Targeting the Scicos Codegenerator for embedded applications

Ing. Roberto Bucher

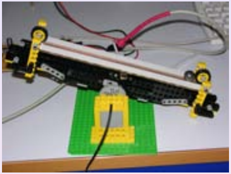
Scuola Universitaria Professionale della Svizzera Italiana (SUPSI)

Paris - 1.7.2009

Objectives

(Prices in Euro)	Commercial Control design environment	Proposed solution
Core SW	1950.-	0.-
Control SW	1000.-	0.-
Graphical GUI	3000.-	0.-
Code generator	7500.-	0.-
RT target	3000.-	0.-

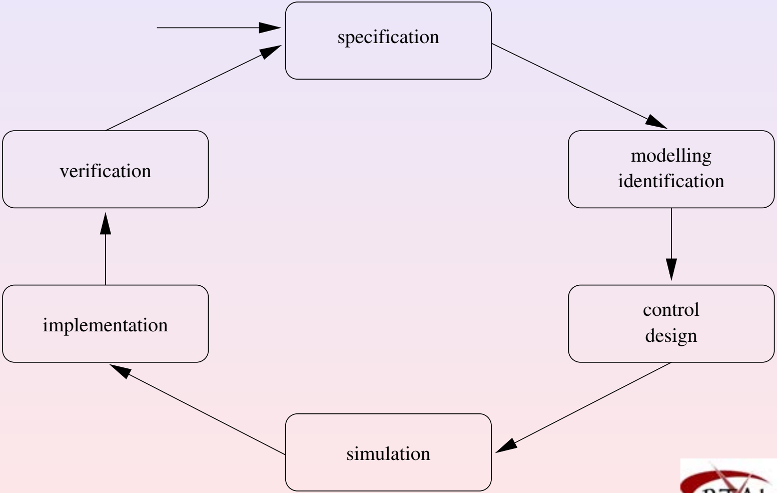
Objectives



Outline

1. Rapid Controller Prototyping
2. Linux RTAI
3. Scilab/Scicos and Linux RTAI
4. Implementation
5. Examples
6. Conclusions

Control Design Loop



Rapid Controller Prototyping - Requirements

- 2 main components
  1. An OS with hard real-time features
  2. A Computer Aided Control System Design environment including a code generator

→ Linux RTAI + RTAI-Lab + Scilab/Scicos

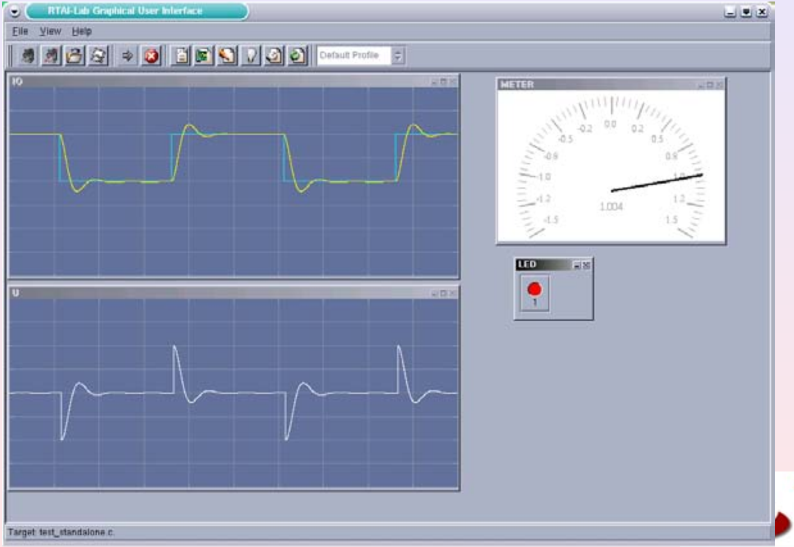
The Linux RTAI project

- Hard real-time extension to the Linux OS
- Based on the ADEOS pipeline (with some improvements!)
- Free Open Source Software (FOSS)
- Implementation of hard real-time controllers using general purpose hardware
- Same PC for the controller design, the hard real-time controller task and the soft real-time monitor task
- Hard real-time in kernel and user space
- Distributed control through the net_rpc module

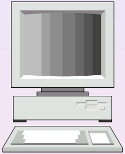
Linux RTAI - Features

- Latency: $< 10\mu s$ depending on the HW
- Typical sampling frequencies: $1 \ldots 10kHz$

The GUI application - xrtailab

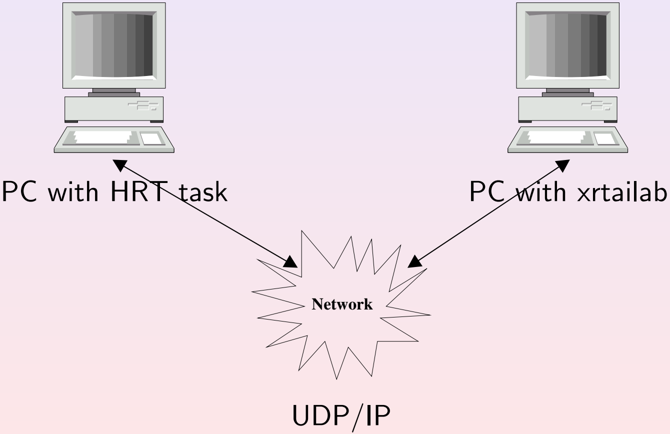


The GUI application - xrtailab



PC with HRT task and xrtailab

The GUI application - xrtailab



PC with HRT task

PC with xrtailab

UDP/IP

Systems controlled with Linux RTAI

Cycab (INRIA)

Biped Robot (INRIA)

http://www.inrialpes.fr/sed

RTAI example 2

National Radio Astronomy Observatory controlled with Linux RTAI - West Virginia

http://www.gb.nrao.edu/43m/

Add-ons for Scilab/Scicos

The link between Scilab/Scicos and Linux RTAI is implemented through:

- A code generator → RTAICodeGen_.sci.
- A *rtmain.c* (main file for the real-time task) specific for the Scilab/Scicos environment.
- An external "template Makefile"
- A new scicos palette *RTAI-Lib.cosf*.
- Macros for the new blocks (Scicos interface functions).
- A RTAI library specific for the new Scicos blocks (Scicos implementation functions).

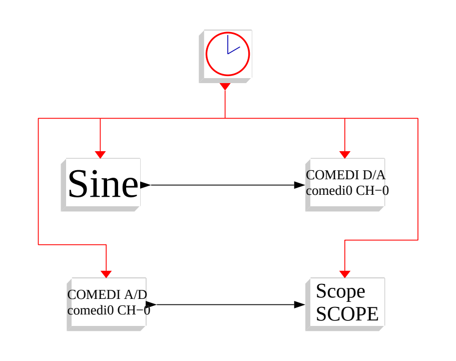
RTAICodeGen_.sci

The code generator used in RTAI is a modification of the standard Scicos code generator

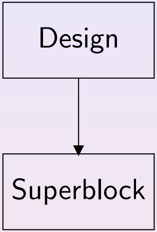
- Only stand-alone generation with specific main procedure (rtmain.c)
- Sensors and actuators are integrated in the Scicos block diagram as specific custom peripheral blocks
- Each block can be identified by a name which is then referred in RTAI-Lab

Code generation

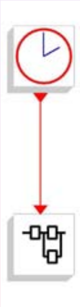
Design



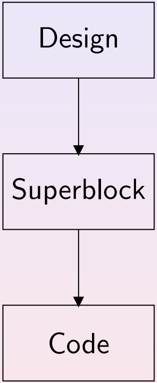
Code generation


Design

Superblock



Code generation



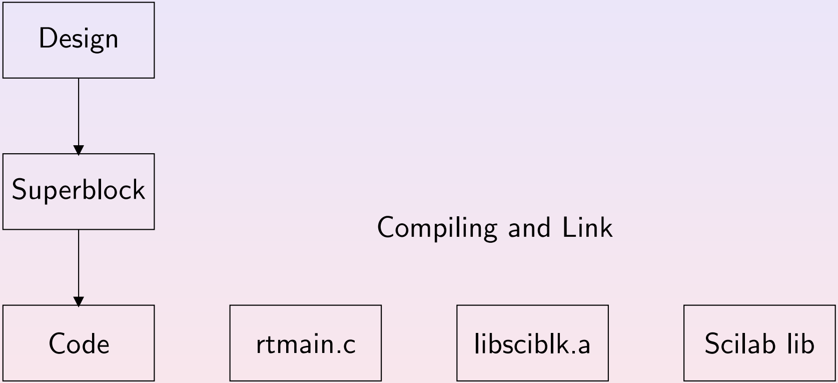
Design

Superblock

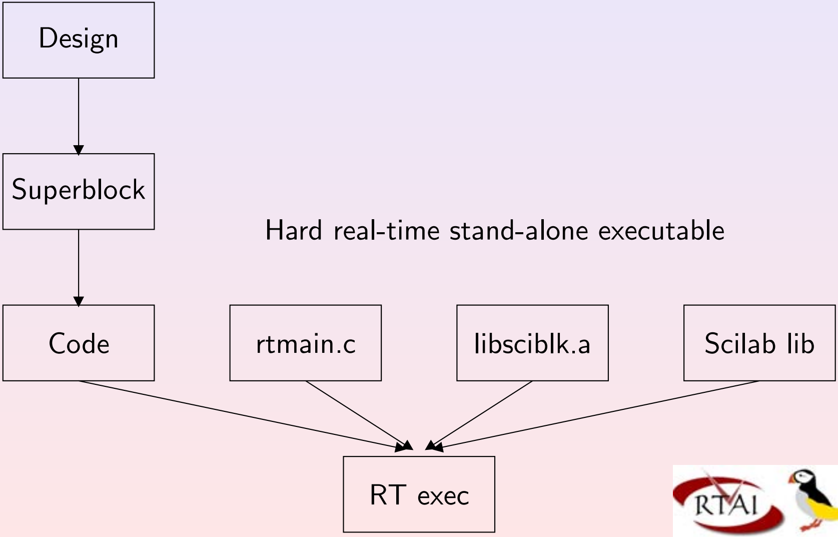
Code

RTAICodegen_.sci

Code generation

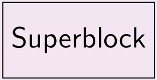


Code generation



Hard real-time stand-alone executable

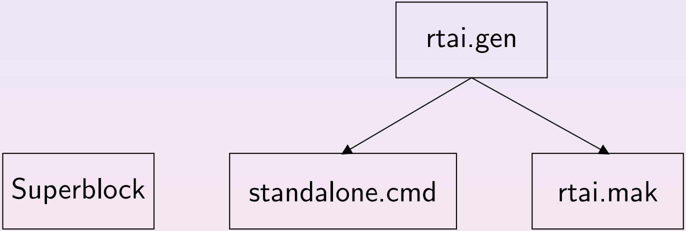
Code generation -Details



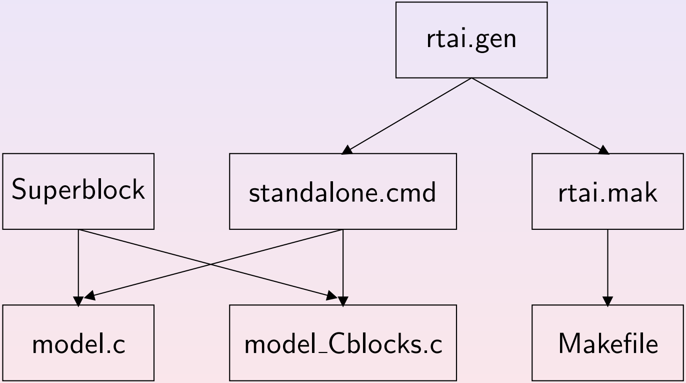
Superblock

Code generation -Details

Code generation -Details



Code generation -Details



Target specific files - The “gen” file

```
rtai.mak
standalone.cmd
```

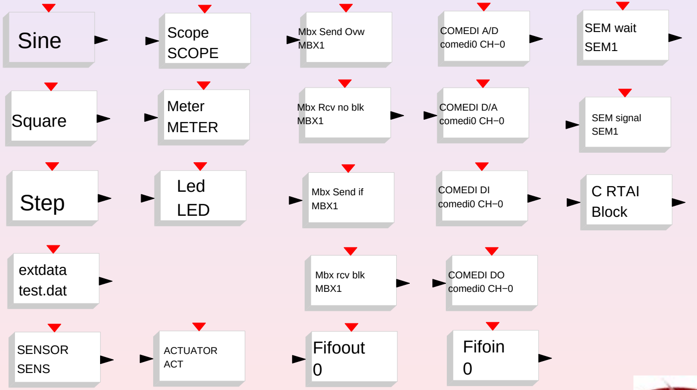
Target specific files - The “cmd” file

```
[CCode,FCode]=gen_blocks()
[Code,Code_common]=make_standalone42();
files=write_code(Code,CCode,FCode,Code_common);
Makename=rt_gen_make(rdnom,files,archname);
ok=compile_standalone();
```

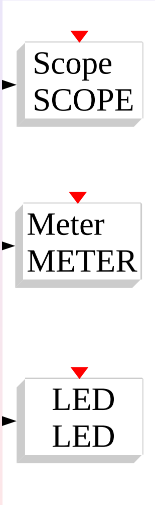
Target specific files - The “mak” file

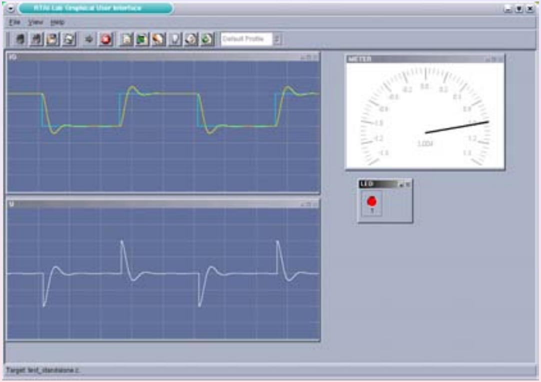
```
all: ../$$MODEL$$
...
CC = gcc
CC_OPTIONS = -O -DNDEBUG -Dlinux -DNARROWPROTO -D_GNU_SOURCE

MODEL = $$MODEL$$
OBJSSTAN = rtmain.o common.o $$MODEL$$.o $$OBJ$$

SCILIBS = $(SCIDIR)/libs/scicos.a $(SCIDIR)/libs/poly.a $(SCIDIR)/libs/calelm.a
OTHERLIBS =
ULIBRARY = $(RTAIDIR)/lib/libsciblk.a $(RTAIDIR)/lib/liblxrt.a

CFLAGS = $(CC_OPTIONS) -O2 -I$(SCIDIR)/routines -I$(SCIDIR)/routines/scicos $(C

rtmain.c: $(RTAIDIR)/share/rtai/scicos/rtmain.c $(MODEL).c
        cp $< .

../$$MODEL$$: $(OBJSSTAN) $(ULIBRARY)
        gcc -static -o $@  $(OBJSSTAN) $(SCILIBS) $(ULIBRARY) -lpthread $(COMED
        @echo "### Created executable: $(MODEL) ###"
```

The new Scicos palette

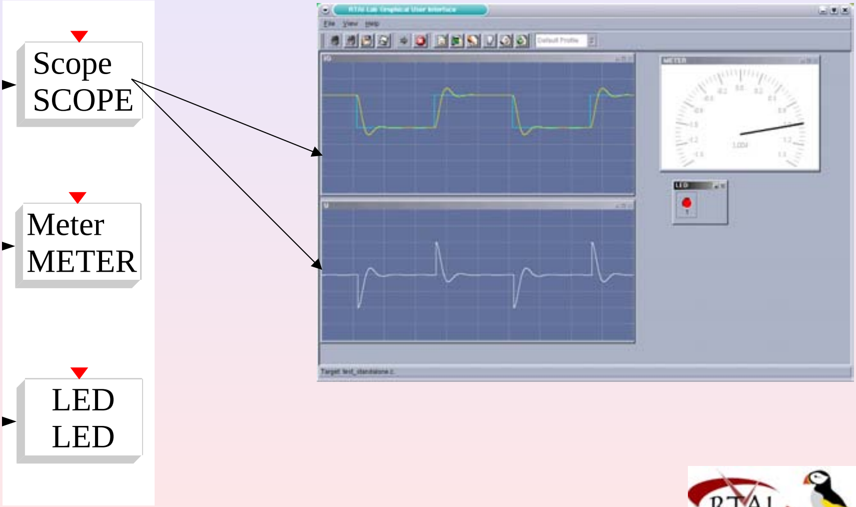


Specific blocks for RTAI-Lab

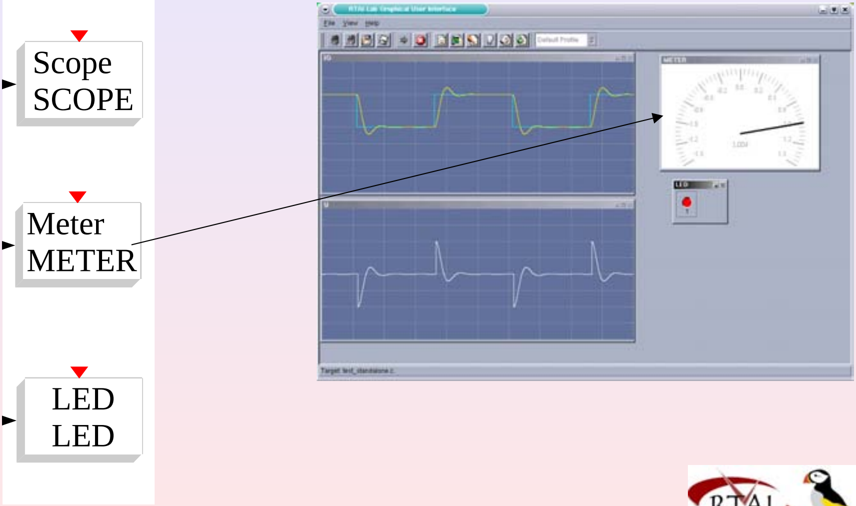




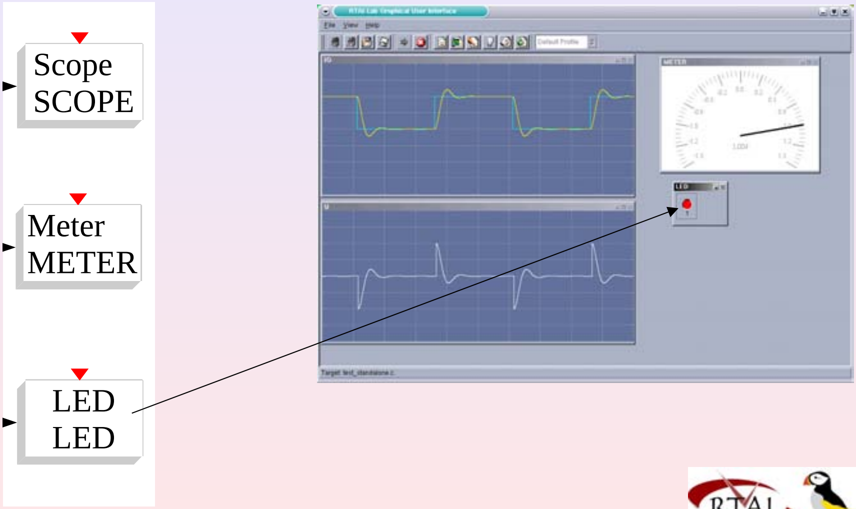
Specific blocks for RTAI-Lab



Specific blocks for RTAI-Lab



Specific blocks for RTAI-Lab

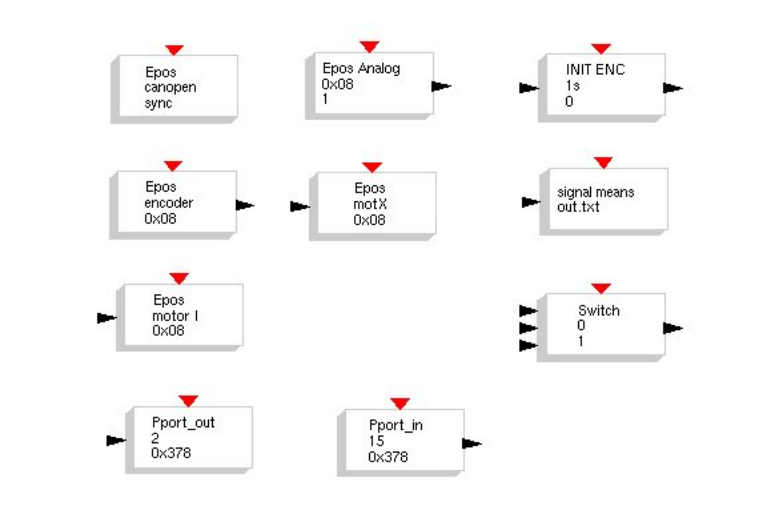


Scope
SCOPE

Meter
METER

LED
LED

Additional palettes

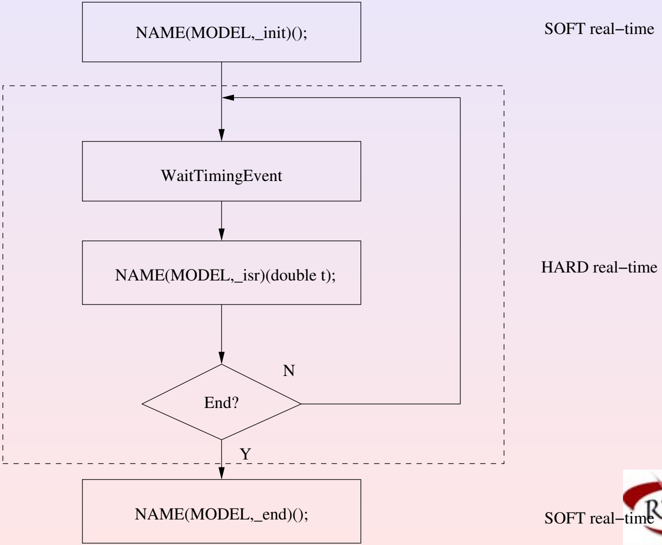


The main file *rtmain.c*

3 threads:

- Main thread *rtMain*
- Hard RT thread *rt_BaseRate*
- Communication thread with the GUI client application *rt_HostInterface*

The *rt_BaseRate* thread



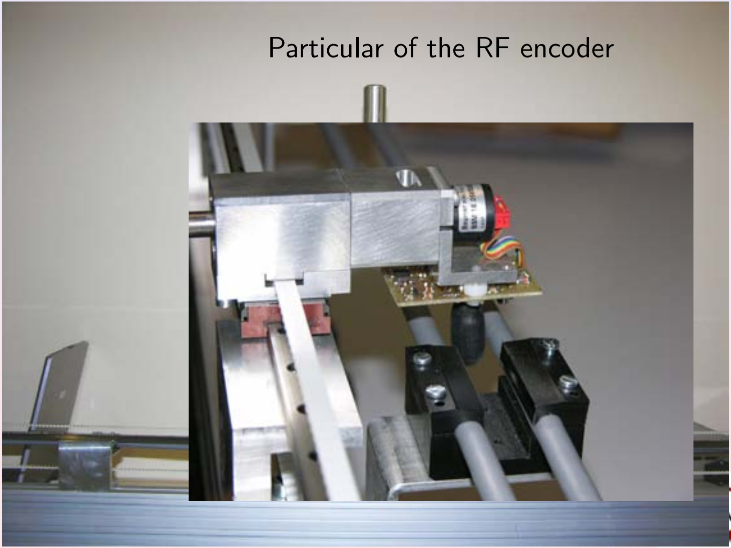
Adding a new block

Each new block needs

- An interface function (".sci")
- An implementation function (".c")

The Inverted Pendulum

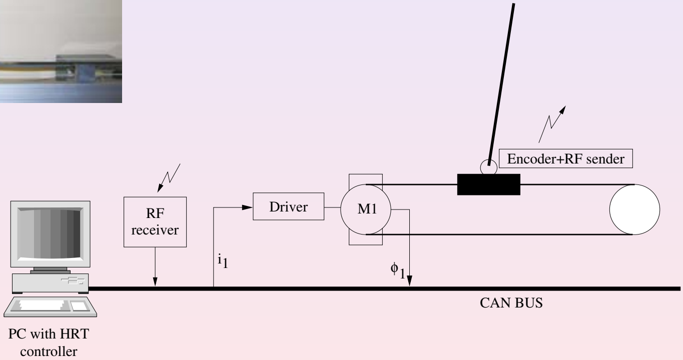
The Inverted Pendulum



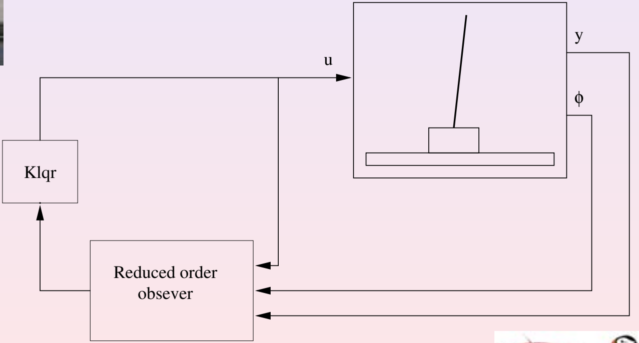
Particular of the RF encoder

The Inverted Pendulum

The Inverted Pendulum

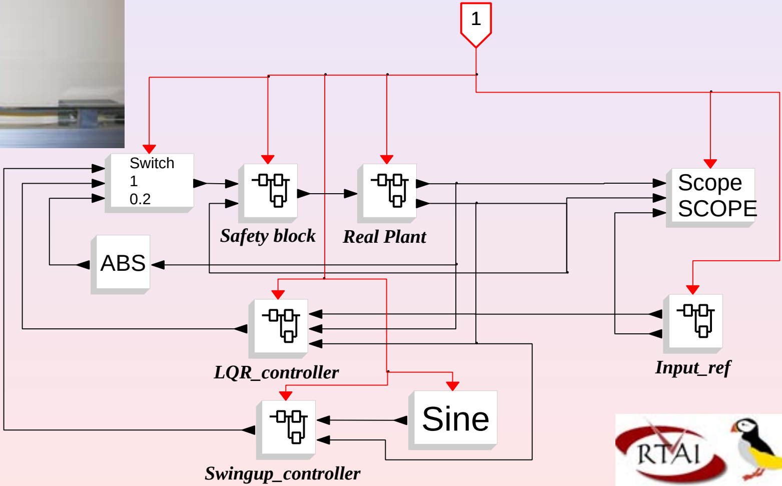


The Inverted Pendulum

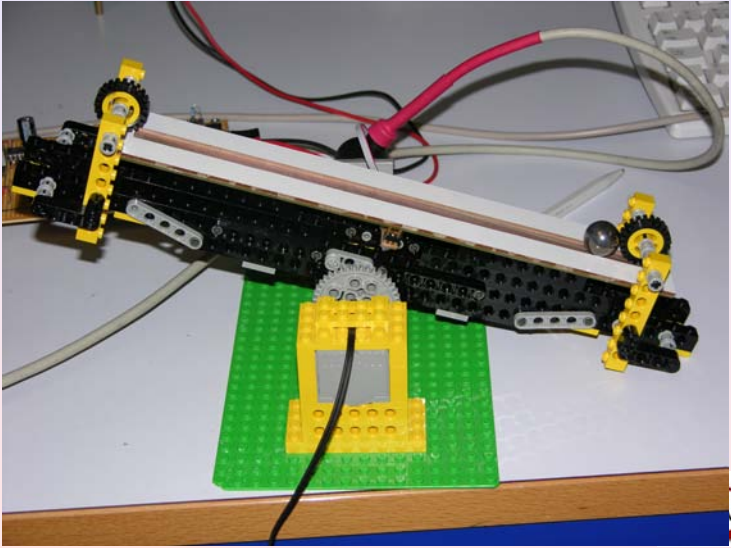


The Inverted Pendulum

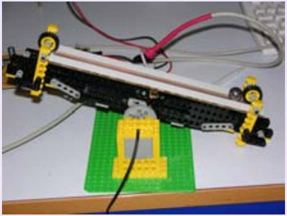
The Scicos scheme



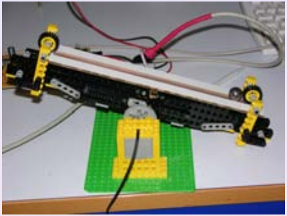
Ball on Beam



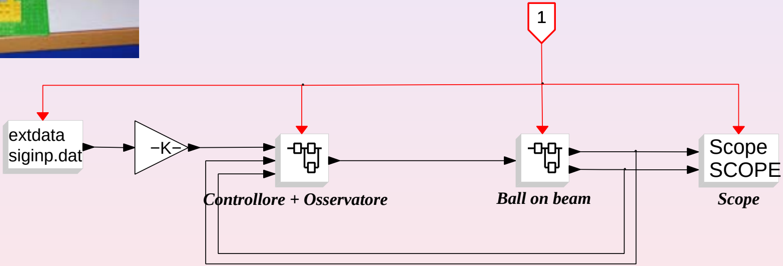
Ball on Beam



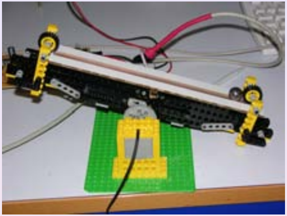
Ball on Beam



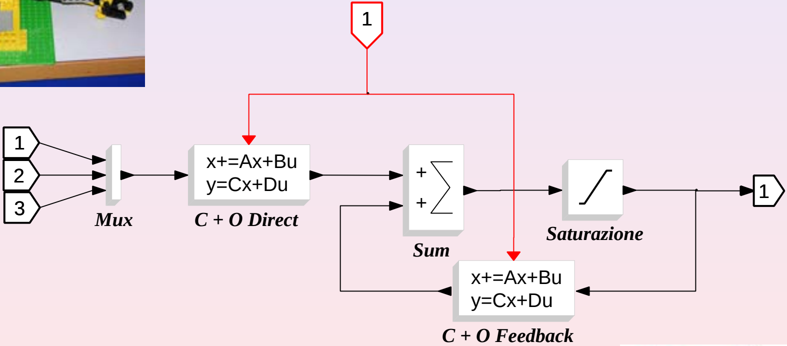
The Scicos scheme



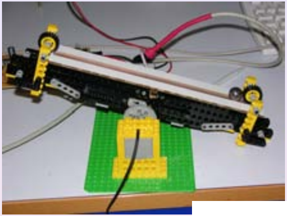
Ball on Beam



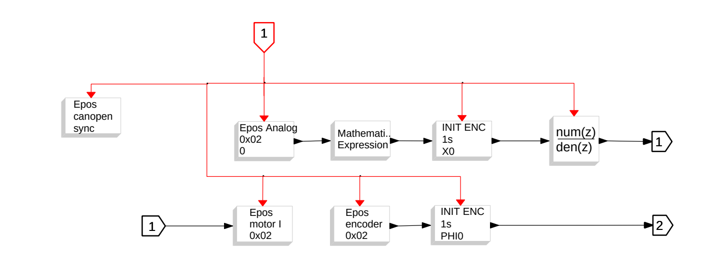
The anti-windup controller



Ball on Beam



The CANopen blocks



Videos

Sample Videos

Conclusions

- Open source!!!
- Complete and stable.
- Able to solve complex control problems in a simple way.
- It runs parallel to a MATLAB/Simulink/RTW environment in my student laboratory

Questions?

Thank You

- www.rtai.org
- www.dti.supsi.ch/∼bucher/scilab.html
- www.dti.supsi.ch/∼smt/labo04.html