UNIVERSITÀ
DI PARMA

Department of Engineering and Architecture,

Degree Course in Computer Engineering, Electronics and

of Telecommunications

Design and development of a steering wheel for a car Formula Student competition

Design and development of a steering wheel for a Formula Student racing car

Speaker:
Dear Prof.
Charles Concari

Graduating:
Michael Bandini

Academic Year 2021-2022

Summary

1 Introduction

1.1 UniPR Racing Team

The UniPR Racing Team is a university project, supported by the University of of Parma, founded in 2007 by a group of mechanical engineering students. Objective The primary aim of the project is to design, develop and manufacture in full autonomy a racing single-seater that can participate in the competition Formula Student International.

Equipped with an internal combustion engine, the first cars built by UniPR Racing Team participate in the *Combustion category.* The excellent results and a greater awareness of the work led the Team to definitively abandon engine to take the important decision to enter the world of Electric category . First car equipped with an electric motor is the PSR01, developed in 2018 and then replaced by the PSR02, single-seater which brought one of the best results in the history of UniPr Racing Team: 5th place in the overall ranking at the Italian stage of FSAE Italy 2021 in Varano de' Melegari. In 2022 a development and optimization path is undertaken culminating in the current single-seater: the PSR02-S. Third electric car of the UniPR Racing Team and direct evolution of the 2021 prototype, the PSR02-S participates in two races foreign: Formula Student Netherlands, one of the most prestigious and challenging races for competitiveness; Formula Student Czech Republic where the Team gets the first top 10 foreign.

Figure 1 PSR02-S

1.2 The Student Formula

Formula Student is a student design championship involving engineering departments of universities around the world organized by SAE

International (Society of Automotive Engineers). All Teams are required to comply to a specific regulation, issued every year by the organizing body, and which intends to leave free interpretation to the designers, binding them, however, with severe restrictions to ensure the safety of participants.

The events are divided into three macro-groups: Technical Inspections, necessary to be able to access to Dynamic Tests, and Static Tests. Each macro-group contains internally several tests that allow the single-seater to be evaluated both in terms of design and competitiveness through scores established by regulation:

	CV & EV	DC
Static Events:		
Business Plan Presentation	75 points	-
Cost and Manufacturing	100 points	-
Engineering Design	150 points	150 points
Dynamic Events:		
Skid Pad	50 points	-
DV Skid Pad	75 points	75 points
Acceleration	50 points	-
DV Acceleration	75 points	75 points
Autocross	100 points	-
DV Autocross	-	100 points
Endurance	250 points	-
Efficiency	75 points	-
Trackdrive	-	200 points
Overall	1000 points	600 points

Figure 2 Score available in a Formula Student race

Fundamental to the competition is knowing what vehicle can be used for just one year, starting from the first day on site of his first race. As mentioned from the regulation, rule A2.2.2: *“To be classified as new, a vehicle must have at least a newly manufactured frame with significant changes in the primary structure compared to its predecessor”.*

1.2.1 Technical inspections

Technical inspections are divided into the following parts:

• [EV ONLY] **Accumulator Inspection,** the entire design is checked, functionality and safety of the battery pack including the charger which will be inspected and sealed, and the basic set of tools used.

• [EV ONLY] **Electrical Inspection,** the insulation resistance between the mass of the TS (Tractive System) and the LVS (Low Voltage System) and, in order for the test to be passed, the measured insulation resistance must be at least 500 ÿ/V in relation to the maximum TS voltage of the vehicle. In addition, the IMD (Insulation Monitoring Device) which must turn off the TS within 30s with a fault resistor 50% lower than the response value.

• **Mechanical Inspection,** where the design, development and commissioning are verified powertrain and chassis safety following all the technical requirements requested by the section T 1-13 of the FSG regulation.

• **Tilt Test,** in which the vehicle, with the pilot inside, is positioned on a tilting table with an angle of 60°. To overcome it there must be no loss of liquid and all wheels must remain in contact with the surface of the deck.

• **Vehicle Weighing,** in ready-to-race condition and with all fluids at maximum level of filling.

• [EV ONLY] **Rain Test** in ready-to-race conditions and with TS active. The water is sprayed onto the vehicle from any possible direction. The test is passed if the IMD does not It activates while water is sprayed on the vehicle for 120 seconds after it water spray has stopped.

• Brake Test whereby all four wheels are locked at the end of a straight line of acceleration. In 11.1.2 [EV ONLY] *“After acceleration, the traction system must be turned off by the pilot and the pilot must brake using only the mechanical brakes”.*

1.2.2 Dynamic tests

Having successfully passed all the technical inspections, the single-seater faces four dynamic tests in which competitiveness, reliability and efficiency are evaluated:

• **Skidpad:** Circuit composed of two concentric circles arranged in a figure eight, like shown in figure 4, and in which two laps are timed in succession complete.

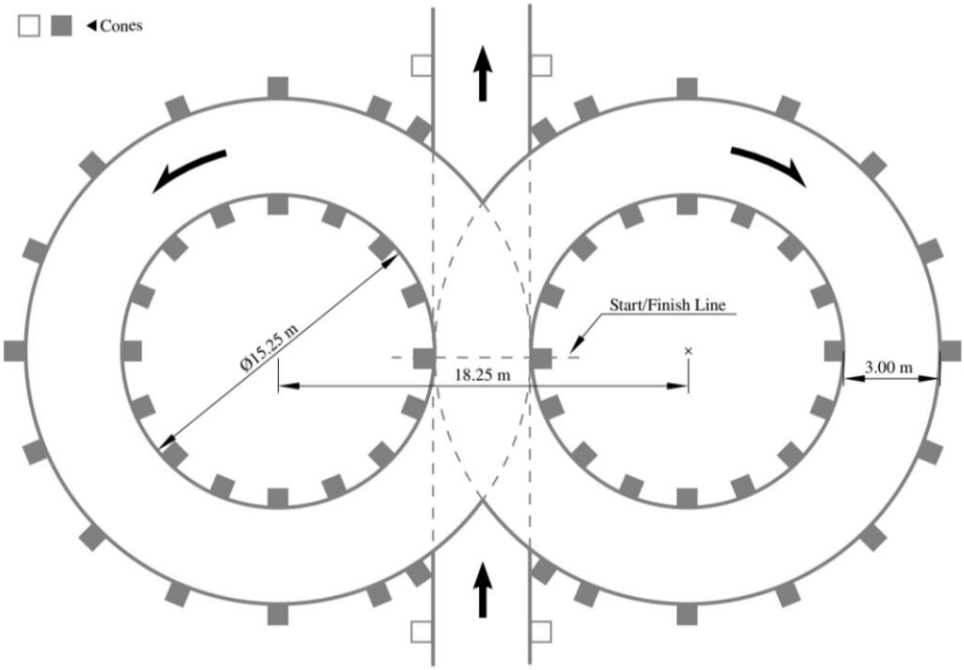

Figure 4 Skidpad Track

$$M\_SKIDPAD\_SCORE = 46.5\left(\frac{\left(\frac{T_{max}}{T_{team}}\right)^2 - 1}{0.5625}\right)$$

Figure 3 Skidpad Score

Tteam = best time achieved by your team including penalties.

Tmax = best overall time multiplied by 1.5 including penalties.

• **Acceleration:** acceleration test on a straight line of 75m and at least 3m wide.

$$M\_ACCELERATION\_SCORE = \left(\frac{\frac{T_{max}}{T_{team}} - 1}{0.5}\right)$$

Figure 5 Acceleration Score

Tteam = best time achieved by your team including penalties.

Tmax = best overall time multiplied by 1.5 including penalties.

• **Autocross:** a timed lap of a track. Track that must respect the following parameters (D6.1.1)

- Straights: no longer than 80 m
- Curves: up to 50 m diameter
- Elbow bends: Minimum external diameter of 9 m (of the bend)
- Slalom: cones in a straight line at a distance between 7.5 and 12 m.
- Miscellaneous: Chicanes, multiple curves, decreasing radius curves, etc. minimum width of the minimum width of the track is 3 m.

$$AUTOCROSS\_SCORE = 95.5\left(\frac{\frac{T_{max}}{T_{team}} - 1}{0.25}\right)$$

Figure 6 Autocross Score

Tteam = best time achieved by your team including penalties.

Tmax = best overall time multiplied by 1.5 including penalties.

• **Endurance & Efficiency:** same circuit as the Autocross test on come 22km completed, halfway through which the drivers change takes place with maximum time 3min. To evaluate the Efficiency are taken into account for the assignment of a score only vehicles that meet the following requirements:

- the vehicle received points for the endurance race.
- the uncorrected endurance time does not exceed 1.333 times the incorrect endurance of the fastest vehicle.

$$ENDURANCE\_SCORE = 300\left(\frac{\frac{T_{max}}{T_{team}} - 1}{0.333}\right)$$

Figure 7 Endurance Score

[EV ONLY] The endurance energy is calculated as the time integrated value of the measured voltage multiplied by the measured current logged by the data logger, see EV 4.6. Regenerated energy is multiplied by 0.9 and subtracted from the used energy.

Efficiency points based on the following formula are given:

$$EFFICIENCY\_SCORE = 75\left(2 - \frac{EF_{team}}{EF_{min}}\right)$$

with

EF_{team} the team's efficiency factor

EF_{min} the lowest efficiency factor of all teams which were considered for efficiency

Figure 8 Efficiency Score

1.2.3 Static tests

During the static tests three different parts are evaluated:

- **Engineering Design,** that is, all the projects are presented to a group of judges the engineering choices applied to the single-seater, always respecting of the objective and the rules of the competition.

Category	CV & EV Points
Overall Vehicle Concept	35
Vehicle Performance	20
Mechanical / Structural Engineering	10
Tractive System / Powertrain	20
LV-Electrics / Electronics / Hardware	15
Driver Interface	10
Autonomous Functionality	30
Engineering Design Report (EDR)	10

Figure 9 Engineering Design Score

- **Business Plan Presentation:**
  - S 1.1.1 *"The objective of the BPP is to assess the team's ability to develop and deliver a comprehensive business model that demonstrates that the their product - a prototype racing car - could become a lucrative business opportunity that creates a profit monetary".*
  - S 1.1.2 *"Judges must be treated as if they were potential investors or partners in the presented business model".*

- S 1.1.3 *“The business plan must refer to the specific prototype”.*

Category	Points
Pitch Video	10
Novelty	10
Content	20
Finances	10
Deep Dive Topic	10
Demonstration and Structure	15
Delivery	10
Questions	10
General Impression	5
Total	100

$$BPP\_SCORE = 70\left(\frac{Pteam}{Pmax}\right)$$

Figure 10 Business Plan Score

Pteam = score obtained by your team.

Pmax = highest score awarded to any team that does not participate in the finals.

- **Cost and Manufacturing,** whose objective is to assess the understanding by of the production process team and the costs associated with building a prototype racing car. This includes trade-off decisions between content and cost, purchase or sale decisions and understanding the differences between prototype and mass production.

Category	Points
Format and Accuracy of Documents	5
Knowledge of Documents and Vehicle	5
BOM and BOM discussion	35
Discussion Part 2 “Cost Understanding”	55
Total	100

Figure 11 Cost and Manufacturing score

1.2 State of the art on steering wheels for cars competition

Taking as a reference the *open wheel* championship par excellence, that is F1, Let's see how steering wheels have changed over the years, adapting over time to what were the requests from the pilots.

1.2.1 1950s and 1960s

In the 1950s they were basically inspired by simple road cars.
dimensions far from small, some reached over 40cm in diameter, the wheel
the internal one was made of aluminum while the external one was made of wood, which was problematic because
caused splinters to appear in the hands of the drivers while driving. In fact, the cars
of the 50's did not have power steering, so the large diameter was necessary for
the pilot to get the traction needed to do
drive the car in the correct direction.

Figure 12 McLaren M7C Steering Wheel

In the 60s the car began to have a
different look: the engine is moved
behind the pilot by modifying the trim of the
body and making the front part very
slimmer. The driver's position has also been
further reduced inside the car
consequently reducing the space for the steering wheel.

1.2.2 1970s

In the 70s the manufacturers began to
think of the steering wheel as a way to
get performance as well as a way to
turn the front wheels. In the second
mid-decade we see the introduction
of the emergency switch, useful in case
where the throttle valve had jammed in
position open. This new
mechanism allowed the driver to
quickly turn off the engine before it came into contact with the barriers.

Figure 13 McLaren M23 steering wheel

1.2.3 1980s

Steering wheel design took another step forward in the 1980s. First change is the use of suede around the rim. Stefan Johansson, driver for the McLaren and Ferrari, called it the biggest improvement made during the his stay in the sport. The suede, more adherent, allowed him to feel the steering wheel much better, leading drivers to request custom-made rims for their driving style.

Figure 14 MP4/4 Steering Wheel

In the second half of the decade the steering wheel begins to take on more responsibilities. A release mechanism is added quick which allowed the pilot to remove the flying quickly in an emergency. Another novelty is a button that can manage the radio communications between pilots and the pit wall. Some stables even had a boost button that enriched the fuel mixture to aid overtaking. However, the evolution of far major was the introduction of the paddle shifter. Designer Bernard had the idea of eliminating the clutch pedal and introducing two paddles on the back of the steering wheel to allow the driver to change gear. This saves space, weight and the need to take your hands off the wheel to change gears, as well as to be much faster once the technology's teething problems are overcome. After initial reliability problems, at the beginning of the 90s all the teams switched to paddle shift. The humble steering wheel had finally had its first major evolution, but the widespread introduction of the paddle shifter was only the beginning of the steering wheel revolution. In the nineties electronics took over the F1 and this has dramatically accelerated the development of technology.

1.2.4 90s / 00s

After 40 years of metal, the steering wheel has finally been renewed and, like the rest of the car, was made of carbon fiber. In this way the weight was immediately reduced. The shape and size of the steering wheel have also undergone a change: from the traditional round specifications we have moved to a flattening general and the removal of the upper and lower sections of the steering wheel.

Even though steering wheels have become smaller, their functionality has increased. The buttons from the 80s have remained, but have been joined by other functions, such as the pit lane speed limiter and reverse gear. These buttons are They also added rotary dials. In the 1990s pilots could select different engine and traction control settings, allowing them to change key parameters while driving.

The end of the century brings a revolution in terms of materials rather than technology: suede has been replaced by rubber, shaped to the shape of the the pilot's hands to provide optimal feel and feedback. The number of buttons and knobs increases, the pilot is now able to make quick changes to perfect the balance and performance of the car. During a Grand Prix could adjust settings such as the differential for the input and the center part of the curve. Engine braking and torque settings, to name a few, allow the driver to find the strong point of his own set-up.

Figure 16 McLaren MP4-14

Figure 15 McLaren MP4-23

1.2.5 1910s

Like the 1990s and the 2000s, the following decade was also marked by
of evolution. The biggest change occurs with the introduction of engines
turbo-hybrids. In 2014, confirming the complexity of the era, the steering wheel appears
a large LCD display that allows the pilot to browse multiple data menus for
make decisions about engine modes or energy recovery functions.

Of fundamental importance is the fact that each pilot decides the configuration of the
flying as it sees fit. This includes not only the layout and type of
buttons, switches, and wheels, but also the organization of the information that is
shown on the on-board display. On the one hand, this approach makes it easier to use by
of the pilots, it allows to save weight, as the number of physical switches was
reduced.

1.2.6 The display

Nowadays many teams use the display integrated into the steering wheel, but there are always some
variations. Williams, for example, kept the LCD display fixed to the frame which,
in addition to being cheaper, it is lighter and easier to handle in terms of
of rotational mass of the steering wheel. The downside of this choice is that the display is
completely obscured by the steering wheel itself when cornering, and this limits
drastically the opportunities that the pilot has to take advantage of the very important
information displayed on the small screen.

1.2.7 The buttons

After this brief historical overview, we come to understand what all the purposes are
buttons on the steering wheel. It is clearly unlikely that you will ever know how to use them
every single button present, just for the fact that the stables tend to hide
pieces of information, but we can analyze those visible from the images released.
To do this, we will take the steering wheel of the 2020 Williams FW43 as an example.

First, let's divide the buttons on an F1 steering wheel into

- Real **buttons** (activate/deactivate a function),

- Adjustment **wheels** (usually have numbers from 1 to 10 in a progressive)
- **Manettini** (normally located in the center of the steering wheel, usually refer to to the most important settings of the single-seater, those of the Power Unit)

Figure 17 Williams F1 Steering Wheel

Once this is clear, let's analyze figure 17 from left to right:

ENTRY: (wheel): Changes the differential when entering corners;

N/R: Selects Neutral or Reverse gear;

Minus: Decreases the value of the setting shown on the screen

OK: Confirms receipt of the communication to the pits;

BB-R: Increases brake balance towards the rear;

PC: Informs the pits to make a pit stop on the next lap

TRQ (wheel): Changes the torque value of the Power Unit;

Multifunction Knob (Green): Various chassis and electronic settings;

MODE knob: Power Unit delivery mode;

GO Knob: Throttle and electronics modes;

TYRE knob: type of tyre fitted and consumption stage;

Multifunction Knob (blue): various chassis and electronic settings;

HPP: Opens the Mercedes Power Unit Menu;

BB F: Increases brake balance towards the front;

RAD: opens communication with the pit wall;

Plus: Increases the value of the setting shown on the screen;

LIM: activates the pit lane limiter;

B-Mig (wheel): Changes how the brake balance changes from front to rear
rear during braking;

EXIT (wheel): Changes the differential when exiting corners.

The image also shows one of the many screens that the on-board display can show.
have. In this case:

PIT: indicates the insertion of the Pit Limiter

DAT: indicates that data download mode is activated

0: Indicates the number of engine revolutions (off, in this case)

N13 Tyre Prime: the type of tyre currently fitted

BBal: setting the brake balance from front to rear

BMig: Current balance migration setting

TRQ: Current torque setting

Diff Ent: Current setting of the input differential

Diff Mid: Current setting of the differential at the top of the curve

Diff Exit: Current setting of the output differential

1.2.8 The PSR02-s steering wheel

The structure and positioning of the steering wheel within the cockpit must follow the indications provided by the *FS Rules Book Germany* published annually in view of the competitions. For this thesis, the season regulations will be taken as a model
2022.

As regards the positioning of the steering wheel, it is necessary to consider the

rules T2.7.6 and T2.7.8 whereby:

T2.7.6 The steering wheel must be no more than 250 mm rearward of the front hoop. This distance is measured horizontally, on the vehicle centerline, from the rear surface of the front hoop to the forward most surface of the steering wheel with the steering in any position.

Figure 20 Rule T2.6 Formula Student regulation

T2.7.8 In any angular position, the top of the steering wheel must be no higher than the top-most surface of the front hoop.

Figure 18 Rule T2.7.8 Formula Student regulation

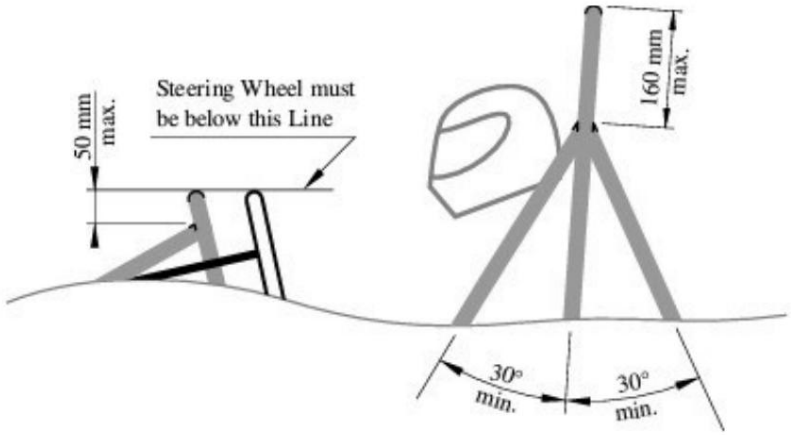

Figure 19 Positioning the steering wheel

This is essential as the pilot must have adequate visibility of the

front and side of the vehicle. Sitting in the normal driving position must

must have a minimum field of vision of 100° on both sides (T4.10.01).

Furthermore, it should be remembered that the permitted play of the steering system is limited to a

total of 7° measured on the steering wheel. Placed laterally to the steering wheel, without having

hindered, and mounted directly to the frame inside the cockpit, we find the

shutdown button with the following characteristics:

T 11.4.4 One shutdown button serves as a cockpit-mounted shutdown button and must

- have a minimum diameter of 24 mm
- be located in easy reach of a belted-in driver
- be alongside of the steering wheel and unobstructed by the steering wheel or any other part of the vehicle

T 11.4.5 The international electrical symbol consisting of a red spark on a white-edged blue triangle must be affixed in close proximity to each shutdown button.

T 11.4.6 Shutdown buttons must be rigidly mounted to the vehicle and must not be removed during maintenance.

Figure 21 Shutdown button adjustment inside the cockpit.

Once we have established the correct positioning of the steering wheel inside the cockpit, let's go to analyze the structure of the steering wheel while inputs and screen will be analyzed in pilot interface chapter.

We will consider three necessary components for this brief analysis: materials, dimensions and mechanical components.

As per rule T2.7.7

T 2.7.7 The steering wheel must have a continuous perimeter that is near circular or near oval. The outer perimeter profile may have some straight sections, but no concave sections.

Figure 22 Rule T2.7.7 Formula Student Regulations

The base of the steering wheel is rigid but at the same time extremely light. thanks to the carbon fibre sandwich structure whose skins are separated between they from a honeycomb, a type of alveolar core made of aramid fiber (Nomex). The handle is applied on both sides to this supporting structure. made using ABS material with 3D printing. Assembled we will have a length total of 232mm for a height of 157mm. Always in ABS material and made via 3D printing, there are two switches for adjusting the driving set-up as from figure Figure 23.

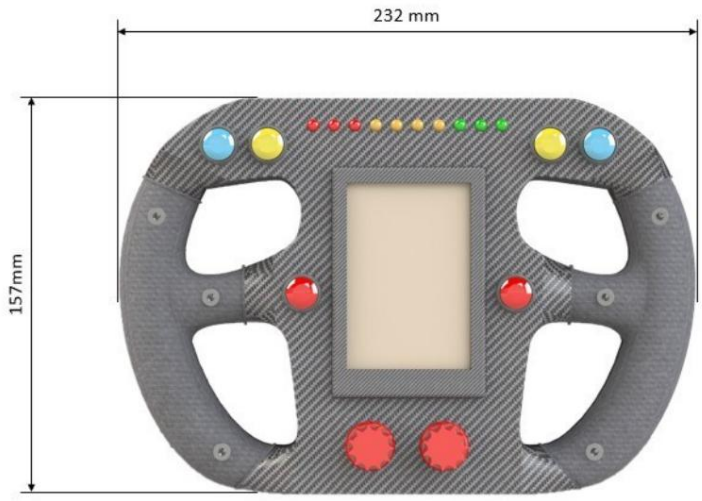

Figure 23 PSR02-S steering wheel dimensions

Moving on to the rear, it must be considered that, as per the regulations, the steering wheel must

be connected to the steering column via a fixed quick release system

to the structure using three M4 screws.

T2.7.5 The steering wheel must be attached to the column with a quick disconnect. The driver must be able to operate the quick disconnect while in the normal driving position with gloves on.

Figure 25 rule T2.7.5 Formula Student regulation

T2.7.10 Joints between all components attaching the steering wheel to the steering rack must be mechanical and visible at technical inspection. Bonded joints without a mechanical backup are not permitted. The mechanical backup must be designed to solely uphold the functionality of the steering system.

Figure 24 rule T2.7.10 Formula Student regulation

The quick release is designed for safety reasons when the rider is
should find himself in a dangerous situation, and therefore need to get out as soon as possible
as quickly as possible from the car. In Formula Student competitions, this

Figure 26 Quick release system

process is verified through an egress test whereby the pilot, dressed in full
the necessary equipment and in the driving position, must necessarily be
able to exit the cockpit in a maximum of 5s as reported in chapter T4 of the
regulation.

T4.11 Driver Egress

T4.11.1 All drivers must be able to exit to the side of the vehicle in less than 5 s with the driver in the fully seated position, hands in the driving position on the connected steering wheel (in all possible steering positions) and wearing the required driver equipment as in T13.3. The egress time will stop when the driver has both feet on the ground.

Figure 27 rule T4.11.1, Formula Student regulation

3 Requirements Analysis

A Formula Student car must be highly adaptable in order to
to tackle the various dynamic tests, the asphalt surfaces and the various driving styles of the
pilots who use it. One of the requirements of the steering wheel is to be able to adjust
through buttons and knobs of the parameters in the control unit that allow you to adapt
the car according to your needs. Among these, the possibility of changing the power
maximum that can be supplied by the battery pack, in order to be able to manually manage consumption in
in case of anomalies in the de-rating system or the power control system. The couple
maximum and the maximum regenerative torque controlled by the inverters. The modification
of the invasiveness of traction control and the effectiveness of torque vectoring so as to
help improve the grip and stability of the car during cornering and
accelerations. A steering wheel that meets these functional requirements can provide the driver
optimal car control and a greater chance of success on the track.

The steering wheel not only allows the driver to give inputs to the car, but it is also
used to display its output. It is essential that the information displayed
are strictly necessary for the pilot and easily readable, so the instrument panel
display must be clearly visible even in sunlight. To facilitate reading, the
characters used should be as large as possible and the use of colours can make
the fastest interpretation of information. Considering that the information to be
display can vary significantly depending on the dynamic test in progress,
it becomes crucial to have several specific pages available for each of them. In
This way, you are guaranteed to always have only the information at hand
strictly necessary, displayed in a clear and easily readable manner. Furthermore, it is
It is necessary to provide some pages dedicated to reporting any errors
of the vehicle and the display of data from specific sensors, in order to make it easier
quick troubleshooting process in case of problems during testing or inspections
techniques. In this way, the driver and the team can act promptly and with
precision to identify and resolve any anomalies, thus ensuring maximum
efficiency and safety of the vehicle.

As for the more technical requirements, it is essential that the steering wheel communicates in efficiently with the rest of the system, using the CAN bus which represents the reference technology for all other components of the car. It is essential that the system is able to provide information in real time with a frequency of at least 30Hz, considering that the display represents a bit of a "bottleneck" in this process. In fact, its maximum refresh rate is precisely 30Hz.

Another important requirement is the use of custom hardware for optimize the space and weight of the steering wheel system, which plays an important role fundamental in the field of motor racing. In addition, the system the steering wheel must be equipped with a microcontroller of the STM32F4 series, already used for the VCU and the BMS. This choice was motivated by the need to ensure easy maintenance of the system by team members, who they may be called upon to work on different projects over the years. Using the same microcontroller allows to simplify considerably learning maintenance techniques, facilitating the transition from one project to another and ensuring maximum efficiency of the work carried out. In this way, it ensures excellent resource management and greater speed in the intervention in case of need.

4 Pilot interface

Figure 28 PSR02-S steering wheel

The steering wheel features 6 buttons, 10 LEDs, two rotary switches and a touch screen screen. In this chapter we will analyze each component individually and all the functionality they enable.

4.1 Buttons

4.1.1 Power regulation

Analyzing the image above, starting from the pair of buttons at the top right, we can notice that they are labeled with the letter "P". These buttons are used to regulate the maximum power delivered by the battery pack, which according to the regulation it cannot exceed 80 kWh. This functionality is very useful during the acceleration test, where each driver has two attempts at arrangement. Depending on the strategy adopted, you can choose to set the power at maximum during the first attempt, and in case the threshold is exceeded allowed, the pilot can slightly reduce the power via the buttons. The power maximum output is visible on the screen.

4.1.2 Engine torque adjustment

As for the pair of buttons at the top left, they are used for
regulate the torque transmitted to the motors. This value is limited to a maximum of 90 Nm,
as specified in the motors datasheet. These buttons are often used
in combination with traction control, in order to obtain the correct sensitivity
on the accelerator. For example, if the asphalt is more slippery than expected and the wheels
tend to slip during the acceleration phase, the traction control must
reduce the engine torque in advance, so as to prevent the wheels from slipping
further. By slightly decreasing the torque, you can accelerate with more
decision without losing time and stability, and in case of skidding, the control of
traction will intervene to correct the problem without having to reduce
drastically the couple.

4.1.3 Regenerative braking regulation

The central pair of buttons instead is used to regulate the regenerative torque, this
It drastically affects braking feel, fuel consumption and temperatures.
During an Endurance test we want this to be very high in order to be able to
recover as much energy as possible during braking, but this involves
the increase in cell temperatures which cannot exceed 60°C. Therefore
if the maximum threshold is approached, it is advisable to reduce the torque
regenerative through the buttons. Another scenario could be that the couple
too high regenerative creates instability during braking causing the wheels to lock
rear, in this case the driver can adjust it as he likes to find the right feeling.

4.1.4 Ready to drive

Figure 29 Ready To Drive Button

As you can see in figure 29 the left button
used for regeneration has a second label
called RTD. In fact, during the procedure of
when turned on the button takes on the role of *Ready To Drive Button.* In this way it was possible to avoid
the addition of an extra button.

4.2 Rotary switches

For both levers, only 8 of the 10 available positions are used,
This choice was made because more than 8 would have been useless and doing so
we gave the pilot the possibility to have a margin of error, this is because the
position 1 corresponds to turning off the associated control, while 10
should have been the maximum invasiveness of the control. Since a
Formula Student circuit is very fast paced and steering wheel modifications need to be
done in a very short time, the risk would have been that the pilot could pass by
one extreme to the other unconsciously. In this way we have two levels of zone
dead which assume the value 8 if the knob was being rotated clockwise,
instead they assume the value 1 if you were rotating counterclockwise, this does not take away
the ability to switch directly from position 1 to 8.

4.2.1 Traction control

The selector on the right allows you to adjust the traction control: with position 1, the
control is disabled; as you increase the value, the control becomes more
invasive, cutting the couple more and in a more timely manner. This
adjustment is very useful since the car is used by drivers with different
background and driving styles. Thanks to the traction control adjustment, the car
can be adapted to the behavior of the pilots, who can maintain their
way to press the accelerator.

4.2.2 Torque Vectoring

The left lever, instead, allows you to adjust the control known as "Torque
vectoring", often referred to as "electronic differential". This control at the
guide allows to supply a differentiated torque to the wheels during the cornering phase,
making steering easier. Again, position 1 represents control
off, while the more you increase the value, the more aggressive the control becomes. Each pilot
has its own driving style and the correct adjustment of the torque vectoring allows you to
increase or decrease understeer.

4.3 LED

Starting from the left we have 3 red LEDs, 4 yellow LEDs and 3 green LEDs. These indicate the status of battery charge. The LEDs can only light up sequentially from left to right, this means we cannot have a led on without that all those to the left of it are lit. When all the LEDs are lit contemporary means that the battery has a charge between 91% and 100%. While if there is only one LED lit it means that the battery is between 1% and 10%.

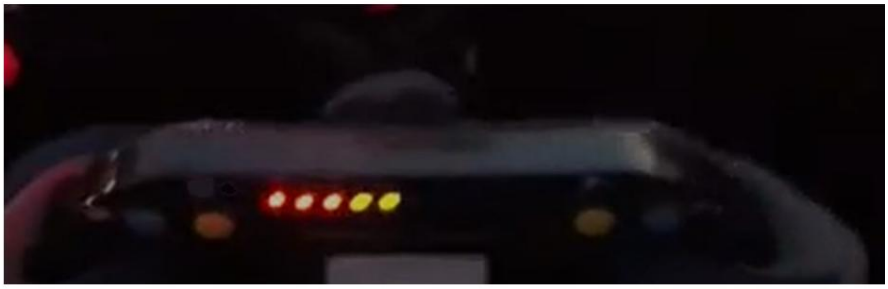

Figure 30 LED lighting

In Figure 30 we can see five LEDs lit up this means that the battery is between 40% and 50%.

4.4 Display

4.4.1 General page

Figure 31 General page

The main page, shown in figure 31, is mainly used for the AutoX test, where the The driver has two laps to try to finish the route in the shortest possible time. Since the test requires maximum concentration on driving, there is no time to look at the display. For this reason, the information displayed on it is useful in the process of preparation for the lap, during which the driver checks that all driving settings are correct and that there are no anomalies, such as low battery or too high temperatures. Furthermore,

Once the driver has completed the timed lap, he can read the time
used on the display and, after completing the second lap, can also read the
time difference from the previous lap. The time display is
possible thanks to an algorithm that uses the GPS system installed on the car, which
however, it will not be explored in depth in this thesis.

4.4.2 Endurance Page

The Endurance page, shown in figure 32, as can be seen
as you can guess from the name, it is used during the test
Endurance. The most important data reported in the center of the
page is the battery charge, as this race of
battery life requires careful battery management and
of temperatures. To help the pilot in managing the
battery, has been reported at the top left, labeled with
“Residual”, an estimate of the percentage value of the battery
remaining at the end of the Endurance, maintaining the same
pace of the last two laps. The driver's goal is to find a race pace that
where that value is as close as possible to zero but positive. In this way it is
possible to extract the maximum potential of the car. Despite the management of the
race pace is up to the driver, the VCU system is equipped with de-rating algorithms that
intervene by limiting power and torque in the event of high consumption. This means
that if the driver adopts too high a race pace, these algorithms come into play and
reduce the car's performance to ensure the finish line is reached.

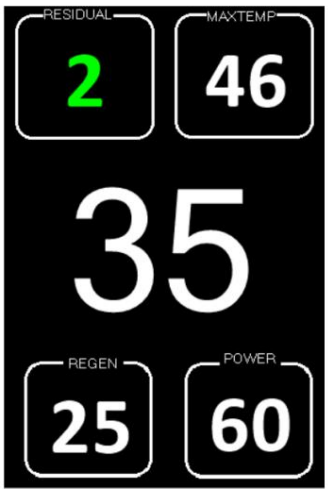

Figure 32 Endurance Page

Other data displayed are:

MAXTEMP: Temperature of the warmest cell in the battery, if this value
If the temperature exceeds 60°C the machine stops automatically as required by regulation.

REGEN: Maximum regenerative torque during braking.

POWER: Maximum power that can be supplied by the battery pack.

4.4.3 Acceleration Page

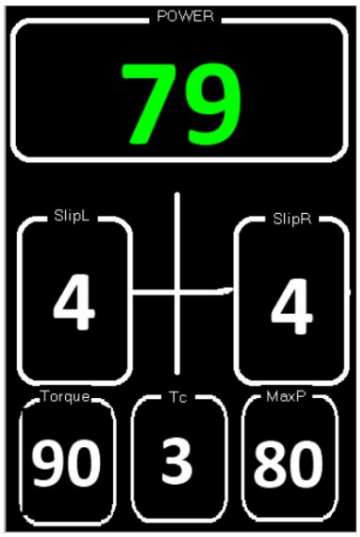

Figure 33 Acceleration Page

A page has been created for the acceleration test dedicated (figure 33), since the data relevant for this test are very different from the others. In this test, each pilot has two attempts, for a total of four, with the aim of covering a straight line of 75 meters in the shortest possible time. For to be able to obtain a good result, it is necessary get as close as possible to the limits allowed by the regulation and tires during acceleration. If the power limit imposed by the regulation is passed, the attempt is void. Using the steering wheel, you can check between a attempt and the other what was the maximum power reached and adjust it accordingly consequence. The same applies to the grip of the tires on the ground: it is displayed the maximum slip coefficient of the drive wheels, so as to know how much these have slipped during acceleration and adjust the torque and the traction control. Let's see what the individual data are for:

POWER: Maximum power delivered during the test.

SlipL: first decimal place of the *slip* coefficient of the left rear wheel.

SlipR: first decimal place of the *slip* coefficient of the right rear wheel.

Torque: Maximum torque that can be controlled by the motors

TC: traction control setting.

MaxP: Maximum power that can be supplied by the battery pack, this is the maximum limit to which the power control present in VCU works.

4.4.5 Brake Test Page

The brake test, or braking test, is one of the tests
Formula Student competition techniques.
During this test, the car is accelerated
at a predetermined speed and then brake sharply,
with the aim of blocking all four wheels
performing braking in a safe and stable manner.

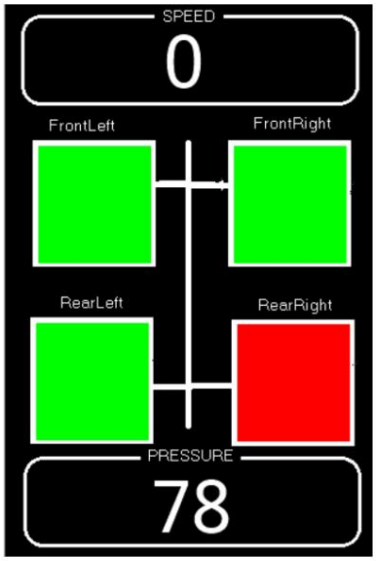

Figure 34 Brake Test Page

On the screen the pilot can see in real time the
speed at which it is going, under the **SPEED heading,** but
most importantly you can see which wheels are not working
blocked during braking. If the square is green it means
say that the wheel has locked correctly during
braking, while if it is red it means that it has not locked correctly. Under the heading
PRESSURE you can see the maximum pressure (in bar) reached by the system
braking during the test.

The algorithm that is responsible for determining whether the wheel is locked compares the
speed detected by the GPS with that detected by the phonic wheel sensor. However,
This aspect will not be explored in depth in this thesis.

4.4.6 Fault Page

There are two pages dedicated to car breakdowns, one of which concerns the breakdowns of the
BMS and the other those of the VCU. These pages are used to understand which
problem caused the car to stop. In fact, if any type of problem occurs
failure, it will no longer be possible to supply torque to the motors and therefore the machine will
will stop.

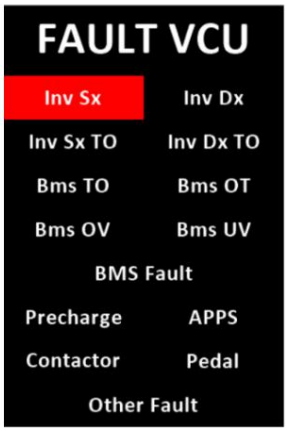

Figure 35 VCU Fault Page

Figure 36 BMS Fault Page

Looking at figure 35 above, thanks to the red colouring you can easily intuit that the faulty component is the left inverter. At the exact moment in which a fault occurs, a popup page appears for about two seconds in which report the fault.

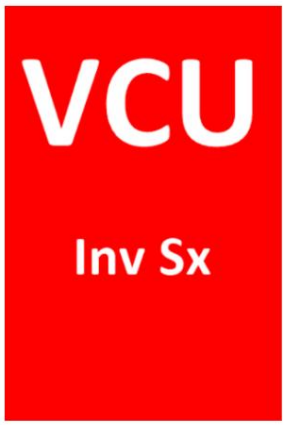

Figure 37 Popup Fault

4.4.7 Sensors page

There is also a page dedicated to the sensors of the vehicle, which is mainly used to check, when you are still in the pits, make sure all the sensors are working properly. Without this page, it would be you need to connect to the computer and download the data or view them in real time through live telemetry.

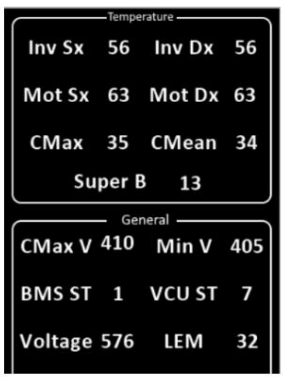

Figure 38 Sensors page

4.4.8 Parameter change popup

Every time we change a parameter through a button or by turning a knob a screen will appear on the screen popup that allows us to easily view which parameter we have changed and what value it will take. Let's suppose that having pressed the button that increases regenerative braking, it will be the page in the figure on the right is displayed on the display.

Figure 39 Parameter change popup

5 Hardware architecture

5.1 Introduction

This chapter describes the main communication protocols and devices hardware used to build the flying system. In figure 40 you can see observe the macro-components involved and how they communicate with each other. It is important say that the steering wheel is only responsible for collecting the driver's inputs and communicating them to the VCU that will manage the change of parameters and then communicate the new data once updated.

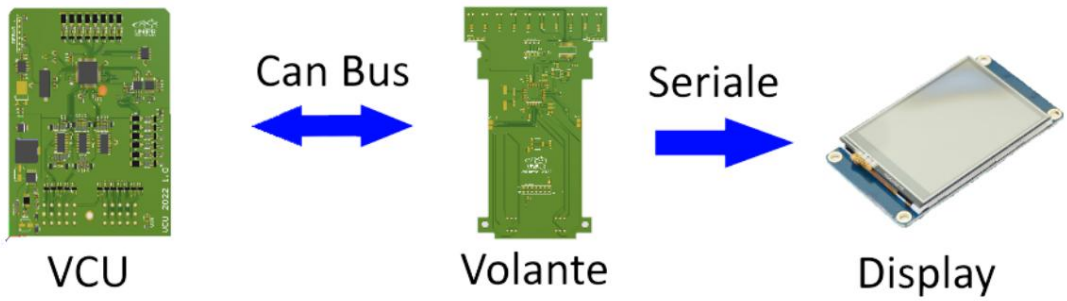

Figure 40 Hardware architecture

5.2 Hardware components involved

To best meet the functional requirements, it was decided to create a circuit dedicated printed for the steering wheel functions. This is able to perform all the necessary features thanks to the following components it is equipped with:

- STM32F412RE microcontroller that achieves a maximum clock frequency of 100MHz, equipped with CAN interface to communicate with the VCU and a USARTs interface that allows the visualization on the display of the necessary information.

Figure 41 Microcontroller

- CAN BUS Transceiver 595-SN65HVD230MDREP for arbitrating sending and receiving receiving CAN messages.

- LDO voltage regulator NCV1117ST33T3G to be able to lower the supply voltage of the flying system (5V) to that required for power the microcontroller and the transceiver (3.3V).
- 16-bit Input/Output expansion interface featuring I2C interface for control the 10 LEDs on the steering wheel.

As for the display, the NX4832T035 was chosen as it meets the functional requirements described in the previous chapter. Furthermore, the quality ratio great price and free software included called Nextion Editor for managing and creating the graphics displayed on the screen made him the ideal candidate. The screen is 3.5'' which is large enough to be able to display all the information but not too much compromise the ergonomics of the steering wheel.

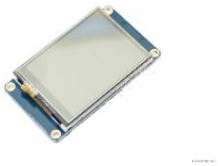

Figure 42 Nextion Display

5.3 Can Bus communication

The CAN (Control Area Network) protocol is a digital serial communication bus of the “broadcast” type (mode of transmission of packets from a broadcaster to a multiplicity of receivers). It is the most used protocol in the automotive sector thanks to the its high immunity to disturbances and its flexibility, in fact the nodes do not have an address which identifies them and can therefore be added or removed without having to reorganize the system. The protocol is defined by the ISO11898 standard which standardizes the wiring implementation methods and message structure. The CAN protocol It consists of two buses called CAN-H and CAN-L, connected by a total resistor of 60ÿ, and shared between all devices connected to the bus. The signals transmitted on the two cables They are of the differential type, that is, one cable transmits the positive value while the other transmits the corresponding value in negative. To ensure a correct reading of the bus, the receiver simply has to subtract the values transmitted on the two wires. If the result is a high voltage value, it will be encoded as "zero", while a value of low voltage will correspond to "one". This mechanism proves particularly useful in case of electrical or magnetic disturbances, as these

are transmitted on the cables with equal values which are then cancelled out via the calculating the difference.

5.3.1 Frame Bus Format

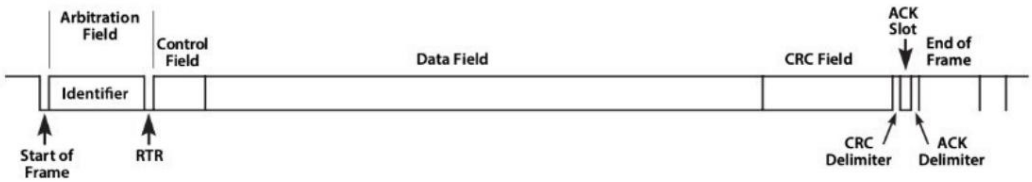

Figure 43 CAN protocol frame

The typical structure of a CAN protocol frame is shown in figure 43 and contains the following fields.

1. **Start of Frame (SOF):** a single start of frame bit that always has value dominant (0) and is used to synchronize devices on the bus.
2. **Arbitration Field:** composed of 11 bits, it is used for bus arbitration and determines the transmission priority of messages.
3. **Control Field:** composed of 6 bits, it is used to specify the length of the data field and to indicate whether the message is a Remote Frame (RTR) or a Data Frame.
4. **Data Field:** can hold up to 64 bits of data, depending on the length specified in the Control Field.
5. **CRC Field:** composed of 15 bits, it is used to detect any errors during the transmission of the message.
6. **Acknowledgment Field:** composed of 2 bits, it is used to confirm the correct reception of the message by the recipient.
7. **End of Frame (EOF):** composed of 7 bits, it is used to indicate the end of the frame and to allow the bus to return to listening mode.

5.4 Serial communication

Serial communication is a communication protocol used in many electronic devices to exchange data between each other. This type of communication is

based on the transmission of data sequentially, one bit at a time, along a single channel of transmission. The device transmitting the data sends a serial signal on the channel of transmission, which is received by the receiving device and decoded for reconstruct the original data. Serial communication has the advantage of being simple, reliable and requiring a limited number of cables for data transmission, which makes it particularly suitable for devices with limited resources such as microcontrollers or sensors.

Serial communication can occur in synchronous or asynchronous mode.

In synchronous mode, communication occurs sequentially and each device involved in the transmission must be synchronized with a signal common clock. Data is transmitted in fixed-sized blocks and with a default frequency, which ensures reliable communication but requires a increased complexity in managing synchronization.

In asynchronous mode, however, data is transmitted in packets that can have variable dimensions and the data flow is controlled by a start signal and stop. This makes communication less complex to manage, but can cause timing and transmission reliability problems if the transmission speed of the data varies.

An asynchronous mode was chosen for communication with the display because It is used for screen display, which requires a frequency of 30Hz refresh. In this case, transmission speed is a critical factor. If some packets get lost, it would not be a problem. since it would be overwritten by the next package. Furthermore, the simplicity of the Wiring plays a fundamental role, especially considering the limited spaces in which the system was built.

6 Printed circuit board design

In the realization of the steering wheel PCB, a different strategy was adopted from the
usually due to the severe silicon crisis that has hit the electronics market.
Before proceeding with the circuit design, I purchased the components
necessary, in order to avoid any supply difficulties during the
of the design process. This allowed me to have all the
components needed to build the circuit, without having to worry
of their availability on the market.

6.1 Altium Designer

Altium Designer was used to design the printed circuit board, which is a
very popular software for designing printed circuit boards. Thanks to its
numerous features and its intuitive interface, it is able to satisfy the
PCB design needs of professional electronic engineers and
students. With Altium Designer you can create electrical diagrams thanks to the vast
component library, plus thanks to the *Altium library loader* plugin it is possible
import electrical diagram and CAD of components purchased on the market. In addition,
Altium Designer offers a number of tools to generate Gerber files in a
simple and precise. These are a standard file format used in the industry
manufacturing for the production of printed circuit boards (PCB). These files contain the
information needed to create the PCB layout, tracks and areas
of copper, as well as all other elements present in the PCB design.

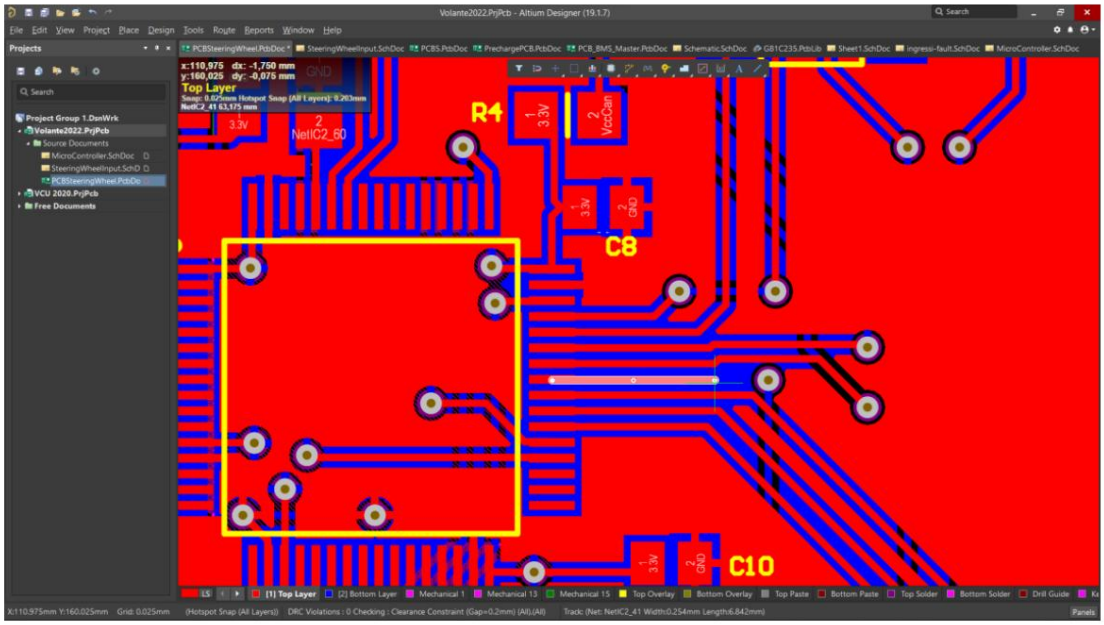

Figure 44 Altium Designer

6.2 Electrical diagram

The wiring diagram is divided into two files with SchDoc extension, one called “SteeringWheelIO” and the other called “MicroController”.

6.2.1 SteeringWheelIO.SchDoc

In this file there is the wiring diagram of the inputs/outputs that the steering wheel receives.

Connector

Starting from the connector you can see how the PCB is powered by the 5V line coming from the converter DCDC 12/5 present on single-seater. This one is present a 1A fuse since the sum of the maximum consumption of the components from datasheet yes it's around 800mA, later verified experimentally.

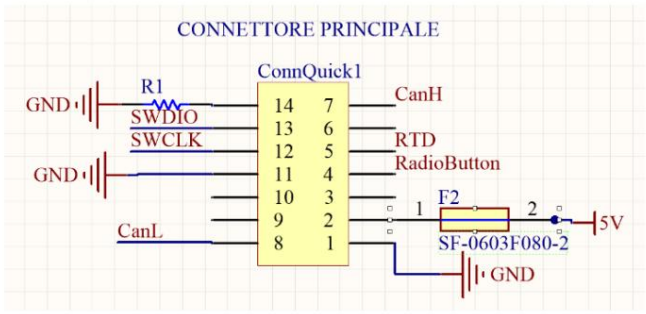

Figure 45 Flying connector diagram

CanH and Can L are the two buses used for CAN communication, in particular the steering wheel communicates on the CAN1 of the car, this is equipped with a second channel used only for communication between Inverter and VCU.

SWDIO and SWCLK are two signals that are used in DEBUG phase and code loading; in fact, they are connected to a specific tool called ST-LINK (integrated in most STM32-Nucleo boards) which allows this operation. Once the steering wheel is installed on the car, the code is loaded start it via CAN-BUS, thanks to the program called *CanBootLoader* created by former team member Eng. Davide Draghi.

RTD is the Ready To Drive button signal that allows the car to complete the ignition procedure, this involves its completion by simultaneously pressing the brake and the RTD button. This is a point critical because without this the machine cannot move, for this reason it is a signal output has been set up directly to the control unit which sends it will read as digital input, the control unit acquires the button reading also through a CAN BUS message forwarded from the steering wheel, in this way we have a signal redundancy that increases the reliability of the car.

RadioButton is a digital output that should activate a possible radio system. It has been prepared for possible future use.

Handcuffs

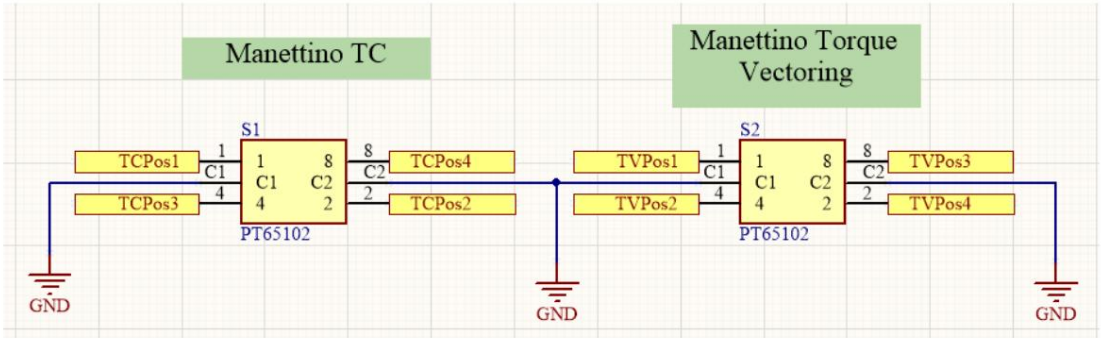

Figure 46 Rotary switch wiring diagram

To change the Traction Control (TC) and Torque Vectoring (TV) settings are
two coded rotary switches were used. For both knobs the coding
of pins 1,2,4 and 8 is in BCD and is represented by the table in figure 47

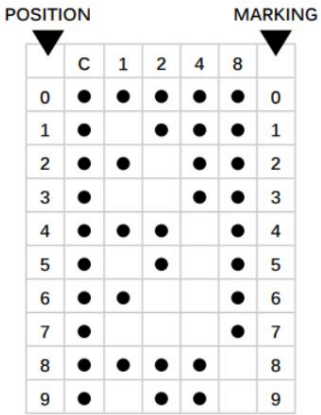

Figure 47 BCD encoding manettini

In this case, since there are only 4 bits, the BCD encoding
It does not differ from the binary number making it simpler
input processing.

Taking the Traction Control knob as an example,
column 1 in the figure corresponds to the TCPos1 signal, column 2
to the TCPos2 signal, and so on.

So for example if the lever is in position 3 the coding will correspond
exactly at binary number 3 and therefore we will have the signals TCPos3, TCPos4
shorted to the common signal (which corresponds to the ground reference of the
circuit) and therefore will have a logical value of 0, while TCPos1 and TCPos2 will have a value
of 3.3V thanks to the internal pull up in the microcontroller (it must be enabled via software)
thus assuming a high logical value. The same goes for the Torque knob
Vectoring.

Buttons

Push buttons are switches that connect to a digital input on the
microcontroller, with one end connected to the circuit's ground signal. When the
button is pressed, the circuit closes, and the microcontroller input takes on
the mass value. To avoid button bounce, i.e. transient oscillation
of the signal when the button is pressed or released, in some cases it is

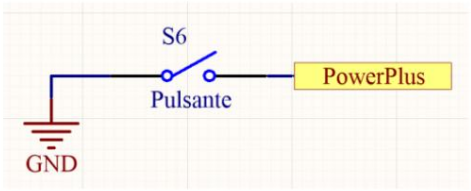

Figure 48 Button wiring diagram

used an anti-bounce circuit. However, in
In this case, the use of such a device was not foreseen
circuit, since the signal processing
digital input occurs at a frequency of

100Hz, reducing the noise due to signal switching. To ensure
that the microcontroller correctly recognizes the state of the button, the software
requires that the variable associated with the button maintains the same value for a
default number of readings before changing state. This helps filter out any
noises or transient oscillations that could influence the read value of the
button.

The Pull-Up circuit, as for the knobs, is internal to the microcontroller and must be
activate it via software.

LED

In the figure you can see the circuit that allows you to turn on the 10 LEDs
present on the steering wheel. In this circuit, the anode of each LED
is connected to one of the I/O expander output signals,
while the cathode is connected to ground via a resistor
of 110ÿ.

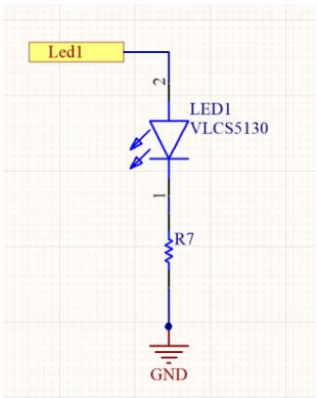

Figure 49 LED diagram

This resistor is used to limit the current
which passes through each LED at a maximum of 25mA. In fact, the
LED voltage drop is about 2.2V and the maximum
current that can be supplied by the expander output is 25mA. Therefore, choosing a
110ÿ resistance, the current flowing through the LED is limited to 25mA,
ensuring the correct functioning of the circuit.

Dashboard Connector

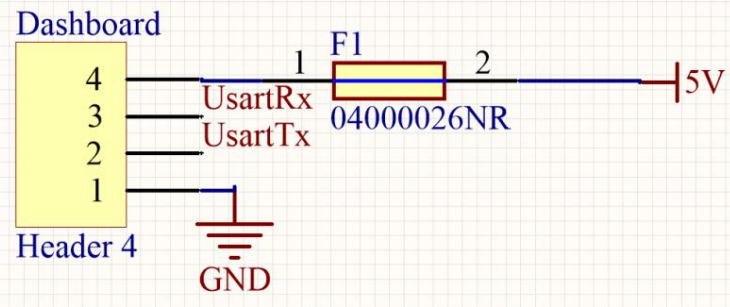

Figure 50 Dashboard Connector

In figure 50 you can see the connector that connects the card to the display.
The 5V power supply of the display is protected by a fuse rated at 500mA, which represents the maximum current supported by the Nextion Display, as indicated in the datasheet.

The UsartRx and UsartTx signals constitute the transmission and reception channel of the serial communication between the display and the microcontroller. When connecting the display, you need to connect the UsartRx signal to the serial transmit pin of the display. This is because the microcontroller's receiving channel is used as transmission channel from the display, similar reasoning for the UsartTx signal.

6.2.2 Microcontroller.SchDoc

This file contains all those components that process signals in order to generate an output.

LDO Converter

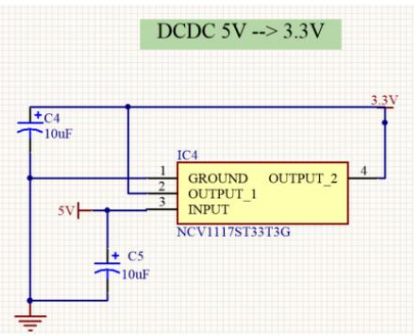

Figure 51 LDO converter

The NCV1117ST33T3G LDO converter takes care of convert supply voltage to 5V present on the board at a voltage of 3.3V, needed to power the microcontroller and the CAN bus transceiver.

The polarized capacitors shown in figure 14 are were chosen following the instructions on the

converter datasheet. The datasheet provides the technical specifications and recommendations for using the component, including the placement of the capacitors and the capacity to be used. In particular, polarized capacitors are used to filter the noise present on the input voltage and on the converter output voltage.

Tranceiver Can Bus

The CAN bus transceiver is responsible for managing the transmission and reception of data via the CAN bus. In practice, The transceiver converts the signals digital coming from microcontroller in signals analog bus-compatible and vice versa.

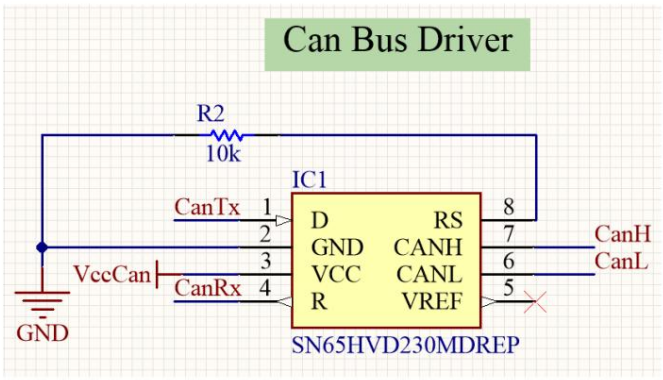

Figure 52 CAN bus transceiver wiring diagram

The CanH and CanL signals are analog signals used by the CAN bus to transmit and receive data differentially. These signals are processed by the transceiver to convert them into digital signals that can be managed by the microcontroller.

CanTx and CanRx are the digital transmission and reception signals respectively. microcontroller. In other words, CanTx is the digital signal used by the microcontroller to transmit data through the CAN bus, while CanRx is the signal used by the microcontroller to receive data from the CAN bus. The transceiver is responsible for converting these digital signals into analog signals compatible with the bus and vice versa, in order to guarantee correct communication between devices connected to the bus.

I/O Expander

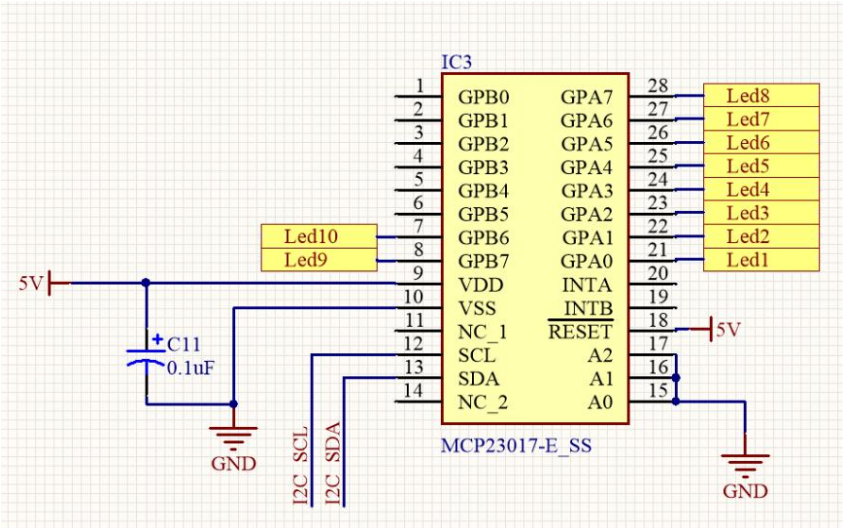

Figure 53 I/O expander wiring diagram

The MCP23017-E_SS I/O expander is responsible for providing power to the 10 LEDs. of the steering wheel control system. This power is generated through the signals sent with the I2C protocol.

Thanks to this component, it was possible to avoid directly connecting the LEDs to the microcontroller using 10 digital inputs. Instead, they were used only the 2 pins of the I2C bus to control the LEDs, simplifying the wiring and reducing the number of pins required for the circuit to function.

Microcontroller

In the figure we can see which pins of the microcontroller have been connected to all the signs mentioned above

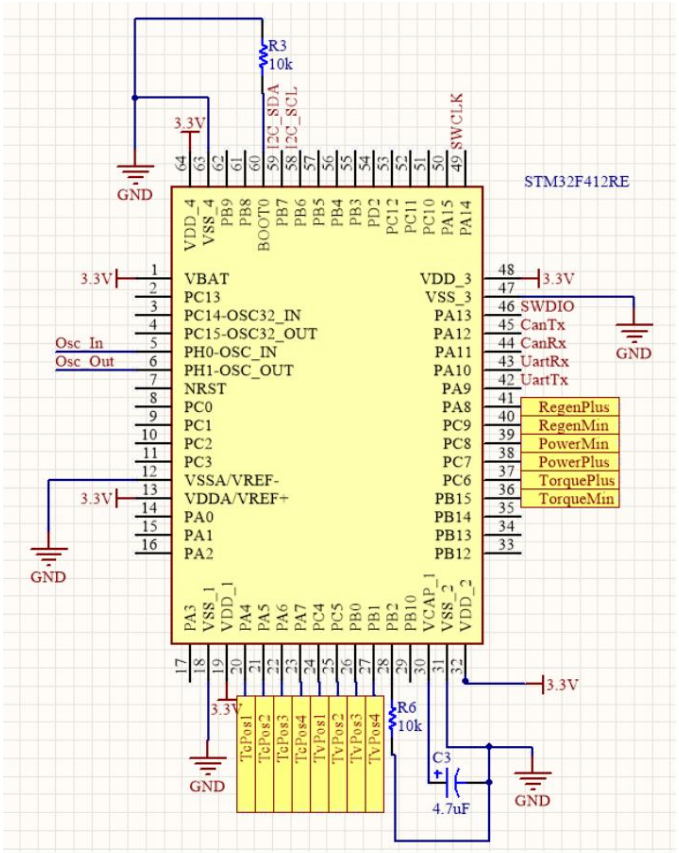

Figure 54 Microcontroller circuit diagram

6.3 PCB Design

The printed circuit board was made of FR4 which is a class of material used for printed circuit boards based on a flame retardant epoxy resin and a fabric composite of glass. FR stands for *Fire Retardant* and meets UL94V-0 requirements.

The printed circuit board is made up of two distinct layers: the first hosts the circuit board power supply and accommodates all the integrated circuits and the connector for the side wiring machine. The second layer, instead, faces the pilot and presents the plane of ground, plus four pads where the display wiring leads will be soldered.
Also, on this layer are the holes where the switches were soldered.
rotary and LEDs.

6.3.1 PCB shape

Before proceeding with the design of the layout of tracks, via and integrated, it was essential to collaborate with the Frame department to define together the external structure of the steering wheel and find a solution to fix the electronic board to it. In particular, it was also necessary to consider the position of the wiring that reaches the steering wheel, in order to maintain the integrity of the same. It was therefore decided to run the cables inside the steering column, fixing them to it in such a way so that they rotate with the steering wheel itself, thus avoiding any possible damage. The electronic board was instead fixed inside the steering wheel frame, using four M3 nylon screws to minimize vibrations that could compromise the functioning of the system.

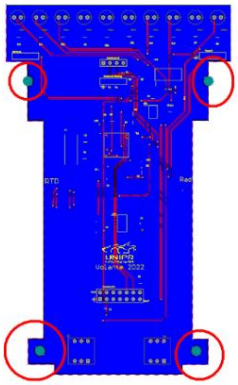

Figure 55 PCB anchor holes

6.3.1 Component Placement

Collaboration was also essential for the positioning of the components with the Chassis department and with the driver, since the positions of the buttons and levers They were chosen on the basis of ergonomics and speed of use when you are on the go guide. Furthermore, the LEDs have been positioned at the highest possible point so as to be visible peripherally to the driver, who will be able to assimilate the information derived from them without having to take your eyes off the track.

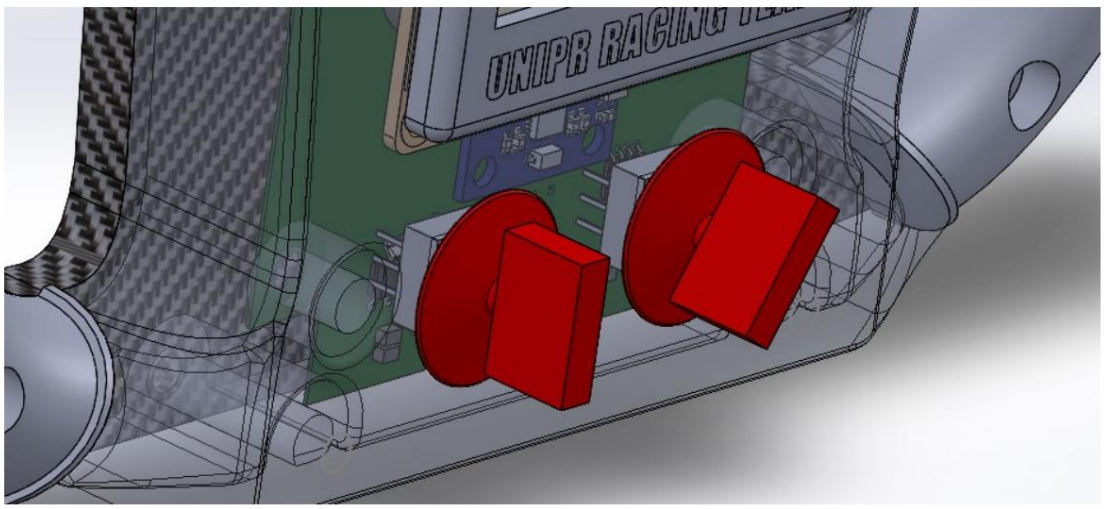

Figure 56 Manettini Positioning

Since the knobs and LEDs are of “through holes” technology it was necessary position the holes where they will be welded exactly below their position in the external structure of the steering wheel. This process was facilitated by the possibility to export the PCB to CAD file so that mechanical engineers could integrate it in the car assembly and check that the positioning of the components was correct or not.

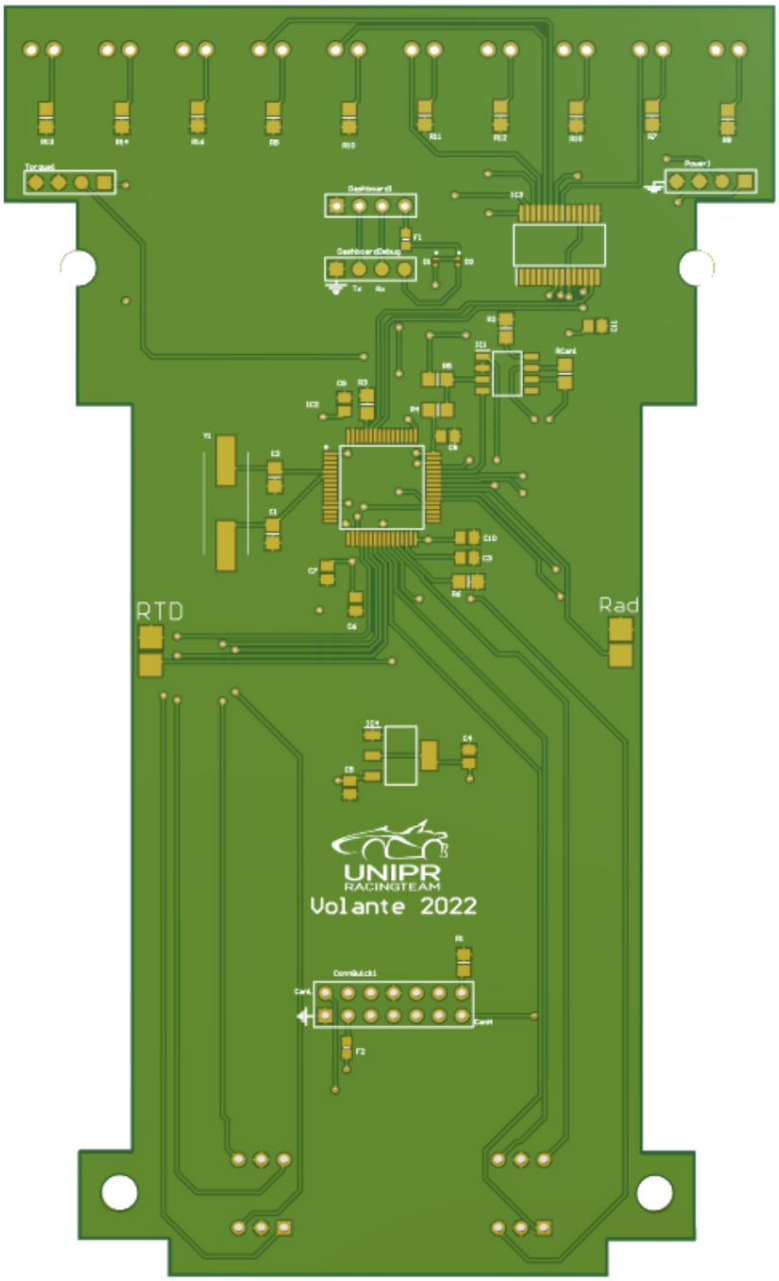

Figure 57 PSR02-S Flying PCB

6.4 Screen integration

To connect the display to the board, directly soldered cables were used
on the holes present on the flying PCB side and, on the other side, a connector was used
(figure 58). It was not possible to use a traditional connector even on the
flying due to limited space. In total four cables were used, two
for power supply (GND and 5V) and the other two for the serial transmission channel (cable
blue), which will be connected to the display receiving pin, and the serial receiving channel
(yellow wire), which will be connected to the display's transmit pin.

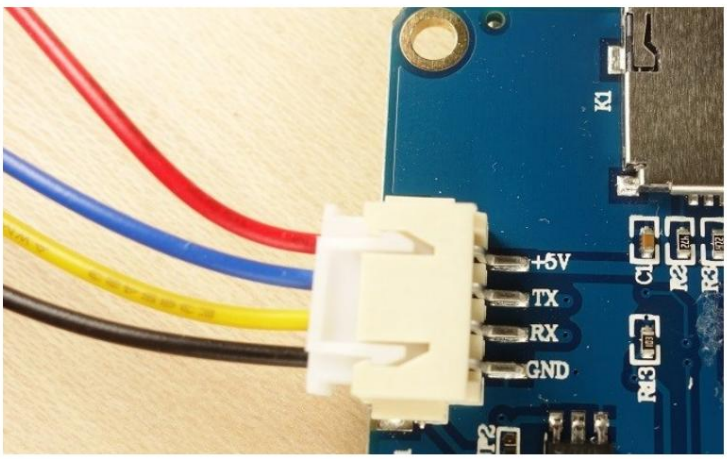

Figure 58 Display side connector

7 Software design

7.1 Introduction to software design

7.1.1 V-model

In the automotive industry, the V-model represents a well-established approach for software development. This method highlights the importance of testing in every stage of the development process, which makes it particularly suitable for creation of systems that require high security, such as those defined Safety Critical. By Safety Critical we mean a system where a possible malfunction or error may cause serious injury or even death of people, or significant damage to property.

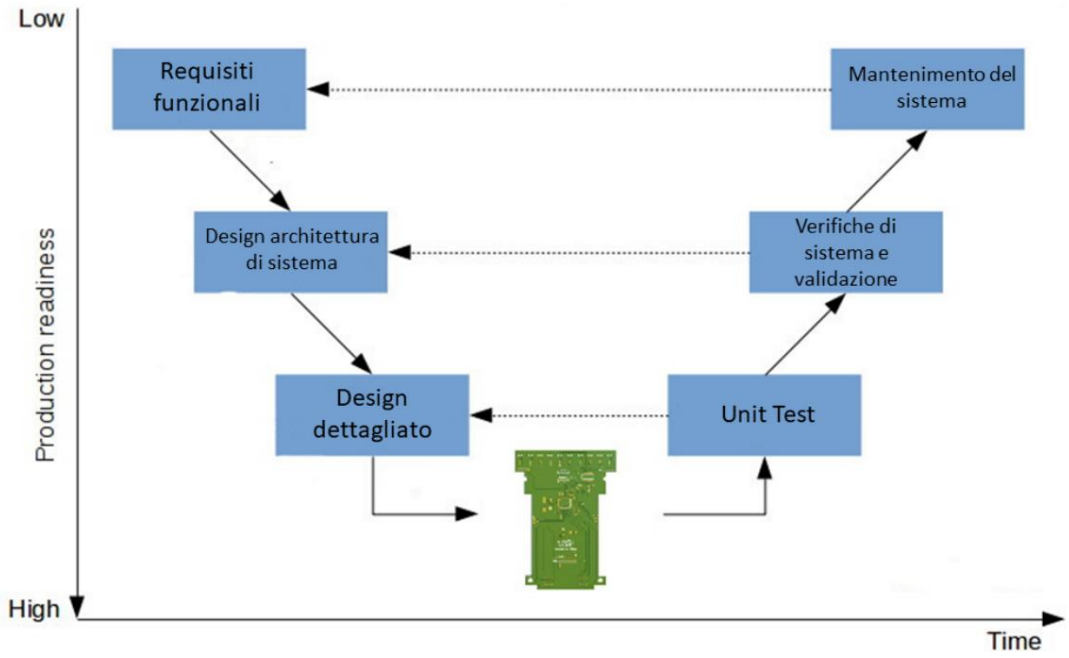

Figure 59 V-model diagram

Figure 59 summarizes the key steps of the V-model, which are:

- **Definition of requirements:** in this phase the high level requirements are established system level, that is, what functions and features it must have the system.

- **Architecture definition:** in this phase the specifications are defined
  architectural aspects of the system, such as the division into modules and the assignment of
  desired requirements for each party.
- **Detailed design:** in this phase the complete specifications of the
  system, detailing how to implement the features.
- **Implementation:** This phase consists of the actual development of the system.
- **Unit testing:** in this phase each part of the application is tested individually
  system to verify that it meets the required specifications.
- **System checks and validation:** after passing the unit tests, we start
  to integrate and verify the parts of the overall system and then test the
  same.
- **System maintenance:** this phase consists in resolving any
  problems that occur after the system is completed and, if
  necessary, in adding new features.

7.1.2 Low coupling system

Low coupling in a software system refers to the reduction of
dependency between different parts of the system.

Low coupling in a software system has several advantages, including:

1. **Greater modularity:** with low coupling, the system can be
   divided into independent and easily replaceable modules, allowing for a
   greater flexibility and making it easier to upgrade or maintain.
2. **Greater scalability:** Low coupling allows for expansion of the
   system without compromising the stability or quality of the software, as
   Adding new modules does not interfere with existing ones.
3. **Easier testing:** A system with low coupling is more
   easy to test, as individual modules can be tested
   independently of each other, allowing you to identify and resolve
   any problems more quickly.

In a Formula Student team, the ease of understanding and maintenance of the software is essential, especially considering the constant changes generational changes that occur within the team. Low coupling of the software allows you to avoid the risk of abandoning a system due to its complexity, as it makes it possible to make improvements and changes in quick and easy way without having to rebuild the entire system from scratch.

For this reason the steering wheel software has been divided into three modules:

1. Firmware
2. Strategy
3. Display libraries

Even though they are three modules with low coupling between them, the strategy and the libraries of the display are integrated into the firmware which is then loaded onto the microcontroller. This concept will be clarified in the following chapters.

7.1.3 Software development with Matlab/Simulink

To develop the most important part of the code, that is, the one that processes the inputs acquired from the firmware, we used Matlab/Simulink. This tool gave us allowed to adopt a block approach, which greatly simplifies the understanding of the logic of the code by anyone approaching the project the first time. Additionally, Simulink allows you to create sub-blocks to hide the details related to them, unless you want to delve deeper into how they work. Ad for example, in figure 60 we have a more general view of the block for the displaying the error pop up, while in figure 61, through a simple double click you can see the content of the block in detail. This

allows you to maintain a clear and precise organization within projects
very large, such as the steering wheel.

Figure 60 DisplayFault Block

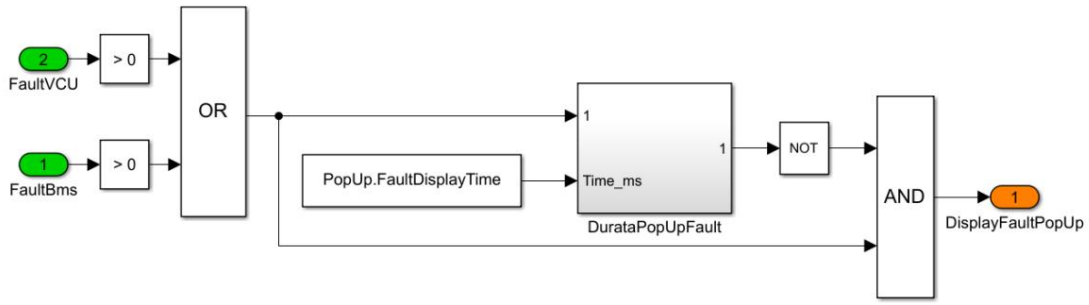

Figure 61 Inside the DisplayFault block

Another important advantage of using Matlab/Simulink is the
ability to simulate the entire system or a portion of it, as well as a single
block, as needed. This integrates perfectly with the V-model
mentioned previously.

7.1.4 Auto-generate code

Once the block logic has been developed and tested, it is possible to generate in
automatically generate C code for a given microcontroller by using
combined Embedded Coder and MATLAB Coder software, also developed by
MathWorks, the company that created Matlab and Simulink. The generated libraries can
be easily integrated into the project, allowing you to incorporate in just a few
simple steps all the work done on Matlab/Simulink in the firmware code.

This approach undoubtedly has the disadvantage of creating less efficient code.
efficient than the one written manually. However, it has several advantages
compared to the traditional approach:

- development is much faster

- software reuse is much easier, in fact to migrate from one family
  from one microcontroller to another, you only need to change the settings
  code generation and not manually rewriting some parts
- the probability of having errors in the code is much lower
- the software is easier and faster to test thanks to the ability to simulate the
  blocks in Simulink.

Since the hardware used is powerful enough to compensate for the lower
code optimization, and since maintainability and ease of understanding of the
code are among the main goals during software development within the
team, it was decided to adopt the approach of self-generating code.

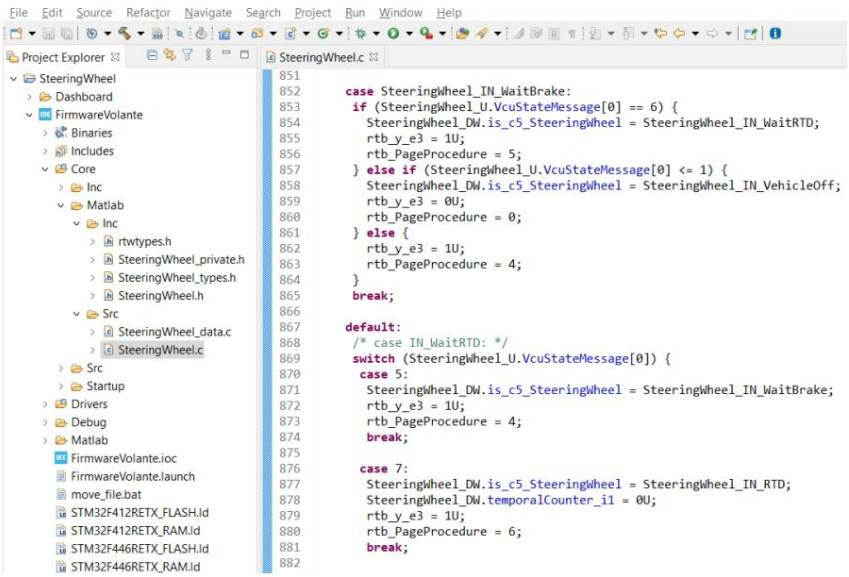

Figure 62 Auto-generated code screenshot

7.2 Firmware

The exact same firmware structure was used for both drives.
Vehicle Control Unit (VCU) and Battery Management System (BMS), as it is
the same family of microcontrollers (STM32F4) was used. This choice has
permission to use the STM32CubeIde development environment and the developed libraries
from eDriveLAB to simplify the management of the main peripherals (CAN bus,
analog inputs, digital inputs, digital outputs, timers).

Thanks to STM32CubeIde, it is possible to configure peripherals through a graphical interface that allows you to automatically generate C code, device settings are saved in the ".ioc" extension file. This approach simplifies and speeds up the process of implementation and maintenance of the code.

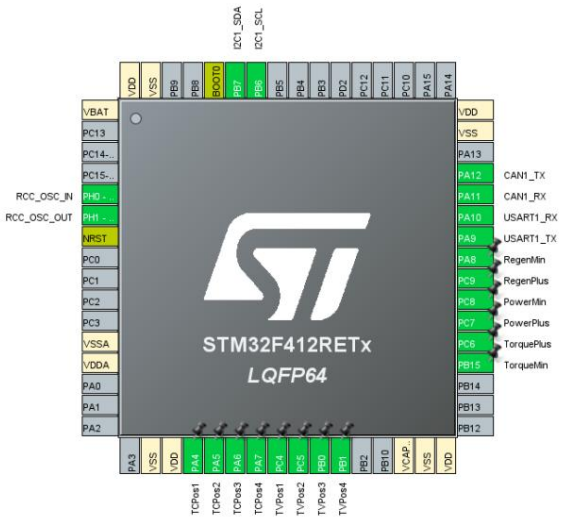

Figure 63 Microcontroller Input Output

7.2.1 CAN Peripheral Configuration

For communication on the CAN bus, used PA12 pins for transmission and PA11 for reception, which communicate on the bus with a baud rate of 250,000 bits/s. In figure 64 the settings used for the are shown communication.

Every time a message is received or sent on the bus, an interrupt is thrown for execute the corresponding routine.

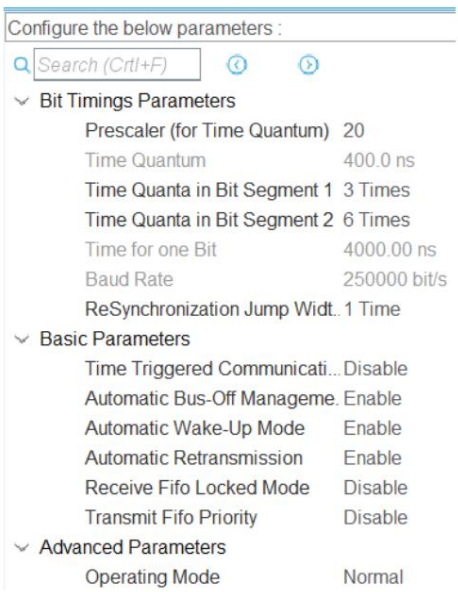

Figure 64 CAN bus BaudRate Configuration

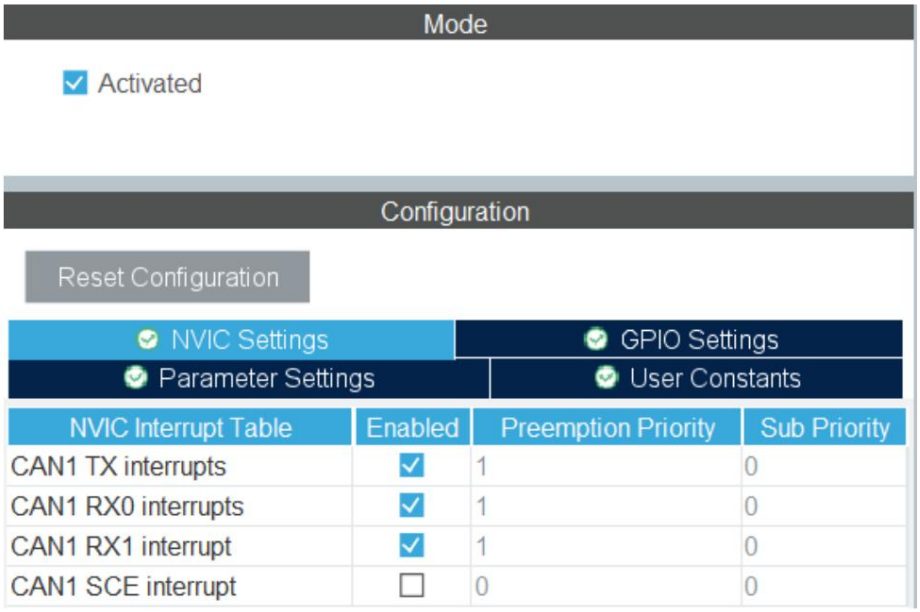

Figure 65 CAN Message Interrupt Configuration

7.2.2 USART Device Configuration

The USART (Universal Asynchronous Receiver-Transmitter) device takes care of the serial communication between the microcontroller and the display. It has been set for transmit at a baud rate of 512,000 bits/s and with a word length of 8
bit. As mentioned in previous chapters, the transmission mode is asynchronous.

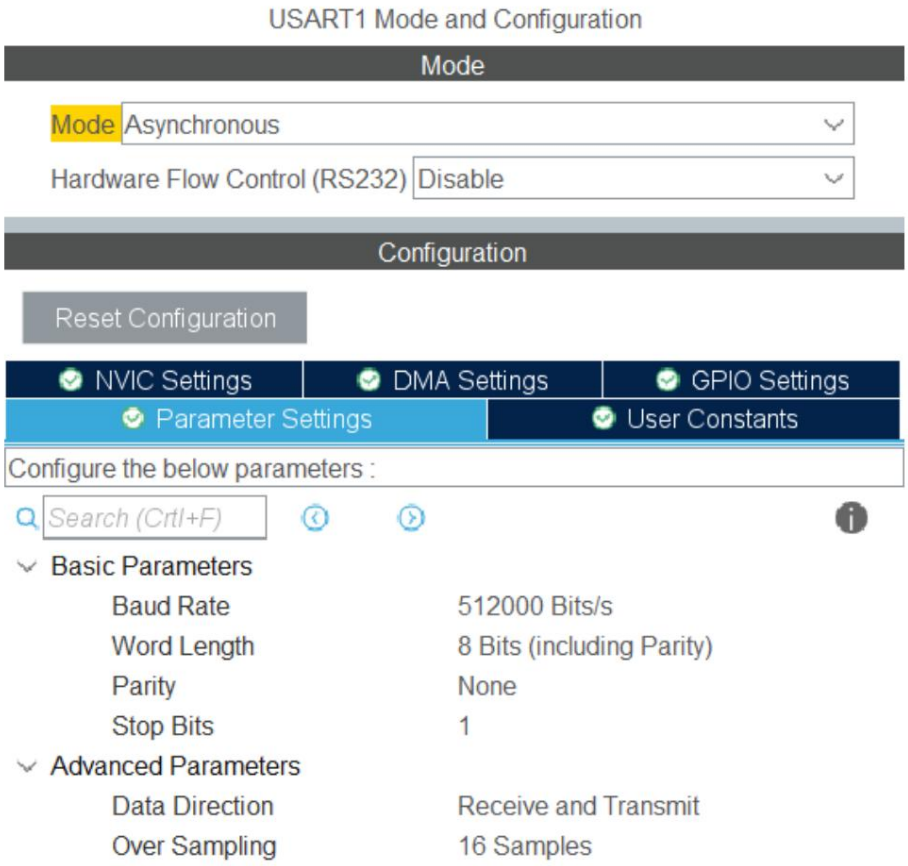

Figure 66 USART BaudRate Configuration

7.2.3 Digital inputs

The digital input pins of the microcontroller have been set with the pull-up internal active. This means that the microcontroller provides a resistor of internal pull-up to the digital input pin, which holds the logic value high when the pin is floating, that is, it is not connected to any input signal. In this way, it is avoided the instability of the input signal value and ensures a correct reading of the signal.

7.2.4 Timer Settings

The firmware has three timers:

- TIM 10, which generates an interrupt with a frequency of 30Hz, takes care of call the routine for updating the information displayed on the monitor.

- TIM 13, which generates an interrupt with a frequency of 100Hz, is responsible for send the output messages generated by the strategy on the CAN bus.
- TIM 14, which generates an interrupt with a frequency of 100Hz, is responsible for recall the basic strategy.

Timers are first enabled in the ".ioc" file and then through the functions listed in figure it is possible to set the frequency with which the interrupt is generated.

```
void STRATEGY_SETUP(void){
    canTxStrategyInit();
    canRxStrategyInit();

    HAL_CAN_Start(&hcan1);
    HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING);
    HAL_CAN_ActivateNotification(&hcan1, CAN_TX_MAILBOX0);
    HAL_CAN_TxMailbox0CompleteCallback(&hcan1);

    mcp23017_init(&hVoltage, &hi2c1, MCP23017_ADDRESS_20);

    SteeringWheel_initialize();
    init_bootloader(ID_BOOTLOADER,"Volante");

    setCanFrequency(100);
    setBaseFrequency(100);
    setPageFrequency(30);
}
```

Figure 67 Functions to set the timer frequency

Whenever the microcontroller detects an interrupt signal from the timers The **HAL_TIM_PeriodElapsedCallback** function is called by passing as parameter a pointer to the timer instance. Thanks to this we can trace back to which timer generated the interrupt and call the routine associated with it.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance == TIM14) /*CONTROLLO LA SORGENTE DELL'INTERRUPT*/
    {
        StrategyMainRoutine();

    }
    if(htim->Instance == TIM13) /*CONTROLLO LA SORGENTE DELL'INTERRUPT*/
    {
        Can1TxRoutine();
    }
    if(htim->Instance == TIM10){
        UpdateDisplay();
    }

}
```

Figure 68 Code portion called when a timer expires

7.2.4 Firmware structure

When the flying system is powered, the firmware is loaded into the microcontroller starts its execution. The main function is executed first , even this one is autogenerated and therefore already has the lines of code needed to make it work peripherals. In the **main program,** only the **StrategySetup** function is called, which is takes care of initializing the timers described above, initializing the communication on CAN bus, initialize communication with I2C protocol with input expander output to be able to turn the LEDs on and off and last but not least calls the **SteeringWheelStrategy_initialize** function which is the function that allows you to initialize libraries autogenerated by Matlab/Simulink.

```c
int main(void)
{
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */
  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_CAN1_Init();
  MX_I2C1_Init();
  MX_TIM10_Init();
  MX_TIM13_Init();
  MX_TIM14_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */
  STRATEGY_SETUP();
  /* USER CODE END 2 */
```

Figure 69 Firmware Main Function

```c
void STRATEGY_SETUP(void){
    canTxStrategyInit();
    canRxStrategyInit();

    HAL_CAN_Start(&hcan1);
    HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING);
    HAL_CAN_ActivateNotification(&hcan1, CAN_TX_MAILBOX0);
    HAL_CAN_TxMailbox0CompleteCallback(&hcan1);

    mcp23017_init(&hVoltage, &hi2c1, MCP23017_ADDRESS_20);

    SteeringWheelStrategy_initialize();
    init_bootloader(ID_BOOTLOADER,"Volante");
    setCanFrequency(100);
    setBaseFrequency(100);
    setPageFrequency(30);
}
```

Figure 70 Firmware SetUp function

The previously mentioned Timer 14 calls the function **StrategyMainRoutine** every 10 milliseconds. The function is responsible for acquiring the digital inputs from the buttons and knobs on the steering wheel and insert them into a variable temporary call **SteeringWheel_U_Tmp** of type ExtU_SteeringWheel_T .

ExtU_SteeringWheel_T structure is a struct autogenerated by the developed strategy in Matlab/Simulink and includes all the inputs it needs to function.

This means that once the structure has been populated, we will have provided all

the inputs needed to execute the entire strategy. The temporary variable is used to avoid changing the inputs during the execution of the strategy itself. In fact, before calling the **SteeringWheelStrategy_step function,** the structure temporary is copied to the one that will actually be used by the portion of autogenerated code. The actual structure is called **SteeringWheel_U.**

```
void STRATEGY_MAIN_ROUTINE()
{
    UpdateStrategyInput();
    memcpy(&SteeringWheel_U, &SteeringWheel_U_Tmp, sizeof(SteeringWheel_U));
    //copio la struct della strategia simulink temporanea in quella che poi verrà usata
    SteeringWheelStrategy_step();
    turnOnLed();
}
```

Figure 71 Firmware main routine

```
void UpdateStrategyInput(){
    SteeringWheel_U_Tmp.PowerPlusButton = !read_PowerPlus();
    SteeringWheel_U_Tmp.PowerMinButton = !read_PowerMin();
    SteeringWheel_U_Tmp.TorquePlusButton = !read_TorquePlus();
    SteeringWheel_U_Tmp.TorqueMinButton = !read_TorqueMin();
    SteeringWheel_U_Tmp.RegenPlusButton = !read_RegenPlus();
    SteeringWheel_U_Tmp.RegenMinButton = !read_RegenMin();
    SteeringWheel_U_Tmp.TcIn = readTcRotativeSwitch();
    SteeringWheel_U_Tmp.TvIn = readTvRotativeSwitch();
}
```

Figure 72 Function that acquires pilot inputs

```
uint8_t readTcRotativeSwitch(){
    return (read_TCPos1() + (read_TCPos2() << 1) + (read_TCPos3()<<2) + (read_TCPos4()<<3));
}
```

Figure 73 Lever position reading function

The switching on of the LEDs is managed within the turnOnLed function, which uses the open source library mcp23017. The latter was written specifically for communicate with the mcp23017 I/O expander via the I2C interface. This choice was carried out exclusively to speed up development times. The validation of the library was made by uploading a short code to the microcontroller, which turned the LEDs on and off in sequence.

7.3 Strategy

As previously anticipated, the strategy is written using Matlab/Simulink and the structure of this is divided into three parts:

- Input processing (in green)
- Output generation (grayed out)
- Output processing (in orange)

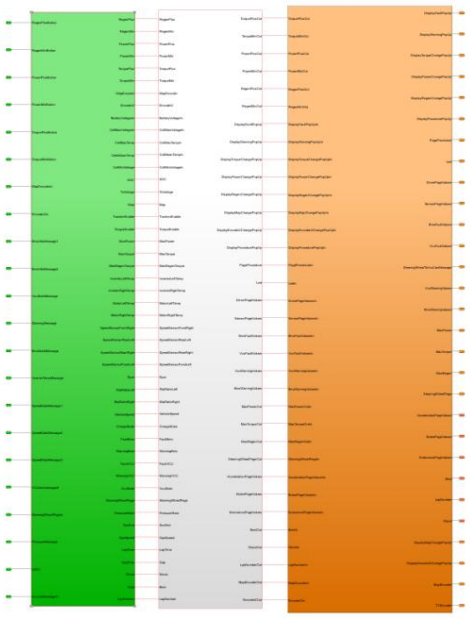

Figure 74 Strategy structure

Values are used within the model constants that are reported in the file extension ".m". Among these we have the parameters to unpack a message CAN, such as the index to which starts the data, whether it was saved in Little Endian rather than Big Endian, etc.

7.3.1 Input processing

Buttons

The buttons, being digital inputs that can assume two values, are represented with a boolean variable, where the value *true* represents the button pressed and the value *false* the button released. Since the strategy is called with a frequency of 100Hz, it was not necessary to apply an anti-bounce filter, since this frequency already acts as a filter in itself. Consequently, the input is reported to the output generation block without undergoing any changes. In case you If you experience button bounce problems, simply apply an anti-bounce as shown in figure 76. This filter allows the logical value to pass

high only if this remains constant for a number of milliseconds equal to the constant "ButtonDebounceTime".

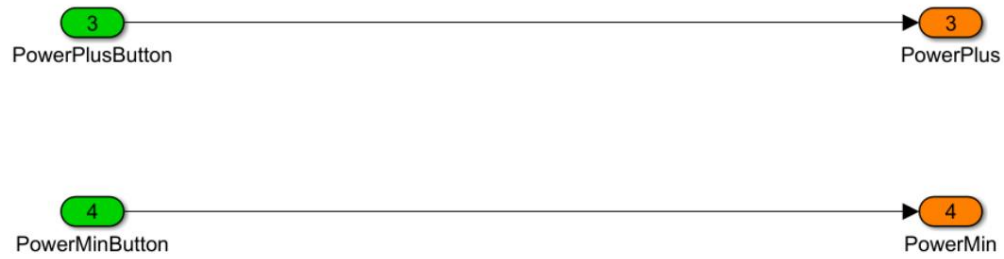

Figure 75 Unfiltered Button Input

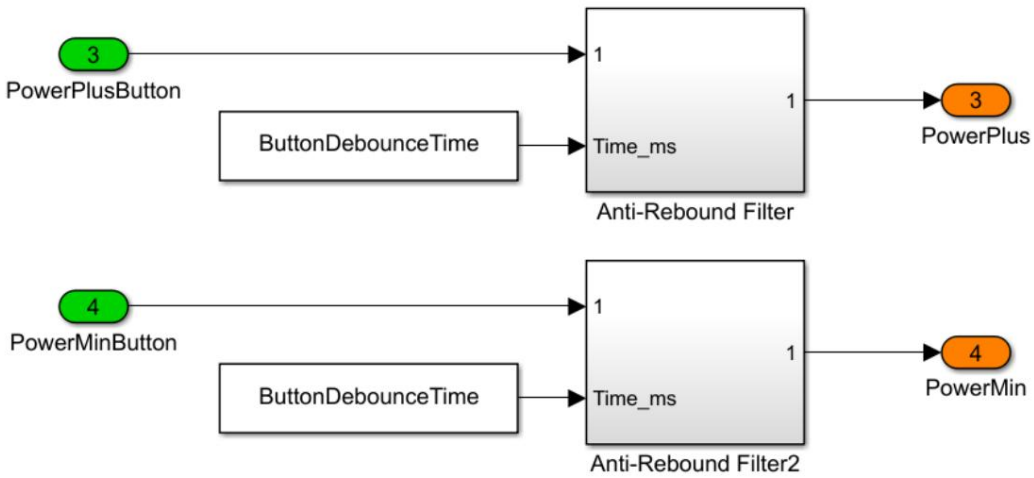

Figure 76 Button input with debounce filter

Rotary Switches

The 4 digital inputs corresponding to a single switch are encoded in BCD on the firmware that "passes" them to the strategy in the form of uint8 or an integer unsigned 8-bit. Here too, no filtering was needed but if if necessary, the same considerations made for the buttons apply.

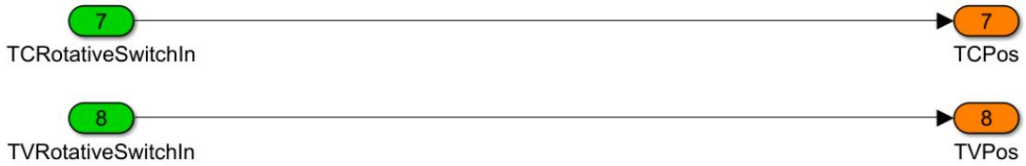

Figure 77 Unfiltered manettini inputs

CAN Messages

Once the firmware has managed to identify the message thanks to the ID it inserts the payload in the corresponding input of the strategy. the payload is represented as an 8-byte array (uint8). This fixed length is a feature of the CAN protocol, which specifies a maximum payload size of 8 bytes.

In the input processing block, the message is broken down to get the data contained within it. The file with the ".m" extension indicates the beginning of the data, its length in bits, its offset and its scale factor. The following is reported an example of a message breakdown from Battery Management System (BMS) containing data relating to the battery.

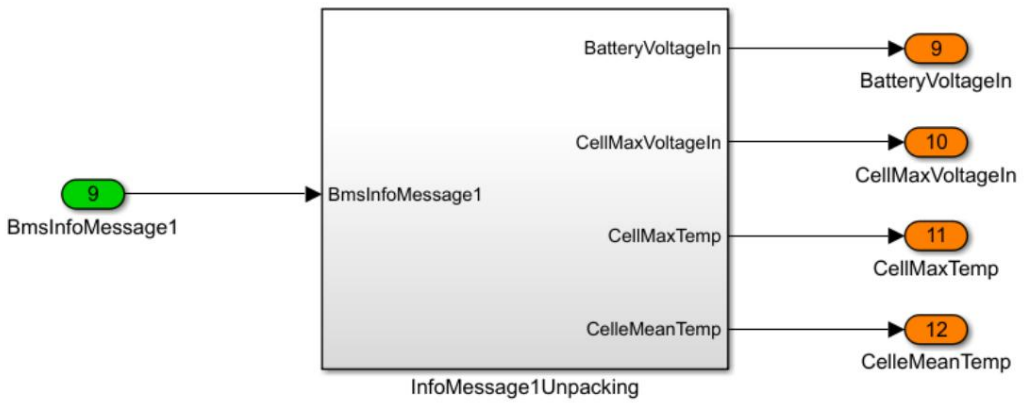

Figure 78 CAN message breakdown block

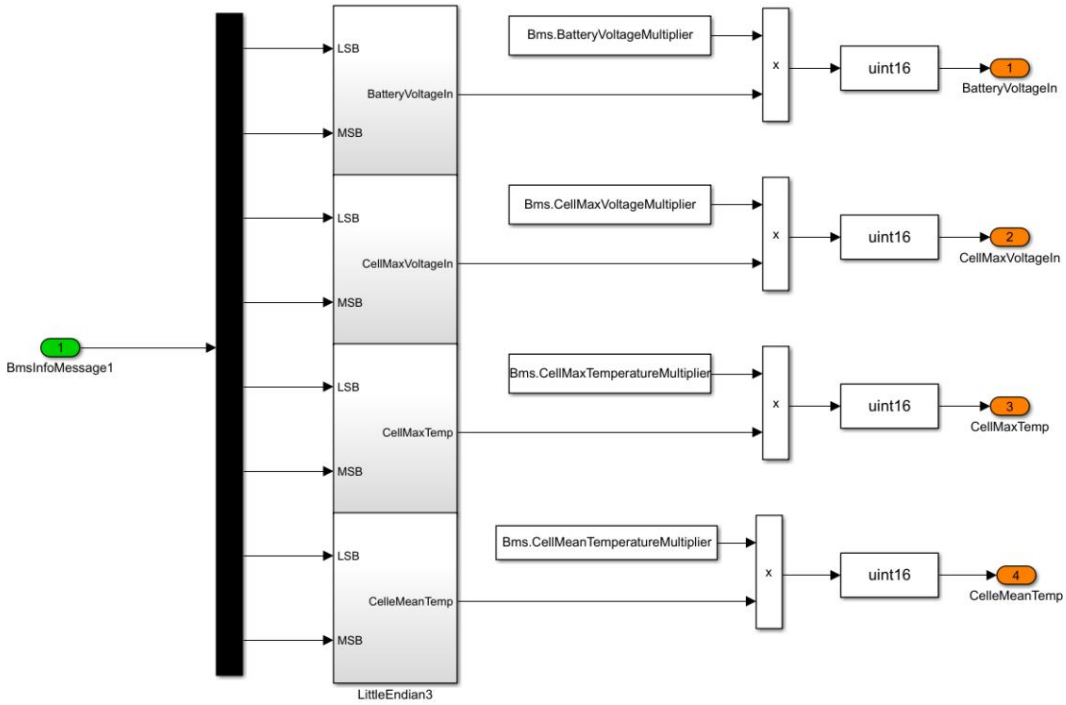

Figure 79 Inside the CAN message breakdown block

7.3.2 Output generation

Error Popup

When a fault occurs from BMS or VCU we want the following to be displayed on the screen:
an error message appears clearly so that the pilot is aware of the
problem and can immediately stop the car and take action
consequence. The strategy is responsible for generating a boolean variable that indicates whether
whether the PopUp should be displayed or not.

When the variable *BmsFault* or *VcuFault* (both obtained in the block
of input processing by breaking down the related CAN message) is greater than
zero I will have to display the popup, after a few seconds I want the error warning
disappear so that the pilot can view the rest of the data and act in the best way.
The same goes for the Warning screen.

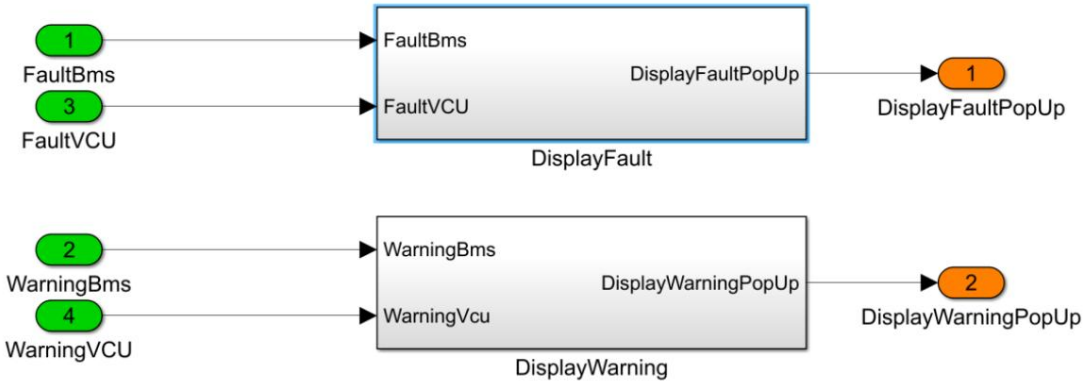

Figure 80 Block to set the DisplayFaultPopUp and DisplayWarningPopUp variables

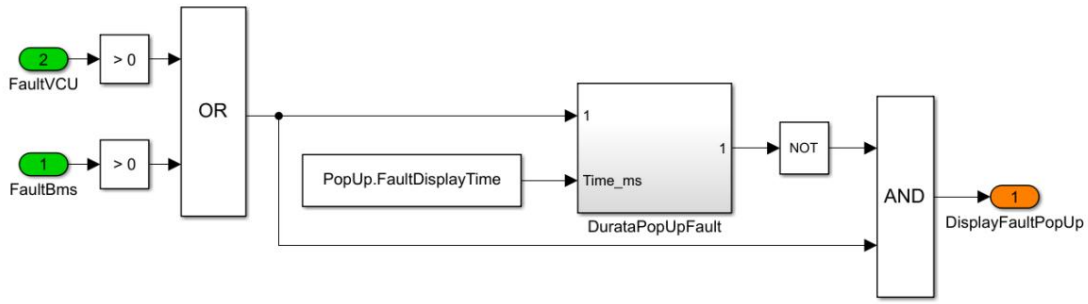

Figure 81 Interior of the block of Figure 80

Popup buttons

Every time the pilot presses a button or changes the position of the lever, it is essential to display a clearly visible message indicating which parameter has been changed and what is the new value it has assumed. Unlike error messages that have absolute priority and are displayed at Regardless, this type of notice may be subject to cases of uncertainty due to the pressing multiple buttons simultaneously or in rapid succession.

This is all managed by a state machine that assigns priority to the last one button pressed. In practice, the boolean variable that is set to "true" indicates whether the popup associated with that button should be displayed or not, while all the others boolean variables are set to "false". After a certain interval of time, corresponding to the value of the constant "PopUp.ChangePopUpTime" expressed

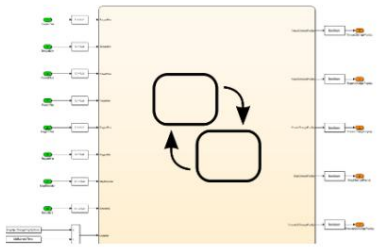

Figure 82 State machine pop-up view

in milliseconds, the last boolean variable is also set to "false"
corresponding to the last button pressed or lever turned.

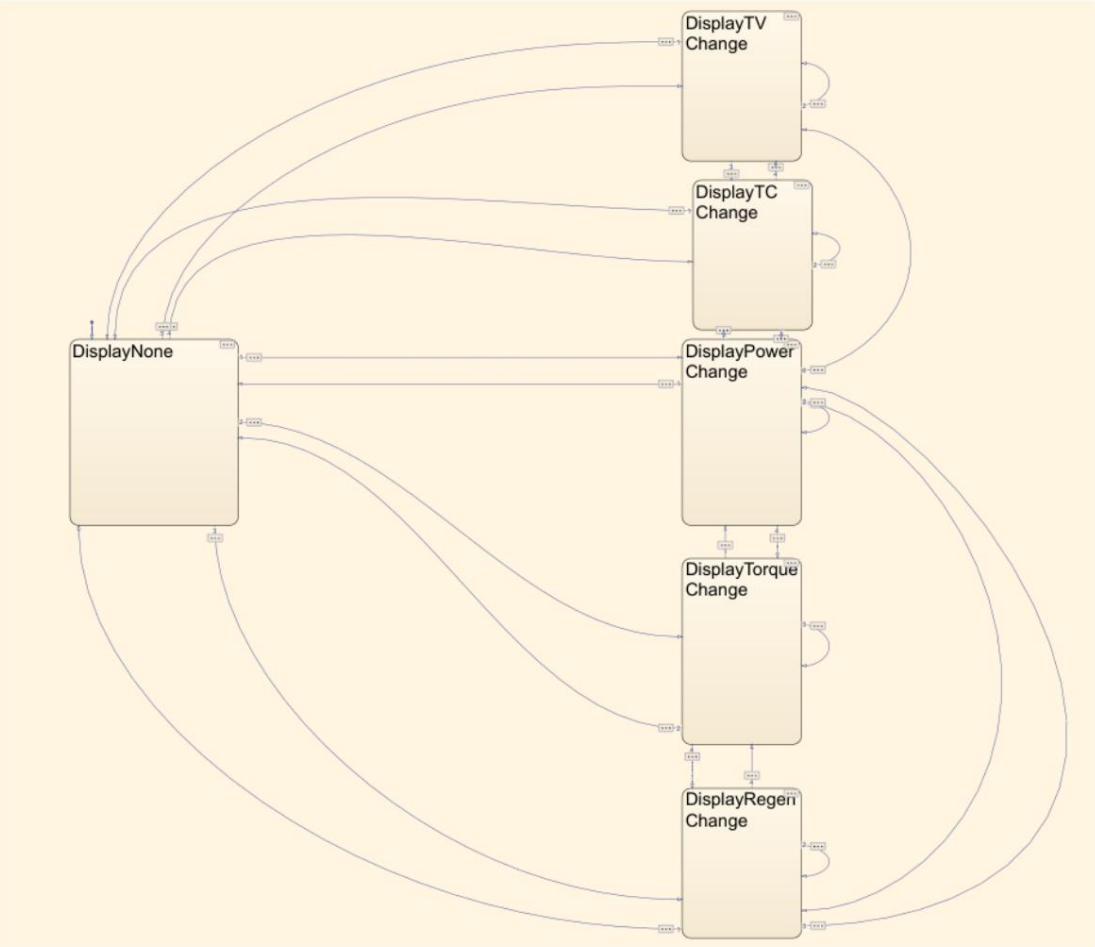

Figure 83 Inside the state machine for displaying popups

Displaying the ignition procedure

Since the ignition procedure is made up of several well-defined phases, some
of which require input from the pilot, such as throttle pressure
brake pedal and RTD button, I decided to display each single one on the screen
passage. This will allow me to understand, in case of problems, which of these
steps went wrong and to give the pilot suggestions on when he should give the input
and in what order.

In this case too, a state machine was used which receives as input the variable "VcuState" (uint8), which indicates the current state of the VCU. Also, as further input, the state machine receives the variable "ChargeState", which allows identify the contactor that closed to provide more details.

Stato	Nome	Descrizione
0	Fault	Uno dei compenenti della vettura o la VCU stessa ha avuto un fault e per questo motivo tutta la vettura viene portata in una condizione di sicurezza dove tutti i componenti vengono disabilitati.
1	Vehicle Off	Alta tensione disabilitata, il pilota non ha richiesto l'accensione del veicolo.
2	Shutdown	Si passa in questo stato durante lo spegnimento normale del veicolo. Vengono prima disabilitati gli inverter e successivamente quando la corrente uscente dalla batteria scende sotto una certa soglia vengono aperti i contattori.
3	Precharge	Si passa in questo stato quando il pilota avvia la procedura di accensione del veicolo. La batteria inizia la procedura di chiusura dei contattori e di precarica. Si passa allo stato successivo quando la batteria segnala la fine della procedura.
4	Hard Braking	In questo stato gli inverter sono abilitati ma la coppia comandata ad entrambi è uguale a zero perché il pilota ha schiacciato il freno con una pressione alta.
5	Wait Brake	Procedura di precarica terminata. Il pilota deve schiacciare il freno al fine di completare la procedura. Gli inverter sono disabilitati.
6	Wait Start	Procedura di precarica terminata, la batteria è collegata al powertrain. Il pilota sta schiacciando il freno e deve schiacciare il bottone di start per terminare la procedura. Gli inverter sono disabilitati.
7	Vehicle On	Procedura di accensione del veicolo terminata. Questo è l'unico stato dove viene può essere comandata una coppia diversa da zero.

Figure 84 VCU States

The results produced are represented by the boolean variable "DisplayProcedure", which indicates whether the description page should be shown or not, and from the variable uint8 *PageProcedure,* which indicates which page should be displayed.

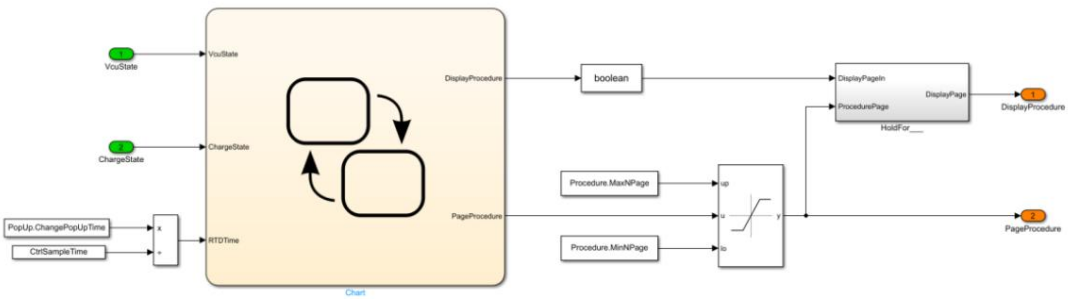

Figure 85 Inside the block for displaying the power-on procedure

The block called "HoldFor" is a function that allows you to keep a variable to a certain value for a certain period of time. In this case, the variable

DisplayProcedure is held high for up to milliseconds

indicated in the *PopUp.ProcedurePopUpTime constant.*

If the page to be displayed changes, the variable can return to true. This way,

If the power-on procedure is blocked, the page corresponding to the phase will not remain

fixed on the screen but will disappear.

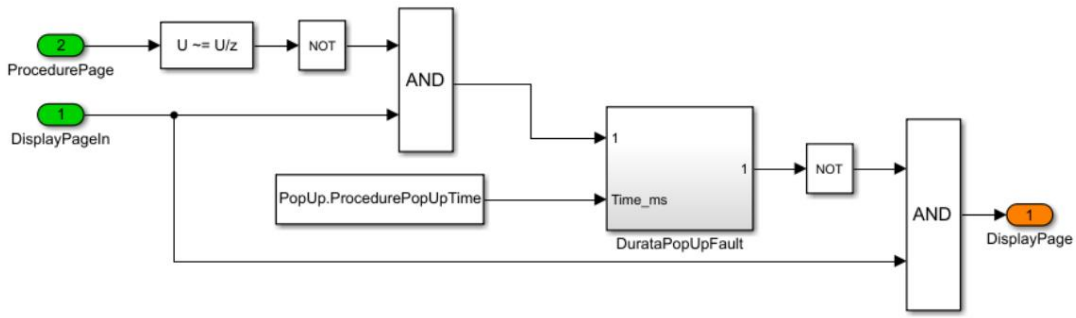

Figure 86 Inside the popup timing block

PageProcedure	Descrizione
0	Nessuna pagina visualizzata
1	Contattore del - chiuso
2	Contattore del Precharge chiuso
3	Contattore del + chiuso
4	Attesa del pedale del freno
5	Attesa dell'RTD button
6	RTD

Figure 87 Values assumed by the PageProcedure variable

LED Management

This block is responsible for determining which LED should be on or off.

LEDs always indicate the battery charge status, except during the charging procedure

ignition, where they instead represent the battery voltage. This is used to have

a visual indicator of the correct functioning of the pre-charge phase, in which the

tension of the traction system gradually increases until it reaches the

battery voltage.

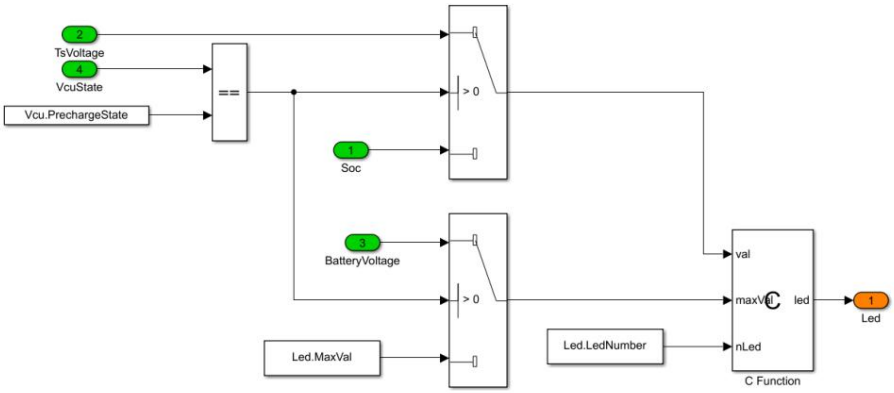

Figure 88 Inside the LED switching block

The output of this block is represented by the variable uint16 *Led,* whose first 10 bits indicate each individual LED. If the corresponding bit is 1, the LED will be on, otherwise it will be turned off.

The writing of individual bits is managed by a special block called “C function” which allows you to integrate an algorithm written in C (or C++) as a Simulink block.

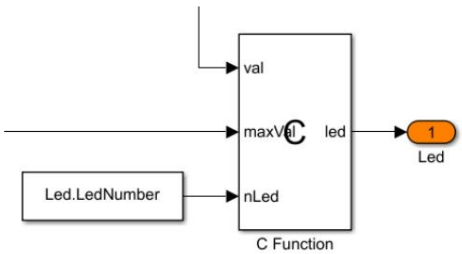

Figure 90 "C function" block

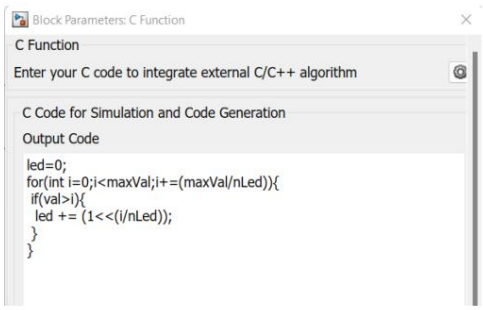

Figure 89 LED power-on algorithm

This simple algorithm takes as input the maximum value that can be represented by the LED and the number of LEDs present. In this way, it is possible to associate each individual LED a range of values. When the current value is in a specific interval, the corresponding bit will be set to 1, as will all the bits of the previous intervals.

The block is appropriately parameterized so that it is independent of the number of LEDs and the value they represent. This allows you to make future changes to the number of LEDs or the value represented without having to modify the entire block, but only the corresponding parameters.

7.3.3 Output Processing

Video output processing

The data to be printed on the screen are grouped into arrays of uint16 and divided by page. These arrays are parameterized based on their size and index of each data within them. This method allows to significantly reduce the amount of code required by the display libraries, as they only receive an array, its size, and an indication of which data is contained in which index. The result is a simple for loop that will write to the serial port. It is evident that This approach slightly increases the coupling between the two modules, but this This is not a problem, as the changes required in case of addition or Drastic page changes are minimal and very fast.

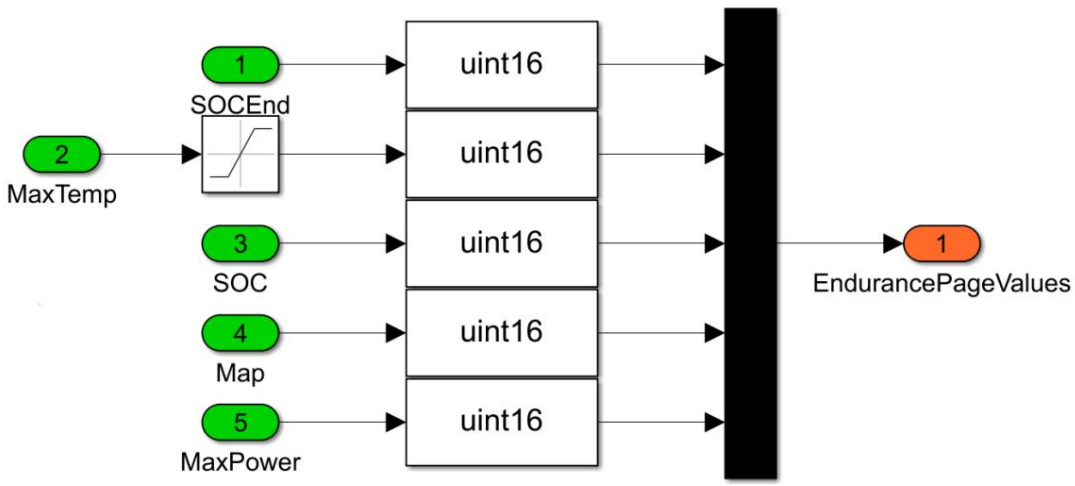

Figure 91 Generating an array containing the values of a single page

CAN bus output processing.

The only CAN message that is emitted by the steering wheel has the ID 0x401 and is intended to communicate the driver's inputs to the VCU. In fact, the steering wheel is only responsible for acquire the inputs and transmit them to the control unit that takes care of carrying out the changes. Once the changes have been made, the control unit will always communicates via the CAN bus to the steering wheel, which displays them on the screen.

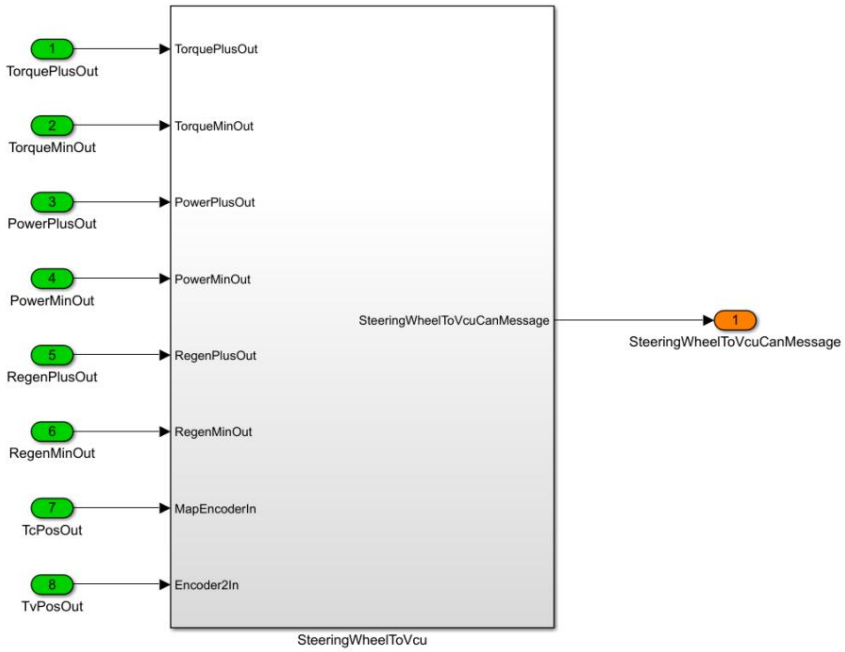

Figure 92 Packaging the SteeringWheelInfo message

Id	Dlc	Period	Name	Signal Name	Start Byte	Start Bit	Size in Bit	Type
0x400	8	100ms	SteeringInfo	TorquePlusButton	0	0	1	Boolean
				TorqueMinButton	0	1	1	Boolean
				PowerPlusButton	0	2	1	Boolean
				PowerMinButton	0	3	1	Boolean
				RegenPlusButton	0	4	1	Boolean
				RegenMinButton	0	5	1	Boolean
				TcPos	1	0	8	Unsigned Char
				TvPos	2	0	8	Unsigned Char

Figure 93 SteeringWheelInfo message structure

7.4 Display Libraries

The display libraries consist of the files “display.h” and “display.c”. In the first one there is no It contains only the functions implemented in display.c but also all those constants that are used to give commands to Nextion, will be analyzed in more details later.

As seen previously, the TIM10 takes care of calling with a frequency of 30Hz the updateDisplay function . This is the only function called from outside the library, in fact if changes are made to the graphic or data structure of the display the internal functions of the libraries will be modified without the other modules they realize it. Obviously these are in possession of a reference to the object of type **ExtY_SteeringWheel_T** which contains the strategy outputs.

7.4.1 Popup View

First in **updateDisplay** the boolean variables are checked which represent whether or not the display of the pop up is necessary. These have a priority. In fact, in case of conflict (more DisplayPopup variables set to true) it is followed the following order:

FAULT -> WARNING -> PROCEDURES -> PARAMETER_CHANGE

```
void updateDisplay(){
    updateSteeringWheelPage();
    if(SteeringWheel_Y.DisplayFaultPopUp){
        if(SteeringWheel_Y.BmsFaultValues > 0){
            displayFaultPopUp(0,SteeringWheel_Y.BmsFaultValues);
        }else if(SteeringWheel_Y.VcuFaultValues > 0 ){
            displayFaultPopUp(1,SteeringWheel_Y.VcuFaultValues);
        }
    }
    else if(SteeringWheel_Y.DisplayWarningPopUp){
        if(SteeringWheel_Y.BmsWarningValues > 0)
            displayWarningPopUp(0,SteeringWheel_Y.BmsWarningValues);
        else
            displayWarningPopUp(1,SteeringWheel_Y.VcuWarningValues);
    }else if(SteeringWheel_Y.DisplayProcedurePopUp){
        displayProcedurePage(SteeringWheel_Y.PageProcedure-1);
    }
    else if(SteeringWheel_Y.DisplayTorqueChangePopUp){
        displayChangePopUp(PCHANGE_PAGE_S,SteeringWheel_Y.MaxPower);
    }else if(SteeringWheel_Y.DisplayPowerChangePopUp){
        displayChangePopUp(TCHANGE_PAGE_S,SteeringWheel_Y.MaxTorque);
    }else if(SteeringWheel_Y.DisplayRegenChangePopUp){
        displayChangePopUp(RCHANGE_PAGE_S,SteeringWheel_Y.MaxRegen);
    }else if(SteeringWheel_Y.DisplayEncoder2ChangePopUp){
        displayTCChangePage(SteeringWheel_Y.TCEncoder);
    }else if(SteeringWheel_Y.DisplayMapChangePopUp){
        displayChangePopUp(MCHANGE_PAGE_S,SteeringWheel_Y.MapEncoder);
    }
```

Figure 94 updateDisplay function

Taking the **displayProcedurePage** function as an example , let's see how it happens more in detail the display of a Popup.

```
void displayProcedurePage(uint8_t pageProcedure){
    char buffer[BUFFER_LENGTH]={};
    sprintf(buffer,"%s.%s%s\"%c%c%c",INFO_PAGE_S,EDIT_PROCEDURE_STRING,PROCEDURE_STRING[pageProcedure],ch,ch,ch);
    HAL_UART_Transmit(&huart1,(uint8_t*) &buffer, strnlen(buffer,BUFFER_LENGTH),UART_TIMEOUT);
    if(SteeringWheelPage != INFO_PAGE_N){
        changeSteeringWheelPage(INFO_PAGE_S,INFO_PAGE_N);
    }
}
```

Figure 95 displayProcedurePage function

With **sprintf()** we write in the buffer the string that we will send via serial to the display. This is composed of:

```
const char *INFO_PAGE_S = "InfoPage";
```

Figure 96 INFO_PAGE_S string contents

```
const char *EDIT_PROCEDURE_STRING = "info.txt=\"";
```

Figure 97 Contents of the EDIT_PROCEDURE_STRING string

```
const char *PROCEDURE_STRING[] = {
        "NEG",
        "PRE",
        "POS",
        "BRK",
        "BTN",
        "RTD"
};
```

Figure 98 Contents of the PROCEDURE_STRING[] array

The index to select in the PROCEDURE_STRING array is dictated by the output of the SteeringWheel_Y.PageProcedure strategy . Described in chapter 7.3.2

```
char ch = 0xff;
```

Figure 99 End of string character

In unicode 0xFF corresponds to *ÿ.*

Based on the official NextionDisply documentation at the end of each command you have to add the character *ÿ* three times . This is necessary to make the user understand display that the command has finished and a new one is about to start.

For example, if the strategy establishes the display of the end-of-procedure page (RTD) the string that will be sent via serial will be the following:

InfoPage.info.txt="RTD" ÿ ÿ ÿ

By sending this string over serial, through the **HAL_UART_TRANSMIT function,** we are going to change the text displayed in the popup, it is important to note that the text editing is done before the page change, so that avoids displaying the previous text for a moment, this in addition to being aesthetically ugly and a potential distraction for the pilot.

7.4.2 Page change

The page change is done using the **changeSteringWheelPage function.**

```
void changeSteeringWheelPage(const char *page,uint8_t page_n){
    char pageBuffer[BUFFER_LENGTH]={};
    char responseBuffer[BUFFER_LENGTH]={};
    sprintf(pageBuffer,"page %s%c%c%c",page,ch,ch,ch);
    HAL_UART_Transmit(&huart1,(uint8_t*) &pageBuffer, strnlen(pageBuffer,BUFFER_LENGTH),UART_TIMEOUT);
    updateSteeringWheelPage();
}
```

Figure 100 Page change function

Following similar reasoning to those made previously, it will be sent on serial
the following string:

page InfoPage ÿ ÿ ÿ

7.4.3 Page Update

Every time the **updateDisplay** function is called or the page is changed
thanks to the **updateSteeringWheelPage** function the variable is updated
steeringWheelPage where the actual page viewed is tracked
that moment of time on the display. This can be done by sending the serial
sendme command which will return the current page number.

```
void updateSteeringWheelPage(){
    char requestBuffer[BUFFER_LENGTH]={};
    char responseBuffer[BUFFER_LENGTH]={};
    __HAL_UART_FLUSH_DRREGISTER(&huart1);
    sprintf(requestBuffer,"sendme%c%c%c",ch,ch,ch);
    HAL_UART_Receive(&huart1, (uint8_t*)&responseBuffer,BUFFER_LENGTH, UART_TIMEOUT);
    HAL_UART_Transmit(&huart1,(uint8_t*)&requestBuffer, strnlen(requestBuffer,BUFFER_LENGTH),UART_TIMEOUT);
    HAL_UART_Receive(&huart1, (uint8_t*)&responseBuffer,BUFFER_LENGTH, UART_TIMEOUT);
    uint8_t nPage = responseBuffer[responseIndex];
    if(nPage > 0 && nPage <= MAX_N_PAGE)
        SteeringWheelPage = nPage;
}
```

Figure 101 Function to get the current page displayed on the screen

7.4.4 Updating Page Values

Taking the Endurance page as an example, as already done for the strategy
Let's see how the values are updated.

Thanks to the *steeringWheelPage* variable I know which page it is
displayed so that you can update only the values of this one.

```
void updateEndurancePage(){
    for(int i=0;i<sizeof(SteeringWheel_Y.EndurancePageValues)/sizeof(SteeringWheel_Y.EndurancePageValues[0]);i++){
        updateDisplayValues(ENDURANCE_PAGE_STRING[i],SteeringWheel_Y.EndurancePageValues[i]);
    }
}
```

Figure 102 Endurance page data refresh function

By looping the output of the *SteeringWheel_Y.EndurancePageValues* strategy I can access the individual values and through the ENDURANCE_PAGE_STRING array I can access the keyword to use in the message to change the value specific.

```
void updateDisplayValues(const char* editString,uint16_t values){
    char buffer[BUFFER_LENGTH]={};
    sprintf(buffer, "%s%d%c%c%c",editString,values,ch,ch,ch);
    HAL_UART_Transmit(&huart1, (uint8_t*) &buffer, strnlen(buffer,BUFFER_LENGTH),UART_TIMEOUT);
}
```

Figure 103 Function to update data displayed on the screen

The **updateDisplayValues** function simply sends a change message via serial. If we assume that the value being updated is that of the state of charge (State Of Charge or SOC) the message sent is the following:

SOC.val=100 *ÿ ÿ ÿ*

7.4.5 Changing colors

Without going into the details of the individual functions to avoid repeating the same ones concepts seen previously let's just see which string needs to be sent on serial to change the color of an object x.

- String to change the background to red: *x.bco=RED ÿ ÿ ÿ*

- String to change the color of the characters to red: *x.pco=RED ÿ ÿ ÿ*

8 Conclusions

In this thesis work all the hardware development processes have been described and of the PSR02-S car steering wheel software.

One of the goals was to be able to give the pilot a tool to monitor the car's key parameters during races and tests, and the possibility to modify the settings through buttons and knobs in a manner simple, fast and precise. The goal was achieved by giving the possibility of change the power output of the battery pack, the torque, the torque regenerative and increase and decrease the invasiveness of traction control and torque vectoring.

One aspect that should not be underestimated was that of creating a software that was not only functional, but also easily understandable by the members of the Software & Vehicle Control and above all easily maintainable, so as to act as a point starting point for future developments. This goal was also achieved by using the same software structure that characterizes VCU and BMS. The firmware was written in C language. Data processing is done with Matlab/Simulink, from which the code to be integrated in the form of libraries into the firmware is then auto-generated. Thanks to this choice, even members who find themselves having to make changes for the first time on the project they will be able to orient themselves easily, given the already familiar structure.

The creation of a specific printed circuit board allowed us to optimize the space and reduce the weight of the entire steering wheel to about 400g, a crucial aspect in the field of car racing. Furthermore, this choice allowed the use of a microcontroller of the STM32F4 family already used on VCU and BMS, allowing thus the implementation of the software structure mentioned above.

The steering wheel described in this thesis was used during the Formula competition Student Netherlands, and during Formula Student Czech in which the Unipr Racing Team is

he placed in the top ten positions in a race for the first time in his history

foreign.

9 Acknowledgements

First of all I would like to thank my thesis supervisor, Professor Carlo Concari
who has always supported the Unipr Racing Team but above all cares about the kids who
they are part of it and have always been available and proactive towards us
even when he wasn't required to, he would have a place in my thanks
even if he hadn't been my thesis supervisor.

I thank my mother who has always taken care of me and supported me by giving me
the unconditional affection that I always carry inside me, I thank you above all for
having passed on to me the tenacity and hunger that pushes me to give my best in every
which I do. This little milestone is yours too.

I thank my father who never missed one of my football matches until I was
played. I thank him above all for every time he judged me objectively
pushing me to improve myself every day. This little milestone is yours too.

I thank my sister, my other half. The person whose advice is worth more to me than any other.
other. Even if I often don't show it, I love you with all my heart and I have the utmost
trust in your abilities. When you get there, remember your little brother.

I thank Professor Marinoni and Professor Camuso, whose teachings have helped me
transmitted the passion for IT and provided a preparation that could
face every programming exam with “the pipe”, without them I would never have
undertaken this path.

I would like to thank all the guys from the Unipr Racing Team for making the last two years of
unforgettable universities, one above all Davide Draghi who showed me that it is possible
be the best computer engineer in Italy while remaining humble. Mention
honorable to Pietro Canuti for being the best mechanical engineer in Italy but also
the funniest (I used funny because this is a university thesis, otherwise
I would have used another word).

I thank EVIL with whom I have always shared my free time, without you I would be
become a very serious person. I especially thank Fly for making me

company during the endless hours of study, and I thank Ghirlo for always raising the bar and push myself to always give something more.

I thank Tia for being close to me in the difficult moments of my life by putting he leaves his side just to help me.

I thank Chiara, who even though we took different paths, contributed to making me get my head straight and keep me on track.

Bibliography

Altium Designer - getting started. (sd). Taken from https://my.altium.com/altium-designer/getting-started

CAN in Automation. (sd). From www.can-cia.org: https://www.can-cia.org

Datasheet CAN transceiver SN65HVD230MDREP. (sd). Taken from https://www.ti.com/lit/ds/symlink/sn65hvd230m-ep.pdf?HQS=dis-mous-null-mousermode-dsf-pf-null-wwe&ts=1677516848529&ref_url=https%253A%252F%252Fwww.mouser.it%252F

Datasheet handcuffs. (sd). Treatment from https://www.mouser.it/datasheet/2/26/PT65%20Series%20Rotary%20code%202018-1314458.pdf

Datasheet MCP23017/ MCP23S17. (sd). Treatment from https://www.mouser.com/datasheet/2/268/20001952C-1920511.pdf

Datasheet VLCS5130. (sd). Treatment from https://www.mouser.it/datasheet/2/427/vlcs5130-1767005.pdf

Datasheets - NX4832T035. (sd). Treatment from nextion.tech: https://nextion.tech/datasheets/nx4832t035/

Instruction Set. (sd). From nextion.tech: https://nextion.tech/instruction-set/

mcp23017 open source libraries. (sd). Treatment from github.com: https://github.com/ruda/mcp23017

MathWork - Embedded coder. (sd). Treatment from https://it.mathworks.com/products/embedded-coder.html

MATLAB Coder. (sd). From https://it.mathworks.com/products/matlab-coder.html

PCB capabilities. (sd). From https://jlcpcb.com/capabilities/pcb-capabilities

Regulation Formula Student 2022. (sd). Treatment from https://formulastudent.de/fileadmin/user_upload/all/2022/rules/FS-Rules_2022_v0.9.pdf

STM32F412RE. (sd). Treatment from www.st.com: https://www.st.com/en/microcontrollers-microprocessors/stm32f412re.html