A Giotto-based Helicopter Control System

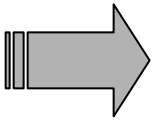
Marco A.A. Sanvido[1], Christoph M. Kirsch[1],
Thomas A. Henzinger[1], Wolfgang Pree[2]

[1]University of California, Berkeley
[2]University of Salzburg, Austria

Second International Workshop on Embedded Software
EMSOFT 2002
7 Oct. 2002, Grenoble, FR

Project Objectives

1. Demonstrate feasibility of Giotto
   - Implementing a real complex control problem
2. Demonstrate benefits of Giotto
   - Comparing it with an existing solution



Re-implementing the autopilot control system developed at ETH Zürich (project OLGA)

The Helicopter

The Helicopter

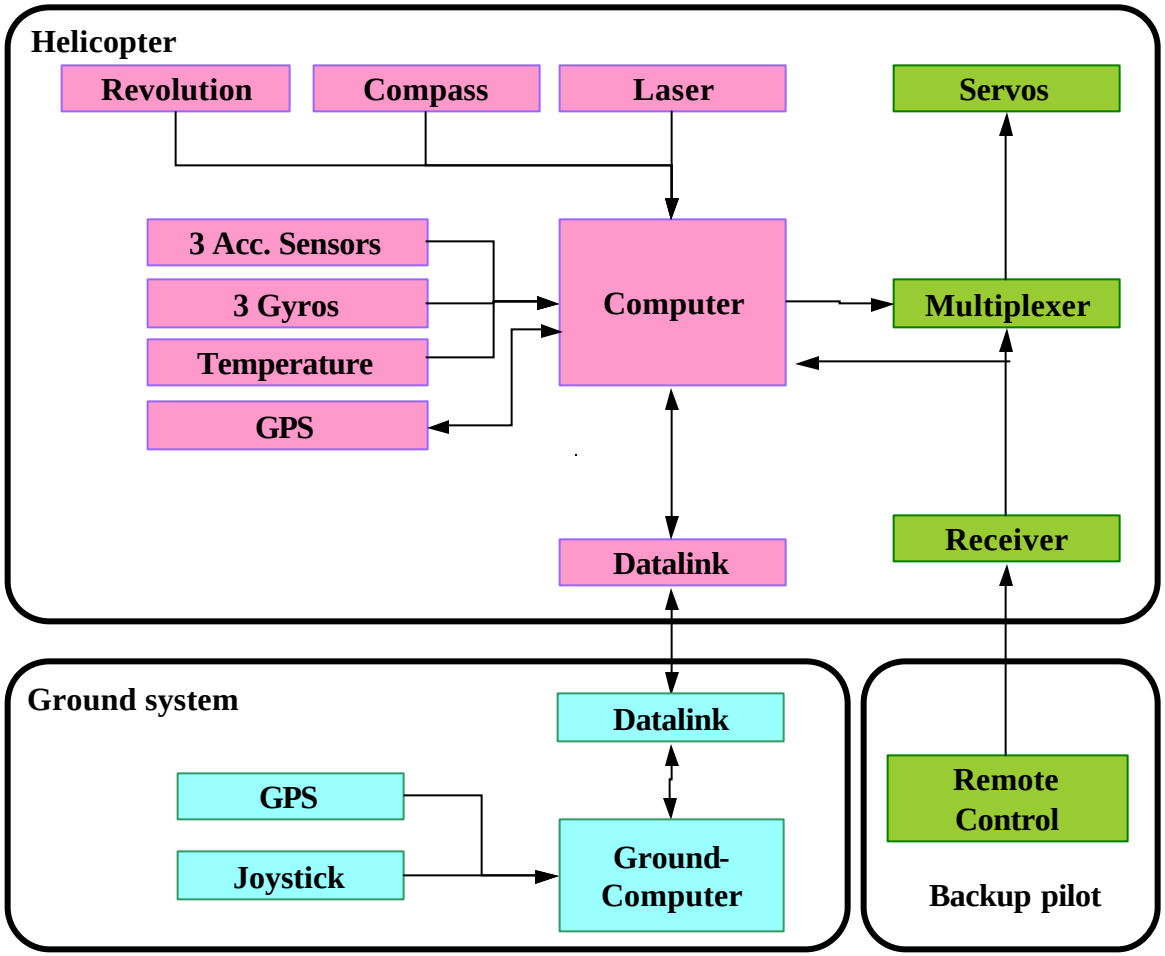
About 2m

Swiss Feral Institute of Technology interdisciplinary project (1997-2001):

- Helicopter Model (Hunziker AG)
- Control (LQR)
- Navigation (Extended Kalmann Filter)
- Computer System (StrongARM)
- RTOS (HelyOS)

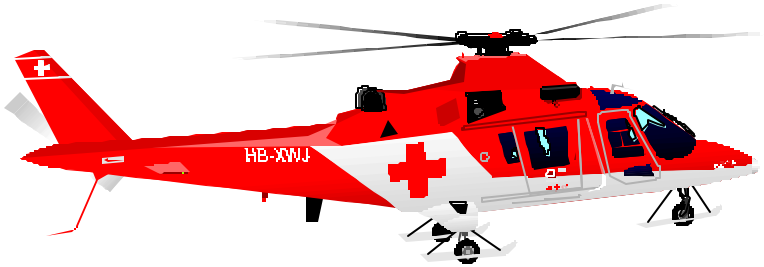
2001 weControl GmbH, makes&produces wePilot1000

The Platform



The Controller

- Data Visualization
- Flight Monitoring
- Flight Planning
- Flight Commands



- Sensor Data Evaluation
- Flight Control
- Trajectory
- Flight Data Recording

The Legacy Control Software Implementation

1. Modular software implementation (Oberon language)
2. Functionality and timing was mixed
3. Functionality (i.e., navigation/control) was hand-coded from Simulink Model

Code:

Set of tasks with different priorities, communicating via shared memory and message passing.

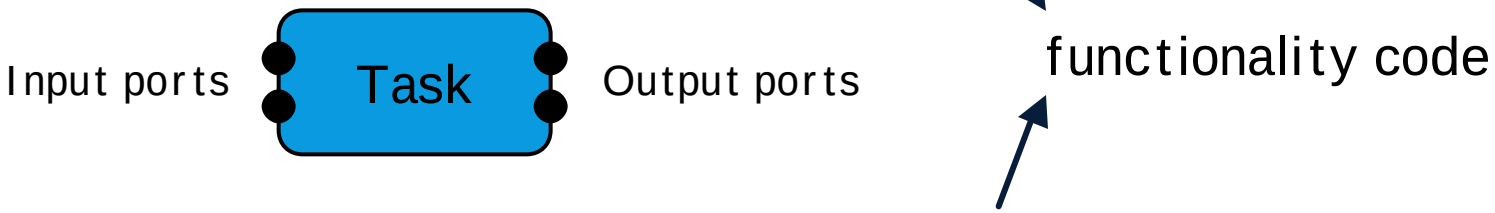
Giotto & E code

why Giotto?

The Giotto Programmer's Model

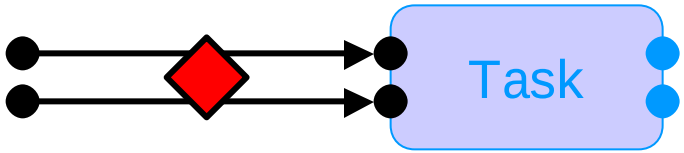
Given: 1. Units of scheduled host code (application-level tasks).

e.g. control law computation



2. Units of synchronous host code (system-level drivers).

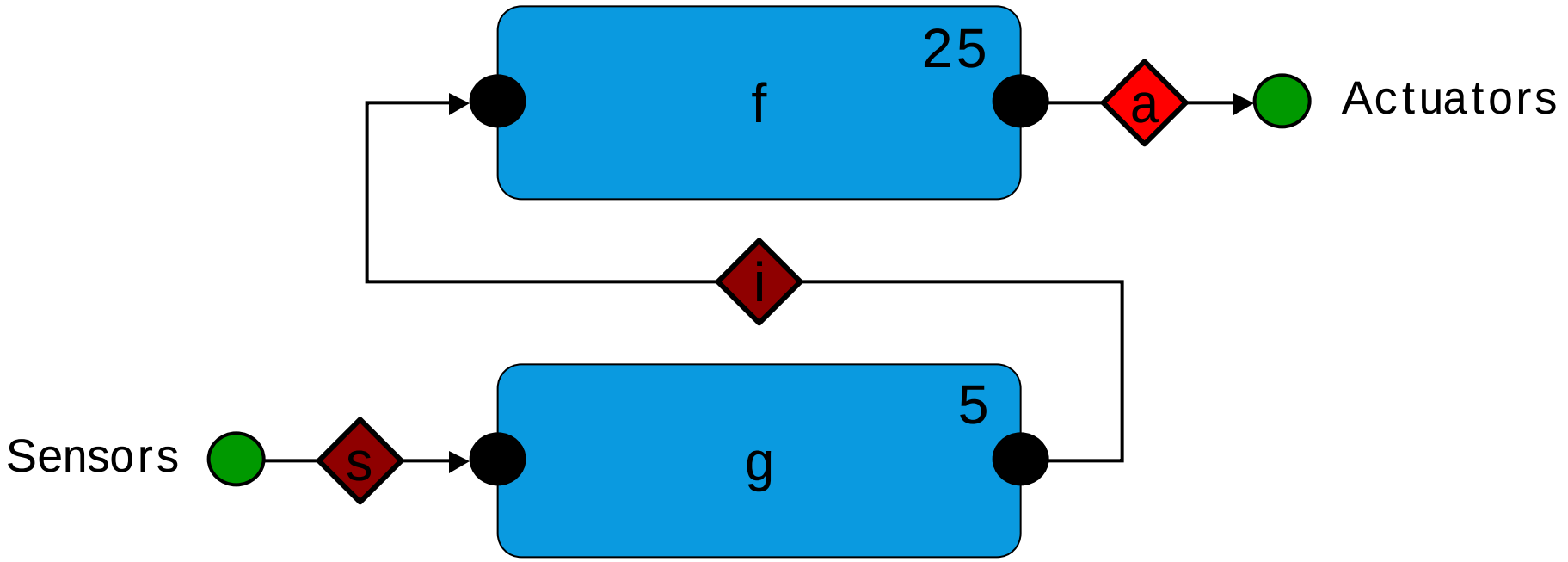
e.g. device drivers



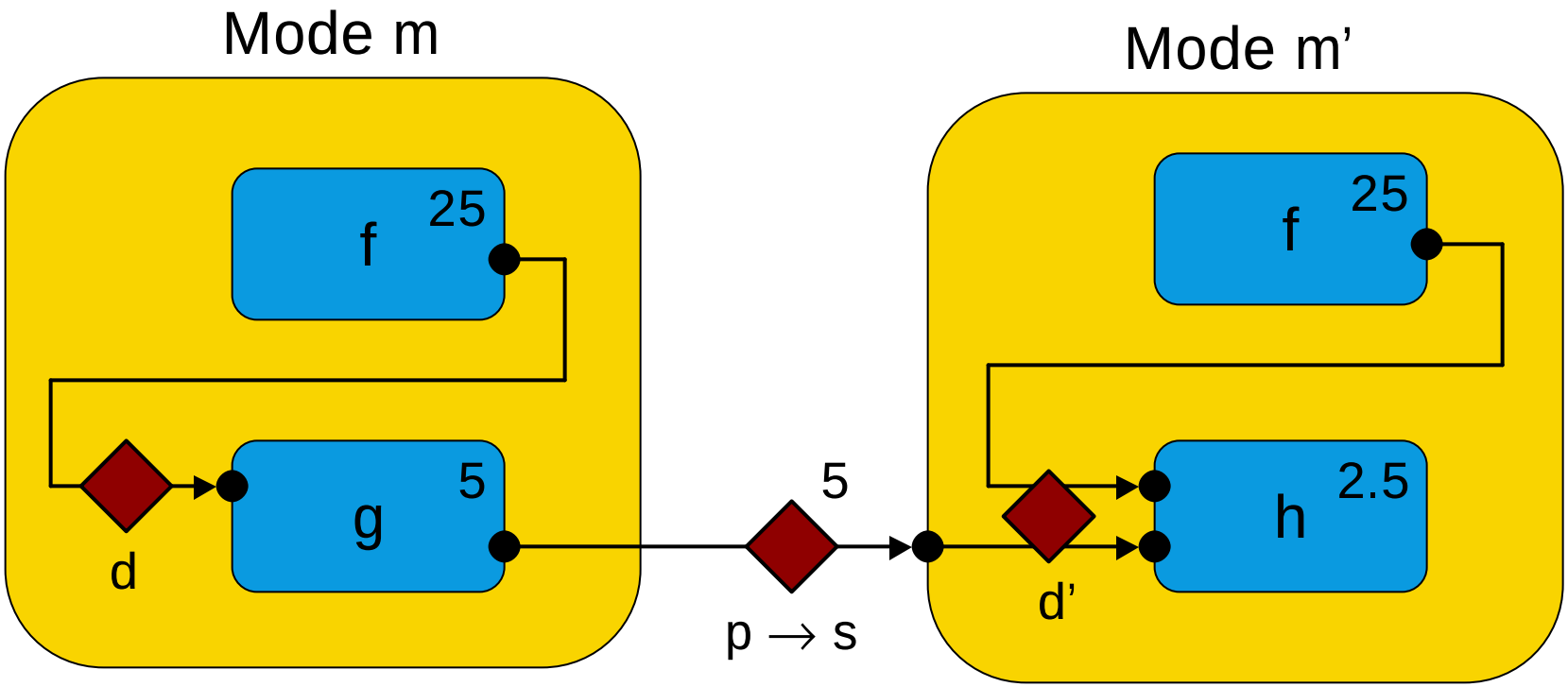
3. Real-time requirements and data flow between tasks.

Giotto: Glue code that calls 1. and 2. in order to realize 3.

Tasks, Sensors, and Actuators Data Flow



Mode Switch



Platform-independent Software Model

Host code
e.g. C, Oberon

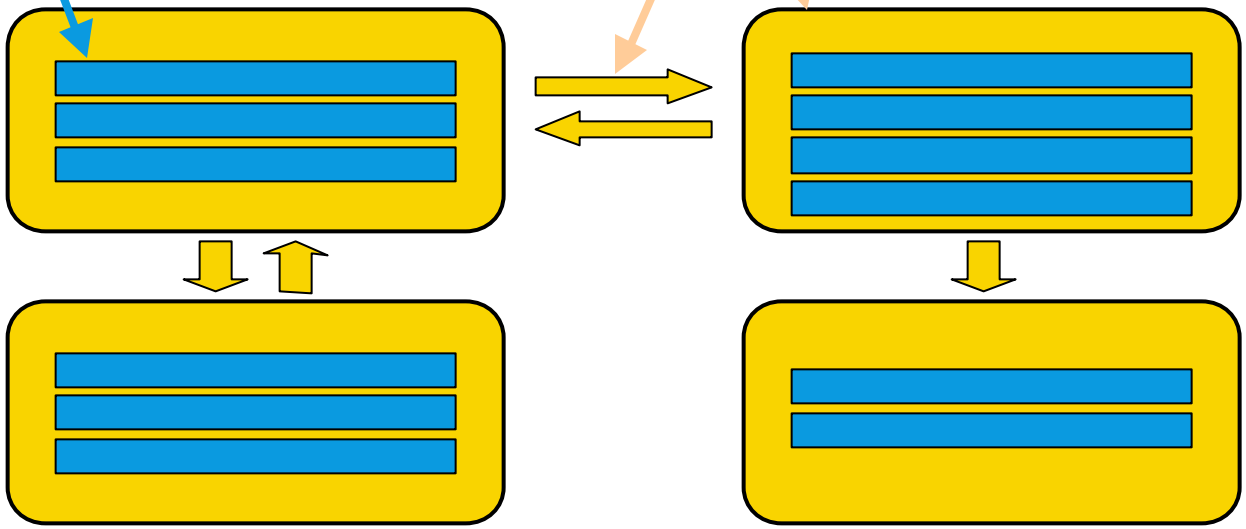
Functionality.

-No time.
-Sequential.

Glue code
Giotto

Timing and interaction.

-Environment time, not platform time.
-Concurrency, not distribution.



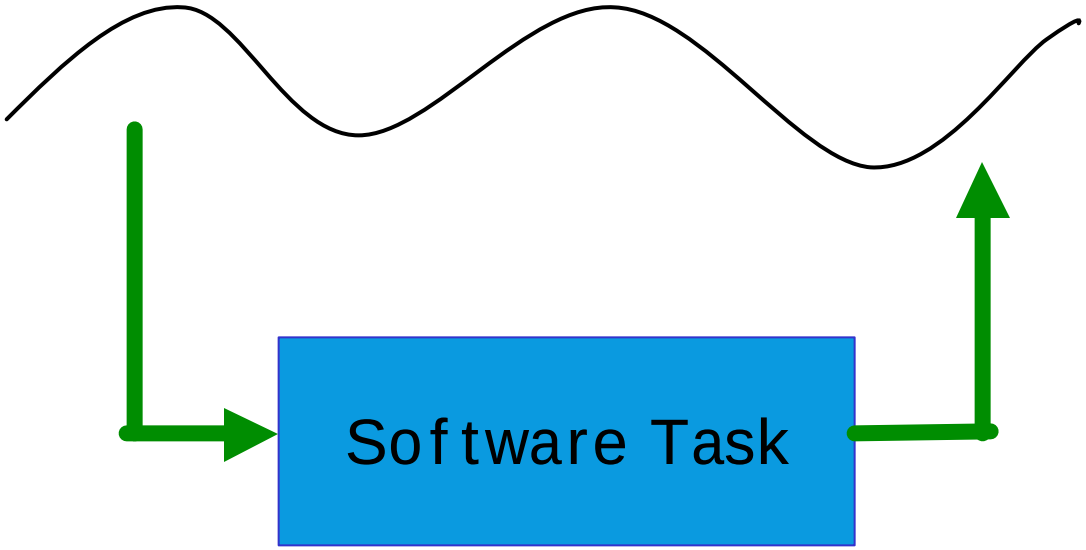
This kind of software is understood:

Host code may (sometimes) be generated automatically.

The software complexity lies in the glue code (minimize jitter!) :

Giotto enables requirements-driven rather than platform-driven glue-code programming.

The FLET Assumption (Fixed Logical Execution Time)

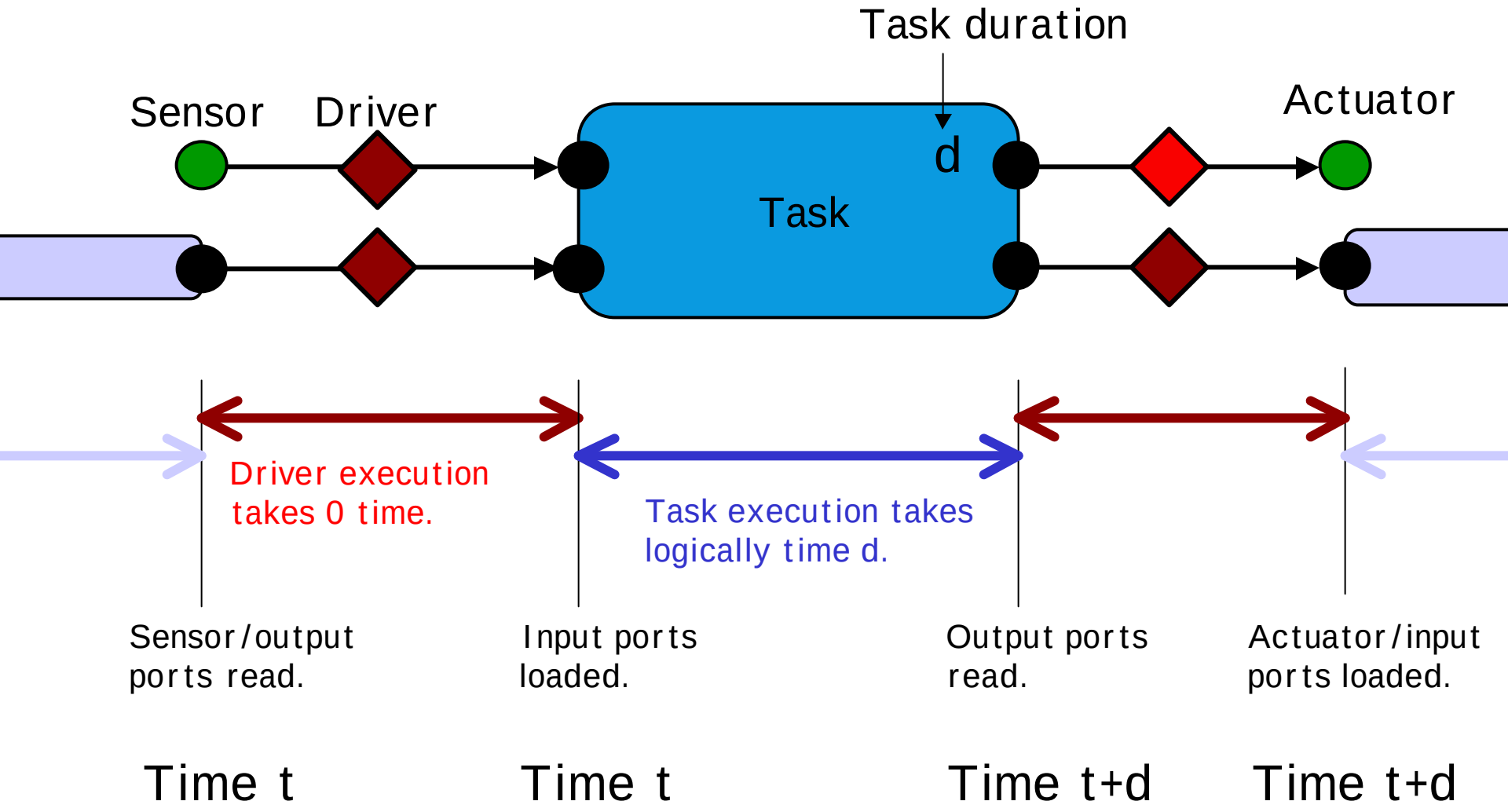


read sensor input at time *t*

d>0 is the task's "logical execution time"

write actuator output at time *t*+*d*, for fixed *d*

The FLET Assumption
(Fixed Logical Execution Time)



The FLET Assumption

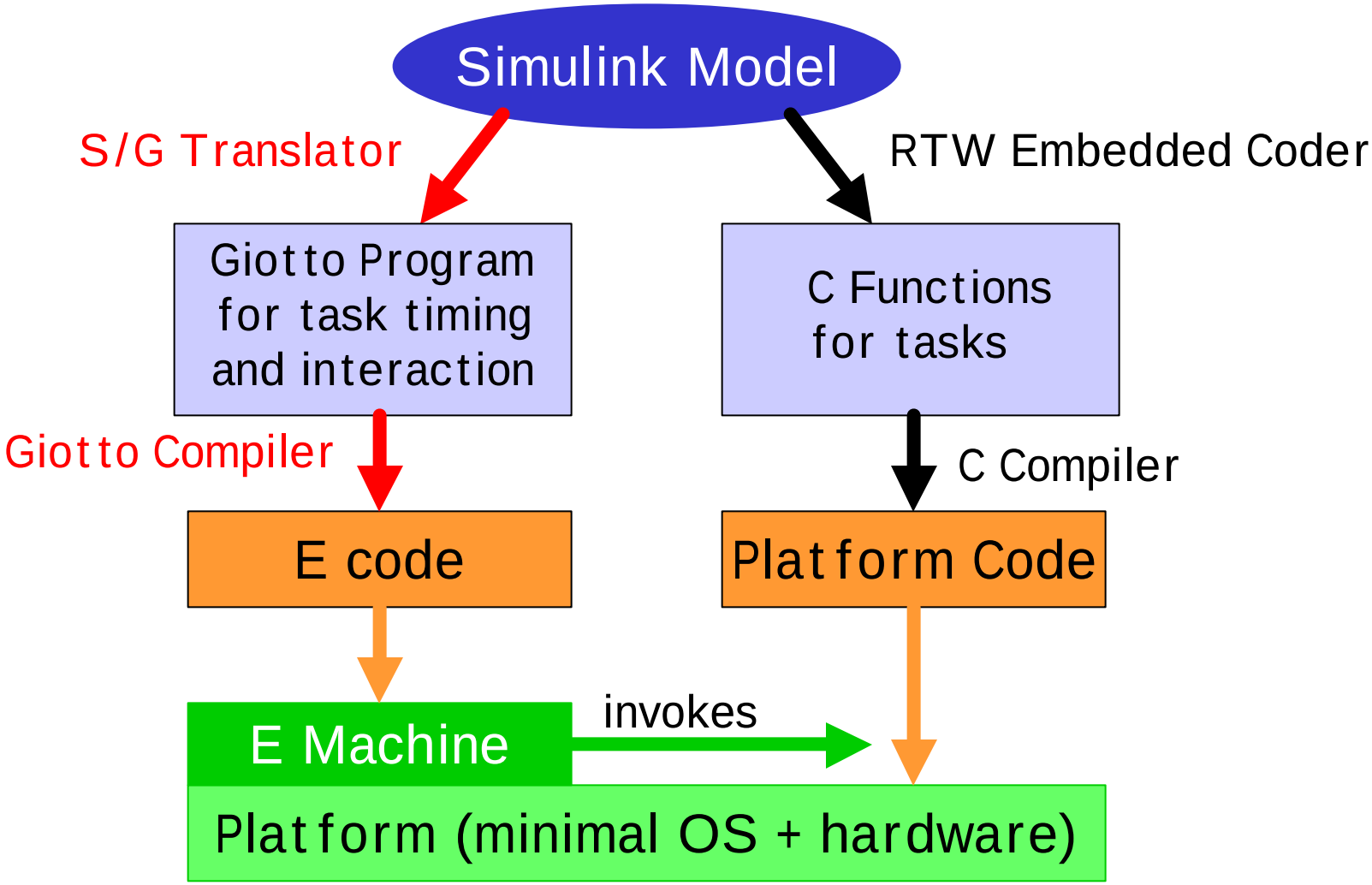
Advantages of the FLET:

- predictable timing and value behavior (environment determined programs: no race conditions, minimal jitter)
- portable, composable code (as long as the platform offers sufficient performance)

Disadvantage of the FLET:

- tasks don't always use latest available data (a small price to pay, e.g. model helicopter)

The Giotto Tool Chain



The Embedded Machine

-a virtual machine that mediates the interaction of physical processes (sensors and actuators) and software processes (tasks and drivers) in real time

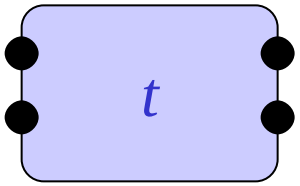
-the Giotto compiler can be retargeted to a new platform by porting the Embedded Machine

The Embedded Machine: Three Instructions

Call driver:
call(*d*)

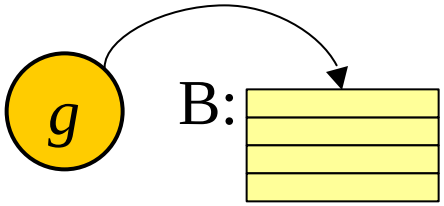
Execute driver *d* now.

Schedule task:
schedule(*t*)



Hand task *t* over to the system scheduler (RTOS).

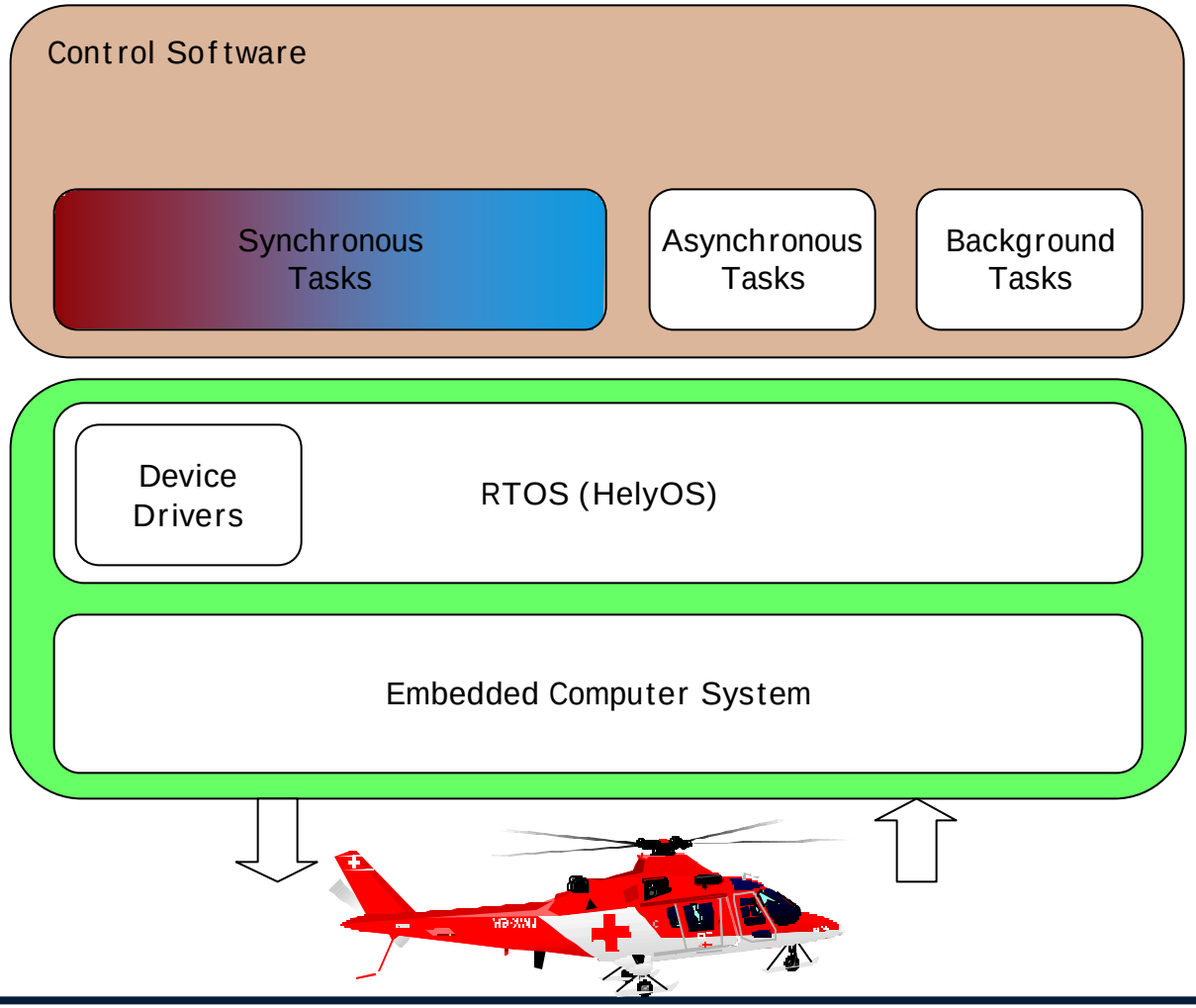
Enable trigger:
future(*g*,B:)



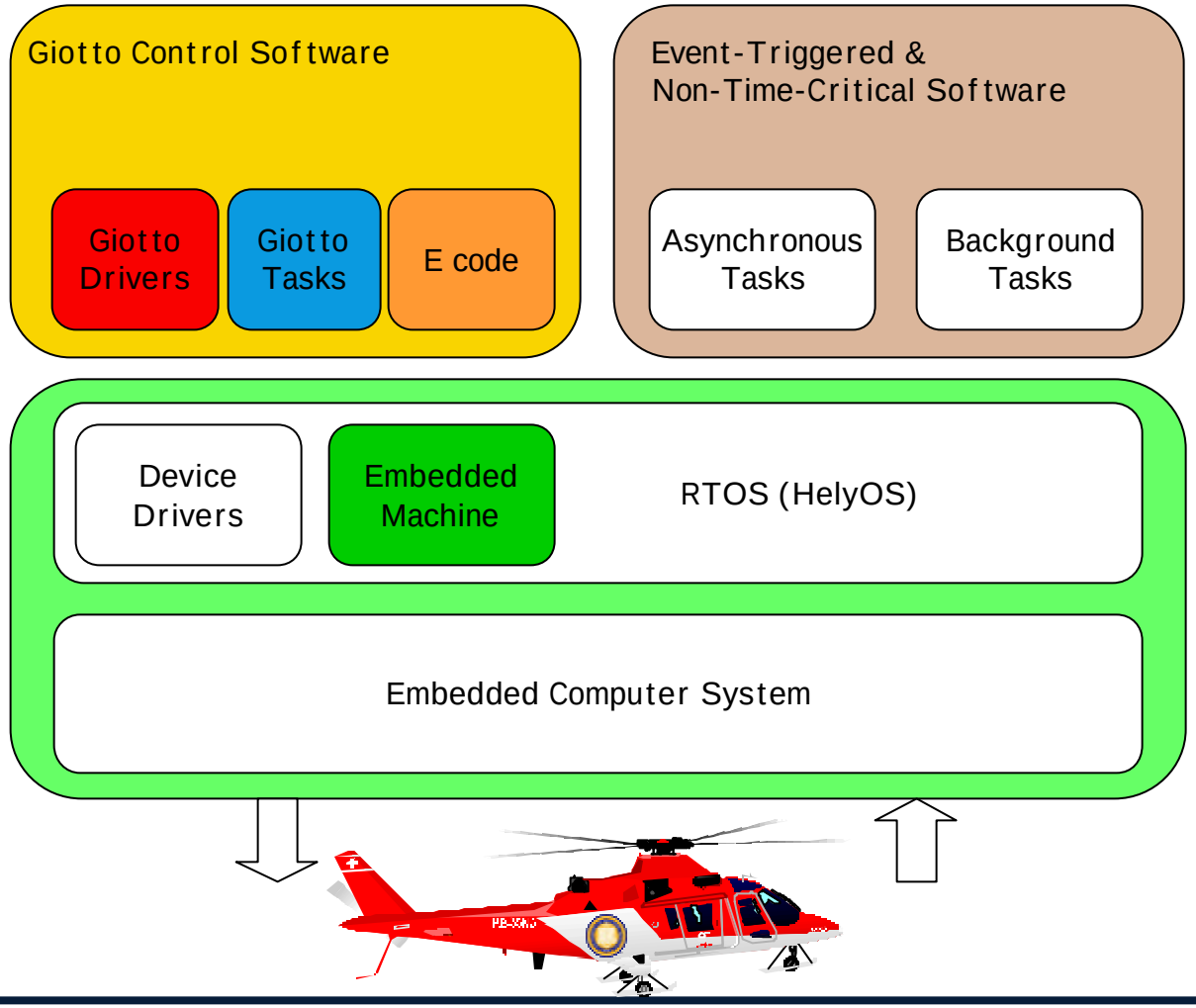
Have code B executed as soon as trigger *g* becomes true.

The Re-Implementation

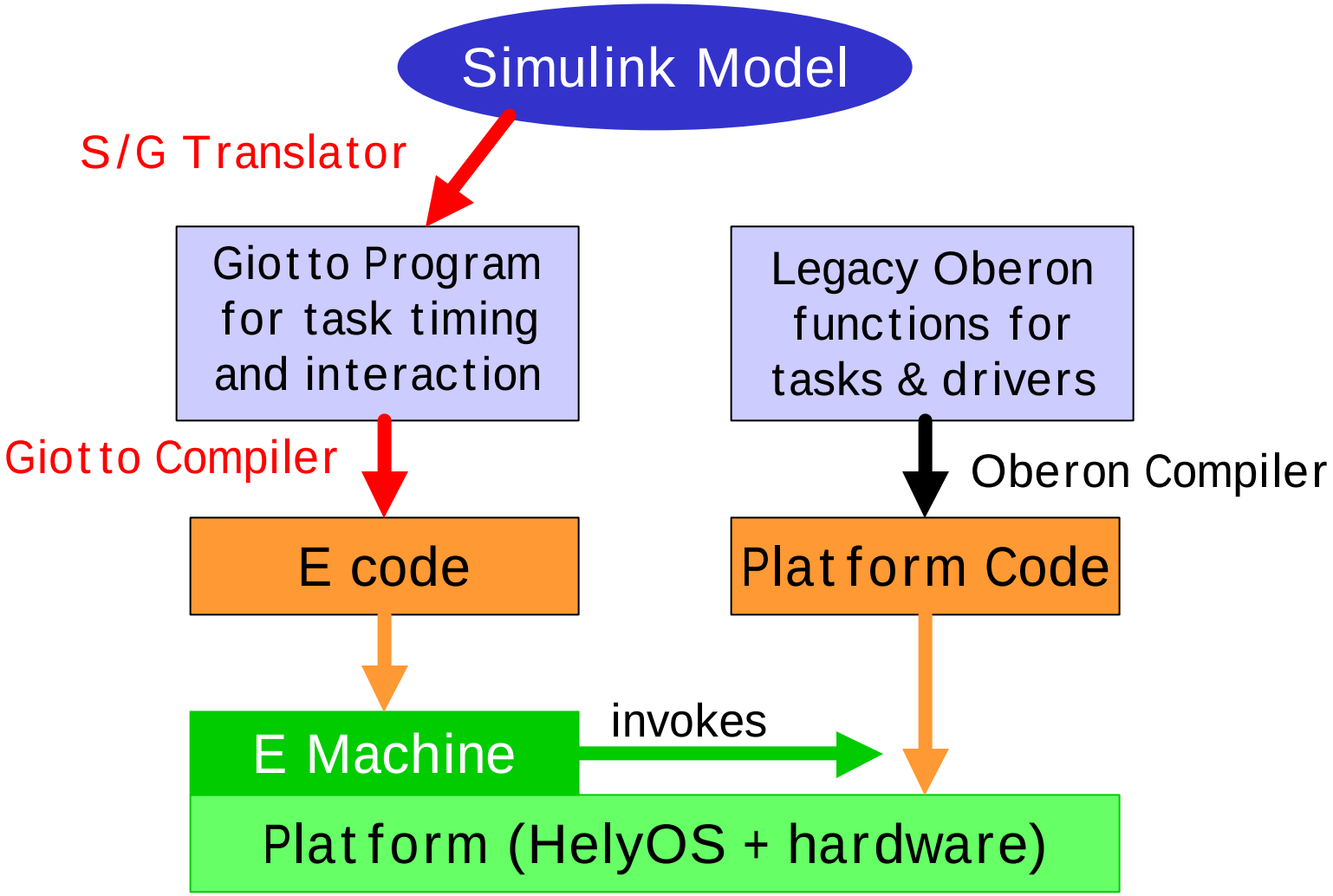
The Legacy Implementation



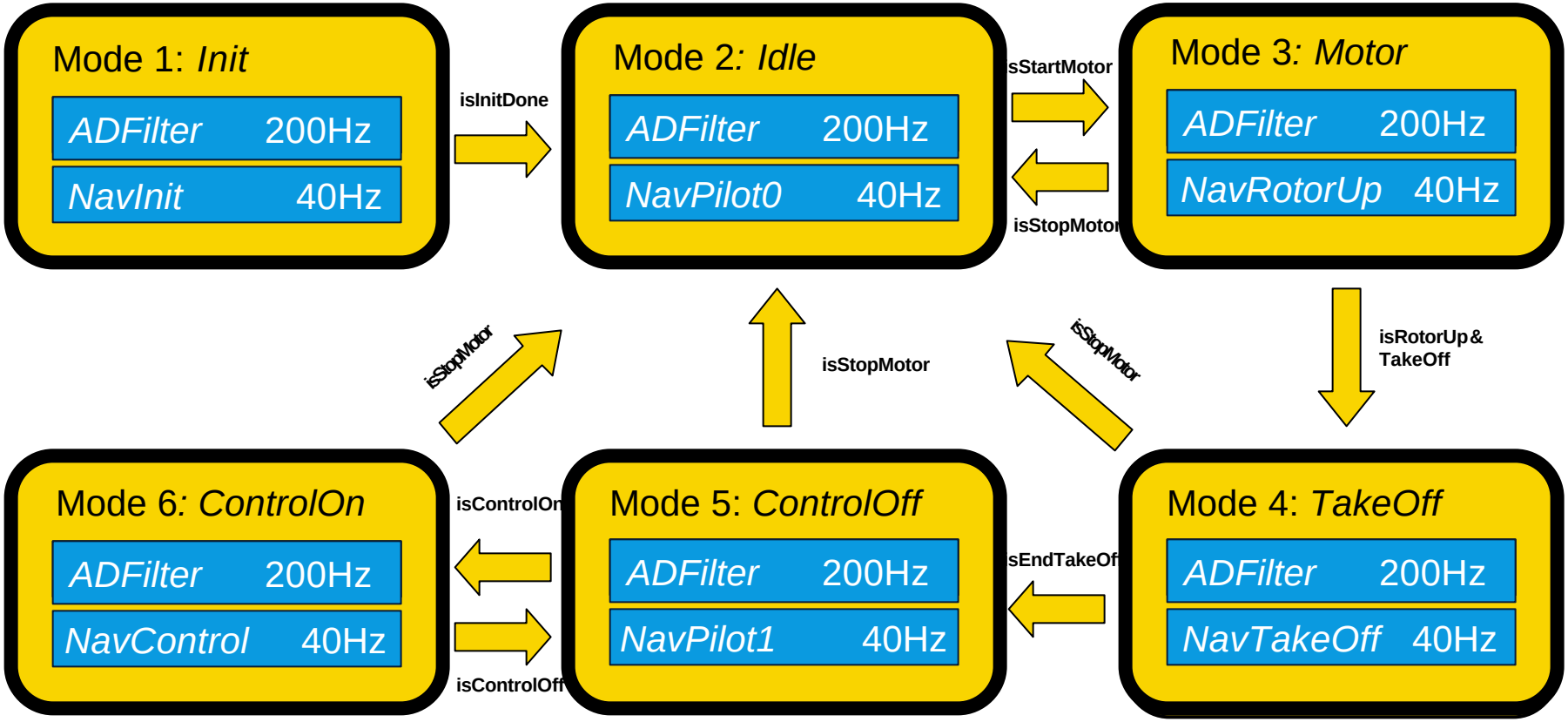
The Re-Implementation



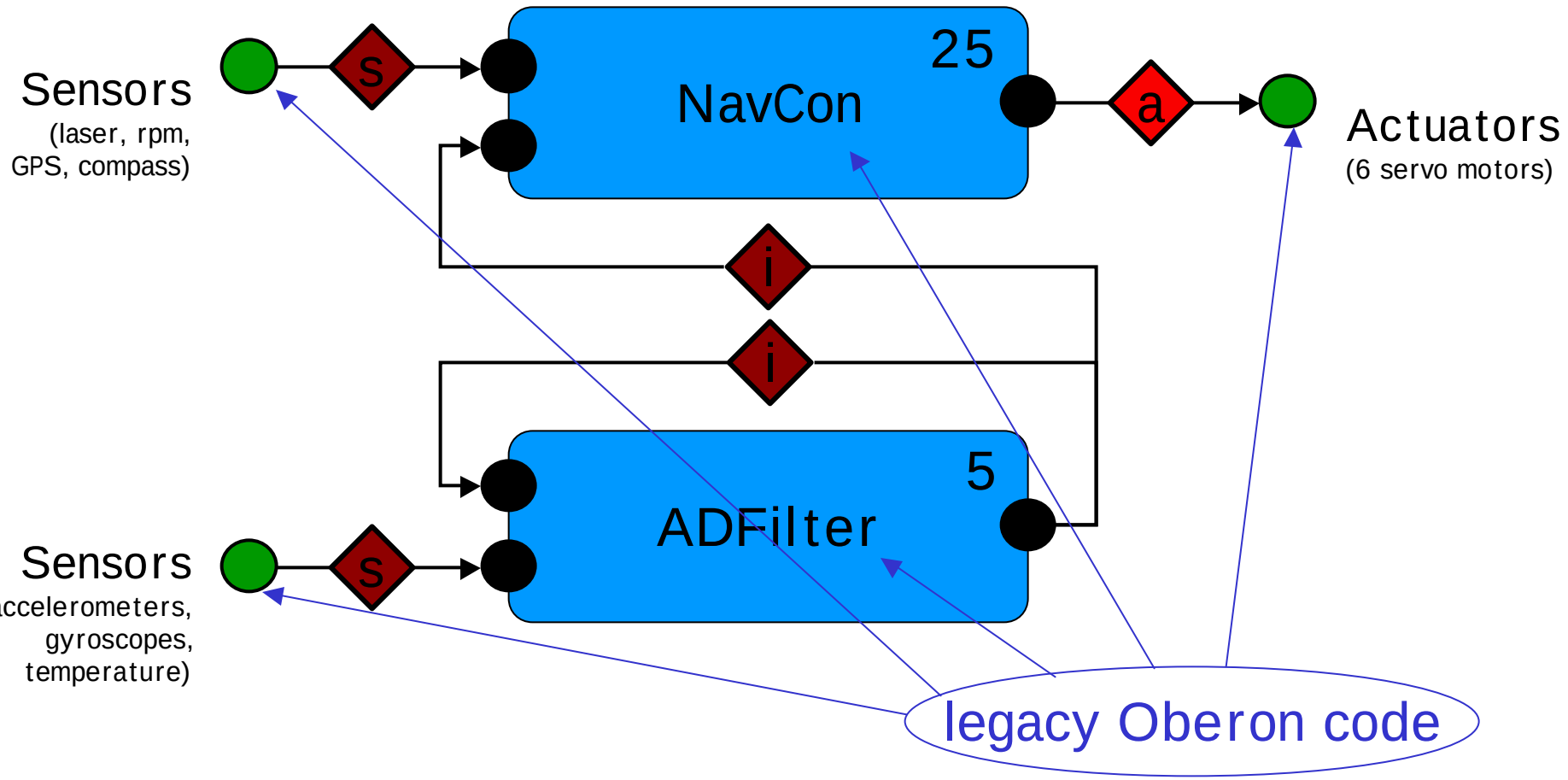
The Giotto Tool Chain



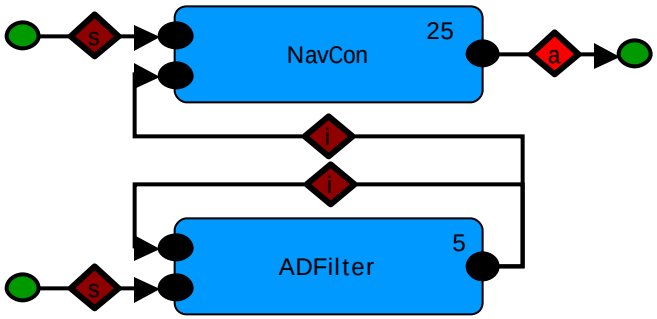
The Controller Modes



Data Flow in *ControlOn* mode

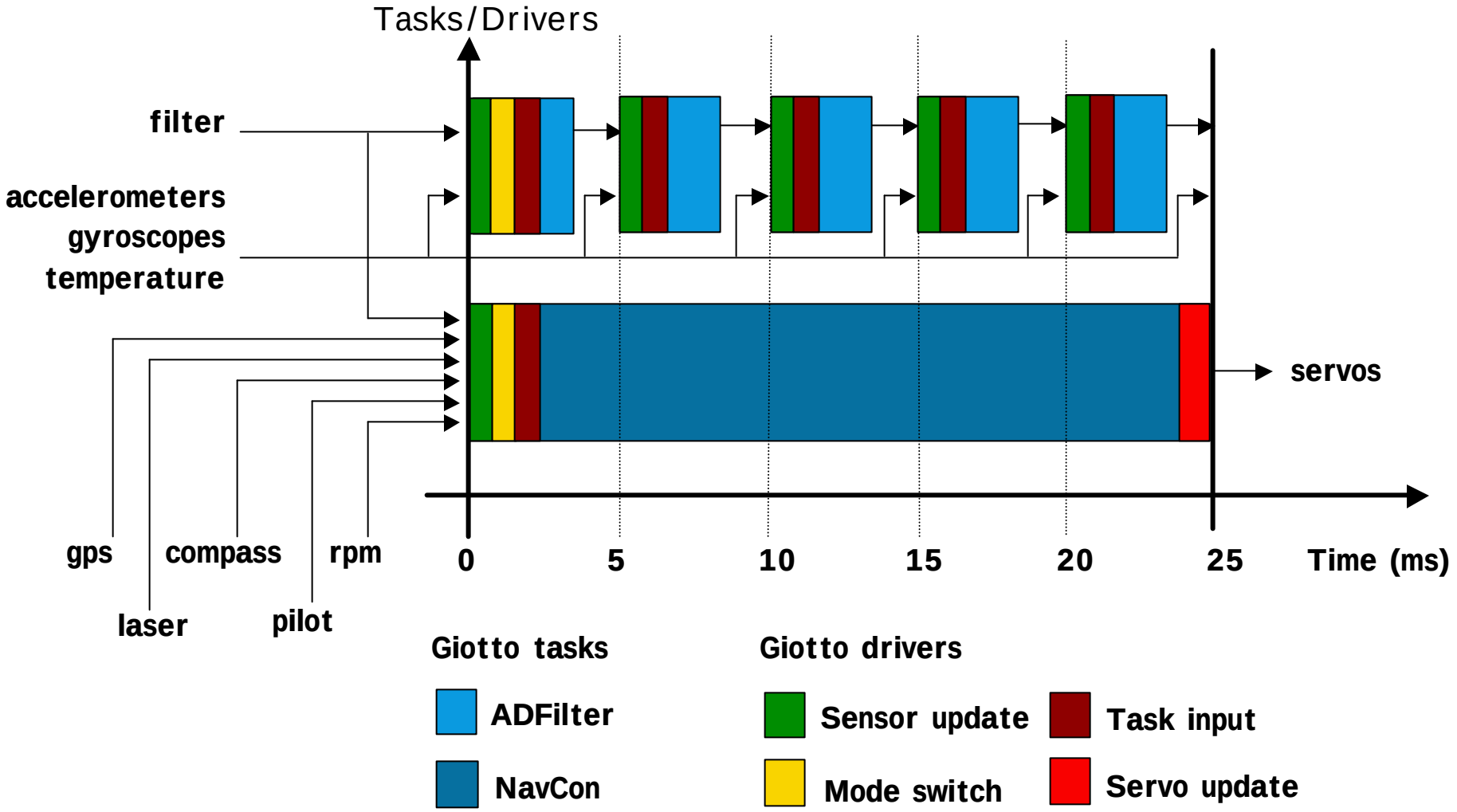


Helicopter Software: Giotto Syntax (Functionality)

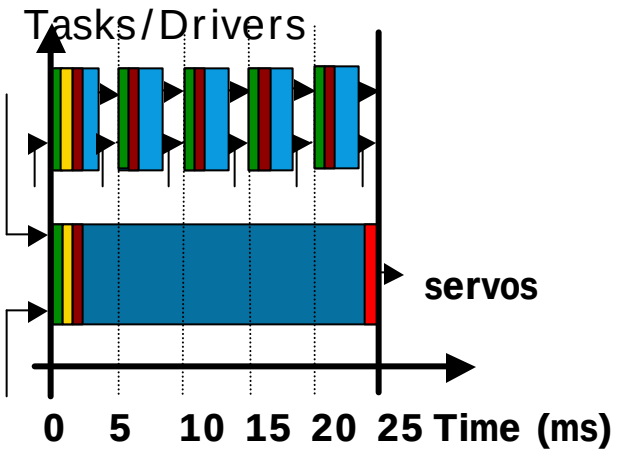



```
sensor    gps_type      GPS     uses gps_device;
          analog_type   Analog  uses analog_device;

actuator servo_type Servo := servo_init
          uses servo_device ;

output

   filter_type  ADFilterOutput := filter_init ;

   servo_type NavConOutput := servo_init ;

driver sensing (GPS) output (gps_type gps)
{ gps_pre_processing ( GPS, gps ) }


task NavCon (filter_type filter, gps_type gps)
          output (NavOutput)

{ navcon_code ( filter, gps, NavOutput ) }

...
```

The FLET Assumption (*ControlOn* mode)

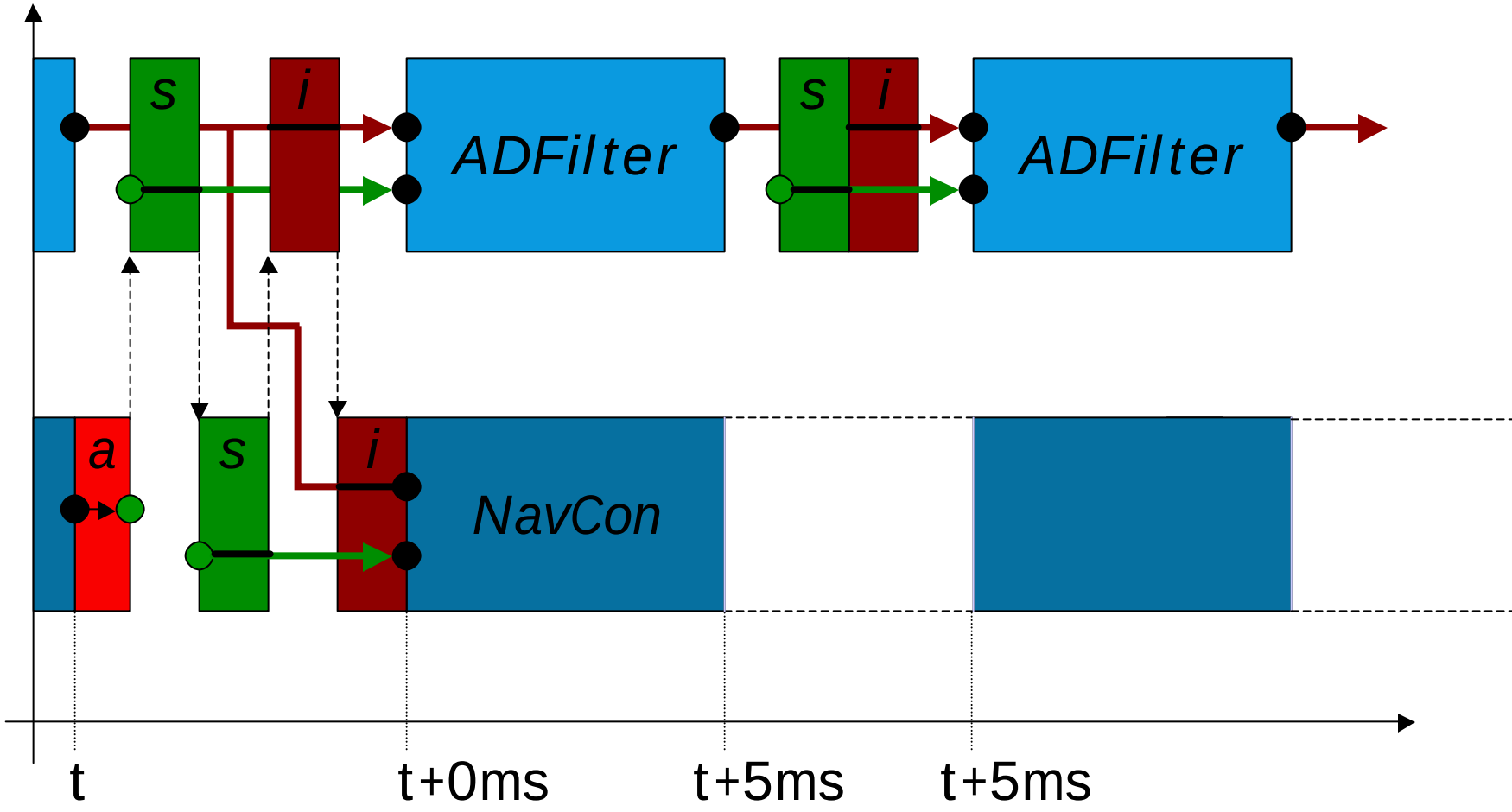


Helicopter Software: Giotto Syntax (Timing)

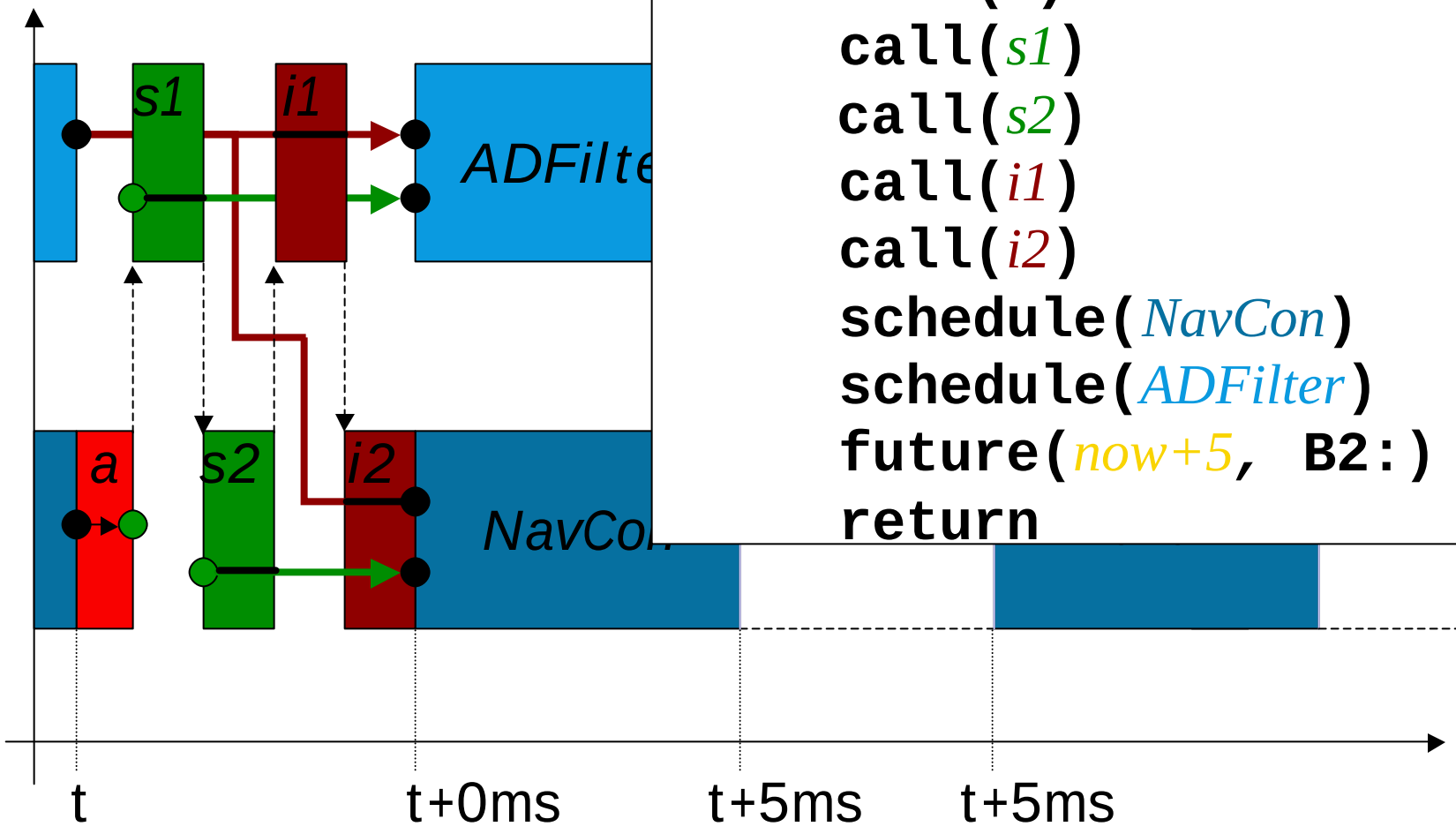



```
...
mode ControlOn() period 25ms
  {
      actfreq 1 do Actuator ( actuating ) ;

      taskfreq 5 do ADFilter ( input ) ;
      taskfreq 1 do NavCon ( sensing ) ;
  }
...
```

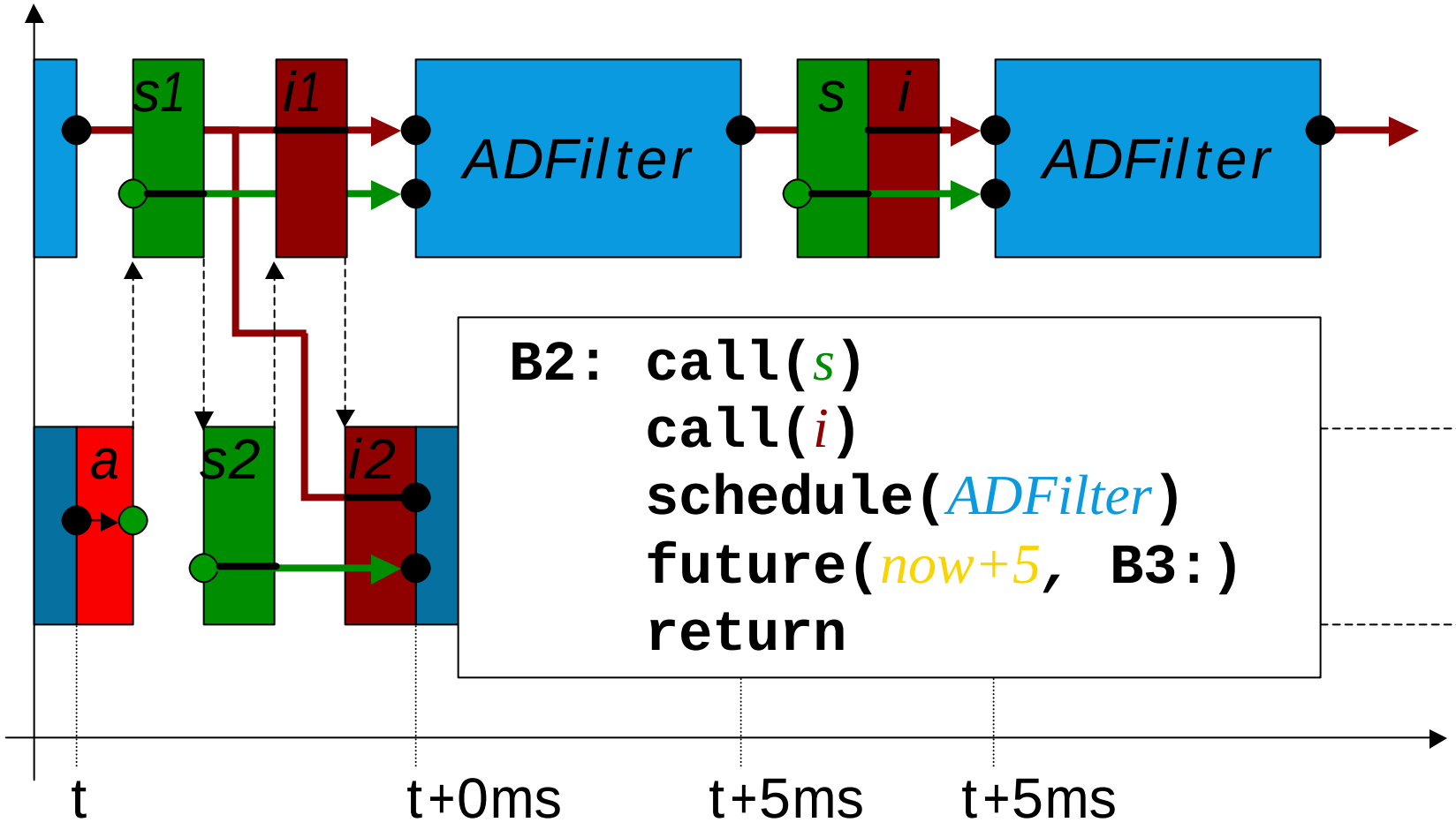
Generated E code (*ControlOn* mode)



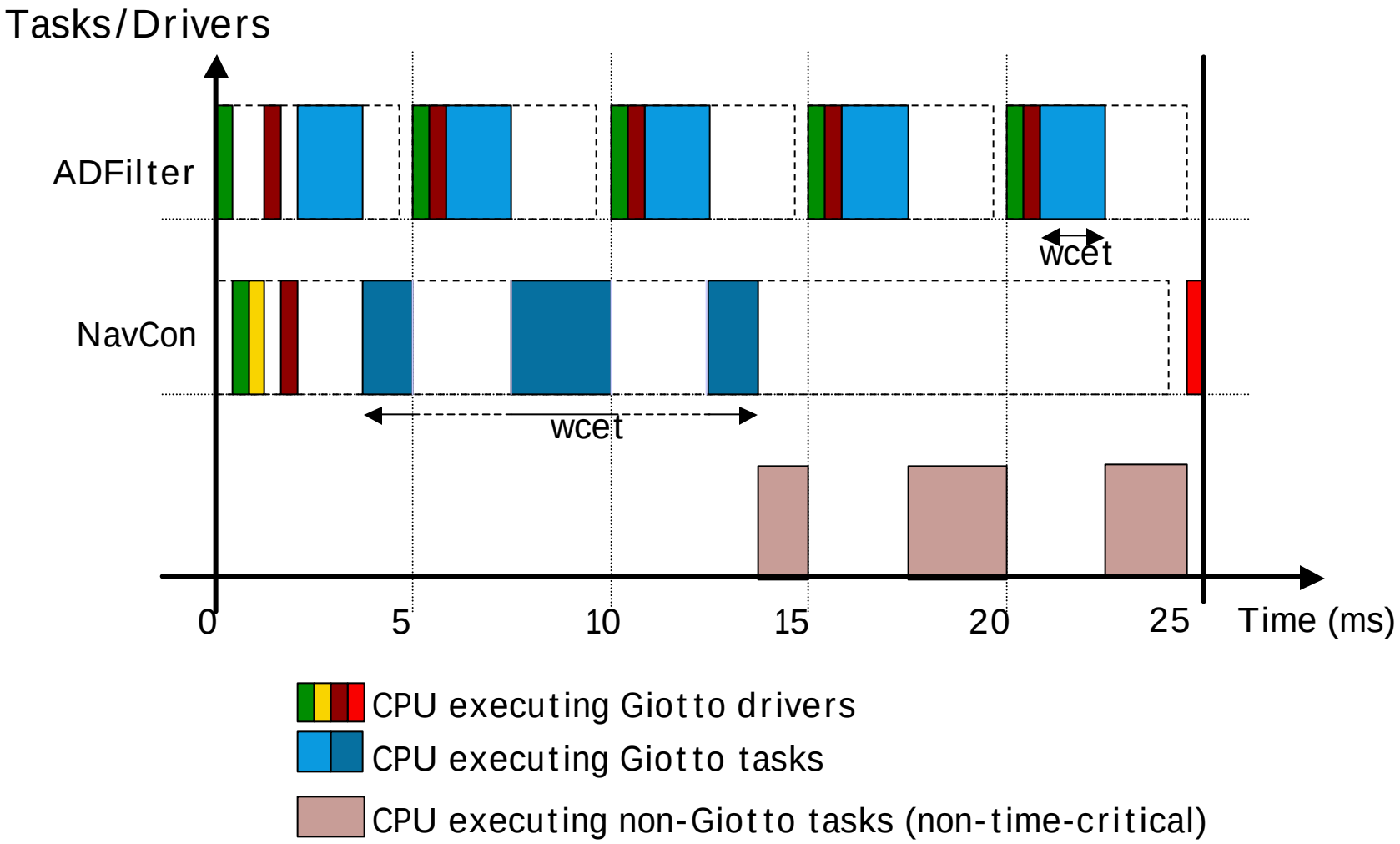
Generated E code (*ControlOn* mode)



Generated E code (*ControlOn* mode)



Execution of *ControlOn* Mode
(Rate Monotonic Scheduler)



Conclusions

- Giotto is applicable to real-complex and high-performance control systems

- Timing is correct-by-construction
- Functionality code is domain-dependent and platform-independent (wcet)
- Tasks are composable

- Giotto introduces computational overhead (~2% /5ms in the helicopter example)

- Simple to implement (2 man/month)

End

(software) www.eecs.berkeley.edu/~fresco/giotto

(helicopter) www.heli.ethz.ch