**Immagine che contiene testo, logo, Elementi grafici, Carattere

Descrizione generata automaticamente**

COMBUSTION & DRIVERLESS

FR-24

**Sviluppo e Analisi Dettagliata del Software per la Main Control Unit nelle Applicazioni di Formula SAE**

**Candidati**

Lorenzo Porcheddu

Emanuele Nencioni

**Docente**

Prof. Laura Carnevali

****

Sommario

[Introduzione 3](#_Toc161335545)

[Presentazione del Progetto e Obiettivi 4](#_Toc161335546)

[Architettura del Veicolo 5](#_Toc161335547)

[Task set e Activity Diagram 7](#_Toc161335548)

[Pedal Task 8](#_Toc161335549)

[Fan Control Task 9](#_Toc161335550)

[Telemetry Task 10](#_Toc161335551)

[Gear Task 11](#_Toc161335552)

[ADC Task 12](#_Toc161335553)

[CAN Handler Task 13](#_Toc161335554)

[AS Accelerator Task 14](#_Toc161335555)

[AS State Handler Task 15](#_Toc161335556)

[AS Error Handler Task 16](#_Toc161335557)

[Check Mode Task 17](#_Toc161335558)

[ASB Check Task 18](#_Toc161335559)

[Timelines e PTPN 20](#_Toc161335560)

# Introduzione

Il Firenze Race Team è la squadra ufficiale di Formula SAE e Formula Student dell’Università degli Studi di Firenze.

È un laboratorio didattico aperto agli studenti di tutte le facoltà dell’ateneo con lo scopo di progettare e realizzare una monoposto da competizione.

Visti i prossimi cambiamenti di regolamento delle competizioni di Formula Student e il futuro del mercato automobilistico, il Team ha iniziato a sviluppare il progetto di **auto a guida autonoma**.

L’obiettivo principale è quello di partecipare nel 2024 alla competizione di FS-East con la vettura a combustione, per poi scendere in pista nella categoria 1D di Formula ATA con la vettura *driverless.*

Canditati

Emanuele Nencioni: Head of Autonomous Department – Vision & System Integration

Lorenzo Porcheddu: Head of Electronics & Controls Department

# Presentazione del Progetto e Obiettivi

La Formula Student è una competizione internazionale in cui squadre di studenti universitari si sfidano nella progettazione e costruzione di automobili da corsa monoposto. In questo progetto di Software Engineering, ci concentreremo sulla programmazione di una scheda elettronica personalizzata per un'auto da competizione, che combina le caratteristiche di un'automobile a combustione con tecnologie driverless.

L'obiettivo principale di questo progetto è sviluppare un sistema di controllo elettronico in grado di gestire il funzionamento del motore, la gestione delle velocità, il controllo del volante, il sistema di frenata, la gestione dell'alimentazione e altre funzionalità essenziali per garantire prestazioni ottimali e sicurezza durante la gara.

La sfida principale consiste nella progettazione di un software che consenta all'auto di operare autonomamente, all'interno dei limiti di velocità e parametri definiti. Ciò richiederà l'integrazione di algoritmi di intelligenza artificiale, sensori avanzati, percezione dell'ambiente e decisioni di guida autonoma.

La scheda elettronica personalizzata dovrà gestire la comunicazione bidirezionale tra l'unità di controllo principale dell'automobile e i vari componenti elettronici, come il sistema di alimentazione, l'impianto di scarico, i sistemi di raffreddamento, i sensori ambientali e molti altri. La programmazione dovrà garantire una corretta interpretazione dei dati forniti dai sensori e una risposta tempestiva per garantire efficienza, sicurezza e prestazioni ottimali.

L'interazione con il pilota sarà un altro aspetto cruciale del progetto. La scheda elettronica dovrà fornire informazioni in tempo reale al pilota, come la velocità dell'auto, lo stato del motore, la temperatura e altre informazioni critiche. Sarà necessario sviluppare un'interfaccia utente intuitiva che permetta al pilota di monitorare e intervenire nel funzionamento dell'auto driverless, garantendo una guida sicura e precisa durante la competizione.

In conclusione, questo progetto di Software Engineering, riguardante la programmazione di una scheda elettronica per la Formula Student, rappresenta una sfida entusiasmante e complessa. Il suo obiettivo principale è sviluppare un sistema sofisticato in grado di controllare in modo autonomo un'automobile a combustione durante la competizione, garantendo sicurezza, efficienza e prestazioni ottimali. L'innovazione tecnologica e l'integrazione di diverse discipline quali l'ingegneria elettronica, l'intelligenza artificiale e la sicurezza informatica saranno fondamentali per il successo di questo progetto.

# Architettura del Veicolo

Come si può vedere dal diagramma di alto livello, la nostra auto integra diverse centraline elettroniche. Sono presenti anche due linee CAN bus per interconnettere il tutto, una dedicata principalmente al motore e l’altra invece al sistema autonomo. Questa suddivisione si rende necessaria data la limitata larghezza di banda disponibile, pari a 1Mbit/s di massimo teorico per ogni linea. In realtà si cerca di non superare il 50% della banda in condizioni normali per evitare la saturazione del bus e il conseguente rallentamento di tutti i messaggi.

Di seguito andiamo ad elencare i componenti del nostro sistema:

* CORE23: si tratta della MCU (Main Control Unit), ovvero della scheda principale di controllo del veicolo. Tutte le altre centraline fanno sempre riferimento a lei per gestire le funzioni interconnesse. Possiede 40 tra ingressi e uscite, analogiche e digitali, che ci permettono di collegare una buona quantità di sensori. Avendo entrambe le linee CAN bus, può essere in grado di leggere una grande quantità di dati, anche dai sensori non collegati direttamente a lei. Per esempio, può leggere la temperatura del motore dalla ECU (Engine Control Unit) e inviarla alla scheda del volante, la PILOT23, per far sì che il pilota possa monitorarla. Viceversa, può leggere il segnale proveniente dal pedale dell’acceleratore e inviarlo alla ECU, che poi lo elaborerà secondo varie strategie per attuare la valvola a farfalla. Su questa scheda si concentrerà lo sviluppo del software, quindi per ora non ci dilunghiamo molto.
* PILOT23: si tratta della HMI del veicolo, infatti è posizionata dentro il volante e si occupa principalmente di gestire l’interazione con il pilota. Sul volante è posto un display, attraverso il quale si possono visualizzare varie informazioni come marcia inserita, velocità, giri motore, temperatura dell’acqua, tempo su giro, ecc. Inoltre, si occupa di leggere i pulsanti ma soprattutto i paddle e la leva della frizione, fondamentali per interagire con il cambio.
* MekTronic MKE7: si tratta della ECU (Engine Control Unit). Come suggerisce il nome, è la centralina che si occupa della gestione del motore a combustione interna; ad essa sono infatti collegati tutti i sensori del motore, come temperatura acqua, pressione turbo, ecc. Possiede una linea CAN per l’interfacciamento con il resto del veicolo, attraverso la quale invia tutta la telemetria del motore in tempo reale.
* Driver Freno: si tratta della scheda atta a gestire il motore del freno, è collegata alla linea CAN per ricevere istruzioni ed inviare dati sul suo funzionamento.
* Driver Sterzo: si tratta della scheda atta a gestire il motore dello sterzo, è collegata alla linea CAN per ricevere istruzioni ed inviare dati sul suo funzionamento.
* AIM EVO 5: si tratta della scheda che ha il compito di registrare e memorizzare tutte le informazioni del veicolo, in tempo reale. Memorizza tutti i dati inviati su entrambe le linee CAN, inoltre ha diversi sensori aggiuntivi collegati, i quali vengono letti e memorizzati. Alla bisogna è possibile inviare i dati di questi sensori sulle linee CAN.
* PC autonomo: si tratta di un PC industriale rugged, in grado di eseguire tantissime operazioni riguardanti il sistema autonomo. Attualmente esegue ROS1 su Ubuntu 20.04, ma si progetta un futuro update a ROS2 per migliorare prestazioni e compatibilità. Monta un Intel Core i7-9700 e una scheda video NVIDIA GTX1650.

# Task set e Activity Diagram

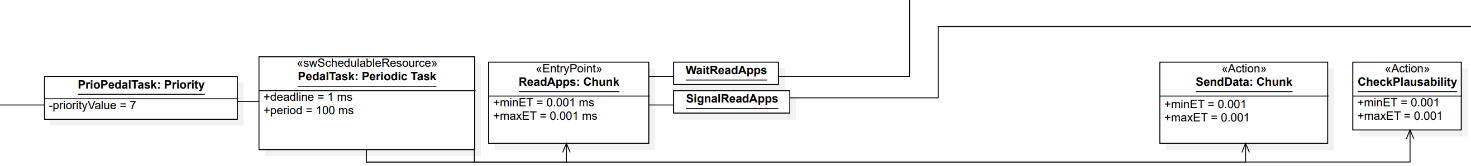
Abbiamo quindi redatto un Activity Diagram, usando il software StarUML.

Data la dimensione del diagramma stesso, lo alleghiamo al presente report.

Di seguito, andiamo a descrivere i singoli task e le relative funzionalità

## Pedal Task

Immagine che contiene testo, diagramma, schizzo, disegno

Descrizione generata automaticamenteSi tratta del task che si occupa della gestione del comando acceleratore, inviato poi al motore a combustione. Funziona solamente durante la modalità manuale, perché durante quella autonoma si utilizza un altro task. Si prende il semaforo sull’ADC per leggere il pedale dell’acceleratore, mandiamo quindi i dati alla centralina motore attraverso il DAC. L’ultima cosa che deve fare è un controllo sulla plausibilità della lettura, per motivi di sicurezza. Leggendo due valori, se vediamo un discostamento superiore al 10% si va in errore.

## Fan Control Task

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamenteImmagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamenteQuesto task si occupa di gestire le ventole del radiatore e dell’intercooler, tramite modulazione PWM. Prendendo il semaforo della CAN motore leggiamo entrambe le temperature; seguendo quindi una strategia, come per esempio una rampa, andiamo a impostare il PWM in uscita. Il tutto dovrà essere gestito nel modo più efficiente possibile, perché le ventole hanno un consumo energetico elevato che potrebbe portare all’esaurimento della batteria, se utilizzate in maniera eccessiva.

## Telemetry Task

Si tratta del task che ci permette di inviare tutti i dati di telemetria, tramite protocollo LoRa. La stazione di terra, quindi, può sapere come si sta comportando la vettura in tempo reale, riconoscendo eventuali problematiche attraverso la sensoristica.

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamenteLa telemetria è. composta principalmente dalle letture dei sensori, quindi degli ADC, ma anche i dati del sistema autonomo, nel caso di modalità driverless, tra i quali lo stato e altri valori di diagnostica. Se dovesse servire si potrebbero inviare anche altri dati riguardanti, per esempio, il motore a combustione oppure le sospensioni che non sono lette dalla CORE direttamente, ma collegate alla AIM Evo 5.

Immagine che contiene testo, linea, Carattere, diagramma

Descrizione generata automaticamente

## Gear Task

Immagine che contiene diagramma, testo, Disegno tecnico, Piano

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamenteSi tratta del task che gestisce il cambio del motore a combustione interna. Per prima cosa andiamo a sempre leggere la marcia corrente, tramite lettura ADC e semaforo, che viene subito mandata al pilota a schermo. Controlliamo quindi se il pilota ha richiesto un cambio marcia, leggiamo il numero di giri e poi andiamo ad azionare gli attuatori per effettuare la cambiata. La lettura del numero di giri si rende necessaria ad ogni cambiata, perché ad un basso numero di giri non possiamo effettuare un’operazione di quick-shift, ovvero la cambiata rapida senza utilizzare la frizione. Sono quindi differenziate due modalità di cambiata, una con e una senza l’utilizzo della frizione.

## ADC Task

Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamenteQuesto task è tanto semplice quanto fondamentale, esso infatti ci permette di leggere gli ADC in ingresso al microcontrollore, andando ad eseguire un polling sui dati letti in continuazione (Continuous Conversion Mode). L’ADC quindi converte sempre i dati in ingresso e li memorizza in un buffer, in modalità di funzionamento circolare. I dati vecchi vengono rimpiazzati continuamente da quelli più nuovi, contemporaneamente noi li leggiamo tramite polling. In questo modo, si va a creare un semplice buffer al quale tutti i task possono accedere tramite semaforo, senza creare problemi di lettura o di accesso ai dati concorrente. L’utilizzo di altre modalità quale il polling diretto, l’interrupt o il DMA con interrupt avrebbe rallentato notevolmente l’esecuzione del codice, o causato cambi di contesto superflui.

## CAN Handler Task

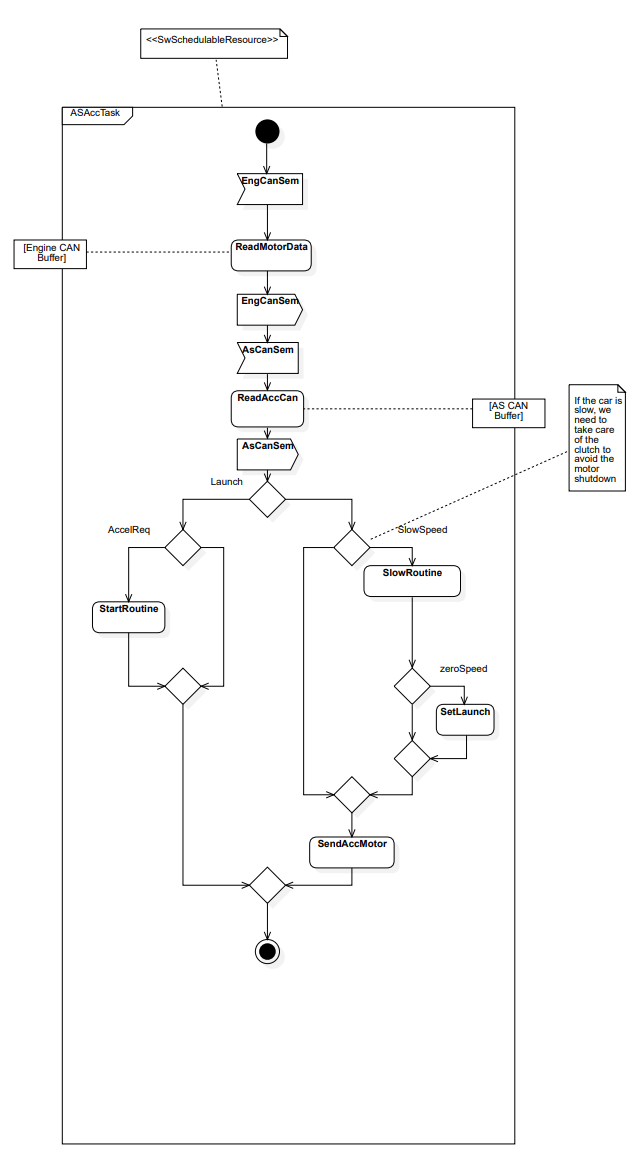
Immagine che contiene testo, schermata, diagramma, Carattere

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamenteSi tratta del task che gestisce la comunicazione di entrambe le linee CAN. Anche in questo caso, si è scelto di non utilizzare l’interrupt per avere un codice più deterministico e per evitare anche un sovraccarico del processore, nel caso ci fossero una grande quantità di messaggi in arrivo. Il task gestisce in autonomia le code in ingresso e in uscita, tramite semafori, in modo tale da evitare situazioni di conflitto.

## AS Accelerator Task

Immagine che contiene testo, schermata, linea, Carattere

Descrizione generata automaticamenteSi tratta del task che gestisce il comando acceleratore nel caso driverless, è infatti il PC autonomo in questo caso a fornirci l’indicazione riguardante la potenza da erogare. Il PC invia il dato via CAN e questo viene inviato alla centralina motore tramite DAC. Non essendoci il pilota umano, il task si occupa anche della gestione della frizione nelle partenze, ma anche quando la velocità e il numero di giri del motore sono bassi, per mantenere il motore acceso.

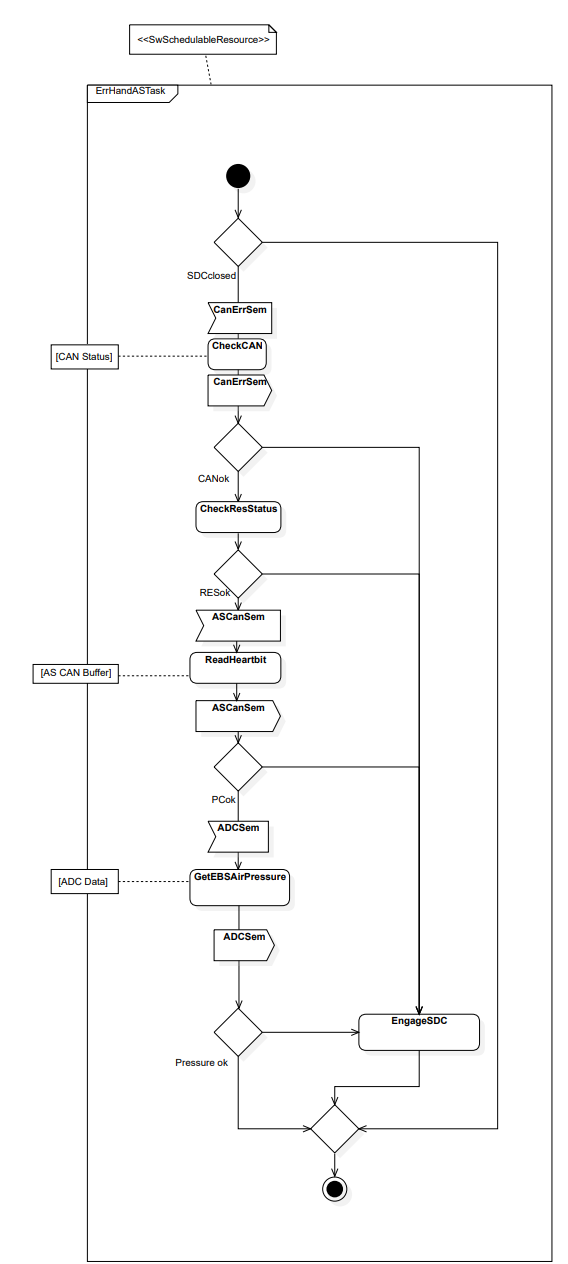
## AS State Handler Task

Immagine che contiene testo, Carattere, linea, numero

Descrizione generata automaticamenteImmagine che contiene testo, diagramma, Piano, Disegno tecnico

Descrizione generata automaticamenteSi tratta del task che gestisce lo stato del sistema autonomo; infatti, durante la competizione driverless, la vettura deve rispettare una serie di stati ben precisi definiti dal regolamento. Abbiamo per esempio lo stato Ready, Driving, Emergency, ecc. Il task va a controllare una serie di parametri, come l’accensione o meno del motore, pressione sistema frenante, ecc. e decide quale stato è il corrente, un po’ come una macchina a stati.

## AS Error Handler Task

Immagine che contiene testo, schermata, Carattere, diagramma

Descrizione generata automaticamenteSi tratta del task di sicurezza della vettura, in grado di ingaggiare l’EBS, ovvero il sistema di frenata di emergenza. Questo task controlla che entrambe le linee CAN siano funzionanti, che il sistema di bloccaggio remoto sia collegato e che ci sia pressione nel sistema EBS. In caso di errore o incongruenze, spegne immediatamente il motore e attiva l’EBS.

## Check Mode Task

Immagine che contiene testo, diagramma, schermata, Parallelo

Descrizione generata automaticamenteSi tratta del task in grado di riconoscere la modalità di funzionamento che vogliamo utilizzare, quindi manuale oppure driverless, e di conseguenza va ad attivare solamente i task necessari. In modalità manuale i task driverless non sono in funzione e viceversa, in modo tale da ottimizzare il funzionamento del programma ed eseguire solo lo stretto necessario. La modalità di funzionamento è commutata tramite un selettore al volante. Per cambiare modalità è necessario eseguire un power-cycle.Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

## ASB Check Task

Immagine che contiene testo, schermata, linea, Carattere

Descrizione generata automaticamenteSi tratta del task che gestisce l’ASB, o Autonomous System Brake. Essendo l’impianto frenante un dispositivo di sicurezza, il regolamento ci impone un controllo sequenziale di tutti i sensori di pressione riguardanti il suddetto impianto. In caso non sia tutto ok, il task va in errore e spegne il motore, non permettendo di fatti che la vettura possa procedere con le operazioni. Inoltre, manda anche un messaggio di errore a display. Nel caso vada tutto bene invece, da il via libera al PC per la modalità di guida.

Immagine che contiene testo, diagramma, linea, Parallelo

Descrizione generata automaticamente

# Timelines e PTPN

Dopo aver completato Activity Diagram e Object Diagram, abbiamo scritto il codice in C e abbiamo misurato i tempi di esecuzione dei vari task. Sorprendentemente, alcuni task hanno tempi di esecuzione di qualche microsecondo, data la frequenza di clock di 216MHz del nostro microcontrollore.