



Corso di Automazione industriale

Lezione 9

PLC – Generazione automatica di testo strutturato

Introduzione

È sempre più spesso necessario fornire ai clienti diverse soluzioni hardware (con diverse marche di PLC).

Lo sviluppo del codice deve quindi essere, in questi casi, il più possibile separato dall'implementazione vera e propria nell'ambiente di sviluppo del PLC.

Per questo motivo, negli ultimi anni stanno nascendo software che consentono, a partire da un linguaggio di più alto livello, di generare codice compatibile con la maggior parte degli IDE presenti sul mercato.

Toolchain

Attraverso l'ambiente di sviluppo Matlab Simulink (e il pacchetto PLC-Coder) è possibile creare software di controllo compatibile con la maggior parte delle marche di PLC.

L'architettura dei controlli in Simulink è diversa dalle normali architetture che si è abituati a vedere, per questo motivo è necessaria un'introduzione all'ambiente.

Introduzione a Matlab

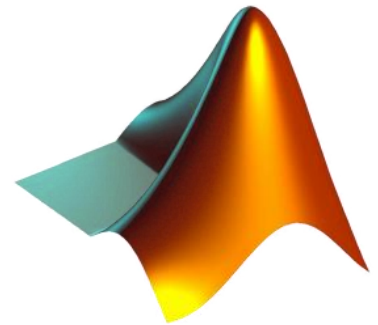
Cos'è Matlab?

Matlab (MATrix LABoratory) è un ambiente di sviluppo utilizzato per il calcolo numerico.

Si basa sull'utilizzo di matrici e sulle operazioni tra esse.

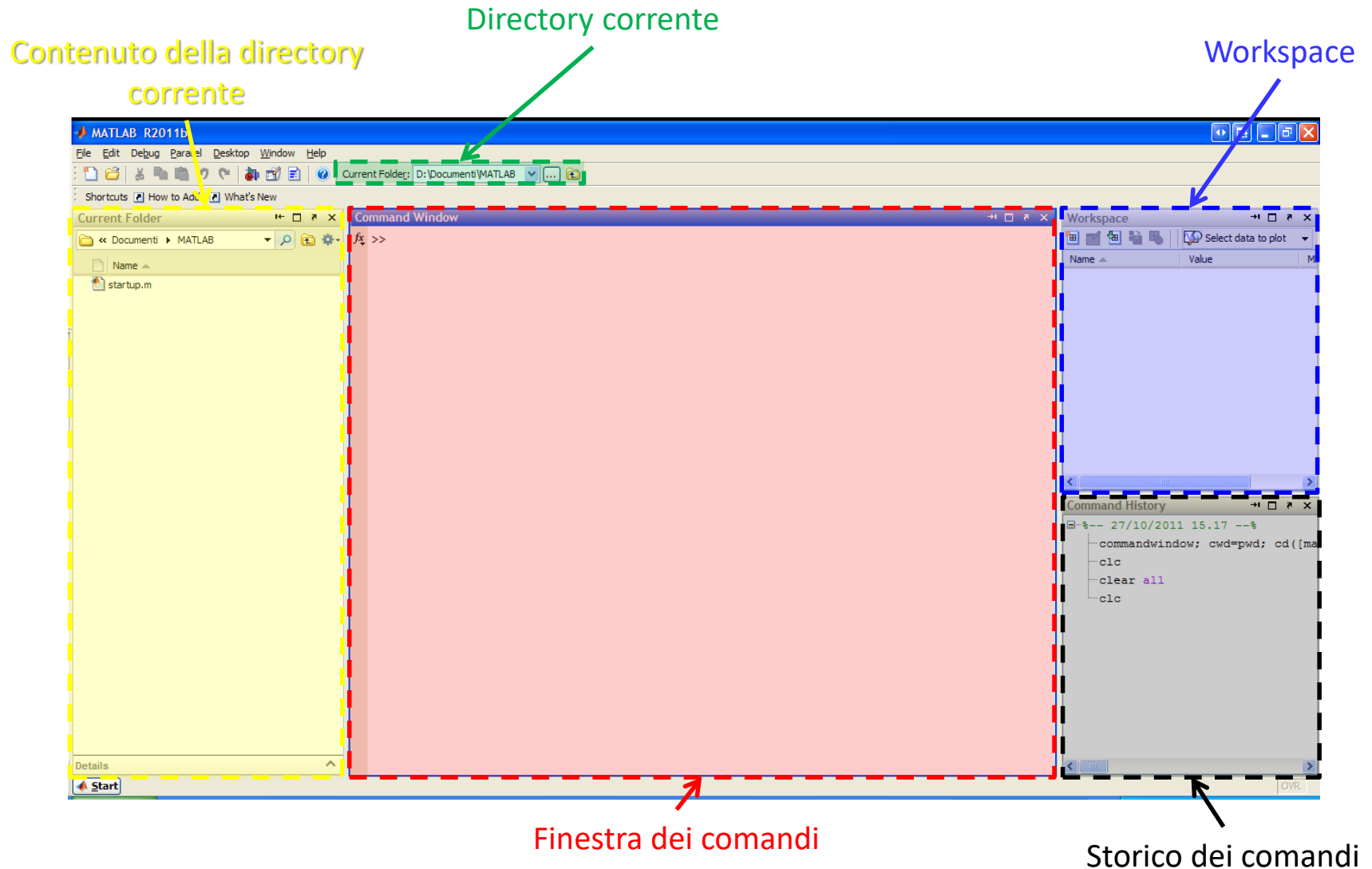
Utilizzato per:

- Calcoli ingegneristici
- Simulazioni
- Implementazioni software di alto livello



N.B.: In generale è molto spinto verso la modularità

Introduzione a Matlab

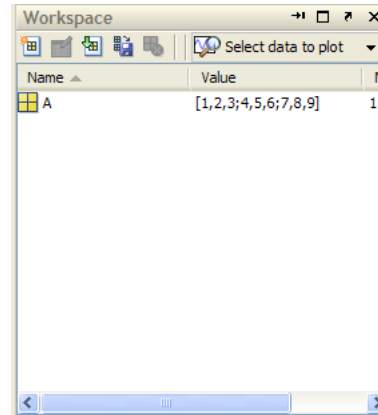


Introduzione a Matlab

```
>> A=[1 2 3;4 5 6;7 8 9]
```

A =

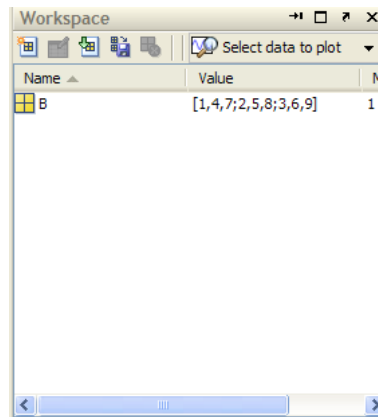
```
1 2 3
4 5 6
7 8 9
```



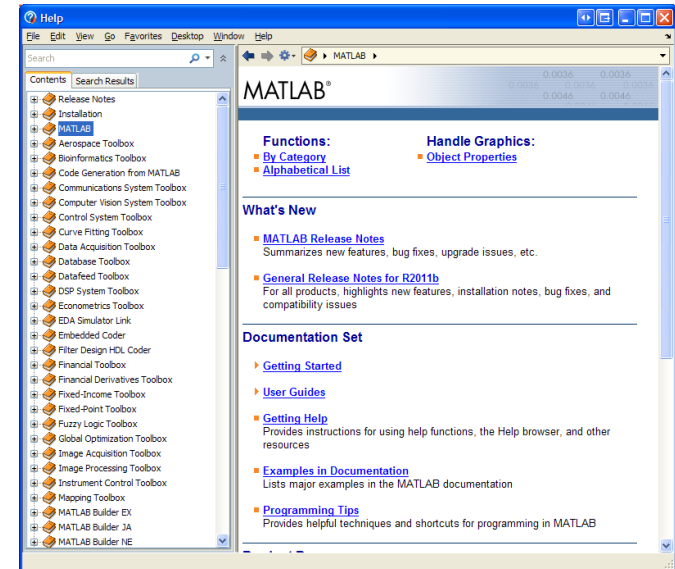
```
>> B=A'
```

B =

```
1 4 7
2 5 8
3 6 9
```



```
>> doc
```



Introduzione a Matlab

Funzioni base:

Strumenti:

- Avvio e spegnimento
- Finestra dei comandi
- Guide
- Gestione file
- Tool di sviluppo
- Sistema

Programmazione e tipi di dato:

- Tipi di dato
- Conversione di tipi
- Operatori e caratteri speciali
- Stringhe
- Operazioni su bit
- Operatori logici
- Operatori relazionali
- Operazioni di set
- Operazioni su data e ora
- Programmazione Matlab

Import ed export:

- Apertura e salvataggio file
- File di testo
- Fogli di calcolo
- I/O di basso livello
- Immagini
- Dati scientifici
- Audio e video
- Documenti XML
- Mapping della memoria
- Costruzione nome file
- Compressione di file
- Accesso ad internet

Programmazione ad oggetti:

- Classi e oggetti
- Classi Handle
- Eventi e listener
- Meta-classi
- Enumeratori
- Array eterogenei

Matematiche:

- Array e matrici
- Algebra lineare
- Matematica elementare
- Polinomi
- Interpolazione e elaborazione geometrica
- Conversione di coordinate cartesiane
- Metodi numerici non lineari
- Funzioni speciali
- Matrici
- Costanti

Grafica:

- Plot e grafici base
- Strumenti di plot
- Plot di annotazione
- Plot specializzati
- Stampa
- Handle grafici

Analisi dei dati:

- Operazioni base
- Statistica descrittiva
- Filtraggio e convoluzione
- Regressione
- Trasformata di Fourier
- Derivate e integrali
- Serie temporali

Visualizzazione 3D:

- Plot di superfici e mesh
- Controlli di visualizzazione
- Illuminazione
- Trasparenza
- Visualizzazione di volumi

Introduzione a Matlab

Toolbox:

Matlab:

- Aerospace Toolbox
- Bioinformatics Toolbox
- Communications System Toolbox
- Computer Vision System Toolbox
- **Control System Toolbox**
- Curve Fitting Toolbox
- Data Acquisition Toolbox
- Database Toolbox
- Datafeed Toolbox
- DSP System Toolbox
- Econometrics Toolbox
- EDA Simulator Link
- **Embedded Coder**
- Filter Design HDL Coder
- Financial Derivatives Toolbox
- Financial Toolbox
- Fixed-Income Toolbox
- **Fixed-Point Toolbox**
- Fuzzy Logic Toolbox
- Global Optimization Toolbox
- Image Acquisition Toolbox
- Image Processing Toolbox
- Instrument Control Toolbox
- Mapping Toolbox
- MATLAB Builder EX
- MATLAB Builder JA
- MATLAB Builder NE
- MATLAB Coder
- MATLAB Compiler
- MATLAB Distributed Computing Server
- MATLAB Report Generator
- Model Predictive Control Toolbox
- Model-Based Calibration Toolbox
- Neural Network Toolbox
- OPC Toolbox
- **Optimization Toolbox**
- Parallel Computing Toolbox
- Partial Differential Equation Toolbox
- Phased Array System Toolbox
- RF Toolbox
- **Robust Control Toolbox**
- Signal Processing Toolbox
- SimBiology
- Spreadsheet Link EX
- Statistics Toolbox
- Symbolic Math Toolbox
- **System Identification Toolbox**
- SystemTest
- Vehicle Network Toolbox
- Wavelet Toolbox

Simulink:

- Aerospace Blockset
- DO Qualification Kit
- Gauges Blockset
- IEC Certification Kit
- Real-Time Windows Target
- SimDriveline
- SimElectronics
- SimEvents
- SimHydraulics
- SimMechanics
- SimPowerSystems
- SimRF
- Simscape
- Simulink 3D Animation
- Simulink Code Inspector
- Simulink Coder
- Simulink Control Design
- Simulink Design Optimization
- Simulink Design Verifier
- Simulink Fixed Point
- Simulink HDL Coder
- **Simulink PLC Coder**
- **Simulink Report Generator**
- Simulink Verification and Validation
- **Stateflow**
- xPC Target

Linguaggio di scripting

Fondamenti del linguaggio

Il linguaggio Matlab è molto simile al classico C.

Le parti di codice possono essere editate direttamente nella finestra dei comandi oppure raccolte in file *.m come sequenza.

Matlab consente una gestione semplificata delle variabili:

- Le variabili di cui non è noto il tipo sono definite double
- Le variabili possono cambiare runtime la dimensione di allocazione (al massimo si ottiene un messaggio di warning)
- Non è necessaria alcuna dichiarazione, è sufficiente assegnare un valore alla variabile
- E' possibile inglobare in matrici particolari diverse tipologie di dato
- Le strutture possono avere campi variabili runtime

Linguaggio di scripting

Fondamenti del linguaggio

Matlab consente l'implementazione di funzioni:

- Ogni funzione potrà avere o meno parametri d'ingresso
- Il numero di parametri d'ingresso può variare (è possibile omettere dei parametri)
- Ogni funzione potrà avere zero, una o più variabili di uscita
- Tutte le variabili locali alla funzione hanno lifetime limitato all'esecuzione della stessa
- Dall'interno delle funzioni è possibile richiamare qualunque altra funzione raggiungibile (scritte a mano o incorporate nei toolbox)

Linguaggio di scripting

Fondamenti del linguaggio

Matlab consente l'utilizzo dei costrutti:

- If
- elseif
- Switch
- For
- While
- Break
- Try / catch

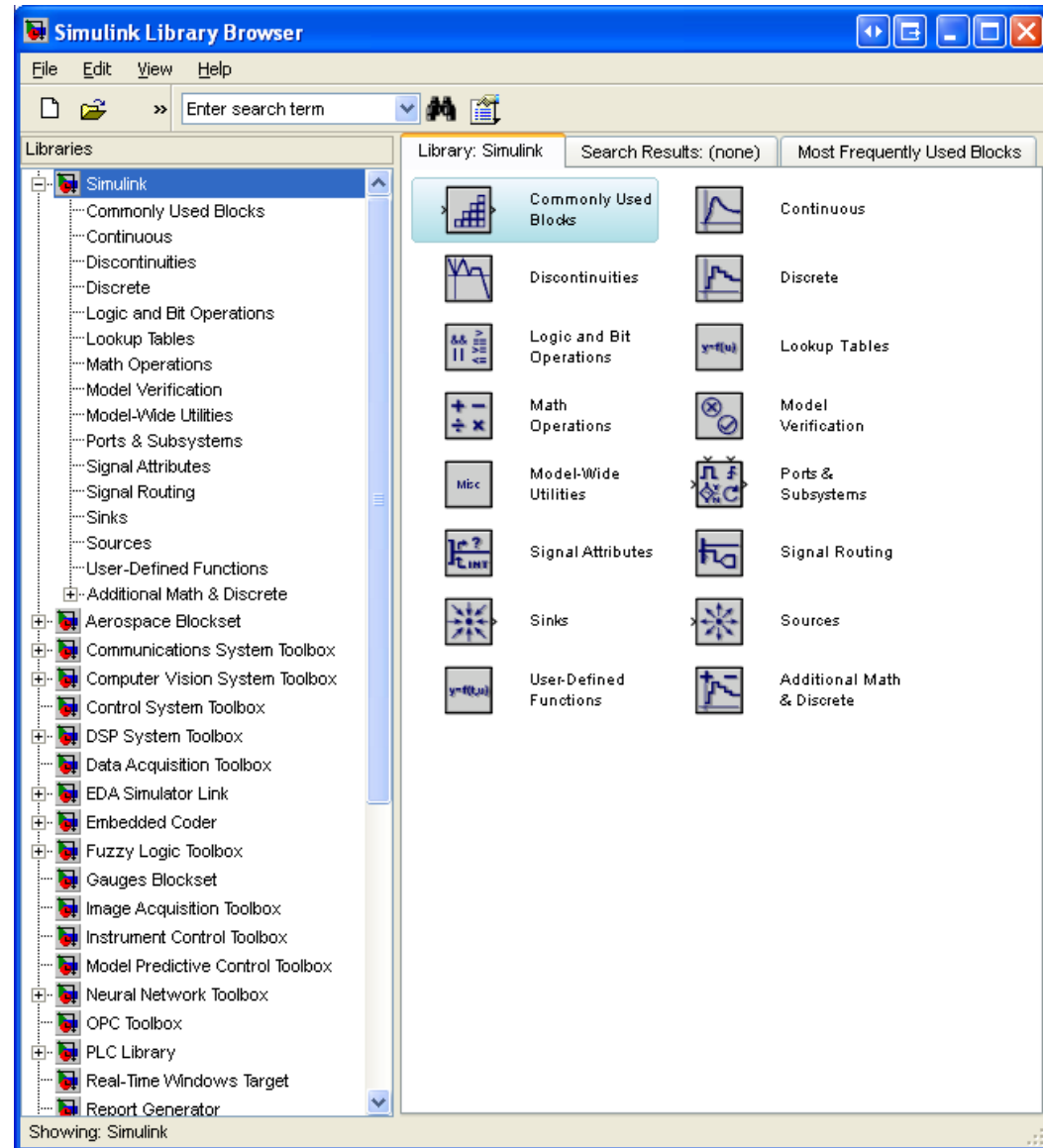
N.B.: i file di scripting hanno estensione .m e sono dei normali file di testo (editabili con un normale editor).

I file contenenti le variabili hanno, invece estensione .mat e sono file binari.

Introduzione a Simulink

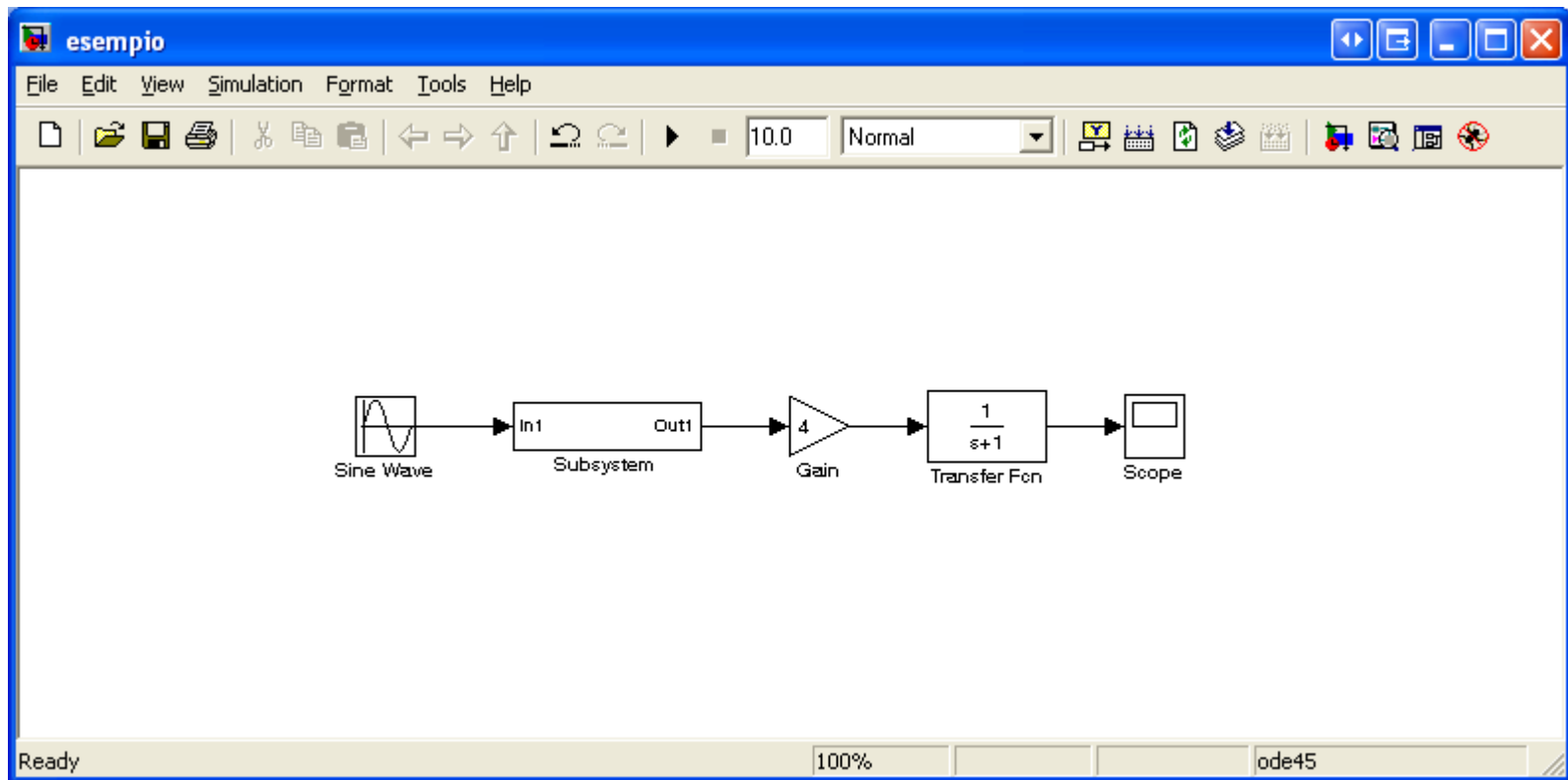
Simulink è un toolbox grafico di MATLAB per la simulazione dei sistemi dinamici lineari e non lineari a tempo continuo, discreto o misto.

Permette inoltre la progettazione model-based dei sistemi dinamici e delle logiche di controllo.



Introduzione a Simulink

Si basa su un editor grafico interattivo per l'assemblaggio e la gestione dei diagrammi a blocchi



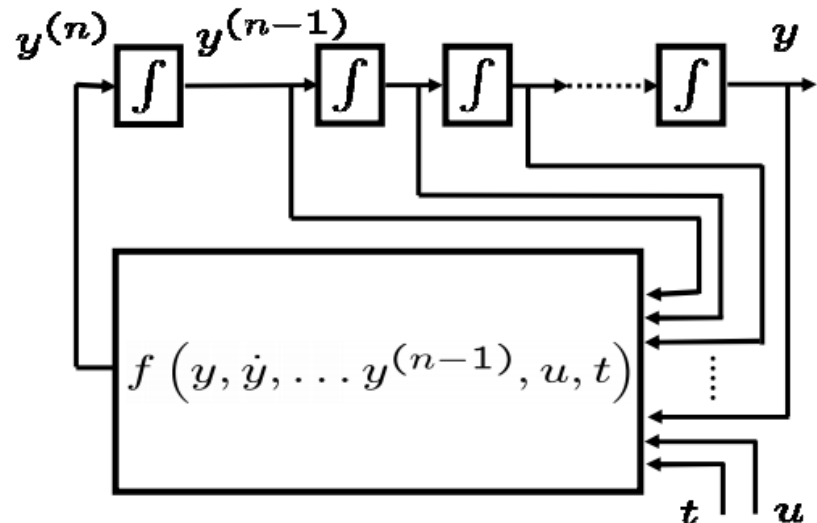
Introduzione a Simulink

Per funzionare utilizza dei risolutori numerici di equazioni differenziali

$$\frac{d^n y}{d t^n} = f\left(\frac{d^{n-1} y}{d t^{n-1}}, \dots, \frac{d y}{d t}, y, u, t\right)$$

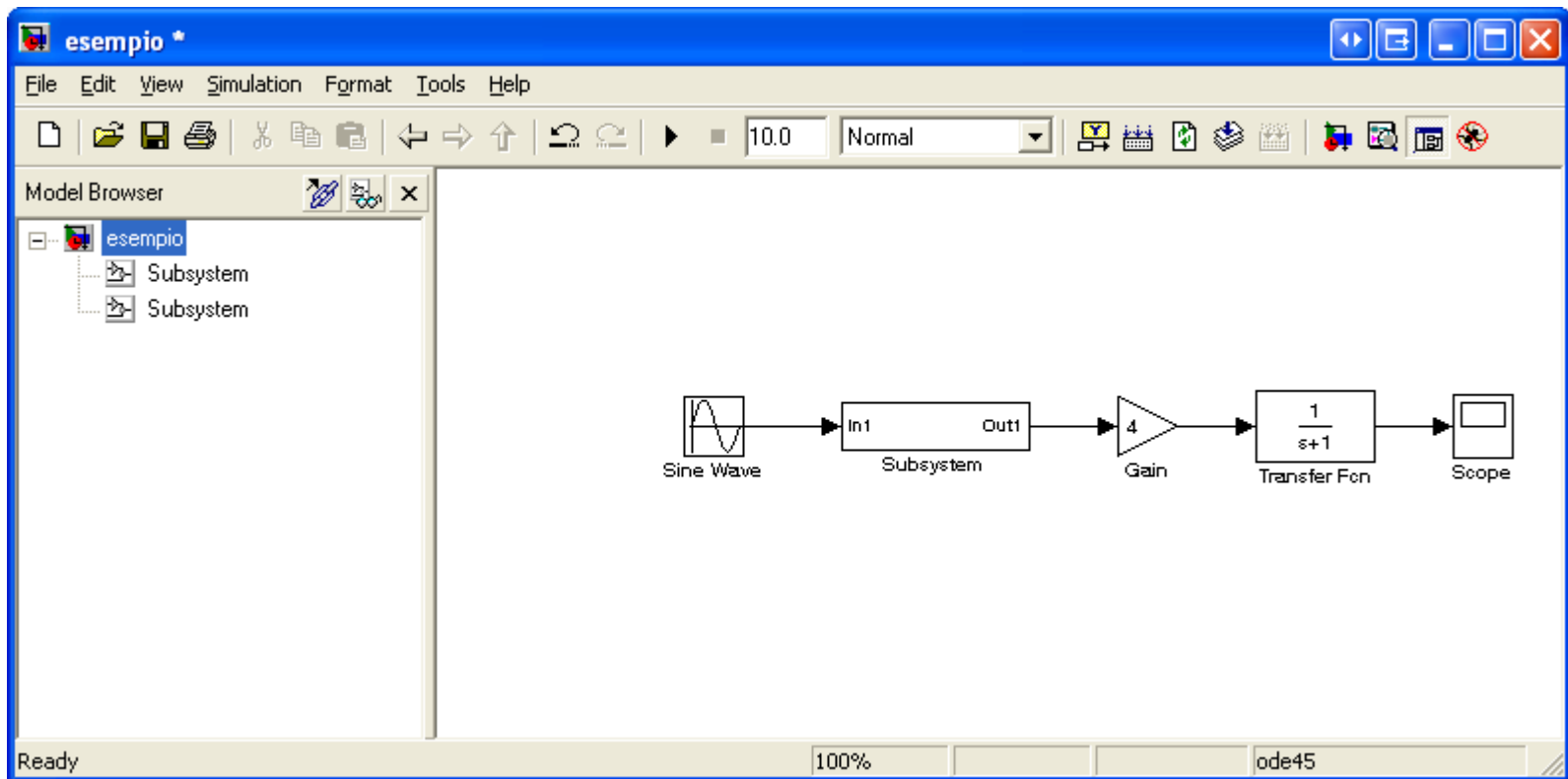
Approccio alla
realizzazione di uno
schema Simulink

Equazione differenziale
generica, anche non-lineare,
anche con elementi tempo-
varianti, in forma esplicita.



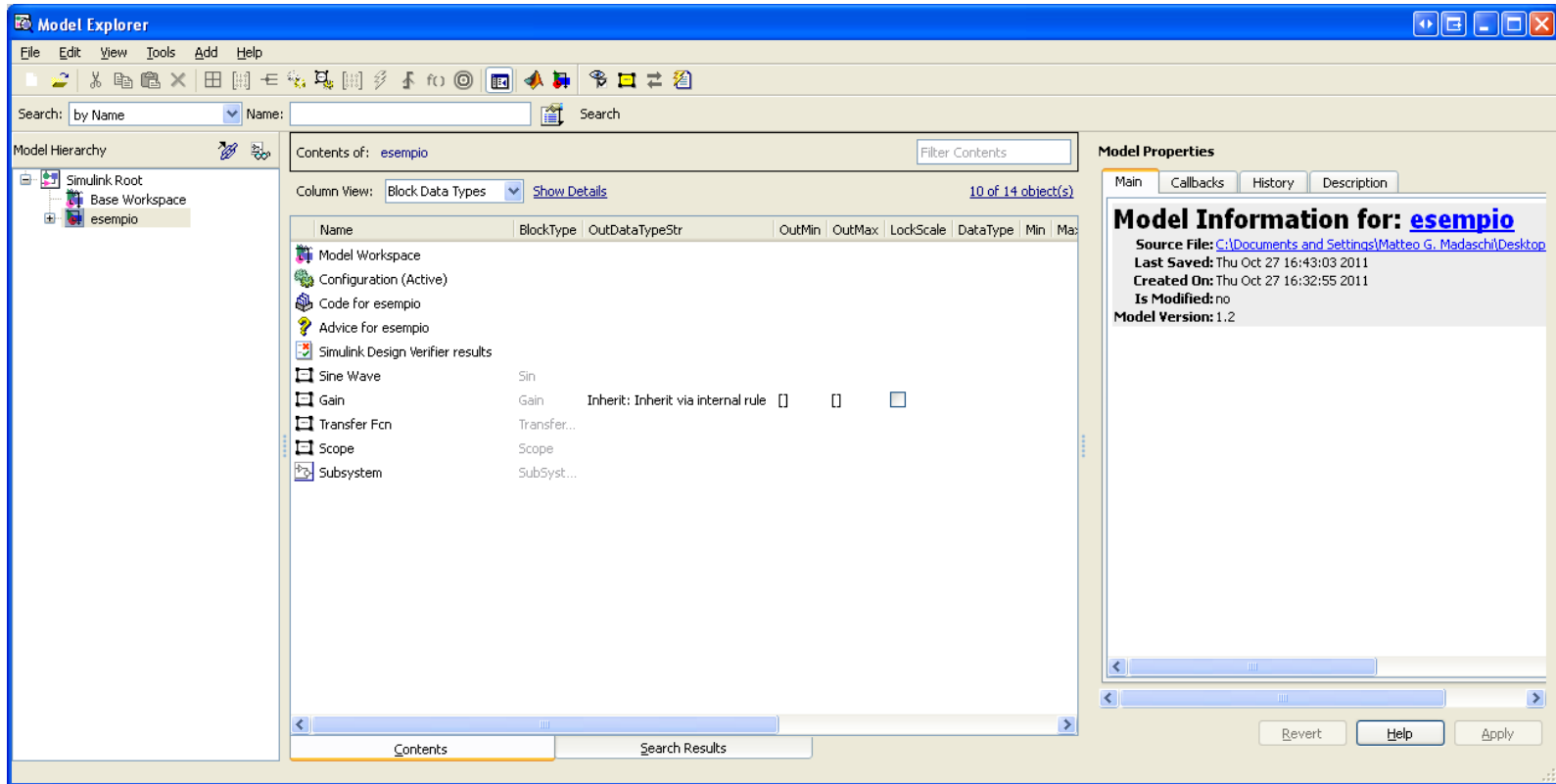
Introduzione a Simulink

Possibilità di gestire progetti complessi dividendo i modelli in gerarchie di componenti con la creazione sottosistemi



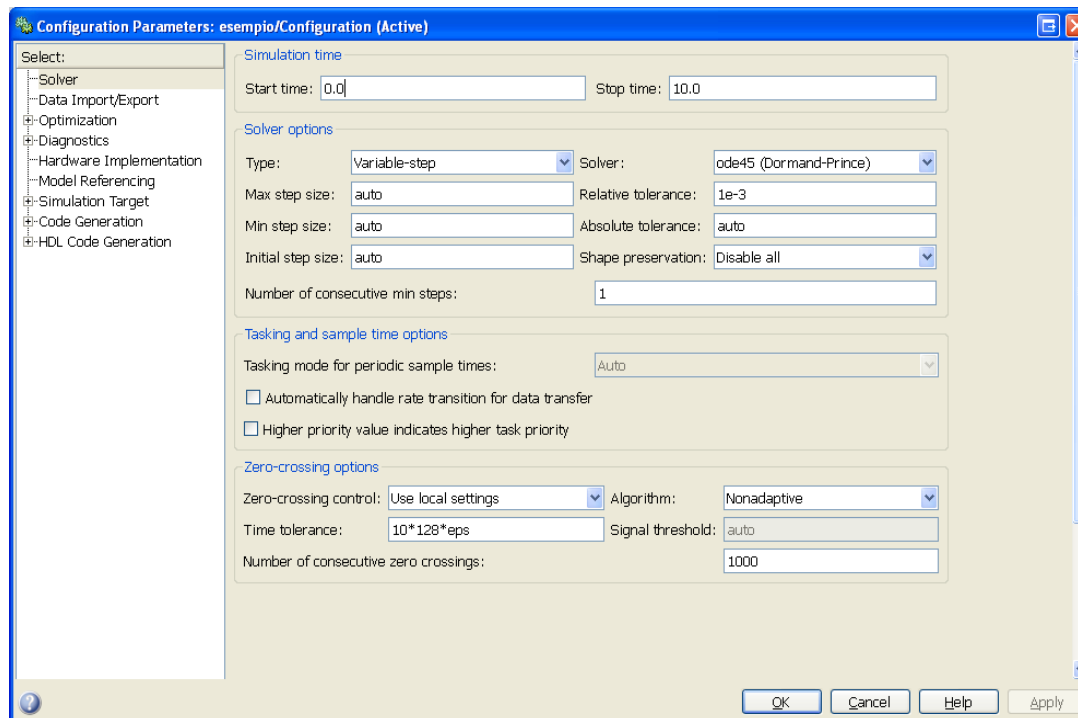
Introduzione a Simulink

Model Explorer consente di navigare, creare, configurare e ricercare tutti i segnali, i parametri, le proprietà e il codice generato associato a un modello



Introduzione a Simulink

Modi di simulazione (Normal, Accelerator e Rapid Accelerator) per l'esecuzione di simulazioni in modo interpretato o alla velocità del codice C compilato mediante solutori fissi o variabili



Introduzione a Simulink

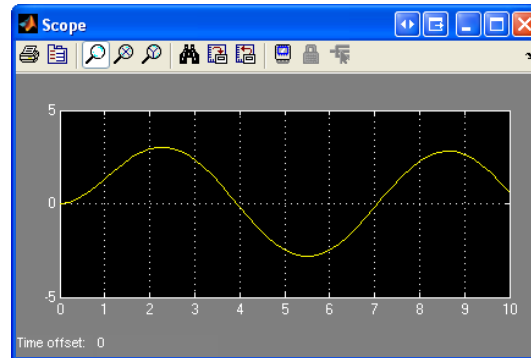
Offre inoltre:

- Possibilità di includere nei modelli funzioni scritte in codice C/Fortran
- Le API (Application Programming Interfaces) abilitano la connessione ad altri programmi di simulazione e consentono di includere codice scritto manualmente
- Blocco funzionale Embedded MATLAB per importare algoritmi MATLAB in Simulink

Introduzione a Simulink

Offre inoltre:

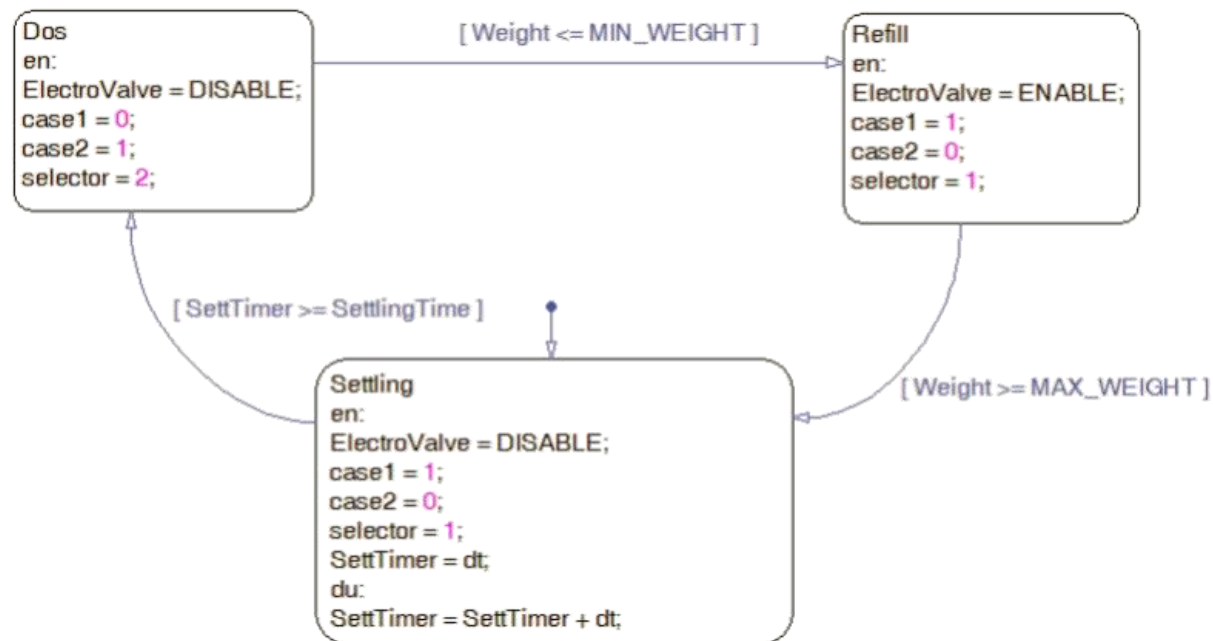
- Accesso totale al Workspace di MATLAB per l'analisi e la visualizzazione dei risultati, per personalizzare l'ambiente di modellazione e definire segnali, parametri e test
- Strumenti di analisi e diagnostica per assicurare la coerenza del modello e identificare potenziali errori
- Debugger e profiler grafici



Introduzione a Simulink

Offre inoltre:

- Implementazione di automi a stati finiti attraverso State Flow



Generazione del codice

Cosa significa generare codice in Simulink?

Generare codice in Simulink significa trasformare un modello (file *.mdl) in uno o più file compilabili, in grado di eseguire le medesime operazioni osservabili in Matlab su una diversa piattaforma.

Attualmente è possibile:

- Generare codice C/C++ compatibile con la maggior parte dei micro (toolbox real-time workshop)
- Generare codice conforme IEC61131 compatibile con la maggior parte dei PLC (toolbox PLC coder)

Generazione del codice

Generazione codice C / C++

Real-time workshop consente di tradurre un qualunque modello, realizzato con componenti delle librerie Simulink, in codice C/C++ da eseguire in una ciclica real-time.

N.B.: in questo corso non ci occupiamo di generazione di codice C / C++ perché non conforme IEC 61131.

Generazione del codice

Generazione codice in testo strutturato

Dalla versione R2010a di Matlab è stato incluso nel tool PLC coder (giunto alla versione 1.2.1), con lo scopo di generare automaticamente codice in testo strutturato a partire da blocchi simulink.

Nell'ambito di un progetto universitario è stato realizzato un controllo macchina (ibrido logico - modulante) in ambiente Simulink (utilizzando le librerie PLC coder).

Esempio di generazione

Generazione codice in testo strutturato

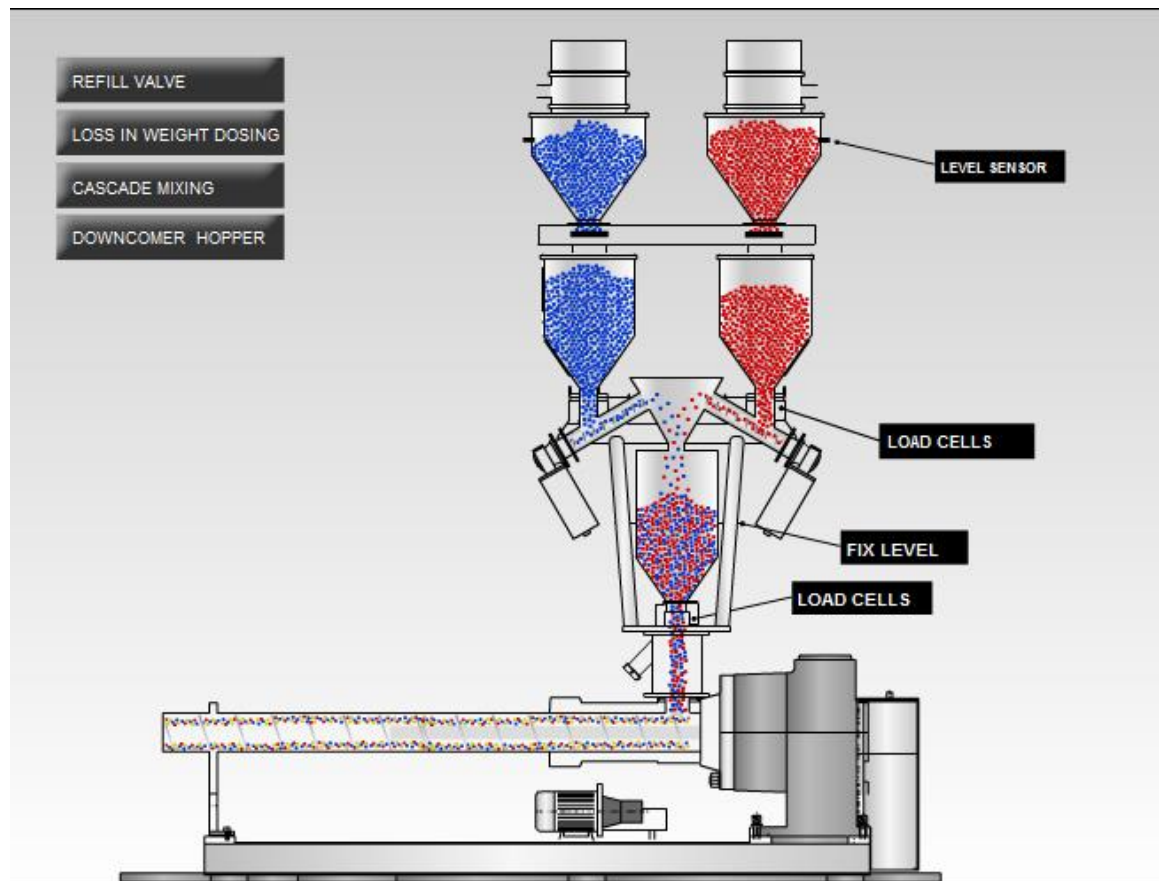
Di questo modello è stato generato testo strutturato per le piattaforme:

- B&R
- Siemens
- Allen Bradley

Il codice auto-generato è stato incluso nei progetti dei diversi ambienti di sviluppo ed è stato testato su una vera e propria macchina.

Esempio di generazione

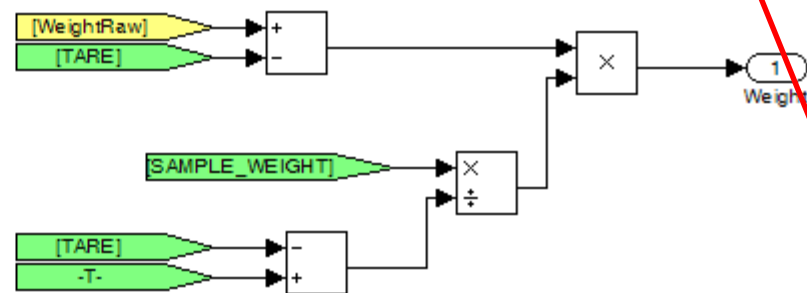
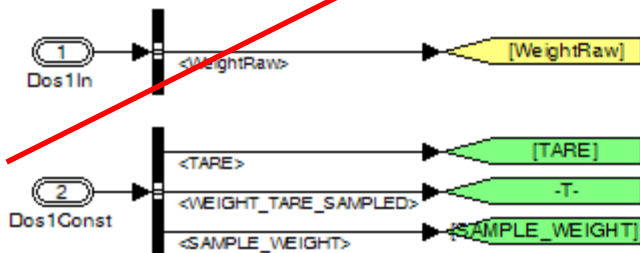
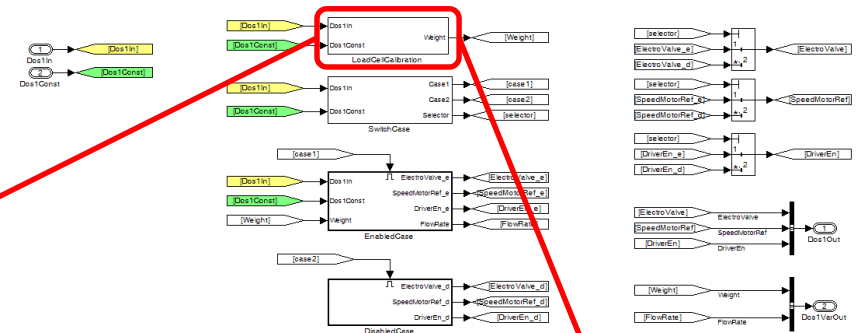
Funzionamento della macchina



Esempio di generazione

Software di controllo

Calibrazione cella di carico



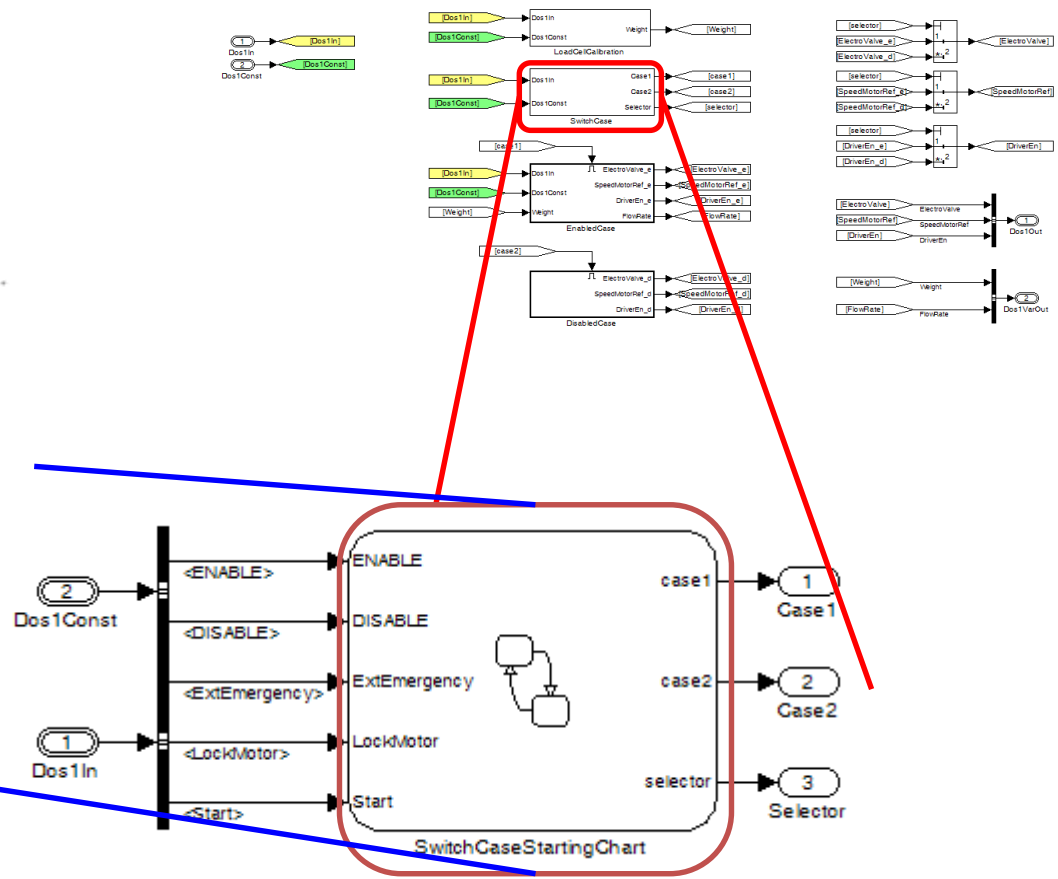
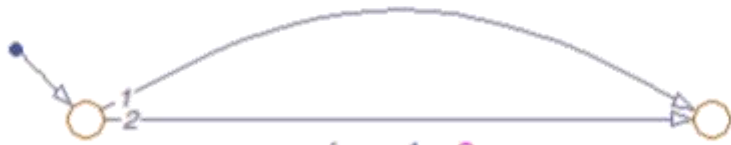
Esempio di generazione

Software di controllo

Switch abilitazione

```
[Start == ENABLE &&...  
ExtEmergency == ENABLE &&...  
LockMotor == ENABLE]  
(case1 = 1;  
case2 = 0;  
selector = 1;)
```

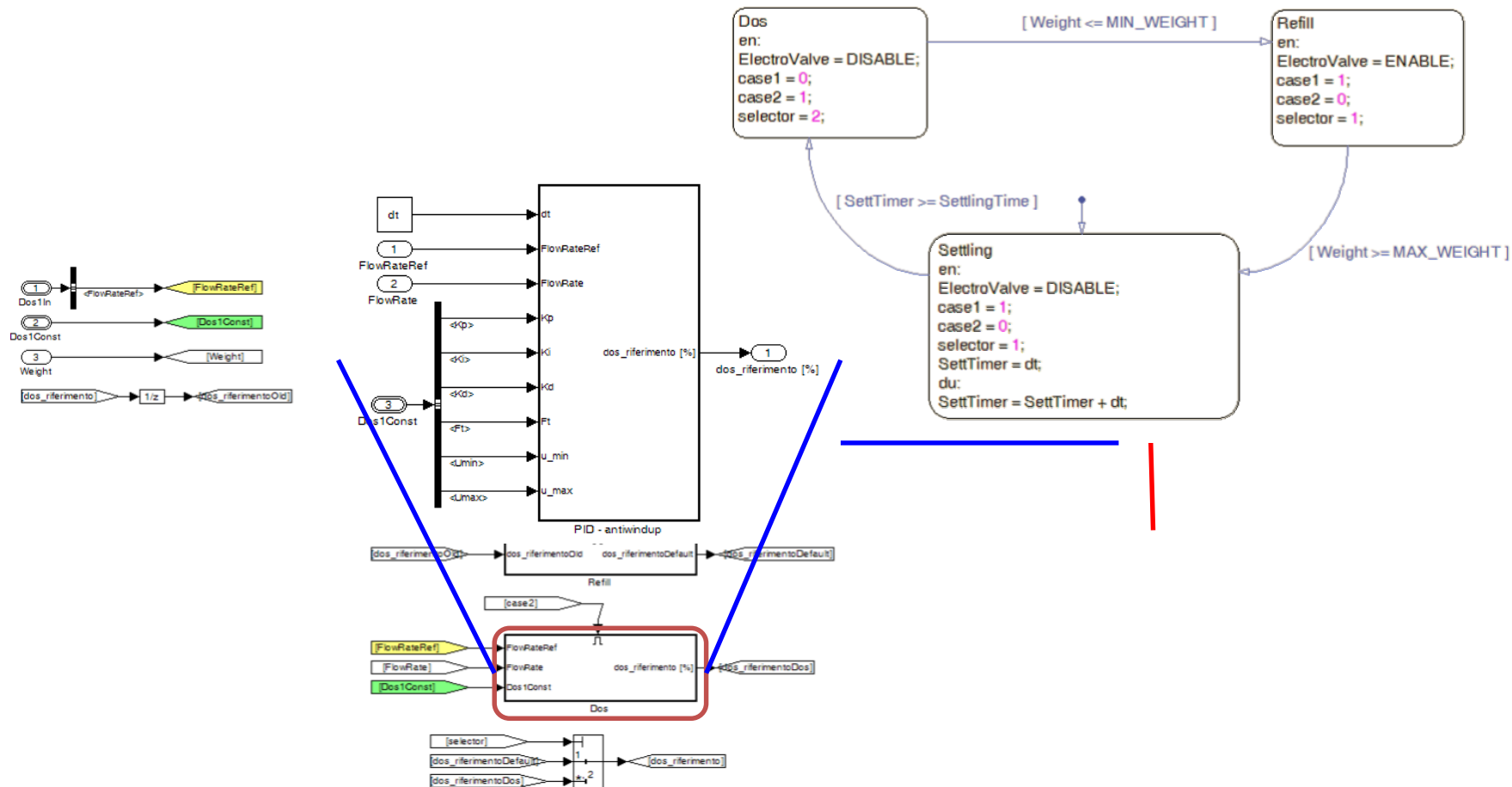
```
{case1 = 0;  
case2 = 1;  
selector = 2;}
```



Esempio di generazione

Software di controllo

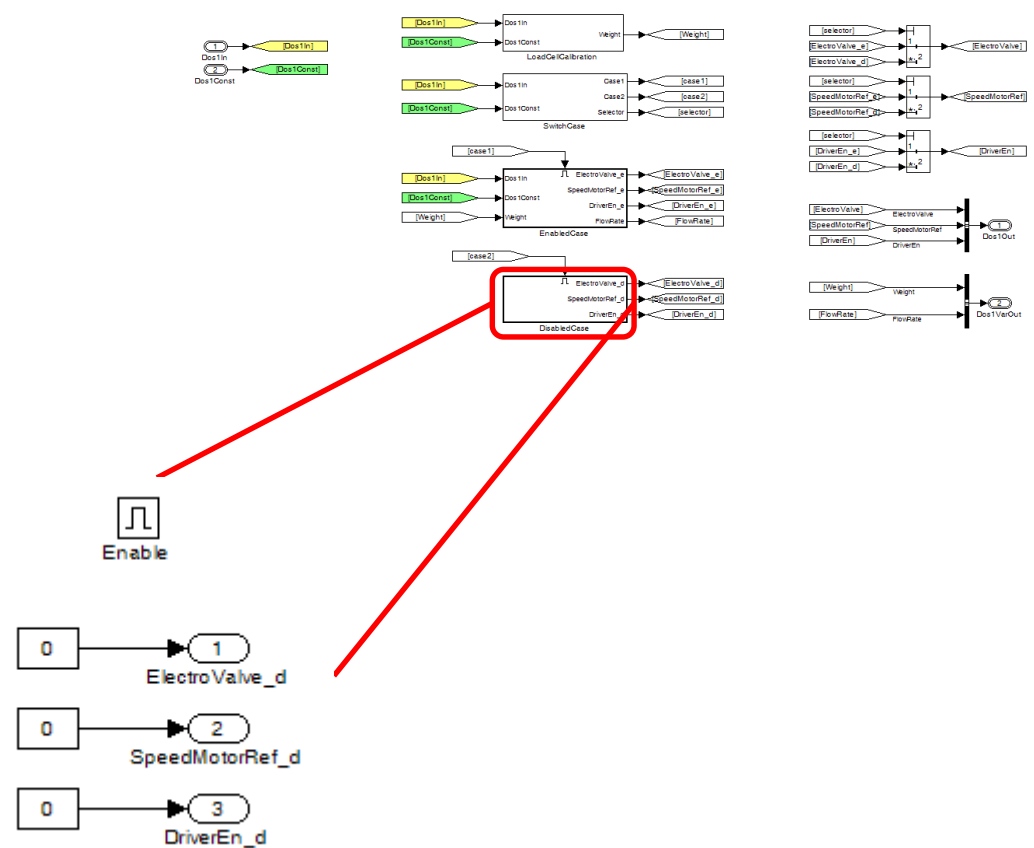
Abilitazione attiva



Esempio di generazione

Software di controllo

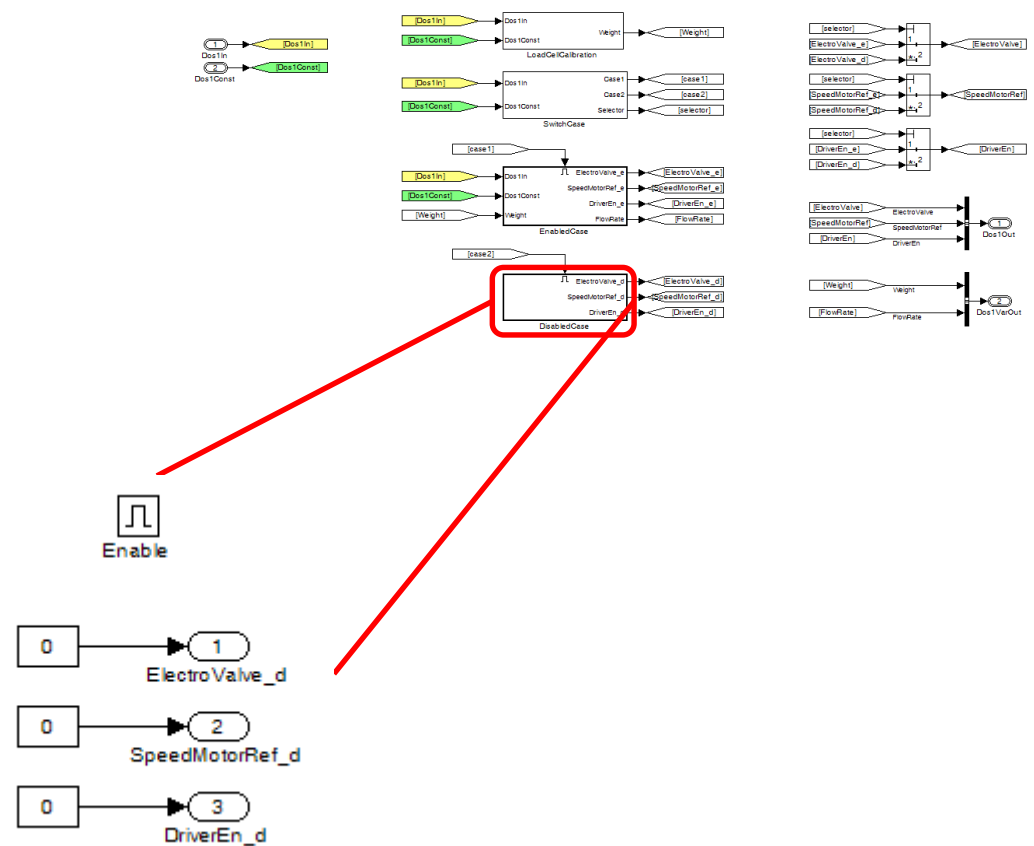
Abilitazione disattivata



Esempio di generazione

Software di controllo

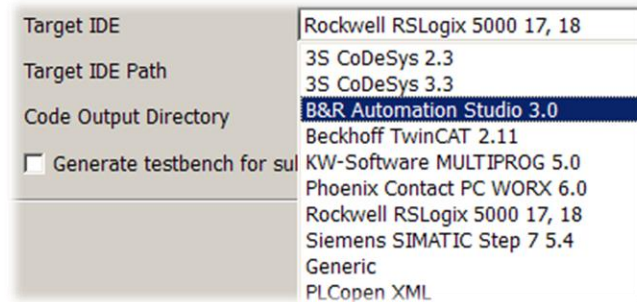
Abilitazione disattivata



Esempio di generazione

Software di controllo

La scelta del target per cui generare è da effettuare nel menu di configurazione del modello:



L'output della generazione è un Function Block, indicato con il nome del sottosistema Simulink (nel nostro caso FBControl).

A seconda del target selezionato PLC coder genera automaticamente i file necessari alla corretta importazione del blocco nell'ambiente di sviluppo.

Esempio di generazione

Una volta generato il Function block in ST è necessario predisporre l'ambiente di sviluppo in maniera tale da:

- Consentire l'esecuzione di una ciclica ad una frequenza prestabilita (da stabilire anche in precedenza alla generazione del codice)
- Consentire la lettura dei segnali provenienti dai sensori e la scrittura sulle uscite necessarie
- Consentire la scrittura di porzioni di codice in testo strutturato necessarie al riempimento delle strutture in ingresso al blocco funzionale

Esempio di generazione

L'esecuzione del codice nei diversi progetti risulterà la seguente:

- Nella fase di inizializzazione vengono riempite le strutture dati del blocco funzionale relative alle costanti
- Con cadenza prestabilita (nel nostro caso 100ms) viene richiamata la porzione di codice denominata «middleware»
- Il middleware dovrà riempire le strutture dati relative agli ingressi, eseguire uno step dell'algoritmo di controllo ed assegnare alle uscite del plc le uscite del blocco funzionale

Esempio di generazione

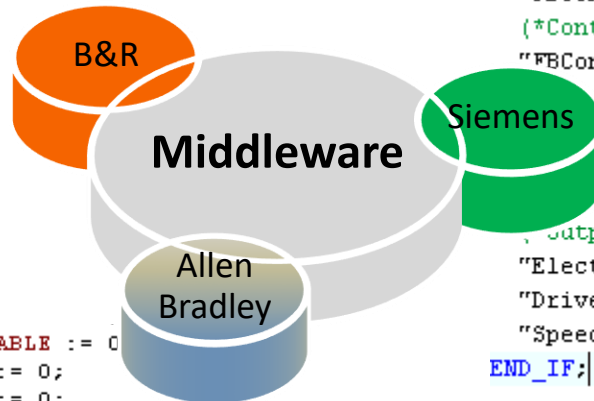
PROGRAM_INIT

```
(* I/O Configuration*)
StrainGaugesADCCConfig := 50]
(* Constants definition*)
Control.Dos1Const.ENABLE := TRUE;
Control.Dos1Const.DISABLE := FALSE;
Control.Dos1Const.TARE := 3.842847877777778E+006;
Control.Dos1Const.WEIGHT_TARE_SAMPLED := 4.500064986000000E+006;
Control.Dos1Const.SAMPLE_WEIGHT := 2.387;
Control.Dos1Const.MIN_WEIGHT := 2.0;
Control.Dos1Const.MAX_WEIGHT := 4.0;
Control.Dos1Const.Ki := 0.1;
Control.Dos1Const.Ti := 0.8;
Control.Dos1Const.Kp := Control.Dos1Const.Ki*Control.Dos1Const.Ti;
Control.Dos1Const.Kd := 0;
Control.Dos1Const.Ft := 0;
Control.Dos1Const.Umin := 0;
Control.Dos1Const.Umax := 100;
(*Initialization*)
Control();
Control.ssMethodType := 1;
END_PROGRAM
```

PROGRAM_CYCLIC

Control();
END_PROGRAM

```
DosConst.DISABLE := 0;
DosConst.Ft := 0;
DosConst.Kd := 0;
DosConst.Ki := 0.05;
DosConst.Kp := DosConst.Ki/0.8;
DosConst.MAX_WEIGHT := 4;
DosConst.MIN_WEIGHT := 2;
DosConst.SAMPLE_WEIGHT := 2.387;
DosConst.TARE := 4632;
DosConst.WEIGHT_TARE_SAMPLED := 5425;
DosConst.Umin := 0;
DosConst.Umax := 100;
FBControl(FBControllo, FBControllo.ssMethodType, DosIn, DosConst, DosOut, DosVarOut);
FBControllo.ssMethodType := 1;
ELSE
DosIn.ExtEmergency := Local:1:I.Data.0;
```



```
IF("FBControl_DB".ssMethodType = 1) THEN
(*Input variables reading*)
"FBControl_DB".Dos1In.ExtEmergency := "ExtEmergency";
"FBControl_DB".Dos1In.LockMotor := "LockMotor";
"FBControl_DB".Dos1In.WeightRaw := INT_TO_DINT("WeightRaw");
"FBControl_DB".Dos1In.ExtEmergency := "ExtEmergency";
"FBControl_DB".Dos1In.LockMotor := "LockMotor";
"FBControl_DB".Dos1In.FlowRateRef := "FlowRateRef";
"FBControl_DB".Dos1In.Start := "Start";
(*Control execution*)
"FBControl_DB"."FBControl" (ssMethodType:="FBControl_DB".ssMethodType,
Dos1In:="FBControl_DB".Dos1In,
Dos1Const:="FBControl_DB".Dos1Const,
Dos1Out=>"FBControl_DB".Dos1Out,
Dos1VarOut=>"FBControl_DB".Dos1VarOut);
(*Output variables assignment*)
"ElectroValve" := "FBControl_DB".Dos1Out.ElectroValve;
"DriverEn" := "FBControl_DB".Dos1Out.DriverEn;
"SpeedMotorRef" := INT_TO_WORD("FBControl_DB".Dos1Out.SpeedMotorRef);
END_IF;
```