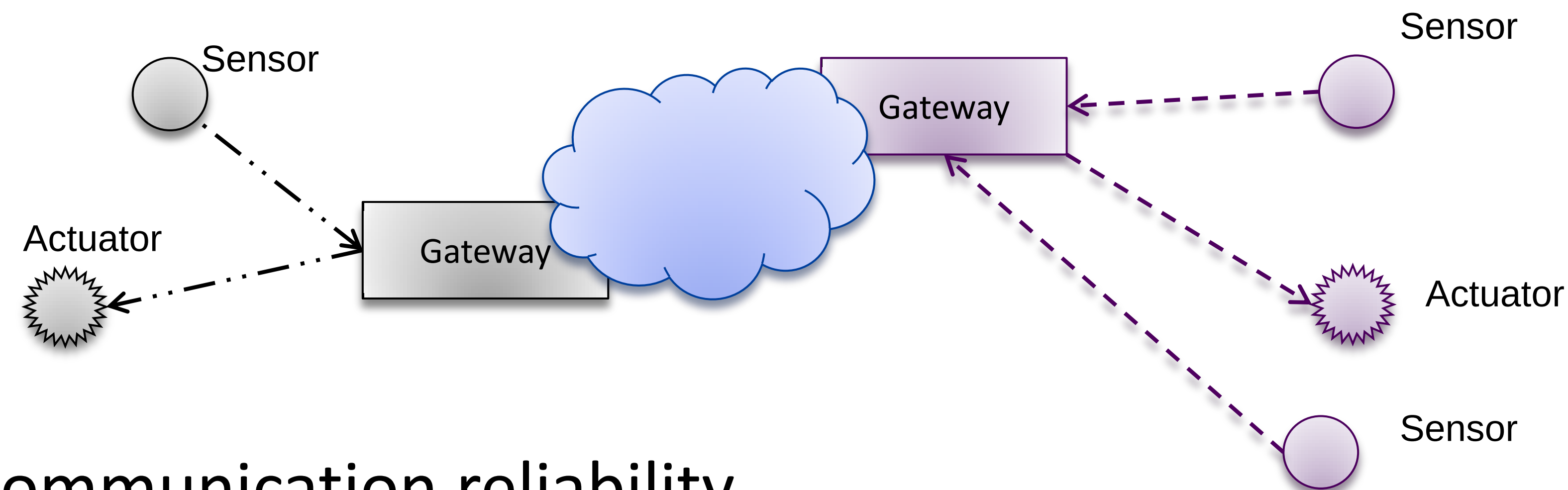
Protocol Awareness:
A Step Towards Smarter Sensors

Hoel Iris, Francois Pacull

CEA-LETI MINATEC Campus, France
Francois.Pacull@cea.fr

Context

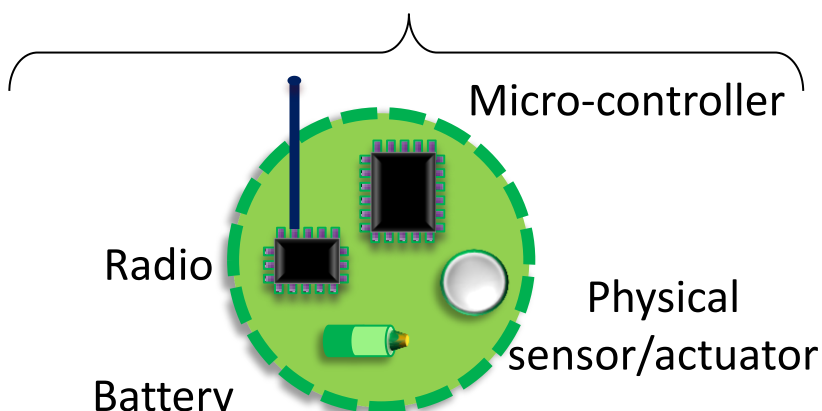
Building Automation Systems



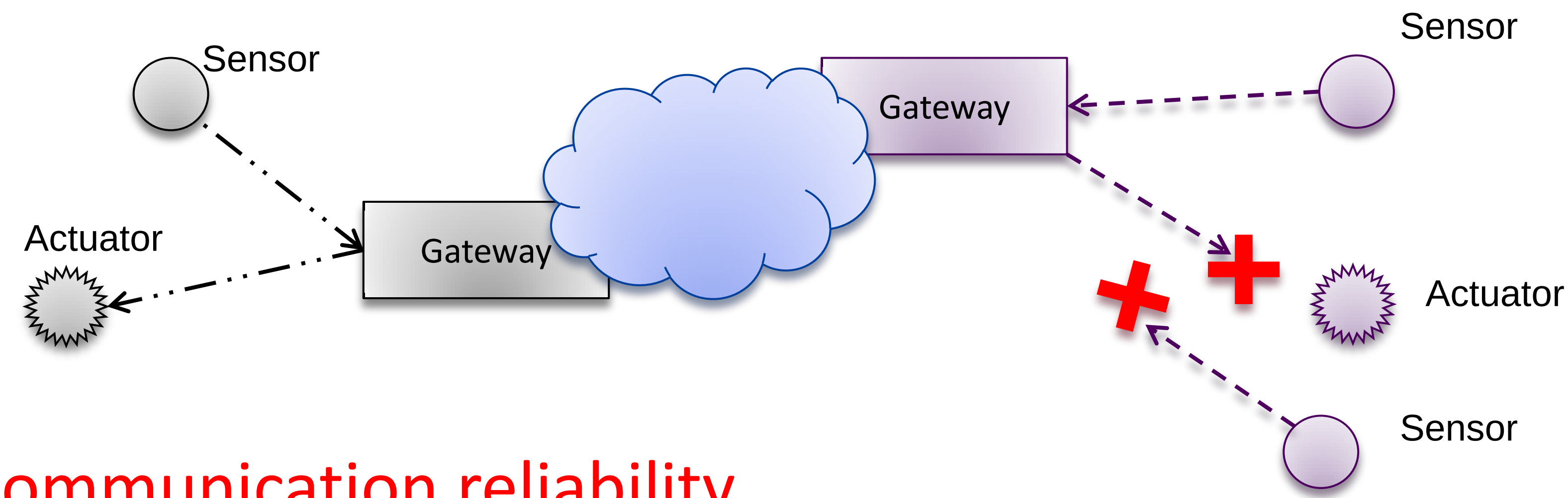
Communication reliability

Energy consumption

Coordinated actions



Context



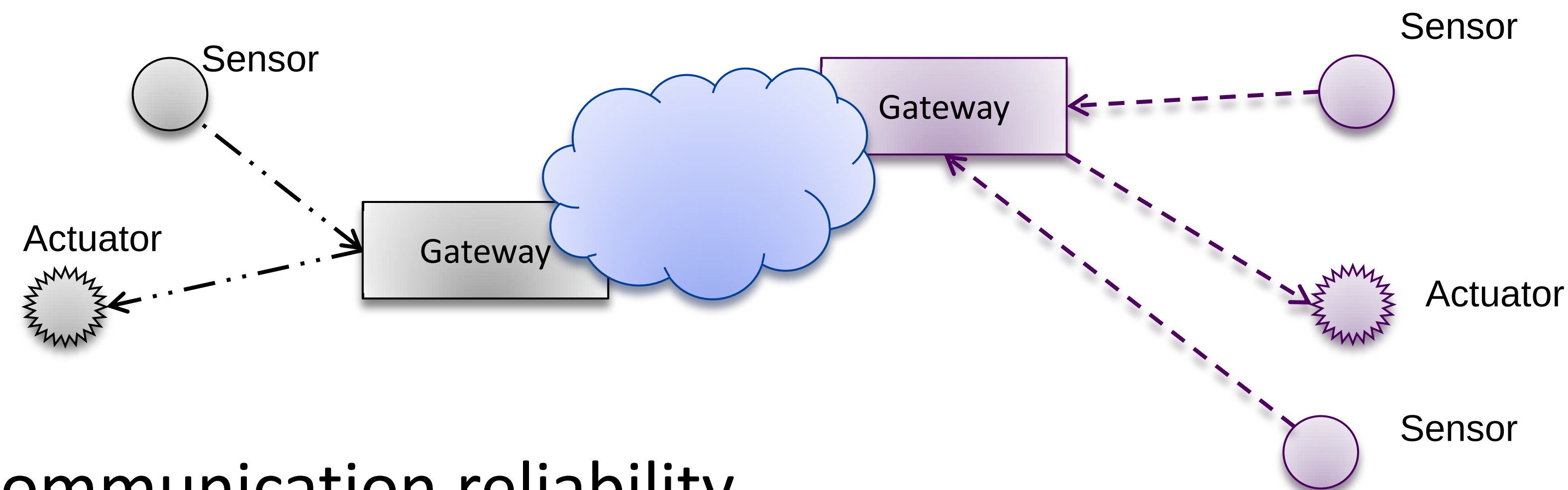
Communication reliability

Energy consumption

Coordinated actions

Most of the communications ensure only a best effort

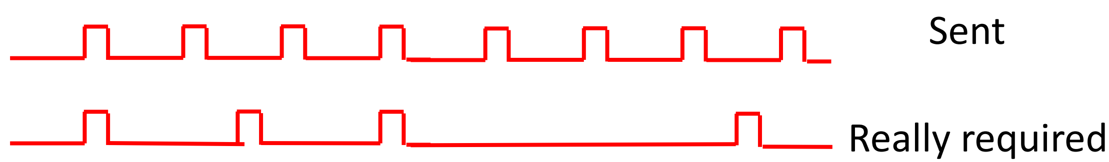
Context



Communication reliability

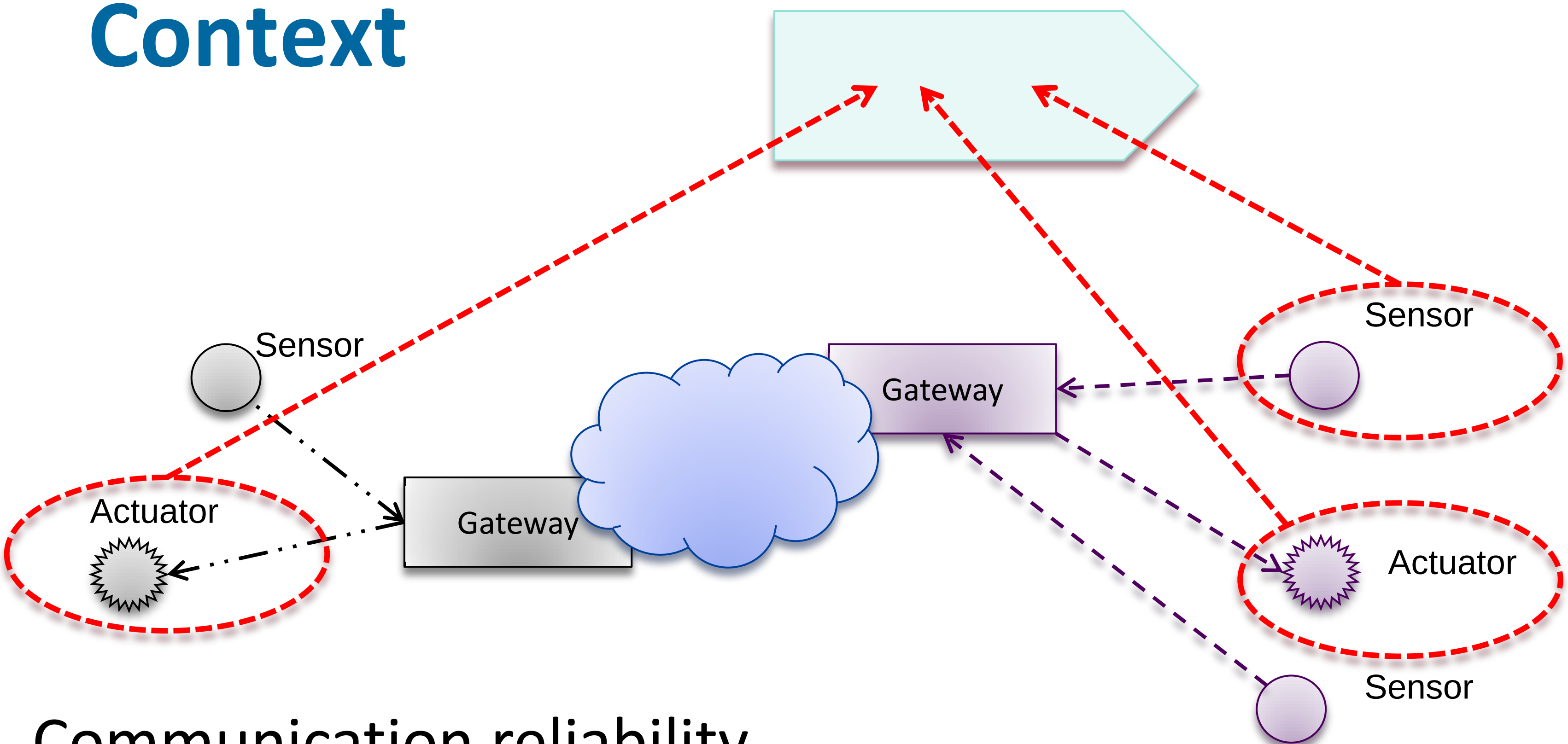
Energy consumption

Coordinated actions



Sensing rate is under the responsibility of the sensors

Context



Communication reliability

Energy consumption

Coordinated actions

Difficult to ensure the performance of a group of actions

Outline of the presentation

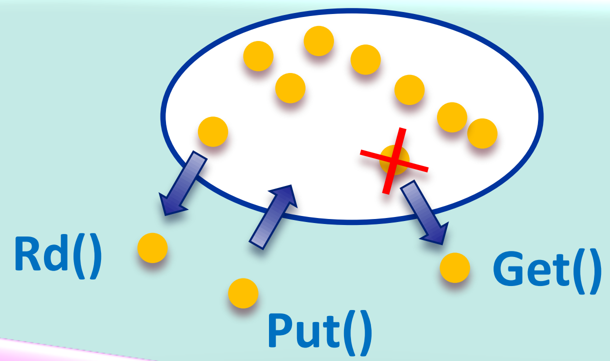
The high level coordination protocol we rely on

How we make the sensors aware of this protocol

2 examples as illustration

Middleware / coordination protocol

Associative Memory



Database record
(field1, field2, field3)

Event
(evenid, type, tm, payload)

Service
(in1, in2, out1, out2, out3)

Sensor
(id, type, value)

Actuator
(id, cmd, p1, p2, p3)

Production Rules

Precondition
based on the **Rd()**

" when these conditions are reached I would trigger something"

Performance
To verify the **Rd()** are still valid
To consume some resources **Rd()**
To consume some resources **Get()**
To produce some resources **Put()**

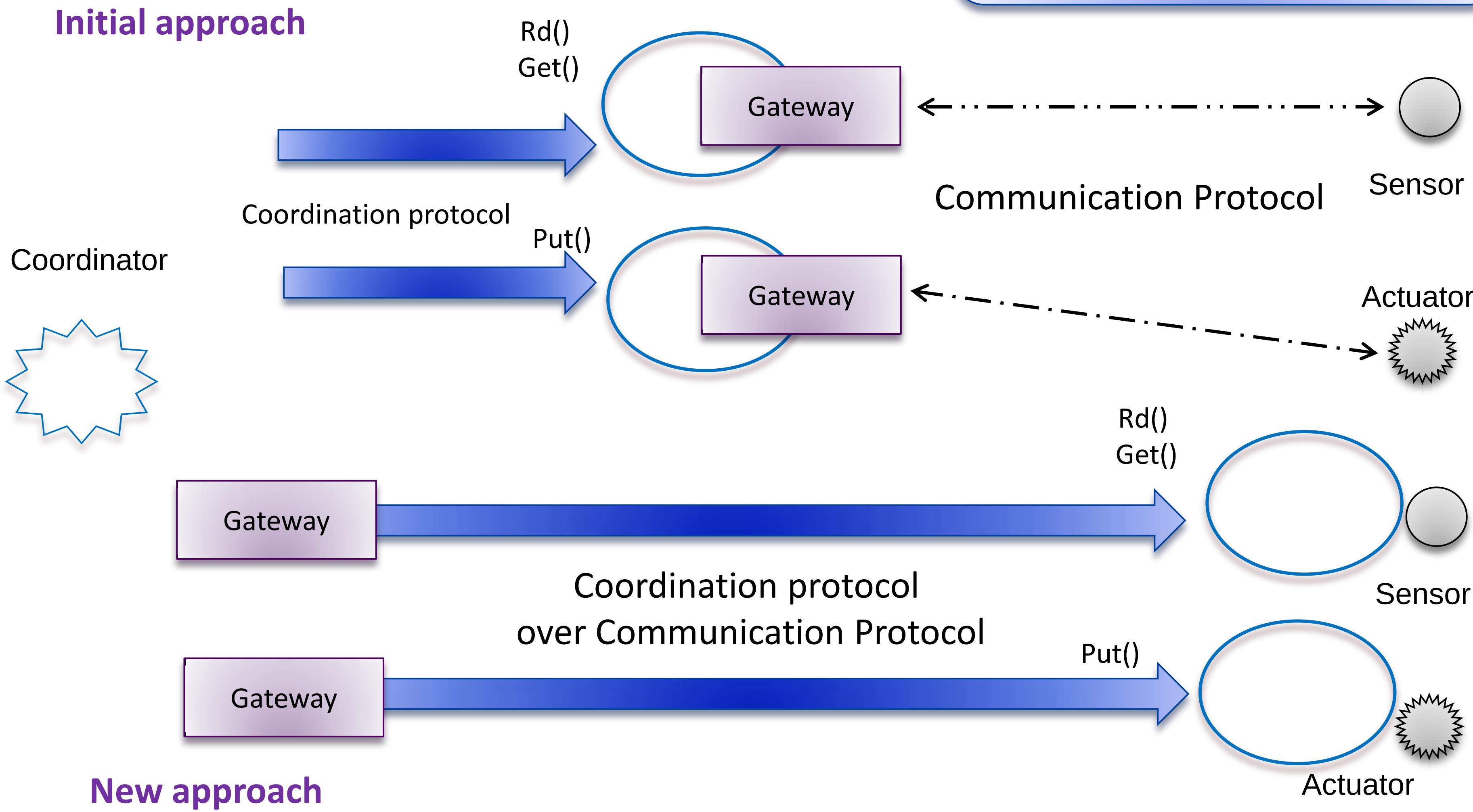
Distributed Transactions

Rd(), **Get()** and **Put()** operations
are performed as a sequence of transactions
{ … } { … } { ….}

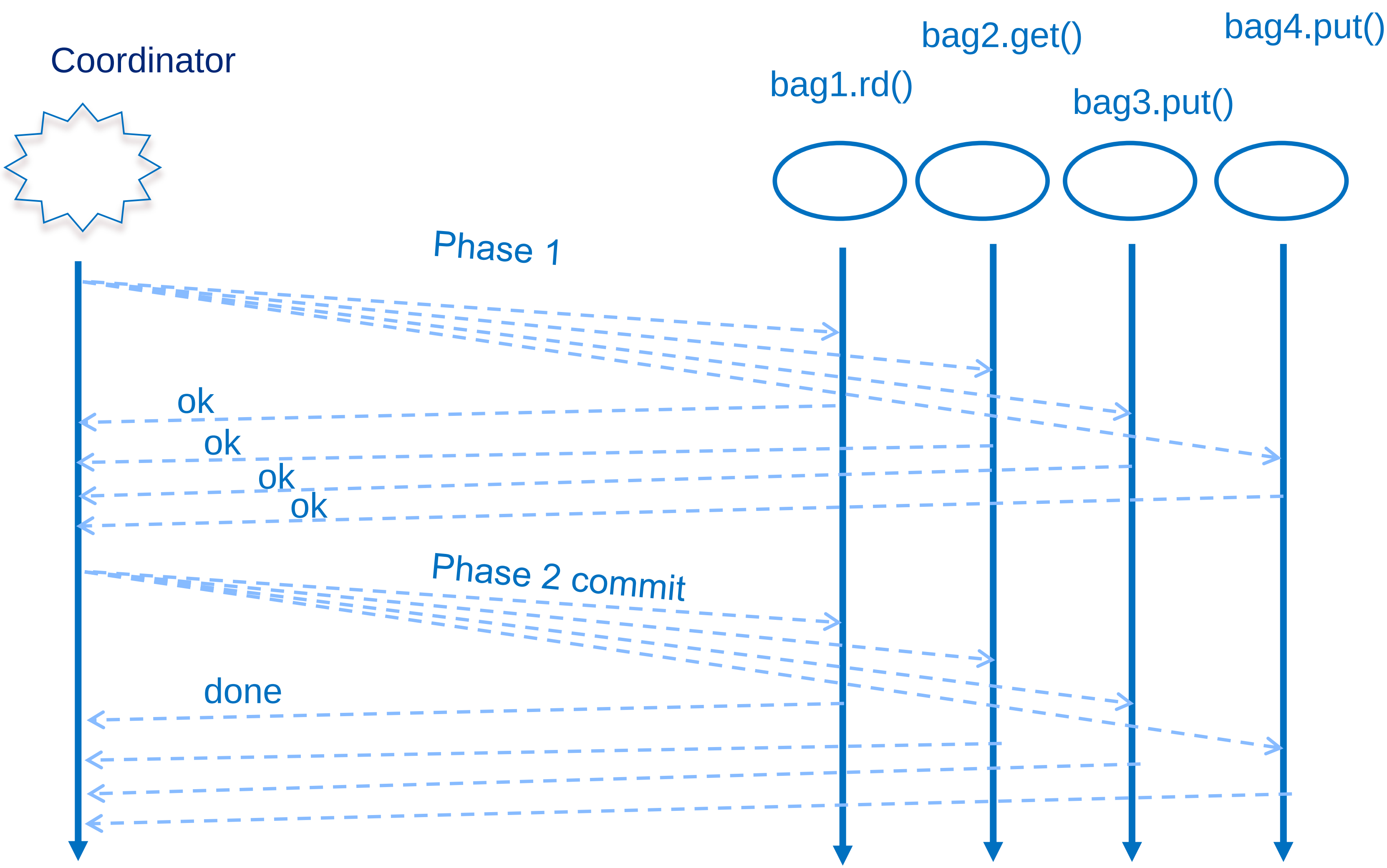
each of the transaction into
curly bracket enforces all-or-nothing

Protocol aware sensor

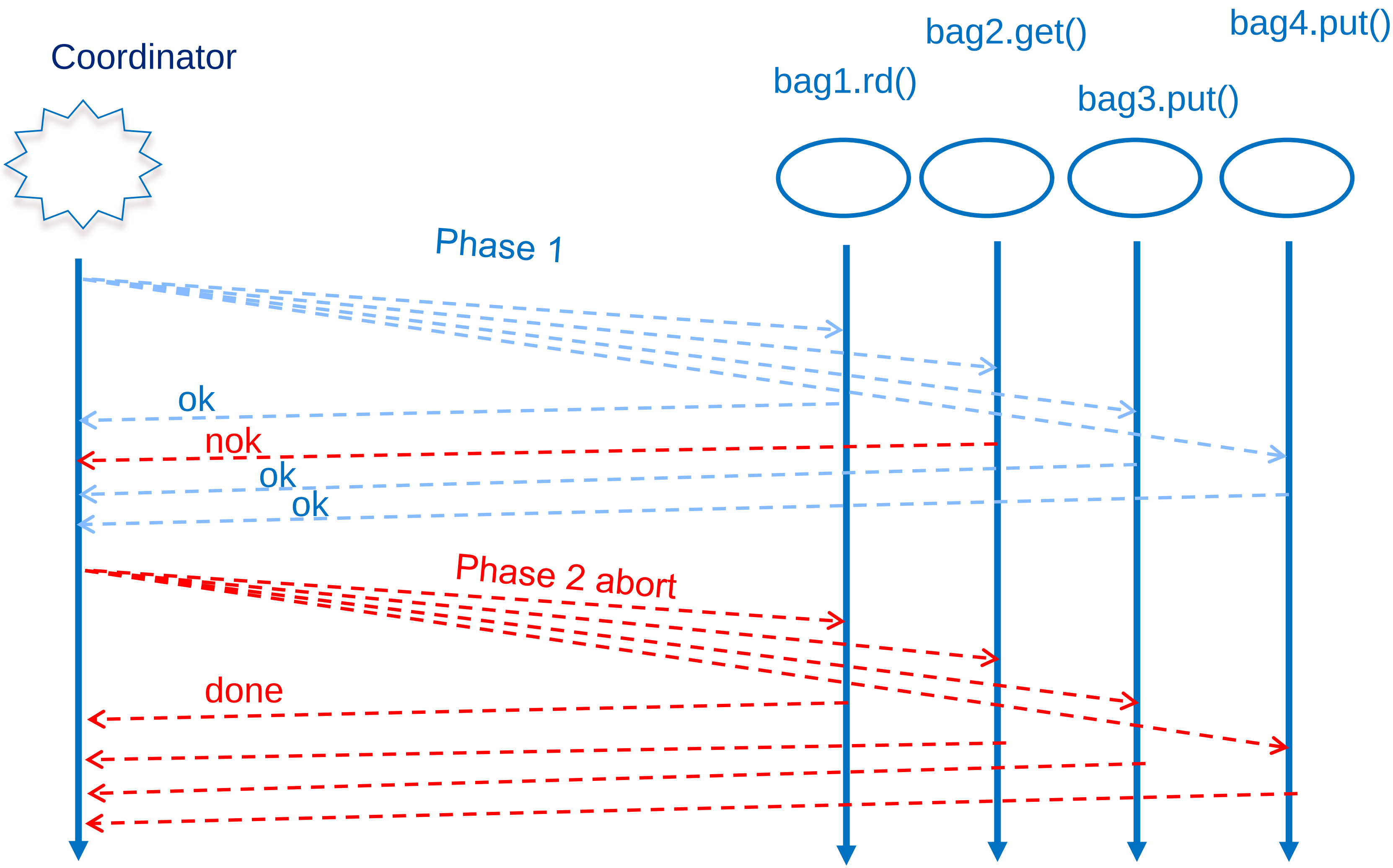
The Rd(), Get() and Put() are embedded in transactions



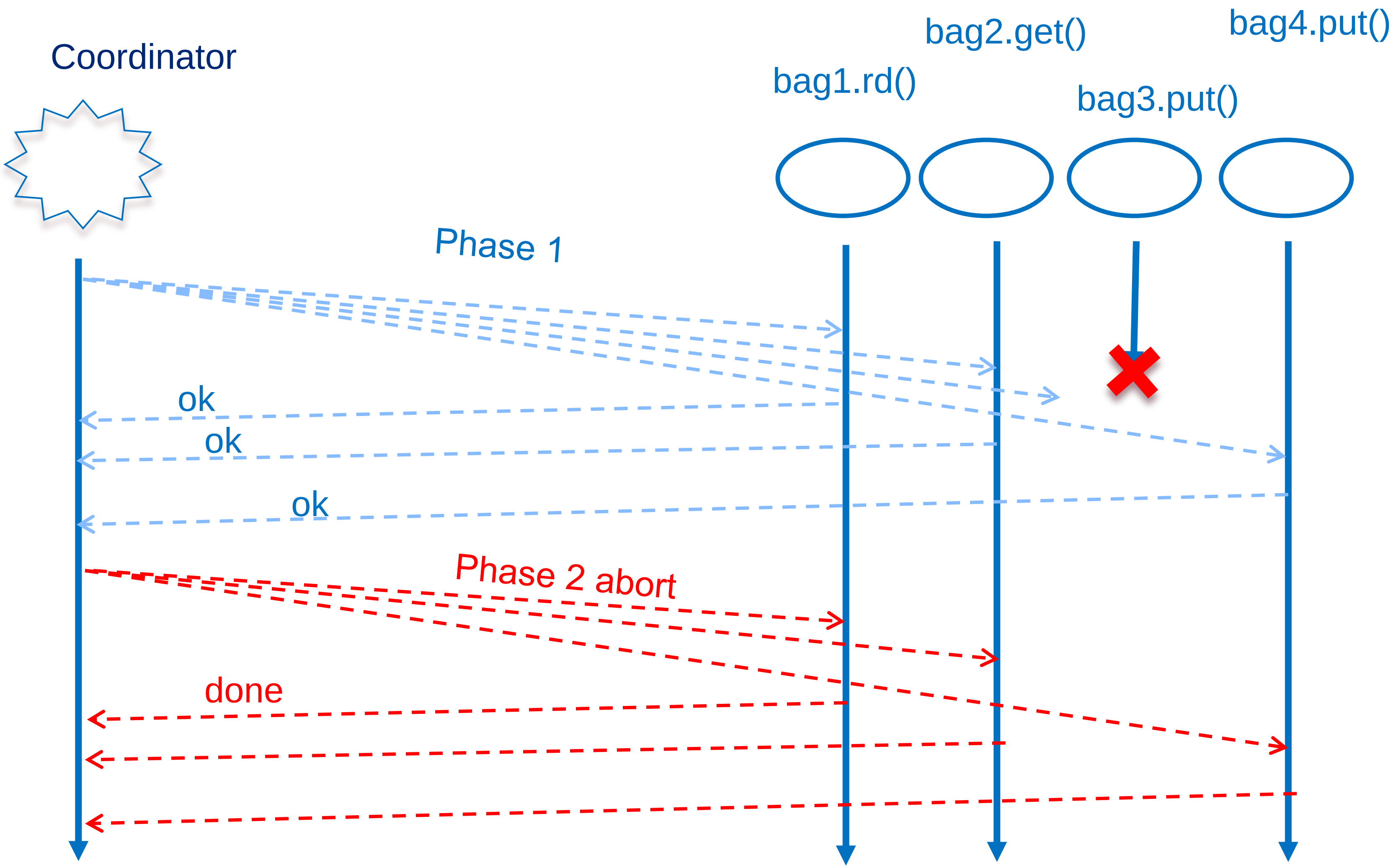
Example of transaction committed



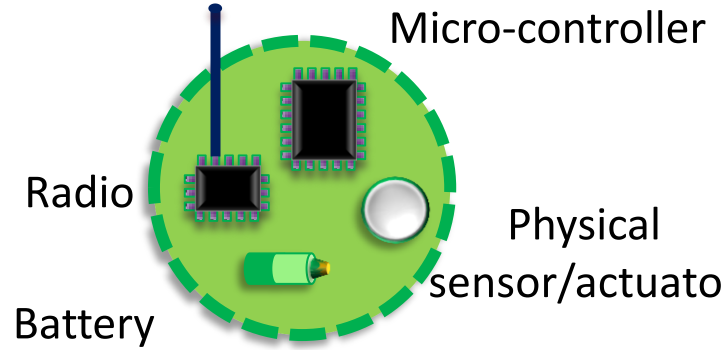
Example of transaction cancelled (processing)



Example of transaction cancelled (failure)



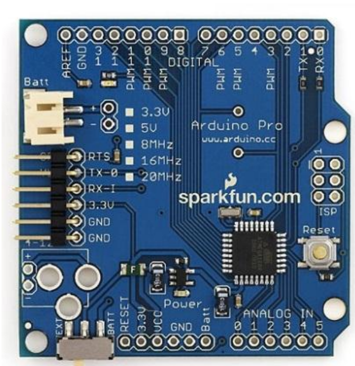
Platforms



OpenPicus Flyport + integrated Wifi (802.11 b/g/n)

16bits micro-controleur, 32MHz, 256Ko Flash, 16Ko Ram
26 I/O
Wifi (802.11 b/g/n)

Arduino - Xbee (802.15.4)



8bits micro-controleur, 8MHz, 32Ko Flash, 2Ko Ram
20 I/O
Xbee (802.15.4)

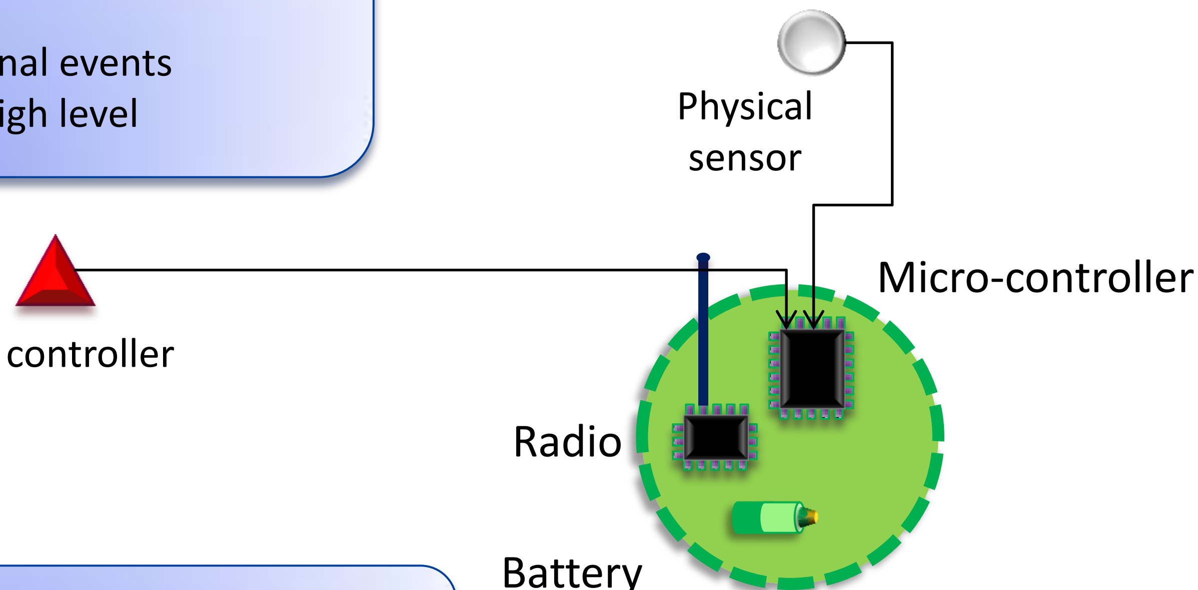
Wake up

Boards can be put in sleep mode
- communication
- micro-controller

Boards can be wake up by external events
- e.g. I/O pin set to high level

open-contact
e.g. Detect the opening of a door

Signal from application to signal that we need to talk to the micro controller



Same mechanism to wake up the micro controller only
- when the physical sensor has something useful to say
- when the application needs to interrogate the sensor

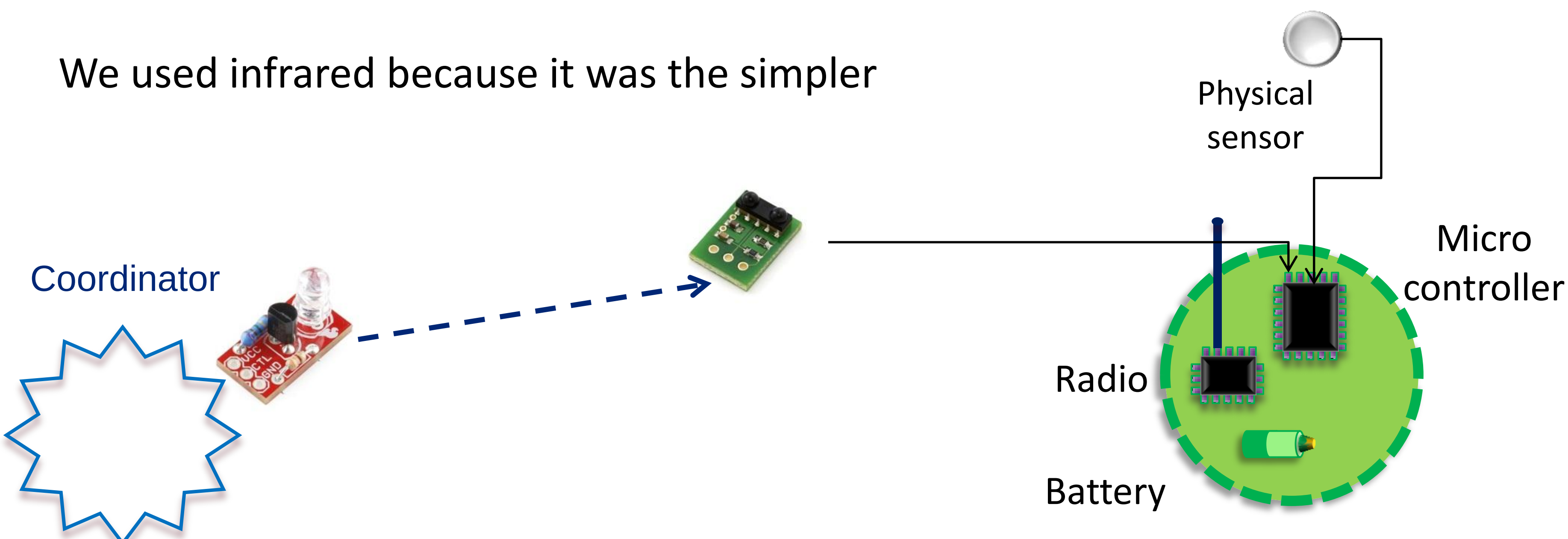
Wake up (current state)

Signal from application to warn that
we need to talk to the micro controller

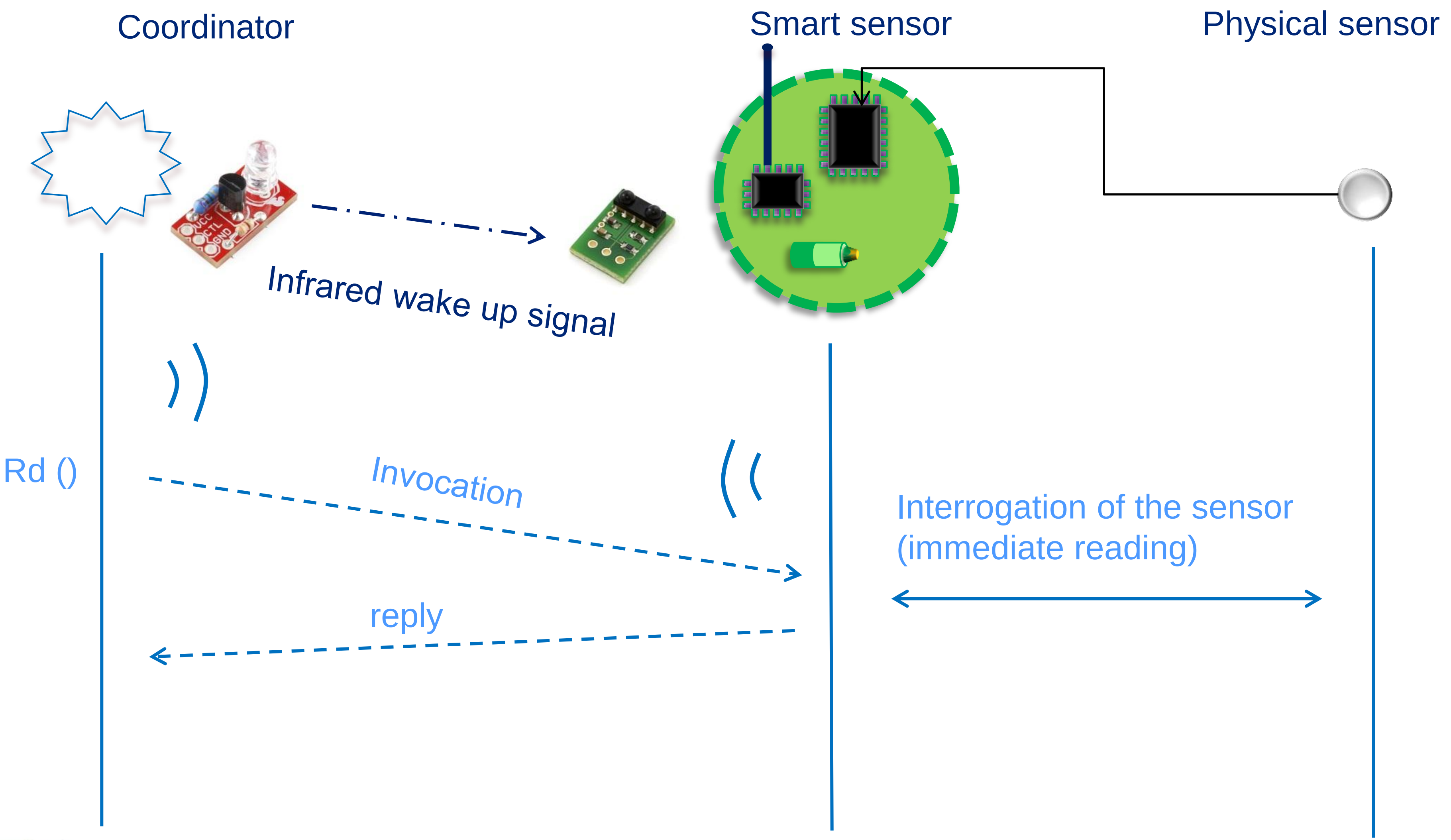
Several possibilities
- low cost wireless signal
- passive RFID
- infrared,
- …

This is out the scope of this paper and
let to further investigation

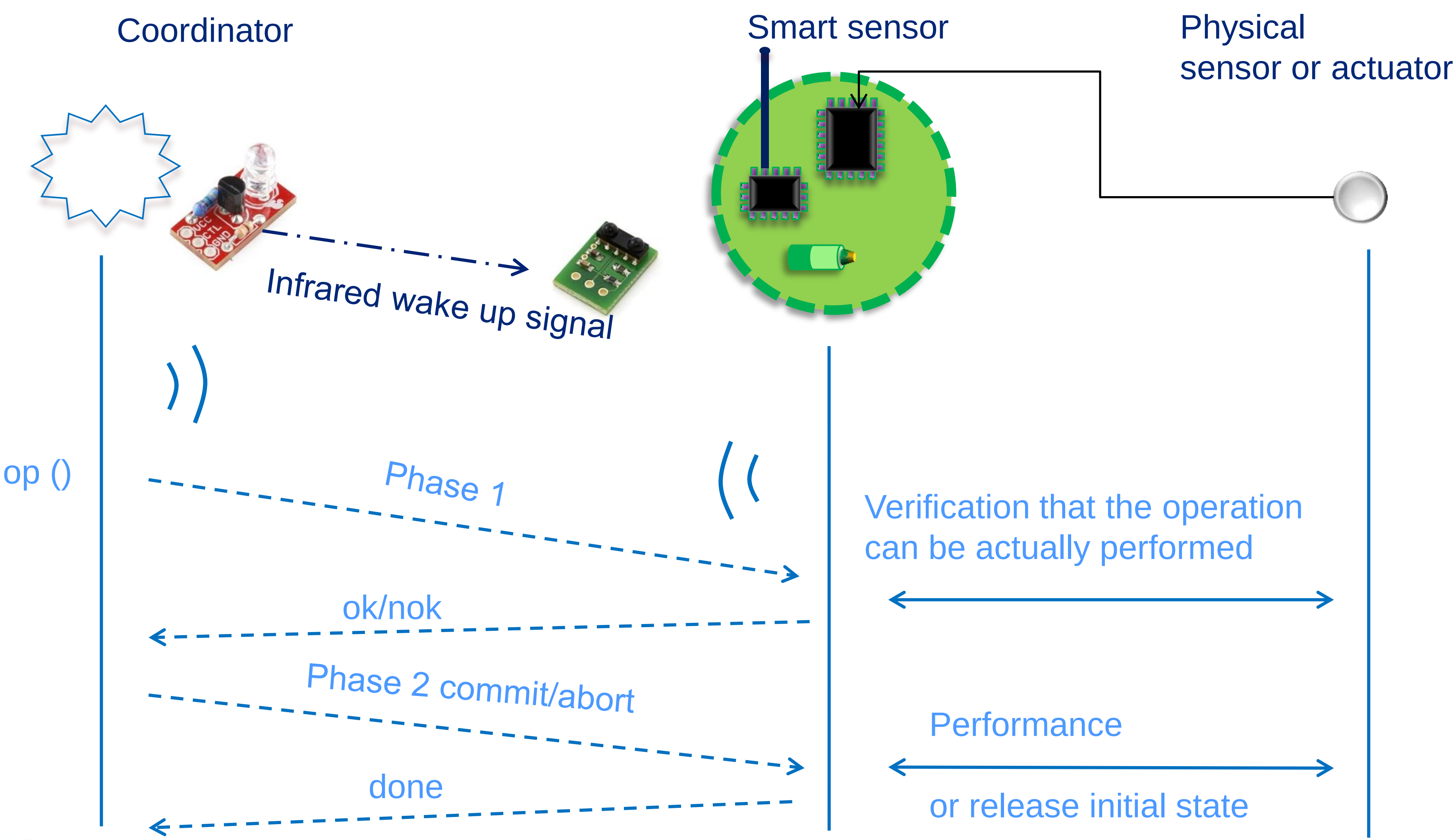
We used infrared because it was the simpler



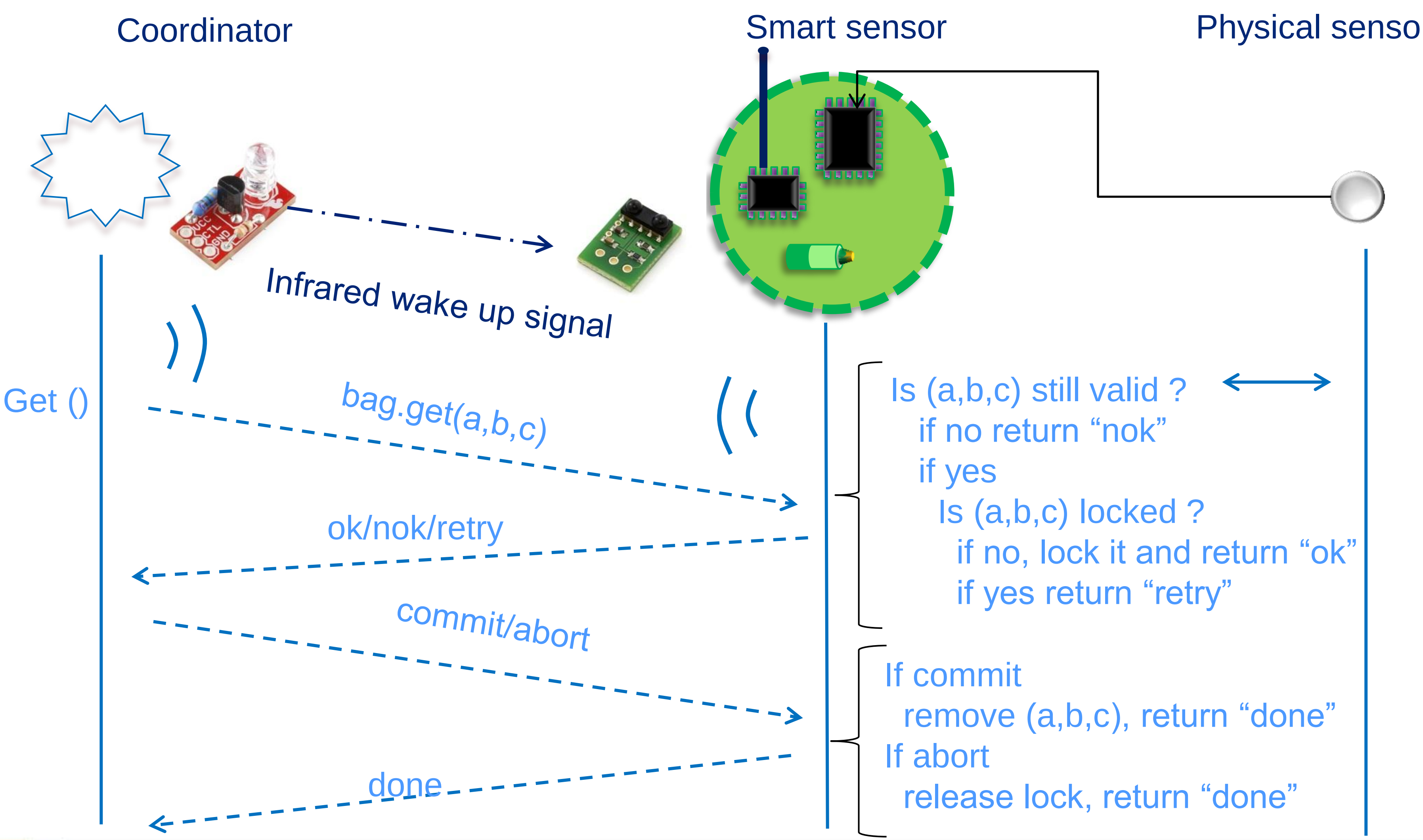
Coordination Protocol
Precondition (not transactional)



Coordination Protocol Performance (transactional)

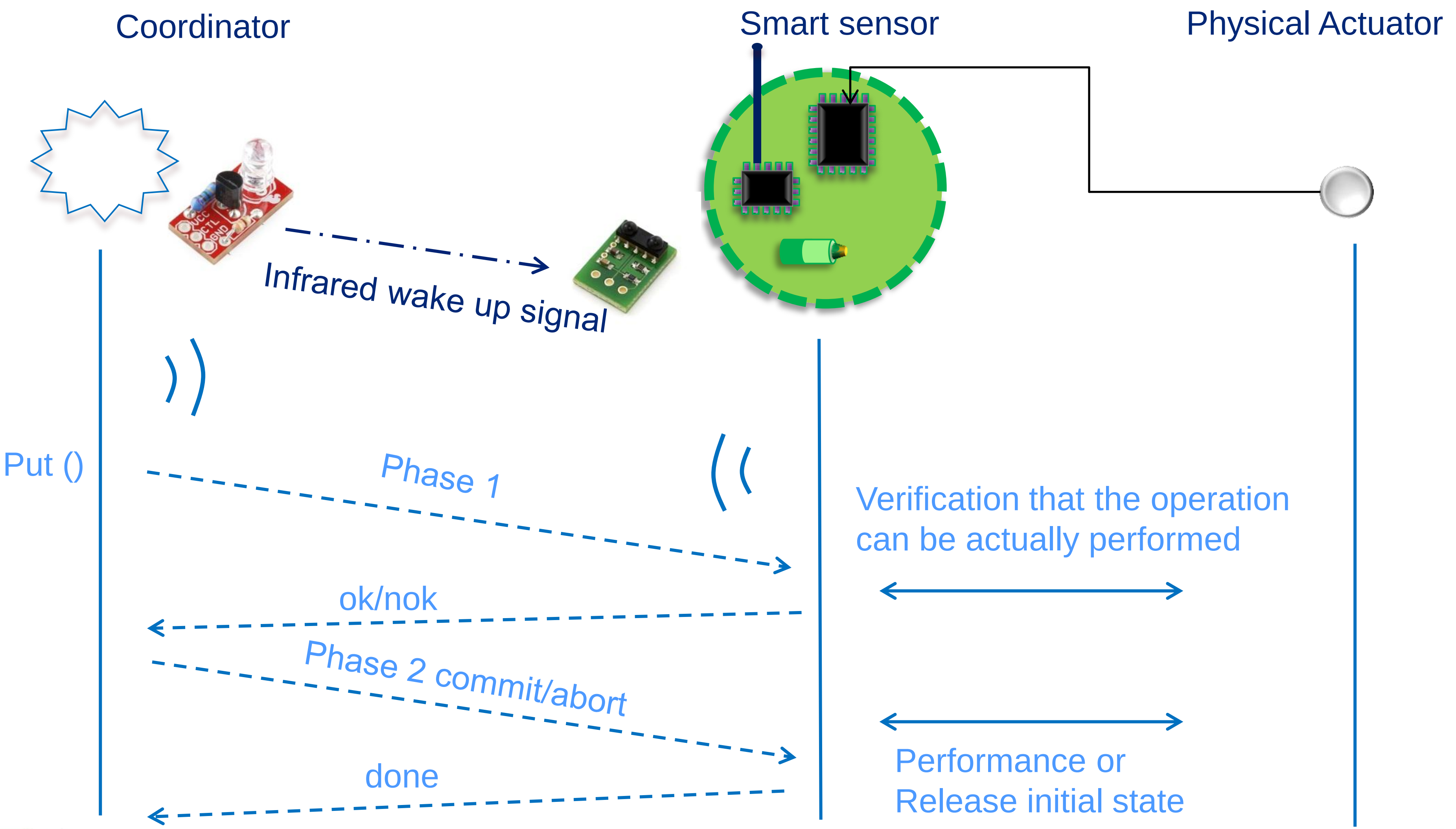


Coordination Protocol
Get()



Coordination Protocol

Put ()



Main interests

Precondition phase:
Interrogate the sensor only
when needed by the application

→ impact on the power consumption

Performance phase:
Verify that the command sent to an actuator is physically possible

→ ease the management of group of actuators

Example 1

Algorithm using temperature sensors where the interrogation of the sensors is not predictable but relies on computation done by the previously read values.
e.g. accelerate the pace when temperature delta increases quickly

Classical approach

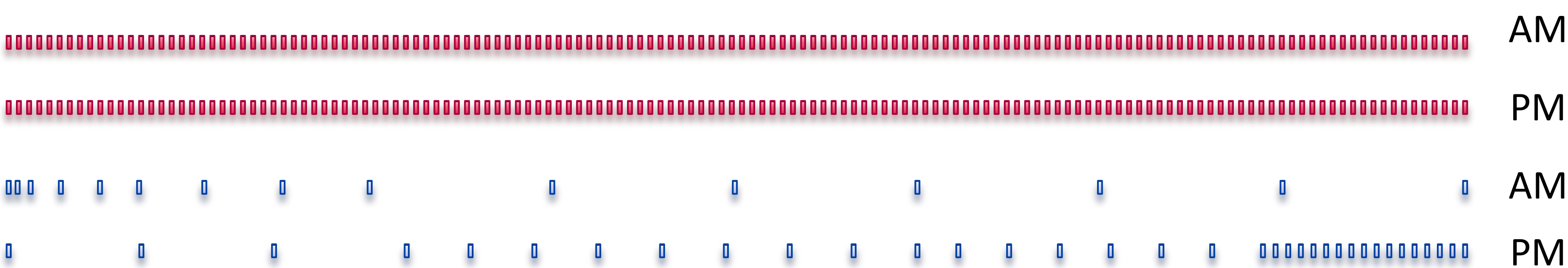
The sensor send the temperature every 5 minutes

24*12 = 288 measures

Application driven approach

The application interrogates the sensor when required

let say that 50 measures are enough



Example 1

Micro + Radio	Idle	running	wakeup + request + sleep
Flyport + Wifi	97µA	127.5mA	1s
Arduino + Xbee	206µA	57.1mA	0,04s

$$R = \frac{\text{Running Time}}{\text{Total Time}}$$

Micro + Radio	classical	Application driven
Flyport + Wifi	0,33%	0,058%
Arduino + Xbee	0,0133%	0,00231%

Sleeping
94,2% of the time
99,769 % of the time

$$Cons = R \; C_{Running} + (1 - R) \; C_{idle}$$

$$Autonomy = Cons / 1300\mu Ah$$

Micro + Radio	classical	App. driven
Flyport + Wifi	105 days	328 days
Arduino + Xbee	253 days	261 days

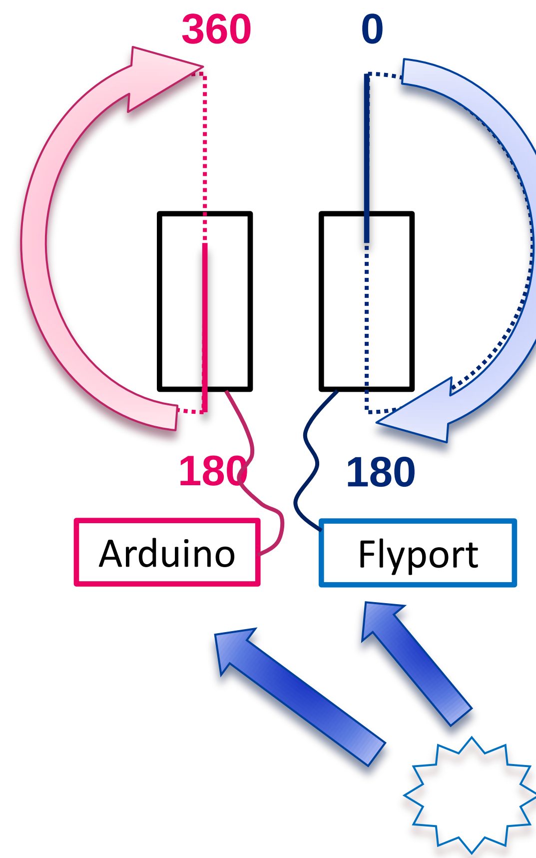
In our example

- More important to save on idle state than on running state
- Costly but simpler to deploy wireless protocol is affordable

Example 2

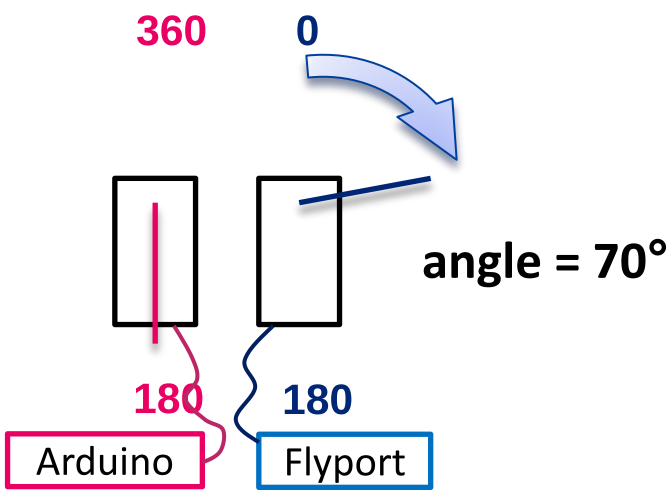
We want to coordinate 2 servo-motors such that their combined moves allow to turn from 0 to 360 degrees while they can only turn 180 degrees each.

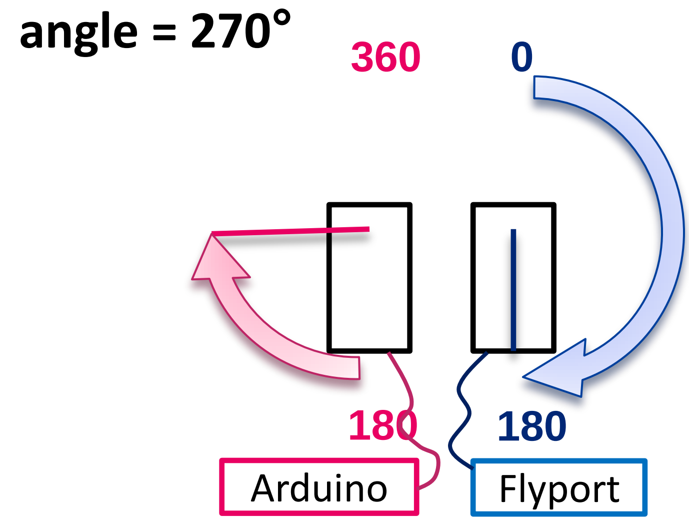
Transaction will fail if servo-motor receive out of range order




```
["Application", "Angle"].rd(angle) &
::
{
   ["Application", "Angle"].get(angle) ;
   ["Flyport", "Actuator"].put("position", angle) ;      Fail if angle not in 0-180
   ["Arduino", "Actuator"].put("position", "180") ;
}
{
   ["Application", "Angle"].get(angle) ;                 Fail if angle not available
   ["Flyport", "Actuator"].put("position", "180") ;
   ["Arduino", "Actuator"].put("position", angle) ;      Fail if angle not in 180-360
}.
```

Example 2





["Application", "Angle"].rd("70") &
::
{
    ["Application", "Angle"].get("70") ;
    ["Flyport", "Actuator"].put("position", "70") ;
    ["Arduino", "Actuator"].put("position", "180") ;
}
{
    ["Application", "Angle"].get(angle) ;
    ["Flyport", "Actuator"].put("position", "180") ;
    ["Arduino", "Actuator"].put("position", "70") ;
}.

["Application", "Angle"].rd("270") &
::
{
    ["Application", "Angle"].get("270") ;
    ["Flyport", "Actuator"].put("position", "270") ;
    ["Arduino", "Actuator"].put("position", "180") ;
}
{
    ["Application", "Angle"].get(angle) ;
    ["Flyport", "Actuator"].put("position", "180") ;
    ["Arduino", "Actuator"].put("position", "270") ;
}.

Conclusion

sensors can be stupid but
they need to be disciplined

- High level coordination protocol on micro-controllers
- Better usage of application knowledge has a significant impact of the consumption.
  - Saving on running mode is not enough
  - “more costly” wireless protocol, easier to deploy is not always a bad idea.
- Embedded distributed actions into transaction
  - Use the 1st phase to verify the action is actually possible
  - Ensure all-or-nothing property

Future work

- Work on the wake up signal

  Involve other teams of CEA-Leti

- More complex scenario
  - Abandoned sensors
    First sensor waked up by alarm, others sensors by application)
    (we are not very far from our 1st example)
  - Robot with motorized camera
    Tracking an object by moving either the camera or the robot
    But the camera can be at the end of the range and the robot blocked by an obstacle.
    (we are not very far from our 2nd example)