



**UNIVERSITÀ
DI PARMA**

Dipartimento di Ingegneria e Architettura,
Corso di Laurea in Ingegneria Informatica Elettronica e
delle Telecomunicazioni

Progettazione e sviluppo di un volante per una vettura da competizione Formula Student

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

Relatore:
Chiar.mo Prof.
Carlo Concari

Laureando:
Michele Bandini

Anno Accademico 2021-2022

Sommario

1 Introduzione	10
1.1 UniPR Racing Team	10
1.2 La Formula Student.....	10
1.2.1 Ispezioni tecniche.....	11
1.2.2 Prove dinamiche	12
1.2.3 Prove statiche	15
1.2 Stato sull'arte sui volanti per vetture da competizione	16
1.2.1 Anni '50 e '60	17
1.2.2 Anni '70.....	17
1.2.3 Anni '80.....	18
1.2.4 Anni '90 / '00	19
1.2.5 Anni '10.....	20
1.2.6 Il display.....	20
1.2.7 I pulsanti	20
1.2.8 Il volante della PSR02-s	22
3 Analisi dei requisiti	27
4 Interfaccia pilota.....	29
4.1 Pulsanti.....	29
4.1.1 Regolazione potenza	29
4.1.2 Regolazione coppia motrice	30
4.1.3 Regolazione frenata rigenerativa.....	30
4.1.4 Ready to drive.....	30
4.2 Switch rotativi.....	31

4.2.1 Traction control	31
4.2.2 Torque Vectoring	31
4.3 LED	32
4.4 Display	32
4.4.1 Pagina generale.....	32
4.4.2 Pagina Endurance	33
4.4.3 Pagina Acceleration.....	34
4.4.5 Pagina Brake Test.....	35
4.4.6 Pagina Fault	35
4.4.7 Pagina sensoristica.....	36
4.4.8 Popup cambio parametro	37
5 Architettura hardware	38
5.1 Introduzione	38
5.2 Componenti hardware coinvolti.....	38
5.3 Comunicazione Can Bus	39
5.3.1 Formato del bus Frame	40
5.4 Comunicazione seriale	40
6 Progettazione del circuito stampato	42
6.1 Altium Designer	42
6.2 Schema elettrico.....	43
6.2.1 SteeringWheelIO.SchDoc	43
6.2.2 Microcontrollore.SchDoc	47
6.3 Design PCB.....	50
6.3.1 Forma del PCB.....	51
6.3.1 Posizionamento dei componenti	51

6.4 Integrazione dello schermo.....	53
7 Progettazione del software.....	54
7.1 Introduzione alla progettazione del software.....	54
7.1.1 Modello a V.....	54
7.1.2 Sistema a basso accoppiamento.....	55
7.1.3 Sviluppo del software con Matlab/Simulink	56
7.1.4 Autogenerazione del codice	57
7.2 Firmware	58
7.2.1 Configurazione periferica CAN	59
7.2.2 Configurazione periferica USART	60
7.2.3 Input digitali.....	61
7.2.4 Impostazioni Timer	61
7.2.4 Struttura del firmware	63
7.3 Strategia	66
7.3.1 Elaborazione input	66
7.3.2 Generazione degli output.....	69
7.3.3 Elaborazione degli output	75
7.4 Librerie display.....	76
7.4.1 Visualizzazione Popup	77
7.4.2 Cambio pagina	79
7.4.3 Aggiornamento pagina	79
7.4.4 Aggiornamento valori pagina	79
7.4.5 Cambio dei colori	80
8 Conclusioni	81
9 Ringraziamenti.....	83

Bibliografia	85
--------------------	----

Figura 1 PSR02-S.....	10
Figura 2 Punteggio disponibile in una gara di Formula Student.....	11
Figura 3 Punteggio Skidpad	13
Figura 4 Tracciato Skidpad	13
Figura 5 Punteggio Acceleration	13
Figura 6 Punteggio Autocross	14
Figura 7 Punteggio Endurance	14
Figura 8 Punteggio Efficency.....	15
Figura 9 Punteggio Engineering Design.....	15
Figura 10 Punteggio Business Plan	16
Figura 11 Punteggio Cost and Manufacturing	16
Figura 12 Volante McLaren M7C.....	17
Figura 13 Volante McLaren M23.....	17
Figura 14 Volante MP4/4.....	18
Figura 15 McLaren MP4-23.....	19
Figura 16 McLaren MP4-14.....	19
Figura 17 Volante Williams F1	21
Figura 18 Regola T2.7.8 regolamento Formula Student	23
Figura 19 Posizionamento del volante	23
Figura 20 Regola T2.6 regolamento Formula Student	23
Figura 21 Regolamentazione pulsanti di shutdown all'interno del cockpit.	24
Figura 22 Regola T2.7.7 Regolamento Formula Student.....	24
Figura 23 Dimensioni volante PSR02-S.....	25
Figura 24 regola T2.7.10 regolamento Formula Student	25
Figura 25 regola T2.7.5 regolamento Formula Student	25
Figura 26 Sistema di sgancio rapido	26
Figura 27 regola T4.11.1, regolamento Formula Student	26
Figura 28 Volante PSR02-S.....	29
Figura 29 Pulsante Ready To Drive.....	30
Figura 30 Illuminazione LED.....	32

Figura 31 Pagina generale.....	32
Figura 32 Pagina Endurance.....	33
Figura 33 Pagina Acceleration.....	34
Figura 34 Pagina Brake Test	35
Figura 35 Pagina Fault VCU	36
Figura 36 Pagina Fault BMS	36
Figura 37 Popup Fault.....	36
Figura 38 Pagina sensoristica	36
Figura 39 Popup cambio parametro	37
Figura 40 Architettura hardware	38
Figura 41 Micrcontrollore	38
Figura 42 Nextion Display	39
Figura 43 Frame protocollo CAN	40
Figura 44 Altium Designer	43
Figura 45 Schema connettore volante	43
Figura 46 Schema elettrico switch rotativi	44
Figura 47 Codifica BCD manettini.....	45
Figura 48 Schema elettrico pulsante.....	45
Figura 49 Schema LED	46
Figura 50 Connnettore dashboard	47
Figura 51 Convertitore LDO	47
Figura 52 Schema elettrico tranceiver CAN bus	48
Figura 53 Schema elettrico espansore I/O	49
Figura 54 Schema elettrico Microcontrollore	50
Figura 55 Fori di ancoraggio del PCB	51
Figura 56 Posizionamento Manettini	51
Figura 57 PCB volante PSR02-S	52
Figura 58 Connnettore lato display.....	53
Figura 59 Diagramma modello a V	54
Figura 60 Blocco DisplayFault	57

Figura 61 Interno del blocco DisplayFault	57
Figura 62 Screenshot codice autogenerato	58
Figura 63 Input Output microcontrollore	59
Figura 64 Configurazione BaudRate CAN bus	59
Figura 65 Configurazione intterrupt messaggi CAN	60
Figura 66 Configurazione BaudRate USART	61
Figura 67 Funzioni per impostare la frequenza dei timer	62
Figura 68 Porzione di codice richiamata all'esaurimento di un timer	63
Figura 69 Funzione Main del firmware.....	64
Figura 70 Funzione di SetUp del firmware	64
Figura 71 Routine principale del firmware	65
Figura 72 Funzione che acquisce gli input del pilota.....	65
Figura 73 Funzione di lettura della posizione dei manettini	65
Figura 74 Struttura strategia.....	66
Figura 75 Input pulsanti non filtrati	67
Figura 76 Input pulsanti con filtro anti-rimbalzo	67
Figura 77 Input manettini non filtrati.....	68
Figura 78 Blocco di scomposizione messaggio CAN.....	68
Figura 79 Interno del blocco di scomposizione di un messaggio CAN	69
Figura 80 Blocco per settare la variabile DisplayFaultPopUp e DisplayWarningPopUp	70
Figura 81 Interno del blocco di Figura 80	70
Figura 82 Macchina a stati visualizzazione popup	70
Figura 83 Interno della macchina a stati per la visualizzazione dei popup	71
Figura 84 Stati VCU	72
Figura 85 Interno del blocco per la visualizzazione della procedura di accensione ..	72
Figura 86 Interno del blocco di timing visualizzazione popup	73
Figura 87 Valori assunti dalla variabile PageProcedure	73
Figura 88 Interno del blocco per l'accensione dei led.....	74
Figura 89 Algoritmo accensione LED	74

Figura 90 blocco "C function"	74
Figura 91 Generazione array contenente i valori di una singola pagina.....	75
Figura 92 Impacchettamento del messaggio SteeringWheelInfo.....	76
Figura 93 Struttura messaggio SteeringWheelInfo.....	76
Figura 94 Funzione updateDisplay	77
Figura 95 Funzione displayProcedurePage.....	77
Figura 96 Contenuto stringa INFO_PAGE_S.....	78
Figura 97 Contenuto della stringa EDIT_PROCEDURE_STRING	78
Figura 98 Contenuto dell'array PROCEDURE_STRING[]	78
Figura 99 Carattere fine stringa	78
Figura 100 Funzione per il cambio di pagina	79
Figura 101 Funzione per ottenere la pagina corrente visualizzata a schermo.....	79
Figura 102 Funzione per l'aggiornamento dei dati della pagina Endurance	80
Figura 103 Funzione per aggiornare un dato visualizzato a schermo.....	80

1 Introduzione

1.1 UniPR Racing Team

L'UniPR Racing Team è un progetto universitario, sostenuto dall'Università degli studi di Parma, nato nel 2007 da un gruppo di studenti di ingegneria meccanica. Obiettivo primario del progetto è quello di progettare, sviluppare e fabbricare in completa autonomia una monoposto da competizione che possa partecipare alla competizione internazionale di Formula Student.

Dotate di un motore a combustione interna, le prime vetture realizzate dall'UniPR Racing Team partecipano alla categoria *Combustion*. Gli ottimi risultati e una maggior consapevolezza del lavoro hanno portato il Team ad abbandonare definitivamente motore per prendere l'importante decisione di entrare nel mondo della categoria *Electric*. Prima vettura equipaggiata di un motore elettrico è la PSR01, sviluppata nel 2018 e poi sostituita dalla PSR02, monoposto che ha portato uno dei migliori risultati nella storia dell'UniPr Racing



Figura 1 PSR02-S

Team: 5° posto nell'overall ranking alla tappa italiana FSAE Italy 2021 a Varano de' Melegari. Nel 2022 viene intrapreso un percorso di sviluppo e ottimizzazione culminato nell'attuale monoposto: la PSR02-S. Terza vettura elettrica dell'UniPR Racing Team e diretta evoluzione del prototipo 2021, la PSR02-S partecipa a due gare estere: Formula Student Netherlands, una delle gare più prestigiose e impegnative per competitività; Formula Student Czech Republic dove il Team ottiene la prima top 10 estera.

1.2 La Formula Student

Formula Student è un campionato di progettazione studentesca che coinvolge i dipartimenti di ingegneria delle università di tutto il mondo organizzata dalla SAE

International (Society of Automotive Engineers). Tutti i Team sono tenuti ad attenersi ad un preciso regolamento, emanato ogni anno dall'ente organizzatore, e che intende lasciare libera interpretazione ai progettisti vincolandoli, però, con severe restrizioni al fine di garantire la sicurezza dei partecipanti.

Gli eventi si suddividono in tre macro-gruppi: Ispezioni Tecniche, necessarie per poter accedere alle Prove Dinamiche, e Prove Statiche. Ogni macro-gruppo contiene al proprio interno diverse prove che permettono di valutare la monoposto sia in termini di progettazione che di competitività attraverso punteggi stabiliti da regolamento:

	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

Figura 2 Punteggio disponibile in una gara di Formula Student

Fondamentale ai fini della competizione è sapere che veicolo può essere utilizzato solo per un anno, a partire dal primo giorno in loco della sua prima gara. Come citato dal regolamento, regola A2.2.2: *"Per essere classificato come nuovo, un veicolo deve avere come minimo un telaio di nuova fabbricazione con cambiamenti significativi nella struttura primaria rispetto al suo predecessore"*.

1.2.1 Ispezioni tecniche

Le ispezioni tecniche si dividono nelle seguenti parti:

- [EV ONLY] **Accumulator Inspection**, ad essere controllato è tutta la progettazione, funzionalità e sicurezza del pacchetto batteria comprensivo di caricatore che sarà ispezionato e sigillato, e il set di strumenti di base utilizzati.
- [EV ONLY] **Electrical Inspection**, viene misurata la resistenza di isolamento tra la massa del TS (Tractive System) e del LVS (Low Voltage System) e, affinché il test sia superato, la resistenza di isolamento misurata deve essere di almeno $500 \Omega/V$ in relazione alla tensione TS massima del veicolo. Inoltre, viene testato l'IMD (Insulation Monitoring Device) che deve spegnere il TS entro 30s con una resistenza di guasto del 50% inferiore al valore di risposta.
- **Mechanical Inspection**, in cui vengono verificati progettazione, sviluppo e messa in sicurezza di powertrain e telaio seguendo tutti i requisiti tecnici richiesti dalla sezione T 1-13 del regolamento FSG.
- **Tilt Test**, in cui il veicolo, con al suo interno il pilota, viene posizionato su di un tavolo inclinabile con un angolo di 60° . Per superarlo non devono esserci perdite di liquido e tutte le ruote devono rimanere a contatto con la superficie del piano.
- **Vehicle Weighing**, in condizione ready-to-race e con tutti i fluidi al massimo livello di riempimento.
- [EV ONLY] **Rain Test** in condizioni ready-to-race e con il TS attivo. L'acqua viene spruzzata sul veicolo da qualsiasi direzione possibile. Il test è superato se l'IMD non si attiva mentre l'acqua viene spruzzata sul veicolo per 120 secondi dopo che lo spruzzo d'acqua è cessato.
- Brake Test per cui tutte e quattro le ruote risultano bloccate alla fine di un rettilineo di accelerazione. In 11.1.2 [EV ONLY] “*Dopo l'accelerazione, il sistema di trazione deve essere spento dal pilota e il pilota deve frenare usando solo i freni meccanici*”.

1.2.2 Prove dinamiche

Superate con successo tutte le ispezioni tecniche, la monoposto affronta quattro prove dinamiche in cui viene valutata la competitività, l'affidabilità e l'efficienza:

- **Skidpad:** Circuito composto da due cerchi concentrici disposti a otto, come mostrato in figura 4, e in cui vengono cronometrati di seguito due giri completi.

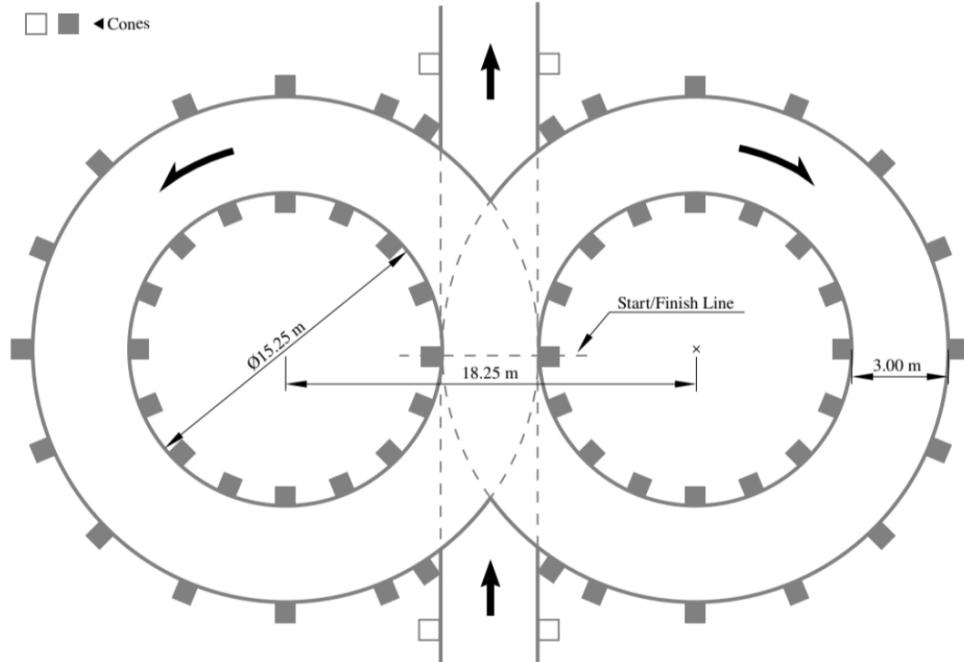


Figura 4 Tracciato Skidpad

$$M_{SKIDPAD_SCORE} = 46.5 \left(\frac{\left(\frac{T_{\max}}{T_{\text{team}}} \right)^2 - 1}{0.5625} \right)$$

Figura 3 Punteggio Skidpad

T_{team} = miglior tempo realizzato dal proprio team incluse le penalità.

T_{\max} = miglior tempo assoluto moltiplicato per 1.5 incluse le penalità.

- **Acceleration:** prova di accelerazione su rettilineo di 75m e largo almeno 3 m.

$$M_{ACCELERATION_SCORE} = \left(\frac{\frac{T_{\max}}{T_{\text{team}}} - 1}{0.5} \right)$$

Figura 5 Punteggio Acceleration

T_{team} = miglior tempo realizzato dal proprio team incluse le penalità.

T_{\max} = miglior tempo assoluto moltiplicato per 1.5 incluse le penalità.

- **Autocross:** un giro di pista cronometrato. Pista che deve rispettare i seguenti parametri (D6.1.1)
 - Rettilinei: non più lunghi di 80 m
 - Curve: fino a 50 m di diametro
 - Curve a gomito: Diametro esterno minimo di 9 m (della curva)
 - Slalom: coni in linea retta a una distanza compresa tra 7,5 e 12 m.
 - Varie: Chicane, curve multiple, curve a raggio decrescente, ecc. La larghezza minima della larghezza minima della pista è di 3 m.

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

Figura 6 Punteggio Autocross

T_{team} = miglior tempo realizzato dal proprio team incluse le penalità.

T_{\max} = miglior tempo assoluto moltiplicato per 1.5 incluse le penalità.

- **Endurance & Efficiency:** stesso circuito della prova di Autocross su vengono svolti 22km, a metà dei quali avviene il cambio piloti con tempo massimo 3min. Per valutare l'Efficiency sono presi in considerazione per l'assegnazione di un punteggio solo i veicoli che soddisfano i seguenti requisiti:
 - il veicolo ha ricevuto punti per la gara di endurance.
 - il tempo di endurance non corretto non supera 1,333 volte il tempo di endurance non corretto del veicolo più veloce.

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

Figura 7 Punteggio Endurance

[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 EV4.6. 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

Figura 8 Punteggio Efficiency

1.2.3 Prove statiche

Durante le prove statiche vengono valutate tre diverse parti:

- **Engineering Design**, ovvero vengono presentate a un gruppo di giudici tutte le scelte ingegneristiche applicate alla monoposto, sempre nel rispetto dell'obiettivo e del regolamento della competizione.

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

Figura 9 Punteggio Engineering Design

- **Business Plan Presentation:**

- S 1.1.1 “L’obiettivo del BPP è quello di valutare la capacità del team di sviluppare e fornire un modello aziendale completo che dimostri che il loro prodotto - un prototipo di auto da corsa - potrebbe diventare un’opportunità commerciale remunerativa che crea un profitto monetario”.
- S 1.1.2 “I giudici devono essere trattati come se fossero potenziali investitori o partner del modello di business presentato”.

- S 1.1.3 “Il business plan deve essere riferito allo specifico prototipo”.

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

$$BPP_SCORE = 70 \left(\frac{P_{team}}{P_{max}} \right)$$

Figura 10 Punteggio Business Plan

P_{team} = punteggio ottenuto dalla propria squadra.

P_{max} = punteggio più alto assegnato a qualsiasi squadra che non partecipa alle finali.

- **Cost and Manufacturing**, il cui obiettivo è valutare la comprensione da parte del team dei processi di produzione e dei costi associati alla costruzione di un prototipo di auto da corsa. Questo include decisioni di compromesso tra contenuto e costo, decisioni di acquisto o di vendita e la comprensione delle differenze tra prototipo e produzione di massa.

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

Figura 11 Punteggio Cost and Manufacturing

1.2 Stato sull'arte sui volanti per vetture da competizione

Prendendo come riferimento il campionato *open wheel* per eccellenza, ovvero la F1, vediamo come sono cambiati i volanti nel corso degli anni, adattandosi nel tempo a quelli che erano le richieste da parte dei piloti.

1.2.1 Anni '50 e '60

Negli anni Cinquanta erano fondamentalmente ispirati a semplici auto da strada. Di dimensioni tutt'altro che ridotte, alcuni raggiungevano oltre 40cm di diametro, la ruota interna era realizzata in alluminio mentre quella esterna in legno, problematica perché causava la comparsa di schegge nelle mani dei piloti durante la guida. Di fatto, le auto degli anni '50 non avevano il servosterzo, quindi il grande diametro era necessario per il pilota per ottenere la trazione necessaria a far andare la vettura nella direzione corretta.

Negli anni '60 la vettura inizia ad avere un aspetto diverso: il motore viene spostato dietro il pilota modificando l'assetto della scocca e rendendo la parte anteriore molto più snella. Anche la posizione del pilota si è ridotta ulteriormente all'interno dell'auto riducendo di conseguenza lo spazio per il volante.



Figura 12 Volante McLaren M7C

1.2.2 Anni '70

Negli anni '70 i costruttori iniziarono a pensare al volante come a un modo per ottenere prestazioni oltre che a un modo per far girare le ruote anteriori. Nella seconda metà del decennio vediamo l'introduzione dell'interruttore di emergenza, utile nel caso in cui la valvola a farfalla si fosse bloccata in posizione aperta. Questo nuovo meccanismo consentiva al conducente di spegnere rapidamente il motore prima che entrasse in contatto con le barriere.



Figura 13 Volante McLaren M23

1.2.3 Anni '80

Il design del volante fece un ulteriore passo avanti negli anni '80. Primo cambiamento è l'utilizzo della pelle scamosciata intorno al cerchio. Stefan Johansson, pilota per la McLaren e Ferrari, lo ha definito il più grande miglioramento apportato durante la sua permanenza nello sport. La pelle scamosciata, più aderente, gli ha permesso di sentire molto meglio il volante, portando poi i piloti a richiedere cerchi su misura per il loro stile di guida.



Figura 14 Volante MP4/4

Nella seconda metà del decennio il volante inizia ad assumere ulteriori responsabilità. Viene aggiunto un meccanismo di sgancio rapido che consente al pilota di rimuovere il volante rapidamente in caso di emergenza. Altra novità, un pulsante in grado di gestire le comunicazioni radio tra piloti e muretto. Alcune scuderie avevano addirittura un pulsante di sovrallimentazione che arricchiva

la miscela di carburante per favorire i sorpassi. Tuttavia, l'evoluzione di gran lunga maggiore è stata l'introduzione del cambio a palette. Il progettista Bernard ha avuto l'idea di eliminare il pedale della frizione e di introdurre due paddle sul retro del volante per consentire al guidatore di cambiare marcia. Ciò consente di risparmiare spazio, peso e la necessità di togliere le mani dal volante per cambiare le marce, oltre a essere molto più veloce una volta superati i problemi di gioventù della tecnologia. Dopo i primi problemi di affidabilità, all'inizio degli anni '90 tutti i team passarono al cambio a palette. L'umile volante aveva finalmente avuto la sua prima grande evoluzione, ma l'introduzione generalizzata del cambio a palette fu solo l'inizio della rivoluzione del volante. Negli anni Novanta l'elettronica ha preso il sopravvento sulla F1 e questo ha accelerato drasticamente lo sviluppo della tecnologia.

1.2.4 Anni '90 / '00

Dopo 40 anni di metallo, il volante è stato finalmente rinnovato e, come il resto dell'auto, è stato realizzato in fibra di carbonio. In questo modo il peso è stato immediatamente ridotto. Anche la forma e le dimensioni del volante hanno subito un cambiamento: dalle tradizionali specifiche rotonde si è passati a un appiattimento generale e alla rimozione delle sezioni superiore e inferiore del volante.

Nonostante i volanti siano diventati più piccoli, le loro funzionalità sono aumentate. I pulsanti degli anni '80 sono rimasti, ma sono stati affiancati da altre funzioni, come il limitatore di velocità in corsia dei box e la retromarcia. A questi pulsanti si aggiunsero anche i quadranti rotanti. Negli anni '90 i piloti potevano selezionare diverse impostazioni di motore e di controllo della trazione, consentendo loro di modificare i parametri chiave durante la guida.

La fine del secolo porta ad una rivoluzione più in termini di materiali che di tecnologia: pelle scamosciata è stata sostituita dalla gomma, modellata in base alle forme delle mani del pilota così da fornire sensazione e feedback ottimali. Il numero di pulsanti e manopole aumenta, il pilota è ora in grado di apportare modifiche rapide per perfezionare il bilanciamento e le prestazioni dell'auto. Nel corso di un gran premio poteva regolare impostazioni come il differenziale per l'ingresso e la parte centrale della curva. Le impostazioni di frenata del motore e di coppia, per citarne alcune, consentono al pilota di trovare il punto di forza del proprio assetto.



Figura 16 McLaren MP4-14



Figura 15 McLaren MP4-23

1.2.5 Anni '10

Come gli anni Novanta e gli anni Duemila, anche il decennio successivo è all'insegna dell'evoluzione. Il più grande cambiamento avviene con l'introduzione dei motori turbo-ibridi. Nel 2014, a conferma della complessità dell'epoca, sul volante appare un grande display LCD che consente al pilota di sfogliare molteplici menu di dati per prendere decisioni sulle modalità del motore o sulle funzioni di recupero dell'energia.

Di fondamentale importanza è il fatto che ogni pilota decide la configurazione del volante come meglio crede. Ciò include non solo la disposizione e la tipologia di pulsanti, switch, e rotelle, ma anche l'organizzazione delle informazioni che vengono mostrate sul display di bordo. Se da un lato questo approccio facilita l'utilizzo da parte dei piloti, permette di risparmiare peso, in quanto il numero di interruttori fisici era ridotto.

1.2.6 Il display

Oggi molti team utilizzano il display integrato al volante, ma ci sono sempre delle variazioni. Williams, ad esempio, ha mantenuto il display LCD fissato al telaio che, oltre ad essere più economico, risulta essere più leggero e più maneggevole in termini di massa rotazionale del volante. Il lato negativo di questa scelta è che il display viene completamente oscurato dal volante stesso durante le curve, e ciò limita drasticamente le occasioni che il pilota ha per usufruire delle importantissime informazioni mostrate nel piccolo schermo.

1.2.7 I pulsanti

Dopo questa breve panoramica storica, arriviamo a capire a che cosa servono tutti i pulsanti presenti sul volante. È chiaramente improbabile arrivare a conoscere l'uso di ogni singolo tasto presente, solo per il fatto che le scuderie tendono a nascondere parti di informazioni, ma possiamo analizzare quelli visibili dalle immagini rese note. Per fare questo, prenderemo ad esempio il volante della Williams FW43 del 2020.

Innanzitutto, dividiamo i pulsanti presenti su di un volante di F1 in

- **Pulsanti** veri e propri (attivano/disattivano una funzionalità),

- **Rotelle** di regolazione (presentano solitamente numeri da 1 a 10 in maniera progressiva)
- **Manettini** (normalmente poste nel centro del volante, di solito si riferiscono alle impostazioni più importanti della monoposto, quelle della Power Unit)



Figura 17 Volante Williams F1

Una volta precisato questo, analizziamo la figura 17 da sinistra verso destra:

ENTRY: (rotella): Modifica il differenziale all'entrata delle curve;

N/R: Seleziona la marcia Neutral (folle) o Reverse (retromarcia);

Meno: diminuisce il valore dell'impostazione mostrata a schermo

OK: Dà conferma di ricezione della comunicazione ai box;

BB-R: Aumenta il bilanciamento dei freni verso il posteriore;

PC: Dà comunicazione ai box di effettuare una sosta al prossimo giro

TRQ (rotella): Modifica il valore della coppia della Power Unit;

Manopola Multifunzione (verde): varie impostazioni del chassis ed elettroniche;

Manopola MODE: Modalità di erogazione della Power Unit;

Manopola GO: Modalità del propulsore e dell'elettronica;

Manopola TYRE: tipo di pneumatico montato e fase di consumo;

Manopola Multifunzione (blu): varie impostazioni del chassis ed elettroniche;

HPP: Apre il Menù della Power Unit Mercedes;

BB F: Aumenta il bilanciamento dei freni verso l'anteriore;

RAD: apre la comunicazione con il muretto dei box;

Più: aumenta il valore dell'impostazione mostrata a schermo;

LIM: attiva il limitatore per la corsia dei box;

B-Mig (rotella): Modifica come cambia il bilanciamento dei freni dall'anteriore al posteriore durante la frenata;

EXIT (rotella): Modifica il differenziale all'uscita delle curve.

Nell'immagine si nota anche una delle tante schermate che il display di bordo può avere. In questo caso:

PIT: segnala l'inserimento del Limitatore dei box

DAT: segnala l'attivazione della modalità di download di dati

0: Indica il numero dei giri del motore (spento, in questo caso)

N13 Tyre Prime: il tipo di pneumatico montato al momento

BBal: impostazione del bilanciamento dei freni dall'anteriore al posteriore

BMig: impostazione corrente della migrazione del bilanciamento

TRQ: impostazione corrente della coppia

Diff Ent: impostazione corrente del differenziale in entrata

Diff Mid: impostazione corrente del differenziale all'apice della curva

Diff Exit: impostazione corrente del differenziale in uscita

1.2.8 Il volante della PSR02-s

La struttura e il posizionamento del volante all'interno del cockpit devono seguire le indicazioni fornite dal *FS Rules Book Germany* pubblicato ogni anno in vista delle competizioni. Per questa tesi, si prenderà a modello il regolamento della stagione 2022.

Per quanto concerne il posizionamento del volante, è necessario considerare le regole T2.7.6 e T2.7.8 per cui:

- 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.

Figura 20 Regola T2.6 regolamento Formula Student

- 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.

Figura 18 Regola T2.7.8 regolamento Formula Student

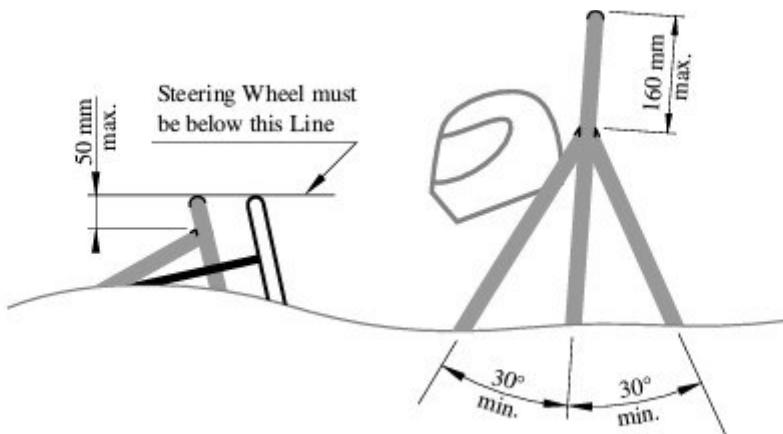


Figura 19 Posizionamento del volante

Questo è indispensabile in quanto il pilota deve avere un'adeguata visibilità della parte anteriore e laterale del veicolo. Seduto in posizione di guida normale deve obbligatoriamente avere un campo visivo minimo di 100° su entrambi i lati (T4.10.01). Inoltre, bisogna ricordare che il gioco consentito del sistema di sterzo è limitato a un totale di 7° misurati sul volante. Posto lateralmente al volante, senza essersene ostacolato, e montato direttamente al telaio all'interno del cockpit, troviamo il bottone dello shutdown con le seguenti caratteristiche:

- T11.4.4 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
- T11.4.5 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.
- T11.4.6 Shutdown buttons must be rigidly mounted to the vehicle and must not be removed during maintenance.

Figura 21 Regolamentazione pulsanti di shutdown all'interno del cockpit.

Una volta stabilito il corretto inserimento del volante all'interno del cockpit, andiamo ad analizzare la struttura del volante mentre input e schermo verranno analizzati nel capitolo dell'interfaccia pilota.

Prenderemo in considerazione tre componenti necessarie per questa breve analisi: materiali, dimensioni e componenti meccaniche.

Secondo quanto riportato dalla regola T2.7.7

- T2.7.7 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.

Figura 22 Regola T2.7.7 Regolamento Formula Student

La base del volante risulta essere rigida ma allo stesso tempo estremamente leggera grazie alla struttura a sandwich in fibra di carbonio le cui pelli vengono separate tra loro da un honeycomb, una tipologia di core alveolare realizzato in fibra aramidica (Nomex). A questa struttura portante viene applicata su entrambe i lati l'impugnatura realizzata tramite materiale ABS con stampa 3D. Assemblati avremo una lunghezza totale di 232mm per un'altezza di 157mm. Sempre in materiale ABS e realizzati tramite stampa 3D, sono presenti due switch di regolazione del set-up di guida come da figura Figura 23.

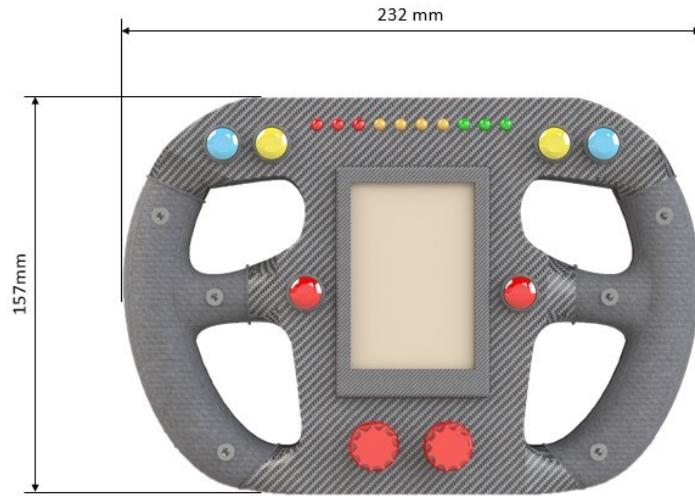


Figura 23 Dimensioni volante PSR02-S

Passando al retro, bisogna considerare che, come da regolamento, il volante deve essere collegato al piantone dello sterzo tramite un sistema di sgancio rapido fissato alla struttura tramite tre viti M4.

- 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.

Figura 25 regola T2.7.5 regolamento Formula Student

- 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.

Figura 24 regola T2.7.10 regolamento Formula Student

Lo sgancio rapido è pensato per motivi di sicurezza nel momento in cui il pilota si dovesse trovare in condizione di pericolo, e quindi di avere la necessità di uscire il più rapidamente possibile dalla vettura. Nelle competizioni di Formula Student, questo



Figura 26 Sistema di sgancio rapido

processo viene verificato tramite una prova di egress per cui il pilota, vestito con tutto l'equipaggiamento necessario e in posizione di guida, deve necessariamente essere in grado di uscire dal cockpit in un massimo di 5s come riportato dal capitolo T4 del regolamento.

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.

Figura 27 regola T4.11.1, regolamento Formula Student

3 Analisi dei requisiti

Una vettura di Formula Student deve essere altamente adattabile per poter affrontare le diverse prove dinamiche, le superfici di asfalto e i vari stili di guida dei piloti che la utilizzano. Uno dei requisiti del volante è quello di poter regolare attraverso bottoni e manettini dei parametri in centralina che permettono di adattare la vettura a seconda delle necessità. Tra questi, la possibilità di cambiare la potenza massima erogabile dal pacco batteria, per poter gestire manualmente i consumi in caso di anomalie nel sistema di de-rating o di quello di controllo di potenza. La coppia massima e la coppia rigenerativa massima comandata dagli inverter. La modifica dell'invasività del controllo di trazione e dell'efficacia del torque vectoring così da contribuire a migliorare il grip e la stabilità della vettura durante le curve e le accelerazioni. Un volante che soddisfa questi requisiti funzionali può fornire al pilota un controllo ottimale dell'auto e una maggiore probabilità di successo in pista.

Il volante non solo consente al pilota di impartire degli input alla vettura, ma è anche utilizzato per visualizzarne gli output. È fondamentale che le informazioni mostrate siano strettamente necessarie al pilota e facilmente leggibili, per cui il pannello del display deve essere ben visibile anche alla luce del sole. Per agevolare la lettura, i caratteri utilizzati devono essere i più grandi possibili e l'uso di colori può rendere l'interpretazione dell'informazione più veloce. Considerando che le informazioni da visualizzare possono variare notevolmente a seconda della prova dinamica in corso, diventa cruciale avere a disposizione diverse pagine specifiche per ognuna di esse. In questo modo, si garantisce di avere sempre a portata di mano solo le informazioni strettamente necessarie, visualizzate in modo chiaro e ben leggibile. Inoltre, è necessario prevedere alcune pagine dedicate alla segnalazione di eventuali errori della vettura e alla visualizzazione dei dati di sensori specifici, in modo da rendere più rapido il processo di troubleshooting in caso di problemi durante i test o le ispezioni tecniche. In questo modo, il pilota e il team possono agire prontamente e con precisione per individuare e risolvere eventuali anomalie, garantendo così la massima efficienza e sicurezza della vettura.

Per quanto riguarda i requisiti più tecnici, è indispensabile che il volante comunichi in modo efficiente con il resto del sistema, utilizzando il CAN bus che rappresenta la tecnologia di riferimento per tutte le altre componenti della vettura. È essenziale che il sistema sia in grado di fornire le informazioni in tempo reale con una frequenza di almeno 30Hz, considerando che il display rappresenta un po' il "collo di bottiglia" in questo processo. Infatti, la sua massima frequenza di refresh è appunto di 30Hz.

Un altro importante requisito consiste nell'utilizzo di un hardware su misura per ottimizzare gli spazi e il peso del sistema del volante, il quale riveste un'importanza fondamentale nell'ambito delle competizioni automobilistiche. In aggiunta, il sistema del volante deve essere equipaggiato con un microcontrollore della serie STM32F4, già utilizzato per la VCU e il BMS. Questa scelta è stata motivata dalla necessità di garantire una facile manutenzione del sistema da parte dei membri del team, i quali potrebbero essere chiamati a lavorare su progetti differenti nel corso degli anni. L'utilizzo dello stesso microcontrollore consente di semplificare notevolmente l'apprendimento delle tecniche di manutenzione, facilitando il passaggio da un progetto all'altro e garantendo la massima efficienza del lavoro svolto. In questo modo, si assicura un'ottima gestione delle risorse e una maggiore rapidità nell'intervento in caso di necessità.

4 Interfaccia pilota



Figura 28 Volante PSR02-S

Il volante è dotato di 6 pulsanti, 10 LED, due switch rotativi e uno schermo touch screen. In questo capitolo analizzeremo singolarmente ogni componente e tutte le funzionalità che essi permettono.

4.1 Pulsanti

4.1.1 Regolazione potenza

Analizzando l'immagine sopra riportata, a partire dalla coppia di pulsanti in alto a destra, possiamo notare che essi sono etichettati con la lettera "P". Questi pulsanti sono utilizzati per regolare la massima potenza erogata dal pacco batteria, che secondo il regolamento non può superare gli 80 kWh. Questa funzionalità risulta molto utile durante la prova di accelerazione, in cui ogni pilota ha due tentativi a disposizione. A seconda della strategia adottata, è possibile scegliere di impostare la potenza al massimo durante il primo tentativo, e nel caso in cui si superi la soglia consentita, il pilota può ridurre leggermente la potenza tramite i pulsanti. La potenza massima erogata è visibile a schermo.

4.1.2 Regolazione coppia motrice

Per quanto concerne la coppia di pulsanti in alto a sinistra, essi sono utilizzati per regolare la coppia trasmessa ai motori. Tale valore è limitato a un massimo di 90 Nm, come specificato nel datasheet dei motori. Questi pulsanti vengono spesso utilizzati in combinazione con il controllo di trazione, al fine di ottenere la corretta sensibilità sull'acceleratore. Ad esempio, se l'asfalto è più scivoloso del previsto e le ruote tendono a slittare durante la fase di accelerazione, il controllo di trazione deve anticipatamente ridurre la coppia motrice, in modo da evitare che le ruote slittino ulteriormente. Diminuendo leggermente la coppia, si può accelerare con più decisione senza perdere tempo e stabilità, e in caso di pattinamento, il controllo di trazione interverrà in modo da correggere il problema senza dover ridurre drasticamente la coppia.

4.1.3 Regolazione frenata rigenerativa

La coppia di bottoni centrali invece serve a regolare la coppia rigenerativa, questa incide drasticamente sul feeling della frenata, sui consumi e sulle temperature. Durante una prova di Endurance vogliamo che questa sia molto alta per poter recuperare più energia possibile in fase di frenata, questa però comporta l'innalzamento delle temperature delle celle che non possono superare i 60°C. Quindi in caso di avvicinamento alla soglia massima è opportuno diminuire la coppia rigenerativa attraverso i pulsanti. Un altro scenario potrebbe essere che la coppia rigenerativa troppo alta crei instabilità in frenata causando il bloccaggio delle ruote posteriori, in questo caso il pilota la regola a piacimento per trovare il feeling giusto.

4.1.4 Ready to drive



Figura 29 Pulsante Ready To Drive

Come si può notare in figura 29 il bottone sinistro usato per la rigenerazione ha una seconda etichetta chiamata RTD. Infatti, durante la procedura di accensione il pulsante assume il ruolo di *Ready To Drive Button*. In questo modo è stato possibile evitare l'aggiunta di un bottone supplementare.

4.2 Switch rotativi

Per entrambi i manettini sulle 10 posizioni disponibili ne sono utilizzate solamente 8, questa scelta è stata fatta perché più di 8 sarebbero state inutili e così facendo abbiamo dato la possibilità al pilota di aver un margine di errore, questo perché la posizione 1 corrisponde allo spegnimento del controllo associato, mentre la 10 sarebbe dovuta essere la massima invasività del controllo. Dal momento che un circuito di Formula Student è molto frenetico e le modifiche da volante devono essere fatto in un brevissimo tempo, il rischio sarebbe stato che il pilota potesse passare da un estremo all'altro inconsciamente. In questo modo abbiamo due livelli di zona morta che assumono il valore 8 se si stava ruotando il manettino in senso orario, assumono invece il valore 1 se si stava ruotando in senso antiorario, questo non toglie la possibilità di passare direttamente dalla posizione 1 alla 8.

4.2.1 Traction control

Il selettore a destra consente di regolare il controllo di trazione: con la posizione 1, il controllo è disattivato; man mano che si aumenta il valore, il controllo diventa più invasivo, tagliando maggiormente la coppia e in modo più tempestivo. Questa regolazione è molto utile poiché la vettura viene utilizzata da piloti con differenti background e stili di guida. Grazie alla regolazione del controllo di trazione, la vettura può essere adattata al comportamento dei piloti, che possono mantenere il loro modo di premere l'acceleratore.

4.2.2 Torque Vectoring

Il manettino di sinistra, invece, consente di regolare il controllo noto come "Torque vectoring", spesso definito come "differenziale elettronico". Questo controllo alla guida permette di fornire una coppia differenziata alle ruote durante la fase di curva, facilitando la sterzata. Anche in questo caso, la posizione 1 rappresenta il controllo spento, mentre più si aumenta il valore, più il controllo diventa aggressivo. Ogni pilota ha il proprio stile di guida e la corretta regolazione del torque vectoring permette di aumentare o diminuire il sottosterzo.

4.3 LED

Partendo da sinistra abbiamo 3 led rossi, 4 led gialli e 3 verdi. Questi indicano lo stato di carica della batteria. I Led possono accendersi solamente in modo sequenziale da sinistra verso destra, questo significa che non possiamo avere un led acceso senza che tutti quelli alla sua sinistra siano accesi. Quando tutti i led sono illuminati contemporanea significa che la batteria ha una carica compresa tra il 91% e il 100%. Mentre invece se c'è un solo led acceso significa che la batteria è compresa tra l' 1% e il 10%.

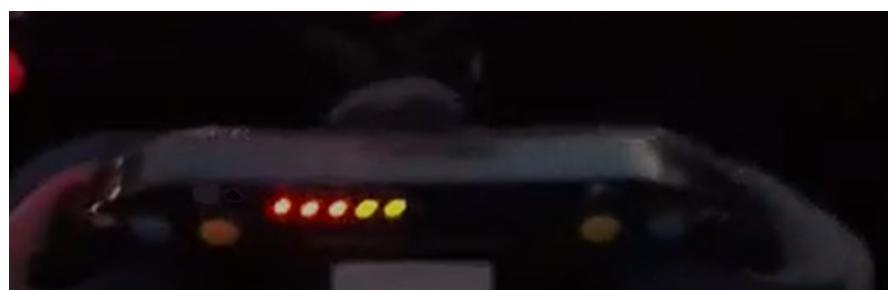


Figura 30 Illuminazione LED

Nella Figura 30 possiamo vedere cinque led accesi questo significa che la batteria è compresa tra il 40% e il 50%.

4.4 Display

4.4.1 Pagina generale

La pagina principale, rappresentata in figura 31, è utilizzata principalmente per la prova di AutoX, in cui il pilota dispone di due giri per cercare di terminare il percorso nel minor tempo possibile. Poiché la prova richiede la massima concentrazione sulla guida, non c'è tempo per guardare il display. Per questo motivo, le informazioni visualizzate su di esso sono utili nella fase di preparazione al giro, durante la quale il pilota verifica che tutte le impostazioni per la guida siano corrette e che non ci siano anomalie, come batteria scarica o temperature troppo elevate. Inoltre,



Figura 31 Pagina generale

una volta che il pilota ha completato il giro cronometrato, può leggere il tempo impiegato sul display e, dopo aver completato il secondo giro, può anche leggere la differenza di tempo rispetto al giro precedente. La visualizzazione del tempo è possibile grazie ad un algoritmo che sfrutta il sistema Gps installato sulla vettura, che però non verrà approfondito in questo lavoro di tesi.

4.4.2 Pagina Endurance

La pagina Endurance, riportata nella figura 32, come si può intuire dal nome, viene utilizzata durante la prova di Endurance. Il dato più importante riportato al centro della pagina è la carica della batteria, poiché questa gara di durata richiede una gestione accurata della batteria e delle temperature. Per aiutare il pilota nella gestione della batteria, è stata riportata in alto a sinistra, etichettata con “**Residual**”, una stima del valore percentuale di batteria rimanente alla fine dell'Endurance, mantenendo lo stesso passo degli ultimi due giri. L'obiettivo del pilota è di trovare un passo di gara tale per cui quel valore sia il più vicino possibile allo zero ma positivo. In questo modo è possibile estrarre il massimo potenziale della vettura. Nonostante la gestione del passo gara spetti al pilota, il sistema VCU è dotato di algoritmi di de-rating che intervengono limitando la potenza e la coppia in caso di consumi elevati. Ciò significa che se il pilota adotta un passo gara troppo elevato, tali algoritmi entrano in gioco e riducono le prestazioni della vettura per garantire il raggiungimento del traguardo.

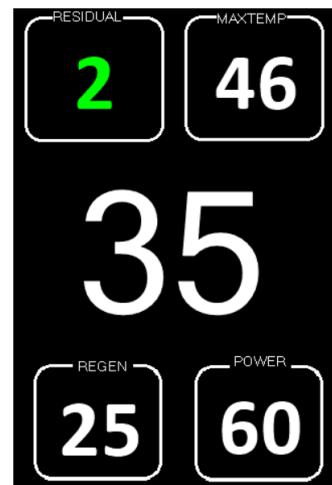


Figura 32 Pagina Endurance

Gli altri dati visualizzati sono:

MAXTEMP: Temperatura della cella più calda nell'accumulatore, se questo valore supera i 60°C la macchina si ferma in automatico come previsto da regolamento.

REGEN: Coppia rigenerativa massima in frenata.

POWER: Potenza massima erogabile dal pacco batteria.

4.4.3 Pagina Acceleration

Per la prova di accelerazione è stata creata una pagina dedicata (figura 33), poiché i dati rilevanti per questa prova sono molto diversi rispetto alle altre. In questa prova, ogni pilota ha a disposizione due tentativi, per un totale di quattro, con l'obiettivo di percorrere un rettilineo di 75 metri nel minor tempo possibile. Per riuscire a ottenere un buon risultato, è necessario avvicinarsi il più possibile ai limiti consentiti dal regolamento e dalle gomme in fase di accelerazione. Se il limite di potenza imposto dal regolamento viene superato, il tentativo è nullo. Utilizzando il volante, è possibile verificare tra un tentativo e l'altro qual è stata la potenza massima raggiunta e regolarla di conseguenza. Lo stesso vale per l'aderenza delle gomme al terreno: viene visualizzato il coefficiente di slittamento massimo delle ruote motrici, così da sapere quanto queste hanno slittato in fase di accelerazione e regolare di conseguenza la coppia e il controllo di trazione. Vediamo ora a cosa servono i singoli dati:

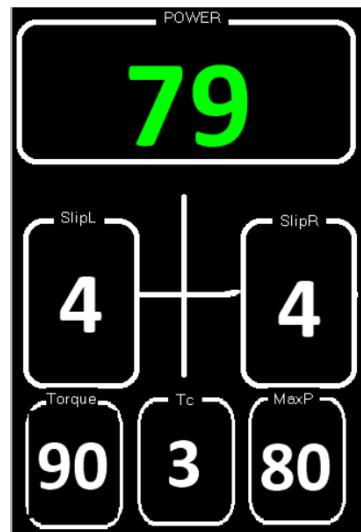


Figura 33 Pagina Acceleration

POWER: Massima potenza erogata durante la prova.

SlipL: prima cifra decimale del coefficiente di *slip* della ruota posteriore sinistra.

SlipR: prima cifra decimale del coefficiente di *slip* della ruota posteriore destra.

Torque: Coppia massima comandabile ai motori

Tc: settaggio controllo di trazione.

MaxP: Massima potenza erogabile dal pacco batteria, questo è il tetto massimo a cui lavora il controllo di potenza presente in VCU.

4.4.5 Pagina Brake Test

Il brake test, o prova di frenata, è una delle prove tecniche della competizione di Formula Student. Durante questa prova, la vettura viene fatta accelerare a una velocità prestabilita e poi frenare bruscamente, con l'obiettivo di bloccare tutte e quattro le ruote eseguendo la frenata in modo sicuro e stabile.

Nella schermata il pilota può vedere in tempo reale la velocità a cui sta andando, sotto la voce **SPEED**, ma soprattutto può vedere quali ruote non si sono bloccate durante la frenata. Se il quadrato è verde vuol dire che la ruota si è bloccata correttamente in fase di frenata, mentre se è rosso significa che non si è bloccata correttamente. Sotto la voce **PRESSURE** si può vedere la massima pressione (in bar) raggiunta dall'impianto frenante durante la prova.

L'algoritmo che ha il compito di determinare se la ruota è bloccata confronta la velocità rilevata dal GPS con quella rilevata dal sensore di ruota fonica. Tuttavia, questo aspetto non verrà approfondito in questo lavoro di tesi.

4.4.6 Pagina Fault

Ci sono due pagine dedicate ai guasti della vettura, una di queste riguarda i guasti del BMS e l'altra quelli della VCU. Queste pagine sono utilizzate per comprendere quale problema abbia causato l'arresto della vettura. Infatti, se si verifica qualsiasi tipo di guasto, non sarà più possibile fornire la coppia ai motori e quindi la macchina si fermerà.

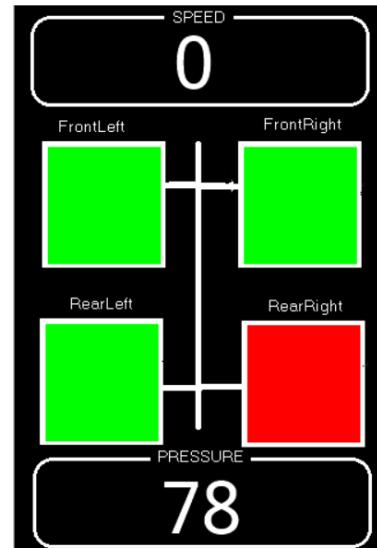


Figura 34 Pagina Brake Test

FAULT VCU	
Inv Sx	Inv Dx
Inv Sx TO	Inv Dx TO
Bms TO	Bms OT
Bms OV	Bms UV
BMS Fault	
Precharge	APPS
Contactor	Pedal
Other Fault	

Figura 35 Pagina Fault VCU

FAULT BMS	
Shutdown	
Cont Pos	
Prech slow	
Cont Neg	
Cell OV	Cell UV
Cell OT	Master OT
VCU TO	
Other Fault	

Figura 36 Pagina Fault BMS

Guardando la figura 35 soprarportata, grazie alla colorazione rossa si può facilmente intuire che il componente in fault è l'inverter sinistro. Nel momento esatto in cui si verifica un fault viene visualizzata per circa due secondi una pagina popup in cui segnala il fault.

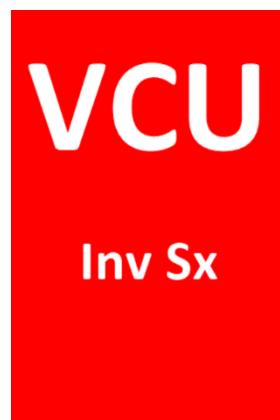


Figura 37 Popup Fault

4.4.7 Pagina sensoristica

È presente anche una pagina dedicata alla sensoristica della vettura, che viene utilizzata principalmente per verificare, quando si è ancora ai box, che tutti i sensori stiano funzionando correttamente. Senza questa pagina, sarebbe necessario collegarsi al computer e scaricare i dati o visualizzarli in tempo reale attraverso la telemetria live.

Temperature	
Inv Sx 56	Inv Dx 56
Mot Sx 63	Mot Dx 63
CMax 35	CMean 34
Super B 13	
General	
CMax V 410	Min V 405
BMS ST 1	VCU ST 7
Voltage 576	LEM 32

Figura 38 Pagina sensoristica

4.4.8 Popup cambio parametro

Ogni volta che cambiamo un parametro attraverso un pulsante o ruotando un manettino apparirà a schermo un schermata popup che ci permette di visualizzare facilmente che parametro abbiamo cambiato e quale valore assumerà. Supponiamo di aver premuto il tasto che aumenta la frenata rigenerativa, verrà visualizzata sul display la pagina in figura a destra.



Figura 39 Popup cambio parametro

5 Architettura hardware

5.1 Introduzione

In questo Capitolo si descrivono i principali protocolli di comunicazione e i dispositivi hardware utilizzati per la realizzazione del sistema volante. In figura 40 si possono osservare i macro-componenti coinvolti e come comunicano tra loro. È importante dire che il volante si occupa solo di raccogliere gli input del pilota e comunicarli alla VCU che gestirà il cambio dei parametri per poi comunicare il nuovo dato una volta aggiornato.

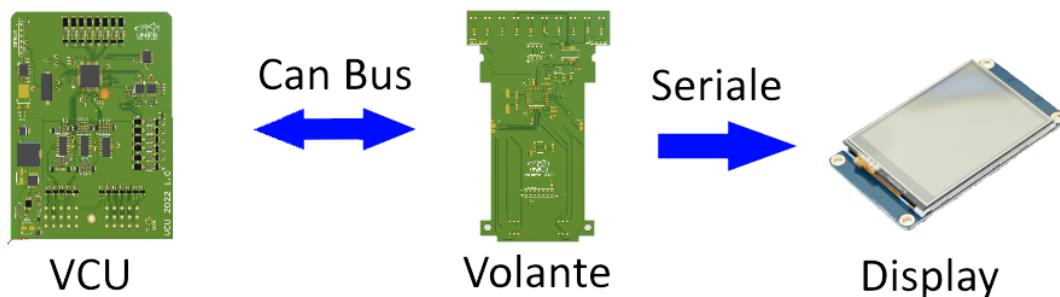


Figura 40 Architettura hardware

5.2 Componenti hardware coinvolti

Per soddisfare al meglio i requisiti funzionali è stato deciso di realizzare un circuito stampato dedicato per le funzionalità del volante. Questa è in grado di svolgere tutte le funzionalità necessarie grazie ai seguenti componenti di cui è dotato:

- Microcontrollore STM32F412RE che raggiunge una frequenza di clock massima di 100MHz, dotato di interfaccia CAN per comunicare con la VCU e di un'interfaccia USARTs che permette la visualizzazione sul display delle informazioni necessarie.
- Tranceiver CAN BUS 595-SN65HVD230MDREP per arbitrare l'invio e la ricezione dei messaggi CAN.



Figura 41 Microcontrollore

- Regolatore di tensione LDO NCV1117ST33T3G per poter abbassare la tensione di alimentazione del sistema volante (5V) a quella necessaria per alimentare il microcontrollore e il tranceiver (3.3V).
- Interfaccia di espansione Input/Output a 16 bit dotata di interfaccia I2C per comandare i 10 led sul volante.

Per quanto riguarda il display è stato scelto Il NX4832T035 in quanto soddisfa i requisiti funzionali descritti nel capitolo precedente. Inoltre, il rapporto qualità prezzo vantaggioso e il software gratuito in dotazione chiamato Nextion Editor per la gestione e creazione della grafica visualizzata a schermo lo hanno reso il candidato ideale.

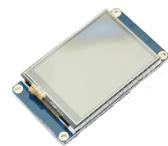


Figura 42 Nextion Display

Lo schermo è di 3.5" risulta essere abbastanza grande da poter visualizzare tutte le informazioni ma non troppo da compromettere l'ergonomia del volante.

5.3 Comunicazione Can Bus

Il protocollo CAN (Control Area Network) è un bus seriale di comunicazione digitale di tipo "broadcast" (modalità di trasmissione di pacchetti da un'emittente ad una molteplicità di riceventi). È il protocollo più utilizzato in ambito automotive grazie alla sua alta immunità ai disturbi e alla sua flessibilità, infatti i nodi non hanno un indirizzo che li identifica e possono quindi essere aggiunti o rimossi senza dover riorganizzare il sistema. Il protocollo è definito dallo standard ISO11898 che standardizza le modalità di realizzazione del cablaggio e la struttura dei messaggi. Il protocollo CAN consiste in due bus chiamati CAN-H e CAN-L, collegati da una resistenza complessiva di 60Ω , e condivisi tra tutti i dispositivi collegati al bus. I segnali trasmessi sui due cavi sono di tipo differenziale, ovvero un cavo trasmette il valore positivo mentre l'altro trasmette il corrispondente valore in negativo. Per garantire una corretta lettura del bus, il ricevente deve semplicemente sottrarre i valori trasmessi sui due cavi. Se il risultato è un valore di tensione alta, verrà codificato come "zero", mentre un valore di tensione bassa corrisponderà a "uno". Questo meccanismo si rivela particolarmente utile in caso di disturbi elettrici o magnetici, in quanto questi

vengono trasmessi sui cavi con valori uguali che vengono poi annullati tramite il calcolo della differenza.

5.3.1 Formato del bus Frame

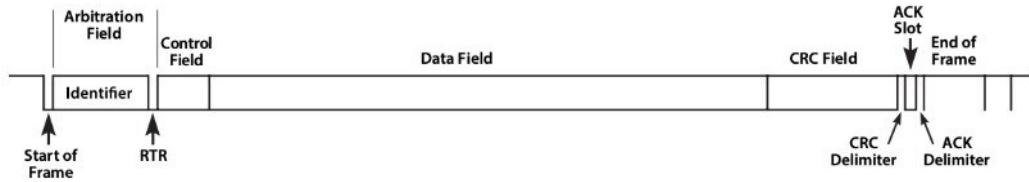


Figura 43 Frame protocollo CAN

La struttura tipica di un frame del protocollo CAN è rappresentata in figura 43 e contiene i seguenti campi.

1. **Start of Frame (SOF)**: un unico bit di inizio di frame che ha sempre valore dominante (0) e viene utilizzato per sincronizzare i dispositivi sul bus.
2. **Arbitration Field**: composto da 11 bit, viene utilizzato per l'arbitraggio del bus e determina la priorità di trasmissione dei messaggi.
3. **Control Field**: composto da 6 bit, viene utilizzato per specificare la lunghezza del campo dei dati e per indicare se il messaggio è un Remote Frame (RTR) o un Data Frame.
4. **Data Field**: può contenere fino a 64 bit di dati, dipendendo dalla lunghezza specificata nel Control Field.
5. **CRC Field**: composto da 15 bit, viene utilizzato per rilevare eventuali errori durante la trasmissione del messaggio.
6. **Acknowledgment Field**: composto da 2 bit, viene utilizzato per confermare la ricezione corretta del messaggio da parte del destinatario.
7. **End of Frame (EOF)**: composto da 7 bit, viene utilizzato per indicare la fine del frame e per consentire al bus di tornare in modalità di ascolto.

5.4 Comunicazione seriale

La comunicazione seriale è un protocollo di comunicazione utilizzato in molti dispositivi elettronici per scambiare dati tra di loro. Questo tipo di comunicazione si

basa sulla trasmissione di dati in sequenza, un bit alla volta, lungo un singolo canale di trasmissione. Il dispositivo che trasmette i dati invia un segnale seriale sul canale di trasmissione, che viene ricevuto dal dispositivo destinatario e decodificato per ricostruire i dati originali. La comunicazione seriale ha il vantaggio di essere semplice, affidabile e di richiedere un numero limitato di cavi per la trasmissione dei dati, il che la rende particolarmente adatta per dispositivi con risorse limitate come microcontrollori o sensori.

La comunicazione seriale può avvenire in modalità sincrona o asincrona.

Nella modalità sincrona, la comunicazione avviene in modo sequenziale e ogni dispositivo coinvolto nella trasmissione deve essere sincronizzato con un segnale di clock comune. I dati vengono trasmessi in blocchi di dimensioni fisse e con una frequenza predefinita, il che garantisce una comunicazione affidabile ma richiede una maggior complessità nella gestione della sincronizzazione.

Nella modalità asincrona, invece, i dati vengono trasmessi in pacchetti che possono avere dimensioni variabili e il flusso dei dati viene controllato da un segnale di start e stop. Questo rende la comunicazione meno complessa da gestire, ma può causare problemi di timing e di affidabilità della trasmissione se la velocità di trasmissione dei dati varia.

Per la comunicazione con il display è stata scelta una modalità asincrona in quanto viene utilizzata per la visualizzazione a schermo, che richiede una frequenza di aggiornamento di 30Hz. In questo caso, la velocità di trasmissione è un fattore critico. Nel caso in cui qualche pacchetto andasse perso, ciò non costituirebbe un problema poiché sarebbe sovrascritto dal pacchetto successivo. Inoltre, la semplicità del cablaggio riveste un ruolo fondamentale, specialmente considerando gli spazi ridotti in cui il sistema è stato realizzato.

6 Progettazione del circuito stampato

Nella realizzazione del PCB del volante, è stata adottata una strategia diversa dal solito a causa della forte crisi del silicio che ha colpito il mercato dell'elettronica. Prima di procedere alla progettazione del circuito, ho acquistato i componenti necessari, in modo da evitare eventuali difficoltà di approvvigionamento nel corso del processo di progettazione. Questo mi ha permesso di avere a disposizione tutti i componenti necessari per la realizzazione del circuito, senza dovermi preoccupare della loro disponibilità a mercato.

6.1 Altium Designer

Per la progettazione del circuito stampato è stato utilizzato Altium Designer che è un software molto popolare per la progettazione di circuiti stampati. Grazie alle sue numerose funzionalità e alla sua interfaccia intuitiva, è in grado di soddisfare le esigenze di progettazione di circuiti stampati di ingegneri elettronici professionisti e studenti. Con Altium Designer è possibile creare schemi elettrici grazie alla vasta libreria di componenti, in più grazie al plugin *altium library loader* è possibile importare schema elettrico e CAD dei componenti acquistati sul mercato. Inoltre, Altium Designer offre una serie di strumenti per generare i file Gerber in modo semplice e preciso. Questi sono un formato di file standard utilizzato nell'industria manifatturiera per la produzione di circuiti stampati (PCB). Questi file contengono le informazioni necessarie per la creazione del tracciato del PCB, delle piste e delle aree di rame, nonché di tutti gli altri elementi presenti nel design del PCB.

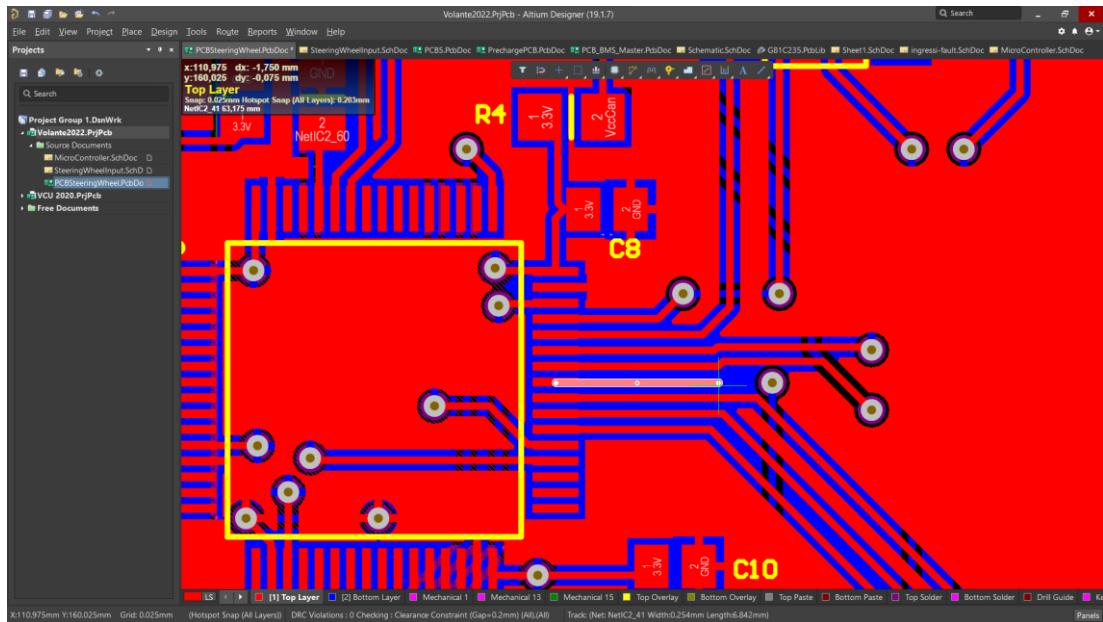


Figura 44 Altium Designer

6.2 Schema elettrico

Lo schema elettrico si divide su due file con estensione SchDoc, uno chiamato “SteeringWheelIO” e l’altro chiamato “MicroController”.

6.2.1 SteeringWheelIO.SchDoc

In questo file c’è lo schema elettrico degli input/output che riceve il volante.

Connettore

Partendo dal connettore si può vedere come il PCB sia alimentato dalla linea a 5V

proveniente dal convertitore DCDC 12/5 presente sulla monoposto. Su questa è presente un fusibile da 1A visto che la somma del consumo massimo dei componenti da datasheet si aggira intorno agli 800mA,

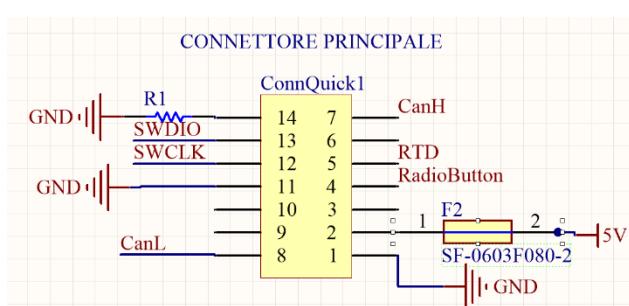


Figura 45 Schema connettore volante

verificato poi sperimentalmente.

CanH e Can L sono i due bus utilizzati per la comunicazione CAN, in particolare il volante comunica sul CAN1 della vettura, questa è dotata di un secondo canale usato solamente per la comunicazione tra Inverter e VCU.

SWDIO e SWCLK sono due segnali che vengono utilizzati in fase di DEBUG e caricamento codice; infatti, vengono collegati a un apposito strumento chiamato ST-LINK (integrato nella maggior parte delle schede STM32-Nucleo) che permette questa operazione. Una volta installato il volante sulla vettura il caricamento del codice avviene tramite CAN-BUS, grazie al programma chiamato *CanBootLoader* realizzato dall'ex membro del team Ing.Davide Draghi.

RTD è il segnale del bottone di Ready To Drive che permette alla macchina di completare la procedura di accensione, questa prevede il suo completamento premendo contemporaneamente il freno e il bottone di RTD. Questo è un punto critico in quanto senza questo la macchina non può muoversi, per questo motivo è stata predisposta un'uscita del segnale direttamente verso la centralina che lo leggerà come input digitale, la centralina acquisisce la lettura del bottone anche attraverso un messaggio CAN BUS inoltrato dal volante, in questo modo abbiamo una ridondanza del segnale che aumenta l'affidabilità della vettura.

RadioButton è un uscita digitale che dovrebbe andare ad azionare un eventuale sistema radio. È stato predisposto per eventuali usi futuri.

Manettini

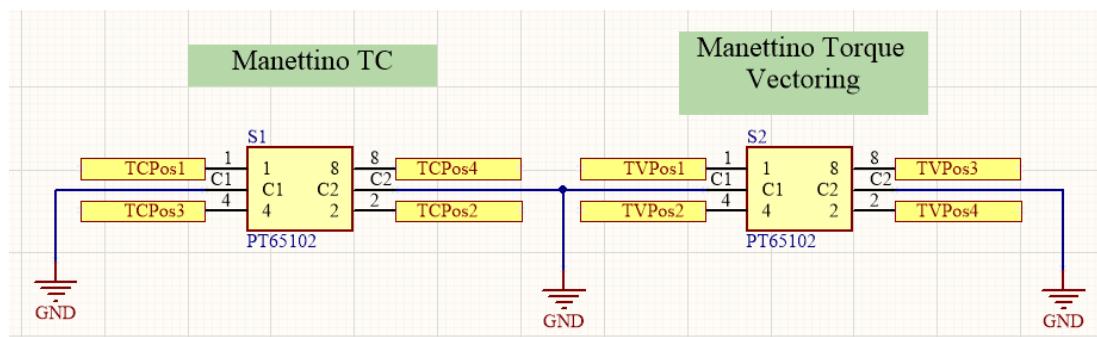


Figura 46 Schema elettrico switch rotativi

Per cambiare le impostazioni di Traction Control (TC) e Torque Vectoring (TV) sono stati utilizzati due interruttori rotativi codificati. Per entrambi i manettini la codifica dei pin 1,2,4 e 8 è in BCD ed è rappresentata dalla tabella in figura 47

POSITION	C	1	2	4	8	MARKING
0	●	●	●	●	●	0
1	●		●	●	●	1
2	●	●		●	●	2
3	●			●	●	3
4	●	●	●		●	4
5	●		●		●	5
6	●	●			●	6
7	●				●	7
8	●	●	●	●		8
9	●	●	●	●		9

In questo caso essendoci solamente 4 bit la codifica BCD non differisce dal numero binario rendendo più semplice l'elaborazione degli input.

Prendendo d'esempio il manettino del Traction Control, la colonna 1 in figura corrisponde al segnale TCP0s1, la 2 al segnale TCP0s2, e così via.

Figura 47 Codifica BCD manettini

Quindi se per esempio il manettino è in posizione 3 la codifica corrisponderà esattamente al numero binario 3 e quindi avremo i segnali TCP0s3, TCP0s4 cortocircuitati con il segnale comune (che corrisponde al riferimento di massa del circuito) e quindi avranno valore logico 0, mentre TCP0s1 e TCP0s2 avranno un valore di 3.3V grazie al pull up interno al microcontrollore (bisogna abilitarlo via software) assumendo così un valore logico alto. Il discorso è analogo per il manettino del Torque Vectoring.

Pulsanti

I pulsanti sono interruttori che si collegano a un ingresso digitale del microcontrollore, con un capo connesso al segnale di massa del circuito. Quando il pulsante viene premuto, il circuito si chiude, e l'ingresso del microcontrollore assume il valore di massa. Per evitare il rimbalzo dei pulsanti, ovvero l'oscillazione transitoria del segnale quando il pulsante viene premuto o rilasciato, in alcuni casi viene

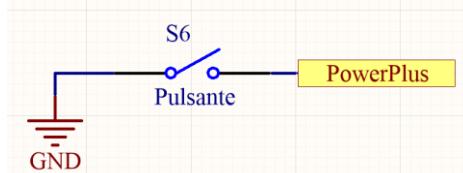


Figura 48 Schema elettronico pulsante

utilizzato un circuito anti-rimbalzo. Tuttavia, in questo caso non è stato previsto l'uso di un tale circuito, poiché l'elaborazione dei segnali digitali in ingresso avviene a una frequenza di

100Hz, riducendo il rumore dovuto alla commutazione del segnale. Per assicurarsi che il microcontrollore riconosca correttamente lo stato del pulsante, il software richiede che la variabile associata al pulsante mantenga lo stesso valore per un numero predefinito di letture prima di cambiare di stato. Ciò aiuta a filtrare eventuali rumori o oscillazioni transitorie che potrebbero influenzare il valore letto del pulsante.

Il circuito di Pull-Up come per i manettini è interno al microcontrollore e bisogna attivarlo via software.

LED

Nella figura è possibile osservare il circuito che permette di accendere i 10 LED presenti sul volante. In questo circuito, l'anodo di ogni LED è collegato ad uno dei segnali di uscita dell'espansore I/O, mentre il catodo è collegato a massa tramite una resistenza di 110Ω .

Questa resistenza viene utilizzata per limitare la corrente che attraversa ogni LED a un massimo di 25mA. Infatti, la caduta di tensione del LED è di circa 2.2V e la massima corrente erogabile dall'uscita dell'espansore è di 25mA. Pertanto, scegliendo una resistenza di 110Ω , la corrente che attraversa il LED risulta limitata a 25mA, garantendo il corretto funzionamento del circuito.

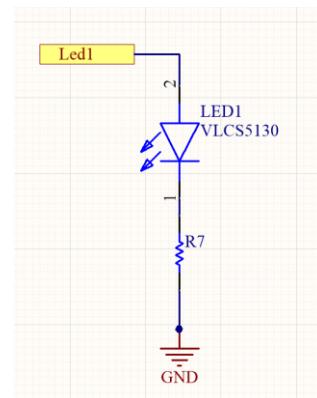


Figura 49 Schema LED

Connettore Dashboard

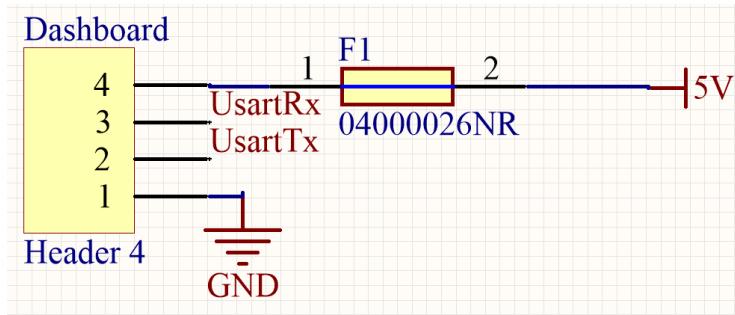


Figura 50 Connettore dashboard

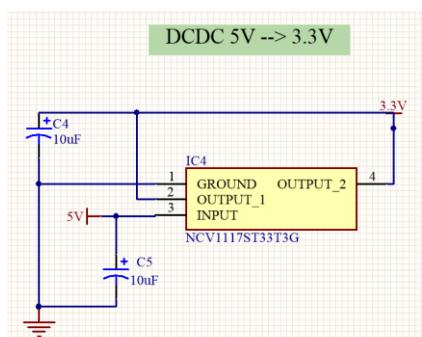
Nella figura 50 si può osservare il connettore che collega la scheda al display. L'alimentazione a 5V del display è protetta da un fusibile con rating di 500mA, che rappresenta la corrente massima supportata dal Nextion Display, come indicato nel datasheet.

I segnali UsartRx e UsartTx costituiscono il canale di trasmissione e ricezione della comunicazione seriale tra il display e il microcontrollore. Durante il collegamento del display, è necessario connettere il segnale UsartRx al pin di trasmissione seriale del display. Questo perché il canale di ricezione del microcontrollore è utilizzato come canale di trasmissione dal display, ragionamento analogo per il segnale UsartTx.

6.2.2 Microcontrollore.SchDoc

In questo file sono presenti tutti quei componenti che elaborano i segnali al fine di generare un output.

Convertitore LDO



Il convertitore LDO NCV1117ST33T3G si occupa di convertire la tensione di alimentazione a 5V presente sulla scheda in una tensione di 3.3V, necessaria per alimentare il microcontrollore e il transceiver CAN bus.

I condensatori polarizzati mostrati in figura 14 sono stati scelti seguendo le indicazioni presenti sul

datasheet del convertitore. Il datasheet fornisce le specifiche tecniche e le raccomandazioni per l'utilizzo del componente, tra cui il posizionamento dei condensatori e la capacità da utilizzare. In particolare, i condensatori polarizzati vengono utilizzati per filtrare il rumore presente sulla tensione di ingresso e sulla tensione di uscita del convertitore.

Tranceiver Can Bus

Il transceiver CAN bus ha il compito di gestire la trasmissione e la ricezione dei dati attraverso il bus CAN. In pratica, il transceiver converte i segnali digitali provenienti dal microcontrollore in segnali analogici compatibili con il bus e viceversa.

I segnali CanH e CanL sono segnali analogici utilizzati dal CAN bus per trasmettere e ricevere i dati in modo differenziale. Questi segnali vengono elaborati dal transceiver per convertirli in segnali digitali che possono essere gestiti dal microcontrollore.

CanTx e CanRx sono rispettivamente i segnali di trasmissione e ricezione digitale del microcontrollore. In altre parole, CanTx è il segnale digitale utilizzato dal microcontrollore per trasmettere i dati attraverso il CAN bus, mentre CanRx è il segnale utilizzato dal microcontrollore per ricevere i dati provenienti dal CAN bus. Il transceiver si occupa di convertire questi segnali digitali in segnali analogici compatibili con il bus e viceversa, in modo da garantire una corretta comunicazione tra i dispositivi collegati al bus.

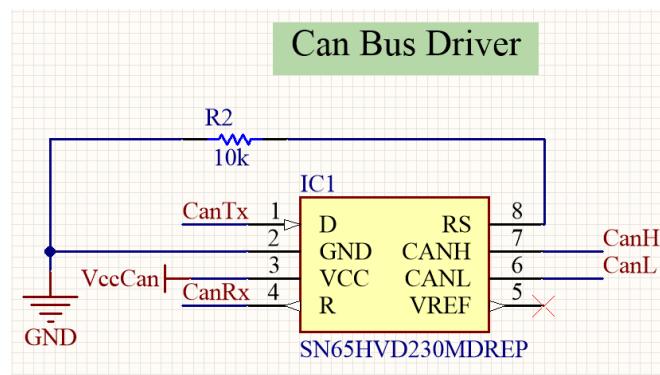


Figura 52 Schema elettrico tranceiver CAN bus

Espansore I/O

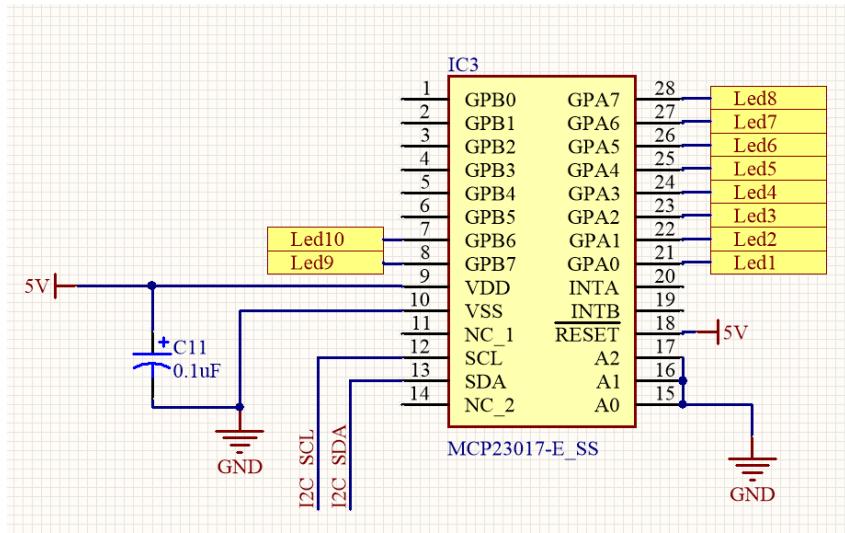


Figura 53 Schema elettrico espansore I/O

L'espansore I/O MCP23017-E_SS è responsabile di fornire l'alimentazione ai 10 LED del sistema di controllo del volante. Questa alimentazione è generata attraverso i segnali inviati con il protocollo I2C.

Grazie a questo componente, è stato possibile evitare di collegare direttamente i LED al microcontrollore utilizzando 10 ingressi digitali. Al contrario, sono stati utilizzati solo i 2 pin del bus I2C per controllare i LED, semplificando il cablaggio e riducendo il numero di pin necessari per il funzionamento del circuito.

Microcontrollore

In figura possiamo osservare su quali pin del microcontrollore sono stati collegati tutti i segnali citati in precedenza

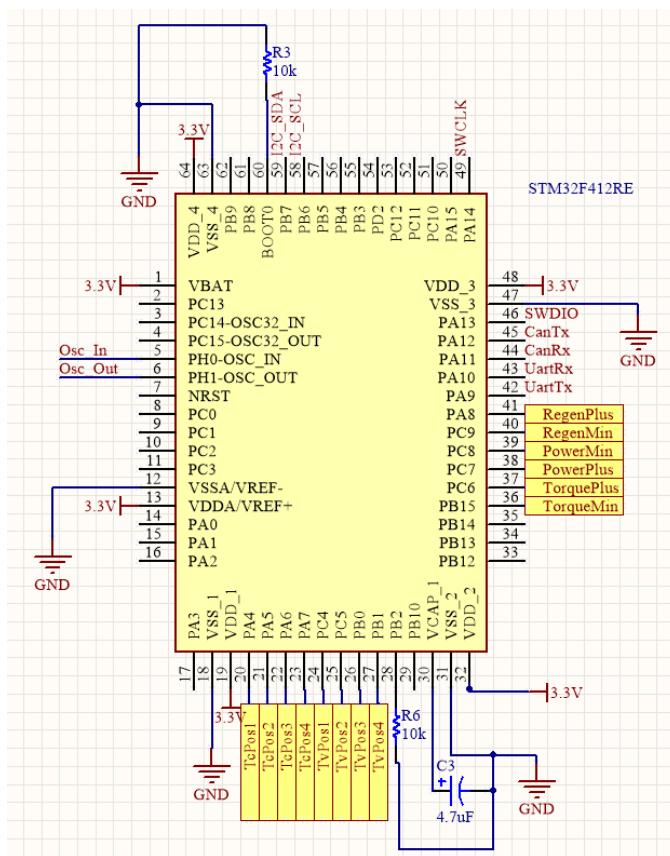


Figura 54 Schema elettrico Microcontrollore

6.3 Design PCB

Il circuito stampato è stato realizzato in FR4 che è una classe di materiale usato per circuiti stampati basato su una resina epossidica ignifuga e un composito di tessuto di vetro. FR sta per *Fire Retardant* e soddisfa i requisiti UL94V-0.

Il circuito stampato è costituito da due strati distinti: il primo ospita il piano di alimentazione e accoglie tutti gli integrati e il connettore per il cablaggio lato macchina. Il secondo strato, invece, è rivolto verso il pilota e presenta il piano di massa, oltre a quattro pad dove i cavi del cablaggio del display verranno saldati. Inoltre, su questo strato sono presenti i fori in cui sono stati saldati gli interruttori rotativi e i led.

6.3.1 Forma del PCB

Prima di procedere con la progettazione della disposizione di piste, via e integrati, è stato fondamentale collaborare con il reparto del Telaio per definire insieme la struttura esterna del volante e trovare una soluzione per fissare la scheda elettronica ad essa. In particolare, è stato necessario considerare anche la posizione del cablaggio che arriva al volante, al fine di mantenere l'integrità dello stesso. È stato quindi deciso di far passare i cavi all'interno del piantone del volante, fissandoli ad esso in modo tale che ruotino con il volante stesso, evitando così eventuali danni. La scheda elettronica è stata invece fissata all'interno del telaio del volante, utilizzando quattro viti in nylon M3 per minimizzare le vibrazioni che potrebbero compromettere il funzionamento del sistema.

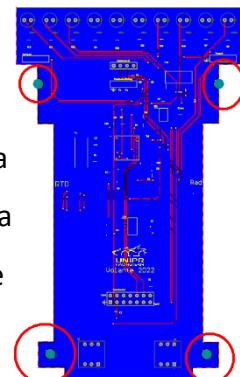


Figura 55 Fori di ancoraggio del PCB

6.3.1 Posizionamento dei componenti

Anche per il posizionamento dei componenti è stata fondamentale la collaborazione con il reparto del Telaio e con il pilota, poiché le posizioni dei bottoni e dei manettini sono state scelte sulla base dell'ergonomia e della rapidità di utilizzo quando si è alla guida. Inoltre, i LED sono stati posizionati nel punto più alto possibile in modo da essere visibili perifericamente dal conducente, il quale sarà in grado di assimilare le informazioni derivanti da essi senza dover togliere gli occhi dalla pista.

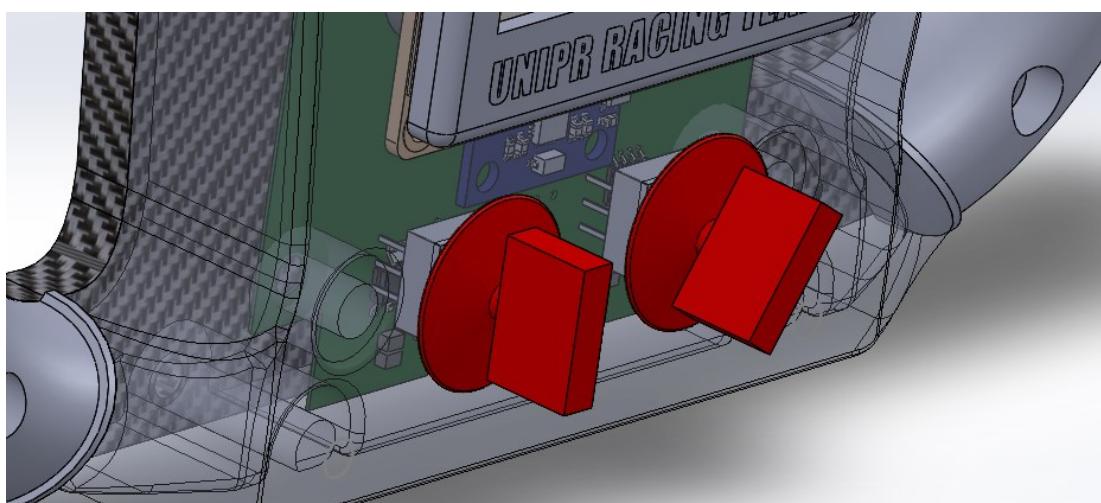


Figura 56 Posizionamento Manettini

Poiché i manettini e i led sono di tecnologia “through holes” è stato necessario posizionare i fori su cui verranno saldati esattamente al di sotto della loro posizione nella struttura esterna del volante. Questo processo è stato facilitato dalla possibilità di esportare in file CAD il PCB così che gli ingegneri meccanici potessero integrarlo nell’assieme della vettura e verificare che il posizionamento dei componenti fosse corretto o meno.

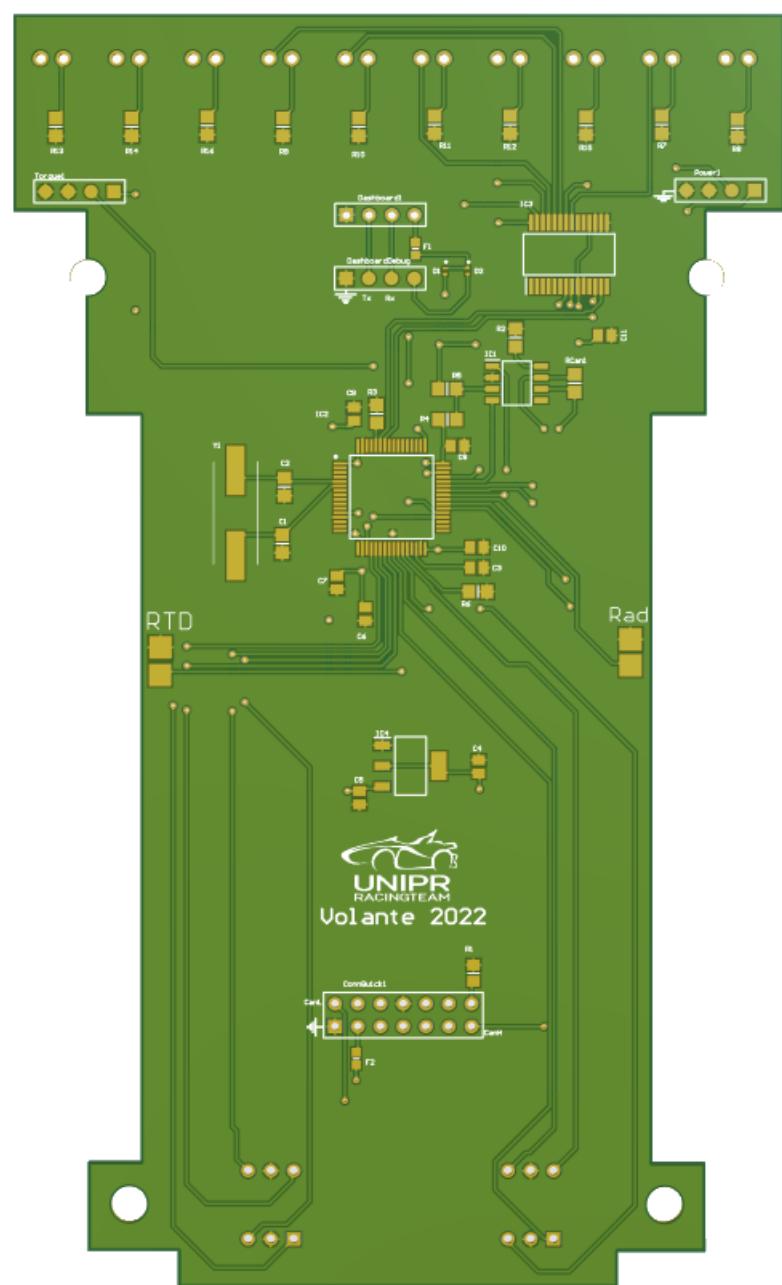


Figura 57 PCB volante PSR02-S

6.4 Integrazione dello schermo

Per collegare il display alla scheda, sono stati utilizzati dei cavi saldati direttamente sui fori presenti sul lato pcb volante e, dall'altro lato, è stato utilizzato un connettore (figura 58). Non è stato possibile utilizzare un connettore tradizionale anche lato volante a causa dello spazio limitato. In totale sono stati utilizzati quattro cavi, due per l'alimentazione (GND e 5V) e gli altri due per il canale di trasmissione seriale (cavo blu), che andrà collegato al pin di ricezione del display, e il canale di ricezione seriale (cavo giallo), che andrà collegato al pin di trasmissione del display.

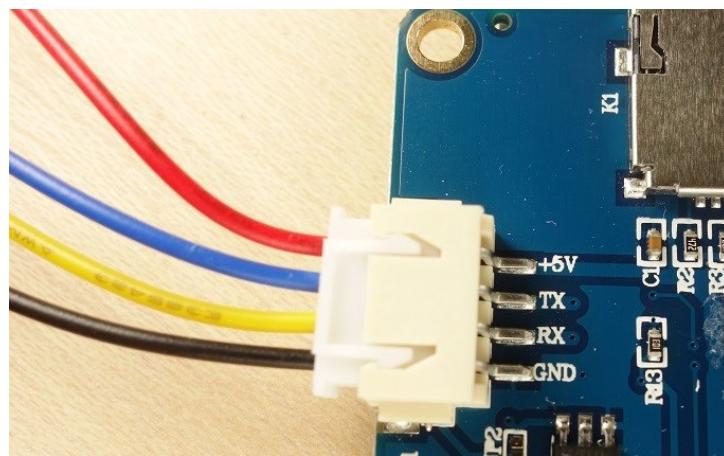


Figura 58 Connettore lato display

7 Progettazione del software

7.1 Introduzione alla progettazione del software

7.1.1 Modello a V

Nell'industria automobilistica, il modello a V rappresenta un approccio consolidato per lo sviluppo del software. Tale metodo mette in risalto l'importanza del testing in ogni fase del processo di sviluppo, il che lo rende particolarmente adatto alla realizzazione di sistemi che richiedono una elevata sicurezza, come quelli definiti Safety Critical. Con Safety Critical intendiamo un sistema dove un eventuale malfunzionamento o errore può causare lesioni gravi o addirittura la morte di persone, o danni rilevanti a proprietà.

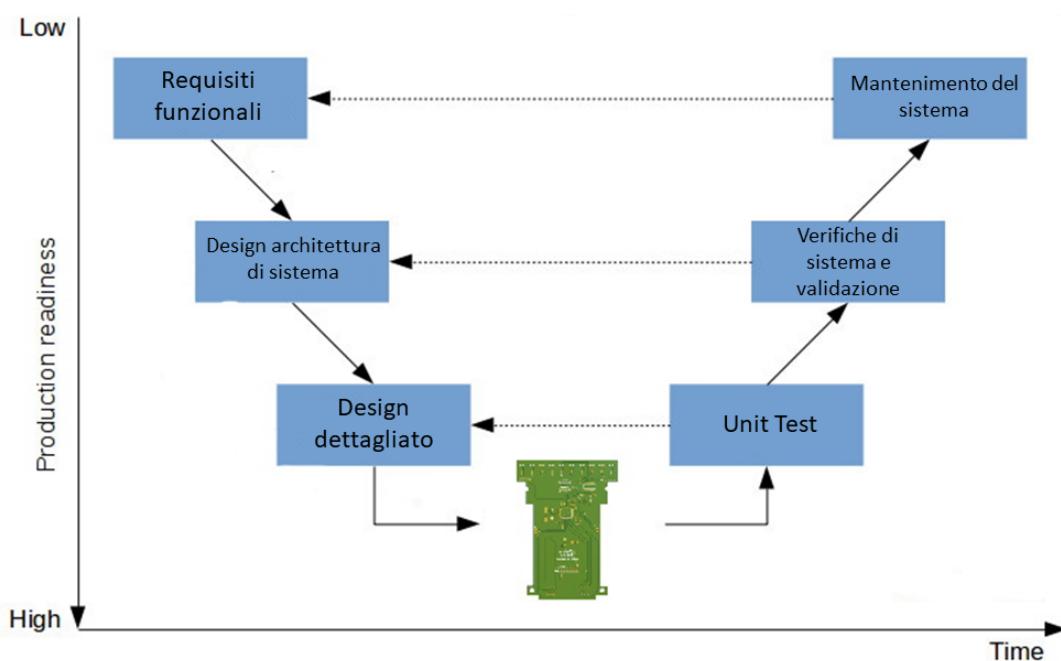


Figura 59 Diagramma modello a V

In figura 59 sono riassunti i passaggi chiave del modello a V, che sono:

- **La definizione dei requisiti:** in questa fase vengono stabiliti i requisiti di alto livello del sistema, ovvero quali funzionalità e caratteristiche deve possedere il sistema.

- **La definizione dell'architettura:** in questa fase vengono definite le specifiche architettoniche del sistema, come la suddivisione in moduli e l'assegnazione dei requisiti desiderati a ciascuna parte.
- **Il design dettagliato:** in questa fase si definiscono le specifiche complete del sistema, specificando in dettaglio come implementare le funzionalità.
- **L'implementazione:** questa fase consiste nello sviluppo effettivo del sistema.
- **Il test unitario:** in questa fase viene testata singolarmente ogni parte del sistema per verificare che soddisfi le specifiche richieste.
- **Verifiche di sistema e validazione:** dopo aver superato i test unitari, si iniziano ad integrare e verificare le parti del sistema complessivo per poi testare lo stesso.
- **Mantenimento del sistema:** questa fase consiste nella risoluzione di eventuali problemi che si verificano dopo il completamento del sistema e, se necessario, nell'aggiunta di nuove funzionalità.

7.1.2 Sistema a basso accoppiamento

Il basso accoppiamento in un sistema software si riferisce alla riduzione della dipendenza tra le diverse parti del sistema.

Il basso accoppiamento in un sistema software presenta diversi vantaggi, tra questi:

1. **Maggiore modularità:** con un basso accoppiamento, il sistema può essere suddiviso in moduli indipendenti e facilmente sostituibili, consentendo una maggiore flessibilità e facilitando l'aggiornamento o la manutenzione.
2. **Maggiore scalabilità:** il basso accoppiamento permette di espandere il sistema senza compromettere la stabilità o la qualità del software, in quanto l'aggiunta di nuovi moduli non interferisce con quelli esistenti.
3. **Maggiore facilità di testing:** un sistema con un basso accoppiamento è più facile da testare, in quanto i singoli moduli possono essere testati indipendentemente gli uni dagli altri, consentendo di individuare e risolvere eventuali problemi più rapidamente.

In un team di Formula Student, la facilità di comprensione e di manutenzione del software è essenziale, soprattutto considerando i continui cambiamenti generazionali che avvengono all'interno del team. Un basso accoppiamento del software permette di evitare il rischio di abbandonare un sistema per la sua complessità, poiché rende possibile apportare miglioramenti e cambiamenti in modo semplice e veloce senza dover ricostruire l'intero sistema da zero.

Per questo motivo il software del volante è stato diviso in tre moduli:

1. Firmware
2. Strategia
3. Librerie display

Pur essendo tre moduli con basso accoppiamento tra loro la strategia e le librerie del display sono integrate nel firmware essendo poi questo ad essere caricato sul microcontrollore. Nei capitoli successivi verrà chiarito questo concetto.

7.1.3 Sviluppo del software con Matlab/Simulink

Per sviluppare la parte più importante del codice, ovvero quella che elabora gli input acquisiti dal firmware, abbiamo utilizzato Matlab/Simulink. Questo strumento ci ha permesso di adottare un approccio a blocchi, il quale semplifica notevolmente la comprensione della logica del codice da parte di chiunque si accosti al progetto per la prima volta. Inoltre, Simulink consente di creare sotto blocchi per nascondere i dettagli legati ad essi, a meno che non si voglia approfondirne il funzionamento. Ad esempio, nella figura 60 abbiamo una visione più generale del blocco per la visualizzazione del pop up di errore, mentre nella figura 61, attraverso un semplice doppio click è possibile vedere nel dettaglio il contenuto del blocco. Questo ci

consente di mantenere un'organizzazione chiara e precisa all'interno di progetti molto grandi, come ad esempio quello del volante.



Figura 60 Blocco DisplayFault

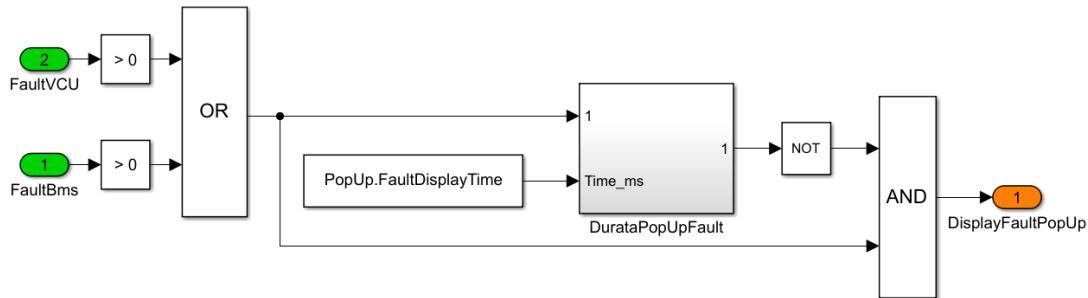


Figura 61 Interno del blocco DisplayFault

Un altro importante vantaggio nell'utilizzo di Matlab/Simulink consiste nella possibilità di simulare l'intero sistema o una porzione di esso, nonché di un singolo blocco, a seconda delle necessità. Ciò si integra perfettamente con il modello a V menzionato in precedenza.

7.1.4 Autogenerazione del codice

Una volta che la logica a blocchi è stata sviluppata e testata, è possibile generare in modo automatico il codice C per un determinato microcontrollore tramite l'uso combinato dei software Embedded Coder e MATLAB Coder, anch'essi sviluppati da MathWorks, l'azienda che ha creato Matlab e Simulink. Le librerie generate possono essere facilmente integrate nel progetto, consentendo di incorporare in pochi e semplici passaggi tutto il lavoro svolto su Matlab/Simulink nel codice del firmware.

Questo approccio presenta indubbiamente lo svantaggio di creare un codice meno efficiente rispetto a quello scritto manualmente. Tuttavia, presenta diversi vantaggi rispetto all'approccio tradizionale:

- lo sviluppo è molto più rapido

- il riutilizzo del software è molto più facile, infatti per migrare da una famiglia di microcontrollori a un'altra è necessario solo cambiare le impostazioni di generazione del codice e non riscrivere manualmente alcune parti
 - la probabilità di avere errori nel codice è molto più bassa
 - il software è più facile e veloce da testare grazie alla possibilità di simulare i blocchi in Simulink.

Poiché l'hardware utilizzato è sufficientemente potente per compensare la minore ottimizzazione del codice, e poiché la manutenibilità e la facilità di comprensione del codice sono tra gli obiettivi principali durante lo sviluppo del software all'interno del team, è stato deciso di adottare l'approccio dell'autogenerazione del codice.

```
File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer SteeringWheel.c

1 851
2 852     case SteeringWheel_IN_WaitBrake:
3 853         if (Steeringwheel_U.VcuStateMessage[0] == 6) {
4 854             Steeringwheel_DW.is_c5_SteeringWheel = SteeringWheel_IN_WaitRTD;
5 855             rtb_y_e3 = 1U;
6 856             rtb_PageProcedure = 5;
7 857         } else if (SteeringWheel_U.VcuStateMessage[0] <= 1) {
8 858             Steeringwheel_DW.is_c5_SteeringWheel = SteeringWheel_IN_VehicleOff;
9 859             rtb_y_e3 = 0U;
10 860             rtb_PageProcedure = 0;
11 861         } else {
12 862             rtb_y_e3 = 1U;
13 863             rtb_PageProcedure = 4;
14 864         }
15 865     break;
16 866
17 867     default:
18 868     /* case IN_WaitRTD: */
19 869     switch (Steeringwheel_U.VcuStateMessage[0]) {
20 870         case 5:
21 871             Steeringwheel_DW.is_c5_SteeringWheel = SteeringWheel_IN_WaitBrake;
22 872             rtb_y_e3 = 1U;
23 873             rtb_PageProcedure = 4;
24 874             break;
25 875
26 876         case 7:
27 877             Steeringwheel_DW.is_c5_SteeringWheel = SteeringWheel_IN_RTID;
28 878             Steeringwheel_DW.temporalCounter_i1 = 0U;
29 879             rtb_y_e3 = 1U;
30 880             rtb_PageProcedure = 6;
31 881             break;
32 882
33 883     }
34 884 }
```

Figura 62 Screenshot codice autogenerato

7.2 Firmware

La stessa identica struttura del firmware è stata utilizzata per entrambe le unità di controllo del veicolo (VCU) e il sistema di gestione della batteria (BMS), in quanto è stata utilizzata la stessa famiglia di microcontrollori (STM32F4). Questa scelta ha permesso di utilizzare l'ambiente di sviluppo STM32CubeIDE e le librerie sviluppate da eDriveLAB per semplificare la gestione delle principali periferiche (CAN bus, ingressi analogici, ingressi digitali, uscite digitali, timer).

Grazie a STM32CubeIDE, è possibile configurare le periferiche attraverso un'interfaccia grafica che consente di generare automaticamente il codice C, i settaggi delle periferiche sono salvate nel file di estensione “.ioc”. Questo approccio

semplifica e accelera il processo di implementazione e manutenzione del codice.

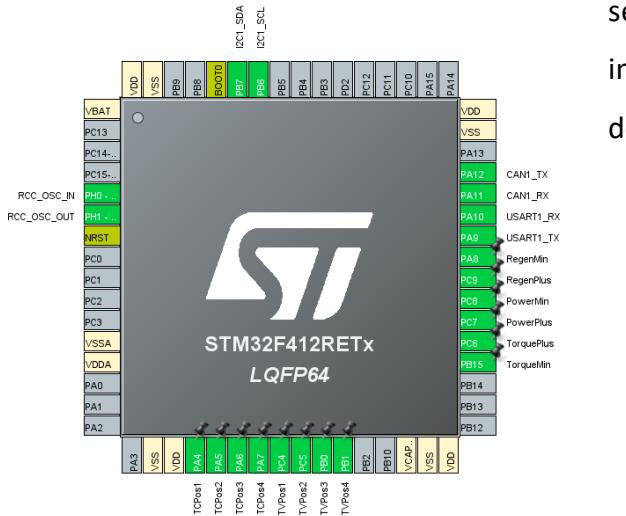


Figura 63 Input Output microcontrollore

7.2.1 Configurazione periferica CAN

Per la comunicazione su CAN bus, sono stati utilizzati i pin PA12 per la trasmissione e PA11 per la ricezione, i quali comunicano sul bus con un baud rate di 250.000 bit/s. Nella figura 64 sono riportati i settaggi utilizzati per la comunicazione.

Ogni volta che un messaggio viene ricevuto o inviato sul bus, viene lanciata un interrupt per eseguire la routine corrispondente.

Configure the below parameters :	
<input type="text"/> Search (Ctrl+F)	
<input checked="" type="checkbox"/> Bit Timings Parameters	
Prescaler (for Time Quantum)	20
Time Quantum	400.0 ns
Time Quanta in Bit Segment 1	3 Times
Time Quanta in Bit Segment 2	6 Times
Time for one Bit	4000.00 ns
Baud Rate	250000 bit/s
ReSyncronization Jump Width	1 Time
<input checked="" type="checkbox"/> Basic Parameters	
Time Triggered Communication	Disable
Automatic Bus-Off Management	Enable
Automatic Wake-Up Mode	Enable
Automatic Retransmission	Enable
Receive Fifo Locked Mode	Disable
Transmit Fifo Priority	Disable
<input checked="" type="checkbox"/> Advanced Parameters	
Operating Mode	Normal

Figura 64 Configurazione BaudRate CAN bus

Mode			
Activated			
Configuration			
Reset Configuration			
<input checked="" type="checkbox"/> NVIC Settings		<input checked="" type="checkbox"/> GPIO Settings	
<input checked="" type="checkbox"/> Parameter Settings		<input checked="" type="checkbox"/> User Constants	
NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
CAN1 TX interrupts	<input checked="" type="checkbox"/>	1	0
CAN1 RX0 interrupts	<input checked="" type="checkbox"/>	1	0
CAN1 RX1 interrupt	<input checked="" type="checkbox"/>	1	0
CAN1 SCE interrupt	<input type="checkbox"/>	0	0

Figura 65 Configurazione interrupt messaggi CAN

7.2.2 Configurazione periferica USART

La periferica USART (Universal Asynchronous Receiver-Transmitter) si occupa della comunicazione seriale tra il microcontrollore e il display. È stata impostata per trasmettere con un baud rate di 512.000 bit/s e con una lunghezza della parola di 8 bit. Come accennato nei capitoli precedenti, la modalità di trasmissione è asincrona.

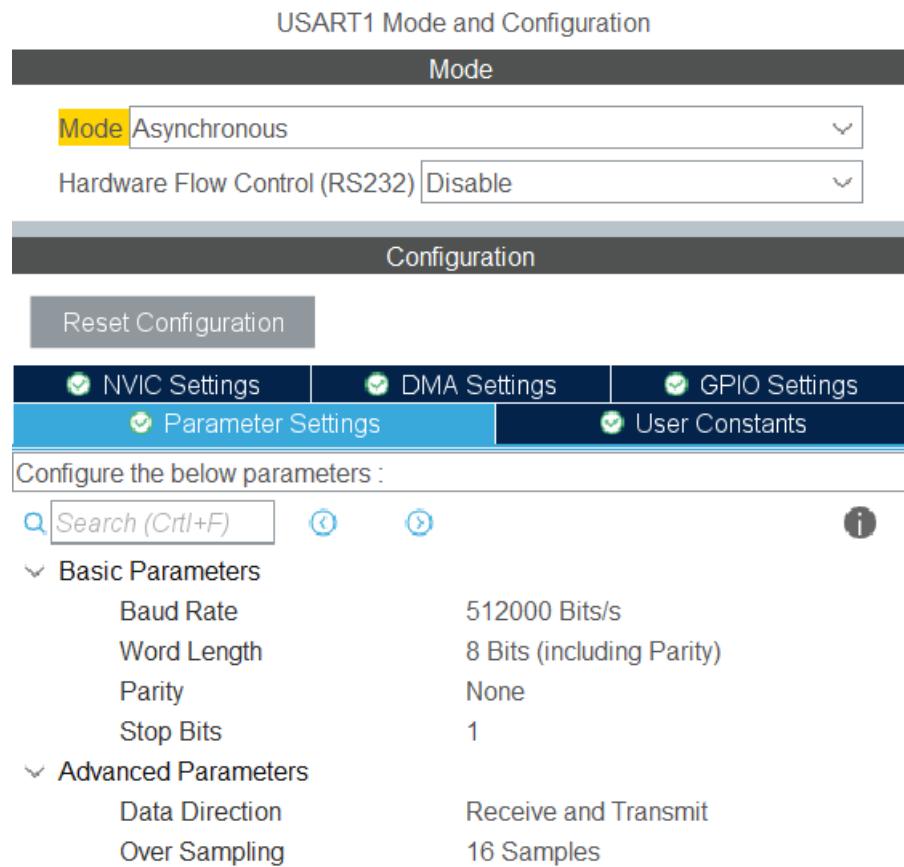


Figura 66 Configurazione BaudRate USART

7.2.3 Input digitali

I pin di input digitali del microcontrollore sono stati impostati con il circuito di pull-up interno attivo. Questo significa che il microcontrollore fornisce una resistenza di pull-up interna al pin di input digitale, che mantiene il valore logico alto quando il pin è flottante, cioè non è connesso a nessun segnale di ingresso. In questo modo, si evita l'instabilità del valore del segnale di input e si assicura una lettura corretta del segnale.

7.2.4 Impostazioni Timer

Il firmware dispone di tre timer:

- TIM 10, che genera un interrupt con una frequenza di 30Hz, si occupa di richiamare la routine di aggiornamento delle informazioni visualizzate sul monitor.

- TIM 13, che genera un interrupt con una frequenza di 100Hz, si occupa di inviare su CAN bus i messaggi di output generati dalla strategia.
- TIM 14, che genera un interrupt con frequenza di 100Hz, si occupa di richiamare la strategia di base.

I timer vengono prima abilitati nel file “.ioc” e poi attraverso le funzioni riportate in figura è possibile settare la frequenza con cui viene generata l’ interrupt.

```
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);
}
```

Figura 67 Funzioni per impostare la frequenza dei timer

Ogni volta che il microcontrollore rileva un segnale di interrupt proveniente dai timer la funzione **HAL_TIM_PeriodElapsedCallback** viene richiamata passando come parametro un puntatore all’istanza del timer. Grazie a questo possiamo risalire a quale timer ha generato l’interrupt e richiamare la routine associata ad esso.

```

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();
    }

}

```

Figura 68 Porzione di codice richiamata all'esaurimento di un timer

7.2.4 Struttura del firmware

Quando il sistema volante viene alimentato, il firmware caricato nel microcontrollore inizia la sua esecuzione. Per primissima cosa viene eseguita la funzione **main**, anche questa autogenerata e quindi con già le righe di codice necessarie per far funzionare le periferiche. Nel **main** viene solamente richiamata la funzione **StrategySetup** che si occupa di inizializzare i timer descritti in precedenza, inizializzare la comunicazione su CAN bus, inizializzare la comunicazione con protocollo I2C con l'espansore input output per poter accendere e spegnere i led e per ultimo ma non per importanza richiama la funzione **SteeringWheelStrategy_initialize** che è la funzione che permette di inizializzare le librerie autogenerate da Matlab/Simulink.

```

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 */
}

```

Figura 69 Funzione Main del firmware

```

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_IT_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);
}

```

Figura 70 Funzione di SetUp del firmware

Il Timer 14, precedentemente menzionato, richiama la funzione **StrategyMainRoutine** ogni 10 millisecondi. La funzione è responsabile di acquisire gli input digitali dai bottoni e dai manettini sul volante e inserirli in una variabile temporanea chiamata **SteeringWheel_U_Tmp** di tipo **ExtU_SteeringWheel_T**. La struttura **ExtU_SteeringWheel_T** è una struct autogenerata dalla strategia sviluppata in Matlab/Simulink e comprende tutti gli input che questa necessita per funzionare. Questo significa che una volta che la struttura è stata popolata, avremo fornito tutti

gli input necessari per eseguire l'intera strategia. La variabile temporanea è utilizzata per evitare di modificare gli input durante l'esecuzione della strategia stessa. Infatti, prima di richiamare la funzione **SteeringWheelStrategy_step**, la struttura temporanea viene copiata in quella che verrà effettivamente utilizzata dalla porzione di codice autogenerata. La struttura vera e propria viene chiamata **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();
}
```

Figura 71 Routine principale del firmware

```
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();
}
```

Figura 72 Funzione che acquisisce gli input del pilota

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

Figura 73 Funzione di lettura della posizione dei manettini

L'accensione dei LED è gestita all'interno della funzione **turnOnLed**, che utilizza la libreria open source **mcp23017**. Quest'ultima è stata scritta appositamente per comunicare con l'espansore I/O **mcp23017** tramite l'interfaccia I2C. Tale scelta è stata effettuata esclusivamente per accelerare i tempi di sviluppo. La validazione della libreria è stata effettuata mediante il caricamento di un breve codice sul microcontrollore, il quale accendeva e spegneva i LED in sequenza.

7.3 Strategia

Come già anticipato in precedenza la strategia è scritta usando Matlab/Simulink e la struttura di questa è divisa in tre parti:

- Elaborazione input (in verde)
- Generazione output (in grigio)
- Elaborazione output (in arancio)

All'interno del modello sono utilizzati valori costanti che sono riportati nel file di estensione ".m". Tra questi abbiamo i parametri per spaccettare un messaggio

CAN, come per esempio l'indice alla quale

inizia il dato, se è stato salvato in Little Endian piuttosto che Big Endian, ecc.



Figura 74 Struttura strategia

7.3.1 Elaborazione input

Pulsanti

I pulsanti, essendo ingressi digitali che possono assumere due valori sono rappresentati con una variabile booleana, dove il valore *true* rappresenta il pulsante premuto e il valore *false* il pulsante rilasciato. Poiché la strategia viene richiamata con una frequenza di 100Hz, non è stato necessario applicare un filtro anti-rimbalzo, poiché tale frequenza funge già da filtro in sé. Di conseguenza, l'input viene riportato al blocco di generazione degli output senza subire modifiche. Nel caso in cui si verifichino problemi di rimbalzo del pulsante, sarà sufficiente applicare un filtro anti-rimbalzo come mostrato in figura 76. Tale filtro consente di far passare il valore logico

alto solo se questo si mantiene costante per un numero di millisecondi pari alla costante "ButtonDebounceTime".

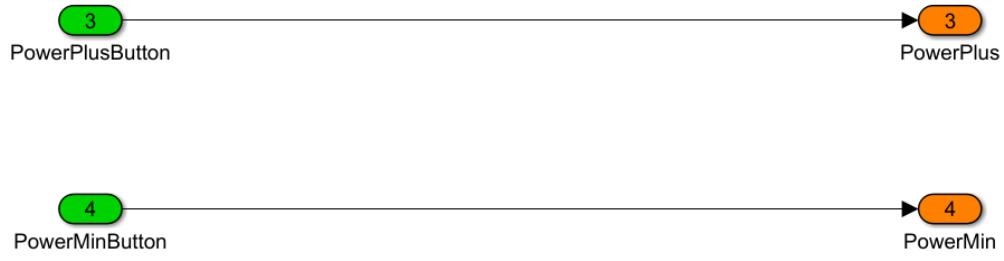


Figura 75 Input pulsanti non filtrati

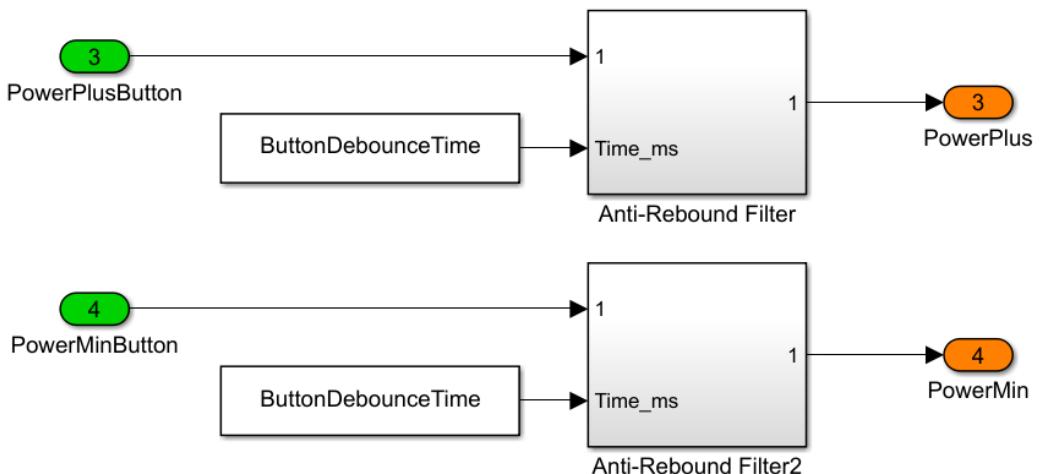


Figura 76 Input pulsanti con filtro anti-rimbalzo

Switch Rotativi

I 4 ingressi digitali corrispondenti a un singolo switch vengono codificati in BCD sul firmware che li “passa” alla strategia sottoforma di uint8 ovvero un numero intero privo di segno a 8 bit. Anche qui non è stato necessario applicare nessun filtro ma se ce ne fosse bisogno valgono le stesse considerazioni fatte per i pulsanti.

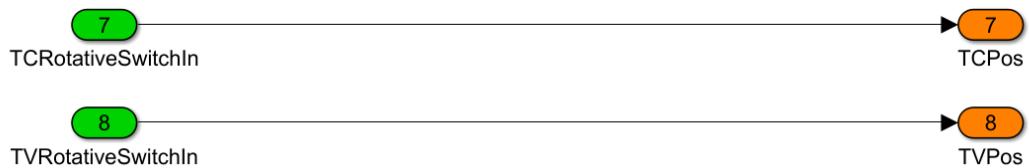


Figura 77 Input manettini non filtrati

Messaggi CAN

Una volta che il firmware è riuscito a identificare il messaggio grazie all'ID inserisce il payload nell'input corrispondente della strategia. Il payload viene rappresentato come un array di 8 byte (uint8). Questa lunghezza fissa è una caratteristica del protocollo CAN, che specifica una dimensione massima per il payload di 8 byte.

Nel blocco di elaborazione degli input, il messaggio viene scomposto per ottenere i dati contenuti al suo interno. Nel file con estensione ".m" sono indicati l'inizio del dato, la sua lunghezza in bit, il suo offset e il suo fattore di scala. A seguire, è riportato un esempio di scomposizione di un messaggio proveniente dal Battery Management System (BMS) contenente i dati relativi alla batteria.

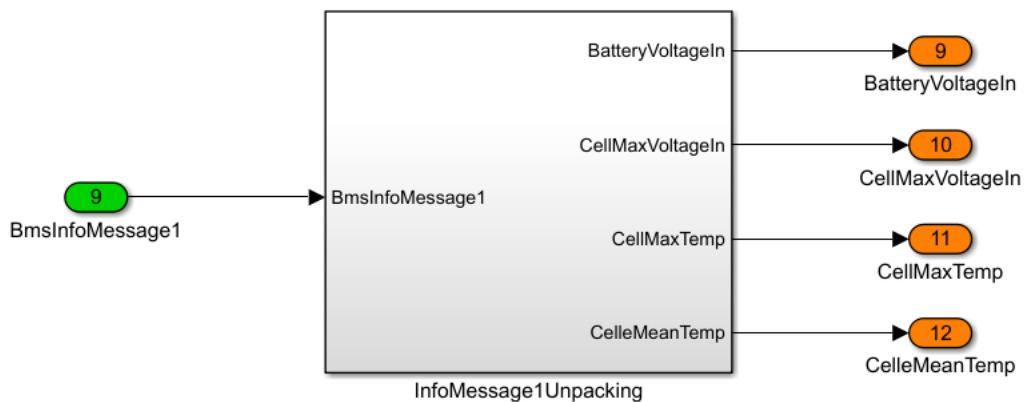


Figura 78 Blocco di scomposizione messaggio CAN

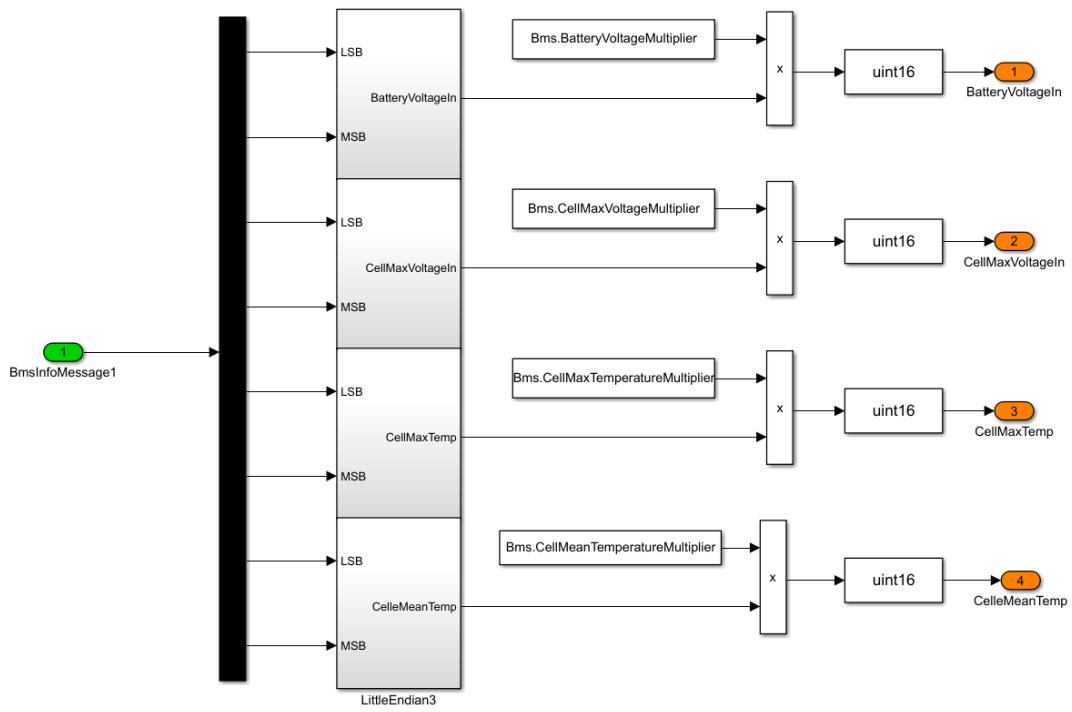


Figura 79 Interno del blocco di scomposizione di un messaggio CAN

7.3.2 Generazione degli output

PopUp di errore

Quando si verifica un fault proveniente da BMS o VCU vogliamo che sullo schermo appaia ben visibile un messaggio di errore in modo che il pilota sia a conoscenza del problema e possa immediatamente fermare la macchina e comportarsi di conseguenza. La strategia si occupa di generare una variabile booleana che indica se il PopUp deve essere visualizzato o meno.

Quando la variabile *BmsFault* o *VcuFault* (ottenute entrambe nel blocco dell'elaborazione degli input scomponendo il relativo messaggio CAN) è maggiore di zero dovrò visualizzare il popup, dopo qualche secondo voglio che l'avviso di errore scompaia così che il pilota possa visualizzare il resto dei dati così da agire al meglio.

Discorso analogo con la schermata di Warning.

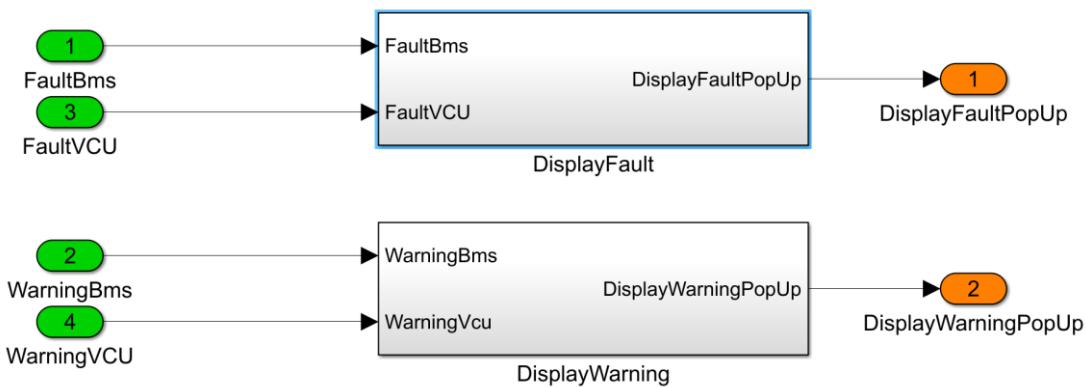


Figura 80 Blocco per settare la variabile DisplayFaultPopUp e DisplayWarningPopUp

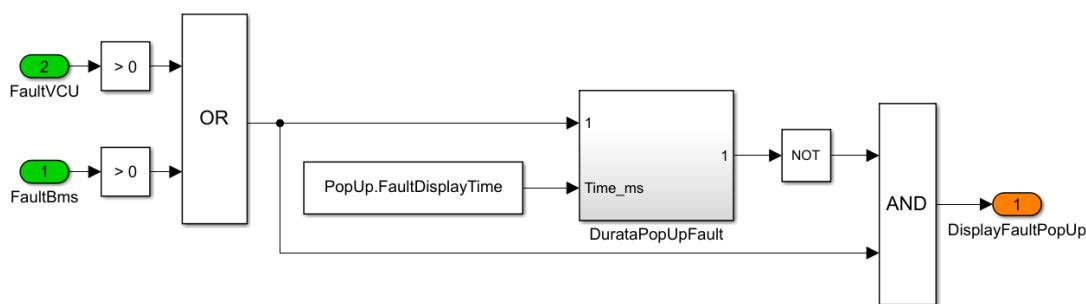


Figura 81 Interno del blocco di Figura 80

Popup pulsanti

Ogni volta che il pilota preme un pulsante o cambia posizione del manettino è fondamentale la visualizzazione di un messaggio ben visibile che indichi quale parametro sia stato cambiato e qual è il nuovo valore che ha assunto. A differenza dei messaggi di errore che hanno priorità assoluta e vengono visualizzati a prescindere, questo tipo di avviso può essere soggetto a casi di incertezza dovuti alla pressione di più pulsanti in contemporanea o in rapida successione.

Il tutto è gestito mediante una macchina a stati che assegna la priorità all'ultimo pulsante premuto. In pratica, viene impostata a "true" la variabile booleana che indica se il popup associato a quel pulsante debba essere visualizzato o meno, mentre tutte le altre variabili booleane vengono impostate a "false". Dopo un certo intervallo di tempo, corrispondente al valore della costante "Popup.ChangePopUpTime" espresso

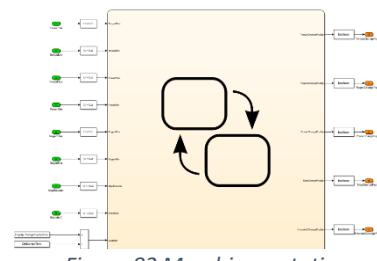


Figura 82 Macchina a stati visualizzazione popup

in millisecondi, viene impostata a "false" anche l'ultima variabile booleana corrispondente all'ultimo pulsante premuto o manettino girato.

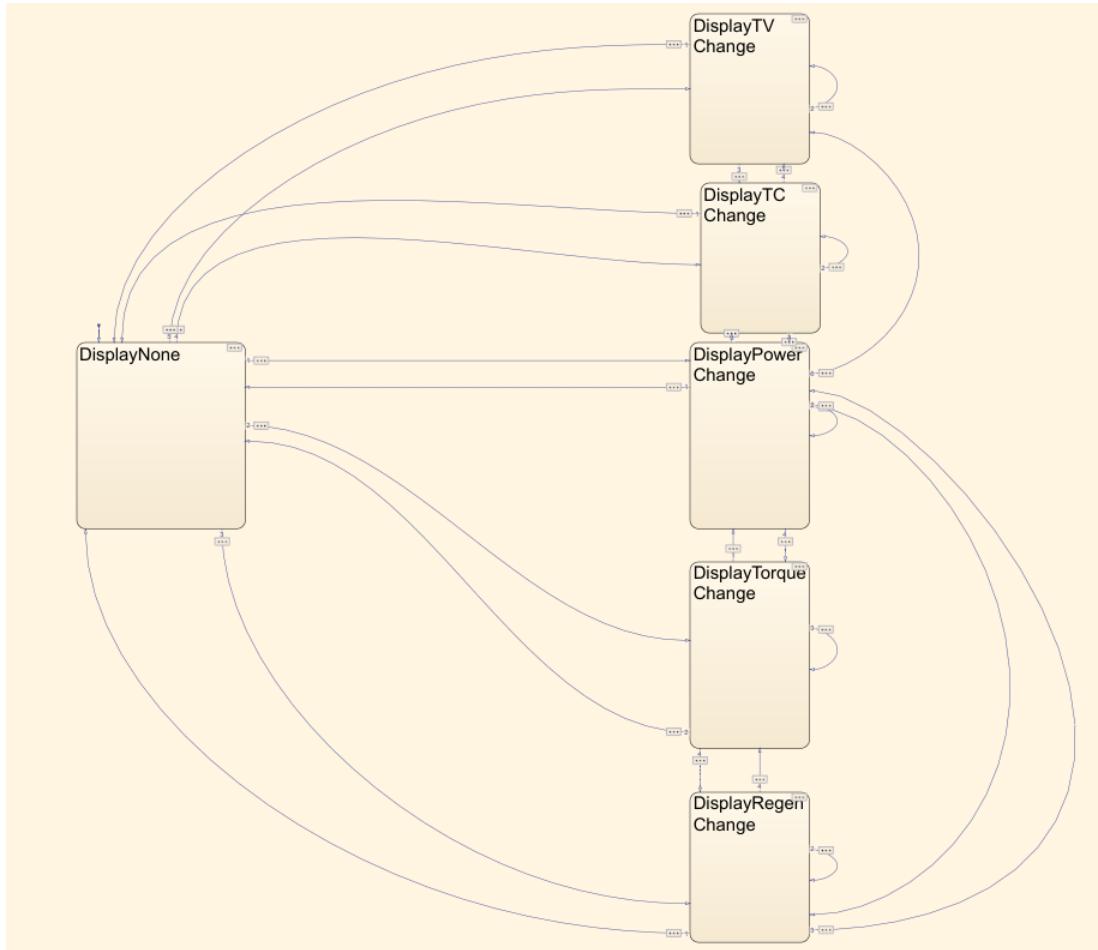


Figura 83 Interno della macchina a stati per la visualizzazione dei popup

Visualizzazione procedura di accensione

Poiché la procedura di accensione è composta da diverse fasi ben definite, alcune delle quali richiedono un input da parte del pilota, come ad esempio la pressione del pedale del freno e del pulsante RTD, ho deciso di visualizzare su schermo ogni singolo passaggio. Questo mi permetterà di capire, in caso di problemi, quale di questi passaggi sia andato storto e di dare suggerimenti al pilota su quando deve dare l'input e in che ordine.

Anche in questo caso è stata impiegata una macchina a stati che riceve come input la variabile "VcuState" (uint8), la quale indica lo stato corrente della VCU. Inoltre, come ulteriore input, la macchina a stati riceve la variabile "ChargeState", che consente di identificare il contattore che si è chiuso per fornire maggiori dettagli.

Stato	Nome	Descrizione
0	Fault	Uno dei componenti 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.

Figura 84 Stati VCU

I risultati prodotti sono rappresentati dalla variabile booleana "DisplayProcedure", che indica se la pagina di descrizione deve essere mostrata o meno, e dalla variabile uint8 *PageProcedure*, che indica quale pagina deve essere visualizzata.

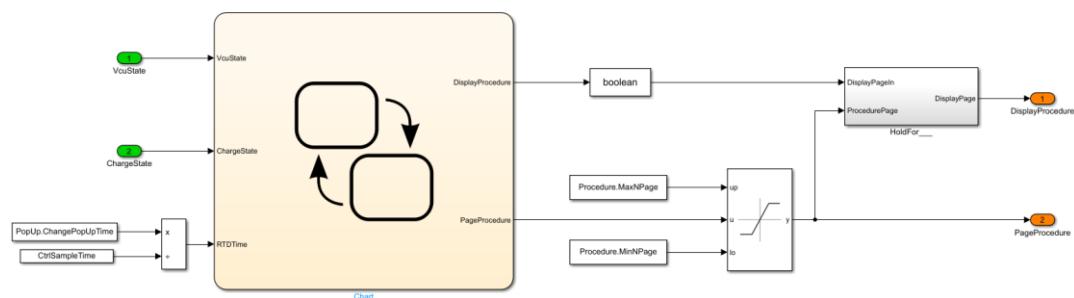


Figura 85 Interno del blocco per la visualizzazione della procedura di accensione

Il blocco chiamato "HoldFor" è una funzione che consente di mantenere una variabile a un certo valore per un determinato periodo di tempo. In questo caso, la variabile

DisplayProcedure viene mantenuta a un valore alto per un massimo di millisecondi indicato nella costante *PopUp.ProcedurePopUpTime*.

Se la pagina da visualizzare cambia, la variabile può tornare a true. In questo modo, se la procedura di accensione si blocca, la pagina corrispondente alla fase non rimarrà fissa sullo schermo ma scomparirà.

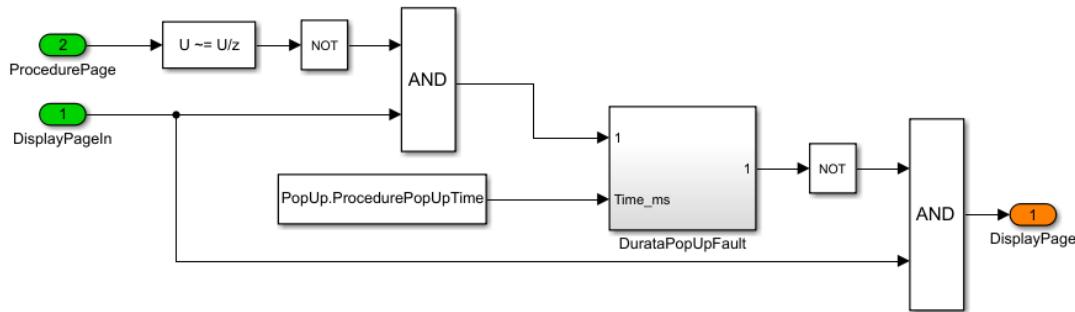


Figura 86 Interno del blocco di timing visualizzazione popup

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

Figura 87 Valori assunti dalla variabile PageProcedure

Gestione LED

Questo blocco è responsabile di stabilire quale LED debba essere acceso o spento. I LED indicano sempre lo stato di carica della batteria, tranne durante la procedura di accensione, in cui rappresentano invece la tensione della batteria. Ciò serve ad avere un indicatore visivo del corretto funzionamento della fase di precarica, in cui la tensione del sistema di trazione aumenta gradualmente fino a raggiungere la tensione dell'accumulatore.

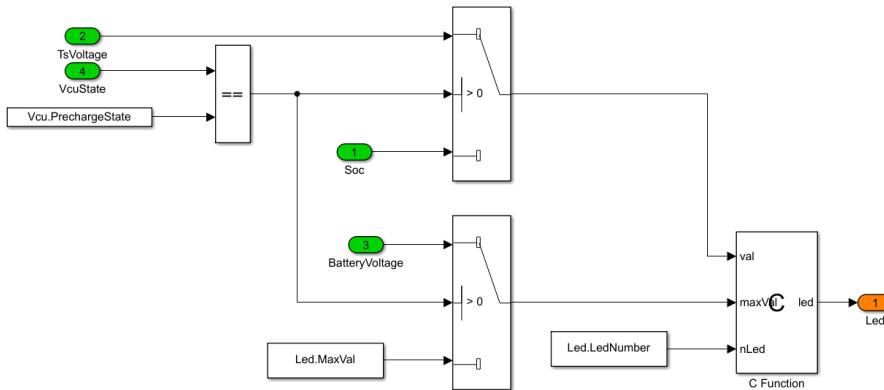


Figura 88 Interno del blocco per l'accensione dei led

L'uscita di questo blocco è rappresentata dalla variabile `uint16 Led`, i cui primi 10 bit indicano ogni singolo LED. Se il bit corrispondente è 1, il LED sarà acceso, altrimenti sarà spento.

La scrittura dei singoli bit è gestita da un blocco particolare chiamato “C function” che permette di integrare un algoritmo scritto in C (o C++) come blocco Simulink.

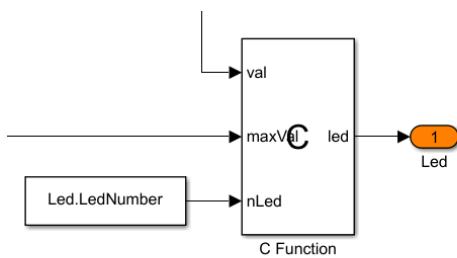


Figura 90 blocco "C function"

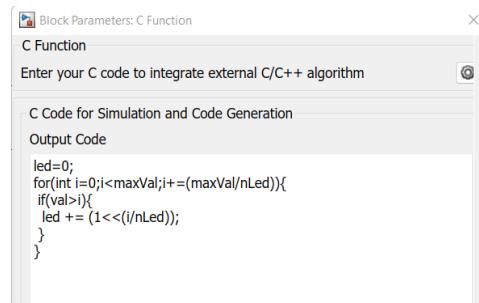


Figura 89 Algoritmo accensione LED

Questo semplice algoritmo riceve come input il valore massimo rappresentabile dai LED e il numero di LED presenti. In questo modo, è possibile associare ad ogni singolo LED un intervallo di valori. Quando il valore corrente si trova in uno specifico intervallo, il bit corrispondente verrà impostato a 1, così come tutti quelli degli intervalli precedenti.

Il blocco è opportunamente parametrizzato in modo da risultare indipendente dal numero di LED e dal valore che essi rappresentano. Ciò consente di apportare modifiche future al numero di LED o al valore rappresentato senza dover modificare l'intero blocco, ma solo i parametri corrispondenti.

7.3.3 Elaborazione degli output

Elaborazione output video

I dati da stampare a video vengono raggruppati in array di uint16 e suddivisi per pagina. Questi array sono parametrizzati sulla base della loro dimensione e dell'indice di ciascun dato all'interno di essi. Questo metodo consente di ridurre notevolmente la quantità di codice richiesta dalle librerie del display, in quanto esse ricevono solo un array, la sua dimensione e l'indicazione di quali dati sono contenuti in quale indice. Il risultato è un semplice ciclo for che scriverà sulla porta seriale. È evidente che questo approccio aumenta leggermente l'accoppiamento tra i due moduli, ma questo non costituisce un problema, in quanto le modifiche necessarie in caso di aggiunta o cambiamenti drastici delle pagine sono minime e molto veloci.

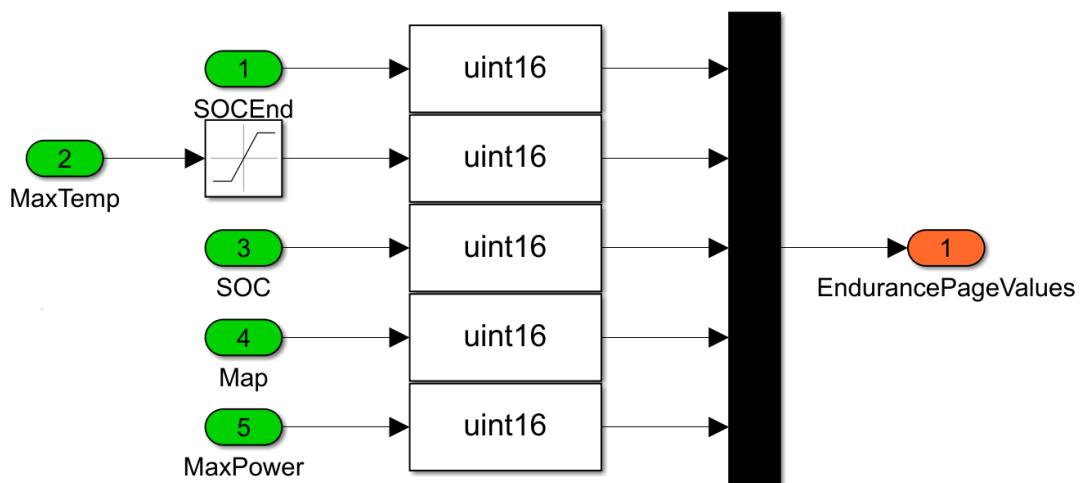


Figura 91 Generazione array contenente i valori di una singola pagina

Elaborazione output CAN bus.

L'unico messaggio CAN che viene emesso dal volante ha l'ID 0x401 ed ha lo scopo di comunicare alla VCU gli input del pilota. Infatti, il volante è responsabile soltanto di acquisire gli input e trasmetterli alla centralina che si occupa di effettuare i cambiamenti. Una volta che i cambiamenti sono stati effettuati, la centralina li comunica sempre tramite il CAN bus al volante, che li visualizza a video.

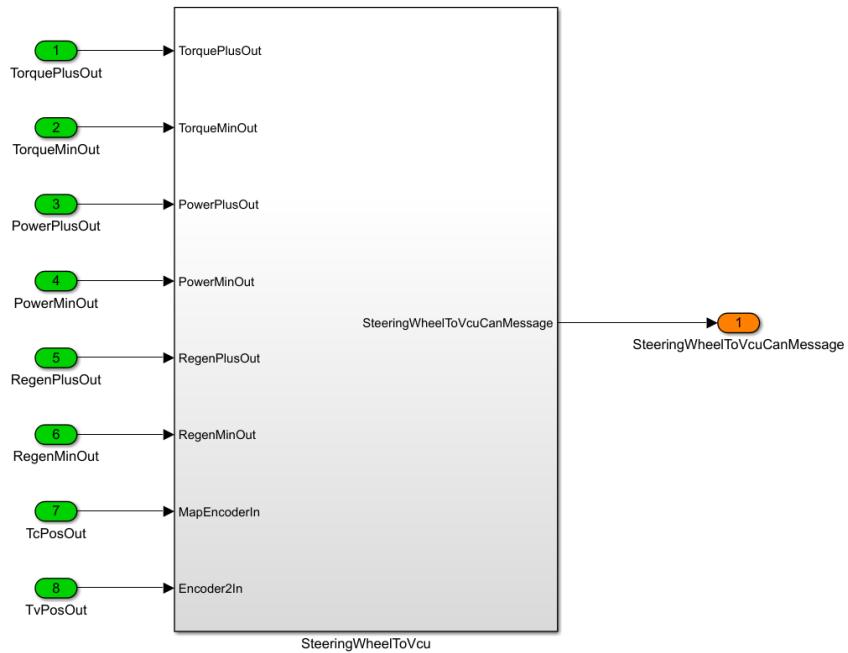


Figura 92 Impacchettamento del messaggio SteeringWheelInfo

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

Figura 93 Struttura messaggio SteeringWheelInfo

7.4 Librerie display

Le librerie del display sono costituite dal file “display.h” e “display.c”. Nel primo non sono contenute solo le funzioni implementate nel display.c ma anche tutte quelle costanti che servono per impartire comandi al Nextion, verranno analizzate in dettaglio più avanti.

Come visto in precedenza il TIM10 si occupa di richiamare con una frequenza di 30Hz la funzione **updateDisplay**. Questa è l'unica funzione richiamata dall'esterno della libreria, infatti se verranno apportate modifiche alla struttura grafica o dei dati del display verranno modificate le funzioni interne alle librerie senza che gli altri moduli se ne accorgano. Ovviamente queste sono in possesso di un riferimento all'oggetto di tipo **ExtY_SteeringWheel_T** che contiene gli output della strategia.

7.4.1 Visualizzazione Popup

Per prima cosa in **updateDisplay** vengono controllate le variabili booleane che rappresentano se è necessario o meno la visualizzazione del pop up. Questi hanno una priorità. Infatti, in caso di conflitto (più variabili DisplayPopup a true) viene seguito il seguente ordine:

FAULT -> WARNING -> PROCEDURE -> 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);
    }
}
```

Figura 94 Funzione updateDisplay

Prendendo d'esempio la funzione **displayProcedurePage** vediamo come avviene più nel dettaglio la visualizzazione di un Popup.

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

Figura 95 Funzione displayProcedurePage

Con **sprintf()** andiamo a scrivere nel buffer la stringa che invieremo via seriale al display. Questa è composta da:

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

Figura 96 Contenuto stringa INFO_PAGE_S

```
const char *EDIT PROCEDURE STRING = "info.txt="";
```

Figura 97 Contenuto della stringa EDIT PROCEDURE STRING

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

Figura 98 Contenuto dell'array PROCEDURE_STRING[]

L'indice da selezionare nell'array PROCEDURE_STRING è dettato dall'output della strategia **SteeringWheel_Y.PageProcedure**. Descritto nel capitolo 7.3.2

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

Figura 99 Carattere fine stringa

In unicode 0xFF corrisponde a ÿ.

Basandosi sulla documentazione ufficiale del NextionDisplay alla fine di ogni comando bisogna aggiungere per tre volte il carattere ÿ. Questo è necessario per far capire al display che il comando è terminato e ne sta per iniziare un nuovo.

Se per esempio la strategia stabilirà la visualizzazione della pagina di fine procedura (RTD) la stringa che varrà inviata su seriale sarà la seguente:

InfoPage.info.txt="RTD" ÿ ÿ ÿ

Inviando questa stringa su seriale, attraverso la funzione **HAL_UART_TRANSMIT**, andremo a cambiare il testo visualizzata nel popup, è importante notare che la modifica del testo viene effettuata prima del cambio di pagina, in questo modo si evita di visualizzare per un istante il testo precedente, questo oltre che a essere brutto esteticamente risulta una potenziale distrazione per il pilota.

7.4.2 Cambio pagina

Il cambio di pagina è effettuato grazie alla funzione **changeSteeringWheelPage**.

```
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();  
}
```

Figura 100 Funzione per il cambio di pagina

Seguendo ragionamenti analoghi a quelli fatti in precedenza verrà inviata su seriale la seguente stringa:

page InfoPage ÿ ÿ ÿ

7.4.3 Aggiornamento pagina

Ogni volta che viene richiamata la funzione **updateDisplay** o viene cambiata pagina grazie alla funzione **updateSteeringWheelPage** viene aggiornata la variabile *steeringWheelPage* dove è tenuta traccia della pagina realmente visualizzata in quell'istante di tempo sul display. Questo è possibile farlo inviando su seriale il comando *sendme* che restituirà il numero della pagina corrente.

```
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;  
}
```

Figura 101 Funzione per ottenere la pagina corrente visualizzata a schermo

7.4.4 Aggiornamento valori pagina

Prendendo d'esempio la pagina dell'Endurance come già fatto per la strategia vediamo come avviene l'aggiornamento dei valori.

Grazie alla variabile *steeringWheelPage* sono a conoscenza di quale sia la pagina visualizzata così da poter aggiornare solamente i valori di questa.

```

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]);
    }
}

```

Figura 102 Funzione per l'aggiornamento dei dati della pagina Endurance

Ciclando l'output della strategia *SteeringWheel_Y.EndurancePageValues* posso accedere ai singoli valori e attraverso l'array *ENDURANCE_PAGE_STRING* posso accedere alla parola chiave da utilizzare nel messaggio per modificare il valore specifico.

```

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);
}

```

Figura 103 Funzione per aggiornare un dato visualizzato a schermo

La funzione **updateDisplayValues** si limita a inviare un messaggio di modifica tramite seriale. Se si ipotizza che il valore che si sta aggiornando sia quello dello stato di carica (State Of Charge o SOC) il messaggio inviato risulta essere il seguente:

SOC.val=100 ÿ ÿ ÿ

7.4.5 Cambio dei colori

Senza entrare nel dettaglio delle singole funzioni per evitare di ripetere gli stessi concetti visti in precedenza vediamo solamente quale stringa bisogna inviare su seriale per cambiare il colore di un oggetto x.

- Stringa per cambiare lo sfondo nel colore rosso: x.bco=RED ÿ ÿ ÿ

- Stringa per cambiare il colore dei caratteri in rosso: x.pco=RED ÿ ÿ ÿ

8 Conclusioni

In questo lavoro di tesi sono stati descritti tutti i processi di sviluppo dell'hardware e del software del volante della vettura PSR02-S.

Uno degli obiettivi era quello di poter dare in mano al pilota uno strumento per monitorare i parametri fondamentali della vettura durante le gare e i test, e la possibilità attraverso pulsanti e manettini di modificare i settaggi in maniera semplice, rapida e precisa. L'obiettivo è stato raggiunto dando la possibilità di modificare la potenza erogabile dal pacco batteria, la coppia motrice, la coppia rigenerativa e aumentare e diminuire l'invasività del controllo di trazione e del torque vectoring.

Un aspetto da non sottovalutare era quello di creare un software non solo funzionante, ma anche facilmente comprensibile dai membri del reparto Software & Vehicle Control e soprattutto facilmente mantenibile, in modo da fungere da punto di partenza per gli sviluppi futuri. Anche questo obiettivo è stato raggiunto utilizzando la stessa struttura del software che caratterizza VCU e BMS. Il firmware è stato scritto in linguaggio C. L'elaborazione dei dati viene effettuata con Matlab/Simulink, da cui viene poi autogenerato il codice da integrare sottoforma di librerie nel firmware. Grazie a questa scelta, anche i membri che si trovano a dover effettuare modifiche per la prima volta sul progetto saranno in grado di orientarsi con facilità, data la struttura già familiare.

La realizzazione di un circuito stampato specifico ha consentito di ottimizzare lo spazio e ridurre il peso dell'intero volante a circa 400g, un aspetto cruciale nel campo delle corse automobilistiche. Inoltre, questa scelta ha permesso l'utilizzo di un microcontrollore della famiglia STM32F4 già impiegato su VCU e BMS, consentendo così l'implementazione della struttura software menzionata in precedenza.

Il volante descritto in questa tesi è stato utilizzato durante la competizione Formula Student Netherlands, e durante Formula Student Czech in cui l'Unipr Racing Team si

è piazzato per la prima volta nella sua storia nelle prime dieci posizioni in una gara estera.

9 Ringraziamenti

Per prima cosa ci tengo a ringraziare il mio relatore di tesi, il professor Carlo Concari che da sempre sostiene l’Unipr Racing Team ma soprattutto ha a cuore i ragazzi che ne fanno parte e si è sempre dimostrato disponibile e propositivo nei nostri confronti anche quando non era tenuto a farlo, avrebbe avuto un posto nei miei ringraziamenti anche se non fosse stato il relatore della mia tesi.

Ringrazio mia madre che da sempre si prende cura di me e mi sostiene dandomi l’affetto incondizionato che mi porto sempre dentro, la ringrazio soprattutto per avermi trasmesso la tenacia e la fame che mi spinge a dare il meglio di me in ogni cosa che faccio. Questo piccolo traguardo è anche tuo.

Ringrazio mio padre che non si è mai perso una mia partita di calcio fino a quando ho giocato. Lo ringrazio soprattutto per ogni volta che mi ha giudicato in modo oggettivo spingendomi a migliorarmi ogni giorno. Questo piccolo traguardo è anche tuo.

Ringrazio mia sorella, la mia metà. La persona il cui consiglio vale per me più di ogni altro. Anche se spesso non lo dimostro ti voglio un bene dell’anima e ho la massima fiducia nelle tue capacità. Quando sarai arrivata ricordati del tuo fratellino.

Ringrazio il professor Marinoni e il professor Camuso, i cui insegnamenti mi hanno trasmesso la passione per l’informatica e dato una preparazione tale da poter affrontare ogni esame di programmazione con “la pipa”, senza di loro non avrei mai intrapreso questo percorso.

Ringrazio tutti i ragazzi dell’Unipr Racing Team per aver reso gli ultimi due anni di università indimenticabili, uno su tutti Davide Draghi che mi ha mostrato che si può essere l’ingegnere informatico migliore d’Italia pur rimanendo umile. Menzione onorevole a Pietro Canuti per essere l’ingegnere meccanico migliore d’Italia ma anche il più divertente (divertente l’ho usato perché questa è una tesi universitaria, se no avrei usato un’altra parola).

Ringrazio gli EVIL con cui da sempre condivido il mio tempo libero, senza di voi sarei diventato una persona troppo seria. Ringrazio in particolare Fly per avermi fatto

compagnia durante le infinite ore di studio, e ringrazio Ghirlo per alzare sempre l'asticella e spingermi a dare sempre qualcosa in più.

Ringrazio Tia per essermi stato vicino nei momenti difficili della mia vita mettendo da parte i suoi pur di aiutarmi.

Ringrazio Chiara, che anche se abbiamo preso strade diverse ha contribuito a farmi mettere la testa a posto e tenermi sulla buona strada.

Bibliografia

Altium Designer - getting started. (s.d.). Tratto da <https://my.altium.com/altium-designer/getting-started>

CAN in Automation. (s.d.). Tratto da www.can-cia.org: <https://www.can-cia.org>

Datasheet CAN transceiver SN65HVD230MDREP. (s.d.). Tratto da 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 manettini. (s.d.). Tratto da <https://www.mouser.it/datasheet/2/26/PT65%20Series%20Rotary%20code%202018-1314458.pdf>

Datasheet MCP23017/MCP23S17. (s.d.). Tratto da <https://www.mouser.com/datasheet/2/268/20001952C-1920511.pdf>

Datasheet VLCS5130. (s.d.). Tratto da <https://www.mouser.it/datasheet/2/427/vlcs5130-1767005.pdf>

Datasheets - NX4832T035. (s.d.). Tratto da [nextion.tech: https://nextion.tech/datasheets/nx4832t035/](https://nextion.tech/datasheets/nx4832t035/)

Instruction Set. (s.d.). Tratto da nextion.tech: <https://nextion.tech/instruction-set/>

Librerie open source mcp23017. (s.d.). Tratto da [github.com: https://github.com/ruda/mcp23017](https://github.com/ruda/mcp23017)

MathWork - Embedded coder. (s.d.). Tratto da <https://it.mathworks.com/products/embedded-coder.html>

MATLAB Coder. (s.d.). Tratto da <https://it.mathworks.com/products/matlab-coder.html>

PCB capabilities. (s.d.). Tratto da <https://jlpcb.com/capabilities/pcb-capabilities>

Regolamento *Formula* *Student* 2022. (s.d.). Tratto da
https://formulastudent.de/fileadmin/user_upload/all/2022/rules/FS-Rules_2022_v0.9.pdf

STM32F412RE. (s.d.). Tratto da [www.st.com:](https://www.st.com/en/microcontrollers-microprocessors/stm32f412re.html)
<https://www.st.com/en/microcontrollers-microprocessors/stm32f412re.html>