Università degli Studi di Cagliari

Facoltà di Ingegneria

Corso di Laurea in Ingegneria delle Telecomunicazioni

Modellazione, Simulazione e Implementazione di un Sistema Brake by Wire su FlexRay

Relatori:
Ing. Alessandro GIUA

Tesi di Laurea di:
Sergio LORRAI

Anno Accademico 2008/2009

Ringraziamenti

Un particolare ringraziamento va al mio Relatore Professore Alessandro Giua per la disponibilità mostrata durante tutto il periodo di realizzazione della tesi.

Ringrazio l'azienda Akhela che mi ha dato la possibilità di fare quest'esperienza studio e lavoro che ha portato a questo risultato. Ringrazio in particolar modo chi mi è stato sempre vicino, mi ha assistito e inserito in un bellissimo team. Non posso non citare e ringraziare Antonio Solinas, Gianluca Pirroni, Graziano Meola e Fausto Sessego.

Sommario

Il presente lavoro è nato grazie all'opportunità che mi è stata offerta nel luglio scorso da Akhela s.r.l di svolgere un tirocinio formativo presso la loro sede sita a *Macchiareddu*. Il contratto formativo è durato fino a gennaio del corrente anno, quindi sei mesi di attività presso l'azienda. Tuttavia per completare il lavoro il tempo non è stato sufficiente, ma devo evidenziare il fatto che il rapporto di lavoro, e quindi di tesi, non si è concluso con il termine del contratto è andato oltre. Coloro che mi hanno seguito e aiutato nei mesi del tirocinio sono stati sempre a disposizione per tutto ciò che riguarda il lavoro che è stato preparato. Quindi posso affermare che è stata un'ottima esperienza di studio nonché lavorativa che mi ha permesso di maturare competenze e esperienza. L'obiettivo del tirocinio era quello di studiare la comunicazione su bus FlexRay di sistemi distribuiti automotive, in particolare Brake-by-Wire, e di creare un modello il più possibile attendibile che potesse essere usato per l'implementazione su board, testato e quindi essere usato sui veicoli, commerciali e non. Sicuramente il compito non è stato facile, però grazie anche al supporto dei colleghi si è riusciti nell'intento. Il progetto si articola di 6 capitoli il primo dei quali introduce l'argomento principale della tesi, i motivi che hanno, e continuano, a spingere i ricercatori, le imprese e il mercato a richiedere sistemi automotive all'avanguardia con sistemi di sicurezza con vincoli sempre più stringenti. Nel secondo capitolo viene fatta una sufficiente descrizione teorica sulla dinamica dei veicoli per avere la consapevolezza del fatto che il modello da realizzare può avere caratteristiche differenti in base alle variabili ambientali che si considerano. Il modello risulterà tanto più fedele alla realtà tanto più gradi di libertà vengono considerati. Naturalmente è impossibile pensare di poter ricalcare fedelmente quanto succede nella realtà, poiché è risaputo che il comportamento delle variabili non è deterministico, e inoltre sarebbe impossibile realizzare un sistema del genere a causa della sua complessità. Il terzo capitolo inizia focalizzare con precisione gli argomenti centrali della tesi. Viene fatta una piccola panoramica sui *sistemi X-by-Wire*, *Brake-by-Wire* e sui *sistemi realtime*. Il quarto capitolo descrive in modo dettagliato il protocollo di comunicazione FlexRay, e il suo funzionamento. Si è cercato di evidenziare le

diversità rispetto ai precedenti, e concorrenti, protocolli di comunicazione usati nel settore *automotive*. Nel quinto capitolo è presentato il fulcro della tesi: il modello distribuito realizzato. Questo rappresentata un veicolo, e in particolare la comunicazione attraverso FlexRay del sistema frenante elettromeccanico. Il modello è formato da un nodo principale più altri quattro nodi (che rappresentano le ruote) che "parlano" in *realtime* attraverso il bus FlexRay distribuito. Il sesto capitolo chiude il lavoro e vengono indicati, oltre il resoconto dei risultati ottenuti, anche gli scenari futuri per questo progetto.

Elenco delle figure

Indice

Capitolo 1

Introduzione

In questo capitolo d'introduzione si è cercato di dare un'idea del perché nei decenni è cresciuta la necessità di sviluppare autoveicoli sempre più "sicuri". Verrà introdotto l'argomento centrale della tesi e le sue problematiche. Per maggiori approfondimenti sugli argomenti trattati in questo capitolo si può far riferimento ai documenti della bibliografia [16], [21], [9].

1.1 Contesto

Dalla sua nascita l' evoluzione dell'automobile ha interessato soprattutto il sistema di propulsione, grazie al quale oggi esistono motori a elevato rapporto di compressione. Oggi riveste grande importanza l'introduzione dei microprocessori. Gli attuali autoveicoli dispongono ormai di decine e decine di centraline elettroniche di controllo (ECU, *Electronic Control Unit*) che controllano in pratica tutti i dispositivi come motore, freni, climatizzazione, serrature che sono gestibili attraverso un telecomando, il tergicristallo che adatta la sua velocità all'intensità della pioggia, le luci che si accendono automaticamente entrando in un tunnel. Ogni dispositivo assume quindi una propria intelligenza, rendendo automatiche molte operazioni.

Ci sono altre centraline che si occupano della sicurezza permettendo di avere un migliore controllo del veicolo in condizioni di bassa aderenza o in caso di incidente (cercando di ridurne gli effetti) ed occupandosi del comfort all'interno dell'abitacolo.

La presenza di "intelligenza" su tutti i dispositivi richiede l'implementazione di funzioni complesse, realizzabili attraverso l'interazione tra i vari dispositivi. Per ottenere funzioni di questo tipo è necessario quindi che tutti i dispositivi siano tra loro interconnessi e possano comunicare. Se su un veicolo moderno fosse realizzata

ogni connessione necessaria, il numero di connessioni, e quindi di fili, sul veicolo sarebbe enorme.

1.1.1 Trend tecnologico

Nel primo dopoguerra le vetture di gamma elevata potevano avere circa 70 collegamenti elettrici; oggi si arriva a 2000 e questo significa diversi km di cavi. E' possibile apprezzare dal grafico della Figura 1.1 come i sistemi di controllo della guida siano stati adottati, e man mano si siano evoluti, fin dai primi anni del 1900.

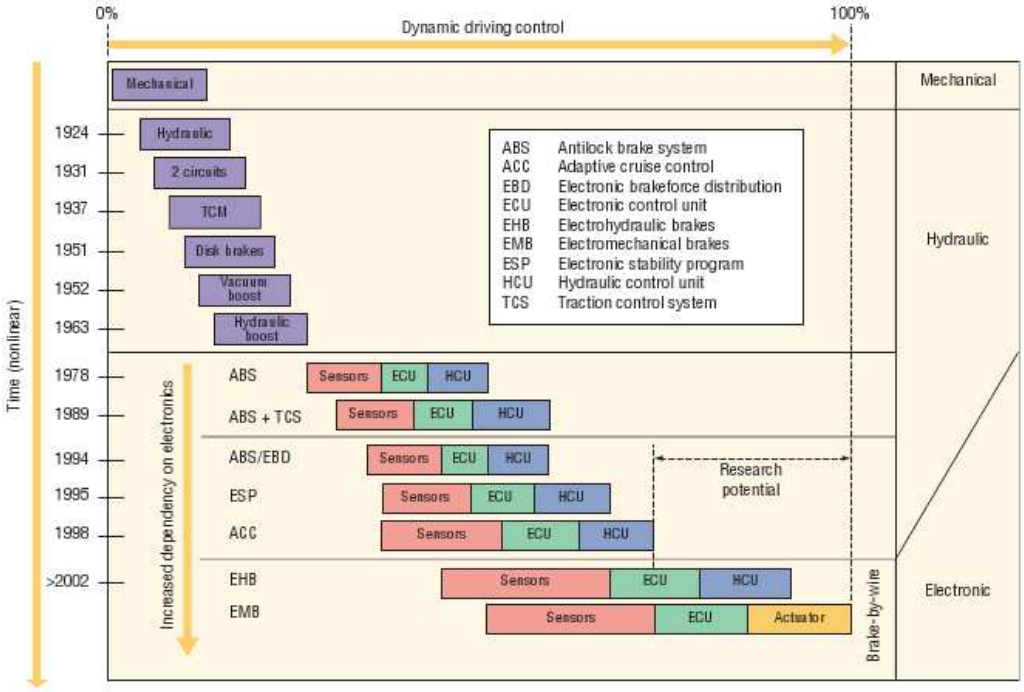

Figura 1.1: Sviluppo dei sistemi di controllo della guida

Questo aumento dei componenti elettronici ha naturalmente modificato la richiesta di potenza da parte del veicolo, in quanto ogni singolo elemento necessita di una certa alimentazione. Per sistemi critici deve essere garantita sempre la potenza richiesta, e inoltre ci sono sempre più componenti elettronici che richiedono una certa quantità di potenza anche in situazione di motore spento. Si può stimare che l'aumento annuo della richiesta di potenza da parte del veicolo, dovuto ai suoi componenti elettronici, sia di circa del 4%.

A questo continuo aumento di potenza c'è da dire che corrispondono dei benefici in termini di comfort, sicurezza, rispetto dell'ambiente (basti pensare ai sistemi che permettono di ridurre il consumo del carburante) e così via.

Una delle strategie ideate per fornire una quantità di potenza sufficiente alle richieste è quella di utilizzare un alternatore integrato su un volano che, operando a 42 V, offre una soluzione accettabile al problema.

La 1.2 mostra come, nel tempo, il numero dei componenti elettronici destinati alla sicurezza passiva e attiva, siano aumentati. Grazie all'elettronica quindi si è riu-

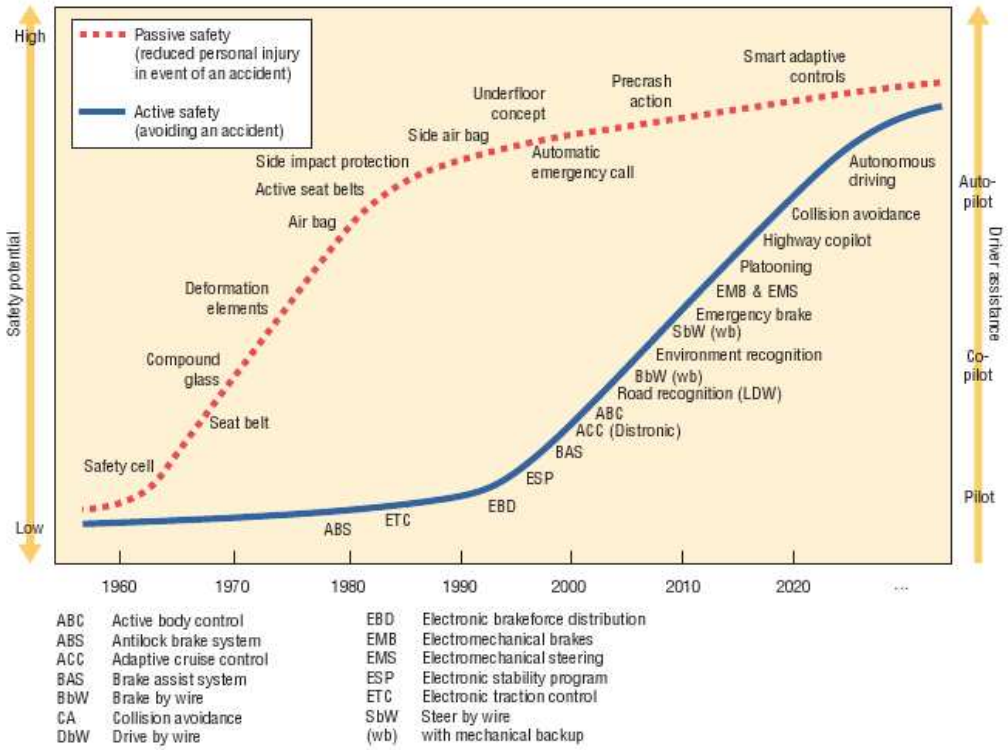

Figura 1.2: Incremento dei componenti elettronici nel veicolo

sciti a ridurre il numero e la lunghezza dei collegamenti attraverso i fili. Oggi sono di gran lunga più utilizzate le connessioni monofilari che sostituiscono con un'unica coppia di fili moltissimi collegamenti. La Figura 1.3 mostra come i componenti del veicolo erano tra di loro collegati, un numero non indifferente di cavi e componenti meccanici:

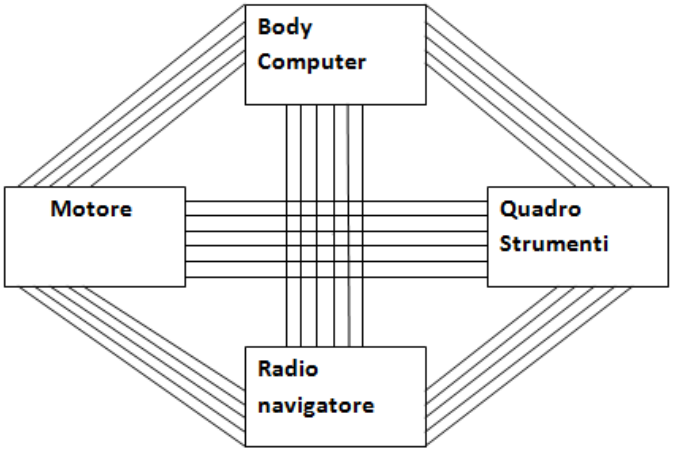

Figura 1.3: Schema multifilare

Con il tempo invece la situazione si è evoluta portando a delle soluzioni più efficienti in termini di affidabilità, peso, ingombro e comfort.

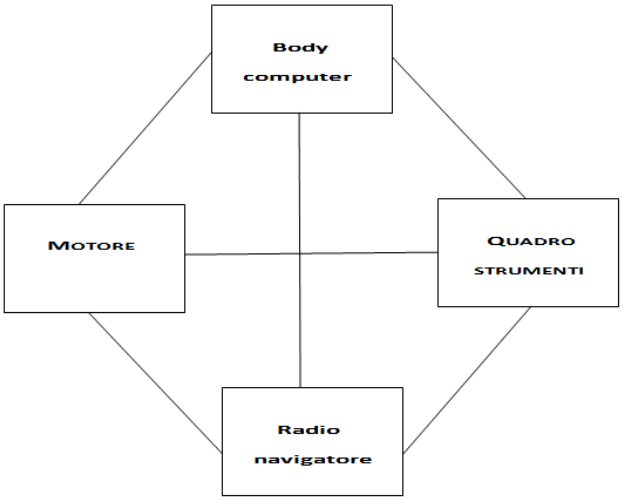

Figura 1.4: Sistema Monofilare

Come si può notare esistono due diverse versioni del sistema monofilare:

- Le connessioni tra due centraline o componenti vengono sostituite da una singola coppia di fili, ed esiste una singola connessione per ogni coppia di centraline connesse (multiplex con linee seriali punto-punto, come mostrato in Figura 1.4;

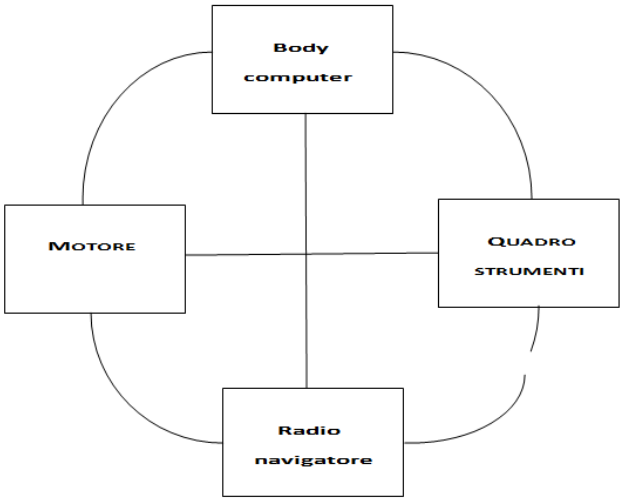

Figura 1.5: Sistema Monofilare a Bus

- Un unico cavo bifilare che connette tra di loro le centraline, come mostrato in Figura 1.5.

Un sistema multifilare è apparentemente più affidabile, in realtà non lo è in quanto la probabilità di rottura di un filo è molto bassa in confronto alla probabilità di guasti dei vari componenti e dei sensori. Il sistema monofilare, in principio, era più complesso e richiedeva maggiori costi di progettazione e sviluppo, oggi la standardizzazione ha permesso di semplificare il tutto. Diventa possibile infatti "simulare" un sistema di dispositivi molto complesso, semplicemente riproducendo il suo comportamento sul bus, valutarne il comportamento ed effettuare le dovute considerazioni. Molte difficoltà a introdurre un sistema a bus o multiplexato risiedevano nel fatto che occorreva definire uno standard unico, in modo da aumentare il numero dei componenti e degli strumenti di sviluppo, riducendo i costi e aumentando le prestazioni.

L'ampio incremento dei sistemi elettronici distribuiti nel settore automobilistico e veicolare, accoppiato con una sempre maggiore richiesta di potenza e comfort, ha creato una serie di nuove opportunità e sfide ingegneristiche.

L'aumento della presenza dell'elettronica nei veicoli è dovuto, da un lato, ad una continua crescita di richieste da parte dei clienti che esigono maggiore sicurezza durante la guida e maggiore comfort dall'altro, alle leggi che mirano a migliorare il comportamento dei gas di scarico e a ridurre il consumo di combustibile.

Le centraline elettroniche che soddisfano a questi requisiti sono da tempo in uso nell'ambito del controllo dei cambi, delle valvole e anche come sistema antibloccaggio

(ABS) o come sistema antislittamento (ASR). La complessità delle funzioni svolte, a tale proposito, rende necessario lo scambio di dati fra le centraline.

In un recente passato, la trasmissione dei dati avvveniva in modo convenzionale attraverso conduttori di segnale dedicati: cablaggio di tipo "point to point". A tale proposito è necessario un certo dispendio soprattutto in termini di cavi, terminali, connettori e quindi spazio e peso considerando la complessità delle funzioni delle centraline.

Le direttive future sono quindi quelle di ridurre la complessità architetturale del veicolo e quindi l'ottimizzazione del veicolo nel suo insieme.

Questo è possibile solo se si collegano in rete le componenti del sistema complessivo a bordo del veicolo, mediante una struttura a comunicazione seriale di dati su bus. Come le LAN collegano computer, queste reti seriali di controllo collegano i dispositivi elettrici ed elettronici dei veicoli.

La riduzione e la facilità del cablaggio aumentano le prestazioni e l'affidabilità del veicolo e riducono il peso, fatto molto rilevante. Infatti è necessario tenere a mente che per ogni $50kg$ di cavi si ha un incremento del consumo di carburante di 0.2 litri per ogni $100km$ percorsi.

La comunicazione seriale nei veicoli ha 4 importanti campi di impiego:

- Collegamento in rete di centraline per la regolamentazione del motore, cambio e freni. La velocità di trasmissione dei dati può assumere valori compresi tra 200 e $10Mbit/s$ per le applicazioni in tempo reale.
- Collegamento in rete di componenti dell'elettronica della carrozzeria e del comfort. Un esempio di queste applicazioni è costituito dal controllo delle luci, dal climatizzatore, dalla chiusura centralizzata e dalla regolazione dei sedili e degli specchi. Particolare attenzione viene data, a questo proposito, ai costi dei componenti utilizzati nei nodi della rete. La velocità di trasmissione in questo caso risulta molto inferiore a quella del caso precedente.
- Collegamento in rete di applicazioni multimedia, quali autoradio, il telefono, dispositivi di navigazione, lettori CD con un'unità di comando centrale a struttura ergonomica.
- Diagnosi a bordo. E' necessaria la presenza di una rete di controllo e di diagnostica per la sicurezza e il buon funzionamento dei sistemi.

Nel settore *Automotive*, soprattutto per motivi di spazio, i principali requisiti del bus di campo riguardano la miniaturizzazione dei dispositivi e a causa dell'elevata

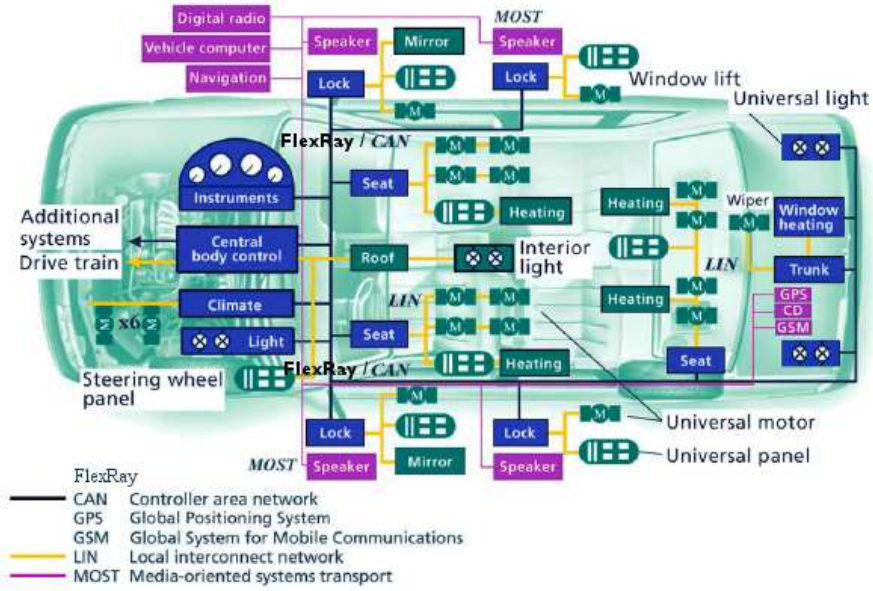

Figura 1.6: Esempio di moderna architettura interna di un veicolo

irradiazione di disturbi, la protezione della trasmissione. Pari importanza riveste inoltre la robustezza meccanica dei dispositivi di campo.

1.1.2 La sicurezza

Negli ultimi dieci anni molta ricerca è stata dedicata alla soluzione di uno dei maggiori problemi della circolazione su strada: *la sicurezza*. La sicurezza stradale è notevolmente aumentata negli ultimi anni, gli incidenti stradali continuano ad aumentare con l'aumentare del traffico (come mostrato in Figura 1.7), mentre le conseguenze degli incidenti, in particolare i morti, diminuiscono. Comunque la cifra di circa 5000 morti all'anno in Italia rimane un tributo alla mobilità individuale ancora molto pesante. Negli incidenti stradali il fattore umano è la causa principale: oltre il 90% è causato da un non corretto comportamento di guida.

Le case automobilistiche, a partire dalla fine degli anni '80 hanno iniziato a collaborare, nel programma di ricerca Europeo PROMETHEUS, al fine di migliorare la sicurezza stradale attraverso lo sviluppo di sistemi di sicurezza "preventiva" che,

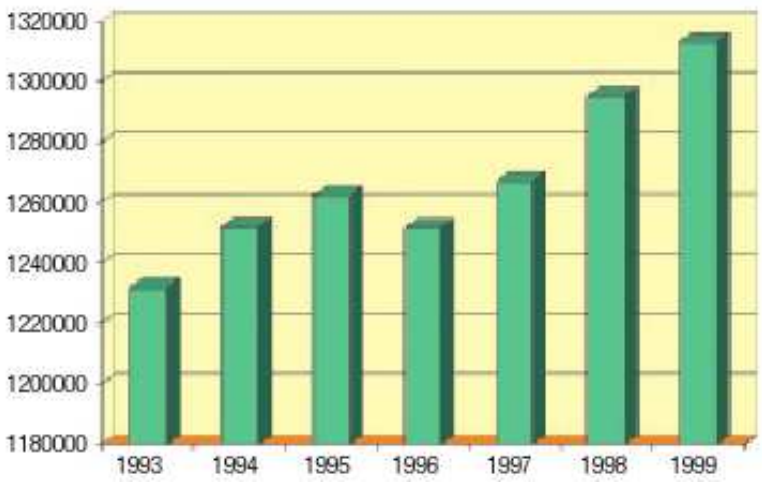

Figura 1.7: Numero incidenti stradali in Europa fino al 2000

grazie a sensori capaci di percepire il traffico circostante, aiutano il guidatore a evitare situazioni pericolose.

Nel migliorare la sicurezza è possibile intervenire su diversi aspetti:

- La sicurezza passiva: ridurre le conseguenze di un incidente.
- La sicurezza attiva: permettere al guidatore il massimo controllo del veicolo, al fine di evitare l'incidente o ridurne le conseguenze.
- La sicurezza preventiva: evitare che si creino le premesse per un incidente.
- L'intervento post-crash: ridurre le conseguenze di un incidente attraverso un tempestivo intervento dei mezzi di soccorso.

I miglioramenti di questi ultimi decenni, in termini di sicurezza stradale sono stati ottenuti soprattutto sulla sicurezza passiva e attiva.

Esistono norme precise che fissano i requisiti minimi da rispettare per un veicolo, su urti standard sia frontali che laterali. Tranne alcuni veicoli di gamma bassa, ormai tutti i veicoli dispongono, di serie, di airbag guidatore e ABS. I trend futuri prevedono di intervenire su tutti i vari aspetti della sicurezza.

Sicurezza passiva

La tendenza futura sarà nel modulare l'intervento dei sistemi di sicurezza passiva, cinture e airbag, in funzione dell'incidente e degli occupanti il veicolo. L'elettronica dovrà fornire soluzioni affidabili ai due problemi critici:

- Rilevare con assoluta sicurezza incidente, con un anticipo tale da permettere un intervento di tipo meccanico, dei sistemi di ritenuta (sensori *pre-crash*).
- Rilevare la presenza, posizione e caratteristiche degli occupanti, per modulare correttamente l'intervento dei sistemi airbag.

Aumenterà, inoltre, l'attenzione agli incidenti che coinvolgono gli utenti della strada più vulnerabili: pedoni e ciclisti. La progettazione dei veicoli, soprattutto nella parte frontale, dovrà in futuro tenere conto anche di questi aspetti, per superare apposite normative in fase di definizione. Gi studi in questo settore saranno utili nello sviluppo di soluzioni attive: per esempio, cofani motore che si sollevano leggermente in caso di urto con un pedone, per attutire un probabile successivo urto della testa del pedone sul cofano stesso.

La sicurezza attiva

Dopo l'ABS, sistema che impedisce alle ruote di bloccarsi in frenata e, quindi, permette di mantenere il controllo della direzione del veicolo anche in situazioni di frenata di emergenza, si stanno diffondendo i sistemi di controllo della dinamica del veicolo. Un guidatore non esperto, di fronte a un ostacolo improvviso, sterza bruscamente con una manovra che spesso può portare a perdere il controllo del veicolo (sbandata, testa coda ecc.) con conseguenze imprevedibili. Grazie ad un'azione di frenatura sulle singole ruote, una centralina elettronica permette di aumentare la capacità del veicolo di modificare la propria traiettoria, senza sbandare. Il guidatore si trova, quindi, a disporre di una capacità di controllo del veicolo che anche un pilota esperto, su un veicolo normale, non avrebbe.

La ricerca si sta ora focalizzando, come detto, sulla tecnologia Drive-by-Wire: eliminando ogni collegamento meccanico tra gli organi di controllo della dinamica del veicolo (ruote, sterzo, sospensioni, motore, cambio, freno e acceleratore) e i comandi (pedale freno e acceleratore, volante) sarà possibile realizzare veicoli con 4 ruote indipendenti, sterzanti e frenanti, sotto il completo controllo di una centralina elettronica.

Il guidatore esprimerà la sua volontà attraverso i comandi, che saranno eseguiti dal veicolo sfruttando al meglio la dinamica possibile, se occorre sterzare a destra per andare a sinistra (e in condizioni di bassa aderenza questo può accadere) il veicolo lo farà senza che il guidatore se ne accorga.

Il livello di affidabilità delle centraline preposte al controllo di tali funzioni deve essere assoluto. Nei sistemi attuali, ABS e VDC, è possibile, qualora si diagnostici un malfunzionamento (anche minimo), disabilitare il controllo: si torna ad una modalità di veicolo "normale", ma pur sempre in grado di frenare e sterzare.

Sicurezza preventiva

Le prime applicazioni di questi sistemi sono nel controllo automatico della velocità. Grazie a radar a microonde o laser, il veicolo riesce a percepire la presenza di un veicolo più lento nella corsia di marcia e rallentare in modo da adeguarsi alla velocità del veicolo che precede. Questa funzione (detta Cruise Control Adattativo) è gia presente sul mercato da alcuni anni. Nel controllo laterale del veicolo sistemi sensoriali basati su telecamere possono aiutare il guidatore a mantenere la corsia di marcia ed evitare uscite di corsia. Una segnalazione, acustica o tattile (per esempio vibrazione al volante) viene generata quando il sistema rileva una imminente, non segnalata, uscita dalla corsia di marcia. *Irisbus*, azienda italofrancese produttrice di autobus, fornisce un sistema che aiuta in questo modo l'autista a mantenere il mezzo nella corsia di marcia e, inoltre, ad accostare con la massima precisione il mezzo durante le fermate.

Sono allo studio sistemi anche per il controllo automatico dello sterzo (mantenimento automatico della corsia, Lane Keeping). I costruttori automobilistici sono molto prudenti su questa funzione, che potrebbe essere interpretata non come un ausilio ma come una vera e propria guida automatica del veicolo.

Quindi questa funzione non potrà probabilmente essere introdotta sul mercato finchè non sarà possibile avere una assoluta affidabilità sia nel riconoscimento della corsia di marcia che nel rilevamento di ostacoli eventualmente presenti sulla stessa.

Altri esempi di funzioni di ausilio alla guida disponibili sul mercato o in fase avanzata di sviluppo sono i seguenti:

- Visione notturna. Telecamere operanti nell'infrarosso consentono al guidatore di avere una migliore percezione in condizioni di bassa visibilità, di notte e con la nebbia.

- Copertura angolo cieco. Gli specchietti retrovisori laterali soffrono dell'angolo cieco, un'area laterale non visibile da parte del guidatore (se non girando la testa). Mediante telecamere e centraline elettroniche di elaborazione immagine è possibile avvisare il guidatore della presenza di un veicolo in fase di sorpasso.

- Ausilio nelle manovre di parcheggio. Sensori di parcheggio sono già ampiamente diffusi su molti veicoli. In futuro, una centralina elettronica rileverà lo spazio tra due veicoli e, se sufficiente, ci aiuterà nella manovra controllando lo sterzo.

Sicurezza post-crash

A incidente avvenuto diventa di importanza vitale che l'informazione sia tempestivamente trasmessa alle centrali di soccorso, per predisporre un tempestivo intervento di ambulanze, vigili del fuoco e polizia. Centraline a bordo veicolo, protette e dotate di alimentazione autonoma, possono eseguire in automatico tale chiamata, tramite un telefono cellulare GSM.

Esempi di questo servizio già avvengono, attraverso centrali di soccorso private e per gruppi specifici di utenti con esigenze particolari, per esempio trasporto valori. Affinché questo servizio si diffonda occorre però che siano definiti degli standard, a livello Europeo, sulla modalità con cui la chiamata viene eseguita. Esistono in questo senso iniziative a livello Europeo di standardizzazione. In particolare si dovrebbe confluire su un unico numero per le chiamate di emergenza.

1.2 Drive-by-Wire vs Brake-by-Wire

Nella dinamica di un veicolo, il controllo accurato dell'azione frenante ha grande importanza. Negli ultimi anni, con il progredire dei sistemi elettronici volti a garantire la stabilità del mezzo e in particolare modo la stabilità durante la frenata, esso ha assunto un ruolo ancora più rilevante. In questo ambito si colloca il progetto X-by-Wire e in particolare il sistema Brake-by-Wire letteralmente "frenare con un filo". Un sistema di questo tipo è inquadrato nella più ampia ottica dei sistemi "Drive-by-Wire", ovvero tutti quei sistemi elettronici di controllo delle funzioni primarie del moto di un veicolo terrestre (combustione del motore, frenatura, sterzata, sospensioni, accensione, etc.).

I sistemi *Drive-by-Wire* devono il loro nome e i loro principi tecnici agli ormai noti e applicati sistemi *Fly-by-Wire*, chiamati in gergo "pilota automatico", che

sono utilizzati da decenni su tutti i tipi di velivoli per controllare il moto degli aerei in particolari condizioni di viaggio. Fino a pochi anni fa sistemi di guida automatica o anche solo assistita di veicoli avevano trovato uno sbocco applicativo solo nel settore militare o nella robotica. Ultimamente invece, grazie soprattutto allo sviluppo dell'elettronica e dei calcolatori elettronici che hanno permesso di disporre di dispositivi molto efficienti a basso costo, anche nel settore dell'*automotive* di serie si è assistito ad un incremento costante degli apparati elettronici di bordo e quindi della presenza di sistemi di assistenza alla guida sempre più complessi e con ruoli sempre più rilevanti: si è passati dall'apertura automatica dei finestrini e delle portiere fino alla gestione dell'iniezione, della frenata (ABS), della trazione e della sterzata (servosterzo, idroguida).

Per facilitare l'integrazione dei sistemi elettronici è nell'interesse delle industrie di automobili sostituire il tradizionale sistema idraulico o meccanico con un sistema meccatronico distribuito fault-tolerance. Questi nuovi sistemi sono appunto chiamati X-By-Wire, dove X rappresenta l'applicazione a cui ci si sta riferendo, come Braking o Steering.

By-Wire denota il sistema di controllo che sostituisce i tradizionali collegamenti idraulici o meccanici con collegamenti tra le unità di controllo che guidano gli attuatori elettronici, in particolare quando i sistemi critici sono controllati da un canale di comunicazione vengono definiti "by Wire".

L'industria automobilistica a differenza dell'industria avionica può puntare sui sistemi X-By-Wire perché la produzione è destinata a tutte le persone, quindi risulta più facile contenere le spese e investire in ricerca.

Quindi risulta fondamentale il bilanciamento tra affidabilità e costo-efficacia per promuovere il sistema X-by-Wire ora e in futuro. Lo sviluppo di nuovi sistemi X-by-Wire rappresenta un grande investimento di risorse per le singole aziende. Per ridurre i costi, le industrie Europee di veicoli hanno condiviso gli sforzi nella ricerca per creare degli standard comuni e stabilire dei framework per i sistemi X-by-Wire.

Un importante punto di svolta lo si è avuto nella metà degli anni '90, quando sistemi elettronici di controllo della trazione, della frenata, dell'iniezione e del meccanismo del cambio sono stati pesantemente introdotti nel campo delle competizioni sportive. Questo successe tra Volvo, Daimler-Benz (chiamata ora Daimler-Chryler), Centro Ricerche Fiat, Ford Europe, Bosch, Mecel e Magneti Marelli, l'Universty of Technology di Chalmers e l'University of Technology di Vienna che utilizzarono tra il 1996 e il 1998 i finanziamenti messi a disposizione dall'EU. In realtà possono distinguersi due grandi gruppi che sviluppano due diverse versioni di protocollo *Time-triggered*:

- Il *Time Triggered Architecture Group*, che include PSA Peugeot Citroen, Audi, Volkswagen, Hneywell and Delphi Automotive System, utilizza il *Time Triggered Protocol* (TTP) originariamente sviluppato all'università di Vienna.
- L'altro gruppo è il *FlexRay Consortium* che ha sviluppato appunto il protocollo FlexRay, basato in parte sul sistema *ByteFlight* e sull'architettura *time-triggered*. Questo protocollo, come si vedrà in seguito, offre una flessibilità che nel *time-triggered* manca. I membri di questo gruppo sono General Motors, DaimlerChrysler, BMW, Motorola, Philips, Semicondutors e Bosch Automotive Group.

Proprio l'utilizzo e lo sviluppo della tecnologia in questo settore ha permesso da un lato di renderla più affidabile e performante, dall'altro di renderla nota al grande pubblico, vero grande motore dell'evoluzione del mercato automobilistico.

Tra tutti i settori applicativi del drive-by-wire nell'autoveicolo, la sterzata e la frenata sono senz'altro quelli più delicati: da un lato un errore o un collasso del sistema del controllo o della frenata porterebbero ad essere l'auto completamente incontrollabile, con ovvie conseguenze sulla sicurezza del pilota e dei passeggeri. Un altro possibile passo in avanti sarebbe poi l'eliminazione delle connessione meccaniche tra il volante e le ruote, che i crash-test dimostrano essere tra i pericoli maggiori in caso di urto frontale per la penetrazione all'interno dell'abitacolo, a meno di progettazioni particolari che però limitano molto l'efficienza del design del telaio. Si noti come queste evoluzioni del sistema di guida non cambino nella sostanza il rapporto tra mezzo e pilota, che resterebbe comunque il governatore principale del veicolo e manterrebbe lo stesso tipo di *feeling* con il moto del mezzo.

Un sistema X-by-Wire deve essere necessariamente fault tolerant e real-time performance, cioè deve in pratica avere un sistema sicuro e pronto a qualsiasi situazione critica. Questi innovativi sistemi Brake-by-Wire offrono interessanti possibilità per un completo controllo attivo del sistema frenate della vettura, garantendo un maggiore livello di integrazione. Offrono inoltre migliori soluzioni di alloggiamento, eliminando i problemi derivanti dalla componente idraulica, con l'ulteriore vantaggio ambientale determinato dall'assenza di fluido freni.

Il problema fondamentale di questo tipo di sviluppo è garantire la sicurezza del veicolo durante la guida e porre l'autista nelle migliori condizioni per essere sempre in grado di avere il pieno controllo dei dispositivi che lo governano.

Il miglioramento delle prestazioni degli autoveicoli e il loro successo è basato su una progettazione integrata della meccanica dell'autoveicolo e dei suoi sistemi di controllo e quindi risultano indispensabili strumenti di modellazione dinamica

e di studio dei sistemi di automazione che consentano le rapida realizzazione del prototipo tramite simulazione al computer.

Fino poco tempo fà i sistemi X-by-Wire comunicavano attraverso un protocollo time-triggered chiamato TTP/C, molto performante. Tuttavia oggi si cerca di sviluppare e utilizzare un nuovo protocollo chiamato FlexRay, che sarà l'argomento su cui si focalizzerà la tesi.

1.3 Formulazione del problema

L'obiettivo fondamentale di questa tesi è quello di modellare, simulare e implementare un semplice sistema Brake-by-Wire su FlexRay. Il concetto di FlexRay sarà accuratamente sviluppato, in riferimento soprattutto a quanto concerne la fault tolerance. Infatti la realizzazione del progetto evidenzia come venga sviluppato un sistema frenante di un autoveicolo, ossia ci sarà un nodo principale che comunica con altri 4 nodi che corrispondono alle 4 ruote del veicolo.

La comunicazione avviene grazie al protocollo di comunicazione FlexRay. Protocollo in via di sviluppo destinato a sostituire il CAN o comunque sia ad affiancarlo. Naturalmente il sistema ha i suoi limiti, dovuti soprattutto al fatto che essendo un argomento ancora in via di sviluppo, presenta problematiche non semplici da risolvere (in tempi brevi), e una letteratura non abbondante. C'è da dire che questo lavoro è il punto di partenza da cui Akhela s.r.l. (azienda in cui è stata sviluppata la tesi) sta iniziando il lavoro di ricerca e di sviluppo sui sistemi di frenata che comunicano attraverso FlexRay.

Attualmente gli impianti frenanti si dividono essenzialmente in due grandi categorie, freni a tamburo (più economici) e i freni a disco, più costosi ma certamente più performanti. Con l'introduzione dei sistemi Brake-by-Wire la modulazione della frenata potrà essere applicata direttamente all'impianto frenante in caso di bloccaggio della ruota dando maggiore libertà ai progettisti e non intervenendo solo parzialmente sulla pressione dell'impianto idraulico del freno come avviene negli impianti classici. Con un impianto Brake-by-Wire, infatti, la frenata viene gestita elettronicamente dal controllore del freno elettrico, giocheranno un ruolo fondamentale i sensori e gli attuatori e pertanto saranno necessarie logiche di controllo più raffinate.

Capitolo 2

Dinamica del veicolo

2.1 Premessa

In questo capitolo verranno, sufficientemente dettagliato, spiegate quale leggi regolano il moto del veicolo. In particolare si vuole evidenziare come cambia lo studio di un sistema in cui si tiene conto di tutte le possibili variabili ambientali rispetto a modelli in cui vengono fatte delle semplificazioni. Viene fatto questo perché il modello sviluppato nella tesi è realizzato in base a delle opportune considerazioni iniziali, quindi la visione del capitolo ha lo scopo di far ragionare sulle possibili modifiche da attuare al modello nel caso non valgano alcune semplificazioni iniziali. I documenti presi in considerazione per questo capitolo, e a cui ci si può riferire per un approfondimento sono [26], [12], [13], [19], [20], [15], [14].

2.2 Attrito

2.2.1 Attrito fra due corpi

In questa parte del capitolo viene data una descrizione del fenomeno fisico dell'attrito a livello macroscopico e microscopico e vengono illustrati diversi comportamenti che tengono conto delle sue caratteristiche fisiche.

Per capire meglio quali sono i fattori che influenzano l'attrito, è necessario concentrarsi sulle superfici a contatto dei corpi, tra le quali si sviluppa attrito.

Due corpi possono essere a contatto *conforme* o *non conforme*. In un contatto conforme l'area macroscopica di contatto è determinata dalle dimensioni dei corpi,

come mostrato nelle Figure2.1, dove la pressione esercitata dal corpo A sul corpo B è proporzionale alle forze esterne e alla forza peso.

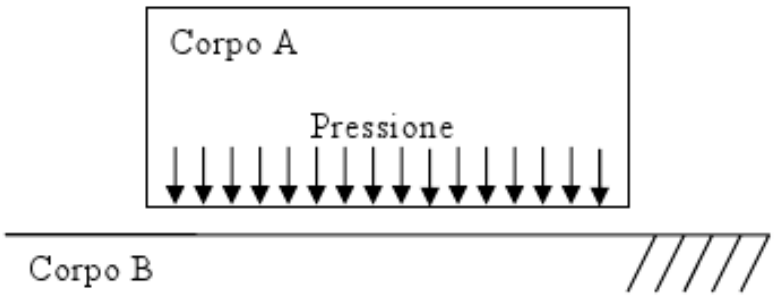

Figura 2.1: Contatto conforme

In particolare, l'area apparente di contatto tra solido e solido a livello macroscopico, è quella della superficie inferiore del corpo A e se su questo agisce una forza di carico perpendicolare alla superficie di contatto, il corpo B fornisce una forza di reazione che la controbilancia. Le frecce disegnate in Figura 2.1 rappresentano, a livello macroscopico, la distribuzione delle forze al contatto, che si assume uniforme. Nel caso in cui i due corpi abbiano raggi di curvatura diversi, allora si parla di contatto non conforme tra corpi, come si vede nella Figura 2.2: Comunque il

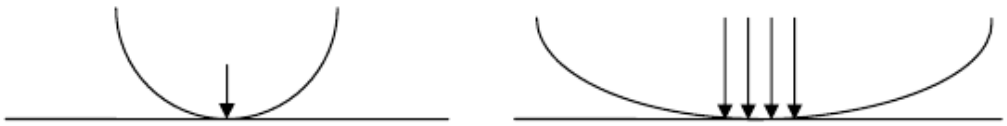

Figura 2.2: Contatto non conforme

contatto si realizza apparentemente su un'area di piccole dimensioni, in cui lo sforzo non è uniforme, ma è massimo nel punto centrale di contatto e decresce verso la periferia. Inoltre si distinguono caso *ideale* e *reale* di contatto non conforme, in quanto idealmente l'area di contatto si riduce ad un punto, mentre nella realtà la

zona macroscopica di contatto è ben più grande ed è dipendente dal carico che i due corpi devono sopportare e dalla rigidità dei materiali.

Contatti *non conformi* si hanno soprattutto tra parti rotanti accoppiate mediante cuscinetti a sfere e tra i denti delle trasmissioni ad ingranaggi come i riduttori di velocità per i motori, in cui la larghezza tipica della zona di contatto non conforme è dell'ordine dei millimetri.

L'analisi macroscopica non consente tuttavia di comprendere il fenomeno dell'attrito. Analizzando il contatto a livello microscopico è possibile constatare che le due superfici, anche se apparentemente lisce, possiedono in realtà delle microscopiche asperità irregolari, le cui giunzioni realizzano il contatto. Queste irregolarità delle superfici fanno si che la vera area di contatto sia molto più piccola rispetto a quella apparente a livello macroscopico in cui le superfici appaiono lisce, così che i veri punti di contatto sono quelli in cui le asperità delle due parti si congiungono.

Le irregolarità si deformano in modo da generare delle aree di contatto locali che sopportino il carico totale. La pressione nei punti di giunzione può essere considerata in prima approssimazione proporzionale al coefficiente di rigidità del materiale, mentre l'area di contatto d'altra parte è proporzionale al carico totale.

L'attrito risulta proporzionale alla somma delle forze di rottura delle giunzioni nelle zone di contatto, in modo tale che l'aumento dell'area microscopica di contatto dovuto a maggior carico incrementa il numero delle forze di rottura, mentre la singola forza resta costante e quindi l'attrito risulta in proporzionalità diretta con il carico.

La presenza di lubrificanti, olio o grasso, o la formazione di ossido di metallo, crea degli strati superficiali di scivolamento. I lubrificanti vengono aggiunti per controllare l'attrito e ridurre l'usura delle parti a contatto; in tal modo si diminuiscono le forze di rottura delle giunzioni delle asperità e così anche l'attrito. Essi infatti permettono la creazione di una barriera fluida tra le due superfici di metallo a contatto che ostacola la formazione di giunzioni ed attenua i fenomeni di fatica ed usura dei materiali.

La barriera fluida all'interfaccia tra le due superfici può essere mantenuta mediante due principali tecniche. Una prima tecnica, di più difficile realizzazione pratica, è chiamata lubrificazione idrostatica, ed è volta a mantenere sotto pressione il lubrificante in modo sostanzialmente indipendente dalla velocità relativa delle parti a contatto.

Un'altra tecnica, assai più frequentemente utilizzata, è quella che va sotto il nome di lubrificazione idrodinamica: essa richiede solo un bagno di olio o grasso,

ma soffre della limitazione che il fluido è mantenuto solo al di sopra di una velocità relativa minima; quindi per basse velocità si ha un contatto solido-solido.

2.2.2 Relazione tra attrito e velocità

Studiando la dipendenza della forza di attrito dalla velocità e con diversi lubrificanti, a basse velocità di scivolamento, è possibile individuare quattro diversi regimi di attrito in funzione di altrettanti regimi di lubrificazione, che individuano la curva di Stribeck, come mostrato in Figura 2.3:

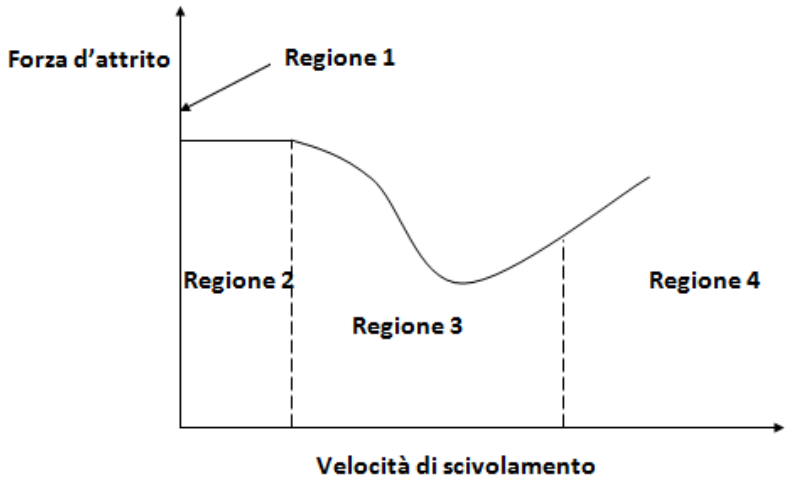

Figura 2.3: Curva di Stribeck generalizzata

I regimi indicati in Figura 2.3 sono i seguenti:

- Regime 1, regime di attrito statico, nessun scivolamento tra i corpi, ma solo una deformazione elastica delle giunzioni tra asperità.
- Regime 2, confine di lubrificazione.
- Regime 3, parziale lubrificazione.
- Regime 4, piena lubrificazione.

2.2.3 Attrito statico e presliding displacement

Se si applica una forza esterna che tende a far muovere le due superfici una rispetto all'altra, inizialmente e finchè la forza non supera la massima forza di attrito, le giunzioni tra la asperità si deformano elasticamente dando luogo a scorrimenti dell'ordine dei micron, denominati *presliding displacement*.

A velocità nulla è presente l'attrito statico e le giunzioni tra le irregolarità possono essere modellizzate come delle micro-molle che mantengono unite le due parti e che si deformano elasticamente determinando uno spostamento di pre-scivolamento. Questo fino a quando la forza applicata, responsabile del movimento, non supera la forza di rottura degli elementi elastici dando luogo ad un vero e proprio movimento, come mostrato in Figura 2.4:

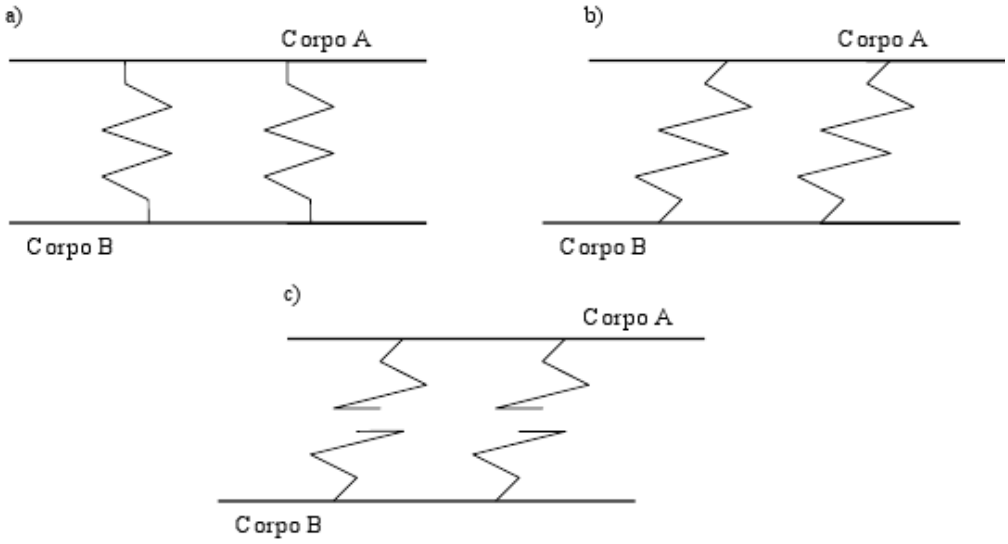

Figura 2.4: Modellazione dell'attrito statico e del prescivolamento

Ad un micro-scorrimento di ampiezza x rispetto alla posizione di riposo le micromolle si inclinano e si estendono, esercitando una forza di richiamo lungo la direzione dello scorrimento proporzionale allo spostamento x, ma rimanendo ben saldate alle estremità (effetto Dahl).

Tali saldature si rompono quando la forza esterna supera la massima forza di attrito, condizione cui corrisponde l'inizio del moto relativo macroscopico (breakaway). Questi fenomeni microscopici si manifestano a livello macroscopico con il cosiddetto attrito statico, per il quale si ha da un lato una brusca transizione fra quiete e moto relativo quando la forza applicata è superiore ad un certo valore (forza o coppia di *stiction*) e dall'altro un brusco arresto del moto relativo quando, annullandosi la velocità relativa, le asperità in contatto si deformano plasticamente formando di nuovo le giunzioni.

Lo spostamento di pre-scivolamento dipende dalla forza tangenziale applicata al corpo A. Essendo il comportamento delle irregolarità come quello di una molla con costante di rigidezza tangenziale, si può scrivere:

$$F_{applicata} = -Kx \tag{2.1}$$

La costante K di rigidezza tangenziale della molla dipende dalla geometria delle asperità, dall'elasticità del materiale e dalla forza normale applicata. In prima approssimazione, se si considera costante lo spostamento di pre-scivolamento alla rottura X_r (cioè la massima deformazione tangenziale delle asperità), allora la costante k è data da:

$$K = \frac{F_{applicata}}{x_r} \tag{2.2}$$

Se la forza normale varia e il coefficiente di attrito statico rimane approssimativamente costante, allora K risulta proporzionale alla forza normale. La conoscenza dello spostamento di pre-scivolamento è importante per il controllo nelle applicazioni ad alta precisione.

2.2.4 Fenomeni di attrito

L'attrito presente tra le parti in movimento dei motori elettrici, delle trasmissioni, dei cuscinetti, è un fenomeno che limita le prestazioni dei sistemi di controllo. La modellizzazione e l'identificazione dell'attrito può consentirne una corretta compensazione, ciò consente di migliorare la precisione del controllo di posizione negli azionamenti elettrici.

La natura non lineare dell'attrito fa sì che un classico regolatore di posizione PID, posto in un anello di retroazione, non sia sufficiente per raggiungere elevate prestazioni nell'inseguimento della traiettoria in tutti quei sistemi meccanici costituiti da masse collegate mediante trasmissioni o in qualche modo elastiche (esempio macchine utensili, robot).

Non potendo linearizzare univocamente questo fenomeno, si cerca di studiarlo approfonditamente: più sono le informazioni raccolte, maggiori sono le possibilità di studiare algoritmi di compensazione per la regolazione digitale, che permettono il raggiungimento di prestazioni migliori.

Possono essere presentati e discussi diversi modelli dell'attrito, poiché esso assume comportamenti differenti, ad esempio in funzione del regime di lubrificazione in cui si lavora, oppure in funzione della velocità di scivolamento delle parti meccaniche in movimento. Lo studio dell'attrito è in generale un argomento complesso in quanto ancora oggi non si è riusciti a definire in modo completo i fenomeni microscopici che portano agli effetti che un osservatore può evidenziare a livello macroscopico.

In letteratura sono stati proposti molti modelli che permettono di descrivere i fenomeni legati all'attrito; essi sono sia di natura empirica, cioè modelli matematici non basati su spiegazioni fisiche che cercano di riprodurre il più fedelmente possibile le conseguenze provocate dall'attrito, sia derivate da teorie che cercano di analizzare su basi fisiche le sue cause.

E' necessario dire che tutti questi modelli possono essere divisi in due categorie:

- Modelli di attrito cosiddetti statici come il modello Stribeck, il modello di Karnopp
- Modelli dinamici quali il modello Dahl, LuGre e Leuven.

Per i modelli statici non si considerano in alcun modo gli aspetti di memoria dell'attrito, i quali, per contro, sono utilizzati in quelli dinamici mediante la descrizione di dinamiche aggiuntive tra velocità e forze d'attrito. Nonostante i modelli dinamici diano una descrizione più accurata dell'attrito, sono molto difficili da implementare nel modello del freno elettrico e da far simulare il comportamento reale. Il vantaggio che si avrebbe nell'utilizzare tali modelli è soprattutto dovuto alla più efficace descrizione dell'attrito durante la fase di attrito statico e dell'inizio del movimento, poi di fatto sono proporzionali alla velocità, almeno fino al raggiungimento di una certa soglia. Data la loro natura, però, tali fenomeni sono visibili solo andandoli ad analizzare su scala molto piccola.

2.2.5 Modelli statici d'attrito

L'osservazione dei fenomeni prodotti dall'attrito ha portato, già dai primi studi, alla definizione di modelli dell'attrito coulombiano, viscoso e statico. Le loro possibili combinazioni portano a quelli che spesso vengono definiti come modelli classici

dell'attrito. Nella Figura 2.5 sono presentati due delle possibili e più frequenti combinazioni:

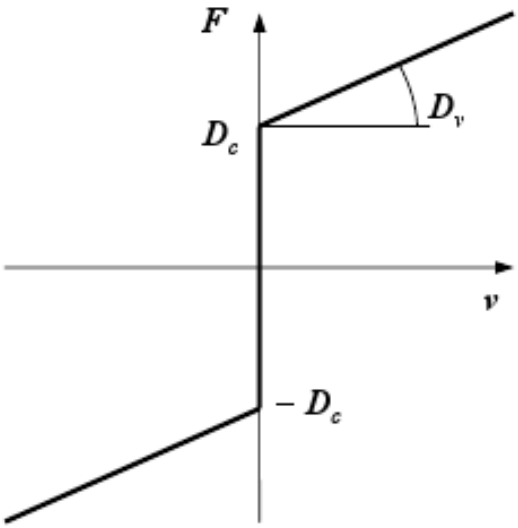

Figura 2.5: Attrito coulombiano e viscoso

In questa classe di modelli la forza di reazione prodotta dall'attrito viene ricavata da una mappa statica, funzione della velocità di scorrimento relativo tra le due superfici a contatto. A riposo però, l'attrito non può essere descritto solo dalla velocità di scorrimento, ma risulta essere funzione anche della somma delle forze esterne; funzione i cui limiti inferiore e superiore sono dati dal valore massimo dell'attrito statico. In virtù di questo si parla d'attrito statico, coulombiano e viscoso. Con riferimento alla Figura 2.6, il parametro F_s, è detto attrito di primo distacco o attrito statico ed ha un valore sensibilmente superiore all'intercetta con l'asse delle ordinate della porzione di curva con pendenza D_v; quest'ultima rappresenta l'attrito viscoso, mentre D_c è pari al valore dell'attrito coulombiano.

Osservando il grafico della forza di attrito in funzione della velocità ci si accorge subito che è presente una discontinuità proprio in corrispondenza del punto a velocità nulla. Ciò ovviamente deriva dal fatto che attraversando tale punto, cioè invertendo la direzione del moto, si inverte la forza di attrito. Tuttavia non è detto che la caratteristica forza-velocità sia simmetrica soprattutto in prossimità di velocità nulle. Nel caso del freno elettrico infatti il coefficiente di primo distacco in caso di spostamenti negativi, cioè quando si vuole far arretrare il pistone, è più ele-

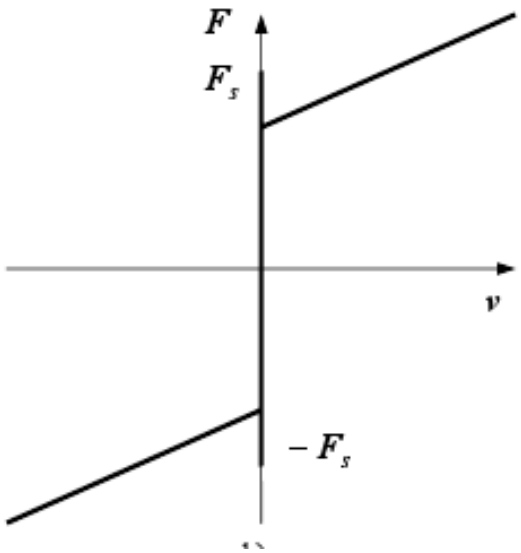

Figura 2.6: Attrito statico, coulombiano e viscoso

vato rispetto all'attrito statico di compressione; ciò è sicuramente dovuto alla non completa reversibilità della vite a ricircolo.

Quando il valore della coppia generata dal motore è minore della forza di reazione del *caliper*, la vite è sottoposta ad una forza che tende a farla arretrare. Data però la natura del componente, è facile pensare che si generino degli attriti di tipo statico elevati che rendono quindi più difficile l'arretramento rispetto all'avanzamento del pistone.

Si può allora definire un modello asimmetrico dell'attrito in cui il valore assoluto dell'attrito statico per velocità negative F_s è maggiore del valore dell'attrito statico per velocità positive F_s ed è rappresentato nella Figura 2.7. Un possibile miglioramento di questi modelli si può ottenere andando a considerare anche l'effetto Stribeck. Il suo utilizzo permette di modellare la variazione nella lubrificazione che si osserva quando le due superfici iniziano a scorrere tra di loro. La curva di Stribeck evidenzia il comportamento particolare che assume l'attrito viscoso quando si verifica una parziale lubrificazione delle superfici di contatto. Si possono presentare diversi tipi di lubrificazione in base alla velocità che generano diversi tipi di attrito.

In particolare possiamo avere parziale lubrificazione quando una delle due parti si muove con velocità maggiore, qui si viene a formare uno strato di fluido che

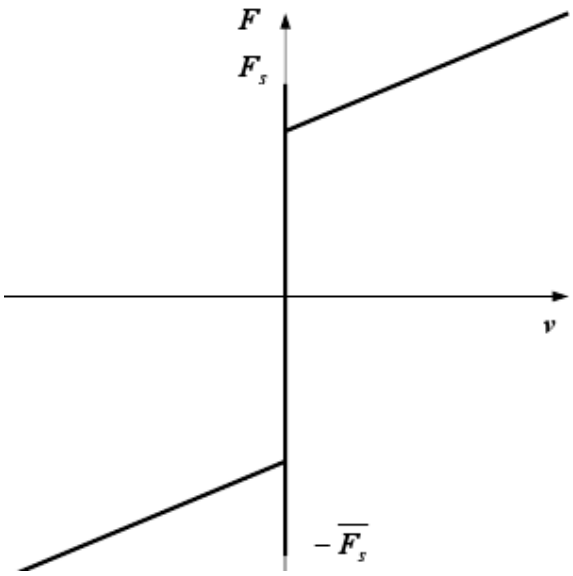

Figura 2.7: Modello d'attrito asimentrico, coulombiano e viscoso

tende a far diminuire l'attrito. Quando si ha piena lubrificazione non c'è contatto tra le due parti che si muovono. Esso è rappresentato, in corrispondenza di velocità molto basse, dal tratto di pendenza negativa della curva ed è all'origine della nascita dei fenomeni di *stick slip* (avanzamento a scatti), responsabili dell'instabilità legata all'insorgere di cicli limite nei sistemi controllati con regolatori PID. Per tali ragioni risulta importante identificare e compensare il comportamento dell'attrito, al fine di eliminare i problemi ad esso legati. Il modello di Stribeck, che descrive tale effetto, sostituisce la discontinuità presente tra attrito statico e attrito coulombiano, con una curva continua a pendenza negativa, come mostrato nella Figura 2.8.

Questi modelli evidenziano che a velocità nulla esiste una discontinuità a gradino della forza d'attrito, la quale è origine dei problemi d'instabilità negli algoritmi di controllo che usano una misura della velocità calcolata tramite una derivazione numerica del segnale di posizione campionato e la discontinuità presente a velocità nulla, inoltre può portare alla *non unicità* della soluzione dell'equazione del moto. Una possibile soluzione potrebbe essere quella di approssimare o addolcire la discontinuità a velocità nulla con una funzione a pendenza finita, come mostrato nella Figura 2.9.

Questa soluzione può essere considerata accettabile, ma solo in applicazioni in cui la transizione della condizione di riposo è occasionale, ad esempio in situazioni in cui si ha un regime a velocità costante o comunque non nulla, dato che non porterebbe a gravi errori. Nel caso di un freno questa non può essere considerata

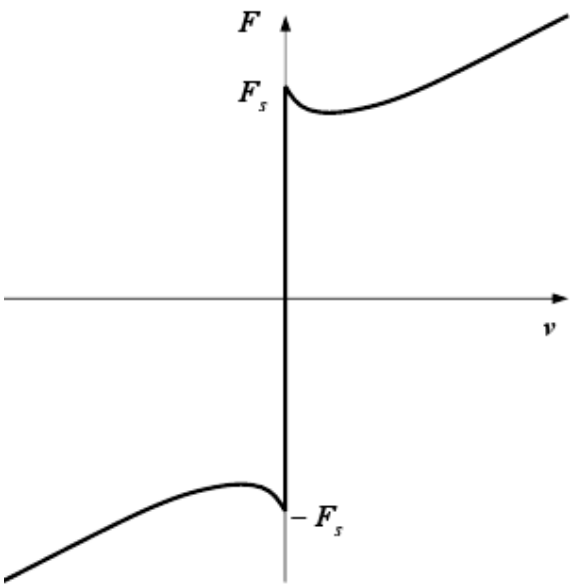

Figura 2.8: Modello di attrito con effetto Stribeck

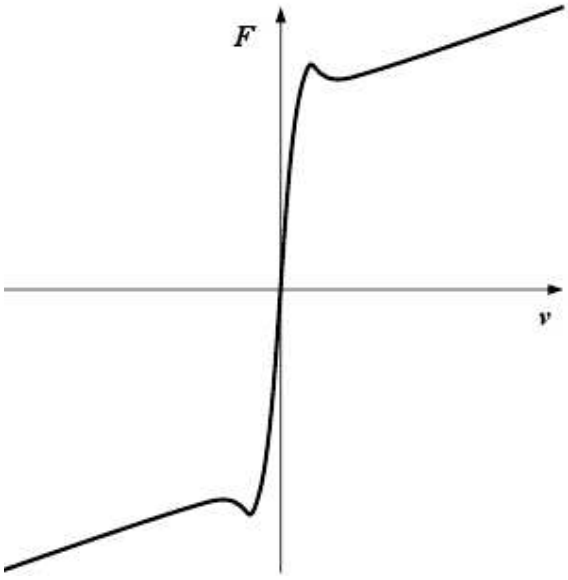

Figura 2.9: Modello d'attrito con curva a pendenza finita

certo la situazione comune ed inoltre l'utilizzo di tale modello porterebbe ad avere accelerazioni non nulle (e quindi un movimento del pistone) anche per valori delle forze esterne non superiori al valore massimo della forza di attrito statico.

Un'altra comune tecnica è quella di impiegare una funzione di commutazione,

come la funzione segno definita dalla 2.3:

$$sign(\dot{x}) = \begin{pmatrix} -1 & \text{se } \forall \dot{x} < 0 \\ 0 & \text{se } \forall \dot{x} = 0 \\ +1 & \text{se } \forall \dot{x} > 0 \end{pmatrix} \tag{2.3}$$

Anche in questo caso si hanno però dei problemi, nello stabilire il momento preciso durante il quale si ha il passaggio per lo zero. Oltre a tali problemi di integrazione numerica, questo tipo di funzione non descrive in alcun modo il fenomeno di attrito statico, dato che forze esterne più piccole del livello di attrito coulombiano non sono compensate in alcun modo a velocità nulla da equivalenti forze di attrito.

Per ovviare ai problemi di robustezza numerica e rivelazione di attraversamento dello zero viene preso in considerazione il modello proposto da Karnopp mostrato in Figura 2.10.

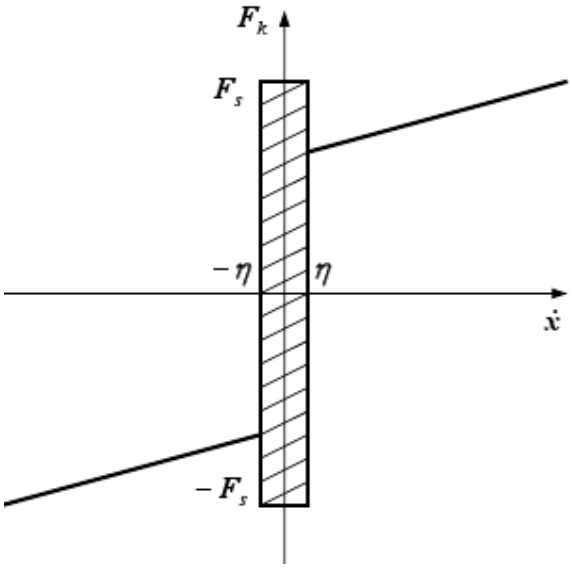

Figura 2.10: Modello d'attrito di Karnopp

Questo modello presenta una *stick region*, cioè un campo di velocità a cavallo dello zero in cui la forza di attrito statico può variare da un valore minimo a un massimo. Con questo modello si evita la ricerca precisa del punto a velocità zero definendo un intorno di tale punto, dato che durante la computazione numerica difficilmente viene calcolato un valore perfettamente nullo. All'esterno di questa stretta regione l'attrito è normalmente funzione della velocità e si parla ancora di attrito viscoso. All'interno invece, si considera nulla la velocità di scorrimento tra le superfici a contatto dei due corpi considerati. In questa regione la forza d'attrito dipende

da una funzione delle altre forze presenti nel sistema, forze che sono necessarie per mantenere a zero la velocità del sistema.

$$F_k(\dot{x}, F) = \begin{pmatrix} F_k(\dot{x}) & \text{se } \forall \mid \dot{x} \mid \geq \eta \\ F_k(F) & \text{se } \forall \mid \dot{x} \mid < \eta \end{pmatrix} \tag{2.4}$$

Il modello può essere descritto da un set di tre equazioni differenziali ordinarie, una per la fase di movimento, una per la fase di attrito statico e una per quella di transizione tra le due fasi precedenti. In questo modo è lo stato del sistema a indicare la fase nella quale ci si trova e quale delle tre equazioni deve essere valutata per ottenere la forza di attrito.

All'esterno della regione definita precedentemente il sistema e nella fase di movimento e verrà applicata la mappa statica funzione della velocità.

All'interno il modello commuta da un'equazione all'altra in base alle valutazioni effettuate sulla forza esterna. Se queste non superano il livello massimo della forza di attrito statico il sistema è fermo e viene bilanciata la forza esterna per mantenerlo tale. Altrimenti il sistema è nella fase di transizione e la forza di attrito è pari al valore massimo della forza d'attrito statico.

Se il sistema si trova in questa fase, necessariamente questo tende ad uscire dalla regione evitando il problema dell'unicità della soluzione. Il problema della definizione dell'intorno del punto a velocità zero (η) si presenta solamente al momento della simulazione.

2.2.6 Modelli dinamici di attrito

Dato che un modello statico dell'attritonon consente di modellizzare nè fenomeni di isteresi, nè la variazione della forza di distacco, nè i presliding displacement, è spesso necessario considerare modelli di tipo dinamico.

Il primo modello dinamico che è stato introdotto e sviluppato con lo scopo di simulare e controllare sistemi con attrito è il modello di Dahl.
Tale modello descrive il comportamento dei sistemi con attrito in regime di *presliding displacement*. Dahl effettua molti esperimenti sull'attrito in sistemi con cuscinetti a sfere e scopre che l'attrito tra cuscinetti sferici è molto simile all'attrito solido piano, riscontrando contatti di metallo fra le superfici che danno origine al fenomeno dinamico del prescivolamento.

Il modello Dahl approssima il fenomeno del presliding con la caratteristica isterica forza-spostamento di Figura 2.11, utilizzando la forza d'attrito come variabile

di stato in relazione allo spostamento relativo dei corpi.

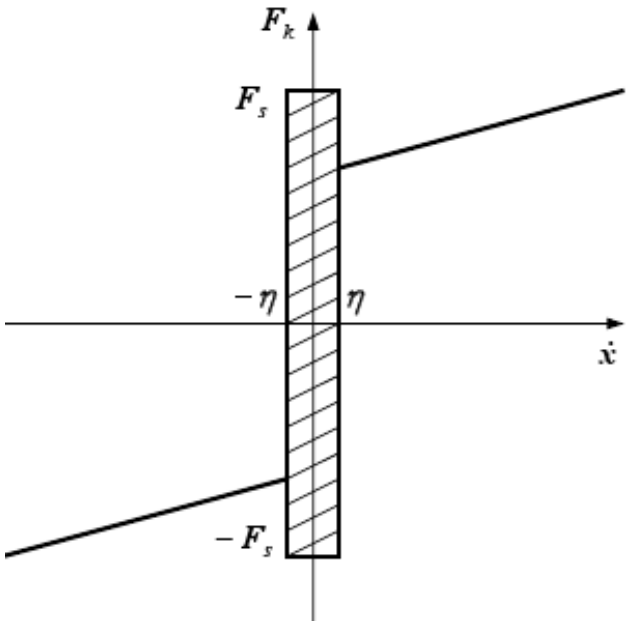

Figura 2.11: Attrito in funzione dello spostamento per effetto Dahl

Il modello è descritto dall'equazione differenziale 2.5:

$$\frac{dF}{dx} = \sigma(1 - \frac{F}{D_c} sign(\dot{x}))^{\alpha} \tag{2.5}$$

dove D_c è la forza di Coulomb, σ è il coefficiente di rigidà delle giunzioni tra i materiali a contatto e α è il fattore di forma della curva di sforzo, che normalmente viene posto uguale a uno; per $\alpha > 1$ la curva è più angolata. Il modello è solo funzione dello spostamento x e del segno della velocità relativa $\dot{x}$. La formula 2.5 può essere riscritta in funzione del tempo:

$$\frac{dF}{dt} = \frac{dF}{dx}\frac{dx}{dt} = \sigma((1 - \frac{F}{D_c} sign(\dot{x}))^{\alpha})\dot{x} \tag{2.6}$$

Il modello Dahl generalizza l'attrito coulombiano, ma non include nè l'effetto Stribeck, nè l'attrito statico. Un modello dinamico capace di interpretare la maggior

parte dei fenomeni connessi all'attrito osservati sperimentalmente è il cosiddetto modello *LuGre* (acronimo derivato dalle città di Lund e Grenoble, sedi delle università cui sono affiliati i ricercatori che hanno proposto ed analizzato il modello). Questo modello combina il modello Dahl, con la caratteristica di possedere anch'esso una variabile di stato e l'effetto Stribeck. Tale modello interpreta il contatto come l'interazione di due spazzole con setole elastiche, come in Figura 2.12:

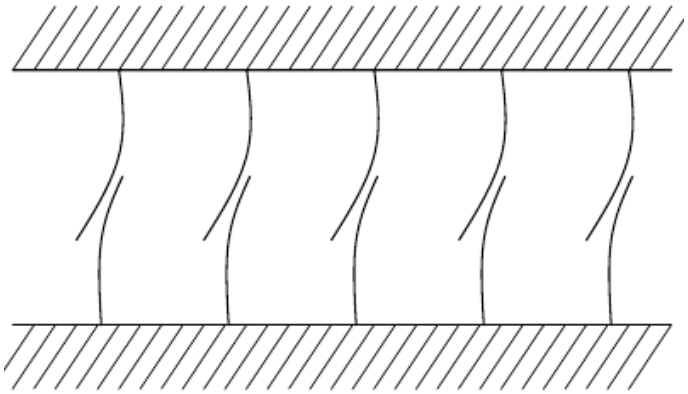

Figura 2.12: Spazzole elastiche

Applicando una forza tangenziale le setole flettono, originando la forza di attrito e, se la forza tangenziale è sufficientemente elevata, la flessione è tale da consentire lo scivolamento delle spazzole. Le equazioni che caratterizzano questo modello d'attrito sono:

$$\frac{dz}{dt} = \dot{z} = \dot{x} - \sigma_0(\frac{|\dot{x}|}{s(\dot{x})})z \tag{2.7}$$

$$F = \sigma_0 z + \sigma_1 \dot{z} + D_v \dot{x} \tag{2.8}$$

$$s(\dot{z}) = D_c + (F_s - D_c)\exp(-(\frac{\dot{x}}{\dot{x}_s}))^\delta \tag{2.9}$$

dove z è la flessione media delle spazzole che costituisce la variabile di stato del modello. Il primo termine a secondo membro della relazione 2.7, fornisce un contributo alla flessione proporzionale all'integrale della velocità relativa, mentre il secondo garantisce il raggiungimento di un valore limite di flessione $\dot{z}$ per velocità costante, che è pari a:

$$s(\dot{z}) = \frac{s(\dot{x})sign(\dot{x})}{\sigma_0}\exp(-(\frac{\dot{x}}{\dot{x}_s}))^\delta \tag{2.10}$$

La funzione $s(\dot{z})$ definita nella formula 2.9 introduce la forza di primo distacco (stiction) F_s, che rappresenta la forza minima necessaria per iniziare il moto e la forza di attrito coulombiano D_c. Questa funzione permette anche di modellizzare l'effetto Stribeck tramite la funzione esponenziale, dove $\dot{x_s}$ è la velocità di Stribeck e δ è il fattore di forma.

Il parametro δ_0, valore che assume valori alti, descrive la rigidezza delle setole, mentre i termini dipendenti dal parametro di smorzamento δ_1 e dal coefficiente di attrito viscoso D_v descrivono gli effetti dissipativi. Se la flessione media delle spazzole si considera costante ci si riconduce a un modello statico poiché $\dot{z}$ è pari a zero (modello di LuGre).

2.3 Dinamica del veicolo

Per poter implementare un controllo automatico, per prima cosa si deve avere una conoscenza del sistema fisico da controllare, capirne le problematiche e formularne un modello matematico che ne descriva le caratteristiche dinamiche nel modo più completo ed esauriente possibile.

Il sistema fisico da controllare sarà un veicolo stradale con due assi e quattro ruote dotate di pneumatici; in particolare, si faranno le varie ipotesi per ottenere risultati il più possibile confrontabili con quelli reali ottenuti da una normale vettura di serie. Si vedrà come, certe ipotesi semplificative che trascurano le non linearità intrinseche, si adattino meglio a un certo tipo di vetture piuttosto che ad altre, ed è per questo che non si potrà fare a meno di considerare nell'analisi le numerose non linearità di modello.

La dinamica del veicolo stradale è una materia in cui teoria ed esperienza sono ugualmente importanti, ma raramente senza un confronto tra modelli matematici ed esperienza se ne riesce ad acquisire una conoscenza organica. La tentazione iniziale può essere quella di creare un modello molto accurato, nella speranza di poter descrivere tutti i fenomeni dinamici che intervengono durante il moto del veicolo stesso.

Purtroppo modelli troppo complicati diventano non controllabili sia dal punto di vista della formulazione matematica che della gestione dei risultati; inoltre comprendono molti parametri difficilmente rilevabili con precisione e quindi portano ad una accuratezza solo fittizia del modello utilizzato.

Infatti in generale si può dire che un veicolo a quattro, prendendo come sistema di riferimento quello mostrato in Figura 2.13 ruote ha 10 gradi di libertà che sono:

- Le coordinate del suo baricentro G nel sistema di riferimento assoluto X, Y, Z.
- I 3 angoli di imbardata (rotazione intorno all'asse verticale), rollio (rotazione intorno all'asse longitudinale) e beccheggio (rotazione intorno all'asse trasversale).
- 4 coordinate per il moto verticale ruota-cassa.

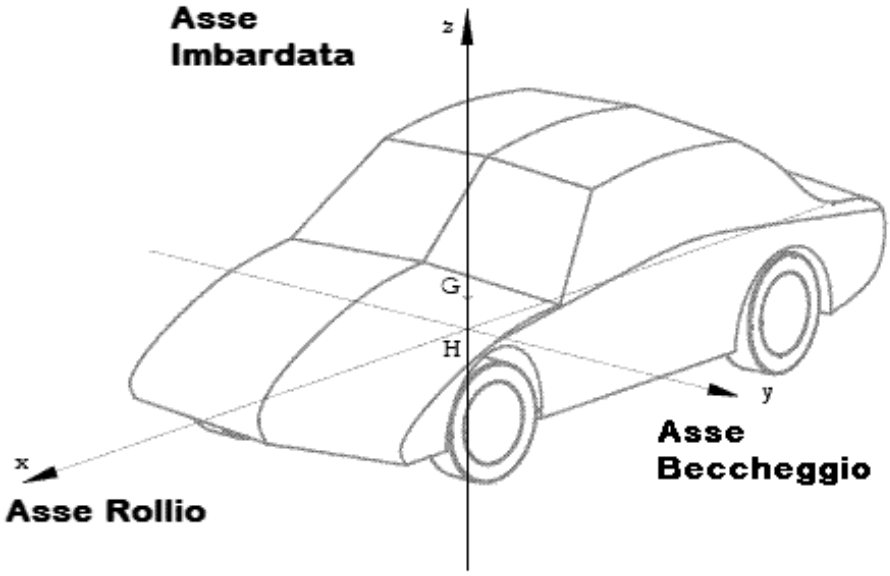

Figura 2.13: Sistema di riferimento per lo studio della dinamica del veicolo

Le tre rotazioni di imbardata, beccheggio e rollio possono essere considerate come le rotazioni da dare al sistema assi-suolo per ottenere il sistema assi-corpo, in particolare questi angoli sono considerati come rotazioni attorno agli assi, rispettivamente z, y, x. Se si considera un sitema fisso solidale al suolo, con gli assi X e Y paralleli al suolo e l'asse Z rivolto verso l'alto e uno solidale al veicolo, con origine nel punto H (intersezione tra l'asse di rollio e la verticale passante per il baricentro), l'asse x orizzontale è rivolto in avanti, l'asse z verticale è rivolto verso l'alto e l'asse y di conseguenza.

È bene allora prendere in esame modelli relativamente molto semplici e classici, con pochissimi gradi di libertà, che permettono comunque di descrivere gli aspetti salienti del moto del veicolo, anche se solo in particolari condizioni di impiego.

2.4 Ipotesi semplificative

Dovendo realizzare un modello molto semplice ma comunque valido nell'analisi delle problematiche di guida, riveste notevole importanza la scelta delle ipotesi semplificative da effettuare sul caso reale e la loro influenza sull'analisi matematica eseguita.

Nella formulazione del modello si considera quindi che:

- Il veicolo sia un corpo perfettamente rigido.
- Il veicolo si muova su strada orizzontale.
- Non esista nessun sistema di sospensioni.
- La velocita di avanzamento abbia variazioni piccole e lente, cosa che permette di trascurare l'influenza sullo spostamento di eventuali moti di beccheggio e di scuotimento e gli scorrimenti laterali dei pneumatici.
- Nel caso si considerino delle curve, queste sono ad ampio raggio e percorse a bassa velocità, così da poter ritenere trascurabile il moto di rollio e, di conseguenza, le variazioni degli angoli di Camber delle ruote. Si noti inoltre che avendo reso trascurabili le rotazioni del veicolo attorno agli assi di riferimento, il moto dello stesso può essere ritenuto piano.
- Le ruote anteriori abbiano massa molto inferiore rispetto al resto del veicolo, quindi la loro posizione non modificherà la posizione del baricentro del sistema.
- L'asse di sterzo di ciascuna ruota anteriore sia assimilato ad un'asse verticale di strada.
- Gli angoli di sterzo, siano piccoli, cosa che ben si concilia con l'ipotesi precedente di curve ad ampio raggio.

L'effetto totale di tutte queste ipotesi è un modello a tre soli gradi di libertà, in cui il veicolo è schematizzato come un unico corpo rigido in moto piano. E' possibile verificare come l'eliminazione di alcune delle ipotesi precedenti porti ad una notevole complicazione del modello.

Un veicolo terrestre presenta tre problematiche principali:

- Dinamica longitudinale.
- Dinamica laterale.
- Dinamica verticale.

2.5 La dinamica longitudinale

La *dinamica longitudinale* si occupa delle leggi in base alla quale il veicolo si muove secondo una traiettoria rettilinea, realizzando moti uniformi, accelerati o decelerati.

Gli aspetti fondamentali legati alla dinamica longitudinale riguardano:

- Dimensionamento del propulsore.
- Dimensionamento dell'impianto frenante o ripartizione delle forze frenanti sugli assi.
- Scelta dei rapporti di trasmissione del cambio.

Le forze che vengono scambiate nel moto del veicolo sono di varia natura:

- Forze alle ruote.
- Forse aerodinamiche.
- Forze motrici generate dal propulsore.
- Forze frenanti dovute all'azione dei freni.

Questi due ultimi tipi di forze sono propriamente forze interne al sistema, ma, poichè generano lavoro, devono essere considerate quando si valuta la dinamica del veicolo. Ci sono diversi modelli che possono essere utilizzati per l'individuazione delle forze che vengono scambiate nel contatto fra ruota e strada. Uno di questi modelli è quello di Coulomb. In questo modello si ipotizza che ruota e strada siano modellati come corpi rigidi in modo da considerare il loro contatto puntiforme. Se non c'è strisciamento fra i due corpi vale quanto segue:

$$\frac{|\vec{T}|}{|\vec{N}|} \leq f_s \tag{2.11}$$

dove $|\vec{T}|$ e $|\vec{N}|$ sono le componenti, rispettivamente, tangenziale e normale della forza di contatto.

Il parametro adimensionale f_s prende il nome di *coefficiente d'attrito statico*. Esso dipende dal tipo di materiale a contatto, dalla natura delle superfici e dalla velocità, quindi risulta indipendente sia dall'aria di contatto sia dal carico applicato.

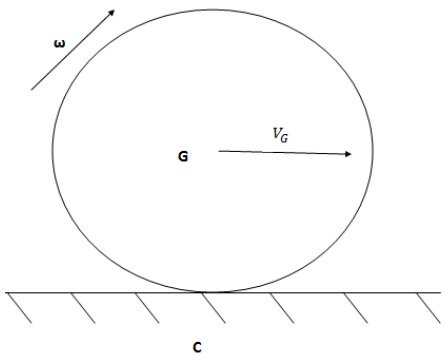

Figura 2.14: Schema di un corpo rigido per una ruota che non stricia

Con il modello di Coulomb applicato al caso ruota-strada, nell'ipotesi che non vi sia strisciamento fra i due corpi, e che quindi non sia superato il limite di aderenza definito dalla precedente formula, il punto di contatto coincide con il centro d'istantanea rotazione C, come si può osservare in Figura 2.14:

La velocità di traslazione del centro della ruota è espressa dalla 2.12:

$$\vec{V_G} = \vec{\omega} \wedge (G - C) \tag{2.12}$$

prodotto vettoriale tra $\vec{\omega}$, che rappresenta la velocità angolare, e $(G - C)$. Quando la forza tangente supera il valore definito dalla formula 2.12 si ha strisciamento tra ruota-suolo, conseguentemente la forza tangenziale ha direzione opposta alla velocità di strisciamento e in modulo vale:

$$|\vec{T}| = f_{cin}|\vec{N}| \tag{2.13}$$

in cui f_{cin} è chiamato coefficiente d'attrito radente.

In generale vale che:

$$f_s > f_{cin} \tag{2.14}$$

In prima approssimazione si è detto che, essendo i corpi rigidi, il contatto fra pneumatico e strada è di tipo puntiforme, ma in realtà si deve tener conto della deformabilità del pneumatico considerando che il contatto avviene su una superficie limitata.

Se si considera il materiale costituente la ruota perfettamente elastico, nel girare si crea un andamento di pressione normale simmetrico, quindi la normale che ne risulta passa per il centro del cerchio, come mostrato in Figura 2.15 per il caso ideale, e in Figura 2.16 per il caso reale:

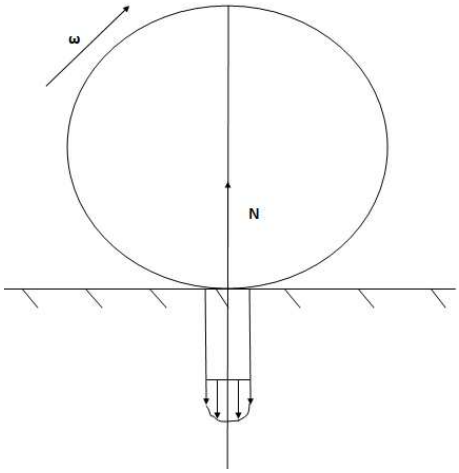

Figura 2.15: Andamento delle pressioni normali nel caso ideale

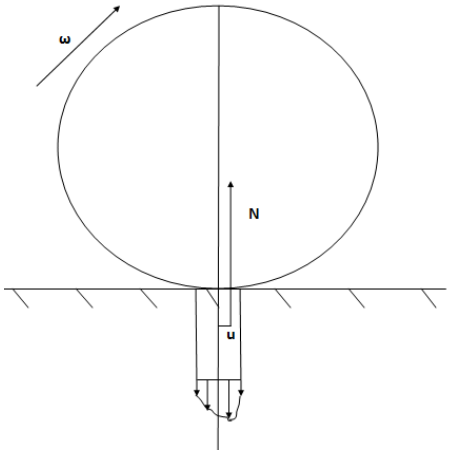

Figura 2.16: Andamento delle pressioni normali nel caso reale

Per tenere conto della non perfetta elasticità dei corpi a contatto si introduce il parametro d'attrito volvente u, in quanto occorre applicare una certa quantità di energia al pneumatico per deformarlo durante il moto. Quest'energia naturalmente in parte va persa, e come mostra disegno della Figura 2.16 c'è un certo scostamento dal caso ideale del tutto prevedibile.

Quindi per poter mantenere in moto la ruota è necessario avere un momento M_x pari a:
$M_x = \mathrm{N}\ u$. Valutando il lavoro dissipato per unità di percorso si può scrivere:

$$\frac{dL_s}{ds} = M_x \frac{d\varphi}{ds} = \frac{M_x}{ds}\frac{ds}{r} = \frac{M_x}{r} = \frac{Nu}{r} \tag{2.15}$$

dove $\frac{d\varphi}{ds}$ è l'energia dissipata per unità di percorso, r è il raggio del corpo rotolante. Viene definito coefficiente d'attrito volvente f_v come:

$$f_v = \frac{\frac{dL_s}{ds}}{N} = \frac{u}{r} \tag{2.16}$$

Il coefficiente d'attrito volvente dipende dalla pressione di gonfiaggio, dal tipo di pneumatico, dal tipo di strada e dalla velocità.

L'andamento di questo parametro può essere schematizzato, al variare della velocità e considerando un pneumatico a tele incrociate, come in Figura 2.17: Come si

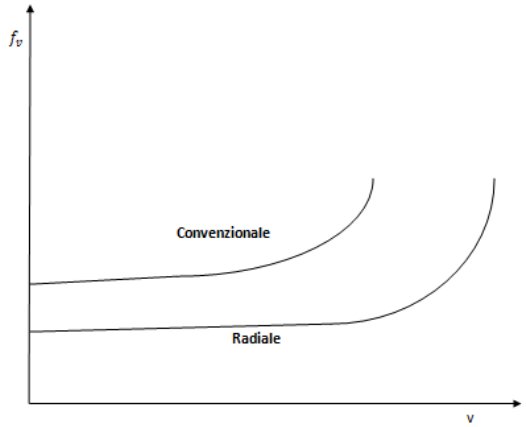

Figura 2.17: Andamento del coefficiente dell'attrito volvente

può notare, per entrambi i pneumatici f_v ha un andamento che cresce lentamente fino a una certa velocità, *velocità critica*, in corrispondenza della quale si individua un aumento evidente della pendenza della curva. Oltre la velocità critica l'attrito di rotolamento subisce un forte incremento che porta a fargli assumere valori elevati.

La *velocità critica* si raggiunge quando nascono moti vibratori che coinvolgono la struttura del pneumatico. In corrispondenza di questa velocità, la lunghezza d'onda delle vibrazioni cui è sottoposta la gomma diventa paragonabile alle dimensioni della superficie di contatto tra ruota e suolo, tanto che il pneumatico tende a staccarsi dal suolo nella parte posteriore del contatto e di conseguenza, la pressione fra ruota e strada deve aumentare notevolmente a seguito della diminuzione della superficie di contatto effettiva.

Per fenomeni di isteresi inoltre la temperatura della gomma aumenta e di conseguenza aumenta anche l'energia da dissipare nel rotolamento.

L'andamento della curva dell'attrito volvente, in corrispondenza della *velocità critica*, può essere rappresentata come mostra la formula 2.17:

$$f_v = f_{v_0} + k\, V^2 \tag{2.17}$$

dove f_{v_0} e K sono dei coefficienti da determinare in modo sperimentale per ogni tipo di pneumatico considerando le diverse condizioni dell'asfalto. Da notare che il valore di f_{v_0} per la neve è alto perché essa tende ad accumularsi davanti al pneumatico, e questo è anche il motivo per cui l'ABS in queste condizioni non è efficace. Infatti la neve tende a bloccare la ruota, ma a questo blocco si oppone il sistema elettronico di controllo che provoca un allungamento della frenata.

In generale si può dire che per ogni tipo di superficie ci sono dei valori indicativi sperimentali di f_{v_0}, infatti:

- Per una superficie formata da asfalto buono f_{v_0} varia tra 0,013 - 0,015.
- Per l'asfalto ricoperto da neve si ha f_{v_0} che varia tra 0,15 e 0,3.
- Asfalto ricoperto da neve f_{v_0} vale 0,025.
- Per asfalto ricoperto di sabbia f_{v_0} varia compreso tra 0,01 e 0,015.
- Se sull'asfalto c'è selciato f_{v_0} è compreso tra 0,033 e 0,065.

Il modello di Coulomb descrive approssimativamente bene il moto rettilineo dei veicoli, finchè le forze longitudinali non superano il limite imposto dalla relazione 2.18:

$$\frac{|\vec{T}|}{|\vec{N}|} \leq f_s \tag{2.18}$$

Questo modello può essere usato in condizioni di rotolamento e bloccaggio delle ruote. Si dimostra inadeguato quando s'intende studiare la deriva del pneumatico in curva o si voglia analizzare il comportamento in caso di sovra o sottosterzo.

2.6 Dinamica laterale

La *dinamica laterale* studia le leggi in base alla quale un veicolo si muove lungo una traiettoria curva (in genere fissando una legge di avanzamento). La traiettoria curvilinea può essere impostata dal sistema di guida (sterzo) oppure da una perturbazione esterna. A seconda di quale sia il modo in cui viene impostata la traiettoria curvilinea, si individuano differenti oggetti di studio:

- Il comportamento sovra o sottosterzo del veicolo, qualora la curva sia imposta per mezzo del sistema di guida, come generalmente avviene nello studio di un autoveicolo.
- Stabilità di marcia ad alte velocità e l'assetto in curva del veicolo, nel caso in cui la dinamica laterale sia dovuta a una perturbazione esterna. Questo campo di indagine si presenta generalmente nello studio della dinamica laterale di un veicolo ferroviario.

L'analisi della cinematica, tornerà utile per presentare i modelli che superano i limiti di quello presentato in precedenza.

E' necessario definire un sistema di riferimento adatto, supponendo che all'asse della ruota sia imposto un moto traslatorio con velocità costante $\vec{V}$ su una superficie perfettamente piana e rettilinea (condizioni queste che nella realtà non potranno mai verificarsi).

Il modello semplificato che tiene conto della dinamica laterale è il modello a bicicletta. Questo prevede che sia la ruota di destra che quella di sinistra abbiano lo stesso comportamento, quindi il veicolo si comporta come un biciclo.

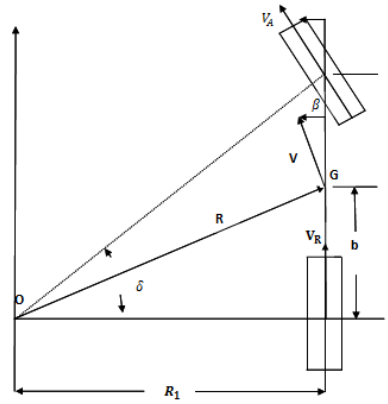

Figura 2.18: Schema di modello a bicicletta

2.7 Dinamica verticale

La *dinamica verticale* studia i moti vibratori con cui il veicolo reagisce in presenza d'irregolarità stradali. Tale studio è legato sia a problemi di comfort, sia a quelli legati al controllo della dinamica di marcia. L'incremento delle prestazioni di un

autoveicolo oggi è strettamente legato all'utilizzo di opportuni sistemi di controllo, sia su strada (ABS, controllo della trazione, ESP, sospensioni attive, ecc..) sia di ferrovia (antislittante, sospensioni attive, ecc..).

Il problema della dinamica verticale consiste nel determinare la risposta del veicolo (in termini di moto vibratorio e di forze scambiate con la strada) indotto dalla geometria del fondo stradale. Di particolare interesse risulta essere la presenza lungo il percorso di irregolarità dovute ad imperfezioni del fondo stradale che si generano sia durante la posa del fondo stesso sia per effetto di cedimenti anelastici del terreno, fenomeno quest'ultimo che porta ad una crescita dell'irregolarità con l'esercizio della via stradale. La presenza di queste irregolarità ha un effetto negativo nel comfort o nella sicurezza di marcia del veicolo. Nello studio della dinamica verticale, perciò, si considerano le sospensioni, che hanno il compito di:

- Dividere le forze scambiate tra terreno e cassa del veicolo (elementi elastici).
- Smorzare queste forze (elementi smorzanti).

Il rapporto tra *massa sospesa* e *massa non sospesa*, che sono rispettivamente, tutto ciò che sta sopra le sospensioni e tutto ciò che sta sotto le sospensioni (in pratica le ruote), è un indicatore di comfort. La sospensione ideale è quella che è capace di permettere solo i moti verticali fra ruota e massa sospesa. Nella pratica, i moti permessi sono determinati dalla geometria e cioè dalla cinematica della sospensione. Nello studio cinematico delle sospensioni si vuole capire come la geometria di ciascuna sospensione determina la cinematica del moto ruota-cassa. Per far questo si studiano le variazioni di campanatura, angolo di sterzo, carreggiata a seguito di variazioni dello scuotimento della sospensione e di rollio della cassa. Analisi che va oltre questo lavoro e che si rimanda, per essere approfondito, alla bibliografia.

Capitolo 3

Brake-by-Wire

3.1 Premessa

Questo capitolo ha come obiettivo quello di focalizzare che tipi di sistemi si terrano in considerazione per la realizzazione del modello che verrà creato. In particolare verranno menzionati i sistemi *Brake-by-Wire* e si farà una piccola introduzione ai sistemi *real-time*. Gli argomenti non verranno trattati dettagliatamente ma possono essere approfonditi analizzando i seguenti documenti della bibliografia: [16], [18],[17].

3.2 X-by-Wire

Quando si parla di Brake-by-Wire ci si riferisce a un particolare sistema X-by-Wire. I sistemi X-by-Wire erano stati introdotti negli aerei una decina di anni fà, prima a livello militare poi commerciale. Questi sistemi sostituiscono i sistemi meccanici e idraulici con sistemi controllati elettronicamente.

3.2.1 Tipologie di rete

Questi tipi di sistemi sono caratterizzati da due diverse architetture di rete:

- *ECU Centric Architecture.*
- *Network Centric Architecture.*

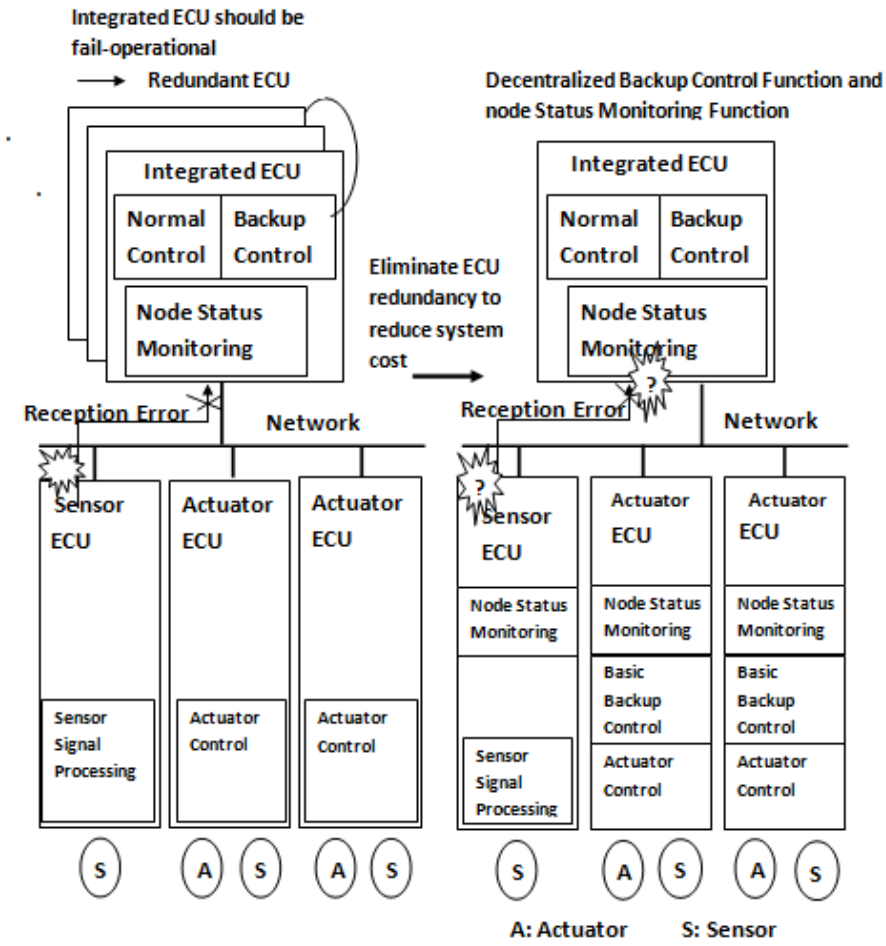

Figura 3.1: Confronto tra un'architettura di rete ECU e Network

La Figura 3.1 mostra le differenze tra un'architettura *ECU* e una *Network*. Nella *ECU Centric Architecture* tutte queste funzioni sono centralizzate e questo significa che quest'architettura è essenzialmente equivalente a una topologia di rete *server-client*. Un ECU è composto da più microcontrollori e da un set di sensori e attuatori. Un attuatore ECU (che funge da client) esegue solamente controllo. In questa architettura un *fault* nell'ECU integrato porterà al fallimento del sistema in quanto centralizzato. Per evitare questa situazione l'ECU integrato dovrebbe essere *fail-operational*, cioè dovrebbe essere in grado di continuare ad operare anche se uno o più *fault* vengono rilevati nell'ECU. Sono utilizzate in genere delle strategie di ridondanza per rimediare ad eventuali guasti anche se questa soluzione tende a far aumentare il costo dell'ECU e di conseguenza il costo del sistema. Infatti ciò che vuole evidenziare la Figura 3.1 è proprio il fatto che, essendo una topologia di rete basata sulla ridondanza, il *fault* su un ECU integrato se ridondato non causa il

blocco del sistema. In pratica tutto potrebbe continuare regolarmente in quanto l'ECU ridondante ha le stesse funzionalità di quello in cui è avvenuto il guasto.

La *Network Centric Architecture* ha la caratteristica di eliminare la ridondanza ECU riducendo così il costo totale del sistema, cioè si ha una struttura di rete conveniente dal punto di vista realizzativo, ma abbastanza lacunosa nel caso in cui si dovesse avere un comportamento anomalo nel sistema. In questa struttura, sebbene la normale funzione di controllo del veicolo è centralizzata nell'ECU integrato, la funzione *backup control* e la *funzione di monitoraggio* del nodo sono decentralizzate in ogni ECU. L'ECU condivide i dati attraverso la rete e ognuno di questi prende autonomamente i dati. In questo modo ogni ECU, includendo sensori e attuatori ECU, ne può monitorare lo stato di ogni altro ed eseguire il *backup control* in base all'ECU che ha subito il guasto. Questa tipologia di architettura può tollerare il *fault* di un ECU però i sensori e gli attuatori devono essere più "intelligenti" rispetto alla *ECU Centric Architecture*.

La scelta di una di queste strutture di rete è notevolmente dipendente dal rapporto tra affidabilità e costo-efficacia in quanto è necessario trovare un giusto bilanciamento tra queste due importanti caratteristiche per poter adeguatamente scegliere la topologia adatta. Si può in generale affermare che per ottenere un equo bilanciamento tra sicurezza e costi si è propensi ad utilizzare una tipologia di rete centrale basata sul concetto di sistema autonomo decentralizzato. Per sistemi che necessitano di alta affidabilità non è sufficiente il controllo standard delle funzioni del veicolo, bensì è necessario poter eseguire nel caso di fallimento di un ECU il *backup control* e poter monitorare i nodi (*membership function*) per identificare l'ECU in cui si è verificato il *fault*.

La Figura 3.2 mostra proprio la *Network Centric Architecture* applicata a un sistema *Brake-by-Wire*. Questo sistema consiste di un ECU integrato, quattro brake ECU, e un brake pedal ECU (come mostrato in Figura 3.2). Ogni ECU è programmato per essere *fail-silent*, in modo che se capita un guasto l'ECU non deve bloccare la comunicazione tra gli altri ECU ma deve essere in grado di bloccare la potenza applicata all'attuatore della frenata. Questi ECU condividono l'informazione richiesta per il controllo della frenata attraverso la rete.

Quando il sistema opera normalmente, come osservabile in Figura 3.2 l'ECU integrato riceve i dati S_1 mette i dati del *brake Pedal ECU* nella rete e calcola il valore di forza della frenata, B_1 e B_4, per i 4 brake ECU. Ogni ECU controlla ogni attuatore di frenata così che il valore di forza della frenata diventi il valore target. Come mostrato nella Figura 3.3, se nell'ECU integrato avviene un guasto la funzione di controllo di frenata dovrebbe essere sospesa. Tuttavia il veicolo può mantenere le

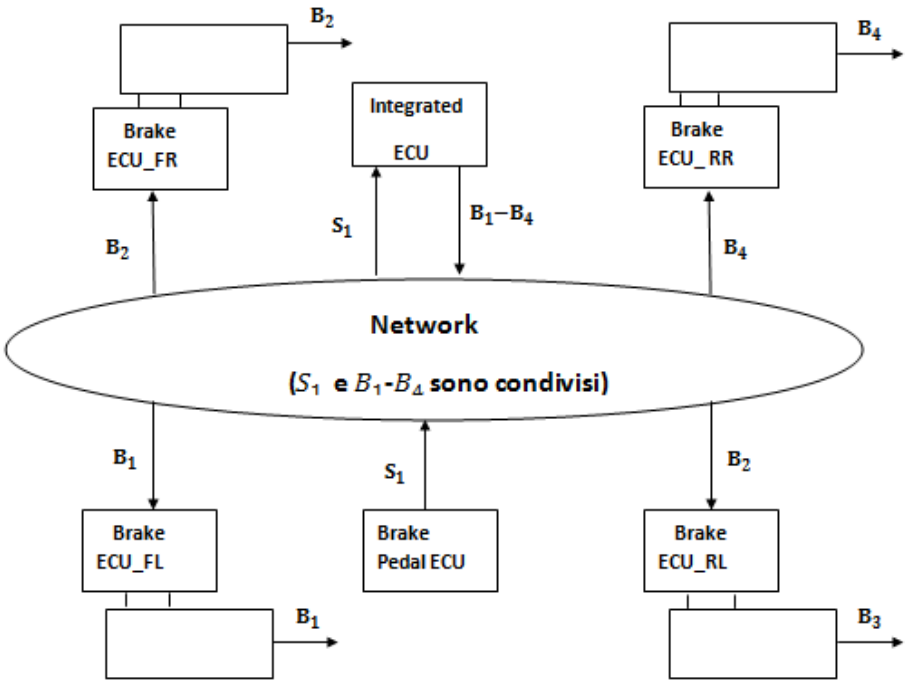

Figura 3.2: NetworkCentricArchitecture applicata a un sistema Brake-by-Wire

minime funzionalità di frenata grazie alla funzione di *backup autonomous* in cui ogni brake ECU determina l'ECU che ha fallito, riceve i dati S1 del brake pedal position condivisi, e indipendentemente calcola la forza della frenata con la logica di backup control integrata in ogni brake ECU.

D'altra parte poiché non c'è il nodo master per il monitoraggio della comunicazione e per eseguire l'*autonomuos backup control* è necessario utilizzare una tecnica che permette di identificare i nodi che sono falliti e assicurare un'adeguata quantità di informazione per individuare quali nodi sono disponibili tra tutti quelli restanti, in modo che si possano evitare ulteriori problemi nel controllo e nella valutazione.

Per esempio, come mostrato nella Figura 3.3, quando l'ECU integrato fallisce, riceve dati da certi sensori/attuatori ECU. Gli ECU non possono determinare se chi invia (sensori/attuatori) o chi riceve (ECU integrato) ha avuto guasti.

3.2.2 Funzionalità di membership

La funzionalità del membership nella Network Centric Architecture fornisce informazioni sulla disponibilità di tutti gli altri nodi attraverso lo scambio di locali in-

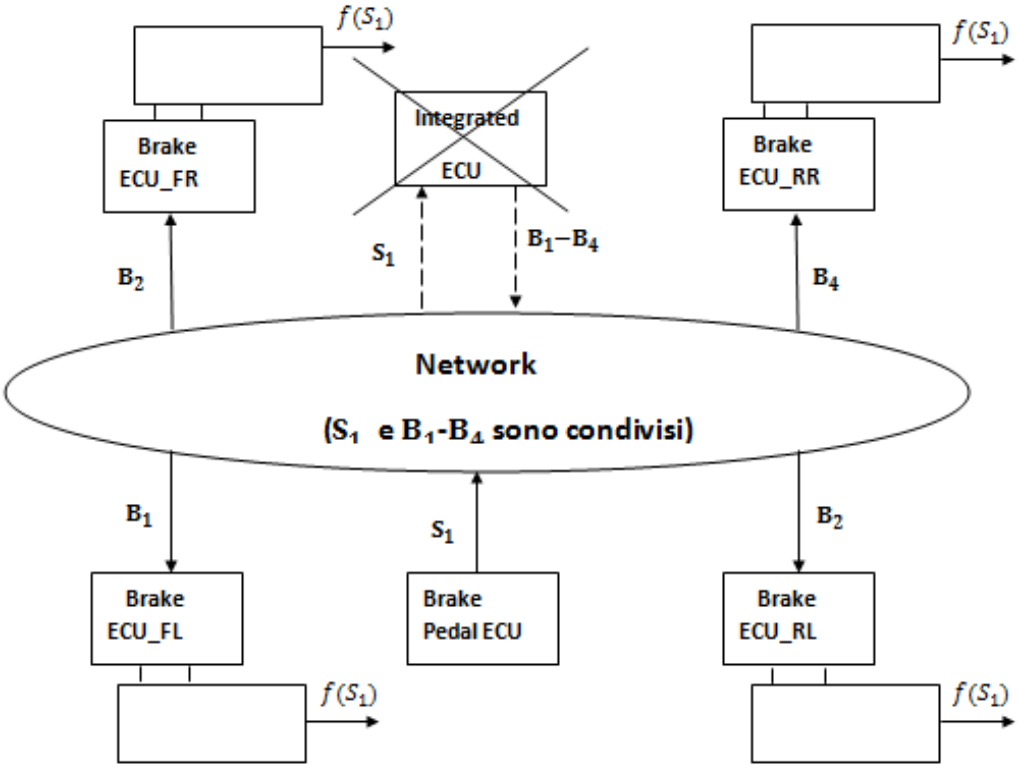

Figura 3.3: Caso in cui l'ECU integrato fallisce

formazioni sullo stato di ogni nodo. Un esempio di sequenza di processo in cui si utilizza la funzionalità membership di un sistema FlexRay è mostrata di seguito.

La comunicazione FlexRay è schedulata così che ogni nodo invia il frame almeno una volta in un *communication cycle* e riceve frame da tutti gli altri nodi. Nel *Communication Cycle i*, ogni nodo analizza lo stato di ogni altro nodo localmente valutando i frame inviati dagli altri. In questo *communication cycle*, anche se qualche nodo rileva un *fault*, in seguito ai dati inviati dagli altri, il nodo non può identificare se il guasto si trova nel nodo che ha inviato o in se stesso perché ognuno di essi valuta lo stato di un altro nodo da solo, quindi rileva un *fault* che in realtà potrebbe essere interno a se stesso.

La Figura 3.4 illustra un esempio concreto della *funzionalità membership* nel sistema FlexRay. Per semplicità qui viene mostrato solo lo stato dei dati nel nodo durante la trasmissione del frame. E' assunto che il sistema è formato da cinque nodi e il *fault* si è verificato nel nodo due. Poiché il *fault* si trova nel nodo quattro nello slot di trasmissione del ciclo i e il nodo due non può normalmente ricevere frame dal

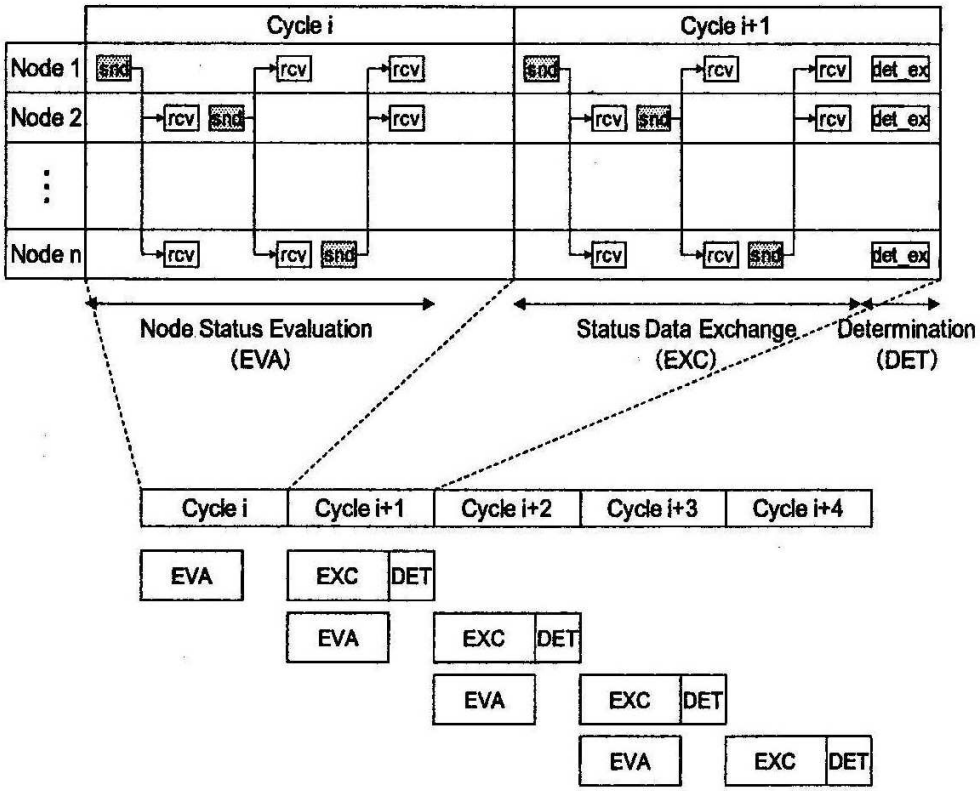

Figura 3.4: Sequenza del processo del Membership Funzionality nel sistema FlexRay

nodo quattro, nel *cycle* seguente il nodo due trasmette lo stato del nodo 00010, ciò significa che il nodo quattro ha fallito, ma gli altri nodi comunicano ai restanti che è tutto apposto. Alla fine del cycle *i+1*, lo *status data buffer* in ogni nodo contiene i dati mostrati in Figura 3.4, e il *fault* nel nodo due è identificato dal voting di questi dati. La stessa situazione continua nel *cycle i+1* e *i+2*. L'applicazione è informata del *fault* nel nodo due quando il valore del *fault counter* supera il valore di soglia, utile per evitare una sensibile reazione conseguente a un *fault transitorio*. Nella struttura proposta tutti i nodi adibiti al controllo, includendo sensori e attuatori intelligenti, condividono diversi dati, ricevendo in broadcast o in rete i dati di cui necessitano. Se qualche nodo blocca le operazioni a causa di un guasto, restano i nodi funzionanti correttamente che autonomamente eseguono il *backup* di controllo per mantenere le minime funzionalità necessarie al sistema, sfruttando i dati condivisi. Questa tipologia di rete può tollerare l'esistenza di un nodo mal funzionante non richiedendo elevati costi ma sfruttando sistemi di ridondanza.

Per individuare il nodo in cui si è verificato il *fault* e assicurare sufficiente informazione a tutti gli altri nodi, si utilizza l'approccio *membership middleware* che

monitora lo stato del nodo, ne valuta lo stato attraverso i dati scambiati con tutti i restanti componenti del sistema.

3.3 Brake-by-Wire

Per Brake-by-Wire s'intende il sistema di frenata del veicolo che da idraulico, in parte o completamente, viene sostituito da un sistema di frenata elettro-meccanico. Il Brake-by-Wire sarà la nuova generazione di sistemi di frenata, grazie al quale il veicolo si fermerà attraverso un segnale elettrico. Con questa nuova tipologia di freno il pedale genera un segnale elettrico che agisce su ogni attuatore del freno elettromeccanico posto su ogni singola ruota. Ci sono due diverse strade per implementare il sistema Brake-by-Wire:

Freno Elettro-Idraulico, EHB. La funzione di Brake-by-Wire è realizzata attraverso una pompa idraulica e delle valvole controllate elettricamente. Non ci sono connessioni meccaniche tra il pedale del freno e il sistema idraulico di frenata ed è possibile fare un sistema idraulico ridondante che possa essere usato in caso di emergenza. Se il sistema rileva un *fault* il sistema eletto-idraulico dovrà fermarsi e il freno idraulico dovrà essere chiuso con l'aiuto di alcune valvole.

*Freno Elettro-Meccanico, EMB.*Il sistema è basato su attuatori elettromeccanici. La forza e il controllo del freno sono realizzati attraverso componenti elettrici. Non c'è la possibilità di fare la ridondanza meccanica, quindi il sistema deve essere in grado di comportarsi in modo regolare anche in presenza di guasti.

Per i sistemi Brake-by-Wire sono richieste alti requisiti in termini di sicurezza, infatti dopo un *fault* il sistema deve essere capace di continuare a funzionare fino ad arrivare a uno stato di sicurezza. Per creare un sistema Brake-by-Wire "sicuro" è necessario avere requisiti hard real-time performanti, ed è necessario sapere i dati dei sensori e delle ruote che quindi devono essere il più precisi ed affidabili. Questo nuovo modello deve essere naturalmente più sicuro dei modelli già esistenti oltre che economicamente accessibile.

Anche se i requisiti di sicurezza sono ancora in via di perfezionamento, ci sono altri grossi vantaggi nell'introduzione del Brake-by-Wire nei veicoli se lo si paragona al tradizionale sistema idraulico:

- Il controllo dell'*Anti Blocking System* (ABS), il *controllo del freno*, i *programmi di stabilità elettronica* possono essere realizzati usando solamente software e sensori, cosicché i componenti meccanici e idraulici risulteranno superflui.

- L'interfaccia elettrica, a differenza di quella idraulica, permette un più facile adattamento ai sistemi di assistenza.
- Riduzione dei problemi di imballaggio. Ci saranno poche parti all'interno del veicolo se non ci sono collegamenti meccanici tra il componente freno e il sistema nella sua totalità. Questo permette di avere un vicolo più leggero, non sarà necessario liquido per i freni e come conseguenza si avrà un risvolto ecologico positivo nonchè una più semplice gestione del veicolo.
- L'assenza di collegamenti meccanici tra i componenti del freno e il motore migliora la sicurezza passiva, cioè non ci sono pezzi meccanici che a seguito di un incidente possono ferire il conducente.
- Riduzione dei costi di assemblaggio durante la produzione.
- È sicuramente più facile mantenere e installare un sistema elettrico che uno idraulico, e risulta più semplice riparare il sistema frenante che guidare il veicolo.
- Le vibrazioni e il rumore durante la frenata potranno essere annullate.
- Risposta immediata e regolata del freno in modo da permettere una guida più affidabile.
- L'interfaccia tra il conducente e il freno può essere realizzata in modo più flessibile. Forse in futuro un'autista diversamente abile potrà utilizzare un joystick invece del pedale.
- Pur essendo per ora costoso realizzare un sistema Brake-by-Wire, con l'espansione, l'assistenza risulterà economica.

3.4 Sistemi real-time

Un sistema real-time è un sistema in cui non è solamente importante il risultato della computazione, bensì anche il tempo di risposta. L'interesse per lo studio dei sistemi *real-time* è motivato dalla crescente diffusione e rilevanza che essi hanno nella nostra società. I settori applicativi in cui sono richieste delle attività di calcolo in *real-time* sono sempre più numerosi e includono:

- Regolazione di impianti chimici e nucleari.

- Controllo di processi produttivi complessi.
- Sistemi di controllo del traffico.
- Sistemi di regolazione delle automobili.
- Sistemi di sorveglianza e di monitoraggio ambientale.
- Sistemi di difesa.
- Automazione industriale.
- Sistemi multimediali e realtà virtuali.
- Sistemi di telecomunicazione.

La caratteristica che distingue l'elaborazione *real-time* dagli altri tipi di elaborazioni è il tempo. In particolare nel sistema *real-time* ci sono due concetti fondamentali:

- La parola *time* indica che la validità dei risultati prodotti da un processo di elaborazione non dipende solamente dalla correttezza delle singole operazioni, ma anche dal tempo in cui arrivano i risultati.
- La parola *real* indica che la risposta del sistema agli eventi esterni deve avvenire durante l'evolversi degli eventi stessi, e quindi il tempo interno di sistema deve essere valutato secondo un tempo interno uguale a quello in cui il sistema opera, cioè il tempo di sistema e il tempo dell'ambiente devono scorrere con la stessa velocità.

La proprietà di un sistema di essere *real-time* non è una caratteristica intrinseca di un sistema di controllo, ma è una qualità strettamente legata all'ambiente in cui il sistema si trova ad operare. In base delle conseguenze provocate da una *deadline* "mancata", i processi *real-time* vengono solitamente distinti in *hard* e *soft*.

Un processo *real-time* è di tipo *hard* se la violazione della propria *deadline* comporta un effetto catastrofico sul sistema, mentre è di tipo *soft* se la violazione della *deadline* comporta un degrado delle prestazioni, senza compromettere il corretto funzionamento del sistema. Esempi di processi *hard real-time* sono:

- Acquisizione di dati sensoriali.
- Rilevamento di condizioni critiche.

- Controllo di dispositivi automatici.
- Pianificazione di azioni senso-motorie si sistemi che interagiscono strettamente con l'ambiente.

Le seguenti possono invece essere collegate ad attività di tipo *soft*:

- Interprete di comandi forniti dall'utente.
- Ingresso da tastiera.
- Visualizzazione di messaggi sul monitor.
- Rappresentazione dello stato del sistema.
- Attività grafiche.
- Salvataggio di dati su disco.

Si può pensare a un sistema *real-time* come ad un sistema che, dato un insieme di n task, ognuno con i propri vincoli temporali (*deadline* del task i-esimo), è in grado di minimizzare la funzione di costo definita come:

$$k_s = \sum_{i}^{n} k_i(t) \tag{3.1}$$

dove $\mathbf{k_i(t)}$ è la *funzione di costo* del task-esimo definita pari a:

- $\mathbf{0 \text{ per } t \leq d_i}$
- $\mathbf{\infty \text{ per } t > d_i}$

se il task *i-esimo* è di tipo *hard real-time*, vale:

- $\mathbf{0 \text{ per } t \leq d_i}$
- $\mathbf{f(t) \text{ per } t > d_i}$

dove si è indicato con $f(t)$ una funzione monotona che cresce al crescere del tempo. I task di tipo *hard real-time* dovranno quindi essere schedulati in modo da terminare ad un istante t' minore della *deadline* in modo che la funzione di costo valga 0 e non inf; mentre i task *soft real-time* dovranno anche loro far valere 0 la funzione di costo relativa ma, in questo caso, il superamento della *deadline* t' non fare si che il costo globale del sistema vada all'infinito (equivalente al disastro).

3.4.1 Limiti dei sistemi real-time

Ancora oggi, anche se si migliora di giorno in giorno, molti sistemi *real-time* disponibili sul mercato sono basati su kernel sviluppati a partire da versioni di sistemi di tipo *timesharing*. Di conseguenza, essi si fondano sugli stessi principi su cui sono progettati i sistemi *timesharing*, ereditando alcuni meccanismi di base per la gestione dei processi, che sono adatti a supportare applicazioni in *real-time*. Sistemi di questo tipo presentano le seguenti caratteristiche:

- Meccanismo per la programmazione concorrente (*multitasking*) con primitive di creazione, terminazione, ritardo, sospensione e riattivazione dei processi.
- Scheduling di tipo prioritario, molto flessibile poiché consente si realizzare diverse strategie di gestione dei processi al variare della regola di assegnazione delle priorità. Tuttavia, nelle applicazioni real-time, la trasformazione di un insieme di vincoli temporali in un insieme di priorità non è sempre di facile attuazione. Inoltre se il sistema non prevede un meccanismo per la gestione esplicita del tempo, risulta difficile verificare il rispetto dei vincoli temporali sui processi.
- Capacità di rispondere velocemente alle interruzioni esterne, in genere minimizzando le parti di codice eseguite con interruzioni disabilitate. Una rapida gestione delle interruzioni minimizza i tempi di risposta agli eventi esterni, ma può anche rallentare in modo imprevedibile l'esecuzione dei processi di controllo, i quali potrebbero essere ancora più importanti.
- Primitive per la sincronizzazione e la cooperazione tra processi. I meccanismi di sincronizzazione e di mutua esclusione, se non sono gestiti mediante opportuni protocolli di accesso, possono causare dei fenomeni indesiderati, come lo stallo e l'inversione di priorità, che introducono dei ritardi indeterminati nell'esecuzione dei processi.
- Piccola dimensione del nucleo e veloce commutazione di contesto. Queste caratteristiche in genere consentono di velocizzare l'esecuzione dei processi, riducendo il tempo di risposta medio di un insieme di attività. Comunque un ridotto tempo medio di risposta non fornisce nessuna garanzia sul tempo di risposta di un particolare task. D'altra parte le piccole dimensioni del nucleo non consentono di realizzare funzionalità sofisticate per il supporto di processi *real-time*.

- Supporto di un *real-time clock* per la generazione di un riferimento temporale interno. Nonostante ciò, i sistemi *real-time* commerciali non prevedono nessuna primitiva per la specifica di vincoli temporali espliciti sui processi. Per esempio non è possibile specificare o monitorare una *deadline* e non esiste alcun meccanismo per la gestione di processi periodici. La gestione di eventi anomali e delle eccezioni è affidata generalmente a segnali dedicati o a segnali di timeout.

Tipicamente questo sistemi *real-time* si basano sulle caratteristiche dei sistemi *timesharing multiutente*, cioè tutti i task sono considerati come dei processi casuali indipendenti, di cui il sistema non conosce praticamente nulla, nemmeno le risorse da essi utilizzati. Al sistema è nota la priorità del processo, nella quale devono essere codificate tutte le informazioni rilevanti. Di conseguenza, l'obiettivo principale di tali sistemi è quello di minimizzare il tempo medio di risposta agli eventi esterni attraverso un insieme di primitive molto veloci.

Anche se questo è un buon risultato, non sempre è sufficiente a garantire i vincoli temporali richiesti dalle attività di tipo *real-time*. Spesso si è costretti a trasformare l'insieme dei vincoli temporali dei task in un insieme di priorità in modo tale che tutti i processi siano schedulati entro le proprie *deadline*. Il prezzo da pagare nel seguire questa strategia è l'estrema inaffidabilità del sistema, dovuta alla difficoltà di prevedere le interazioni tra i processi invocati dinamicamente e gli altri processi attivi del sistema, oltre che quella di prevedere gli effetti che queste interazioni hanno sui vincoli temporali di tutti i task applicativi.

3.4.2 Caratteristiche desiderabili nei sistemi real-time

In un sistema di controllo *real-time*, il codice di ogni processo è noto a priori ed il suo comportamento può essere completamente descritto in funzione dello stato del sistema e dei suoi dati di ingresso. Inoltre, invece di minimizzare il tempo medio di risposta agli eventi esterni, è importante assicurare che tutti i task critici completino la loro attività entro la propria *deadline*.

In un'applicazione *real-time*, i vari processi di controllo cooperano per portare a termine un unico obiettivo generale. Di conseguenza i task non sono tutti indipendenti e non è necessario che il sistema imponga spazi di indirizzamento separati.

Un sistema *real-time* dovrebbe seguire i seguenti presupposti e possedere le seguenti caratteristiche:

- *Timeliness*: i risultati devono essere corretti non solo nei valori, ma anche nel dominio del tempo, quindi il sistema operativo deve fornire specifici meccanismi per la gestione del tempo e dell'attivazione di task (sia periodici che aperiodici) con vincoli temporali espliciti e differenti criticità.

- *Prevedibilità*: il sistema deve essere in grado di prevedere le conseguenze delle decisioni di scheduling e valutare la fattibilità della schedulazione in funzione delle proprietà temporali dei processi. Per fare questo, tutte le primitive di sistema devono avere un tempo di esecuzione massimo definito, cioè non ci devono essere ritardi indesiderati.

- *Tolleranza ai sovraccarichi*: in modo tale che il sistema non collassi in situazioni di sovraccarico.

- *Monitorabilità*: in quanto il sistema deve poter monitorare lo stato di esecuzione dei processi al fine di segnalare eventuali eccezioni dovute al superamento dei vincoli temporali ed intraprendere opportune azioni di recupero.

- *Flessibilità*: il sistema deve essere realizzato secondo una struttura modulare in modo che sia facilmente modificabile, al fine di adattare i meccanismi di nucleo alle esigenze dell'applicazione.

- *Schedulazione ottima*: tutti i task sono noti a priori così come i vincoli temporali. Dovrebbe essere possibile dunque avere uno schedulatore minimizzi la funzione di schedulazione stessa.

- *Condivisione delle risorse*: i task sono entità separate ma che concorrono ad uno stesso scopo, pertanto non è necessario avere spazi di indirizzamento separati.

- *Garanzia di esecuzione*: tutti i task di tipo *hard real-time* devono terminare entro le proprie *deadline* quindi, nel caso in cui arrivi un nuovo task, o un task non possa completare entro la *deadline*, una notifica anticipata del sistema può essere utilizzata per impedire l'esecuzione del nuovo task o per recuperare l'esecuzione il task che si avvia al superamento della propria *deadline*.

E' importante inoltre ricordare che in un sistema *real-time* la proprietà fondamentale non è la velocità, intesa come capacità del sistema si rispondere a degli impulsi esterni, ma ciò che risulta fondamentale in un sistema di questo tipo è la *prevedibilità*. Questa si può ottenere solo impiegando metodologie di analisi e strategie di gestione

dei processi che possano garantire il rispetto dei requisiti richiesti sotto tutte le condizioni di carico previste.

La differenza principale tra un sistema *real-time* e *non real-time* è che il primo deve terminare entro una scadenza temporale prefissata, la *deadline*, che rappresenta il tempo massimo entro cui un processo *real-time* deve terminare la propria esecuzione. In questi sistemi un risultato prodotto oltre la propria *deadline* non è solo in ritardo, ma è dannoso.

3.4.3 Fault management

Un sistema è considerato *critico* in termini di sicurezza se un qualsiasi comportamento anomalo da esso generato o subito può provocare dei danni rilevanti, a persone o cose. Per esempio una centrale nucleare è definita *safety critical system* perchè se qui venisse commesso un errore nel controllo del sistema potrebbero essere generati danni catastrofici.

I sistemi *safety critical system* sono definiti nel *safety critical computer system* come sistemi che non devono essere solo sicuri ma devono mostrare e ispirare sicurezza. Una delle proprietà che un *safety related system* deve possedere è la predittività. La sicurezza non è solo da considerare nello sviluppo del software, ma deve essere una parte di tutto il sistema di produzione e manutenzione. I *safety related systems* possono essere divisi in:

- *Control system* usati per il controllo delle operazioni di sistema e in alcuni casi come fornitori di funzioni di sicurezza.
- *Protection system* usati per rilevare gli errori e le carenze di altri sistemi.

Questi sistemi vengono anche chiamati *sistemi d'arresto* perchè sono in grado di bloccare il sistema nel caso in cui si verifichi un comportamento ritenuto pericoloso. Tipiche situazioni in cui si può trovare il sistema sono:

- *Fault* quando si ha un'imperfezione all'interno del sistema. Questi possono essere di due tipi, *random* e *sistematici*. Il *fault random* avviene generalmente una volta ed è connesso spesso ad un guasto hardware, mentre un *fault sistematico* è un errore nel sistema generato durante la sua creazione ripetendosi.
- *Failure* si ha quando il sistema non riesce a svolgere la sua funzione.

- L'*Error* è il risultato del verificarsi di un *fault* durante le operazioni del sistema e di conseguenza il sistema mostra un comportamento indesiderato.

Quando si creano dei *safety critical system* è molto importante riuscire a gestire nel migliore modo possibile le situazioni di guasto che possono crearsi. Ci sono alcune tecniche per fare questo, ognuna della quali è adatta a particolari situazioni:

- La *fault avoidance* è una tecnica che ha lo scopo di prevenire i *fault* nel sistema.
- La *fault removal* è la tecnica che cerca di rilevare tutti i *fault* introdotti durante la progettazione. Questo viene fatto prima che il sistema venga messo in funzione, attraverso una serie di test approfonditi sia a livello hardware che software.
- I metodi *fault detection* sono applicati quando il sistema è in uso per rilevare i *fault* nell'intervallo di tempo più piccolo per minimizzare gli errori sul sistema. Il sistema che usa il *fault detection algorithm* è più adatto per sistemi *safety critical* perchè loro cercano ininterrottamente i possibili *fault* nel sistema. Il metodo include il *functionality testing*, per il controllo del funzionamento dell'hardware, per verificare che sia performante e rispetti le specifiche richieste, e l'*information redundancy* con lo scopo di rilevare gli errori nei dati usando il *CRC*, il *checksums* e l'*error correcting codes*.
- Le tecniche di *fault tolerance* dipendono spesso dall'esistenza di tecniche di *fault detection*. La *fault tolerance* è una proprietà legata all'abilità del sistema di resistere ai *fault* senza cambiare il comportamento del sistema. Ci sono molte tecniche per la *fault tolerance* come la ridondanza hardware e software.

Capitolo 4

FlexRay Protocol

4.1 Premessa

In questo capitolo l'attenzione si focalizza sul protocollo di comunicazione FlexRay, che negli ultimi anni sta sostituendo i "vecchi" protocolli di comunicazione usati nel settore automotive poiché offre caratteristiche e potenzialità migliori. Verrano quindi descritte in maniera sufficientemente dettagliate le caratteristiche relative alla trasmissione, all'affidabilità di questo protocollo. Inoltre verrà fatta una panoramica sugli altri protocolli utilizzati, in passato ma anche oggi, nel settore dell'automobile, cercando di evidenziare pregi e difetti di ognuno di essi. Inoltre saranno fatto gli opportuni confronti con il FlexRay per capire con maggior chiarezza perché si sta tanto investendo su questa nuova tecnologia. E' possibile approfondire gli argomenti trattati in questo capitolo consultando le reference della bibliografia, in particolare la [25], [4], [2], [18], [27], [28],[29], [30].

4.2 Introduzione

Esistono diversi protocollo di comunicazione usati nel settore auto, tuttavia è necessario lo sviluppo di nuovi in quanto i bus esistenti non riescono a garantire la velocità di trasmissione dei dati, sempre crescente, e i livelli di sicurezza indispensabili per le applicazioni nelle future auto.

In questo capitolo verrà analizzato in modo dettagliato il **FlexRay** che è il protocollo più recente nonché quello meno utilizzato e sfruttato perchè ancora in via di sviluppo. Verranno inoltre fatti dei confronti fra i diversi protocolli sviluppati per

il settore automotive per avere chiari quali sono i motivi che spingono i ricercatori e tanto più i costruttori ad utilizzare il FlexRay.

Uno dei bus più utilizzati nei veicoli, il CAN, non è in grado di superare la frequenza massima di trasmissione di $1Mbit/s$ (mentre ormai è necessario arrivare almeno a $10Mbit/s$) e introdurre nuovi concetti di rete che tengano conto del numero crescente di componenti elettronici presenti nella vettura e del relativo software a loro associati.

In media al giorno d'oggi si possono contare in ogni veicolo almeno 40 componenti interconnessi tra loro. Il grande numero di controllori e la necessità di implementare funzioni critiche di sicurezza, rende indispensabile un bus ad alta velocità, sicuro e pianificabile. Questo è il motivo per cui i produttori hanno stanno investendo sul nuovo standard FlexRay. Questo è una soluzione per la comunicazione ad alta velocità, in grado di reagire adeguatamente in caso di guasto, adatto per le applicazioni più sofisticate nel settore dell'auto, come il controllo elettronico della frenata, dello sterzo e dell'accelerazione.

A differenza dei sistemi chiamati *event-triggered* (attivati da un evento definito), come il *CAN*, i sistemi elettronici *time-triggered* (ad attivazione temporale) utilizzano un meccanismo ciclico di comunicazione dei dati sul bus, a intervalli regolari e predefiniti. In questo modo non viene sovraccaricato il mezzo fisico di comunicazione (il cavo), come potrebbe invece accadere nel caso della presenza contemporanea di diversi eventi ad alta priorità.

FlexRay è particolarmente adatto per applicazioni critiche di sicurezza, in cui le attuali soluzioni meccaniche devono essere completamente sostituite da un controllo elettronico. Oltre ad essere affidabile, in grado di reagire correttamente e in modo adeguato in caso di guasto, FlexRay offre il vantaggio di una maggiore facilità di configurazione e collaudo. Questo permette ai produttori di auto e ai loro fornitori di ridurre il *time-to-market* e i costi di sviluppo, prova e manutenzione dei nuovi sistemi elettronici.

BMW e DaimlerChrysler fondarono nel 1998 il *FlexRay Consortium* di cui oggi ne fanno parte oltre ai fondatori Bosch, freescale, General Motors, Philips e Volkswagen.

4.3 Architettura

Il FlexRay è configurabile in diverse topologie di rete, quali:

- **Active Star** a doppio o singolo canale:

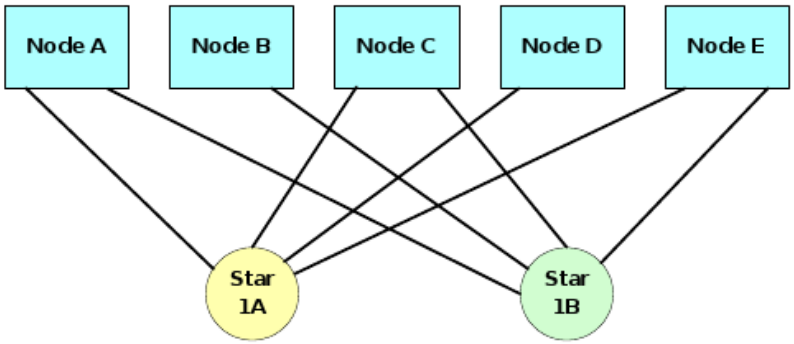

Figura 4.1: Configurazione a singola stella con doppio canale

Mentre la Figura 4.2 rappresenta una rete a doppia stella accoppiate a singolo canale:

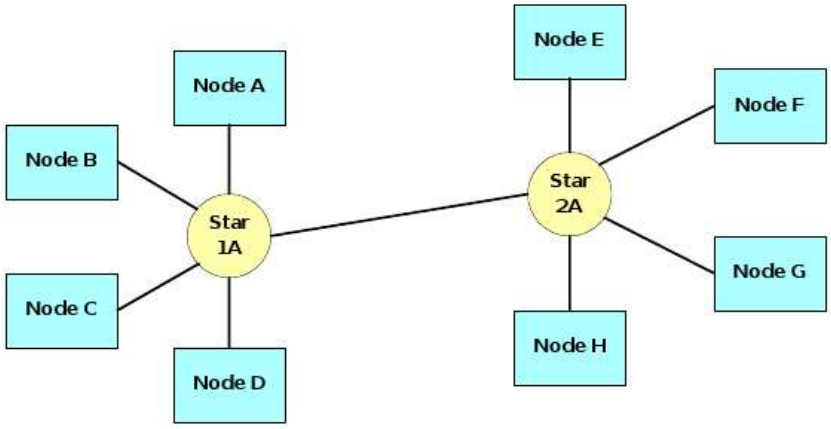

Figura 4.2: Configurazione a doppia stella in cascata con singolo canale

E' possibile anche avere la configurazione a doppia stella in cascata con doppio canale, (come mostrato in Figura 4.3).

- **Linear Passive Bus**, può essere a doppio canale e ogni nodo può essere collegato a uno o entrambi i canali (come mostra la Figura 4.4).

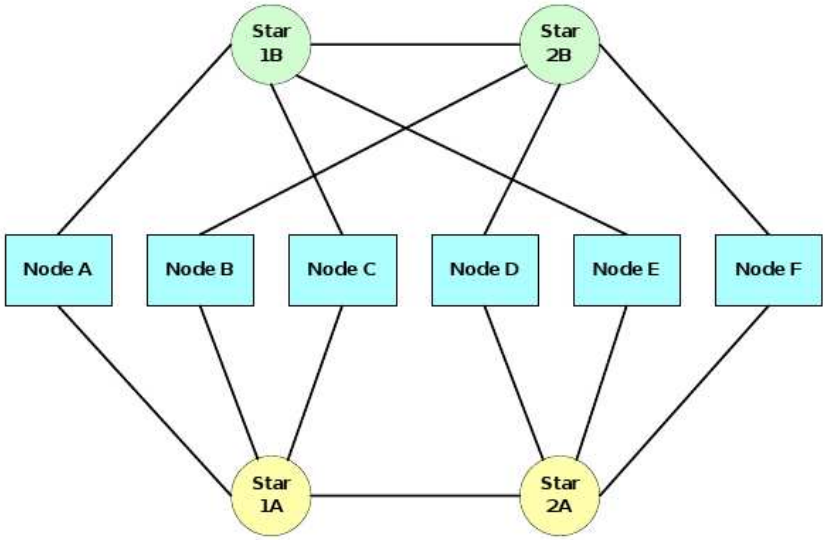

Figura 4.3: Configurazione a doppia stella in cascata con doppio canale

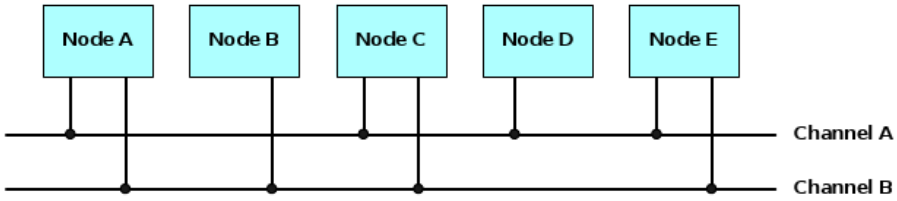

Figura 4.4: Configurazione lineare passiva a doppio canale

- **Ibrida**, formata dalla combinazione active star e passive bus topology. Anche

questa topologia di rete può essere a singolo canale, come mostra la Figura 4.5:

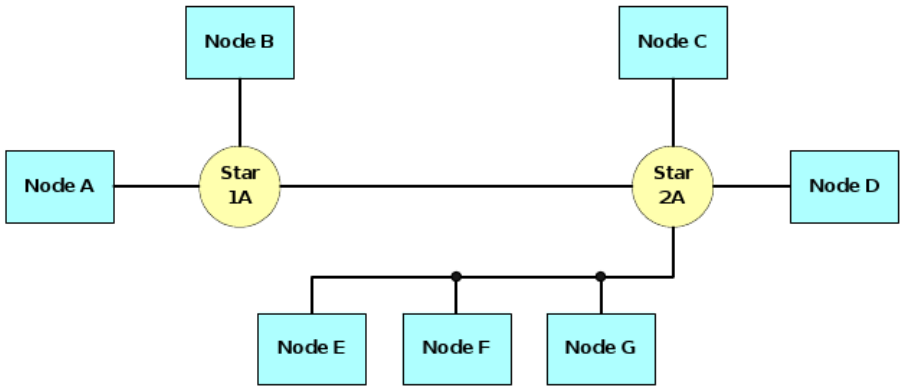

Figura 4.5: Topologia di rete ibrida a singolo canale

o a doppio canale come mostra la 4.6:

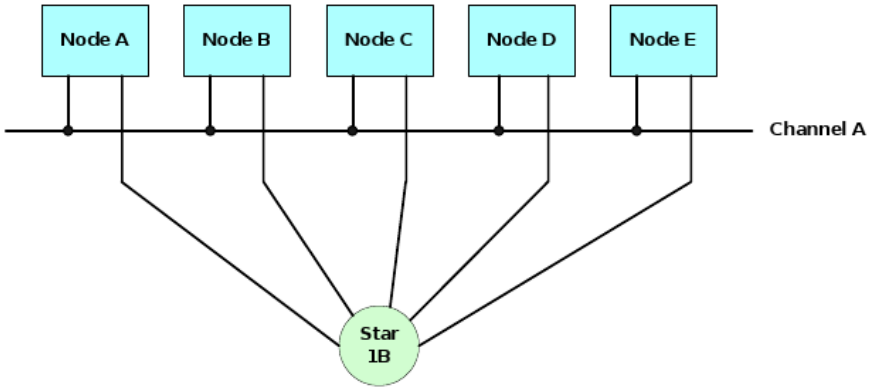

Figura 4.6: Topologia di rete ibrida a doppio canale

La rete consiste di nodi che sono indipendenti e connessi attraverso uno o due canali di comunicazione (bus). Un tipico nodo FlexRay è composto dai componenti mostrati in Figura 4.7:

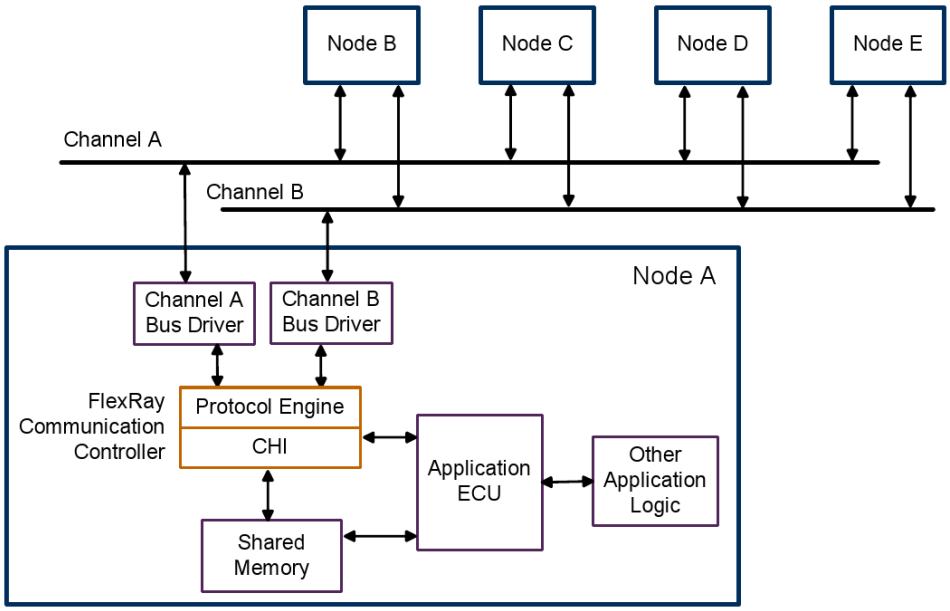

Figura 4.7: Esempio di FlexRay Cluster

- Un *Electronic Control Unit* (ECU) e altre specifiche applicazioni hardware.
- Un FlexRay *Communication Controller* in genere definisce l'implementazione della configurazione, del controllo, il *register status* per i *message buffers* nel CHI, usa le regioni di memoria condivisa per conservare gli effettivi *header*, *payload data*, *slot status data* associato al message buffer ed è responsabile della sincronizzazione dei *macrotick*. Inoltre essendo implementato con un blocco IP riutilizzabile fornisce all'hardware la possibilità di configurazione per scegliere il numero di *message buffer* da implementare. Questo permette di ottimizzare la dimensione della memoria, nonché la memoria condivisa richiesta per i *message buffer* e necessari all'applicazione.
- Il FlexRay *Bus Driven* che è implementato per ogni canale FlexRay.

- Una *Memoria* che è condivisa tra l'applicazione *ECU* e il FlexRay *communication controller*.

Come si può vedere il blocco del *communication controller* è formato da due blocchi principali:

- *FlexRay Protocol Engine* implementa la maggior parte delle funzioni del FlexRay compresa la trasmissione e la ricezione dei frame, la sincronizzazione del clock con gli altri nodi nel cluster e la generazione e il controllo del *CRC*.

- *Controller Host Interface* (CHI) esegue le seguenti operazioni:

- Configura, controlla e visiona il *Protocol Engine*.

- Permette lo scambio dei *message dati* e di *status* fra l'applicazione e il *Protocol Engine* attraverso un set di buffer messaggi implementati nel nodo; ogni *message buffer* è assegnato ad uno slot nel bus FlexRay *timing hierarchy*.

4.3.1 Bus Guardian

Il *bus guardian* provvede alla gestione di schemi e dati indipendentemente dal controller della comunicazione. Protegge la comunicazione nel canale da possibili guasti e da possibili comportamenti anomali del controller garantendo l'invio dei messaggi sugli slot pre-allocati. Per ogni controller esiste un corrispondente bus guardian. La trasmissione del frame ha inizio quando il bus guardian da l'accesso al bus, monitora i tempi e valuta se c'è la possibiltà di inviare un nuovo messaggio. Questo è dato solamente nel segmento dinamico o se il nodo è schedulato per essere inviato nello slot corrente. La trasmissione deve essere segnalata sul bus e alla fine il frame è inviato.

Questi step sono necessari affinchè non vengano inviati messaggi quando il bus è occupato o per evitare l'invio di messaggi in slot di altri nodi. Non è inoltre permesso di inviare un messaggio quando l'intervallo di tempo per la chiusura dello slot è piccolo, in quanto ci sarebbe il rischio che la trasmissione non venga conclusa.

Possono essere definite quattro principali proprietà per il bus guardian:

- *Correct Relay*. Se il controller invia un messaggio corretto, il suo *non-fault* bus guardian invia il messaggio.

- *Validity.* Se il *non-fault* bus guardian invia il messaggio, tutti i controller corretti ricevono il messaggio.
- *Agreement.* Se il *non-fault controller* riceve il messaggio allora tutti i *non-fault controller* ricevono il messaggio.
- *Integrity.* Se il messaggio è ricevuto dal *non-fault controller*, il messaggio deve essere stato inviato da un altro *non-fault controller*.

Verifica delle proprietà

In questa sezione vengono dimostrate le proprietà caratteristiche del bus guardian, in particolare il *Correct Relay* (in cui viene fatta un'analisi simile a quella che si potrebbe fare per la *Validity*), e l'*Integrity* (in cui viene fatta un'analisi simile a quella che si dovrebbe fare per dimostrare la proprietà) di *Agreement*. Si assume che il controller e il bus guardian siano in uno stato stabile, cioè si trascuri l'attivazione e l'inizio del processo.

È necessario tenere conto di due definizioni del tempo, *clocktime* e *realtime*. Formalmente per il nodo p', il *virtual time* (o *logical clock*) è definito dalla funzione:

$\mathbf{VC_p}$: realtime$\rightarrow$ *clocktime*

L'interpretazione logica di questa scrittura è che $\mathbf{VC_p}$ è la lettura del *logical clock* di p' al realtime t.

Inoltre si assume che il sistema abbia le seguenti caratteristiche:

- *Bounded Clock Drift Rate.* Per *non-fault clock* c'è una costante positiva ρ che indica il gap fra il *clock time* e il *realtime*.
- *Bounded Trasmission Delay.* Il ritardo di comunicazione tra due *non-fault* nodi è delimitato da una costante ϵ.

La velocità delimitata di clock è formalizzata dalla seguente scrittura:

$$\forall\, \mathbf{t_1},\ \mathbf{t_2}\ \in\ \mathbf{realtime},\ \mathbf{t_1}\ <\ \mathbf{t_2}\ :\ (\ \mathbf{1}-\rho)(\mathbf{t_2}-\mathbf{t_1})\ \leq\ \mathbf{VC}(\ \mathbf{t_2})\ \mathbf{VC}(\mathbf{t_1}) \leq (\mathbf{1}+\rho)(\mathbf{t_2}-\mathbf{t_1})$$

Si presuppone la correttezza della sincronizzazione del clock, cioè che i clock dei controller e dei bus guardian vengano mantenuti. Indicando con BG il bus guardian, CC il controller e con ρ la *precisione* del cluster, si ha :

$$\forall\, \mathbf{p},\ \mathbf{q}\ \in\ \mathbf{BG}\ \bigcup\ \mathbf{CC},\ \mathbf{t}\ \in\ \mathbf{realtime}: |\ \mathbf{VC_p}(\mathbf{t})\ -\ \mathbf{VC_q}(\mathbf{t})\ |\ \leq\ \rho.$$

Il *correct relay* e *validity* descrivono la relazione di tempo tra il controller e il proprio bus guardian in uno slot.

Il teorema del *correct relay* vuole mostrare che la finestra di trasmissione del bus guardian copre la finestra di invio del mittente.

La figura 4.8 mostra la funzione di schedulazione che definisce il *clocktime* all'inizio dello slot:

sched : **Slot** $\rightarrow$ **Clocktime**

dove il tipo *Slot* è definito da un numero naturale.

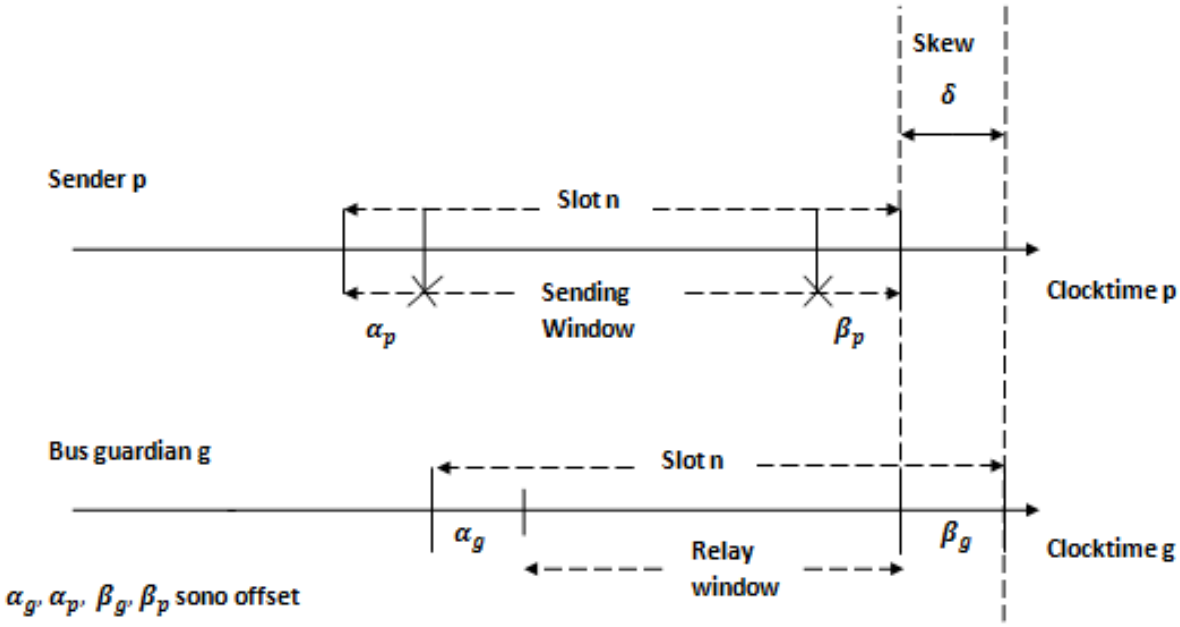

Figura 4.8: Informazioni sul tempo di invio del mittente e del bus guardian

In questo modo **sched**(**n**) è il *clocktime* all'inizio dello slot *n*.

Si supponga che, dopo l'offset $\alpha_{\mathbf{p}}$, il mittente p inizia la trasmissione del messaggio nello slot n.

Sia t il *realtime* tale che:

$\mathbf{VC_p(t)} = \mathbf{sched(n)} + \alpha_{\mathbf{p}}$.

Ovviamente, t è il *real-time* quando il mittente p apre la propria finestra di invio. È necessario che al *real-time* $(t + \epsilon)$ il bus guardian sia pronto per ricevere il messaggio dal mittente nello slot n.È dunque necessario verificare che:

$\mathbf{VC_g(t\ +\ \epsilon)\ \geq\ sched(n)\ \alpha_g}$

α_g è l'offset dello slot n per il bus guardian g. Quando ϵ è piccolo, risulta insignificante dimostrare che $\mathbf{VC_p(t\ +\ \epsilon)}$ è approssimativamente uguale a $\mathbf{VC_p(t)\ +\ \epsilon}$. Per semplicità si assume che:

$\mathbf{VC_g(t\ +\ \epsilon)\ =\ VC_p(t)\ +\ \epsilon}$.

Con l'assunzione sulla sincronizzazione del clock, l'obiettivo è di mostrare se la relazione $\alpha_\mathbf{p}\ \geq\ \alpha_\mathbf{g}\ +\ \delta\ -\ \epsilon$ può essere soddisfatta.

Le specifiche del bus guardian nel FlexRay richiedono che:

- $\alpha_\mathbf{p}\ \geq\ \mathbf{2}\delta$
- $\alpha_\mathbf{g}\ +\ \alpha$

Si supponga che $\triangle$(n) sia la durata di trasmissione del messaggio nello slot n. Sia t' il *real-time* tale che:

$\mathbf{VC_p(t')\ =\ sched(n)\ +\ \triangle(n)\ +\ \beta_p}$.

Naturalmente t' è l'ultimo realtime che chiude la finestra di invio. È necessario che il bus guardian mantenga aperta la finestra di invio fino al *realtime*,

$\mathbf{(t'\ +\epsilon):\ VC_g(t'\ +\ \epsilon)\ \leq\ sched(n)\ +\ \triangle(n)\ +\ \beta_p}$.

Secondo le ipotesi sulla sincronizzazione e l'assunzione $\beta_\mathbf{g}\ \geq\ (\beta_\mathbf{p}\ +\ \delta\ +\ \epsilon)$ la disuguaglianza è verificata.

Per definire l'*integrity* dobbiamo definire un set di C azioni, che sono di due tipi:

- Invio
- Ricezione

Ogni nodo è interpretato dal nodo. Il nodo è definito come un tipo di dati con due costruttori:

- $\mathbf{CC_p}$ per alcuni communication controller.
- $\mathbf{BG_q}$ per alcuni bus guardian q.

dove p e q sono numeri naturali.

Il set C di possibili azioni di comunicazione tra controller e bus guardian è definito da tre regole:

- $(\mathbf{invia}\ (\mathbf{CC_p})\ \mathbf{m}) \in \mathbf{C} \implies (\mathbf{riceve}\ (\mathbf{BG_p})\ \mathbf{m}) \in \mathbf{C}$
- $(\mathbf{riceve}\ (\mathbf{BG_p})\ \mathbf{m}) \in \mathbf{C} \implies (\mathbf{invia}\ (\mathbf{BG_p})\ \mathbf{m}) \in \mathbf{C}$
- $\mathbf{p} \neq \mathbf{q} \wedge (\mathbf{invia}\ (\mathbf{BG_p})\ \mathbf{m}) \in \mathbf{C} \implies (\mathbf{riceve}\ (\mathbf{CC_q})\ \mathbf{m}) \in \mathbf{C}$

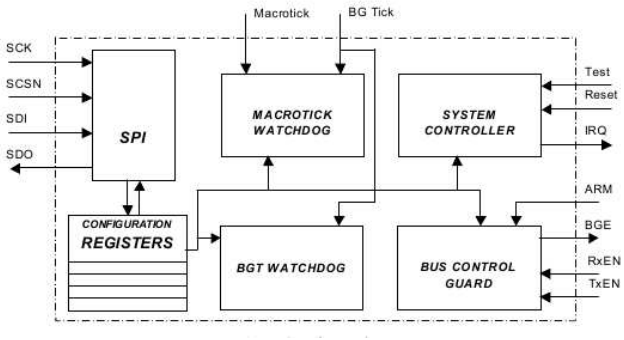

Figura 4.9: Diagramma a blocchi del bus guardian

La funzionalità è determinata dalla configurazione dei registri, programmati dal software attraverso interfaccie seriali. Ci sono due controllori per la rilevazione di comportamenti anomali dei segnali di sincronizzazione. Il *bus control guard* attiva o disattiva l'accesso ai canali di comunicazione. Il *system controller* contiene l'*interrupt controller*, due *finite state machine* e determina il task corrente del bus guardian. Il *finite state machine* prevede il passaggio tra i vari modi di funzionamento del bus guardian.

Ci sono 4 possibili modi di funzionamento:

- *BG-Fail Silent*: in questa modalità l'accesso alla trasmissione è bloccato; il nodo lavora in questo modo dopo che si verifica un errore. Nel caso di errore sono generati gli adeguati interrupt e vengono settati i flag. Questo è anche il modo di transizione tra tutti.
- *BG-Config*: questo modo fornisce accesso in lettura e scrittura ai registri responsabili della configurazione del bus guardian. Il software, girando sul nodo host, deve riconfigurare le sue funzionalità. Questo modo deve essere "visitato" solo dopo il modo *BG-Fail*, per evitare i rischi associati al cambio della configurazione durante la trasmissione.
- *BG-WakeUp*. E' un modo speciale a cui è permesso di svegliare il resto degli altri nodi. Quest'azione avviene inviando un *Wakeup simbol*.

- BG-Guarding: fornisce le operazioni standard del nodo (esempio la comunicazione). Nel modo *BG-Guardian*, il *bus guardian* controlla la comunicazione, rilevando eventuali errori.

Due controllori supervisionano i segnali di sincronizzazione dal *Communication Controller*, *Macrotick* e*BG Tick*. Il *controllore Macrotick* conteggia il numero di *BG Tick* che si succedono in un intervallo di tempo determinato dal numero definito di *macrotick*, riuscendo a controllare la sincronicità e con questo obiettivo i contatori del *controllore Macrotick* sono temporizzati dal segnale *BG Tick*. Il *controllore BGT* controlla il segnale *BG Tick* in modo simile, eseguendo il confronto con il clock principale. Una delle funzioni aggiuntive dei due controllori è l'individuazione del fallimento del segnale di sincronizzazione.

4.3.2 Serial Peripheral Interface

La comunicazione fra il *bus guardian* e l'*host* è basata su *SPI*, che è l'interfaccia seriale che si trova in molti microcontrollori a singolo chip.

Per esempio se si analizza l'*ATMEL*, si può vedere che questo fornisce la comunicazione sincronizzata sia in modalità master che slave, con trasmissione simultanea dei dati in entrambe le direzioni. Il *SCK clock* e il *SCSN slave* selezionano i segnali generati dal master.

Le caratteristiche particolari del *SPI* sono le seguenti:

- L'*host* è sempre master e gli altri blocchi, come il bus guardian, sono slave. Il blocco slave deve rispondere ai vari comandi che arrivano dall' host, includendo l'insieme (collection) dei dati, inviando dati e interrompendo le operazioni in atto.

- L'indirizzamento dei registri del bus guardian. Ci deve essere un meccanismo di selezione che permette l'accesso al registro. In realtà non tutte le versioni del FlexRay hanno delle specifiche che definiscono quest'aspetto. In questi casi è utile considerare che la prima porzione dei dati inviati all'host corrisponde all'indirizzo del primo registro su cui scrivere (in alcuni casi si potrebbero anche configurare tutti i registri ogni volta).

L'host può sempre fermare le operazioni di default inviando un commando di *break*. In questo caso, attraverso delle specifiche esegue la configurazione e la macchina a stati controlla le varie fasi delle operazioni eseguite dal *SPI*. La macchina a stati finiti può trovarsi in uno dei seguenti stati:

- *Idle state*, quando il bus guardian è in attesa che l'host diventi attivo; quando l'host inizia la trasmissione (*SCK* e *SCSN* si accorgono di qualche variazione), la macchina cambia lo stato in *address receive*.
- *Address receive*, quando l'indirizzo è ricevuto nello *shift register* e da qui si passerà nel nuovo stato *data receive*.
- *Data receive*, quando i dati di configurazione sono ricevuti dall'host; questo processo deve essere interrotto da un commando di *break* inviato dall'host; solo in questo caso il nuovo stato è *data trasmit*.
- *Data transmit*, quando il contenuto di tutti i registri è trasmesso uno dopo l'altro all'host; quest'azione può essere fermata dal comando break e in tal caso si tornerà allo stato idle.

Gli *shift register* dell'*SPI* sono sincronizzati dal clock principale (più veloce del *SCK*) e sono bloccati quando si raggiunge il limite del *SCK*. Il limite inferiore del *SCK* determina il momento di raccolta dei dati sicuri.

4.4 ByteFlight

Il FlexRay si basa sul protocollo *ByteFlight* che unisce i vantaggi dei protocolli sincroni e asincroni e garantisce l'integrità dei dati con una velocità di trasmissione di $10Mbit/s$.

Sviluppato principalmente da BMW, Motorola, Elmos e Infineon, viene utilizzato per applicazioni che richiedono alta affidabilità. Al fine di ridurre l'*EMI* è stata sviluppata una soluzione a livello fisico utilizzando la trasmissione ottica. In una rete a stella con una coppia di star intelligenti la comunicazione su una singola fibra è bidirezionale. Il chip di ricetrasmissione, il diodo a emissione di luce, e il fotodiodo sono integrati nel connettore ottico.

Il *ByteFlight* ha fondamentalmente le seguenti caratteristiche (inglobate quindi nel FlexRay):

- Possibilità di trasmissione asincrona e sincrona.
- Elevatissima velocità di trasmissione ottica (fino a $10Mbit/s$ con transceiver ottico).
- Tempi di latenza noti per messaggi di alta priorità e messaggi orientati in base al loro identificativo.
- Flessibile larghezza di banda per ogni nodo.
- Protocollo recente, nato come combinazione di tempo e priorità per l'accesso al bus.
- Formato del messaggio: *ID*, *LEN*, *DATA*, *(0...12)*, *15 bit CRC (h=6)*. È più piccolo del frame del FlexRay e supporta solo 12 byte di dati.

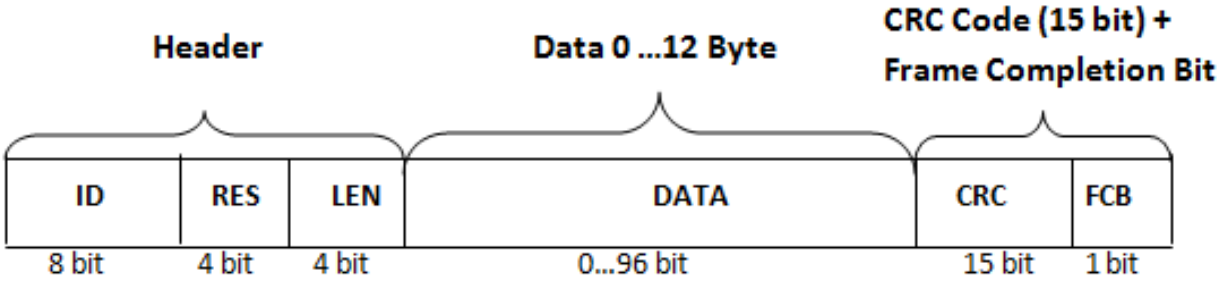

Figura 4.10: Formato del frame ByteFlight

- Comunicazione con libera collisione.
- Velocità di trasmissione massima pari a $10Mbit/s$ e in condizioni di carico del bus maggiore di $5Mbit/s$.
- Protocollo/hardware garantiscono la latenza per una certa quantità di messaggi ad alta priorità e comportamento deterministico.
- Controllo analitico in caso di comportamento anomalo da parte di messaggi ad alta priorità.
- Accesso flessibile al bus per i messaggi a bassa priorità, come i protocolli asincroni (comportamento statistico).

4.5 Formato del Frame

Il FlexRay è diviso in tre segmenti:

- **Header**
- **Pyaload**
- **Trailer**

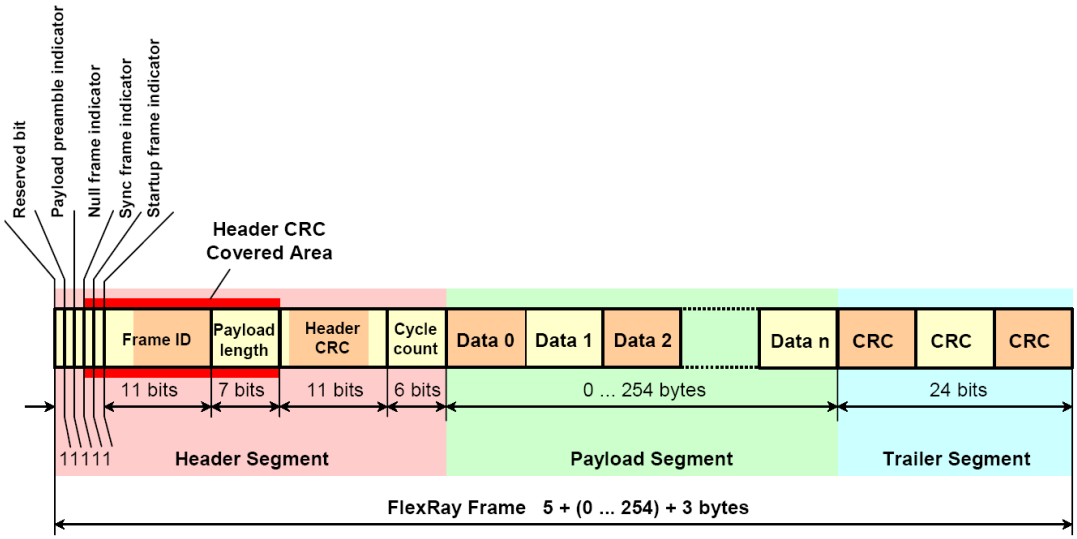

Figura 4.11: Formato del frame FlexRay

Header:

- **Reserved** è un bit riservato per future espansioni.
- **Payload preamble indicator** è un bit che indica l'esistenza di un vettore di informazioni nel segmento del payload del frame.
- **Null frame indicator** è un bit che indica se il data frame nel payload è nullo.
- **Sync frame indicator** è un bit che indica l'esistenza del messaggio di sincronizzazione.
- **Start-up Frame Indicator** è un bit che indica se il nodo che sta inviando il frame è lo *star-up node*.
- **Frame ID** identifica il frame, è usato per la priorità nel segmento dinamico e indica gli slot nella parte statica del frame.
- **Lenght** contiene il numero di byte del campo data.
- **Header CRC** è usato per rilevare gli errori durante il trasferimento.È calcolato inoltre per la sincronizzazione e per il campo DLC.
- **Cycle** indica il numero di cicli del nodo, il quale è incrementato dal controller della comunicazione ad ogni inizio di un nuovo ciclo.

Payload:

- Il range **Data** valido è compreso tra 0 e 254 byte ed è indicato nel campo *DLC*.

Trailer:

- Il **CRC** è calcolato e specificato dall'hardware. Cambia il valore del "seme" sul canale collegato per prevenire errate connessioni.

4.6 Data Trasmission

La comunicazione nel FlexRay si basa sul ripetersi di 64 *communication cycle*, numerati da 0 a 63. Ogni nodo tiene traccia nel *cluster* del *communication cycle* corrente, attraverso una variabile chiamata *vCycleCounter* che va da 0 a 63 ciclicamente. Il *current cycle count* è lo stesso per tutti i nodi del cluster.

La durata di un *cycle communication* è il *macrotick gMacroPerCycle*, che è un parametro globale per il cluster. I *macritick* sono le unità di tempo comuni usati nel cluster. Il trasferimento del frame nel FlexRay è organizzato all'interno del ciclo di comunicazione. A livello più alto livello c'è il *communication cycle level* che è composto da quattro parti principali:

- **Statico**
- **Dinamico**
- **Symbol Window**
- **Network Idle Time (NIT)**

Al livello successivo si ha l'*arbitration grid level*, in cui il *segmento statico* è composto da intervalli di tempo consecutivi chiamati *static slot* e dal *segmento dinamico* composto da intervalli di tempo consecutivi denominati *minislot*.

Ogni *static slot* e ogni minislot sono divisi, nel livello *macrotick*, in diversi *macrotick*. Quando si va ad analizzare a livello *microtik*, ogni *macrotick* è formato da diversi *microtick*.

Nel *segmento statico* il numero, la lunghezza, l'intervallo di trasmissione dello slot è una variabile che assume un valore costante. La lunghezza dello slot statico deve garantire che il frame anche nel caso peggiore venga inviato.

Nel segmento dinamico, l'intervallo di comunicazione e la lunghezza del *minislot* possono variare. I diversi *minislot* sono trasmessi quando arriva la richiesta di trasmissione. La *symbol window* indica la rilevazione di conflitto in trasmissione e ricezione del *frame header* con il bit *startup*. Il tempo di riposo della rete serve al nodo per calcolare e applicare la correzione del clock e per eseguire i processi relativi a quel particolare *communication cycle*. Quindi il communication cycle in funzione del tempo è così schematizzato:

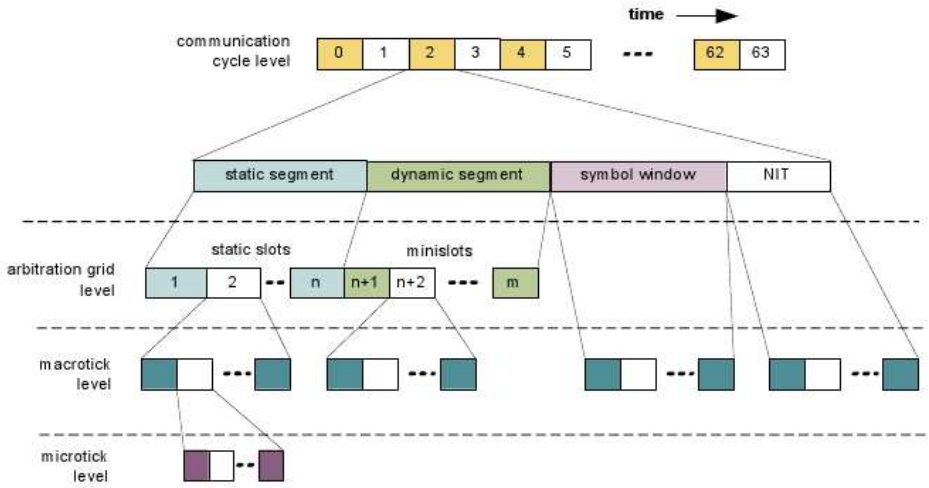

Figura 4.12: FlexRay Communication Cycle

Nel **segmento statico** ci sono i messaggi con priorità alta e la trasmissione segue la strategia *TDMA* fissa. Qui il numero di slot è determinato dalla configurazione del *gNumberOfStaticSlots* e deve avere almeno il valore minimo pari a 2 e massimo pari 1023, ci sono cioè sempre 2 slot statici nel *segment static* per ogni canale.

Vengono inviati e ricevuti messaggi con *time trigger* a lunghezza fissa definita nel *gdStaticSlot* (cioè il segmento statico è formato da time slot della stessa durata). Ogni nodo nel *cluster* mantiene traccia del *current slot count* nella variabile *vSlotCounter* (presente in ogni canale).

È inoltre garantita la protezione del bus guardian, il tempo di latenza e il jitter di questa parte di frame. I nodi che sono connessi a entrambi i canali inviano in modo sincrono su ognuno di essi.

La **parte dinamica** è quella in cui ci sono i messaggi con più bassa priorità e l'invio avviene seguendo l'ordine crescente dell'ID con la tecnica *TDMA* flessibile. Il *dynamic segment* è opzionale nel *communication cycle* ed è usato ad-hoc. Ci sono in questo segmento i *minislot* configurati nella variabile *gNumberOfMinislots*. Se il valore di questa variabile è 0, non c'e il segmento dinamico nel corrente *communication cycle*. La numerazione del *minislot* continua dall'ultimo *macroslot* del segmento statico, cioè se l'ultimo slot del segmento statico ha il numero 9, il primo *minislot* del segmento dinamico avrà identificativo pari a 10. La comunicazione dinamica

inizia solamente se la comunicazione parte con un *minislot* nel segmento dinamico e la dimensione degli slot dipende dal payload associato al frame, che può avere una lunghezza massima in questa parte di frame pari a 254 byte e può variare da uno slot all'altro.

I messaggi vengono inviati e ricevuti con modalità *event trigger* conformemente al protocollo *Byteflight* a lunghezza variabile e non c'è la protezione del bus guardian perchè il tempo di invio è indeterminato. Questo permette di inviare diversi frame su diversi canali allo stesso istante e i frame a più alta priorità prima di quelli con prioritàa più bassa.

Nel *communication cycle* può inoltre essere presente la *symbol window* che è usata per trasmettere i simboli *FlexRay-defined*. La durata della *symbol window* è definita nella variabile *gdSymbolWindow* (misurata in *macrotick*). Se *gdSymbolWindow* è 0 la *symbol window* non è presente.

Per quanto riguarda il *NIT* è usato da ogni nodo per calcolare e applicare la correzione del clock, in pratica si occupa della sincronizzazione. La durata del *NIT* nei *macrotick* è dato dal tempo del *communication cycle* meno il tempo (sempre in termini di *macrotick*) del segmento statico, dinamico e del symbol window.

Quindi il *segmento statico* consiste di diversi *slot statici*, formati da diversi *macrotick*, mentre il *segmento dinamico* consiste di diversi *minislot* formati da diversi *macrotick*. Le unità di tempo *macrotick* sono formate da diversi microtick che sono le unità di tempo che derivano direttamente dal *clock tick* del *communication controller*.

Il primo obiettivo del meccanismo di sincronizzazione in un sistema distribuito è di garantire le differenze di tempo tra i nodi e di restare entro certi limiti di precisione richiesti. Queste differenze possono essere di *offset* e di *rate*.

Per avere una sincronizzazione ottimale del *local clock* il FlexRay utilizza contemporaneamente sia il meccanismo per la correzione dell'offset (*offset correction*) sia quello per la correzione del rate (*rate correction*). L'*offset correction* indica il numero di *microtick* che sono stati aggiunti nel segmento del *NIT*. Il *rate correction* indica un numero intero di *microtick* che sono aggiunti al numero di *microtick* configurato nel ciclo di comunicazione.

La Figura 4.13 mostra la relazione di tempo tra *media access schedule*, *clock synchronization correction* e il *clock synchronization calculation schedule*.

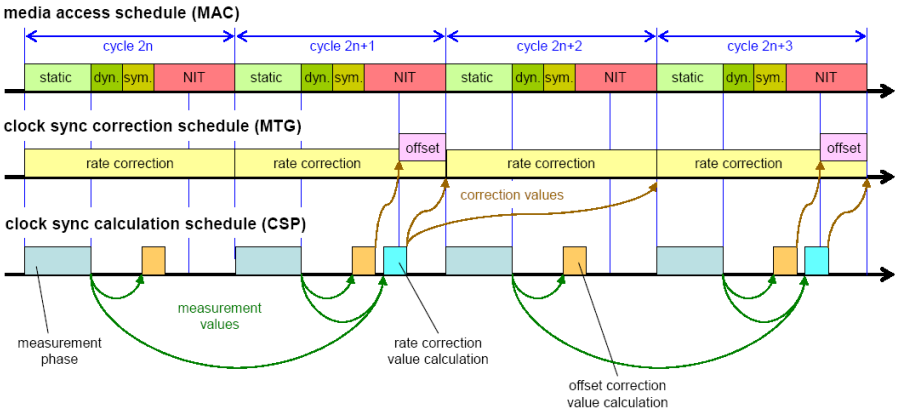

Figura 4.13: Relazione di tempo tra media access schedule, clock sync. schedule e clock sync. calculation schedule

4.7 Panoramica sul message buffer

La Figura 4.14 mostra in dettaglio le due unità di misura di tempo nel FlexRay:

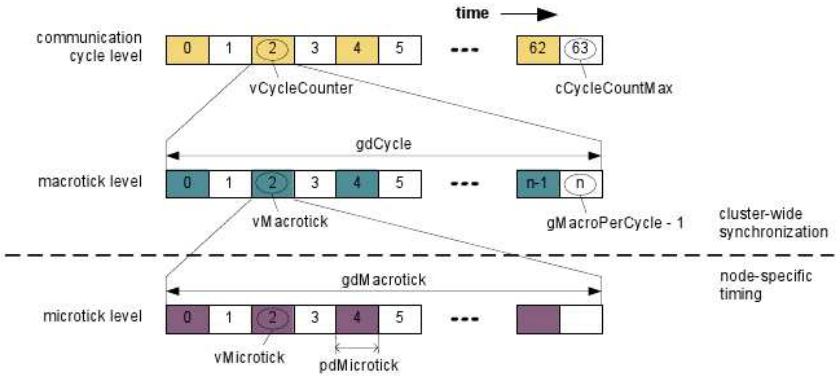

Figura 4.14: FlexRay Macrotick Timing

- *Macrostick* è l'unità di tempo base, ogni slot statico dura un macrotick *gdStaticSlot*. La lunghezza del *communication cycle* è determinata dal *macrotick gMacroPerCycle* mentre il conteggio del corrente macrotick è mantenuto in ogni nodo dalla variabile *vMacrotick*. La durata assoluta del macrotick è determinata dalla variabile *gdMacrotick* e va da 1 a 6 μs.

- *Microtick* è l'unità di tempo locale che deriva direttamente dall'oscillatore. La durata di ogni *macrotick* a livello nodo è espressa in termini di *microtick* e il numero di questi ultimi per *macrotick* (*pMicroPerMacroNom*) può variare da nodo a nodo. Nel nodo, il numero di *microtick* all'interno di ogni singolo *macrotick* può essere diverso e il nodo tenta di tenere la durata media di ogni *macrotick* pari alla variabile *gdMacrotick*.

La trasmissione del messaggio (tra host e protocollo) avviene su buffer *non-queued*, in base alle specifiche definite e alle seguenti caratteristiche:

- L'host ha l'accesso in lettura/scrittura per trasmettere *message buffer data*.
- I dati nuovi sostituiscono quelli vecchi nel *message buffer*.

L'implementazione della trasmissione del *message buffer* consiste di due parti:

- Memoria richiesta per la configurazione, il controllo e lo stato dei dati del message buffer che in genere è implementato come registro *host-accessible* nel *CHI*.
- Regione di memoria condivisa allocata per il message buffer; la dimensione di questa dipende dalla dimensione massima del payload del message buffer. La memoria condivisa contiene l'*header*, il *payload data* e lo *stato* dello slot.

Si consideri il seguente esempio di trasmissione dati, considerando le premesse necessarie. Per avviare una comunicazione dobbiamo settare prima di tutto i seguenti parametri:

- Frame ID
- Canale

Inoltre per ogni buffer, il *CHI* deve fornire le risorse adeguate per permettere all'host di accedere alle informazioni di stato e di controllo quali:

- *Data valid*, indicazioni che permettono all'host di capire se i dati sono validi per la trasmissione.
- *Slot status*, che sono un set di specifiche per riferire sullo stato dello slot.

Altre informazioni di configurazione che devono essere implementate dal *CHI* includono:

- *Cycle count filtering* così che la trasmissione del messaggio avvenga nello slot assegnato nel *communication cycle* assegnato (usato solamente nel segmento dinamico).
- Indicazione della posizione dell'*header* e del *payload data* del messaggio nella memoria condivisa.
- *State o event* tipologia di trasmissione, che determina se il nodo trasmette sempre i dati corretti in ogni slot assegnato o solamente quando i dati sono stati aggiornati dall'ultima trasmissione del message buffer.

4.8 Funzionamento del protocollo

Start up: I nodi devono essere sincronizzati al loro *start* e l'algoritmo di sincronizzazione deve essere distribuito; solo i nodi che sono connessi in entrambi i canali di comunicazione sono autorizzati e vengono fatti partire.

Clock syncronization: Per implementare funzioni di sincronizzazione e ottimizzare la larghezza di banda è necessario avere la più piccola possibile distanza fra due messaggi contigui ed è richiesto che i componenti distribuiti nella rete di comunicazione abbiano un comune tempo di base (*global time*). Perché ci sia effettivamente la sincronizzazione, vengono trasmessi dei specifici messaggi di sincronizzazione nella parte statica del *communication cycle*. Con l'aiuto di speciali algoritmi, il *clock locale* del componente è corretto in modo tale che tutti siano allineati al *global time*.

4.9 Sicurezza e fault tolerance

Nel settore automotive quando oggi si parla di sicurezza e di FlexRay, ci si riferisce alla specifica *IEC6158* per *Functional safety of electrical/electronic/programmable electronic safety-related systems* che permette di determinare i requisiti per i sottosistemi elettronici. Il tipico approccio per ricercare l'adeguato livello di sicurezza è di incrementare il livello di ridondanza dei componenti "critici", a livello livello spaziale, temporale o di informazioni.

Molto importante risulta essere la gestione di questa strategia in quanto si deve riuscire ad applicare la regola fondamentale, *fault-containment regions*, cioè si deve fare in modo che tutti i componenti correttamente funzionanti continuino a lavorare senza considerare i *fault* di componenti vicini o collegati ad essi. Questo generalmente richiede l'uso di sorgenti di potenza e di clock indipendenti, isolamento elettrico delle interfaccie e può richiedere anche separazione fisica per evitare comuni *fault*.

Questi requisiti sono perseguibili in sistemi *multi-chip/multi-package*, mentre le soluzioni *single-chip* non possono essere divise in diverse *fault containment regions* e per questo sono soggette a guasti comuni. Il FlexRay fornisce un'infrastruttura per raggiungere i requisiti di sicurezza e quindi di ridondanza richiesti per ogni specifica applicazione.

Un aspetto che deve essere adeguatamente considerato quando si parla di questo protocollo è la *fault-tolerance* e più precisamente la *fault-tolerance scalabile*. Questa caratteristica permette al FlexRay di essere usato economicamente in sistemi distribuiti non *fault-tolerance* come se fossero *fault-tolerance*. La sola ridondanza

non garantisce la *fault-tolerance*, per questo il FlexRay fornisce ulteriori specifiche perché ciò sia raggiunto:

- *Flessibilità topologica* (singolo contro doppio canale, connettività mista).
- *Clock sinchronization fault-tolerance*, utilizzabile anche quando si ha non fault tolerance.
- Separazione concettuale di dominio funzionale e strutturale.

Per supportare il concetto di *fault-tolerance scalabile* sono stati presi opportuni accorgimenti.

Il primo è che il FlexRay deve supportare diverse interconnessioni topologiche, il progettista dovrà scegliere se lavorare con un solo sistema a singolo canale, a doppio canale oppure a doppio canale con connettività mista, dove alcuni nodi sono connessi a un solo canale e alcuni a entrambe. Inoltre il FlexRay può essere sviluppato usando a scelta il canale *guardian* locale o remoto che protegge i canali di comunicazione dai *fault* in trasmissione che violano lo schema *TDMA*.

Il secondo è che l'algoritmo di *clock sinchronization* supporta la sincronizzazione *fault-tolerance* e *non fault-tolerance*. Per la *fault-tolerance* l'algoritmo considera i fault *transitori* (*transient fault*, intesi come singoli eventi di disturbo causati in genere da radiazioni esterne o disturbo elettrico che causano ionizzazione e *permanenti* (*permanent fault* eventi che causano il blocco del sistema o riducono drasticamente la sicurezza dello stesso) come quelli simmetrici e asimmetrici.

Infine il FlexRay rispetta la separazione del dominio funzionale da quello strutturale che presentano requisiti concettuali separati. La tabella 4.1 mostra i concetti chiave su cui si basa il FlexRay per operare queste scelte e ne mostra le differenze con il *CAN* e il *ByteFlight*.

4.10 Errori di gestione

Possono presentarsi tre livelli di errori durante il processo, *normal active*, *normal passive* e *fault*:

- **Normal active**: Il bus controller opera normalmente, non vengono rilevati errori.

- **Normal passive**: L'errore rilevato può essere fissato. In questo stato, il bus controller non permette di trasmettere nessun dato, ascolta il bus e cerca di reinserirsi appena possibile.
- **Fault**: Errore fatale, il controllore ferma completamente le operazioni.

4.11 CAN

CAN sta per *Controller Area Network*, che è una delle prime, più resistenti e oggi più utilizzate reti di controllo in Automotive sviluppato dalla Bosch nella seconda metà degli anni ′80. Originariamente non era stato sviluppato come protocollo per garantire la sicurezza di sistemi *real-time*, ma come elemento elettronico all'interno del veicolo.

La rete ha una struttura del tipo *Master/Slave*, dove un nodo master della rete assume il controllo dell'accesso al bus interrogando ciclicamente in sequenza gli altri nodi (slave). Una tipica struttura all'interno del veicolo, può contenere 2 o 3 separate reti CAN operanti a differenti velocità di trasmissione dati.

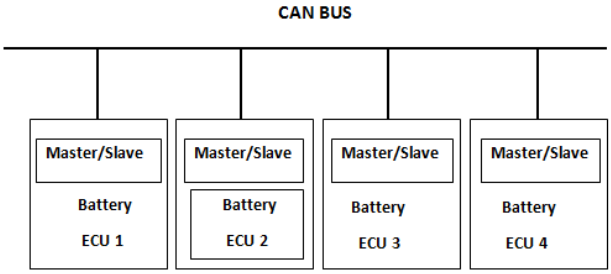

Figura 4.15: Struttura master/slave con il protocollo CAN

Le basse velocità (< 125 kbit/s) vengono normalmente utilizzate per gestire comfort e i dispositivi secondari dell'autovettura come la regolazione dei sedili, specchi, finestrini. In questo caso è possibile anche un funzionamento in *sleep mode* che permette un arresto degli oscillatori dei rispettivi nodi fino a quando un nuovo messaggio diretto al nodo interrogato lo riattiva. Con questa modalità si ottiene un risparmio energetico della batteria.

Le alte velocità (fino a 1 Mbit/s) vengono utilizzate per la gestione complessiva dei dispositivi del motore, per il controllo dei freni, per il sistema di guida e via dicendo, cioè per tutti quei sistemi in cui si richiedono brevissimi tempi di attuazione e/o di acquisizione.

Per l'elevata velocità di trasferimento dati è necessaria una schermatura del bus contro le radiazioni elettromagnetiche presenti.

Nel *CAN* il bus non viene occupato secondo uno schema definito: l'accesso al bus è di tipo casuale. Tutti i componenti collegati sul bus hanno gli stessi diritti (peer-to-peer), "ascoltano" in parallelo tutti i messaggi, sono sempre pronti a ricevere e,

quando necessario, iniziano a trasmettere se nessun altro componente sta inviando messaggi sul bus in quel momento. Si rende quindi necessario, un sistema di controllo che gestisce gli accessi, cioè il *Carrier Sense Multiple Access*. Il vantaggio è dato dalla possibilità di attuare la comunicazione "event-driven", ossia la trasmissione viene inizializzata solo quando necessario.

Quindi abbiamo una comunicazione di tipo *event-triggered*, questo significa che i nodi invieranno il messaggio a seguito dell'inizio di un evento all'interno del nodo. Il tempo di quando questi messaggi vengono inviati non è possibile prevederlo, non appena il sensore si accorgerà che il valore è cambiato il nodo invierà il messaggio sul bus.

Questo può provocare collisioni fra i messaggi e per far fronte a questo il *CAN* usa la tecnica di accesso *CSMA/AMP* (*Carrier Sense Multiple Access with Arbitration by Message Priority*). Questo è la miglior variante alla tecnica di accesso *CSMA/CD* (*Carrier Sense Multiple Access with Collision Detection*). Nel *CDMA/CD* tutti i nodi che rilevano una collisione sul bus devono bloccare l'invio e aspettare un tempo random prima di riiniziare la trasmissione.

Nel *CSMA/AMP* ad ogni messaggio da inviare viene data la massima priorità e il nodo può iniziare a trasmettere solo se il bus è libero. Se viene rilevata una collisione dopo l'invio del messaggio, quello con priorità più alta non verrà mai bloccato.E' facile capire che la criticità di questo sistema si ha quando il nodo deve attendere che il bus sia libero.

Il contenuto di una comunicazione (per esempio il numero di giri o la temperatura del motore, un comando di un attuatore ecc.) viene contrassegnato da un identificatore inequivocabile in tutta la rete. L'identificatore determina anche la priorità del messaggio: questo è molto importante per l'assegnazione del bus, quando esistono più nodi che concorrono per il diritto di accesso. Rispetto ad un accesso al bus di tipo deterministico, questa soluzione comporta un carico medio molto inferiore sul bus ed un tempo di latenza molto breve. Per contro, in questo tipo di comunicazione i tempi di risposta non sono definibili proprio per la natura del tipo di accesso. In questo protocollo ci sono 5 diversi modi per rilevare gli errori:

- *Monitoring*. Il mittente controlla che il bit inviato è lo stesso sul bus. Ricordando che, con lo 0 si indicano i messaggi con più alta priorità e con 1 quelli con priorità inferiore, se è inviato uno 0 ed è rilevato un 1 allora è avvenuto un errore.

- *Bitstuffing*. Il CAN usa la codifica *NRZ* per la rilevazione degli errori nel frame e un errore nel frame inizia con sei 1 oppure con sei 0. Nessun altro messaggio

conterrà stringhe di questo tipo poichè il trasmettitore inserisce il *bit stuf* (che verrà rimosso in ricezione) con un valore complementare rispetto ai 5 che lo precedono.

- *Frame check*. Ogni nodo controlla il frame ricevuto, se viene trovato un errore o un'anomalia rispetto alle specifiche segnala "format error".
- *ACK errors*. Ogni nodo dovrebbe riferire se la ricezione del messaggio era corretta oppure no scrivendo 0 per l'ok o 1 in caso contrario nel campo ack del messaggio.
- *Cyclic Redundancy Check* (CRC). Viene calcolato il checksum sul messaggio prima dell'invio e è trasmesso sempre con il messaggio. Il ricevente esegue lo stessa operazione e se il risultato è uguale si ha una conferma che il messaggio ricevuto è corretto.

Da quanto descritto è sicuramente possibile dire che questo protocollo è adatto per sistemi *real-time* ma per sistemi *safety-critical* la tecnica *event-triggered* è mediocre.

Il bus *CAN* è estremamente semplice e per questo molto affidabile e sicuro, ma ha un limite importante che ne pregiudica l'impiego in sistemi dove l'affidabilità deve essere assoluta.

4.12 TTCAN

Rispetto al CAN il *TTCAN* è un protocollo di livello superiore che sincronizza la comunicazione e fornisce il *global time-base*.

Il *TTCAN* è diviso in due livelli ed è standardizzato in parte come lo standard *CAN*. Per raggiungere la schedulazione sincronizzata in un sistema distribuito tutte le attività devono avere un comune *time base*, ed è per questo che tutti i nodi che usano questo protocollo utilizzano un contatore che indica il *global time base*. Il *counter* deve essere sempre aggiornato e il messaggio di aggiornamento è inviato dal *time master* con una frequenza molto alta in modo da permettere a tutti i nodi di aggiornarsi tempestivamente ed essere quindi sempre sincronizzati.

Il *TTCAN* implementa l'architettura *time-triggered* e la schedulazione avviene utilizzando i *time-slot*. In ogni slot può essere inviato solamente un messaggio, cosi viene evitata la competizione per il bus è la latenza del messaggio non dipende più dal numero di nodi connessi alla rete; in questo modo può essere calcolato il caso peggiore di latenza.

Questo protocollo può essere usato per sistemi *X-by-Wire* con sistemi meccanici/idraulici ma non per sistemi esclusivamente elettronici in quanto le sue specifiche non rispettano completamente i requisiti di sicurezza. Un grosso vantaggio del *TTCAN* è che può sfruttare i tool già sviluppati per la rete *CAN*, come il *CANalyzer*. Inoltre i nodi di una rete *CAN* possono essere usati come nodi passivi in una rete *TTCAN*, in pratica possono ricevere ma non inviare messaggi in una topologia di rete che utilizza il protocollo *TTCAN*. La tecnica d'accesso utilizzata è la *TDMA* (*Time-Division Multiple Access*) che divide il dominio del tempo in piccoli slot. In alcuni *time slot* sono permessi degli *event message* e in questi casi è possibile che più di un trasmettitore abbia accesso al bus nello stesso istante. Quando ciò accade la trasmissione verrà regolarizzata in base alla priorità dei task. Questo significa che con questo protocollo ci possono essere *time slot* in cui l'accesso è gestito con tecnica *TDMA* e *time slot* in cui è pre-definito l'utilizzo della tecnica *CSMA*. Questa caratteristica, insieme al fatto che alcuni messaggi in pre-determinati slot possono essere *even-based*, mette in evidenza la flessibilità del *TTCAN*. Il *TTCAN* si basa su alcuni elementi fondamentali:

- *Reference message.* La comunicazione periodica è sincronizzata grazie alla trasmissione di un messaggio di riferimento da parte *time master*. Il *reference message* è un regolare messaggio *CAN* con uno speciale identificatore noto a priori.

  A livello 1 del *TTCAN* il primo byte è di informazione di riferimento, a livello 2 dal primo al quarto byte vengono contenute le informazioni per la sincronizzazione e il *global time*, mentre gli ultimi tre bit del *reference message* determinano la priorità nel set dei potenziali *time master*. La priorità determina quale nodo inizierà o continuerà ad inviare e quale fallirà.

- *Basic cycle.* È il periodo tra due reference message consecutivi e contiene la finestra di tempo pre-definita per la trasmissione dei messaggi nel sistema. La lunghezza del *base cycle* è sempre la stessa mentre le finestre all'interno di un *base cycle* possono essere diverse di dimensioni e di tipi diversi per permettere la trasmissione di messaggi periodici e di messaggi attivati da un evento.

- *Matrix cycle.* Viene costruita per creare maggiore flessibilità . La *matric cycle* definisce la schedulazione dei messaggi di trasmissione. Un nodo in una rete *TTCAN* non ha bisogno di conoscere tutto il sistema matrice, ma è sufficiente che abbia le informazioni dei messaggi che dovrà inviare e ricevere. Questo permette non solo di ottimizzare l'uso della memoria nei nodi ma anche di facilitare la schedulazione e la gestione di possibili situazioni anomale.

- *Window type.* In *TTCAN* si sono diversi tipi di finestre temporali (*time window*):

- *Exclusive time window.* Questo tipo di finestre sono assegnate ai messaggi periodici, nessun altro messaggio può essere schedulato con la stessa finestra. La ritrasmissione automatica dei messaggi *CAN* non è permessa in queste finestre temporali.

- Arbitrating time window. Questa permette l'invio di messaggi spontanei e può essere assegnata a più di un nodo e i possibili conflitti che possono nascere sono gestiti con le politiche viste in precedenza. La trasmissione dei *spontaneous message* possono essere inviati solo se il tempo necessario è sufficiente e la ritrasmissione dei messaggi *CAN* non è permessa.

- *Free time window.* Sono finestre lasciate libere per usi futuri.

- *Sending, receveing* e *NTU.* All'interno del *basic cycle* le azioni del *time triggered* sono guidate dalla progressione del tempo. Questa percezione del tempo è chiamata *cycle time* ed è resetato all'inizio di ogni *cycle.* La connessione tra *cycle time* e *matrix time* è chiamata *time mark.* La *time mark* specifica l'inizio della *time window* e potrebbe essere *TxTrigger* per la trasmissione, o *RxTrigger* per la ricezione. La *time mark* consiste di una *base mark* che determina il numero del primo ciclo di base dopo l'inizio della matrice quando i messaggi devono essere processati e il *rereat count*, che non è altro che il numero di *basic cycle* tra due successive trasmissioni/ricezioni del messaggio.

- *Level difference.* Livello 1 è una versione meno esigente di *TTCAN* in cui la tempistica è solo sulla base delle trasmissioni dei *reference message.* Ogni nodo ha un contatore che viene aggiornato dal *NTU.* La sincronizzazione a livello 2 è più fine e gli *NTU* sono aggiornati ogni 2^n.

4.13 TTP

Il *TTP* è un altro "concorrente" del FlexRay sia perchè ha ottime prestazioni sia perchè è utilizzato nel campo dell'automatica oltreché nell'industria aereomobile.

Progettato per sistemi distribuiti in *real-time* che sono *critici* e *tolleranti* ai gusti, il *TTP* (*Time Triggered Protocol*) assicura che non ci siano punti di guasto o difetti del sistema.

La tolleranza ai difetti (*fault tolerance*), come già detto, è la capacità di un sistema di operare anche in presenza di guasti. Questo comportamento può essere ottenuto con l'uso della *ridondanza*, cioè si ha ridondanza ogni qual volta viene inserita una risorsa che non è necessaria se il sistema è senza errori. La ridondanza si presenta sotto forma di circuiti addizionali hardware o come aggiunta di codice in un programma.

Questo protocollo è nato alla fine degli anni '90 all'Università di Vienna in collaborazione con la *TTTech Computertechnik GmbH*, per essere utilizzato nei sistemi alternativi alla frenatura idraulica e/o meccanica tradizionale, cioè tutti quei sistemi in cui è richiesta alta affidabilità e sicurezza. In realtà la versione sviluppata per la sicurezza dei sistemi critici è il *TTP/C*. La sua caratteristica particolare è che implementa un indipendente *bus guardian* dentro lo stesso *controller TTP*.

L'utilizzo del *TTP/C* porta ad una velocità di trasferimento dati doppia rispetto a quella massima del *CAN*, si arriva quindi fino a 2 Mbit/s. Il *TTP/C* consiste di un numero di nodi connessi ad ogni altro attraverso canali replicati. Ogni nodo è considerato come *failt-silent*, per esempio ogni funzione corretta non deve interagire con il resto del sistema se è stato registrato un errore interno. Questo protocollo è *time-triggered* in quanto la schedulazione è decisa prima del *run-time*, ogni nodo tiene le informazioni dell proprie azioni sul bus su un elenco chiamato *MEDL*. In questa *descriptor list* è specificato quando il nodo preposto a trasmettere o ricevere informazioni relative alla sincronizzazione.

Il *TTP/C* fornisce i servizi di seguito elencati:

- Membership service
- Servizio di sincronizzazione del clock fault tolerant.
- Supporto change mode.
- Rilevazione dell'errore con piccola latenza.
- Gestione distribuita della ridondanza.

La tecnica di accesso utilizzata è *TDMA*, dove il tempo è diviso in slot che rappresentano le più piccole unità sostituibili (*SRU*). L'unità sostituibile è un modulo elettronico che rappresenta la più piccola unità sostituibile in caso di *fault*; questa può sempre monitorare e ricevere messaggi dal bus.

La *SRU* consite di un computer host che si interfaccia al *TTP/C controller* grazie all'interfaccia di comunicazione di rete (*CNI*) e ai sensori e agli attuatori attraverso

un'interfaccia input/output. Quando l'applicazione vuole inviare il messaggio questo è posto nel *CNI* in un determinato istante di tempo e il *controller* assume e gestisce la trasmissione. Quando un messaggio è ricevuto viene posizionato nel *CNI* per poter essere recuperato dall'applicazione nel momento desiderato. Il *CNI* contiene anche il controllo e lo status delle posizioni così l'applicazione può controllare il proprio *controller* e ottenere informazioni sullo stato di tutta la rete e del controller locale.

Il *CNI* può essere diviso in due grandi aree:

- *Stutus/Control* è usato per il controllo dello stato degli host e il controllo del controller.
- *Message area* è quella in cui i messaggi inviati e ricevuti sono presi prima che venga intrapresa l'azione.

Una sequenza di slot *SRU* forma un *TDMA round*, mentre una serie di *TDMA round* che si ripetono uno dopo l'altro formano un *cluster cycle*. Il *cluster cycle* è ripetuto periodicamente per tutta l'esecuzione del sistema. Questo protocollo contiene due tipi di frame:

- *I-Frame* (*initialization frame*) usati per inizializzare il sistema e contengono lo stato interno del *TTP controller*.
- *N-Frame* (*normal frame*) sono utilizzati durante le normali operazioni e contengono i dati inviati tra i nodi de sistema. Gli indirizzi di destinazione del frame non stanno nel frame ma nel *MEDL* permettendo di ridurre l'*overhead* e garantendo una maggiore efficienza nell'utilizzo del bus.

Lo schema di messaggio che deve essere inviato e ricevuto viene memorizzato in una struttura di dati statici chiamata *descriptor list* (*MEDL*) prima dell'avvio del sistema. Tutti i *controller* agiscono in base a questa lista e il *global time* li tiene sincronizzati.

La *description list* contiene due strutture dati:

- *Configuration Parameter* contiene 5 (*Controller Configuration Parameter*, *Guardian Parameter*, *Reconfiguration Role List*, *Mode Control Block*, *Implementation Specific Parameter*) blocchi con i parametri per la configurazione del controller.

- *Trasmission block* contiene tutte le informazioni necessarie per la trasmissione e la ricezione dei frame.

Il *TTP* usa il *Controller State* (*C-state*) per garantire la consistenza del sistema, questo è costituito da tre campi:

- *MEDL* contiene il current mode e la posizione nel MEDL.
- *Time* che indica il corrente global time.
- *Membership vector* che può variare nelle dimensioni da uno a quattro parole in base al numero di *SRU* nel cluster.

Se tutti i nodi concordano su queste informazioni il sistema è consistente. Se le informazioni del *C-state* fra chi invia e chi riceve sono diverse, quest'ultimo non può capire il messaggio che verrà quindi scartato.

4.14 LIN

Il *LIN* nasce alla fine degli anni '90 da un gruppo di lavoro, formato da cinque aziende automobilistiche (Audi, BMW, Daimelr-Chryler, Volvo e Volkswagen), da un'azienda di informatica (Volcano Communication Technologies) e da un'azienda di semiconduttori (Motorola), che hanno creato un consorzio con l'obiettivo di realizzare un nuovo protocollo di comunicazione seriale a basso costo (soprattutto per veicoli), dove la larghezza di banda e la versatilità del *CAN* non erano richieste. Il *LIN* (*Local Interconnect Network*) fornisce una comunicazione seriale su bus costituito da un singolo filo con una bassa velocità di trasferimento dati (20 kbit/s). Questo protocollo risulta essere complementare al *CAN* ma non in grado di sostituirlo. Il *LIN* viene utilizzato come sottorete locale del *CAN*: per esempio per il controllo degli azionamenti dei finestrini, specchi, sedili, temperature, ecc. Le principali caratteristiche di questo protocollo sono le seguenti:

- Ha una configurazione *single-master multiple-slave*, quindi non è direttamente compatibile col *CAN* e richiede un nodo di interfaccia.

- Come conseguenza del tipo di configurazione, non è necessario un sistema di arbitraggio.

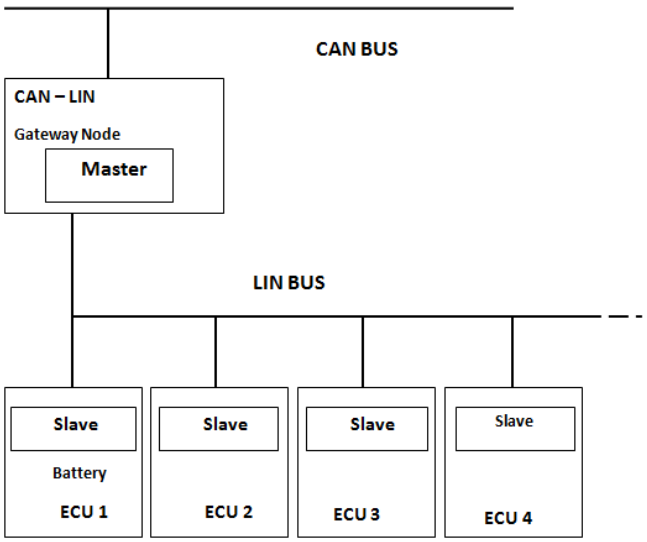

Figura 4.16: Esempio di topologia della rete LIN

- E' una rete a basso costo (*single-wire*) e la sua implementazione è basata su una comune interfaccia hardware *UART/SCI*.
- Mancanza di oscillatori ceramici al quarzo nei nodi slave, in quanto si utilizza un proprio meccanismo di sincronizzazione (riduzioni costi dei nodi).
- La bassa velocità di trasferimento (fino a 20 $kbit/s$) limita i problemi di interferenza elettromagnetica (*EMI*).
- Garanzia sui tempi di latenza.

Un nodo di rete *LIN* non ha informazioni circa la configurazione del sistema complessivo, eccetto per il nodo master. Questo porta una importante conseguenza: i nodi *LIN* possono essere aggiunti anche in tempi successivi alla configurazione di rete, senza modiche hardware e/o software in altri nodi slave e questo diventa un vantaggio dal punto di vista della semplicità e flessibilità.

In letteratura internazionale il *LIN* è affiancato al *CAN*, cioè il suo utilizzo è abbinato a quello del *CAN* come standard della comunicazione seriale via bus in automotive.

4.15 D2B e MOST

Nei primi anni ′90 Matsuhita e Philips si unirono per sviluppare il *D2B (Domestic Data Bus*), quale protocollo di comunicazione utilizzato per sistemi audio-video, per computer portatili e per applicazioni di automotive media. L'architettura è un semplice anello di fibra ottica formato dai singoli dispositivi (GPS, lettore CD, telefono mobile, speaker, fax, sistemi satellitari, ecc..) e dai collegamenti *point-to-point*.

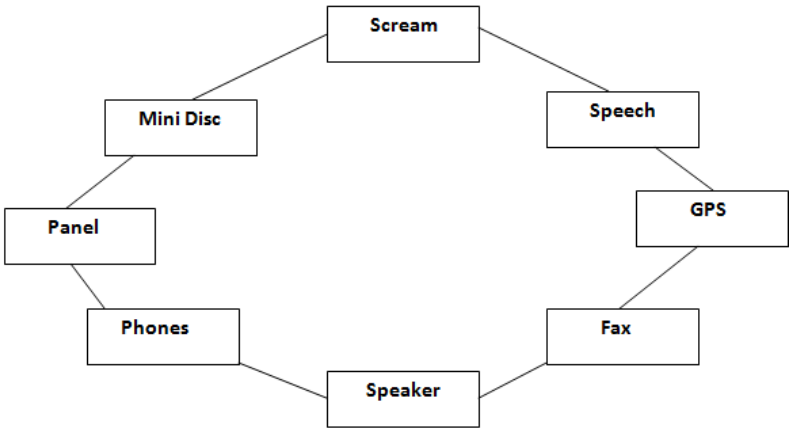

Figura 4.17: Possibili dispositivi in un anello D2B

Lo scambio di trasmissione dati avviene a velocità di trasmissione di circa $12 Mbit/s$. Daimler-Chrysler è stata la prima azienda a implementare il sistema ottico *D2B* su proprie vetture di serie.

Il *MOST* (*Media Oriented System Transport*) è un protocollo con rete a fibra ottica ed una capacità di grandi volumi di dati scambiati. L'architettura è simile a quella del *D2B*, con la differenza di una maggiore velocità di trasmissione dati (40 Mbit/s). E' stato sviluppato grazie ad consorzio di aziende fra le quali Audi, BMW, Daimler-Chrysler ed Oasis Silicon Systems, con l'obiettivo di migliorare il *D2B*. Entrambi questi sistemi hanno una modalità molto utile in applicazioni automotive, dove l'alimentazione viene solitamente fornita dalla batteria, lo *sleep mode*. Questo permette, in mancanza di attività sul bus, il minimo consumo di energia da parte dei chip del sistema, che ovviamente vengono automaticamente riattivati da una nuova attività sul bus.

Ci sono inoltre altri protocolli realizzati ma che sono scomparsi come quello americano *J1850* nella versione *SPC* di Ford e nella versione *DLC* di *GM/Delco*.

Lo stesso discorso si può fare per il protocollo giapponese *PalmNet* e per quello francese *VAN*. In particolare quest'ultimo è stato utilizzato da Renault fino alla metà degli anni '90, sostituendolo poi con il *CAN*.

C'è da dire che in passato anche il protocollo di comunicazione bluetooth è stato affiancato al *CAN* per applicazioni automotive. Non ha avuto molto successo questa tipologia architetturale perché la velocità dei dati di trasmissione con questa strategia è troppo bassa, $1Mbit/s$ operando alla frequenza libera ISM $2.4GHz$. In realtà viene utilizzata per effettuare monitoraggio da remoto in tempo reale (di controllo e/o di diagnostica) sull'intera struttura di bordo. Presenta tuttavia grandi vantaggi dal punto di vista del cablaggio interno all'autovettura (riducendo per esempio la probabilità di un corto-circuito fra cavi o connettori), si riduce il peso (importanti ai fini del consumo) e la potenza in gioco.

4.16 Protocolli a confronto

La tecnologia time-triggered è il risultato di 20 anni di studi svolti presso l'Università di Vienna. FlexRay è stato creato perchè era necessario avere un protocollo più flessibile che utilizzasse la tecnica *time-triggered*.

TTCAN è, fra questa tipologia di protocollo, il meno conosciuto e utilizzato, a differenza del *TTP* e al *CAN*, che verranno probabilmente sostituiti dal FlexRay. In generale si può dire che questi protocolli hanno delle caratteristiche comuni, ma sono le loro particolarità che determinano il loro maggiore o minore successo. In generale i requisti base sono:

- *Maggiore larghezza di banda.* La larghezza di banda del *CAN*, che è limitata a $500kbit/s$ non è adeguata per le future applicazioni.
- *Determinismo.* Per riuscire a capire e a prevedere il comportamento del sistema è condizione necessaria il determinismo.
- *Fault tolerance.* L'architettura deve essere fault tolerance per dare maggiore sicurezza dei sistemi meccanici odierni.
- *Supporto per il controllo distribuito.* L'algoritmo per il controllo distribuito ha bisogno della comunicazione sincronizzata.
- *Unificare nei veicoli i sistemi bus.* È necessario ridurre il numero di differenti architetture bus all'interno dei veicoli. Un sistema bus non basterà, ma 2 o 3 dovranno essere sufficienti.

TTP possiede inoltre la *composability*, che altro non è se non l'abilità dei suoi differenti componenti di lavorare senza mettere a rischio la sicurezza del sistema, cioè provocando anomalie in termini di tempo e/o di valori.

C'è un conflitto fondamentale tra *flessibiltà* da una parte e *sicurezza* dall'altra. In una situazione non flessibile il nodo decide di inviare il proprio messaggio in un tempo definito prima dello start. Se i messaggi non arrivano nel tempo previsto la rilevazione dell'errore e il recupero può iniziare immediatamente. Quando abbiamo a che fare con un protocollo di tipo flessibile il nodo può inviare più messaggi e gli altri nodi non sanno se il messaggio è stato inviato in ritardo perchè non hanno informazioni sul messaggio stesso. In questo caso non è possibile rilevare immediatamente l'errore.È facile capire che il primo caso risulta molto più rivolto alla sicurezza rispetto al secondo, ma dall'altra parte è molto meno flessibile. Le specifiche del *TTP* non accettano nessuna comunicazione flessibile nella rete, sono permessi solo *time-slot* predefinite. Quindi questo è il modo con cui viene garantita la sicurezza nel *TTP* mentre il FlexRay punta ad avere maggiore alla flessibiltà usando il protocollo *ByteFlight* e garantendo alti livelli di sicurezza in sistemi *X-by-Wire*.

Quando si parla di *efficienza* e di *throughput* dei dati, *TTP* è il protocollo più efficace nella trasmissione corretta dei dati, con un'efficienza del $70 - 80\%$. Questo può essere raggiunto perchè il *TTP* ha un più ampio campo dati rispetto al FlexRay che infatti raggiunge un'efficienza pari al 50% a pari condizioni del *TTP*.

Anche la velocità di trasmissione è maggiore nel *TTP* in cui si ha un valore massimo pari a 25 Mbit/s rispetto ai 10 Mbit/s nel FlexRay. Rispetto all'ultima versione del protocollo, le specifiche del campo dati del FlexRay sono state aumentate e ora sono uguali a quelle del *TTP*. Il discorso è molto orientato al solo confronto fra FlexRay e *TTP* perchè in realtà il *TTCAN* non ha ancora raggiunto livelli di sicurezza paragonabili a quelli di questi 2 protocolli e poichè questo è un requisito fondamentale ha tanto senso paragonarlo al FlexRay e al *TTP*. Il *TTCAN* non supporta ancora la ridondanza della comunicazione che è un requisito indispensabile per la *fault tolerance*. Naturalmente i confronti fra i diversi protocolli sono fatti in base a documenti che non sempre sono obiettivi, documenti che non vengono aggiornati per mantenere la confidenzialità dei risultati ottenuti, se non in relazione al lavoro svolto in questa tesi e alle considerazioni che se ne potranno trarre. Riassumendo dalla tabella 4.2 è possibile apprezzare facilmente le differenze fra i diversi protocolli:

Criterio di valutazione	CAN	ByteFlight	FlexRay
Concetto	Event-triggered	Event-triggered	Time/event-triggered
Banda	Carico su diversi sub-bus Can	$10Mb/s$	Rete con data rate da $5Mb/s$ a $10MB/s$, con uso flessibile della banda
Topologia	Richiede bus con stato dominante/recessivo	Stella e bus	Stella che fornisce caratteristiche elettriche isolamento dai fault, flessibile
Comunicazione deterministica	Comportamento non-deterministico e ad alto carico del bus la qualità è bassa	Latenza deterministica per messaggi ad alta priorità	Latenza deterministica (garantito il tempo di trasmissione nel segmento statico)
Fault-tolerance	Non applicata	Sono presenti alcune applicazioni, come la comunicazione deterministica	Si considera e applica la fault-tolerance, come la ridondanza del canale, fault-tolerance clock synchronization
Integrazione di sistemi	Può causare effetti indesiderati e imprevedibili	Non cambia la sincronizzazione	Non cambia la sincronizzazione
Sincronizzazione	Richiede comunicazioni addizionali	Il time base del master serve per la sincronizzazione	La sincronizzazione dei task è realizzata con il time base

Tabella 4.1: Fault-tolerance a confronto

Caratteristiche	CAN	TTP	ByteFlight	FlexRay
Messagge Trasmission	asincrono	sincrono	sincrono e asincrono	sincrono e asincrono
Message identif.	Message identifier	Time slot	Message identifier	time slot
Livello fisico	ricetrasmettitore $1Mbps$	Non definito	ricetrasmettitore ottico $10Mbps$	10 $Mbps$ con segnali differenziali
Clock syncronization	Non fornito	Distribuito, in un range di s	By master, in $100s$	Distribuito, in un range di s
Lattenza, jetter	Dipendente dal carico del bus	Costante per tutti i messaggi	Costante per messaggi ad alta priorità	Costante per tutti i messaggi
Determinazione e limitazione degli errori	Non fornito	Fornito con uno speciale ricetrasmettitore a livello fisico	Fornito dalla fibra ottica e dal chip ricetrasmettitore	Fornito con uno speciale ricetrasmettitore a livello fisico
Bloccaggio di errori	Non fornito	Solamente dal bus guardian	Fornito attraverso la coppia di stella	Fornito dalla coppia di stella o bus guardian
Estensibilità	Molto buona in applicazioni non-time critiche	Solo se l'estensione è prevista nel progetto originale	Estensione possibile per messaggi con alta priorità	Separazione del dominio strutturale e funzionale
Flessibilità	Flessibile larghezza di banda per ogni nodo	Solo un messaggio per nodo e cycle TDMA	Larghezza di banda flessibile per ogni nodo	Multislot per nodo, dynamic segment

Tabella 4.2: Caratteristiche a confronto

Capitolo 5

Modellazione del sistema di frenata e simulazioni

5.1 Premessa

In questo capitolo si è realizzato un sistema che simula il comportamento di un'autovettura: in particolare il suo moto, la sua frenata e il bus FlexRay distribuito attraverso il quale vengono inviate le informazioni che viaggiano nella rete e che vengono ricevute dai diversi nodi. In particolare si è lavorato per simulare, cercando di rispettare il più possibile la realtà, la risposta del veicolo ad un input esterno, la forza sul pedale che genera la frenata, e come i dati vengano scambiati attraverso il protocollo di comunicazione FlexRay.

Per sviluppare il progetto si è utilizzato *MATLAB* e in particolare si sono sfruttate le potenzialità di *Simulink*. In generale il modello descrive un autoveicolo, formato da un nodo principale e da altri quattro nodi, che sostanzialmente identificano le ruote.

5.2 Introduzione

All'avvio del progetto apparirà la schermata mostrata in Figura 5.1 composta dal blocco *KeyOn* che può assumere i valori 0 e 1 indicando rispettivamente macchina in moto o macchina spenta; nel caso in cui questo blocco non fosse a 1 nessuna simulazione potrebbe partire e inoltre se la macchina viene spenta durante la sua corsa la velocità torna a 0. Durante questo capitolo verranno utilizzate spesso al-

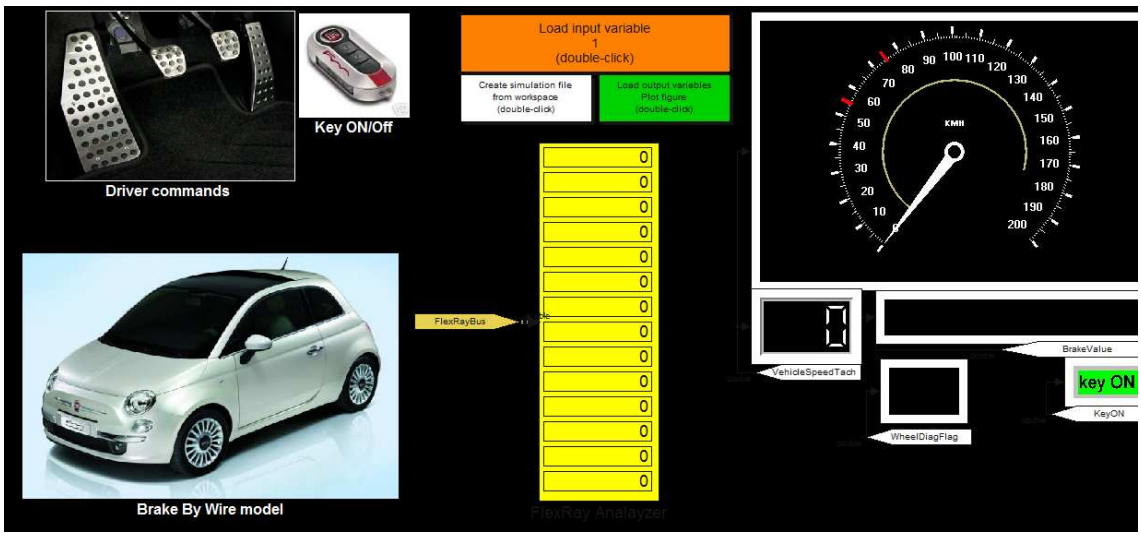

Figura 5.1: Interfaccia

cune variabili di non immediata interpretazione, per cui risulta opportuno definirle brevemente subito:

- Il *BrakeSignal* è la forza frenante, espressa in N, che la centralina *MainNode* decide sia necessaria per poter adeguatamente frenare. Questo valore lo invia attraverso il bus distribuito FlexRay alle quattro ruote.
- Il *ForceSetPoint* è il valore di forza, espresso in N, assume gli stessi valori del *BrakeSignal*.
- *ForceFB* è la forza di feedback, espressa in N, che la ruota manda alla centralina (o in generale al bus) che indica quale forza è stata realmente applicata al freno corrispondente. In questo modo la centralina *MainNode* può controllare se la ruota è riuscita a sviluppare la forza che gli era stata imposta dal nodo centrale, ossia il *MainNode*.
- Il *SpeedSetPoint* è la velocità, in km/h, che le ruote devono raggiungere (nella simulazione è imposto da tastiera). Da notare che questa è creata opportunamente per la simulazione e non è reale.
- *PedalPosition*, indica la posizione del pedale del freno ed è indicato in termini percentuali.
- *PedalPressure*, indica il valore di pressione del pedale del freno, in questo documento è stato assunto possa prendere uno tra tre valori, *Low*, *Med*, *High*.

- *AccPedalPosition*, indica il valore di accelerazione espressa in percentuale, che si viene applicato al sistema.

La schermata iniziale è realizzata in modo tale da poter inserire in ingresso i valori dei parametri necessari per la simulazione che andranno a caratterizzare i valori delle variabili che si analizzeranno. I segnali in questione sono il *PedalPosition*, il *SpeedSetPoint*, il *PedalPressure*, *AccPedalPosition*.

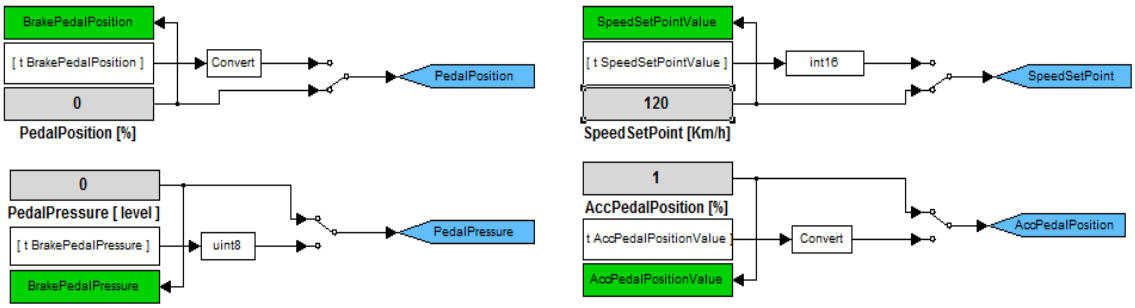

Figura 5.2: Segnali in input

Come si può notare dalla Figura 5.2, tutti questi segnali sono collegati a uno switch, che permette di associare ad ogni segnale una variabile di riferimento. In particolare al *PedalPosition* è associato il valore della variabile *BrakePedalPosition*, allo *SpeedSetPoint* è associato il valore della variabile *SpeedSetPointValue*, al *PedalPressure* il *BrakePedalPressure* ed infine al *AccPedalPosition* il valore caricato nella variabile *AccPedalPositionValue* in ogni istante di tempo.

Il tasto arancione, permette di caricare le tre (o più) simulazioni, in cui vengono considerati i possibili casi reali di moto del veicolo (con relative frenate), a cui corrisponde il tasto verde con cui sono visualizzabili i relativi grafici. Nella directory principale, a ognuno di questi tasti, sia arancione, verde e bianco è associato uno o più file.m. I file *.m*, che vengono "caricati" dal pulsante arancione, sono quelli relativi alla simulazione da eseguire, *my_opensimreq.m*, *Import_input_from_file.m*; nel caso del tasto verde, che permette la creazione del grafico, sono associati i file *Create_outputs_variables*, *createfigure.m*, *my_closereq.m*; per quanto riguarda il pulsante bianco, la cui pressione permette la creazione del file excel, sono associati i file *Create_input_variables.m*. Per far partire la simulazione è necessario effettuare dei passaggi intermedi: è necessario prima fare doppio clic sul pulsante arancione per caricare i dati, questo permetterà di far apparire tre finestra che illustrano il percorso da seguire per completare il caricamento. La prima finestra mostra le simulazioni che sono state caricate e permette di scegliere quella desiderata. La

seconda finestra, che si apre in automatico dopo aver dato confermato la scelta della simulazione, chiede se si vuole far partire quanto selezionato. Una volta aver confermato con *Yes*, il modello inizierà a caricare tutti i dati, e dopo aver confermato con *Ok* nell'ultima finestrella che appare, può iniziare il *running* premendo *start*. In questo modo le simulazioni sono già caricate, e nel caso se ne vogliano fare delle altre è sufficiente collegare lo switch ai segnali in ingresso, che la prima volta sono degli input da tastiera, far girare, switchare nuovamente e caricare i dati per essere poi simulati nuovamente. Il blocco giallo rappresenta il *bus FlexRay* che nel progetto è formato da 14 slot statici, in ognuno dei quali viene rappresentato il valore della variabile che varia al passare del tempo in base ai dati inseriti per la simulazione. Il numero di slot definito nel progetto, per definizione stessa di *segmento statico* del FlexRay, sono fisse. Cliccando sulla schermata iniziale del progetto apparirà la pagina mostrata in Figura 5.3:

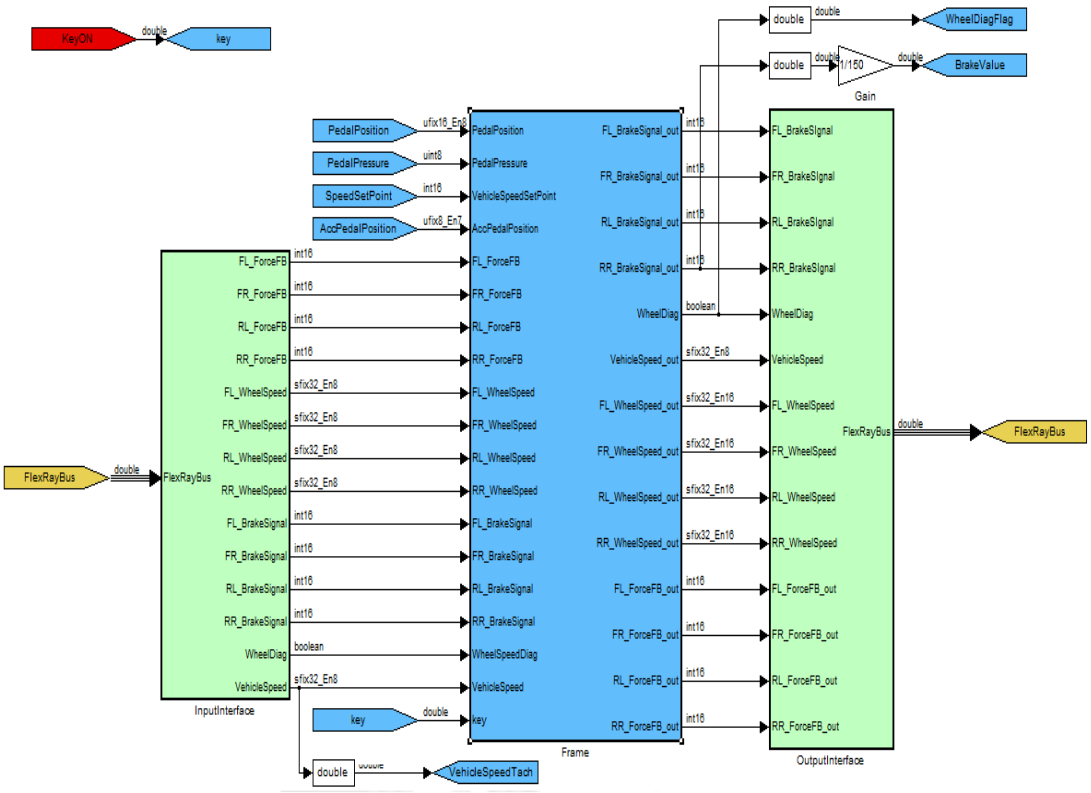

Figura 5.3: Segnali

Nei due blocchi verde, chiamati *InputInterface* e *OutputInterface*, che formano il bus distribuito, vi sono i 14 segnali in ingresso rappresentati nel bus e i rispettivi valori in output. In particolare si hanno in ingresso i seguenti segnali:

- *XX_ForceFB*, è la forza di feedback che la ruota manda alla centralina *MainNode*.
- *XX_WheelSpeed*, che è la velocità relativa ad ogni ruota.
- *XX_BrakeSignal*, è la forza frenante che la centralina *MainNode* decide che è necessaria per poter frenare.
- *WheelDiag*, che indica lo stato del sistema.
- *VehicleSpeed*, indica la velocità dell'autoveicolo.

In uscita, cioè i valori visualizzati nel bus nell'interfaccia, avremo:

- *XX_ForceFB*, è la forza di feedback che la ruota manda alla centralina *MainNode*.
- *XX_WheelSpeed*, che è la velocità relativa ad ogni ruota.
- *XX_BrakeSignal_out*, è la forza frenante che il nodo *MainNode* decide essere necessaria per la frenata.
- *WheelDiag*, che indica lo stato del sistema.
- *VehicleSpeed* indica la velocità dell'autoveicolo.

XX denota le diverse ruote e può assumere i valori:

- *FL*: *forward left* (anteriore sinistra).
- *FR*: *forward right* (anteriore destra).
- *RL*: *rear sinistra* (posteriore sinistra).
- *RR*: *rear destra* (posteriore destra).

Da notare che questi due blocchi, essendo il sistema distribuito, sarebbero potuti essere realizzati come un unico blocco collegato a quello blu di Figura 5.3. Cliccando sul blocco *InputInterface* apparirà la pagina mostrata in Figura 5.4, mentre in uscita verrà visualizzata l'immagine in Figura 5.5.

Come anticipato il sistema macchina è stata stata realizzato con un nodo principale più altri quattro nodi che rappresentano le ruote, come possibile vedere dalla Figura 5.6.

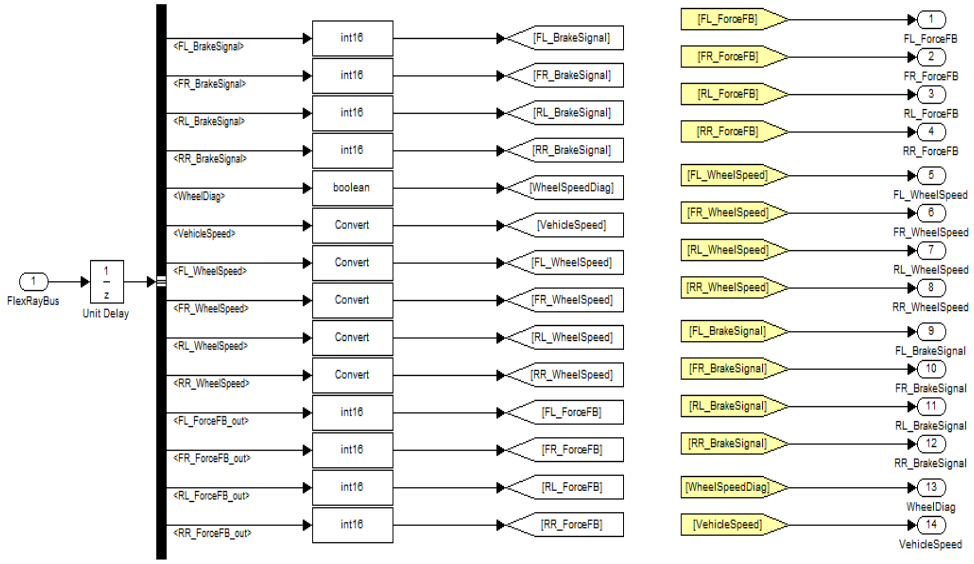

Figura 5.4: Segnali in ingresso nel bus

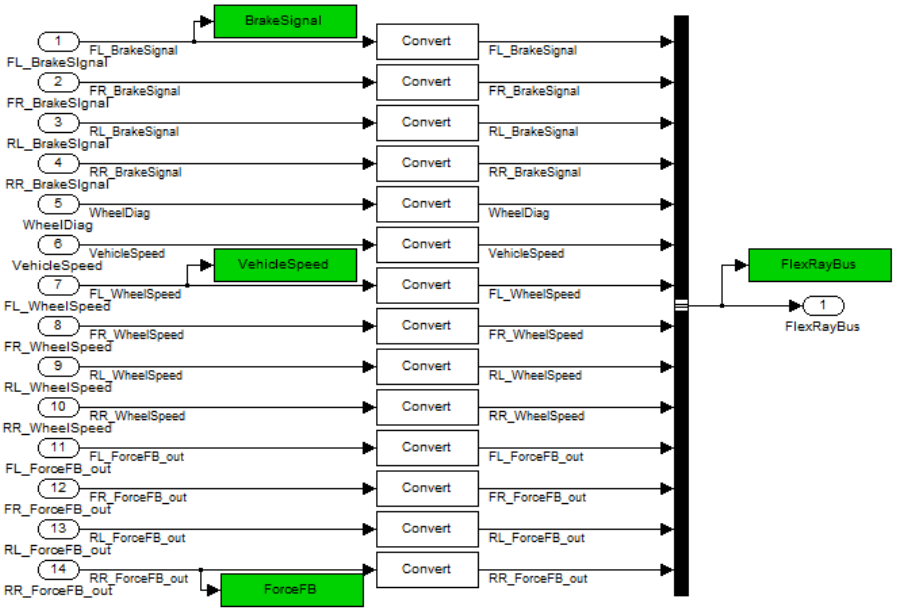

Figura 5.5: Segnali in uscita dal bus

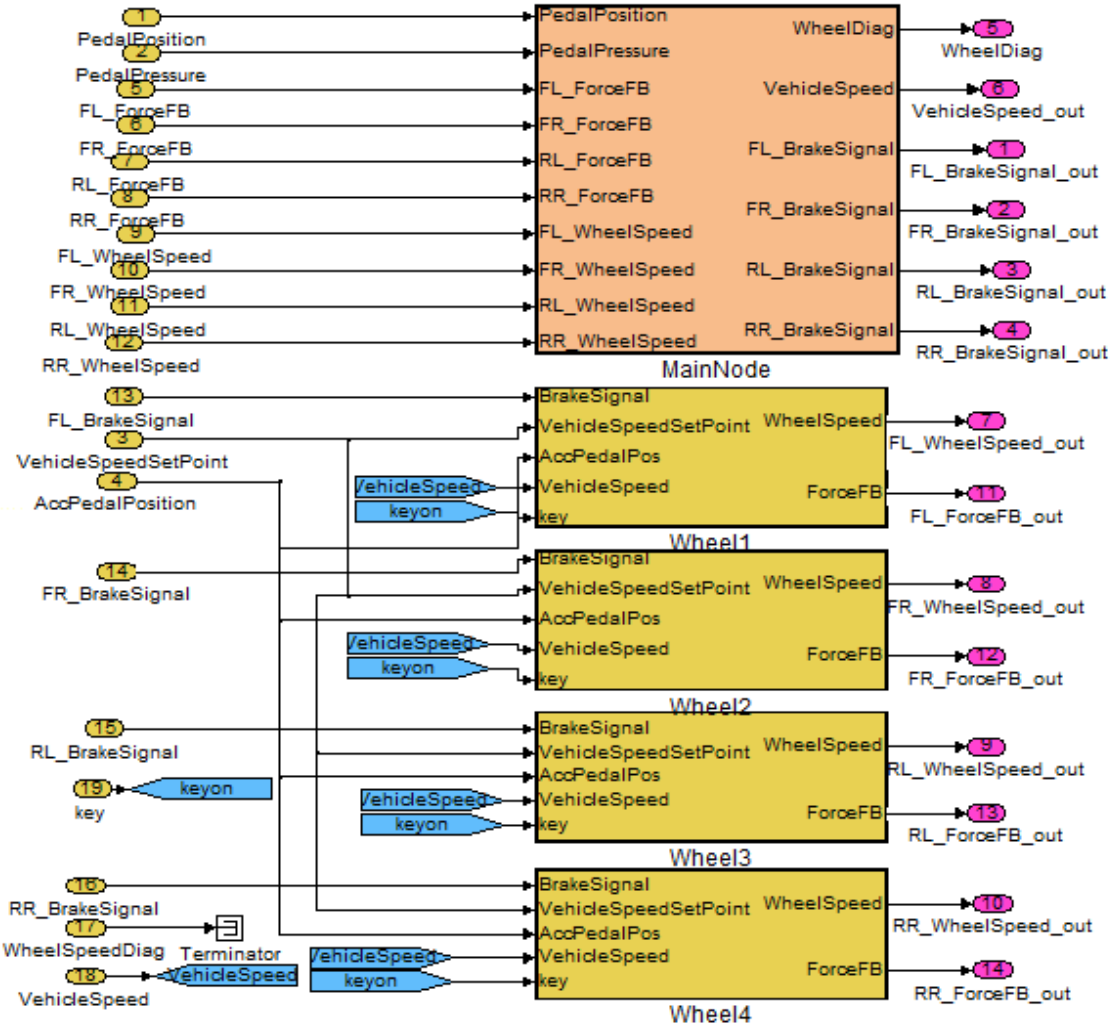

Figura 5.6: MainNode e WheelNode

5.3 MainNode

Questo modulo rappresenta la centralina di controllo del freno elettromeccanico (*Brake-by-Wire*). Esso rileva in ingresso dall'esterno la posizione del pedale freno (mediante vari switch di posizione situati all'interno della corsa pedale) che indica l'entità della forza che il guidatore vuole imprimere alla sua frenata. Oltre alla posizione viene rilevata la pressione sul freno: infatti una medesima corsa del pedale può rappresentare una frenata più o meno violenta a seconda che la pressione sia maggiore o minore. Queste due informazioni combinate permettono alla centralina di interpretare il tipo di frenata che si intende elargire e prendere le decisioni

corrispondenti.

La centralina quindi ricava e spedisce al sistema un valore di forza di frenata (*BrakeSignal*) per ciascuna delle quattro ruote dell'auto. Queste informazioni sono trasferite mediante FlexRay ai quattro nodi.

La centralina controlla lo stato delle ruote, soprattutto la loro velocità ed il feedback di forza applicata in modo da rilevare eventuali problemi e diversità rispetto al comando che gli è stato inviato. Se la centralina si accorge che le quattro ruote non stanno decelerando allo stesso modo oppure la forza applicata non è adeguata a quella richiesta prende decisioni per riportare il sistema frenante, e quindi la vettura, nella condizione che per lei è ideale (una sorta di controllo in retroazione).

Infine vi è una piccola implementazione di *Diagnosi*, in cui vengono controllati i flag di errore o i range delle variabili in modo da garantire la sicurezza del sistema. Per cui la centralina può essere suddivisa in tre macro funzioni:

- *BrakeManagment*.
- *BrakeControl*.
- *Diagnostic*.

I segnali in ingresso al blocco sono i seguenti:

- *PedalPosition*.
- *PedalPressure*.
- *XX_ForceFB*.
- *XX_WheelSpeed*.

I segnali in uscita da questo blocco e inviati nella rete sono:

- *WheelDiag*.
- *VehicleSpeed_out*.
- *XX_BrakeSignal_out*.

BrakeManagement

Il pedale del freno non è più collegato meccanicamente al sistema frenante ma è composto da un sistema di switch e sensori posizionati in modo da rilevare:

- *Posizione pedale freno* (corsa 0% - 100%).
- *Pressione sul pedale freno* (Low, Med, High).

Viene implementato un controllo di questi input semplice in modo da combinarli e determinare in uscita un valore di forza frenante desiderata. Il modello più semplice è ovviamente creare una corrispondenza 1 : 1 tra posizione freno e forza frenante ovvero la posizione 100% del freno (completamente premuto) corrisponde alla massima forza richiedibile dalla centralina e applicabile dal sistema frenante e così via in modo lineare.

L'interpretazione del valore di pressione non è molto semplice, in quanto l'algoritmo di controllo è piuttosto complicato. Non essendo questo lo scopo primario del progetto, verrà interpretato come segue: la pressione che si può applicare sul pedale del freno e ristretta ad un range da definire, viene diviso in step di pari ampiezza in modo da coprirlo interamente. Se la pressione misurata cade in un range avrà associato un certo aumento della forza frenante. I range definiti per tenere in considerazione una frenata caratterizzata *pressione pedale freno* sono:

- *Low*, a cui corrisponde il valore 0 di pressione, cioè la variabile *BrakePedalPressure* ha un valore compreso tra 0 e 33,3%.
- *Med*, a cui corrisponde il valore di pressione 1, cioè il valore della variabile *BrakePedalPressure* sta nel range 33,3 e 66,9% circa.
- *High* a cui corrisponde il valore 2 di pressione e si ha quando il valore della variabile *BrakePedalPressure* ha un valore in percentuale compreso tra 66,9 circa al 100%.

Tuttavia nelle simulazioni effettuate il valore della variabile *BrakePedalPressure* è stato sempre settato a zero. In questo modo la forza applicata puoò assumere un valore compreso tra 50 e $10050N$:

range forza = 50 - $10050N$. Il valore di forza associato alla frenata può essere definito come:

$$BrakeSignal = 100 * PedalPosition[\%] + 50 \tag{5.1}$$

dove $50N$ è l'offset che in un sistema reale è sempre presente, anche se il pedale non è premuto.

BrakeControl

Questa funzione serve per controllare che la frenata stia avvenendo regolarmente, che le quattro ruote stiano decelerando come desiderato e che la forza applicata dal sistema frenante di ogni singola ruota corrisponda alla forza che il *MainNode* ha scelto e inviato.

I controlli possono essere quindi i seguenti:

- Se la forza realmente applicata alla ruota si discosta oltre una certa soglia dalla forza necessaria calcolata; quest'ultima viene aumentata o diminuita a seconda dei casi di una quantità pari alla differenza tra queste:

$$F_{correttiva} = F_{ideale} - F_{reale} = BrakeSignal - ForceFB \tag{5.2}$$

- Se una ruota non decelera come le altre nonostante le forze siano identiche, viene aumentata la forza frenante su di essa di una quantità in grado di portarla alla velocità delle altre e ripristinare l'equilibrio. Si consideri la Figura 5.13, in cui sono mostrate le forze agenti su un veicolo in salita, in moto rettilineo e in assenza di vento laterale. il modello è considerato come un corpo rigido in moto traslatorio. Con le ipotesi fatte vengono considerati nulli gli effetti delle sospensioni (relativo al beccheggio della cassa) e si suppone che il comportamento della ruota destra e sinistra dello stesso asse siano uguali, trascurando così rotazioni di imbardata. La legge del moto per il veicolo preso in esame può essere scritta come segue:

$$M\ddot{x} = -F_{x_1} - F_{z_2} - F_{aer} - Mg\sin\alpha M - \sum_i f_{v_i} F_{z_i} \tag{5.3}$$

  dove si è indicato con F_{x_1} e F_{x_2} le forze frenanti applicate all'asse anteriore e posteriore, con F_{aer} la resistenza aerodinamica in cui ρ è $1,29\frac{kg}{m^2}$ ed è la densità dell'aria, V è la velocità del veicolo, S è uguale a $1.5m^2$ e rappresenta la sezione trasversale del veicolo, mentre C_x è un valore compreso tra $0,3$ e $0,4$ e rappresenta il coefficiente di penetrazione aerodinamico del veicolo. La sommatoria $\sum_i f_{v_i} F_{z_i}$ rappresenta la resistenza dovuta all'attrito volvente. Considerando l'attrito volvente uguale per tutte le ruote si può scrivere:

$$\sum_i f_{v_i} = f_{v_i} \sum_i F_{z_i} \tag{5.4}$$

  Inoltre esprimendo le forze di frenata per mezzo del coefficiente d'aderenza si ha:

$$F_{x_i} = \mu_{x_i} F_{z_i} \tag{5.5}$$

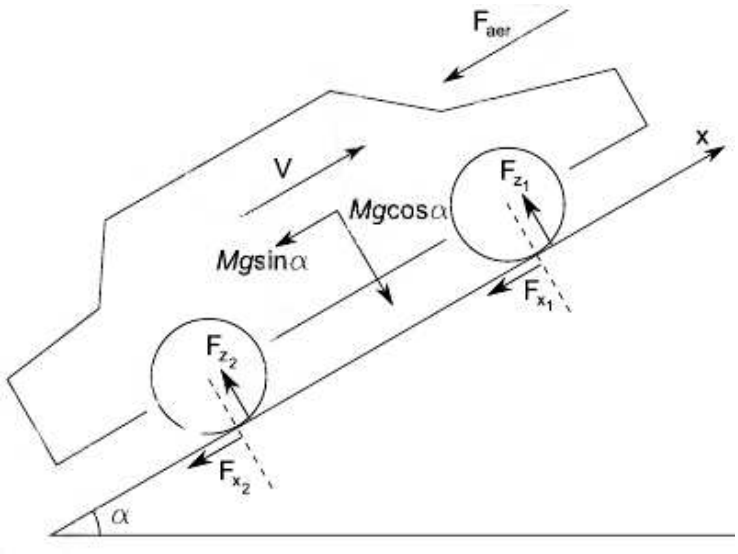

Figura 5.7: Forze agenti su un veicolo a causa della pendenza

La decelerazione può essere espressa come segue:

$$\ddot{x} = \frac{\sum_i \mu_{x_i} F_{z_i} - 0.5\rho_a V^2 S C_x - f_v \sum_i F_{z_i} - Mg\sin\alpha}{M} \tag{5.6}$$

dove:

- Il primo temine rappresenta la forza frenante dovuta al contributo dei freni.
- Il secondo termine è il contributo di frenata aerodinamica.
- Il terzo è il contributo dovuto all'attrito volvente.
- L'ultimo termine dipende dalla pendenza del veicolo.

Se $|\ddot{x}|$ è minore per una ruota, rispetto alle altre, è necessario sommare una forza correttiva $\ddot{x}_c$ tale che risulti:

$$|\ddot{x} + \ddot{x}_c| = decelerazioneideale \tag{5.7}$$

e inoltre

$$|\ddot{x}_c| = \frac{\mu F}{M} \tag{5.8}$$

perché si considera solo il contributo dei freni alla decelazione, da cui si ricava:

$$F_{correttiva} = \frac{|\ddot{x}_c|}{\mu} \tag{5.9}$$

Questa forza si sommerà con la forza che viene comunicata alla ruota. La ruota a questo punto dovrebbe reagire e decelerare in modo da inseguire il comportamento delle altre ruote.

È importante notare che, se la forza realmente applicata alla ruota è minore di quella comandata, l'effetto è quello di una decelerazione minore e quindi si avrebbe una correzione in cascata e quindi una correzione errata. Le due correzioni devono essere quindi contemporanee e praticamente si ha che:

- A decelerazioni diverse dovute a forze diverse si agisce solo sulla forza relativa al primo caso esposto in precedenza ($F_{correttiva}$), mentre se si hanno decelerazioni diverse ma forze uguali si agisce seconda la seconda correzione.
- Se si hanno delle decelerazioni uguali ma forze diverse non si effettua nessuna correzione, il sistema si autoregola dinamicamente.

Quindi ogni ruota ha in ingresso un segnale di forza dato dalla somma della *Brake-Signal* e della $F_{correttiva}$.

Diagnostic

Questa funzione serve a controllare il funzionamento del sistema frenante monitorando le variabili interessate e in caso di errori o di condizioni fuori dal normale prendere le adeguate decisioni per garantire la sicurezza dell'intero sistema.

Per il nostro sistema si implementa una semplice funzione di diagnosi che controlla lo stato delle variabili che viaggiano su FlexRay e che son utilizzate per il controllo applicativo.

Le variabili possono assumere oltre ai valori fisici previsti anche due valori di errore:

- 0xFE00 *Error*: quando la variabile è fuori range o assume un valore insensato.
- 0xFF00 *NotAvailable*: quando non è possibile stimarla a causa ad esempio di c.c. o c.a. Se una delle variabili assume uno di questi valori viene attivato un *flag di Alarm.*

5.3.1 Realizzazione MainNode

Verrà di seguito analizzato come è stata realizzata la centralina principale mostrata in Figura 5.8.

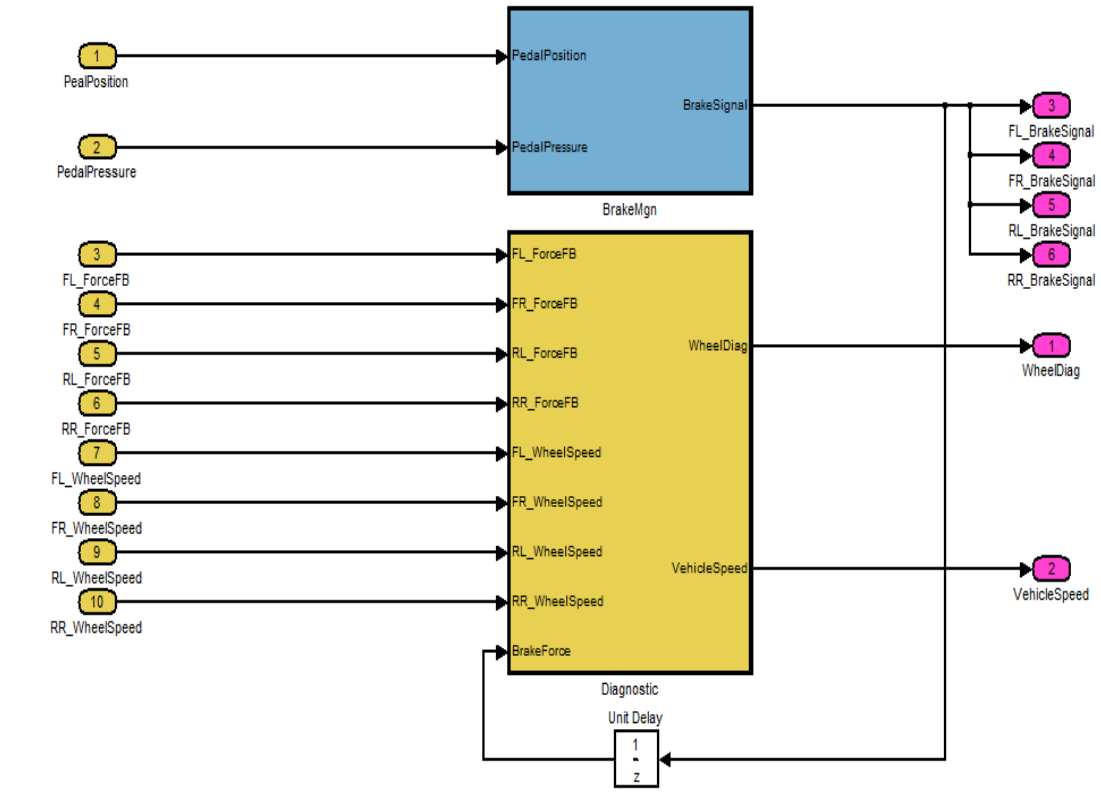

Figura 5.8: Main-Node

Come si può vedere il *MainNode* è costituito da due blocchi importanti, uno che tiene conto della diagnosi dei segnali, e l'altro da in uscita la forza da applicare alla frenata. Il primo di questi blocchi ha in ingresso le variabili *XX_ForceFB* e *XX_WheelSpeed* che vengono verificate per valutare se il sistema sta funzionando correttamente. In uscita il blocco avrà il segnale *WheelDiag* e *VehicleSpeed*. Il secondo blocco prende in ingresso il *PedalPosition* e il *PedalPressure* per restituire in uscita la forza da applicare alla frenata.

BrakeMgn

Con questo blocco il sistema calcola la forza da applicare nella frenata, dando in uscita il segnale *BrakeSignal*, come mostrato in Figura 5.9.

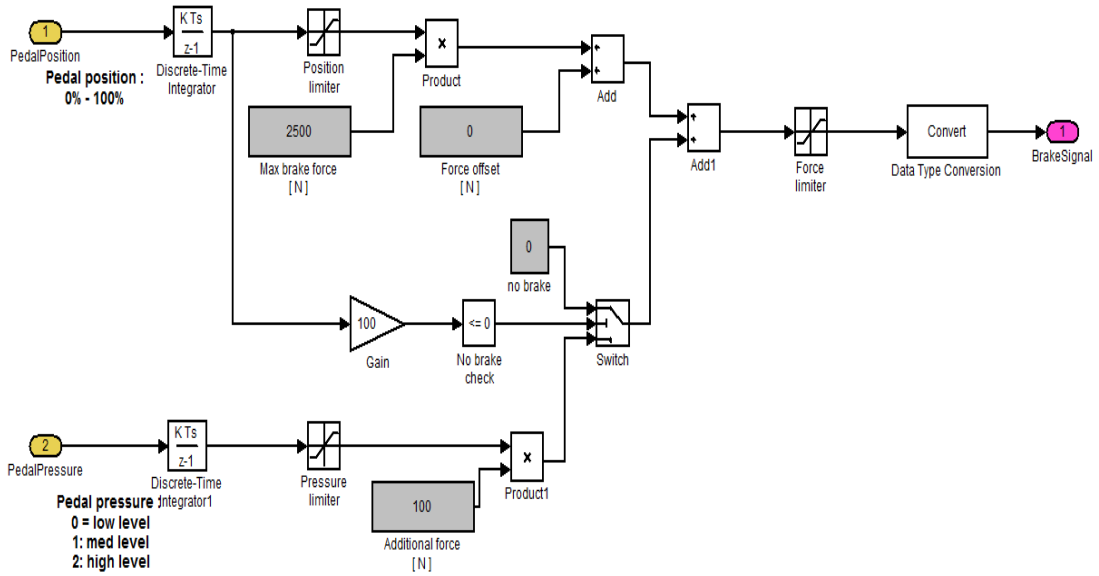

Figura 5.9: Segnale BrakeSignal

Si osservi che in ingresso al *Position limiter* e al *Pressure limiter* vi sia un filtro; questo è stato messo per fare in modo che la frenata simuli nel modo più reale il movimento del pedale del freno dell'automobile. In pratica si è descritto il suo movimento di pressione e di rilascio come quello di una molla. Il valore massimo di forza che può assumere il *PedalPosition* risulta essere $2500N$, valore ricavato dall'esperienza di aziende leader in questo settore. Il segnale totale, cioè il *BrakeSignal*, viene poi dato dato dalla somma del *PedalPosition* e del *BrakeSignal*.

Blocco Diagnostic

All'interno del nodo principale è presente anche il blocco *Diagnostic*, come mostrato in Figura 5.10.

In ingresso ci sono tutti i segnali che danno informazioni sulle ruote, in modo tale da poter valutare i dati, quindi fare confronti e decidere se è necessaria un'azione riparatrice in caso di *fault*, per evitare che il guasto crei danni a cosa o a persone. In particolare i segnali in ingresso sono:

- *XX_ForceFB*.
- *XX_WheelSpeed*.

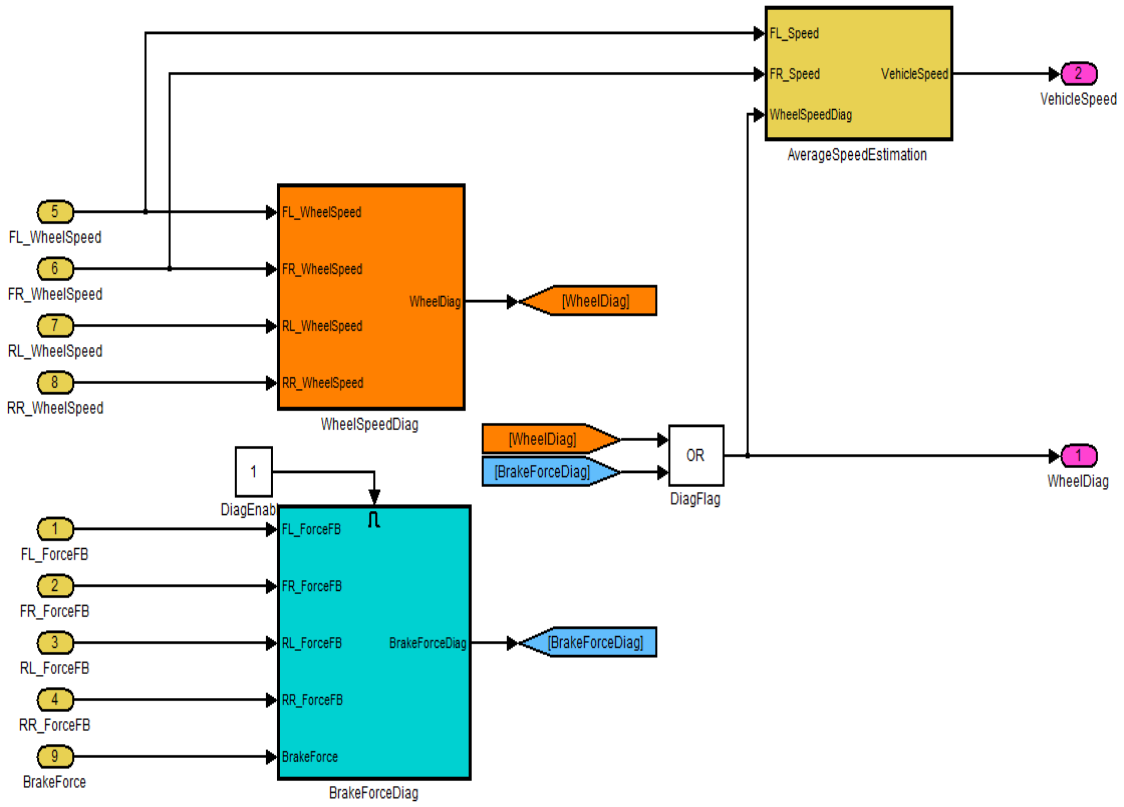

Figura 5.10: Sottoblocco di diagnosi

I segnali di uscita da questo blocco saranno i seguenti:

- *WheelDiag*.
- *WheelSpeed*.

Andando a livello inferiore si può apprezzare come la diagnosi sia stata effettuata in relazione alla velocità tenendo conto della velocità di ogni singola ruota, come mostrato in Figura 5.11. Il controllo che viene effettuato è mostrato dal diagramma a stati in Figura 5.12 che evidenzia come in caso di *WhellSpeedDiag* pari a 1 ci sia la reazione del sistema per controllare il *fault*, mentre in caso di *WhellSpeedDiag* pari a *0* tutto proceda normalmente.

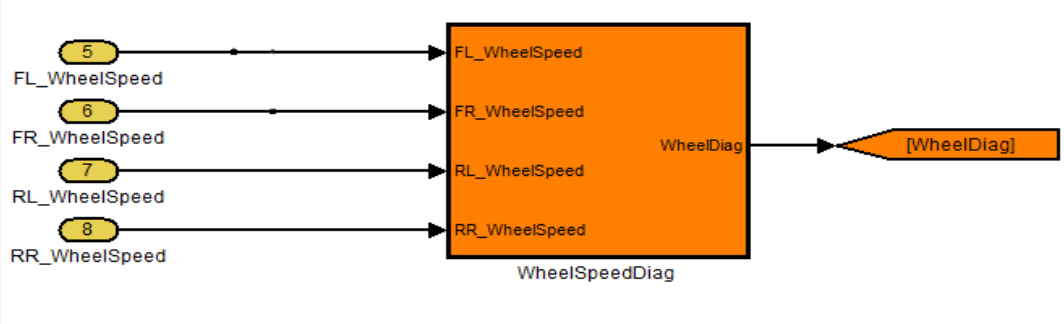

Figura 5.11: WheelSpeedDiag

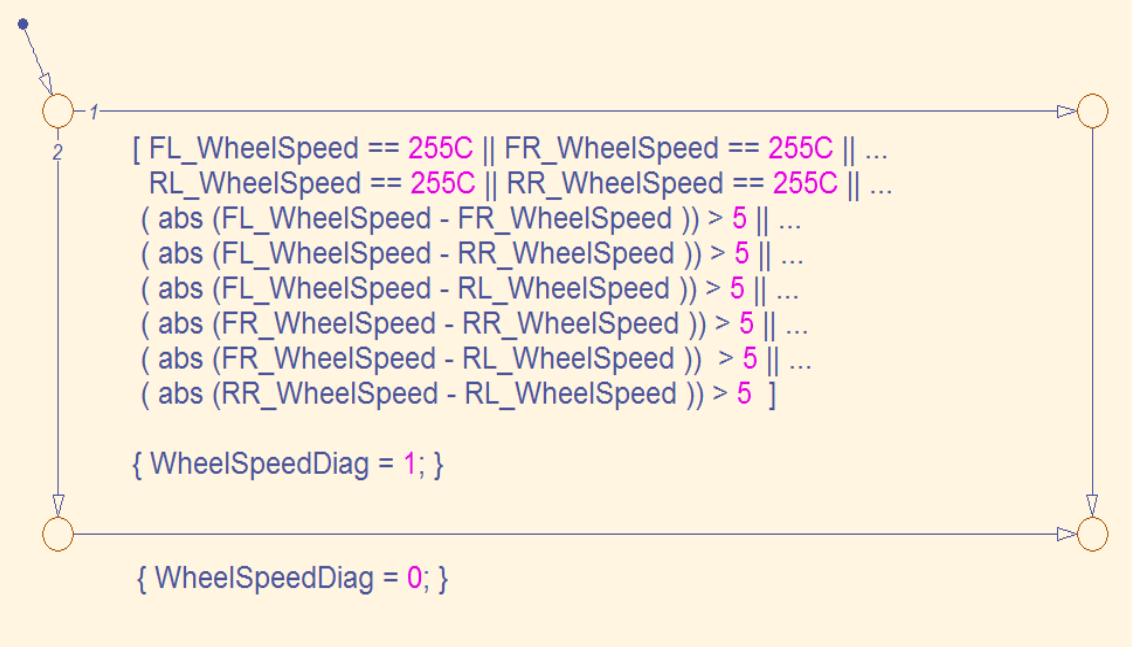

Figura 5.12: Diagramma a stati per la diagnosi della ruota

5.4 WheelNode

Rappresenta il sistema ruota compreso di sistema frenante (pinze disco e corpo motore brushless) e come tale deve essere capace di fornire in uscita le grandezze che sarebbero di interesse misurare da essa, come velocità ruota e i feedback sulla forza frenante (sensore di forza) e posizione della corsa del pistone sulla pinza (sensore di posizione). Riceve in ingresso il comando da parte della centralina Brake by Wire sulla forza che deve essere applicata (*BrakeSignal*).

Questa viene adeguatamente convertita in segnale elettrico *PWM* in modo da

attivare il motore *brushless*. Il motore permette la pressione del pistone sul corpo pinza disco in modo da frenare la ruota.

Si possono individuare i seguenti fenomeni e le seguenti funzioni all'interno del sistema ruota:

- La ruota riceve il *SetPoint* di forza che deve essere raggiunto per frenare/rallentare la ruota. Questo *SetPoint* viene raggiunto con un piccolo gradiente supposto per semplicità lineare. In realtà il gradiente è del tipo esponenziale perché la forza aumenta maggiormente quando la pastiglia tocca il disco e inizia a comprimere le pareti, cioè alla fine della corsa. Supponiamo però che la forza venga data con un piccolo gradiente lineare in cui il motore riesce a sviluppare $1000N$ al secondo. Quindi il tempo necessario, in secondi, per raggiungere il *SetPoint* è pari a:

$$T = \frac{ForceSetPoint}{1000} \tag{5.10}$$

  Si ha quindi che la F_c è data dalla seguente relazione:

$$F_c = max\{1000t, ForceSetPoint\} \tag{5.11}$$

- La forza frenante determina una decelerazione della ruota determinabile dall' equazione (5.6) ricavata in precedenza e qui riportata per comodita:

$$\ddot{x} = \frac{\sum_i \mu_{x_i} F_{z_i} - 0.5\rho_a V^2 S C_x - f_v \sum_i F_{z_i} - Mg\sin\alpha}{M}$$

  Come detto in precedenza, il primo temine della 5.6 rappresenta la forza frenante dovuta al contributo dei freni. Il secondo termine è il temine di frenata aerodinamica, mentre il terzo è il contributo dovuto all'attrito volvente, mentre l'ultimo dipende dalla pendenza del veicolo, come osservabile dalla Figura 5.13.

  Dall'equazione 5.6 si può determinare la velocità che la ruota possiede (teoricamente) in quell'istante:

$$V(t) = V_0 - \ddot{x}(t)T \tag{5.12}$$

  Questa è la velocità che verrà mandata sul FlexRay per poter essere letta dalla centralina del *MainNode* in modo da controllare la dinamica della frenata e reagire in caso di problema. Considerando i contributi alla frenata si ha:

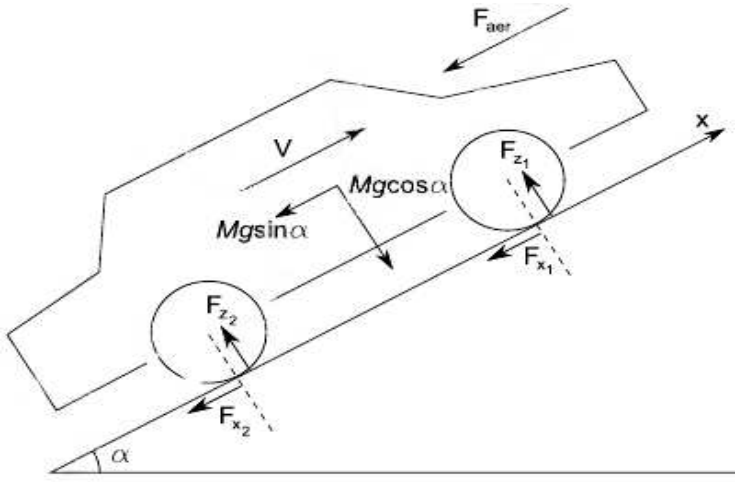

Figura 5.13: Forze agenti su un veicolo a causa della pendenza

- Contributo dovuto alla forza del sistema frenante viene determinato considerando che si ipotizza che il 90% dell'energia che il freno riesce a sviluppare sia effettivamente per la frenata sulla ruota, ossia che una parte (10%) sia dispersa e che il restante 90% sia utilizzata dalla ruota.

  La forza F_c è data:

  $$F_c = 0.9 * BrakeSignal \tag{5.13}$$

  La forza frenante sulla ruota lungo l'asse x è data dal prodotto dall'equazione5.13 per il coefficiente di aderenza longitudinale μ.

  Quindi:

  $$F_x = F_c\mu = \mu * \ 0.9 * BrakeSignal \tag{5.14}$$

- Il contributo dovuto alle forze aerodinamiche è dato dalla relazione seguente:

  $$F_{aerodinamica} = \rho V^2 S C_x \tag{5.15}$$

  in cui ρ è $1,29\frac{kg}{m^2}$ ed è la densità dell'aria, V è la velocità del veicolo, S è uguale a $1.5m^2$ e rappresenta la sezione trasversale del veicolo, mentre C_x è un valore compreso tra $0,3$ e $0,4$ e rappresenta il coefficiente di penetrazione aerodinamico del veicolo. Tutti i valori che caratterizzano il sistema e che verranno usati nelle simulazioni sono mostrati in seguito.

- Il contributo dovuto alle forze di attrito volvente vale:

  $$F_v = f_v F_c \tag{5.16}$$

in cui f_v varia tra $0,015$ e $0,035$ ed esprime il coefficiente di attrito volvente pneumatico-strada.

- Contributo dovuto alla pendenza stradale, pari a:

$$F_{pend} = Mg \sin \alpha \tag{5.17}$$

dove il valore M è la massa del veicolo e vale $1500kg$, g è l'accelerazione di gravità e α indica il valore della pendenza stradale.

- Il sistema ruota deve comprendere anche una componente di accelerazione positiva che deve essere simulata in modo da permettere al sistema di incrementare la velocità quando richiesto. Non essendo di particolare importanza implementare l'accelerazione esatta, si ipotizza di comunicare al sistema la velocità di crociera a cui si vuole andare e di è creata una funzione capace di far raggiungere al veicolo la velocità in questione gradualmente secondo un gradiente lineare. In pratica si è preso come punto di riferimento che un veicolo passi da 0 a $100km/h$ in 10 secondi.

5.4.1 Realizzazione WheelNode

Per quanto riguarda il nodo ruota, si avrà lo schema mostrato in Figura 5.14. Questa mostra in particolare la *Wheel1*, che altro non è se non la ruota anteriore sinistra.

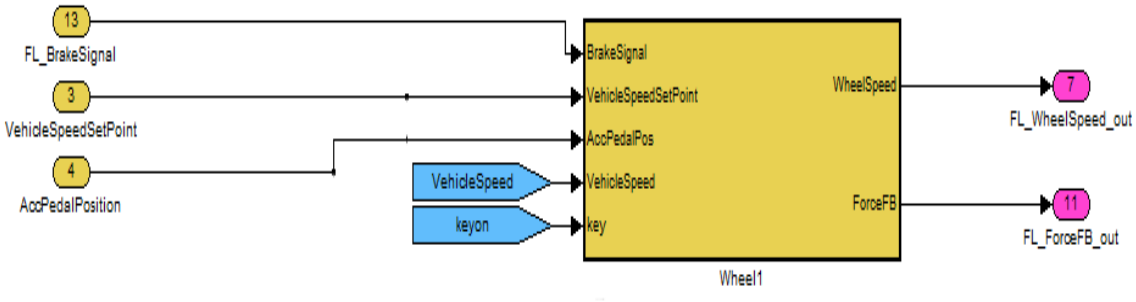

Figura 5.14: Wheel-Node

L'analisi che verrà effettuata vale anche per le altre ruote, cambiano solo i nomi delle variabili in quanto sono riferite al caso particolare.

In ingresso al blocco, considerando il caso della *Wheel1*, si hanno i seguenti segnali:

- *F_LBrakeSignal*.
- *VehicleSpeedSetPoint*.
- *AccPedalPosition*.
- Sono inoltre forniti, come *Tag*, i valori della *VehicleSpeed* e del *keyon*, che permettono di poter operare in caso di variazione del valore di queste due variabili.

Espandendo il blocco, come mostrato in Figura 5.15, è possibile vedere i sottoblocchi *BrakePedal*, *AccPedal*, *WheelSpeedEstimation* che elaborano i segnali per poi dare in uscita la *WheelSpeed* e la *ForceFB*.

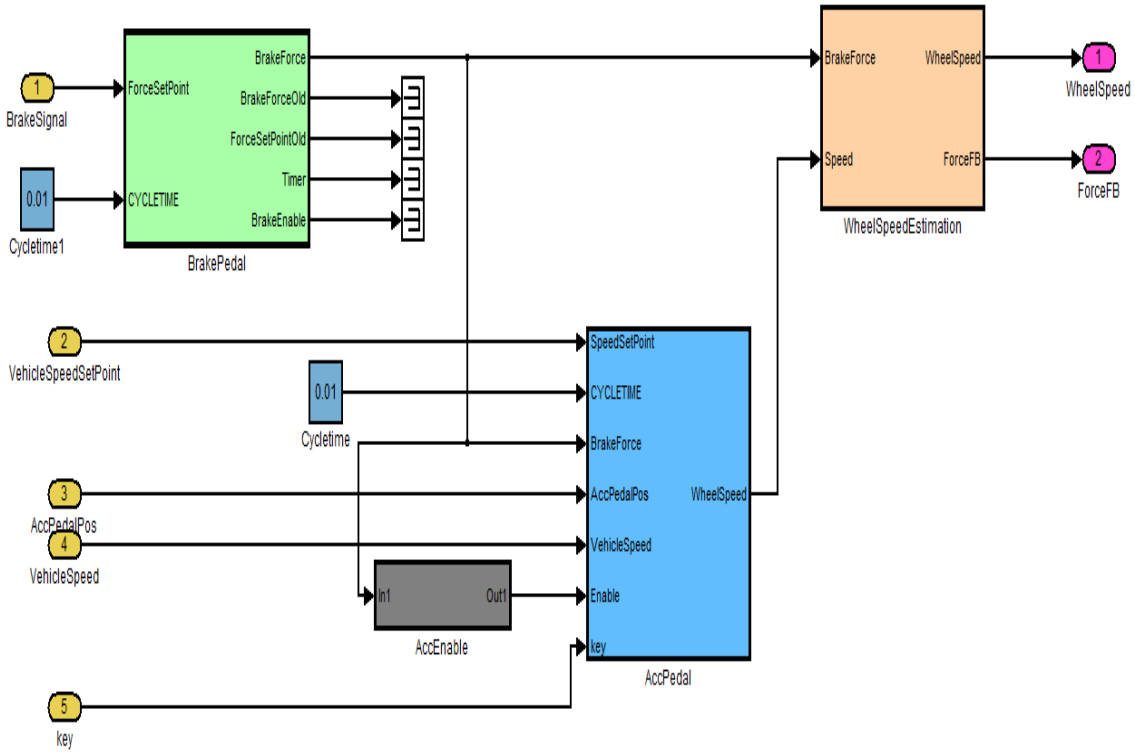

Figura 5.15: Sottoblocco Wheel-Node

BrakePedal

Questo sottoblocco prende in ingresso il *BrakeSignal* per poi dare in uscita il *BrakeForce* che sarà l'ingresso del sottoblocco *WheelSpeedEstimation*, come mostrato in

Figura 5.16. E' possibile vedere inoltre la presenza in ingresso della costante *Cycle-time1*, questo determina il tempo con cui verranno aggiornati i blocchi (esempio il *Timer*), infatti tutto il modello gira con un certo *StepTime*.

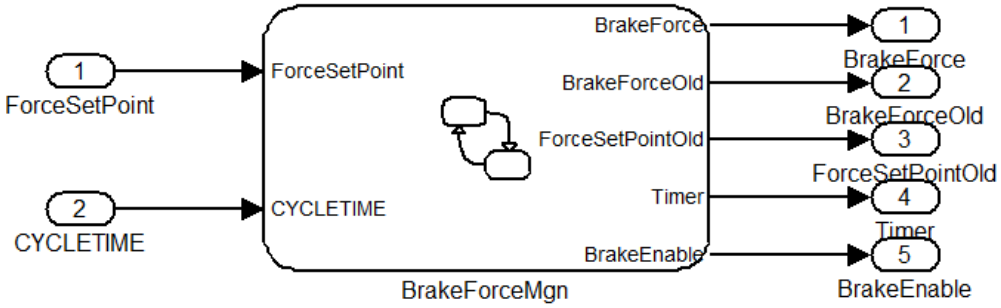

Figura 5.16: Sottoblocco BrakePedal

Il diagramma a stati che governa questo blocco è mostrato nella Figura 5.17. Questo in pratica simula il comportamento del pedale del freno. Ci sono due stati principali *Applying* e *Released* che sono appunto gli stati di pedale premuto e pedale rilasciato. La stato di pedale premuto simula la frenata graduale (stato *Timer*) sino ad arrivare al *SetPoint* richiesto (*SetPointON*). Nello stato *released* viene memorizzato il *SetPoint*. Questo diagramma a stati, come del resto gli altri, sono delle modelli approssimati per riprodurre il comportamento della ruota, cioè per simulare per esempio frenata, accelerazione.

AccPedal

Stesso discorso può essere fatto per quanto riguarda il sottoblocco *AccPedal*, dove in ingresso si sono i seguenti segnali:

- *SpeedSetPoint*.
- *CYCLETIME*.
- *BrakeForce*.
- *AccPedalPosition*.
- *VehicleSpeed*.
- *Enable*.

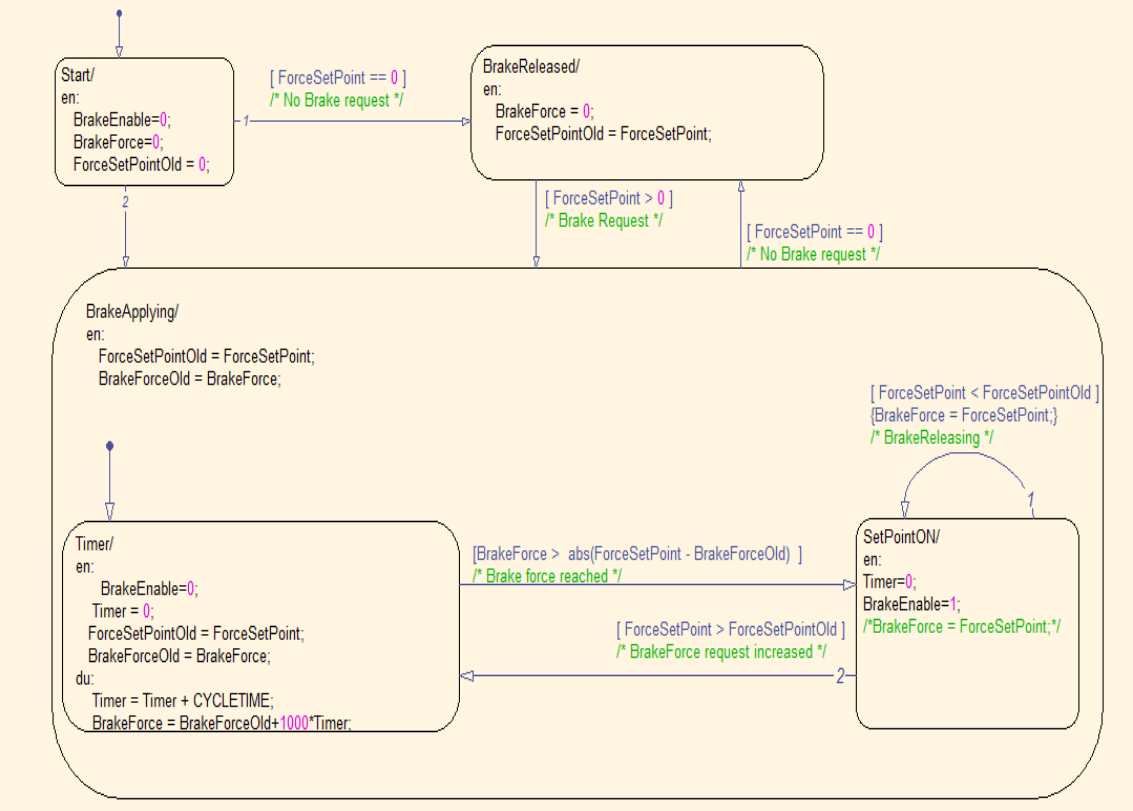

Figura 5.17: Diagramma a stati sottoblocco BrakePedal

- *Key*.

Questi segnali permettono di calcolare la *WheelSpeed*, valore che risulterà in ingresso al sottoblocco *WhellSpeedEstimation*.

Le Figure5.18 e 5.19 permettono di osservare come è stato realizzato il sottoblocco *AccPedal*.

Il diagramma a stati mostrato in Figura 5.19 in pratica simula il comportamento del pedale acceleratore. Ci sono due stati principali *Applying* e *Released* che sono appunto gli stati di pedale premuto e pedale rilasciato. La stato di pedale premuto simula l'accelerazione graduale (stato *TimeAcc*) sino ad arrivare al *SetPoint* richiesto (*SetPointON*). Nello stato *released* viene memorizzato il *SetPoint*. Lo stato simmetrico al *TimeAcc* è il *TimeDec* che gestisce e riproduce la decelerazione. La loro struttura è identica cambia il segno nell'equazione caratteristica. Questo diagramma a stati, come del resto gli altri, sono delle modelli approssimati per riprodurre il comportamento della ruota.

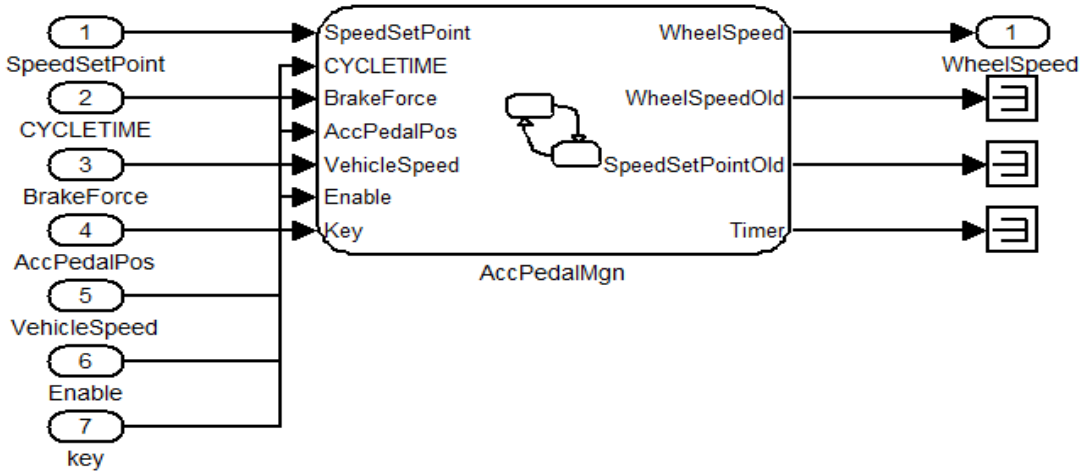

Figura 5.18: Sottoblocco AccPedal

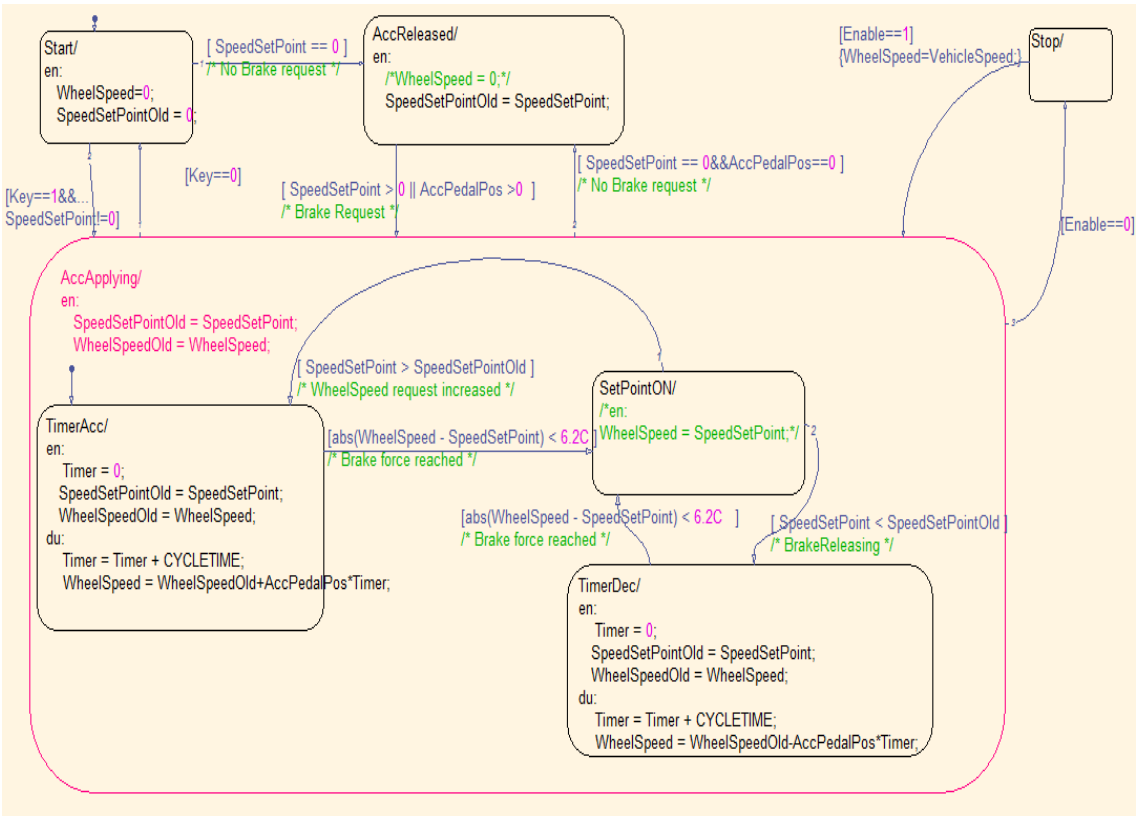

Figura 5.19: Diagramma a stati sottoblocco AccPedal

WheelSpeedEstimation

Infine il terzo sottoblocco da esaminare, che in ingresso prende le uscite *BrakePedal* e *AccPedal*, è il *WheelSpeedEstimation*. Come output questo restituisce il segnale

di velocità *WheelSpeed* e il segnale di forza *ForceFB*. Come si potrà notare dalle Figura 5.20 e in particolare dalla Figura 5.21 è possibile visualizzare i parametri caratteristici del sistema. Questi valori possono essere modificati in base al tipo di studio e di simulazioni che si desiderano effettuare. Risulterà molto semplice modificare quindi caratteristiche del veicolo, dell'asfalto, della frenata ecc. E' possibile quindi valutare il comportamento del sistema nelle diverse situazioni. Naturalmente la bontà di questi dati, insieme al resto della realizzazione del sistema, determinano quanto il sistema sia fedele alla realtà.

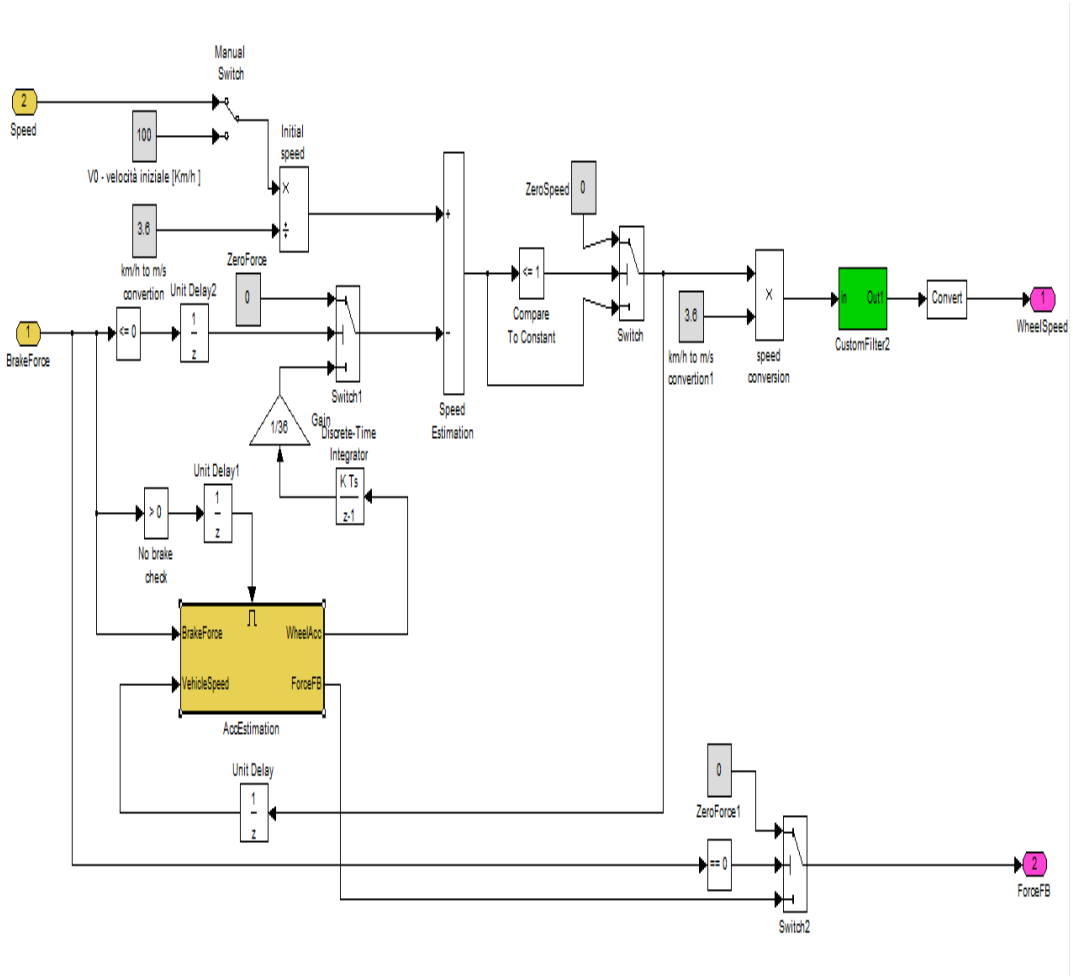

Figura 5.20: Sottoblocco WheelSpeedEstimation

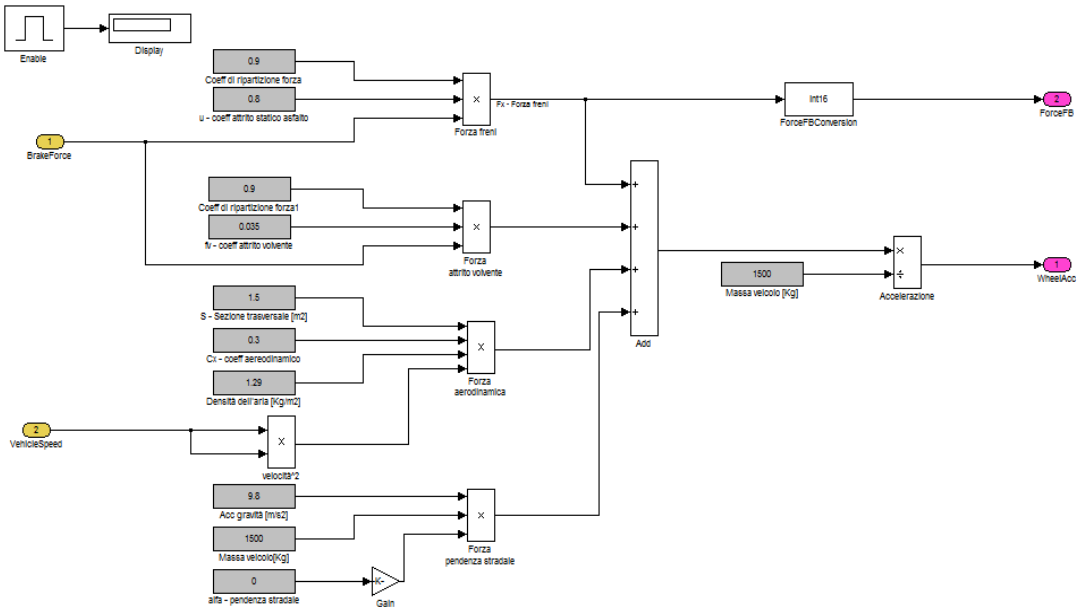

Figura 5.21: Diagramma a stati sottoblocco WheelSpeedEstimation

5.5 Simulazione

In questa sezione si riporta il risultato di diverse simulazioni che mostrano come risponde il modello in situazioni particolari. In questo lavoro sono state presentate tre delle tante prove fatte per verificare la bontà dei risultati ottenuti con il modello realizzato. Le simulazione che verranno mostrate vogliono rappresentare tre casi particolari, ma reali, che possono verificarsi durante la guida.

5.5.1 Prima simulazione

In questa simulazione si considera un veicolo che viaggia in una strada rettilinea, con un'accelerazione che può variare. Si può pensare che la simulazione rappresenti nella realtà la vista improvvisa, da parte del conducente, di un ostacolo che porta ad una brusca frenata. Le condizioni ambientali sono considerate standard, in particolare si assume:

- *Keyon* pari a 1, altrimenti il modello risulta spento e pertanto non sarebbe possibile effettuare nessuna operazione.
- *SpeedSetPoint* pari a $130 km/h$, che rappresenta il massimo valore di velocità teoricamente raggiungibile dal veicolo. In realtà il valore massimo di *SpeedSet-*

point non può mai essere raggiunto, come del resto nella realtà. Infatti si può pensare al *SpeedSetPoint* come il valore massimo di velocità presente nel contachilometri di un veicolo, e questo valore, salvo modifiche non è teoricamente raggiungibile.

- *AccPedalposition* pari a 1.
- *BrakePedalPosition* inizialmente nullo, poi per simulare la frenata verrà modificato.
- *BrakePedalPressure* lo fissiamo a zero, quindi abbiamo livello di pressione, mentre viene effettuata la frenata, basso.
- *Coefficiente di ripartizione forza* pari a 0.9, valore caratteristico.
- *Coefficiente d'attrito statico* normale, ovvero pari a 0.8. Questo significa che il suolo crea un ottimo contato con il pneumatico e quindi permette di avere ottime risposta in frenata.
- *Coefficiente di ripartizione forza1* pari a 0.9 (valore ideale).
- *coefficiente d'attrito volvente* pari a $0,035$. Il prodotto di questo valore con il *Coefficiente di ripartizione forza1* genera la forza esercitata dall'attrito volvente.
- *Forza aerodinamica* pari al prodotto fra la *Sezione trasversale* chè pari $1,5m^2$, la *Coefficiente aerodinamico* posto pari a $0,3$ e la *Densità dell'aria* considerata pari a $1{,}29kg/m^2$.
- *Massa del veicolo* considerata pari a $1500kg$.

Tutti questi parametri, naturalmente sono visualizzabili e modificabili dal progetto e adattabili al tipo di situazione che si vuole rappresentare. In questa simulazione, viene messo in evidenza come, con l'applicazione della forza frenante il veicolo si fermi in un certo range di tempo e come vengano rilevati, aggiornati e visualizzati i dati nel bus distribuito. Per far partire il modello è necessario caricare la variabile tempo nel sistema premendo il tasto arancione (doppio clic); il secondo passo da fare è quello di impostare i valori di input alle variabili che determinano velocità, accelerazione, frenata.

Arrivati a questo punto risulta interessante modificare il valore del pedale del freno, applicando un'energica frenata. Teoricamente il valore massimo di frenata, dovrebbe riuscire ad imprimere una forza pari a $2500N$, in realtà (come si vedrà dal

grafico della Figura 5.22) quella realmente applicata è minore. Risulta molto utile visionare il grafico delle variabili per poter osservare come variano nel tempo e per vedere come sono tra loro collegate. Altro particolare da tenere in considerazione è la differenza tra forza reale applicata, mostrata, in Figura 5.22, dalle curve tratteggiate di colore rosso e mentre quella realmente applicata dalle curve tratteggiate di colore verde. Per verificare i dati relativi al grafico è possibile i particolari della simulazione realizzata cliccando due volte sul pulsante bianco, in questo modo viene creato il file excel che documenta, istante per istante, quanto è stato processato nella simulazione.

La visualizzazione del grafico avviene con un doppio clic sul pulsante verde dell'interfaccia. L'andamento dei segnali al variare del tempo può essere osservato dal grafico mostrato in Figura 5.22.

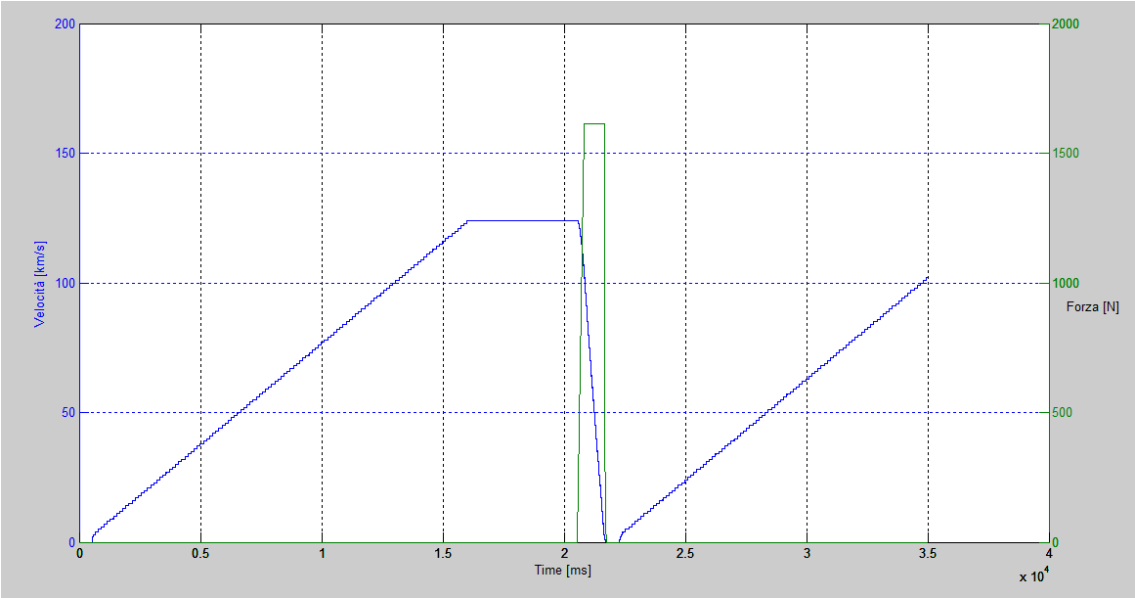

Figura 5.22: Prima simulazione, andamento dei segnali al variare del tempo

Dalla Figura 5.24 è possibile apprezzare meglio l'andamento di ogni singola variabile in funzione del tempo, in particolare sono qui graficate, con doppia scala per avere una mappatura migliore la *VehicleSpeedValue* con la *FR_ForceFB* e la *FR_ForceFB*. La Figura 5.23 mostra, in doppia scala, la relazione tra la velocità, *FL_Speed* e *FR_Speed*, e le variabili *FL_BrakeSignal* e *FR_BrakeSignal*. Questi grafici sono stati separati perché i quattro valori di forza hanno una scala diversa, quindi quanto mostrato sarebbe risultato meno chiaro. Inoltre si consideri che in questo caso i valori di velocità totale del veicolo, la *VehicleSpeedValue*, e delle singole ruote *FL_Speed* e *FR_Speed*, sono identici quindi in realtà i due grafici sono, sotto questo punto di vista, speculari. I segnali graficati sono quindi:

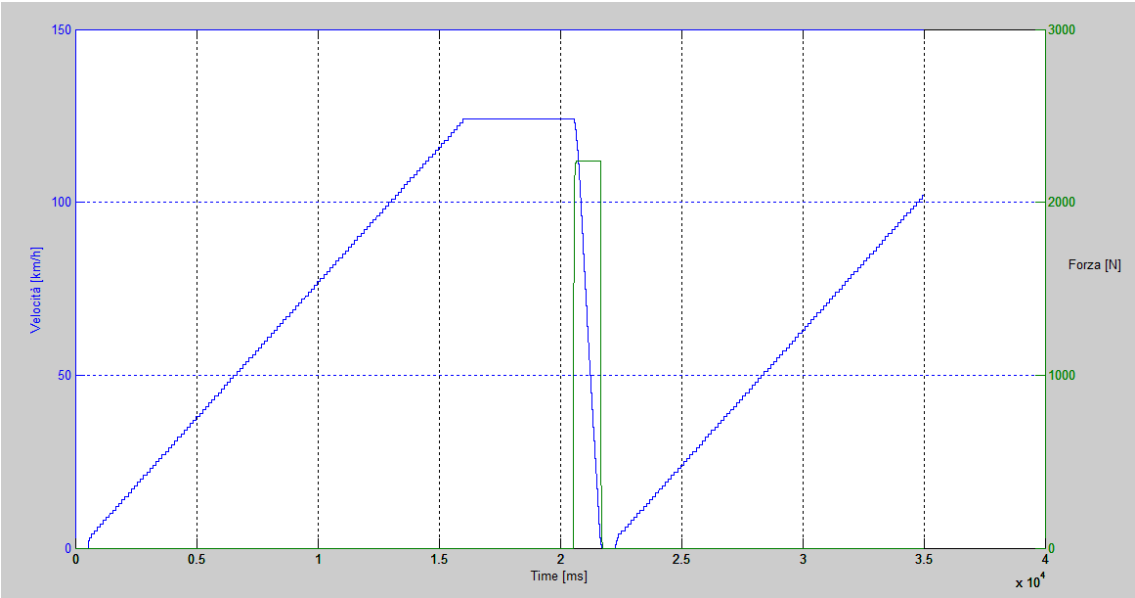

Figura 5.23: Prima simulazione, andamento dei segnali al variare del tempo

- *VehicleSpeedValue*, che rappresenta il valore di velocità acquisito dal *MainNode* durante l'intervallo di simulazione. Come si può notare il veicolo quando arriva a $100Km/h$ subisce una frenata che lo porta a fermarsi. La retta lineare con cui la velocità cresce mostra che il veicolo raggiunge, con una certa accelerazione, i $124km/h$. Questo valore rappresenta il massimo raggiungibile dal veicolo, in quanto anche nella realtà generalmente un veicolo non raggiunge il massimo del valore a cui potrebbe teoricamente arrivare.

- *FL_Force*, è il segnale di feedback che la ruota manda alla centralina (o in generale sul bus) che indica quale forza è stata realmente applicata al freno relativo alla ruota FL. Come si può notare il valore è pari a $1600N$.

- *FR_Force*, rappresenta il segnale di feedback che la ruota manda alla centralina, o al bus, in questo caso relativo alla ruota FR.

- Il *SpeedSetPointValue*, come si può notare indirettamente dai grafici della Figura 5.24, rappresenta il valore di input inserito tramite tastiera che rappresenta il valore massimo teorico che può raggiungere il *VehicleSpeedValue*. Infatti la velocità non supera mai i $124km/h$, perchè in questo caso gli è stato imposto il valore di $130km/h$.

- *BrakePedalPosition* è posto pari a zero, quindi all'interno del range 0, ossia livello *Low*.

- *FL_BrakeSignal* è appunto la forza frenante che il *MainNode* decide sia necessaria da applicare alla ruota per poter frenare. Il valore determinato viene inviato al nodo FL dove in realtà la forza realmente applicata è la *FL_ForceFB*.

- Ragionamento identico al precedente vale per la *FR_ForceSignal*, l'unica differenza è che in questo caso si sta parlando della ruota FR.

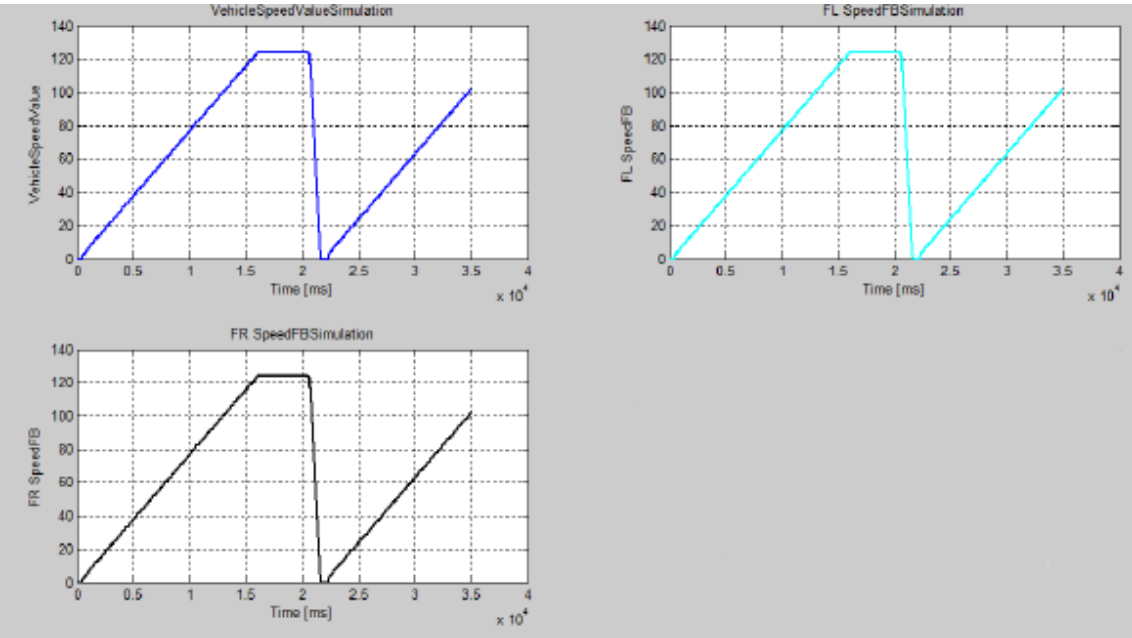

Figura 5.24: Prima simulazione, andamento nel tempo dei segnali di velocità

Dalla Figura 5.24 e la Figura 5.26 si osserva che:

- All'istante dell'applicazione della frenata ogni ruota, come del resto ci si aspettava, perde velocità fino ad arrivare a 0. Non appena si arriva alla minima velocità, si è cioè fermi e viene rilasciato il pedale del freno, il veicolo a cui viene impressa un'accelerazione pari a 1, cioè 100%, riprende la marcia.

- A circa $20s$ dall'inizio della simulazione, si osserva che il valore della variabile *VehicleSpeedValue* resta costante a $124km/h$. Questo risultato, per altro atteso, è dovuto al fatto che il segnale *SpeedSetPointValue* è stato impostato a $130km/h$.

- Importante è da osservare inoltre che la *FL_ForceFB* e la *FR_ForceFB* sono identiche, come giusto che sia se il sistema non subisce *fault*.

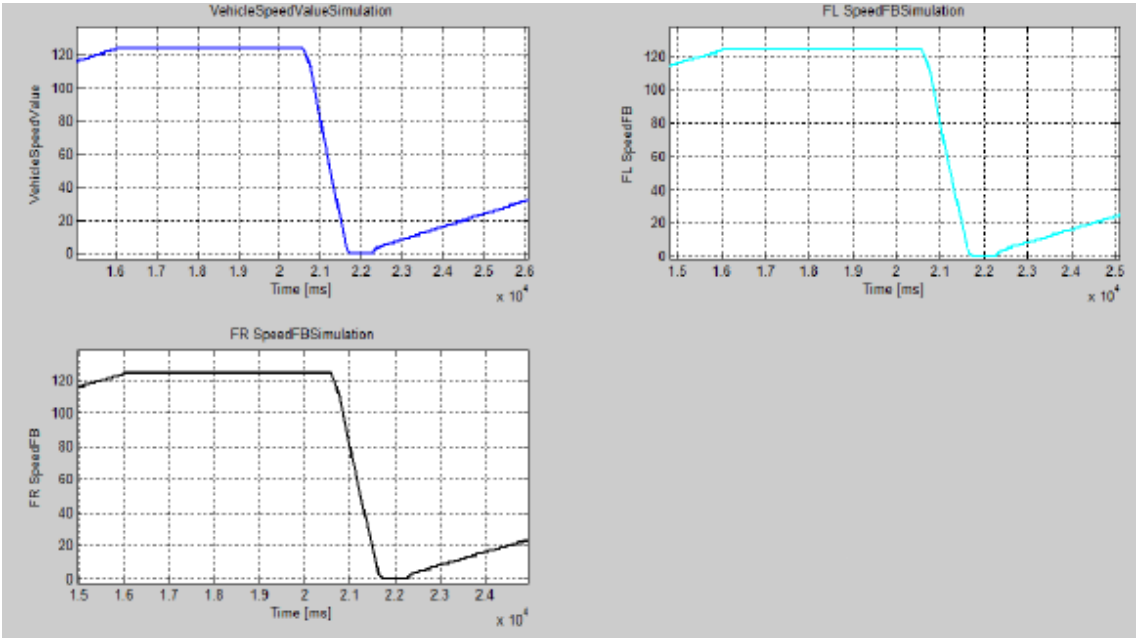

Figura 5.25: Prima simulazione, zoom andamento nel tempo dei segnali di velocità

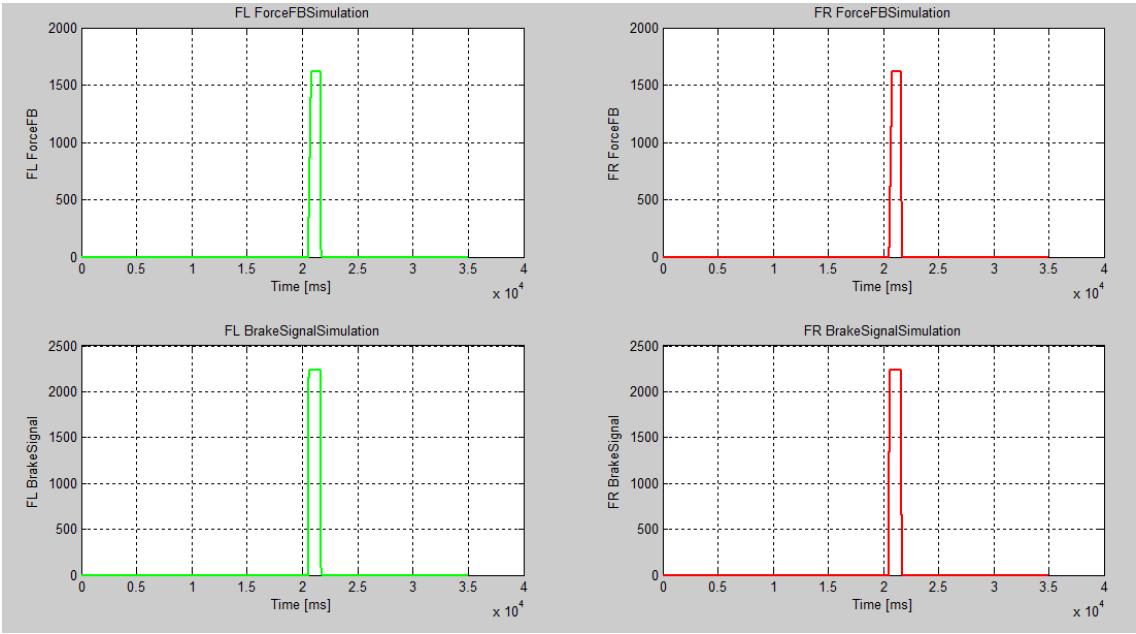

Figura 5.26: Prima simulazione, andamento nel tempo dei segnali relativi alla frenata

- Se si osserva il valore della variabile *BrakeSignal*, sia sulla ruota anteriore che posteriore, si può notare che questo sia uguale per entrambe le ruote, risultato che ci si aspettava. Inoltre si nota che al valore del *BrakeSignal*,

nelle rispettive ruote, corrisponde un valore di forza applicata realmente sulle ganasce inferiore. Questo, come previsto, è dovuto a dei coefficienti inserite nella simulazione per rendere reale la frenata, in particolare giocano un ruolo importante il coefficiente di ripartizione della forza e il coefficiente d'attrito.

5.5.2 Seconda simulazione

A differenza della precedente simulazione qui vengono applicate due frenate delle quali la prima non risulta eccessivamente forte, mentre la seconda applica un valore di forza molto alto. Questa simulazione potrebbe bene rappresentare il moto di un veicolo in una strada a scorrimento veloce, in cui la velocità del mezzo è tendenzialmente alta e la riduzione della velocità può essere causata, per esempio, dal rallentamento dovuto a macchine che procedono più lentamente, oppure ad operazioni di sorpasso che non possono avvenire nel momento desiderato e così via. Sostanzialmente questo test viene fatto per mostrare come cresce la velocità, come può variare e apprezzare come le informazioni viaggiano nel bus distribuito FlexRay. Nella Figura 5.27 è mostrato come cambiano le diverse variabili in funzione del tempo. Come per l'esempio precedente è possibile verificare ed osservare dettagliatamente i valori relativi ad ogni variabile, e il relativo cambiamento nel tempo, aprendo file excel creato effettuando un doppio clic il pulsante bianco posto all'interfaccia. Questa seconda simulazione può essere richiamata ogni qualvolta si desidera con un doppio clic su *PROVA2* attraverso il tasto arancione.

I grafici mostrati nella Figura 5.27 e 5.28 mostrano la relazione tra i diversi segnali e come uno ognuno di essi varia in relazione all'altro. Si è scelto di mettere le due immagini separate per facilitare la lettura dei grafici. Inoltre si può notare che, nella Figura 5.27, si può osservare una sola velocità e un solo valore di forza. In realtà, come si vedrà meglio nella Figura 5.29 le velocità, la *VehicleSpeedValue*, *FL_Speed* e la FR_Speed, come del resto ci si aspettava, sono identiche. Quindi il grafico della Figura 5.27 e quello della Figura 5.28 mostrano la relazione fra le velocità e i valori di frenata, sia quello realmente applicato, cioè le *FL_ForceFB* *FR_ForceFB*, sia quello dettato dal *MainNode*, *FL_BrakeSignal* e *FR_BrakeSignal*. In Figura 5.29 sono mostrate tutte le variabili considerate nel grafico precedente ma ognuna su un grafico diverso. Analizzando in dettaglio la Figura 5.29 è importante rilevare quanto segue:

- Per quanto riguarda la variabile *VehicleSpeedValue* è percettibile dal grafico ad essa associato che vengono effettuate due frenate seguite da altrettante accelerazioni. Infatti la *VehicleSpeedValue* cresce linearmente, fino ad arrivare

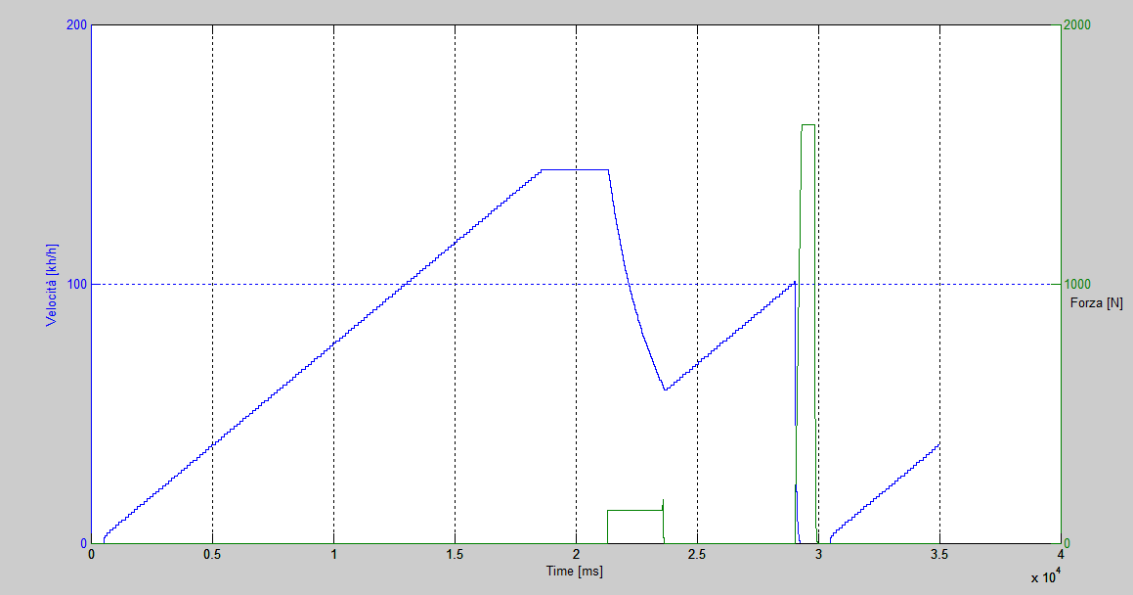

Figura 5.27: Seconda simulazione: grafico andamento della VehicleSpeedValue e FL_ForceFB

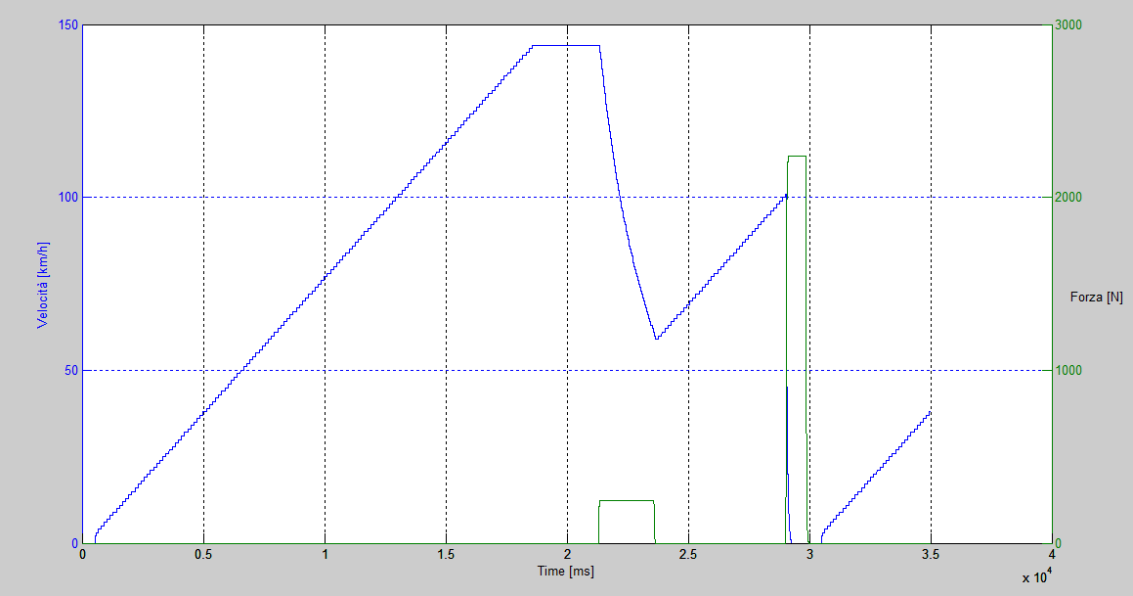

Figura 5.28: Seconda simulazione: grafico andamento delle variabili FL_Speed e FL_BrakeSignal

al suo valore massimo. Quando arriva al valore massimo resta costante, in quanto si è imposto che la velocità massima fosse leggermente inferiore al valore teoricamente raggiungibile. Dopo un certo intervallo di tempo t, il

VehicleSpeedValue subisce una diminuzione del valore relativamente lento (si può pensare ad un rallentamento causato da un veicolo che procede con velocità inferiore). La velocità arriva al valore di $55km/h$ e riprende a crescere. Questo significa che il valore della variabile relativa alla posizione del pedale è cambiato passando a zero, permettendo il riavvio dell'accelerata. Al raggiungimento dei $100km/h$, come previsto, il veicolo perde bruscamente velocità, fino a fermarsi per poi ripartire a seguito del rilascio del pedale del freno.

- Il segnale *FL_ForceFB* evidenzia quanto detto al punto precedente, poiché è possibile osservare che sulla ruota FL vengono effettuate due frenate, una che applica una forza relativamente piccola e l'altra invece che imprime una forza pari a $1600N$.

- Discorso analogo per il segnale *FR_ForceFB* applicato alla ruota FR. Da notare con particolare attenzione che le forze applicate alla ruota FL e FR sono identiche. Questo, che è quanto ci si aspettava, è molto importante perché permette di capire che sicuramente il sistema non ha subito un *fault* dovuto ad un guasto nella distribuzione della forza frenante.

- I segnali *FL_BrakeSignal* e *FR_BrakeSignal* rappresentano rispettivamente i valori di forza che il *MainNode* ha deciso di applicare alla ruota FL e FR. Anche in questo caso è importante osservare che la forza da applicare ai due diversi nodi è uguale, segno che a livello di nodo centrale la ripartizione della forza per la frenata non ha subito *fault*, ma funziona bene. Di rilievo inoltre il fatto che ad un valore x di forza determinato dal *MainNode* da applicare alle ruote, corrisponde un valore pari a $x - \nabla$. Questo è dovuto al fatto che nella realizzazione del modello si tenuto conto del coefficiente d'attrito volvente, del coefficiente di ripartizione della forza e di altri valori che causano la riduzione della forza da applicare alle ruote.

- La variabile *SpeedSetPointValue*, come mostrato nel grafico della Figura 5.29, risulta costante e pari a $150km/h$, come impostato da tastiera. Il grafico della Figura 5.27 evidenzia che questo valore di velocità non è stato mai raggiunto in quanto è stato imposto che il veicolo non potesse raggiungere mai il valore massimo teorico.

- Il valore del segnale *BrakePedalPressure*, non è stato grafico, ma risulta in questa simulazione costante e pari a zero. Se si volesse rendere il sistema sensibile anche al tipo di pressione effettuata nella frenata basterebbe modificare questo parametro e verificare che la forza applicata, nelle stesse condizioni, risulta diversa e in particolare maggiore.

- Il valore assunto dalla variabile *BrakePedalPosition* rappresenta che tipologia di frenata è stata effettuata. Pur non essendo stata graficata, è facile intuire che la prima frenata è molto più dolce rispetto alla seconda che risulterà molto brusca.

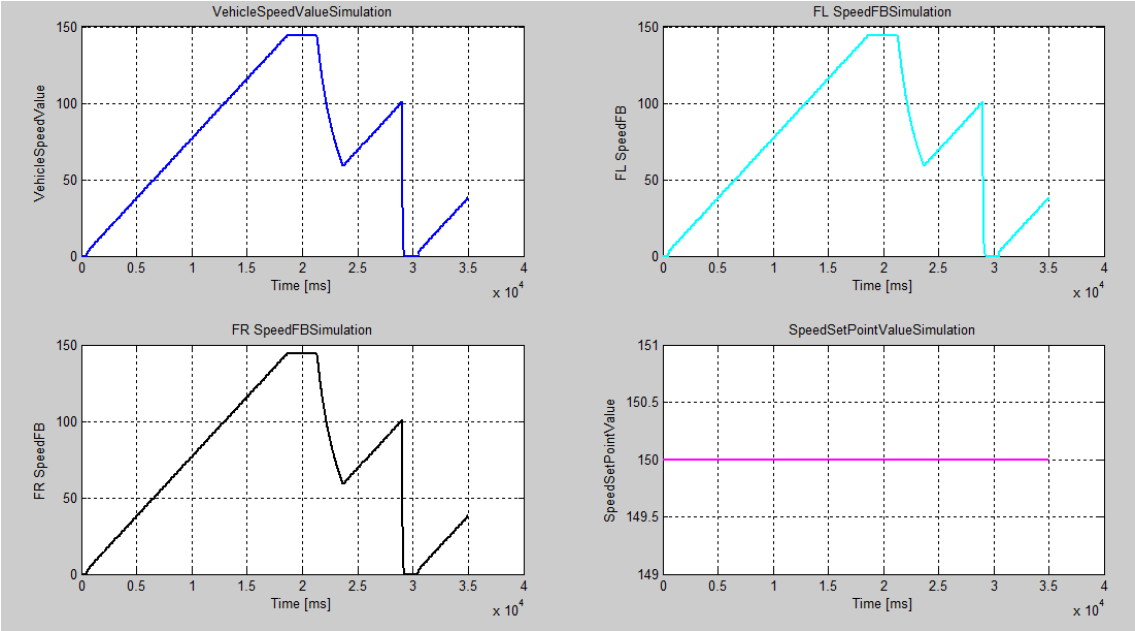

Figura 5.29: Seconda simulazione, andamento nel tempo delle variabili relative alla velocità

5.5.3 Simulazione con fault

Con l'ultima simulazione viene trattato il caso in cui il sistema ha un *fault* su una ruota e in particolare come reagisce a questo evento il modello. Il progetto, che da questo punto di vista deve essere migliorato per riuscire a garantire la massima sicurezza, va, in questo caso, in *fault* quando la differenza di velocità tra le ruote assume un valore superiore al range imposto. Questo range è definito per evitare che il sistema assuma comportamenti anomali, ingestibili e imprevisti. Quindi si vuole mettere in evidenza che il sistema risulta, in questo caso *fault-tolerante*. In questa simulazione si assume il margine massimo di divergenza fra le velocità delle singole ruote pari a $5Km/h$, in quanto un valore più elevato, potrebbe generare uno sbandamento del veicolo, o comunque il sistema potrebbe essere ingestibile. Le Figure 5.31 e 5.32 mostrano il comportamento del sistema quando lo stesso non ha rilevato ancora nessun comportamento anomalo. Tutto risulta perfettamente

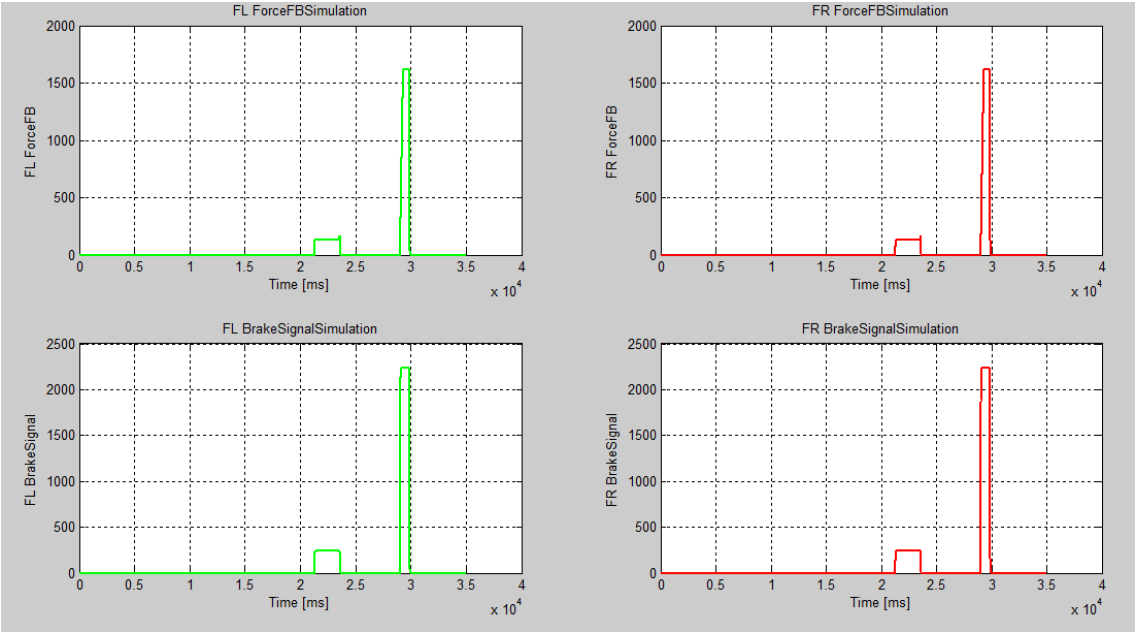

Figura 5.30: Seconda simulazione, andamento nel tempo delle variabili relative alla frenata

funzionante, la velocità procede secondo quanto stabilito, la frenata è regolare e il bus distribuito acquisisce e invia regolarmente i dati. In particolare se si osserva la Figura 5.32 si può apprezzare come, durante la simulazione, il valore della *WheelSpeedDiag* sia pari a zero e come il sistema resta in questo stato. Come accennato, si genera un *fault* quando la variabile di diagnosi *WheelSpeedDiag* assume il valore pari a 1, ossia il sistema, e in particolare il *MainNode*, si è accorto che qualcosa non sta funzionando correttamente, e che è necessaria un'azione correttiva. Nella simulazione, il sistema va in *fault* a seguito della crescente differenza tra la velocità della FL e della FR. Le ruote hanno questo diverso comportamento perché la FL ha due coefficienti ripartizione della forza settati in modo scorretto. Quindi non appena si effettua la frenata in queste condizioni il modello inizia ad rispondere in modo non corretto. Il *MainNode* si accorge dell'anomalia quando la FL si discosta dalla velocità delle altre tre ruote di un valore maggiore di $5km/h$.Questo valore è stato imposto, quindi è possibile modicarlo e renderlo adeguato ad ogni tipo di sistema. Tanto più piccolo è questo range, tanto più selettivo e sicuro risulta il sistema. Non appena il *MainNode* rileva il guasto egli stesso provvede a informare i restanti nodi. Quindi il bus FlexRay distribuito e la tipologia di rete assumono un ruolo fondamentale nella comunicazione e quindi nella sicurezza. In questo semplice caso tutto ciò risulta immediato, tuttavia è opportuno considerare che i casi in cui la quantità di dati da trasmettere è elevata, il sistema è più complicato potrebbe

non essere così banale. Dunque risulta necessaria una adeguato studio su sistemi complicati. In questa trascorso un certo intervallo di tempo t,pari a $20ms$, il *Main-*

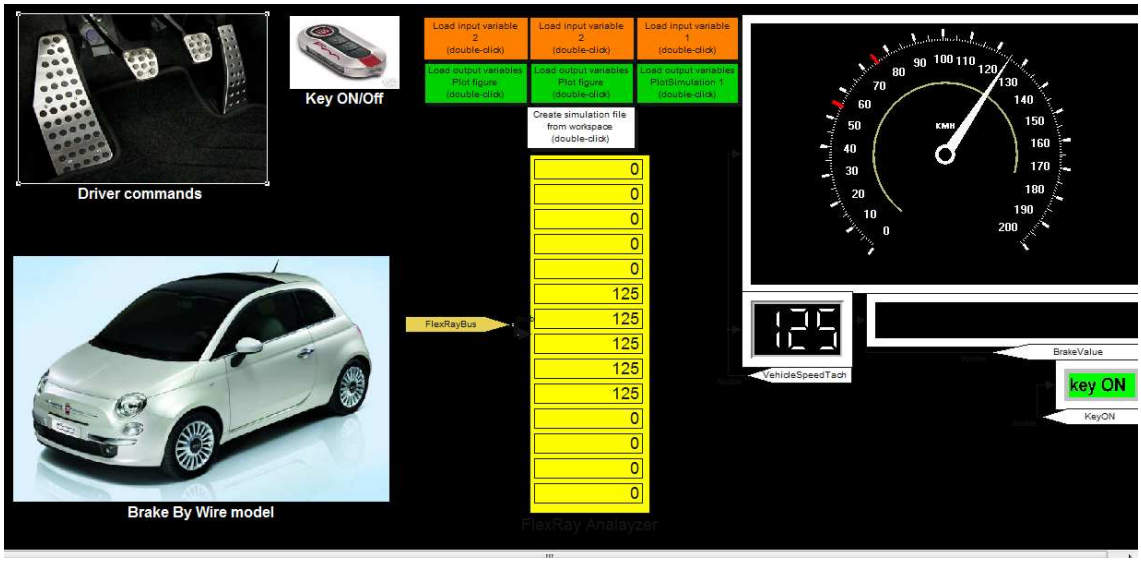

Figura 5.31: Interfaccia grafica prima del fault

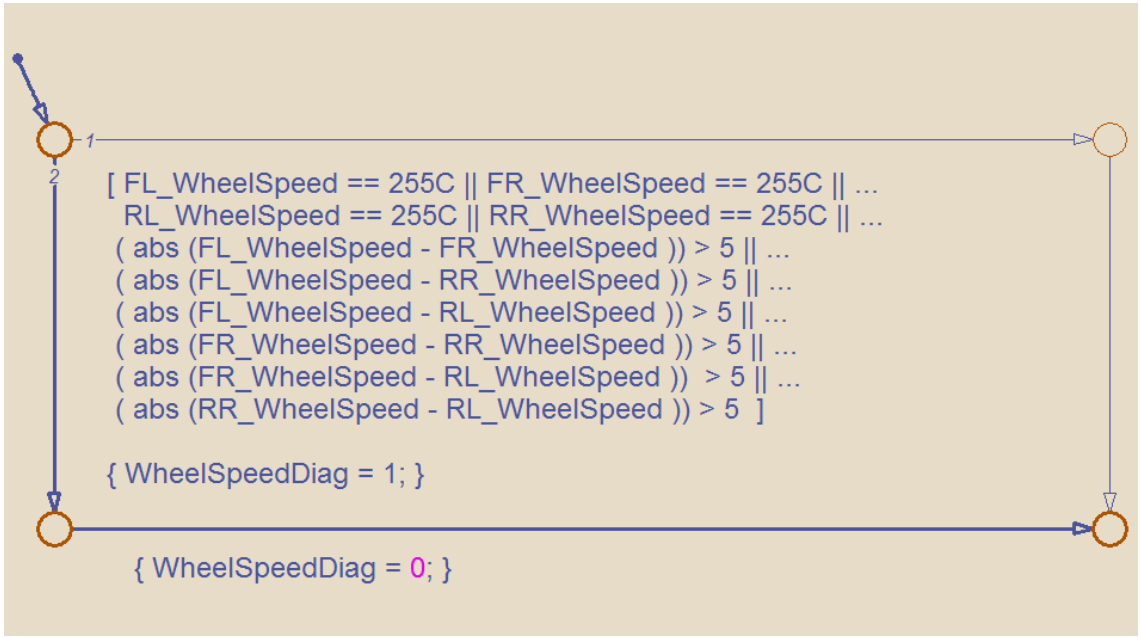

Figura 5.32: Diagramma a stati prima del fault

Node si accorge che c'è un *fault* su una ruota, in particolare sulla ruotaFL. Come conseguenza il valore della variabile *WheelSpeedDiag* va a 1, cioè viene segnalato il

comportamento anomalo della ruota FL. Dalle Figura 5.33 e 5.34 si può apprezzare meglio quanto detto, ossia quanto generato dal sistema dopo la scoperta del *fault*. Infatti nell'interfaccia, per renderlo visibile è stato inserito il segnale *WheelDiagFlag*, che genera un segnale luminoso, inizialmente lampeggiante e poi fisso, che avverte del guasto. Di particolare interesse risulta osservare come, il diagramma a stati della Figura 5.32 non appena il sistema si accorge del *fault* cambi stato per passare a quello di guasto mostrato in Figura 5.34. Si vede come la variabile *WheelSpeedDiag* passi dal valore 0 a 1 innescando il nuovo stato. Osservando gli slot del bus si vede

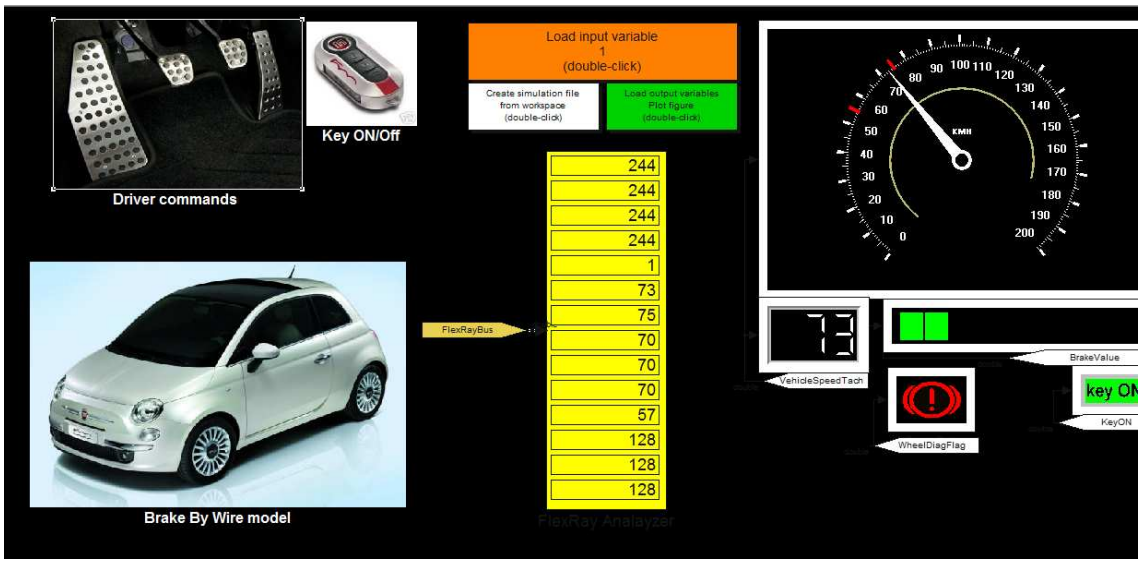

Figura 5.33: Interfaccia grafica dopo il fault

come la ruota FL si discosta dalla velocità delle altre ruota di un valore superiore al range imposto, conseguenza del fatto che la variabile *WheelSpeedDiag* ha cambiato stato passando al valore 1. Inoltre è possibile osservare dal bus distribuito, come durante la frenata, i valori di forza N impressi alla ruota con fault e a quelle che funzionano correttamente sono diverse, come del resto ci si aspettava. Zoom sulla velocità

- La variabile *VehicleSpeedValue* prima della generazione del *fault* assume i valori derivanti dai nodi ruota coerentemente a quanto stabilito. Una volta che si rileva il guasto questa inizia a discostare dal valore di velocità del nodo con *fault*.
- I segnali FL_Speed e FR_Speed rappresentano la velocità rispettivamente della ruota FL e FR. Questi due valori devono avere valore identico o comunque che

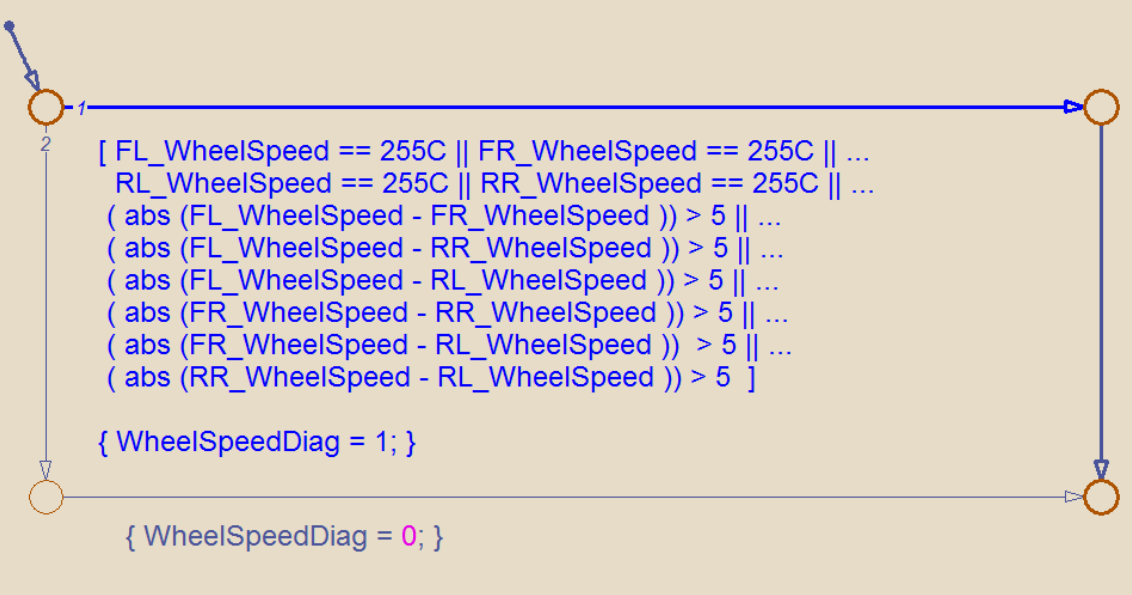

Figura 5.34: Diagramma a stati dopo il fault

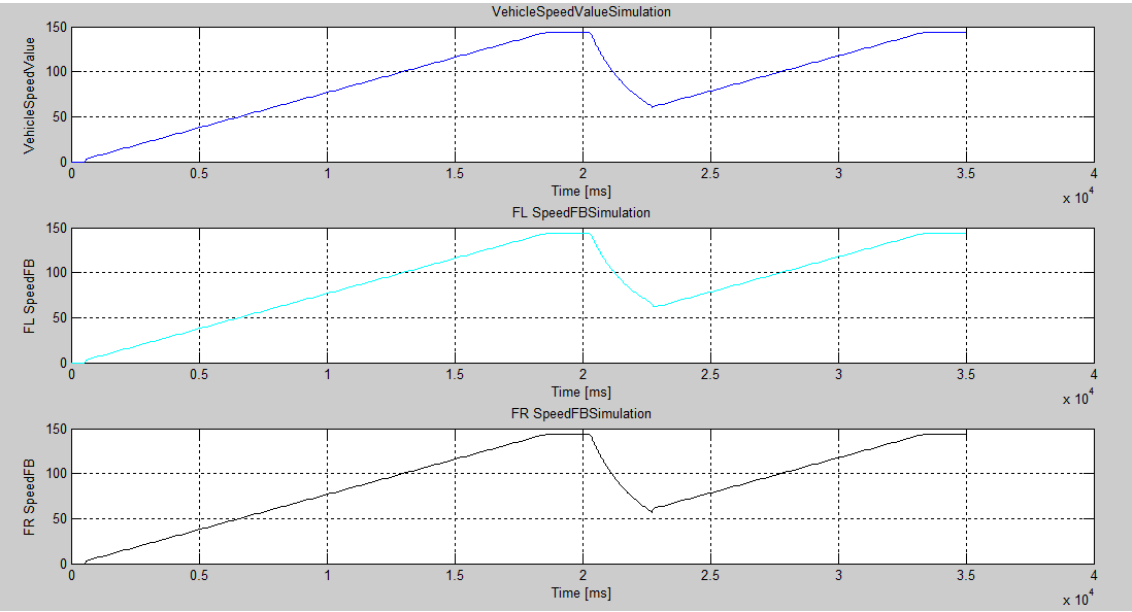

Figura 5.35: Terza simulazione, andamento nel tempo della velocità

non discosti dal valore di range imposto perché non si crei un comportamento del veicolo non controllabile.

- Il *BrakePedalPosition* ci indica il valore di posizione del pedale a cui corrisponde una determinata forza di frenata.

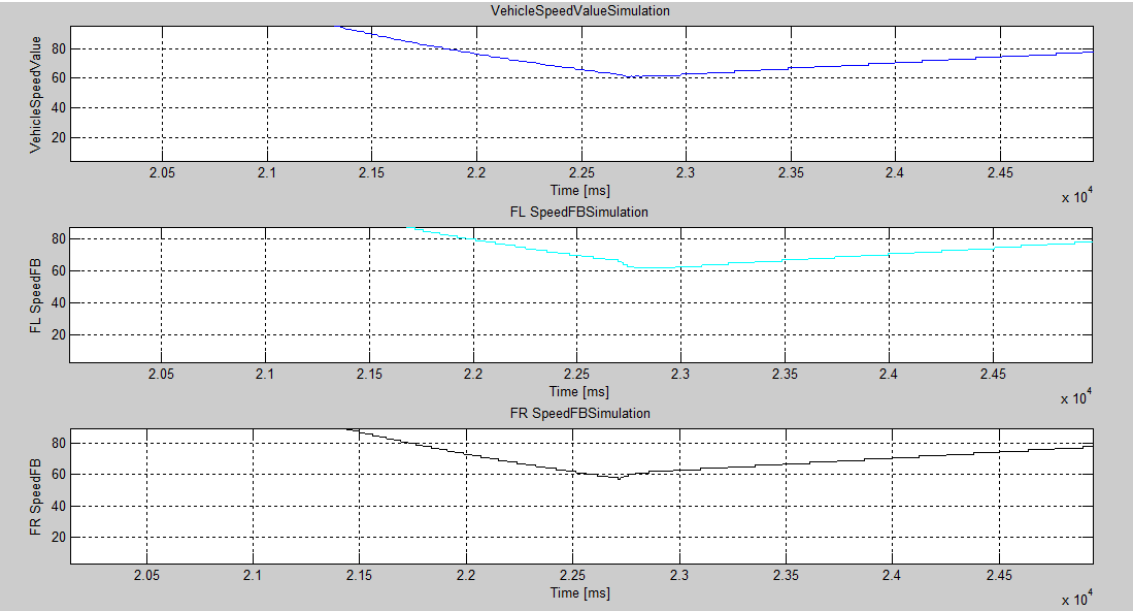

Figura 5.36: Zoom dei segnali di velocità

- I segnali *FL_BrakeSignal* e *FR_BrakeSignal* mostrano come il valore di frenata che il *MainNode* calcola e invia nella rete per essere applicato in ogni singolo nodo sia uguale per entrambe le ruote, mentre il valore realmente applicato in ognuna di esse è diverso. Questo fa capire che il *fault* è a livello ruota e non a livello *MainNode*.

Capitolo 6

Conclusioni e sviluppi futuri

6.1 Conclusioni

Il progetto di tesi, nato dalla collaborazione fra l'Università e Akhela, è risultato molto produttivo sia dal punto di vista strettamente accademico sia dal punto di vista umano e personale.

Il lavoro presentato inizia con un'introduzione in cui si spiegano i motivi per cui negli ultimi anni lo sviluppo di sistemi di sicurezza nel settore automobilistico è cresciuto, e continua a crescere, costantemente. Nell'ultima parte dell'introduzione si è introdotto l'argomento principale della tesi. Con il secondo capitolo si sono volute dare delle nozioni basilari sulla dinamica del veicolo, per poter avere una visione opportunamente critica durante la scelta delle caratteristiche del modello che si andrà poi a realizzazione. Il terzo capitolo presenta i sistemi *Brake-by-Wire*, vengono spiegate le loro caratteristiche e l'importanza dei sistemi *real-time* ad essi collegati. Si è poi analizzato il protocollo di comunicazione usato per la realizzazione del modello. Alcuni aspetti sono stati trattati sia a livello teorico che a livello applicativo con il progetto creato. Alcuni aspetti, che non sono stati applicati, sono stati comunque spiegati per poter avere una panoramica su tutto ciò che riguarda il FlexRay. Non meno interessante risulta la parte dedicata al confronto fra questo nuovo protocollo e quelli sviluppati in precedenza, come il CAN, il TTP. La parte più corposa e dispendiosa dal punto di vista del tempo è stata dedicata alla realizzazione del modello presentato. Il lavoro non è stato semplice e breve. Il modello realizzato può considerarsi fedele alla realtà, anche se è necessario precisare, come chiaramente spiegato durante la realizzazione della tesi, che sono state fatte delle semplificazioni nelle scelte di progetto. Queste semplificazioni si sono mostrate d'obbligo nel momento in cui il sistema risultava ingestibile a causa del numero

troppo elevato di variabili non deterministiche. Inoltre gli ottimi risultati ottenuti fino a questo livello di progettazione non devono nascondere le problematiche che si potrebbero presentare con casi di studio più articolati. Sarà infatti importante valutare, per esempio, il comportamento del *bus FlexRay* in casistiche in cui sarà necessario gestire situazioni critiche, come per esempio l'eventuale utilizzo di 1023 slot statiche per ogni canale. Nel nostro sistema il numero di slot utilizzati e i dati in trasmissione sono ridotti rispetto alle capacità del FlexRay.

Per la realizzazione di questo documento sono risultati fondamentali:

- I documenti della letteratura, non facili da reperire e non in grande numero. Infatti moltissimi documenti, essendo comunque il FlexRay un argomento ancora non studiato abbastanza, sono molto recenti.
- Indispensabile sono state le conoscenze e la competenza offerta in azienda, nonché gli strumenti messi a disposizione, anche se questi risulteranno più importanti nel proseguo del progetto.
- Le funzionalità di SIMULINK si sono rivelate indispensabili per la realizzazione del modello.

6.2 Sviluppi futuri

Il progetto realizzato in questa tesi, è il punto di partenza per lo studio del FlexRay da parte dell'azienda. Il lavoro svolto, iniziato nel Luglio del 2008, termina oggi con questo elaborato che è solo l'inizio del progetto ideato. Il prossimo passo del lavoro sarà predire un prototipo di architettura, distribuita e modulare, in grado di ospitare una generica funzionalità automotive distribuita. Tale prototipo di architettura distribuita sarà composto da schede che colloquiano su bus FlexRay. Il monitoring delle schede verrà fatto tramite comunicazione fra 2 PC sfruttando il FlexRay Analizer. Verranno quindi sviluppati e configurati gli ambienti *mathwork* in modo da generare del codice compilabile da downloadare e debuggare direttamente sulle schede finali. *MlabSimulink* deve generare il codice per il microprocessore e per il driver. Akhela, in particolare, usa il Driver *FlexRay UNIFIED Driver* e il microprocessore $MPC5567$. Il *FlexRay UNIFIED Driver* separa le funzionalità specifiche hardware in funzioni driver con una certa API (*Application Program Interface*) e fornisce l'indipendenza hardware per le applicazioni utente. Inoltre sostituisce la maggior parte delle caratteristiche dei moduli con delle funzioni disponibili e fornisce un possibile metodo (*Low Label Access Support*) che supporta completamente le funzionalità

dei modelli FlexRay. Il pacchetto del *FlexRay UNIFIED Driver* è distribuito con due esempi di applicazioni per mostrare i possibili usi del driver. La connettività del *Cluster* basata su *FlexRay UNIFIED Driver* può essere stabilita usando vari *Freescale FlexRay Communication Controllers*, come gli indipendenti MFR4300 e MFR4310 oppure con *microcontroller* che hanno i moduli FlexRay integrati, quali l'MC9512XFR128, MPC5567, MPC5561, MPC5516 e MC9512XF12/384/256/128.

Il *UNIFIED DRIVER* implementa le seguenti funzionalità dei moduli:

- Inizializzazione e configurazione delle funzioni.
- Le funzioni di Wakeup, startup, Media Access Test Symbol e Change Mode.
- Supporto alla trasmissione dei dati e alla ricezione (i supporti trasmettono *message buffer* singoli o doppi e ricevono *message buffer*).
- Supporto alla modalità *Poll driven*, al *monitoraggio dello stato*, all'*interrupt*, al *timer*.
- Supporto al *Low Level Access*.

Per quanto riguarda il microprocessore quello che si utilizzerà è l'MPC5567 *Embedded CONTROLLER*. Questo è il primo controller a 32-bit, per *Freescale Semicondutor's MPC55xx Family*, che offre un controller FlexRay su chip. Contenendo il *Book E-compliant Power PC core* con codifica a lunghezza variabile, il MPC5567 è ideale per qualsiasi applicazione che richiede sia alta performance che FlexRay in ambiente automotive /industriale. Offre performance cinque volte superiori rispetto al MPC500 e permette, grazie alla sua struttura, il facile passaggio da sistemi che utilizzano la famiglia degli $MPC500$, nonchè un facile riuso delle architetture software. Quindi questo microprocessore è adatto all'uso nel settore automobilistico, avionico, robotico per il controllo di turbine, dell'iniezione, per il controllo del movimento, controllo delle trasmissioni, ecc. Tale codice permette di monitorare il comportamento on line (con interfaccia grafica dove visualizzare tramite grafici il comportamento delle variabili dentro i due sistemi e sul BUS). La board sviluppata per questo microprocessore è la MPC5567EVB, adatta per lo sviluppo di piattaforme hardware o software.

Bibliografia

[1] "FlexRay Consortium."

[2] P. Koopman, "The FlexRay Protocol," Significant material draw from FlexRay Specification Version 2.0, June 2004.

[3] P. Koopman, "Significant material drawn from FlexRay Specification," Version 2.0, June 2004.

[4] B. Zhang, "On the Formal Verification of the FlexRay Communication Protocol," Department of Informatics Technische Univeisitat Munchen Munich, Germany, 2006.

[5] R. Makowitz, C. Temple "A Communication Network for Automotive Control Systems," Factory Communication Systems, 2006 IEEE International WorkShop on, June 27 3006.

[6] K. Parnell, "Telematic Digital Convergenze," Springer Berlin Heidelberg, 2004.

[7] P.M. Szecowka, M.A. Swiderski, "On hardware implementation of flexray bus guardian module," Wroclaw University of Technology, Poland.

[8] S. Moggi, "Sistemi Distribuiti in Automotive, " 2006.

[9] G. Burzio, "Autonica," *Mondo Digitale n°3, pp. 35-47*, Settembre 2004.

[10] M. Miller, "Scientific Reserch Laboratory MD1170," Dearborn 1996. *Mondo Digitale n°3, pp. 35-47*, Settembre 2004.

[11] G. Leen, D. Heffernan "Expanding Automotive Electronic Systems," University of Limerick, January 2002.

[12] G. Genta, "Meccanica dell'autoveicolo, Collana di Progettazione e Costruzione delle Macchine," Libreria Editrice Universitaria Levrotto & Bella, Torino 2002.

[13] G. L.Morlini, "Modellistica e controllo di un freno elettromagnetico," Università di Bergamo, 2004.

[14] D. Dupont, B. Armstrong, V. Hayward, "Elasto-Plastic friction Model: Contact Compliance and Stiction," Proc. 2000 American Control Conference, June 28-30, 2000, Chicago.

[15] D. Darnopp, "Computer simulation of stick-slip friction in mechanical dynamic systems," ASME Journal of Dyinamic Systems, Measurement and Control, 1985.

[16] P. Miller, "A Propotype Distributed Archicteture for Safety Critical Automotive Systems," SAE 2007, Transactional Journal of Enginers, SAE Paper Number 2007-01-1617.

[17] G. C. Buttazzo, "Sistemi in Tempo Reale," Pitagore Editrice Bologna, Terza edizione 2006.

[18] R. K. Jurgen, "X-by Wire Automotive Systems," Publisched by SAE International 400 Commonwealth Drive Warrendale PA 15096-0001 U.S.A., 2008.

[19] J. Swevers,F. Al-Bender, G.Ganseman,T. Prajogo "An Integrated Friction Model Structure with Improved Presliding Behavior for Accurate Friction Compensation," IEEE Transactions on automatic control, april 2000.

[20] V. Lampaert, J. Swevers,F. Al-Bender "Modification of the Leuven Integrated Friction Model Structure," IEEE Transactions on automatic control, April 2000.

[21] G. Leen, D. Heffernan, A. Dunne "Digital Network in the Automotive Vehicle," in *IEE Computer and Control Eng.J.*, Dec 1999, pp. 257-266.

[22] K. Hwang, "Computer Arithmetic - Principles, architecture and Design," Wiley, 1979.

[23] E. R. Fossum, "Digital camera system on a chip," *IEEE*, 1998.

[24] A. Rindi, S. Papini, L. Pugi, J. Auciello, "Dinamica del veicolo," Università di Firenze, 2006.

[25] E. Armengaud, A. Steining, "A Monitoring concept for an automotive distributed network - The Flexray example," Vienna University of Thecnology.

[26] A. Rindi, S. Papini, L. Pugi, J. Auciello "Dinamica del veicolo," Università di Firenze, 2006.

[27] R. Murphy, F. Walsh, B. Jackaman, "Migration Framework from CAN to FlexRay," Automotive Control Group, Waterford Institute of Technology, Cork Road, Waterford, Ireland, 2008.

[28] J. Ferreira, P. Pedreiras, L Almeida, J. Fonseca, "The FTT-CAN Protocol Flexibility in Safety-Critical Systems," Istituto Politécnico de Castello Branco and Universidade de Aveiro, 2002.

[29] H. Kopetz, "Fault Containment and Error Detection in TTP/C and FlexRay," Research Report 23/2002 Version 1.5, Technical University of Vienna, August 28, 2002.

[30] H. Kopetz, "A Comparison of TTP/C and FlexRay," Research Report 10/2001, Technishe Universitat Wien, Austria, May 9, 2001.

[31] Freescale Semiconductor, "FlexRay Freescale UNIFIED Driver User Guide," UG REV.1.2, 11/2006.

[32] Freescale Semiconductor, "MPC5567EVB," DOC-0381-010 REV. B, 2006.

[33] Freescale Semiconductor, "MPC5567FS," REV. 0, 2006.

Appendice A: script MATLAB e file excel

.1 Script MATLAB e file excel

Vengono di seguito riportati gli script necessari per eseguire la simulazione e gli script e le tabelle che vengono generati.

- Questo script permette la creazione delle variabili di input, naturalmente deve essere adatto per ogni specifica simulazione, rappresenta quindi il caso generale. *Create_input_varibles*:

```
PRESENT = 2; num = 48; fileName = strcat ('InputValue', num);
while (exist(strcat(fileName,'.xls'),'file')== PRESENT)
num = num+1;
fileName = strcat ( 'InputValue', char(num) );
end


xlswrite( fileName , ' ', '', 'A1')


xlswrite( fileName, 't' , 'prova', 'A1' );
xlswrite( fileName, 'SimulationTime' , 'prova', 'B1' );
xlswrite( fileName, 'SpeedSetPointValue' , 'prova', 'C1' );
xlswrite( fileName, 'AccPedalPositionValue' , 'prova', 'D1' );
xlswrite( fileName, 'BrakePedalPosition' , 'prova', 'E1' );
xlswrite( fileName, 'BrakePedalPressure' , 'prova', 'F1' );
xlswrite( fileName, 'FL_Speed' , 'prova', 'G1' );
xlswrite( fileName, 'FR_Speed' , 'prova', 'H1' );
```

```
xlswrite( fileName, t , 'prova', 'A2' );
xlswrite( fileName, SimulationTime , 'prova', 'B2' );
xlswrite( fileName, SpeedSetPointValue , 'prova', 'C2' );
xlswrite( fileName, AccPedalPositionValue , 'prova', 'D2' );
xlswrite( fileName, BrakePedalPosition , 'prova', 'E2' );
xlswrite( fileName, BrakePedalPressure , 'prova', 'F2' );
xlswrite( fileName, 'FL_Speed' , 'prova', 'G2' );
xlswrite( fileName, 'FR_Speed' , 'prova', 'H2' );
```

- Il seguente script permette di creare le variabili in uscita.

```
SimulationTime = VehicleSpeed.time;

FL_BrakeSignal = FlexRayBus.signals.values(:,1);
FR_BrakeSignal = FlexRayBus.signals.values(:,2);
RL_BrakeSignal = FlexRayBus.signals.values(:,3);
RR_BrakeSignal = FlexRayBus.signals.values(:,4);
WheelDiagnosis = FlexRayBus.signals.values(:,5);
VehicleSpeedValue = FlexRayBus.signals.values(:,6);
FL_Speed = FlexRayBus.signals.values(:,7);
FR_Speed = FlexRayBus.signals.values(:,8);
RL_Speed = FlexRayBus.signals.values(:,9);
RR_Speed = FlexRayBus.signals.values(:,10);
FL_ForceFB = FlexRayBus.signals.values(:,11);
FR_ForceFB = FlexRayBus.signals.values(:,12);
RL_ForceFB = FlexRayBus.signals.values(:,13);
RR_ForceFB = FlexRayBus.signals.values(:,14);

my_closereq(SimulationTime, VehicleSpeedValue , FL_Speed, FR_Speed,
```

- Importa i dai dal file excel e permette quindi di caricare la simulazione che si intende far eseguire.
  function importfile(myFileName)

```
newData1 = importdata('InputValue.xls');
```

fields = fieldnames(newData1.data);
newData1.data = newData1.data.(fields1);
fields = fieldnames(newData1.textdata);
newData1.textdata = newData1.textdata.(fields1);
fields = fieldnames(newData1.colheaders);
newData1.colheaders = newData1.colheaders.(fields1);

colheaders = genvarname(newData1.colheaders);
for i = 1:length(colheaders)
dataByColumn1.(colheadersi) = newData1.data(:, i);
end vars = fieldnames(dataByColumn1);
for i = 1:length(vars) assignin('base', varsi, dataByColumn1.(varsi));
end

- Questo script, adattato per ogni tipo di simulazione che si vuole eseguire, permette la creazione dei grafici e la visualizzazione al variare del tempo delle variabili.

  function my_closereq(SimulationTime, VehicleSpeedValue , FL_Speed,FR_Speed, SpeedSetPointValue, FL_ForceFB, FR_ForceFB , FL_BrakeSignal , FR_BrakeSignal)

  selection = questdlg('Plot result on new FIGURE ?',...
  'Close Request Function',... 'Yes','No','Yes');

  switch selection,
  case 'Yes' 'ListString', segnali); figure1 = figure('PaperType','a4letter','PaperSize',[34 33],...
  'Name','FlexRaySimulationDiagram');
  subplot(8,1,1)
  plot(SimulationTime, VehicleSpeedValue,'b')
  xlabel('Time [ms]');
  ylabel('VehicleSpeedValue');
  title('VehicleSpeedValueSimulation');
  grid on;
  subplot(8,1,2)
  plot(SimulationTime, FL_ForceFB ,'g–')
  xlabel('Time [ms]');
  ylabel('FL ForceFB');
  title('FL ForceFBSimulation');

```
grid on;
subplot(8,1,3)
plot(SimulationTime, FR_ForceFB ,'g:')
xlabel('Time [ms]');
ylabel('FR ForceFB');
title('FR ForceFBSimulation');
grid on;
subplot(8,1,4)
plot(SimulationTime, SpeedSetPointValue,'m')
xlabel('Time [ms]');
ylabel('SpeedSetPointValue');
title('SpeedSetPointValueSimulation');
grid on;

subplot(8,1,5)
plot(SimulationTime, FL_Speed,'c–')
xlabel('Time [ms]');
ylabel('FL SpeedFB');
title('FL SpeedFBSimulation');
grid on;
subplot(8,1,6)
plot(SimulationTime, FR_Speed,'k')
xlabel('Time [ms]');
ylabel('FR SpeedFB');
title('FR SpeedFBSimulation');
grid on;

subplot(8,1,7)
plot(SimulationTime, FL_BrakeSignal,'r–')
xlabel('Time [ms]');
ylabel('FL BrakeSignal');
title('FL BrakeSignalSimulation');
grid on;
subplot(8,1,8)
plot(SimulationTime, FR_BrakeSignal,'r:')
xlabel('Time [ms]');
ylabel('FR BrakeSignal');
title('FR BrakeSignalSimulation');
```

grid on;

case 'No'
return
end
end

- Questo è lo script che permette la visualizzazione delle finestre di dialogo prima della simulazione, carica i dati relativi alle simulazioni che si vogliono eseguire e richiamando gli script precedentemente inseriti. function my_StartSimulation(Start)
prove = 'PROVA 1', 'PROVA 2', 'PROVA 3';
myFileNameStr=' ';

[scelta, v] = listdlg('PromptString','Select a file:',... 'SelectionMode','single',.. . 'ListString', prove); switch scelta,
case '0'
myFileNameStr = 'InputValue0';
case '1'
myFileNameStr = 'InputValue1';
case '2'
myFileNameStr = 'InputValue2';
otherwise
myFileNameStr = 'InputValue';
end
fileSim = strcat(myFileNameStr, '.xls');
disp(fileSim);
disp(scelta);
searchFile = exist(fileSim, 'file');
if searchFile==2
myFileName = str2num(myFileNameStr);
stringaDaVisulaizzare = strcat ('Start with simulation n°', num2str(scelta));
selection = questdlg(stringaDaVisulaizzare ,...
'Close Request Function',...
'Yes','No','Yes');
switch selection,
case 'Yes'
msgbox('CLICK OK','WAIT script is running ','help',10);
if scelta==0

```
run Import_input_from_file;
elseif scelta == 1
run Import_input_from_file0;
elseif scelta == 2
run Import_input_from_file1;
elseif scelta == 3
run Import_input_from_file2;
else
run run Import_input_from_file;
end
msgbox('NOW CLICK PLAY TO START','Running ','help',10);
case 'No'
return
end
else
msgbox('FILE NOT FOUND !','ERROR ','error',10);
end
end
```

- I seguenti dati rappresentano una tipica tabella che viene creata dal modello quando viene premuto il tasto bianco nell'interfaccia grafica. Naturalmente modificando opportunamente gli script possono essere tabellate le variabili desidarate. In questo caso la prima colonna rappresenta tempo t, la seconda il *SimulationTime*, la terza la variabile *SpeedSetPointValue*, la quarta la varibile*AccPedalPositionValue*, a seguire il *BrakePedalPosition* e il *BrakePedalPressure*. La tabella va da 0 a 35020.

0	0	0	0	0	0	
1	0	0	0	0	0	
2	2	0	0	0	0	
3	3	0	0	0	0	
4	4	0	0	0	0	