ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA MECÁNICA

TEMA: "DISEÑO Y SIMULACIÓN DE UN CONTROLADOR PARA UN PÉNDULO INVERTIDO VIRTUAL EJECUTANDO EL MIDDLEWARE OROCOS E IMPLEMENTACIÓN DEL CONTROLADOR EN UNA RASPBERRY PI CON COMUNICACIÓN AL SIMULADOR"

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGÍSTER EN MECATRÓNICA Y ROBÓTICA

FRANCISCO XAVIER ÁLVAREZ ERAZO

francisco.alvarez@epn.edu.ec

DIRECTORA:
ANA VERÓNICA RODAS BENALCAZAR

ana.rodas@epn.edu.ec

Quito, marzo, 2019

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por **AUTOR FRANCISCO XAVIER ÁLVAREZ ERAZO**, bajo mi supervisión.

_____

Ana Verónica Rodas Benalcázar

DIRECTORA DE PROYECTO

DECLARACIÓN

Yo, **Francisco Xavier Álvarez Érazo**, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Francisco Xavier Álvarez Érazo

DEDICATORIA

Dedico mi logro académico a mi Madre Mercedes Erazo y a mi Abuelito Mariano Erazo, por permanecer conmigo cuando más los he necesitado.

También dedico mi triunfo a Johanna Flores, no olvidaré lo que hiciste por mí cuando había tormenta, espero que al leer estas líneas, comprendas lo que significó para mí la ayuda que me brindaste en aquellos momentos difíciles.

Francisco Álvarez

AGRADECIMIENTO

Agradezco a la Ing. Ana Rodas por confiar en mí una vez más, por el cariño que me ha brindado y por el apoyo en mi vida profesional.
Un agradecimiento a mi amigo Julio Merizalde por su apoyo a pesar de la distancia.
Mil gracias a todas las personas que hicieron posible un sueño más.

Francisco Álvarez

ÍNDICE

RESUMEN

El presente trabajo de titulación contiene el diseño y la simulación un controlador para un péndulo invertido virtual ejecutando el middleware OROCOS e implementación en una Raspberry Pi con comunicación al simulador. En primer lugar se seleccionó la Raspberry Pi adecuada para instalar y ejecutar UBUNTU MATE, en la cual se instaló y ejecutó OROCOS TOOLCHAIN, lo que permitió crear un componente y ejecutar el deployer. Además se instaló en una máquina virtual el sistema operativo UBUNTU MATE para poder instalar ROS, OROCOS y V-REP. Luego se diseñó el controlador para el péndulo invertido virtual para que sea ejecutado tanto en la máquina virtual como en la Raspberry Pi. A continuación se implementó la escena del péndulo invertido virtual para ejecutar su controlador escrito en Python por medio de la API para enlazar OROCOS TOOLCHAIN y PHYTON. La conexión fue establecida, por medio de una interfaz de adquisición de datos, entre la Raspberry Pi y la escena del péndulo virtual en V-REP para su respectivo control.

Palabras clave: OROCOS TOOLCHAIN, Raspberry Pi, V-REP, Péndulo Invertido

ABSTRACT

The project contains a controller's design and simulation for a virtual inverted pendulum executing the middleware OROCOS and its implementation in a Raspberry Pi communicated with the simulator. First the Raspberry Pi was selected to install and run UBUNTU MATE, in which OROCOS TOOLCHAIN was installed and executed, that allowed to create a component and execute the deployer. In addition, the UBUNTU MATE operating system was installed in a virtual machine in order to run ROS, OROCOS and V-REP. In a second stage, the controller for the virtual inverted pendulum was designed to be executed in both the virtual machine and the Raspberry Pi. Later on, the virtual inverted pendulum scene was implemented to execute its driver, written in Python, through the API to link OROCOS TOOLCHAIN and PHYTON. Finally, the connection was established by means of a data acquisition interface between the Raspberry Pi and the virtual pendulum scene in V-REP for its respective control.

Keywords: OROCOS TOOLCHAIN, Raspberry Pi, V-REP, Inverted Pendulum

“DISEÑO Y SIMULACIÓN DE UN CONTROLADOR PARA UN PÉNDULO INVERTIDO VIRTUAL EJECUTANDO EL MIDDLEWARE OROCOS E IMPLEMENTACIÓN DEL CONTROLADOR EN UNA RASPBERRY PI CON COMUNICACIÓN AL SIMULADOR”

INTRODUCCIÓN

Desde el 2001 comenzó el proyecto OROCOS ("Open RObot Control Software". En contextos no robóticos, "Servicios de control en tiempo real abiertos" podría ser un nombre más apropiado) [24], cuyo objetivo fue convertirse en un paquete de software de control de robot de uso general y abierto.

Los controladores en tiempo real implementados por software con el middleware OROCOS utilizan como hardware computadoras Pentium 4, debido a la disponibilidad del Bus PCI.

A pesar de sus grandes expectativas se han desarrollado pocos proyectos implementados en una Raspberry Pi.

En el 2013, se concluyó que la Raspberry Pi ha destacado por su bajo precio y alta potencia, aunque no fue posible integrar Orocos+Ros en ella, en consecuencia no fue posible aprovechar las facilidades y utilidades de Ros en proyectos realizados con Orocos, y en el 2015, se realizó un estudio sobre la implementación de ROS (Robot Operating System) en Raspberry Pi realizando la validación experimental en una planta de levitación magnética [8]

Los proyectos simulados que utilizan el middleware OROCOS son escasos, lo que es una desventaja ya que se debe probar los controladores en plantas reales.

En el 2015, en la publicación Deployment of Model Based Robotic Control algorithms, designed using Matlab/Simulink, in the form of OROCOS components operating under Linux Xenomai, se buscó optimizar el código de C ++ utilizado para crear OROCOS, a pesar de que las simulaciones no se ejecutan en tiempo real. [9]

El presente proyecto pretende la simulación de la aplicación, en este caso del controlador péndulo invertido, con un enlace que vincula Python (lenguaje de programación que permite trabajar rápidamente e integrar sus sistemas de manera más efectiva) y OROCOS, por medio del simulador V-REP que ofrece una API (Application Programming Interface) remota para controlar una simulación (o el simulador en sí) desde una aplicación externa o un hardware remoto (por ejemplo, un robot real, una

computadora remota, etc.). Además, se implementará un componente de OROCOS en una Raspberry Pi en el sistema operativo Ubuntu, en cual deberá estar instalado el paquete rtt_ros_integration para demostrar la ejecución del deployer.

Dentro del desarrollo de la robótica, la implementación de controladores en una Raspberry Pi, continuaría con las investigaciones más relevantes que son "Desarrollo de aplicaciones embebidas de control" de la Universidad Politécnica de Valencia donde no fue posible integrar Orocos+Ros y "Tool to Perform Software-In-the-Loop through Robot Operating System". La instalación del middleware OROCOS en una Raspberry Pi se ha hecho sobre el sistema operativo Raspbian, desafortunadamente, no ha sido posible instalar Ros con Orocos integrado [10].

En el proyecto que se presenta se utilizará el middleware OROCOS sobre el sistema operativo Ubuntu Mate, ya que en una investigación anterior, en una Raspberry Pi 2 se pudo montar dos sistemas operativos que fueron Raspbian Jessie y Ubuntu Mate, donde se comprobó que el desempeño en cuanto a tiempo de instalación y ejecución de las aplicaciones dio mejor resultado en Ubuntu Mate, con una diferencia no muy extensa de Raspbian Jessie [12].

La simulación se la realizará en V-REP, el cual es un simulador de robots, con entorno de desarrollo integrado, que se basa en una arquitectura de control distribuido: cada objeto - modelo puede controlarse individualmente a través de un script incrustado, un complemento, un nodo ROS o BlueZero, un cliente API remoto o un sistema personalizado. V-REP se utiliza para el desarrollo rápido de algoritmos, simulaciones de automatización de fábricas, prototipado y verificación rápidos, educación relacionada con la robótica, monitoreo remoto, doble verificación de seguridad, etc. [11]

Debido a las ventajas que presenta V-REP, será posible enlazar el middleware OROCOS a través de una interfaz de programación de aplicaciones, API (Application Programming Interface) a dicho simulador.

Objetivo general

Diseñar y simular un controlador para un péndulo invertido virtual ejecutando el middleware OROCOS e implementarlo en una Raspberry Pi con comunicación al simulador

Objetivos específicos

- Seleccionar la Raspberry Pi adecuada para instalar y ejecutar UBUNTU MATE, en la cual se instalará y ejecutará OROCOS TOOLCHAIN, para crear un componente y ejecutar el deployer.
- Instalar en una máquina virtual el sistema operativo UBUNTU MATE para poder instalar ROS, OROCOS y V-REP.
- Diseñar el controlador para el péndulo invertido virtual para que sea ejecutado tanto en la máquina virtual como en la Raspberry Pi.
- Implementar la escena del péndulo invertido virtual y ejecutar su controlador escrito en Python por medio de la API para enlazar OROCOS TOOLCHAIN y PHYTON.
- Establecer una conexión, por medio de una interfaz de adquisición de datos, entre la Raspberry Pi y la escena del péndulo virtual en V-REP para su respectivo control.

Alcance

Instalar el sistema operativo UBUNTU MATE, ROS, PYTHON y OROCOS TOOLCHAIN tanto en la máquina virtual como en la Raspberry Pi.
Realizar ficheros en Python para el manejo de los GPIO
Diseñar el controlador para el péndulo invertido virtual.
Crear el programa del controlador del péndulo invertido por componentes.
Implementar el controlador diseñado en una máquina virtual y en una tarjeta Raspberry Pi, para probar que el componente creado con OROCOS se ejecuta satisfactoriamente.
Para controlar la posición vertical del péndulo invertido, se necesitará información de su sentido de giro, ya sea este en sentido horario u antihorario, lo cual nos indicará si se mueve hacia la derecha o hacia la izquierda, con el propósito de mantener el punto de equilibrio establecido.
El controlador minimizará el error de desviación de la barra del punto de equilibrio. Para minimizar el error hasta el valor de cero, se manipulará los datos que permitan aumentar o disminuir la velocidad del servomecanismo.
Para la simulación en la máquina virtual del péndulo invertido en V-REP, se crearán dos ficheros de texto, en el uno el middleware OROCOS escribirá los parámetros y se enviará al V-REP. En el otro fichero V-REP escribirá los datos de los sensores y se enviará al middleware OROCOS.
Se establecerá una conexión, con una interfaz de adquisición de datos, entre la Rasberry Pi y la escena del péndulo invertido virtual de V-REP, para lograr simular el controlador del péndulo invertido en V-REP.

1. MARCO TEÓRICO

1.1. Middleware OROCOS

En diciembre del año 2000, la Red Europea de Robótica tuvo la iniciativa para desarrollar un proyecto cuyo eje principal sería el uso de software libre. Por más de veinte años experimentando, se obtuvieron resultados fallidos debido al uso de software comercial para el control de robots, en una investigación de robótica que exigía tener software con ventajas de programación. OROCOS en ese momento se convirtió en una idea innovadora, ya que permitiría a muchos usuarios tener acceso al control de robots con software de código abierto. [1]

El objetivo del proyecto OROCOS es que los desarrolladores tengan la flexibilidad en el software de control de robots, que les permita construir su sistema desde cero o continuar un proyecto de robótica, contribuyendo con los módulos en los cuales estén interesados. Los desarrolladores cuentan con un sistema de alta calidad técnica, con documentación e ingeniería de software, que les permite crear componentes que pueden ser agregados o eliminados de sus sistemas. [1]

Debido a la importancia de OROCOS para el control de robots, el presente trabajo de titulación pertenece al proyecto de investigación PIMI (Proyecto de Investigación Multi e Inter Disciplinario) 15-04 de la Escuela Politécnica Nacional, por lo que es indispensable realizar la correcta ejecución de OROCOS, saber el procesos para la creación de un componente e instalar ROS (Robot Operating System) y PYTHON (es un lenguaje de programación que le permite trabajar más rápidamente e integrar sus sistemas de manera más efectiva) [25], realizando todos estos procesos en open hardware, ayudando de esta manera a que los desarrolladores tengan una herramienta más para sus proyectos.

1.1.1. Orocos Real-Time Toolkit (RTT)

El RTT permite la creación de aplicaciones robóticas en C++, con los medios básicos y funcionalidad similares a las de una aplicación. Los componentes tiene la característica principal de ejecutarse en tiempo real, permitiendo a los diseñadores crear aplicaciones de control en tiempo real basadas en componentes interactivas y altamente configurables.

La biblioteca Real-Time Toolkit ofrece opciones de scripting en tiempo real, la API de distribución y comunicación de componentes y la configuración XML (eXtensible Markup Language). [1]

Los componentes creados pueden ser usados para controlar desde sensores hasta robots complejos, para organizar flujos de datos, para conectarse a una interfaz de usuario o para optimizar un algoritmo ajustando su tiempo de ejecución. [1]

Cada componente es una extensión de la primitiva TaskContext, un objeto activo que ofrece puertos seguros y eficaces para subprocesos para el intercambio de datos (sin bloqueo), puede reaccionar ante eventos, procesar comandos o ejecutar máquinas de estados finitos en tiempo real. [1]

El lenguaje de máquina de estado RTT permite la construcción de máquinas de estado simples, no jerárquicas, que pueden cargarse en componentes en el tiempo de ejecución. Esta es una gran ventaja en comparación con los enfoques de ingeniería basados en modelos como MDA (Model Driven Architecture) y xUML (método de desarrollo de software y un lenguaje de software altamente abstracto) que generalmente requieren una transformación del modelo y / o un paso de compilación antes de ser ejecutables.

La necesidad de un modelo de máquina de estado jerárquico más expresivo se ha hecho evidente. La solución obvia sería extender la implementación existente del RTT. Sin embargo, esto no resolvería el problema de la portabilidad de las máquinas de estado y los scripts. Además, este enfoque aumentaría aún más la complejidad y el costo constante del mantenimiento de la implementación.

Un enfoque alternativo es implementar las máquinas de estado jerárquicas requeridas como un lenguaje interno específico del dominio (DSL). Esta técnica es una forma eficiente de implementar nuevos lenguajes de programación al construirlos sobre un lenguaje host existente. Esto tiene la ventaja de evitar el desarrollo de una nueva sintaxis y un analizador asociado. Pero, lo que es más importante, el DSL interno permite la construcción de idiomas que pueden ejecutarse directamente como scripts interpretados en el idioma principal sin la necesidad de pasos de transformación adicionales.

1.2. Raspberry Pi

En [20] se encuentra el análisis del tiempo medio y el tiempo máximo de ejecución, tanto para un único proceso como para cuatro procesos concurrentes, entre tres tarjetas open hardware que son IGEP, Raspberry Pi y BeagleBone, concluyendo que los tiempos fueron mejores en la Raspberry Pi, ya que para un único proceso el tiempo medio en la Raspberry Pi fue de 2.99ms y el máximo de 18.2ms, mientras que en la BeagleBone fueron de 11.25ms y de 33ms y en la IGEP de 11.14ms y de 32.8ms, respectivamente. Para cuatro procesos concurrentes el tiempo medio en la Raspberry Pi fue de 21.9ms y

el máximo de 48.5ms, mientras que en la BeagleBone fueron de 49.7ms y de 142.2ms y en la IGEP de 18.62ms y de 139.89ms, respectivamente.

Pero en el trabajo de [20] no lograron integrar OROCOS+ROS en dicha tarjeta. En vista que la Raspberry Pi ofrece alta eficiencia y bajo costo, fue escogida para el presente proyecto con el fin de integrar OROCOS+ROS en ella y crear un componente que nos asegure una implementación exitosa de OROCOS. A continuación se detalla las características de la Raspberry Pi.

La Raspberry Pi es una computadora de bajo costo, del tamaño de una tarjeta de crédito que se conecta a un monitor de computadora o televisor, y usa un teclado y mouse estándar. Es un pequeño dispositivo que permite a personas de todas las edades explorar y aprender a programar en lenguajes como Scratch y Python. Es capaz de hacer todo lo que se espera que haga una computadora de escritorio, desde navegar por Internet y reproducir videos de alta definición hasta hacer hojas de cálculo, procesamiento de textos y juegos. Además, la Raspberry Pi tiene la capacidad de interactuar con el mundo exterior y se ha utilizado en una amplia gama de proyectos electrónicos. [2]

Raspberry Pi es una buena opción como computadora de escritorio de uso general y de baja potencia. Aunque nunca alcanzará el mismo nivel de rendimiento que una computadora de escritorio o una computadora portátil estándar, su consumo de energía es bajo y respetuoso con el medio ambiente lo que ayuda a compensar cualquier problema con el rendimiento ocasionalmente lento.

Además, si se prefiere no utilizar un servicio basado en la nube, la alternativa es instalar LibreOffice. Diseñado como una alternativa multiplataforma de código abierto para el popular conjunto de aplicaciones Microsoft Office y basado en el proyecto OpenOffice.org, LibreOffice es poderoso y ofrece tanta funcionalidad como su inspiración de código cerrado.

1.2.1. Raspberry Pi 3 Model B

El Raspberry Pi 3 Model B es el primer modelo de la tercera generación de Raspberry Pi como reemplazo de la Raspberry Pi 2 Model B. En la Figura 1 se muestran los modelos de Raspberry Pi desde el año 2012.

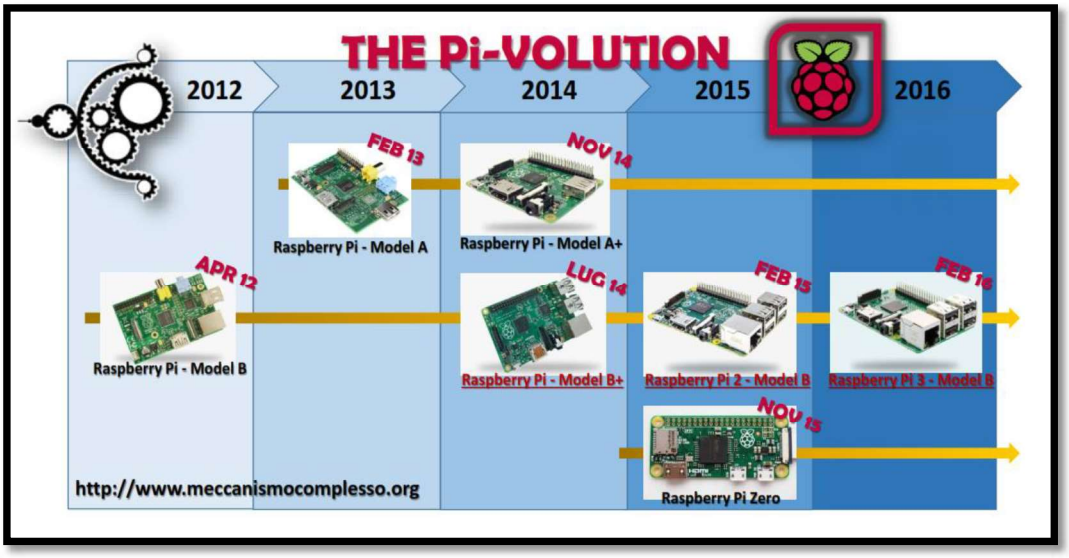

Figura 1. Evolución de los modelos de Raspberry Pi.
(Fuente: https://www.meccanismocomplesso.org/wp-content/uploads/2016/08/Meccanismo-Complesso-The-Pi-volution.pdf)

Posee las siguientes características:

- CPU: Quad Core 1.2GHz Broadcom BCM2837 64bit
- RAM: 1 GB
- LAN inalámbrica : BCM43438
- Bluetooth Low Energy (BLE)
- 100 Base Ethernet
- GPIO extendido: 40 pines
- USB: 4 puertos
- Salida de 4 polos estéreo
- Puerto de video compuesto
- HDMI de tamaño completo
- Puerto de cámara CSI para conectar una cámara Raspberry Pi
- Puerto de visualización DSI para conectar una pantalla táctil Raspberry Pi
- Puerto micro SD para cargar su sistema operativo y almacenar datos
- Fuente de alimentación Micro USB conmutada actualizada de hasta 2.5ª

En la figura 2 se muestra el Raspberry Pi 3 Model B.

Figura 2. Evolución de los modelos de Raspberry Pi.
(Fuente: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/)

USB es uno de los métodos más comunes para conectar periféricos y dispositivos de almacenamiento a una computadora. La Raspberry Pi 3 Model B viene equipada con cuatro de ellas, lo que permite conectar un teclado y un mouse cuando se comienza su uso y un puerto micro USB para alimentar el dispositivo.

Además posee el puerto de interfaz multimedia de alta definición (HDMI) que permite que la Raspberry Pi 3 Model B se conecte a televisores y monitores de alta definición que sean compatibles. Esto proporciona una opción mejorada al puerto RCA que se presentan en versiones anteriores de Raspberry Pi, ya que el puerto HDMI soporta video y audio a la vez. El puerto HDMI también sirve para la transmisión de audio y video desde la web a su televisor.

La Raspberry Pi 3 Modelo B tiene otro puerto importante, el cual es el puerto de la tarjeta SD que se convierte en el principal mecanismo de almacenamiento. En la tarjeta SD es donde se instala el sistema operativo y actúa como el disco duro básico. Por supuesto, este almacenamiento se amplía con el uso de los puertos USB.

También es importante destacar el puerto Ethernet que tiene la Raspberry Pi 3 Model B, ya que es el medio principal de enlace para comunicarse con otros dispositivos e Internet. Se puede usar el puerto Ethernet para conectar la Raspberry Pi a un enrutador doméstico, como el que utiliza actualmente para acceder a Internet, o un conmutador de red si existe uno configurado.

La forma principal de conectarse con otras tarjetas electrónicas como Arduino que tiene la Raspberry Pi 3 Modelo B son los pines de entrada / salida de uso general (GPIO). En

el presente proyecto se utiliza un Arduino Nano como medio de enlace entre la Raspberry Pi 3 Modelo B y el ordenador en el que se encuentra instalado el simulador, por lo que es importante detallar esta característica. Como su nombre lo indica, los pines GPIO pueden aceptar comandos de entrada y salida, los cuales pueden ser programados en la Raspberry Pi 3 Modelo B. Las tarjetas de Arduino se conectarán al GPIO a través de sus pines de entrada y salida de datos, lo que nos permite transferir datos de dispositivos como sensores que se encuentren conectados a la placa de Arduino de vuelta al Raspberry Pi 3 Model B. Esto es especialmente útil en proyectos de automatización del hogar, donde es posible que deseemos almacenar datos de sensores o manipular motores basados en un programa que se ejecute en el sistema operativo de la Raspberry Pi 3 Model B.

1.3. Ubuntu Mate

Después de realizar las pruebas con varios sistemas operativos disponibles, los cuales fueron: Ubuntu Core 16, Ubuntu Classic Server 16.04, Lubuntu 16.04.2, Xubuntu 16.04.2 y Ubuntu Core 18 para Raspberry Pi 3 Model B. El sistema operativo Ubuntu presenta gran compatibilidad de hardware y un amplio número de herramientas de software, se lo puede instalar en ordenadores de escritorio, portátiles y servidores en distribuciones de 32 y 64 bits. El sistema operativo Lubuntu está diseñado para trabajar con limitados recursos de RAM y CPU y optimizar el consumo de energía y tiene disponibles versiones de 32 y 64 bits para ordenadores de escritorio y portátiles. El sistema operativo Xubuntu es similar a Lubuntu, usa el entorno gráfico Unity optimizando recursos de procesamiento y tiene disponibles versiones de 32 y 64 bits para ordenadores de escritorio y portátiles. [26]

Ubuntu MATE 16.04 fue seleccionado como el sistema operativo óptimo para el presente proyecto, ya que fue el sistema operativo que no presentó problemas en su ejecución y lo más importante es que se pudo compilar el componente creado en OROCOS, en otras palabras el Deployer se ejecutó con éxito.

Actualmente existe una imagen de disco que permite ejecutar Raspberry Pi 2 y Raspberry Pi 3 de Ubuntu MATE 16.04 [4]. La estructura y datos contenidos en un medio de almacenamiento como por ejemplo disco duro o un disco óptico, tal es el caso de los CD o DVD pueden ser guardados en un archivo especial, el cual es llamado imagen de disco.

En la Figura 3 se muestra la opción de descarga de Ubuntu MATE para Raspberry Pi 2 o 3.

Ubuntu MATE tiene una serie de características relevantes las cuales son:

- Presenta una accesibilidad independientemente del lenguaje y la capacidad física.
- Aumento de los usuarios que utilizan Ubuntu MATE.
- Alternativa de Ubuntu para computadoras que no son lo suficientemente potentes para ejecutar un escritorio compuesto.
- Plataforma Ubuntu seleccionada para soluciones de estaciones de trabajo remotas como LTSP (Linux Terminal Server Project) y X2Go. X2Go usa un protocolo llamado NX Technology el cual es un software de código libre para escritorios remotos en Linux.
- Recrea un escritorio tradicional de ordenador.
- Usa temas e ilustraciones similares a Ubuntu para que Ubuntu MATE se familiarice de inmediato.
- Cuando es posible, contribuye a Debian para que se beneficien tanto las comunidades de Debian como las de Ubuntu. El proyecto Debian GNU / Linux es uno de los proyectos de Software libre más ambiciosos, que involucra a un gran número de desarrolladores que crean un sistema operativo totalmente gratuito [22].

Los mínimos requisitos de hardware son [17]:

- Pentium M 1.0 gigahertz
- 1 gigabyte (GB) de RAM
- 9 gigabytes (GB) de espacio disponible en el disco duro
- Unidad de DVD-ROM de arranque
- Teclado y mouse (u otro dispositivo señalador)
- Adaptador de video y monitor con resolución de 1024 x 768 o superior
- Tarjeta de sonido
- Altavoces o auriculares

Los requisitos de hardware recomendados son [17]:

- Core 2 Duo 1.6 gigahertz
- Gigabytes (GB) de RAM
- 16 gigabytes (GB) de espacio disponible en el disco duro
- Unidad flash USB de arranque
- Teclado y mouse (u otro dispositivo señalador)
- Adaptador de video compatible con 3D y monitor de pantalla ancha con resolución de 1366 x 768 o superior

- Tarjeta de sonido
- Altavoces o auriculares

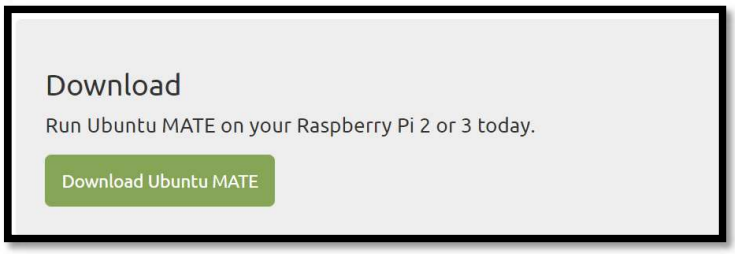

Figura 3. Opción de descarga de Ubuntu MATE para Raspberry Pi 2 o 3.
(Fuente: https://ubuntu-mate.org/raspberry-pi/)

En la figura 4 se muestra las versiones disponibles de Ubuntu MATE según el tipo de arquitectura que se requiera.

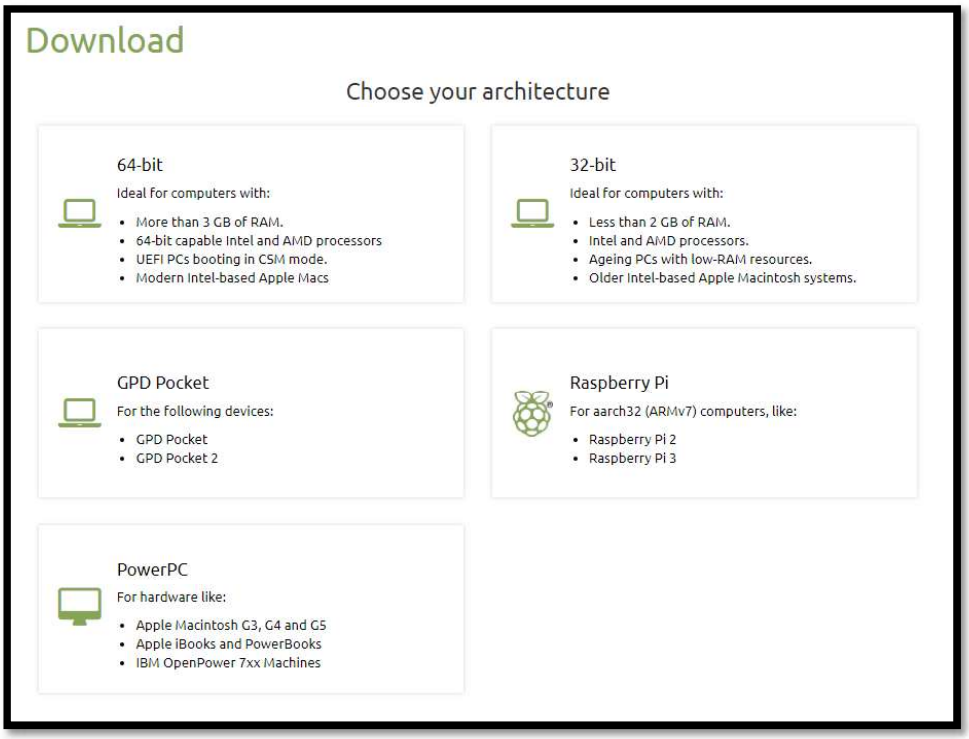

Figura 4. Opción de descarga de las versiones disponibles de Ubuntu MATE.
(Fuente: https://ubuntu-mate.org/download/)

En la Figura 5 se muestra la opción disponible de Ubuntu MATE para Raspberry Pi 2 o 3.

Figura 5. Versión disponible de Ubuntu MATE para Raspberry Pi 2 o 3.
(Fuente: https://ubuntu-mate.org/download/)

1.4. V-REP

En [21] se presenta un estudio de tres simuladores relevantes que son Gazebo, ARGoS y V-REP, donde V-REP sobresale sus características sobre los otros dos simuladores tanto en sus herramientas de comunicación con otros programas por medio de una API (Application Programming Interface) la cual logra la comunicación con V-REP para el control de los robots, esta característica no posee Gazebo y ARGoS. Además por medio de la API "CustomUI" se puede crear interfaces personalizadas para conectar a robot individuales, en Gazebo las interfaces personalizadas solo se pueden adjuntar a toda la escena y no a robots individuales, mientras que en ARGoS las interfaces personalizadas se pueden crear en C ++ mediante una subclasificación de una clase de API de ARGoS y se adjuntan a toda la escena o a robots individuales. V-REP destaca también en las herramientas de programación para la simulación de robots, ya que mediante scripts o complementos se puede programar en otros lenguajes de programación como por ejemplo Python, C / C++, Java, Lua, Matlab u Octave para controlar al robot, en Gazebo se puede programar como complementos de C ++ compilados o como programas ROS y en ARGoS se pueden programar a través de scripts Lua o en C ++.

El simulador de robots V-REP tiene un entorno de desarrollo integrado, basado en una arquitectura de control distribuido: cada objeto o modelo se puede controlar individualmente mediante un script incorporado, un complemento, un nodo ROS o BlueZero, un cliente API remoto o un cliente personalizado. Esto hace que V-REP sea muy versátil e ideal para aplicaciones de múltiples robots. Los controladores se pueden escribir en C / C ++, Python, Java, Lua, Matlab o Octave.

V-REP se utiliza para el desarrollo rápido de algoritmos, simulaciones de automatización de fábricas, creación rápida de prototipos y verificación, educación relacionada con robótica, monitoreo remoto, verificación de seguridad, etc. En la Figura 6 se muestra el entorno de trabajo de V-REP con una aplicación.

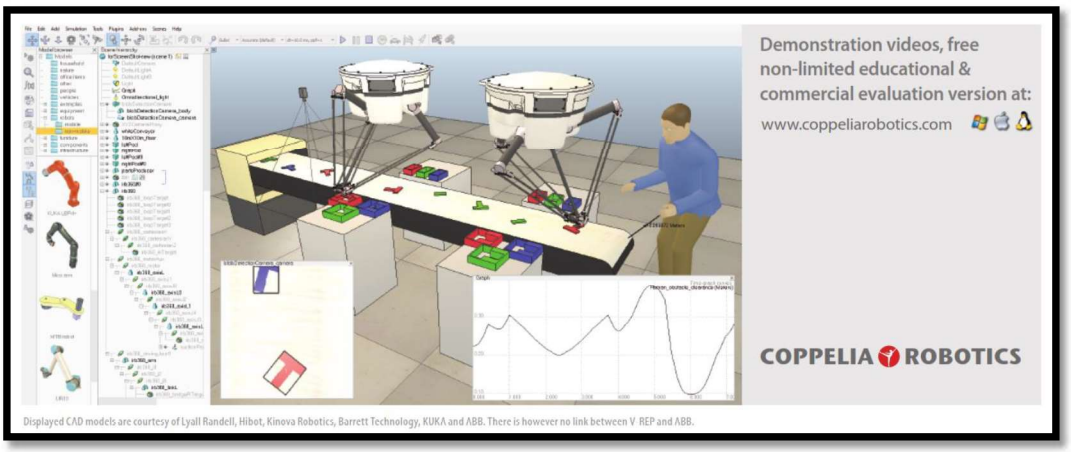

Figura 6. Entorno de V-REP.
(Fuente: http://www.coppeliarobotics.com/assets/v-repspecifications.pdf)

Una característica notable de V-REP es la API remota que está compuesta por aproximadamente cien funciones específicas y una función genérica, que se puede llamar desde una aplicación C / C ++, una secuencia de comandos de Python, una aplicación Java, un programa Matlab / Octave o una secuencia de comandos Lua. Las funciones API remotas interactúan con V-REP a través de la comunicación de socket (u, opcionalmente, a través de la memoria compartida) de una manera que reduce el retraso y la carga de la red en gran medida.

Todo esto sucede de forma oculta para el usuario. La API remota puede permitir que una o varias aplicaciones externas interactúen con V-REP de forma síncrona o asincrónica (asíncrona por defecto), e incluso se admite el control remoto del simulador (por ejemplo, cargar una escena de forma remota, iniciar, pausar o detener una simulación).

A continuación se detalla sobre la API remota y otras características de V-REP que lo hacen un simulador ideal para proyectos de robótica.

1.4.1. Embedded scripts

Representa una característica poderosa y distintiva de V-REP. El bucle principal de simulación es un simple script de Lua (llamado script principal) parte de una escena de simulación dada, que maneja la funcionalidad general (por ejemplo, llamará a diferentes funciones para manejar cinemática o dinámica). El script principal también es responsable de llamar a los scripts secundarios en cascada (con respecto a la jerarquía de la escena).

Un proceso secundario, a diferencia del proceso principal, se adjunta a un objeto específico en la escena de simulación y maneja una parte particular de la simulación.

Es una parte integral de su objeto de escena, y se duplicará y se serializará junto con él. Como tal, representa un elemento de control perfectamente portátil y escalable; hay un solo archivo que contiene la definición del modelo junto con su control o funcionalidad, no hay problemas de compatibilidad entre plataformas, no es necesario compilar explícitamente, no hay conflicto entre ellas, hay varias versiones del mismo modelo, la creación de instancias del modelo es implícita, etc. Los scripts secundarios se pueden ejecutar de forma inalámbrica.

El programador de V-REP maneja los procesos de una manera que los hace comportarse como componentes de programa, que permiten controlar con precisión el momento en que la ejecución del proceso cambia, lo que permite una excelente sincronización con la secuencia de comandos principal u otras secuencias de comandos. Además, cada subproceso puede solicitar secuencialmente que se establezca en modo de ejecución libre (es decir, permitir que se comporte temporalmente como subprocesos reales).

Los scripts incrustados también pueden verse como un "componente de pegamento", que se une a las diversas técnicas de programación compatibles en torno a V-REP. Los scripts secundarios pueden registrar a los editores o suscriptores de ROS, pueden abrir y administrar líneas de comunicación (por ejemplo, socket o puerto serie), iniciar ejecutables, cargar o descargar complementos o iniciar servicios de servidor de API remota. Los scripts incrustados también incluyen scripts de devolución de llamada, utilizados, por ejemplo, como controladores de conjuntos personalizados de bajo nivel. El usuario puede ampliar la funcionalidad de los scripts incorporados a través de dos mecanismos: con las bibliotecas de las extensiones Lua o con las funciones personalizadas de Lua registradas a través de un complemento.

1.4.2. Add-ons

De manera similar a los scripts incrustados, los complementos Add-ons se admiten en V-REP a través de los scripts Lua. Se pueden usar como funciones independientes (convenientes para escribir importadores o exportadores), o como código ejecutado regularmente.

1.4.3. Plug-ins

Los complementos Plug-ins se utilizan en V-REP como una herramienta de personalización de simulador conveniente. Pueden registrar comandos Lua personalizados, lo que permite la ejecución de funciones de devolución de llamada rápida desde un script incrustado. También pueden ampliar la funcionalidad de un

modelo u objeto de simulación en particular. A menudo, también implementan importadores o exportadores específicos u ofrecen una interfaz para un hardware específico. La interfaz API remota y la interfaz ROS se implementan a través de complementos Plug-ins.

1.4.4. Remote API client

Como se mencionó anteriormente, la interfaz API remota permite interactuar con V-REP o una simulación, desde una entidad externa a través de la comunicación de socket. Está compuesto por servidores o clientes de API remotos. En el lado del cliente se puede incrustar como un pequeño código (C / C ++, Python, Java, Matlab y Urbi) en prácticamente cualquier hardware, incluidos los robots reales, y permite la función remota de llamadas, así como la transmisión rápida de datos.

En el lado del cliente, las funciones se llaman casi como funciones regulares, con dos excepciones: las funciones API remotas aceptan un argumento adicional que es el modo de operación y devuelven un mismo código de error. El modo de operación permite las funciones de llamada como bloqueo (esperará hasta que el servidor responda), o no bloqueo (leerá los comandos transmitidos desde un búfer, iniciará o detendrá un servicio de transmisión en el lado del servidor). La facilidad de uso de la API remota, su disponibilidad en todas las plataformas y su tamaño reducido, lo convierten en una alternativa interesante a la interfaz ROS.

1.4.5. Nodos ROS

V-REP implementa un nodo ROS con un complemento que permite a ROS llamar comandos V-REP a través de servicios ROS, o transmitir datos a través de publicadores o suscriptores de ROS. Los editores o suscriptores se pueden habilitar con una llamada de servicio, y también se pueden habilitar directamente desde V-REP, a través de un comando de script incorporado.

1.4.6. Funcionalidad de simulación

V-REP está diseñado alrededor de una arquitectura versátil. No hay funcionalidad principal o central en V-REP, por el contrario, V-REP posee varias funcionalidades relativamente independientes, que pueden habilitarse o deshabilitarse según sea necesario.

Imagine un escenario de simulación en el que un robot industrial tiene que recoger cajas y moverlas a otra ubicación, V-REP calcula la dinámica para agarrar y sostener las cajas y realiza una simulación cinemática para las otras partes del ciclo cuando los efectos

dinámicos son despreciables. Este enfoque permite calcular el movimiento del robot industrial de forma rápida y precisa, lo que no sucedería si se hubiera simulado completamente utilizando bibliotecas de dinámica compleja. Este tipo de simulación híbrida se justifica en esta situación, si el robot es rígido y fijo y no está influenciado por su entorno.

Además de habilitar varias de sus funcionalidades de una manera selectiva, V-REP también puede usarlas de manera asociante, haciendo que una coopere con otra. En el caso de un robot humanoide, por ejemplo, V-REP puede manejar los movimientos de las piernas calculando primero la cinemática inversa para cada pierna (es decir, desde una posición y orientación del pie deseadas, se calculan todas las posiciones de las articulaciones de la pierna) y luego asignando las posiciones conjuntas calculadas para ser utilizadas por el módulo de dinámica. Esto permite especificar el movimiento humanoide de una manera muy versátil, ya que cada pie simplemente debería asignarse para seguir una trayectoria, el resto de los cálculos se realizan automáticamente.

1.4.7. Objetos de escena

Una escena o modelo de simulación de V-REP, contiene varios objetos de escena u objetos elementales que se ensamblan en una jerarquía en forma de árbol. Los objetos de escena que son compatibles con V-REP son las juntas, las formas, los sensores de proximidad, sensores de visión, sensores de fuerza, gráficos, cámaras, luces, rutas, maniquíes, molinos; los cuales se detallan a continuación.

Juntas: las uniones son elementos que unen dos o más objetos de escena con uno a tres grados de libertad (prismático, revoluta, similar a un tornillo o esférico). Pueden operar en varios modos (por ejemplo, modo de fuerza o torsión, modo de cinemática inversa, etc.)

Formas: las formas son mallas triangulares, utilizadas para la simulación y visualización de cuerpos rígidos. Se pueden optimizar para un rápido cálculo dinámico de respuesta de colisión, como una agrupación de formas primitivas o convexas. Otros objetos de escena o módulos de cálculo dependen en gran medida de las formas para sus cálculos (por ejemplo, sensores de proximidad, el módulo de dinámica o el módulo de cálculo de distancia de malla-malla).

Sensores de proximidad: realizan un cálculo de la distancia mínima exacta en un volumen de detección configurable, en lugar de realizar simplemente la detección basada en rayos. Esto resulta en una operación más continua y por lo tanto permite una simulación más realista.

Sensores de visión: los sensores de visión permiten extraer información de imágenes complejas de una escena de simulación (colores, tamaños de objetos, mapas de

profundidad, etc.). Una función incorporada de filtrado y procesamiento de imágenes permite la composición de bloques de elementos de filtro. Los sensores de visión utilizan la aceleración de hardware para la adquisición de imágenes sin procesar.

Sensores de fuerza: representan enlaces rígidos entre formas, que pueden registrar fuerzas y pares aplicados, y que pueden romperse condicionalmente cuando se sobrepasa un umbral determinado.

Gráficos: los gráficos pueden registrar una gran variedad de datos predefinidos o personalizados. Los datos se pueden mostrar directamente (gráfico de tiempo de un tipo de datos determinado), o combinados entre sí para mostrar gráficos de X e Y, o curvas 3D.

Cámaras: permiten la visualización de escenas cuando se asocian con una ventana gráfica.

Luces: las luces iluminan una escena u objetos de escena individuales, e influyen directamente en las cámaras o sensores de visión.

Rutas: permiten definiciones complejas de movimientos en el espacio (sucesión de traducciones, rotaciones y/o pausas libremente combinables). Se utilizan por ejemplo para guiar la antorcha de un robot de soldadura a lo largo de una trayectoria predefinida, o para permitir movimientos de la banda transportadora.

Maniquíes: un maniquí es un marco de referencia, que se puede usar para varias tareas, y se usa principalmente en conjunto con otros objetos de escena, se puede ver como una ayuda para la simulación.

Molinos: representan volúmenes convexos personalizables que se pueden usar para simular operaciones de corte de superficie en formas (por ejemplo, fresado, corte por láser, etc.).

1.4.8. Módulos de cálculo

Los objetos de escena rara vez se usan solos, sino que operan en conjunto con otros objetos de escena (por ejemplo, un sensor de proximidad detectará formas). Además, V-REP ofrece varios módulos de cálculo que pueden operar directamente en uno o varios objetos de escena. El módulo de cinemática, dinámica, de detección de colisión, de cálculo de distancia malla-malla, de planificación de ruta o movimiento, son los principales y se detallarán a continuación.

Módulo de cinemática: permite cálculos cinemáticos para cualquier tipo de mecanismo (ramificado, cerrado, redundante, que contenga bucles anidados, etc.). El módulo se basa en el cálculo de los mínimos cuadrados. Es compatible con resolución condicional, amortiguada o no amortiguada y ponderada.

Módulo de dinámica: permite manejar la interacción y el cálculo de dinámica de cuerpos rígidos (respuesta de colisión, agarre, etc.). Las simulaciones basadas en la dinámica a menudo son basadas en aproximaciones, es importante no solo confiar en un solo tipo de objeto de simulación para validar los resultados.

Módulo de detección de colisión: permite la comprobación rápida de interferencias entre cualquier forma o conjunto de formas. Este módulo es totalmente independiente del algoritmo de cálculo de respuesta de colisión del módulo de dinámica. Utiliza estructuras de datos binarios con límites orientados para las aceleraciones.

Módulo de cálculo de distancia malla-malla: permite cálculos de distancia mínima entre cualquier forma o conjunto de formas (convexa, cóncava, abierta, cerrada, etc.). El módulo utiliza las mismas estructuras de datos que el módulo de detección de colisiones.

Módulo de planificación de ruta o movimiento: maneja las tareas de planificación de ruta holonómica y no holonómica (para vehículos similares a automóviles). También se admiten tareas de planificación de rutas de cadenas cinemáticas.

Para mayor versatilidad, los módulos anteriores se implementan de una manera general, sin hacer cálculos en las escenas o modelos de simulación subyacentes. El propósito de tenerlos integrados en V-REP, en lugar de depender de bibliotecas externas es algo similar al de tener scripts incrustados.

Una gran mayoría de simulaciones o modelos de simulación no requieren ninguna información específica o herramienta de alta gama. En su lugar, requieren un buen conjunto de herramientas básicas. Si están integrados en el simulador y sus definiciones de tareas están directamente relacionadas con los modelos de simulación, entonces los modelos se vuelven extremadamente portátiles; la distribución de un modelo de simulación a una máquina o plataforma diferente se realiza a través de un archivo de modelo único; no es necesario distribuir, recompilar, instalar o volver a cargar un complemento. De manera similar, esto también hace que los modelos sean muy escalables, los modelos duplicados son automáticamente funcionales, sin la necesidad de modificarlos.

El enfoque tradicional de extender la funcionalidad a través de un complemento, con el fin de soportar un modelo de simulación específico, también es compatible con V-REP.

1.4.9. Escena del péndulo invertido en V-REP

La aplicación seleccionada en el presente trabajo es la de un péndulo invertido, el cual fue escogido por ser uno de los problemas más importantes y clásicos de la teoría de control. Se trata de un control inestable y no lineal. A menudo, es utilizado como ejemplo académico, principalmente por ser un sistema de control más accesible [23].

La escena de [13] de V-REP del péndulo invertido utilizada en el presente proyecto consta de cuatro formas, la primera es un prisma de masa 30 kg que corresponde al cuerpo del péndulo con dimensiones de X=0.45[m], Y=0.5[m] y Z=0.1[m], la segunda y tercera son cilindros de masa 10[kg] de radio de 0.25[m] y de espesor de 0.1[m] que corresponden a las ruedas y la cuarta es un prisma de masa 70 kg que corresponde a la carga del péndulo con dimensiones de X=0.2[m], Y=0.4[m] y Z=1.5[m]. También posee tres juntas, la primera y la segunda corresponden a las uniones de las ruedas con el cuerpo del péndulo, en otras palabras representan los motores que mueven a las ruedas, las cuales tienen una velocidad angular de 57.296 [deg/s] y un torque máximo de 100[Nm], y la tercera corresponde a la unión del cuerpo con la carga, representa otro motor que tiene un torque máximo de 50[Nm] el cual está bloqueado para velocidad angular igual a cero. Además tiene dos gráficos que presentan los datos del torque y la velocidad angular en función del tiempo.

Para el funcionamiento de la simulación del péndulo invertido en V-REP se deberá realizar las instalaciones del sistema operativo Ubuntu MATE, ROS (Robot Operating System) el cual proveerá librerías y herramientas para desarrollar la aplicación del presente proyecto en mencionada instalación se provee el soporte para que se pueda utilizar el lenguaje de programación PYTHON y se deberá instalar OROCOS TOOLCHAIN la cual es la herramienta principal para crear aplicaciones de robótica en tiempo real utilizando componentes de software configurables. Las instalaciones se realizarán tanto en la Raspberry Pi 3 Model B como en la máquina virtual, adicionalmente V-REP debe ser ejecutado en la máquina virtual. Dichas instalaciones se de tallan más adelante.

1.5. Instalación de Ubuntu MATE, ROS, PYTHON y OROCOS TOOLCHAIN en la Raspberry Pi

Ubuntu MATE 16.04 es el sistema operativo donde se desarrolló la instalación de ROS, Python y OROCOS TOOLCHAIN. Las versiones probadas sin éxito por razones de compatibilidad del sistema operativos con la tarjeta Raspberry Pi 3 Model B o ya sea por la incompatibilidad de OROCOS con el sistema operativo cargado en la tarjeta, fueron Ubuntu Core 16, Ubuntu Classic Server 16.04, Lubuntu 16.04.2, Xubuntu 16.04.2 y Ubuntu Core 18. Como se presenta en la sección de pruebas, en las versiones mencionadas anteriormente no se logró ejecutar el Deployer e incluso algunos sistemas operativos no se ejecutaron.

1.5.1. Ubuntu MATE

De la página web oficial de Ubuntu MATE, se debe seleccionar Ubuntu MATE para la arquitectura de Raspberry Pi que pertenecen a las computadoras aarch32 (ARMv7), la versión que se encuentra disponible para descargar es la 16.04.2(Xenial). La opción que se debe descargar es el que contiene la imagen de disco en un archivo con extensión .xz como fichero comprimido.

Con la ayuda de una herramienta gráfica de escritura de tarjetas SD que soporte la escritura de imágenes de disco desde archivos comprimidos, como es el caso de balenaEtcher, se graba el sistema operativo Ubuntu MATE descargado en una Micro SD.

Una vez que en balenaEtcher se ha seleccionado la imagen de disco, la unidad respectiva de almacenamiento y ha culminado el proceso de grabación, aparecerá el mensaje de indicando que el sistema operativo ha sido grabado con éxito en la micro SD.

La Micro SD con Ubuntu MATE debe ser insertada en la Raspberry Pi 3 Model B, la cual al ser conectada a su fuente de alimentación comenzará el proceso de instalación del sistema operativo. Lo primero que empezará a realizar la Raspberry Pi 3 Model B es a redimensionar la partición raíz. Luego el sistema operativo se empezará a configurar. Ciertas configuraciones se las realiza de forma manual, la primera selección que se debe realizar es el del idioma, en la cual seleccionamos en Español, en la segunda pedirá la configuración de la red inalámbrica que es conveniente realizarla en el momento de la instalación del sistema operativo, en la tercera selección deberá seleccionar la ubicación geográfica, en la cuarta se debe configurar el teclado lo que es conveniente Español(Latinoamericano), en la última configuración pedirá su nombre, nombre de equipo, nombre de usuario y contraseña; la contraseña será la misma que se ingrese para instalar cualquier programa en la Raspberry Pi 3 Model B. Cuando se ha terminado de realizar las configuraciones manuales, se aplicarán los cambios y si el proceso ha sido exitoso, aparecerá el escritorio de Ubuntu MATE.

1.5.2. ROS y PYTHON

El procedimiento de instalación de ROS Kinetic se encuentra en la página oficial de ROS como se indica en la Figura 7.

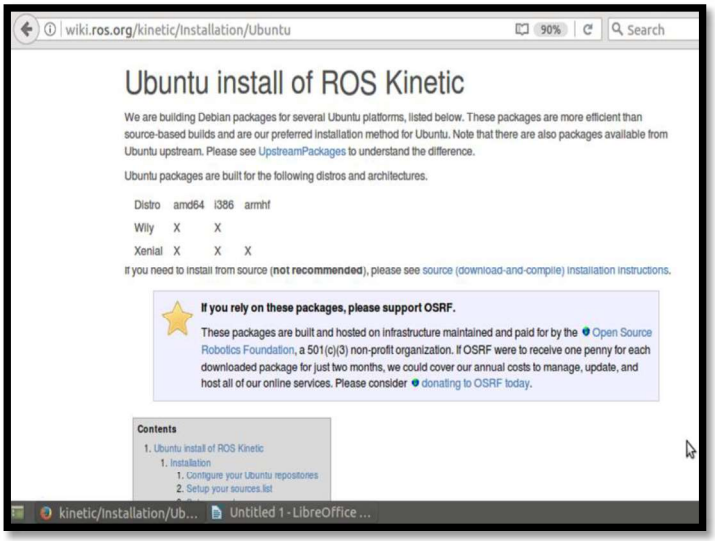

Figura 7. Instalación de ROS Kinetic para Ubuntu.
(Fuente: http://wiki.ros.org/kinetic/Installation/Ubuntu)

El inicio de la instalación consiste en la configuración en la Raspberry Pi 3 Model B para aceptar software de packages.ros.org. y de la contraseña. El siguiente paso es actualizar el índice de paquetes Debian. Realizadas las anteriores configuraciones se ejecuta la instalación completa de ROS en la Raspberry Pi 3 Model B con la instrucción que se presenta en la Figura 8.

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

Figura 8. Instrucción para la instalación completa de ROS en la Raspberry Pi 3 Model B.
(Fuente: http://wiki.ros.org/kinetic/Installation/Ubuntu)

Para continuar con la instalación se debe introducir la instrucción para encontrar paquetes disponibles.

Para utilizar ROS, previamente se debe inicializar de Rosdep, el cual permite instalar las dependencias del sistema para la fuente que desea compilar y además se instala componentes centrales en ROS.

Es conveniente configurar la instalación para que las variables de entorno de ROS se agreguen automáticamente a su sesión de bash cada vez que se lanza un nuevo shell. Con el proceso anterior se puede ejecutar los paquetes ROS principales, pero para crear y administrar sus propios espacios de trabajo ROS, existen diversas herramientas y requisitos que se distribuyen por separado para lo cual se ingresa la instrucción de la Figura 9, con la cual finalizaremos las instrucciones de instalación.

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

Figura 8. Instrucción para instalar herramientas y dependencias que permiten construir paquetes ROS en la Raspberry Pi 3 Model B.
(Fuente: http://wiki.ros.org/kinetic/Installation/Ubuntu)

1.5.3. OROCOS

En primer lugar se ejecuta la instrucción para instalar el paquete rtt_ros_integration [7] se lo muestra en la Figura 9.

```
$ sudo apt-get install ros-jade-rtt-ros-integration
```

Figura 9. Instrucción para instalar el paquete rtt_ros_integration en la Raspberry Pi 3 Model B.
(Fuente: https://github.com/orocos/rtt_ros_integration)

En el código de la Figura 9. se debe realizar la modificación correspondiente según la versión de ROS que se haya instalado.
En el espacio de trabajo que se lo crea de forma manual, se puede crear un componente con las instrucciones de Figura 10 que se lo debe escribir en un terminal.

```
$ cd ~/catkin_ws/src
$ rosrun rtt_ros orocreate-pkg my_component component
```

Figura 10. Instrucción para crear el componente en la Raspberry Pi 3 Model B.
(Fuente: https://github.com/orocos/rtt_ros_integration)

En la instrucción de la Figura 10 se debe reeemplazar el nombre del componente que queremos crear por my_component. Cuando se realiza este proceso, en el espacio de trabajo se creará una carpeta con el nombre del componente.
Para confirmar que el proceso de creación de creación del componente fue exitoso se ejecuta la instrucción de la Figura 11, con lo cual en el terminal debe aparecer el mensaje en el cual indique que se ha completado el 100% de la construcción del componente.

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

Figura 10. Instrucción para verificar la construcción del componente en la Raspberry Pi 3 Model B.

(Fuente: https://github.com/orocos/rtt_ros_integration)

En la Figura 11 se presenta la instrucción para ejecutar el deployer, la cual confirma que el componente está listo para ser ejecutado.

```
rosrun rtt_ros deployer /directory/to/my/script/my_script.ops
```

Figura 11. Instrucción para ejecutar el deployer en la Raspberry Pi 3 Model B.
(Fuente: https://github.com/orocos/rtt_ros_integration)

En la instrucción de la Figura 11 se debe ingresar la dirección para poder acceder al archivo con extensión de .ops del componente creado.

1.6. Instalación de Ubuntu, ROS, PYTHON, V-REP y OROCOS TOOLCHAIN en la Máquina Virtual.

1.6.1. Ubuntu

La imagen de disco de Ubuntu MATE 16.04.2 se instala en la máquina virtual VMware como se indica en la Figura 12.

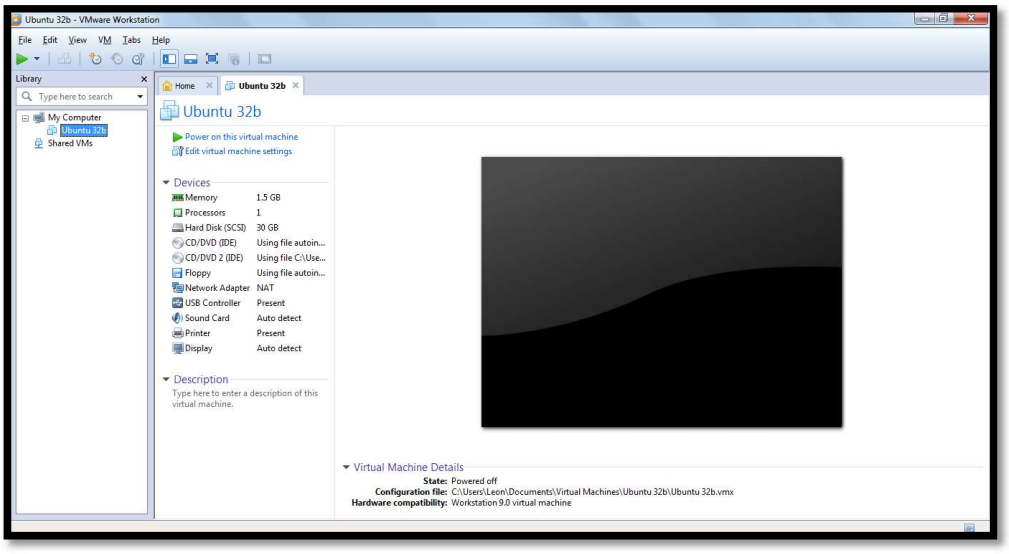

Figura 12. Imagen de disco de Ubuntu 16.04 LTS instalado en la máquina virtual VMware Workstation.
(Fuente: Francisco Álvarez 2018)

1.6.2. ROS, PYTHON y OROCOS

El procedimiento es el mismo que se realizó en la Raspberry Pi 3 Model B.

1.6.3. V-REP

En comparación con los simuladores Gazebo y ARGoS, V-REP es el más complejo y el que más recursos consume de los tres simuladores. Ofrece una serie de características útiles, como una biblioteca de modelos completa, la capacidad de un usuario para interactuar con el mundo durante la simulación y, lo más importante, la manipulación y optimización de mallas. [21]

Además, V-REP genera automáticamente nuevos subprocesos en múltiples núcleos de CPU y, por lo tanto, utiliza la cantidad total de potencia de CPU cuando es necesario. Por lo tanto, es adecuado para el modelado de alta precisión de aplicaciones robóticas como el transporte de objetos o la vigilancia de área, así como para varias aplicaciones industriales, donde solo se requieren unos pocos robots para operar al mismo tiempo. [21]

V-REP tiene varias opciones para la funcionalidad de programación, incluyendo scripts adjuntos a robots, complementos, nodos ROS o programas separados que se conectan a V-REP a través de RemoteAPI. [21]

Para ejecutar el programa, de la página oficial de Coppelia Robotics V-REP se descarga la versión V3.3.2 para Linux de V-REP, se descomprime la carpeta, se ejecuta un terminal en la dirección correspondiente y se escribe ./vrep.sh para inicializar el programa.

El espacio de trabajo aparecerá luego ejecutar la instrucción de inicio. El terminal que se ejecutó debe permanecer abierto para cargar o crear la escena que se requiera.

Una vez instalados los programas necesarios para la simulación y ejecutada la escena del péndulo invertido, se procede a la selección del control para el péndulo invertido. El control PID fue seleccionado para el presente trabajo ya que es un controlador muy robusto y la dinámica del péndulo invertido permite utilizar este tipo de control. El control PID es utilizado cada vez más en aplicaciones industriales [23].

En los siguientes capítulos se detalla el diseño del controlador y los resultados obtenidos.

2. DISEÑO DEL CONTROLADOR PARA EL PÉNDULO INVERTIDO VIRTUAL

2.1. Modelo del péndulo invertido

En la Figura 13 se presenta el péndulo simple, el cual presenta los parámetros que se tomarán en cuenta para analizar el sistema mecánico del mismo, los cuales son la masa del péndulo (m_p), la masa de la base (m_c), el ángulo de desviación (θ), la longitud del punto medio del péndulo (L) y la fuerza que mantiene al sistema en equilibrio (F).

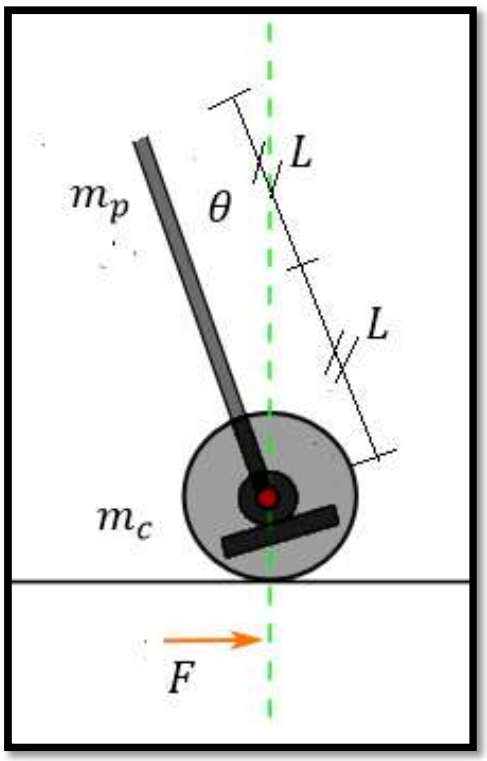

Figura 13. Sistema del péndulo.
(Fuente: Francisco Álvarez 2018)

En la Figura 14 se presenta el diagrama de cuerpo libre de la base que nos permitirá analizar todas las fuerzas que se ejercen sobre ella, las cuales son la fuerza que el péndulo ejerce sobre la base ($\vec{T}$), la fuerza que ejerce la superficie sobre la base ($\vec{N}$), el peso de la base ($m_c * \vec{g}$) y la fuerza para mantener el equilibrio del sistema ($\vec{F}$).

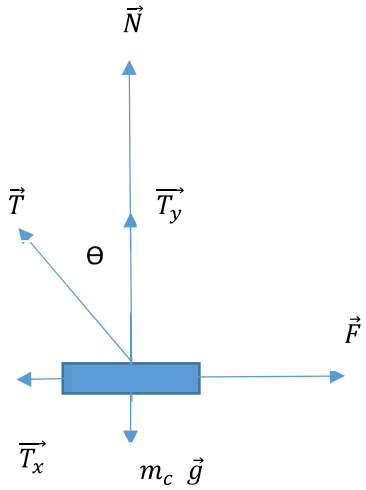

Figura 14. Diagrama de cuerpo libre de la base.
(Fuente: Francisco Álvarez 2018)

En la Figura 15 se presenta el diagrama de cuerpo libre del péndulo, que tiene a la fuerza que ejerce sobre la base ($\vec{T}$) y su peso ($m_p * \vec{g}$)

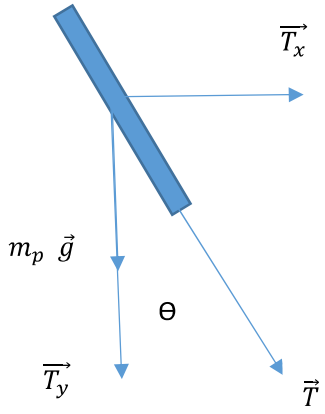

Figura 15. Diagrama de cuerpo libre del péndulo.
(Fuente: Francisco Álvarez 2018)

Para el diagrama de cuerpo libre de la Figura 15, se presenta el siguiente análisis dinámico.

Sumatorio de fuerzas en el *eje x*, donde a_{px} es la aceleración en el *eje x*.

$$\sum F_x = m_p * a_{px}$$

$$T_x = m_p * a_{px}$$

$$T * sin\,\theta = m_p * a_{px} \quad \text{Ec. (1)}$$

Sumatorio de fuerzas en el *eje y*, donde a_{py} es la aceleración en el *eje y*.

$$\sum F_y = m_p * a_{py}$$

$$-T_y - m_p * g = m_p * a_{py}$$

$$-T * cos\,\theta - m_p * g = m_p * a_{py} \quad \text{Ec. (2)}$$

La aceleración del péndulo respecto a la de la base ($a_{p/c}$) es igual a la aceleración del péndulo (a_p) menos la aceleración de la base (a_c)

$$a_{p/c} = a_p - a_c$$

$$a_p = a_c + a_{p/c}$$

De forma vectorial la aceleración del péndulo ($\overrightarrow{a_p}$) se presenta de la siguiente forma:

$$\overrightarrow{a_p} = \ddot{x}\,\vec{\imath} + \left[L\,\ddot{\theta}\,\vec{e}_{tangencial} + L\,\dot{\theta}^2\,\vec{e}_{normal}\right]$$

$$\overrightarrow{a_p} = \ddot{x}\,\vec{\imath} + L\ddot{\theta}\,[-cos\,\theta\;\vec{\imath} - sin\,\theta\,\vec{\jmath}\,] + L\,\dot{\theta}^2[sin\,\theta\;\vec{\imath} - cos\,\theta\;\vec{\jmath}]$$

Reemplazando $\overrightarrow{a_p}$ en Ec. (1) y Ec. (2)

$$T * sin\,\theta = m_p * \ddot{x} - m_p * L * \ddot{\theta} * cos\,\theta + L * \dot{\theta}^2 * sin\,\theta * m_p \quad \text{Ec. (3)}$$

$$-T * cos\,\theta - m_p * g = -m_p * L * \ddot{\theta} * sin\,\theta - L * \dot{\theta}^2 * cos\,\theta * m_p \quad \text{Ec. (4)}$$

Realizamos la siguiente operación matemática Ec. (3)* $cos\,\theta$ + Ec. (2)* $sin\,\theta$

$$-sin\,\theta * m_p * g = m_p * \ddot{x} * cos\,\theta - m_p * L * \ddot{\theta} \quad \text{Ec. (5)}$$

Para el diagrama de cuerpo libre de la Figura 14, se presenta el siguiente análisis dinámico.

Sumatorio de fuerzas en el *eje x*, donde a_x es la aceleración en el *eje x*.

$$\sum F_x = m_c * a_x$$

$$F - T_x = m_c * a_x$$

$$F - T * sin\,\theta = m_c * a_x = m_c * \ddot{x} \qquad \text{Ec. (6)}$$

Reemplazo de Ec. (3) en Ec. (6)

$$F + m_p * L * \ddot{\theta} * cos\,\theta - L * \dot{\theta}^2 * sin\,\theta * m_p = (m_c + m_p) * \ddot{x} \qquad \text{Ec. (7)}$$

Para la función de transferencia se asume $\theta * \dot{\theta} = 0$, $sin\,\theta = \theta$, $cos\,\theta = 1$, por lo que se realizará mencionado reemplazo en Ec. (5) y Ec. (6)

$$\ddot{x} = -\theta * g + L * \ddot{\theta} \qquad \text{Ec. (8)}$$

$$F = \left(m_c + m_p\right) * \ddot{x} - m_p * L * \ddot{\theta} \qquad \text{Ec. (9)}$$

Reemplazo de Ec. (8) en Ec. (9)

$$F = \left(-m_c - m_p\right) * \theta * g + m_c * L * \ddot{\theta} \qquad \text{Ec. (10)}$$

Aplicando la Transformada de Laplace en Ec. (10)

$$F(s) = -\left(m_c + m_p\right) * \theta(s) * g + m_c * L * s^2 * \theta(s) \qquad \text{Ec. (10)}$$

$$\frac{\theta(s)}{F(s)} = \frac{1}{m_c * L * s^2 - \left(m_c + m_p\right) * g}$$

Donde: $m_c = 30kg, L = 4m, m_p = 70kg$

2.2. Diseño del controlador

Con el modelo lineal del sistema se ingresan los datos de la planta como se indica en la Figura 16 en un Scrip de Matlab para determinar los datos del PID. Se analiza la respuesta a una entrada escalón unitario para identificar el método más adecuado para hallar las constantes del controlador PID.

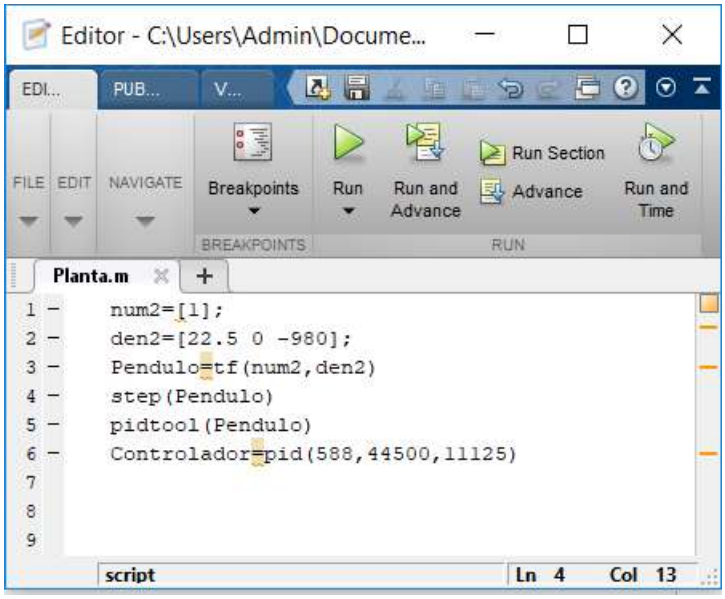

Figura 16. Scrip de Matlab con el modelo lineal de la planta.
(Fuente: Francisco Álvarez 2018)

En la Figura 17 se muestra la respuesta a una entrada escalón unitario de la planta en lazo abierto. Como se observa no es posible utilizar el primer método de la regla de sintonía de Ziegler-Nichols, como la curva no es en forma de S, no se puede calcular el tiempo de retardo y la constante de tiempo. [19]

Figura 17. Respuesta a la entrada escalón unitario de la planta.
(Fuente: Francisco Álvarez 2018)

Por lo tanto se utiliza el segundo método de la regla de sintonía de Ziegler-Nichols, que consiste en usar únicamente la acción de control proporcional y se halla la constante K_{cr}, cuyo valor provoca en la salida oscilaciones sostenidas. Una vez que se obtiene K_{cr}, se calcula el periodo P_{cr}. [19]

En la Figura 18 se muestra que con el valor de $K_{cr} = 980$ se obtienen en la salida oscilaciones sostenidas y el $P_{cr} = 0.89x10^5$[seg].

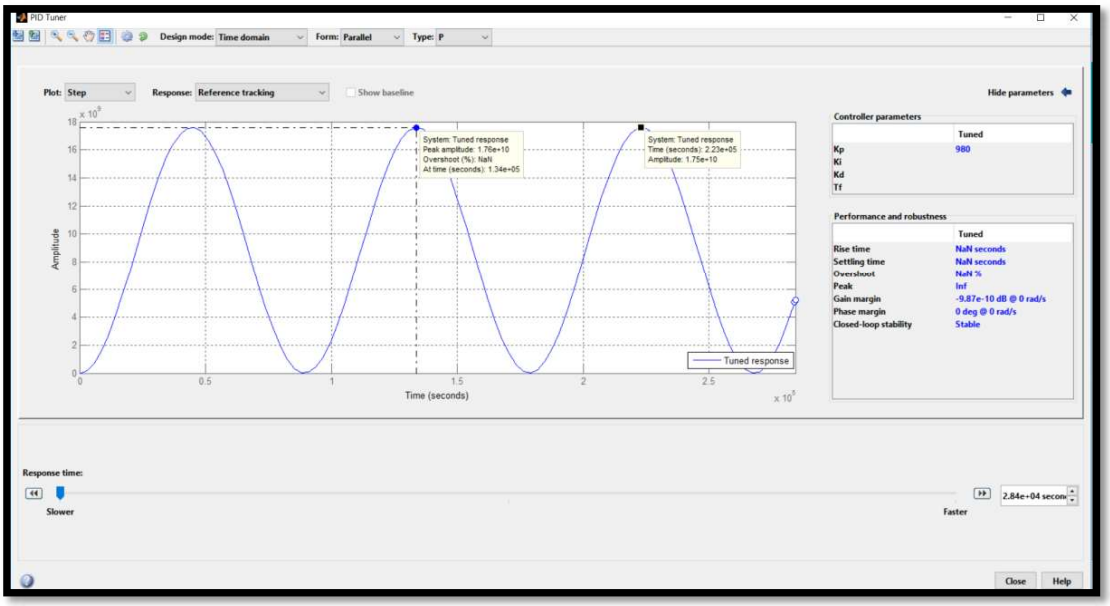

Figura 18. Respuesta a la entrada escalón unitario con una acción de control proporcional.
(Fuente: Francisco Álvarez 2018)

En la Figura 19 se presenta la regla de sintonía de Ziegler-Nichols basada en K_{cr} y P_{cr}, la cual se utiliza para calcular las constantes del controlador.

Tipo de controlador	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Figura 19. Segundo método de la regla de sintonía de Ziegler-Nichols.
(Fuente: Ogata, K., (2010). Ingeniería de Control Moderna, México: PEARSON)

Para el caso de nuestra planta, se utilizó un controlador PID y se obtuvieron las siguientes constantes: $K_p = 588$, $T_i = 44500$ y $T_d = 11125$, con las cuales se obtuvo una respuesta con un tiempo de estabilización de 0.00786[seg] y un máximo sobreimpulso de 0.000574%, como se muestra en la Figura 20, en la cual la curva de color azul es la respuesta con el controlador calculado.

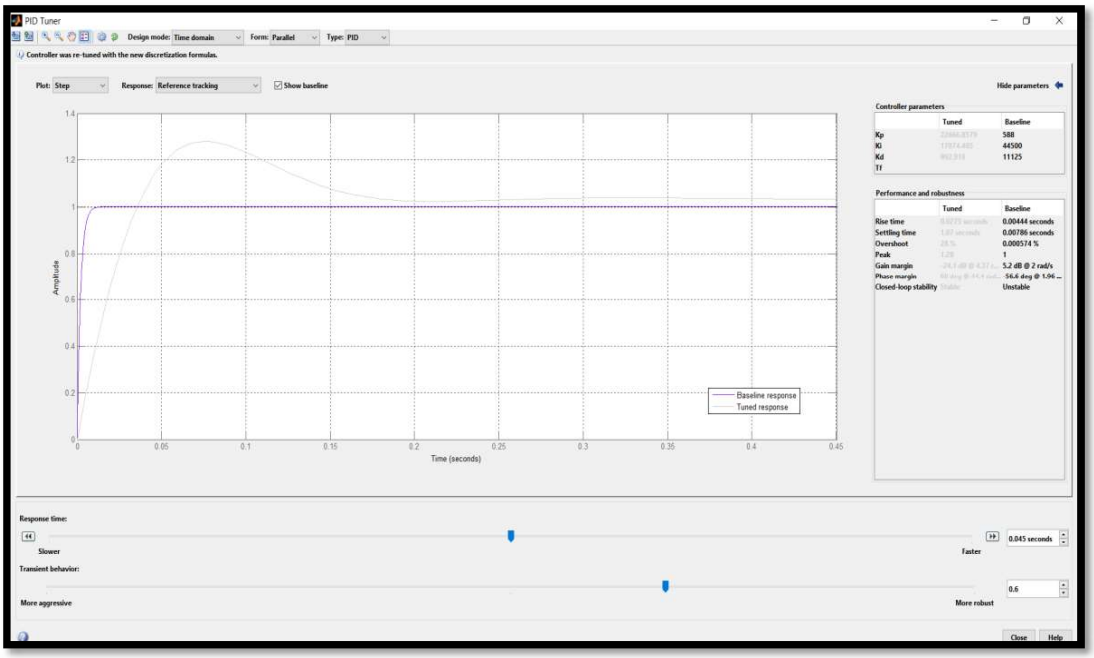

Figura 20. Respuesta de la planta con el controlador PID.
(Fuente: Francisco Álvarez 2018)

2.3. Integración de los sistemas

La integración de los sistemas se la realiza principalmente por ficheros los cuales integran OROCOS con PHYTON y luego PHYTON con V-REP. Con el propósito de ejecutar con éxito el Deployer y realizar una aplicación, en este caso, la simulación del control del péndulo invertido.

En la Figura 21 se muestra el código para la conexión entre OROCOS y Python, por medio de un archivo de texto denominado arranque.txt.

```
2 ingreso=0
3 ingreso2=5
4 while ingreso==0:
5   f=open("arranque.txt")
6   dato=f.read()
7   dato=dato[:1]
8   if dato=="1":
9    ingreso=1
10  print dato
11  f.close()
12  ingreso2=ingreso2-1
13  print ingreso2
```

Figura 21. Enlace entre OROCOS y Python.
(Fuente: Francisco Álvarez 2018)

Al ejecutar el archivo de enlace entre OROCOS y Python, se enviará la información a un Arduino Nano, el cual esperará por la instrucción de OROCOS de la Figura 22, que le permitirá por comunicación serial ejecutar el control para que la simulación en V-REP comience a funcionar.

pendulo-component.cpp

```
20 }
21
22 void Pendulo::updateHook(){
23   ofstream fs("arranque.txt");
24   fs<<"1"<<endl;
25   fs.close();
26   ifstream fichero;
27   fichero.open("ficheroSalida.txt");
28   fichero.getline(dato,90);
29   fichero.close();
30   std::cout << "arrancado=" << dato <<std::endl;
31 }
32
```

Figura 22. Instrucción del componente.
(Fuente: Francisco Álvarez 2018)

En la Figura 23 se muestra el controlador implementado en Phyton, con la ayuda del algoritmo proporcionado en [13].

```
class PID:

    def control():

        Kp = 588
        Td = 11125
        Ti = 44500
        control = Kp + Ti + Td

        return control
```

Figura 23. Implementación del controlador.
(Fuente: Francisco Álvarez 2018)

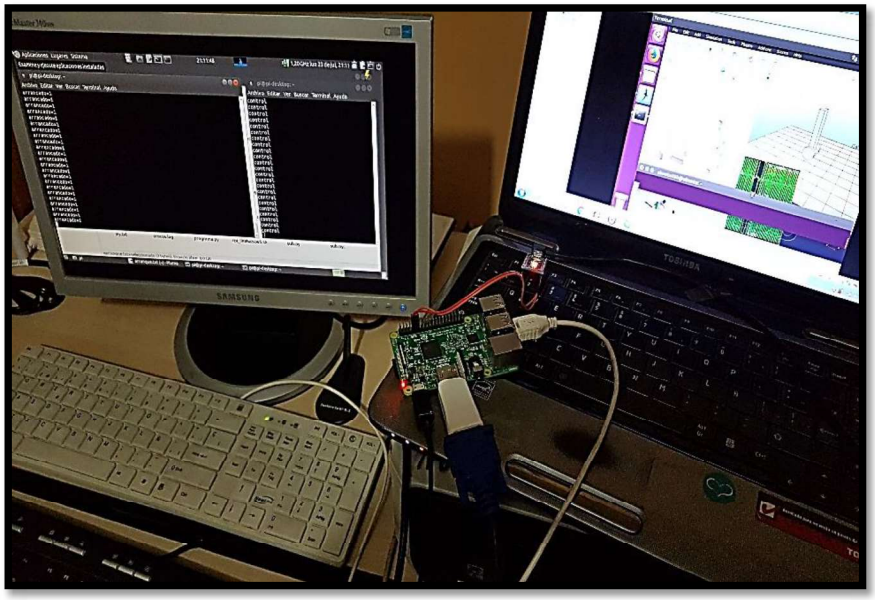

Figura 24. Implementación del sistema completo.
(Fuente: Francisco Álvarez 2018)

2.4. Simulación del controlador utilizando V-REP

Para poner a prueba el controlador PID diseñado, se lo implementará en la escena de V-REP del péndulo invertido para visualizar la estabilidad del péndulo invertido.

Para la ejecución del controlador del archivo de Python, se debe crear la API de enlace para lo cual se debe copiar los archivos remoteApi.so, vrep.py y vrepConst.py de la carpeta de V-REP, en el directorio donde se encuentra el archivo con el controlador.

Luego se ejecutará el entorno de V-REP y se abrirá la escena de [13] que se indica en la Figura 25.

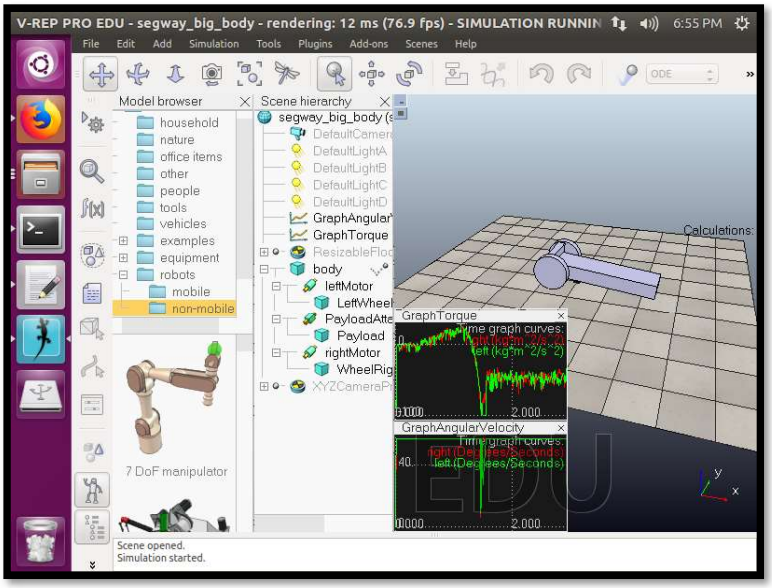

Figura 25. Escena del péndulo.
(Fuente de la imagen: Francisco Álvarez 2018, Fuente de la escena https://github.com/famalgosner/roboticsControl)

Cuando el controlador quiera ejecutarse se producirá el error de la Figura 26, para lo cual se tiene que habilitar el puerto serial que contiene la información que viene de la Raspberry Pi 3 Model B y que pasa por el Arduino Nano.

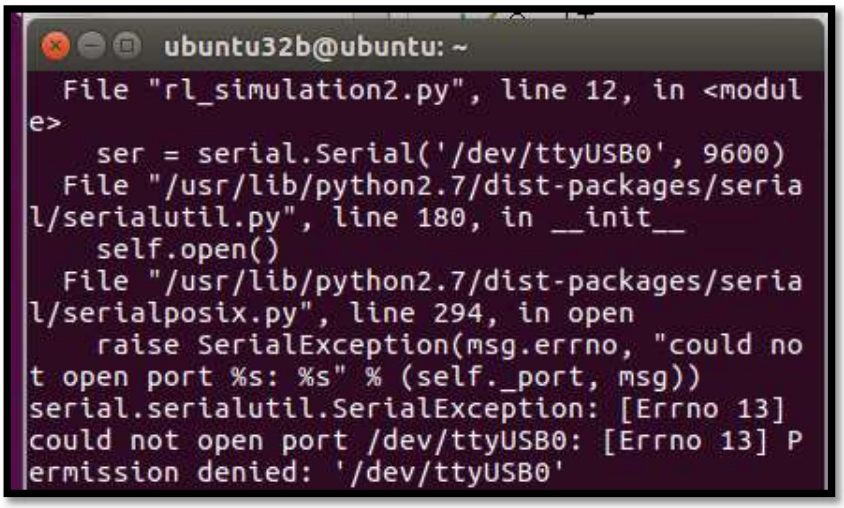

Figura 26. Error previo al enlace con la comunicación serial.
(Fuente: Francisco Álvarez 2018)

Una vez que la comunicación serial se ha establecido el controlador comenzará a funcionar como se muestra en la Figura 27.

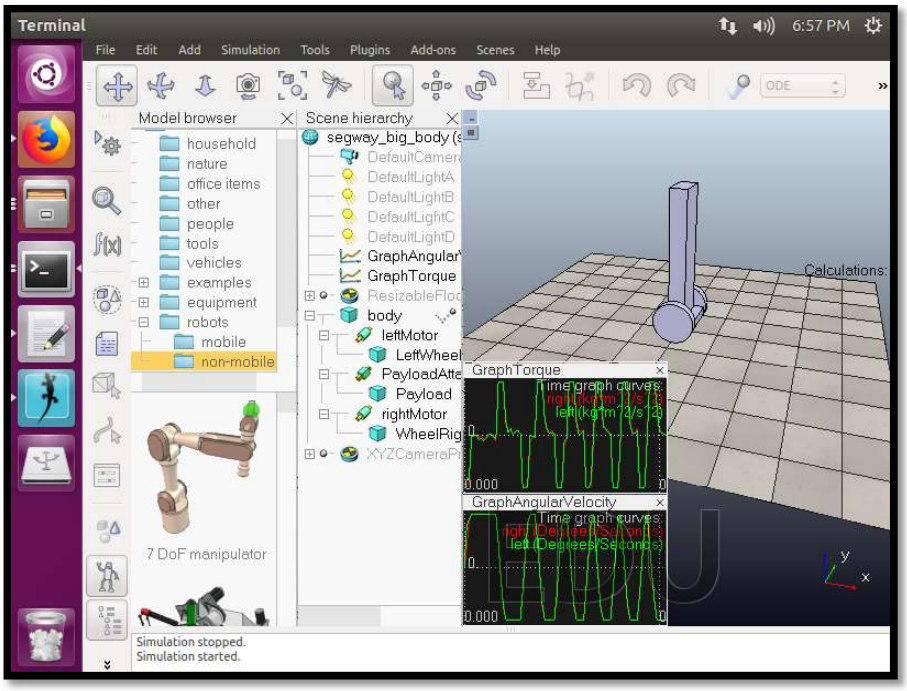

Figura 27. Escena del péndulo con el control.
(Fuente de la imagen: Francisco Álvarez 2018, Fuente de la escena https://github.com/famalgosner/roboticsControl)

3. ANÁLISIS DE RESULTADOS

3.1. Pruebas de funcionamiento del sistema operativo Ubuntu MATE en la Raspberry Pi 3 Model B

3.1.1. Sistemas operativos Ubuntu probados previamente

Las pruebas fueron realizadas para tener un sistema operativo Ubuntu instalado correctamente en la Raspberry Pi 3 Model B, en el cual se pueda ejecutar el Deployer. Los sistemas operativos probados fueron: Ubuntu Core 16, Ubuntu Classic Server 16.04, Lubuntu 16.04.2, Xubuntu 16.04.2 y Ubuntu Core 18. Para cada uno de los sistemas operativos se grabó la imagen de disco en la MicroSD y luego se la insertó en la Raspberry Pi 3 Model B para comenzar su configuración.

3.1.2. Ubuntu Core 18 y Ubuntu Core 16

En los dos sistemas operativos ocurrió un error al inicio de la configuración de los sistemas operativos, ya que pedía una dirección de correo electrónico con cuenta en https://login.ubuntu.com/, a pesar de crear una cuenta y conectarse a internet, apareció el mensaje que aparece en la Figura 28, lo cual no permitió siguir con la configuración en ninguno de los dos sistemas operativos.

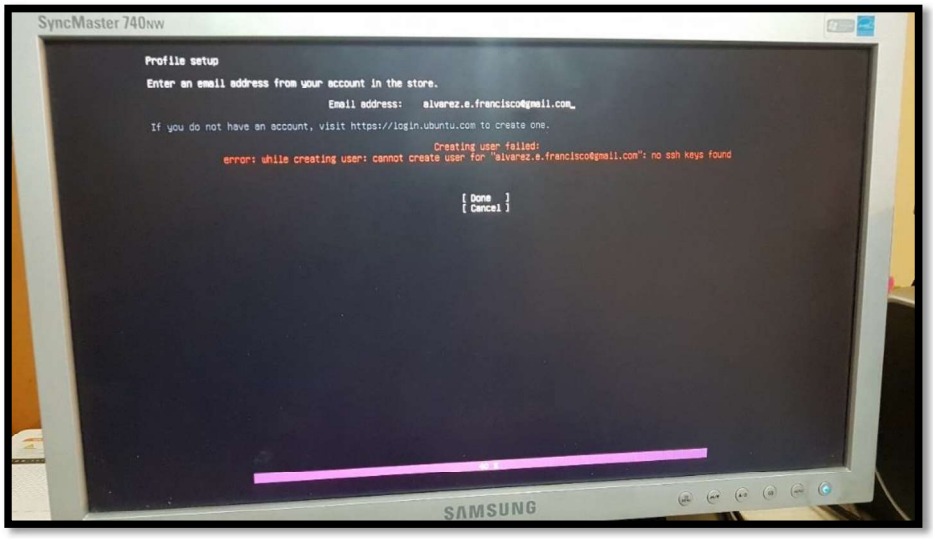

Figura 28. Error en la configuración de Ubuntu Core 18, Ubuntu Classic Server 16.04 y Ubuntu Core 16.
(Fuente: Francisco Álvarez 2018)

3.1.3. Ubuntu Classic Server 16.04

En Ubuntu Classic Server 16.04 no se logró realizar ningún tipo de configuración ya que al ejecutarse el sistema operativo, apareció el mensaje que se muestra en la Figura 29, sin pasar a un menú de configuración.

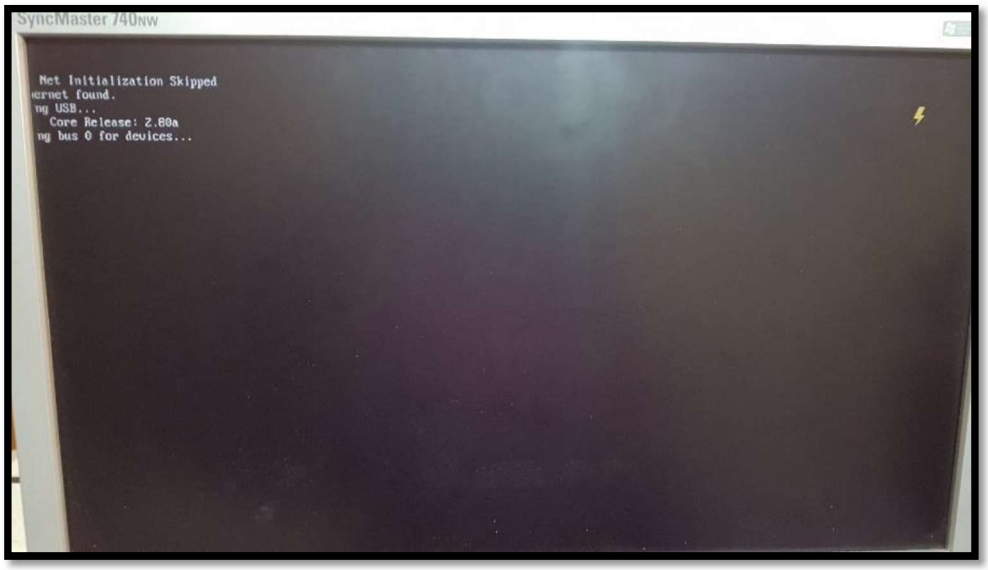

Figura 29. Error en la ejecución de Ubuntu Classic Server 16.04.
(Fuente: Francisco Álvarez 2018)

3.1.4. Xubuntu 16.04.2

En el sistema operativo Xubuntu 16.04.2 se logró completar su configuración, pero el Deployer no se logró ejecutar. Como se muestra en la Figura 30, se presentaron muchos errores que no permitieron la ejecución del archivo .ops peor aún comenzar a ejecutar OROCOS.

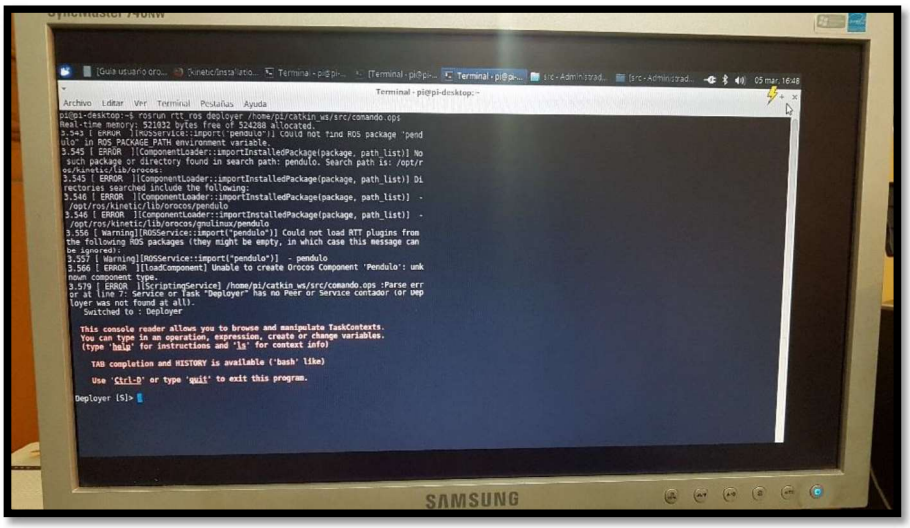

Figura 30. Error en la ejecución del Deployer en Xubuntu 16.04.2.
(Fuente: Francisco Álvarez 2018)

3.1.5. Lubuntu 16.04.2

En el sistema operativo Lubuntu 16.04.2 se logró completar su configuración, pero el Deployer no se logró ejecutar. Como se muestra en la Figura 31, se presentaron muchos errores que no permitieron la ejecución del archivo .ops peor aún comenzar a ejecutar OROCOS.

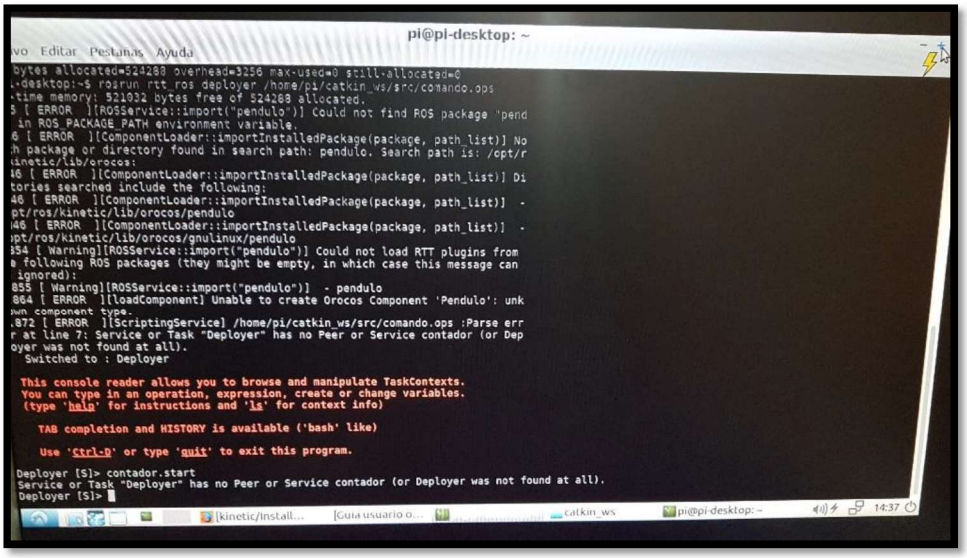

Figura 31. Error en la ejecución del Deployer en Lubuntu 16.04.2.
(Fuente: Francisco Álvarez 2018)

3.1.6. Sistema operativo compatible con ROS y OROCOS (Ubuntu MATE 16.04.2)

La grabación del sistema operativo en la Micro SD fue completado con éxito como se indica en la Figura 32.

Figura 32. Grabación completa de Ubuntu MATE en una Micro SD.
(Fuente: Francisco Álvarez 2018)

Al terminar las configuraciones correspondientes el escritorio de Ubuntu MATE se inicializó correctamente como se indica en la Figura 33, lo cual ya permitió trabajar en la Raspberry Pi 3 Model B.

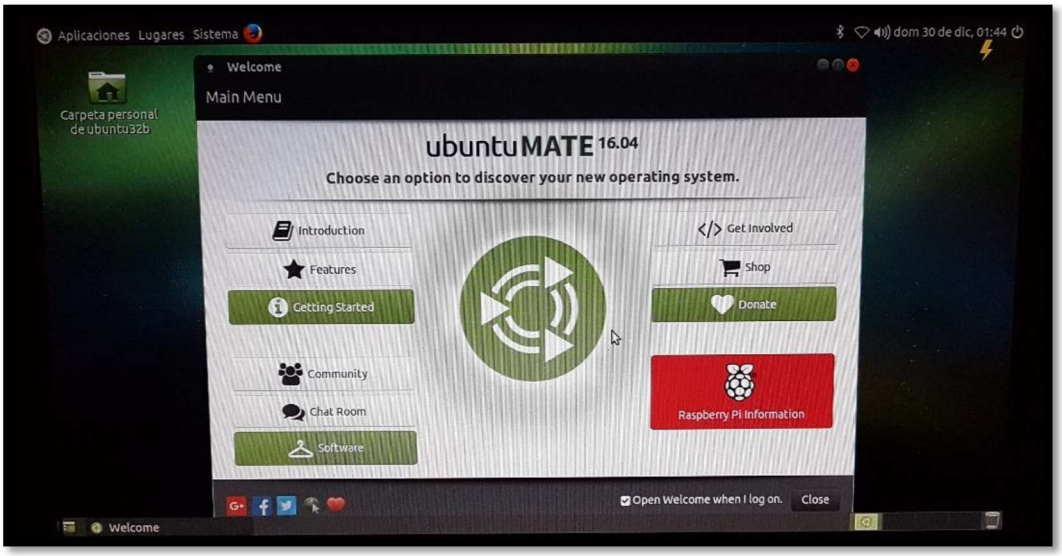

Figura 33. Escritorio de Ubuntu MATE en una Raspberry Pi 3 Model B.
(Fuente: Francisco Álvarez 2018)

3.1.7. Prueba adicional del sistema operativo Ubuntu MATE en la Raspberry Pi 3 Model B+

Se realizó una prueba adicional con la Raspberry Pi 3 Model B+, que se muestra en la Figura 34, ya que es el último modelo en el mercado. Al parecer no hay compatibilidad con Ubuntu MATE, ya que al insertar la Micro SD con el sistema operativo Ubuntu Mate en la pantalla no se ejecuta y aparece la pantalla de la Figura 35.

Figura 34. Raspberry Pi 3 Model B+.
(Fuente: Francisco Álvarez 2018)

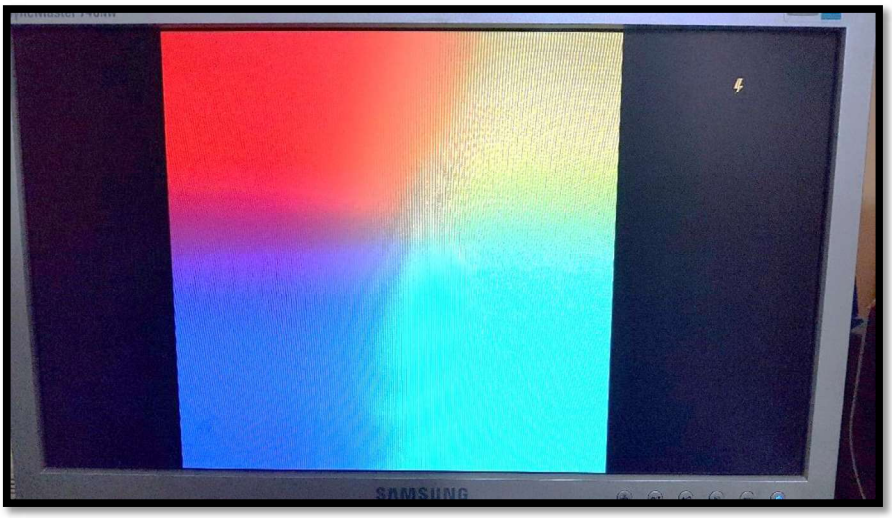

Figura 35. Error al ejecutar Ubuntu MATE en la Raspberry Pi 3 Model B+.
(Fuente: Francisco Álvarez 2018)

Con esta prueba se logró instalar Ubuntu MATE 16.04.2 el cual fue el sistema operativo adecuado para ejecutar el Deployer. El mencionado sistema operativo fue instalado en la Raspberry Pi 3 Model B, que es el penúltimo modelo, ya que Ubuntu MATE 16.04.2 tiene problemas al ejecutarse en la última versión que es la Raspberry Pi 3 Model B+.

3.2. Pruebas de funcionamiento del middleware OROCOS y el componente

Esta prueba fue realizada para comprobar que ROS, OROCOS y el componente estuvieran correctamente ejecutados en la Raspberry Pi 3 Model B. Lo cual permitió que la aplicación del presente proyecto sea realizada con éxito.
La instalación de ROS comienza con la instrucción de la Figura 36 y se construyó las dependencias con la instrucción de la Figura 37.

Figura 36. Instrucción para la instalación completa de ROS en la Raspberry Pi 3 Model B.
(Fuente: Francisco Álvarez 2018)

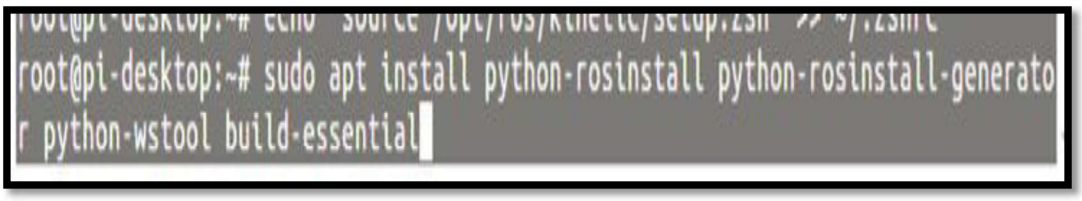

Figura 37. Instrucción para la construcción de dependencias en la Raspberry Pi 3 Model B.
(Fuente: Francisco Álvarez 2018)

Se creó el componente *pendulo* con la instrucción de la Figura 38, lo que permitió que se creara la carpeta para el componente en el directorio donde se encontraba el espacio de trabajo de OROCOS como se indica en la Figura 39.

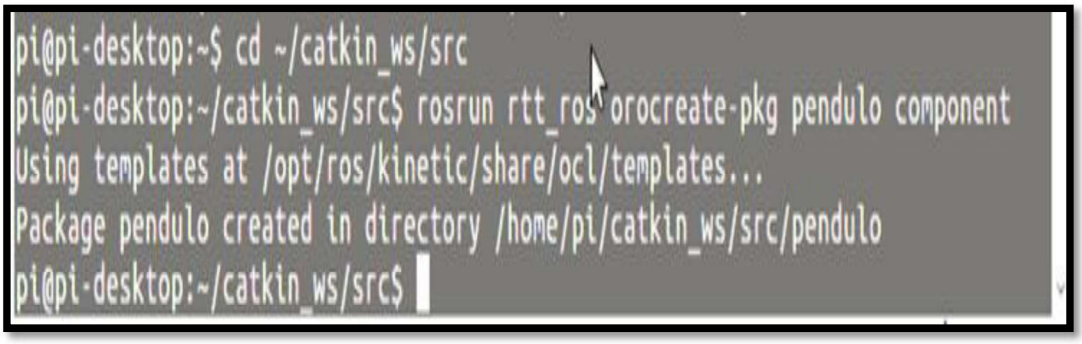

Figura 38. Instrucción para crear el componente péndulo en la Raspberry Pi 3 Model B.
(Fuente: Francisco Álvarez 2018)

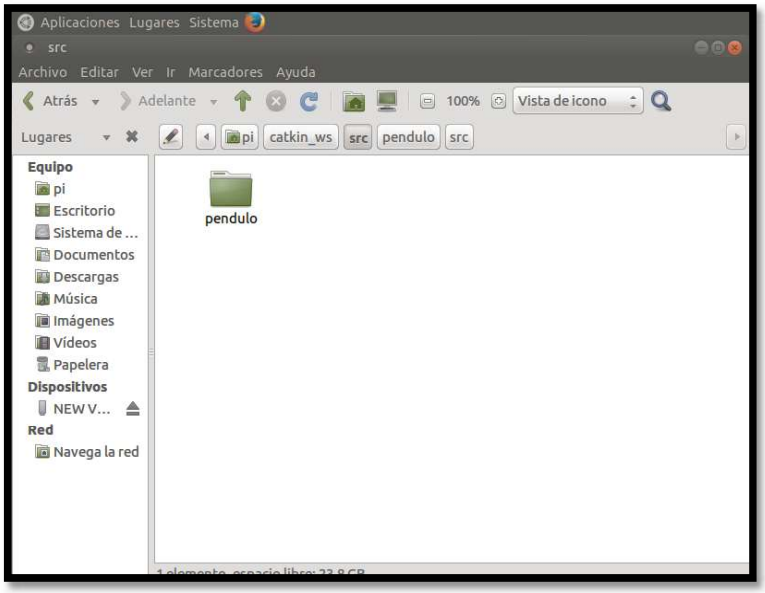

Figura 39. Carpeta pendulo creada en la carpeta catkin_ws en la Raspberry Pi 3 Model B.
(Fuente: Francisco Álvarez 2018)

La creación del componente fue realizada con la instrucción de la Figura 40, una vez creado el componente completamente, se confirmó que su construcción se la haya hecho correctamente, lo cual se presenta en la Figura 41, donde muestra el proceso de compilación al 100%.

Figura 40. Creación del componente pendulo en la Raspberry Pi 3 Model B.
(Fuente: Francisco Álvarez 2018)

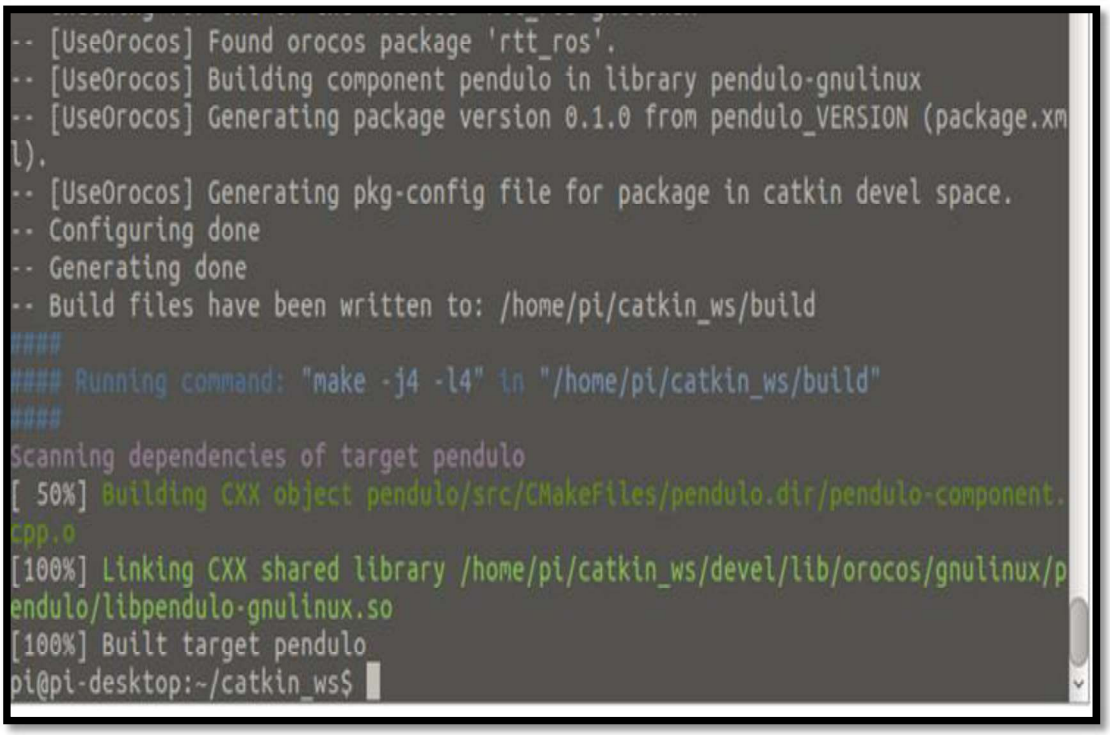

Figura 41. Compilación correcta del Componente pendulo en la Raspberry Pi 3 Model B.
(Fuente: Francisco Álvarez 2018)

Para que se confirme que OROCOS está funcionando en la Raspberry Pi 3 Model B, el deployer debe ejecutar el componente como se indica en la Figura 42.

La prueba garantizó el correcto funcionamiento de ROS y OROCOS, al ejecutar con éxito el Deployer y construir completamente al componente.

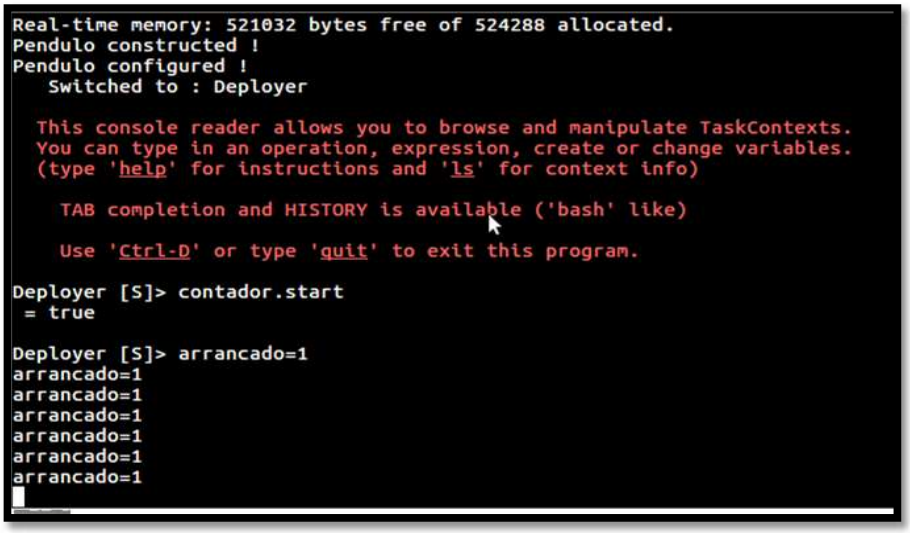

Figura 42. Deployer ejecutado correctamente.
(Fuente: Francisco Álvarez 2018)

3.3. Pruebas del controlador usando el simulador V-REP

Con el propósito de verificar el controlador calculado con el segundo método de la regla de sintonía de Ziegler-Nichols, se puso a prueba el controlador con las constantes $K_p = 588$, $T_i = 44500$ y $T_d = 11125$ logrando establecer un tiempo de equilibrio del péndulo de 267 segundos como se muestra en la Figura 43.

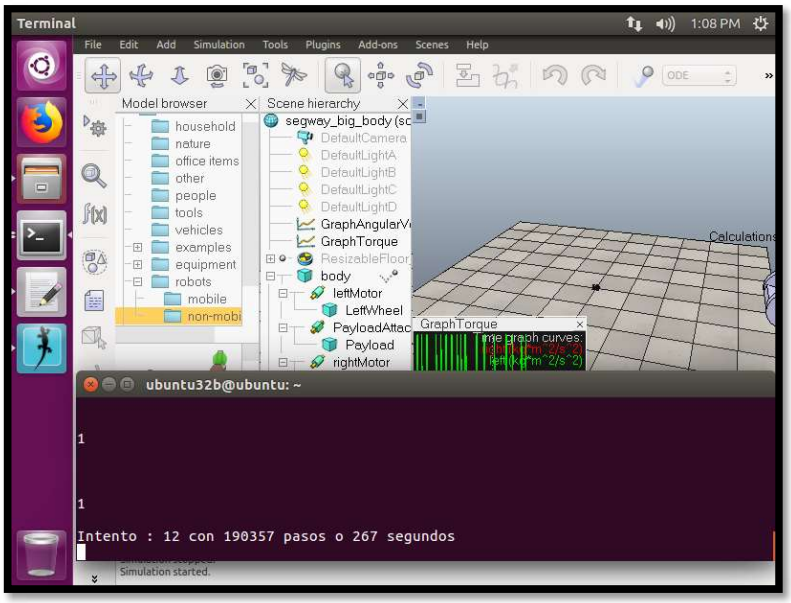

Figura 43. Control implementado en V-REP.
(Fuente de la imagen: Francisco Álvarez 2018, Fuente de la escena https://github.com/famalgosner/roboticsControl)

La prueba mostró la eficiencia del controlador ya que el péndulo permaneció en equilibrio por 267 segundos como un valor máximo de tiempo de estabilidad.

4. Conclusiones y recomendaciones

4.1. Conclusiones

- Se diseñó y se ejecutó la simulación de un controlador para un péndulo invertido virtual ejecutando el middleware OROCOS e implementarlo en una Raspberry Pi con comunicación al simulador con resultados satisfactorios.
- Se creó un componente con el 100% de compilación y se ejecutó el Deployer en una Raspberry Pi 3 Model B, en el sistema operativo Ubuntu MATE.
- Se implementó la escena del péndulo invertido virtual y se ejecutó su controlador escrito en Python por medio de la API que enlaza OROCOS TOOLCHAIN y Python y la API que enlaza V-REP y Python
- Se estableció una conexión, por medio de una interfaz de adquisición de datos, entre la Raspberry Pi y la escena del péndulo virtual en V-REP para su respectivo control.
- El péndulo se estabilizó durante 267 segundos como tiempo máximo, con las constantes $K_p = 588$, $T_i = 44500$ y $T_d = 11125$ implementado en un controlador PID.
- Se disponen de varios simuladores en software libre como son V-REP, Gazebo o ARGoS, pero V-REP se vuelve el más adecuado para el proyecto realizado y para diversas aplicaciones con robots, ya que tiene varias opciones para la funcionalidad de programación como la de programas separados que se conectan a V-REP a través de RemoteAPI, dicha funcionalidad permitió que V-REP se conecte con Python creando así una conexión con OROCOS.
- En [20] una de sus conclusiones fue la siguiente “Raspberry ha destacado por su bajo precio y alta potencia, aunque no ha sido posible integrar Orocos+Ros en ella, por lo que no es posible aprovechar las facilidades y utilidades de Ros en proyectos realizados con Orocos”. En el presente trabajo de titulación se logró instalar e integrar OROCOS+ROS lo cual constituye un avance importante en el desarrollo de aplicaciones que utilizan OROCOS.

4.2. Recomendaciones

- En las instalaciones que realizan en el Terminal es necesario seguir todos los pasos que se presentan. Si algún procedimiento pide permiso, se los pueden ejecutar como super usuario con el fin de cumplir todo el procedimiento.

- El programa que contiene el controlador debe tener un proceso de reinicio para que logre estabilizarse el proceso.
- Es importante que en la programación de los controladores para los robots y en la instalación de OROCOS, ROS y V-REP, leer los textos que se presentan en el terminal, ya que pueden aparecer errores que los omitimos ya que en ocasiones se presentan con el mismo color de los textos que indican una adecuada ejecución.
- Se puede continuar realizando proyectos de investigación para la simulación de robots paralelos, una vez simulados se puede llevar la investigación a la prueba de robots reales con la Raspberry Pi 3 Model B y sus respectivos circuitos de potencia.

5. Referencias Bibliográficas

[1] Ceriani, S., & Migliavacca, M. (2012). Middleware in robotics. Internal Report For „Advanced Methods of Information Technology for Authonomous Robotics", Politecnico di Milano.

[2] Página web oficial de Raspberry Pi: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/

[3] Pi, R. (2015). Raspberry Pi Model B.

[4] Página web oficial de Ubuntu Mate: https://ubuntu-mate.org/

[5] Página web oficial de V-REP: http://www.coppeliarobotics.com/

[6] Página web oficial de V-REP (Manual de usuario): http://www.coppeliarobotics.com/helpFiles/

[7] Página web de descarga de los archivos para la integración Orocos RTT / ROS: https://github.com/orocos/rtt_ros_integration

[8] M. Mauledoux et al., "Tool to Perform Software-in-the-Loop through Robot Operating System", Applied Mechanics and Materials, Vols. 713-715, pp. 2391-2394, 2015

[9] Arent, K., Domski, W., & Cholewiński, M. (2015, August). Deployment of model based robotic control algorithms, designed using Matlab/Simulink, in the form of OROCOS components operating under Linux Xenomai. In Methods and Models in Automation and Robotics (MMAR), 2015 20th International Conference on (pp. 632-637). IEEE.

[10] Vallés, J. C. (2014). Desarrollo de aplicaciones embebidas de control en robots móviles.

[11] Página web oficial de V-REP (Principales características): http://www.coppeliarobotics.com/helpFiles/

[12] Espinel Rivera, K. V. (2016). Evaluación del rendimiento y prestaciones de un decodificador de contenidos interactivos basado en el Ginga-NLC sobre una Raspberry PI (Bachelor's thesis, Universidad de las Fuerzas Armadas ESPE. Carrera de Ingeniería en Electrónica y Telecomunicaciones.).

[13] Página web de descarga de los archivos de la escena y el algoritmo de referencia para el controlador: https://github.com/famalgosner/roboticsControl

[14] Klotzbücher, M., Soetens, P., & Bruyninckx, H. (2010, November). Orocos rtt-lua: an execution environment for building real-time robotic domain specific languages. In International Workshop on Dynamic languages for RObotic and Sensors (Vol. 8).

[15] Upton, E., & Halfacree, G. (2014). Raspberry Pi user guide. John Wiley & Sons.

[16] Dennis, A. K. (2013). Raspberry Pi home automation with Arduino. Packt Publishing Ltd.

[17] Página web oficial de Ubuntu Mate (Objetivos): https://ubuntu-mate.org/about/

[18] Rohmer, E., Singh, S. P., & Freese, M. (2013, November). V-REP: A versatile and scalable robot simulation framework. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 1321-1326). IEEE.

[19] Ogata, K., (2010). Ingeniería de Control Moderna, México: PEARSON

[20] CORONADO VALLÉS, J. O. R. G. E. (2014). Desarrollo de aplicaciones embebidas de control en robots móviles.

[21] Pitonakova, L., Giuliani, M., Pipe, A., & Winfield, A. (2018, July). Feature and performance comparison of the V-REP, Gazebo and ARGoS robot simulators. In Annual Conference Towards Autonomous Robotic Systems (pp. 357-368). Springer, Cham.

[22] González-Barahona, J. M., Perez, M. O., de las Heras Quirós, P., González, J. C., & Olivera, V. M. (2001). Counting potatoes: the size of Debian 2.2. Upgrade Magazine, 2(6), 60-66.

[23] Beltrán Alonso, J. L. (2011). Simulación de un péndulo invertido(Doctoral dissertation).

[24] The Orocos Project Smarter control in robotics & automation (Official website): http://www.orocos.org/content/history

[25] Página oficial de Python: https://www.python.org/

[26] Lara, R. C. H. (2013). Herramientas de software libre para aplicaciones en ciencias e ingeniería. Revista Politécnica, 32.