KTH Mechatronics Advanced Course

MF2063, HT 2018

FINAL REPORT

ESS-NW/ESS-CAR

Jonas Ekman
Yini Gao
Jacob Kimblad
Leon Fernandez
Fredrik Hyyrynen
Yifan Ruan

December 11, 2018

Abstract

Abstract starts here, what should be included:
The problem issue subject being addressed
How the problem is tackled
Overview of the results, and indication as to what level they solve the problem.
Implications of the results

Contents

List of Figures

List of Tables

1 Introduction

This report presents the process and results of two projects "Embedded Service for Self-adaptive Network" (ESS-NW) and "Embedded Service for Self-adaptive Car" (ESS-CAR). This chapter will start by describing the background of the two projects. The next thing to be described is formulation, goals and motivation of the two projects. Following this will be a short discussion on the delimitations for our team. The last part of this chapter will present an explicit report disposition which helps readers to get a sense of the overall report.

1.1 Background

1.1.1 Background subsection blabla

1.2 Project Description

1.2.1 Project Description sub blabla

1.3 Delimitations

1.4 Report disposition

Chapter 1 describes the projects' background and gives an introduction to the two projects. Chapter 2 provides a short description of theories used in the projects. In chapter 3, The methods, tools and techniques used to solve the problems in the project are specified. The implementation of the autonomous vehicle prototype is presented in chapter 4. Chapter 5 describes how we verify and validate the prototype. The results of the two projects are shown in chapter 6, then chapter 7, has a discussion and a conclusion on the final results. The last chapter, chapter 8, lists some future work.

2 Literature Review and State of the Art

3 Network

3.1 Software defined network

Software-defined network (SDN) is a type of network where a controller decides how the traffic in the network should go. In a traditional network is the intelligence in the switches and they decided on what port the package should be sent out on. In SDN is a device called controller connected to all switches and monitors the traffic load on the links and find the most optimal path between node A to node B. The controller's task is to request information from the switches about what links are up or down and the traffic load with this information and decides how packages should be forwarded on the switch. Because the controller can monitor the topology of the network is it easy to scale the network with new nodes and switches[1].

3.2 Serial Peripheral Interface

Serial Peripheral Interface (SPI) is a synchronous serial interface specification for short distance communication. SPI supports full duplex mode using a master-slave architecture with one single master, and the SPI master originates the whole communication. The SPI bus has four logic signals: serial clock (SCK), master output slave input (MOSI), master input and slave output (MISO), slave select (SS). The detailed pin mapping of SPI master and slaves is shown in figure ??.

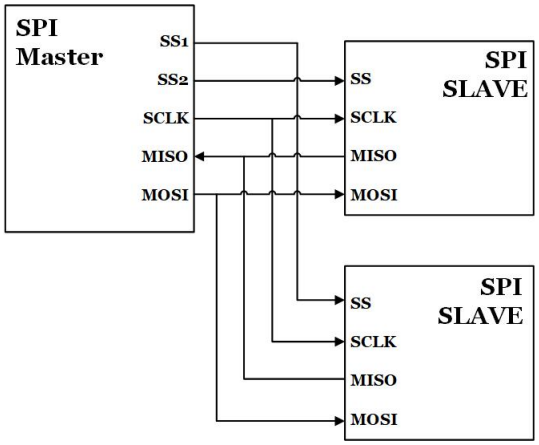

Figure 1: Connection of one SPI master and two SPI slaves

SPI has a higher throughput than those traditional transaction protocols, e.g. I^2C, UART, as it is not limited to any maximum clock speed enabling potentially high speed. And the hardware interface is pretty simple where slaves only use master's clock and are able to share the other three signals but the SS. Meanwhile, the software is not hard to implementation. For the opposite side, SPI has its limitations. There is no slave acknowledgement or hand-shaking mechanism in SPI, so the master could keep sending data to nowhere and not know it.

3.3 Assembly of the car

To make the car moving all the devices and components on the car has to placed on the car and because of there many devices on the car that communicates via SDN and has all of them be placed in an efficient way. To do that needs some kind of platform to be created to mount all the devices on and they are close to the other devices its depended on.

3.4 Power Supply

The car uses a 2 cell lithium battery what delivers a voltage of 7.2V. On the car is two different voltage level used, the first one is at 9V for a switch, and the second level is at 5V to power all the other components.

To generate a voltage at 9V from 5V or 7.2V has some kind of DC-DC converter be build to generate a higher voltage.

4 Methodology

4.1 Network

4.1.1 Software defined network

The SDN switches used in this implementation is the Zodiac FX from Northbound Networks. They are switches made to support OpenFlow protocol and is designed to be used in SDN networks. On the switch is 4 ports there one port that has to be connected to the controller, all the other ports can be used to connect devices on. As shown in figure ?? is 5 devices connected to the network and the switches has to be connected to each other has the network 3 switches.

Mininet is a Python-based application what an SDN network topology can be created and simulated to test that the controller works and analyse how the network will operate. The program also has the opportunity to connect the simulated topology to your physical SDN controller.

The car network topology was simulated in Mininet to analyse its behaviour, and to develop code for the controller. Under this test did we discover what that Mininet simulation had a much lower delay then the real implementation had, even then the extra settings on the links and nodes were added. This resulted that Mininet could not be used to analyse the best topology for the car. It was instead used to simulate the network to create a code for the Ryu controller. To analyse the Car network was two different Mininet scripts created, one was to use the physical raspberry pi controller and the second one was to use the internal version of Ryu in the Mininet VM. For the external simulation to work has the script be configured with the right ip-address to the raspberry pi.

4.2 VSOME/IP

4.3 Engineering approaches ?

4.4 Tool-chains ?

4.5 Project management

Scrum project management is used during the process of our projects.

4.6 Assembly of the car

The car platform used in this project is a car platform is Turnigy SCT 2WD 1/10 Brushless Short Course Truck (KIT) upgraded version and it was provided by the stakeholders at the start of the project.

To place all the required components on the car had some kind of platform be created to mount everything on. The requirements for the platform is if passible the car chassis should be able to fit on top of the car, all the device should be mounted on it and it should be easy to remove from the car.

To make this platform was it first draw in a Cad Fusion 360 to make a 3D model of the car. This model could then be used to add models of the devices used in the car to find the optimal place for them, so the are places in at an easy position to get access too and close to its devices its depended on. The result of the design is shown in figure 2.

4.7 Power Supply and PCB

The power supply for the devices on the car is done via two different levels one at 5V and the second one at 9V. This requires some converter to converts the 7.2V from the battery to 5V and 9v. The 5V DC-DC converter is a Turnigy USBEC-15A it as voltage regulator made especially for lithium batteries and has input voltage range between 6V-12.6V and it can deliver 5V volt as output. This converter was used because it was already provided in the material we received from the stockholders

and because it indicates how much power there is left in the battery. The most important reason the converter module was used is because lithium batteries are sensitive can easily be damaged, so it was a smaller risk the battery would be damaged and affect the project development.

The second converter is to generate the 9V and this is done via a DC-DC converter called boost converter. The idea of the boost converter was it should be connected after the first converter and would have an input voltage at 5V to generate a 9V output and a current of 0.6A. The reason it uses the 5V voltage level as input and not the 7.2V from the battery is to have all the powering done via the Turing converter and would not have something connected to the battery, that could damage it. The design for the boost converter was done in a program called MPLAB Mindi and is an analogue simulation program from Microchip, and it was used because the IC chip MIC2253 is from Microchip and it was provided as a PSpice simulation component so the circuit could be analysed. The MIC2253 is DC-DC boost converter wat has a switching frequency of 1MHz has an input voltage range between 2.5V-10V. The schematic for the boost converter is shown in figure 3 and designed form the recommendation provided in the datasheet for the MIC2253. The MIC2253 has the functionality of an Overvoltage Protection pin that is used to shit down the switch if the pin has a voltage higher then 5.6V, to solve this was to resistors R5 and R6 in parallel added to the OVP pin as shown in figure 3. The values of R5 and R6 is calculated from formula 1. To get the correct output voltage had R1 and R2 in figure 3 be changed to correct values. They are calculated from formula 2 and the V_{out} was set to 9V. This circuit was then simulated in MPLAB and it showed that the circuit should give an output voltage at 9V and a current peak of 0.9A, as shown in figure 4.

$$9V < 1.245 * \frac{67k\Omega * (R5 + R6)}{15k\Omega * R6} \tag{1}$$

$$V_{out} = V_{ref} * (\frac{R1}{R2} + 1) \tag{2}$$

All the powering of the car's devices should be handled by one PCB board, that contained USB ports and the boost converter to power the external switch. There the USB-A type ports should be used to power the Arduinos, Beaglebones, Raspberry Pi and Zodiac FX. For the powering of the external switch should a cable be connected from the PCB to the switch, it also to power outlets to power the PCB board for the Arduinos. This PCB board was designed in Eagle and the result is shown in figure **??**.

For the Arduinos was two PCB boards made to easily connect the SPI connection between the Beaglebones and the Arduinos, and to connect external sensors. The first board was designed to easily connect the ultrasonic sensors and the steering servo with correct Arduino. The schematic and layout of this board was also made in Eagle and is shown in figure 6. The board is powered via a power via the DC power jack, to supply the Arduinos. On the top of the board is three connectors and they are for the Ultrasonic to be connected, and in the bottom is the connector for the steering servo. In the middle of the board is the connection to add the Logic level converter and the pins to connect the connection from the Beaglebone.

The second Arduino board has the same functionality as the previous one. The difference is that this is designed to connect the speed sensor and the ESC to control the engine. The layout of this design is showed in figure 7.

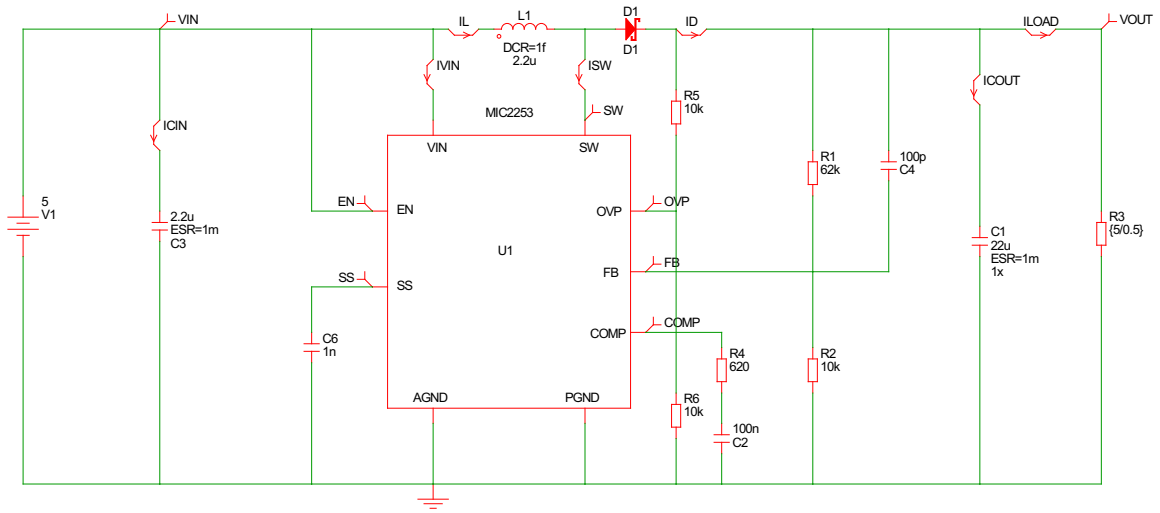

Figure 3: Circuit of the boost converter

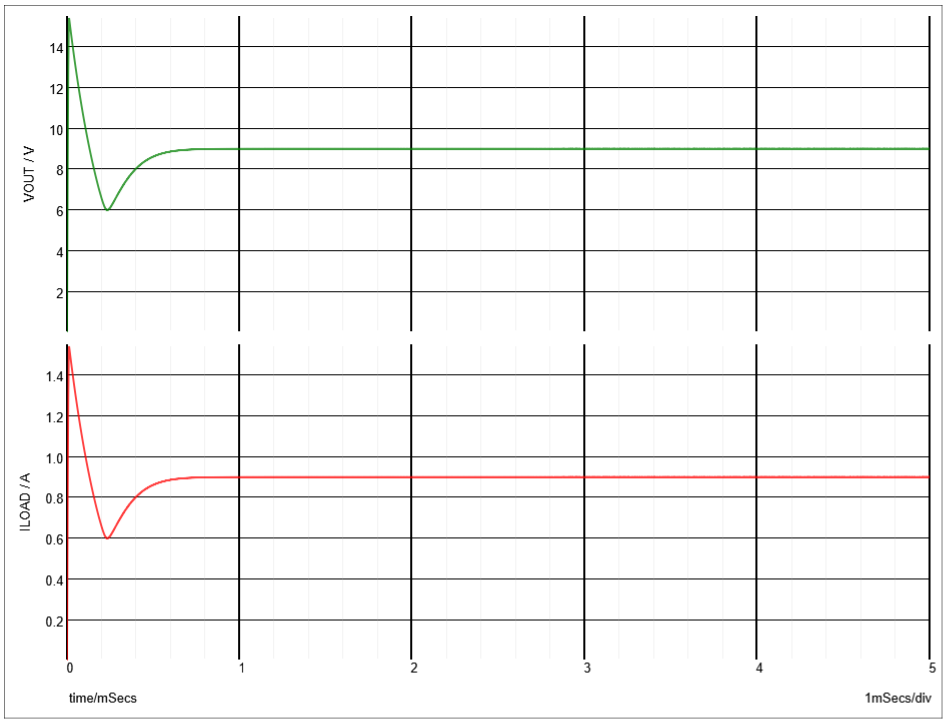

Figure 4: Out puth grapth of the Boost converter

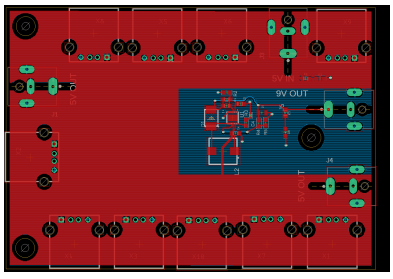

Figure 5: Layout of Power PCB board

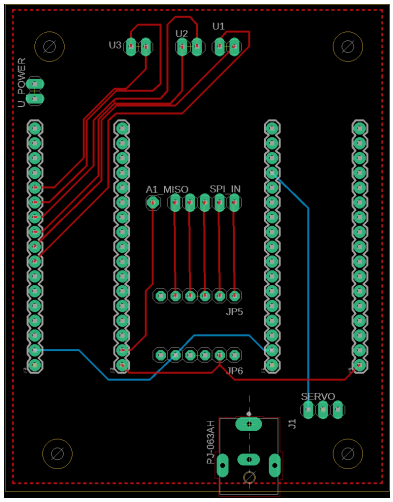

Figure 6: Layout of the Board for ultrasound and steering servo

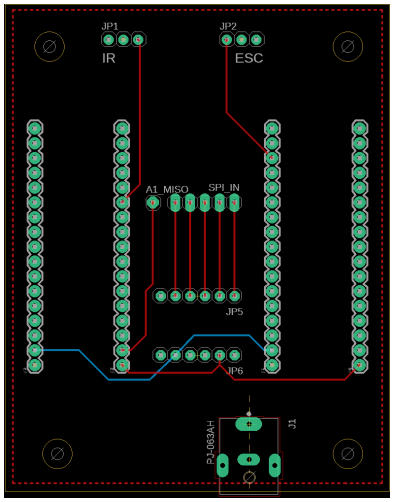

Figure 7: Layout of the Board for speed sensor and ESC

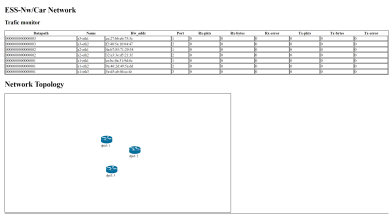

Figure 8: Webpage to monitor the SDN network

5 Implementation

5.1 System overview

maybe put communication diagram here

5.2 SDN network Implementation

At the beginning of the project was floodlight that SDN controller framework we decided to work with. Under the project way was a lot of problem coming up to integrate floodlight on raspberry pi due to some java packages used in Floodlight was not supported, and it had difficulties to communicate with the SDN switches. This resulted in that another SDN framework was selected, and it was a framework called Ryu. Ryu was selected because it was well documented with some example code to start with, and it was developed in python, so it should be no problem with integrating it on a raspberry pi.

The code for the Ryu controller is based on the example simple-monitor.py there the scripts requests information from the switches about how many ports that are used and the amount received and send traffic. The modification don to it is to save the received information so it can be presented in on a webpage.

To monitor the traffic load on the network was a webpage made to present the user with data. The webpage is developt from the provided network webpage included in Ryu. On the webpage, as shown in figure **??** is a table what presented the data that has been transmitted and received on each port, and the number of packages has been dropped. The web pages are designed to update itself every 10s to be updated with the latest information. This has to do with the controller request new information from the switches every 10s. In the bottom of the webpage is a picture of how the network is configured.

5.3 shared memory things

5.4 Communication between Beaglebone and Arduino ?

As the Beaglebone only supports to be the master in a SPI connection, so we have two beaglebones with SPI configurations and each of them has two Arduino slaves. One Beaglebone has two spi devices: SPIDEV0 and SPIDEV1, and the SPIDEV1 device is disabled initially but is used for HDMI interface.

In order to have one SPI device on Beaglebone connecting to multiple Arduino slaves, we use a third party C++ library called SimpleGPIO which enables us to use other general purpose pins as slave select of the SPI connection. Before we start communication with a specific Arduino slave, the corresponding slave select will be set to low, then we get access to that Arduino slave.

In the Arduino side, the communication request from the Beaglebone is treated as an interrupt. Then the Arduino writes/reads to/from the Serial Peripheral Data Register (SPDR).

5.5 Sensors

Three categories of sensors are implemented in the prototype vehicle to monitor its surrounding environments. Data from distance sensors and speed sensor will be sent to an Arduino initially, then sent to corresponding Beaglebone. Data from Pi Camera will be sent to the Raspberry Pi which is directly connected to the main network.

5.5.1 Ultrasonic sensor

To get data from HC-SR04, a short 10µs pulse should be supplied to the trig pin of ultrasonic sensor, then the sensor will send out an 8 cycle burst of ultrasound at 40 kHz and raise echo. The echo signal we get is a distance object which is pulse width and the range of the signal is in proportion.

An Arduino Micro handles both the generating the trig pulse and interpreting the echo signal. The Arduino will set the output pin to low, wait for 5ms, set the pin to high, wait for 10ms, set the pin back to low. This is the process of generating the trig pulse. After the Arduino sends out the trig pulse, it waits for 2 ms, then reads the value from the pin connected to sensor's echo pin. The last step is convert the received value to distance in unit of centimeter.

5.5.2 Reflective object sensor

5.5.3 Pi Camera

In computer vision part, we tried to compare the performance of two different methods, which are object detection based on neural network and color detection using OpenCV.

The technique of transfer learning is used to apply MobileNet on RPI, which is an efficient model for mobile and embedded vision applications. The speed remains slow although hardware acceleration and multiprocessing have been carried out to improve.

By using OpenCV's DNN module, we are able to pass input images through the deep neural network and get a output image with bounding box of specific object and label.

Since we are working on source constrained device Raspberry Pi, we need to make the network simple and decrease the computational cost, so we combined MobileNet with Single Shot Detector.

In order to optimize the Raspberry efficiently and use sufficiently limited resource and memory on it when running neural network, we firstly apply hardware optimization which is to install optimized OpenCV complie. So, ARM NEON as an optimization architecture extension for ARM processors is used. It is used for faster video stream processing, image analysis and speech recognition that is exactly what we are look for in our application. This architecture can execute multiple processing in the pipeline by a single instruction. Besides, VFPV3 as a floating point optimization is also used. After the first stage improvement, we get a 30 percent speed increase since we have make full use of the 4× ARM Cortex-A53,

When it comes to second stage of optimization, multiprocessing is used to increase the speed of processing video stream. The I/O tasks, dislike CPU bound operations, always take lots of time and delay the process. So moving the reading of frames to a separate thread from frame processing can obtain higher speed. Otherwise, every time I/O port access Pi camera, the main thread of script is blocked until the frame is captured and return to script. Multiprocessing can decrease the influence of I/O on CPU heavy application like video stream processing, especially in our real-time case. Now we can obtain a detection result within 1 second.

Additionally, there is a trade off between accuracy and output speed. In our application, we set a threshold for the output, in more details, only when the confidence score of detection result is high enough (above the specified threshold), the result can be output as a signal to steer the car. Otherwise, it will grab another frame and do the object detection for the other iteration. As a result, if the threshold confidence score of output increase, the output speed will decrease.

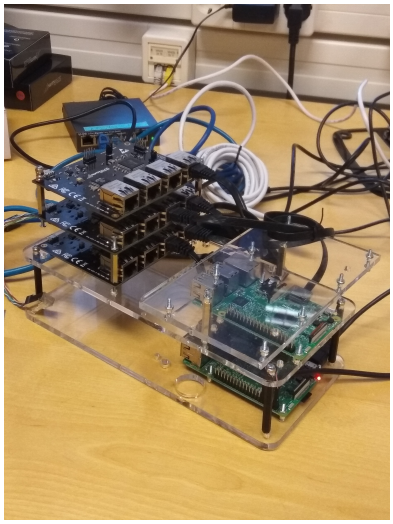

Figure 9: Platform

Since the object detection method outputs result slowly, we move on to the other method which is color detection using OpenCV to see whether the speed is fast enough to be applied on a car prototype.

We define the upper and lower limits for pixel values to classify three colors. Then specify which pixels fall into specified upper and lower range by masking. The speed of color detection is fast and also accurate enough for real-time application.

So we finally apply the color detection on our car, we use the result of detection to steer the car.

5.6 Controlling actuators

5.6.1 Steering servo

5.6.2 Motor ESC

5.7 Assembly of the car

The platform pieces are achyle plastic and were cut via a laser cutter, the schematic for the pieces was exported from the 3D model to Illustrator to be sent to the laser cutter. The result of it is shown in figure 2. The platform was mounted via board spacer of different sizes to mount the levels together. All the devices on were mounted via screws and nuts to keep them in place.

The design of the platform was done to use PCB bards and not prototyping bards, this made the assembly of the car not to be as the idea was. This is because it was difficult to make the prototyping bards small and compact as you can do with PCB. Two of the boards The power board and one of the Arduino boards were able to place in the targeted position but could not be mounted to the platform with screws but with cable ties. The other board had to be placed in front of the car due to it could not be fit at its target place because the power board and the Arduino board was too thick of the wiring on the boards.

5.8 Power Supply and PCB

The idea of was to order the PCBs from a company, but due to course regulation ware we not allowed to order PCBs from China, and companies in Europe were too expensive to order from. This resulted in what we could not implement the boost converter we hade design due to the milling machine in both prototype centre and in Mentorspace was broken and they could not mill the layout for the boost

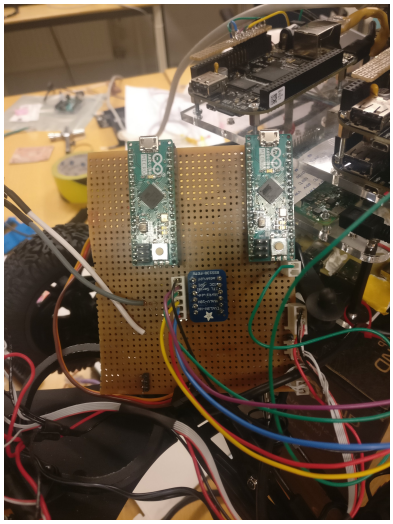

Figure 10: Arduino board for the steering and Ultrasound

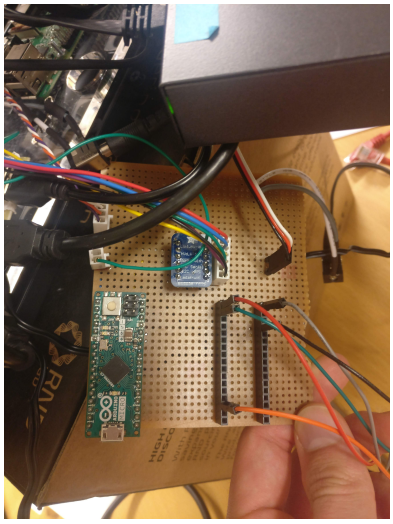

Figure 11: Arduino board for the engine and speed sensor

converter. Because the milling machines were broken could the PCB for the Arduinos also not be made. The only way we could continue with was to make the design on a prototyping board.

The implementation on the prototype board was built from the same concept the PCB should have had, with USB-A ports cables to power the Arduino boards and external switch. The bard connects via a cable from the battery to a fuse, this fuse is implemented to protect the system if a short circuit happens. To solve the boost converter circuit did we buy a boost converter module MT3608 to receive a 9V. This module is not connected to the 5V level but instead to the to the 7.2V from the battery, to receive a current higher than 0.6A. To prevent previous years failure then they used the module, did we add the same solution to add an led with a resistor in series and $6.8\mu F$ in parallel to the output pins on the boost converter.

The Arduino boards hade also be made at prototyping board. They received the same functionality as the PCB version should have, but it takes larger space due to the cable wiring. This lead to one of the board could not be placed at its position on the platform.

The result of the prototyping is shown in figure 10, 11 and **??**. All of them has the functionality has the same PCBs should have had.

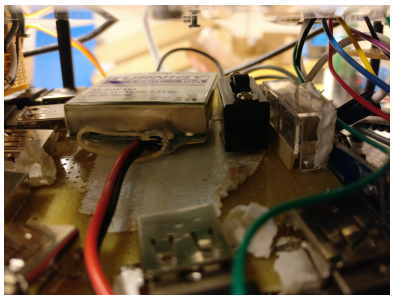

Figure 12: Power board

6 Verification and Validation

7 Results

8 Discussion and Conclusion

8.1 Assembly and PCB

Due course regulation was we not allowed to order PCBs from China and to order it from a European company was too expensive. The idea for the project was to design our own PCBs because the project should combine the knowledge we hade learned during our masters and some of the students in the project is taking the electronics track. But due to the course regulation were we not allowed to order from companies that did not accept invoice as a payment, it resulted that we could not order from the cheap PCB manufactures in China. The group tried to find a European company but they were too expensive. This lead to a redesign of the powering and we decided to use the milling machines but they there also broken so the only solution was to use prototype boards. There is kind of strange that there is no machines or ways to order PCB because the electronic track is to about design the electronics and there is no way for us to be able to satisfy the course goals.

Because we hade to use prototyping boards and the platform was designed to have PCB mounted to it and not prototyping boards. This led to the assembly of the car was not as optimal as it could have been and the placement of come components and boards had to be placed at a different location. If the car would have been able to move around all it would have been affected of this, due the cable wiring was not optimal and the reliability of connections and wound have affected the car's performance.

9 Future Work

References

[1] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (sdn): a survey," *Security and Communication Networks*, vol. 9, no. 18, pp. 5803–5833. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1737