Real Time Systems

Automatic Drowsiness Detection System

Automatic Drowsiness Detection System

The goal of this system is to detect driver sleepiness (drowsiness).

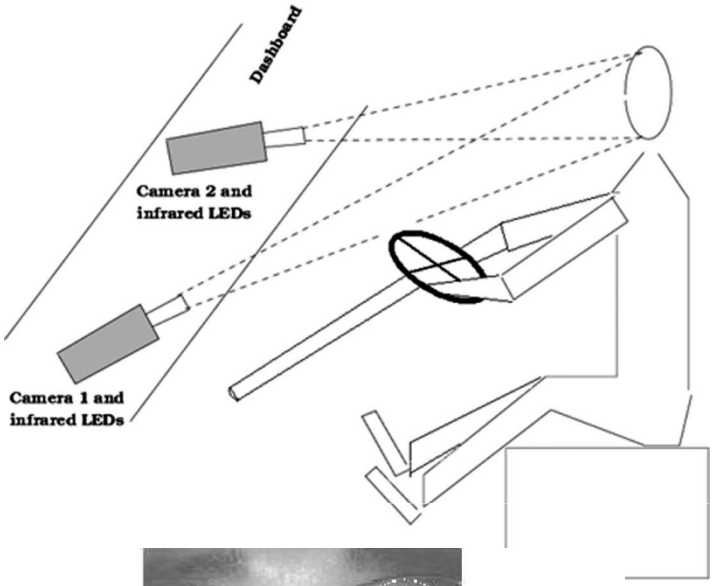
The system will activate some devices to make the driver react and avoid an accident.

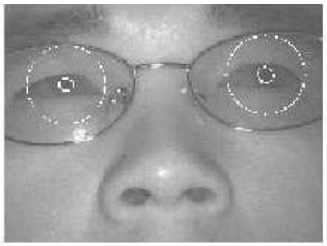
In order to detect the the fatigue of the driver, the systems will use:

- Eyes aperture
- Heart pulse rate
- Electroencephalogram
- Car deviation
- Head position

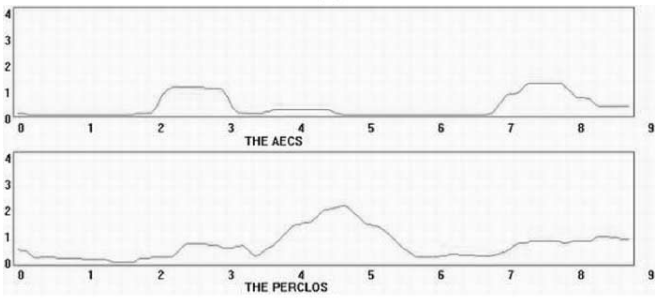
Input Devices

Eyes aperture





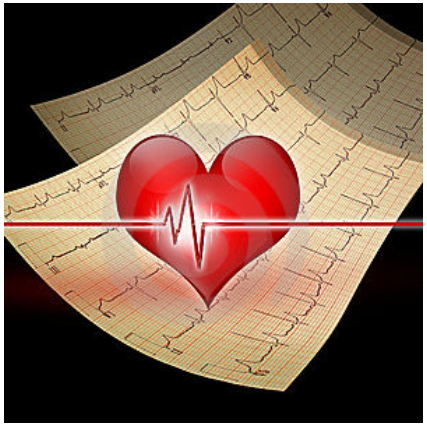
(a)



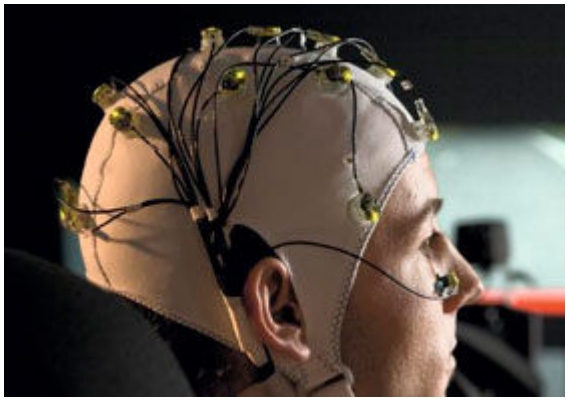
(b)

Input Devices

Heart pulse rate



Electroencephalogram (EEG)



Output devices

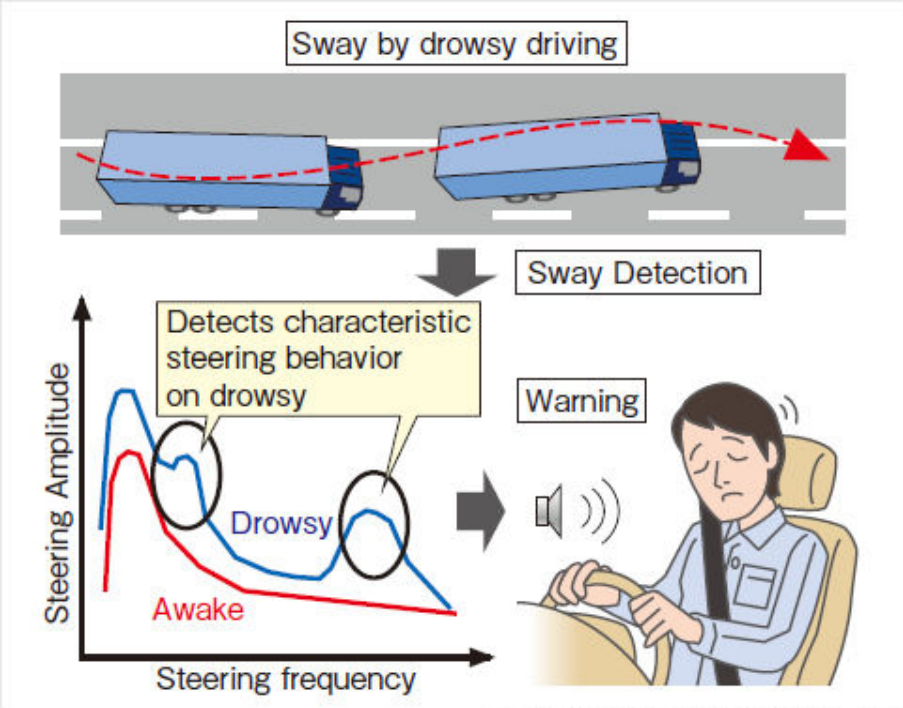
Light

Alarm

Display

Extensions

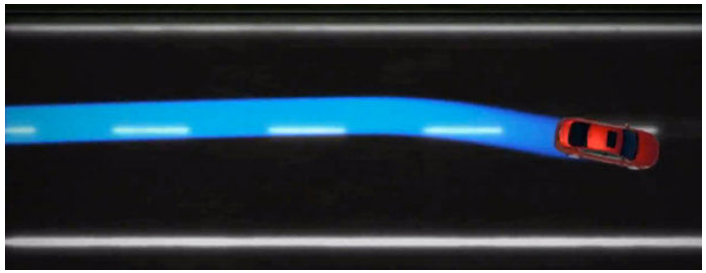
Steering behaviour



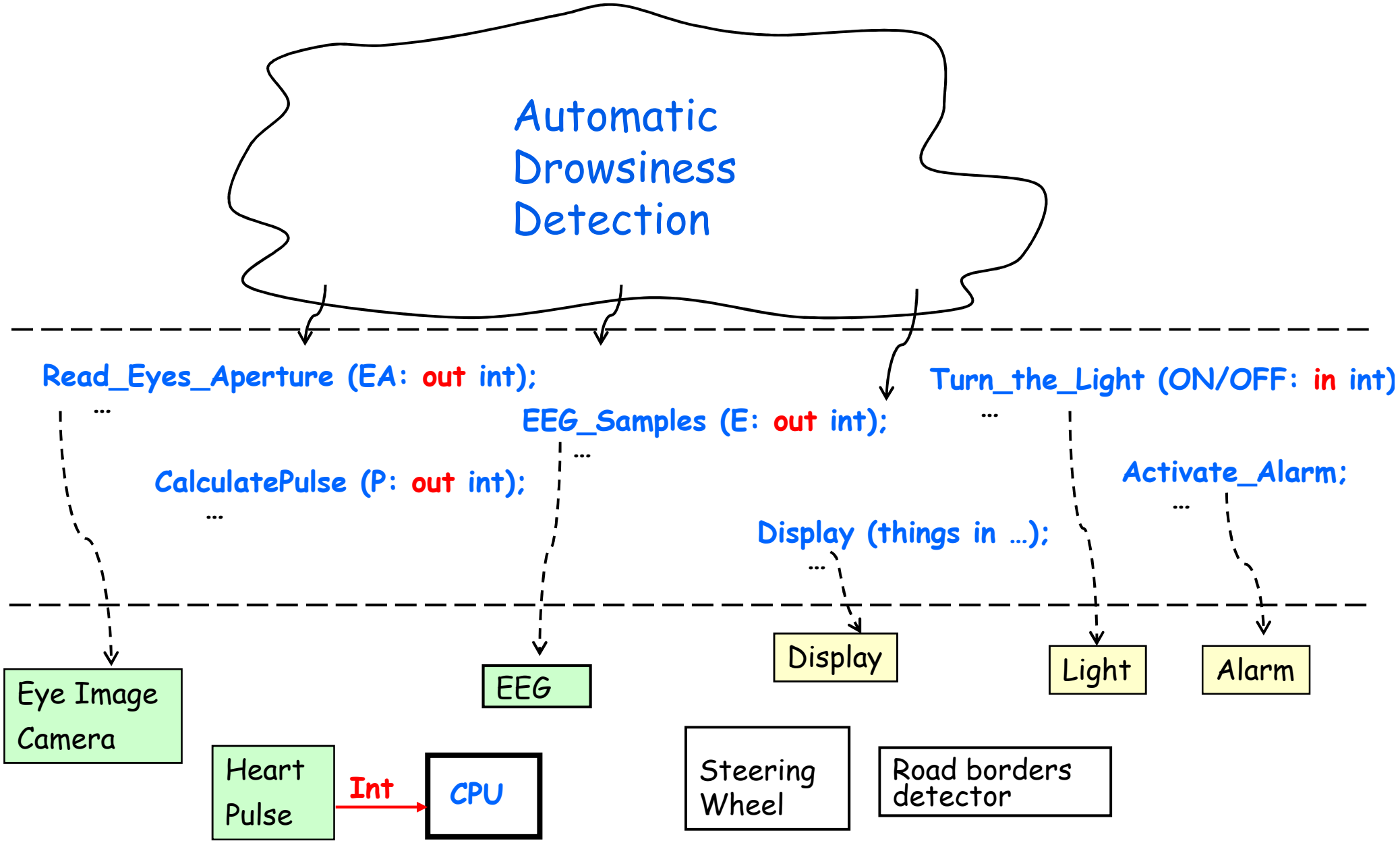
Head position

Extensions

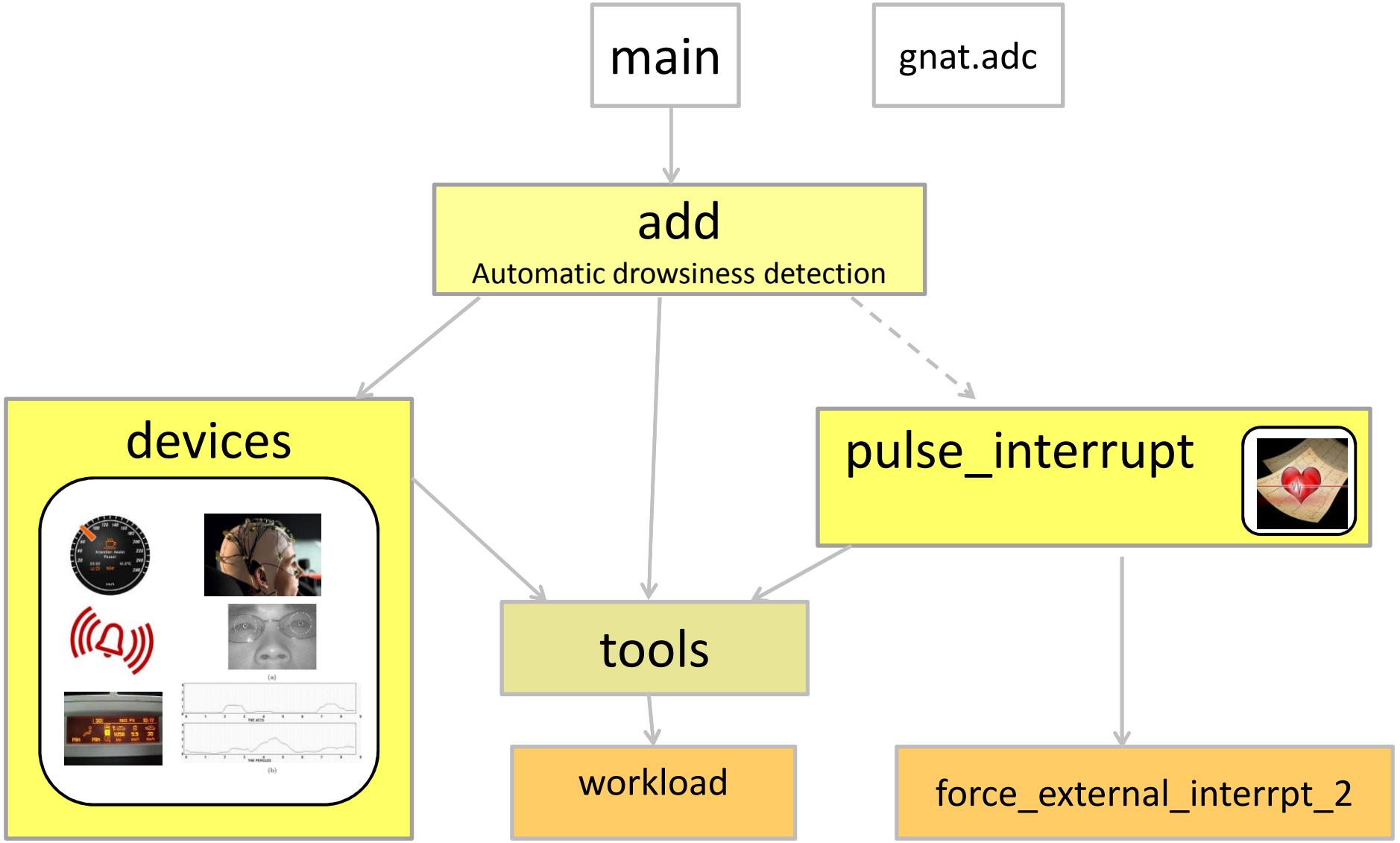
Automatic deviation detection



Devices Library



Packages


main
gnat.adc
add
Automatic drowsiness detection
devices
pulse_interrupt
tools
workload
force_external_interrpt_2

Devices interface

```
with Ada.Real_Time; use Ada.Real_Time;

package devices is
   ------------------------------------------------------------------
   ------ INPUT devices interface
   ------------------------------------------------------------------
   ------------------------------------------------------------------
   ------ ELECTRODES ------------------------------------------------

   type Value_Electrode is new natural range 0..10;
   Number_Electrodes: constant integer := 10;

   type EEG_Samples_Index is new natural range 1..Number_Electrodes;
   type EEG_Samples_Type is array (EEG_Samples_Index) of Value_Electrode;

   procedure Reading_Sensors (L: out EEG_Samples_Type);
   -- It reads a sample of Electrode Sensors and returns a array of 10 values
```

Devices interface

```
-----------------------------------------------------------------
------ EYES -----------------------------------------------------

type Eyes_Samples_Index is (left,right);
type Eyes_Samples_Values is new natural range 0..100;
type Eyes_Samples_Type is array (Eyes_Samples_Index) of Eyes_Samples_Values;

procedure Reading_EyesImage (L: out Eyes_Samples_Type);
-- It reads an image of the eyes, analyses the image and returns
--- the percentage of aperture (0..100) of every eye (left, right)
```

Devices interface

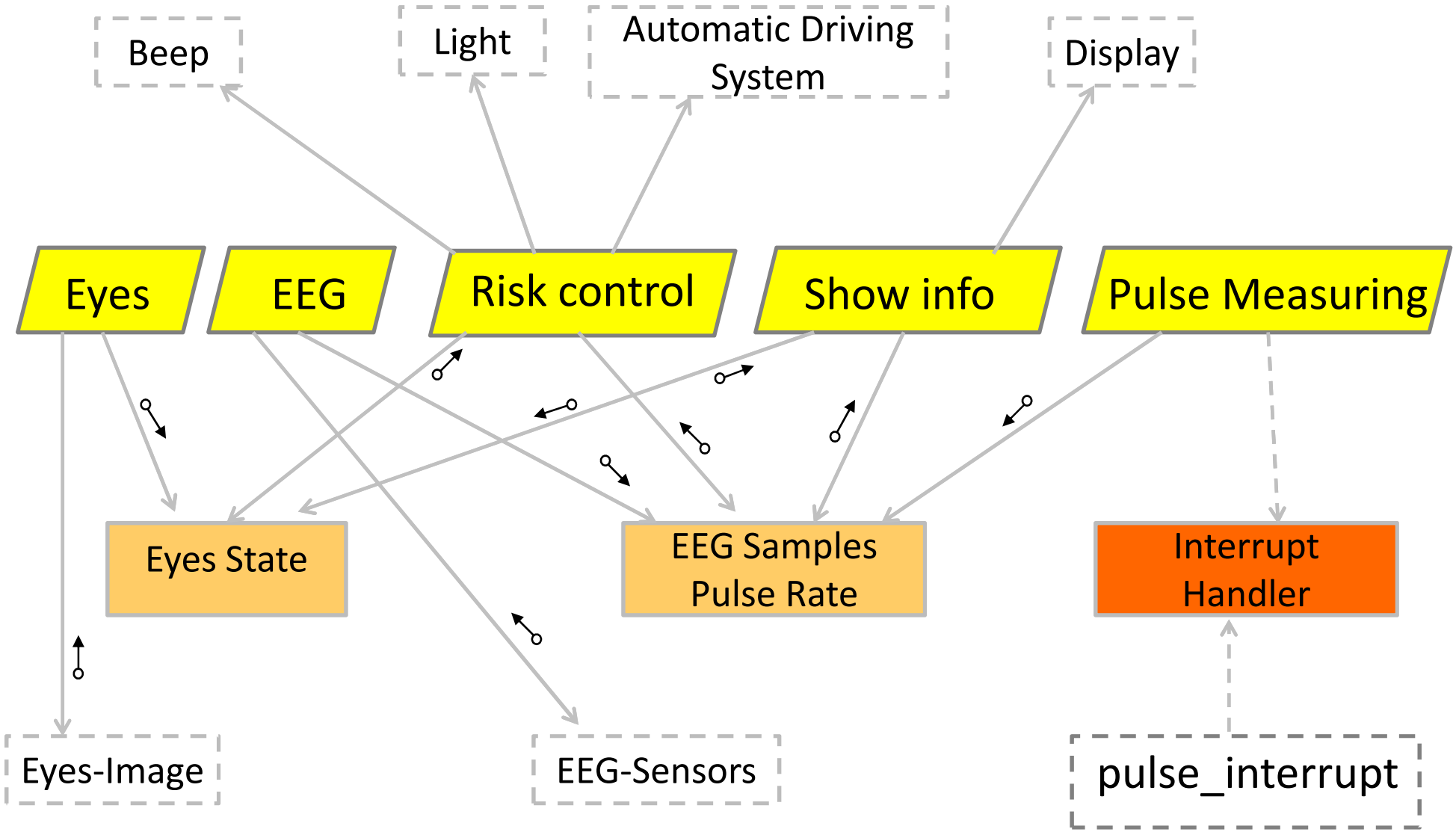
```
-------------------------------------------------------------------
------ OUTPUT devices interface
-------------------------------------------------------------------
   type Values_Pulse_Rate is new float range 20.0..300.0;
   procedure Display_Pulse_Rate (P: Values_Pulse_Rate);
   -- It displays the pulse rate P
   -------------------------------------------------------------------
   procedure Display_Electrodes_Sample (R: EEG_Samples_Type);
   -- It displays the 10 values of the electrodes sample
   -------------------------------------------------------------------
   procedure Display_Eyes_Sample (R: Eyes_Samples_Type);
   -- It displays the values of eyes aperture (left and right)
   -------------------------------------------------------------------
   type Light_States is (On, Off);
   procedure Light (E: Light_States);
   -- It turns ON/OFF the light
```

Devices interface

```
   ------------------------------------------------------------------
   procedure Display_Cronometro (Origen: Ada.Real_Time.Time; Hora: Ada.Real_Time.Time);
   -- It displays a chronometer
   ------------------------------------------------------------------
   Type Volume is new integer range 1..5;
   procedure Beep (v: Volume);
   -- It beeps with a volume "v"
    ------------------------------------------------------------------
   procedure Activate_Automatic_Driving;
   -- It activates the automatic driving system

end devices;
```

Process map

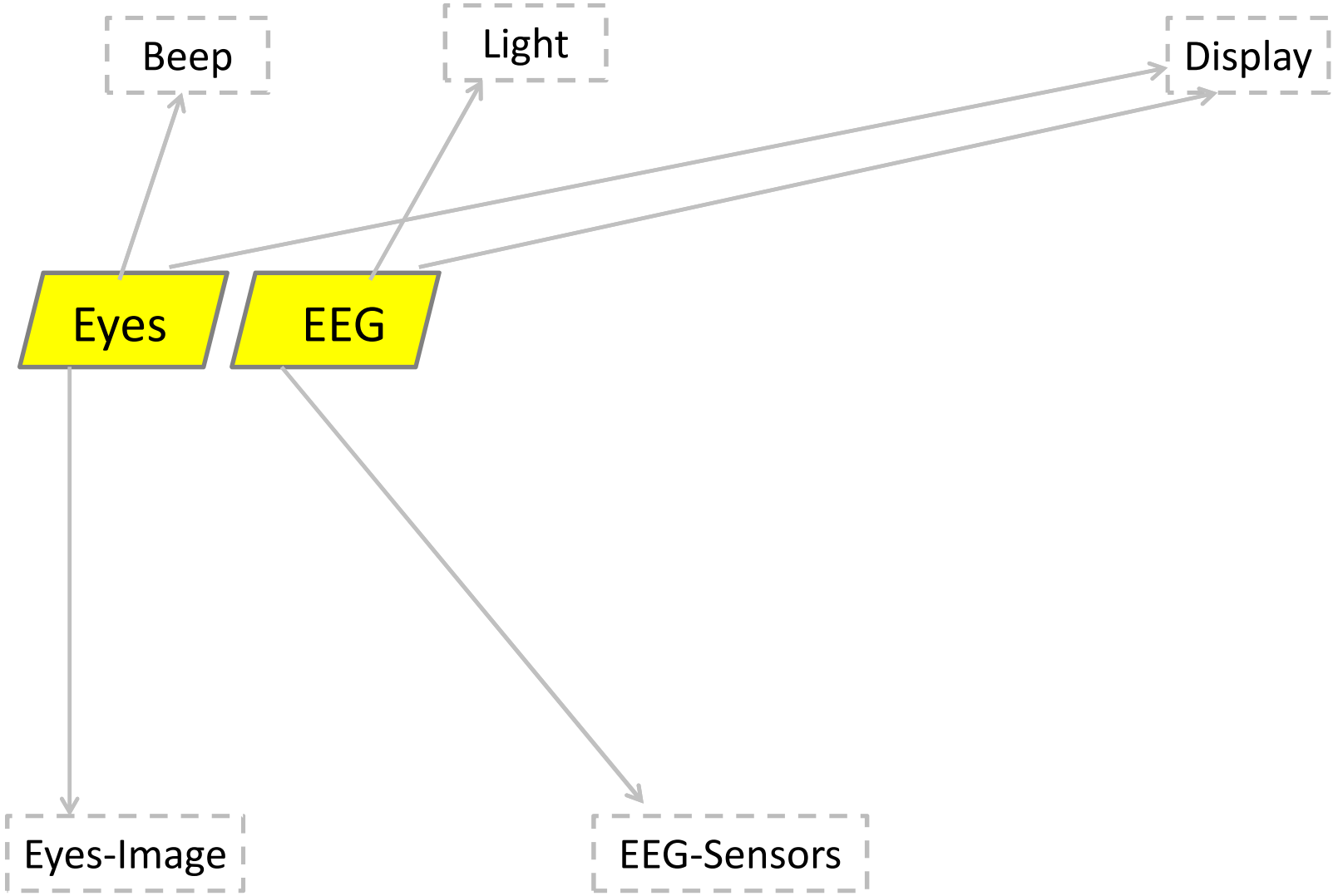

Beep
Light
Automatic Driving System
Display
Eyes
EEG
Risk control
Show info
Pulse Measuring
Eyes State
EEG Samples
Pulse Rate
Interrupt Handler
Eyes-Image
EEG-Sensors
pulse_interrupt

Time Requirements

Tasks and PO	Type	Period	Deadline	WCET
Eyes	C	150	150	?
EEG	C	300	300	?
Pulse Measuring	S	?	?	?
Risk Control	C	250	100	?
Show Info	C	1000	1000	?
Eyes_State	P	-	-	?
EEG_Samples Pulse Rate	P	-	-	?

Devices	Eyes Image	EEG Sensors	Beep	Light	Display	Automatic driving system
Access time	30 ms	20 ms	5 ms	5 ms	15 ms	10 ms

Implementing step by step



Some directions about periodic tasks implementation

Let's know when a task is starting and finishing

```
loop
   Starting_Notice ("name of the task");
    ..... Activity of the task
   Finishing_Notice ("name of the task");
   delay ......
end loop;
```

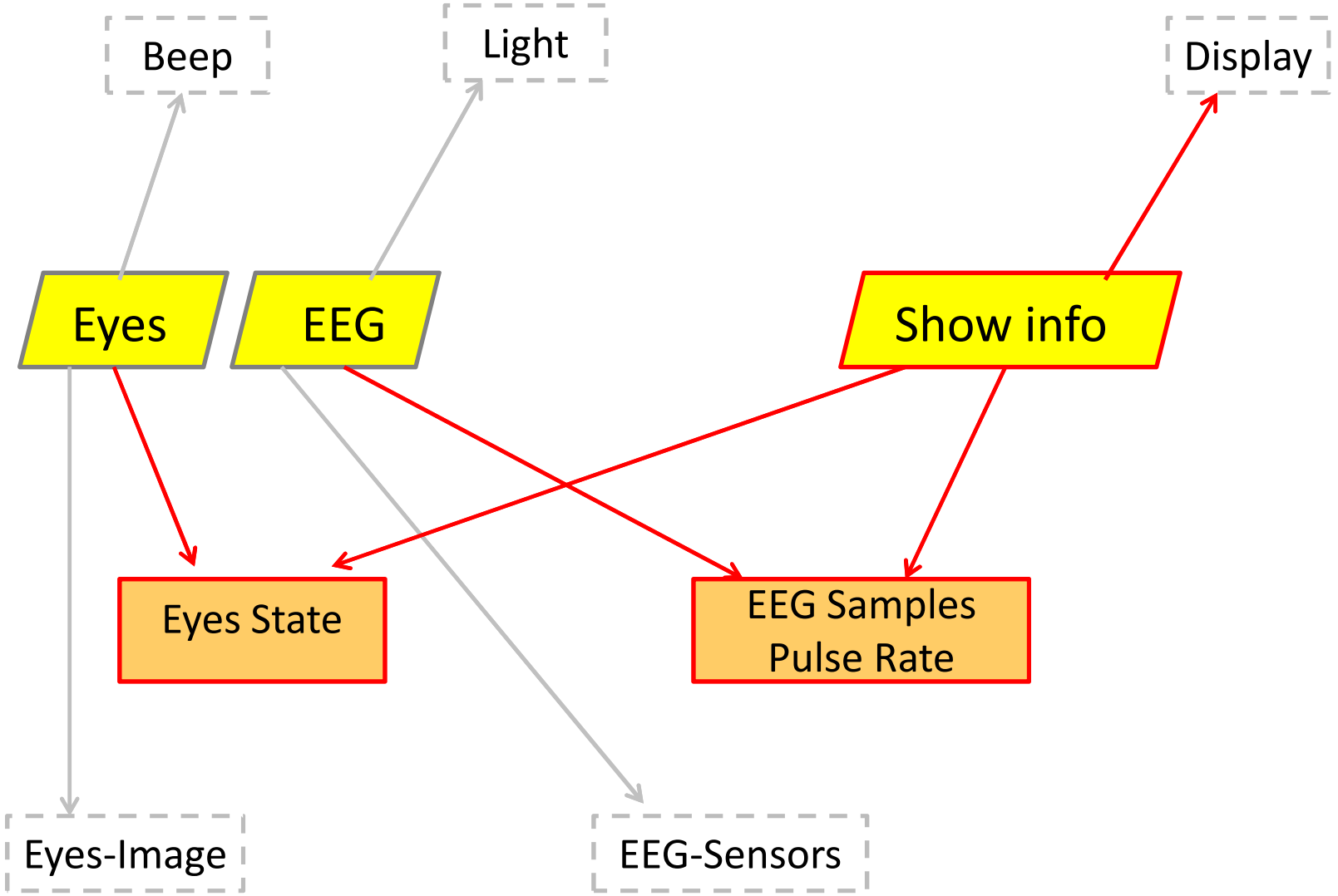
Assign priorities of tasks: higher number implies higher priority

```
task A is
   pragma priority (Priority_of_A);
end A;
```

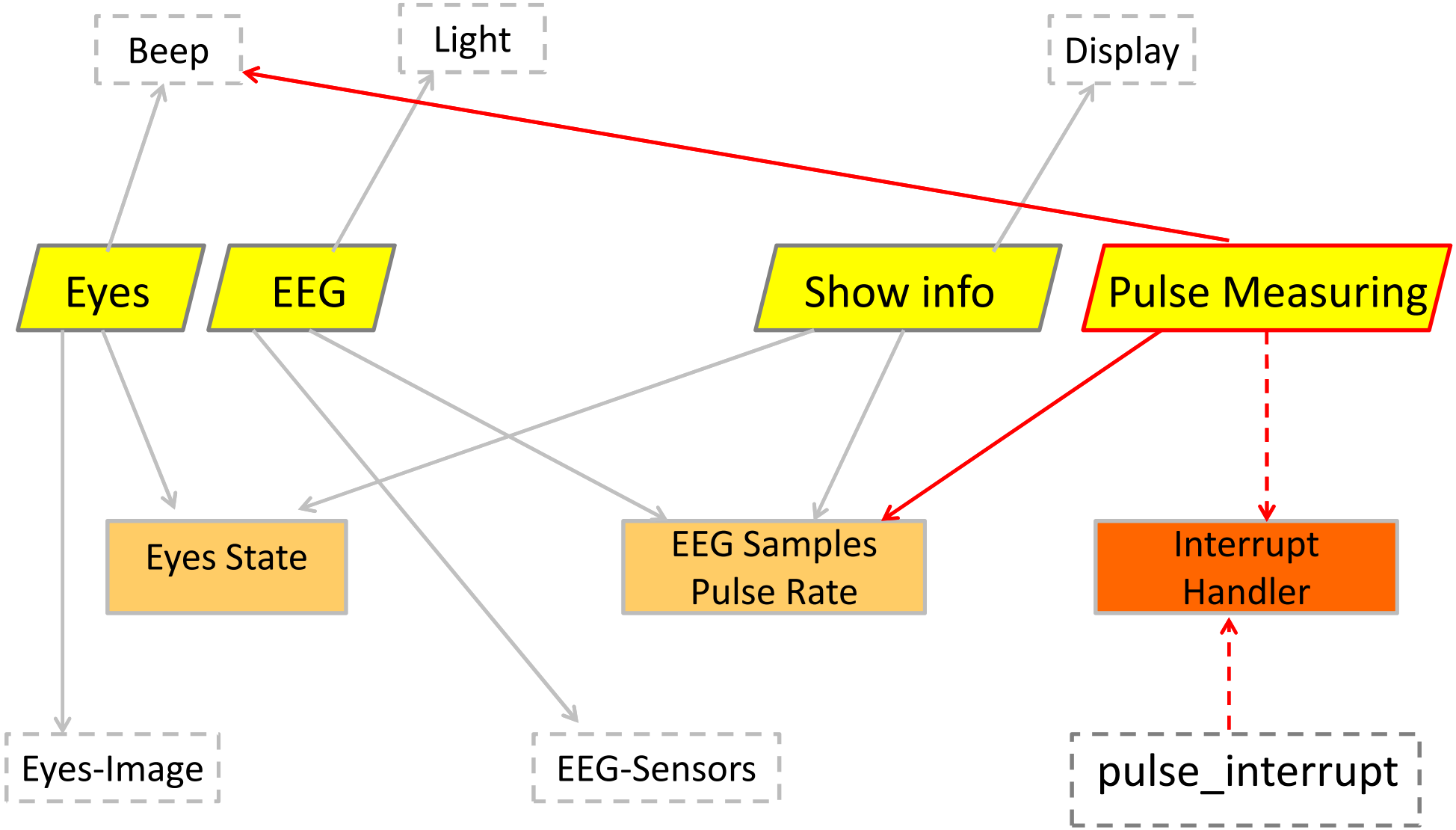
Assign priorities of protected objects: ceiling priority

```
Protected P is
    pragma priority (Ceiling_Priority_of_P);
    procedure Pr ;
    ...
end P;
```

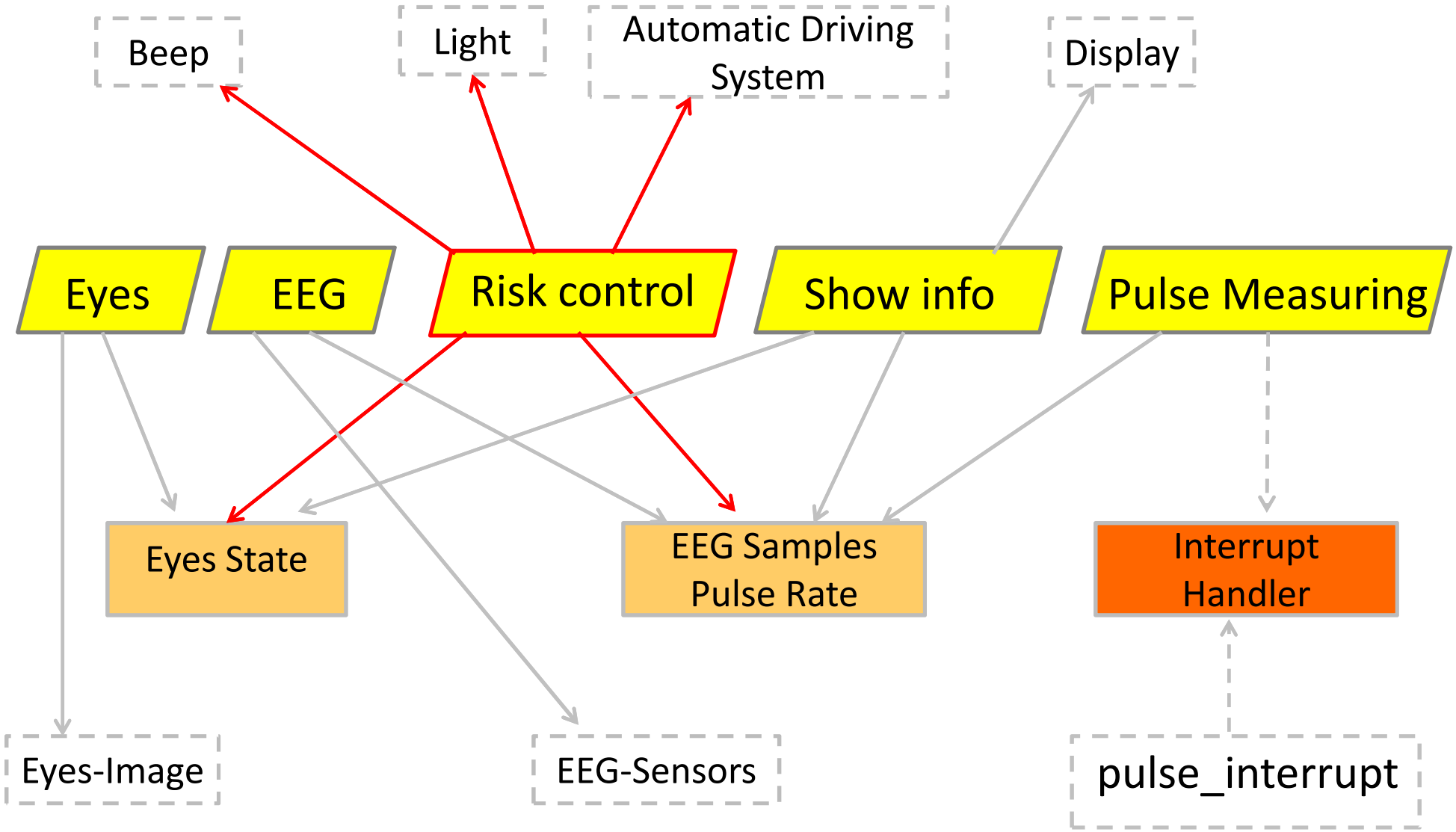
Implementing step by step



Implementing step by step



Implementing step by step



Checking correctness

PERIODIC TASKS

- Is every task activated in the accurate moment according to its period?
- Can you identify some local drifts during the release of the tasks?
- Does these local drifts keep accumulated in next activations of tasks? How can you check it?
- How long does it take each activation of each task? (execution time of each activation)
- Is each activation of each task meeting its deadline?
- Can you identify some preemptive situations? (A task is preempted by other task with higher priority)
- Are tasks running according to their priorities?
- Is the priority order being met?
- Are access to shared objects (protected object) correct? How can you be sure?

Directions about Interrupt programming

In the package *Pulse Interrupt* we have implemented a tasks that generates the interrupt *External_Interrupt_2* at the moments indicated in the package *Pulse_Interrupt.ads* (constants *Interr_n);*

In order to attend this interrupt, that means, to implement the Interrupt Routine, next directions are given:

- Interrupt Routine should be a protected procedure
- The priority of the interrupt will be based on the priorities defined in the package *System:*

```
System.Interrupt_Priority'First + 9
```

- The name of the interrupt is defined in the package *Ada.Interrupts.Names:*

```
Ada.Interrupts.Names.External_Interrupt_2
```

- Installation of the interrupt will be carried out by using the pragma *Attach_Handler (...)* inside the protected object

Once the interrupt handler has been implemented, uncomment the line
-- with Pulse_Interrupt ;
Pulse interrupts will start to come to your system

Checking correctness

SPORADIC TASKS

- Is the interrupt routine executed as soon as a new interrupt is generated?
- Is the sporadic task correctly unlocked every time interrupt routine is executed?
- Is the pulse rate correctly calculated according to the intervals between interrupts?
- What is the drift between the moment in which the interrupt is generated and the sporadic task is released (it starts a new activation)?
- How can this drift affect on the precision of pulse calculation?
- Is the pulse driver calculated correctly, taking into account that drift?

Hand-in

- ✓ Source code of packages *add.adb* y *add.ads*
- ✓ Source code of other packages that have been modified
- ✓ Printout of the output generated by the program execution

```
$  tsim-erc32 main > output.txt
   Type in  "go"
   Wait for 3 or 4 seconds
   Type in ^c
   Type in "quit"
```

- ✓ Write a table to indicate local drifts (milliseconds precision, 3 decimal) produced in task activation during the first 2 seconds.
- ✓ Detect the preemptive situations produced during the first 2 seconds. Specify the time when they happen, the task that has been preempted (and its priority) and the task that forces the preemptive situation (and its priority).
- ✓ Mark in the printout both, local drifts and preemptive situations.