Poppy Humanoid Robot, Version 1.0

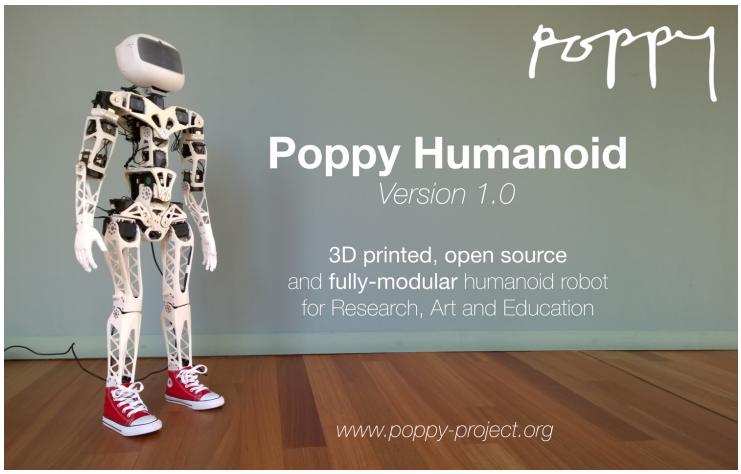
Manon Cortial, Generation Robots

May 27, 2015

Part I
Introducing the Poppy Humanoid robot

Table of Contents

1 The Poppy project



1.1 Philosophy

Poppy is an open-source platform for the creation, use and sharing of interactive 3D printed robots. It gathers an interdisciplinary community of beginners and experts, scientists, educators, developers and artists, that all share a vision: robots are powerful tools to learn and be creative.

Poppy tools are designed to be modular, easy to use, and easy to integrate providing a set of building blocks that can be easily assembled and reconfigured.

The Poppy community develops robotic creations that are easy to build, customise, deploy, and share. It addresses mechanics, electronics, and software.

It promote open-source by sharing hardware, software, and web tools.

1.2 Target public

The Poppy project targets the needs of a multidisciplinary community where researchers, teachers, artists, and robotic enthusiasts can share their work and ideas.

1.2.1 Scientists

The Poppy platform was initially designed within a publicly funded scientific project (Flowers project-team funded by ERC Grant Explorers, Inria and Rgion Aquitaine). It targeted to respond jointly to three complementary needs in robotics and cognitive science.

Make the body an experimental variable: The properties of the body (shapes, elasticity, distribution of mass,) have a crucial impact on sensorimotor control, cognitive skills and social interaction. For example, adequate shapes of legs can considerably simplify the acquisition of locomotion. The social and

emotional response of a human in front of a robot is also strongly impacted by its body appearance. Thus, a scientific enquiry of the role of the body requires the possibility of fast design, building and experimentation of alternative morphologies. 3D printing and other rapid prototyping techniques now make it possible, and this was leveraged in the design of the Poppy platform.

Make scientific output openly accessible, reproducible and cumulative: Robots are often complex systems, and this has so far often resulted in robotics research project which results were relying on either closed platforms, hiding sometimes crucial details, or experiments which cannot be reproduced. This is a barrier to scientific progress. The Poppy platform is a step towards addressing this issue, being accessible both in terms of cost and complexity, and allowing researchers to share hardware and experimental details in addition to code and algorithms;

Make scientific output usable outside science labs: A number of research projects in science labs could in principle be reused in educational, FabLab or industrial projects. Yet, the difficulty of accessing information through the formalism of scientific publications is often a strong obstacle. The sharing of experimental materials through the Poppy platform opens new possibilities in this perspective.

The Poppy platform allows users to create and modify easily the shape of robots. Poppy creatures can also be used directly as reproducible and accessible experimental platforms. For example:

- Investigate quickly, and at a reasonable cost, various mechanical design for body part such as the leg, or test several mechanism for the feet, and study their impact on balance or energy consumption.
- Explore how different body appearances can impact human-robot interaction, e.g. how humans perceive the movements or the internal state of the robot.
- Test how autonomous learning algorithms allow a robot to learn how to control new morphologies and interact with objects, such as with the compatible open-source Explauto python library for curiosity-driven learning.

Thanks to the availability of simulators, one can also study how skills acquired by the robot in simulation (e.g. through stochastic optimization), can be transferred efficiently to the physical robot.

Poppy is accessible, and other science lab in the world will be able to reproduce your research, and extend it with their own idea. Thanks to the 3D printing technology, off the shelf affordable components, and true open source community, you can have very short cycles of development. Your research can go fast.

1.2.2 Education

Poppy targets a major societal challenge: allowing everyone to understand and become creator of the digital world around us. This extends from learning

programming, to learning how to invent and build physical objects through digital fabrication, including 3D printing.

The modularity and accessibility of the platform allows to build educational project addressing two essential features met in many real world industrial and creative contexts:

- Integration of multiple technologies and methodologies within a system: 3D printing, programming, electronics, control theory, learning algorithms, design, human-machine interaction, software tools for collaborative projects;
- Connections and collaborations between multiple disciplines: engineering, science, design, human sciences, art;

Because it is based on open online sharing, it also allows for cumulative sharing of pedagogical and technological contributions.

The Poppy platform can be used in many different ways in educational projects. A common thread is integration and collaborative work. This means it will find its full potential within projects where students work by groups on complementary sub-projects (like the mechanical design of a new hand, the assembly of an arm, the addition of Arduino based sensors, the programming of a behavior), and target to integrate their work.

For example, educators recently explored how to use the platform for teaching mechanical design and programming at high-school level (Lyce Saintonge, Bordeaux) or teaching rapid prototyping, material sciences and programming in an engineering school (ENSAM, Bordeaux). The platform could also be used to teach human-robot interaction and human factors at graduate level in universities, or statistical machine learning applied to robot control.

1.2.3 Artists

Poppy is an open-source modular platform based on simple and robust technology, which can be hacked by non-technicians, opening the way to new avenues for creation.

The platform is based on technological bricks allowing the creation of novel robotic shapes or behaviours. It also includes examples of full creatures like the Poppy Humanoid, or the Poppy Ergo, which can directly be reused or modified for artistic purposes.

1.3 Poppy creatures

Poppy creatures are robots created using the Poppy philosophy and controlled through the Poppy software. Poppy_ergo and poppy_ergo_jr are little arms and Poppy Humanoid is a fully actuated humanoid robot.

Other creatures will be created soon by the community.

2 The Poppy Humanoid robot

This is the iconic robot of the Poppy platform, one that was made first. This humanoid robot is 85 cm high and is formed using 3D printed parts, dynamixel actuators and various electronic materials. The robot is open source for both hardware and software. It is possible to design variations and develop according to your needs whether scientific, educational or artistic.

The Poppy Humanoid robot is particularly lightweight (3.5kg). It has 25 degrees of freedom, and above all assets, one of its original features, is to have a multi-articulated trunk with 5 degrees of freedom. It has a ODROID U3 board in the head, to run your programs and communicate through WiFi and Ethernet. A wide angle USB camera (170 FOV) located in the head can be used for artificial vision.

By default, Poppy Humanoid features an ODROID U3 but it is possible to replace this card with one Raspberry Pi 2, although for the moment, the use of the Raspberry Pi 2 with Poppy Humanoid robot is less documented.

2.1 Bill of Material

2.1.1 3D printed parts

lower_limbs:

- 2x leg (white polished polyamid)
- 1x thigh_right (white polished polyamid)
- 1x thigh_left (white polished polyamid)
- 1x hip_right (white polished polyamid)
- 1x hip_left (white polished polyamid)
- 1x pelvis (white polished polyamid)
- 1x simple_foot_left (white polished polyamid)
- 1x simple_foot_right (white polished polyamid)
- 2 x hip_z_to_hip_y-connector (white polished polyamid)

torso

- 2x double_rotation_MX64_link (white polyamid)
- 1x i101-Set_to_MX64_link (white polyamid)
- 1x abdomen (white polished polyamid)
- 1x spine (white polished polyamid)
- 2x double_rotation_MX28_link (white polyamid)
- 1x i101-Set_to_MX28_link (white polyamid)
- 1x chest (white polished polyamid)

upper_limbs

- 1x shoulder_right (white polished polyamid)
- 1x shoulder_left (white polished polyamid)
- 2x arm_connector (white polished polyamid)
- 2x upper_arm (white polished polyamid)
- 1x forearm_left (white polished polyamid)
- 1x forearm_right (white polished polyamid)
- 1x hand_right (white polished polyamid)
- 1x hand_left (white polished polyamid)

head

- 1x neck (white polished polyamid)
- 1x head-back (white polished polyamid)
- 1x head-face (white polished polyamid)
- 1x support_camera (white polyamid)
- 1x screen (resine transparent)
- 1x hide-screen (black polyamid)
- 1x speaker_left (black polyamid)
- 1x speaker_right (black polyamid)
- 1x fake_manga_screen (black polyamid)

2.1.2 Electronics

Embedded control

- 2x USB2AX
- Hardkernel Odroid U3
- 8GB eMMC Module U Linux
- Cooling Fan U3 (Optionnal)
- USB hub

Power Supply

- DC Plug Cable Assembly 2.5mm
- 5V DC convertor

Audio/video

- 2x speakers

- ampli stereo

Communication

- Nano Wifi Dongle

Sensors

- Camera Videw with FOV 120 or 170!! + USB cable
- Sparkfun Razor 9DoF IMU (Optionnal)
- Manga Screen (Optionnal)

2.1.3 Robotis

Actuators Dynamixel

- 19 x MX-28AT
- 4 x MX-64AT
- 2 x AX-12A

Parts

- 19x HN07-N101 set
- 12x HN07-i101 Set
- 4x HN05-N102 Set
- 4x HN05-i101 Set

Screws:

- 1x Wrench Bolt M2*3 (200 pcs)
- 1x Wrench Bolt M2.5*4 (200 pcs)
- 1x Wrench Bolt M2.5*6 (200 pcs)
- 1x Wrench Bolt M2.5*8 (200 pcs)
- 1x BIOLOID Bolt Nut Set BNS-10
- 1x Nut M2.5 (400 pcs)
- 1x N1 Nut M2 (400 pcs)

Cables

- 3x SMPS2Dynamixel
- 1x SMPS 12V 5A PS-10
- 3x BIOLOID 3P Extension PCB
- 1x Robot Cable-3P 60mm 10pcs

- 1x Robot Cable-3P 100mm 10pcs
- 1x Robot Cable-3P 140mm 10pcs
- 1x Robot Cable-3P 200mm 10pcs

Custom:

- 3x cable-3P 22cm
- 3x cable-3P 25cm
- 2x cable-3P 50cm
- 2x Cable-4P 200mm, wires D+/D- cut

2.2 Distributors

The only official ditributor is Generation Robots. They ship world-wide and provide the complete kit with pre-soldered electronics. You can buy the kit with or without the 3D printed parts.

Part II
Assembly Guide

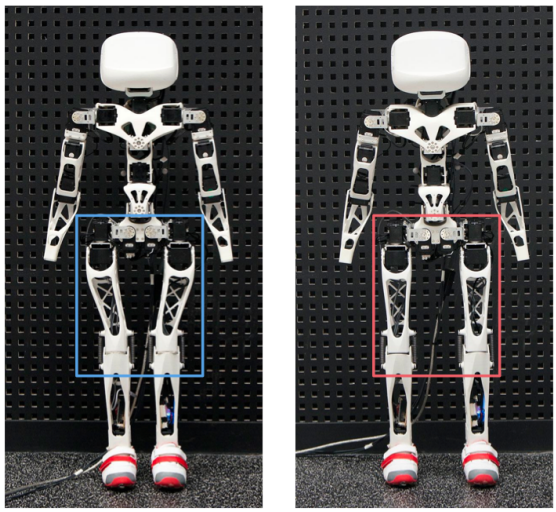
Table of Contents

1 Introduction

1.1 The Poppy project

Poppy is an open hardware and open-source robotics project. It has been designed to allow researchers and students to easily remove and replace some parts of the body.

For example, different leg shapes have been tested on the Poppy Humanoid robot to make the robot walk.



(a) bended thighs (b) straight thighs

1.2 Safety warning

The Poppy humanoid robot is built using mainly MX-28 Dynamixel servomotors, which are pretty powerful and may be harmful to your fingers or materials.

So be very careful and put the robot in a free space while testing your programs.

1.2.1 About this documentation

This documentation contains some help and tips to build a Poppy Humanoid robot. It does not replace the videos made by Inria, but complete and sometimes corrects or updates them.

It contains a bit of context about Dynamixel servomotors and how to assemble them (section 2) and also how to set the right parameters for them (section 3).

You will also find pictures of all the parts to help you name them (section 4), then assembly tips and links to the assembly videos (section 5). As there is no video for the head assembly, this doc contains a pretty complete guide for head assembly (section 5.4).

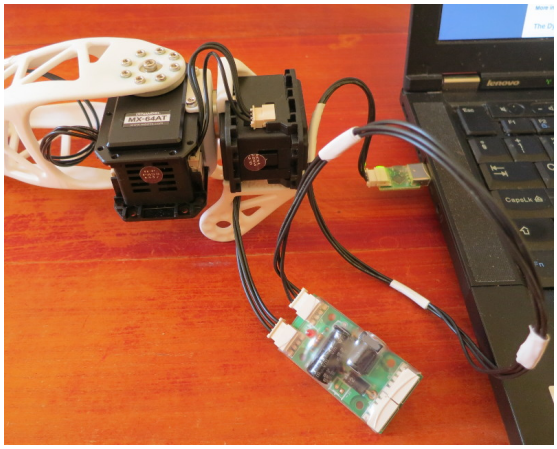
At the end, you will find a list of useful links (section 7) to help you find more information and help.

Please don't hesitate to comment and correct this documentation on the Poppy forum !

2 Dynamixel hardware

The Poppy Humanoid robot is mainly built with MX-28AT Dynamixel servomotors (MX-28T are the previous version and can be used without any problem). The other servomotors are MX-64T (bigger and stronger) and AX-12A (smaller, used for the head).

Each Dynamixel servomotor embeds an electronic board allowing it to receive different kind of orders (about goal, torque...) and communicate with other Dynamixel servos. Therefore, you can chain up several Dynamixel servomotors (each with a different ID) and command them all from one end of the chain: each servomotor will pass the orders to the next one.

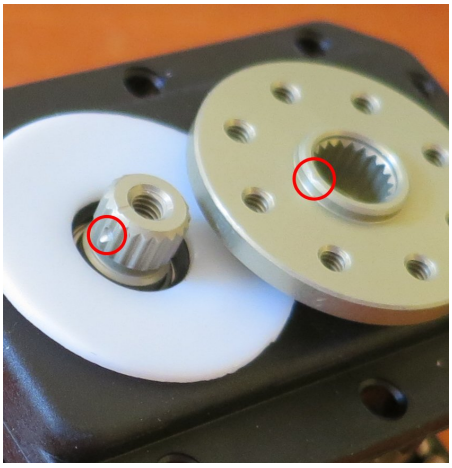


2.1 Putting the Dynamixel horns to zero

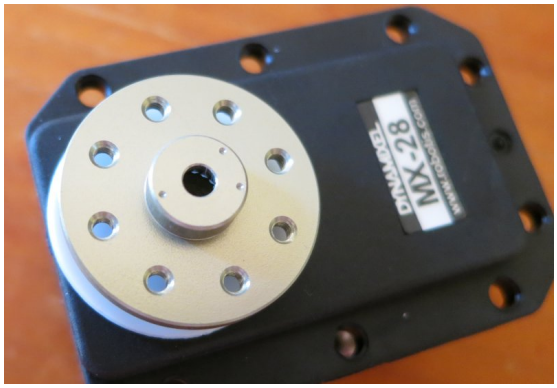
This step is critical to avoid damaging your Dynamixel servomotors !

When you receive your Dynamixel servomotors, the horns are not mounted. They are included in the packaging if the servo is packaged alone or packaged separately for 6-pieces bulks (see next section to know what horn goes to what servo).

When putting the controlled horn, be very careful to **put the dot on the horn at the same point than the dot on the servo axis**. Once the horn is put, it is most of the time **impossible to remove** ! This will ensure that the zero position of the servo matches with the zero position of the structure around.



On the outside of the horn, you also have three dots indicating the orientation. You should find the same three dots on structural parts, so be sure to match them.



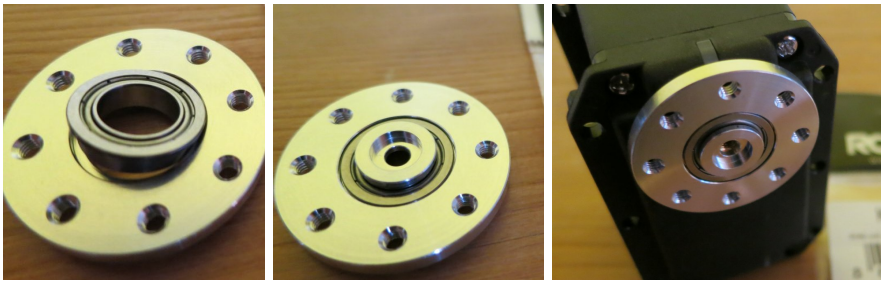
2.2 Horns of MX-28 and MX-64

On each Dynamixel servomotor apart from the AX-12A, you will have to mount a horn to the motor axis. Most of the time, you will also have to mount a free horn on the opposite side to provide better fixation points for the structure parts.

To mount the main horn, put the plastic ring (white or black) and drive the horn on the axis. **Be careful of the zero when putting the main horn!** Then put thread locker on the big screw and screw it in the middle.

Main horn mounted on a MX-28

For the free horn, first clip the ball bearing and the cap on the side without shaft shoulder. Then put the horn on servomotor (with shaft shoulder on servo side). Put thread locker on the big screw and screw it. The horn should turn freely.


Free horn mounted on a MX-64

Quick reminder of horn names and screw sizes:

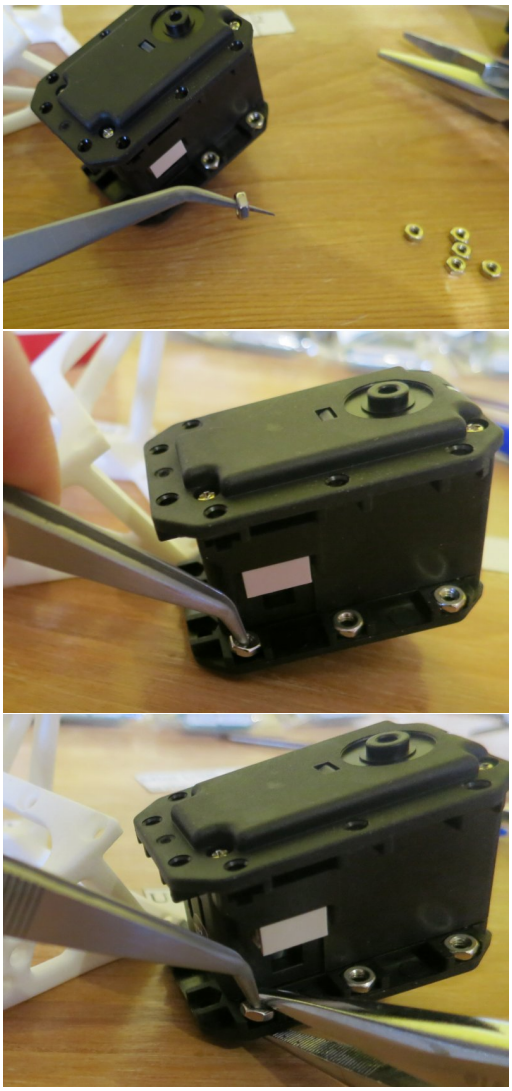
Servomotor	main horn	free horn	big horn screw	horn screws	case screws
AX12-A	none	none	⌀3x10mm	⌀2	⌀2
MX28	HN07-N101	HN07-I101	⌀2.5x8mm	⌀2x3mm	⌀2.5x6mm
MX64	HN05-N102	HN05-I101	⌀3x8mm	⌀2.5x4mm	⌀2.5x6mm

You need 1.5mm for ⌀2 screws, an allen wrench of size 2mm for ⌀2.5 screws and 2.5mm for ⌀3 screws.

2.3 Putting the nuts

To attach structural parts on the body of the servomotors, you have to first insert the nuts in their sites. This step may be quite painful if you don't have elfic fingers.

Here's my tip: take the nut using thin tweezers and bring it in the site with the right orientation. Put the end of the tweezers in the hole to ensure good alignment. Then use flat pincers to ajust the nut.



These nuts correspond to diameter 2.5mm screws, Allen wrench 2mm.

To build a full Poppy Humanoid robot, an electrical screwdriver is strongly advised!

3 Addressing Dynamixel motors

By default, every Dynamixel servomotor has its ID set to 1. To use several servomotors in a serial way, each of them must have a unique ID.

3.1 Installing the driver for USB2AX

USB2AX is the device that will connect the Poppy Humanoid robot's head to the Dynamixel servomotors. It can also be used to control the servomotors directly from your computer and that's what we will do to address the motors.

On Linux, no installation is needed, but you must add yourself in the dialout group to have access to the USB port:

```
sudo addgroup "username" dialout
```

Otherwise, the driver is available here.

Don't forget to power up your motors (using a SMPS2Dynamixel) otherwise they won't be detected !

3.2 Installing the scanning software

Use one of the two following software to access the Dynamixel servomotors registers:

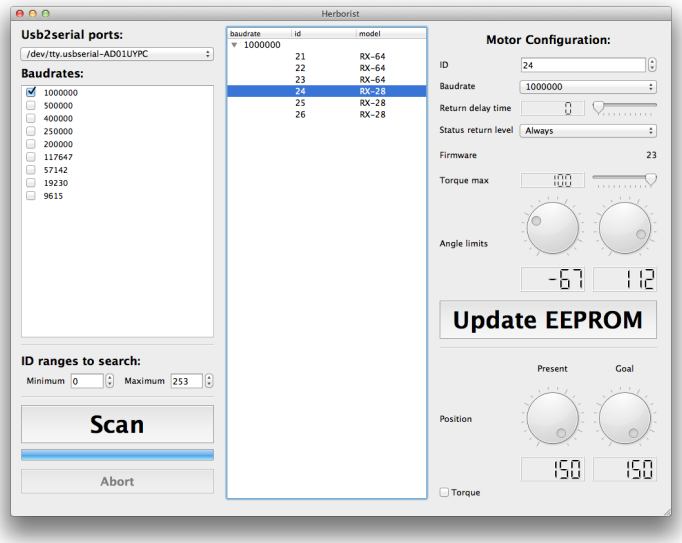
- Herborist: tool created by the Poppy Project team.
- Dynamixel Wizard: windows-only tool provided by Robotis.

Herborist comes with the Pypot library, but needs the additional library PyQt4 for graphical interface.

```
sudo pip install pypot
sudo apt-get install python-qt4
```

It should then be directly accessible for a terminal:

```
sudo herborist
```



Connect each motor **one by one** to the USB2AX and use the 'scan' button in Herborist or Dynamixel Wizard to detect it. If it's a new motor, it should have ID 1 and baudrate 57600bps, apart from AX-12A servos who already have a 1000000 baudrate.

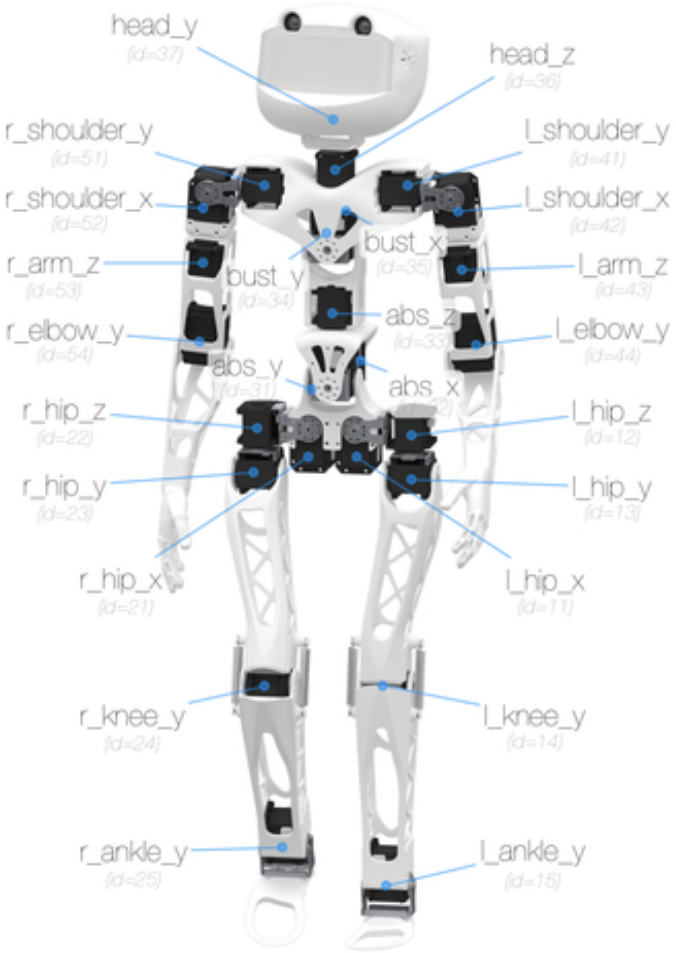
You have to set:

- ID corresponding to the naming convention
- Baudrate to 1 000 000 bps
- Return delay time to 0 ms instead of 0.5 ms

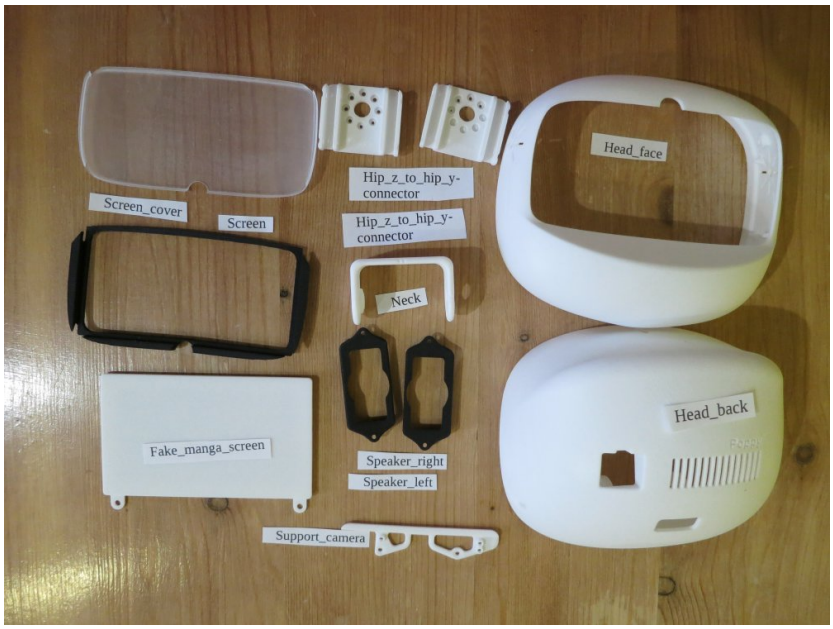
In Herborist, don't forget to click on the 'Update EEPROM' button so the changes are taken in account.

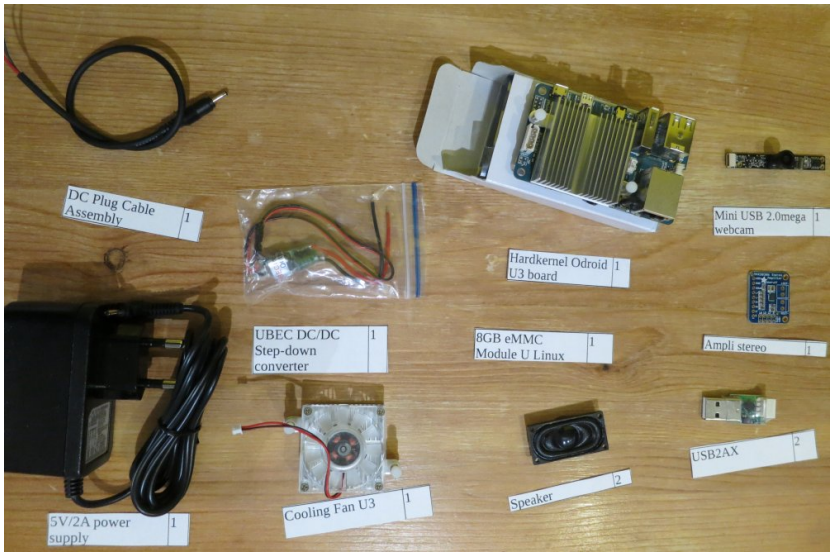
3.3 Naming conventions

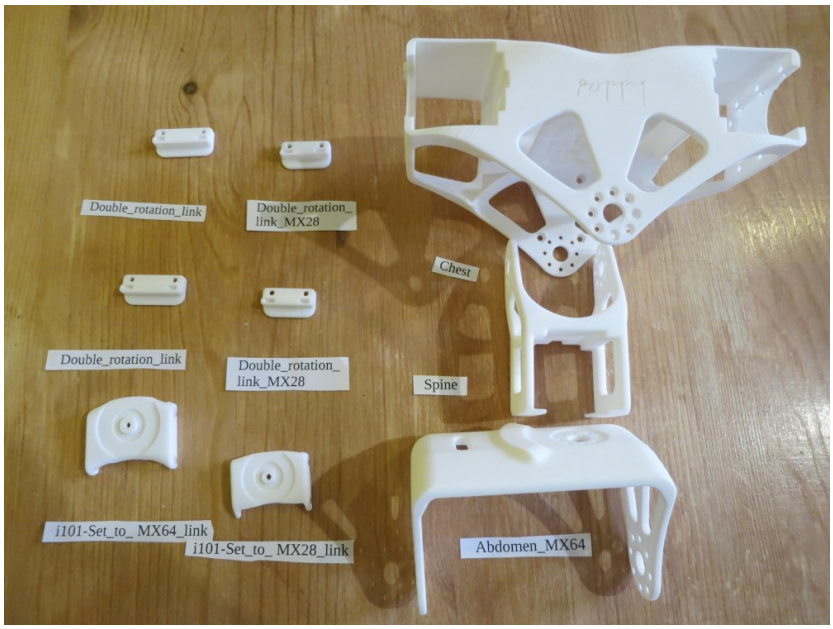
If you want your PoppyHumanoid object to correspond to your robot without having to modify the configuration file, you should stick to the Poppy Humanoid robot naming and addressing convention. This will ensure that, in your code, when you use a motor's name, you will really send orders to the corresponding physical motor.

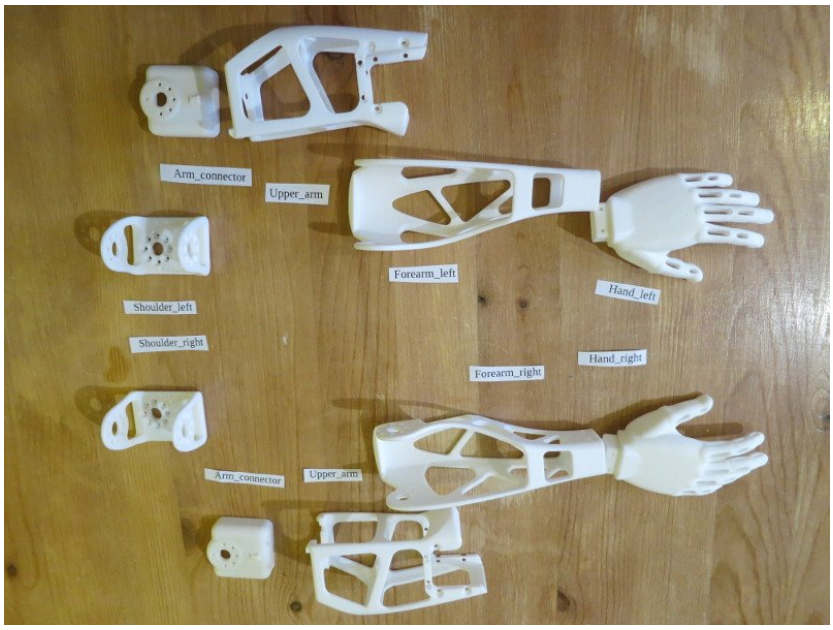

head_y
(id=37)
head_z
(id=36)
r_shoulder_y
(id=51)
l_shoulder_y
(id=41)
r_shoulder_x
(id=52)
l_shoulder_x
(id=42)
bust_x
(id=35)
r_arm_z
(id=53)
bust_y
(id=34)
l_arm_z
(id=43)
abs_z
(id=33)
r_elbow_y
(id=54)
l_elbow_y
(id=44)
abs_y
(id=31)
abs_x
r_hip_z
(id=22)
l_hip_z
(id=12)
r_hip_y
(id=23)
l_hip_y
(id=13)
r_hip_x
(id=21)
l_hip_x
(id=11)
r_knee_y
(id=24)
l_knee_y
(id=14)
r_ankle_y
(id=25)
l_ankle_y
(id=15)

4 Structural parts





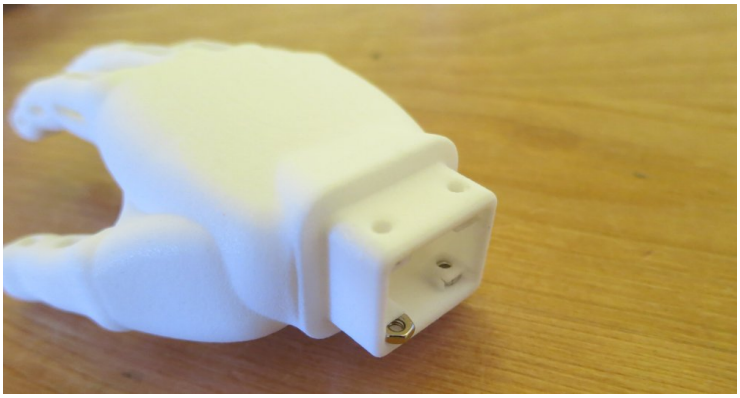

Double_rotation_link
Double_rotation_
link_MX28
Chest
Double_rotation_link
Double_rotation_
link_MX28
Spine
i101-Set_to_ MX64_link
i101-Set_to_ MX28_link
Abdomen_MX64


Arm_connector
Upper_arm
Forearm_left
Hand_left
Shoulder_left
Shoulder_right
Forearm_right
Hand_right
Arm_connector
Upper_arm

5 Assembly tips

5.1 Arms

- **Right/Left forearm** The hand design slightly changed from the videos, but the nuts and screws remain the same.



- **Right/Left upper arm** Plug a 200mm cable in the unused plug before screwing the arm_z motors (ids 43 and 53), because it will be really hard to plug once the motor is inside the structure part.
- **Right/Left upper arm/shoulder**

- **Right/Left arm assembly**
- **Trunk and arms assembly** To distinguish between left and right shoulder parts, look at the three dots: the single dot should be down when the shoulder is in "zero" position (along the shoulder_y motor).

Motors lists:

Sub-assembly name	Motor name	Type	ID
Left upper arm/shoulder	l_shoulder_x	MX-28AT	42
Left upper arm	l_arm_z	MX-28AT	43
Left upper arm	l_elbow_y	MX-28AT	44

Sub-assembly name	Motor name	Type	ID
Right upper arm/shoulder	r_shoulder_x	MX-28AT	52
Right upper arm	r_arm_z	MX-28AT	53
Right upper arm	r_elbow_y	MX-28AT	54

5.2 Trunk

- **Double MX64**
- **Double MX28** Don't screw the i101-Set_to_ MX28_link (the plastic part with a free horn on it) too tightly, or don't screw it at all since you will need to unscrew it during the trunk assembly.
- **Spine**
- **Chest** The video shows a HN07_I101 in the prepared parts, but you don't need it.
- **Trunk assembly** You have to insert the nuts in the chest before mounting the double MX-28 part. You also have to put nuts in the abdomen before mounting the double MX-64 part.

  The abdomen part you have has a "Poppy" mark on the back, while the one on the video don't. You also have holes to screw the SMPS2Dynamiel, instead of sticking it (use 2.5*8mm screws).

Motors list:

Sub-assembly name	Motor name	Type	ID
Double MX64	abs_y	MX-64AT	31
Double MX64	abs_x	MX-64AT	32
Spine	abs_z	MX-28AT	33
Double MX28	bust_y	MX-28AT	34
Double MX28	bust_x	MX-28AT	35
Chest	head_z	AX-12A	36
Chest	l_shoulder_y	MX-28AT	41
Chest	r_shoulder_y	MX-28AT	51

5.3 Legs

There is only a video for left leg assembly. While assembling the right leg, be sure to put your motors symmetrical compared to the left leg. Also don't forget to change the motors IDs from 12-15 to 22-25.

- **Hip**
- **Tight**
- **Shin** If you received your Poppy kit from Generation Robots, you can use the custom 220mm cables instead of really short 200mm cables.
- **Right/Left leg assembly**
- **Pelvis** The videos shows ⌀2x5mm screws. Use the ⌀2x6mm screws that you can find in the Bolt-nut set BNS-10.

- **Torso and legs assembly**

Motors lists:

Sub-assembly name	Motor name	Type	ID
Pelvis	l_hip_x	MX-28AT	11
Left hip	l_hip_z	MX-28AT	12
Left hip	l_hip_y	MX-64AT	13
Left thigh	l_knee_y	MX-28AT	14
Left shin	l_ankle_y	MX-28AT	15

Sub-assembly name	Motor name	Type	ID
Pelvis	r_hip_x	MX-28AT	21
Right hip	r_hip_z	MX-28AT	22
Right hip	r_hip_y	MX-64AT	23
Right thigh	r_knee_y	MX-28AT	24
Right shin	r_ankle_y	MX-28AT	25

5.4 Head

Sub-assembly name	Motor name	Type	ID
Head	head_y	AX-12A	37

5.4.1 Setup of the Odroid board

The Odroid is normally shipped with a eMMC module with Ubuntu 1.14 already flashed (it should have a red sticker on it). Simply plug it on the Odroid board and power it. After boot time, it should have the red light steady and the blue light flashing.

If you don't have a pre-flashed eMMC module, follow these instructions: `https://github.com/poppy-project/poppy_install`

Connect the Odroid board to your network using an ethernet cable. You have to access a wired network for initial setup (I tried link-local without success).

Windows users may wish to install the Bonjour software (the link is for the printer version, which does very well what we want it to do). Bonjour is installed by default on Linux and Mac. It is used to communicate with another device using its name instead of its IP.

You should get an answer if you type:

```
ping odroid.local
```

Windows users now probably wish to install Putty or any SSH client. Linux and Mac users have one installed by default. Then:

```
ssh odroid@odroid.local
```

Password is odroid. Congratulations, you are now inside the Odroid!

Make sure the Odroid board has access to the internet and enter:

```
curl -L https://raw.githubusercontent.com/poppy-project/
poppy_install/master/poppy_setup.sh | sudo bash
```

Enter the odroid password. This command will download and run a script which will download and prepare installation. The board asks for a reboot:

```
sudo reboot
```

You loose the SSH connection. The board has changed hostname and password, so wait for the blue light to flash regularly and connect with:

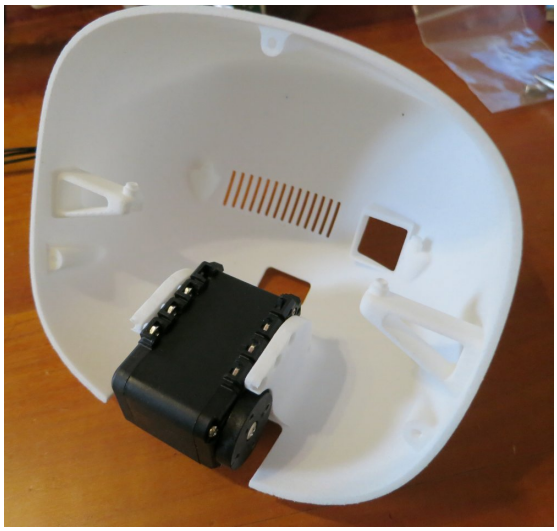
```
ssh poppy@poppy.local
```

As you guessed, password is poppy. Installation process takes place automatically (and takes a while). When you see 'System install complete', do a Ctrl+C to finish. After a new reboot, your Odroid board is ready.

5.4.2 Neck assembly

The last servomotor is head_y, a AX-12A. Set its ID to 37 and response time to 0 (baudrate is already at 1000000).

Screw the neck to head_z servo (⌀2x8mm screws). There are marks on the neck and on the servo to help you determine the orientation.



Put ⌀2 nuts in the servo case and attach it in the head_back part.

Assemble the servo on the neck (⌀2 screws on the controlled side, the big screw on the other side). You again have marks on the neck and on the servo for orientation.

Connect head_y to the dispatcher by passing the cable through the hole in the head.

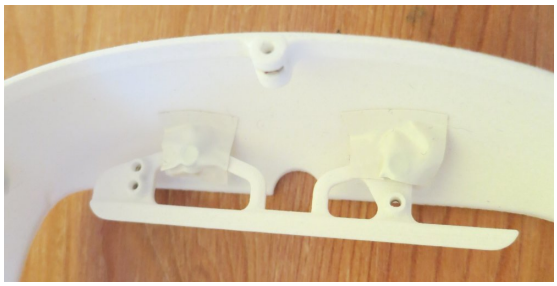
Plug a 500mm cable from the pelvis SMPS2Dynamixel board to the back of the head. Attach a USB2AX at the end of this cable in the head.

Use a 140mm cable to connect the head_y motor to another USB2AX.

5.4.3 Camera and screen

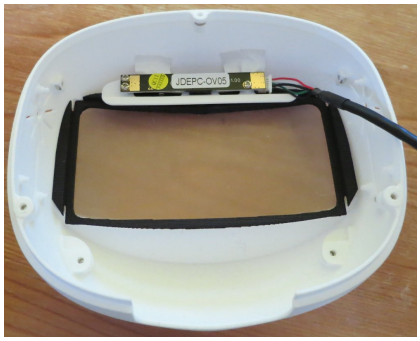
Put 3 ⌀2 nuts in the camera support and attach the camera using ⌀2 x6mm screws.

Attach the camera support to head_front using ⌀2.5x4mm screws. Put tape on the screws to avoid electrical interferences with the camera board.



Attach the camera to its support using ⌀2x6mm screws.

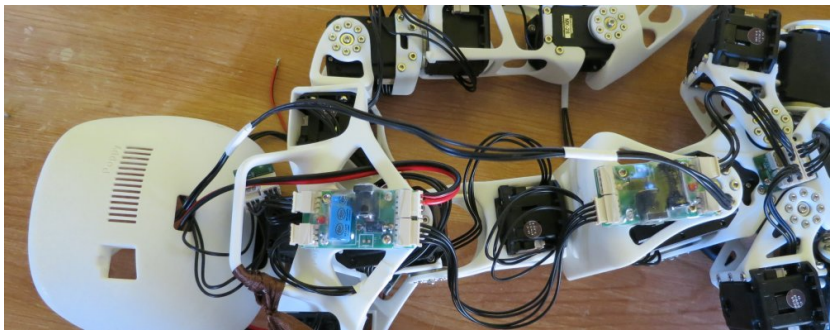
Put the screen and screen cover in the head. Attach the manga screen (or the fake one) with 2 ⌀2.5x6mm screws.



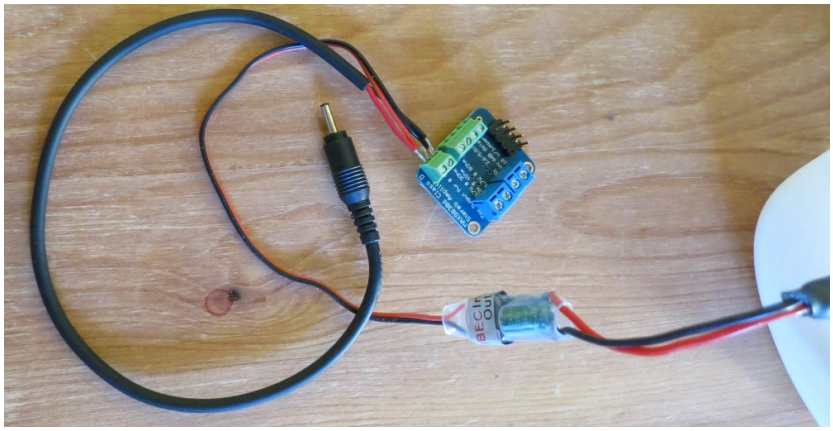
5.4.4 Electronics

If you don't have pre-soldered components, see: `https://github.com/poppy-project/Poppy-minimal-head-design/blob/master/doc/poppy_soldering.md`

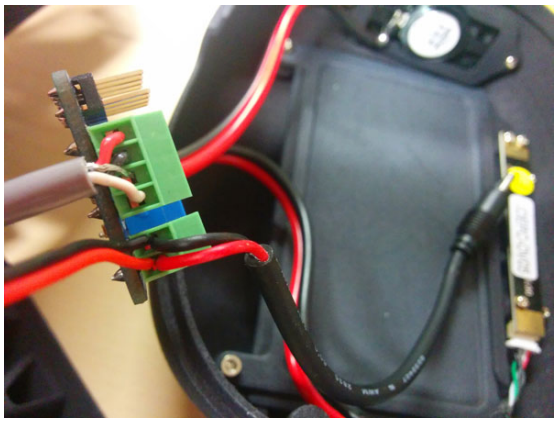
Pass the Dynamixel connector of the Ubec through the hole in the head and connect it to the torso SMPS2Dynamixel.



Attach both the other side of Ubec and the Odroid power cable to the audio amplifier. Be sure no to allow any short-circuit.

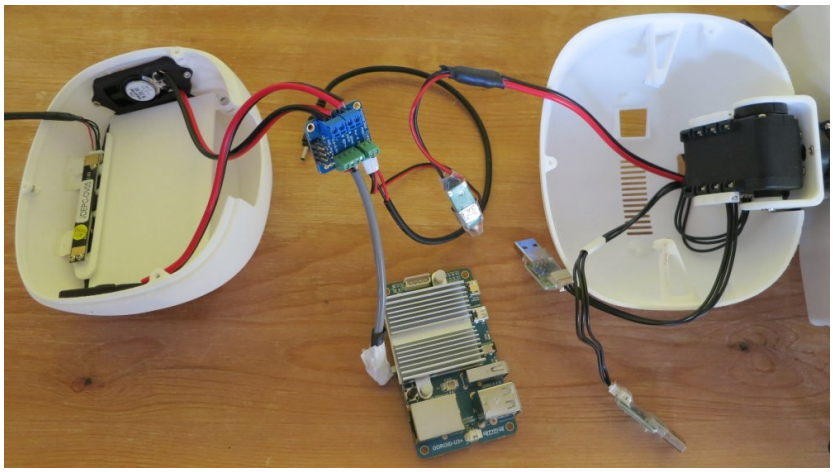


Plug the audio jack. Wires order from left to right (when power terminal is farthest right): red-black-uncolored-white

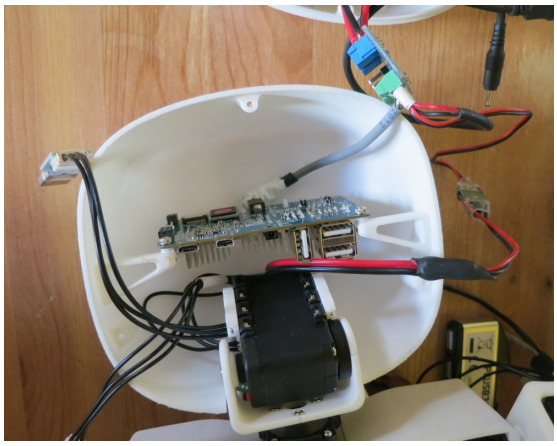


Put ⌀2 nuts around the flowers openings then attach the speakers using ⌀2 x3mm screws.

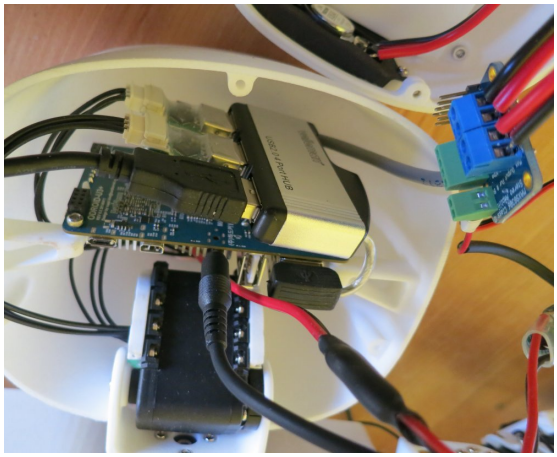
Connect the speakers to audio ampli, left speaker black wire on Lout, Right ampli black wire on Rout.



Plug audio jack in Odroid, then use 2 ⌀2.5x8mm screws to attach the Odroid board. Make sure the Ethernet connector is correctly placed in front of the corresponding hole.



Plug the power jack. On the hub, plug the camera and the two USB2AXs. Plug the wifi dongle if you have one. Push the hub above the Odroid.



Then close the head using 3 ⌀2x8mm screws.

6 Useful links

Assembly instructions:

https://github.com/poppy-project/poppy-humanoid/blob/master/hardware/doc/Poppy_Humanoid_assembly_instructions.md

Bill of Material:

https://github.com/poppy-project/Poppy-lightweight-biped-legs/blob/master/doc/BOM.md

Poppy project website: `https://www.poppy-project.org/`
Poppy project forum: `https://forum.poppy-project.org/`
herborist doc.: http://poppy-project.github.io/pypot/herborist.html
Dynamixel wizard doc.:

http://support.robotis.com/en/software/roboplus/dynamixel_monitor/quickstart/dynamixel_monitor_connection.htm

Bonjour software:

https://support.apple.com/kb/DL999?locale=fr_FR&viewlocale=fr_FR

STL files:

https://github.com/poppy-project/poppy-humanoid/releases/download/Official_1.0_Hardware_release/STL_3D_printed_parts.zip

Part III
Software Guide

Table of Contents

1 Introduction

1.1 How to use this guide?

This guide gives you an introduction to Poppy programming and helps you to set up your system and find the relevant documentations and tutorial in the Poppy environment.

If you are a **Poppy user** (beginner, teacher, artist...), you will mostly use the webserver (section 2) and Snap! (section 5).

If you are a **Poppy developper** (student, researcher, application developper...), you want to know about the webserver (section 2) and the pypot library (section 6)

If you **don't have a Poppy robot**, see section 3 to setup your computer, section 4 for the V-REP simulator, then have a look at Snap! (section 5) or Pypot (section 6).

If you are a **Poppy contributor**, you want to know about it all! ;-)

1.2 Safety warning

The Poppy humanoid robot is built using mainly MX-28 Dynamixel servomotors, which are pretty powerful and may be harmful to your fingers or materials.

So be very careful and put the robot in a free space while testing your programs.

1.3 Useful tools

1.3.1 Bonjour

Bonjour is a software made by the Apple company to communicate with a device using its name instead of its IP address.

It is installed by default on Mac and most Linux. Windows users can find an installer here: `https://support.apple.com/kb/DL999?locale=fr_FR&viewlocale=fr_FR`

If you don't want to use Bonjour, replace all occurences of 'poppy.local' by the IP address of the robot in this tutorial.

1.3.2 SSH

SSH means Secure SHell. It's a protocol allowing to open a terminal from another device (say, your robot) on your computer.

We will use SSH to have a look at the documents in the robot, change the settings and install new programs. If you are only using the webservices and/or Snap!, you don't need SSH.

SSH is installed by default on Mac and most Linux. The most popular SSH software for Windows is Putty.

1.3.3 Git and Github

Git is a versioning software, very useful to secure older versions of your code while workig on a new one. It also allows to merge different modifications from different developpers on the same project.

Github is a website offering free remote hosting for Git project (if they are public). All sources for Poppy are on Github.

Git is not installed by default. For Debian and Fedora based computer, the packet is called git. For Mac, have a look at `http://sourceforge.net/projects/git-osx-installer/`. For Windows, find the installer here: `http://msysgit.github.io/`.

If you are mainly a user, you will use *git clone* to get new packages and *git pull* to update them.

If you are a developer, quickly create your own account on `https://github.com/` and follow the instructions.

2 Poppy Humanoid webservices

Let assume you have a brand new Poppy Humanoid robot with and embedded Odroid (or Raspberry Pi) board already prepared using the poppy-install script.

Connect your Poppy Humanoid robot on your network with an ethernet cable. Be sure your computer is connected to the same network. Open your web browser and enter:

`http://poppy.local`

It opens the Poppy Humanoid web interface, which allows you to:

- Connect to a local wifi

soon Put the robot in standing, sitting or compliant position

soon Record and play moves

soon Open a ipython console to

3 Setting up your computer

You may wish to install the Poppy development tool on your computer for several reasons:

- you have want to use a simulator
- you want to control a Poppy creature directly pluggued on your computer
- You do lots of development and want an environment friendlier than a webservice or SSH.

There are two ways of installing pypot and Poppy-Humanoid on your computer : using pip is easier, but you will get less frequent (but hopefully more stable) software updates. Installing from the sources will get you the latest version of pypot and the possibility to contribute to its development.

3.1 From pip

If you're using Ubuntu, Python and pip should be already installed. Otherwise:

```
sudo apt-get install python2.7
sudo apt-get install python-pip
```

Then you simply need to install the Poppy-humanoid package. As pypot is a dependency of Poppy-humanoid, it will be installed too, as well a a few other libraries including numpy and poppy-creature.

```
sudo pip install poppy_humanoid
```

3.2 From the sources

The sources of Pyppot are hosted in the Poppy project Github.

To download them, you need to have git installed on your computer. Then, in the folder where you want to put Pypot, you do:

```
git clone --recursive https://github.com/poppy-project/pypot.git
```

Then, you have to install it (so Python can find it):

```
cd pypot
python setup.py build
python setup.py install
```

For the poppy-humanoid sources, follow the same procedure:

```
git clone --recursive https://github.com/poppy-project/poppy-humanoid.git
cd ppoppy-humanoid/software
python setup.py build
python setup.py install
```

3.3 Test your installation

In a Python console:

```
import pypot, poppy, poppy_humanoid
```

If one of the imports fails, try to reinstall, to check your PYTHONPATH or check the Pypot issues.

4 The v-rep Simulator

v-rep is a 3D robots simulator allowing virtual robots to act in a world with gravity, friction and collisions. The 3D model for Poppy is included in the poppy_humanoid package.

Be aware that V-rep asks for a fair amount of computation power, so use a decently recent computer.

4.1 Getting started with v-rep

First, install the v-rep software corresponding to your computer (v-rep download page). This software is free for people of the academical world and there is a free but limited version for other users.

Launch v-rep. On linux:

```
cd <path-to-vrep>
./vrep.sh
```

You should get an empty world with a floor and a tree structure of the elements of the world on the right.

You can explore the world by drag-and-dropping the simulated world and you can start/pause/stop the simulation with the up-right corresponding buttons. As v-rep uses a lots a computing power, it is advised to pause the simulation while your robot don't move.

4.2 Adding a Poppy

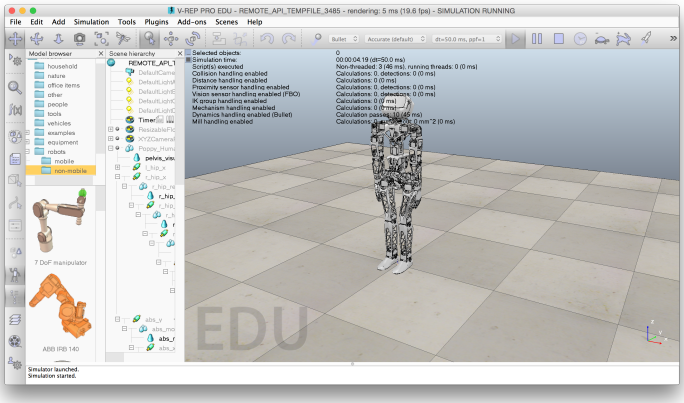
Open a Python console and type in:

```
from poppy_humanoid import PoppyHumanoid
poppy = PoppyHumanoid(simulator='vrep')
```

The first line imports the PoppyHumanoid object from the poppy_humanoid library. This object is a Poppy creature with the characteristics of a Poppy robot (see sections 6 and **??**).

The second line instanciates a PoppyHumanoid creature. It has the *simulator='vrep'* argument to know it should connect to the v-rep server and create a simulated v-rep robot.

In your v-rep world, you should now have a Poppy Humanoid robot.

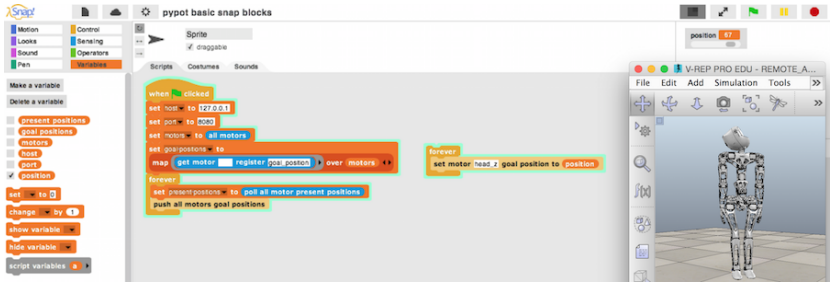


Now you can control the simulated Poppy robot as a real robot, as presented in section 6.

5 Visual programming with snap!

5.1 What is snap?

Snap! is a "very powerful visual, drag-and-drop programming language. It is an extended reimplementation of Scratch (a project of the Lifelong Kindergarten Group at the MIT Media Lab) that allows you to Build Your Own Blocks". It is an extremely efficient tool to learn how to program for kids or even college students and also a powerful prototyping method for artists.



Snap! is open-source and it is entirely written in javascript, you only need a browser connected to the Poppy Creature webserver. No installation is required on your computer!

WARNING! As there is no browser installed by default in Poppy Humanoid's head, you can't use Snap! directly on the robot. You will have to connect the USB2AX to your computer or to use a simulated robot (see V-rep, section 4).

5.2 Creating the server

The first step is to have a server running in the robot or in your computer. The Snap! page will then connect to this server to send orders to the robot.

5.2.1 On a real Poppy Humanoid robot

SSH inside the robot and enter the following command:

```
poppy-snap poppy-humanoid --no-browser
```

This will create a PoppyHumanoid object and launch a snap server. It will also return a url (typically `http://snap.berkeley.edu/snapsource/snap.html#open:http://<IP>:6969/snap-blocks.xml`).

Enter this address in your web browser.

5.2.2 On a simulated Poppy Humanoid robot

For the v-rep simulator, open a Python console and enter:

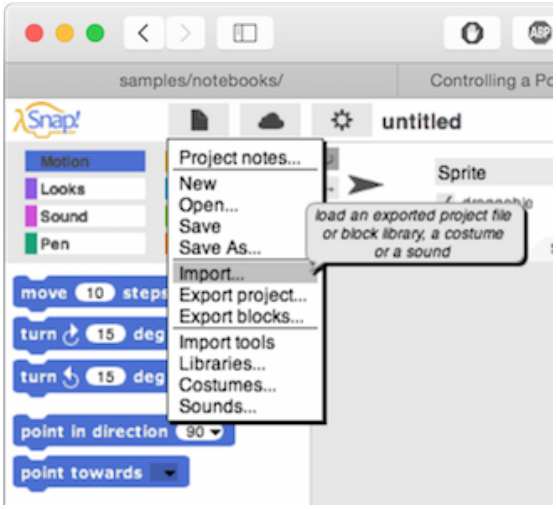
```
from poppy_humanoid import PoppyHumanoid
poppy = PoppyHumanoid(simulator='vrep', use_snap=True)
poppy.snap.run()
```

Then, open the following url in your web browser: `http://snap.berkeley.edu/snapsource/snap.html#open:http://127.0.0.1:6969/snap-blocks.xml`

5.2.3 Running Snap! locally

The previous paragraphes make you conect to the Berkeley server and use Snap! online. If you want to avoid this, you can install Snap! locally.

Then, in your web browser, open the snap.html file. Import the Poppy specific blocks:

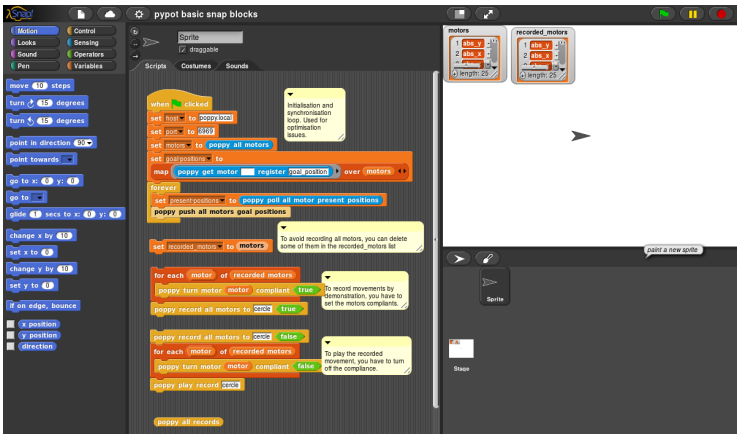


If you have Pypot installed from source, the file pypot-snap-blocks.xml is in the pypot/server folder. Otherwise, you can download only the file:

```
wget https://raw.githubusercontent.com/poppy-project/pypot/master/pypot/server/pypot-snap-blocks.xml
```

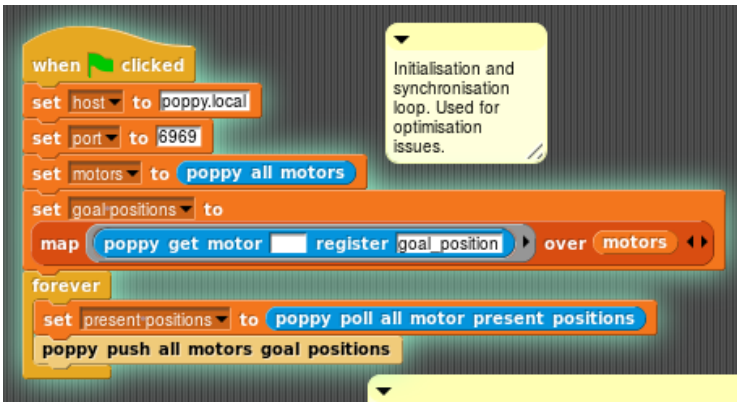
5.3 Snap! quick start

The webpage you opened should like like this:



5.3.1 Synchronization

The first thing to do is to launch the synchronization loop. Replace the hostname by poppy.local if you use a real Poppy Humanoid robot. Then click on the loop, it should stay highlighted.



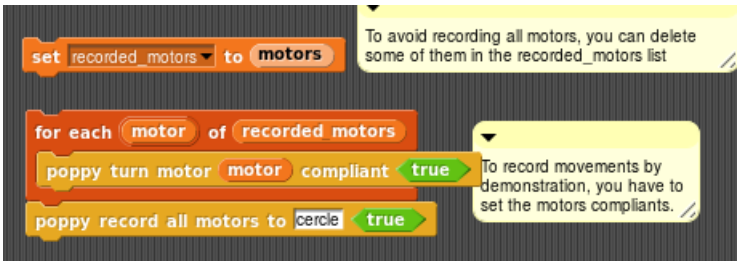
This piece of code first set some variables: host, port, motors and goal-position, the later being created by reading the goal position ('poppy get motor') of each motor present in the 'motors' array. You can check it by looking at the motors block top right:

Then we start a 'forever' loop that read the motors positions ('poppy poll all motors present positions') and store them in a present-positions array. Then it sends to the motors the content of the goal-position array.

This loop is the link between our Snap! code (we will read the present-positions array and modify the goal-positions array) and the real or simulated Poppy robot. You can stop the synchronization loop by clicking on it.

5.3.2 Poppy blocks

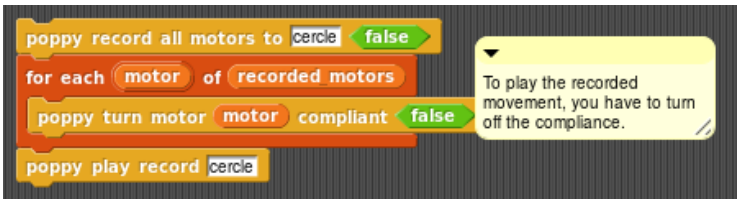
The default program shows you how to record a movement (called cercle by default):



The first line initializes an array containing all motors names. You need to run it only once.

The next block turns all the motors to compliant, so you can move the robot by hand, and starts recording.

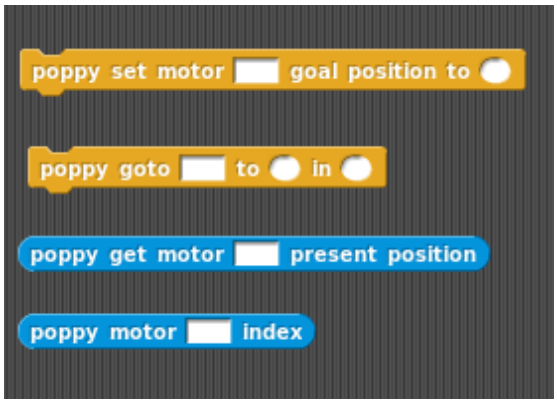
To stop recording, click on the next block:



This block stops the recording, then removes the compliance so that the robot can move by itself. Then, it starts playing the move you just recorded.

Warning, if you record a move with a name that has already been used, the new movement will be added after the previous one! This may lead to unexpected movements from the robot. Moves are recorded as .record files in the home folder of the Poppy robot.

Other Poppy specific blocks allow you to command each motor separately:



For other blocks, see the Snap! reference manual.

6 The Pypot library for Poppy

Pypot is a framework developed in the Inria FLOWERS team to make it easy and fast to control custom robots based on Dynamixel motors. This framework provides different level of abstraction corresponding to different types of use, i.e. direct control Robotis motors through a USB2AX, or high-level command of a robot defined by its particular structure.

Pypot has been entirely written in Python to allow for fast development, easy deployment and quick scripting by non-necessary expert developers. The serial communication is handled through the standard library and thus allows for rather high performance (10ms sensorimotor loop). It is crossed-platform and has been tested on Linux, Windows and Mac OS. It is distributed under the GPL V3 open source license.

Pypot is also compatible with the V-REP simulator. This allows you to seamlessly switch from a real robot to its simulated equivalent without having to modify your code.

Documentation can be found here.

6.1 PoppyHumanoid

Open a Python console inside the head of your PoppyHumanoid:

```
python
```

The first step to control a Poppy robot with Pypot is to create a Python object corresponding to your robot. We call this object poppy and define it as a PoppyHumanoid:

```
from poppy_humanoid import PoppyHumanoid
poppy = PoppyHumanoid()
```

This PoppyHumanoid object contains a list of motor objects that you can list with:

```
print poppy.motors
```

You get something like:

```
[<DxlMotor name=l_elbow_y id=44 pos=-0.0>,
 <DxlMotor name=r_elbow_y id=54 pos=-0.0>,
 <DxlMotor name=r_knee_y id=24 pos=0.2>,
 <DxlMotor name=head_y id=37 pos=-22.7>,
 <DxlMotor name=head_z id=36 pos=0.0>,
 ...
```

Each motor contain several fields, some static, like name and other directly linked to the corresponding Dynamixel register:

```
for m in poppy.motors:
  print "motor ",m.name
  print "  compliance: ",m.compliant,", position: ",m.present_position
```

You end up with:

```
motor  abs_y
  compliance:  True , position:  45.14
motor  abs_x
  compliance:  True , position:  -21.41
motor  abs_z
  compliance:  True , position:  24.22
...
```

Remember that the angular values are in degrees. When all motors are set to 0, the robot is standing with arms along the body.

You can also control each motor directly:

```
print "right ankle angle: ",poppy.r_ankle_y.present_position
```

6.2 Simple motor control

To control a motor, you first need to allow it to use its torque, i.e. to stop being compliant. Let test on the left-right head motor (head_z).

```
poppy.head_z.compliant = False
```

To prevent the robot from being dangerous, we will limit its maximum torque. You will be able to slightly push the motor with your hand, but it will go back to its goal position. Max torque is an integer value between 0 and 100, a proportion of the physical motor max torque (depends on the motor model).

```
poppy.head_z.max_torque=20
```

Then you have to set the goal position of the register. This operation is instantaneous and the servomotor will try to reach this position using a PID controller. But this is so quick that you will likely not see any delay.

```
poppy.head_z.goal_position=20
```

The communication with the Dynamixels itself takes some time and is done in parallel. So if you're using Python script, you need to add a bit of delay before the end, as the end of the script destroys you Poppy creature.

```
import time
time.sleep(0.1)
```

6.3 Timed movements

If you want a slower movement for your Dynamixel servomotor, you can use the goto_position function and setting a time for the movement.

```
poppy.head_z.goto_position(-20, 3)
```

The head should go to angle -20 in 3 seconds.

The wait optional argument allows you to decide if you want the script to return immediately or to wait for the movement completion:

```
print "before moving"
poppy.head_z.goto_position(40, 3, wait=True)
print "after first movement completion"
poppy.head_z.goto_position(-40, 3, wait=False)
print "during second movement"
```

7 Useful links

7.1 Forum and documentations

Poppy project website:	`https://www.poppy-project.org/`
Poppy project forum:	`https://forum.poppy-project.org/`
Pypot documentation:	`http://poppy-project.github.io/pypot/`
Github:	`https://github.com/poppy-project`
v-rep resources:	`http://www.coppeliarobotics.com/resources.html`

7.2 Other tutorials

Poppy in V-REP:

https://forum.poppy-project.org/t/howto-connect-pypot-to-your-simulated-version-of-a-poppy-humanoid-in-v-rep/332

Poppy and Snap!:

http://nbviewer.ipython.org/github/poppy-project/pypot/blob/master/samples/notebooks/Controlling%20a%20Poppy%20Creature%20using%20SNAP.ipynb

Introduction to Pypot:

http://nbviewer.ipython.org/github/poppy-project/pypot/blob/master/samples/notebooks/QuickStart%20playing%20with%20a%20PoppyErgo.ipynb

Record and save moves:

http://nbviewer.ipython.org/github/poppy-project/pypot/blob/master/samples/notebooks/Record%2C%20Save%2C%20and%20Play%20Moves%20on%20a%20Poppy%20Creature.ipynb