



Grupo de Robótica Educativa – ROBIND

Intel Edison

Manual de Instalación - Ubilinux & ROS

v16.03.07

Manual con las instrucciones para instalar tanto el sistema operativo Ubilinux como el sistema operativo robótico ROS en la Intel Edison y un ejemplo de como implementar un bus I2C con esta placa.

Autor: Jesús Martín - Pozuelo Martín - Mantero

Índice

Capítulos	Página
1. Instalación de Ubilinux en la Intel Edison	1
1.1. Eliminación de contenido innecesario	1
1.2. Instalación de programas necesarios	1
1.3. Instalación de Ubilinux en la Intel Edison	1
2. Instalación de la librería MRAA	5
3. Instalación de ROS en Ubilinux	7
3.1. Instalación de dependencias y paquetes necesarios	7
3.2. Herramientas y archivos necesarios para la instalación	7
3.3. Instalación de librerías externas esenciales para ROS	8
3.3.1. Console bridge	8
3.3.2. URDF	9
3.3.3. Collada	9
3.4. Instalación de ROS	10
4. Ejemplo de comunicación I2C	11
4.1. Componentes del circuito	12
4.2. Instalación del programa en la Intel Edison	12
4.3. Ejecución del programa	13

1. Instalación de Ubilinux en la Intel Edison

Una vez montada la Intel Edison con la BreakBoard tal y como viene explicado en el [tutorial](#) de la página de Intel, hay que instalar el sistema operativo Ubilinux. El motivo es que el sistema operativo que viene por defecto, Yocto, no posee algunas herramientas necesarias para instalar ROS que si tiene Ubilinux, como el gestor de paquetes *apt-get*.

Para instalar Ubilinux hay que bajarse la imagen disponible en la página oficial de [Ubilinux](#) y descomprimirlo en cualquier directorio que el usuario desee.

1.1. Eliminación de contenido innecesario

Una vez está conectada la Intel Edison al ordenador, se elimina cualquier archivo innecesario que pueda tener la Intel Edison. Para ello se abre un terminal y se ejecutan los siguientes comandos:

```
$ cd /media/<username>
$ ls -lag Edison
$ rm -rf Edison/*
$ ls -lag Edison
```

Estos comandos en principio no son necesarios, pero cuando se va a instalar otro sistema operativo o reinstalar el existente podría ser necesario ejecutarlos.

1.2. Instalación de programas necesarios

Para llevar a cabo los siguientes pasos es necesario instalar un par de programas en el ordenador: *dfu-util* y *screen*. El programa *dfu-util* permite instalar Ubilinux en la Intel Edison a través del USB y el programa *screen* permite acceder de forma remota a la Intel Edison por comunicación serie. Para instalarlos hay que ejecutar los siguientes comandos:

```
$ sudo apt-get install dfu-util
$ sudo apt-get install screen
```

Con estos programas ya puede comenzar la instalación de Ubilinux

1.3. Instalación de Ubilinux en la Intel Edison

Lo primero de todo es desconectar el cable de alimentación de la Intel Edison, pero no el de comunicación serie. Una vez hecho esto, se abre un nuevo terminal y se inicia el programa *screen* para iniciar comunicación serie por el puerto que está conectado el cable de comunicación serie. El comando a ejecutar es el siguiente:

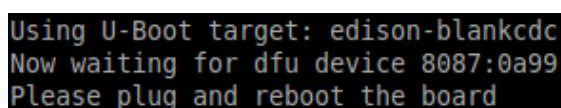
```
$ sudo screen /dev/ttyUSB0 115200
```

NOTA: Pulsar INTRO tras ejecutar el comando. Los argumentos de este comando son el puerto utilizado (/dev/ttyUSB0) y la velocidad en baudios de la comunicación (115200 baudios).

A continuación se inicia la instalación de Ubinlinux ejecutando los siguientes comandos en el primer terminal que se abrió:

```
$ cd <ruta_de_extracción>/toFlash  
$ sudo ./flashall
```

El instalador muestra por pantalla la petición de alimentar de nuevo la placa (Figura 1.1).

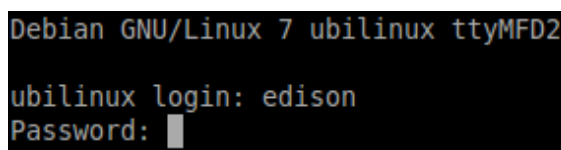


```
Using U-Boot target: edison-blankcdc  
Now waiting for dfu device 8087:0a99  
Please plug and reboot the board
```

Figura 1.1: Alimentar la Intel Edison.

Al conectarla empieza la instalación que tardará en torno a 10 - 15 minutos.

Un detalle muy importante es que al acabar la instalación, la Intel Edison se reinicia y muestra un mensaje pidiendo que no se desenchufe en 2 minutos para evitar fallos graves. La utilidad del terminal que esta ejecutando *screen* es la de comprobar cuando termina este proceso y es que cuando esto ocurra, mostrará por pantalla un pantalla pidiendo el usuario y la contraseña (Figura 1.2).



```
Debian GNU/Linux 7 ubilinux ttyMFD2  
ubilinux login: edison  
Password: █
```

Figura 1.2: Logeo en la Intel Edison con *screen*.

NOTA: El usuario y su contraseña son:

- **Usuario:** edison.
- **Contraseña:** edison.

Una vez dentro de la intel se modificará el archivo que permite conectarse a la Intel Edison a una red WiFi. Para ello hay que ejecutar los siguientes comandos:

```
$ su  
$ vim /etc/network/interfaces
```

NOTA: Contraseña para su: edison.

Hay que modificar el archivo para que el usuario se pueda conectar a ella vía WiFi con *ssh*. Para ello hay que deshabilitar la opción de conexión USB añadiéndole un # delante (Figura 1.3) y habilitar la opción de conexión WiFi (Figura 1.4) borrandoselo.

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto usb0
iface usb0 inet static
    address 192.168.2.15
    netmask 255.255.255.0

#auto wlan0
iface wlan0 inet dhcp
    # For WPA
    wpa-ssid Emutex
    wpa-psk passphrase
    # For WEP
    #wireless-essid Emutex
    #wireless-mode Managed
    #wireless-key s:password
# And the following 4 lines are for when using hostapd...
#auto wlan0
#iface wlan0 inet static
#    address 192.168.42.1
#    netmask 255.255.255.0
```

Figura 1.3: Archivo interfaces sin modificar.

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

#auto usb0
iface usb0 inet static
    address 192.168.2.15
    netmask 255.255.255.0

auto wlan0
iface wlan0 inet dhcp
    # For WPA
    wpa-ssid nombre_de_red
    wpa-psk contraseña
    # For WEP
    #wireless-essid Emutex
    #wireless-mode Managed
    #wireless-key s:password
# And the following 4 lines are for when using hostapd...
#auto wlan0
#iface wlan0 inet static
#    address 192.168.42.1
#    netmask 255.255.255.0
```

Figura 1.4: Archivo interfaces modificado.

NOTA: Algunos comandos imprescindibles para usar vim:

- i iniciar la edición de texto.
- ESC finalizar la edición de texto.
- :q cerrar vim.
- :wq guardar cambios y cerrar vim.
- :q! cerrar vim sin guardar cambios.

Una vez se ha modificado el archivo hay que cargar la configuración que permite conectarse a la Intel Edison a una red WiFi, para eso se escribe el siguiente comando:

```
$ ifup wlan0
```

Se espera hasta que se conecte a la red configurada en el archivo `interfaces` y se comprueba que la conexión funciona correctamente:

```
$ ping www.google.es
```

NOTA: Cuando se ejecuta este comando esperar unos segundos y cortar su ejecución con CTRL + C. Cuando se corte la ejecución debe aparecer información sobre como se desarrolla la conexión, si hay un 0 % de paquetes perdidos, la conexión funciona perfectamente.

Finalmente, hay que comprobar la dirección IP asignada para la configuración `wlan0` para poder conectarse a la Intel Edison posteriormente con `ssh`. Para ello nos fijamos en la información que proporciona sobre `wlan0` (Figura 1.5) el siguiente comando:

```
$ ifconfig
```

Una vez conocida la dirección IP, se cierra la conexión con la Intel Edison y el terminal que se comunica por USB con la Intel Edison. En el terminal que queda abierto se ejecuta el siguiente comando para conectarse vía WiFi a la Intel Edison:

```
$ sudo ssh edison@<Direccion_IP>
```

NOTA: En ocasiones este comando mostrará un error indicando que la conexión no es posible por un problema con el host. Para solucionarlo hay que fijarse en el mensaje de error, ya que proporciona el comando (Figura 1.6) que hay que introducir para resolverlo.

Una vez hecho todo esto, cada vez que se encienda la Intel Edison, el usuario podrá conectarse directamente a ella vía WiFi con `ssh`.

```
root@ubilinux:/home/edison# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlan0     Link encap:Ethernet  HWaddr 78:4b:87:a3:ab:2e
          inet addr:192.168.2.15  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::7a4b:87ff:fea3:ab2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:186 errors:0 dropped:0 overruns:0 frame:0
          TX packets:141 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:26224 (25.6 KiB)  TX bytes:18060 (17.6 KiB)
```

Figura 1.5: Dirección IP de wlan0.

```
remove with: ssh-keygen -f "/root/.ssh/known hosts" -R 192.168.2.16
```

Figura 1.6: Modificar lista de hosts conocidos.

2. Instalación de la librería MRAA

Para poder utilizar los pines de la Intel Edison es necesario emplear una librería capaz de acceder a ellos y configurarlos, de esta tarea se encarga la librería MRAA.

Esta librería viene instalada por defecto en la Intel Edison, pero en una versión antigua. Para actualizarla es necesario eliminar la versión que viene instalada por defecto e instalar la versión más reciente, para ello hay que ejecutar los siguientes comandos:

```
$ apt-get update
$ apt-get upgrade
$ cd
$ rm -rf mraa
$ apt-get install swig
$ git clone https://github.com/intel-iot-devkit/mraa.git
$ mkdir mraa/build && cd $_
$ cmake .. -DBUILD_SWIGNODE=OFF
$ make
$ make install
$ cd
```

Una vez instalada la librería a la última versión hay que configurar el sistema para que pueda ser encontrada por el compilador, esto se puede hacer añadiendo la línea `/usr/local/lib/i386-linux-gnu/` al final del archivo `ld.so.conf`:

```
$ vim /etc/ld.so.conf
$ ldconfig
```

Tras esto se puede comprobar que el sistema ya tiene localizada la librería (Figura 2.1) con el siguiente comando:

```
$ ldconfig -p | grep mraa
```

```
root@ubilinux:/home/edison# ldconfig -p | grep mraa
libmraa.so.0 (libc6) => /usr/local/lib/i386-linux-gnu/libmraa.so.0
libmraa.so (libc6) => /usr/local/lib/i386-linux-gnu/libmraa.so
```

Figura 2.1: ruta de libmraa.

Con la librería MRAA ya configurada hay que configurar los 2 pines que dispone la Intel Edison para implementar un bus I2C. Para configurarlos es necesario seguir los pasos expuestos en el manual del hardware que viene con la Intel Edison. Estos pasos consisten en introducir una serie de comandos para configurar los 2 pines como SDA y SCL respectivamente. Los comandos a introducir son:

```
# echo 28 > /sys/class/gpio/export
# echo 27 > /sys/class/gpio/export
# echo 204 > /sys/class/gpio/export
# echo 205 > /sys/class/gpio/export
# echo 236 > /sys/class/gpio/export
# echo 237 > /sys/class/gpio/export
# echo 14 > /sys/class/gpio/export
# echo 165 > /sys/class/gpio/export
# echo 212 > /sys/class/gpio/export
# echo 213 > /sys/class/gpio/export
# echo 214 > /sys/class/gpio/export
# echo low > /sys/class/gpio/gpio214/direction
# echo low > /sys/class/gpio/gpio204/direction
# echo low > /sys/class/gpio/gpio205/direction
# echo in > /sys/class/gpio/gpio14/direction
# echo in > /sys/class/gpio/gpio165/direction
# echo low > /sys/class/gpio/gpio236/direction
# echo low > /sys/class/gpio/gpio237/direction
# echo in > /sys/class/gpio/gpio212/direction
# echo in > /sys/class/gpio/gpio213/direction
# echo mode1 > /sys/kernel/debug/gpio_debug/gpio28/current_pinmux
# echo mode1 > /sys/kernel/debug/gpio_debug/gpio27/current_pinmux
# echo high > /sys/class/gpio/gpio214/direction
```

Tras hacer estos pasos, los pines están configurados para ser utilizados en un bus I2C.

3. Instalación de ROS en Ubilinux

A diferencia de Ubuntu, ROS no está totalmente integrado en Ubilinux, por lo que su instalación no está tan automatizada y es necesario instalar muchas de sus dependencias y paquetes a mano. Los siguientes pasos a realizar son muy parecidos a los pasos que hay que hacer para instalar ROS en otras placas que utilizan Debian Wheezy.

3.1. Instalación de dependencias y paquetes necesarios

Antes de instalar ROS, hay que instalar una serie de librerías

```
$ apt-get update
$ apt-get install liblz4-dev
$ sh -c 'echo "deb http://packages.ros.org/ros/ubuntu wheezy main"
> /etc/apt/sources.list.d/ros-latest.list'
$ wget https://raw.githubusercontent.com/ros/rosdistro/master/ros.
key -0 - | apt-key add -
$ apt-get update
$ apt-get upgrade
$ dpkg-reconfigure tzdata //reconfigurar la fecha
$ apt-get install python-setuptools python-pip python-yaml python-
argparse python-distribute python-docutils python-dateutil python-
setuptools python-six
$ pip install rosdep rosinstall_generator wstool rosinstall
$ rosdep init
$ rosdep update
$ apt-get install python-empy
$ apt-get install python-nose
$ apt-get install libboost-filesystem1.49-dev
$ apt-get install libboost-program-options1.49-dev
$ apt-get install libboost-regex1.49-dev
$ apt-get install libbz2-dev
$ apt-get install libboost-signals1.49-dev
$ apt-get install python-mock
$ apt-get install libboost-all-dev
$ apt-get install python-netifaces
$ apt-get install python-rosdep
```

3.2. Herramientas y archivos necesarios para la instalación

A continuación se instalan varias herramientas y archivos necesarios para instalar ROS Indigo en la Intel Edison:

```
$ cd /home/edison/
$ mkdir ros_catkin_ws
$ cd ros_catkin_ws
```

```
$ rosinstall_generator ros_comm --rostdistro indigo --deps --wet-  
only --exclude roslisp --tar > indigo-ros_comm-wet.rosinstall  
$ wstool init src indigo-ros_comm-wet.rosinstall  
$ mkdir external_src  
$ apt-get install checkinstall cmake  
$ sh -c 'echo "deb-src http://mirrordirector.raspbian.org  
/raspbian/ testing main contrib non-free rpi" >> /etc/apt  
/sources.list'  
$ apt-key adv --keyserver keyserver.ubuntu.com --recv-keys  
9165938D90FDDD2E  
$ apt-get update  
$ cd external_src
```

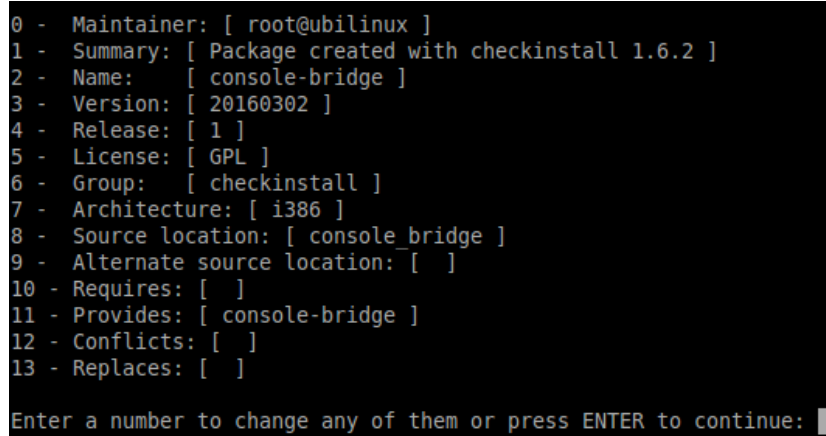
3.3. Instalación de librerías externas esenciales para ROS

Las librerías que se van a instalar a continuación son esenciales para ROS, ya que añaden funciones muy útiles.

3.3.1. Console bridge

Esta librería permite mostrar al usuario información relativa a lo que esta pasando mediante avisos, errores o información general al ejecutar algún proceso de ROS.

NOTA: Cuando se ejecuta `checkinstall make install` (Figura 3.1), teclear 2 cuando pida introducir un número y poner como nuevo nombre `libconsole-bridge-dev`.



```
0 - Maintainer: [ root@ubinux ]  
1 - Summary: [ Package created with checkinstall 1.6.2 ]  
2 - Name: [ console-bridge ]  
3 - Version: [ 20160302 ]  
4 - Release: [ 1 ]  
5 - License: [ GPL ]  
6 - Group: [ checkinstall ]  
7 - Architecture: [ i386 ]  
8 - Source location: [ console_bridge ]  
9 - Alternate source location: [ ]  
10 - Requires: [ ]  
11 - Provides: [ console-bridge ]  
12 - Conflicts: [ ]  
13 - Replaces: [ ]  
  
Enter a number to change any of them or press ENTER to continue: █
```

Figura 3.1: Cambio de nombre con checkinstall.

```
$ apt-get install libboost-system-dev libboost-thread-dev  
$ git clone https://github.com/ros/console_bridge.git  
$ cd console_bridge  
$ cmake .
```

```
$ checkinstall make install
$ make install
```

3.3.2. URDF

Las siguientes librerías constituyen el parser utilizado para Unified Robot Description Format (URDF), un método de representación de modelos robóticos en formato XML.

NOTA: Cuando se ejecuta `checkinstall make install`, teclear 2 cuando pida introducir un número y poner como nuevo nombre `liburdfdom-headers-dev`.

```
$ cd ..
$ git clone https://github.com/ros/urdfdom_headers.git
$ cd urdfdom_headers
$ cmake .
$ checkinstall make install
```

NOTA: Cuando se ejecuta `checkinstall make install`, teclear 2 cuando pida introducir un número y poner como nuevo nombre `liburdfdom-dev`.

```
$ cd ..
$ apt-get install libboost-test-dev libtinyxml-dev
$ git clone https://github.com/ros/urdfdom.git
$ cd urdfdom
$ cmake .
$ checkinstall make install
```

3.3.3. Collada

Collada es un formato de archivos 3D desarrollado a partir de XML, por lo que está especialmente indicado para representar los modelos definidos con URDF.

NOTA: Cuando se ejecuta `checkinstall make install`, teclear 2 cuando pida introducir un número y poner como nuevo nombre `collada-dom-dev`.

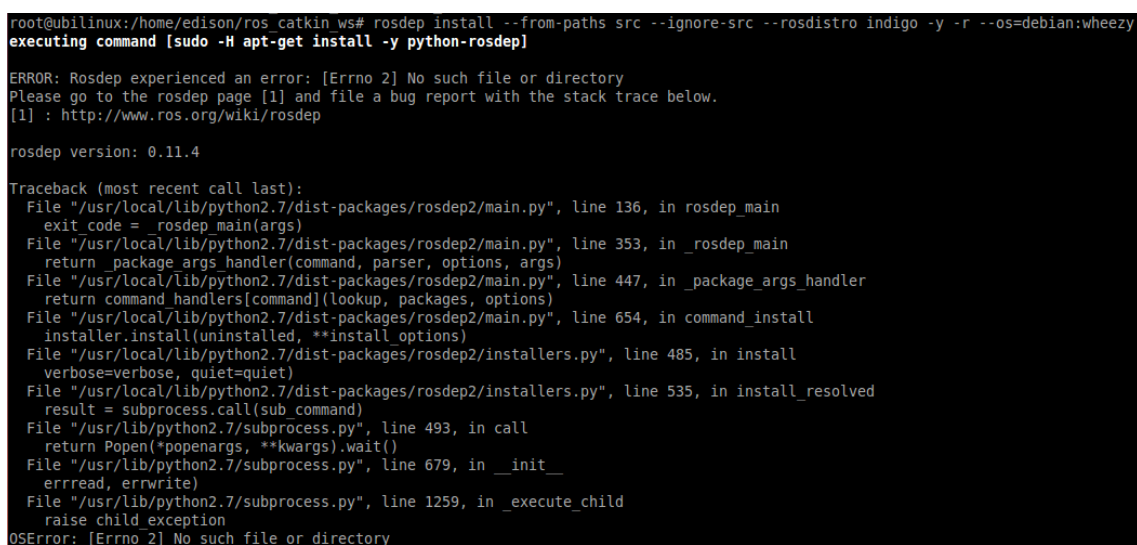
```
$ cd ..
$ apt-get install libboost-filesystem-dev libxml2-dev
$ wget http://downloads.sourceforge.net/project/collada-dom/
Collada%20DOM/Collada%20DOM%202.4/collada-dom-2.4.0.tgz
$ tar -xzf collada-dom-2.4.0.tgz
$ cd collada-dom-2.4.0
$ cmake .
$ checkinstall make
$ make install
```

3.4. Instalación de ROS

Con todos los paquetes y librerías anteriores instalados se puede proceder a instalar finalmente ROS con los siguientes comandos:

```
$ cd ../../..
$ rosdep install --from-paths src --ignore-src --rosdistro
indigo -y -r --os=debian:wheezy
```

NOTA: Cuando se ejecuta este comando devuelve un mensaje de error (Figura 3.2) advirtiéndole que falta uno paquete por instalar. En realidad este paquete ya se instaló en el apartado 3.1 y según los desarrolladores es un error sin importancia que no altera el funcionamiento normal de ROS.



```
root@ubinux:/home/edison/ros_catkin_ws# rosdep install --from-paths src --ignore-src --rosdistro indigo -y -r --os=debian:wheezy
executing command [sudo -H apt-get install -y python-rosdep]
ERROR: Rosdep experienced an error: [Errno 2] No such file or directory
Please go to the rosdep page [1] and file a bug report with the stack trace below.
[1] : http://www.ros.org/wiki/rosdep
rosdep version: 0.11.4
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/rosdep2/main.py", line 136, in rosdep_main
    exit_code = _rosdep_main(args)
  File "/usr/local/lib/python2.7/dist-packages/rosdep2/main.py", line 353, in _rosdep_main
    return _package_args_handler(command, parser, options, args)
  File "/usr/local/lib/python2.7/dist-packages/rosdep2/main.py", line 447, in _package_args_handler
    return command_handlers[command](lookup, packages, options)
  File "/usr/local/lib/python2.7/dist-packages/rosdep2/main.py", line 654, in command_install
    installer.install(uninstalled, **install_options)
  File "/usr/local/lib/python2.7/dist-packages/rosdep2/installers.py", line 485, in install
    verbose=verbose, quiet=quiet)
  File "/usr/local/lib/python2.7/dist-packages/rosdep2/installers.py", line 535, in install_resolved
    result = subprocess.call(sub_command)
  File "/usr/lib/python2.7/subprocess.py", line 493, in call
    return Popen(*popenargs, **kwargs).wait()
  File "/usr/lib/python2.7/subprocess.py", line 679, in _init_
    errread, errwrite)
  File "/usr/lib/python2.7/subprocess.py", line 1259, in _execute_child
    raise child_exception
OSError: [Errno 2] No such file or directory
```

Figura 3.2: Error en rosdep.

```
$ ./src/catkin/bin/catkin_make_isolated --install-DCMAKE_
BUILD_TYPE=Release --install-space /opt/ros/indigo
```

Cuando la instalación finalice hay que realizar unos pasos adicionales para terminar de configurar ROS. Para ello se introducen los siguientes comandos:

```
$ source /opt/ros/indigo/setup.bash
$ export ROS_HOSTNAME=ubinux
$ export ROS_MASTER_URI=http://ubinux:11311
$ vim /etc/hosts
```

Se modifica el archivo `hosts` para que detecte al propio Ubinux (Figura 3.3) y finalmente se puede comprobar que se ha instalado ROS Indigo y ejecutarlo:

```
$ rosversion -d
$ roscore
```

```
127.0.0.1 localhost
127.0.0.1 ubilinux
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Figura 3.3: Identificación de hosts.

4. Ejemplo de comunicación I2C

En este ejemplo se emplea una Intel Edison como maestro del bus I2C y un Arduino Uno como esclavo. El circuito (Figura 4.1) funciona de tal forma que el Arduino Uno lee la señal analógica que proporciona el potenciómetro, la envía por el bus I2C hasta la Intel Edison y esta la vuelve a mandar por el bus I2C hasta el Arduino.

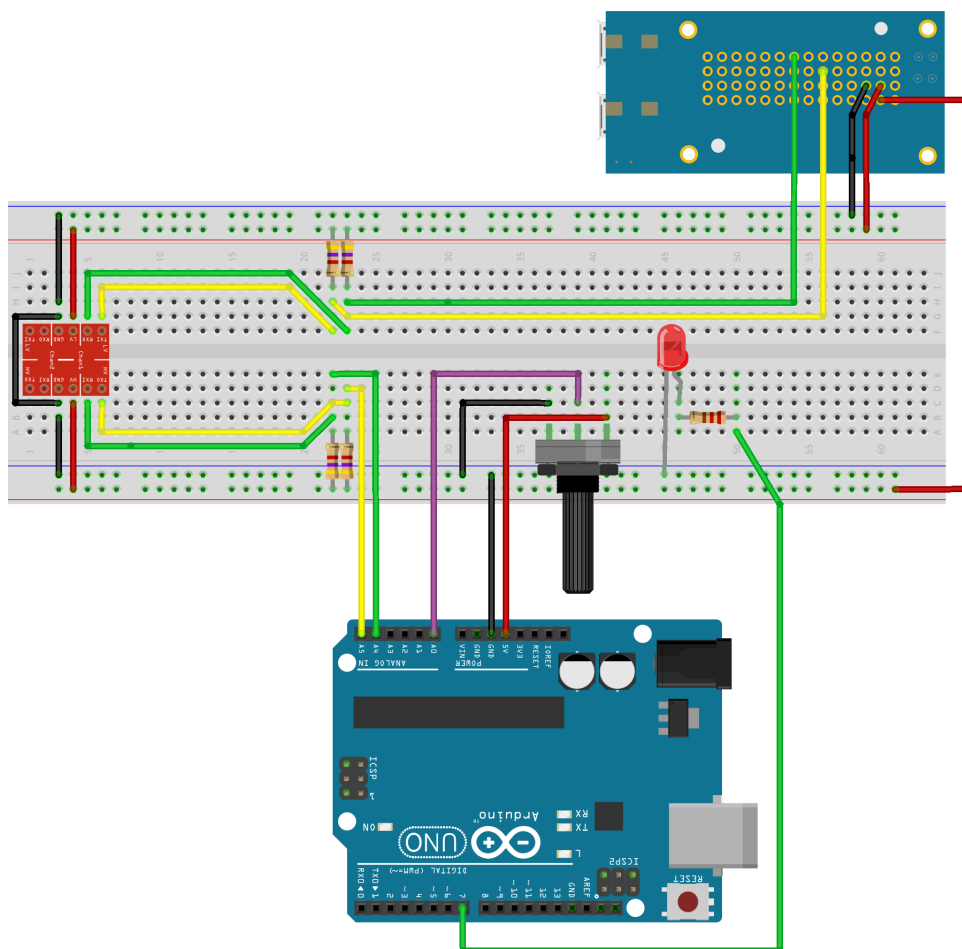


Figura 4.1: Circuito.

4.1. Componentes del circuito

Los componentes necesarios para montar el circuito son:

- Resistencia de $4k7\ \Omega$ x 4.
- Resistencia de $120\ \Omega$ x 1.
- LED Rojo x 1.
- Level Shifter (1.8 V - 3.3 V) x 1.
- Arduino Uno x 1.
- Intel Edison (con breakboard) x 1.
- Protoboard x 1

4.2. Instalación del programa en la Intel Edison

Para instalar el paquete `maestro_i2c` que creara el maestro en el bus I2C es necesario crear un workspace de ROS siguiendo estos pasos:

```
$ cd /home/edison/  
$ mkdir -p ros_ws/src  
$ cd ros_ws/src  
$ catkin_init_workspace  
$ cd ..  
$ catkin_make  
$ source devel/setup.bash
```

Ahora hay que copiar el paquete desde el ordenador hasta la Intel Edison:

```
$ sudo scp -r maestro_i2c edison@<Direccion_IP>:
```

De nuevo en la Intel Edison, hay que colocar el paquete en la carpeta `src` del workspace y compilarlo:

```
$ cd /home/edison/  
$ cp -r maestro_i2c ros_ws/src  
$ cd ros_ws/  
$ catkin_make
```

NOTA: Para más información sobre como crear un workspace de ROS, visitar la [wiki de ROS](#).

Una vez hecho esto, el programa del maestro estará listo para ser ejecutado en la Intel Edison.

4.3. Ejecución del programa

Con el paso anterior realizado solo queda cargar el programa del esclavo a la placa de Arduino con el IDE. El esclavo empezará a ejecutarse, pero para ejecutar el maestro es necesario ejecutar el siguiente comando en la Intel Edison:

```
$ rosrun maestro_i2c maestro_i2c
```

Una vez que empiece a ejecutarse nos mostrará por pantalla si la conexión con el esclavo ha podido realizarse. Si es así, el usuario podrá ir variando la frecuencia de parpadeo del LED moviendo el potenciómetro.

NOTA: El código tanto para el Arduino Uno como para la Intel Edison se proporcionan junto al manual.