**POPPY-PROJECT ROBOT**

Montaje final, instalación software y uso

**Autor:** Daniel García Gonçalves (daniel.garcia.goncalves@udc.es)

**Propiedad:** Grupo Integrado de Ingeniería

## Número revisión: 0.2

**Fecha revisión:** 31/05/2017

Índice

1. [TRABAJOS PREVIOS A LA DOCUMENTACIÓN……………………………….……………...3](#_Toc508639500)

[2. HARDWARE 4](#_Toc508639501)

[2.1. ALIMENTACIÓN PLACA ODROID 4](#_Toc508639502)

[2.2. COMUNICACIÓN 5](#_Toc508639503)

[3. SOFTWARE 6](#_Toc508639504)

[3.1. DRIVERS 6](#_Toc508639505)

[3.2. ROBOPLUS 7](#_Toc508639506)

[3.3. ANACONDA 9](#_Toc508639507)

[3.4. BONJOUR SERVICE 10](#_Toc508639508)

[3.5. CLIENTE SSH: PuTTY 11](#_Toc508639509)

[3.6. PUESTA APUNTO ODROID XU4 13](#_Toc508639511)

[3.6.1. Método directo: copia imagen preconfigurada 13](#_Toc508639512)

[3.6.2. Método repositorio Git 15](#_Toc508639515)

[3.7. VREP 16](#_Toc508639516)

[4. USO 20](#_Toc508639518)

[4.1 MEDIANTE ORDENADOR DIRECTAMENTE 20](#_Toc508639519)

[4.2 MEDIANTE PLACA ODROID-PANTALLA 20](#_Toc508639520)

[4.3 MEDIANTE SSH 21](#_Toc508639521)

[4.4 MEDIANTE EXPLORADOR WEB 21](#_Toc508639522)

[4.1 MEDIANTE ORDENADOR DIRECTAMENTE 22](#_Toc508639523)

[4.2 MEDIANTE PLACA ODROID-PANTALLA 26](#_Toc508639524)

[4.3 MEDIANTE SSH 27](#_Toc508639525)

[4.3.1. Terminal 27](#_Toc508639526)

[4.3.2. Núcleo remoto 28](#_Toc508639527)

[4.4 MEDIANTE EXPLORADOR WEB 31](#_Toc508639528)

[4.4.1. Monitor and Control 33](#_Toc508639529)

[4.4.2. Build Your Own Blocks 34](#_Toc508639530)

[4.4.3. Python, terminal 34](#_Toc508639531)

[4.4.4. Configure the robot 37](#_Toc508639532)

[4.4.5. Reboot the robot 38](#_Toc508639533)

[4.4.6. What happened? 38](#_Toc508639534)

[4.4.7. Shutdown the Odroid 38](#_Toc508639535)

[5. PROGRAMACIÓN EN VREP 39](#_Toc508639537)

[6. COMANDOS PARA PROGRAMAR EL POPPY 41](#_Toc508639538)

[7. PROBLEMAS ENCONTRADOS Y FUTURAS LÍNEAS 43](#_Toc508639539)

[7.1. HARDWARE 43](#_Toc508639540)

[7.2. SOFTWARE 44](#_Toc508639541)

[7.3. USO 45](#_Toc508639542)

[8. LISTA ARCHIVOS 46](#_Toc508639543)

[9. REFERENCIAS Y ENLACES DE INTERÉS 47](#_Toc508639564)

[10. CAMBIOS REVISIÓN DOCUMENTO 50](#_Toc508639565)

# TRABAJOS PREVIOS A LA DOCUMENTACIÓN

El Grupo Integrado de Ingeniería dispone de un robot humanoide *open source* llamado Poppy. Al inicio de esta documentación, el robot se encontraba ensamblado prácticamente en su totalidad con todas las piezas y servos colocados. Solo faltaba el montaje de unos pocos cables (alimentación 12V con convertidor a 5V y tres cables del bus *Dynamixel)*, el montaje de la placa Odroid XU4 y la corrección de la orientación de dos piezas (los dos antebrazos).

Dicho lo anterior, este manual parte de la base que el montaje hasta ese punto se conoce adecuadamente por lo que todo lo escrito en él versará entorno a la finalización del montaje físico y la instalación del software necesario para el uso del robot así como ideas de uso. En todo caso el montaje está documentado bastante bien en el repositorio del proyecto ([https://github.com/poppy-](https://github.com/poppy-project/poppy-humanoid/blob/master/hardware/doc/en/assemblyGuide.md) [project/poppy-humanoid/blob/master/hardware/doc/en/assemblyGuide.md](https://github.com/poppy-project/poppy-humanoid/blob/master/hardware/doc/en/assemblyGuide.md))

# HARDWARE

### ALIMENTACIÓN PLACA ODROID

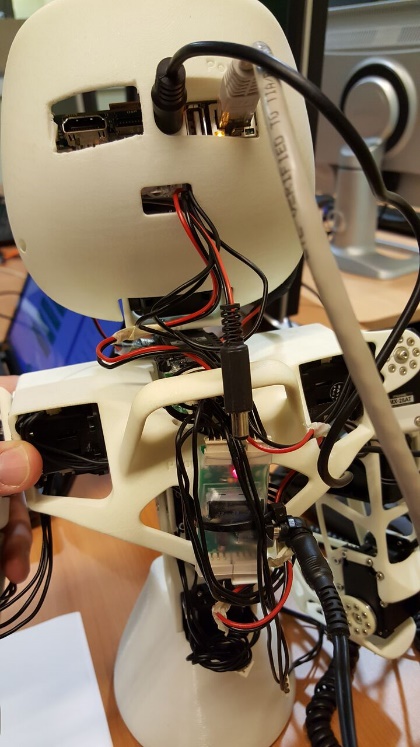
La alimentación a la placa Odroid se puede realizar, por comodidad de ahorrarse un cable, mediante un convertidor DC/DC que convierta los 12V de la alimentación de los servos a la tensión nominal de 5V de la Odroid. Esto se consigue conectando al conector de 4 pines del SMPS2Dynamixel de la parte superior un par de cables que lleven la tensión de 12V al convertidor.

El par de cables (rojo y negro) tiene una longitud aproximada de 30 cm. Tras pasar por el convertidor DC/DC[1](#_bookmark3) se suelda un conector DC de 5,5 mm de diámetro externo y 2,1 diámetro interno con polaridad positiva en el conector interno y negativa en el externo.

***Figura 1:***SMPS2Dynamixel parte superior. *Ejemplo conexión eléctrica de 12V. PINES: +-*

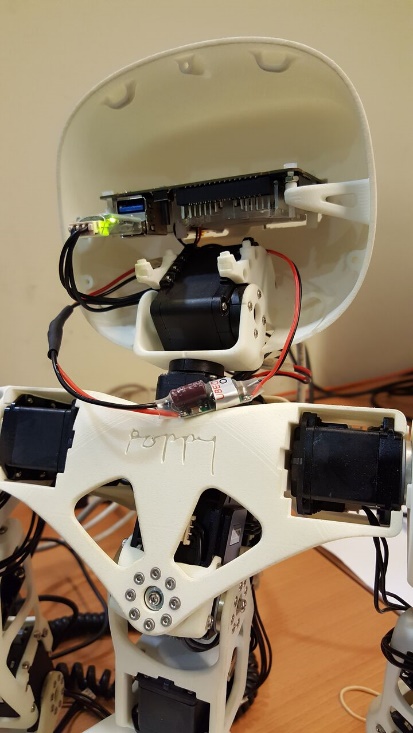
Con este montaje se consigue obviar el cable de alimentación de la Odroid. Cabe destacar que los dos SMPS2Dynamixel se alimentan con una sola fuente de alimentación (da igual cuál de los dos sea) que debe ser superior a 5A de 12V (véase SMPS 12V 5A PS-10[2](#_bookmark4) que se expone en la documentación del repositorio). El conector tiene un diámetro interno de 2,5 mm, 5,5 mm externo y polaridad positiva en el interno.

1. El propuesto en la documentación y el instalado es el siguiente: DC/DC UBEC DC/DC Step-Down (Buck) Converter - 5V @ 3A output (<https://www.adafruit.com/product/1385>)
2. <http://www.robotis.us/smps-12v-5a-ps-10-us-110v/>

En la imagen de la derecha se pueden ver las conexiones. En el círculo amarillo se encuentra conectado el cable Ethernet. En el círculo azul aparece el cable de alimentación de la Odroid. En verde aparece el cable de alimentación de los servos. Para ahorrarse un cable, como se comentaba anteriormente, se podría conectar el cable que aparece en el círculo rojo en el lugar del cable de alimentación de la Odroid (círculo azul).

***Figura 2:*** *Conexiones.*

### COMUNICACIÓN

La comunicación de los servos se realiza mediante dos USB2AX[3](#_bookmark6), una para la parte superior y otra para las piernas. Por lo tanto, cada parte necesita de se propio puerto. Por motivos de espacio, es necesario disponer de un HUB con un conector **USB reducido** que permita el cierre de la cabeza (véase [Cabeza](#_bookmark59)). En la imagen siguiente se puede ver la conexión del cable USB2AX de la cabeza.

***Figura 3****: Conexión del USB2AX.*

Para la instalación de los drivers véase [3.1. DRIVERS](#_bookmark8)*.*

1. http://www.xevelabs.com/doku.php?id=product:usb2ax:usb2ax

# SOFTWARE

### DRIVERS

Para el uso de Poppy desde USB directamente desde un ordenador Windows, es necesario, en primer lugar instalar los drivers del USB2AX. Esto también es necesario para realizar la configuración inicial, por lo que este paso es imprescindible. En Linux no es necesario realizar la instalación de ningún driver.

La instalación sigue el siguiente proceso:

1. Descargar el archivo de la siguiente dirección (es probable que se te abra como un archivo de texto en el navegador, por lo que deberás clickar el botón derecho, *guardar como...* y guardarlo sacándole la extensión .txt, dejando el nombre como **USB2AX.inf** simplemente):

*https://raw.github.com/Xevel/usb2ax/master/firmware/lufa\_usb2ax/USB2AX.inf*

Se anexa en la carpeta *\drivers\USB2AX.inf*

1. Introducir el USB2AX en el ordenador. Windows intentará buscar los controladores y no los encontrará. Por lo tanto se puede omitir la búsqueda en Windows Update clickando en el icono de instalación de controladores de la barra de tareas.
2. Ir al administrador de dispositivos de Windows (Panel de control>Hardware y sonido>Administrador de dispositivos).
3. Buscar el dispositivo y clickar con el botón derecho sobre él y darle *Actualizar controlador*. Buscar la carpeta del archivo descargado e instalarlo.
4. La luz del USB2AX deberá pasar de color rojo a verde.
5. Instalación completada y dispositivo listo para usarse.

### ROBOPLUS

Mediante este software, se configura las IDs de los servos por lo que no es necesario hacer este apartado si se tiene la seguridad de que la configuración y el funcionamiento de estos es correcto.

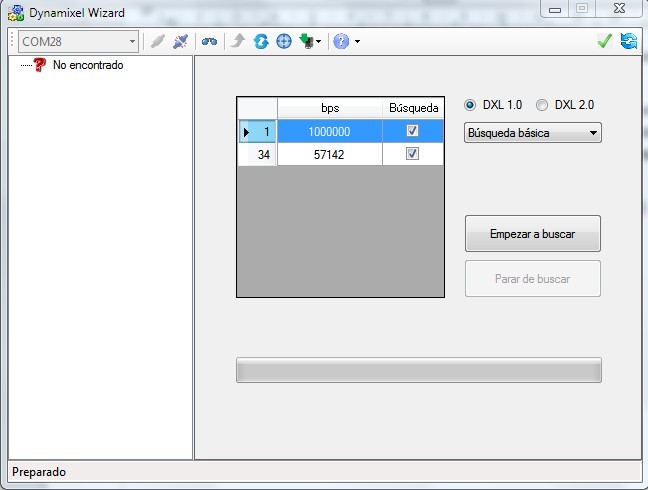
El software se puede encontrar en esta dirección:

[*http://en.robotis.com/BlueAD/board.php?*](http://en.robotis.com/BlueAD/board.php?bbs_id=downloads&amp;mode=view&amp;bbs_no=1132559&amp;page=2&amp;key&amp;keyword&amp;sort&amp;scate=SOFTWARE)[*bbs\_id=downloads&mode=view&bbs\_no=1132559&page=2&key=&keyword=&sort=&scate=SOFTWARE*](http://en.robotis.com/BlueAD/board.php?bbs_id=downloads&amp;mode=view&amp;bbs_no=1132559&amp;page=2&amp;key&amp;keyword&amp;sort&amp;scate=SOFTWARE)

Se anexa en la carpeta *\configuracion\RoboPlus(v1.1.3.0).exe*

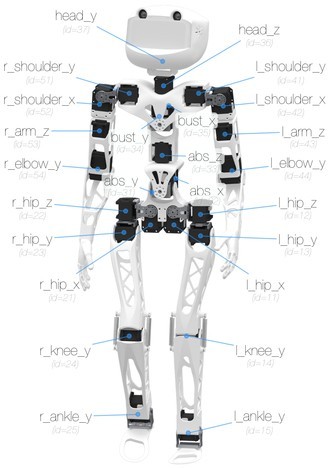
Tras la descarga, y con los drivers instalados tal como se explica en el apartado [3.1. DRIVERS](#_bookmark8) el proceso sería el siguiente:

1. Insertar el USB2AX en el puerto USB. Se tiene que poner verde al poco tiempo (puede tardar unos segundos). El color verde significa que está correctamente instalado y funcional. Si no fuera el caso, compruebe en el administrador de dispositivos su estado y/o extraiga y vuelva a insertar el dispositivo.
2. Conectar el servo al dispositivo USB con un conector Dynamixel de tres hilos al SMPS2Dynamixel y este a un servo. Es necesario esto último porque al principio todos tienen el mismo ID.
3. Alimentar el SMPS2Dynamixel mediante un transformador de 12V.
4. Abrir la aplicación RoboPlus.
5. En la pestaña Expert, abrir Dynamixel Wizard.
6. Seleccionar el puerto donde se encuentra el servo y pulsar abrir puerto.



***Figura 4:*** *Interfaz Dynamixel Wizard. Rodeado con un círculo selección de puerto y la flecha apuntando al botón abrir puerto.*

1. Dar a empezar a buscar.
2. Tras unos segundos aparecerá el dispositivo con el ID. Asignar el ID de acuerdo a las especificaciones de Poppy:
   * ID correspondiente a la siguiente imagen:



***Figura 5:*** *Nombres e IDs de los servos de Poppy Humanoid*

* + Velocidad media de transferencia a 1 000 000 bps
  + Retraso del tiempo de retorno a 0 ms en vez de 0.5 ms

1. Cuando se haga con todos esta configuración inicial al conectarlos todos en el bus se puede comprobar que el programa nos muestra todos los conectados y la información de estos.

Por último, como detalle final, hay que decir que este software permite la calibración de los servos, realizar la actualización del *firmware* o la recuperación de este.

### ANACONDA

Para el uso de Poppy tanto virtual[4](#_bookmark17) como mediante la conexión directa del PC usando los USB2AX es necesario instalar todas la librerías de Python necesarias y el entorno de Python correctamente configurado. Para ello la documentación recomienda usar la distribución de Python Anaconda.

Poppy está programado en Python 2.7 por lo que aunque puede funcionar mediante Python 3.3, está recomendado usar las distribuciones basadas en versiones de Python 2.7. En primer lugar se debe descargar Anaconda desde este enlace:

[*https://www.continuum.io/downloads#windows*](https://www.continuum.io/downloads#windows)(recodar que mejor la versión 2.7)

También se anexa en la carpeta: *\python\Anaconda2-4.3.1-Windows-x86\_64.exe*

Tras la descarga, realizar la instalación. Se recomienda instalarlo para todos los usuarios y es imprescindible que si se quiere usar este entorno y seguir lo indicado en estas líneas se marquen las dos casillas de Opciones Avanzadas para configurar el PATH de Python predeterminado.

Cualquier otra distribución como Canopy y Miniconda debería funcionar sin problemas también.

El siguiente paso es instalar los llamados poppy softwares. Para ello, abrimos cmd.exe (Símbolo del sistema de Windows) con privilegios de administrador (botón derecho y ejecutar como administrador). En esa pantalla tecleamos lo siguiente (cambiamos poppy-humanoid por poppy-torso o poppy-ergo-jr para instalar respectivamente a Poppy Toso o Poppy Ergo Jr.)[5](#_bookmark18):

*pip install poppy-humanoid*

Con este comando instalamos todos los paquetes necesarios para instalar poppy humanoide.

Hay que tener en cuenta que los comandos anteriores pueden no ser exactos del todo debido a cambios que se realicen en la librería del Poppy a lo largo del desarrollo del proyecto. Para conocer los pasos necesarios a seguir para la instalación correcta de los diferentes tipos de robots, dirigirse a:

*https://docs.poppy-project.org/en/installation/install-poppy-softwares.html*

1. No se habla de la simulación mediante V-REP porque sale fuera del área de este documento. Para más información léase la siguiente página: <https://docs.poppy-project.org/en/installation/install-vrep.html>
2. En la documentación pone **pip install poppy-torso --user -U** pero por algún motivo suele dar problemas por las actualizaciones desde la consola en vez desde el entorno de anaconda.

### BONJOUR SERVICE

Este servicio nos permite comunicarnos con miembros de una misma red LAN de una manera más sencilla sin necesidad de conocer la dirección IP y simplemente tecleando el nombre del dispositivo.

Bonjour es la implementación de Apple de las técnicas de configuración automática de red Zeroconf.

El cliente Zeroconf publica un nombre de dominio local con la siguiente estructura, siendo odroid el nombre del equipo:

*odroid.local*

Además, entre otras herramientas Zeroconf hay una implementación de DHCP descentralizado el cual permite a los ordenadores obtener una dirección IP y conectarse con otros sin un servidor DHCP. La dirección automáticamente obtenida se encuentra en el rango APIPA (de169.254.0.0 a 169.254.255.255). Se puede conectar directamente el robot al ordenador directamente con un cable Ethernet sin ningún router y con su dominio local[6](#_bookmark22).

La instalación en Windows se realiza, como se ha dicho, mediante la instalación del servicio Bonjour de Apple. Este se obtiene de la siguiente dirección:

*https://support.apple.com/kb/DL999?locale=es\_ES*

Se anexa en la siguiente carpeta: *\configuracion\BonjourPSSetup.exe*

Simplemente se instala y no se necesita realizar ninguna configuración adicional.

La instalación en Linux no se realiza mediante el servicio Bonjour de Apple y se tiene que hacer con los siguientes códigos:

*sudo apt-get install avahi-daemon avahi-autoipd* (en Debian/Ubuntu) *yum install avahi-daemon avahi-autoipd* (en Fedora)

1. Esto no está testado.

### CLIENTE SSH: PuTTY

Para poder comunicarse con otro ordenador desde Windows es necesario instalar un cliente SSH. Solo es necesario instalar un cliente SSH para tener flexibilidad en el uso de la placa Odroid y/o realizar instalaciones/configuraciones sin necesidad de pantalla aunque es muy recomendable tener uno. El elegido en este caso es PuTTY, uno de los recomendados en la documentación el cual se puede descargar desde la siguiente dirección:

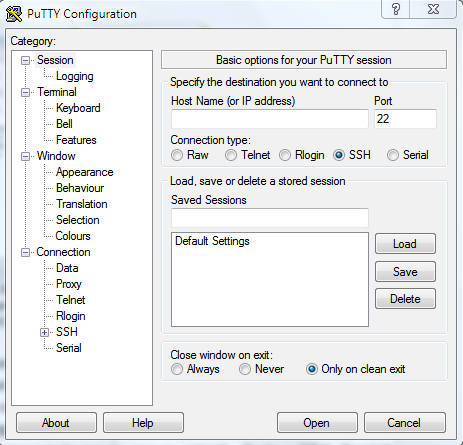
## [*http://www.putty.org/*](http://www.putty.org/)

Se anexa en la carpeta: *\configuracion\putty-64bit-0.68-installer.msi*

Ya instalado, se pueden hacer dos tareas básicas que en otros apartados veremos en detalle:

1. Conexión SSH:
   1. Se abre el programa.
   2. Se introduce la dirección IP del ordenador o su nombre si se ha hecho el paso indicado en [3.4. BONJOUR SERVICE](#_bookmark19)[7](#_bookmark26). En este caso como ya está configurado, se introduce en el nombre:

*Poppy.local*



***Figura 6:*** *Pantalla inicial PuTTY. Rodeado en rojo el lugar donde se introduce la IP o el nombre del ordenador.*

* 1. Se da a OPEN. En la primera conexión saldrá una pantalla explicando que si se quiere guardar la clave, conectarse solo una vez o cancelar. Esto es un protocolo de seguridad para intentar evitar caer en una conexión interceptada. El sistema de seguridad almacena la clave generada en el registro para las siguientes conexiones. Si esta no coincidiera podría estar la conexión interceptada por lo que se avisaría. Darle a YES. También funcionaría con NO pero el mensaje se volvería a repetir cada vez que te conectes y sin la seguridad adicional que proporciona la clave SSH generada.

1. poppy.local si se instaló poppy en la Odroid
   1. Listo! Seguir indicaciones (por ejemplo introducir nombre de usuario).

Nombre de usuario: *poppy*

Contraseña: *poppy*

1. Copia archivos a distancia:
   1. Abrir el terminal de Windows como administrador.
   2. Teclear el siguiente código:

*pscp user@host:source target*

*user=usuario del host (Ej. poppy)*

*host=dirección del host (Ej. Poppy.local)*

*source=ruta origen de archivo*

*target= ruta destino copia de archivo*

### PUESTA APUNTO ODROID XU4

Existen dos formas de instalar todo lo necesario para el manejo del humanoide desde la placa Odroid XU4 (o cualquier otra): mediante una copia de una imagen preconfigurada y mediante la descarga desde el repositorio.

## Método directo: copia imagen preconfigurada

Mediante este método se carga una imagen preconfigurada que puede no estar actualizada con las últimas librerías.

Lo primero para realizar este paso, como siempre, es realizar la descarga de la imagen. Se descarga la imagen correspondiente indicada en *https://docs.poppy-project.org/en/installation/burn-* [*an-image-file.html*](https://docs.poppy-project.org/en/installation/burn-an-image-file.html). Para el caso de Poppy Humanoid en Odroid XU4 desde esta dirección:

## [*https://github.com/poppy-project/poppy-humanoid/releases/*](https://github.com/poppy-project/poppy-humanoid/releases/)

Se anexa en la siguiente carpeta: *\humanoid\_xu4\poppy-humanoid-xu4-2016-05-26.img*

Descomprimir el archivo, si lo estuviera, y copiar la imagen a la SD.

Para copiar la imagen a la SD se puede usar una aplicación muy sencilla llamada Etcher que se puede descargar desde la siguiente dirección:

## [*https://etcher.io/#downloads*](https://etcher.io/#downloads)

Se anexa en la siguiente carpeta: *\humanoid\_xu4\Etcher-1.0.0-beta.19-win32-x64.exe*

El manejo de este programa es muy simple. Tras la instalación y apertura del mismo, se selecciona la imagen, se selecciona la unidad y se da a flashear.



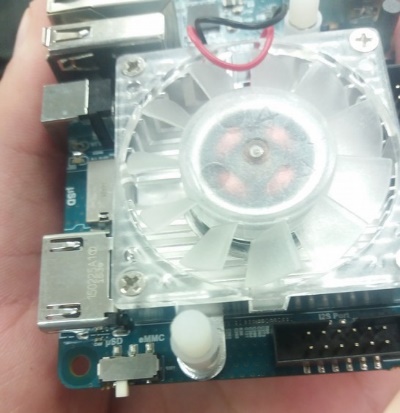
***Figura 7:*** *Interfaz Etcher*

Tras unos minutos de copiado, se extrae la unidad del lector de microSD y se separa la memoria eMMC del adaptador microSD y se introduce la memoria en su lugar correspondiente fijándose a continuación que el selector de eMMC esté seleccionado.



***Figura 9:*** *Señalada con una flecha la posición para seleccionar la memoria eMMC.*

***Figura 8:*** *Rodeado con un círculo el lugar donde se introduce la eMMC*



***Figura 10:*** *eMMC en adaptadora microSD*

Llegados a este punto, ya se puede pasar al manejo de la placa en el apartado [4. USO](#_bookmark35)

## Método repositorio Git

La memoria que se incluye suele traer por defecto una versión de Ubuntu (si no la trajera, habría que realizar el apartado [3.6.1. Método directo: copia imagen preconfigurada](#_bookmark28)pero descargando una versión de Ubuntu como la del siguiente enlace *http://odroid.in/ubuntu\_14.04lts/ubuntu-14.04.1lts-lubuntu-odroid-xu3-20150212.img.xz* en vez de la que se indica en dicho apartado).

Antes de continuar, se debe tener un cliente SSH con el que poder comunicarse con la placa Odroid[8](#_bookmark34). Si no se tuviera ninguna aplicación ir al apartado [3.5. CLIENTE SSH: PuTTY](#_bookmark23)antes de continuar. Durante todo el proceso no desconectar ni apagar la Odroid.

1. Iniciar el servicio SSH (si se usa PuTTY abrirlo). La dirección original de la placa Odroid es

**odroid.local**. El nombre de usuario es **odroid** y contraseña **odroid**.

1. Teclear el siguiente código. Remplazar poppy-humanoid por poppy-torso si es el Robot Poppy-Torso:

*wget* [*https://raw.githubusercontent.com/poppy-project/odroid-poppysetup/master/poppy\_setup.sh -O poppy\_setup.sh*](https://raw.githubusercontent.com/poppy-project/odroid-poppysetup/master/poppy_setup.sh%20-O%20poppy_setup.sh)

*sudo bash poppy\_setup.sh poppy-humanoid*

1. Después de unos minutos, la tarjeta se reiniciará y se perderá la conexión con el SSH. Tras el reinicio, se puede conectar al nuevo usuario y seguir el proceso de instalación (dirección=Poppy.local; usuario=poppy; constraseña=poppy). Al ser una compilación de paquetes de Python pesados, el proceso puede llevar horas, un proceso largo y susceptible de fallos. Puedes salir del terminal con **ctrl+C** y ver todo el proceso leyendo el archivo install\_log (teclea para ello **tail -f install\_log**).
2. Tras mucha paciencia, la placa se reiniciará de nuevo. Las últimas líneas del proceso serán:

*System install complete!*

*Please share your experiences with the community : https://forum.poppy-project.org/*

1. Ya se puede ir al apartado [4. USO](#_bookmark35)*.*

Con este método se garantiza la imagen más actualizada de Poppy pero es un proceso mucho más lento que el de la imagen preconfigurada.

1. Quizás también se pueda realizar mediante una pantalla, teclado y ratón conectados a la propia placa y desde el propio terminal de Linux. No se ha realizado la prueba.

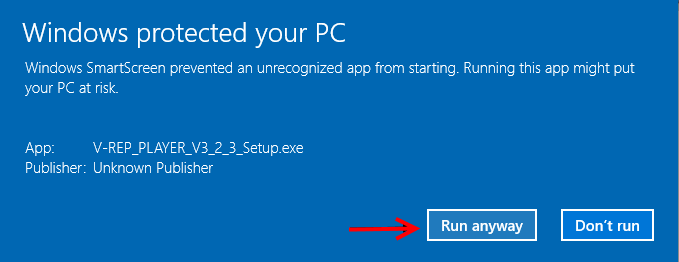
### VREP

V-REP es un simulador robótico principalmente Open Source (GNU GPL), que es distribuido con licencia gratuita para instituciones educativas y tiene licencia comercial para fines lucrativos. Hay además una versión PRO EVAL que tiene un soporte limitado. Como el proyecto Poppy es un proyecto OpenSource dedicado a investigación y al trabajo en universidad, se recomienda la descarga de la versión educativa.

Para instalar V-REP en Windows accederemos primero al enlace de descarga:

*http://www.coppeliarobotics.com/downloads.html*

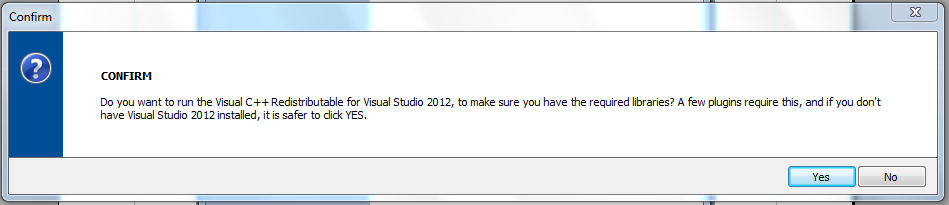
Como V-REP no está registrado, ignoraremos el aviso de Windows y comenzaremos con la instalación.



***Figura 11:*** *Mensaje de Windows. Seleccionar “Ejecutar”.*

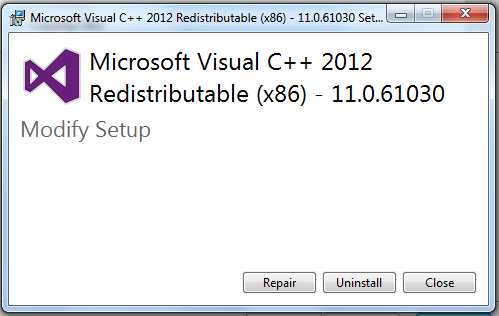
Durante la instalación, será necesario instalar *Visual C++ Redistributable 2010* y *Visual C++ Redistributable 2012*.





***Figura 12:*** *Mensaje de si se desea instalar las versiones de Visual C++ Redistributable.*

Incluso si ya se tienen estas versiones o superiores es recomendable “repararlas” (es un proceso de reinstalación) si aparece el mensaje para hacerlo.



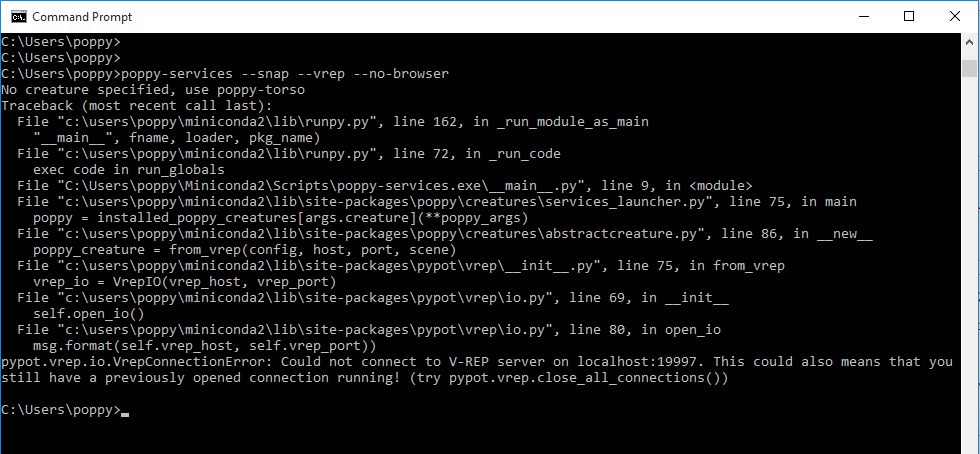
***Figura 13:*** *Mensaje para la reparación de Microsoft Visual C++ Redistributable.*

Tras la instalación puede comprobarse si V-REP se instaló correctamente siguiendo los siguientes pasos:

1. Abrimos V-REP.
2. Abrimos el prompt de la distribución de Python (si se instaló Anaconda se puede hacer tecleando en el buscador de Windows *Anaconda Prompt*) o de Windows (el símbolo del sistema usado anteriormente, *cmd* en el buscador de Windows) el siguiente comando y presionando *Enter* para ejecutarlo.

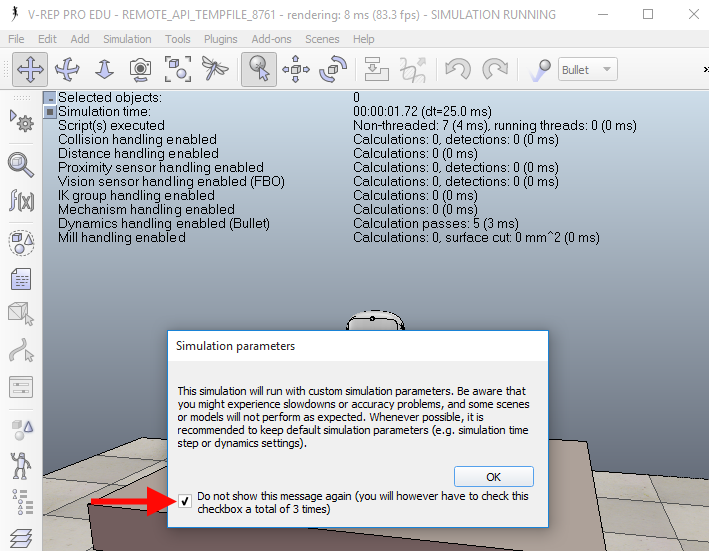
*poppy-services --snap --vrep --no-browser poppy-torso*

1. Después de unos segundos, debería verse un mensaje como el siguiente en la ventana de comandos



***Figura 14:*** *Comandos insertados en la consola de Windows.*

1. Si cambiamos a la ventana de V-REP, un mensaje aparece informando de que la simulación utiliza parámetros personalizados. Este mensaje bloquea la comunicación de V-REP con la API de Python. Marcamos en el cuadro: “*Do not show this message again*” y presionamos “*OK*”.



***Figura 15:*** *Ventana de VREP.*

1. Cambiamos a la ventana de comandos donde ejecutaremos el siguiente comando

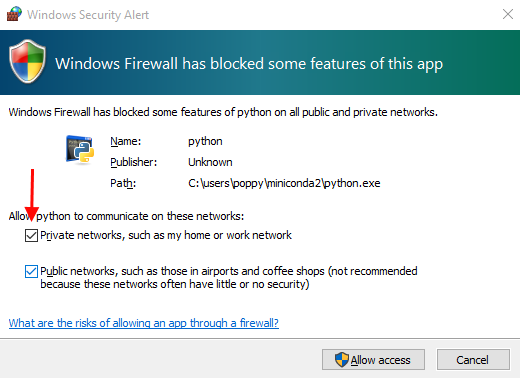
*poppy-services --snap --vrep --no-browser poppy-torso*

y clicamos nuevamente en la ventana de V-REP (con el cuadro del cuadro de diálogo marcado).

1. Este proceso habrá que hacerlo tres veces para asegurarse de que todo funciona bien. Para evitar reescribir el mismo comando de nuevo, se puede presionar la tecla ↑ para llamar a la última línea escrita.
2. Cuando la puesta a punto de V-REP esté lista, podemos ejecutar el comando sin la parte *--no-browser* .

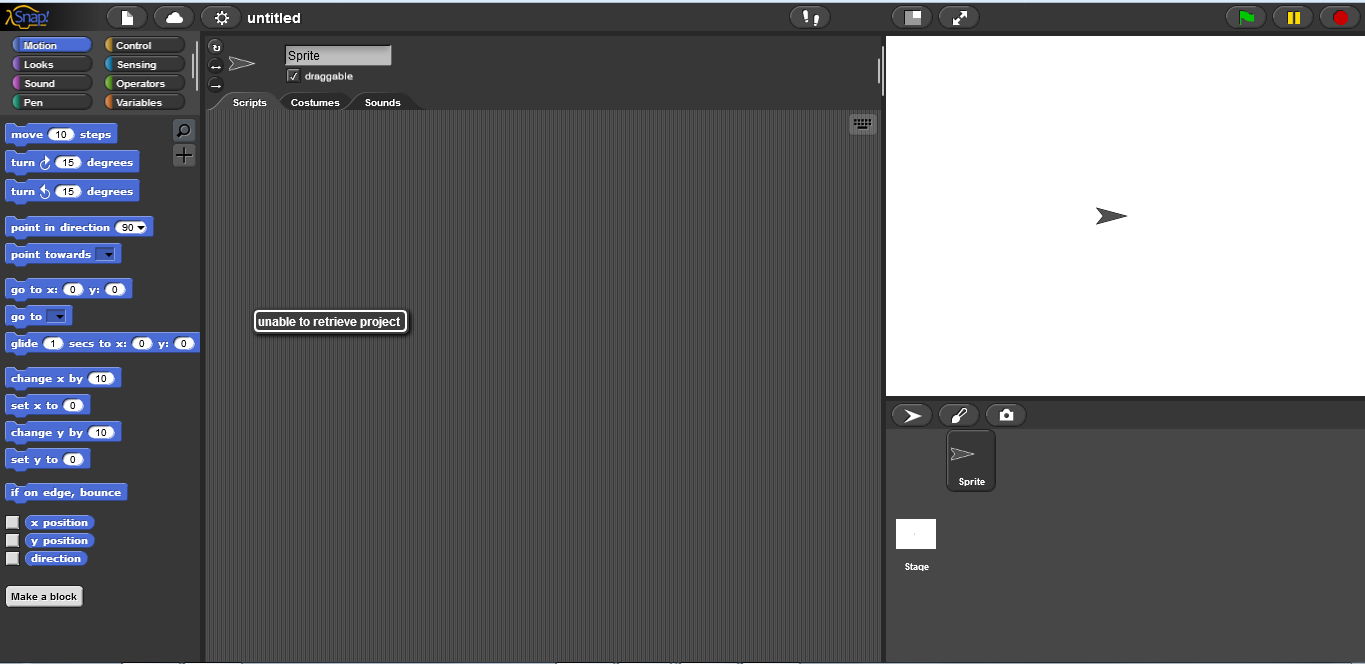
*poppy-services --snap --vrep poppy-torso*

1. Vemos un aviso del Firewall como en la imagen, marcamos el cuadro de *Private network*.



***Figura 16:*** *Mensaje del Firewall de Windows.*

1. Si todo funcionó correctamente veremos que se abre una nueva pestaña en el navegador web que tenemos por defecto, tal y como se ve en la imagen de a continuación. Podemos programar el robot en Snap! O en Python.



### Figura 17: Ventana de Snap.

# USO

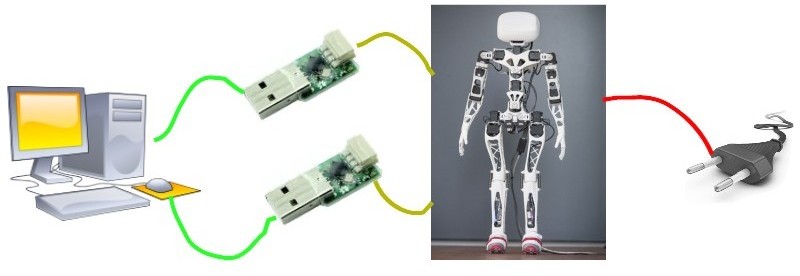
El manejo de Poppy puede realizarse de múltiples formas. Todas con sus ventajas e inconvenientes. En las siguientes imágenes se resume un poco algunas de las diferentes formas de interaccionar con el robot.

Cabe recordar que predeterminadamente la placa está configurada con los siguientes datos:

* + Dirección LAN: Poppy.local
  + Nombre de usuario: poppy
  + Contraseña: poppy

En apartados posteriores se entrará en detalle de como funciona cada método. Para acceder a dichos apartados, puede clickarse directamente en las imágenes correspondientes.

## MEDIANTE ORDENADOR DIRECTAMENTE



[Figura 18: Ordenador controla directamente servos Robot](#_bookmark36)

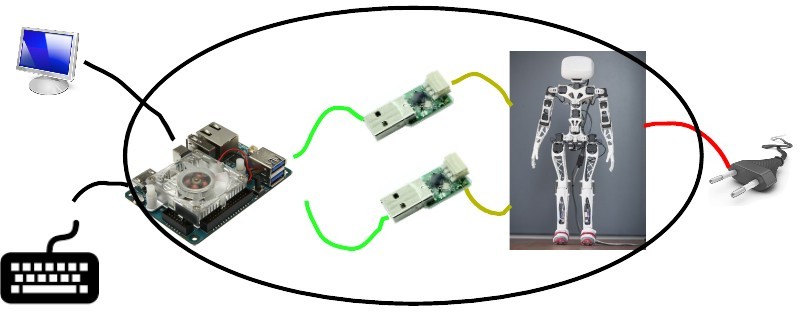
* [Breve descripción: se conecta los USB del bus Dynamixel al ordenador y desde este se controla.](#_bookmark36)
* [Ventajas: simplicidad en la configuración.](#_bookmark36)

[Principalmente para desarrollo, ventaja que puede ser superada. Véase](#_bookmark36) [E](#_bookmark66)[jecutar](#_bookmark36)[D](#_bookmark66)[ebugger desde PC a](#_bookmark36) Odroid.

[Potencia de cálculo.](#_bookmark36)

* [Desventajas: mayor número de cables e incomodidad manejo robot. Alargar bus.](#_bookmark36)

## [MEDIANTE PLACA ODROID-PANTALLA](#_bookmark39)



[Figura 19: Odroid controla directamente servos Robot mediante el uso de pantalla y periféricos. El conjunto del robot dentro del círculo.](#_bookmark39)

[Breve descripción: se conecta a la placa Odroid la pantalla y los periféricos.](#_bookmark39)

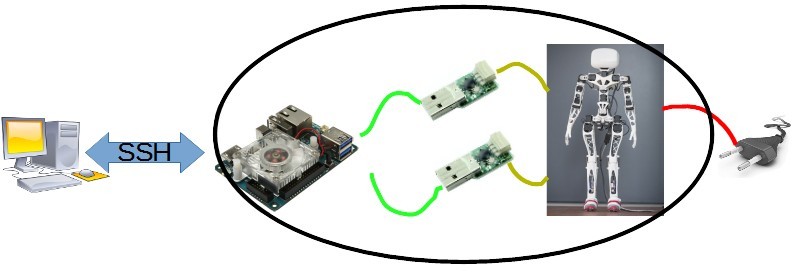
* [Ventajas: no necesitas ordenador externo y](#_bookmark39)

[simplicidad en la configuración. Principalmente para desarrollo, ventaja que puede ser superada. Véase](#_bookmark39) [E](#_bookmark66)[jecutar Debugger](#_bookmark39)[d](#_bookmark66)[esde PC a](#_bookmark39) Odroid.

* [Desventajas: cabeza no preparada para aplicar](#_bookmark39)

[este método (no dispone de las salidas necesarias para los cables). La potencia no es tan elevada como un PC corriente. Además dispone de cables que pueden ser molestos.](#_bookmark39)

## [MEDIANTE SSH](#_bookmark41)



[Figura 20: Mediante un ordenador usando SSH se controla la Odroid para esta controlar los servos del Robot. El conjunto del robot dentro del círculo.](#_bookmark41)

[Breve descripción: la placa Odroid incluida en el robot se controla mediante un protocolo SSH desde otro PC. Mediante un cable ethernet o incluso WIFI.](#_bookmark41)

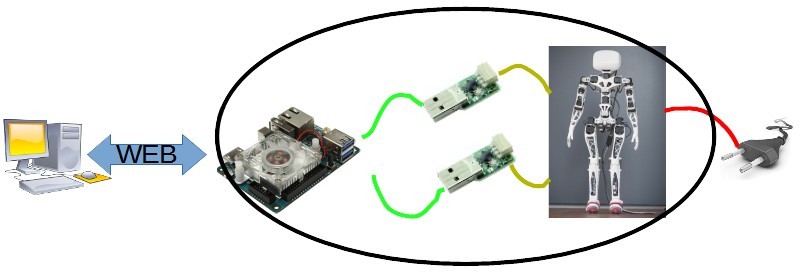
* [Ventajas: se reduce el número de cables que](#_bookmark41)

[tiene el robot y se puede ganar tanto en comodidad de desarrollo como potencia.](#_bookmark41)

* [Desventajas: Hay que intentar superar ciertas](#_bookmark41)

[dificultades que aumentarían enormemente la comodidad. Véase](#_bookmark41) [E](#_bookmark66)[jecutar Debugger](#_bookmark41) desde[P](#_bookmark66)[**C a**](#_bookmark41) **Odroid**.

## [MEDIANTE EXPLORADOR WEB](#_bookmark45)



[Figura 21: Mediante un ordenador usando WEB se controla la Odroid para esta controlar los servos del Robot. El conjunto del robot dentro del círculo.](#_bookmark45)

[Breve descripción: la placa Odroid incluida en el robot se controla mediante un protocolo WEB desde otro PC. Mediante un cable ethernet o incluso WIFI.](#_bookmark45)

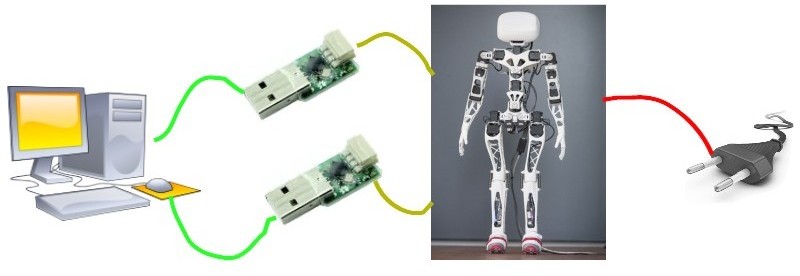
* [Ventajas: sistema implementado ya en el Robot](#_bookmark45)

[y susceptible de ampliar características. Se reduce el número de cables que tiene el robot](#_bookmark45)

* [Desventajas: el sistema actualmente está](#_bookmark45)

[bastante limitado. Gran dificultad actualmente en el uso no final (desarrollo de aplicaciones).](#_bookmark45)

### MEDIANTE ORDENADOR DIRECTAMENTE



[Figura 22: Ordenador controla directamente servos Robot](#_bookmark36)

[Breve descripción: se conecta los USB del bus Dynamixel al ordenador y desde este se controla.](#_bookmark36)

* + - [Ventajas: simplicidad en la configuración.](#_bookmark36)

[Principalmente para desarrollo, ventaja que puede ser superada. Véase](#_bookmark36) [E](#_bookmark66)[jecutar](#_bookmark36)[D](#_bookmark66)[ebugger desde PC a](#_bookmark36) Odroid.

[Potencia de cálculo.](#_bookmark36)

* + - [Desventajas: mayor número de cables e incomodidad manejo robot. Alargar bus.](#_bookmark36)

Este método es el más sencillo, empleado para comunicarse directamente con los motores del Poppy, puenteando la ODROID y no usando las características que ésta nos ofrece.

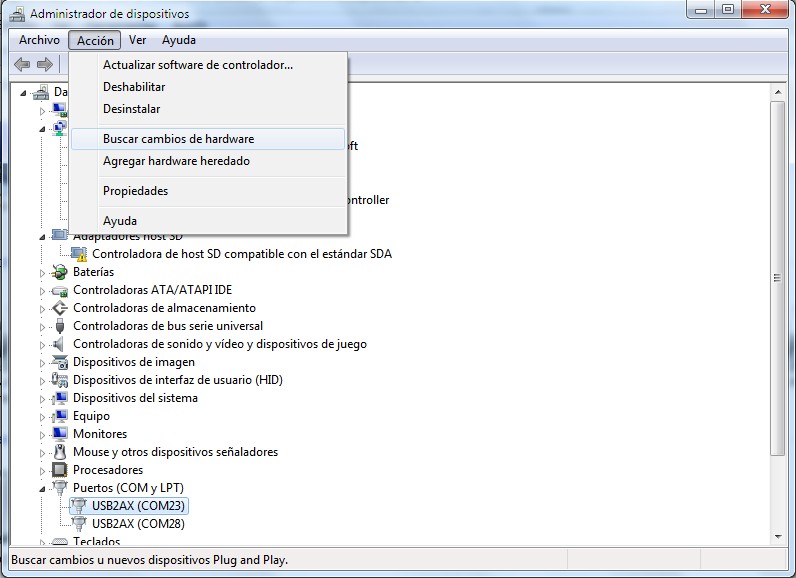
PASOS NECESARIOS:

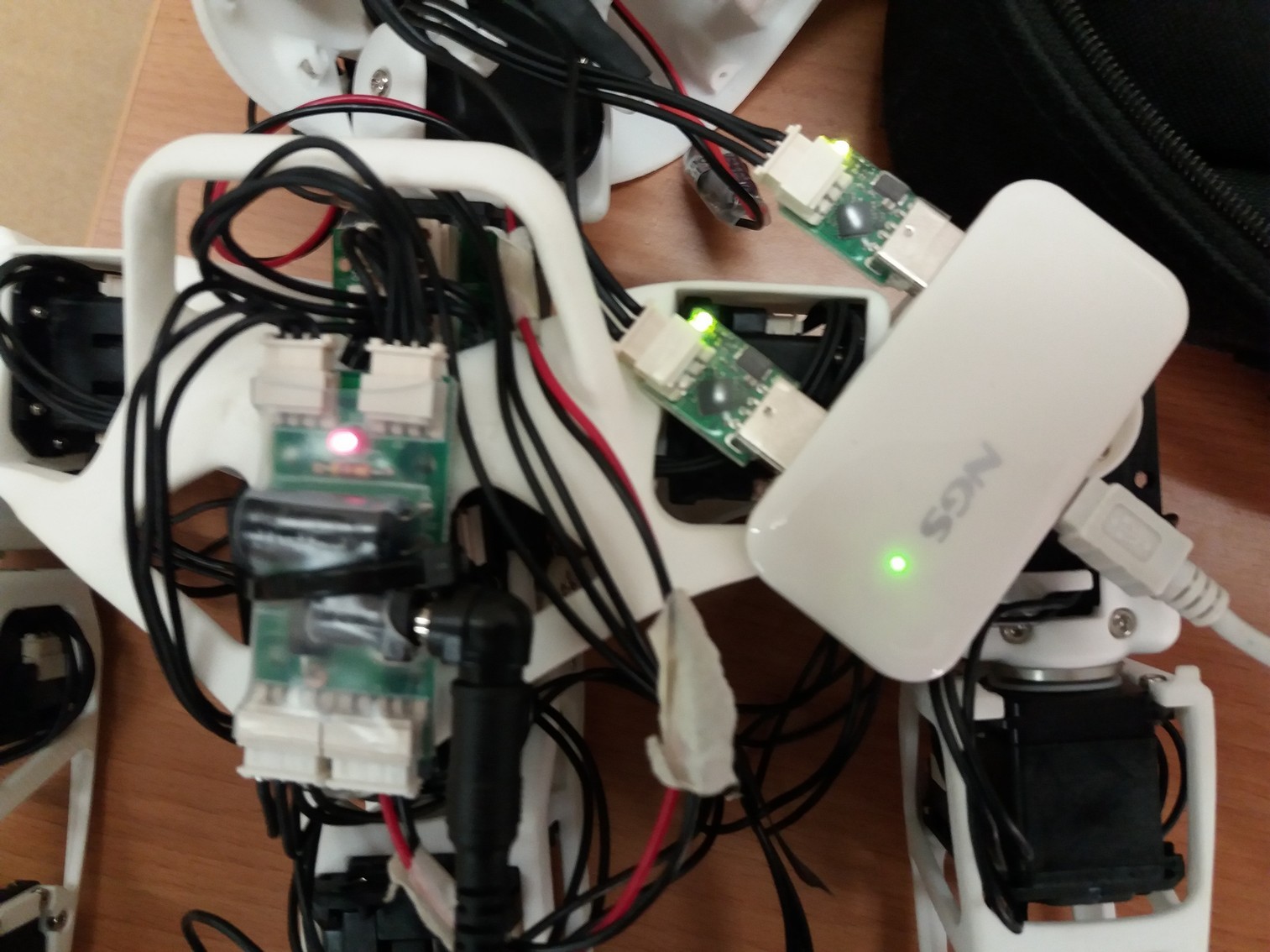
1. Tener instalados los drivers correctamente: [3.1. DRIVERS](#_bookmark8).
2. Conectar los USB2AX en los puertos USB de la computadora.
3. Alimentar el bus Dynamixell con los 12V.
4. ¡Listo para abrir la consola de Python!

EJEMPLO DE USO:



***Figura 23:*** *Se conectan los USB2AX. Después de esperar un poco continúa en rojo por lo que debemos asegurarnos de la instalación del dispositivo.*

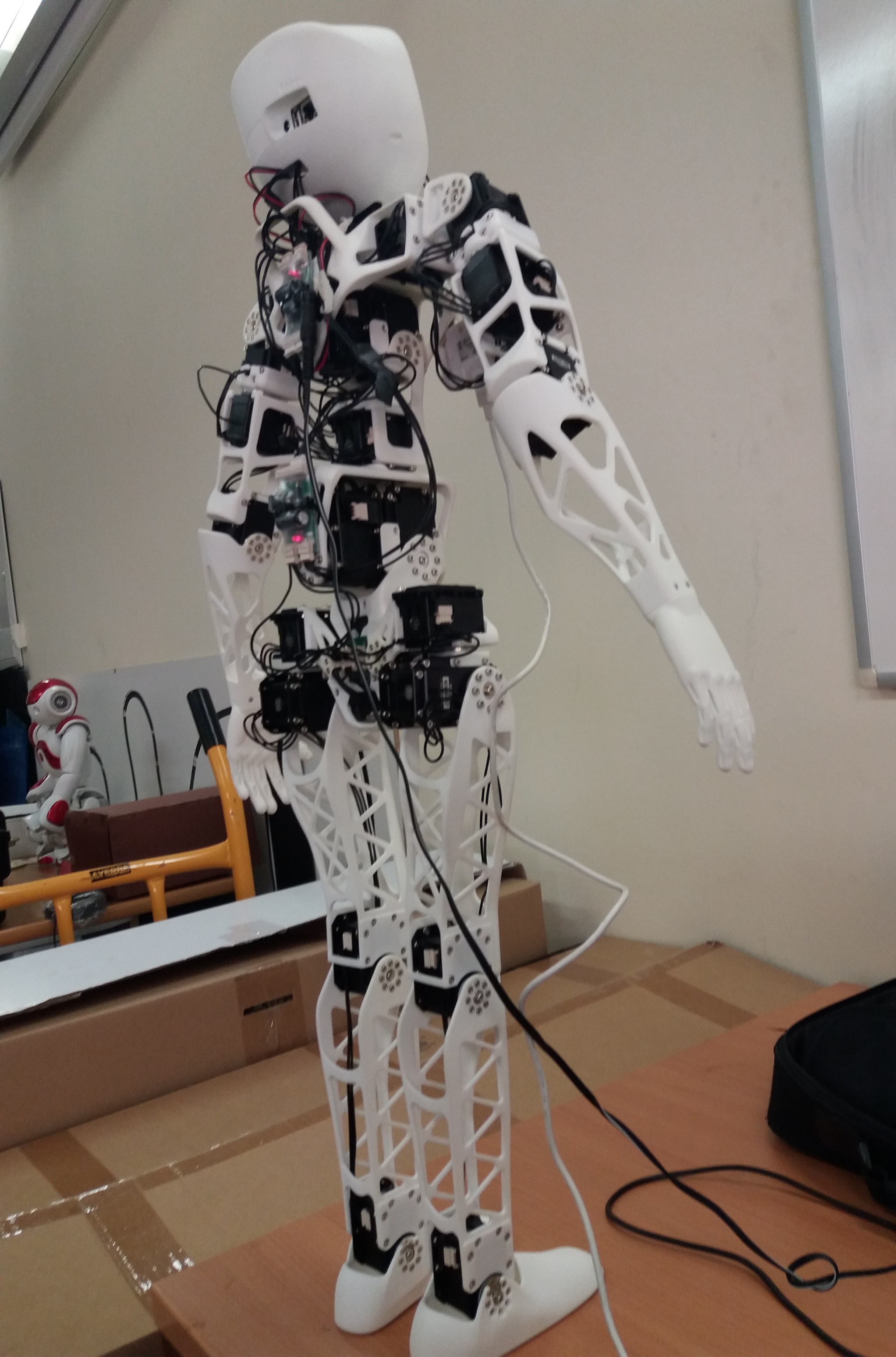
***Figura 24:*** *Forzamos a que encuentre los dispositivos yendo a Administrador de dispositivos, Acción, Buscar cambios de hardware. Esto lo hacemos debido a que estaba tardando mucho en encontrar los puertos y podemos así ver porque no los detecta.*

**

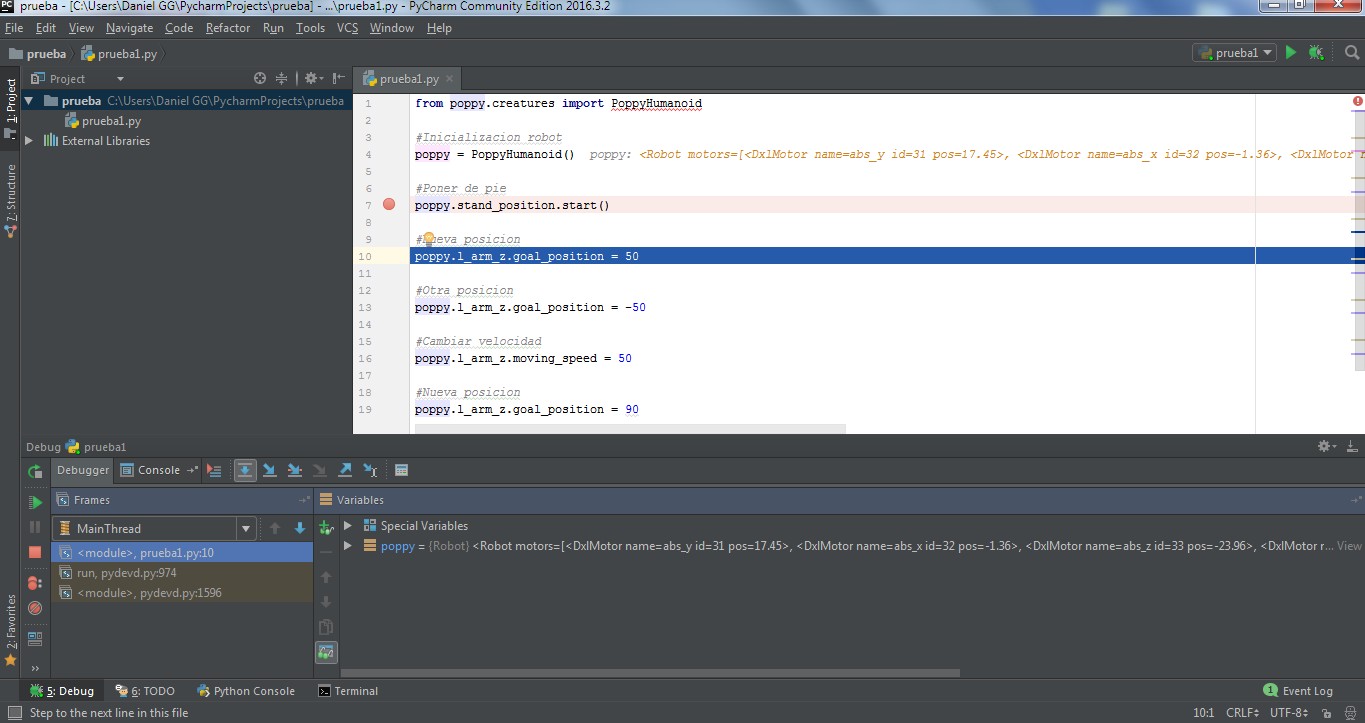
***Figura 25:*** *El LED de los USB2AX se pone en verde y alimentamos el* SMPS2Dynamixel



***Figura 26:*** *Se abre un IDE de programación con el código y se da al debugg. En este caso se pone el breakpoint en la línea 7, después de cargar el robot y antes de ponerse de pie.*

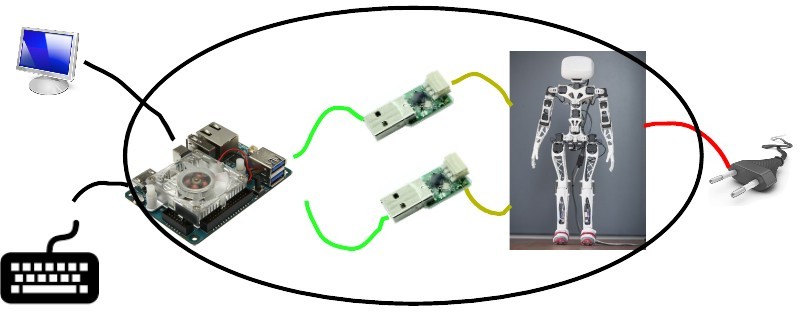
**

***Figura 27:*** *Inmediatamente el robot se pone de pie.*



***Figura 28:*** *Continuamos con el proceso de d**epurado y vamos viendo como se va moviendo paso a paso. Ver vídeo: Debugger en la carpeta* ***\videos\debugger.mp4***

### MEDIANTE PLACA ODROID-PANTALLA



[Figura 29: Odroid controla directamente servos Robot mediante el uso de pantalla y periféricos. El conjunto del robot dentro del círculo.](#_bookmark39)

[Breve descripción: se conecta a la placa Odroid la pantalla y los periféricos.](#_bookmark39)

* + - [Ventajas: no necesitas ordenador externo y](#_bookmark39)

[simplicidad en la configuración. Principalmente para desarrollo, ventaja que puede ser superada. Véase](#_bookmark39) [E](#_bookmark66)[jecutar Debugger](#_bookmark39)[d](#_bookmark66)[esde PC a](#_bookmark39) Odroid.

* + - [Desventajas: cabeza no preparada para aplicar](#_bookmark39)

[este método (no dispone de las salidas necesarias para los cables). La potencia no es tan elevada como un PC corriente. Además dispone de cables que pueden ser molestos.](#_bookmark39)

Este método será el método principal de conexión cuando queramos actuar directamente sobre la ODROID. Nos permite trabajar en la ODROID como si ésta fuese un PC y realizar tareas como la instalación de los componentes antes mencionados (Anaconda, Jupyter, etc) y la descarga e instalación de nuevos elementos que queramos añadir a nuestro programa en Python. Por ejemplo, será el método empleado para instalar en la ODROID librerías como:

* OpenCV.
* La librería de DEAP.
* NEAT.
* Etc.

Recordar que este sistema nos permite acceder a las dos cuentas que por defecto se encuentran en el Poppy:

Cuenta 01:

Nombre: odroid

Pass: odroid

Cuenta 02:

Nombre: poppy

Pass: poppy

PASOS NECESARIOS:

1. Disponer de un rediseño de la cabeza: solución a problemas detectados, por lo que se puede aprovechar el rediseño (véase [Cabeza](#_bookmark59))
2. Tener correctamente instalada la placa Odroid: [3.6. PUESTA APUNTO ODROID XU4](#_bookmark27).
3. Disponer de pantalla HDMI y periféricos USB.
4. Alimentar el bus Dynamixell con los 12V.

EJEMPLO DE USO:

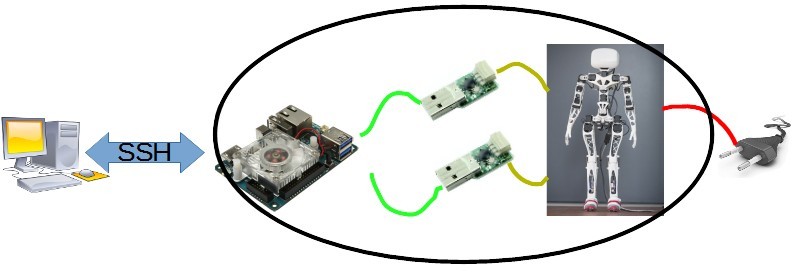
Falta pantalla pero en principio sería semejante al caso [4.1 MEDIANTE ORDENADOR](#_bookmark36)[DIRECTAMENTE](#_bookmark36) pero en Ubuntu

### MEDIANTE SSH

[Figura 30: Mediante un ordenador usando SSH se controla la Odroid para esta controlar los servos del Robot. El conjunto del robot dentro del círculo.](#_bookmark41)

[Breve descripción: la placa Odroid incluida en el robot se controla mediante un protocolo SSH desde otro PC. Mediante un cable ethernet o incluso WIFI.](#_bookmark41)

* + - [Ventajas: se reduce el número de cables que](#_bookmark41)



[tiene el robot y se puede ganar tanto en comodidad de desarrollo como potencia.](#_bookmark41)

* + - [Desventajas: Hay que intentar superar ciertas](#_bookmark41)

[dificultades que aumentarían enormemente la comodidad. Véase](#_bookmark41) [E](#_bookmark66)[jecutar Debugger](#_bookmark41) desde[P](#_bookmark66)[**C a**](#_bookmark41) **Odroid**.

Será el tipo de conexión que se empleará con mayor frecuencia. Permite controlar los procesos que se ejecutan en el Poppy a través de un PC normal y corriente. Desde el PC se ejecutan, lanzan y controlan los procesos de la ODROID.

PASOS NECESARIOS:

1. Tener correctamente instalada la placa Odroid: [3.6. PUESTA APUNTO ODROID XU4](#_bookmark27).
2. Tener correctamente instalado el servicio Bonjour: [3.4. BONJOUR SERVICE](#_bookmark19)
3. Tener instalado un servicio SSH y SCP: [3.5. CLIENTE SSH: PuTTY](#_bookmark23)
4. Conectar la Odroid a la misma LAN que el PC desde que se va a acceder al servicio SSH.
5. Alimentar el bus Dynamixell con los 12V.

EJEMPLO DE USO:

Podemos usar este esquema para realizar un control de la Odroid absoluto usando la consola de Linux a distancia. Usando la conexión SSH, por tanto, tenemos control muy grande de la placa lo que permite mucha flexibilidad y posibilidades. A continuación se muestran dos usos que se han explorado hasta el momento al que se le puede dar a la conexión SSH: Terminal y como conexión a un núcleo remoto de Python.

## Terminal

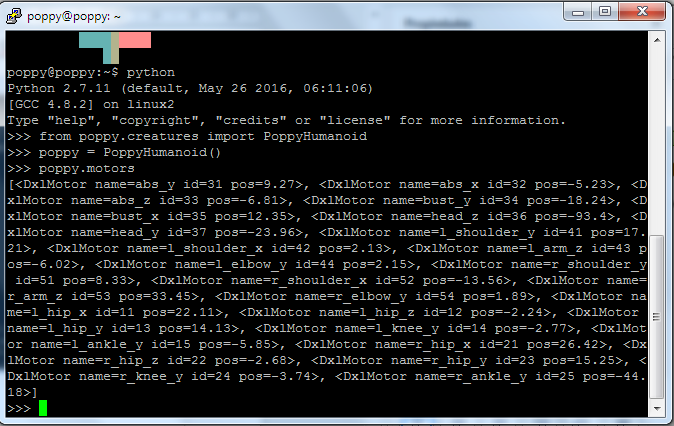
Podemos ejecutar código directamente desde el terminal o simplemente manejar la Odroid sin necesidad de pantalla mediante los comandos de Linux (bash). Para ello, abrimos una conexión SSH. Para ello, debemos tener instalado un servicio SSH como PuTTY. Como ya se ha dicho en el apartado [3.5. CLIENTE SSH: PuTTY](#_bookmark23), se introduce la dirección IP de la placa o el nombre de esta (predeterminadamente *Poppy.local*) y se teclea el nombre de usuario (predeterminadamente *poppy*) y la contraseña (predeterminadamente *poppy*).

NOTA: se accede por defecto al usuario *poppy* porque será este el usuario en el que se instale lo relacionado con el poppy y sus librerías de uso. No obstante, también se puede acceder por SSH al usuario *odroid.* No obstante hay un detalle y es que, para acceder al usuario *odroid*, la conexión se establece a través de la dirección de *Poppy.local*, por ser éste el nombre que se le ha puesto a la ODROID. Para acceder al usuario odroid basta con escribir como nombre de usuario y contraseña:

login: *odroid*

pass: *odroid*

Se abre la conexión SSH en la cuál podemos teclear cualquier comando del terminal de Linux mediante los cuáles podemos hacer cualquier tarea que sea posible a través de una terminal. Una de estas tareas es abrir un terminal de Python simplemente tecleando python en la consola. A partir de ese momento el robot puede ser manejado, de un modo un tanto tedioso, comando a comando.



*Figura 31: Ejemplo Python a través de SSH*

## Núcleo remoto

Se puede controlar un núcleo remoto desde un IDE, como por ejemplo Spyder. Este método está en fase de exploración de su potencial. Véase: [Ejecutar Debugger desde PC a Odroid](#_bookmark66).

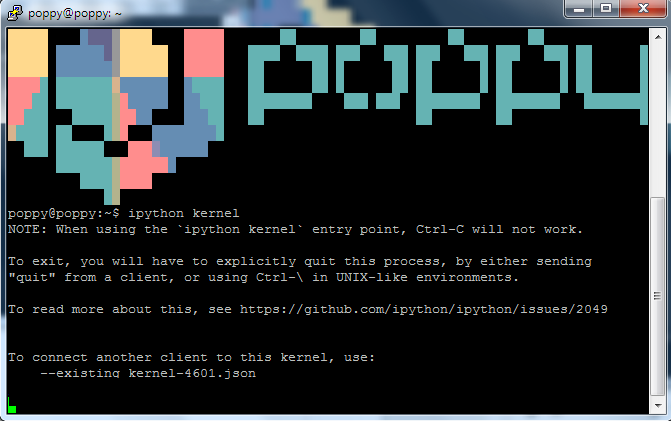
1. Instalar paramiko (es necesario para que funcione con Spyder, con otros no se ha probado). Para instalar Paramiko, teclear en un terminal con permisos de administrador lo siguiente:

pip install paramiko

1. En el terminal de la Odroid se abre un núcleo de IPython tecleando el siguiente código:

ipython kernel

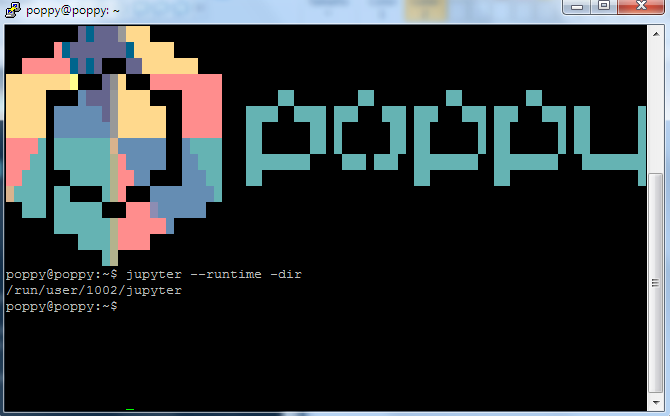
¡No cerrar la pantalla de PuTTY o dejará de funcionar el núcleo! Haciéndolo desde el explorador WEB se puede cerrar la pestaña ya que hasta que no se le diga para detener el terminal, no se cerrará. Véase: [4.4.3. Python, terminal](#_bookmark49)

**

*Figura 32: crear núcleo remoto*

1. El archivo que aparece en la última línea, debemos copiarlo a nuestro PC desde donde vamos a ejecutar el núcleo creado en la Odroid. Para ello abrimos un nuevo terminal localizamos la carpeta donde está guardado tecleando:

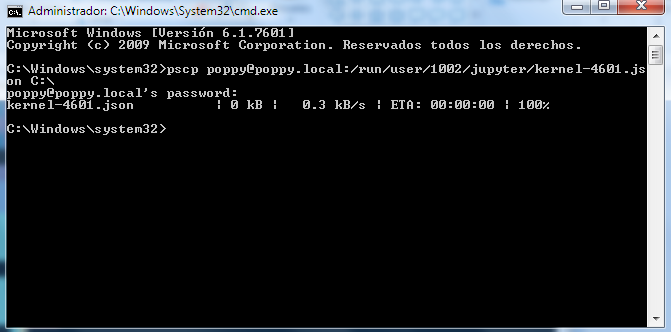
jupyter --runtime -dir



*Figura 33: localización del archivo*

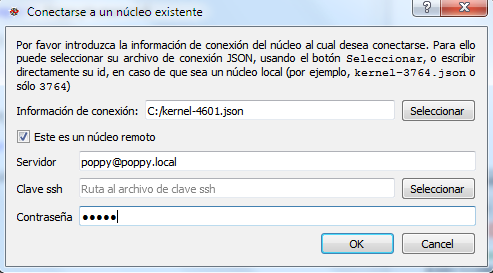
1. En esa ruta está el archivo generado por lo que ahora abrimos un terminal como administrador y que tecleamos el siguiente código para copiar el archivo de la ruta indicada en los pasos anteriores a C:\:

pscp poppy@poppy.local:/run/user/1002/jupyter/kernel-4601.json C:\



*Figura 34: copiar archivo mediante scp*

1. Abrir el IDE que admita controlar núcleos remotos como el incluido en Anaconda (Spyder). Para otros IDE mirar la documentación. Por ejemplo PyCharm solo admite núcleos remotos en la versión profesional.
2. Cargar el archivo .json y dar el nombre de usuario, dirección y contraseña.



*Figura 35: activar núcleo remoto en Spyder*

1. ¡Núcleo remoto activado!

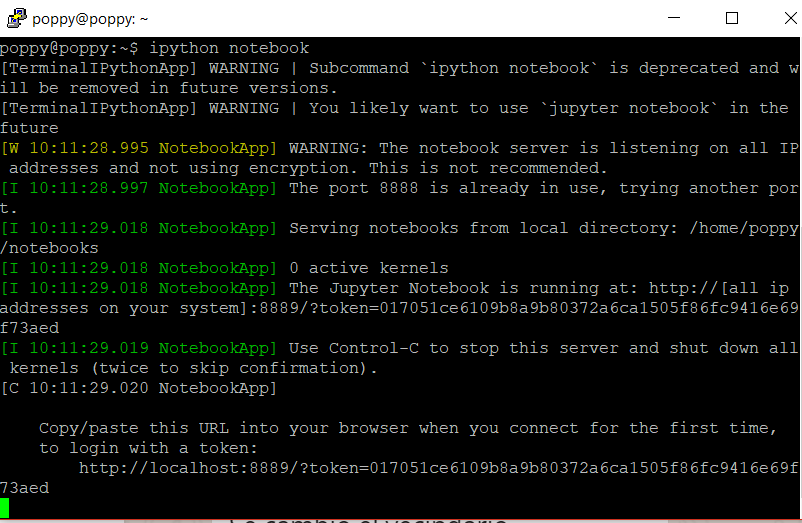
**4.3.3.- Empleo de un notebook en ODROID.**

Esta será **la forma preferente** de trabajar con el Poppy.

Un vez se ha establecido la conexión por SSH. Se lanza un notebook desde la terminal de SSH mediante la introducción del comando:

*ipython notebook*

Una vez que tenemos el notebook ejecutándose, debemos copiar la última línea que aparece en el notebook y cambiar el apartado de *local.host* por *Poppy.local*. Con esto, podemos acceder al notebook que se ha lanzado en la ODROID a través del PC.



Para lanzar los programas a través del notebook, es conveniente crear una carpeta en la ruta *notebook* en la que se copiaran los archivos necesarios para ejecutar el programa. El archivo a ejecutar desde el notebook será un *.ipynb.*

Destacar que para ejecutar este archivo, se debe de incluir en la carpeta (en la misma ruta de la carpeta) las librerías y/o archivos auxiliares que son necesarios para ejecutar el programa.

Recordar que para crear un directorio se usará el comando:

*mkdir nombre\_que\_queremos\_para\_la\_carpeta*

NOTA: recordar que para copiar archivos y/o carpetas de una máquina a otra (de PC a la ODROID) se siguen los comandos descritos previamentes:

*pscp user@host:source target*

*user=usuario del target (Ej. poppy)*

*host=dirección del target (Ej. Poppy.local)*

*source=ruta origen de archivo*

*target= ruta destino copia de archivo*

**Ejemplo: De Windows a Linux**

Ruta origen: D:\TESIS\poppy-tesis\PythonSolo\NEAT\2018\_04\_09\_VREP\_new\_json

Ruta destino: /notebooks/prog\_para\_ejecutar/2018\_04\_09\_moveArm

pscp D:\TESIS\poppy-tesis\PythonSolo\NEAT\2018\_04\_RealRobot\2018\_04\_11\_vrepAndReal\poppySimRobot.py [poppy@Poppy.local:notebooks/prog\_para\_ejecutar/2018\_04\_12\_startNEAT](mailto:poppy@Poppy.local:notebooks/prog_para_ejecutar/2018_04_12_startNEAT)

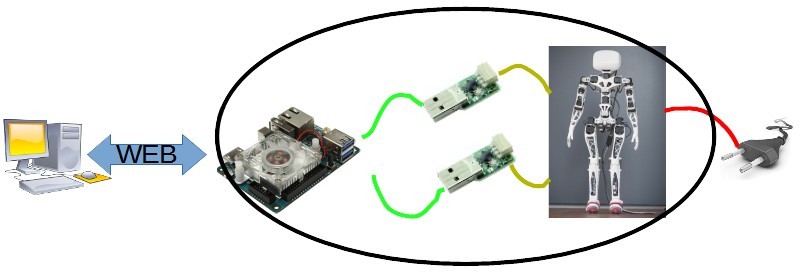
**Ejemplo 2: Copiar un directorio De Windows al cluster de la UDC**

Ruta origen: C:/Users/martin/Downloads/opencv-2.4.13.6

Ruta destino: /home/martin.naya

scp –r C:/Users/martin/Downloads/opencv-2.4.13.6 martin.naya@pluton.des.udc.es:/home/martin.naya

### MEDIANTE EXPLORADOR WEB



[Figura 36: Mediante un ordenador usando WEB se controla la Odroid para esta controlar los servos del Robot. El conjunto del robot dentro del círculo.](#_bookmark45)

[Breve descripción: la placa Odroid incluida en el robot se controla mediante un protocolo WEB desde otro PC. Mediante un cable ethernet o incluso WIFI.](#_bookmark45)

* [Ventajas: sistema implementado ya en el Robot](#_bookmark45)

[y susceptible de ampliar características. Se reduce el número de cables que tiene el robot](#_bookmark45)

* [Desventajas: el sistema actualmente está](#_bookmark45)

[bastante limitado. Gran dificultad actualmente en el uso no final (desarrollo de aplicaciones).](#_bookmark45)

PASOS NECESARIOS:

* + 1. Tener correctamente instalada la placa Odroid: [3.6. PUESTA APUNTO ODROID XU4](#_bookmark27).
    2. Tener correctamente instalado el servicio Bonjour: [3.4. BONJOUR SERVICE](#_bookmark19)
    3. Conectar la Odroid a la misma LAN que el PC desde que se va a acceder al servicio HTML.
    4. Alimentar el bus Dynamixell con los 12V.

EJEMPLO DE USO:

Poppy tiene un servicio Web desde el que se pueden realizar múltiples tareas. Las tareas que se pueden realizar son:

1. Monitor y control: una interfaz gráfica que muestra el estado de todos los motores y permite ejecutar ciertas aplicaciones.
2. Programar en Snap.
3. Abrir un “explorador de archivos” de Jupyter en el cuál se pueden iniciar programas hechos en Jupyter, crear nuevos programas a través de Jupyter y abrir una consola entre otras cosas.
4. Configurar el robot: cambiarle el nombre y actualizarlo.
5. Reiniciar el robot.
6. Ver el LOG de errores.
7. Apagar la Odroid.



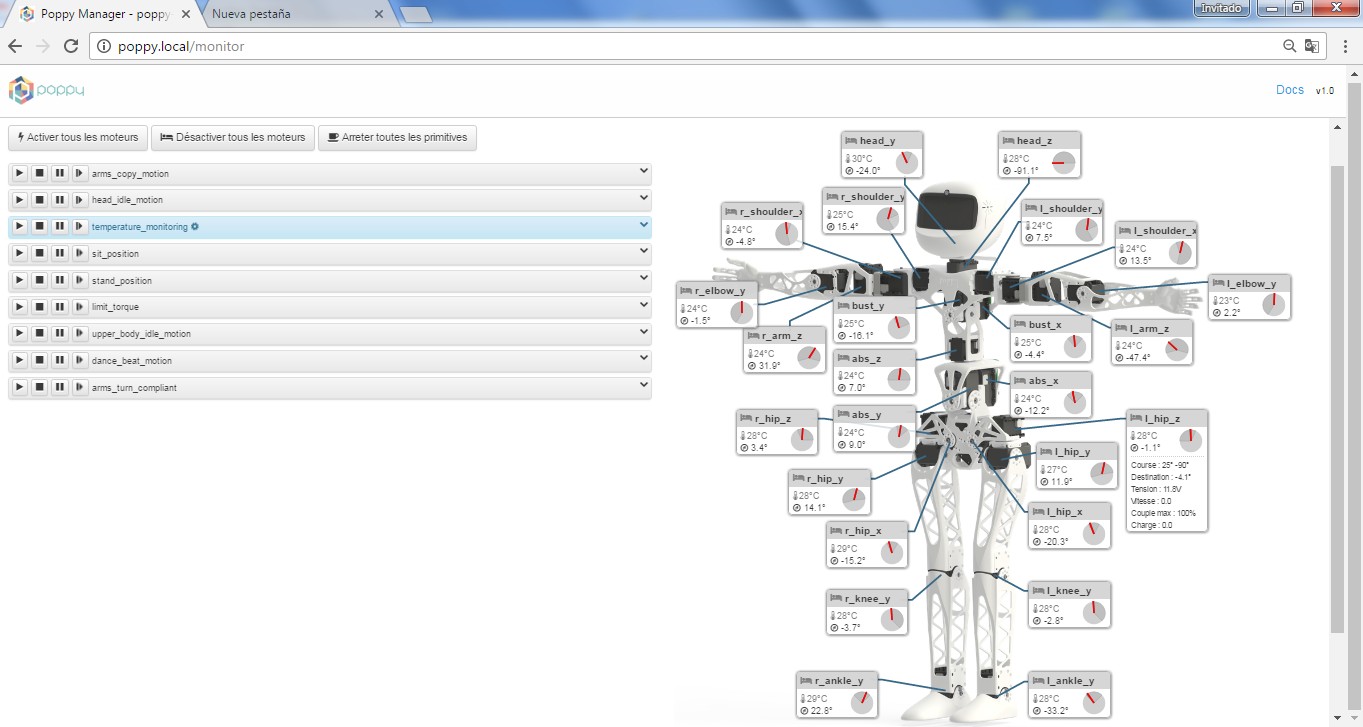
*Figura 37: Interfaz de Poppy Manager que se accede tecleando poppy.local/ en un navegador WEB. Los números indican las diferentes tareas que puede realizar y listadas arriba.*

## Monitor and Control

Para poder usar este servicio debemos inicializar el servicio. Para ello debemos teclear en una consola de la Odroid un comando. La consola puede hacerse mediante SSH o mediante el explorador Jupyter. Véase respectivamente: [3.5. CLIENTE SSH: PuTTY](#_bookmark23) y [4.4.3. Python, terminal](#_bookmark49)*.* El comando es:

poppy-services --http poppy-humanoid

La consola no se debe cerrar para que funcione este servicio y además mientras se use este servicio no se puede manejar el robot desde otro punto porque ocupa el puerto. Hay que procurar cerrar correctamente el proceso si no provocará que quede ocupado los puertos y puede ser necesario el reinicio de la Odroid.



*Figura* *38: Se puede ver como es la interfaz del monitor de Poppy*

Tal como se puede ver en la [Figura 38](#_bookmark47) se muestra varios datos interesantes:

* + - * Temperatura
      * Posición
      * Rango de movimiento
      * Posición solicitada
      * Estado del servo
      * Voltaje
      * Carga

Además, se pueden ejecutar varias funciones como:

* + - * Un brazo copia el movimiento del otro
      * Ponerse de pie
      * Ponerse sentado
      * Movimientos de espera
      * Otros movimientos

## Build Your Own Blocks

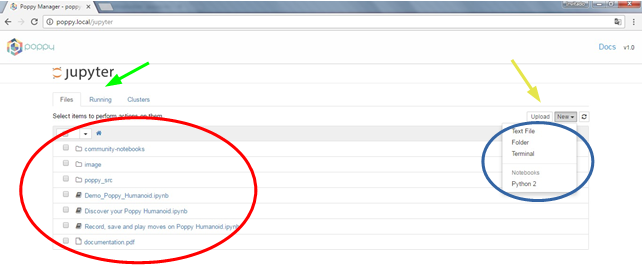
Programar mediante el lenguaje visual Snap!

## Python, terminal

Para acceder a este servicio, simplemente debemos pulsar sobre él. A continuación, se pide una contraseña. Para acceder a esta contraseña, debemos:

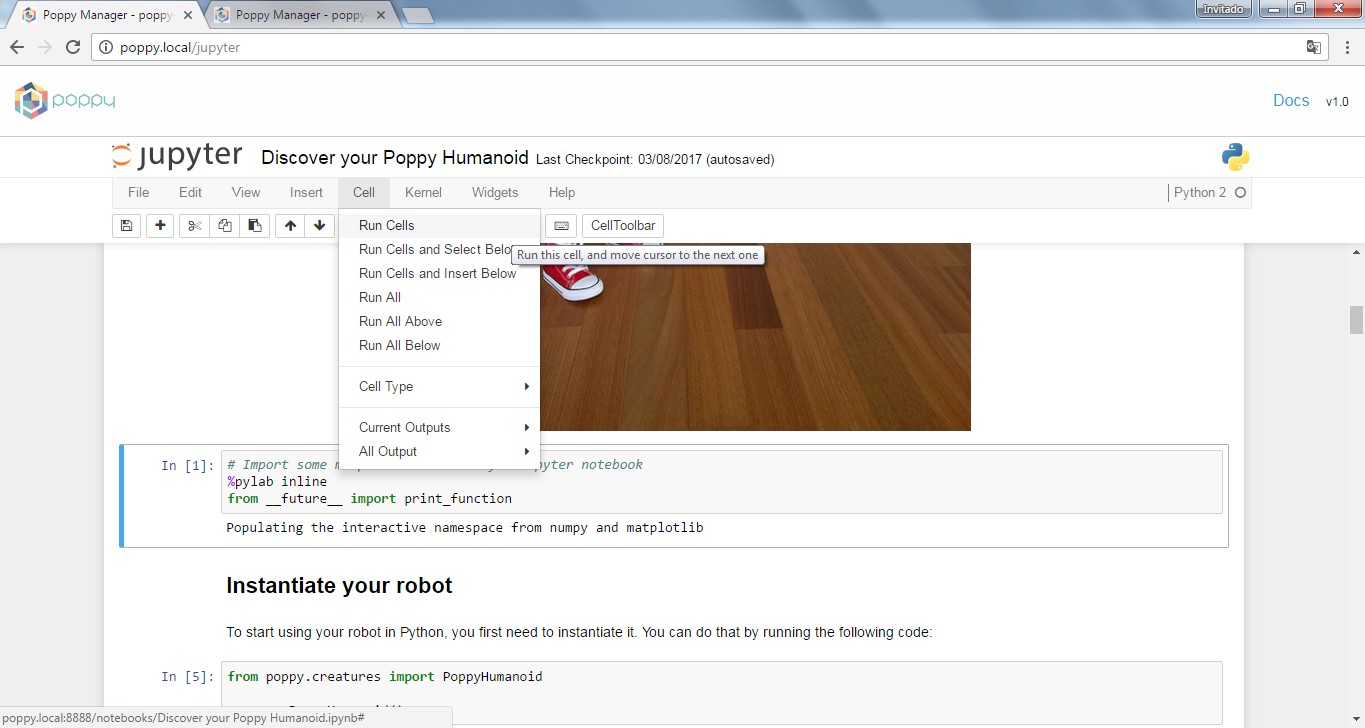
1. En un terminal SSH teclear jupyter notebook list apareciendo una dirección WEB.
2. Copiar los caracteres que salen **después** de token= y antes de :: (hay que tener cuidado al copiarlo directamente del Putty debido a que se copian también los espacios en caso de que los hubiera; por ello es recomendable copiar primero la contraseña en el blog de notas, por ejemplo, para comprobar que no hay espacios en la contraseña.) e introducirlos en la casilla que aparece en el navegador WEB. Se probó a insertar esta dirección en los navegadores web Internet Explorer y Google Chrome en un ordenador con un sistema operativo Windows 7 de 64 bits y se obtuvo error al no cargarse la página. Al pegar la dirección en los navegadores Mozzilla Firefox y Microsoft Edge sí que se pudo acceder a la página, por lo que se recomienda utilizar estas dos últimas opciones. Ver problema [Contraseña Notebook](#_bookmark67)
3. Inmediatamente salen los archivos que se encuentran en la ruta

**/home/poppy/notebooks** de la placa Odroid

**

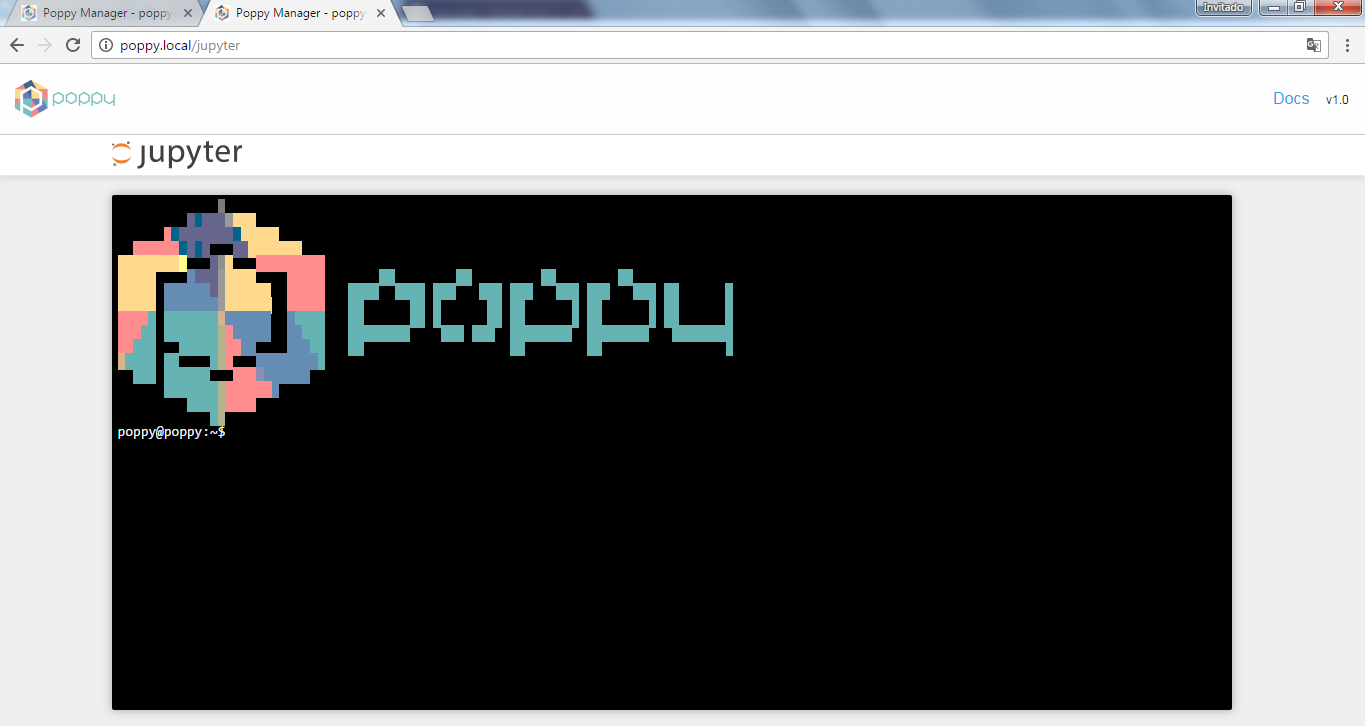
*Figura* *39: Interfaz del explorador de archivos. Los círculos y las flechas señalan cosas importantes que se detallan abajo.*

En la [Figura 39](#_bookmark50) podemos distinguir varias cosas importantes:

* + - * Dentro del círculo rojo podemos encontrar un listado de archivos y carpetas. Para abrir uno simplemente se pulsa sobre él. Seleccionándolos se pueden borrar, cambiar de nombre o duplicar. Los archivos con extensión .ipynb son los llamados Notebooks de Jupyter. Estos archivos contienen código explicado e interactivo que puede ser ejecutado celda a celda. Para ejecutar una celda se coloca el cursor sobre ella y se selecciona Cell→Run Cells.

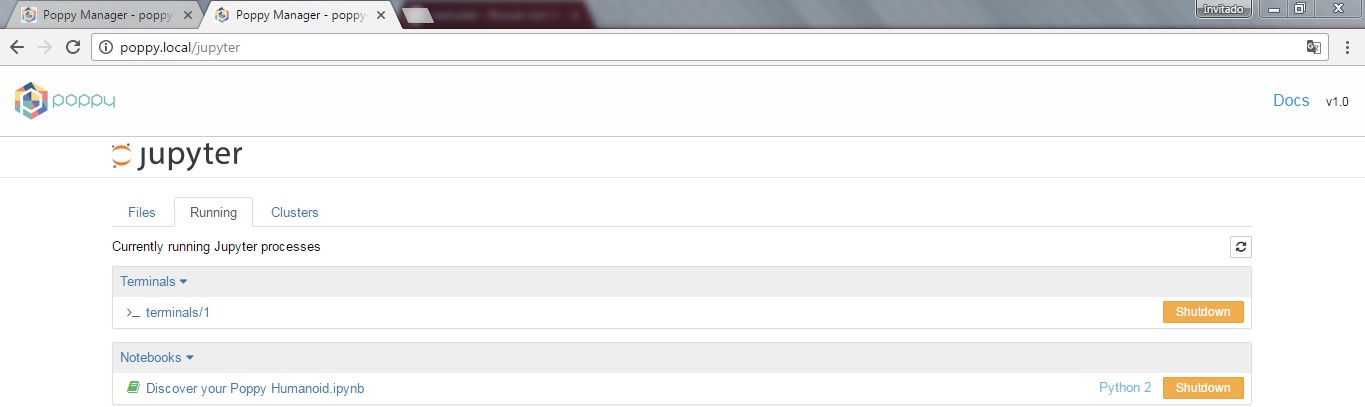
*Figura 40: Ejemplo de Jupyter Notebook.*

* + - * Dentro del círculo azul se pueden ver las opciones de creación: Texto, Carpeta, Terminal y Notebook de Python.



*Figura 41: Terminal desde la WEB.*

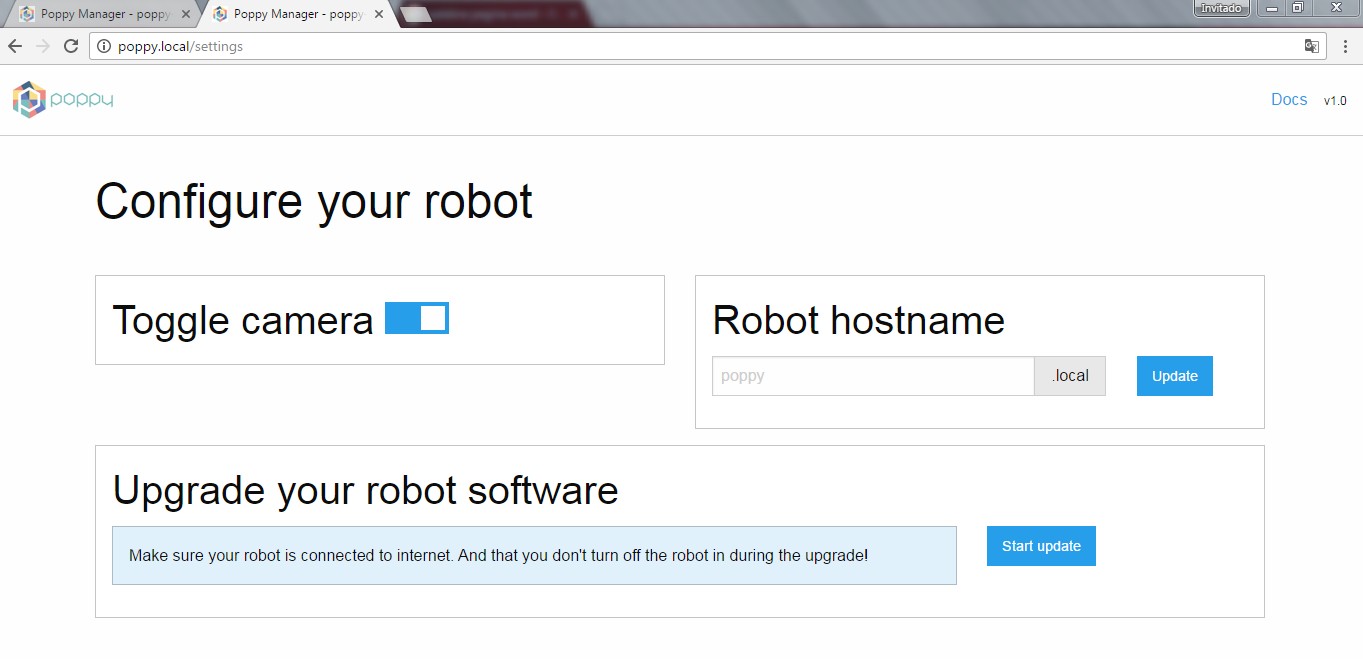
* + - * El botón que señala la flecha amarilla sirve para enviar archivos.
      * La pestaña a la que señala la flecha verde sirve para ver los procesos en ejecución y detenerlos o reanudar los que están en ejecución.



*Figura 42: Pestaña running. Se puede ver que hay un terminal y un Jupyter Notebook. Para cerrarlos se pulsa el botón shutdown y para abrirlos encima del nombre.*

## Configure the robot

En este apartado podemos cambiar el nombre que tiene en la red (predeterminadamente es poppy.local). También parece que se pueda realizar una actualización[9](#_bookmark52).



*Figura 43: Apariencia función settings*

1. Función no probada

## Reboot the robot

Resetea la placa Odroid.

Para reiniciar el sistema: sudo reboot o sudo shutdown -r now o sudo shutdown -r +0

Para reiniciar el sistema en un tiempo determinado (en minutos): sudo shutdown -r +5

Para reiniciar el sistema a una hora específica: sudo shutdown -r 23:30

## What happened?

Muestra el LOG del robot, es decir, los eventos importantes tales como errores o warnings.

## Shutdown the Odroid

Apaga la placa Odroid[10](#_bookmark56).

Para apagar el sistema: sudo halt (este comando detiene todas las CPU del sistema) o sudo poweroff (envía un comando ACPI -"Advanced Configuration and Power Interface" (Interfaz Avanzada de Configuración y Energía)- para indicarle a la PSU -"Power Supply Unit" (Fuente de Alimentación) que desconecte la energía)

Para que apague el sistema ahora mismo: *sudo shutdown -h now* o sudo *shutdown -h +0* (detiene o apaga el sistema dependiendo de cuál es la configuración del sistema, pudiendo ser cambiada en ocasiones en la BIOS)

Para que apague el sistema en un tiempo determinado *sudo shutdown -h +m* donde m es el número de minutos que deben transcurrir para que el sistema se apague (en Ubuntu es posible omitir el argumento -h dejando únicamente sudo shutdown +m). Por ejemplo: *sudo shutdown +10*

Para que se apague a una hora específica (utiliza el sistema de 24 horas, es decir, de 00 a 23). Por ejemplo a las 17:30: sudo shutdown -h 17:30.

1. Mirar el apartado [Apagado](#_bookmark63) ya que la placa antigua no apaga bien la Odroid ya que el ventilador continua en funcionamiento. Con esta placa usar SSH con el comando sudo poweroff

# 5. PROGRAMACIÓN EN VREP

# Este apartado se tratará como una especie de guía para empezar a trabajar con el robot Poppy en simulación. Destacaremos aquí, que la intención de esta guía inicial de programación es de ser lo más portable posible al robot Poppy real.

Para programar un robot virtual en el simulador V-REP seguiremos los siguientes pasos:

1. Abrimos V-REP.

2. Para programar nuestro robot en el simulador empleando Python, abrimos Anaconda Prompt, tecleamos el siguiente código y presionamos Enter:

*ipython notebook*

Se recomienda hacerlo de esta manera para evitar problemas. Por ejemplo, si se hubiera accedido a Jupyter desde Anaconda Navigator, posiblemente no se hubiese podido conectar con V-REP.

3. Tras unos segundos se abrirá en nuestro navegador predeterminado una pestaña como la del paso 5. de programación con el robot, donde clicaremos la opción Python 2 en el desplegable de New.

4. Se abrirá una pestaña donde podremos programar y ejecutar programas que estarán dispuestos en celdas. Para ejecutar el código de una celda presionamos Shift+Enter o clicamos en el botón Play.

5. Un código que se puede escribir para comprobar que todo está instalado correctamente es el siguiente:

*from pypot.vrep import from\_vrep*

*from pypot.creatures import PoppyTorso*

Si se ejecuta sin mostrar ningún error, es probable que todo esté correctamente instalado.

1. Para crear en el simulador nuestro robot, ejecutaremos el siguiente código:

*from pypot.creatures import PoppyTorso*

*poppy = PoppyTorso(simulator='vrep')*

Si quisiéramos programar el Poppy Humanoide escribiríamos PoppyHumanoid en vez de PoppyTorso.

Sin embargo para programar el Poppy Ergo Jr. usaríamos el siguiente código:

*from poppy\_ergo\_jr import PoppyErgoJr*

*creature = PoppyErgoJr(simulator='vrep')*

1. Una vez ejecutado el código mencionado anteriormente, debería aparecer en el simulador V-REP el Poppy Torso.

Para ver más información respecto a la programación en V-REP se remite a la siguiente página web:

*https://github.com/poppy-project/poppy-humanoid/blob/master/software/samples/*

*notebooks/Controlling%20a%20Poppy%20humanoid%20in%20V-REP%20using%20pyp*

*ot.ipynb*

Téngase en cuenta que se emplea el Poppy Humanoide habiendo que adaptar el código al Poppy Torso.

Si se trabaja con el VREP no se puede trabajar mediante SSH el robot real. Para alternar:

# Working with the real robot

import pypot.robot

poppy = pypot.robot.from\_config(config)

poppy.walk.start()

# Working with the simulated version

import pypot.vrep

poppy = pypot.vrep.from\_vrep(config, vrep\_host, vrep\_port, vrep\_scene)

poppy.walk.start()



*Figura 44: Imagen del Poppy.torso en VREP.*

Agregar Sensores a librería Pypot

Sensores de visión y proximidad en simulador V-REP

Creado por: Jose Ruiz Jara

Fecha de Creación 08/03/2018

Contenido

[1. Descripción del documento 2](#_Toc508283068)

[2. Crear clases para sensores 2](#_Toc508283069)

[3. Ejemplos de implementación: Clase ProximitySensor y VrepVisionSensor 4](#_Toc508283070)

[4. Cargando sensores desde archivo de configuración 5](#_Toc508283071)

[5. Cambios realizados para cargar configuración en el robot simulado 5](#_Toc508283072)

[6. Instalación de librería modificada 6](#_Toc508283073)

[7. Ejemplo de uso 6](#_Toc508283074)

# Añadido de nuevas funcionalidades a la librería Pyplot.

Este apartado tiene como objetivo detallar el trabajo a realizar para poder agregar nuevos sensores y/o actuadores a la librería Pypot[[1]](#footnote-1), ya sean sensores que funcionan solo en el simulador, en el robot físico o en ambos.

Los ejemplos y explicaciones de este documento se basarán principalmente en los sensores de visión y sensores de proximidad del simulador V-REP[[2]](#footnote-2).

En las siguientes secciones se describirá el proceso a seguir para agregar sensores nuevos, y los cambios realizados a la librería para poder cargar la configuración de dichos sensores al robot simulado, desde un archivo de configuración.

Las nuevas funcionalidades se encuentran en el branch *sensors\_from\_config* del repositorio[[3]](#footnote-3)

# Crear clases para sensores.

La librería Pypot cuenta con una clase abstracta *Sensor* que se encuentra en el folder *pypot/robot*, de la cual deben heredar los nuevos sensores que se deseen agregar.

Los nuevos tipos de sensores que se vayan a agregar deben ser creados dentro de la carpeta *pypot/sensor*. Por ejemplo, para poder utilizar los sensores de proximidad de V-REP con Pypot se creó la carpeta *proximity\_sensor* dentro de *pypot/sensor*. Dentro de la carpeta “proximity\_sensor” se puede encontrar el archivo *“proximitysensor.py”*, el cual contiene la clase *ProximitySensor* que hereda de la clase abstracta *Sensor* y a su vez implementa los respectivos métodos para obtener la información del sensor desde la simulación.

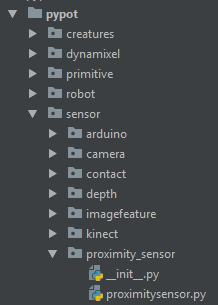


Figura 45.

Es importante crear el archivo *\_\_init\_\_.py* dentro de las carpetas de los nuevos tipos de sensores, pues dicho archivo es necesario para que la carpeta sea considerada como un paquete[[4]](#footnote-4) de la librería. En cada archivo *\_\_init\_\_.py* se deben importar los distintos tipos de sensores que podrán ser importados directamente por alguna otra clase, por ejemplo, en la carpeta *camera* el contenido de ese archivo es el siguiente:

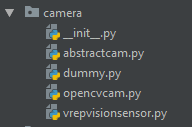
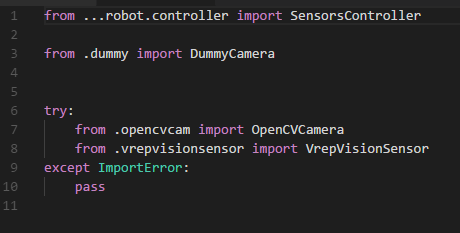
 

Figura 46. Figura 47.

Todas las clases de sensores que solo se utilizarán con el simulador, deben tener una propiedad “*simulator\_only*” inicializada en *True* para que sean ignoradas al utilizar el robot real*.*

El constructor de las nuevas clases debe recibir un parámetro llamado *vrep\_io* para poder acceder a las funciones del API remoto[[5]](#footnote-5) de V-REP. Dicho parámetro es enviado por el método que se encarga de cargar los sensores y motores desde el archivo de configuración, por lo que basta con agregarlo en el constructor de cada nueva clase de sensor.

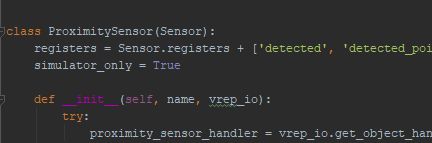


Figura 48.

Para inicializar un sensor desde un archivo de configuración (el archivo de configuración es el archivo con extensión *.json*) se debe agregar dentro del aparatado de *“sensors”* (figura 49). Por ejemplo, si se desea agregar un sensor que se llame *“floating\_camera”* debe agregarse al archivo como un objeto que contenga ‘type’ (el nombre de la clase del sensor) y las demás propiedades definidas en el constructor de dicha clase.



Figura 49.

# Ejemplos de implementación: Clase ProximitySensor y VrepVisionSensor.

Para poder obtener datos desde los sensores del simulador V-REP, se hace uso del Remote API para Python del simulador. Dado que la librería Pypot ya hace uso de dicho API solo basta con utilizar las funciones de la clase VrepIO que se encuentra en pypot/vrep/io.py

Las clases *ProximitySensor* y *VrepVisionSensor* son clases creadas para poder utilizar sensores de proximidad y sensores de visión de V-REP, directamente desde la librería Pypot. En ambas clases se puede encontrar una propiedad llamada *simulator\_only* inicializada en *True,* esto permite que dichos sensores sean ignorados a la hora de utilizar un robot real y prevenir un error en su inicialización.

Tanto el constructor de *ProximitySensor* como el de *VrepVisionSensor* reciben un parámetro llamado *vrep\_io*, con el que pueden acceder a las funciones de API remoto del simulador.

Como se puede observar en el código fuente de las clases, la implementación de ambas es muy similar y su constructor realiza los siguientes pasos:

1. Se agregan al array de registros de la clase padre, las propiedades específicas del sensor.
2. Se obtiene el *Handle* del sensor, por medio de su nombre utilizando la función *vrep\_io.get\_object\_handle.*
3. Si el objeto se encuentra en el robot, se intenta obtener valores desde el sensor, mediante una llamada al API remoto (por ejemplo *vrep\_io.call\_remote\_api(‘simxReadProximitySensor’…)* en modo de operación *Streaming*). En caso contrario…..
4. Se define la función que obtendrá los valores del sensor, dependiendo del sensor, la documentación del API de V-REP recomienda utilizar el modo de operación *Buffer.*
5. Se inicializan las variables de la clase.
6. Se ejecuta el constructor de la clase de la que hereda (ProximitySensor hereda directamente de Sensor mientras que VrepVisionSensor hereda de AbstractCamera).
7. Se definen los métodos de la clase y se implementan los métodos abstractos de la clase padre.

Los modos de operación pueden variar dependiendo del tipo de sensor a utilizar, por lo que se aconseja verificar los parámetros necesarios y los valores que retorna cada función en la respectiva documentación[[6]](#footnote-6) del API de V-REP.

Tal como se mencionó anteriormente, la clase VrepVisionSensor hereda funcionalidades de la clase AbstractCamera que se encuentra en pypot/sensor/camera, por lo que al inicializar un sensor de visión se crea un nuevo hilo que va a intentar obtener una determinada cantidad de frames por segundo, la cual debe ser especificada en el archivo de configuración. Por su parte, la actual implementación del sensor de proximidad obtiene valores del sensor únicamente cuando la función *scan* es ejecutada.

# Cargando sensores desde archivo de configuración.

Para poder cargar el robot con una configuración personalizada basta con especificar la ruta del archivo JSON en el que se encuentran los sensores y motores que el robot utilizará. En caso de utilizar algún tipo de sensor en el robot simulado, es necesario especificar la ruta de la escena a la cual se le han agregado dichos sensores.

Robot simulado:

Para trabajar con el robot en simulación, se deben de realizar los siguientes pasos. Para la realización de los pasos que se indican a continuación, se asume que se trabaja con un SO Windows y con Anaconda como entorno de trabajo de Python.

* Abrir el Anaconda.
* Establecer como ruta de trabajo, la ruta en la que se encuentra la escena con la que queremos trabajar.

Para el caso concreto en el que estamos trabajando, la escena se encuentra en la carpeta de *“samples”* del repositorio. Por lo tanto, si se tiene descargado el repositorio, la ruta desde la cual se debe de lanzar en notebook debería de ser similar a ésta:

"D:\PoppyRobot\pypot\_sensors\samples\notebooks”

* En esta dirección, una vez lanzado el notebook, en la pestaña “Home” del navegador, existe un programa a modo de ejemplo, llamado: *“Using sensors added to V-REP scene.ipynb”* en el cual ya aparecen paso a paso, las instrucciones que se usan como ejemplo.

Robot real:

En el robot real el comando a ejecutar sería el siguiente:

config\_path = abspath('./vrep-sensors/torso\_vrep\_sensors\_config.json')

poppy = PoppyTorso(config=config\_path)

mientras que para el robot simulado es necesario ejecutar las siguientes líneas:

scene\_path = abspath('./vrep-sensors/torso\_sensors\_scene.ttt')

config\_path = abspath('./vrep-sensors/torso\_vrep\_sensors\_config.json')

poppy = PoppyTorso(simulator='vrep', scene=scene\_path, config=config\_path)

La función *abspath* es utilizada para obtener la ruta absoluta del archivo, y así evitar problemas a la hora de cargar la escena y el archivo de configuración.

Se debe tomar como base el archivo de configuración predeterminado de cada robot. Se puede encontrar dentro del directorio donde ese encuentra instalado Python o el ambiente utilizado.

En Windows:

[**directorio\_python**]/Lib/site-packages/[**nombre\_robot**]/configuration’

En Ubuntu:

/usr/local/lib/python[**version**]/dist-packages/[**nombre\_robot**]/configuration

# Cambios realizados para cargar configuración en el robot simulado

Los cambios realizados para poder cargar la configuración desde un archivo JSON pude revisarse en el commit 8be97e1 del repositorio dentro del branch *sensors\_from\_config*

Originalmente la librería no permite que el usuario especifique un archivo de configuración cuando utiliza el simulador. Dicha restricción fue removida y se agregaron las instrucciones necesarias para poder cargar dicha configuración al robot simulado.

Para poder lograrlo se modificaron los siguientes archivos:

* pypot/robot/config.py: se verifica si el sensor tiene la propiedad ‘simulator\_only’ para ignorar dicha clase en caso de que se vaya a inicializar el robot real.
* pypot/vrep/\_\_init\_\_.py: se copia el mismo método utilizado en el robot real para cargar los sensores, con la salvedad que se envía como parámetro la variable que se conecta con el simulador mediante el API remoto (vrep\_io) para que pueda ser utilizado a la hora de intentar obtener los valores de los sensores.
* pypot/vrep/io.py: Algunos métodos del API de V-REP requieren un modo de operación específico para funcionar correctamente. Actualmente en el archivo se encuentran definidos cuatro modos en el diccionario *vrep\_mode:* normal, streaming, sending y buffer. En caso de que sea necesario agregar algún otro modo bastará con agregar el nombre del modo de operación como la llave, y como valor se deberá especificar alguna de las variables nombradas con el prefijo ‘*simx\_opmode\_’* de la variable *remote\_api* (los modos disponibles se pueden encontrar en pypot/vrep/remoteApiBindings/vrepConst.py)

# Instalación de librería modificada

Para poder utilizar la librería modificada, es necesario seguir los siguientes pasos:

* 1. Desinstalar la librería original de Pypot si se encuentra instalada
     + $ pip uninstall pypot
     + Se debe verificar que la librería se haya desinstalado correctamente. Al ejecutar el comando ‘$ python -c “import pypot”’ debe mostrarse un error como este ‘ImportError: No module named pypot’
  2. En cualquier directorio, clonar el repositorio del proyecto
     + $ git clone <https://github.com/Jant1921/pypot_sensors.git>
  3. Entrar a la carpeta del proyecto
     + $ cd pypot\_sensors
  4. Cambiar al branch que contiene las modificaciones
     + $ git checkout sensors\_from\_config
  5. Instalar la librería desde el código fuente
     + $ python setup.py install

Si en alguno de los pasos ocurre un error debido a alguna dependencia (por ejemplo, la librería setuptools) será necesario instalarla antes de continuar.

Si se presenta un error de privilegios, debe agregar ‘sudo’ al comando (en sistemas Linux) o iniciar la terminal con privilegios de administrador (en Windows).

Si la versión de la librería pypot ya se encuentra instalada, en caso de que se haya realizado alguna modificación en la librería y se hayan subido los cambios al repositorio, ésta nueva actualización deberá descargarse. Una vez hecho el pull de la librería y actualizada, habrá que instalar nuevamente la librería en el entorno de trabajo para que los cambios surjan efecto. En este caso, trabajando con Windows y Anaconda:

* Con el Anaconda abierto con permisos de administrador, nos posicionamos en el directorio donde se encuentra el archivo de instalación de la librería pypot, el archivo *“setup.py”.*
* En Anaconda, escribir:

*pthon setup.py install*

* Esto nos sirve para actualizar la librería cada vez que se descarga una nueva actualización.

# Ejemplo de uso.

En la carpeta samples/notebooks se ha incluido un notebook llamado ‘Using sensors added to V-REP scene.ipynb’ el cual puede ser ejecutado con jupyter. En dicho notebook se muestran ejemplos de uso de sensores de visión y de proximidad que han sido agregados a una escena de V-REP y especificados en un archivo de configuración.

# COMANDOS PARA PROGRAMAR EL POPPY

Algunos comandos que se pueden ejecutar en las IDE’s de Python y que pueden ser de utilidad son los siguientes:

* Para saber cuántos motores tiene nuestro robot:

*print(len(poppy.motors))*

*poppy.motors* es una lista de todos los motores conectados al robot.

* Para saber la posición actual de todos los motores:

for m in poppy.motors:

print(m.present\_position)

* También se puede acceder a un motor en concreto. Para hacerlo, todo lo que se necesita saber es el nombre del motor al que se quiere acceder. Se puede encontrar la lista en la documentación del ensamblado del robot. También se puede obtener la lista del nombre de todos los motores directamente desde Python:

for m in poppy.motors:

print(m.name)

también se podría usar la siguiente expresión en Python:

*print([m.name for m in poppy.motors])*

* Entonces se puede acceder directamente al motor deseado por su nombre:

*m = poppy.m3*

u obtener su posición:

*print(poppy.m3.present\_position)*

* Los valores más comunes para los motores son:
* *present\_position*
* *present\_speed*
* *present\_load*
* Dependiendo del robot Poppy que se tenga, habrá diferentes sensores disponibles. Para obtener la lista de todos los sensores, se hace como con los motores:

*print([s.name for s in poppy.sensors])*

* Y se puede acceder a un sensor específico por su nombre, al igual que sucedía con los motores. Por ejemplo, para obtener una imagen de la cámara:

*img = poppy.camera.frame*

* Para mover los motores es necesario que estén rígidos; es decir, que no se puedan mover con la mano. Para ello, hay que desactivar su supresión. Suponiendo un motor *m3* que se quiere que se mueva:

*poppy.m3.compliant = False*

El motor *m3* debería estar ahora rígido. Y para que se mueva a su posición 0:

*poppy.m3.goal\_position = 0*

Por ejemplo, si se quisiera que el motor fuera de 30° a -30° tres veces:

*import time*

*for \_ in range(3):*

*poppy.m3.goal\_position = 30*

*time.sleep(0.5)*

*poppy.m3.goal\_position = -30*

*time.sleep(0.5)*

Nótese que después de cada nuevo valor asignado a *goal\_position* se espera a que el motor tenga tiempo suficiente para llegar a su nueva posición.

* Otra forma de conseguir lo anterior es usando el comando *goto\_position:*

*import time*

*for \_ in range(3):*

*poppy.m3.goto\_position(30, 0.5, wait=True)*

*poppy.m3.goto\_position(-30, 0.5, wait=True)*

Como se puede ver, con este método se introducen 3 argumentos, la posición objetivo, la duración del movimiento y si hay que esperar o no al final del movimiento.

* Si se quieren mover varios motores al mismo tiempo, se puede hacer a través de un código como el siguiente:

for \_ in range(3):

poppy.m1.goal\_position = -20

poppy.m3.goal\_position = 30

time.sleep(0.5)

poppy.m1.goal\_position = 20

poppy.m3.goal\_position = -30

time.sleep(0.5)

o usar un diccionario de Python almacenando la posición objetivo del motor que se quiere que se mueva, que puede ser dada al comando *goto\_position*:

pos\_1 = {'m1': -20, 'm3': 30}

pos\_2 = {'m1': 20, 'm3': -30}

for \_ in range(3):

poppy.goto\_position(pos\_1, 0.5, wait=True)

poppy.goto\_position(pos\_2, 0.5, wait=True)

* Se puede devolver un motor a su modo supresión (donde lo puedas mover libremente) poniendo su registro *compliant* a *True*:

*poppy.m3.compliant = True*

# PROBLEMAS ENCONTRADOS Y FUTURAS LÍNEAS

Código colores:

|  |  |
| --- | --- |
|  | Problema no grave y sin estar en fase de resolución/exposición de alternativas |
|  | Problema grave sin fase de resolución |
|  | Problema en fase de resolución |

### HARDWARE

1. Cabeza:
   * Problema 1: es demasiado pequeña para poder conectar los USB de un tamaño normal. Véase [2.2. COMUNICACIÓN](#_bookmark5).
     + Solución actual: no cerrar la cabeza.
     + Solución futura 1: se podría rediseñar la cabeza.
     + Solución futura 2: comprar HUB que se adapte al espacio y sea pequeño[11](#_bookmark60),[12](#_bookmark61).
   * Problema 2: la placa Odroid no se adapta al 100% a los soportes.
     + Solución actual: tornillo con dos tuercas en uno de los lados.
     + Solución futura 1: se podría rediseñar la cabeza.
     + Solución futura 2: se podría imprimir las nuevas versiones existentes en el repositorio.
   * Problema 3: falta cámara, pantalla, hub, altavoz, micrófono y sensor de contacto.
     + Solución actual: no es necesario.
     + Solución futura: búsqueda WEB.
2. Los propuestos por la documentación son estos: [https://www.amazon.fr/gp/product/B003NF0MSE?](https://www.amazon.fr/gp/product/B003NF0MSE?psc=1&amp;redirect=true&amp;ref_=oh_aui_detailpage_o02_s00) psc=1&redirect=true&ref\_=oh\_aui\_detailpage\_o02\_s00 // [https://www.pixmania.fr/p/abix-hub-4-ports-usb-20- croix-noire-1231476](https://www.pixmania.fr/p/abix-hub-4-ports-usb-20-croix-noire-1231476)
3. A falta de comprobar si pueden caber en la cabeza por el tamaño estas son unas propuestas de RS: [http://es.rs- online.com/web/p/hubs-usb/8988400/](http://es.rs-online.com/web/p/hubs-usb/8988400/) //<http://es.rs-online.com/web/p/hubs-usb/6845424/>(specs. corregidas:<http://www.cablesdirect.co.uk/Search-Catalog/NEWlink-4-Port-USB-20-Hub-with-Cable/NLUSB2-204B>)

### SOFTWARE

La placa actual se apaga correctamente con el comando:

*sudo poweroff now -p*

1. La documentación de la Odroid XU4 se encuentra en este enlace: [http://magazine.odroid.com/wp-](http://magazine.odroid.com/wp-content/uploads/odroid-xu4-user-manual-espanol.pdf) content/uploads/odroid-xu4-user-manual-espanol.pdf

### USO

1. Ejecutar Debugger desde PC a Odroid
   * Problema: usar el PC para desarrollar las aplicaciones y mediante un cable Ethernet o red LAN Wifi hacer funcionar el robot durante el debugg del programa.
     + Trabajar a través de servidor: Ej: samba
     + Búsqueda WEB ~~2. Sostener el robot~~
   * ~~Problema: manejar el robot cómodamente para su uso.~~
     + ~~Solución actual: ninguna~~
     + ~~Solución futura: crear base.~~
2. Sincronizar proyectos con Notebook Odroid
   * Problema: requiere subir archivos de forma manual
     + Trabajar a través de servidor.
3. Contraseña Notebook:
   * Problema: pide código aleatorio para acceder.
     + Solución actual: trabajar con ese código.
     + Solución futura: cambiar a contraseña.
4. Evitar choques brazos Poppy-Torso con mesa y sujetar adecuadamente el robot.
   * Problema: Poppy puede quedar inestable con brazos extendidos y algo de carga. Además puede chocar con el suelo y caerse.
     + Solución actual: ninguna.
     + Solución futura: agregar código que impida choques y buscar la forma de anclar la base correctamente.

# LISTA ARCHIVOS

|  |  |  |  |
| --- | --- | --- | --- |
|  | Carpeta | Web | Versión archivo adjunto |
| Anaconda: distribución Python | [\python\Anaconda2-4.3.1-Windows-](#_bookmark16) [x86\_64.exe](#_bookmark16) | [https://www.](#_bookmark15)[continuum](http://www.continuum.io/downloads#windows)[.](#_bookmark15)[io/downloads#windows](http://www.continuum.io/downloads#windows) | Windows v.-4.3.1  Python 2.7 |
| Bonjour: Zeroconf para Windows | [\configuracion\BonjourPSSetup.exe](#_bookmark21) | [https://support.apple.com/kb/DL999?locale=es\_ES](#_bookmark20) | Windows (descarga Marzo\_17) |
| Etcher: flashear \*.iso de SO | [\humanoid\_xu4\Etcher-1.0.0-beta.19-](#_bookmark32) [win32-x64.exe](#_bookmark32) | [https://etcher.io/#downloads](#_bookmark31) | Windows v.-1.0.0 beta 19 |
| PuTTY: SSH, SCP,... | [\configuracion\putty-64bit-0.68-](#_bookmark25) [installer.msi](#_bookmark25) | [http://www](#_bookmark24)[.putty.org/](http://www.putty.org/) | Windows  v.-0.68  (64 bits) |
| Roboplus: configuración Dynamixel servos | [\configuracion\RoboPlus(v1.1.3.0).exe](#_bookmark13) | [http://en.](#_bookmark12)[robotis.com/BlueAD/board.php?](http://en.robotis.com/BlueAD/board.php) [bbs\_id=downloads&mode=view&bbs\_no=1132559& page=2&key=&keyword=&sort=&scate=SOFTWAR](#_bookmark12) [E](#_bookmark12) | Windows Vista or higher  v.-1.1.3.0 |
| Ubuntu Poppy-Humanoide Odroid XU4 | [\humanoid\_xu4\poppy-humanoid-xu4-](#_bookmark30) [2016-05-26.img](#_bookmark30) | [https://github.com/poppy-project/poppy-humanoid/releases/](#_bookmark29) | Poppy- Humanoid XU4  26-5-2016 |
| USB2AX driver | [\drivers\USB2AX.inf](#_bookmark10) | [https://raw.github.com/Xevel/usb2ax/master/firmware/lufa\_usb2ax/USB2AX.inf](#_bookmark9) | Windows |
| Vídeo Debugger | [\videos\debugger.mp4](#_bookmark37) |  | MP4 |

# REFERENCIAS Y ENLACES DE INTERÉS

Documentacion Poppy:

Wiki Poppy: <https://docs.poppy-project.org/en/> GitHub Poppy: <https://github.com/poppy-project>

Referencias por apartado:

1. TRABAJOS PREVIOS A LA DOCUMENTACIÓN

-Guía de ensamblaje: [https://github.com/poppy-project/poppy-](https://github.com/poppy-project/poppy-humanoid/blob/master/hardware/doc/en/assemblyGuide.md) [humanoid/blob/master/hardware/doc/en/assemblyGuide.md](https://github.com/poppy-project/poppy-humanoid/blob/master/hardware/doc/en/assemblyGuide.md)

-Piezas y componentes Poppy: <https://github.com/poppy-project/poppy-humanoid/blob/master/hardware/doc/BOM.md>

1. HARDWARE

-Ficha técnica UBEC: <https://www.adafruit.com/product/1385>

-Cabeza: [https://github.com/poppy-project/Poppy-minimal-head-](https://github.com/poppy-project/Poppy-minimal-head-design/blob/master/doc/head_assembly_instructions.md) [design/blob/master/doc/head\_assembly\_instructions.md](https://github.com/poppy-project/Poppy-minimal-head-design/blob/master/doc/head_assembly_instructions.md)

-Odroid: <https://github.com/poppy-project/Poppy-minimal-head-design/blob/master/doc/head_back_xu4.md>

-Cableado: <https://github.com/poppy-project/Poppy-minimal-head-design/blob/master/doc/wiring.md>

* 1. DRIVERS

-Wiki USB2AX: [http://www.xevelabs.com/doku.php?id=product:usb2ax:usb2ax](http://www.xevelabs.com/doku.php?id=product%3Ausb2ax%3Ausb2ax)

-Instalación: [http://www.xevelabs.com/doku.php?id=product:usb2ax:quickstart](http://www.xevelabs.com/doku.php?id=product%3Ausb2ax%3Aquickstart)

* 1. ROBOPLUS

-Direccionamiento Dynamixel: [https://github.com/poppy-project/poppy-](https://github.com/poppy-project/poppy-humanoid/blob/master/hardware/doc/en/addressing_dynamixel.md) [humanoid/blob/master/hardware/doc/en/addressing\_dynamixel.md](https://github.com/poppy-project/poppy-humanoid/blob/master/hardware/doc/en/addressing_dynamixel.md)

-Manual Roboplus: <http://support.robotis.com/en/software/roboplus/dynamixel_monitor/quickstart/dynamixel_monitor_connection.htm>

* 1. ANACONDA

-Guía instalación: <https://docs.poppy-project.org/en/installation/install-poppy-softwares.html>

* 1. BONJOUR SERVICE

-Guía instalación: <https://docs.poppy-project.org/en/installation/install-zeroconf.html>

* 1. CLIENTE SSH: PuTTY

-Documentación PuTTY: <https://the.earth.li/~sgtatham/putty/0.68/htmldoc/index.html>

-PSCP: <http://stackoverflow.com/questions/10235778/scp-from-linux-to-windows>

-Vía Comandos: <http://kb.site5.com/shell-access-ssh/putty/putty-how-to-start-a-ssh-session-from-the-command-line/>

* + 1. MÉTODO DIRECTO: COPIA IMAGEN PRECONFIGURADA

-Guía instalación: <https://docs.poppy-project.org/en/installation/burn-an-image-file.html>

* + 1. MÉTODO REPOSITORIO GIT

-Guía instalación: <https://docs.poppy-project.org/en/installation/install-a-poppy-board.html>

3.7. VREP

-Enlace de descarga: <http://www.coppeliarobotics.com/downloads.html>

* 1. MEDIANTE SSH 23

-Conexión a Spyder: [http://wdb.ugr.es/~fjesusmartinez/es/2016/04/como-conectar-a-un-kernel-remoto-de-ipython-en-](http://wdb.ugr.es/~fjesusmartinez/es/2016/04/como-conectar-a-un-kernel-remoto-de-ipython-en-spyder/) [spyder/](http://wdb.ugr.es/~fjesusmartinez/es/2016/04/como-conectar-a-un-kernel-remoto-de-ipython-en-spyder/)

-Ruta Ipython: https://github.com/ipython/ipython/wiki/Cookbook:-Connecting-to-a-remote-kernel-via-ssh

-Paramiko: <http://www.paramiko.org/installing.html>

* 1. MEDIANTE EXPLORADOR WEB

-Poppy-monitor: <https://github.com/poppy-project/poppy-monitor>

Otros enlaces de interés:

Manual Odroid: <http://magazine.odroid.com/wp-content/uploads/odroid-xu4-user-manual-espanol.pdf> Poppy ejemplos: <https://github.com/HumaRobotics/poppy-examples> Python programación Poppy: https://docs.poppy-project.org/en/programming/python.htm

# CAMBIOS REVISIÓN DOCUMENTO

-Página 4 (v.-0.2 // v.-0.3): Cambio *El conector es idéntico a la de la Odroid.* por *El conector tiene un diámetro interno de 2,5 mm, 5,5 mm externo y polaridad positiva en el interno.*

*-*Página 14 (v.-0.2 // v.-0.3): Borrada: *Recomiendo hacerlo directamente desde una copia de una imagen preconfigurada.*

*-*Página 14 (v.-0.2 // v.-0.3): Añadido: *Mediante este método se carga una imagen preconfigurada que puede no estar actualizada con las últimas librerías.*

-Página 16 (v.-0.2 // v.-0.3): Añadido: *Con este método se garantiza la imagen más actualizada de Poppy pero es un proceso mucho más lento que el de la imagen preconfigurada.*

*-*Página 34 (v.-0.2 // v.-0.3): Añadidos nuevos pasos para abrir notebook: A continuación, se pide una contraseña. Para acceder a esta contraseña, debemos (...)

*-*Página 38 (v.-0.2 // v.-0.3): Cambiado pie de página a: *Mirar el apartado* [Apagado](#_bookmark63) *ya que la placa antigua no apaga bien la Odroid ya que el ventilador continua en funcionamiento. Con esta placa usar SSH con el comando sudo poweroff*

*-Página 38 (v.-02 //v.-03): añadidos varios métodos adicionales de apagado y reseteo de la Odroid.*

*-*Página 43 (v.-0.2 // v.-0.3): Agregado *hub, altavoz, micrófono y sensor de contacto* al problema 1.3.

*-*Página 44 (v.-0.1 // v.-0.2): Tachado problema de apagado. La Odroid actual se apaga correctamente desde SSH mediante sudo poweroff. Agregado: *La placa actual se apaga correctamente con el comando sudo poweroff la otra no. Solo se apaga bien con el sistema operativo original.*

*-*Página 45 ( *v.-0.2 // v.-0.3*): Borrado problema OpenCV

*-*Página 45 (v.-0.2 // v.-0.3): Problema 2 (Alimentación: Problema: transformador de 12V poco potente por lo que la placa sufre reinicios). Solucionado mediante la fabricación de un cable.

-Página 45 (v.-0.1 // v.-0.3): Problema 2 (Falta instalar torso en Odroid) solucionado instalando torso por completo ya que este va a ser su uso final.

-Página 45 (v.-0.2 // v.-0.3): Añadido problema: *Sincronizar proyectos con Notebook Odroid.*

-Página 45 (v.-0.2 // v.-0.3): Añadido problema: *Contraseña Notebook*

*-*Página 45 (v.-0.2 // v.-0.3): Añadido problema: *Evitar choques brazos Poppy-Torso con mesa y sujetar adecuadamente el robot.*

*-*Página 45 (v.-0.2 // v.-0.3): Tachado problema 2 debido a que se le extrajeron las piernas y se le puso una base para usar simplemente Poppy-Torso. De esta forma el problema no es que esté solucionado, simplemente no afecta al uso que se le está dando.

-Cambiados números de figuras a partir de la Figura 2.

-Añadidos puntos 3.7., 5. y 6.

-Cambiado nombre de dirección IP con respecto a v2 (que era poppy.local) por Poppy.local.

1. Repositorio de la librería Pypot https://github.com/poppy-project/pypot [↑](#footnote-ref-1)
2. http://coppeliarobotics.com/ [↑](#footnote-ref-2)
3. https://github.com/Jant1921/pypot\_sensors/ [↑](#footnote-ref-3)
4. https://docs.python.org/2/tutorial/modules.html#packages [↑](#footnote-ref-4)
5. V-REP Remote API Python http://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsPython.htm [↑](#footnote-ref-5)
6. http://coppeliarobotics.com/helpFiles/index.html [↑](#footnote-ref-6)