UMEÅ UNIVERSITET

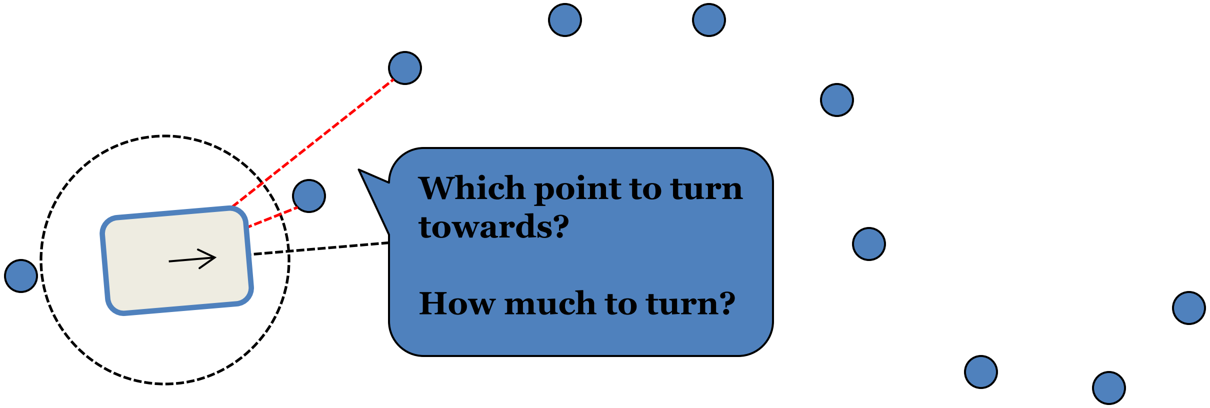
Fundamentals of Artificial Intelligence

Introduction to Assignment 1

www.cs.umu.se/kurser/5DV121/HT13

Follow the Path

- Create a controller that can guide the robot along a given path
  - The path is given as a sequence of coordinates

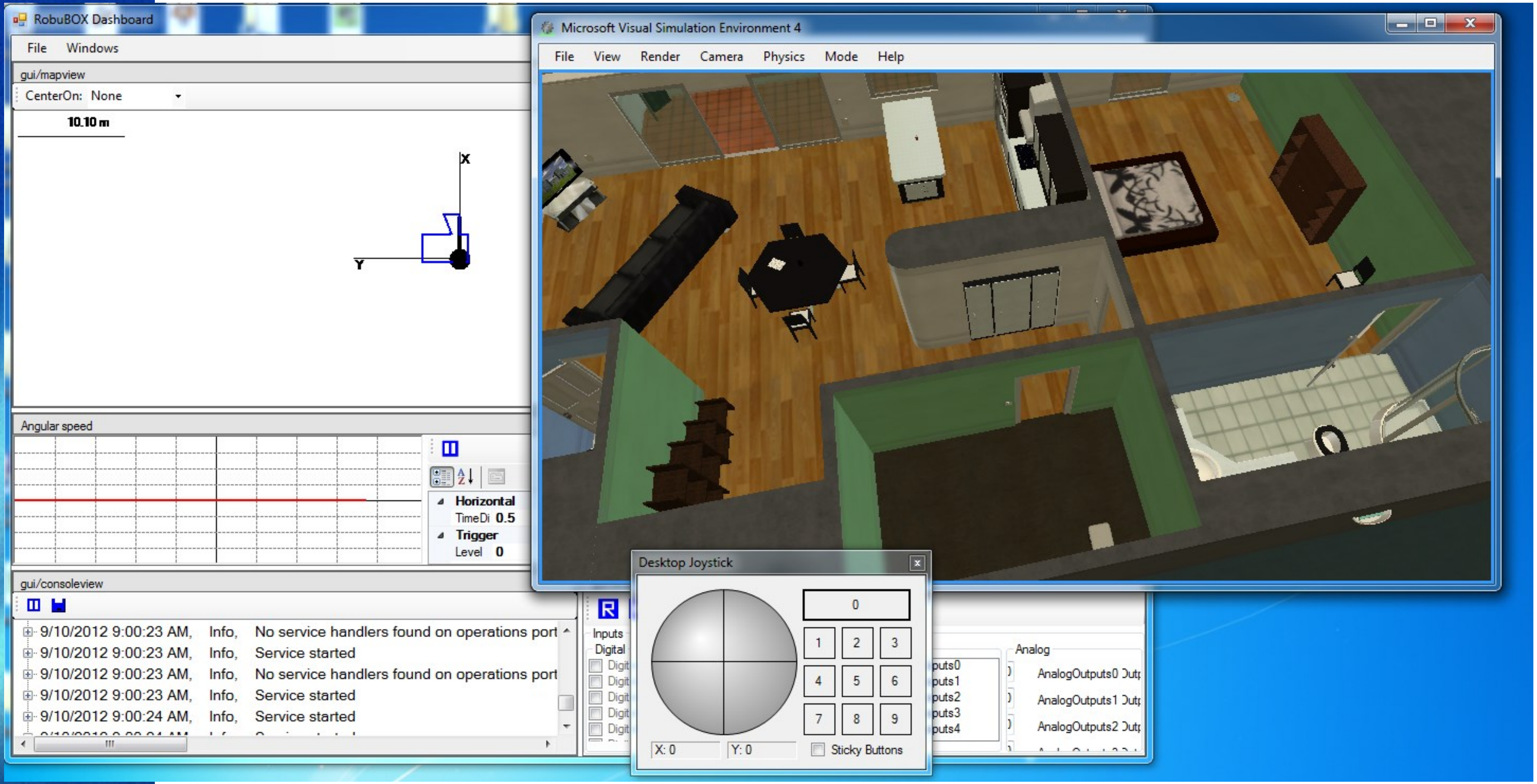


Follow the Path

- You will work with a simulated robot, in Microsoft Robotic Developer Studio 4 (MRDS)
- Create a program that can read sensors of the robot, and post commands to it
  - Communicate with the robot over HTTP

Microsoft Robotic Developer Studio 4

- Installed in all MA316 and MA326 computers
- Run it by executing **StartLokarria.bat**, located in **c:/MRDS4/store/launchers**



CenterOn: None

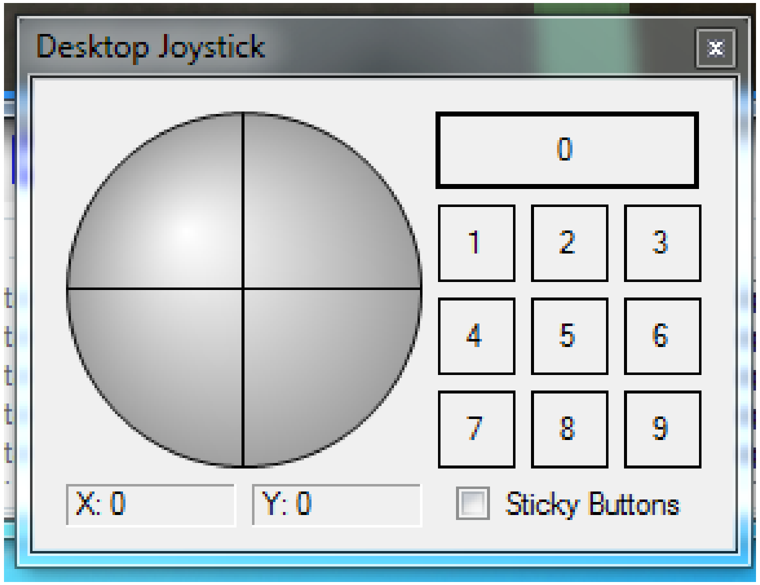
Horizontal
TimeDi 0.5
Trigger
Level 0

```
9/10/2012 9:00:23 AM,   Info,   No service handlers found on operations port
9/10/2012 9:00:23 AM,   Info,   Service started
9/10/2012 9:00:23 AM,   Info,   No service handlers found on operations port
9/10/2012 9:00:23 AM,   Info,   Service started
9/10/2012 9:00:24 AM,   Info,   Service started
```

X: 0   Y: 0

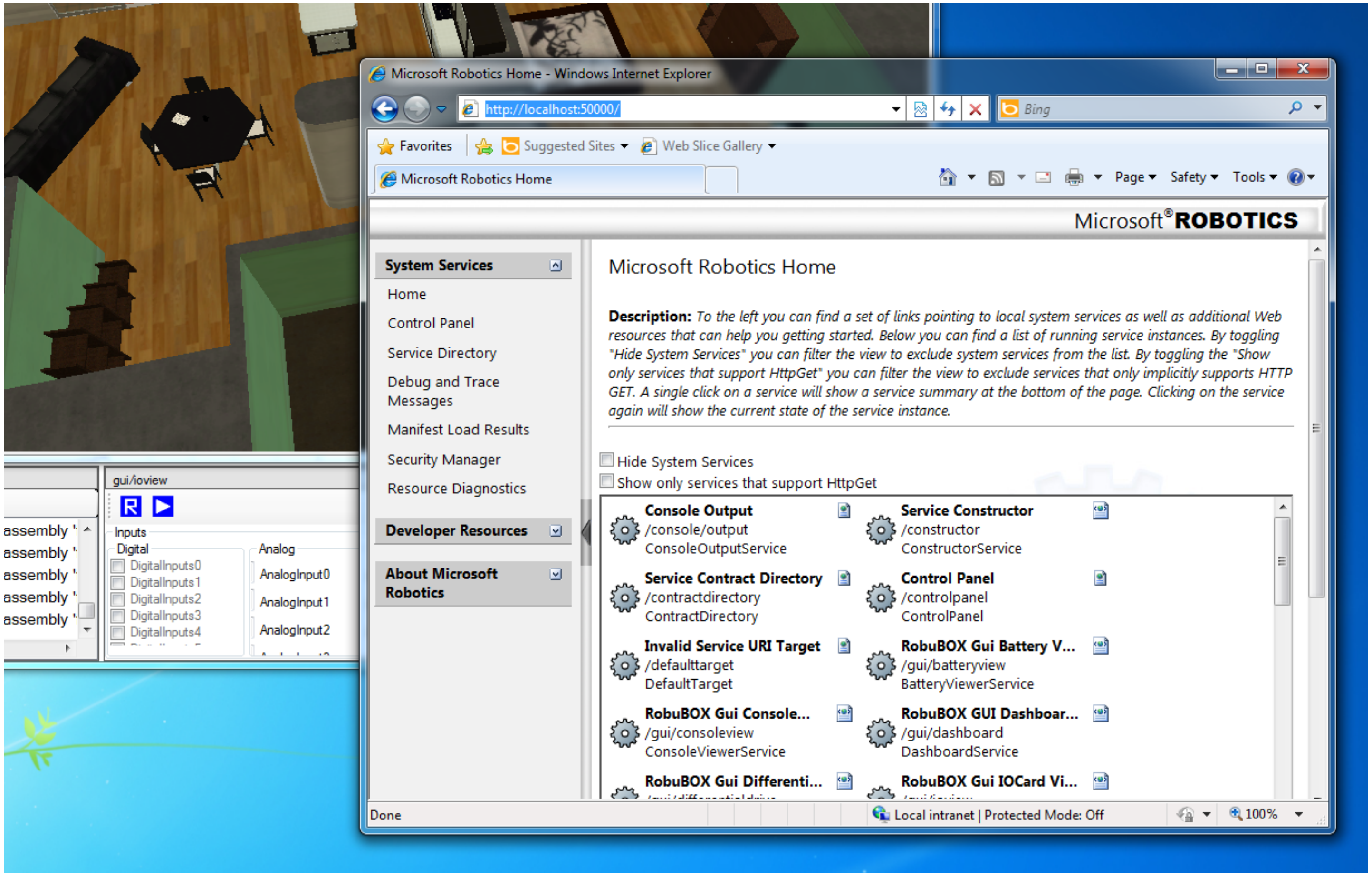
Controlling the robot

- Press Sticky Buttons
- Press 0
- Click and drag the joystick wheel



Connecting to MRDS

- Go to http://localhost:50000/ with your web browser



Microsoft Robotics Home

Description: *To the left you can find a set of links pointing to local system services as well as additional Web resources that can help you getting started. Below you can find a list of running service instances. By toggling "Hide System Services" you can filter the view to exclude system services from the list. By toggling the "Show only services that support HttpGet" you can filter the view to exclude services that only implicitly supports HTTP GET. A single click on a service will show a service summary at the bottom of the page. Clicking on the service again will show the current state of the service instance.*

☐ Hide System Services
☐ Show only services that support HttpGet

Console Output
/console/output
ConsoleOutputService

Service Constructor
/constructor
ConstructorService

Service Contract Directory
/contractdirectory
ContractDirectory

Control Panel
/controlpanel
ControlPanel

Invalid Service URI Target
/defaulttarget
DefaultTarget

RobuBOX Gui Battery V...
/gui/batteryview
BatteryViewerService

RobuBOX Gui Console...
/gui/consoleview
ConsoleViewerService

RobuBOX GUI Dashboar...
/gui/dashboard
DashboardService

RobuBOX Gui Differenti...

RobuBOX Gui IOCard Vi...

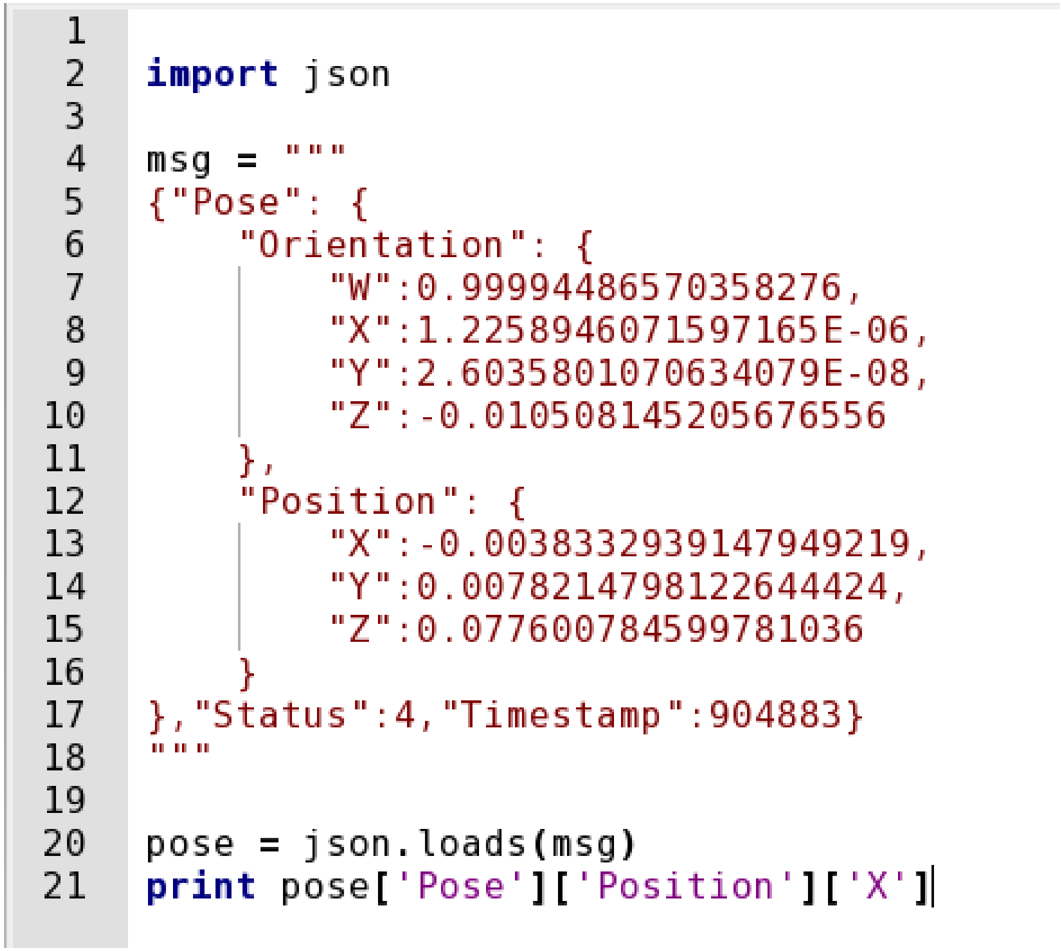
Reading the position of the robot

- http://localhost:50000/lokarria/localization
- Response is given as a JSON message:

```
{"Pose”: {
   "Orientation": {
      "W":0.99994486570358276,
      "X":1.2258946071597165E-06,
      "Y":2.6035801070634079E-08,
      "Z":-0.010508145205676556
   },
   "Position": {
      "X":-0.0038332939147949219,
      "Y":0.0078214798122644424,
      "Z":0.077600784599781036
   }
},"Status":4,"Timestamp":904883}
```

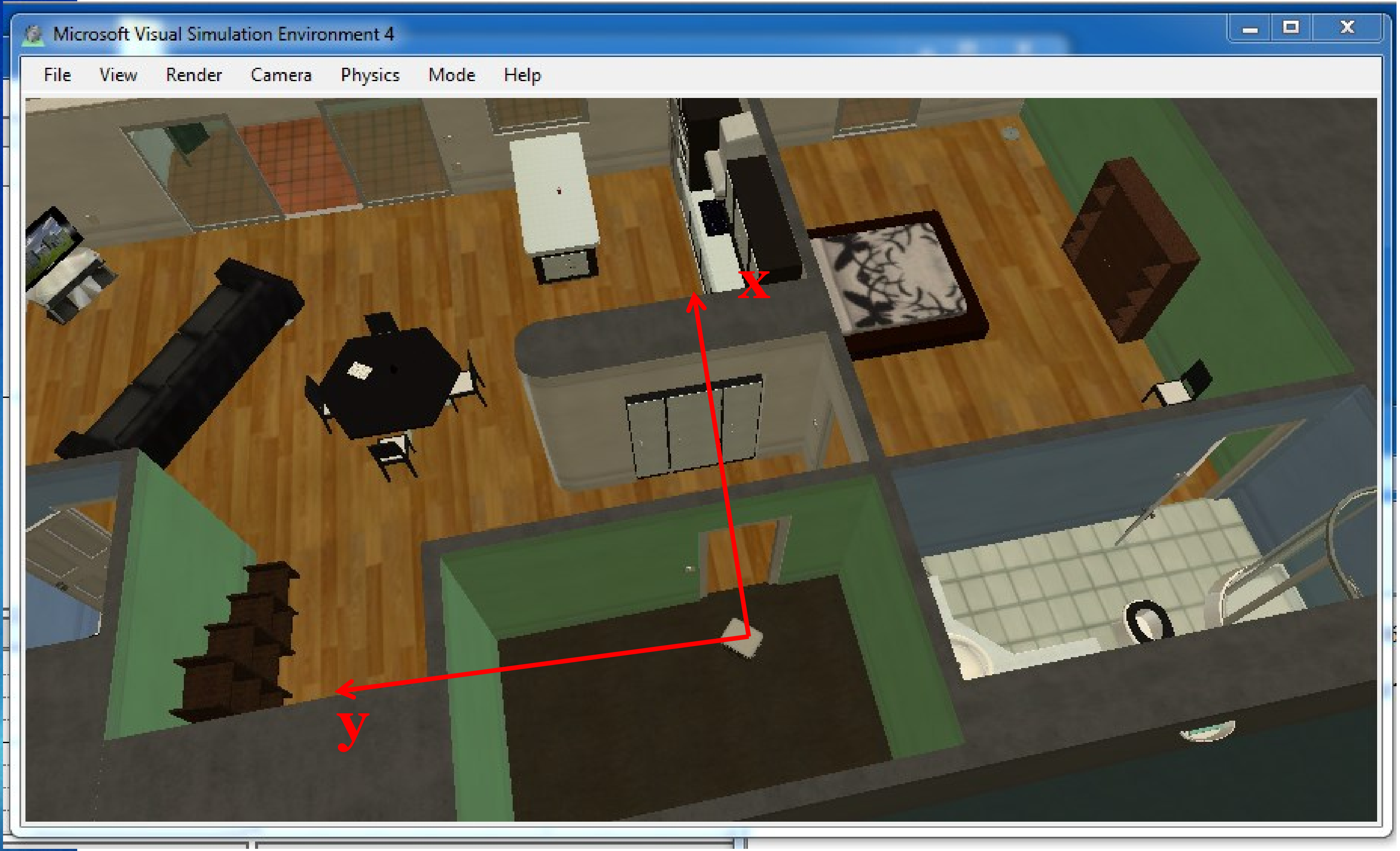
Parsing JSON

- Example in Python

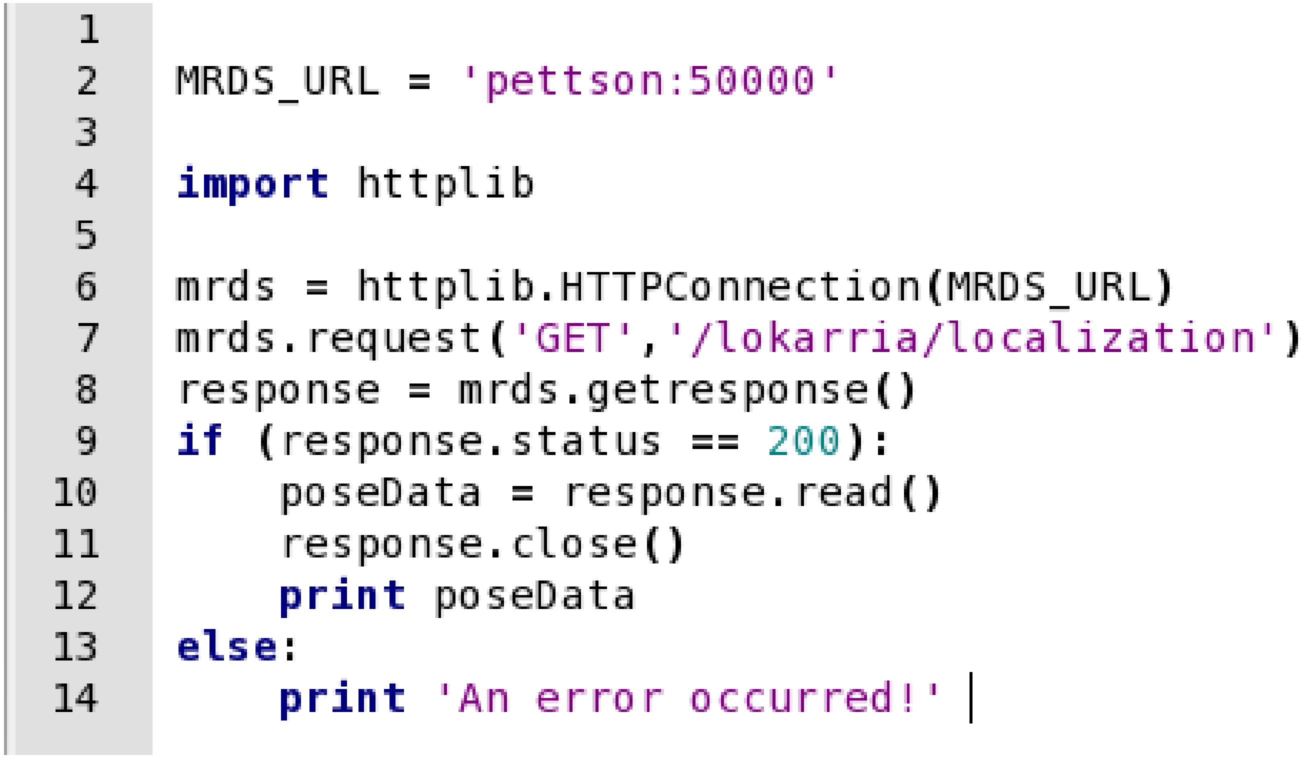


```python

import json

msg = """
{"Pose": {
    "Orientation": {
        "W":0.99994486570358276,
        "X":1.2258946071597165E-06,
        "Y":2.6035801070634079E-08,
        "Z":-0.010508145205676556
    },
    "Position": {
        "X":-0.0038332939147949219,
        "Y":0.0078214798122644424,
        "Z":0.077600784599781036
    }
},"Status":4,"Timestamp":904883}
"""

pose = json.loads(msg)
print pose['Pose']['Position']['X']
```

The position

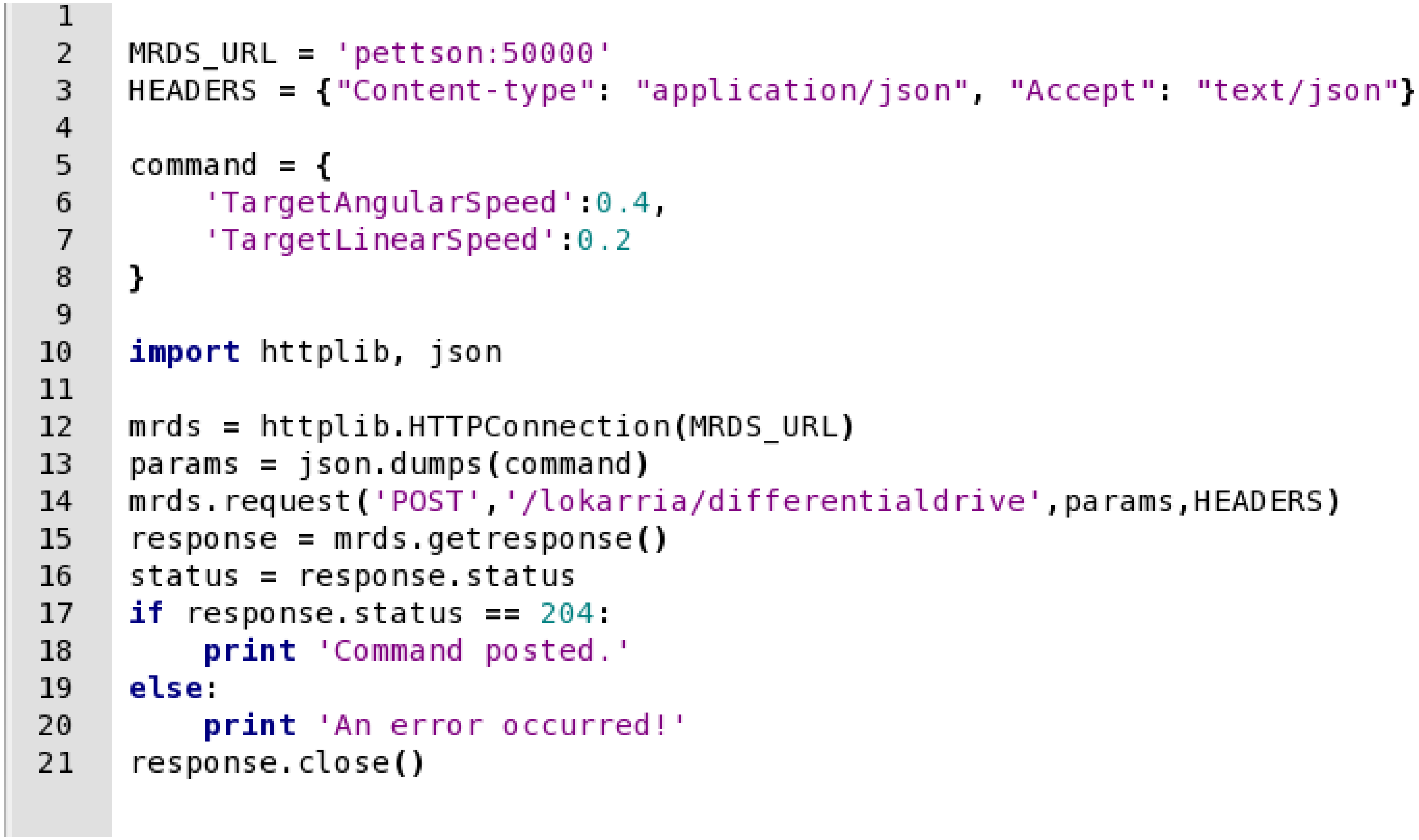


Requesting the Pose



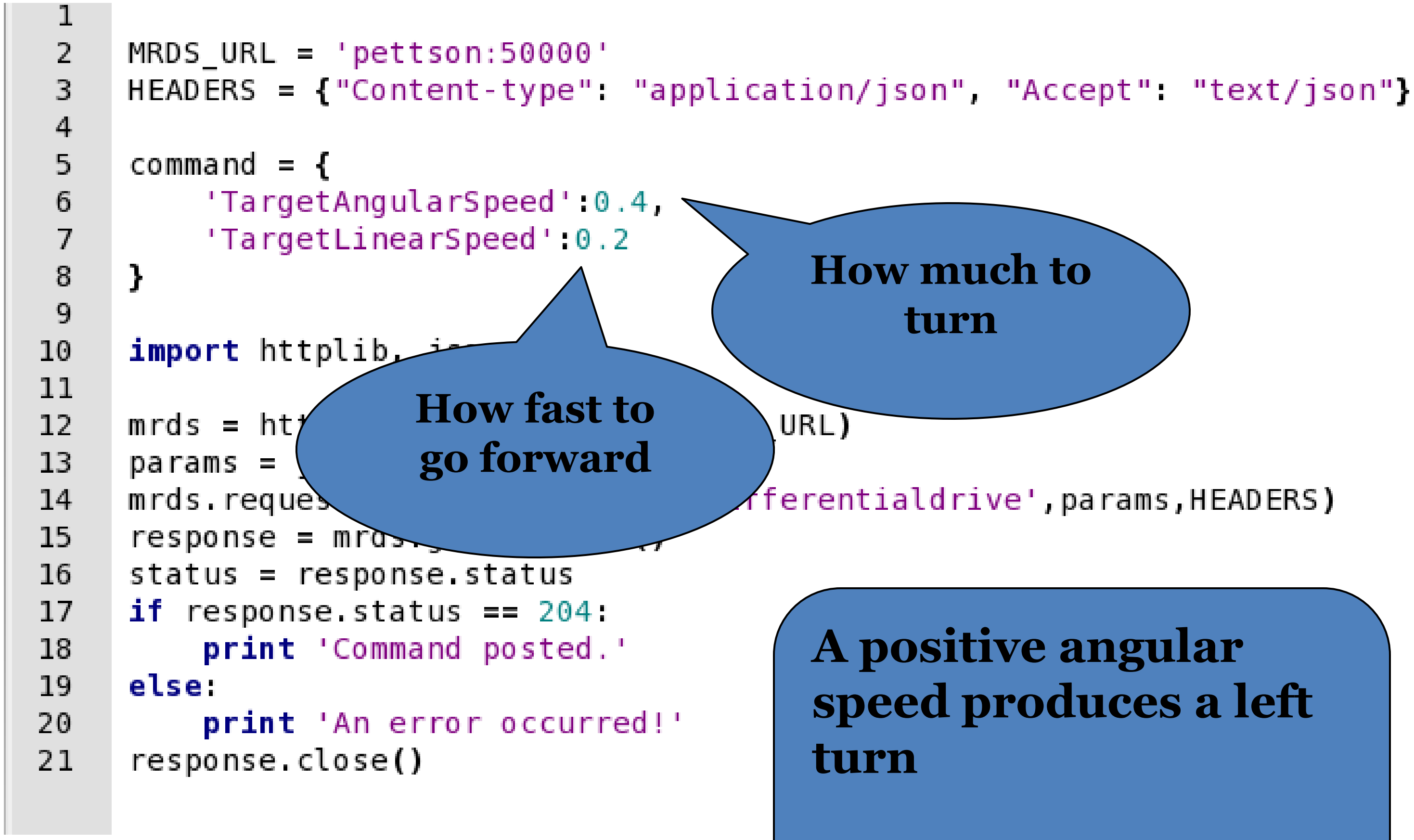
```

MRDS_URL = 'pettson:50000'

import httplib

mrds = httplib.HTTPConnection(MRDS_URL)
mrds.request('GET','/lokarria/localization')
response = mrds.getresponse()
if (response.status == 200):
    poseData = response.read()
    response.close()
    print poseData
else:
    print 'An error occurred!'
```

Controlling the robot



```

MRDS_URL = 'pettson:50000'
HEADERS = {"Content-type": "application/json", "Accept": "text/json"}

command = {
    'TargetAngularSpeed':0.4,
    'TargetLinearSpeed':0.2
}

import httplib, json

mrds = httplib.HTTPConnection(MRDS_URL)
params = json.dumps(command)
mrds.request('POST','/lokarria/differentialdrive',params,HEADERS)
response = mrds.getresponse()
status = response.status
if response.status == 204:
    print 'Command posted.'
else:
    print 'An error occurred!'
response.close()
```

Controlling the robot



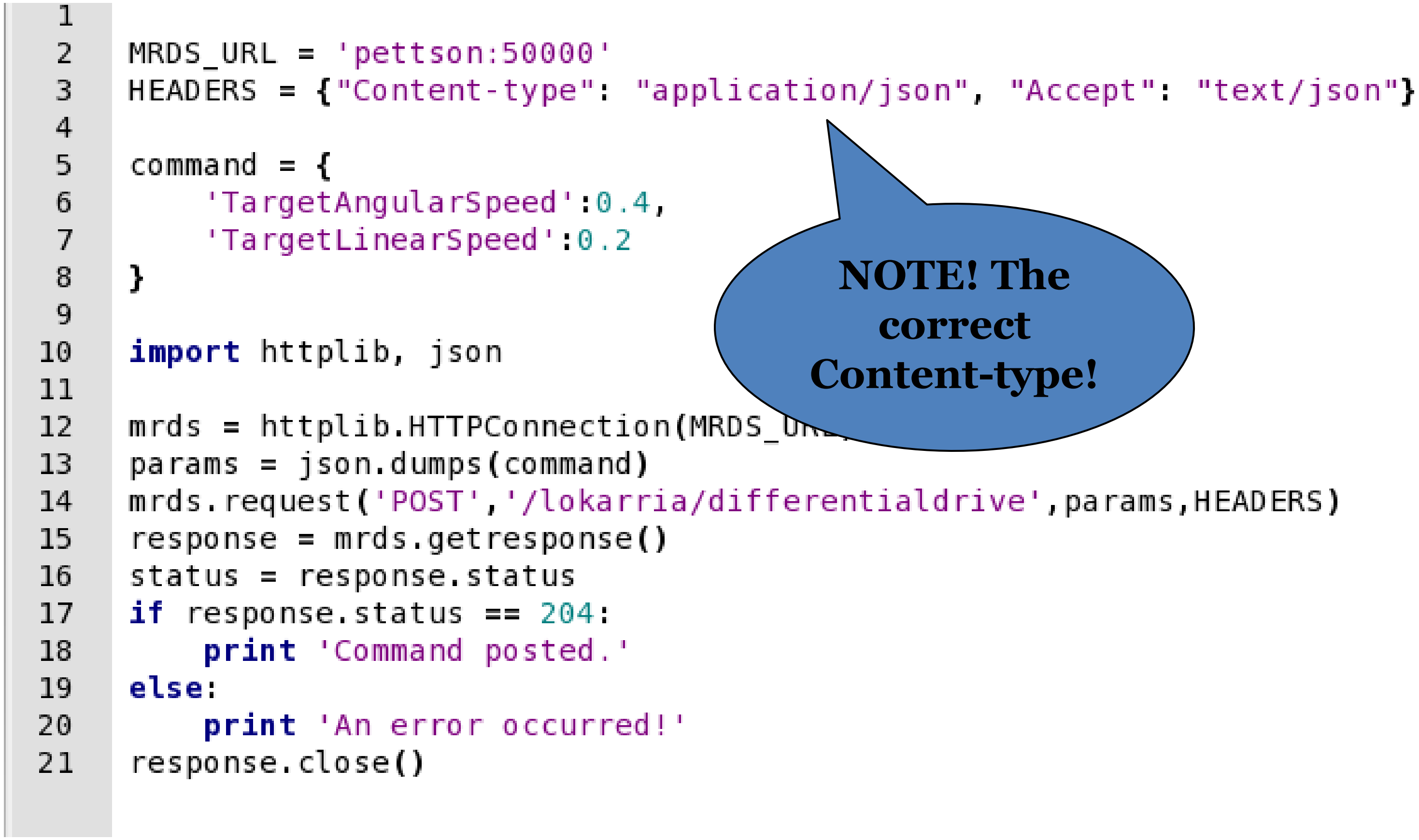
```
MRDS_URL = 'pettson:50000'
HEADERS = {"Content-type": "application/json", "Accept": "text/json"}

command = {
    'TargetAngularSpeed':0.4,
    'TargetLinearSpeed':0.2
}

import httplib, js

mrds = ht                                URL)
params = j
mrds.reques                             fferentialdrive',params,HEADERS)
response = mrds.                    ()
status = response.status
if response.status == 204:
    print 'Command posted.'
else:
    print 'An error occurred!'
response.close()
```

How much to turn

How fast to go forward

A positive angular speed produces a left turn

Controlling the robot



```

MRDS_URL = 'pettson:50000'
HEADERS = {"Content-type": "application/json", "Accept": "text/json"}

command = {
    'TargetAngularSpeed':0.4,
    'TargetLinearSpeed':0.2
}

import httplib, json

mrds = httplib.HTTPConnection(MRDS_URL)
params = json.dumps(command)
mrds.request('POST','/lokarria/differentialdrive',params,HEADERS)
response = mrds.getresponse()
status = response.status
if response.status == 204:
    print 'Command posted.'
else:
    print 'An error occurred!'
response.close()
```

NOTE! The correct Content-type!

Reading the Laser scanner

- http://pettson:50000/lokarria/laser/echoes
  - Returns 271 laser echoes (distances in m)
  - A JSON string