Distributed, Modular HTL

Christoph Kirsch
Universität Salzburg

TU München
June 2009

htl.cs.uni-salzburg.at[#]

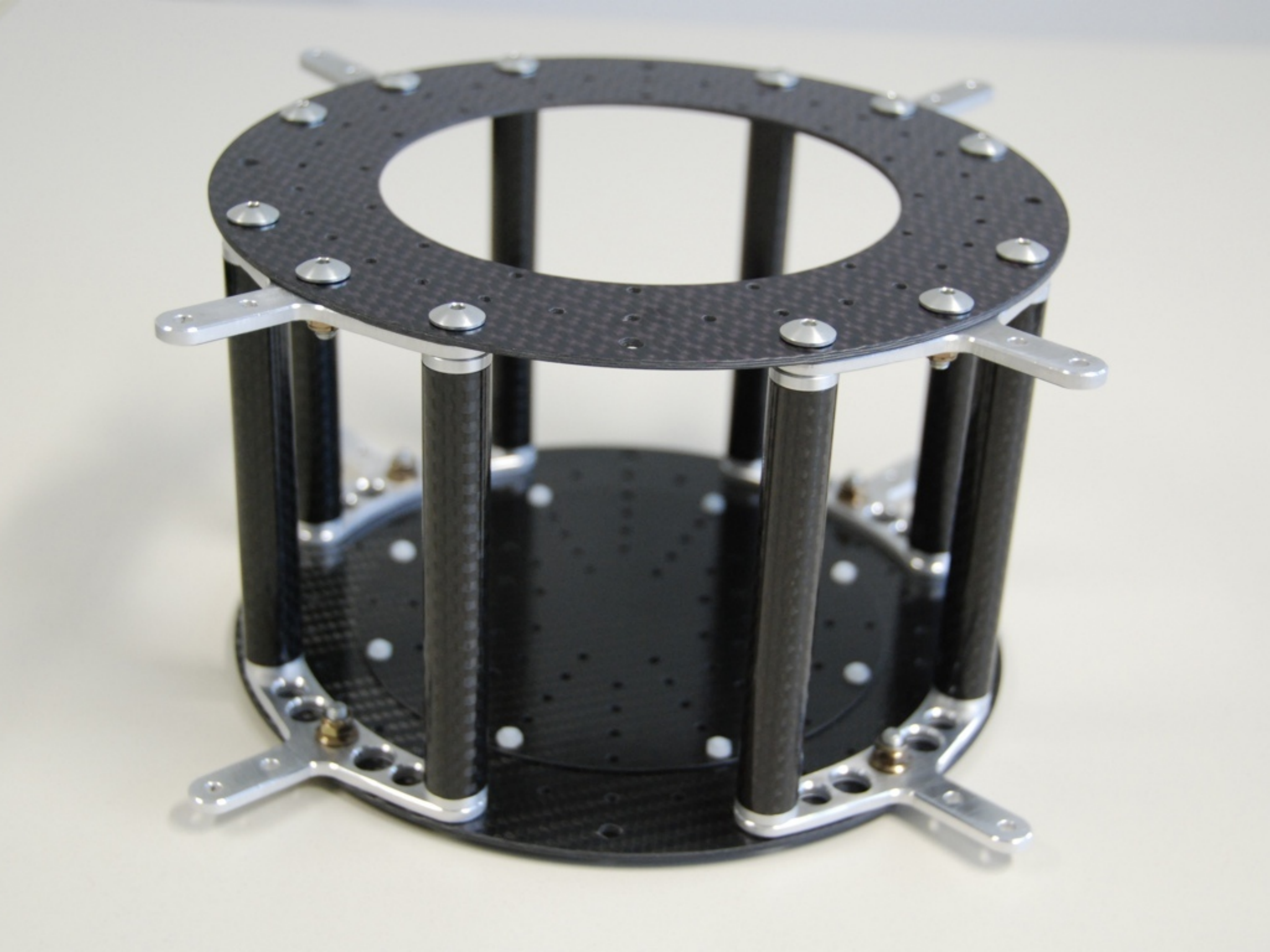
- Thomas Henzinger (IST Austria)
- Christoph Kirsch (University of Salzburg)
- Eduardo Marques (University of Porto)
- Ana Sokolova* (University of Salzburg)

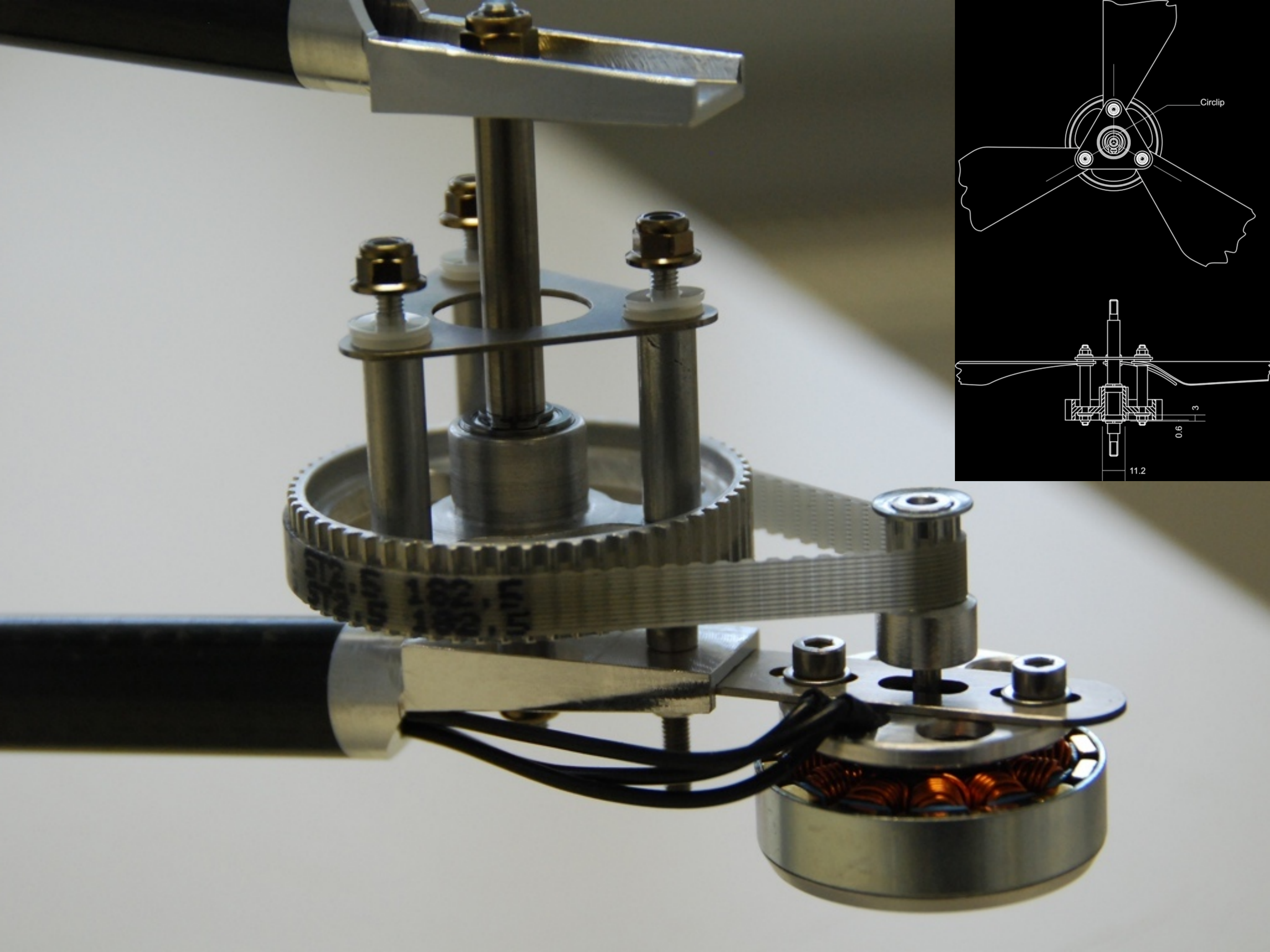
[#]Supported by a 2007 IBM Faculty Award, the EU ArtistDesign Network of Excellence on Embedded Systems Design, and Austrian Science Fund Project P18913-N15.
*Supported by Austrian Science Fund Project V00125.

The JAviator

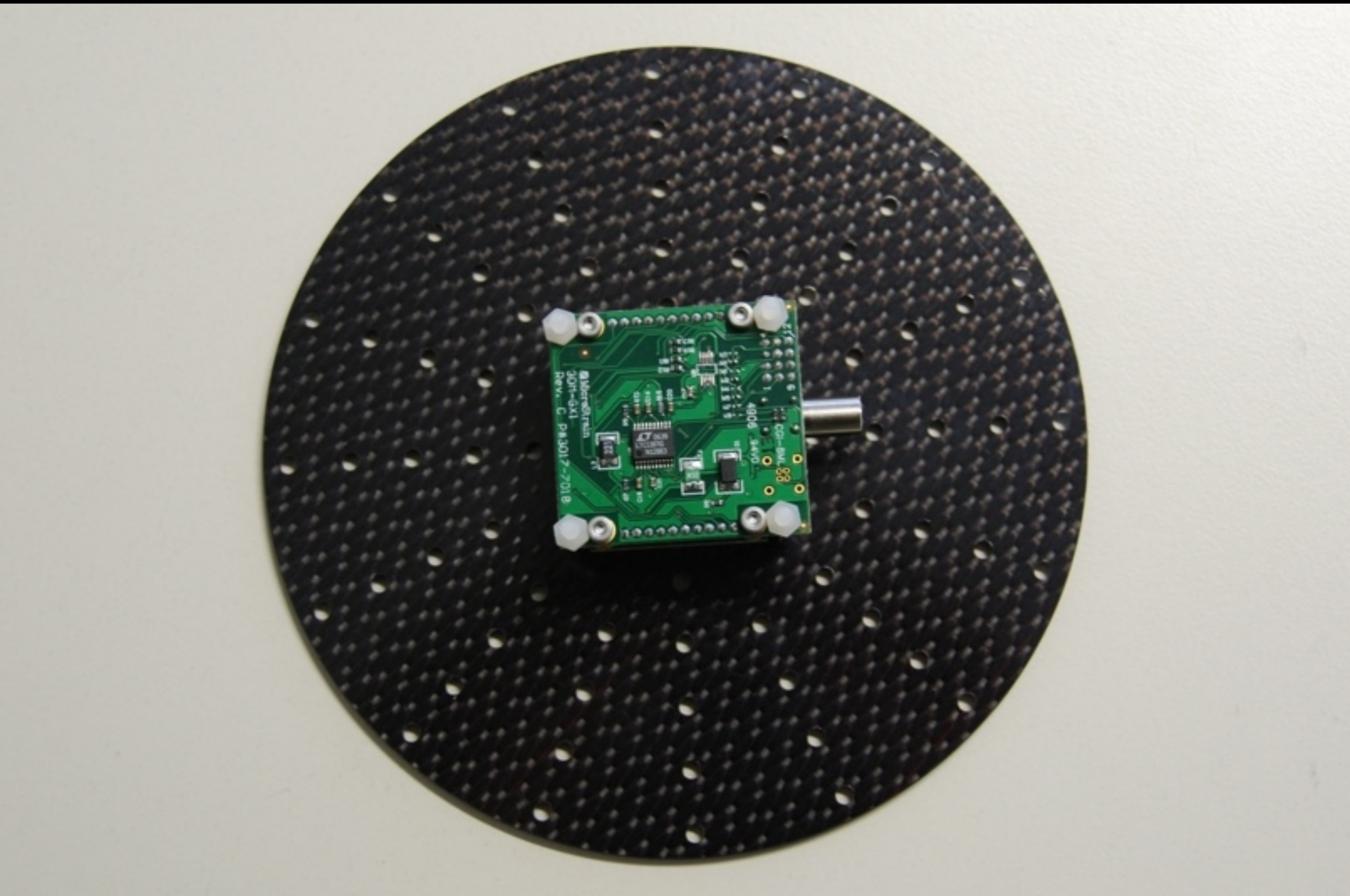
javiator.cs.uni-salzburg.at

Quad-Rotor Helicopter




Circlip
3
0.6
11.2

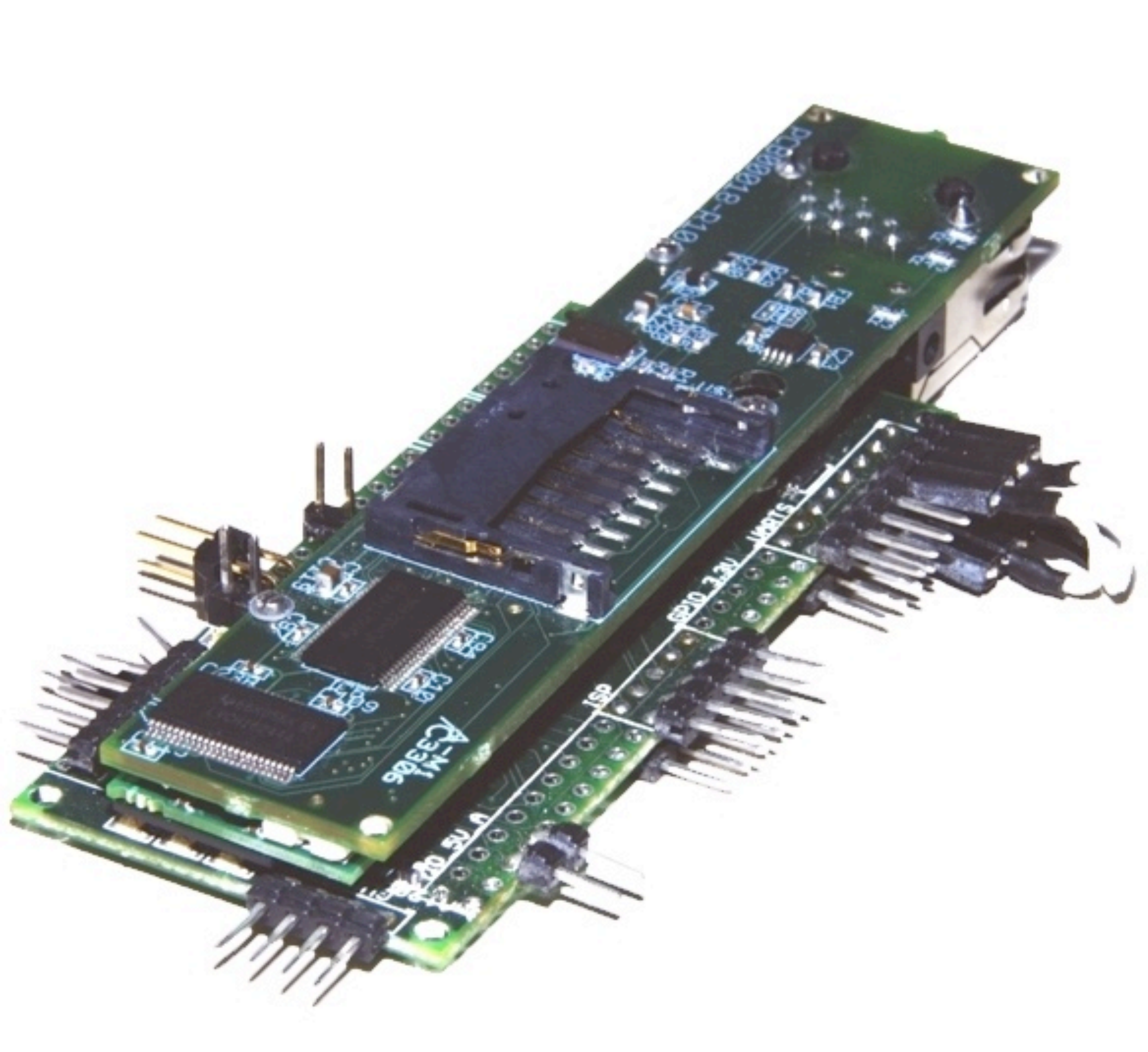

25/12/2005



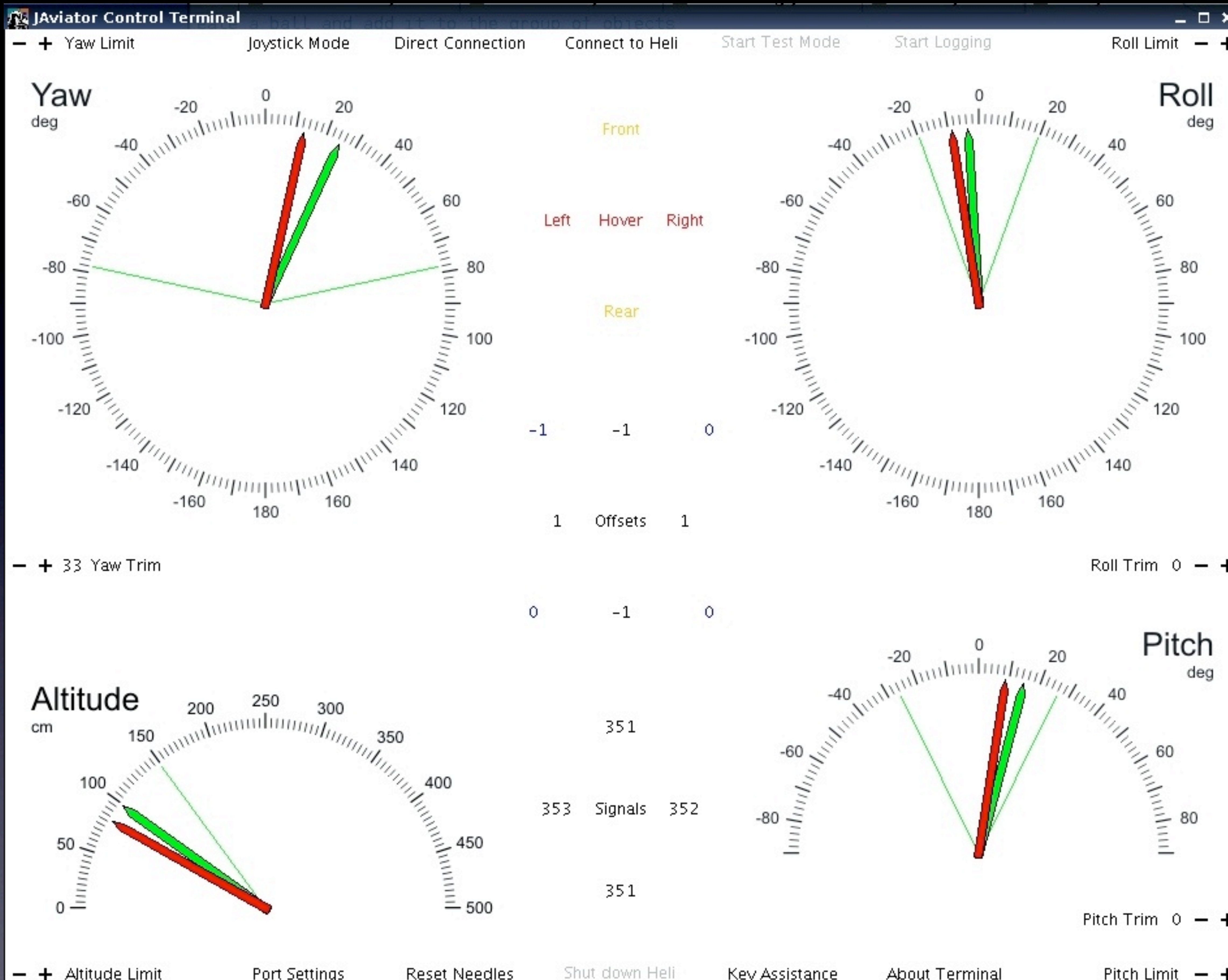
Gyro

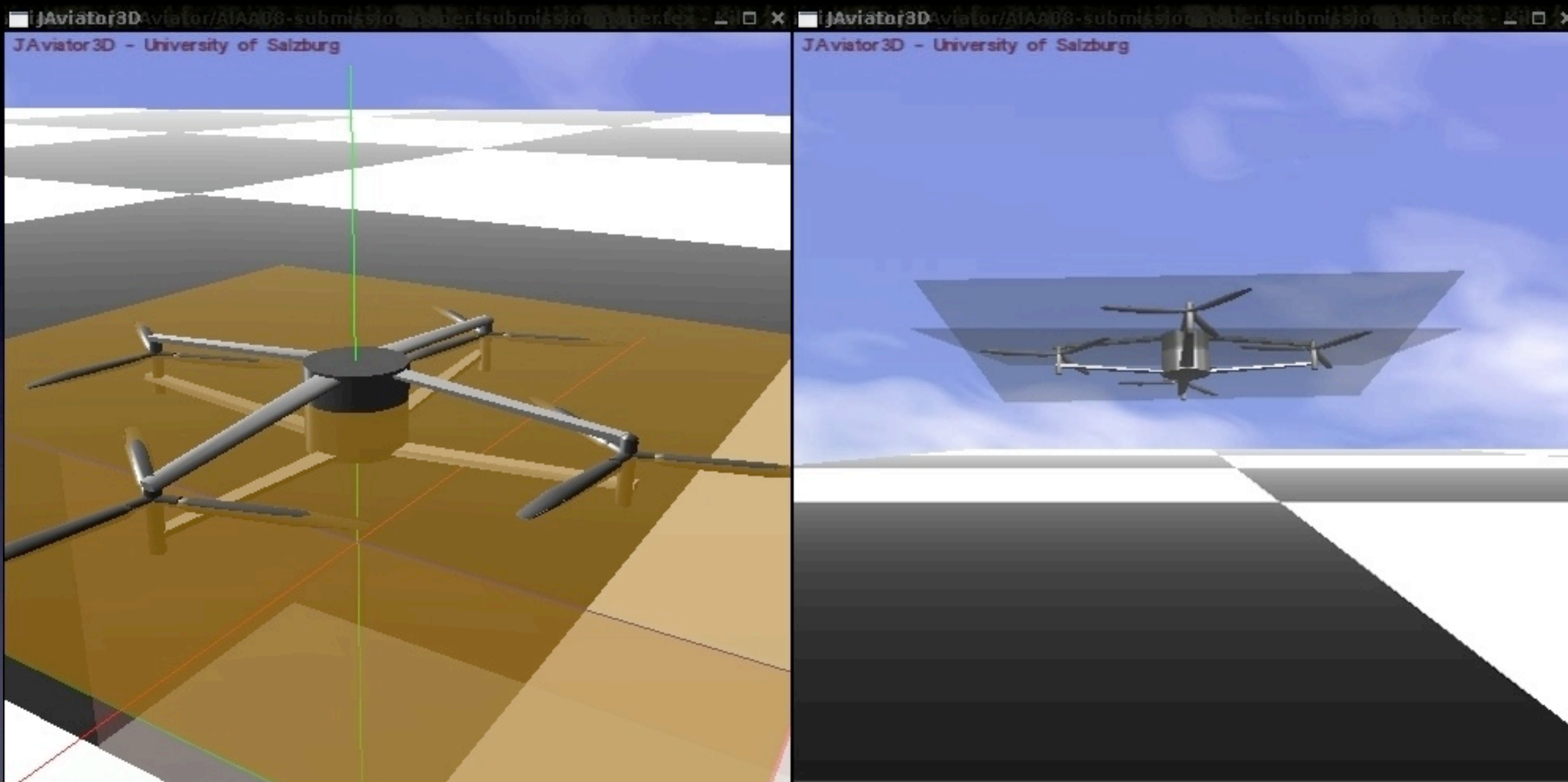
Propulsion

Gumstix

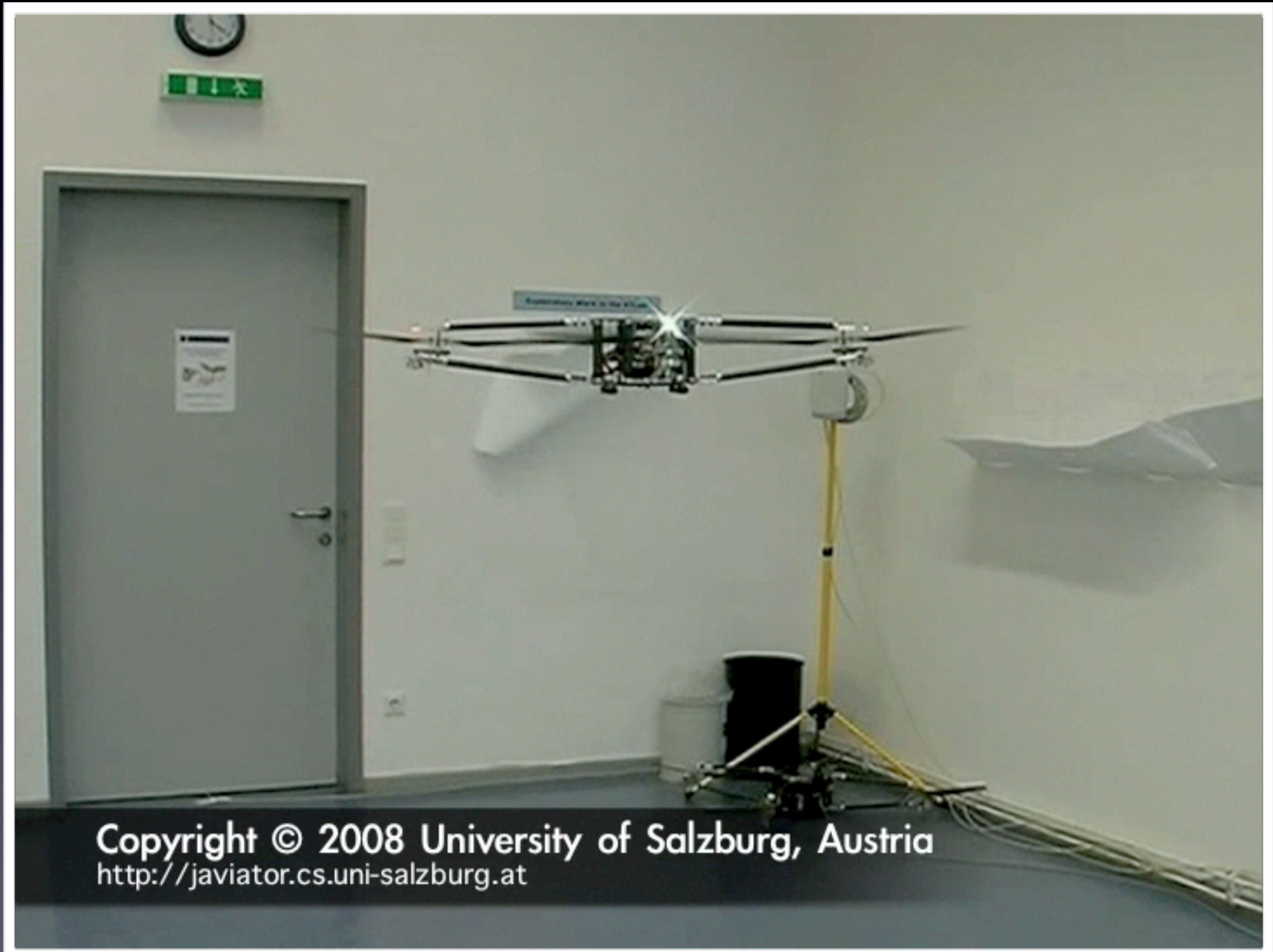


600MHz XScale, 128MB RAM, WLAN, Atmega uController


JAviator Control Terminal
− + Yaw Limit
Joystick Mode
Direct Connection
Connect to Heli
Start Test Mode
Start Logging
Roll Limit − +
Yaw
deg
Roll
deg
Front
Left
Hover
Right
Rear
-1
-1
0
1
Offsets
1
− + 33 Yaw Trim
Roll Trim 0 − +
0
-1
0
Pitch
deg
Altitude
cm
351
353
Signals
352
351
Pitch Trim 0 − +
− + Altitude Limit
Port Settings
Reset Needles
Shut down Heli
Key Assistance
About Terminal
Pitch Limit − +


JAviator3D
JAviator3D - University of Salzburg
JAviator3D
JAviator3D - University of Salzburg

Indoor Flight
STARMAC Controller

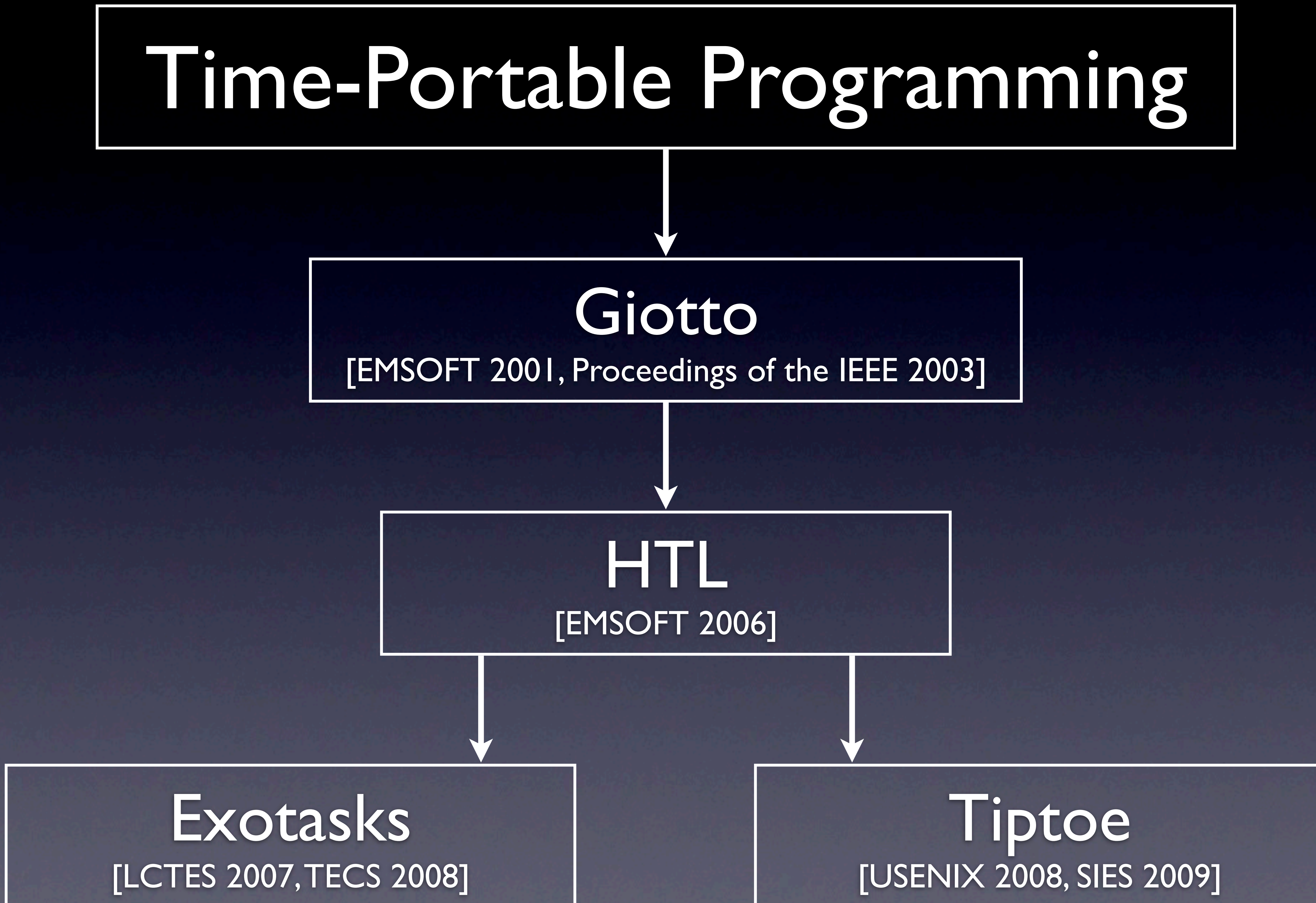


Outdoor Flight STARMAC Controller

Outdoor Flight
Salzburg Controller

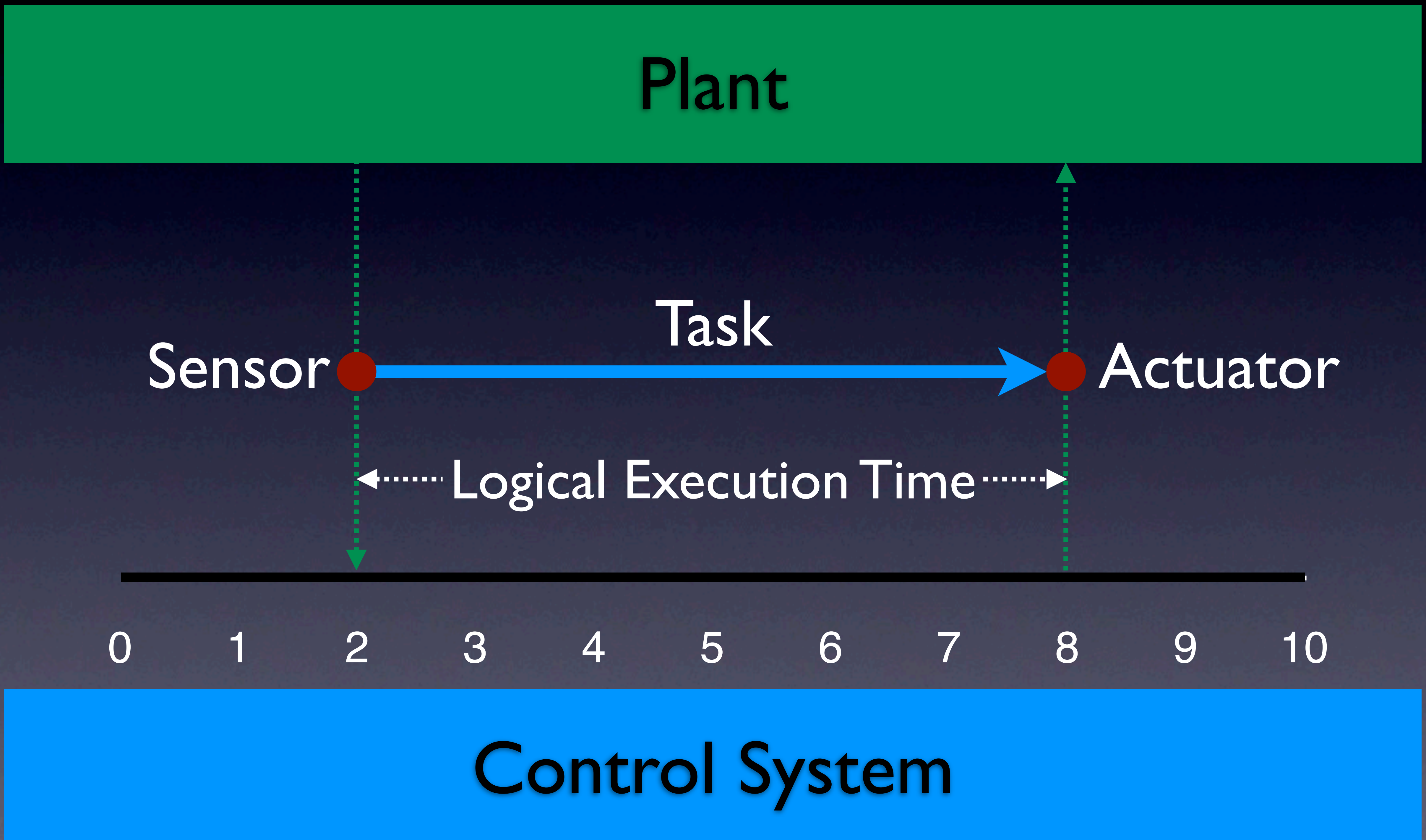
What's next?

- Autonomous single-vehicle flights
  - position controller
  - waypoint controller
- Autonomous multi-vehicle flights
  - mission controller

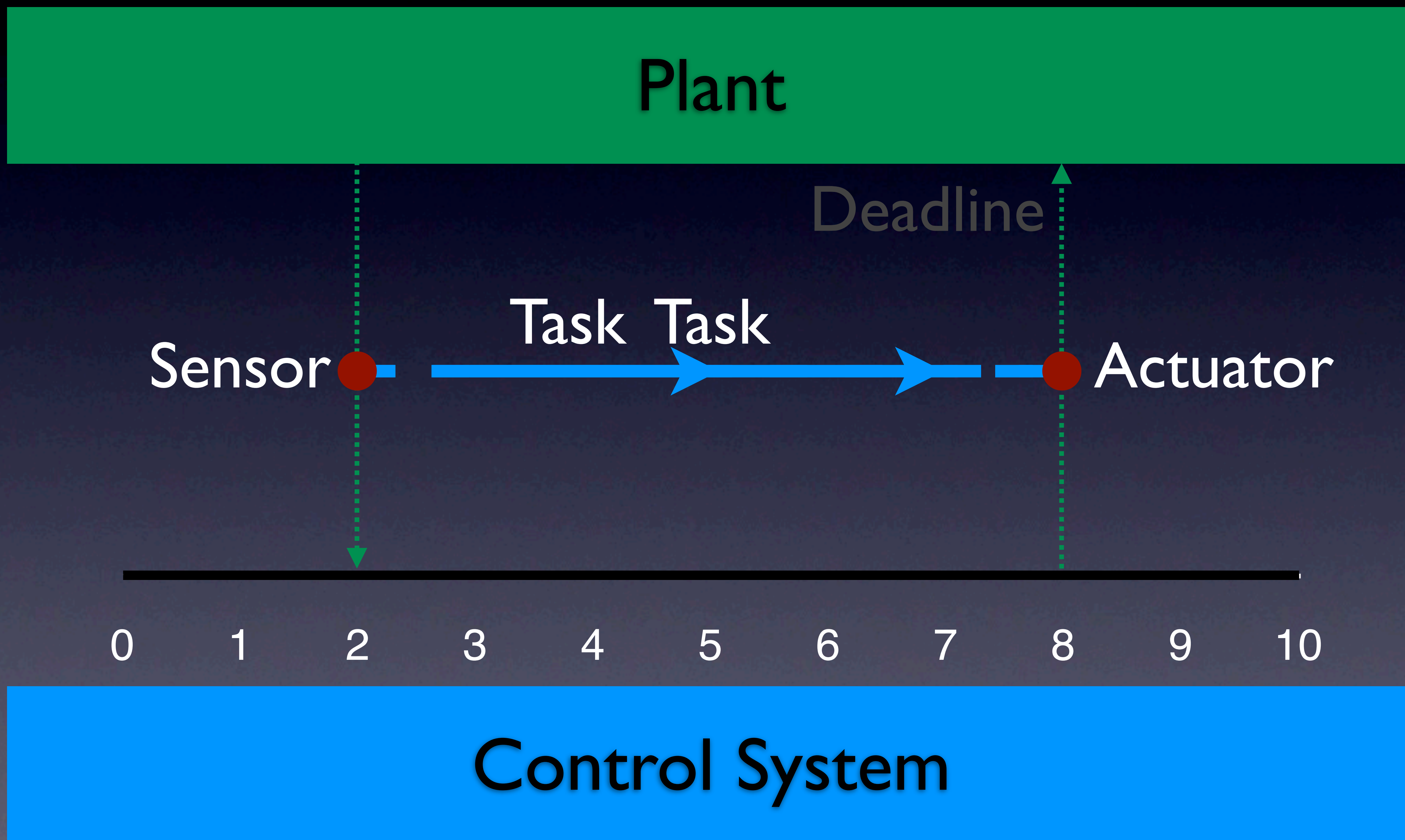


Time-Portable Programming
Giotto
[EMSOFT 2001, Proceedings of the IEEE 2003]
HTL
[EMSOFT 2006]
Exotasks
[LCTES 2007, TECS 2008]
Tiptoe
[USENIX 2008, SIES 2009]

Logical Execution Time



Actual Execution Time



Time Determinism

Plant

A system's I/O behavior is *input-determined* if,
for all sequences *I* of input values and times,
the system always produces
unique sequences *f(I)* of output values and times.

Control System

Time-Portable Programming
with
Exotasks
=
Java + HTL +
Real-Time Garbage Collection

Time-Portable Programming
with
Tiptoe
=
Virtual Machines +
Variable-Bandwidth Servers +
Compact-fit Memory Management

Hierarchical Timing Language

- HTL is a real-time coordination language
- HTL essentially has four building blocks:
  - task (computation, implemented in C/Java)
  - mode (sequential composition)
  - module (parallel composition)
  - program (abstraction, refinement)
- an HTL program is a hierarchical, tree-like structure whose nodes are such blocks

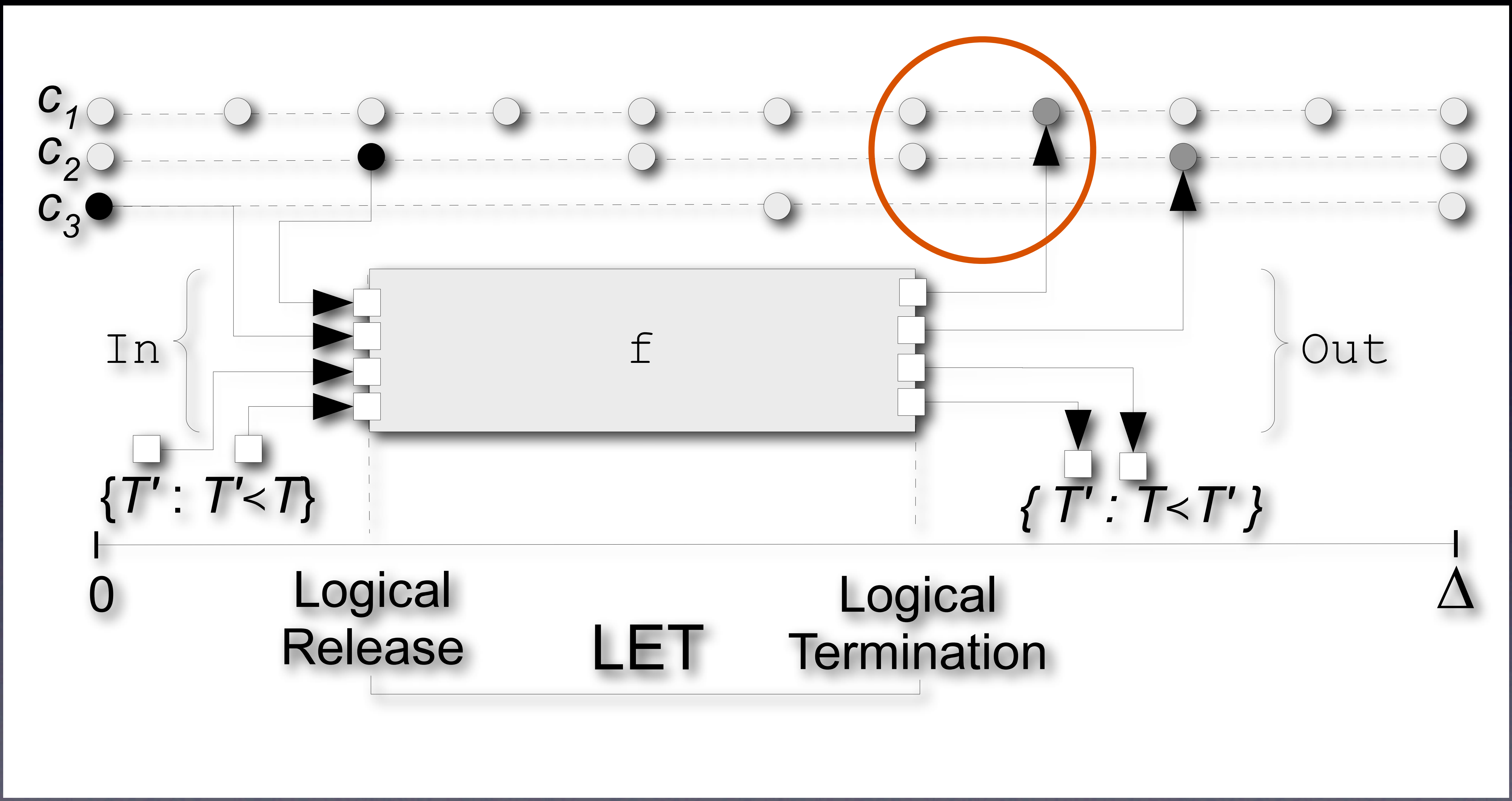
Tasks

- tasks have input and output ports
- tasks compute outputs from inputs
  - outputs are determined by the inputs
  - no side effects, no synchronization
- tasks execute <u>periodically</u>

Communicators

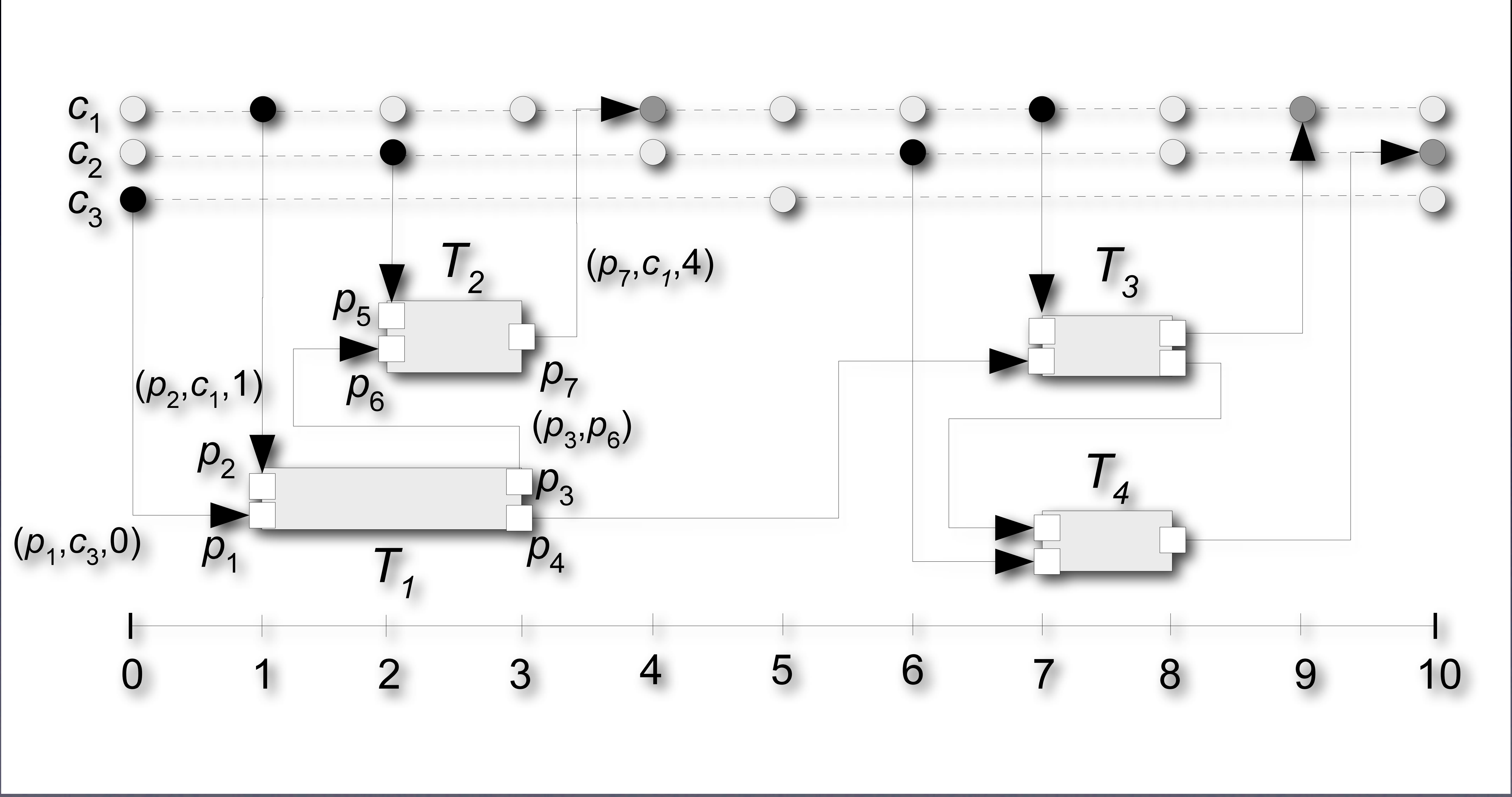
- tasks with the same period may communicate through ports
  - creates task precedences
- tasks with different periods must communicate through communicators
  - creates logical execution time (LET)
- communicators are periodically updated, program-wide variables

Example



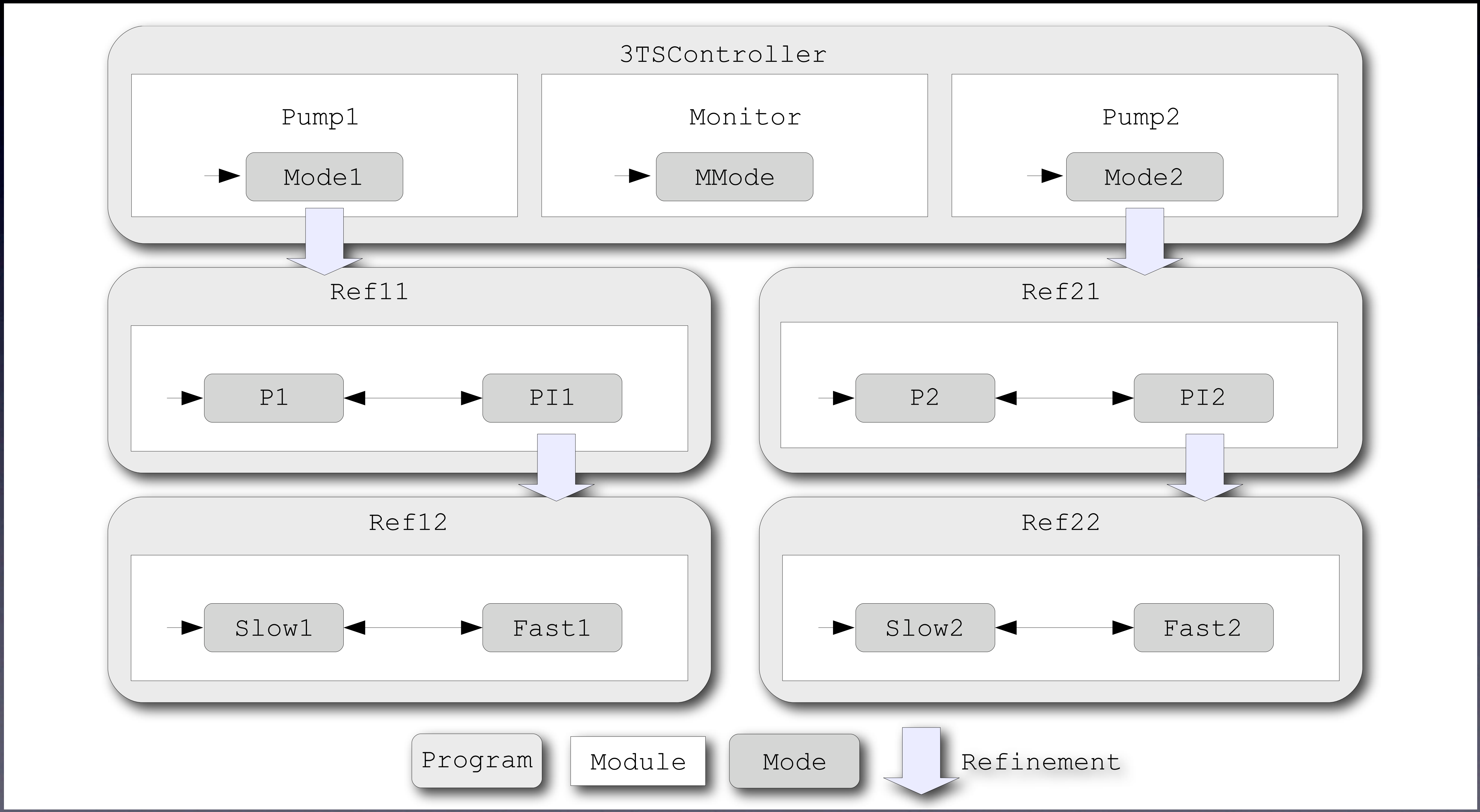
c1
c2
c3
In
f
Out
{T' : T'<T}
{ T' : T<T' }
0
Logical
Release
LET
Logical
Termination
Δ

Mode

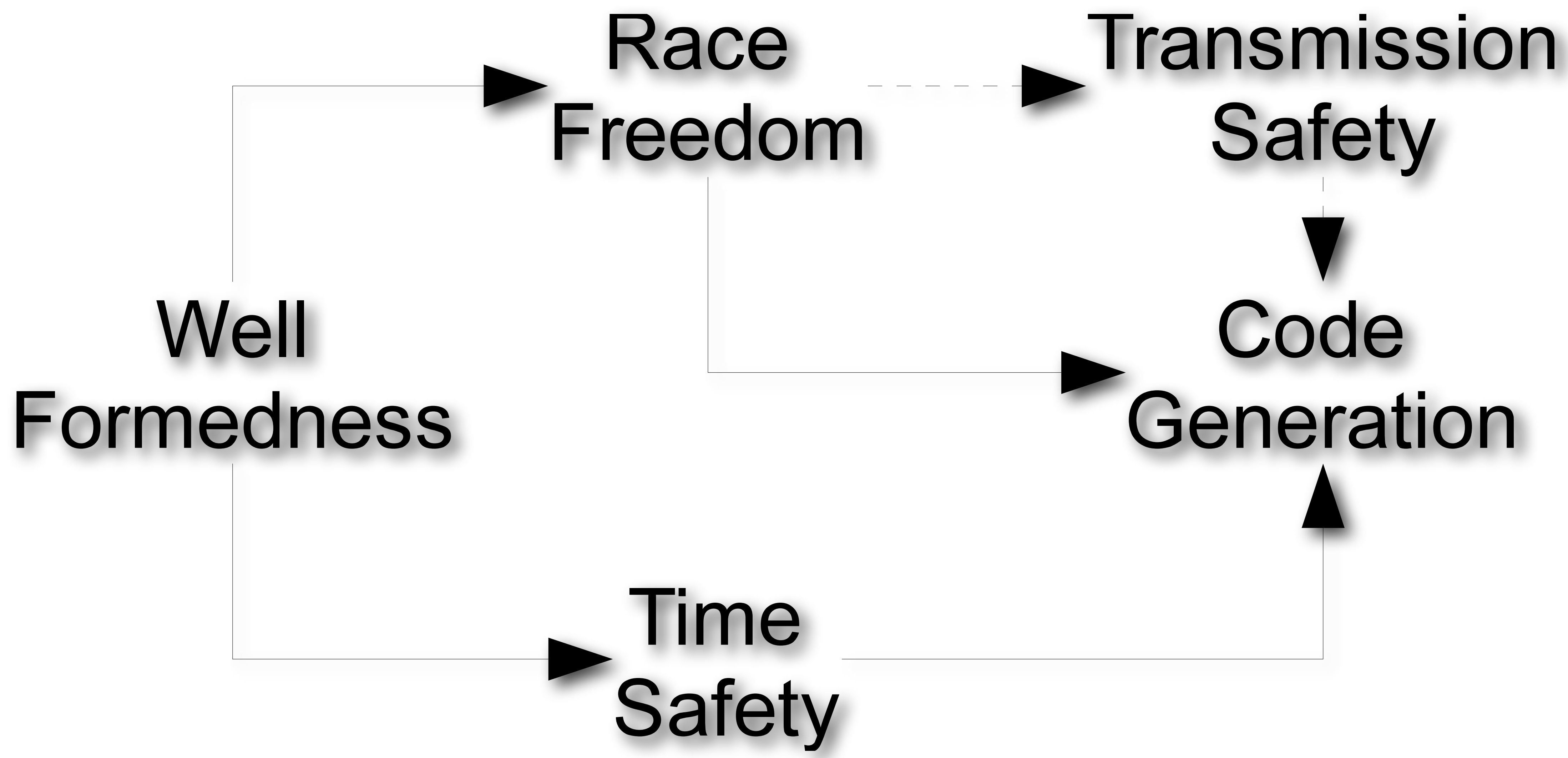


c_1
c_2
c_3
T_2
$(p_7,c_1,4)$
T_3
p_5
p_6
p_7
$(p_2,c_1,1)$
(p_3,p_6)
p_2
p_3
T_4
$(p_1,c_3,0)$
p_1
p_4
T_1
0
1
2
3
4
5
6
7
8
9
10

Program and Module



Compilation



Platform-Independent

- well-formedness
  - syntactic constraints (periods, task precedences, refinement)
- race freedom
  - at most <u>one</u> update of a given communicator per time instant

Platform-Dependent

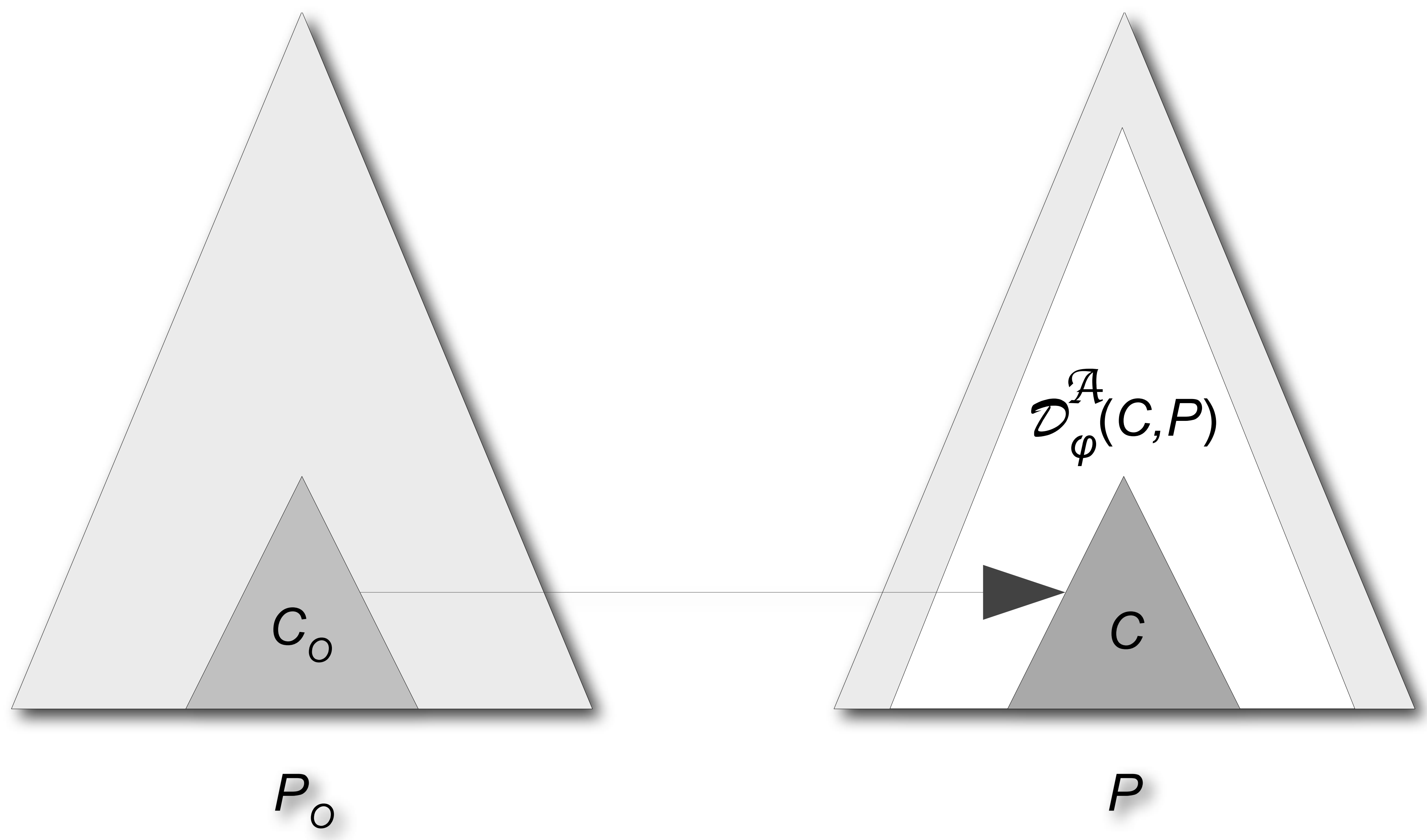
- time safety (computation schedulability)
  - each task invocation completes before the end of its logical execution time
- transmission safety (communication schedulability)
  - each communicator update is transmitted within one instance of the communicator's period

Well-formed, race-free, time-safe, and transmission-safe HTL programs

are

time-deterministic

Modularity

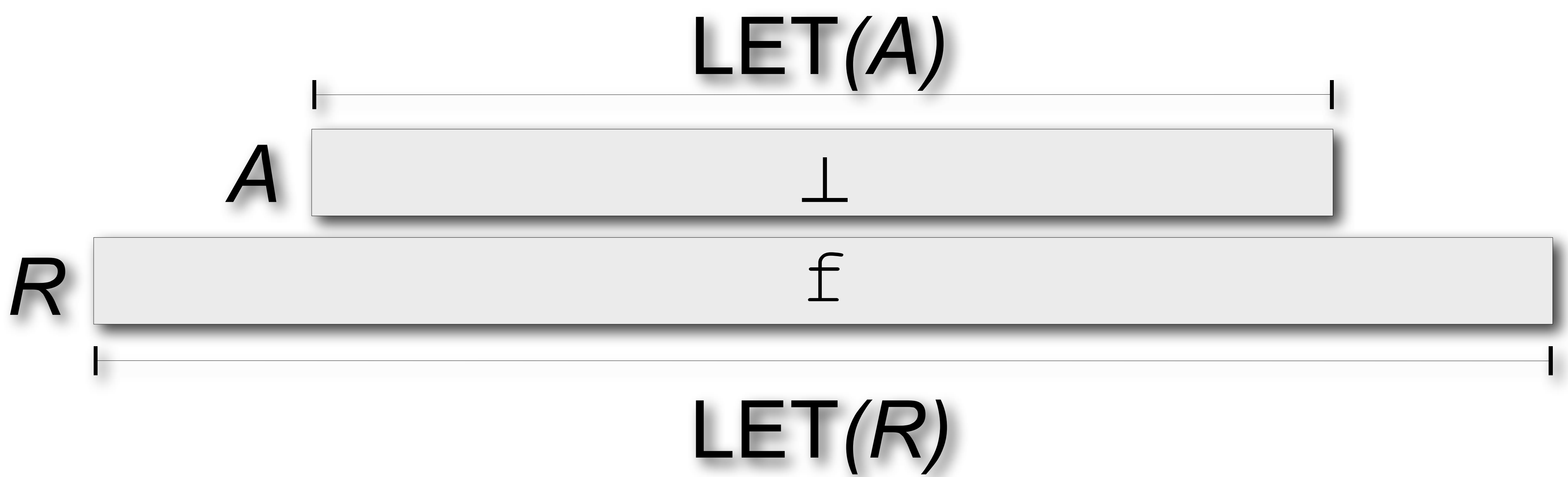


Complexity

φ	C	$\mathcal{D}_\varphi^{\mathcal{A}}(C,P)$	$\mathcal{C}_\varphi^{\mathcal{A}}(C,P)$	$\overline{\mathcal{C}}_\varphi^{\mathcal{A}}(P)$
Well-formedness	any	C	$n_{m\downarrow}^C \, n_T \, n_p$	$n_{m\downarrow}^P \, n_T \, n_p$
Race freedom	top	P	$n_{T\uparrow}^C \, n_w + n_M \, n_c$	$n_{T\uparrow}^P \, n_w + n_M \, n_c$
	ref.	C	1	
Transmission safety	any	C	1	n_c
Time safety	top	P	$(n_m \, \Delta_{max})^{n_M}$	$(n_m \, \Delta_{max})^{n_M}$
	ref.	C	1	
Code generation	any	C	$n_{m\downarrow}^C \, (n_T \, n_a + n_m)$	$n_{m\downarrow}^P \, (n_T \, n_a + n_m)$

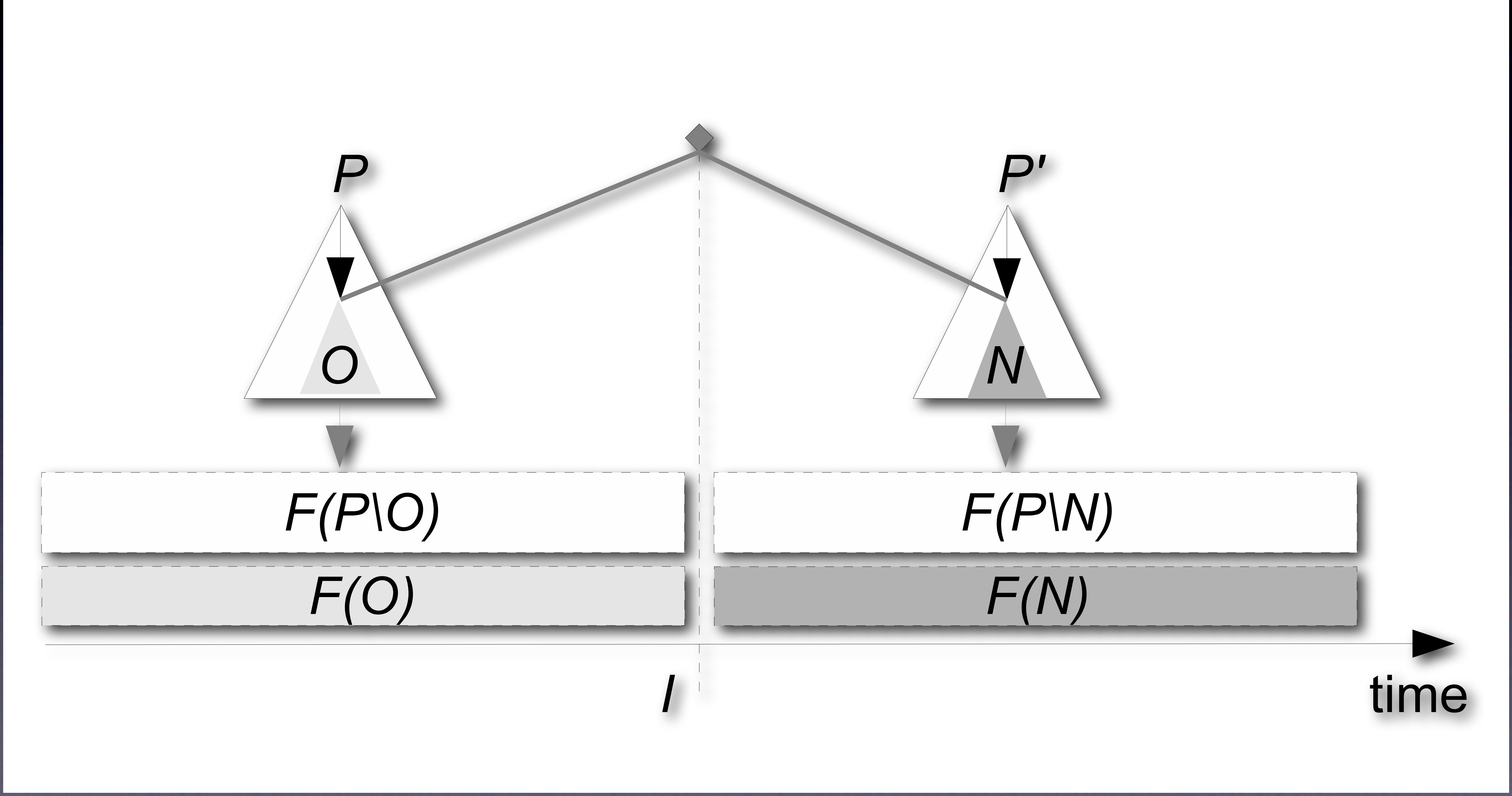
n_a	number of communicator accesses per task	n_c	number of communicators	n_M	number of modules per program
n_m	number of modes per module	n_p	number of ports per task	n_T	number of tasks per mode
n_w	number of communicator writes per task	$n_{m\downarrow}^C$	total number of modes in C	$n_{m\downarrow}^P$	total number of modes in P
$n_{T\uparrow}^C$	number of top-level tasks in C	$n_{T\uparrow}^P$	number of top-level tasks in P	$\Delta_{\max}$	maximal value of mode periods

Refinement

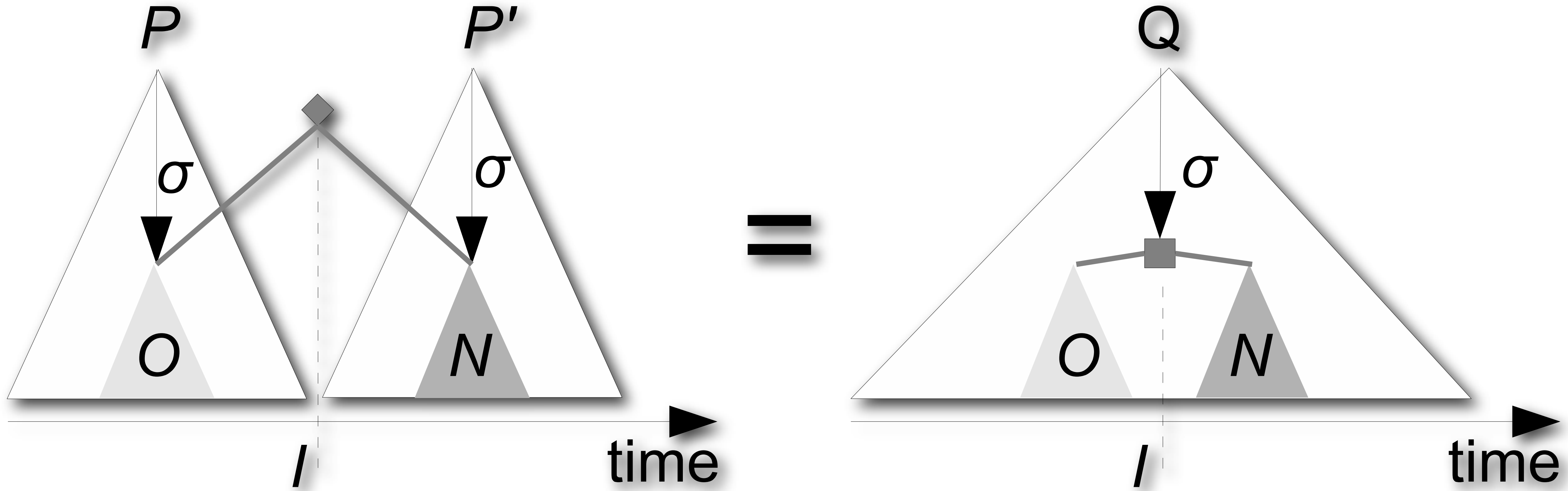


A concrete HTL program
that refines
a time-safe, abstract HTL program
is also
time-safe

Runtime Patching



Preserving Semantics




Thank you