Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Desarrollo de un sistema basado en ROS (Robotic Operating System) para teleoperar un vehículo agrícola e integración de sensor filoguiado para navegación autónoma.

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido

Fecha: 4 de julio de 2014

Índice Pág.

Tema

1. Introducción
1.1.Abstract 4
1.2.Presentación del proyecto 4
1.3.Objetivos. 5

2. Descripción del sistema
2.1. Descripción del sistema hardware 7

2.2.Comunicación del sistema 10

2.3.Sistema sensorial del carro
2.3.1. Introducción a los subsistemas del vehículo. 12
2.3.2. Filoguiado 13
2.3.3. Cable guía 14
2.3.4. Sensores 15
2.3.5. Diseño PCB 22
2.3.6. Procesado de la señal de los sensores 27

2.4.Descripción del sistema Software
2.4.1. Sistemas operativos 28
2.4.2. ROS, *Robotic Operating System.* 28

3. Resultados y conclusiones
3.1. Resultados y conclusiones 35
3.2. Results and conclusions 36

4. Mejoras y trabajos futuros 36

5. Referencias 37

6. Anexos

6.1. Anexo 1. Instalación de Ubuntu en PC.
6.2. Anexo 2. Instalación de Ubuntu en Beaglebone.
6.3. Anexo 3. Instalación de Angstrom en Beaglebone.
6.4. Anexo 4. Instalación de Raspbian en Raspberry pi
6.5. Anexo 5. Instalación de ROS en PC con Ubuntu.
6.6. Anexo 6. Instalación de ROS en Beaglebone con Ubuntu.
6.7. Anexo 7. Instalación de ROS en Beaglebone con Angstrom.
6.8. Anexo 8. Instalación de ROS en Raspberry pi con Raspbian.
6.9. Anexo 9. ROS. Conceptos básicos.
6.10. Anexo 10. Nodos en ROS.
6.11. Anexo 11. Maestro esclavo en ROS.
6.12. Anexo 12. Descripción de Beaglebone.
6.13. Anexo 13. Descripción Raspberry.
6.14. Anexo 14. Descripción Mbed.

1. Introducción

1.1 Abstract

The project develops the designs and implementation of an AGV (Automatic Guided Vehicle), too called "Paca", with the aim to help workers of Finca Las Lucanas and make the heavy works easier between outdoor and indoor points of the greenhouse. The automatic guided vehicle using a magnetic wired to improve the transport line in a farming greenhouse. The system has sensors, motors, and microprocessors for control, pick up information, analyze and act in the environment working with ROS.

1.2 Presentación

Este trabajo fin de grado (TFG) se enmarca dentro de un proyecto que se está desarrollando entre finca Las Lucanas y la ULL.

Figura 1. Vehículo objeto del TFG.

Con el objetivo de ayudar al personal de la empresa y para facilitar labores de carga entre puntos interiores y exteriores del vivero. El proyecto desarrolla el diseño y la implementación de un AGV *(Automatic Guided Vehicle)*, también denominado *"Paca"*, un vehículo de guiado automático por medio de un hilo magnético para mejorar la línea de

transporte en un invernadero de producción agrícola. El sistema está compuesto por sensores, actuadores y micro controladores para controlar y recoger información, analizarla y actuar en el entorno.

Aunque "Paca" actualmente está funcionando con ciertos recursos hardware/software, somos conscientes que el sistema necesita ciertas mejoras, tales como microprocesadores más potentes y con más recursos y sistemas operativos específicos para robotizar el vehículo (ROS).

1.3 Objetivo

El objetivo de nuestro trabajo es desarrollar una plataforma basada en ROS (*Robotic Operating System)*, que sirva para dar soporte al vehículo y que a la vez sirva para posibilitar mejoras en un futuro.

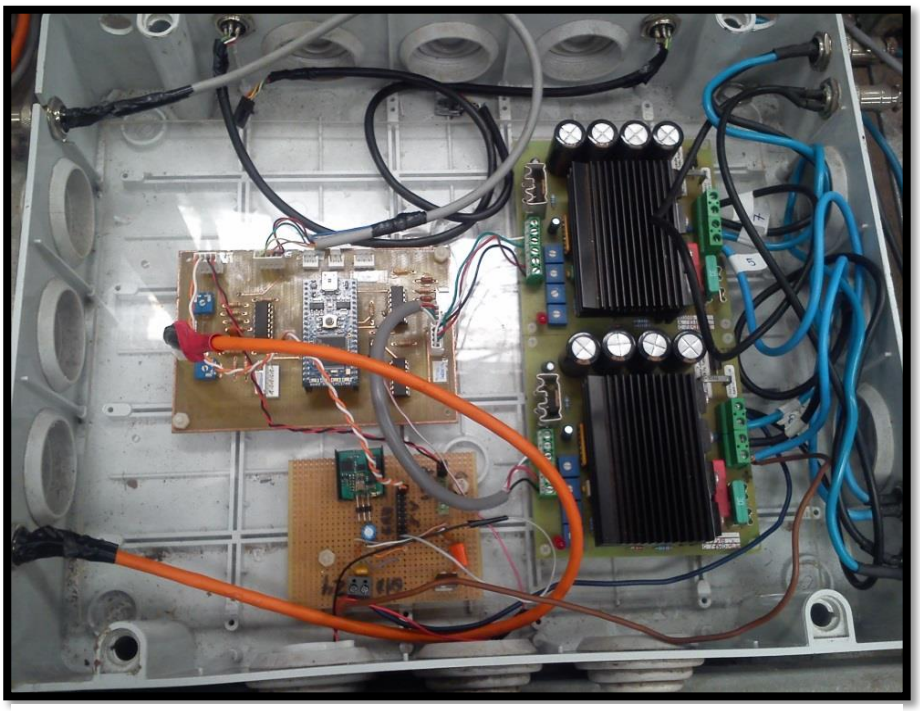


Figura 2. Sistema electrónico antiguo.

Para desarrollar el trabajo empezaremos por seleccionar los sistemas computacionales que instalaremos en el vehículo que tendrán que soportar ROS y poder comunicarse entre sí para compartir datos.

Una vez tengamos esta elección hecha, el trabajo será instalar los sistemas operativos correspondientes, instalar ROS y luego enlazar los sistemas para su comunicación.

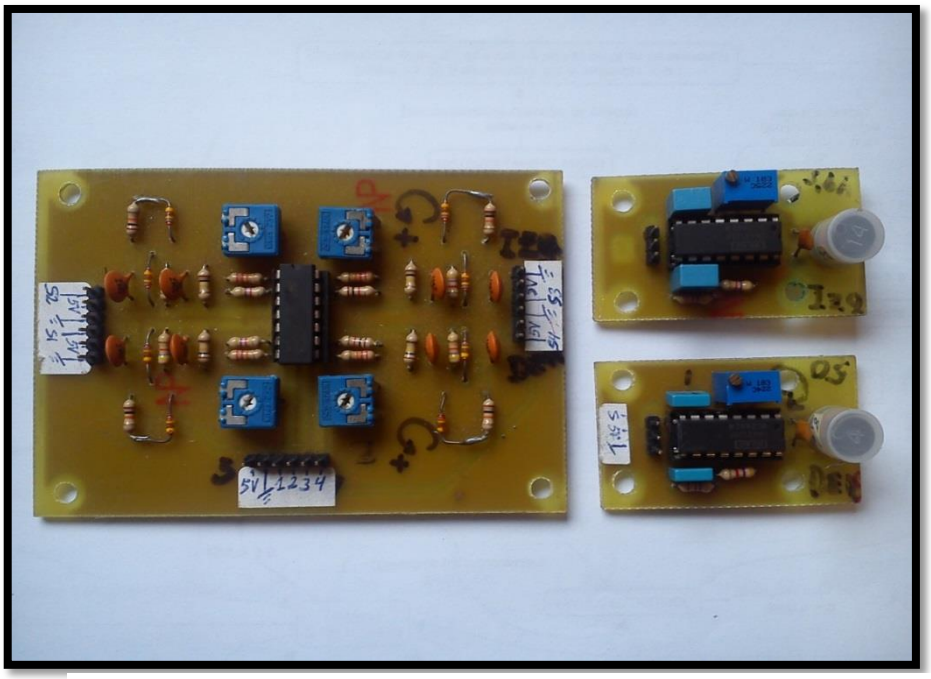


Figura 3. Sensores y placa de acondicionamiento.

Otro objetivo es, rediseñar el sensor magnético que hay instalado en el carro para mejorar su funcionamiento. Para ello habrá que estudiar el diseño inicial que se muestra en la figura 3, y analizar sus posibles mejoras.

En la figura 4 se muestra el antiguo soporte del sistema filoguiado, podemos ver dos sensores laterales de guiado y uno central para detectar cruces.

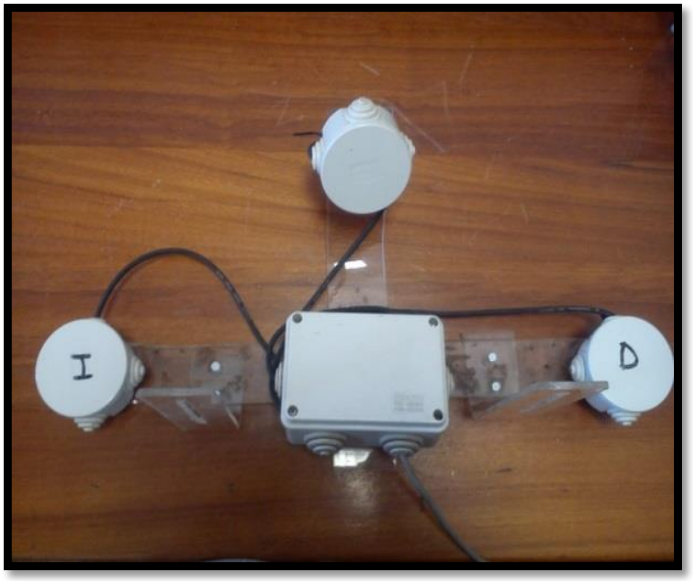


Figura 4. Soporte para los sensores y la placa de acondicionamiento.

2. Descripción del sistema

2.1 Descripción del sistema hardware.

El sistema hardware del vehículo es todo componente físico que se usa para llevar a cabo las comunicaciones, cálculos, recogida de información o cualquier acción necesaria para el funcionamiento correcto del vehículo.

En la arquitectura anterior del vehículo se basaba en un microcontrolador *Mbed* el cuál era el encargado de gestionar toda la información.

La figura 5 muestra dicha arquitectura. Esta se basaba en un procesador central, Mbed, gestionando la información de entrada que proviene del filoguiado, del mando PS3, del joystick y del selector de navegación y tras procesar la información daba las órdenes correspondientes a los motores.

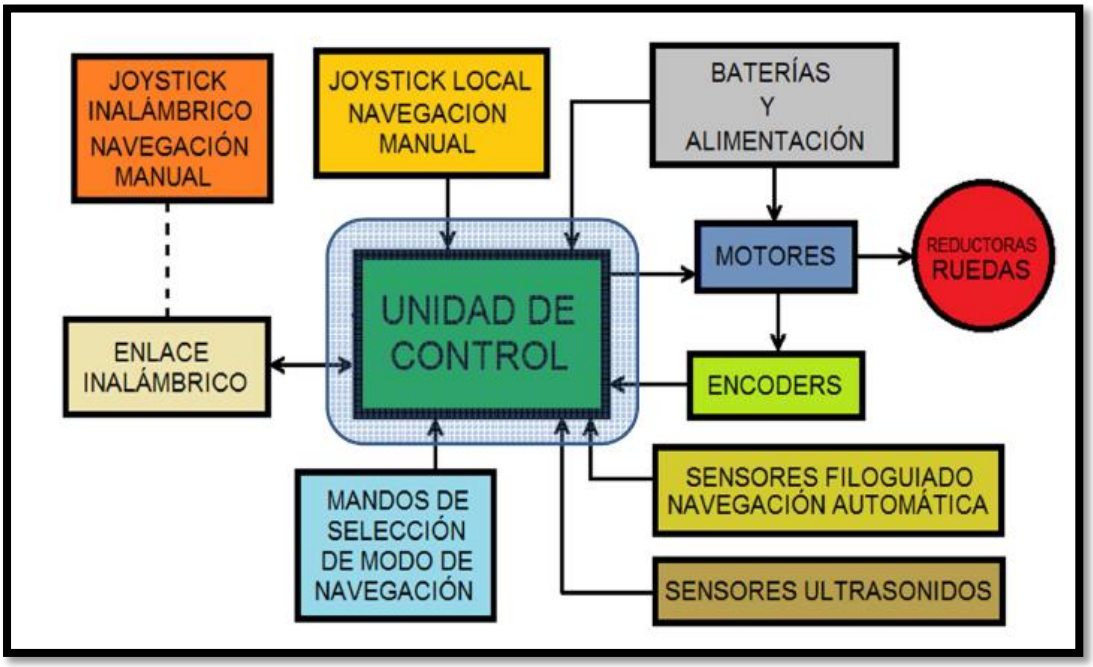


Figura 5. Esquema del sistema antes de modificarlo.

Las opciones que tiene este sistema para integrar mejoras son limitadas por lo que ante la necesidad de querer mejorar la funcionalidad hemos basado el sistema en ROS y consecuentemente hemos tenido que dotar al sistema de un hardware donde se pueda instalar dicho sistema operativo.

En la figura 6 se ve el nuevo modelo de abstracción del sistema donde podemos ver que está implementado con un soporte hardware formado por Beaglebone, Raspberry pi y Mbed. Este soporte hardware trabaja con un software Linux a excepción de las Mbed para tener un sistema operativo donde instalar nuestra plataforma ROS. Todo ello, es con el fin de garantizar el correcto funcionamiento de nuestra aplicación, un sistema de filoguiado.

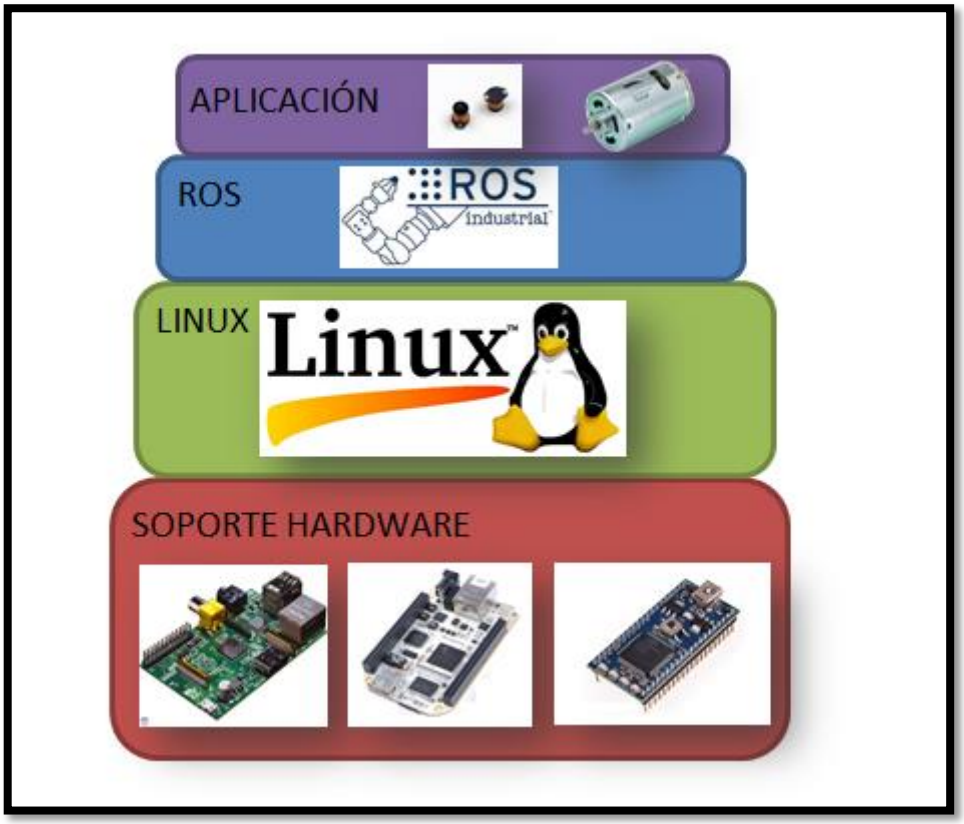


Figura 6. Nuevo modelo de abstracción

Inicialmente y como maestro del sistema tenemos un ordenador portátil trabajando sobre una plataforma Linux, más concretamente sobre un Ubuntu (Anexo 1). Este ordenador tiene instalado ROS (Anexo 5) y está conectado vía WiFi con "*Paca"* usando la red WiFi local de las instalaciones. En el vehículo se encuentra la Beaglebone (Anexo 12), la Raspberry pi (Anexo 13), las dos Mbed (Anexo 14). Todos estos, con nodos esclavos ROS (Anexo 6, 7 y 8).

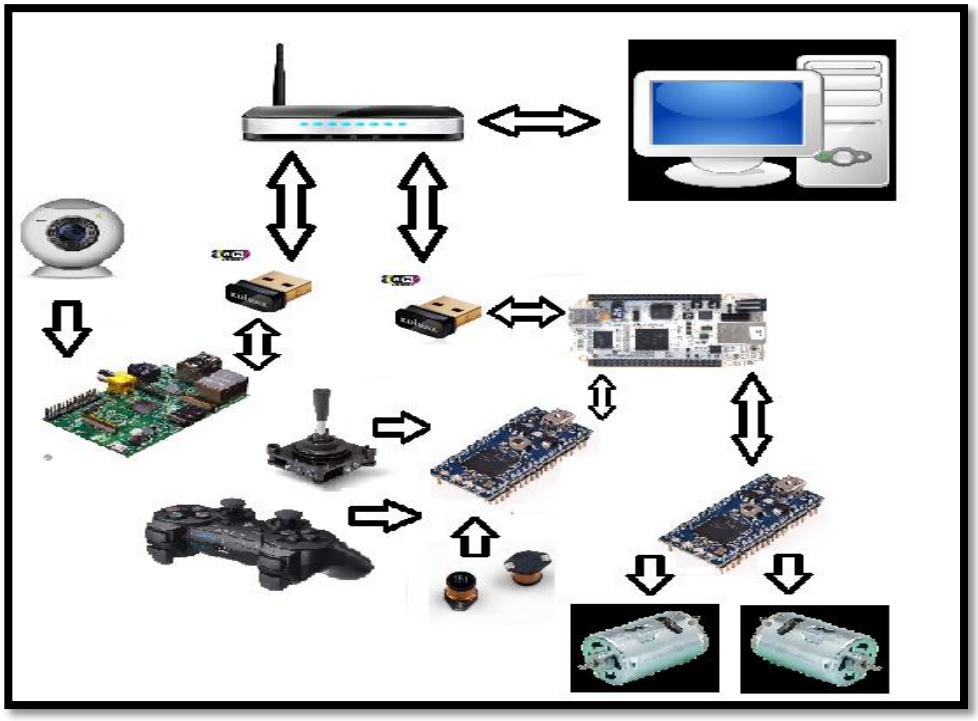


Figura 7. Esquema del diseño Hardware de nuestro objetivo

El vehículo también dispone de más subsistemas: los sensores, los motores, joystick para manejo manual, selector de modo de conducción y el receptor *bluetooth* así como unas baterías para su alimentación. La conexión WiFi se establece tanto en la Raspberry pi como en la Beaglebone mediante un WiFi USB.

El sistema hardware, como se puede apreciar en la figura 9, tiene una jerarquía. Esta jerarquía empieza en el PC con un ROS Master. Tanto la Raspberry pi como la Beaglebone son esclavos del PC. Esto quiere decir que se podrá tener un control total del vehículo desde el PC.

Además, se puede conectar una Raspicam. La Raspicam es la cámara comercializada para Raspberry. Esta tiene un nodo en ROS que fue instalado con errores. En la figura 8 se puede ver la cámara.

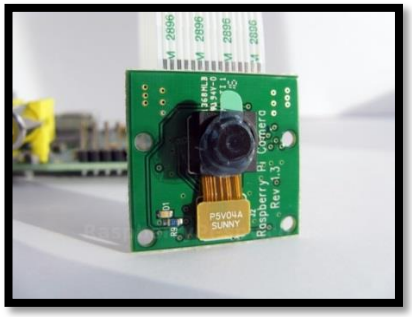


Figura 8. Cámara Raspicam

2.2 Comunicación del sistema

Con el fin de poder tener acceso desde el maestro a los esclavos ROS en todas las instalaciones de la finca. En las instalaciones se instalará una red local WiFi conectada a internet. Dicha red, tendrá un router en su oficina y conectado a este, repetidores distribuidos a lo largo del vivero.

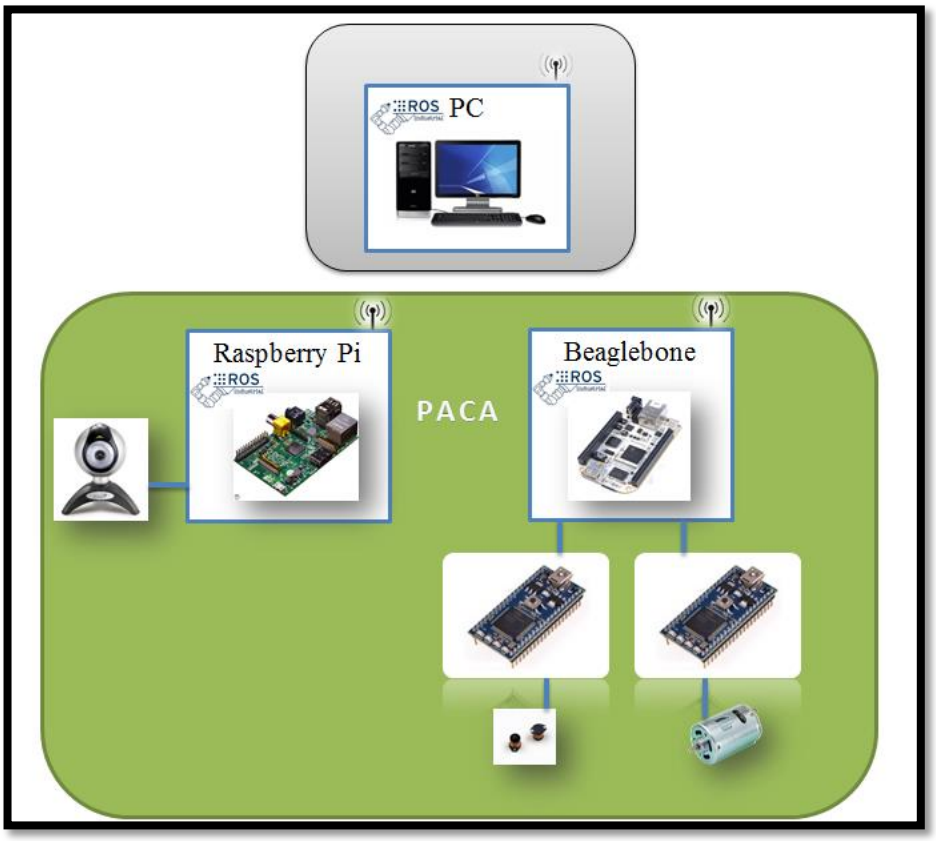


Figura 9. Visión general de nuestro sistema

En la configuración de la red se ha de tener en cuenta que tanto el sistema maestro en este caso el PC y sus dos esclavos; la BeagleBone y la Raspberry pi, deben tener una dirección IP estática asignada, para que nunca exista un error de comunicación.

El PC estará conectado a la red local ya sea mediante cable de Ethernet o vía WiFi, pudiendo así comunicarse tanto para enviar como recibir datos de sus dos esclavos, la Beaglebone y la Raspberry pi.

Estos esclavos se conectan a la red local, mediante el USB WiFi: EW-7811Un. Este ha sido seleccionado por ser uno de los más vendidos, siendo así quien más soporte tiene en internet y quien trabaja sobre más plataformas diferentes.

Figura 10. USB WiFi.

La Beaglebone se conecta a su vez a dos Mbed mediante conexión serial. Recibiendo de una de las Mbed datos con información obtenida por los sensores y enviando a la otra Mbed datos necesarios para el funcionamiento de los actuadores.

La Mbed a la que se le conecta los sensores tendrá un joystick inalámbrico y un joystick local, ambos para navegación manual del vehículo, sensores de filoguiado para una navegación autónoma, un mando para seleccionar el modo de navegación y en un futuro se implementará sensores de ultrasonidos, figura 11, para detección de objetos, animales o personas.

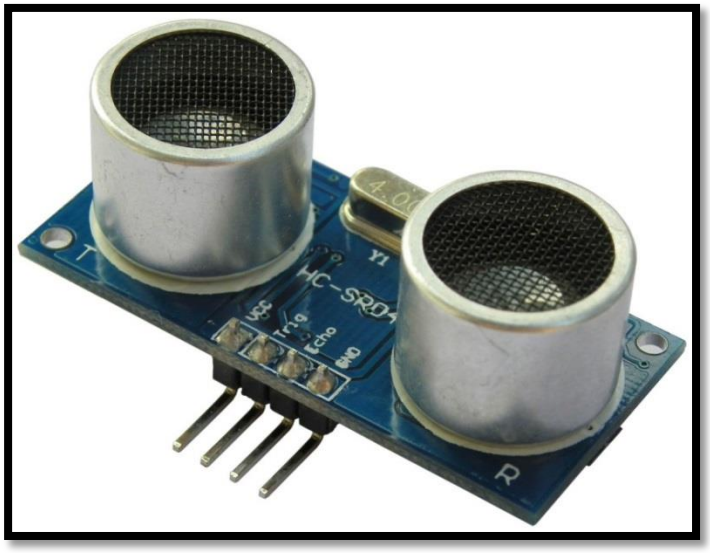


Figura 11. Sensores ultrasonido.

La Mbed a la se le conecta los motores: tendrá conectado los motores del vehículo y encoders para poder monitorizar los mismos motores.

2.3 Sistema sensorial

2.3.1 Filoguiado.

El filoguiado es un sistema de navegación utilizado en sistemas que no tienen contacto físico para tomar referencias. En todo sistema de navegación deben existir por un lado, un sistema de guía y un sistema que con la información que recoge de la guía lleve a cabo acciones de forma autónoma.

Los sistemas guía pueden ser de varios tipos, formas y colores. Todas las opciones se van acotando para finalmente elegir el más apropiado para cada aplicación ya que no todos los sistemas guías son iguales ni para las mismas condiciones de trabajo.

El filoguiado es un sistema basado en un campo magnético alrededor de un hilo conductor el cual va enterrado a pocos centímetros en los caminos como se muestra en la figura 12, por donde circula el vehículo. Este campo es medido por sensores mediante los cuales se toma decisiones de navegación. El campo magnético es generado un generador de campo basado en un oscilador que hace circular una corriente por el hilo conductor. El hilo conductor define los caminos por los que circulará el vehículo.

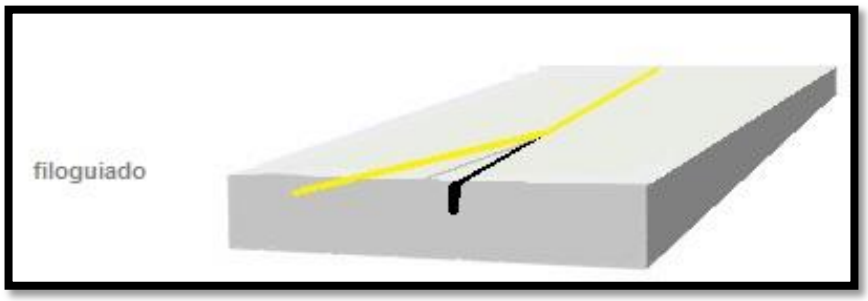


Figura 12. Detalle del circuito de filoguiado.

En cuanto al sensor que dispondremos para recoger la información del campo magnético, será un sensor basado en el principio teórico del magnetismo para recoger el campo y traducirlo en un nivel de tensión. Este nivel de tensión es proporcional a la distancia. Esto se consigue con un circuito tanque que al ser expuesto a un campo magnético se induce una corriente en la bobina del mismo.

2.3.2 Fundamento teórico.

Cuando circula una corriente eléctrica se crea un campo magnético. Si el hilo conductor es rectilíneo el campo forma circunferencias concéntricas al cable (figura 13). La intensidad

del campo magnético es inversamente proporcional a la distancia al hilo conductor donde se genera.

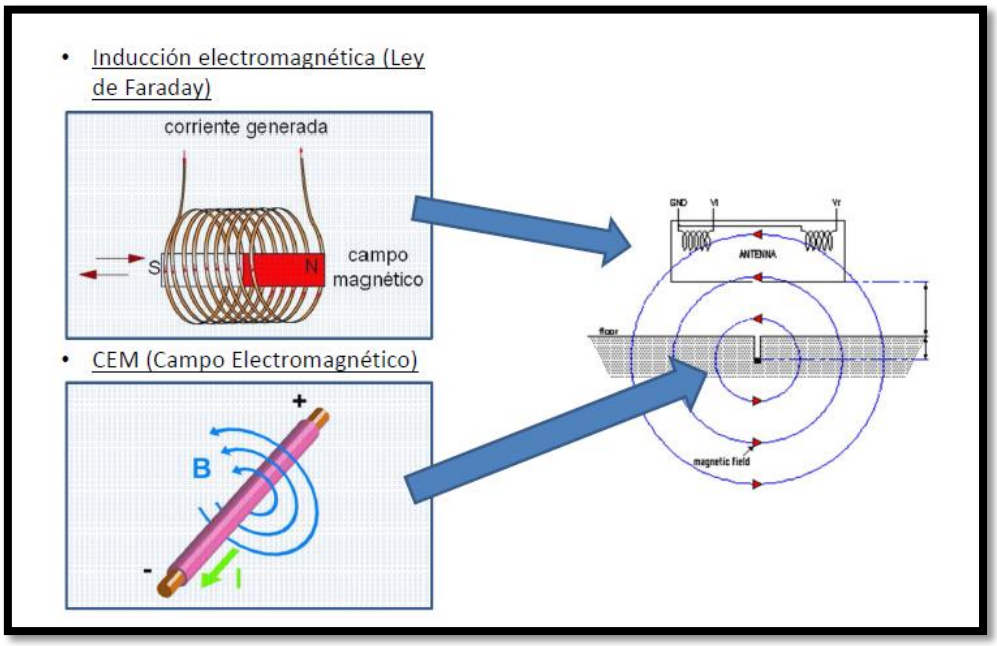


Figura 13. Explicación gráfica del campo magnético

La fórmula para obtener el campo magnético en nuestra aplicación es:

$$B = \frac{m\,I}{2\,\pi\,d}$$

Dónde:

- B: Campo magnético
- m: es la permeabilidad del aire
- I es la corriente por el cable
- Π es el número Pi.
- d es la distancia desde el cable.

Ahora, de igual manera que una corriente que atraviesa un conductor genera un campo, un campo que atraviesa un conductor genera una corriente. Es aquí donde se fundamenta nuestro sensor basado en una bobina orientada de forma que la atraviese el flujo de campo. La orientación debe ser tal que el campo magnético entre por el núcleo de la misma sino no se inducirá corriente alguna.

En la figura 14 se puede ver una representación del campo electromagnético y la colocación de las bobinas siendo la posición ideal la colocación superior ya que es de la forma en que más flujo la atraviesa.

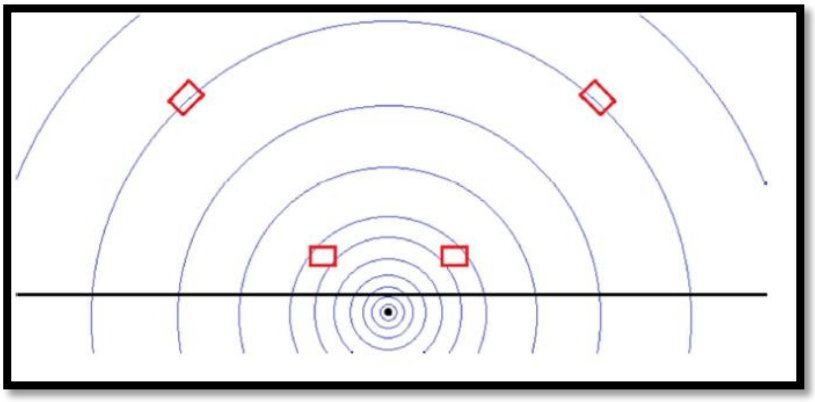


Figura 14. Campo magnético en un hilo conductor

2.3.3 Cable guía

El cable guía será un cable enterrado a lo largo de todo el recorrido que queramos habilitar para el recorrido del carro.

Este generador es un diseño basado en un generador de funciones, el ICL8083 que se ajusta mediante varios potenciómetros para controlar la forma de la onda. Luego, se amplifica la señal para que tenga intensidad suficiente para generar el campo que va a ser generado por el cable. En la figura 15 se ve la placa del generador de campo, vemos los cables de entrada de tensión por la derecha, siendo alimentada a +-12V, luego, en la parte superior, vemos el disipador del transistor y a la izquierda de la fotografía, la salida del cable para la generación del campo (bornes rojo y negro).

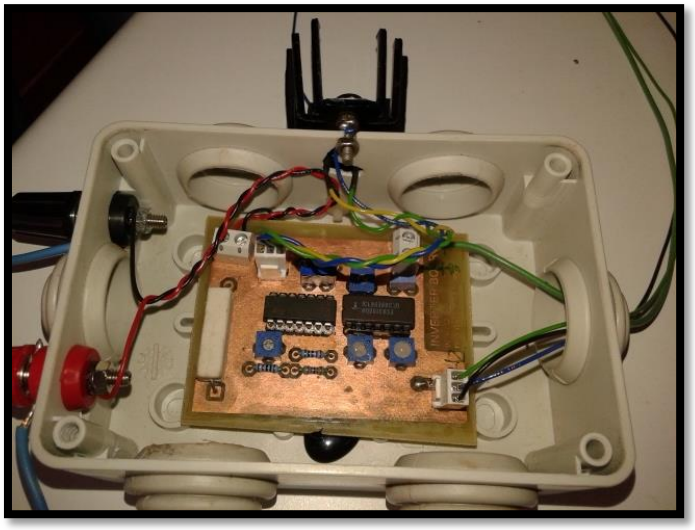


Figura 15. Generador de campo magnético

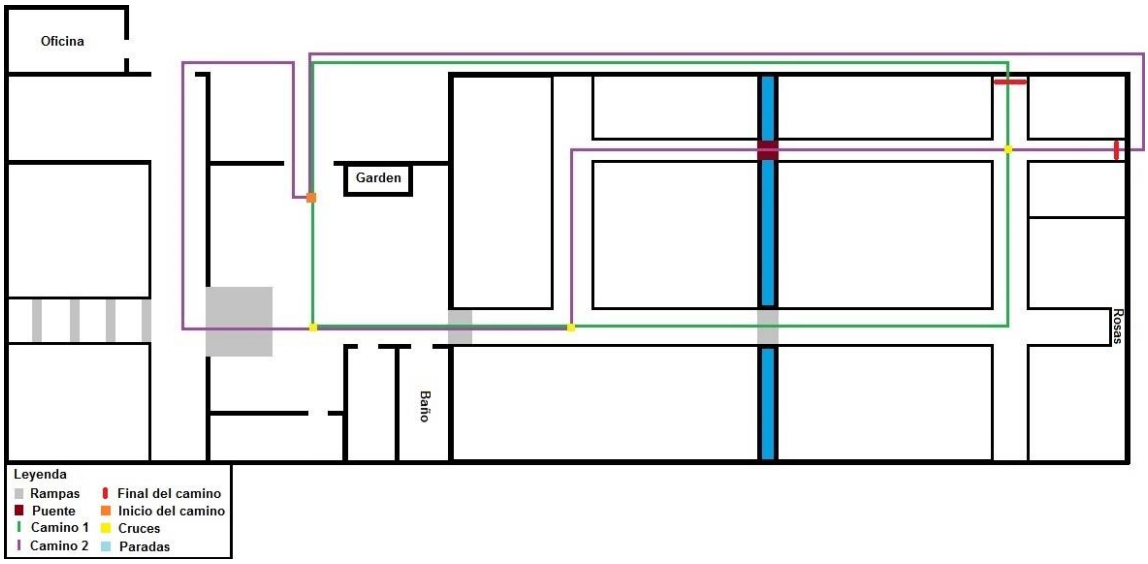


Figura 16. Mapa del espacio habilitado para el vehículo.

2.3.4Sensores

El circuito sensor que estaba instalado se basaba en un circuito tanque con un amplificador de instrumentación. Se disponía de un circuito por bobina y esta señal se llevaba a otro circuito acondicionador de señal antes de pasar a la *Mbed.*

Como comentamos al principio, la mejora de este circuito sensor es uno de nuestros objetivos. Tal es así que analizando el circuito nos percatamos que teniendo una de las salidas

del circuito tanque a tierra, este perdía su sentido ya que haciendo esto, no tendríamos un circuito tanque en sí sino la amplificación de una señal diferencial con respecto a tierra.

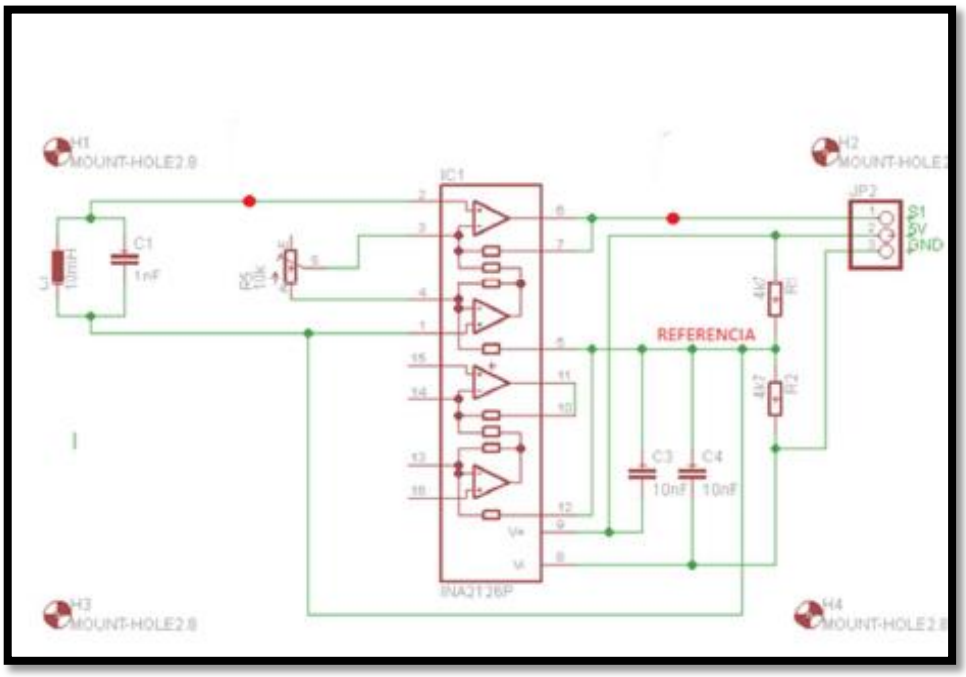


Figura 17. Esquemático del anterior diseño del sensor.

Teniendo esto en cuenta, intentamos no diseñar un circuito usando el amplificador de instrumentación INA126 debido a que además de su alto costo, no era necesario.

EL diseño propuesto está basado en el siguiente principio. La señal recogida en los sensores es una señal producida por el campo modulada por los desplazamientos/orientación de las bobinas sobre el campo. Por este motivo nuestro circuito de detección basa su principio en demodulador de AM, cuya portadora es la señal generada por el campo en el circuito tanque y la moduladora es la señal que se produce de desplazar los sensores sobre el hilo.

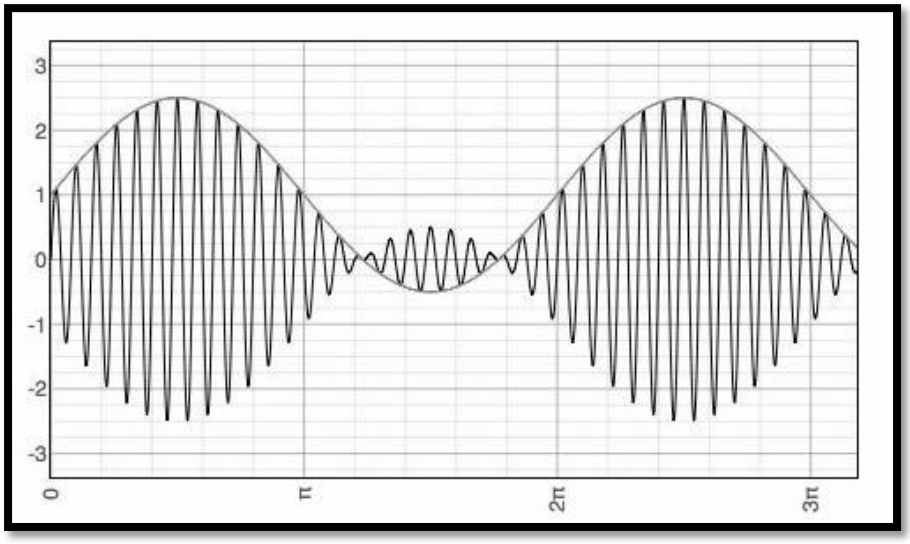


Figura 18. Forma de una onda AM. La amplitud de la onda portadora se modula con la amplitud de la onda moduladora, así su frecuencia no varía pero sí su amplitud.

La amplitud de la señal moduladora es proporcional a la distancia por lo que sólo tendríamos que adaptar la señal para obtener la variación de la moduladora. En la figura 19 se ven las etapas del circuito, primero el circuito tanque con un seguidor de tensión y a continuación un amplificador operacional con ganancia regulable. A la salida del mismo, punto 7, viene un rectificador de media onda junto con una resistencia en paralelo y un seguidor de tensión para aislar la corriente. La resistencia en paralelo es para que circule la corriente ya que en caso contrario el diodo sólo no rectificaría la señal. Por último, tenemos un filtro paso-bajo y un amplificador de nuevo para regular la tensión de salida.

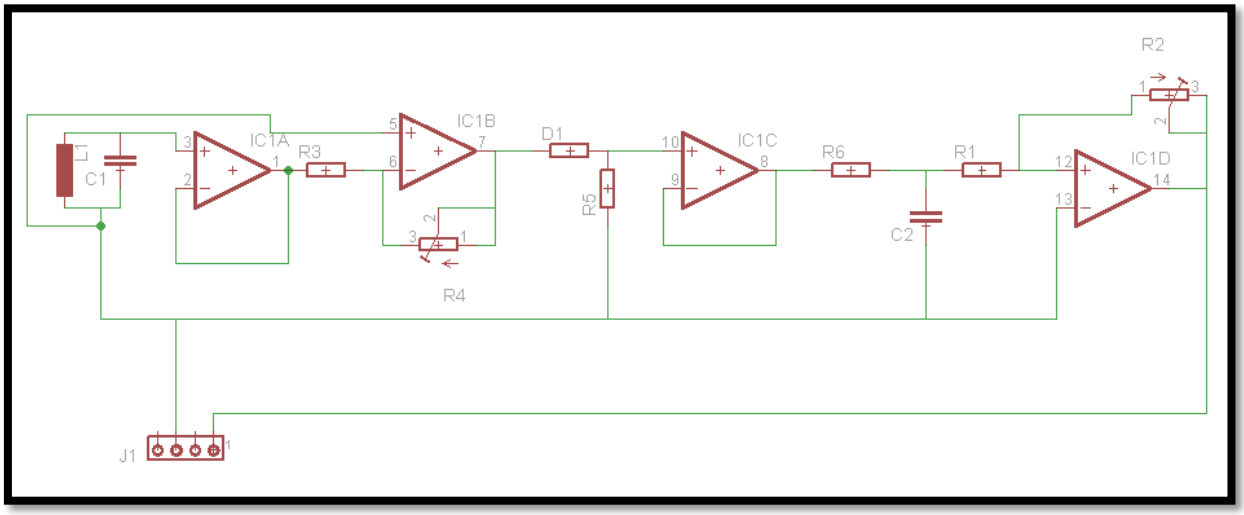


Figura 19. Esquemático del nuevo sensor

Una vez tenemos el circuito diseñado, queda dimensionar todos los componentes. Primero dimensionamos el circuito tanque:

Para una frecuencia de resonancia de 50 KHz y teniendo unas bobinas de tecnología SMD de 10 mH.

$$f = \frac{1}{2\pi\sqrt[2]{CL}} = \frac{1}{2\pi\sqrt[2]{C * 0.001}} \qquad Ec.1.1.4.1$$

Donde C= 1nF.

Tras aislar esta parte con un seguidor, pasamos al amplificador. Se puede ver una resistencia y un potenciómetro. De esta forma, podemos ajustar la ganancia. Los valores son los siguientes:

$$\frac{V_1 - 0}{R_3} = \frac{0 - V_7}{R_4} \qquad Ec.1.1.4.2$$

$$\frac{V_1}{R_3} = \frac{-V_7}{R_4} \qquad Ec.1.1.4.3$$

$$V_7 = -V_1\left(\frac{R_4}{R_3}\right) \qquad Ec.1.1.4.4$$

$$R_4 = potenciómetro, = 20K\Omega$$

$$R_3 = 1K\Omega$$

Esto nos da hasta una ganancia de 20, lo que es suficiente para nuestra aplicación. A continuación viene el diodo que nos rectifica la señal a media onda, simplemente hay que tener en cuenta que en el diodo caen 0.6 ~0.7 V. la resistencia a tierra se pone para que circule corriente sino, debido a la alta impedancia de entrada del operacional no obtendríamos señal. Seguidamente lo aislamos con un seguidor de tensión.

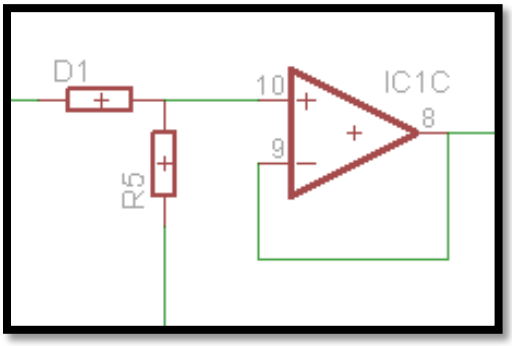


Figura 20. Rectificador de media onda

Para el cálculo del demodulador AM que consiste en un diodo y un filtro pasa-baja, sin embargo, en este caso, vamos a aislar el diodo del filtro pasa baja con un seguidor de tensión poniendo el diodo y una resistencia, R5, a tierra de 5.5K Ω. Siendo el diodo un 1N4148 o similar. Hay que tener en cuenta que el diodo nos rectifica a media onda y caen 0.6V.

Posteriormente, tras aislar el rectificador del filtro con el seguidor de tensión, hallamos el filtro pasa bajas.

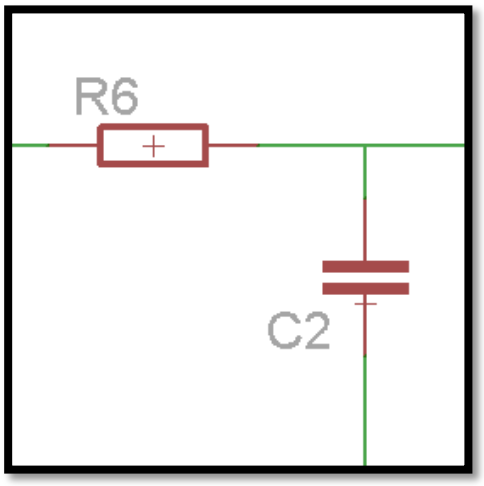


Figura 21. Filtro paso-bajo.

Para hallar los valores del filtro, primeramente deberíamos conocer la frecuencia de corte. Por consiguiente, para conseguir la frecuencia de corte es necesario saber la ganancia del sistema. En nuestro caso, hemos hecho alguna prueba en protoboard. En las pruebas hemos simulado las distancias que tiene el soporte de los sensores (se puede ver en la figura 4) y la entrada que hemos obtenido es de 100 mV. Nuestro circuito, tendrá una ganancia que depende de varios factores. En primer lugar, de la ganancia del generador de campo y luego, en segundo lugar de la regulación de la primera etapa de ganancia del circuito y de la última etapa del diseño donde se ajusta a 3.3 V. por lo tanto, nuestra ganancia es ajustable. Debido a

esto, para el cálculo del filtro no podemos calcular con exactitud la ganancia pero sí podemos poner una aproximación de 30.

La frecuencia de corte es aquella a la cual la ganancia del circuito toma el valor 0.707 del valor máximo, es decir, cuando la ganancia se ha reducido al 70% o lo que es lo mismo, a -3dB, y la tensión de salida tiene una amplitud de 0.707 veces la señal de entrada. Por lo tanto, la ganancia a la frecuencia de corte es:

$$G_c = \frac{1}{\sqrt[2]{2}} * \ G_{V_{max}} = 0.707 * \ G_{V_{max}} \qquad Ec.\ 1.1.4.5$$

Como sabemos, la ganacia es 1 a 0Hz, por lo tanto.

$$G_c = \frac{1}{\sqrt[2]{2}} = \frac{1}{\sqrt[2]{1 + (w_c RC)^2}} \qquad Ec.\ 1.1.4.6$$

$$2 = 1 + (w_c RC)^2 \qquad Ec.\ 1.1.4.7$$

$$w_c = \frac{1}{RC} \qquad Ec.\ 1.1.4.8$$

$$f_c = \frac{1}{2\pi RC} \qquad Ec.\ 1.1.4.9$$

Como no tenemos la relación de ganancia con frecuencia y queremos que nos filtre todo menos los 50KHz del hilo, fijamos una frecuencia de corte muy baja, una frecuencia de 5 Hz y un valor de C = 100nF.

$$5Hz = \frac{1}{2\pi R 100nF} \qquad Ec.\ 1.1.4.10$$

La resistencia del filtro, finalmente queda R= 3.2KΩ.

Finalmente, hay un amplificador en configuración no inversora que nos da la opción de atenuar o amplificar para adaptar la señal a aproximadamente 3.3V. Esto lo conseguimos poniendo una resistencia fija de 10KΩ y un potenciómetro variable de 20KΩ.

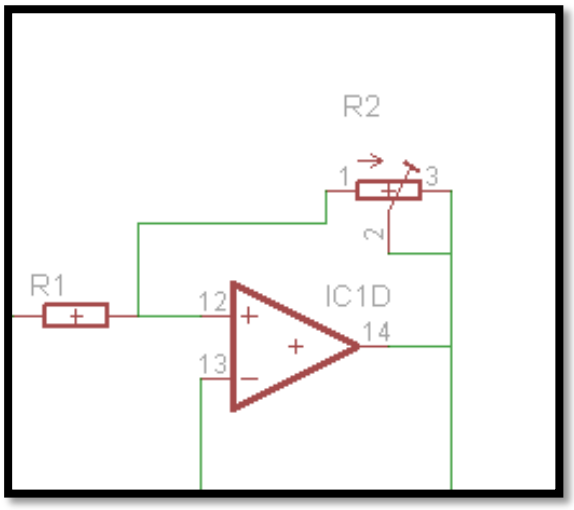


Figura 22. Amplificador operacional para ajustar a 3.3V.

Una vez tenemos el diseño correcto, lo montamos en una placa de pruebas, protoboard, para ver si funcionamiento antes de proceder a imprimir la placa. El resultado no ha sido completamente fiable debido a que el sistema es muy sensible a las variaciones de orientación de las bobinas y los elementos que puedan producir ruido del entorno pero en general las pruebas han sido buenas como se muestra en la figura 23 donde se ve que estando el cable a la misma distancia de las bobinas la tensión que producían era la misma. Posteriormente, al mover el hilo conductor para variar la distancia, variaban de forma inversa las tensiones producidas como se ve en la figura 24.

Figura 23. Señal resultante de las 2 bobinas centradas.

Figura 24. Señal resultante de variar la distancia de las bobinas al hilo conductor.

2.3.5 Diseño PCB

Una vez tenemos el diseño realizado en Eagle, pasamos del diseño esquemático, el anterior, al *board*. En este paso realizamos el diseño de las pistas, vías y organización de los componentes en la placa, proceso place and route.

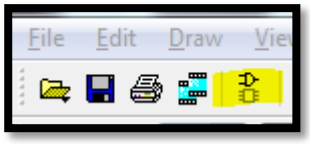


Figura 25. Menú herramienta Eagle.

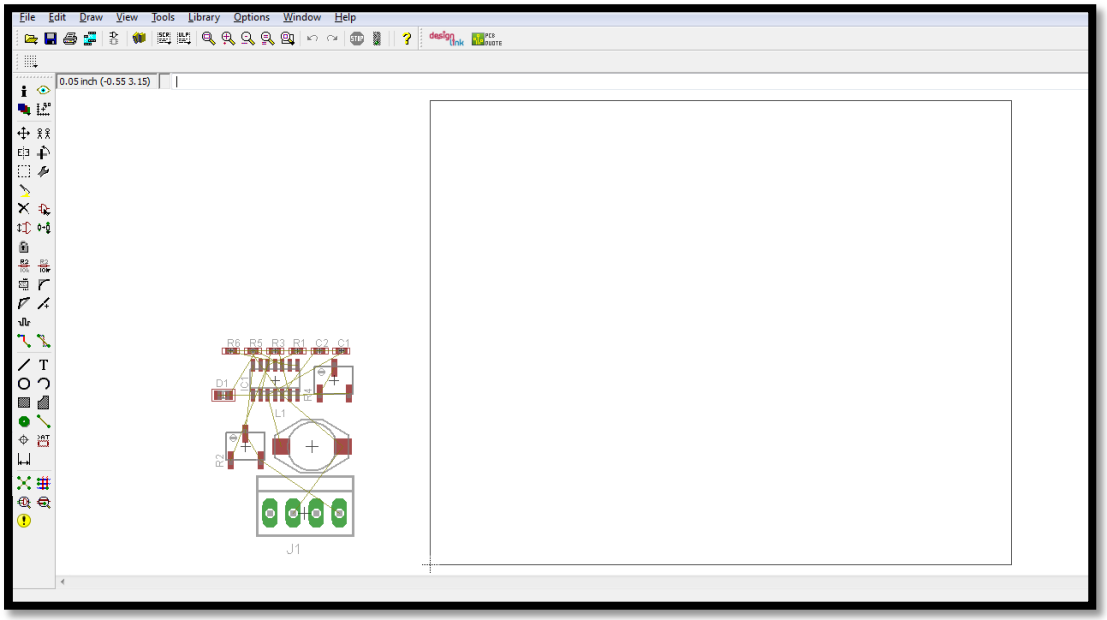


Figura 26. Diseño PCB

En este paso, dimensionamos el tamaño de la placa.

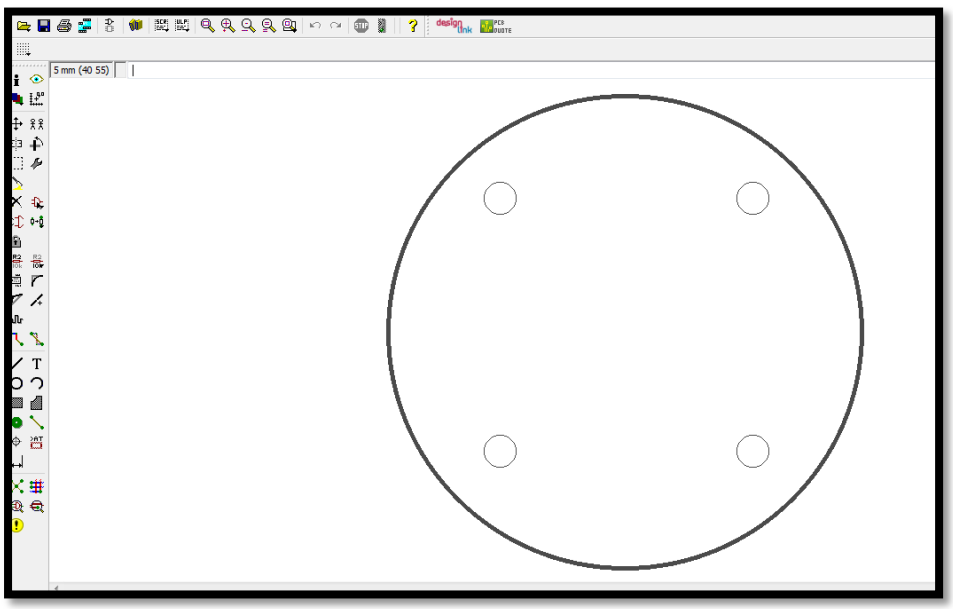


Figura 27. Diseño PCB, creando el borde de la placa

Organizamos los componentes y ruteamos.

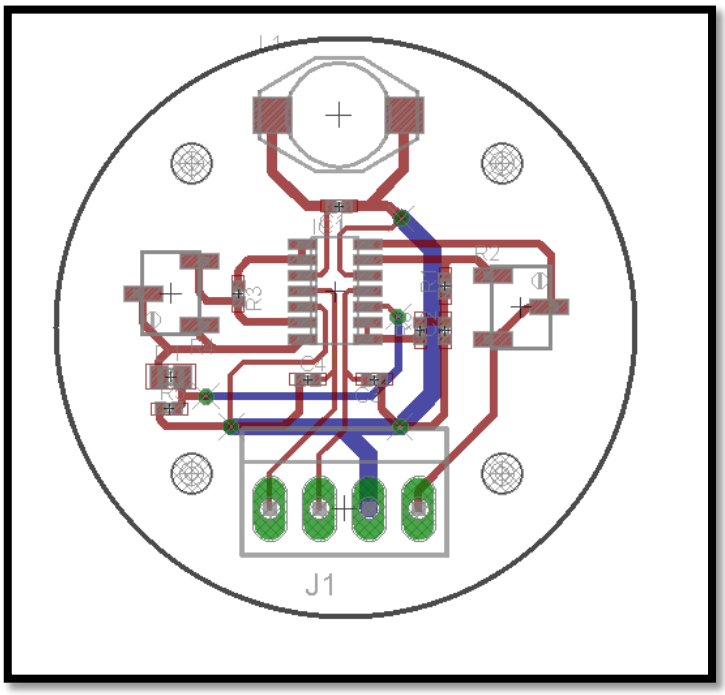


Figura 28. Diseño de la placa del sensor

El programa permite obtener varias vistas del diseño como pistas, pads, componentes, etc.

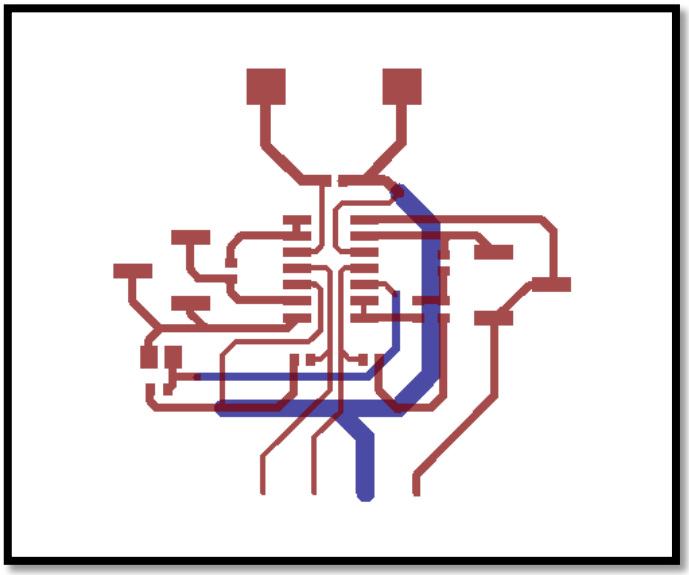


Figura 29. Vista de las pistas del sensor.

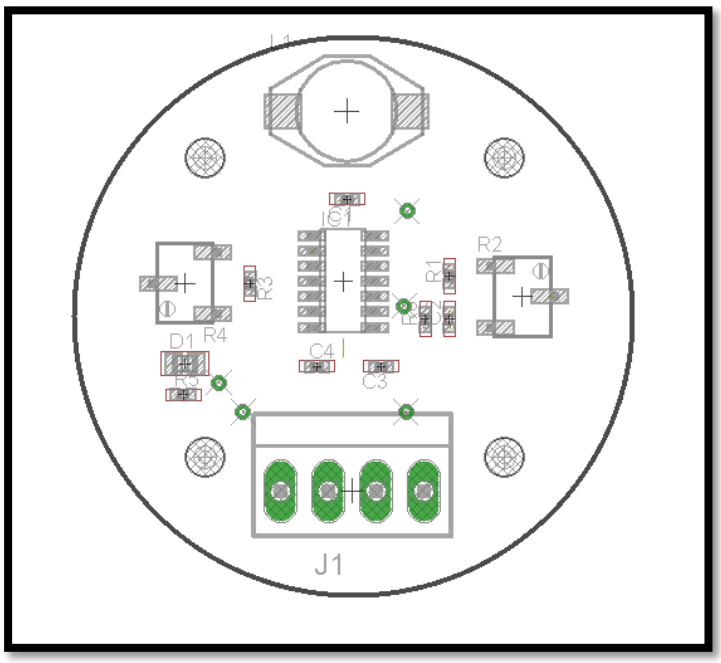


Figura 30. Vista de la organización de los componentes.

Una vez tenemos todo en su sitio, ruteado y comprobado, pasamos a la fase de generar los ficheros *gerber* para la impresión.

Para ello, vamos a *File -> run* y abrimos el archivo *drillcfg.ulp*. Luego vamos a *CAM*. Donde podremos crear los ficheros.

Para ello entramos en la herramienta.

Figura 31. Menú CAM de la herramienta Eagle.

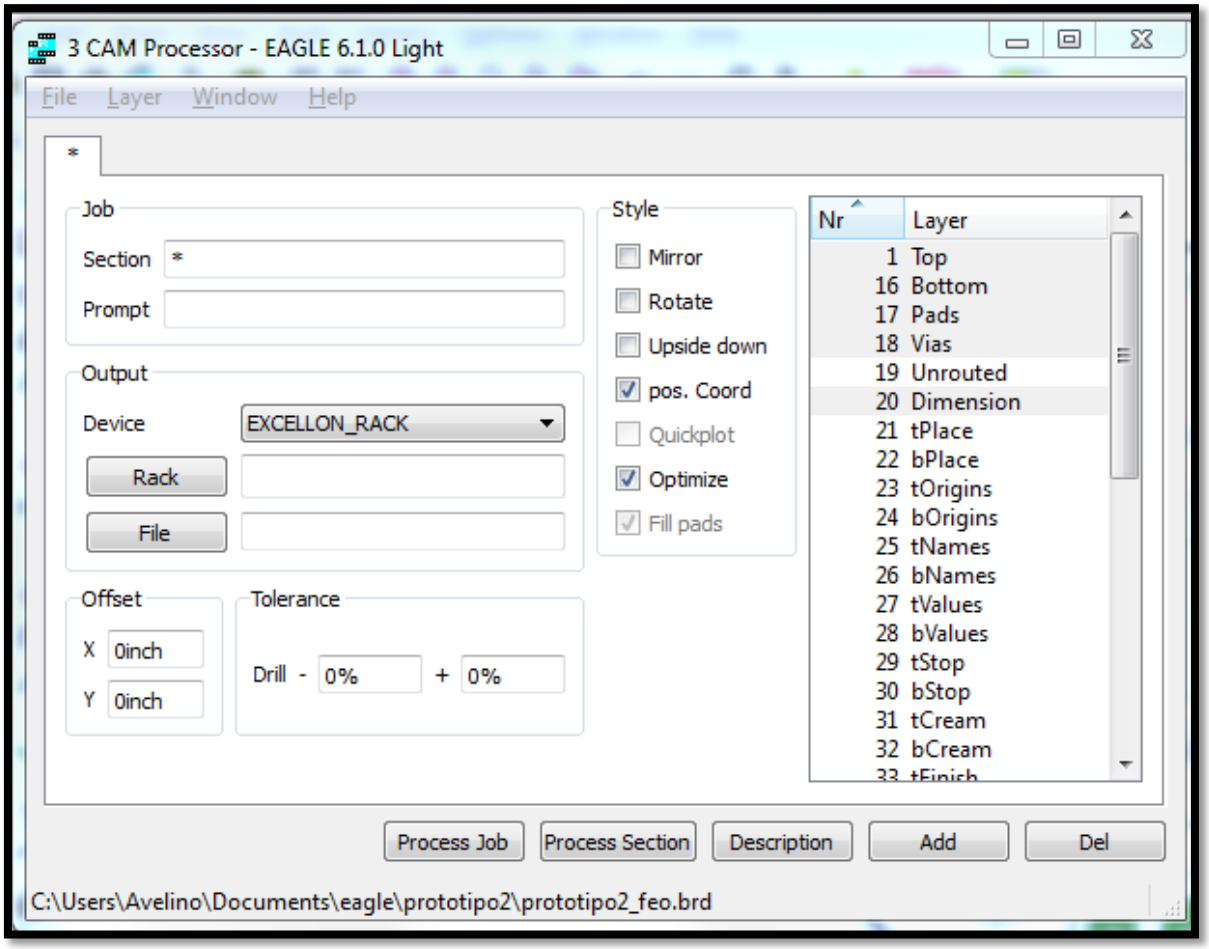


Figura 32. Herramienta CAM de Eagle.

Una vez aquí, vamos a *file -> open -> job*. En *Device* seleccionamos *EXCELLON_RACK*, en *Rack* ponemos la dirección de nuestro archivo y en *file* le cambiaremos el nombre al archivo como *_drill.gbr*, seleccionamos *EXCELLON* en *Device* y le damos a *ProcessJob*.

A continuación, abrimos de nuevo la herramienta, y seleccionamos en *Device* la opción *Gerber_RS274X* y luego, dependiendo de la capa que queramos, seleccionamos *Top*, *Pad* y *vías* o *Bottom*, *Pad* y *Vías* o *Dimension*.

Con estos pasos, ya tenemos creados los archivos *gerber* para imprimir nuestra placa.

En la figura 33 se puede apreciar el sensor terminado con sus 4 terminales por donde alimentaremos y recibiremos la señal del sensor.

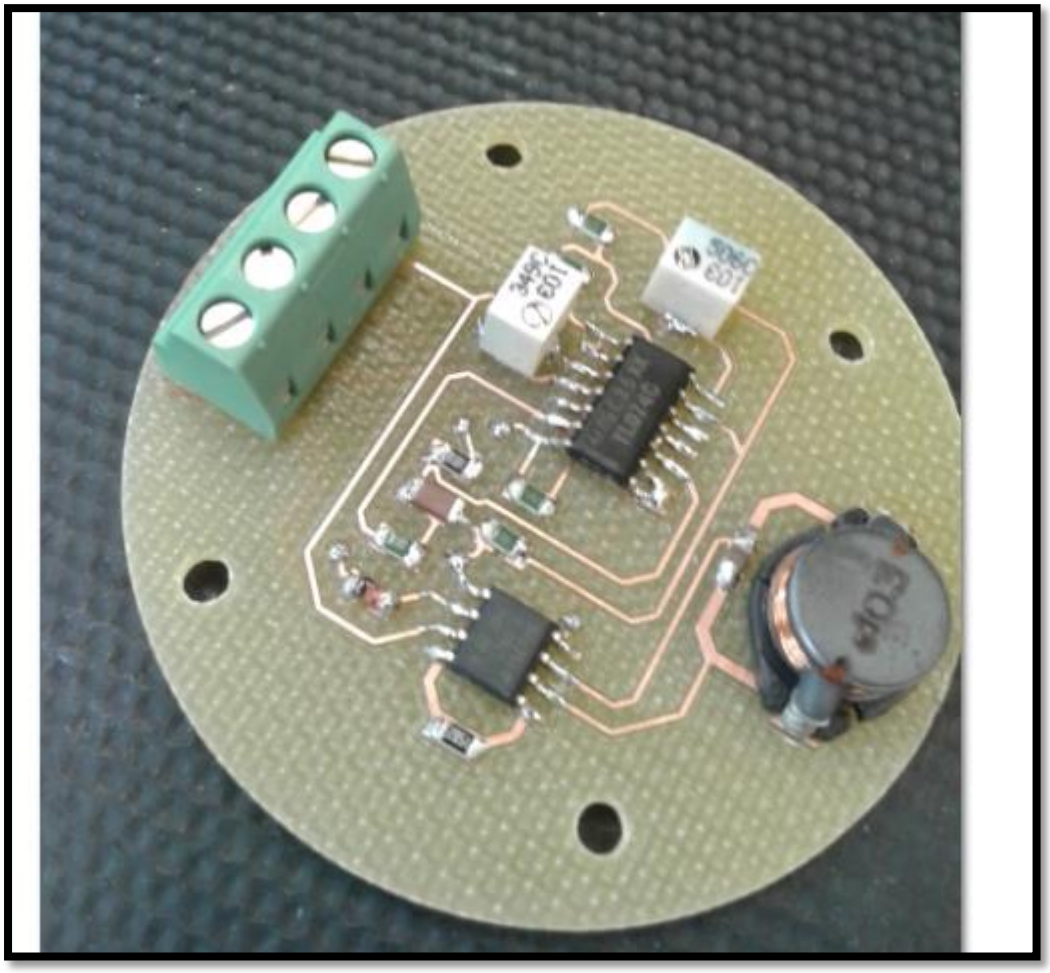


Figura 33. Imagen de la placa PCB terminada del nuevo sensor.

2.3.6 Procesado de la señal de los sensores

Una vez tenemos la señal de continua proporcional a la distancia de los sensores al cable guía, queda calibrar el sistema teniendo el carro centrado y saber la proporción de nivel de continua y distancia. Luego, la señal es recogida por la Mbed por uno de sus puertos convertidores analógicos – digitales y enviada vía serial a la Beaglebone donde se calcula la distancia, el error del vehículo con respecto a su trayectoria se envía al nodo ROS a la Beaglebone y esta envía la información a la otra mbed, la de los motores mediante un nodo ROS.

2.4 Descripción del sistema software

En este apartado hablaremos tanto de las plataformas y los sistemas que conformas la arquitectura propuesta basada en ROS.

Figura 34. Imagen de ROS. Logotipo.

2.4.1 Sistemas operativos.

En el proyecto se instalan varios sistemas operativos en las distintas plataformas que conforman el sistema. A continuación se detalla cada sistema operativo, su razón de instalación y sus principales características.

2.4.2 Sistema operativo en el PC. Ubuntu.

Ubuntu es sistema operativo basado en Linux que se distribuye de forma libre. Es un sistema de código abierto. Esto significa que la comunidad colabora en mejorar el sistema y difundir las mejoras.

La razón principal de la elección de este sistema es ROS ya que es una plataforma en la que está actualmente funcionando de forma estable.

La elección de este sistema operativo también se ve influenciada por ser de código abierto y son sistemas operativos gratuitos. Otra de las razones es que los equipos que montan Ubuntu no necesitan tanta potencia como otros con Windows aumentando así su rendimiento. Actualmente casi cualquier dispositivo puede montar un sistema Linux ya que, como se comentaba, no necesitan grandes procesadores.

Su instalación se puede ver en el Anexo 1 donde se explica detalladamente los pasos a seguir para tener instalado correctamente el sistema.

2.4.2.1 Sistema operativo en Raspberry pi. Raspbian.

Raspbian es un sistema operativo resultado de la necesidad de tener un sistema óptimo para Raspberry. Está basado en Debian Wheezy el cual es una distribución del sistema operativo Linux y por tanto, libre.

Raspbian es una optimización de Debian para Raspberry pi, de ahí su nombre. Es un sistema que porta los programas básicos y utilidades que hacen que la Raspberry pi funcione. Sin embargo, no viene sólo con lo esencial, sino que viene con más de 35.000 paquetes precompilados y de fácil instalación en la Raspberry pi.

La razón de esta elección, ROS. Es un sistema operativo que soporta bien la plataforma ROS por lo que es el ideal para nuestra aplicación.

Su instalación puede verse en el Anexo 4 donde se detallan los pasos para su puesta en marcha.

2.4.2.2 Sistema operativo en Beaglebone. Ubuntu

En la Beaglebone, a lo largo del desarrollo del proyecto se han instalado dos sistemas operativos distintos. Uno de ellos es Ubuntu y el otro es Angstrom.

Inicialmente, la Beaglebone viene con Angstrom instalado. Cuando tomamos por primera vez la Beaglebone en los comienzos del proyecto y nos conectamos a la misma vimos que la versión de Angstrom estaba desactualizada. Tras varios intentos fallidos de intentar actualizar el S.O. en parte por la inoperatividad de su página oficial, nos decidimos a instalar Ubuntu, más concretamente Ubuntu 12.10.

Los cambios de Ubuntu a Angstrom y viceversa han sido varios desde que empezamos pero finalmente nos quedamos con Ubuntu debido a que es más estable, tiene más soporte y, lo principal, ROS funciona mejor en Ubuntu que en Angstrom y es más fácil programar la Beaglebone en Ubuntu que en Angstrom.

La instalación de Ubuntu en Beaglebone paso a paso se puede encontrar en el Anexo 2, así como su configuración general. Además, como instalamos Angstrom, la instalación está detallada en el Anexo 3.

2.4.3 ROS

2.4.3.1 Introducción a ROS

ROS (Robotic Operating System) proporciona un entorno de programación distribuido de código abierto (open source) para el control de robots, tanto físicamente como en simulación, que aporta abstracción hardware, control de bajo nivel, comunicación entre procesos y gestión de paquetes software.

El matiz de *open source* es fundamental. Existen actualmente infinidad de repositorios de libre acceso con soluciones ROS de todo tipo (SLAM, reconocimiento 3D, planificación de movimiento o *machine learning).* Esta disponibilidad de soluciones a problemas cotidianos en el campo de la robótica ahorra tiempo y esfuerzo y permite al diseñador centrarse en su área de interés sin ocuparse de solucionar problemas ya resueltos por otros.

El objetivo principal de ROS es compartir y colaborar. Además persigue la facilidad de integración con otras plataformas, independencia respecto al lenguaje de programación, facilidad de realización de pruebas y escalabilidad.

En la actualidad ROS funciona sobre plataformas tipo UNIX, está mayoritariamente probado en sistemas Linux Y Mac OS X. Regularmente se publican distribuciones ROS, consistentes en el núcleo del sistema ROS y un conjunto de utilidades compatibles.

2.4.3.2 Nomenclatura y conceptos básicos de ROS

Repositorio: son ficheros compuestos por uno o más paquetes y pilas, y que nos permiten descargarnos los ficheros necesarios de forma más cómoda.

Pila (Stack): es una colección de paquetes que comparten una misma funcionalidad, sería el equivalente a una librería en otros sistemas de programación, y dentro de estos podemos encontrar unos ficheros con metadatos que nos proporcionan la información para que estos funcionen de forma correcta (dependencias, compilación...).

Paquete: es la unidad principal de organización en ROS, contiene todo lo necesario para hacer ser funcional y es análogo a un paquete en C.

Nodo: un nodo es un programa, que realiza una función ya sea publicar un mensaje, suscribirse a él o hacer cualquier procesamiento. La estructura de un nodo es la misma, estructura que cualquier programa elaborado en Python y C++, salvo que necesitan de una

serie de librerías o módulos para actuar como nodo dentro de ROS. En la figura 35 podemos ver un nodo ejecutándose con sus tópicos correspondientes.

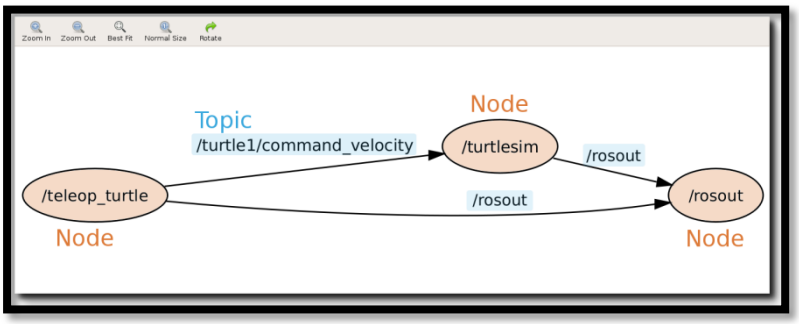


Figura 35. Nodos en ROS

Servicio: es el tipo de arquitectura encargada de realizar la comunicación entre nodos, utilizan dos tipos de mensajes, uno para la solicitud, y otro que es la respuesta dada por el otro nodo a esa petición.

En los programas podremos encontrar nodos servidor y nodos clientes. Tras realizar la solicitud, el nodo servidor se queda en modo espera hasta recibir respuesta por parte del nodo cliente, y en el caso de que sea el cliente el que realiza una petición, el nodo servidor la procesará y responderá al cliente con la información requerida. Esta comunicación se ve en la figura 36, donde vemos que la comunicación es bidireccional.

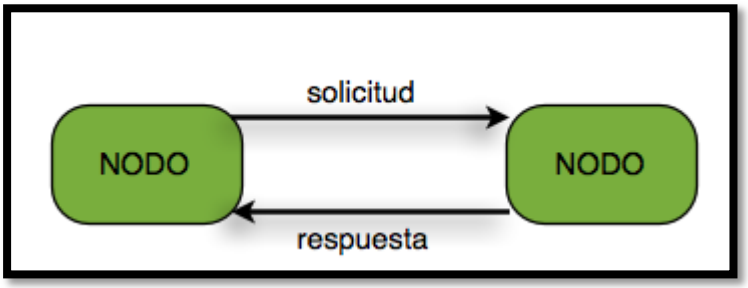


Figura 36. Comunicación de nodos en ROS.

Tópicos: tópicos o temas, son los nombres que identifican el contenido de un mensaje; y estos se enrutan de dos formas, una publicador y otra suscriptor.

Un nodo que está interesado en un determinado tipo de datos se suscribe al tema correspondiente.

Puede haber varios editores y suscriptores concurrentes a un mismo tema, y un único nodo puede publicar y / o suscribirse a múltiples temas. En general, los editores y suscriptores no son conscientes de la existencia de los demás.

Se puede pensar en un tema como un Bus de mensajes. Cada Bus tiene un nombre, y cualquier persona puede conectarse al bus para enviar o recibir mensajes, siempre y cuando sean del tipo correcto.

Mensajes: los nodos se comunican entre sí pasando mensajes. Un mensaje es simplemente una estructura de datos, que comprende los tipos de campos. Los mensajes pueden incluir estructuras arbitrariamente anidadas y matrices (al igual que las estructuras de C).

Maestro: el Maestro proporciona registro de nombres y la búsqueda para el resto de los nodos. Sin el Maestro, estos no serían capaces de encontrar mensajes entre sí, intercambiar, o invocar los servicios, lo que hace que sea totalmente indispensable a la hora de ejecutar cualquier tipo de programa.

Bags: las bolsas son un formato para guardar y reproducir datos de un mensaje de ROS, permitiéndonos almacenar una serie de órdenes y después repetirlas secuencialmente. Las bolsas son un mecanismo importante para el almacenamiento de datos, tales como datos de un sensor, que puede ser difícil de recoger, pero es necesaria para desarrollar y probar algoritmos.

2.4.3.3 Arquitectura de nuestro sistema basado en ROS

Nuestra arquitectura se basa en 1 maestro y 2 esclavos como se ve en la figura 37. Además, uno de los esclavos tiene 2 Mbed asignadas como esclavo para la lectura de los sensores y escritura en los motores.

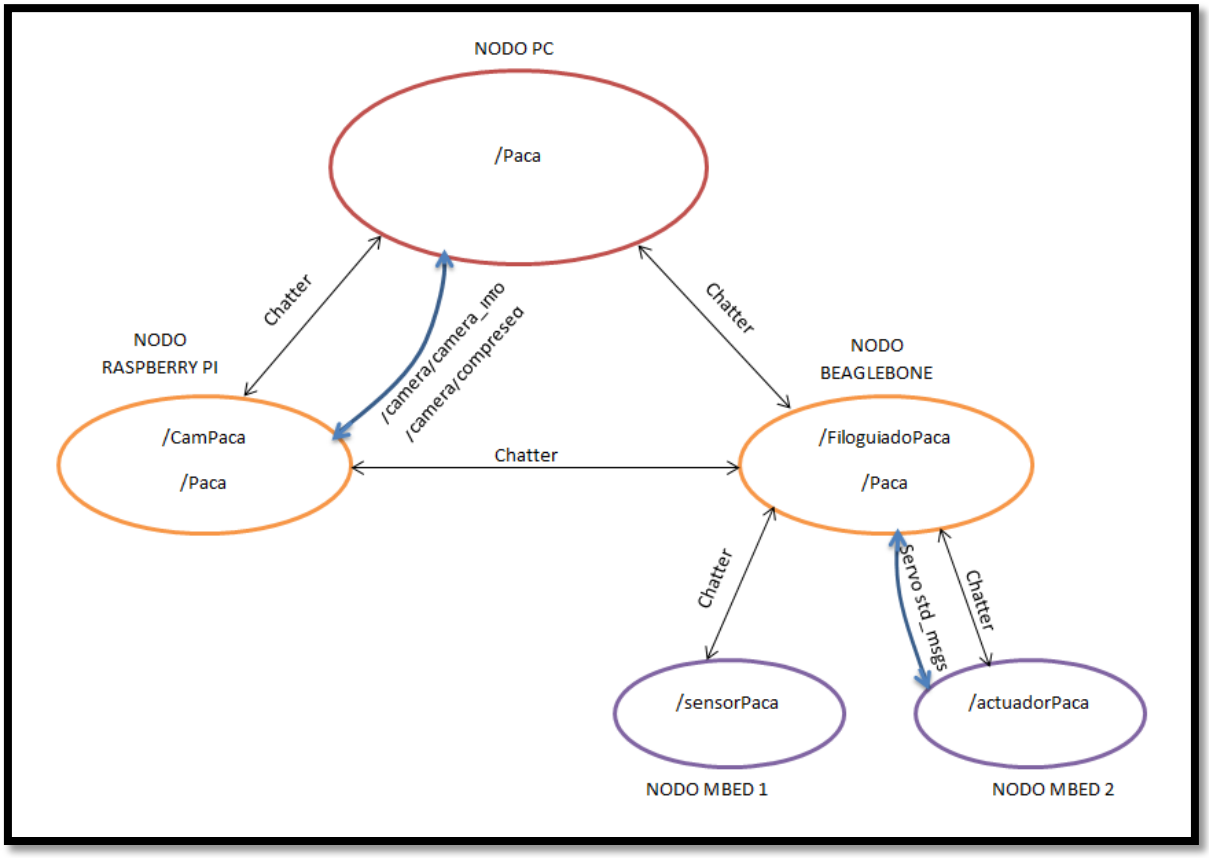


Figura 37. Mapa de nodos.

En nuestro vehículo, ROS debe ser capaz de comprender docenas, incluso cientos de nodos, que se distribuyen a través de nuestro sistema.

Debido a la configuración de nuestro sistema, nos interesa que los nodos seriales de la Beaglebone hacia las Mbed puedan enviar los datos obtenidos mediante otro nodo de red hacia el PC, para que desde este se pueda monitorizar al vehículo en sí al igual que con el nodo de cámara, para poder seguir vía remota el trabajo del vehículo.

Para llevar a cabo esta tarea, tenemos que configurar la red ROS para que exista conectividad completa y bi-direccional entre el PC, la Beaglebone y la Raspberry pi. Como esta comunicación se realiza vía internet, se debe dar una dirección fija a todos los sistemas. Toda esta configuración se explica paso a paso en el Anexo 11.

Nuestros nodos son un nodo de cámara en la Raspberry pi, un nodo de comunicación de red entre el maestro y los dos esclavos y un nodo serial entre la Beaglebone y las dos Mbed. Esto se puede apreciar gráficamente en la figura 37 donde se ven los nodos y sus tópicos. Los nodos se pueden denominar por el nombre deseado a la hora de lanzarlos. Cada nodo tiene una serie de tópicos a los cuales nos podemos subscribir o publicar en los mismos.

rosrun raspicam raspicam_node_name:=CamPaca

Topic:

/camera/compressed

/camera/camera_info

rosrun rosserial_python serial_node.py_name:=Paca tcp

Topic:

/chatter

rosrun rosserial_python serial_node.py_name:=FiloguiadoPaca _port:=/dev/ttyACM0

Topic:

/servo std_msgs

Principalmente hay que tener una instalación previa de un sistema operativo compatible con ROS. Y posteriormente proceder a su instalación ayudándonos de la propia wiki de ROS, en la cual se explica su instalación.

En el sistema hemos tenido que instalar ROS en el PC, en la Beaglebone y en la Raspberry pi:

Instalación de ROS en ROS en PC con Ubuntu: Anexo 5.

Instalación de ROS en ROS en Beaglebone con Ubuntu: Anexo 6.

Instalación de ROS en ROS en Beaglebone con Angstrom: Anexo 7.

Instalación de ROS en Raspberry pi con Raspbian: Anexo 8.

En el Anexo 9 y Anexo 10 se detallan algunos conceptos básicos de ROS y los nodos.

3. Resultados y conclusiones

3.1 Resultados y conclusiones

A lo largo del desarrollo de este trabajo fin de grado se ha desarrollado un sistema basado en ROS para un AGV, "*Paca*". Además se estudió la parte del sensor filoguiado haciendo un nuevo diseño del sensor.

Dicha configuración de ROS se compone de un maestro y dos esclavos situados en el vehículo que sirven para la comunicación y para llevar a cabo el control de "*Paca*".

La forma en la que se enfocó el trabajo desde un principio nos llevó a tener que buscar información y aprender sobre microprocesadores, sistemas operativos, ROS y aplicar conocimientos adquiridos en el grado para la parte del sensor filoguiado.

Finalmente queda desarrollar el sistema en conjunto debido a que por falta de tiempo no se ha podido implementar todo en conjunto.

3.2 Results and conclusions.

Along the life of this final work degree, we have developed a ROS system for an AGV, "*Paca".* In addition, we study the magnetic sensor making a new design.

This ROS configuration is composed by a master and two slaves. The slaves are in the vehicle and they work for "*Paca"* communication and control.

The principal aim made us to look for information and learn about microprocessors, Operating systems, ROS and apply knowledge that we have of the degree for apply them in the magnetic sensor.

Finally, it is not finished the develop of the all microprocessors, sensors and other components together because we had not time.

4. Mejoras y trabajos futuros

Integración de todo el sistema en *"Paca* ". En cuanto a mejoras, queda pendiente modificar los códigos de la Mbed para la transferencia de datos vía ROS.

También está pendiente el problema a la detección de los cruces en el circuito. Para ello se debe implementar un sistema RFID tanto a nivel de hardware en PACA y en el vivero como a nivel software en la Mbed que recibe la información.

Otra mejora sería instalar los sensores de ultrasonido para tener un sistema de seguridad en el vehículo debido a que actualmente no hay ningún sistema de este tipo.

En un futuro, tras analizar las prestaciones del sistema, se puede valorar el quitar una de las Mbed y dejar la otra para las dos funciones de leer los sensores y actuar sobre los motores.

Esto es posible siempre que se compruebe si con una sola Mbed hay potencia de cálculo suficiente para poder hacer estas dos acciones.

5. Referencias

[1] Ubuntu. Lector tarjetas SD:

[1.1]http://www.armhf.com/index.php/getting-started-with-ubuntu-img-file/

[1.2]http://linuxenandalu.blogspot.com.es/2013/04/detectar-tarjetas-sd-en-linux.html

[1.3]http://modogeek.blogspot.com.es/2013/01/ubuntu-1204-problema-con-el-lector-de.html

[1.4]http://www.ubuntu-guia.com/2010/08/ubuntu-no-lee-tarjeta-memoria.html

[1.5]http://mundo-linux-b.blogspot.com.es/2013/05/problemas-con-lector-de-tarjetas-sd-en.html

[2]Borrar una carpeta desde consola con permisos de súper usuario:

[2.1]http://blog.vortexbird.com/2012/05/02/borrar-carpeta-en-linux-desde-la-consola/#.U0LcZPl5OSp

[3]Beaglebone. Configuración de red para Angstrom:

[3.1]https://github.com/rlrosa/uquad/wiki/BeagleNet

[4]Código en C++ de puerto serial:

[4.1]http://www.lawebdelprogramador.com/codigo/C_Visual_C/212-Comunicacion_de_dos_PCs_por_puerto_serial_en_C++.html

[5] Linux. Configuración de kernel 3.8:

[5.1]http://elinux.org/BeagleBone_and_the_3.8_Kernel

[5.2]http://elinux.org/EBC_Exercise_08_Installing_Development_Tools

[6] Herramientas. Mux Pin y re compilación de un kernel:

[6.1]http://processors.wiki.ti.com/index.php/Device:AM335x:Device_Evaluation

[6.2]http://processors.wiki.ti.com/index.php/Pin_Mux_Utility_for_ARM_MPU_Processors

[6.3]http://www.ti.com/lit/ds/sprs717f/sprs717f.pdf

[6.4]http://www.ti.com/tool/pinmuxtool

[7]Código de ejemplo de programación en C++ para la Beaglebone donde se usa el puerto I2C:

[7.1]http://derekmolloy.ie/beaglebone/beaglebone-an-i2c-tutorial-interfacing-to-a-bma180-accelerometer/

[8] ROS. Comandos:

[8.1]http://wiki.ros.org/ROS/CommandLineTools

[9] ROS. SistemaMaestro-Esclavo:

[9.1]http://www.instructables.com/id/Getting-Started-with-ROS-Robotic-Operating-Syste/?ALLSTEPS

[9.2]http://nootrix.com/2012/06/ros-networking/

[9.3]http://wiki.ros.org/ROS/NetworkSetup

[10] Beaglebone. Configuración de red con Ubuntu:

[10.1]https://wiki.ubuntu.com/ARM/BuildArmPackages

[11] Cambiar contraseña administrador en lo SO de las placas:

[11.1]http://www.taringa.net/posts/linux/2154655/Cambiar-la-contrasena-del-root-en-Ubuntu.html

[12] Beaglebone. Configuración inicial con Angstrom:

[12.1]https://learn.adafruit.com/downloads/pdf/beaglebone.pdf

[13] Dar permisos completamente a una carpeta en Linux (útil para construcciones de kernel con Mux Pin):

[13.1]http://www.ubuntu-es.org/node/8996#.U0Lczfl5OSp

[14] Herramientas. Editor VIM:

[14.1]http://blyx.com/public/docs/curso-linux-principiantes/vi1.html

[14.2]http://rm-rf.es/guardar-un-fichero-dentro-de-vim-cuando-no-tenemos-permisos/

[14.4]http://intervia.com/doc/instalar-y-configurar-vim/

[14.5]http://www.guia-ubuntu.com/index.php?title=VIM

[15] Error común de fallo conexión sistema Maestro-Esclavo:

[15.1]http://wiki.ros.org/rosnode/Troubleshooting

[16] Beaglebone. Forooficial:

[16.1]http://beagleboard.org/Community/Forums

[17] Beaglebone. Imagen Angstrom:

[17.1]http://downloads.angstrom-distribution.org/demo/beaglebone/

[18] Instalar Cloud9 y Bonescript en Beaglebone con ubuntu:

[18.1]http%3A%2F%2Fdotnetdavid.wordpress.com%2F2013%2F09%2F16%2Fbeaglebone-black-cloud9-and-bonescript-install-guide%2F&h=8AQGYzy9X

[19] Herramientas. I2C:

[19.1]http://www.linuxcolombia.com.co/?q=node/30

[19.2]http://www.industriaembebidahoy.com/efsystems/es/node/39

[19.3]http://dev.ardupilot.com/wiki/building-for-beaglebone-black-on-linux/

[19.4]http://www.michaelhleonard.com/understanding-and-using-i2c/

[19.5]http://elinux.org/Interfacing_with_I2C_Devices#Beagleboard_I2C2_Enable

[20] Ubuntu. Instalación de ubuntu-12.10-console-armhf-2013-09-26:

[20.1]http://wind.cs.purdue.edu/doc/beaglebone_ubuntu.html#sec7

[21] Herramientas. Instalación de *screen:*

[21.1]http://www.guia-ubuntu.com/index.php?title=Screen

[22] Herramientas. Configuración IP estática Linux:

[22.1]http://www.utilizalinux.com/2013/06/configurar-ip-en-ubuntu.html

[23] Beaglebone. Instalación de USB WiFi rtl8192cu con S.O. Angstrom:

[23.1]http://www.codealpha.net/864/how-to-set-up-a-rtl8192cu-on-the-beaglebone-black-bbb/

[23.2]http://bonenotes.tumblr.com/

[23.3]https://groups.google.com/forum/#!topic/beaglebone/Q92uD9F1us8

[23.4]http://blog.deancos.com/2014/05/08/beaglebone-configuracion-wifi/

[24] Ubuntu. Instalación de USB WiFi rtl8192cu con S.O. Ubuntu:

[24.1]https://groups.google.com/forum/#!topic/beaglebone/Q92uD9F1us8

[24.2]http://www.internetdelascosas.cl/2012/05/30/conectando-el-beaglebone-via-wifi/

[25] ROS. Instalación para Ubuntu-ARM:

[25.1]http://wiki.ros.org/hydro/Installation/UbuntuARM

[26]Modificando kernel de la Beaglebone para añadir usb wifi

[26.1]http://www.youtube.com/watch?v=HJ9nUqYMjqs

[27] Herramientas. Programar la Beaglebone:

[27.1]http://www.ciudadoscura.com/programacion/13/primeros-pasos-con-beaglebone-parte-2-configuracion-de-gpio.html

[27.2]http://beagleboard.org/Support/bone101/

[28] Raspberry pi. Configuración del teclado:

[28.1]http://wiki.bandaancha.st/C%C3%B3mo_espa%C3%B1olizar_tu_Raspberry_Pi

[29] ROS. Paquete de puerto serial:

[29.1]http://wiki.ros.org/cereal_port

[30] Raspberry pi. Programar el puerto GPIO con QT Creator:

[30.1]http://aquihayapuntes.com/%C3%ADndice-sistemas-embebidos/raspberry-pi-programando-el-puerto-gpio-con-qt-creator.html

[31] ROS. Paquetes para Ubuntu-ARM:

[31.1]http://packages.namniart.com/repos/status/hydro.html

[32] BeagleBone. PINOUT:

[32.1]http://elinux.org/BeagleBone_Community

[33] Beaglebone. Programar desde otro dispositivo.

[33.1]http://visualgdb.com/tutorials/beaglebone/

[34] Ubuntu. Puerto serial en la Mbed y en PC.

[34.1]http://mbed.org/users/karls/notebook/linux-serial-terminal/

[34.2]https://mbed.org/handbook/Serial

[35]Proyectos donde se instala ROS en la Beaglebone:

[35.1]http://sysadminfixes.wordpress.com/2013/03/18/ros-on-beaglebuntu/

[35.2]http://eigendreams.wordpress.com/2013/08/10/installing-ros-and-opencv-on-ubuntu-arm-12-10-with-lxde-on-a-beagleboneblack-and-configuring-ssh-and-tightvnc/

[35.3]http://beagleros.wordpress.com/

[35.4]http://www.eps.uam.es/nueva_web/intranet/ga/tfdm/trabajos/Julio_Francisco_Acosta_Nunnez.pdf

[36] Beaglebone. Instalación de ROS:

[36.1]http://wiki.ros.org/BeagleBone

[37] Raspberry pi. Instalación de ROS:

[37.1]http://wiki.ros.org/groovy/Installation/Raspbian

[38] ROS. Rosserial:

[38.1]https://github.com/ros-drivers/rosserial

[38.2]http://mbed.org/users/nucho/code/rosserial_mbed/

[38.3]http://mbed.org/questions/200/Rosserial-to-mbed-with-Xbee/

[38.4]http://answers.ros.org/question/53297/rosserial-for-mbed-on-ros-groovy-solved/

[39] RFID:

[39.1]http://wiki.ros.org/hrl_rfid

[40] Ubuntu. Server SSH:

[40.1]http://ubuntulife.wordpress.com/2007/03/15/instalar-un-servidor-ssh/

[40.2]http://embeddedprogrammer.blogspot.com.es/2012/10/beaglebone-installing-ubuntu-1210.html

[41] SPI:

[41.1]http://hipstercircuits.com/enable-spi-with-device-tree-on-beaglebone-black-copy-paste/

[41.2]http://hipstercircuits.com/enable-spi-1-0-and-1-1-with-device-tre-overlays-on-beaglebone/

[41.3]http://yellowfeather.co.uk/blog/2012/03/26/userland-spi-on-the-beaglebone-with-ubuntu/

[41.4]http://www.brianhensley.net/2012/02/spi-working-on-beagleboard-xm-rev-c.html

[41.5]https://www.linux.com/learn/tutorials/746860-how-to-access-chips-over-the-spi-on-beaglebone-black

[41.6]http://theredblacktree.wordpress.com/2012/10/01/multiple-spi-bus-beaglebone/

[41.7]http://mbed.org/handbook/SPI

[42] ROS. Tutoriales:

[42.1]http://jbohren.com/tutorials/

[42.2]http://tdrobotica.co/tutoriales/robotica

[42.3]http://tdrobotica.co/index.php?option=com_content&view=article&id=3

[42.4]https://wiki.citius.usc.es/inv:por-clasificar:ros66

[43] Herramientas. Explicación MUX PIN en Beaglebone:

[43.1]https://www.youtube.com/watch?v=iXa75qWg9Xo

[44] Herramientas. Explicación configuración pines GPIO manualmente en Beaglebone.

[44.1]https://www.youtube.com/watch?v=iXa75qWg9Xo

[45] Herramientas. Web de Robert C. Nelson:

[45.1]https://github.com/RobertCNelson

[46] ROS. Página oficial:

[47.1]http://wiki.ros.org/

[48] Herramientas. Web para compartir códigos:

[48.1]http://pastebin.com/

[48.2]https://github.com/

[49] Beaglebone. Wiki sobre configuración de Angstrom.

[49.1]http://www.gru.ucr.ac.cr/GRU/mediawiki-1.20.2/index.php/BeagleBone

[50] Webs útiles a la hora de empezar a usar la Beaglebone:

[50.1]http://beagleboard.org/Getting%20Started

[50.2]http://tecbolivia.com/index.php/venta-de-componentes-electronicos-11/tableros-de-experimentacion/placa-beaglebone-detail

[50.3]http://tecbolivia.com/index.php/articulos-y-tutoriales-microcontroladores/61-tutorial-de-beaglebone

[50.4]http://www.armhf.com/index.php/boards/beaglebone-black/

[50.5]http://www.internetdelascosas.cl/2012/05/28/usando-ubuntu-en-beaglebone/

[50.6]http://www.congdegnu.es/2012/01/28/beaglebone-primeros-pasos/

[50.7]http://www.lvr.com/beaglebone.htm

[51] Web muy interesante sobre el uso de ROS, incluye códigos en C++:

[51.1]http://robotica.unileon.es/mediawiki/index.php/Fernando-TFM-ROS02

[52] Raspberry pi. Configuración USB WiFi:

[52.1]http://www.raspyfi.com/wi-fi-on-raspberry-pi-a-simple-guide/

[53] Raspberry pi. Imágenes S.O:

[53.1]http://www.raspberrypi.org/downloads/

[54] Webcam:

[54.1]http://pharos.ece.utexas.edu/wiki/index.php/How_to_Use_a_Webcam_in_ROS_with_the_usb_cam_Package

[54.2]http://pharos.ece.utexas.edu/wiki/index.php/How_to_install_ROS_from_Scratch#Install_the_Brown_ROS_packages

[54.3]http://wiki.oz9aec.net/index.php/Gstreamer_cheat_sheet

[54.4]https://code.google.com/p/brown-ros-pkg/wiki/ROSProcessingjs

[54.5]https://github.com/fpasteau/raspicam_node

[54.6]http://pharos.ece.utexas.edu/wiki/index.php/How_to_Use_a_Webcam_in_ROS_with_the_usb_cam_Package

[54.7]http://geekytheory.com/video-streaming-live-con-raspberrypi-y-playstation-eye/

[55] Problemas comunes de ROS:

[55.1]http://answers.ros.org/question/41652/ros-beginner-tutorials-problem/

[55.2]http://answers.ros.org/question/114326/roslaunch-not-working-during-tutorial-rosversion/

[55.3]http://answers.ros.org/question/9478/problem-installing-gscam/

[54] Web de Beaglebone con códigos de ejemplos para ROS:

[54.1]https://fleshandmachines.wordpress.com/category/beaglebone-2/

[55] Como instalar y utilizar un paquete ROS:

[55.1]http://geus.wordpress.com/2010/10/03/ros-como-instalar-y-utilizar-un-paquete-de-un-repositorio-federado/

[56] Python:

[56.1]http://www.ubuntu-es.org/node/166641#.U67Yxfl5OSp

[57] Programar Raspberry pi:

[57.1]http://geekytheory.com/tutorial-raspberry-pi-gpio-parte-2-control-de-leds-con-python/

[57.2]http://geekytheory.com/arduino-raspberry-pi-raspduino/

[57.3]http://geekytheory.com/arduino-raspberry-pi-lectura-de-datos/

[58] Instalación de librerias esenciales para Beablebone:

[58.1]http://www.stuffaboutcode.com/2013/08/compile-raspberry-pi-userland-raspivid.html

[59] Como usar ROS Groovy:

[59.1]http://www.tdrobotica.co/index.php?option=com_content&view=article&id=366

[60] Como instalar paquetes de repositorios no oficiales de ROS:

[60.1]http://wiki.ros.org/ROS/Tutorials/StackInstallation

[61] Rviz. ROS:

[61.1]http://wiki.ros.org/rviz/UserGuide

[62] Plugin necesario para camaras en ROS:

[62.1]http://answers.ros.org/question/28191/gscam-over-ubuntu-server/

[63]SD Formatter:

[63.1]https://www.sdcard.org/downloads/formatter_4/

[64]PuTTY:

[64.1]http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

[65]Hyperterminal:

[65.1]http://www.danielmunoz.com.ar/blog/2009/10/28/hyperterminal-en-windows-7-o-windows-vista/

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 1. Instalación de Ubuntu en PC

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción 4

2. Instalación del sistema operativo 4

3. Referencias 9

1. Introducción

La instalación de Ubuntu en un PC se puede llevar a cabo mediante un CD o USB. Ambos deberán tener la imagen de Ubuntu grabada para hacer la instalación. Esta instalación es sencilla e intuitiva ya que el menú de instalación es bastante simple.

2. Instalación del sistema operativo.

La elaboración de esta guía es la redacción paso a paso de la instalación de Ubuntu en el ordenador disponible en la finca, el cuál será el ROS maestro.

Lo primero es descargar Ubuntu desde su web oficial [1]. Posteriormente, una vez descargada, grabamos la ISO de Ubuntu en un CD o bien en un USB. Para ello deberemos usar un creador multiboot que podemos conseguir en la referencia [2]. En esta última referencia también se explica su uso.

La instalación empieza arrancando el ordenador con el dispositivo, CD o USB, conectado al PC. En su arranque deberemos entrar a la BIOS, para que esto ocurra, lo más común es pulsar la tecla *Del, Supr, F12 o F2*. Para que aparezca el menú de la BIOS. Una vez dentro, arrancaremos el sistema desde *Boot CD* o *Boot USB*. Esperamos y aparece la pantalla en la que debemos seleccionar el idioma de instalación. En esta pantalla también podemos entre probar Ubuntu, si queremos entrar en el sistema sin instalarlo o instalar Ubuntu en nuestro equipo.

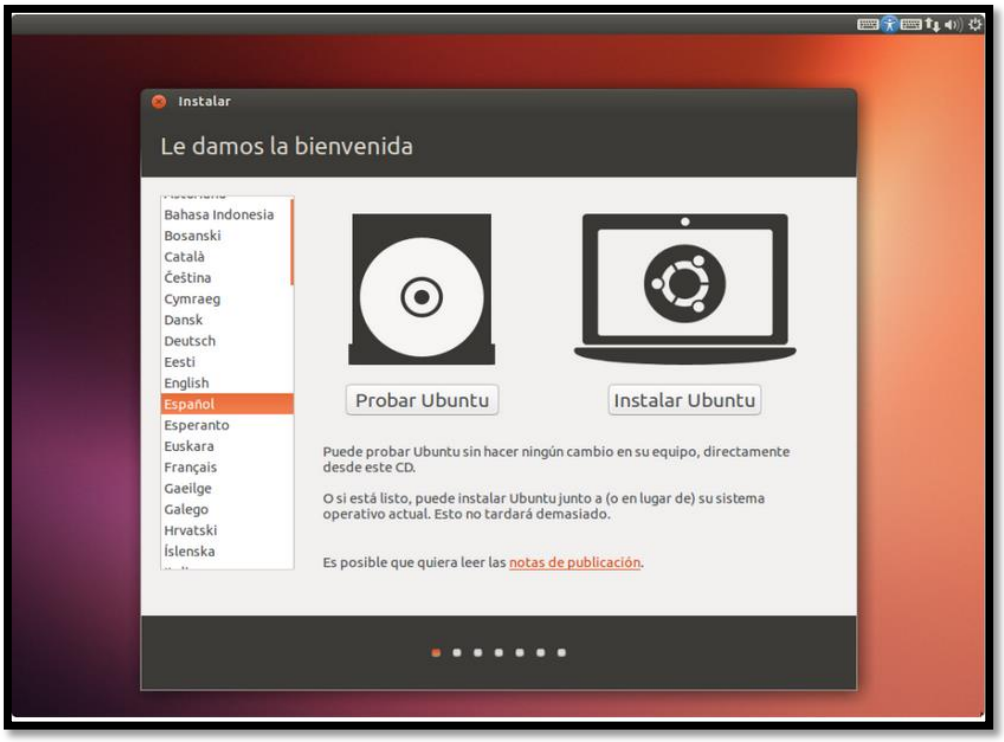


Figura 1. Menú instalación Ubuntu

Una vez seleccionado el idioma y la opción de probar o instalar, el programa de instalación verifica que el equipo cumple todos los requisitos mínimos para la instalación, así como el espacio en disco y conexión a internet.

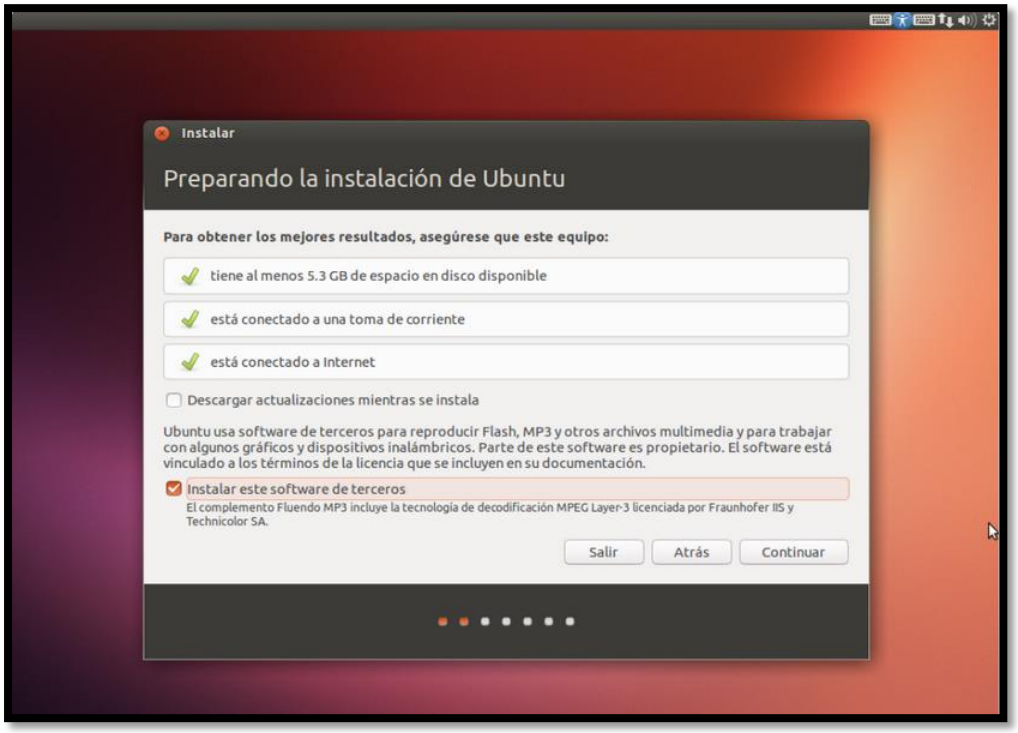


Figura 2. Menú comprobación instalación Ubuntu

Aquí se muestran dos opciones. Una de ellas que dice *descargar actualizaciones mientras se instala* y otra de ellas que dice *instalar este software de terceros.*

No se recomienda la primera opción porque ralentiza mucho el proceso de instalación y la segunda, lo que hace es instalar codecs adicionales por lo que es recomendable la segunda activada y la primera no. La actualización se hará después de tener listo el sistema. Una vez hecho esto, le damos a *continuar.*

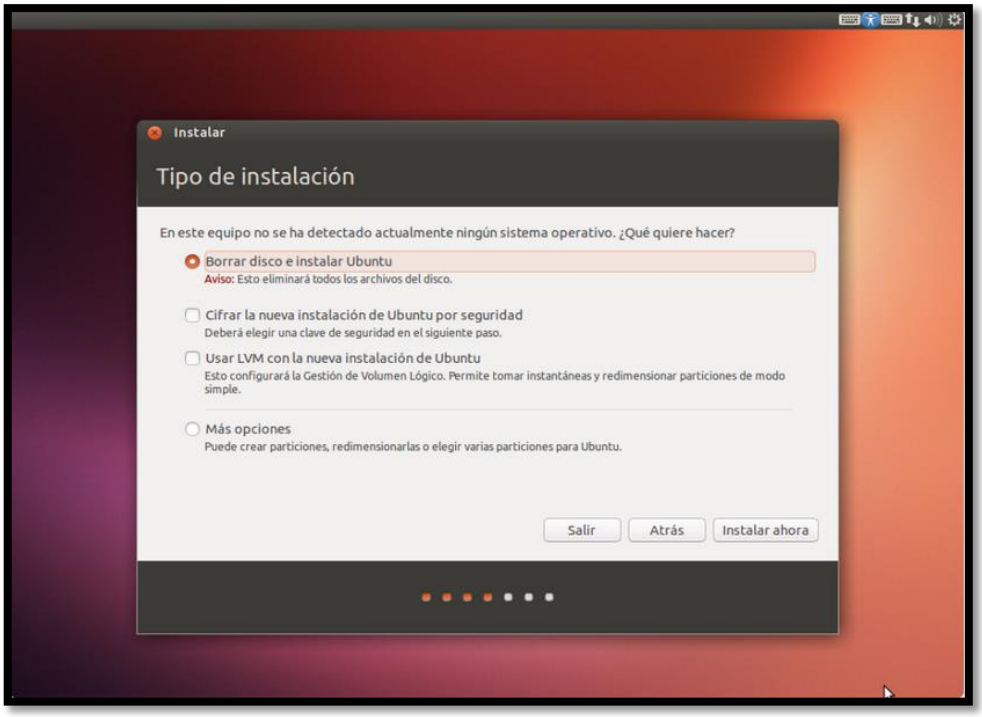


Figura 3. Elección tipo de instalación Ubuntu

Llegamos a un punto decisivo donde tenemos tres opciones:

- Instalar junto a otro sistema operativo si queremos compartir con otro S.O. el cual no es nuestro caso
- Reemplazar sistema operativo con Ubuntu. Formatea el PC y luego instala Ubuntu como único sistema. Esta es nuestra opción.
- La tercera y última opción es un menú donde podemos crear, editar y borrar particiones.

Seleccionamos la segunda opción y procedemos a instalar haciendo clic en *Instalar ahora.*

Luego, aparecerá una nueva pantalla del mismo menú donde empezará la configuración del sistema. Para comenzar tenemos que poner dónde nos encontramos, escribimos nuestra ciudad y le damos a continuar.

Figura 4. Menú configuración zona horaria Ubuntu

A continuación el teclado, indicamos nuestro idioma y la variación de idioma seleccionando en ambos español.

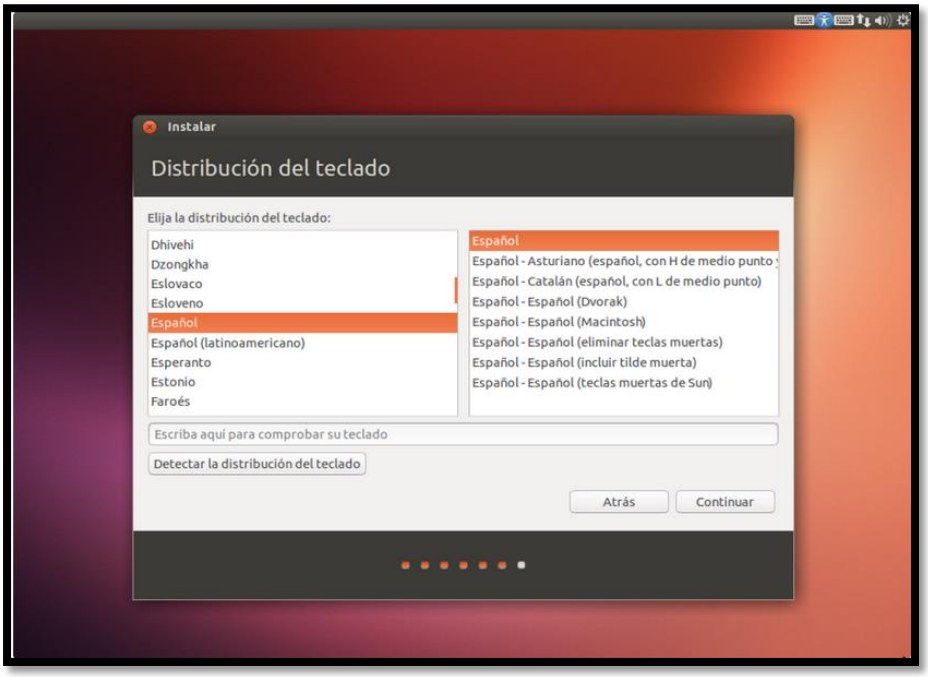


Figura 5. Configuración teclado en Ubuntu

Continuar y nos pregunta por nuestra información para crear el usuario. Lo rellenamos según corresponda.

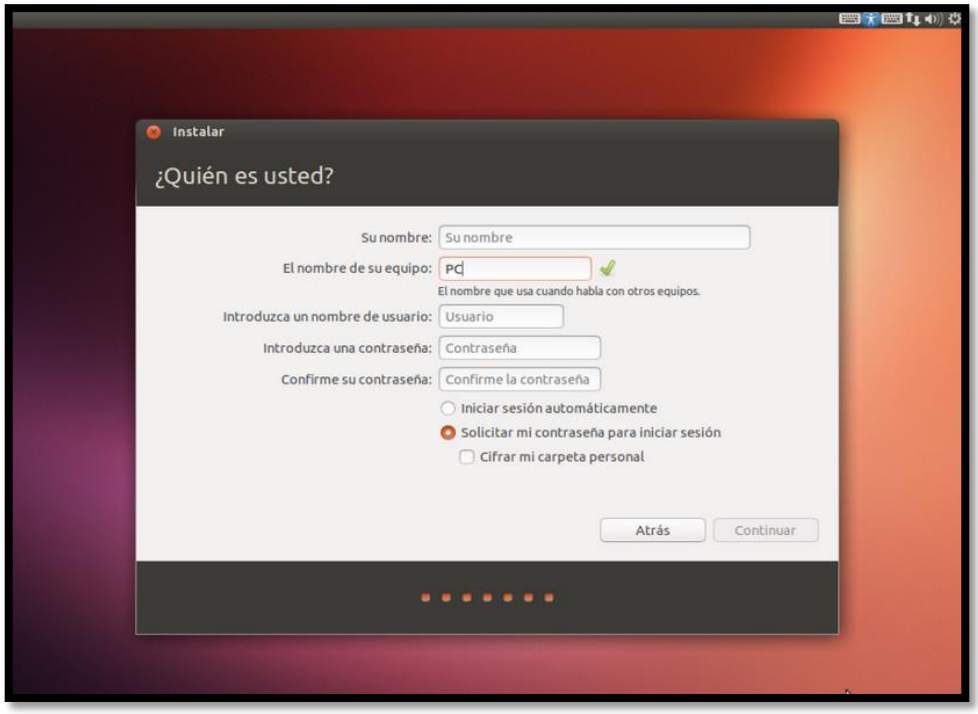


Figura 6. Configuración cuenta usuario

Con esto terminamos y toca esperar a que le sistema se instale.

Figura 7. Pantalla espera de la instalación Ubuntu

Cuando acabe el proceso, nos mostrará la opción de reiniciar para completar la instalación.

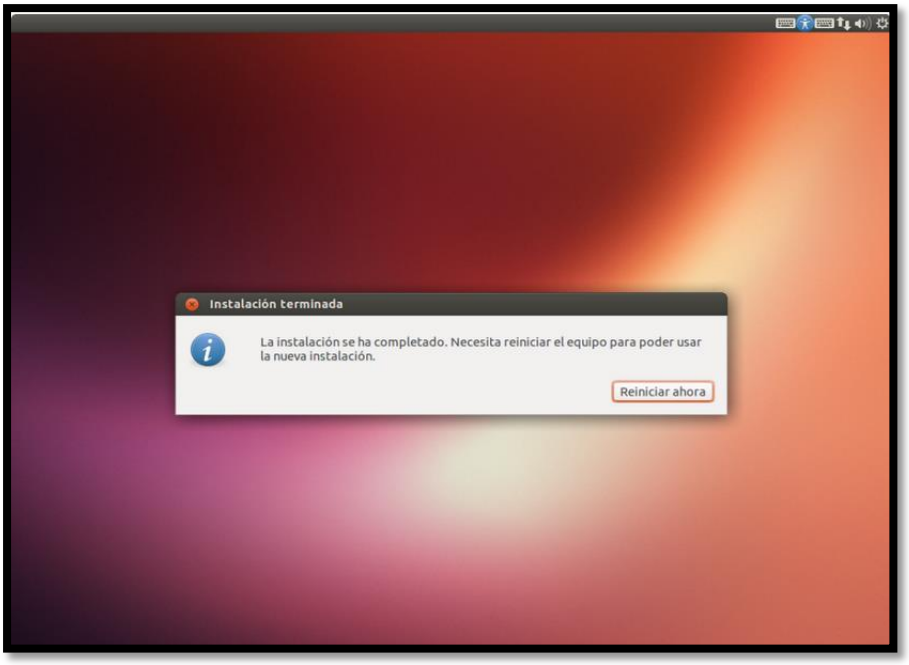


Figura 8. Pantalla final instalación Ubuntu, reinicio.

Una vez se reinicie, tenemos nuestro equipo configurado, nos pide la contraseña y entramos al escritorio donde ya tenemos todo disponible para uso y disfrute de Linux Ubuntu.

3. Referencias

[1] Ubuntu. Página de descarga http://www.ubuntu.com/download/desktop/

[2] Ubuntu.Yumi http://lignux.com/yumi-multiboot-usb-creator-windows/

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 2. Instalación de Ubuntu en Beaglebone

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción 4

2. Instalación del sistema operativo 4

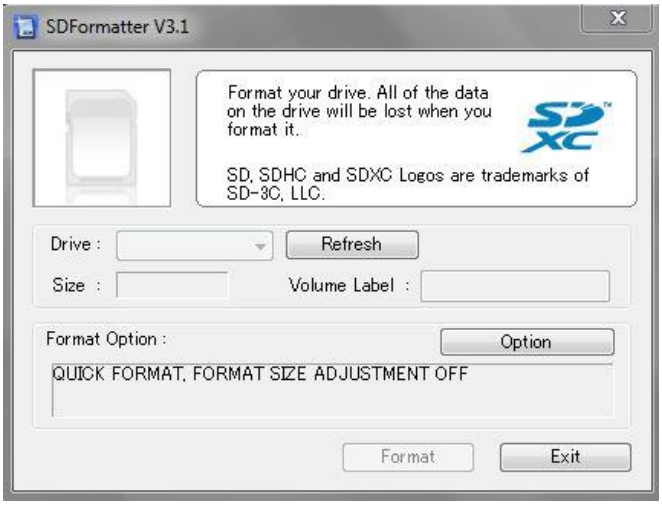
3. Referencias 18

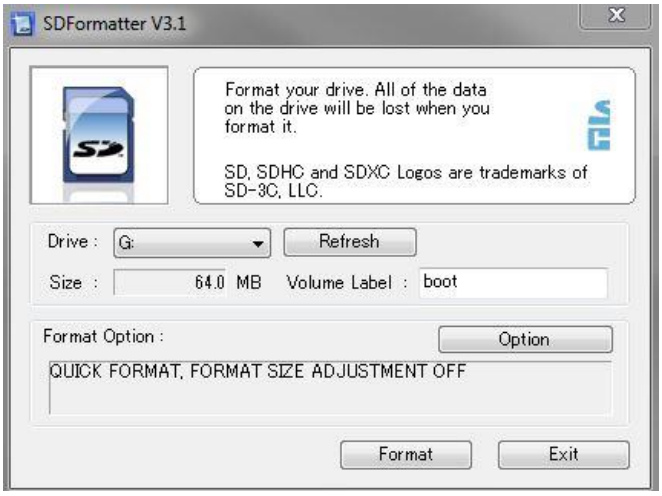
1. Introducción

Vamos a proceder a explicar la instalación de Ubuntu en Beaglebone, concretamente la versión Ubuntu-12.10-console-armhf-2013-09-26 (la cual se puede obtener de la referencia [1]). La instalación del sistema operativo Ubuntu en nuestro dispositivo consiste en grabar la imagen en una tarjeta SD.

2. Instalación del sistema operativo.

Lo primero de todo es darle formato a la tarjeta SD, para ello usaremos el programa SDFormatter (el cual se puede obtener de la referencia [2]):





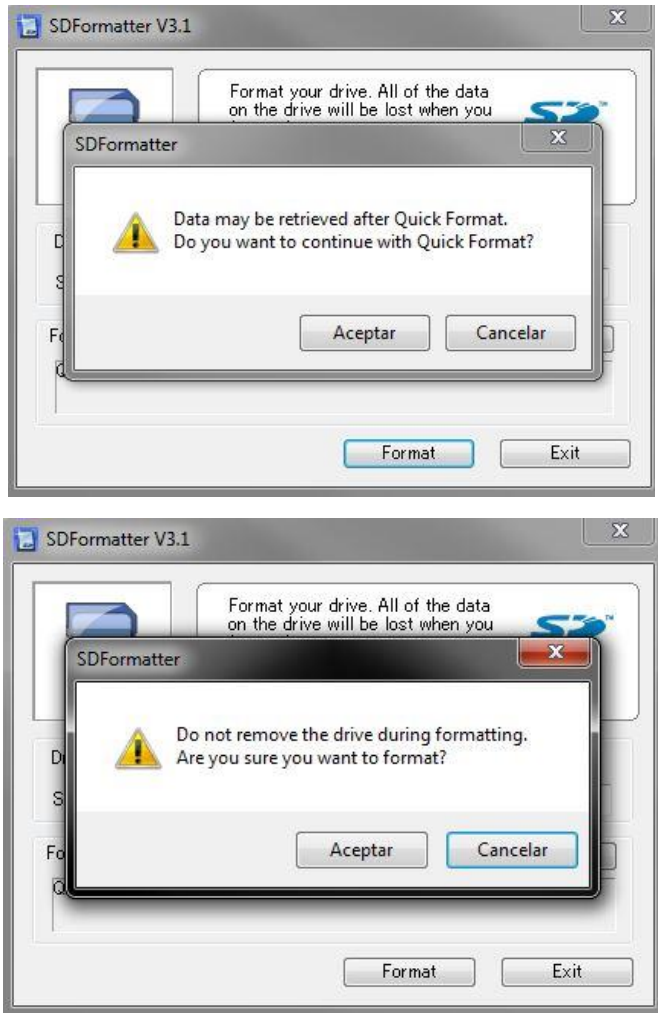


Figura 1. Visualización del uso de SDFormatter

Luego desde nuestro PC con Ubuntu, procedemos a la instalación de la imagen SD y compilación del kernel:

Para ello primero vemos donde tenemos montada nuestra tarjeta SD:

df –h

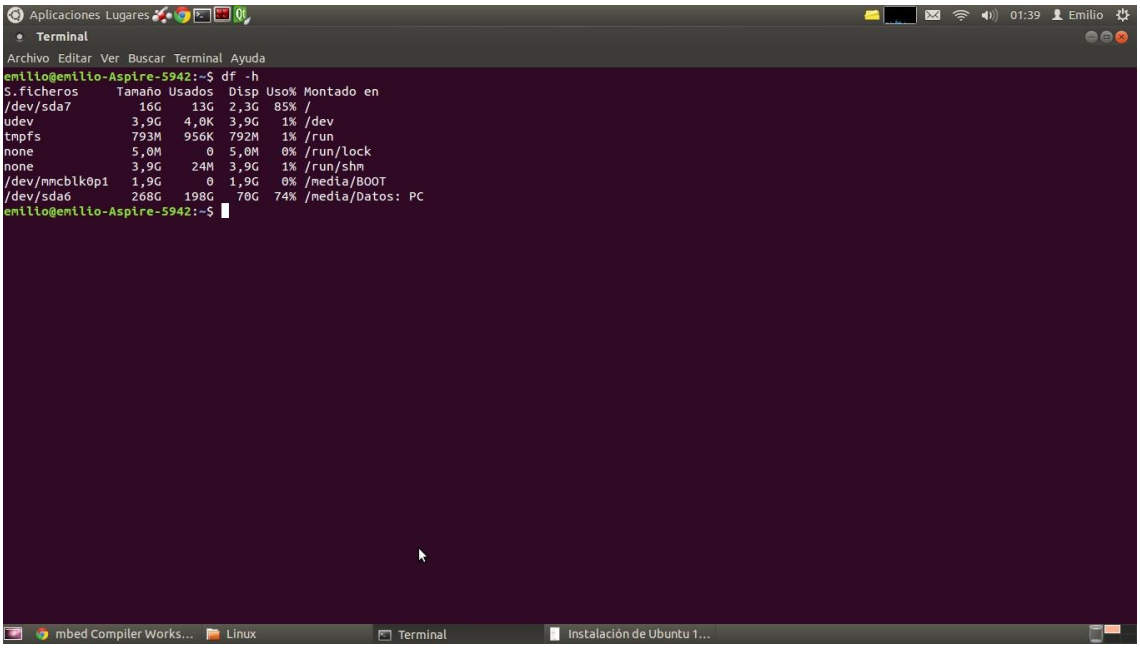


Figura 2. Dispositivos montados

Nos movemos hasta la carpeta que contiene la imagen de Ubuntu:

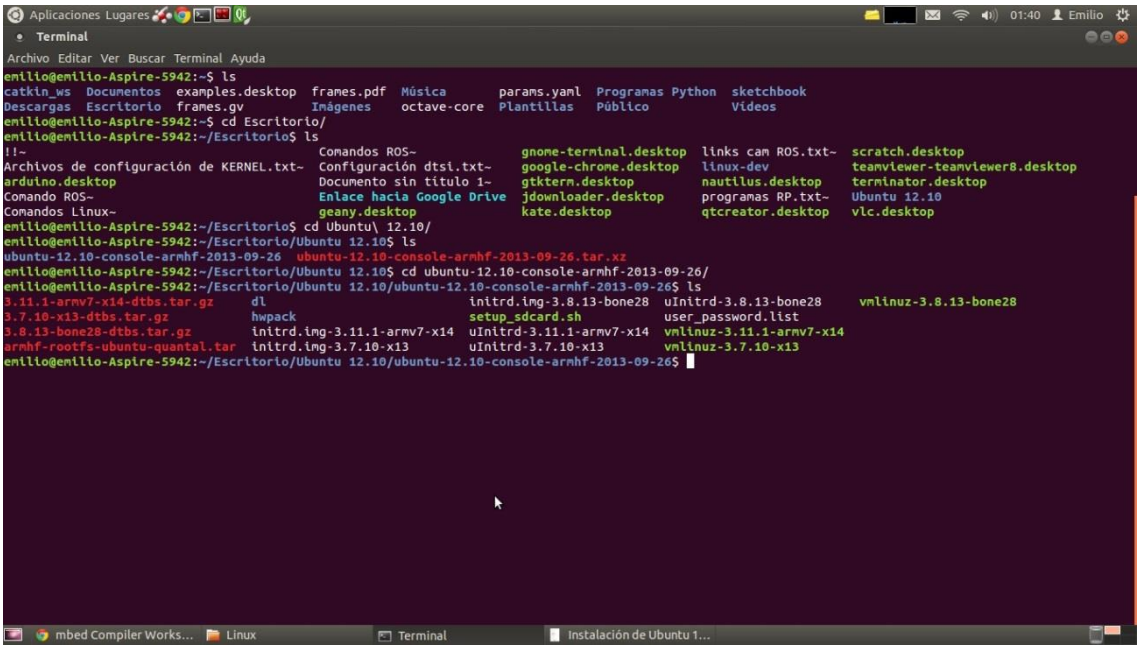


Figura 3. Directorio de la imagen de Ubuntu

Instalamos la imagen:

```
sudo ./setup_sdcard.sh --mmc /dev/mmcblk0 --uboot bone_dtb
```

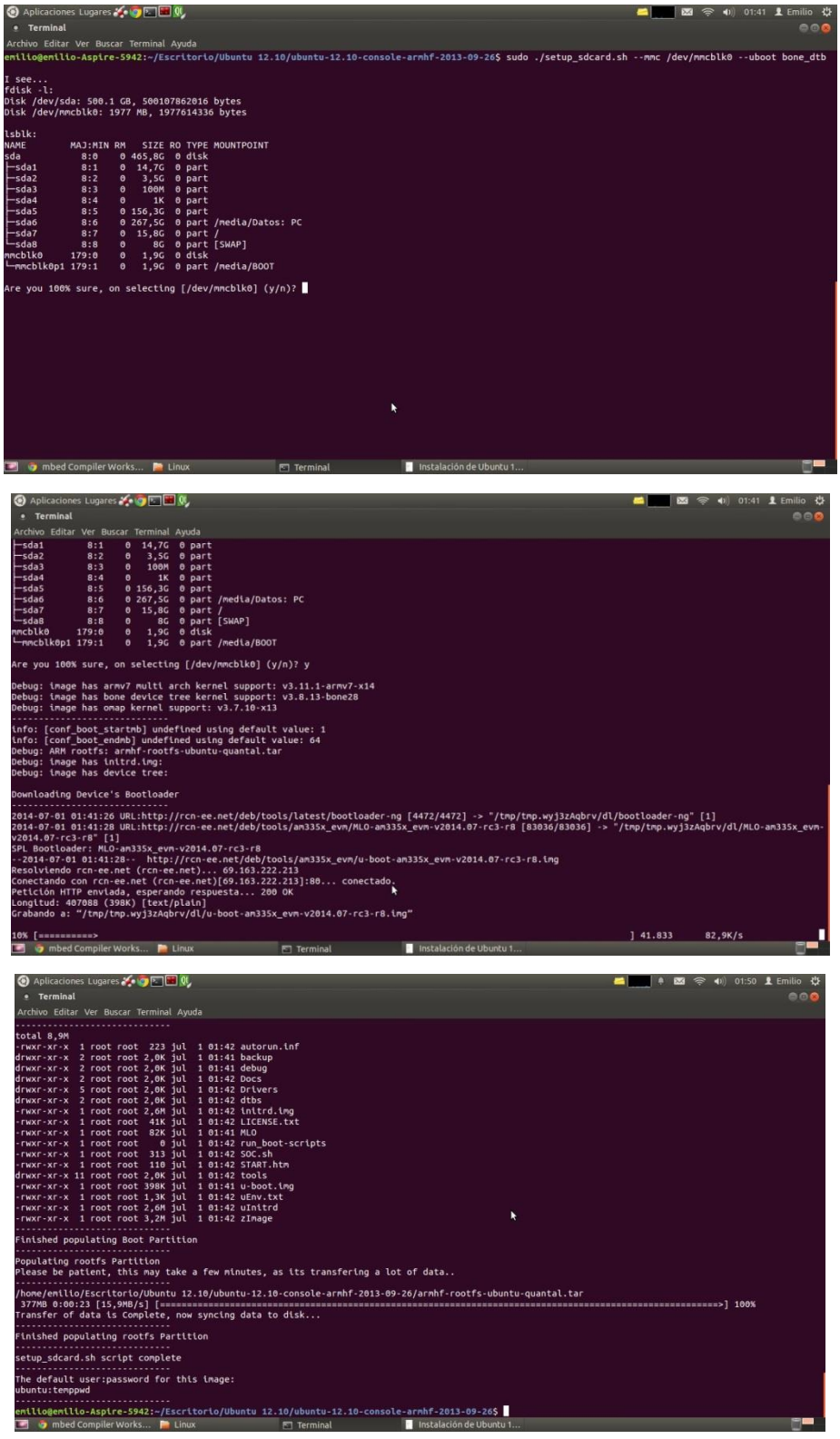


Figura 4. Instalación de Ubuntu en la SD

Después de instalar la imagen, nos movemos a otro directorio y descargamos el kernel:

sudo apt-get install git

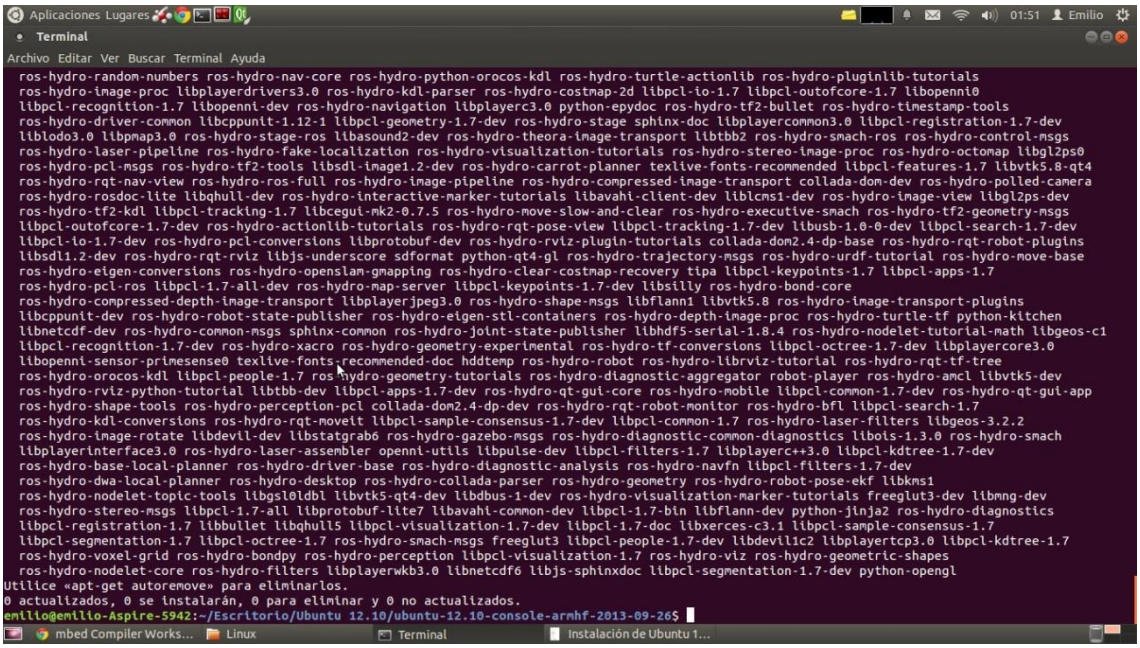


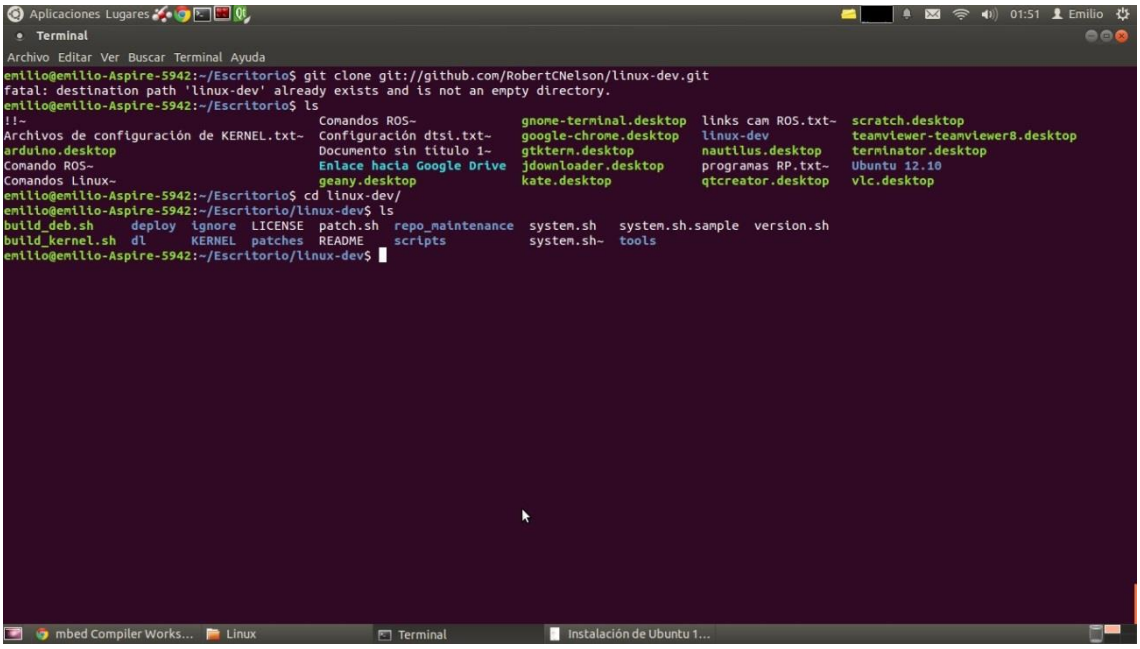
Figura 5. Instalación de git

git clone git://github.com/RobertCNelson/linux-dev.git

cd linux- dev/

git checkout origin/am33x-v3.8 -b tmp

cp system.sh.sample system.sh



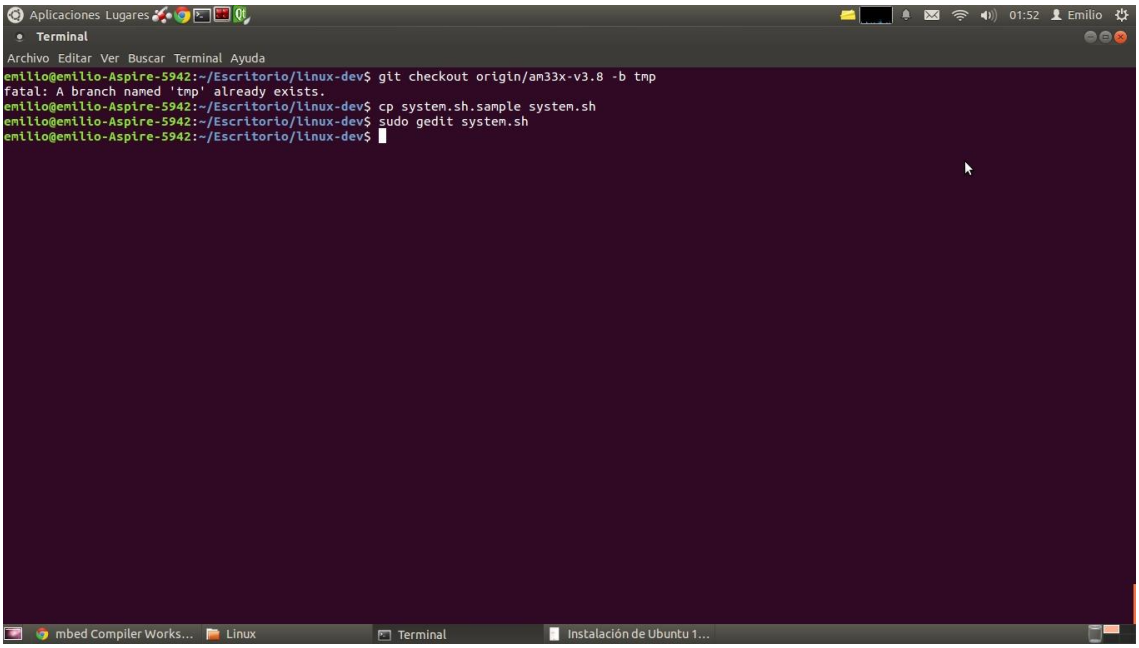


Figura 6. Descarga del kernel de Robert Nelson

sudo gedit system.sh

CC=arm-linux-gnueabihf- //Escribir esto va donde dice CC=

MMC=/dev/mmcblk0 //Escribir esto va al final del documento

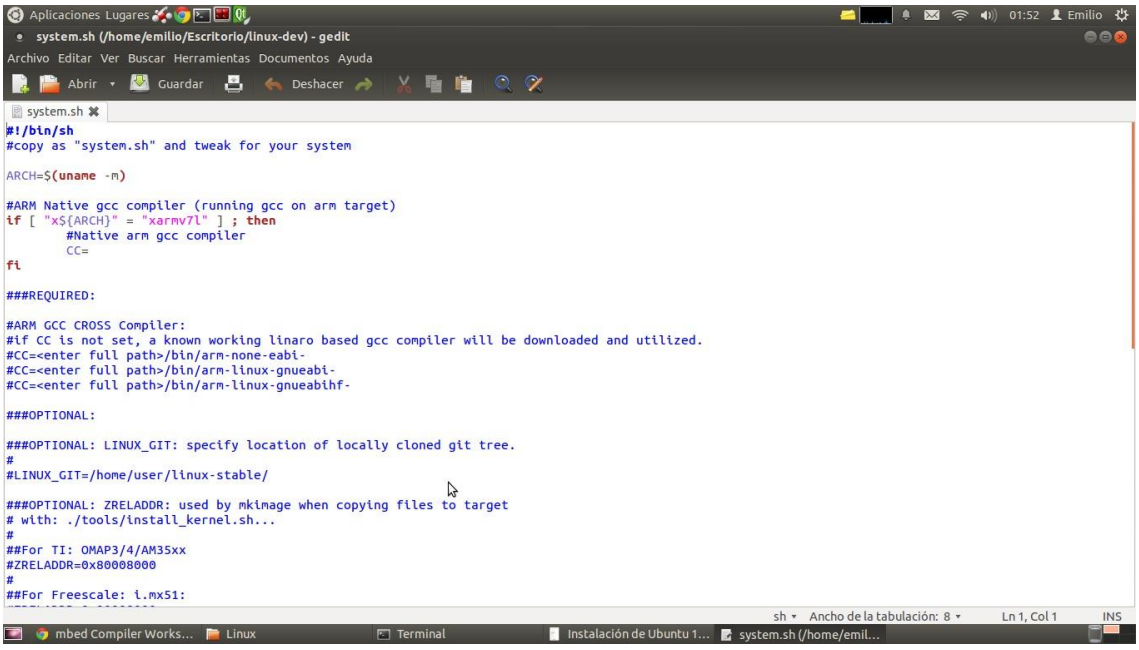


Figura 7. Documento system.sh

sudo gedit /etc/apt/sources.list

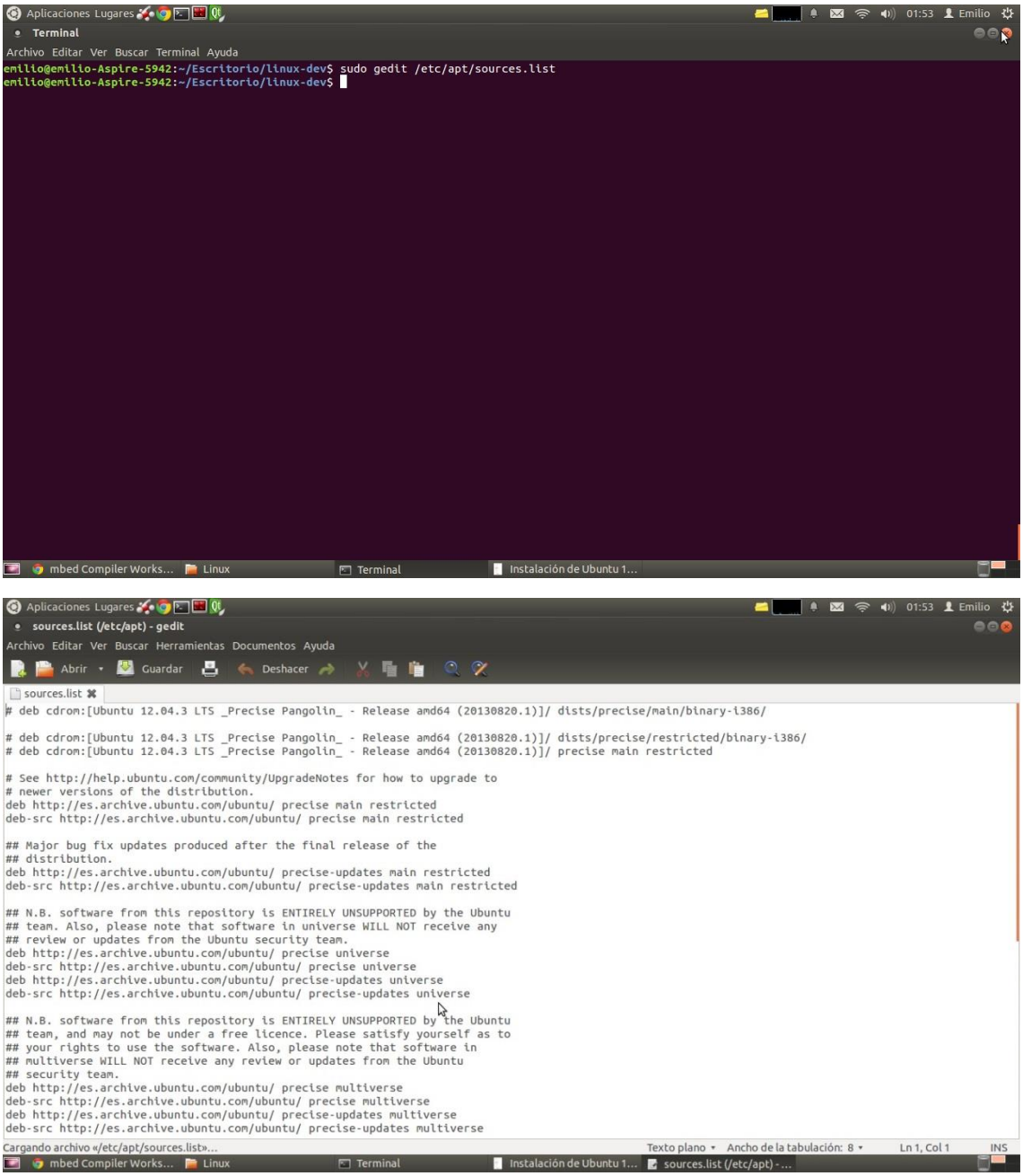


Figura 8. Documento donde editamos las fuentes de descargas

```
sudo apt-get update
```

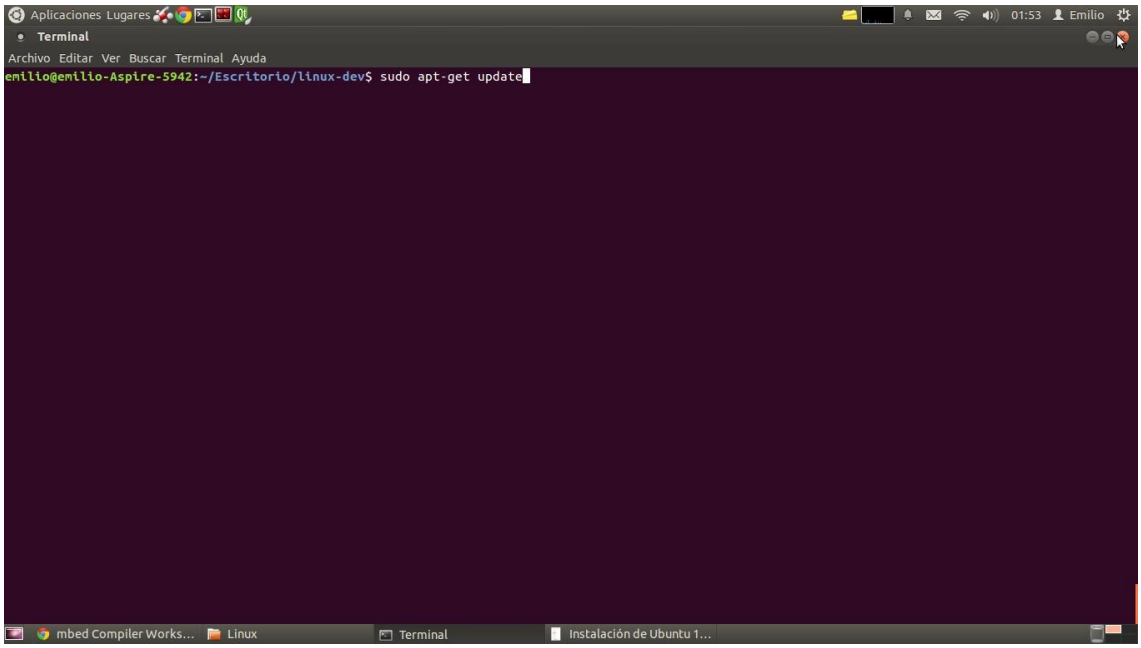


Figura 9. Actualización del sistema con repositorios nuevos

sudo apt-get install cpp-arm-linux-gnueabihf

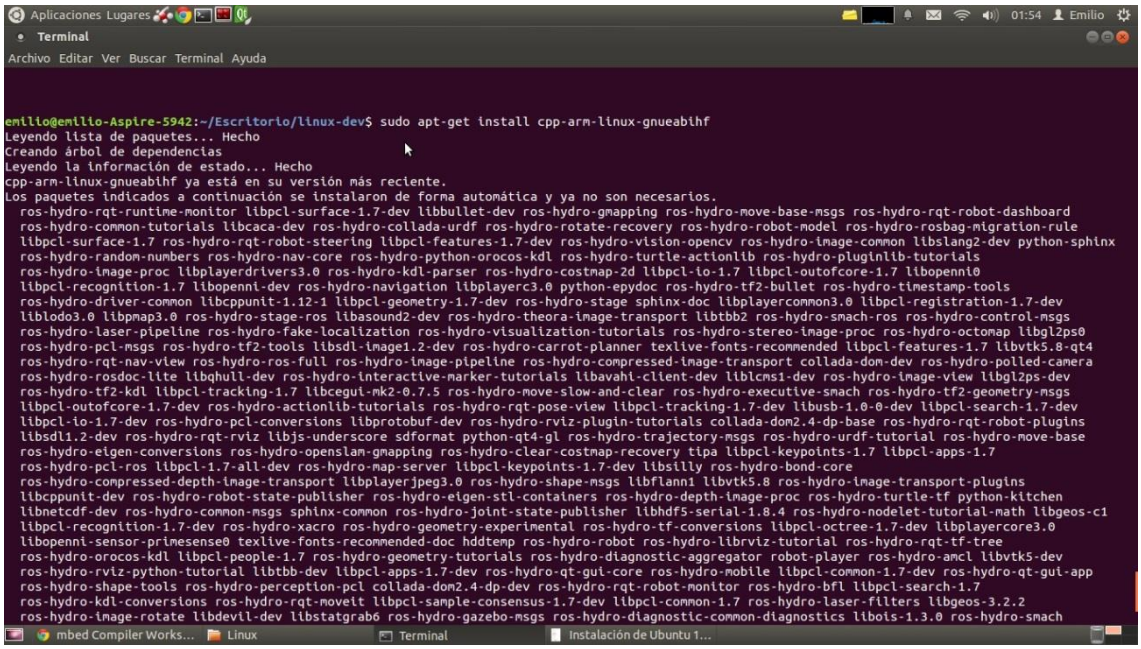


Figura 10. Instalación del compilador de GCC para armhf

git config --global user.name "Avelino y Emilio"

git config --global user.email tni.emilio@gmail.com

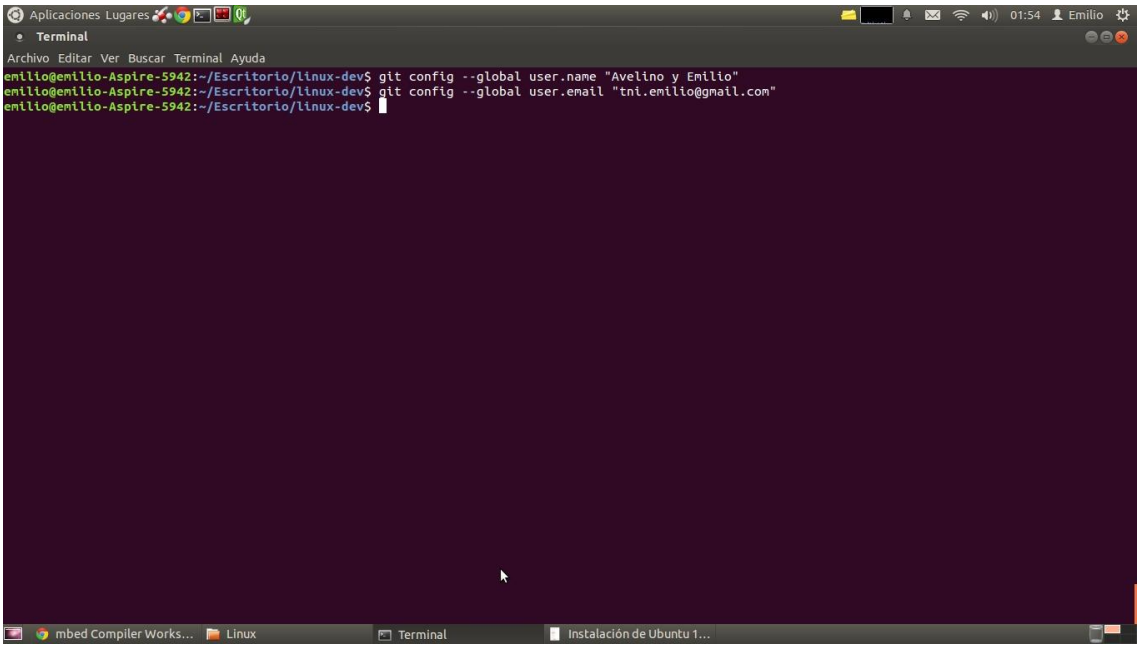


Figura 11. Configuración de repositorio git

Finalmente antes de proceder a la construcción del kernel, si quisiéramos cambiar un pin especifico de la Beaglebone, podemos usar el programa PINMUX (el cual se puede obtener del link de [3]), y añadir las librerías generadas a la carpeta kernel para su construcción:

Los archivos de cabecera generados están en el formato utilizado para el código fuente AM35x / AM37x / OMAP35x U-Boot.

Para otros dispositivos compatibles, el código fuente proporciona una descripción completa de todos los ajustes, pero no es directamente utilizable como código fuente U-Boot.

Para AM35x / AM37x / OMAP35x la configuración de PINMUX generada se pueden utilizar para personalizar el código fuente de U-Boot.

Antes de reconstruir U-Boot para su sistema, los pasos siguientes son necesarios:

1) Sustituir el archivo de cabecera mux.h con el archivo de salida mux.h de utilidad PINMUX.

2) Copia el archivo de salida pinmux.h en el directorio que contiene el archivo evm.h.

3) Modificar el archivo evm.h original, comentando o eliminando la sección original de código que hace que los pines de programación tenga diferentes llamadas.

4) Reemplace este código con # include "pinmux.h"

A continuación se muestra modificaciones al archivo evm.h original para incluir ajustes PINMUX de un archivo de cabecera por separado.

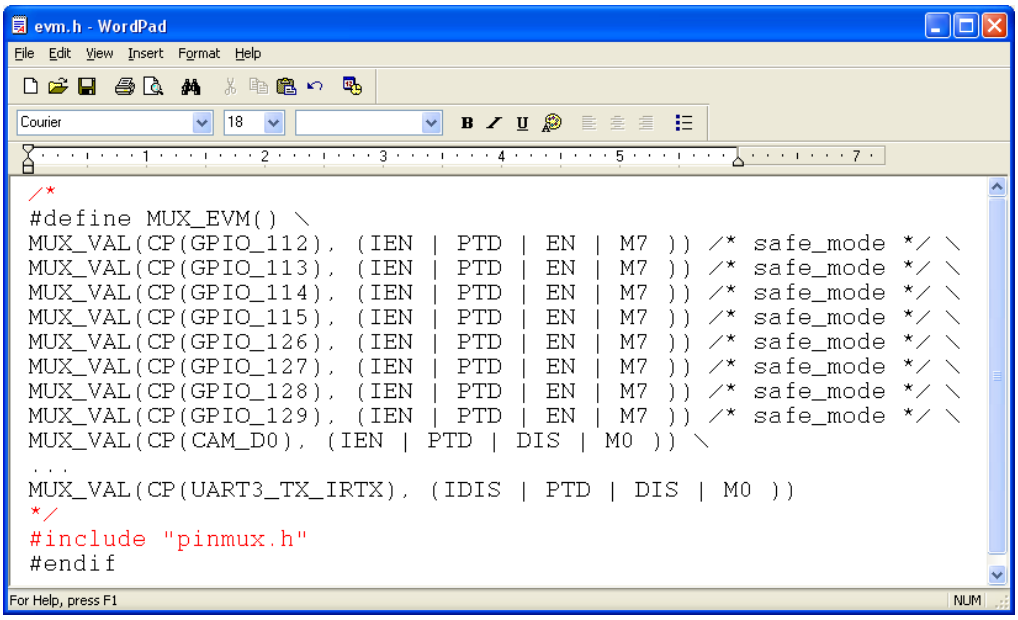


Figura 12. Ejemplo de modificación de pin

Locations for U-Boot Linux Source Files

Device Type	Device-Dependent Header File	Board-Dependent Header File
OMAP35xx	arch/arm/include/asm/arch-omap3/mux.h	board/ti/evm/evm.h board/ti/evm/pinmux.h (new)
AM37xx	arch/arm/include/asm/arch-omap3/mux.h	board/ti/evm/evm.h board/ti/evm/pinmux.h (new)
AM35xx	arch/arm/include/asm/arch-omap3/mux.h	board/logicpd/am3517evm/am3517evm.h board/logicpd/am3517evm/pinmux.h (new)

Figura 13. Localización de los archivos según microprocesador

El archivo mux.h original incluye el nombre de registro para OMAP35xx, AM35xx y AM37xx. Al establecer la configuración con el comando make, U-Boot se puede construir para cualquiera de estas plataformas. Sin embargo, la utilidad de PINMUX, genera archivo mux.h el cual sólo contendrá un registro para el dispositivo que se seleccionó cuando se ejecutó la utilidad PINMUX. Así, U-BOOT puede ser reconstruido para ese dispositivo seleccionado solamente.

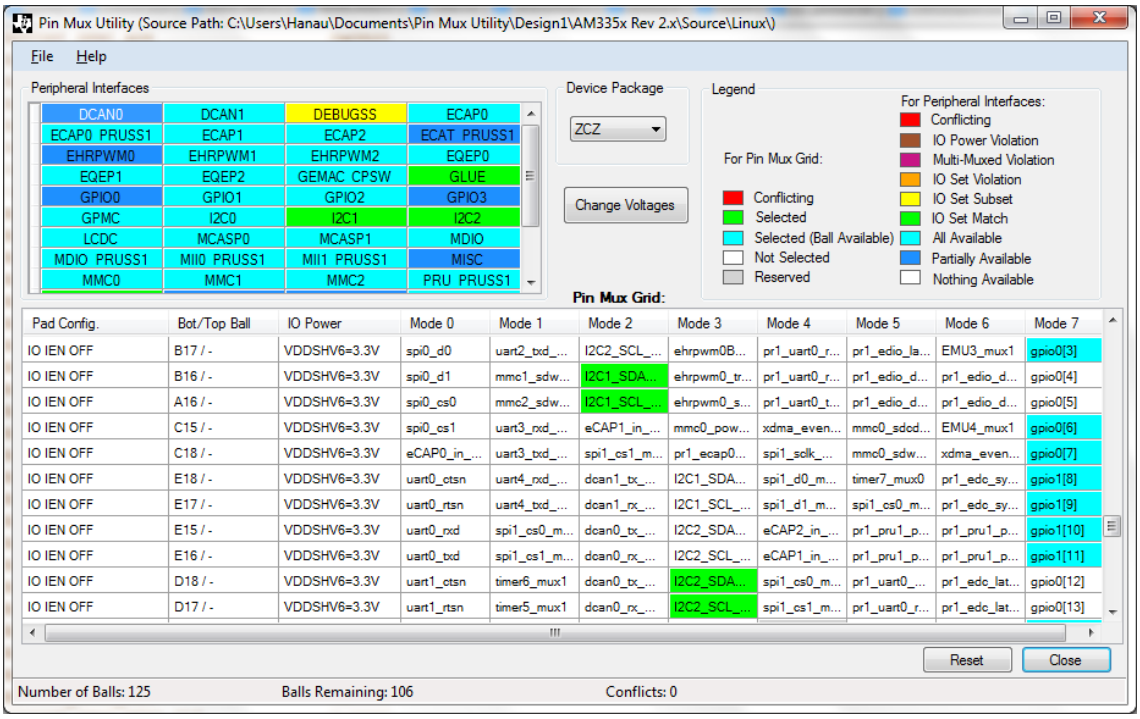
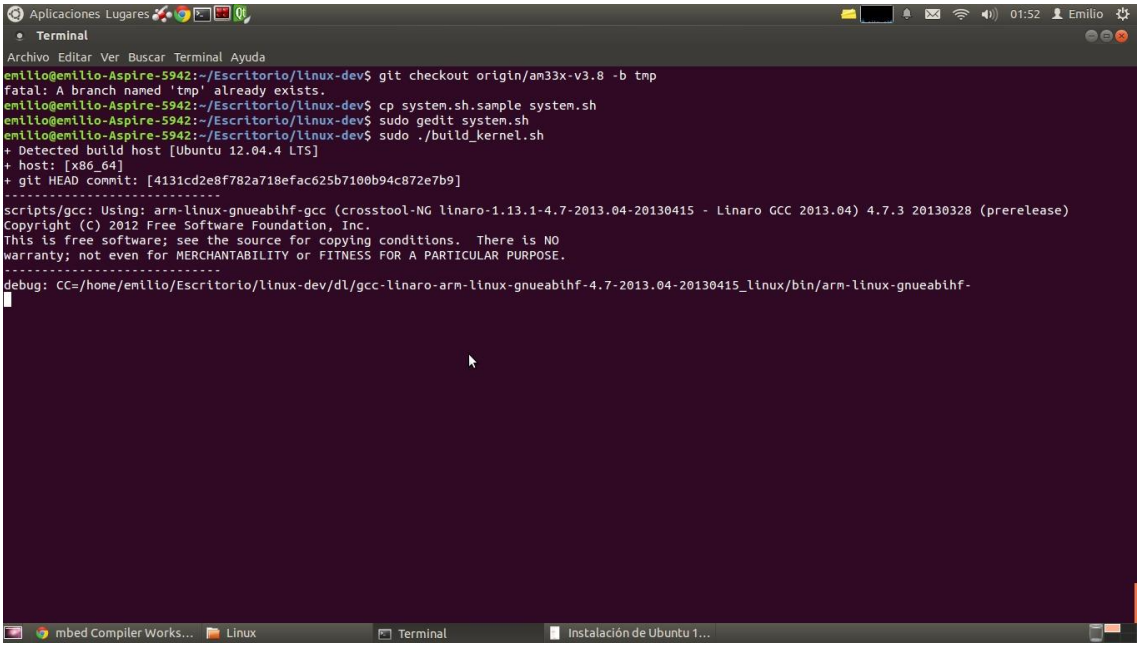


Figura 14. Herramienta PINMUX

Finalmente procedemos a la construcción del kernel:

```
sudo ./build_kernel.sh
```



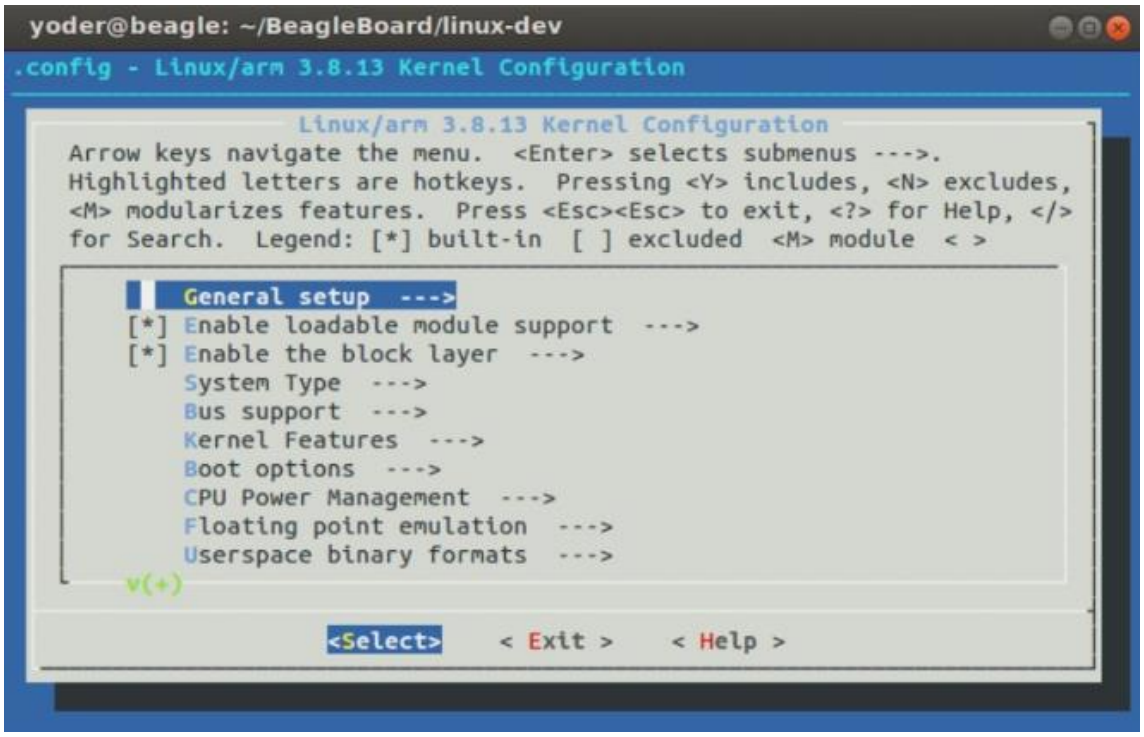


Figura 15. Proceso de construcción del kernel

Y finalmente instalamos el kernel compilado en la tarjeta SD con el Ubuntu ya instalado:

```
sudo tools/install_kernel.sh
```

Lo siguiente a hacer es conectar la Beaglebone al ordenador con Ubuntu o Windows, para poder trabajar con ella vía serial ya que no dispone de interfaz gráfica.

Para ello en Ubuntu tenemos que cargar un módulo a mano ya que no reconoce las uuid (Identificador Universalmente Único). Lo hacemos creando un fichero llamado /etc/udev/rules.d/73-beaglebone.rules con el siguiente contenido:

```
nano /etc/udev/rules.d/73-beaglebone.rules
ACTION=="add", SUBSYSTEM=="usb", ENV{DEVTYPE}=="usb_interface",
ATTRS{idVendor}=="0403", ATTRS{idProduct}=="a6d0",
DRIVER=="", RUN+="/sbin/modprobe -b ftdi_sio"
ACTION=="add", SUBSYSTEM=="drivers",
ENV{DEVPATH}=="/bus/usb-serial/drivers/ftdi_sio",
ATTR{new_id}="0403 a6d0"
ACTION=="add", KERNEL=="ttyUSB*",
ATTRS{interface}=="BeagleBone",
ATTRS{bInterfaceNumber}=="00",
SYMLINK+="beaglebone-jtag"
```

```
ACTION=="add", KERNEL=="ttyUSB*",
ATTRS{interface}=="BeagleBone",
ATTRS{bInterfaceNumber}=="01",
SYMLINK+="beaglebone-serial"
```

Ahora cada vez que conectemos la Beaglebone se detectara sin problema y podremos acceder a ella con el siguiente comando:

```
screen /dev/ttyUSB0 115200
```

Esto nos creara una interfaz en /dev/ttyUSBX done X es un número que depende de los dispositivos que tengas conectados, y el número 115200 es la velocidad de transmisión serial en baudios.

En caso de que exista algún problema con esto último, es recomendado instalar el programa Monitor Serial. Y con el accederemos a la BeagleBone tecleando este comando:

```
gtkterm --port /dev/ttyUSB0 --speed 115200
```

Desde Windows podemos acceder a la placa usando el software PuTTY[4] o Hiperterminal[5]. Pero antes debemos instalar los drivers de la Beaglebone desde su página oficial (los cuales se pueden obtener del link de [6]):

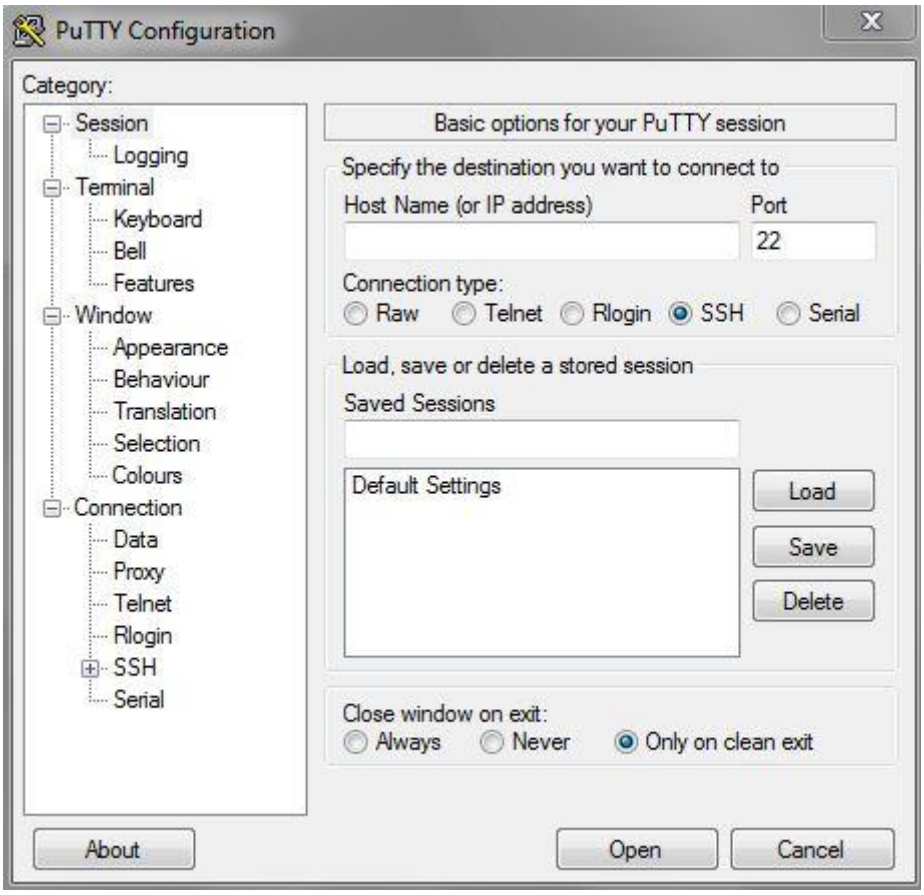


Figura 16. Visualización del uso de PuTTY

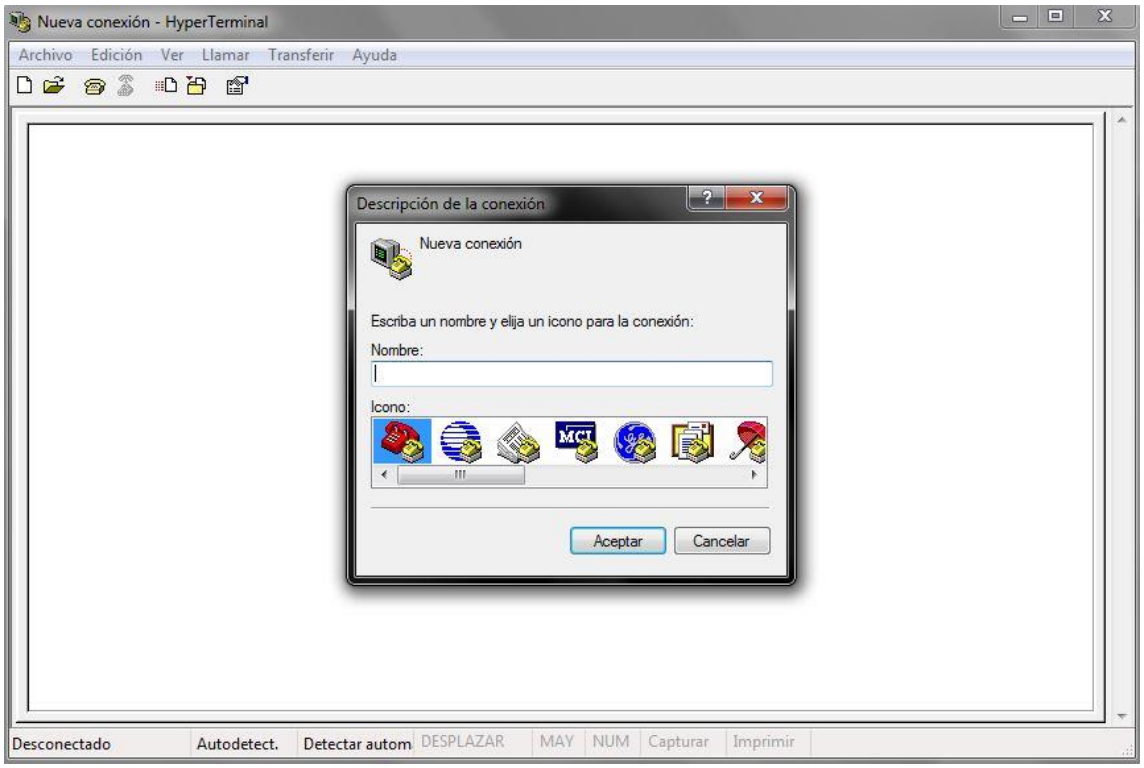


Figura 17. Visualización del uso de Hyperterminal

En nuestro primer acceso a la placa el usuario será ubuntu y la contraseña temppwd:

login=ubuntu

passwd=temppwd

Si quisiéramos cambiar la contraseña:

passwd

Para cambiar la contraseña de alce unta de administrador:

sudo-s

sudo passwd

Instalaciones necesarias en el sistema:

sudo apt-get install build-essential

sudo apt-get install gcc-arm-linux-gnueabi

sudo apt-get install screen

sudo apt-get install gstreamer0.10-plugins-good

sudo ntpdate pool.ntp.org

sudo apt-get update

sudo apt-get install build-essential python-dev python-setuptools python-pip python-smbus –y

sudo apt-get install openssh-server

sudo pip install Adafruit_BBIO

```
sudo apt-get install python
```

Para configurar el usbWiFi:

Conectarlo vía hub usb, ya que si no por defecto de la palca ocurre un error y se reinicia.

Primero de todo ejecutar el siguiente comando:

```
ifconfig -a
```

Debe aparecer una interfaz wlan0 en la lista.

Ahora ejecutar:

```
sudo nano /etc/network/interfaces
```

Y agregamos la configuración de nuestra red WIFI:

```
auto wlan0
iface wlan0 inet dhcp
wpa-ssid "Mi_Wifi"
wpa-psk "Clave"
```

Si quisiéramos tener IP estática:

```
iface eth0 inet static
address 192.168.0.2
netmask 255.255.255.0
gateway 192.168.0.1
```

Configuración del DNS:

```
sudo nano /etc/resolv.conf
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Ahora reiniciamos los servicios de red con el siguiente comando:

```
sudo /etc/init.d/networking restart
```

Si no se conecta usar directamente el siguiente comando:

```
sudo reboot
```

3. Referencias

[1] Ubuntu. Instalación de ubuntu-12.10-console-armhf-2013-09-26: http://wind.cs.purdue.edu/doc/beaglebone_ubuntu.html#sec7

[2]SD Formatter:]https://www.sdcard.org/downloads/formatter_4/

[3]Herramientas. Mux Pin y re compilación de un kernel: http://www.ti.com/tool/pinmuxtool

[4]PuTTY: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

[5]Hyperterminal: http://www.danielmunoz.com.ar/blog/2009/10/28/hyperterminal-en-windows-7-o-windows-vista/

[6] Webs útiles a la hora de empezar a usar la Beaglebone: http://beagleboard.org/Getting%20Started

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 3. Instalación de Angstrom en Beaglebone

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción 4

2. Instalación del sistema operativo 4

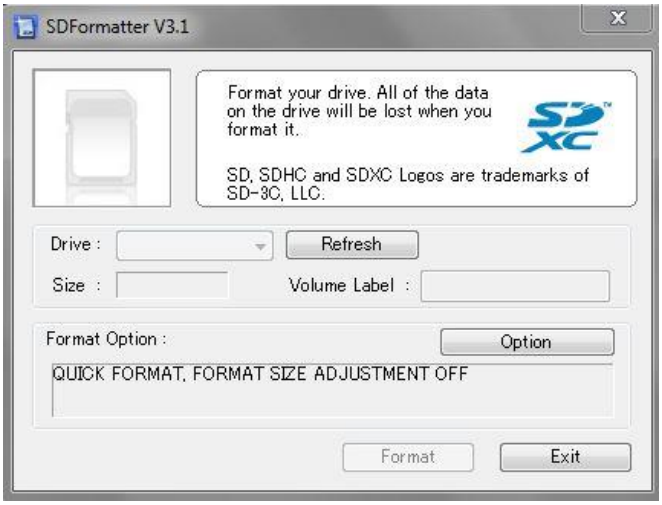
3. Referencias 10

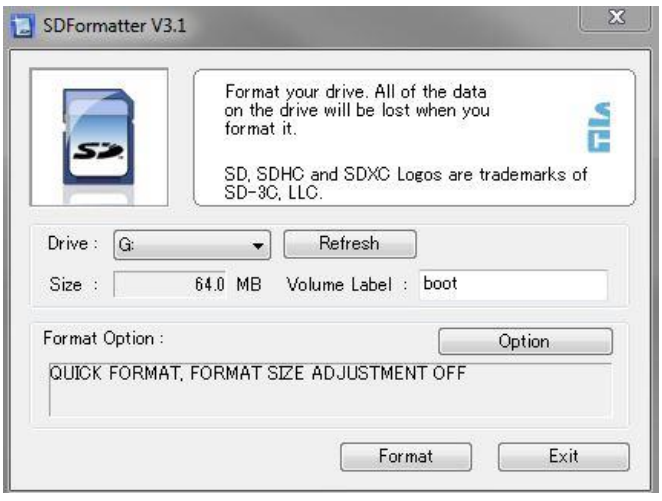
1. Introducción

Vamos a proceder a explicar la instalación de Angstrom en Beaglebone, concretamente la versión Angstrom-Cloud9-IDE-GNOME-eglibc-ipk-v2012.12-beaglebone-2013.09.05.img.xz (la cual se puede obtener en la referencia [1]). La instalación del sistema operativo Angstrom en nuestro dispositivo consiste en grabar la imagen en una tarjeta SD, mínima de 4GB de forma que el sistema operativo arranque desde la misma y no tenga que instalarse en la placa.

2. Instalación del sistema operativo.

Lo primero de todo es darle formato a la tarjeta SD, para ello usaremos el programa SDFormatter (el cual se puede obtener en la referencia [2]):





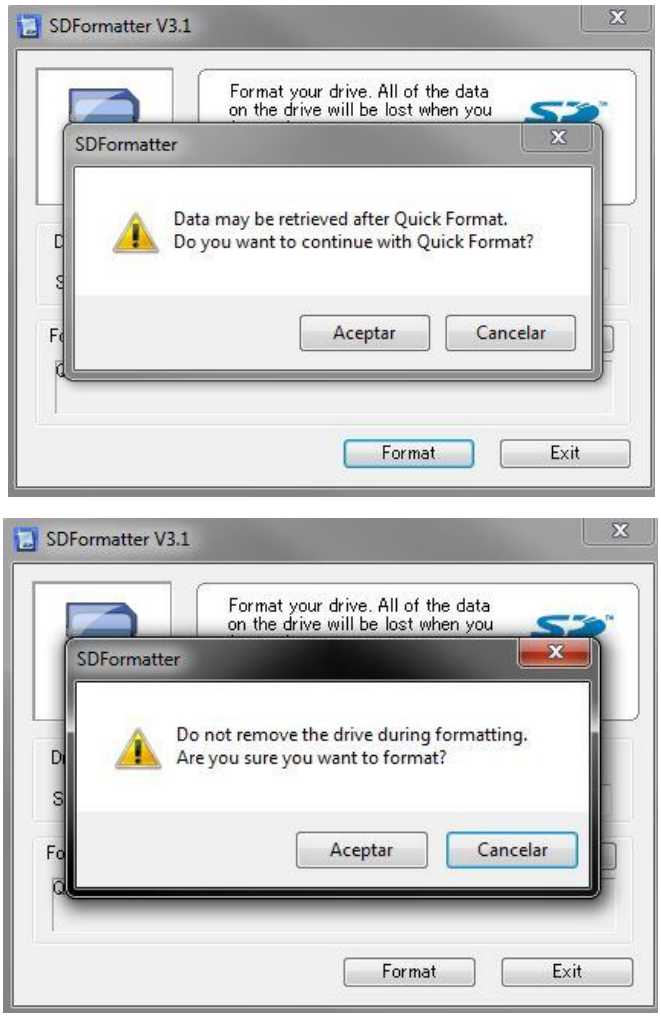


Figura 1. Visualización del uso de SDFormatter

Luego descomprimiremos la imagen con el programa 7-Zip (el cual se puede obtener en la referencia [3]):

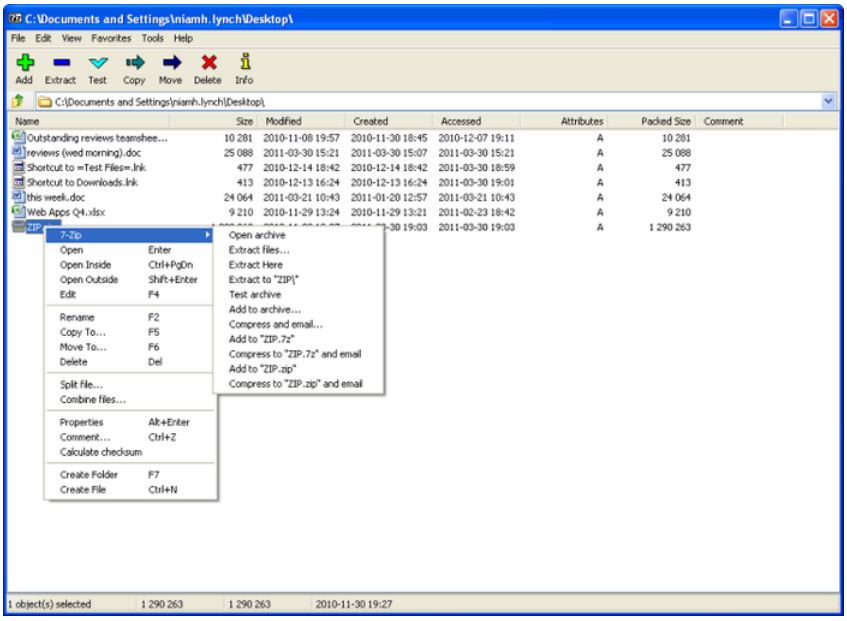


Figura 2. Visualización del uso de 7-Zip

Y seguidamente instalaremos la imagen en la SD con la ayuda de Win32 (el cual se puede obtener de la referencia [4]):

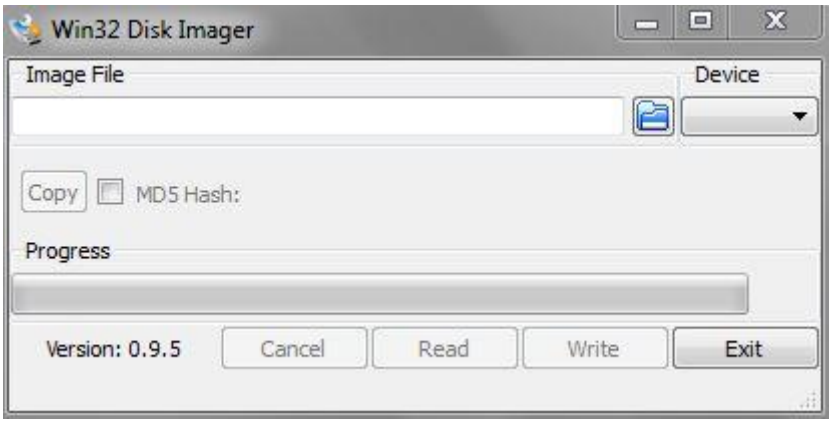


Figura 3. Visualización del uso de Win32

Lo siguiente a hacer es conectar la Beaglebone al ordenador con Ubuntu o Windows, para poder trabajar con ella vía serial ya que no dispone de interfaz gráfica.

Para ello en Ubuntu tenemos que cargar un módulo a mano ya que no reconoce las uuid (Identificador Universalmente Único). Lo hacemos creando un fichero llamado /etc/udev/rules.d/73-beaglebone.rules con el siguiente contenido:

```
nano /etc/udev/rules.d/73-beaglebone.rules
ACTION=="add", SUBSYSTEM=="usb", ENV{DEVTYPE}=="usb_interface",
ATTRS{idVendor}=="0403", ATTRS{idProduct}=="a6d0",
DRIVER=="", RUN+="/sbin/modprobe -b ftdi_sio"
ACTION=="add", SUBSYSTEM=="drivers",
ENV{DEVPATH}=="/bus/usb-serial/drivers/ftdi_sio",
ATTR{new_id}="0403 a6d0"
ACTION=="add", KERNEL=="ttyUSB*",
ATTRS{interface}=="BeagleBone",
ATTRS{bInterfaceNumber}=="00",
SYMLINK+="beaglebone-jtag"
ACTION=="add", KERNEL=="ttyUSB*",
ATTRS{interface}=="BeagleBone",
ATTRS{bInterfaceNumber}=="01",
SYMLINK+="beaglebone-serial"
```

Ahora cada vez que conectemos la Beaglebone se detectara sin problema y podremos acceder a ella con el siguiente comando:

```
screen /dev/ttyUSB0 115200
```

Esto nos creara una interfaz en /dev/ttyUSBX done X es un número que depende de los dispositivos que tengas conectados, y el número 115200 es la velocidad de transmisión serial en baudios.

En caso de que exista algún problema con esto último, es recomendado instalar el programa Monitor Serial. Y con el accederemos a la BeagleBone tecleando este comando:

```
gtkterm --port /dev/ttyUSB0 --speed 115200
```

Desde Windows podemos acceder a la placa usando el software PuTTY [5] o Hyperterminal [6]. Pero antes debemos instalar los drivers de la Beaglebone desde su página oficial (los cuales se pueden obtener del link de [7]):

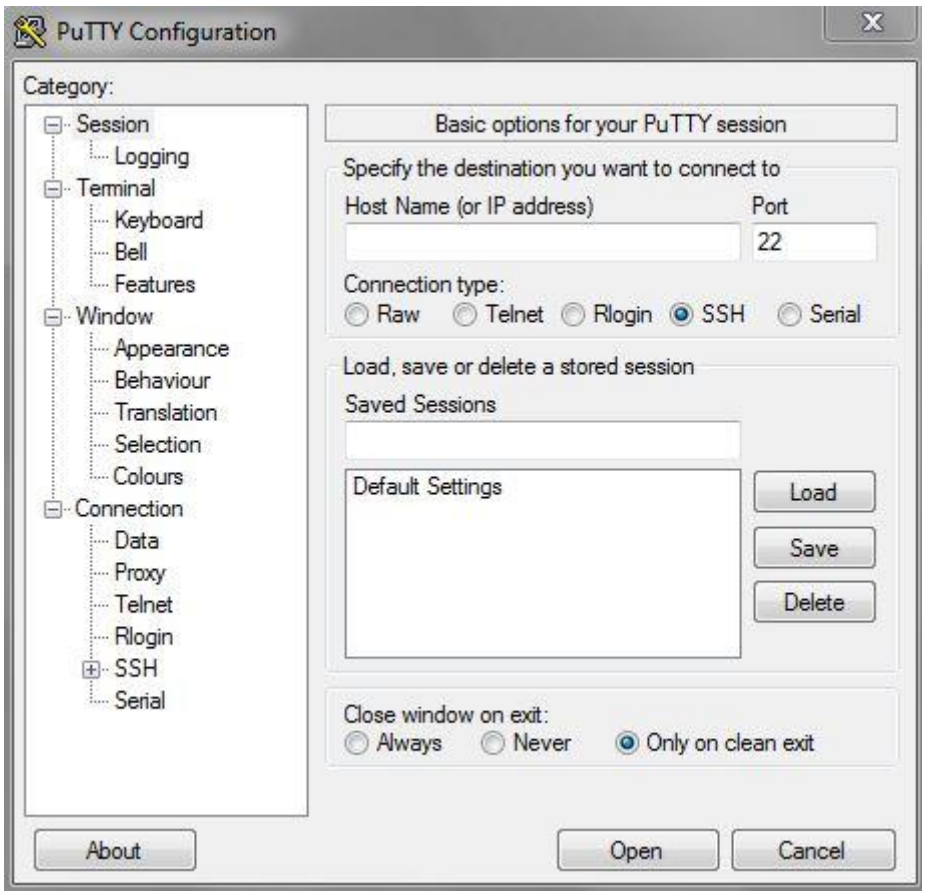


Figura 4. Visualización del uso de PuTTY

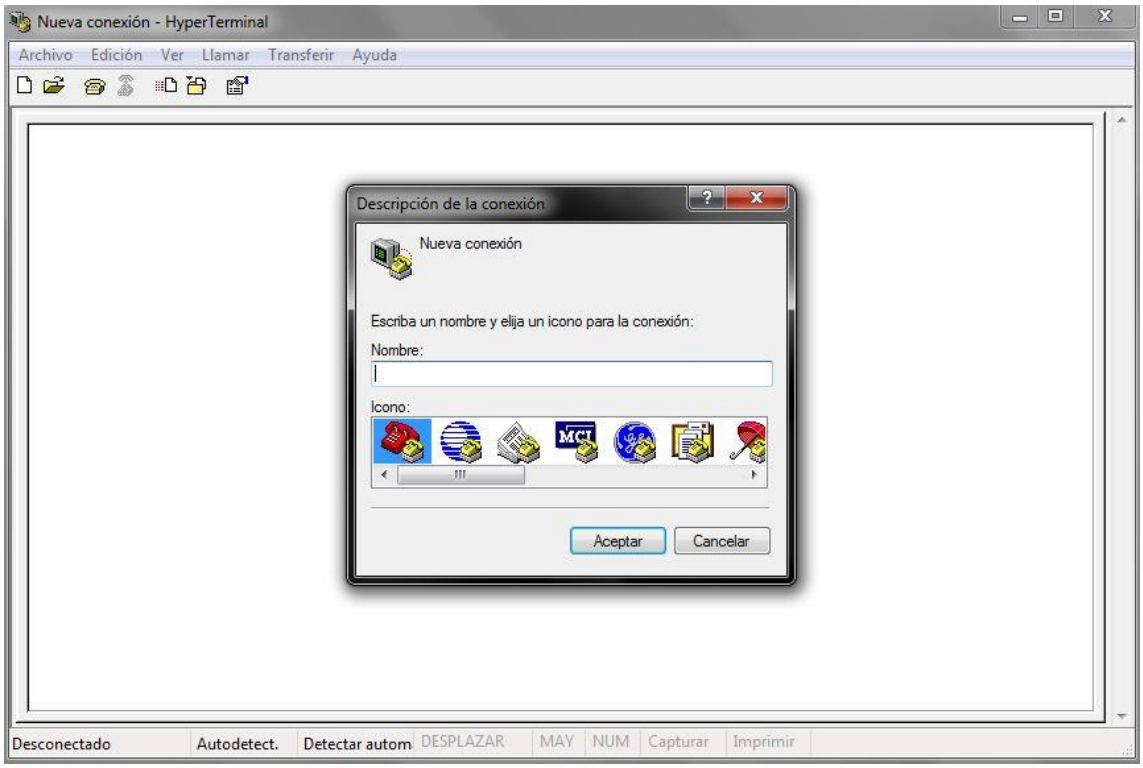


Figura 5. Visualización del uso de Hyperterminal

Figura 6. Pantalla de inicio de Angstrom en la Beaglebone

En nuestro primer acceso a la placa el usuario es root y no tiene contraseña.

login=root

passwd=

Instalaciones necesarias en el sistema:

opkg install build-essential

opkg install gcc-arm-linux-gnueabi

opkg install screen

/usr/bin/ntpdate -b -s -u pool.ntp.org

opkg update && opkg install python-pip python-setuptools python-smbus

sudo apt-get install openssh-server

```
sudo apt-get install gstreamer0.10-plugins-good
pip install Adafruit_BBIO
opkg install python
```

Para instalarle el usbWiFi:

```
opkg update
opkg upgrade
opkg install kernel-dev
opkg install kernel-headers
```

Reiniciamos la placa.

```
cd /usr/src/kernel
make scripts
ln -s /usr/src/kernel /lib/modules/$(uname -r)/build
cd ~
git clone git://github.com/cmicali/rtl8192cu_beaglebone.git
cd rtl8192cu_beaglebone
make CROSS_COMPILE=""
```

Instalamos el driver:

```
mv 8192cu.ko /lib/modules/$(uname -r)
depmod -a
cd /etc/modules-load.d
echo "8192cu" > rtl8192cu-vendor.conf
```

Eliminamos el driver de la lista negra de linux:

```
cd /etc/modprobe.d
echo "install rtl8192cu /bin/false" >wifi_blacklist.conf
echo "install rtl8192c_common /bin/false" >>wifi_blacklist.conf
echo "install rtlwifi /bin/false" >>wifi_blacklist.conf
```

Editamos /var/lib/connman/settings y activamos el WIFI:

```
nano /var/lib/connman/settings
[global]
OfflineMode=false
[Wired]
Enable=true
```

```
Tethering=false
[WiFi]
Enable=true
Tethering=false
```

Encriptamos la clave:

```
wpa_passphrase YourSSID YourPassphrase
```

Editamos /var/lib/connman/wifi.config:

```
nano /var/lib/connman/wifi.config
[service_home]
Type=wifi
Name=YourSSID
Passphrase=YourEncryptedPassphrase
```

Conectamos el usbwifi y reiniciamos la Beaglebone:

```
shutdown -r 0
```

3. Referencias

[1] Beaglebone. Imagen Angstrom: http://downloads.angstrom-distribution.org/demo/beaglebone/

[2]SD Formatter: https://www.sdcard.org/downloads/formatter_4/

[3] 7-Zip:]http://www.7-zip.org/

[4] Webs útiles a la hora de empezar a usar la Beaglebone:

http://beagleboard.org/Getting%20Started

[5]PuTTY: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

[6]Hyperterminal: http://www.danielmunoz.com.ar/blog/2009/10/28/hyperterminal-en-windows-7-o-windows-vista/

[7] Webs útiles a la hora de empezar a usar la Beaglebone:
http://beagleboard.org/Getting%20Started

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 4. Instalación de Raspbian en Raspberry pi

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción 4

2. Instalación del sistema operativo 4

3. Configuración adicional
   3.1. Instalación de cámara 10
   3.2. Configuración puente WiFi-Ethernet. 15

4. Referencias 17

1. Introducción

La instalación de Raspbian se lleva a cabo mediante la instalación de la imagen del sistema operativo en una tarjeta Micro SD. En nuestro caso utilizaremos una tarjeta Micro SD de 8 Gb de memoria ya que es el mínimo que acepta esta instalación.

Los pasos para tener el sistema Raspbian funcionando en nuestro dispositivo son, primeramente, descargar la imagen, instalar la misma en la tarjeta SD, insertar la tarjeta en el dispositivo y alimentarlo. Posteriormente, se configura a nuestras necesidades.

2. Instalación del sistema operativo.

En este apartado se detallan los pasos para la instalación de Raspbian en Raspberry pi. El primer paso es descargar la imagen del sistema operativo. En nuestro caso, descargaremos la imagen desde su página oficial [1]. En esta página se puede encontrar todo sobre Raspberry pi.

Vamos al apartado de descargas y nos encontramos con todos los sistemas operativos que soporta Raspberry pi. En nuestro caso escogemos el Raspbian y lo descargamos en nuestro dispositivo.

A continuación, una vez descargado el archivo, necesitamos una aplicación que nos instale el sistema en la tarjeta SD. Este paso lo solventamos con la aplicación *win32DiskImager*. Podemos encontrar toda su información en la referencia [2].

Una vez tenemos la tarjeta SD finalizada, la introducimos en el conector de la Raspberry pi y encendemos. En nuestro caso conectamos la Raspberry pi a un monitor vía HDMI pero también se puede hacer desde la consola de nuestro equipo con Ubuntu tecleando el siguiente comando:

$$ssh - X\ pi@ < dirección\ IP\ de\ tu\ Raspberry\ pi >$$

La dirección IP del dispositivo se puede conocer accediendo al router al que esté conectado.

Arrancamos la placa. Nos mostrará el proceso de arranque y posteriormente un menú. A este menú se puede acceder cuando queramos, no sólo en el arranque, tecleando el siguiente comando:

$$sudo\ raspi - config$$

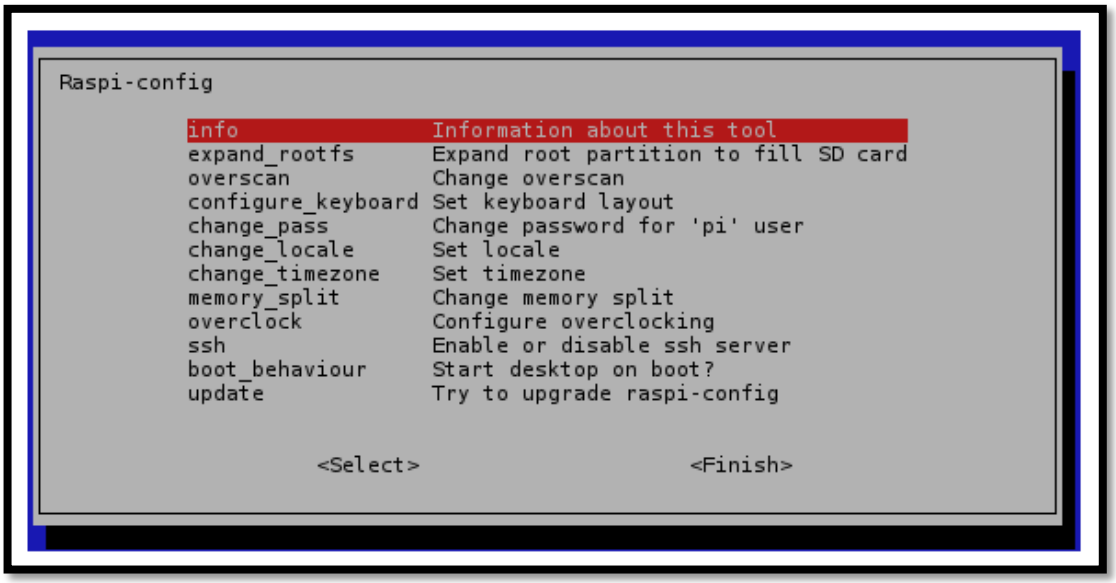


Figura 1. Menú configuración Raspberry pi

En este menú se puede hacer las siguientes acciones:

- ✓ *Configure_keyboard.* En este apartado podemos cambiar la configuración del teclado. Una vez dentro, seleccionaremos con la barra espaciadora la opción *es_ES* y luego en el conjunto de caracteres la opción de *es_ES UTF.8* y siguiente.

- ✓ *Change_pass.* Aquí podremos cambiar la contraseña para acceder al sistema. Por defecto son:

  $Usuario: pi$
  $Contraseña: raspberry$

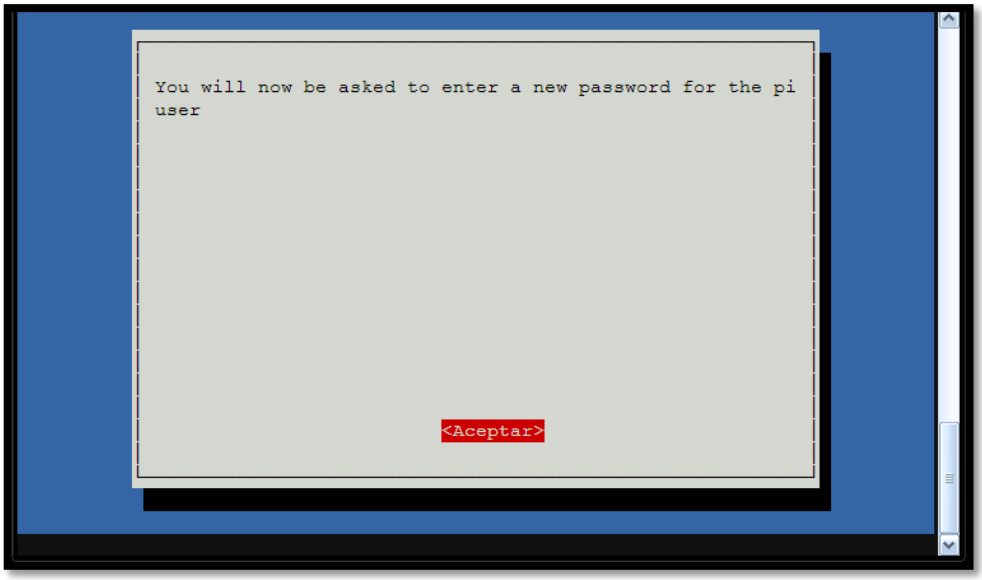


Figura 2. Menú configuración Raspberry pi. Confirmación cambio de contraseña.

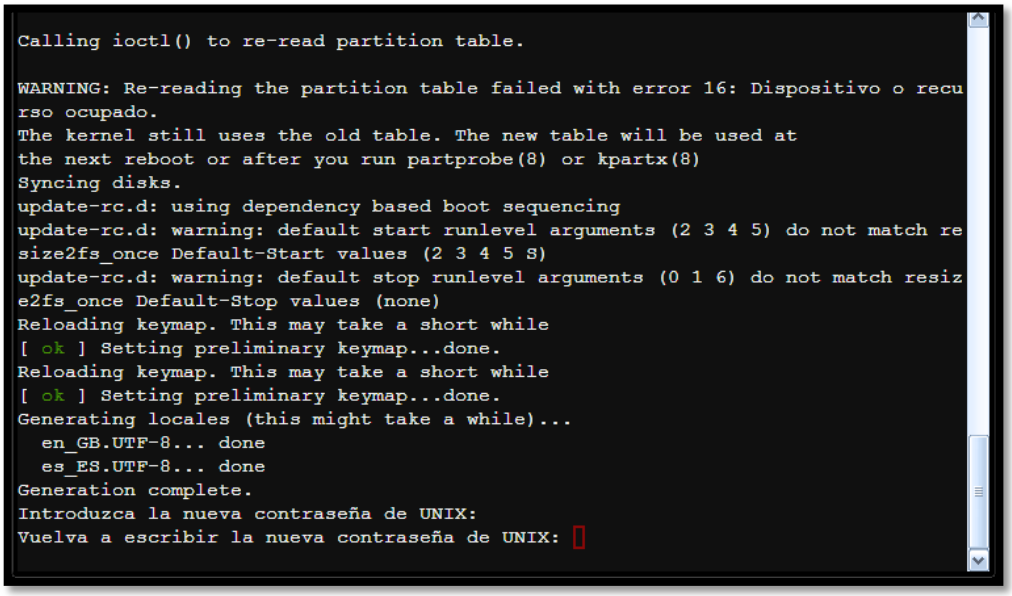


Figura 3. Configurar nueva contraseña

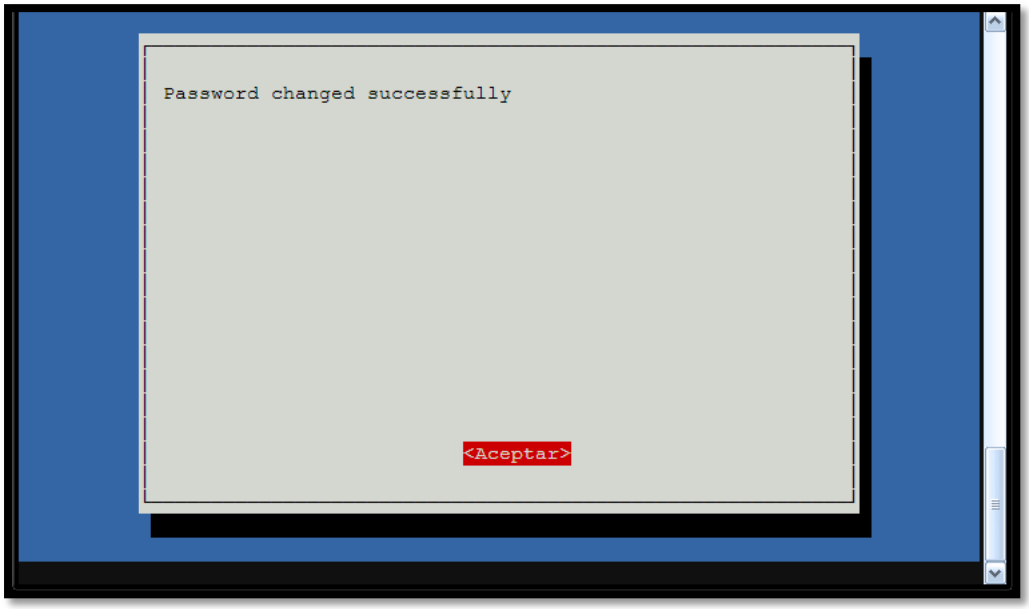


Figura 4. Confirmación contraseña.

- ✓ *Change locale:* donde podremos cambiar la configuración de idioma

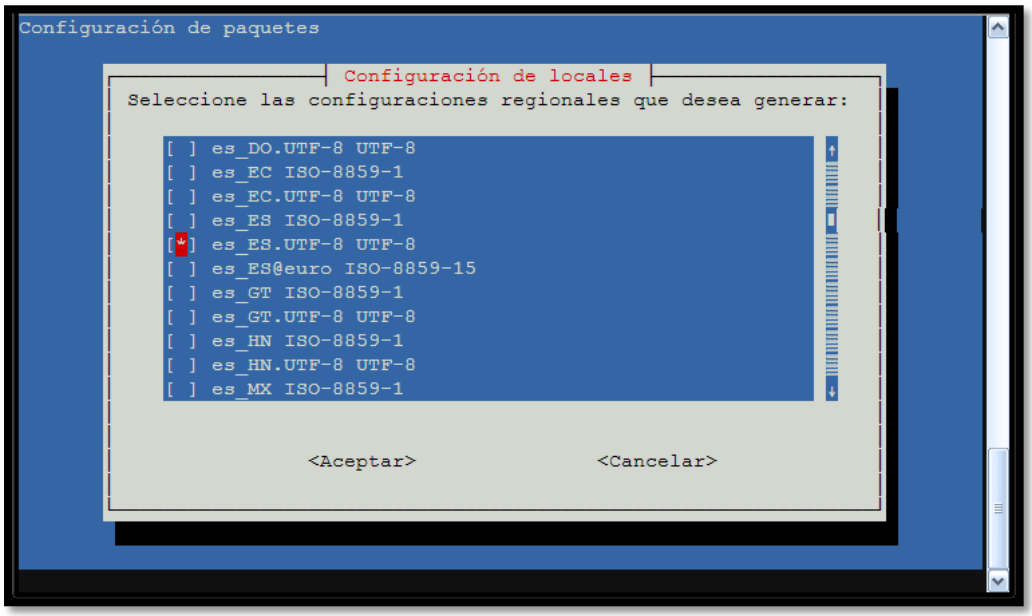


Figura 5. Selección de idioma en Raspbery pi

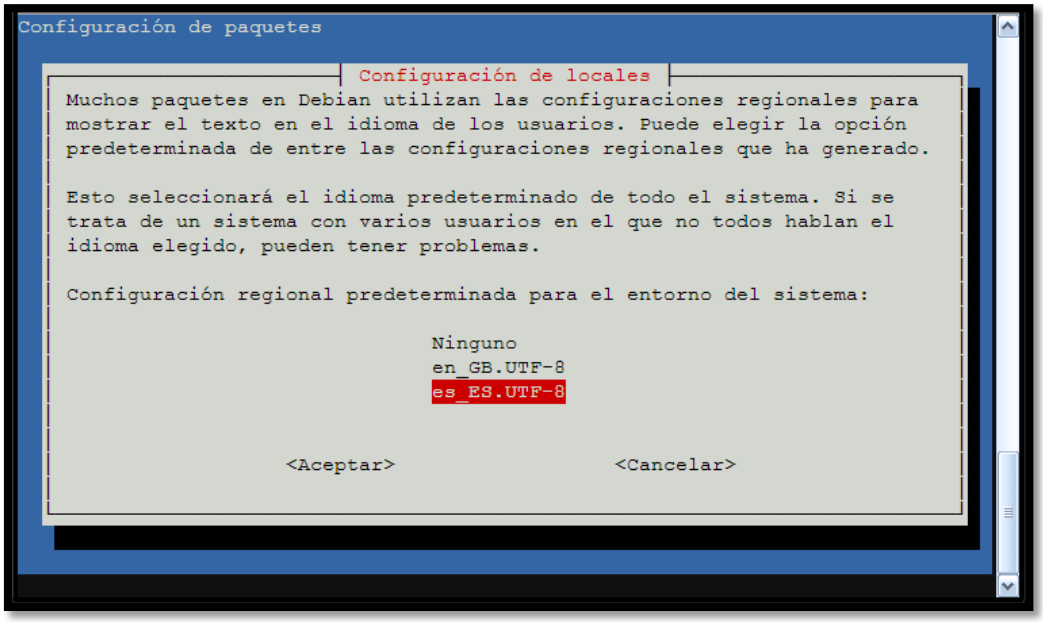


Figura 6. Selección de combinación de teclas en Raspberry pi.

- ✓ *Change_timezone:* donde podremos cambiar la zona horaria, fecha y hora.

- ✓ *Ssh:* para habilitar la conexión por ssh

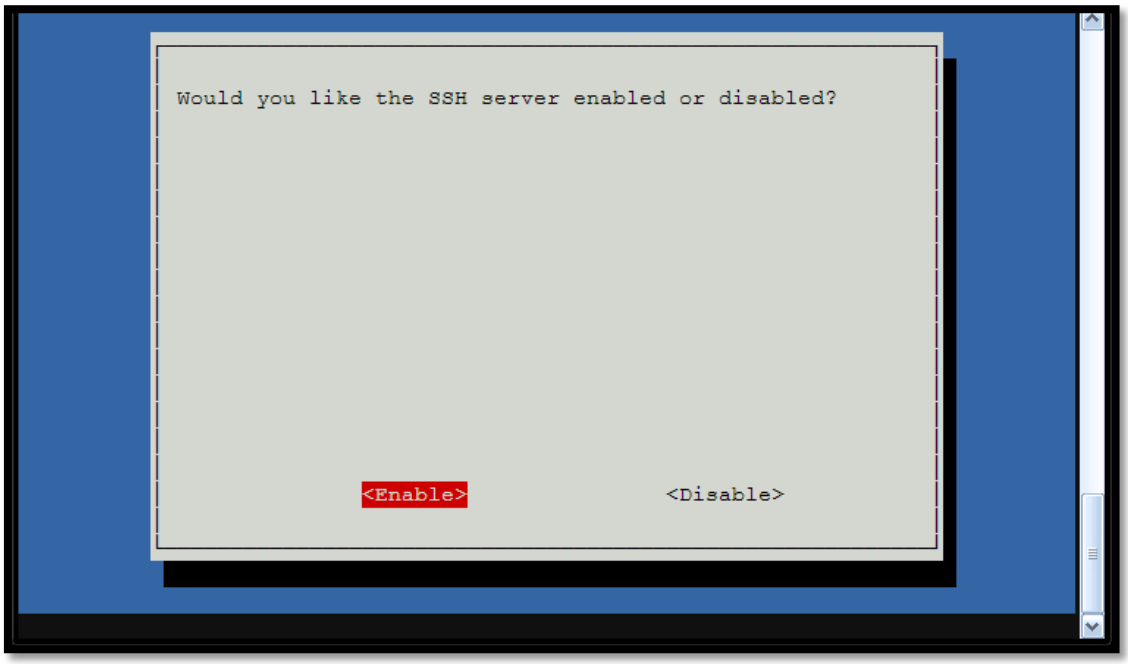


Figura 7. Activación de SSH.

Figura 8. Estado del servidor SSH.

- ✓ *Boot_behaviour:* para elegir el modo de inicio de la Raspberry pi, aquí se pude elegir por ejemplo, iniciar por consola o entorno gráfico.

Figura 9. Configurar inicio gráfico.

- ✓ *Update:* para actualizar la Raspberry pi.

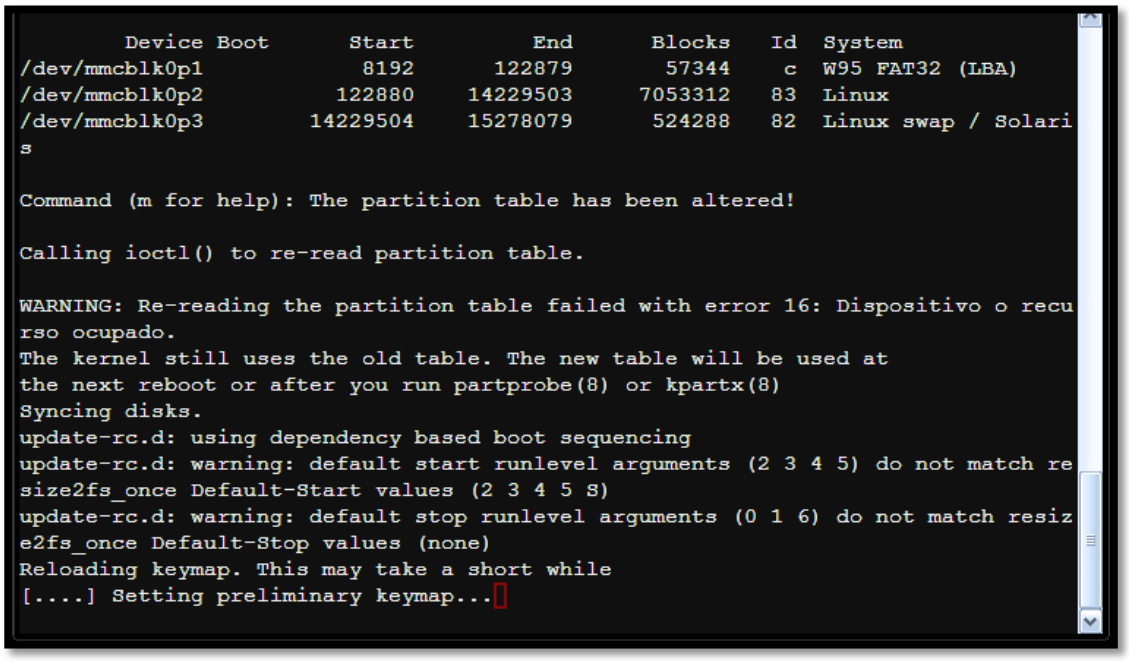


Figura 10. Actualización de Raspberry pi.

Instalaciones necesarias en el sistema:

sudo apt-get install build-essential

sudo apt-get install gcc-arm-linux-gnueabi

sudo apt-get install screen

sudo apt-get install gstreamer0.10-plugins-good

sudo ntpdate pool.ntp.org

sudo apt-get update

sudo apt-get install build-essential python-dev python-setuptools python-pip python-smbus –y

sudo apt-get install openssh-server

sudo apt-get install python

3. Configuración adicional

3.1 Instalación de cámara.

Lo primero de todo será actualizar nuestra distribución Raspbian instalada en la RaspberryPi. Para ello ejecutaremos los siguientes comandos desde una terminal:

sudo apt-get update

sudo apt-get upgrade

```
sudo reboot
```

Una vez actualizada la RaspberryPi, conectaremos nuestra Webcam USB a uno de los dos puertos USB disponibles. Para comprobar que la cámara ha sido detectada, ejecutaremos los siguientes comandos desde una terminal:

```
ls -l /dev/video*
```

```
lsusb
```

Con el último comando obtendremos información de la Webcam que hemos conectado: aparecerá en una línea Bus, Device, ID y nombre de la Webcam que ha sido detectada por el sistema.

Una vez hayamos comprobado que la Webcam ha sido detectada correctamente, procederemos a la instalación de las siguientes librerías que necesita la herramienta MJPG-Streamer y la utilidad subversion que nos servirá más adelante para descargarla. Para ello ejecutamos los siguientes comandos desde una terminal:

```
sudo apt-get install libjpeg8-dev
```

```
sudo apt-get install imagemagick
```

```
sudo apt-get install subversion
```

```
sudo apt-get install gstreamer0.10-plugins-good
```

Instaladas las librerías, el siguiente paso es descargar la herramienta necesaria para realizar el streaming de vídeo que captura nuestra Webcam. Para empezar, crearemos un nuevo directorio para trabajar más cómodamente. Para ello, ejecutamos los siguientes comandos desde una terminal:

```
mkdir mjpg
```

```
cd mjpg
```

Con el segundo comando hemos accedido al directorio que acabamos de crear (mjpg), el cual utilizaremos para alojar la herramienta MJPG-Streamer que vamos a descargar. Para obtener dicha herramienta, situados en el directorio creado anteriormente, lo descargaremos mediante la ejecución del siguiente comando:

```
svn co https://svn.code.sf.net/p/mjpg-streamer/code/mjpg-streamer/ mjpg-streamer
```

Se nos habrá creado un nuevo directorio con el nombre mjpg-streamer . Dentro se encontrarán los archivos necesarios para realizar su compilación, la cual vamos a realizar a continuación ejecutando el siguiente comando:

```
make
```

La compilación generará los archivos necesarios para ejecutar la herramienta MJPG-Streamer. Ya tenemos lista la herramienta para poder ser ejecutada, pero antes vamos a explicar en qué consiste esta herramienta.

¿Qué es MJPG-Streamer?

MJPG-Streamer es una aplicación que se ejecuta por línea de comandos. A grandes rasgos, se encarga de obtener frames JPG (imágenes) capturadas desde una cámara compatible y transmitirlas como M-JPEG (secuencia de vídeo) mediante el protocolo HTTP para poder visualizarlo en navegadores, VLC y otras herramientas.

¿Cómo funciona?

Su funcionamiento se basa en unos plugins de entrada y salida. Es decir, un plugin (de entrada) copia las imágenes JPEG a un directorio de acceso global, mientras que otro plugin (de salida) procesa las imágenes, sirviéndolas como un simple fichero de imagen, o bien, emite las mismas de acuerdo a los estándares MPG existentes.

Por lo tanto, con la compilación anterior, lo que hemos hecho ha sido generar éstos plugins. Vamos a explicar en qué consisten:

mjpg_streamer: Herramienta de línea de comandos, que copia las imágenes JPG de un plugin de entrada, a uno o más plugins de salida.

input_uvc.so: Captura los frames JPG de una Webcam conectada. (A una resolución máxima de 960×720 píxels y un elevado frame rate (>=15fps), con poca carga sobre la CPU).

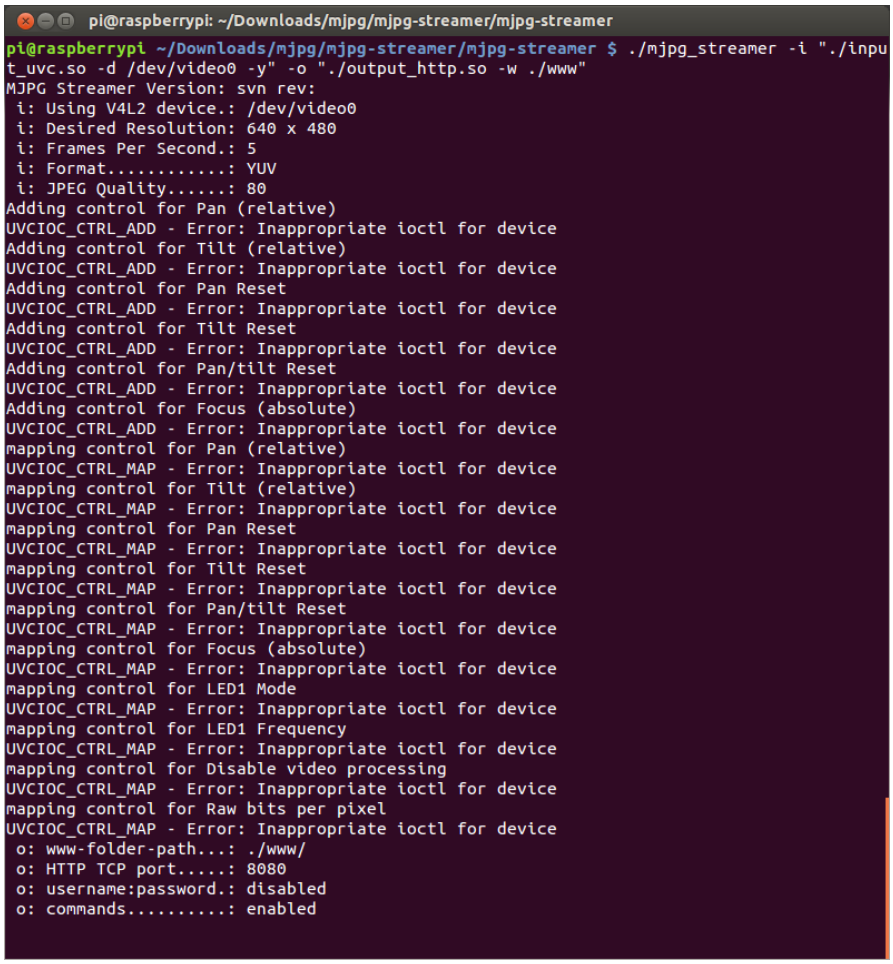
output_http.so: Servidor Web HTTP 1.0. Sirve una única imagen JPG o bien las emite de acuerdo al estándar M-JPEG.

Como se ha explicado anteriormente, MJPG-Streamer funciona bajo línea de comandos, por lo que para iniciarla deberemos ejecutar el siguiente comando:

¡IMPORTANTE! Asegurarse de que la Webcam fue detectada correctamente como se explicó y que está conectada a la Raspberry Pi.

```
./mjpg_streamer -i "./input_uvc.so -d /dev/video0 -y" -o "./output_http.so -w ./www"
```

Ejecutada la herramienta, debería de aparecernos lo siguiente, como se observa en la imagen:



Muestra de mjpg_streamer ejecutandose

Como observaréis, han aparecido una serie de errores, pero no son importantes, ya que la herramienta se está ejecutando correctamente, y para prueba de ello, ¡vamos a acceder a ella!

¿Cómo accedemos a la herramienta para ver nuestro vídeo streaming en directo?

Antes de nada, vamos a detenernos un momento a leer los mensajes que aparecen en la terminal:

MJPG Streamer Version: svn rev:

1: Using V4L2 device.: /dev/video0

2: Desired Resolution: 640 x 480

3: Frames Per Second.: 5

4: Format.....: YUV

5: JPEG Quality.....: 80

6: www-folder-path...: ./www/

7: HTTP TCP port.....: 8080

8: username:password.: disabled

9: commands.....: enabled

Como se puede observar, los mensajes son bastantes descriptivos:

1: Se nos notifica de que se está usando el dispositivo /dev/video0 (¡nuestra Webcam conectada!)

2: La resolución con la que se está emitiendo vídeo: 640×480

3: Se están tomando imágenes a una razón de: 5 FPS

4: El formato de paleta de colores: YUV

5: Calidad de las imágenes JPEG: 80%

6: Directorio de la aplicación Web de la herramienta: ./www

7: Puerto TCP donde se emite el vídeo: 8080

8: Usuario: contraseña (por si se quiere restringir el acceso): disabled

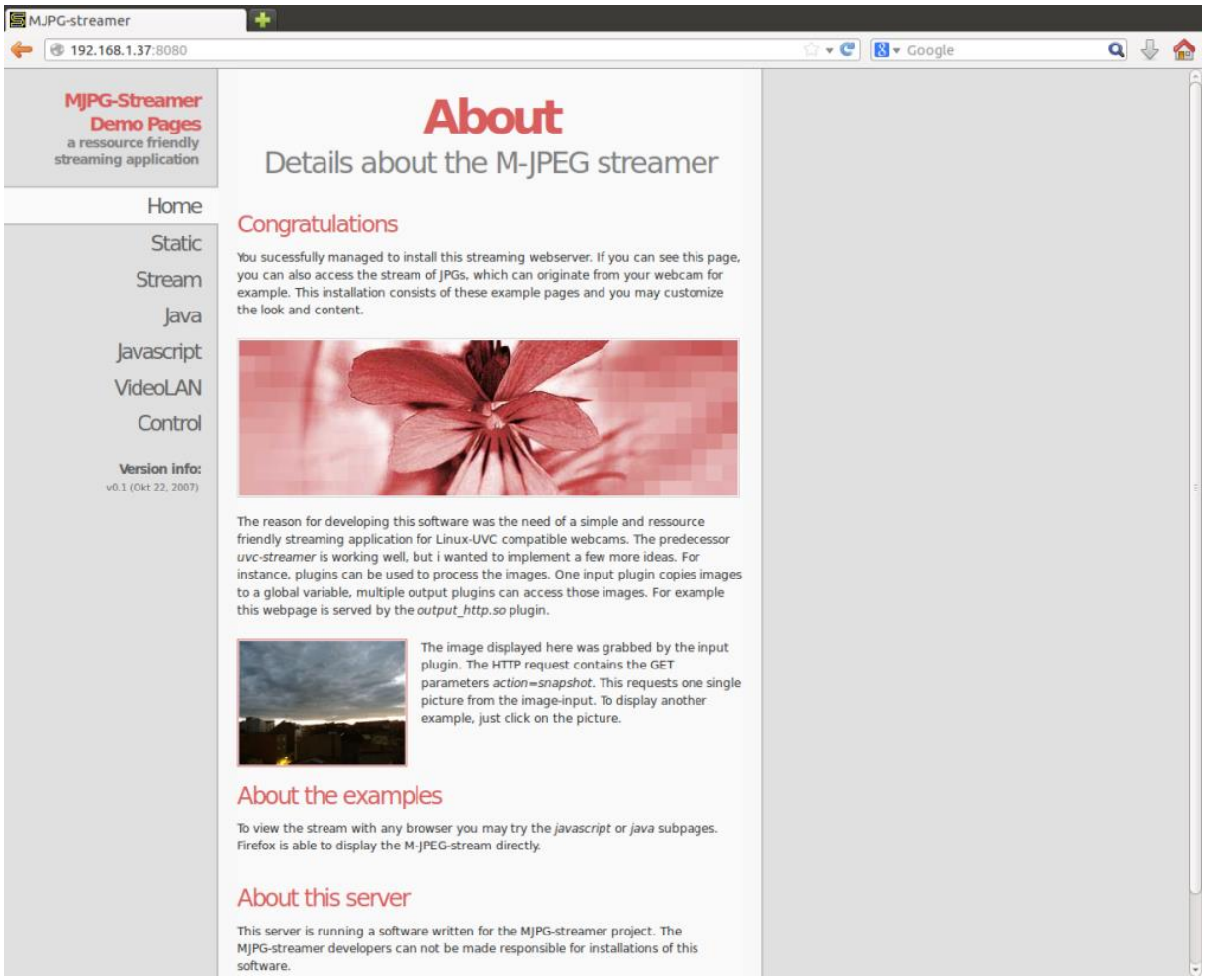
9: Comandos: enabled

De todos ellos nos quedamos con el más importante, el 8. : La aplicación Web de la herramienta se está ejecutando en el puerto 8080.

A continuación abriremos un navegador, y accederemos a la siguiente dirección:

http://192.168.1.35:8080

Hemos accedido a la aplicación web de la herramienta MJPG-Streamer. El aspecto que presenta su dashboard es el que se puede observar en la siguiente imagen:



WEB de mjpg_streamer

3.2 Configuración puente WiFi-Ethernet

Raspberry pi puede ser modificada a nivel software para que exista un *puente* o *bridge* entre el puerto WiFi y el puerto Ethernet. Esto es que con nuestro USB WiFi, tendríamos una salida de Ethernet en la propia Raspberry pi.

Esta idea surgió de la necesidad de darle conectividad a la Beaglebone, pero finalmente se descartó ya que se logró hacer funcionar el usbWiFi y además este método no aseguraba una gran fiabilidad en él sistema. No obstante detallamos como llevarlo a cabo, por si en un futuro fuese necesario recurrir a este método para conectar algún otro sistema o dispositivo.

Para ello vamos a configurar la placa de la siguiente manera. Lo primero que haremos es crear un perfil de red inalámbrica, para ello vamos al archivo *wpa_supplicant.conf*.

```
sudo nano/etc/wpa_supplicant/wpa_supplicant.conf
```

Aquí ponemos nuestro punto de acceso:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="Your SSID Here"
    proto=RSN
    key_mgmt=WPA-PSK
    pairwise=CCMP TKIP
    group=CCMP TKIP
    psk="YourPresharedKeyHere"
}
```

Posteriormente, guardamos y accedemos al archivo *interfaces*:

sudo nano /etc/network/interfaces

Y editamos el archivo para que aparezca lo siguiente:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

auto wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

A continuación, testeamos que se conecta al punto de acceso:

sudo ifdown wlan0

sudo ifup wlan0

Ahora es el momento en el que creamos el *puente:*

sudo apt-get install bridge-utils

sudo brctl addbr br0

sudo brctl addif br0 wlan0 eth0

Cuando acaben estos tres comandos, modificamos de nuevo el archive interfaces.

```
auto lo
iface lo inet loopback

iface eth0 inet manual

auto wlan0
iface wlan0 inet manual

auto br0
iface br0 inet dhcp
   bridge_ports wlan0 eth0
   bridge_stp off
   bridge_maxwait 5

wpa-iface wlan0
wpa-bridge br0
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Una vez hecha esta modificación, guardamos, cerramos y reiniciamos. Una vez iniciado de nuevo el sistema, debemos tener configurado nuestro puente WiFi correctamente.

4. Referencias

[1] Raspberry pi. Página oficial. http://raspberrypi.org

[2] Raspberry pi. Cámara. http://geekytheory.com/video-streaming-live-con-raspberrypi-y-playstation-eye/

[3] Raspberry pi. Puente WiFi. http://blog.slor.net/2013/09/turning-your-raspberri-pi-into-wireless.html

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 5. Instalación de ROS en PC con Ubuntu

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción 4

2. Instalación de ROS 4

3. Referencias 9

1. Introducción

Vamos a proceder a instalar ROS en Linux, en nuestro caso se ha escogido Ubuntu 12.04 ya que es la última actualización estable disponible en el momento de realizar este documento.

2. Instalación de ROS.

Primero comprobamos que tengamos actualizado nuestro sistema, en esta versión podemos encontrar en la parte izquierda de la pantalla principal el programa llamado "updatemanager".

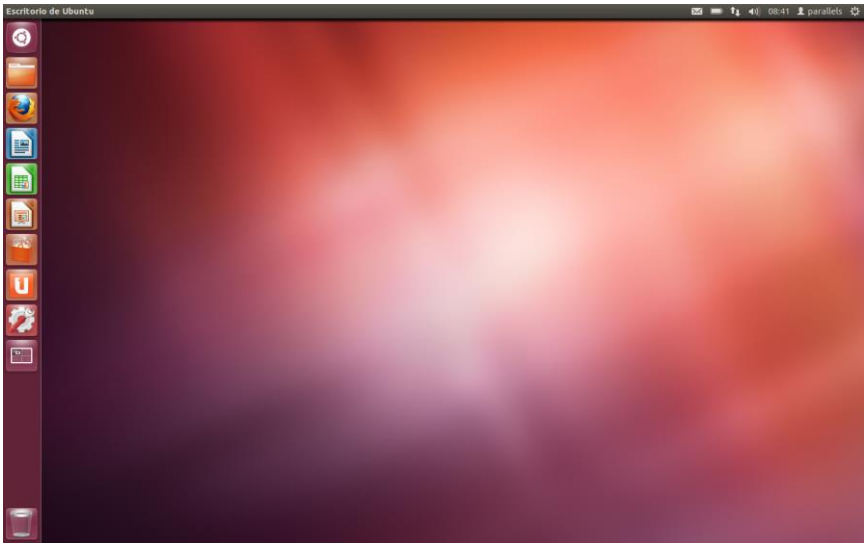


Figura 1. Vista principal de Ubuntu

Para comenzar abrimos la utilidad de Terminal:

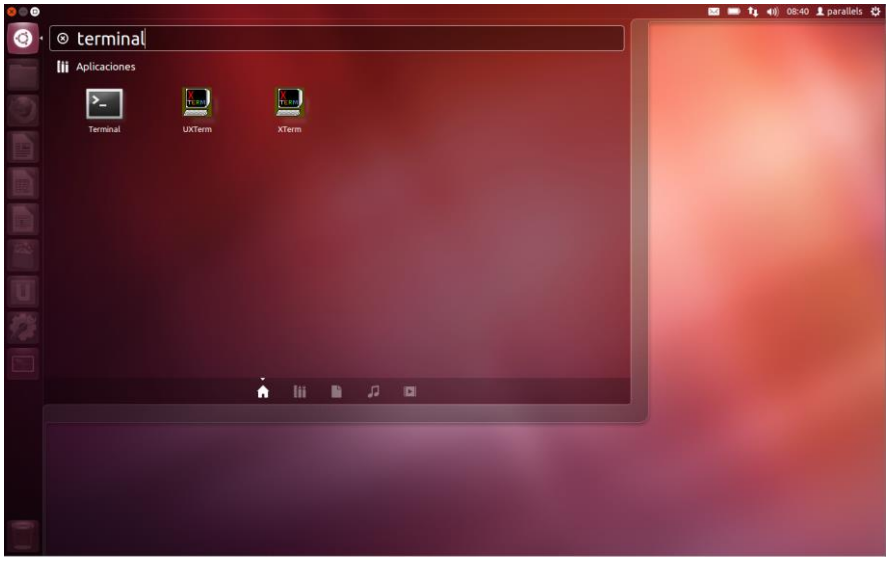


Figura 2. Vista de acceso a terminal

A continuación preparamos el sistema para que acepte los paquetes de información de la página del repositorio packages.ros.org, para ello usamos la función correspondiente a nuestro sistema.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main" > /etc/apt/sources.list.d/ros-latest.list'
```

Tras introducir la función anterior, nos pedirá la contraseña de usuario para confirmar que queremos aceptar esta función, y terminal volverá a modo espera.

Figura 3. Ventana de terminal tras preparar el sistema

Lo siguiente es descargarte el fichero correspondiente a tu teclado, para ello usamos la función:

```
wget https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -O - | sudo apt-key add -
```

Figura 4. Ventana de terminal tras instalar el teclado ROS

Una vez preparado nuestro sistema para instalar ROS procedemos a la instalación del sistema operativo, para ello usamos la función que nos descargará el sistema por completo (en el caso de tener algún tipo de problema con la instalación de alguna fase se puede instalar uno por uno todos los componentes necesarios).

Primero comprobamos que esté todo actualizado con la función:

```
sudo apt-get update
```

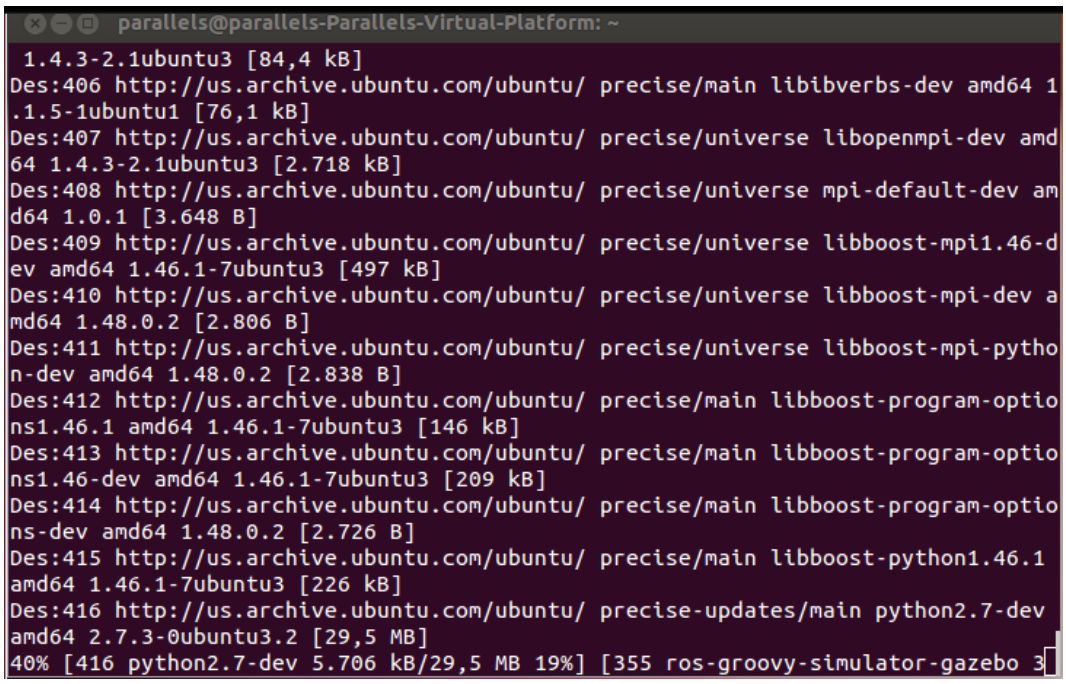


Figura 5. Ventana de ejemplo del uso de update

Y luego usamos:

```
sudo apt-get install ros-hydro-desktop-full
```

Y nada más pulsar la tecla *intro* el programa se pondrá a funcionar descargando todo lo necesario para que el sistema funcione.

Durante la instalación podremos configurar varios aspectos que tendrá nuestro sistema, como por ejemplo si solo va a trabajar en modo local, o se podrá utilizar también en remoto.

Cuando termine de descargar e instalar todos los paquetes deberemos ver algo parecido a esto:

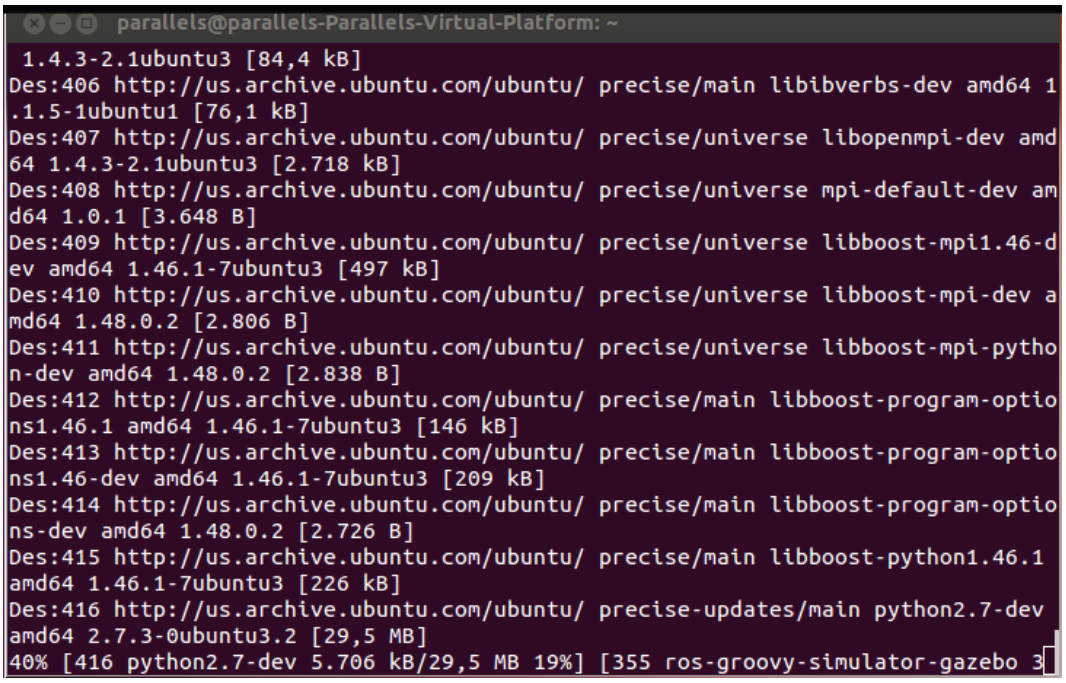


Figura 6. Ventana de terminal tras instalar ROS

A continuación iniciamos el programa rosdep y comprobamos que esté actualizado, para ello usamos primero la función:

```
sudo rosdep init
```

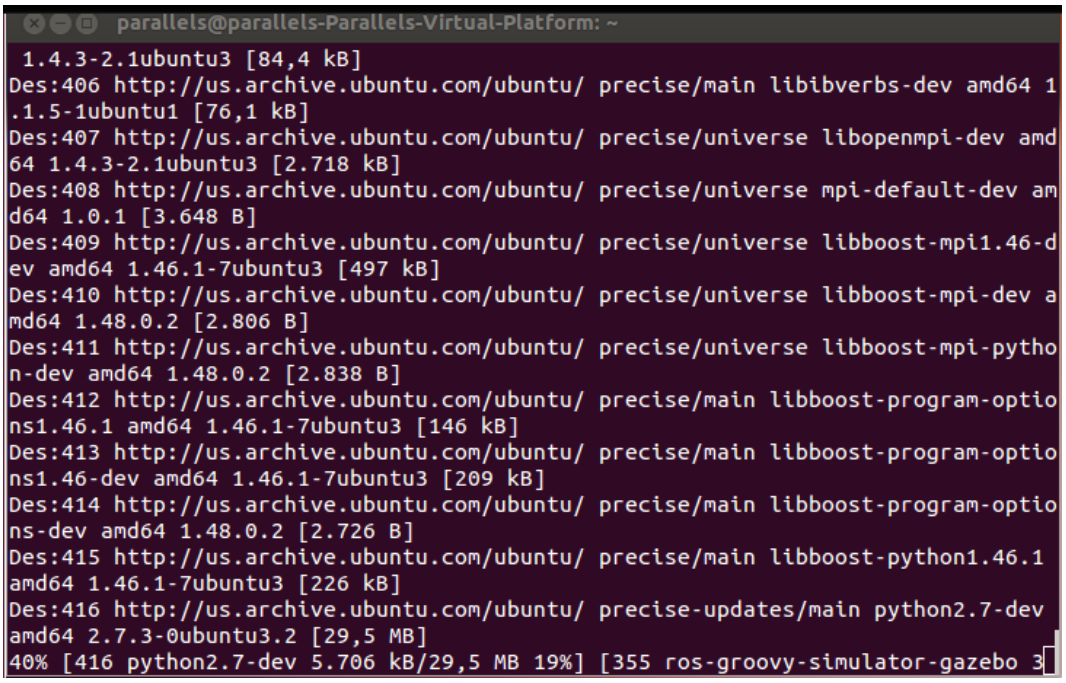


Figura 7. Ventana de terminal tras inicializar rosdep

Y luego la función:

```
rosdep update
```

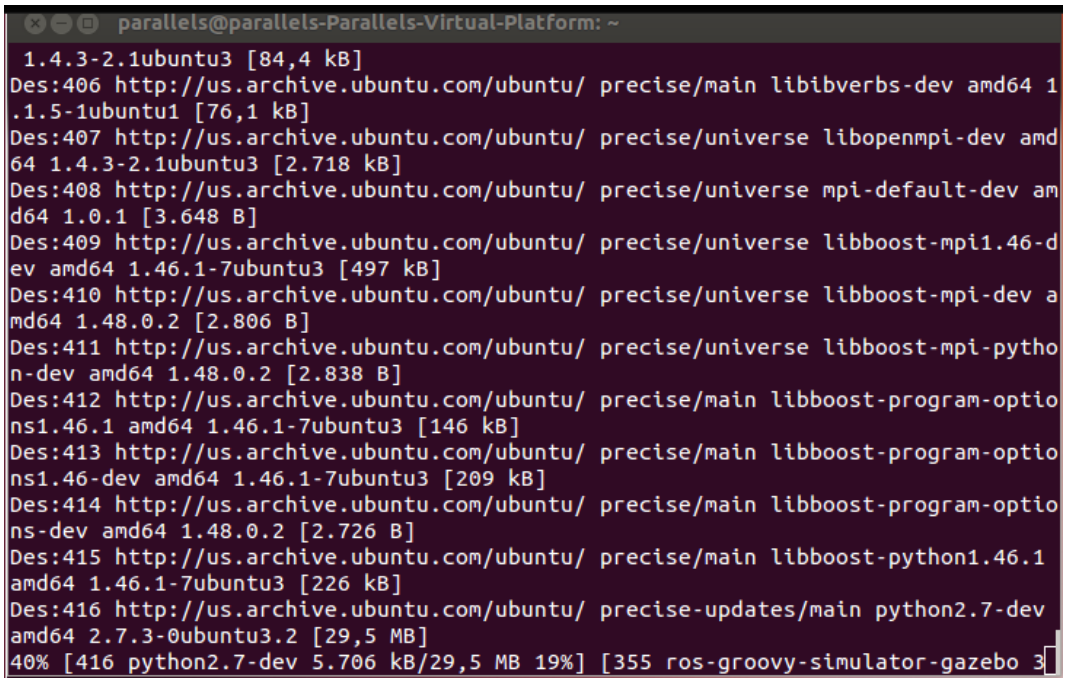


Figura 8. Ventana de terminal tras actualizar rosdep

Ya tenemos ROS instalado en nuestro sistema Ubuntu, y ahora solo nos quedaría añadir un par de configuraciones más que son recomendables.

La primera es para que las variables de entorno que creemos se añadan automáticamente a nuestra sesión, para ello usamos la función:

```
echo "source /opt/ros/hydro/setup.bash" >> ~/.bashrc
```

Y luego:

```
source ~/.bashrc
```

También es conveniente instalar rosinstall, que es un añadido que nos permitirá descargarnos fácilmente el código fuente de muchos añadidos con solo un comando, para ello usamos el comando:

```
sudo apt-get install python-rosinstall
```

Tras esto ya tendremos nuestro sistema listo para trabajar con él.

```
roscore
```

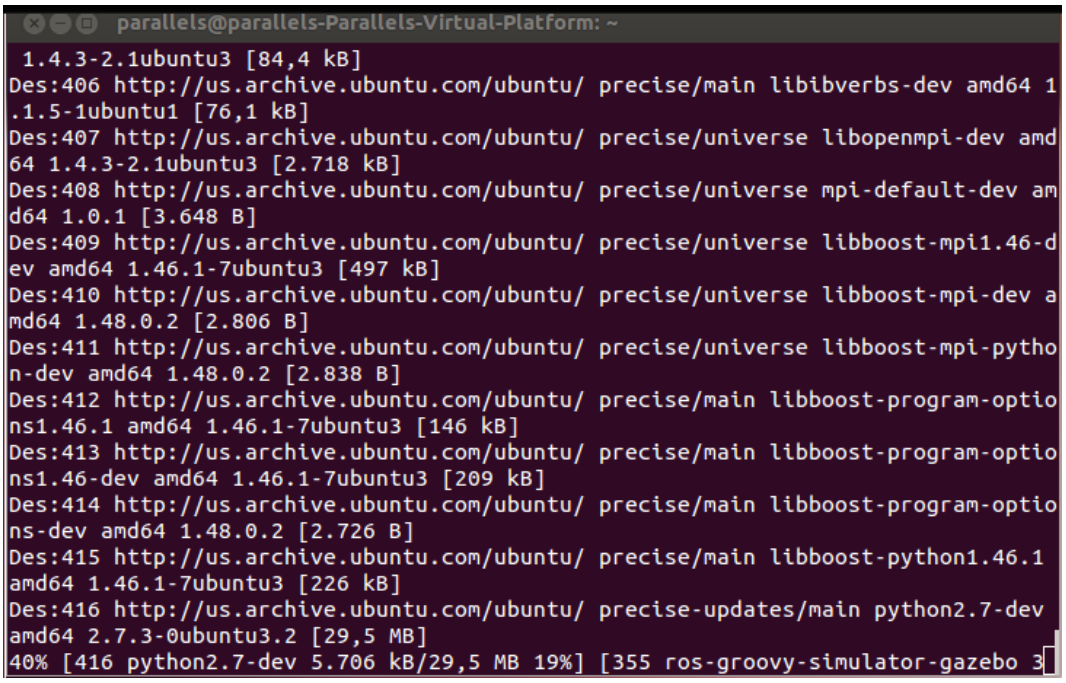


Figura 9. Muestra de roscore ejecutandose

3. Referencias

ROS. Ubuntu: http://wiki.ros.org/hydro/Installation/Ubuntu

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 6. Instalación de ROS en Beaglebone con Ubuntu

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción 4

2. Instalación de ROS 4

3. Referencias 8

1. Introducción

La versión de ROS para ubuntu armhf, se encuentra en fase experimental, pero para nuestro proyecto encontramos todos los paquetes necesarios.

2. Instalación de ROS.

Lo primero es descargarse el fichero correspondiente a tu teclado, para ello usamos la función:

```
sudo update-locale LANG=C LANGUAGE=C LC_ALL=C LC_MESSAGES=POSIX
```

Figura 1. Ventana de terminal tras instalar teclado

Luego configuramos el sistema para que acepte la descarga del repositorio:

```
sudo sh -c 'echo "deb http://packages.namniart.com/repos/ros quantal main" > /etc/apt/sources.list.d/ros-latest.list'
```

Figura 2. Ventana tras preparar el sistema

Configuramos las claves:

```
wget http://packages.namniart.com/repos/namniart.key -O - | sudo apt-key add –
```

Figura 3. Ventana de terminal tras instalar el teclado de ROS

Actualizamos los paquetes del sistema:

```
sudo apt-get update
```

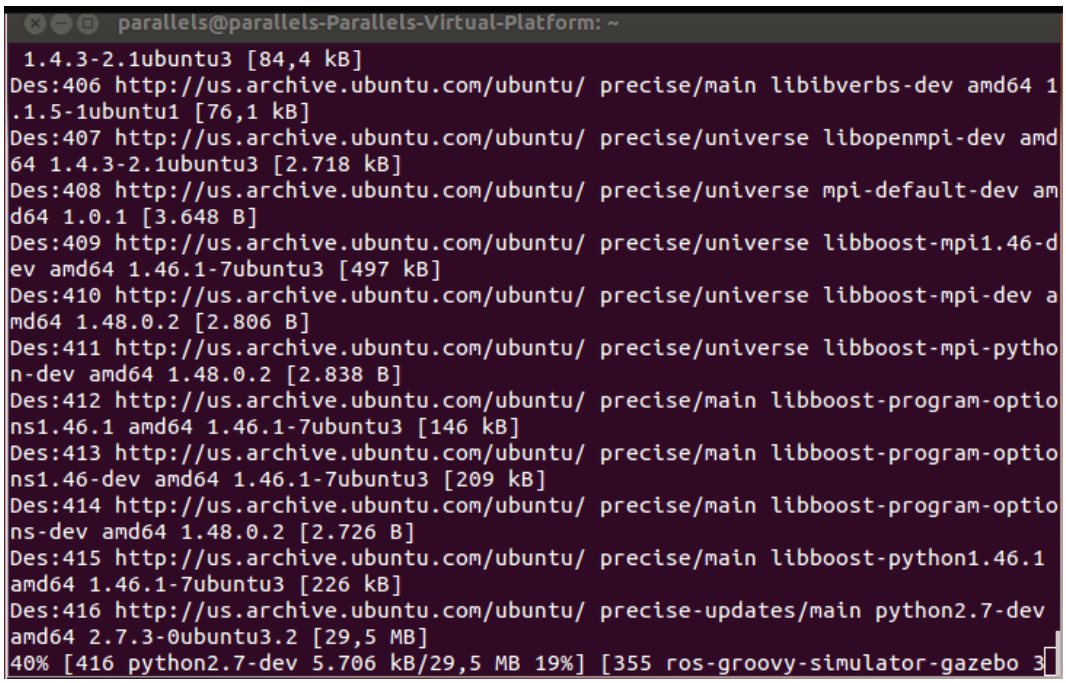


Figura 4. Ventana de ejemplo del uso de update

Instalamos ROS:

```
sudo apt-get install ros-hydro-ros-base
```

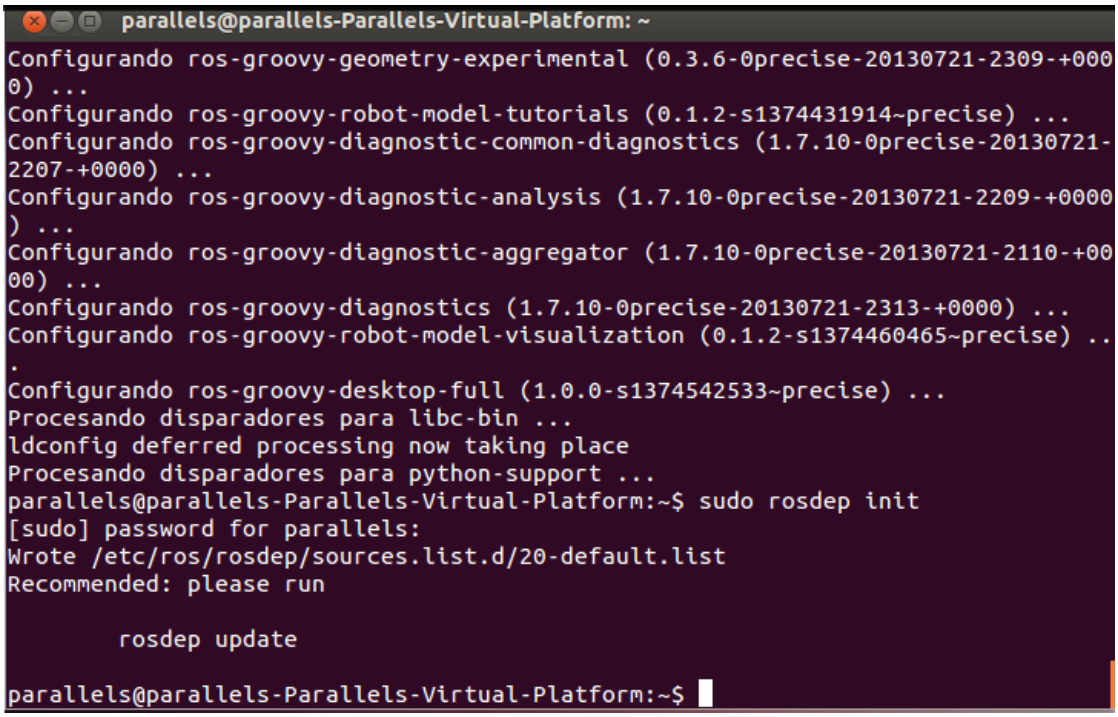


Figura 5. Ventana de terminal tras la instalación de ROS

Instalamos rosdep:

```
sudo apt-get install python-rosdep
```

Actualizamos rosdep:

```
sudo rosdep init
```

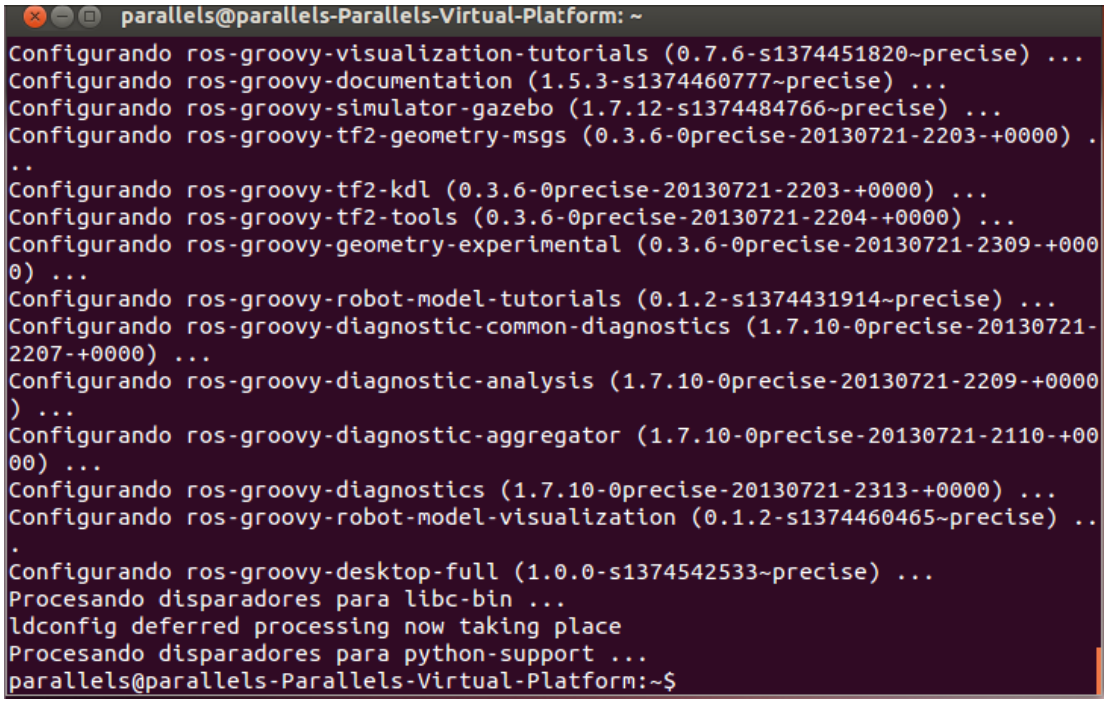


Figura 6. Ventana de terminal tras inicializar rosdep

```
rosdep update
```

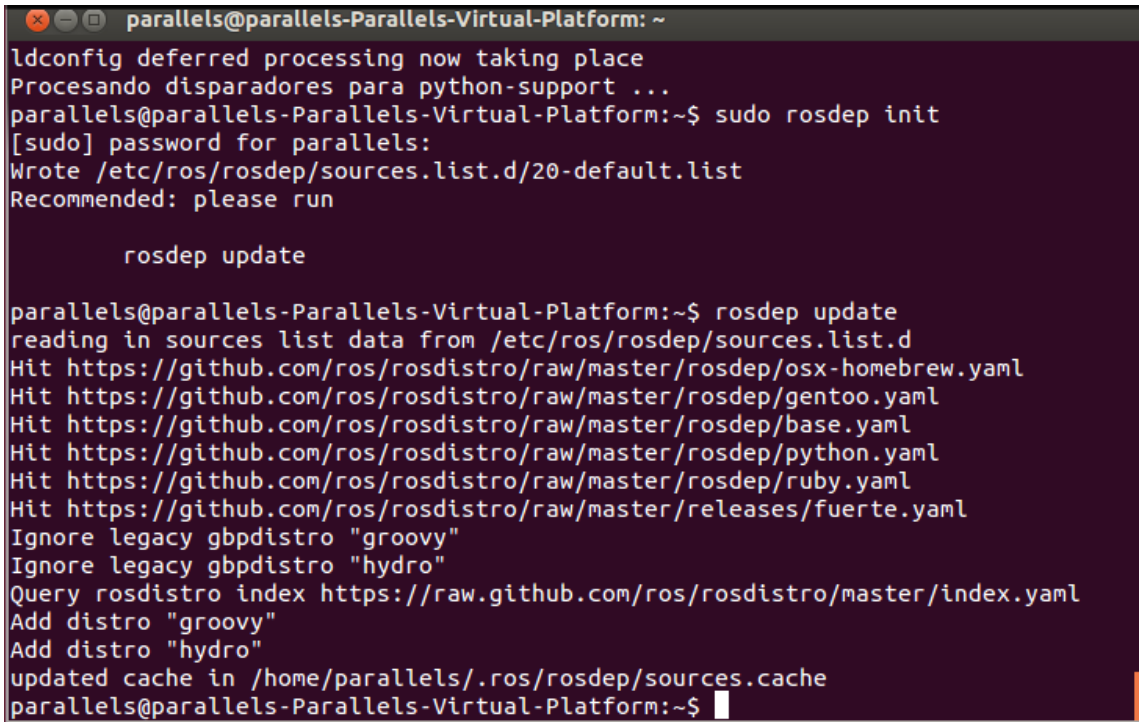


Figura 7. Ventana de terminal tras iactualizar rosdep

Añadimos variables de entorno:

```
echo "source /opt/ros/hydro/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Instalamos rosinstall:

```
sudo apt-get install python-rosinstall
```

Y finalmente ejecutamos ROS:

```
Roscore
```

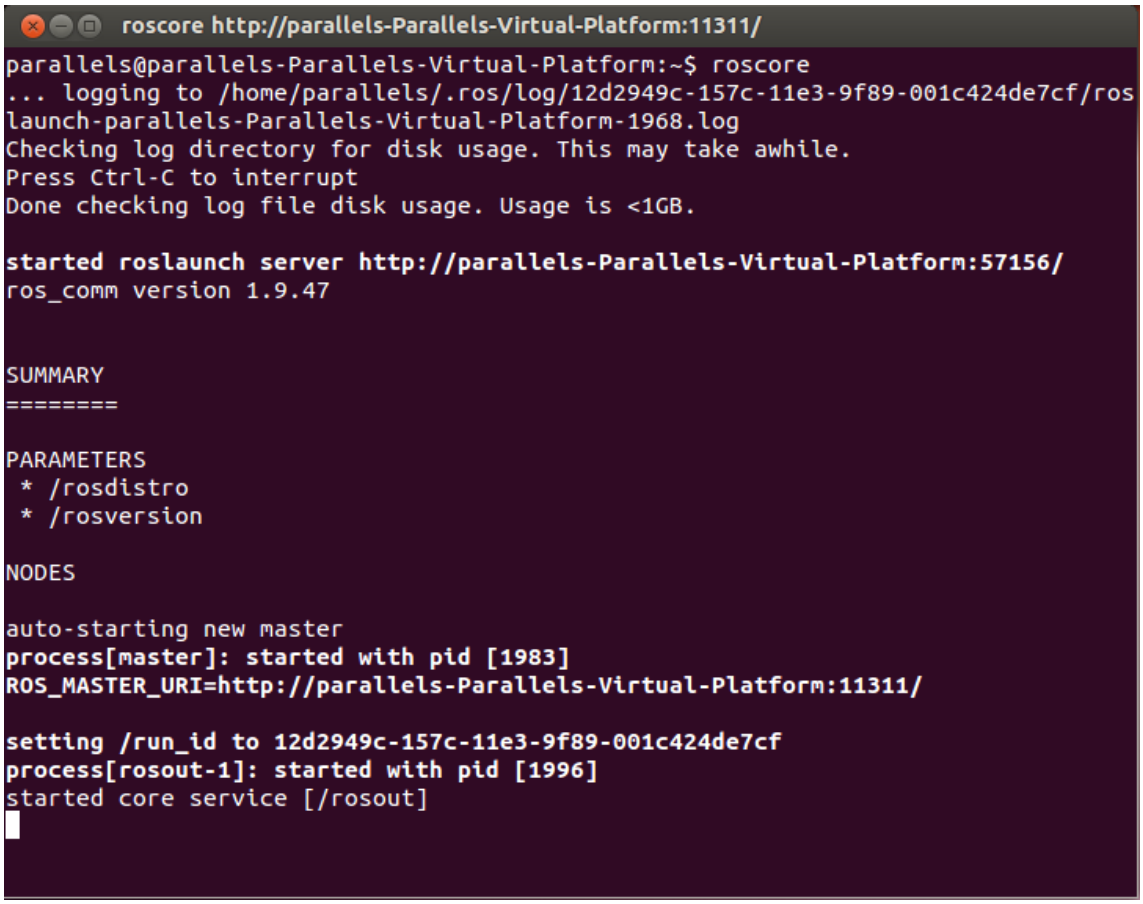


Figura 8. Muestra de roscore ejecutandose

3. Referencias

ROS. Ubuntu. Beaglebone: http://wiki.ros.org/hydro/Installation/UbuntuARM

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 7. Instalación de ROS en Beaglebone con Angstrom

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción 4

2. Instalación de ROS 4

3. Referencias 9

1. Introducción

ROS Hydro medusa se puede utilizar en la Beaglebone junto con Ångström gracias a la acreditación a través del proyecto de Beagle-ROS, el cual tiene como objetivo integrar el Sistema Operativo Robot (ROS) y la BEAGLEBONE a través de la meta-ros capa.

2. Instalación de ROS.

Lo primero que tenemos que hacer para instalar ROS es actualizar el Ångström:

```
opkg update
```

Luego instalar la herramienta de repositorios git:

```
opkg install git
```

Descargar el beagle-ros código:

```
git clone git://github.com/vmayoral/beagle-ros.git
```

Instalar todos los paquetes de ROS en Ångström:

```
cd beagle-ros/scripts
sh minimal-ros-install-angstrom.sh
```

Y finalmente ejecutar ROS:

```
roscore
```

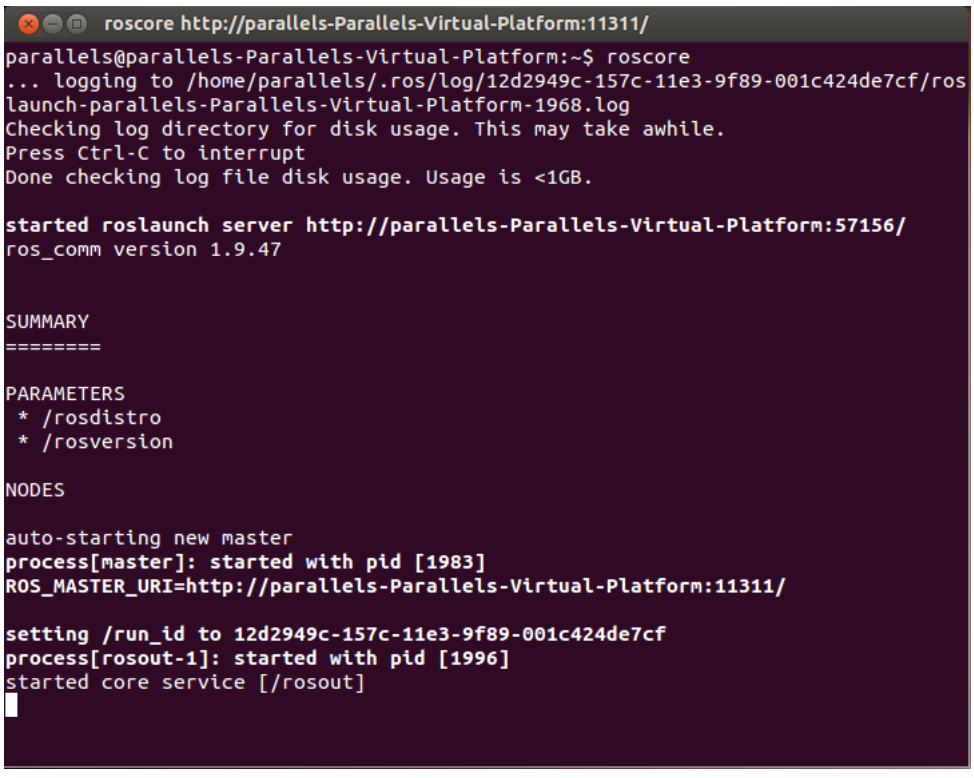


Figura 1. Muestra de roscore ejecutandose

3. Referencias

ROS. Angstrom. Beaglebone: http://wiki.ros.org/BeagleBone

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 8. Instalación de ROS en Raspberry pi con Raspbian

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción 4

2. Instalación de ROS 4

3. Referencias 9

1. Introducción

Se trata de un repositorio experimental de ROS Groovy (versión anterior a ROS Hydro), pero con los paquetes necesarios para nuestro sistema:

2. Instalación de ROS.

Lo primero es configurar el sistema para que acepte el repositorio:

```
sudo sh -c 'echo "deb http://64.91.227.57/repos/rospbian wheezy main" > /etc/apt/sources.list.d/rospbian.list'
```

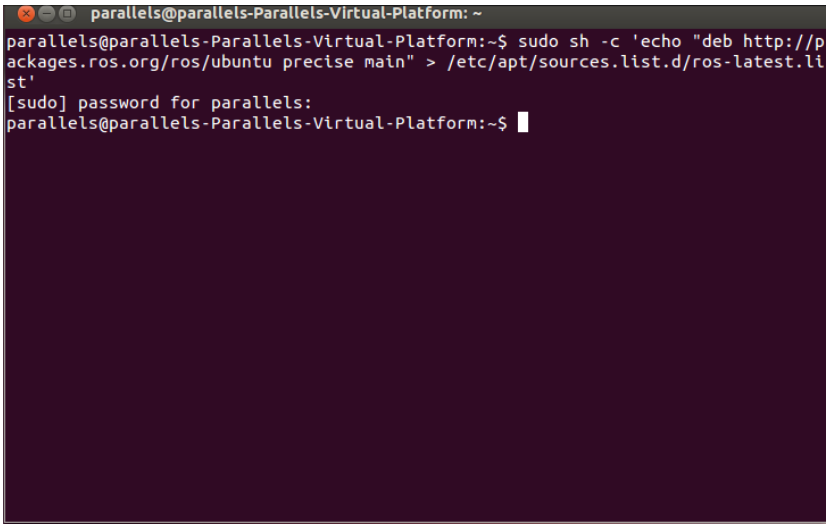


Figura 1. Ventana de terminal tras preparar el sistema

Configuramos las claves:

```
wget http://64.91.227.57/repos/rospbian.key -O - | sudo apt-key add –
```

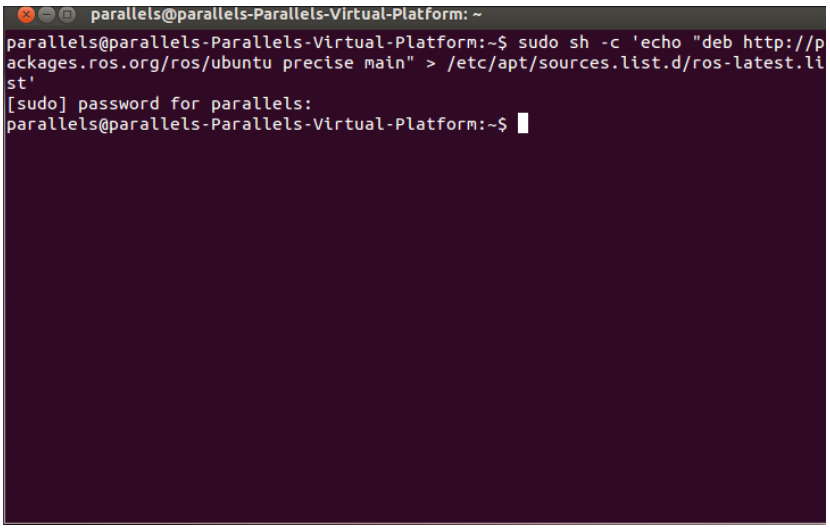


Figura 2. Ventana de terminal tras instalar el teclado de ROS

Actualizamos los paquetes del sistema:

```
sudo apt-get update
```

Figura 3. Ventana de ejemplo del uso de update

Instalamos ROS:

```
sudo apt-get install ros-groovy-ros-comm
```

Añadimos variables de entorno:

```
echo "source /opt/ros/groovy/setup.bash" >> ~/.bashrc
. ~/.bashrc
```

Y finalmente ejecutamos ROS:

```
Roscore
```

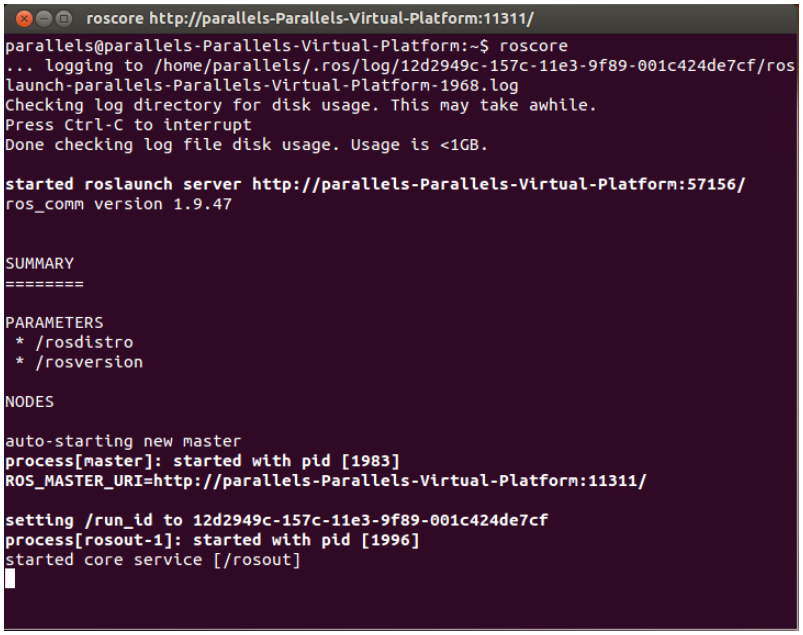


Figura 4. Muestra del roscore ejecutandose

3. Referencias

ROS. Raspbian. Raspberry pi: http://wiki.ros.org/groovy/Installation/Raspbian

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 9. ROS. Conceptos básicos

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Conceptos básicos 4

2. Referencias 9

1. Conceptos básicos

Estructura interna de ROS:

Para entender cómo se organizan los paquetes de nuestras aplicaciones, será conveniente explicar los tipos de carpetas y archivos que podemos encontrarnos normalmente, sin entrar en detalles de cada uno, en un paquete de ROS:

Carpeta bin: contiene los ejecutables del paquete, los que se podrán lanzar como nodos.

Carpeta build: contiene los archivos de compilación internos del paquete.

Carpeta src: contiene los códigos de los programas (.cpp), que al compilarlos crearán ejecutables en bin.

Carpeta include: contiene los archivos de cabeceras (.h) propios de los programas del paquete.

Carpeta lib: contiene los archivos de librerías (.lib, .so).

Carpeta launch: contiene los archivos de lanzamiento (.launch) de aplicaciones completas.

Carpeta yaml: contiene los archivos de lista de parámetros (.yaml).

Carpeta msg: contiene los tipos de mensajes (.msg) definidos en el paquete.

Carpeta msg_gen: contiene los archivos de cabeceras (.h) auto-generadas al compilar los tipos de mensajes definidos en el paquete.

Carpeta srv: contiene los tipos de mensajes de los servicios definidos en el paquete.

Carpeta srv_gen: contiene los archivos de cabeceras (.h) auto-generadas al compilar los tipos de servicios definidos en el paquete.

Archivo CMakeLists.txt: es una lista en la que se especifica al compilador que debe compilar de nuestro paquete: nodos, mensajes, servicios, librerías, etc.

Archivo manifest.xml: contiene una descripción del paquete (función, autor, licencia, etc.) y una lista con los paquetes de los que depende.

Según la funcionalidad del paquete, es posible encontrarnos más o menos carpetas con archivos diferentes.

Para compilar un paquete en ROS se utiliza CMake, una plataforma de código abierto para la generación, compilación y comprobación de paquetes de software. Su uso es bastante sencillo, basta con que el paquete de ROS contenga un archivo llamado CMakeLists.txt compuesto por una serie de macros según lo que se quiera crear y compilar.

Obviando aquellas que por defecto aparecen para lanzar CMake y configurar la ubicación predeterminada para las salidas, las macros más comunes que se utilizan son las siguientes:

rosbuild_add_executable(ejecutable src/programa.cpp): crea en la carpeta bin el ejecutable de programa.cpp descrito en src.

rosbuild_add_library(librería src/programa.cpp): crea en la carpeta lib la librería de programa.cpp descrito en src.

rosbuild_genmsg(): auto-genera todos las cabeceras y archivos necesarios de los tipos de mensaje definidos en la carpeta msg del paquete y nos los guardará en una nueva carpeta denominada msg_gen.

rosbuild_gensrv():auto-genera todos las cabeceras y archivos necesarios de los tipos de servicio definidos en la carpeta srv del paquete y nos los guardará en una nueva carpeta denominada srv_gen.

Para compilar un paquete, basta con hacer en una terminal:

rosmake nombre_paquete

Esto compilará todo lo especificado en CMakeLists.txt, además de todos los paquetes de los que depende éste, es decir, aquellos especificados enmanifest.xml de la forma:

<depend package=“nombre_paquete”>

Como en cualquier otro programa, en nuestro sistema necesitaremos un lugar donde trabajar, en este caso se requiere de un área para crear nuestras pilas y paquetes, y su posterior modificación en caso de ser necesario.

Para crear un nuevo espacio de trabajo utilizaremos el comando catkin_make, que se utiliza de la forma:

Primero creamos nuestro espacio de trabajo:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
catkin_init_workspace
```

A pesar de que el espacio de trabajo creado no contenga ningún CmakeList.txt, se puede construir igualmente el espacio de trabajo ROS, usando:

```
cd ~/catkin_ws/
catkin_make
```

Lo único que habría que hacer ahora es añadir a las variables de entorno esta ruta:

```
export ROS_PACKAGE_PATH=~/catkin_ws:$ROS_PACKAGE_PATH
echo $ROS_PACKAGE_PATH
```

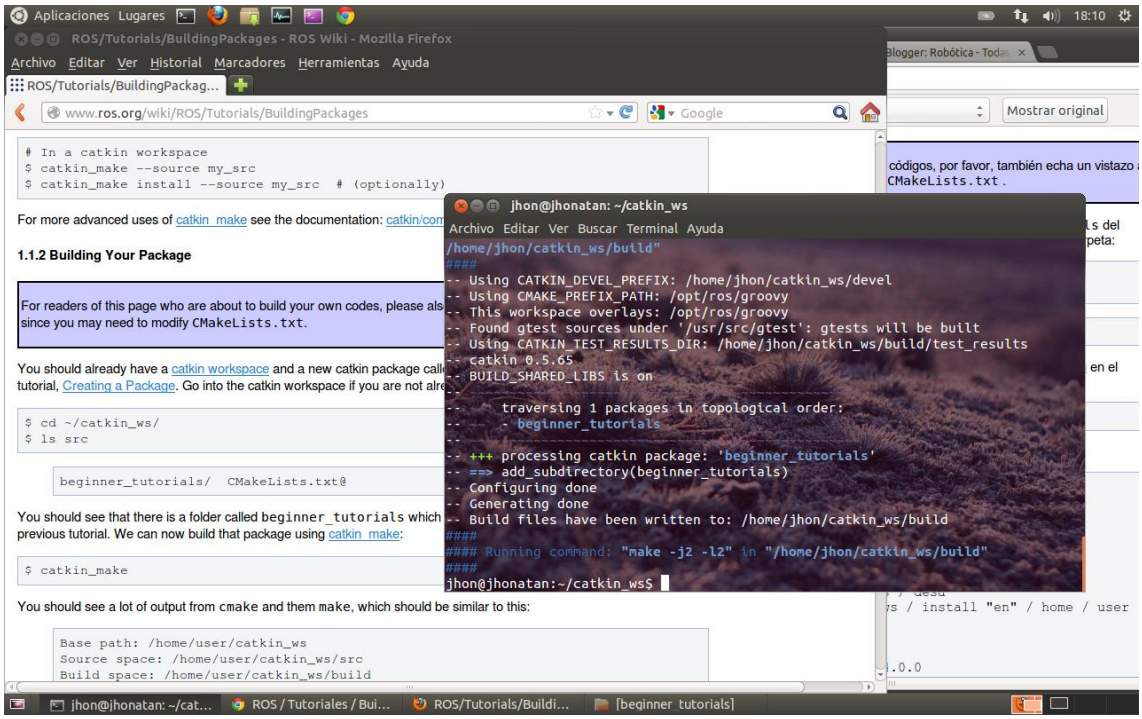


Figura 1. Muestra de cómo crea la carpeta en nuestro sistema de ficheros

Rviz:

ROS dispone de una herramienta de visualización en 3D llamada RVIZ que posibilita que nuestro robot Qbo, o prácticamente cualquier otra plataforma robótica, pueda ser representado en imagen 3D, respondiendo en tiempo real a lo que le ocurre en el mundo real.

RVIZ se puede usar para mostrar lecturas de sensores, datos devueltos por la visión estereoscópica (Cloud Point), hacer SLAM (localización y mapeo simultáneo) evitando obstáculos, etc. Esta herramienta dispone así mismo, de muchísimas opciones de configuración.

Para generar cada modelo de robot en particular, se tiene que editar en un archivo XML y escribir en URDF (Unified Robot Description Format) donde se especifican las dimensiones del robot, los movimientos de las articulaciones, parámetros físicos como masa e inercia, etc. En el caso de robots con muchas articulaciones ROS dispone de otra herramienta llamada TF que es una biblioteca que facilita la elaboración en estos casos.

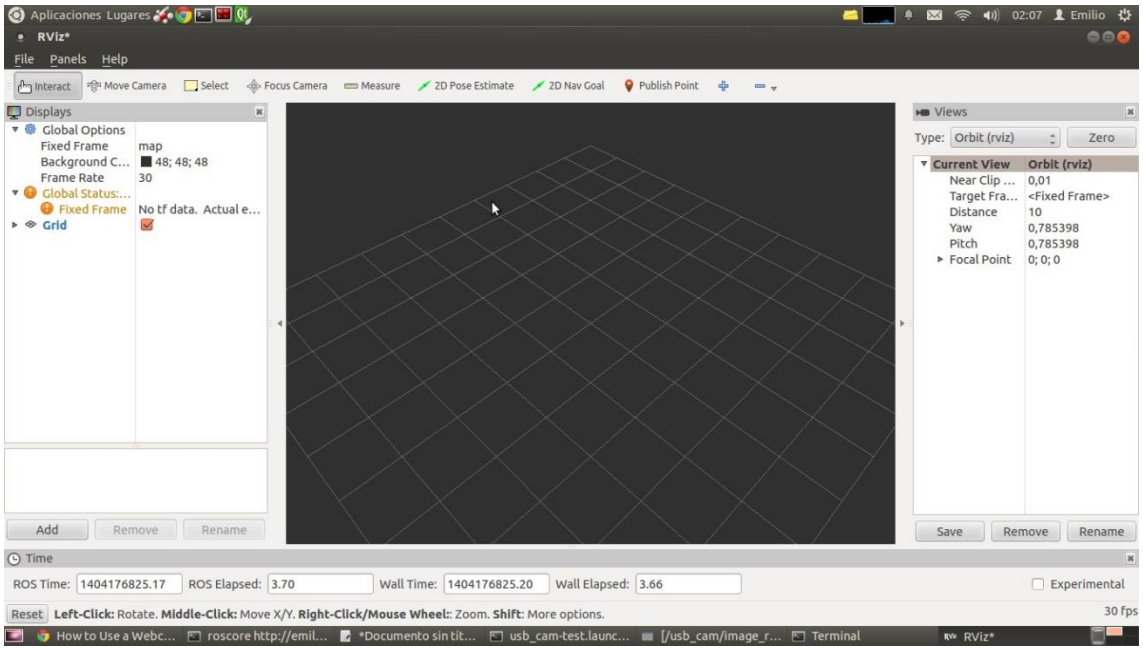


Figura 2. Entorno Rviz

2. Referencias

ROS: http://wiki.ros.org/

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 10. Nodos en ROS

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción. 4

2. Configuración maestro-esclavo. 4

3. Referencias. 16

1. Introducción

Nuestro sistema ROS se basa en un nodo de cámara en la Raspberry pi, un nodo de comunicación de red entre el maestro y los dos esclavos y un nodo serial entre la Beaglebone y las dos Mbed.

En futuras versiones del proyecto se implementara un nodo RFID.

2. Nodos.

El nodo de cámara, a usar en la Raspberry pi, es específicamente para el modelo de cámara Raspicam. El cual funciona a 90 fps, para poder usarlo tenemos que:

Primero actualizar el sistema:

```
sudo apt-get update
sudo apt-get upgrade
```

Descargar e instalar librerías userland:

```
sudo apt-get install git-core gcc build-essential cmake vlc
mkdir ~/code
cd ~/code
git clone git://github.com/raspberrypi/userland.git
cd ~/code/userland
sed -i 's/if (DEFINED CMAKE_TOOLCHAIN_FILE)/if (NOT DEFINED
CMAKE_TOOLCHAIN_FILE)/g' makefiles/cmake/arm-linux.cmake
mkdir ~/code/userland/build
cd ~/code/userland/build
sudo cmake -DCMAKE_BUILD_TYPE=Release ..
sudo make
sudo make install
```

Ir a nuestro espacio de trabajo concretamente:

```
cd catkin_ws
```

Si no tenemos las variables de entorno configuradas, lanzar el directorio de trabajo ROS.

```
source /opt/ros/groovy/setup.bash
export ROS_WORKSPACE=/home/pi/catkin_ws
```

Ir a la carpeta src:

```
cd src
```

Descargar el nodo raspicam:

```
git clone https://github.com/fpasteau/raspicam_node.git raspicam
```

Ir nuevamente al directorio de trabajo ROS, y compilar y crear el nuevo paquete:

```
cd ..
catkin_make
source devel/setup.bash
```

Y finalmente podremos ejecutar nuestro nodo de cámara:

```
rosrun raspicam raspicam_node
```

Intentamos usar nodos de cámara de los paquetes gscam y usb_cam, pero solo pudimos ejecutarlos en el PC. La Beaglebone, no tiene interfaz gráfica y por lo tanto no se podía lanzar los nodos de cámara y en la Raspberry pi, la versión de ROS actual no tiene disponible los paquetes de cámara gscam y usb_cam.

No obstante explicare como lanzar el nodo usb_cam para PC, en el cual recalco que esta funcionando:

Lo primero de todo sería instalar el paquete, el cual se puede hacer desde la misma fuente de ROS con:

```
sudo apt-get install ros-hydro-usb-cam
```

O desde repositorio:

```
roscd catkin_ws
svn co https://bosch-ros-pkg.svn.sourceforge.net/svnroot/bosch-ros-pkg/trunk/stacks/bosch_drivers
```

```
rospack profile
```

Nos movemos al directorio de usb_cam:

```
roscd usb_cam
```

Compilamos y construimos:

```
rosmake
```

Elegimos el dispositivo de cámara a usar:

```
ls /dev/video*
```

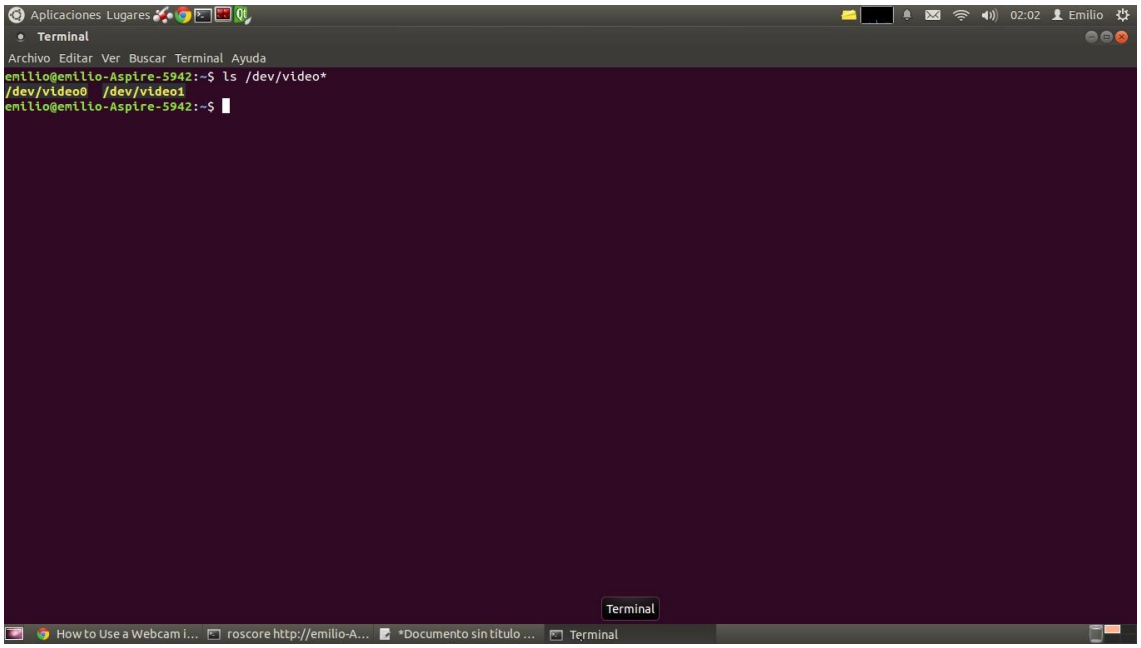


Figura 1. Dispositivos USB conectados al equipo

Editamos el archive de cámara:

```
nano usb_cam-test.launch
```

```
<launch>
 <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
  <param name="video_device" value="/dev/video0" />
  <param name="image_width" value="640" />
 <param name="image_height" value="480" />
  <param name="pixel_format" value="mjpeg" />
  <param name="camera_frame_id" value="usb_cam" />
```

```
  <param name="io_method" value="mmap"/>
 </node>
 <node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
  <remap from="image" to="/usb_cam/image_raw"/>
  <param name="autosize" value="true" />
 </node>
</launch>
```

Y lanzamos el nodo, antes habiendo lanzado antes ROS:

```
roscore
```

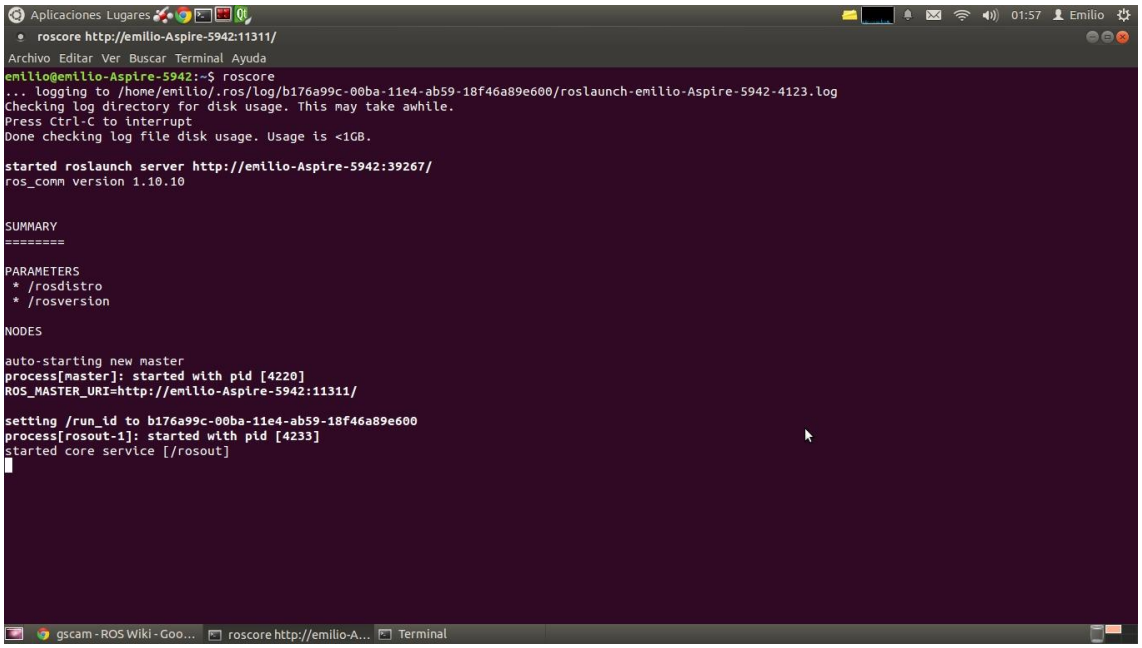


Figura 2. Ventana de terminal tras lanzar ROS

```
roslaunch usb_cam-test.launch
```

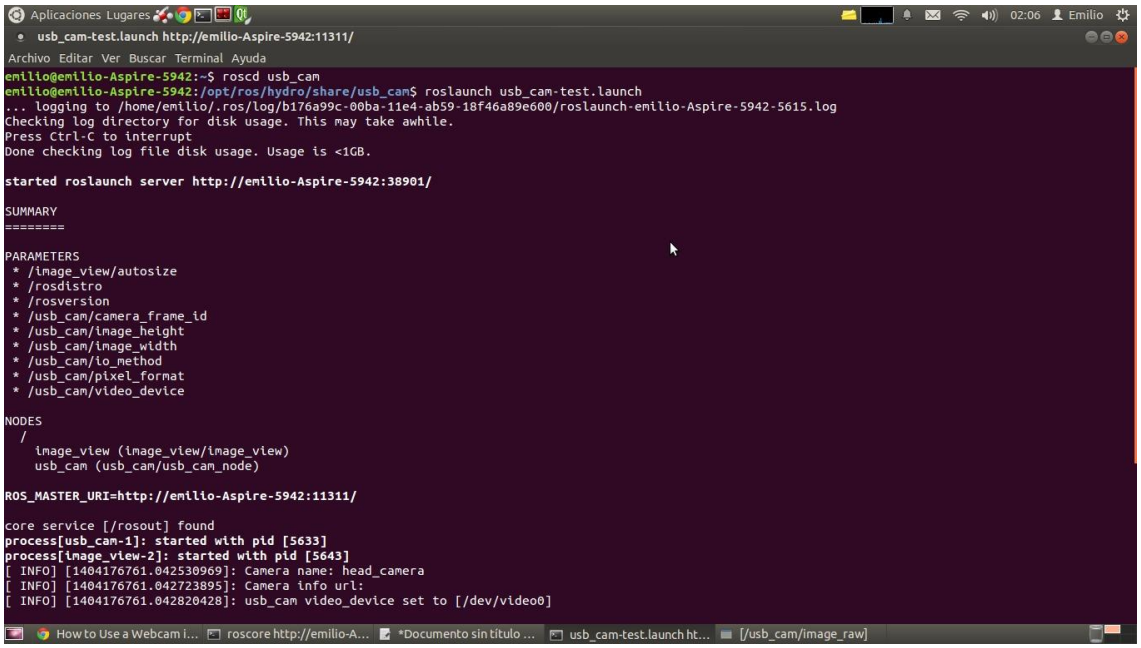


Figura 3. Ventanas de terminal tras lanzar USB_CAM (1 de 2)

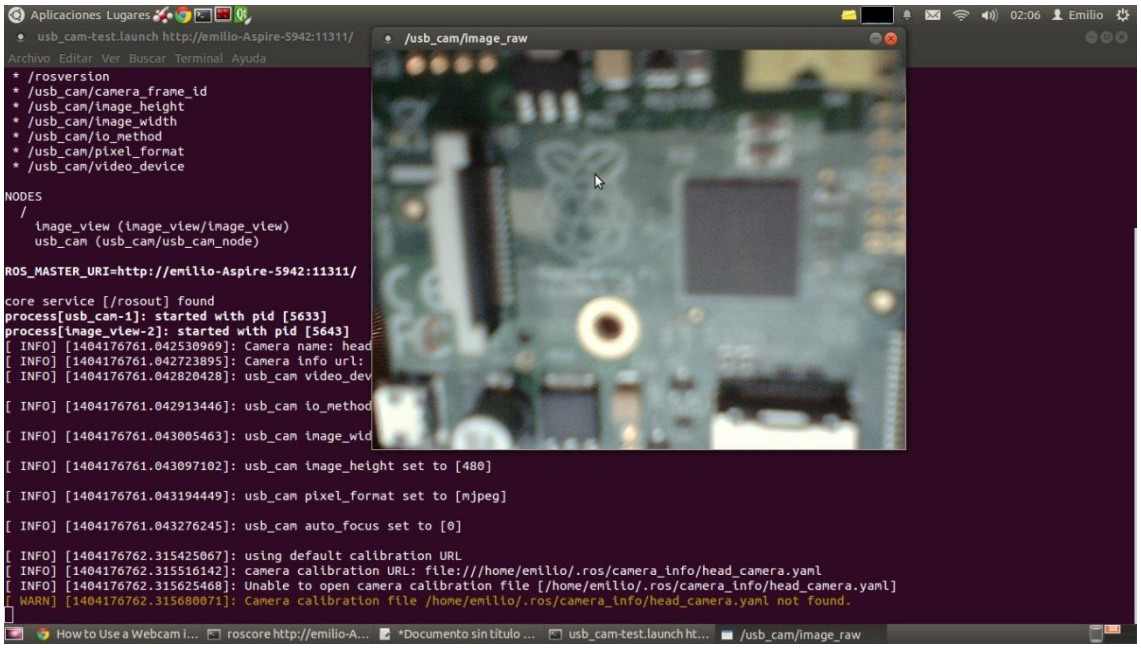


Figura 4. Ventanas de terminal tras lanzar USB_CAM (2 de 2)

```
rostopic hz --window=1000 /usb_cam/image_raw
```

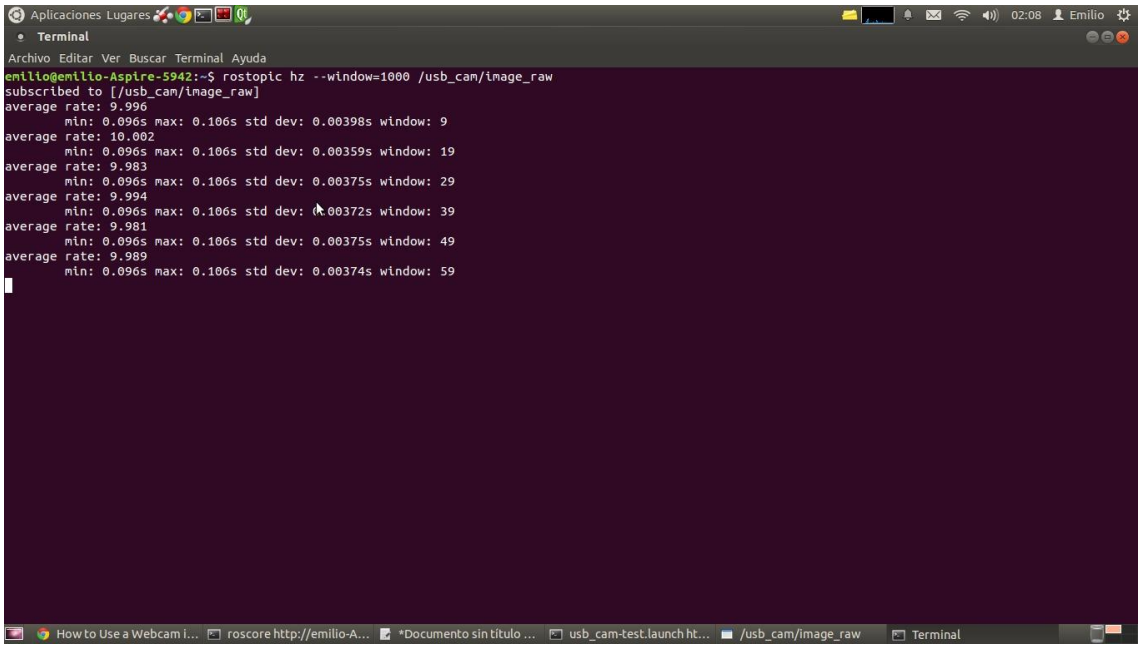


Figura 5. Modificación de uno de los tópicos de USB_CAM

Ejectutamos Rviz:

```
rosrun rviz rviz
```

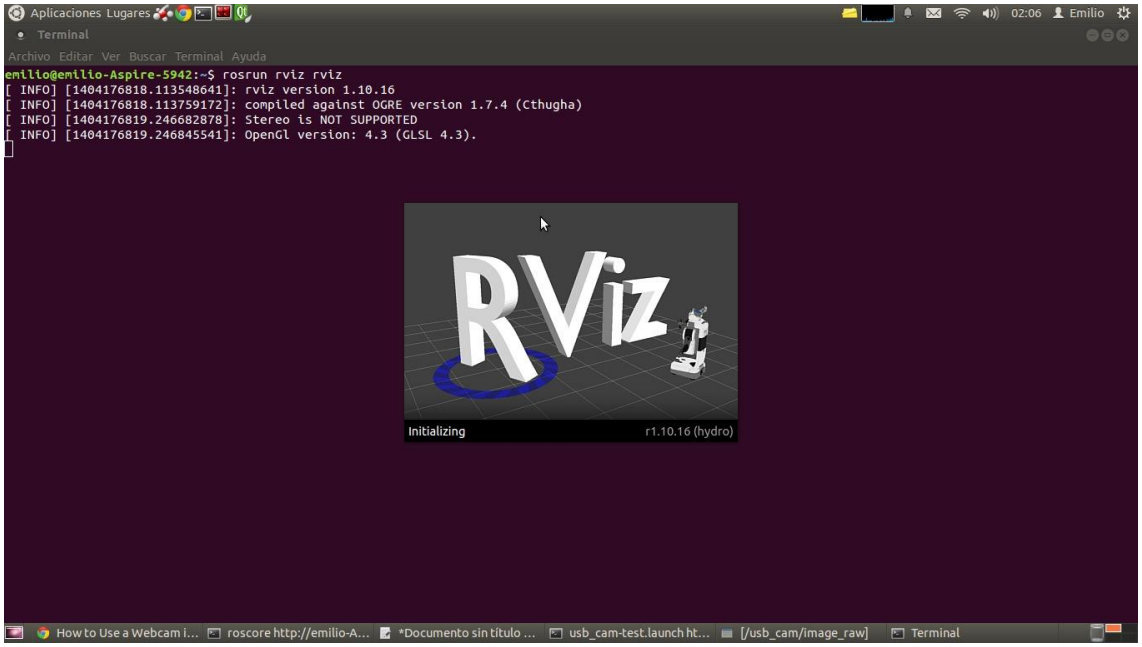


Figura 6. RViz

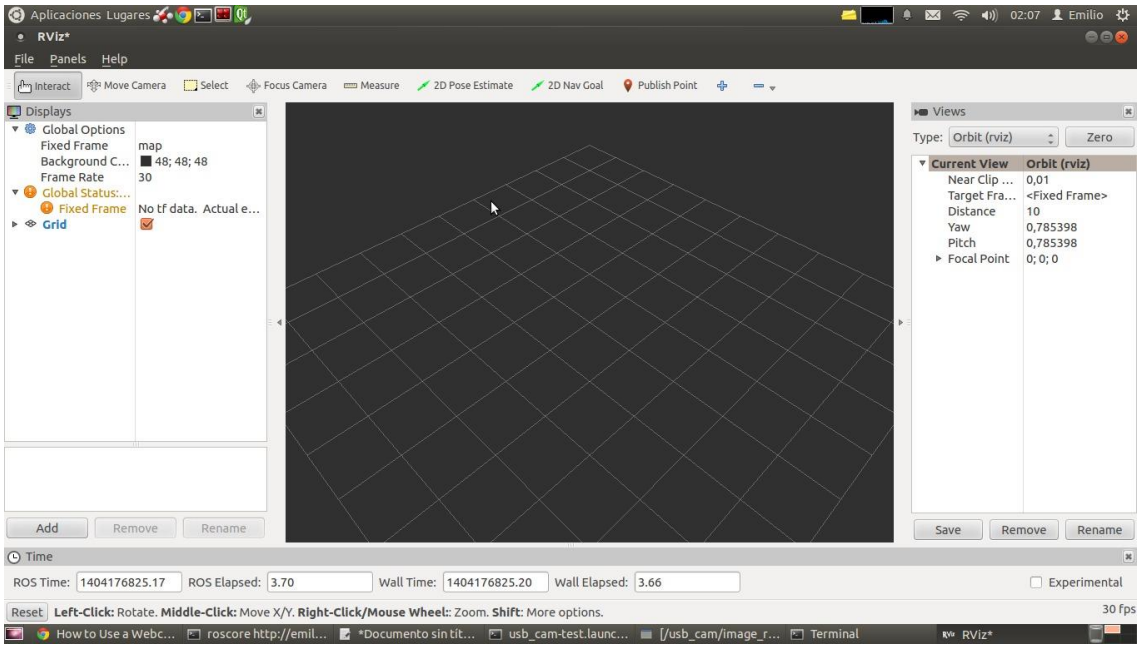


Figura 7. Interfaz RViz.

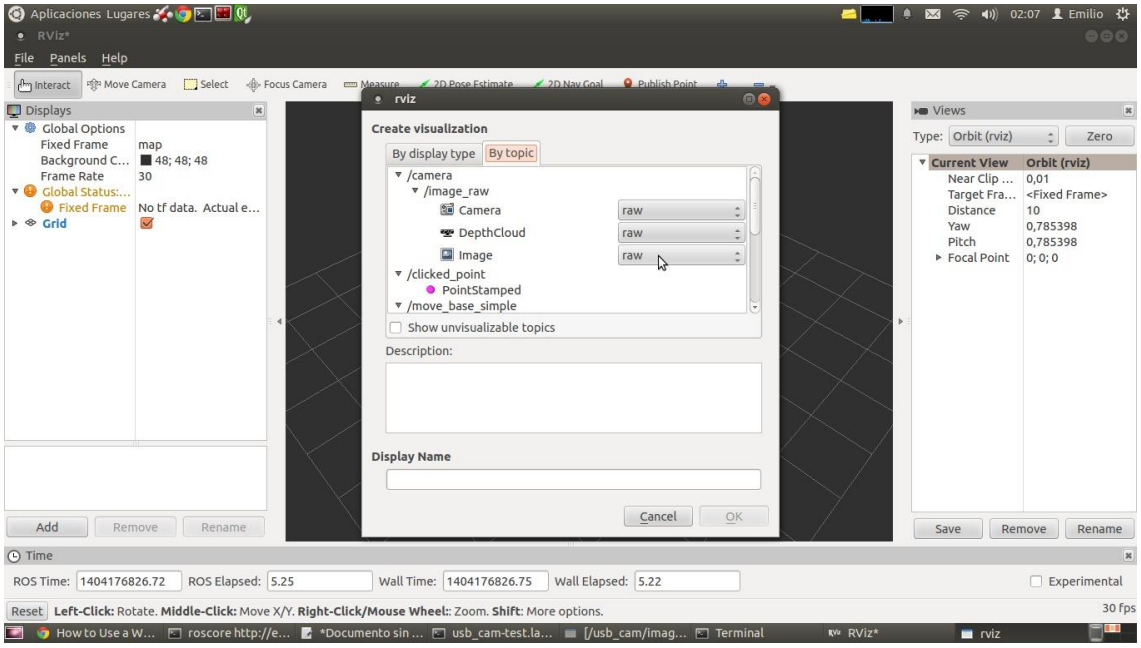


Figura 8. Interfaz RViz.

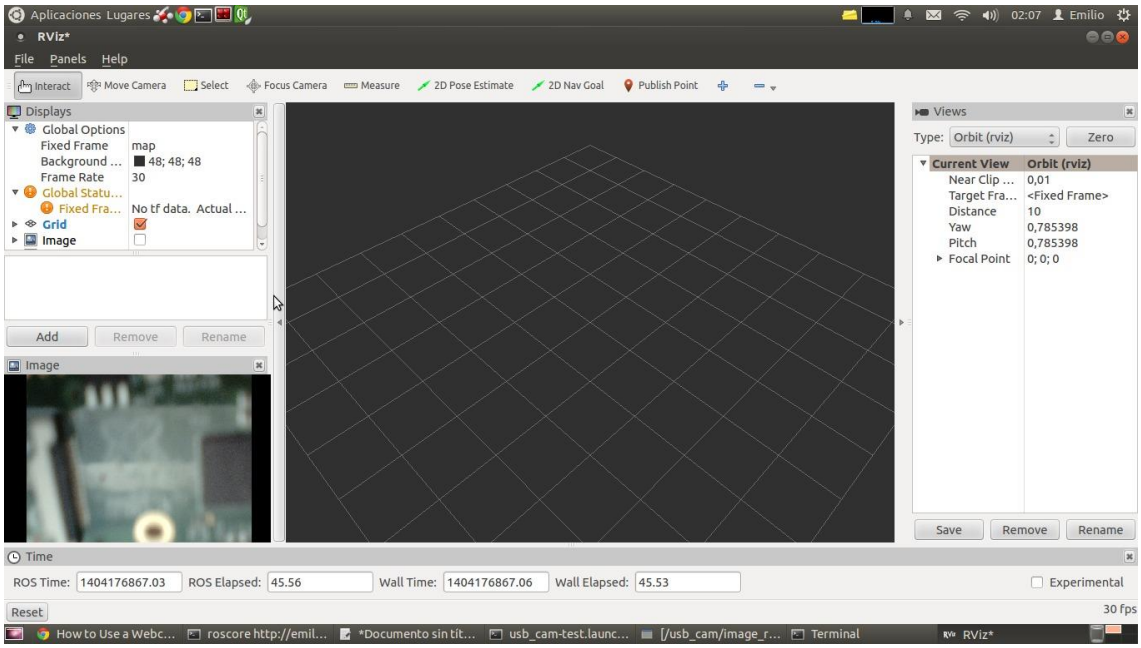


Figura 9. Interfaz RViz para ver la imagen en red de un esclavo

En nuestra red el Maestro es el PC y de esclavos tenemos a la Beaglebone y a la Raspberry pi, lo cual lo hemos comentado a lo largo del proyecto. Pero él envió de datos entre ellos se ha de realizar con un nodo de red. Y para ello usamos el rosserial.

Primero de todo hay que proceder a la instalación del nodo:

```
rocd /catkin_ws/src
git clone https://github.com/ros-drivers/rosserial
cd ..
catkin_make
catkin_make install
source ~/catkin_ws/install/setup.bash
roscd /sketchbook/libraries
rosrun rosserial_arduino make_libraries.py
```

Luego ya podemos proceder con el ejemplo típico de mensaje de comunicación entre dos sistemas ROS:

Editamos el archivo:

```
roscd /HelloRos/HelloROS.cpp
sudo nano HelloROS.cpp
```

```
/*
 * rosserial Ejemplo de publicador
 * Muestra "Hola Proyecto!"
 */
#include <ros.h>
#include <std_msgs/String.h>
#include <stdio.h>
```

Como parte de todo programa linux embebido ROS, es necesario incluir la ros.h archivo de cabecera y archivos de cabecera para los mensajes que se van a utilizar.

```
ros::NodeHandle  nh;
```

A continuación, tenemos que crear una instancia del controlador de nodo, lo que permite a nuestro programa para crear editores y suscriptores. El controlador de nodo también se ocupa de las comunicaciones a través del puerto de serie o LAN.

```
std_msgs::String str_msg;
ros::Publisher chatter("chatter", &str_msg);
```

Tenemos que crear una instancia de los editores y suscriptores que vamos a utilizar. Aquí creamos una instancia de un publicador con un nombre del tema de la "chatter". El segundo parámetro para Publisher es una referencia a la instancia de mensaje que se utilizará para su publicación.

```
char *rosSrvrIp = "192.168.1.35";
char hello[13] = "Hola Proyecto!";
```

RosSrvrIp es la dirección IP de la estación de trabajo ros donde rosserial_python corre; editar este valor y ponerlo en la dirección IP de su estación de trabajo ros. Rosserial_python redirigirá los mensajes publicados con el resto de ROS. Hola es el //mensaje que se publicará - una cadena estática.

```
int main()
    {
```

```
//nh.initNode();
nh.initNode(rosSrvrIp);
```

En la función principal que hay que inicializar el controlador de nodo ROS. Tiene que pasar en la dirección IP (y número de puerto si es necesario) del rosserial_python sistema en funcionamiento. Si usted no pasa en un parámetro, el sistema utilizará un puerto serie predeterminada.

NodeHandle.initNode () acepta argumentos como:

/ Dev/ttyUSB1 utiliza el puerto serie designado por mensajes a rosserial_python

192.168.1.2 se conecta a rosserial_python a la dirección IP especificada en el puerto predeterminado 11411

192.168.1.2:12345 conecta a rosserial_python a la dirección IP especificada en el número de puerto especificado

```
nh.advertise(chatter);
```

Publicidad ningún tema que se publican, y suscribirse a cualquiera de los temas que desea escuchar.

```
while(1) {
    str_msg.data = hello;
```

Por último, el programa entra en un tiempo (1) bucle en el que se publica datos luego duerme durante un segundo, para siempre. Usamos el nombre de la cadena como un puntero y lo asignamos al puntero de datos de mensaje.

```
chatter.publish( &str_msg );
nh.spinOnce();
printf("chattered\n");
sleep(1);
```

En el bucle, el nodo publica "Hola ROS" y llama ros :: spinOnce () donde todas las devoluciones de llamada de comunicación ROS son manejados.

```
}
}
```

Lanzando el código:

```
roscore
rosrun rosserial_python serial_node.py tcp
```

Construyendo el ejecutable (modificar CmakeList.txt):

```
cd<download_directory>
./helloros
```

Publicando mensaje:

```
rostopic echo chatter
```

Ahora comunicaremos las Mbed con las Beaglebone y viceversa con el nodo rosserial, pero vía serial y no en red:

Lo primero que tendremos que hacer, es crear un programa en la Mbed que mande información vía serial a la Beaglebone acorde al protocolo ROS. Para crear un programa en la Mbed tenemos que ir a su compilador e importar la librería rosserial_mbed_lib, y ya posteriormente programar la Mbed añadiendo las sentencias necesarias de ROS:

```
#define COMPILE_HELLOWORLD_CODE_ROSSERIAL
#ifdef  COMPILE_HELLOWORLD_CODE_ROSSERIAL
/*
 * rosserial Publisher Example
 * Prints "hello world!"
 */
#include"mbed.h"
#include <ros.h>
#include <std_msgs/String.h>
ros::NodeHandle  nh;
std_msgs::String str_msg;
ros::Publisher chatter("chatter", &str_msg);
char hello[13] = "hello world!";
```

```
int main() {
    nh.initNode();
    nh.advertise(chatter);
    while (1) {
        str_msg.data = hello;
        chatter.publish( &str_msg );
        nh.spinOnce();
        wait_ms(1000);
    }
}
#endif
```

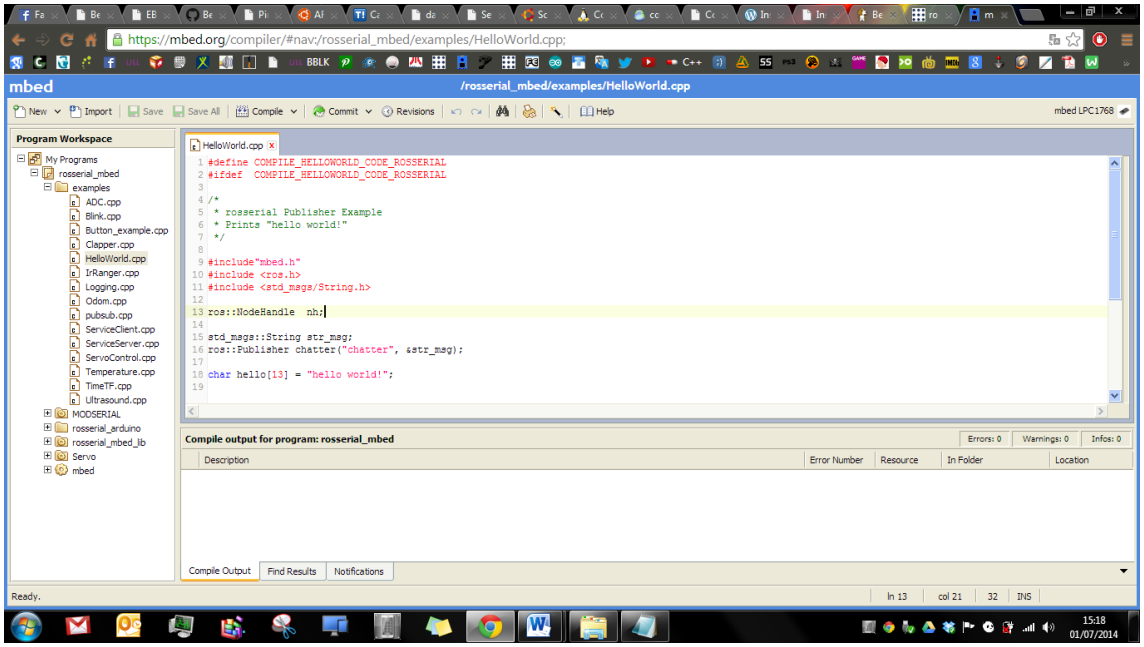


Figura 10. Compilador online de Mbed.

Lanzando el nodo:

```
roscore
rosrun rosserial_python serial_node.py _port:=/dev/ttyACM0
rostopic echo chatter
```

3. Referencias

ROS. Página oficial. https://wiki.ros.org

Mbed. Compilador: http://mbed.org/compiler/

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 11. ROS. Maestro-Esclavo

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción. 4

2. Configuración maestro-esclavo. 4

3. Referencias. 5

1. Introducción

Para configurar la conexión maestro esclavo principalmente lo que se hace es asignar direcciones IP fijas a los dispositivos, tanto a los maestros como a los esclavos. Si no se hiciera esto, habría que configurar la conexión cada vez que se inicie el sistema, por otra parte, hay que indicar quién es el esclavo y quién es el maestro.

Toda esta información se puede conseguir en la referencia 1.

2. Configuración maestro-esclavo

En primer lugar tenemos que saber los nombres de los equipos y sus direcciones IP dentro de la red local. Para ello escribiremos en una terminal "ifconfig" y anotaremos su dirección IP y el nombre de los equipos, fácilmente lo podemos visualizar en una terminal, justamente es lo que sigue después del nombre de usuario y la @.

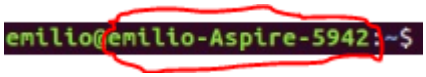


Figura 1. Captura de nuestro nombre de equipo

Lo siguiente es conectar los equipos entre si y finalmente añadir las variables de entorno necesarias para la jerarquía del sistema Maestro-Esclavo:

PC-Maestro:

Nombre del equipo: emilio-Aspire-5942

IP: 192.168.1.33

```
sudo gedit /etc/hosts
```

192.168.1.34 arm

192.168.1.35 raspberrypi

```
gedit ~/.bashrc
```

ROS_MASTER_URI=http://emilio-Aspire-5942:11311/

ROS_HOSTNAME=emilio-Aspire-5942

Beaglebone-Esclavo:

Nombre del equipo: arm

IP: 192.168.1.34

sudo nano /etc/hosts

192.168.1.33 emilio-Aspire-5942

192.168.1.35 raspberrypi

nano ~/.bashrc

ROS_MASTER_URI=http://emilio-Aspire-5942:11311/

ROS_HOSTNAME=arm

Raspberry Pi-Esclavo

Nombre del equipo: raspberrypi

IP: 192.168.1.35

sudo nano /etc/hosts

192.168.1.33 emilio-Aspire-5942

192.168.1.34 arm

nano ~/.bashrc

ROS_MASTER_URI=http://emilio-Aspire-5942:11311/

ROS_HOSTNAME=raspberrypi

3. Referencias

[1] https://wiki.ros.org

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 12. Descripción técnica de Beaglebone.

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción 4

2. Especificaciones generales 4
   2.1. Procesador 4
   2.2. Memoria 5
   2.3. Interface USB 5
   2.4. Puerto Serial 5
   2.5. Puerto JTAG 5
   2.6. Puertos USB 6
   2.7. Conector Micro SD 6
   2.8. Puerto USB cliente 6
   2.9. Fuentes de alimentación 6
   2.10. Botón RESET 7
   2.11. Entrada CTI JTAG 7
   2.12. Interfases de expansión 7
   2.13. LCD 7
   2.14. MMC1 7
   2.15. SPI 7
   2.16. I2C 8
   2.17. Puerto Serial 8
   2.18. Conversores A/D 8
   2.19. GPIO 8
   2.20. CAN BUS 8
   2.21. TIMERS 8
   2.22. PWM 8

3. Comunicaciones y puertos de expansión de la placa 9

4. Referencias 14

1. Introducción

Beaglebone es una placa del tamaño de una tarjeta de crédito construida por Texas Instruments, que tiene la funcionalidad de un ordenador en esas dimensiones. Posee un sistema operativo Linux Embebido, aunque también puede funcionar con Android como sistema operativo. Posee un procesador ARM Cortex-A8 AM335x de 720MHz, con una completa funcionalidad de puertas de entrada-salida completamente configurables y de aplicación general o particular a cada una de sus "capas" de expansión diseñadas para ella.

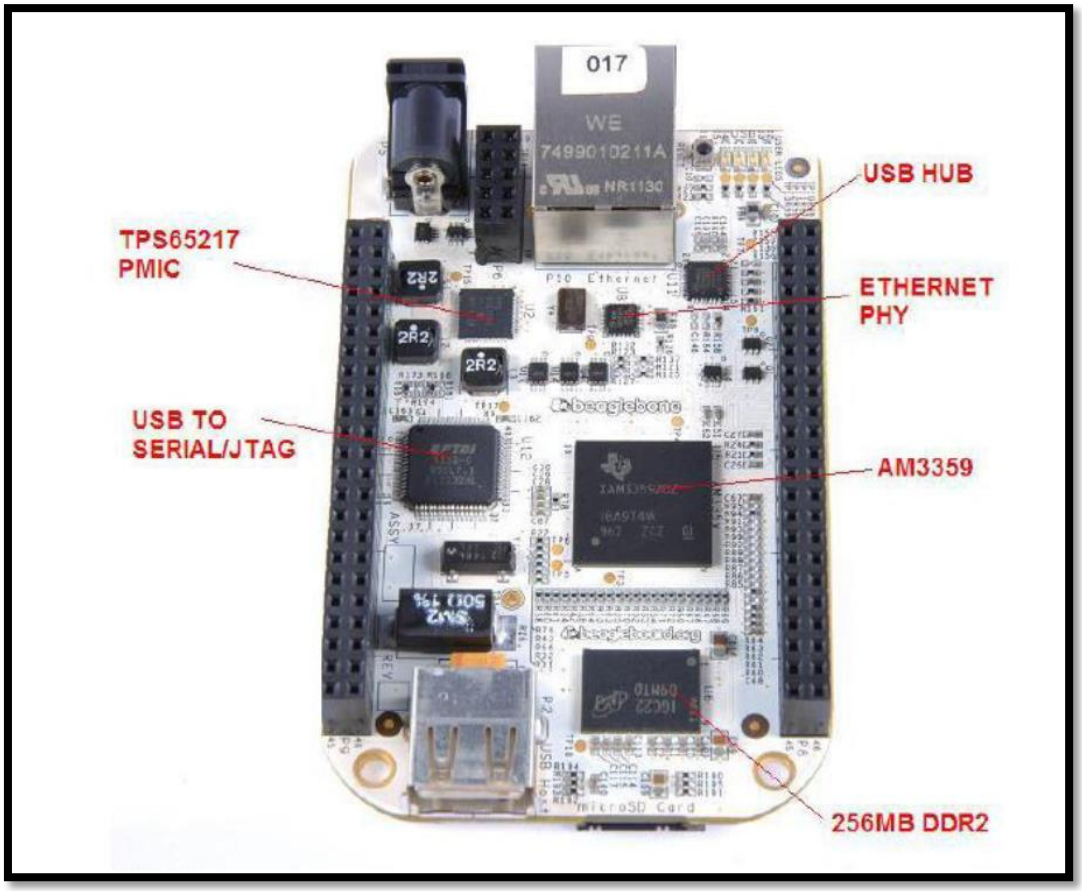


Figura 1: Placa Beaglebone

2. Especificaciones generales

2.1. Procesador

Utiliza un procesador AM3359 AM 3358. La velocidad real del procesador determinada por los dispositivos reales suministrados, y la forma de conexión.

2.2. Memoria

Memoria DDR2 de 16 bits, puede configurarse como de 128 MB o 256 MB, la configuración estándar es 256 MB a 400MHz. Contiene una memoria EPROM de 32 KB, con la información básica de la tarjeta

2.3. Interface USB

Concentra dos puertos USB, como funcionalidad para operar con:

- Comunicación serial USB.
- JTAG por USB
- Puerto de acceso USB al procesador.

Cuando se conecta a un Pc cada una de ellas se mostrará como un puerto en el pc.

2.4. Puerto serie

Posee un puerto serie a través de un UART0 con doble canal FT2232H USB, incluyen señales de Tx, Rx, RTS, CTS.

Una sola EEPROM se proporciona en el FT22323H para permitir la programación de la información del proveedor de manera que cuando se conecta, la tarjeta puede ser identificada y el controlador apropiado instalado.

2.5. Puerto JTAG

El segundo puerto en el FT2232H será utilizado para el puerto JTAG. Conexión directa con el procesador se realiza desde el FT2232H. Hay un encabezado JTAG proporcionado en la placa como una opción adicional.

2.6. Puerto USB0

El HUB se conecta directamente al puerto USB0 en el procesador. Esto permite que el puerto sea accesible desde el mismo conector USB como los puertos serie y JTAG.

2.7. Conector Micro SD

La tarjeta está equipada con un conector microSD para actuar como fuente de arranque principal. Una tarjeta microSD de 4 GB se suministra con cada placa. En el conector se admiten tarjetas SD de mayor capacidad.

2.8. Puerto USB1

En el tablero hay un solo conector USB tipo A con el apoyo total Host LS / FS / HS que se conecta a un USB1 en el procesador. El puerto puede proporcionar alimentación de encendido/apagado y hasta 500 mA ce corriente a 5V. Bajo la alimentación USB, la tarjeta no será capaz de suministrar los 500 mA completamente, pero debe ser suficiente para suministrar suficiente corriente para un dispositivo USB de baja potencia.

Puede utilizar una configuración de teclado /ratón inalámbrico o puede agregar un HUB para el teclado y el ratón de interfaz estándar si es necesario.

2.9. Puerto USB Cliente

Acceso a USB0 proporcionado por el concentrador USB integrado. Se mostrará en un PC como un dispositivo USB estándar.

2.10. Fuentes de poder

El sistema puede ser alimentado por un puerto USB de un PC o desde una fuente de alimentación de 5 VDC opcional. La fuente de alimentación no se suministra con la tarjeta y debe ser una fuente de alimentación conectada a tierra. El cable USB se incluye con la tarjeta.

Cuando se alimenta desde USB, la tarjeta está limitada a 500 MHZ. El consumo de energía FT2232H no puede proporcionar la corriente de 500 mA para el proceso de arranque. Para la operación 720 MHZ, se requiere alimentación de CC.

Se puede suministrar energía a través de un conector de 2,1 mm x 5,5 mm cuando se conecta a una fuente de alimentación positiva con valor nominal a 5 Vcc ± 0,1V y 1 A.

2.11. Botón reset

Cuando se presiona y se suelta, provoca un reset de la placa. Debido al pequeño tamaño del interruptor, no se experimenta una gran cantidad de rebotes al pulsar el interruptor.

2.12. Indicadores de estado

Hay 5 LED's verdes en el tablero. Cuatro pueden ser controlador por el usuario y uno de ellos es estático para indicación de funcionamiento. Los cuatro LED's verdes pueden ser controlados a través del SW mediante el manejo de los puertos GPIO.

2.13. Entrada CTI JTAG

Una entrada para facilitar el desarrollo de WE y la depuración de la tarjeta con diversos emuladores JTAG. Con el fin de usar el conector, debemos aislar el USB a la función de JTAG con unas resistencias.

2.14. Interfaces de expansión

Dos conectores de 46 pines de doble fila de 0,1 x 0,1 mm con conexión de Jack se suministra con la tarjeta para el acceso a las señales de expansión. Debido al número de pines se ha instalado un conector de baja fuerza de inserción para facilitar la extracción de los cabos, sin embargo, la extracción puede ser difícil y se debe tener cuidado cuando se realiza.

PERIFÉRICOS ADICIONALES QUE SE PUEDEN CONECTAR:

2.15. LCD

Un panel LCD completo de 24 bits puede ser conectado. Tiene luz de fondo y la funcionalidad de pantalla táctil, la potencia para la retroiluminación está limitada a 25 mA, por lo que se debe consultar el uso de paneles más grandes.

También se pueden conectar pantallas de LCD de 16 bits. La ventaja aquí es que esta utiliza menos patillas de los conectores de expansión dejando más señales para ser utilizadas por otras tarjetas de expansión.

2.16. Puerto GPMC

Se proporciona acceso al bus GPMC. Dependiendo de la configuración necesaria, esto puede resultar en la pérdida de la interfaz LCD, pues no existiría la capacidad disponible, y limitará el uso a una pantalla LCD para 16 bits solamente.

2.17. MMC1

A través de esta interface se proporciona acceso directo a la tarjeta MMC.

2.18. SPI

Existen dos puertos SPI (seriales) disponibles SPI00 y SPI01 accesibles por los pines de acceso.

2.19. I2C

Hay dos puertos I2C en el bus de expansión, I2C1 y I2C2. I2C2 se utiliza para las EEPROM sobre las tarjetas de expansión y siempre debe ser accesible. Se debe tener la precaución de no utilizar estas señales, pues otros componentes en un momento determinado pueden utilizar este bus.

2.20. Puerto serial

Hay 4 puertos serie con señales Tx, Rx, RTS, y CTS. Numerados como UART 1, 2, 4, el UART5 solamente tiene Tx y Rx y el UART 3 no está disponible para el usuario.

2.21. Conversores A/D

Siete conversores A/D de 100 Kmuestras por segundo, con un voltaje máximo de 1,8 Vcd.

2.22. GPIO

Un máximo de 66 pines GPIO son accesibles desde los puertos de expansión. Todos estos son pines de 3,3 V y se pueden configurar como entradas o salidas digitales.

2.23. CAN BUS

Hay dos interfaces CAN bus disponibles en la conexión de expansión BUS CAN versión 2 partes A y B. Las señales digitales RX y TX se identifican como se ve en la tabla al final.

2.24. TIMERS

Cuatro temporizadores de alta velocidad.

2.25. PWM

8 salidas PWM de alta resolución y modelo terminación simple.

3. Comunicaciones y puertos de expansión de la placa

La Beaglebone posee una serie de formas comunes de comunicación con el exterior, que van desde una simple comunicación serial, hasta redes y protocolos basados en patrones digitales que son usados a nivel industrial.

A continuación se presenta la placa Beaglebone vista desde su parte superior, aquí se enmascara la enorme capacidad de comunicaciones existentes, pues otras placas, como los arduinos, Raspberry, etc. Presentan muchos conectores específicos de cada puerto. La placa Beaglebone solamente posee un Jack de RJ45 para la red Ethernet (resaltado en verde), dos jacks, USO, uno tipo A (rojo) y otro tipo B (azul) y un lector de tarjetas micro SD (celeste). Los demás accesos a los medios de comunicaciones están en los dos header de 46 pines que posee en sus extremos (resaltados en color marrón), y que dan la posibilidad de personalizar la entrada del hardware que se necesite, ya que cada uno de estos pines no está dedicado ni personalizado para una sola entrada o salida, sino más bien manejan 7 modos de operación de cada uno, controlados por la configuración software de un multiplexor central, además posee un conector de expansión de 10 pines (amarillo).

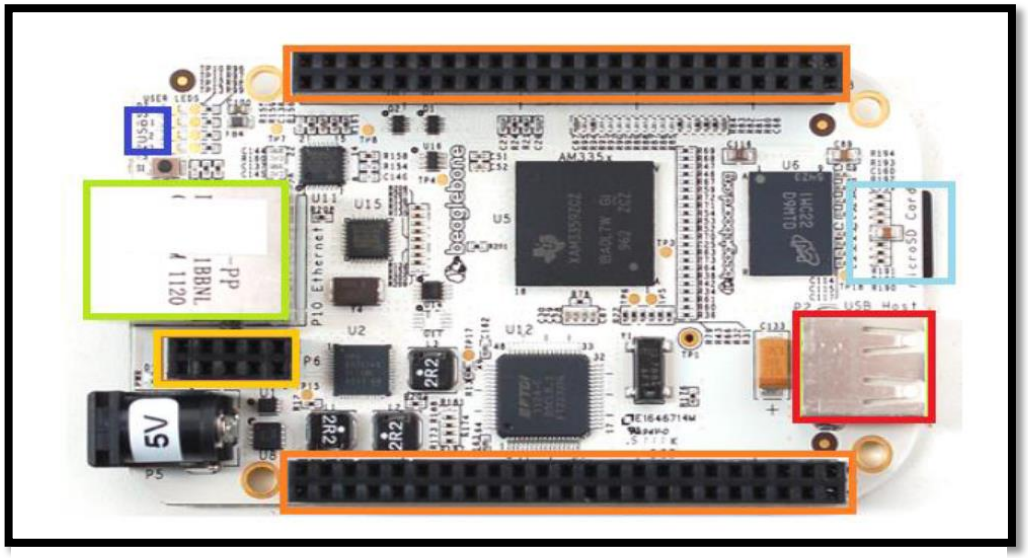


Figura 2: Composición Placa Beaglebone

Los headers de entradas y salidas se han identificado como P8 y P9, siento P8 el que está en la parte superior de la figura y P9 el otro. La descripción de la configuración disponible de cada uno de estos pines se puede observar en las siguientes tablas de descripción de cada uno de los puertos.

En primer lugar, los puertos en forma general, con los nombres que el fabricante ha decidido darle y luego las posibilidades de los pines.

SIGNAL NAME	PROC	CONN		PROC	SIGNAL NAME
	GND	1	2	GND	
GPIO1_6	R9	3	4	T9	GPIO1_7
GPIO1_2	R8	5	6	T8	GPIO1_3
TIMER4	R7	7	8	T7	TIMER7
TIMER5	T6	9	10	U6	TIMER6
GPIO1_13	R12	11	12	T12	GPIO1_12
EHRPWM2B	T10	13	14	T11	GPIO0_26
GPIO1_15	U13	15	16	V13	GPIO1_14
GPIO0_27	U12	17	18	V12	GPIO2_1
EHRPWM2A	U10	19	20	V9	GPIO1_31
GPIO1_30	U9	21	22	V8	GPIO1_5
GPIO1_4	U8	23	24	V7	GPIO1_1
GPIO1_0	U7	25	26	V6	GPIO1_29
GPIO2_22	U5	27	28	V5	GPIO2_24
GPIO2_23	R5	29	30	R6	GPIO2_25
UART5_CTSN	V4	31	32	T5	UART5_RTSN
UART4_RTSN	V3	33	34	U4	UART3_RTSN
UART4_CTSN	V2	35	36	U3	UART3_CTSN
UART5_TXD	U1	37	38	U2	UART5_RXD
GPIO2_12	T3	39	40	T4	GPIO2_13
GPIO2_10	T1	41	42	T2	GPIO2_11
GPIO2_8	R3	43	44	R4	GPIO2_9
GPIO2_6	R1	45	46	R2	GPIO2_7

Figura 3: Puertos Beaglebone

PIN	PROC	NAME	MODE0	MODE1	MODE2	MODE3
1		GND				
2		GND				
3	R9	GPIO1_6	gpmc_ad6	mmc1_dat6		
4	T9	GPIO1_7	gpmc_ad7	mmc1_dat7		
5	R8	GPIO1_2	gpmc_ad2	mmc1_dat2		
6	T8	GPIO1_3	gpmc_ad3	mmc1_dat3		
7	R7	TIMER4	gpmc_advn_ale		timer4	
8	T7	TIMER7	gpmc_oen_ren		timer7	
9	T6	TIMER5	gpmc_be0n_cle		timer5	
10	U6	TIMER6	gpmc_wen		timer6	
11	R12	GPIO1_13	gpmc_ad13	lcd_data18	mmc1_dat5	mmc2_dat1
12	T12	GPIO1_12	GPMC_AD12	LCD_DATA19	MMC1_DAT4	MMC2_DAT0
13	T10	EHRPWM2B	gpmc_ad9	lcd_data22	mmc1_dat1	mmc2_dat5
14	T11	GPIO0_26	gpmc_ad10	lcd_data21	mmc1_dat2	mmc2_dat6
15	U13	GPIO1_15	gpmc_ad15	lcd_data16	mmc1_dat7	mmc2_dat3
16	V13	GPIO1_14	gpmc_ad14	lcd_data17	mmc1_dat6	mmc2_dat2
17	U12	GPIO0_27	gpmc_ad11	lcd_data20	mmc1_dat3	mmc2_dat7
18	V12	GPIO2_1	gpmc_clk_mux0	lcd_memory_clk	gpmc_wait1	mmc2_clk
19	U10	EHRPWM2A	gpmc_ad8	lcd_data23	mmc1_dat0	mmc2_dat4
20	V9	GPIO1_31	gpmc_csn2	gpmc_be1n	mmc1_cmd	
21	U9	GPIO1_30	gpmc_csn1	gpmc_clk	mmc1_clk	
22	V8	GPIO1_5	gpmc_ad5	mmc1_dat3		
23	U8	GPIO1_4	gpmc_ad4	mmc1_dat4		
24	V7	GPIO1_1	gpmc_ad1	mmc1_dat1		
25	U7	GPIO1_0	gpmc_ad0	mmc1_dat0		
26	V6	GPIO1_29	gpmc_csn0			
27	U5	GPIO2_22	lcd_vsync	gpmc_a8		
28	V5	GPIO2_24	lcd_pclk	gpmc_a10		
29	R5	GPIO2_23	lcd_hsync	gpmc_a9		
30	R6	GPIO2_25	lcd_ac_bias_en	gpmc_a11		
31	V4	UART5_CTSN	lcd_data14	gpmc_a18	eQEP1_index	mcasp0_axr1
32	T5	UART5_RTSN	lcd_data15	gpmc_a19	eQEP1_strobe	mcasp0_ahclkx
33	V3	UART4_RTSN	lcd_data13	gpmc_a17	eQEP1B_in	mcasp0_fsr
34	U4	UART3_RTSN	lcd_data11	gpmc_a15	ehrpwm1B	mcasp0_ahclkr
35	V2	UART4_CTSN	lcd_data12	gpmc_a16	eQEP1A_in	mcasp0_aclkr

Figura 4: Opciones puertos Beaglebone

PIN	PROC	NAME	MODE0	MODE1	MODE2	MODE3
36	U3	UART3_CTSN	lcd_data10	gpmc_a14	ehrpwm1A	mcasp0_axr0
37	U1	UART5_TXD	lcd_data8	gpmc_a12	ehrpwm1_tripzone_in	mcasp0_aclkx
38	U2	UART5_RXD	lcd_data9	gpmc_a13	ehrpwm0_synco	mcasp0_fsx
39	T3	GPIO2_12	lcd_data6	gpmc_a6		eQEP2_index
40	T4	GPIO2_13	lcd_data7	gpmc_a7		eQEP2_strobe
41	T1	GPIO2_10	lcd_data4	gpmc_a4		eQEP2A_in
42	T2	GPIO2_11	lcd_data5	gpmc_a5		eQEP2B_in
43	R3	GPIO2_8	lcd_data2	gpmc_a2		ehrpwm2_tripzone_in
44	R4	GPIO2_9	lcd_data3	gpmc_a3		ehrpwm0_synco
45	R1	GPIO2_6	lcd_data0	gpmc_a0		ehrpwm2A
46	R2	GPIO2_7	lcd_data1	gpmc_a1		ehrpwm2B

Figura 5: Opciones puertos Beaglebone

A continuación, los modos 4, 5, 6 y 7.

PIN	PROC	NAME	MODE4	MODE5	MODE6	MODE7
1		GND				
2		GND				
3	R9	GPIO1_6				gpio1[6]
4	T9	GPIO1_7				gpio1[7]
5	R8	GPIO1_2				gpio1[2]
6	T8	GPIO1_3				gpio1[3]
7	R7	TIMER4				gpio2[2]
8	T7	TIMER7				gpio2[3]
9	T6	TIMER5				gpio2[5]
10	U6	TIMER6				gpio2[4]
11	R12	GPIO1_13	eQEP2B_in			gpio1[13]
12	T12	GPIO1_12	EQEP2A_IN			gpio1[12]
13	T10	EHRPWM2B	ehrpwm2B			gpio0[23]
14	T11	GPIO0_26	ehrpwm2_tripzone_in			gpio0[26]
15	U13	GPIO1_15	eQEP2_strobe			gpio1[15]
16	V13	GPIO1_14	eQEP2_index			gpio1[14]
17	U12	GPIO0_27	ehrpwm0_synco			gpio0[27]
18	V12	GPIO2_1			mcasp0_fsr	gpio2[1]
19	U10	EHRPWM2A	ehrpwm2A			gpio0[22]
20	V9	GPIO1_31				gpio1[31]

Figura 6: Puertos de la Beaglebone, del 4 al 7

21	U9	GPIO1_30				gpio1[30]
22	V8	GPIO1_5				gpio1[5]
23	U8	GPIO1_4				gpio1[4]
24	V7	GPIO1_1				gpio1[1]
25	U7	GPIO1_0				gpio1[0]
26	V6	GPIO1_29				gpio1[29]
27	U5	GPIO2_22				gpio2[22]
28	V5	GPIO2_24				gpio2[24]
29	R5	GPIO2_23				gpio2[23]
30	R6	GPIO2_25				gpio2[25]
31	V4	UART5_CTSN	uart5_rxd		uart5_ctsn	gpio0[10]
32	T5	UART5_RTSN	mcasp0_axr3		uart5_rtsn	gpio0[11]
33	V3	UART4_RTSN	mcasp0_axr3		uart4_rtsn	gpio0[9]
34	U4	UART3_RTSN	mcasp0_axr2		uart3_rtsn	gpio2[17]
35	V2	UART4_CTSN	mcasp0_axr2		uart4_ctsn	gpio0[8]
36	U3	UART3_CTSN			uart3_ctsn	gpio2[16]
37	U1	UART5_TXD	uart5_txd		uart2_ctsn	gpio2[14]

PIN	PROC	NAME	MODE4	MODE5	MODE6	MODE7
38	U2	UART5_RXD	uart5_rxd		uart2_rtsn	gpio2[15]
39	T3	GPIO2_12				gpio2[12]
40	T4	GPIO2_13	pr1_edio_data_out7			gpio2[13]
41	T1	GPIO2_10				gpio2[10]
42	T2	GPIO2_11				gpio2[11]
43	R3	GPIO2_8				gpio2[8]
44	R4	GPIO2_9				gpio2[9]
45	R1	GPIO2_6				gpio2[6]
46	R2	GPIO2_7				gpio2[7]

Figura 7: Puertos de Beaglebone, del 4 al 7

Como se puede observar al combinar todos los pines existentes en los header de la placa Beaglebone, las posibilidades son muy grandes, de tal forma que sería casi imposible encontrar una aplicación en la que todas las opciones fueran utilizadas, es por esto que se utilizan los modos distintos de operación del header. Además de que se pueden utilizar directamente los pines correspondientes, el fabricante da la posibilidad de utilizar recursos estándar de los sistemas como conectores e indicadores, es decir sus modos de entrada y salida estándar, esto se realiza mediante la adquisición de interfaces dedicadas para ello, conocidas como "capes", que se conectan en forma directa y muy fácil a los header, con la utilización de conectores de 46 pines machos.

4. Referencias

[1]Beaglebone: http://beagleboard.org/bone

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 13. Descripción técnica de Raspberry pi

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción 4

2. Características técnicas de la placa 4

3. Posibles proyectos con Raspberry pi. 5

4. Referencias. 6

1. Descripción

Raspberry Pi es un ordenador de placa reducida de bajo costo, desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

Figura 1. Raspbery pi.

El diseño incluye un System-on-a-chip Broadcom BCM2835, que contiene un procesador central (CPU) ARM1176JZF-S a 700 MHz (el firmware incluye unos modos "Turbo" para que el usuario pueda hacerle overclock de hasta 1 GHz sin perder la garantía), un procesador gráfico (GPU) VideoCore IV, y 512 MiB de memoria RAM (aunque originalmente al ser lanzado eran 256 MB). El diseño no incluye un disco duro ni unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente; tampoco incluye fuente de alimentación ni carcasa. El 29 de febrero de 2012 la fundación empezó a aceptar órdenes de compra del modelo B, y el 4 de febrero de 2013 del modelo A.

2. Características técnicas

A pesar de su pequeño tamaño y de su no aparente potencia, la Raspberry pi es un dispositivo de desarrollar grandes proyectos.

	Modelo A	Modelo B
SoC:[5]	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM + puerto USB)[3]	
CPU:	ARM 1176JZF-S a 700 MHz (familia ARM11)[3]	
Juego de instrucciones:	RISC de 32 bits	
GPU:	Broadcom VideoCore IV,[59] , OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia),[57] 1080p30 H.264/MPEG-4 AVC[3]	
Memoria (SDRAM):	256 MiB (compartidos con la GPU)	512 MiB (compartidos con la GPU)[4] desde el 15 de octubre de 2012
Puertos USB 2.0:[53]	1	2 (vía hub USB integrado)[52]
Entradas de vídeo:[60]	Conector MIPI CSI que permite instalar un módulo de cámara desarrollado por la RPF	
Salidas de vídeo:[5]	Conector RCA (PAL y NTSC), HDMI (rev1.3 y 1.4),[61] Interfaz DSI para panel LCD[62] [63]	
Salidas de audio:[5]	Conector de 3.5 mm, HDMI	
Almacenamiento integrado:	SD / MMC / ranura para SDIO	
Conectividad de red:[5]	Ninguna	10/100 Ethernet (RJ-45) vía hub USB[52]
Periféricos de bajo nivel:	8 x GPIO, SPI, I²C, UART[59]	
Reloj en tiempo real:[5]	Ninguno	
Consumo energético:	500 mA, (2.5 W)[5]	700 mA, (3.5 W)
Fuente de alimentación:[5]	5 V vía Micro USB o GPIO header	
Dimensiones:	85.60mm × 53.98mm[64] (3.370 × 2.125 inch)	
Sistemas operativos soportados:	GNU/Linux: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux. RISC OS[2]	

Figura 2. Características técnicas de Raspberry Pi.

Sus características técnicas hacen que podamos imaginar casi cualquier proyecto a nivel usuario y avanzado para desarrollar con la misma.

3. Posibles proyectos con Raspberry pi

Con la Raspberry pi se nos abre un mundo lleno de posibilidades gracias a su potencia y su gran desarrollo que está teniendo en la comunidad Raspberry pi.

Con ella podemos crear proyectos como controlar las luces de casa remotamente, crear un sistema telefónico con buzones de voz, crear un servidor web, o hacer de nuestro dispositivo un ordenador portátil trabajando sobre Linux.

Además puedes incorporarle teclado, ratón y demás dispositivos vía USB aparte de la cámara *raspicam* comercializada por Raspberry. La raspicam es un módulo de Raspberry pi que, aunque no de una gran resolución, permite por ejemplo montar un sistema de vigilancia casero.

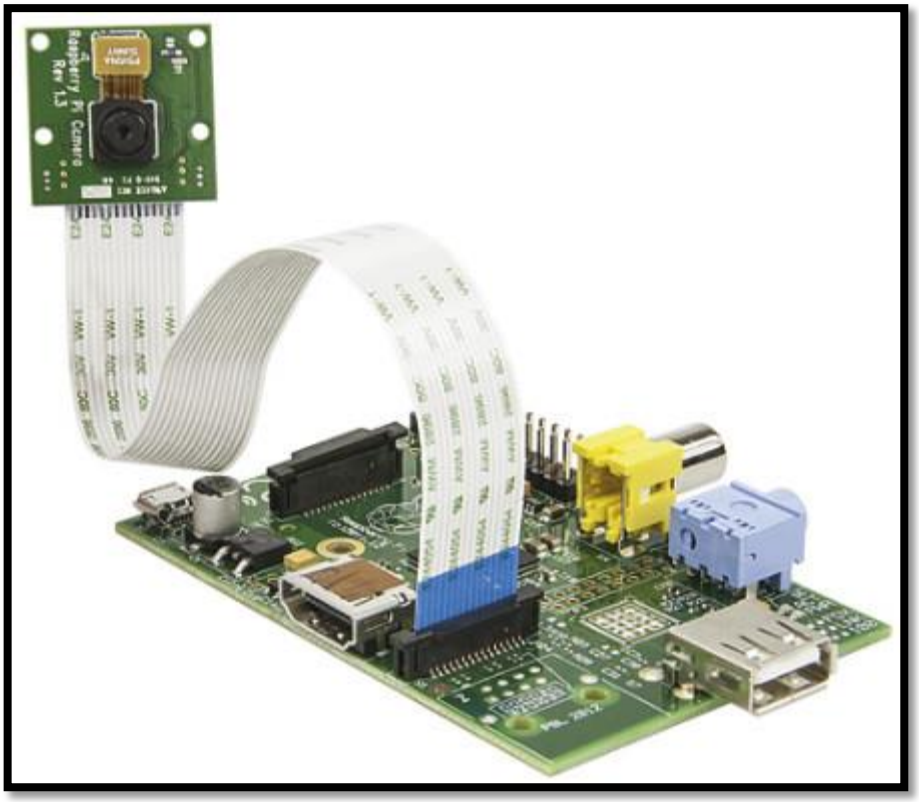


Figura 3. Raspberry pi con el módulo Raspicam

4. Referencias

[1]http://www.abc.es/tecnologia/informatica-hardware/20130716/abci-raspberry-como-201307151936.html

[2] http://es.wikipedia.org/wiki/Raspberry_Pi#Especificaciones_t.C3.A9cnicas

[3]http://www.raspberrypi.org/blog/

Universidad de La Laguna

Escuela Técnica Superior de Ingeniería Civil e Industrial

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA CIVIL E INDUSTRIAL

Anexo 14. Descripción técnica de Mbed.

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Alumno/a: Emilio Magariños Triviño y Avelino Labrador Fleitas

Tutor Académico: Manuel Rodríguez Valido.

Fecha: 04 de Julio de 2014

ÍNDICE Pág.

1. Introducción. 4

2. Especificaciones generales. 4

3. Compilador. 6

4. Referencias. 6

1. Introducción

Mbed es el primer hardware utilizado para desempeñar las tareas de la plataforma mbed. La plataforma mbed es utilizada para la creación de dispositivos inteligentes basados en micro controladores ARM Cortex-M.

El microcontrolador mbed, referenciado como *mbed NXP LPC1768* se ha diseñado en base a un controlador NXP. Tiene un núcleo ARM Cortex M3 funcionando a 96 MHz con 512 KB de memoria flash. Dispone de 64 Kb de RAM así como varias interfaces incluyendo Ethernet, USB, can, SPI, i2c y otras entradas y salidas.

2. Especificaciones generales

2.1. Apariencia física

La mbed es más pequeña que una tarjeta de crédito. En su cara frontal tiene el microprocesador, 4 leds, 1 botón y en la parte inferior tiene el conector USB. Por su cara inferior, en sus aristas laterales tiene los pines para la conexión de entrada y salida.

2.2. Pinout

El microcontrolador tiene una gran variedad de entradas y salidas. A continuación se puede ver una imagen del microcontrolador con el nombre de sus entradas/salidas.

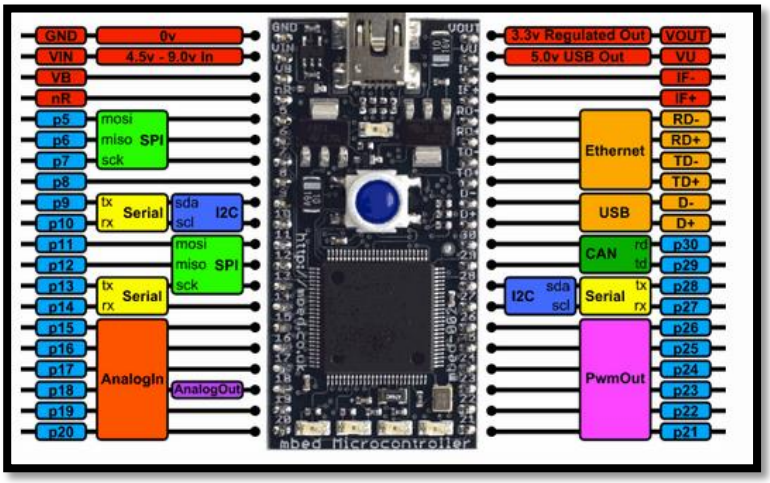


Figura 1: Conexiones Mbed

2.3. Alimentación

La placa se alimenta con el mismo cable USB a una tensión normalizada de 5V. No se requiere adaptadores externos ni otro tipo de fuentes.

2.4. Características y beneficios

La siguiente lista se extrajo de la página del controlador, referencia 1. En ella podemos ver una serie de características del mismo y de sus beneficios.

- ✓ Procesador ARM Cortex-M3, que funciona a frecuencias de hasta 100 MHz.
- ✓ Hasta 512 kB de memoria de programación de flash en el chip
- ✓ Hasta 64 kB RAM en el chip
- ✓ In-System Programming (ISP) e In-programación de aplicaciones (IAP)
- ✓ Controlador de DMA de uso general de ocho canales (GPDMA)
- ✓ MAC Ethernet con interfaz RMII y el controlador DMA dedicado
- ✓ Controlador de dispositivo full-speed USB 2.0 / Host / OTG
- ✓ Cuatro UART con generación fraccional velocidad de transmisión, FIFO interna y compatibilidad con DMA
- ✓ CAN CAN 2.0B controlador con dos canales
- ✓ Controlador SPI sincrónico,, comunicación full duplex de serie
- ✓ Dos controladores de SSP con FIFO y capacidad multi-protocolo
- ✓ Tres interfaces de bus mejorada I2C
- ✓ I2S (Inter-IC Sound) Interfaz
- ✓ 12-bit/8-ch convertidor analógico / digital (ADC) con las tasas de conversión de hasta 200 kHz
- ✓ 10-bit Digital / Analog Converter (DAC) con el temporizador de conversión dedicado y DMA
- ✓ Cuatro generales temporizadores / contadores de uso
- ✓ Uno PWM de control del motor con soporte para control de motores trifásicos
- ✓ Interfaz de codificador de cuadratura que puede controlar un codificador de cuadratura externa
- ✓ Temporizador de interrupción repetitiva para proporcionar alarmas temporizadas programables y repetición
- ✓ Cada periférico tiene su propio divisor de reloj para un mayor ahorro de energía
- ✓ Integra PMU (Unidad de Administración de energía)
- ✓ Cuatro modos de reducción de potencia: el sueño, sueño profundo, la Energía-abajo y profundo de apagado
- ✓ Solo 3,3 V fuente de alimentación (2,4 V a 3,6 V)
- ✓ Wake-up Interrupt Controller (WIC)
- ✓ Procesador de atención desde el modo de la Energía-abajo a través de cualquier interrupción

- ✓ Power-On Reset (POR)
- ✓ Oscilador de cristal con un rango de operación de 1 MHz a 25 MHz
- ✓ 4 MHz oscilador RC interno recorta a 1% de precisión
- ✓ Código protección de lectura (CRP) con diferentes niveles de seguridad

3. Compilador

El compilador de la mbed tiene una curiosidad y es que, aunque se pueda tener instalado en nuestro equipo, podemos tener todo nuestro trabajo guardado en la nube en el compilador online que tiene la plataforma mbed.

Podemos acceder a él registrándonos en www.mbed.org donde encontraremos toda la información del microcontrolador, códigos ejemplo hechos, foros, etc. En este microcontrolador se guarda todo nuestro trabajo lo que nos ayuda a no perder nada y tener a mano cualquier programa que hayamos realizado en otro momento.

En cuanto a complejidad no hay nada que decir puesto que una vez editado el programa basta con darle a compilar y automáticamente el compilador online descarga el archivo que debe ser copiado en la memoria del microcontrolador.

Una vez insertado el archivo, basta con darle al botón de la placa para que comience a ejecutarse el programa.

4. Referencias.

[1] http://www.nxp.com/products/microcontrollers/cortex_m3/LPC1768FBD100.html

[2] https://mbed.org/