SISTEMA AUTÓNOMO DE RECORRIDO DINÁMICO

David Fernando Garzón Torres

Juan Camilo Ríos Ardila

Daniel Ricardo Vergara Fonseca

Proyecto de grado para optar al título de Ingeniero de Sistemas

Director: Jorge Villalobos Alvarado

Escuela Colombiana de Ingeniería Julio Garavito

Ingeniería de Sistemas

Bogotá, 8 de diciembre de 2015

HOJA DE APROBACIÓN

El Proyecto de Grado titulado “Sistema Autónomo de Recorrido Dinámico” presentado por David Fernando Garzón Torres, Juan Camilo Ríos Ardila y Daniel Ricardo Vergara Fonseca, en cumplimiento parcial de los requisitos para optar al Título de Ingeniero de Sistemas, fue aprobado el 1 de diciembre de 2015 por el siguiente jurado:

_____
Oswaldo Castillo Navetty
Decano de la Facultad de Ingeniería de Sistemas

_____
Gerardo Ospina Hernández
Ingeniero de Sistemas

CONTENIDO

GLOSARIO

BROADCOM: uno de los principales fabricantes de circuitos integrados para comunicaciones de banda ancha de los Estados Unidos.

DETERMINISTA: se denomina sistema determinista a aquel en que el azar no está involucrado en el desarrollo de los futuros estados del sistema. Un modelo determinista producirá siempre la misma salida a partir de las mismas condiciones de partida o el estado inicial.

ESTOCÁSTICO: sistema cuyo comportamiento es intrínsecamente no determinista. Un proceso estocástico es aquel cuyo comportamiento es no determinista, en la medida que el subsiguiente estado del sistema está determinado tanto por las acciones predecibles del proceso como por elementos aleatorios.

IDLE: entorno integrado de desarrollo y aprendizaje (Integrated Development and Learning Environment), es un entorno de desarrollo integrado para el Python que fue introducido con la implementación por defecto de dicho lenguaje desde la versión 1.5.2b1. Este paquete es una parte opcional de las librerías de Python con muchas distribuciones de Linux.

LXDE: entorno de escritorio libre para Unix y otras plataformas POSIX, como Linux o BSD. El nombre corresponde a "Lightweight X11 Desktop Environment", que en español significa Entorno de escritorio X11 ligero.

MIDORI: navegador web ligero basado su motor principal en Webkit conocido por su eficacia en el navegador Safari de Mac OS X , Chrome de Google, Opera de Opera Software.

PYTHON: lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

RASPBERRY: computador de placa reducida o (placa única) (SBC) de bajo costo.

REST: Transferencia de Estado Representacional (Representational State Transfer) es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.

SCRATCH: lenguaje de programación que permite aprender a escribir de forma sintácticamente correcta.

SOC: System-on-a-chip describe la tendencia cada vez más frecuente de usar tecnologías de fabricación que integran todos o gran parte de los módulos componentes de un computador o cualquier otro tipo de sistema informático o electrónico en un único circuito.

RESUMEN

Un Raspberry Pi es un ordenador de placa reducida, la cual cuenta con las mismas funcionalidades que un ordenador de sobremesa y soporta varios componentes necesarios en un computador convencional. Fue diseñado por un grupo de estudiantes y profesores de la Universidad de Cambridge para fomentar la enseñanza de las Ciencias de la Computación a los niños. Cuenta con una gran cantidad de aplicaciones, entre las cuales se cuentan la robótica y el desarrollo de diversas aplicaciones.
Este documento describe cómo se diseñó e implementó un dispositivo basado en un Raspberry Pi capaz de realizar un recorrido de forma autónoma, basándose en un algoritmo estocástico.

INTRODUCCIÓN

El avance tecnológico de hoy en día nos ha brindado la capacidad de diseñar dispositivos de un tamaño bastante reducido, que ofrecen una gran funcionalidad a un costo muy bajo. Tenemos, por ejemplo, el Raspberry Pi, el cual es un ordenador de placa reducida que ofrece las mismas funcionalidades de un computador de sobremesa. Este dispositivo tiene el tamaño aproximado de una tarjeta de crédito y cuanta con puertos para conectar pantalla, mouse, teclado, adaptador Wi-Fi, entre otros, lo cual le da la característica de ser manipulado de la misma forma que un PC convencional.

El Raspberry Pi fue desarrollado en el Reino Unido en el año 2006 por la fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. Actualmente este pequeño ordenador ha encontrado una gran cantidad de aplicaciones, entre las que pueden contarse el diseño de robots, supercomputadores, desarrollo de software, videojuegos, entre otros.

El modelo B cuanta con un SoC (System on a Chip) Broadcom BCM2835 que incluye CPU, GPU, DSP, SDRAM y puerto USB. Su CPU es una ARM 1176JZF a 700 MHz. Cuenta con una memoria SDRAM de 512 MB la cual comparte con la GPU. Su unidad de procesamiento gráfico (GPU) es una Broadcom VideoCore IV, OpenGL ES 2.0. Adicionalmente cuenta con Dos puertos USB 2.0 vía hub USB, una salida de video HDMI, una salida de audio de 3.5mm, una ranura MicroSD para el almacenamiento y una conectividad de red 10/100 Ethernet (RJ-45).

En cuanto al software, el Raspberry Pi usa sistemas operativos basados en el núcleo de Linux. En Julio de 2012 se lanzó Raspbian, el cuál es un sistema operativo libre derivado de Debian, que está optimizado para el hardware de Raspberry Pi, por lo cual, es el más usado en estos dispositivos. Este sistema operativo usa el entorno de escritorio LXDE y Midori como navegador web. Adicionalmente, este sistema operativo ofrece herramientas de desarrollo como IDLE para lenguajes de programación como Python o Scratch.

Dadas las características técnicas de la Raspberry Pi, su economía y portabilidad, el equipo que desarrollo el presente proyecto seleccionó este dispositivo como base para la implementación del mismo. SARD (Sistema Autónomo de Recorrido Dinámico) es un proyecto que tiene como objetivo el diseño y la implementación de un Algoritmo Estocástico en un dispositivo autónomo con el fin de realizar un recorrido parcialmente aleatorio de un área previamente determinada.

Un sistema estocástico es aquel cuyo comportamiento es no determinista, en la medida que el subsiguiente estado del sistema está determinado tanto por las acciones predecibles del proceso como por elementos aleatorios. En inteligencia artificial, un algoritmo estocástico opera utilizando métodos probabilísticos para solucionar problemas.
El resultado del desarrollo del proyecto fue un sistema compuesto, esencialmente por 3 partes: Un vehículo pequeño (22cm x 16cm) alimentado por dos baterías, el cual cuenta con dos motores MC-2, que son controlados por el Raspberry mediante un circuito electrónico. La segunda parte consta de un Servidor REST, el cual permite iniciar la ejecución de operaciones sobre los datos en cualquier formato. El último elemento que compone el sistema es cualquier dispositivo que pueda conectarse a internet (Pc, Tablet, Smartphone), con el cual se determinan los datos de entrada (Área de recorrido) mediante una página Web.

Una vez se han ingresado los parámetros de entrada en la página del servidor web, este le indica al servidor REST que genere el área que recorrerá el dispositivo. Hecho esto, el dispositivo pude iniciarse para indicarle a los motores en qué sentido moverse, dependiendo de la salida del algoritmo estocástico. Durante el recorrido, es el servidor REST quien, por medio de la interfaz lógica, calcula cada uno de los movimientos del dispositivo.

Las aplicaciones de este sistema son muy variadas, entre las que se pueden mencionar la detección de minas antipersonales en el ejército, la vigilancia de hogares y empresas, e incluso usos tan elementales como el aseo del hogar mediante una pequeña aspiradora.

INFORMACIÓN GENERAL DEL PROYECTO

Nombre del Proyecto: Sistema Autónomo de Recorrido Dinámico

Director: Jorge Villalobos Alvarado

Estudiantes: David Fernando Garzón Torres, Juan Camilo Ríos Ardila, Daniel Ricardo Vergara Fonseca

Grupo de Investigación: Ciencia, Tecnología y Gestión – CTG-Informática

Línea de Investigación: Informática, Educación y Conocimiento

Duración: 2 Semestres

JUSTIFICACIÓN

Con el auge de diferentes tecnologías autónomas, vemos la posibilidad de la crear una aplicación capaz de recorrer un área acordada para luego ser implementada en diferentes campos de la sociedad, por ejemplo vigilancia en el hogar, en la fuerzas armadas, empresarial, entre otras instituciones públicas o privadas.

OBJETIVOS

El objetivó de esta fase del proyecto se enfocó en adaptar el código desarrollado a un dispositivo manipulado por un Raspberry Pi, con el fin de verificar que el software es, efectivamente, adaptable a una gran variedad de dispositivos.

OBJETIVO GENERAL

Poner en marcha un sistema autónomo de barrido de áreas, mediante un algoritmo implementable en diferentes prototipos.

OBJETIVOS ESPECÍFICOS

- Diseñar e implementar el software de barrido dinámico y autónomo.
- Implementar el software básico de movimiento en un prototipo basado en Raspberry Pi.
- Generar un ambiente de manipulación remota para los dispositivos.

MARCO TEÓRICO Y ESTADO DEL ARTE

Sistema Estocástico
Se denomina estocástico al sistema cuyo comportamiento es intrínsecamente no determinista. Un proceso estocástico es aquel cuyo comportamiento es no determinista, en la medida que el subsiguiente estado del sistema está determinado tanto por las acciones predecibles del proceso como por elementos aleatorios.

Cadena de Markov
En la teoría de la probabilidad, se conoce como cadena de Márkov o modelo de Márkov a un tipo especial de proceso estocástico discreto en el que la probabilidad de que ocurra un evento depende solamente del evento inmediatamente anterior.

Definición
En matemática se define como un proceso estocástico discreto que cumple con la propiedad de Márkov, es decir, si se conoce la historia del sistema hasta su instante actual, su estado presente resume toda la información relevante para describir en probabilidad su estado futuro. Una cadena de Márkov es una secuencia X1, X2, X3,... de variables aleatorias. El dominio de estas variables es llamado espacio estado; el valor de Xn es el

estado del proceso en el tiempo n. Si la distribución de probabilidad condicional de Xn+1 en estados pasados es una función de Xn por sí sola, entonces:

$$P(X_{(n+1)}=x_{(n+1)} \mid X_n=x_n, X_{(n-1)}=x_{(n-1)}, \ldots, X_1=x_1) = P(X_{(n+1)}=x_{(n+1)} \mid X_n=x_n)$$

Donde xi es el estado del proceso en el instante i. La identidad mostrada es la propiedad de Márkov. El uso de esta secuencia en nuestro algoritmo permite al robot tomar diferentes caminos.

Patrones Usados
Para este proyecto se usaron varios patrones de arquitectura de software, para que la construcción de las diferentes aplicaciones interactuarán entre sí, y se adaptaran a los diferentes dispositivos.

Los patrones usados fueron:

- Patrón adaptador
- Patrón fábrica abstracta
- Controlador

El uso de estos patrones aprendidos durante nuestra carrera, nos permite una gran adaptabilidad a diferentes lenguajes y diferentes dispositivos, permitiendo que algoritmo funcione en cualquier medio instalado al que se le pueda aplicar movimiento.

Lenguaje Python
Básicamente es un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Aunque el lenguaje Python es fácil de usar, brinda un entorno robusto, ofreciendo una mayor estructura y soporte para programas de complejidad considerable en comparación con lo que pueden ofrecer los scripts de Unix o archivos por lotes. Por otro lado, Python ofrece mucho más chequeo de error que C, y siendo un lenguaje de muy alto nivel, tiene tipos de datos incorporados como arreglos de tamaño flexible y diccionarios. Debido a sus tipos de datos más generales, Python puede aplicarse a un dominio de problemas mayor que Awk o incluso Perl, y aun así muchas cosas siguen siendo más fáciles en Python que en esos lenguajes.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Dentro de los elementos necesarios para el desarrollo del proyecto con respecto al lenguaje Python tenemos los siguientes:

- Manejo de errores
- Programas ejecutables
- Codificación
- Listas
- Cadenas
- Diferentes tipos de sentencias
- Funciones y procedimientos
- Estructuras de datos
- Diccionarios
- Herencia

CRITERIOS DE EVALUACIÓN

De acuerdo a los aspectos fundamentales del proyecto y nuestro enfoque orientado al logro, y considerando que el proyecto es de carácter investigativo y práctico decidimos que el principal criterio de evaluación para el proyecto, deberá ser relativo a la consecución de tareas y actividades referentes al avance del proyecto.

El desarrollo del proyecto será evaluado de acuerdo a las actividades a realizar, que se plantearán en un cronograma de acuerdo común con el director de proyecto.

Como último criterio de evaluación postulamos el cumplimiento de las actividades y trabajos propuestos en el aula virtual para el desarrollo del proyecto.

DESARROLLO DE LA PROPUESTA

DIAGRAMA DE CLASES

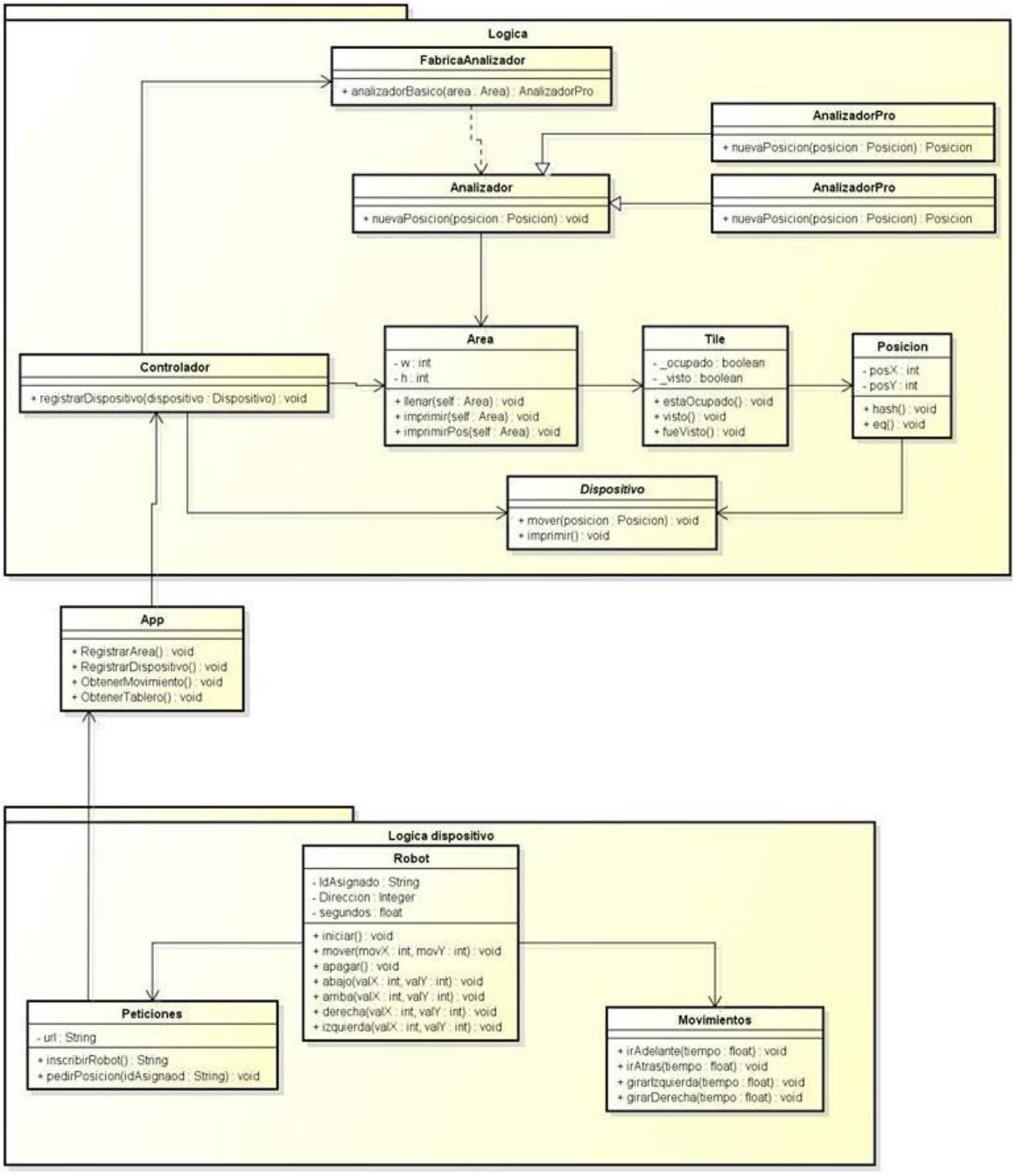
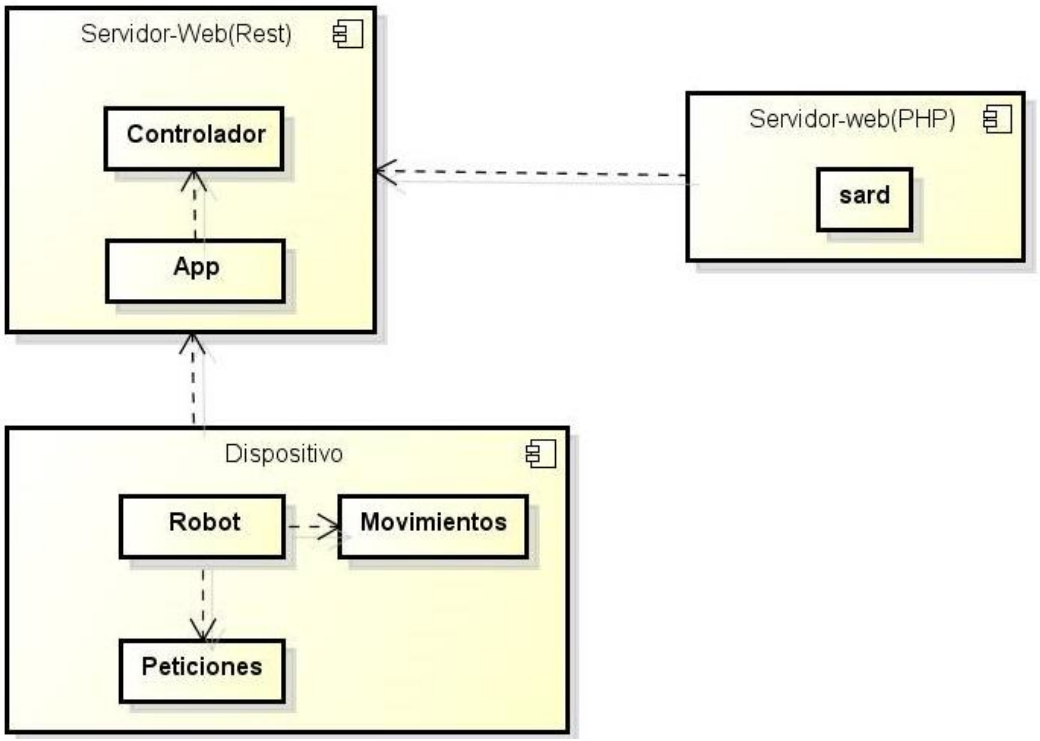


DIAGRAMA DE COMPONENTES



DESCRIPCIÓN DEL PROCESO

El desarrollo del presente proyecto se dividió en dos partes. Inicialmente, en Proyecto de Grado I se diseñó el algoritmo estocástico, el cual permite realizar el recorrido de un área previamente determinada de cualquier geometría. Este algoritmo se implementó usando el lenguaje de programación Python, usando patrones controlador, fábrica abstracta y adaptador.

El patrón controlador sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separa de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control.

En este caso particular, el Patrón Controlador se utilizó para que los servicios REST pudieran acceder a la lógica interna que calcula los movimientos, por medio del algoritmo estocástico.

El patrón Fábrica Abstracta se utiliza para crear conjuntos o familias de objetos, mediante una interfaz, que dependen mutuamente. Este patrón se aplica cuando un sistema debe ser independiente de cómo sean creados los objetos, también cuando un sistema debe ser configurado con una cierta familia de productos o cuando resulta necesario forzar la noción de dependencia mutua entre objetos.

El patrón Fábrica Abstracta se utilizó con el objetivo de implementar clases que permitieran incluir diferentes tipos de algoritmos para calcular las posiciones. Así, es posible configurar el nivel de aleatoriedad del dispositivo, según sea requerido.

El Patrón Adaptador se utiliza para trasformar una interfaz en otra, de tal modo que una clase que no pudiera utilizar la primera, haga uso de ella a través de la segunda. Este patrón permite a las clases trabajar juntas, lo que de otra manera no podría hacerse debido a sus interfaces incompatibles.

Constricción del Dispositivo

Para la segunda parte del desarrollo de este proyecto (Proyecto de Grado II), la construcción del dispositivo se dividió en varias etapas, iniciando con la elaboración de un circuito que permitiera dirigir la energía de la Raspberry a los motores del carro. Este circuito es alimentado por una batería independiente y está conectado a los puertos GPIO (General Purpose Input/Output) de la Raspbery Pi, como muestra la *figura 1*.

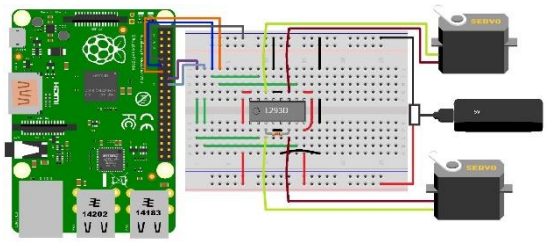


Figura 1: Circuito electrónico

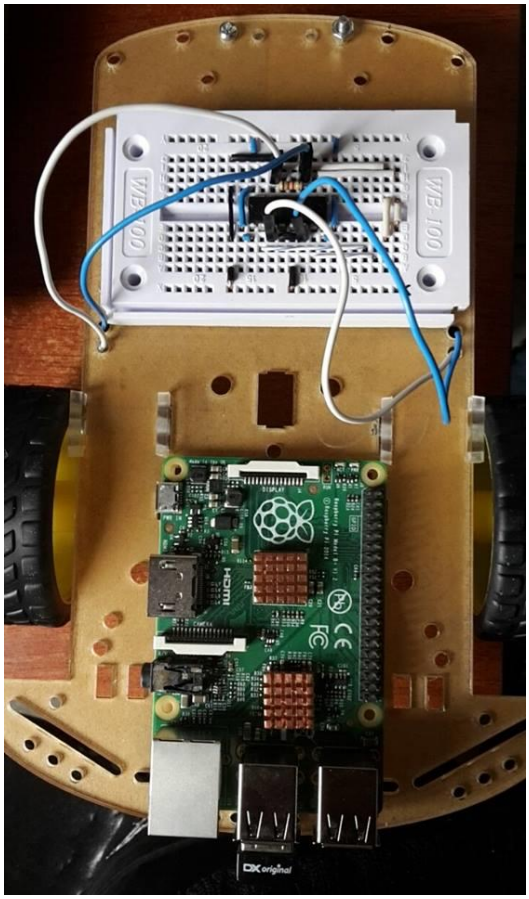


Figura 2: Estructura del carro.

Una vez configurado el circuito, fue necesario crear un script de Python que permitiera manipular los motores del carro. Este script permite manipular 4 pines GPIO de la Raspberry. Estos pines pueden ser configurados para ser entrada o salida y pueden ser activados o desactivados con valores de entrada alto = 1 o bajo = 0. Cada uno de estos pines estaba conectados los polos positivo y negativo de cada uno de los motores, de forma que al activarlos, el carro podrá ir hacia adelante, atrás o girar en cualquier sentido.

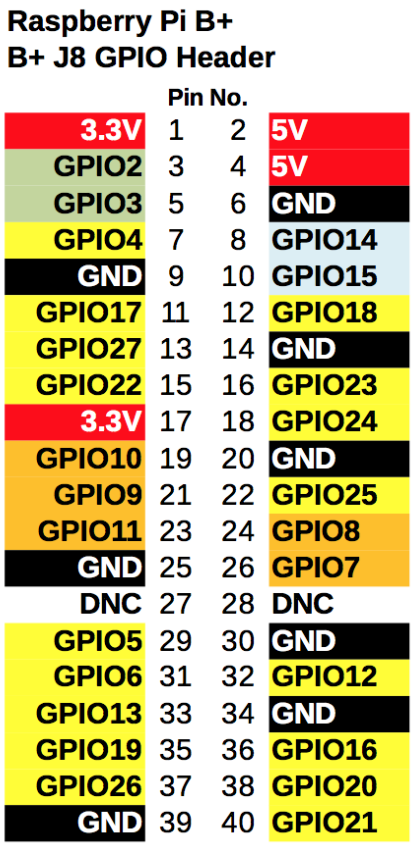


Figura 3: Puertos GPIO de Raspberry Pi B+.

Tal como muestran las figuras 1 y 3, el polo positivo del motor derecho está conectado al pin GPIO 23, mientras que el negativo está conectado al pin GPIO 24. Así mismo, los polos positivo y negativo del motor izquierdo están conectados a los pines GPIO 17 y 27 respectivamente. De esta forma, si se quiere que el carro avance hacia adelante, por ejemplo, es necesario activar los pines GPIO 27 y 23 con valores de entrada alto. De igual forma, si se quiere que el carro rote a la derecha, por ejemplo, habrán de activarse los pines GPIO 17 y 24 con valor de entrada alto.

Una vez implementado el script que controla los motores, fue necesario adaptar el software que ejecuta el algoritmo estocástico, de manera que, una vez éste determine la siguiente posición, los motores realicen los movimientos necesarios para que el carro se ubique en ella. Para lograrlo fue necesario identificar el patrón de movimiento del carro, sin tener en cuenta la dirección del mismo, de esta forma, se alcanza la siguiente posición ejecutando un solo conjunto de instrucciones.

0	1	2
3	Actual	4
5	6	7

Figura 4: Patrón de movimiento.

Como se muestra en la figura 4, y asumiendo que el carro está apuntando en la dirección de (4), si se quiere llegar a la posición (0), por ejemplo, se ejecuta un conjunto de instrucciones que hacen que el carro avance una posición hacia adelante, luego gire a la derecha y, finalmente, avance hacia adelante nuevamente. Una vez se ha logrado alcanzar dicha posición, independiente de la dirección del carro, se tomará el mismo patrón para realizar el siguiente movimiento. De esta forma contamos con 8 métodos para realizar todos los movimientos, evitando así replicar código. Cabe resaltar que en el caso de que el carro tenga que moverse a la posición (2), asumiendo las condiciones iniciales descritas anteriormente, el carro se moverá hacia atrás, de forma que ahorrará hacer dos movimientos y esto permite hacer un uso más eficiente de la batería.

Configuración del Servidor

Con el objetivo de manipular el carro de forma remota, fue necesario configurar un servidor REST en lenguaje Python, el cuál obtiene los datos del área a recorrer. De esta forma, es posible, a través de una página Web, la cual puede ser accedida desde cualquier dispositivo (Computador, Tablet, Smartphone, etc.), indicar el área del recorrido que se desea e iniciar el proceso de recorrido (Figura 5).

En cuanto al servidor, este utiliza un protocolo cliente/servidor sin estado, en el cual cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Adicionalmente, el servidor cuenta con un conjunto de operaciones bien definidas, que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE. Un Diseño REST permite una sintaxis universal para identificar los recursos, los cuales son direccionables únicamente a través de su URI. Dado que el proyecto en su totalidad fue desarrollado el lenguaje de programación Python, fue necesario utilizar el Famework Flask, el cuál es un marco de referencia minimalista escrito en dicho lenguaje.

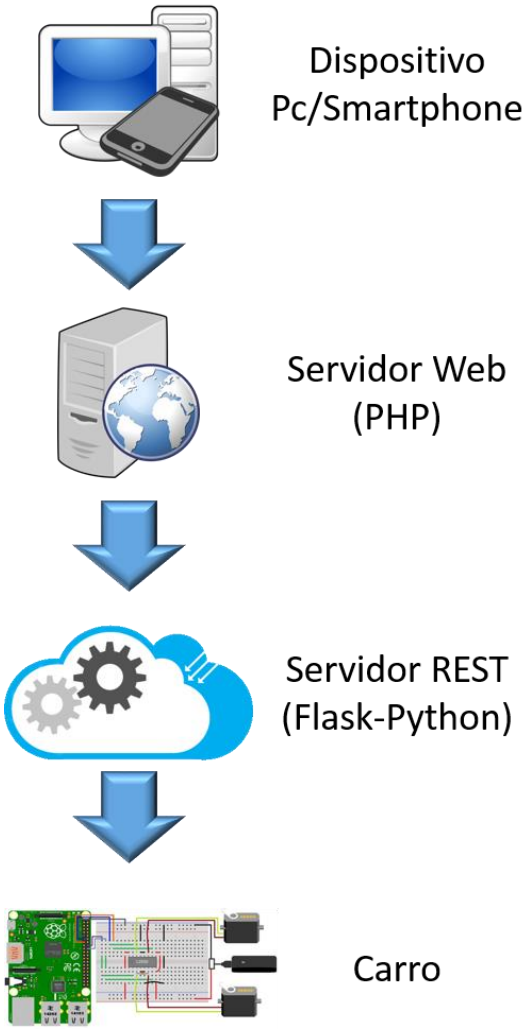


Figura 5: Sistema SARD

Finalmente se obtuvo un sistema que manipula de forma remota un carro que recorre un área rectangular especificada en metros. La geometría especificada es recorrida de forma parcialmente aleatoria, determinada por el algoritmo estocástico. Una vez se especifican las dimensiones del área y se da inicio al proceso a través

de la página web, el software divide dicha área en una cuadrícula, donde cada unidad mide 20cm x 20cm, para este caso específico. Una vez se han recorrido cada uno de los campos de la cuadricula el algoritmo da por terminado el proceso y el carro se detiene.

CONCLUSIONES

Si bien ya se han implementado diferentes algoritmos estocásticos, que permiten realizar recorridos dinámicos, estos están diseñados con el objetivo de optimizar al máximo el uso de la batería. Esto hace que el recorrido sea demasiado predecible, lo cual no cumplía con el objetivo inicial del proyecto: disminuir el factor determinista del recorrido, haciéndolo impredecible, de manera que el sistema resulte más seguro en caso de que decida usarse, por ejemplo, en aplicaciones de seguridad o vigilancia.

Otro factor diferenciador de este proyecto es la capacidad del algoritmo para adaptarse o incorporarse en cualquier tipo de dispositivo.

El uso de patrones y estándares es fundamental en el desarrollo de proyectos software de todo tipo. Mantener propiedades de mantenibilidad y escalabilidad en los proyectos garantiza la continuidad y una mejor aplicabilidad de los mismos. En este caso particular, tener la capacidad de configurar qué tan determinista es el algoritmo, nos brinda la posibilidad de ampliar el rango de aplicaciones que puede tener un algoritmo de este tipo.

REFERENCIAS

1. **Kall, Peter y Wallace, Stein W.** Stochastic Programming. *Benjamin M. Statler College of Engineering and Mineral Resources.* [En línea] 2003. [Citado el: 4 de Agosto de 2015.] http://www.csee.wvu.edu/~xinl/library/books/stochastic_programming.pdf.

2. **Shapiro, Alexander y Philpott, Andy.** A Tutorial on Stochastic Programming. *H. Milton Stewart School of Industrial & Systems Engineering.* [En línea] 2007. [Citado el: 4 de Agosto de 2015.] http://www2.isye.gatech.edu/people/faculty/Alex_Shapiro/TutorialSP.pdf.

3. **Alterovitz, Ron, Simeon, Thierry y Goldberg, Ken.** The Stochastic Motion Roadmap: A Sampling Framework for Planning with Markov Motion Uncertainty. *Robotics Proceedings.* [En línea] 2009. [Citado el: 4 de Agosto de 2015.] http://www.roboticsproceedings.org/rss03/p30.pdf.

4. **Bosetti, Gabriela.** Design Patterns with UML. *Blogspot.* [En línea] 2013. [Citado el: 4 de Agosto de 2015.] http://design-patterns-with-uml.blogspot.com.ar/2013/02/adapter-pattern.html.

5. **Línea de Código.** Patrón Abstract Factory. *Línea de Código.* [En línea] 2013. [Citado el: 4 de Agosto de 2015.] http://lineadecodigo.com/patrones/patron-abstract-factory/.

6. **Python Software Foundation.** Applications for Python. *Python Software Foundation.* [En línea] 2015. [Citado el: 4 de Agosto de 2015.] https://www.python.org/about/apps/.

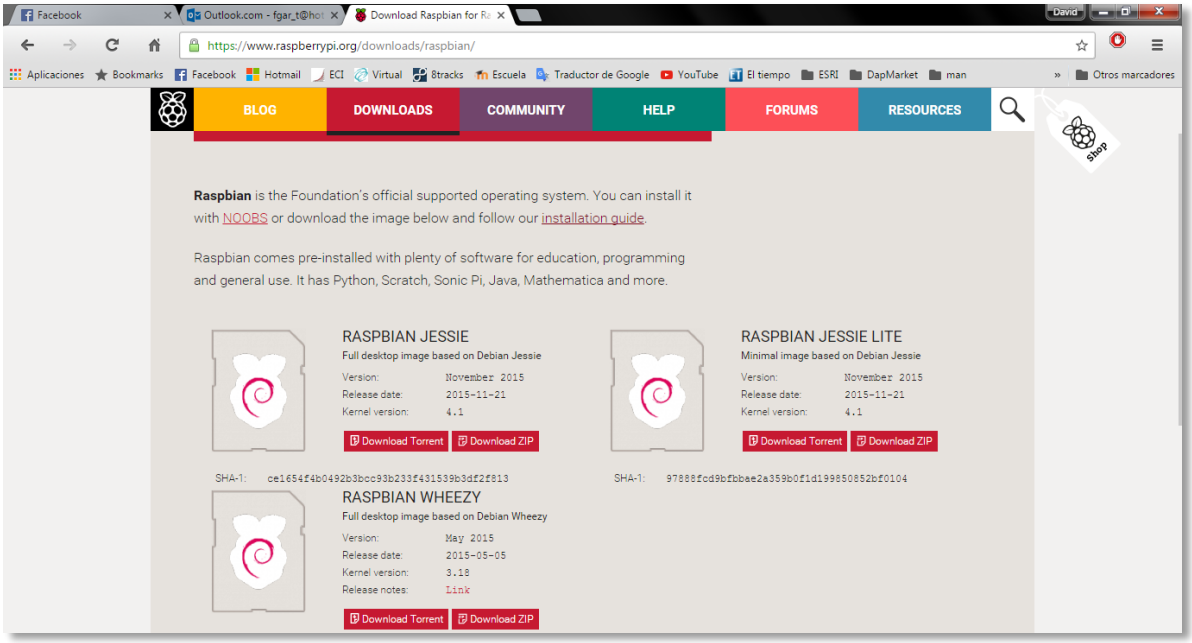
7. **Colaboradores de Wikipedia.** Estocástico. *Wikipedia, La enciclopedia libre.* [En línea] 2015. [Citado el: 4 de Agosto de 2015.] https://es.wikipedia.org/w/index.php?title=Estoc%C3%A1stico&oldid=87200527.

8. —. GRASP. *Wikipedia, La enciclopedia libre.* [En línea] 2015. [Citado el: 4 de Agosto de 2015.] https://es.wikipedia.org/w/index.php?title=GRASP&oldid=83571053.

9. —. Representational State Transfer. *Wikipedia, La enciclopedia libre.* [En línea] 2015. [Citado el: 4 de Agosto de 2015.] https://es.wikipedia.org/w/index.php?title=Representational_State_Transfer&oldid=86118824.

ANEXO 1: Instalación Raspbian para Raspberry Pi B+

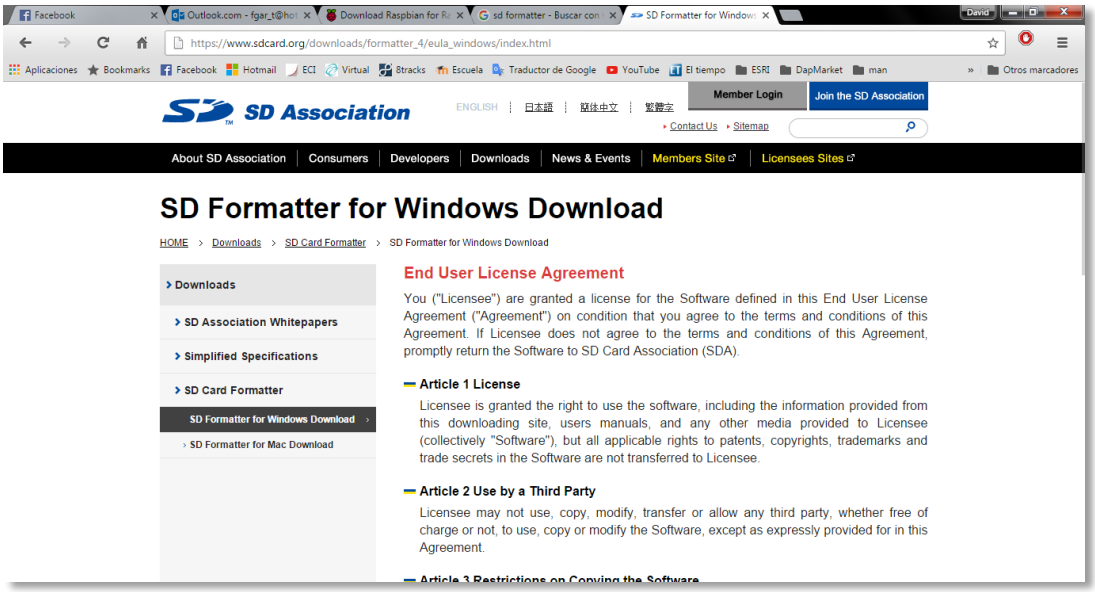
Requisitos: Memoria SD mínimo 4GB



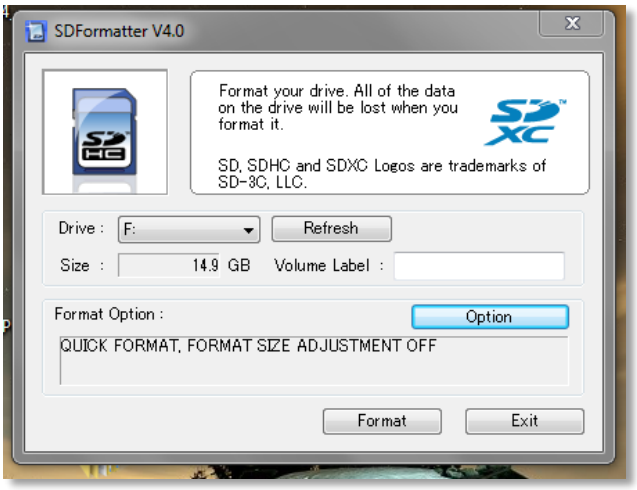
Descargamos la última actualización de la página oficial de Raspberry Pi que mejor se ajuste a nuestra necesidad, y la descomprimimos, esta queda como imagen en nuestro equipo, la cual posteriormente se cargara en la tarjeta SD.

Preparamos la tarjeta SD para la instalación

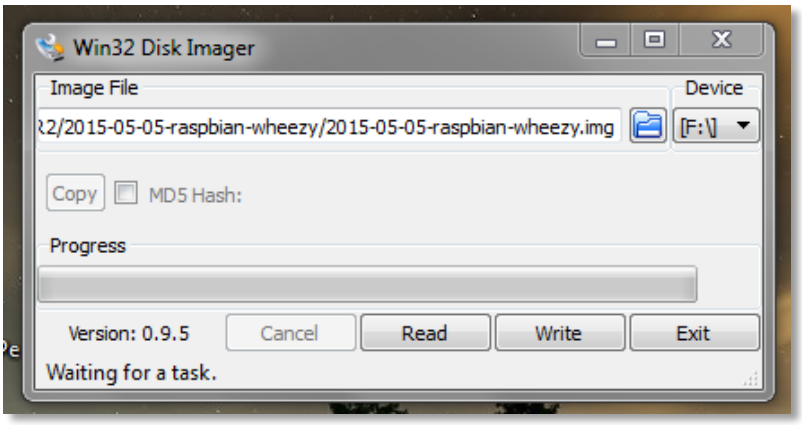
1. Descargamos e instalamos SD FORMATER
https://www.sdcard.org/downloads/formatter_4/index.html



2. Formateamos la memoria con SD FORMATER



3. Descargamos e Instalamos Win32DiskImager
http://sourceforge.net/projects/win32diskimager/

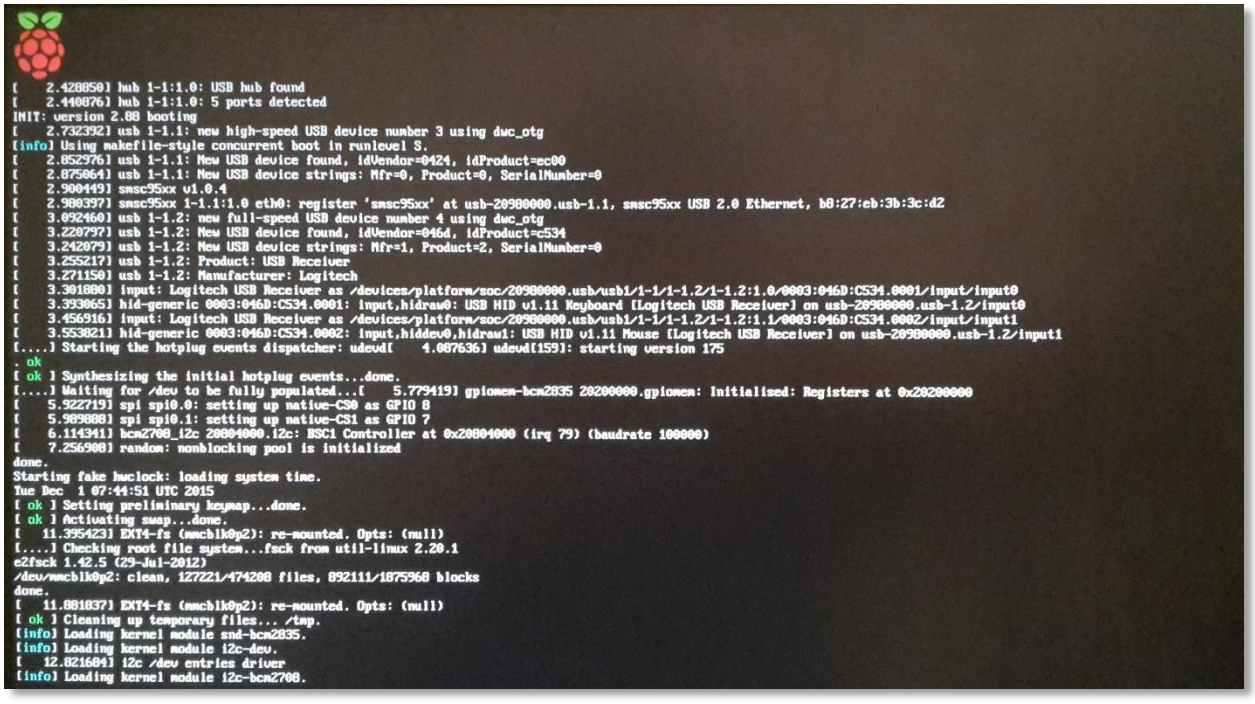


4. Cargamos la imagen donde está Raspbian y le damos en Write
Se demora alrededor de 10 minutos copiando el sistema operativo en la memoria

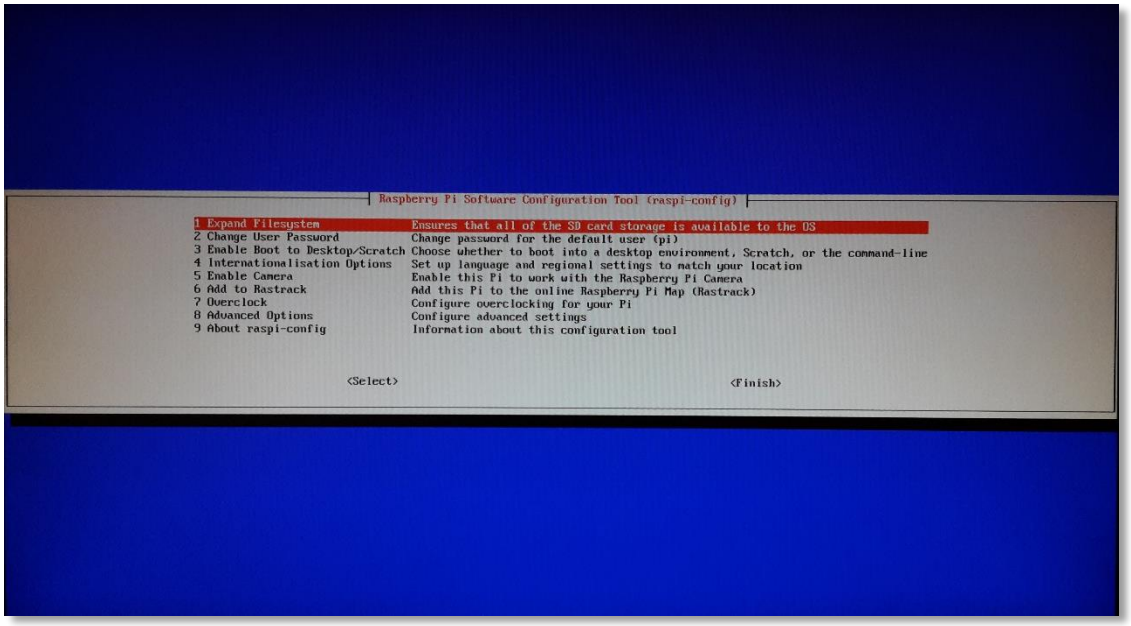
Cuando haya terminado, retiramos la memoria y la introducimos en la tarjeta Raspberry PI

Luego conectamos la Raspberry Pi por HDMI a un TV o monitor, también conectamos un teclado y mouse usb, la fuente de voltaje 5 Voltios - 2 Amperios, y el cable de Ethernet, para poder hacer las actualizaciones.

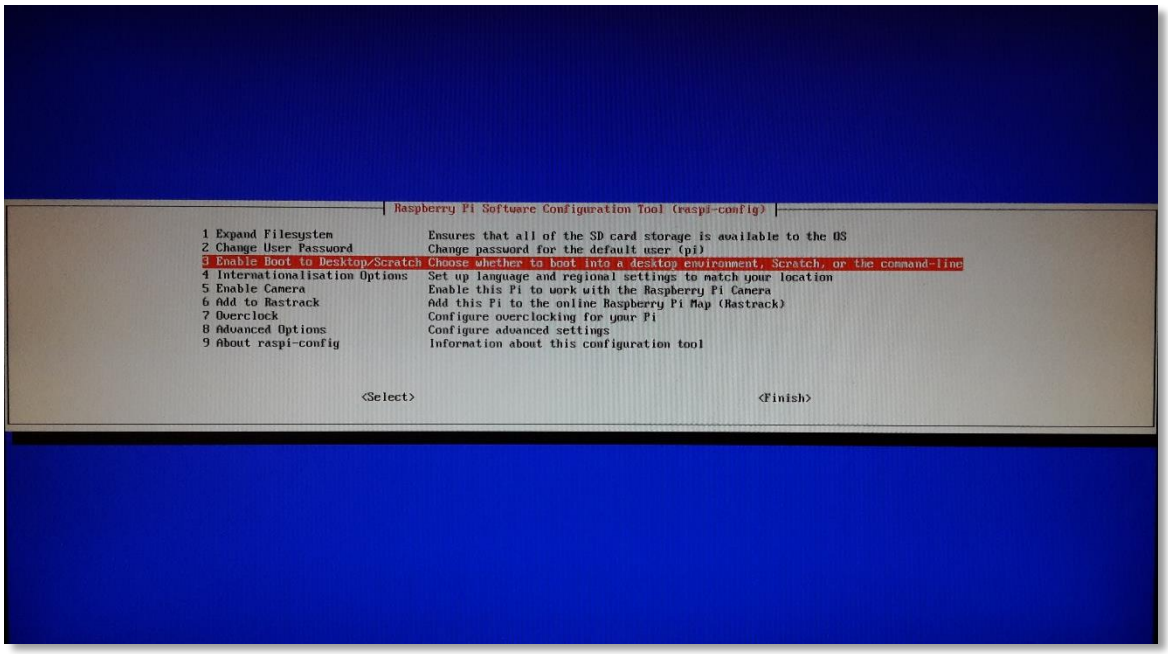
Una vez conectemos la Raspberry Pi se empezaran a cargar controladores y demás archivos del Kernel.



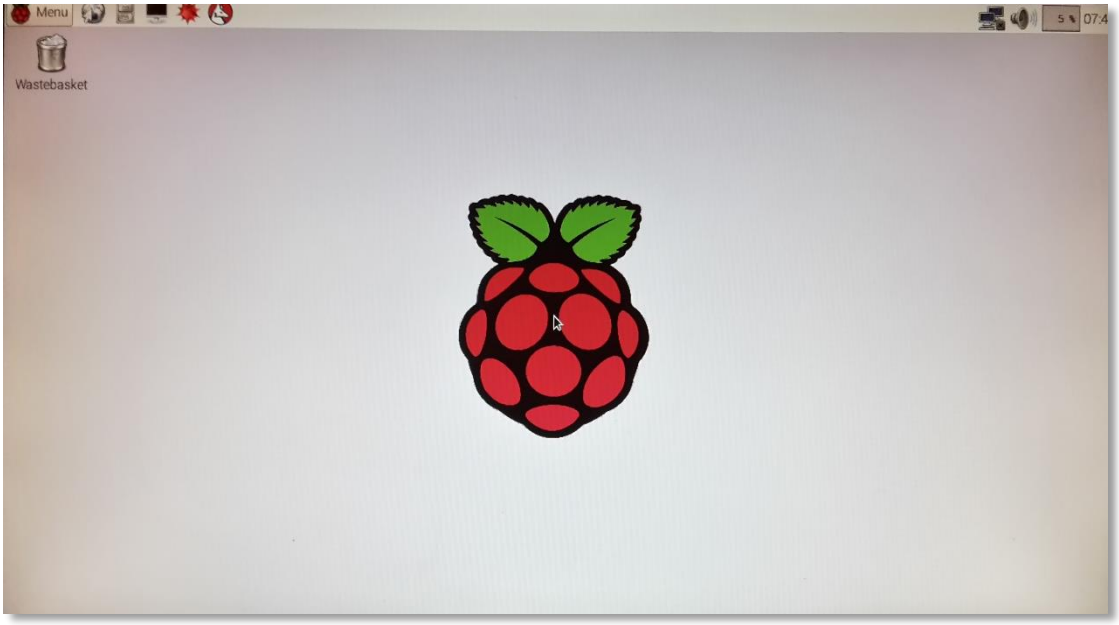
Una vez iniciado por primera vez damos en la primera opción la cual redimensiona la memoria SD, para darnos más espacio, para archivos y aplicaciones.



Ahora vamos a la opción 3, la cual nos habilita el inicio del entorno gráfico, cada vez que conectemos las Raspberry Pi.

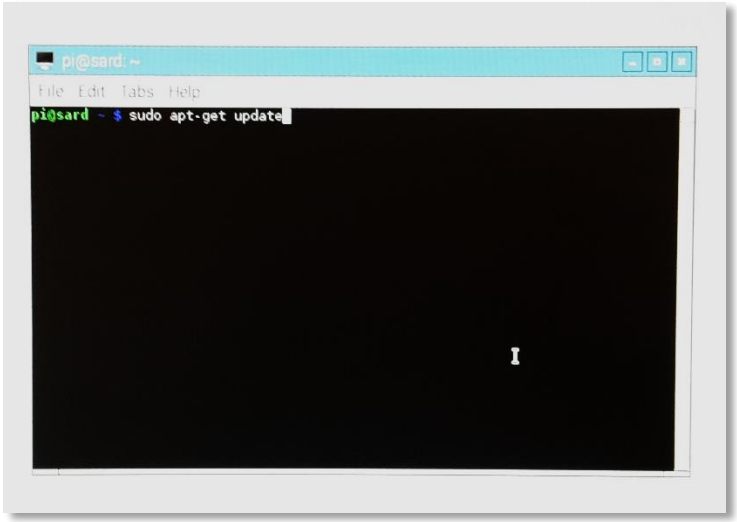


Una vez se reinicie la Raspberry Pi ingresara directamente al entorno gráfico.



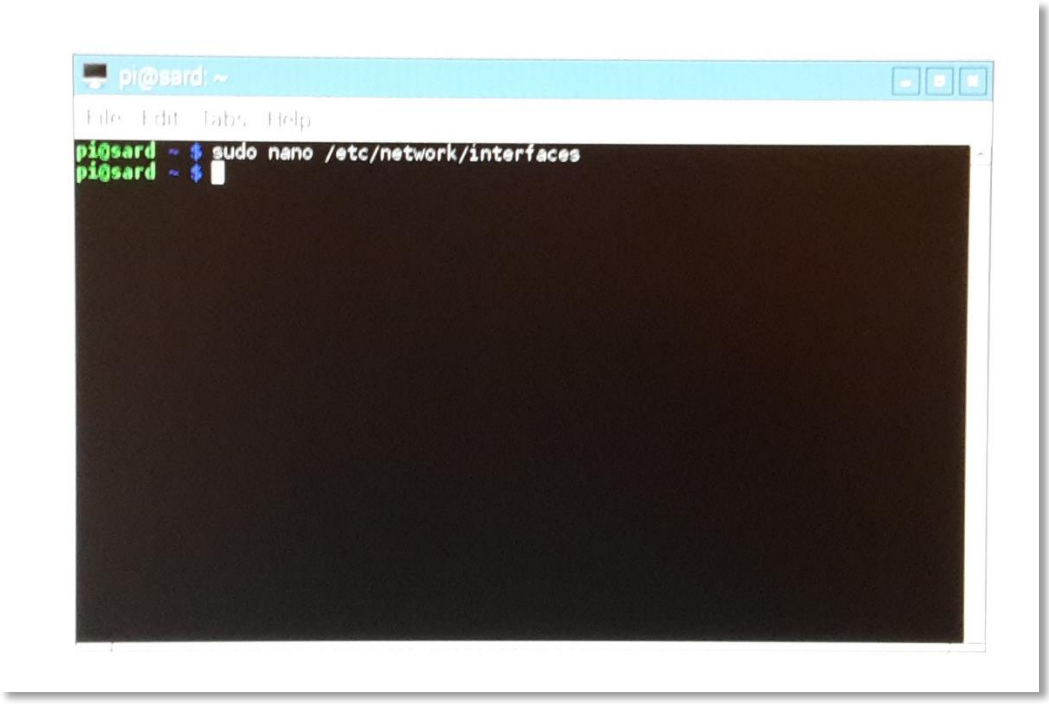
Abrimos la terminal y escribimos el siguiente comando para actualizar los repositorios y el sistema, recuerde tener conectado el cable de internet a la Raspberry Pi

sudo apt-get update

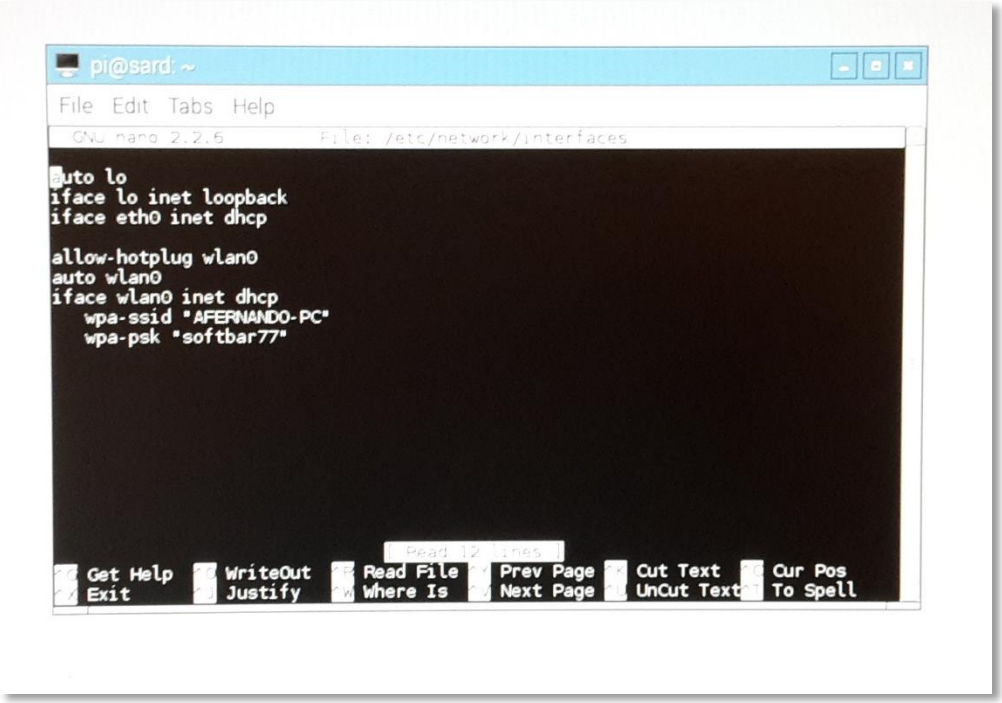


Para conectar el wifi necesitamos a través de un módulo conectado por usb, necesitaremos del comando anterior, una vez actualizado el sistema, escribimos el siguiente comando:

sudo nano /etc/network/interfaces



Una vez demos enter nos saldrá un editor de texto dentro de la terminal, y configuraremos el archivo de la siguiente manera:



El wpa-ssid se debe cambiar por el nombre de la red wifi a la que se deseen conectar, igual con el wpa-psk el cual es la contraseña de la red.

Una vez hecho esto, guardamos con *ctrl+o* y cerraremos el archivo dando *ctrl+x,* y reiniciaremos con el comando:
sudo reboot

Con esto ya tendremos configurada la Raspberry Pi.

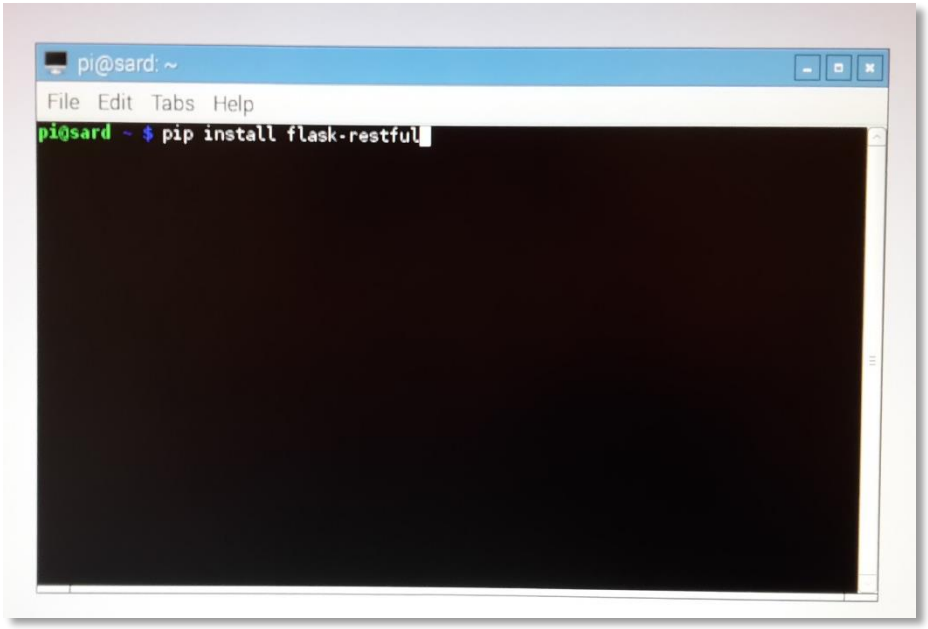
Si hay problemas de instalación se recomienda seguir los pasos que están en el siguiente link:

https://www.raspberrypi.org/help/noobs-setup/

ANEXO 2: Instalación del REST Framework Flask

Para instalar Flask-RESTful utilizaremos el siguiente comando

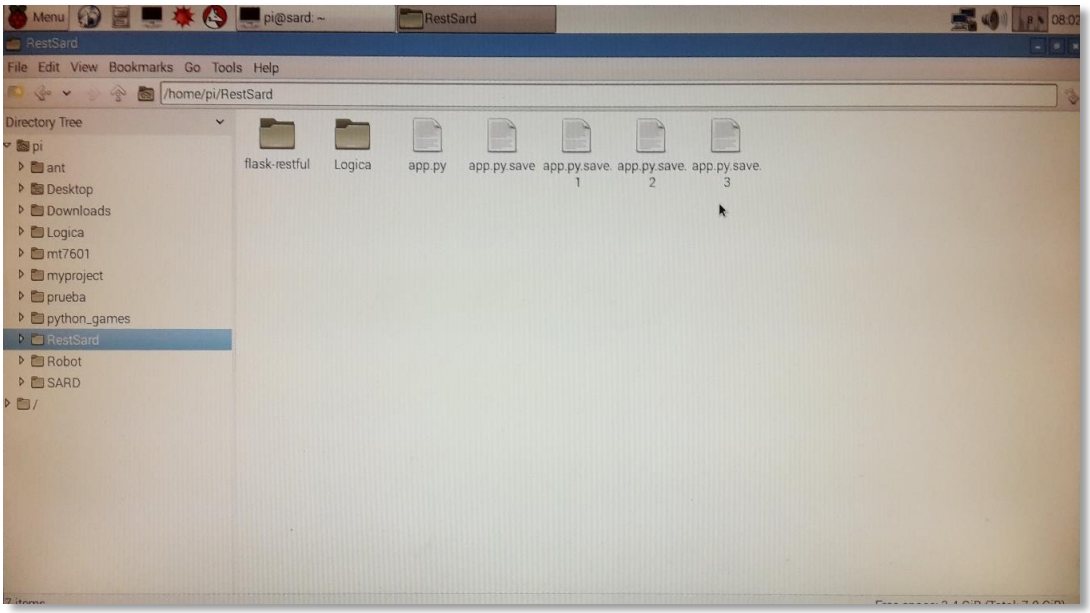
sudo pip install flask-restful



Flask-RESTful requiere alguna de las siguientes versiones de python 2.6, 2.7, 3.3, o 3.4.

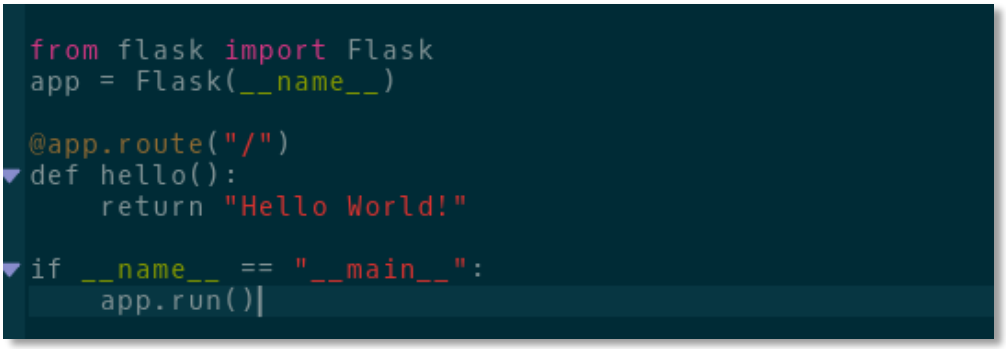
Este comando instalará los módulos necesarios para que nuestra aplicación pueda recibir peticiones ya sea post, get, put, delete, a través de url's web.

Una vez instalado creamos una carpeta las cuál contendrá una archivo Python, y si necesitamos otros módulos más los podremos incluir.



Para nuestro caso hemos creado un archivo app.py, el cuál contendrá las diferentes configuraciones para poder realizar las peticiones que nuestra aplicación requiera.

Este es un ejemplo básico de lo que se puede hacer:



Flask-RESTful ofrece un api sencillo de implementar, con definiciones simples y efectivas, las cuales fueron muy beneficiosas en el desarrollo el proyecto.

ANEXO 3: Instalando y Configurando Servidor Web en Raspberry Pi

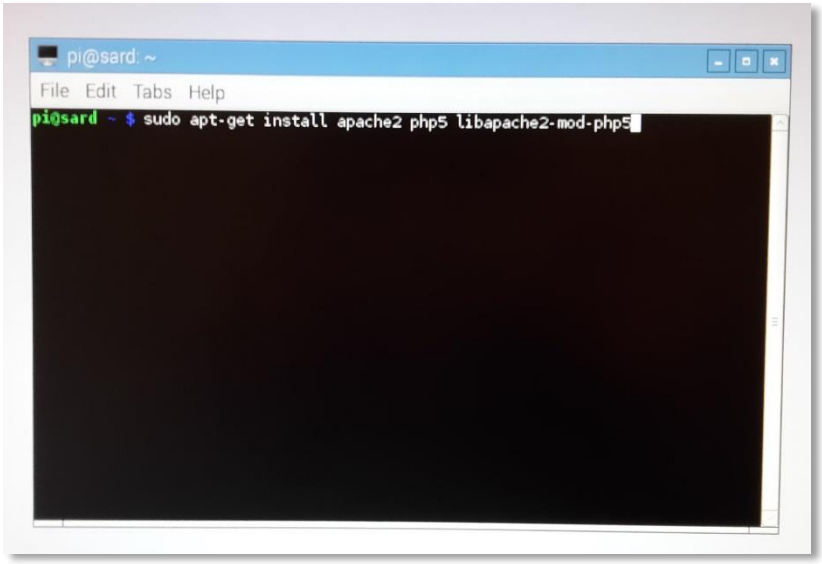
Ahora vamos a instalar y configurar servidor APACHE, con PHP.

Primero que todo ejecutamos los siguientes comandos:

sudo apt-get update
sudo apt-get upgrade

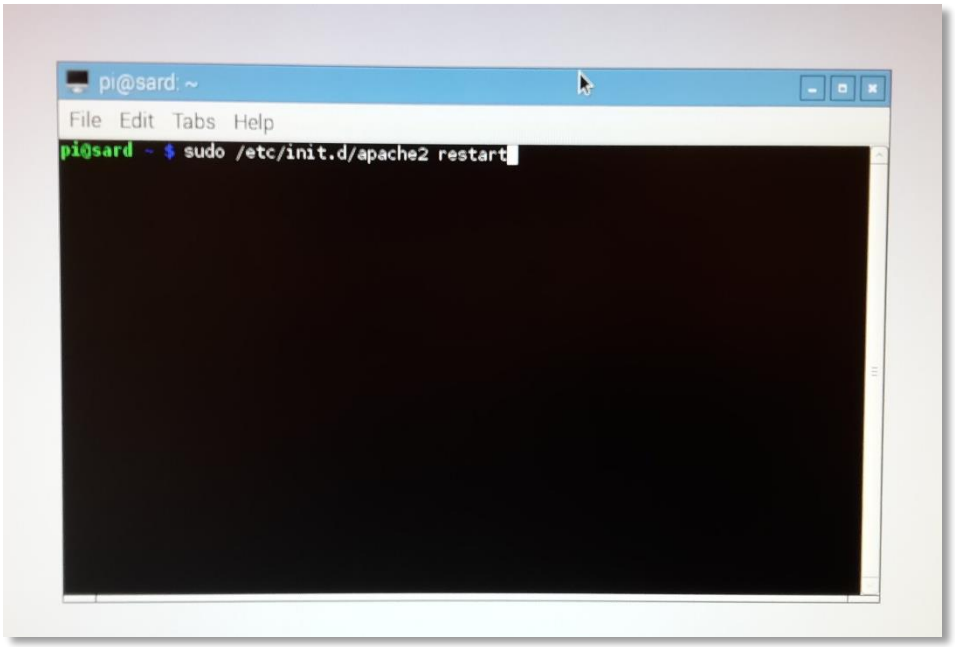
Luego procedemos a instalar apache y php, con el siguiente comando:

sudo apt-get install apache2 php5 libapache2-mod-php5



Una vez instalado, reiniciamos apache:

sudo /etc/init.d/apache2 restart



Otorgamiento de permisos

Los directorios típicamente utilizados por un servidor web en linux se sitúan en ***/var/www***, y el usuario típico para dicho entorno suele ser ***www-data***. Ahora vamos a crear el grupo y usuario estándar para nuestro servidor, a la par que otorgamos los permisos pertinentes y añadimos a nuestro usuario por defecto (pi) al grupo comentado. De esta forma no será preciso que el usuario root (su) sea siempre el que pueda operar en /var/www.

sudo chown www-data:www-data /var/www/

Damos los permisos a la carpeta www/

sudo chmod 775 /var/www/

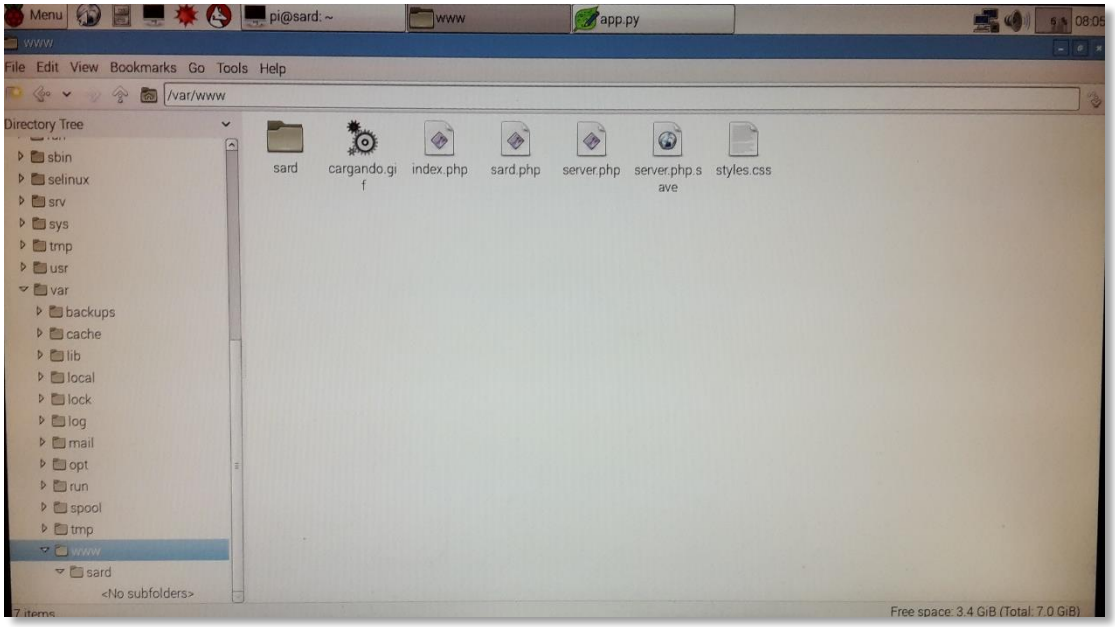
Añadimos el usuario pi al grupo www-data

sudo usermod -a -G www-data pi
sudo visudo

Agregamos el siguiente código al final.

www-data ALL=(root) NOPASSWD:ALL

Una vez hecho esto podremos empezar a crear archivos en la carpeta **/var/www,** además podremos agregar imágenes, archivos de estilos en cascada, archivos de JavaScript, entre otro tipo de archivos que utilizan normalmente una página web completa.



Y probar nuestra página web directamente en la Raspberry Pi colocando la siguiente url:

http://127.0.0.1/index.php

En caso de que nuestro archivo se llame index.php.