NAO control w/ Raspberry Pi

https://github.com/realgabriele/NAO_with_RPi

Gabriele Tagliente

Intelligent Systems and Robotics Laboratory
13/11/2020

Introduction

state of the art of the components I’ve worked on

NAO

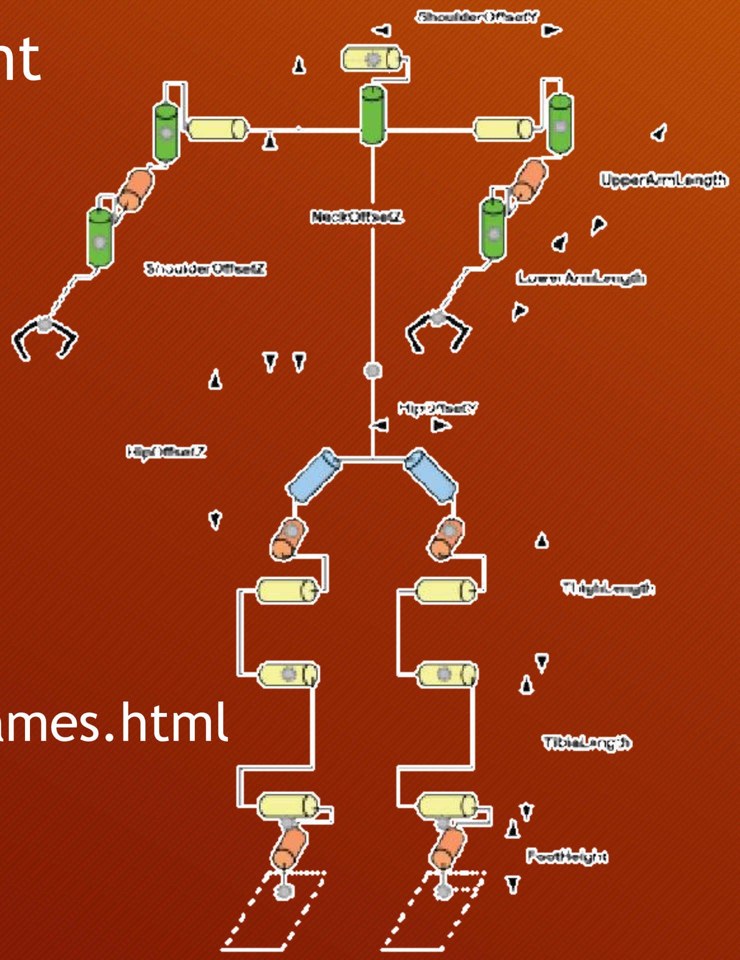
- NAO is a humanoid robot developed by Aldebaran Robotics
- 25 Degrees of Freedom (DoF)

- http://doc.aldebaran.com/2-1/home_nao.html

25 DoF

All the joints have sensors that keep track of them:

1. Position of the Actuator (rad) - programmed movement
2. Position from Sensor (actual value)
3. Temperature
4. Electric Current
5. Stiffness of the joint



http://doc.aldebaran.com/2-1/family/nao_dcm/actuator_sensor_names.html

Specs

- interaction: 2 speakers, 4 microphones, 2 HD cameras, LEDs, all joints control
- sensors: FSRs (feet), 3-axis gyroscope & accelerometer, 2 sonars, joint position, contact & tactile sensors
- Effectors and Chains

- OS: NAOqi (Linux-based)
- connectivity: Wi-Fi + Ethernet

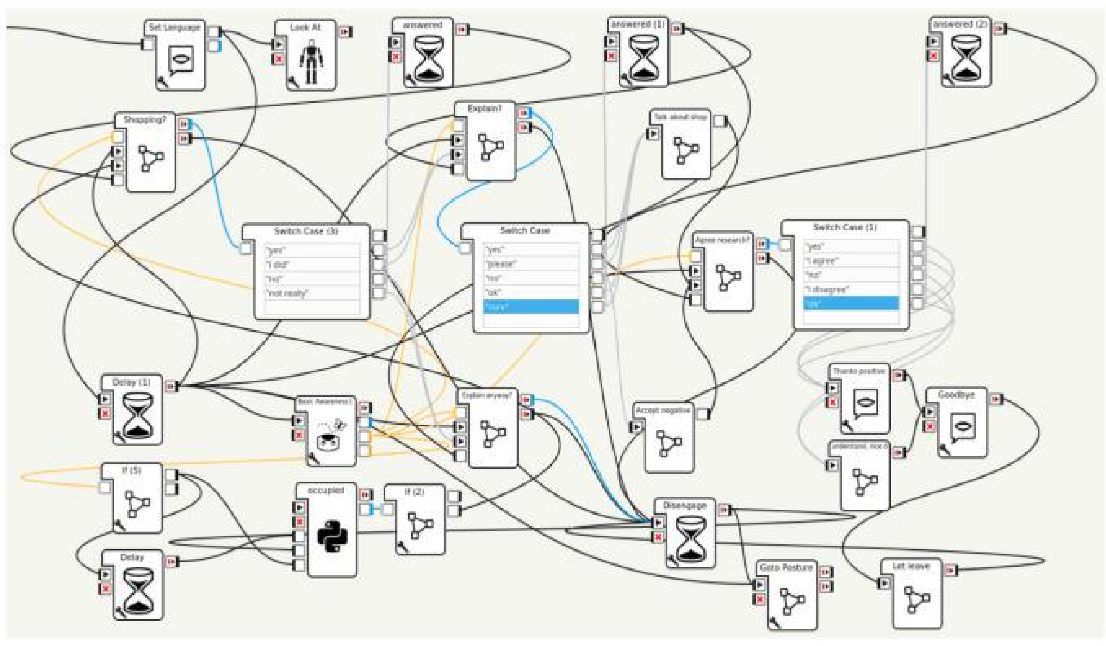
What we can do

SOFTWARE:

- Choregraphe controlled
  - programmable via a graphical visual language

- NAOqi SDK
  - libraries for Python and C++
  - PyNAOqi library

Downsides



Choregraphe:

- Spaghetti code via the interface graphical language
- Difficult (if not impossible) to implement custom high level or CPU intensive projects
  - everything is run on board

SDK:

- Limited resources
  - proprietary Linux distro
    - limited set of programs; difficult/slow to upgrade
  - only python2, not all libraries

Goal

1. Increase modularity

- all different modules are executed on different machines:
  - joint controller on NAO
  - intelligent controller on external computer

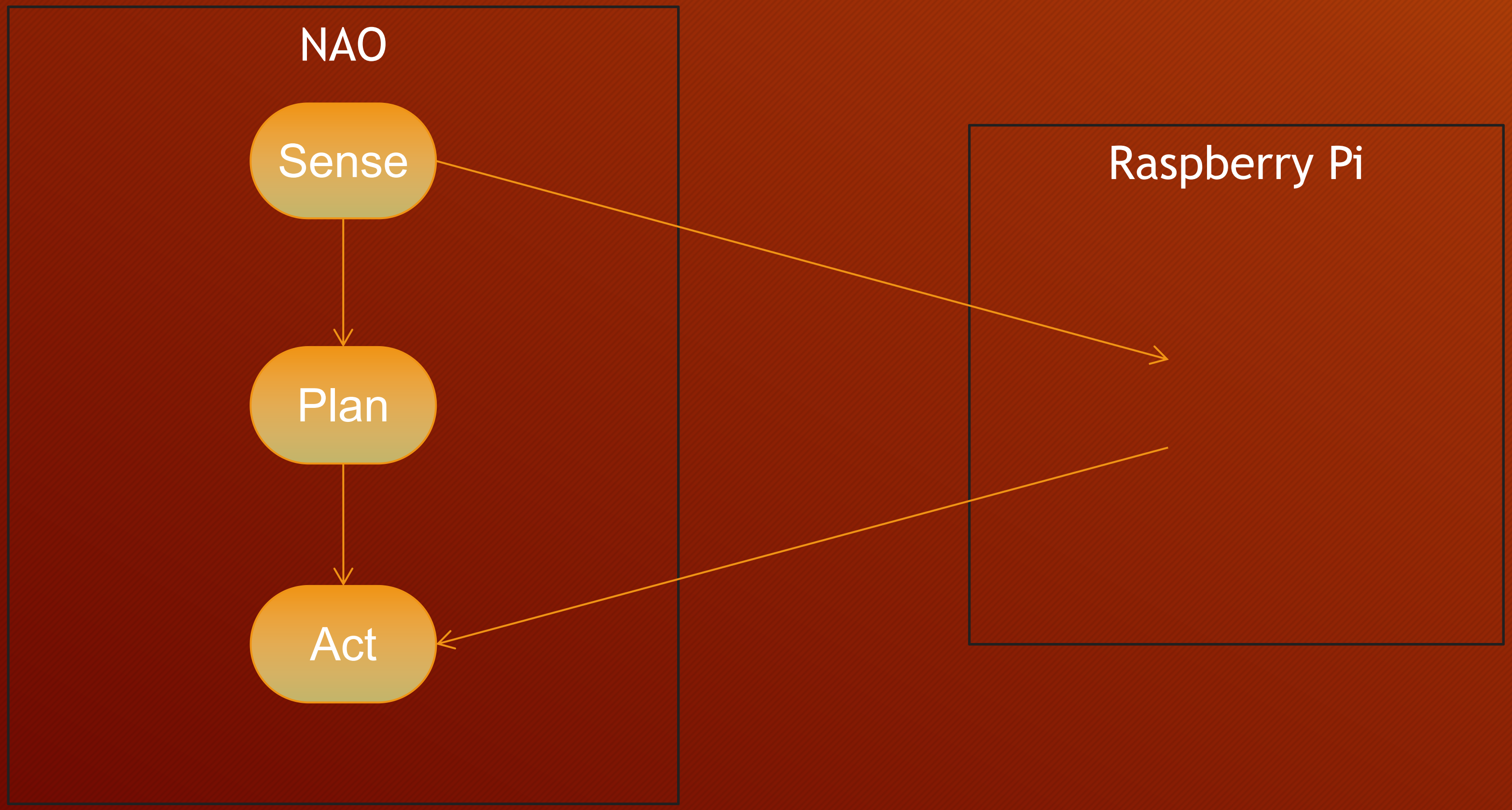
2. Target: take an object

- listen(S2T), see (object detection), search (AI), move, take, respond to exceptions (fall, obstacle)

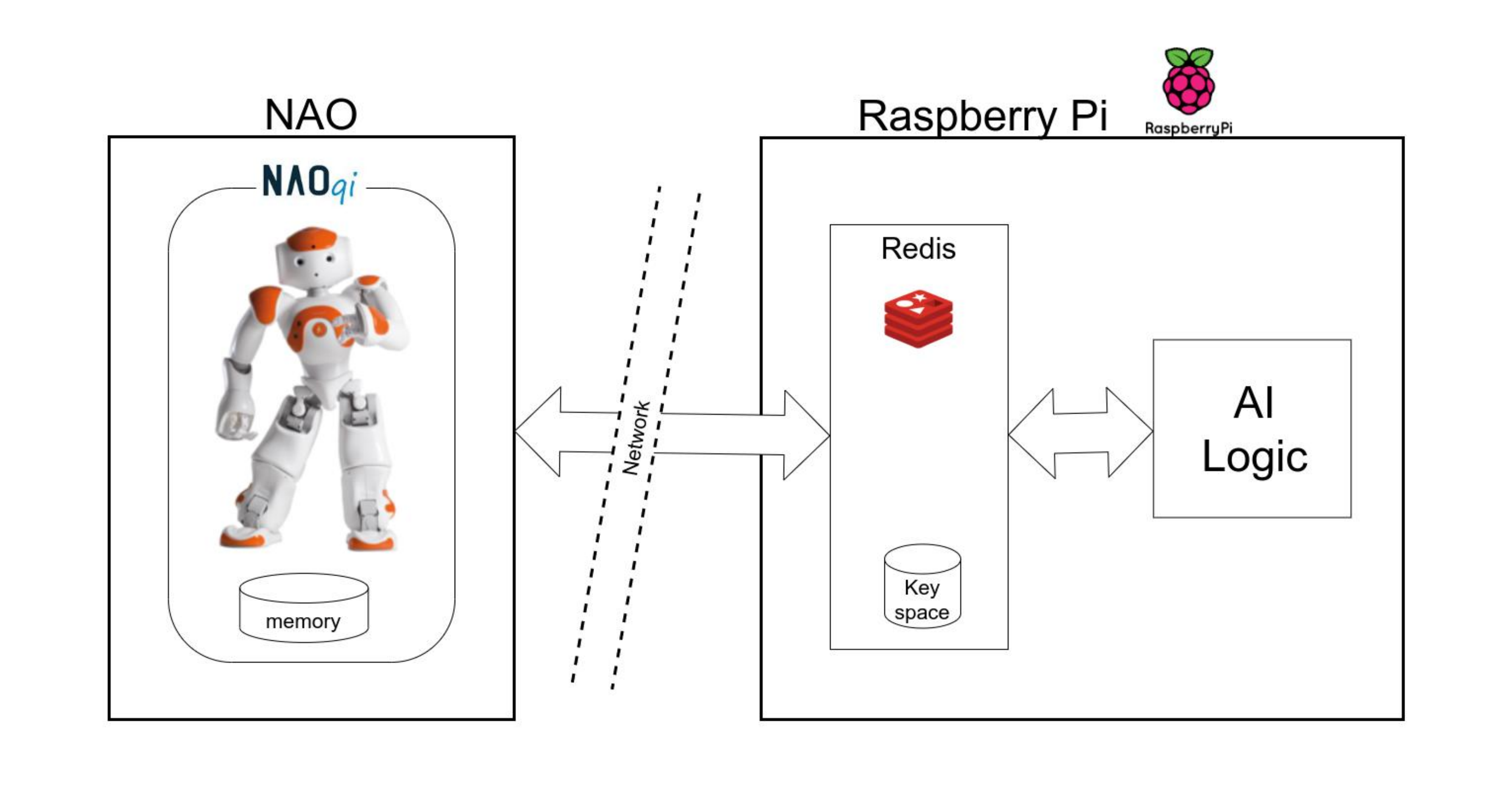
Interface

- Requisites:
  - work upon network
  - few requirements needed to be used
  - fast (CPU speed)
  - not entirely based on NAO

- ==> Redis: database/message broker, fast and distributed
  - channels, key, lists, …
- ==> Raspberry Pi (on knapsack)
  - increase of resources, reliable connection


NAO
Sense
Plan
Act
Raspberry Pi

Implementation Overview



- NAO different and separated from AI logic

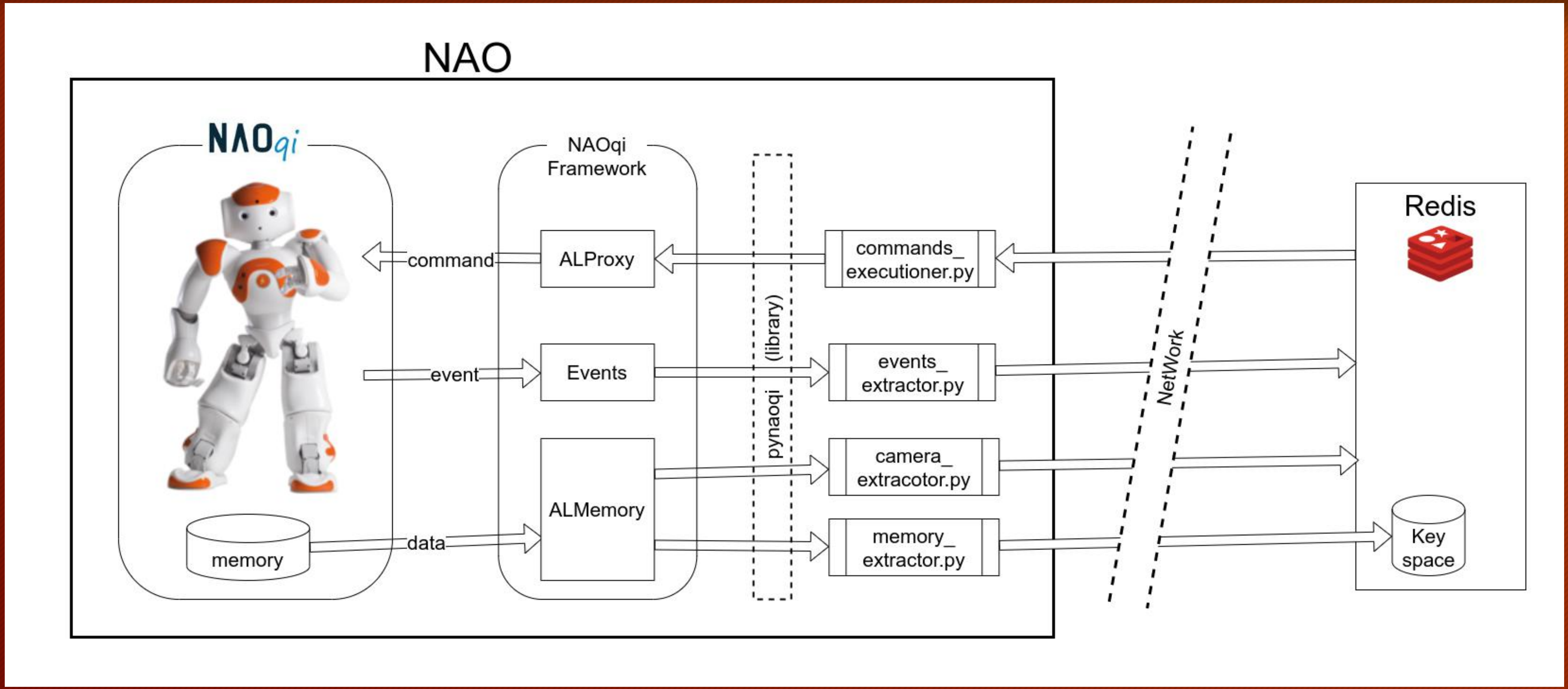
NAO Interface

NAO Software

- NAOqi is the software that controls the robot
- offers an interface via numerous Proxies and MemoryDB
- access all of above with PyNAOqi library (downloadable with NAOqi SDK)

- export everything to REDIS (over the network)

Implementation



- utilizes reflection and introspection techniques
- list of values / proxies to be exported / executed
- http://doc.aldebaran.com/2-1/naoqi/index.html

Resources - References

- Events: http://doc.aldebaran.com/2-1/naoqi-eventindex.html
- Memory: http://doc.aldebaran.com/2-1/naoqi-memoryindex.html
  http://doc.aldebaran.com/2-1/family/nao_dcm/actuator_sensor_names.html
- Commands: http://doc.aldebaran.com/2-1/naoqi/index.html

conclusion NAO interface

- everything said can be treated as a Black-Box
- we don’t need to know how it’s implemented on low level

Robot Controller

Raspberry Pi Software

- REDIS server
  - channels for: events, sensors, execution of commands
  - keyspace: memory

- AI - Logic
  - Python: iterative code
  - ROS: nodes, topics and services communicating
  - QuLog: declarative language, very powerful

My software

- **Goal**: find and take an object
- **Abilities**: move, see (object detection), listen (speech-to-text), sense the world, take an object (es. ball, duck)
- **Knowledge**: shape and dimension of the object, height of table
- **Stimuli**: camera, sonar, microphones, gyroscopes/accelerometers

- **assumptions**:
  - determine action 1:1 from sensor reading

Steps

1. Listen to the object to be taken (speech recognition & S2T)
2. Perform Object Detection on camera images
3. Wander around the room, searching for the object
4. Approach the object
   - Respond to exceptions (obstacle, fall)
5. Grab it

Listen

- Perform speech recognition + Speech-to-Text
- We use functions provided by NAO

Move

- Use of commands of ALMotionProxy and ALRobotPustureProxy
- We set the speed for each axis and NAO does it all of its own

Grab

- Once near the object, we perform a series of pre-defined actions
  - raise arms to avoid the table
  - join arms
  - close hands around the object
  - lift it
- There is currently no response after each action
- Only at the end we can see if there is the object in hands

Object Detection

- Very difficult to perform OD on RPi.
- Following a brief list of what I tried and went wrong:

1. Cannot install TensorFlow on Raspberry Pi (32-bit)
2. Cannot compile it from source (1GB RAM and slow CPU)
3. Cannot install OpenCV on raspberry (same issues as above)
4. Sol.1 ROSBots!: OS image with ROS preinstalled and OpenCV precompiled on board.
5. OpenCV Yolo/Darknet takes about 30s to recognize the image.
6. Sol.2 OpenCV blob detector, based on colour and shape of the object

Object Detection

- Blob Detection w/ OpenCV
  - i.e. search the object based on color and shape
- Previously learn color and shape of the object.
  That constitutes the Previous Knowledge of the robot
- Then, with OpenCV + NumPy, we can detect the object (with a degree of reliability)
- Returns normalized center and radius of the object, based on levels of minimum acceptable reliability

Approaching: synchronization

- wait for a new image
- pop from list
- perform object detection

- listen for events
- update robot perceptions

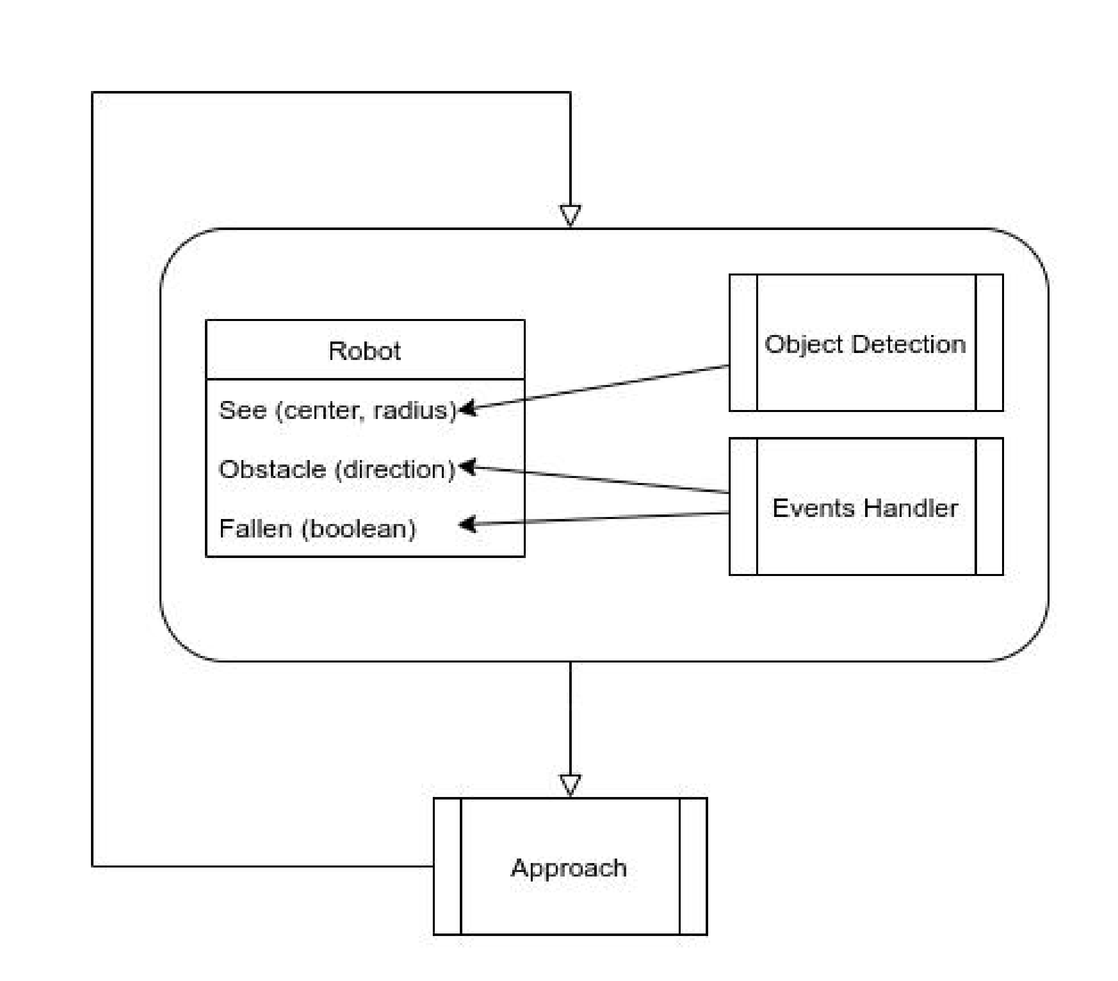
- choose the actions based on the perceptions
- stop to wait new updated perceptions

Approaching: actions

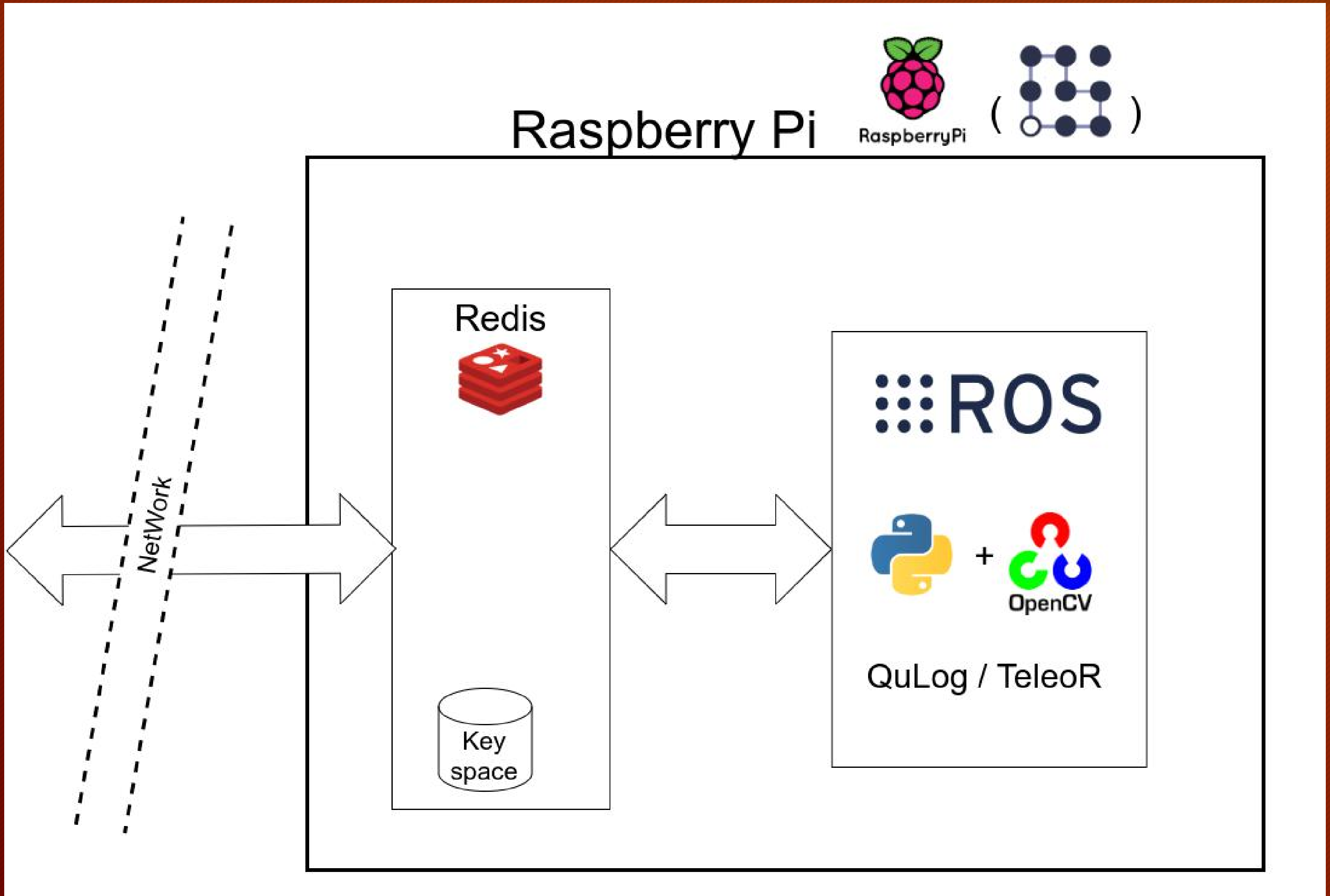
- Based on the perceptions: see_object, center_object, obstacle, fallen

```
if fallen: get up
elif obstacle: move sideways
elif see_object is False: move around
else:
    if center < 0.5 : move left; elif > 0.5: move right
    if radius < size : move forward
    else : grab the object
```

Approaching: block diagram

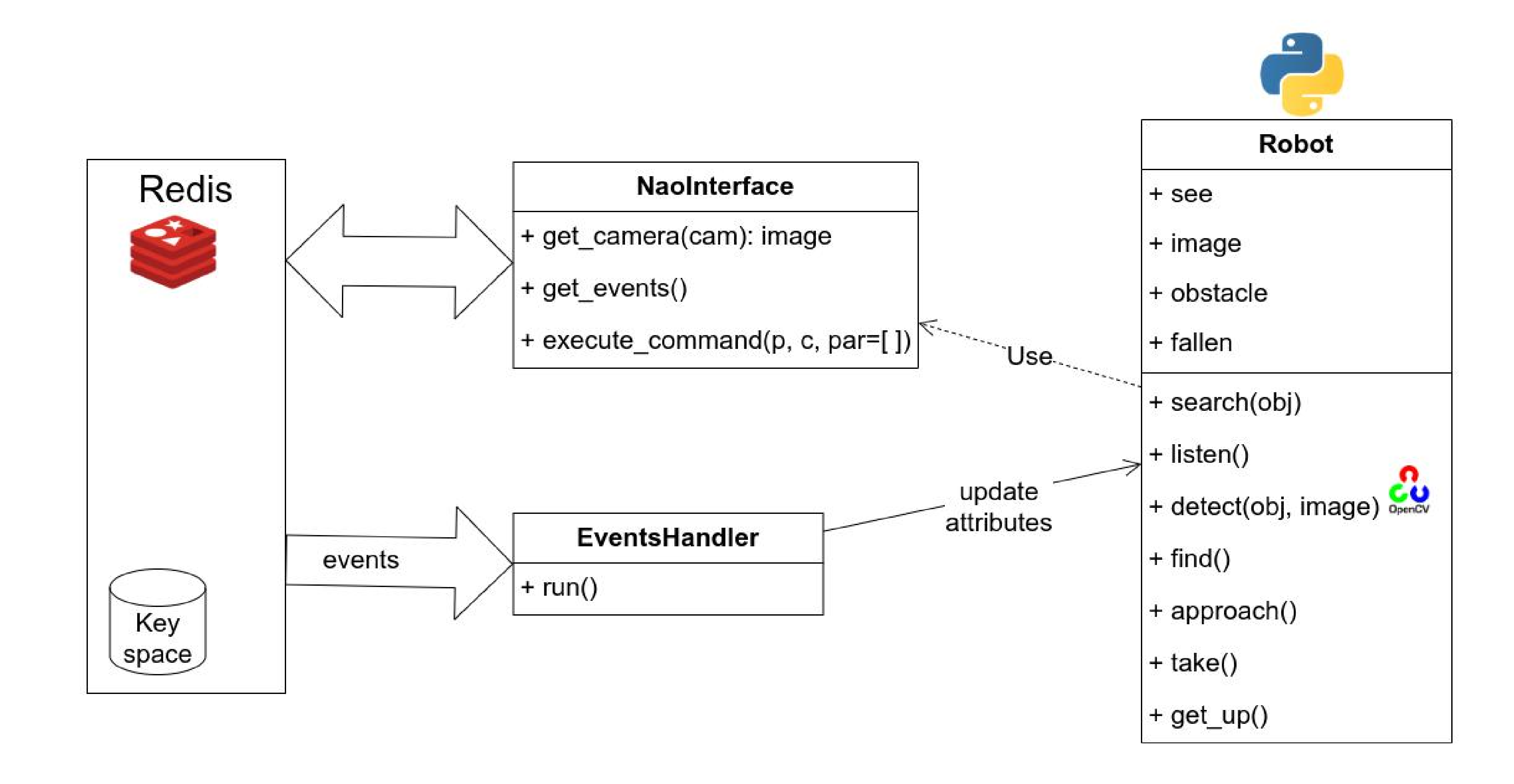


AI logic


Raspberry Pi
RaspberryPi
NetWork
Redis
Key space
ROS
+
OpenCV
QuLog / TeleoR

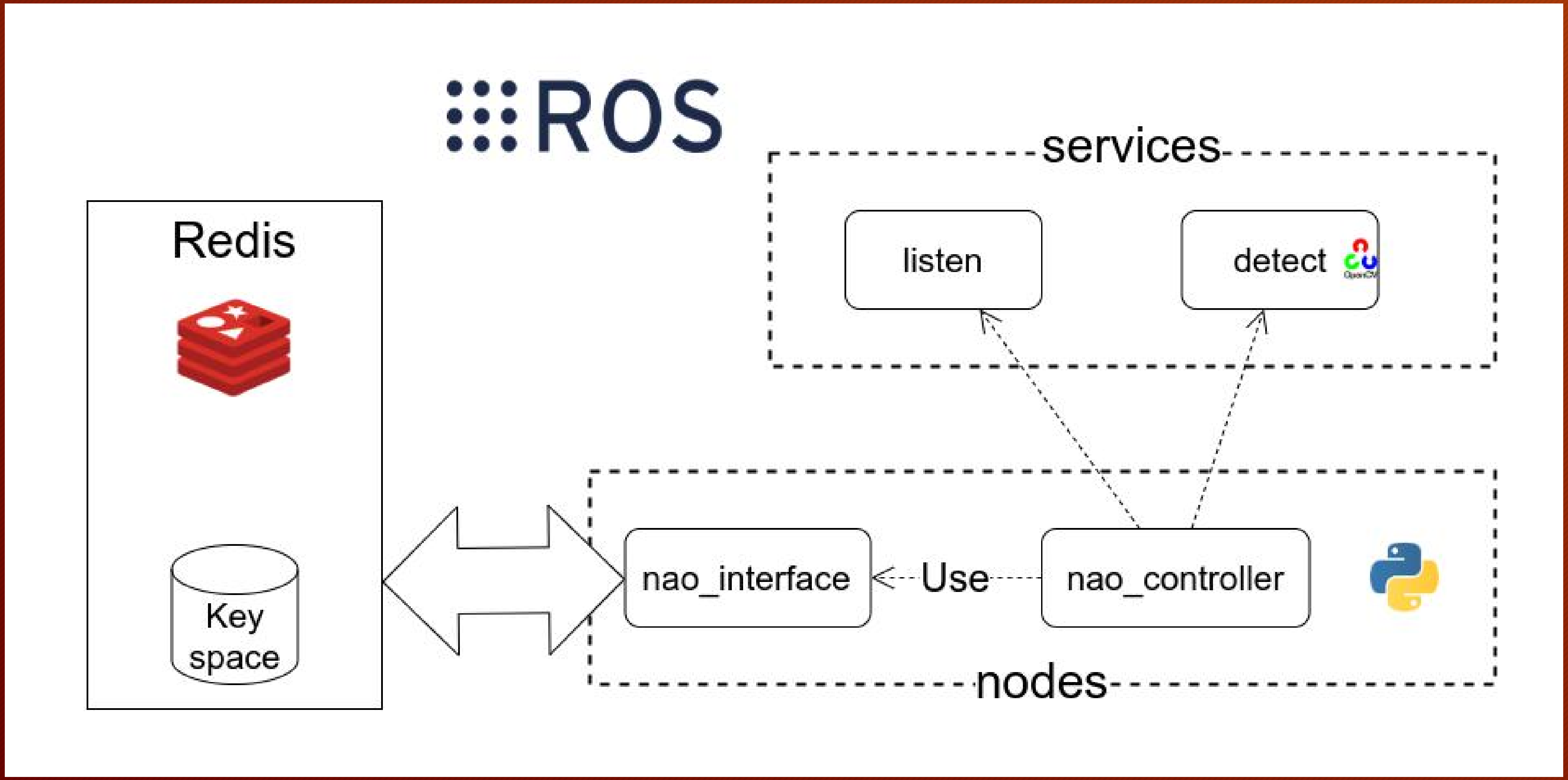
Python

- All the logic using only python



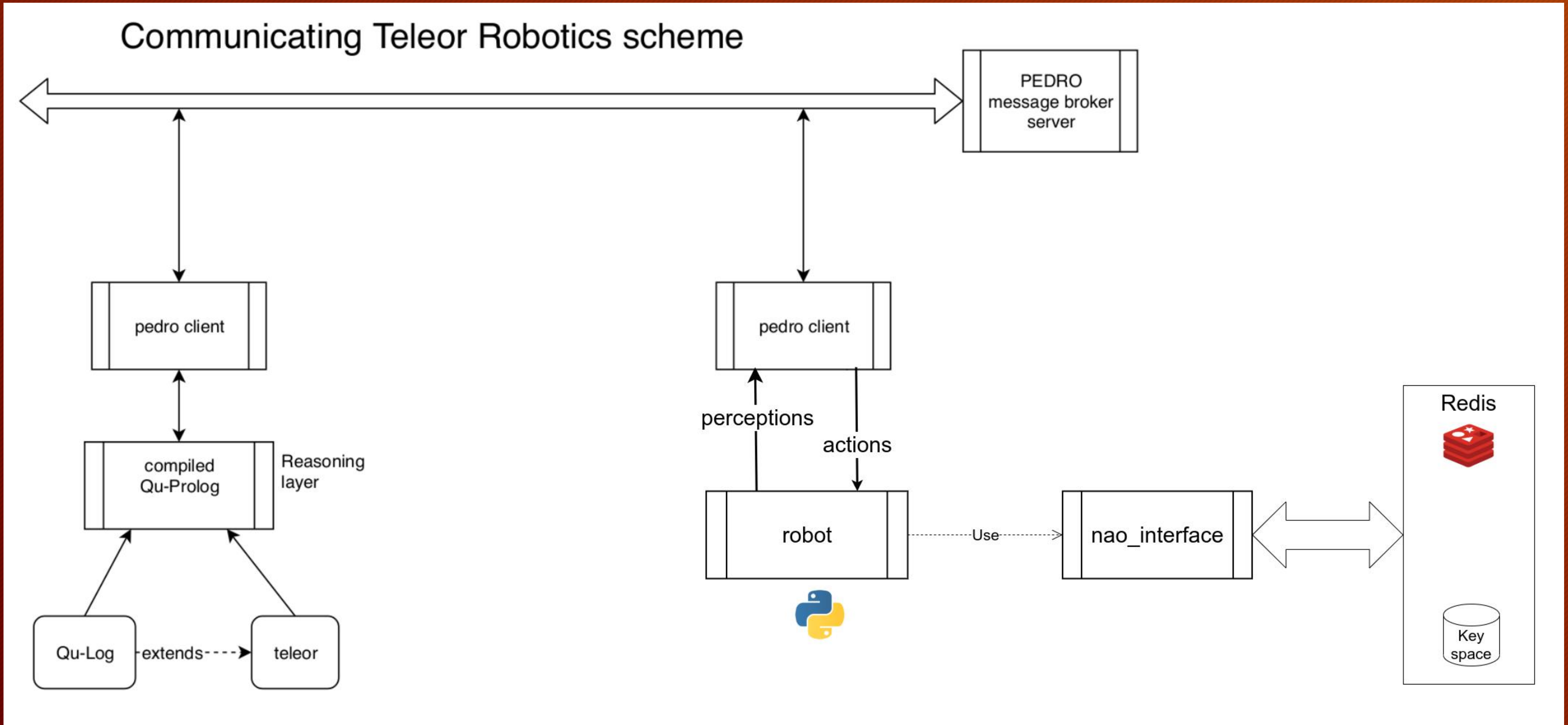
ROS

- implemented using ROS, but with the same logic

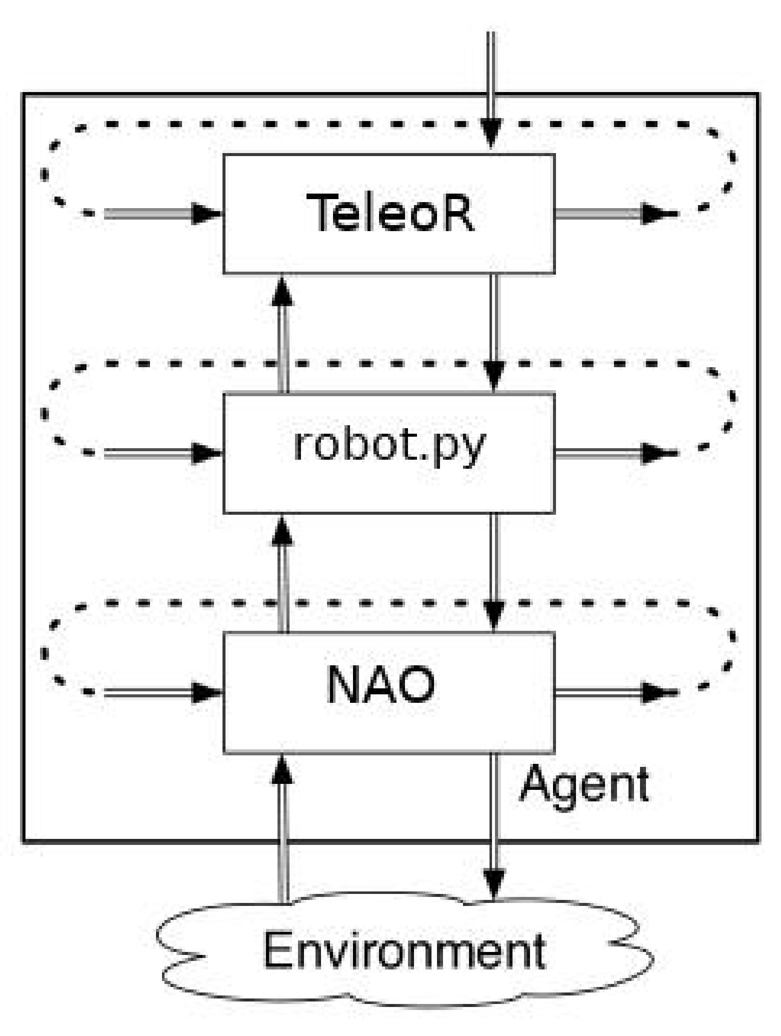


QuLog

- **Declarative** implementation using QuLog / TeleoR



TeleoR - abstraction



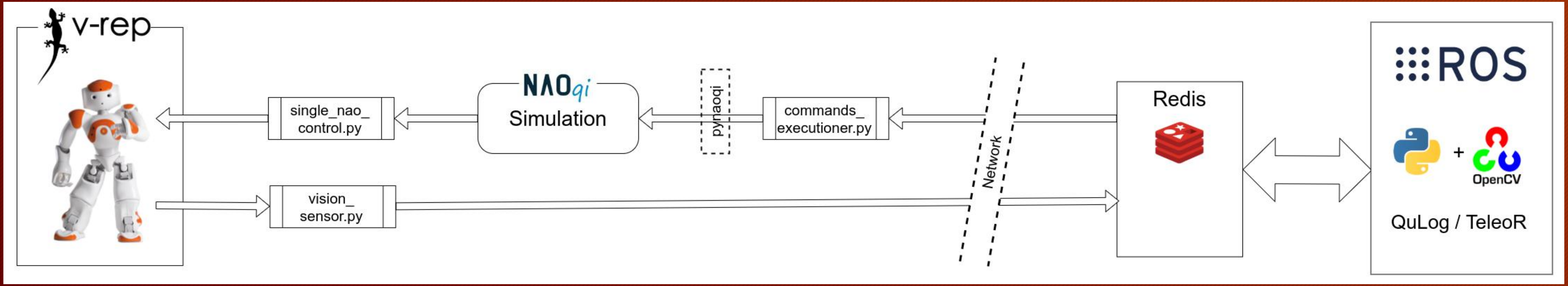
- we have a double level of abstraction

Conclusions

Simulation: everything on same computer

- VREP (CoppeliaSim)
- NAOqi simulator: obtainable with Choregraphe
- PyNAOqi SDK
- Redis server

- ref. README.md on repo



Test on simulator

- [nao_simulation.mp4]

Test on simulator: obstacle

- [nao_obstacle.mp4]
  - The sonar event are not generated by VREP, but instead simulated via an ad-hoc python script
  - The robot responds well to the events

Test on simulator: QuLog

- [nao_qulog.mp4]

Real World test

- [VID_20201103_171625.mp4]

- grab: [VID_20201103_171625.mp4]
  - The robot doesn't sense if he has taken the ball, instead it executes a series of predefined actions to take it.

Conclusions

- it’s just the beginning….
- The interface works for every other piece of code we will ever develop
- We can extend the code with more environment responses
- We can implement more complex patterns to take the object

The End

- Thank You!
- Questions?

Gabriele Tagliente