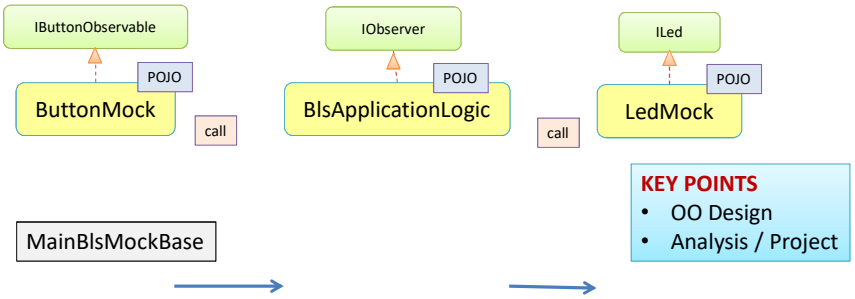
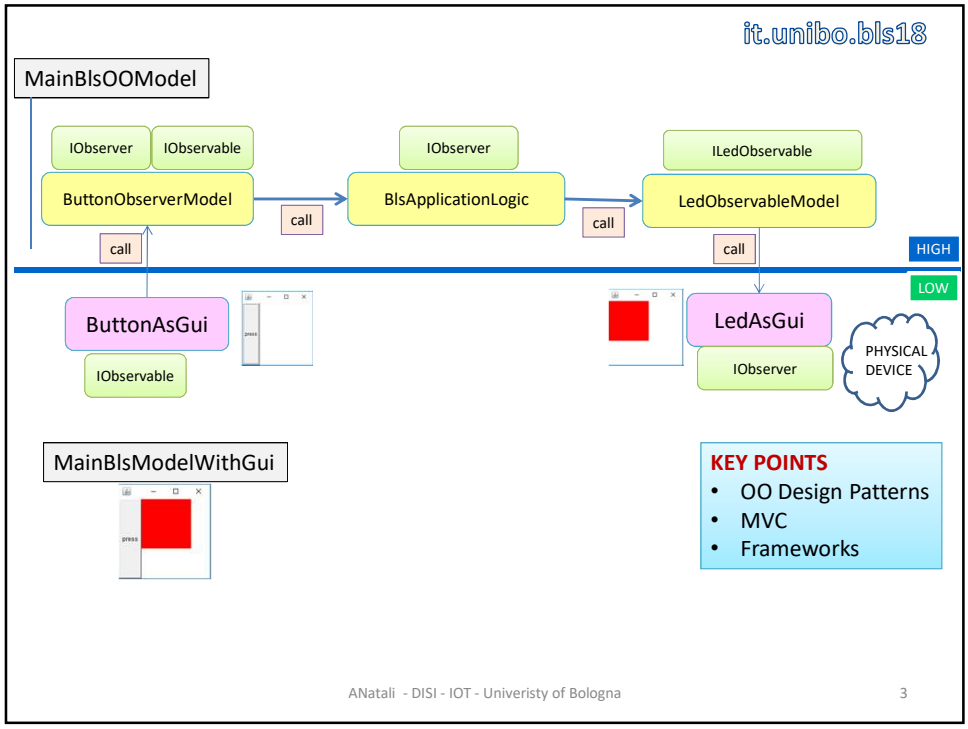
BLS2018

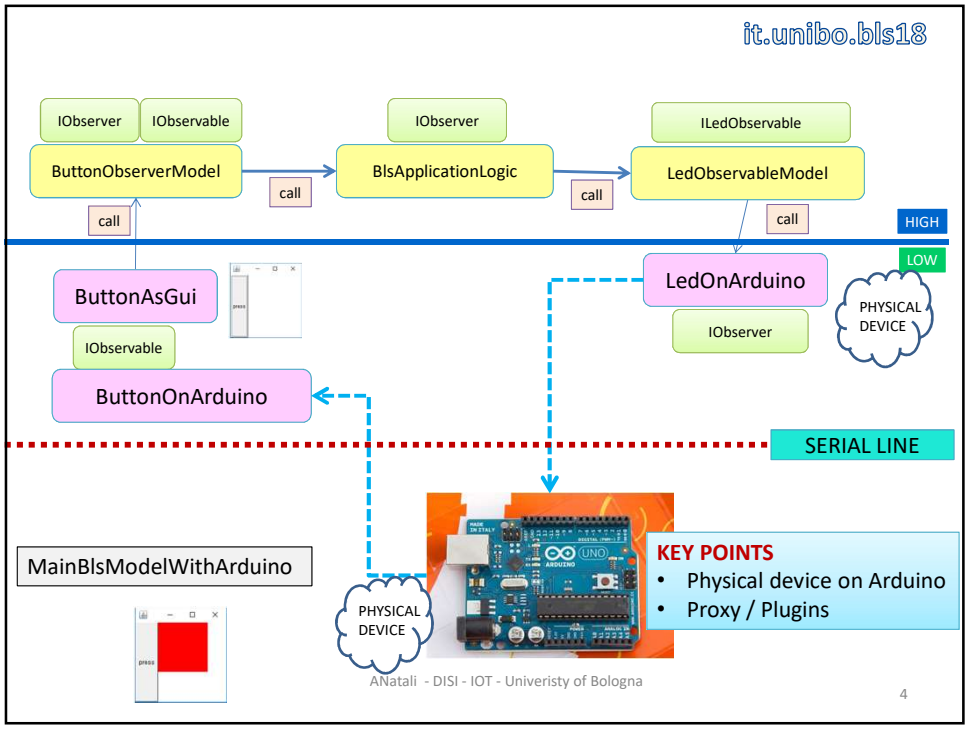
From oop to distributed systems

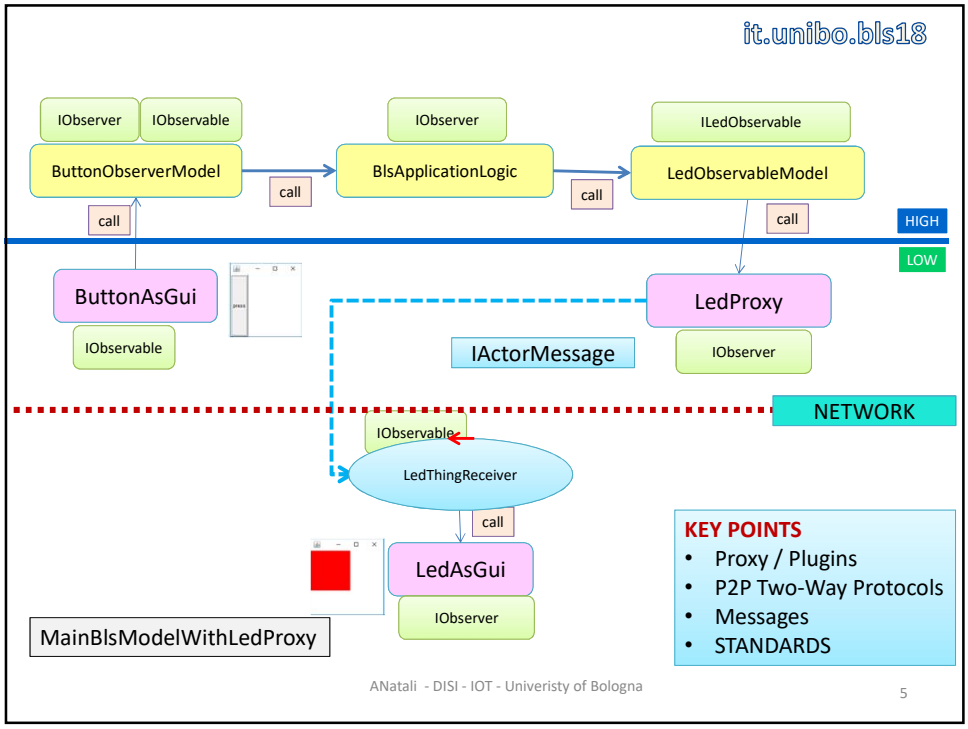
demoBls.pdf (1-3) it.unibo.bls18

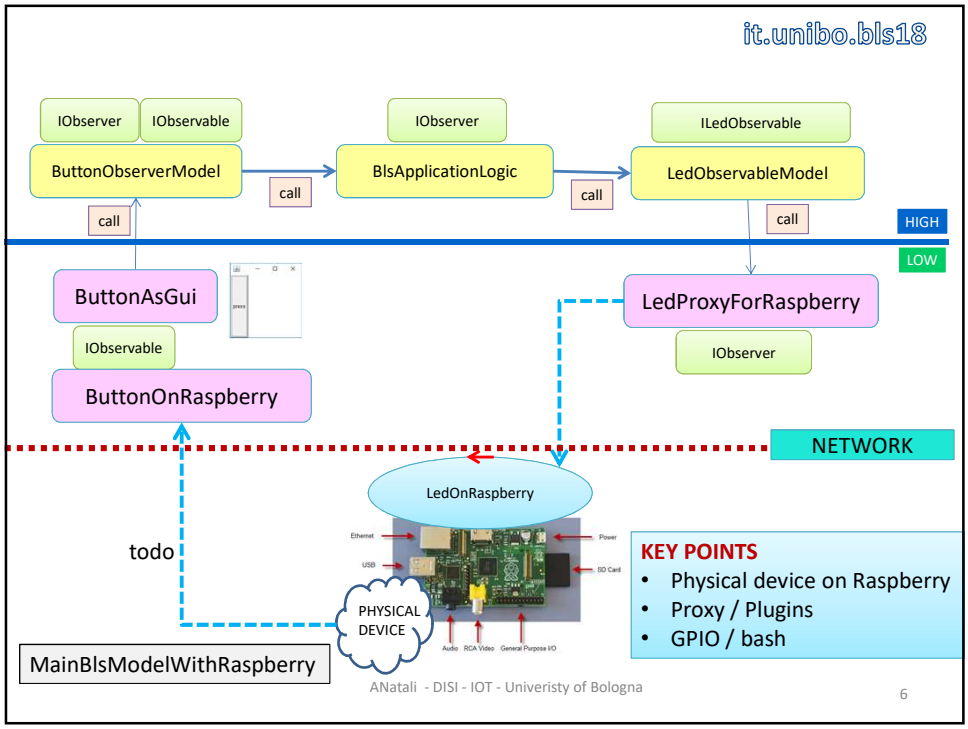
Req0: accendere un Led premendo un Pulsante




it.unibo.bls18
MainBlsOOModel
IObserver
IObservable
ButtonObserverModel
call
IObserver
BlsApplicationLogic
call
ILedObservable
LedObservableModel
call
call
HIGH
LOW
ButtonAsGui
IObservable
LedAsGui
IObserver
PHYSICAL DEVICE
MainBlsModelWithGui
KEY POINTS
• OO Design Patterns
• MVC
• Frameworks
ANatali - DISI - IOT - Univeristy of Bologna
3


it.unibo.bls18
IObserver
IObservable
ButtonObserverModel
call
IObserver
BlsApplicationLogic
call
ILedObservable
LedObservableModel
call
call
HIGH
LOW
ButtonAsGui
IObservable
ButtonOnArduino
LedOnArduino
IObserver
PHYSICAL DEVICE
SERIAL LINE
MainBlsModelWithArduino
PHYSICAL DEVICE
KEY POINTS
• Physical device on Arduino
• Proxy / Plugins
ANatali - DISI - IOT - Univeristy of Bologna
4


it.unibo.bls18
IObserver
IObservable
ButtonObserverModel
call
IObserver
BlsApplicationLogic
call
ILedObservable
LedObservableModel
call
call
HIGH
LOW
ButtonAsGui
IObservable
LedProxy
IObserver
IActorMessage
NETWORK
IObservable
LedThingReceiver
call
LedAsGui
IObserver
MainBlsModelWithLedProxy
KEY POINTS
• Proxy / Plugins
• P2P Two-Way Protocols
• Messages
• STANDARDS
ANatali - DISI - IOT - Univeristy of Bologna
5

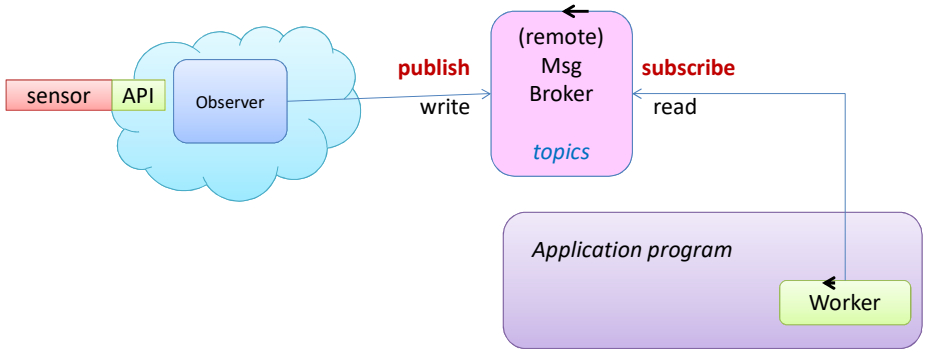

it.unibo.bls18
IObserver
IObservable
ButtonObserverModel
call
IObserver
BlsApplicationLogic
call
ILedObservable
LedObservableModel
call
call
HIGH
LOW
ButtonAsGui
IObservable
ButtonOnRaspberry
LedProxyForRaspberry
IObserver
NETWORK
LedOnRaspberry
todo
PHYSICAL DEVICE
MainBlsModelWithRaspberry
KEY POINTS
• Physical device on Raspberry
• Proxy / Plugins
• GPIO / bash
ANatali - DISI - IOT - Univeristy of Bologna
6

/boot/mywifi.conf

Req1: accendere un insieme di Led premendo un Pulsante

MOM

Publish–subscribe pattern



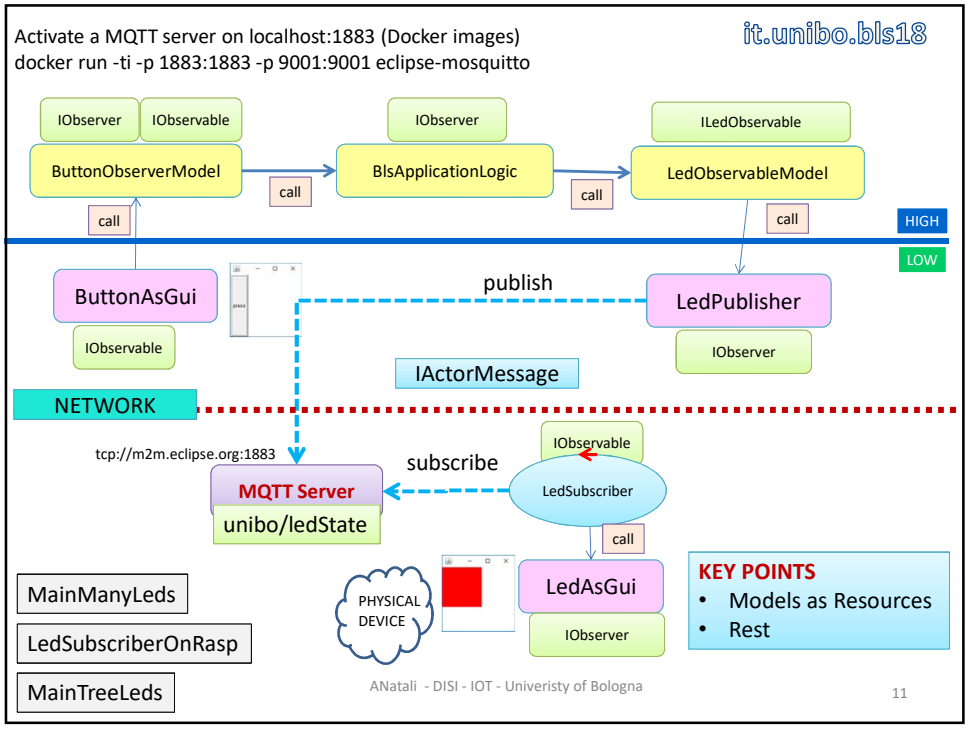
The MQ Telemetry Transport (MQTT) is an ISO standard (ISO/IEC PRF 20922) supported by the OASIS organization.
Eclipse Mosquitto™ is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 3.1 and 3.1.1.

Mosquitto on Docker

- Images (docker pull …)
  - docker images
- Container (a runnable instance of an image)
  - docker run -p 8484 -a stdin -a stdout -i -t --name natdocker node:nat /bin/bash
  - docker ps
  - docker start /stop --container

docker ps -a
docker start d8a5f1fefc5b
docker ps
docker exec -it d8a5f1fefc5b /bin/bash

docker run -ti -p 1883:1883 -p 9001:9001 eclipse-mosquitto

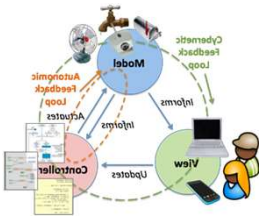


KEY POINTS at the end of phase1

- SISTEMI (DISTRIBUITI ETEREOGENEI
- ARCHITETTURE – DESIGN PATTERN
- STANDARD DI INTERAZIONE/COMUNICAZIONE:
  - Messaggi (dispatch, invitation, request, …)
  - Eventi (messaggi 'senza destinatario'???)
  - Protocolli P2P (TCP, UDP, CoAP, HTTP, …) o publish/subscribe
  - Payload (vocabolari)
  - M2M, Man2M, M2Man (ManToMan)
- SCHEMI DI COMPORTAMENTO
  - Message-Event BASED (**FSM**)
  - Message-Event DRIVEN

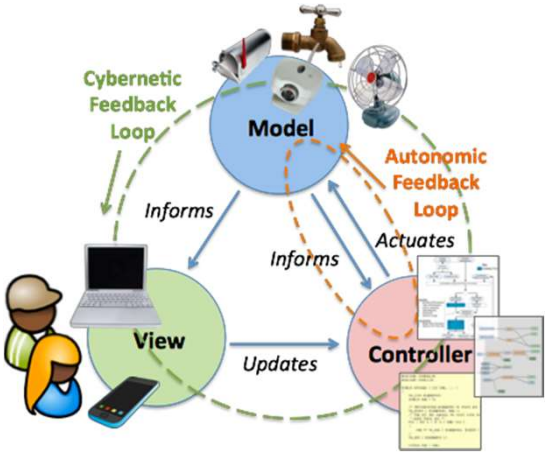
IL SISTEMA BLS (iot minimal)

- DOPO avere costruito il sistema:
  - Cosa è un LED? Cosa è un BUTTON?
  - Cosa fa il sistema?
  - Come fa un utente a sapere quello che fa?
  - IL PATTERN MVC



MVC

The **Model-VIew-Controller macro pattern** provides a framework for the structured division of responsibility between people and software in IoT applications. It also provides a framework for high level interoperability between data sources, control elements, and UI elements.

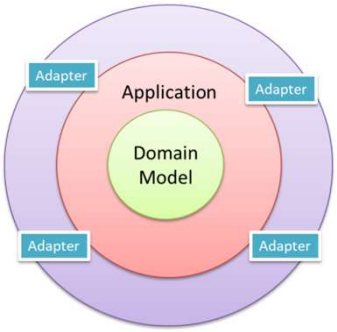


The **Model** is a representation or an abstraction of the physical things and their attributes, which *informs* a Controller.
The **Controller** is software which makes *actuation* decisions based on the information, and sends actuation commands to the thing using it's modeled affordances.
The software goal is to maintain a desired state of the thing through it's model.

SISTEMI: una visione 'olistica'

- VISTE DI UN SISTEMA
  - Dal di fuori (cosa fa)
  - Dal di dentro (come è fatto)
- ARCHITETTURA ESAGONALE

The connection between the *inside* and the *outside* part of the system is realized via abstractions called *ports* and their implementation counterparts called *adapters*.



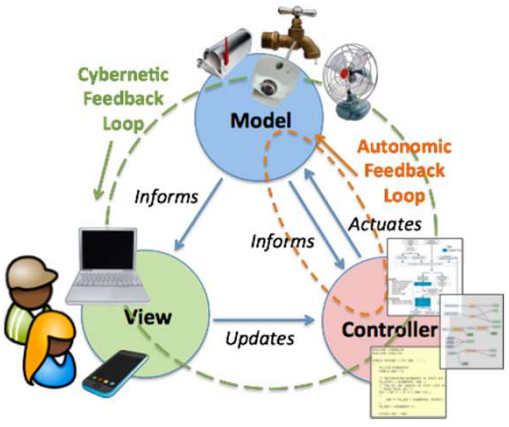
BLS

BLS (Tree)

- Visto 'dal di fuori' mette a disposizione:
  - una funzione di attivazione/stop attraverso una user interface (BUTTON? WebPage? …)
  - una vista (dello stato) delle risorse che gestisce (WebPage)
  - funzioni di accesso/modifica delle risorse
    - mediante interazioni REST (HTTP/CoAP)

MVC (WoT) loops

This cycle of **Observation => Information => Actuation** creates a *feedback control loop* where the observed property is controlled in a system known as a *closed loop*.



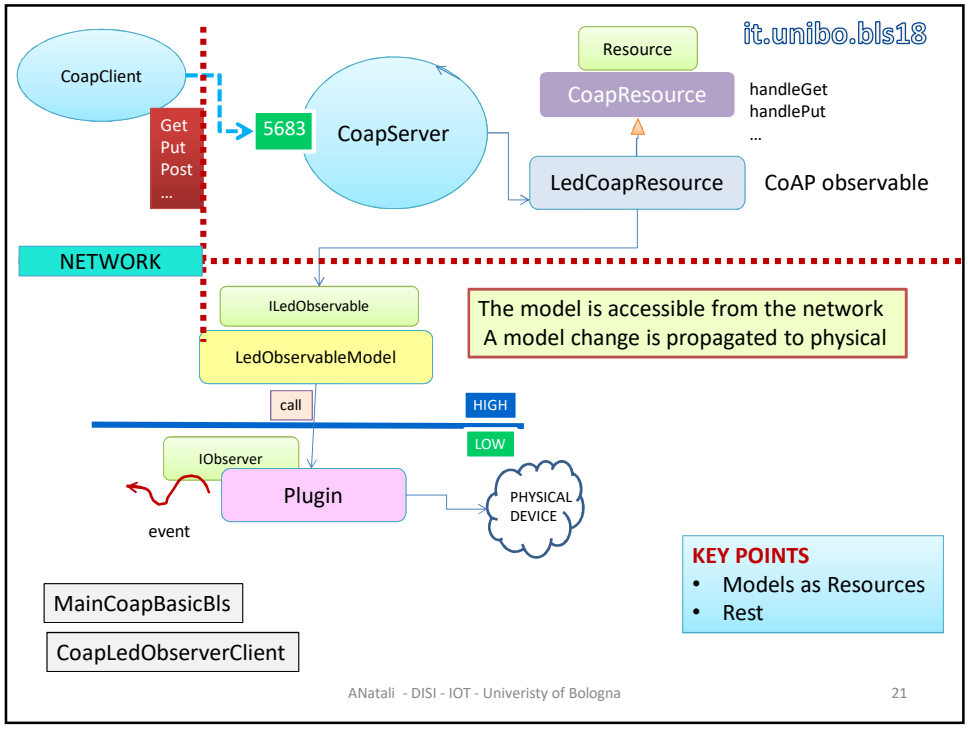
Cybernetic feedback loops, which involve a person in the loop participating in making decisions. For example, the person in the cybernetic loop is updating settings of an autonomic loop controller.

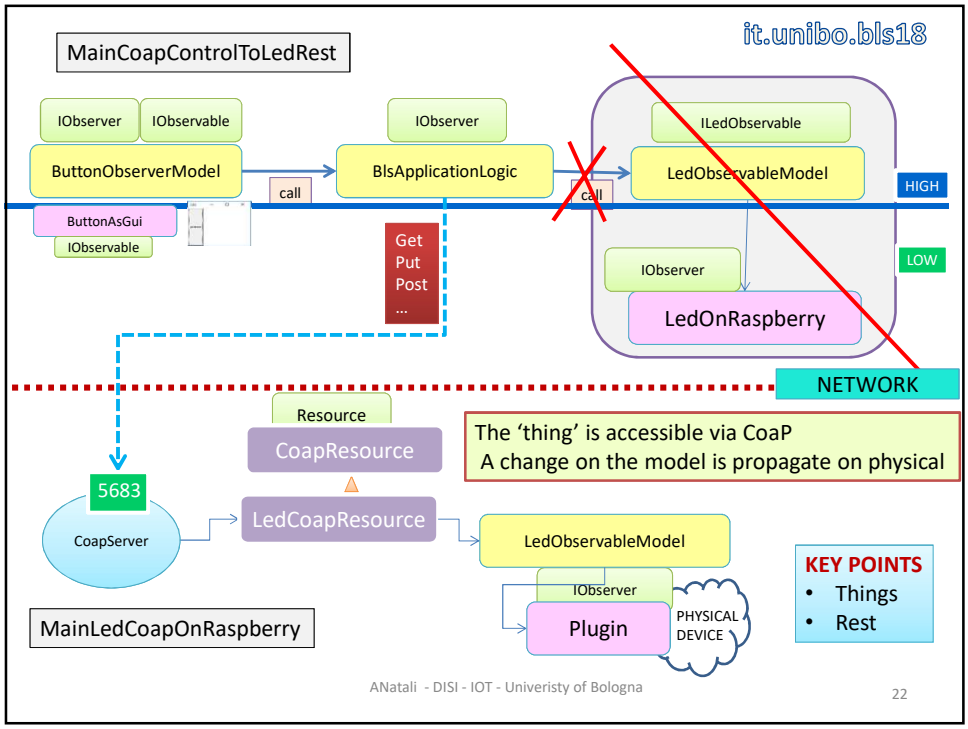
Autonomic feedback loops involve only software making decisions. An example of this is a motion sensor turning on a light.

Req2: accedere a un Led via web/REST

Coap Californium

- A CoAP server hosts a tree of Resources which are exposed to clients by means of one or more Endpoints which are bound to a network interface.
- CoapResource is a basic implementation of a resource.
- CoapResource uses four distinct methods to handle requests: handleGET(), handlePOST(), handlePUT() and handleDELETE().
- Each resource is allowed to define its own executor.
- CoapResource supports CoAP's *observe mechanism*. Enable a CoapResource to be observable by a CoAP client by marking it as observable with **setObservable(boolean)**.
- The class **ResourceObserver** has nothing to do with CoAP's observe mechanism but is an implementation of the general observe-pattern.


it.unibo.bls18
CoapClient
Get
Put
Post
...
5683
CoapServer
Resource
CoapResource
handleGet
handlePut
...
LedCoapResource
CoAP observable
NETWORK
ILedObservable
LedObservableModel
The model is accessible from the network
A model change is propagated to physical
call
HIGH
LOW
IObserver
Plugin
PHYSICAL DEVICE
event
KEY POINTS
• Models as Resources
• Rest
MainCoapBasicBls
CoapLedObserverClient
ANatali - DISI - IOT - Univeristy of Bologna
21


it.unibo.bls18
MainCoapControlToLedRest
IObserver
IObservable
ButtonObserverModel
ButtonAsGui
IObservable
call
IObserver
BlsApplicationLogic
ILedObservable
LedObservableModel
call
HIGH
LOW
IObserver
LedOnRaspberry
Get
Put
Post
...
NETWORK
Resource
CoapResource
The 'thing' is accessible via CoaP
A change on the model is propagate on physical
5683
CoapServer
LedCoapResource
LedObservableModel
IObserver
Plugin
PHYSICAL DEVICE
KEY POINTS
• Things
• Rest
MainLedCoapOnRaspberry
ANatali - DISI - IOT - Univeristy of Bologna
22

Inherits or use? → The resource USES the model

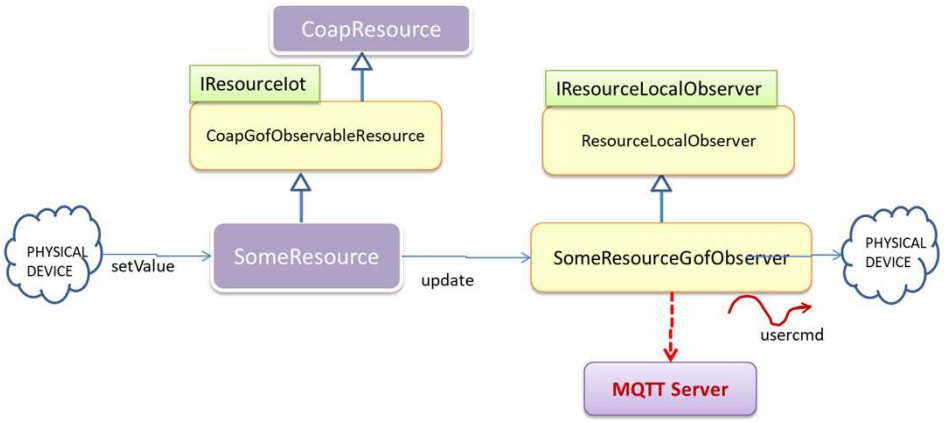
```
public class LedCoapResource extends CoapResource{    //in it.unibo.bls18.coapBasic.led
privateILedObservable ledModel ;

public LedCoapResource(String name, ILedObservable model ) {
            super(name);
            ledModel = model;
}
  @Override //CoapResource
   public void handleGET(CoapExchange exchange) {
       exchange.respond( ResponseCode.CONTENT, getValue(), MediaTypeRegistry.TEXT_PLAIN) ;
   }
   @Override //CoapResource
   public  void handlePUT(CoapExchange exchange) {
      try {
     String value = exchange.getRequestText();//new String(payload, "UTF-8");
        if( value.equals("switch"))  switchValue();   else setValue(value);
        exchange.respond(CHANGED,  value);
     } catch (Exception e) {
         exchange.respond(BAD_REQUEST, "Invalid String");
     }
   }
```

```
protected void setValue(String v) {
 if( v.equals("true")) ledModel.turnOn();
 else ledModel.turnOff();
  }
```

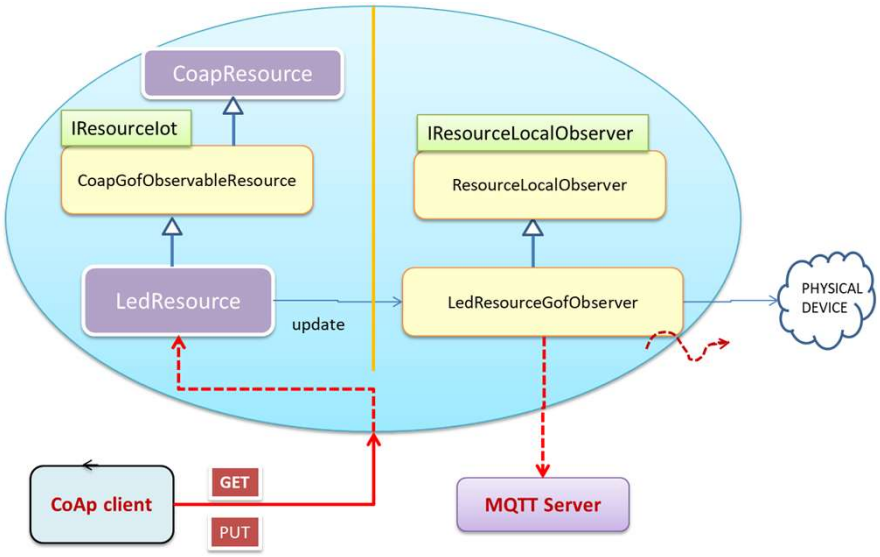

it.unibo.coap

Observable (CoAP) resource



it.unibo.coap

Led Thing



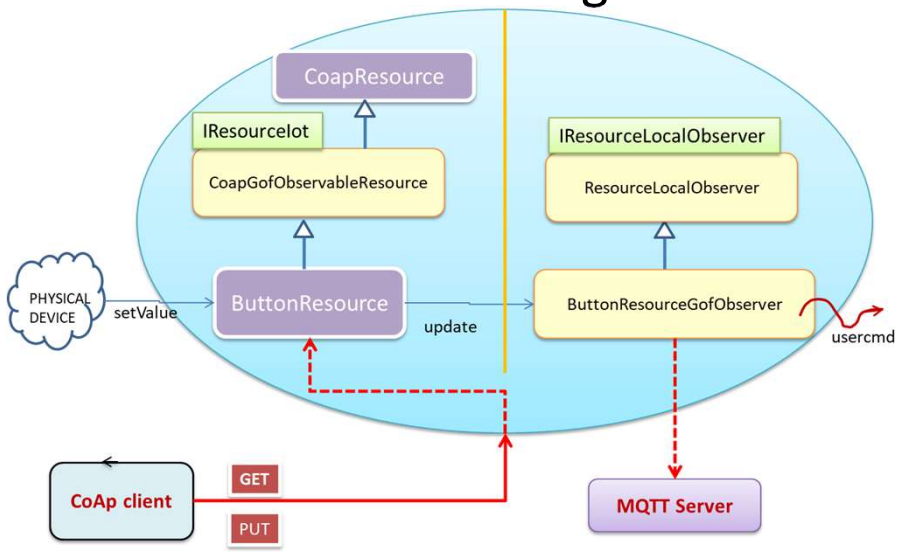
Inherits or use? → The resource INHERITS (it is the model)

```
public  class LedResource extends CoapGofObservableResource {
public static final String resourcePath = "led";
private String value = "false";
   public LedResource() {
      super(resourcePath);
   }
@Override //CoapResource
   public void handleGET(CoapExchange exchange) {
            exchange.respond( value );
   }
 @Override //CoapResource
   public  void handlePUT(CoapExchange exchange) {
   try {
        value = exchange.getRequestText();
         setValue(value);
         exchange.respond(CHANGED,  value);
     } catch (Exception e) {
          exchange.respond(BAD_REQUEST, "Invalid String");
     }   } }
```

```
 @Override // CoapGofObservableResource
public void setValue(String v) {
  value = v ;
  update(value);//notify the GOF observer
}
```

it.unibo.coap

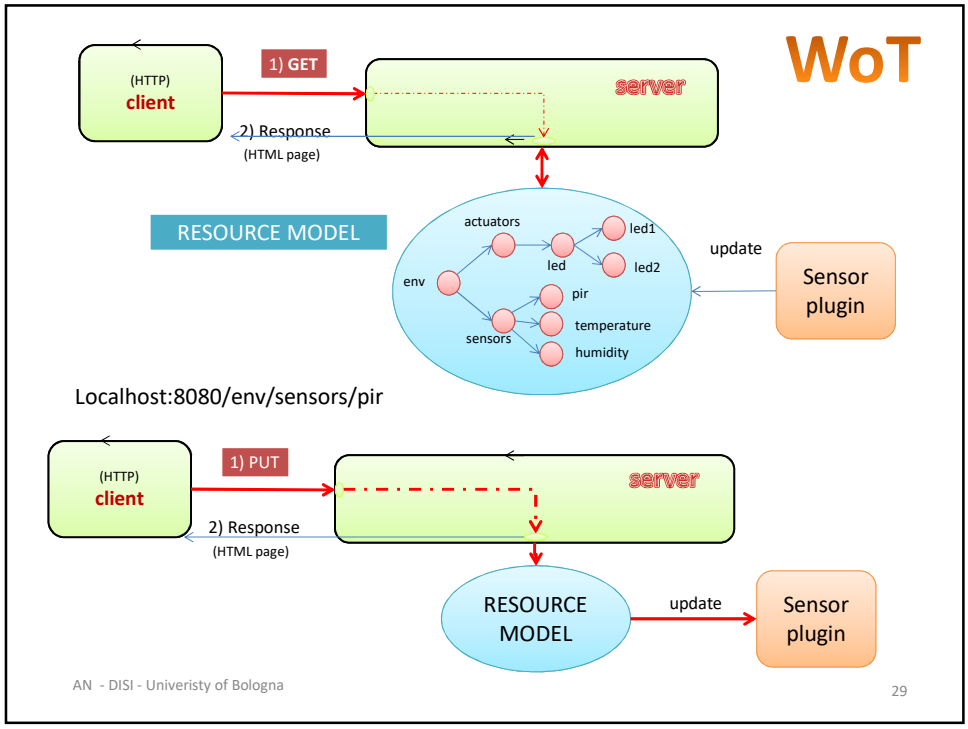
Button Thing

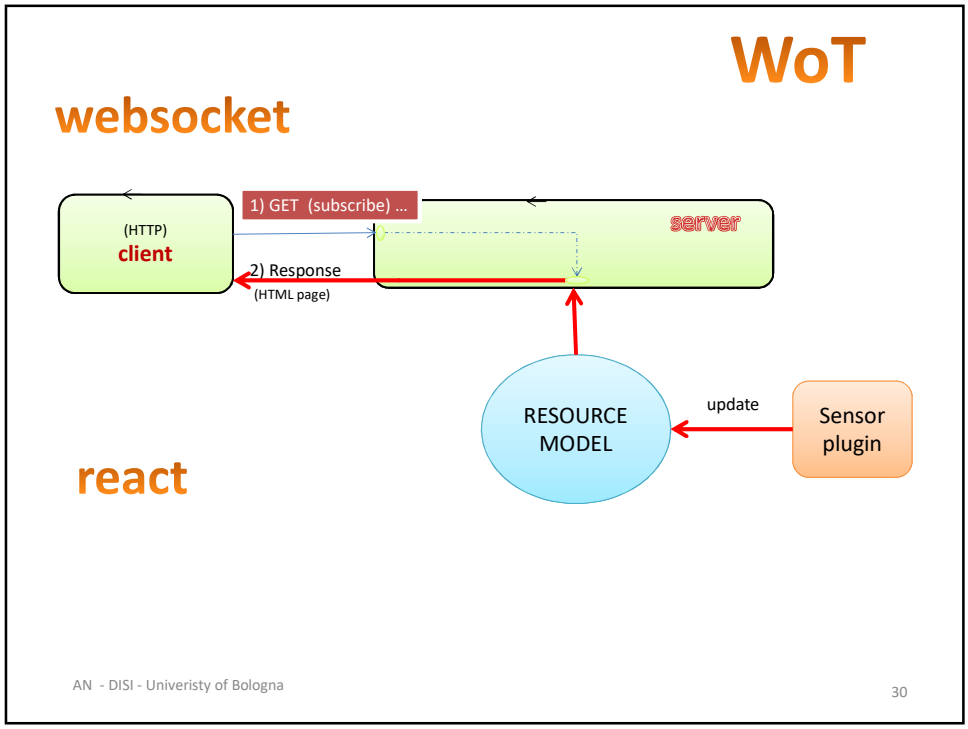


WoT

←

Integration patterns:	REST on device, Gateway (CoAP), Cloud (MQTT)
Resource model design :	(ontology) tree, knowledge base, ..
Representation design :	json, prolog, HTML, MessagePack, …
Interface design :	GET, PUT, etc
Resourse binding design :	HAETOAS (web linking,..), findability, …


WoT
(HTTP)
client
1) GET
server
2) Response
(HTML page)
RESOURCE MODEL
actuators
led1
led
led2
env
pir
sensors
temperature
humidity
update
Sensor plugin
Localhost:8080/env/sensors/pir
(HTTP)
client
1) PUT
server
2) Response
(HTML page)
RESOURCE MODEL
update
Sensor plugin
AN - DISI - Univeristy of Bologna
29


WoT
websocket
(HTTP)
client
1) GET (subscribe) ...
server
2) Response
(HTML page)
RESOURCE MODEL
update
Sensor plugin
react
AN - DISI - Univeristy of Bologna
30