Real-time control systems: an application to fusion experimental devices

Università degli Studi di Salerno, April 21, 2016

G. De Tommasi[1]
[1]DIETI, Università di Napoli Federico II

Outline

Real-time systems

Real-time for control

Fusion and Tokamaks

Magnetic control in a tokamak device

A real-time framework for control systems in fusion devices

JETRT

From JETRT to MARTe

What is a real-time system?

- A real-time system is a system (hardware+software) subject to “real-time constraints”.
- In a real-time system, the result of a computation is correct if
  - is correct (!)...
  - ... AND meets specified time constraints – the so called “deadlines”

Non-real-time VS real-time - 1/2

Example of non-real-time algorithm

- **Functional requirement:** Given the two weights `w1` and `w2`, compute the weighted sum of the two inputs `u1` and `u2`

```
double weightedSum(double u1, double u2, double w1, double w2){
            double result = w1*u1+w2*u2;
            return result;
}
```

Non-real-time VS real-time - 2/2

Example of Real-time algorithm

- **Functional requirement:** Given the two weights w1 and w2, compute the weighted sum of the two inputs u1 and u2
- **Non-functional requirement:** perform the computation in **at most** 1 ms

Now writing...

```
double weightedSum(double u1, double u2, double w1, double w2){
            double result = w1*u1+w2*u2;
            return result;
}
```

...is no more sufficient to fulfill the requirements!

We should exploit (indirectly) the hardware architecture and (directly) the operating system, in order to meet the time constraint

Time constraints 1/2

- A computation must be performed **every X time units**
  - is a *periodic* activity (task), and the time constraint must be met with a given accuracy (*jitter*)

Examples

- "the control action to be applied by the aerosurfaces of an aircraft must be computed every 5 ms"
- "System A must send a message to system B every 10 s"
  - **Remember: real-time does not necessarily means "fast"!**

Time constraints 2/2

- A computation must be completed **within Y time units after its triggering**
  - is a task with a *deadline* (*cyclic* or *event-based*)

Examples

- “the cyclic execution of a PLC must terminates within 200 ms”
- “stop the cruise control within 50 ms after the break press”

- **Note: usually a periodic task should also meet a deadline**

Hard and soft real-time

- **Hard** real-time systems
  - **Missing (even a single) deadlines means system failure (!)**
- **Safety critical** systems
  - Missing deadlines can cause serious loss
- **Soft** real-time systems
  - Deadlines may be missed and mainly cause a deterioration of the QoS

- Real world (real-time) systems have a mix of hard/soft components
- The distinction between hard and soft real-time is somewhat subjective
- **Soft real-time is not Non-real-time (!)**

Design real-time systems

Assess schedulability

- Given *n* real-time tasks...
- ...given the correspondent time constraints (deadlines)...
- ...given the hardware (and software) architecture...
  - is it possible to meet all the timing requirements, i.e. is it possible to schedule the tasks?
    - Are the deadlines met for all the cyclic and event-driven tasks?

      `End_time(task_k)-Start_time(task_k)` $\leq$ `Deadline(task_n)`
    - Are the periodic tasks executed with the required accuracy? Do they meet their deadlines?
- There exist formal methods that permits to assess schedulability (under given assumptions)

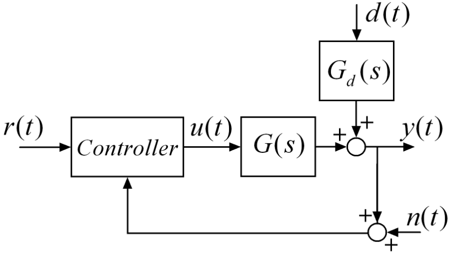
Real-time operating systems (RTOS)

- Interrupts/Polling
- Multitasking (concurrency)
- Timer support
- Static scheduling/Preemptive scheduling (priorities)
- Task Segregation
- ...

Some RTOS

- WindRiver VxWorks
- QNX Neutrino
- RTAI (Linux patch)
- FreeRTOS
- Windows CE

Example - Continuous control system 1/2



The plant

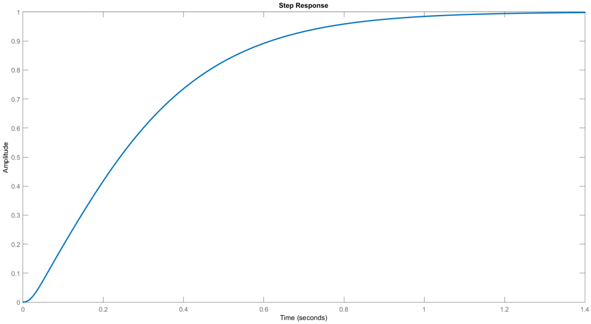
$$G(s) = \frac{2.5 \cdot 10^5}{(s+10)(s^2+80s+2500)}$$

Example - Continuous control system 2/2

The continuous-time controller

$$C(s) = \frac{2.24(s+25)^2}{s(s+200)} \tag{1}$$

Open-loop step response



Example - Controller discretization

The discrete-time controller

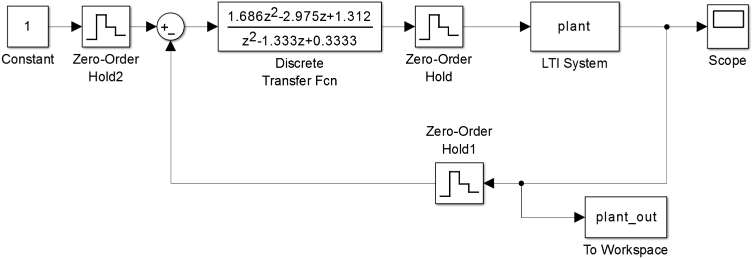
Given the sampling frequency $f_s = 200$ Hz, the Tustin approximation of the controller (1) is

$$\hat{C}_d(z) = \frac{1.686(z-0.882)^2}{(z-1)(z-1/3)} \tag{2}$$

- Implementing the discrete-control law (2) means
  - **Functional requirement:** to write a task that computes the correspondent difference equation
  - **Non-functional requirement:** to execute the task every 5 ms (assuming negligible execution time)

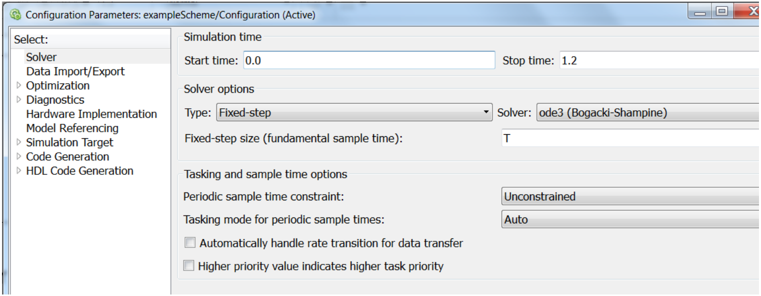
To meet or not to meet (the deadlines)?

Use Simulink...



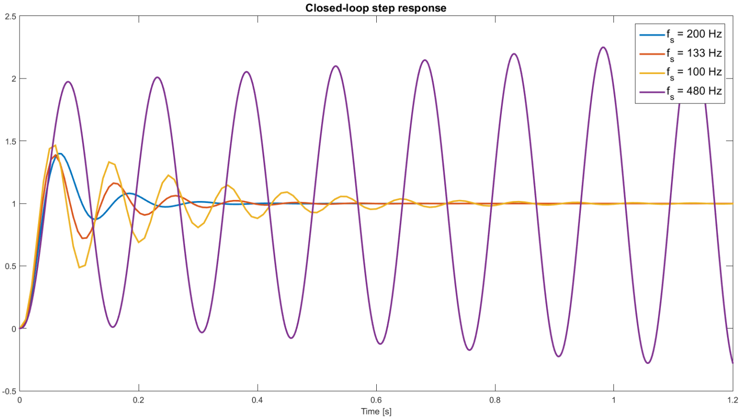
To meet or not to meet (the deadlines)?

...with Fixed time-step solver...



To meet or not to meet (the deadlines)?

. . .changing the time step

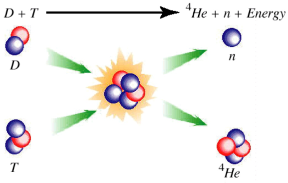


Nuclear Fusion

Main Aim

Production of energy by means of a fusion reaction

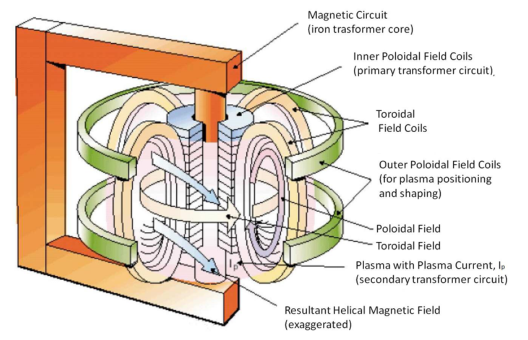
$$D + T \rightarrow {}^{4}\mathrm{He} + n$$



Plasma

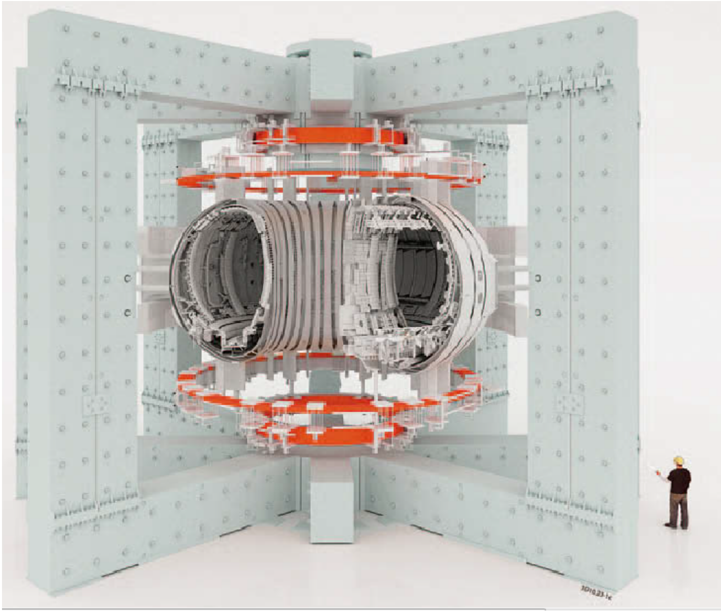
- High temperature and pressure are needed
- Fully ionised gas $\mapsto$ Plasma
- Magnetic field is needed to confine the plasma

What is a Tokamak?

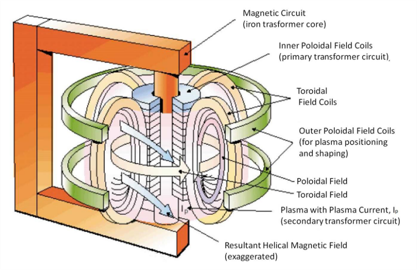


A tokamak is an electromagnetic machine containing a fully ionised gas (plasma) at about 100 million degrees within a torus shaped vacuum vessel. Poloidal and toroidal field coils, together with the plasma current, generate a spiralling magnetic field that confines the plasma.

The JET tokamak

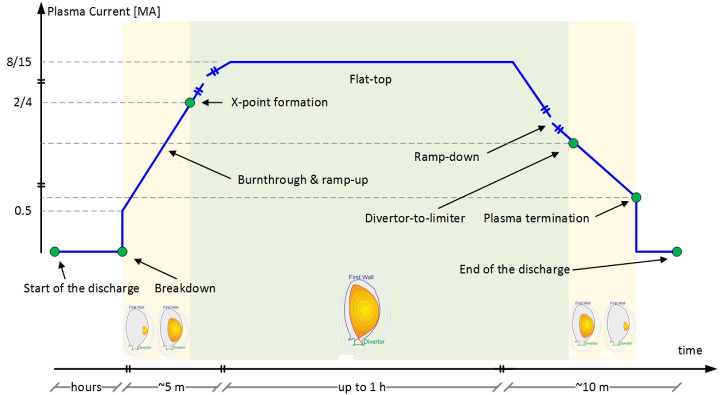


Magnetic control in tokamaks



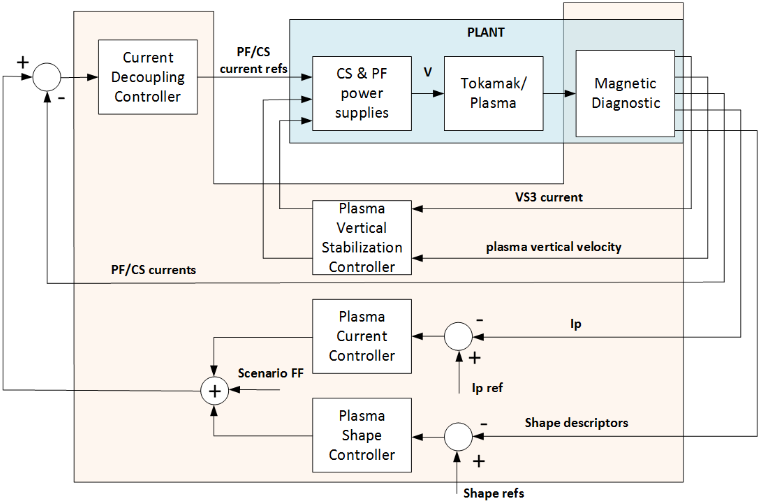
- In tokamaks, **control of the plasma is obtained by means of magnetic fields produced by the external active coils**
- In order to obtain good performance, it is necessary to have a plasma with **vertically elongated cross section** ⇒ **vertically unstable plasmas**
- It is important to **maintain adequate plasma-wall clearance during operation**

A tokamak discharge

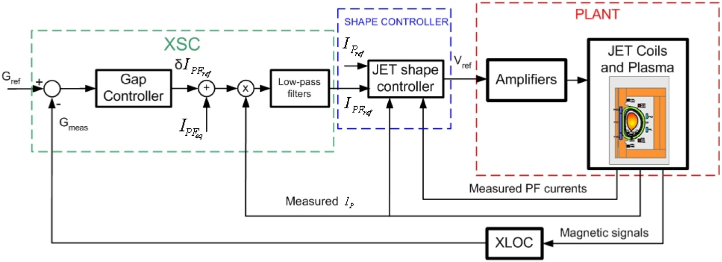


A JET discharge

Example of magnetic control system - A proposal for the ITER tokamak



Example of magnetic control system - The JET tokamak



PPCC Control systems in the mid 90s

PPCC systems for plasma magnetic control

The two main systems run at JET by the Plasma Position and Current Control Group were (and still are!):

Shape Controller (SC)
: C code deployed on a VxWorks/VME/Motorola68k platform

the Vertical Stabilization System (VS)
: C code deployed on 4 Texas Instruments DSPs

- The code was *tailored* for the specific platform
- Lack of modularity
- Different software solutions to interface with the JET software infrastructure (pre-pulse system configuration, post-pulse data collection,. . .)

M. Lennholm et al.,
Plasma control at JET,
Fus. Eng. Design, vol. 48(1–2), Aug. 2000

A new framework for RT applications

- In 2001/2002 the revamping of the SC was planned in order to add the eXtreme Shape Controller algorithm (XSC)
- Within the PPCC group, it was decided to move to a common framework for the development of real-time applications

Aims (User Requirements)

- Standardize the development of real-time applications
- Increase the code reusability
- **Separate (as much as possible) the user application from the software required to interface with the plant infrastructure**
- **Reduce the time needed for commissioning**

High Level System Requirements

The new framework would have been:

- **portable (multi-OS and multi-platform)**
- **modular – the user application would have been easily plugged into an executor of real-time application**
- written in C++ (object oriented approach)

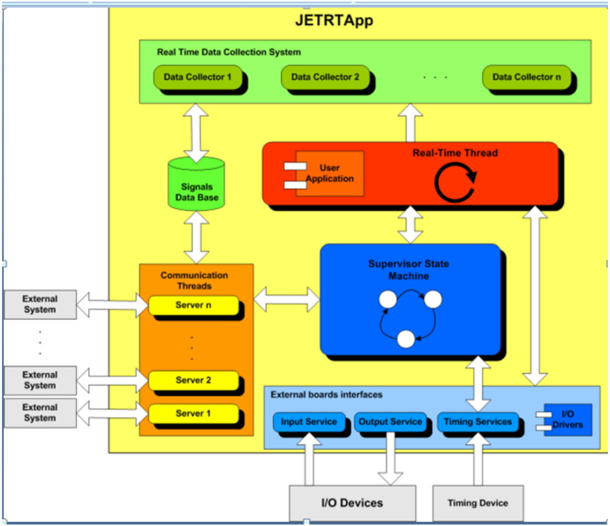
Separation between application and infrastructure software

Why we want to separate application from infrastructure software?

- Scientists (process experts) can abstract from the plant interfaces
- Increase code reusability
- Achieve standardization

JETRT

- The JETRT framework was developed in 2002/2003 to deploy the XSC
- JETRT is based on the cross-platform BaseLib library (developed within the PPCC group)



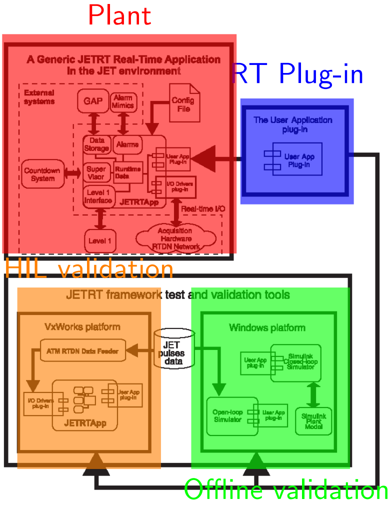
Services and servers

1. Identification of the services
2. Definition of the *servers* interface
3. Implementation (technological solutions)

Real-time application plug-in

The **Real-time Application Plug-in** that can be used to:

- perform offline validation against a plat model
- perform real-time validation with hardware-in-the-loop
- run the real-time system on the plant



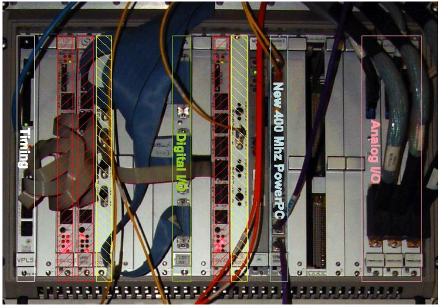
XSC

- The *new* SC (including the XSC) was deployed on a 400 MHz G4 PowerPC running VxWorks
- 2 ms control loop (but it can easily run at 1 ms)

Commissioning of the JETRT framework and of the XSC

- Thanks to portability, an exhaustive debug of both the JETRT framework and the XSC was performed
  - **offline**, on a Windows–based platform
  - **in lab**, with a mockup of the JET timing system and of the I/O
- **Only 3 days of testing on the plant were needed for the commissioning of the new system**

The JET XSC control system



- VME architecture
- PowerPC 400 MHz
- 512 MB RAM
- ATM (for real-time comms) and Ethernet (for non-real-time comms) network interfaces
- VxWorks OS
- Sampling frequency 500 Hz

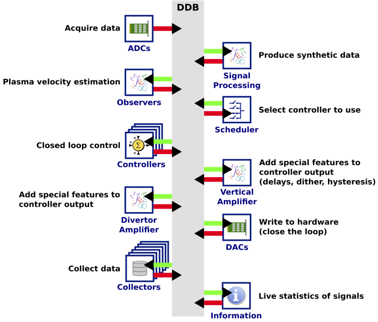
JETRT limitations

- JETRT didn't provide a real separation between the user application from the plant-interface software!
- In 2011, about 1 ppm was needed to include a new component in the XSC (the Current Limit Avoidance system)!

From JETRT to MARTe

- *More* modularity → Generic Application Modules (GAMs)
- *Real* separation → Dynamic Data Buffer (DDB)

MARTe

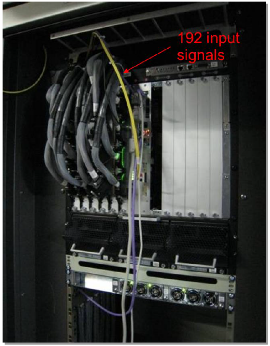


- To solve the existing problems, the JETRT framework *evolved* into MARTe
- MARTe allows to exploit tasks (thread) segregation on multi-core architectures in order to achieve hard-real-time also with a *vanilla* Linux!
- First used in 2008 to implement the JET *vertical stabilization system* (sampling frequency 20 *kHz*, with jitter $< 1\ \mu s$)
- MARTe is currently used in different fusion laboratories – JET (UK), COMPASS (Czech Republic), KSTAR (South Corea), FTU (Italy), RFX (Italy), ISTTOK (Portugal)
- MARTe is distributed under EU open-source licence → http://efda-marte.ipfn.ist.utl.pt/

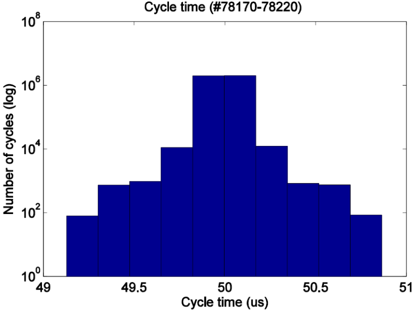
The Vertical Stabilization at JET 1/3

- Bus architecture based on ATCA+PCIe
- Multi-core processor (Inter Core2 Quad)
- Linux+RTAI OS
- 192 signals acquired by ADCs (18 bits 2 MHz) and transferred at each cycle
- 50 μs control loop cycle time with jitter < 1 μs
- Always in real-time (24 hours per day)
  - 1.728×10^9 50 μs cycles/day

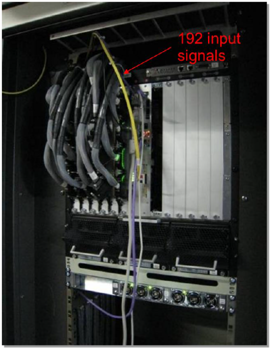
The Vertical Stabilization at JET 2/3

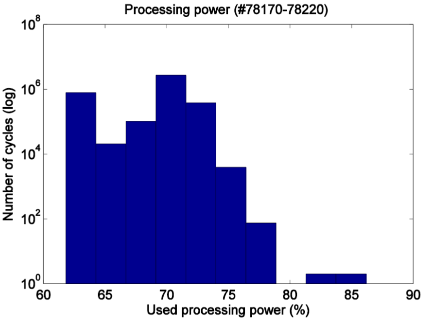


jitter < 1 µs



The Vertical Stabilization at JET 3/3





Conclusive remarks

- Real-time systems are required whenever time constraints are included within the requirements
- The implementation of control/automation systems always call for real-time systems
- The deployment of a real-time systems usually requires a detailed knowledge of both the hardware architecture and of the software infrastructure (mainly the OS)
- For specific application, the developer can abstract from the underlying architecture
  - PLC development environments
  - Microcontroller SDKs
  - ...
  - Frameworks for specific applications (e.g. the nuclear fusion control applications)
- Multi-processor/multi-core architectures allows to achieve real-time behavior without necessarily relying on RTOS

Thank you!