Mälardalen University
School of Innovation Design and Engineering
Västerås, Sweden

DVA425 & 24055H16 - Project in Advanced Embedded Systems 30.0 credits

SOLAR CHALLENGE 2017

Palm, William
wpm12001@student.mdh.se

Heidari, Mohammad
mhi15001@student.mdh.se

Fischer, Fredrik
ffr11001@student.mdh.se

Hedlund, Daniel
dhd11001@student.mdh.se

Ramberg, Andreas
arg12002@student.mdh.se

Supervisor: Mikael Ekstrom
Mälardalen University, Västerås, Sweden

January 27, 2017

Abstract

In October 2017 MDH Solar Team will compete in the challenger class of Word Solar Challenge held in Australia. This project focuses on electrical design of the solar car both in term of hardware and software. A communication link is needed between the solar car and a following car in order to monitor and control systems in the solar car. To maximize efficiency, mathematical model optimization were used. Firstly a mathematical model estimates available solar power, depending on time and location, Secondly a model of race conditions and solar car were utilized to predict/optimize power consumption. Since the car's body design has not been finished yet the mechanical model is created with mainly estimated parameters of the final solar car. Both mechanical and available solar energy model are used in two different velocity optimizing programs to simulate the race. The first program divides the race into smaller sections and then, by an iterative process, calculates an estimated optimal velocity. The second program keeps track of the State of Charge (SoC) of the battery system. And with a PID controller keeps the SoC at a predefined drain through the whole race and so gets an estimated optimal velocity for the car. An intuitive and easy to use GUI is developed. The main purpose is for the following car to monitor and control critical systems of the Solar car. All car electronics that are necessary according to the regulations of the race are built on a small size electrical quad. PCB's are developed to give an easy connected and robust electrical system. Critical components for the final cars electrical system are chosen and ordered. A dashboard for switches and display is manufactured and mounted on the quad. The communication system together with the GUI is integrated on the quad and a manipulative cruise control is developed and tested. The battery system for the Solar car is developed together with a Polish company and the motor system is ordered from Japan.

Abbreviations

following-Car	The Car that follows the Solar car
GMT	Greenwich Mean Time
GUI	Graphical User Interface
MVVM	Model-View-ViewModel
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter

Table of Contents

1 Introduction

In October 2017, MDH Solar Team will be competing in World Solar Challenge. World Solar Challenge is a competition/event held in Australia every second year and has received a great amount of attention over the last decade. The attention is mainly because of the environmental concerns, but also because of the positivity on research towards sustainable energy. There are a number of classes within the competition and MDH Solar Team will be competing in the most challenging one, the challenger class. In this class electrical vehicles will race against each other, driving a course over 3000km long, straight through Australia (Darwin to Adelaide). The catch is that the electrical vehicles can only be powered by solar energy throughout the whole race and the first vehicle to reach Adelaide in the south of Australia wins.
In this project, the complete electrical design of the car has been made according to the regulations of the race. A smart and user friendly interface with predictive functions has been developed to keep track of system parameters and let the user estimate a optimal velocity for the car.
This projects main components are the following:

- Communication; hardware and software
- Car electronics; lights, horn etc.
- Battery configuration and design;
- Motor systems parameter determination;
- User friendly interface;
- Smart cruise control system;
- Velocity optimization program;

Since the chassis and car body is not yet built, the electrical system has been built on a small size electrical quad. The electrical system has been built to be modular, robust and simple so that the transfer process to the solar car in the future will be as easy as possible.

2 Communication

A communication link between the Solar car and the following car is needed for the smart energy system to work. The system needs heavy calculations and GPS monitoring that is preferably not done in the solar car to minimize energy consumption. The idea is to calculate an optimal speed that is depending on the incoming energy from the solar panels, voltage levels of the battery, time left of the day and road slope information. The optimal speed is to be presented to the leading car as well as to the driver of the solar car. The need for a reliable communication is obvious and the chosen platform is an Arduino with a Xbee module. It provides a reliable UART communication between the two cars.

2.1 Hardware

Below are the hardware used in this project in regards to enable communication between two cars and electronics/sensors/equipment inside the solar car.

- 1 Arduino Mega 2560
- 1 Xbee Shield
- 2 XBee Pro S2B
- 1 CAN-BUS Shield DEV-13262
- 1 TTL to USB connector
- DB9 Connectors

Arduino Mega is used as the main embedded system for the solar car due to its processing capabilities, abundant hardware ports and IOs, ease of implementation and being open source hardware. Solar car will have different electronics and sensors that their output data will be directed to the Arduino Mega and in turn will be transferred to the following car for further processing. More information about hardware and specifications can be found in the appendix,

2.2 Software

- Visual Studio 2015.3
- Visual Macro's Addon for VS
- Arduino IDE v1.6
- MATLAB R2016b
- Kvaser CanKing

2.3 Connecting Hardware

Since command center and solar car are physically in different places (Command center is a MATLAB program which is running on a laptop in the following-Car), there was a need for a wireless solution to enable communication between these two units, so the choice of xBee module. Below is the configuration needed to let both xBee units talk to each other.

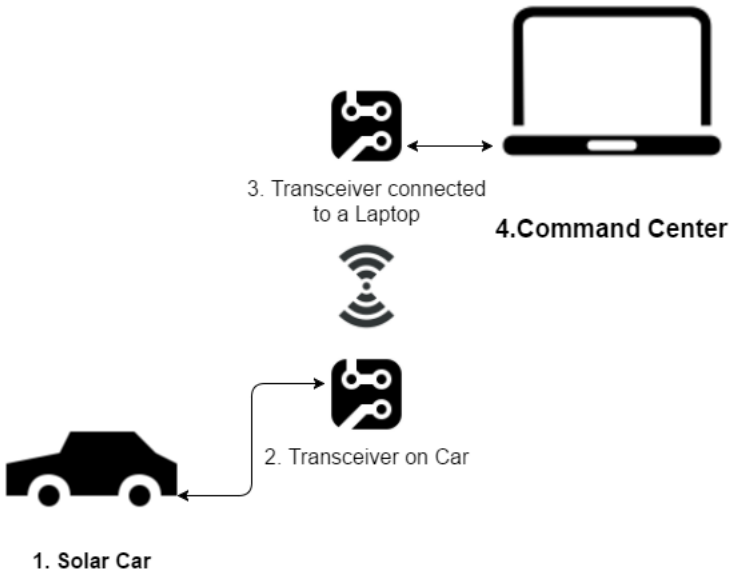


Figure 1: Communication between car and command center

Transceiver 2 is embedded in the solar car and collects all the information (By Arduino Mega board) coming from all available sensors/metrics on the car and sends them over to Command Center(MATLAB Application) via xBee module. In return there is another xBee on Transceiver 3 which is connected to Command center(Laptop) and receives all the information coming from the car and pass them to processing unit. Furthermore command center will analyze the incoming data and make decision based on the facts and figures and sends back relevant commands via the same route.

2.4 Xbee configuration

To configure xBees the following steps should be taken.

1. Remove xBee shields from Arduino boards
2. Upload this code to both roms
$$setup()\{\};loop()\{\};$$
3. Download X-CTU v5 (download here)
4. Connect a xBee Pro on xBee Shield then connect it to Arduino board
5. There is a jumper on the shield make sure it is set to USB
6. Run X-CTU go to PC Setting tab click test/query make sure Baudrate is 9600(should get OK)
7. Go to Modem Configuration tab, click read, you should see the modem info
8. Write down the Serial Number High and Serial Number Low (you will need them)
9. Choose a PAN ID eg. 1111 (only modems with same PAN ID will communicate)

10. Make sure the function set is Router AT and then tick Always Update Firmware and click Write to update the firmware.
11. If anything goes wrong, start from step 1
12. Disconnect this Arduino and Connect the other Arduino
13. Run X-CTU go to PC Setting tab click test/query make sure Buadrate is 9600(should get OK)
14. Go to Modem Configuration tab, click read, you should see the modem info
15. Change the function set to Coordinator AT
16. Write the same PAN ID that you wrote before
17. In the field Destination Address High write Serial Number High of previous board that you took note.
18. In the field Destination Address Low write Serial Number Low of previous board that you took note.
19. Make sure the function set is Coordinator AT and then tick Always Update Firmware and click Write to update the firmware.
20. If anything goes wrong, start over

Note: if the jumper on shield set to USB, communication will happen between Arduino board and PC via Serial port, and there is no communication between Arduino board and xBee. If the jumper set to Micro, xBee will try to communicate with another xBee in the same PAN, there will be no communication between Arduino board and USB port. It is because the single UART port on Arduino is connected to USB port. By default xBee is operating at 9600 baud rate, make sure it is common among all connected devices.

2.5 CAN Bus

There are several parts in the solar car which use CAN-bus interface for communication, Sparkfun's CAN-Bus Shield is used in this project to provide communication in the car between different nodes (parts) such as Solar panels, Batteries, Motors and so forth. The shield is based on Arduino's open shield platform so it can be easily mounted on Arduino Mega. Since this shield does not have SPI headers, the solution was to remap the pins on the shield to the Arduino's Mega pin for SPI communication, below is the remapped PIN numbers:

SPI Pins	<=>	CAN Sheild Pins	<=>	Arduino Mega Pins
SCK	<=>	13	<=>	52
MISO	<=>	12	<=>	50
MOSI	<=>	11	<=>	51
SS	<=>	10	<=>	53

Table 1: SPI rerouting pins

Once these configuration (rewiring) is done on the board and shield the default pin configuration in the c file needs to be changed accordingly. Below is the file to change and new values, after including the CAN library by Sparkfun into the project few files will be added to the system, look for defaults.h and change the values according to the table below (more configurations can be found in appendix A and B):

```
// Pins remapped to Arduino Mega 2560
#define    P_MOSI              B,2 // pin 51
#define    P_MISO              B,3 // pin 50
#define    P_SCK               B,1 // pin 52
#define    MCP2515_CS          B,0 // pin 53
```

Below is the current configuration scheme for the hardware. At the very bottom there is an Arduino Mega, due to some custom wiring an special extension shield has been created and mounted on top of the Mega. Next xBee shield is paced on top of the extension shield and finally the CAN-Bus shield on top of them all.

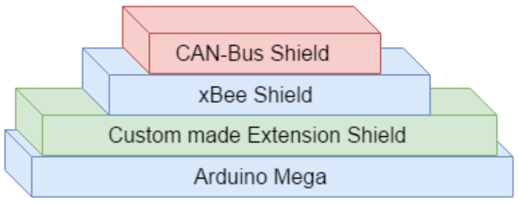


Figure 2: Hardware stackup

3 Energy Optimization

In order to optimize energy usage of the car during the race, some mathematical models have been developed to predict and control energy usage. Below are the mathematical models:

3.1 Solar energy calculation

Sun is the only power source available during this race in order to maximize and optimize the power usage a MATLAB program was developed to calculate and predict the available solar energy at any time and anywhere during the race. It is a function of time and place which outputs the solar radiance. Calculation of the solar power based on these factors:

- Date
- Time of day
- Solar time
- Geo Location (Longitude & Latitude)
- Solar panel surface tilt
- Solar panel's Azimuth angle
- Solar panel's efficiency
- Solar panel's area
- Atmospheric condition (excluded for the sake of simplicity)

First step is to calculate solar time which is dependent on current date, time and standard meridian of current location. Time and date can be retrieved using MATLAB's built-in function, and to calculate standard meridian this equation can be used:

$$\textit{Standard Meridian} = \Delta T * 15^\circ \tag{1}$$

where ΔT is difference between current time zone and GMT (in hours), it is multiplied by 15°due to the fact that each 15°is considered 1 solar hour. Next step is to calculate local solar time, to do so equation 2 is used which is based on [A Solar Car Primer]

$$T_s = \text{Standard time} + 4(L_{st} - L_{ng}) + E \tag{2}$$

Where $\mathrm{T_s}$ is solar time, $\mathrm{L_{st}}$ is standard meridian value from equation 1, $\mathrm{L_{ng}}$ is the longitude of the observant and E is residual error which is caused by small variation in earth's rotation speed according to Spencer 1971 and can be calculated using equation 3:

$$E = 9.87 * \sin(2 * B) - 7.53 * \cos(B) - 1.5 * \sin(B) \tag{3}$$

Where B is the day of year adjusted according equation 4

$$B = (360/365) * (N - 81) \tag{4}$$

Where N is the day of the year using MATLAB built-in function. And 81 is an adjusting value to account for variations in the earth's eccentricity and obliquity[1]. Now it is possible to calculate hour angle, this is the solar time in angle with respect to local meridian. Considering solar noon is perpendicular to the surface and having an hour angle(ω) of 0 and hour value of 12, each hour towards sunset will be positive, on the other hand morning hour angles will be negative (each 15°measured as 1 solar hour). Below figure will depict this matter with assumption of sunrise = 6AM and sunset = 6PM

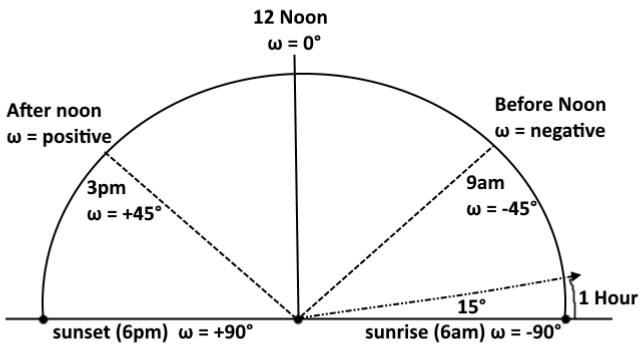


Figure 3: Solar hour angle

Equation 5 can be used to calculate hour angle:

$$\omega = 15° * (T_s - 12) \tag{5}$$

Where (ω) is hour angle and $\mathrm{T_s}$ is local solar time retrieved using equation 2. Declination angle (δ) is the rate in which sun is declined and it is the same for the entire globe for any given day. Depending on year's equinox (during autumnal and winter equinox moves towards negative values whereas during vernal and summer equinox it will be positive) and it fluctuates between -23.45°to +23.45°. Calculation of Declination angle (δ) is shown in equation 6:

$$\delta = \sin^{-1}(\sin(23.45) * \sin((360/365) * (N - 81)) \tag{6}$$

Where (δ) is declination angle and N is the number of day such as 1_{st} of Jan, N = 1. All angles are in radians and one need to convert them to angle based on the needs. Based on (paper here) Solar flux constant $G_{\mathrm{sc}} = 1366 watt/m^2$ and is the average amount of energy striking $1m^2$ (perpendicular to sun's ray) of earth surface every second. Considering extra-terrestrial radiation $\mathrm{G_{sc}}$ will change to:

$$G_{adjusted} = G_{sc} * (1 + 0.033 * \cos(360N/365))$$

Where N is the number of day. The amount of solar energy received on earth's surface (perpendicular to radiation) is called solar insolation and is about $\approx 1000W/m^2$ on a clear day which can be affected by radiation angle and atmospheric conditions (such as cloud coverage). In order to compensate for these issues and for the sake of simplicity a scaling factor ($\mathrm{K_{scaling}}$) is used to adjust the solar insolation calculation for this project. So

$$G_S = K_{scaling} * G_{adjusted}$$

Where $\mathrm{G_s}$ is the new(scaled) solar flux, $\mathrm{K_{scaling}}$ can be between 0 to 1 depending on the conditions. Sunrise and Sunset values are very useful entities to know whilst calculating solar energy and they can be extracted using equation 7

$$\begin{aligned} \omega_S &= \cos^{-1}(-\tan(\phi) * \tan(\delta)), \\ \omega_R &= -\omega_S \end{aligned} \tag{7}$$

Where ω_S is sunset hour angle and ω_R is sunrise hour angle, ϕ is observant latitude, and δ is declination angle. Observing figure 3, it is clear that sunset angle is a positive number and sunrise angle will be sunset times -1. Knowing all the necessary parts of the equation, it is possible to predict total amount of solar insolation available during a day using equation 8:

$$H_d = \frac{24 * 3600}{\pi} * G_{adjusted}(\cos(\phi) * \cos(\delta) * \sin(\omega_S) + \frac{\omega_S \pi}{180} * \sin(\phi) * \cos(\delta)) \tag{8}$$

Where H_d is the average amount of daily solar insolation. Below is the comparison of available solar insolation for one week in Darwin, Australia and Västerås, Sweden in kWh/m^2.

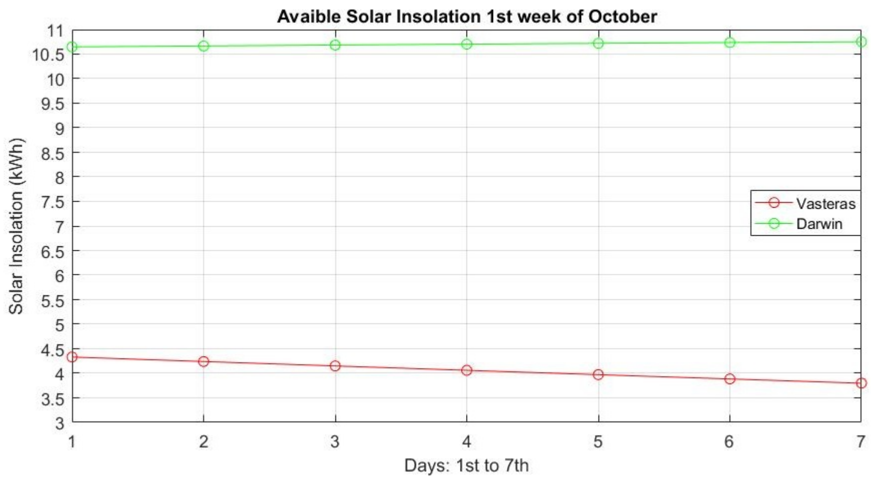


Figure 4: Average available solar insolation for a week

These calculation are based on ideal weather condition with scaling factor($K_{scaling}$) of 1 and shows that during first week of October 2016 approximately constant 10.7kWh (average) of solar energy will be available through out the week in Darwin, Australia. On the contrary this will be much less in Vasteras, Sweden starting at 4.4 kWh and dropping to 3.6 kWh as the week goes by. Solar panel efficiency or tilt angle is not considered it also assumes perpendicular radiation. The amount of harvesting energy from available flux depends on the design (tilt angle), efficiency of solar panel (different working temperature), time of day, weather condition and area of solar panel [2] [3]. Since the panel will be static and mounted horizontally on the car so the tilt angel is not considered but sun's azimuth angle will be taken into account which can have a greater impact, as well as panel's efficiency and area. By tweaking equation 8 it is possible to generate a function to find the amount of available solar energy between two specific hours anywhere in the world that can be of great use for optimization scenarios

$$E_s(t_2, t_1) = \frac{12 * 3600}{\pi} G_S(\cos(\phi)\cos(\delta)(\sin(\omega_2) - \sin(\omega_1) + \frac{\pi(\omega_2 - \omega_1)}{180}\sin(\phi)\sin(\delta)) \qquad \omega_2 > \omega_1 \tag{9}$$

Where G_S is the adjusted solar flux, and t_2, t_1 are the two time(s) corresponding to ω_2, ω_1 hour angles. Other parameters are latitude ϕ and declination angel δ. For instance the available solar power in Darwin, Australia on October 1st between 10am to 11am with latitude of -12.4667 will be: 0.90 kWh/m^2

$$\begin{aligned} E_s(11,10) &= \frac{12 * 3600}{\pi} 974.64 \\ *(\cos(-0.2175)\cos(-0.2895)) \\ *(\sin(-0.2638) - \sin(-0.5256)) \\ + \frac{\pi(-0.2638 - 0.5256)}{180} \\ * \sin(-0.2175)\sin(-0.2895)) = 0.90 kWh/m^2 \\ Where \\ \omega_2 &= 11am = -15.12^\circ = -0.2638 Rad \\ \omega_1 &= 10am = -30.12^\circ = -0.5256 Rad \\ G_S &= 974.64 \\ \phi &= -12.4667^\circ = -0.2175 Rad \\ \delta &= -16.59^\circ = -0.2895 Rad \end{aligned} \tag{10}$$

Above equation shows the available solar energy considering scaling factor, in order to find out the amount of energy for this project, solar panel's specifications need to be taken into account as well, final formula to calculate net collected solar energy is given by:

$$E_s(t_2, t_1) = \eta * A_s * I_s \tag{11}$$

Where η is efficiency of solar panel (0.229), A_s is area of solar panel and I_s is available solar insolation for that time period. The insolation (I_s) calculated using equation 9 considers vertical sun's ray to the surface, in order to accommodate for sun's beam angle towards solar panel's surface, sun's azimuth angle can be added to the equation and a more realistic result will be given by:

$$E_s(t_2, t_1) \approx E_s(t_2, t_1)/(\cos(\theta_{z1}) + \cos(\theta_{z2})) \tag{12}$$

Where θ_{z1} is azimuth angle of ω_1 and θ_{z2} is azimuth angle of ω_1 and azimuth angle can be calculated by:

$$\theta_z = \cos^{-1}(\cos(\phi)\cos(\delta)\cos(\omega) + \sin(\phi)\sin(\delta)) \tag{13}$$

Where ϕ is latitude of observer, δ is declination angle and ω is the hour angle. Figure 5 shows average solar energy available during 1st day of October 2016, in Darwin, Australia, The blue bell shows the available energy before scaling (G = 1 = ideal condition) which is about 1.27kWh at peak and more than 1kWh from 9am to 2pm. The red bell shows the same with scaling factor of (G = 0.7 = maybe cloudy) with peak value of 0.9 kWh, for the same time period (9am to 2pm) it will provide minimum of 0.7kWh and maximum of 0.9kWh solar energy. The two straight lines (black line with scaling factor of G = 1 and green line G = 0.7) also shows the available solar energy but the main difference is that solar azimuth angle is considered during calculation, hence much lower available power. If one tilt the solar panel during charging to face the solar's ray the available energy will be close to bells instead of straight lines. According to the calculation it can be concluded that there will be always a minimum of 0.45kWh energy available to the car using current specifications and scaling factor.

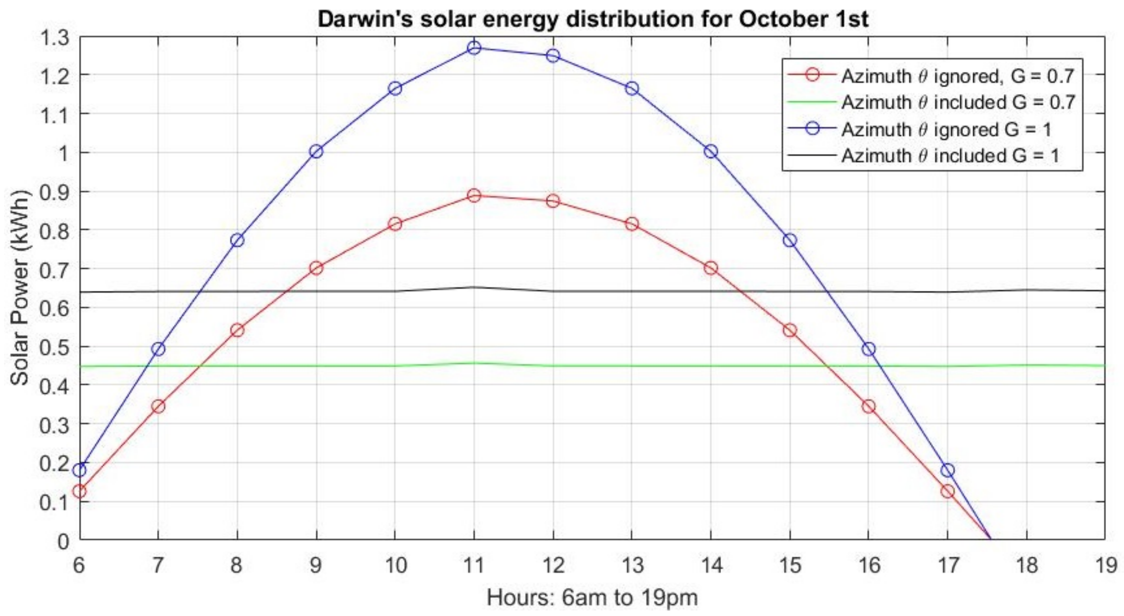


Figure 5: Solar power in 1 day

4 Car mechanical model

The forces acting on the solar car is:

$$m_v \frac{d}{dt} v(t) = F_t(t) - (F_a(t) + F_r(t) + F_g(t) + F_d(t)) \tag{14}$$

Where F_t is the force generated by the traction system, F_a is the aerodynamic drag, F_r is the rolling resistance and F_d is a disturbance force that summarizes all other effects such as forces at bearings etc.
As the solar car will be driven at a constant speed most of the race, the term $m_v \frac{d}{dt} v(t)$ can be neglected. The term F_d is relatively small when compared to the other forces and has therefore also been neglected in the following calculations.

4.1 Air drag

The aerodynamic friction that arises is due to air flowing along the surface of the car. It is depending on the velocity of the car, the frontal area and the drag coefficient[1] of the car. The aerodynamic friction is one of the bigger affecting factors on the car and is defined as:

$$F_a(t) = \frac{1}{2} * \rho_a * A_f * c_d * v^2 \tag{15}$$

Where ρ_a is the density of air, A_f if the frontal area of the car, c_d is the drag coefficient and v the velocity of the car.

4.2 Rolling resistance

Rolling resistance is the force that resists the motion when a body rolls on a surface, in this case, the force is caused by friction of the tires on the road. The rolling resistance is defined in various ways, but one equation repeatedly found in papers [4],[5] is the following:

$$F_r = c_{rr} * m * g = (c_{r1} + c_{r2} * v) * m * g * cos(\alpha) \tag{16}$$

The term $cos(\alpha)$ taking slope angle into account can be neglected since since even an unlikely case of 10% slope (10% = 5.7 deg) would only affect the normal force with approximately 0.5%.
c_{r1} is a static rolling coefficient that is depending on the type of tyre, the road surface etc. while c_{r2} is the dynamic rolling coefficient that is dependent on the number of wheels and the velocity of the car.
Some papers use a constant value of c_r [4] while other consider both c_{r1} and c_{r2} that is depending on the velocity. The different friction coefficients are usually specified by the tire manufacturer but the coefficients for the Solar car in Eskilstuna could not be found. G.Tamai [6] provides parameter values for the different coefficients of the best tire for this application at that time (Michelin Radial tubeless, 1999) :

$$c_{r1} = 0.0023; c_{r2} = n * 4.1 * 10^{-5} (m/s)^{-1} \tag{17}$$

Where n is the number of wheels in contact with the road.
These parameter values are only valid on smooth, dry asphalt and may need adjustment to be accurate on other surfaces. Since the car wheel friction coefficients are not certain of the MDH Solar car, the following simplified equation (excluding the dynamic rolling resistance) for the rolling friction will be used:

$$F_r = c_{rr} * m * g \tag{18}$$

[1]Drag coefficient is depending on the shape of the car

4.3 Gravity force

The gradient force acting on the car when driving in a sloped terrain can is determined by:

$$F_g = m * g * sin(\alpha) \tag{19}$$

Where α is the angle of the road slope at a certain location.
The gravity force F_g is acting in the opposite direction of movement when α is positive and in the direction of movement when α is negative.

4.4 MDH Solar car model

The power consumption is calculated as:

$$P = F * v \tag{20}$$

Where F is the force affecting the car and v is the velocity.
By combining equation 15,18 and 20 we get the following:

$$P = (\frac{1}{2} * \rho_a * A_f * c_d * v^2 + c_{rr} * m * g + m * g * sin(\alpha)) * v \tag{21}$$

As many of the parameters and coefficients are not measured/determined for the MDH Solar Car in the time of this report, due to the fact that the car is not completed, the following estimations has been made.

- $m = 280kg$ (Based on predictions from the construction team)
- $g = 9.81$
- $\rho_a = 1.184$
- $A_f = 1.2m^2$ (Based on estimations from the design team)
- $c_d = 0.13$ (Based on rough simulations and with comparison to other teams [7])
- $c_{rr} = 0.003$ (Based on comparison to other teams [4], [8])

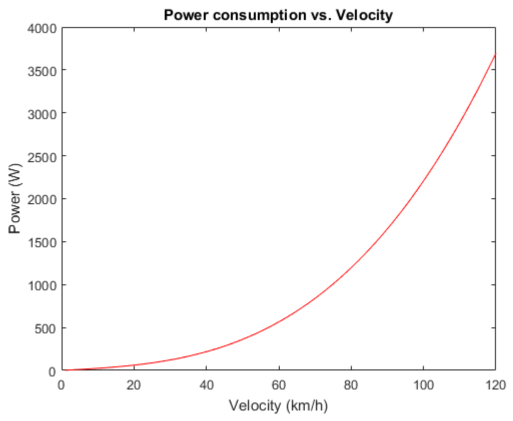


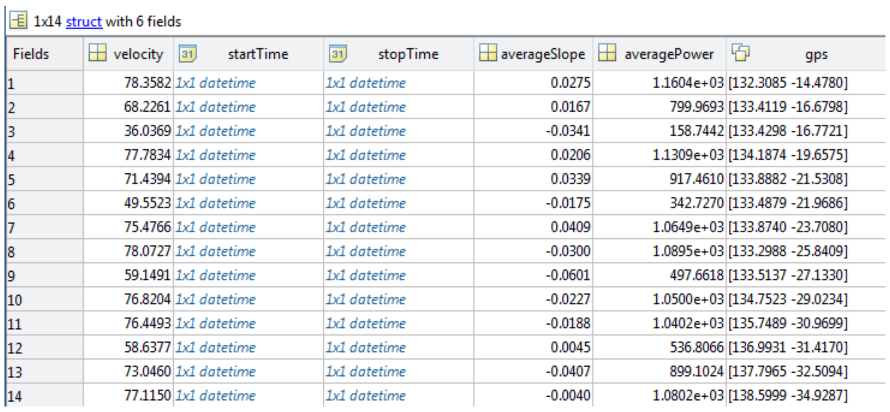
Figure 6: The power consumption plotted against the velocity.

Once the parameters and coefficients are set the power consumption can be calculated depending on the speed and slope angle of the road.
If not considering the slope angle of the road, the power consumption can be plotted against the velocity of the car (figure 6).

5 Velocity optimization

It is critical to be able to monitor the State of Charge (SoC) of the battery during the race and to have some sort of strategy.
How fast the car can be driven during the race depends on the amount of incoming solar power as well as how much the battery is draining.
The main purpose of the battery system in this race is to act as a buffer for the incoming solar power. But in order to utilize all available power, the battery should be slightly drained each day, ending up at ”zero” capacity at the finish line. This is a hard task as many parameters play role on how much the battery is draining and some parameters, such as oncoming weather conditions, are hard to estimate.
Two programs, with very different approach, has been made in order to give an approximation of the optimal velocity for the race. In the following sections, the two approaches are explained and are here by referred to as ”Velocity optimization by section” and ”SoC tracking”.

5.1 Velocity optimization by section


1x14 struct with 6 fields

Fields	velocity	startTime	stopTime	averageSlope	averagePower	gps
1	78.3582	1x1 datetime	1x1 datetime	0.0275	1.1604e+03	[132.3085 -14.4780]
2	68.2261	1x1 datetime	1x1 datetime	0.0167	799.9693	[133.4119 -16.6798]
3	36.0369	1x1 datetime	1x1 datetime	-0.0341	158.7442	[133.4298 -16.7721]
4	77.7834	1x1 datetime	1x1 datetime	0.0206	1.1309e+03	[134.1874 -19.6575]
5	71.4394	1x1 datetime	1x1 datetime	0.0339	917.4610	[133.8882 -21.5308]
6	49.5523	1x1 datetime	1x1 datetime	-0.0175	342.7270	[133.4879 -21.9686]
7	75.4766	1x1 datetime	1x1 datetime	0.0409	1.0649e+03	[133.8740 -23.7080]
8	78.0727	1x1 datetime	1x1 datetime	-0.0300	1.0895e+03	[133.2988 -25.8409]
9	59.1491	1x1 datetime	1x1 datetime	-0.0601	497.6618	[133.5137 -27.1330]
10	76.8204	1x1 datetime	1x1 datetime	-0.0227	1.0500e+03	[134.7523 -29.0234]
11	76.4493	1x1 datetime	1x1 datetime	-0.0188	1.0402e+03	[135.7489 -30.9699]
12	58.6377	1x1 datetime	1x1 datetime	0.0045	536.8066	[136.9931 -31.4170]
13	73.0460	1x1 datetime	1x1 datetime	-0.0407	899.1024	[137.7965 -32.5094]
14	77.1150	1x1 datetime	1x1 datetime	-0.0040	1.0802e+03	[138.5999 -34.9287]

Figure 7: Velocity optimization by section results

The first program had an approach where the whole race is divided into smaller sections. On the WSC race route, there are nine obligatory control stops. All the stops are about equally spread with approximately 300km in between. This approach tries to maximize the speed of the car in between these landmarks considering the speed limits, elevation of the road and incoming solar power during the race days. It is an iterative process and the pseudo code for it can be seen in Algorithm 1.
The process initializes the optimal velocity as the average of the speed limits of that section of the race. This is the fastest one legally could drive and also a good aim.
Once the velocity is set we can calculate the amount of time it will take for the solar car to travel this section as:

$$t = \frac{s}{v} \tag{22}$$

Once the driving time is determined, the ”collected” solar power can be calculated. The solar power is calculated as an average of energy at a predetermined number of location/time points along the way.
Once the solar power is known, the velocity can be recalculated using the dynamic model of the car (equation 21).
The new velocity will give a new driving time and therefore a new amount of solar power. The process is then iterated until the velocity converges and the velocity error between two iterations is adequate.
This process is executed for all the predefined sections of the race and an ”optimal velocity” for that section as well as starting- stopping time etc. is saved in a structure, see Figure 7. Why some of the velocities are very small compare to others are because those sections are ”end of day

sections" where the solar car can only drive a shorter distance before the clock is 17. The solar energy at the end of the day is very small and so the velocity as well.

As mentioned earlier, the race is sectioned according to the obligatory stops of the race and are therefore quite big. The program is made in such way that by changing some numbers, defining start and stops of the sections in the data set, would give the possibility to decrease the size of the sections and make the velocity estimations more useful.
The program does also consider the special case where a section stop point is out of reach with regard to the time left of the day. If it is estimated that a stop point cannot be reached before 17PM (daily stop time for the competitors according to the WSC 2017 regulations), the program will try to maximize the distance to be traveled, instead of minimizing the driving time between sections, by maximizing the velocity.
What this program does not take into account is the battery system of the car. Instead it tries to maximize the velocity and still match the power consumption with the incoming solar power.
After a discussion with Giacomo Spampinato, Senior Lecturer at MDH, about how to continue with the velocity optimization, it was concluded that a method that estimates the optimal velocity in a static way might not be very useful for the real race. This due to all the different affecting parameters that cannot be accounted for.
For a program to be useful in the WSC 2017 race it should not be dependent on static models and should not calculate a static "velocity" for sections of the race. It should update the optimal velocity continuously during the whole race, according to dynamic parameters such as the SoC of the battery. Peter Pudney presents a strategy to drive the solar car according to a predicted SoC curve instead of following a predetermined speed profile in "Optimal energy management for solar-powered cars" [9]. In this way one can ensure that the battery is not totally emptied before the end of the race, without any dependency on static models.

Algorithm 1: Velocity optimization by section Pseudo code

```
$while(stopPoint \neq finish)$
    $while(velocityError > acceptableError)$
        $travelTime = distToNext/velocity$
        $stopTime = currentTime + travelTime$
        $energy = averageEnergy(currentTime, stopTime, routeData);$
        $alpha = averageSlope(routeData);$
        $velocityNew = velocityCalculation(energy, alpha);$
        $velocityError = abs(velocity - velocityNew)$
    $end$
    $stopPoint = nextStopPoint;$
$end$
```

5.2 SoC tracking

This approach is all about tracking the SoC and trying to keep it according to a predefined drainage curve for the battery system.
A battery model for the car was built to match the Panasonic 18650GA cells characteristics. No exact data was used for this but points from the drainage curve, seen in Fig 8, found in the data sheet for the cell was extracted and used.
Since the final battery system configuration will be 29(S)x14(P), the capacity is multiplied with 14 and the voltage with 29 for the cell. The battery system curve is a stretched and mirrored copy of the cell curve. It is mirrored because it simplifies calculations in the program. A battery systems voltage decreases with the decrease of capacity in the system and vice versa.

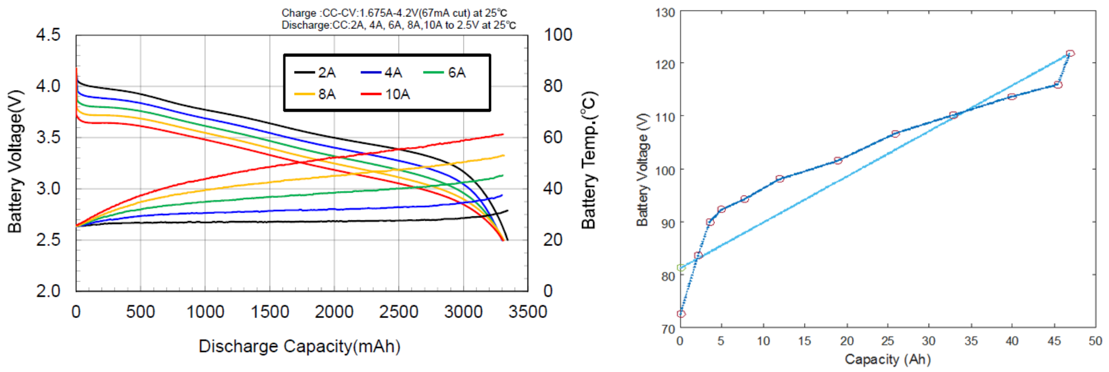


Figure 8: Panasonic NCR18650GA cell & battery system drain characteristics

By following Giacomo Spampinato's suggestions, a program was built were one can manipulate the speed of the car in real time to get a feeling of the drainage. If one is able to keep the drainage according to the predefined curve, so should a control system.
For simplicity and after discussion with Giacomo Spampinato, it was decided that a PID controller were to be used to control the speed of the car in the simulations in order to follow the predefined drainage curve.
How to define the drainage curve for each day is a hard task since many external affecting parameters, that cannot be estimated, have a great influence on the battery system's drainage. Moustafa Elshafei et al [4] based the daily drainage on road difficulties but also leaves a safety margin of 5% at the finish line. Their proposed battery status at 5PM everyday was (0.76, 0.36, 0.26, 0.05) percent of full capacity.
The proposed battery status of each day for this program has been set to (0.85, 0.7, 0.55, 0.4, 0.25, 0) percent of full capacity, with a linear drainage over the day. This is not an optimal solution and the battery status points has been chosen without any further considerations but this will be discussed further down in this section.

Algorithm 2: Velocity optimization by SoC tracking Pseudo code

```
$while(distTravelled \neq distRace)$
    $while(currentTime < dayEndTime)$
        $distTravelled = distTravelled + velocity * loopTime$
        $alpha = estimateSlope(currentLocation);$
        $collectedPower = calculateSolar(currentTime, currentLocation)$
        $powerConsumption = calculatePowerConsumption(velocity, alpha);$
        $SoC = batteryModel(loopTime, powerConsumption - collectedPower, capacity);$
        $SoCRef = calculateSoCRef(drainCurve, currentTime);$
        $error = SoCRef - SoC;$
        $velocity = PIDControl(error, loopTime);$
        $currentTime + +;$
    $end$
    $currentTime = newDayTime;$
$end$
```

A simplified Pseudo code for the program can be seen in Algorithm 2.
This approach does also consist of two while loops, just like the other approach, one while loop

that checks whether the race is completed and one that checks whether the day race time is over or not.
First it calculates the current distance traveled to keep track of where the car is in the race. Then it estimates the road slope of the current location as it has an impact on the power consumption. This is followed by calculating the incoming solar power as well as the power consumption at the current speed and location. The incoming solar power and the power consumption is then sent to the battery model together with the current battery capacity and loop time and the current SoC of the the battery is given as an output. The current SoC is then compared to a predefined SoC and an error is calculated. This error is fed to the PID controller to "generate" a new velocity in order to keep the SoC at the predefined curve.
This is just a simplified Pseudo code and the real program also handles the obligatory stops and recalculates the predefined curve after 30 minutes of charging at the stop.

A problem encountered was that the PID controller was not stable with the original battery model. This is due to the non-linearity in the model and so the model was linearized according to the light blue line in Figure 8.

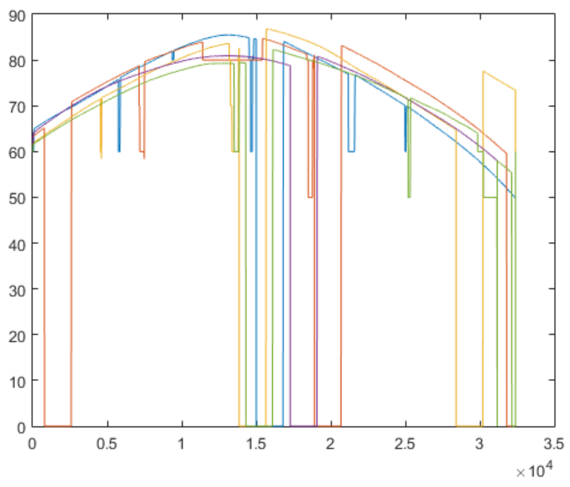


Figure 9: Resulting speed profiles from SoC tracking

The resulting velocities for each day of the race is plotted against each other in Figure 9. The velocity is similar to a bell curve over the day thus matching the solar power over the day. The lines that deviate from the curve are either because of mandatory stops or speed limits along the road.
One thing to comment on is that the bell shape in the speed profile might not be optimal, since a constant speed over the day should be more energy efficient. To over come the bell shape of the curve, the drainage for the battery should be defined differently (not linear). A suggested shape would be the incoming solar bell curve combined with the linear drainage curve, something similar to what is presented in Figure 10.

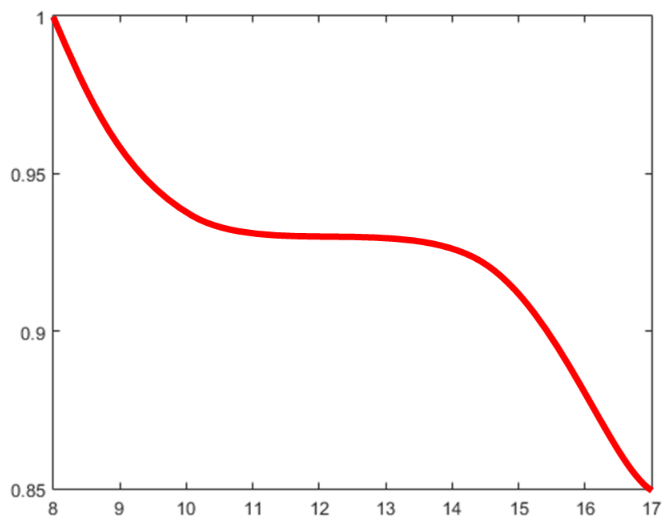


Figure 10: Suggested battery drainage curve

6 Software

It is of prime interest to develop an intuitive and easy to use GUI to control the entire process. To do so MATLAB[2] is used. Performance and scalability was taken into account whilst developing the control panel. The entire control panel was divided into smaller parts and glued together in the GUI provided by MATLAB. Although MATLAB does not support advance design patterns and conventional windows programming by nature, MVVM design pattern were widely practiced and enforced were possible to guarantee future extension hence high number of files in the project. Due to extensive and time consuming calculation of some models, MATLAB timers were used to mimic fire-and-forget function calling instead of multithreading which is much more complex in MATLAB. Timer's callback and GUI code are running preemptively and provides multithreaded-like behavior, however, all the MATLAB code is being executed in the single main thread. The software on the Arduinos in the solar car has been developed using C/C++ (Visual Studio 2015.3[3]). There are two Arduinos installed in the solar car (test platform) an Arduino Micro which is only calculating the Speed (via hall sensor) and an Ardiono Mega which do the main processing and communicate with the MATLAB program via connected xBee.

6.1 Serial manager

Peripherals such as GPS and xBee are connected to the PC using (USB — RS-232) serial port and in order to communicate with them, all the necessary codes are gathered under serial manager in the solar project. The Arduino on the other end uses the built in UART chip on the micro controller. In the configuration part of the software end user can choose port number, baud rate and terminator character. A serial port object will be created in the MATLAB and a call back function will be assigned to the On Available event of the port, upon every incoming terminator character this call back function will be executed which in turn triggers other functions based on the incoming data. Currently there are two serial port listener in the project, one which is communicating with xBee at 9600 baud rate (9600 baud is the recommended speed for xBee) and the second serial port object which talks to the connected GPS (G-Star IV) at 4800 baud rate. Once instantiated, both serial ports are constantly listening to the incoming data and act upon them by updating relevant GUI, variables and function calls.

[2]MATLAB R2016b Academic license.
[3]Visual Studio 2017 RC with Visual Macro plugin was used to compile the code

6.2 Communication protocol

There are different type of stats available on the car such as Engine RPM, Speed, Battery State and so forth, in order to send these information from the car to the PC a simple serial protocol is used to capture the relevant info. Protocol is as follow(xx is an integer value):

Incoming Data	**Information**
SxxD	This is the speed eg. S55D means speed = 55
PxxIxxIxxD	This is the PID value. First xx is P second xx is I and last is D
CxxS	This is cruse control speed eg. C50S means cruse peed = 50
TxxT	This is the throttle eg. T55T means throttle = 55
CxC	This is cruse control status, 0 means disable 1 means enabled
CxxIxxIxxIxxR	First Speed then cruse speed then collected solar power lastly charge state

Table 2: Serial Messaging Protocol

Note: It is important to remember any changes on the messaging protocol should be applied on both MATLAB codes as well as Arduino code which is responsible for gathering, formatting and transmitting the information.

6.3 Solar calculator

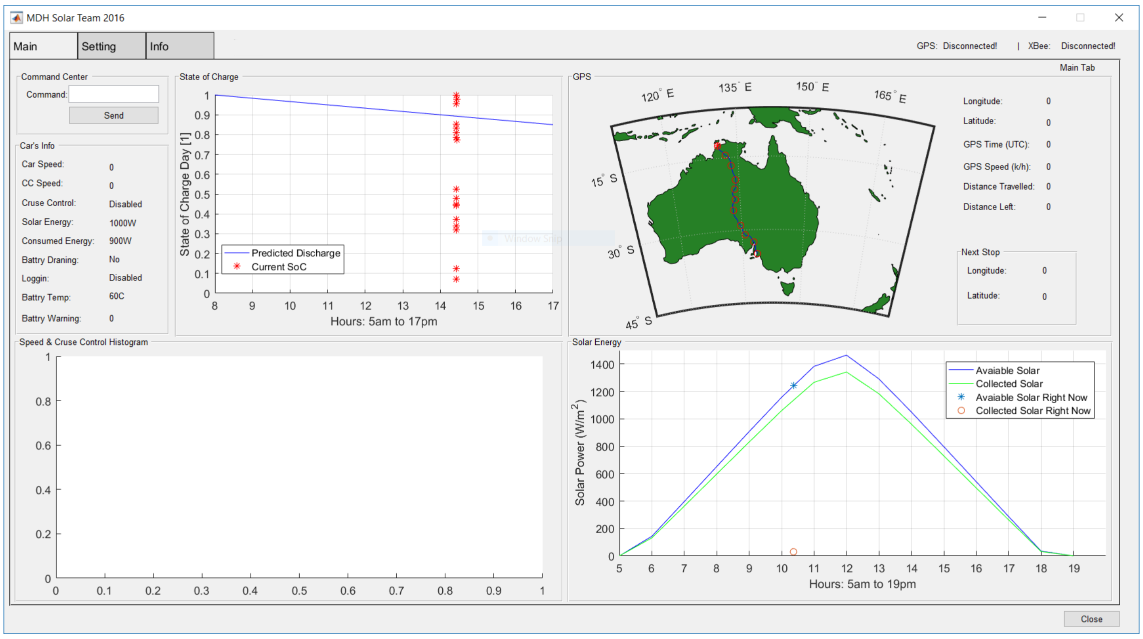
Calculation of the solar parameters[4] such as available solar power at a specific time and location, sun's azimuth angle, sun's zenith angle, daylight duration, sun rise and sun set and so forth are done in a function based manner. This will enable the end user to call a function with desired parameters and get the output directly, for instance one can pass longitude, latitude and local solar time of a location as input argument and calculate (get) the available solar power of that place just by calling the appropriate function. More information about MATLAB program can be found in the appendix.

6.4 GUI

The main GUI consist of three Tabs, main tab houses the control components and visual indicators such as Speed graph, battery stat, Map and so forth. Second tab is named Setting and user can change configuration parameters such as com port for xBee, GPS and other arguments, the last tab (Info) is the place which shows all the incoming (RAW) data on both ports without filtering, One can intercept those information and put appropriate filter for the interested value. In order to run the project, there are some initialization that needs to be done, in the main folder which contains all the m files and figure files for MATLAB program there is a file named Startup.m which should be executed first, all the initialization and preliminary configuration is done in that file. Figure below shows the main page of the MATLAB program. Top left graph shows the battery information such as state of charge and discharging, the top right graph shows the current location of the car whilst high lighting the current path plus the coordinates for the next calculated stop. Bottom left graph is the speed graph which shows current speed and cruse speed. Bottom right shows the predicted available solar power for that day and location (weather info is not considered) plus collected solar power by the solar panels installed on the car. Solar panels' specifications such as area and efficiency can be altered in the setting tab. More information regarding GUI-configurations and other tabs can be found in the appendix section.

Upon GUI load all the variables are instantiated and initialized then user can go to the setting page update the port and speed then press the connect button to start reading information from connected devices. User can set a time period e.g one second which will create a timer and send a get info command to Arduino which in turn causes the serial object to be opened and start listening to incoming data and filter them out, below is an event diagram for the events happening in the GUI:

[4]Note: All the calculated parameters were compared with online tools for accuracy and they are all within [accepted] range since there is no unanimous formula/results that can be validated.



GPS: Disconnected! | XBee: Disconnected!

Command:

Car's Info

Car Speed:	0
CC Speed:	0
Cruse Control:	Disabled
Solar Energy:	1000W
Consumed Energy:	900W
Battry Draning:	No
Loggin:	Disabled
Battry Temp:	60C
Battry Warning:	0

Longitude:	0
Latitude:	0
GPS Time (UTC):	0
GPS Speed (k/h):	0
Distance Travelled:	0
Distance Left:	0

Next Stop

Longitude:	0
Latitude:	0

Figure 11: Main GUI

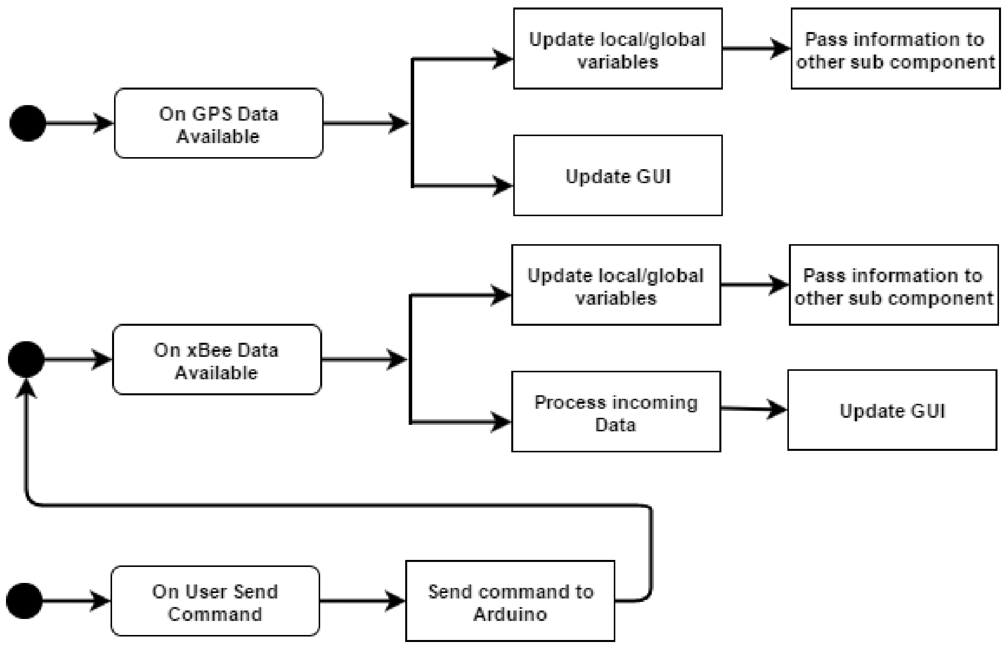

Update local/global variables
Pass information to other sub component
On GPS Data Available
Update GUI
Update local/global variables
Pass information to other sub component
On xBee Data Available
Process incoming Data
Update GUI
On User Send Command
Send command to Arduino

Figure 12: Event diagram

6.5 Embedded code for Arduinos

The Arduino Mega on the car needs to process a lot of information and communicate with the control system as well. In order to maximize performance and implement an extensible system, a simple version of Command design pattern in used to act upon incoming commands from the main MATLAB program. Application s broken down into smaller modules and each module can be called in the main loop. Main parts are as follows:

- Serial Manager: Always listen to the Rx pin for incoming Command.
- CAN Reader: Read the incoming CAN messages connected to the CAN Card.

- SD Writer: Log all the information on the SD card on the board.
- Commands: Call back functions that need to be called upon request.

Few libraries such as Spark's fun library for CAN and SD card is used to work with CAN shield which house both CAN communication and SD card, in Arduinos setup section all the sub sections will be initialized and in the main loop, it will execute codes line by line. In order to get a more efficient speed calculation a separate Arduino Micro is used, the purpose of which is only to calculate the speed. It is connected to Mega via hardware serial, at maximum speed of 250000 bit per seconds. Since Arduino is not a real-time board and there is no real-time OS on the board the execution will be a traditional sequential execution.

6.6 Cruise Control

To allow the following car to set the optimal speed, a cruise control was implemented in the solar car. The cruise control changes the motor throttle to reach a speed that is set by the MatLab software in the following car. Data is sent via Xbee to the Solar car, where an Xbee reciever is mounted. A traditional PI controller was used to regulate the throttle value. The following code describes the implemented Arduino PI loop:

```
Error = CruiseSpeed - Speed;

if (Integral < IntThresh) {  // prevent integral 'windup'
  Integral = Integral + Error;// accumulate the error integral
}
else {
  Integral = 0; \\zero it if out of bounds
}
float P = Error*(float)kP*0.01;// calc proportional term
float I = Integral*(float)kI*0.01*(1/updateTime);// integral term
Throttle = Throttle + (P + I) * Throttle;// Total drive = P+I
if (Throttle > 255){
  Throttle = 255;
}
else if (Throttle < 0) {
  Throttle = 0;
}
Last = Speed;
return Throttle;
```

Working PI values on the prototype quad with this code setup: Proportional gain = 3, Integral gain = 1. As described in the code, the PI will continuously change throttle until error reaches zero. Or until the cruise control button is deactivated.

6.7 PID calibration

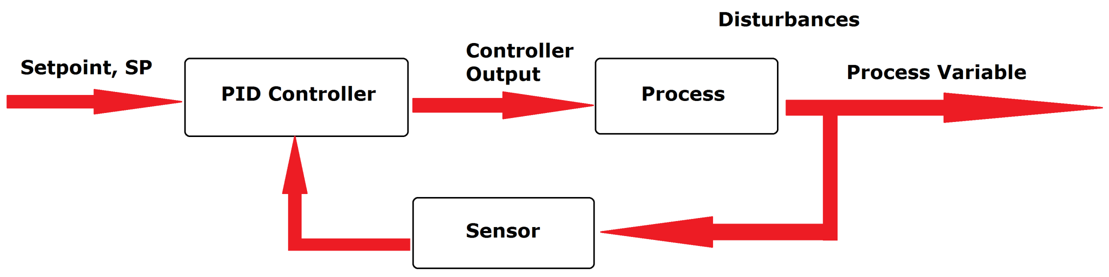


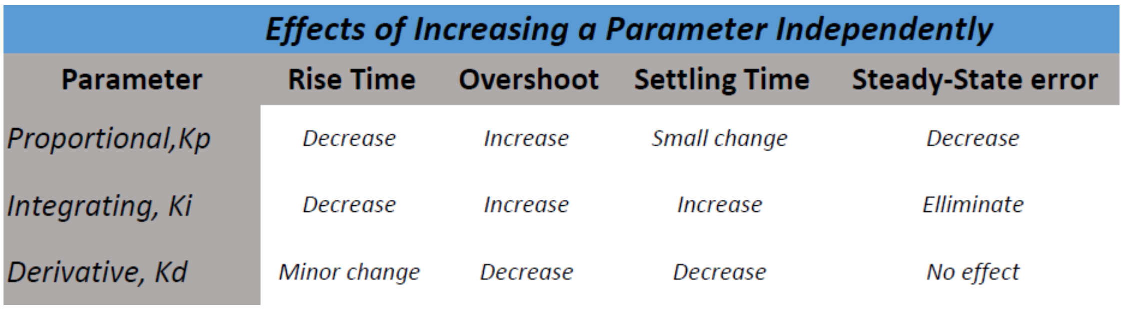
Figure 13: Schematics for a general PID controller

PID is one of he most comonly used regulators as is easy to implement, it solves most regulating problems and it is easy to tune. A PID control system is a closed loop system, it is system with a feedback. A closed loop system relates the current value in comparison with the value from last iteration. Therefore it is imperatve that the feedback relates to the PID frequency, that the feedback update frequency is higher than the PID frequency. PID in depth:

- Proportional value (P), usually abbreviated by the factor Kp corresponds directly to the error e(t). A large error will return a large proportional error and thereby a large proportional segment. Theoretically, a P regulator would be sufficient for every regulator, but as most systems have a moment of inertia that needs to be taken into consideration. This will result in a steady state error which is not negligible.
- Integral value (I), usually abbreviated by the factor Ki will continuously buffer the historical error and will continuously reduce the steady state error. A PI regulator is sufficient for most system, as long as the required responsiveness is not too high. It is important to notice that,

when using an integrator/buffer, it must be reset when the value gets to high otherwise it will have to big of an impact in the resulting control signal.

- Derivative value (D), usually abbreviated by the factor Kd corresponds to the "rate of change" in the error. This increases the responsiveness of the system as the control system allows for higher (Proportional) Kp value without excessive overshoot. As Kd will reduce the proportional correction when the rate of change gets high. This implies that systems with low requirements for responsiveness would not require a D-type regulator, a PI regulator would be enough.



Effects of Increasing a Parameter Independently

Parameter	Rise Time	Overshoot	Settling Time	Steady-State error
Proportional,Kp	Decrease	Increase	Small change	Decrease
Integrating, Ki	Decrease	Increase	Increase	Elliminate
Derivative, Kd	Minor change	Decrease	Decrease	No effect

Figure 14: PID effects table

6.7.1 Integral Windup

A large change in setpoint occurs and the integral terms accumulates a significant error during the rise(windup), thus overshooting and continuing to increase as this accumulated error is unwound, (compensated by accumulating error in the opposite direction). The specific problem results in an excessive overshoot. Solution:

- Initializing the controller integral to a desired value, for instance to the value before the problem.
- Increase the setpoint in a suitable ramp.
- Disabling the integral function until the to-be-controlled process variable (PV) has entered the controllable region.
- Preventing the integral term from accumulating above or below pre-determined bounds.
- Back-calculating the integral term to constrain the process output writing feasible bounds.

7 Car electronics

7.1 Components

Reading through the regulations it was found that the following components are necessary.

1. Turn indicators
2. Stop lamps
3. Horn
4. Instrumentation

All the lamps must be compliant with UNECE Regulations 6, 7 and 37, or the SAE/DOT equivalents. Which has to be proven with either documentation or markings on the lamp. After contact with the Swedish ”Transportstyrelsen” it was confirmed that the E-mark is approved according to the UNECE Regulations and so all lamps purchased are E-marked.

The regulations for the instrumentation are that the following information must be provided to the driver at all time:

- The speed of the solar car
- Whether the direction indicators are operating
- Whether the hazard lights are operating
- Energy storage system warnings
- Electronic rear vision images (if fitted)

Some research was done in how this information best is presented to the driver. Commercial speedometers for motorcycles were discussed as an option but since more information than the speed needs to be presented we decided to use a bigger LCD screen to present all the necessary information. An affecting factor was that two LCD screens of the right size were available in the lab.

7.2 Dashboard

This dashboard was made with the intention to be placed in the solar car. Since the design of the solar car is not completed at this time, the final design of the dashboard may be different. This dashboard will be placed and used on the quad for simulating the final electrical system and depending on the circumstance, it may be used in the solar car. The motor control actuators are not considered in this design since the size and shapes are unknown.
The final construction is made of milled aluminium, mainly to reduce weight but also because of the milling machines material limitations.

7.2.1 Component placement

The components are placed fairly close to one another to reduce the size of the dashboard, an exception is the emergency stop because of safety reasons. The components that will be more frequently used are placed further to the left and the display is placed higher up so that nothing is blocking the view. All the mounting holes are made slightly bigger than specified in the data sheets to make sure that the buttons and mounting screws will fit. The dashboard has four mounting holes, one in each corner, for fastening in the car.
The components placed on the dashboard:

1. Display

2. Potentiometer acting as a dimmer for the display
3. Turn switch
4. Turning indicator
5. Hazard light switch
6. Hazard light indicator
7. Driver switch
8. Horn switch
9. Arduino switch
10. Emergency button

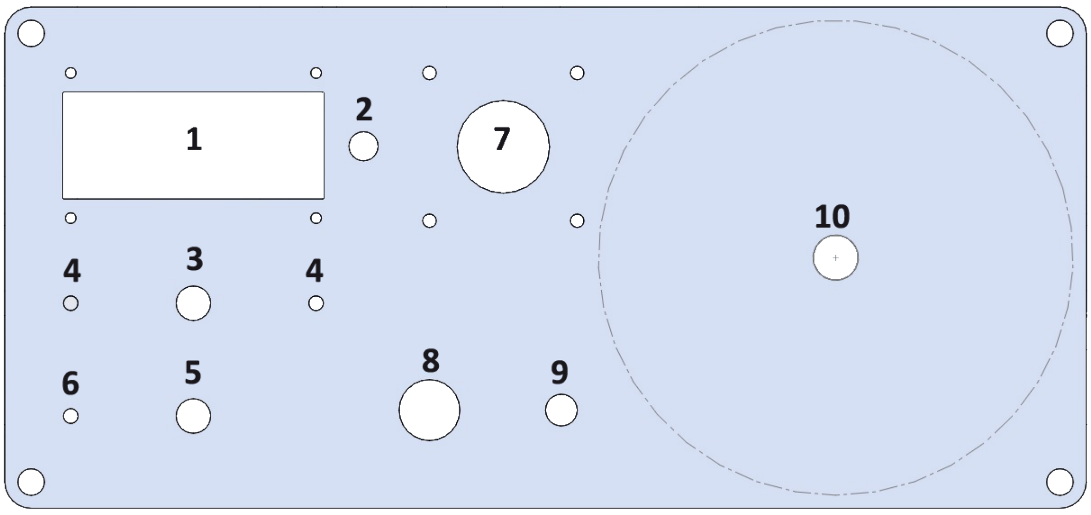


Figure 15: Dashboard layout

7.3 Fuse Box

The Solar car needs a power distribution box that can also contain fuses. There where several ideas for solving this problem, either a fuse box would be bought or one would be designed and manufactured. The second option was chosen since it would give greater control over the shape, size and weight of the fuse box. Unfortunately some problems appeared during the manufacturing process and the decision to simply buy a fuse box was taken, but for future reference the design of the original fuse box will still be documented. The entire box would be made out of polyurethane since this is a fairly light material that is easy to mill which would simplify the manufacturing process. The dimensions of the box is 218x132x99 mm, see figure 16 , the top of the box has several 5 mm holes that are meant for screws to attach the lid which would be made out of 4 mm thick Plexiglas, see figure 17. The box would weight approximately 607 grams and the lid 137 grams.
The Fuse box was never manufactured because of various reasons such as milling machine access etc. Instead a waterproof fuse box for four circuit breaker was bought at Biltema and insted of connecting the positive voltage with ring terminals, bolts and nuts, a screw rail that fitted inside the fuse box was bought.

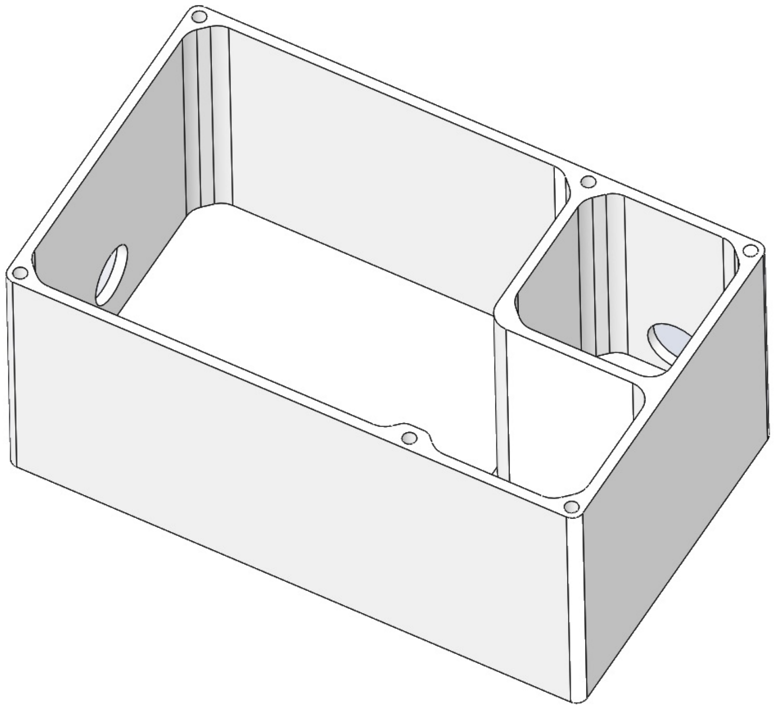


Figure 16: Fuse Box

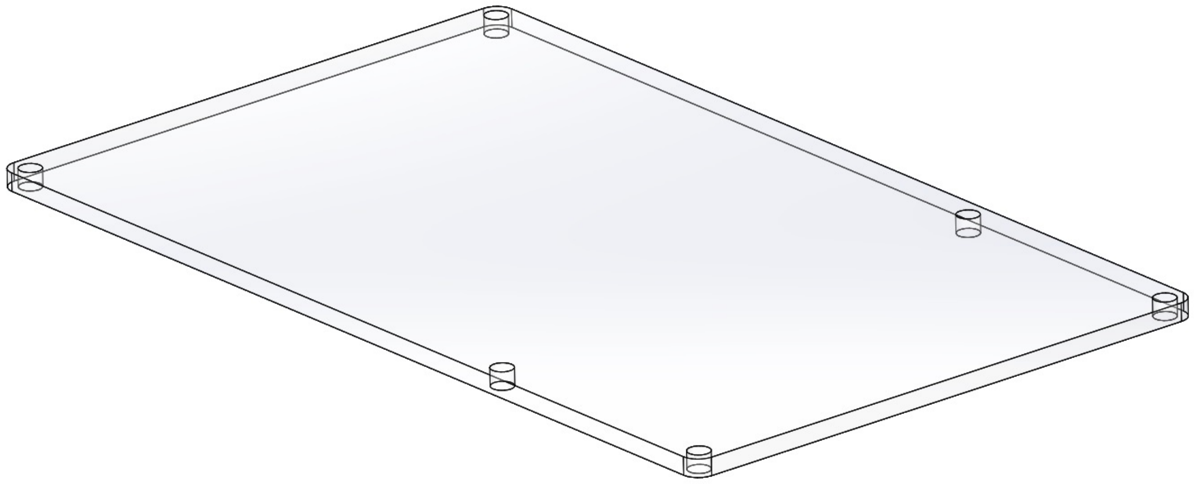


Figure 17: Plexi glass lid

7.3.1 Power distribution part

The part of the box that was meant to supply up to 122 volts and also be the ground connection. This part of the box consists of two chambers that have a 11 mm hole that is meant for a 40 mm long M10 bolts that would be the connection point. Uninsulated receptacles was used to connect the cables to the bolts, the receptacles have a sectional area of $16mm^2$ and the inner diameter is 10.5 mm. The floor of the power distribution part is thicker and two larger holes are cut halfway through the floor to fit the heads of the bolts, as seen in figure 18.The side of the chamber that houses the ground bolt has a 22 mm hole to allow cable access.

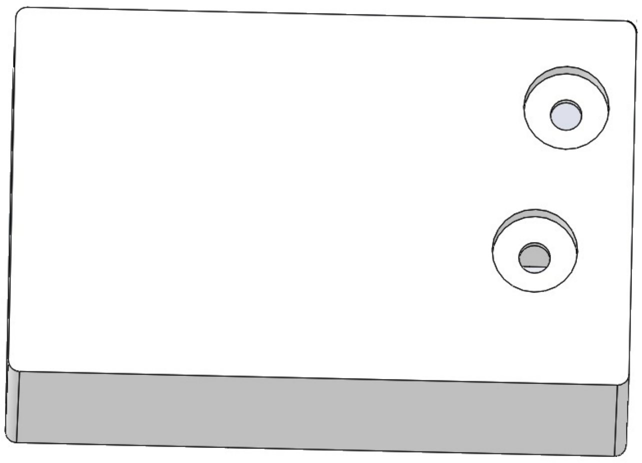


Figure 18: Floor of power distribution chamber

7.3.2 Fuse storage

The fuse storage is a larger chamber with a metal rail to mount the fuses. The dimensions of the fuses are 76x18x90 mm so the chamber has to be fairly large, the dimensions of the chamber is 146x124x95 mm. The far wall of the chamber has a 22 mm hole to allow for cable access.

7.4 Schematics

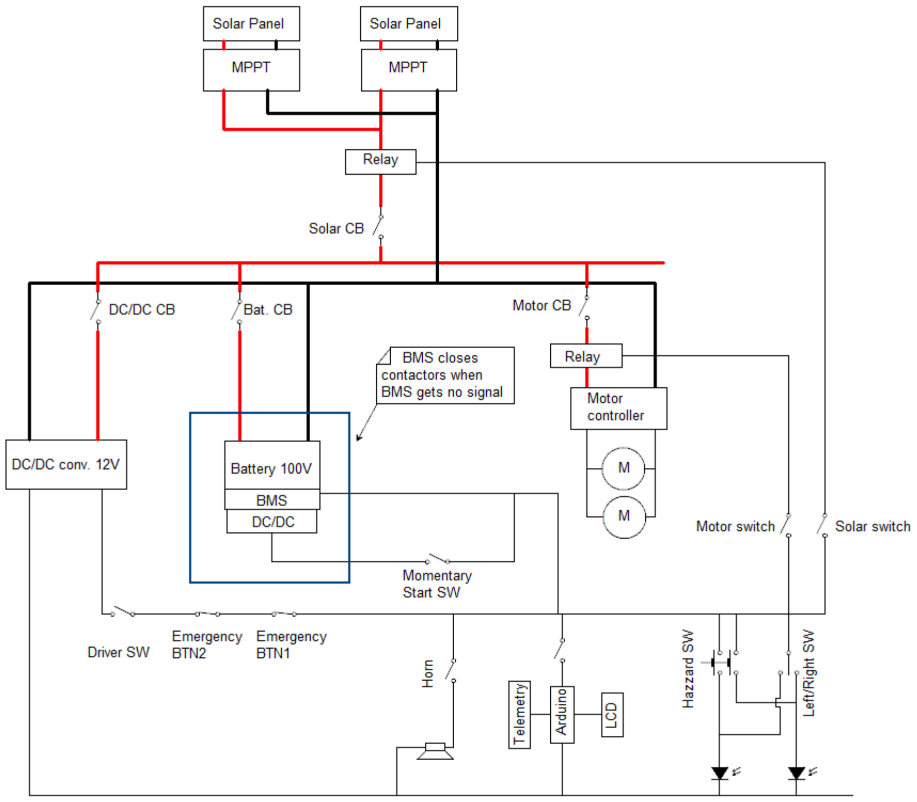


Figure 19: MDH Solar Car overview schematics

According to all the requirements of the race an overview schematic of the electrical system was made. Also schematics for the sub systems are presented in the following sections.

7.4.1 Overview schematics

The Overview schematics for the MDH Solar Car is presented in Fig 19. As can be seen in the overview schematics, Fig 19, the battery system includes an internal dc/dc converter acting as a "start battery" for the car. The internal dc/dc will supply the BMS with enough power to close the contactors, within the battery system, and therefore start the car.
Critical components of the cars system such as relays, circuit breakers and dc/dc converter (Fig 19) has been chosen and ordered.

Relay OMRON G9EA1B12DC.
The relays are of industrial type and are of the type "normally open" as a malfunctioning should break the circuit. They are made for this kind of application with high-capacity DC Loads and High-voltage DC Circuits. The G9EA1B12DC model is rated for a load up to 100 A at 120 VDC, which is higher than needed as the biggest load in our system will be approximately 50A at 120VDC.
This relay was chosen for the solar panel as well. The downfall of using this high power relay is that it has a power consumption of approx 5W and so further investigation for other relays might be beneficial.
At the time of the order it was not noticed that the relay has a mounting direction. Since the power will flow in both directions from the motor system, the relay must work in both directions. In the data sheet it is stated that the switching characteristics specified cannot be assured if the terminals are connected with reverse polarity. Reversing the polarity will not harm the relay nor the system and the switching mechanism will work but the characteristics, such as switching time, cannot be guaranteed.
Reversing the polarity was tested in the lab and worked just fine. In our case it does not matter if the characteristics are changed in case of motor braking. As long as the relay is able to open the circuit when the emergency button is pressed no problem will arise. The relay located at the motor system is there for redundancy as the battery system it self has contactors that will open when the car is turned off or put into safe state. The relay and redundancy is needed since Wamtechnik mentioned that, if the power consumption is around 50A and the contactors try to open, they will probably get welded. To overcome this scenario, the power consumption will be cut off and the relay will open, before the contactors do.
The motor regeneration cannot give such a big amount of power, according to the manufacturer but this have to be tested.
One thing to keep in mind is that a delay circuit might be needed depending on how fast the BMS can close the contactors within the battery system. Tests should be executed once the battery system is delivered.

Circuit breakers ABB S201M-C50UC, S201M-C8UC, S201M-C25UC.
In the motor system data sheet it is stated that a Non Fuse Breaker (NFB) at the positive line from the battery system to the motor system must be used. And the NFB must be of DC type. According to the manufacturer a 50A breaker should be used.
It was also decided, by the involved project members, that all of the components on the high power bus should be protected with a circuit breaker of the right sizing. The decision is mainly based on the schematic design in "A Solar Car Primer" [10]. The components connected on the high power bus are: Motor system, battery system, MPPTs and dc/dc converter.
The dc/dc converter has a max output current of 6.25A and a over current protection of 150%, and a 8A circuit breaker (S201M-C8UC) was ordered, but the current is not the same on the high voltage bus thus: $6.25A * 12V \approx 75W$
$\frac{75W}{120V} = 0.625A$ and so this circuit breaker might need to be exchanged.
The solar panel circuit breaker was ordered before the correct solar cells to use was decided. It will break the circuit at 25A even though the solar panels together can give a "shorted circuit"

current of approximately 13A thus this circuit breaker might need to be exchanged.
The circuit breakers for the motor system and the battery system will both break the circuit at 50A (S201M-C50UC) since that was what was recommended from the motor system manufacturer.To have both a circuit breaker at the battery system and the motor system can seem kind of redundant but was decided to use with the argument that other teams have it [10].

DC/DC converter XP POWER RCQ75110S12.
The dc/dc converter converts 120V to 12V which is necessary for the car electronics, communication etc. This dc/dc converter is also what will be used internally in the battery system.
It has been chosen because of its big temperature range, efficiency, size and simplicity to use. The converter has a maximum output current of 6.25A which should be enough to supply all the car electronics and communication. The only downfall was that the converter did not have any robust connectors and because of this reason a PCB has been manufactured as explained in section 8.4.
The converter has only been bench tested and converts 120V to 12V, as it's suppose to. It has not yet been integrated nor tested together with the real system.

7.4.2 Sub system schematics

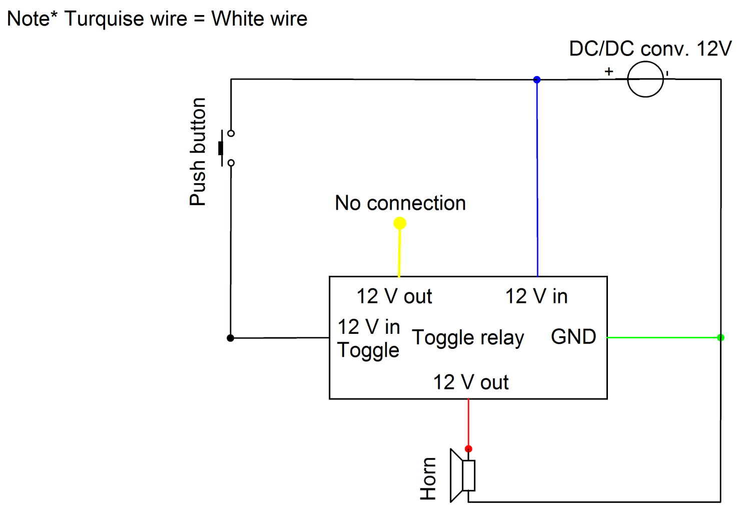


Figure 20: Schematics for the horn system

Horn The following descriptions refer to the schematic in figure 20. The horn system obtain power from the DC/DC converter mounted on the quad and rely on a toggle relay which opens on a 12V input. This relay have 5 wires with the following functions;

Blue wire is the main input for relay and should be directly connected to the DC/DC converters 12V output.

Black wire is the input which controls whether the relay is open or closed, this wire should be connected to the horn button which is connected to the DC/DC converters 12V output. This way the relay will obtain 12V on this input whenever the driver pushes the horn button.

Green wire is the ground (GND) input for the relay and should be directly connected to the DC/DC converters GND output.

Red and Yellow wire are the outputs of the relay which both deliver 12V each when the horn button is pressed. Only one output is needed here so the Yellow wire should not be connected. The Red wire should be directly connected to the horn, and the horn should then be connected to the GND output of the DC/DC converter. This way the voltage level becomes 12V over the horn whenever the driver pushes the button.

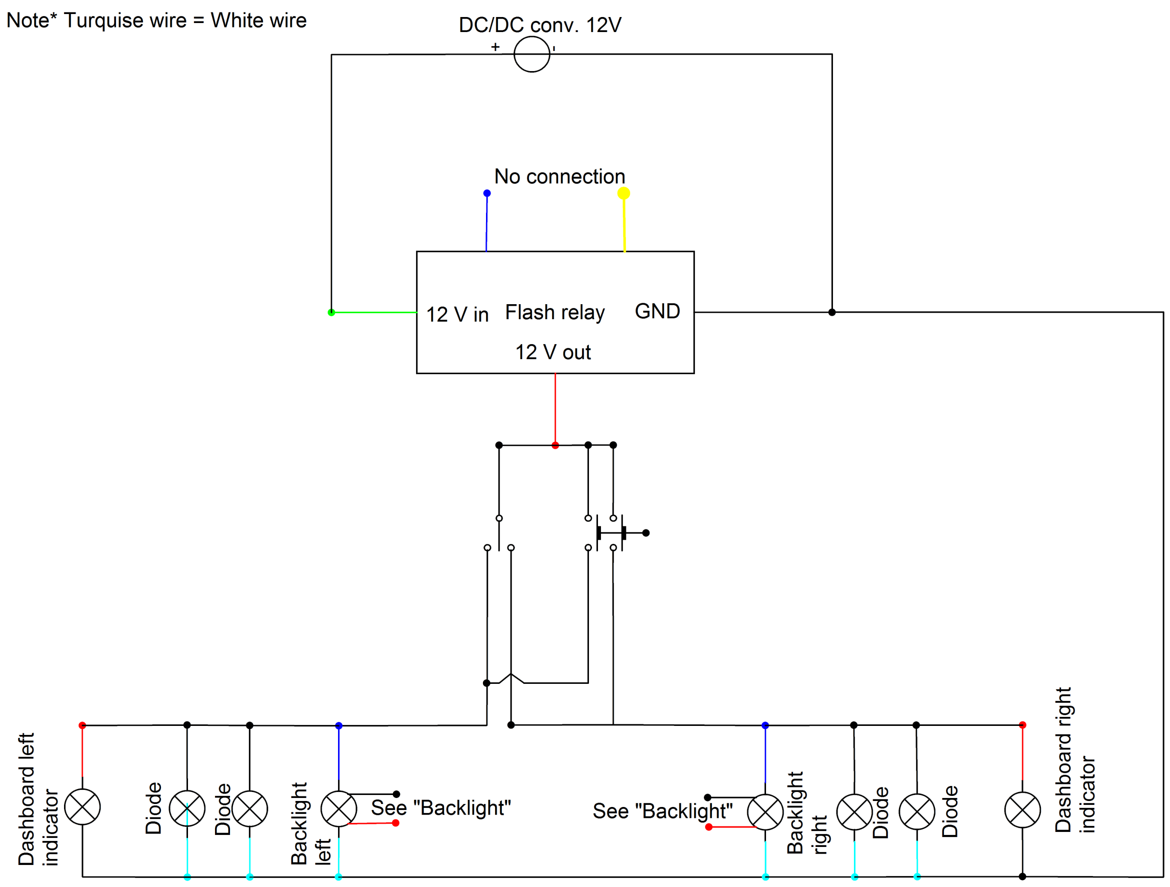


Figure 21: Schematics for the turn indicator system

Turn indicator The following descriptions refer to the schematic in figure 21. Just like the horn system, the turning indicator system is supplied from the DC/DC converter. This means that both these systems are connected in parallel with each other. The turning indicator system rely on a flash relay which toggles the output between sending 12V and 0V when supplied with 12V. This relay have 5 wires, but only 3 wires should be connected.

The Blue and Yellow wires are not connected to anything on the relay and can thereby be ignored or cut-off.

The Green wire is the input supply and should be directly connected to the DC/DC converters 12V output.

The Black wire is the GND input for the relay and should be directly connected the GND output of the DC/DC converter.

The Red wire is the output of the relay which delivers 12V when connected to a load (this load must also be connected to the common GND). This wire should be connected to the turn indicator switch and the hazard light button which are connected parallel to each other. The turn indicator switch have two outputs and three modes. Mode 1 means that none of the outputs are activated, mode 2 activates one of the outputs and mode 3 activates the other. Each output for this switch should be connected to all the turn indicator diodes on one side of the quad. Each side have four diodes; one dashboard indicator, one diode in front of the quad, one diode on the side, and one backlight turn indicator. These diodes should all be connected in parallel with each other to the output of the switch. The diodes should also be connected to the to the GND output of the DC/DC converter. The diodes on the other side of the quad should be connected the same way, but connected to the second output of the switch. Note that it is the Blue wire of the left and right backlight which should be connected to the switch, the black and red wire are for the backlight system. So when the driver activates the switch the relay will start to toggle between sending 12V and 0V to the diodes which will make the diodes on one side blink.

The hazard light button have two inputs and two outputs. The inputs should be parallel connected to the 12V output of the relay. One of the outputs of the button should be connected to one of the outputs of the turn indicator switch, and the other output of the button should be connected to the second output of the switch. This means that whenever the driver pushes the hazard light button all of the diodes will be parallel connected to the 12V output of the relay and thereby blink simultaneously.

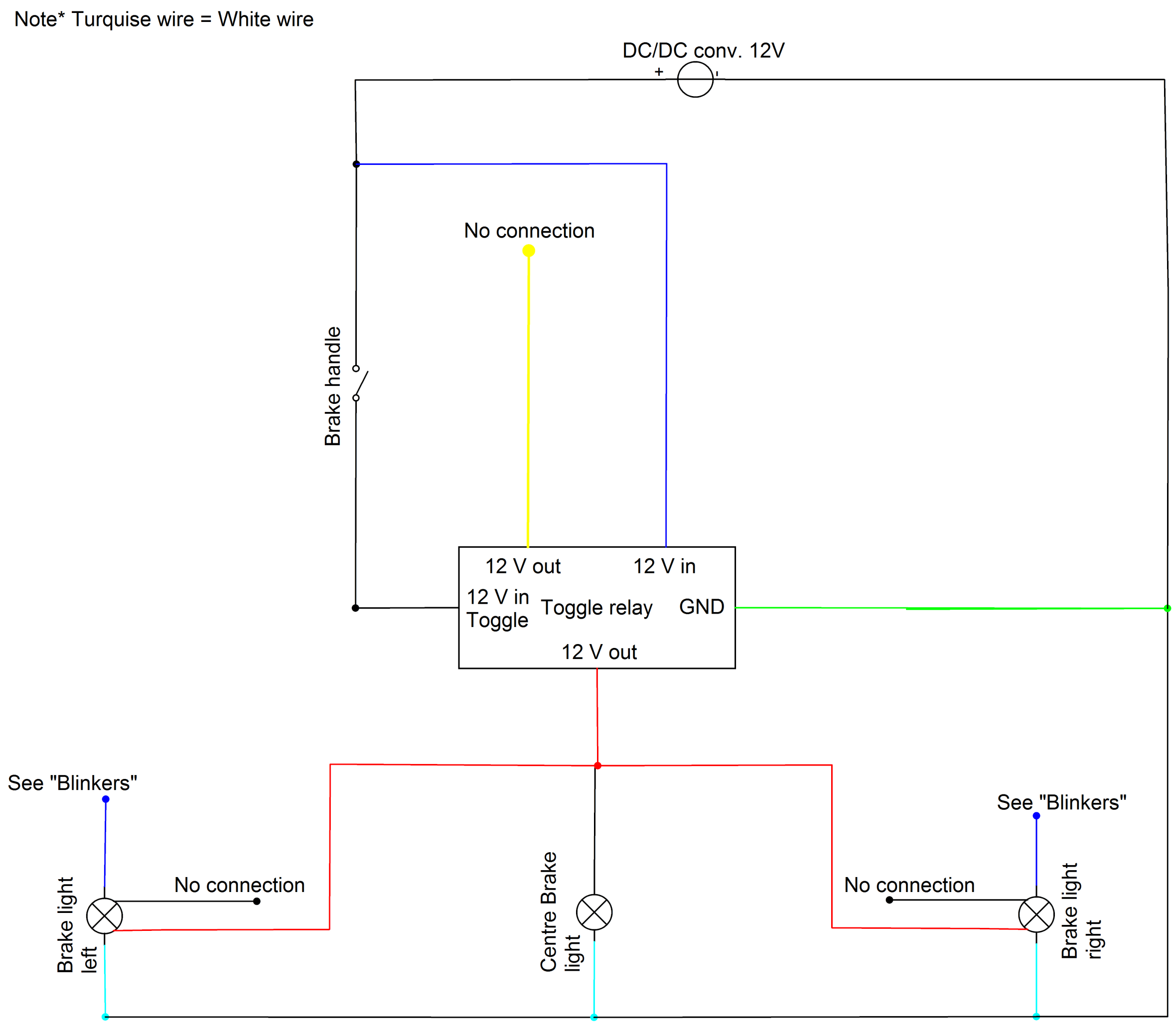


Figure 22: Schematics for the brake light system

Brake light The following descriptions refer to the schematic in figure 22. Just like the above mentioned system this system should be connected to the DC/DC converter. This system rely on the same kind of relay as in the horn system.

The Blue wire is the supply input of the relay and should be directly connected to the DC/DC converters 12V output.

The Black wire is the input which controls whether the relay is opened or closed, and this wire should be connected to the break handle, which is directly connected to the DC/DC converters 12V output. So when the handle is pressed the toggle input is supplied with 12V.

The yellow wire should not be connected since it is not needed (this is a secondary output).

The Red wire is the 12V output of the relay and this wire should be connected the brake lights. These lights are the red wires of the left and right lights and the black wire of the middle light.

These three lights should be connected in parallel with each other so that they all obtain 12V. The white wires of all these three lights should be connected to the DC/DC converters GND output.

To conserve power, the black wires from the left and right brake lights are not connected to anything. These wires are for the back-light diodes.

The Green wire of the relay is the GND input for the relay. Initially this wire was connected to a transistor circuit and the break handle. Another system involving an Arduino Mega had to read the output from the break handle, and since the pins on an Arduino Mega cannot handle 12V the brake handle was supplied with 5V instead. The transistor circuit was then used as a switch which connected the brake relay to ground when pressed. It was decided that removing the transistor, supply the brake handle with 12V, and create a voltage divider circuit which scales the output from the handle down to 5V for the Arduino was an easier solution. Founded on this, the transistor circuit was removed, and the brake handle connected to the toggle input of the relay instead.

8 PCB

External components such as relays. lights, horn, buttons and sensors needs to be easily, and robust, connected to the main electric system. In order to make every connection reliable and safe, it was decided to make individual PCB:s for the subsystems. Each PCB has been given a name suitable for its purposes: Mega-, Adapter-, LCD-, Relay-, Front-, and Rear PCB Each PCB will be described briefly below:

8.1 LCD PCB

The LCD screen is an essential part of the driver instrumentation. As this is the case, it is important that it works no matter how bumpy the road is etc. Additional functionality includes that it must be possible to change the back-light of the screen, as current consumption is an issue. Each pin should be safely and reliably connecting with the screen and also with the Arduino regardless of the situation. A descriptive list of the connectors found on the PCB is found in appendix EA.

8.2 Adapter PCB

Break sensor, Cruise sensor, Throttle In, Throttle Out, Direct (bypass button). The sesors mentioned must be connected in a safe and reliable way to the main processing unit. Adapter PCB uses a ribbon cable to connect to the Arduino shield and allows for the sensors to be plugged in. A descriptive list of the connectors found on the PCB is found in appendix EB.

8.3 Mega PCB

The functionality of the shield include bypassing pins according to the needs of the CAN shield and routing the ribbon connectors to necessary pins input/output pins. It supports two ribbon connectors and an extra throttle output depending on where it's most easily accessed. A descriptive list of the connectors found on the PCB is found in appendix EC.

8.4 DC/DC PCB

To provide the main electrical system with the neccesary 12 volt supply, a 120 volt to 12 vol DC/DC converter was ordered. The DC/DC converter have the drawback that it did not have any robust connectors. In order to make the connectors stable, a shield was constructed. With a "on/off" switch and input and output pins. A descriptive list of the connectors found on the PCB is found in appendix ED.

8.5 Delay PCB

The delay circuit is a simple circuit with following specifications: Delay a 12 volt digital signal in the high state in 100ms, when power supply is turned off. This functionality will be used when power of the solar car is turned of and the connectors within the battery system needs to be disconnected last. A descriptive list of the connectors found on the PCB is found in appendix EE.

8.6 Micro PCB

It was decided to use an external arduino to manage the interrupt routine to save processing power on arduino mega. To make the connections as stable as possible, a shield with stable and robust connections were constructed. To make the most out of the shield, as many arduino pins as possible is routed to external connectors. A descriptive list of the connectors found on the PCB is found in appendix EF

8.7 Relay PCB

This PCB is referred to as the Relay PCB since it should be the only PCB connected to all the relays. This is to ensure that the relays can be assembled in a box along with this PCB and then easily hidden. The idea for this PCB was that it should be supplied by the DC/DC converter and then distribute the supply to the subsystems. The Relay PCB should be connected to the Front PCB and buttons on the dashboard, and should thereby be mounted in the front of the car for minimizing the wiring. A descriptive list of the connectors found on the PCB is found in appendix EG.

8.8 Front

This PCB is referred to as the Front PCB since it should be mounted somewhere in the front-section of the car, control the turn indicator diodes, and connect to buttons on the dashboard. The card shall also obtain the supply for the brake lights from the relay PCB and pass that to the Rear PCB along with the supply for the turn indicators. The main reason for this cards existence is to gather the components in the front of the car in one compact area. A descriptive list of the connectors found on the PCB is found in appendix EH.

8.9 Rear PCB

This PCB is referred to as the Rear PCB since it should be mounted in the rear of the car and control the turn indicator diodes in the rear as well as the brake lights. This PCB was mainly created to gather the supplying of the rear components in one compact area. A descriptive list of the connectors found on the PCB is found in appendix EI.

9 Motor system

Prior to this project it was decided that the electrical motor Mitsuba M1096-III were going to be used in the MDH Solar Car. In order for the motor manufacturer to manufacture the electrical motor, the following parameters was needed:

- Target max rpm (overtaking)
- Voltage at target max rpm
- Power consumption at target max rpm
- Target cruising rpm
- Voltage at target cruising rpm
- Power consumption at target cruising rpm

Since a simulation of the aerodynamic friction could not be made in time for the motor system order, the Power consumption needed to be estimated. The power consumption was estimated with the parameters presented in section 4.4.

The cruising and overtaking rpm for the car was chosen through consulting with the motor manufacturer which has a lot of insight in the competition. And a cruising rpm was set to 650 which is approximately 67 km/h. According to the mechanical model of the car, the power consumption at this speed would be approximately 750W.
The overtaking rpm was set to 750, but this speed is with the MTA (magnetizing timing adjustment) affecting which adds about 5% to the rpm. And so the maximum speed without the MTA is $750/1.05 = 714 \approx 74km/h$. The maximum speed of the car will, with the MTA, be $750rpm \approx 78km/h$. This speed is at nominal voltage = 104.4V and the voltage affect the speed of the motor in a linear way. Keeping this in mind, the maximum speed of the motor is at maximum voltage = 121.8V approximately $78 * (121.8/104.4) = 91km/h$.

10 Battery system

Finding a company willing to help and develop the battery system for the car was not an easy task. A lot of companies are interested in the project but did not, according to them, have the extra resources to put on this project.
After contacting the Danish company Lithium Balance, that specializes in battery management systems for Lithium batteries, contact information to their partner company Wamtechnik was given. Wamtechnik is a company that specializes in design and production of battery packs for special applications.
After email contact with the company, they seemed very interested in the project and were willing to help. They could offer us the Non recurring engineering (NRE) and Labour for free, as well as all the materials, without any mark-up, for free. This sounded like a fair deal as other companies could estimate the total cost for the development and production up to 400.000SEK.

10.1 Battery configuration

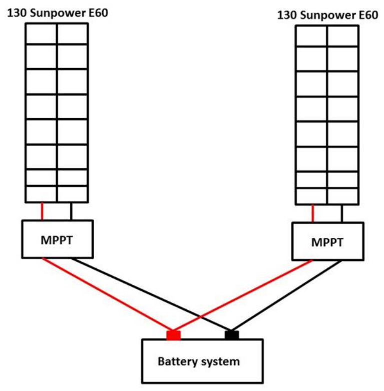


Figure 23: Solar cell setup

All the different components in the car are important in order to design the system as energy efficient as possible. As the Max Power Point Trackers (MPPTs), Battery system and motor system all work together and are connected on the same "High Voltage Bus", it is highly important to choose the correct parameters of these.
Since the WSC regulations of 2015 allowed a total area of $6m^2$ of silicon cells, it was decided in the preparatory work of spring 2016 that 391 Sunpower 125x125 cells were to be used. These cells were decided to work in a 3 panel configuration, with MPPTs from a company named Drivetek, in order to match the maximum on the efficiency curve. Since the rules changed for the WSC 2017 race and the maximum allowed area of silicon cells now is $4m^2$, one of the panels could be skipped and now 260 cells are to be used in a 2 panel configuration as can be seen in figure 23.
In the preparatory work it was also decided that a good configuration for the battery system

is $14(P)x29(s) = 406cells$[5]. After discussion with Wamtechnik it was decided that Panasonic NCR18650GA cells were to be used as they are the cells with the highest capacity/weight ratio on the market. With a max weight of 48.0g, the configuration with 406 cells would still work (According to the WSC regulations) as $406x0.048 = 19.488kg$.

The delivery of the battery system was, in the beginning of the project, estimated to the end of January by Wamtechnik but due to encountered problems and delayed quotations, Wamtechnik now estimates a lead time of 10-12 weeks.
The delivery date for the battery system is not critical. All the testing, of motor system etc, will be done with motorcycle batteries.

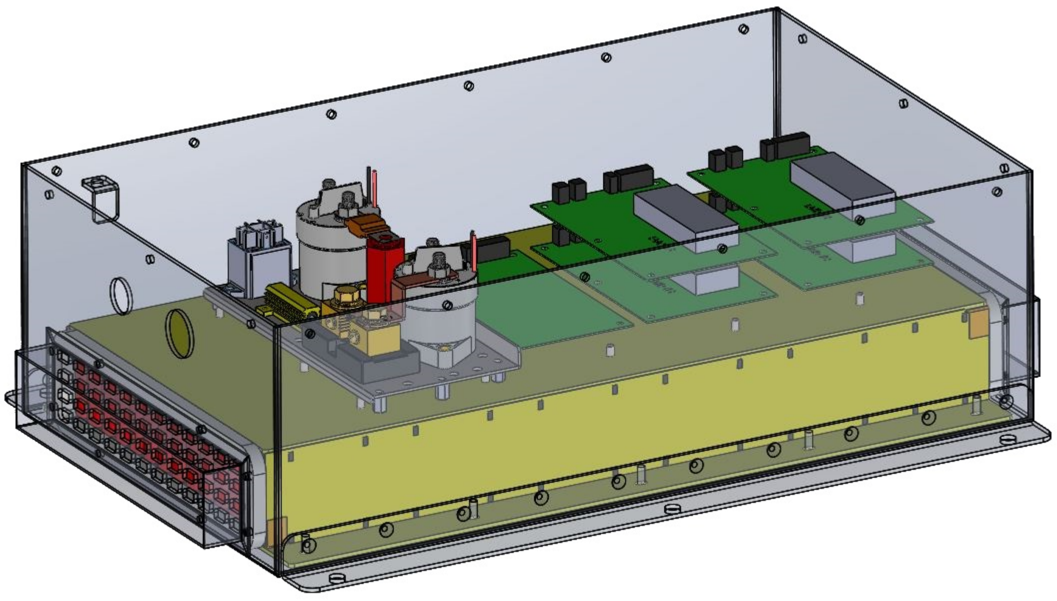


Figure 24: First battery concept

The mechanics of the battery system was decided and modeled by Wamtechnik once the configuration of cells were set. They could offer a aluminum casing for the battery pack for free as they could manufacture it in-house. An aluminum casing should be okay according to the regulations as long as the high-voltage parts are double isolated.
A concept CAD of the battery system was made and sent in the beginning of the development process and can be seen in fig 24.
The only input on the CAD model, given to Wamtechnik, was that fans should be added for cooling in case the car is standing still. The reply was that this should not be any problem to add, just that some extra relays were needed to control the fans by the BMS. The fans might block the air intake some, and so they might need to run on lower speeds even when the car is driving if the cells get too hot.

The contact with Wamtechnik has been sporadic as an email sometimes take more than a week to be answered and other times it is a matter of minutes. In the beginning of the cooperation they stated that they would help us with this project on the side of all the other projects, and so it is understandable that an email reply sometimes take time. They have always been very helpful and all the problems encountered have been solved together even though their emails can be somewhat hard to understand due to the language barrier.
We have only had email contact with the company and below are the names and email addresses of the persons of interest:

[5]Why this configuration is suitable for the system can be read in the preparation work report.

- Sebastian Ryciak, sryciak@wamtechnik.pl, projects main contact
- Jakub Zreda, jzreda@wamtechnik.pl, head of R&D
- Jacek Paliwoda, jpaliwoda@wamtechnik.pl, responsible for the mechanics
- Elzbieta Krzysztosiak, ekrzysztosiak@wamtechnik.pl, marketing
- Chuong Hoang Tran, cht@lithiumbalance.com, Lithium balance contact

11 State Of The Art

David Menasce et al.[11] discusses a methodology and considerations when designing a High Power Electrical System for a Solar car. In their design methodology the main parameters used in the selection of components are electrical efficiency, mass and power consumption. With these parameters in mind the components for the solar car were chosen. A battery system using battery cells of Lithium-Ion type and holders, separating the battery system into blocks, were designed an manufactured. Calculations showing the optimal amount of solar cells in series/parallel for a specific boost converter are made, dividing the solar cells into different panels. Problems encountered are discussed as well as the outcome of the project, which was the solar car LLandal.I that participated in the South African Solar Challenge.

Moustafa Elshafei et al. [4] presents an optimization method for a solar cars speed and battery management in the World solar challenge competition. The method tries to maximize the speed for the solar car in between different landmarks marked out by the organizer of the competition. Expected solar radiation is calculated and used to estimate the solar energy availability during the race and the dynamic model of the solar car is described. With this data a cost function is defined and the MATLAB built in search-based optimization algorithm “fmincon.m” is used to find the optimal velocities which minimizes the cost function. The results of the method are presented with graphs and conclusions are made that the presented optimization algorithm can lead to a more efficient energy management during the WSC race.

Graham S . Wright [12] describes a solar car simulation and race strategy that rely on classical analytical optimization methods. The simulations and race strategy is developed for the ”sunChief”, a solar car built by the University of Illinois for the GM Sunrayce 95. The problems when developing a race strategy is presented as well as the final strategy solution consisting of a program implemented on a Microsoft Excel spreadsheet. Different energy and power formulas were used to calculate the energy gains and losses in the system. All of this as well as weather conditions are used to derive a long term and short term strategy. Discussion on the system performance as well as future upgrades/refinements are made.

Engin Yesil et al.[7] presents a new optimization method approach based on Big Bang - Big Crunch (BB-BC) to be employed in the solar car ”ARIBA 6”, built by Istanbul Technical University (ITU), in World Solar Challenge (WSC) 2013. A litterature study was performed with the aim of determining the optimal strategy to minimize the racing time while considering the race regulations as well as constraints imposed by environmental conditions and a race model was built in order to evaluate the effects of these.
The solar car model was explained as well as the effects of the different parameters in the model. A brief overview of BB-BC optimization method was shown followed by explanation of how the method was used in in this specific application. The proposed strategy divides the WSC route into 11 segments due to regulation checkpoint stops and the optimal speed references are computed for each segment. A flowchart of the strategy optimization algorithm was shown and an explanation of the iterating method was given.
A userfriendly GUI was created where the user give inputs such as covered distance, remaining energy, day number and clock during the race and the optimal speed is computed and displayed. The result of the BB-BC optimization were retrived through various simulations and later studied

by an expert group that found the results promising.

Alexandra Boulgakov did her bachelor thesis on modelling the mechanical systems of "Sunswift IV", a solar car built by the UNSW Solar Racing Team [13]. In addition to the modelling, a current integrator device was redesigned, built and tested. The power consumption model used by many solar car racing teams are explained as well as practical strategy considerations for the WSC, such as weather uncertainty and limited model accuracy. Some research on different strategy approaches was done in order to gain insight on how the Sunswift IV strategy system could be improved.
In chapter three, the car model is built and explained. Here both the mechanical and electrical model is considered.
Many parameters are firstly estimated and later experimentally determined by a number of tests. In chapter four Alexandra explains why a current integrator is needed and an evaluation of a previous existing device made. The device is redesigned to become more robust and accurate. Bench testing showed that the device worked well and scored a high accuracy.
The WSC race that Sunswift IV competed in is then explained day by day and many observations were made by the team:

- The strategy software was very useful even though a lot of manual adjustments was required during the race
- The telemetry feedback was extremely important in order to compare calculated and real power consumption.
- The current integrator provided a practical way of tracking the actual battery state of charge.

12 Summary and discussion

The entire project has been conducted in 20 weeks with very limited resources.
A communication system mainly consisting of Xbee modules and Arduino has been built.
Software to handle incoming CAN messages within the car as well as information from various sensors and send it through the wireless connection to the following car has been made.
Software for logging all important data on a SD card has also been made.
An user friendly GUI for monitoring and manipulating critical systems within the solar car through the communication system has been built.
The complete electrical design of the solar car has been determined according to the WSC 2017 regulations and all the bigger electrical components has been chosen.
Modular and easy to install PCB's has been developed and manufactured with robust connections for the car electronic components.
A solar power model has been made with the purpose to give an idea of the amount of solar power that can be used during the race.
A mechanical car model for the MDH Solar Car has been built. Most of the parameters are estimates as the final car body design is not finished and tests must be conducted in order to get a more accurate model. The power consumption at different speeds using the estimated parameters can be seen in Figure 6.
Both the Solar model and the car model has been integrated in two different velocity optimizing algorithms that gave various results. An important fact to realize is that the results should not be taken as accurate answers to what the optimal velocity is. Multiple tests must be conducted once the real car is built in order to tune parameters and change models for the programs.
This project has built a steady base for further work that can be conducted once the solar car chassis and body is built and once all the bigger components such as motor system, battery system and "solar system" has been delivered. Future work is discussed below.

13 Future work

Control software in the solar car is currently running on Arduino platform which is not a real time system. Although the architecture of the software take into account such limitations and tries to minimize overheads, an enhancement to the processing board might be beneficial in long run. Companion software which is developed using MATLAB can also be extended to accommodate for more scenarios and a more suitable/optimized algorithm can be investigated to be used in the optimization part of the system.
An problem encountered has been noise and Electromagnetic compatibility (EMC) picked up in signal cables and PCB's. Most of the noise could be canceled using ferrite magnets on signal cables and twisted pair cabling but this is something to keep in mind when placing all the components in the real car.
As the motor system just got delivered from Japan, the next step in the project should be to buy motorcycle batteries to simulate the final battery system and test the motor system. A suggested set up for the testing batteries can be seen in Figure 25. Nine motorcycle batteries would give a nominal voltage of 108V which would be suitable for testing purposes. In order to charge the test batteries, they should be connected as can be seen in Figure 26 where the arrows symbolizes jumper cables. In this configuration, all batteries are connected in parallel and can be charged with an ordinary car battery charger.
Once the final car chassis has been built, the electrical system should be transferred from the electrical quad. And once the battery system and MPPT's has been delivered they should be integrated with the rest of the system. The CAN communication between these should be set up with the "base code" built in this project. All important data, incoming from the CAN bus and sensors, should be programmed to be logged on the SD card in the solar car and sent through the communication system to the following car for processing.
The velocity optimizing program should be integrated with the GUI in an intuitive manner, giving the user options to recalculate the battery drain curve. In the future, the idea is to integrate the velocity optimizing program with the cruise control so that the set speed is adaptive to what the

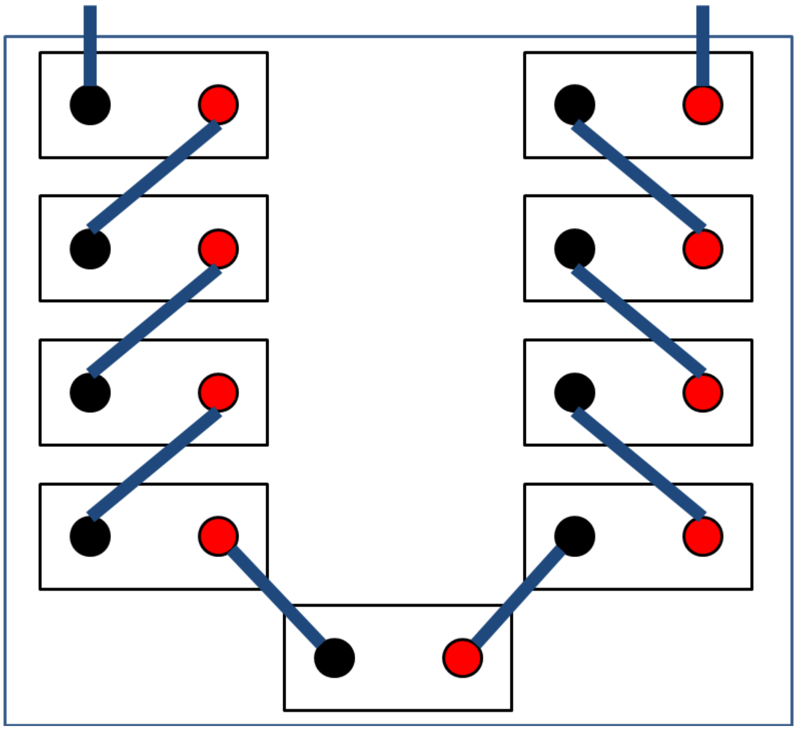


Figure 25: Suggested test battery configuration

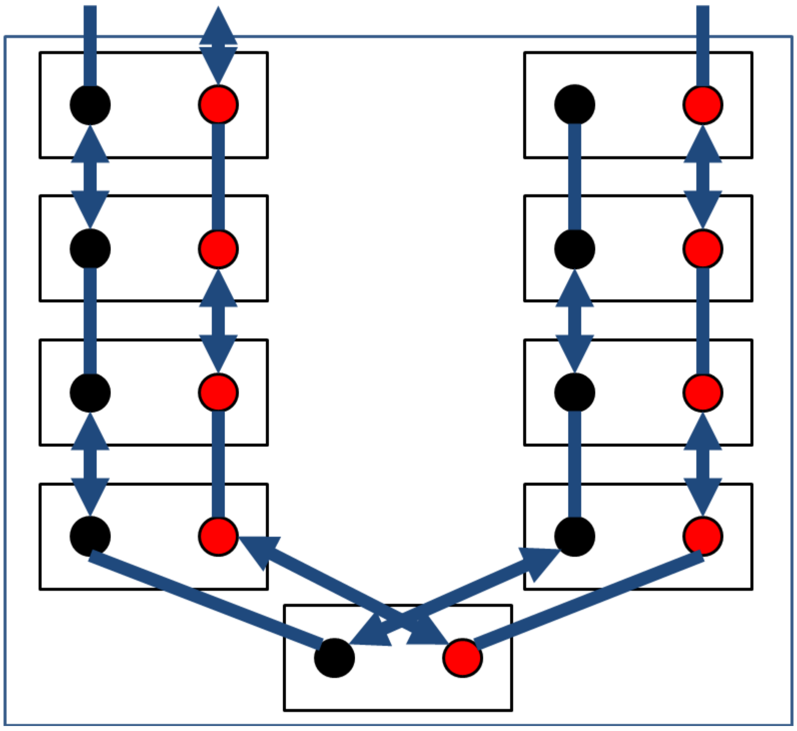


Figure 26: Suggested charge configuration for test battery

program calculates.
Now that the motor system has been delivered, the speed should be tested and measured using the hall sensor mounted on the motor. Parameters in the code, such as number of pulses per lap etc. will have to be manipulated for this.

Testing will be the key to success in this project. Once a major component in the electrical system has been delivered it should be tested, integrated and tested again together with the rest of the system. All critical components should have spare parts ready to be substituted, in this way malfunctions can be easier and faster fixed during the race.

References

[1] A. B. of Meteorology, "The equation of time," 2015, the Equation of Time. [Online]. Available: http://www.sws.bom.gov.au/Educational/2/1/14

[2] T. H. Blair, *Solar and Wind Energy.* Wiley-IEEE Press, 2017, pp. 832–. [Online]. Available: http://ieeexplore.ieee.org.ep.bib.mdh.se/xpl/articleDetails.jsp?arnumber=7794197

[3] X. Zhu, J. Yan, and N. Lu, "A graphical performance-based energy storage capacity sizing method for high solar penetration residential feeders," *IEEE Transactions on Smart Grid*, vol. 8, no. 1, pp. 3–12, Jan 2017.

[4] M. Elshafei, A. Al-Qutub, and A. W. A. Saif, "Solar car optimization for the world solar challenge," in *2016 13th International Multi-Conference on Systems, Signals Devices (SSD)*, March 2016, pp. 751–756.

[5] C. Mocking, "Optimal design and strategy for the solutra," Master's thesis, University of Twente, 2006.

[6] G. Tamai, *The Leading Edge: Aerodynamic design of ultra-streamlined land vehicles.* Robert Bentley, 1999.

[7] E. Yesil, A. O. Onol, A. Icke, and O. Atabay, "Strategy optimization of a solar car for a long-distance race using big bang x2014; big crunch optimization," in *Computational Intelligence and Informatics (CINTI), 2013 IEEE 14th International Symposium on*, Nov 2013, pp. 521–526.

[8] W. contributors. (2017) Automobile drag coefficient. [Online]. Available: https://en.wikipedia.org/wiki/Automobile_drag_coefficient

[9] P. Pudney, *Optimal energy management for solar-powered cars.* University of South Australia, 2000.

[10] E. F. Thacher, *A Solar Car Primer: A Guide to the Design and Construction of Solar-powered Racing Vehicles.* Springer, 2015.

[11] D. Menasce, M. Grobler, and P. J. van Rensburg, "High power electrical systems design for a solar car: Designing and building the solar car ilangai. i," in *2013 Africon*, 2013.

[12] G. S. Wright, "Optimal energy management for solar car race," in *Circuits and Systems, 1996., IEEE 39th Midwest symposium on*, vol. 3, Aug 1996, pp. 1011–1014 vol.3.

[13] A. Boulgakov, "Sunswift iv strategy for the 2011 world solar challenge," *Bachelor's Thesis*, 2012.

A Who wrote what

Person	**Section**
William, Mohammad	Abstract, Summary and discussion, Future work
William Palm	Introduction
William Palm	Car mechanical model
William Palm	Velocity optimization
William Palm	Car electronics - Overview schematics
William Palm	Motor system
William Palm	Battery system
William Palm	State of the art
Mohammad Heidari	Formatting
Mohammad Heidari	Communication - Hardware
Mohammad Heidari	Communication - Connecting Hardware
Mohammad Heidari	Communication - Xbee configuration
Mohammad Heidari	Communication - CAN Bus
Mohammad Heidari	Energy Optimization
Mohammad Heidari	Software - Serial manager
Mohammad Heidari	Software - Communication protocol
Mohammad Heidari	Software - Solar calculator
Mohammad Heidari	Software - GUI
Mohammad Heidari	Software - Embedded code for Arduino
Mohammad Heidari	Appendix A,B,C,D
Andreas Ramberg	Sub system schematics
Andreas Ramberg	PCB - Relay
Andreas Ramberg	PCB - Front
Andreas Ramberg	PCB - Rear
Andreas Ramberg	Appendix EG, EH, EI
Fredrik Fischer	Software - Cruise Control
Fredrik Fischer	Software - PID calculation
Fredrik Fischer	PCB - LCD
Fredrik Fischer	PCB - Adapter
Fredrik Fischer	PCB - Mega
Fredrik Fischer	PCB - DCDC
Fredrik Fischer	PCB - Delay
Fredrik Fischer	PCB - Micro
Fredrik Fischer	Appendix EA, EB, EC, ED, EE, EF
Daniel Hedlund	Car electronics - Dashboard
Daniel Hedlund	Car electronics - Fuse Box

B Sparkfun CAN-Bus Library Configuration

```
#include <mcp2515_defs.h>
#include <mcp2515.h>
#include <global.h>
#include <defaults.h>
#include <Canbus.h>

void setup()
{
  // Other configurations are omitted for simplicity
  setupCANBus();
}

void setupCANBus()
{
  HardSerial_xBeeToxBee->println("Setting up CAN");
  HardSerial_ArduinoToPC->println("Setting up CAN");
  /* Initialize MCP2515 CAN controller at the specified speed */
  if (Canbus.init(CANSPEED_500))
  {
    HardSerial_xBeeToxBee->println("CAN Init Ok");
    delay(500);
  }
  else
  {
    HardSerial_xBeeToxBee->println("Can't init CAN");
    delay(500);
  }
}
```

C Reading Messages form CAN-Bus

```
void loop()
{
  // Other codes are omitted for simplicity
  delay(500);
  scanCANbus();
}

void scanCANbus()
{
  tCAN message;
  if (mcp2515_check_message())
  {
    if (mcp2515_get_message(&message))
    {

      HardSerial_xBeeToxBee->print("ID: ");
      HardSerial_xBeeToxBee->print(message.id, HEX);
      HardSerial_xBeeToxBee->print(", ");
      HardSerial_xBeeToxBee->print("Data: ");
      HardSerial_xBeeToxBee->print(message.header.length, DEC);
      HardSerial_xBeeToxBee->print(", ");

      for (int i = 0;i < message.header.length;i++)
      {
        HardSerial_ArduinoToPC->print(char(message.data[i]));
      }
      HardSerial_xBeeToxBee->println("");
    }
  }
}
```

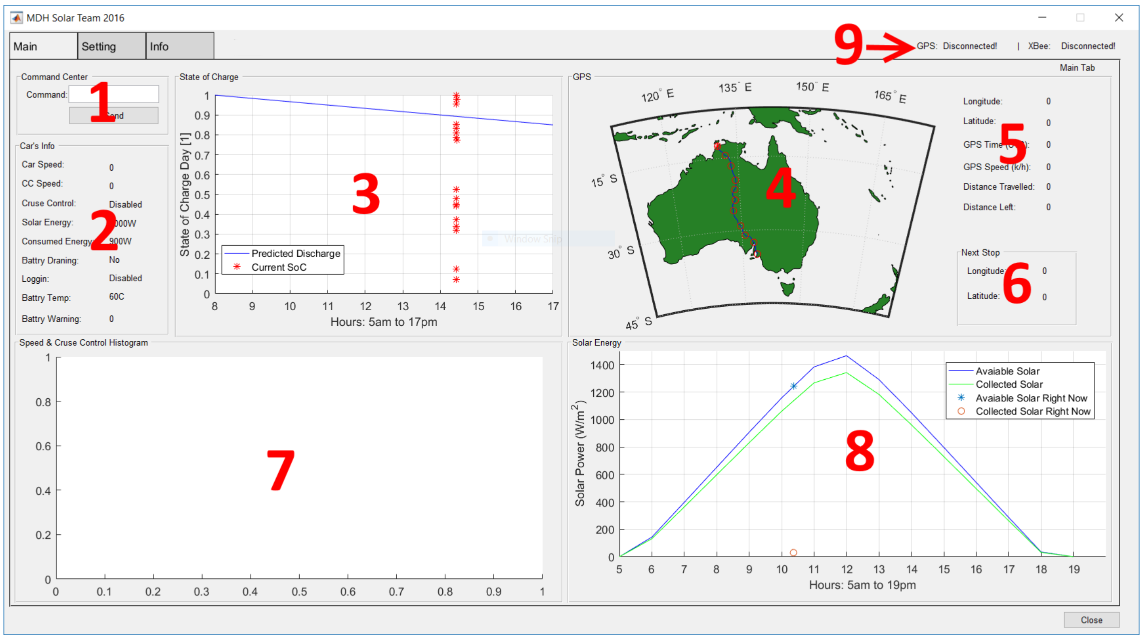
D MATLAB Control Program Manual

A File structure

File Name	Content
Startup.m	As the name suggest this file should be run first, which is responsible for all data initialization and variable declaration
SerialManager.m	This file contains all the functions that are neccessry to communicate with serial object and updating relevant GUI elements
SolarHelpersFunc.m	All the calculation about solar parameters such as azimuth angle zenith angle, sunrise/sunset and so forth are reside here
SolarCaller.m	This file is a mediator between functions in SolarHelpersFunc and the GUI
FileHelper.m	All file handling such as opening/writing happens inside this file
MapHelper.m	This file load the desired map and plot the planning path on it
NMEAlineRead.m	This file include the codes to filter the incoming data from GPS
BatteryHelper.m	Functions inside this file manipulate battery information and update GUI accordingly
SolarProject.m	The code behind file for the SolarProject.fig GUI
SolarProject.fig	The main GUI elements are placed inside this figure file
wsc_data.xlsx	This is an excel file which contains all the coordinates about the race path such as longitude, latitude and so forth

Totally there are 11 files in the project, it is important to follow the instruction and start the Startup file first, afterwards you can configure the port numbers in the setting tab and listening to ports for incoming data.

B Main Tab


MDH Solar Team 2016

Main | Setting | Info

GPS: Disconnected! | XBee: Disconnected!

Main Tab

Command Center

Command:

Send

Car's Info

Car Speed: 0
CC Speed: 0
Cruse Control: Disabled
Solar Energy: 000W
Consumed Energy: 900W
Battry Draning: No
Loggin: Disabled
Battry Temp: 60C
Battry Warning: 0

Longitude: 0
Latitude: 0
GPS Time (U): 0
GPS Speed (k/h): 0
Distance Travelled: 0
Distance Left: 0

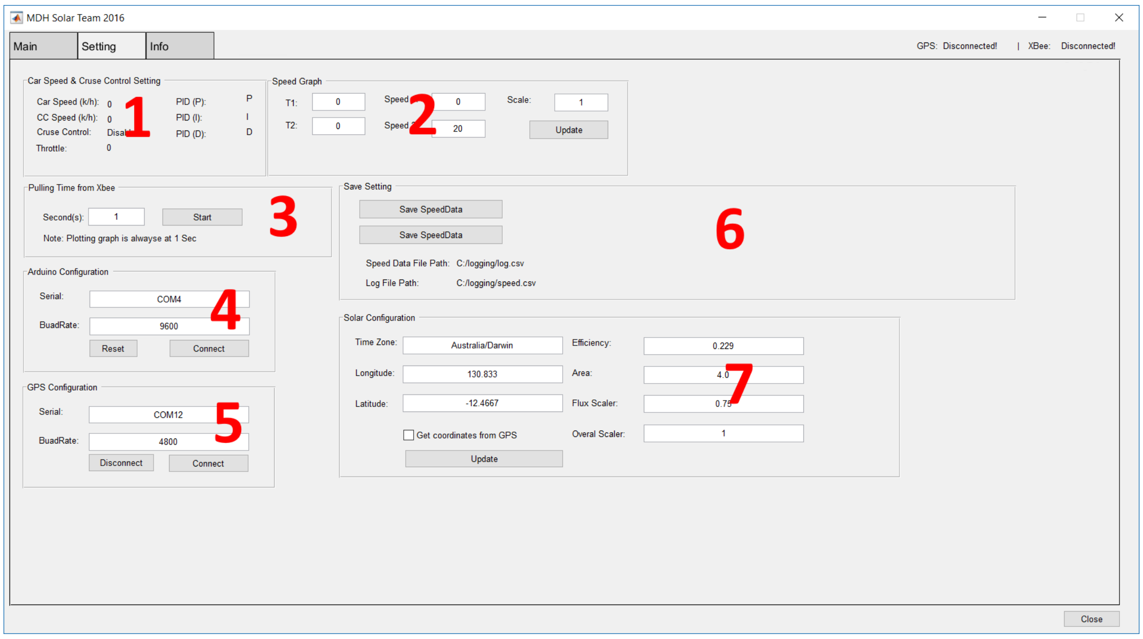
Next Stop
Longitude: 0
Latitude: 0

Close

Figure 27: Main Tab

1. This is the place that you can send command to Arduino, after typing the command press Send button.
2. This section contains car's information such as current speed, cruse control status and so forth.
3. Shows the graph for battery's state of charge and predicted usage.
4. Map of Australia and race rout
5. Current locations GPS info
6. Next stop point coordinates
7. Speed and cruse control graph over time. (scale such as time and resolution can be changed in the setting tab)
8. Predicated solar energy and collected solar power for interested day.
9. It shows if GPS or xBee is connected to the system.

C Setting Tab



GPS: Disconnected! | XBee: Disconnected!

Car Speed & Cruse Control Setting

Car Speed (k/h): 0
CC Speed (k/h): 0
Cruse Control: Disal
Throttle: 0
PID (P): P
PID (I): I
PID (D): D

Speed Graph

T1: 0
T2: 0
Speed: 0
Speed: 20
Scale: 1

Pulling Time from Xbee

Second(s): 1
Note: Plotting graph is alwayse at 1 Sec

Save Setting

Speed Data File Path: C:/logging/log.csv
Log File Path: C:/logging/speed.csv

Arduino Configuration

Serial: COM4
BuadRate: 9600

GPS Configuration

Serial: COM12
BuadRate: 4800

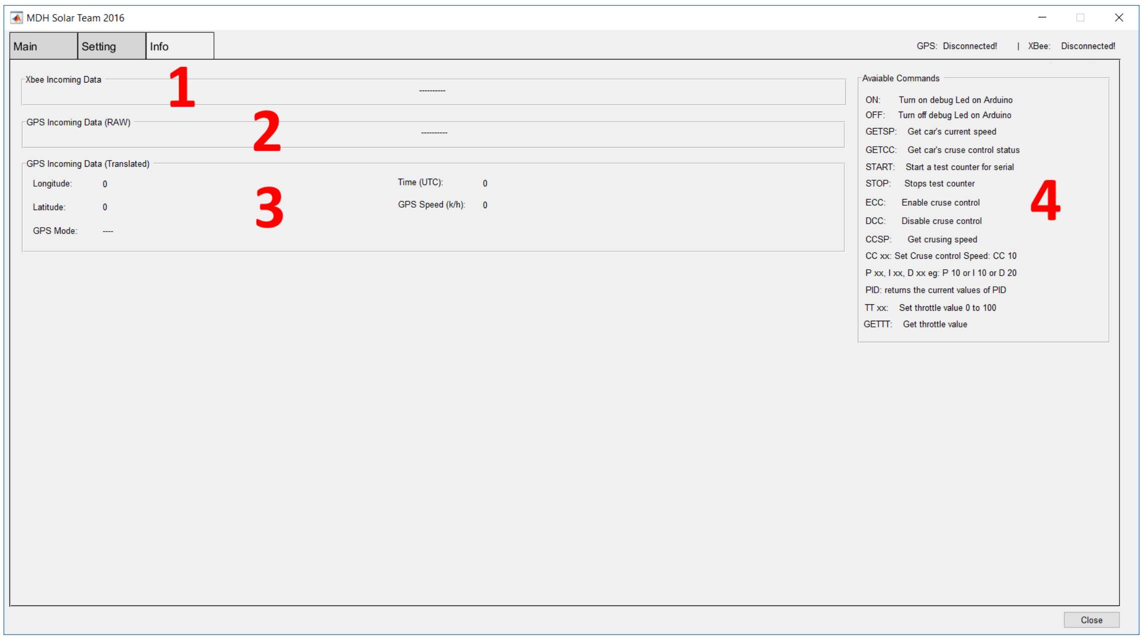
Solar Configuration

Time Zone: Australia/Darwin
Longitude: 130.833
Latitude: -12.4667
☐ Get coordinates from GPS
Efficiency: 0.229
Area: 4.0
Flux Scaler: 0.75
Overal Scaler: 1

Figure 28: Setting Tab

1. Car's information coming from Arduino such as PID values and throttle
2. This is to scale the speed graph, one can change the y axes and set the graph to display speed from x to y with resolution of scale
3. This is the period that sets pulling speed of data from Arduino, the default is 1 second
4. End-user can change the Arduino (xBee) port and speed here
5. Connected GPS port and speed can be changed here, bare in mind that the default speed in the text-box is recommended by manufacturer and if different speed is desired, one need to update the GPS firmware via TTL command to the same baud rate.
6. Paths to save the incoming information
7. This is the solar panel and solar configuration, end-user can update solar panel info here as well as other information, coordinate of the race's start point should be entered here, and if this check box (Get coordinates form GPS) is selected, it will get the coordinates from GPS and all the solar calculation will be updated based on that location

D Info Tab



GPS: Disconnected! | XBee: Disconnected!

Xbee Incoming Data

GPS Incoming Data (RAW)

GPS Incoming Data (Translated)

Longitude: 0
Latitude: 0
GPS Mode: ----
Time (UTC): 0
GPS Speed (k/h): 0

Avaiable Commands

ON: Turn on debug Led on Arduino
OFF: Turn off debug Led on Arduino
GETSP: Get car's current speed
GETCC: Get car's cruse control status
START: Start a test counter for serial
STOP: Stops test counter
ECC: Enable cruse control
DCC: Disable cruse control
CCSP: Get crusing speed
CC xx: Set Cruse control Speed: CC 10
P xx, I xx, D xx eg: P 10 or I 10 or D 20
PID: returns the current values of PID
TT xx: Set throttle value 0 to 100
GETTT: Get throttle value

Figure 29: Info Tab

1. All the incoming (RAW) data from xBee
2. All the incoming (RAW) data from GPS
3. Filtered information such as longitude, latitude, UTC time and so forth from GPS
4. List if available commands that can be sent to the Arduino via MATLAB program

E PCB

A LCD PCB

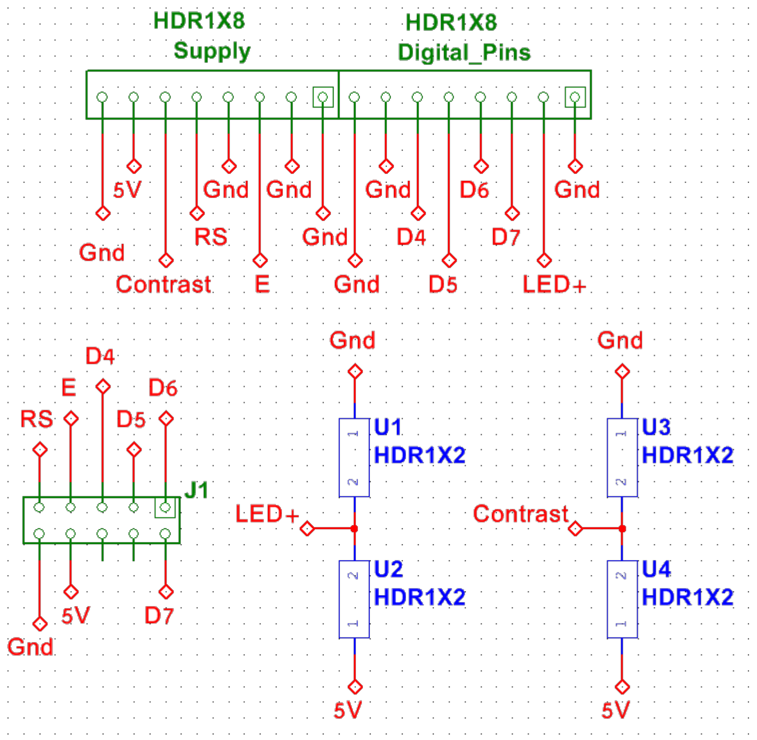


Figure 30: Schematics for the LCD system

Two potentiometers are added to let the user set the contrast and backlight of the LCD. Required connectors and components:

- Male Angled Box Header: 612 010 229 21, Wurth
- Vertical PCB Header with Flanges, 3.5mm: 691 323 100 003, Wurth
- Female Header pins 2.54mm
- Potentiometer 5KOhm

B Adapter PCB

Break sensor, a digital sensor that sense if the driver brakes or not, at 12 volt. Cruise sensor, a digital sensor that sense if cruise control is activated, at 12 volt. Throttle In, an analog signal that retrieves in which extent the driver uses the throttle, an analog signal in the interval 0-5 volt. Throttle Out, a digital PWM signal that controls the speed of the motors, needs to be protected against back current and high voltage. Direct, a digital switch that allows the driver to bypass Arduinos PWM based throttle. Pins that uses a 12 volt logic needs to be transformed to 5 volt, to be compatible with arudino. This is done through a voltage division with a 180Kohm and 120Kohm resistor. The throttle out pin is protected by a 5.1 volt zener diode, which limits the voltage on the pin to 5.1 volt. To protect against back current, there is a kisel diode in outward direction. Cruise sensor has a 3.9Kohm pull down resistor to make sure its default logic level is ground.

Required connectors and components:

- Male Angeled Box Header: 612 010 229 21, Wurth

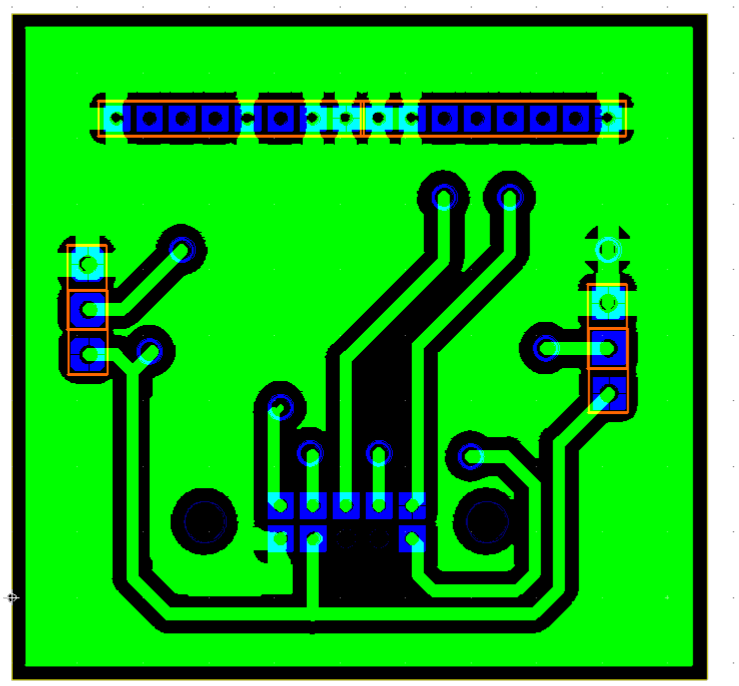


Figure 31: Layout of LCD top layer

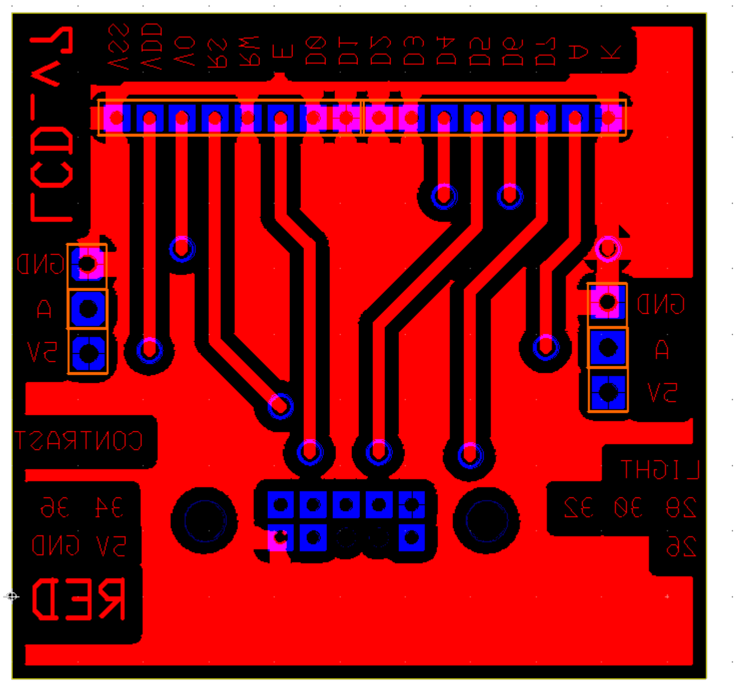


Figure 32: Layout of LCD bottom layer

- Vertical PCB Header with Flanges, 3.50mm: 691 323 100 003, Wurth
- Vertical PCB Header with Flanges, 3.50mm: 691 323 100 002, Wurth
- Kisel diode: 10 526 22, Elfa

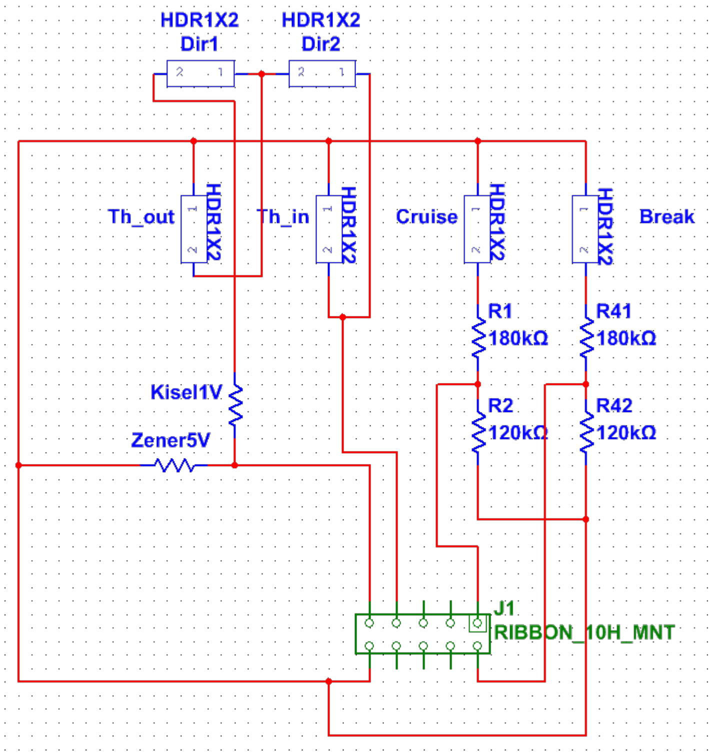


Figure 33: Schematics for the Adapter system

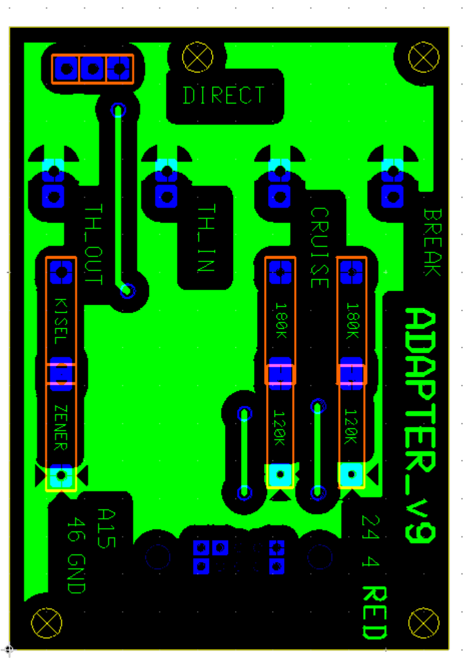


Figure 34: Layout of Adapter top layer

- Zener diode 5.1V: 70 065 88, Elfa
- THT Resistor: 180Kohm
- THT Resistor: 120Kohm

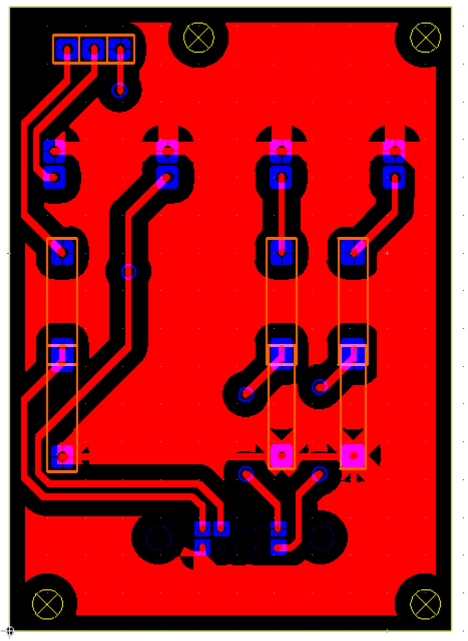


Figure 35: Layout of Adapter bottom layer

C Mega PCB

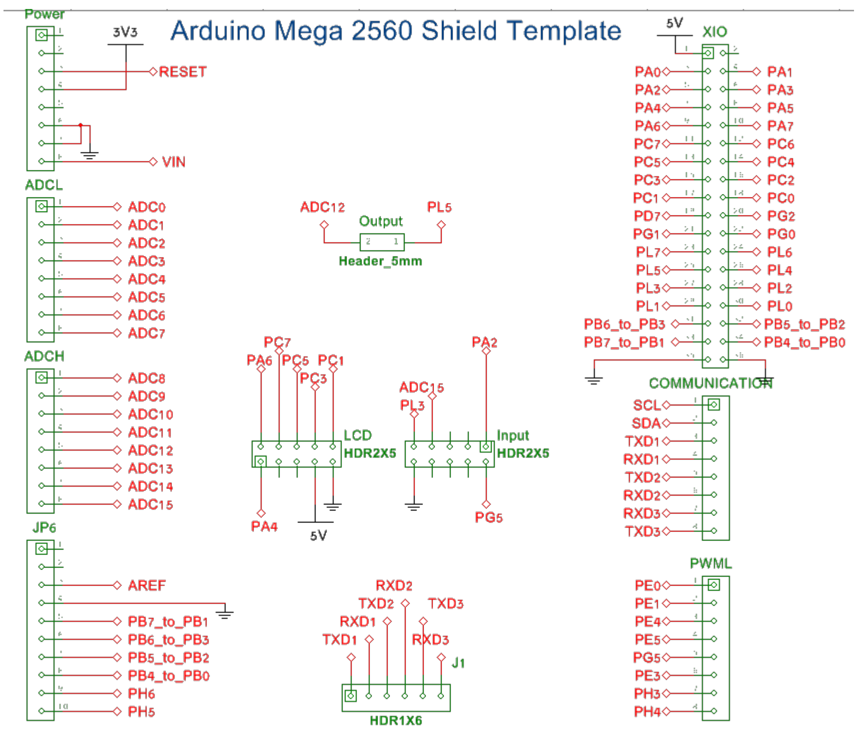


Figure 36: Schematics for the Mega system

Indications where the red side of the ribbon cables are fitted on the top layer. As the schematic shows, the right ribbon connector are intended for LCD ribbon connector as it has 8 connections. The remaining ribbon connector are intended for input ribbon connector. As can be seen in the

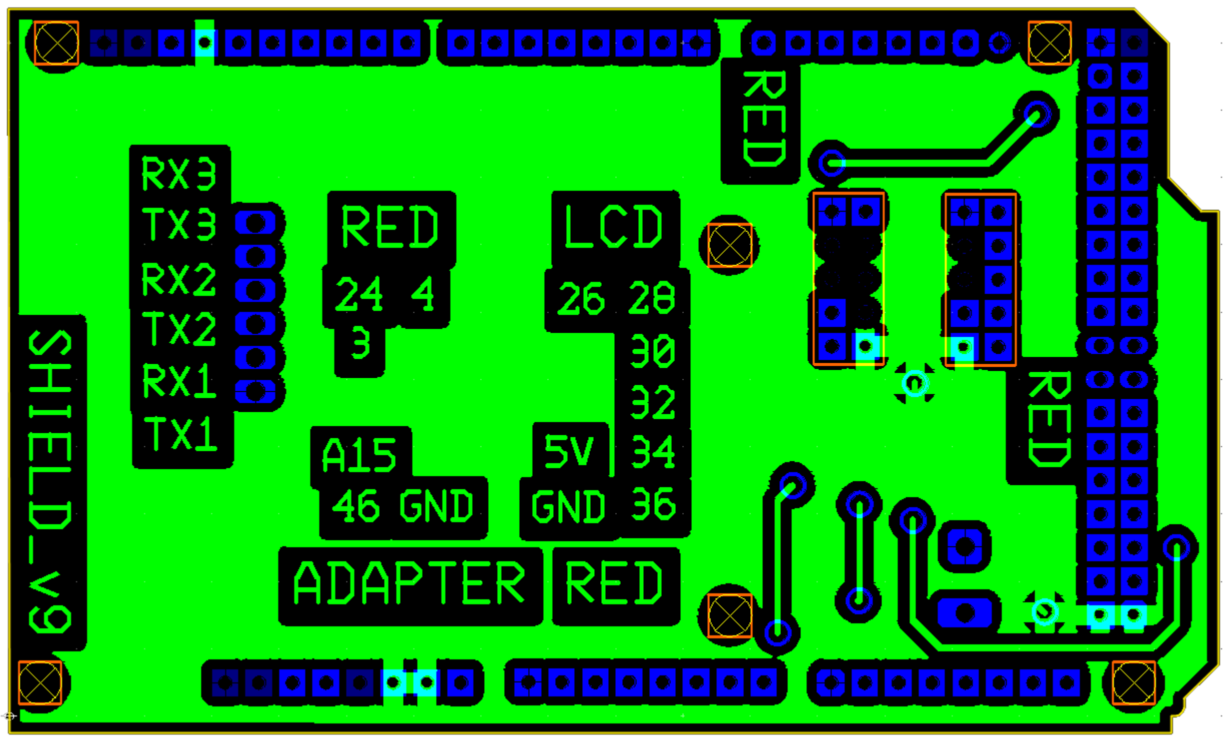


Figure 37: Layout of Mega top layer

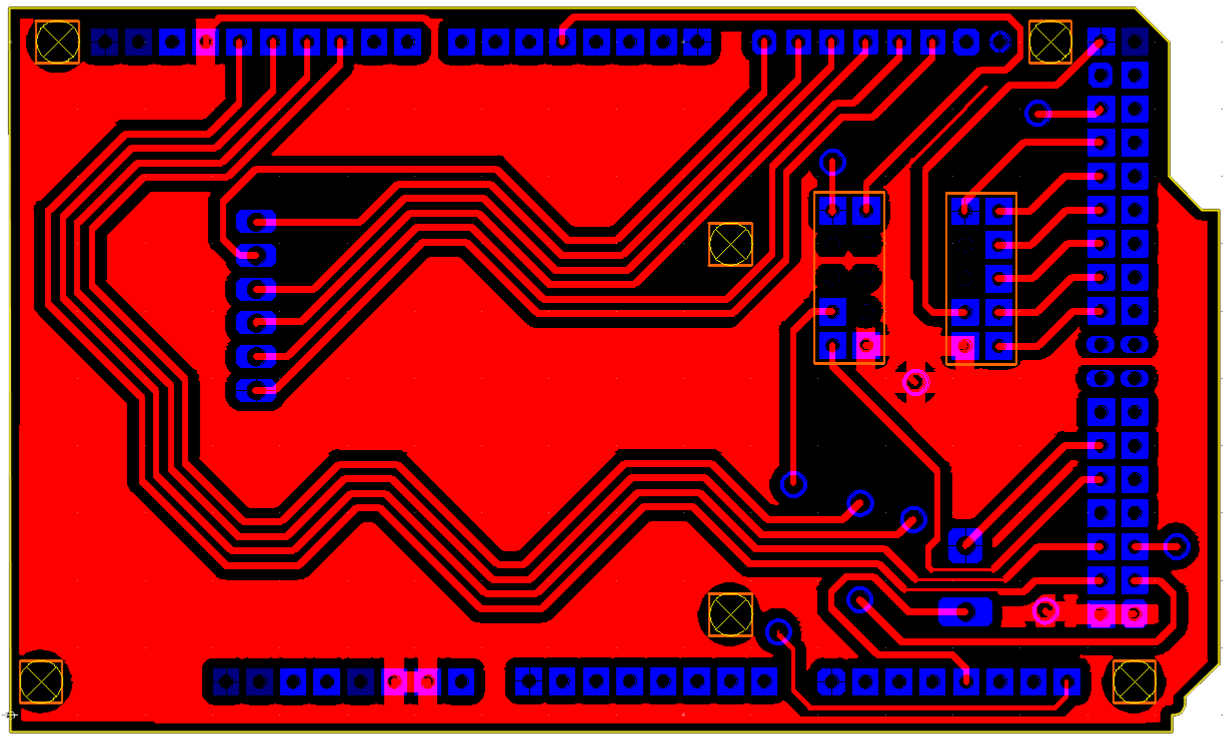


Figure 38: Layout of Mega bottom layer

schematics pins 10,11,12,13 is connected to 53,51,50,52 respectively. It is done because the CAN shield uses soft serial through these pins.

Required connectors:

- Horizontal Entry with Rising Edge Clamp 2.54mm: 691 210 910 006, Wurth
- 45degrees Entry with Rising Cage Clamp 5mm: 691 211 720 002
- Male Box Header: 612 010 228 21, Wurth
- Female Header pins for Arduino Mega Shield

D DCDC PCB

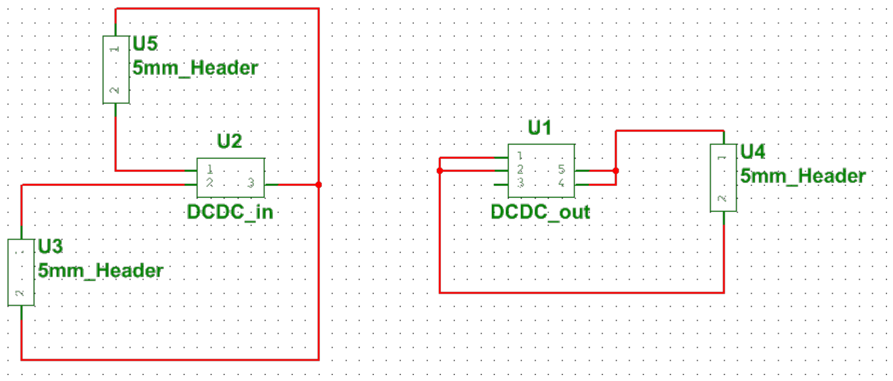


Figure 39: Schematics for the DCDC system

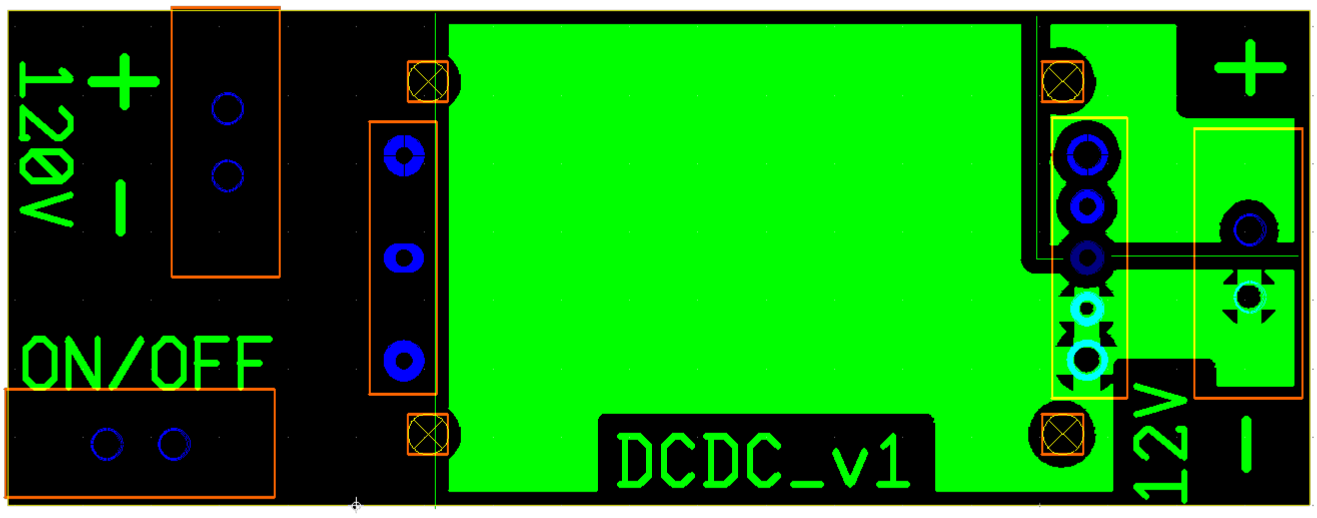


Figure 40: Layout of DCDC top layer

DCDC converter, RCQ75110S12 are supposed to be place on top of the PCB and soldered so it is firmly placed on the PCB. There is only one possible way to place it.

Required connectors:

- Closed Vertical PCB Header with Flanges 5mm: 691 318 700 002, Wurth

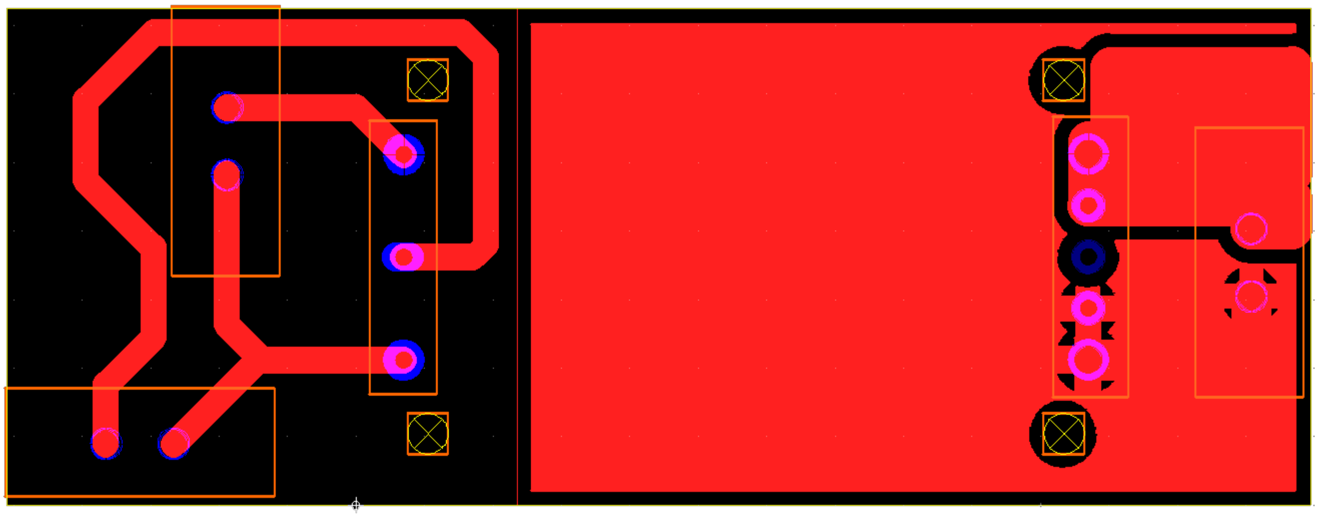


Figure 41: Layout of DCDC bottom layer

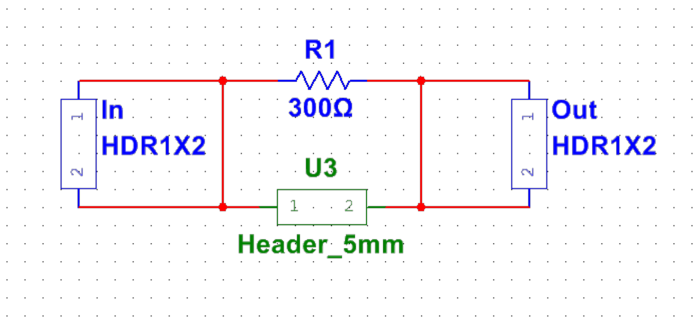


Figure 42: Schematics for the Delay system

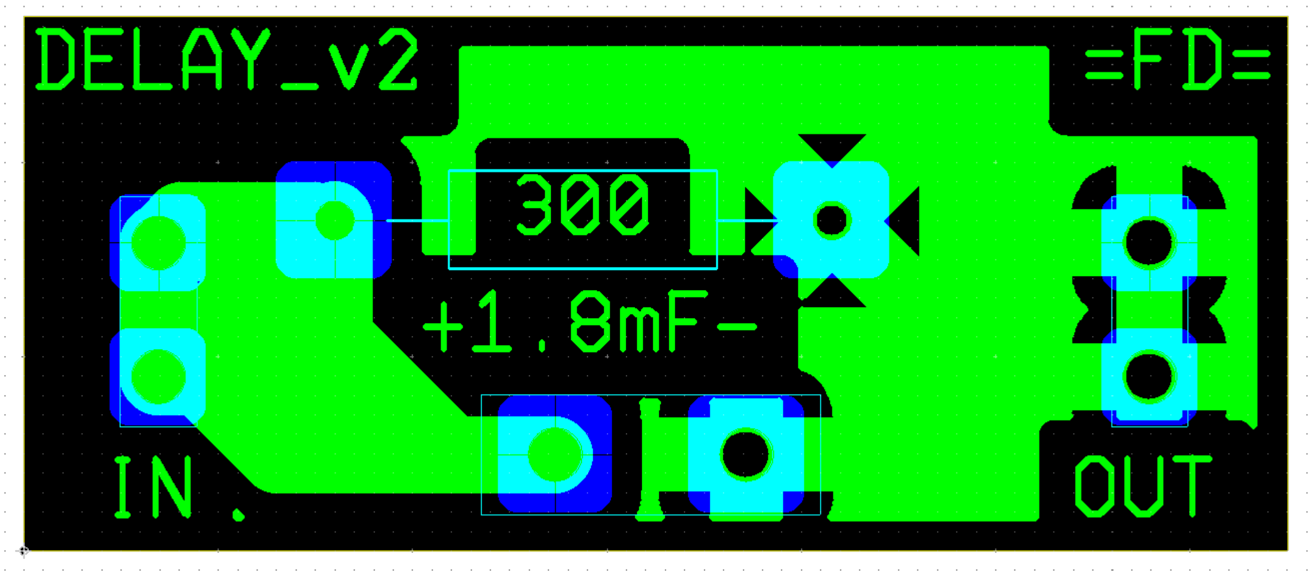


Figure 43: Layout of Delay top layer

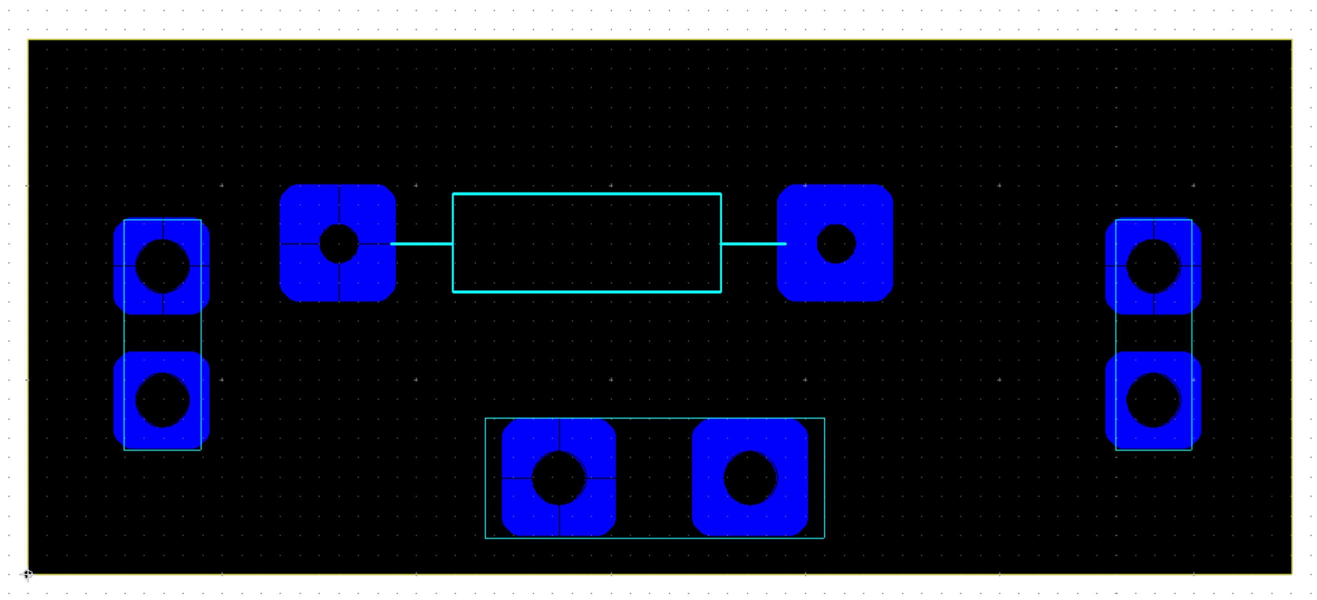


Figure 44: Layout of Delay bottom layer

E Delay PCB

Notice the polarity of the capacitor, it is noted on the PCB.
Required connectors:

- Vertical PCB Header with Flanges, 3.5mm: 691 323 100 002, Wurth
- THT Resistor 300Ohm
- THT Capacitor 1.8mF

F Micro PCB

Arduino Micro is supplied from the Vin pin, which supports supply voltage from 5-20V. Be warned not to exceed the limit.
Required connectors:

- Horizontal Entry with Rising Edge Clamp 2.54mm: 691 210 910 006, Wurth
- Horizontal Entry with Rising Edge Clamp 2.54mm: 691 210 910 003, Wurth
- Horizontal Entry with Rising Edge Clamp 2.54mm: 691 210 910 002, Wurth
- Male Header pins 2.54mm

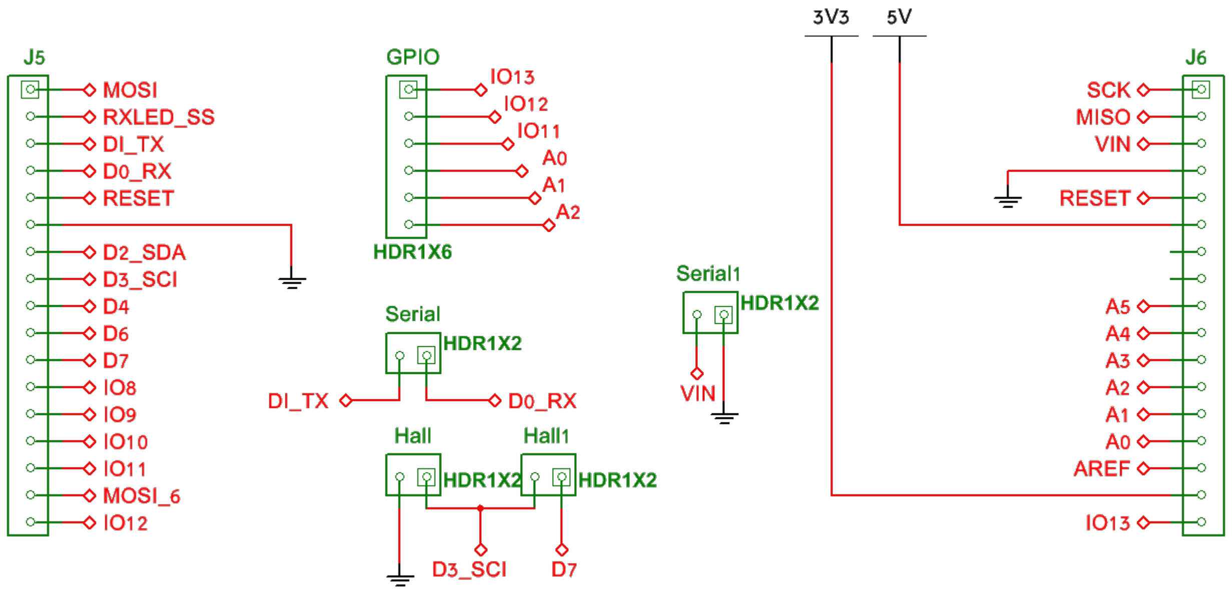


Figure 45: Schematics for the Micro system

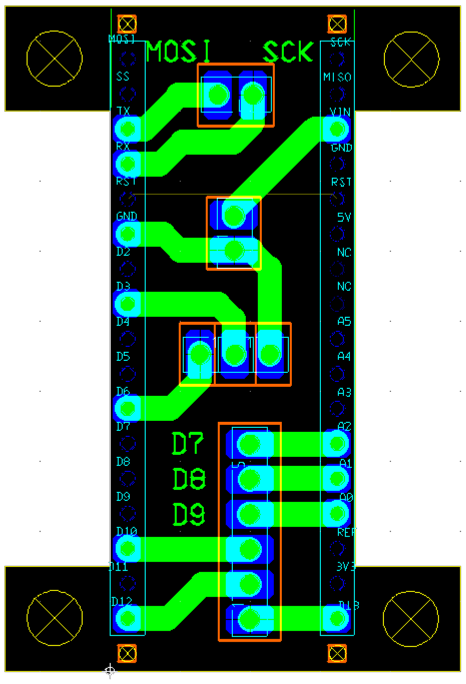


Figure 46: Layout of Micro top layer

G Relay PCB

The connectors found on this card are; (1) 5mm Closed Vertical PCB with Flanges, 2 pin: 691 318 700 002, (2) 5mm Closed Vertical PCB with Flanges, 3 pin: 691 318 700 003 which are both bought from Wurth. The following list describes the ports/pins found on the Relay PCB and their respective purpose. See figure 48 and 49 for layout.

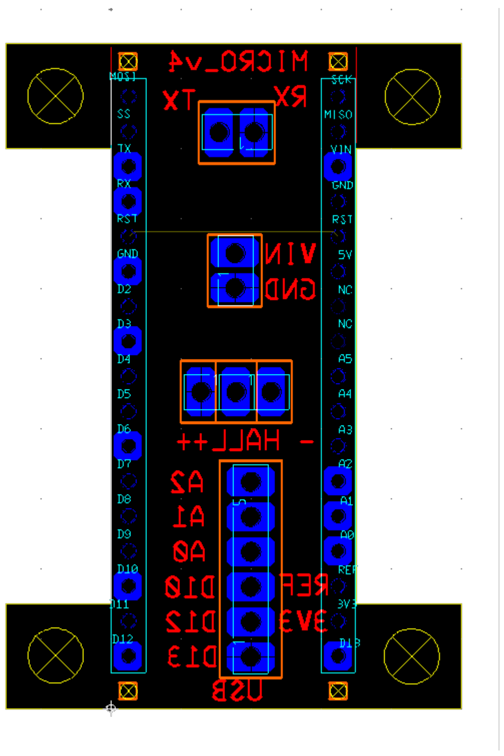


Figure 47: Layout of Micro bottom layer

- Supply connector (+ IN -) – This 2 pin connector should be connected to the DC/DC converter so that the PCB is supplied with 12V and GND.
- 12V and GND – Two 3 pin connectors are connected in parallel to the supply connector so that 12V and GND can easily be obtained for unaccounted systems which may be created in the future.
- Arduino – A 2 pin connector which delivers 12V and GND can be used to supply the voltage divider circuit for the Arduino Mega which was previously mentioned.
- Horn – A 2 pin connector is found on the PCB which is for the input/output of the horn button on the dashboard. Another 2 pin connector can also be found for supplying the horn when the button is pressed.
- Brake – A 2 pin connector is found for the brake handle as well as a 2 pin connector for supply the brake lights. The brake lights connector is connected with wires to the other front PCB, which will be further explained for that PCB.
- Turn indicator switches – A 3 pin connector is found on the PCB which should be connected to the input of the turn indicator and the hazard light button found on the dashboard.
- Horn Relay – Two 2 pin connectors is found on the PCB which should be connected to the horn relay. One of these connectors supply the relay with 12V and GND while the other connector is connected to the toggle input and the output of the relay.
- Brake Relay – Two 2 pin connectors is found on the PCB which should be connected to the brake relay. One of these connectors supply the relay with 12V and GND while the other connector is connected to the toggle input and the output of the relay.
- Flash Relay – A 3 pin connector is found on the PCB which should be connected to the flash relay. The pins on the far ends of the connector are 12V and GND while the middle pin is for the output of the relay.

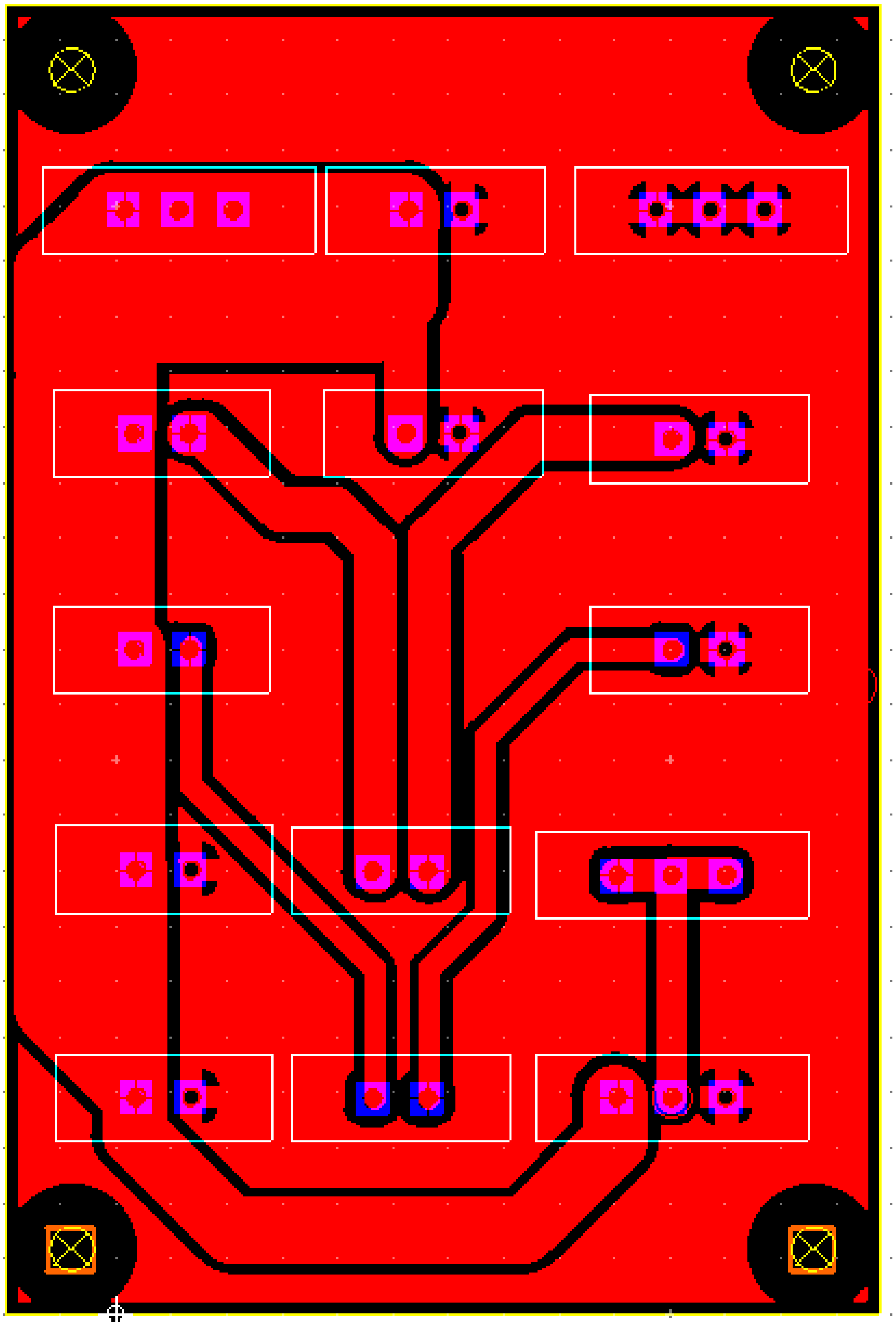


Figure 48: The figure depict the bottom layer of the Relay PCB.

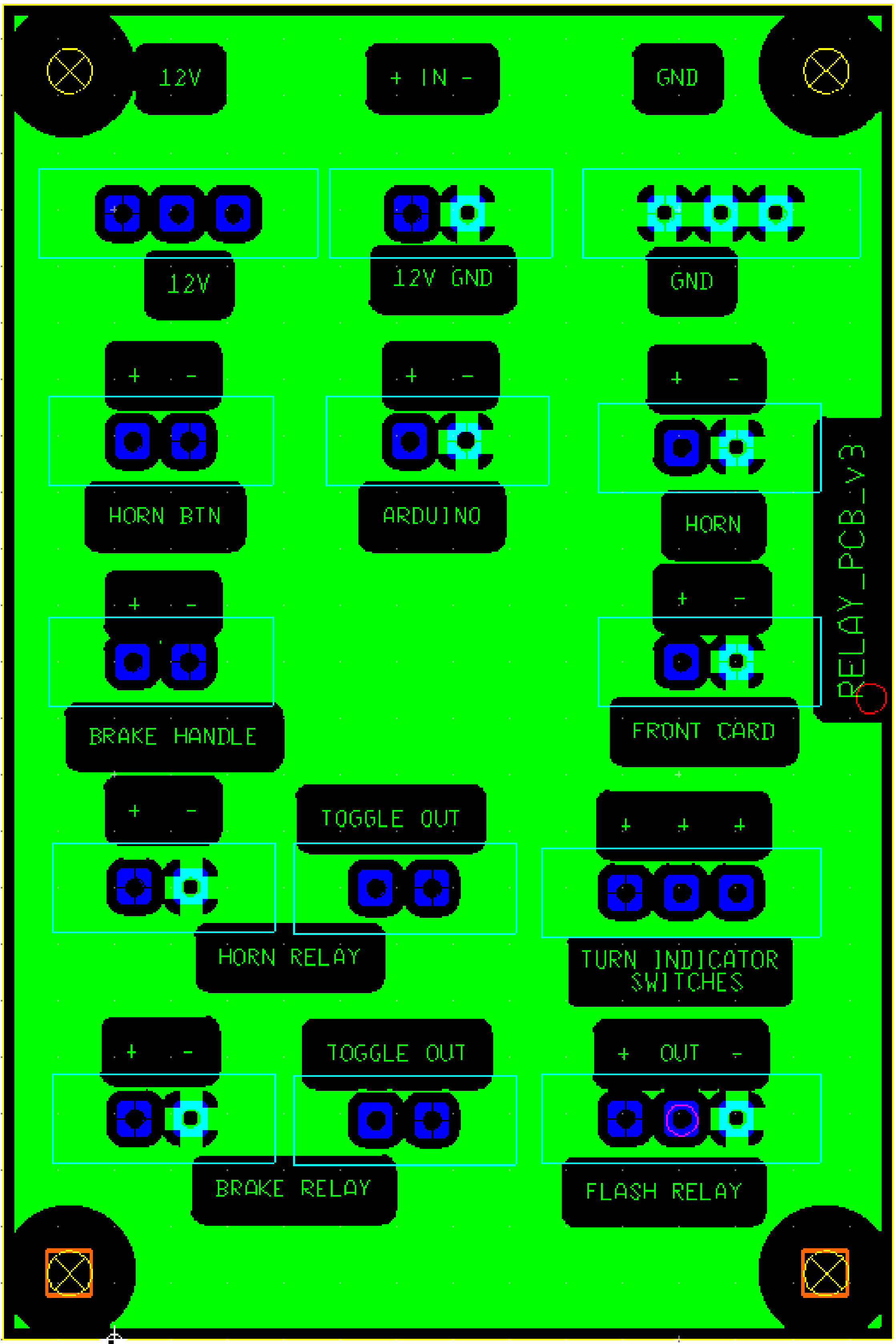


Figure 49: The figure depict the top layer of the Relay PCB.

H Front PCB

The connectors found on this card are; (1) 5mm Closed Vertical PCB with Flanges, 2 pin: 691 318 700 002, (2) 5mm Closed Vertical PCB with Flanges, 3 pin: 691 318 700 003, and (3) 10p Ribbon connector: 612 010 229 21 which are all bought from Wurt. The following list describes the ports/pins found on the Front PCB and their respective purpose. See figure 50 and 51 for layout.

- Relay Card – This is a 2 pin connector which is connected to the brake light connector on the Relay card. Through this connector the Front card obtains GND and supply voltage for the brake lights.
- Hazard button – This is a 2 pin connector which should be connected to the outputs from the hazard light button found on the dashboard.
- Turn Switches – This is a 2 pin connector where the turn switch outputs should be connected. One pin is for when the switch is clicked for left turn and one pin is for right turn.
- Left- and Right Diodes – There are six 2 pin connectors which are for supplying the left and right turn indicators in front, on the side, and on the dashboard (not for the rear diodes).
- Ribbon connector – The ribbon connector is connected to the Rear PCB. This connector is used to supply the turn indicators as well as the brake lights in the rear of the car. This was only the ribbon cable has to go from the front to the rear of the car.

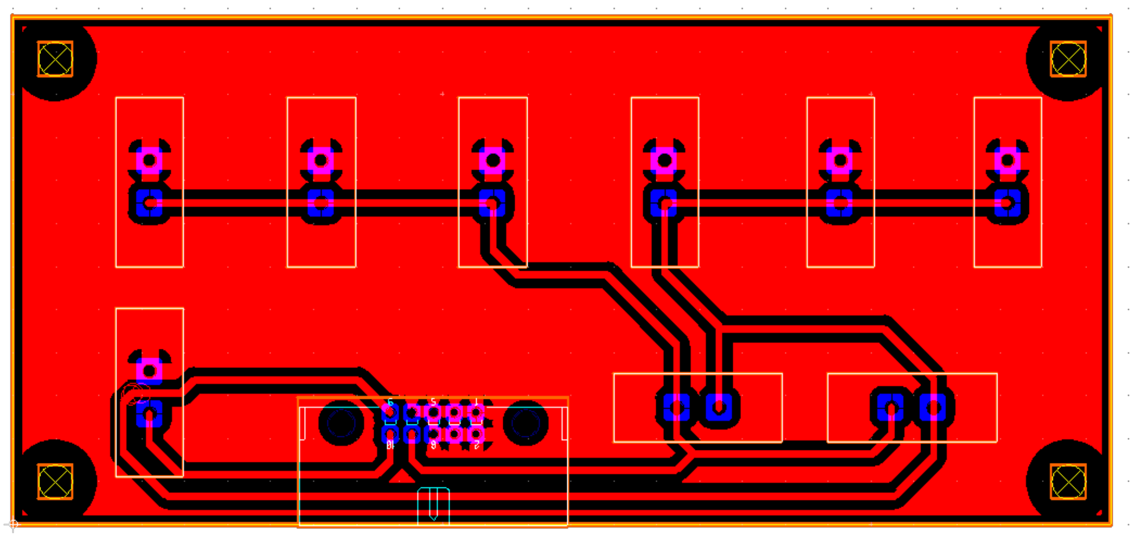


Figure 50: The figure depict the bottom layer of the Front PCB.

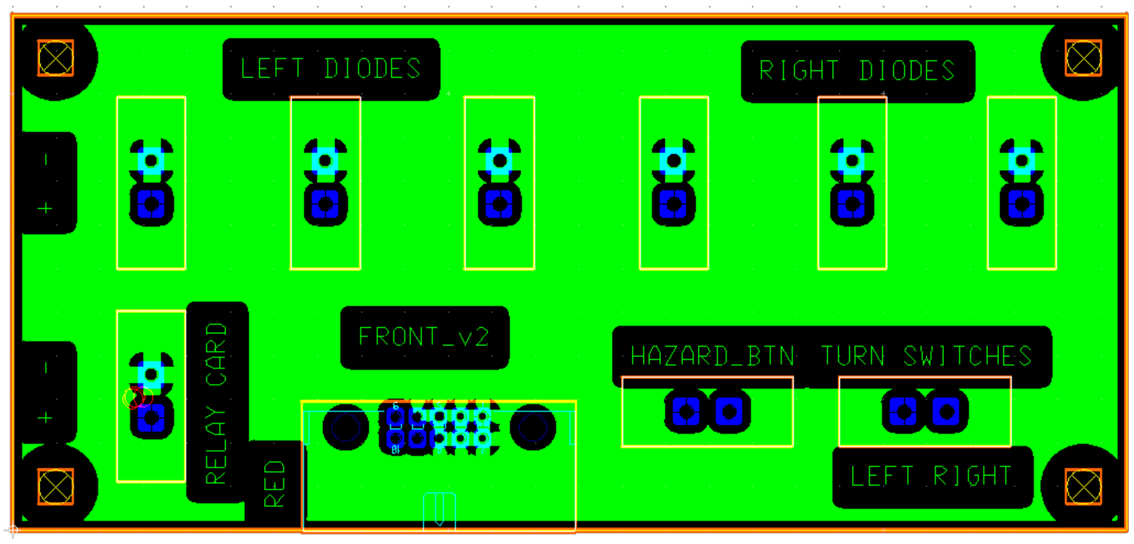


Figure 51: The figure depict the top layer of the Front PCB.

I Rear PCB

The connectors found on this card are; (1) 5mm Closed Vertical PCB with Flanges, 2 pin: 691 318 700 002, (2) 5mm Closed Vertical PCB with Flanges, 3 pin: 691 318 700 003, and (3) 10p Ribbon connector: 612 010 229 21 which are all bought from Wurt. The following list describes the ports/pins found on the Rear PCB and their respective purpose. See figure 52 and 53 for layout.

- Ribbon connector – The ribbon connector should be connected to the front PCB and is used to obtain the supply for left- and right turn indicators as well as the supply for the brake lights.
- Left – There is one 3 pin connector which supply the left turn indicator and brake light.
- Right – There is one 3 pin connector which supply the right turn indicator and brake light.
- Middle – There is one 2 pin connector which supply the middle brake light.

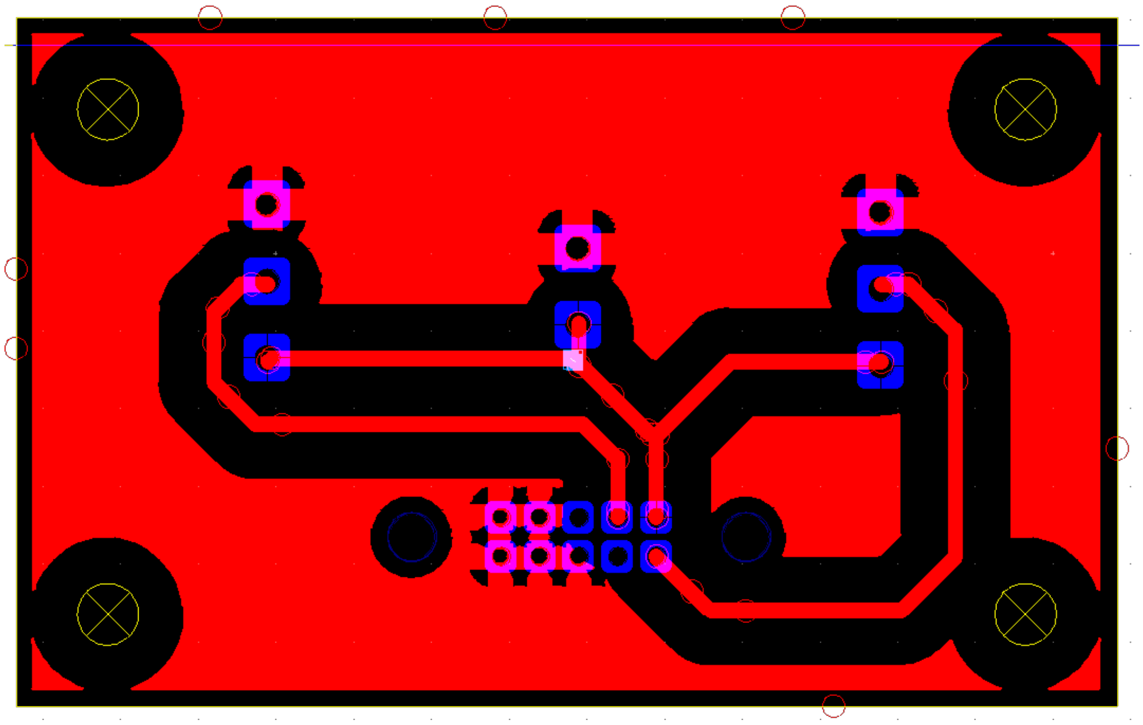


Figure 52: The figure depict the bottom layer of the Rear PCB.

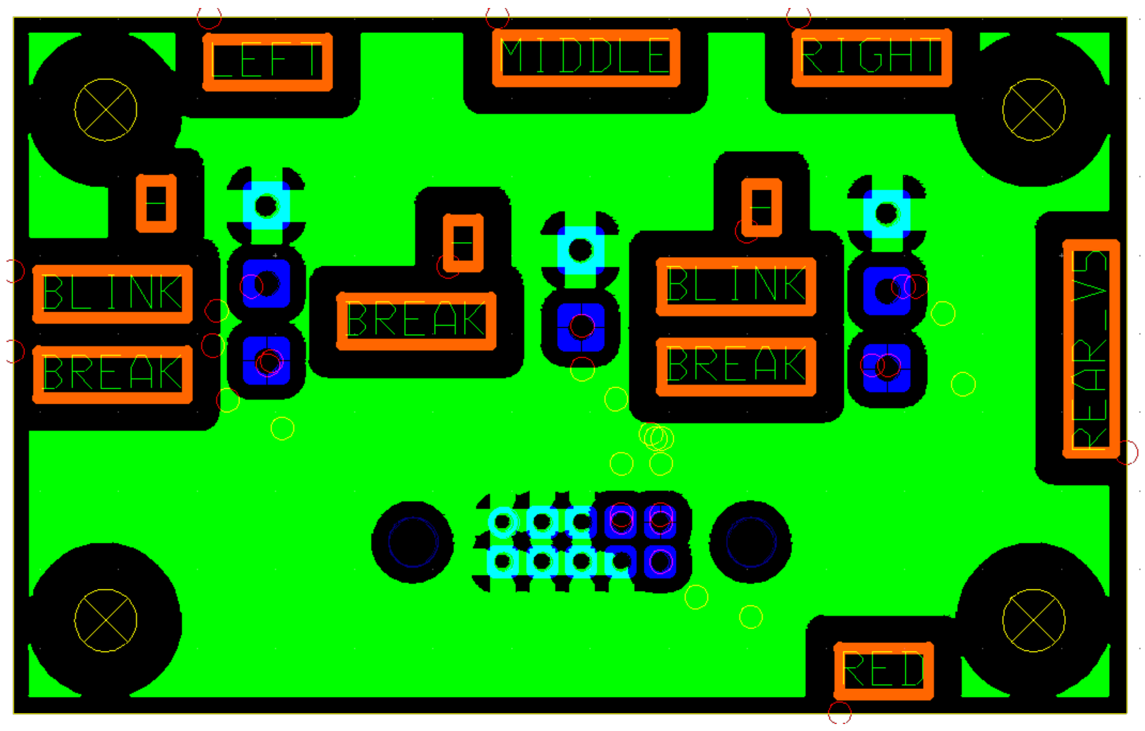


Figure 53: The figure depict the top layer of the Rear PCB.