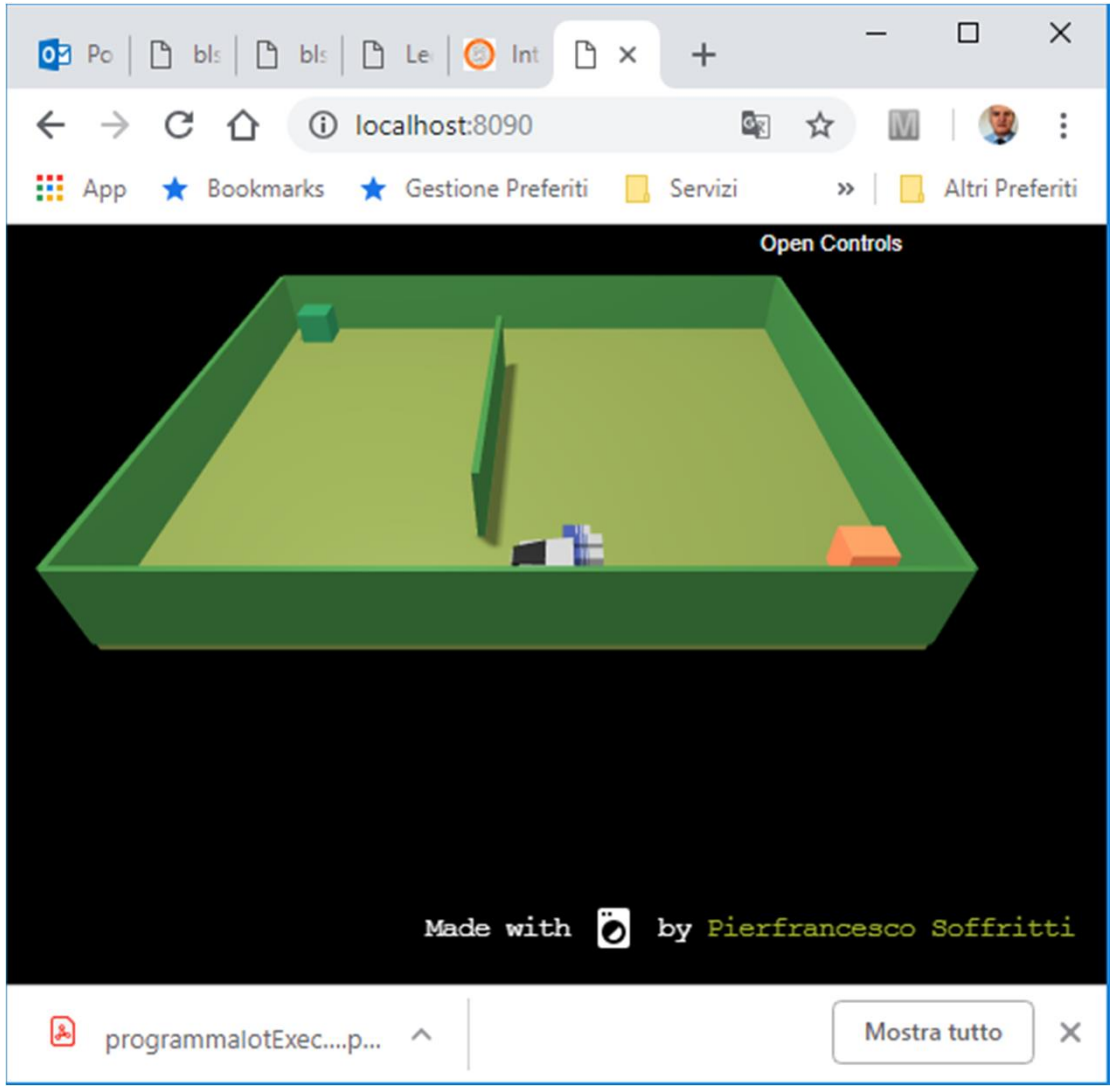
Unibo DDR robots


localhost:8090
App
Bookmarks
Gestione Preferiti
Servizi
Altri Preferiti
Open Controls
Made with by Pierfrancesco Soffritti
programmaIotExec....p...
Mostra tutto

Linguaggi per la interazione con il robot virtuale

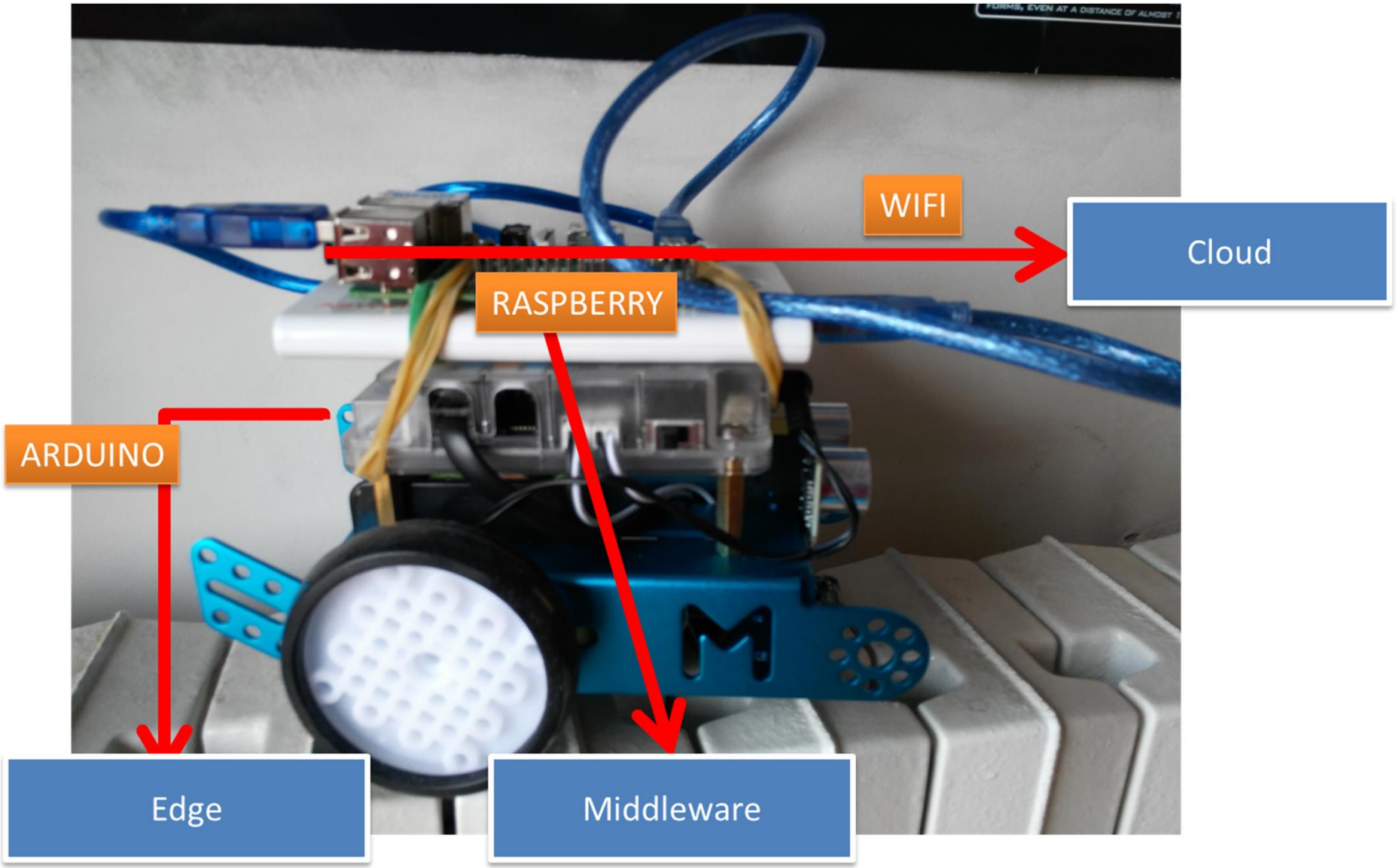
commands

```
- moveForward -    `{ "type": "moveForward", "arg": 300 }`
- moveBackward -   `{ "type": "moveBackward", "arg": 300 }`

- turnRight -      `{ "type": "turnRight", "arg": 300 }`
- turnLeft -       `{ "type": "turnLeft", "arg": 300 }`

- alarm -          `{ "type": "alarm" }`
```

info

```
webpage-ready - `{ "type": "webpage-ready, "arg": {} }`
sonar-activated - `{
    "type:" "sonar-activated",
    "arg": { "sonarName": "sonarName", "distance": 1, "axis": "x" }
}`
collision - `{
   "type": "collision",
   "arg": { "objectName": "obstacle-1" }
}`
```

it.unibo.mbot.virtual.clientTcp.java


WIFI
Cloud
RASPBERRY
ARDUINO
Edge
Middleware

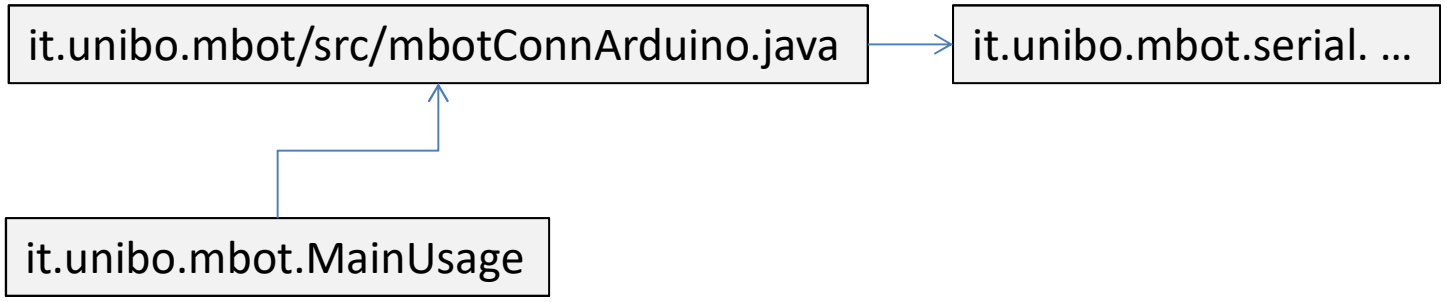
Linguaggi per la interazione con il robot fisico

commands

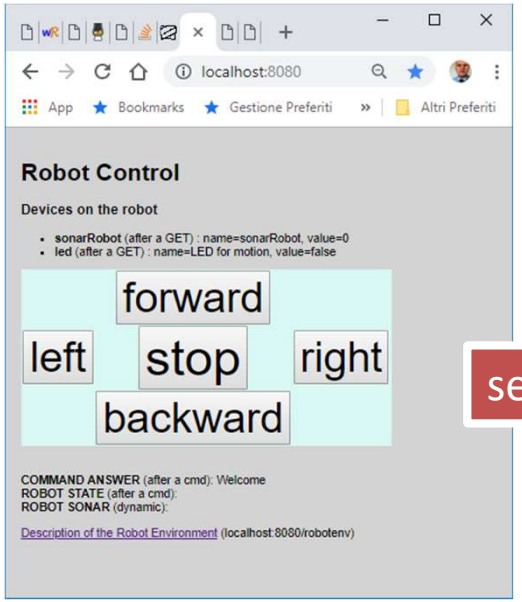
CMD = w | s | h | a | d

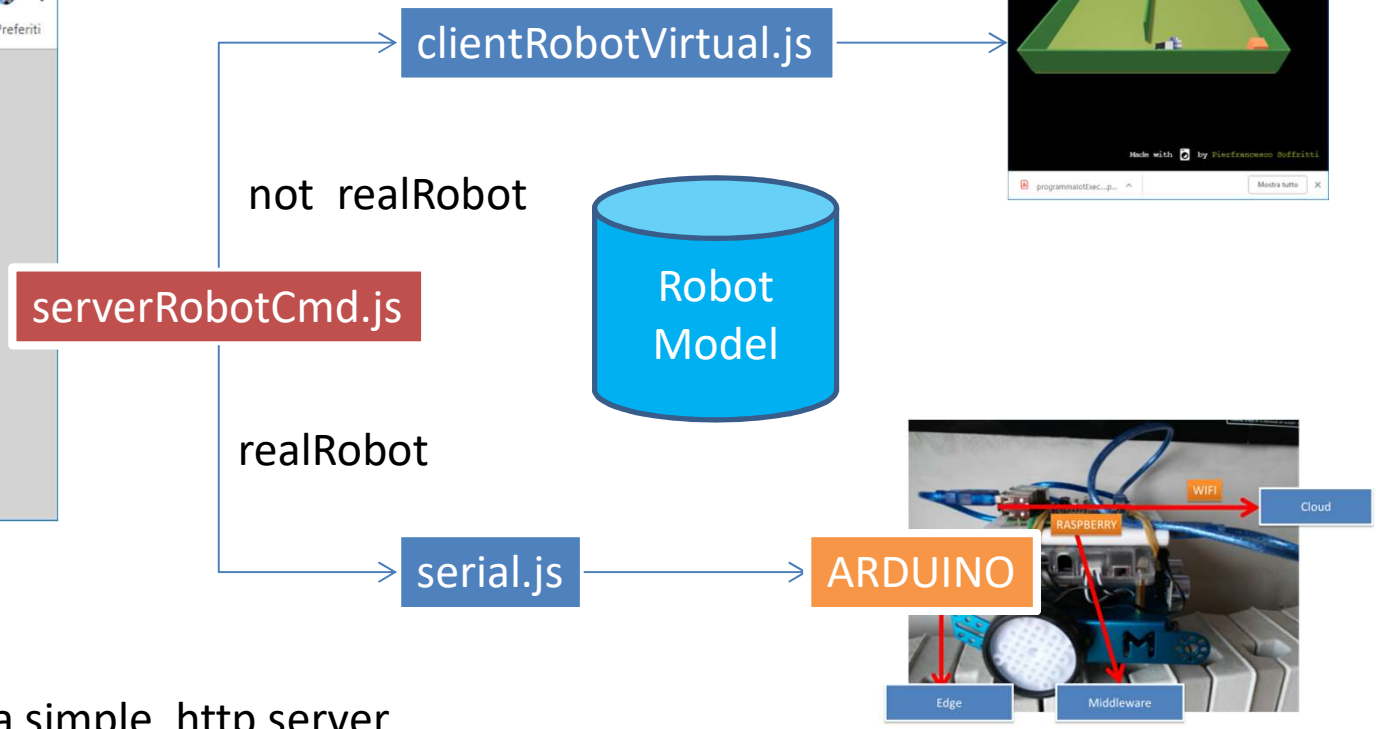
info

A double



Un server per la interazione remota





A solution based on a simple http server
with a model, views and controllers
and a limited usage of Express

Note di costruzione del serverRobotCmd.js

1. Definizione del modello del robot (e del suo ambiente)
   - **url, robot, robotEnv, meta ajaxAccess.html**

```
{"url":"http://localhost:8080/model/",
"robot": {
   "name": "UniboDdrRobot",
   "description": "A simple robot model",
   "properties":{
            "link": "/robotstate",
            "resources":{ "state": "stopped“ }
   },
   "devices":{  "link": "/robotdevices",
                  "resources":{ … }
      },
    "actions":{  …     }
   },
 "robotenv": {
   "link": "/robotenv",
   "name": "RobotEnv",
   "description": "The robot environment.",
   "devices":{
      "link": "/robotenv/devices",
     “ resources":{     …  }
 }
…
```

nodeCode\robot\models\robot.json

jquery

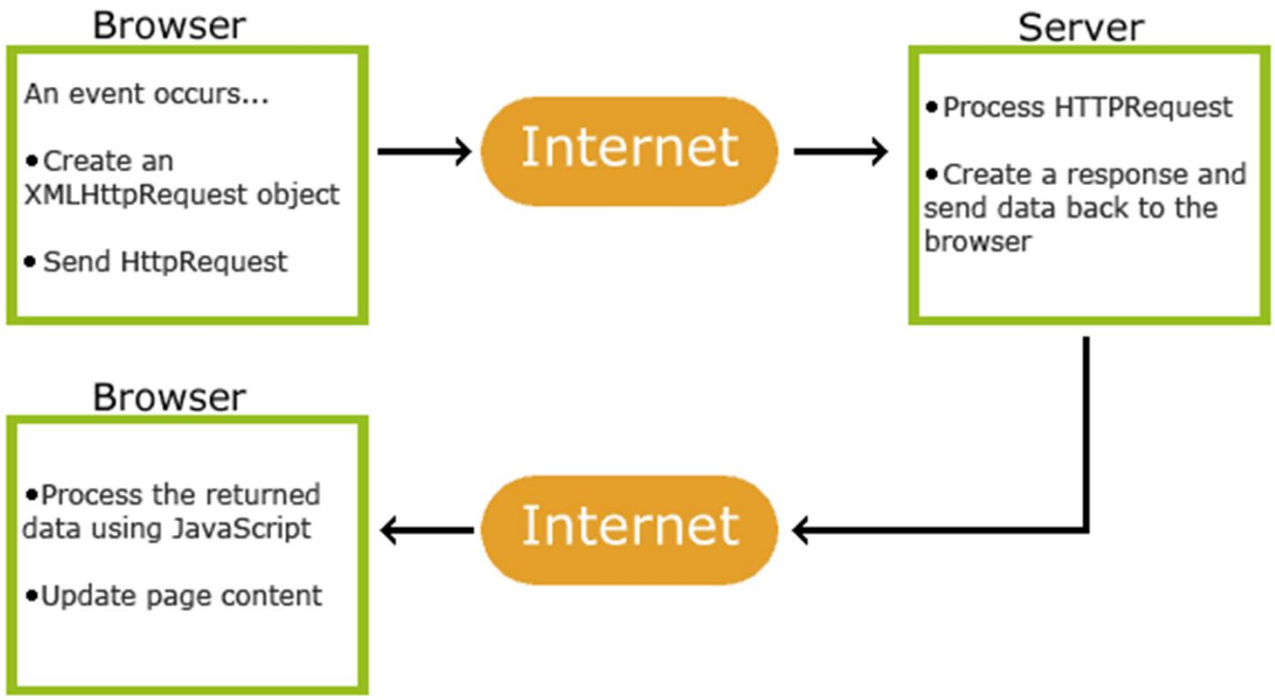
it.unibo.mbot2018/nodeCode/jQueryAjax

- jQuery is a JavaScript Library.
- jQuery greatly simplifies JavaScript programming.

ajax

- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

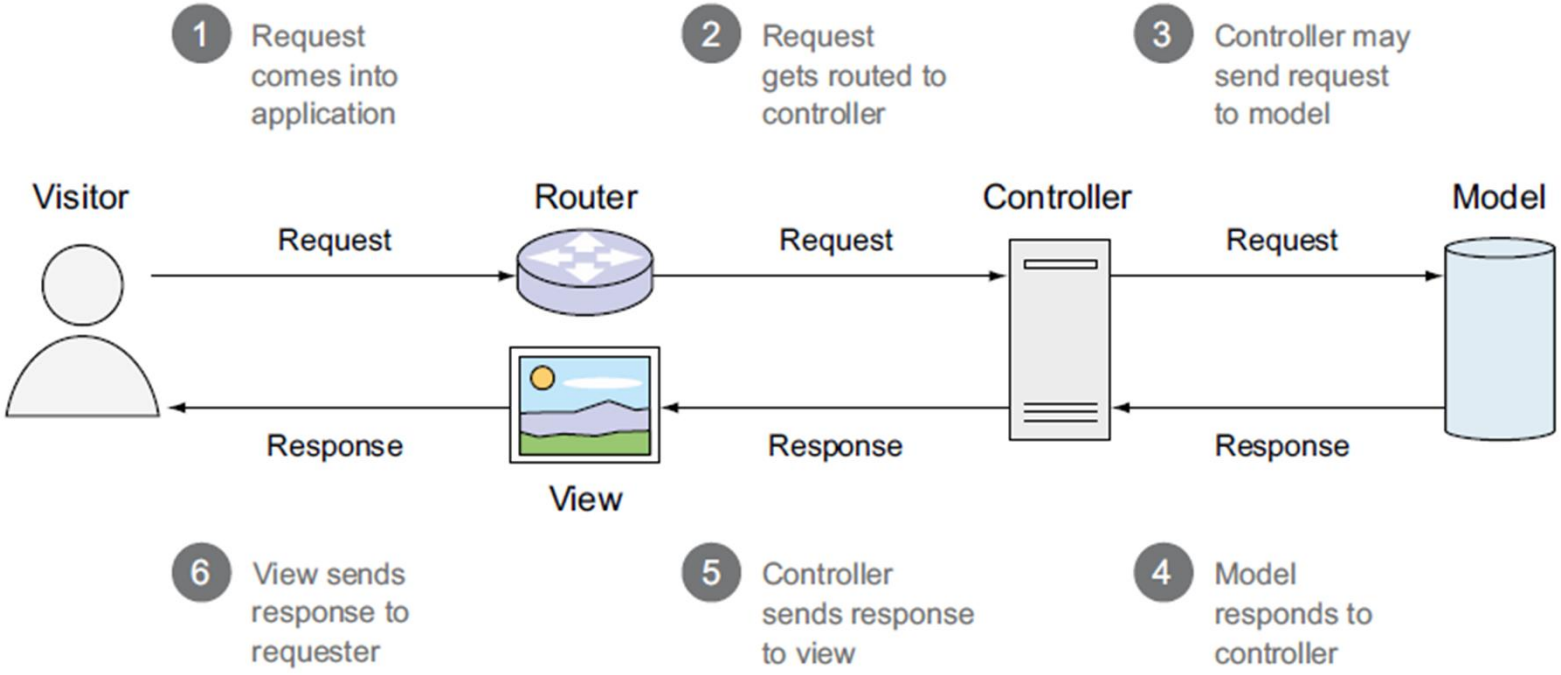
testJQueryAjax.html



MVC

In MVC web applications, the user will typically request a resource from the server, which will cause the *controller* to request application data from the *model* and then pass the data to the *view*, which will finally format the data for the end user.

The view is often implemented using one of various templating languages. When an application uses templating, the view will relay selected values, returned by the model, to a *template engine*, and specify what template file should define how to display the provided values.



https://expressjs.com/en/api.html

Express (basic)

app.js (Middleware)

```
var express = require("express");
var http = require("http");

var app = express();

app.use ( <ARGS> );

app.get ( <ARGS> )
app.put ( < ARGS> )

http.createServer(app).listen(3000);
```

starts a new Express application and returns a request handler function.

Run this on ALL requests.

routing

Run this on GET, PUT … requests.

```
( <ARGS> ) =
('/',  function( request, response ){..} )
or
( … ,  function( request, response, next ){…} )
```

```
response.send( … )
response.end( … )
```

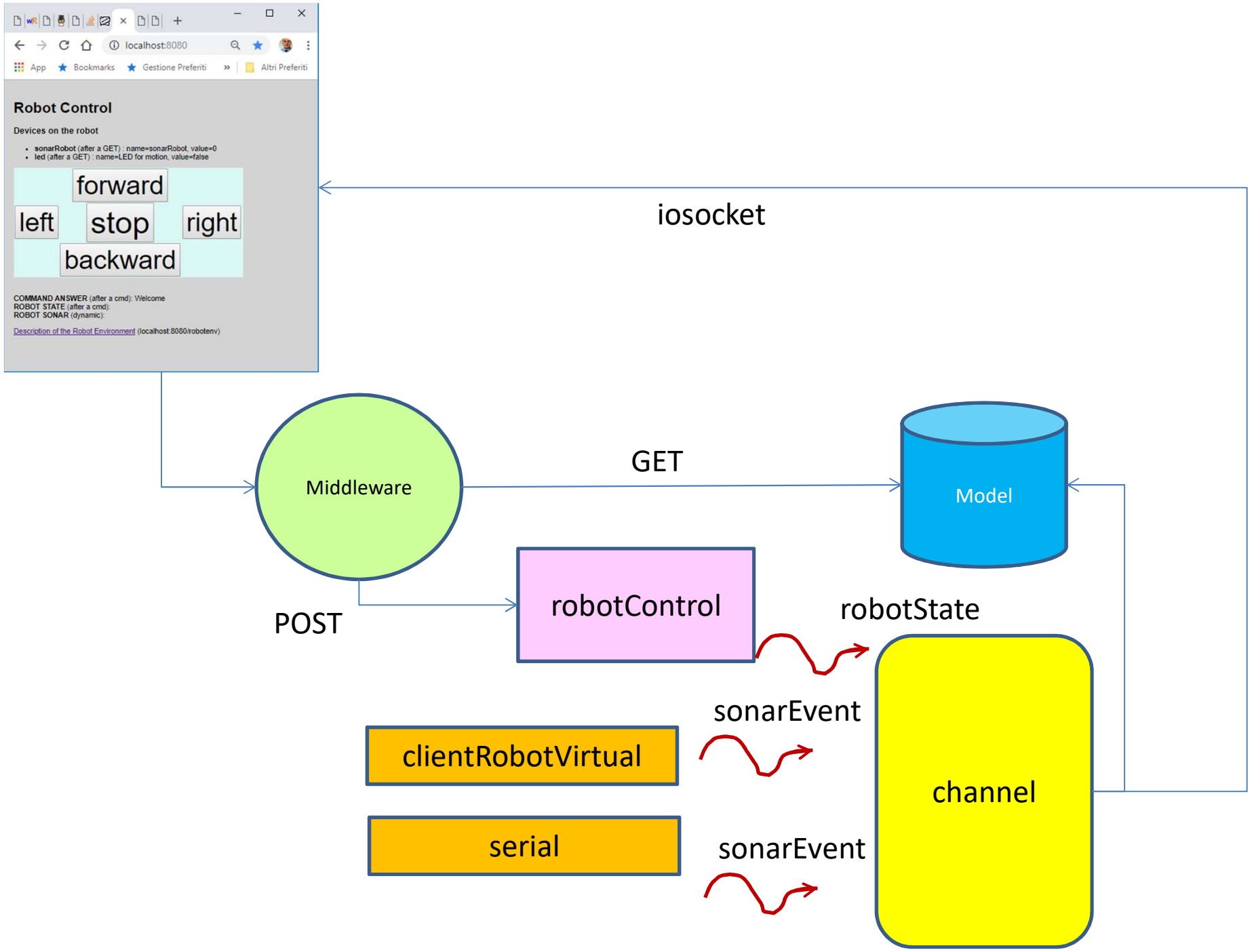
```
response.render( … )
```

views

https://expressjs.com/en/api.html#res

Note di costruzione del serverRobotCmd.js

1. Definizione del modello del robot (e del suo ambiente)
   - **robot, robotEnv, links/meta ajaxAccess.html**
2. Predisposizione del server
   - **http, express, iosocket**
3. Impostazione dei rendering engines
   - **access.ejs, robotenv.ejs**
4. Impostazione dei controllers
   - **robotControl**
5. Impostazione del middleware
   - **express.static, app.use, app.get, app.post, ..**
6. Creazione di un supporto per la notifica di eventi
   - **channel**
7. Creazione di supporti per la interazione con i robot fisici e/o virtuali
   - **clientRobotVirtual, serial**

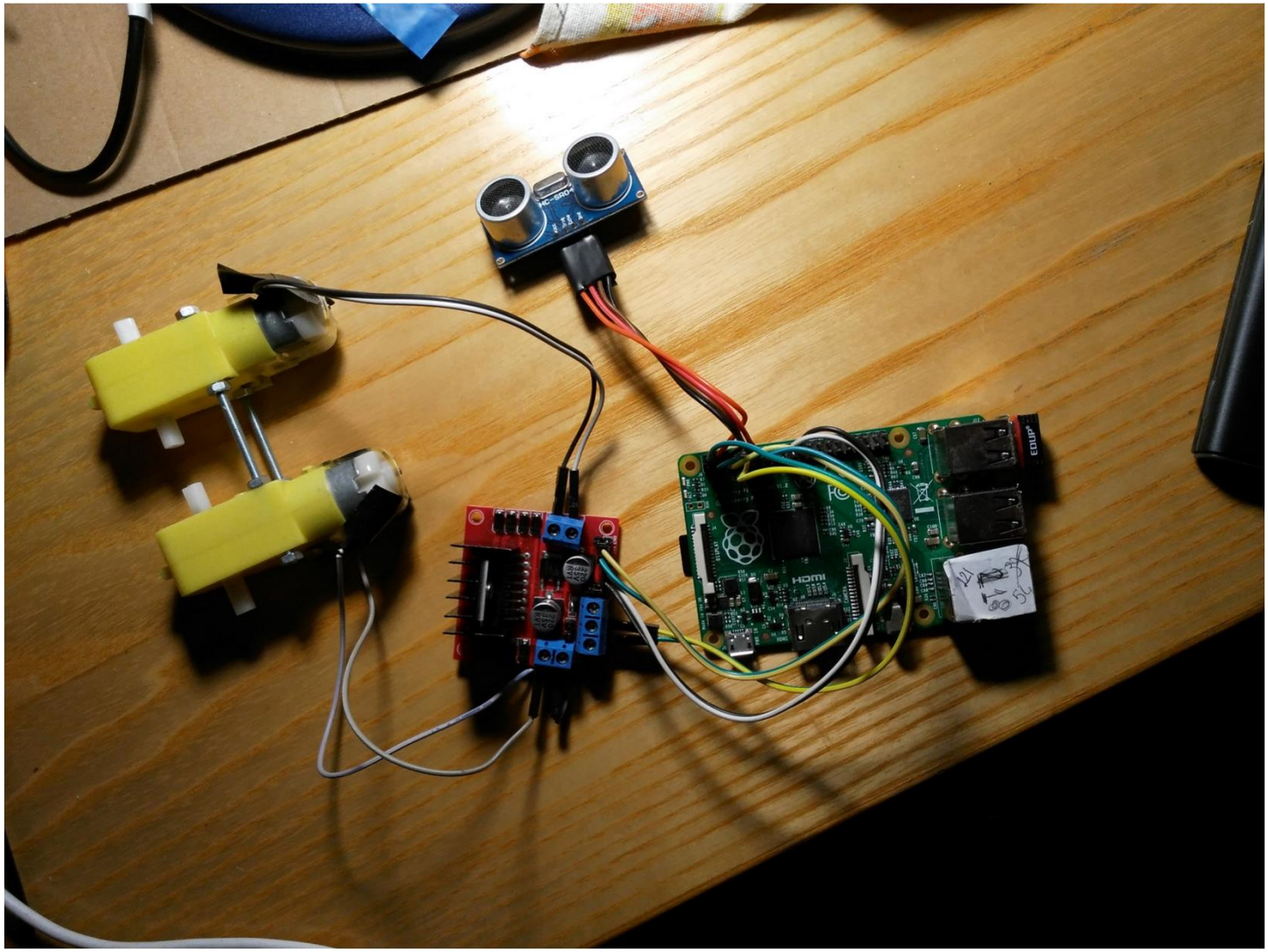

localhost:8080
App
Bookmarks
Gestione Preferiti
Altri Preferiti
Robot Control
Devices on the robot
sonarRobot (after a GET) : name=sonarRobot, value=0
led (after a GET) : name=LED for motion, value=false
forward
left
stop
right
backward
COMMAND ANSWER (after a cmd): Welcome
ROBOT STATE (after a cmd):
ROBOT SONAR (dynamic):
Description of the Robot Environment (localhost:8080/robotenv)
iosocket
Middleware
GET
Model
POST
robotControl
robotState
sonarEvent
clientRobotVirtual
channel
serial
sonarEvent

Funzionalità

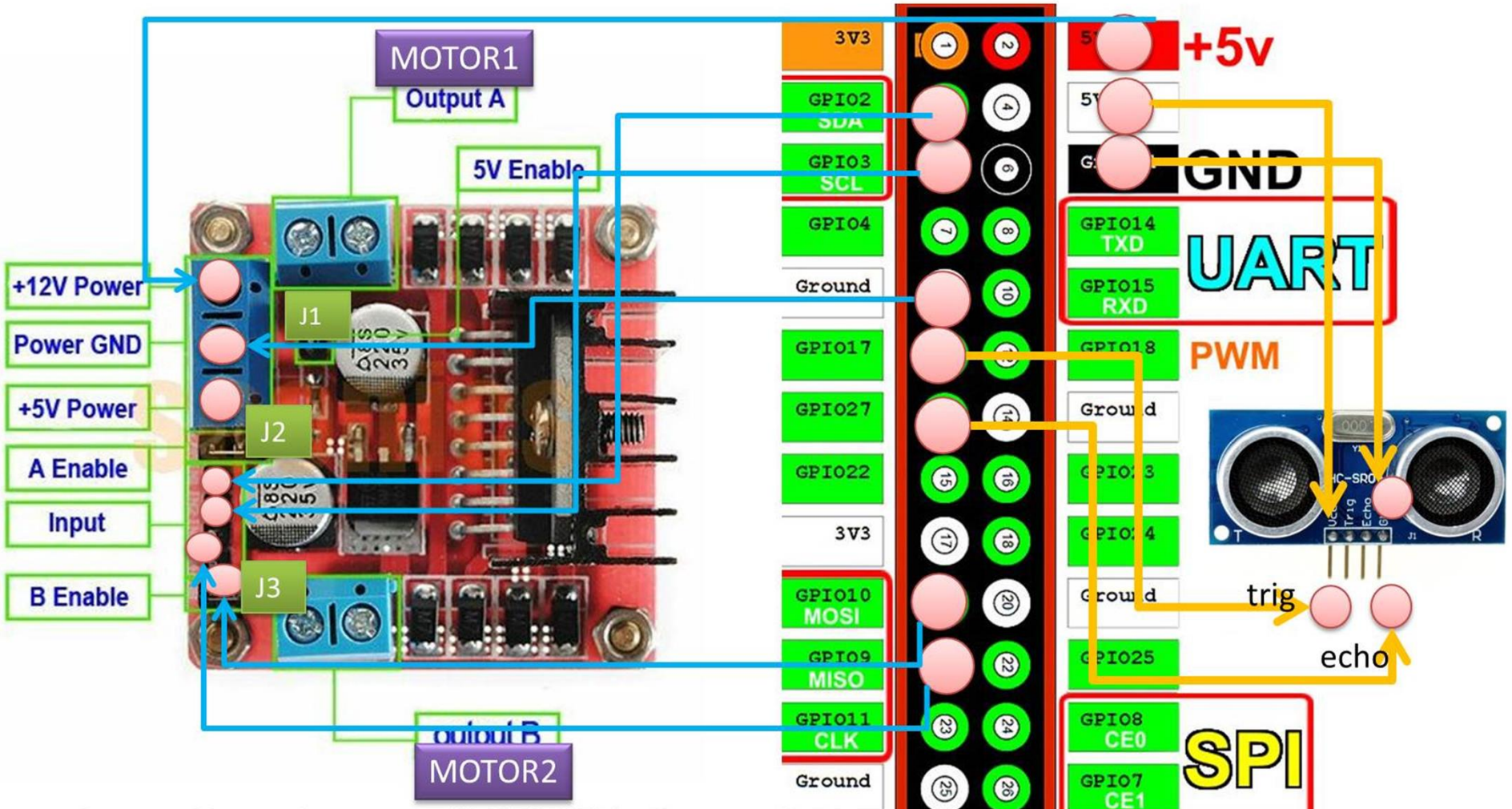
- Consente di inviare comandi a un robot
- Visualizza la risposta a un comando e lo stato del robot dopo la sua esecuzione
- Permette l'ispezione del modello del robot e dell'ambiente in cui il robot opera
- Visualizza lo stato del sonar montato sul robot
- ….

ROBOT DDR RASPBERRY-BASED

Skeleton



Connections



Jamper J1 : va rimosso se 12 V (5-35) è alimentato
Jamper J2, J3 : se inseriti si va al max speed
se rimossi il pin esterno va connesso a un GPIO PWM

it.unibo.mbot2018/raspberry

nanoMotorDriveA.sh

nanoMotorDriveB.sh

Motors.c

```
#define inp1m1 8
#define inp2m1 9

#define inp1m2 12
#define inp2m2 13

void setup() {
wiringPiSetup();
pinMode(inp1m1, OUTPUT);
pinMode(inp2m1, OUTPUT);
pinMode(inp1m2, OUTPUT);
pinMode(inp2m2, OUTPUT);
}
```

SonarAlone.c

it.unibo.mbot2018

configuration/nano/iotRobot.properties

hardwareConfiguration.properties

```
motor.left=gpio.motor
motor.left.pin.cw=8
motor.left.pin.ccw=9
motor.left.private=false
#
motor.right=gpio.motor
motor.right.pin.cw=12
motor.right.pin.ccw=13
motor.right.private=false
# --------- SENSORS ---------
distance.front_top=hcsr04
distance.front_top.trig=0
distance.front_top.echo=2
distance.front_top.private=false
# ---------COMPOSED COMPONENT ---------
actuators.bottom=ddmotorbased
actuators.bottom.name=motors
actuators.bottom.comp=motor.left,motor.right
actuators.bottom.private=true
# ---------MAIN ROBOT ---------
baserobot.bottom=differentialdrive
baserobot.bottom.name=nano
baserobot.bottom.comp=actuators.bottom
baserobot.bottom.private=false
```

src/it.unibo.robotRaspOnly/basicRobotUsageNaive.java

BasicRobotUsageNaive .java

introductionUniboDisiRobots.pdf

Work to do (Nov 14)

- Dato un DDR-Unibo-robot, costruire un sistema software che :
  - Consente ad un utente umano di inviare comandi (w|s|a|d|h) al robot
  - Consente ad un utente umano di conoscere lo stato di moto del robot
  - Permette di attivare/deattivare una attvità di 'esplorazione' in cui il robot:
    - Si muove in modo autonomo in un territorio piano.
    - Evita eventuali ostacoli (fissi e/mobili).
    - Si ferma ad un comando di STOP inviato dall'utente.
    - Si ferma nel caso di un segnale di allarme.
    - Cerca di creare una mappa del territorio in cui si muove.

Resources

- C:\Didattica\git\lab2014Bo\it.unibo.lab.baseRobot (robotSAM)
  - **labbaseRobotSam.jar**
- C:/Didattica/git/lab2014Bo/it.unibo.qactors/report/qactors/qactors2017e.pdf (pg. 77)

- it.unibo.robotRaspOnly.BasicRobotUsageNaive
  - Executes the commands **w|a!s|d|h** sent from the input console
  - On *robotSkeleton*: gpio version: 2.46
  - On *robotNano1*: gpio version: 2.29