Politecnico di Torino

Corso di Laurea Magistrale in

"Ingegneria Informatica"

TESI DI LAUREA

Sviluppo di un'interfaccia per la gestione di un cruscotto per veicoli elettrici

Laureando:

Angelo Marcianò

Relatore:

Ch.mo Prof. Stefano Carabelli

ANNO ACCADEMICO 2020/2021

Politecnico di Torino

Corso di Laurea Magistrale in

"Ingegneria Informatica"

TESI DI LAUREA

Sviluppo di un'interfaccia per la gestione di un cruscotto per veicoli elettrici

Laureando:

Angelo Marcianò

Relatore:

Ch.mo Prof. Stefano Carabelli

ANNO ACCADEMICO 2020/2021

Voglio rivolgere un sentito grazie al Prof. Stefano Carabelli per la sua disponibilità e la sua preziosa guida, che non hanno mai soffocato in me la curiosità e l'interesse, ma accresciuto sempre più il bisogno di conoscenza e il desiderio di ricerca, operatività e sperimentazione.

Sento anche il dovere di ringraziare la mia famiglia, in particolare i miei genitori, che con il loro sostegno e la loro costruttiva presenza, mi hanno permesso di diventare la persona che sono e di raggiungere questo traguardo.

Un grazie speciale va ad una persona a me tanto cara, la nonna Pina, che fino a qualche giorno fa mi ha incoraggiato, dandomi sempre fiducia e dimostrandomi il suo affetto e che adesso mi protegge, mi benedice e mi guarda da lassù.

Abstract - Italiano

La presente trattazione ha come campo d'indagine la gestione di sistemi di acquisizione, elaborazione e utilizzo dei dati di veicoli elettrici.

Da qualunque veicolo, grazie all'uso di sensori e trasduttori, è possibile acquisire segnali di vario tipo: elettrico, fisico e meccanico. Tali segnali forniscono informazioni che possono essere visualizzate dall'utilizzatore del mezzo di trasporto al fine di ottenere preziose indicazioni sullo stato del veicolo, sia da fermo sia in movimento. Trattando ed elaborando opportunamente i dati acquisiti è possibile, inoltre, generare messaggi di *alert* o, in determinate casistiche, limitare alcune funzionalità o addirittura non consentire totalmente l'utilizzo del mezzo, permettendo di perseguire due obiettivi: salvaguardare l'integrità del mezzo stesso ed evitare di mettere in pericolo la vita delle persone.

L'elaborazione dei segnali in ingresso e in uscita può essere gestita servendosi di schede programmabili con microcontrollore, capaci di interfacciarsi con il veicolo, connettendosi alla sua centralina.

La scelta di sistemi con "*MicroController Unit*" (MCU) è dettata da una molteplicità di fattori: la semplicità di programmazione, la facile gestione, l'assenza di un sistema operativo, i ridotti consumi energetici e il basso costo di acquisto.

Il software sviluppato e l'hardware utilizzato risultano robusti, ma allo stesso tempo versatili, infatti riescono a garantire: stabilità, sicurezza, possibilità di ampliamento con modifiche e sviluppi futuri. E' proprio grazie a tali caratteristiche che l'intero sistema potrà essere installato su una vasta gamma di veicoli commerciali.

Abstract - English

This thesis deals with the management of systems for the acquisition, processing and use of electric vehicles data.

From any vehicle, thanks to the use of sensors and transducers, it is possible to acquire signals of various types: electrical, physical and mechanical. These signals provide information that can be displayed by the user of the means of transport in order to obtain valuable information on the status of the vehicle, both when stationary and in motion. Moreover, by suitably treating and processing the acquired data, it is possible to generate *alert* messages or, in certain cases, to limit some functions or even not to allow the use of the vehicle at all, thus permitting to reach two goals: to safeguard the integrity of the vehicle itself and to avoid endangering people's lives.

The processing of the input and output signals can be managed by using programmable boards with microcontroller, able to interface with the vehicle, connecting to its control unit.

The choice of systems with "*MicroController Unit*" (MCU) is dictated by a variety of factors: the simplicity of programming, the easy management, the absence of an operating system, the reduced energy consumption and the low purchase cost.

The software developed and the hardware used are robust, but at the same time versatile, in fact they can guarantee: stability, security, possibility of expansion with future changes and developments. It is thanks to these features that the entire system can be installed on a wide range of commercial vehicles.

Riassunto

Il lavoro di questa tesi ha come campo d'indagine la gestione di sistemi di acquisizione, elaborazione e utilizzo dei dati di veicoli elettrici.

Da qualunque veicolo, grazie all'uso di sensori e trasduttori, è possibile acquisire segnali di vario tipo: elettrico, fisico e meccanico. Tali segnali forniscono informazioni che possono essere visualizzate dall'utilizzatore del mezzo di trasporto al fine di ottenere preziose indicazioni sullo stato del veicolo, sia da fermo sia in movimento. Nello specifico sono stati acquisiti i dati prodotti da sensori posizionati su una *FIAT Panda 900* del 2000. Su tale vettura, completamente trasformata in elettrica, sfruttando buona parte della strumentazione già esistente, è stato installato un sistema di gestione, controllo e visualizzazione di tutti i parametri fondamentali.

Trattando ed elaborando opportunamente i dati acquisiti è stato possibile, inoltre, generare messaggi di *alert* o, in determinate casistiche, limitare alcune funzionalità o addirittura non consentire totalmente l'utilizzo del mezzo, permettendo di perseguire due obiettivi: salvaguardare l'integrità del mezzo stesso ed evitare di mettere in pericolo la vita delle persone.

L'elaborazione dei segnali in ingresso e in uscita è stata gestita servendosi di schede programmabili con microcontrollore, capaci di interfacciarsi con il veicolo. Nello specifico, in questa tesi, sono state utilizzate le schede **Arduino Mega 2560 Rev3** [1] e **NXP Freedom-K64F** [2].

La prima *board* è stata utilizzata come interfaccia per dati relativi a: temperatura, umidità, velocità, livello di carica della batteria e altri indicatori di segnalazione e/o pericolo. La seconda *board* è stata utilizzata, invece, sfruttando il suo accelerometro/magnetometro a sei assi (**FXOS8700CQ**), come inclinometro.

La scelta di sistemi con "*MicroController Unit*" (**MCU**) è stata dettata da una molteplicità di fattori: la semplicità di programmazione, la facile gestione, l'assenza di un sistema operativo, i ridotti consumi energetici e il basso costo di acquisto.

Il software progettato è stato generato interamente tramite l'*Embedded Coder* [3] di **MATLAB/Simulink**, sfruttando il *Simulink Support Package for Arduino* [4] e il *Simulink Coder™ Support Package for NXP™ FRDM-K64F* [5].

Dopo varie simulazioni su *Simulink*, il software è stato caricato sui sistemi con microcontrollore e sono state effettuate le varie connessioni fra i pin di output delle due *board* e il quadro strumenti della *FIAT Panda*, integrando il tutto con un display e con dei led di indicazione allo scopo di testare il prototipo del sistema completo.

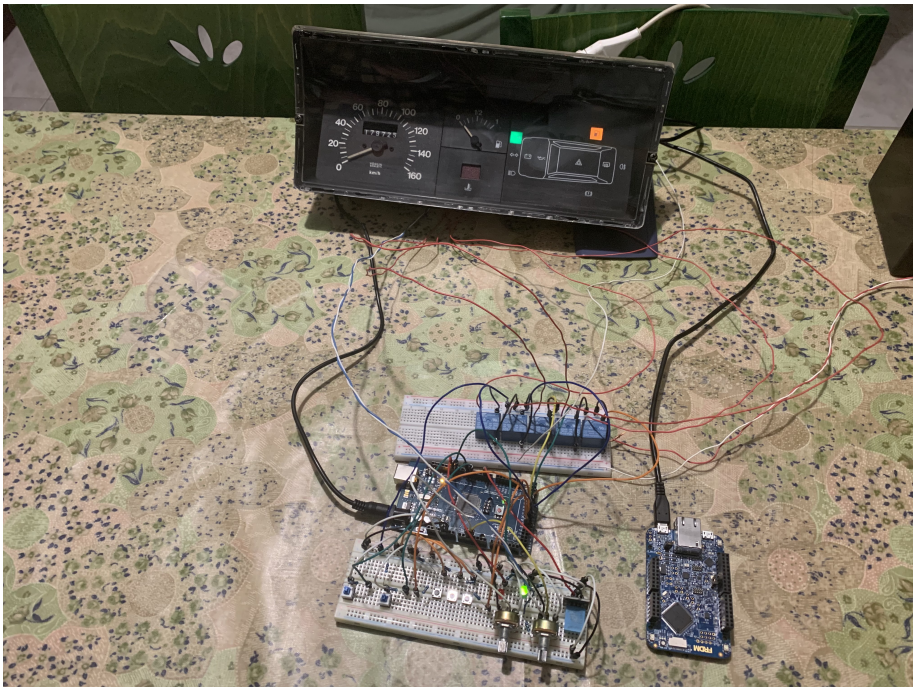


Fig. 1: Prototipo dimostrativo del sistema realizzato

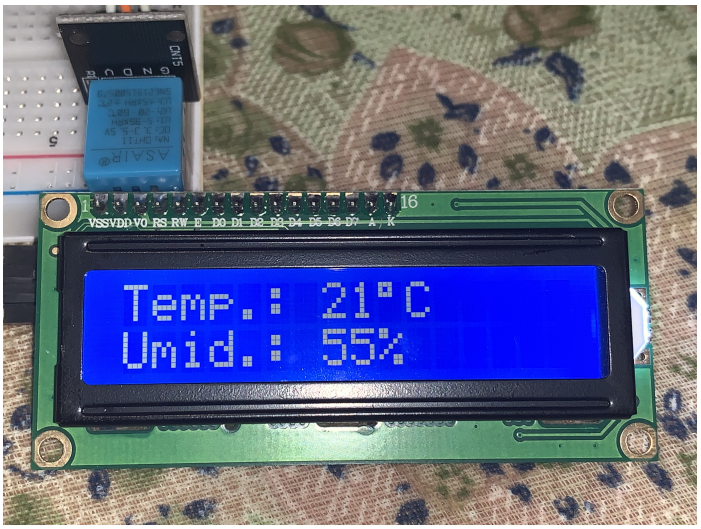


Fig. 2: Display LCD

Oltre ai sistemi di indicazione e *alert* tramite display, spie e led, è stata implementata, ad esempio, una funzionalità di sicurezza funzionale (*functional safety*) [6] per la gestione del cambio, capace di impedire il passaggio diretto da marcia avanti a marcia indietro o viceversa, quando l'auto viaggia ad una velocità pari o superiore a 5km/h. Allo stesso modo si potrebbero implementare funzioni di sicurezza similari come: impostare un avviso per la manutenzione programmata dopo un certo numero di chilometri o un certo intervallo di tempo; limitare la velocità del veicolo se la pressione degli pneumatici scende sotto una certa soglia; impedire la messa in moto se il livello di carica della batteria al litio è molto basso.

Il software sviluppato e l'hardware utilizzato risultano robusti, ma allo stesso tempo versatili, infatti riescono a garantire: stabilità, sicurezza, possibilità di ampliamento con modifiche e sviluppi futuri. E' proprio grazie a tali caratteristiche che l'intero sistema potrà essere installato su una vasta gamma di veicoli commerciali.

E' altresì opportuno precisare che, nel caso in cui si decida di installare il sistema su un mezzo dotato di centralina elettronica, sarà certamente possibile utilizzare i dati acquisiti ed elaborati per implementare ulteriori funzioni di notifica e/o sicurezza.

Indice

Capitolo I

Introduzione

L'acquisizione dei segnali relativi a tutti i parametri di qualsiasi tipo di veicolo si rivela di fondamentale importanza e di estrema necessità per consentirne il corretto utilizzo e garantire non solo la sicurezza e l'integrità del mezzo stesso, ma anche l'incolumità delle persone che si trovano a bordo o sulla strada.

Recentemente sono state condotte interessanti ricerche sul campo e vari sistemi sono stati già installati sui veicoli di fascia alta. Nelle utilitarie invece e, in generale, nei veicoli economici, tali dispositivi, se pur molto utili, non sono stati implementati per motivi legati al contenimento dei costi del prodotto finale.

Considerate le previsioni, che stimano il termine di produzione delle auto con motore termico fra circa dieci anni, con l'aumentare della diffusione dei veicoli elettrici, sarà sempre più importante prevenire ogni tipo di evento avverso come ad esempio il *thermal runaway* [7]. Nella fattispecie si tratta del fenomeno, per cui le batterie agli *ioni di litio* installate sulle autovetture elettriche (o ibride), possono surriscaldarsi al punto tale da innescare un incendio, che danneggerebbe irrimediabilmente il pacco batterie e l'intero veicolo.

Escludendo possibili difetti di fabbricazione, detto fenomeno, *thermal runaway*, potrebbe essere causato da:

- problematiche di tipo elettrico (eccessiva scarica della batteria, eccessiva carica della batteria, cortocircuito);
- problematiche di tipo termico (surriscaldamento del pacco batterie dovuto a cause interne o a cause esterne);
- urti e collisioni.

L'acquisizione, con opportuni sensori e trasduttori, di segnali di carattere elettrico, meccanico e dinamico fornisce informazioni, che possono essere visualizzate dall'utilizzatore del mezzo di trasporto, al fine di ottenere preziose indicazioni sullo stato del veicolo sia da fermo sia in movimento. Ciò renderebbe possibile il confronto delle stesse con dei valori limite di riferimento, per poter mettere in atto opportune tecnologie di protezione.

Pertanto, si rende necessario progettare dei sistemi di analisi e controllo finalizzati alla sicurezza e alla prevenzione più semplici ed economici, che garantiscano efficienza e versatilità in modo da essere installati su una vasta gamma di veicoli commerciali.

I.1 Struttura della tesi

Il resto della tesi è organizzato come segue:

- **Capitolo 2** presenta brevemente gli aspetti teorici sull'acquisizione, elaborazione ed utilizzo dei segnali.
- **Capitolo 3** discute la progettazione e lo sviluppo del software realizzato.
- **Capitolo 4** illustra la configurazione sperimentale di questa dissertazione, presenta la sperimentazione effettuata e analizza i risultati ottenuti.
- **Capitolo 5** conclude la tesi presentando i possibili sviluppi futuri e le eventuali opportunità di ampliamento del sistema realizzato.

Capitolo II

Aspetti teorici

I *sistemi embedded* sono sistemi di computer che appartengono a sistemi più grandi e ne soddisfano in parte i requisiti. Tra tali sistemi ricordiamo: i sistemi di controllo mobili per auto, i sistemi di controllo dei processi industriali, i telefoni cellulari o i controller di piccoli sensori. I *sistemi embedded* riguardano una vasta gamma di sistemi informatici, da dispositivi ultra piccoli basati su computer a grandi sistemi di monitoraggio e controllo di processi complessi. Oggi possiamo dire che quasi tutte le unità di calcolo, precisamente il 99%, appartengono a *sistemi embedded* [8].

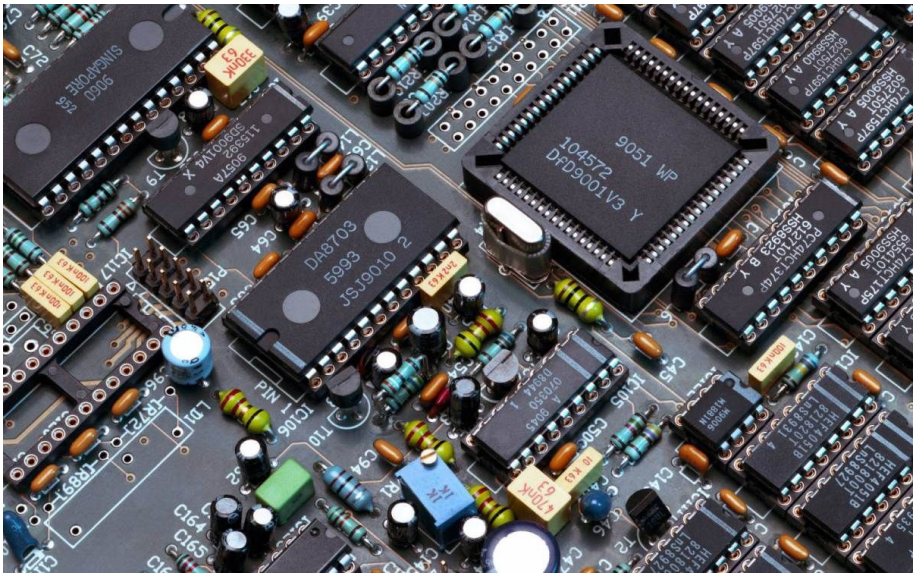


Fig. II.1: Ingrandimento sistema embedded

Questi sistemi sono tenuti a soddisfare in modo rigoroso specifiche di sicurezza, affidabilità, disponibilità e altri attributi di affidabilità [9]. La maggior parte di essi sono anche sistemi **real-time**, in quanto possiedono proprietà correlate al tempo reale, come il tempo di risposta o il tempo di esecuzione nel *worst case*, ecc. Dati i requisiti di mobilità, le dimensioni contenute e i costi di produzione molto bassi, questi sistemi richiedono un consumo ridotto e controllato di risorse e presentano capacità hardware limitate. La complessità, sempre più elevata dei *sistemi embedded* in tempo reale, porta a richieste maggiori rispetto all'ingegneria dei requisiti, alla progettazione di alto livello, al rilevamento precoce degli errori, alla produttività, all'integrazione, alla verifica e alla manutenzione, conseguentemente cresce la necessità di una gestione quanto più possibile efficiente delle proprietà relative al ciclo di vita come: la manutenibilità, la portabilità e l'adattabilità [10].

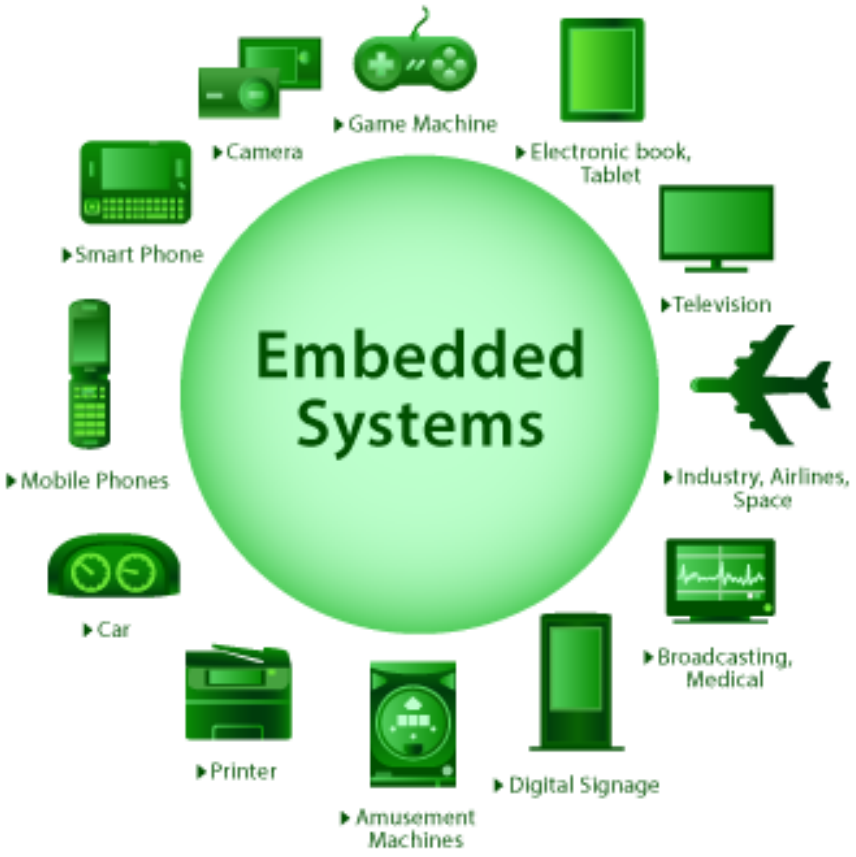


Fig. II.2: Vari sistemi embedded

II.1 Sistemi embedded per automotive

Nei sistemi automobilistici le attrezzature che utilizzano sistemi meccanici vengono sempre più sostituite da quelle che utilizzano sistemi elettronici. Il cuore del sistema elettronico di un veicolo è costituito dal *sistema embedded*, grazie alla sua versatilità e flessibilità.

L'elettronica ha di certo rivoluzionato il settore automobilistico, interessando: il design, la combustione del carburante, la protezione da varie tipologie di urti, ecc. L'uso avanzato del sistema integrato può senz'altro contribuire a controllare l'inquinamento, aumentando la possibilità di fornire funzioni di monitoraggio dei sistemi richiesti dai consumatori. Oggi un veicolo comune contiene circa 25-35 microcontrollori, mentre alcuni veicoli di lusso ne contengono circa 60-70 [11].

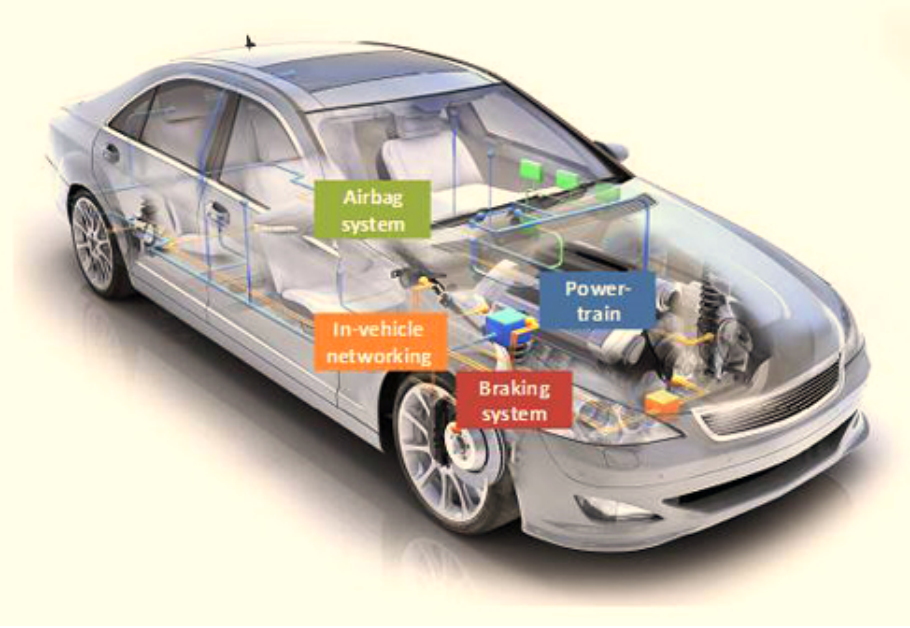


Fig. II.3: Sistemi embedded per automotive

Una qualsiasi automobile sulla strada ha sistemi elettronici controllati da computer; i *sistemi embedded* per *automotive* più largamente utilizzati sono:

- airbag;
- sistema di frenata antibloccaggio;
- scatola nera;
- radio satellitare;
- drive by wire;
- telematica;
- visione notturna;
- heads up display;
- cruise control adattivo;
- controllo delle emissioni;
- controllo della trazione;
- parcheggio automatico;
- sistemi di intrattenimento a bordo del veicolo;
- sensori di collisione posteriore;
- sistemi di navigazione;
- monitor della pressione degli pneumatici;
- controllo del clima;
- ecc.

II.2 ISO 26262

ISO 26262, "Veicoli stradali - Sicurezza funzionale", è uno standard internazionale definito nel 2011 e rivisto nel 2018 dall'Organizzazione Internazionale per la Standardizzazione (ISO). Si occupa della **sicurezza funzionale** dei sistemi elettrici e/o elettronici installati nei veicoli stradali di serie, relativamente a tutto il loro ciclo di vita [12].

Le caratteristiche di sicurezza funzionale, infatti, accompagnano tutte le fasi di sviluppo del veicolo, dalle specifiche alla progettazione, dall'implementazione all'integrazione, dalla verifica alla convalida e al rilascio in produzione.

Lo standard ISO 26262 è un adattamento dello standard di sicurezza funzionale *IEC 61508* e ha lo scopo di evitare gli eventuali pericoli derivanti dal malfunzionamento dei sistemi elettronici ed elettrici nei veicoli. Anche se denominata "Veicoli stradali - Sicurezza funzionale", la norma si riferisce non solo alla sicurezza funzionale dei sistemi elettrici ed elettronici, ma anche a quella dei sistemi nel loro insieme o dei loro sottosistemi meccanici.

Come la norma madre (IEC 61508), la **ISO 26262** è una norma di sicurezza sul rischio: in situazioni operative pericolose, il rischio viene valutato qualitativamente e le misure di sicurezza sono definite non solo per evitare o controllare i guasti sistematici, ma anche per rilevare o controllare i guasti casuali dell'hardware o mitigare i loro effetti [12].

Obiettivi di **ISO 26262**:

1. Fornire un ciclo di vita della sicurezza automobilistica (gestione, sviluppo, produzione, funzionamento, assistenza, smantellamento), supportando la scelta e l'adattamento delle attività richieste.

2. Coprire gli aspetti della sicurezza funzionale di tutto il processo di sviluppo (comprese attività come la specifica dei requisiti, la progettazione, l'implementazione, l'integrazione, la verifica, la convalida e la configurazione).

3. Fornire un approccio basato sul rischio specifico del settore automobilistico per determinare i livelli di rischio (livelli di integrità della sicurezza automobilistica, *ASIL*).

4. Utilizzare gli *ASIL* al fine di elencare i requisiti di sicurezza richiesti per raggiungere un rischio residuo accettabile.

5. Indicare i requisiti relativi alle misure di convalida e conferma per poter garantire un sufficiente e accettabile livello di sicurezza.

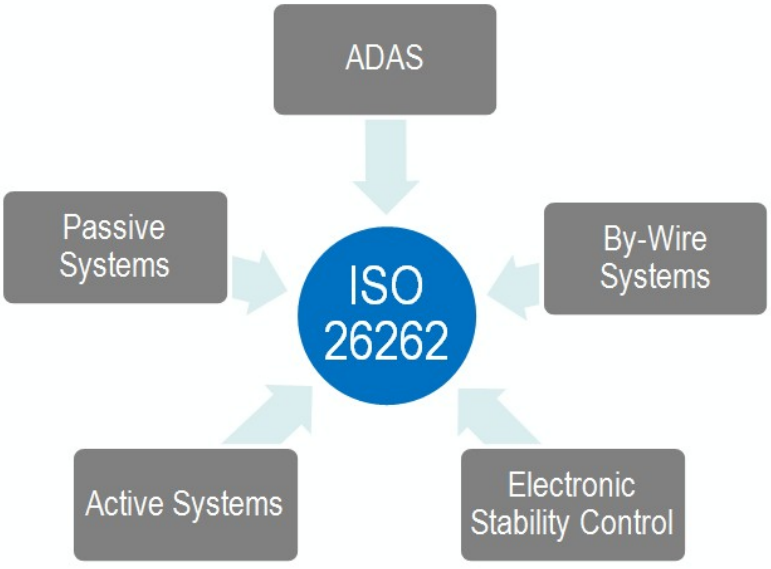


Fig. II.4: Obiettivi ISO 26262

II.3 AUTOSAR

AUTOSAR, AUTomotive Open System ARchitecture, è una partnership di sviluppo globale concernente il settore automobilistico. Fondata nel 2003 mirava a creare e a stabilire un'architettura software aperta e standardizzata per le unità di controllo elettronico (**ECU**) automobilistiche, perseguendo importanti obiettivi quali: la scalabilità a diverse varianti di veicoli e piattaforme, la trasferibilità del software, la considerazione dei requisiti di disponibilità e sicurezza, una collaborazione tra vari partner, un uso sostenibile delle risorse naturali e la manutenibilità durante l'intero ciclo di vita del prodotto.

AUTOSAR prefigge un insieme di specifiche, come: descrivere i moduli software di base, definire le interfacce applicative e costruire una metodologia di sviluppo comune basata su un formato di scambio standardizzato [13].

I moduli software di base, fruibili proprio grazie all'architettura software a strati **AUTOSAR**, si possono usare sia in veicoli di diversi produttori che in componenti elettronici di diversi fornitori. Ciò risulta vantaggioso non solo per ridurre le spese relative alla ricerca e allo sviluppo, ma anche per risolvere i problemi legati alla maggiore complessità delle architetture elettroniche automobilistiche e software correlati.

E' evidente che AUTOSAR è stata concepita per dar vita a sistemi elettronici innovativi atti a migliorare sempre più le prestazioni, la sicurezza e la compatibilità ambientale e per facilitare lo scambio e l'aggiornamento di software e hardware durante la vita del veicolo. La finalità di fondo, dunque, è quella di essere proiettati verso tecnologie all'avanguardia e di abbassare il livello dei costi senza compromettere la qualità.

II.4 Model-Driven Engineering (MDE)

Il *Model-Driven Engineering* (**MDE**) è un approccio per sviluppare il software riguardante i modelli, come gli artefatti principali creati e usati dai processi del ciclo di vita del software stesso [14]. Gli strumenti e le tecnologie abilitanti comprendono un ventaglio abbastanza vasto di capacità, rivelandosi perciò prezioso e indispensabile per gli sviluppatori, gli acquirenti e gli utenti finali.

MDE è un approccio che non usa il codice sorgente per svolgere la stragrande maggioranza dei compiti di ingegneria del software, ma direttamente i modelli.
Nello specifico, MDE si occupa dell'analisi statica di prototipazione rapida (architettura, completezza, correttezza), dell'analisi dinamica attraverso modelli eseguibili, dell'acquisizione della documentazione, del refactoring della documentazione ottenuta, della generazione del codice eseguibile, del test automatico e di altri *task*.

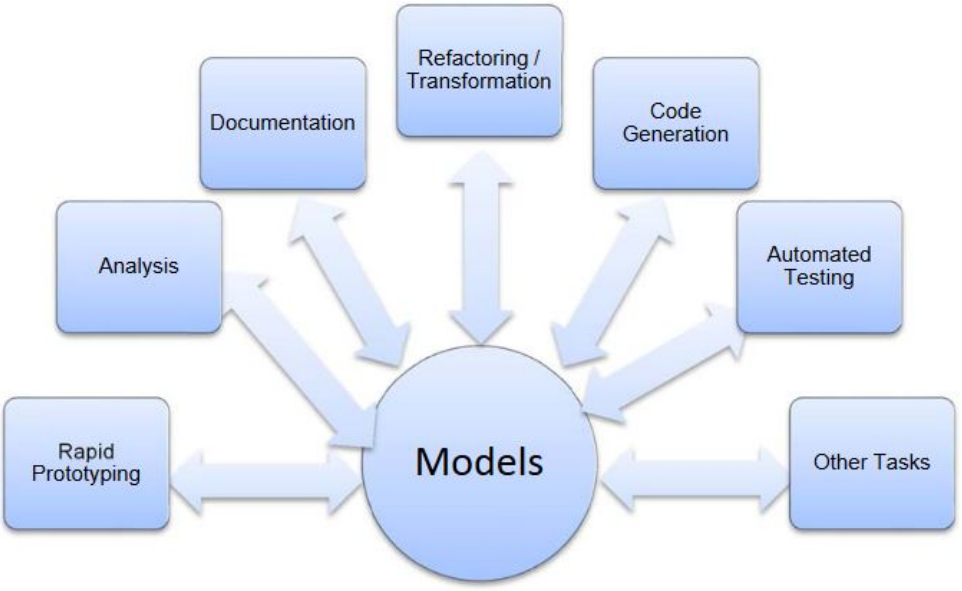


Fig. II.5: Model-Driven engineering

II.4.1 Simulink/Embedded coder

Simulink Coder™ (in passato chiamato Real-Time Workshop) genera ed esegue codice **C** e **C++**, utilizzando modelli **Simulink**, diagrammi **Stateflow** e funzioni **MATLAB**.

Il codice sorgente generato può essere impiegato sia per applicazioni in tempo reale che non *real time*, inclusi il *rapid prototyping* e il test dell'*hardware-in-the-loop* [15]. E' possibile sviluppare e monitorare il codice generato, sfruttando le potenzialità di *Simulink*; si può, inoltre, interagire con il codice sorgente eseguendolo al di fuori da *MATLAB* e *Simulink*.

Simulink Coder si serve di compilatori di terze parti atti a generare: eseguibili di controllori *embedded*, di sistemi autonomi e di sistemi fisici, simulati in tempo reale e non, sfruttando modelli creati con Simulink e, se necessario, ricorrendo a tool aggiuntivi.

Embedded Coder genera codice in linguaggio C e C++ leggibile, compatto e veloce destinato a processori embedded, necessari nella produzione di massa di dispositivi di vario genere. Estende *MATLAB Coder*™ e *Simulink Coder*™ riuscendo, grazie a procedure di altissimo livello, a controllare in modo rigoroso e puntuale funzioni, file e dati generati. In tal modo, risulta ottimizzata l'efficienza del codice e facilitata l'integrazione con il codice *legacy*, con i tipi di dati e con i parametri di calibrazione. Inoltre, inserendo uno strumento di sviluppo di terze parti, è possibile la costruzione di un eseguibile per distribuire il software su un *sistema embedded* o su una scheda di prototipazione rapida [16]. In definitiva è evidente la versatilità di *Embedded Coder* e la sua importanza per le molteplici possibilità di utilizzo. Esso infatti: fornisce supporto integrato per gli standard software **AUTOSAR**, MISRA C e ASAP2; offre rapporti di tracciabilità, documentazione del codice e verifica automatica del software, rivelandosi capace di sviluppare software adeguato agli standard DO-178, IEC 61508 e **ISO 26262**; fornisce pacchetti di supporto con ottimizzazioni avanzate e driver per dispositivi con hardware specifici; essendo portabile, può essere compilato ed eseguito su qualsiasi processore.

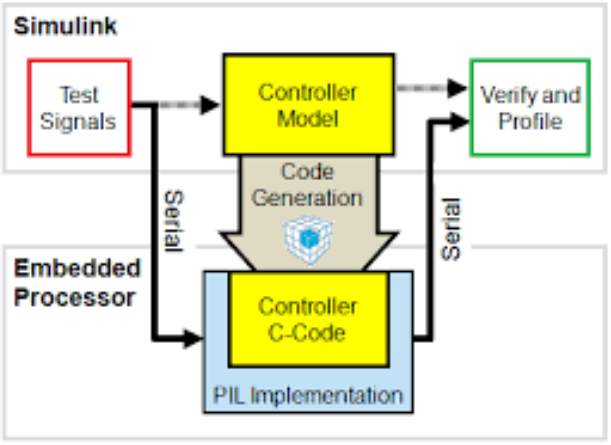


Fig. II.6: MATLAB/Simulink code generation

Con **MATLAB/Simulink Coder**, è possibile:

- Generare codice per file e funzioni **MATLAB**;
- Selezionare il processore e l'output di generazione del codice;
- Scegliere le ottimizzazioni di Embedded Coder.

Con **MATLAB/Embedded Coder**, è possibile:

- Generare codice per modelli e sottosistemi **Simulink**;
- Selezionare il processore e l'output della generazione del codice;
- Scegliere Embedded Coder per ottimizzare la RAM o la velocità di esecuzione.

II.5 Verifica e validazione del software

La verifica e la validazione (**V&V**) è il processo attraverso il quale viene effettuato il controllo del software, per rilevare se le specifiche sono state soddisfatte e se gli obiettivi prefissati sono stati raggiunti. E' pertanto comprensibile perché tale processo venga denominato: "controllo della qualità del software" [17] e perché venga considerato parte integrante del ciclo di vita dello sviluppo del software.

La verifica delle specifiche è indispensabile, e va effettuata non tramite l'esecuzione del software, ma attraverso l'analisi e l'esame dei suoi artefatti associati.

La validazione del software controlla se il prodotto soddisfi o si adatti all'uso previsto (controllo di alto livello), cioè se il software risponde o meno ai requisiti, come espressione dei bisogni di tutte le parti interessate, dagli utenti agli operatori, dagli amministratori ai manager, dagli investitori agli analisti.

Due sono le modalità per realizzarla: una interna e una esterna.

Nel primo caso si presume che gli obiettivi di tutte le parti coinvolte siano stati definiti in modo chiaro, inequivocabile e completo e che perciò possano essere correttamente compresi. Quando il software soddisfa le specifiche dei requisiti viene validato internamente.

Nel secondo caso la validazione avviene quando alle parti interessate viene chiesto se il software soddisfa i loro bisogni. A tal proposito esistono varie metodologie di sviluppo del software; tutte quante si basano su livelli diversi di coinvolgimento e feedback sia degli utenti che degli *stakeholder*. La validazione esterna si può rivelare come un fenomeno continuo o discreto. Quando tutti gli *stakeholder* accettano il prodotto software e riconoscono che rispecchia i loro bisogni [17], allora la validazione esterna finale ha avuto successo.

La rilevazione può avvenire sia tramite l'uso di test dinamici (test di accettazione), che attraverso test statici interni atti a far emergere se le specifiche dei requisiti siano state soddisfatte.

- La verifica del software avviene attraverso le seguenti fasi:

  1. Requirement gathering checking;
  2. Functional design testing;
  3. Internal system design testing;
  4. Module design testing.

- La validazione del software, invece, segue le seguenti fasi:

  1. Unit testing;
  2. Integration testing;
  3. System testing;
  4. Functional testing;
  5. User acceptance testing (UAT).

In Fig. II.7 è illustrato il processo di verifica e validazione (V&V).

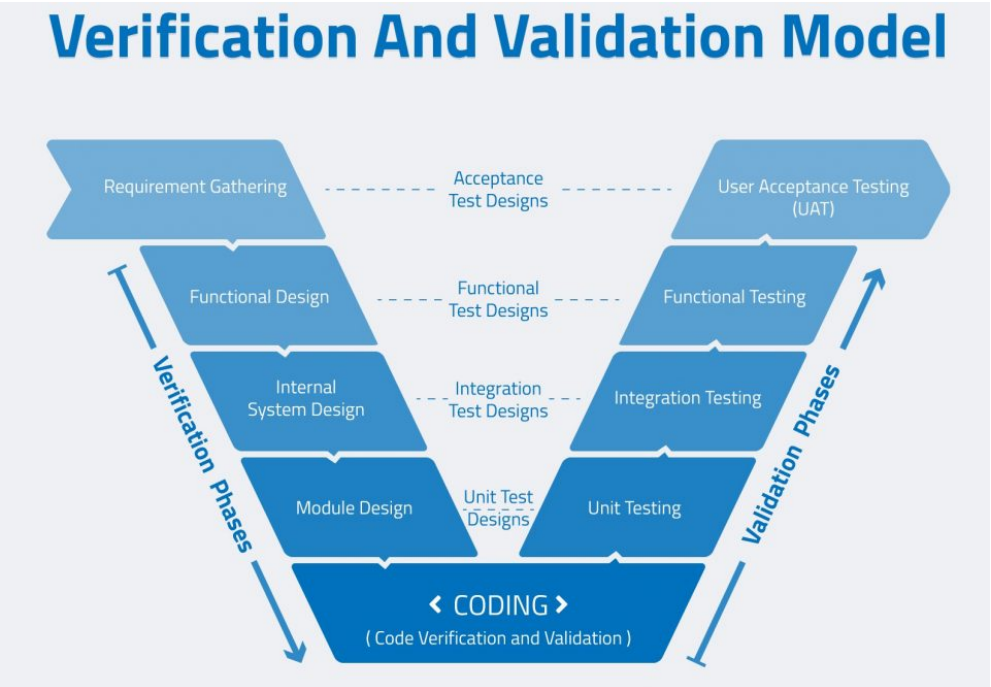


Fig. II.7: Verification & Validation flow (V&V)

Capitolo III

Progettazione e sviluppo

L'elaborazione dei segnali in ingresso e in uscita è stata gestita servendosi di schede programmabili con microcontrollore, capaci di interfacciarsi con il veicolo. Nello specifico, sono state utilizzate le schede **Arduino Mega 2560 Rev3** [1] e **NXP Freedom-K64F** [2].

Specifiche tecniche di **Arduino Mega 2560 Rev3**:

- Microcontrollore: ATmega2560
- Tensione operativa: 5V
- Tensione in input (raccomandata): 7-12V
- Tensione in input (limiti): 6-20V
- Pin di I/O digitali: 54 (di cui 15 output PWM)
- Pin di input analogici: 16
- Corrente DC per il pin I/O: 40 mA
- Corrente DC per il pin 3.3V: 50 mA
- Memoria flash: 256 KB di cui 8 KB usati dal bootloader
- SRAM: 8 KB
- EEPROM: 4 KB
- Frequenza di clock: 16 MHz

- LED_BUILTIN: 13
- Lunghezza: 101.52 mm
- Larghezza: 53.3 mm
- Peso: 37 g

Specifiche tecniche di **NXP Freedom-K64F**:

- Microcontrollore: MK64FN1M0VLL12
- Frequenza MCU: 120 MHz
- Memoria: 1 MB flash/256 KB RAM
- SD: 1 slot per scheda microSD (SDHC)
- USB: 2 porte USB micro-B
- Ethernet: 1 porta Ethernet
- Connettori I/O: R3: J3 e J4 compatibili con Arduino
- LED: 1 LED RGB
- Accelerometro e magnetometro FXOS8700CQ
- Debug: OpenSDAv2
- Pulsanti: 2 pulsanti
- Alimentatore: USB OpenSDAv2, USB Kinetis K64 e fonte esterna

La prima *board* è stata utilizzata come interfaccia per dati relativi a: temperatura, umidità, velocità, livello di carica della batteria e altri indicatori di segnalazione e/o pericolo. La seconda *board* è stata utilizzata, invece, sfruttando il suo accelerometro/magnetometro a sei assi (**FXOS8700CQ**), come inclinometro.

La scelta di sistemi con "*MicroController Unit*" (**MCU**) è stata dettata da una molteplicità di fattori: la semplicità di programmazione, la facile gestione, l'assenza di un sistema operativo, i ridotti consumi energetici e il basso costo di acquisto.

La progettazione del modello è stata effettuata sfruttando le potenzialità offerte dal software **MATLAB**.

Per necessità progettuali è stato necessario utilizzare i seguenti Add-On scaricabili dall'*Add-On Manager* di MATLAB:

- Simulink;
- Stateflow;
- MATLAB Coder;
- Simulink Coder;
- Embedded Coder;
- Simulink Coder Support Package for NXP FRDM-K64F Board;
- MATLAB Support Package for Arduino Hardware;
- Simulink Support Package for Arduino Hardware;
- Arduino Additional Sensors Library (DHT, LPS331);
- Simulink library for Arduino Liquid Crystal Display.

La scelta degli applicativi utilizzati per lo sviluppo dell'intero sistema è stata determinata: dalla possibilità di sfruttare la semplice interfaccia grafica, dalla possibilità di simulare e testare il sistema prima ancora di essere in possesso dei componenti hardware e del veicolo stesso, dalla scalabilità del sistema per eventuali modifiche e sviluppi sia in itinere che futuri.

Il modello è stato suddiviso in varie unità, una per ogni funzione. Ciascuna di esse è stata suddivisa in tre sotto-unità: signal acquisition unit, signal processing unit, output generation unit.

III.1 Inclinometro

Nel caso in cui il veicolo utilizzato sia un fuoristrada, è opportuno inserire un dispositivo in grado di segnalare un'eccessiva inclinazione rispetto al piano orizzontale, tale da fornire un valore limite, al di là del quale il veicolo potrebbe ribaltarsi.

L'unità Simulink per la gestione dell'inclinazione è suddivisa in 3 sotto-unità:

1. Input Acquisition Unit: si occupa dell'acquisizione dei dati grezzi dall'accelerometro (**FXOS8700CQ**) integrato sull' **FRDM-K64f**, ogni 100ms legge le triplette (Ax, Ay, Az) in g (range +/- $2g$);
2. Processing Unit: si occupa del processamento dei dati in ingresso;
3. Output Generation Unit: si occupa della generazione dei segnali di output (spie di indicazione).

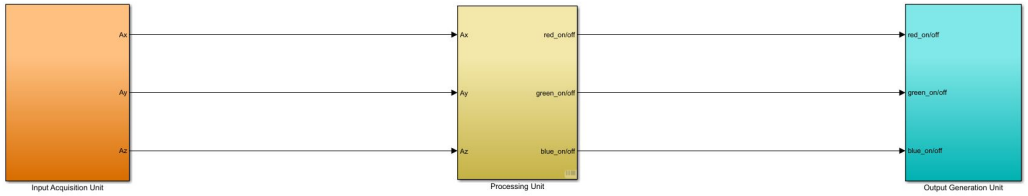


Fig. III.1: Inclinometer Subsystems

La *Processing Unit* è stata a sua volta suddivisa in 2 subsystem:

1. angles_processing: si occupa della generazione degli angoli di inclinazione;
2. output_processing: si occupa del pre-processamento dei segnali di output.

Fig. III.2: Inclinometer Processing Unit

Come si può notare nell'immagine III.3, l'*angles_processing* unit, dopo la fase di calibrazione, invia le accelerazioni [g] ad una *MATLAB function*.

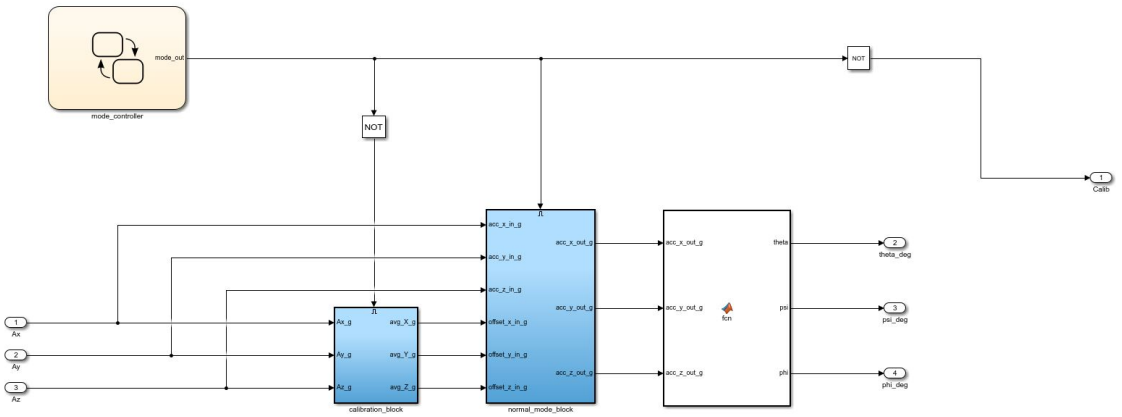
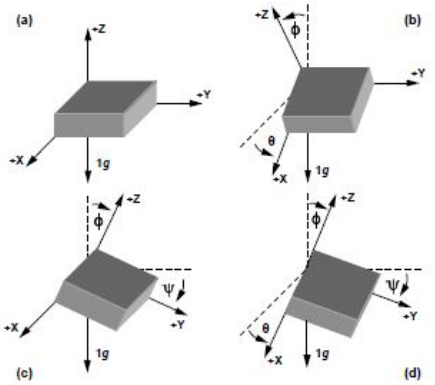


Fig. III.3: Inclinometer angle_processing

La funzione calcola gli angoli di inclinazione sfruttando le seguenti formule:



$$\theta = \tan^{-1}\left(\frac{A_{X,OUT}}{\sqrt{A^2{}_{Y,OUT} + A^2{}_{Z,OUT}}}\right)$$

$$\psi = \tan^{-1}\left(\frac{A_{Y,OUT}}{\sqrt{A^2{}_{X,OUT} + A^2{}_{Z,OUT}}}\right)$$

$$\phi = \tan^{-1}\left(\frac{\sqrt{A^2{}_{X,OUT} + A^2{}_{Y,OUT}}}{A_{Z,OUT}}\right)$$

Fig. III.4: Formule per il calcolo degli angoli di inclinazione

Tale funzione riceve come parametri in ingresso le accelerazioni lungo gli assi x, y e z in g (range +/- $2g$) e restituisce come output gli angoli *theta, psi, phi* in gradi.

La divisione per 0 è impossibile da risolvere, di conseguenza non gestire questa casistica potrebbe generare falsi segnali di output. La *MATLAB function*, dopo aver calcolato il denominatore in radianti, lo confronta con 0 e, nel caso in cui sia uguale, lo sostituisce con *eps()*: un valore molto piccolo, ma comunque diverso da 0.

Infine la funzione completa il calcolo degli angoli ed esegue la loro conversione in gradi, restituendo il valore di *theta, psi, phi*.

Nell'immagine sottostante è possibile osservare la funzione descritta:

```
function [theta, psi, phi] = fcn(acc_x_out_g, acc_y_out_g, acc_z_out_g)
% This function calculates the angles in degrees from the acceleration in g

den_theta = sqrt((acc_y_out_g.^2)+(acc_z_out_g.^2));
den_psi = sqrt((acc_x_out_g.^2)+(acc_z_out_g.^2));
den_phi = acc_z_out_g;

%The following block of code is used to avoid the division by 0 problem
%If the denominator == 0, set the denominator to a very small value
if (den_theta == 0)
    den_theta = eps();
end
if (den_psi == 0)
    den_psi = eps();
end
if (den_phi == 0)
    den_phi = eps();
end

theta = rad2deg(atan(acc_x_out_g/den_theta));
psi = rad2deg(atan(acc_y_out_g/den_psi));
phi = rad2deg(atan(sqrt((acc_x_out_g.^2)+(acc_y_out_g.^2))/den_phi));
```

Fig. III.5: MATLAB funcion per il calcolo degli angoli di inclinazione

In seguito l'*output_processing* unit riceve i valori degli angoli in gradi, ne calcola il valore assoluto, li confronta con un valore soglia e li invia ad uno *Stateflow Diagram*:

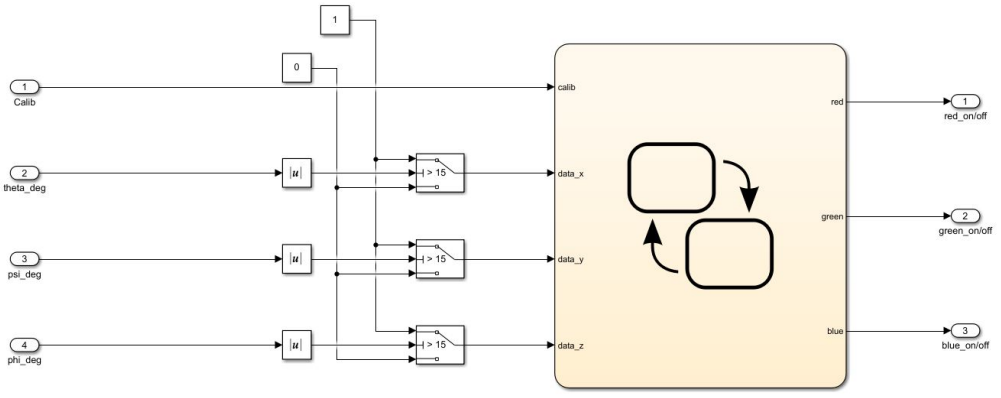


Fig. III.6: Inclinometer output_processing unit

Lo *Stateflow Diagram* inizia settando i tre led di indicazione utilizzati a 0 (spenti). Durante la fase di calibrazione viene acceso un led, mentre gli altri due rimangono spenti. Terminata questa fase, viene generato un *warning* se si verifica almeno una delle seguenti condizioni:

- L'angolo di inclinazione rispetto a X (*theta*) è maggiore di +/- 15 gradi;
- L'angolo di inclinazione rispetto a Y (*psi*) è maggiore di +/- 15 gradi;
- L'angolo di inclinazione rispetto a Z (*phi*) è maggiore di +/- 15 gradi.

In Fig. III.7 è possibile osservare il diagramma descritto:

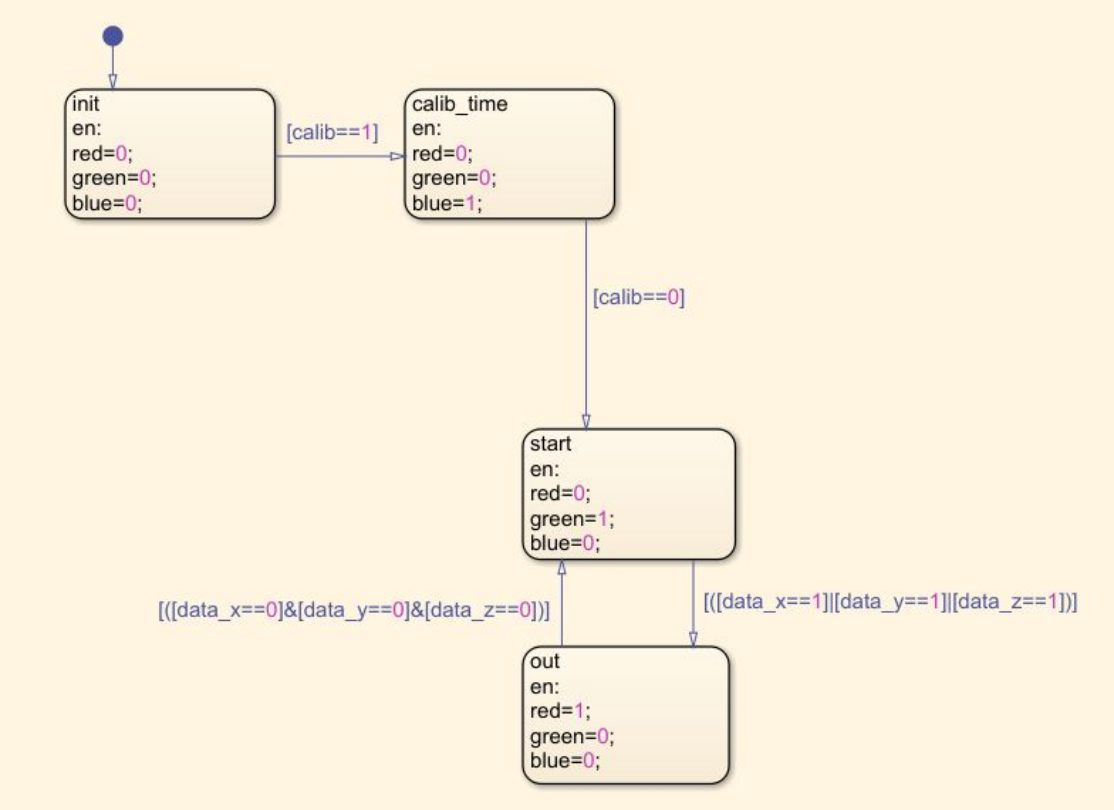


Fig. III.7: Stateflow Diagram output_processing unit

Infine il segnale generato viene inviato all' *Output Generation Unit*, che accenderà il led blue durante la calibrazione, il led rosso in caso di *warning* e il led verde se i valori di inclinazione rientrano nei limiti prestabiliti.

III.2 Cambio

Una delle funzioni che merita particolare attenzione è la gestione del cambio. Il veicolo, essendo elettrico, è dotato di marcia avanti, marcia indietro e posizione di folle.

L'unità Simulink per la funzione cambio è suddivisa in 3 sotto-unità:

1. Trasmission Input Acquisition Unit: si occupa dell'acquisizione della velocità istantanea del veicolo e delle posizioni della leva del cambio;
2. Trasmission Processing Unit: si occupa del processamento dei dati in ingresso;
3. Trasmission Output Generation Unit: si occupa della generazione dei segnali di output (led di indicazione).

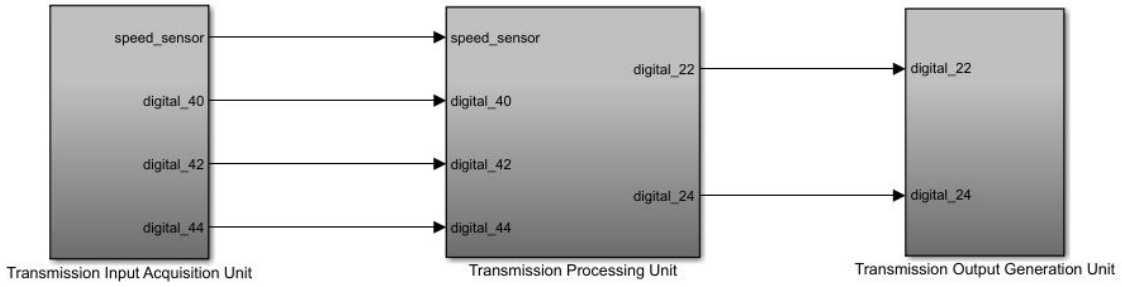


Fig. III.8: Trasmission Subsystem

Nell'immagine seguente è possibile osservare il contenuto del subsystem centrale:

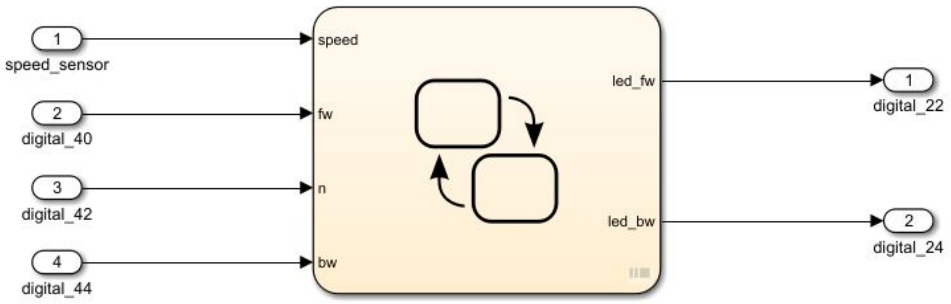


Fig. III.9: Trasmission Processing Unit

Come mostrato in figura III.9, i dati di input sono stati processati utilizzando uno *Stateflow Diagram*.

Sono stati progettati tre stati corrispondenti alle posizioni della leva del cambio:

- *forward*;
- *neutral*;
- *backward*.

Di norma, il passaggio fra i vari stati avviene in seguito allo spostamento della leva e ciò è possibile, senza limitazione alcuna, per le transizioni da neutral a forward (o viceversa) e da neutral a backward (o viceversa).

Nel caso in cui si volesse passare da marcia avanti a marcia indietro (o viceversa), viene effettuato un controllo sulla velocità del veicolo. Se la velocità è inferiore ai 5km/h viene consentito il cambio di marcia, se superiore o uguale a 5km/h non viene dato il consenso per il cambio della marcia. Questa scelta implementativa di *functional security* è stata adottata per ovvi motivi di sicurezza, in quanto previene eventuali danni al veicolo in sé, agli occupanti e a chi si trova nei pressi dell'auto.

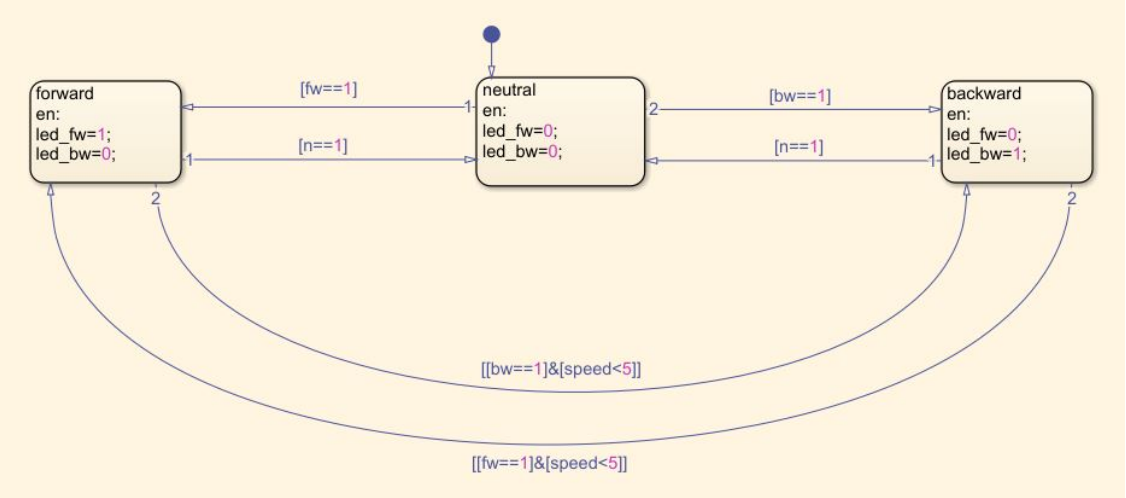


Fig. III.10: Stateflow Diagram Processing Unit

III.3 Livello batteria

In qualunque veicolo elettrico è di fondamentale importanza il controllo del livello di carica della batteria principale.

L'unità Simulink per la gestione del livello di carica del pacco batterie è suddivisa in 3 sotto-unità:

1. Battery Level Input Acquisition Unit: si occupa dell'acquisizione del livello di carica del pacco batterie;
2. Battery Level Processing Unit: si occupa del processamento dei dati in ingresso;
3. Battery Level Output Generation Unit: si occupa della generazione dei segnali di output (indicatore analogico di livello, spia della riserva).

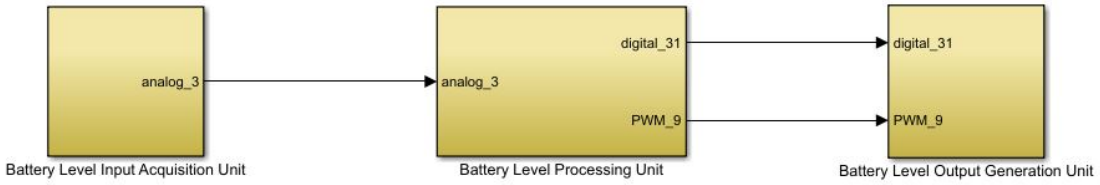


Fig. III.11: Battery Level Subsystem

Nell'immagine seguente è possibile osservare il contenuto del subsystem centrale:

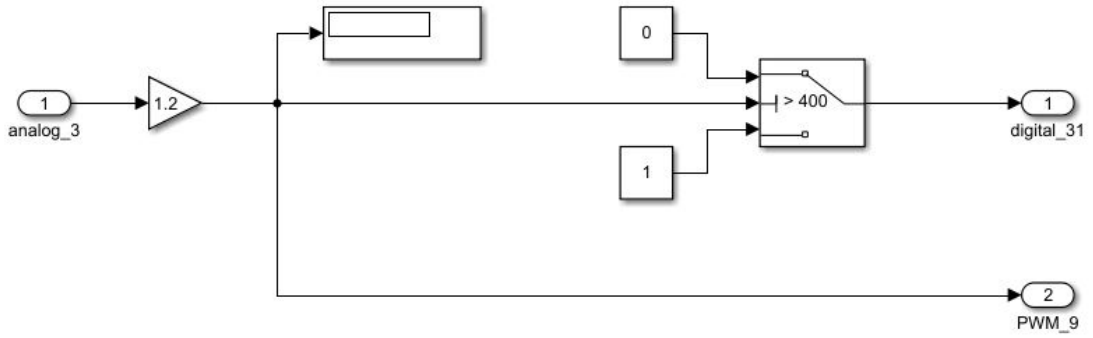


Fig. III.12: Battery Level Processing Unit

Come mostrato in figura III.12, la tensione di input è stata adeguata alla scala dell'indicatore analogico grazie ad un amplificatore.

La tensione processata dovrebbe essere inviata all'indicatore dello stato di carica della batteria principale tramite un'output analogico. *Arduino Mega 2560 Rev3* non possiede uscite di tipo analogico, dunque per pilotare l'indicatore analogico è stato necessario sfruttare le potenzialità dell'output **PWM**.

PWM (*Pulse Width Modulation*) non è altro che una tecnica utilizzata al fine di ottenere output analogici con mezzi digitali. Il controllo digitale è utile per dare origine ad un'onda quadra, che rappresenta un segnale commutato tra on e off. Questo modello può simulare tensioni tra la tensione massima e 0 Volt, in base alla porzione di tempo che il segnale trascorre nello stato on rispetto al tempo trascorso nello stato off [18]. La durata del "tempo di accensione" è denominata larghezza dell'impulso, che può essere modulata per ottenere valori analogici variabili. Se lo schema on-off viene ripetuto abbastanza velocemente, il risultato sarà un segnale di output corrispondente ad una tensione costante tra 0 e Vmax.

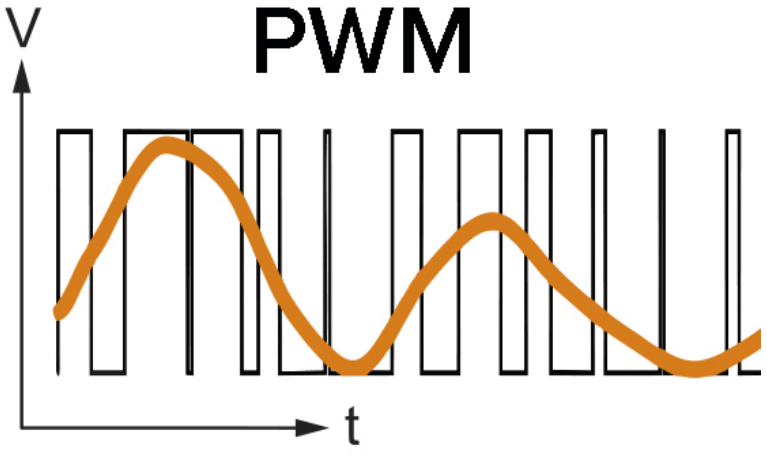


Fig. III.13: Pulse Width Modulation (PWM)

Infine, il segnale processato è stato inviato ad uno switch che, al di sotto di una soglia preimpostata, dà alla *Battery Level Output Generation Unit* il consenso per attivare la spia della riserva.

III.4 Livello batteria servizi

E' stata prevista una batteria dei servizi per garantire il funzionamento di tutti i sistemi di controllo, anche in caso di malfunzionamento della batteria principale.

L'unità Simulink per la gestione del livello di carica della batteria dei servizi è suddivisa in 3 sotto-unità:

1. Service Battery Input Acquisition Unit: si occupa dell'acquisizione del livello di carica della batteria dei servizi;

2. Service Battery Processing Unit: si occupa del processamento dei dati in ingresso;

3. Service Battery Output Generation Unit: si occupa della generazione del segnale di output (spia di indicazione).

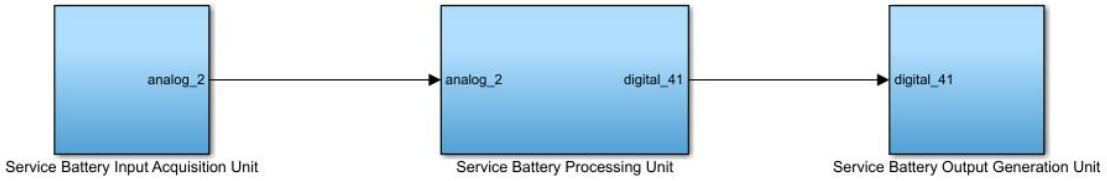


Fig. III.14: Service Battery Subsystem

Come mostrato in figura III.15, la tensione di input è stata adeguata grazie ad un amplificatore ed è stata inviata ad uno switch che, al di sotto di una soglia pre-impostata, dà alla *Service Battery Output Generation Unit* il consenso per attivare la spia di indicazione.

Fig. III.15: Service Battery Processing Unit

III.5 Livello temperatura e umidità

Nei veicoli elettrici è importante misurare la temperatura della batteria principale, in quanto un livello di temperatura che supera una certa soglia potrebbe determinare il fenomeno del *thermal runaway* [7]. Inoltre, si è scelto di rilevare anche il valore relativo all'umidità.

L'unità Simulink per la gestione dei livelli di temperatura e umidità è suddivisa in 3 sotto-unità:

1. Temp-Hum Input Acquisition Unit: si occupa dell'acquisizione dei dati relativi a temperatura e umidità;
2. Temp-Hum Processing Unit: si occupa del processamento dei dati in ingresso;
3. Temp-Hum Output Generation Unit: si occupa della generazione dei segnali di output (messaggi di testo sul display, spia di indicazione).

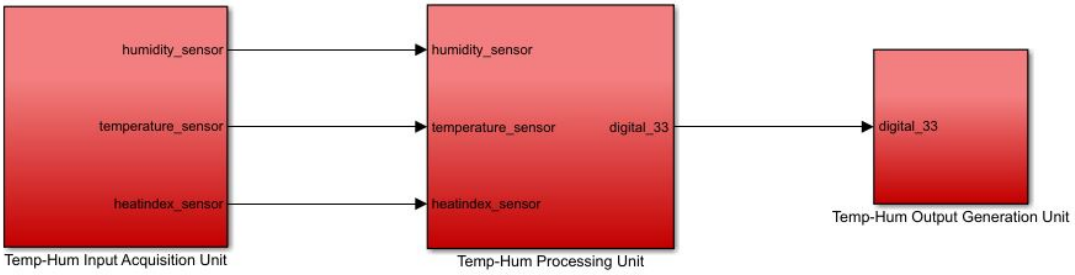


Fig. III.16: Temp-Hum Subsystem

Nell'immagine seguente è possibile osservare il contenuto del subsystem centrale:

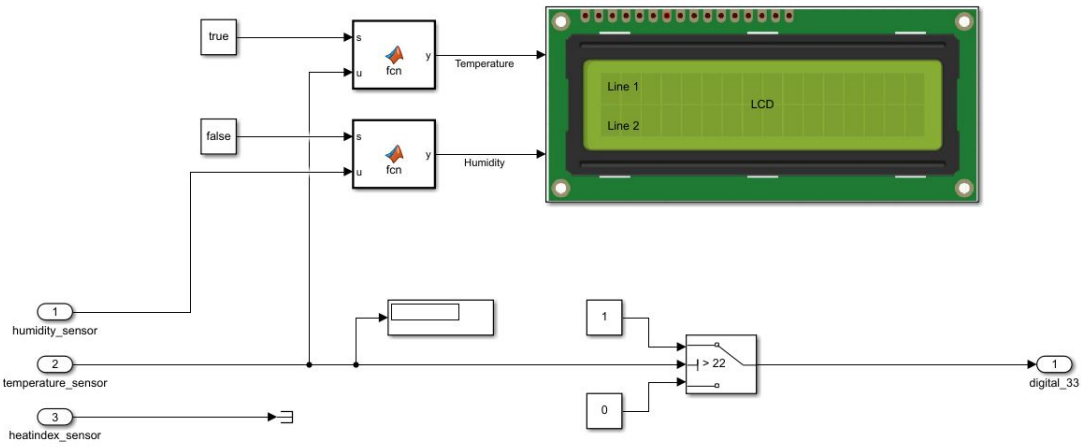


Fig. III.17: Temp-Hum Processing Unit

Come mostrato in figura III.17, il valore relativo all'umidità viene inviato ad una *MATLAB function* presente nella *Simulink library for Arduino Liquid Crystal Display*, che si occupa della conversione del valore acquisito in messaggio di testo, che sarà poi visualizzato a scopo informativo su un display LCD.

In modo analogo, il valore relativo alla temperatura del pacco batterie viene inviato ad una *MATLAB function* presente nella *Simulink library for Arduino Liquid Crystal Display*, che si occupa della conversione del valore ottenuto in messaggio di testo, che sarà visualizzato a scopo informativo su un display LCD.
Inoltre, tale valore è stato inviato ad uno switch che, al di sopra di una soglia preimpostata, dà alla *Temp-Hum Output Generation Unit* il consenso per attivare la spia di pericolo: "TEMPERATURA ECCESSIVA".

III.6 Indicatori di direzione

E' stata progettata la gestione relativa agli indicatori di direzione.

L'unità Simulink per la gestione degli indicatori di direzione è suddivisa in 3 sotto-unità:

1. Turning Signal Input Acquisition Unit: si occupa dell'acquisizione del segnale proveniente dal selettore degli indicatori di direzione;

2. Turning Signal Processing Unit: si occupa del processamento dei dati in ingresso;

3. Turning Signal Output Generation Unit: si occupa della generazione del segnale di output (spia lampeggiante di indicazione).

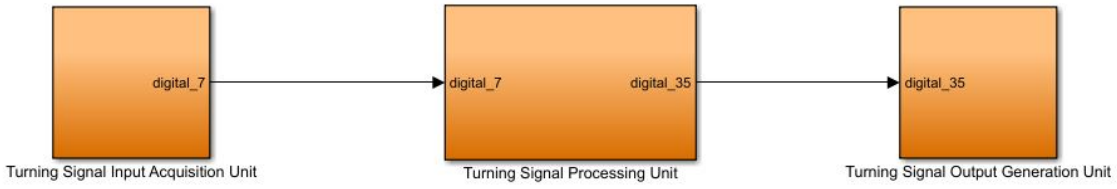


Fig. III.18: Turning Signal Subsystem

Come mostrato in figura III.19, il segnale di input viene inviato ad uno switch che, attivando un'onda quadra con opportuna frequenza, dà alla *Turning Signal Output Generation Unit* il consenso per attivare la spia lampeggiante di indicazione.

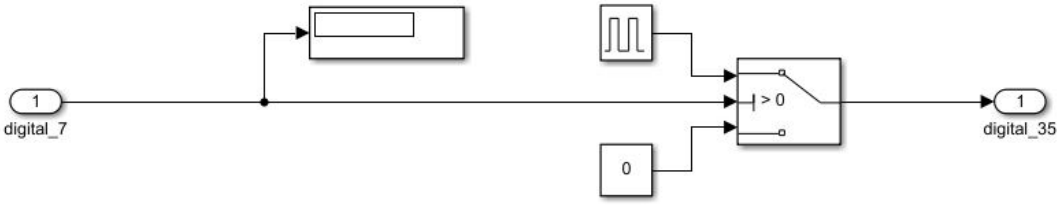


Fig. III.19: Turning Signal Processing Unit

In modo analogo è stata progettata la gestione relativa alle **luci di emergenza**.

III.7 Luci di posizione

E' stata progettata la gestione relativa alle luci di posizione.

L'unità Simulink per la gestione delle luci di posizione è suddivisa in 2 sotto-unità:

1. Position Lights Input Acquisition Unit: si occupa dell'acquisizione del segnale proveniente dal selettore delle luci di posizione;

2. Position Lights Output Generation Unit: si occupa della generazione del segnale di output (spia di indicazione).

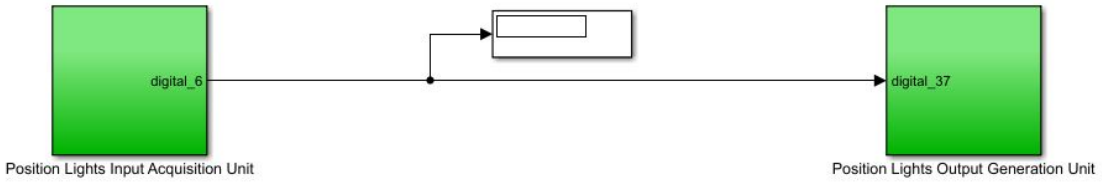


Fig. III.20: Position Lights Subsystem

Come mostrato in figura III.20, il segnale di input invia alla *Position Lights Output Generation Unit* il consenso per attivare la spia di indicazione.

In modo analogo è stata progettata la gestione relativa a:

- Luci abbaglianti;
- Lunotto termico;
- Luci retronebbia;
- Indicazione freno di stazionamento.

III.8 Testing del modello

Per verificare il corretto funzionamento del modello progettato è stato fondamentale eseguire vari test sul sistema. E' stato importante procedere con il testing in fase di progettazione, in modo da individuare eventuali *bug* prima di aver installato il sistema sul veicolo, in questo modo è stato possibile procedere alla correzione degli stessi tramite software, limitando di gran lunga tempi e costi di correzione.

Sono state effettuate tre tipologie di test:

- Test funzionali: controllano che tutte le specifiche funzionali siano state implementate e funzionino correttamente;
  - Unit testing: viene controllata ogni unità singolarmente;
  - Integration testing: vengono verificate più unità in modo combinato;
  - System testing: viene testato l'intero sistema.
- Test di performance/robustezza: misurano il grado di affidabilità e di efficienza del sistema.

Sono stati generati vari *test harness* in modo da poter verificare tutte le possibili casistiche del sistema. Per ciascuno di essi è stato scelto uno specifico stimolo di input ed ipotizzato un *expected output*. E' stata effettuata la simulazione di tutti i *test harness* e l'output della simulazione è stato confrontato con l'*expected output*.

I vari test hanno prodotto output ragionevolmente vicini agli *expected output* per tutti gli stimoli di input. Il numero di test effettuati ha raggiunto un alto grado di coverage.

III.9 Generazione del codice sorgente

Dopo aver implementato e testato il modello, è stato necessario generare il codice sorgente, in modo da poterlo successivamente caricare sull'hardware di destinazione scelto (*Arduino Mega 2560 Rev3* [1] e *NXP Freedom-K64F* [2]).

Per far ciò, dopo aver configurato correttamente il *solver* di MATLAB, sono state sfruttate le potenzialità dell'**Embedded Coder**.

Durante la *build* del modello è stato generato automaticamente un *Code Generation Report*, come mostrato in Fig. III.21:

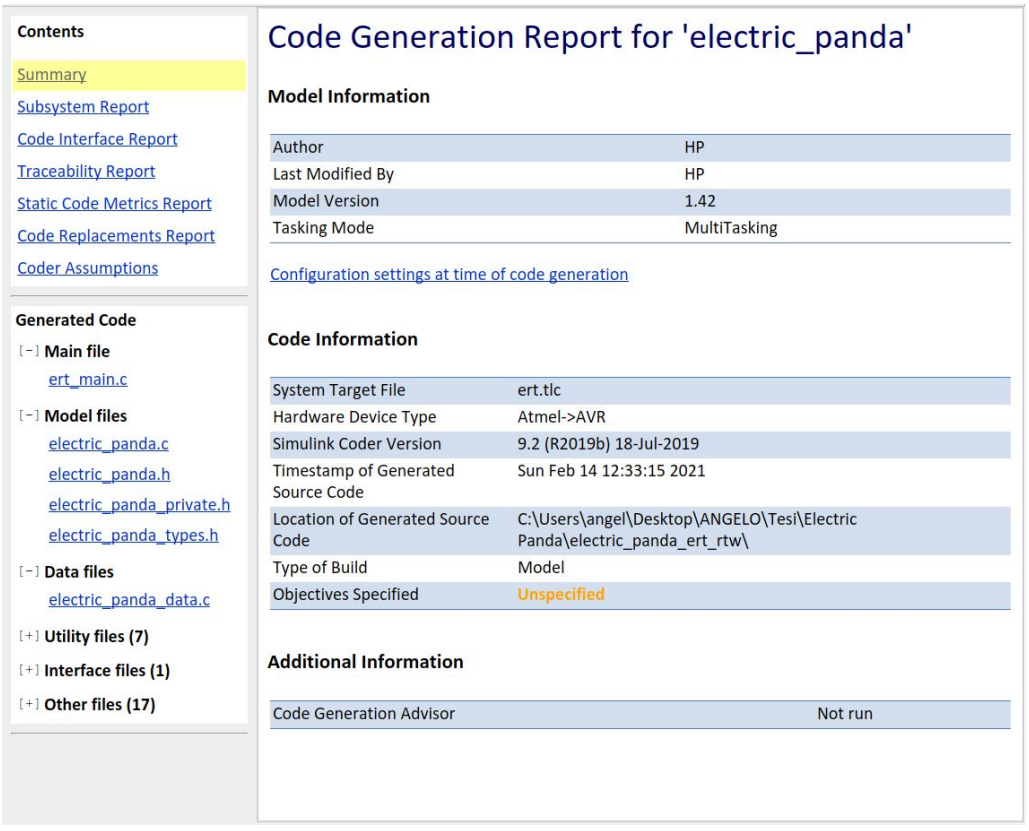


Fig. III.21: Code Generation Report

Il processo di *code generation* produce vari file di output, fra i quali quelli da attenzionare sono:

- ert_main.c (Testbench per il modello);
- nome_modello.c (Codice sorgente per il modello);
- nome_modello.h (Tipi di dati specifici del modello);
- rtwtypes.h (Tipi di dati specifici per hardware).

Il file nome_modello.c il esegue la traduzione del modello in tre funzioni:

1. initialize(M, U, Y): resetta lo stato M del modello;
2. step(M, U, Y): esegue un *integration step* per il modello *fixed-step/discrete time*;
3. terminate(M, U, Y): svuota la memoria dopo l'ultima esecuzione del modello.

Il file rtwtypes.h, invece, traduce i tipi di dati nativi di *Simulink* (ad esempio, *char*) nei tipi di dati specifici dell'hardware (ad esempio, *int8_T*).

III.9.1 Integrazione del software sull'hardware

Per quanto riguarda l'integrazione del software generato sull'hardware, si è scelto di utilizzare la tecnica *full executable* per le seguenti ragioni:

- L'applicazione e la piattaforma software sono generate automaticamente a partire dal modello;
- Non è necessario l'intervento manuale;
- Consente di eseguire simulazioni del tipo *hardware-in-the-loop*.

Capitolo IV

Sperimentazione effettuata sul veicolo

Una volta terminata la fase di progettazione, sviluppo e simulazione del *sistema embedded*, è stato opportuno installare e testare il tutto su un **veicolo target** al fine di verificare che il modello progettato e testato dia gli stessi risultati sul prodotto finale.

Per motivi economici e per la facile reperibilità delle parti di ricambio, come veicolo target, è stata scelta la *FIAT Panda 900* del 2000.

Per procedere all'installazione è stato necessario acquisire i dati relativi all'impianto elettrico dell'autovettura scelta.

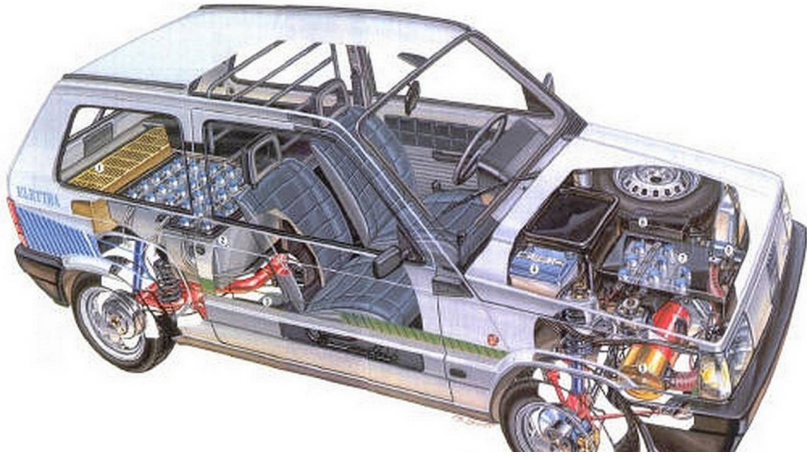


Fig. IV.1: FIAT Panda Elettra [19]

In figura IV.2 sono illustrati i componenti del quadro di controllo, nello specifico:

1. Tachimetro/odometro;
2. Indicatore livello di carica del pacco batterie;
3. Indicatore eccessiva temperatura del pacco batterie;
4. Quadretto comprendente le seguenti spie di indicazione:

    - Luci di posizione;
    - Indicatori di direzione;
    - Luci abbaglianti;
    - Indicazione tensione insufficiente batteria servizi;
    - Luci di emergenza;
    - Indicazione basso livello di carica del pacco batterie;
    - Lunotto termico;
    - Luci retronebbia;
    - Indicazione freno di stazionamento inserito.

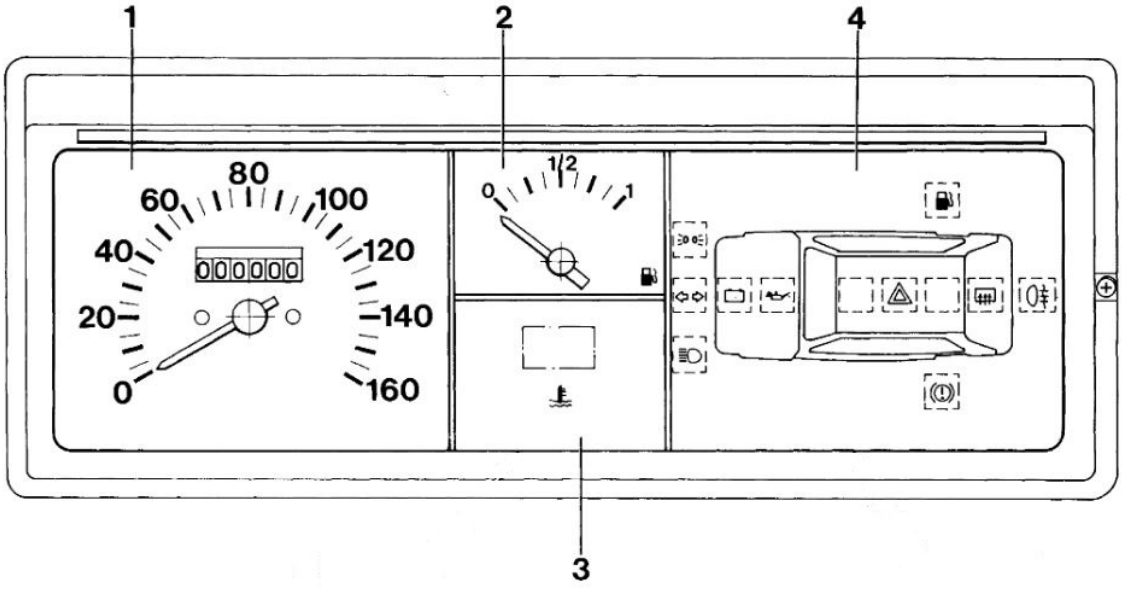


Fig. IV.2: FIAT Panda 900 - Vista frontale quadro di controllo

La seguente figura mostra il retro del quadro di controllo. E' possibile notare che il *tachimetro/odometro* è di tipo meccanico, pertanto in questa sperimentazione è rimasto tale.

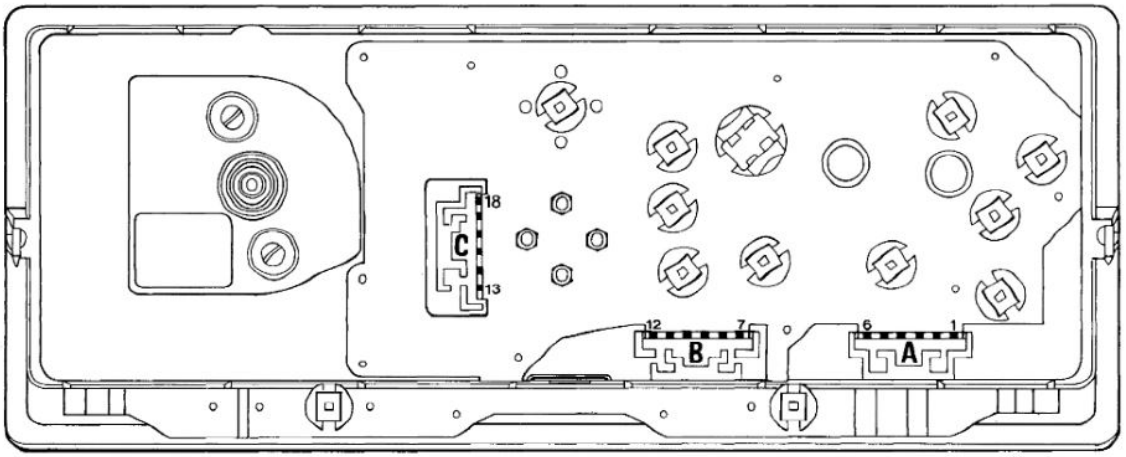


Fig. IV.3: FIAT Panda 900 - Vista posteriore quadro di controllo

In Fig. IV.4 è illustrata la piedinatura dei connettori relativi al quadro strumenti, utile per effettuare le varie connessioni; in Fig. IV.5, invece, si può osservare lo schema elettrico.

CONNETTORE A			CONNETTORE B			CONNETTORE C		
CB	1	+ dalla centralina di derivazione fusibile 4	-	7	Libero	GR	13	Indicatore ottico luci di posizione
VG	2	Indicatore ottico riserva carburante	HG	8	Indicatore ottico insufficiente pressione olio motore	BR	14	Indicatore ottico luci di direzione
HR	3	Indicatore ottico luci retronebbia	NZ	9	Indicatore ottico insufficiente ricarica batteria	VN	15	Indicatore ottico luci abbaglianti
RV	4	Indicatore ottico lunotto termico	N	10	Massa	SN	16	Comando indicatore livello carburante
RN	5	Indicatore ottico luci di emergenza	-	11	Libero	AN	17	Commutatore d'accensione 15/54 (MAR)
-	6	Libero	VB	12	Indicatore ottico eccessiva temp.liquido refrigerante	ZB	18	Ind. ottico livello liquido freni o freno a mano inserito

Fig. IV.4: FIAT Panda 900 - Connessioni quadro di controllo

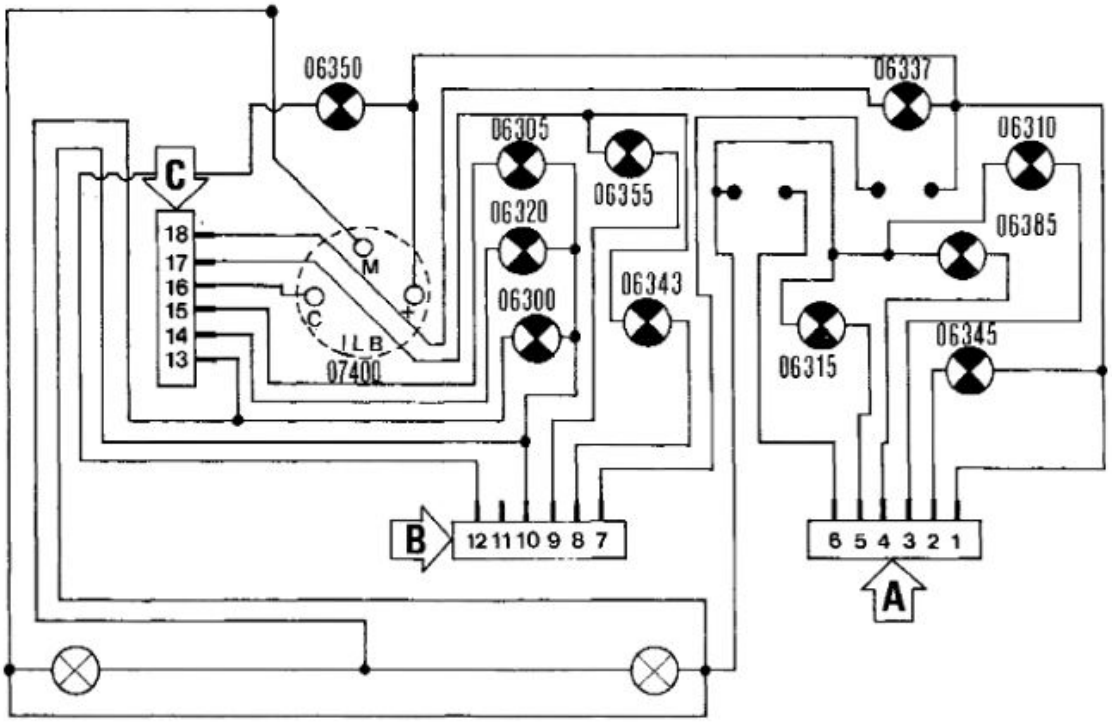


Fig. IV.5: FIAT Panda 900 - Schema elettrico quadro di controllo

IV.1 Componenti utilizzati

Per pilotare il quadro di controllo della FIAT Panda 900, tramite gli output delle *board* con *microcontrollore*, sono stati utilizzati vari componenti elettronici.

IV.1.1 Componenti base

- Led 5mm vari colori;
- Resistori vari;
- Diodi 1N4004;
- Switch e temporary switch.

IV.1.2 Sensore di temperatura e umidità DHT11

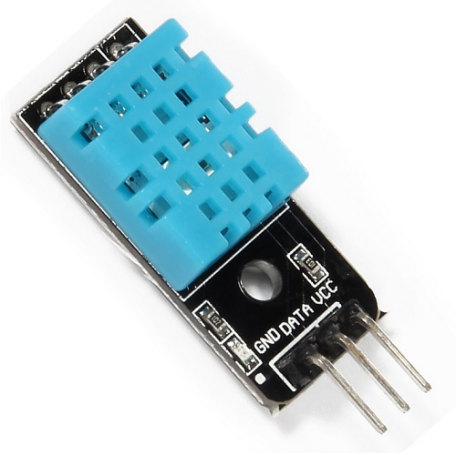


Fig. IV.6: Sensore di temperatura e umidità DHT11

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Fig. IV.7: DHT11 - Specifiche tecniche [20]

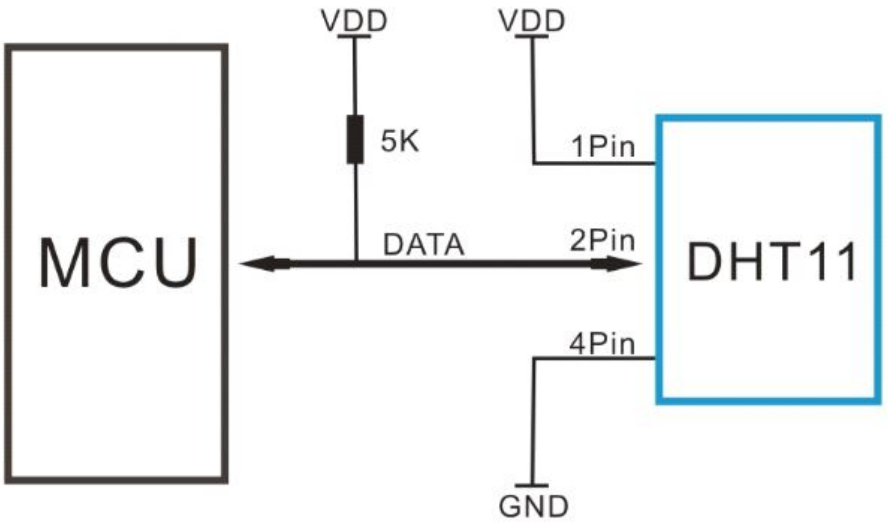


Fig. IV.8: DHT11 - Schema di connessione [20]

	Conditions	Minimum	Typical	Maximum
Power Supply	DC	3V	5V	5.5V
Current Supply	Measuring	0.5mA		2.5mA
	Average	0.2mA		1mA
	Standby	100uA		150uA
Sampling period	Second	1		

Fig. IV.9: DHT11 - Caratteristiche elettriche [20]

IV.1.3 Stabilizzatore di tensione LT1107

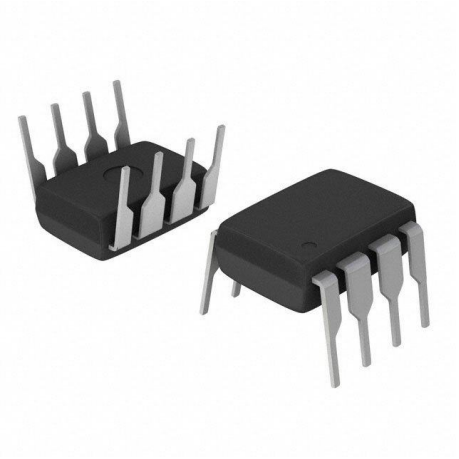


Fig. IV.10: Stabilizzatore di tensione LT1107 [21]

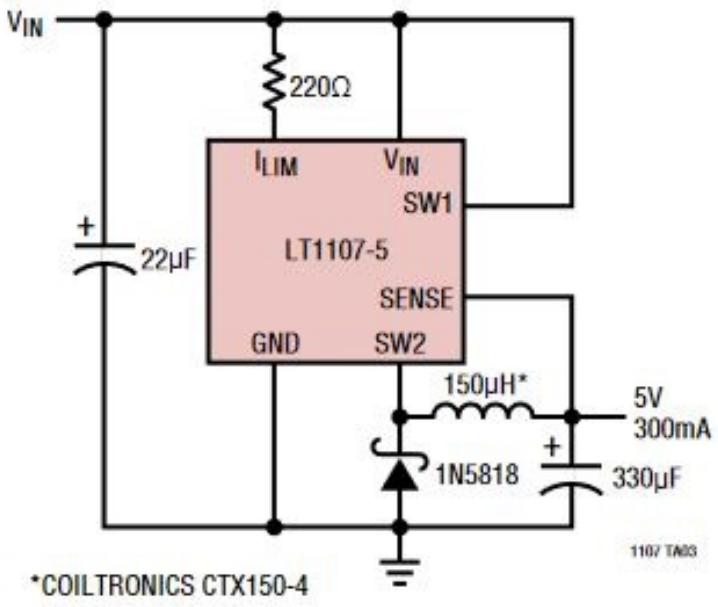


Fig. IV.11: LT1107 - Applicazione tipica [21]

IV.1.4 Relè SRD-05VDC-SL-C

Fig. IV.12: Relè SRD-05VDC-SL-C [22]

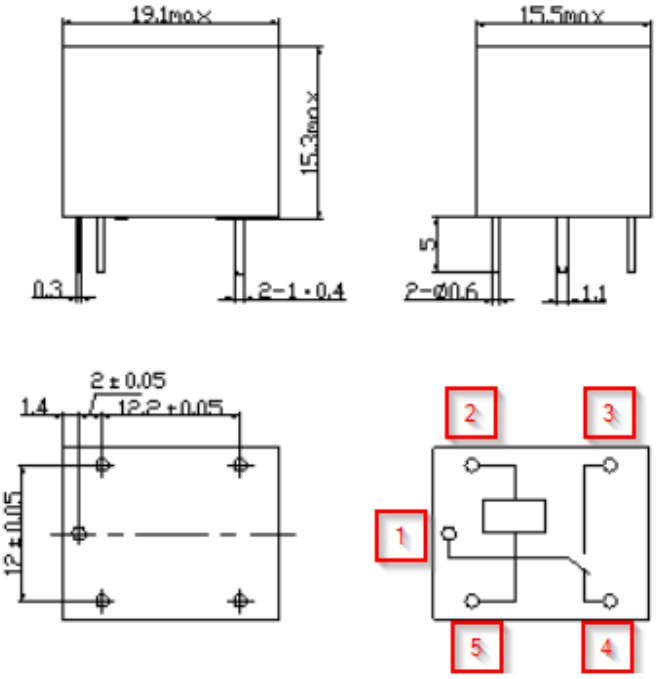


Fig. IV.13: SRD-05VDC-SL-C - Piedinatura [22]

IV.1.5 Display LCD 1602A

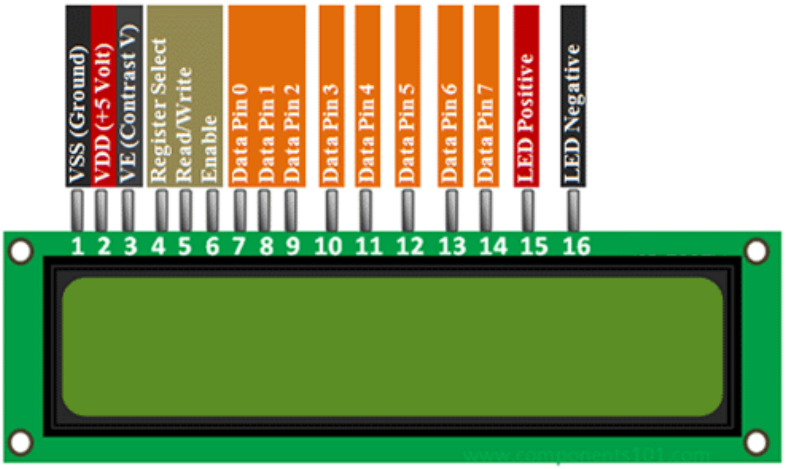


Fig. IV.14: Display LCD 1602A (16x2 caratteri) [23]

No.	Symbol	Level	Function	
1	Vss	--	0V	Power Supply
2	Vdd	--	+5V	
3	V0	--	for LCD	
4	RS	H/L	Register Select: H:Data Input L:Instruction Input	
5	R/W	H/L	H--Read L--Write	
6	E	H,H-L	Enable Signal	
7	DB0	H/L	Data bus used in 8 bit transfer	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L	Data bus for both 4 and 8 bit transfer	
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		
15	BLA	--	BLACKLIGHT +5V	
16	BLK	--	BLACKLIGHT 0V-	

Fig. IV.15: LCD 1602A - Piedinatura [23]

IV.2 Cablaggio dell'impianto elettrico

Prima di procedere alla fase di test per la verifica del corretto funzionamento dell'intero sistema, è stato realizzato il cablaggio dei vari componenti. Per motivi logistici il tutto è stato effettuato su banco di prova.

Varie scelte costruttive sono state dettate dal fatto che l'auto possiede una batteria dei servizi a 12V. Le board utilizzate (*Arduino Mega 2560 Rev3* e *NXP Freedom-K64F*) vanno alimentate a 5V (corrente continua). La tensione è stata quindi regolata sfruttando il **DC-DC Converter 12V/5V**.

Una volta alimentate le board con microcontrollore sono stati cablati i loro input e i loro output (quadro di controllo, display e led di indicazione), come riportato nel capitolo III e nel paragrafo IV.1.

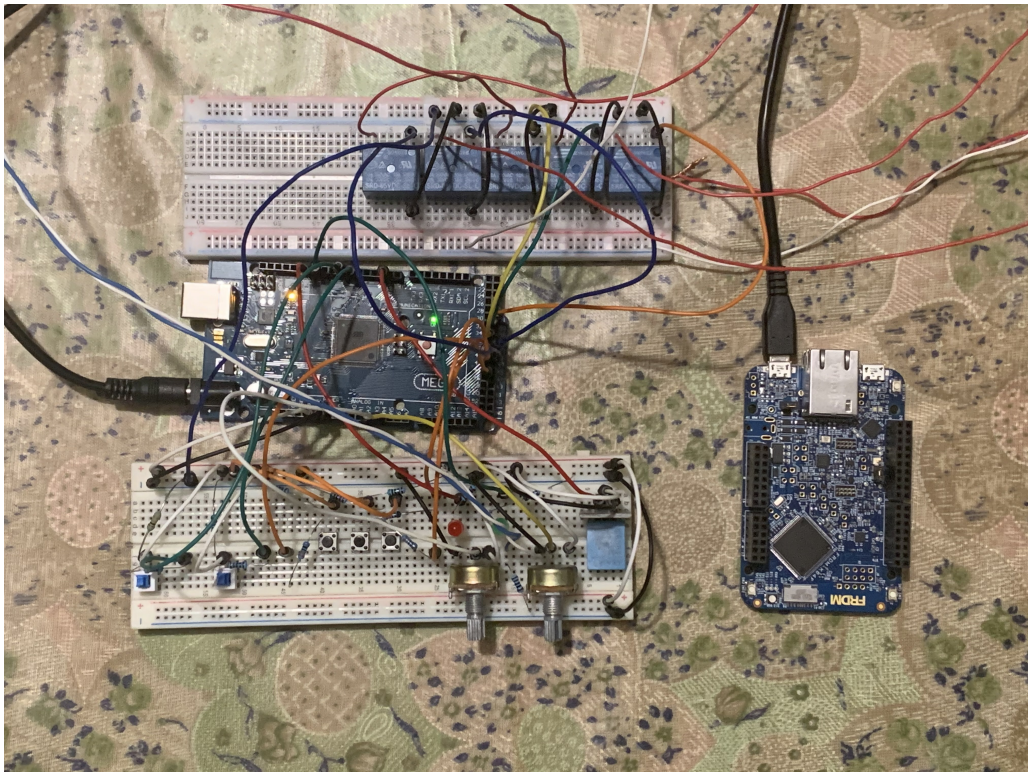


Fig. IV.16: Circuito cablato - focus componenti

Gli *switch* e i *temporary switch* sono stati connessi agli input digitali delle *board* (5V), sfruttando dei resistori di **pull-down** per garantire il corretto funzionamento dei due livelli logici (0/5V).

Per quanto riguarda gli output, tenendo conto che i componenti del quadro strumenti della *FIAT Panda 900* vanno alimentati a 12V, per attivare le varie spie, è stato necessario utilizzare dei Relè pilotati a 5V; inoltre, in questi ultimi sono stati posizionati dei *diodi 1N4004* per proteggere il circuito dalle sovratensioni dovute all'apertura dei contatti dei Relè.
Infine, i led aggiuntivi sono stati protetti da resistori opportunamente dimensionati per limitare le correnti massime ammissibili.

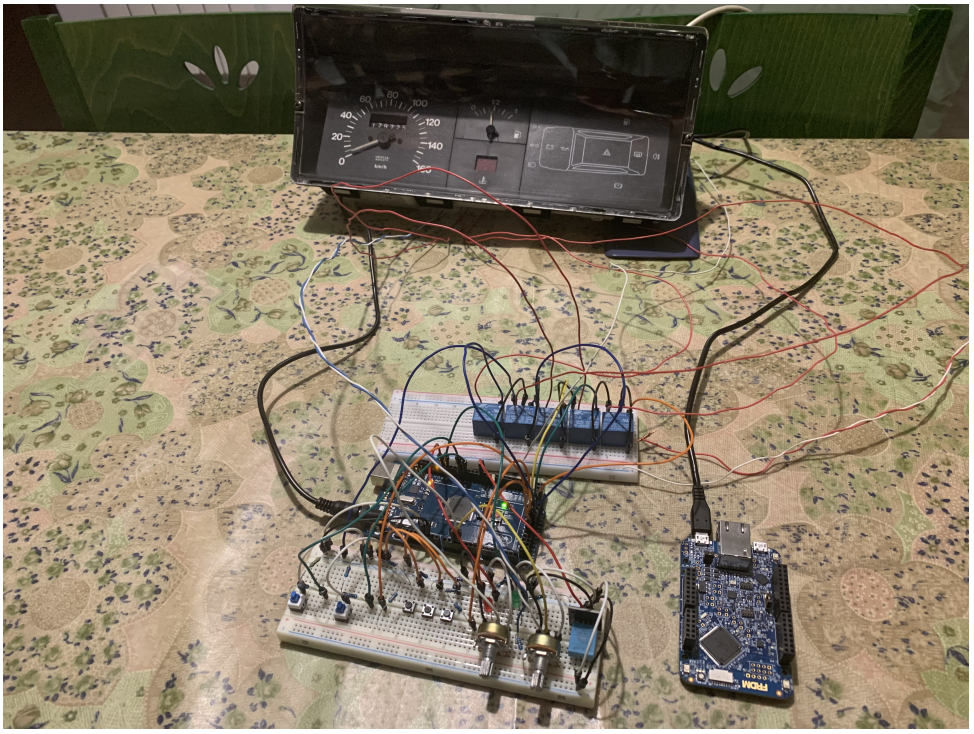


Fig. IV.17: Panoramica circuito cablato

IV.3 Test del sistema e risultati ottenuti

Durante la fase di test è stato verificato il corretto funzionamento dell'intero sistema installato del quale sono state verificate tutte le unità.

Sono stati scelti i vari segnali di input e per ciascuno di essi è stato ipotizzato un *expected output*. Ad ogni pin di ingresso sono stati inviati gli stimoli di input scelti, è stato verificato e registrato il comportamento di ogni singolo output al variare dei vari stimoli di input e infine sono stati confrontati gli *expected output* con i valori ottenuti sperimentalmente.

Per testare il livello di carica della batteria principale e l'accensione della spia relativa al basso livello di carica, non avendo la possibilità di variare, nell'immediato, il livello di carica di una batteria reale, è stata simulata, tramite un potenziometro da 10KΩ, la diminuzione della tensione della batteria, corrispondente al livello di carica.

In modo analogo è stato effettuato il test relativo al livello di carica della batteria dei servizi.

Nelle immagini IV.18, IV.19 e IV.20 è possibile osservare alcune fasi relative al test del sistema.

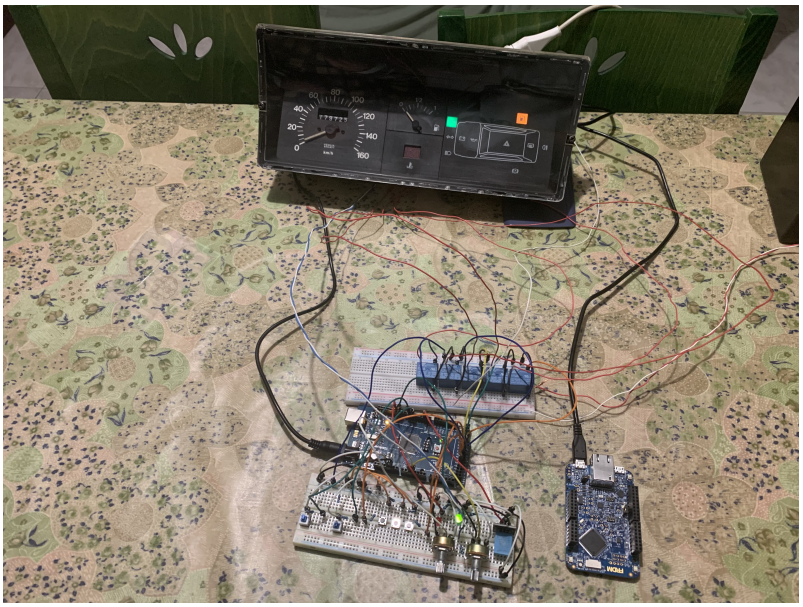


Fig. IV.18: Panoramica sistema testato

Fig. IV.19: Test del sistema - livello batteria e luci di posizione

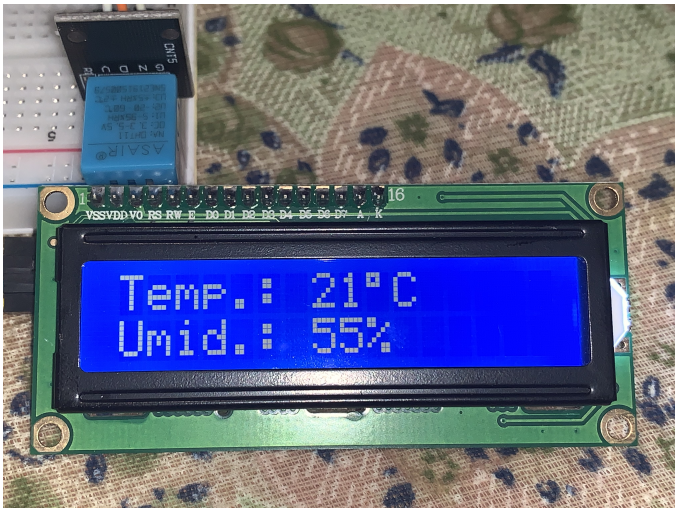


Fig. IV.20: Display LCD

I vari test sono stati effettuati verificando prima ogni singola unità, poi testando più unità contemporaneamente e infine testando l'intero sistema, ottenendo così risultati sovrapponibili agli output previsti.

Capitolo V

Conclusioni

Il software sviluppato e l'hardware utilizzato risultano robusti, ma allo stesso tempo versatili, infatti riescono a garantire: stabilità, sicurezza, facile utilizzo e revisione. E' proprio grazie a tali caratteristiche che l'intero sistema potrà essere installato su una vasta gamma di veicoli commerciali.

In questo lavoro di tesi sono state analizzate, progettate ed implementate varie specifiche: l'inclinometro a tre assi, la gestione del cambio marcia a tre posizioni, la gestione di vari indicatori (livello batteria, temperatura, umidità) e la gestione delle varie luci. Allo stesso modo si potrebbero implementare funzioni di sicurezza similari come: impostare un avviso per la manutenzione programmata dopo un certo numero di chilometri o un certo intervallo di tempo, limitare la velocità del veicolo se la pressione degli pneumatici scende sotto una certa soglia, impedire la messa in moto se il livello di carica della batteria al litio è molto basso.

L'intero lavoro sperimentale presente in questa dissertazione è suscettibile di ulteriori sviluppi e ampliamenti, che consentirebbero l'aggiunta e il controllo di varie specifiche implementative, procedendo secondo l'iter già seguito.

E' altresì opportuno precisare che, nel caso in cui si decida di installare il sistema su un mezzo dotato di centralina elettronica, sarà certamente possibile utilizzare i dati acquisiti ed elaborati per implementare ulteriori funzioni di notifica e/o sicurezza.

A tale scopo è possibile sfruttare le potenzialità delle *board* con microcontrollore già utilizzate, installando su di esse il *CAN-BUS Shield* [24] mostrato in Fig. V.1.

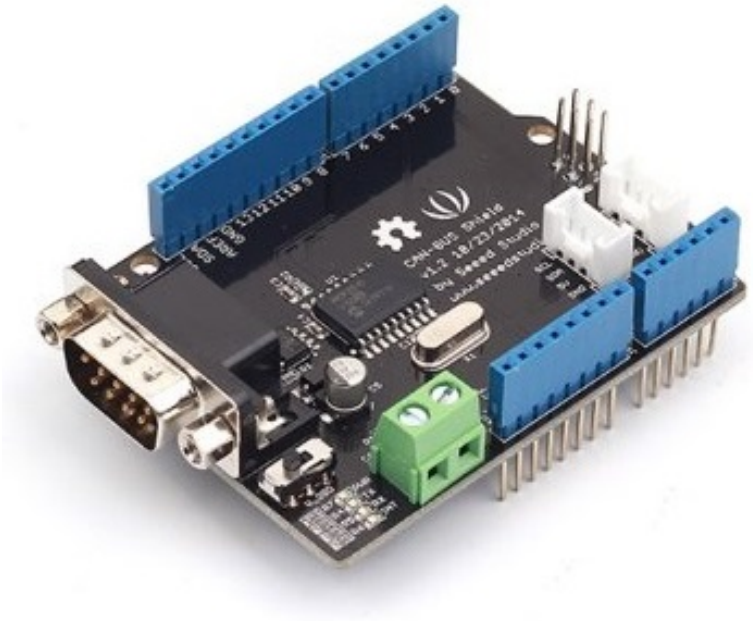


Fig. V.1: CAN-BUS Shield V2.0 [24]

Grazie ad un'espansione di questo tipo, è possibile creare una rete CAN [25] fra microcontrollore e centralina elettronica del veicolo, servendosi di un cavo di collegamento tra l'uscita seriale *DB9 Interface* del *CAN-BUS Shield* e la porta *OBD* del veicolo.

Bisogna precisare, infine, che per quanto riguarda la fase finale di test, il tutto è stato effettuato su banco prova in condizioni statiche.
Per una verifica completa, sarebbe necessario testare l'intero sistema installato sul veicolo in varie situazioni di marcia e in condizioni meccaniche e ambientali variabili, con sbalzi termici ed eventuali vibrazioni e sobbalzi ripetuti.

Bibliografia

[1] A. Nayyar and V. Puri, "A review of arduino board's, lilypad's & arduino shields," in *2016 3rd international conference on computing for sustainable global development (INDIACom)*, pp. 1485–1492, IEEE, 2016.

[2] J. Jabin, A. M. Chowdhury, E. T. Efaz, M. E. Adnan, and M. R. Islam, "An automated agricultural shading for crops with multiple controls," in *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pp. 1–7, IEEE, 2020.

[3] D. De Niz, G. Bhatia, and R. Rajkumar, "Model-based development of embedded systems: The sysweaver approach," in *12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*, pp. 231–242, IEEE, 2006.

[4] A. Kurniawan, *Getting Started with Matlab Simulink and Arduino*. PE Press, 2013.

[5] M. E. C. Team, "Simulink coder support package for nxp frdm-k64f board," 2016. Available on line.

[6] Wikipedia contributors, "Functional safety — Wikipedia, the free encyclopedia," 2021. [Online; accessed 14-February-2021].

[7] Q. Wang, P. Ping, X. Zhao, G. Chu, J. Sun, and C. Chen, "Thermal runaway caused fire and explosion of lithium ion battery," *Journal of power sources*, vol. 208, pp. 210–224, 2012.

[8] S. Yuncheng, "Research on modeling and design of real-time embedded systems," in *2014 7th International Conference on Intelligent Computation Technology and Automation*, pp. 547–550, 2014.

[9] A. K. Rath, D. Roy, D. H. Teja, and G. B. Kumar, "Embedded hardware testing using bootloader," in *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, pp. 1–6, IEEE, 2020.

[10] D. Chikurtev, S. Bogdanov, N. Spasova, and V. Ivaniv, "Prerequisites for a self-sustaining embedded system with artificial intelligence," in *2020 XXIX International Scientific Conference Electronics (ET)*, pp. 1–4, IEEE, 2020.

[11] N. Navet and F. Simonot-Lion, *Automotive embedded systems handbook*. CRC press, 2017.

[12] Wikipedia contributors, "Iso 26262 — Wikipedia, the free encyclopedia," 2021. [Online; accessed 25-February-2021].

[13] Wikipedia contributors, "Autosar — Wikipedia, the free encyclopedia," 2021. [Online; accessed 25-February-2021].

[14] J. Klein, H. Levinson, and J. Marchetti, "Model-driven engineering: Automatic code generation and beyond," 2015.

[15] R. Hỳl and R. Wagnerová, "Fast development of controllers with simulink coder," in *2017 18th International Carpathian Control Conference (ICCC)*, pp. 406–411, IEEE, 2017.

[16] S. M. Prabhu and P. J. Mosterman, "Model-based design of a power window system: Modeling, simulation and validation," in *Proceedings of IMAC-XXII: A Conference on Structural Dynamics, Society for Experimental Mechanics, Inc., Dearborn, MI*, sn, 2004.

[17] Wikipedia contributors, "Software verification and validation — Wikipedia, the free encyclopedia," 2020. [Online; accessed 27-February-2021].

[18] K. Nielsen, "A review and comparison of pulse-width modulation (pwm) methods for analog and digital input switching power amplifiers," in *Audio Engineering Society Convention 102*, Audio Engineering Society, 1997.

[19] G. Brusaglino, "Electric vehicle development in fiat," tech. rep., SAE Technical Paper, 1991.

[20] W. Gay, "Dht11 sensor," in *Advanced Raspberry Pi*, pp. 399–418, Springer, 2018.

[21] J. Williams, "Circuits designed for a cruel and unyielding world," 1994.

[22] H. Yin, "Smart home smoke detection and relay contract based on stm32," in *2016 5th International Conference on Environment, Materials, Chemistry and Power Electronics*, Atlantis Press, 2016.

[23] W. Gay, "I2c lcd displays," in *Custom Raspberry Pi Interfaces*, pp. 35–54, Springer, 2017.

[24] P. Pimple, "Sniffing the automotive can bus for real-time data-logging and real time diagnostics display," in *2018 International Conference on Smart Electric Drives and Power System (ICSEDPS)*, pp. 167–170, IEEE, 2018.

[25] A. García Osés, "Diseño de una red can bus con arduino," 2015.

Elenco delle figure