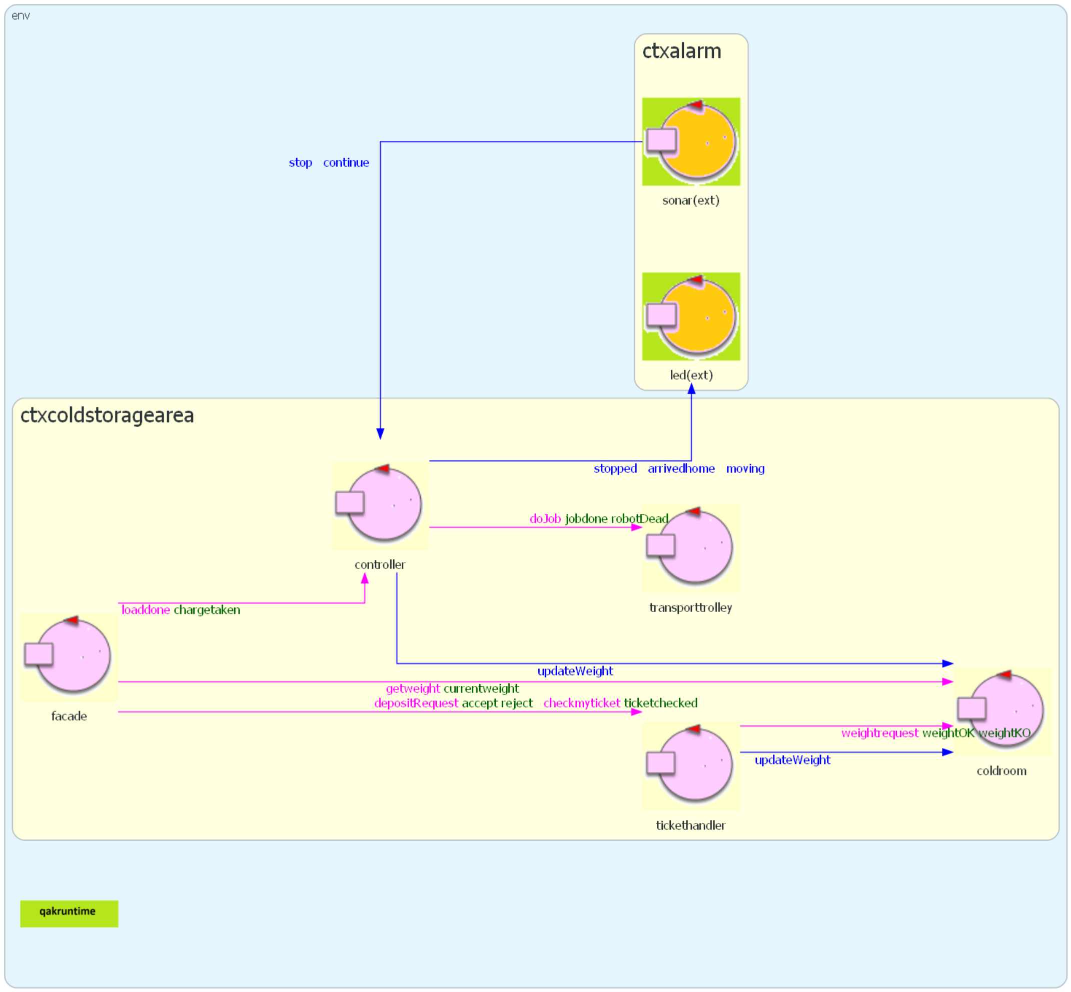
Goal Sprint 3

ServiceStatusGui e grafiche migliorate Sprint 3

Descrizione

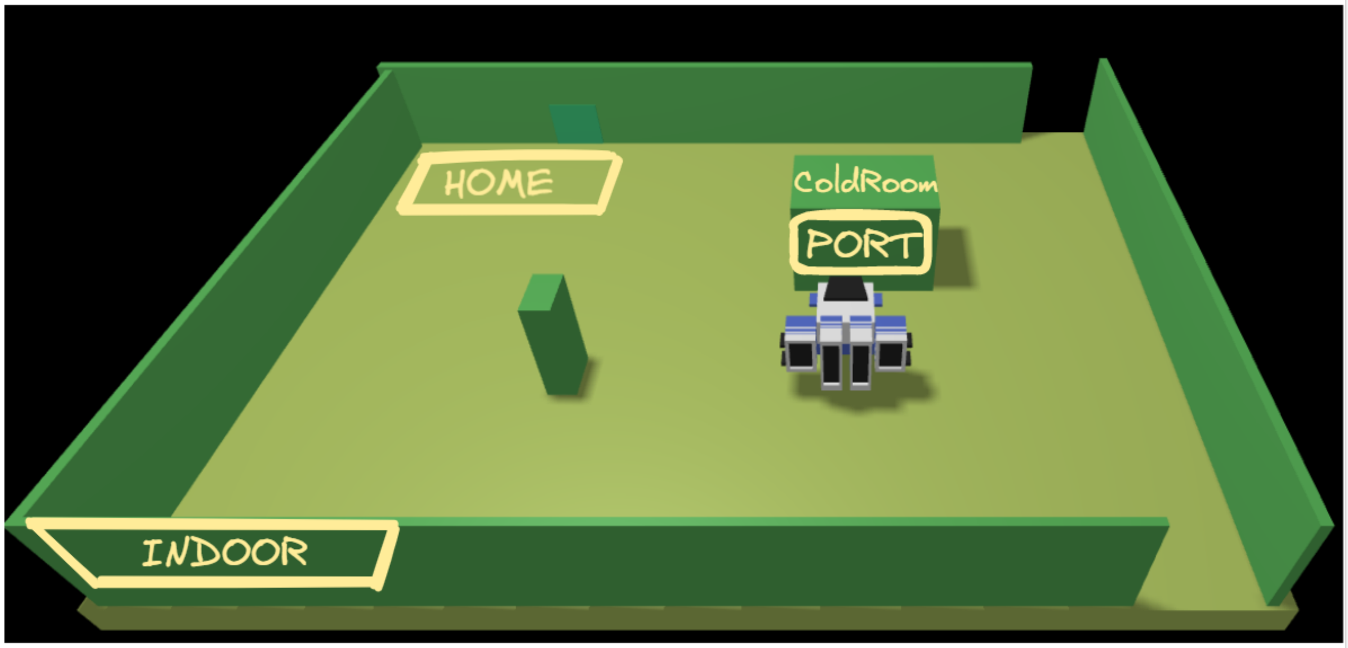
Nel terzo sprint ci occuperemo della ServiceStatusGUI e delle interfacce grafiche finali.

Modello dello sprint precedente.



coldstorage2Arch

Requisiti



Requisiti

Analisi dei Requisiti

requisiti sprint 0

Domande al committente:

Per la posizione quanto dobbiamo essere precisi? Serve sapere la posizione corrente ad ogni step? per il cliente non è necessario quest'ultimo punto dipende da noi. --> per semplicità forniamo le coordinate ad ogni cambiamento del macrostato (in home, fase di load, fase di unload, posizione in caso di errore)

Analisi del Problema

Cosa implica lo stato del servizio?

Lo stato del servizio comprende:

- Lo stato e la posizione del TransportTrolley.
- Lo stato della ColdRoom (peso corrente su totale).
- Il numero di richieste negate dall'inizio del servizio.

Numero di richieste negate

Aggiorniamo il ticket handler per tener traccia delle richieste negate

```
QActor tickethandler context ctxcoldstoragearea {
        [# var Rejected = 0     #]
        ...

        State reject {
                [# Rejected++ #]
                ...
}
```

Rendo tutti i componenti observable

Stato del TransportTrolley (sfruttiamo RobotPos):

```
updateResource [# planner.robotOnMap() #]
```

ColdRoom:

```
updateResource[# "" + PesoEffettivo + "_" + PesoPromesso + ""#]
```

TicketHandler:

```
updateResource [# "" + Rejected #]
```

Caricare i dati iniziali nella GUI

All'avvio della ServiceStatusGui per visualizzare i valori possiamo:

- Aspettare che i componenti aggiornino i propri valori normalmente
- Richiedere esplicitamente il valore corrente
  Decidiamo di fare richiesta esplicita poiché in mancanza di richieste

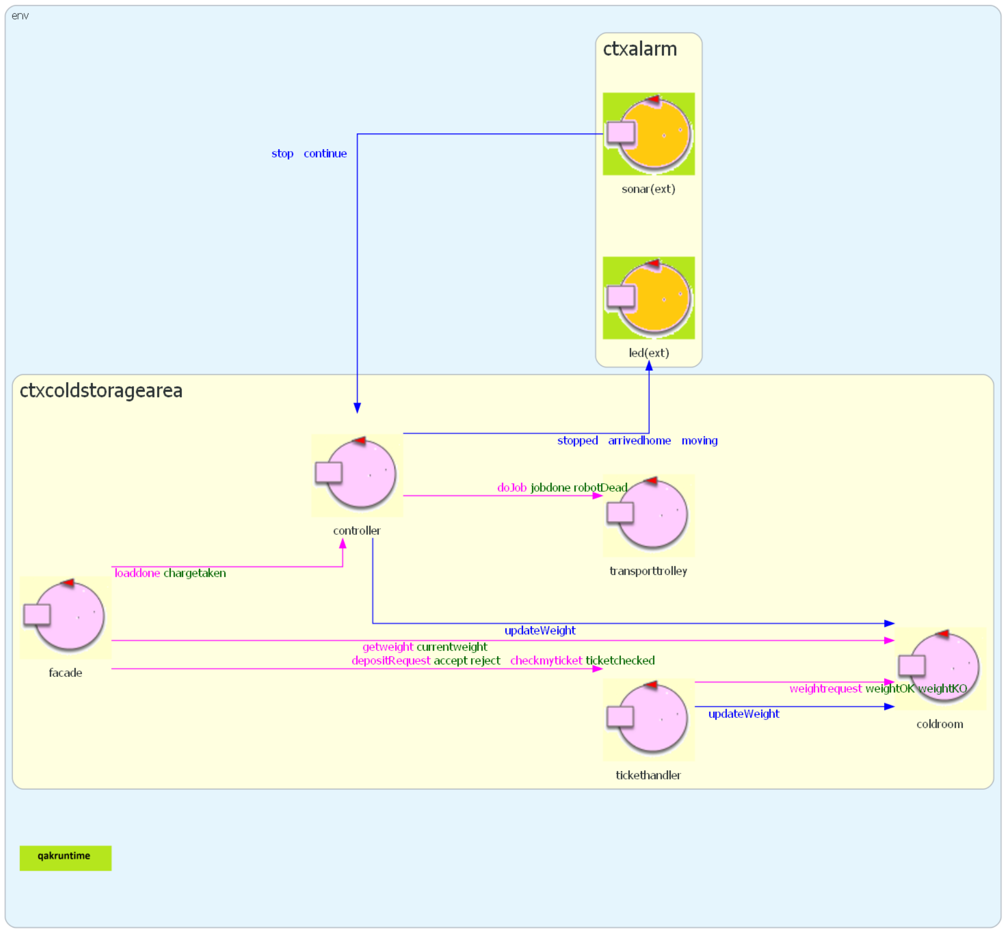
da parte degli utenti potrei dover aspettare un tempo indefinito prima di vedere lo stato del sistema.

Come mandare le richieste

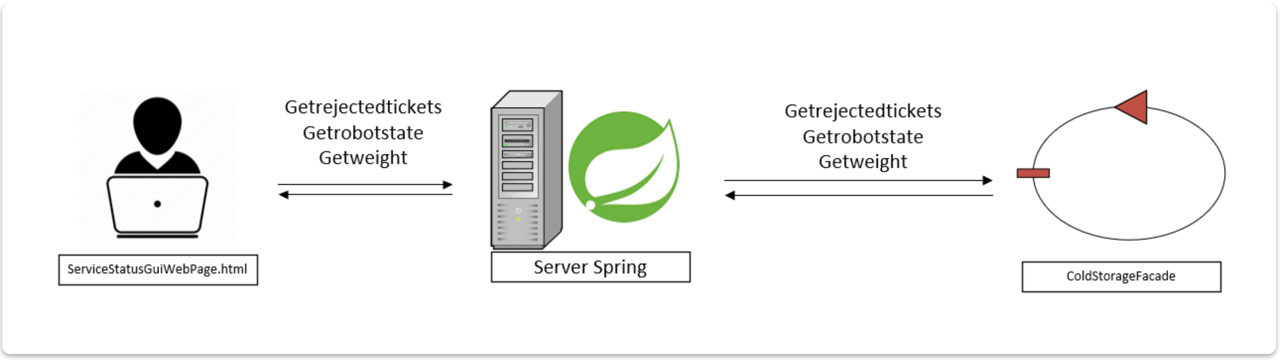
Sfruttiamo la facade già creata in precedenza per richiedere al sistema i valori iniziali per la ServiceStatusGui.

Architettura logica dopo l'analisi del problema

- [ ] Problema del grafico che non mostra i delegate



coldstorage3Arch



Test Plan

1. A seguito di una richiesta rifiutata devono aumentare le richieste rifiutate.
2. Nel caso di loadDone fallita la posizione stampata deve essere diversa da home.
3. All'avvio della Gui devono essere caricati i parametri attuali del sistema.

Progettazione

Facade aggiornato

```
QActor facade context ctxcoldstoragearea {
        [#
                var Ticket = ""
                var PesoEff = 0
                var PesoProm = 0
                var Valid = true
        #]

        State s0 initial{
                delegate "getrejectedtickets" to tickethandler
                delegate "getrobotstate" to robotpos
                delegate "getweight" to coldroom
                println("ColdStorageFacade - in attesa") color
blue
                printCurrentMessage
        } Goto work


        State work {
        }Transition t0  whenRequest depositRequestF->
depositreqhandler
                                whenRequest loaddoneF ->
loadcontroller
                                whenRequest checkmyticketF ->
checktickethandler
                                whenRequest getweightF ->
getweightcoldroom

        State depositreqhandler {
                ...
        }Transition t1 whenReply accept -> returnticket
                                        whenReply reject ->
rejectticket

        State rejectticket {
                ...
        } Goto work

        State returnticket {
                ...
        } Goto work

        State checktickethandler {
```

```
		...
	}Transition tc whenReply ticketchecked -> checkresponse

	State checkresponse {
		...
	}Goto work

	State getweightcoldroom{
		...
	} Transition tg whenReply currentweight -> returnweight

	State returnweight{
		...
	}Goto work

	State loadcontroller{
		...
	}Transition t0 whenReply chargetaken -> returnload

	State returnload{
		...
	}Goto work
}
```

La classe per la parte web

```
package unibo.statusgui;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
```

```
import
org.springframework.scheduling.annotation.EnableScheduling;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.CrossOrigin;
import
org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.GetMapping;

@CrossOrigin
@Controller
@EnableScheduling
public class ControllerStatusGui {
    @Value("${spring.application.name}")
    String appName;
    MessageSender sender = new MessageSender();

    @GetMapping("/")
    public String homePage(Model model) {
        this.aggiornaPesoCorrente(model);
        this.aggiornaBigliettiRifiutati(model);
        this.aggiornaRobotPos(model);
        return "/static/ServiceStatusGui";
    }

    @ExceptionHandler
    public ResponseEntity handle(Exception ex) {
        HttpHeaders responseHeaders = new HttpHeaders();
        return new ResponseEntity(
                "HIControllerDemo ERROR " + ex.getMessage(),
                responseHeaders, HttpStatus.CREATED);
    }

    private void aggiornaPesoCorrente(Model model){
        String msg =
"msg(getweight,request,statusgui,facade,getweight(NO_PARAM),1)\
n";
        String response = sender.sendMessage(msg);
        System.out.println(response);
        String[] weights = response.split("\\(|\\)")
[2].split(",");
        model.addAttribute("ew", weights[0]);
        model.addAttribute("pw", weights[1]);
    }

    private void aggiornaBigliettiRifiutati(Model model){
        String msg =
"msg(getrejectedtickets,request,statusgui,facade,getrejectedtic
kets(NO_PARAM),1)\n";
        String response = sender.sendMessage(msg);
        System.out.println(response);
        String rejectednum = response.split("\\(|\\)")[2];
        model.addAttribute("rt", rejectednum);
    }

    private void aggiornaRobotPos(Model model){
        String msg =
"msg(getrobotstate,request,statusgui,facade,getrobotstate(NO_PA
RAM),1)\n";
        String response = sender.sendMessage(msg);
        System.out.println(response);
        String[] robotpos = response.split("\\(|\\)|,");
        model.addAttribute("maintext", "RobotPos=("+
robotpos[7] +"," + robotpos[7] +") direction="+robotpos[10]);
    }
}
```

Gli observer

```java
package unibo.statusgui;

import org.eclipse.californium.core.CoapHandler;
import org.eclipse.californium.core.CoapResponse;
import org.springframework.stereotype.Component;
import unibo.basicomm23.coap.CoapConnection;
import unibo.basicomm23.utils.CommUtils;

@Component
public class ColdRoomObserver implements CoapHandler{
    String CSIPADDRESS = "127.0.0.1";
    int CSPORT = 8040;
    String ctxqakdest = "ctxcoldstoragearea";

    public ColdRoomObserver(){
        System.out.println("crobserver started");

        CoapConnection coldroomconn = new
CoapConnection(CSIPADDRESS+":"+CSPORT, ctxqakdest+"/coldroom"
);
        coldroomconn.observeResource( this );
    }


    @Override
    public void onLoad(CoapResponse response) {
        CommUtils.outcyan("ColdRoomObserver changed! " +
response.getResponseText() );
        WebSocketConfiguration.wshandler.sendToAll("cr_" +
response.getResponseText());
    }

    @Override
    public void onError() {
        System.out.println("ColdRoomObserver observe error!");
    }
}
```

```
package unibo.statusgui;



import org.eclipse.californium.core.CoapHandler;
import org.eclipse.californium.core.CoapResponse;
import org.springframework.stereotype.Component;
import unibo.basicomm23.coap.CoapConnection;
import unibo.basicomm23.utils.CommUtils;

@Component
public class RobotPosObserver implements CoapHandler{
    String CSIPADDRESS = "127.0.0.1";
    int CSPORT = 8040;
    String ctxqakdest = "ctxcoldstoragearea";

    public RobotPosObserver(){
        System.out.println("rpobserver started");

        CoapConnection robotposconn = new
CoapConnection(CSIPADDRESS+":"+CSPORT, ctxqakdest+"/robotpos"
);
        robotposconn.observeResource( this );
    }



    @Override
    public void onLoad(CoapResponse response) {
        CommUtils.outcyan("RobotPosObserver changed! " +
response.getResponseText() );
        WebSocketConfiguration.wshandler.sendToAll("rp_" +
response.getResponseText());
    }
    @Override
    public void onError() {
        System.out.println("RobotPosObserver observe error!");
    }
}
```

```
package unibo.statusgui;



import org.eclipse.californium.core.CoapHandler;
import org.eclipse.californium.core.CoapResponse;
import org.springframework.stereotype.Component;
import unibo.basicomm23.coap.CoapConnection;
import unibo.basicomm23.utils.CommUtils;

@Component
public class TicketHandlerObserver implements CoapHandler{
    String CSIPADDRESS = "127.0.0.1";
    int CSPORT = 8040;
    String ctxqakdest = "ctxcoldstoragearea";

    public TicketHandlerObserver(){
        System.out.println("thobserver started");

        CoapConnection tickethandlerconn = new
CoapConnection(CSIPADDRESS+":"+CSPORT,
ctxqakdest+"/tickethandler" );
        tickethandlerconn.observeResource( this );
    }



    @Override
    public void onLoad(CoapResponse response) {
        CommUtils.outcyan("TicketHandlerObserver changed! " +
response.getResponseText() );
        WebSocketConfiguration.wshandler.sendToAll("th_" +
response.getResponseText());
    }

    @Override
    public void onError() {
        System.out.println("TicketHandlerObserver observe
error!");
    }
}
```

Socket Handler

```
package unibo.statusgui;



import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.springframework.web.socket.BinaryMessage;
import org.springframework.web.socket.CloseStatus;
import org.springframework.web.socket.TextMessage;
import org.springframework.web.socket.WebSocketSession;
import
org.springframework.web.socket.handler.AbstractWebSocketHandler
;

import java.io.IOException;
import java.util.Iterator;
import java.util.List;
import java.util.concurrent.CopyOnWriteArrayList;

public class WebSocketHandler extends AbstractWebSocketHandler
{
    private final List<WebSocketSession> sessions = new
```

```
CopyOnWriteArrayList<>();
    private JSONParser jsonparser = new JSONParser();

    @Override
    public void afterConnectionEstablished(WebSocketSession
session) throws Exception {
        sessions.add(session);
        System.out.println("WebSocketHandler | Added the
session: " + session);
        super.afterConnectionEstablished(session);
    }

    @Override
    public void afterConnectionClosed(WebSocketSession session,
CloseStatus status) throws Exception {
        sessions.remove(session);
        System.out.println("WebSocketHandler |
afterConnectionClosed:" + session);
        super.afterConnectionClosed(session, status);
    }

    @Override
    protected void handleTextMessage(WebSocketSession session,
TextMessage message) throws IOException {
        String movecmd = message.getPayload();
        System.out.println("WebSocketHandler |
handleTextMessage Received: " + movecmd);
        try {

            JSONObject json = (JSONObject)
jsonparser.parse(movecmd);
            String move = json.get("robotmove").toString();
            System.out.println("WebSocketHandler |
handleTextMessage doing: " + move);
```

```
//RobotUtils.sendMsg(RobotController.robotName,move);
        } catch (Exception e) {
            System.out.println("WebSocketHandler |
handleTextMessage ERROR:"+e.getMessage());
        }

    }
    @Override
    protected void handleBinaryMessage(WebSocketSession
session, BinaryMessage message) throws IOException {
        System.out.println("WebSocketHandler |
handleBinaryMessage Received " );
        //session.sendMessage(message);
        //Send to all the connected clients
                Iterator<WebSocketSession> iter =
sessions.iterator();
        while( iter.hasNext() ){
            iter.next().sendMessage(message);
        }
    }

    public void sendToAll(String message)  {
        try{
                        //CommUtils.outblue("WebSocketHandler |
sendToAll String: " + message);
                        //JSONObject jsm = new
JSONObject(message);
                        //IApplMessage mm = new
ApplMessage(message);
                        //String mstr    =
mm.msgContent();//.replace("'","");
            sendToAll( new TextMessage(message)) ;
        }catch( Exception e ){
            System.out.println("WebSocketHandler | sendToAll
String ERROR:"+e.getMessage());
```

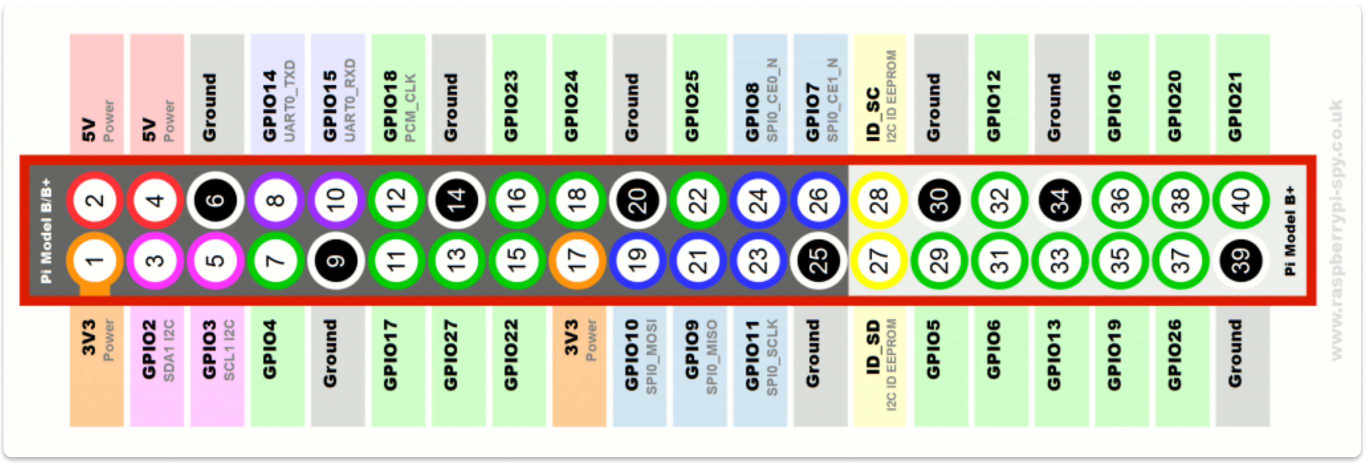
```
        }
    }
    public void sendToAll(TextMessage message) {
        //CommUtils.outblue("WebSocketHandler | sendToAll " +
message.getPayload() + " TextMessage sessions:" +
sessions.size());
        Iterator<WebSocketSession> iter = sessions.iterator();
        //CommUtils.outyellow("WebSocketHandler | sendToAll
TextMessage " );
        while( iter.hasNext() ){
            try{
                WebSocketSession session = iter.next();
                // + message.getPayload() + " for session " +
session.getRemoteAddress() );
                synchronized(session){
                //CommUtils.delay(5000);
                //CommUtils.outyellow("WebSocketHandler |
sendToAll session " );
                                session.sendMessage(message);
                }
            }catch(Exception e){
                System.out.println("WebSocketHandler |
TextMessage " + message + " ERROR:"+e.getMessage());
            }
        }
    }
}
```

Ticket Handler per contare quanti ticket vengono rifiutati

```
QActor tickethandler context ctxcoldstoragearea {


        [#
                ...
                var Rejected = 0
        #]

        State s0 initial{
                ...
                updateResource [# "" + Rejected #]
        } Goto work



        State work {
        }Transition t0  whenRequest depositRequest ->
checkforweight
                                        whenRequest
checkmyticket -> checktheticket
                                        whenRequest
getrejectedtickets -> sendrejectedticketnumber

        State checkforweight {
                ...
        }Transition t1 whenReply weightKO -> checkdeadlines
                                        whenReply weightOK ->
returnticket

        State checkdeadlines{
                ...
        } Goto returnticket if [# Accepted #] else reject

        State reject {


                [# Rejected++ #]
                updateResource [# "" + Rejected #]
                println("tickethandler - non c'è comunque
posto, vai a casa") color blue
                replyTo depositRequest with reject : reject(
reject )
        } Goto work

        State returnticket {
                ...
        } Goto work

        State checktheticket {
                ...
        } Goto work

        State sendrejectedticketnumber{
                onMsg(getrejectedtickets :
getrejectedtickets(NO_PARAM)){
                        replyTo getrejectedtickets with
rejectedtickets : rejectedtickets($Rejected)
                }
        }Goto work
}
```

Deployment

Deployment on RaspberryPi 3B/3B+



Led

- braccino corto: pin fisico 39 (GND)
- braccino lungo: pin fisico 40 (GPIO21)

Sonar

- VCC : pin fisico 4 (+5v)
- GND : pin fisico 6 (GND)
- TRIG: pin fisico 11 (GPIO 17)
- ECHO: pin fisico 13 (GPIO 27)

1. Avvia main alarm

Main system deployment

1. Avviare il container itunibovirtualrobot23 su docker
   Viene lanciato l'ambiente virtuale con il robot all'indirizzo http://localhost:8090/
2. In intellij avviare il file MainCtxColdStorageArea.kt del progetto coldStorage
3. Avviare la parte web

Lica Uccini	Luca Lombardi	Giacomo Romanini
[github: LisaIU00]()	[github: Lombax99]()	[github: RedDuality]()