# L’utilizzo dei controlli Real Time nell’Automotive

Flavio Corradini

CF3000 Engineering & Electronics

# L’evoluzione

- Ingegneria elettronica nel 1988:
  - Sono nati i microcontrollori a basso costo
  - L’elettronica digitale sta facendosi spazio sull’elettronica analogica
  - Arrivano i primi personal computers
    - Non esiste l’hard disk, si usa il DOS su floppy

# L’automotive nel 1988

- Il carburatore Weber domina il mondo
- L’accensione a transistor elimina i contatti striscianti
- L’iniezione Accensione Weber (IAW) e’ nata sulla Ferrari GTO
- La strumentazione e’ elettromeccanica

# Prime iniezioni elettroniche


- Programmazione software in Assembler
- Studio di nuovi controlli con Centro Ricerche Fiat
- Vetture di sperimentazione:
  - Lancia Delta Integrale, Ford Sierra Cosworth, Ferrari

# Direttive anti inquinamento

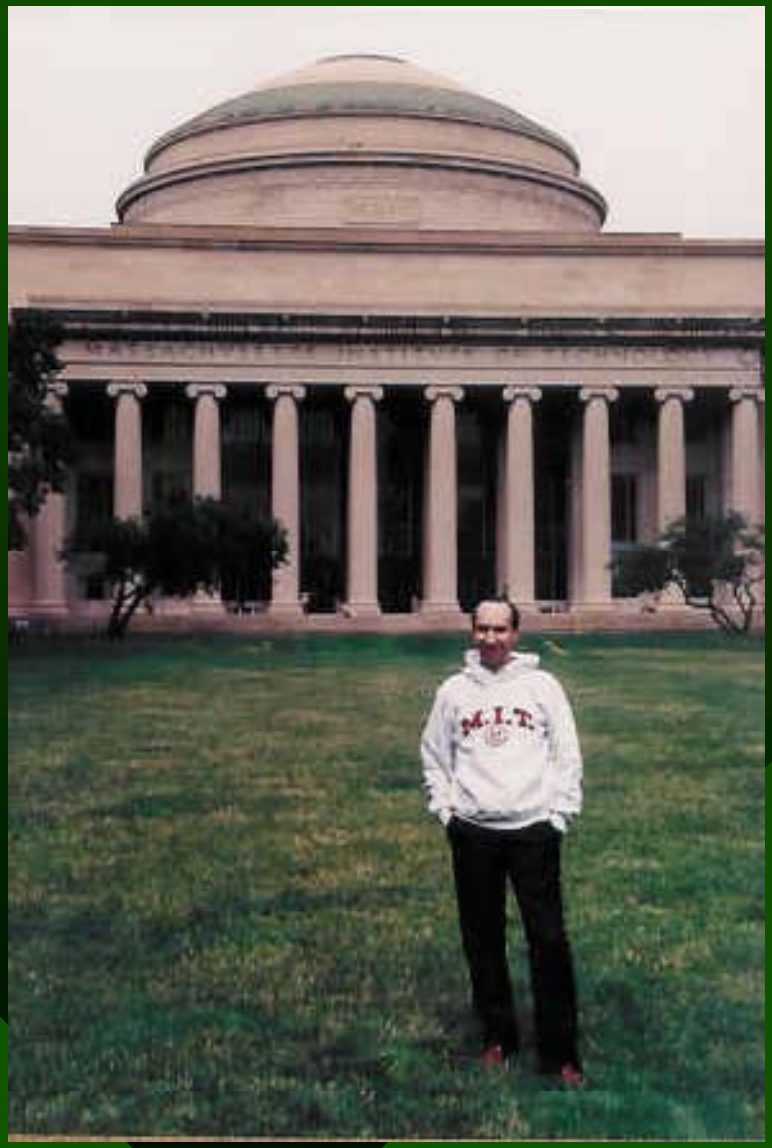
| DIRETTIVA | ARGOMENTO | DATA INTRODUZIONE |
|---|---|---|
| **91/441/CEE Euro 1** | Modifica la 70/220/CEE | 26/06/1991 |
| **93/59/CE** | Nuovi limiti emissioni | 01/10/1993 |
| **94/12/CE** | Nuovi limiti emissioni | 19/04/1994 |
| **96/44/CE Euro 2** | Modifica metodi rilevamento emissioni | 01/07/1996 |
| **96/69/CE** | Emissioni veicoli benzina, gasolio | 01/10/1996 |

# I controlli

- Tra i vari softwaristi, si distinguono quelli con metodo empirico e quelli che riescono ad applicare al motore endotermico le teorie di controllo
- Mentre gli ingegneri meccanici faticano a capire l’elettronica, gli ingegneri elettronici piu’ preparati non sfigurano nell’affrontare e risolvere elettronicamente problemi meccanici del motore

# La specializzazione

- Corso professionale post laurea sui controlli robusti al prestigioso MIT di Boston



# Al giorno d'oggi …

- Tutti i motori hanno controllo elettronico a microprocessore


- I nuovi requisiti di autodiagnosi in tempo reale spingono a verifiche sempre piu' sofisticate

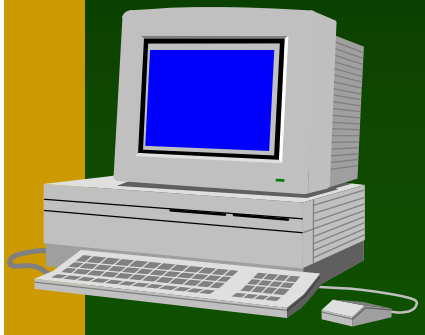
# Infomobilita’

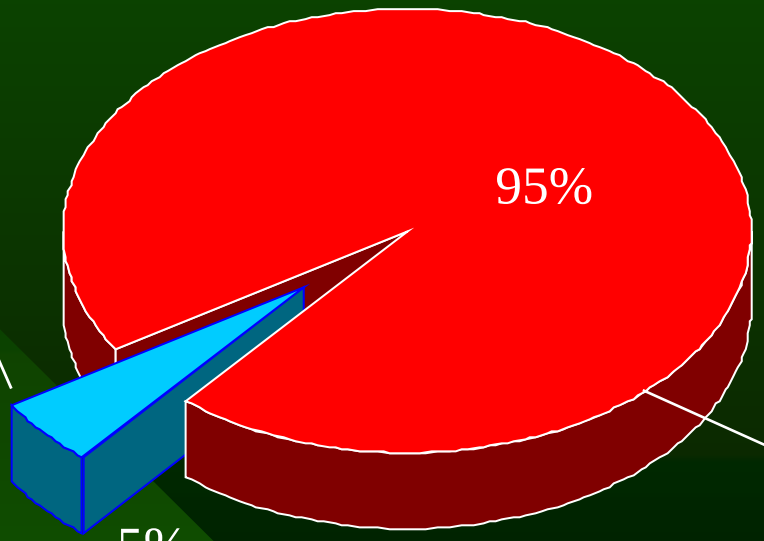
- Gestione GPS per Sistemi di navigazione

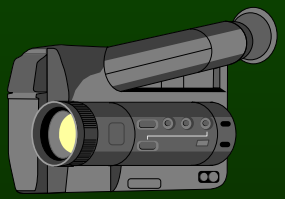
- Gestione informativa di bordo

# Ripartizione microprocessori / microcontrollori

**Computer Generici**
*(PC, workstations, mainframes)*





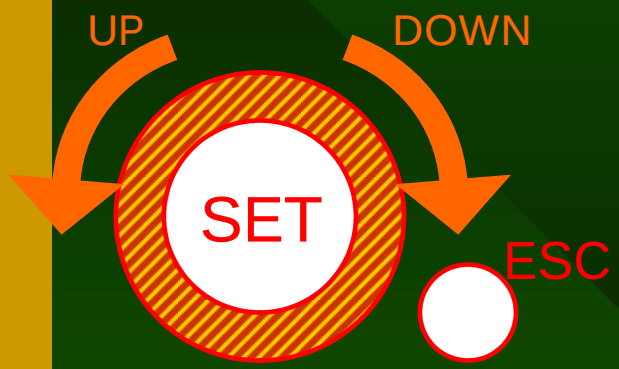
**Sistemi Embedded**
*(cellulari, telecamere, Veicoli, ecc.)*

# Criticita’ Real Time Automotive

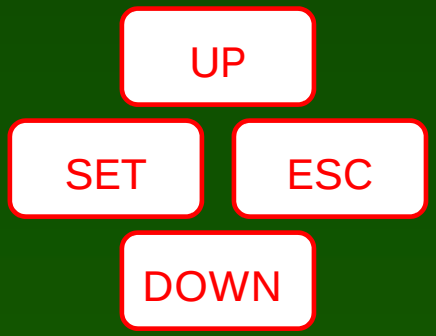
- Sistemi operativi real time
- Campionamenti
- Filtraggi

# Input dall'utente

L'interazione può avvenire tramite pulsanti o manopole.




Utilizzo combinato di manopola con 2 pulsanti (SET e ESC). La manopola consente di muoversi tra le voci disponibili mentre i 2 pulsanti permettono di selezionare o uscire dalla funzione.

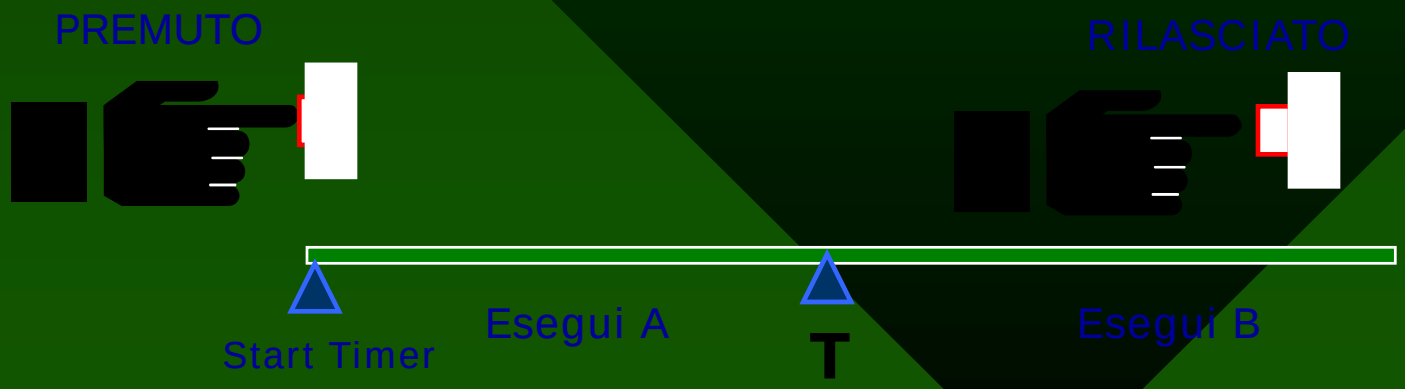



Utilizzo di 4 pulsanti, può rallentare la navigazione in liste lunghe.

# Input dall'utente

Gli eventi sui pulsanti possono essere associati alla pressione o al rilascio dei pulsanti stessi.

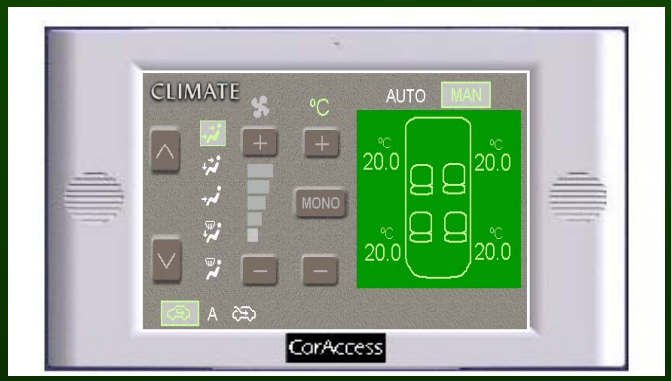
Nel secondo caso è possibile associare eventi differenti in base alla durata della pressione del pulsante.
(Es. pressione breve esegui A, pressione lunga esegui B).
In questo caso la pressione del pulsante abilita un timer.



# Input dall'utente

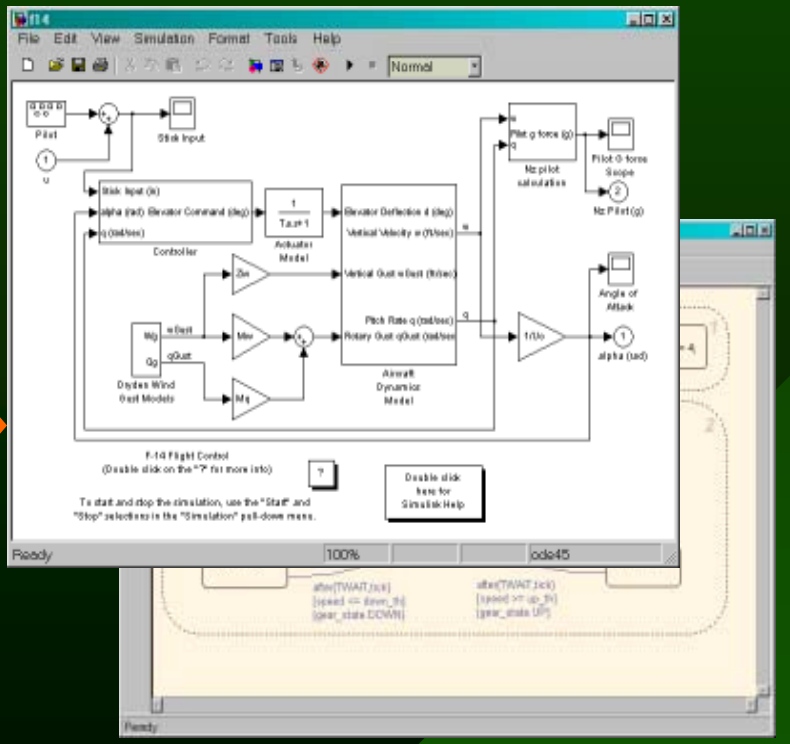
MECCANICO

TOUCH SCREEN

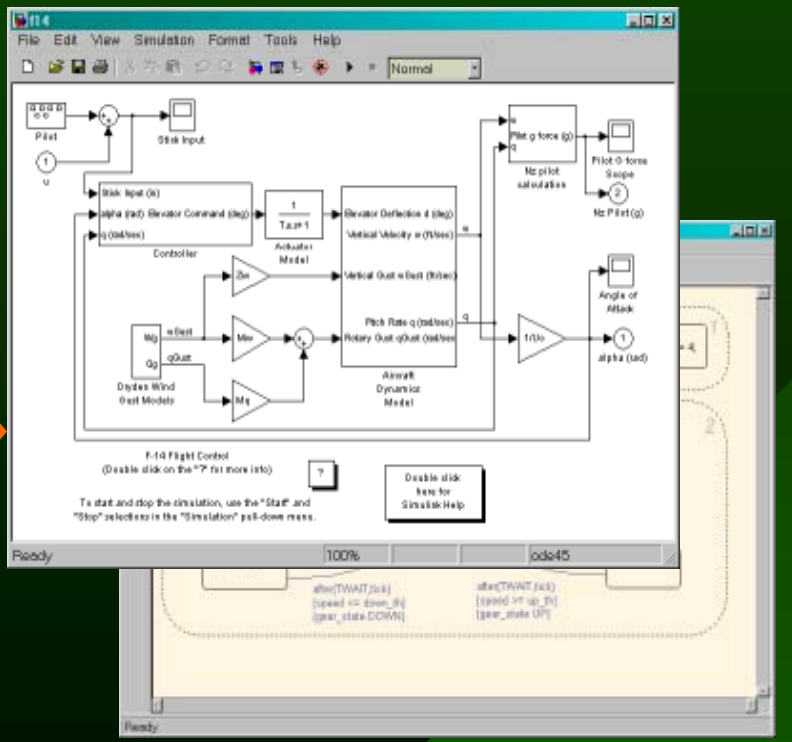



MISTO

In particolari occasioni visualizzare su grafico i segnali di interesse risulta meno chiaro rispetto alla visualizzazione diretta sulla strumentazione

La modifica di parametri del modello può avvenire direttamente attraverso l'interfaccia grafica che riproduce la strumentazione.

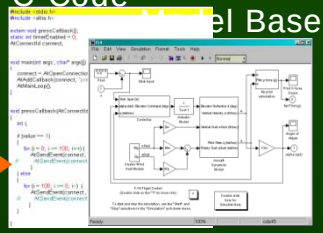
- Esempio di modello virtuale di cruscotto


Con particolari tools (es. Altia) è possibile animare rapidamente l’intera strumentazione

Con il partizionamento grafico è possibile testare funzionalità differenti.

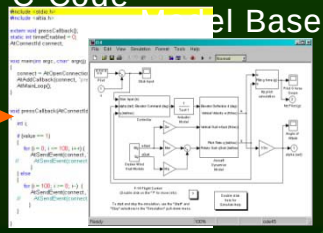



C-Code



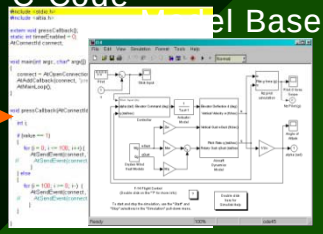
el Based

C-Code



el Based

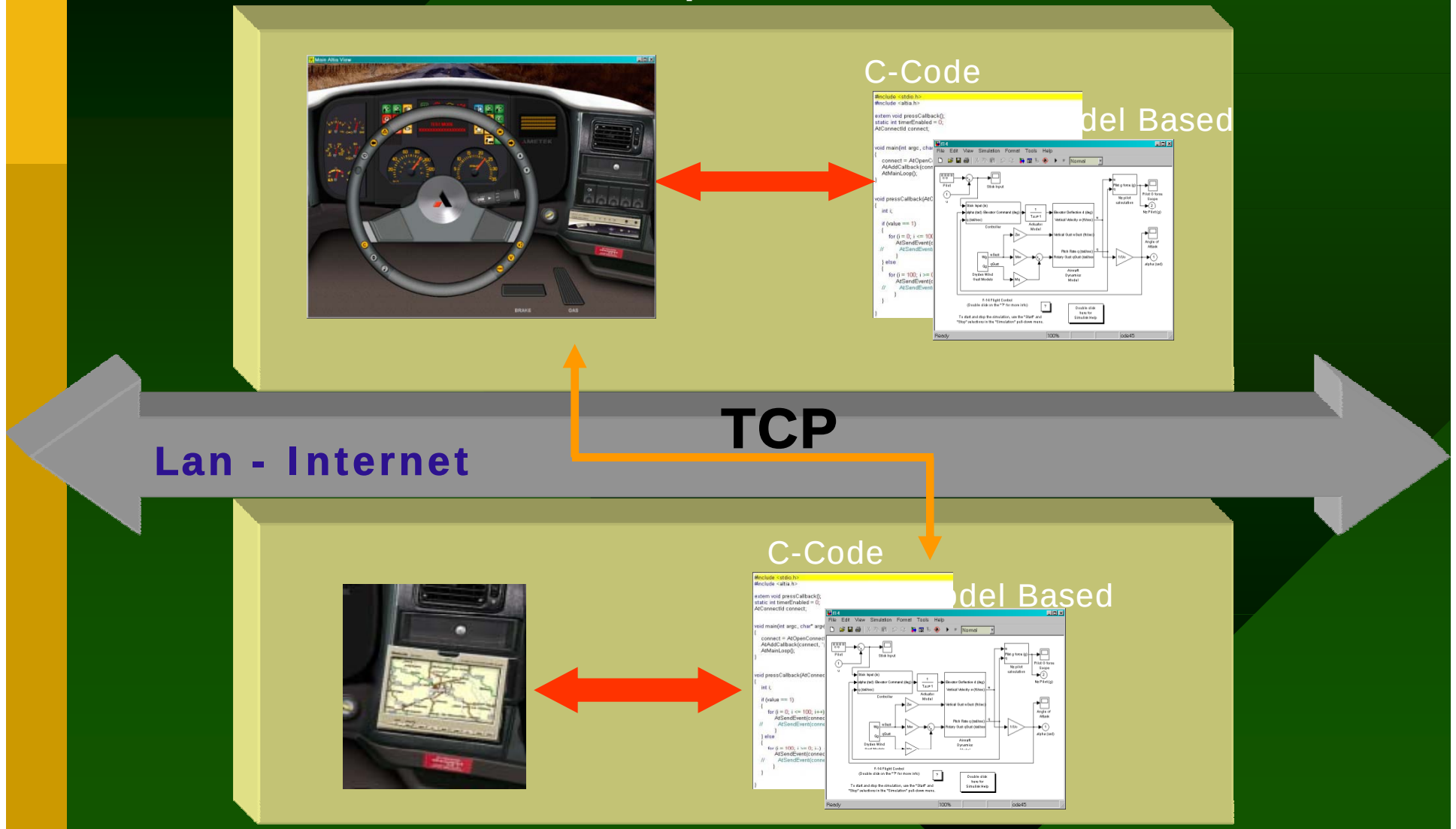
C-Code



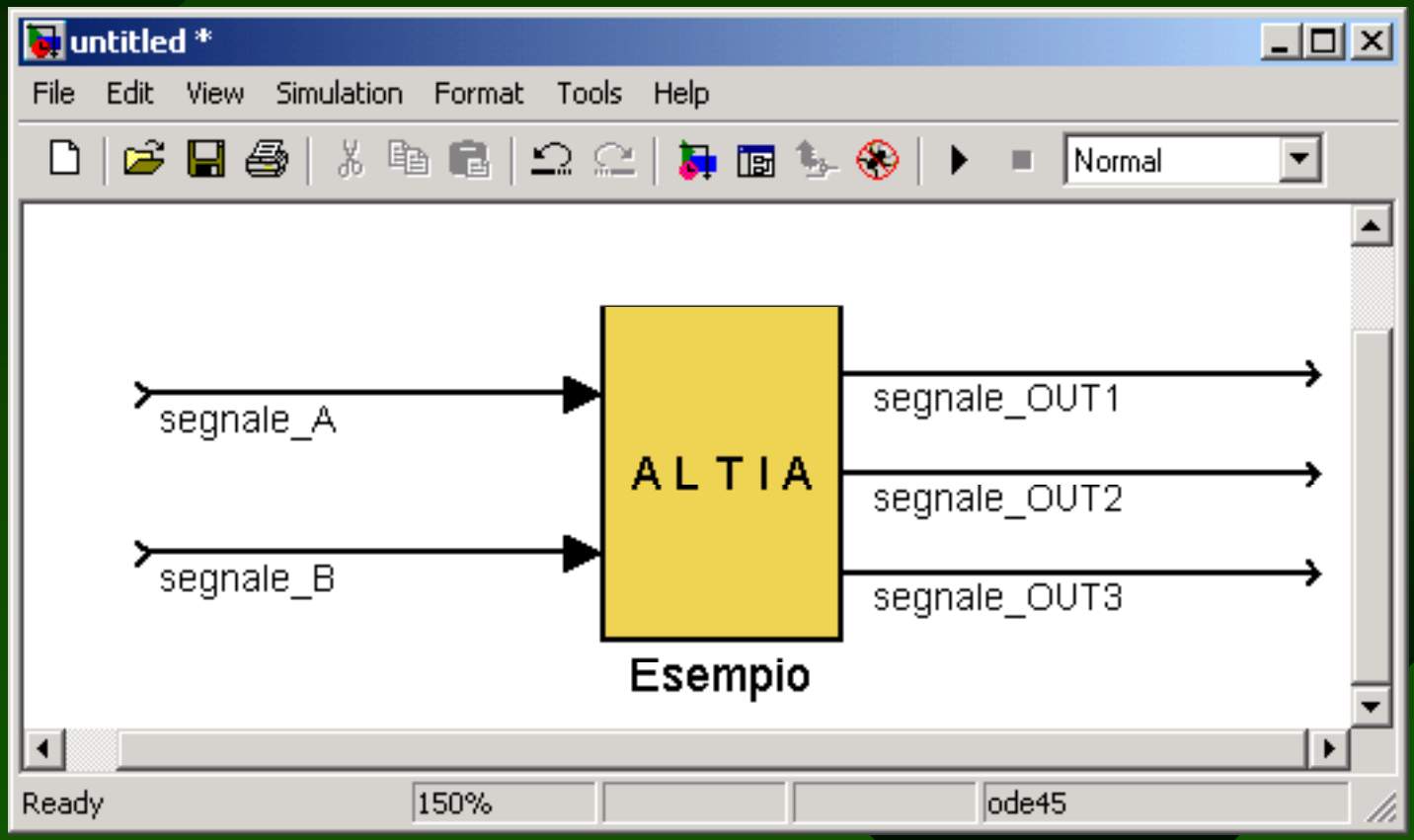
el Based

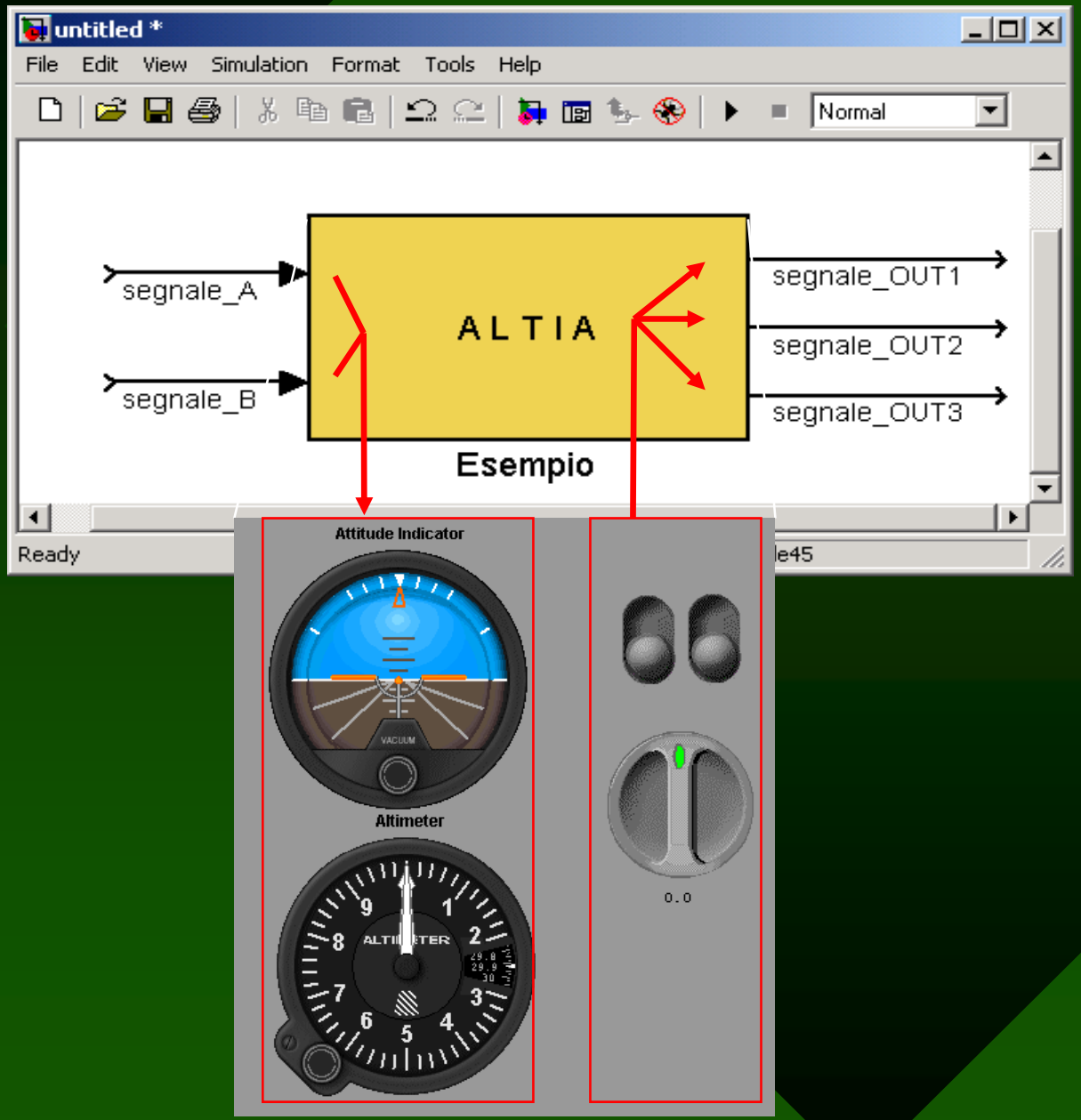
Ogni porzione di interfaccia è collegata a un modello (o codice)

- La comunicazione può avvenire via TCP

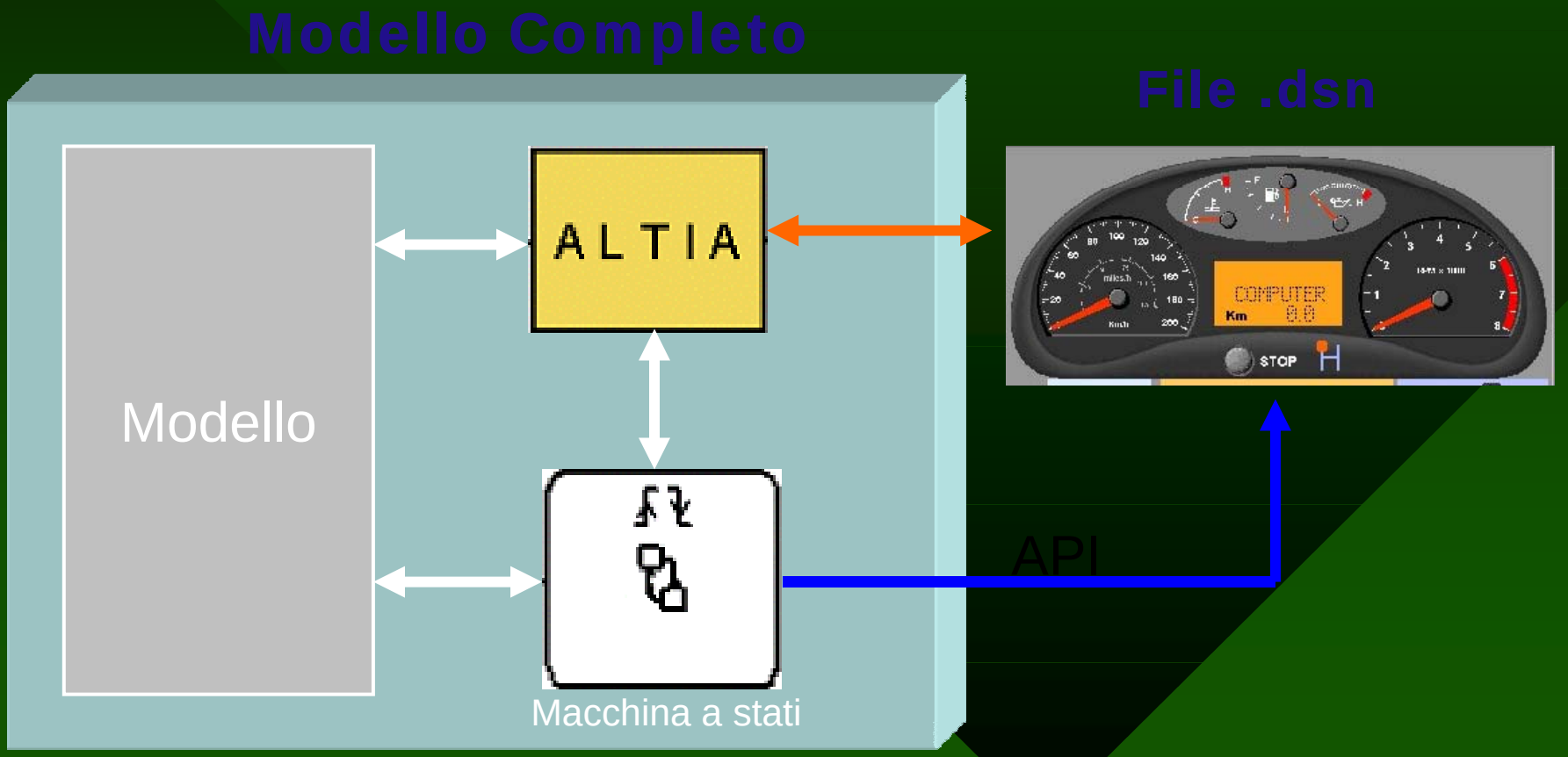


•La Connection permette ad una o più interfacce Altia di interagire con un modello Simulink

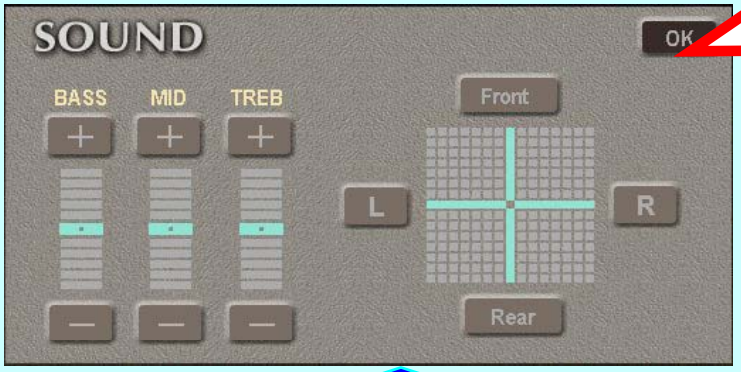



untitled *
File Edit View Simulation Format Tools Help
Normal
segnale_A
segnale_B
ALTIA
segnale_OUT1
segnale_OUT2
segnale_OUT3
Esempio
Ready
Attitude Indicator
Altimeter
0.0

•Nelle Chart di Stateflow è possibile fare chiamate alle API di Altia. Questa funzionalità è utile quando si intende visualizzare sull'interfaccia delle stringhe di testo.



## Applicazione




Altia
API

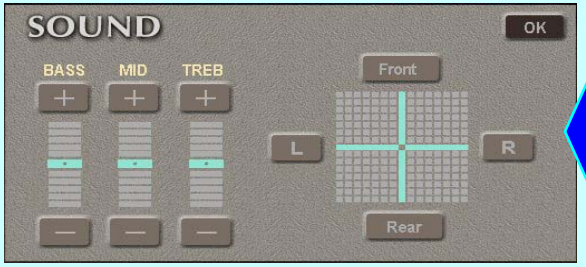
Logica
(logica di controllo
dell'applicazione)


Codice utente

# Integrazione con codice esistente

Sviluppo interfaccia

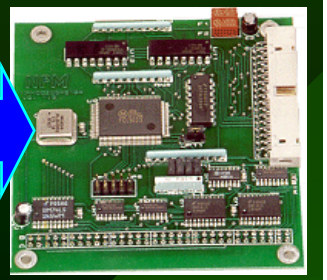
Sviluppo codice di controllo



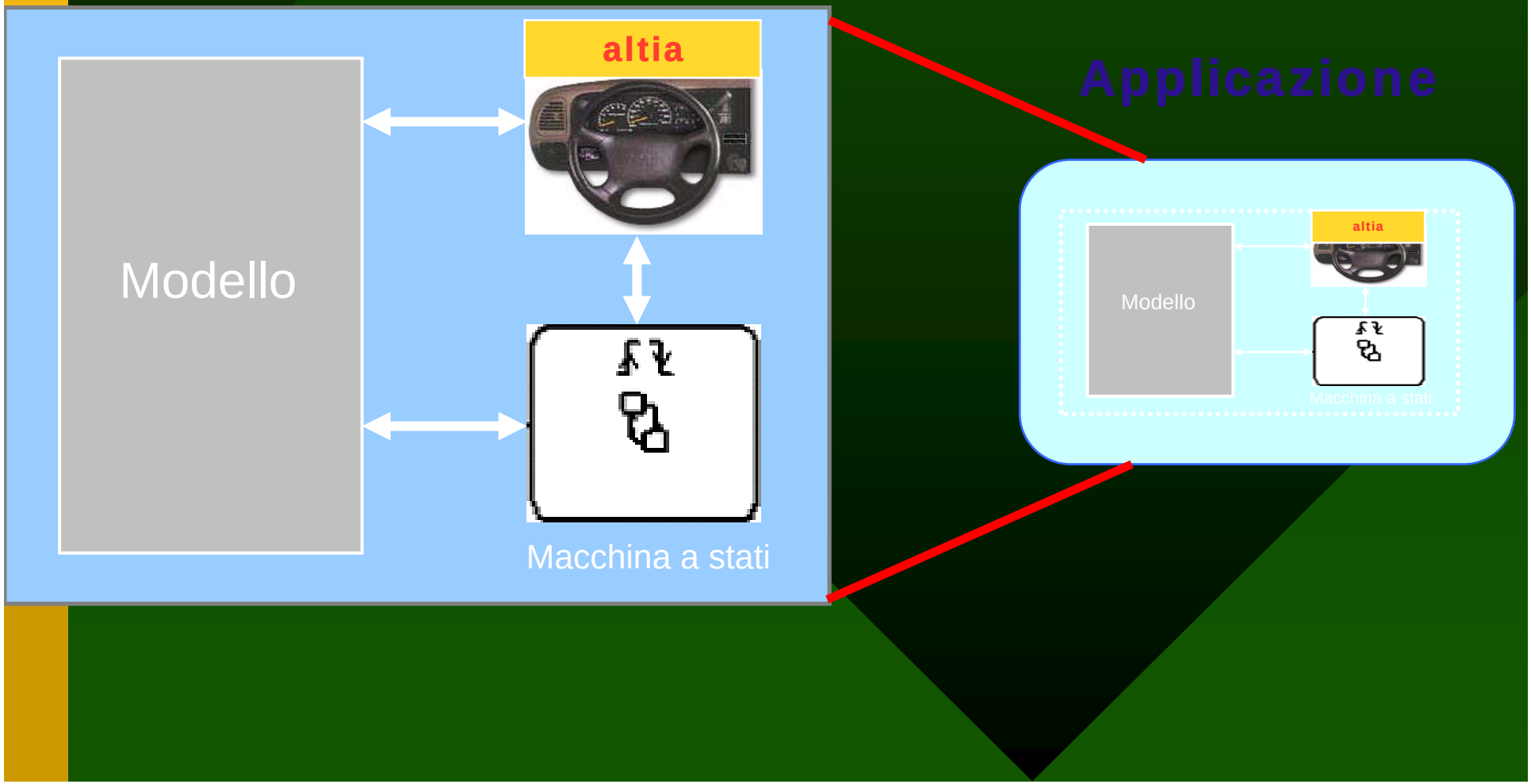

Codice utente

Altia API

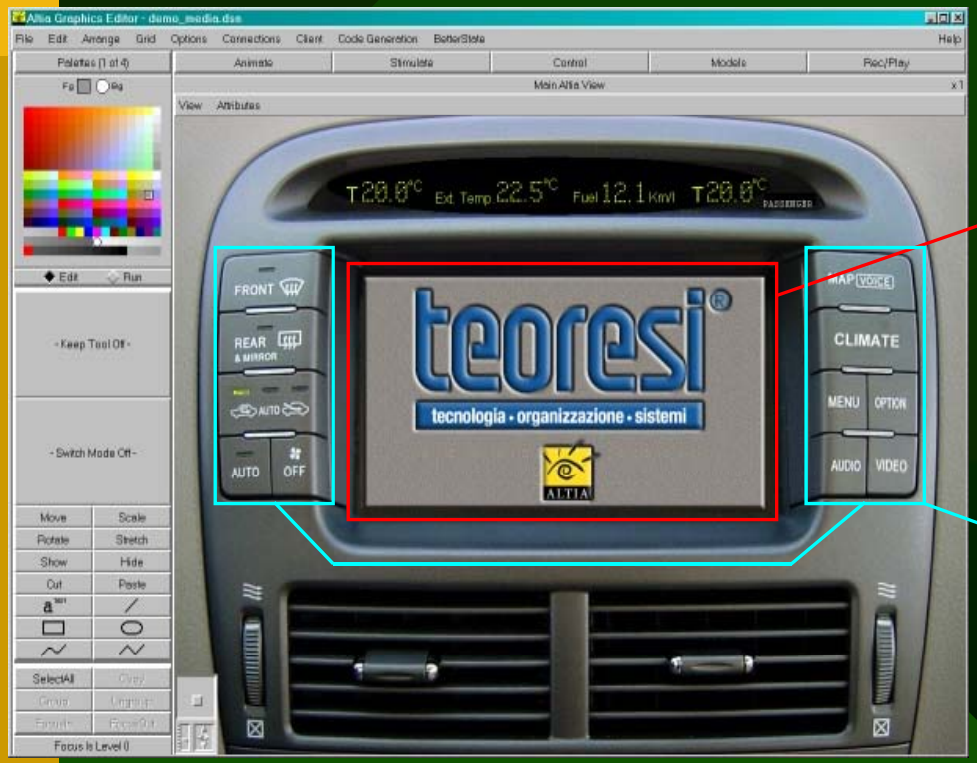
I / O API



E’ possibile integrare codice generato da altri tool di simulazione (es. RTW)



- **Simulazione dell’intero modello**




Applicazione

Codice Gestione Display

LOGICA

Codice Gestione Pulsanti Virtuali

```
int readButton()
{
  readVirtualButton();
}
```

- **Display Simulato - Cruscotto Reale**


Prototipo Reale


TCP

CAN

Applicazione

Codice Gestione Display

LOGICA

Codice Gestione Pulsanti Reali

```
int readButton()
{
  readRealButton();
}
```

# Display Simulato - Cruscotto Reale

# Rapid Prototyping

- **HW Real Time**

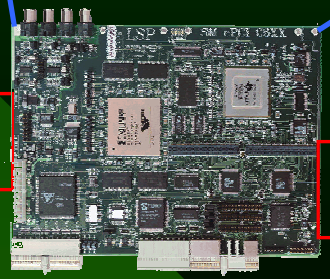
Codice Gestione Display

LOGICA

Codice Gestione Pulsanti Reali

```
int readButton()
{
   readRealButton();
}
```


HOST



TARGET RT

- **Generazione di codice con DeepScreen**



Applicazione

Codice Interfaccia

Codice Gestione Display

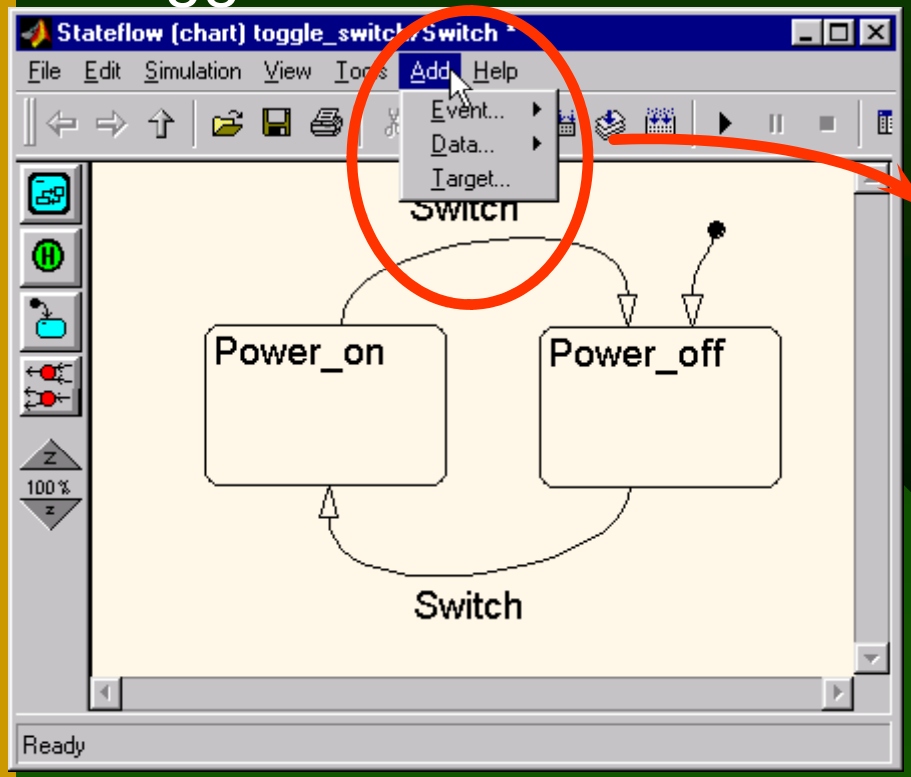
LOGICA

Codice Gestione Pulsanti Reali

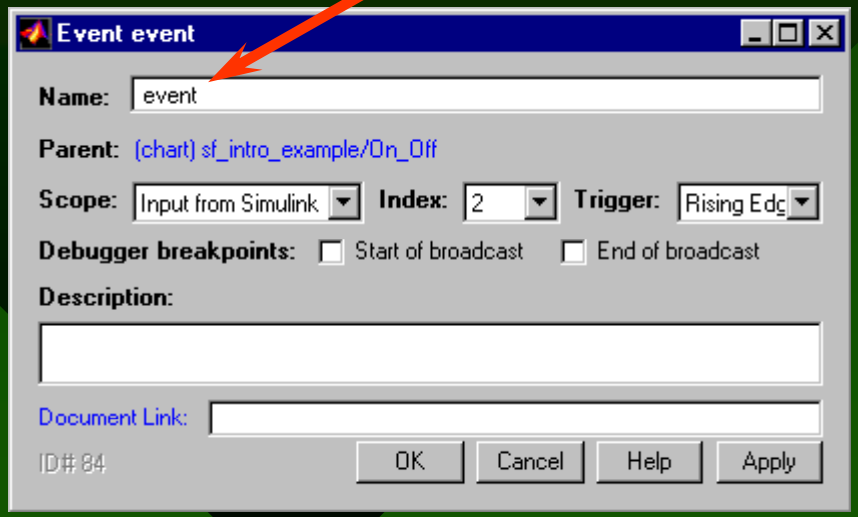
```
int readButton()
{
   readRealButton();
}
```

# Definire Dati ed Eventi con Simulink/Stateflow

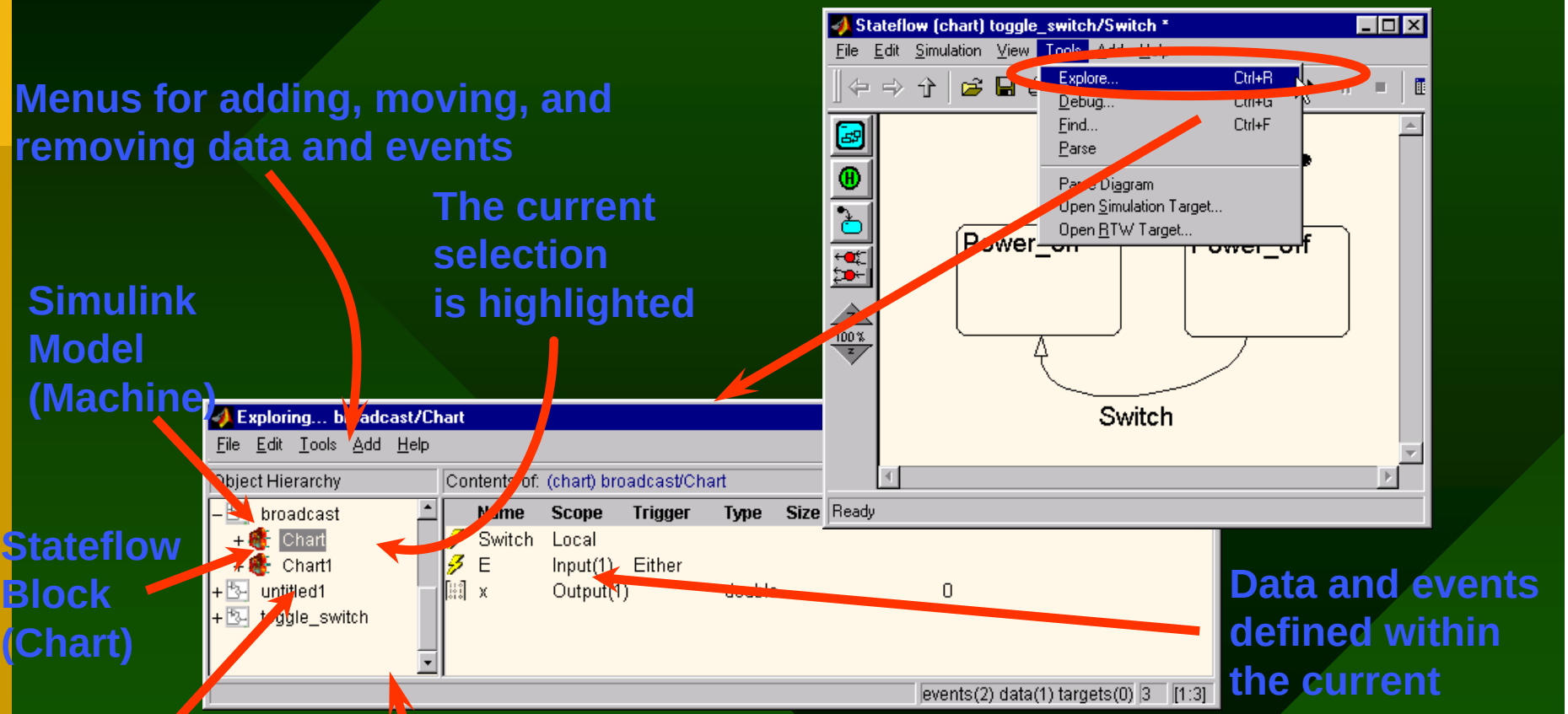
- Si devono definire i dati e gli eventi prima di verificare il diagramma Stateflow
- Usare il menu Add per definire un nuovo oggetto dato o evento




**Cambiare sempre il nome di default**



# Dati ed Eventi con l'Explorer



# Interfacciare Stateflow/Simulink

Data and Events servono come input e output tra Simulink e Stateflow

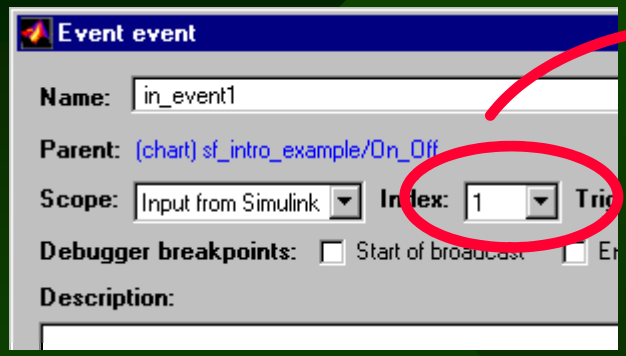
## Inputs

- **I Data inputs rappresentano valori numerici, il flusso di segnale di Simulink.**
- **Gli Event inputs corrispondono a triggers in Simulink. Le occorrenze degli eventi sono semplicemente I fronti per la simulazione**

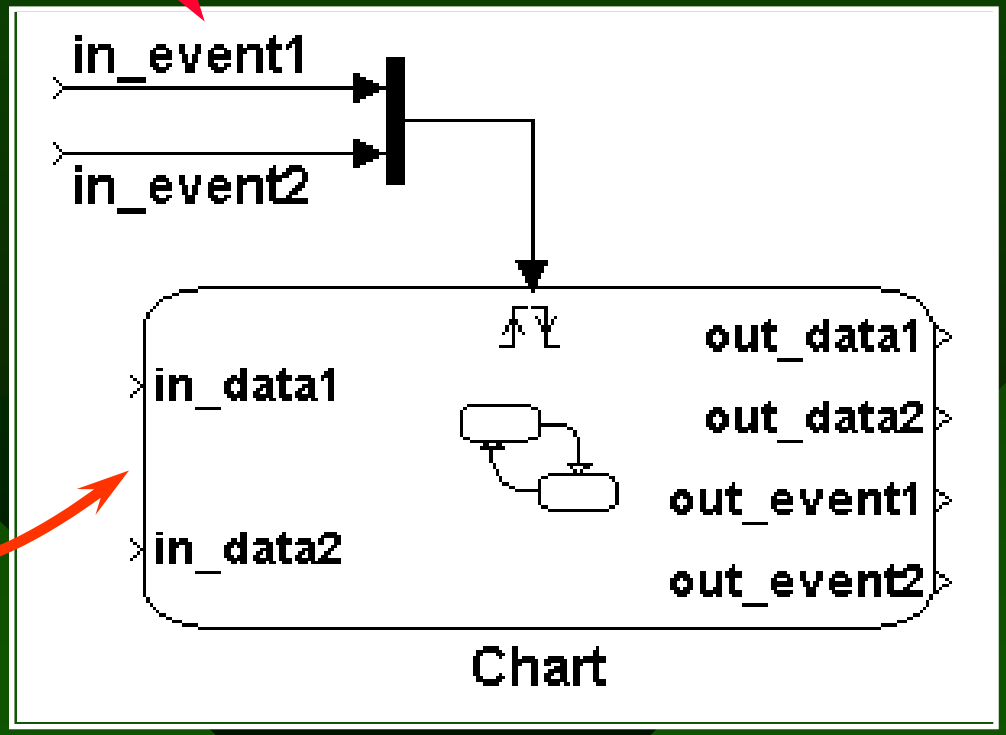
## Outputs

- **I Data outputs sono usati come segnali Simulink**
- **Gli Event outputs sono triggers usati in altri blocchi Stateflow e sottosistemi triggerati**

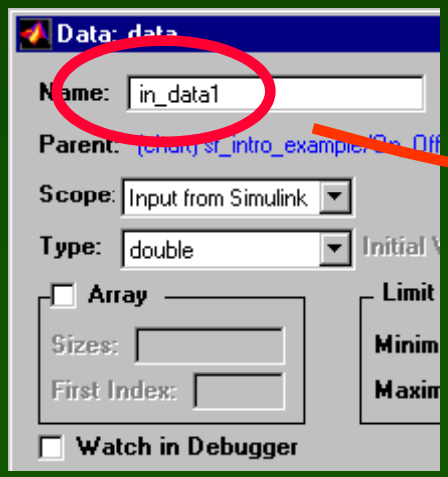
# Simulink Inputs



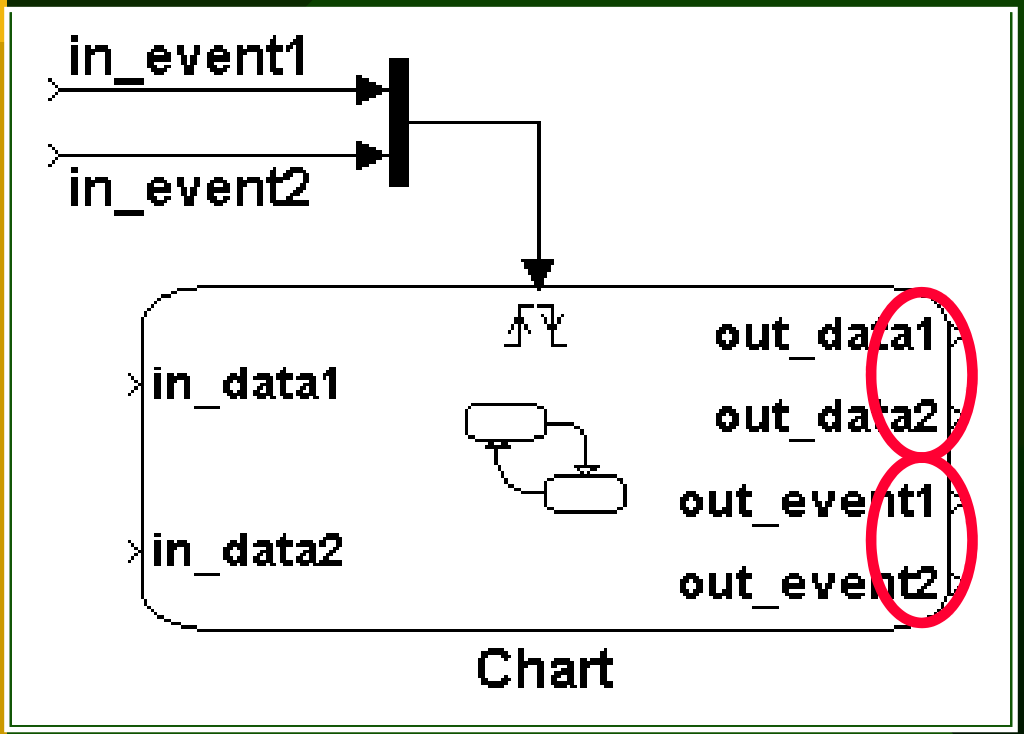

Events are grouped together as a vector and the index is specified

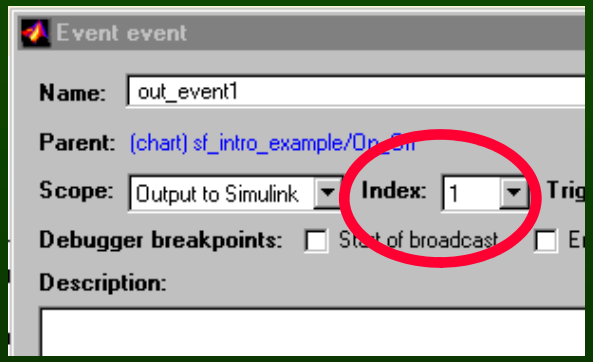



Data is separated into different inputs and the port is specified

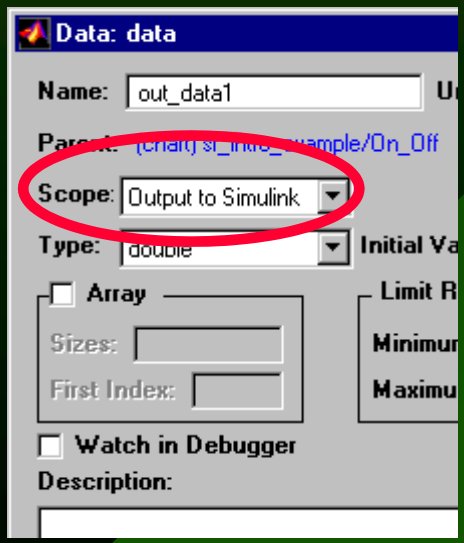


# Simulink Outputs



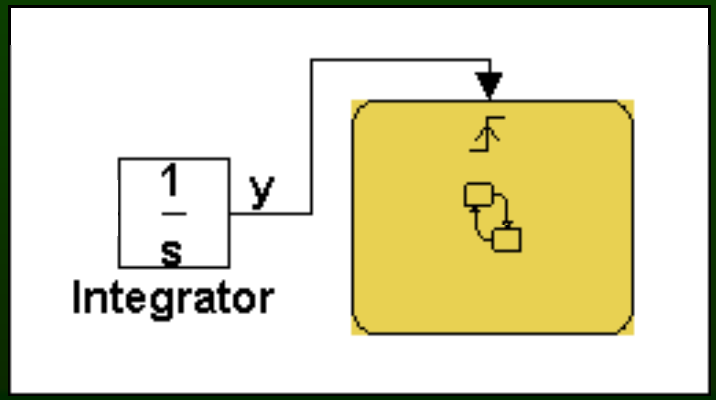




**Data and event outputs to Simulink originate as separate signals. You specify the port.**

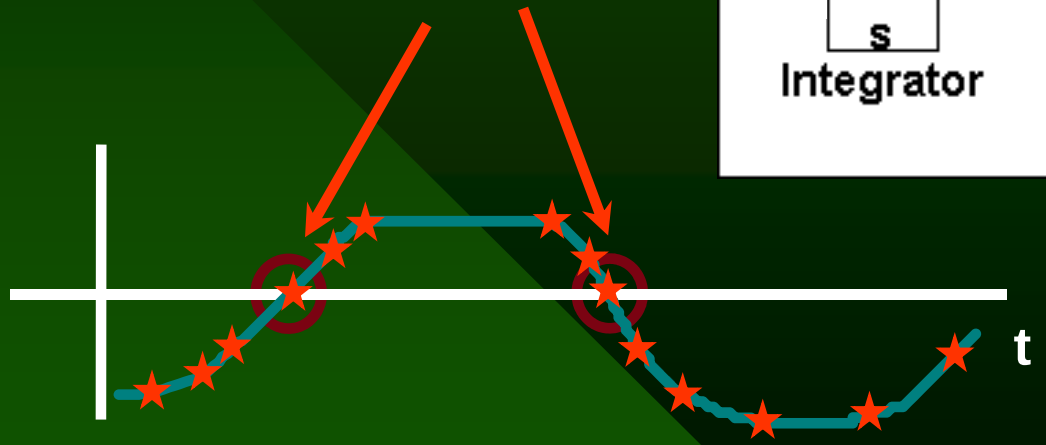


# Cosa significa Event Driven ?

La Stateflow chart viene eseguita solo quando il segnale di trigger va a zero

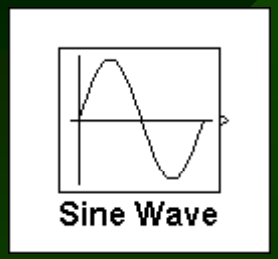



**At times of zero crossing the Stateflow block executes.**

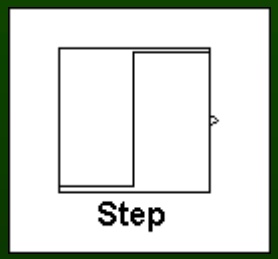



**The Stateflow block executes as a *result* of another signal.**

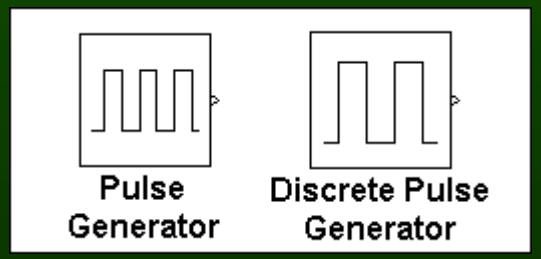
# Eventi in Simulink




Blocks with outputs that cross zero.

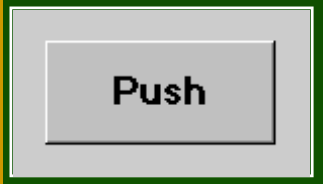


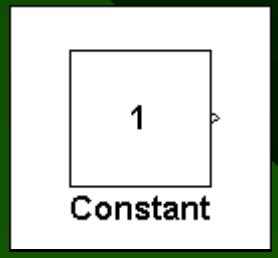

Step Functions



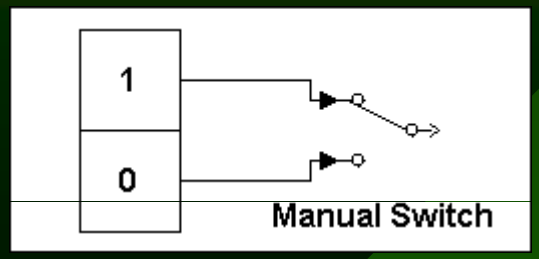

Pulse Generators

```
set_param('x/Constant','value','0')
```




HG Callback to a Constant Block




Manual switch between zero and one

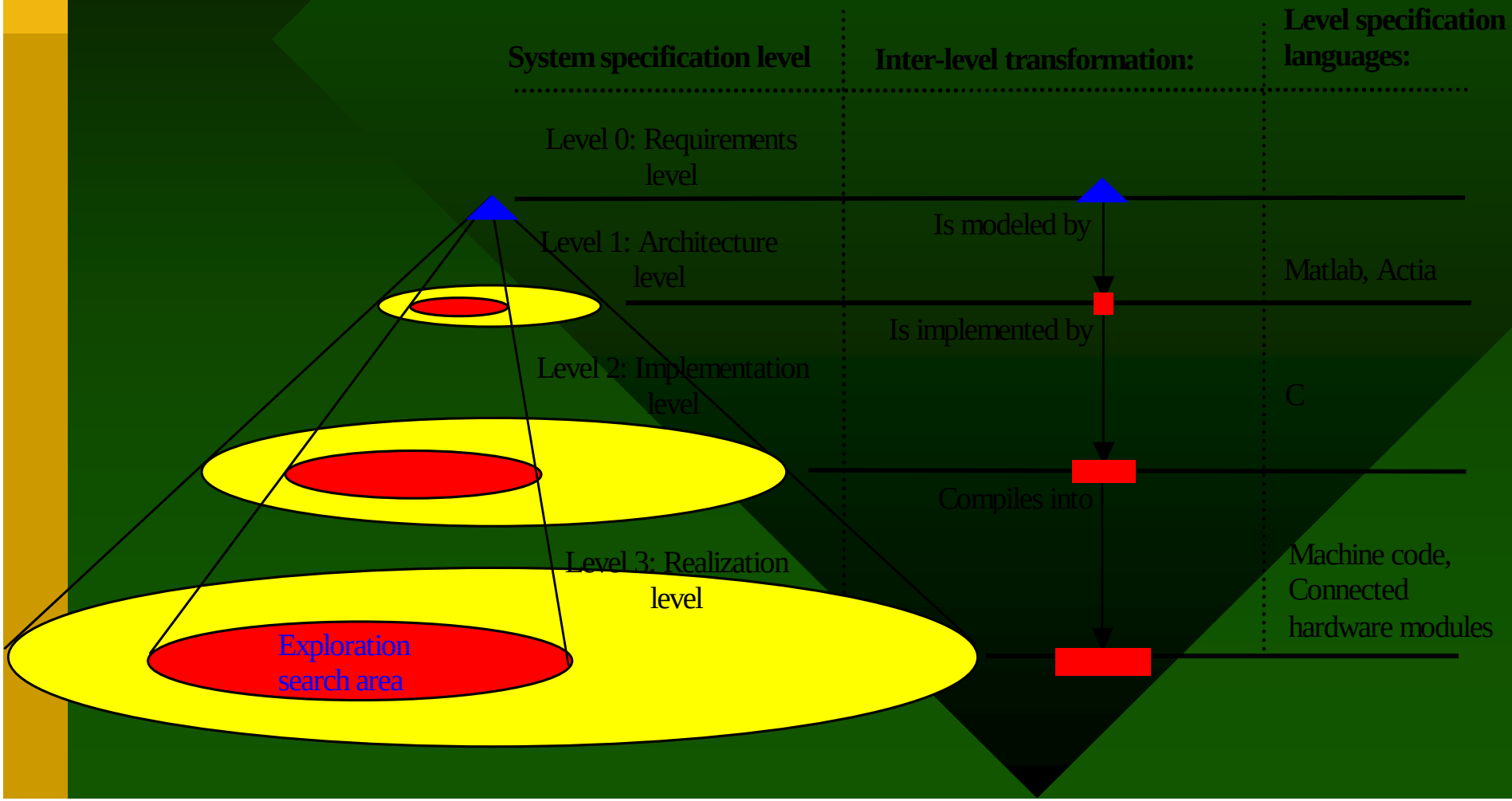
# Esempio MASTER e SLAVE

- Il MASTER (antislittamento) decide di attuare una riduzione di coppia motrice
- La centralina SLAVE (controllo motore) diminuisce la potenza erogata dal motore

# Arbitraggio

- Quando ci sono dei conflitti tra le centraline un arbitro deve prendere una decisione
- Le priorita’ di arbitraggio devono essere ben valutate a priori

# Progettazione a strati dei sistemi embedded



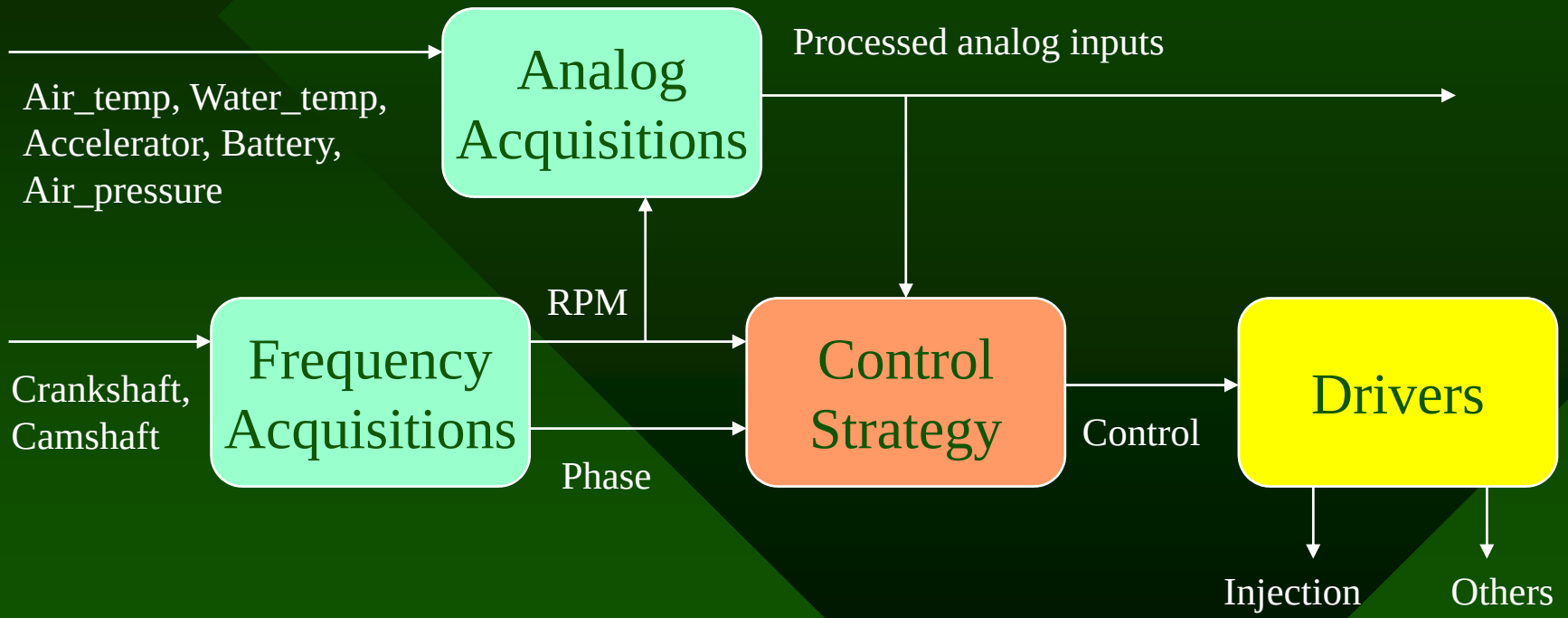
# Progetto di ECU controllo motore

- Specifiche della centralina di controllo motore
- Simulazione funzionale
- Partitioning e Mapping
- Ottimizzazioni

# Controllo motore

- Calcolare il tempo di iniezione ottimale
  - Minor consumo carburante
  - Minori emissioni inquinanti
  - Maggiori coppia e potenza
- Necessario trovare un trade-off tra requisiti in conflitto

# Controllo motore




- Tre fasi
  - Acquisizione, Controllo, Attuazione

# Specifiche della ECU

- Funzionalita’ principale di una ECU
- Tre blocchi gerarchici
  - Analog Acquisitions,
  - Frequency Acquisitions,
  - Control and Drivers.
- Matlab usato per tutti I moduli
- Simulink usato per simulazione, partitioning and mapping

# Specifiche per i sensori

- Tensione batteria: per determinare il comportamento di alcuni attuatori
- Debimetro, per calcolare il flusso d’aria entrante
- Sensore di posizione acceleratore
- Sensori di temperature aria e acqua
- Sensore di posizione con ruota fonica, che misura la rotazione del motore contando I denti che passano di fronte a un sensore induttivo
- Posizione albero a camme, usato alla ruota fonica dei giri motore per identificare la fase di ogni pistone

# Esempio specifica di un modulo

```
module Compute_T_Inj:
input RESET, VOLT_BATT: integer, Q_Fuel: integer;
output TIME_Inj: integer;
constant G_INJECT: integer;
function sb_itp_CKBATT(integer): integer;
var k_batt, q_fuel_curr, time_inj, volt_batt: integer in
loop
    k_batt := 0; q_fuel_curr := 0; time_inj := 0; volt_batt := 0;
    do
        loop
            await [VOLT_BATT or Q_FUEL];
            present VOLT_BATT then
                if (?VOLT_BATT<>volt_batt) then
                    volt_batt := ?VOLT_BATT;
                end if;
            end present;
            present Q_Fuel then
                k_batt := sb_itp_CKBATT(volt_batt); q_fuel_curr := ?Q_Fuel+(?Q_Fuel*k_batt)/256;
                time_inj := (G_INJECT*q_fuel_curr)/ 32; emit TIME_Inj(time_inj);
            end present;
        end loop;
    watching RESET;
end loop
end var
end module
```

# Simulazione funzionale

- Modello semplificato del motore, usando blocchi funzionali speciali
- La temporizzazione non e’ considerata
- Uscite e stati verificati usando display grafici e testuali
- Verifica raggiungimento comportamento corretto in tutte le condizioni operative

# Partitioning and Mapping

- Motorola 68332 a 16 MHz microprocessore scelto
- Prima, partitioning grossolano cambiando l’implementazone di intere unita’
- Poi, partitioning fine guardando I singoli moduli entro una unita’
- Considerata anche la politica di “schedulazione” per i componenti software

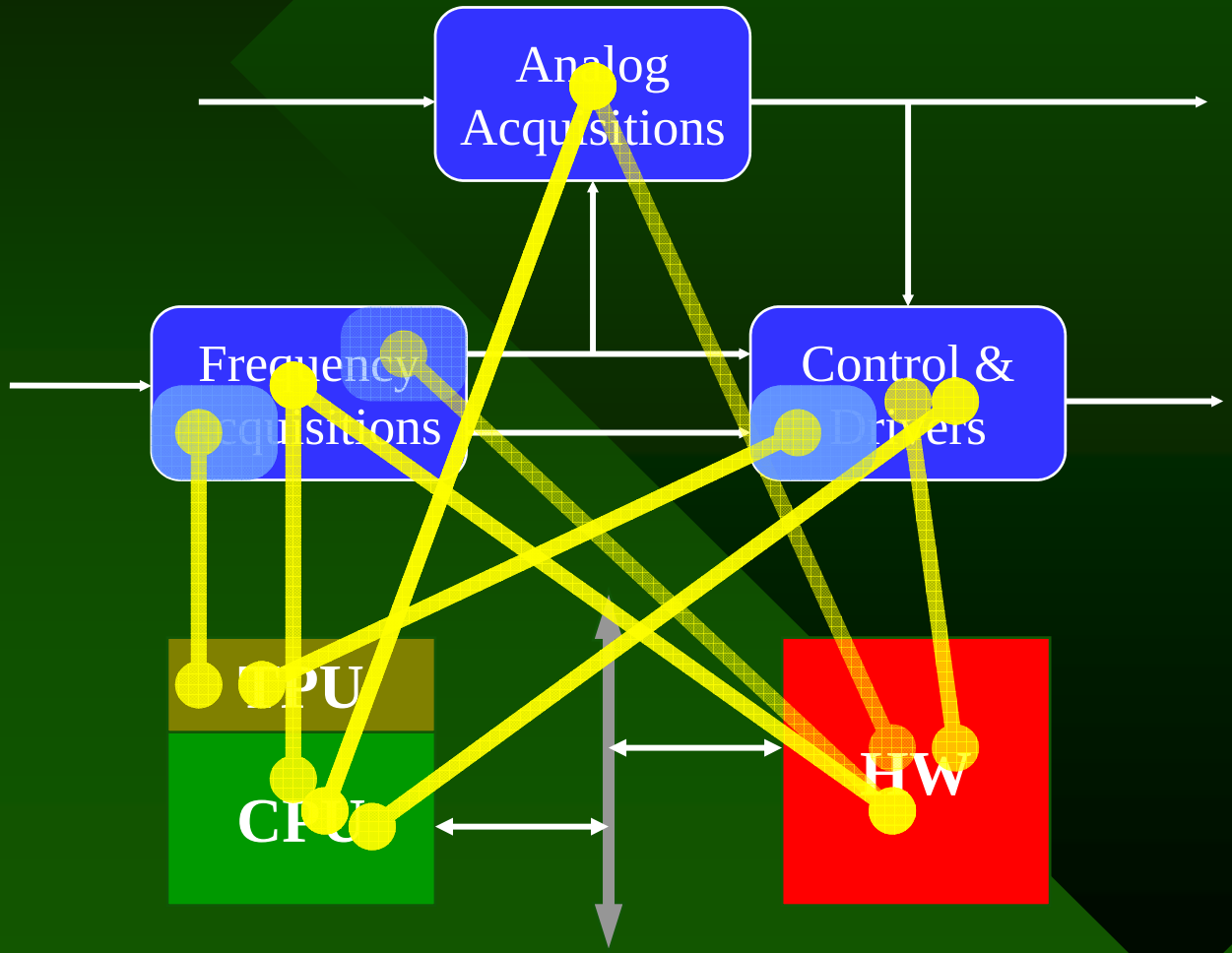
# Partitioning and Mapping

- La sintesi del software e la stima del tempo di esecuzione permettono una esplorazione rapida dello spazio di progetto
- Si possono simulare e stimare parecchie implementazioni differenti
- Gli obiettivi mancati identificano I comportamenti critici, di solito portando a risultati sbagliati
- Comportamento temporale assegnato alla CPU Timing Processing Unit (TPU)

# Partitioning and Mapping

- Partitioning iniziato con un’implementazione completamente hardware
- Gradualmente spostati blocchi verso il software
- Scoperto ad esempio che il collo di bottiglia sta nel modulo delle acquisizioni in frequenza
- Determinato che il problema risiede nella latenza elevata nel reagire agli eventi entranti
- Uno schedulatore con arbitraggio puo’ risolvere il problema

# Partitioning and mapping

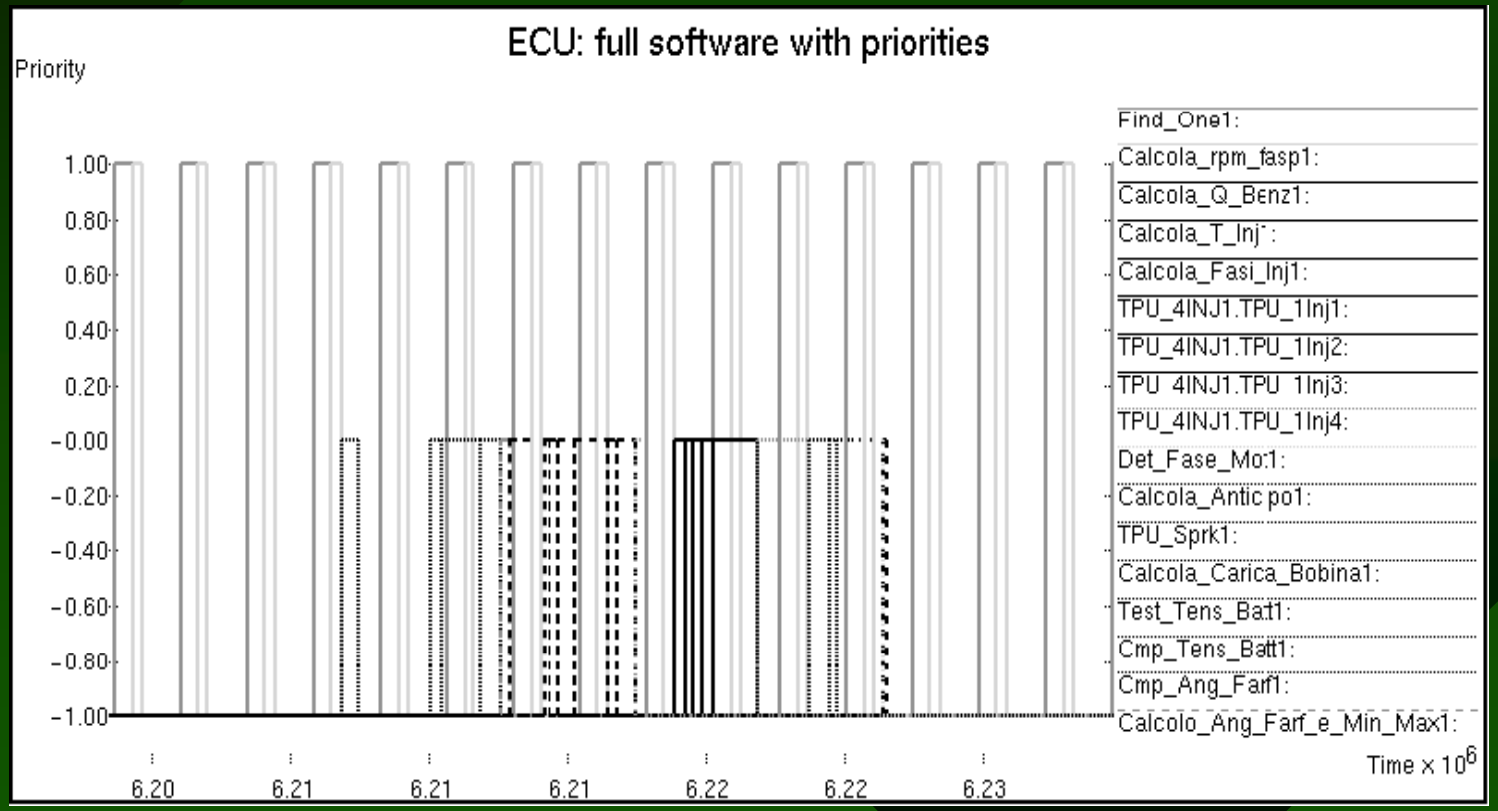



- Missed deadlines!
- Expensive scheduler
- PWM and

# Partitioning and Mapping

| Analog Acq | Freq Acq | Control Driver | TPU | Scheduler | Miss |
|---|---|---|---|---|---|
| HW | HW | HW | - | - | NO |
| SW | HW | HW | - | RoundRobin | NO |
| SW | HW | SW | Driver | RoundRobin | NO |
| SW | SW | SW | Driver Freq.Acq. | RoundRobin | YES |
| SW | SW/HW | SW | Driver Freq.Acq. | RoundRobin | YES |
| SW | SW (high priority) | SW | Driver Freq.Acq. | Interrupt | NO |

# Scheduling chart



# Ottimizzazioni

- La granularita’ del progetto ha un grosso impatto sulle prestazioni e sui costi
- I moduli vengono svolti in catena per redurre la sovrapposizione di scheduling e comunicazione
- Viene effettuato il merge dei moduli per ridurre la dimensione del codice
- I risultati sono comparabili a quelli di altre implementazioni sia come costi che come prestazioni, ma con minore tempo

# Simulatori RTHIL

- Realtime Hardware In The Loop

# Tipi di input/output

- Campionamento dei valori
- Filtraggi hardware e software dei segnali

# Discretizzazione segnali

- Il microcontrollore e’ una macchina che funziona a step
- I segnali che dobbiamo trattare sono continui
- E’ necessario quindi “discretizzarli”, creandone una rappresentazione sufficientemente veritiera

# Campionamento

- Il microcontrollore raccoglie quindi in certi istanti dei “campioni” del segnale da misurare
- La sequenza dei campioni viene memorizzata per l’opportuna ricostruzione del segnale desiderato

# Frequenza e fasatura di campionamento

- La frequenza di campionamento e' determinante per il segnale
- Anche la fasatura del campionamento pero' e' molto importante per acquisire un valore significativo

# Fasatura

- La fasatura del campionamento puo’ essere fatta su base tempo o su base evento
- Un esempio della fasatura su base evento e’ la fasatura su base angolare rispetto all’albero motore

# Sperimentazione

- La fasatura corretta dipende dal fenomeno che dobbiamo gestire
- Spesso sono necessarie parecchie prove sperimentali per determinare la fasatura migliore

# Filtraggio dei segnali

- Un segnale e’ sempre associato a disturbi estranei al suo significato primario
- Per eliminare i disturbi e’ assolutamente necessario un “filtraggio”

# Filtraggio sofware

- Operazioni simili a quelle effettuate con l’hardware possono essere effettuate con il software, ma con una metodologia completamente diversa

- Il sofware puo’ effettuare operazioni di filtraggio molto piu’ complesse, associando ad esempio l’eliminazione dei disturbi alla linearizzazione delle curve dei sensori, ecc.

# Matematica

- Con i filtraggi software si possono ottenere anche operazioni complesse, come filtri del secondo o terzo ordine
- Si puo’ anche inserire una diagnosi dipendente da una macchina a stati di validazione del disturbo

# Contatto

**fcorradini@cf3000.it**

**www.cf3000.it**

**Tel. +390522361134**

**Fax +390522360803**

*CF3000 Engineering&Electronics*

*Via Tonino Gualtieri, 1*

*42100 Reggio Emilia*

*Italy*