从一点点历史开始

Nao 2008
Pepper 2014

Robots & Services

We create robots and services that define the roles robotics play in our lives today, and in the future.

Global Network

SoftBank Robotics

Asia
※Except China and Taiwan

SoftBank Robotics Europe

Europe/
Middle East/Africa

SoftBank Robotics America

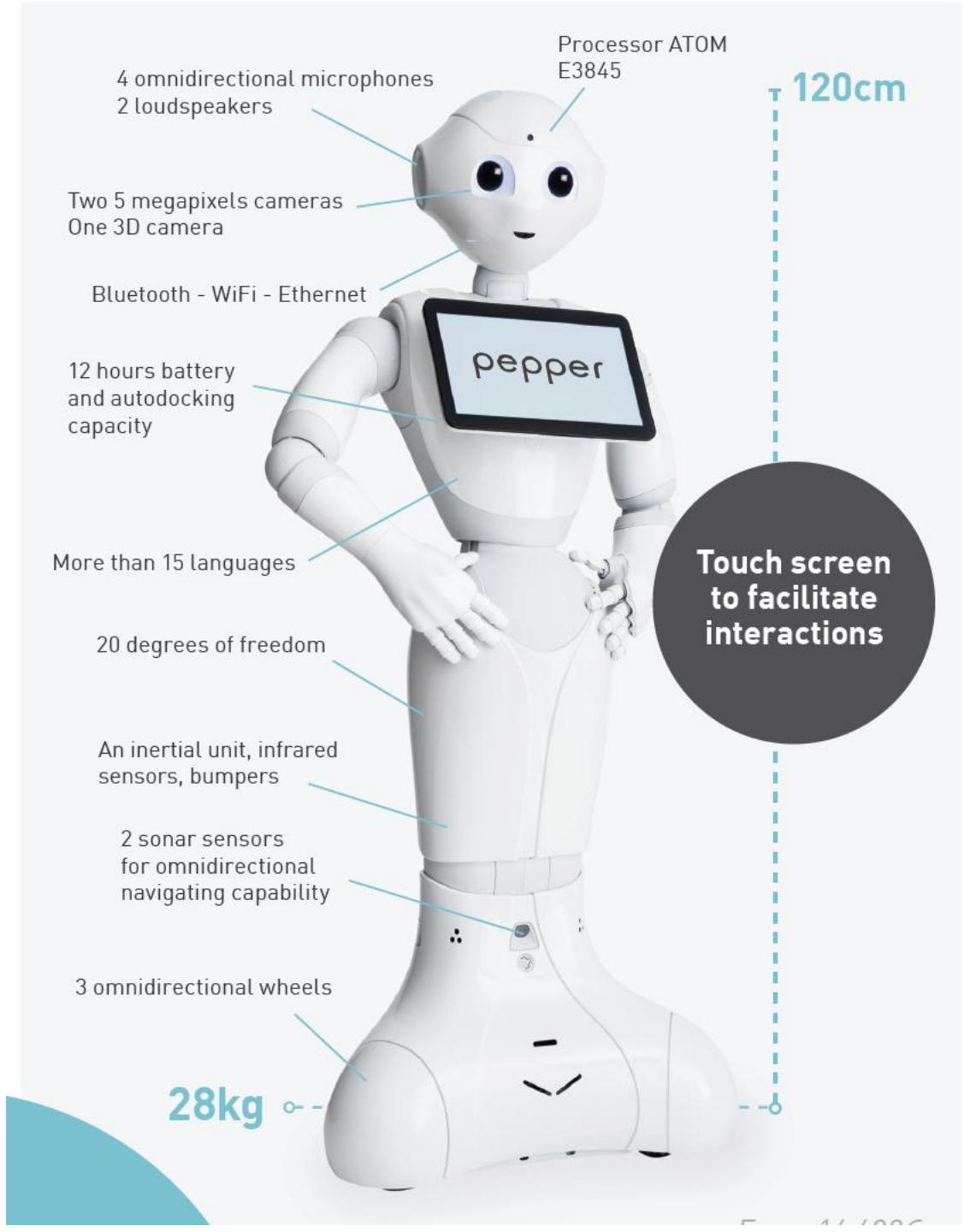
Americas

SoftBank Robotics China

Greater China

Pepper 机器人

完全可编程的人行机器人综合平台



欢迎!

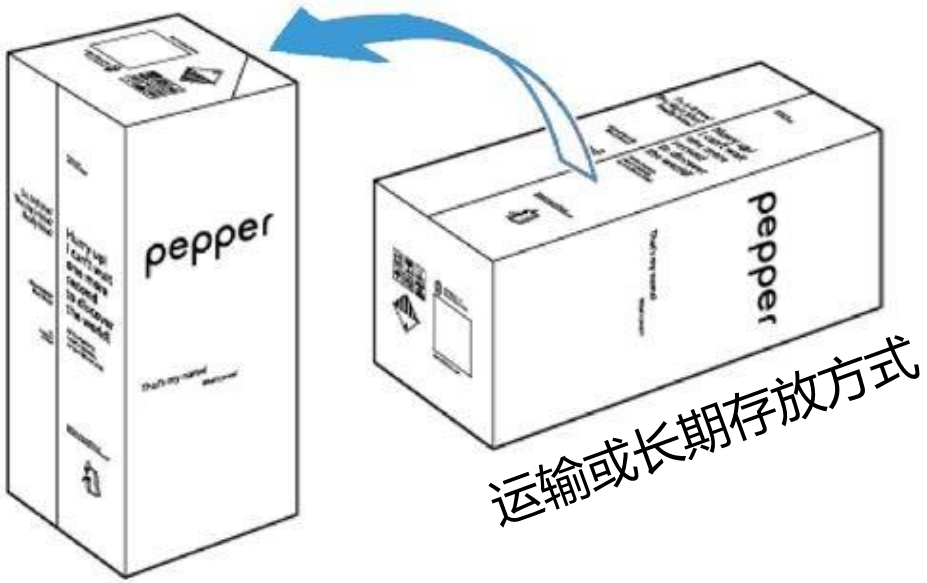
准备好探索你的新机器人!

主要内容:

- 第一章: Pepper 入门,硬件1.8/1.8a
- 第二章: 技术参数, 人机交互, 软件, 软银机器人开发工具
- 第三章: ChoregraphePython服务: 系统APIs, 交互APIs, Tablet APIs
- 第四章: 2.9介绍,

机器人拆箱

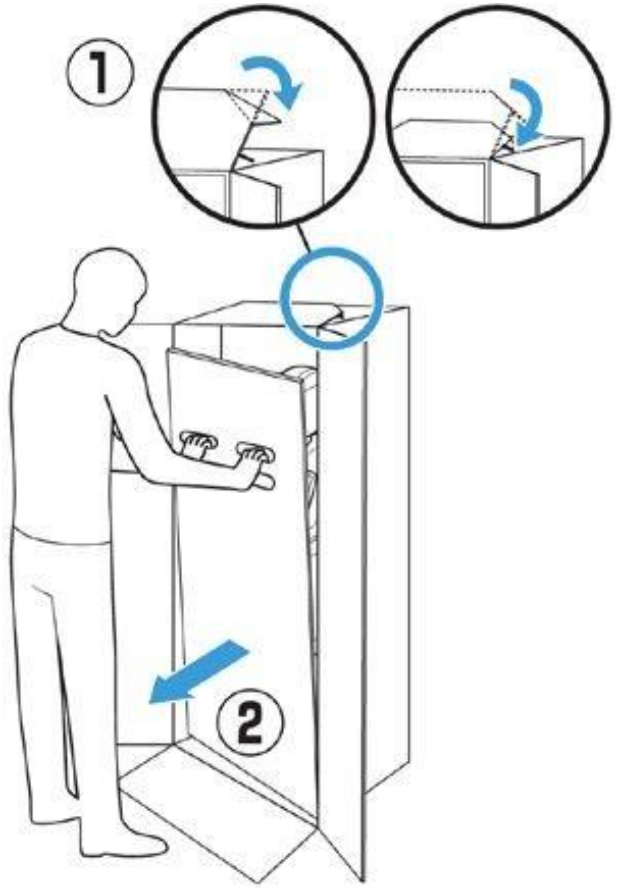
1) 将箱子竖立起来



运输或长期存放方式

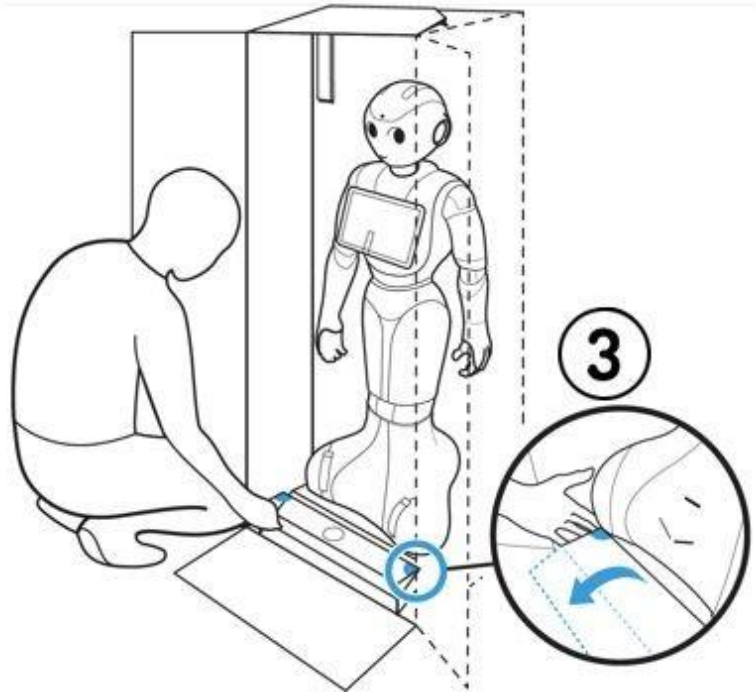
短期存储方式

2) 打开箱子并移除盖板



机器人拆箱

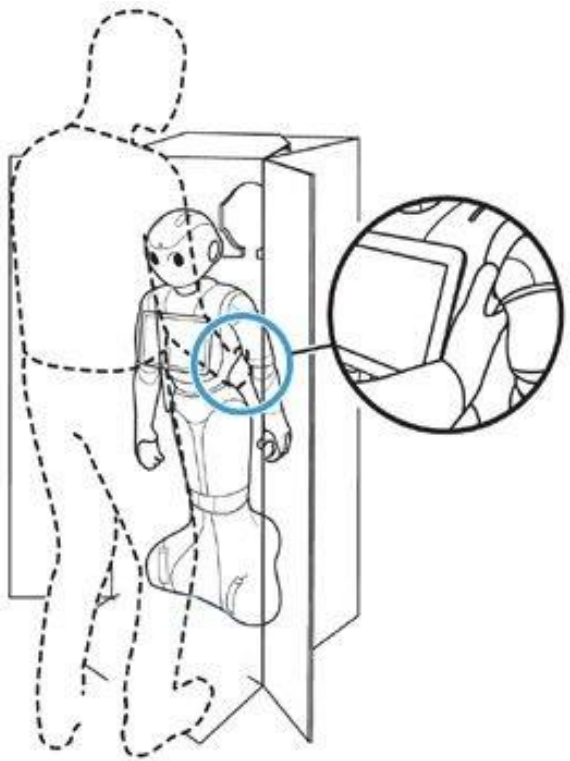
3) 铺平斜道



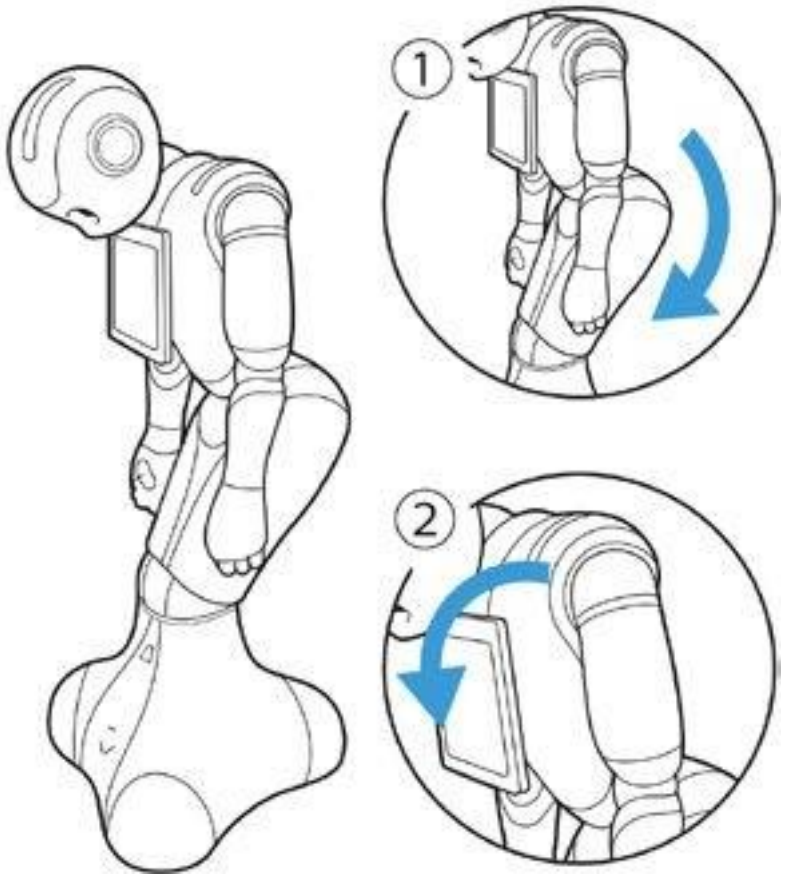
4) 将头和手臂拨出

机器人拆箱

5) 将双手放到Pepper的腋下, 然后将Pepper从箱子中抬出来

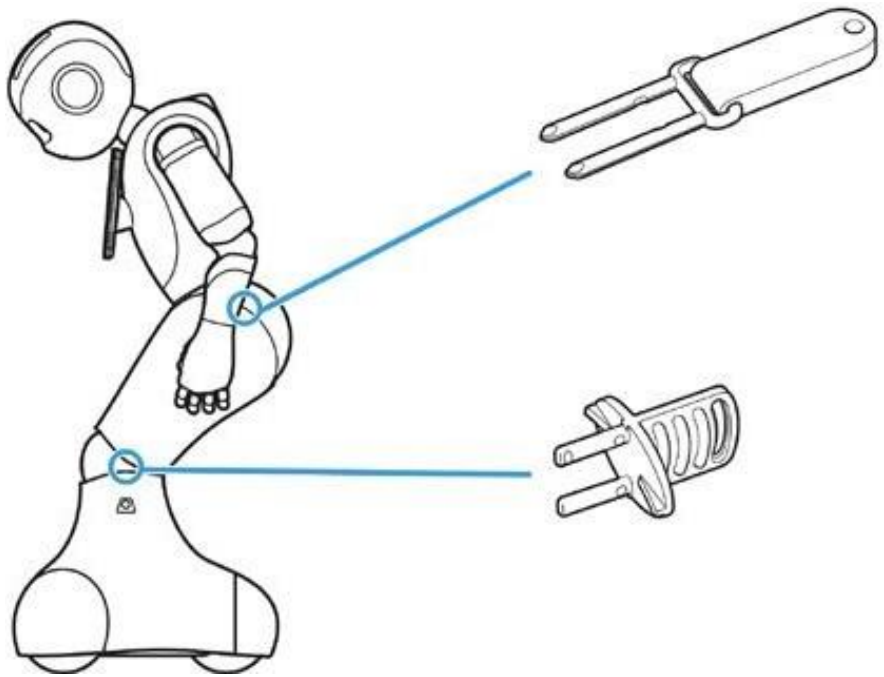


6) 将Pepper摆放在休息的姿势

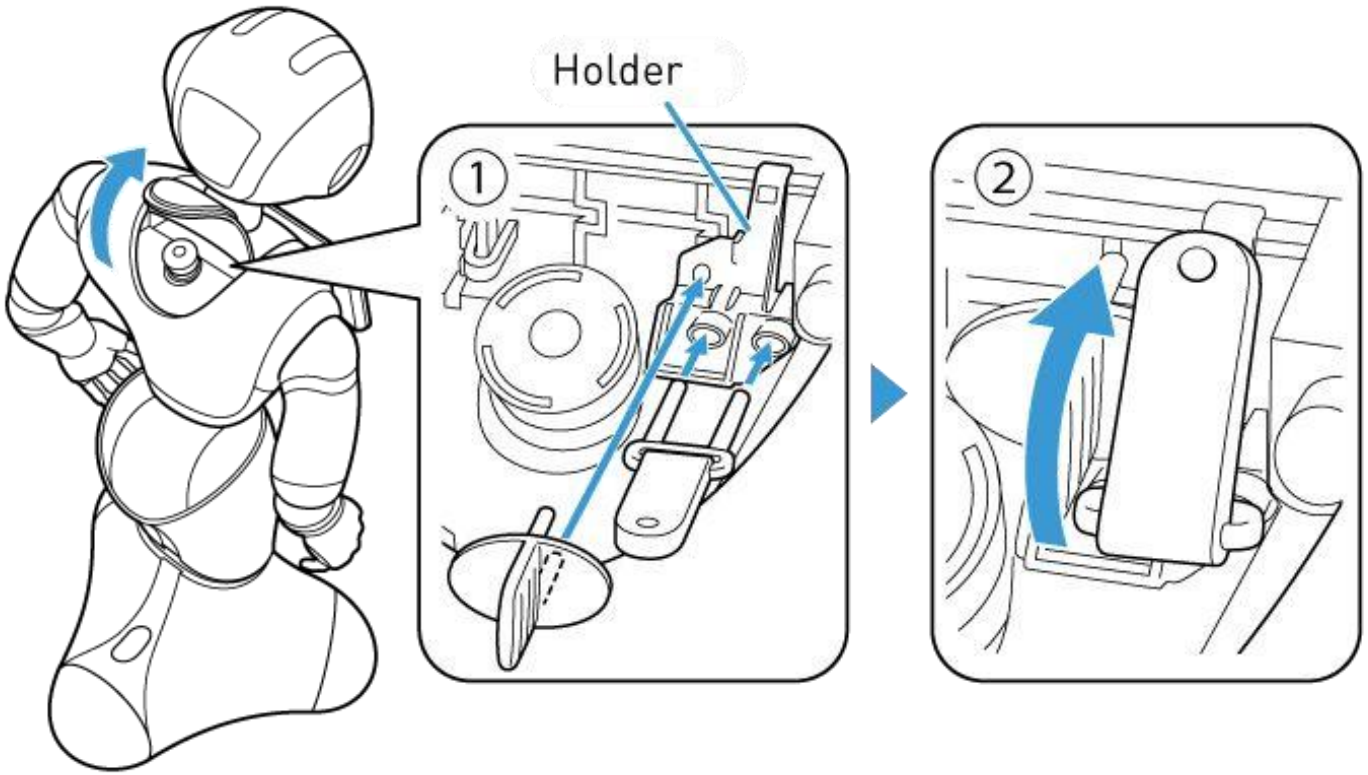


机器人拆箱

7) 移除2处的插销

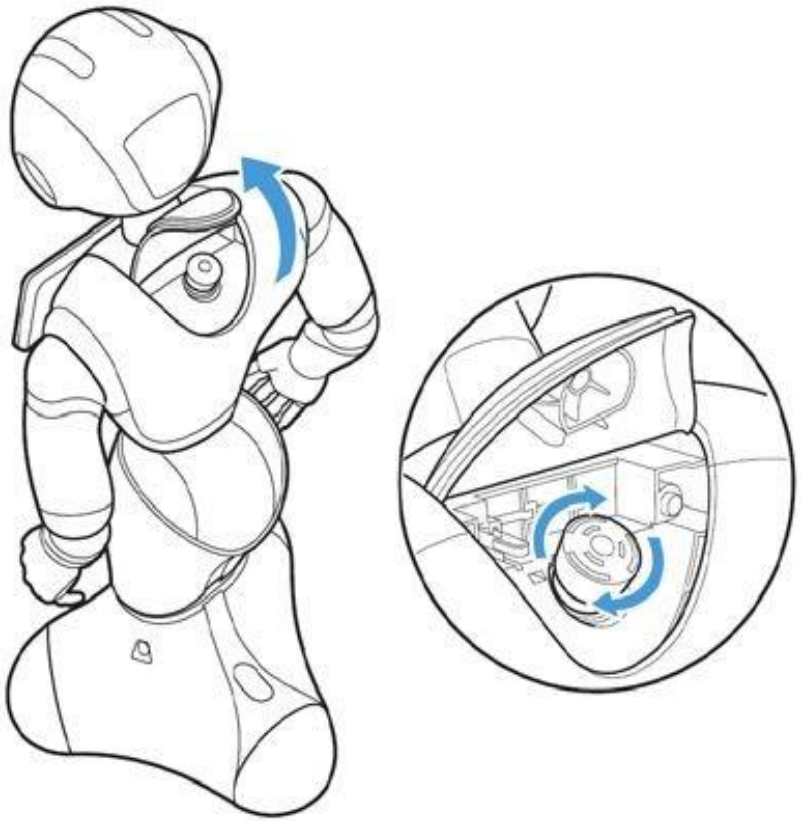


8) 打开Pepper颈部后面的软盖，将插销按位置存放妥当

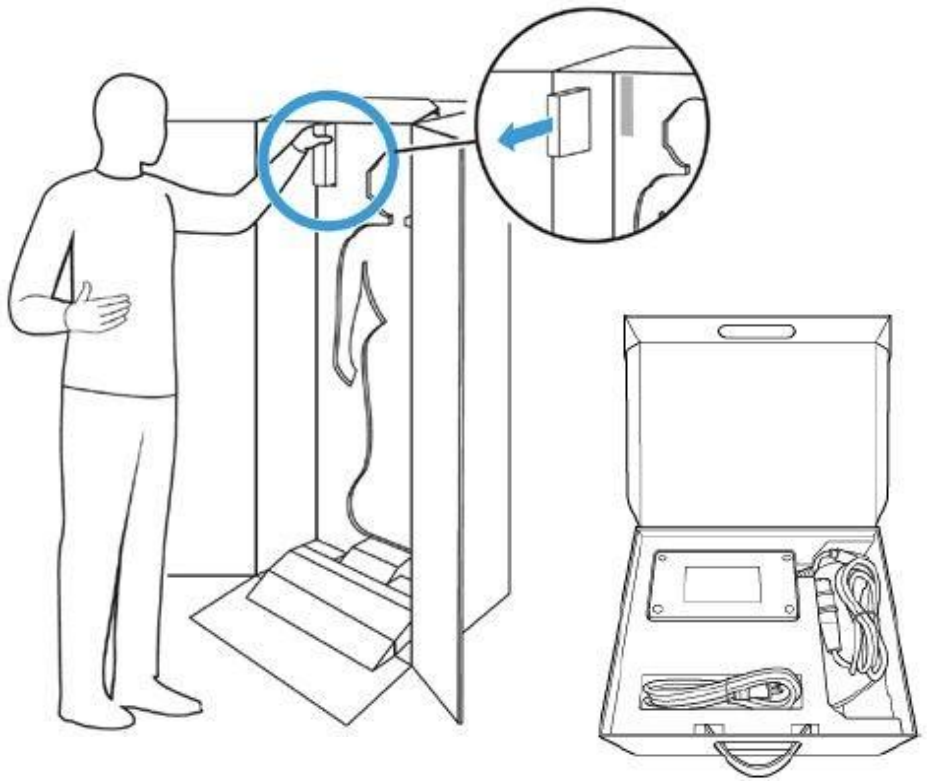


机器人拆箱

9) 解锁紧急制动按钮



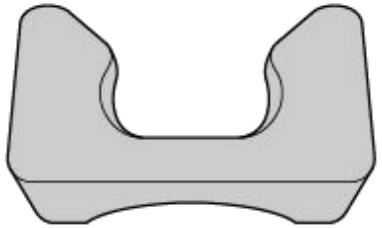
10) 取出充电器



机器人拆箱

剩下的东西:

- 平板的保护罩



- 充电器盒子

将这些存放在箱子中，你需要这些来存放或运送Pepper

恭喜你！你的机器人已经准备好了！

姿势

休眠(Rest): 安全姿势

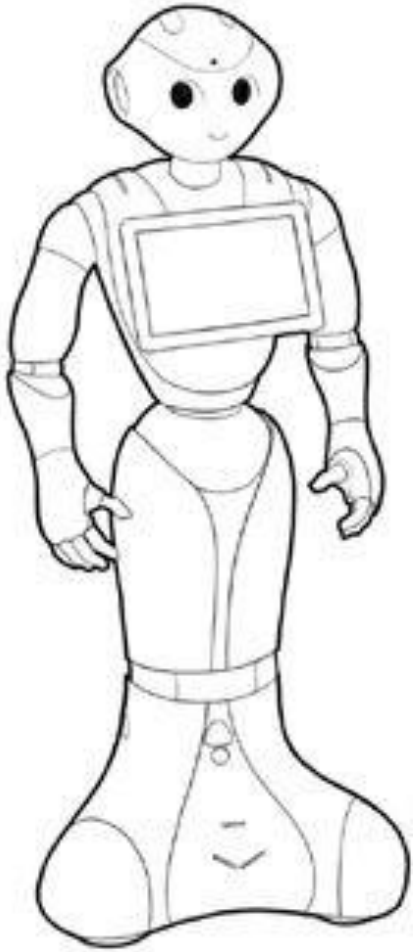
- 头朝下
- 膝盖和臀部弯曲

以下情况下使用:

- 电机关闭
- 休息模式
- Pepper关机状态

站立(Standing): 工作姿势

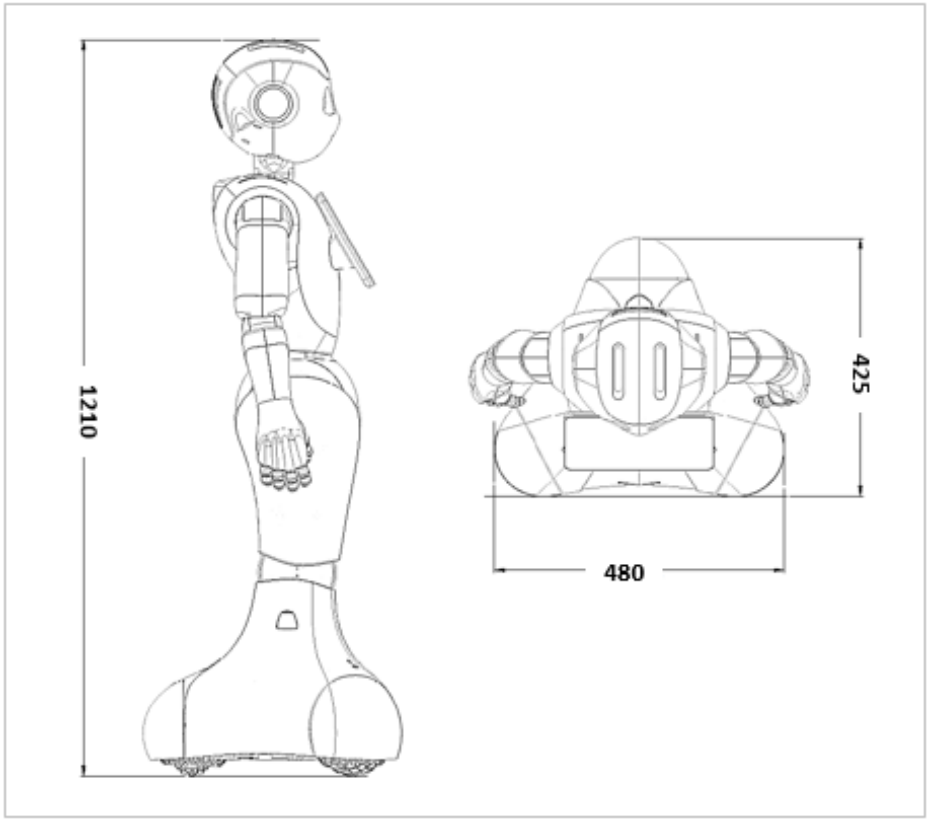
- 站立，
  双手置于身体两侧
- Pepper已经唤醒，可以使用

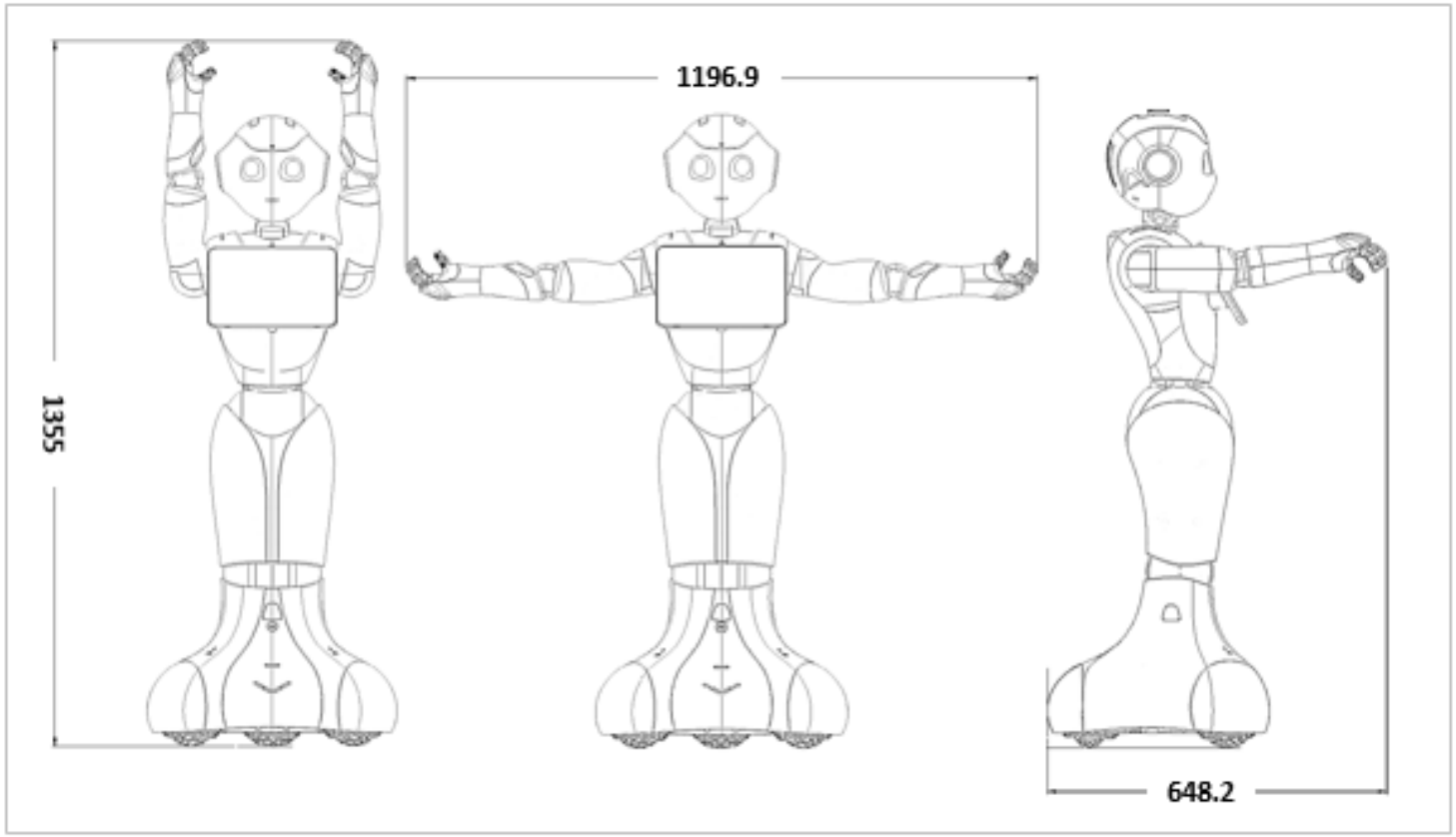


硬件介绍（基于1.8a/1.8）

硬件介绍

Pepper外形尺寸

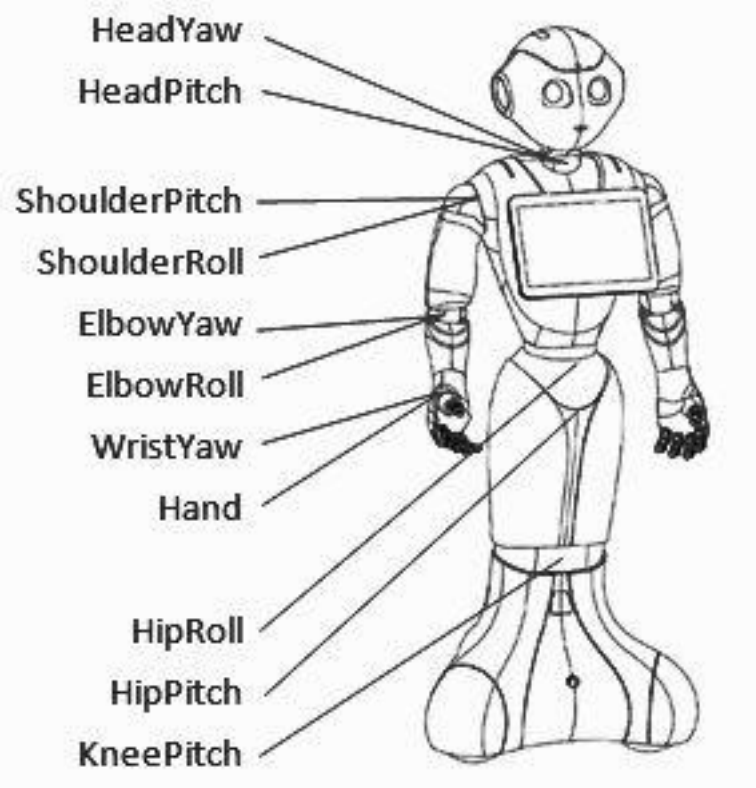




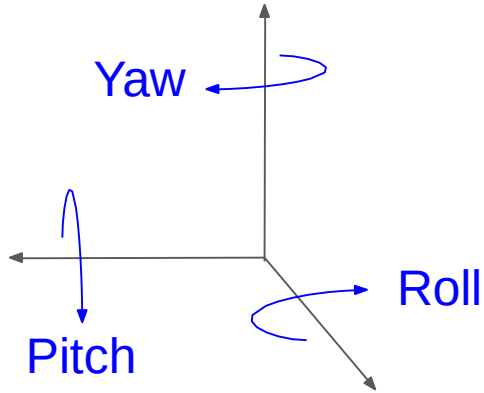
硬件介绍

执行器

Leg 电机 BLDC 直流无刷 大扭矩
手臂、头部电机 主流有刷空心杯电机

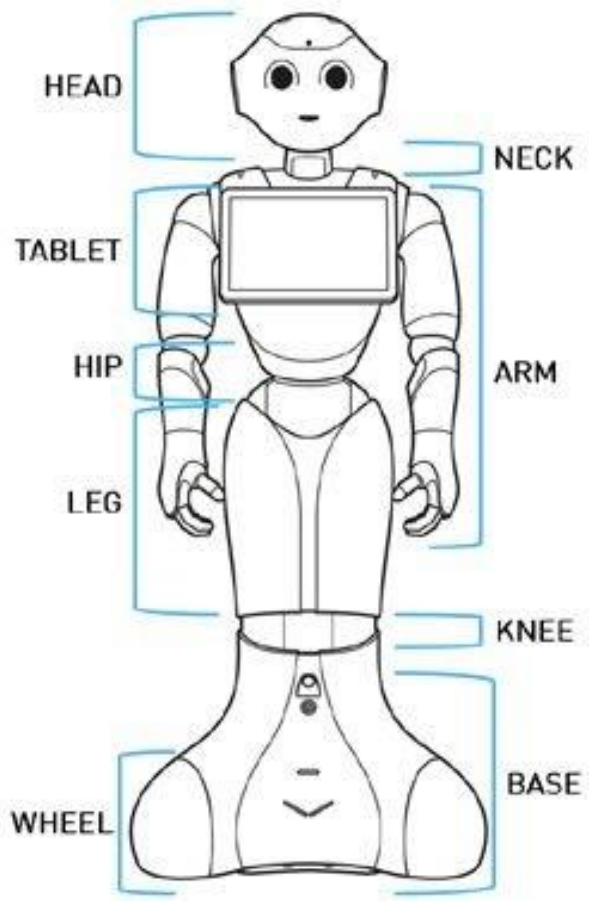


根据关节和方向来命名电机:

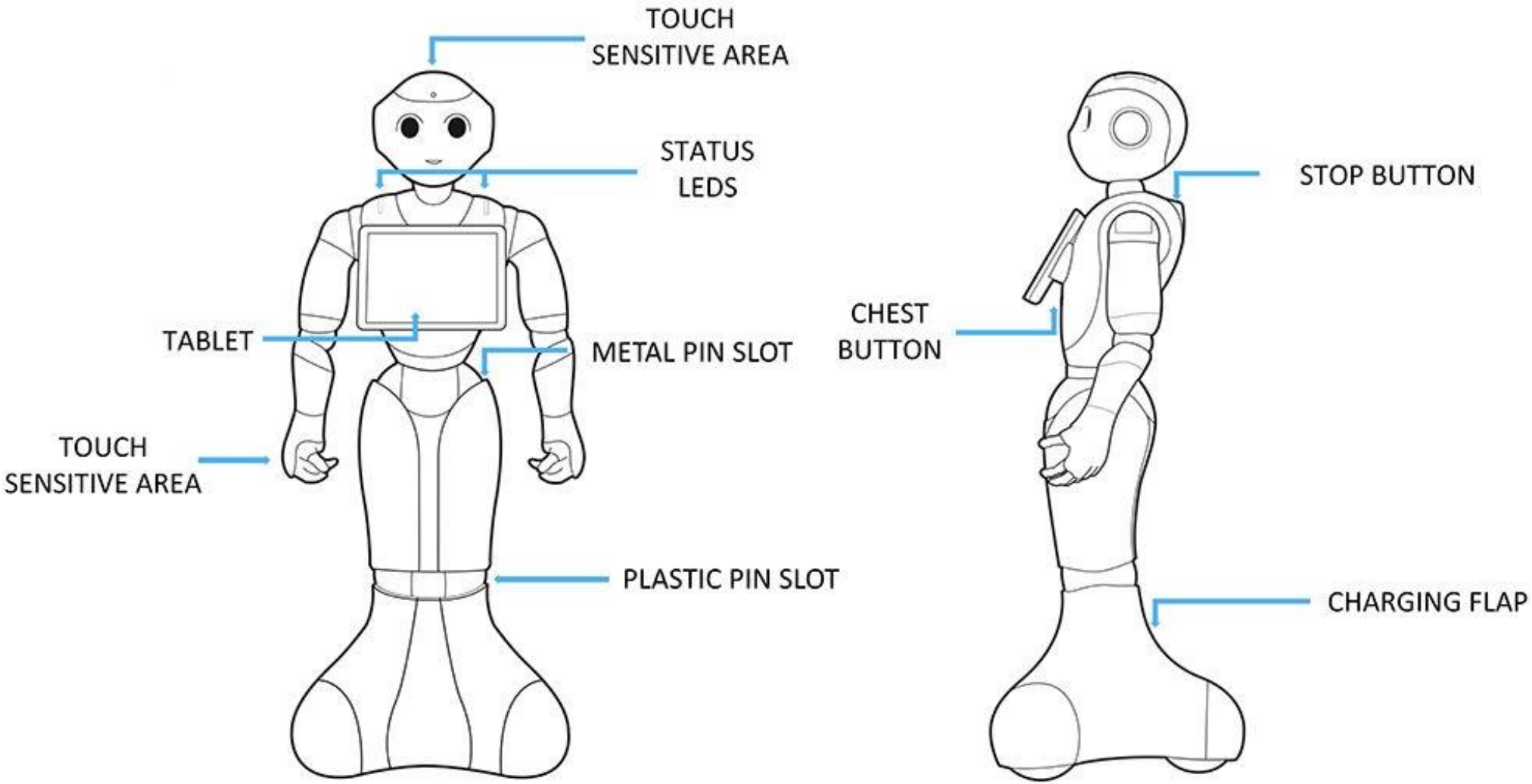


硬件介绍

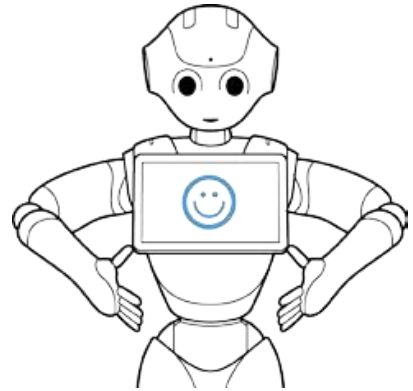
Pepper分为数个部分



硬件介绍

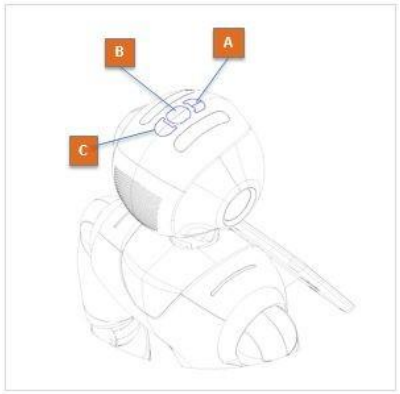

TOUCH
SENSITIVE AREA
STATUS
LEDS
TABLET
METAL PIN SLOT
TOUCH
SENSITIVE AREA
PLASTIC PIN SLOT
CHEST
BUTTON
STOP BUTTON
CHARGING FLAP

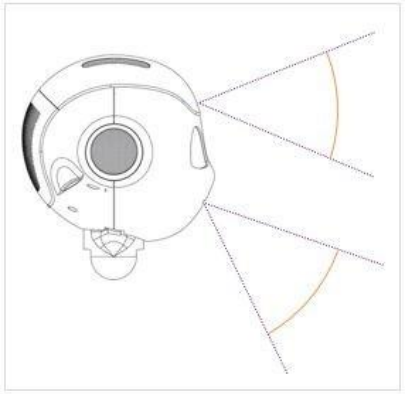
硬件介绍

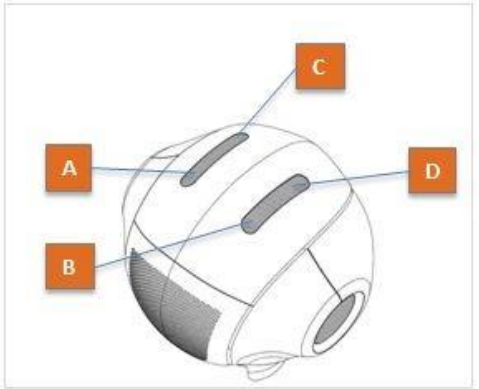


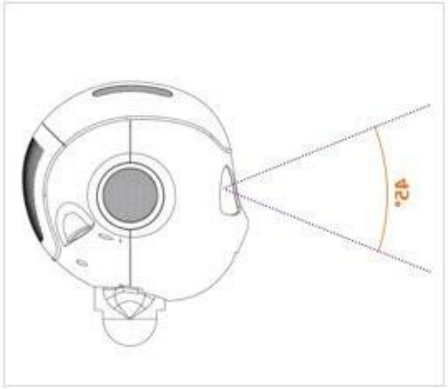
传感器

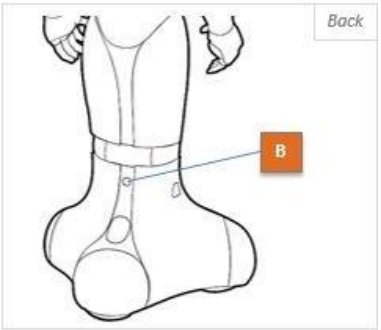
激光、红外传感器、声呐、触摸传感器、2D/3D摄像头

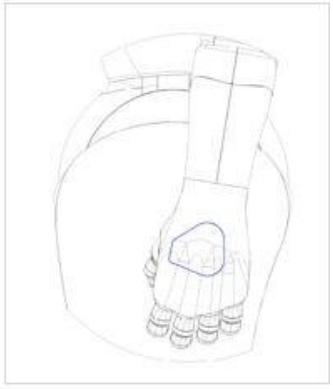


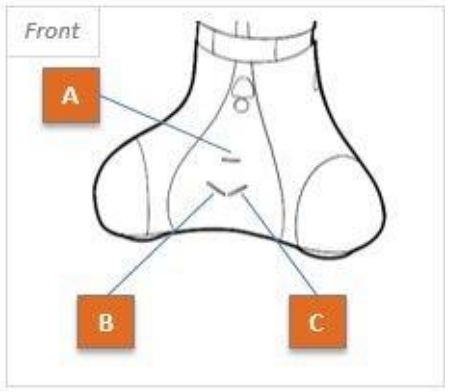


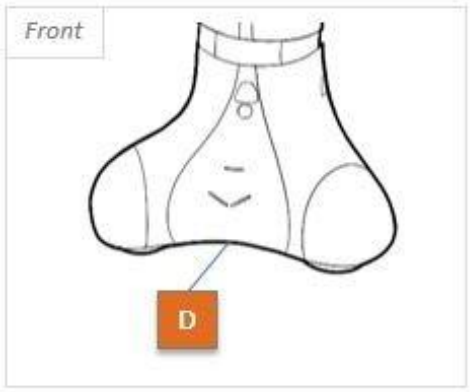


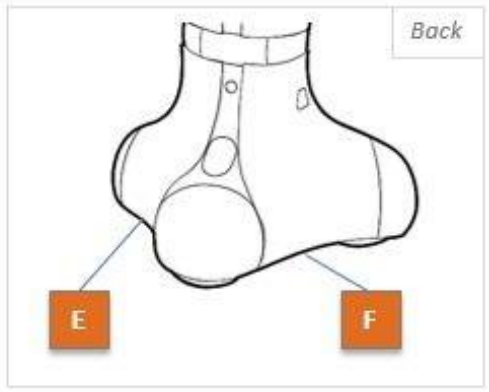


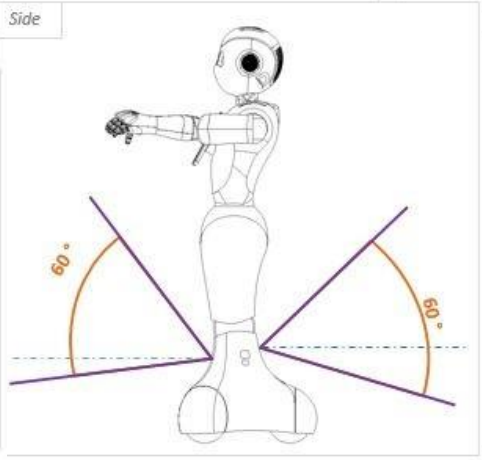


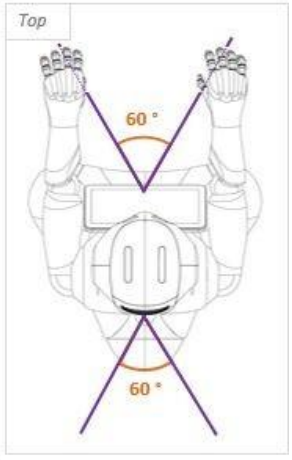




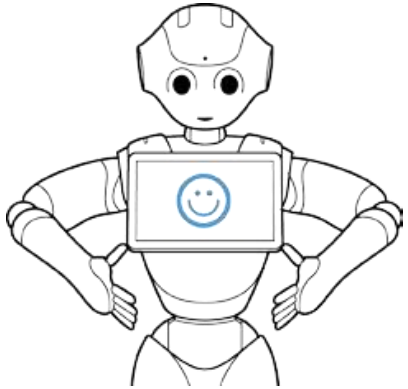




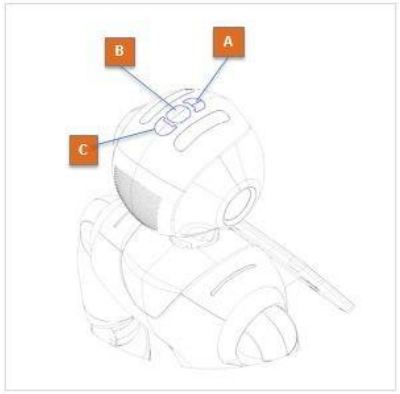


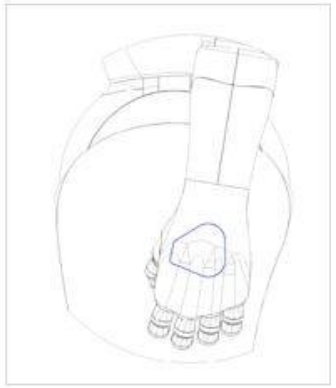


硬件介绍



传感器

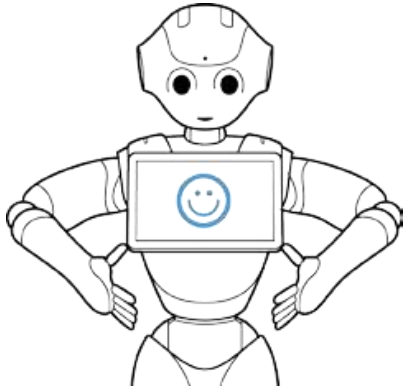




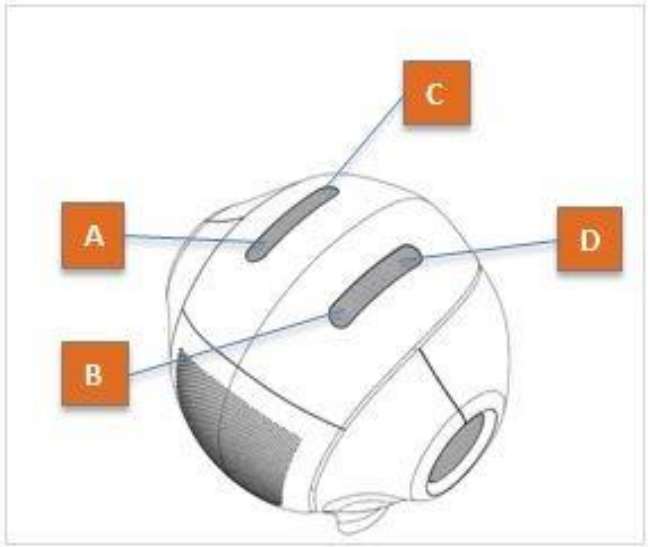
头顶处触摸传感器（三个）：Doc

手背处触摸传感器（双手各一个）：Doc

硬件介绍

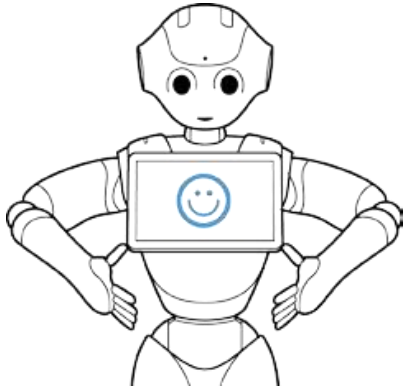


传感器

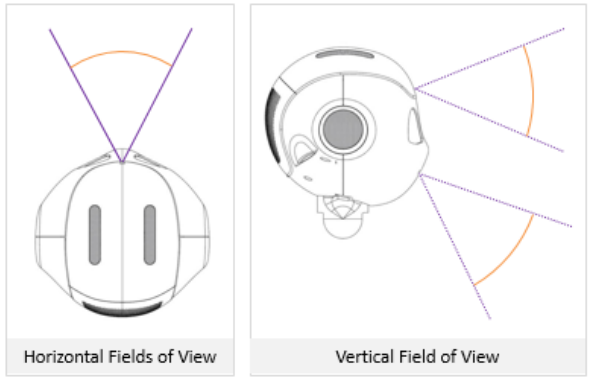


麦克风：头顶网罩下面（4个）Doc

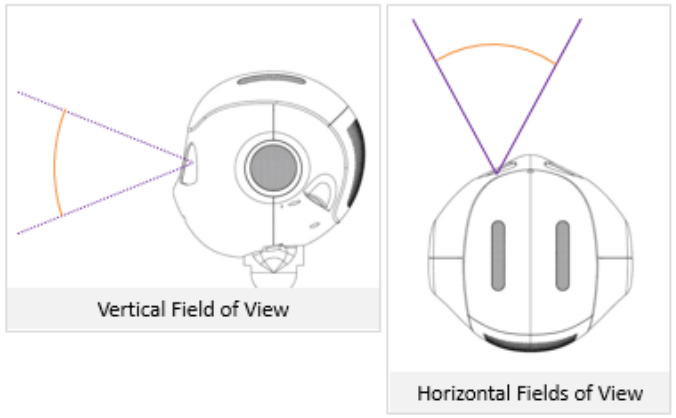
硬件介绍



传感器



2D摄像头：额头/嘴巴处 各一个



3D摄像头：左眼处 一个

Doc

硬件升级

Hardware improvement

Pepper 1.8

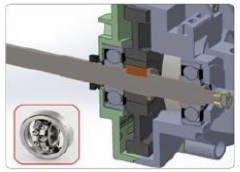
HEAD
New Eye board
New Face board
New USB Ethernet flex
New Top skull board
New Top & Bottom camera flex
New Carrier board
New audio codec
Audio vocal recogntion improvment
Trusted platform module (data security)
New CMOS battery
New WiFi + Bluetooth module
1 Fan removed for noise reduction
New Head structure

3D camera
Stereovision technology

NECK
New neck connection
New USB communication

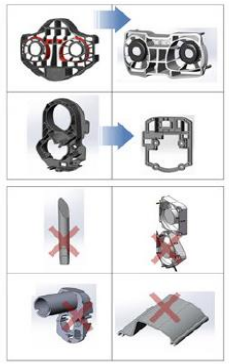
ACTUATOR (Neck-Shoulder-Elbow)
New Spur gear
New Top Satellite
New Bearing

FINGERS
New root & middle knuckles
New Hand Comb



WHEELBOX
Wheelbox Bushing replaced by bearings

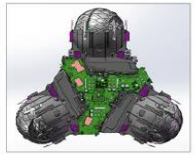
HEAD
Redesign Soft LED support
Redesign Fan support
Remove caps from back shell
1 Fan removed
Soft fan part removed
Cooler cover removed



HAND
New Hand board
New motor

SINGLE MOTOR BOARD CHANGE
Change transistor (cheaper)
Remove Ferrite

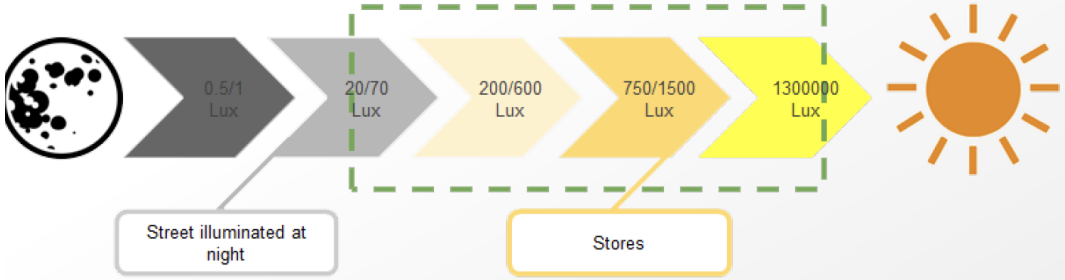
NEW QUAD MOTOR BOARD
Change transistor (cheaper)
Remove Ferrite

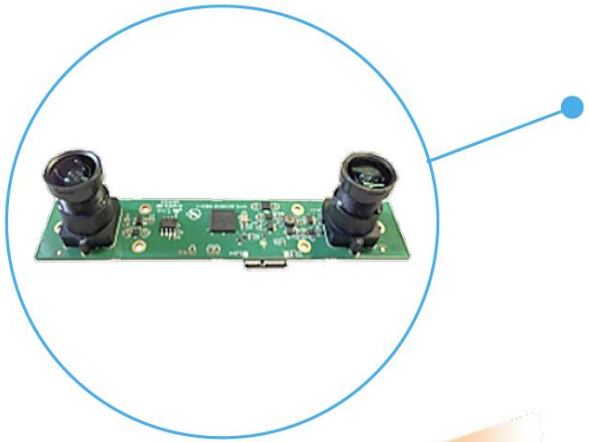


PLATFORM
New Platform
New Wheelbox motors: Fulling
Remove Wheelbox flex

⚠ Many configuration keys have been changed!

双目摄像头

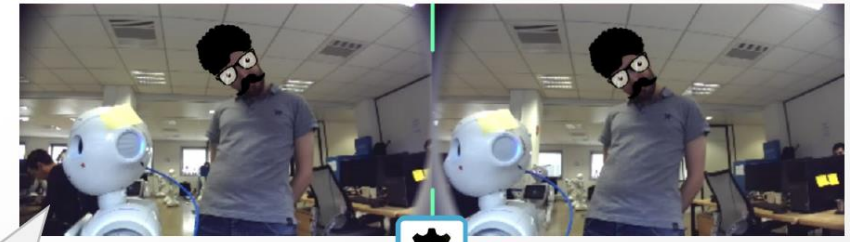




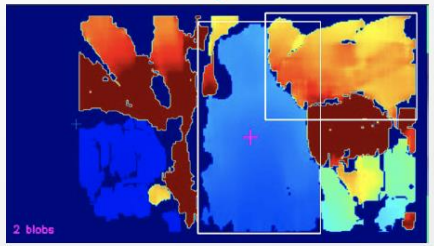
New stereovision technology

Pepper 1.8

Stereo
16:9 aspect ratio
wider FOV



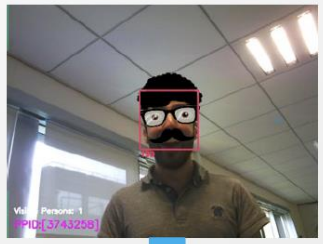
CPU

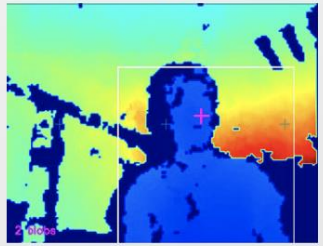


Depth data calculated by the CPU using the two images sent by the stereovision pair.

Pepper <1.8

Infrared
4:3 aspect ratio

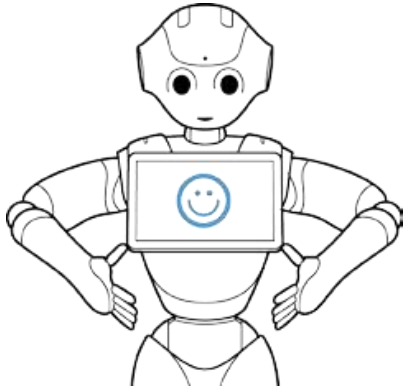




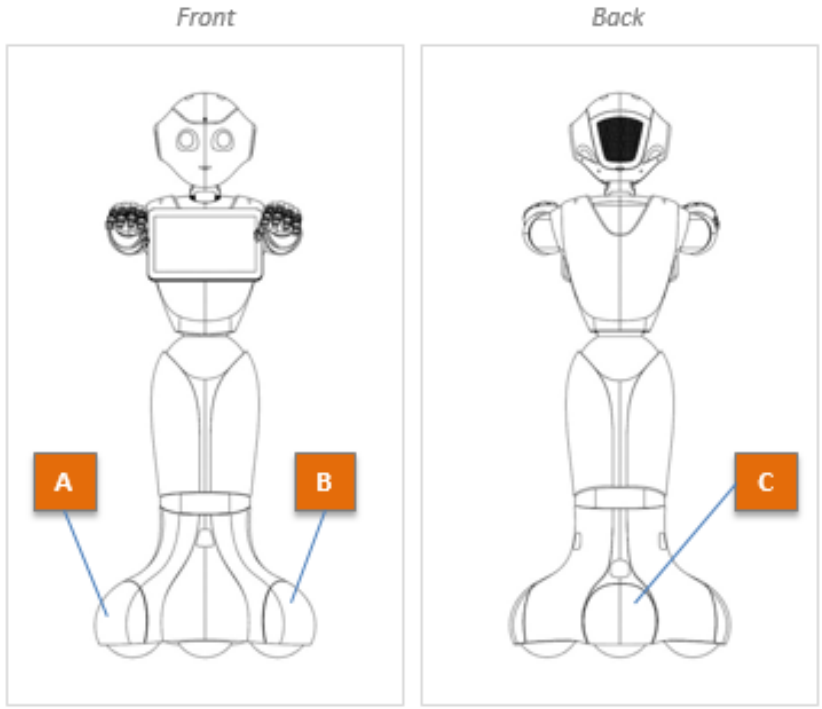
Depth data directly sent by the 3D camera

Face Detection algorithm uses Top camera image on Pepper 1.0 to 1.8a. Since Pepper 1.8, Face Detection algorithm uses Stereovision left camera.

硬件介绍

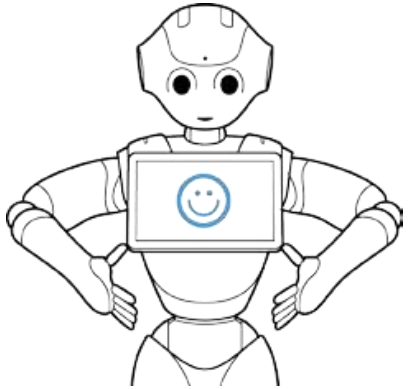


传感器

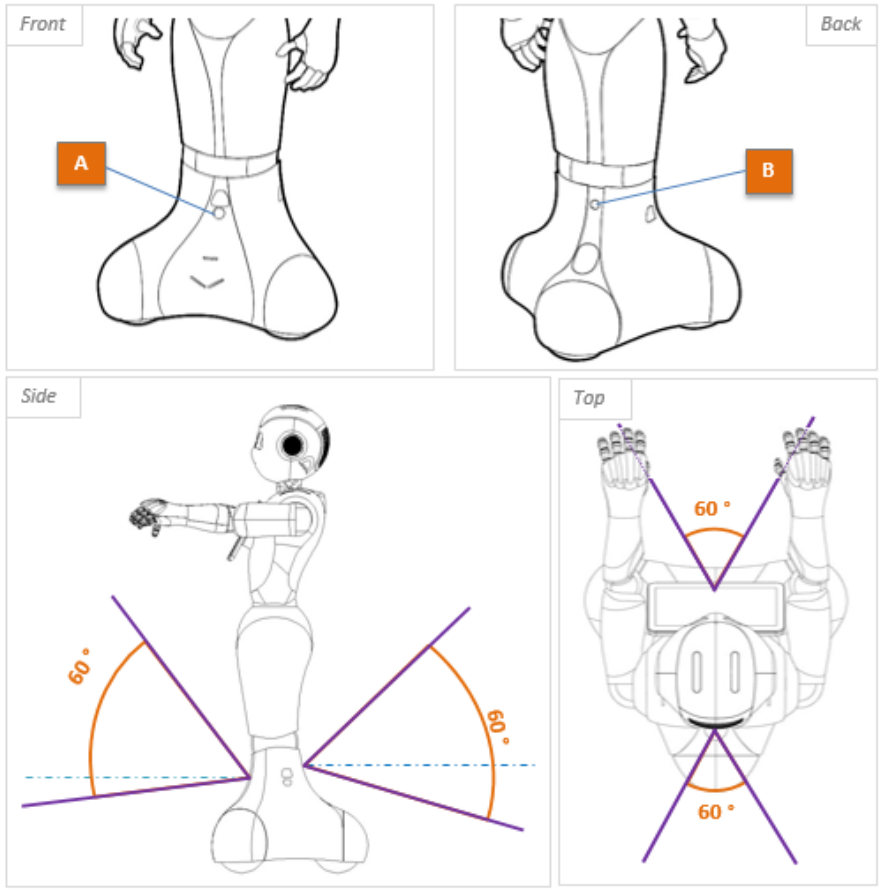


碰撞传感器：左、右、后侧 各一个 Doc

硬件介绍

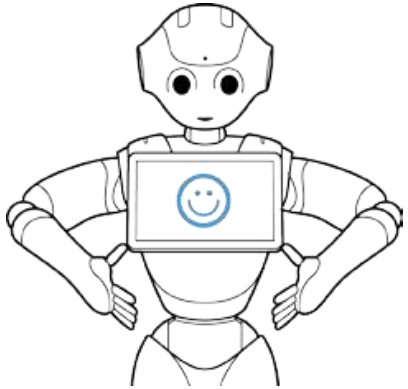


传感器

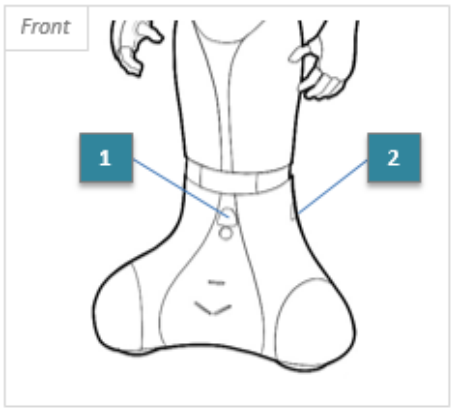


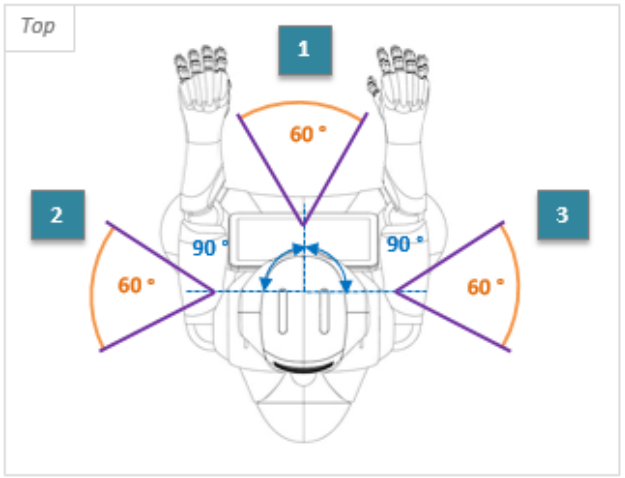
声纳：前后 各一个　　Doc

硬件介绍



传感器

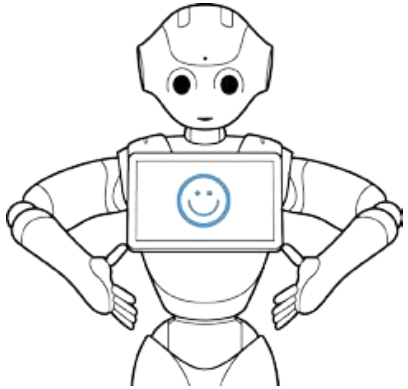




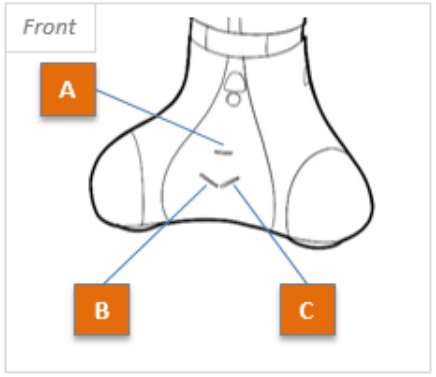
激光传感器：前、左、右 各一个

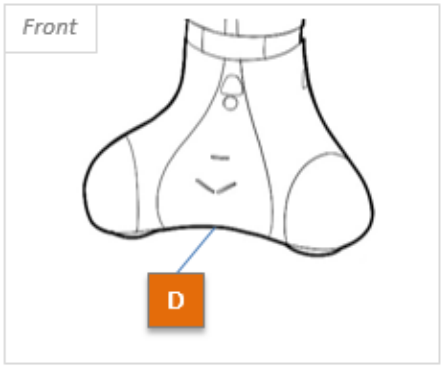
Doc

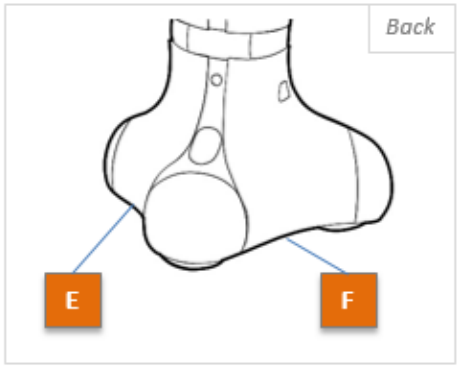
硬件介绍



传感器



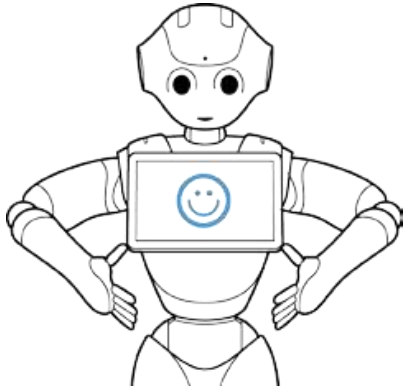




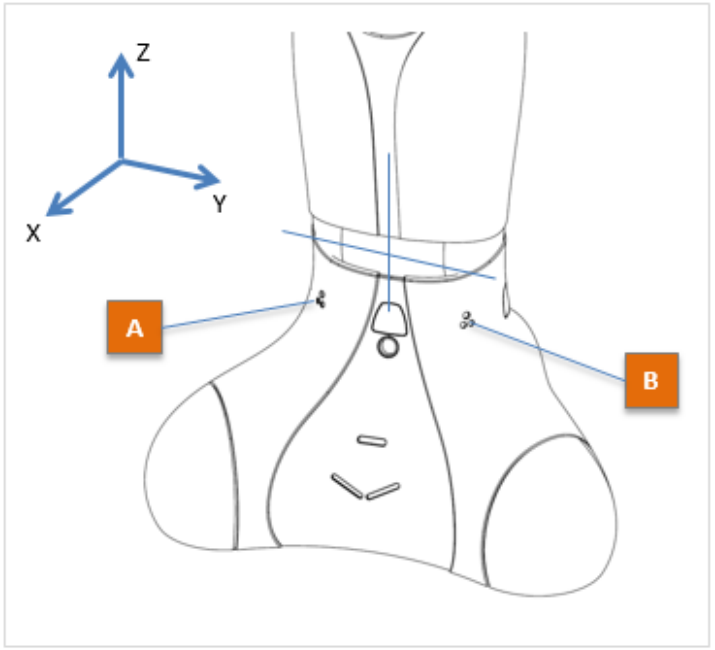
激光发射器：前、左、右 各一个

Doc

硬件介绍



传感器

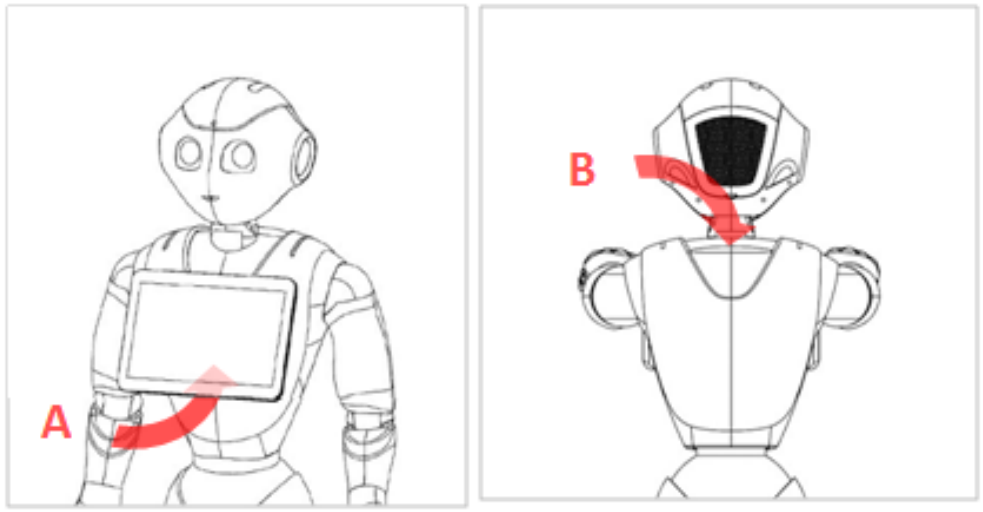


红外传感器：左、右 各一个　　Doc

硬件介绍

按钮

电源按钮：胸口

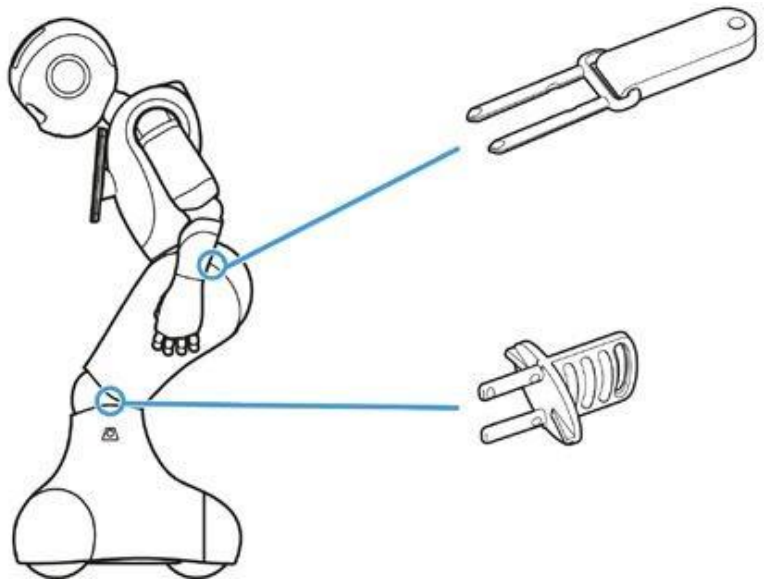


紧急制动按钮：软盖里面

硬件介绍

制动器和插销

Pepper的臀部和膝盖处都有制动器。它们可以防止Pepper摔倒



使用插针松开制动器的几种情况：

- 将Pepper放在盒子中
- 手动设置Pepper姿势时
- 搬运Pepper时

硬件介绍

机器人充电

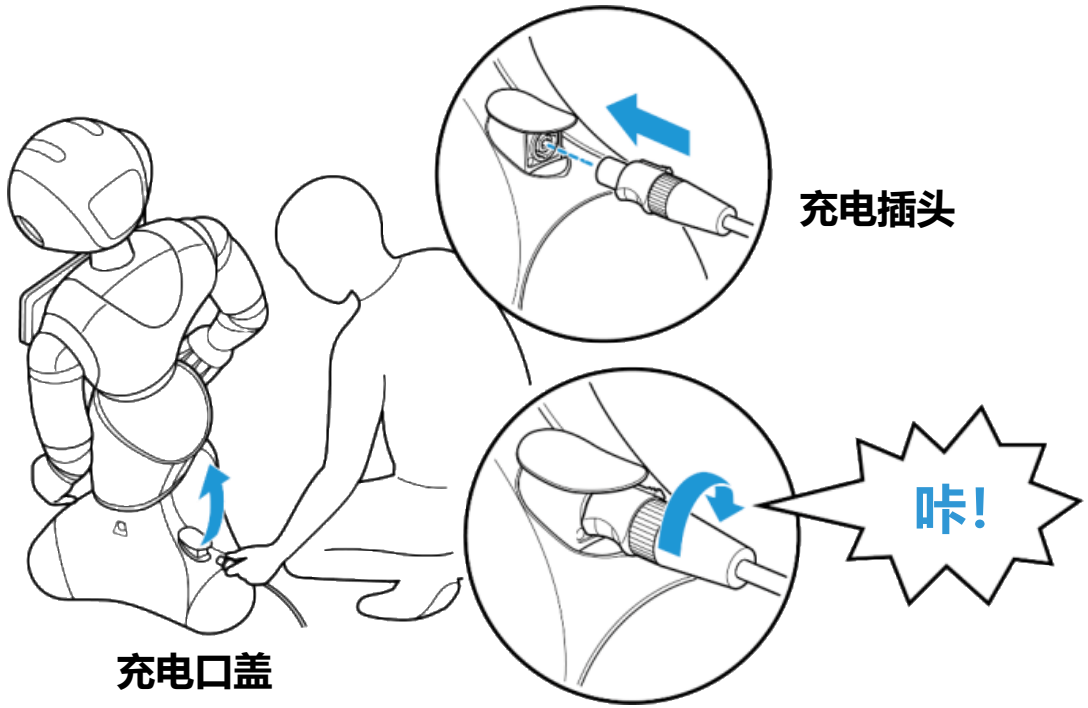
打开充电口盖给Pepper充电.

安全事项:
当充电口盖抬起时, Pepper的轮子将不会自主驱动.

技巧：如果你不想Pepper到处走，又能正常使用，可以将充电口盖抬起.

充电时间:

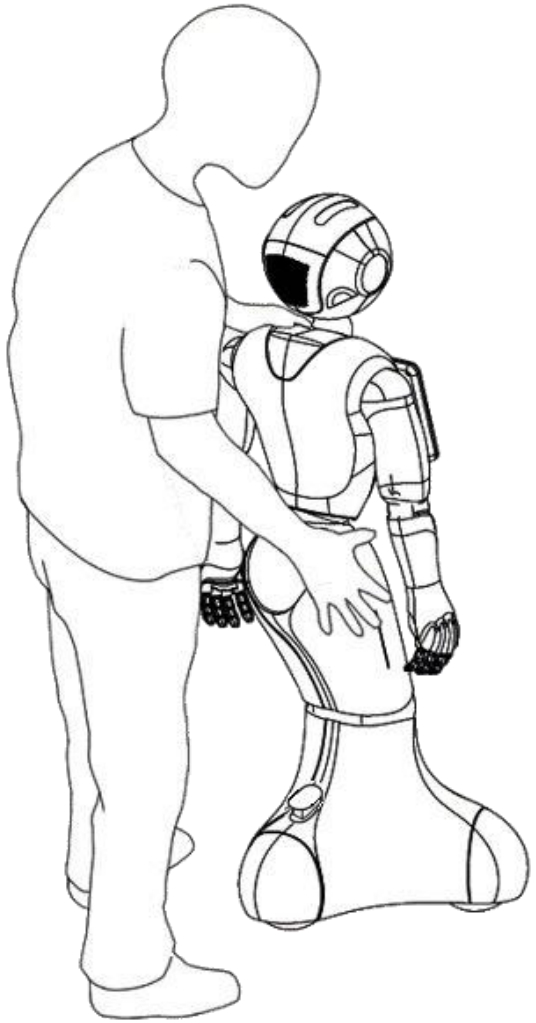
- 约3.5小时充到80%
- 约8小时充到100%



硬件介绍

移动你的机器人

- 回复到休息姿势
- 确保充电口盖已抬起
- 抓紧机器人
  - 一只手放在Pepper的肩上
  - 一只手放在Pepper的臀部
- 小心的移动Pepper



硬件介绍

胸部按钮

胸部按钮的多种用途:

- 当Pepper关机状态时:
  - 按一下: 启动Pepper
  - 长按(5秒): 检查微控制器并启动Pepper

- 当Pepper开机状态时:
  - 按一下: 获得当前的状态 和 警告/错误信息（如有）
  - 连续按压两次: 休眠 / 唤醒 （交替）
  - 按压持续3秒: 将Pepper关机（伴随关机音）
  - 按压持续8秒: 将Pepper强制关机（Pepper立即断电，注意当前姿势，防止摔倒）

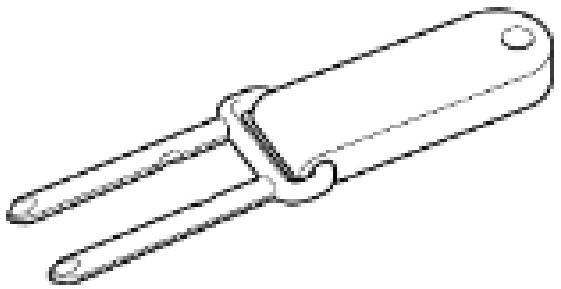
硬件介绍

网络连接方式

- 无线网连接

- 以太网连接.

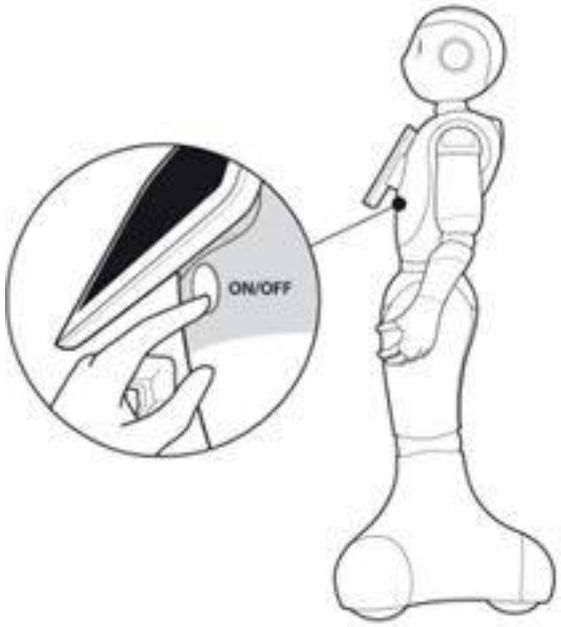
使用(臀部的)插销插入头部后盖下方的两个孔位，打开头部后盖就可以看到以太网接口。



启动并配置Pepper

启动并配置Pepper

按平板下方的胸部按钮一下来开机



在开机过程中:

- 肩部的LED灯缓慢的闪烁
- 耳朵上的LED灯作为开机的进度条

整个启动过程大约需要1.5分钟

当Pepper说:

“OGNAK GNOUK”

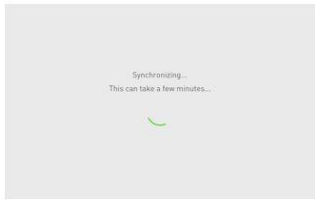
这表示Pepper已经开机!

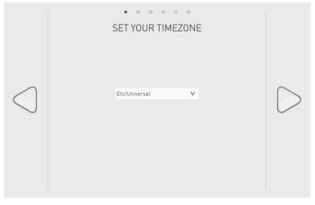
开机完成后，Pepper将会根据预装APP或设置的不同，出现不同的运行状态。如果Pepper未安装任何APP并第一次开机时，平板上会出现初始化设置页面。此时需要用户完成各项设置之后，Pepper才能启动完毕。

启动并配置Pepper　NAOqi2.5

当你第一次启动机器人的时候，一个向导会自动运行。

向导会指引你完成每一步的配置。（你需要一个SBR账号来绑定你的Pepper，账号申请。）

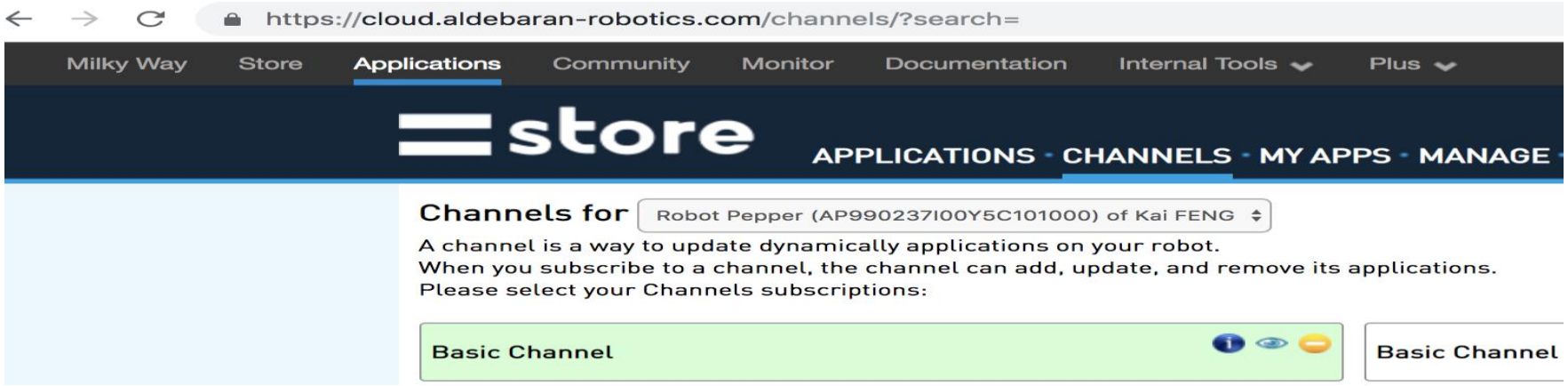




基本交互

在你的机器人上, 我们 (有可能) 已经安装了一系列特殊的软件来让你试着跟Pepper进行交互:

- 当Pepper看到有人靠近他时， “The Dialog” 这个App会自动启动 。
  这个App会触发一些“对话主题”，使你能够跟机器人交流。
- Basic channel : 这是一系列“对话主题” （内容库）



如果没有从在线平台上下载过这些APP，Pepper将不具备这个功能。

基本交互

基本交互

在你的机器人上, 我们 (有可能) 已经安装了一系列特殊的软件来让你试着跟Pepper进行交互:

- 当Pepper看到有人靠近他时， “The Dialog” 这个App会自动启动 。
  这个App会触发一些“对话主题”，使你能够跟机器人交流。

- Basic channel : 这是一系列“对话主题” （内容库）

如果没有从在线平台上下载过这些APP，Pepper将不具备这个功能。

基本感知 (Basic Awareness)

当你的机器人启动后，她会站起来并且开始寻找附近的人
这就是所谓的基本感知
Pepper 现在能够对基本刺激作出反应:

- 声音
- 移动物体
- 触觉
- 人

目的就是为了能够找到人，并且和人进行交互!

NAOqi2.5系统

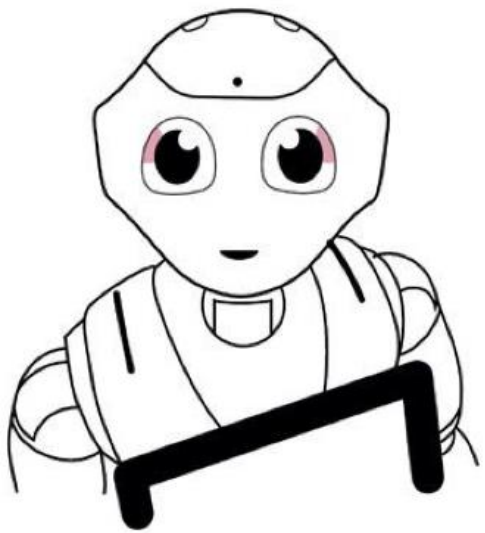
SoftBank Robotics

和Pepper对话

眼睛可以告诉你...

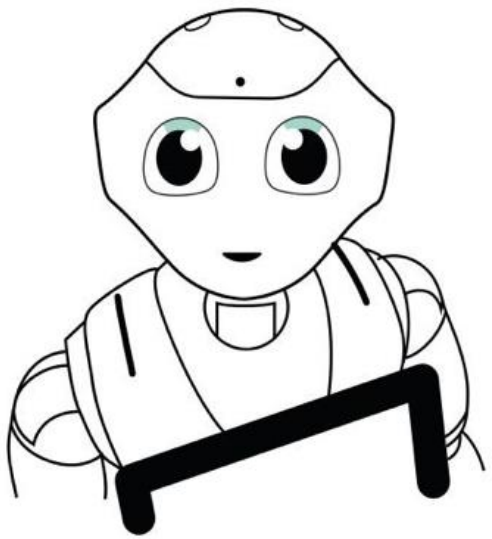
粉红色:

跟踪到一个人



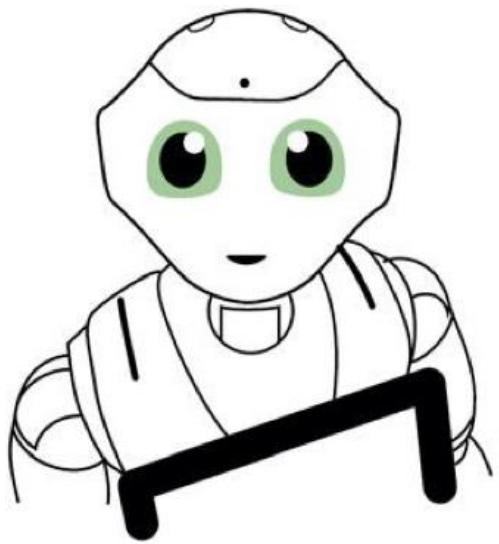
蓝色 (旋转):

正在听



绿色:

思考中



NAOqi2.9系统 和Pepper对话

肩部的LED和Pad上的SpeechBar用来提示是否处于听的状态以及听到的内容

NAOqi2.9系统 和Pepper对话

Hearing Human Voice

Not Understood

Understood

三个状态分别代表机器人：正在识别、没听懂、识别 成功结果

和Pepper对话

交互区域

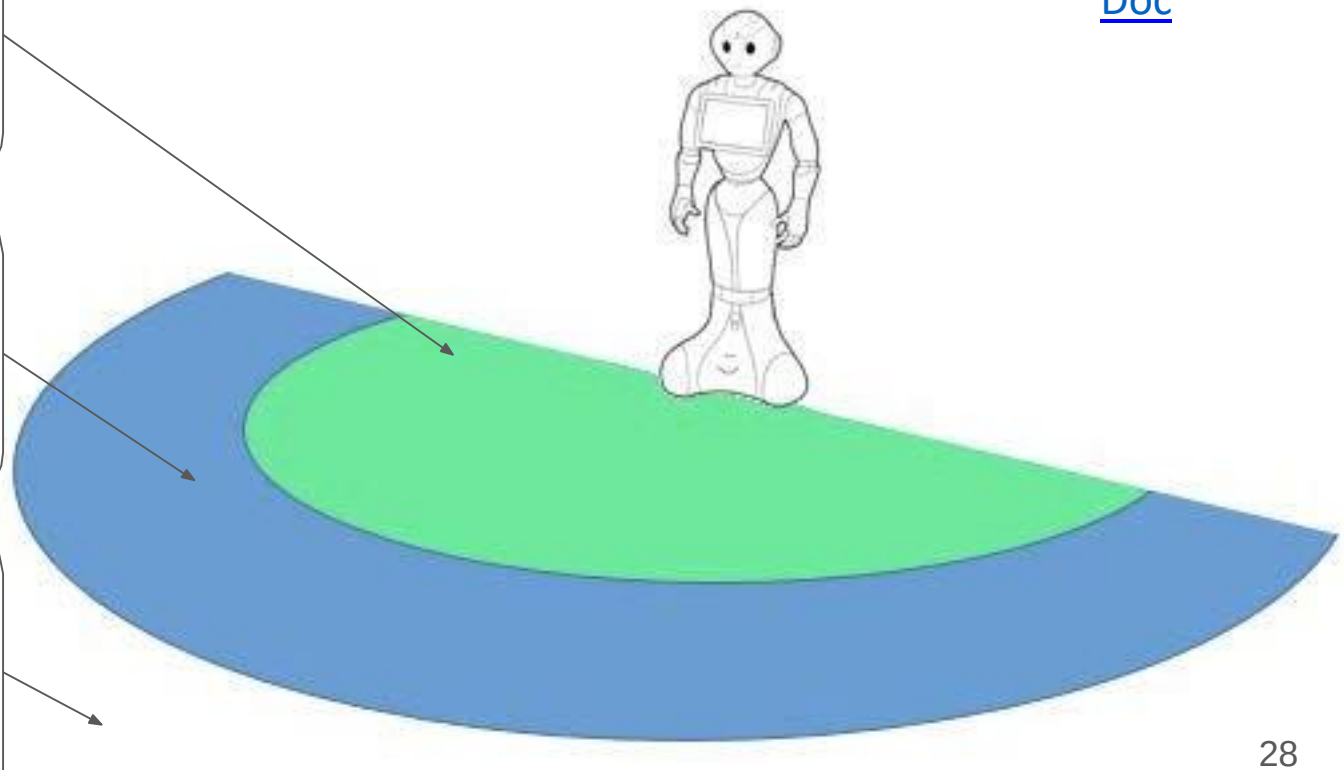
各区域的距离可以进行设定。
默认距离：
区域1: 0 - 1.5m
区域2: 1.5 - 2.5m
区域3: >2.5m

Doc

区域 1 (Zone 1)
你离Pepper够近了，可以跟他对话了。

区域 2 (Zone 2)
你离得有点远了，但你可以听到她叫你。

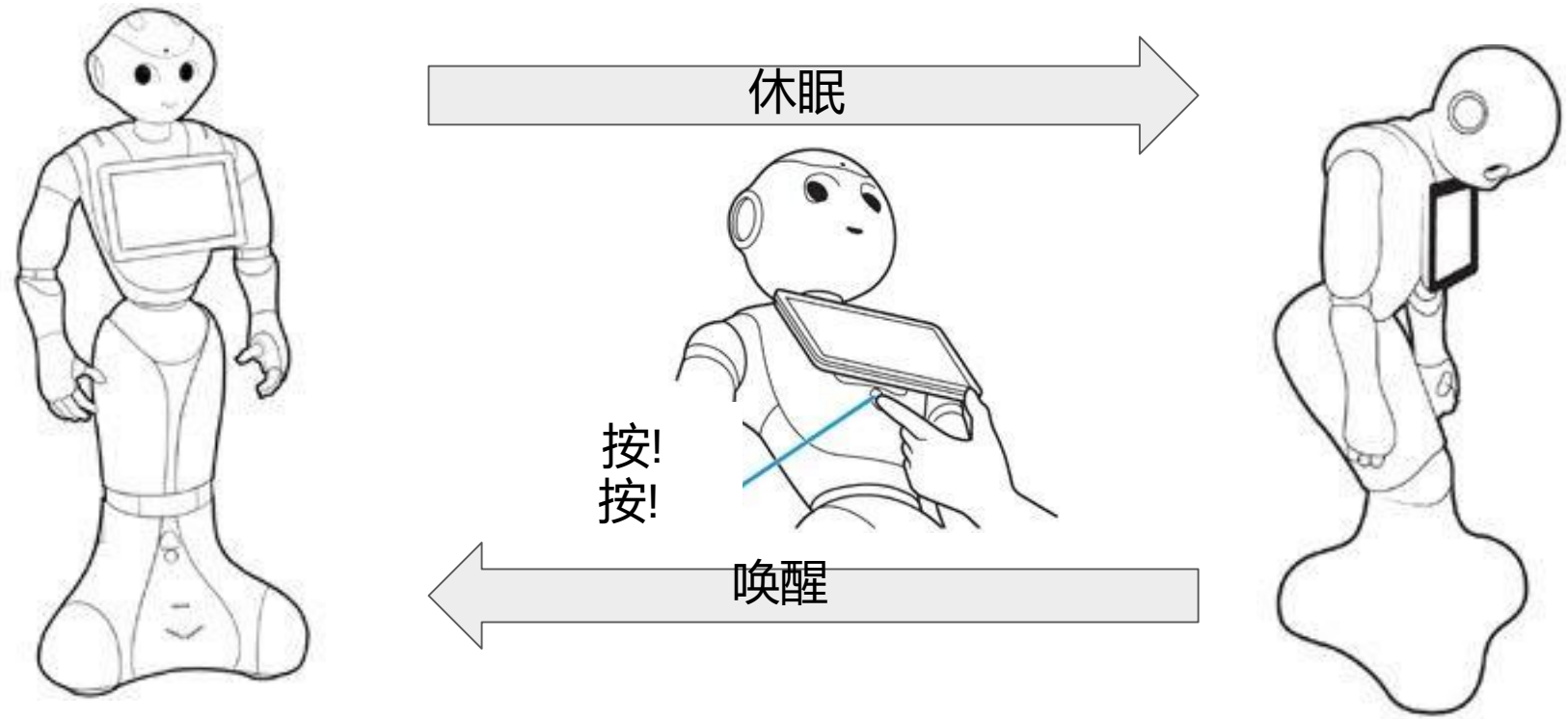
区域 3 (Zone 3)
你离得太远了，Pepper能看见你，但你们互相之间听不见。



休眠 / 唤醒

休眠 / 唤醒

→ 按两次胸部按钮



回到设置

Pepper 设置 NAOqi 2.5

Pepper启动之后, 你可以使用机器人网页进行设置.

1) 按一下胸部按钮. Pepper给你他的IP地址.
2) 在浏览器中输入IP地址.

3) 你会被要求提供密码(用户名是 nao, 默认密码是 nao), 然后你就可以访问设置页面了。

Pepper 设置 NAOqi2.5

我的机器人　　显示并设置机器人的基本设置。

网络设置　　设置网络连接

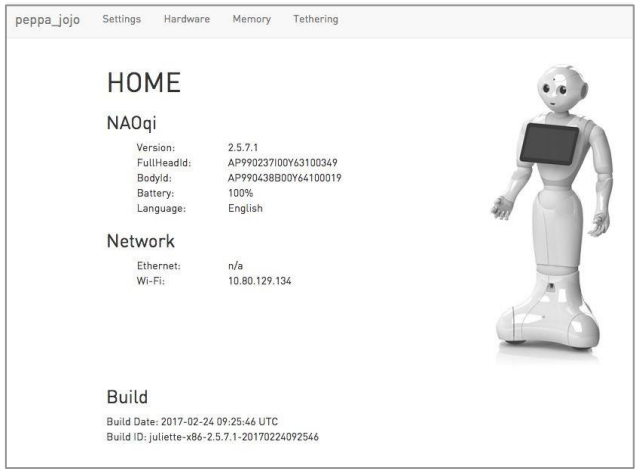
更新　　获取应用更新，设置和管理您的云帐户。

高级设置　　显示并设置机器人的高级设置。

Pepper 设置 NAOqi 2.5

“advanced” 页面可以看到更多细节。

- 在浏览器输入http://机器人的IP地址/advanced



- 在这里你可以找到一些重要的信息，比如序列号。(头部 ID 和 身体 ID).
- 不同的页面使您能够访问有关硬件、软件的详细信息。

关闭你的机器人

关闭你的机器人

- 长按胸部按钮3秒, 直到机器人说“GNUK GNUK”

- 机器人会进入“休眠”姿势

- 这个过大概会花30秒

软件
基于 NAOqi 2.5

软件 – 在机器人上的...

两个设备: 头部和平板电脑, 通过以太网连接(USB)

- 头部:
  - 运行Linux系统
  - 执行程序
  - 存储数据
  - 连接到互联网
  - 为平板电脑提供应用程序网页
- 平板电脑:
  - 运行Android系统
  - 在浏览器中显示应用程序中网页、图像、视频
  - 连接到互联网

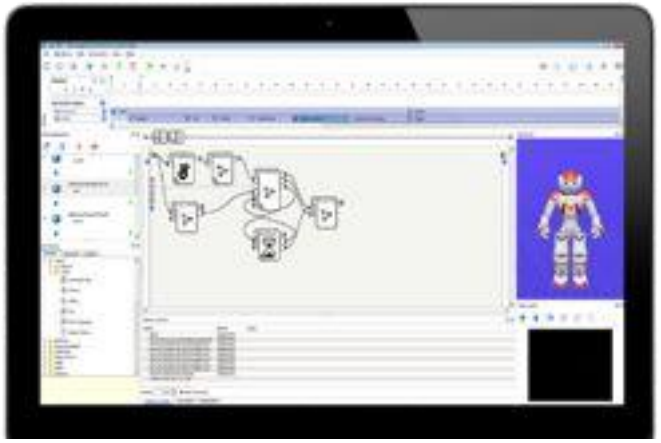
软件 – 在机器人上的...

操作系统： NAOqi

NAOqi 和服务(Service)一起工作:

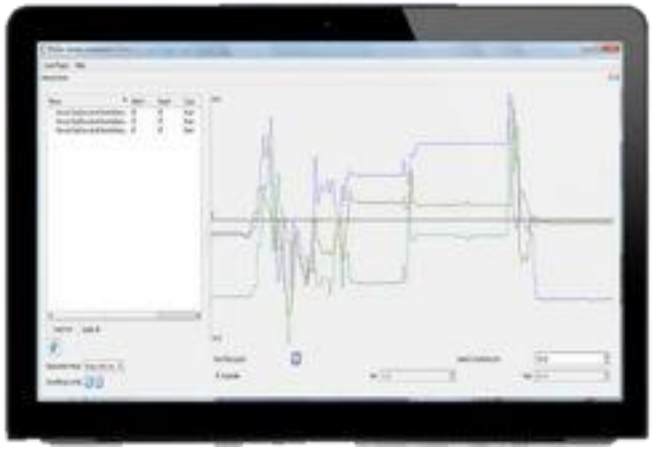
（应用程序(APP)将调用NAOqi的接口来让机器人工作。）

软件 – 在电脑上的（开发工具）…



Choregraphe

简易可视化编程工具



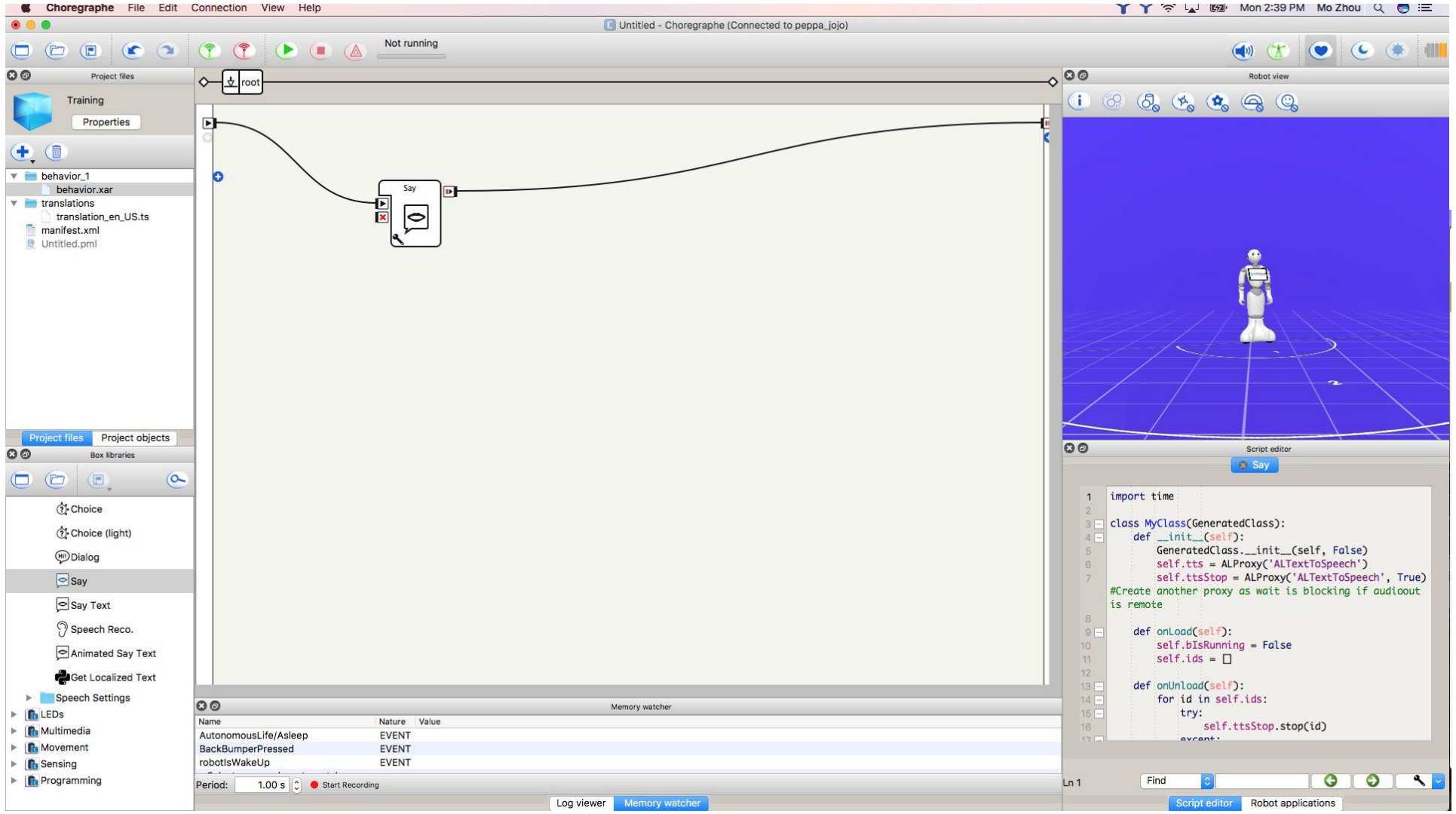
Monitor

监视机器人内部传感器数据、日志...

Software Development Kit

提供给C++ 或者 Python 的综合API

软件 - Choregraphe



软件 - Choregraphe

轻松创建动画、简单行为和对话。

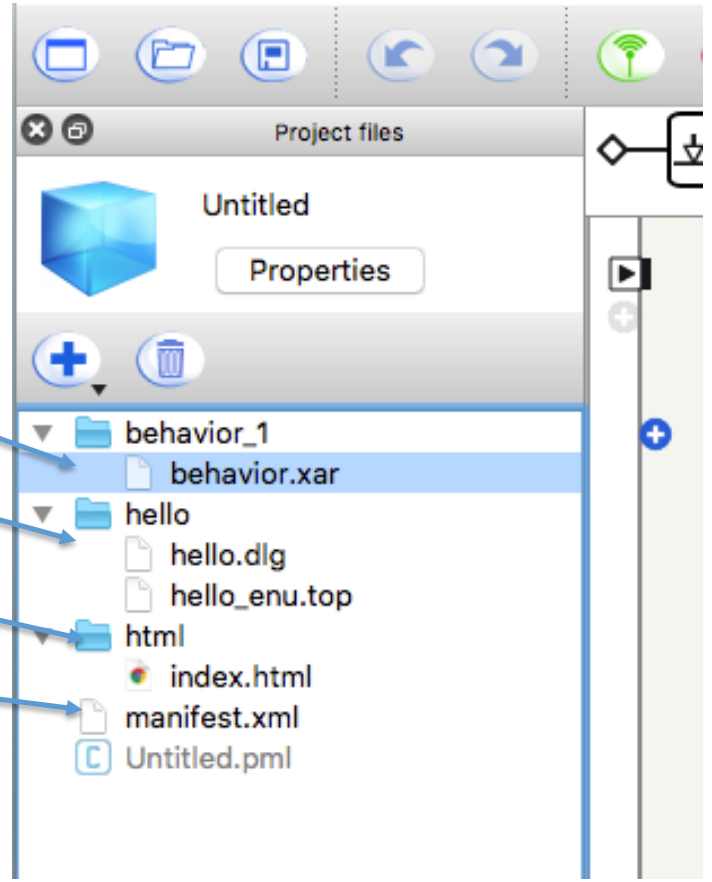
在一个模拟机器人上、或者直接在一个真实的机器人上运行程序，同时可以监控和控制你的机器人。

打包并部署（安装）应用程序 至 机器人本地。

软件 - Choregraphe

一个应用程序(APP)包含什么?

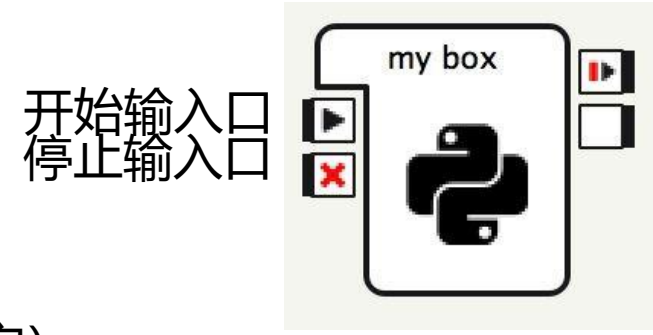
- 行为 (.xar):　Pepper能做什么
- 对话主题 (.dlg / .top):　Pepper可以谈论什么
- 其他资源(媒体, 脚本, 网页…)
- 属性 (图标, 名字, …)

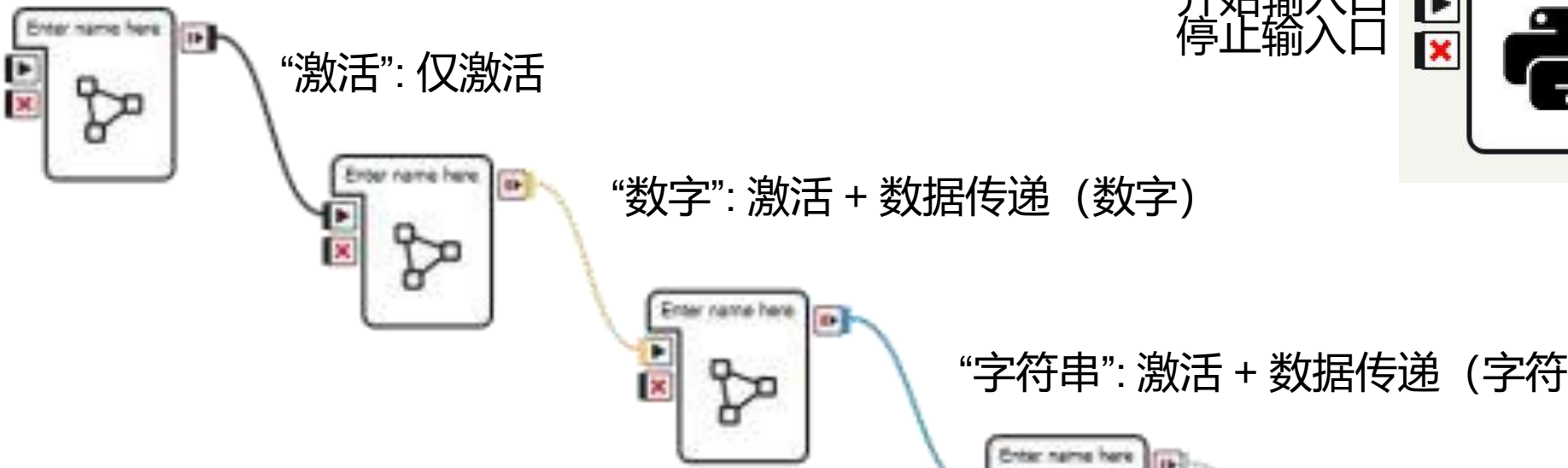


软件 - Choregraphe

指令盒 (Choregraphe编程的基本组件)

指令盒
输入输出口





“激活”: 仅激活

“数字”: 激活 + 数据传递（数字）

“字符串”: 激活 + 数据传递（字符串）

“动态”: 激活 + 数据传递（动态数据）

指令盒间的流程：

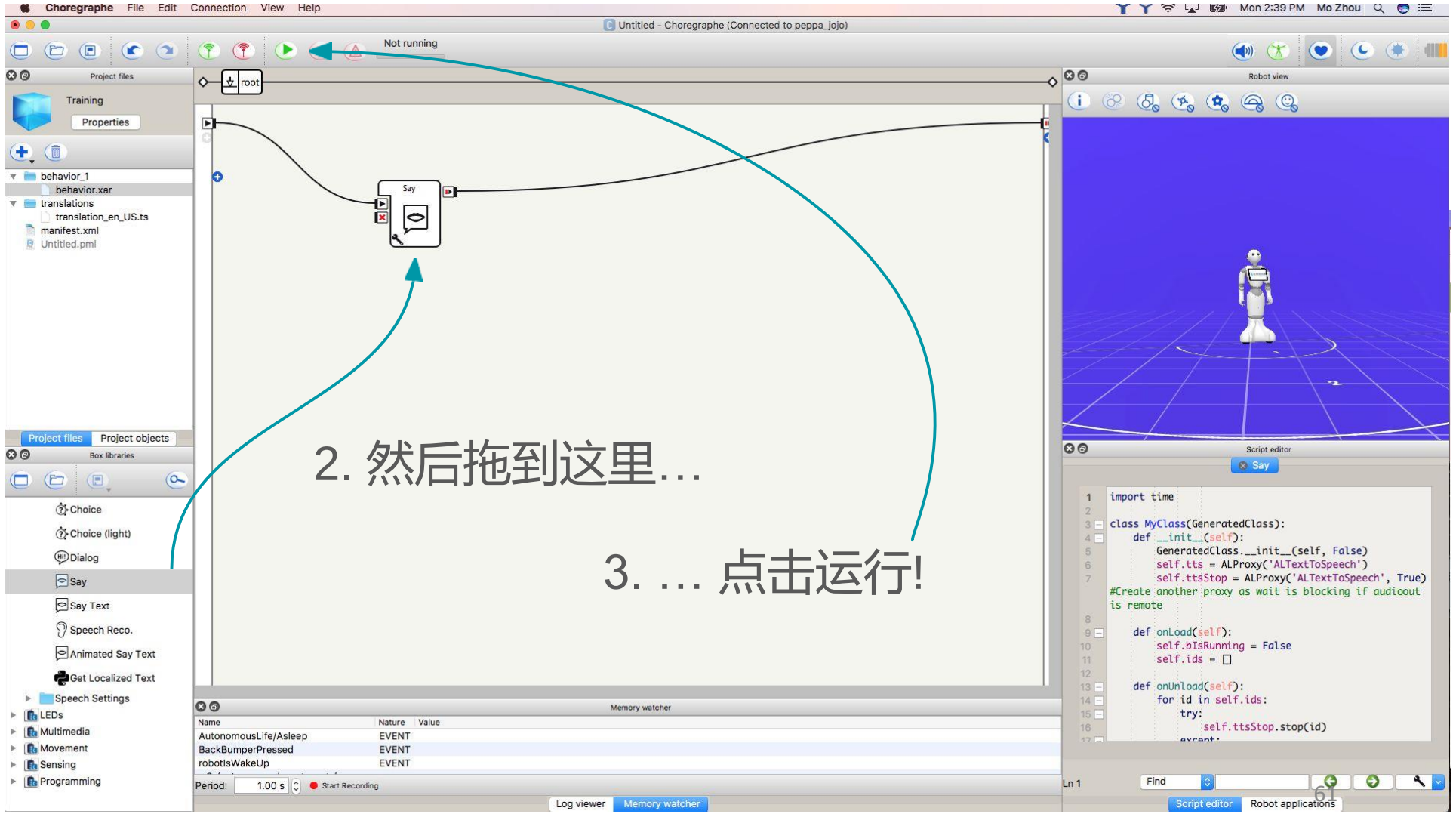
可以触发，同时可以传递数据

软件 - Choregraphe

试试编写
第一个App…

Hello world!

1. 找到 “Say” →



软件 - Choregraphe

现在，我们看看机器人的语音…

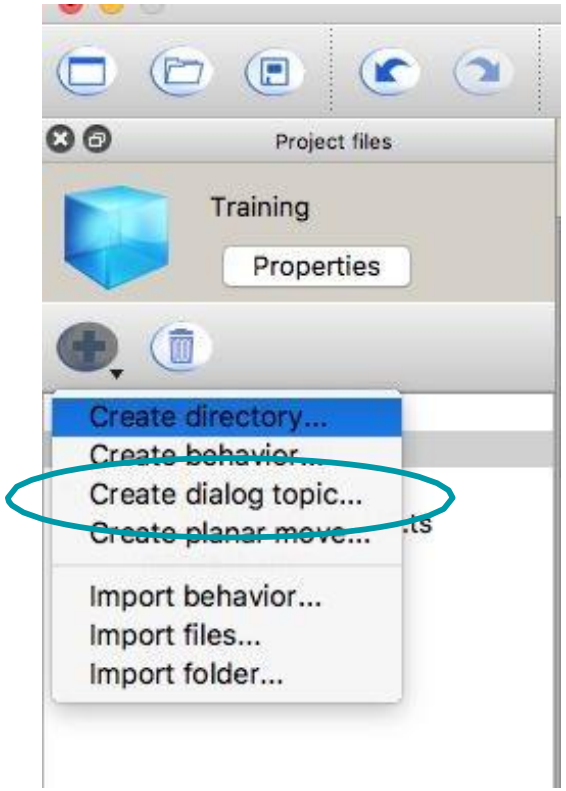
1) 点击 + 选择 “Create dialog topic…”
2) 输入一个topic名字(无空格), 选择语言为英语
3) 生成.dlg文件
4) 拖.dig到右边的编程区域中
5) 编辑 `.top` 文件…（双击.top文件）
6) 学习QiChat规则，编写对话:

   基本规则：“ *u:(需要识别的话) 机器人的回答*”

   示例:

```
u:( hello ) Hello sir!
u:( how are you ) I’m good, thanks!
```

7) 启动behavior!



软件 - Choregraphe

对话中还可以：使用事件、函数…

u:(what user says) what robot says

u:(e:event) what robot says

u:(…) $output_of_the_box=123

u:(…) $memory_event=123

u:(…) ^run(application_id / behavior_id)

软件 - Choregraphe

关于对话…

更好的语法结构: “concept” = 一组同义语

```
concept:(hello) [hello hi hey]
concept:(good) {pretty very} [well good]
```

文件头的concept可以:
方便阅读，及避免正文太冗长

```
u:(~hello) hi Jack ! ^gotoReactivate(askhowareyou)
```

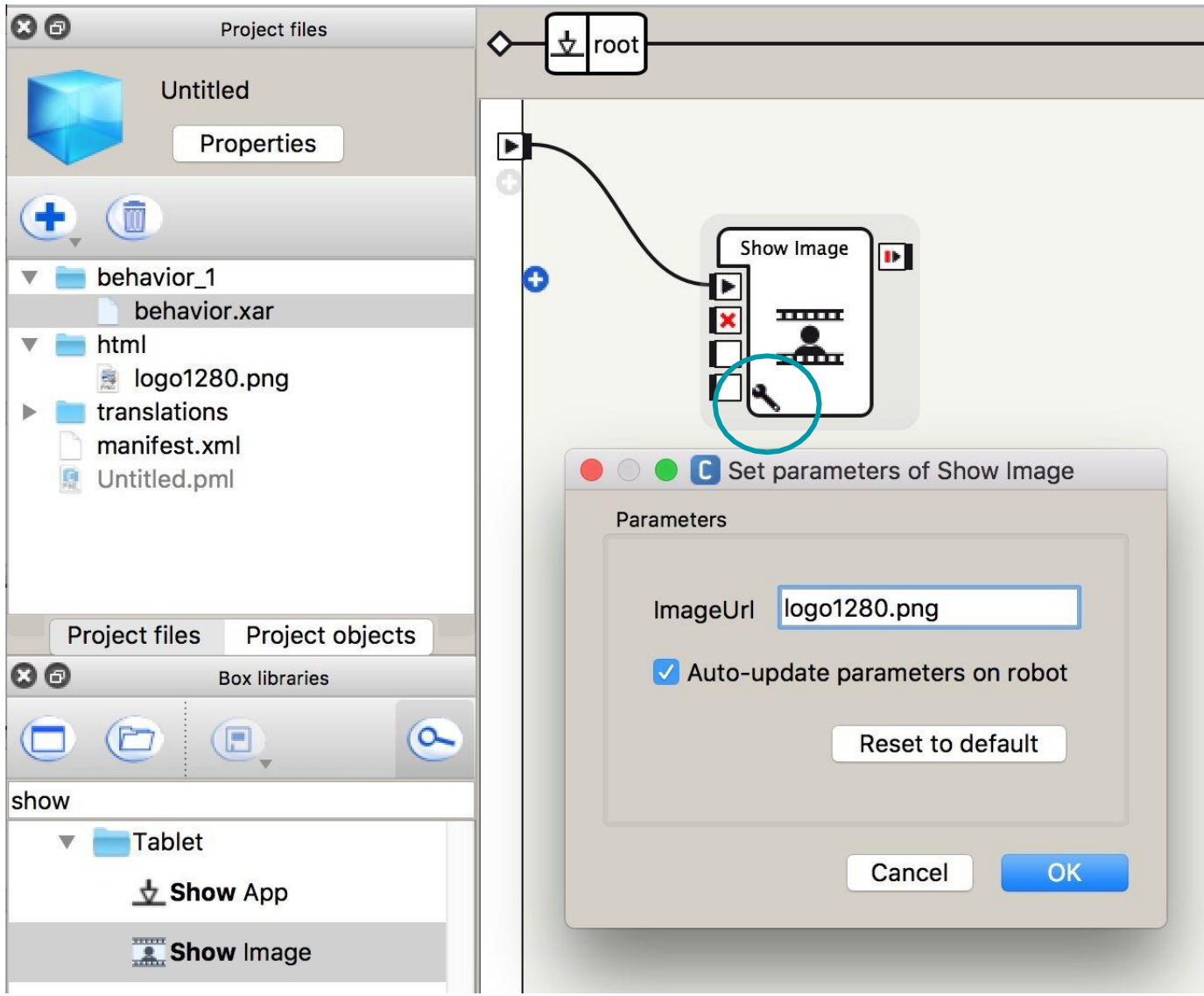
```
proposal: %askhowareyou How are you?
u1:(bad) oh no! ^gotoReactivate(...)
u1:(~good) I’m happy to hear that!
```

U1 和 proposal一起使用

软件 - Choregraphe

在平板上显示图片…

1) 建立新文件夹 “html”
2) 放一张图片
3) 拖拽“Show image”盒子
4) 连线
5) 填写参数框
6) 运行!



软件 - Choregraphe

编写自己的指令盒…

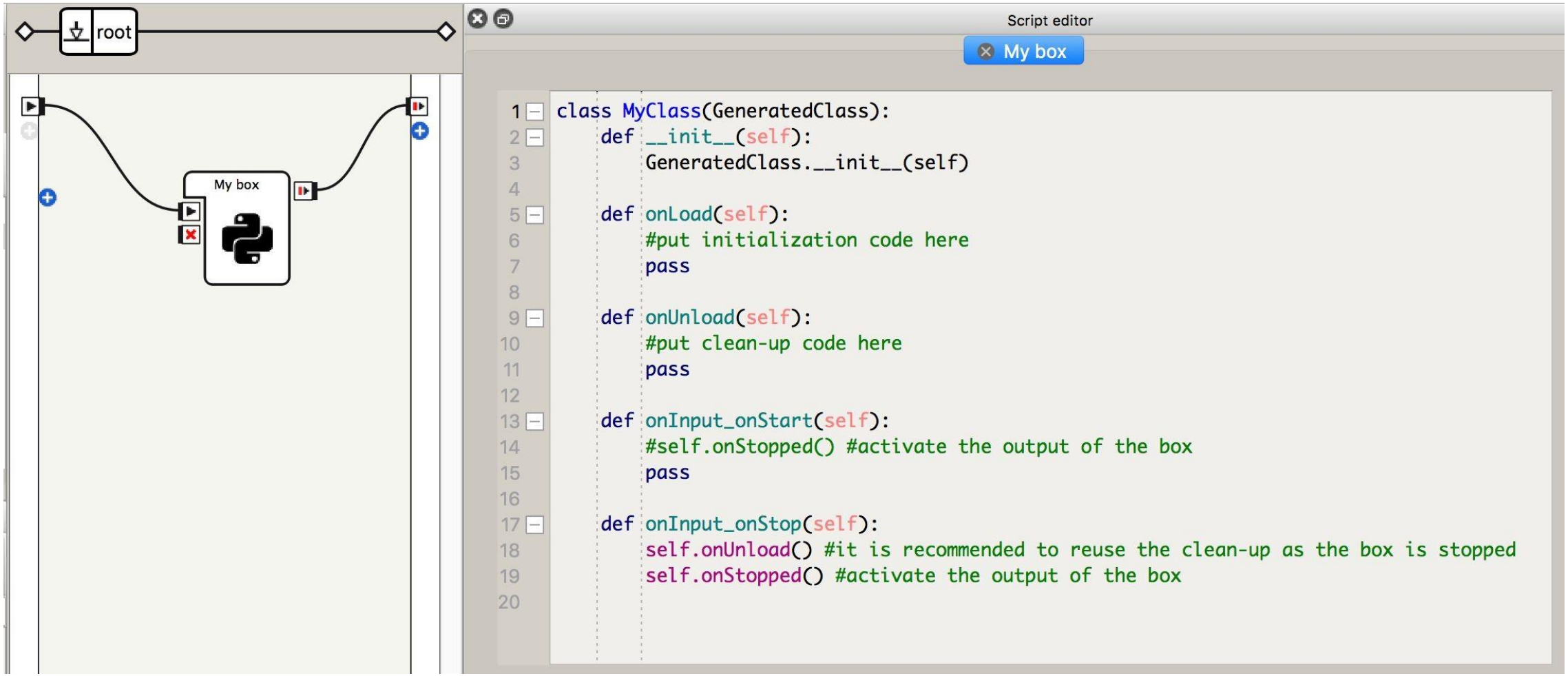
右击编程区域空白处,

选择”添加新盒子”…

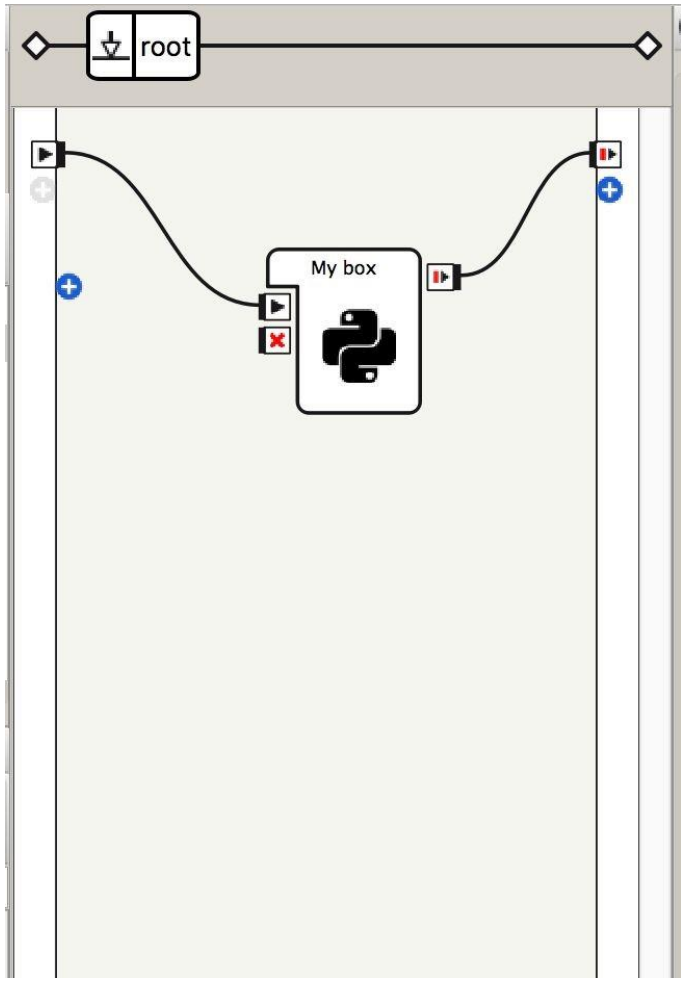
Python !

给名字然后双击盒子，撰写代码.

软件 - Choregraphe



软件 - Choregraphe



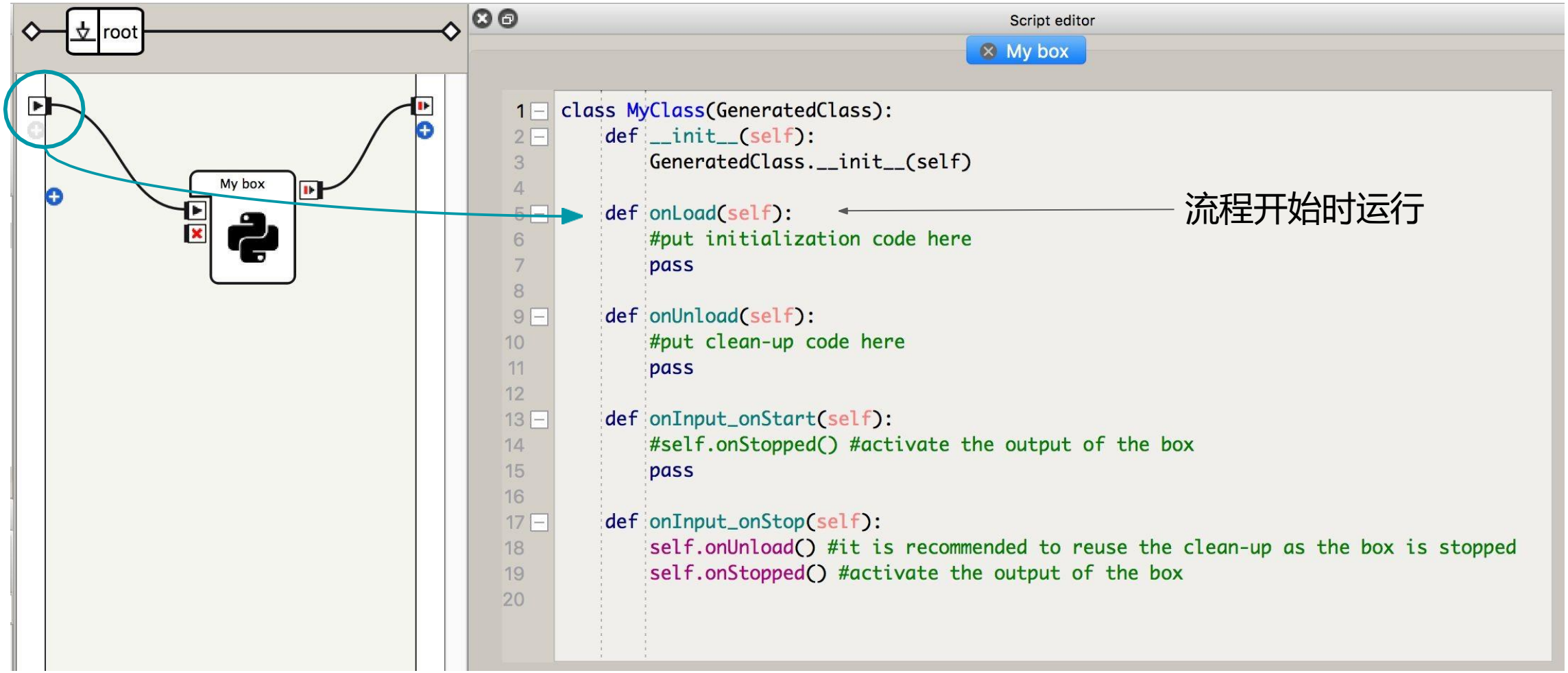
Script editor

My box

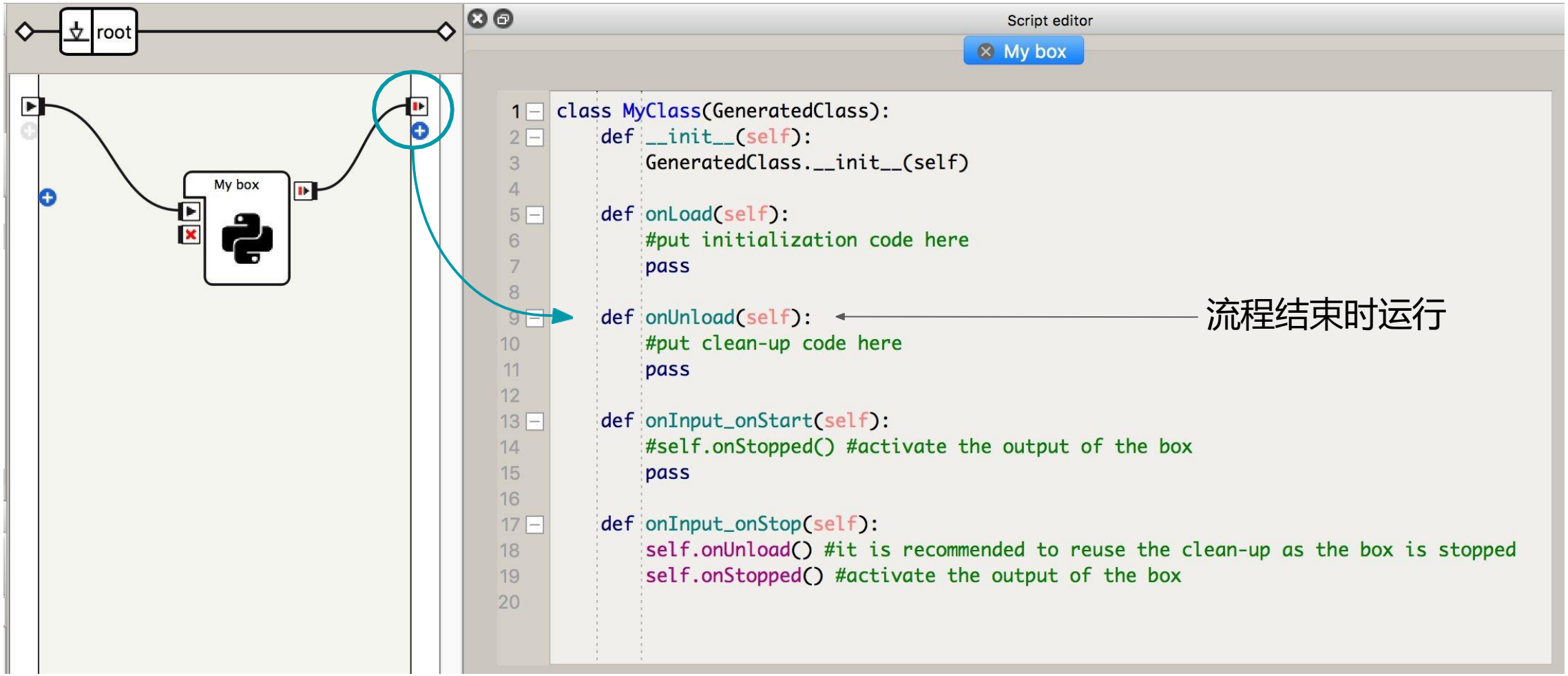
```
1  class MyClass(GeneratedClass):
2      def __init__(self):
3          GeneratedClass.__init__(self)
4
5      def onLoad(self):
6          #put initialization code here
7          pass
8
9      def onUnload(self):
10         #put clean-up code here
11         pass
12
13     def onInput_onStart(self):
14         #self.onStopped() #activate the output of the box
15         pass
16
17     def onInput_onStop(self):
18         self.onUnload() #it is recommended to reuse the clean-up as the box is stopped
19         self.onStopped() #activate the output of the box
20
```

APP启动时即会运行

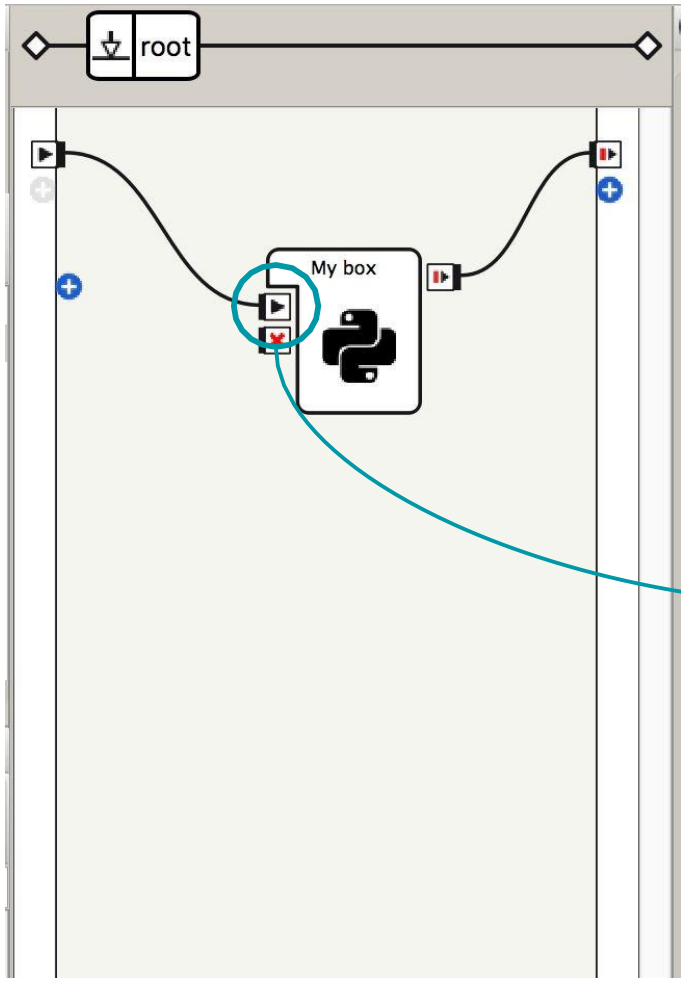
软件 - Choregraphe



软件 - Choregraphe



软件 - Choregraphe



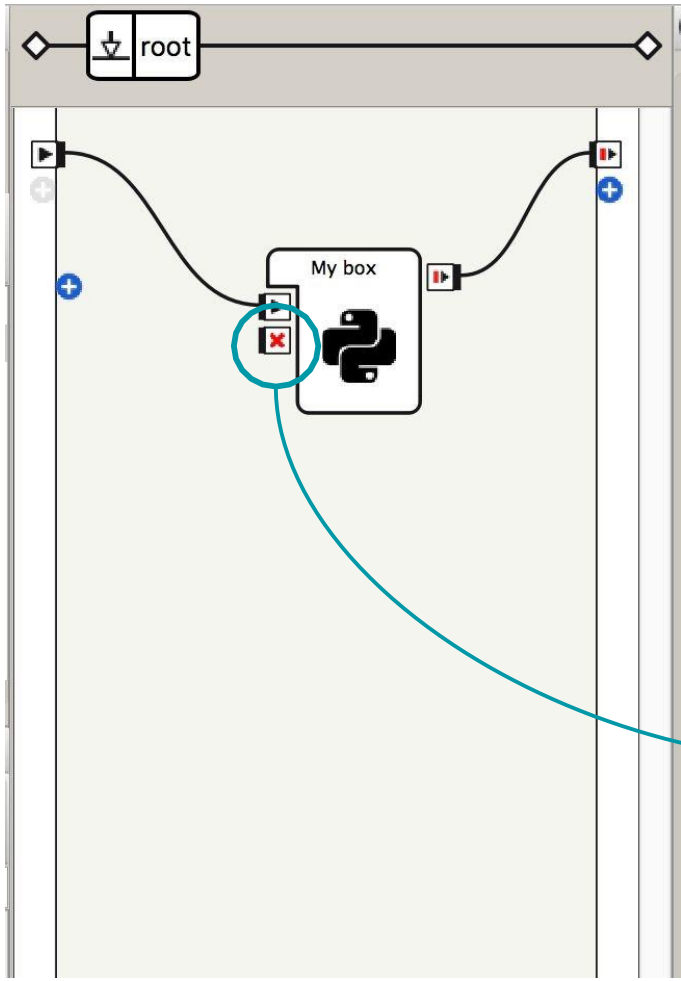
Script editor

My box

```
1  class MyClass(GeneratedClass):
2      def __init__(self):
3          GeneratedClass.__init__(self)
4
5      def onLoad(self):
6          #put initialization code here
7          pass
8
9      def onUnload(self):
10         #put clean-up code here
11         pass
12
13     def onInput_onStart(self):
14         #self.onStopped() #activate the output of the box
15         pass
16
17     def onInput_onStop(self):
18         self.onUnload() #it is recommended to reuse the clean-up as the box is stopped
19         self.onStopped() #activate the output of the box
20
```

“onStart” 入口被触发时运行

软件 - Choregraphe



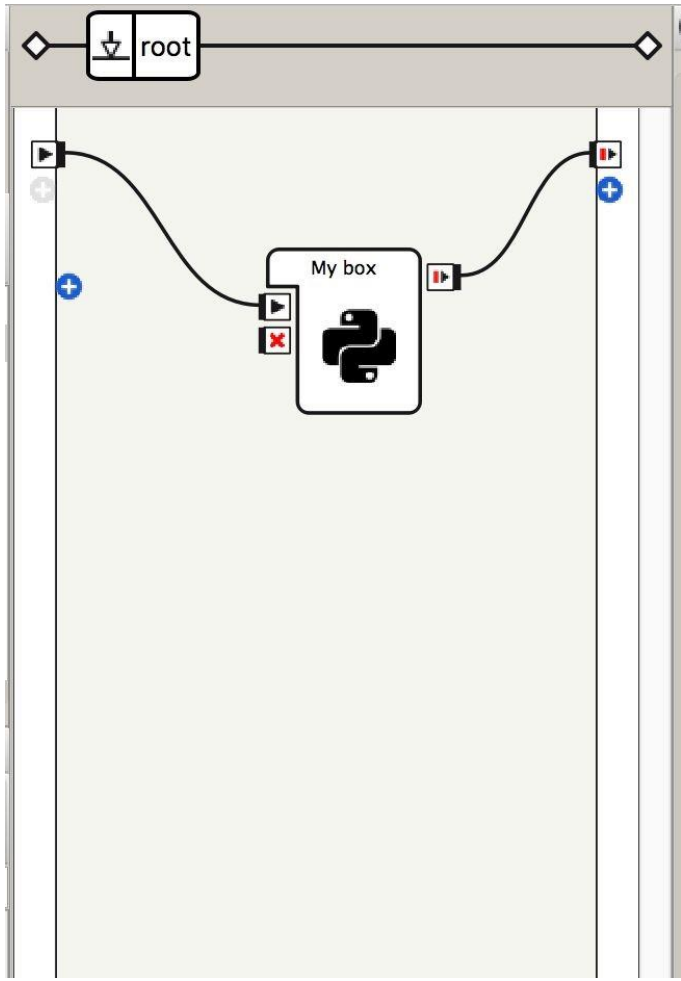
Script editor

My box

```
1  class MyClass(GeneratedClass):
2      def __init__(self):
3          GeneratedClass.__init__(self)
4
5      def onLoad(self):
6          #put initialization code here
7          pass
8
9      def onUnload(self):
10         #put clean-up code here
11         pass
12
13     def onInput_onStart(self):
14         #self.onStopped() #activate the output of the box
15         pass
16
17     def onInput_onStop(self):
18         self.onUnload() #it is recommended to reuse the clean-up as the box is stopped
19         self.onStopped() #activate the output of the box
20
```

“onStop” 入口被触发时运行

软件 - Choregraphe



Script editor

My box

```
1  class MyClass(GeneratedClass):
2      def __init__(self):
3          GeneratedClass.__init__(self)
4
5      def onLoad(self):
6          #put initialization code here
7          pass
8
9      def onUnload(self):
10         #put clean-up code here
11         pass
12
13     def onInput_onStart(self):
14         #self.onStopped() #activate the output of the box
15         pass
16
17     def onInput_onStop(self):
18         self.onUnload() #it is recommended to reuse the clean-up as the box is stopped
19         self.onStopped() #activate the output of the box
20
```

APP启动时即会运行

流程开始时运行

流程结束时运行

“onStart” 入口被触发时运行

“onStop” 入口被触发时运行

软件 - Choregraphe

指令盒

```
def onInput_onStart(self):
    self.logger.info("Box is running!")
    #~ self.onStopped() #~ activate output of the box
    pass
```

出现在“日志查看器”

激活输出端

软件 - Choregraphe

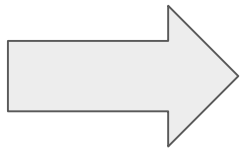
指令盒

```
def onInput_onStart(self):
    self.logger.info("Box is running!")

    session = self.session()

    tts = session.service("ALTextToSpeech")

    tts.say("Hello from my box!")

    self.onStopped() #~ activate output of the box
    pass
```

创建一个 session
(`self.session()` 是Choregraphe提供的一个工具)

获取一个“服务”

调用“服务”中的一个api接口



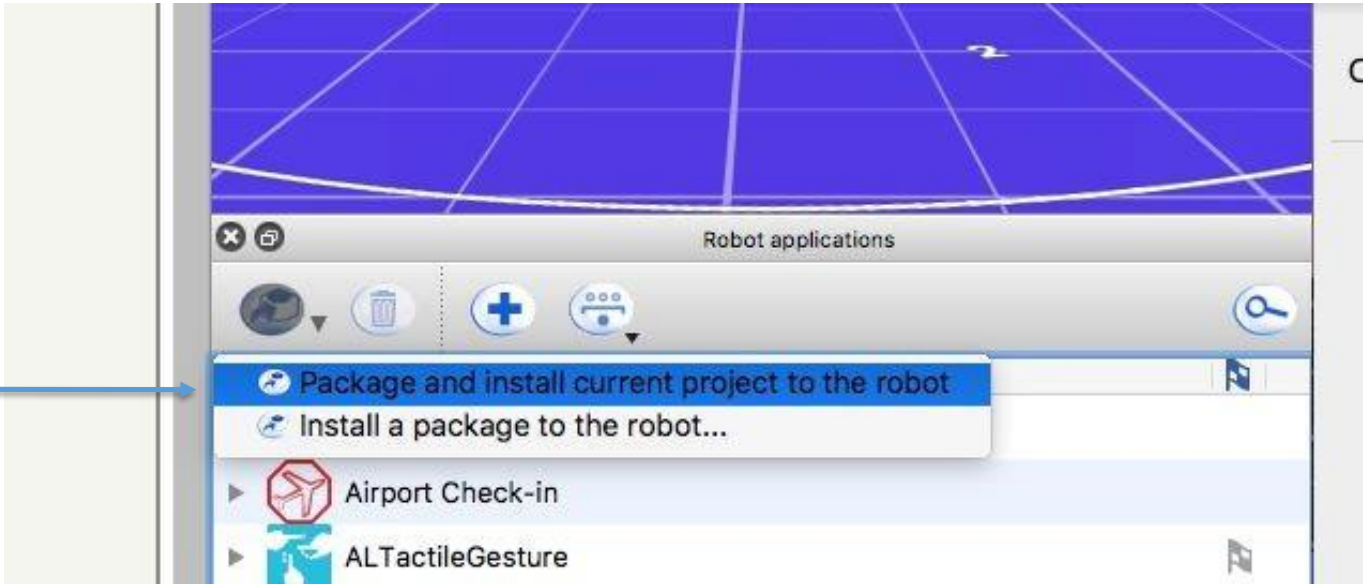
更多: http://doc.aldebaran.com/2-5 → NAOqi → NAOqi APIs

软件 - Choregraphe

运行App

→ 所有单次运行的app都有一个统一的名字 “.lastUploadedChoregrapheBehavior”
(临时文件，不保存。)

如果需要保存并安装
到机器人身上:



软件 - Choregraphe

如果想要导出pkg文件包! （APP安装包）

注意: 填写属性栏各项属性:

Application:

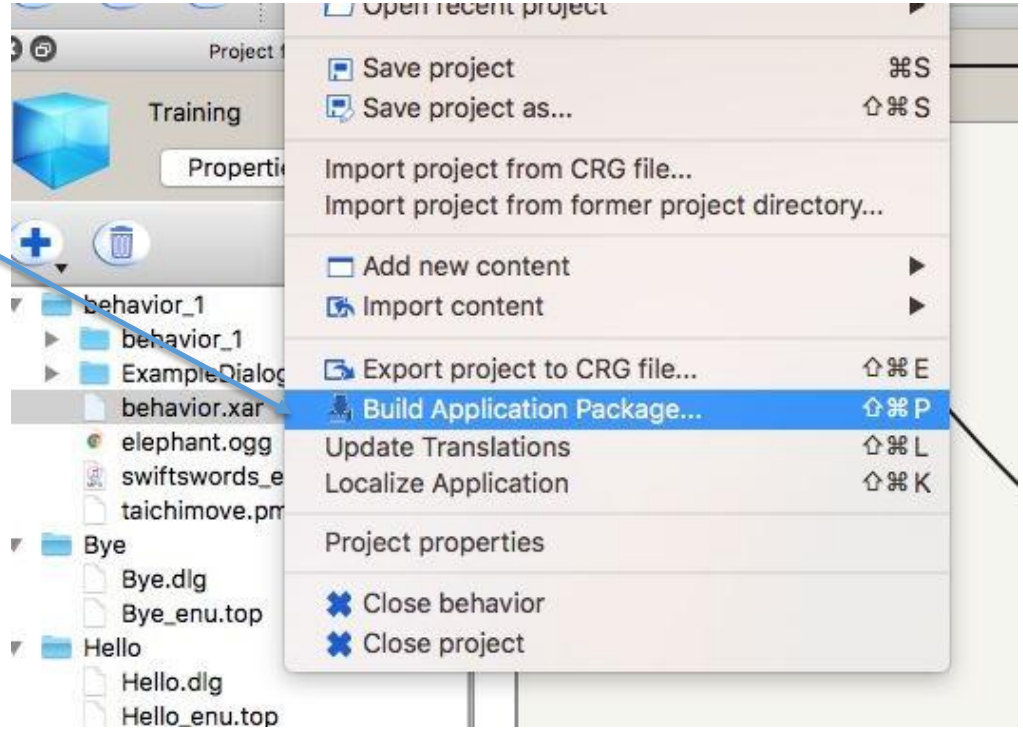
- Name
- ID
- Description…

Behaviors:

- Name
- Trigger sentences

Dialogs:

- Availability from autonomous life



软件 - Choregraphe

是不是很　方便!　快捷!

等等!

这只是个**开始**!

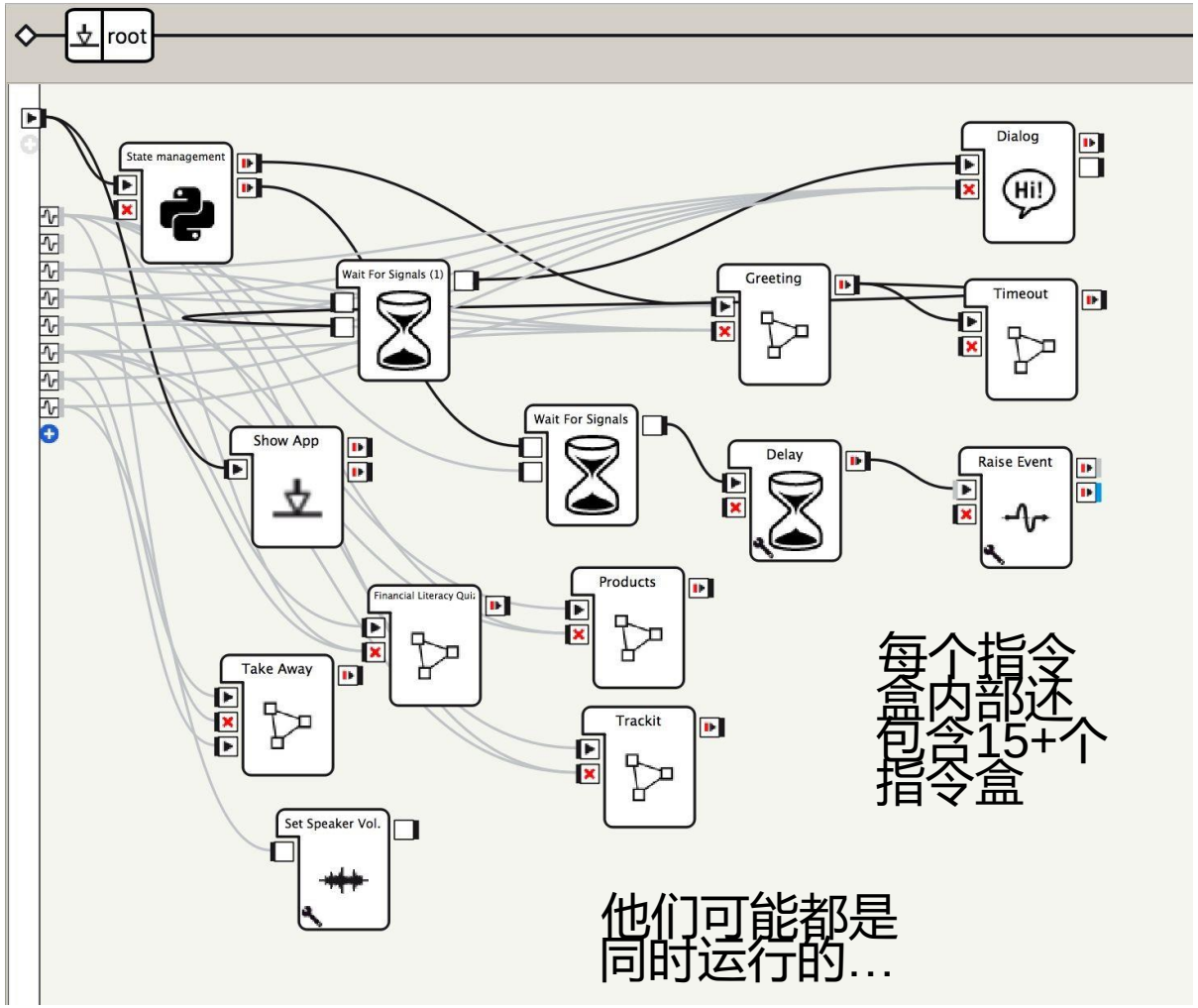
软件 - Choregraphe

一个“正式”的App，通常有2000行以上的代码，如果全用指令盒的话，很快就会形成一个“蜘蛛网”…

以至于使程序有以下缺点:

- 难以阅读
- 很难debug
- 没有清晰的逻辑结构

所以:
⇒ 还是用**Python**吧!



在一个 Choregraphe 项目里, 使用这样的结构 和 main.py:

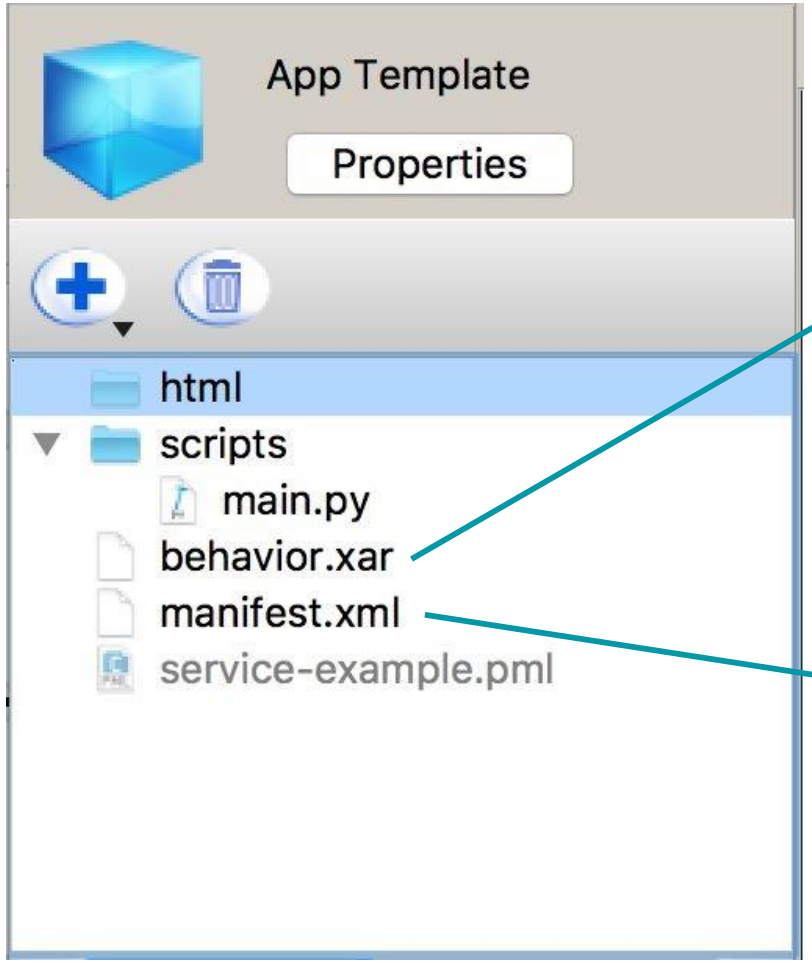
软件 - Python Apps

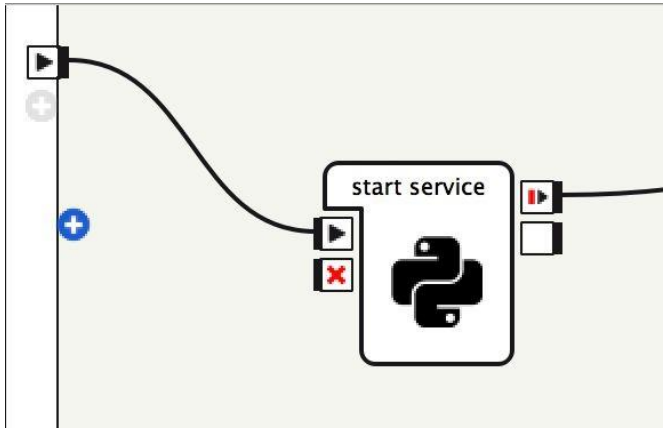
“App Template”服务

用这个start service指令盒来启动

```
s = self.session()
mgr = s.service(“ALServiceManager”)
```

```
mgr.startService(“AppTemplate”)
```






```
<executableFiles>
  <file path="scripts/main.py"/>
</executableFiles>
<services>
  <service
     autorun="false"
execStart="scripts/main.py"
name="AppTemplate"/>
  </services>
```

在manifest中声明main.py 为一个可执行的“AppTemplate”服务

用文本编辑器编辑manifest.xml文件

软件 - 其他

你可以在机器人上用哪些软件呢?

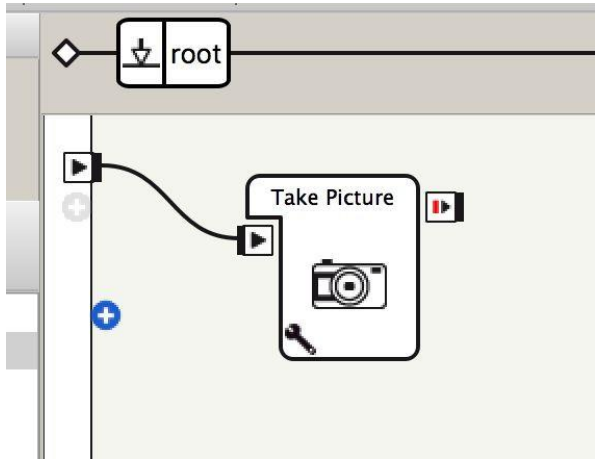
- 访问文件系统文件：SFTP
- 访问终端：SSH

当你在机器人上工作时，机器人头部处理你的所有命令

- 录音存储在头部
- 日志存储在头部

软件 - 文件系统

测试：拍照



用SFTP工具连接到机器人，并查看“recordings”文件夹下的文件

Host: IP Address （
机器人联网ip）
Username: nao
Password: Robot Password (default nao)
Port: 22

软件 - SSH

现在让我们使用SSH访问机器人的头部!

工具：“putty” （Windows系统）
“ssh”命令行工具 （Linux / Mac系统）

用账号密码登录，默认都是 nao
ssh nao@<IP-of-robot>

默认密码：nao

测试:

```
pepper [0] ~ $  [type here:] say hello
```

软件 - SSH

NAOqi有一个非常好的命令行工具 “qicli”

使您能够访问机器人中可用的所有服务Service，方法Method 或信号Signal，您将在代码中使用这些服务，方法或信号。

示例: （通过直接调用api拍一张照片）

```
qicli call ALPhotoCapture.takePicture “/home/nao” “photo.jpg”
```

...然后可以找到那个相关的图片文件

软件 - SSH

完成测试:

```
qicli info                  (列出所有载入的服务模块)
qicli info ALSystem         (列出ALSystem中所有函数)
qicli call ALSystem.shutdown  (调用ALSystem中shutdown函数)
```

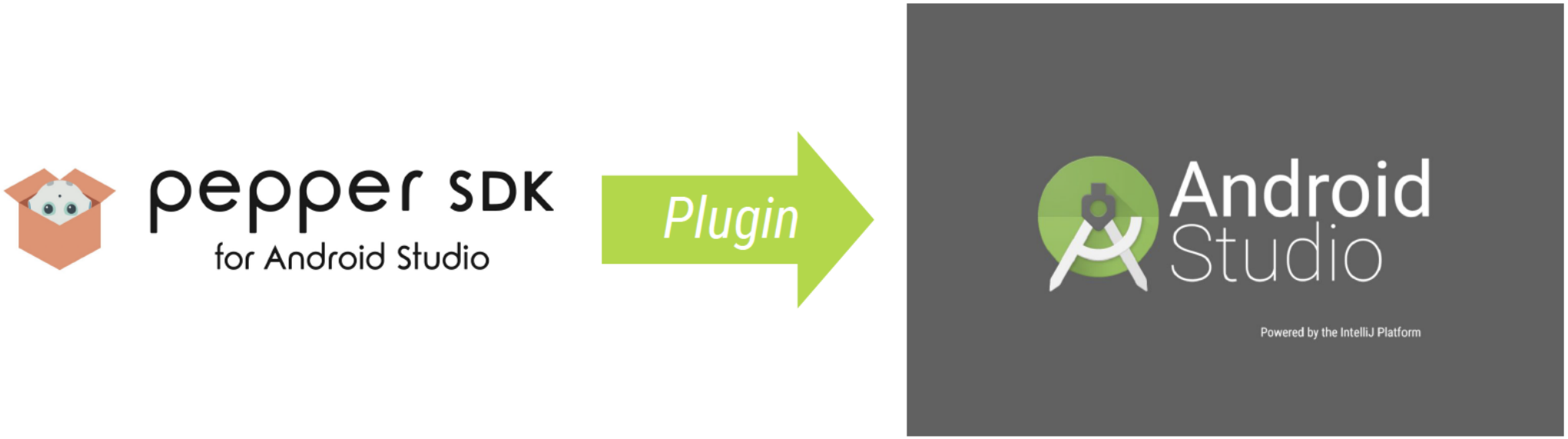
软件
基于 NAOqi 2.9

Android &Pepper

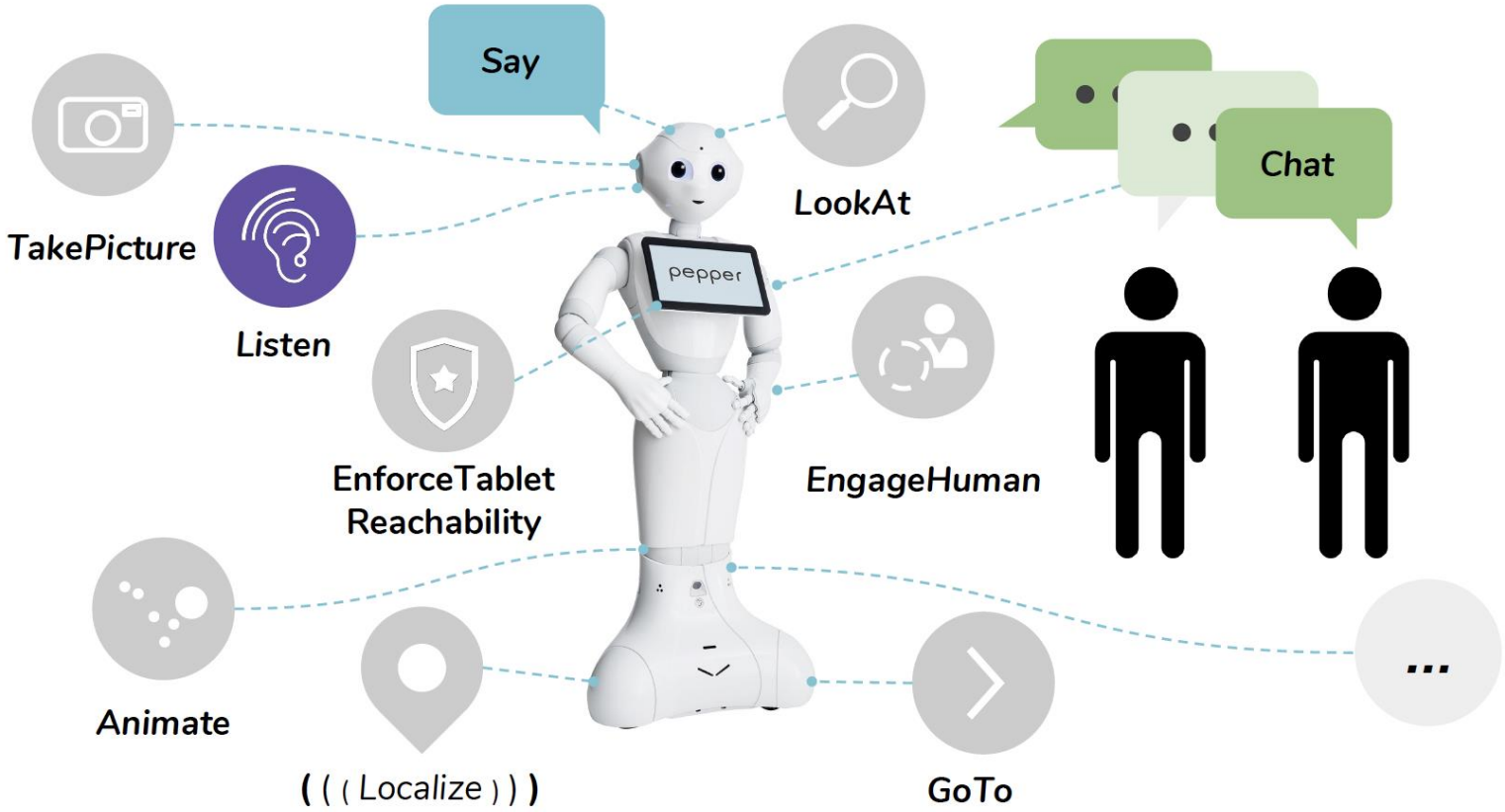
- Running NAOqi OS *(based on Linux)*
- CPU: 1.91 GHz quad-core Atom E3845
- Wi-Fi

- Running Marshmallow
- Certified GMS (CTS)
- CPU: 1.3 GHz quad-core ARM
- Wi-Fi, Bluetooth
- DDR3 SDRAM: 1GB *(512MB * 2)*
- 15GB
- Size: 10" Type: IPS Resolution: 1280x800
- Multitouch

机器人应用程序开发环境：Android Studio



现有API方向包括:


Say
TakePicture
Listen
LookAt
Chat
pepper
EnforceTablet
Reachability
EngageHuman
Animate
(((Localize)))
GoTo
...

机器人竞赛

RoboCup@Home

比赛目的:
- 人机交互
- 社交相关
- 实际应用为导向
- 科学上挑战
- 趣味性

软件架构:

操作系统 Ubuntu 16.04 LTS; NAOqi OS; OpenNao VM 2.4.3

中间件　ROS Kinetic; RSB 0.16 ; NAOqi 2.5.5

SLAM　ROS Gmapping

导航　ROS planning pipeline

物体识别 Classification Fusion (CLAFU)

人脸识别 strands perception people

Behavior 控制 SMACH

语音合成 Mary TTS

语音识别 Speech Recognition PocketSphinx with context dependent ASR

Code: https://github.com/CentralLabFacilities