shows a big number that indicates the speed and then the rest few informations are also visible in the lower area. Warnings are written in the empty area in the middle with a red background.

4) Object-oriented code: During the project we developed an experimental version of the steering-board code. This version of the code was written to be very object-oriented, with the intention to make cleaner, more maintainable code. The theory behind object-oriented code is that physical objects that ”look alike” can be classified as the same object in the code. This object will then contain the code to work with the specific data. This is desirable because we can then avoid many duplicates of the same code. The most important aspect of the object-oriented code was the display, and more specifically, the widgets on the display. We implemented a widget to draw (static) text, and a widget to display the values of variables. We also had more specialized widgets to display logos, and time. We also implemented a screen-object, because we wanted to be able to show different ”screens” (see *Visuals*) on the display, and well as a display-object to bind it all together.

All of this meant that we could add new widgets to the display with a single line code, and because the redraw-logic was handled in the object-code, we didn't have to worry about redrawing elements on the screen at all. This is much simpler than the old version of the code where every element on screen should have code (specific to that element) to calculate position and size, erase, and finally (re-)draw. This code would often take 5-6 lines of code at multiple locations (per element on screen).

B. Light board

The car carries two light boards – one for the front and one for the back. These light boards consists of a Teensy

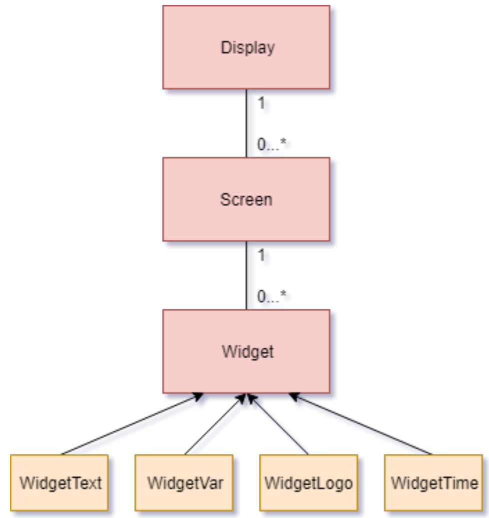

Fig. 3. Illustration of the classes in the code. The numbers between the classes shows how many each object can be linked to. I.e. *Display* can have from zero to many objects of *Screen* but *Screen* can only belong to a single *Display* object. The orange boxes are not classes but different specializations of the widget class to handle different types of widgets

3.6, CAN-bus chip, up to three buck converters† and pull-up/pull-down MOSFETs. The function of the front light board is to power and control the front light, wiper, fan and horn operation as well as reading the break sensor. The back light board powers and controls the back light and starter lights. This year the design of these boards were changed to control a servo for the wiper and the LED-strip in the front of the car, that is used for all the lighting operations in the front such as normal front light, blink and emergency light, but also as the autonomous light.

The LED-strip thus works as an all-in-one solution, and as a bonus it does not change the physical aspect of the car, thus preserving the aerodynamic shape. The wiper has been changed from a standard automobile heavy and bulky(but reliable) wiper motor to a smaller and lighter servo motor. This was done to cut down some weight of the car, which leads to better mileage. The old light board only had one power converter to power the Teensy and other internal circuits, whereas the MOSFETs were powered directly from the 12 V battery and controlled by the Teensy to power up the 12V lights strips used back then. This year we went with a 5V, 5 meter long RGB LED strip, that can consume up to 9A[6] of current. These 9 amps is a problem because the current buck converter could only provide 600mA. A servo can consume quite a lot under load, and a wiper is not exactly easy to drag, since the moment of inertia is large and therefore we needed to find a solution for this. We ended up going with two 5V 5A buck converters – the Polulu D24V50F5[5]. A small and yet powerful converter that can be connected to female pin headers, so they are suspended over the MOSFETs,

†The back light board only uses one.

thus reducing the amount of space consumption on the PCB. In this way we could keep the same dimensions for the PCB, thus only a small change in the 3d-printed casing was needed. The casing is used to mount the boards on the car and protect the boards for possible damages during driving.

V. Problems during the competition

During the competition there were several complications with the steering and light electronics. The steering electronics had a habit of working perfectly whenever we tested it, but whenever the car got fully assembled and was driving on the track, it crashed.

At the time the weather was very hot, and the sun was shining directly on the steering wheel through the window in the car. It is suspected the heat in the car during driving is making the steering board crash. We were unable to reproduce these problems whilst testing, therefore only hypotheses could be made with overheating being the most plausible cause. One mitigation of the heat-problem was to cover the steering wheel with a piece of cardboard, with the hope that the steering wheel would become less hot. The cardboard cover did not mitigate the problem.

If something was not working, an easy solution was often to just re upload the code on the boards. This seemed to fix the strange problems of boards suddenly not working anymore which for instance happened to the back light board.

VI. Future development

The ecocar is a dynamic process, where a lot can change each year. The changes can happen for many reasons such as optimization, new rules for the competition or a new category for the car to compete in, so logically the electronics must be developed to comply with these rules.
Testing the car in a warmer environment might be necessary to ensure the stability of the boards. Adding a sd-card to the microcontrollers and logging the temperatures, time and other values would help troubleshooting a lot.
To help improve testing the cars light functions, the light boards could be relocated to car itself instead of being implemented on the shell of the car, this along with also moving the LED-strips to the car would make it possible to test new light features functionality at all time.

VII. Conclusion

The resulting steering wheel was working as intended and the improvements using object based c++ code now makes additional features convenient to add/remove. The issue with the micro controllers restarting/freezing has not been rectified, and no concrete solution to the freezing was found. The steering board has all the required functions such as; horn operation, wiper, fan, light control and display of vital information. A big part of why this was possible, is thanks to the communication system CAN bus, which makes communication between boards an ease. The old light board was also redone with new step down converters to drive the new LED-strip that was used as both normal and autonomous lighting. Unfortunately the steering electronics was not so robust during the competition, and thus needs some further troubleshooting.

References

[1] ”Shell Eco-marathon rules chapter I”, *https://www.shell.com*, June-2018. [Online]. Available: https://www.shell.com/energy-and-innovation/shell-ecomarathon.html [Accessed: 13-06-2018]

[2] ”Autodesk Eagle”, *www.autodesk.com*, June 2018 [Online]. Available: https://www.autodesk.com/products/eagle/overview [Accessed: 14-06-2018].

[3] ”Introduction to the Controller Area Network (CAN)”, *www.ti.com*, June 2018 [Online]. Available: http://www.ti.com/lit/an/sloa101b/sloa101b.pdf [Accessed: 14-06-2018].

[4] ”Adafruit 3.5 TFT FeatherWing”, *www.adafruit.com*, June-2018 [Online]. Available: https://learn.adafruit.com/adafruit-3-5-tft-featherwing?view=all [Accessed: 13-06-2018].

[5] ”Polulo buck converter”, *www.pololu.com*, June-2018. [Online]. Available: https://www.pololu.com/product/2851 [Accessed: 08-06-2018].

[6] ”5 meter RGB LED strip”, *www.aliexpress.com*, June-2018. [Online]. Available: https://www.aliexpress.com/item/1m-5m-APA102-Smart-led-pixel-strip-Black-White-PCB-24-36-60-96-leds-pixels/32322326979.html [Accessed: 08-06-2018].

[7] ”Hitec servo motor”, *www.robotshop.com*, June-2018. [Online]. Available: https://www.robotshop.com/media/files/pdf/hs805.pdf [Accessed: 08-06-2018].