POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale

Corso di Laurea in Ingegneria Aeronautica

Sviluppo di un sistema di controllo real-time per un modello di aerogeneratore in galleria del vento

Relatore: Prof. Carlo L. BOTTASSO
Tutor: Ing. Filippo CAMPAGNOLO

Tesi di Laurea di:

Luca MAFFENINI Matr. 720719

Anno Accademico 2009 - 2010

Ringrazio innanzitutto il Professor Carlo Bottasso per avermi proposto questo lavoro di tesi, e l'Ingegnere Filippo Campagnolo per avermi aiutato a realizzarlo. Inoltre, ringrazio sentitamente il Professor Roberto Bucher per la sua disponibilità e pazienza nel chiarire i miei dubbi sulle parti più tecniche del lavoro.

Un sentito grazie va anche ai miei amici e compagni di studi, vicini e lontani, che hanno condiviso con me questo lungo viaggio e senza i quali non sarebbe stato lo stesso.

Per ultimi, ma non per importanza, desidero ringraziare i miei genitori, per avermi aiutato e sostenuto in questi anni e aver contribuito in maniera così fondamentale al raggiungimento di questo traguardo.

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

In questo elaborato di tesi si presenterà lo sviluppo di un sistema di controllo real-time per un modello di aerogeneratore da utilizzare in galleria del vento. Il lavoro fa parte di un progetto triennale e si inserisce in una fase in cui il l'apparato meccanico è già stato definito e realizzato.

In particolare, si discuterà della configurazione degli attuatori da utilizzare per il controllo del sistema e individuati in una fase precedente del progetto, nonché le modalità di comunicazione e funzionamento in un ambiente real-time.

Si analizzerà successivamente il metodo di funzionamento di un vero aerogeneratore, in particolare la logica della macchina agli stati che ne gestisce il controllo, riproponendo una soluzione per il modello scalato; si evidenzieranno nel dettaglio le differenze tra gli stati delle due macchine, i punti in comune e si discuterà della necessità di aggiungere delle modalità che permettano il tracciamento delle curve prestazionali per validare il modello in galleria del vento.

Una volta definita la logica di controllo e supervisione se ne studierà quindi l'implementazione in RTAI, un applicativo real-time opensource sviluppato al Politecnico di Milano, e se ne discuteranno i differenti aspetti legati alla sua configurazione e al suo utilizzo, i punti di forza e le criticità; se ne costruirà quindi una realizzazione preliminare che permetterà di valutare il buon funzionamento della componentistica meccanica, degli attuatori e della logica di controllo.

Si analizzerà in seguito un prodotto industriale utilizzato sui veri aerogeneratori, il sistema Bachmann, studiandone l'adattabilità al modello scalato e confrontandone le possibilità di sviluppo e le capacità operative con quelle offerte da RTAI.

Si vedrà infine l'utilizzo del modello scalato in galleria del vento con la soluzione real-time sviluppata, evidenziando la programmazione delle prove, i risultati operativi ottenuti e le criticità rilevate.

Parole chiave: controllo real-time, aerogeneratore, modello scalato, galleria del vento, macchina agli stati, RTAI, Bachmann, prove in galleria.

Abstract

In this report it will be presented the development of a real-time control system for a wind turbine model to be used in a wind tunnel. This work is a part of a three years project and takes place when the mechanical parts has been already defined and realised.

It will be discussed the actuators configuration to be used to control the system, actuators already chosen in earlier phase of the project, and the communication and operating modes in a real time environment.

Then it will be analysed how a real wind turbine works, focusing on the state machine logic that runs on-board, and studying a new solution for the scaled model; it will be highlight the differences between the states of the real wind turbine and the scaled mode, the common aspects, and it will emerge the needs to add some functionalitys to run the model in order to make the project data validation tests in the wind tunnel.

Ones the control logic has been defined, it will be studied his implementation in RTAI, an open-source real time project developed at Politecnico di Milano, and it will be discussed the different aspects about his configuration and use, the advantages of the system and the his criticalities. A preliminary realisation will be constructed in order to evaluate the good behaviour of the mechanical parts , the actuators and the control logic.

It will be later analysed an industrial product used on the real wind turbines, the Bachmann system, making a study about the possibility to use it with the scaled model and making a comparison with RTAI's development possibilities and capabilitys.

Finally it will be shown the developed real time solution behaviour in the wind tunnel, highlighting the tests organisation, the operative results and the criticalities detected.

Keywords: real time control, wind turbine, scaled model, wind tunnel, state machine, RTAI, Bachmann, wind tunnel tests.

1. Introduzione

1.1. Il progetto $(WT)^2$

Questo lavoro di tesi fa parte del progetto $(WT)^2$*, the Wind Turbine in a Wind Tunnel,* il cui obiettivo è realizzare un modello scalato aero-servo-elastico di un aerogeneratore di taglia multi Mega-watt da utilizzare in galleria del vento.

Lo scopo principale di tale modello è di essere utilizzato come strumento complementare ai metodi usati nella progettazione degli aerogeneratori, attualmente basata su successive iterazioni tra analisi numeriche e sperimentazioni sul campo con prototipi in scala 1:1. Questo approccio soffre di due problemi principali:

1. incertezza sui risultati forniti dai codici di simulazione aeroelastici, soprattutto nel caso di riproduzione di condizioni operative estreme definite dalle normative.

2. scarsità di dati sperimentali in tali condizioni, dovuta all'eccezionalità con la quale queste si verificano.

Lavorare in galleria del vento permette di avere un ambiente controllato nel quale riprodurre condizioni di prova arbitrarie, in particolare quelle difficilmente verificabili sul campo; risulta quindi il contesto ideale per lo studio e la verifica delle leggi di controllo degli aerogeneratori.

Il Dipartimento di Ingegneria Aerospaziale lavora da diverso tempo in questo ambito, sviluppando dei controllori progettati per ridurre i carichi agenti sulla struttura e massimizzare l'energia prodotta. In galleria del vento è possibile sperimentare diverse leggi e confrontarne le prestazioni in condizioni fissate e ripetibili, operazione altrimenti difficile da realizzare su un vero aerogeneratore che lavora in un ambiente dall'andamento imprevedibile come l'atmosfera.

Realizzando più modelli si può inoltre studiare l'influenza della scia sul comportamento aerodinamico degli aerogeneratori in un parco eolico, permettendo così:

1. lo studio e la realizzazione di leggi di controllo ancora più sofisticate, che tengano conto dell'interazione tra più macchine

2. lo studio della disposizione degli aerogeneratori nel parco, così da minimizzare gli effetti di scia e massimizzare la produzione energetica.

Il progetto prevede una durata di 3 anni ed è sponsorizzato da *Vestas*, leader mondiale nel settore dell'energia eolica. Il modello aeroelastico è la riproduzione in scala della macchina *V90* di *Vestas*, un aerogeneratore da 3 MW di potenza nominale, tripala, con un rotore di 90 metri di diametro.

1.2. Obiettivi

Obiettivo di questa tesi è la realizzazione del sistema di comunicazione, acquisizione dati e controllo del modello aeroelastico, arrivando alla configurazione di uno strumento che deve essere il più possibile semplice e affidabile.

Il lavoro si inserisce come proseguo del progetto sviluppato in due tesi precedenti: in [1] è stato definita la scalatura aeroelastica dell'aerogeneratore, mentre in [2] è stata sviluppata la progettazione preliminare dei vari componenti, tra cui le parti meccaniche e il sistema di attuazione e sensoristica necessario al controllo del sistema.

Dato che l'assemblaggio del modello è avvenuto parallelamente a questo lavoro di tesi, si è reso necessario adattare il progetto alle problematiche che si presentavano di volta in volta durante l'avanzamento dei lavori, senza trascurare il rispetto delle deadline temporali prefissate.

Per la realizzazione preliminare del sistema di controllo si è utilizzato RTAI, un'estensione real-time del kernel linux sviluppata all'interno del DIA. L'obiettivo è stato quello di ottenere un programma completo che fosse in grado di comunicare con il modello aeroelastico in tempo reale e offrire allo stesso tempo la possibilità di implementare e testare diverse leggi di controllo: si è ottenuta così una soluzione funzionante di tipo prototipale che ha permesso di verificare le funzionalità base del sistema.

Successivamente si è valutata l'ipotesi di sostituire questa prima realizzazione con un apparato di tipo industriale espressamente progettato per operare nell'ambito del controllo degli aerogeneratori. L'apparato in questione è il modello M1 della Bachmann, costituito da elementi hardware e software basati su elevati standard industriali che permettono di innalzare il livello globale di qualità del sistema completo. Essendo inoltre uno strumento normalmente usato su veri aerogeneratori, grazie al suo utilizzo si ha un'ulteriore verosimiglianza dell'operatività del modello scalato.

In questo lavoro si propone l'analisi delle problematiche dell'implementazione del controllo del sistema inteso come gestione dell'apparato durante le prove in galleria del vento (test delle funzionalità, gestione delle modalità operative, scelta delle leggi di controllo da testare, ecc) e della sua sperimentazione sul campo. Dati i vincoli temporali di disponibilità della galleria e di realizzazione del modello, quest'ultima parte non ricopre tutti gli aspetti possibili, ma si concentra sull'utilizzo in modalità manuale del sistema, realizzata per ricavare le curve prestazionali dell'aerogeneratore.

2. Modello dell'aerogeneratore

In questo capitolo si espongono e si analizzano i componenti impiegati per acquisire le misure che si vogliono ricavare dal modello, i sensori utilizzati e gli attuatori necessari al controllo dello stato del sistema.

Figura 2.1.: Aerogeneratore scalato

Nel lavoro descritto in [2] sono state definite due fasi di realizzazione del sistema, caratterizzate da due diverse modalità operative:

modello 0 il sistema è azionato manualmente, ruota a velocità costante e il passo delle pale è fisso

modello 1 il sistema è azionato in modo autonomo dal controllore, che può modificare dinamicamente il passo delle pale e la coppia erogata dal generatore

A queste due diverse condizioni operative corrispondo scopi diversi: il *modello 0* è usato per tracciare le curve di prestazione dell'aerogeneratore, mentre il *modello 1* viene utilizzato per lo studio e la verifica sperimentale delle leggi di controllo sviluppate al DIA.

2.1. Modello 0

Come già accennato precedentemente, lo scopo principale del *modello 0* è il tracciamento delle curve di prestazione dell'aerogeneratore, in particolare le $C_P - \lambda$, $C_Q - \lambda$ e $C_T - \lambda$, dove:

- λ è il *tip speed ratio*, definito come:

$$\lambda = \frac{\omega R}{V_0}$$

  con ωR velocità tangenziale all'estremità delle pale e V_0 velocità del vento che investe l'aerogeneratore.

- C_P è il coefficiente adimensionale di potenza, definito come:

$$C_P = \frac{P_r}{\frac{1}{2}\rho A V_0^2}$$

  con P_r potenza aerodinamica ceduta alle pale, A area del disco rotore formato dalle pale e ρ densità dell'aria.

- C_Q è il coefficiente adimensionale di coppia, definito come:

$$C_Q = \frac{Q}{\frac{1}{2}\rho A R V_0^2}$$

  con Q coppia aerodinamica esercitata sull'albero del rotore e R raggio del rotore stesso.

- C_T è il coefficiente di spinta, definito come:

$$C_T = \frac{T}{\frac{1}{2}\rho A V_0^2}$$

  con T spinta esercitata sul rotore nella direzione del vento.

Ad ogni passo di pala β corrisponde una diversa curva prestazionale, che viene tracciata effettuando delle prove in galleria dove, fissato il passo di pala, si misurano coppia, spinta, velocità del vento e di rotazione dell'albero per diversi valori di λ.

2.2. Modello 1

Il *modello 1* rappresenta lo stadio successivo del progetto. In questo caso le curve prestazionali sono già state tracciate, quello che interessa è implementare le leggi di controllo e verificarne l'efficacia.

Si rende necessario quindi prevedere diverse modalità operative dell'aerogeneratore, che permettano di passare da uno stato di funzionamento a un altro in modo rapido e continuo. Si terrà conto di questo nel capitolo 4, dove vengono analizzati gli aspetti di gestione del sistema.

Questa fase non può essere realizzata senza aver prima concluso quella precedente: il *modello 0* infatti fornisce le informazioni necessarie a verificare che i requisiti di progetto siano rispettati, come ad esempio l'aerodinamica delle pale e le frequenze proprie della struttura.

Sebbene il funzionamento del *modello 1* sia stato progettato durante questa tesi, a causa dei tempi di realizzazione del sistema (assemblaggio, test funzionali, ecc.) non è stato possibile sperimentarlo in galleria del vento.

2.3. Stato avanzamento progetto

Il modello aeroelastico utilizzato è un sistema che si trova a metà strada tra il *modello 0* e il *modello 1.*

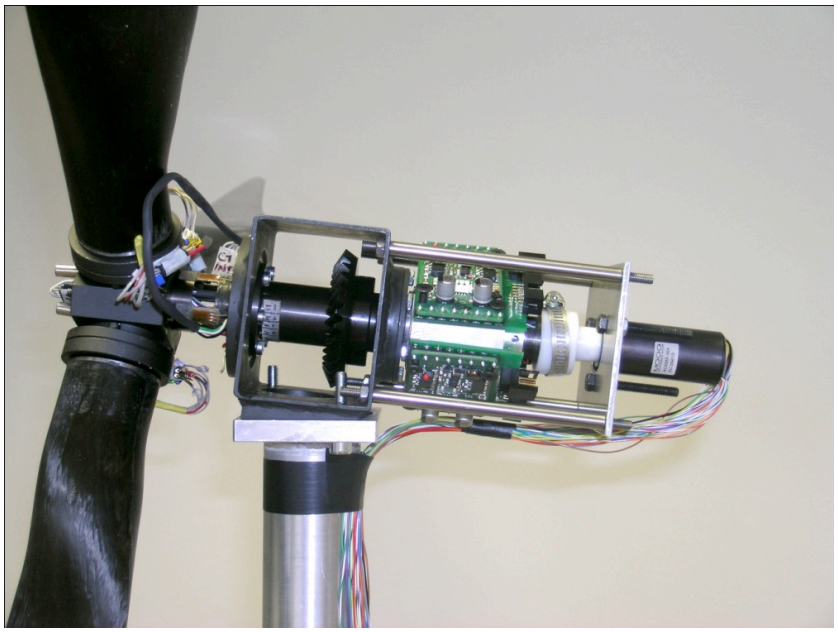

Figura 2.2.: Aerogeneratore: stato dell'arte

Per quanto riguarda la struttura, questa è costituita da una torre rigida, le cui frequenze proprie non corrispondono a quelle scalate, e da pale non aeroelastiche,

ossia rigide come la torre, ma costruite in modo tale da riprodurre l'aerodinamica di progetto.

Gli attuatori a disposizione sono:

- 3 motori in corrente continua per il controllo del passo di pala
- 1 motore brushless per il controllo della velocità di rotazione del rotore

I motori di passo pala sono controllati in posizione e impongono l'angolo β desiderato in modo indipendente per ciascuna delle 3 pale. Il motore brushless presenta invece due diverse modalità operative: la prima è il controllo in velocità, dove il motore fornisce la coppia necessaria a far ruotare il rotore a velocità fissata, qualunque sia la coppia aerodinamica generata dalle pale; la seconda è il controllo in coppia, dove il motore agisce da freno ed equilibra la coppia prodotta dall'aerodinamica, come accade in un vero aerogeneratore.

Per quanto riguarda le misure si ritrovano i seguenti sensori:

- torsiometro sul mozzo, per la misura della coppia sull'albero del rotore
- encoder sui motori di passo pala, per la misura dell'angolo di pitch delle pale
- inverter sul motore di coppia, per la misura della velocità di rotazione del rotore
- potenziometri sulle pale, per un'ulteriore misura dell'angolo di pitch

2.3.1. Attuatori di passo

Gli attuatori di passo devono imporre l'angolo di pitch delle pale e mantenerlo fisso nel *modello 0*, mentre devono poterlo variare con una dinamica assegnata nel *modello 1*. Queste specifiche statiche e dinamiche sono riportate in [2] e sono:

Tabella 2.1.: Specifiche attuatori di passo

Max pitch rate	76 rpm
Coppia max	57 Nmm
Potenza max	0.45 W
Banda passante	30 Hz
Escursione	-5° +90°
Precisione posizionamento	±0.1°

Il sistema utilizzato è costituito dai seguenti elementi:

Tabella 2.2.: Attuatori di passo

Motore	Faulhaber DC-Micromotors Series 1524, 012 SR da 2.5 mNm
Encoder	Faulhaber Series IE2 - 512
Centralina	Faulhaber MCDC 3003 C

Tra l'albero del motore e l'albero a cui è fissato il carico si trova inoltre un riduttore di tipo zero backlash, che garantisce l'assenza di gioco tra le due parti. Usando questi componenti il controllo di posizione è affidato alla centralina, la quale riceve in ingresso un setpoint di posizione e produce in uscita la tensione necessaria per portare il motore al riferimento imposto.

Verifica prestazioni statiche e dinamiche

Il sistema motore più centralina deve rispondere alle specifiche statiche e dinamiche della tabella 2.1. Questi requisiti non vengono soddisfatti con le configurazioni di default della centralina, il cui sistema di controllo risulta troppo blando.

Se infatti si misura la precisione statica, imponendo al sistema un disturbo di coppia esterno, si ha che la pala riesce a compiere una rotazione di qualche grado prima di essere fermata dal sistema di controllo: è quindi necessario incrementare le prestazioni del controllore per renderlo più robusto ai disturbi esterni.

Per quanto riguarda la dinamica invece, si ha la seguente risposta allo scalino:

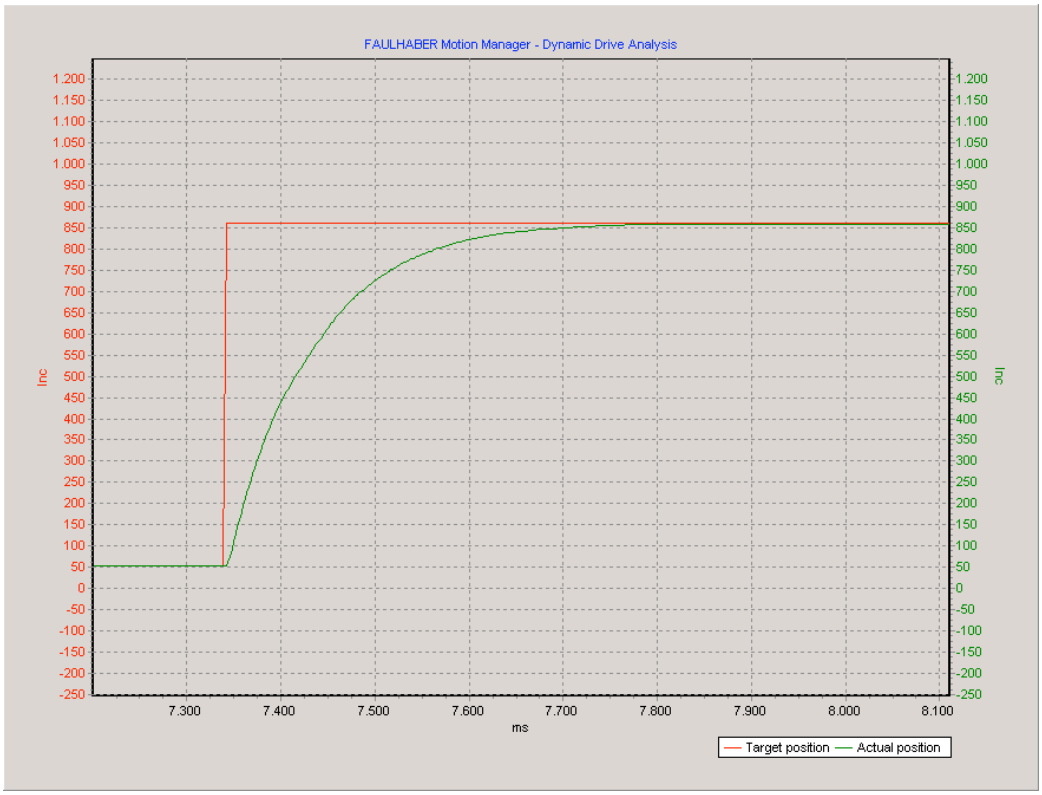

Figura 2.3.: Pitch setup originale: risposta allo scalino

La linea rossa rappresenta il riferimento di posizione, quella verde la posizione della pala; lo scalino imposto è di circa 2°, visti in figura come 860 variazioni degli impulsi dell'encoder.

Si nota come il tempo di assestamento sia di circa 260 ms. Approssimando il sistema a un modello del 1° ordine, si ha che la costante di tempo del sistema è circa 1/5 del tempo di assestamento, da cui si ricava la banda passante, che risulta essere

$$f_t = \frac{1}{2\pi\tau} = \frac{1}{2\pi 0.052} = 3.06\,Hz$$

Questa frequenza è molto più piccola di quella richiesta dalla specifiche, si rende quindi necessario aumentare anche questo tipo di prestazioni.

Setup e taratura

La centralina Faulhaber MCDC 3003 C funziona con due differenti regolatori: un PI per il controllo in velocità e un PD per il controllo in posizione.

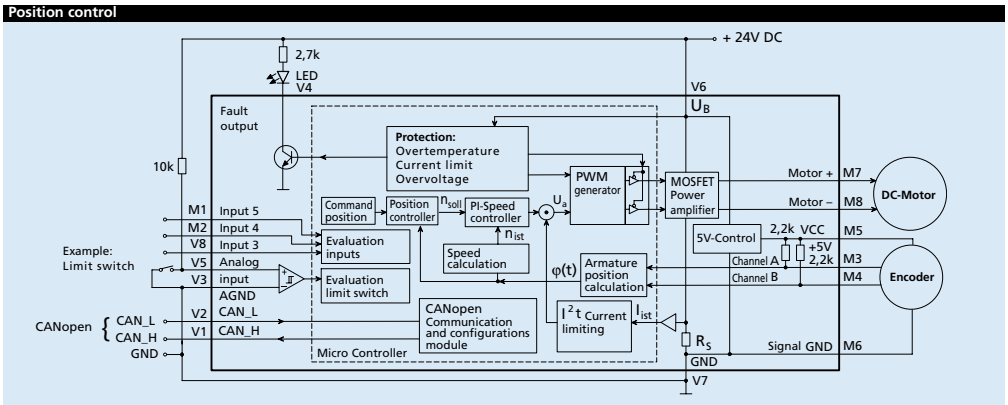

Figura 2.4.: Schema di regolazione MCDC 3003 C

Il manuale di istruzioni della Faulhaber [3] riporta la possibilità di modificare i parametri dei controllori, in particolare i seguenti elementi:

Tabella 2.3.: Parametri del controllore MCDC 3003 C

Command	Description	Value
POR	Load velocity controller amplification.	1 – 255
I	Load velocity controller integral term	1 – 255
PP	Load position controller amplification	1 – 255
PD	Load position controller D-term	1 – 255
SR	Sampling rate setting	1...20 ms/10

Non vengono riportati il significato fisico e le unità di misura di tali valori, rendendo difficile implementarli nelle equazioni classiche di un PID da utilizzarsi nello schema proposto in figura 2.4.

Si è scelto quindi di seguire un approccio diverso, ossia quello suggerito dal manuale: si sono utilizzate le regole di Ziegler-Nichols [4]. Il vantaggio di questo metodo è che non richiede un modello matematico del sistema e, in questo caso, risulta di semplice applicazione. Utilizzando questo algoritmo si è tarato prima il regolatore di velocità e successivamente quello di posizione, fino ad ottenere la seguente risposta allo scalino:

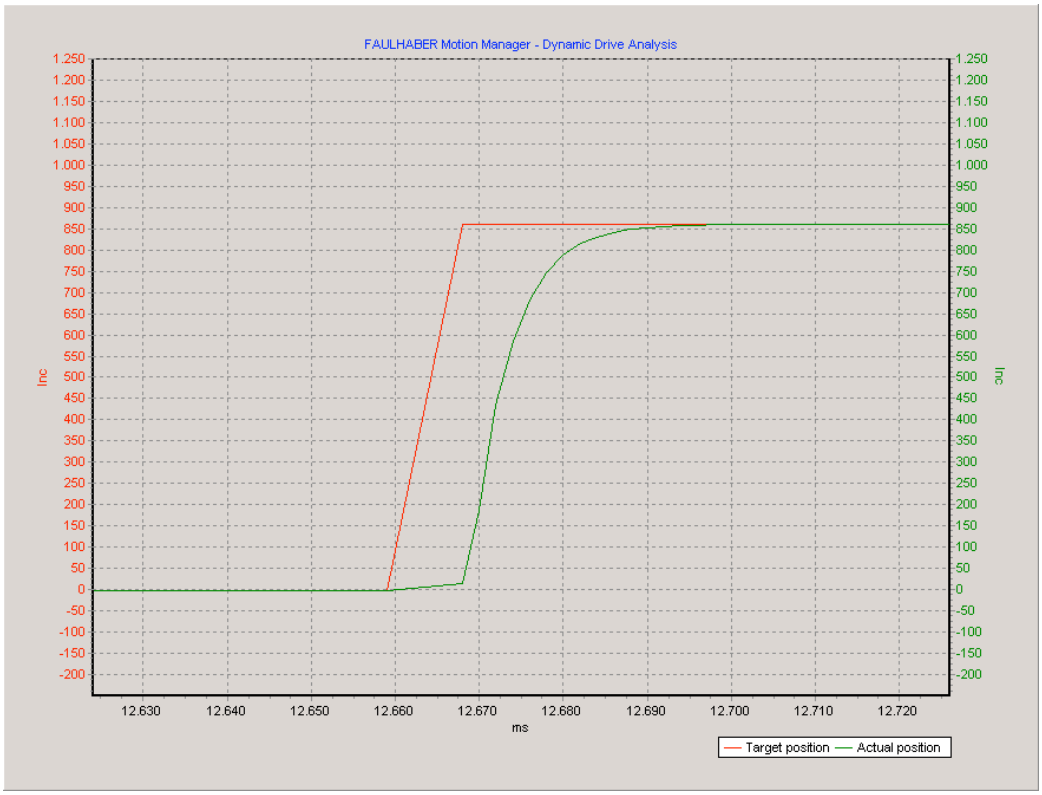

Figura 2.5.: Pitch setup ottimizzato: risposta allo scalino

La risposta in questo caso è molto più veloce rispetto alla precedente: in particolare si nota come il tempo di assestamento sia diminuito a circa 25 ms, da cui si ricava una banda passante di 31.83 Hz, risultato pienamente compatibile con le specifiche di progetto.

2.3.2. Attuatore di coppia

Come già accennato nel capitolo 2.3 l'attuatore di coppia deve funzionare in due diverse modalità operative:

1. nel *modello 0* deve fornire una velocità costante, così da poter tracciare le curve prestazionali della macchina

2. nel *modello 1* deve frenare il sistema, fornendo una coppia tale da eguagliare quella prodotta dall'aerodinamica

Le specifiche di progetto sono le seguenti:

Tabella 2.4.: Specifiche attuatore di coppia

Velocità rotore	367 rpm
Coppia max	5 Nm
Potenza max	194 W
Banda passante	571 Hz
Inerzia generatore	0.011 Kgm^2

Il sistema utilizzato consiste in un motore e un inverter, in particolare:

Tabella 2.5.: Attuatore di coppia

Motore	Portescap B1515-150A
Inverter	ARTDriveS-EV

L'inverter è stato riadattato e modificato da MCM EnergyLab [5] per poter essere usato con questo tipo di motore, in quanto la Portescap non produce un'elettronica di controllo dedicata per il B1515-150A.

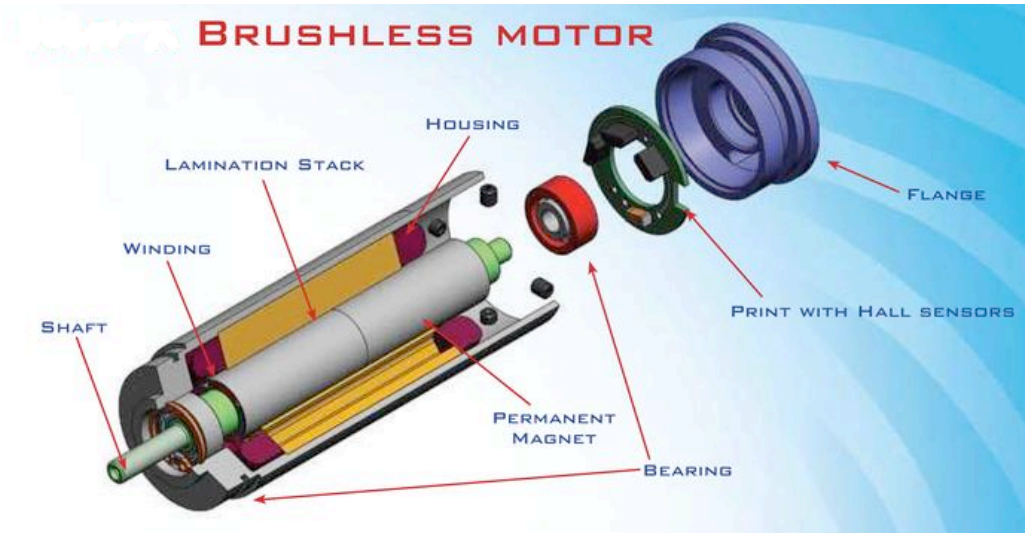

Figura 2.6.: Attuatore di coppia

Essendo stato originariamente pensato per altri usi, l'accoppiata motore e inverter presenta alcune problematiche operative: la più caratteristica è quella di non riuscire a mantenere una velocità di rotazione pulita a bassi regimi. Questo aspetto

non risulta comunque critico per il progetto, in quanto il sistema non si trova mai ad operare in questa situazione.

La parte di verifica delle prestazioni statiche e dinamiche e la relativa taratura sono state eseguite da MCM EnergyLab, la quale ha fornito così un prodotto completo e rispondente alle specifiche operative richieste.

2.3.3. Misure

Misura passo pala

La misura dell'angolo di passo delle pale viene eseguita attraverso due strumenti: l'encoder applicato al motore e le strisce potenziometriche sulle pale.

L'encoder utilizzato (vedi tabella 2.2) presenta un'elevata risoluzione, che sommata all'effetto del riduttore produce un'accuratezza di lettura dell'angolo di passo di 0.0023° per ogni variazione di impulso dell'encoder. Sebbene il riduttore sia di tipo zero backlash, si è scelto di avere una ridondanza sulla misura, andando a leggere l'angolo di passo direttamente sulla pala.

Questa operazione è eseguita attraverso delle strisce potenziometriche, che forniscono una tensione variabile a seconda della posizione del cursore.

Figura 2.7.: Struttura striscia potenziometrica per misura passa pala

La pressione applicata applicata sulla superficie varia la tensione in uscita, permettendo, una volta calibrata, di determinare l'angolo di passo pala tramite un cursore solidale con il mozzo.

Misura coppia albero

La coppia generata dall'aerodinamica viene trasferita all'albero, dove viene misurata direttamente tramite due ponti estensimetrici. Questi misurano la deformazione dell'albero data dalla coppia applicata e sono due ponti completi, offrendo così una maggiore affidabilità della misura.

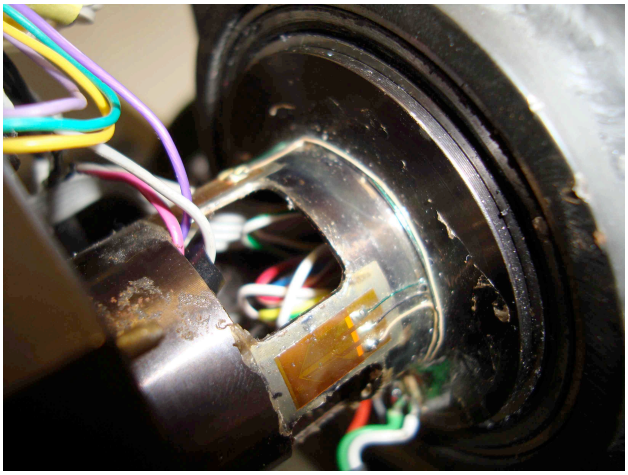

Figura 2.8.: Torsiometro

Il segnale misurato dai ponti è subito amplificato e filtrato da una scheda posta sul mozzo; il segnale viene poi portato a terra in formato analogico tramite i cavi che passano attraverso lo slipring.

Misura forze globali e angolo di yaw

La misura delle forze globali è affidata alla bilancia della Galleria del Vento. Questa è costituita da 7 ponti estensimetrici che consentono la misura delle 6 componenti di forza e momento agenti sul modello.

La base del modello è fissato alla bilancia, la quale si trova direttamente sotto il pavimento della camera di prova.

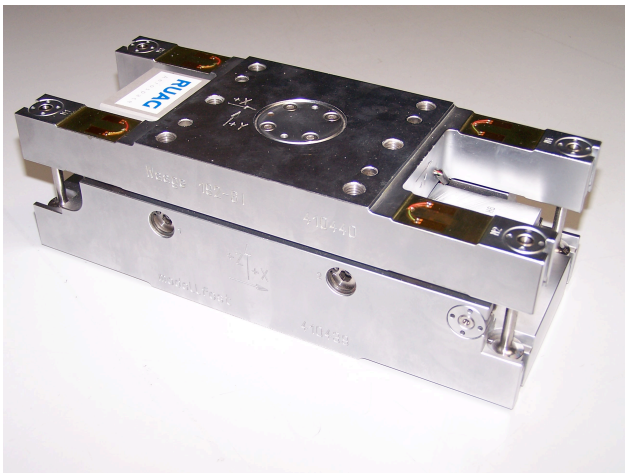

Figura 2.9.: Bilancia della Galleria del Vento

Allo stato attuale il segnale non è acquisito dal modello, ma direttamente dal sistema della galleria: il motivo di tale scelta è dovuto al fatto che l'acquisizione di 7 ponti estensimetrici necessita di un sistema di amplificazione e filtraggio dedicati.

Questo apparato, concettualmente identico a quello del torsiometro, è stato progettato ma non ancora realizzato. Le principali conseguenze sono di carattere operativo:

1. nel *modello 1* il sistema di controllo non ha le informazioni di forza a disposizione. Non è quindi possibile implementare un controllore che abbia bisogno di tali informazioni per elaborare la legge di controllo

2. nel *modello 0* si deve prevedere una adeguata sincronizzazione per poter successivamente associare i dati registrati dal modello con quelli di galleria. Questo aspetto verrà approfondito nel capitolo 6.

Per quanto riguarda la misura dell'angolo di yaw rispetto all'asse del vento, il sistema utilizzato è quello messo a disposizione dalla Galleria del Vento. Questo sistema è costituito da una base circolare di 2.40 metri di diametro che permette la rotazione a 360° del modello con una precisione di qualche centesimo di grado.

Misura velocità rotazione albero

La velocità di rotazione dell'albero è misurata utilizzando le informazioni provenienti dai sensori di Hall del motore di coppia. Data la velocità del motore, passando attraverso il rapporto di riduzione tra l'albero del motore e l'albero del rotore, si ha la velocità di rotazione delle pale.

Come già accennato in 2.3.2, la lettura a bassa velocità non è precisa, in quanto i sensori di Hall generano un basso numero di impulsi per giro. Questo è anche il motivo per cui il motore non gira pulito ma presenta degli scatti a bassi regimi, mentre funziona molto bene ad alte velocità.

Misura azimuth pala

La posizione azimutale della pala rispetto al piano verticale del rotore è un'informazione che servirà nello studio delle leggi di controllo più sofisticate. È in fase di sviluppo un progetto che prevede l'installazione di un encoder sull'albero rotore, per determinare la rotazione compiuta dall'albero. La posizione delle pale è fissa rispetto a un riferimento posto sull'albero stesso, sul quale è applicata una bolla che permette di calcolare la verticale esatta di una singola pala e azzerare così l'encoder.

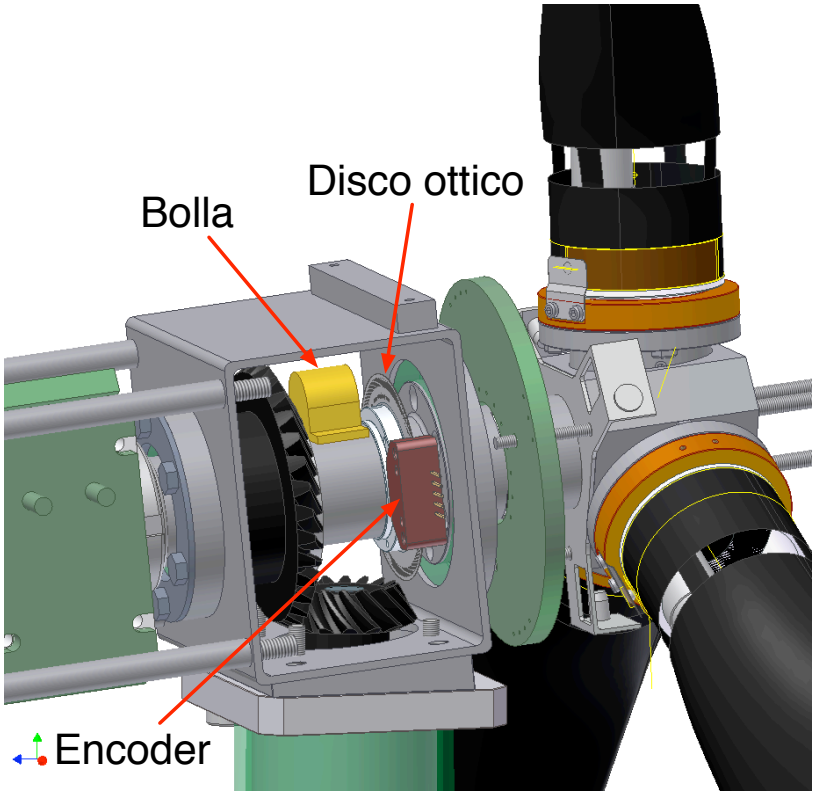

Figura 2.10.: Misura dell'angolo di azimuth delle pale

Nella figura 2.10 si evidenziano la bolla, allineata col riferimento della pala, l'encoder, agganciato alla nacelle, e il disco ottico, fissato all'albero. Combinando le informazioni fornite dai due strumenti si ottiene l'angolo di azimuth relativo alla sola pala 0, dal quale si ricava quindi la posizione delle altre 2, considerando che le pale sono distanziate di 120° tra loro.

Misura momento flettente radice pala

Nel progetto iniziale è presente anche la misura del momento flettente agente sulla radice della pala, misurata attraverso dei ponti estensimetrici. Il sistema prevede una lavorazione della pala per aumentare le deformazioni in radice, così da aumentare la sensibilità degli estensimetri, che allo stato attuale non è ancora stata effettuata.

3. Logica di controllo

3.1. Funzionamento di un vero aerogeneratore

Il funzionamento di un vero aerogeneratore è gestito attraverso una macchina agli stati, che ne commuta le funzionalità a seconda delle situazioni: sono previste fasi di avvio, di produzione di energia, arresto di emergenza, ecc. Sul modello scalato la logica di funzionamento è simile, se non più complessa: è necessario infatti prevedere ulteriori accorgimenti dovuti alle diverse condizioni operative rispetto alla macchina reale.

Si procede quindi ad analizzare tale sistema, così da evidenziare le fasi fondamentali che verranno mantenute e quelle che possono essere tralasciate. Dallo studio di un controllore realizzato dal DIA si sono estratti 6 stati:

Stato 0: idling / standing still

Nello stato di *idling* le pale sono in bandiera, il sistema è inattivo ma il rotore può ruotare a basse velocità al fine di trovare una condizione di equilibrio.

Stato 1: power production

La fase di *power production* è quella in cui l'aerogeneratore lavora normalmente, utilizzando la coppia fornita dall'aerodinamica per produrre energia elettrica. Le operazioni principali che vengono eseguite sono la verifica delle condizioni di emergenza e l'elaborazione delle leggi di controllo, quali:

- controlli di passo collettivo (PID, LQR), dove ognuna delle tre pale presenta lo stesso angolo di pitch.
- controlli di passo ciclico (LQR, HHC), dove ogni singola pala agisce in modo indipendente rispetto alle altre due.
- controllo di coppia (Lookup table, LQR, HHC), in modo da trimmare l'aerogeneratore attorno ad una velocità di rotazione schedulate in funzione del vento.

Si ha inoltre la valutazione di tutta una serie di ricostruttori degli stati, come il ricostruttore del vento o della flessione della torre, e un controllo separato per la gestione dell'angolo di yaw.

Stato 2: normal shutdown

La procedura di *normal shutdown* viene eseguita nei casi in cui ci sia poco vento o l'energia prodotta sia al di sotto di una certa soglia. Le pale sono portate lentamente in bandiera e la coppia prodotta dal generatore viene mantenuta fino a che la velocità di rotazione si abbassa; quando questa velocità è quasi nulla il generatore viene quindi scollegato.

Stato 3: emergency shutdown with generator disconnected

In caso di disconnessione del generatore elettrico le pale vengono portate rapidamente in bandiera e, una volta che il rotore ha diminuito la proprio velocità di rotazione, si applica una freno meccanico che permette di arrestare completamente il sistema.

Stato 4: emergency shutdown with generator connected

Lo stato di *emergency shutdown* può essere dovuto a diverse condizioni, come:

- overspeed
- differenza di angolo di pitch fra le pale maggiore di una certa soglia
- perdita di un attuatore
- raffica prolungata per un periodo di tempo di alcuni secondi

L'operazione che si esegue è sempre quella di portare le pale in bandiera, così da diminuire la velocità del rotore, sfruttando, qualora previsto, un sistema di frenatura elettrico oltre che meccanico.

Stato 5: startup

Quando si ha un vento sufficientemente forte per un dato intervallo di tempo, il sistema passa nella fase di *startup*. Il passo pala, i cui valori sono stati precedentemente tabulati, viene impostato in funzione del TSR e l'aerogeneratore è operato in modo tale da ricercare la coppia massima. Tutta la coppia prodotta dall'aerodinamica è utilizzata per portare il sistema in rotazione nel minor tempo possibile, motivo per cui il generatore è scollegato in questa fase.

Stato 6: Short circuit

Un corto circuito all'interno del generatore provoca l'erogazione di un picco di coppia elettrica di brevissima durata ma forte intensità, prima che il sistema di diagnostica riesca a isolare il guasto e disconnettere il generatore. Le operazioni che si eseguono sono le stesse dello stato 3.

3.2. Macchina agli stati per il modello scalato

Nel progettare la macchina agli stati del modello scalato si è tenuto conto di quanto descritto nel capitolo precedente: alcune fasi infatti sono state mantenute praticamente invariate, è stato possibile tralasciarne altre, mentre si è reso necessario crearne di nuove. La seguente tabella mostra i risultati dell'analisi.

Tabella 3.1.: Confronto stati macchina reale e scalata

Macchina reale	**Macchina scalata**
0: idling	0: idle
1: power production	5: automatic operation
2: normal shutdown	2: parking
3: emergency shutdown with generator disconnected	-
4: emergency shutdown with generator connected	-
5: startup	1: testing
6: short circuit	-
-	4: manual operation
-	3: init

Gli stati 0, 1, 2 e 5 della macchina reale sono stati conservati, seppur adattandoli. Nello stato 0, ad esempio, le pale sono libere di muoversi, così come il rotore, in modo tale da poter verificare manualmente le corse e il buon movimento degli elementi, differendo dalle condizioni di idling/standing still dove le pale sono bloccate ed al rotore sono concesse rispettivamente basse e nulle velocità di rotazione.

Gli stati 3 4 e 6 invece sono stati raggruppati in un unico stato, che prevede la gestione di tutte le condizioni di emergenza con il generatore, che può rispettivamente ricreare: coppia nulla, coppia data dal freno meccanico, oppure coppie molto intense per brevi periodi.

Rappresentando tutte le possibili fasi di lavoro del sistema, è stata definita la seguente macchina agli stati:

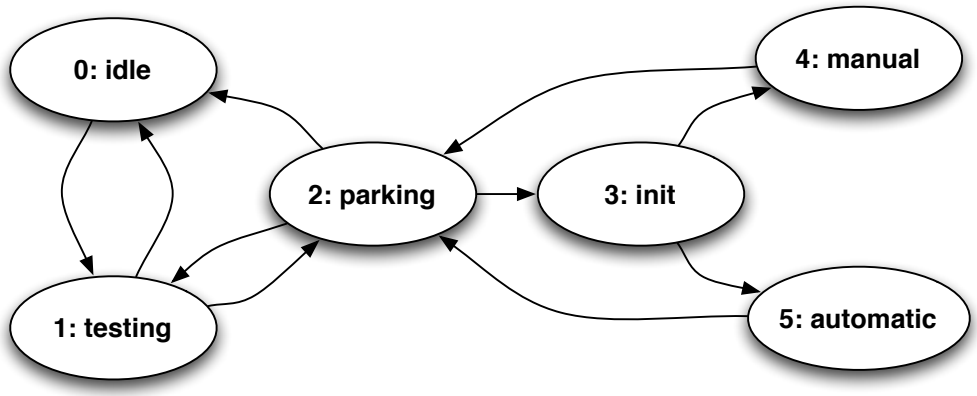

Figura 3.1.: State machine

Nella figura 3.1 è rappresentato il passaggio da uno stato all'altro, che segue dei percorsi obbligati ed è gestito dall'utente. La differenza principale rispetto al vero aerogeneratore è la necessità di poter operare il sistema anche in modalità manuale, ossia poter imporre delle velocità di rotazione e posizioni di passo pala arbitrarie per tracciare le curve prestazionali della macchina.

Stato 0: idle

Lo stato 0 è quello in cui si trova il sistema una volta acceso; i sensori e gli attuatori sono alimentati, ma il sistema è inattivo. In questa fase è possibile avviare l'interfaccia grafica sul pc monitor, connettersi al controllore e verificare così la comunicazione tra i due.

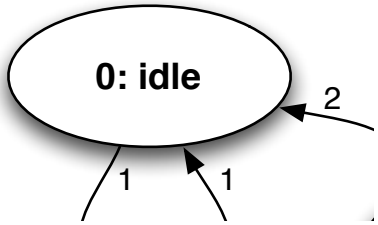

Figura 3.2.: Stato 0: idle

L'unica transizione permessa è quella verso lo stato 1, ossia quello di testing del sistema.

Stato 1: testing

Il sistema diventa attivo: le pale vengono portate in posizione 0 gradi rispetto al piano del disco rotore, che inizia a girare alla minima velocità possibile, e vengono attivati i sensori. Questa procedura, che deve essere effettuata ad ogni avvio della macchina, permette la verifica del funzionamento degli attuatori e l'inizializzazione delle misure. Nello specifico, le operazione includono:

settaggio a zero dell'encoder delle pale: all'avvio del sistema le pale si trovano in una posizione qualunque; la procedura di azzeramento dell'encoder consiste nell'aumentare il passo pala fino a giungere al fondo corsa meccanico, posizione individuata dal sistema monitorando la corrente assorbita dal motore: quando supera il doppio del valore nominale vuol dire che la coppia erogata sta aumentando, e il motivo è proprio l'ostacolo del fondo corsa. Conoscendo la distanza in gradi tra questo punto e lo 0 si può conseguentemente resettare l'encoder.

settaggio a zero dell'offset dei ponti: la misura dei ponti estensimetrici presenta un offset, che in questa fase viene memorizzato e successivamente sottratto al segnale originale. L'offset è calcolato come il valore medio del segnale registrato per durante un finestra temporale definibile dall'utente.

applicazione taratura delle strisce potenziometriche: l'uscita delle strisce potenziometriche poste sulle pale è una tensione espressa in Volt. In questa fase si applica la legge di taratura determinata a banco per ogni singola pala così da avere l'informazione di rotazione espressa direttamente in gradi.

settaggio a zero dell'azimuth: si legge il valore fornito dalla bolla e una volta che la pala si trova in posizione verticale si azzera l'encoder.

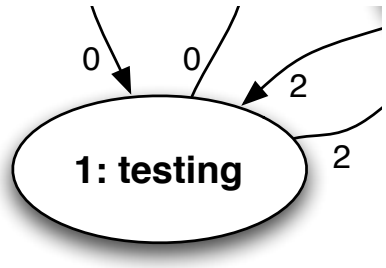

Figura 3.3.: Stato 1: testing

Come si evince dalle connessioni in figura, se tutti i sistemi risultano funzionanti si passa allo stato 2, altrimenti in caso di anomalie si può tornare allo stato 0 e spegnere il sistema per procedere alle riparazioni.

Stato 2: parking

Nello stato *parking* il rotore viene fermato e le pale sono riportate in bandiera. Arrivando dallo stato 1 il sistema è quindi stato avviato, testato ed è pronto per essere utilizzato in modalità manuale o automatica.

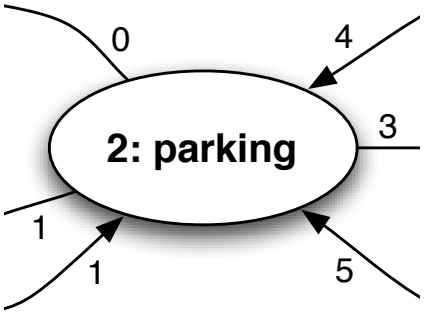

Figura 3.4.: Stato 2: parking

Si può raggiungere questa modalità anche dagli stati 4 e 5, transizione che sancisce il parcheggio del modello.

Stato 3: init

La procedura di *init* avvia il rotore e lo porta ad una data velocità, imponendo un passo di pala predefinito con ratei di velocità di rotazione e passo pala adeguati fissati a priori. Con questa operazione si ha un punto di funzionamento iniziale sempre uguale per tutte le prove manuali, garantendo così un avvio standard e sicuro.

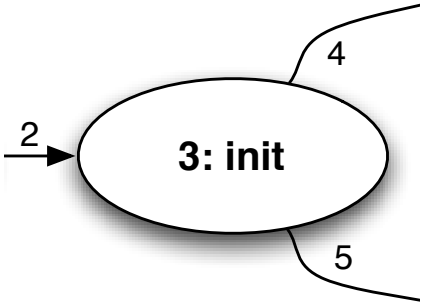

Figura 3.5.: Stato 3: init

Stato 4: manual operation

In modalità manuale l'operatore è libero di impostare il passo di pala e la velocità di rotazione desiderati, permettendo così di tracciare le curve prestazionali definite nel capitolo 2.1 per diversi angoli β. Dato che i punti di queste curve sono

determinati analizzando le condizioni statiche del sistema in diverse configurazioni di funzionamento (Ω e β), il transitorio tra un assetto e l'altro risulta ininfluente. Si è scelto quindi di imporre una legge di posizionamento fissa per entrambi i parametri che imponga una rampa di accelerazione costante per Ω e una velocità di spostamento per β.

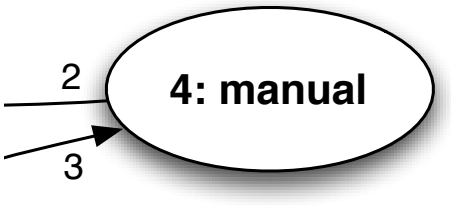

Figura 3.6.: Stato 4: manual operation

Analizzando lo schema del diagramma degli stati si nota come l'unica transizione consentita sia il ritorno allo stato di parcheggio. L'ipotesi di poter passare direttamente alla modalità automatica è stata esclusa per ragioni di sicurezza, in quanto si avrebbe una variazione brusca nei parametri con un possibile conseguente salto dei carichi agenti sul sistema.

Stato 5: automatic operation

Dallo stato di parcheggio è anche possibile passare al funzionamento automatico, dove il sistema è gestito dalle leggi di controllo implementate. In questo caso oltre alle condizioni statiche si studiano anche i valori dei transitori, così da poter analizzare la dinamica e l'efficacia dei controllori sviluppati.

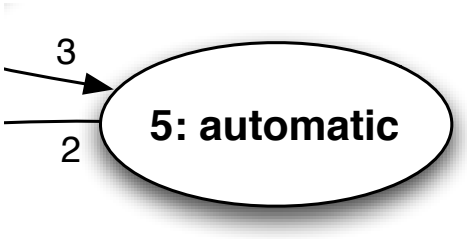

Figura 3.7.: Stato 5: automatic operation

All'interno di questo stato sono contenuti anche i possibili test e le condizioni di emergency, realizzati attraverso il controllo della coppia erogata dal generatore, quali:

- perdita di un attuatore di passo
- raffica

- perdita dell'attuatore di coppia
- picchi di coppia su brevi periodi

4. Sistema di controllo RTAI

La scalatura del modello di aerogeneratore, oltre ad essere un'operazione di tipo geometrico è anche un'operazione che tiene conto del tempo: il sistema deve agire in tempi finiti e fissati a priori da questo rapporto, che, come riportato in [1], è 1:45. Unendo questo requisito alla necessità di prestazioni e robustezza, si arriva alla scelta di utilizzare un controllore che lavori in tempo reale, ossia che possa garantire l'esecuzione di certe istruzioni in tempi massimi predicibili.

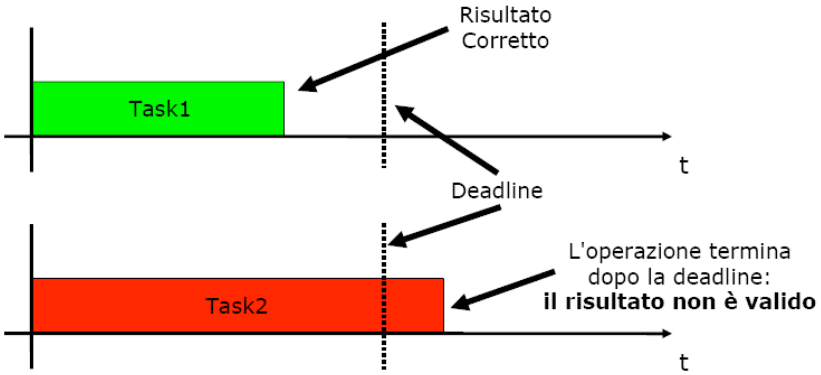

Figura 4.1.: Task real-time

Il Dipartimento di Ingegneria Aerospaziale sta sviluppando da alcuni anni il progetto *RTAI [6]*, *Real Time Application Interface for Linux*[1], il cui scopo è quello di permettere di scrivere applicazioni la cui esecuzione è strettamente legata a vincoli temporali. Questo progetto è opensource e gratuito, e risulta la scelta naturale per questo lavoro, in quanto permette di avere un sistema real-time con costi estremamente ridotti, garantendo allo stesso tempo un certo grado di assistenza durante lo sviluppo del sistema.

Per la risoluzione di problematiche real-time si usano di solito architetture hardware dedicate, dove le tre componenti (hardware, software di base e software applicativo) sono spesso strettamente legate, in modo da conseguire le necessarie ottimizzazioni sui tempi. Con l'evoluzione della tecnologia informatica, molte problematiche legate all'hardware sono state gradualmente riportate sul software e, dato che in questo caso il vincolo temporale non è particolarmente stringente, si può pensare di utilizzare del materiale non specificatamente progettato per lavorare

[1] http://www.linux.org/

in tempo reale, come ad esempio un normale PC desktop arricchito dei componenti necessari all'interazione col modello.

Il pacchetto RTAI mette a disposizione RTAI-Lab, un insieme di strumenti nato per generare e monitorare codici real-time, che semplificano la fase di sviluppo e codifica del software applicativo. Quest'ultimo aspetto è stato decisivo nella scelta di utilizzare questa suite, in quanto permette in tempi relativamente brevi di creare un eseguibile real-time e di avere un'interfaccia grafica con la quale controllarlo. Questa interfaccia si chiama xrtailab, e utilizzando degli strumenti appositi messi a disposizione da RTAI-Lab consente la realizzazione di un sistema semplice ma completo con tempi compatibili con lo sviluppo di questo lavoro di tesi.

4.1. Configurazione hardware

Per determinare la configurazione del sistema di controllo si sono analizzate innanzitutto la struttura della Galleria del Vento, la camera di prova e le modalità di gestione delle sessioni di prova.

Partendo dalla struttura dello stabile, si hanno tre componenti principali:

1. la camera di prova
2. la parte sottostante alla camera di prova
3. la control room

Il controllore deve essere collegato al modello e la distanza tra i due deve essere la minima possibile, così da ridurre la lunghezza dei cavi e il conseguente rumore sui segnali analogici. Il posizionamento in camera di prova non è realizzabile per questioni di pulizia del flusso d'aria; di conseguenza si procede facendo passare i cavi attraverso la base del modello, con il risultato di poter collocare il sistema al di sotto della camera stessa, ottenendo così una condizione ottimale.

Le possibili configurazioni diventano quindi:

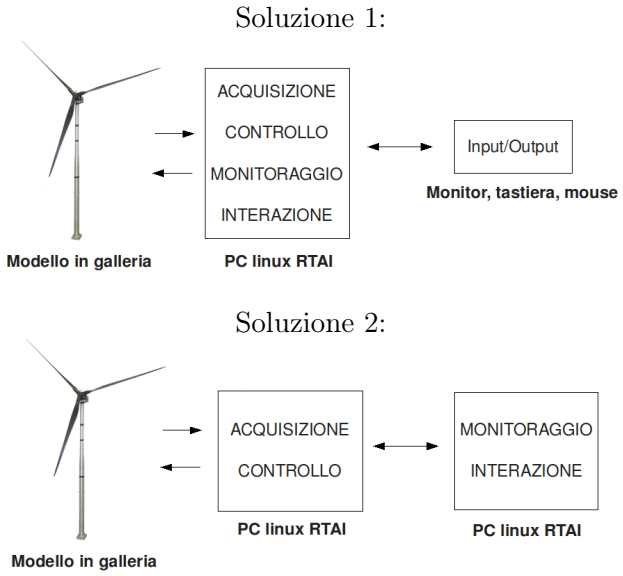

Figura 4.2.: Possibili configurazioni del sistema di controllo

La soluzione 1 prevede un singolo computer che assolve ai compiti di controllo e monitoraggio posto sotto la camera di prova: questa soluzione è la più economica, ma comporta un carico computazionale elevato, e di conseguenza la richiesta di un sistema più performante; inoltre la posizione dell'operatore al di sotto della camera non è ottimale ai fini della sicurezza.

La soluzione 2 invece prevede un sistema di controllo costituito da due unità fondamentali:

- una prima unità, denominata *controllore*, che acquisisce i segnali, elabora le leggi di controllo e comanda gli attuatori
- una seconda unità, denominata *monitor*, che si collega alla prima e permette di gestirne il funzionamento e monitorare le variabili di sistema

Questa configurazione richiede due computer, quindi ha un costo superiore alla prima, ma permette una maggiore flessibilità nello sviluppo e una maggiore sicurezza durante le prove, in quanto i sistemi di monitoraggio e controllo sono distanti l'uno dall'altro. Il luogo di lavoro ideale risulta essere quindi la control room, posizione che presenta numerosi vantaggi, tra cui avere una visione ottimale di ciò che succede in camera di prova e lavorare a stretto contatto con l'operatore della galleria.

Il controllore dovrà quindi essere un sistema espandibile, modulare, altamente affidabile e prestante, mentre le capacità del monitor potranno essere più modeste.

4.1.1. PC controllore

La scalatura aeroelastica dell'aerogeneratore impone un vincolo temporale sull'elaborazione delle leggi di controllo, individuando così delle prestazioni minime da raggiungere.

Tradurre questo vincolo in termini di costo computazionale non è un'operazione semplice: esso dipende dalla potenza necessaria a calcolare l'azione di controllo, dalla frequenza di lettura e scrittura di ingressi e uscite del sistema, dalla quantità di dati che si intende memorizzare su disco. Le prestazioni di un computer dipendono poi da molti fattori, quali la frequenza di calcolo della CPU, la velocità del bus di sistema, la quantità e il tipo di memoria RAM, la velocità di accesso al disco fisso, ecc.

Tradurre tutti questi parametri in numeri e specifiche minime sui componenti è un'impresa complessa, anche perché, oltre all'hardware impiegato, la velocità di esecuzione di un task dipende dal sistema operativo su cui gira (Windows, Linux, sistema real-time o non real-time..), dal codice con cui è stato scritto (C, Java..) e dalla priorità stessa del task.

Si è quindi scelto un approccio di tipo empirico: per realizzare il controllore si è utilizzato un computer già esistente, costituito da componenti di non ultimissima generazione. Una volta configurata la macchina si è installato il software e sono stati eseguiti dei test funzionali per determinare le prestazioni globali del sistema: nel caso queste fossero state insufficienti si sarebbe successivamente provveduto ad incrementarle acquistando componenti più performanti.

Il computer utilizzato per lo sviluppo iniziale è costituito dai seguenti elementi:

Tabella 4.1.: PC controllore

Scheda madre	ASUS A7V8X
Processore	AMD Athlon XP 2600+
Memoria	MDT Technologies 512Mb PC333, 232.4 MHz
Hard disk	Maxtor 6Y080L0 80Gb ATA-133
Scheda grafica	Matrox Millennium G450 Dual Head, AGP 4x 16 MB SDRAM
Scheda di rete	VIA Rhine II Fast Ethernet Adapter 100Mbps

A questa configurazione standard è stato necessario aggiungere due schede PCI per la piena comunicazione con il modello: una di acquisizione dati e l'altra di comunicazione per il CAN bus.

Scheda acquisizione dati

Per l'acquisizione dei segnali analogici si usa la scheda NI PCI-6014 della National Instruments, un dispositivo per il quale esistono i driver real-time implementati in RTAI.

Figura 4.3.: Scheda National Instruments NI PCI-6014

Il suo utilizzo non era previsto nel progetto originale, ma si è reso necessario a causa degli elevati tempi di realizzazione del sistema di acquisizione "on board".

Tabella 4.2.: Scheda NI PCI-6014

Bus	PCI
Analog inputs	16SE / 8 DI
Input resolution [bit]	16
Sampling rate [kS/s]	200
Input range [V]	±0.05 to ±10
Analog outputs	2
Output resolution [bit]	16
Output rate [kS/s]	10
Output range [V]	±10
Digital I/O	8
Counter/timers	2
Triggers	Digital

La scheda lavora a 16 bit, consente di applicare quattro diversi guadagni ai segnali e acquisire fino a 8 canali in modalità differenziale.

Di questi 8 canali, attualmente ne vengono usati 5:

- 3 per gli angoli di passo pala

- 2 per i ponti del torsiometro

Le prestazioni di acquisizione risultano adeguate nella configurazione attuale, mentre il numero di ingressi non è sufficiente se si pensa agli sviluppi futuri, dove la quantità di segnali è destinata ad aumentare. Quest'ultimo aspetto non risulta essere un problema dal momento in cui sarà disponibile il sistema di acquisizione "on board" originale, che andrà di fatto a sostituire la PCI-6014.

Scheda CAN bus

La comunicazione con gli attuatori avviene tramite il CAN bus, un particolare tipo di bus per la trasmissione dati che verrà analizzato meglio nel capitolo 4.3.

Questo tipo di connessione necessita di una scheda aggiuntiva che fornisca le interfacce di collegamento con il bus. In questo caso i requisiti da soddisfare sono di compatibilità: per poter lavorare in real-time sono infatti necessari driver compatibili con l'interfaccia RTAI.

Una scheda che risponde a questi requisiti è quella della Peak System, e in particolare si è scelta la versione PCI.

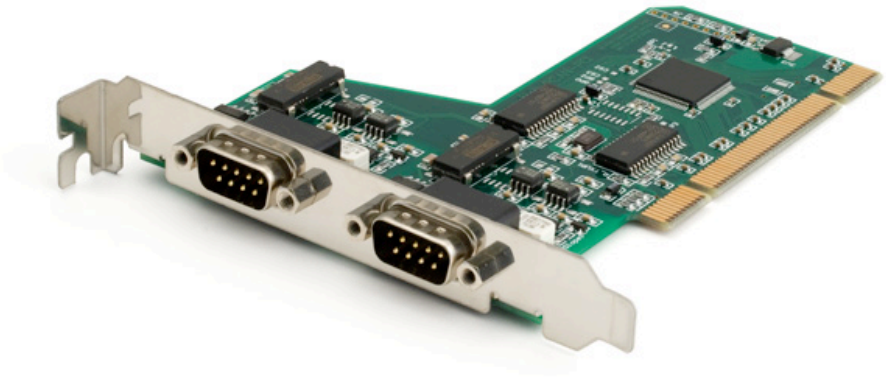

Figura 4.4.: Scheda Peak System PCAN-PCI

Le specifiche hardware e software sono:

- PC plug-in card for PCI slot
- Transfer rates up to 1 Mbit/s

- Compliant with CAN specifications 2.0A (11-bit ID) and 2.0B (29-bit ID)
- CAN bus connection via D-Sub, 9-pin (to CiA® 102)
- NXP SJA1000 CAN controller, 16 MHz clock frequency
- 82C251 CAN transceiver
- 5-Volts supply to the CAN connection can be connected through a solder jumper, e.g. for external transceiver
- Galvanic isolation on the CAN connection up to 500 V, separate for each CAN channel

4.1.2. PC monitor

Il PC monitor non richiede elementi particolari dal punto di vista hardware, deve solo possedere una scheda di rete per poter comunicare con il PC controllore. Dal punto di vista delle prestazioni invece deve essere più performante, in quanto la parte grafica che mostra le informazioni richiede una certa potenza di calcolo per essere eseguita in modo fluido.

Durante la configurazione del PC monitor si incontra il problema dell'interfaccia grafica, ossia come progettarla dal punto di vista strutturale (finestre, bottoni, ecc.) e come realizzarla dal punto di vista del codice (Java, Qt, ecc.). Per comunicare con il codice real-time si devono utilizzare le API fornite da RTAI e implementarle nel linguaggio scelto per la realizzazione del programma, mentre per rappresentare le parti grafiche è necessario scrivere il codice che le disegni, ne gestisca le variazioni a seconda dei dati ricevuti e interagisca con l'utente.

Come già accennato all'inizio del capitolo, queste operazioni richiedono una quantità di tempo e di lavoro non compatibili con la fase preliminare del progetto. Si è quindi optato per l'utilizzo di *xrtailab*, un software contenuto nella suite RTAI-Lab, che dà la possibilità di monitorare gli eseguibili generati in RTAI e mostrarne i valori a video tramite oscilloscopi o altri elementi grafici. Questo strumento consente di disegnare semplici interfacce grafiche personalizzate, permettendo così la verifica delle funzionalità base del sistema.

L'utilizzo di xrtailab è vincolato alla generazione di eseguibili tramite Scilab, un software progettato disegnare il sistema di controllo tramite schemi a blocchi e che verrà analizzato nel dettaglio nel capitolo 4.2.2.

4.2. RTAI-Lab

RTAI-Lab[7] è un insieme di strumenti nato per creare e sviluppare software di controllo real-time. Permette di avere un sistema CACSD[2] gratuito, open-source e modificabile secondo le proprie esigenze.

In particolare, RTAI-Lab permette di:

[2]Computer Aided Control System Design

- sviluppare ed eseguire software real-time in modalità locale, remota o distribuita
- monitorare l'esecuzione locare, remota o distribuita di un processo
- cambiare i parametri di un controllore durante l'esecuzione di un processo

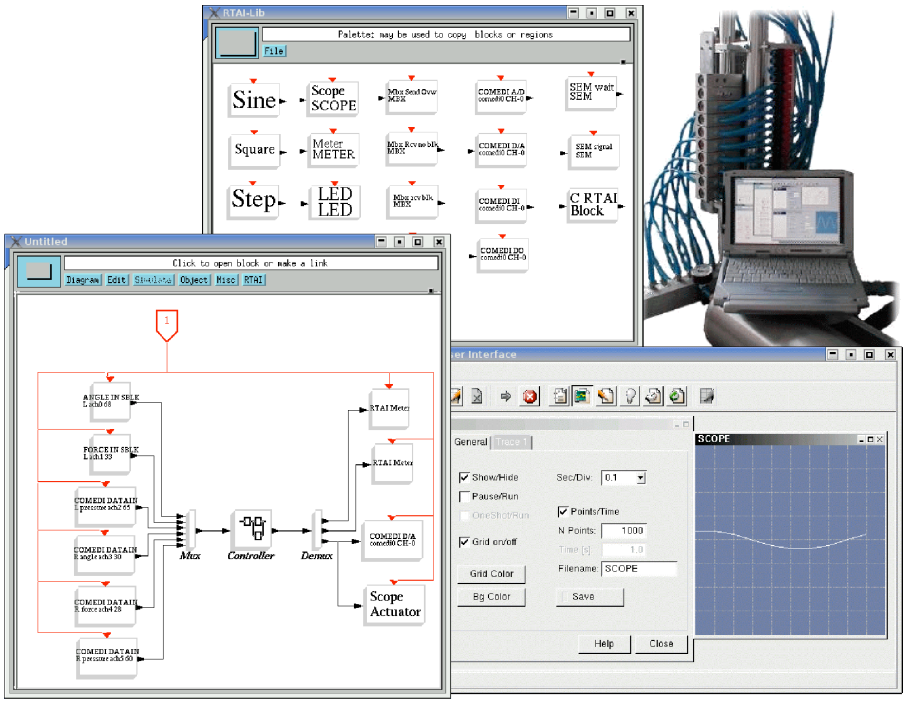

Figura 4.5.: RTAI-Lab

Questo insieme di strumenti comprende diversi moduli software, tra cui:

Scilab/Scicos. Scilab [8] è un software CACSD open-source per il calcolo numerico. Include Scicos [9], un editor di diagrammi a blocchi che può essere usato per creare simulazioni e generare automaticamente il codice eseguibile.

Comedi. Comedi [10] fornisce i driver, le funzioni di libreria, e un set di API[3] per interagire con l'hardware usato per l'acquisizione di segnali.

RTAI. Il software RTAI (Real-Time Application Interface) è distribuito come un pacchetto contenente una patch da applicare al kernel Linux. RTAI inserisce un sub-kernel dove è possibile eseguire task real-time aventi differenti priorità.

[3] Application Programming Interface

RTAI-Lib. RTAI-Lib è una *palette* formata da blocchi Scicos che permettono di disegnare diagrammi a blocchi contenenti sensori e attuatori, fornendo l'interfaccia fra RTAI e l'hardware di acquisizione dei segnali. Gli schemi che usano RTAI-Lib possono essere quindi compilati come codice eseguibile in RTAI. È inclusa nel pacchetto RTAI.

xrtailab. xrtailab è un software simile ad un oscilloscopio che può essere connesso agli eseguibili real-time. Permette di visualizzare segnali ed eventi attraverso indicatori a barra, oscilloscopi e LED, e di modificare i parametri del codice durante la sua esecuzione. Fa anch'esso parte di RTAI.

4.2.1. RTAI

Il progetto RTAI, Real-Time Application Interface, è nato nel 1998 all'interno del DIAPM[4] ed è un'estensione real-time del kernel Linux. Consiste principalmente in due parti:

1. una patch al kernel Linux che introduce un hardware abstraction layer
2. un'ampia varietà di servizi che rendono agevole la programmazione in tempo reale

Grazie a questo codice un normale PC desktop può essere utilizzato come sistema real-time azzerando i costi del software e riducendo notevolmente quelli dell'hardware, che non necessita così di particolari configurazioni.

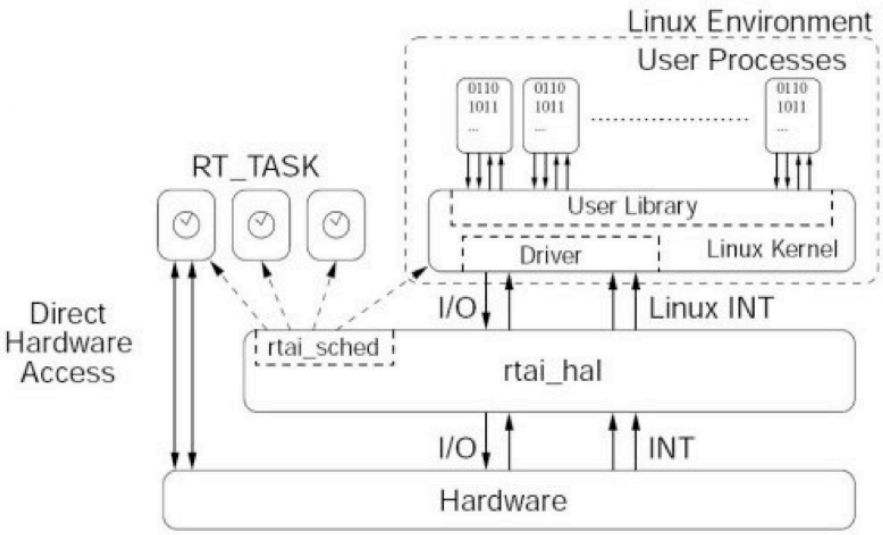

Figura 4.6.: Struttura progetto RTAI

[4]Dipartimento di Ingegneria Aerospaziale del Politecnico di Milano

RTAI non viene fornito come un programma stand-alone, ma è appunto una patch al kernel Linux. Questo significa che per essere utilizzato si deve prima avere un computer dotato di una distribuzione GNU-Linux sulla quale aggiungere successivamente RTAI. Si è scelto di usare la distribuzione Ubuntu [11] per la sua facilità di installazione e configurazione, nonché per la quantità di documentazione disponibile online.

L'installazione di Ubuntu avviene con il metodo classico pensato dai produttori della distribuzione, ossia tramite live CD. Per quanto riguarda RTAI, per i passaggi standard si è seguita la procedura indicata nel documento *RTAI-KubuntuJaunty-ScicosLab-Qrtailab.txt* [12], che copre la maggior parte degli aspetti di configurazione di un sistema RTAI completo. Altri passaggi, quali l'installazione di componenti aggiuntivi specifici, come ad esempio la scheda CAN, o problemi riscontrati con l'hardware a disposizione, sono stati affrontati avvalendosi del supporto delle case produttrici (Peak System) o dei relativi forum (RTAI, COMEDI).

Questo lavoro di ricerca ha portato alla revisione e ampliamento del documento utilizzato, culminato con la creazione di un'ulteriore guida che viene riportata in appendice.

4.2.2. Scilab

Scilab è una piattaforma interattiva per il calcolo numerico che fornisce un potente ambiente di sviluppo per applicazioni di tipo scientifico e ingegneristico. Include centinaia di funzioni matematiche, un linguaggio di programmazione ad alto livello e capacità grafiche 2-D e 3-D.

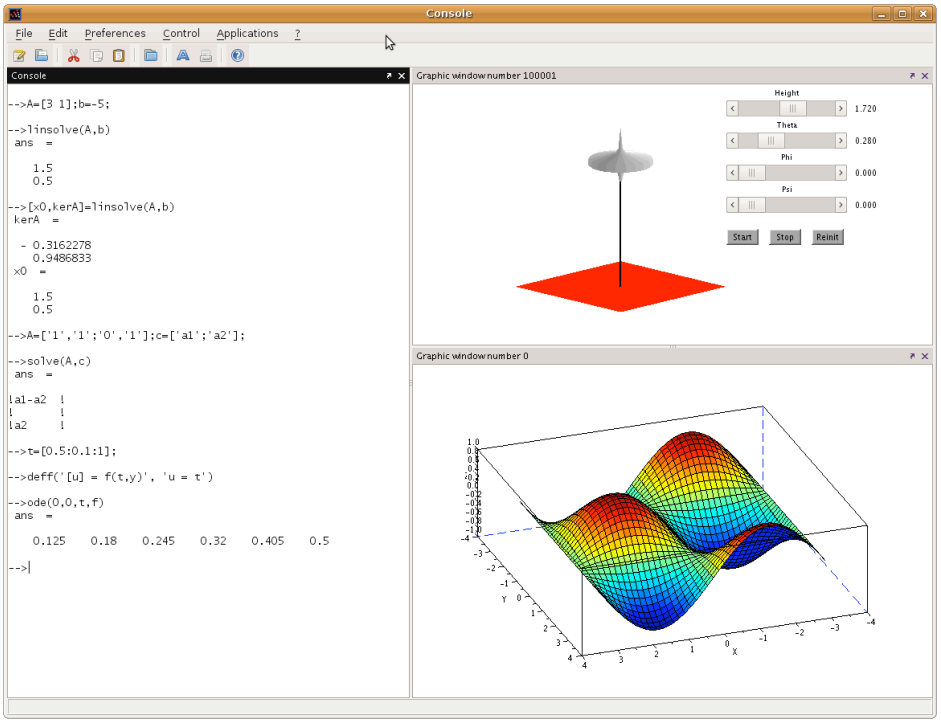

Figura 4.7.: Scilab

Scilab è stato sviluppato dalla INRIA[5] agli inizi degli anni '90, ed include Scicos, progetto maggiormente utilizzato durante questo lavoro di tesi. Il primo è considerato l'equivalente di Matlab in ambito open-source, mentre il secondo è paragonabile a Simulink.

[5](French National Institute for Research in Computer Science and Control

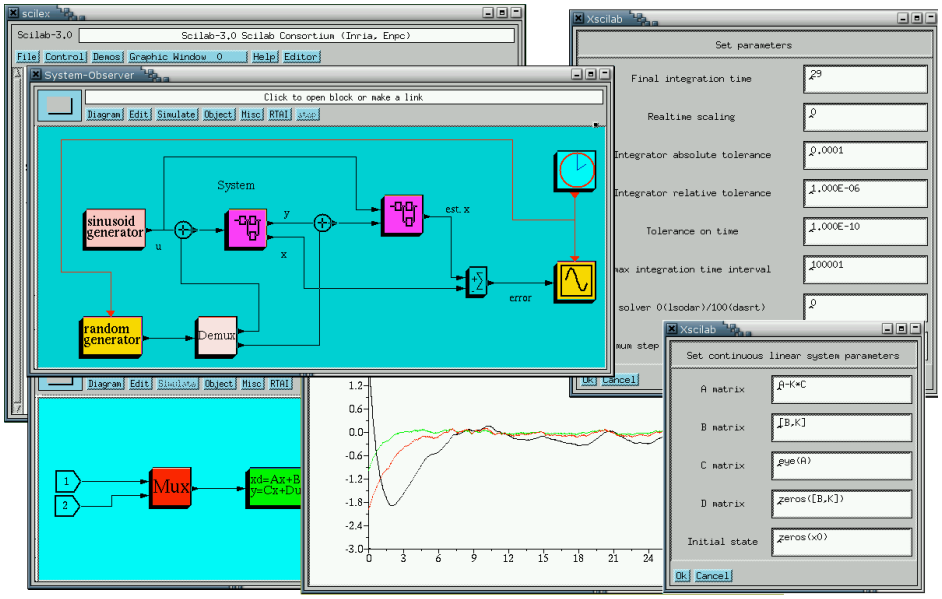

Figura 4.8.: Scicos

L'ambiente Scicos mette a disposizione diversi blocchi che possono essere usati per creare uno schema funzionale del sistema di controllo e simularlo in ambito numerico.

Creazione blocchi personalizzati

Oltre ai blocchi già presenti, è possibile crearne di personalizzati utilizzando il linguaggio C, come spiegato in [13]. Tale operazione risulterà indispensabile, dovendo realizzare l'interfaccia di comunicazione con le centraline su CAN bus.

Tutti i blocchi in Scicos sono costituiti da due funzioni:

1. la interfacing function
2. la computational function

La prima, che deve essere in linguaggio Scilab, gestisce l'interazione con l'editor di Scicos e definisce le geometria e il tipo di blocco, quante porte di input e output deve avere, ecc. Definisce inoltre l'interfaccia con l'utente, in particolare l'aggiornamento dei parametri e l'inizializzazione degli stati.

La seconda funzione controlla il comportamento del blocco durante la simulazione. Essa contiene le operazioni che devono essere valutate ad ogni istante temporale e gli eventuali ingressi e uscite da leggere e scrivere.

Integrazione con RTAI

L'integrazione con RTAI è data dalla *palette* RTAI-Lib, la quale fornisce dei blocchi aggiuntivi che consentono di realizzare schemi contenenti sensori e attuatori che possono essere compilati generando un eseguibile real-time.

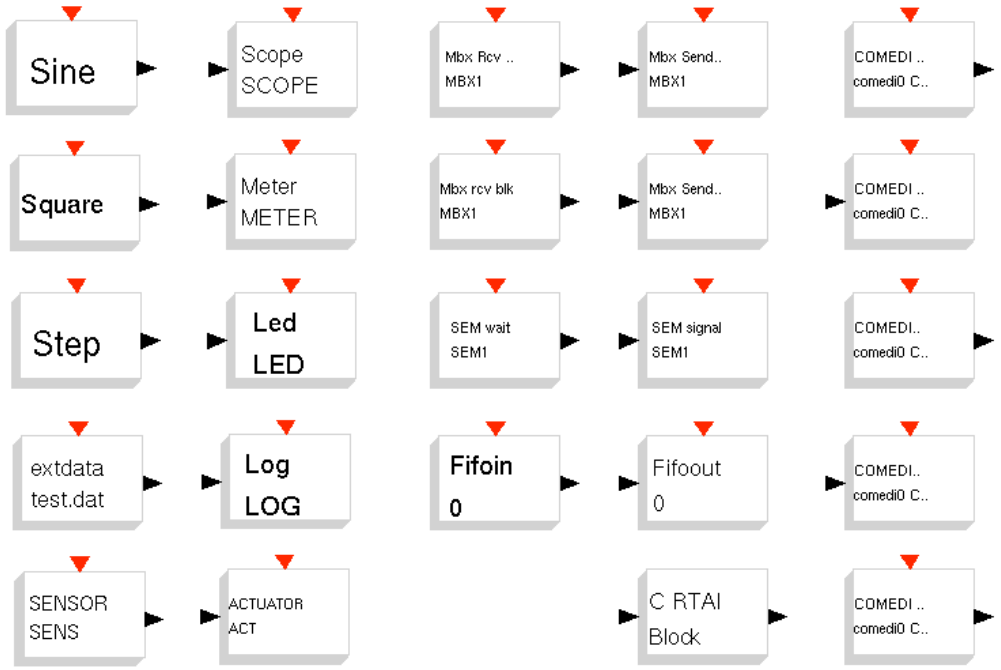

Figura 4.9.: Palette RTAI

All'interno di questi blocchi sono presenti diverse funzionalità, organizzate per colonne, quali:

Inputs: generatori di funzione (seno, onda quadra, scalino), lettura da file, sensori generici programmabili

Outputs: display (oscilloscopio, indicatori a barra, LED), attuatori generici programmabili

Mailboxes: invio messaggi, ricezione non bloccante, ricezione incondizionata, invio messaggi vincolato ai limiti temporali del task, blocchi FIFO

Comedi hardware drivers: ingressi analogici A/D, uscite analogiche D/A, input e output digitali

Semaphores and more: blocchi di attesa di segnali, blocchi generici programmabili

La dotazione di base fornita da RTAI-Lib è abbastanza completa e permette di realizzare una buona parte della logica di controllo. Quello che manca è l'integrazione e la comunicazione attraverso il CAN bus, indispensabile per leggere e comandare gli attuatori.

4.3. CAN e CANopen

Il *Controller Area Network* [14], noto anche come CAN-bus, è uno standard seriale per bus di campo usato principalmente in ambiente *automotive*, di tipo multicast, introdotto negli anni ottanta dalla Robert Bosch GmbH per collegare diverse unità di controllo elettronico (ECU). È un bus robusto, progettato per lavorare in ambienti fortemente disturbati, e con un bit rate che può raggiungere 1 Mbit/s per reti lunghe meno di 40 m.

CANopen [15] è un protocollo standard internazionale (EN 50325-4) di alto livello basato sul CAN, che viene normalmente utilizzato sui sistemi di controllo integrato. Il set delle specifiche CANopen comprende il livello delle applicazioni e il profilo di comunicazione, così come le configurazioni dei dispositivi, dell'interfaccia e degli applicativi. Il protocollo fornisce capacità di configurazioni molto flessibili, e le sue specifiche sono sviluppate e mantenute dalla Cia[6].

Questo tipo di bus presenta due caratteristiche fondamentali: innanzitutto permette di avere una comunicazione real-time, in quanto definisce un messaggio di SYNC che consente di sincronizzare e forzare in tempi stabiliti l'invio e la ricezione dei messaggi. In secondo luogo collega più dispositivi a un unico bus, riducendo il numero di cavi e semplificando le connessioni sul modello.

4.3.1. Servizi CANopen

CANopen fornisce diversi oggetti di comunicazione, che permettono al progettista di implementare nel dispositivo il comportamento desiderato all'interno della rete. Attraverso questi strumenti si creano elementi che possono comunicare dati relativi a un processo, indicare errori e status interni o influenzare e controllare il comportamento della rete.

Vengono riportati di seguito i dettagli relativi al protocollo CANopen.

Bit timing

La seguente tabella mostra il bit timing CANopen e la conseguente massima lunghezza della rete.

[6] CAN in Automation

Tabella 4.3.: CANopen bit timing

Bit rate	Bus length
1 Mbit/s	25 m
800 kbit/s	50 m
500 kbit/s	100 m
250 kbit/s	250 m
125 kbit/s	500 m
50 kbit/s	1000 m
20 kbit/s	2500 m
10 kbit/s	5000 m

Tutti i dispositivi CANopen devono supportare almeno uno di questi bit rate. Le centraline utilizzate nel modello sono state testate e utilizzate alla velocità di 500 kbit/s.

Error control protocol

I servizi *Guarding* o *Heartbeat* sono usati per verificare la presenza nella rete dei dispositivi e verificare che stiano funzionando correttamente. La specifica CANopen richiede che almeno uno di questi due protocolli sia supportato. Normalmente il servizio Heartbeat è il più utilizzato, essendo questo il più flessibile dei due e potendo funzionare senza remote frames (RTR).

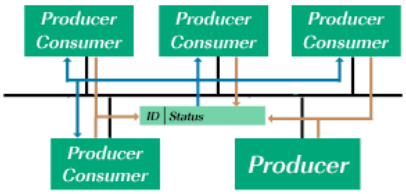

Figura 4.10.: Heartbeat protocol

Network management (NMT)

Tutti i dispositivi CANopen devono supportare il *CANopen network management (NMT)*. Questo servizio è una macchina agli stati che definisce le modalità di comunicazione del sistema ed è costituita dalle fasi denominate Initialization, Pre-operational, Operational e Stopped. Dopo un power-on o un Reset il dispositivo entra nello stato Initialization.

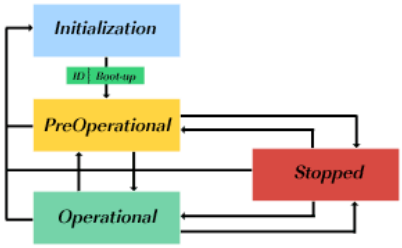

Figura 4.11.: CANopen NMT slave state machine

Una volta terminata la fase di inizializzazione, si passa automaticamente alla modalità Pre-operational, indicata dall'emissione di un messaggio di boot-up. In questa fase è possibile trasmettere messaggi di tipo SYNC, Time Stamp o Heartbeat, così come comunicare attraverso gli SDO; i PDO invece sono disabilitati e possono essere utilizzati solo nello stato Operational. Infine, una volta entrato nello stato di Stopped il dispositivo reagisce solo ai comandi di tipo NMT, ai quali risponde tramite il supporto del protocollo di controllo degli errori.

Service data objects (SDO)

I *Service data objects (SDO)* abilitano l'accesso a tutte le voci contenute in un dizionario di oggetti CANopen. Un SDO consiste in due CAN data frame con differenti identificativi, e tramite questo servizio è possibile stabilire una comunicazione di tipo client-server tra due dispositivi.

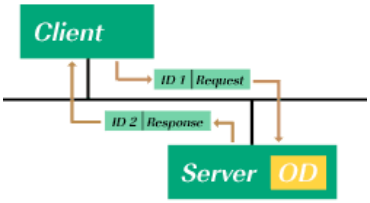

Figura 4.12.: SDO communication

Inoltre il protocollo SDO consente di trasferire una quantità di dati qualsiasi in modo frammentato, motivo per cui è principalmente usato nelle attività di configurazione.

Process data objects (PDOs)

I *Process data objects (PDOs)* sono brevi messaggi CAN ad alta priorità che vengono inviati in broadcast. Risultano quindi ideali per trasmettere dati in real-time, come ad esempio informazioni sullo stato di un dispositivo o di un modulo di I/O, valori misurati da sensori, ecc. Bisogna comunque tenere presente che i PDO sono

spediti in una modalità senza conferma, ossia non esiste un segnale che attesti che l'informazione è stata ricevuta da uno specifico partecipante della rete.

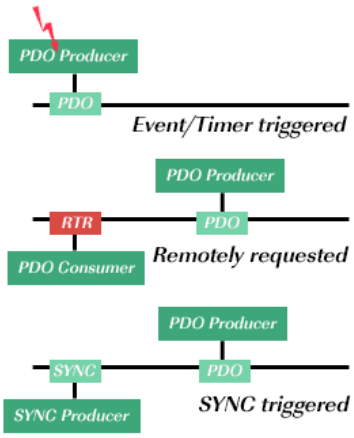

Figura 4.13.: PDO communication

La trasmissione di un PDO può essere attivata da diversi eventi: in questo progetto viene usata la modalità sincrona, che garantisce una comunicazione di tipo real-time, in quanto le trasmissioni vengono attivate una volta ricevuto il messaggio di Sync.

Synchronisation object (SYNC)

Il *Synchronisation object (SYNC)* è inviato periodicamente dal SYNC producer. Il tempo compreso tra due messaggi consecutivi è il periodo di comunicazione sul bus. In questo progetto il SYNC producer è il codice real-time, che alla frequenza fissata attiva la trasmissione e rende così predicibile l'invio e la recezione dei messaggi.

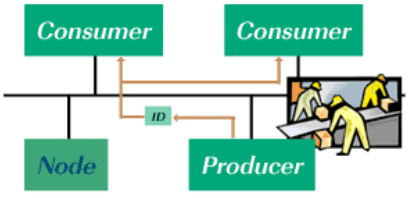

Figura 4.14.: SYNC object

4.3.2. CANopen in RTAI

La comunicazione su bus CAN e tramite protocollo CANopen è stato implementata in RTAI dall'ingegnere Roberto Bucher, professore all'università svizzera SUPSI.

Egli ha integrato le funzionalità del bus in RTAI realizzando una libreria scritta in C, denominata *canopen.c* [16], la quale:

- inizializza il driver RTDM[7]
- contiene una funzione per la registrazione dei Tx PDO
- avvia un thread ad alta priorità per la ricezione dei messaggi CAN
- gestisce i Tx PDO1, PDO2 e PDO3, con la possibilità di aggiungere ulteriori PDO

Questa libreria contiene inoltre una funzione che genera il messaggio di SYNC e lo gestisce in modo tale che ad ogni invio i dispositivi aventi dei Tx PDO reagiscono fornendo i valori richiesti al master, mentre quelli con Rx PDO impostano l'ultimo comando ricevuto.

La lettura del bus CAN avviene tramite un thread real-time schedulato come FIFO e avente alta priorità. La procedura di ricezione viene gestita utilizzando gli interrupt forniti dal driver RTDM, mentre i messaggi PDO sono salvati all'interno di opportune matrici, i cui valori vengono successivamente estratti tramite specifiche procedure.

Questo lavoro è stato tradotto nella creazione di blocchi per Scicos che consentono di utilizzarne le funzionalità all'interno di Scilab. Questi elementi sono stati raccolti in una *palette* denominata SUPSI-Lib che consente di disegnare schemi come quello proposto in figura, dove viene evidenziata la possibilità di variare il valore dei parametri.

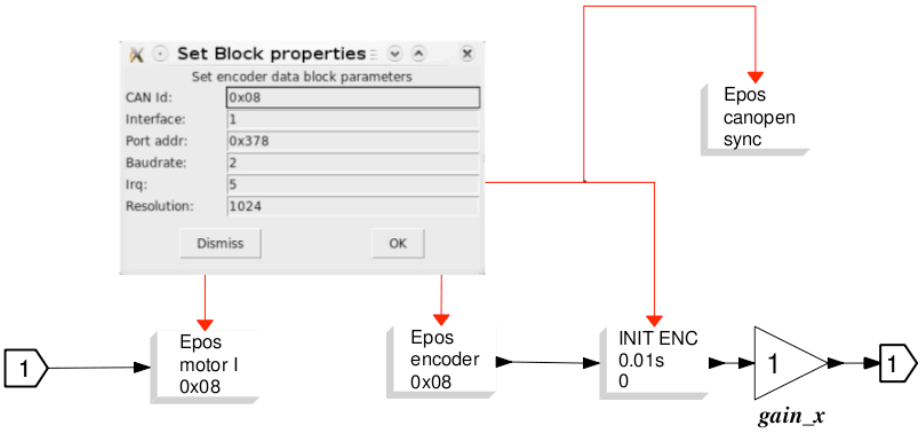

Figura 4.15.: Schema Scicos con blocchi della libreria SUPSI-Lib

I parametri dei blocchi sono quindi modificabili, potendo così essere adattati a differenti esigenze, quali bitrate o numerazione dei nodi della rete.

Andando ad analizzare la libreria si trovano gli oggetti riportati in figura.

[7] *Real Time Driver Model*, funziona da interfaccia tra l'applicazione che richiede un certo servizio a un device e il driver del device stesso.

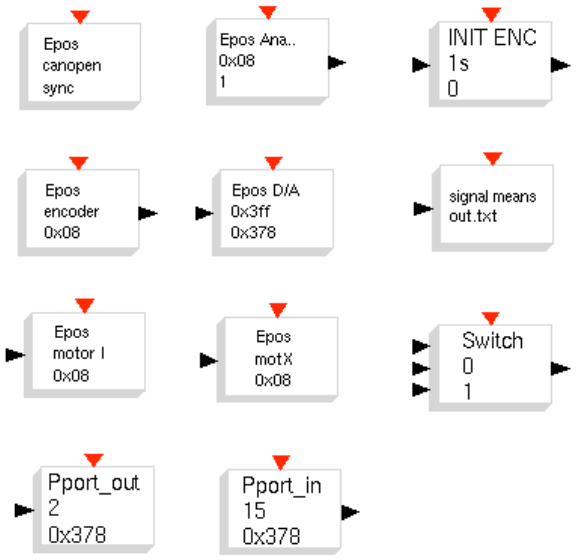

Figura 4.16.: Blocchi delle libreria SUPSI-Lib

Tra questi elementi si riconoscono i blocchi di input e output di vari dispositivi CANopen, come ad esempio le centraline EPOS, ma non di quelli utilizzati in questo progetto. Per poter comunicare con questi oggetti si è quindi reso necessario creare dei nuovi elementi, che contengono le istruzione specifiche delle singole centraline utilizzate.

4.4. Configurazione centraline CAN

Ogni centralina CANopen dispone di un dizionario di oggetti contenente i comandi che è possibile ricevere, che risulta diverso a seconda del tipo di dispositivo e della casa produttrice. Non solo il contenuto, ma anche le modalità di accesso e configurazione sono differenti, potendo essere rigide e fissate di default oppure modificabili e impostabili dall'utente.

La comunicazione via CAN permette quindi di modificare parametri della centralina, (costante di coppia del motore, limiti di velocità), assegnare i comandi da eseguire (setpoint di velocità e posizione) e leggere i dati di interesse (posizione raggiunta, messaggi di errore).

La parte di configurazione avviene normalmente tramite l'invio di SDO, mentre per la gestione del processo si utilizzano i PDO, essendo quest'ultimi gestibili in modalità real-time. In entrambi i casi, la comunicazione consiste nell'invio al dispositivo di un messaggio in formato esadecimale, sempre costituito da due elementi:

1. il COB-ID, l'indirizzo del dispositivo, ossia l'identificativo del nodo nella rete
2. il messaggio vero e proprio, formato da un massimo di 8 byte, contenente il tipo e il valore dei parametri da assegnare o leggere

Tabella 4.4.: Struttura messaggi PDO

COB-ID	**Messaggio esadecimale**
11 bit	8 byte

Si riportano le modalità di configurazione e comunicazione delle due diverse centraline Faulhaber e ARTDriveS.

4.4.1. Faulhaber MCDC 3003 C

Canali PDO

La centralina Faulhaber MCDC 3003 C dispone di 3 servizi PDO e 1 SDO, il cui identificativo è dato da un valore numerico standard a cui va sommato l'ID del nodo.

Tabella 4.5.: Canali PDO Faulhaber

PDO	**Tipo di PDO**	**Identificativo a 11 bit**
RxPDO1	Controlword	0x200 + Node-ID
TxPDO1	Statusword	0x180 + Node-ID
RxPDO2	FAULHABER command	0x300 + Node-ID
TxPDO2	FAULHABER data	0x280 + Node-ID
RxPDO3	Trace configuration	0x400 + Node-ID
TxPDO3	Trace data	0x380 + Node-ID

Il numero del nodo deve essere un valore unico all'interno della rete, in modo tale da potersi riferire a una data centralina in modo univoco.

La configurazione scelta è tale da utilizzare i canali PDO2 e PDO3: tramite il RxPDO2 si inviano i comandi con le impostazioni e i setpoint di posizione, mentre sul TxPDO3 si leggono i valori impostati con il RxPDO3.

Parametri e messaggi

Una volta impostato il COB-ID e il baudrate di una centralina, gli altri parametri da configurare riguardano le caratteristiche del motore e la struttura del canale TxPDO3.

I valori sono trasmessi con messaggi in modalità FAULHABER, una modalità di comunicazione semplificata dove la lunghezza del frame è di 5 byte e il primo byte indica il parametro da modificare, mentre gli altri 4 il valore da assegnare.

Nella seguente tabella sono riportati i diversi messaggi utilizzati per modificare le proprietà della centralina e i valori specifici relativi al materiale utilizzato.

Tabella 4.6.: Configurazione parametri motore Faulhaber

Parametro motore	**Parametro HEX [1 byte]**	**Valore [4 byte]**
Speed constant [V/rpm]	0x9E	840
Motor resistance [mOhm]	0x9F	19800
Maximum speed [rpm]	0x8F	5787
Peak current limit [mA]	0x81	593
Continuos current limit [mA]	0x80	320
Velocity proportional term	0x89	40
Velocity integral term	0x7B	150
Position proportional term	0x9B	220
Position differential term	0x9C	5

Sia il parametro motore che il suo valore sono inseriti nel messaggio in formato esadecimale. Per comodità di gestione, il valore di quest'ultimo è riportato in formato decimale, sarà poi il software ad occuparsi della conversione in base 16.

Il canale TxPDO3 viene invece configurato in modo diverso, ossia inviando un singolo messaggio attraverso il RxPDO3 contente la struttura e il tipo di dati che si vogliono leggere. È anch'esso un messaggio di 5 byte, dove però ciascun byte ha una funzionalità particolare, riportata nella seguente tabella.

Tabella 4.7.: Struttura messaggio RxPDO3

Byte	**Funzione**	**Significato**
0	Parametro 1	vedi tab 4.8
1	Parametro 2	vedi tab 4.8
2	Trasmissione con time code	1 = con time code 0 = senza time code
3	Numero di pacchetti trasmessi per richiesta	default = 1
4	Intervallo di tempo fra i pacchetti [ms]	default = 1

È possibile avere la trasmissione di uno o due parametri, a seconda del valore contenuto nel byte 0 e nel byte 1. Le possibilità sono:

Tabella 4.8.: Configurazione canale Trace

Parametro 1	Parametro 2	Significato
0	0	velocità attuale [rpm]
1	1	setpoint velocità [rpm]
2	2	uscita del controllore
4	4	corrente del motore [mA]
44	44	temperatura carcassa [°C]
46	46	temperatura bobina [°C]
200	200	posizione attuale [Inc]
201	201	setpoint posizione [Inc]
	255	parametro 2 vuoto

La configurazione scelta è quella di una trasmissione contemporanea di due valori: la posizione attuale del motore e la corrente assorbita.

4.4.2. ARTDriveS

Canali PDO

La parte di comunicazione su CAN bus dell'ARTDriveS funziona in modo diverso da quello Faulhaber: si hanno sempre a disposizione 3 PDO e 1 SDO, ma la configurazione di questi canali è completamente programmabile dall'utente. Il contenuto dei PDO viene impostato tramite l'interfaccia grafica E@syDrives della Gefran, collegando il dispositivo a un PC tramite la porta seriale. Questo ha permesso di utilizzare un singolo canale, il PDO1, per effettuare tutte le operazioni di comunicazione necessarie durante l'utilizzo operativo.

La possibilità di aver un doppio collegamento, seriale e CAN, permette di programmare l'inverter da Windows con i suoi tool specifici via seriale, e successivamente utilizzarlo con RTAI via CAN.

La struttura del canale PDO1 è stata impostata come segue:

Tabella 4.9.: Struttura canale PDO1 ARTDriveS

	Parametro 1	Parametro 2	Parametro 3
RxPDO1	Virt DI Status	Speed Ref	Ramp Exp Factor
TxPDO1	Motor Speed	Out Current	Alarm Status

In questo modo si riescono a gestire tutte le informazioni fondamentali con un unico canale, con il vantaggio di dover inviare il minor numero di byte possibili e ottenere così una maggior efficienza nella comunicazione. I parametri essenziali che vengono utilizzati nella comunicazione col dispositivo sono quindi:

Tabella 4.10.: Parametri di comunicazione con ARTDriveS

Parametro	ID	Significato	Dimensione
Virt DI Status	20186	Visualizza ed imposta lo stato degli ingressi digitali virtuali.	16 bit
Speed Ref	21200	Impostazione del riferimento di velocità. [rpm]	16 bit
Ramp Exp Factor	21110	Fattore di espansione della rampa, utilizzato per incrementare il massimo valore consentito dei parametri delle rampe.	16 bit
Motor Speed	18777	Velocità del motore. [rpm]	16 bit
Out Current	18735	Corrente del motore (filtrata). [Arms]	16 bit
Alarm Status	24000	Stato degli allarmi.	32 bit

Il *Virt DI Status* viene utilizzato per avviare, arrestare e resettare l'inverter, mentre il *Ramp Exp Factor* è usato per variare la rampa di velocità che il motore segue quando si setta un nuovo riferimento. L'*Alarm Status* invece riporta le informazioni sullo stato del sistema, indicando anomalie di tipo hardware e software.

Parametri e messaggi

Tutti i parametri relativi al motore sono stati impostati ai valori di ottimo dai fornitori dell'insieme inverter più motore tramite l'interfaccia grafica collegata per via seriale. Non è quindi necessario prevedere nessun altro tipo di comunicazione, se non quella descritta nel paragrafo precedente.

4.5. Libreria POLIMI-Lib

Una volta determinati tutti i parametri di configurazione e i comandi si procede all'integrazione del codice in RTAI, passando attraverso Scilab.

Nel capitolo 4.3.2 si è vista la libreria SUPSI-Lib, contenente i blocchi per comunicare con le periferiche EPOS. Il lavoro svolto è stato simile: si sono creati i blocchi di comunicazione con le centraline Faulhaber e ARTDriveS e si è unito il risultato in una libreria per Scilab denominata POLIMI-Lib.

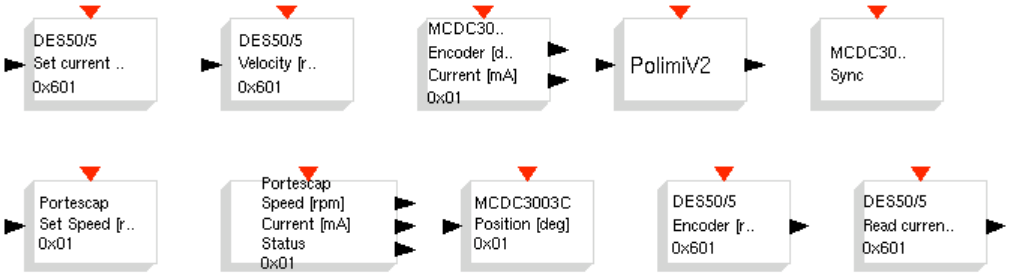

Figura 4.17.: Blocchi delle libreria POLIMI-Lib

Le *computational functions* di questi blocchi sono basate sul lavoro di Roberto Bucher, e sfruttano la libreria canopen.c per inviare i messaggi sul bus CAN. La loro struttura è pressoché identica per tutti i blocchi e consiste in tre funzioni fondamentali:

init: contiene il codice da eseguire durante la fase di inizializzazione dell'applicativo real-time

inout: contiene le istruzioni da eseguire nel task real-time su come manipolare gli ingressi e le uscite del blocco

end: contiene le operazione per la fase di arresto

Nella fase di *init* si sono inseriti quei comandi che agiscono sulle macchine agli stati delle centraline e le portano in modalità operational. Nel caso delle centraline Faulhaber qui si trovano anche i comandi che impostano i valori dei parametri motore e il canale Trace.

Nella funzione di *inout* invece si ha la lettura dei valori dal dispositivo o l'invio di comandi tramite messaggi CAN o entrambi. Questa fase viene ripetuta ciclicamente durante l'esecuzione dell'applicativo real-time con l'intervallo di tempo fissato all'interno di Scilab.

Il codice contenuto in *end* viene invece eseguito all'arresto del programma, disattivando o resettando le centraline e chiudendo tutti i thread precedentemente avviati.

4.5.1. Blocchi Faulhaber

Nella figura 4.17 si nota la presenza di 3 blocchi Faulhaber per l'azionamento della centralina MCDC3003C:

- MCDC3003C Position
- MCDC3003C Encoder

- MCDC3003C Sync

Il loro scopo consiste nel comandare il funzionamento dei motori di passo pala attraverso la centralina di controllo MCDC3003C, la quale riceve in ingresso i comandi inviati dal controllo real-time, esegue il posizionamento del motore e fornisce in uscita le informazioni di passo e corrente.

MCDC3003C Position

Figura 4.18.: MCDC3003C Position

Il blocco MCDC3003C Position presenta una sola porta di input attraverso la quale entra il riferimento di posizione. durante la normale esecuzione dell'eseguibile real-time. Nella computational function questo valore viene letto, convertito da gradi a numero di impulsi dell'encoder e successivamente inviato alla centralina per essere impostato.

All'attivazione del blocco si eseguono le operazioni di inizializzazione della centralina stessa (attivazione del nodo CAN), si definisce lo zero dell'encoder e si settano i parametri motore, quali la costante di velocità, la resistenza, la corrente massima, ecc.

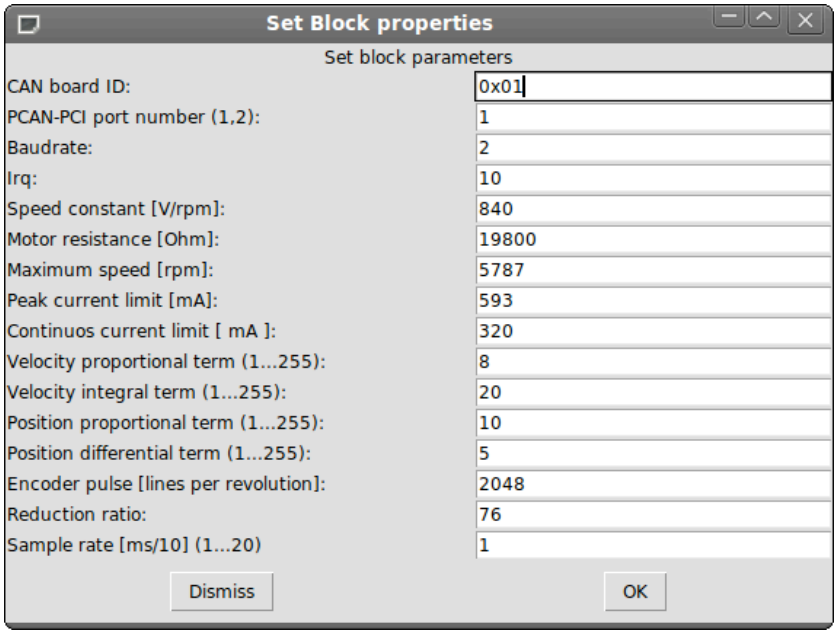

Figura 4.19.: MCDC3003C Position: parametri

Le prime quattro proprietà del blocco, riportate in figura 4.19 impostano rispettivamente: il numero del nodo della centralina, la porta di comunicazione della scheda CAN PCI presente nel computer, la velocità di trasmissione dati del bus e il valore di interrupt.

Gli elementi successivi riguardano i parametri motore, i guadagni dei controllori PI e PD, il numero di impulsi dell'encoder, il rapporto di riduzione tra albero motore e carico e infine il sample rate di funzionamento.

MCDC3003C Encoder

Figura 4.20.: MCDC3003C Encoder

Questo blocco fornisce due output: la posizione del motore, espressa come rotazione in gradi, e la corrente assorbita in mA. Nella fase di init del blocco si definisce la configurazione del canale RxPDO3 Trace, impostandolo in modo da avere la trasmissione di queste due dati. Il dato di corrente viene utilizzato per determinare

il fondo corsa della pala, come spiegato nella descrizione dello Stato 1 nel capitolo 3.2, e per verificare che non ci siano picchi o anomalie nel funzionamento.

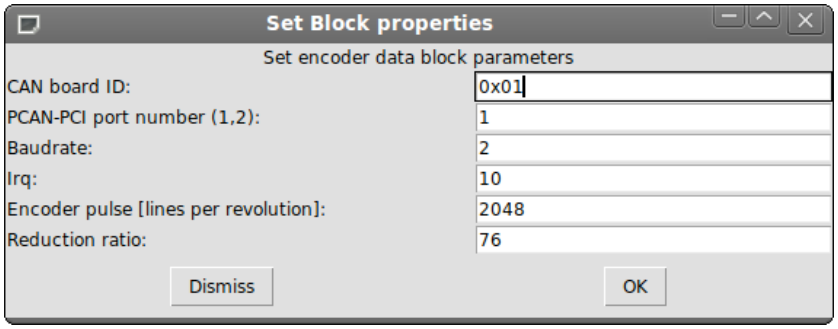

Figura 4.21.: MCDC3003C Encoder: parametri

Nella finestra dei parametri si ritrovano le proprietà relative alla comunicazione sul CAN bus, mentre non ci sono i settaggi del motore. Restano modificabili la risoluzione dell'encoder e il rapporto di riduzione.

MCDC3003C Sync

Figura 4.22.: MCDC3003C Sync

Il messaggio di SYNC è stato implementato per la centralina MCDC, ma è valido anche per l'ARTDriveS. Il blocco viene attivato attraverso l'orologio del codice real-time ed invia in broadcast il messaggio di SYNC, garantendo così l'invio e la ricezione nei tempi stabiliti di informazioni e comandi.

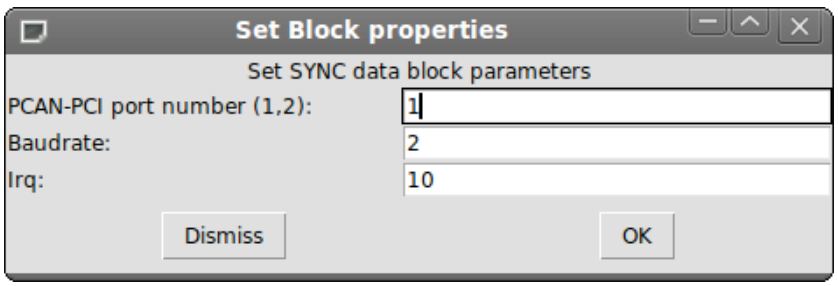

Figura 4.23.: MCDC3003C Sync: parametri

Essendo un messaggio diretto a tutte le unità CAN presenti sul bus, gli unici parametri da impostare sono quelli relativi alla porta della scheda PCI e al baudrate utilizzati.

4.5.2. Blocchi Portescap

Riferendosi sempre alla figura 4.17 si evidenziano i seguenti blocchi per la gestione del motore Portescap:

- Portescap Set Speed
- Portescap Speed

I due elementi impostano e leggono la velocità di rotazione del motore, riportano la corrente assorbita dal motore e la variabile status, che indica gli errori riscontrati dalla centralina.

Portescap Set Speed

Figura 4.24.: Portescap Set Speed

Il blocco presenta un solo ingresso: il riferimento di velocità, che viene aggiornato durante l'esecuzione del codice. Questo valore, riferito al rotore, viene convertito e riferito al motore, tenendo conto delle riduzioni applicate e della coppia conica presenti tra i due elementi.

Nella parte di init computational function si trova la procedura di attivazione del nodo, che porta la macchina agli stati interna alla centralina in modalità *operational*, mentre all'arresto del sistema il drive viene completamente disattivato: in

questo modo si ha la certezza di poter eventualmente ispezionare l'aerogeneratore in tutta sicurezza.

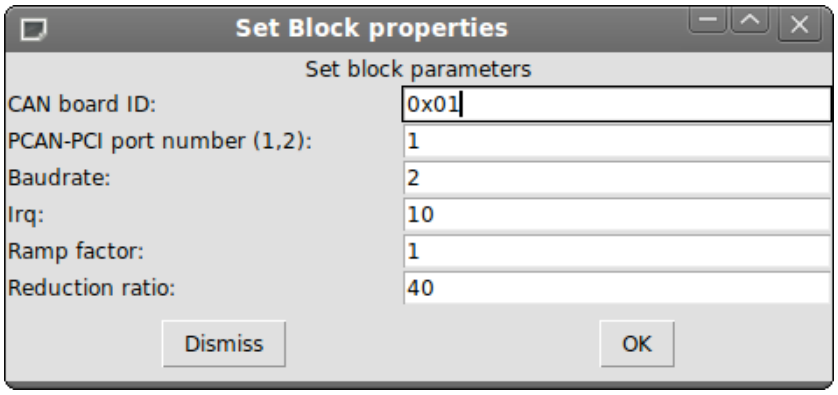

Figura 4.25.: Portescap Set Speed: parametri

Oltre ai consueti parametri relativi al CAN, nella scheda delle proprietà del blocco è possibile impostare il *Ramp Factor*, un guadagno che modifica l'ampiezza della rampa di accelerazione, e il *Reduction ratio*, il rapporto di riduzione tra albero motore e albero rotore.

Portescap Speed

Figura 4.26.: Portescap Speed

Le uscite del blocco di lettura sono 3, rispettivamente:

- la velocità di rotazione del rotore, in rpm
- la corrente assorbita dal motore, in mA
- lo stato della centralina

In questo caso tutte le impostazioni relative al motore sono già state eseguite attraverso il software E@syDrives della Gefran, quindi il codice risulta più semplice di quello contenuto nei blocchi Faulhaber. Le sue funzioni si limitano all'attivazione del drive e alla gestione della propria velocità.

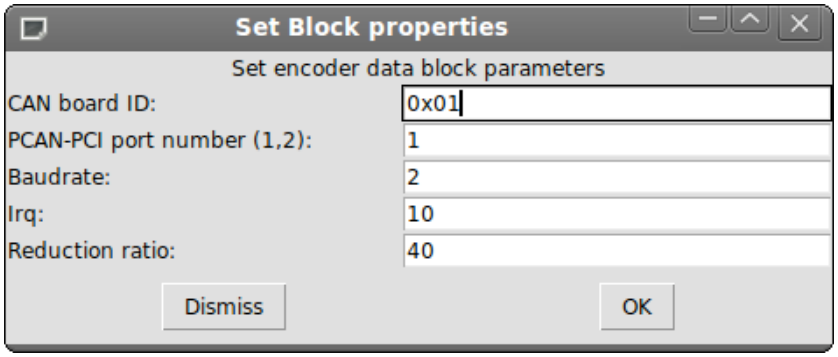

Figura 4.27.: Portescap Speed: parametri

Per quanto riguarda le proprietà del blocco queste sono uguali a quelle usate nel Portescap Set Speed, con l'unica differenza di aver rimosso il *Ramp Factor*, inutile in fase di lettura.

4.5.3. Blocchi Maxon Motor

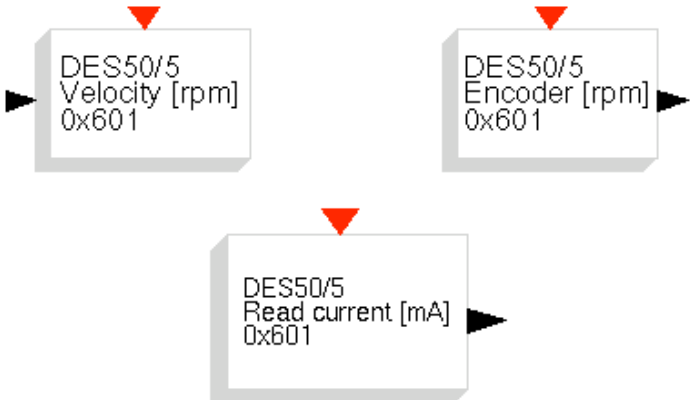

Figura 4.28.: Portescap Speed

Nella libreria sono presenti anche dei blocchi di interfaccia con la centralina DES 50/5 della Maxon Motor, inizialmente utilizzata in fase di progettazione in sostituzione dell'ARTDriveS. Attualmente non sono più utilizzati, ma in caso di guasto al motore Portescap potrebbero facilmente sostituire quest'ultimo. Il sistema risulterebbe limitato nelle operazioni, in quanto il motore Maxon presenta delle prestazioni più limitate, ma si avrebbe comunque un sistema funzionante e in grado di eseguire buona parte dei test in tempi rapidi.

4.6. Sviluppo macchina agli stati in RTAI

L'implementazione della macchina agli stati all'interno di Scilab non è stata un'operazione banale: come già accennato all'inizio del capitolo 3, questo strumento non consente di realizzare logiche troppo complesse per quel che riguarda l'interazione con l'utente. La prima problematica che si è presentata è stata quella di garantire il corretto percorso tra gli stati: per questioni di sicurezza l'operatore deve infatti poter passare dallo stato di *init* a quello di *parking*, ad esempio, ma non può saltare dall'*idle* all'*automatic operation*. Realizzare questa logica di controllo utilizzando i blocchi messi a disposizione da Scicos si è rivelata un'operazione piuttosto complessa.

Si ha inoltre che le leggi di controllo sviluppate in Dipartimento sono già state codificate all'interno dei software di simulazione in linguaggio C. Scilab offre la possibilità di realizzare dei blocchi la cui computational function sia scritta in questo linguaggio, possibilità che risulta naturale sfruttare nello stato 5 del sistema per testare le leggi di controllo.

Alla luce di queste due considerazioni si è scelto di costruire un sistema dove tutta la logica di funzionamento è contenuta in un unico elemento, un blocco denominato *PolimiV2* e codificato in linguaggio C, il quale comunica con sensori e attuatori attraverso le funzionalità già implementate nella POLIMI-Lib.

Il vantaggio di questo approccio consiste nella possibilità di recuperare la potenza e flessibilità della programmazione in C per nella progettazione del funzionamento del sistema, mantenendo allo stesso tempo la comodità di poter generare un eseguibile real-time che si interfacci automaticamente con xrtailab.

Blocco PolimiV2

Il blocco *PolimiV2* è quindi costituito da una computational function scritta in C e da una interfacing function scritta in linguaggio Scilab. Quest'ultima serve a disegnare gli ingressi e le uscite del blocco, ed è progettata in modo da poterne modificare il numero in modo rapido all'interno di Scilab, mentre attraverso la computational function si procede successivamente ad assegnare a ciascun canale il proprio significato numerico.

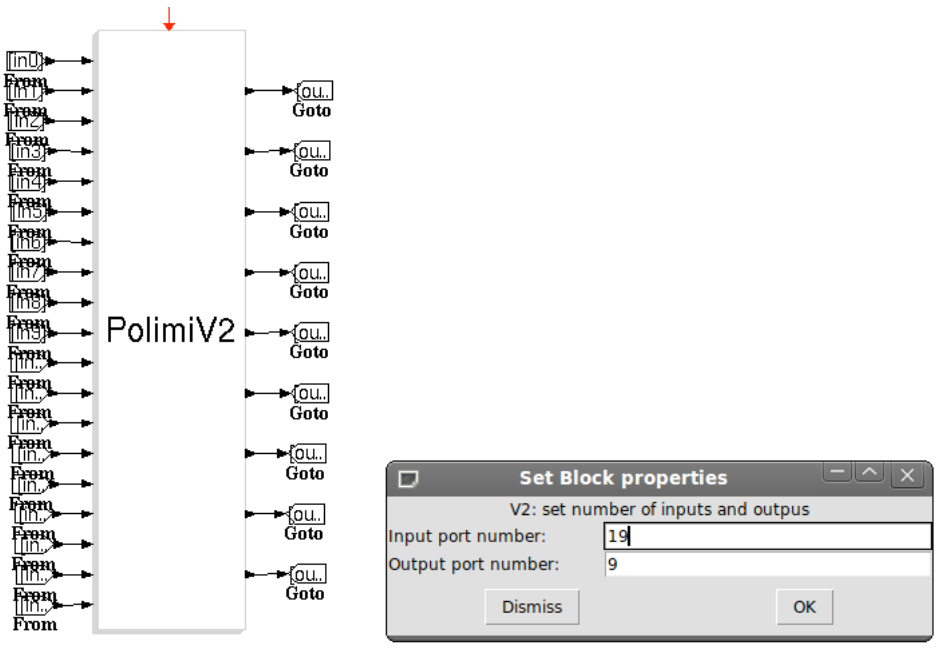

Figura 4.29.: Blocco PolimiV2 in Scilab

La logica di funzionamento del blocco è quella di un ciclo infinito: ad ogni attivazione comandata dal clock real-time si ha l'esecuzione del codice, le cui operazioni sono:

gestione della macchina agli stati: determina il passaggio da uno stato all'altro, verificando la possibilità di effettuare la transizione in funzione del comando dell'utente.

stato 0: disabilita i drive di passo e di coppia, lasciando il sistema inattivo e libero di muoversi.

stato 1: attiva tutti drive ed effettua i test preliminari, azzera l'encoder tramite la procedura definita nel capitolo 3.2, determina e rimuove l'offset dai ponti estensimetrici del torsiometro.

stato 2: porta il sistema nello stato di parcheggio, diminuendo la velocità del rotore fino all'arresto e portando le pale in bandiera.

stato 3: aumenta la velocità di rotazione e varia il passo pala fino ai valori preimpostati per l'avvio della macchina.

stato 4: legge i comandi di velocità e posizione immessi dall'utente e comanda i drive di conseguenza.

stato 5: opera l'aerogeneratore elaborando ingressi e uscite tramite le leggi di controllo implementate.

L'interazione con l'utente è invece affidata a dei segnali visti come ingressi dal blocco, il cui valore numerico viene cambiato attraverso l'interfaccia xrtailab. L'esempio più immediato di questa funzionalità è la macchina agli stati: a seconda del valore dell'ingresso denominato "state" si ha il passaggio da una modalità operativa all'altra, passaggio regolato dalla logica contenuta nel blocco che permette solo i percorsi descritti nel capitolo 3.2.

Questo sistema si inserisce quindi all'interno di un superblocco, dove la funzione PolimiV2 è nella parte centrale, collegata tramite ingressi e uscite ai blocchi della libreria POLIMI-Lib che la interfacciano con l'esterno.

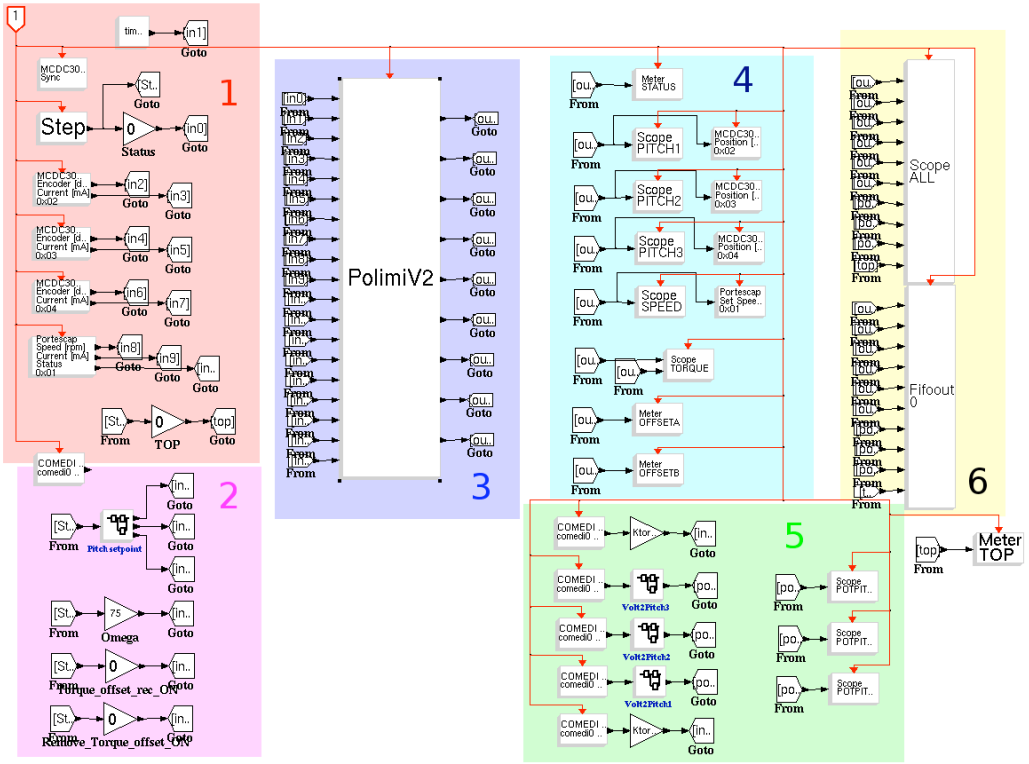

Figura 4.30.: Superblocco RTAI

A sinistra si trovano i blocchi di lettura di posizione e velocità degli attuatori, dei canali della scheda di acquisizione e dei comandi inviati sotto forma di segnale numerico attraverso xrtailab. A destra ci sono invece i comandi per gli azionamenti, le uscite a video delle misure acquisite e la loro memorizzazione.

Nel dettaglio, analizzando il superblocco per aree funzionali, si ha:

Area 1: lettura segnali attuatori si evidenziano in figura i 3 blocchi per la lettura degli encoder delle 3 pale, quello per la lettura della velocità del rotore, il blocco di Sync e quello per attivare la transizione della macchina agli stati.

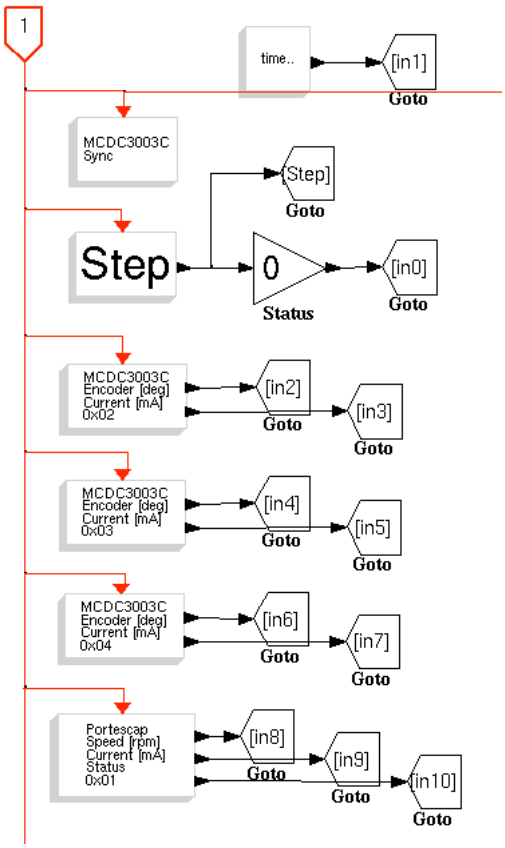

Figura 4.31.: Superblocco: lettura segnali attuatori

Area 2: acquisizione comandi in questa zona ci sono i comandi di passo pala, contenuti in un sottosistema utilizzato per azionare le pale in modo sincrono o indipendente, il comando di velocità di rotazione e quelli per la registrazione e rimozione dell'offset del torsiometro.

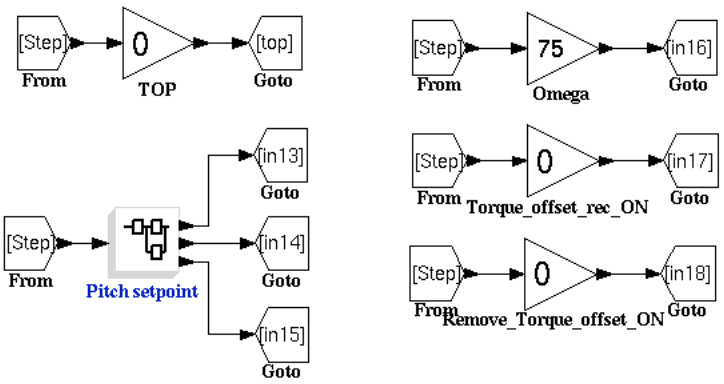

Figura 4.32.: Superblocco: acquisizione comandi

Area 3: elaborazione codice V2 questo è il blocco V2 precedentemente descritto, che attraverso i suoi input e output gestisce il comportamento del modello.

Area 4: invio comandi e visualizzazione qui il codice invia i comandi di passo e pala e velocità alle centraline e si ha la visualizzazione delle variabili misurate, come la coppia e i suoi offset.

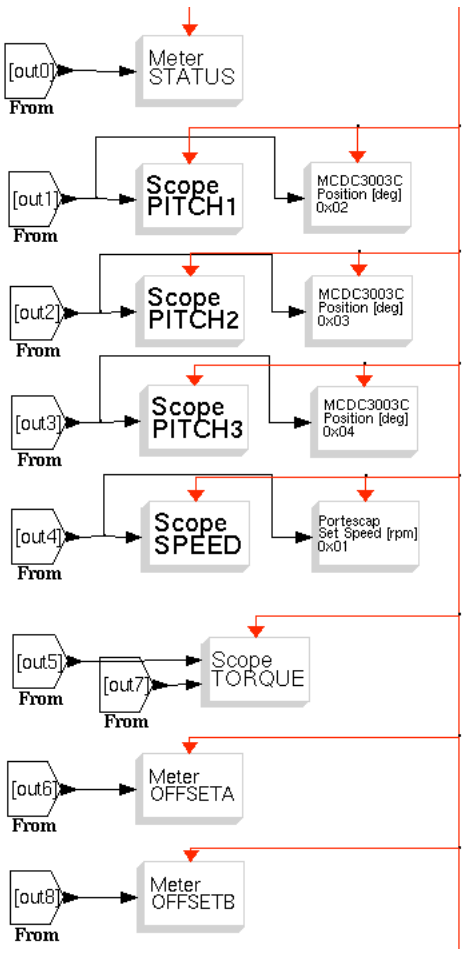

Figura 4.33.: Superblocco: invio comandi e visualizzazione

Area 5: acquisizione segnali analogici questi sono i blocchi COMEDI della libreria RTAI-Lib che leggono i segnali in ingresso alla scheda di acquisizione analogica, tra cui i valori di coppia dei due ponti del torsiometro e quelli di passo pala forniti dalle strisce potenziometriche.

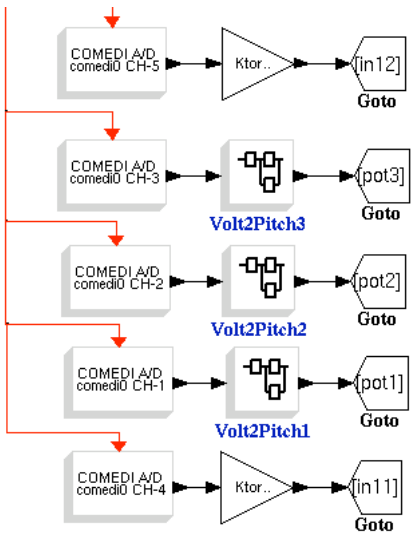

Figura 4.34.: Superblocco: acquisizione segnali analogici

Area 6: memorizzazione segnali la memorizzazione dei segnali avviene attraverso due sistemi: uno SCOPE che viene attivato tramite QRtaiLab e registra i valori sul computer monitor, e una FIFO che, attraverso un eseguibile esterno, memorizza i valori in locale sul controllore.

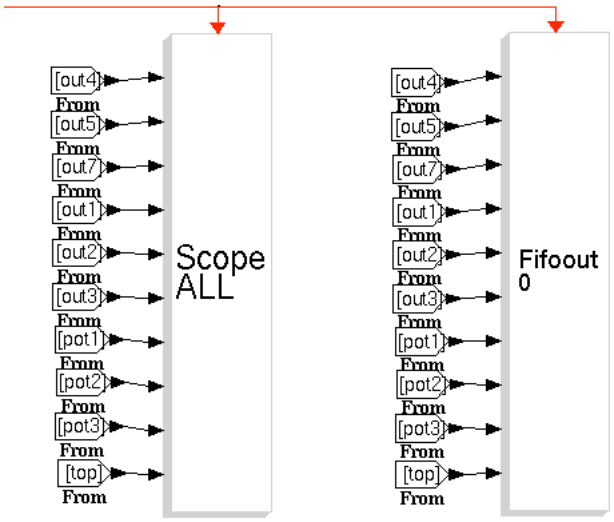

Figura 4.35.: Superblocco: memorizzazione segnali

5. Sistema di controllo Bachmann

Bachmann è un'azienda altamente tecnologica fondata nel 1970, il cui scopo è offrire soluzioni complete nel campo dell'automazione. La combinazione di tecnologie informatiche di ultima generazione, elettronica industriale e un'estensiva conoscenza dei processi industriali ne fanno un'azienda leader per soluzioni di automazione all'avanguardia.

I suoi prodotti hardware e software vengono utilizzati in diversi settori, quali:

- automotive
- biogas plants
- hydroelectric power plants
- laboratory measurement technology
- water power
- ship controls

e ovviamente il settore della produzione di energia eolica.

I leader internazionali in questo settore scelgono l'Automation System della Bachmann come elemento centrale per il controllo e l'interconnessione degli impianti eolici. Con un market share di oltre il 40% e l'utilizzo in più di 20000 aerogeneratori, la Bachmann è il leader mondiale nel settore dell'automazione per l'energia eolica. Il sistema si compone essenzialmente di due parti: il controllore hardware, *M1 controller*, e la suite software, l'applicativo *Solution Center*.

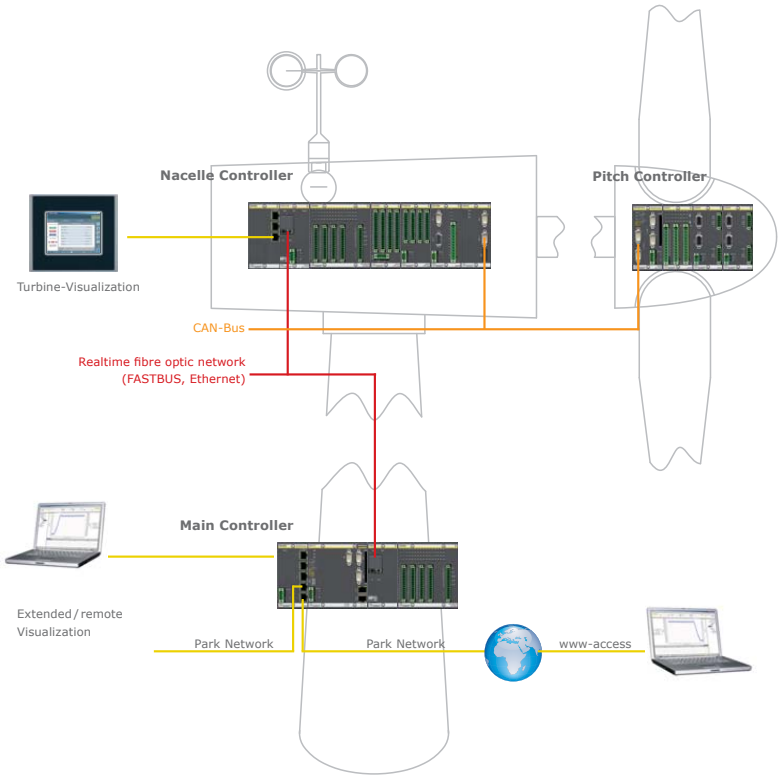

Figura 5.1.: Wind energy plants

Durante lo sviluppo di questo elaborato, l'autore si è recato 4 giorni a Feldkirch, in Austria, per seguire un corso base sul funzionamento e sviluppo di applicativi relativi al sistema Bachmann.

Si procede a un'analisi della struttura hardware, per identificare i componenti necessari all'utilizzo col modello in galleria del vento, e del software, evidenziando le differenze di approccio al problema della gestione della macchina rispetto a quanto sviluppato con RTAI.

5.1. M1 controller

Il controller M1 è un sistema PC-compatible basato su processori e chip-set di tipo industriale; è compatto, modulare e facilmente intercambiabile. Lavora con differenti tipi di bus, tra cui il CAN bus, e presenta tutti i moduli montati su slitte standard.

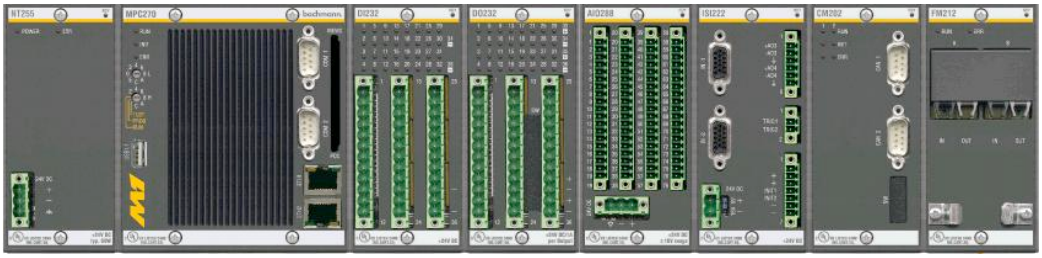

Figura 5.2.: M1 controller

I programmi possono essere copiati ed eseguiti su supporti removibili come le PC-Card o le Compact Flash.

5.1.1. Configurazione hardware

La struttura hardware è completamente modulare: le diverse funzioni sono rappresentate attraverso singoli componenti che vengono montati su delle slitte standard, le quali permettono la comunicazione con il modulo principale costituito dal controllore.

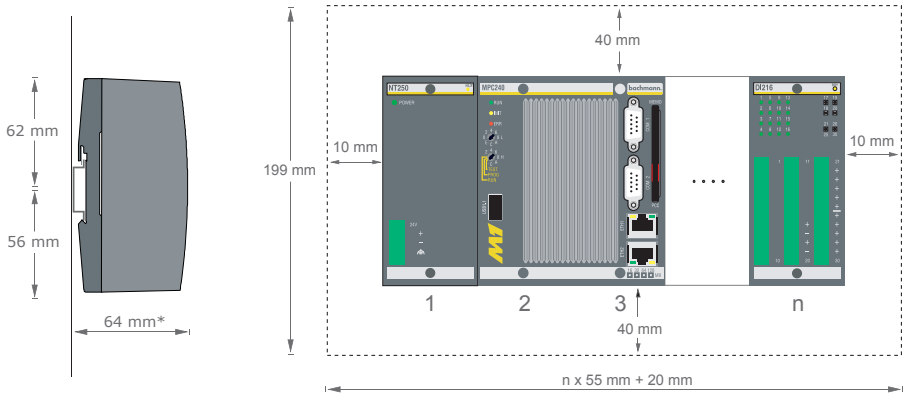

Figura 5.3.: Struttura modulare M1

Questi moduli sono di dimensioni standard e facilmente intercambiabili, opzione che permette di aggiornare o modificare rapidamente il controllore in caso di avaria di una sua singola parte.

Dallo studio del catalogo si è stilata la seguente lista di componenti:

Tabella 5.1.: Configurazione M1

Modulo	Tipologia	Descrizione
MPC270	Processor	CPU 700Mhz, SRAM 512 Kb, 1x USB, 1x RS232, 1x RS232/422/485, 2x Ethernet 10/100 Mbit/s
AIO288	Analog I/O	Number of inputs: 8 (±10 V DC / ±1 V DC or 0 .. 20 mA), Resolution: 14 bits, Galvanic isolation from system: 500 V
CM202	CAN bus	2 independent CAN buses per CANopen master module, Up to 64 nodes per CAN bus, Transfer rate: 10 Kbit/s up to 1 Mbit/s
NT255	Power supply	Power supply of modules via backplane, Supply voltage: 18 .. 34 V DC, Galvanic isolation

Il primo modulo è quello di base: contiene il controllore, un sistema simile a un PC le cui caratteristiche però sono basate su componenti di qualità standard industriale. Si ha quindi un modulo per le connessione analogiche, 8 ingressi per ciascun elemento; si è scelto di comprarne almeno 4 in previsione delle modifiche future al modello e al relativo aumento di segnali da acquisire. Il modulo successivo serve per interfacciarsi con CAN bus e comunicare con gli attuatori, mentre l'ultimo elemento fornisce l'alimentazione a tutto il sistema.

Il numero di moduli che si possono aggiungere è molto elevato e questo permette di progettare una struttura iniziale di base, dove inserire successivamente in modo semplice eventuali moduli e funzioni richieste dallo sviluppo del modello.

5.2. Solution Center

Il pacchetto software fornito con il sistema Bachmann contiene il programma *Solution Center*, una suite creata per gestire tutti gli aspetti di sviluppo sul controllore, dalla configurazione dei moduli hardware fino alla creazione degli applicativi.

Figura 5.4.: Solution Center

Questa soluzione consente di avere un unico strumento per controllare e sviluppare tutti gli aspetti del progetto: configurazione delle centraline, sviluppo della logica di controllo, sviluppo dell'applicativo grafico di monitoraggio, esecuzione di test e debugging.

5.2.1. Configurazione moduli hardware

Una funzionalità importante inclusa nel *Solution Center* è quella del *Device Manager*. Questo software offre infatti la possibilità di configurare in modo rapido ed efficiente sia i moduli hardware direttamente collegati al sistema, sia le periferiche che vi comunicano attraverso i bus. In questo lavoro, nel capitolo 4, si sono configurati i canali PDO e la comunicazione col CAN in modo manuale, settando i parametri da inviare lungo il bus e i relativi canali; il sistema Bachmann invece imposta in modo automatico queste proprietà, scegliendole da una lista generata dai file di configurazioni inclusi nelle centraline.

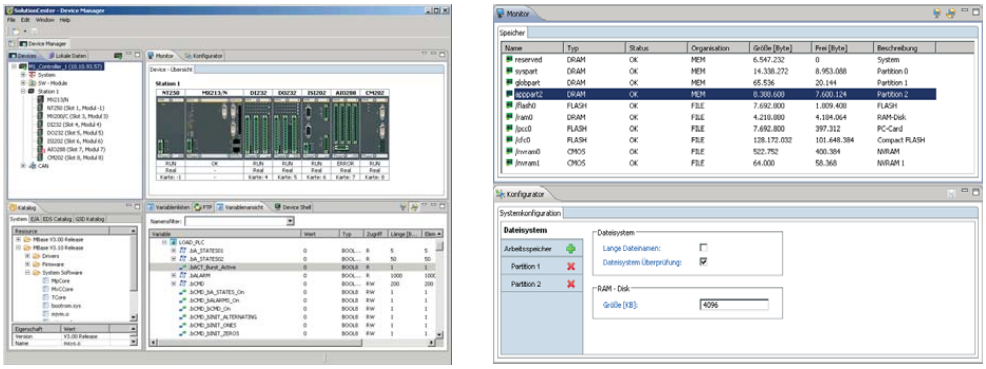

Figura 5.5.: Device Manager

Questo permette di collegare e operare nuovi dispositivi in modo rapido e, allo stesso tempo, evita al programmatore di concentrarsi su aspetti tecnici della gestione delle centraline, che risultano troppo di basso livello e tolgono tempo allo sviluppo del progetto.

5.2.2. Programmazione in C e PLC

La gestione degli applicativi sul controllore è gestita a tre livelli: c'è il livello più alto che è quello delle applicazioni, poi uno strato sottostante che per la configurazione del sistema e infine il *core*, dove si esegue il nucleo realtime e l'interfaccia con le periferiche.

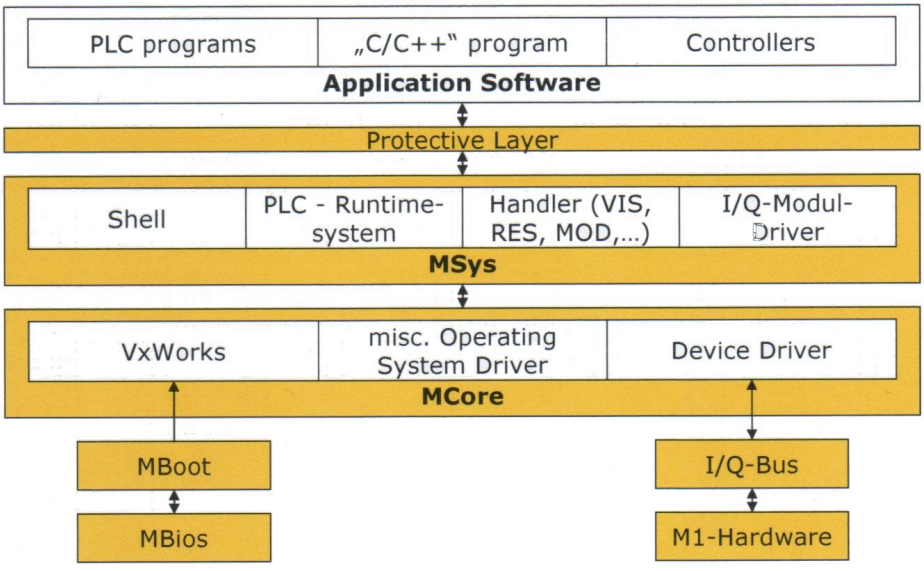

Figura 5.6.: Struttura software

La suite include anche un ambiente di programmazione per lo sviluppo degli applicativi software, che possono essere generati attraverso due differenti linguaggi: C e PLC, un linguaggio specializzato in origine nella gestione dei processi industriali.

La programmazione in C è quella usata comunemente anche all'interno del DIA e offre alcune vantaggi rispetto a quella PLC, tra i quali: un accesso più facile alle funzioni di sistema e del kernel, performance migliori e possibilità di creare strutture software molto sofisticate.

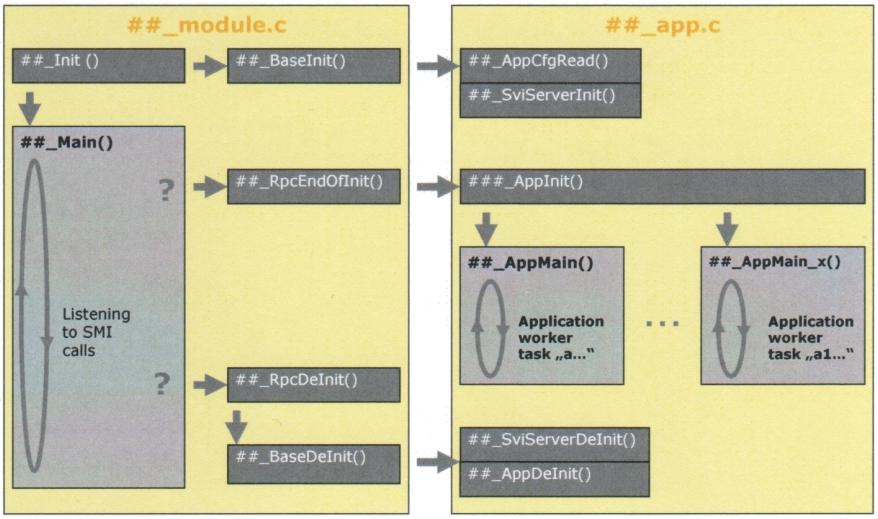

Figura 5.7.: Struttura codice C

La scrittura di codice in PLC invece risulta più semplice, e quindi meno potente, ma offre performance migliori per quanto riguarda la comunicazione fra i processi e assicura automaticamente la consistenza dei dati in fase di lettura e scrittura.

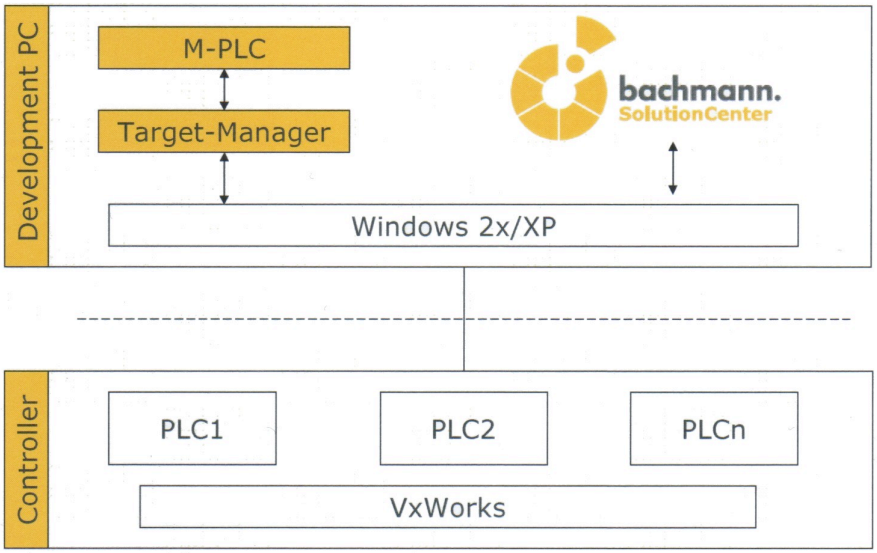

Figura 5.8.: Struttura codice PLC

5.2.3. Sviluppo interfacce grafiche con M-JVIS Designer

A differenza di RTAI-Lab, il sistema non comprende un'applicazione completa per la visualizzazione dei segnali come QRtaiLab. Si ha però a disposizione un software avanzato di nome *M-JVIS Designer*, che consente il disegno di interfacce grafiche avanzate in modo rapido e con ridotta scrittura di codice.

Per le parti più semplici, come i bottoni, lo sviluppo è completamente grafico, in quanto il software fornisce tutti gli strumenti necessari per gestire il comportamento degli elementi.

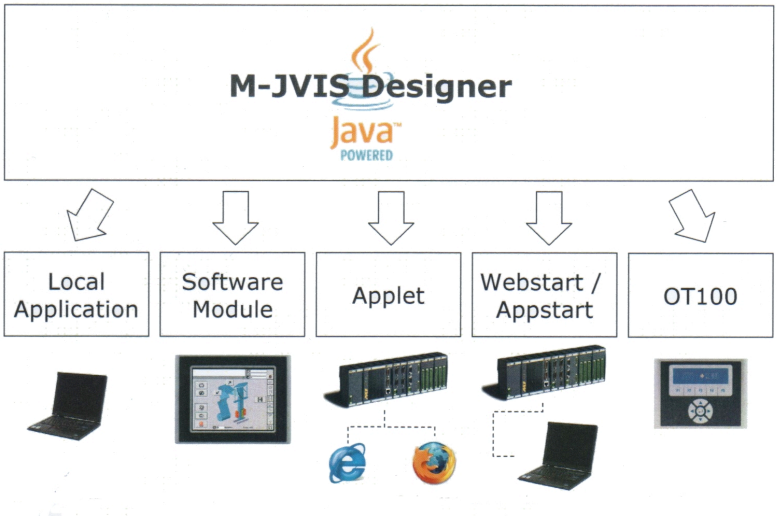

Figura 5.9.: Codice sviluppato in M-JVIS Designer

Il codice sorgente è compilato in Java, permettendo di realizzare applicativi multi piattaforma, la cui esecuzione può avvenire su PC, IPC o addirittura attraverso pagine web. In questo lavoro di tesi la scelta principale resta sempre quella di avere un'applicazione grafica installata su un normale PC che si interfacci con il controllore e ne gestisca il funzionamento, come mostrato nella seguente figura.

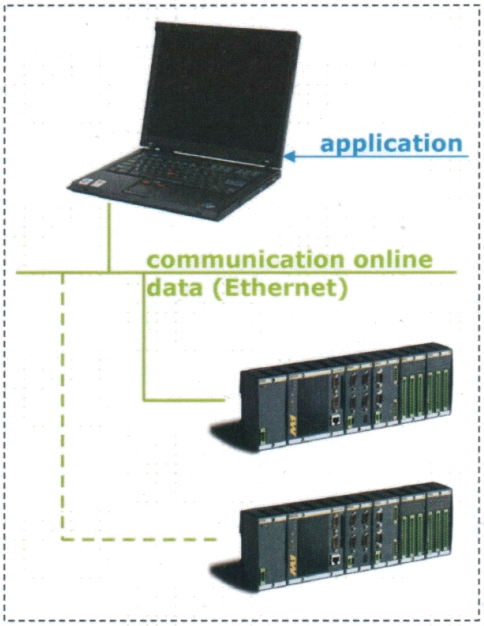

Figura 5.10.: Collegamento controllore - software di gestione

Con questo strumento è possibile quindi realizzare in tempi brevi interfacce grafiche anche piuttosto complesse, eliminando così uno dei limiti più forti di QRtaiLab.

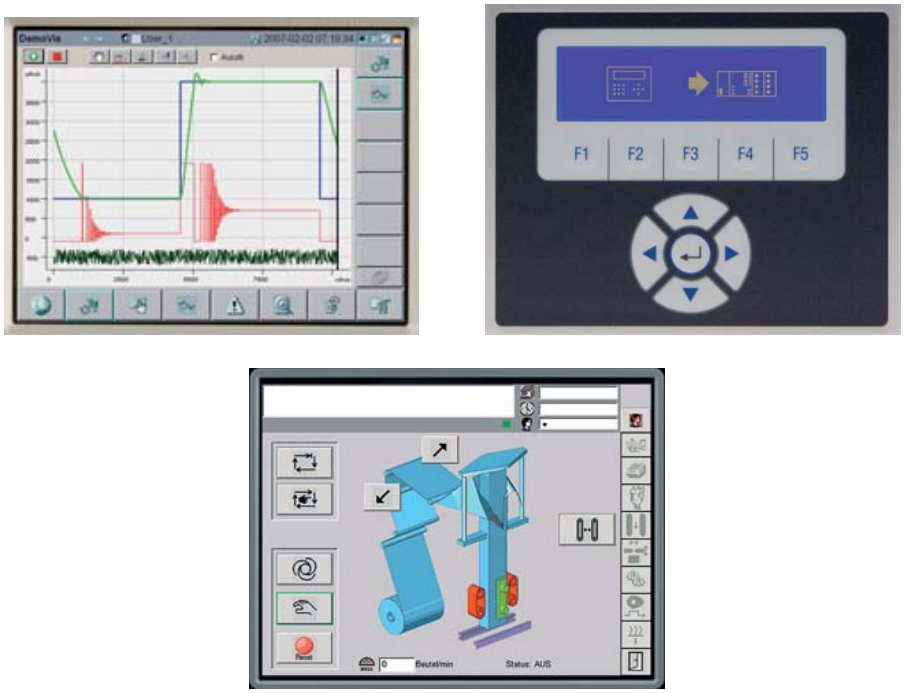

Figura 5.11.: Esempi interfacce grafiche realizzate con M-JVIS Designer

5.3. Sviluppo macchina agli stati nel sistema Bachmann

Al momento della stesura di questo lavoro di tesi non è stato possibile sviluppare in modo completo una macchina degli stati nel sistema Bachmann, in quanto il sistema stesso non era ancora disponibile. Si è eseguito quindi uno studio preliminare, basato su quanto appreso durante il corso base a Feldkirch, stendendo le linee guida del progetto e della sua codifica.

5.3.1. Studio preliminare

L'architettura che si intende realizzare prevede la presenza di più applicazioni, denominate *Software Modules* all'interno del sistema Bachmann, il cui funzionamento è gestito ancora una volta attraverso una piccola macchina agli stati.

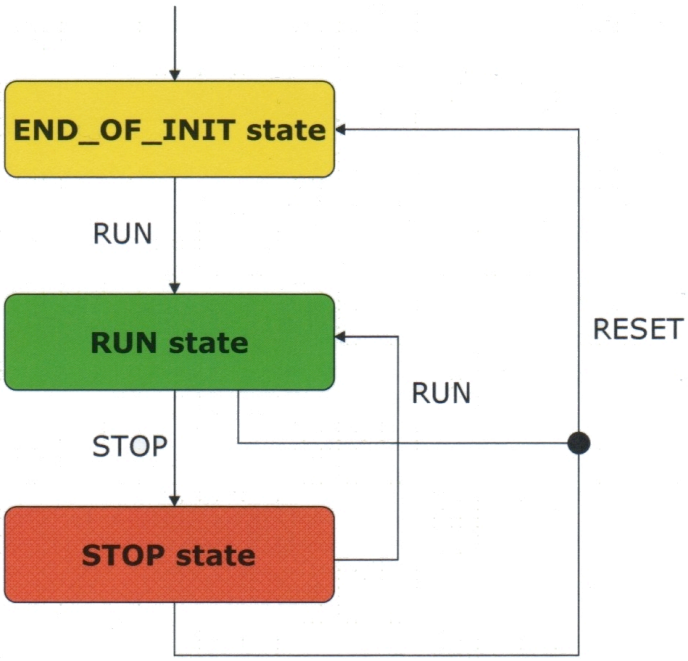

Figura 5.12.: Funzionamento modulo software

Questi moduli software si dividono in 3 categorie:

Msys contente i moduli di sistema

Applications le applicazioni scritte in linguaggio C

PLC programs le applicazioni scritte in linguaggio PLC

I moduli Msys gestiscono i diversi aspetti del sistema, tra i quali:

MIO gestione di input/output

RES gestione delle risorse di sistema

MOD gestione dei moduli software

EHD gestore degli errori

Ognuna di queste applicazioni può comunicare con le altre attraverso la propria interfaccia SVI (Standard Variable Interface) e andare così a modificare i parametri del processo.

Si intende quindi realizzare la gestione generale attraverso 3 applicazioni intercomunicanti secondo il seguente schema:

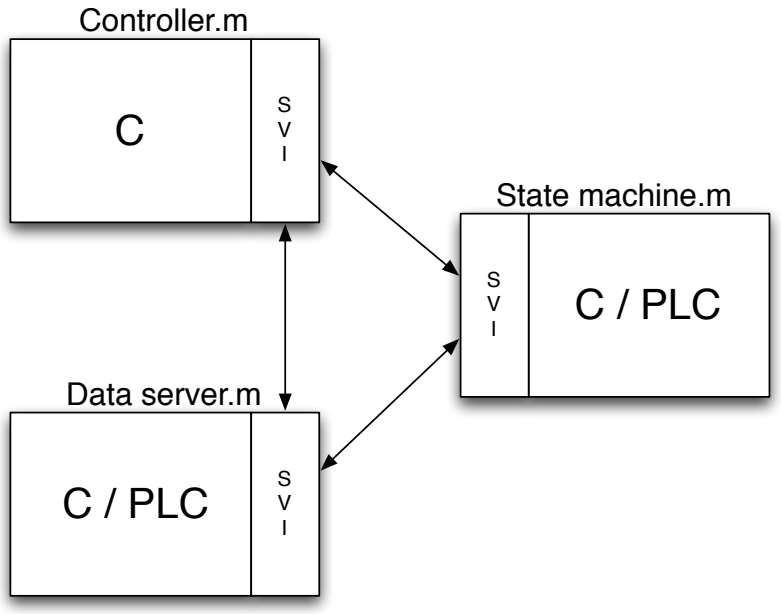

Figura 5.13.: Schema gestione sistema Bachmann

dove si ha:

- una prima applicazione, *Data Server.m*, che funziona da server, e mette a disposizione degli altri processi le variabili acquisite attraverso le funzioni di input/output
- una secondo modulo, *Controller.m*, che calcola le variabili in uscita attraverso le leggi di controllo implementate
- une terza funzione, la macchina agli stati *State machine.m*, che regola e gestisce l'attività delle altre 2.

Questo tipo di approccio modulare al problema permette lo sviluppo delle 3 applicazioni in modo separato; i vantaggi riguardano la possibilità di modificare e aggiornare le diversi parti del sistema in modo indipendente e scrivere le singole applicazioni con diversi linguaggi, sfruttando i punti di forza di entrambi.

La scelta più logica per quanto riguarda il modulo software che calcola l'azione di controllo è quella del linguaggio C, già utilizzato per la sua creazione durante le simulazioni; per la macchina agli stati e il server dei dati può essere invece più conveniente svilupparli in linguaggio PLC, sfruttandone così la velocità di scrittura e l'automatica consistenza dei dati.

In particolare la macchina agli stati risulta una funzione di questo tipo:

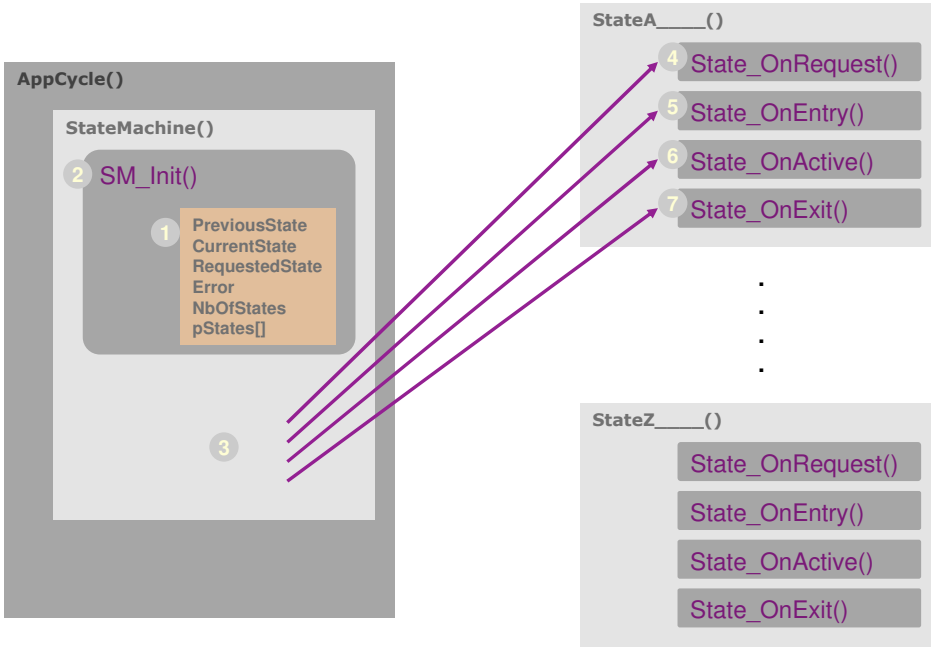

Figura 5.14.: Schema funzionamento macchina agli stati Bachmann

dove il passaggio fra gli stati è comandato dall'utente (punto 3 nella figura), controllato dalla logica implementata in (1). Il comportamento del singolo stato della macchina è definito in (4), (5) , (6) e (7), dove vengono eseguite le operazioni già definite nel capitolo 3.

5.4. Confronto con RTAI

Il confronto del sistema Bachmann con RTAI può essere fatto solo dal punto di vista funzionale, sulle modalità operative e di implementazione del codice.

Innanzitutto bisogna ricordare che RTAI è un sistema real-time opensource generico installabile su diversi tipi di PC, mentre Bachmann produce un sistema software dedicato alla gestione dei processi industriali, e in particolare quelli legati alle energie rinnovabili, e il relativo hardware. Di conseguenza possiede degli strumenti specifici che non si trovano in RTAI.

Inoltre quest'ultimo è un software opensource e gratuito, mentre il sistema Bachmann è chiuso e a pagamento e può contare su 40 anni di esperienza nello sviluppo di controllori industriali.

Dal punto di vista di configurazione hardware e software, RTAI presenta una buona comunità alla quale rivolgersi in caso di problemi nell'installazione delle varie parti, mentre Bachmann fornisce un prodotto dove queste operazioni sono facilitate e ridotte al minimo, consentendo così allo sviluppatore di concentrarsi

più sull'aspetto della logica del codice piuttosto che sulla messa a punto della macchina.

Tabella 5.2.: Confronto sistemi Bachmann e RTAI

Caratteristiche	**Bachmann**	**RTAI**
Codice sorgente	proprietario	opensource
Competenze specifiche per aerogeneratore	sì	no
Configurazione hardware	procedure semplificate	configurazione manuale con driver specifici
Comunicazione con nodi CAN bus	procedure semplificate	creazione manuale degli oggetti di comunicazione
Ambiente di sviluppo software dedicato	Control Center	nessuno
Ambiente di sviluppo interfacce grafiche dedicate	M-JVIS Designer	nessuno

Dal punto di vista del costo il codice RTAI è gratuito, quindi le note di spesa riguardano il computer e le schede di comunicazione installate. Per quanto riguarda il prodotto Bachmann invece è necessario acquistare sia l'hardware che il software, il cui prezzo è però fortemente scontato se gli acquirenti sono istituti di ricerca universitari.

Tabella 5.3.: Confronto costi sistema Bachmann e RTAI

Bachmann			**RTAI**	
	Costo generico	**Costo per università**		**Costo**
hardware	15000	2500	Computer standard	800
software	10000	gratuito	schede di comunicazione	1200
			software	gratuito
TOTALE	25000	**2500**		**2000**

Comparando la situazione dei costi si evince come l'acquisto del prodotto Bachmann comporti una spesa leggermente più elevata, ma offra un prodotto completo, facilmente gestibile e di elevata qualità.

In conclusione, confrontando i due sistemi dal punto di vista funzionale ed economico, si nota come il prodotto Bachmann, viste le sue caratteristiche specifiche ereditate dal suo utilizzo nell'ambito delle energie rinnovabili, possa essere più adatto allo sviluppo di un apparato complesso come quello in esame, fornendo strumenti

che riducono il carico di lavoro nella scrittura del codice e permettendo di creare architetture e interfacce grafiche sofisticate in modo semplice.

6. Prove in galleria del vento

Le prove effettuate in galleria del vento sono state condotte esclusivamente con il sistema basato su RTAI e utilizzato in modalità manuale; i tempi e le problematiche incontrate nella fase di test non hanno infatti permesso la valutazione della modalità automatica.

Lo scopo di questi ingressi in galleria è stato quello di verificare la funzionalità del sistema completo in termini di operatività, resistenza dei componenti e verifica della corrispondenza con i dati di progetto.

6.1. Programmazione prove

La programmazione delle prove è stata progettata in funzione dei risultati attesi, ossia il tracciamento delle curve prestazionali della macchina, in particolare le $C_P - \lambda$, $C_Q - \lambda$ e $C_T - \lambda$. Per ottenere queste curve è necessario misurare i valori di coppia sul rotore e le forze agenti alla base della struttura in funzione del TSR.

Le prove sono di tipo statico: si porta il rotore in una data configurazione di passo pala e TSR e si registrano i valori per un periodo di tempo sufficientemente lungo (circa 10 secondi); successivamente si varia il TSR, mantenendo costante il valore di passo pala, e si registrano nuovamente le informazioni di coppia e forza agenti sul rotore. Una volta eseguita la spazzata di TSR a β costante la sessione è conclusa, si cambia passo e si esegue una nuova prova.

La registrazione dei dati avviene in modo continuo su ogni test, e per individuare più facilmente la finestra temporale di una data configurazione si è scelto di implementare un sistema denominato TOP. Il suo utilizzo è molto semplice: ogni volta che si cambia il valore di TSR si incrementa un contatore, così da avere un'indicazione facilmente individuabile del passaggio da una condizione all'altra.

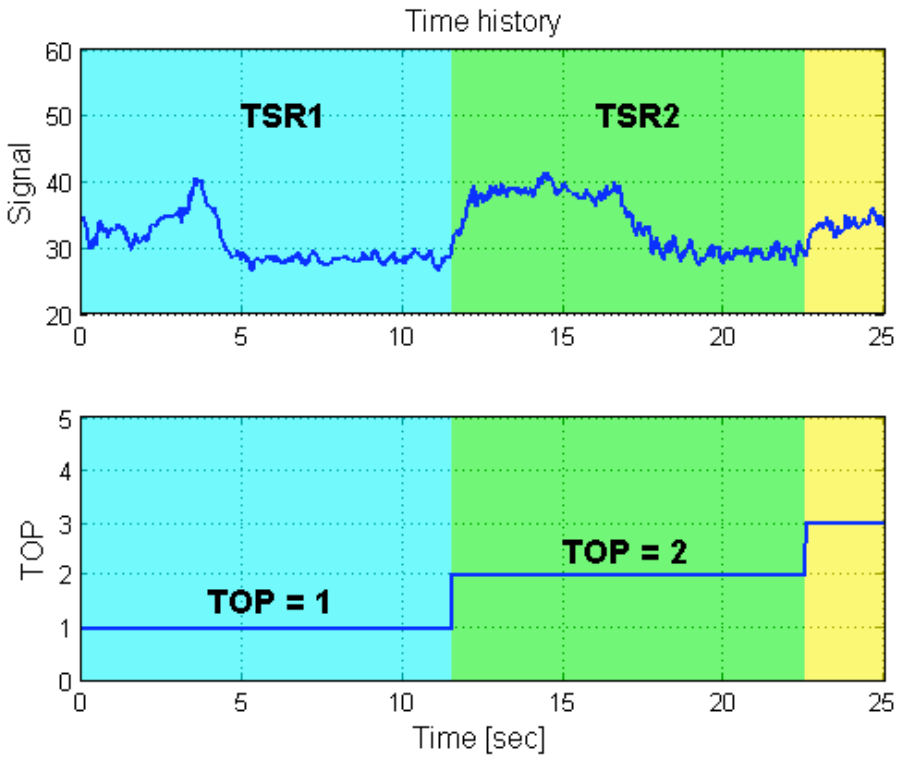

Figura 6.1.: Funzionamento TOP

6.2. Test card

Dalla programmazione delle prove è stata scritta una test card, il cui scopo è regolamentare l'esecuzione delle prove in galleria del vento e renderle il più efficaci e rapide possibili. La sua struttura ricalca quella utilizzata normalmente dagli addetti alla gestione della galleria, in modo tale da permettere una rapida integrazione dei dati da loro misurati con quelli acquisiti attraverso il controllore.

La procedura da utilizzare durante le prove risulta quindi essere:

1. avvio del sistema e verifica dell'operatività
2. avvio della galleria
3. impostazione del sistema al TSR desiderato
4. mantenimento dello stato per circa 10 secondi
5. ripetizione dei punti 3 e 4 fino ad esaurimento dei punti di prova
6. arresto della galleria e del sistema

A partire da questa sequenza si è quindi generata la test card riportata in figura 6.2. Questa comprende una spazzata di 9 TSR di default, ai quali se ne possono aggiungere altri durante l'esecuzione del test; ogni passo di pala β corrisponde ad una diversa prova e quindi ad una diversa scheda.

Data	Attività
13 maggio 2010	Prove statiche aerogeneratore

Identificativo prova	
Allestimento	
Descrizione prova	

Dati e parametri dell'aerogeneratore						
Passo pala		Ora avvio prova		Ora fine prova		

TSR	Vento [m/s]	Rotore [rpm]	TOP	Note
3				
4				
5				
6				
7				
8				
9				
10				

Annotazioni paricolari	

Figura 6.2.: Test card

L'operatore deve compilare ogni scheda riportando:

l'identificativo della prova da concordarsi in modo univoco con la galleria, così da avere delle registrazioni titolate in modo comune facilmente rintracciabili e confrontabili

l'allestimento di prova per poter risalire a quali materiali sono stati utilizzati in caso di analisi di failure

la descrizione della prova contenente gli obiettivi che si intendono raggiungere e le modalità di operazione

Durante lo svolgimento del test saranno poi annotati l'ora di avvio e di arresto, il passo pala e le velocità di vento e rotore ai diversi valori di TSR.

6.3. Monitoraggio con QRtaiLab

Una volta lanciato il codice real-time sul PC controllore si passa a monitorarlo tramite l'interfaccia QRtaiLab, una versione di xrtailab realizzata tramite le librerie grafiche Qt.

Per potersi collegare, i due PC monitor e controllore devono essere sulla stessa rete ethernet, ed avere attivati i moduli *rtai_ netrpc;* questi moduli non garantiscono una comunicazione in tempo reale stretto, avendo la rete ethernet un protocollo di comunicazione di tipo non deterministico, ma consentono la comunicazione tra i due processi e quindi la possibilità di gestione della prova.

Come già accennato nei capitoli precedenti, QRtaiLab non permette lo sviluppo di interfacce grafiche complesse, con bottoni e menu, ma mette a disposizione degli elementi che possono essere adattati alle diverse necessità.

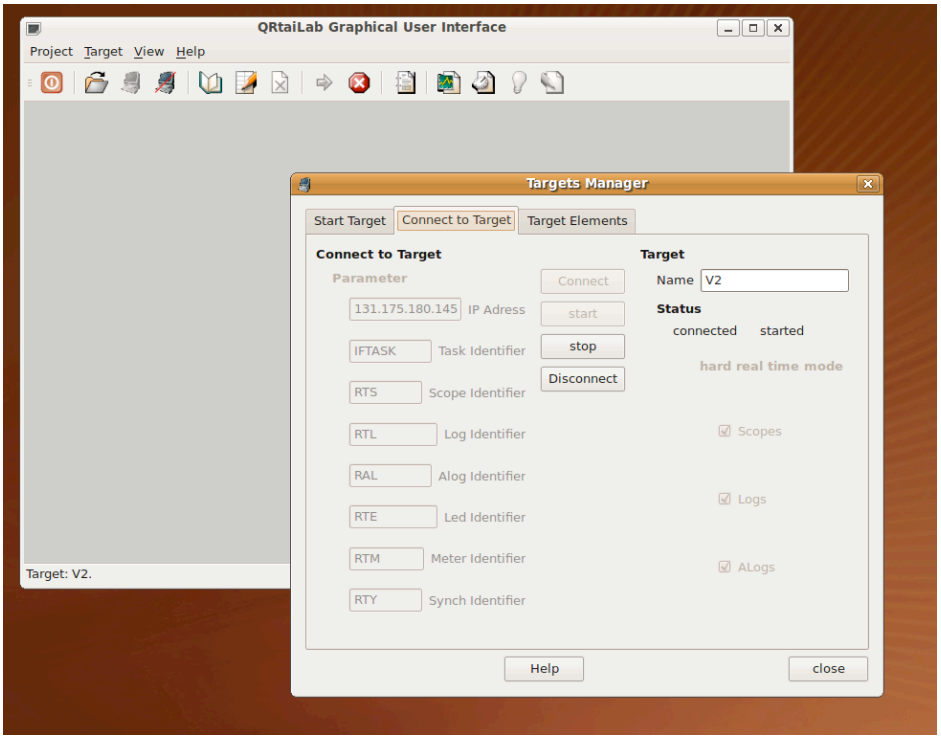

Figura 6.3.: QRtaiLab

In figura si evidenziano, a partire da sinistra, i pulsanti per connettersi all'eseguibile real-time, quelli per salvare e gestire i profili, l'avvio e arresto del programma controllato, l'apertura e gestione di SCOPE, METER e LED, e infine la modifica dei parametri.

L'utilizzo dei profili permette di salvare il layout dell'interfaccia una volta aperti e configurati tutti gli elementi di interesse, come SCOPE e METER, andando così a realizzare una plancia che offre tutte le informazioni importanti direttamente all'avvio.

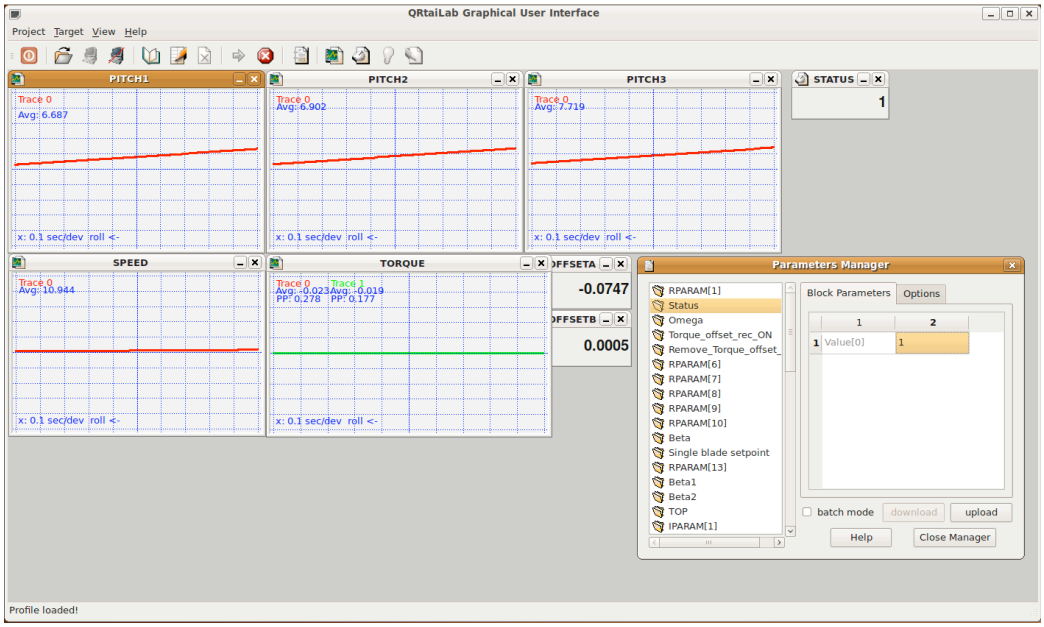

Figura 6.4.: Layout controllo in QRtaiLab

Partendo da in alto a sinistra e spostandosi verso destra si evidenzia la presenza di:

- 3 SCOPE che riportano le informazioni di passo delle 3 pale indipendenti,
- 1 METER che indica lo stato attuale del sistema di controllo
- 1 SCOPE che visualizza la velocità di rotazione del rotore
- 1 SCOPE per la visualizzazione dei due segnali di coppia generati dai ponti estensimetrici
- 2 METER che forniscono il valore di offset dei segnali estensimetrici

Nascosti nel layout di default, ma sempre attivabili, ci sono anche i 3 SCOPE che riportano i valori di passo pala forniti dai 3 potenziometri.

Per ultimo, ma non per importanza, si trova il pannello che permette la modifica dei parametri, tramite il quale si gestisce il comportamento dell'eseguibile real-time.

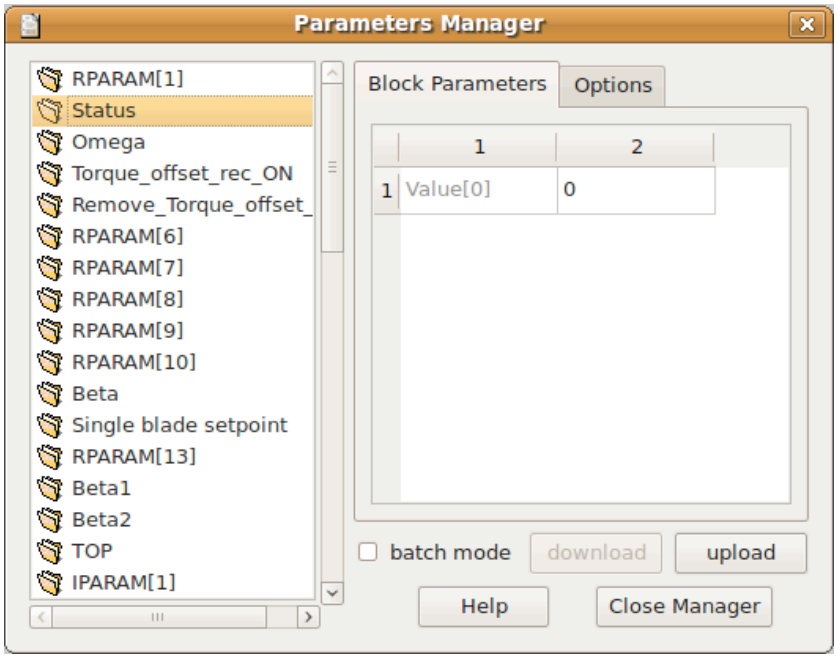

Figura 6.5.: Pannello di controllo in QRtaiLab

I parametri di interesse per quanto riguarda la gestione della prova sono:

Status tramite questo parametro l'utente gestisce le transizioni della macchina agli stati. I valori accettati dal sistema vanno da 0 a 5 e possono essere inseriti solo secondo lo schema logico di funzionamento descritto in 3.2.

Omega una volta raggiunto lo stato 4 questo valore modifica la velocità di rotazione del rotore.

Torque_offset_rec_ON tramite questo parametro è possibile registrare l'offset dei torsiometri. Impostando il valore a 0 si inizia la registrazione del segnale, operazione che dura fino a quando il parametro non è impostato a 1: a questo punto si procede al calcolo del valore medio e alla sua memorizzazione.

Remove_Torque_offset_ON quando questo valore è impostato a 1 il segnale dei torsiometri è depurato del valore medio calcolato precedentemente.

Beta in modalità manuale, a seconda del valore assunto dal parametro *Single blade setpoint*, questo comando permette di cambiare l'angolo di passo delle 3 pale in modo collettivo oppure della sola pala 0.

Single blade setpoint quando questo valore è 0, le pale funzionano in modo collettivo, quando è 1 i valori *Beta*, *Beta1* e *Beta2* controllano le 3 pale in modo indipendente.

Beta1 determina l'angolo di pitch della pala 1 quando il controllo di passo è individuale

Beta2 determina l'angolo di pitch della pala 2 quando il controllo di passo è individuale

TOP questo non è altro che un numero da incrementare ad ogni variazione della configurazione di prova, per avere l'indicazione nella time history del cambio di setup.

6.4. Prove effettuate

Durante il periodo di sviluppo di questo lavoro di tesi si è avuta l'occasione di effettuare un ingresso in galleria del vento, che ha permesso di verificare il funzionamento della macchina in generale e in particolare del sistema di controllo. I problemi sorti durante questa sessione di test sono stati utili per delimitare i limiti operativi del sistema e hanno suggerito le modifiche necessarie dal punto di vista hardware e software.

6.4.1. Condizioni di prova

Si sono eseguite le prove su due giornate, verificando il buon funzionamento dei singoli componenti e poi dell'insieme. Una volta completati questi primi test si è passati alla fase di prova vera e propria, dove la macchina è stata operata in modalità manuale con l'obiettivo di estrarne le curve prestazionali.

Nel dettaglio, le prove eseguite durante questa fase di test sono state:

Tabella 6.1.: Prove eseguite in galleria del vento

Passo pala [deg]	**TSR**	**Velocità del vento [m/s]**	**Velocità del rotore [rpm]**
1	3	10.723	307
1	4	8.013	306
1	5	7.350	351
1	6	6.110	350
1	7	5.240	350
1	8	4.580	350
-3	5	7.330	350
-3	6	6.110	350
-3	7	5.240	350
-3	8	4.581	350
-3	9	4.072	350

Le prove sono state condotte a due soli angoli di pitch a causa della perdita dell'azionamento di una pala. Questo inconveniente tecnico ha diminuito la capacità

operativa del sistema, le cui misure sono risultate di un'accuratezza troppo bassa per poter essere ritenute affidabili ed essere utilizzate per un confronto con il codice simulato.

6.4.2. Criticità del sistema

Le prove effettuate hanno contribuito a determinare le criticità del sistema, sia dal punto di vista hardware che software. Concentrandosi su questo secondo aspetto, il primo problema incontrato è stato la gestione della registrazione dei segnali, che inizialmente venivano memorizzati individualmente attraverso i singoli SCOPE indicanti angolo di pitch, velocità di rotazione e coppia. L'attivazione di questa funzione su ogni elemento richiede un certo lasso di tempo che, seppur minimo, sfasa di qualche secondo la finestra temporale di acquisizione dei singoli segnali. La soluzione adottata è stata quella di realizzare un ulteriore SCOPE che raccogliesse tutti gli input che si intendono memorizzare e attivare solo quest'ultimo elemento per le operazioni di registrazione. Così facendo si ottiene un unico file contenente tutte le informazioni disposte su più colonne e perfettamente sincronizzate tra loro.

File Edit View Search Tools Documents Help

New Open Save Print... Undo Redo Cut Copy Paste Find Replace

Log.dat

	Time [sec]	Speed [rpm]	TorqueA [Nm]	TorqueB [Nm]	Pitch1 [deg]	Pitch2 [deg]	Pitch3 [deg]	POT Pitch1 [deg]	POT Pitch2 [deg]	POT Pitch3 [deg]	TOP
1	18.159744	19.350004	-0.238615	-0.217079	19.350004	19.350004	19.350004	-20.654364	-20.867796	-19.275627	1.000000
2	18.163744	19.393003	-0.381888	-0.226845	19.393003	19.393003	19.393003	-20.506699	-19.889130	-19.665340	1.000000
3	18.167744	19.436003	-0.238615	-0.045724	19.436003	19.436003	19.436003	-21.589569	-21.048077	-19.648020	1.000000
4	18.172743	19.479002	-0.033499	-0.045724	19.479002	19.479002	19.479002	-23.156448	-22.232777	-20.262901	1.000000
5	18.176743	19.522001	-0.114927	-0.130070	19.522001	19.522001	19.522001	-21.532145	-21.649012	-19.786585	1.000000
6	18.180740	19.565001	-0.257169	-0.198434	19.565001	19.565001	19.565001	-20.342628	-20.678930	-19.180363	1.000000
7	18.185740	19.608000	-0.325197	-0.130958	19.608000	19.608000	19.608000	-21.286037	-19.863377	-20.202278	1.000000
8	18.189739	19.650999	-0.053083	0.008435	19.650999	19.650999	19.650999	-22.098188	-21.906553	-20.973045	1.000000
9	18.193739	19.693998	-0.029376	-0.064369	19.693998	19.693998	19.693998	-22.319685	-21.159678	-21.025007	1.000000
10	18.198738	19.736998	-0.316951	-0.098107	19.736998	19.736998	19.736998	-20.547718	-19.700266	-20.488070	1.000000
11	18.202738	19.779997	-0.368488	-0.162032	19.779997	19.779997	19.779997	-20.473885	-19.459890	-19.033138	1.000000
12	18.206738	19.822996	-0.266445	-0.082126	19.822996	19.822996	19.822996	-21.400887	-20.249691	-19.812565	1.000000
13	18.210737	19.865995	-0.163372	0.035958	19.865995	19.865995	19.865995	-22.852917	-21.391468	-20.843142	1.000000
14	18.215736	19.908995	-0.221093	-0.099883	19.908995	19.908995	19.908995	-21.638790	-21.949478	-21.042328	1.000000
15	18.219736	19.951994	-0.371580	-0.201986	19.951994	19.951994	19.951994	-20.949692	-20.301199	-20.410126	1.000000
16	18.223734	19.994993	-0.244800	-0.273014	19.994993	19.994993	19.994993	-20.900471	-19.726019	-20.141657	1.000000
17	18.228733	20.037992	-0.140696	-0.010210	20.037992	20.037992	20.037992	-22.237650	-21.065245	-20.210939	1.000000
18	18.232742	20.080992	-0.003608	-0.090117	20.080992	20.080992	20.080992	-23.090818	-21.108170	-21.380081	1.000000
19	18.236732	20.123991	-0.136573	-0.142500	20.123991	20.123991	20.123991	-21.491125	-21.717690	-19.959791	1.000000
20	18.241732	20.166990	-0.409718	-0.167360	20.166990	20.166990	20.166990	-20.030893	-21.030907	-20.418787	1.000000
21	18.245731	20.209990	-0.317982	-0.122967	20.209990	20.209990	20.209990	-20.162149	-20.739023	-19.526775	1.000000
22	18.249731	20.252989	-0.188109	0.003995	20.252989	20.252989	20.252989	-22.163818	-22.095419	-20.115675	1.000000
23	18.253729	20.295988	-0.039683	-0.070584	20.295988	20.295988	20.295988	-22.024357	-22.198437	-20.496729	1.000000

Figura 6.6.: File dati unico

Un altro aspetto critico riguarda la sincronizzazione dei dati acquisiti attraverso il sistema e quelli di galleria, in particolare i valori di forze e momenti misurati dalla bilancia. La soluzione adottata è stata quella della test card, la cui struttura è progettata in modo tale da avere un file per ogni prova, come avviene durante i test in galleria, e avente lo stesso nome. Questo sistema accelera notevolmente la fase di postprocessing, soprattutto in presenza di un elevato numero di prove caratterizzate da piccole variazioni delle condizioni operative, dove è difficile distinguere le prove in base ai soli valori misurati.

La test card è una soluzione utile fintanto che la macchina è utilizzata in modalità manuale, ma per le applicazioni future di sperimentazione di leggi di controllo l'acquisizione dei dati di bilancia da parte del controllore diventa un requisito fondamentale. I dati riguardanti forze e momenti devono infatti essere disponibili al

sistema per poter elaborare l'azione di controllo, e l'acquisizione diretta è l'unica strada praticabile per ottenere queste informazioni in tempo reale.

7. Conclusioni e sviluppi futuri

Partendo dall'analisi del modello scalato e dei suoi sensori e attuatori si è impostato il sistema in modo tale da rispondere alle specifiche statiche e dinamiche di progetto. Verificato il buon funzionamento dell'insieme si è configurato il bus di comunicazione con gli attuatori e si è appurato come tale scelta fosse compatibile con l'utilizzo in real-time.

Lo studio dei metodi di funzionamento di un vero aerogeneratore ha portato a un confronto con le necessità operative del modello in galleria, evidenziando l'esigenza di dover prevedere nuove funzionalità rispetto a quanto avviene sulla macchina reale; in particolare si è aggiunto l'azionamento manuale usato nel tracciamento delle curve prestazionali, utili a stabilire la corrispondenza del modello con le specifiche di progetto.

Una volta stabilità la logica di controllo si è passati all'implementazione in un sistema real-time che fosse disponibile e affidabile e si è scelto di utilizzare RTAI, essendo quest'ultimo un progetto opensource ad uso gratuito, nato all'interno del Dipartimento di Ingegneria Aerospaziale, caratteristica che ha garantito un'assistenza "diretta" durante lo sviluppo del codice. L'apparato, inteso come somma di un normale computer e di adeguate periferiche di comunicazione, è stato quindi configurato e tarato per comunicare con il modello.

Dovendo tener conto di specifiche deadline nello sviluppo del software, dopo aver analizzato le possibilità di sviluppo del codice in RTAI si è scelto di utilizzare la suite RTAI-Lab, al cui interno si trovano Scilab, un ambiente di simulazione numerica che permette di generare eseguibili a partire da schemi funzionali a blocchi, e xrtailab, un'applicazione grafica che si interfaccia automaticamente con i programmi generati con Scilab e permette di monitorarne le variabili e modificarne i parametri. Questa scelta ha permesso di realizzare piuttosto rapidamente un sistema di controllo completo, imponendo però delle limitazione sulla massima complessità del codice e un'interfaccia grafica relativamente rigida e poco adattabile alle richieste di progetto.

Si è quindi presa in considerazione un'altra soluzione: il sistema Bachmann, un prodotto hardware e software completo utilizzato come controllore nei veri aerogeneratori, fornito di applicazioni avanzate per la gestione dell'impianto, la creazione e il debug di eseguibili e il disegno di interfacce grafiche complesse. Si è eseguito uno studio preliminare sull'implementazione della logica di controllo utilizzata e un confronto con RTAI, dove sono state analizzate le differenze riguardanti le modalità e i tempi di sviluppo del codice, l'espandibilità e le possibilità di aggiornamento e modifica a lungo termine. Da questo confronto è nata la scelta di acquisire anche il prodotto Bachmann e prevedere uno sviluppo in parallelo su questa piat-

taforma, così da avere un apparato di elevata qualità industriale su cui proseguire l'evoluzione della prima realizzazione prototipale.

Le prove in galleria del vento hanno infine validato il controllo dal punto di vista funzionale e operativo, verificando il buon funzionamento delle parti meccaniche e del software, evidenziando i limiti e le criticità da risolvere, come ad esempio l'acquisizione della bilancia da effettuare direttamente attraverso il modello e non tramite galleria.

Allo stato attuale dei lavori il progetto prevede svariati sviluppi futuri e approfondimenti, tra i quali:

- realizzazione del controllo con il sistema Bachmann e verifica delle prestazioni rispetto a RTAI
- progettazione del sistema di acquisizione della bilancia da parte del controllore
- riverifica delle prove in galleria del vento per validare i dati di progetto del modello
- realizzazione di nuove prove in galleria del vento per verificare le diverse leggi di controllo

Bibliografia

[1] Filippo Campagnolo, Preliminary study for the realization of wind tunnel aero-elastic model of a Megawatt-size wind turbine. Tesi laurea magistrale, Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, 2008.

[2] Stefano Calovi, Progetto Preliminare di un Modello Aeroelastico di Aerogeneratore di Taglia Multi-MW per Galleria del Vento. Tesi laurea magistrale, Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, 2009.

[3] Faulhaber, Motion Control Systems Series 3564K024B CC, Motion Controller Series MCBL 3003/06 C, Series MCDC 3003/06 C Instruction Manual. http://www.faulhaber.com/uploadpk/EN_MCxx3003-06C_im_Rev3_DFF.pdf.

[4] http://it.wikipedia.org/wiki/Controllori_PID#Regole_di_Ziegler-Nichols.

[5] MCM EnergyLab, http://www.mcmenergylab.com/MCMEnergyLab/Welcome.html

[6] RTAI - the RealTime Application Interface for Linux from DIAPM, https://www.rtai.org/.

[7] RTAI-Lab, https://www.rtai.org/index.php?module=pagemaster&PAGE_user_op=view_page&PAGE_id=8&MMN_position=23:23.

[8] Scilab, http://www.scilab.org/.

[9] Scicos, http://www-rocq.inria.fr/scicos/.

[10] Comedi, http://www.comedi.org/.

[11] Ubuntu, http://www.ubuntu.com/.

[12] https://www.rtai.org/RTAILAB/RTAI-KubuntuJaunty-ScicosLab-Qrtailab.txt.

[13] Stephen L. Campbell, Jean-Philippe Chancelier and Ramine Nikoukhah, *Modeling and Simulation in Scilab/Scicos*, Springer Science+Business Media, Inc., New York, 2006.

[14] Controller Area Network (CAN), http://www.can-cia.org/index.php?id=46.

[15] CANopen, http://www.can-cia.org/index.php?id=47

[16] http://linux3.dti.supsi.ch/~bucher/pdf/canOpen.pdf

A. Configurazione sistema RTAI

RTAI & ScicosLab & xrtailab & QRtaiLab on Ubuntu Jaunty - 2010-02-09 - V1

By Luca Maffenini (luca dot maffenini at mail dot polimi dot it). Modified version of Guillaume Millet's guide (guimillet at users dot sourceforge dot net)

Introduction

This guide will compile and install all the packages needed to use RTAI with Comedi, ScicosLab and QRtaiLab on an Kubuntu Jaunty machine. Always check if newer versions can be found on their websites, and adjust commands as required.

Listings, downloadings, extractions, renamings and linkings are given as command lines but all these actions can be done graphically.

The guide uses the checkinstall program to create installable packages. Each time, customize your package to your liking, but as a general rule, you should use these values:

0. Maintainer Your Name (youremail@address.com)

1. Summary Use a short description as found in the source, on the website, or how you best remember it.

2. Name In this guide, resp. comedilib, comedi-calibrate, comedi, rtai.

3. Version Either the version number ('3.7.1') or the cvs download date ('cvs20090501').

Preparations

Add your account in the src group (replace [yourusername] with your own user name

```
user@rtai:~$ sudo usermod -a -G src [yourusername]
```

Change the group of /usr/local/src with src and allow writing for this group:

```
user@rtai:~$ sudo chown root:src /usr/local/src
user@rtai:~$ sudo chmod g+w /usr/local/src
```

Install the required packages:

```
user@rtai:~$ sudo aptitude install build-essential automake
checkinstall fakeroot kernel-package libtool flex bison
libboost-program-options-dev libgsl0-dev libgtk2.0-dev
libgail18 gfortran libqwt5-qt4-dev libqt3-mt-dev tcl8.5-dev
tk8.5-dev freeglut3 freeglut3-dev libglut3-dev subversion cvs
```

Download RTAI

Download, extract and link RTAI in /usr/local/src from either:

- the stable release (check for the latest in https://www.rtai.org/RTAI):

```
user@rtai:~$ cd /usr/local/src
user@rtai:~$ wget -no-check-certificate -P /tmp
https://www.rtai.org/RTAI/rtai-3.7.1.tar.bz2
user@rtai:~$ tar xjf /tmp/rtai-3.7.1.tar.bz2
user@rtai:~$ ln -s rtai-3.7.1 rtai
```

- or the development version (requires cvs package):

```
user@rtai:~$ cd /usr/local/src
user@rtai:~$ cvs -d:pserver:anonymous@cvs.gna.org:/cvs/rtai co
magma
user@rtai:~$ mv magma rtai-3.7.1-magma_date # adapt with your
download date
user@rtai:~$ ln -s rtai-3.7.1-magma_date rtai
```

Patch Linux Kernel 2.6

Check the kernel version numbers of available RTAI patches:

```
user@rtai:~$ ls /usr/local/src/rtai/base/arch/x86/patches/
```

and look for the kernel version numbers in hal-linux-[kernel-version].patch. Kernel versions under 2.6.23 are in usr/local/src/rtai/base/arch/i386/patches/.

Download one of these kernels from http://www.kernel.org. You should get a 2.6.x version of the kernel, but a more recent 2.6.x.y version may work too.

```
user@rtai:~$ wget -P /tmp http://www.kernel.org/pub/linux/
kernel/v2.6/linux-2.6.29.4.tar.bz2
```

You can also try to use the Ubuntu kernel source, but "there could be conflicting, additional kernel-patches done by the Ubuntu packagers which are not part of the default kernel source. If it worked (i.e., if the RT patch patched without problems), you're probably fine." [Arno Stienen]. The Ubuntu kernel source metapackage is named linux-source, which installs the latest kernel source with Ubuntu patches (/usr/src/linux-source-2.6.28.tar.bz2).

Extract the source in /usr/src, rename and create a link named "linux" to it:

```
user@rtai:~$ cd /usr/src
user@rtai:~$ tar xjf /tmp/linux-2.6.29.4.tar.bz2
user@rtai:~$ mv linux-2.6.29.4 linux-2.6.29.4-rtai
user@rtai:~$ ln -s linux-2.6.29.4-rtai linux
```

Patch the kernel with a patch with the same 2.6.x number:

```
user@rtai:~$ cd /usr/src/linux
user@rtai:~$ patch -p1 < /usr/local/src/rtai/base/arch/
x86/patches/hal-linux-2.6.29.4-x86-2.4-01.patch
```

If you use the 2.6.x kernel, the patch should work perfectly. For a 2.6.x.y kernel, you may get some Hunk-succeeded messages (which are okay), but if you get real errors, redo the process with a 2.6.x (no .y) kernel.

Copy the original Ubuntu kernel configurations:

```
user@rtai:~$ cp /boot/config-‘uname -r‘ .config
```

Change some configurations; Three configuration interfaces are available, choose one of them: menuconfig (ncurses gui, requires libncurses5-dev) or xconfig (Qt3 gui, requires libqt3-mt-dev) or gconfig (GTK gui, requires libgtk2.0-dev libglib2.0-dev libglade2-dev):

```
user@rtai:~$ make xconfig (or gconfig or menuconfig)
```

Load the configuration file that we saved as .config. Then configure the kernel with these settings:

- Enable loadable module support –> Module versioning support –> disabled
- Processor type and features
  –> Preemption Model –> Preemptible Kernel (Low-Latency Desktop)
  –> Interrupt pipeline –> enabled
  –> Timer frequency –> 1000 Hz
- Power management and ACPI options
  –> CPU Frequency scaling –> CPU Frequency scaling –> disabled
  –> APM (Advanced Power Management) BIOS support –> disabled

For non-SMP systems (for instance, single processor P3 or P4 machines (UPC - UniProcessor)), disable symmetric multi-processing support:

- Processor type and features –> Symmetric multi-processing support –> disabled

Choose the exact processor flavour for your machine. For example, for a Core2Duo processor:

- Processor type and features –> Processor family –> Core 2/Newer Xeon

Disable USB support (not necessary but suggested if you want to use analogic I/O cards)

- Device Drivers –> USB support –> disabled

Create and install Ubuntu kernel package (kernel_source can also be added to the same make-kpkg line, but are not needed), replace -rtai37-core2 by adequate information:

```
user@rtai:~$ sudo make-kpkg clean
user@rtai:~$ make-kpkg -rootcmd fakeroot -append-to-version
-rtai37-core2 -revision r1 -initrd kernel_image kernel_headers
user@rtai:~$ sudo dpkg -i
../linux-headers-2.6.29.4-rtai37-core2_r1_i386.deb
../linux-image-2.6.29.4-rtai37-core2_r1_i386.deb
```

Reboot on your new RT kernel.

RTAI

Compile and install RTAI (for now without Comedi support):

```
user@rtai:~$ cd /usr/local/src/rtai
user@rtai:~$ make menuconfig (or xconfig or gconfig)
```

Configure, optionally adjusting:

- Menu Machine (x86): adjust Number of CPUs (default = 2)

```
user@rtai:~$ make
user@rtai:~$ sudo make install
```

Add ':/usr/realtime/bin' to the path variable in '/etc/environment':

```
user@rtai:~$ sudo sed -i
's/\(PATH=\"\)/\1\/usr\/realtime\/bin:/' /etc/environment
```

Comedilib

```
user@rtai:~$ cd ../comedi
user@rtai:~$ sh autogen.sh
user@rtai:~$ ./configure -disable-pcmcia # you can remove
-disable-pcmcia if you need pcmcia
user@rtai:~$ make
```

Create install script:

```
user@rtai:~$ nano install-rtai
```

and copy the following text:

```
==== START COPY FROM LINE BELOW ==== (Do not include this line!)
#!/bin/bash
make install
mkdir -p /usr/local/include/linux
```

```
cp include/linux/comedi.h include/linux/comedilib.h /usr/local/include/linux
==== STOP COPY AT LINE ABOVE ==== (Do not include this line!)
```

```
user@rtai:~$ sudo checkinstall -fstrans=no bash install-rtai
# Use 'comedi' as name; for other values, see introduction.
```

RTAI + Comedi

```
user@rtai:~$ cd /usr/local/src/rtai
user@rtai:~$ make xconfig (or menuconfig or gconfig)
```

Configure RTAI with COMEDI:

- Menu Add-ons: Select Real Time COMEDI support in user space

```
user@rtai:~$ make
user@rtai:~$ sudo checkinstall -fstrans=no
# Use 'rtai' as name; for other values, see introduction.
```

EFLTK[1]

Download to folder with current date, compile and install:

```
user@rtai:~$ cd /usr/local/src
user@rtai:~$ svn co
https://ede.svn.sourceforge.net/svnroot/ede/trunk/efltk
user@rtai:~$ cd efltk
user@rtai:~$ autoconf
user@rtai:~$ ./configure -disable-mysql -disable-unixODBC
-enable-xft -prefix=/usr
user@rtai:~$ ./emake
user@rtai:~$ sudo checkinstall bash emake install
# Use 'efltk' as name; for other values, see introduction.
```

[1] only if you want to compile xrtailab, based on http://equinox-project.org/wiki/UbuntuInstallation

PEAK driver

Download and compile the CAN driver:

```
user@rtai:~$ cd /usr/local/src
user@rtai:~$ wget -O /tmp/peak-linux-driver.6.15.tar.gz
http://p103112.typo3server.info/fileadmin/media/linux/
files/peak-linux-driver.6.15.tar.gz?view=tar
user@rtai:~$ tar xzf /tmp/peak-linux-driver.6.15.tar.gz
user@rtai:~$ cd peak-linux-driver-6.15
user@rtai:~$ make RT=RTAI
user@rtai:~$ sudo make install
```

ScicosLab + RTAI-Lab add-ons

Download and install ScicosLab from its website:

```
user@rtai:~$ wget -P /tmp http://cermics.enpc.fr/~jpc/
scilab-gtk-tiddly/files/scicoslab-gtk_4.3-3_i386.jaunty.deb
user@rtai:~$ sudo dpkg -i
/tmp/scicoslab-gtk_4.3-3_i386.jaunty.deb
```

Install RTAI-Lab add-ons as root and as user:

```
user@rtai:~$ cd /usr/local/src/rtai/rtai-lab/scilab/
user@rtai:~$ sudo make install # to install rtmain.c in
/usr/realtime/share/rtai/scicos
user@rtai:~$ cd ../scicoslab/macros
user@rtai:~$ sudo su -c "make install"
user@rtai:~$ make user
```

QRtaiLab

Download QRtaiLab source on http://qrtailab.sourceforge.net and extract it in /usr/local/src:

```
user@rtai:~$ cd /usr/local/src
user@rtai:~$ wget -P /tmp http://downloads.sourceforge.net/
qrtailab/QRtaiLab-0.1.7.tar.gz
user@rtai:~$ tar xzf /tmp/QRtaiLab-0.1.7.tar.gz
```

Modify qrtailab.config as follows (to use the Ubuntu version of libqwt5):

```
INCLUDEPATH += . /usr/realtime/include /usr/include/qwt-qt4
LIBS += -lqwt-qt4
```

Make and install:

```
user@rtai:~$ cd qrtailab-0.1.7
user@rtai:~$ qmake-qt4
user@rtai:~$ make
user@rtai:~$ sudo cp qrtailab /usr/local/bin
```

Post Installation

Copy the following lines in a read_IP script that you can use to automatically set the realtime IP address

```
user@rtai:~$ sudo nano /usr/realtime/bin/read_IP
```

```
==== START COPY FROM LINE BELOW ==== (Do not include this line!)
#!/bin/sh
# Shell script scripts to read ip address
# ------------------------------------------------------------
# Copyright (c) 2005 nixCraft project <http://cyberciti.biz/fb/>
# This script is licensed under GNU GPL version 2.0 or above
# ------------------------------------------------------------
# This script is part of nixCraft shell script collection (NSSC)
# Visit http://bash.cyberciti.biz/ for more information.
# ------------------------------------------------------------
# Get OS name
OS=‘uname‘
IO="" # store IP
case $OS in
   Linux) IP=‘ifconfig  | grep ’inet addr:’| grep -v ’127.0.0.1’
```

```
 | cut -d: -f2 | awk '{ print $1}'`;;
   FreeBSD|OpenBSD) IP=`ifconfig  | grep -E 'inet.[0-9]' |
grep -v '127.0.0.1' | awk '{ print $2}'` ;;
   SunOS) IP=`ifconfig -a | grep inet | grep -v '127.0.0.1'
 | awk '{ print $2} '` ;;
   *) IP="Unknown";;
esac
echo "$IP"
==== STOP COPY AT LINE ABOVE ==== (Do not include this line!)
```

Copy the following lines in a rlinsmod script that you can use to quickly load RTAI-Lab and Comedi modules in kernel and configures the drivers.

```
user@rtai:~$ sudo nano /usr/realtime/bin/rlinsmod
```

```
==== START COPY FROM LINE BELOW ==== (Do not include this line!)
# RLINSMOD - RTAI-Lab Insmod.
# Inserts RTAI-Lab modules in kernel and configures the drivers.
MOD_DIR=/usr/realtime/modules
IP=`/usr/realtime/bin/read_IP`
sync
insmod ${MOD_DIR}/rtai_hal.ko
insmod ${MOD_DIR}/rtai_lxrt.ko
insmod ${MOD_DIR}/rtai_fifos.ko
insmod ${MOD_DIR}/rtai_sem.ko
insmod ${MOD_DIR}/rtai_mbx.ko
insmod ${MOD_DIR}/rtai_msg.ko
insmod ${MOD_DIR}/rtai_netrpc.ko  ThisNode="$IP"
insmod ${MOD_DIR}/rtai_rtdm.ko
modprobe pcan       # only if you want to use Peak CAN boards
modprobe comedi     # only if you want to use comedi
modprobe kcomedilib
modprobe comedi_fc
insmod ${MOD_DIR}/rtai_comedi.ko
modprobe ni_pcimio  # comedi driver for analog I/O board
comedi_calibrate -f /dev/comedi0
echo "Realtime IP address: $IP"
==== STOP COPY AT LINE ABOVE ==== (Do not include this line!)
```

Copy the following lines in a rlrmmod script that you can use to quickly remove RTAI-Lab and Comedi modules from kernel.

```
user@rtai:~$ sudo nano /usr/realtime/bin/rlrmmod
```

```
==== START COPY FROM LINE BELOW ==== (Do not include this line!)
# RLRMMOD - RTAI-Lab Rmmod.
# Removes RTAI-Lab modules from kernel.
comedi_config -r /dev/comedi0
modprobe -r ni_pcimio         # comedi driver for analog I/O board
rmmod rtai_comedi             # only if you want to use comedi
modprobe -r kcomedilib        # only if you want to use comedi
modprobe -r pcan
# only if you want to use Peak CAN boards
rmmod rtai_rtdm
rmmod rtai_netrpc
rmmod rtai_msg
rmmod rtai_mbx
rmmod rtai_sem
rmmod rtai_fifos
rmmod rtai_lxrt
rmmod rtai_hal
==== STOP COPY AT LINE ABOVE ==== (Do not include this line!)
```

Allow the scripts to be executed:

```
user@rtai:~$ sudo chmod +x /usr/realtime/bin/read_IP
user@rtai:~$ sudo chmod +x /usr/realtime/bin/rlrmmod
user@rtai:~$ sudo chmod +x /usr/realtime/bin/rlinsmod
```

If you want to start rlinsmod after every boot (check first that rlinsmod does not freeze your computer...), add the line '/usr/realtime/bin/rlinsmod' (without the quotes) to your rc.local file. Place this line above 'exit 0'.

```
user@rtai:~$ sudo nano /etc/rc.local
```

Done

Read the RTAI-Lab tutorial for more information. In particular, you can find a description of the RTAI-Lib palette, a guide to make a real-time sinewave, and a few examples showing how to use FIFOS and semaphores and how to convert a continuous-time plant in a discrete-time model. The tutorial is in the RTAI-Lab repository, https://www.rtai.org/RTAILAB.