Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Sistema didattico per laboratorio di controllo

Studente/i

Giannini Mattia

Relatore

Bucher Roberto

Correlatore

Vaghi Alessandro

Committente

-

Corso di laurea

Ingegneria elettronica

Modulo

AA2019/20: Progetto di diploma

Anno

2019/2020

Data

29 agosto 2020

STUDENTSUPSI

Indice

1 Abstract **1**

2 Progetto assegnato e specifiche di progetto **3**

2.1 Scheda descrittiva 4

2.1.1 Persone coinvolte 4

2.1.2 Dati generali 4

2.1.3 Compiti 4

2.1.4 Obiettivi 5

2.1.5 Tecnologie 5

2.2 Gantt 7

2.3 Specifiche del progetto 8

2.4 Valutazione SMART delle specifiche del progetto 9

3 Introduzione **11**

3.1 Stato dell'arte 11

3.2 Model-Base Design 11

4 Requisiti hardware e software **13**

4.1 Software 13

4.2 Hardware 14

4.2.1 Scelta hardware 15

5 Test scheda di controllo **17**

5.1 Test alimentazione principale 17

5.2 Test del processo Heater 18

5.3 Test del processo Inverter 20

5.4 Test del processo Booster 23

5.5 Test processo Driver 24

5.6 Osservazioni 26

STUDENTSUPSI

6 Modellizzazione **29**
6.1 Modelli Heater . . . 29
6.1.1 Modello Heater SISO . . . 29
6.1.2 Modello Heater MIMO . . . 30
6.2 Modello Inverter . . . 30

7 Identificazione **33**
7.1 Heater . . . 33
7.1.1 Identificazione heater SISO . . . 34
7.1.1.1 Elaborazione dati heater SISO . . . 34

8 Sviluppo software **37**
8.1 Obbiettivo . . . 37
8.1.1 Visione del funzionamento finale . . . 37
8.2 Introduzione alle S-Function . . . 39
8.3 Funzionamento del toolbox STM32MAT/TARGET . . . 39
8.3.1 Generazione file.ioc . . . 40
8.3.2 Test del toolbox . . . 40
8.3.3 Analisi SPI . . . 41
8.3.3.1 Problemi generazione codice . . . 42
8.3.3.2 Analisi codice generato . . . 42
8.3.3.3 Osservazioni . . . 43
8.3.4 Generazione di codice per makefile . . . 43
8.3.5 Osservazioni toolbox STM32MAT/TARGET . . . 46

9 PysimCoder **47**
9.1 Demo con PysimCoder . . . 48

10 Migliorie **51**
10.1 Modifiche dell'hardware . . . 51
10.1.1 Resistenze R2 e R4 . . . 51
10.1.2 Sostituzione di componenti . . . 52
10.1.3 Possibili aggiunte . . . 52
10.2 Osservazioni . . . 53

11 Costi e spese **55**

12 Conclusioni **57**
12.1 Sitografia . . . 59
12.1.1 Heater . . . 59
12.1.2 STM32MAT/TARGET . . . 59

STUDENTSUPSI

12.2 Bibliografia 59

13 Allegati **61**

A Programma di test **63**
A.1 Descrizione funzionamento 63
A.2 Simulazione 64
A.2.1 Documentazione 64

B Manuale d'uso STM32MAT/TARGET **67**
B.1 Introduzione 67
B.2 Programmi 67
B.2.1 STM32-MAT/TARGET 67
B.2.2 STM32CubeMX 68
B.2.3 IAR Embedded Workbench for Arm 68
B.2.4 MATLAB/Simulink 68
B.3 Configurazione MATLAB 68
B.3.1 Configurazione STM32-MAT/TARGET 68
B.3.2 Configurazione MATLAB Add-Ons 70
B.3.3 Verifica configurazione 72
B.4 Generazione file .ioc 72
B.4.1 Creazione progetto 72
B.4.2 Abilitazione delle periferiche 73
B.4.3 Parametri project manager 74
B.5 Generazione di codice tramite Simulink 75
B.5.1 Configurazione MCU 75
B.5.2 Sviluppo codice esempio tramite Simulink 78
B.5.3 Debuggin codice 83

C Scheda Heater **87**
C.1 Introduzione 87
C.2 Obbiettivi 87
C.3 Modellizzazione 87
C.4 Controllori SISO 88
C.4.1 Controllore di stato 89
C.4.2 Controllore PID 89

D Soluzioni Heater **91**
D.1 Modellizzazione MIMO 91
D.1.1 Osservazioni 94
D.2 Modellizzazione SISO 95

STUDENTSUPSI

D.3 Controllore di stato SISO 95
D.3.1 Risultati 96
D.4 Controllore PID SISO 97
D.4.1 Risultati 98
D.5 Osservazioni 99

E Scheda Inverter 101
E.1 Introduzione 101
E.2 Obiettivi 101
E.3 Modelizzazione 101
E.4 Controllori 102
E.4.1 Controllore di stato 102
E.4.2 Controllore polinomiale 103
E.5 Data 103

F Soluzioni Inverter 105
F.1 Modellizzazione Inverter 105
F.2 Controllore di stato 106
F.2.1 Risultati 107
F.3 Controllore polinomiale 108
F.3.1 Risultati 109
F.4 Osservazioni 110
F.5 Dati 110

G Nomenclature 111
G.1 Nomenclature Heater 111
G.2 Nomenclature Inverter 112

STUDENTSUPSI

Elenco delle figure

4.1 Scheda controllo 14
4.2 Evaluation board STM32F746ZG 15
4.3 Evaluation board STM32F746ZG dati tecnici 16

5.1 Misure effettuate sull'alimentazione 18
5.2 Misure effettuate sul processo Heater 19
5.3 Flowchart test Heater 20
5.4 Misure effettuate sul processo Inverter 21
5.5 Flowchart test Inverter 22
5.6 Misure effettuate sul processo Booster 23
5.7 Flowchart test Booster 24
5.8 Misure effettuate sul processo Driver 25
5.9 Flowchart test Driver 26

6.1 Circuito di modellizzazione Inverter 31

7.1 Processo di acquisizione dati 34
7.2 Modello identificazione heater SISO 34

8.1 Simulazione Simulink 38
8.2 Modello controllore reale Simulink 38
8.3 Processo generazione codice 38
8.4 Test STM32MAT/TARGET 41
8.5 STM32MAT/TARGET SPI block 42
8.6 Struttura file generati 43
8.7 STM32MAT/TARGET SPI test makefile 44

9.1 Evaluation board STM32H745ZI 47
9.2 Schema PysimCoder controllore di stato 48
9.3 Schema PysimCoder visualizzazione dati 49

10.1 Errore nello schematico TopLevel 54

STUDENTSUPSI

A.1 Risposta al gradino del sistema riscaldante 64
A.2 Collegamenti scheda controllo - evaluation board 64

B.1 MATLAB Set Path 69
B.2 MATLAB Add with Subfolders 69
B.3 MATLAB select Folder 70
B.4 MATLAB STM32 folders 70
B.5 MATLAB Add-Ons 71
B.6 MATLAB Manage Add-Ons 71
B.7 Simulink library 72
B.8 STM32CubeMX start project 72
B.9 STM32CubeMX MCU selection 73
B.10 STM32CubeMX peripherals selection 74
B.11 STM32CubeMX project manager 74
B.12 STM32CubeMX save project 75
B.13 Simulink project setting 75
B.14 Simulink project solver setting 76
B.15 Simulink code generation setting 76
B.16 Simulink code generation browse setting 77
B.17 Simulink code generation STM32 options 77
B.18 Simulink setting save 78
B.19 Project MCU setting 78
B.20 Project MCU setting ioc file seletion 79
B.21 Project GPIO setting 80
B.22 Project block diagram step 1 80
B.23 Project state machine 81
B.24 Project block diagram step 2 81
B.25 Project code generation error 82
B.26 Project simbol wyzard 82
B.27 Project block diagram step 3 83
B.28 Project debuggin step 1 83
B.29 Project debuggin step 2 84
B.30 Project debuggin step 3 84
B.31 Project debuggin step 4 85

C.1 Rappresentazione del processo Heater 88
C.2 Struttura controllore di stato 89
C.3 Struttura controllore PID 89

D.1 Differenziazione $[A_s]$ e $[A]$ 94

STUDENTSUPSI

D.2 Simulazione controllore di stato Heater SISO 96
D.3 Struttura controllore PID 97
D.4 MATLAB autotune PID 98
D.5 Simulazione controllore PID Heater SISO 98

E.1 Circuito di modellizzazione Inverter 102
E.2 Struttura controllore di stato 102
E.3 Struttura controllore polinomiale 103

F.1 Circuito di modellizzazione Inverter 105
F.2 Struttura controllore di stato 106
F.3 Piazzamento poli controllore di stato 107
F.4 Simulazione controllore di stato Inverter 107
F.5 Struttura controllore polinomiale 108
F.6 Struttura piazzamento poli 109
F.7 Simulazione controllore polinomiale Inverter 110

STUDENTSUPSI

STUDENTSUPSI

Elenco delle tabelle

2.1 Persone coinvolte nel progetto 4
2.2 Tabella dati progetto 4
2.3 Valutazione SMART delle specifiche 9

5.1 Rilevamento tensioni di alimentazione 18
5.2 Rilevamento tensioni IC Heater 19
5.3 Rilevamento tensioni IC Inverter 21
5.4 Rilevamento tensioni IC Booster 23
5.5 Rilevamento tensioni IC Driver 25

8.1 Fasi si simulazione S-Function 39

10.1 Valori resistenze processo heater 52

11.1 Costi di progetto 55

B.1 Configurazione GPIO 73

E.1 Parametri del sistema 103

F.1 Parametri del sistema 110

G.1 Parametri del sistema 111
G.2 Parametri del sistema 112

STUDENTSUPSI

STUDENTSUPSI

Capitolo 1

Abstract

La pandemia ha imposto e imporrà probabilmente anche in futuro l'accesso limitato a laboratori ed esperienze comuni, per questo motivo l'obbiettivo di realizzare un sistema didattico portatile è di grande interesse per il prossimo futuro. In questo senso la pandemia ha avuto il merito di accelerare la tendenza a permettere agli studenti di lavorare come, quando e dove vogliono; per fare ciò si necessitano di nuovi sistemi didattici compatti e portatili.

L'applicazione di sistemi di controllo è alla base di qualsiasi sistema meccatronico o elettronico. Esistono un numero notevole di tipi di controllori che possono essere utilizzati per un sistema. Lo sviluppo di un controllore, in molti casi, si basa su un'attenta analisi del processo: attraverso le specifiche richieste e la dinamica fisica del sistema. L'esperienza dello sviluppatore nel campo dei controllori, può ridurre il numero di tentavi da effettuare. Per questo motivo si vuole avere una scheda flessibile dove poter fare esperienze, dalla modellazione, all'identificazione parametrica, fino a controllori complessi come possono essere la famiglia dei controllori ottimi e predittivi.

L'obbiettivo è quello di poter fornire una scheda con un'interfaccia che ne permetta l'utilizzo da vari sistemi embedded (per esempio: Arduino, Briki, Raspberry PI, STM32, sistemi coustom, ecc. e fornire agli studenti un sistema didattico dove poter fare esperienze in modo autonomo, e dare la possibilità di sperimentare nuove tecniche, in modo semplice, veloce e a basso costo.

In questo progetto le difficoltà riscontrate con i vari tool di sviluppo (MATLAB, Simulink, ST tool e i vari IDE) non hanno permesso di sviluppare il sistema desiderato; il toolbox STM32MAT/TARGET di ST presenta innumerevoli problemi, dovuti sopratutto alla scelta di mantenere un sistema chiuso. In corso d'opera si è cercato di risolvere i problemi riscontrati concentrandosi nel far funzionare un unico processo (Heater).

STUDENTSUPSI

Nelle ultime settimane del progetto con l'aiuto del relatore si è voluto evidenziare il vantaggio di costruire un sistema personalizzato open source a discapito di un sistema chiuso.

Il relatore ha sviluppato il proprio sistema open source, basato su python per un'altra evaluation board la STM32H745ZIT6, il sistema presenta un ampio spazio di manovrabilità, che ha permesso di ottenere una demo.

STUDENTSUPSI

Capitolo 2

Progetto assegnato e specifiche di progetto

Il progetto viene sviluppato per far fronte alle problematiche riscontrate e "messe in evidenza" dal virus Covid-19. L'idea è quella di sviluppare un sistema didattico compatto che permetta agli studenti che seguono corsi nell'ambito di controllo, di poter mettere in pratica le nozioni acquisite durante i corsi.

Il sistema fornisce 4 tipi diversi di processi su una scheda realizzata in SUPSI, i processi che mette a disposizione sono i seguenti:

- Un sistema MIMO con due elementi riscaldanti e sensori di temperatura
- Un ponte H per il controllo di motori
- Un booster per il controllo di una tensione
- Un sistema RLC

Si vuole disporre di alcune esperienze guidate, e con le relative soluzioni, da distribuire agli studenti per verificare la buona riuscita del proprio lavoro.

STUDENTSUPSI

2.1 Scheda descrittiva

2.1.1 Persone coinvolte

Proponente	Bucher Roberto
Relatore	Bucher Roberto
Correlatore	Vaghi Alessandro
Studente	Giannini Mattia

Tabella 2.1: Persone coinvolte nel progetto

2.1.2 Dati generali

Codice	C10148
Anno Accademico	2019/2020
Semestre	Semestre estivo
Corso di laurea	Ingegneria elettronica
Opzione	/
Tipologia di progetto	diploma
Stato	In corso
Confidenziale	NO
Pubblicabile	SI

Tabella 2.2: Tabella dati progetto

2.1.3 Compiti

Partendo da quanto già esiste (blocchi in pysimCoder, toolbox particolari di MATLAB, toolbox specifici sviluppati da altre università), costruire le interfacce verso la scheda di controllo per poi accedere a tutti i segnali da leggere e scrivere per i 4 processi.
Il progetto prevede diverse fasi:

- Interfacciare la scheda con uno dei seguenti processori Raspberry, Briki, STM32H7, STM32F7
- Analizzare i processi sulla scheda e modellarli matematicamente
- Verificare sperimentalmente i modelli trovati
- Costruire blocchi MATLAB/Simulink o per l'ambiente pysimCoder per interfacciarsi con la scheda dei processi, tenendo conto di quanto già è stato fatto (per esempio MATLAB ha un toolbox per Raspberry PI)

STUDENTSUPSI

- Per ognuno dei processi sulla scheda definire diversi tipi di controllori, dal PID al controllo di stato fino ev. al controllo MCP.
- Documentare ogni esperienza preparando:
  - Una scheda con l'esercizio di laboratorio da fare
  - Una scheda con la soluzione
- Documentare il progetto (Documento, Presentazione, Poster)

2.1.4 Obiettivi

- Testare il funzionamento della scheda con i processi da controllare
- Definire i blocchi di interfaccia
- Analizzare il SW già presente, sia in pysimCoder, sia in MATLAB/Simulink
- Costruire eventuali nuovi blocchi in collaborazione con il relatore
- Modellare i 4 processi
- Preparare diverse esperienze, secondo discussione con il relatore, per ognuno dei 4 processi
- Preparare la documentazione delle varie esperienze per gli studenti, descrizione, soluzione.

2.1.5 Tecnologie

- MATLAB/Simulink
- Python
- Sistemi embedded
- Programmazione in Python
- Programmazione in C
- Analisi dei segnali
- Controllo di sistemi
- Metodologie di prototipaggio rapido di controllori

STUDENTSUPSI

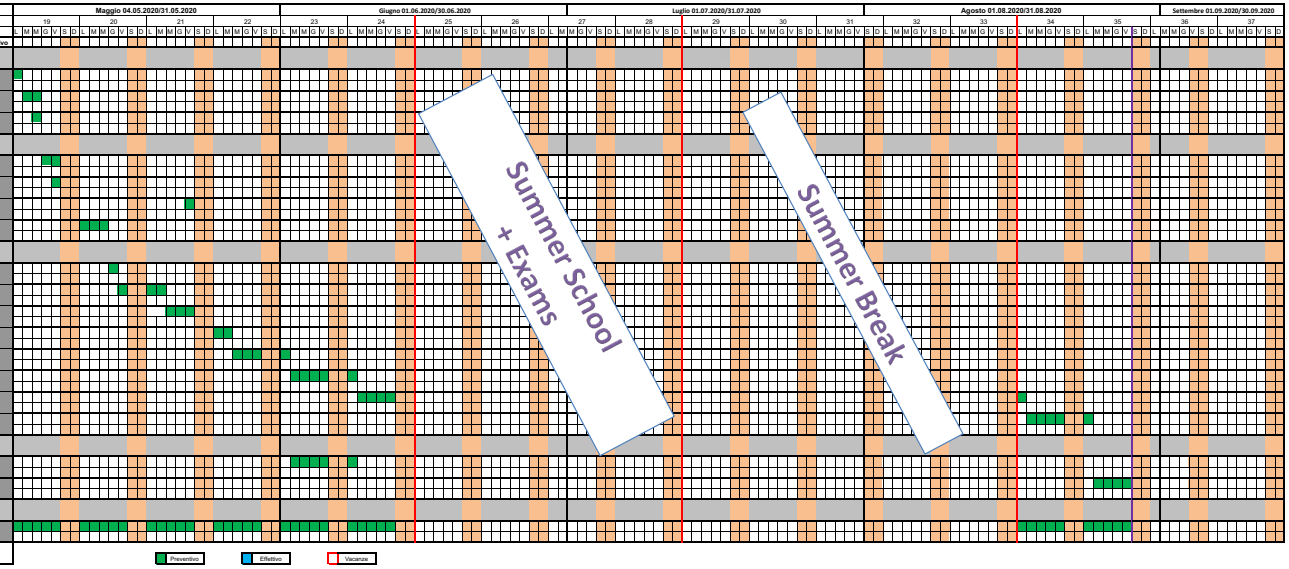
2.2 Gantt

Progetto: Sistema didattico per laboratorio di controllo

SUPSI-DTI - Ingegneria Elettronica

Inizio Progetto:	04.05.20	Studenti:	Giannini Mattia
Termine Progetto:	31.08.20	Relatore:	Prof. Roberto Bucher
		Committente:	SUPSI

Pianificazione progetto di semestre			
	Attività	Preventivo	Consuntivo
1	Consegna del lavoro		
1.1	Introduzione / Consegna	1	
1.2	Pianificazione del lavoro	2	
1.3	Specifica dei requisiti	1	
2	Ricerca		
2.1	Analisi stato dell'arte	2	
2.2	Valutazione componenti necessari (uC)	1	
2.3	Studio della scheda di controllo	1	
2.4	Studio crosscompilazione con simulink per piattaforma ST	3	
3	Sviluppo		
3.1	Testing scheda di controllo	1	
3.2	Interfacciare microcontrollore con simulink	3	
3.3	Acquisire i dati con microcontrollore dalla scehda di controllo	3	
3.4	Sviluppo struttura dell'esperienze	2	
3.5	Modellizzazione dei processi (Heater, Ponte H, Booster, RLC)	4	
3.6	Sviluppo controllori (PI, PID, Osservatore, Osservatore ridotto, ecc.)	5	
3.7	Sviluppo soluzioni delle esperienze (su carta)	5	
3.8	Sviluppo soluzioni delle esperienze (su simulink)	5	
4	Verifiche e ottimizzazione		
4.1	Simulazioni in matlab (parallelo al 3.6)	5	
4.1	Correzione e ottimizzazione	4	
5	Documentazione e diario di lavoro		
5.1			
	Totale giorni:	48	0



STUDENTSUPSI

2.3 Specifiche del progetto

Le seguenti specifiche sono state concordate con il relatore del progetto, il quale le ha approvate con riserva di modifica per eventuali migliorie a discrezione del tempo rimanente.

Elemento/Caratteristica	**Specifica**
Ambiente di sviluppo	*- ST IDE* *- Simulink* *- MATLAB* *- PysimCoder* *- Python*
Sviluppo di controllori	*- PID* *- Controllo di stato (osservatore, ottimo, ...)* *- ev. MPC*
Sviluppo esperienze (tramite scheda)	*- Heater* *- RLC* *- Booster* *- Ponte H*
Sviluppo soluzioni esperienze	*- Heater* *- RLC* *- Booster* *- Ponte H*
Budget	*- 300 CHF*
Documentazione	*- Rapporto* *- Presentazione* *- Poster*
Termine	*- 28.08.2020*
Osservazione	Procedere processo per processo

STUDENTSUPSI

2.4 Valutazione SMART delle specifiche del progetto

Con la valutazione SMART si verifica se le specifiche concordate con il relatore sono sufficientemente dettagliate per l'esecuzione del progetto.

Specifica	Specifico	Misurabile	Attuabile	Realistico	Time-related
Ambiente di sviluppo	Yes	Yes	Yes	Yes	28.08.2020
Sviluppo di controllori	Yes	Yes	Yes	Yes	28.08.2020
Sviluppo esperienze	Yes	Yes	Yes	Yes	28.08.2020
Sviluppo soluzioni esperienze	Yes	Yes	Yes	Yes	28.08.2020
Budget	Yes	Yes	Yes	Yes	
Documentazione	Yes	Yes	Yes	Yes	28.08.2020
Termine massimo	Yes	Yes	Yes	Yes	28.08.2020

Tabella 2.3: Valutazione SMART delle specifiche

Dalla Tabella 2.3 risulta infatti che tutte le specifiche sono SMART.

STUDENTSUPSI

STUDENTSUPSI

Capitolo 3

Introduzione

I sistemi di controllo sono fondamentali al giorno d'oggi, permettono di aumentare di controllare in maniera precisa un qualsiasi processo, ad esempio: un riscaldamento, un inverter, un motore, un booster, ... È dunque fondamentale prendere confidenza con le varie tipologie di controllori esistenti come: controllori di stato, controllori PID, controllori polinomiali, controllori vettoriali e/o predittivi. Quale modo migliore di un sistema di laboratorio compatto e versatile che può fornire allo studente un modo di ampliare le sue conoscenze sui controllori. Con questo progetto lo scopo è di munire lo studente di un sistema portatile dove poter fare pratica.

3.1 Stato dell'arte

Vi sono molteplici laboratori di controllori portatili in commercio, la stragrande maggioranza è basata su un microcontrollore Arduino, il quale presenta un massimo di due processi per scheda. Questi laboratori solitamente sono da interfacciare con un generatore di codice (per esempio: Simulink), il quale ha a disposizione molteplici librerie per la generazione di codice.
Nella generazione automatica di codice alcune case costruttrici stanno mettendo a disposizione alcuni tool aggiuntivi per Simulink, i quali permetto di generare il codice da Simulink e scaricarlo sul microcontrollore attraverso un IDE.
STmicroelttronics è una di quelle aziende che ha messo a disposizione un tool aggiuntivo per Simulink, in grado di generare codice per alcune evaluation board della famiglia STM32.

3.2 Model-Base Design

Model-Based Design è una metodologia per progettare software per sistemi embedded. Come dice il nome, il processo di sviluppo di un sistema di controllo è focalizzato sul modello, sfruttando la modellazione grafica e le capacità di generazione del codice. Mentre la

STUDENTSUPSI

modellazione grafica riduce la complessità della rappresentazione matematica del sistema, la generazione del codice evita il processo di trascrizione nel linguaggio di programmazione che potrebbe comportare errori difficili da rilevare. In questo senso, il problema di progettazione è semplificato dalla creazione del modello grafico e dalla sua convalidazione con i test e le simulazioni corrette, mentre il resto del processo è praticamente automatizzato.

La scrittura, la verifica e i test del codice C richiedono molto tempo, specialmente quando il sistema di destinazione diventa più complesso. Questi passaggi possono essere saltati con la generazione del codice, ottenendo un importante risparmio di tempo. Il rilevamento degli errori e i dettagli matematici sono più facili da gestire dal punto di vista di alto livello della modellazione grafica.

STUDENTSUPSI

Capitolo 4

Requisiti hardware e software

Questo capitolo ha come obbiettivo di chiarire quale software e hardware sono necessari per realizzare il progetto presentato. Il progetto è basato su STM32 evaluation kit, ovvero una evaluation board STM32 nucleo e di una scheda sviluppata in SUPSI con 4 diversi processi.

4.1 Software

- **MATLAB/Simulink** di MathWorks (versione R2019a o successive):Ambiente per modellazione grafica, simulazione e analisi di sistemi dinamici, basati su interconnessione di blocchi di alto livello che rappresentano funzioni logiche e matematiche.
  Alcuni componenti aggiuntivi sono necessari per questo lavoro:
  - Stateflow: fornisce un ambiente per progettare una macchina a stati in Simulink.
  - Simscape: include sistemi elettrici pre-modellati, come sorgenti di tensione e motori elettrici.
  - Embedded Coder: strumento per tradurre una rappresentazione di sistema da a Simulink modello in codice C.
- **STM32CubeMX** di STMicroelectronics (versione 5.6.1 o successive): Strumento grafico di configurazione del software che consente la generazione del codice di inizializzazione C per MCU STM32 mediante procedure guidate grafiche.
- **STM32-MAT/TARGET** di STMicroelectronics (versione 5.6.0): È un'estensione MATLAB che collega un modello Simulink a un file STM32CubeMX al fine di abilitare i blocchi periferici del MCU STM32. Il suo scopo è collegare graficamente un sistema modellato alla MCU per generare un codice totalmente integrato utilizzando Embedded Coder.

STUDENTSUPSI

- **KEIL uVision IDE** di KEIL (versione 5 o successive): Ambiente di sviluppo integrato con il toolchain MDK-ARM per compilare e caricare un progetto C sui MCU STM32.

- **STM32CubeIDE** di STMicroelectronics (versione 1.3.0 o successive): È un ambiente di sviluppo software per MCU della casa produttrice STMicroelectronics, al suo interno vi sono le funzionalità di STM32CubeMX per la generazione del codice di inizializzazione. Inoltre possiede un tool di tracciamento attivo delle variabili, oltre al compilatore e debbugger di default.

- **IAR Embedded Workbench for Arm** di IAR (versione 8.50.1 o successive): IAR Embedded Workbench for Arm è un tool di debugging, in alternativa a KEIL uVision IDE. Il tool è a pagamento, ma fornisce una prova di 30 giorni.

- **CoolTerm** di Xojo (versione 1.7.0 o successive): CoolTerm è una semplice applicazione terminale per porta seriale (non è un emulatore di terminale) rivolta ad hobbisti e professionisti con la necessità di scambiare dati con hardware collegato a porte seriali.

4.2 Hardware

Il relatore a messo a disposizione una scheda (vedi Figura 4.1) sviluppata in SUPSI, la quale dispone di 4 processi che poi dovranno essere controllati tramite un microcontrollore interfacciato con Simulink.

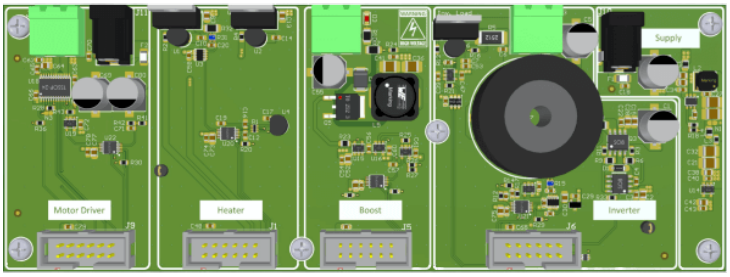


Figura 4.1: Scheda controllo

STUDENTSUPSI

4.2.1 Scelta hardware

Data la necessità di lavorare con un tool come Simulink per l'auto generazione di codice, l'obiettivo era quello di trovare un hardware adatto. Su consiglio del relatore si è optato di sfruttare un tool STM32-MAT/TARGET messo a disposizione da ST microelettronics per l'auto generazione di codice tramite Simulink. Il tool ad oggi non è compatibile con tutte le evaluation board di ST ma solo con alcune. Tra le compatibili si è scelta la scheda più performante, tenendo sempre presente le periferiche necessarie al controllo dei processi. Le periferiche di seguito elencate permettono di controllare almeno un processo per volta.

- 1 x USART
- 1 x SPI
- 1 x I2C
- 2 x PWM
- 2 x PIN_Output
- 1 x PIN_Input

La scheda scelta é la **STM32F746ZG**, la quale soddisfa ampiamente le caratteristiche minime sopra elencate. La Figura 4.2 rappresenta la scheda scelta.

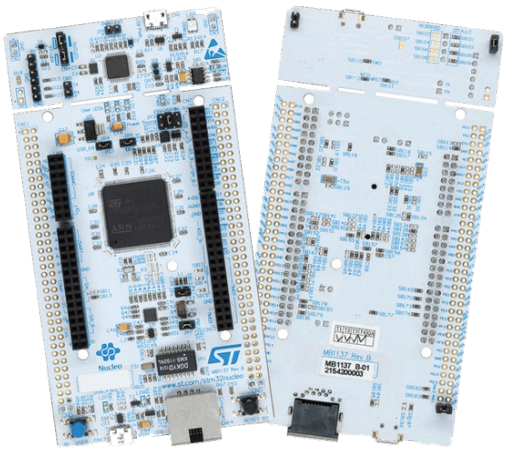


Figura 4.2: Evaluation board STM32F746ZG

STUDENTSUPSI

Come rappresentato nella Figura 4.3 le periferiche sono addirittura sufficienti a controllare uno o più processi simultaneamente. Bisogna però tenere in considerazione che non tutte le periferiche sono attivabili contemporaneamente.

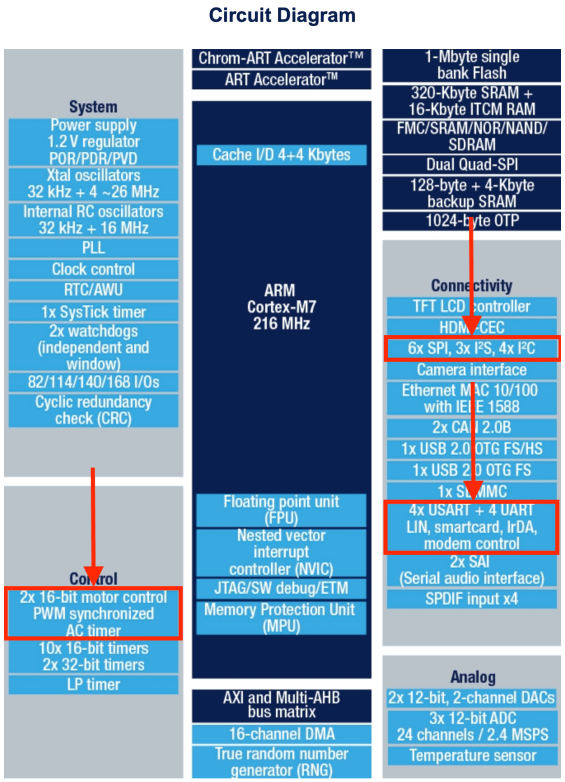


Figura 4.3: Evaluation board STM32F746ZG dati tecnici

STUDENTSUPSI

Capitolo 5

Test scheda di controllo

La scheda di controllo sviluppata in SUPSI da Alessandro Vaghi è la prima versione realizzata, di conseguenza come richiesto e definito dagli obbiettivi si deve testare la scheda per il suo corretto funzionamento. La scheda è suddivisa in 5 parti rappresentate dai 4 processi più l'alimentazione principale. L'obbiettivo è identificare eventuali problemi di design o di funzionalità.

5.1 Test alimentazione principale

In questo caso, i test da effettuare sull'alimentazione principale sono:

- Verificare le tensioni che andranno a fornire energia ai 4 processi prima si saldare il ponte N1

- Verificare la tensione sull'IC del processo Driver collegando un secondo alimentatore

In Figura 5.1 sono rappresentati i punti di misura rilevati.

STUDENTSUPSI

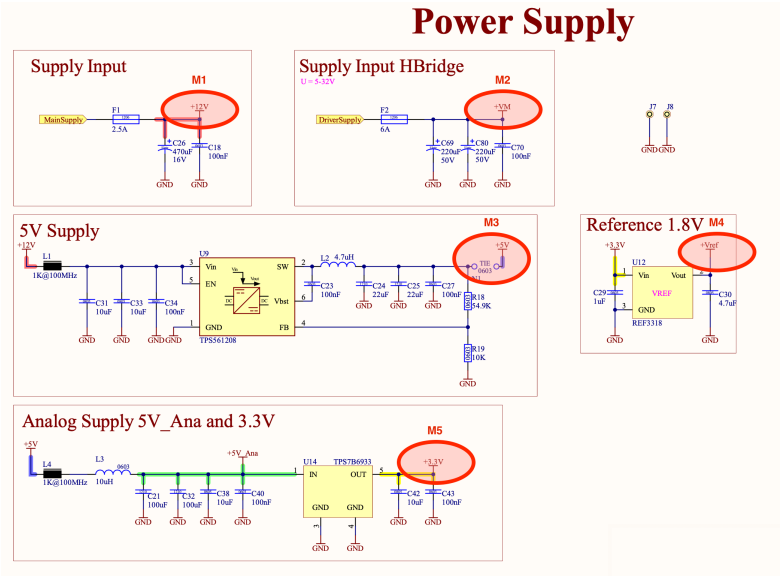


Figura 5.1: Misure effettuate sull'alimentazione

Appena si è collegato l'alimentatore alla scheda il fusibile F1 è saltato, per ovviare al problema si è rimpiazzato il fusibile con un ponte. Con questo accorgimento le misure effettuate sulla scheda rientrano nel range di valori accettabili, come dimostra la Tabella 5.1

Punto di misura	Valore atteso	Valore misurato
M1	12 [V]	12.1 [V]
M2	12 [V]	12.1 [V]
M3	5 [V]	4.8 [V]
M4	1.8 [V]	1.6 [V]
M5	3.3 [V]	3.0 [V]

Tabella 5.1: Rilevamento tensioni di alimentazione

Per la misura nel punto M2 è stato utilizzato il medesimo alimentatore utilizzato per il power supply principale, di conseguenza è stato dimostrato che in quel specifico caso la tensione corrisponde. Non significa che è garantito per tutto il range possibile (U = 5-32[V]).

5.2 Test del processo Heater

Per una scrupolosa verifica si controllano le tensioni su ogni singolo IC per il processo Heater, i punti di misura sono indicati sulla Figura 5.2.

STUDENTSUPSI

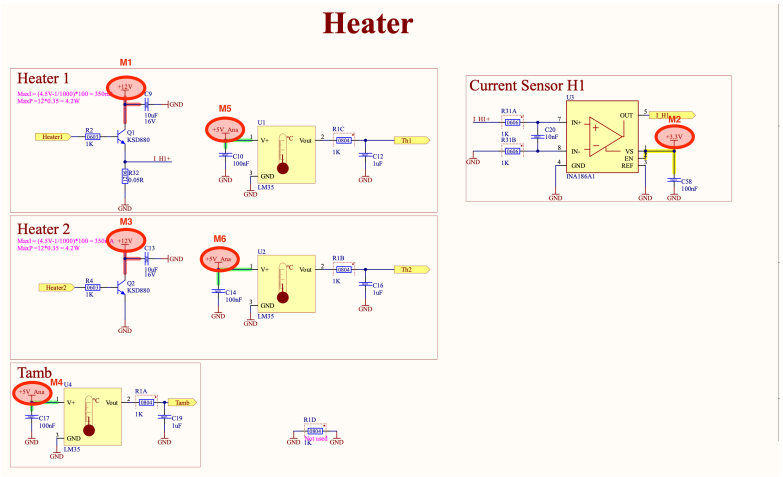


Figura 5.2: Misure effettuate sul processo Heater

La Tabella 5.2 riassume i valori attesi e le misure effettive eseguite sulla scheda, le tensioni rilevate rientrano nelle aspettative.

Punto di misura	Valore atteso	Valore misurato
M1	12 [V]	12.1 [V]
M2	3.3 [V]	3.0 [V]
M3	12 [V]	12.1 [V]
M4	5 [V]	4.8 [V]
M5	5 [V]	4.8 [V]
M6	5 [V]	4.8 [V]

Tabella 5.2: Rilevamento tensioni IC Heater

Con l'ausilio del software STM32CubeIDE si è sviluppato il test di funzionamento, attraverso scrittura e lettura via SPI; è in grado di testare le seguenti funzionalità del sistema:

- Scrittura e lettura dell'ADC
- Switching del canale di conversione
- Switching dei transistor

Il codice sviluppato permette di abilitare e disabilitare i transistor Q1 e Q2 tramite due segnali PWM, il secondo stadio è quello di acquisire i dati dei 4 canali, ed infine vengono inviati tramite seriale su di un terminale.
Il progetto STM32CubeIDE è in allegato al seguente percorso
Cartella: Software/STM32CubeIDE/Heater_test, il funzionamento è descritto dal flowchart in Figura 5.3.

STUDENTSUPSI

Il test non ha rilevato alcun problema di funzionamento, il processo risulta dunque funzionante come da aspettativa.

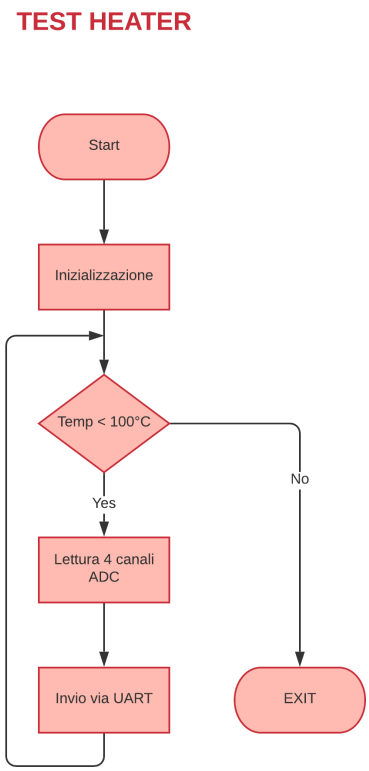


Figura 5.3: Flowchart test Heater

Nella fase di inizializzazione vengono fatti partire due segali PWM constanti con duty-cicle diversi, inizializzati precedentemente nella fase di impostazioni delle periferiche. Inoltre si manda sulla porta seriale la denominazione dei dati che verranno inviati.

5.3 Test del processo Inverter

Come per l'Heater effettuiamo un'ulteriore verifica, controllando le tensioni su ogni singolo IC per il processo Inverter, i punti di misura sono indicati sulla Figura 5.4.

STUDENTSUPSI

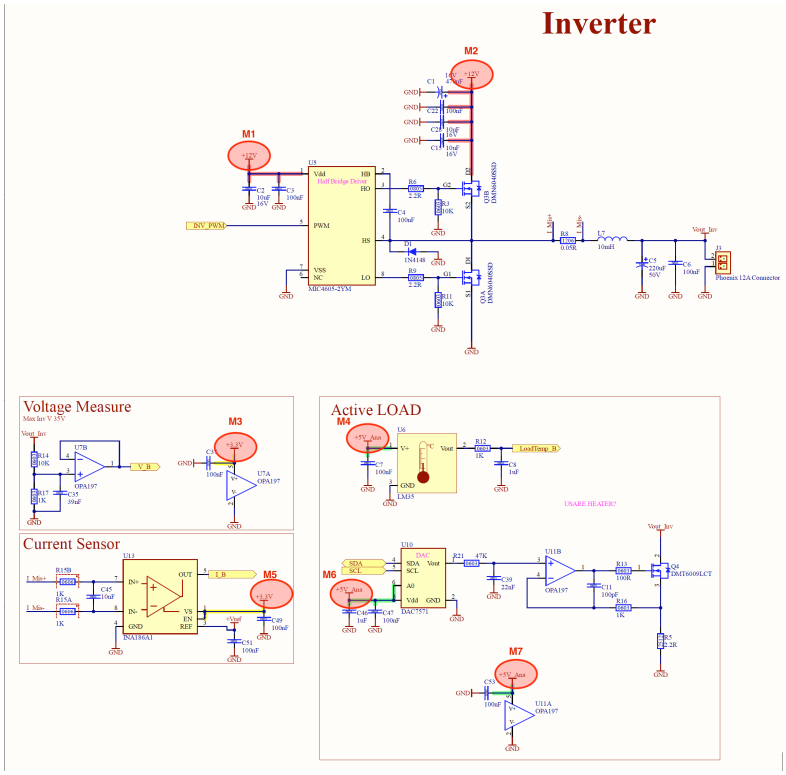


Figura 5.4: Misure effettuate sul processo Inverter

La Tabella 5.3 riassume i valori attesi e le misure effettive eseguite sulla scheda, le tensioni rilevate rientrano nelle aspettative.

Punto di misura	Valore atteso	Valore misurato
M1	12 [V]	12.1 [V]
M2	12 [V]	12.1 [V]
M3	3.3 [V]	3.0 [V]
M4	5 [V]	4.8 [V]
M5	3.3 [V]	3.0 [V]
M6	5 [V]	4.8 [V]
M7	5 [V]	4.8 [V]

Tabella 5.3: Rilevamento tensioni IC Inverter

Con l'ausilio del software STM32CubeIDE si è sviluppato il test di funzionamento, attraverso scrittura e lettura via SPI per l'ADC, e tramite scrittura I2C inviare una sinusoide al DAC; è in grado di testare le seguenti funzionalità del sistema:

- Scrittura e lettura dell'ADC
- Switching del canale di conversione

- Scrittura via I2C su DAC

La scheda di controllo fornita non presenta la bobina L7, per questo motivo il test non può essere completato al 100%. Il codice è sviluppato per eseguire un test completo, evidentemente i risultati devono tenere in considerazione la mancanza della bobina. Una possibilità resta quella di inserire un ponte.
Il progetto STM32CubeIDE è in allegato al seguente percorso
Cartella: Software/STM32CubeIDE/Iverter_test, il funzionamento è descritto dal flowchart in Figura 5.5.

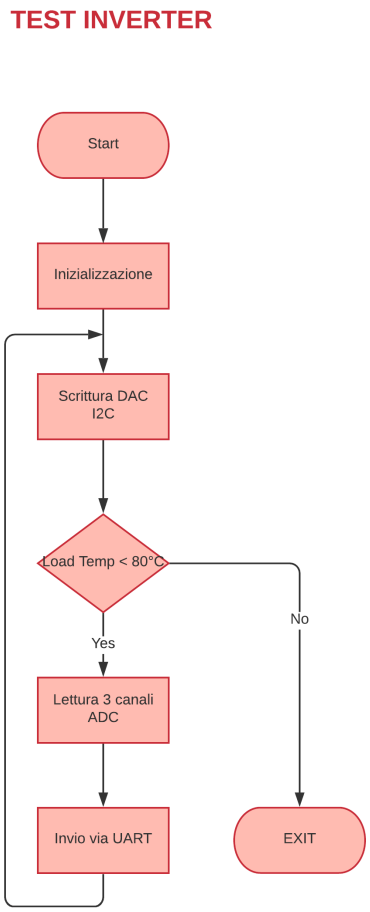


Figura 5.5: Flowchart test Inverter

Nella fase di inizializzazione del codice viene avviato un segnale PWM costante, inizializzato precedentemente nella fase di impostazioni delle periferiche. Inoltre si manda sulla porta seriale la denominazioni dei dati che verranno inviati.

STUDENTSUPSI

5.4 Test del processo Booster

Come per i processi precedenti effettuiamo un'ulteriore verifica, controllando le tensioni su ogni singolo IC per il processo Booster, i punti di misura sono indicati sulla Figura 5.6.

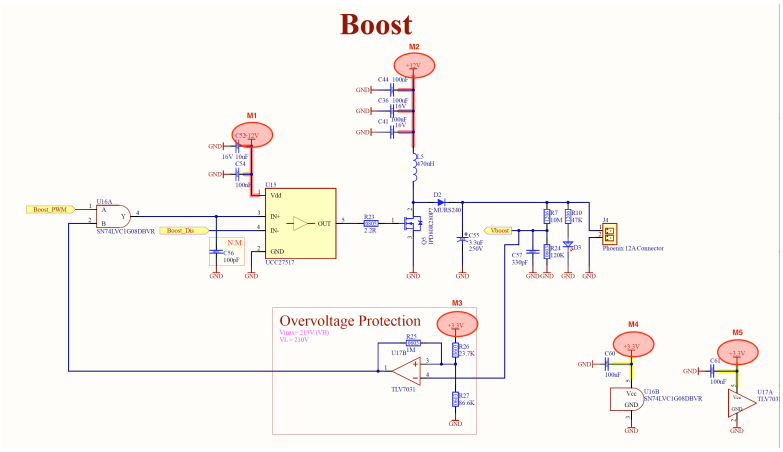


Figura 5.6: Misure effettuate sul processo Booster

La Tabella 5.4 riassume i valori attesi e le misure effettive eseguite sulla scheda, le tensioni rilevate rientrano nelle aspettative.

Punto di misura	Valore atteso	Valore misurato
M1	12 [V]	12.1 [V]
M2	12 [V]	12.1 [V]
M3	3.3 [V]	3.0 [V]
M4	3.3 [V]	3.0 [V]
M5	3.3 [V]	3.0 [V]

Tabella 5.4: Rilevamento tensioni IC Booster

Con l'ausilio del software STM32CubeIDE si è sviluppato il test di funzionamento, attraverso la scrittura e lettura via SPI; è in grado di testare la seguente funzionalità del sistema:

- Scrittura e lettura ADC

Con degli accorgimenti al codice di test del processo Heater, lo si è potuto riciclare in questa situazione.
Il progetto STM32CubeIDE è in allegato al seguente percorso
Cartella: Software/STM32CubeIDE/Booster_test, il funzionamento è descritto dal flowchart in Figura 5.7.

STUDENTSUPSI

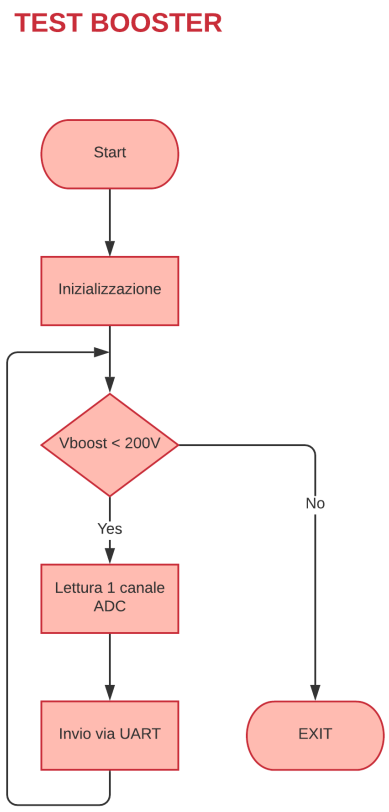


Figura 5.7: Flowchart test Booster

Nella fase di inizializzazione del codice viene avviato un segnale PWM costante, inizializzato precedentemente nella fase di impostazioni delle periferiche, e fissato il livello logico del pin Boost_Dis. Inoltre si manda sulla porta seriale la denominazione dei dati che verranno inviati.

5.5 Test processo Driver

Come per i processi precedenti effettuiamo un'ulteriore verifica, controllando le tensioni su ogni singolo IC per il processo Driver, i punti di misura sono indicati sulla Figura 5.8.

STUDENTSUPSI

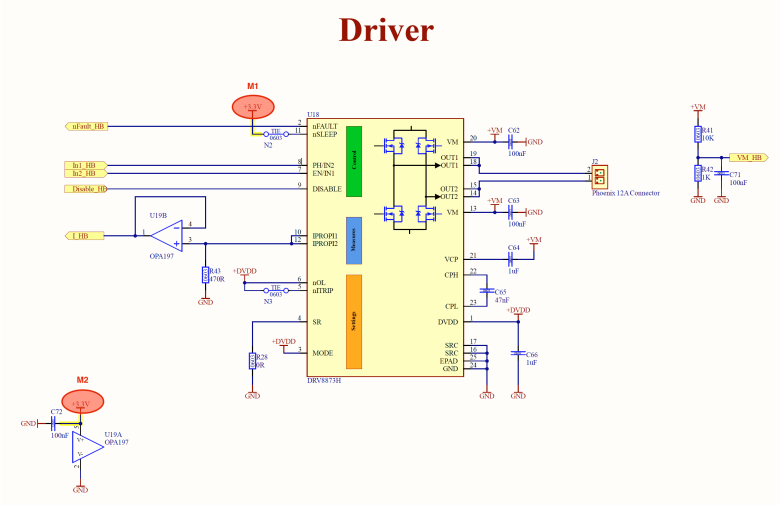


Figura 5.8: Misure effettuate sul processo Driver

La Tabella 5.5 riassume i valori attesi e le misure effettive eseguite sulla scheda, le tensioni rilevate rientrano nelle aspettative.

Punto di misura	Valore atteso	Valore misurato
M1	3.3 [V]	3.0 [V]
M2	3.3 [V]	3.0 [V]

Tabella 5.5: Rilevamento tensioni IC Driver

Con l'ausilio del software STM32CubeIDE si è sviluppato il test di funzionamento, attraverso scrittura e lettura via SPI; è in grado di testare la seguente funzionalità del sistema:

- Scrittura e lettura dell'ADC
- Switching del canale di conversione

Con degli accorgimenti al codice di test del processo Heater, lo si è potuto riciclare in questa situazione.
Il progetto STM32CubeIDE è in allegato al seguente percorso
Cartella: Software/STM32CubeIDE/Driver_test, il funzionamento è descritto dal flowchart in Figura 5.9.

STUDENTSUPSI

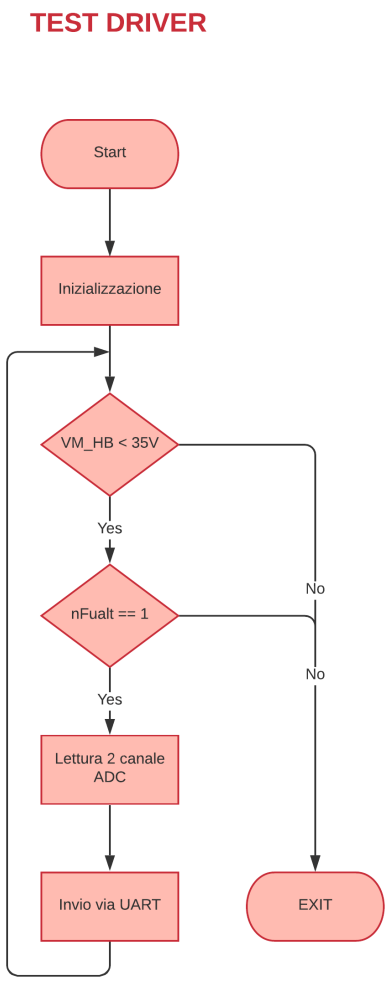


Figura 5.9: Flowchart test Driver

Nella fase di inizializzazione del codice viene avviato un segnale PWM costante, inizializzato precedentemente nella fase di impostazioni delle periferiche, e fissato il livello logico dei pin:

- In1_HB
- In2_HB
- Disable_HB

Inoltre si manda sulla porta seriale la nomenclature dei dati che verranno inviati.

5.6 Osservazioni

Il problema principale dell'IC presente sulla scheda di controllo è che non permette di leggere in una singola lettura tutti i canali, ma bensì prima si deve configurare il canale desiderato

STUDENTSUPSI

poi fare una lettura dopo aver atteso un tempo minimo di setting pari circa a ($\sim 250[\mu s]$). Come riportato nel Capitolo 10 sarà problematica da risolvere nella prossima versione della scheda di controllo.
Questo problema lo si nota maggiormente nei processi rapidi come possono essere:

- Inverter
- Booster (In questo caso nessun problema perchè non si deve cambiare canale dell'A-DC)
- Driver

Nel processo Heater, essendo un sistema molto lento, non sussiste alcun problema.

STUDENTSUPSI

STUDENTSUPSI

Capitolo 6

Modellizzazione

Il modello del sistema è la parte fondamentale di ogni tipo di controllore, senza un modello matematico oppure un identificazione del processo non si può sviluppare un controllore. Vi sono due metodi per modellare un sistema:

- Modellizzazione attraverso le leggi della fisica
- Modellizzazione attraverso metodi di identificazione

Il primo metodo elencato permette di modellare sistemi semplici, non è adatto per sistemi complessi. Mentre il secondo metodo permette l'identificazione anche di modelli complessi attraverso set di dati raccolti in funzione di ingressi determinatati. In questo capitolo trattiamo il primo metodo ovvero la modellizzazione attraverso le leggi della fisica.

6.1 Modelli Heater

Il processo di riscaldamento "Heater" è un processo abbastanza complesso, in questo senso modellarlo attraverso le leggi della fisica è molto più complesso che attraverso un'identificazione. Il processo completo presenta una situazione di tipo MIMO ovvero Multi Input Multi Output.

6.1.1 Modello Heater SISO

In questa prima parte trattiamo invece il processo come SISO ovvero Single Input Single Output. In quest'ottica il processo presenta un elemento riscaldante (transistor), un sensore accoppiato al transistor e infine un dissipatore anch'esso accoppiato al transistor.
Si suppone che il dissipatore e il transistor siano un unico elemento. Con questa assunzione si semplifica leggermente il modello. Breve commento sulla dinamica del processo: il dissipatore dissipa il calore per irraggiamento e per conduzione nell'ambiente circostante, la quantità di calore disperso dipende dalla superficie esposta.

STUDENTSUPSI

Le leggi della fisica che governano questo sistema sono:
Irraggiamento:

$$Q = \epsilon\sigma A \cdot \Delta T^4 \tag{6.1}$$

Conduzione:

$$Q = UA \cdot \Delta T \tag{6.2}$$

Il modello matematico risulta[1]:

$$\begin{cases} mc_p \frac{\partial T_{H1}}{\partial t} = UA \cdot (T_a - T_{H1}) + \epsilon\sigma A \cdot (T_a - T_{H1})^4 + \alpha_1 Q \\ \tau \frac{\partial T_{C1}}{\partial t} = T_{H1} - T_{C1} \end{cases} \tag{6.3}$$

Dove τ vale:

$$\tau = \frac{m_s c_{p,s} \Delta x}{k A_c}$$

Il sistema è chiaramente non lineare data la presenza di un termine elevato alla quarta. Per lo sviluppo del controllore bisogna dunque linearizzare il sistema attorno al punto di lavoro.

6.1.2 Modello Heater MIMO

Come accennato precedentemente il sistema MIMO è un sistema complesso. Difatti entra in gioco anche lo scambio di calore tra i due transistor, quindi si deve considerare l'iterazione fra i due elementi riscaldanti, inoltre resta ancora la dissipazione nell'ambiente come nel caso SISO [2].

$$\begin{cases} mC_p \cdot \frac{\partial T_{H1}}{\partial t} = UA(T_a - T_{H1}) + \epsilon\sigma A(T_a^4 - T_{H1}^4) + U_s A_s (T_{H2} - T_{H1}) + \epsilon\sigma A_s (T_{H2}^4 - T_{H1}^4) + \alpha_1 Q_1 \\ mC_p \cdot \frac{\partial T_{H2}}{\partial t} = UA(T_a - T_{H2}) + \epsilon\sigma A(T_a^4 - T_{H2}^4) - U_s A_s (T_{H2} - T_{H1}) - \epsilon\sigma A_s (T_{H2}^4 - T_{H1}^4) + \alpha_2 Q_2 \\ \tau \frac{\partial T_{C1}}{\partial t} = (T_{H1} - T_{C1}) \\ \tau \frac{\partial T_{C2}}{\partial t} = (T_{H2} - T_{C2}) \end{cases} \tag{6.4}$$

Dove τ vale:

$$\tau = \frac{m_s c_{p,s} \Delta x}{k A_c}$$

6.2 Modello Inverter

Il processo Inverter presente sulla scheda lo si può considerare come un sistema semplice, in questo caso è vantaggioso modellare il sistema attraverso le leggi della fisica, o meglio

[1] Le nomenclature e i valori dei dati utilizzati sono riassunti nella Tabella G.1.
[2] La soluzione proposta nell'Appendice D presenta i passagi per la linearizzazione.

STUDENTSUPSI

le leggi che governano i circuiti elettrici. Nella Figura 6.1 è rappresentato il circuito da modellare.

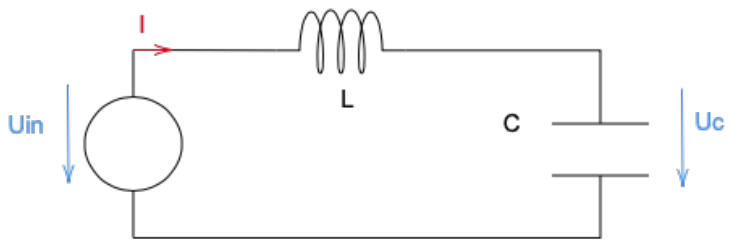


Figura 6.1: Circuito di modellizzazione Inverter

Utilizzando l'analisi circuitale otteniamo la seguente relazione matematica che descrive il comportamento della tensione in uscita U_c in funzione dell'ingresso ovvero la tensione U_{in}.

$$\begin{cases} U_L = L \cdot \dfrac{\partial I}{\partial t} \\ I_C = C \cdot \dfrac{\partial U_c}{\partial t} \end{cases} \rightarrow \begin{cases} U_{in} = U_L + U_C \\ I = I_L = I_C \end{cases} \rightarrow \begin{cases} L \cdot \dot{I}_L = U_{in} - U_c \\ C \cdot \dot{U}_C = I_L \end{cases} \tag{6.5}$$

STUDENTSUPSI

STUDENTSUPSI

Capitolo 7

Identificazione

Per identificare un processo necessitiamo di un data-set di dati in funzione di un segnale di stimolo. Vi sono molteplici sistemi di identificazione, sistemi di tipo single shot come possono essere:

- il metodo di Least-Square
- il metodo della regressione polinomiale

oppure vi sono sistemi di tipo recursivo come possono essere:

- il metodo recursive Least-Square
- il metodo di Gauss-Newton

Per identificare i processi in questo capitolo i metodi utilizzati sono Lest-Square oppure una regressione polinomiale.

7.1 Heater

Il processo Heater è pilotato da un segnale PWM in entrata ed in uscita si ottiene una temperatura. Il segnale PWM in ingresso permette la regolazione della potenza assorbita dal transistor. L'obbiettivo dell'identificazione in questo caso è mettere in relazione il duty-cycle del PWM con la temperatura in uscita.
Per ottenere questo si necessita di un segnale PWM costante, attendere che il processo si stabilizzi ad una determinata temperatura ed infine acquisire i dati.
Con l'ausilio dell'evaluation board si realizzano i segnali di stimolo e l'acquisizione dei dati, mentre l'elaborazione dei dati avviene tramite MATLAB. Il processo di acquisizione è rappresentato nella Figura 7.1

STUDENTSUPSI

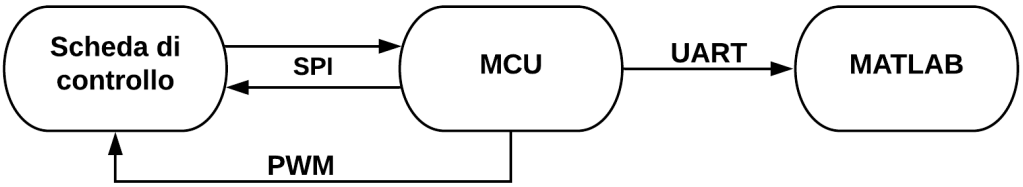


Figura 7.1: Processo di acquisizione dati

7.1.1 Identificazione heater SISO

Il processo heater SISO permette di fare un'identificazione attraverso un unico segnale PWM di stimolo come rappresenta la Figura 7.2, a differenza di come si può pensare per il processo MIMO.

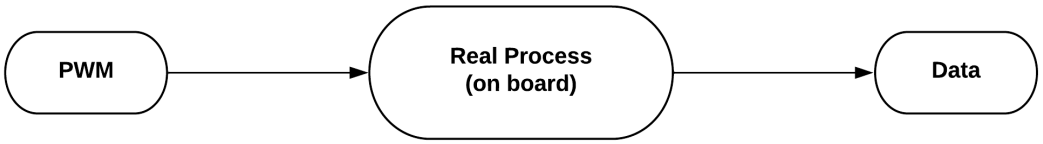


Figura 7.2: Modello identificazione heater SISO

Il codice di acquisizione dei dati sviluppato permette di generare un PWM variabile e di leggere il valore della temperatura tramite SPI e poi di trasmetterlo al PC tramite porta seriale, come da Figura 7.1. Con l'ausilio del software CoolTermWin si salvano i dati in un file.txt, importabile poi in MATLAB per l'elaborazione dei dati.

7.1.1.1 Elaborazione dati heater SISO

Per affrontare l'elaborazione dei dati si sono fatte alcune supposizioni per semplificare l'identificazione, ovvero:

- Il processo non è affetto da alcun tipo di disturbo
- Il sistema di acquisizione dei dati non presenta alcun tipo di errore, come errori di:
  - Approssimazione
  - Quantizzazione

STUDENTSUPSI

grazie a queste supposizioni, si può usare il metodo di Least-Square. Il metodo di Least-Square richiede una prima costatazione, ovvero l'ordine o grado della funzione che si vuole identificare. In questo caso si è optato per una funzione di secondo ordine di questo tipo:

$$y[k] = b0 \cdot u[k] + b1 \cdot u[k-1] + b2 \cdot u[k-2] - a1 \cdot y[k-1] - a2 \cdot y[k-2] \tag{7.1}$$

A questo punto si identifica il regressore e il vettore dei parametri, in modo tale da ottenere questa forma:

$$Y = \Phi^T \cdot \Theta \tag{7.2}$$

In questo caso:

$$\begin{bmatrix} y[1] \\ y[2] \\ \vdots \\ y[k] \end{bmatrix} = \begin{bmatrix} u[0] & u[-1] & u[-2] & -y[-1] & -y[-2] \\ u[1] & u[0] & u[-1] & -y[0] & -y[-1] \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u[k] & u[k-1] & u[k-2] & -y[k-1] & -y[k-2] \end{bmatrix} \cdot \begin{bmatrix} b0 \\ b1 \\ b2 \\ a1 \\ a2 \end{bmatrix} \tag{7.3}$$

Dove:

- Y $\rightarrow$ Vettore dati
- $\Phi^T \rightarrow$ Regressore
- $\Theta \rightarrow$ Vettore parametri

Ora per identificare i parametri è sufficiente applicare la seguente formula:

$$\Theta_{LS} = (\Phi \cdot \Phi^T)^{-1} \cdot \Phi \cdot Y \tag{7.4}$$

Con MATLAB tutto il procedimento è stato automatizzato attraverso un script, è sufficiente importare il file con i dati raccolti ed eseguire il programma. Il codice MATLAB lo si può trovare nel seguente percorso all'interno della cartella di progetto **Cartella: Software/Heater/Identificazione**

STUDENTSUPSI

STUDENTSUPSI

Capitolo 8

Sviluppo software

Lo scopo di questo capitolo è quello di spiegare l'approccio affrontato nello sviluppo del codice di comunicazione tra i vari software, le problematiche riscontrate e i test eseguiti durante tutto il progetto.

8.1 Obbiettivo

Come citato nelle specifiche, l'obbiettivo principale è quello di utilizzare il toolbox STM32-MAT/TARGET per sviluppare i dei schemi Simulink di ogni singolo processo. Questi blocchetti devono poter comunicare con la scheda di controllo, evidentemente ogni processo sulla scheda ha le sue caratteristiche da rispettare. Per questo motivo il relatore ha voluto approcciare lo sviluppo del codice processo per processo, portandone a termine uno alla volta e testarne la completa funzionalità.
Lo sviluppo di questi blocchetti deve permettere l'auto-generazione di codice da Simulink, così che il codice ottenuto sia pronto da scaricare sull'evaluation board e verificarne il funzionamento del controllore sviluppato.

8.1.1 Visione del funzionamento finale

Come di consuetudine per sviluppare un controllore si effettuano delle simulazioni tramite Simulink (o altri tool) con un modello matematico come in Figura 8.1 ottenuto tramite le leggi della fisica o tramite identificazione. In simulazione si verifica il funzionamento desiderato del controllore, di seguito una volta raggiunta la soluzione che meglio soddisfa le esigenze del sistema si deve testare sul sistema reale.
A questo punto sorge il problema della generazione di codice, ovvero se la simulazione ottenuta rispecchia a pieno la realtà, si può scrivere il codice di proprio pugno. Qualora invece vi sia un errore o un problema con il controllore una volta testato, si deve riprendere la simulazione, aggiustare il problema e riscrivere il codice. Questo automatismo porta ad

STUDENTSUPSI

un grande dispendio di tempo. Da qui la necessità di poter fare dell'auto-generazione di codice.

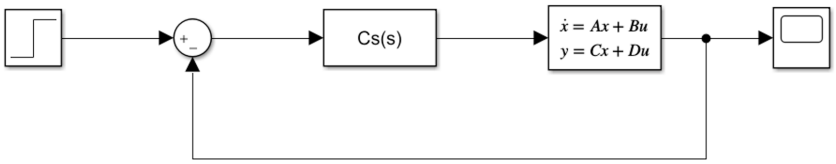


Figura 8.1: Simulazione Simulink

Per fare ciò in questo progetto serve un blocchetto Simulink che venga sostituito a quello matematico utilizzato in simulazione. Il codice all'interno di questo blocchetto deve poter scrivere e leggere i dati sulla scheda di controllo. Per esempio il caso del processo heater, se si considera il caso MIMO (il più completo), come minimo si deve poter inviare 2 segnali PWM e leggere le temperature dei 2 elementi riscaldanti (transistor). Questo perché il controllore deve poter interfacciarsi con il sistema reale, come nella Figura 8.2

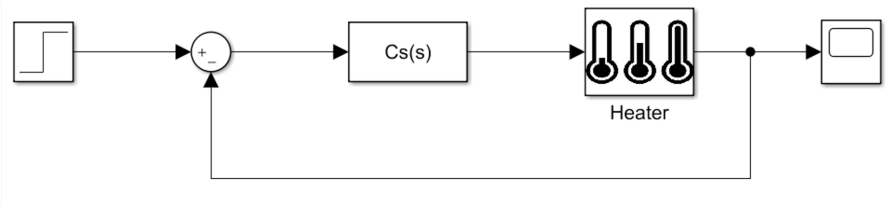


Figura 8.2: Modello controllore reale Simulink

Il processo finale che si vuole per l'utilizzatore è il seguente: costruzione del controllore attraverso simulazione, generazione di codice attraverso Simulink, compilazione/download del codice su un IDE e feedback su PC. La Figura 8.3 illustra il procedimento desiderato.

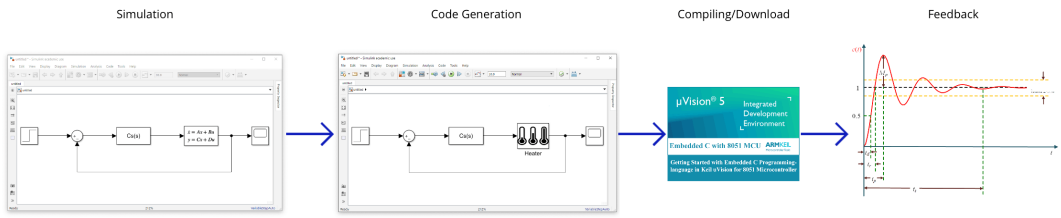


Figura 8.3: Processo generazione codice

STUDENTSUPSI

8.2 Introduzione alle S-Function

Le S-Function permettono nell'ambiente Simulink di generare dei blocchetti con funzioni personalizzate. Ce ne sono di vario tipo e sono racchiuse in 2 categorie principali. La prima categoria è la "M-File" la quale sfrutta il linguaggio MATLAB, la seconda categoria è la "C-MEX", la quale sfrutta il linguaggio C, questa categoria ha un vantaggio che durante l'elaborazione la funzione può essere eseguita più velocemente.
Nella S-Function si definiscono le entrate e le uscite desiderate nel blocchetto Simulink, nella Tabella 8.1 sono elencate le fasi di simulazione delle S-Function di tipo "M-File" e di tipo "C-MEX".

Fase di simulazione	C-MEX S-functio	M-File S-Function
Inizializzazione	mdlInitializeSizes	flag=0
Prossimo passo	mdlGetTimeOfNextVarHit	flag=4
Output	mdlOutputs	flag=3
Update stati discreti	mdlUpdate	flag=2
Calcolo derivate	mdlDerivatives	flag=1
Fine simulazione	mdlDerivatives	flag=9

Tabella 8.1: Fasi si simulazione S-Function

Le S-Function sono lo strumento ideale per realizzare sistemi con l'obbiettivo di generare codice da Simulink, difatti si può includere librerie già esistenti[1]. A tal proposito la S-Function scritta viene riportata di pari passo nel codice generato da Simulink.

8.3 Funzionamento del toolbox STM32MAT/TARGET

Il toolbox STM32MAT/TARGET fornisce alcuni blocchetti da importare in Simulink i quali danno accesso alle periferiche desiderate. Il toolbox necessita il supporto di almeno altri 2 software per essere funzionale. Uno di questi è l'STM32CubeMX il quale permette di selezionare tutte le periferiche da interfaccia grafica, e inoltre definire il software di debugging che si vuole utilizzare, la scelta comprende la seguente lista di software:

- EWARM
- MDK-ARM
- SW4STM32
- TrueSTUDIO
- STM32CubeIDE
- Makefile
- Other Toolchains (GPDSC)

[1] Per Esempio: librerie di HAL per processori STM32

Di conseguenza ecco il secondo software che il toolbox necessita, il software selezionato permette di compilare il codice generato da Simulink e di scaricarlo sulla scheda.

8.3.1 Generazione file.ioc

Il file.ioc è un file generato da STM32CubeMx il quale contiene tutte le informazioni riguardanti le periferiche abilitate, la gestione dei clock interni e la referenza all'IDE che si intende usare.
Se si desidera generare codice da Simulink con il toolbox STM32MAT/TARGET serve un file.ioc, con le configurazioni delle periferiche necessarie, dai GPIO ai protocolli di comunicazione, tutte le periferiche disponibili richiedono una configurazione adeguata dall'interfaccia grafica di STM32CubeMX, per esempio per il protocollo SPI si devono configurare le caratteristiche della comunicazione, eccone alcune:

- Modalità: Full-Duplex Master, Full-Duplex Slave, HAlf-Duplex Master,...
- Data size: dimensione del vettore dato in bit
- Prescaler: per selezionare la velocità di trasmissione
- CHPOL: regola la polarità del clock
- CPHA: regola la fase del clock

Salvando il "progetto" autonomamente STM32CubeMX genera il file.ioc, poi da importare nella cartella di progetto Simulink.

Osservazione: il toolbox è completamente chiuso, i codici sorgenti non sono disponibili, e gli unici file a cui si può mettere mano sono alcuni file.tlc praticamente illeggibili.

8.3.2 Test del toolbox

In questo paragrafo si vuole mettere in funzione e verificare le funzionalità del toolbox, seguendo la procedura descritta nel manuale d'uso che si trova all'Appendice B. In Figura 8.4 è riportato lo schema a blocchi Simulink utilizzato per il test del toolbox.

STUDENTSUPSI

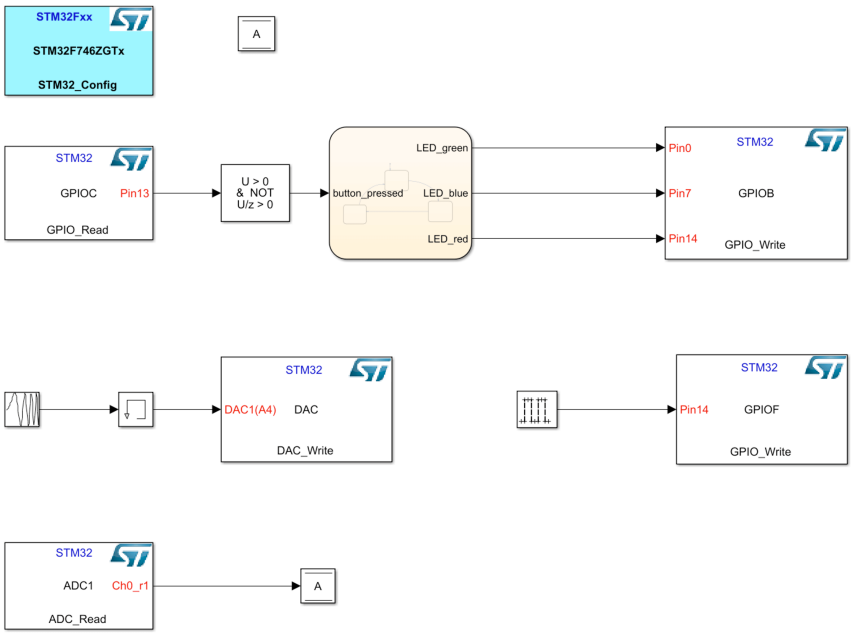


Figura 8.4: Test STM32MAT/TARGET

Con il test sviluppato verifichiamo che le periferiche principali funzionino, in questo caso si testano le seguenti periferiche:

- Lettura Pin
- Scrittura Pin
- Generazione segnale PWM
- Acquisizione segnale analogico (ADC)
- Scrittura di un segnale analogico (DAC)

Dopo aver generato il codice con Simulink, il codice viene compilato con il software EWARM e scaricato sulla scheda il pulsante di user cambia lo stato dei LED correttamente. Per verificare l'ADC introduciamo un segnale sinusoidale lento sul pin configurato (PA0), e con la modalità di debuggin tracciamo la variabile dove viene salvato il segnale. Mentre per il PWM e il DAC con l'oscilloscopio si misurano i relativi pin ovvero PWM (PF14) e DAC (PA4). Il file di test è disponibile nel seguente percorso: Tesi_Giannini/Software/EWARM_STM32_M_TARGET. Il sistema ha passato il test, per ora non sono ancora state testate le periferiche di comunicazione come: SPI, I2C e UART perché in corso d'opera ci si è resi conto che davano dei problemi che devono essere analizzati ulteriormente.

8.3.3 Analisi SPI

In un primo momento l'idea era quella di testare tutte le periferiche in una sola volta, ma purtroppo i protocolli di comunicazione davano subito dei problemi di generazione del codice da parte di Simulink, dunque si è deciso di analizzare unicamente l'SPI, in quanto è l'unico

STUDENTSUPSI

protocollo che serve per il funzionamento del sistema.
Data la struttura della scheda di controllo la quale presenta un ADC a 4 canali a conversione singola, si deve in un primo momento mandare all'ADC la configurazione di 16 bit i quali contengono le varie impostazioni oltre al canale che si vuole campionare, infine richiedere il valore via SPI. Il problema riscontrato è che la lettura del canale fornisce 16 bit di cui solamente 12 contengono le informazioni necessarie. Questo vuol dire che si necessita un elaborazione esterna dei dati tramite operatori bitwise.
Il toolbox STM32MAT/TARGET mette a disposizione 2 blocchetti SPI in Figura 8.5, senza alcuna descrizione dei tipi di dato attesi e forniti nelle rispettive entrate e uscite, come documentazione mettono a disposizione unicamente un esempio dove nemmeno viene descritto il funzionamento atteso. L'esempio lo si può trovare nell'help dei blocchetti.

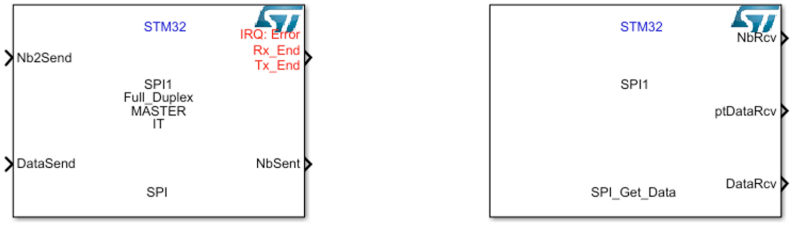


Figura 8.5: STM32MAT/TARGET SPI block

Il problema principale riscontrato nella generazione di codice è fornire il tipo di dato corretto, comunque tenendo conto della formattazione richiesta dal dispositivo slave. Durante la fase di debugging effettuando come primo step un test della generazione di codice si è cercato di far generare da Simulink unicamente la funzione minima del blocco SPI, la quale richiede in ogni caso anche il blocchetto di configurazione del MCU contenente il file.ioc. Lo scopo è quello una volta generato il codice analizzarlo e verificare come viene eseguita la traduzione da Simulink a codice C.

8.3.3.1 Problemi generazione codice

Simulink non permette di generare codice se le entrate e uscite principali del blocchetto non sono collegati. Per ovviare al problema si è dovuto costruire un sistema il quale permetta di generare il codice in modo da poterlo analizzare in seguito.
Tramite una S-Function, la quale fornisce alla sua uscita gli input necessari al blocchetto SPI, si è potuto generare codice da Simulink.

8.3.3.2 Analisi codice generato

Il codice generato da Simulink varia a seconda del IDE selezionato nel file.ioc importato. Nel caso dell'IDE KEIL uVision viene generata la struttura in Figura 8.6:

STUDENTSUPSI

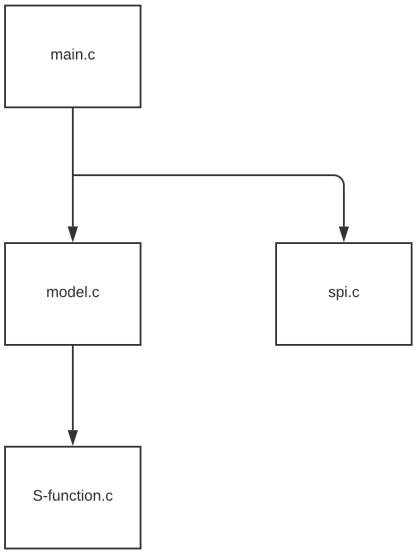


Figura 8.6: Struttura file generati

L'analisi ha evidenziato molteplici problemi nel codice generato. Un problema riscontrato è che nel file main.c non vengono chiamate le funzioni principali, un altro problema indica che in alcuni dei file generati non vengono incluse tutte le librerie necessarie. Sistemati i precedenti problemi avviando la compilazione dall'IDE la console identifica un problema con i casting dei puntatori al valore da inviare con SPI.
Si è tentato di risolvere il problema a monte cambiando il tipo di dato fornito dalla S-Function ma il risultato ottenuto, è un errore nella generazione di codice da parte di Simulink.

8.3.3.3 Osservazioni

La decisione da parte di ST di mantenere un toolbox gratuito con un sistema chiuso "isolato", non permette alcun margine di manovra per personalizzarlo e poterlo utilizzare in qualsiasi modo.
Date le innumerevoli difficoltà nella creazione di un automatismo snello da parte dell'utilizzatore finale, con il consulto del relatore si è deciso di passare all'opzione di generare codice ed ottenere un makefile.

8.3.4 Generazione di codice per makefile

Come già detto precedentemente, a causa delle problematiche riscontrate utilizzando il toolbox con l'IDE durante la compilazione, si è optato per la generazione di codice un modo da ottenere un makefile dove poter lavorare.

STUDENTSUPSI

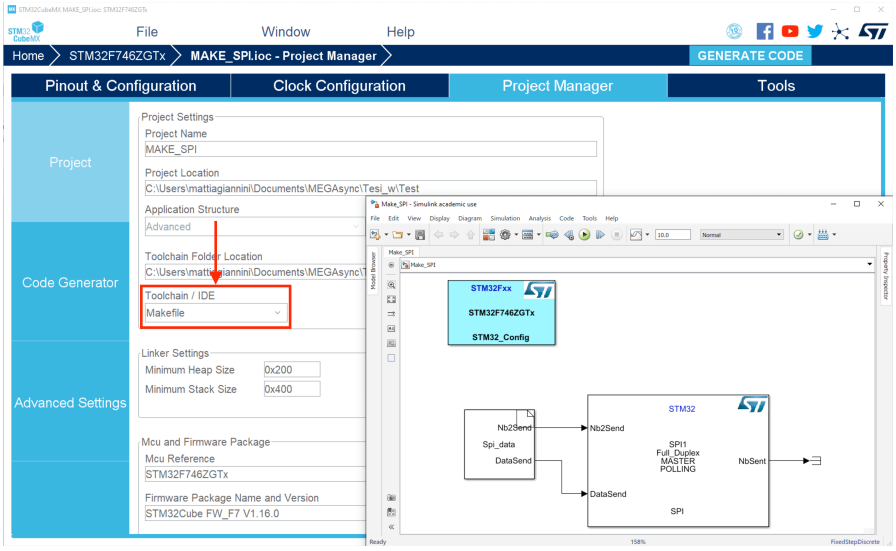


Figura 8.7: STM32MAT/TARGET SPI test makefile

Per ottenere il makefile come descritto precedentemente nell'introduzione a questa sezione é sufficiente scegliere il target makefile nella generazione del file.ioc, come da Figura 8.7 La generazione di codice da parte di Simulink non presenta le difficoltà riscontrate in precedenza. I problemi riscontrati in questo caso sono molteplici, ma a dispetto di prima sono nella fase di compilazione del codice. I problemi riscontrati sono:

- Impossibilità di compilazione da windows
- Mancanza di direttive di compilazione nel makefile
- Mancanza di alcune librerie in alcuni file
- Tipi di dati definiti più volte (collisione)
- Percorsi dei file relativi sbagliati siccome la compilazione avviene tramite linux

Per identificare i problemi sopra elencati si sono sviluppati altre versioni del codice:

- **MAKE_SPI_COM_1:** Tenta di leggere da SPI i quattro canali dell'ADC
- **MAKE_SPI_COM_1.1:** Tenta di leggere da SPI i quattro canali dell'ADC (correzione definizioni tipo di dati)
- **MAKE_SPI_COM_1.2:** Tenta di scrivere su LED valore 1 o 0 con le funzioni di HAL
- **MAKE_SPI_COM_1.3:** Tenta di scrivere via SPI sul registro dell'ADC

Sfruttando i codici sopra elencati sono state trovate le soluzioni agli errori di compilazione. Applicando le dovute correzioni che in ogni caso risultano essere diverse, almeno per quanto riguarda il percorso relativo dei vari file e librerie.

STUDENTSUPSI

Correggendo ogni singolo errore, infine si ottiene il file.elf scaricandolo sulla scheda STM32F746ZG tramite il software STM32CubeProgrammer non viene eseguito alcun tipo di programma. Le possibilità che questo accada sono 2: o il codice contiene errori, oppure la compilazione non ha generato il file correttamente.
Il codice si può trovare nella **Cartella: Sofware/Test/MAKE_SPI**. [2] Per identificare il problema di compilazione finale si è sviluppato un altro test il quale deve accendere e spegnere un LED sulla scheda. Sfruttando unicamente il blocchetto di scrittura su Pin (già testato in precedenza tramite generazione di codice per IDE), quindi "funzionante".
Il codice di test LED generato presenta le stesse problematiche elencate precedentemente, risolte anche per questo caso per procedere alla compilazione da makefile su linux. Il programma scaricato poi sulla scheda STM32F46ZG non viene eseguito. Il codice si può trovare nella **Cartella: Software/Test/MAKE_LED_ON**. [3]
Per sicurezza in modo da eliminare ogni ragionevole dubbio, il codice di accensione e spegnimento del LED è stato sviluppato in 4 versioni, ovvero:

- **MAKE_LED_ON:** unicamente utilizzando i blocchetti Simulink e ST (Generazione codice per makefile)
- **MAKE_LED_ON_sfunc:** utilizzando le funzioni di HAL in un'unica S-Function (Generazione codice per makefile)
- **MDK_LED_ON:** unicamente utilizzando i blocchetti Simulink e ST (Generazione codice per IDE)
- **MDK_LED_ON_sfunc:** utilizzando le funzioni di HAL in un'unica S-Function (Generazione codice per IDE)

In tutte e 4 le versioni il problema persiste, la compilazione da makefile non è in grado di generare un file.elf consistente, o meglio funzionante. Un'ulteriore conferma è data dalle dimensioni del file.elf generato tramite makefile e un codice di test (con la stessa funzione di accensione e spegnimento di un LED) sviluppato interamente su STM32CubeIDE, la differenza riscontrata è di oltre 1 [MB] fra i 2 file. Ovvero il file.elf generato da STM32CubeIDE pesa 1 [MB] in più, il confronto è stato fatto mantenendo la medesima configurazione delle periferiche, in questo caso è stato abilitato in entrambi i casi un unico Pin.
I file messi a confronto si possono trovare nei seguenti percorsi nella cartella di progetto:

- Software/STM32CubeIDE/Blinking_LED/Debug (Blinking_LED.elf IDE 1.17 [MB])
- Software/Test/MAKE_LED_ON/build (MAKE_LED_ON.elf makefile 188 [KB])

[2] Il makefile potrebbe essere quello nativo generato da Simulink senza le dovute correzioni
[3] Il makefile potrebbe essere quello nativo generato da Simulink senza le dovute correzioni

STUDENTSUPSI

8.3.5 Osservazioni toolbox STM32MAT/TARGET

I test eseguiti dimostrano che il toolbox non presenta una valida soluzione per ottenere una piattaforma didattica adeguata allo scopo. Il problema principale riscontrato dipende soprattutto dal fatto è che il toolbox non è open source, e questo rende impossibile correggere qualsiasi bug, e di comprenderne a pieno le funzionalità. Inoltre ST non ha rilasciato un manuale esaustivo, per questo motivo il lavoro di comprensione delle funzionalità ha richiesto più tempo del previsto.
Il massimo risultato ottenibile con questo toolbox è una generazione di codice incompleta, alla quale ogni volta si deve rimediare alle problematiche prima di poter avviare una compilazione che possa essere terminata con successo. Oltre al fatto, che paradossalmente la generazione di codice per STM32CubeIDE (IDE di ST), presenta molti più problemi che per IDE appartenenti a terzi.
La mancanza di una documentazione adeguata rende difficile il saper utilizzare soprattutto i blocchetti di comunicazione, in questo caso la lettura tramite il blocchetto SPI non è sufficiente, il dato letto deve essere ancora elaborato prima dell'utilizzo. Per questo motivo anche scrivere una S-Function con l'unico obbiettivo di elaborare il dato, senza sapere il tipo di dato ritornato dal blocchetto SPI risulta complicato.

In conclusione anche supponendo che la generazione di codice non dia problemi, e che il codice generato unicamente con i blocchetti del tooolbox sia compilabile, non si può sviluppare un blocchetto con le caratteristiche richieste dal processo senza l'ausilio di una S-Function, per gestire la formattazione dei dati in ingresso ed in uscita dall'IC ads1018 (ADC). Con questa situazione si ritorna ai problemi riscontrati precedentemente quando si includono S-Function nello schema si Simulink, ovvero che l'associare i tipi di dato corretti ai blocchetti SPI genera problemi o di generazione di codice, o di compilazione del codice generato, quest'ultimo presenta i seguenti problemi:

- Inclusione di alcune librerie assente
- Funzioni generate ma non chiamate all'interno del main.c

STUDENTSUPSI

Capitolo 9

PysimCoder

Pysimcoder è un tool che sfrutta la tecnica di Model-Base Design, che permette di effettuare simulazioni e di generare codice, come Simulink, a differenza del fatto che è open source. Il professor Bucher ha utilizzato questo sistema con l'obbiettivo di far generare automaticamente codice da PysimCoder per l'evaluation board STM32H745ZI in Figura 9.1. Il sistema è ancora in fase di sviluppo ma è già in grado di fornire sufficienti blocchetti PysimCoder per ottenere una demo funzionate.

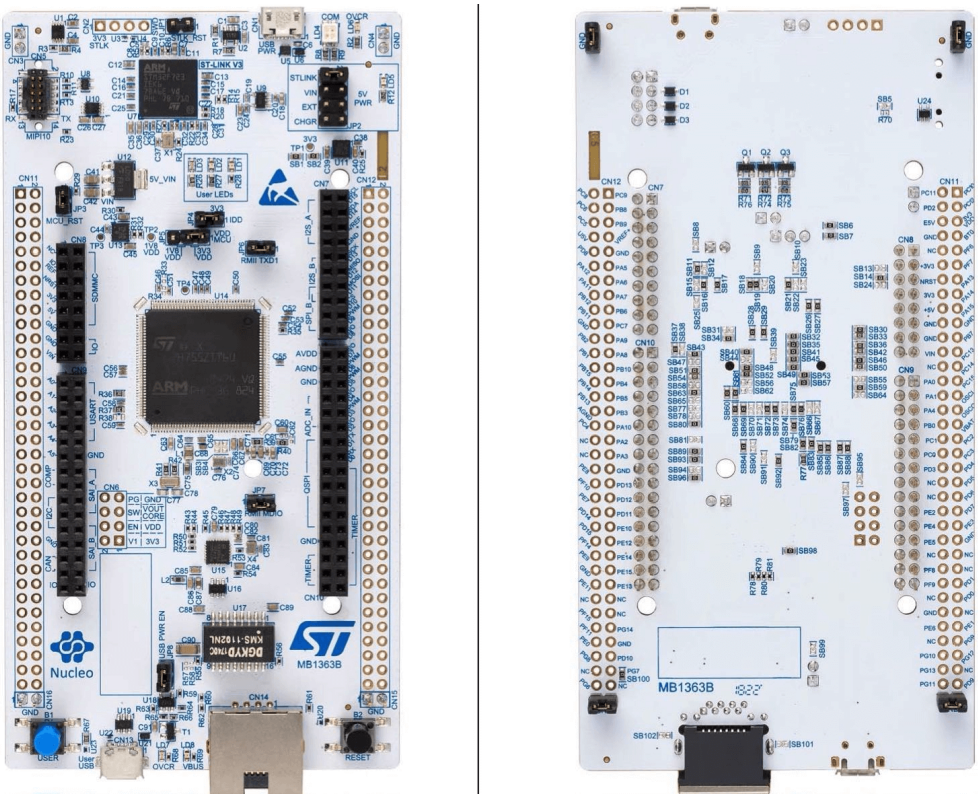


Figura 9.1: Evaluation board STM32H745ZI

STUDENTSUPSI

9.1 Demo con PysimCoder

Con PysimCoder si vuole ottenere una semplice demo basta sul processo Heater che verrà presentata il giorno della presentazione.
Per semplificare il sistema si tralascia nella modellizzazione del processo l'effetto termico dovuto all'irraggiamento. Con il metodo di Least-Square si ottiene un'identificazione di primo ordine del processo, in modo da costruire in seguito il controllore di stato. PysimCoder permette di salvare direttamente i dati su file con un apposito blocchetto, in questo modo sono stati ricavati i dati per eseguire l'identificazione del sistema.
In Figura 9.2 è rappresentato lo schema PysimCoder utilizzato per testare il controllore di stato sviluppato per il processo Heater.

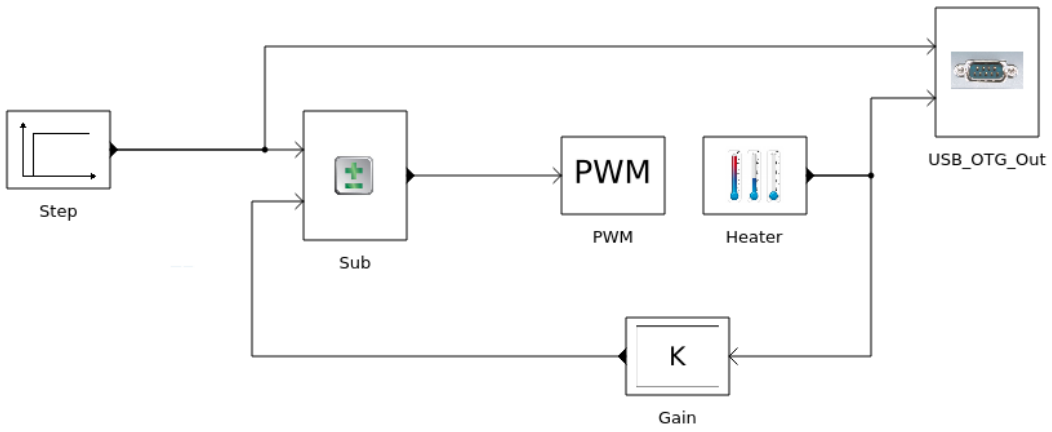


Figura 9.2: Schema PysimCoder controllore di stato

Per visualizzare a schermo e salvare i dati si necessita di un secondo schema PysimCoder, che legga i dati in arrivo sulla porta seriale del PC. Per questa demo si necessita di visualizzare il gradino di riferimento e la risposta del sistema, ovvero la temperatura rilevata dal sensore. In Figura 9.3 viene mostrato lo schema per la demo.

STUDENTSUPSI

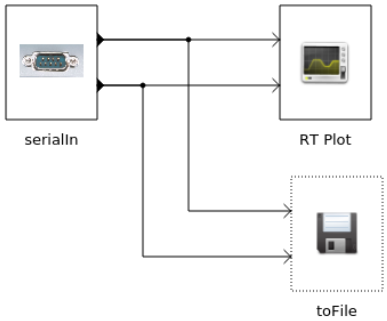


Figura 9.3: Schema PysimCoder visualizzazione dati

La realizzazione del controllore avviene tramite il linguaggio di programmazione python, che fornisce sia funzioni di identificazione, che funzioni di calcolo per controllori.

STUDENTSUPSI

STUDENTSUPSI

Capitolo 10

Migliorie

10.1 Modifiche dell'hardware

La prima versione della scheda di controllo progettata in SUPSI presenta alcuni accorgimenti che si potranno sistemare nella versione successiva.

10.1.1 Resistenze R2 e R4

La scheda stata concepita basandosi sulle caratteristiche tecniche di Raspberry, in cui le tensioni di pilotaggio dei pin risultano essere di 5V. Per questo motivo nel processo di heater della scheda si era scelta una determinata resistenza per garantire il corretto apporto di corrente al transistor. Il problema è sorto nella fase di test del processo utilizzando l'evaluation board scelta precedentemente (vedi appendice A), il sistema non riesce a oltrepassare la temperatura di 50$°C$.
Sapendo che la tensione di pilotaggio dei pin per la scheda STM32-F764ZG é di 3.3V e con carico si riduce fino a 3V possiamo calcolare corrente e potenza attraverso il transistor. Difatti la corrente massima nel transistor, senza l'apporto di modifiche alla scheda di controllo risulta.

$$MaxI = \frac{3V - 1V}{1000\Omega} \cdot 100 = 200[mA]$$

Per una potenza dissipata di:

$$MaxP = 12V * 200mA = 2.4[W]$$

Come rappresentato nella Tabella 10.1 si è dovuto sostituire le resistenze R2 e R4 per aumentare l'apporto di corrente ai transistor, così da mantenere la potenza dissipata sul transistor costante.

STUDENTSUPSI

Resistenza	Raspberry	STM32-F764ZG
R2	1 $[k\Omega]$	510 $[\Omega]$
R4	1 $[k\Omega]$	510 $[\Omega]$

Tabella 10.1: Valori resistenze processo heater

Diminuendo la resistenza otteniamo una corrente di:

$$MaxI = \frac{3V - 1V}{510\Omega} \cdot 100 = 392[mA]$$

Per una potenza dissipata di:

$$MaxP = 12V * 392mA = 4.71[W]$$

Le modifiche apportate alla scheda permettono ora di dissipare maggior calore sul transistor aumentandone la potenza, quindi di aumentare la temperatura massima del sistema controllato. Questa modifica permette di bilanciare quantità di calore emessa dal transistor, con la capacità di dissipazione del dissipatore accoppiato al transistor.

10.1.2 Sostituzione di componenti

Durante lo sviluppo del progetto sono emersi alcuni componenti non adeguati sulla scheda di controllo.

- **Fusibile power supply F1:** è troppo piccolo, di conseguenza appena si attacca l'alimentatore salta.
  Si deve optare per un fusibile con una corrente nominale di almeno 3 [A].

- **Fusibile power supply F2:** è troppo sensibile, di conseguenza appena si attacca l'alimentatore salta e introduce disturbi sull'alimentazione.

- **Convertitore AD:** non permette un campionamento continuo su tutti i canali, nello "scheduling" dei canali, ovvero quando si cambia canale di conversione in single shot, si deve attendere un tempo di setup ($\sim 250[\mu s]$) prima che il dato sia pronto.
  Si deve optare per un convertitore a 4 canali simultaneo. Questa problematica influisce sull'ottenimento di un controllore performante su processi rapidi come l'Inverter.

10.1.3 Possibili aggiunte

Durante tutte le fasi di test una problematica riscontrata sulla scheda di controllo è il fatto di non avere dei punti di misura predisposti. La loro presenza nella fase di sviluppo è necessaria per il fatto che alcuni dei punti più importanti sono difficilmente accessibili con

STUDENTSUPSI

la sonda di un oscilloscopio. Per questo motivo infatti durante una misura su di un sensore, involontariamente si è creato un cortocircuito sul transistor vicino bruciandolo, il quale è stato poi sostituito.
Si consiglia di aggiungere i seguenti punti di misura sulla prossima versione della scheda:

- Su tutti gli IC LM35, ovvero sensori di temperatura
- Su tutti i sensori di corrente (I_B, I_HB)
- Tutte le tensioni campionate (V_boost, V_B, VM_HB)

in modo da poter confrontare il valore analogico con il valore digitale letto via SPI.

Un'altra aggiunta potrebbe essere quella di inserire un selettore, il quale permetta di selezionare la resistenza in funzione dell'evaluation board utilizzata. Di conseguenza si deve prevedere di inserire anche la seconda resistenza da 510[Ω].

Si è inoltre notato che i componenti sensibili come i transistor e i sensori di temperatura presenti sul processo di Heater sono difficilmente sostituibili. A questo proposito una miglioria da considerare è la possibilità di utilizzare degli zoccoli in modo da ottimizzarne la sostituzione.

10.2 Osservazioni

Gli schemi della scheda di controllo presentano un piccolo errore, ovvero la denominazione della del convertitore analogico digitale in alto a destra nel Top Level non è ADC Heater ma bensì ADC Booster. Come si può notare dalla Figura 10.1.

STUDENTSUPSI

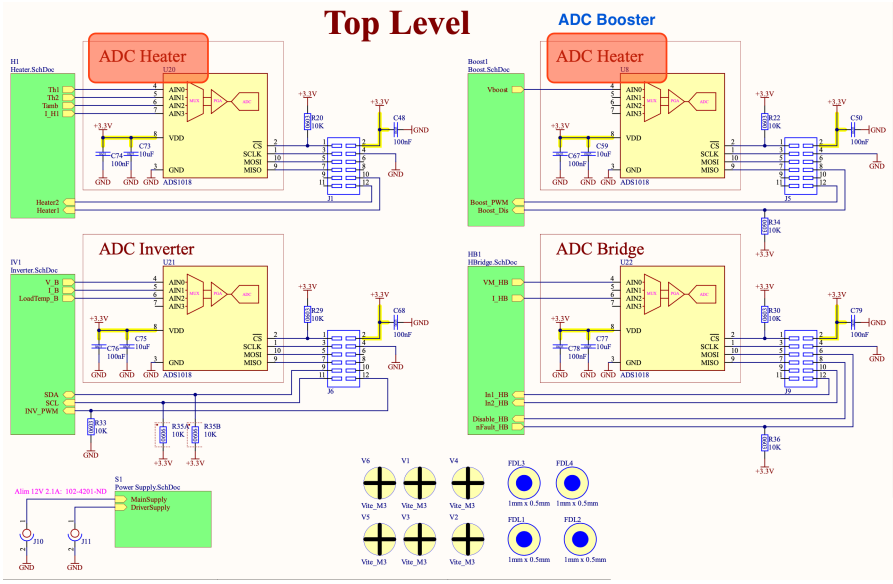


Figura 10.1: Errore nello schematico TopLevel

STUDENTSUPSI

Capitolo 11

Costi e spese

Il budget del progetto messo a disposizione è di 300 CHF. La Tabella 11.1 mostra il resoconto dettagliato delle spese effettuate durante il progetto. Le spese a cui si è fatto fronte sono dovute principalmente alle evaluation board, inoltre in corso d'opera a causa di alcuni imprevisti si è dovuto acquistare componenti da sostituire sulla scheda di controllo.
Altro materiale utilizzato per il progetto non riportato in tabella è stato preso dal laboratorio, di conseguenza i costi non sono riportati.

Componente	Descrizione	Fornitore	Costo U.	Quantità	Costo Tot
STM32F746ZG	Board MCU	Digi-key Electronics	23.74	2	47.48
STM32H745ZI	Board MCU	Digi-key Electronics	28.13	1	28.13
Cavo USB	USB-A to USB-B	Digi-key Electronics	4.37	1	4.37
Fili jumper	Femmina-Femmina	Digi-key Electronics	5.77	2	11.54
Connettore rapido	Electronics	Digi-key Electronics	1.85	12	22.20
Transistor	T0-220 NPN 60V 3A	Digi-key Electronics	0.591	10	5.91
Sensore temp	LM35 TO92 $0-100°C$	Digi-key Electronics	1.546	10	15.46
Resistenza SMD	510 OHM 0603	Digi-key Electronics	0.146	10	1.46
Connettore header	verticale 36 pos	Digi-key Electronics	1.449	10	14.49
Spedizione	-	Digi-key Electronics	20	1	20
				TOTALE +IVA	**184.21**

Tabella 11.1: Costi di progetto

STUDENTSUPSI

STUDENTSUPSI

Capitolo 12

Conclusioni

In questo lavoro si è cercato di costruire un sistema didattico per il laboratorio di controllo, per ovviare alle problematiche poste dalla pandemia in corso. Lo scopo, fornire allo studente un sistema compatto che possa essere usato dove, come e quando vuole.
Da non sottovalutare che un laboratorio portatile, consentirebbe di diminuire: gli spazi dedicati per svolgere i laboratori di controllo e i costi annui sostenuti dall'istituto per i locali adibiti ai laboratori; in quanto il costo per l'acquisto del materiale per il sistema didattico si limiterebbe ad un costo unico iniziale e non si protrarrebbe negli anni a venire.

Dalla ricerca è emerso che, la via intrapresa nello sviluppo di questo sistema didattico, non ha permesso di raggiungere tutti gli obbiettivi prefissati. Le problematiche riscontrate con i vari tool (MATLAB, Simulink, ST tool, STM32CubeIDE, ...) sono tali da non poter essere risolte. Il motivo principale per cui interfacciare tutti i tool risulta complesso, perché non sono sistemi open source, di conseguenza non si può accedere al codice.

Per sviluppare il sistema si è utilizzato il toolbox STM32MAT/TARGET di ST basato sulla tecnica del Model-Base Design, rivelandosi al di sotto delle aspettative, creando problemi di generazione e compilazione del codice. In corso d'opera si è cercato di risolvere i problemi generati dal tool. Per ovviare ai problemi sopra esposti si è contattato ST, richiedendo l'accesso ai codici sorgente, l'azienda ha dato risposta negativa. Perseverando nella risoluzione degli errori, le tempistiche pianificate non sono state rispettate, difatto si pensava di poter risolvere in meno tempo di quello impiegato. In accordo con il relatore per ottimizzare i tempi, alfine di ottenere almeno un processo funzionante, si è stabilito di concentrarsi unicamente sull'Heater.

Lo studio ha evidenziato che, i sistemi open source sono più manovrabili e di miglior utilizzo per lo scopo prefissato, in quanto la personalizzazione del codice è attuabile; a differenza di un sistema chiuso come sono i tool (MATLAB, Simulink, ST tool, ...).

STUDENTSUPSI

Nelle fasi di stallo del progetto si è avanzato con la realizzazione delle schede di esercitazione per i futuri allievi; dapprima modellando i sistemi ed in seguito procedendo allo sviluppo dei vari controllori. Dato il protrarsi delle problematiche riscontrate nell'interfacciarsi alla scheda di controllo, le schede di esercitazione sono state sviluppate solo fino alla fase di simulazione su PC.
Vi è sicuramente la possibilità di realizzare il sistema, come dimostra la soluzione con PysimCoder realizzata dal Prof. Bucher, di conseguenza la fattibilità del progetto é dimostrata.
Vi sono altre varianti che permetterebbero di realizzare il sistema: la scrittura di un toolbox per Simulink specifico, con la via mostrata attraverso PysimCoder o che ST in futuro fornisca gli accessi ai codici sorgenti.

STUDENTSUPSI

Bibliografia

12.1 Sitografia

12.1.1 Heater

https://apmonitor.com/pdc/index.php/Main/ArduinoTemperatureControl (last visit 24.08.2020)

12.1.2 STM32MAT/TARGET

https://ch.mathworks.com/de/hardware-support/st-nucleo.html?s_tid=AO_HS_info (last visit 24.08.2020)

https://community.st.com/s/question/0D50X00009XkY5v/stm32mat-Simulink-spi-example-compilation (last visit 24.08.2020)

https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-utilities/stm32-mat-target.html (last visit 24.08.2020)

https://ch.mathworks.com/de/videos/deploy-your-Simulink-algorithms-to-stm32-96485.html (last visit 24.08.2020)

https://ch.mathworks.com/help/supportpkg/stmicroelectronicsstm32f4discovery/ug/external-mode-using-stm32f4-discovery-board.html (last visit 25.08.2020)

12.2 Bibliografia

Ing. Roberto Bucher & Dr. Silvano Balemi - Dinamica e stabilità, edizione 18 settembre 2018
Ing. Ivan Furlan & Dr. Silvano Balemi - Progettazione di controllori, edizione 29 settembre

STUDENTSUPSI

2018 Dr. Mikael Bianchi - System identification: methods and algorithms, edizione 7 maggio 2020

STUDENTSUPSI

Capitolo 13

Allegati

In allegato:

- Pianificazione del Progetto (Gannt)
- Codici sorgente
- Schemi elettrici

STUDENTSUPSI

STUDENTSUPSI

Appendice A

Programma di test

Dato il problema riscontrato nella generazione di codice per quanto riguarda la comunicazione SPI, si è optato di scrivere un codice di test tramite CubeIDE di ST in modo da testare sia le funzionalità della scheda di controllo che la comunicazione SPI. Il problema riscontrato nella generazione automatica di codice non permetteva di assegnare al blocco SPI di Simulink un tipo di dato adeguato. Con questo test si vuole capire come strutturare un nuovo blocchetto Simulink per l'applicazione.

A.1 Descrizione funzionamento

Il programma di test svolge 3 operazioni, la prima è la configurazione del registro interno al convertitore analogico/digitale (ADS1018), la seconda funzione è quella di commutare lo stato dell'elemento riscaldante tramite un segnale di controllo, ed infine tramite 3 LED comunica lo stato del sistema all'utilizzatore. I LED forniscono le seguenti informazioni:

- LED rosso $\rightarrow$ elemento riscaldante **ON**
- LED verde $\rightarrow$ elemento riscaldante **OFF**
- LED blu $\rightarrow$ errore tra riferimento ed uscita inferiore al $\pm$ 1%

L'obbiettivo è quello di poter scegliere una temperatura e di mantenerla costante nel tempo grazie al controllore feed-forward.

STUDENTSUPSI

A.2 Simulazione

Scegliendo una temperatura da raggiungere il sistema andrà a stabilizzarsi attorno al valore desiderato. Tramite il tracciamento attivo delle variabili che offre CubeIDE è possibile tracciare direttamente il grafico di risposta al gradino.

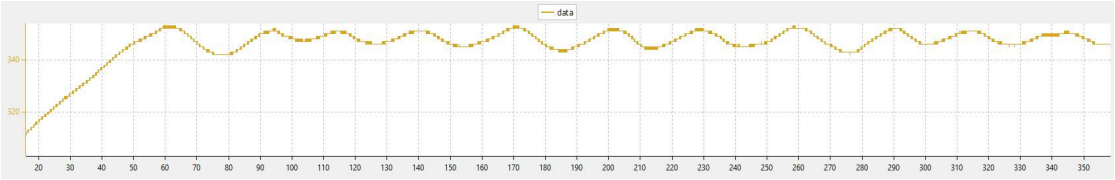


Figura A.1: Risposta al gradino del sistema riscaldante

Come rappresentato nella Figura A.1 il sistema tende sempre ad oscillare attorno al valore imposto al sistema (35°C). L'oscillazione risulta irregolare perché i fattori ambientali esterni non sono costantemente stabili.

A.2.1 Documentazione

Il progetto è in allegato alla documentazione, lo si può trovare nel seguente percorso: **Cartella: Software/SPI_Test**. Per ottenere il tracciamento della variabile bisogna attivare il Serial Wire Viewer nelle impostazioni di debug e attivare la finestra (SWV Data Trace Timeline Graph), nelle impostazioni attivare il tracciamento della variabile "data" e l'ITM Stimulus Port selezionare "0". Prima di far partire il debug premere il pulsante di record (pallino rosso).

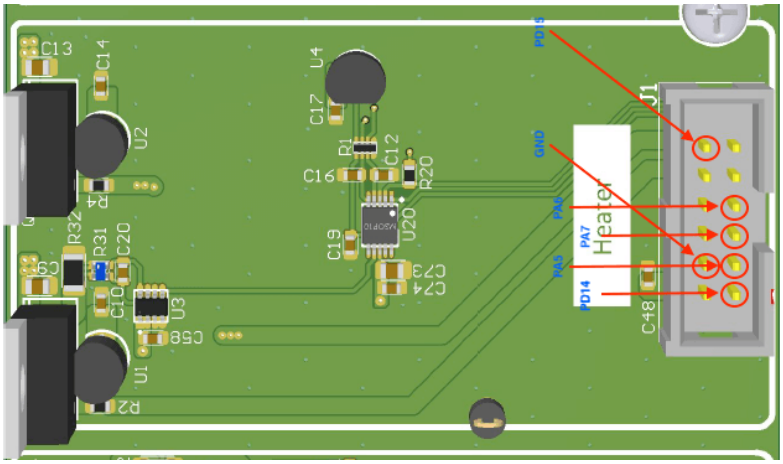


Figura A.2: Collegamenti scheda controllo - evaluation board

STUDENTSUPSI

Nella Figura A.2 sono descritti i collegamenti da effettuare per eseguire il test senza apportare modifiche al file (.IOC), il quale stabilisce la configurazione dei piedini.

STUDENTSUPSI

STUDENTSUPSI

Appendice B

Manuale d'uso STM32MAT/TARGET

B.1 Introduzione

In questo documento l'obbiettivo è quello di descrivere tutte le fasi a cui portano poi all'auto generazione di codice da Simulink per poi scaricarlo sul nostro microcontrollore. Andremmo a vedere quali programmi necessitiamo e tutti i passaggi necessari per raggiungere l'obbiettivo.

B.2 Programmi

Per utilizzare questo sistema vi sono 4 programmi da installare:

- STM32-MAT/TARGET
- STM32CubeMX
- IAR Embedded Workbench for Arm
- MATLAB/Simulink

B.2.1 STM32-MAT/TARGET

STM32-MAT/TARGET è un pacchetto fornito da ST il quale contiene dei blocchetti Simulink che hanno l'obbiettivo di generare codice per alcuni tipi di processori STM32, di seguito i processori supportati:

- Nucleo-F401RE
- Nucleo-F103RB
- Nucleo-F302R8
- Nucleo-F031K6
- Nucleo-L476RG
- Nucleo-L053R8
- Nucleo-F746ZG
- Nucleo-F411RE
- Nucleo-F767ZI

STUDENTSUPSI

Purtroppo ST Microelettronics fornisce questo pacchetto **solamente per la piattaforma Windows.** Di seguito il link per scaricarlo[1]:
`https://www.st.com/en/development-tools/stm32-mat-target.html`

B.2.2 STM32CubeMX

STM32CubeMX è un tool fornito da ST microelettronics che permette di generare il file **.ioc**, un file che racchiude le abilitazioni delle periferiche desiderate accoppiate ai pin di uscita. Anche questo programma purtroppo non è multipiattaforma ma **esiste solamente per Windows.** Per generare questo tipo di file su altre macchine si può usare il tool di ST Microelettronics: STM32CubeIDE. Di seguito il link per scaricare STM32CubeMX:
`https://www.st.com/en/development-tools/stm32cubemx.html`

B.2.3 IAR Embedded Workbench for Arm

IAR Embedded Workbench for Arm è un tool di debugging fornito dall'azienda IAR. È un tool a pagamento, ma fornisce una prova di 30 giorni. Per ora è l'unico tool con cui funziona il codice generato da MATLAB/Simulink. Di seguito il link per scaricarlo:
`https://www.iar.com/iar-embedded-workbench/#!?currentTab=free-trials`

B.2.4 MATLAB/Simulink

MATLAB/Simulink è il tool dove andremo a costruire ad elaborare i controllori per i vari processi disponibili sulla scheda. Tramite i blocchetti di Simulink e quelli forniti da ST Microelettronics, possiamo sviluppare vari tipi di controllori. MATLAB/Simulink è messo a disposizione dalla SUPSI per gli studenti, contattare il servizio informatico per l'installazione.

B.3 Configurazione MATLAB

Una volta installato il pacchetto STM32-MAT/TARGET, necessita una configurazione di MATLAB, in modo che nella libreria di Simulink vi siano i blocchetti desiderati. Di seguito i passaggi per configurare il tutto nel modo corretto.

B.3.1 Configurazione STM32-MAT/TARGET

Per prima cosa apriamo MATLAB e clicchiamo su **Set Path** nel menu a tendina home, come mostrato nella Figura B.1.

[1] Per scaricare da ST Microelettronics si necessita la registrazione al sito

STUDENTSUPSI

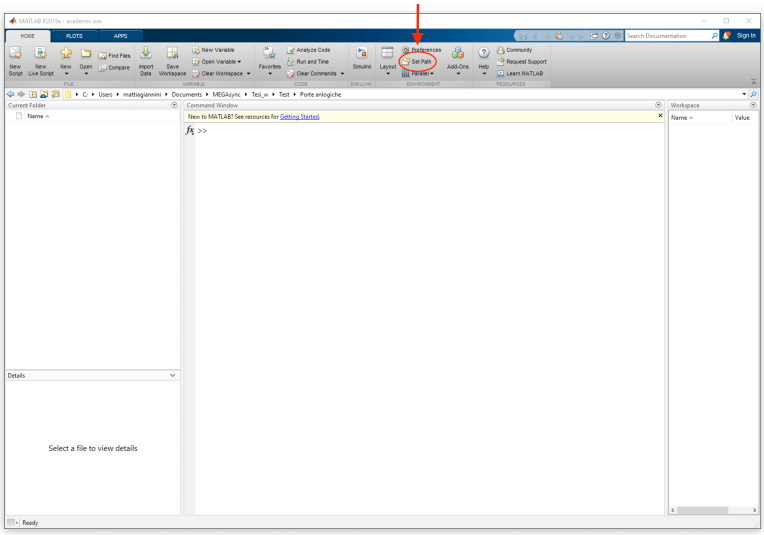


Figura B.1: MATLAB Set Path

Dopodiché apparirà un tendina come mostrato nella Figura B.2, si dovrà premere sul tasto **Add with Subfolders...**

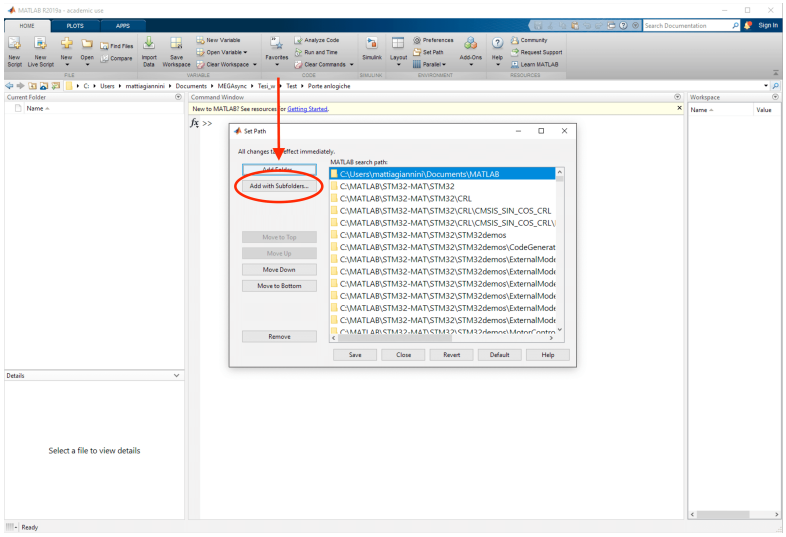


Figura B.2: MATLAB Add with Subfolders

Ora bisogna selezionare la cartella STM32, la quale è stata salvata nel percorso fornito durante l'installazione del pacchetto STM32-MAT/TARGET. Se installato tramite impostazioni di default il percorso della cartella è C:\MATLAB\STM32-MAT\STM32.
Una volta raggiunto il percorso premere **Select Folder** come in Figura B.3

STUDENTSUPSI

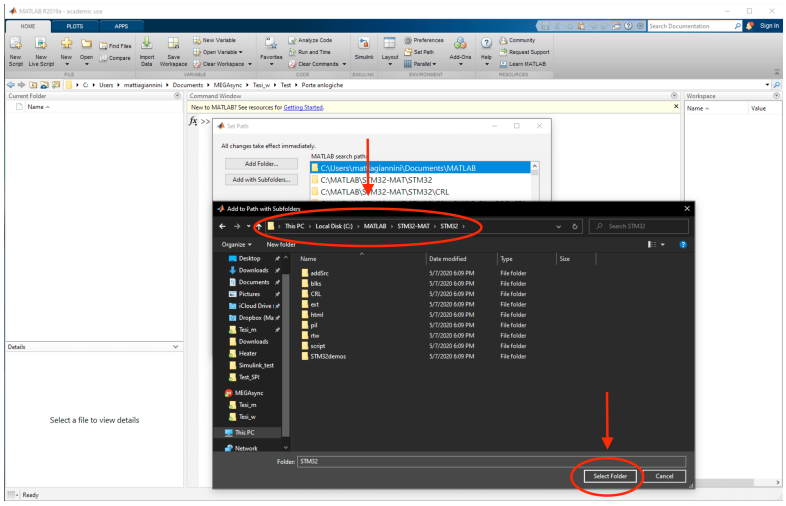


Figura B.3: MATLAB select Folder

Una volta terminato il processo, come mostrato nella FiguraB.4 vedrete i percorsi di tutte le cartelle importate dal pacchetto STM32-MAT/TARGET.

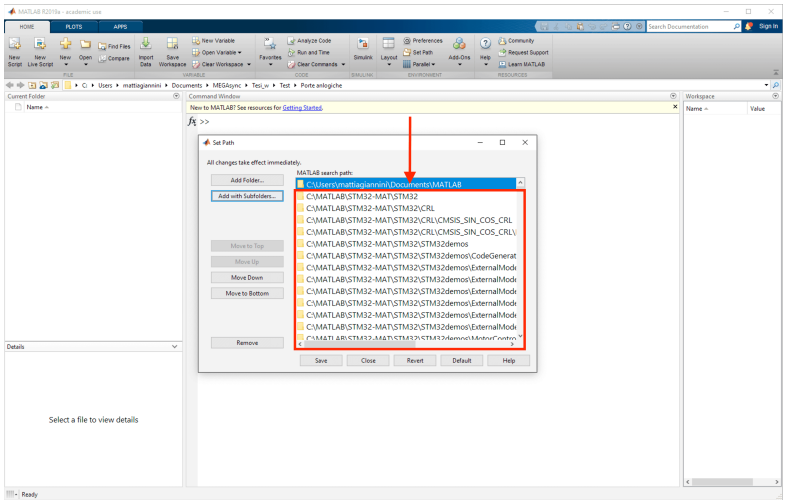


Figura B.4: MATLAB STM32 folders

B.3.2 Configurazione MATLAB Add-Ons

Prima di procedere ad utilizzare Simulink dobbiamo scaricare alcuni Add-Ons di MATLAB, alcuni potrebbero già essere installati, di seguito i 3 Add-On necessari:

- Arm_Compiler_Support_Package
- Embedded Coder
- MATLAB Coder

STUDENTSUPSI

Per aggiungere un Add-Ons bisogna selezionare dal menu a tendina HOME l'icona Add-Ons, come dalla Figura B.5. Premendo su Manage Add-Ons possiamo controllare se vi sono già gli add-on di cui necessitiamo, altrimenti tramite la barra di ricerca possiamo trovarli.

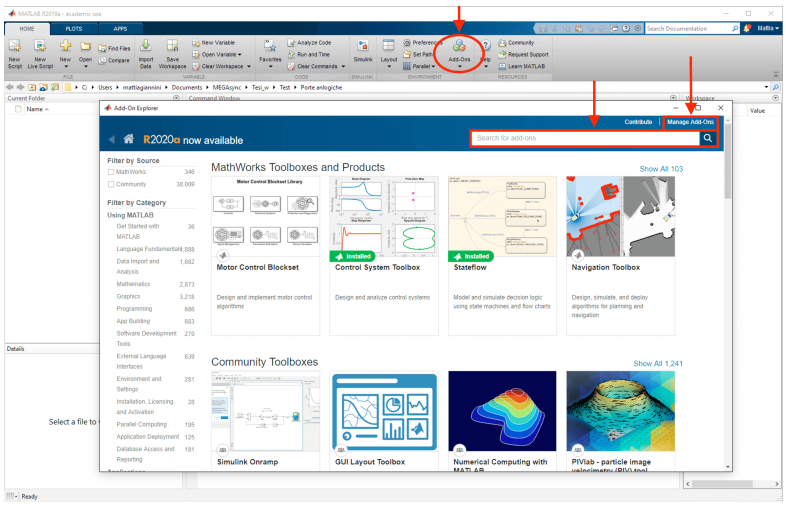


Figura B.5: MATLAB Add-Ons

Nella Figura B.6 si nota esattamente quali sono gli add-on corretti da installare.

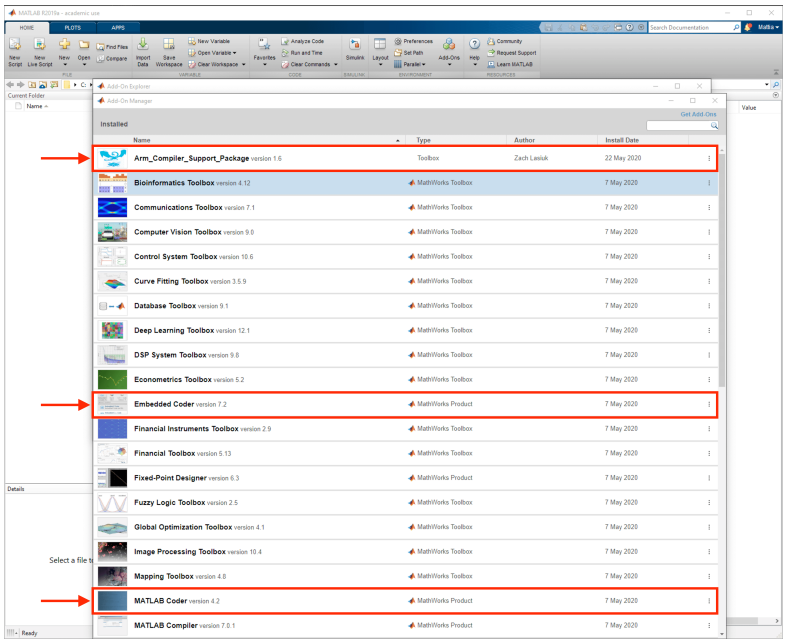


Figura B.6: MATLAB Manage Add-Ons

B.3.3 Verifica configurazione

Se il procedimento è stato eseguito correttamente, aprendo la libreria di Simulink, troveremo i blocchetti delle periferiche per le schede STM32-Nucleo, come da Figura B.7.

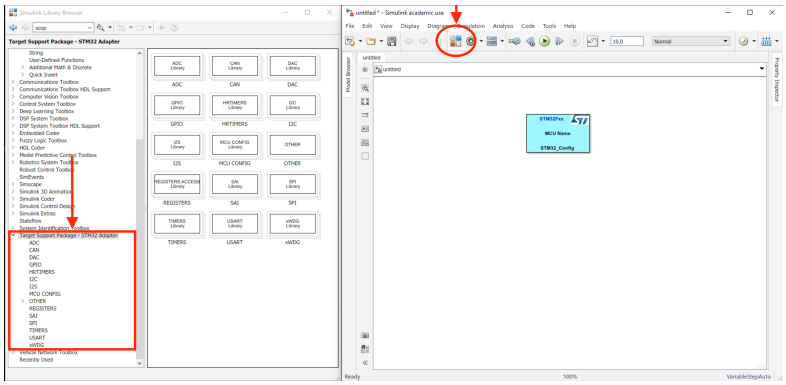


Figura B.7: Simulink library

B.4 Generazione file .ioc

B.4.1 Creazione progetto

Per generare il file .ioc ci occorre programma STM32CubeMX scaricato dal link precedente (vedi Capitolo 2.2). Per prima cosa aprire STM32CubeMX, selezionare **Access to MCU selector** come da Figura B.8

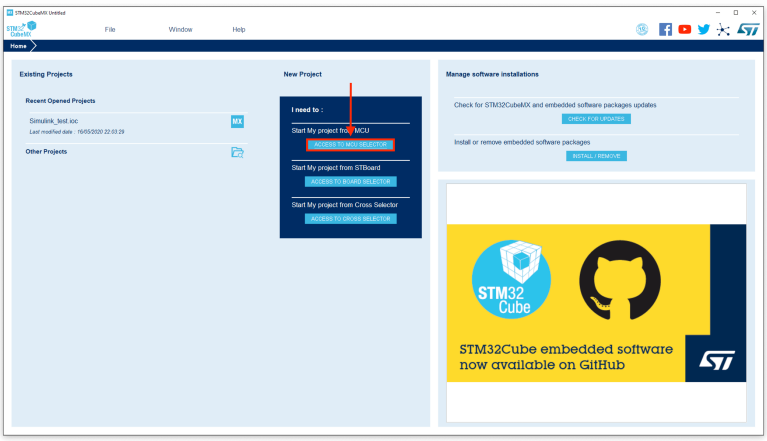


Figura B.8: STM32CubeMX start project

Si presenterà la seguente finestra dove andremo a selezionare il processore che della nostra evaluation board. Nel nostro caso il processore è il **STM32F746ZG**, tramite ricerca nel box dedicato. Una volta selezionato il processore premere su **Start Project** come mostra la Figura B.9

STUDENTSUPSI

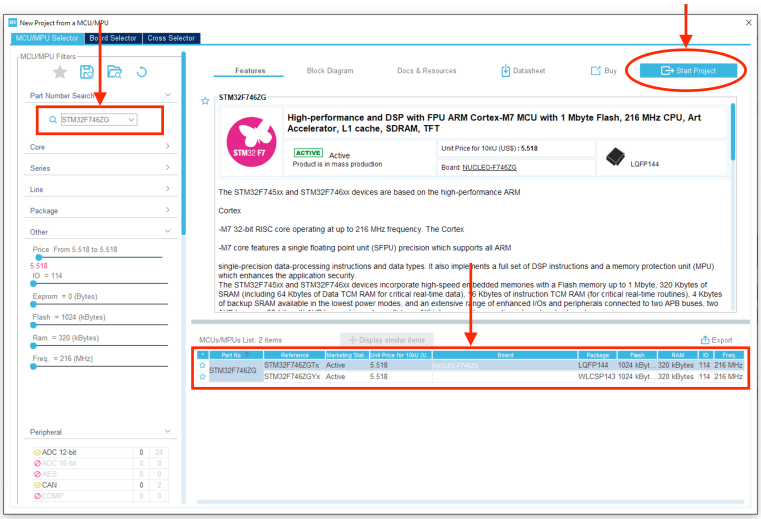


Figura B.9: STM32CubeMX MCU selection

B.4.2 Abilitazione delle periferiche

Il prossimo passaggio è quello di abilitare tutte le periferiche di cui necessitiamo, tramite l'apposita interfaccia grafica fornita dal tool. Per prendere un po' di confidenza faremo un piccolo esempio. Abilitiamo i seguenti GPIO come descritto nella Tabella B.1

Nome pin	Tipo	Funzione
PB0	GPIO_Output	LED green
PB7	GPIO_Output	LED blue
PB14	GPIO_Output	LED red
PC13	GPIO_Input	User Botton

Tabella B.1: Configurazione GPIO

Nella Figura B.10 vediamo esattamente dove si trovano i pin del MCU, nel caso in cui non si riescono ad identificare esattamente, tramite la barra di ricerca in basso a destra si può trovare il pin attraverso il suo nome. Una volta identificato il pin, facendo un click sinistro su di esso si potrà selezionare la funzione desiderata tramite il menu a tendina che appare come in Figura B.10. PPer proseguire nel nostro esercizio selezionare le funzioni come riportato nella Tabella B.1

STUDENTSUPSI

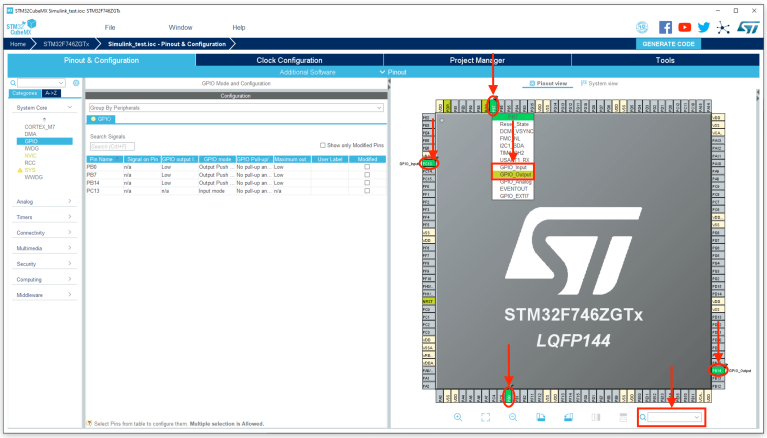


Figura B.10: STM32CubeMX peripherals selection

B.4.3 Parametri project manager

Dopo aver configurato tutte le periferiche nel modo corretto proseguiamo impostando alcuni parametri nel Project Manager, al quale si accede premendo sul pulsante in alto a destra indicato nella Figura B.11. Dopodiché assegniamo un nome al progetto, un percorso di salvataggio ed infine il parametro più importante dal menù a tendina come da figura e selezioniamo[2] in questo caso **EWARM**.

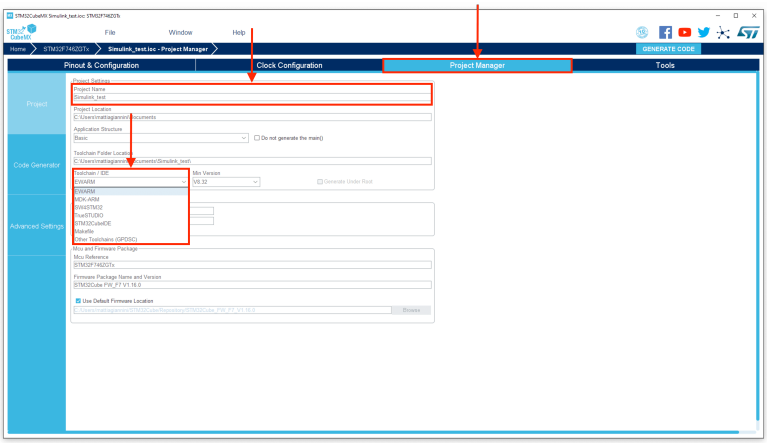


Figura B.11: STM32CubeMX project manager

Come ultima cosa si deve salvare il file .ioc come mostrato dalla Figura B.12, tramite il menù a tendina che appare premendo su **File** → **Save Project**.

[2]Sono stati testati alcuni altri IDE disponibili, ma forniscono alcuni errori nella fase di debugger. Consiglio vivamente EWARM al momento

STUDENTSUPSI

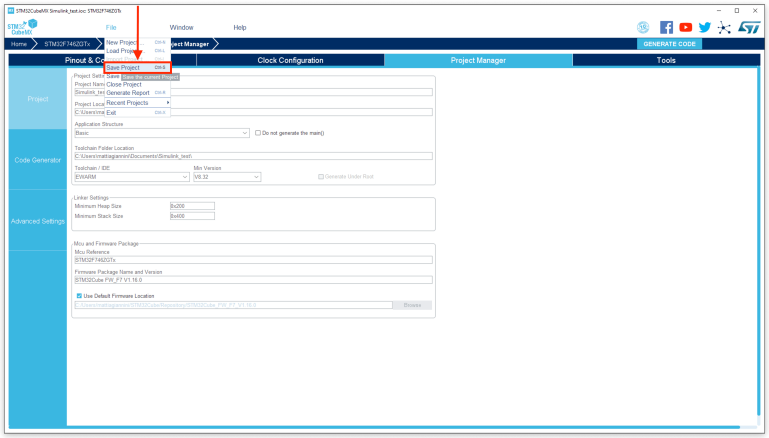


Figura B.12: STM32CubeMX save project

B.5 Generazione di codice tramite Simulink

Ora tramite Simulink possiamo costruire lo schema a blocchi che poi verrà tradotto in codice, rimangono alcune configurazioni che andremo a vedere prima di iniziare.

B.5.1 Configurazione MCU

Per prima cosa aprire un nuovo progetto Simulink, salvarlo nella cartella in cui abbiamo salvato precedentemente il file .ioc generato da STM32CubeMX. Per accendere alle impostazioni premere il tasto evidenziato nella Figura B.13.

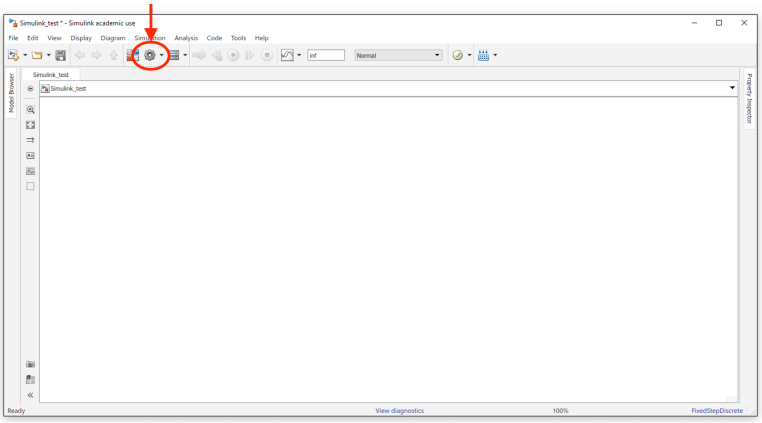


Figura B.13: Simulink project setting

Apparirà la finestra come da Figura B.14, modificare i parametri come mostrato.

STUDENTSUPSI

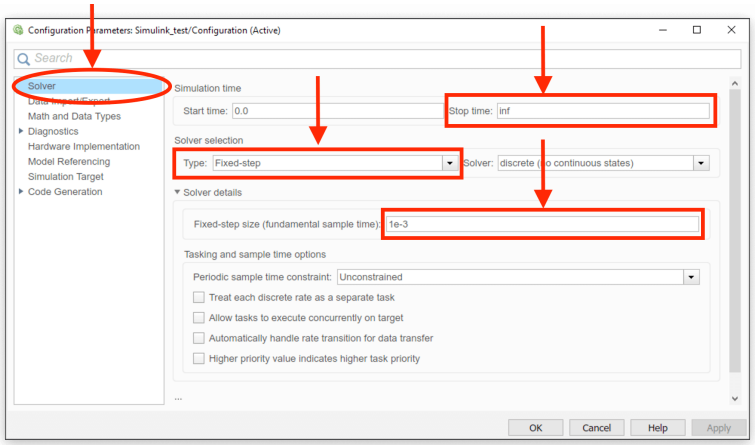


Figura B.14: Simulink project solver setting

Tramite il menù a tendina laterale selezionare **Code Generation** come mostrato nella Figura B.15, vistare la casella come illustrato, ed infine premere sul tasto **Browse**.

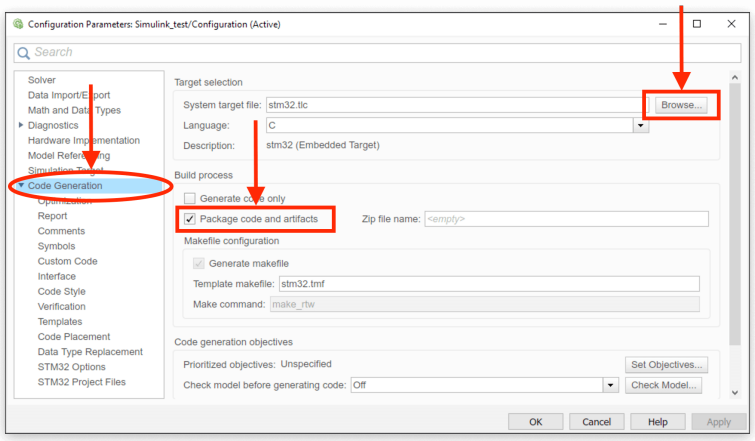


Figura B.15: Simulink code generation setting

Si aprirà una tendina come da Figura B.16, selezioniamo la voce **stm32.tlc** come illustrato e premiamo **OK**.

STUDENTSUPSI

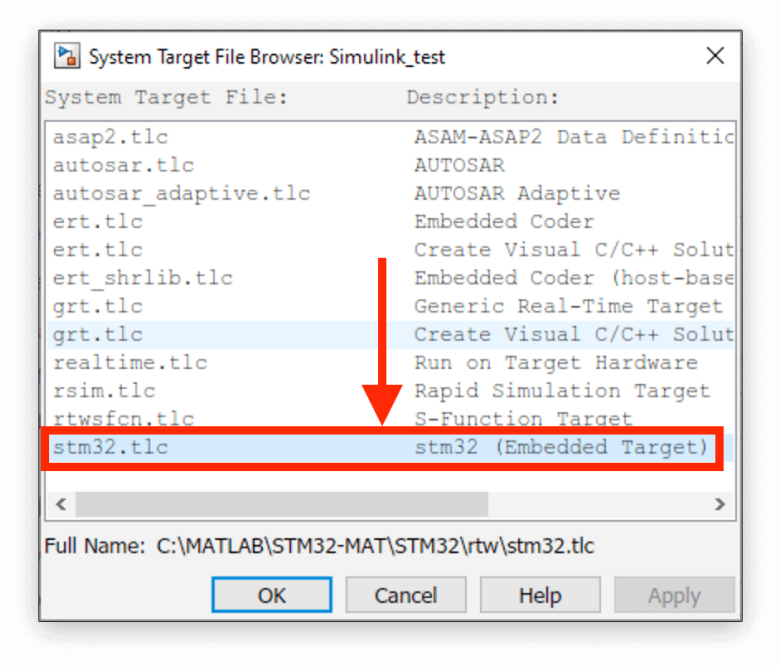


Figura B.16: Simulink code generation browse setting

Infine tornando nella sezione Code Generation ci spostiamo nella sottosezione **STM32 Options** come da Figura B.17, appena metteremo il visto alla voce **STM32CubeMx Path Update** apparirà e dopo pochi secondi sparirà la finestra di avvio di STM32CubeMx. Se l'operazione è stata eseguita in modo corretto vedremo come illustrato il percorso d'installazione del tool STM32CubeMx.

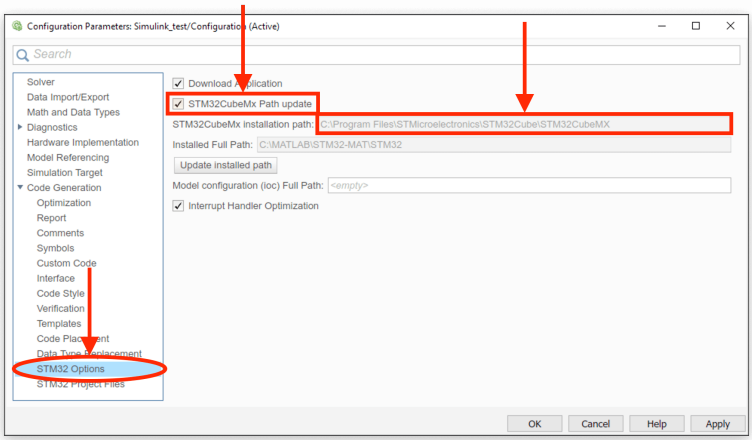


Figura B.17: Simulink code generation STM32 options

Per confermare il tutto premere su **Apply** poi su **OK** come da Figura B.18.

STUDENTSUPSI

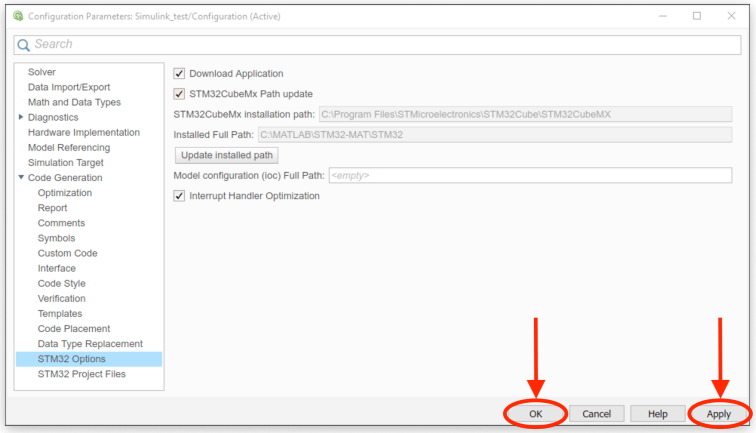


Figura B.18: Simulink setting save

B.5.2 Sviluppo codice esempio tramite Simulink

Con questo esempio vogliamo mostrare alcuni dei passaggi necessari per lo sviluppo di un qualsiasi progetto. In primo luogo dobbiamo aprire un nuovo progetto Simulink dove si deve importare il blocchetto di configurazione del MCU tramite la libreria, come nella Figura B.19.

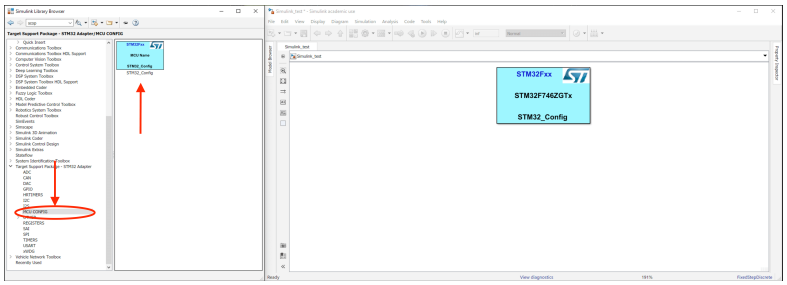


Figura B.19: Project MCU setting

Questo blocchetto configura le periferiche abilitate durante la generazione del file.ioc. Ora si deve assegnare il file al blocchetto. Dunque doppio click sul blocchetto e apparirà la finestra come nella Figura B.20. Come da figura premere sul bottone **Select STM32 configuration file**. Dopodiché selezionare il file generato con STM32CubeMx in precedenza.

STUDENTSUPSI

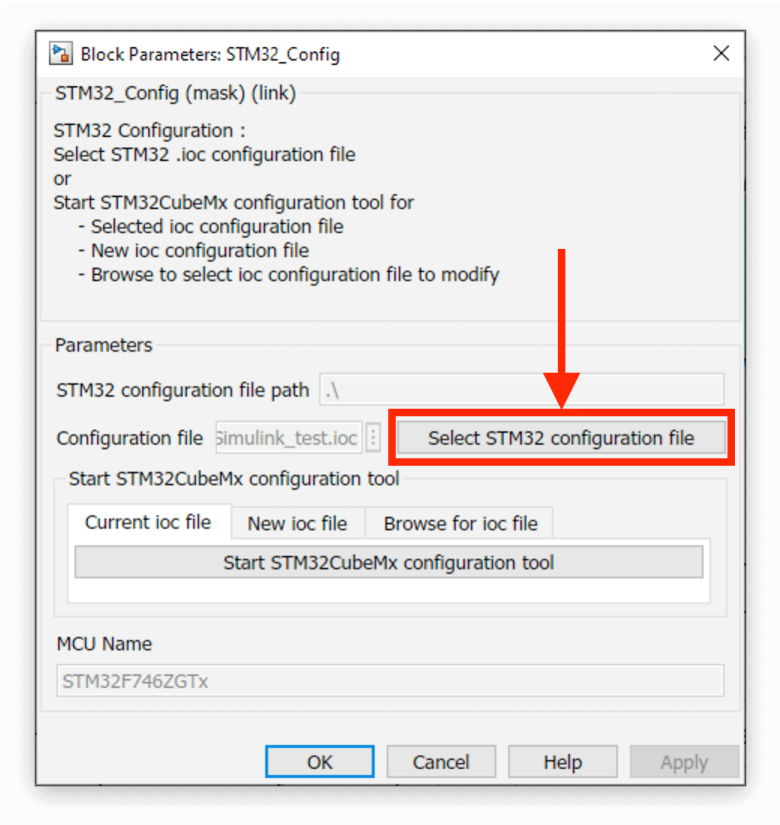


Figura B.20: Project MCU setting ioc file seletion

Dopo averlo selezionato premere su **Open** e poi su **OK**.
Torniamo ora al progetto Simulink ed importiamo 2 blocchetti, il GPIO_Write e il GPIO_Read presenti entrambi nella libreria di Simulink. Entrando nelle impostazioni dei blocchetti si può scegliere quali pin attivare tra quelli abilitati tramite la generazione del file.ioc. I pin che non sono stati abilitati tramite il file.ioc non saranno attivabili, dunque per abilitare altri pin si deve aprire il file.ioc e tramite STM32CubeMx abilitare i pin desiderati salvare il file. A quel punto i blocchetti di Simulink potranno essere abilitati sui nuovi pin configurati. Dalla Figura B.21 si vede come i pin abilitati tramite il file.ioc sono selezionabili mentre gli altri no.

STUDENTSUPSI

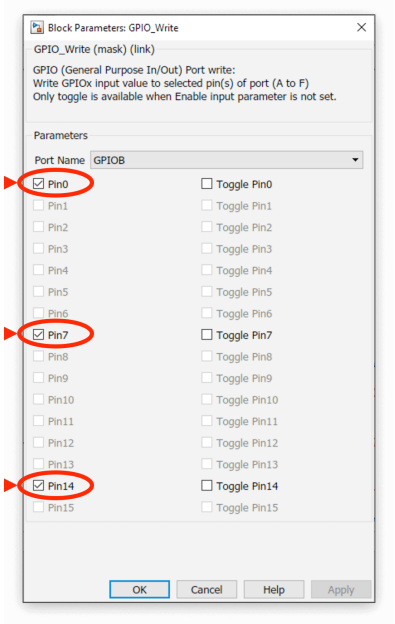


Figura B.21: Project GPIO setting

Ora necessitiamo di un chart per costruire la state machine e un detect raise positive il quale permette di detettare solamente il fianco di salita mentre si preme il bottone. Nella Figura B.22 vediamo come si presenta lo schema a blocchi dopo avere abilitato tutti i pin nei relativi blocchetti.

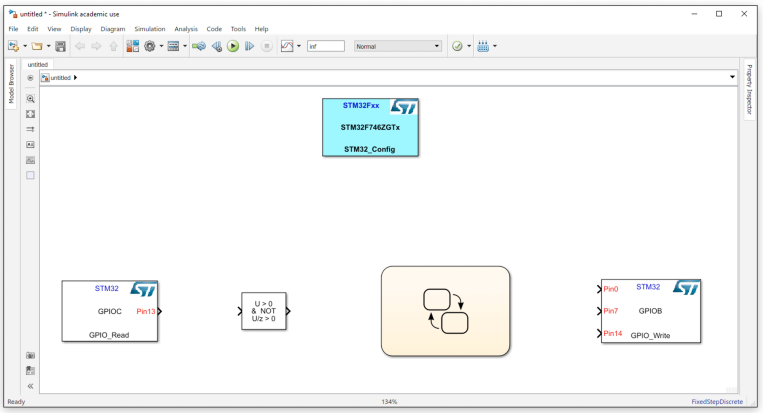


Figura B.22: Project block diagram step 1

Entrare nel blocchetto chart e costruire la state machine come da Figura B.23, per aggiungere uno stato premere il tasto come evidenziato nella figura.

STUDENTSUPSI

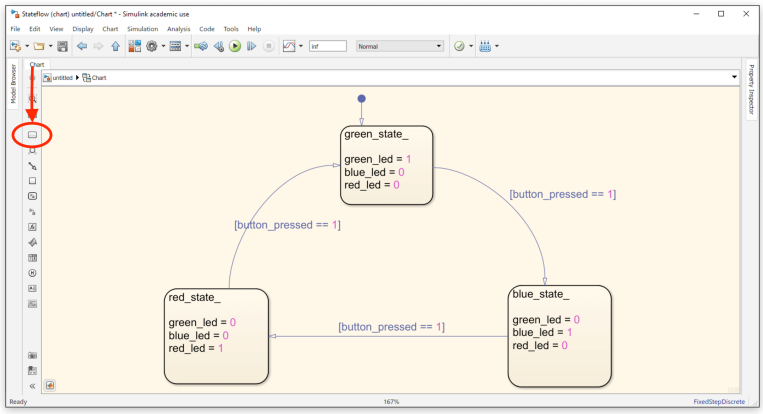


Figura B.23: Project state machine

Tornare ora allo schema principale, a questo punto siamo pronti per generare il codice, premere il pulsante come indicato nella Figura B.24. Appariranno degli errori, perché come di nota dallo schema a blocchi le variabili create nella state machine non sono assegnate.

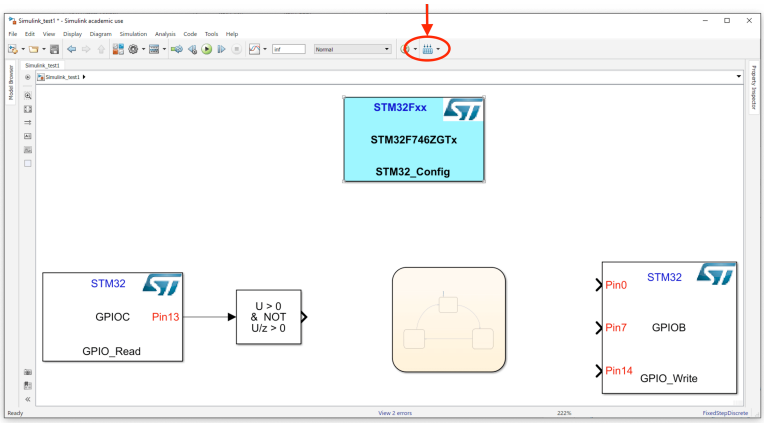


Figura B.24: Project block diagram step 2

Nella finestra d'errore, premiamo sul link in blu come da Figura B.25.

STUDENTSUPSI

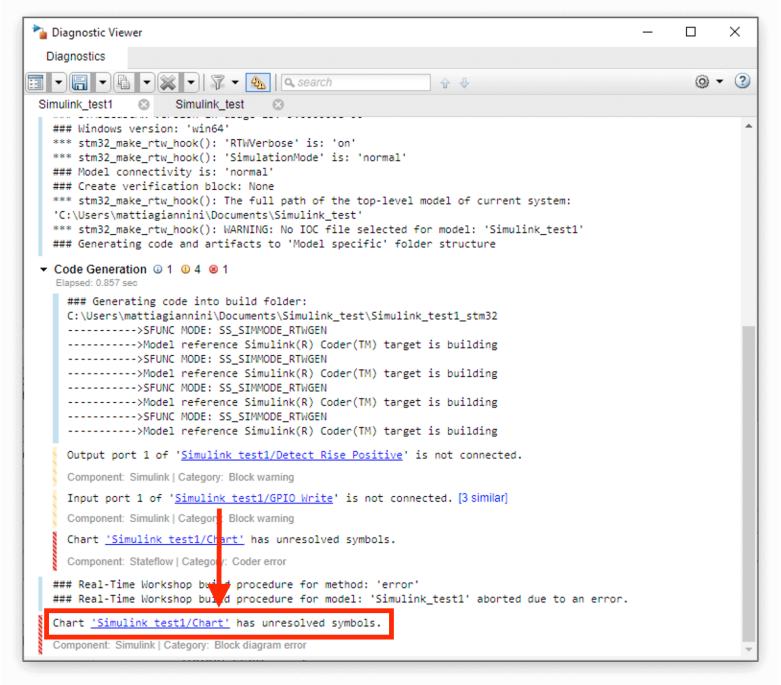


Figura B.25: Project code generation error

Dopo aver cliccato apparirà il tool Symbol Wizard come da Figura B.26 il quale va ad assegnare il tipo di variabile utilizzato nella state machine, basandosi su di essa per identificarne il tipo. Dunque premiamo su **OK**.

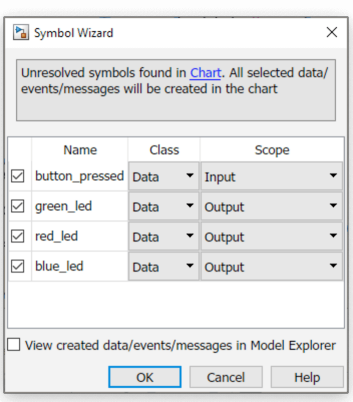


Figura B.26: Project simbol wyzard

A questo collegare i blocchi fra loro come da Figura B.27 e generare il codice tramite il pulsante come da figura. Se vi dovessero essere altri errori controllare di avere eseguito il procedimento correttamente.

STUDENTSUPSI

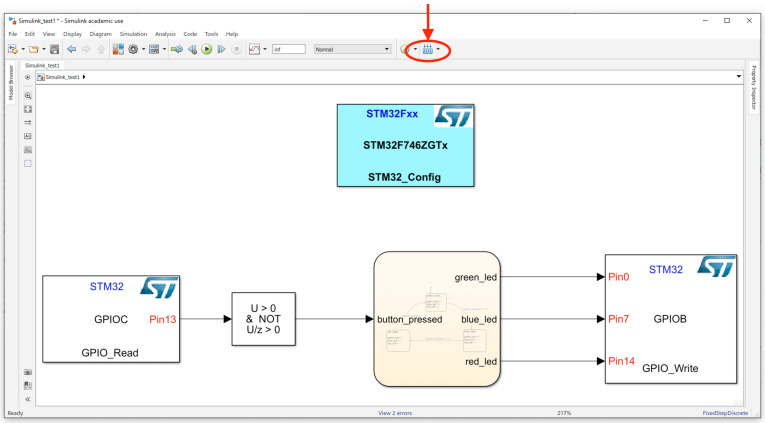


Figura B.27: Project block diagram step 3

B.5.3 Debuggin codice

Ora il codice è stato generato, per scaricarlo sulla board si deve aprire il progetto tramite il tool EWARM, per farlo andare nella cartella dove è stato salvato il file Simulink precedente e, troverete una cartella intestata EWARM, come da figura B.28 entrare nella cartella.

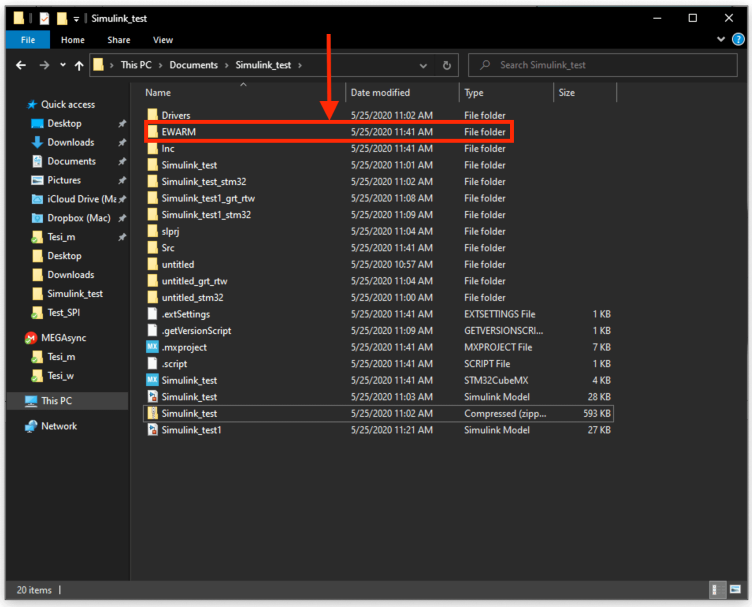


Figura B.28: Project debuggin step 1

Aprire il progetto tramite il file evidenziato in Figura B.29, attendere l'apertura del progetto.

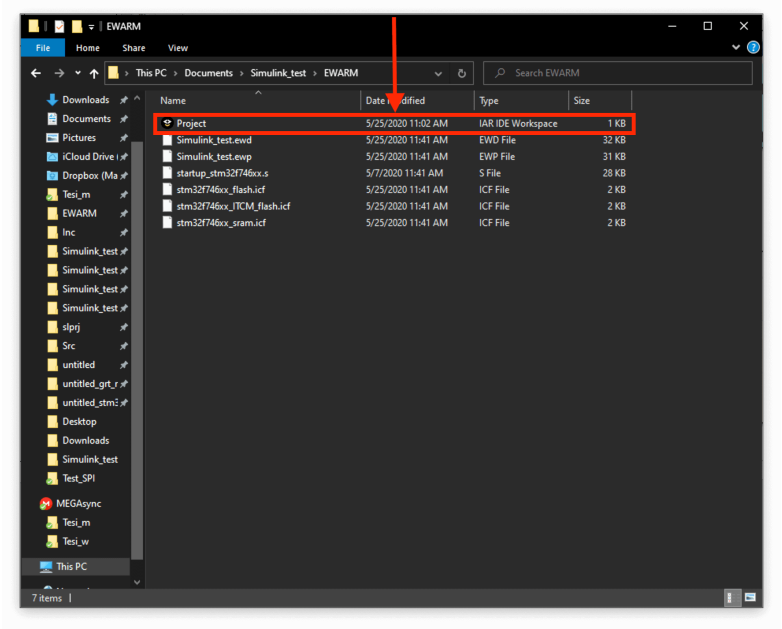


Figura B.29: Project debuggin step 2

A questo punto collegare la scheda al PC, attendere l'installazione dei driver, poi scaricare il codice tramite il pulsante illustrato nella Figura B.30

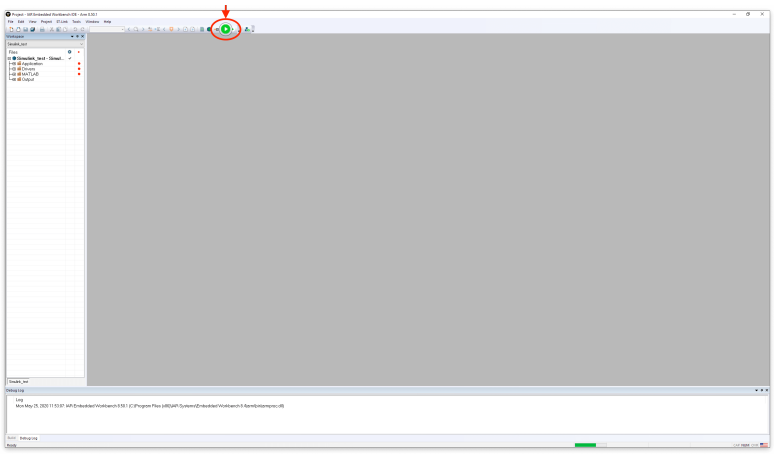


Figura B.30: Project debuggin step 3

Dopo aver scaricato il codice si aprirà il file che contiene la funzione main, il debugger attenderà il comando di esecuzione, quindi premere il tasto come da Figura B.31

STUDENTSUPSI

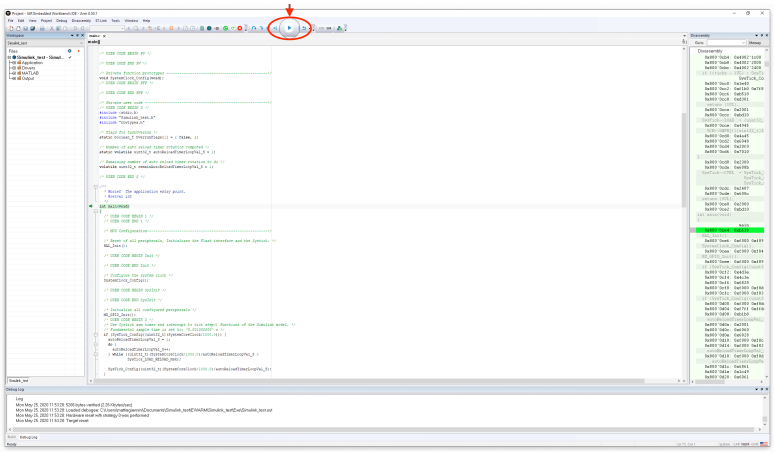


Figura B.31: Project debuggin step 4

A questo punto il codice è all'interno del MCU ed è possibile testarlo. Il codice sviluppato permette di accendere e spegnere una sequenza di LED ogni volta che si preme il pulsante. Nel seguente link troverete un breve video dove si mostra l'esecuzione del codice.
`https://mega.nz/file/PlkVGART#IuhxDfTgTAEiu34YNyDWQKOSSLwvD_E4sLY_qb2TjA8`

STUDENTSUPSI

STUDENTSUPSI

Appendice C

Scheda Heater

C.1 Introduzione

Sfruttando la teoria dei corsi di controllo possa sviluppare autonomamente vari tipi di controllori dal controllore di stato fino al MPC. Con l'ausilio di MATLAB e Simulink, si vuole simulare la soluzione migliore per il processo in questione. Date le problematiche riscontrate in corso d'opera del progetto, l'esercitazione si limiterà alla simulazione.

C.2 Obbiettivi

Partendo dal modello fisico, modellizzare il processo matematicamente. Sviluppare tramite MATLAB e Simulink i controllori desiderati. Prima di generare codice e scaricarlo sul microcontrollore fare una simulazione per verificare le aspettative. Infine documentare il tutto tramite un rapporto dettagliato.

C.3 Modellizzazione

Il sistema deve essere modellato attraverso le leggi della fisica. Il consigli in questo caso è di partire dal sistema SISO, ovvero considerare il sistema unicamente con un elemento riscaldante. Così da ottenere una modellizzazione semplificata.
Dopo aver ottenuto il modello SISO passare alla modellizzazione del sistema completo risulta più semplice. La Figura C.1 rappresenta lo scambio di calori tra ambiente e transistor[1].

[1] Le nomenclature e i valori dei dati per la modellizzazione sono riassunti nella Tabella G.1.

STUDENTSUPSI

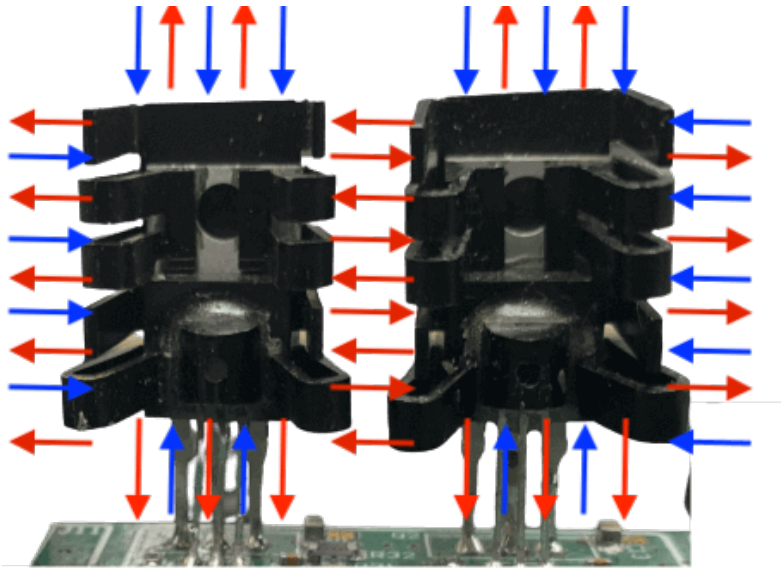


Figura C.1: Rappresentazione del processo Heater

Le leggi della fisica che governano questo sistema sono:

Irraggiamento:

$$Q = \epsilon \sigma A \cdot \Delta T^4 \tag{C.1}$$

Conduzione:

$$Q = UA \cdot \Delta T \tag{C.2}$$

Di seguito il procedimento consigliato:

- Identificare i parametri in gioco
- Modellizzare il sistema SISO
- Linearizzazione del sistema SISO
- Identificare le grandezze che variano tra sistema SISO e MIMO
- Modelizzare il sistema MIMO
- Linearizzazione del sistema MIMO

C.4 Controllori SISO

Come primo passo l'idea è quella di sviluppare una serie di controllori del sistema SISO. In modo che da prendere confidenza con il sistema.

STUDENTSUPSI

C.4.1 Controllore di stato

Il controllore di stato permette in alcuni casi di spostare i poli del sistema. Data la struttura del controllore riportata dalla Figura C.2 calcolare:

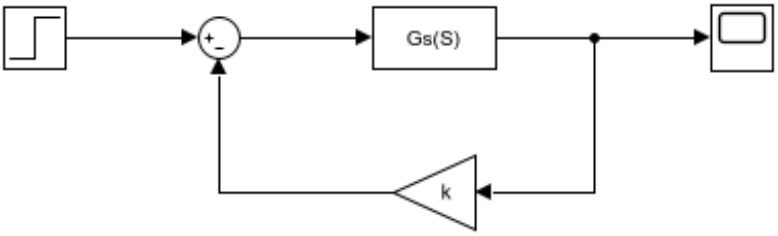


Figura C.2: Struttura controllore di stato

- Funzione di trasferimento open loop
- Scegliere i poli adeguati
- Applicare la formula per calcolare k
- Eseguire una simulazione "continua" con Simulink
- Eseguire una simulazione "discreta" con Simulink

C.4.2 Controllore PID

Il controllore PID permette di avere 3 gradi di libertà i quali permetto di andare a modificare i poli del sistema in modo da rendere stabile il sistema. Data la struttura del controllore riportata dalla Figura C.3 calcolare:

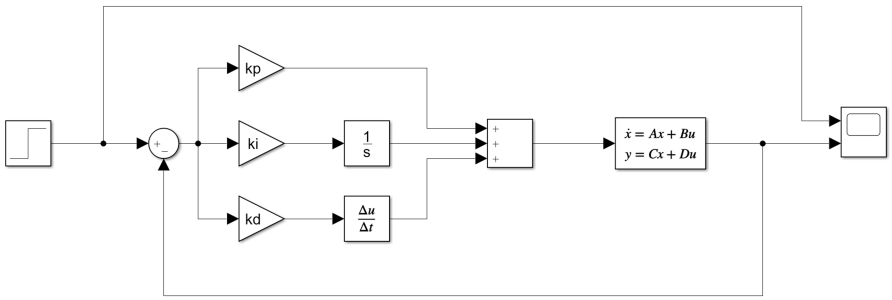


Figura C.3: Struttura controllore PID

Eseguire i seguenti passaggi:

- Trovare la funzione di trasferimento closed loop conoscendo la struttura del PID
- Posizionare i poli adeguatamente per rendere il sistema stabile

STUDENTSUPSI

- Eseguire una simulazione "continua" con Simulink
- Eseguire una simulazione "discreta" con Simulink

STUDENTSUPSI

Appendice D

Soluzioni Heater

D.1 Modellizzazione MIMO

Il sistema deve essere modellato seguendo le leggi della fisica che riguardano la trasmissione di calore.

Le leggi della fisica che governano questo sistema sono:

Irraggiamento:

$$Q = \epsilon\sigma A \cdot \Delta T^4 \tag{D.1}$$

Conduzione:

$$Q = UA \cdot \Delta T \tag{D.2}$$

Applicandole otteniamo il seguente sistema di equazioni che descrivono il sistema[1].

$$\begin{cases} \frac{\partial T_{H1}}{\partial t} = \frac{UA(T_a - T_{H1}) + \epsilon\sigma A(T_a^4 - T_{H1}^4) + U_s A_s(T_{H2} - T_{H1}) + \epsilon\sigma A_s(T_{H2}^4 - T_{H1}^4) + \alpha_1 Q_1}{m \cdot C_p} \\ \frac{\partial T_{H2}}{\partial t} = \frac{UA(T_a - T_{H2}) + \epsilon\sigma A(T_a^4 - T_{H2}^4) - U_s A_s(T_{H2} - T_{H1}) - \epsilon\sigma A_s(T_{H2}^4 - T_{H1}^4) + \alpha_2 Q_2}{m \cdot C_p} \\ \frac{\partial T_{C1}}{\partial t} = \frac{(T_{H1} - T_{C1}) \cdot kA_c}{m_s C_{p,s} \Delta x} \\ \frac{\partial T_{C2}}{\partial t} = \frac{(T_{H2} - T_{C2}) \cdot kA_c}{m_s C_{p,s} \Delta x} \end{cases} \tag{D.3}$$

Ora che abbiamo il modello lo si porta nello spazio degli stati linea-rizzandolo, si deve ottenere un sistema di questo tipo:

$$\begin{cases} \dot{x} = A \cdot x + B \cdot u \\ y = C \cdot x + A \cdot u \end{cases}$$

[1] Le nomenclature e i valori dei dati utilizzati sono riassunti nella Tabella G.1.

STUDENTSUPSI

In questo caso risulta:

$$\begin{cases} \begin{bmatrix} \dot{T}'_{H1} \\ \dot{T}'_{H2} \\ \dot{T}'_{C1} \\ \dot{T}'_{C2} \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \cdot \begin{bmatrix} T'_{H1} \\ T'_{H2} \\ T'_{C1} \\ T'_{C2} \end{bmatrix} + \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \\ b_{4,1} & b_{4,2} \end{bmatrix} \cdot \begin{bmatrix} Q'_1 \\ Q'_2 \end{bmatrix} \\ \begin{bmatrix} T'_{C1} \\ T'_{C2} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} \\ c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} \end{bmatrix} \cdot \begin{bmatrix} T'_{H1} \\ T'_{H2} \\ T'_{C1} \\ T'_{C2} \end{bmatrix} + \begin{bmatrix} d_{1,1} & d_{1,2} \\ d_{2,1} & d_{2,2} \end{bmatrix} \cdot \begin{bmatrix} Q'_1 \\ Q'_2 \end{bmatrix} \end{cases} \tag{D.4}$$

La linearizzazione come sappiamo non include le matrici C e D, le quali non cambiano rispetto a quello di partenza di conseguenza risultano:

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Per trovare le componenti delle matrici A e B del sistema linea-rizzato si deve derivare il sistema di Equazioni D.3 in modo da costruire le jacobiane. Quindi le componenti della

STUDENTSUPSI

matrice A sono:

$$a_{1,1} = \left.\frac{\partial f_1}{\partial T_{H1}}\right|_{\overline{T}_{H1}} = -\frac{UA + 4\epsilon\sigma A T_0^3 + U_s A_s + 4\epsilon\sigma A_s T_0^3}{mC_p}$$

$$a_{1,2} = \left.\frac{\partial f_1}{\partial T_{H2}}\right|_{\overline{T}_{H2}} = \frac{U_s A_s + 4\epsilon\sigma A_s T_0^3}{mC_p}$$

$$a_{1,3} = \left.\frac{\partial f_1}{\partial T_{C1}}\right|_{\overline{T}_{C1}} = 0$$

$$a_{1,4} = \left.\frac{\partial f_1}{\partial T_{C2}}\right|_{\overline{T}_{C2}} = 0$$

$$a_{2,1} = \left.\frac{\partial f_2}{\partial T_{H1}}\right|_{\overline{T}_{H1}} = \frac{U_s A_s + 4\epsilon\sigma A_s T_0^3}{mC_p}$$

$$a_{2,2} = \left.\frac{\partial f_2}{\partial T_{H2}}\right|_{\overline{T}_{H2}} = -\frac{UA + 4\epsilon\sigma A T_0^3 + U_s A_s + 4\epsilon\sigma A_s T_0^3}{mC_p}$$

$$a_{2,3} = \left.\frac{\partial f_2}{\partial T_{C1}}\right|_{\overline{T}_{C1}} = 0$$

$$a_{2,4} = \left.\frac{\partial f_2}{\partial T_{C2}}\right|_{\overline{T}_{C2}} = 0$$

$$a_{3,1} = \left.\frac{\partial f_3}{\partial T_{H1}}\right|_{\overline{T}_{H1}} = \frac{kA_c}{m_s C_{p,s}\Delta x}$$

$$a_{3,2} = \left.\frac{\partial f_3}{\partial T_{H2}}\right|_{\overline{T}_{H2}} = 0$$

$$a_{3,3} = \left.\frac{\partial f_2}{\partial T_{C1}}\right|_{\overline{T}_{C1}} = -\frac{kA_c}{m_s C_{p,s}\Delta x}$$

$$a_{3,4} = \left.\frac{\partial f_2}{\partial T_{C2}}\right|_{\overline{T}_{C2}} = 0$$

$$a_{4,1} = \left.\frac{\partial f_4}{\partial T_{H1}}\right|_{\overline{T}_{H1}} = 0$$

$$a_{4,2} = \left.\frac{\partial f_4}{\partial T_{H2}}\right|_{\overline{T}_{H2}} = \frac{kA_c}{m_s C_{p,s}\Delta x}$$

$$a_{4,3} = \left.\frac{\partial f_4}{\partial T_{C1}}\right|_{\overline{T}_{C1}} = 0$$

$$a_{4,4} = \left.\frac{\partial f_4}{\partial T_{C2}}\right|_{\overline{T}_{C2}} = -\frac{kA_c}{m_s C_{p,s}\Delta x}$$

STUDENTSUPSI

Per la matrice B invece sono:

$$
\begin{aligned}
b_{1,1} &= \left.\frac{\partial f_1}{\partial Q_1}\right|_{\overline{Q}_1} = \frac{\alpha_1}{mC_p} \\
b_{1,2} &= \left.\frac{\partial f_1}{\partial Q_2}\right|_{\overline{Q}_2} = 0 \\
b_{2,1} &= \left.\frac{\partial f_2}{\partial Q_1}\right|_{\overline{Q}_1} = 0 \\
b_{2,2} &= \left.\frac{\partial f_2}{\partial Q_2}\right|_{\overline{Q}_2} = \frac{\alpha_2}{mC_p} \\
b_{3,1} &= \left.\frac{\partial f_3}{\partial Q_1}\right|_{\overline{Q}_1} = 0 \\
b_{3,2} &= \left.\frac{\partial f_3}{\partial Q_2}\right|_{\overline{Q}_2} = 0 \\
b_{4,1} &= \left.\frac{\partial f_4}{\partial Q_1}\right|_{\overline{Q}_1} = 0 \\
b_{4,2} &= \left.\frac{\partial f_4}{\partial Q_2}\right|_{\overline{Q}_2} = 0
\end{aligned}
$$

Ora abbiamo il nostro sistema linearizzato, nello spazio degli stati.

D.1.1 Osservazioni

La differenza nelle superfici A ed A_s è dovuta alla struttura del modello fisico, infatti per il modello MIMO si deve tener conto della superficie laterale $[A_s]$ dell'heater 1 che "scambia calore" con l'heater 2. Mentre il resto della superficie $[A]$ laterale dei 2 heaters è ha contatto con l'ambiente circostante. Nella Figura D.1 il concetto risulta essere più chiaro.

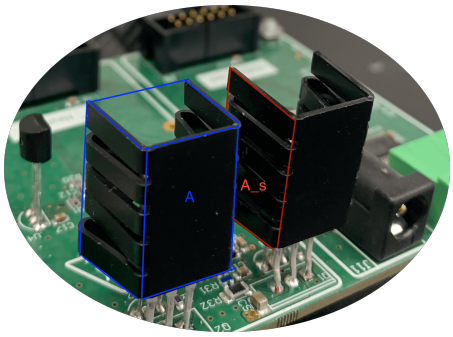


Figura D.1: Differenziazione $[A_s]$ e $[A]$

STUDENTSUPSI

D.2 Modellizzazione SISO

Sfruttando la modellizzazione realizzata per il sistema MIMO deriviamo il modello del sistema SISO. Otteniamo le seguenti relazioni.

$$\begin{cases} \frac{\partial T_{H1}}{\partial t} = \frac{UA(T_a - T_{H1}) + \epsilon\sigma A(T_a^4 - T_{H1}^4) + \alpha_1 Q_1}{m \cdot C_p} \\ \frac{\partial T_{C1}}{\partial t} = \frac{(T_{H1} - T_{C1}) \cdot kA_c}{m_s C_{p,s} \Delta x} \end{cases} \tag{D.5}$$

Sempre sfruttando la modellizzazione MIMO ricaviamo anche la linearizzazione del modello SISO. Per poi ottenere lo spazio degli stati.

$$\begin{cases} \dot{x} = A \cdot x + B \cdot u \\ y = C \cdot x + A \cdot u \end{cases}$$

Il sistema linearizzato in questo caso risulta:

$$\begin{cases} \begin{bmatrix} \dot{T}'_{H1} \\ \dot{T}'_{C1} \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \cdot \begin{bmatrix} T'_{H1} \\ T'_{C1} \end{bmatrix} + \begin{bmatrix} b_{1,1} \\ b_{2,1} \end{bmatrix} \cdot \begin{bmatrix} Q'_1 \end{bmatrix} \\ \begin{bmatrix} T'_{C1} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} \end{bmatrix} \cdot \begin{bmatrix} T'_{H1} \\ T'_{C1} \end{bmatrix} + \begin{bmatrix} d_{1,1} \end{bmatrix} \cdot \begin{bmatrix} Q'_1 \end{bmatrix} \end{cases} \tag{D.6}$$

Attenzione: le derivate ottenute nel sistema MIMO vanno adattate alla situazione del sistema SISO. Il processo non sarà riportato.

D.3 Controllore di stato SISO

Il controllore di stato ha un solo grado di libertà per questo può risultare anche impossibile realizzare un controllore funzionante.
Con l'ausilio della matrice di controllabilità si controlla se il processo è controllabile oppure no, in modo da non perdere tempo.

La matrice di controllabilità è definita nel seguente modo:

$$C = \begin{bmatrix} B & A \cdot B & A^2 \cdot B & \dots & A^{n-1} \cdot B \end{bmatrix} \tag{D.7}$$

L'informazione data dalla matrice di controllabilità è data dal determinante di essa:

$$det(C) = \begin{cases} 0, & \text{Non controllabile} \\ \neq 0, & \text{Controllabile} \end{cases}$$

STUDENTSUPSI

Il sistema risulta controllabile come dimostrano i calcoli eseguiti in MATLAB. Si procede con lo sviluppo del controllore, per identificare i il k necessario vi sono 2 vie:

- Scelta dei poli e utilizzo del comando "place" in MATLAB
- Letterario applicando la formula del controllore si stato, paragonando la funzione di trasferimento desiderata a quella del sistema closed-loop

Di seguito è presentato il secondo procedimento, gli script associati presentano entrambi i metodi La formula del controllore di stato:

$$G_{cl} = det(s * I - A - B * K) \tag{D.8}$$

I poli desiderati sono p1 e p2, di conseguenza la funzione di trasferimento desiderata è:

$$G_{des} = (s - p1) \cdot (s - p2) = s^2 + s(-p1 - p2) + p1 \cdot p2 \tag{D.9}$$

Per trovare il valore di K è necessario eguagliare l'Equazione D.8 e l'Equazione D.9, dopodiché risolvere il sistema.

$$\begin{gathered} G_{cl} = G_{des} \\ det(s * I - A - B * K) = s^2 + s(-p1 - p2) + p1 \cdot p2 \end{gathered} \tag{D.10}$$

D.3.1 Risultati

Le simulazioni effettuate in entrambi i modi hanno mostrato che il controllore non è efficace come ci si aspettava. A prova del fatto la Figura D.2 evidenzia un'ofset tra il valore desiderato e quello atteso.

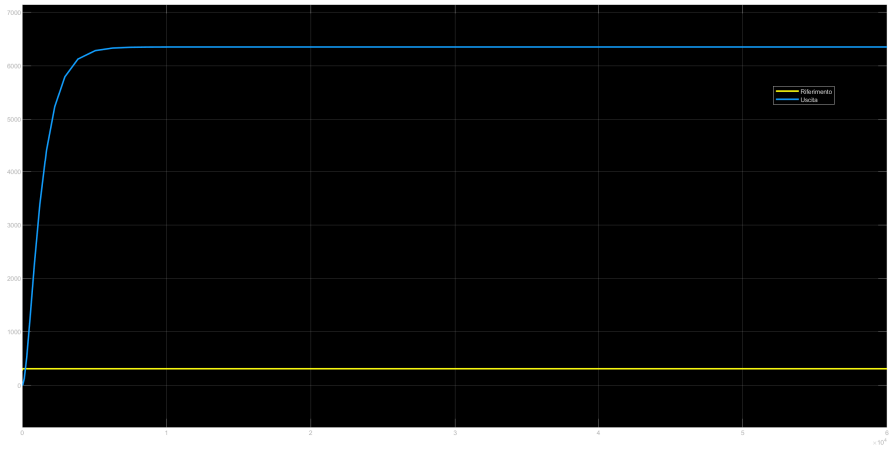


Figura D.2: Simulazione controllore di stato Heater SISO

STUDENTSUPSI

D.4 Controllore PID SISO

Dalla Modellizzazione effettuata precedentemente ricaviamo il sistema SISO linearizzato, valutandolo in un punto il quale può essere la temperatura ambiente per esempio. Data la struttura del controllore in Figura D.3,

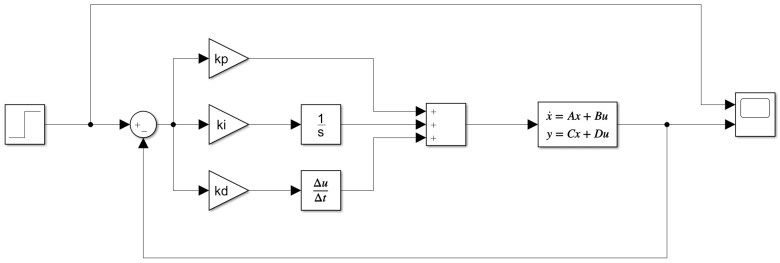


Figura D.3: Struttura controllore PID

la funzione di trasferimento del controllore PID è:

$$G_c(s) = \frac{K_d \cdot s^2 + K_p \cdot s + K_i}{s} \tag{D.11}$$

Ora si deve trovare la funzione di trasferimento closed loop con l'ausilio di Maison:

$$G_{cl}(s) = \frac{G_c(s) \cdot G(s)}{1 + G_c(s) \cdot G(s)} \tag{D.12}$$

Per piazzare identificare i guadagni del PID, si devono confrontare i denominatori della funzione G_{cl} e della G_{des} le quali deve essere dello stesso ordine. Dopodiché si scelgono i poli e si risolve il sistema di equazioni, in modo da trovare i vari [kd, kp, ki].
Lo sviluppo letterale per questo processo è molto complesso di conseguenza le soluzioni sono sviluppate su un script MATLAB.
Per velocizzare la costruzioni del controllore MATLAB mette a disposizione un blocchetto PID, il quale integra l'autotune. Questo tool permette di scegliere la risposta al gradino del sistema, la Figura D.4 mostra l'interfaccia del tool autotune.

STUDENTSUPSI

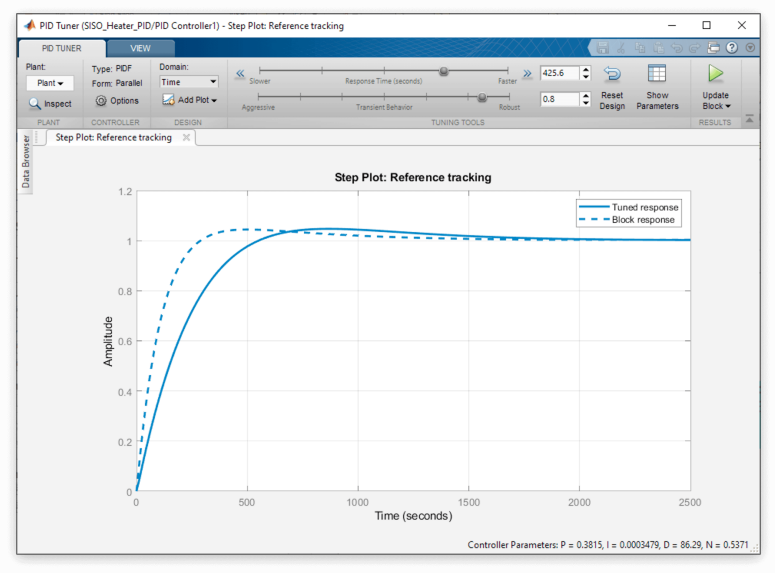


Figura D.4: MATLAB autotune PID

L'autotune permette di regolare la velocità di risposta all'impulso e la robustezza, regola l'overshooting del sistema. Automaticamente calcola i guadagni del controllore e li imposta al controllore.

D.4.1 Risultati

La simulazione effettuata tramite Simulink ha confermato la risposta al gradino scelta nell'autotune, come mostra la Figura D.5.

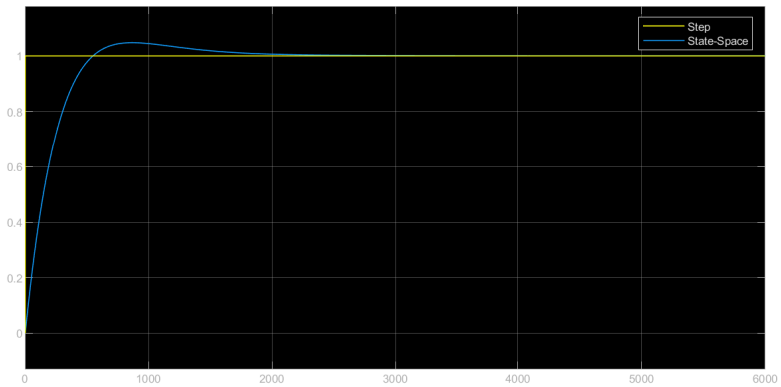


Figura D.5: Simulazione controllore PID Heater SISO

STUDENTSUPSI

D.5 Osservazioni

Gli script di MATLAB e i file Simulink sviluppati per questo laboratorio sono disponibili al seguente percorso all'interno della cartella di progetto **Cartella: Software/Heater**

STUDENTSUPSI

STUDENTSUPSI

Appendice E

Scheda Inverter

E.1 Introduzione

Sfruttando la teoria dei corsi di controllo possa sviluppare autonomamente vari tipi di controllori dal controllore di stato fino al MPC. Con l'ausilio di MATLAB e Simulink, si vuole simulare la soluzione migliore per il processo in questione. Date le problematiche riscontrate in corso d'opera del progetto, l'esercitazione si limiterà alla simulazione.

E.2 Obiettivi

Partendo dal modello fisico, modellizzare il processo matematicamente. Sviluppare tramite MATLAB e Simulink i controllori elencati. Dopodiché dopo i risultati ottenuti in fase di simulazione dedurre se i controllori proposti sono ottimali per l'applicazione. Eventualmente testare altre tipologie di controllori per identificare una soluzione soddisfacente. Infine documentare il tutto tramite un rapporto dettagliato.

E.3 Modelizzazione

Il sistema deve essere modellato attraverso le leggi della fisica o altrimenti tramite sistemi di identificazione. Visto che il processo fisico dell'inverter si può riassumere in un circuito molto semplice, il consiglio è quello di sfruttare le leggi dell'analisi circuitale. In Figura E.1 il circuiti dell'inverter.

STUDENTSUPSI

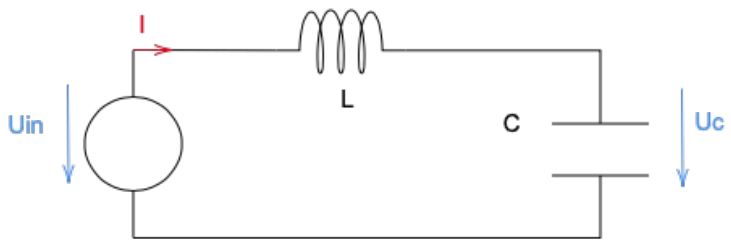


Figura E.1: Circuito di modellizzazione Inverter

E.4 Controllori

I controllori da sviluppare sono:

- Controllore di stato
- Controllore polinomiale

E.4.1 Controllore di stato

Il controllore di stato permette in alcuni casi di spostare i poli del sistema. Data la struttura del controllore riportata dalla Figura E.2 calcolare:

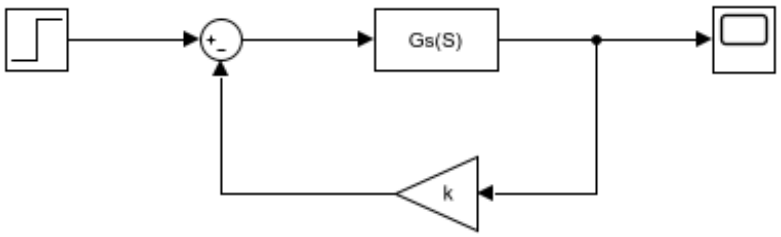


Figura E.2: Struttura controllore di stato

- Funzione di trasferimento open loop
- Scegliere i poli adeguati
- Applicare la formula per calcolare k
- Eseguire una simulazione "continua" con Simulink
- Eseguire una simulazione "discreta" con Simulink

STUDENTSUPSI

E.4.2 Controllore polinomiale

Il controllore polinomiale permette di spostare i poli del sistema a piacimento, per questo motivo è molto interessante capire se, per ottenere un buon controllore è sufficiente spostare i poli. Data la struttura del controllore riportata dalla Figura E.3, seguire il procedimento:

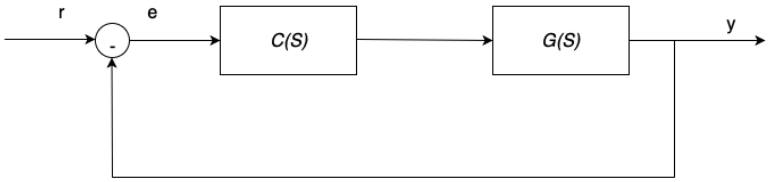


Figura E.3: Struttura controllore polinomiale

- Identificare la funzione di trasferimento open-loop del sistema
- Scegliere la struttura del controllore polinomiale adeguata per il sistema.
- Scegliere i poli (utilizzare le tecniche conosciute per il piazzamento dei poli)
- Identificare i parametri del controllore polinomiale

La funzione di trasferimento del controllore polinomiale rispetta la seguente forma:

$$C(s) = \frac{S^N\alpha_N + S^{N-1}\alpha_{N-1}\dots S\alpha_1 + \alpha_0}{S^N + S^{N-1}\beta_{N-2}\dots S\beta_1 + \beta_0} \tag{E.1}$$

E.5 Data

Nella Tabella E.5 sono descritti i parametri del sistema

Variabile	**Valore**
Induttanza $[L]$	10 $[mH]$
Condensatore $[C]$	220 $[\mu F]$

Tabella E.1: Parametri del sistema

STUDENTSUPSI

STUDENTSUPSI

Appendice F

Soluzioni Inverter

F.1 Modellizzazione Inverter

Si deve modellare il sistema, il processo dell'inverte in questo caso lo si può vedere come un circuito elettrico RLC come da figura F.1. Analizzando il circuito si ricava la funzione di trasferimento in laplace.

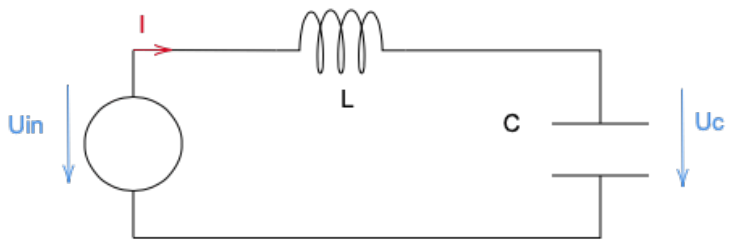


Figura F.1: Circuito di modellizzazione Inverter

$$G(S) = \frac{U_c}{U_{in}} = \frac{\frac{1}{SC}}{\frac{1}{SC} + SL} = \frac{1}{s^2LC + 1} \tag{F.1}$$

Oppure grazie alle leggi di kirchoff otteniamo il modello nello spazio degli stati, attraverso le seguenti relazioni.

$$\begin{cases} U_L = L \cdot \dfrac{\partial I}{\partial t} \\ I_C = C \cdot \dfrac{\partial U_c}{\partial t} \end{cases} \rightarrow \begin{cases} U_{in} = U_L + U_C \\ I = I_L = I_C \end{cases} \rightarrow \begin{cases} L \cdot \dot{I}_L = U_{in} - U_c \\ C \cdot \dot{U}_C = I_L \end{cases}$$

Nello spazio degli stati si ottiene il seguente sistema.

STUDENTSUPSI

$$\begin{cases} \begin{bmatrix} \dot{I}_L \\ \dot{U}_C \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & 0 \end{bmatrix} \cdot \begin{bmatrix} I_L \\ U_C \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} \cdot U_{in} \\ \begin{bmatrix} I_L \\ U_C \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} I_L \\ U_c \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \cdot U_{in} \end{cases} \tag{F.2}$$

F.2 Controllore di stato

Lo sviluppo di un controllore di stato richiede un buon posizionamento dei poli, anche se non è detto che si possano spostare i poli del sistema a piacimento.

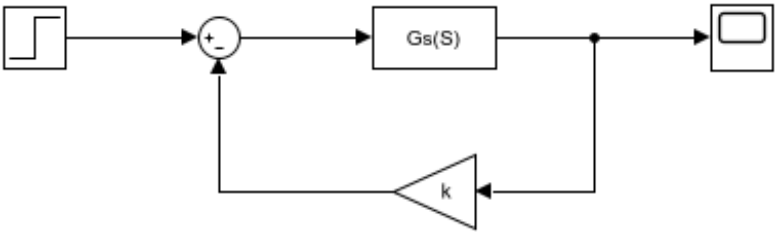


Figura F.2: Struttura controllore di stato

Per trovare il valore di K si deve applicare la seguente formula del controllore di stato.

$$\begin{aligned} det(S \cdot I - A - B \cdot K) &= (S - P_1) \cdot (S - P_2) \\ S^2 + S \cdot \frac{k_1}{L} + \frac{1 + k_2}{L \cdot C} &= s^2 + s \cdot (-P_1 - P_2) + P_1 \cdot P_2 \end{aligned} \tag{F.3}$$

In questo caso otteniamo il seguente sistema di equazioni.

$$\begin{cases} \frac{k_1}{L} = -P_1 - P_2 \\ \frac{1+k_2}{L \cdot C} = P_1 \cdot P_2 \end{cases} \rightarrow \begin{cases} k_1 = L \cdot (-P_1 - P_2) \\ k_2 = L \cdot C \cdot P_1 \cdot P_2 - 1 \end{cases} \tag{F.4}$$

Ora basta scegliere i poli in modo adeguato, in modo da mantenere il sistema stabile. Una buona prassi è verificare i poli del sistema ad anello aperto e verificarne il posizionamento sul piano di Gauss. Se i poli sono soddisfacenti si possono mantenere i poli del sistema aperto altrimenti dovremmo cercare di spostarli.

STUDENTSUPSI

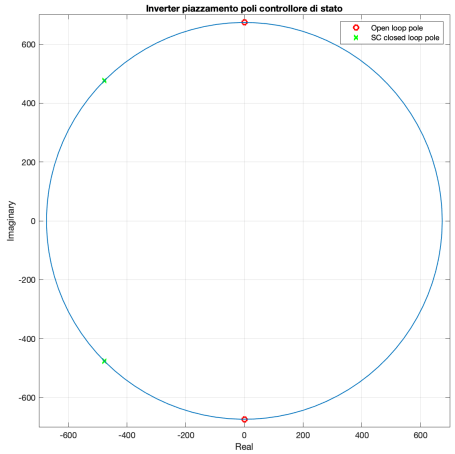


Figura F.3: Piazzamento poli controllore di stato

In questo caso i poli si trovano entrami sull'asse dei numeri immaginari ovvero sono due poli puramente complessi come mostra la Figura F.3. Per spostare i poli è necessario applicare una regola empirica, ovvero cercare di spostare i poli sulla circonferenza generata dai poli ad anello aperto. Questo vuol dire che i poli che andremmo a scegliere dovranno avere lo stesso modulo dei poli ad anello aperto.

F.2.1 Risultati

La simulazione eseguita con Simulink mostra il sistema perfettamente controllato come ce lo si aspetta. La Figura F.4 riporta la simulazione ottenuta con il controllore di stato.

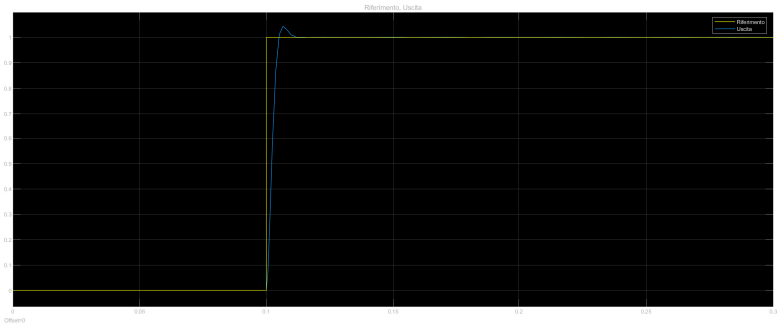


Figura F.4: Simulazione controllore di stato Inverter

Come si nota dalla simulazione il sistema controllato presenta un overshooting, in questo caso è inferiore al 10% che per un'inverter è la minima condizione per non recare danno agli utilizzatori connessi.

STUDENTSUPSI

F.3 Controllore polinomiale

Ora con la funzione di trasferimento si può realizzare un controllore polinomiale con la forma seguente.

$$C(S) = \frac{S\alpha_1 + \alpha_0}{S + \beta_0} \tag{F.5}$$

Secondo le regole di Maison calcoliamo la funzione di trasferimento ad anello chiuso basandoci sulla struttura del controllore polinomiale come da Figura F.5.

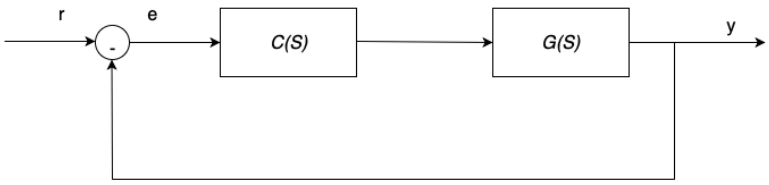


Figura F.5: Struttura controllore polinomiale

$$\begin{aligned} G_{cl} &= \frac{C(S) \cdot G(S)}{1 + C(S) \cdot G(S)} \\ &= \frac{S\alpha_1 + \alpha_0}{S^3 \cdot LC + S^2 \cdot LC\beta_0 + S \cdot (\alpha_1 + 1) + \beta_0 + \alpha_0} \end{aligned} \tag{F.6}$$

Ora si devono identificare i parametri del controllore, come appreso durante i corsi. Si sceglie dunque la funzione desiderata di ordine pari alla funzione ad anello chiuso. In questo caso si sceglie un polinomio di questo tipo.

$$\begin{aligned} G_{des}(S) &= (S - p_1) \cdot (S - p_2) \cdot (S - p_3) \\ &= S^3 + S^2(-p_1 - p_2 - p_3) + S(p_1p_2 + p_1p_3 + p_2p_3) - p_1p_2p_3 \end{aligned} \tag{F.7}$$

Si deve ora risolvere il seguente sistema di equazioni per piazzare i poli nel punto desiderato.

$$\begin{cases} -p_1 - p_2 - p_3 = \beta_0 \\ p_1p_2 + p_1p_3 + p_2p_3 = \dfrac{\alpha_1 + 1}{LC} \\ -p_1p_2p_3 = \dfrac{\beta_0 + \alpha_0}{LC} \end{cases} \tag{F.8}$$

L'ultimo passaggio resta quello di scegliere i poli, per scegliere in modo adeguato i poli ad anello chiuso si deve analizzare i poli ad anello aperto. Dalla funzione di trasferimento G(S) otteniamo i poli.

STUDENTSUPSI

$$0 = S^2 + \frac{1}{LC}$$
$$s_1 = \frac{i}{\sqrt{LC}} = i \cdot 674$$
$$s_2 = \frac{-i}{\sqrt{LC}} = -i \cdot 674$$

Come per il controllore di stato per piazzare i poli è meglio se sono tutti su di una circonferenza. In questo caso si possono usare i medesimi poli del controllore di stato, aggiungendone uno puramente reale come mostrato in Figura F.6. I poli scelti permettono ora di ottenere un sistema stabile.

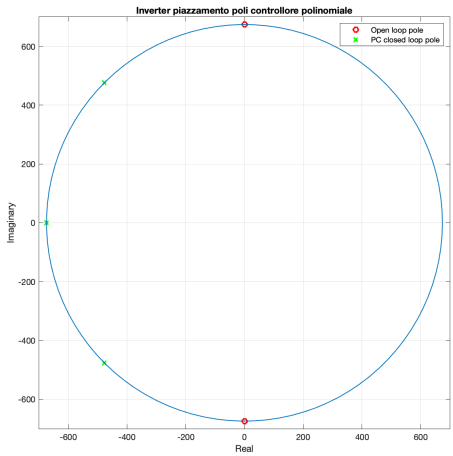


Figura F.6: Struttura piazzamento poli

F.3.1 Risultati

La simulazione ha riportato un buon risultato come mostra la Figura F.7.

STUDENTSUPSI

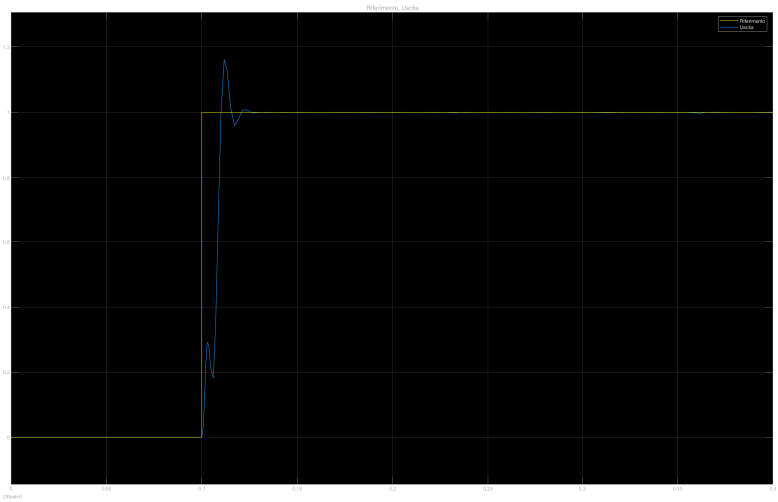


Figura F.7: Simulazione controllore polinomiale Inverter

Dalla simulazione si nota che il controllore deve essere ancora ottimizzato, in quanto l'overshooting e superiore al 10%, inoltre la transizione del segnale non è pulita è presente un spike, associabile ai poli scelti.

F.4 Osservazioni

Gli script di MATLAB e i file Simulink sviluppati per questo laboratorio sono disponibili al seguente percorso all'interno della cartella di progetto **Cartella: Software/Inverter**

F.5 Dati

Per la realizzazione del modello descritto dalle Equazioni del sistema F.1 necessitiamo di alcuni dati, i quali sono descritti in dettaglio nella Tabella F.5.

Variabile	**Valore**
Induttanza $[L]$	10 $[mH]$
Condensatore $[C]$	220 $[\mu F]$

Tabella F.1: Parametri del sistema

STUDENTSUPSI

Appendice G

Nomenclature

Qui sono riassunte tutte le nomenclature particolari utilizzate nel documento

G.1 Nomenclature Heater

Variabile	**Valore**
Calore emesso dal transistor $[Q_1]$	0 to 1 $[W]$
Calore emesso dal transistor $[Q_2]$	0 to 1 $[W]$
Calore specifico ferro $[C_p]$	450 $[\frac{J}{Kg \cdot K}]$
Calore specifico plastica $[C_{p,s}]$	$\sim 2500[\frac{J}{Kg \cdot K}]$
Coefficiente di calore $[\alpha_1]$	0.01 $[\frac{W}{\%calore}]$
Coefficiente di calore $[\alpha_2]$	0.01 $[\frac{W}{\%calore}]$
Conducibilità termica $[U]$	10 $[\frac{W}{m \cdot K}]$
Conducibilità termica $[U_s]$	2 $[\frac{W}{m \cdot K}]$
Conducibilità termica $[k]$	0.2 $[\frac{W}{m \cdot K}]$
Costante di Bolzmann $[\sigma]$	$5.67 \cdot 10^{-8}[\frac{W}{m^2 \cdot k^4}]$
Emissività $[\epsilon]$	0.9
Massa del sensore $[m_s]$	0.002 $[Kg]$
Massa del transistor $[m]$	0.004 $[Kg]$
Temperatura ambiente $[T_a]$	Misurata $[K]$
Temperatura heater 1 $[T_{H1}]$	Stato $[K]$
Temperatura heater 2 $[T_{H2}]$	Stato $[K]$
Temperatura sensore 1 $[T_{C1}]$	Misurata $[K]$
Temperatura sensore 2 $[T_{C2}]$	Misurata $[K]$
Spessore di tra heater e sensore $[\Delta x]$	0.001 $[m]$
Superficie di contatto tra heater $[A_s]$	$2 \cdot 10^{-4}[m^2]$
Superficie di contatto tra heater e sensore $[A_c]$	$2.5 \cdot 10^{-5}[m^2]$
Superficie di dissipazione tra heater e ambiente $[A]$	$1 \cdot 10^{-3}[m^2]$

Tabella G.1: Parametri del sistema

STUDENTSUPSI

G.2 Nomenclature Inverter

Variabile	**Valore**
Induttanza $[L]$	10 $[mH]$
Condensatore $[C]$	220 $[\mu F]$

Tabella G.2: Parametri del sistema

STUDENTSUPSI