DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA ROBÓTICA MÓVIL ORIENTADO A LA VIGILANCIA Y CONTROLADO POR DISPOSITIVOS MÓVILES

DIRECTORA DEL PROYECTO:
MSc. CLAUDIA LEONOR RUEDA GUZMÁN

INVESTIGADORES:
NELSON FERNANDO MONROY RÍOS
GABRIEL SCHNEIDER LAVERDE ARIAS

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA ELECTRÓNICA
FACULTAD DE INGENIERÍAS
BUCARAMANGA
AÑO 2015

DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA ROBÓTICA MÓVIL ORIENTADO A LA VIGILANCIA Y CONTROLADO POR DISPOSITIVOS MÓVILES

NELSON FERNANDO MONROY RÍOS
GABRIEL SCHNEIDER LAVERDE ARIAS

TRABAJO DE GRADO PARA OPTAR A TÍTULO DE INGENIERO ELECTRÓNICO

DIRECTOR:

MSc. CLAUDIA LEONOR RUEDA GUZMAN

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA
INGENIERÍA ELECTRÓNICA
FLORIDABLANCA
2015

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

AGRADECIMIENTOS

A todos los docentes y estudiantes de la facultad de ingeniería electrónica seccional Bucaramanga que apoyaron incondicionalmente el desarrollo de este proyecto de grado.

A nuestras familias por su paciencia, cariño y apoyo durante todo el proceso de pregrado y la realización de este proyecto de grado.

A Dios por darnos la oportunidad de convertirnos ingenieros electrónicos.

RESUMEN GENERAL DE TRABAJO DE GRADO

TITULO: DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA ROBÓTICA MÓVIL ORIENTADO A LA VIGILANCIA Y CONTROLADO POR DISPOSITIVOS MÓVILES

AUTOR: Gabriel Schneider Laverde Arias
Nelson Fernando Monroy Ríos

FACULTAD: Ingeniería electrónica

DIRECTOR: Msc. Claudia Leonor Rueda Guzmán.

RESUMEN:

En este proyecto se desarrolló una plataforma robótica móvil utilizada para la exploración de lugares terrestres orientado a la vigilancia y seguridad de los mismos, Se cuenta con una cámara que permite la visualización de video en línea para recorrer los espacios a vigilar. Este sistema robótico se controla por dispositivos móviles con sistemas operativos iOS y ANDROID utilizando comunicación inalámbrica WI-FI, las aplicaciones se diseñaron en lenguaje Swift y JAVA respetivamente, En la aplicación se ordena la dirección de la translación de la plataforma y a la vez se puede observar la señal de video online. Cuenta con un mapa que permite mostrar la posición del usuario que se encuentra controlando el sistema robótico. El enlace de comunicación lo realiza una tarjeta de computación embebida la cual es configurada para conectarse a una red vía wifi, la conexión puede realizarse punto a punto o puede accederse al móvil desde cualquier dispositivo, con solo colocar la dirección IP, permitiendo que el sistema pueda ser considerado como un dispositivo robótico IOT, también procesa la señal de video procedente de una cámara de video para posteriormente transmitirla al dispositivo móvil. Se cuenta con un microcontrolador encargado de controlar la velocidad de la plataforma gracias a que se ha programado controladores PID para cada motor, por medio de la lectura de los encoder, adicionalmente procesa la información de sensores de ultrasonidos lo cual permite la detección de obstáculos en el camino durante el recorrido del sistema robótico, impidiendo que se genere algún choque, también posee un módulo de posicionamiento global lo que permite ubicar a la plataforma robótica por medio de las coordenadas latitud y longitud que se pueden visualizar en la aplicación móvil.

PALABRAS CLAVES:

Android, Dispositivos móviles, iOS, Plataforma robótica, Raspberry Pi.

V° B° DIRECTOR DE TRABAJO DE GRADO

GENERAL SUMMARY OF WORK OF GRADE

TITLE: DESIGN AND IMPLEMENTATION OF A MOBILE ROBOTIC PLATFORM ORIENTED TO SURVEILLANCE AND CONTROLLED BY MOBILE DEVICES.

AUTHOR(S): Gabriel Schneider Laverde Arias
Nelson Fernando Monroy Ríos

FACULTY: Facultad de Ingeniería Electrónica

DIRECTOR: Msc. Claudia Leonor Rueda Guzmán.

ABSTRACT

This project presents the development of a mobile robotic platform for terrestrial exploration oriented to the surveillance and security of the places. It includes a camera to allow online video visualization in order to cover the spaces to be watched. This robotic system is controlled by mobile devices with iOS and Android operative systems using WiFi wireless communication and the apps were designed in Swift language and Java respectively. In the application is possible to establish the direction for the translation of the platform and watch the online video signal at the same time. It also counts with a map that presents the position of the user who controls the robotic system. The communication link is performed by a embedded computing board configured to be connected to a WiFi network. It is possible to connect peer-to-peer or acceding to the mobile from any device, just using the IP address, allowing the system to be considered as a IoT-aided robotic system capable of processing the video signal from a video camera to transmit it to the mobile device. It includes a micro controller that controls the speed of the platform due to the PID controllers programmed for each motor using the data from the encoders. Besides, it processes the information from the ultrasound sensors allowing the detection of obstacles in the road of the robotic system, preventing any collisions. Finally, it also has a global- positioning module that gives the possibility to locate the robotic platform with the longitude and latitude coordinates visualized in the mobile application.

KEYWORDS:

Android, iOS, Mobile devices, Raspberry Pi, Robotic platform.

V° B° DIRECTOR OF GRADUATE WORK

1 INTRODUCCIÓN

La vigilancia constante de lugares tanto cerrados como abiertos, se ha convertido en una necesidad continua de la sociedad, debido a los altos índices de criminalidad. Esto hace que se haya producido un incremento en los últimos años de la demanda de diferentes métodos para aumentar la seguridad de nuestros hogares y empresas.

Actualmente se utilizan diferentes métodos como la contratación de personal, la adquisición de perros guardianes y la instalación de equipos electrónicos como circuitos cerrados de tv, para supervisar la seguridad de sus propiedades u objetos valiosos, e incluso a sus familiares que no se pueden defender por si solos, como niños y personas de avanzada edad.

Aunque al implementar algunos de estos métodos, aumentan el nivel de seguridad dentro de una área determinada, no son ciento por ciento efectivos, dejando una brecha de seguridad y por lo tanto una oportunidad a los delincuentes para realizar sus acciones.

A raíz de esta problemática, se ha incrementado las investigaciones para aumentar la efectividad de los equipos electrónicos, al mismo tiempo, reducir los posibles errores de factor humano, y en los últimos años se ha incrementado para esta función el uso de dispositivos de computación móvil como *Smartphone* y tabletas. Debido al incremento de la popularidad de estos, se observa la oportunidad del fortalecimiento de los equipos electrónicos orientados a la seguridad, usando los dispositivos de computación móvil como mando de control de una plataforma robótica, la cual enviara una señal de video que se visualizara en los teléfonos inteligentes y tabletas.

Para dar una solución a la problemática se desarrolló una plataforma robótica móvil en la que se integran diversos tipos de hardware proporcionándole conectividad al mando de control por medio de una red Wi-Fi; al mismo tiempo se integran diversos sensores como ultrasónicos que permiten detectar obstáculos y así poder evitarlos para un mejor desplazamiento durante el recorrido de la plataforma robótica, adicionalmente se integró un sistema de geoposicionamiento en el cual permite conocer su ubicación espacial; ya que esta tiene la habilidad de desplazarse dentro de una área determinada. Por último posee la capacidad de captar por medio de video el recorrido de la plataforma y transmitirlo al mando de control para su previa visualización.

En este documento inicialmente se desarrolla una explicación de la base teórica donde muestra todo lo necesario para la elaboración del proyecto, posteriormente se explica con detalle el procedimiento que se realizó para la construcción y

pruebas de la plataforma robótica, el sistema de enlace y la aplicación móvil para el control de la plataforma.

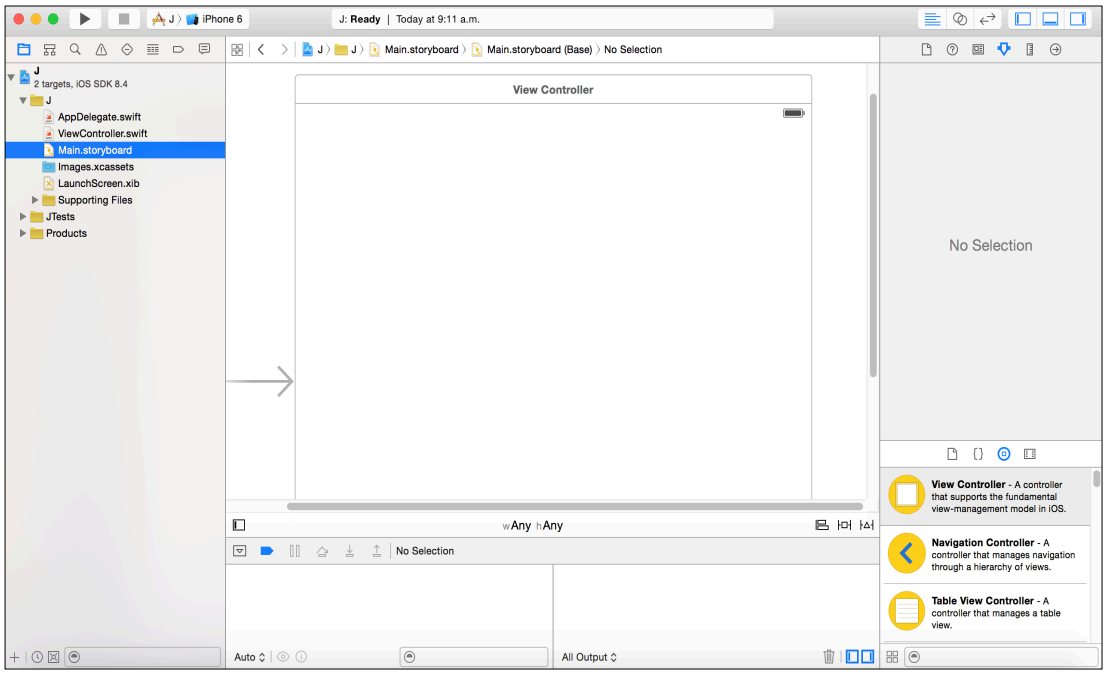
En las pruebas realizadas se hizo un controlador PID de velocidad para el desplazamiento del sistema robótico, se muestran las diferencias de un control realizado sin peso en la plataforma (datos tomados sin fricción en las llantas), y otro control con el peso total de la plataforma robótica (datos tomados con fricción en las llantas) por medio de gráficas. Finalmente se muestra lo concluido de este proyecto.

2 MARCO TEÓRICO

2.1.1 Xcode

Xcode [1] es un entorno de desarrollo que provee todo lo necesario para la realización de aplicaciones, ya sea para sistemas operativos IOS, OSX o watchOS. IOS es el sistema operativo para los dispositivos móviles como iPad, iPhone y iPod, OSX es el sistema operativo de los computadores marca Apple como los MacBook, MacBookPro, iMac, MacBook Air; por último watchOS es el nuevo sistema operativo para el iWatch de Apple. Xcode 6 incluye el lenguaje de programación Swift, un lenguaje innovador con un área de trabajo interactivo, en el cuál, el usuario puede diseñar con facilidad interfaces gráficas con el entorno de programación. *Xcode* proporciona diferentes tipos de proyectos como aplicaciones de una sola vista, tablas, juegos etc. Además, el software *Xcode* provee un simulador que se utiliza para que el usuario puede ejecutar la aplicación que está diseñando y observar el funcionamiento de esta antes de subirla a un dispositivo móvil y así hacer las correcciones necesarias. En la Figura1 se observa el entorno de trabajo de Xcode.

Figura1. Interfaz Xcode



[1] APPLE. IOS Developer Library. Junio 30, 2015. [online]. Disponible en: <https://developer.apple.com/library/ios/documentation/DeveloperTools/Conceptual/WhatsNewXcode/Articles/xcode_6_0.html>

A continuación se explica un poco sobre las áreas de trabajo que *Xcode* utiliza para el desarrollo de aplicaciones; en primer lugar se encuentra el *Main.storyboard* que es el área de trabajo donde se va a implementar todo lo necesario para el desarrollo de la interfaz gráfica de la aplicación. Otra área importante es el *ViewController.swift* donde se escribe la programación necesaria para que ejecute las acciones de la aplicación. *El AppDelegate.swift* es el encargado de administrar la ejecución de las aplicaciones en un segundo plano para que estas funcionen sin necesidad de visualizarlas, además notifica los permisos requeridos que la aplicación necesita para acceder a datos del usuario como geoposicionamiento.

2.1.2 Sistemas operativos de dispositivos móviles

Las funcionalidades de los dispositivos móviles como tablets y teléfonos inteligentes son administradas por sistemas operativos desarrollados por empersas como Google, Apple, BlackBerry; teniendo sus núcleos basados en diferentes tecnologías.

IOS es desarrollado y soportado por Apple, esta empresa implementó *Xcode* que le permite al usuario desarrollar apliaciones con un entorno fácil e intuitivo. IOS se caracteriza por la optimización al Hardware de los dispositivos móviles para asegurar la mayor autonomía del consumo de bateria. Este sistema es programado en lenguaje *Swift* el cual fue desarrollado por Apple para mejorar la experiencia y optimizar todos los procesos de programación de las aplicaciones para IOS[2].

Android junto a IOS son los sistemas operativos más usados en dispositivos móviles a nivel mundial, este es desarrollado y soportado por Google, tiene la capacidad de ser modificado por cualquier usuario ya que su licensia es abierta, por esta razón los diferentes dispositivos móviles que tienen Android poseen apariencia y funcionalidades diferentes. Android está desarrollado en lenguaje de programación Java, por la cual puede ser desarrollada en diferentes sistemas de desarrollo como Android Studio, Eclipce o NetBeans. En la Figura2 se observa algunos de los dispositivos móviles disponibles en el mercado con su respectivo sistema opertivo.

[2] APPLE. IOS Developer Library – Prerelease. Agosto 24, 2015. [online]. Disponible en: <https://developer.apple.com/library/prerelease/ios/documentation/Swift/Conceptual/Swift_Programming_Language/index.html>

Figura2. Dispositivos móviles[3]

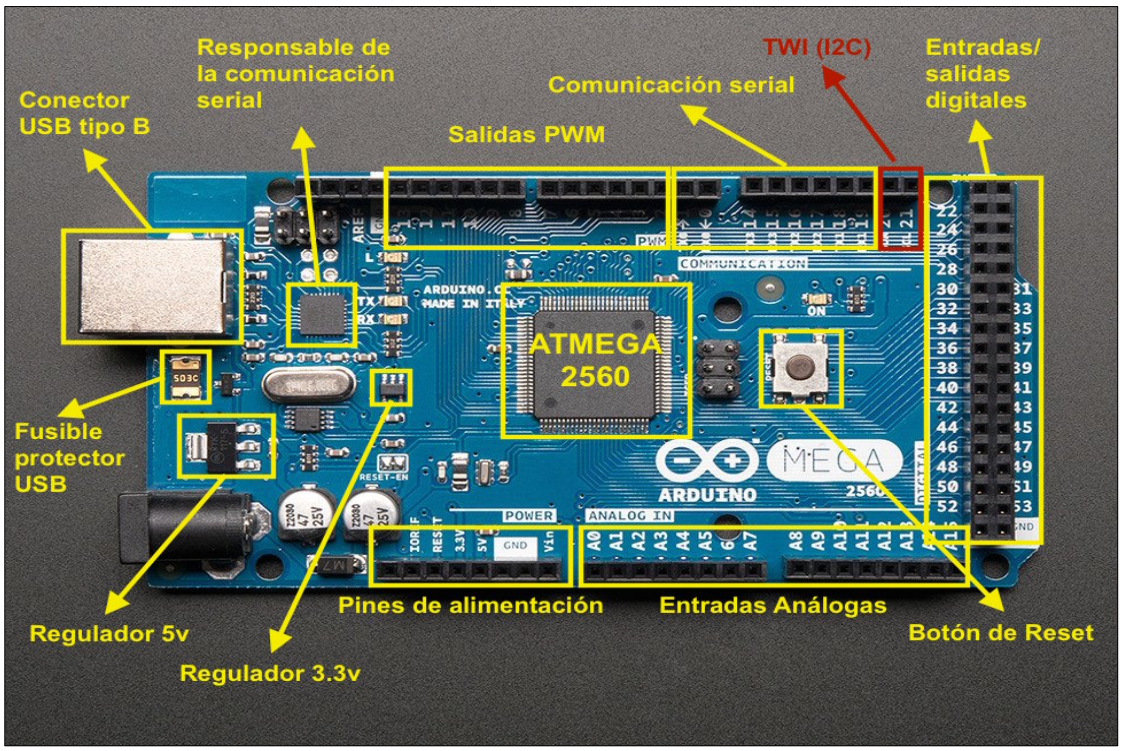
2.1.3 Arduino

Arduino es una sistema de desarrollo de código abierto, consiste en un sistema que cuenta con entradas y salidas analógicas y digitales, posee un entorno de desarrollo basado en el lenguaje de programación Processing. En el mercado existen muchas clases de Arduino, como por ejemplo Arduino uno, Arduino Pro, Arduino Mega, Arduino Cero, Arduino Due, Arduino Yun, Arduino Gemma, Arduino Lilypad, Arduino Lilypad Sample, Arduino Lilypad USB, todos estos pertenecen a la parte tarjetas (Boards), también existen módulos tales como Arduino Motor Shield, Arduino Proto Shield, Arduino Ethernet Shield, Arduino GSM Shield, y por último los kits como lo es el Arduino Starter Kit, Arduino Basic Kit, Matreria 101.

El Arduino Mega (ver Figura3) está basado en una placa Atmega2560, cuenta con 54 pines digitales de entrada / salida (de los cuales 15 se puede utilizar como salidas PWM), 16 entradas analógicas, 4 UARTs (hardware puerto serie), un reloj de 16 MHz, una conexión USB. Estas características hacen que el Arduino Mega sea compatible con muchas *Shield* (son tarjetas que se pueden conectar en la parte superior del Arduino que extienden las cualidades de esta tarjeta de desarrollo) existentes en el mercado.

[3] Tomado de: <http://cdn2.ticbeat.com/wp-content/uploads/2013/12/moviles.png>

Figura3. Arduino Mega



En la Tabla1 se puede observar con mas detalles las especificaciones técnicas del Arduino Mega.

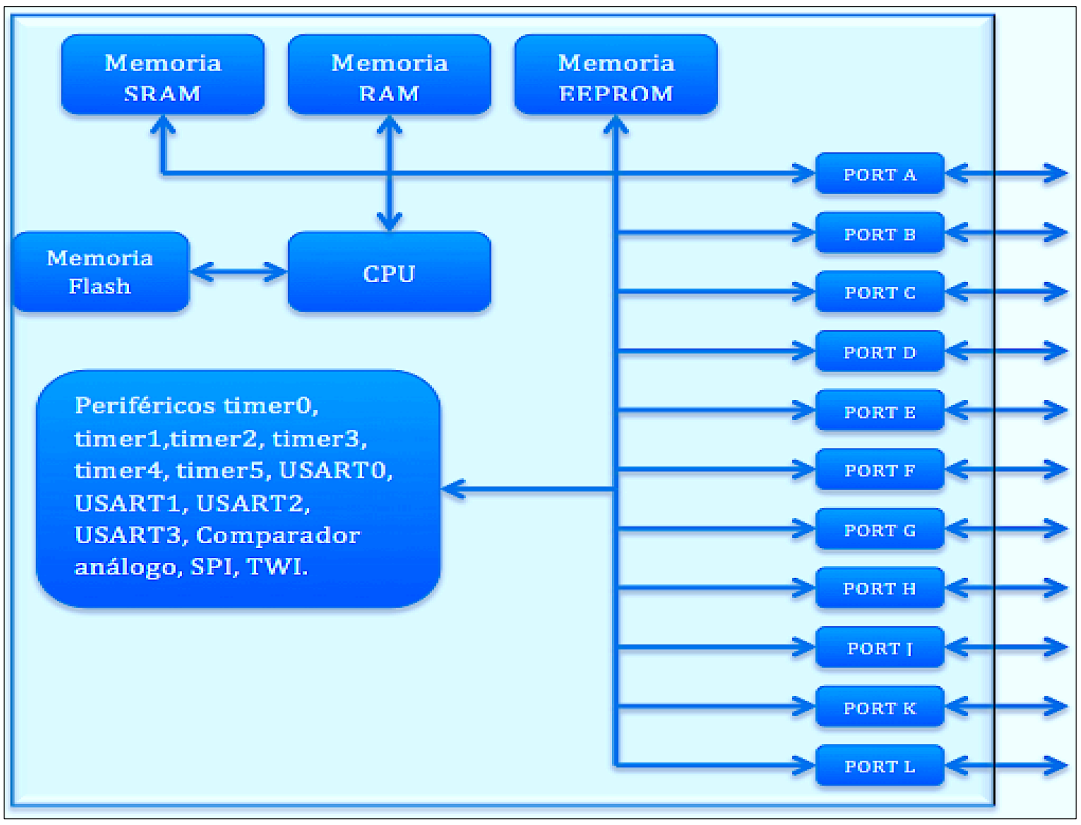
Tabla1. Ficha Técnica[4]

Microcontroladores	Atmega2560
Tensión De Funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
Digital pines I / O	54 (de las cuales 15 proporcionan salida PWM)
Botones de entrada analógica	16
Corriente DC por E / S Pin	20 mA
Corriente DC de 3.3V Pin	50 mA
Memoria Flash	256 KB de los cuales 8 KB utilizado por el gestor de arranque
SRAM	8 KB
EEPROM	4 KB
Velocidad De Reloj	16 MHz
Largo	101.52 mm
Anchura	53,3 mm
Peso	37 g

El microcontrolador Atmega2560 es de alto rendimiento y bajo consumo, combina la memoria ISP Flash 256KB, SRAM de 8KB y EEPROM 4KB. Posee 6 *timers/counters*, convertidor de 10 bits de 16 canales A/D. En la Figura4 se puede ver el diagrama de bloques del Atmega2560.

[4] Tomado de: <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>

Figura4. Diagrama de bloques Atmega2560[5]

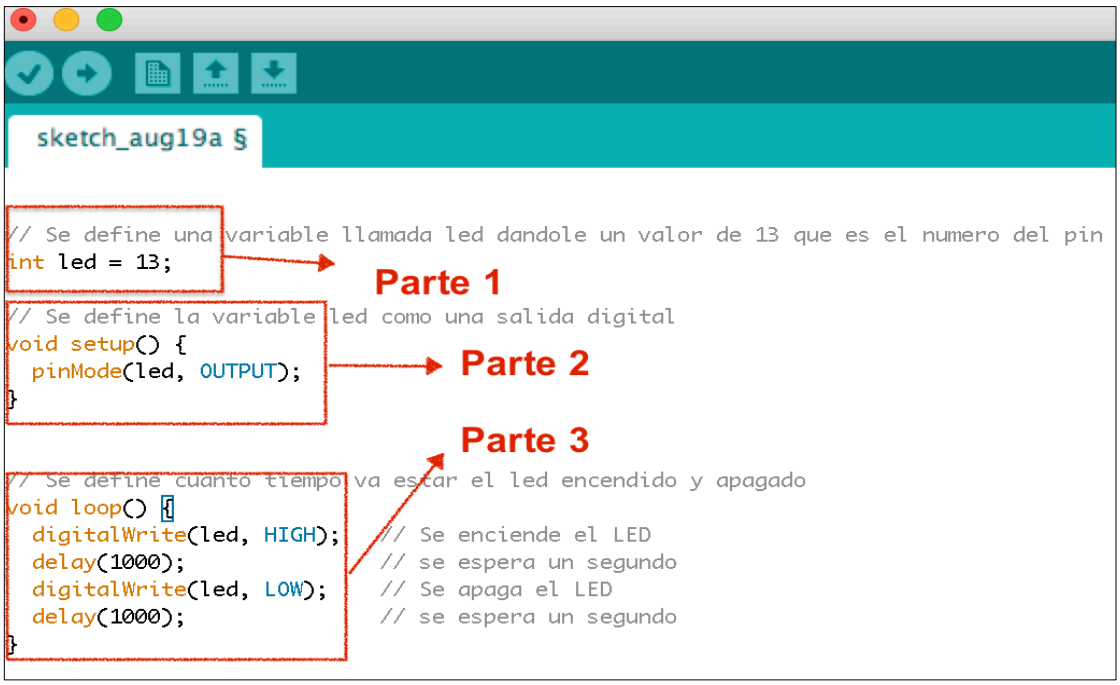


El Arduino se programa utilizando un lenguaje propio basado en Processing, el cual es un lenguaje desarrollado en Java, También es posible programar el Arduino por medio de otros entornos de programación como el que utiliza Matlab, el cual es un paquete de soporte para Arduino que permite comunicar este hardware con Matlab atreves del cable USB[6].

En la Figura5 se puede ver el entorno de desarrollo del Arduino, lo cual cuenta de tres partes, la primera se utiliza para declarar constantes y librerías, la segunda es utilizada para declarar pines como entradas / salidas y también algunas librerías, y la tercera es donde se escribe el programa que va ser ejecutado infinitamente.

[5] Tomado de: <http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_Summary.pdf>
[6] MATHWORKS. Hardware Support. [online]. Disponible en : <http://www.mathworks.com/hardware-support/arduino-matlab.html>

Figura5. Entorno de Programación de Arduino.

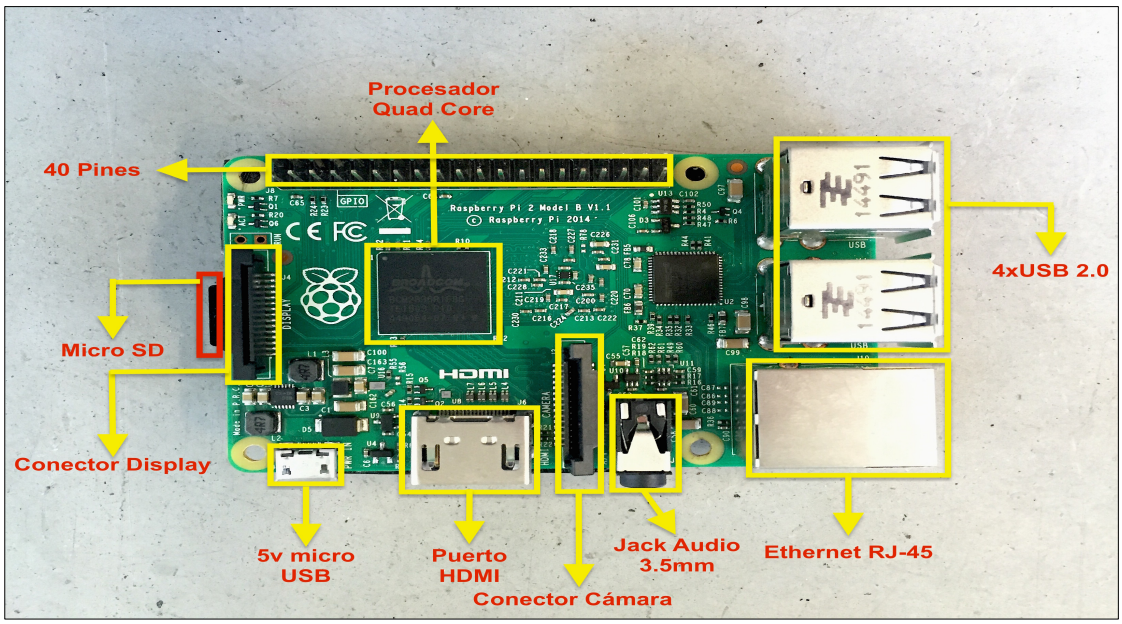


2.1.4 Raspberry Pi

La Raspberry Pi es un microcomputador de bajo costo, como se observa en la Figura6. Su primera versión fue lanzada al mercado en el 2012, por la fundación Raspberry Pi y desde entonces se han desarrollado cinco versiones diferentes con respecto al original, en la Tabla2 se puede observar una comparación entre los cinco modelos existentes en el mercado, con ligeros cambios como aumento de memoria y de conectividad[7].

[7] IEEE SPECTRUM. Create a “Wheel of excuse” With BASIC and the New Raspberry Pi. Octubre 24, 2014. [online]. Disponible en: <http://spectrum.ieee.org/geek-life/hands-on/create-a-wheel-of-excuses-with-basic-and-the-new-raspberry-pi>

Figura6. Raspberry Pi 2 Modelo B



Este microcomputador se basa en un encapsulado Soc BCM2836, en donde se integra el procesador de bajo consumo ARM Cortex-A7 y una GPU VideoCore IV®, Al ser una tarjeta embebida se ve la necesidad de realizar un acceso remoto a la Raspberry Pi para realizar la instalacion y ejecución de algunos paquetes de software sin la necesidad de conectar una pantalla y un teclado. Para realizarlo se puede utilizar una serie de herramientas compatibles con los sistemas operativos que soporta cualquier modelo de Pi, por ejemplo VNC y SSH permiten la visualización del entorno grafico o proporciona un acceso al terminal del sistema, protocolos como SCP y SFTP son utilizados para copiar archivos entre la Raspberry Pi y una computadora portátil o escritorio[8].

[8] RASPBERRY PI FOUNDATION. Remote Access. [online]. Disponible en: < https://www.raspberrypi.org/documentation/remote-access/README.md >

Tabla2. Comparación entre modelos de Raspberry Pi[9]

	Modelo A	**Modelo A+**	**Modelo B**	**Modelo B+**	**Modelo 2 B**
SoC	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2836
CPU	700MHz ARM1176JZF-S	700MHz ARM1176JZF-S	700MHz ARM1176JZF-S	700MHz ARM1176JZF-S	900MHz Quad-core ARM Cortex-A7
GPU	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV
RAM	256Mb	256Mb	512Mb	512Mb	1Gb
USB	1	1	2	4	4
Video	RCA, HDMI	Jack, HDMI	RCA, HDMI	Jack, HDMI	Jack, HDMI
Audio	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI
Boot	SD	MicroSD	SD	MicroSD	MicroSD
Red	-	-	Ethernet 10/100	Ethernet 10/100	Ethernet 10/100
Consumo	300mA /1.5w /5v	400mA /2w /5v	700mA /3.5w /5v	500mA /2.5w /5v	800mA /4w /5v
Alimentación	MicroUSB /GPIO	MicroUSB /GPIO	MicroUSB /GPIO	MicroUSB /GPIO	MicroUSB /GPIO

2.1.4.1 Sistemas operativos

Esta tarjeta embebida es compatible con sistemas operativos basado en el núcleo de Linux, distribuidos principalmente por la web de la fundación Raspberry Pi, pero desarrollados por diferentes grupos de programadores. Entre sus diferentes versiones se destaca principalmente Raspbian, este es un sistema operativo

[9] Tomado de: <https://www.raspberrypi.org/products/>

gratuito basado en Debian, optimizado para el hardware de este microcomputador; este ofrece más de 35000 paquetes pre compilado para una instalación más sencilla de otras clases de software[11].

Raspbian al ser un sistema basado en Debian puede gestionar las instalaciones y eliminación de cualquier software utilizando APT (herramienta avanzada de empaquetado). Por esta razón cualquier software en Debian que funcione en arquitectura ARM, se puede encontrar disponible en Raspbian[12].

Entre los demás sistemas operativos se encuentra Pidora, el cual es una versión optimizada de Fedora Remix para los procesadores ARMv6, en donde se encuentra el procesador integrado en la Raspberry Pi; entre sus cualidades se puede encontrar la integración de algunos lenguajes de programación como C y Perl, compilado para optimizar el hardware de Raspberry Pi, proporcionándole un arranque más rápido y la capacidad de utilizar los pines para el manejo de hardware externo. Pero principalmente se caracteriza por su tamaño reducido, acortando los tiempos de descarga e instalación[13].

También se encuentra sistemas operativos orientados a convertir el microcomputador en un dispositivo reproductor multimedia, OpenELEC y Raspbmc son los encargados de realizar estas acciones. Ambos traen XBMC incorporado, el cual es un software multimedia y centro de entretenimiento. Además incorporan una serie de codificadores que permiten reproducir y visualizar contenido en 1080p[14].

Por último se encuentra el sistema operativo RISC OS, de fabricación británica, diseñado especialmente para los procesadores ARM, diseñado por el equipo que desarrollo el ARM original. Se caracteriza por ser rápido y eficiente, pero sobretodo se destaca entre los demás sistemas para Raspberry Pi por ser un software independiente de Linux y de Windows, por tal motivo tiene unas características únicas

2.1.4.2 Lenguajes de programación en Raspberry Pi

Raspbian al ser el sistema operativo más usado entre los usuarios de Raspberry Pi es precargado con múltiples compiladores de programación para manejar diferentes lenguajes. Python es un lenguaje de programación poderoso y fácil de

[11] BYTEMARK HOSTING. Welcome to Raspbian. [online]. Disponible en : < http://raspbian.org/FrontPage >
[12] RASPBERRY PI FOUNDATION. APT. [online]. Disponible en : < https://www.raspberrypi.org/documentation/linux/software/apt.md >
[13] PIDORA 2014. Raspberry Pi Fedora 2014.[online]. Disponible en: < http://pidora.ca/pidora/releases/20/release-announcement.txt >
[14] KODI. Raspberry Pi. Agosto 18, 2015. [online]. Disponible en: <http://kodi.wiki/view/Raspberry_Pi>

usar, ya que facilita el proceso de leer y escribir; junto a la Raspberry Pi se puede diseñar software que se comunique al mundo exterior por medio de algún hardware. Dentro de Python se puede imprimir algún dato, realizar operaciones matemáticas, igualar variables, además permite comentar las líneas necesarias para un posterior entendimiento del código, también acepta las iteraciones para repetir líneas de código necesarias según sea el caso[15].

2.1.5 Lenguaje PHP

Hypertext Preprocessor o PHP es un lenguaje de propósito general de código abierto, usado principalmente en el desarrollo de contenido web, esto es debido a que dentro de un código HTML se puede ejecutar *script* escritos en PHP como se puede observar en la Figura7; para realizar esta acción es necesario que las funciones de PHP estén dentro de las etiquetas <?php y ¿>, las cuales indican el comienzo y el final de la ejecución del *script*[16].

Figura7. Código php embebido a HTML[17]

```
<html>
 <head>
  <title>Prueba de PHP</title>
 </head>
 <body>
 <?php echo '<p>Hola Mundo</p>'; ?>
 </body>
</html>
```

Una de las características principales de un *script* de PHP dentro de un HTML es en el momento de la ejecución, la cual se realiza en el servidor generando como respuesta un HTML puro al cliente. Este lenguaje de programación se destaca por su rápido aprendizaje y la facilidad de uso ya que este permite la libre elección del sistema operativo y la clase de servidor, en donde se va desarrollar y procesar los *script*[18].

[15] RASPBERRY PI FOUNDATION. Python. [online]. Disponible en: <https://www.raspberrypi.org/documentation/usage/python/README.md>
[16] THE PHP GROUP. Su primera página con php. [online]. Disponible en: <https://secure.php.net/manual/es/tutorial.firstpage.php>
[17] Tomado de: <https://secure.php.net/manual/es/tutorial.firstpage.php>
[18] THE PHP GROUP. ¿Qué pude hacer php?. [online]. Disponible en: <https://secure.php.net/manual/es/intro-whatcando.php>

Aunque su funcionalidad más destacable de PHP es la capacidad de permitir a una página web tener acceso a una base de datos, debido a que este tiene soporte a una gran cantidad de gestores de bases datos como MySQL, DB++, etc; se puede utilizar este lenguaje para comunicarse con otros servicios utilizando protocolos como son el COM, HTTP, POP3; además puede generar archivos como PDF, XML, imágenes y los puede almacenar en el sistema de ficheros del servidor[19].

Como en cualquier lenguaje de programación PHP tiene la capacidad de manejar diferentes tipos de variables, las cuales son declaradas por el signo dólar seguido del nombre de la variable; al ejecutar el *script* reconoce la minúsculas y mayúsculas de las variables, en la Figura8.se observa algunos de los tipos de datos más usados en PHP[20].

Figura8. Descripción de los tipos de variable en PHP[21]

TIPO DE DATO	DEFINICIÓN

integer	Los integers, o enteros, pueden tener distintos valores numéricos enteros que se expresan con diferentes notaciones.
	$variable = 18; // Número entero positivo
	$variable = -18; // Número entero negativo
	$variable = 0x12; // Notación hexadecimal, en principio no la utilizaremos

float o double	Este tipo de datos son los números de punto flotante a los que normalmente llamamos "números decimales", por ejemplo, 9.876. Ambos tienen mucha precisión, pero double es el más preciso (con más decimales). La sintaxis para utilizarlos es bastante simple: $variable = 9.876;
string	El tipo de datos string, también conocido como cadena de caracteres, se expresa con la siguiente sintaxis: $variable = "Yo soy una cadena";
boolean	Se trata de un tipo lógico. Sus posibles valores son true (verdadero) o false (falso). $variable = true; $variable = false;

2.1.6 APACHE

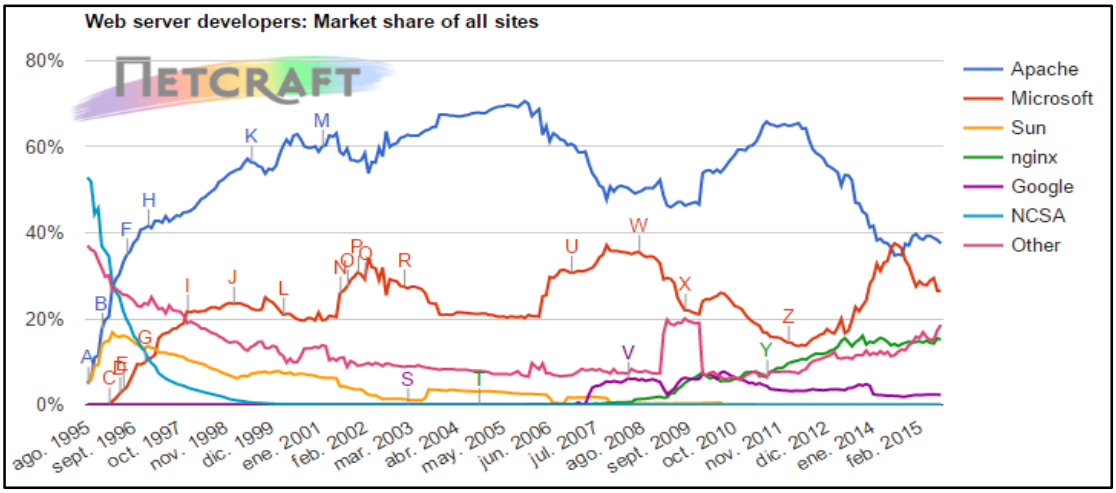
Es un servidor web soportado por el Apache Software *Foundation*; apache se originó cuando los responsables del servidor web creado por la NCSA (*National*

[19] THE PHP GROUP. ¿Qué es php?. [online]. Disponible en: <https://secure.php.net/manual/es/intro-whatis.php>
[20] APR. Tipos de variables en PHP. Declaración y asignación. Sentencia echo: insertar texto en el HTML (CU00816B). [online]. Disponible en: <http://www.aprenderaprogramar.com/index.php?option=com_content&id=544:tipos-de-variables-en-php-declaracion-y-asignacion-sentencia-echo-insertar-texto-en-el-html-cu00816b&Itemid=193>
[21] Tomado de: <http://www.aprenderaprogramar.com/index.php?option=com_content&id=544:tipos-de-variables-en-php-declaracion-y-asignacion-sentencia-echo-insertar-texto-en-el-html-cu00816b&Itemid=193>

Center for Super Computing Applicationes) abandonan su suporte, dejándolo libre para que los usuarios comparten información sobre el servidor por medio de parches[22].

Los servidores apache se caracterizan por ser desarrollados bajo una arquitectura modular, la cual le brinda la posibilidad a los administradores de sitios web seleccionar los módulos requeridos para mejorar la personalización del servidor, mejorando así características como la estabilidad o compatibilidad, además permite a los programadores independiente desarrollar sus propios módulos. En la Figura9 se observa que es uno de los servidores más usados en el internet, debido a que este es multiplataforma, gratuito, con un gran nivel de seguridad y óptimo rendimiento[23].

Figura9. Tasa de mercado mundial de servidores[24]



2.1.7 Cámara de video

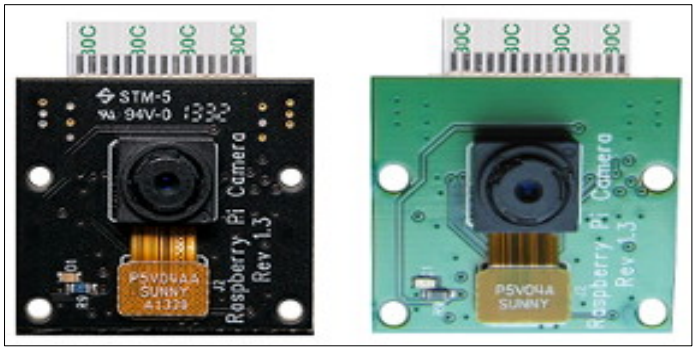
La Raspberry pi tiene un puerto CSI en donde se comunica por medio de un cable *ribbon* a una pequeña *board* en donde se encuentra una cámara basada en un sensor *CMOS Omnivision*[25], la cual consume 250mA para su funcionamiento. Esta cámara tiene la capacidad de tomar fotos de 5 MP con una resolución máxima de

[22] THE APACHE SOFTWARE FOUNDATION. About Apache. [online]. Disponible en: <http://www.apache.org/history/timeline.html>
[23] THE APACHE SOFTWARE FOUNDATION. Módulos de MultiProcesamiento (MPMs). [online]. Disponible en: < http://httpd.apache.org/docs/2.0/mpm.html >
[24] Tomado de: <http://news.netcraft.com/archives/category/web-server-survey/>
[25] OMNIVISION. 5-megapixel ¼" Image Sensor with 1.4 μm Omni BSI Technology Offering HD Video. Enero, 2010. [online]. Disponible en: < http://www.ovt.com/uploads/parts/OV5647.pdf>

2592x1944, además permite la grabación de video a 1080p a una velocidad de 30 *frames* por segundo[26].

Actualmente existe dos versiones de la cámara de Raspberry pi, las cuales se pueden observar en la Figura10. La que se observa al lado derecho es conocida como *pi camera* y la otra como *pi camera NoIR*, La diferencia entre las dos es la inexistencia de los filtros infrarrojos, la cual permite a la versión *NoIR* visualizar cualquier entorno oscuro por medio de iluminación infrarroja[27], este es el principal motivo por la que no se recomienda para usos de fotografía en condiciones normales de luminosidad, ya que no es capaz de captar toda la gama de colores[28], esto se puede apreciar en la Figura11.

Figura10. Versiones comerciales de cámaras Raspberry pi[29]



[26] RASPBERRY PI FOUNDATION. WHAT IS THE CAMERA BOARD?. [online]. Disponible en: < https://www.raspberrypi.org/help/faqs/#camera>
[27] RASPBERRY PI FOUNDATION. PI NOIR CAMERA. [online]. Disponible en: < https://www.raspberrypi.org/products/pi-noir-camera/>
28 RASPBERRYCONNECT. Raspberry Pi NoIR Infrared camera module. Noviembre 2, 2013. [online]. Disponible en: <http://www.raspberryconnect.com/hardware-add-ons/item/148-raspberry-pi-noir-infrared-camera-module>
[29] Tomado de :< http://www.raspberryconnect.com/hardware-add-ons/item/148-raspberry-pi-noir-infrared-camera-module>

Figura11. Comparación de una imagen tomada por las dos cámaras de Raspberry pi[30]

Las dos cámaras pueden ser controladas por comandos ejecutados directamente en el terminal[31], adicionalmente existe una librería en Python que permite diseñar completos scripts para manipular las acciones de la cámara; esta permite añadirle efectos a las imágenes tomadas, modificar parámetros como el brillo, saturación, entre otros[32].

2.1.8 GPS

El Sistema de Posicionamiento Global (GPS) es un servicio propiedad de los EE.UU. que proporciona a los usuarios información sobre posicionamiento, navegación y cronometría. Este sistema está constituido por tres segmentos: El segmento espacial, el segmento de control y el segmento del usuario[33].

- **El segmento espacial**
  Está constituido por satélites y constelaciones de diferente número de satélites o por conjuntos de plataformas estratosféricas ubicadas en posiciones fijas en el espacio, diseñado para establecer la radiocomunicación[34].

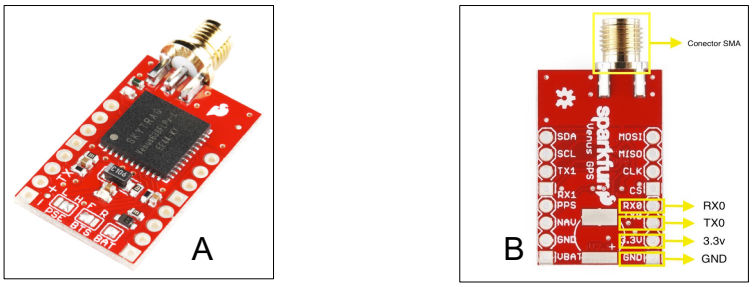
[30] Tomado de: < http://pimylifeup.com/raspberry-pi-camera-vs-noir-camera/>
31 RASPBERRY PI FOUNDATION. RASPICAM COMMAND. [online]. Disponible en: < https://www.raspberrypi.org/documentation/usage/camera/raspicam/README.md>
[32] RASPBERRY PI FOUNDATION. PYTHON PICAMERA. [online]. Disponible en: < https://www.raspberrypi.org/documentation/usage/camera/python/README.md >
[33] SPARKFUN. SparkFun Venus GPS with SMA Connector. [online]. Disponible en: < https://www.sparkfun.com/products/11058>
[34] GPS.GOV. Space Segment. Agosto 17, 2015. [online]. Disponible en: < http://www.gps.gov/systems/gps/space/ >

- **Determinación de Posiciones del GPS**
  Las posiciones se obtienen mediante la distancia que hay hacia los satélites visibles, este proceso es llamado "trilateración". El momento de transmisión de la señal al satélite, con el momento de la señal de recepción, esa diferencia entre esos dos tiempos da la distancia que se tardo la señal del satélite al receptor[35].

- **Segmento de control**
  Son estaciones que están ubicadas en diferentes partes de la tierra y se encargan de monitorear cada satélite analizando las señales emitidas, y a su vez actualizando los datos de navegación, correcciones de reloj de los satélites etc. Estas estaciones se ubican estratégicamente cerca al plano ecuatorial[36].

- **Segmento usuario**
  El segmento usuario se encuentra en los receptores GPS, estos son los encargados de registrar las señales emitidas por los satélites para hacer el cálculo de posición tomando como base la velocidad de la luz y el tiempo de que tarda la señal en viajar, así se obtienen las distancias que hay de cada satélite al receptor, para obtener una distancia buena, al menos se deben captar 4 satélites para hacer el cálculo adecuado de posición[37].

Figura12. GPS Venus Figura12A, Pines GPS Venus Figura12B[38]



El GPS Venus de SparkFun (ver Figura12A) incluye un conector SMA para conectar una antena externa, esta placa requiere una tensión de 3.3V regulados para un correcto funcionamiento, el consumo máximo de esta tarjeta es de hasta 90mA y el mínimo consumo es alrededor de 60mA.

[35] INSTITUCIÓN NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA. Sistema de Posicionamiento Global (GPS). [online]. Disponible en: < http://www.inegi.org.mx/geo/contenidos/geodesia/gps.aspx?dv=c1 >
[36] GPS.GOV. Control Segment Elements. Marzo 27, 2015. [online]. Disponible en: < http://www.gps.gov/systems/gps/control/ >
[37] INSTITUCIÓN NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA. Op. cit., p. 1.
[38] Tomado de: <https://www.sparkfun.com/products/retired/9133>

Además el GPS SparkFun cuenta con un segundo puerto serie (RX1, TX1) y la interfaz I2C (SDA, SCL) (ver Figura12B), pero solo se utiliza el primer puerto serie (RX0, TX0) y la alimentación del GPS (3.3V, GND).

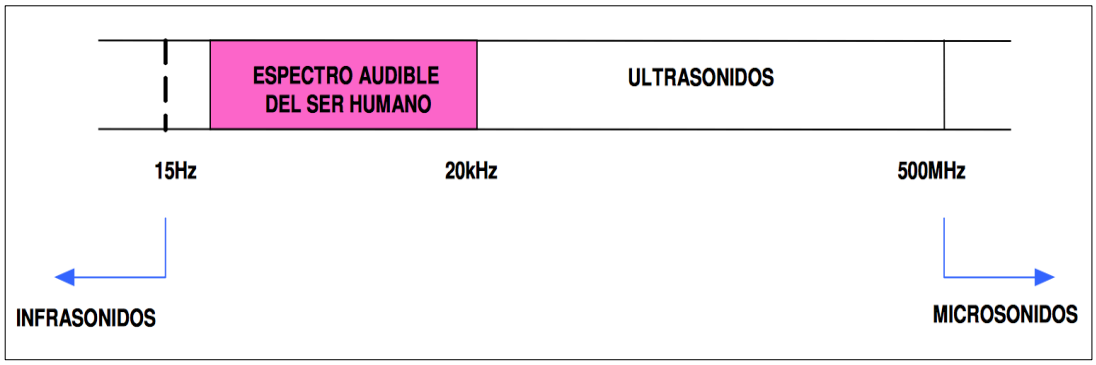
2.1.9 Sensores ultrasónicos

Los sensores de ultrasonido pueden usarse para detectar piezas u obstáculos sin contacto en muchas áreas de la automatización. Estos sensores se utilizan frecuentemente cuando se necesita medir distancias en el aire, ya que no solo detectan objetos, también pueden indicar y evaluar la distancia absoluta entre ellos y el objeto[40].

Estos sensores pueden detectar cualquier objeto, ya sea en estado sólido, líquido, granular o polvo. El material puede ser transparente o coloro, liso o rugoso. Una de las desventajas de estos sensores es que se alteran con el cambio de temperatura, un material como hierro fundido puede dispersar la onda sonora debido al cambio de temperatura que se maneja en el material, llegando a un dato de medición errónea; los sensores de ultrasonido están diseñados para trabajar en condiciones ambientales normales[41].

Los sensores de ultrasonido operan en un rango de unos 20 KHz hasta un limite superior de aproximadamente 500 MHz (ver Figura13.)

Figura13. Rango de operación del sonido[42]



[40] CONTRINEX. Sensores de ultrasonidos. [online]. Disponible en: < http://www.contrinex.com/file-download/myPDFFiles/products/ultra/documentation/es/sensores_ultra-sonicos.pdf >
[41] Ibid., p. 4.
[42] Tomado de: <http://www.marcombo.com/Descargas/9788426715753/SENSORES/TEMAS/SA%20TEMA%2010-ULTRASONIDOS.pdf >

En la Figura14 se puede observar el sensor de ultrasonido SRF05, lo cual es un módulo que integra un receptor y emisor en una salo *board*, debido a su baja zona muerta de 1.7 cm. En la Tabla3 se puede ver las características del sensor ultrasónico SRF05[43].

Figura14. Sensor ultrasónico SRF05[44]

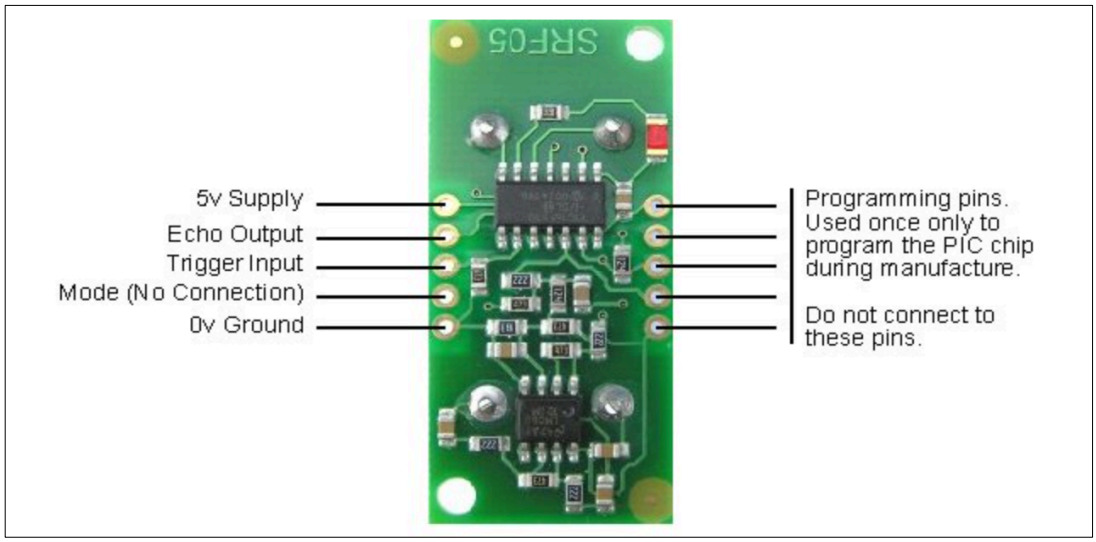


Tabla3. Ficha Técnica SRF0545[45]

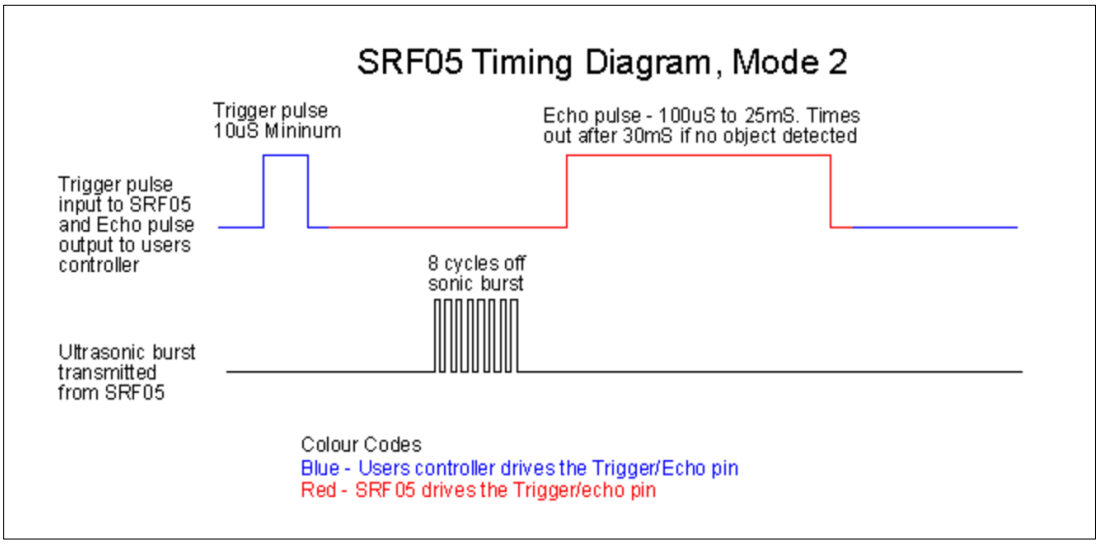
Rango	1.7 cm a 4m
Voltaje	5v
Consumo	4 mA
Frecuencia	40 KHz
Tamaño	43 mm, 20 mm, 17mm

En la Figura15 se puede observar el diagrama de tiempo del sensor ultrasónico SRF05; el trigger es el pulso de entrada de $10\mu s$ como mínimo para que envie 8

[43] Tomado de: <http://www.iesluisdelucena.es/dpp/docs/presentaciones/El_sensor_de_ultrasonidos_sfr05_rev091210.pdf>
[44] Tomado de:<http://www.iesluisdelucena.es/dpp/docs/presentaciones/El_sensor_de_ultrasonidos_sfr05_rev091210.pdf>
[45] Tomado de: <http://www.dynamoelectronics.com/index.php?page=shop.product_details&flypage=dynamo.tpl&product_id=626&category_id=78&option=com_virtuemart&Itemid=58>

ciclos de ultrasonido a 40 KHz, si se detecta un objeto, se escucha un eco, y tan pronto se detecta baja la línea de eco a cero, Por lo tanto la línea de eco es un pulso cuya anchura es proporcional a la distancia del objeto.

Figura15. Diagrama de tiempo sensor SRF05[46]



Si trascurrido $30\ ms$ la onda de sonido no es interferida por algún objeto, esta vuelve a cero inmediatamente.

2.2 ESTADO DEL ARTE

Investigando sobre los robots que se encargan de vigilar campos peligrosos o comerciales, además de supervisar alguna área determinada, se encontraron los siguientes robots comerciales y algunas tesis de pregrado, maestría y doctorado que fueron útiles para el desarrollo del presente trabajo.

- **Medusa Robots**[47].

Los AUV (vehículo submarino autónomo) son dispositivos remotos a distancia usado por diferentes países como Estados Unidos y Reino Unido, utilizado en

[46] <http://www.robot-electronics.co.uk/htm/srf05tech.htm>
[47] KONGSBERTG. Norwegian University of Technology and Science purchases REMUS 100 Mayo 12, 2012. [online]. Disponible en: <http://www.km.kongsberg.com/ks/web/nokbg0238.nsf/AllWeb/B1A85BBB450D17A6C1257A00002CA7BD?OpenDocument>

aplicaciones civiles y militares. Uno de ellos es el REMUS 100, su diseño consiste en un torpedo con hélice en la parte posterior, como se puede observar en la Figura16. Su principal característica son los 100 m que puede sumergir. Utilizado frecuentemente por la marina de los Estados Unidos y el instituto de oceanografía Woods Hole, en funciones de supervisión de un área determinada. Además incorpora una serie de instrumentos como un ADCP (Perfilador de Corrientes Acústico Doppler) y un sensor de dispersión de luz.

Figura16. Remus 100[48]

En este artículo se discute la funcionalidad de los nuevos diseños de un UAV, los cuales son inspirados en las formas de animales acuáticos como son las medusas, este dispositivo trata de emular los movimientos de estos celentéreos. Un ejemplo de estos UAV es el fabricado por Festo, el AquaJelly posee 8 aletas distribuidas en una circunferencia, estas aletas son instaladas alrededor del cuerpo, además son accionadas por un motor eléctrico, ubicado en el centro del dispositivo.
Los UAV diseñados con forma y funcionalidades de medusas se destacan en la eficiencia de la locomoción, la forma adecuada y la posibilidad de incorporar alguna carga como sensores. Variando así su rendimiento y consumo energético.

- **Diseño de un robot explorador para tuberías de alcantarillado con cámara integrada[49].**

[48] Tomado de: < http://www.km.kongsberg.com/ks/web/nokbg0238.nsf/AllWeb/B1A85BBB450D17A6C1257A00002CA7BD?OpenDocument>
[49] HUGO GUERRERO PABON. DISEÑO DE UN ROBOT EXPLORADOR PARA TUBERIAS DE ALCANTARILLADO CON CAMARA INTEGRADA. PROYECTO DE TRABAJO DE GRADO PARA OPTAR POR TITULO DE INGENIERO MECATRÓNICO. PAMPLONA COLOMBIA. UNIVERSIDAD DE PAMPLONA 2007. 5.

En todas las ciudades tienen un sistema de aguas residuales, lo cual consta de alcantarillados y tuberías que conducen las aguas negras a ciertos puntos permitidos para su respectivo desecho. Ciertamente estos tubos con el paso del tiempo se acumulan suciedades, provocando así un taponamiento del flujo del agua y posibles rupturas del tubo en algunos puntos, los cuales pueden ser perjudiciales para el ser humano, ya que puede contener gases tóxicos.

Es muy importante estar monitoreando este sistema de evacuación de las aguas negras para realizar su respectivo mantenimiento preventivo y posteriormente correctivo.

Para solucionar dicho problema se hace la necesidad de desarrollar un dispositivo robótico, el cual recibe el nombre de HEXPLOBOT, es un robot teleoperado por medio de radiofrecuencia, está comprendido por un sistema de control y otro de exploración en interiores de tuberías de alcantarillado. Dentro de sus funciones se encuentra la localización de taponamiento y roturas, las cuales producen fugas de cualquier fluido.

HEXPLOBOT tiene un diámetro de aproximadamente 20.32 cm, el cual es el mínimo diámetro de las tuberías de la ciudad de pamplona. Su desplazamiento completamente horizontal y controlado por servomotores lo incapacita en recorrer todo el sistema de alcantarillado, debido a que cada 50 o 100 metros se encuentran pozos recolectores en donde posiblemente se encuentran rampas con diferentes ángulos de inclinación.

Se diseñó por medio del software Solid Edge donde se puede construir con gran facilidad la estructura del robot y dimensionar cada parte respectivamente. Además es controlado por radio frecuencia por módulos de AUREL, los cuales envían una señal a la computadora, desde aquí en una aplicación en Visual Basic se puede enviar los comandos para controlar los movimientos de HEXPLOBOT además tiene una cámara que se controla desde la aplicación. Esta envía las imágenes durante el recorrido por la tubería a un receptor LCD.

- **Prototipo de Robot Móvil Teleoperado[50].**

Los robots son muy importantes cuando se tiene un trabajo de bastante riesgo donde el ser humano no puede exponerse, para solucionar este problema se desarrolló este proyecto, este robot se basa en un robot cuadrúpedo comercial, siendo modificado en los tamaños de sus componentes.

[50] CHÁVEZ GONZÁLEZ , Manuel Alberto. Prototipo de Robot Móvil Teleoperado. Tesis para obtener el grado de Maestría en Tecnología Avanzada. Querétaro. Instituto Politécnico Nacional. 2012. I.

Sus extremidades se encuentran unidas por pares brindándole una mayor estabilidad al momento de desplazarse, además estas sujetadas por barras las cuales les permite un mejor movimiento cuando sus patas se acercan al cuerpo. Debido a la simetría de los componentes se simplifica el análisis de movimiento y el análisis de posición.

Además de la translación, el robot puede rotar sobre su propio eje. Esta rotación puede ser de 360 grados en ambas direcciones, resultando como una ventaja para su labor de inspección, cuenta con dos motores, uno para cada movimiento. Cuenta con una cámara inalámbrica con una visión panorámica, esta está ubicada en la tapa superior del robot móvil para obtener un mejor ángulo de visión.

En la estación de control se cuenta con un joystick inalámbrico el cual es capaz de controlar los cuatro movimientos del robot. En ese mismo sitio se cuenta con la información visual del lugar monitoreado, siendo visualizado en su computadora personal o en un monitor en tiempo real.

- **Optimus Max**[51]

La mayoría de los robots poseen sensores de ultrasonidos que detectan obstáculos en el camino y así lograr esquivarlos sin cocharsen, alguno de estos robots son operados de forma inalámbrica por medio de un control remoto. En este proyecto se desarrolla un robot móvil controlado por un dispositivo móvil como el iPad, este robot tiene caracteristicas como sensores de utrasonidos para la detección de obstáculos, una brújula para saber la orientación del robot y un módulo GPS lo cual indica en que lugar se encuentra el sistema robótico. Además este robot llamado “Optimus Max”, hace una conexión a internet por medio de una aplicación realizada en el iPad, lo que permite el control del mismo desde cualquier parte del mundo.

Optimus Max tiene la capacidad de establecer dos tipos de conexiones, la primera es por via internet, la segunda es controlar por conexión punto a punto el sistema robótico. Cuando es controlado via internet se tiene acceso a los datos de los sensores de ultrasonidos, brújula y GPS, cuando se controla punto a punto, solo tiene la capacidad de manejar el direccionamiento del robot (adelante, atrás, izquierda, derecha, parar). Este robot se puede observar en la Figura17.

Este proyecto es la base para este nuevo denominado “DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA ROBÓTICA MÓVIL ORIENTADO A LA VIGILANCIA Y CONTROLADO POR DISPOSITIVOS MÓVILES”, lo cual se

[51] PÁEZ ARDILA , Diego Ricardo PÁEZ ARDILA, Diseño e implementación de una plataforma hardware y software para el control de un robot móvil utilizando el iPad, Trabajo de grado para optar a título de Ingeniero Electrónico, Bucaramanga Colombia. Universidad Pontificia Bolivariana Seccional Bucaramanga, 2012, p. 28.

mejaron varias cosas como la estructura, permitiendo la capacidad de recorrer terrenos difíciles como lodo, arena, piedras, pasto entre otros. Además el sistema robótico tiene la capacidad de permitir la visualización de video captada por la cámara de la Raspberry Pi y este video se puede ver en la aplicación. Otra mejora es que este robot tiene un control PID en los motores, lo que permite una velocidad constante en las cuatro ruedas algo que no tenia Optimux Max; posteriormete el sistema robótico se puede manejar tanto con el sistema operativo IOS como Android, mientras que Optimux Max solo era en IOS específicamente en iPad.

Figura17. Optimus Max.

3 ESTRUCTURA DEL PROYECTO

3.1 OBJETIVO GENERAL

Construir un robot de vigilancia controlado por dispositivos móviles utilizando dispositivos de computación embebida y comunicación inalámbrica para la transmisión y recepción de diversos datos.

3.2 OJETIVOS ESPECÍFICOS

- Implementar un software para comunicar la plataforma móvil con el dispositivo de computación embebida y con el dispositivo móvil.

- Diseñar la aplicación de control en sistemas operativos de dispositivos móviles con una interfaz gráfica amigable con el usuario.

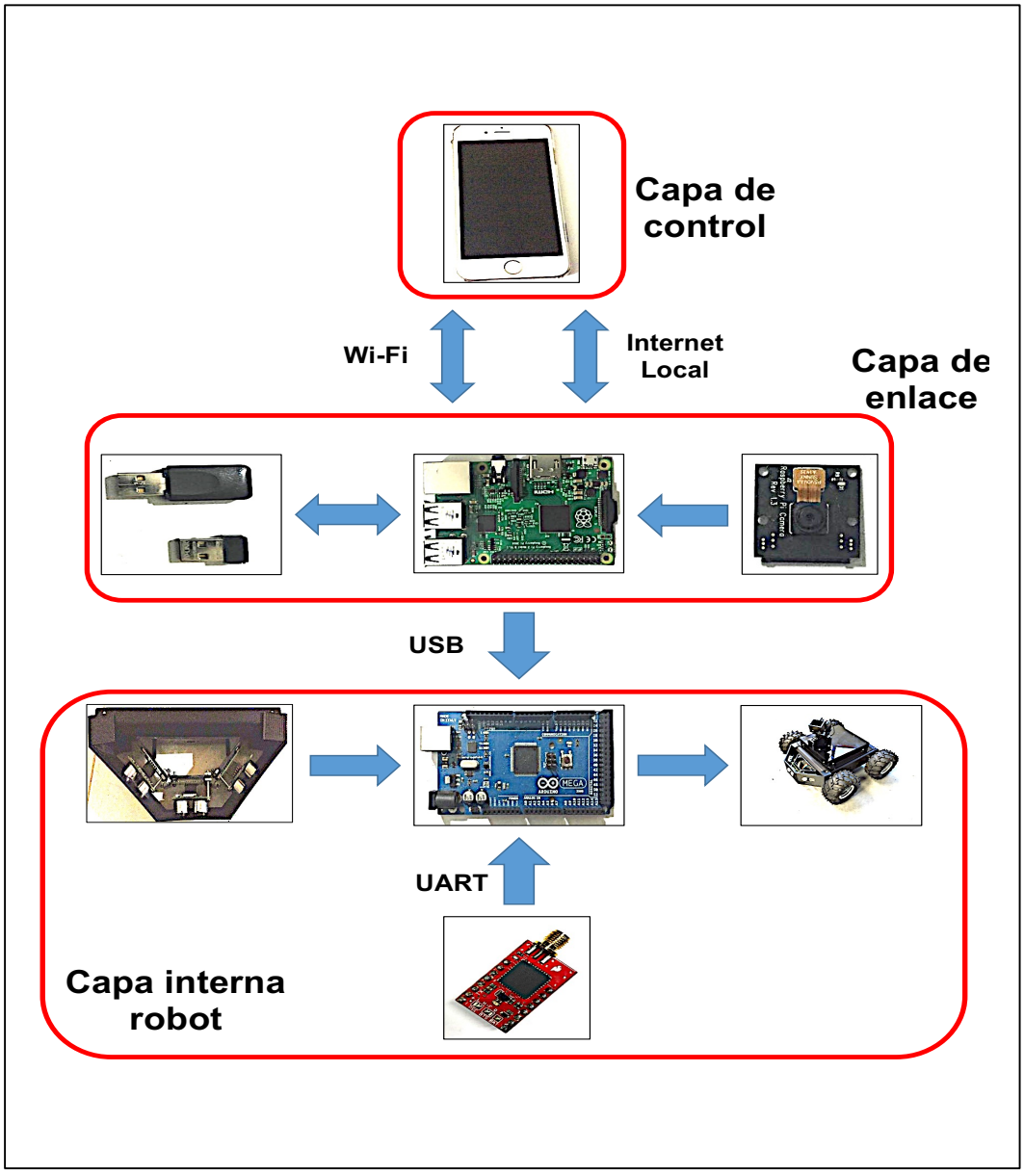
- Controlar los movimientos de la plataforma móvil usando la aplicación de control.

- Visualizar la señal de video en tiempo real, del entorno o área de posicionamiento del robot.

4 DISEÑO E IMPLEMENTACIÓN DEL HARDWARE

El sistema diseñado se compone de varios módulos como se puede ver en la Figura18. La primera parte (Capa de enlace) se compone de una Raspberry Pi que es la encargada del envío y recepción de datos, esto se hace por medio de dos módulos *Wireless*, un módulo encargado de crear una red Wi-Fi internet local, el otro de crear una red punto a punto.

Figura18. Diseño del sistema



Por otra parte (Capa interna robot), aquí se encuentra la parte móvil la cual está compuesta por un Arduino Mega que se encarga del control de los cuatro motores que necesita para trasladarse, adicionalmente cuenta con tres sensores de ultrasonido SRF05 encargados de la detección de obstáculos en el camino, además posee un módulo GPS Venus que se utiliza para obtener las coordenadas de posicionamiento de la plataforma robótica.

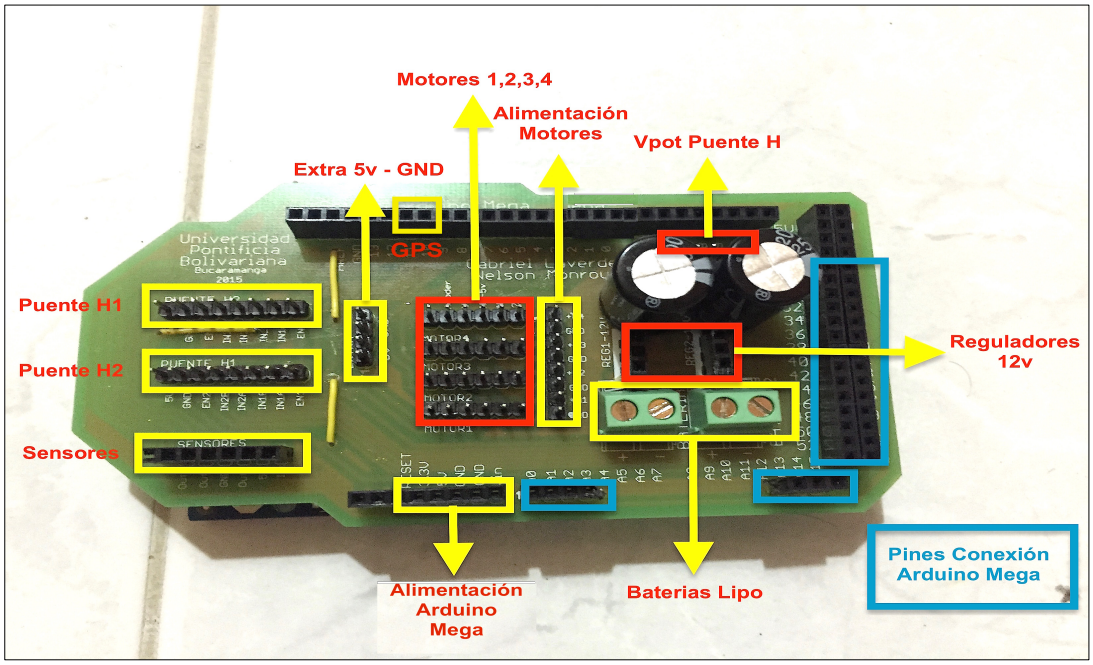
Se utiliza la cámara de la Raspberry Pi 2 para la visualización del recorrido de la plataforma robótica, la cual tiene una resolución de 5 MP con una grabación HD 1080P; la visualización es en tiempo real por vía Wi-Fi o internet local.

La plataforma robótica se controla por medio de una aplicación móvil (Capa de control), donde el usuario elige el tipo de conexión que desee enlazar con el dispositivo, ya sea vía internet local o punto a punto. En la primera se necesita de una dirección IP ingresada por el usuario la cual será la IP de la red WI-Fi donde se halla conectado el dispositivo móvil, este tipo de conexión ofrece una ventaja y es el alcance que se obtiene para controlar la plataforma robótica ya que depende netamente del modem que genera la red WI-FI; una desventaja de esta conexión es que si la red no es estable, el control de la plataforma robótica obtendrá retardos en los comandos de acción señalados por el usuario. En la segunda se crea una conexión punto a punto, en este tipo de conexión no se necesita de internet como en la anterior, ofrece una ventaja y es el control casi inmediato(casi porque hay un retardo en el proceso de datos en python) de la plataforma robótica, además se puede controlar en cualquier área sin importar si hay conexión a internet o no, una desventaja es el poco alcance que el módulo wireless ofrece para este tipo de enlace, ya que el alcance depende solamente de dicho módulo, al momento de salir del rango, la plataforma robótica pierde el control realizado por el usuario.

4.1 CAPA INTERNA DE LA PLATAFORMA ROBÓTICA

La plataforma robótica se compone de varios módulos para el funcionamiento, donde se encuentran los sensores de ultrasonido, GPS, puente H, motores, *Shield* para el Arduino Mega, Arduino Mega. Esta tarjeta a parte de integrar los módulos mencionados anteriormente, cuenta con una parte de potencia que es la encargada de alimentar los motores y proporcionarles la corriente y el voltaje necesario para el correcto funcionamiento. Adicionalmente se dejaron los pines libres para seguir con conexiones extras al Arduino Mega. La descripción de la *Shield* se puede observar en la Figura19.

Figura19. Características Shield Arduino Mega



4.1.1 Motores y puente H

Para el control de los motores se utilizó dos tarjetas, cada una contiene un puente h dual L298N. Este módulo puente H cuenta con cuatro pines de dirección, 2 para cada motor. Dos habilitadores, uno para cada motor. En la Tabla4 se puede observar como es el funcionamiento de los pines digitales de dirección.

Tabla4. Pines de dirección del motor

Int A	Int B	Resultado
0	0	Parar
0	1	Giro (en un sentido)
1	0	Giro (en el otro sentido)
1	1	Parar

Cuando Int A e Int B son “0-0”, el motor queda totalmente quieto, no se mueve en ninguna dirección, cuando Int A e Int B son “0-1”, el motor gira para un sentido, si

es “1-0”, gira para el sentido contrario donde giró anteriormente, si es “1-1“, el motor queda quieto totalmente.

Para controlar la velocidad de los motores, se hace el uso de los habilitadores del puente H dándoles un pulso PWM, si el pulso es mínimo el motor se detiene y si es máximo el motor alcanza su velocidad máxima. Los motores utilizados para la plataforma robótica se pueden observar en la Figura20.

Figura20. Motorreductor 100 RPM[53]

Este es un motorreductor de alto torque con piñoneria metálica, tiene acoplado un encoder de cuadratura de efecto hall que provee una resolución de 64 pulsos por vuelta. Ya que el enconder se encuentra acoplado al eje del motor, en la salida del motorreductor se obtiene 6533 pulsos por vuelta dando una excelente precisión. El encoder cuenta con 6 cables, rojo la potencia del motor, el negro la tierra de la potencia del motor, el verde hace referencia a la tierra del sensor, el azul es el positivo del sensor, el amarillo y el blanco son salidas del encoder nombradas A y B respectivamente. En la Tabla5 se observan las características del motor.

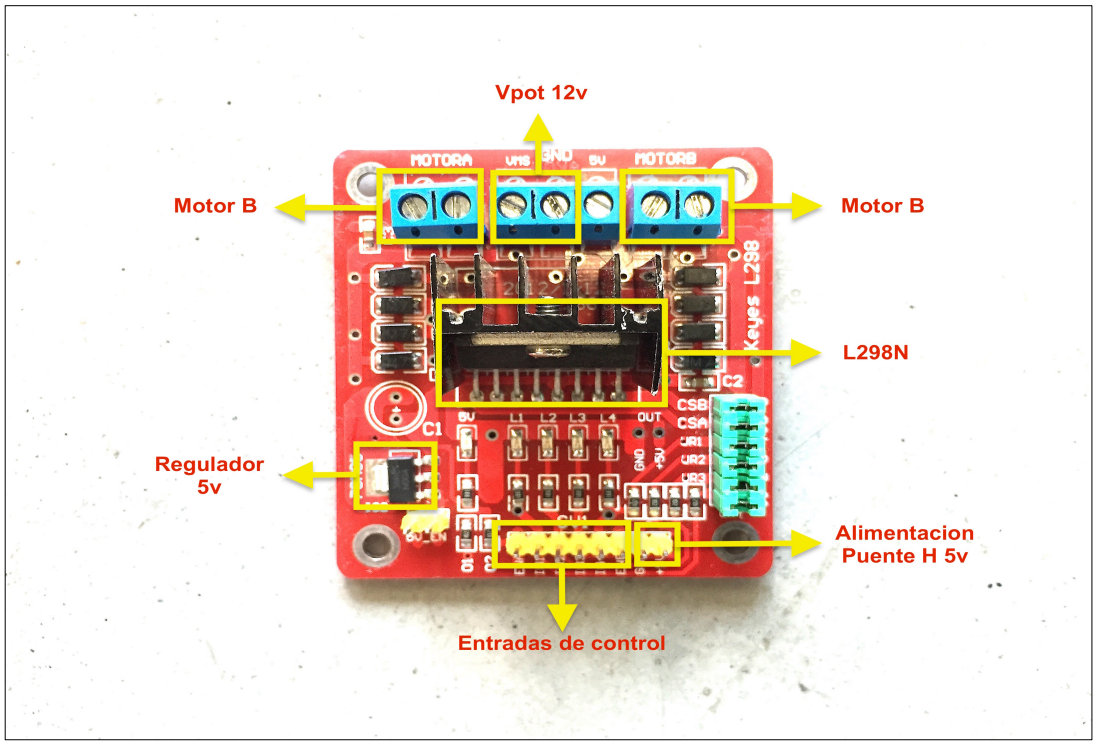
[53] Tomado de: <http://www.dynamoelectronics.com/index.php?page=shop.product_details&flypage=dynamo.tpl&product_id=994&category_id=100&option=com_virtuemart&Itemid=58>

Tabla5. Características del motor[54]

Voltaje	12v
Revoluciones	100 RPM
Torque Stall	16 Kg-cm
Corriente Stall	5 A
Corriente free-run	300 mA

En la Figura21 se puede observar las partes del módulo puente H y en la Figura22 la conexión respectiva de los motores y el puente h. Se aclara que cada puente h puede manejar 2 motores.

Figura21. Partes del módulo puente h



[54] DYNAMO. Datos del motor. [online]. Disponible en: <http://www.dynamoelectronics.com/index.php?page=shop.product_details&flypage=dynamo.tpl&product_id=994&category_id=100&option=com_virtuemart&Itemid=58>

El puente h cuenta con 6 entradas, 4 de ellas encargadas del giro del motor, y otras 2 utilizadas de habilitadores. Las 4 entradas se denominan IN1, IN2, IN3, IN4, dónde IN1 e IN2 controlan el giro del motor A, y el giro del motor B la controla IN3 e IN4. Los otros 2 pines de entrada se denominan ENA y ENB, una para habilitar el motor A y la otra para el motor B respectivamente. Posteriormente cuenta con 2 pines para energizar el puente h a 5v denominados +5v y GND.

Adicionalmente el puente h cuenta con 4 salidas que son las encargadas de alimentar los motores, denominadas motor A y motor B. Además los motores se pueden alimentar por aparte de la alimentación del puente h, dependiendo del voltaje que necesiten los motores se alimentan estos pines. En la Tabla6 se observan las características del módulo puente h.

Tabla6. Características del módulo puente h[55]

Driver	Puente h dual L298N
VMS	5 a 35v
Corriente puente h	2 A
Voltaje de entrada	4.5 a 7v
Corriente de operación	0 a 36 mA

Esta conexión de la Figura22 es la ideal para el correcto funcionamiento de los motores, para mayor comodidad en el diseño de la plataforma robótica se implementó el montaje de la Figura23 la cual, los 4 motores se conectan dentro de la *Shield* Arduino Mega.

<http://www.dx.com/p/l298n-stepper-motor-driver-controller-board-for-arduino-120542#.VddrJrTldEQ(22%20de%20agosto)>

Figura22. Conexión de puente h y motores

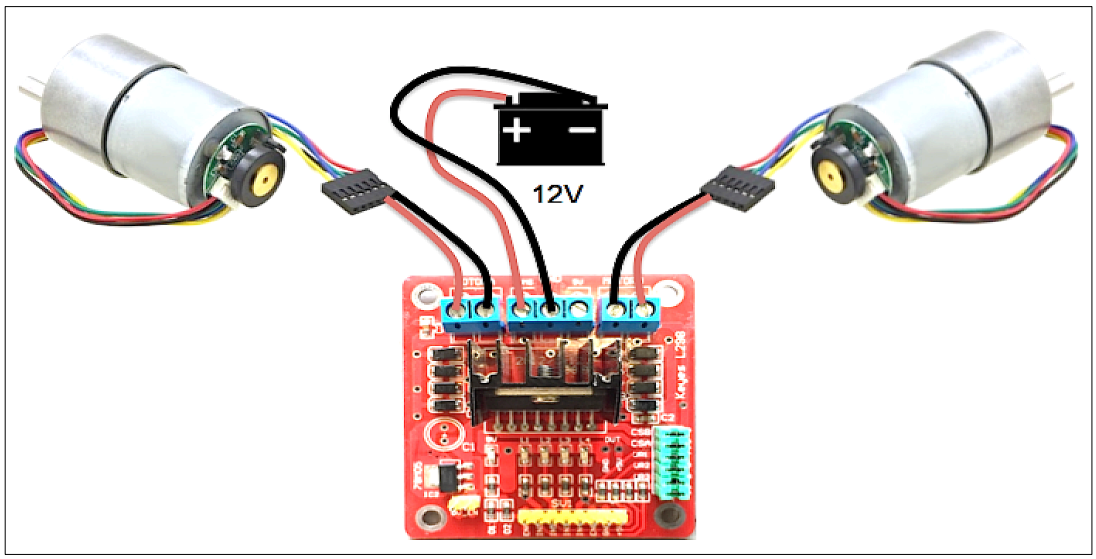
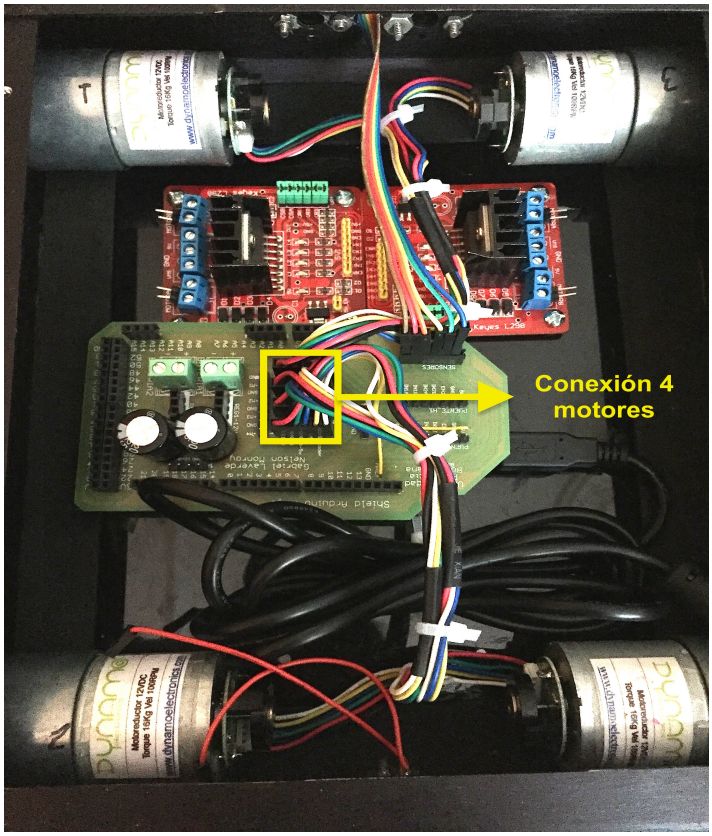


Figura23. Conexión de motores en la plataforma robótica



Esta tarjeta es alimentada por dos baterías de lipo 7.4v a 2200 mAh, las dos baterías se encuentran conectadas en serie obteniendo un total de 14.8v, los cuáles serán suministrados a los reguladores de 12v para obtener una tensión regulada y alimentar los puente h en la zona te potencia de motores (VMS). Esas conexiones nombradas anteriormente ya se encuentran impresas en la Shield de Arduino Mega.

De esta manera se pueden controlar los motores de la plataforma robótica, aunque los 4 motores sean de 100rpm, esto no garantiza que a un PWM los 4 motores lleguen a la misma velocidad por lo cual el desplazamiento no será netamente en línea recta. Para solucionar este problema se hace necesario un control PID para los motores.

4.1.2 DISEÑO E IMPLEMENTACIÓN DEL CONTROLADOR PID DE VELOCIDAD DEL MÓVIL.

4.1.2.1 Datos obtenidos sin peso en el sistema robótico (sin fricción en las llantas)

La velocidad de los motores se ve afectada por el *dutty cicle* de una señal cuadrada de 490 Hz proveniente de los pines 7,8,9 y 10 del Arduino Mega; estas señales PWM son conectadas a los pines habilitadores de cada canal de los puente h, debido a que al variar los tiempos de activación de los motores se puede variar la velocidad de estos.

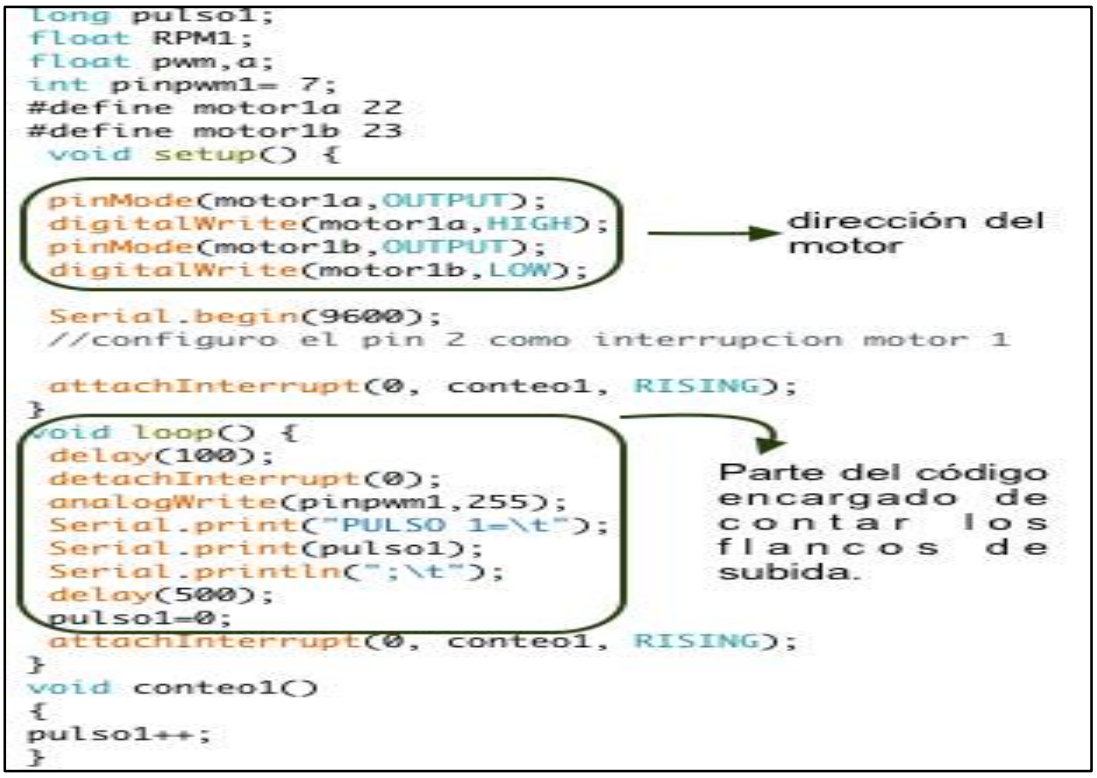
Cada motor tiene un encoder de caudratura de efecto hall, el cuál proporciona dos señales cuadradas, una de ellas indica la velocidad del motor al variar su ancho de pulso, la otra señal varia su ancho de pulso a la misma velocidad que la primera señal, pero esta se encuentra desfasada +/- 90 grados dependiendo la orientación del giro del motor; debido a limitaciones de hardware se usará solamente la señal que indica la velocidad actual del motor, por tal motivo se requiere la implementación de 4 interrupciones externas.

El Arduino Mega posee 6 interrupciones externas de las cuáles se usarán las ubicadas en los pines 2,3,21 y 20, para detectar los flancos de subida de la señal cuadrada de los encoder.

Para calcular las RPM de los motores se halló un factor que multiplica la cantidad de pulsos contados en 100 milésimas de segundo para hallar la velocidad del motor; para encontrar este número se alimenta el puente h que a la vez alimenta el motor con una fuente regulada y con la ayuda de un tacómetro se aumenta la tensión hasta conseguir 50 RPM medidas por el tacómetro.

Luego de obtener esta tensión, se procede a programar el Arduino Mega con el siguiente *sketch* que se observa en la Figura24 para ver los pulsos detectados por la interrupción externa, es importante aclarar que el *dutty cicle* es del 100 % para este ejemplo, ya que la velocidad del motor se varia a partir de la tensión de la fuente regulada.

Figura24. sketch para contar los pulsos de subida



Por medio de la interrupción externa reconoce los flancos de subida de la onda cuadrada, esta interrupción redirige el código a la función conteo1 la cual es la encargada de contar los flancos detectados durante 100 ms, luego de eso se desactiva la interrupción, se imprime el resultado y se vuelve activar.

El resultado de la ejecución de este *sketch* se puede observar en el monitor serial del *Idle* de Arduino, allí se puede visualizar que la variable pulso detecta 136 flancos de subida en 100 ms. Al realizar la división entre 50 que es la RPM observada en el tacómetro y 136 que es el resultado, se observa que el factor que se está buscando es 0,367.

Con este dato se procede en la escritura del *sketch,* que envíe un PWM a los cuatro motores y a la vez debe detectar los flancos de subida proveniente de los

cuatro encoders, para calcular y visualizar en el monitor serial las RPM de los cuatro motores al mismo tiempo; para comprobar el correcto funcionamiento de este código que se observa en la ,Figura25 se utiliza como segundo instrumento de medición un tacómetro, además a partir de este momento la tensión de los motores siempre serán 12v regulados.

Figura25. Sketch para calcular las 4 RPM de la plataforma robótica

Las configuraciones que se realizaron para que este código funcionara son las observadas en la Figura26.

Figura26. Configuración del sketch para obtener las RPM.


```
void setup() {
  // put your setup code here, to run once:
  pinMode(motor1a, OUTPUT);
  digitalWrite(motor1a, HIGH);
  pinMode(motor1b, OUTPUT);
  digitalWrite(motor1b, LOW);
  pinMode(motor2b, OUTPUT);
  digitalWrite(motor2b, HIGH);
  pinMode(motor2a, OUTPUT);
  digitalWrite(motor2a, LOW);
  pinMode(motor3a, OUTPUT);
  digitalWrite(motor3a, HIGH);
  pinMode(motor3b, OUTPUT);
  digitalWrite(motor3b, LOW);
  pinMode(motor4a, OUTPUT);
  digitalWrite(motor4a, LOW);
  pinMode(motor4b, OUTPUT);
  digitalWrite(motor4b, HIGH);
  analogWrite(pinpwm1, 255);
  analogWrite(pinpwm2, 255);
  analogWrite(pinpwm3, 255);
  analogWrite(pinpwm4, 255);
  //pinMode(2, INPUT);
  Serial.begin(9600);
  //configuro el pin 2 como interrupcion motor 1
  attachInterrupt(0, conteo1, RISING);
  //configuro el pin 3 como interrupcion motor 2
  attachInterrupt(1, conteo2, RISING);
  //configuro el pin 21 como interrupcion motor 3
  attachInterrupt(2, conteo3, RISING);
  //configuro el pin 20 como interrupcion motor 4
  attachInterrupt(3, conteo4, RISING);
}
```

Al realizar esta prueba se puede observar que aunque todos los motores tengan el mismo *dutty cicle y la* tensión sea igual para cada uno, no tienen una RPM en común por lo tanto es necesario realizar un control PID para asegurar que los motores tengan la misma velocidad en régimen permanente. Para realizar esto es necesario hallar la planta de cada motor, luego observar la respuesta de cada modelo en lazo abierto, sintonizar las constantes de PID, observar la respuesta en lazo cerrado de cada motor, observar la respuesta en lazo cerrado con *stateflow* y por ultimo pasar el PID al Arduino.

Para hallar el modelo de cada motor es necesarito tener una tira de datos de la velocidad del motor correspondientes a un cambio del *dutty cicle* definido, para obtener estos datos se diseña un *sketch* capaz de realizar este cambio y tomar los datos de los RPM del motor, esta programación se puede observar en la Figura27.

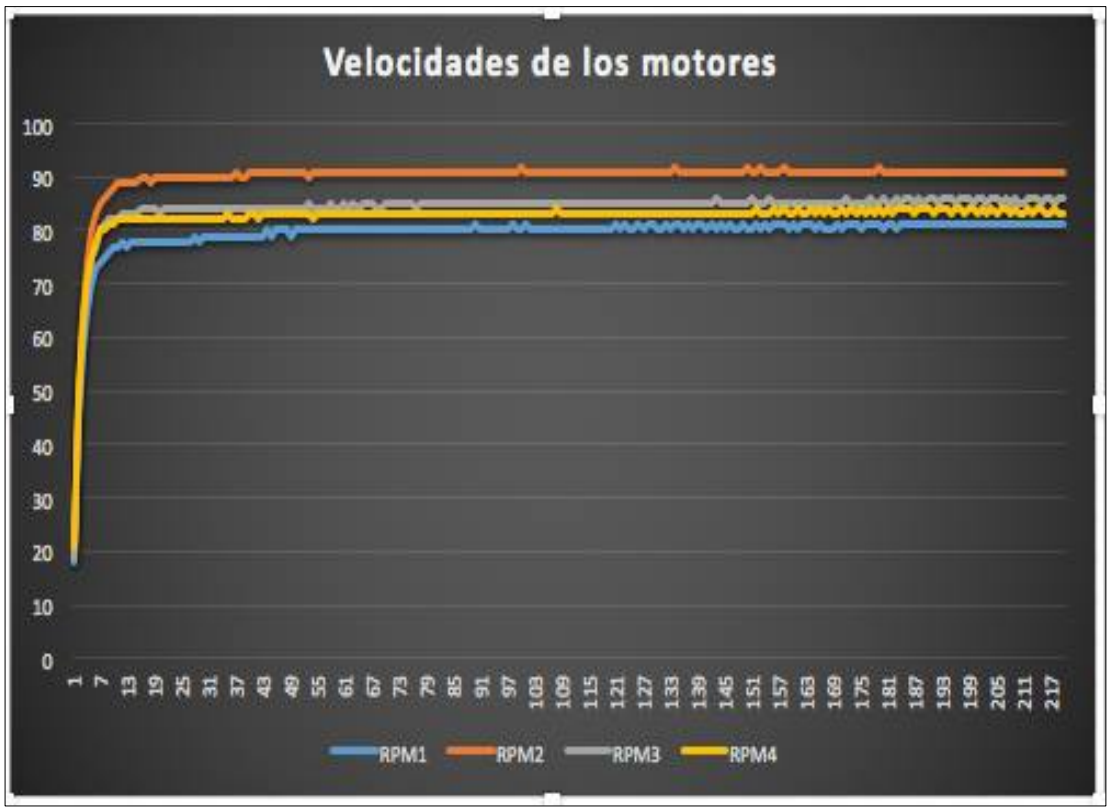
Figura27. Captura de datos para la sintonización.

```
void loop()
{
   delay(100);
   detachInterrupt(0);
   DATO[i]=0;
   RPM1=pulso1*(0.367);
   DATO[i]=RPM1;
   i++;
   a=i;
   pulso1=0;
 if (i==220)
 {

   Serial.println("comienzo de la transmision de datos");
   Serial.print("ITERACIONES=\t");
   Serial.println(i);
   Serial.print("dato maximo=\t");
   Serial.println(pwm1);
   analogWrite(pinpwm1,0);
   for (int i=0;i<a;i++)
   {
   Serial.println(DATO[i]);
   }
   Serial.println("fin de la transmision de datos");
  delay(100);
  i=0;
  a=0;
  attachInterrupt(0,conteo1,RISING);
  analogWrite(pinpwm1,166);
 }
  attachInterrupt(0,conteo1,RISING);
}
```

Al iniciar este *sketch se* registra 220 muestras, pero como aún no se a dado el cambio del *dutty cicle,* la velocidad que se obtiene es 0 RPM, una vez impreso los 220 datos se realiza el cambio y se activa la interrupción**,** la cual registrara cada 100 ms la velocidad actual del motor. Cuando el registro de muestra se acabe, se procederá a visualizar todos los datos en el monitor serial para ser copiados y llevados al *workspace* de Matlab. Para realizar este proceso con los demás motores únicamente es necesario realizar pequeñas modificaciones al código, por ejemplo el escalón se realizara al motor 2, para ello basta con cambiar pinpwm1 con pinpwm2.

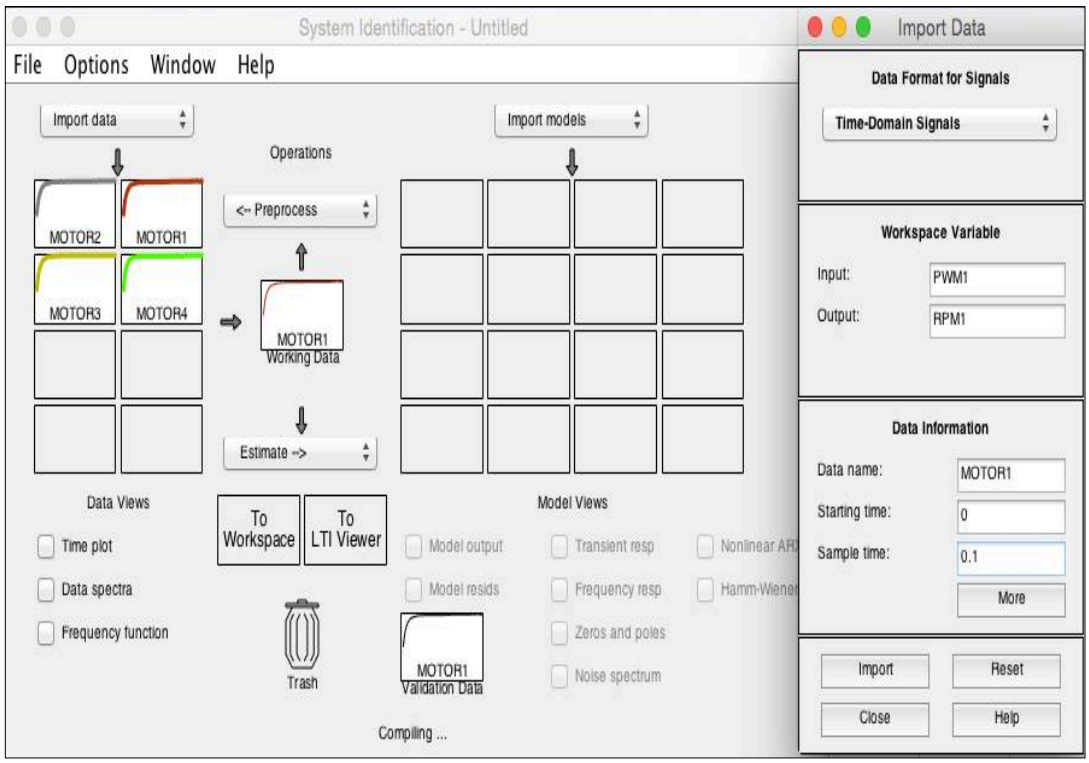
La respuesta de los cuatro motores se puede visualizar en la Figura28 allí se rectifica la necesidad de implementar un control PID para garantizar la misma velocidad en régimen permanente.

Figura28. Velocidad de los 4 motores con PWM fijo.



Utilizando la herramienta Ident de Matlab se ingresa estos datos para aproximar los modelos matemáticos de cada motor, para ello se crea un vector de 220 posiciones en donde todos sus elementos son de 166 en el campo de entrada, esto equivale al cambio que se realiza para la toma de datos. En el campo de salida se ingresa los datos almacenados de cada motor. Como se puede observar la Figura29.

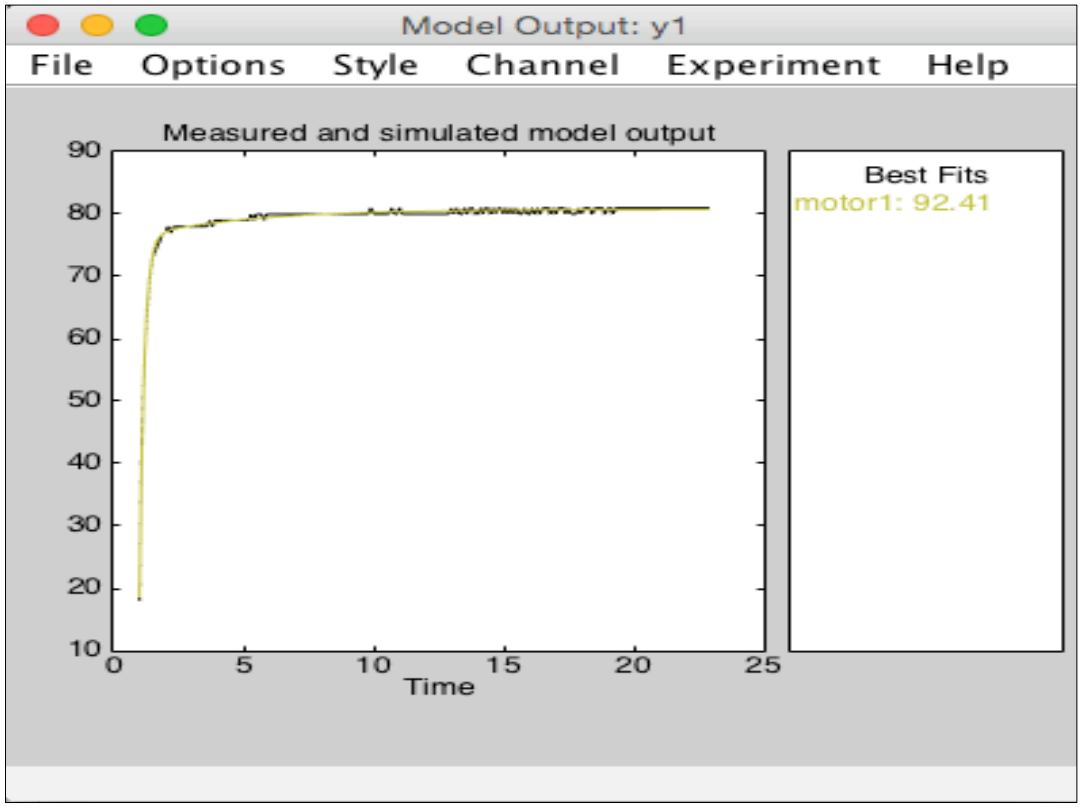
Figura29. Datos importados a Ident.



Luego estos datos son ingresados en Ident, se procede hallar los modelos aproximados de las plantas, las cuales son los cuatro motores. Para validar los modelos de proceso es necesario arrastrar los datos importados en *working data y en Validation data.*

Se examina los modelos de proceso arrojado por la herramienta Ident y se escogen aquellos que tienen una aproximación del mas de 90 % con respecto a la salida de los modelos como se puede observar en la Figura30.

Figura30. Comparación entre la respuesta real y la del modelo de proceso aproximado.



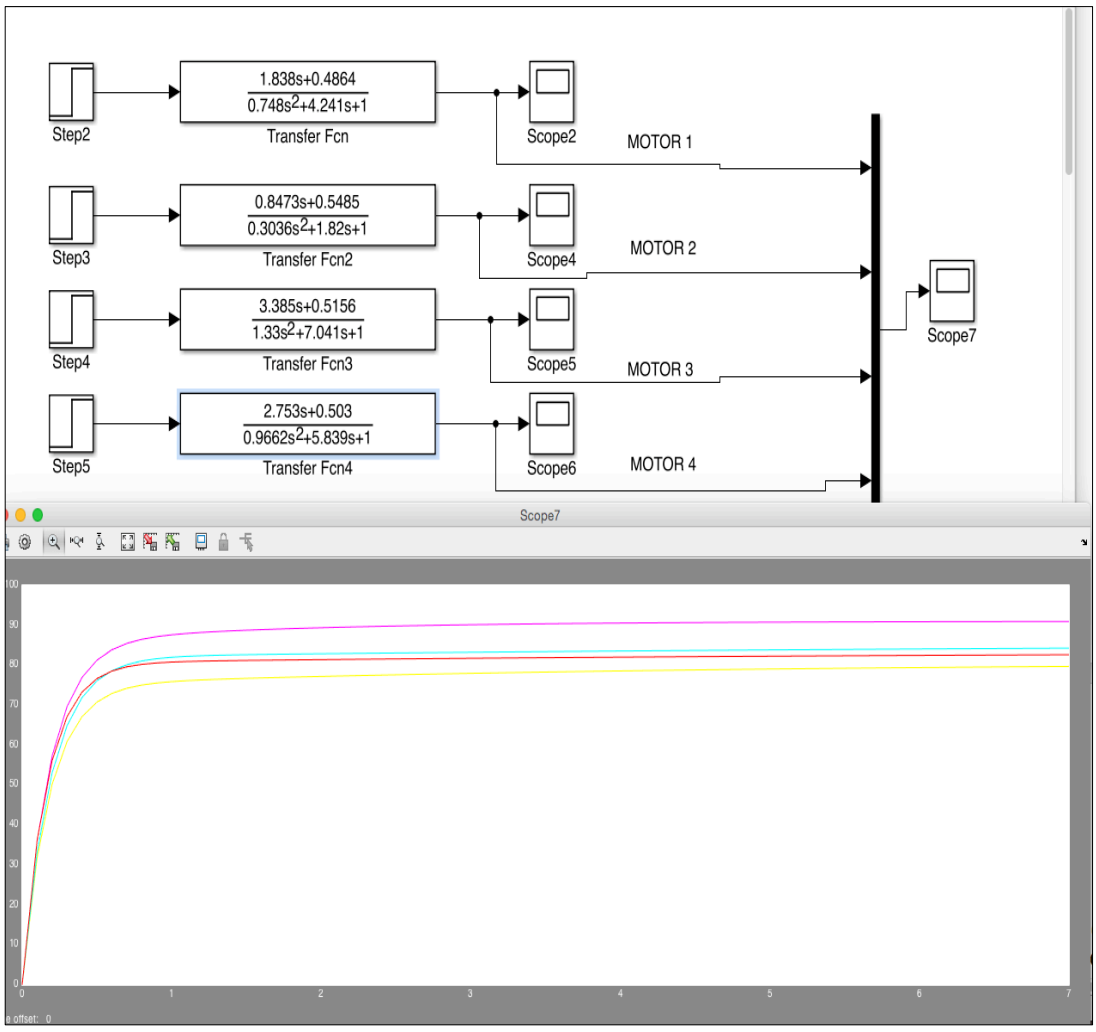
Una vez realizado el proceso anterior con los demás datos importados se puede exportar los modelos de procesos al *workspace* de Matlab en donde se ejecuta el comando "tf" sobre los elementos exportados para obtener la función de transferencia de cada motor. Estas funciones se encuentran en la Figura31 son requeridas para observar la respuesta en lazo abierto y así poder sintonizar el controlador PID.

Figura31. Funciones de transferencia de cada motor.

```
>> tf(motor1)

ans =

  From input "u1" to output "y1":
     1.838 s + 0.4864
  -----------------------
  0.748 s^2 + 4.241 s + 1

Name: motor1
Continuous-time transfer function.

>> tf(motor2)

ans =

  From input "u1" to output "y1":
     0.8473 s + 0.5485
  ----------------------
  0.3036 s^2 + 1.82 s + 1

Name: motor2
Continuous-time transfer function.

>> tf(motor3)

ans =

  From input "u1" to output "y1":
     3.385 s + 0.5156
  ---------------------
  1.33 s^2 + 7.041 s + 1

Name: motor3
Continuous-time transfer function.

>> tf(motor4)

ans =

  From input "u1" to output "y1":
      2.753 s + 0.503
  ----------------------
  0.9662 s^2 + 5.839 s + 1

Name: motor4
Continuous-time transfer function.
```

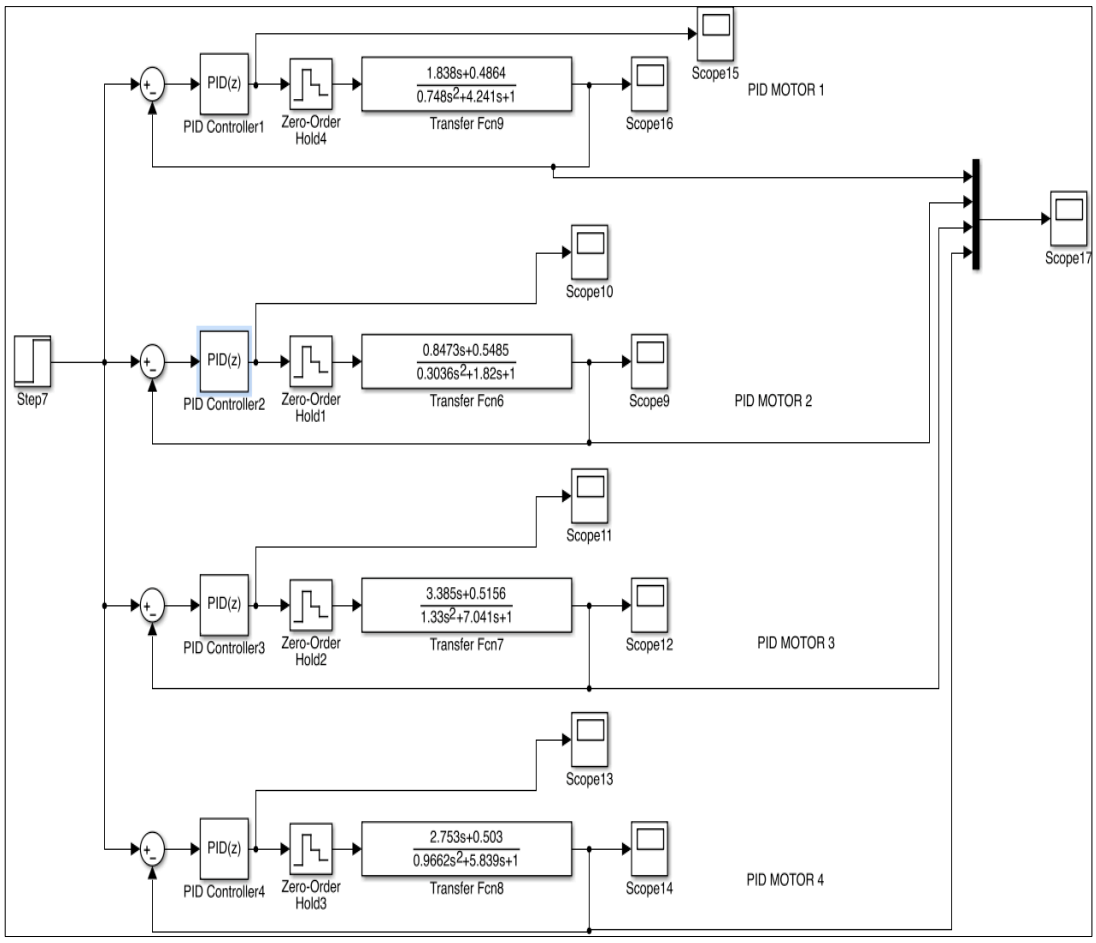
Para observar la respuesta en lazo abierto es necesario ingresar estas funciones de transferencia a *Simulink* por medio del bloque *Transfer Fcn*, el cual recibe como entrada un escalón producido por el bloque *step*; la salida de las funciones de transferencia son conectada a un *mux* y luego a un *scope* para visualizar su respuesta. En la Figura32 se observa el archivo de Simulink, donde se encuentra las funciones de transferencia en lazo abierto y el resultado es visualizado en el *scope*.

Figura32. Funciones de transferencia en lazo abierto.



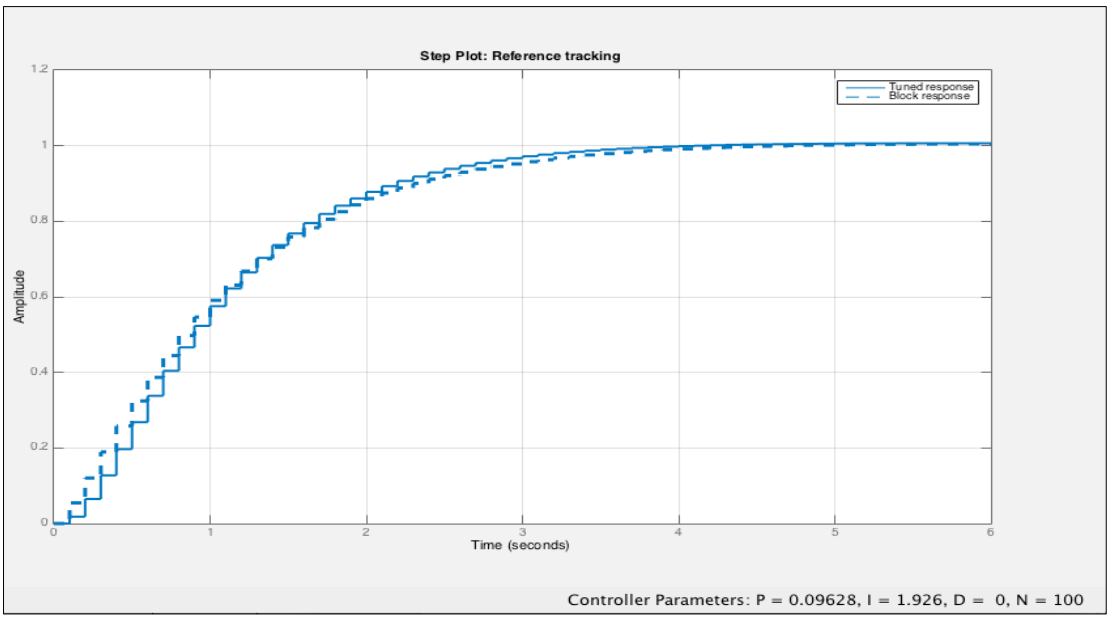
Para sintonizar el controlador PID es necesario cerrar el lazo con alimentación negativa y ubicar el bloque de control entre la señal de error y la planta del motor. En la Figura33 se observa todas las cuatro plantas de los motores con su respectivo bloque PID y adicionalmente se insertaron dos *scope* para observar las respuestas del controlador y la planta de cada motor.

Figura33. Realimentación negativa de cada motor.



En la sintonización de cada motor se busca que la respuesta final de la planta no tenga *overshoot* y sea lo más rápida posible; en la Figura34 se puede observar la sintonización del motor uno, la cual fue realizada por el bloque PID al ejecutar la opción *tune*; se puede observar que esta sintonización reduce significativamente el *overshoot.*

Figura34. Sintonización del motor 1.



Este proceso se vuelve a realizar para los motores restantes, en la Figura35 se observa la respuesta del motor 2, la cual tiene un gran parecido con la respuesta del motor anterior, lo mismo se puede observar en la Figura36 que corresponde a la sintonización del motor 3.

Figura35. Sintonización del motor 2.

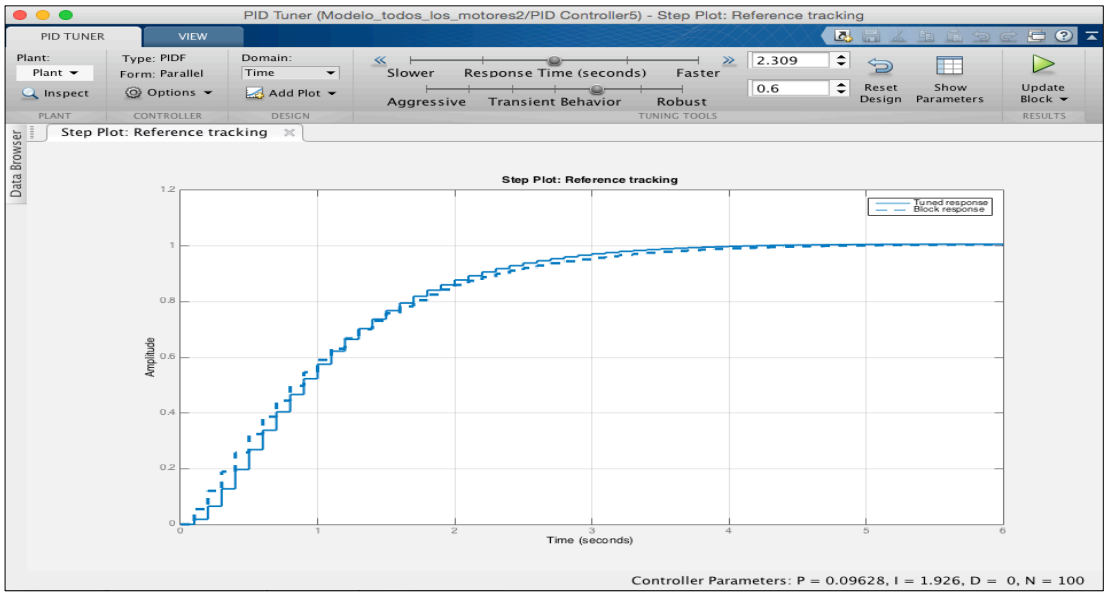
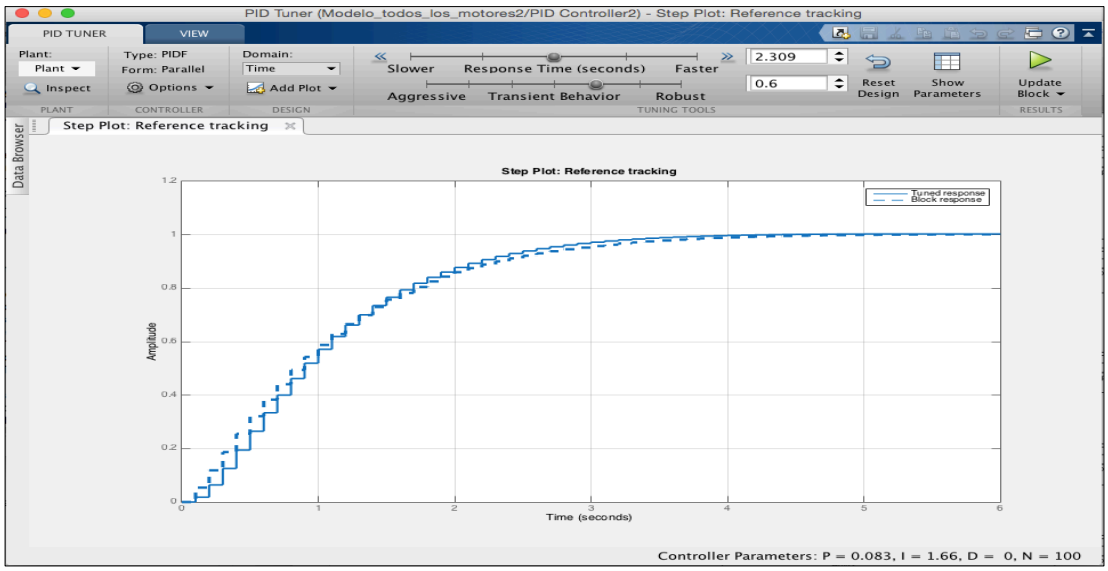
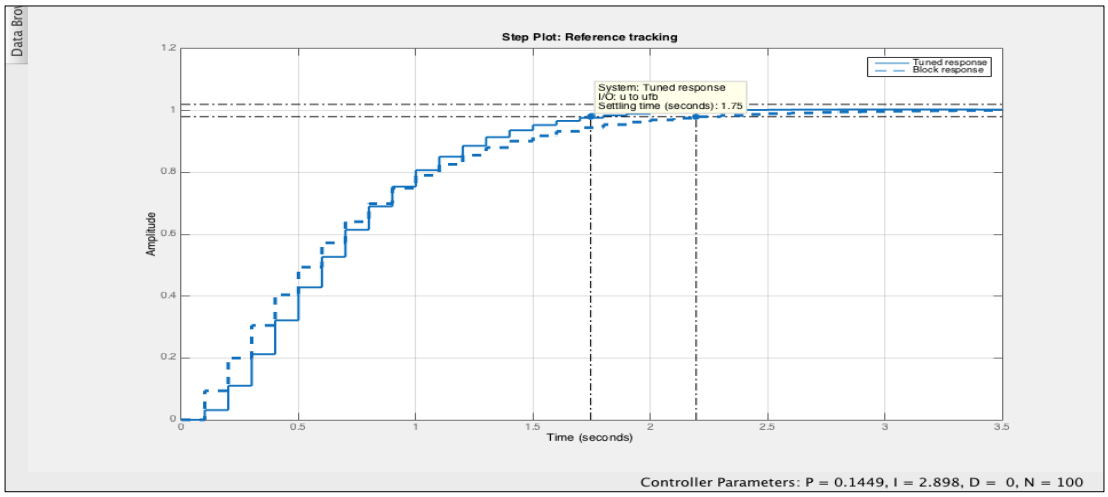


Figura36. Sintonización del motor 3.



La sintonización del motor 4 es diferente debido a que el tiempo de establecimiento de este es menor que en los motores anteriores, en la Figura37 se puede observar por medio de la opción *settling time* la sintonización que tiene esta característica; en las anteriores figuras se encuentra que el tiempo de establecimiento es mayor a 3 segundos.

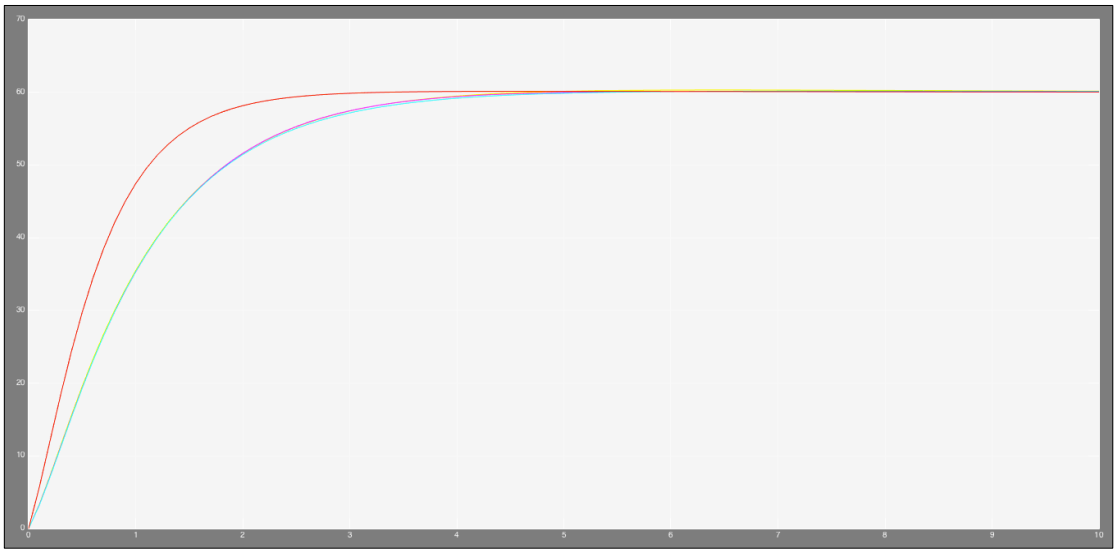
Figura37. Sintonización del motor 4.



Si se diseña el control de velocidad con estas constantes, en el Arduino se tendrá un resultado similar al que se muestra en la Figura38. Además al realizar la

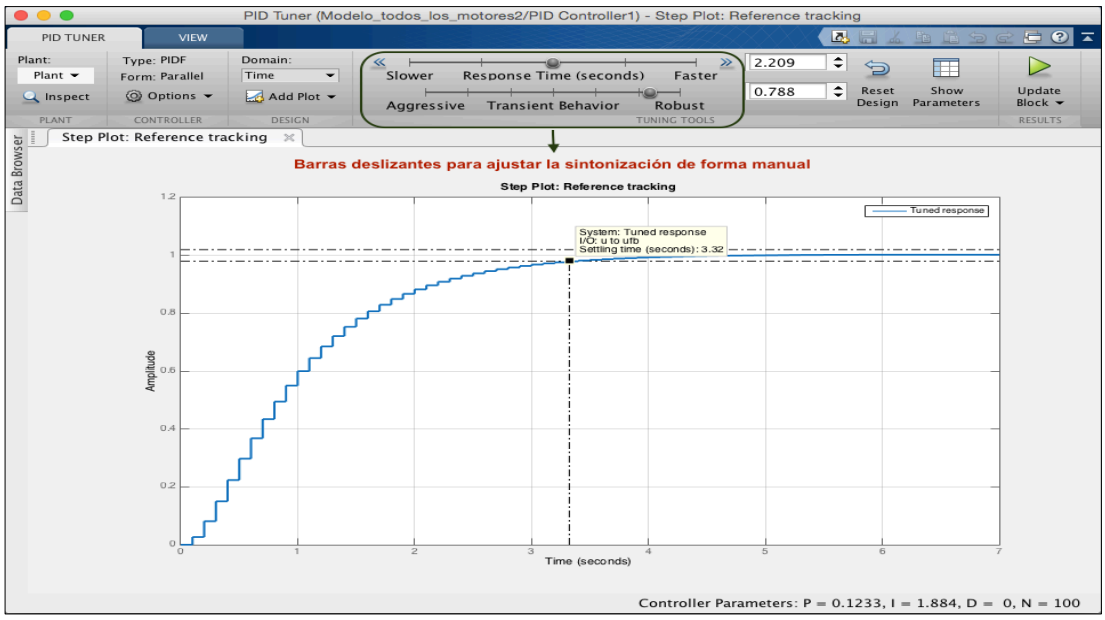
programación en el Arduino de los controladores de los motores; no se estaría garantizando igualdad de comportamiento en el régimen permanente ni en el transitorio, debido a que el sistema en general no se estabiliza al mismo tiempo, por lo tanto se modifica la sintonización del motor 4 para que este alcance el régimen permanente al mismo tiempo que en los demás motores y tenga alguna similitud al régimen transitorio.

Figura38. Velocidad de los motores con PID solo con el método de auto sintonización.



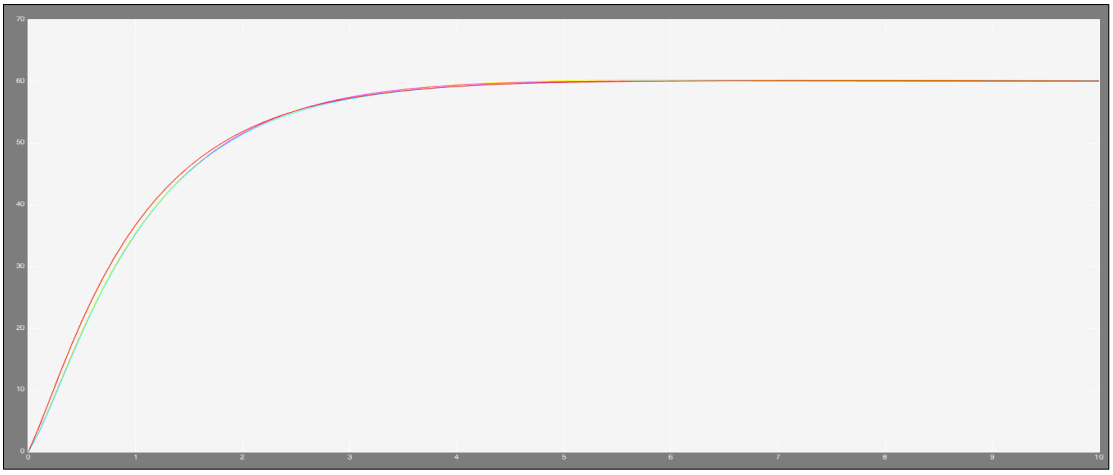
Para ajustar el funcionamiento del control que se realiza en el motor 4 es necesario auto sintonizarlo de nuevo y utilizar las dos barras deslizantes de la parte de arriba de la Figura39 para igualar el tiempo de establecimiento de esta planta sin provocarle algún *overshoot.*

Figura39. Sintonización del motor 4 ajustada por el usuario.



La respuesta de los 4 motores con sus respectivos controladores; algunos ajustados manualmente y otros auto sintonizados, se pueden visualizar en la Figura40, aquí el comportamiento general de la plataforma robótica en movimiento se tiene una mejoría; se puede observar que en el régimen transitorio la diferencia entre velocidades no es demasiada alta y además los 4 motores llegan al mismo tiempo de régimen permanente.

Figura40. Velocidad de los motores con control PID sintonizados de manera manual y automáticamente.



Para asegurar la robustez del controlador de la plataforma robótica se realizó una nueva toma de datos del robot moviéndose en un terreno en donde no se presente irregularidades y que tenga la más mínima inclinación. Además todos los dispositivos como sensores, baterías, tarjetas y demás elementos de hardware deben estar ubicados en su localización final; esto se realiza para eliminar los efectos de peso ejercido por todo estos elementos y facilitar las respuestas de los controladores.

Para realizar esta nueva toma de datos se hacen modificaciones leves al *sketch* del Arduino, esto es necesario para que registre las RPM de los 4 motores; en esta nueva toma de datos se realizó en el suelo en donde el movimiento de estos son activados bajo la misma señal de PWM proveniente del Arduino, otra consideración que se debe tomar es la carga de las baterías, ya que este parámetro afecta a gran medida la velocidad que genera los motores.

En la Figura41 se observa los nuevos datos importados a la herramienta *Ident* con la finalidad de hallar los modelos de procesos de los 4 motores, en la Figura42 se visualiza la gráfica de los datos importados en *Ident*, para habilitar, sólo es necesario seleccionar el cuadro de *time plot* en la herramienta *Ident*. La forma que tienen estos datos importados es debido a las nuevas perturbaciones generadas por el peso del dispositivo robótico que se encuentran en el sistema.

Figura41. .Herramienta *Ident* con la importación de los nuevos datos de velocidad.

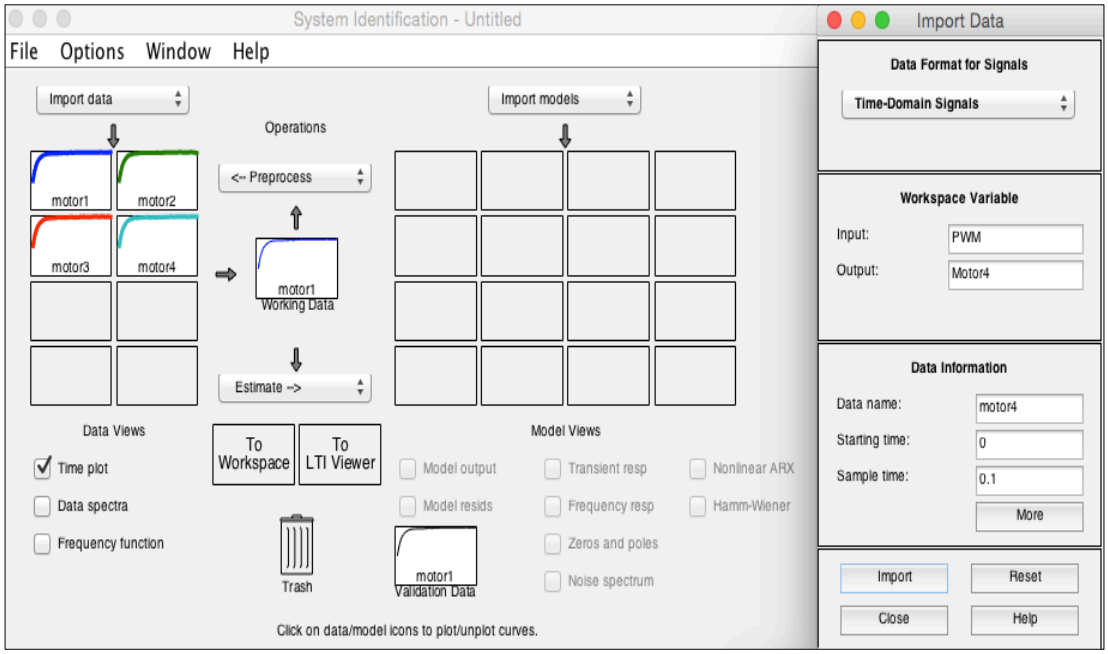
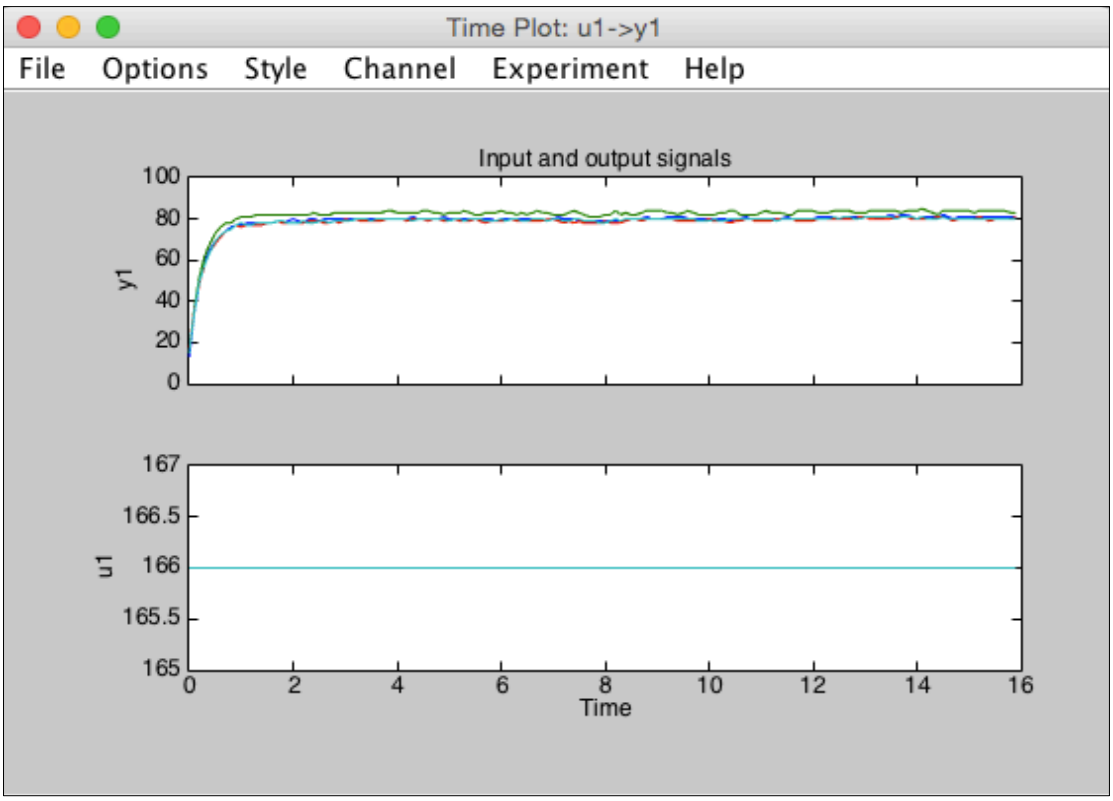


Figura42. Gráfica de las velocidades de los motores generada por *Ident*.



4.1.2.2 Datos obtenidos con peso completo del sistema robótico (con fricción en las llantas)

Al hallar las aproximaciones de los modelos de proceso de cada planta, se observa una reducción en el porcentaje de precisión, esto es debido a que los datos registrados presentan irregularidades debido a las perturbaciones presentes. En la Figura43 se observa las aproximaciones de los modelos de proceso de los cuatro motores, en esta nueva sintonización se eligieron modelos de grado 1 y sin ceros debido a la rápida respuesta de los motores. En la Figura44 se observa los modelos de proceso utilizando la función tf en el *workspace de Matlab.*

Figura43. Modelos de proceso de grado 1 de los 4 motores.

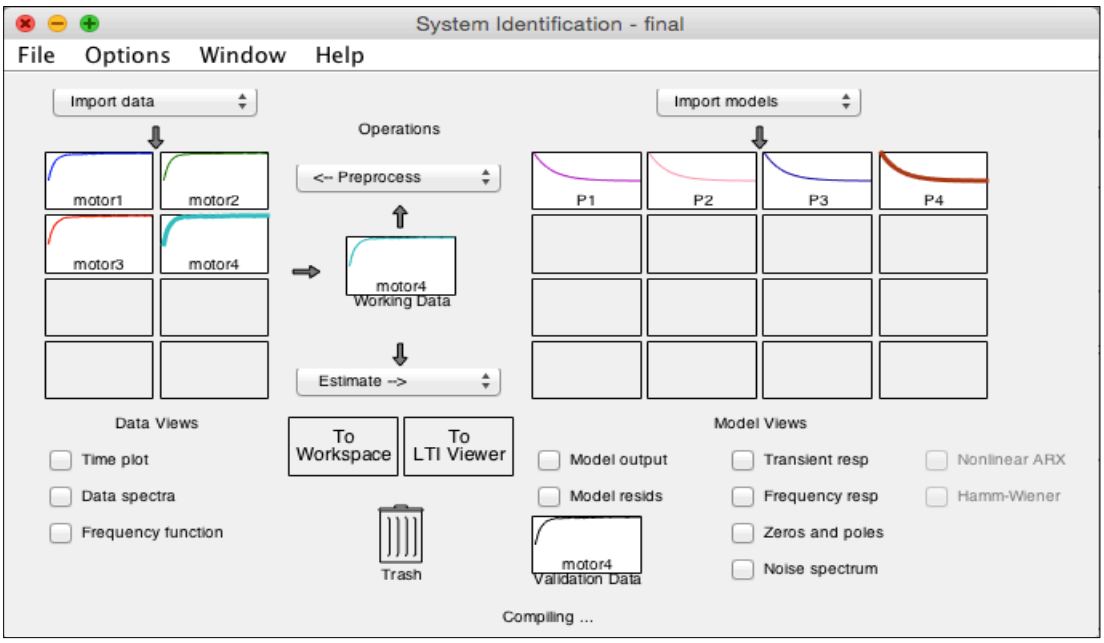
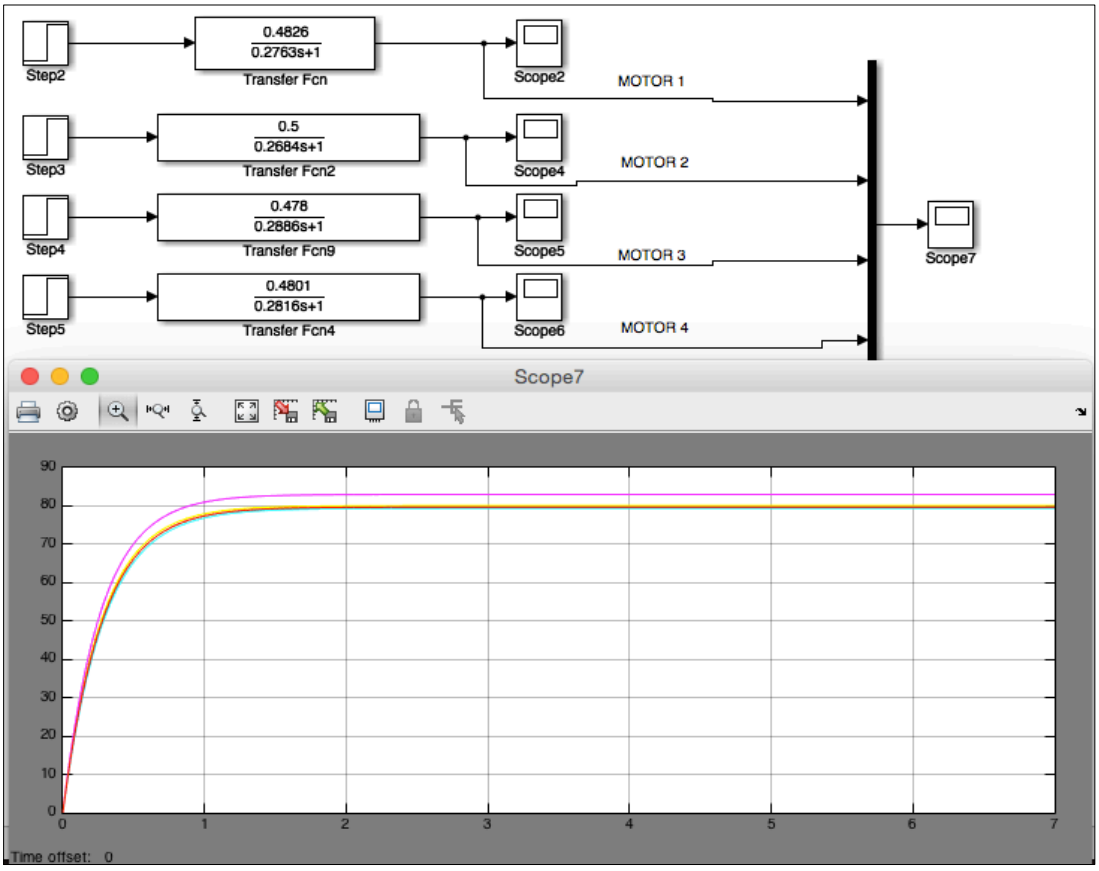


Figura44. Funciones de transferencia de los 4 motores.

```
    From input "u1" to output "y1":
        0.4826
    ---------------
    0.2763 s + 1

Name: P1
Continuous-time transfer function.

>> tf(P2)

ans =

    From input "u1" to output "y1":
         0.5
    ---------------
    0.2684 s + 1

Name: P2
Continuous-time transfer function.

>> tf(P3)

ans =

    From input "u1" to output "y1":
        0.478
    ---------------
    0.2886 s + 1

Name: P3
Continuous-time transfer function.

>> tf(P4)

ans =

    From input "u1" to output "y1":
        0.4801
    ---------------
    0.2816 s + 1
```

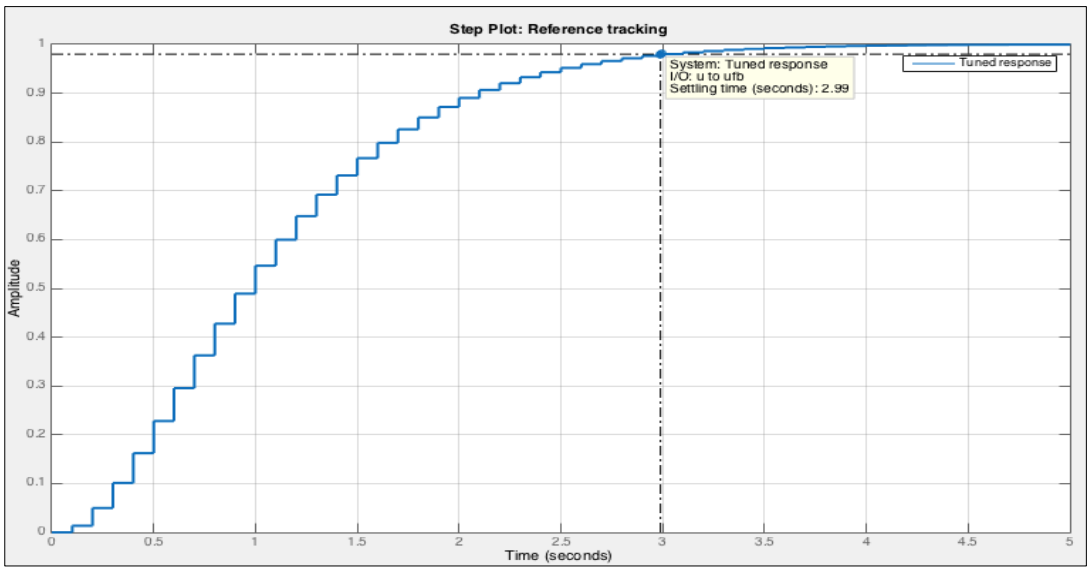
En simulink se agregrega estas nuevas funciones de transferencia usando el bloque Transfer Fcn para observar la respuesta en lazo abierto de estas plantas obtenidas por la función Ident. En la Figura45 se observa la salida de todos los motores, además se puede observar que obtuvo una respuesta mas rápida comparada con la anterior respuesta en lazo abierto.

Figura45. Respuesta en lazo abierto de los cuatro motores con sistemas de orden 1.



Para encontrar estas nuevas constantes es necesario cerrar el lazo de control, para ello se utiliza la función tune del bloque PID. En la Figura46 se observa el tiempo de establecimiento del motor 1, a partir de este tiempo se ajustara los controladores de los demás motores para que respondan de manera similar tanto en el régimen transitorio como en el régimen permanente. Además se debe observar el esfuerzo del controlador, el cual nos indica la variación de la salida del control por cada unidad de diferencia en la entrada.

Figura46. Tiempo de establecimiento del motor 1.



Como se puede observar el tiempo es de 2.99 segundos, por lo tanto el controlador de los demás motores los debemos hacer mas rápido o mas lentos según sea el caso de cada motor. El esfuerzo del controlador del motor 1 se observa en la Figura47. Aquí se datalla que por cada RPM de diferencia entre la lectura actual y el setpoint; aumenta o disminuye 2 unidades en el PWM, esto es importante para observar si el controlador se satura al realizar sus acciones.

Figura47. Esfuerzo del controldor del motor 1.

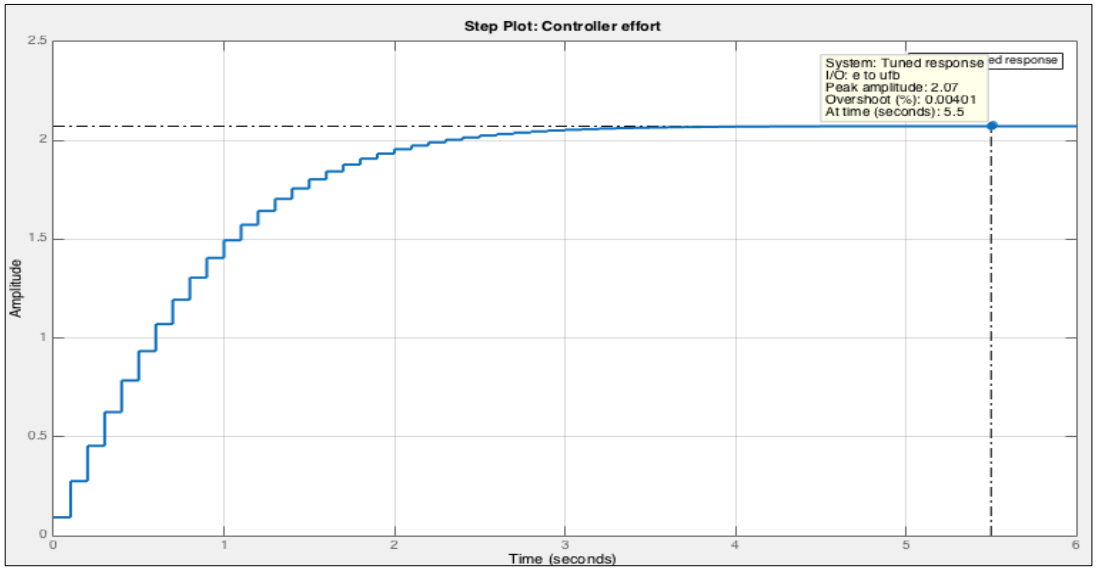


Figura48 se observa la acción del controlador y el esfuerzo del mismo para el motor 2 en laFigura49, aquí se ajustó el tiempo de establecimiento y el esfuerzo del controlador son dos unidades exactas.

Figura48. Tiempo de establecimiento en el motor 2.

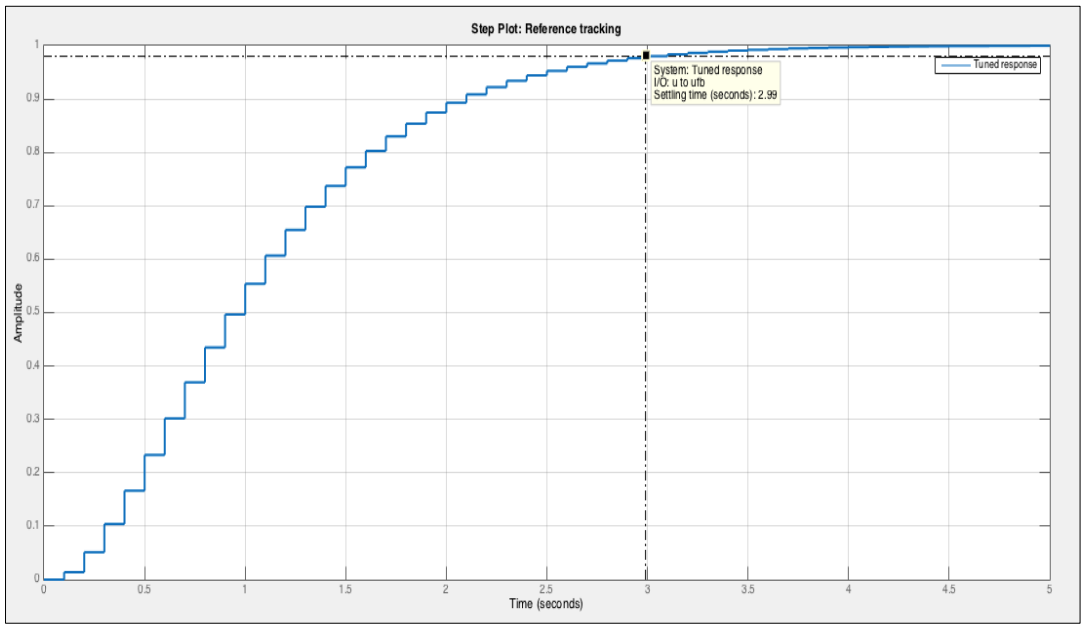
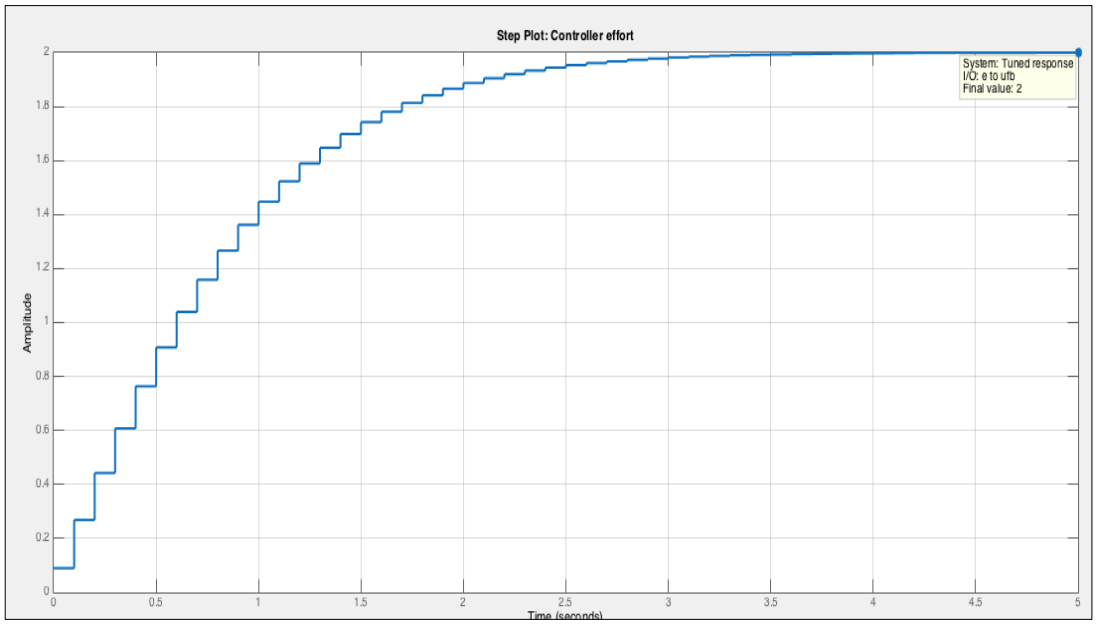
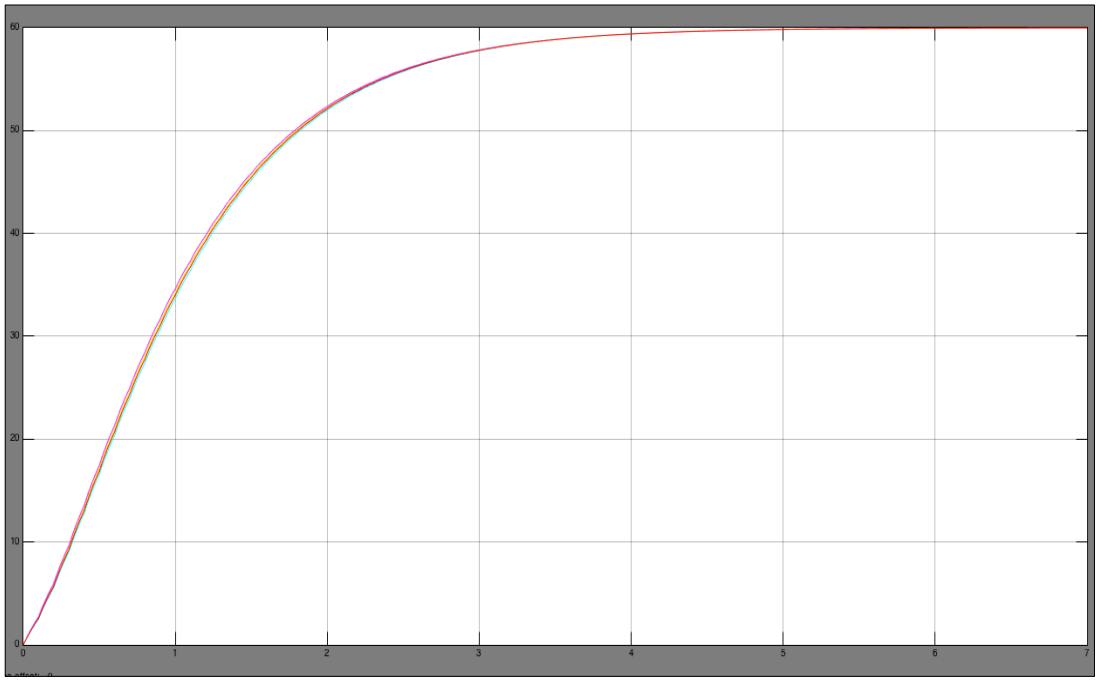


Figura49. Esfuerzo del controlador del motor 2.



Para los motores 3 y 4 se realizo el mismo procedimiento que en las dos primeras plantas, se observo que el esfuerzo del controlador se encuentra aproximadamente en dos unidades para estos motores; lo que nos indica que estos controladores son los apropiados para ejecutarlo en el ARDUINO, en la Figura50 se observa el comportamiento de los cuatro motores con su respectivo control.

Figura50. Comportamiento de los cuatro motores con control.



Para asegurar su correcto funcionamiento es recomendable ejecutar el control PID en *stateflow*, el cual es una herramienta en donde uno escribe el controlador en lenguaje C; es importante que dentro de este lenguaje se declaren las constantes obtenidas en la sintonización.

En la Figura51 se observa que el bloque PID fue sustituido por la herramienta *stateflow* en donde se programo tanto sus entradas como salidas, un anti *windup* el cual permitirá reestablecer el funcionamiento del control después de presentarse una saturación. La Figura52 se observa el comportamiento que se tendrá al implementar estos controladores.

Figura51. control PID realizado con *stateflow*.

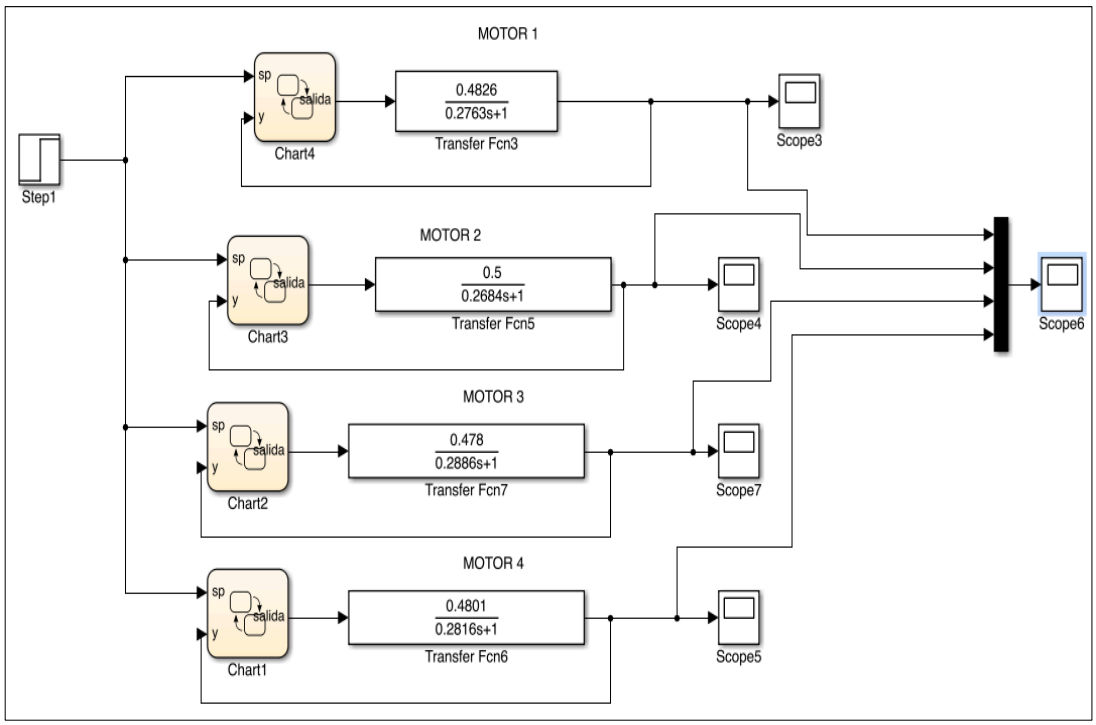
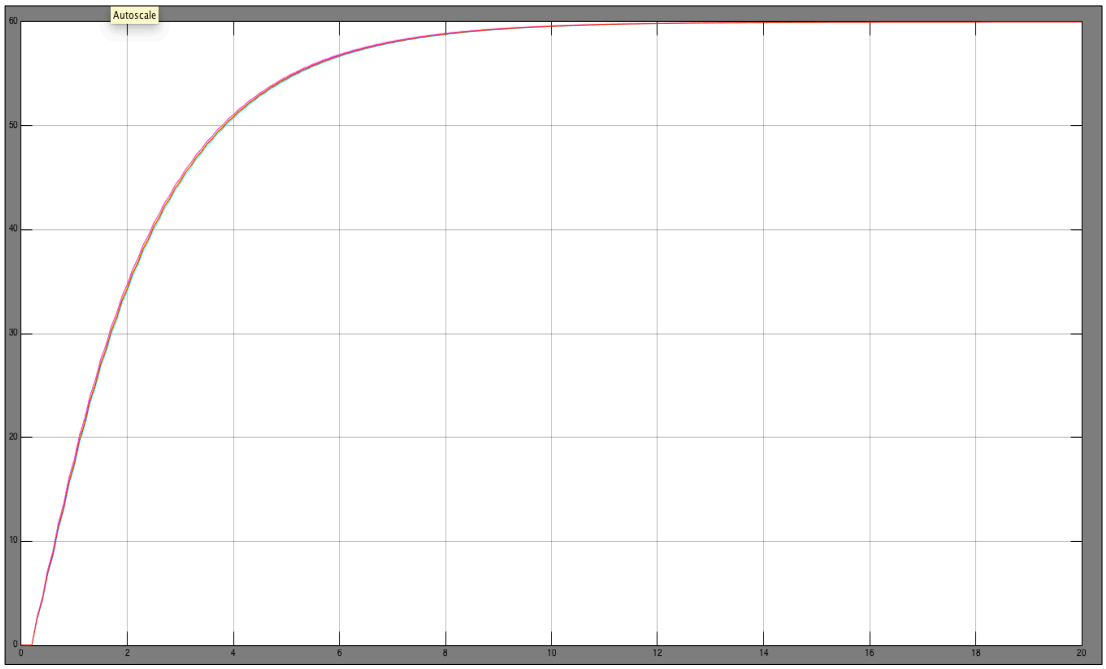
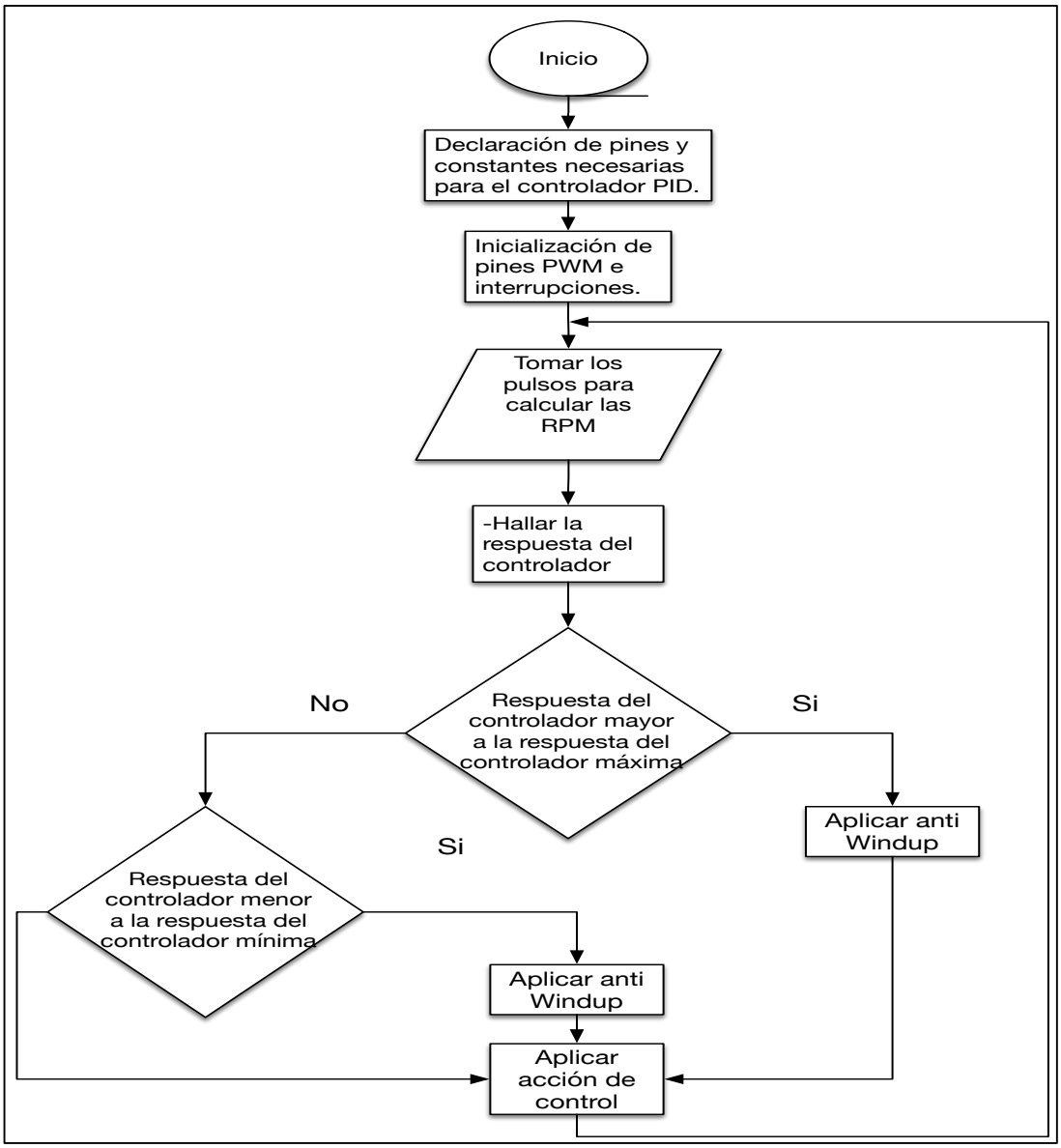


Figura52. Respuesta de los controladores en *stateflow.*



con este control se podría afirmar que los motores de la plataforma robótica tienen tanto el régimen transitorio como el régimen permanente. La aceleración y velocidad son iguales para los cuatro motores, por lo tanto los movimientos que realice son mas uniformes que los anteriores. En el Arduino se trascribio la programación del *Stateflow* a un *Sketch* del mismo, para que este realice la acción de control PID. En el siguiente diagrama de flujo de la Figura53 se puede observar la secuencia que se realizó en el Arduino para el controlador PID, este proceso se hace para cada uno de los motores.

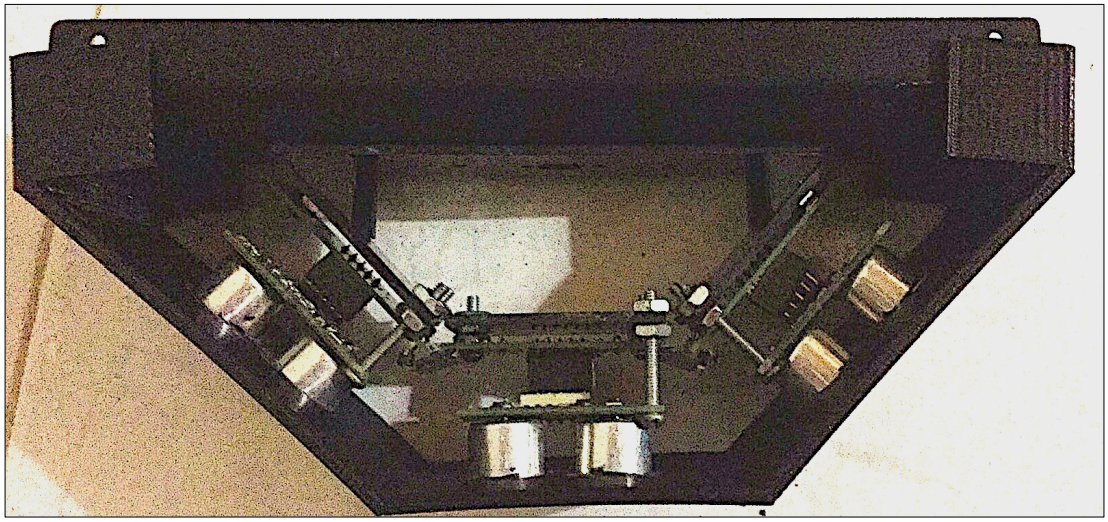
Figura53. Proceso del controlador PID



4.1.3 Sensores

Los sensores de ultrasonidos son los encargados de la detección de obstáculos; para la estructura robótica se utilizaron los SRF05 lo cuál trabajan en un rango de 1.7 cm a 4m. Estos sensores están ubicados como se muestra en la Figura54.

Figura54. Sensores de ultrasonidos.



Se utiliza el pin (4) del Arduino Mega para enviar un pulso de $10\mu s$ (tiempo de pulso) al trigger input de los tres sensores de ultrasonidos los cuáles están conectados entre sí, si se detecta algún obstáculo, el pin echo de los sensores detecta el sonido y cambia su estado a LOW, el ancho de pulso que tardo en ir y volver, será la distancia que hay del sensor al obstáculo como se observó en la Figura14. En la Figura55 se puede observar el código empleado para el funcionamiento de los sensores.

Figura55. Módulo de sensores de ultrasonidos.

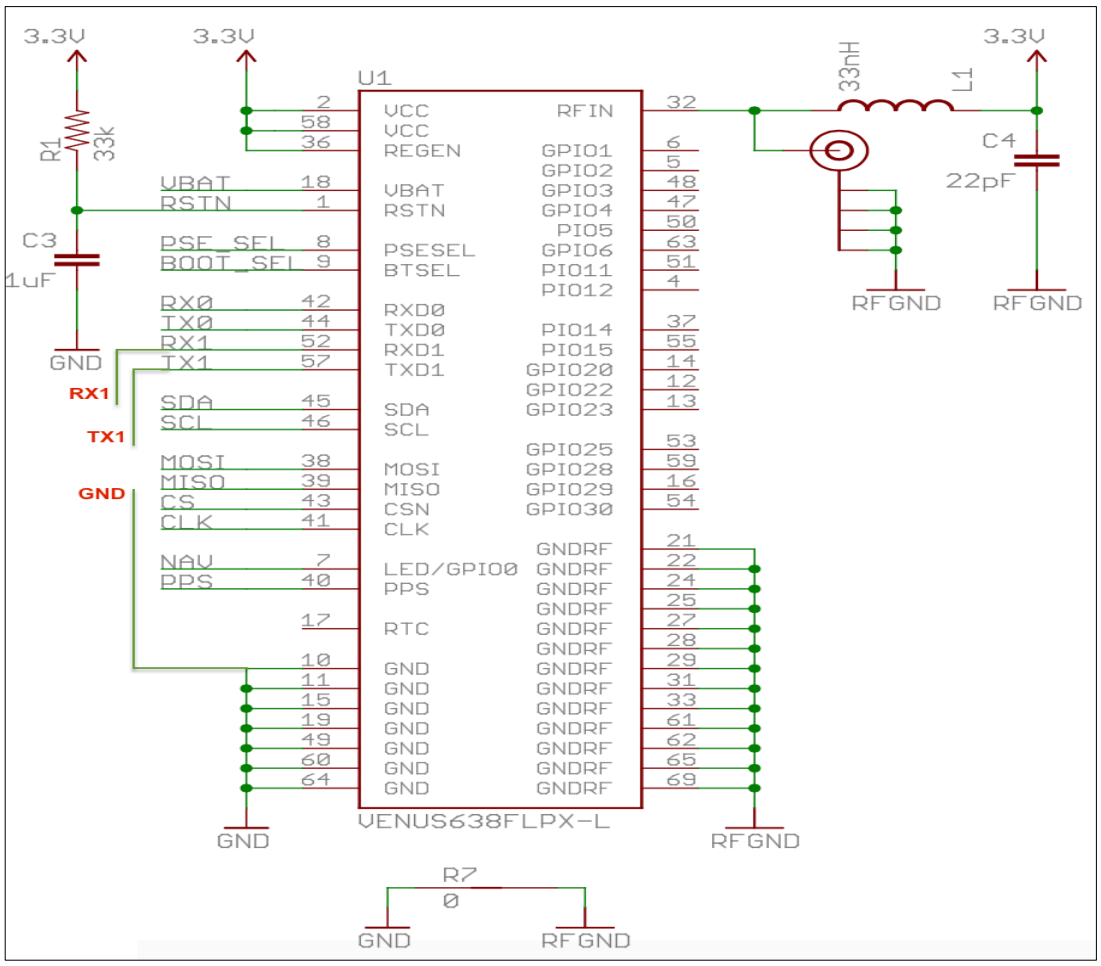
```
const int trigger=4;
const int echo=5;
const int echo2=6;
const int echo3=13;


float distance,distance2,distance3;

void setup(){
  Serial.begin(9600);
  pinMode(trigger,OUTPUT);
  pinMode(echo,INPUT);
  pinMode(echo2,INPUT);
  pinMode(echo3,INPUT);

}

void loop()
{
  //INICIALIZAMOS EL SENSOR 1
  digitalWrite(trigger,LOW);
  delayMicroseconds(5);
  // Comenzar  mediciones
  // Enviar una señal activando la salida trigger durante 10 microsegundos
  digitalWrite(trigger,HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger,LOW);
  // Adquirirs los datos y convertir a la medida a metros
  distance=pulseIn(echo,HIGH);
```

Para enviar de nuevo un pulso, se debe esperar $50\,ms$ como mínimo, esto asegura que el 'bip' ultrasónico haya desaparecido completamente y no ocasione un falso eco en la siguiente medición de distancia.

4.1.4 Módulo GPS

El GPS transmite la información al microcontrolador por puerto serial, para poder comunicar el módulo al Arduino Mega, se usan algunas librerías como *SoftwareSerial.h* y *Tinygps.h*. La primera permite configurar el Arduino Mega con un segundo puerto serie específicamente sólo para el módulo GPS, en la Figura56 se pude ver el diagrama esquemático interno del GPS Venus. La segunda librería se encarga de convertir el código NMEA para poder ver la latitud y longitud donde se encuentra el dispositivo.

Figura56. Diagrama esquemático módulo GPS[56].



En la Figura57 se observa el código (tomado de[58])para el funcionamiento del GPS, se declara los pines (10, 11 – RX, TX respectivamente) del Arduino Mega como puerto serial a 9600 baudios para la recepción de datos de los satélites al GPS. Se pregunta si en el puerto serial hay algún dato, si lo hay entonces se muestran en pantalla la longitud y latitud, esto es posible gracias a la librería *Tinygps.h* que se encarga de convertir los datos NMEA. El GPS Venus tiene un led rojo el cuál indica la recepción de satélites para la obtención de datos, lo mínimo para indicar la posición que se encuentra el dispositivo es de 6 satélites, esto se sabe ya que el led rojo empieza a parpadear al momento de encontrar los satélites necesarios.

[56] <http://cdn.sparkfun.com/datasheets/Sensors/GPS/Venus_GPS-v21.pdf>
[58] <SPARKFUN. Tutorial. [online]. Disponible en: <https://www.sparkfun.com/products/11058>

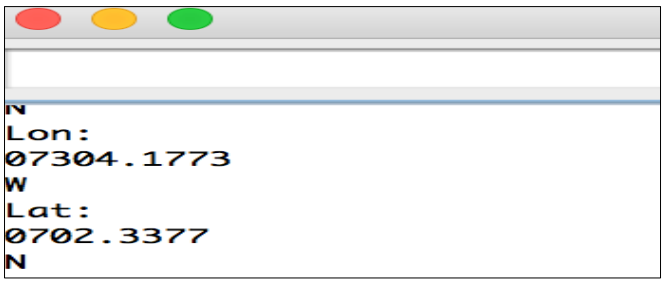
Figura57. Código de identificación del módulo GPS

```
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>    // LCD
#include <Tinygps.h>

SoftwareSerial gpsSerial(10, 11); // RX, TX (TX not use
const int sentenceSize = 80;

char sentence[sentenceSize];


void setup()
{
  Serial.begin(9600);
  gpsSerial.begin(9600);
  lcd.begin(16, 2);
  lcd.print("Esperando GPS...");
}

void loop()
{
  static int i = 0;
  if (gpsSerial.available())
  {
    char ch = gpsSerial.read();
    if (ch != '\n' && i < sentenceSize)
    {
      sentence[i] = ch;
      i++;
    }
    else
    {
     sentence[i] = '\0';
     i = 0;
```

Para realizar la prueba, los datos obtenidos se mostraron en una pantalla LCD 2x16, esta pantalla solo fue para hacer la prueba, en la plataforma robótica no se encuentra integrada.

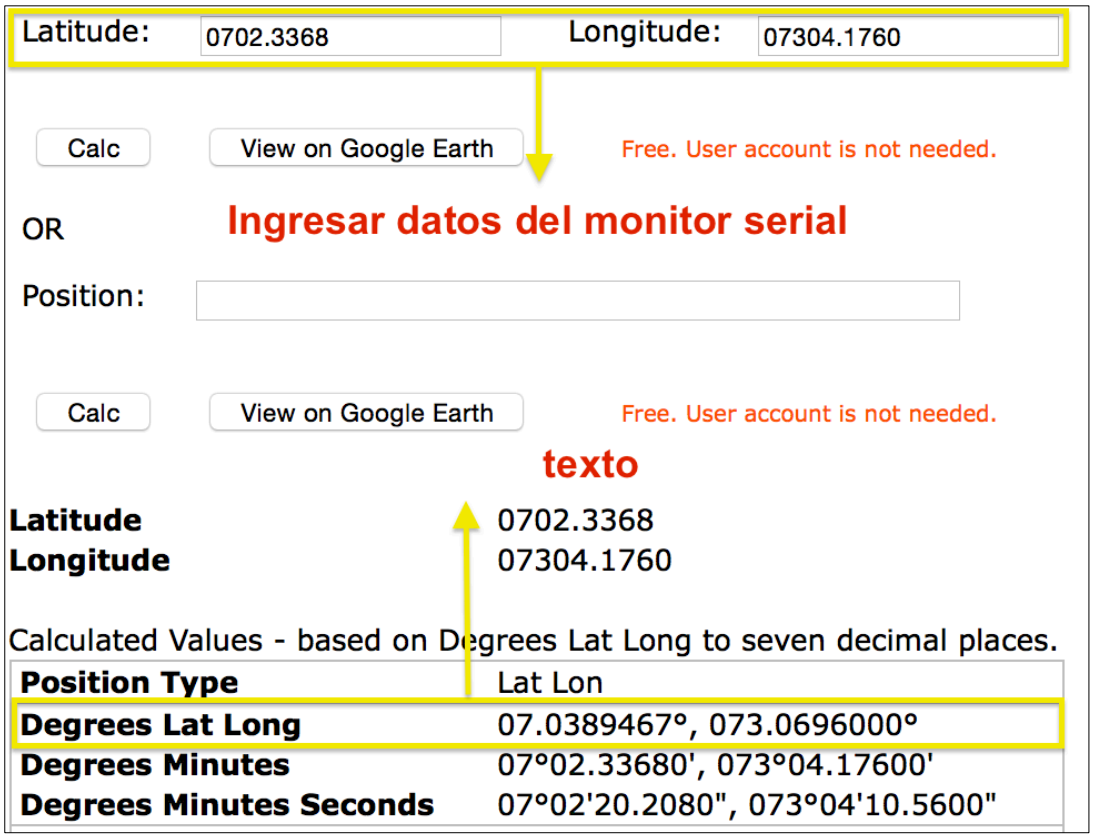
En la Figura58 se puede ver los datos del GPS en el monitor serial de Arduino, lo cual muestra tanto longitud como latitud, los datos como N (norte), E (este) son datos positivos al momento de buscar las coordenadas en Google Maps, S (sur), W (oeste), son datos negativos en Google Maps.

Figura58. Datos de latitud y longitud en monitor serial de Arduino.



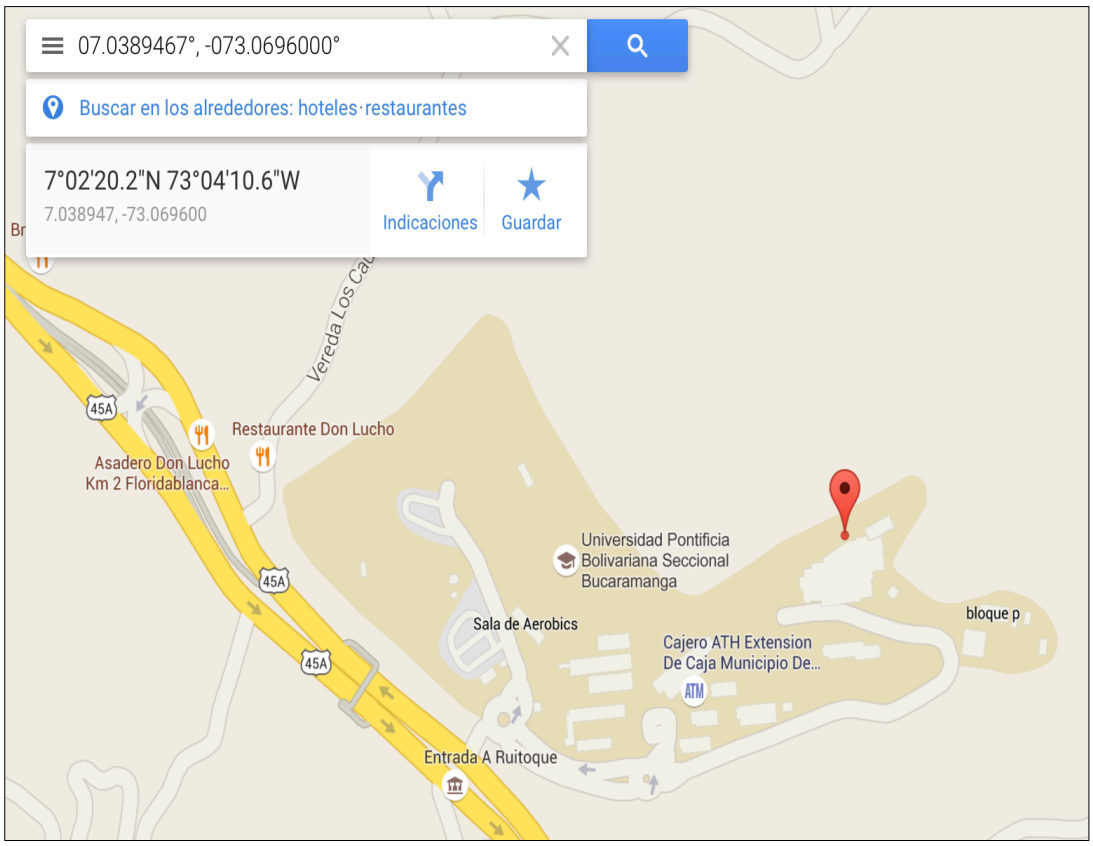
Los datos anteriores se ingresan en una pagina[59] para convertir las coordenadas (ver Figura59) y poderlas ingresar en Google Maps. En la Figura60 se muestra las coordenadas convertidas e ingresadas a Google Maps, muestra la ubicación del GPS en la Universidad Pontificia Bolivariana Seccional Bucaramanga.

Figura59. Página para convertir coordenadas.



[59] EARTH POINT. Convert Coordinates - Calculate a position in a variety of formats. 2015. [online]. Disponible en: < http://www.earthpoint.us/Convert.aspx>

Figura60. Coordenadas en Google Maps.



4.1.5 Estructura robótica

Se utilizó una estructura robótica (ver Figura61) robusta lo cuál fuera capaz de desempeñarse en cualquier situación de terreno. El chasis de esta plataforma está hecha por 4 partes de aluminio. Tanto la parte inferior como la superior están hechas de paneles lexan. En la Tabla7 se puede ver las medidas de la plataforma robótica.

Figura61. Estructura robótica[60].

Tabla7. Especificaciones plataforma robótica[61]

Largo	11.00” (27.94 cm)
Ancho	12.50” (31.75 cm)
Alto	4.75” (12.065 cm)
Chasis largo	8.75” (22.225 cm)
Chasis ancho	7.00” (17.78 cm)
Chasis alto	3.50” (8.89 cm)
Distancia al piso	1.63” (4.1402 cm)
Peso	4 lbs 6 oz (1.984 Kg)

La superficie de las llantas son de caucho con rin plástico, cada llanta tiene un diámetro de $12\ cm$ con 4 *hubs* para acoplar las llantas al eje del motor. La forma de las llantas le permiten atravesar diferentes terrenos como, arena, lodo, rocas, pasto, entre otros. La fuerza de la plataforma robótica depende del tipo de motor que se le acople, para este caso fueron cuatro motores de 16 Kg de torque a 100 rpm.

[60] Tomado de: <http://www.dynamoelectronics.com/index.php?page=shop.product_details&flypage=dynamo.tpl&product_id=1066&category_id=63&option=com_virtuemart&Itemid=58>

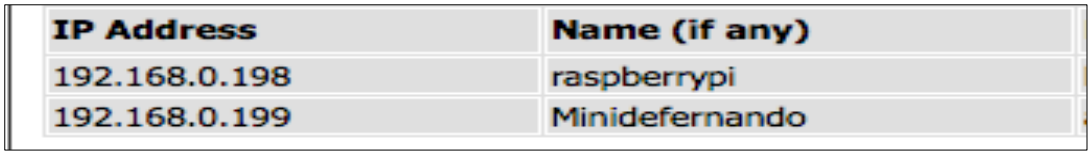
[61] DYNAMO. 4WD2 Plataforma robótica 4x4 configurable. Septiembre 5, 2015. [online]. Disponible en: <http://www.dynamoelectronics.com/index.php?page=shop.product_details&flypage=dynamo.tpl&product_id=1066&category_id=63&option=com_virtuemart&Itemid=58>

4.1.6 Raspberry Pi

Para utilizar la Raspberry Pi 2 se eligió como sistema operativo la ultima versión de Raspbian, ya que este sistema es el que ofrece mayor compatibilidad con los paquetes de software desarrollado por diferentes comunidades y empresas. Para acceder al dispositivo de computación embebida sin la necesidad de utilizar un teclado y una pantalla se utiliza el *Secure Shell (SSH)* el cual es una aplicación que utiliza el puerto 22 del protocolo TCP/IP[62].

Para realizarlo es necesario conectar la computadora y la tarjeta de computación embebida al mismo *router* por medio de cable Ethernet y accediendo a este por medio de un navegador como Google Chrome se puede conocer la dirección IP que se le asigna a la Raspberry Pi. Como se puede observar en la Figura62.

Figura62. Dirección IP de la Raspberry Pi conectada por cable Ethernet

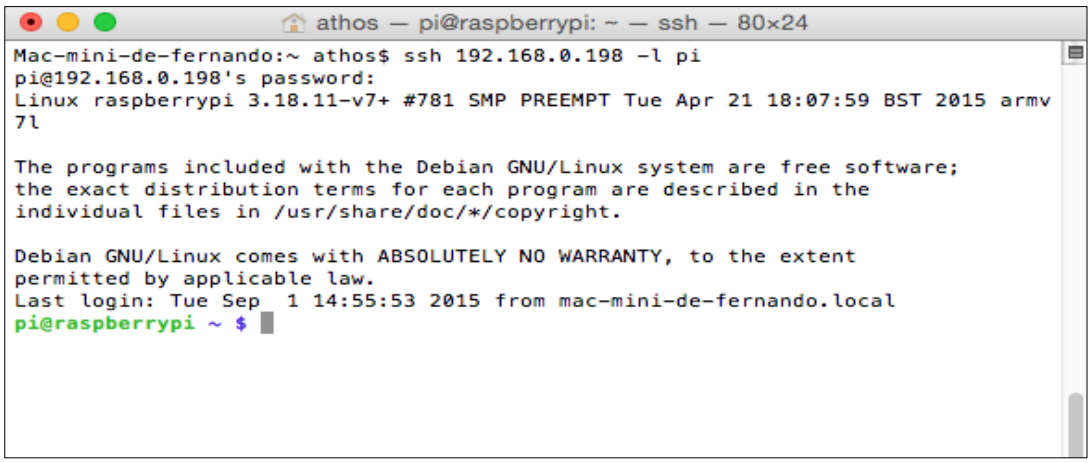


IP Address	Name (if any)
192.168.0.198	raspberrypi
192.168.0.199	Minidefernando

Luego desde el terminal de la computadora con un sistema operativo basado en Linux se escribe el comando ssh y posteriormente la dirección IP que se observó anteriormente, a esto se le agrega el nombre de usuario como se observa en la Figura63.

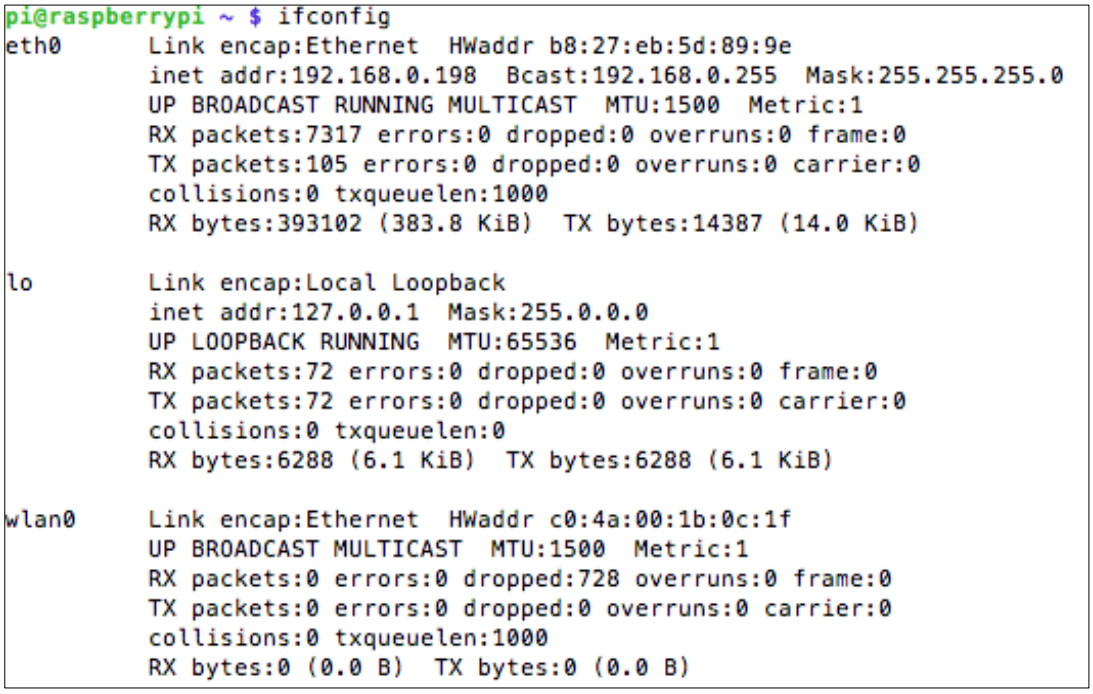
[62] Javier Smaldone. Introducción a Secure Shell. Enero 20, 2004. [online]. Disponible en: < http://es.tldp.org/Tutoriales/doc-ssh-intro/introduccion_ssh-0.2.pdf>

Figura63. Conexión ssh a Raspberry pi.



Posteriormente se conecta a un puerto usb el dispositivo TL-823N desarrollado por tp-link el cual permite a la Raspberry Pi tener conectividad a Wi-Fi. Es necesario ejecutar el comando *ifconfig* para conocer las interfaces de red que reconoce la tarjeta embebida. En la Figura64 se observa la ejecución de este comando, el cual informa la existencia de un dispositivo Wi-Fi nombrado wlan0

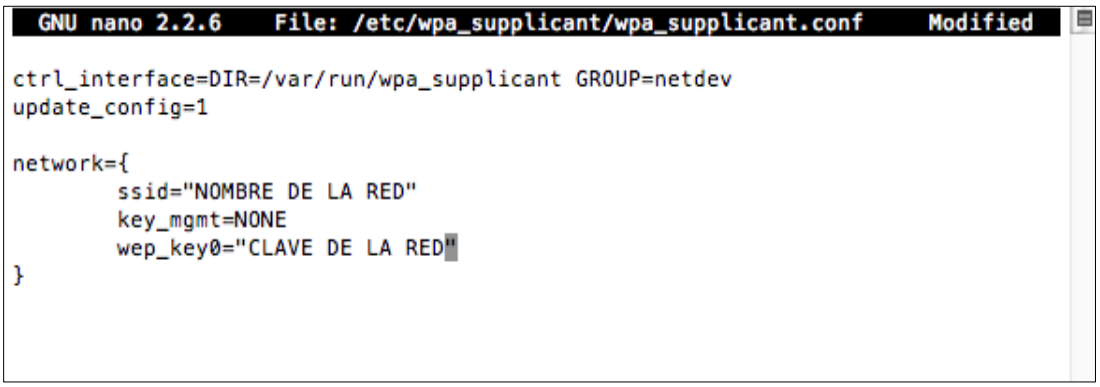
Figura64. Respuesta del comando ifconfig.



Para utilizar este dispositivo es necesario indicarle a que red debe conectare y su respectiva autenticación, para realizar esta conexión y poder desconectar la tarjeta embebida del cable Ethernet, es necesario modificar el archivo *wpa_supplicant.conf* que se encuentra en la ruta */etc/wpa_supplicant/;* para lograr desarrollar esto se necesita autenticar el usuario de la Raspberry Pi por medio del comando *sudo* ejecutando el comando de edición de texto *nano* junto a la ruta del archivo que se desea modificar es decir en el terminal es necesario ejecutar la línea de código que se observa en la Figura65, este archivo debe ser modificado como se muestra en la Figura66.

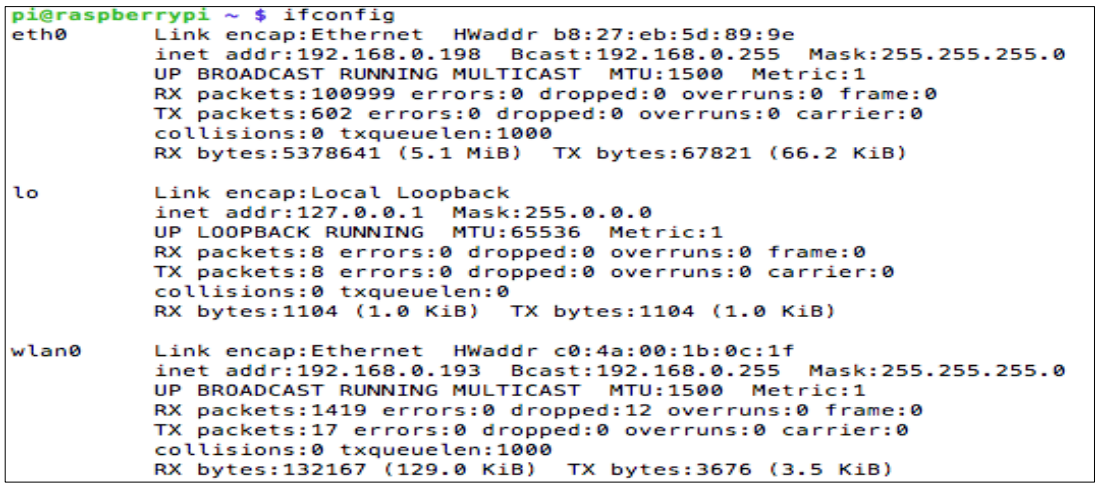
Figura65. Código para modificar el archivo wpa_supplicant.conf.

Figura66. Contenido modificado del archivo wpa_supplicant.conf.



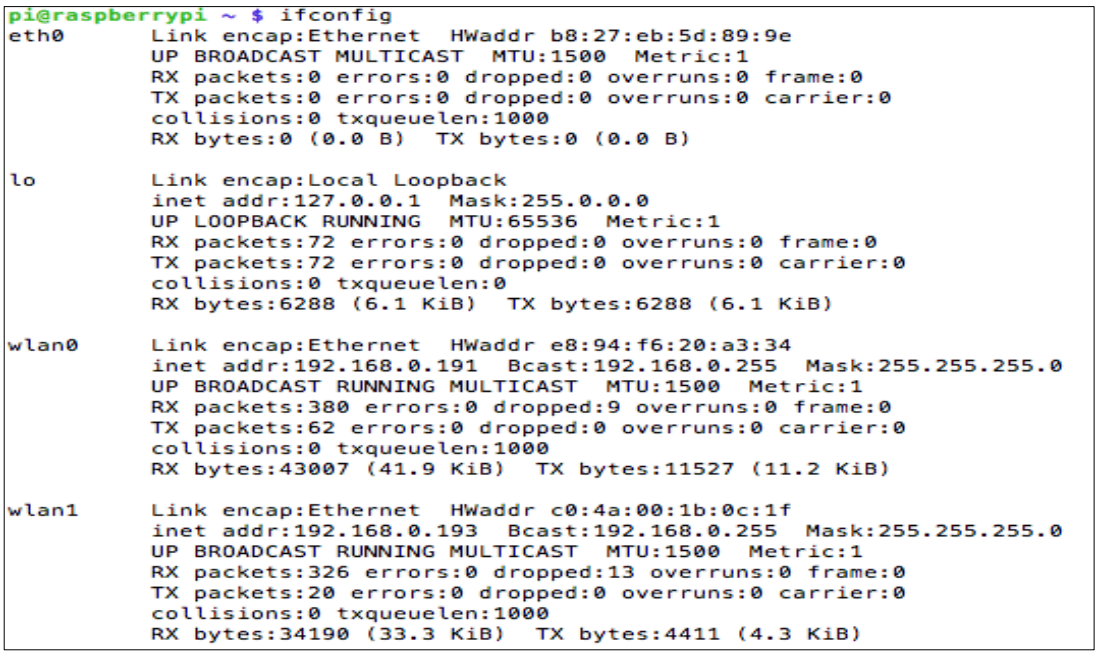
Para que el sistema operativo ejecute los cambios realizados por este archivo es necesario reiniciarlo utilizando la línea de comando *sudo reboot*. Cuando se vuelve a iniciar sesión en Raspbian se ejecuta el comando *ifconfig* para observar si se estableció la conexión con la red Wi-Fi. Como se observa en la Figura67 el dispositivo wlan0 se conecto a la red y esta proporcionó una dirección IP.

Figura67. Conexión inalámbrica de Raspberry pi.



Como la tarjeta embebida debe tener la capacidad de conectarse a una red y crear otra para realizar una conexión punto a punto, es necesario conectar otro dispositivo Wi-Fi desarrollado por TP-LINK, este dispositivo es el modelo WN 725N, el cual no tiene el controlador instalado por defecto en Raspbian, para hacerlo funcionar se necesita instalar el controlador adecuado para la versión de Raspbian que se tiene instalado, reiniciar la tarjeta embebida y observar con el comando *ifconfig* si este es reconocido por la Raspberry pi. En la Figura68 se observa que el dispositivo fue reconocido y además se conectó a la misma red Wi-Fi ,mencionada anteriormente.

Figura68. Conexión a la misma red por medio de los dos dispositivos Wi-Fi.



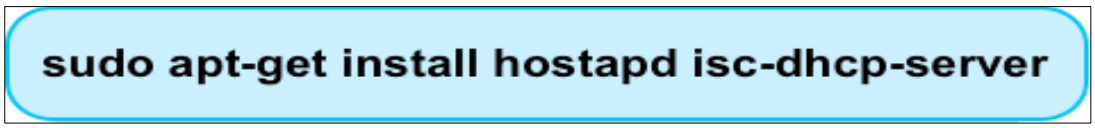
El primer TP-LINK que reconoce la tarjeta embebida se configura para crear una red Wi-Fi y así permitir que los dispositivos móviles pueda comunicarse con la Raspberry Pi, de tal modo que éste pueda recibir las acciones enviadas por la aplicación y al mismo tiempo visualizar la señal de video enviada por la tarjeta embebida. Este tipo de comunicación es conocido como *full-duplex* en donde se transmite y recibe al mismo tiempo por el mismo medio de transmisión[63].

Para convertir este dispositivo como un punto de acceso que transmite la señal de una red creada por la tarjeta embebida, es necesario la utilización de dos paquetes de software los cuales se descargan e instalan por medio del comando *install*. En la Figura69 se observa la ejecución de este comando para instalar los paquetes de software necesarios[64].

[63] Dani Korpi, Lauri Anttila, Mikko Valkama, Risto Wichman Taneli Riihonen, Ville Syrja¨la¨. Full-Duplex Transceiver System Calculations: Analysis of ADC and Linearity Challenges. Junio 5, 2013. [online]. Disponible en < http://www.cs.tut.fi/~valkama/full-duplex/DKorpi_TWireless_submit.pdf >

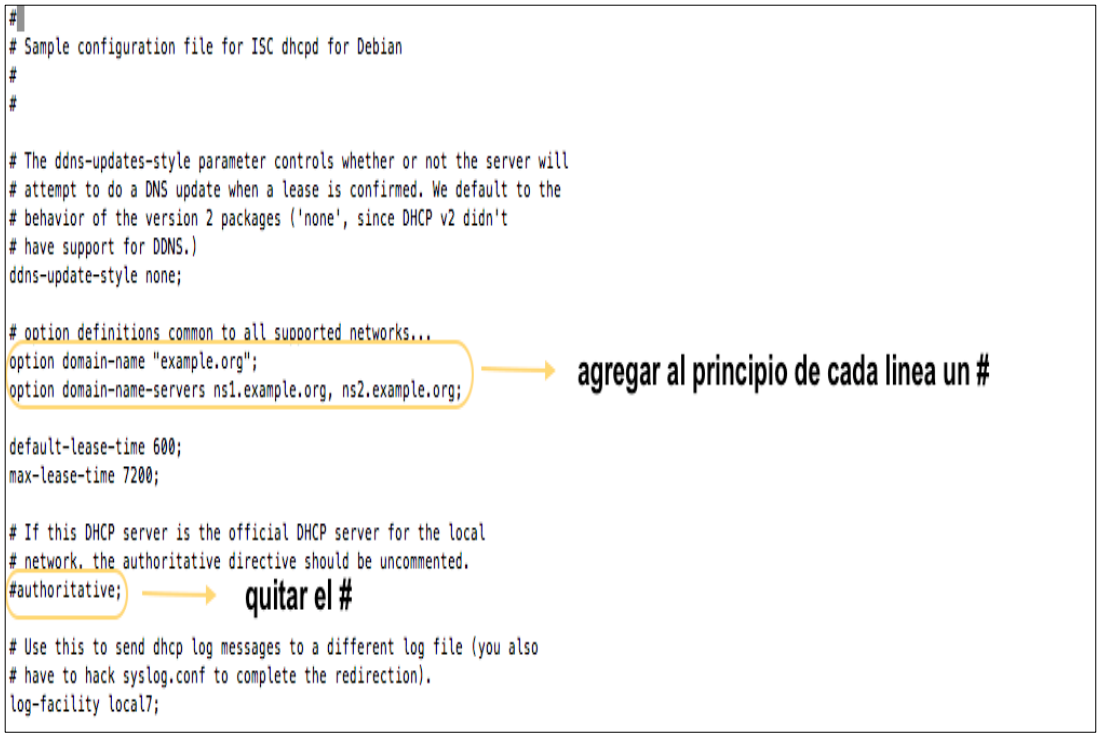
[64] GEEKYTHEORY. Instalar los programas necesarios. Mayo 11, 2015. [online]. Disponible en: < https://geekytheory.com/tutorial-rasbperry-pi-como-crear-un-punto-de-acceso-wifi/ >

Figura69. Instalación de programas para la creación de una red Wi-Fi.



Una vez instalado los paquetes de software, se procede a configurar sus archivos para poder establecer la conexión, lo primero que se realizó fue la modificación del archivo *dhcpd.conf,* el cual se encuentra en la ruta */etc/dhcp/*. Es necesario agregar y quitar algunos # dentro de este archivo, los # impiden la ejecución de esa línea al usar el archivo *dhcpd.conf* . En la Figura70 se muestra la líneas importantes que se tienen que modificar[65].

Figura70. Primera modificación del archivo dhcpd.conf.



Adicionalmente se agrega al final del archivo las líneas que se muestran en la Figura71, estas líneas controlan parámetros tales como la dirección IP del punto

[65] Ibid., Configurar el servidor DHCP.

de acceso, la máscara de *subred*, el rango de direcciones que se otorgan a los dispositivos que se conectan a esta red, entre otros.

Figura71. Líneas agregadas al final del archivo dhcpd.conf.

```
subnet 192.168.42.0 netmask 255.255.255.0 {
    range 192.168.42.2 192.168.42.20;
    option broadcast-address 192.168.42.255;
    option routers 192.168.42.1;
    default-lease-time 600;
    max-lease-time 7200;
    option domain-name "local";
    option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

Se modifica el archivo *isc-dhcp-server* como se muestra en la Figura72 este archivo se encuentra en la ruta /etc/default.

Figura72. Modificación del archivo isc-dhcp-server.

```
# Defaults for isc-dhcp-server initscript
# sourced by /etc/init.d/isc-dhcp-server
# installed at /etc/default/isc-dhcp-server by the maintainer scripts

#
# This is a POSIX shell fragment
#

# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
#DHCPD_CONF=/etc/dhcp/dhcpd.conf

# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPD_PID=/var/run/dhcpd.pid

# Additional options to start dhcpd with.
#       Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
#       Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACES="wlan0"
```

Para indicarle al dispositivo Wi-Fi que dirección debe tener al ser creada la red, se desactiva utilizando la línea *sudo ifdown wlan0* y luego se modifica el archivo *interfaces*, este se encuentra en */etc/network/.* Con estas modificaciones se configura la Raspberry Pi para crear la red a partir del primer dispositivo que detecte, ya que este recibe el nombre de *wlan0.* Al finalizar esta modificación se debe ejecutar la línea de comando *sudo ifconfig wlan0 192.168.42.1;* esta dirección debe ser la misma que se ingresó al archivo *interfaces*[66] ver Figura73.

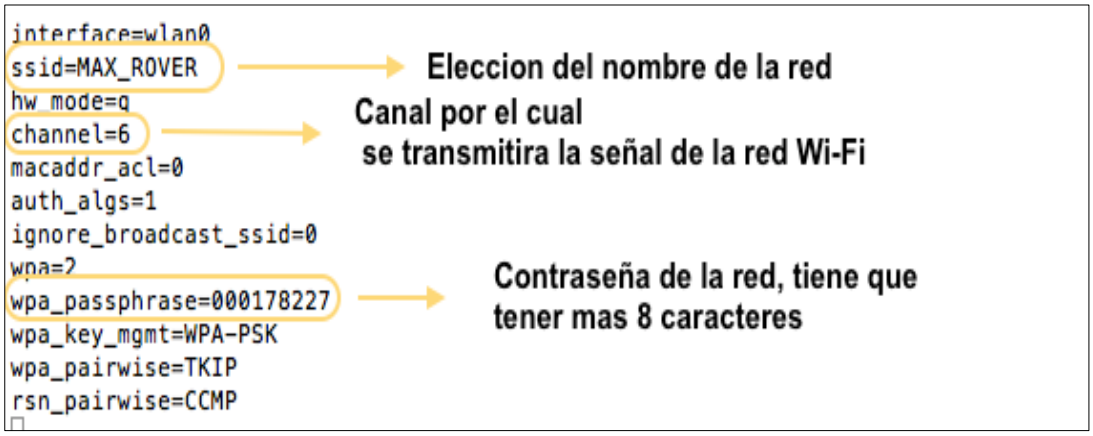
Figura73. Modificación realizada al archivo interfaces.

```
auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
iface eth0 inet manual

allow-hotplug wlan1
auto wlan1
iface wlan1 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan0
iface wlan0 inet static
address 192.168.42.1
netmask 255.255.255.0
#iface wlan0 inet manual
#wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
#iface default inet dhcp
```

[66] Ibid., Configurar una IP estática en wlan0.

El archivo *hostapd.conf* si existe se debe modificar, en caso contrario se debe crear este archivo y ubicarlo en la extensión */etc/hostapd/*. Este archivo le permite configurar opciones propias de las redes Wi-Fi. En el se puede elegir el nombre de la red, el tipo de encriptación, la contraseña para acceder, entre otras características. En la Figura74 se observa la configuración que se aplicó a la red creada[67].

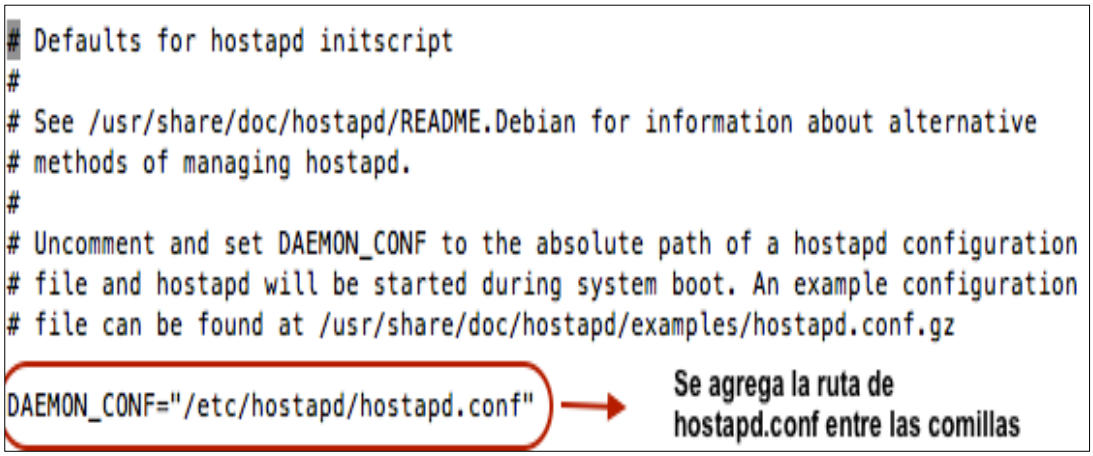
Figura74. Archivo hostapd.conf.



Para iniciar la configuración del archivo *hostapd.conf* es necesario indicarle al sistema operativo la ubicación de este. Para esto se agrega la extensión, ejecutando la edición del archivo el cual se tiene acceso a través del comando *sudo nano /etc/default/hostapd*. En la Figura75 se muestra la modificación que se tiene que realizar.

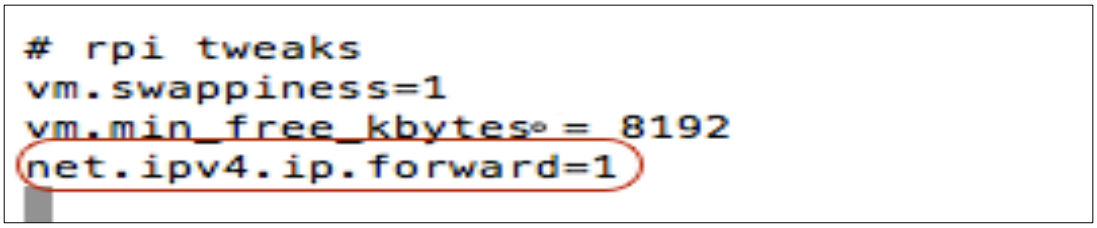
[67] Ibid., CONFIGURAR EL PUNTO DE ACCESO.

Figura75. Modificación para que el sistema operativo cargue el archivo hostapd.conf.



Ahora es necesario configurar el NAT, el cual permite pasar los paquetes de información de una IP privada a una IP pública, lo que permite que todas las conexiones sean enviadas con la misma IP pública[68]. Para esto se utiliza el archivo *sysctl.cof* el cual será modificado, como se aprecia en la Figura76. En donde se agregó la última línea al final del archivo. Adicionalmente es necesario ejecutar las líneas de comando que se observa en la Figura77; estas son necesarias para dar los permisos de ejecución al NAT[69].

Figura76. Configuración de sysctl.cof.



[68] XATAKAON. NAT (Network Address Translation): Qué es y cómo funciona. Agosto 24, 2011. [online]. Disponible en: < http://www.xatakaon.com/tecnologia-de-redes/nat-network-address-translation-que-es-y-como-funciona >
[69] Op. cit., Configuración de NAT

Figura77. Otras modificaciones para el funcionamiento de la NAT.

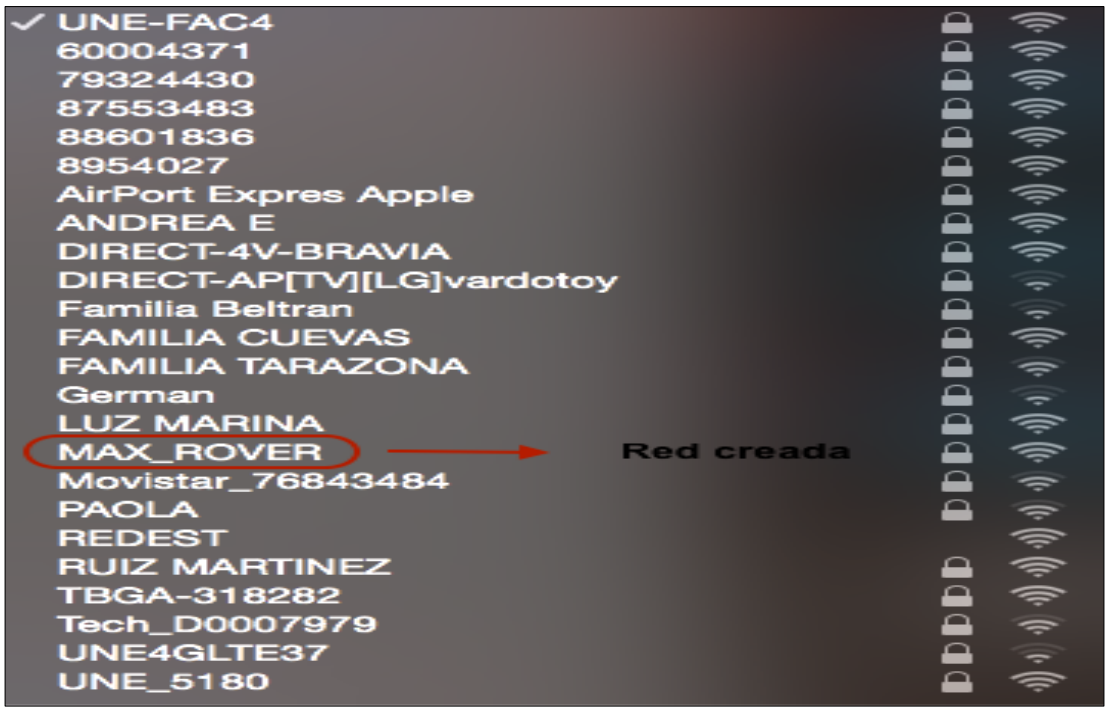
```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m estado --state RELACIONADOS, ESTABLECIDO -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

Una vez instalado y modificado todo este software es necesario habilitar la ejecución de estos archivos al inicio de arranque de Raspbian, para eso se ejecuta la líneas de comandos observadas en la Figura78, el resultado final es lo observado en la Figura79

Figura78. Comandos ejecutados en el terminal para crear la red al iniciar Raspbian.

```
pi@raspberrypi ~ $ sudo service hostapd start
[ ok ] Starting advanced IEEE 802.11 management: hostapd.
pi@raspberrypi ~ $ sudo service isc-dhcp-server start
[ ok ] Starting ISC DHCP server: dhcpd.
pi@raspberrypi ~ $ sudo update-rc.d hostapd enable
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LC_CTYPE = "UTF-8",
        LANG = "en_GB.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
update-rc.d: using dependency based boot sequencing
pi@raspberrypi ~ $ sudo update-rc.d isc-dhcp-server enable
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LC_CTYPE = "UTF-8",
        LANG = "en_GB.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
update-rc.d: using dependency based boot sequencing
pi@raspberrypi ~ $
```

Figura79. Prueba de creación de la red en Raspberry Pi.



Para transmitir el video captado por la cámara de la Raspberry Pi de forma inalámbrica y visualizarlo en la aplicación, es necesario enviar esta información usando la conectividad Wi-Fi que existe entre la tarjeta de computación embebida y el dispositivo móvil. Para realizar esto se eligió el paquete de software *motion,* ya que existe un paquete sub derivado que brinda compatibilidad con la cámaras conectadas al puerto CSI de la tarjeta de computación embebida.

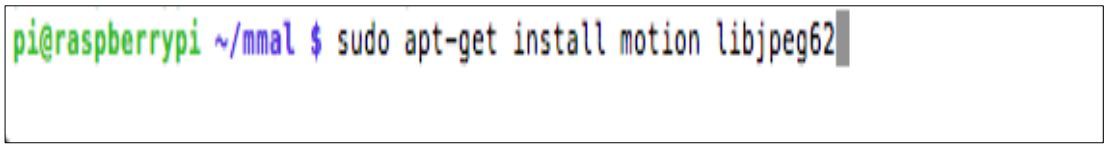
Motion es un software escrito en C y distribuido para Linux que convierte cualquier cámara instalada a la Raspberry pi en una cámara IP en la cual supervisa la señal de video para detectar cambios en una parte significativa de la imagen[70], este proceso se realiza comparando el cambio de los pixeles que se observan por la cámara. Además posee unas herramientas integradas que se pueden activar, desactivar o cambiar para obtener una mejor transmisión de video[71].

Motion genera múltiples archivos de extensión JPEG, por lo tanto Raspbian tiene que soportar el manejo de estos tipos de imágenes y para ello se instala "libjpeg62" el cual es una biblioteca de ejecución independiente para el manejo de

[70] MOTION. Motion - Motion Examples. Octubre 12, 2004. [online]. Disponible en: < http://www.lavrsen.dk/foswiki/bin/view/Motion/MotionExamples >
[71] MOTION. What is Motion?. Julio 30, 2015. [onine]. Disponible en: < http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome >

estos archivos[72] para instalar en el sistema operativo estos dos paquetes de software es necesario escribir en el terminal lo que se observa en la Figura80.

Figura80. instalación de los paquetes necesarios para transmitir video.



Cuando finalice la instalación de los paquetes de software es necesario descargar los archivos que permite la compatibilidad de estos programas con la cámara conectadas al puerto CSI de la Raspberry Pi. Para esto es necesario ejecutar el comando que se muesta en la Figura81, este realiza una descarga de un archivo comprimido que se encuentra alojado en Dropbox. Este se guarda en una carpeta creada por el usuario; para realizar esto se utiliza el comando “mkdir” junto al nombre deseado de la nueva carpeta, dentro de ella es necesario crear otra para almacenar los archivos que genera *motion*.

Figura81. Comando para descargar el paquete de compatibilidad.

Una vez descargado este archivo se descomprime utilizando el comando “tar –zxvf” , este requiere el nombre exacto junto con sus extensiones para poder ser ejecutado, al realizar este proceso se observa la existencia de dos archivos más dentro de la carpeta que se descargó, el *motion-mmalcam.conf y el motion.log.*

Se debe modificar algunos parámetros dentro del archivo *motion-mmalcam.conf* para visualizar una imagen de buena calidad y gran fluidez. Aunque existen muchos parámetros dentro de este archivo para este proyecto se modificaron los siguientes.

[72] UBUNTUUPDATES.ORG. Package "libjpeg62". [online]. Disponible en: < http://www.ubuntuupdates.org/package/core/vivid/main/base/libjpeg62>

- *logfile:* aquí se ingresa la dirección del archivo *motion.log* el cual se encuentra en la carpeta donde se descomprimió el archivo descargado de Dropbox. Para este proyecto la dirección es */home/pi/mmal/motion.log*

- *width* y *height:* son las resoluciones que se otorga al video transmitido por medio Wi-Fi, al reducir este parámetro genera videos de mayor fluidez, pero con áreas difíciles de observación, se usó como tamaño de la imagen 480x480 el cual genera una imagen de tamaño considerable.

- *Framerate:* es la tasa de *frames* capturados por cada segundo, entre mayor sea este parámetro se obtiene un video sin interrupciones pero con una calidad de imágen muy mpobrecida. Este parámetro junto a la resolución determinan la calidad de imágen y fluidez del video, por este motivo se eligió 25 *frames por* segundo, este configuración proporciona con una velocidad notoria en el video y nitidez en la imagen observada.

- *output_pictures:* este parámetro decide en qué momento empieza a guardar el video dentro de la carpeta especificada por el usuario, si se configura este parámetro como *off* se asegura que al momento de iniciar *motion* se comienza a generar el archivo que se guardará cuando la ejecución del software finalice.

- *target_dir:* aquí se ingresa la dirección en donde se va a guardar el archivo de video producido por la ejecución de *motion.*

- *stream_maxrate:* indica el máximo flujo de *framerate* de la transmisión, se recomienda utilizar en este campo 100.

Para volver la ejecución automática de este software al momento de encender la Raspberry Pi es necesario la creación de dos archivos .sh, uno para iniciarlo y otro para finalizar. Starmotion.sh ejecuta la inicialización de *motion*. El contenido de este software se observa en la Figura82.

Figura82. Contenido de startmotion.sh.

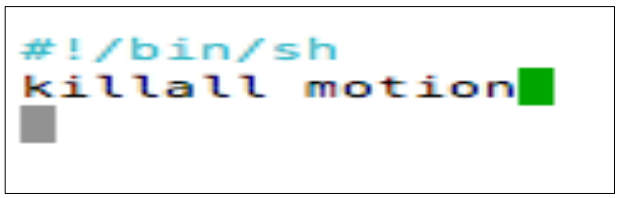
Al ejecutar este programa transmite la señal de video por el puerto “8081”, por lo tanto para visualizar su imágen basta con escribir la dirección IP de la tarjeta embebida seguido de “:8081” en un navegador de la computadora como Safari.

Esta computadora debe estar enlazado con la Raspberry Pi por medio de alguna de sus interfaces de comunicación Wi-Fi; dentro del navegador se observara lo que capta la cámara de la Raspberry pi. Como se muestra en la Figura83.

Figura83. Transmisión de lo observado por la cámara de la Raspberry Pi.

El archivo stopmotion.sh detiene la ejecución de motion, este guardará la información recopilada durante su ejecución para luego poder visualizar los videos y las fotos realizadas por el software. El contenido de este archivo es el que se visualiza en la Figura84.

Figura84. Contenido de stopmotion.sh.



Una vez creado estos archivos .sh es necesario decirle a Raspbian cual debe iniciar al encender la Raspberry pi. Por lo tanto se necesita modificar el archivo *rc.local*, el cual se encuentra en */etc/*. En este archivo se debe agregar al final de su contenido la dirección exacta de *startmotion.sh* En la Figura85 se observa esta modificación.

Figura85. Contenido modificado de rc.local.

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi


sh /home/pi/mmal/startmotion.sh

exit 0
```

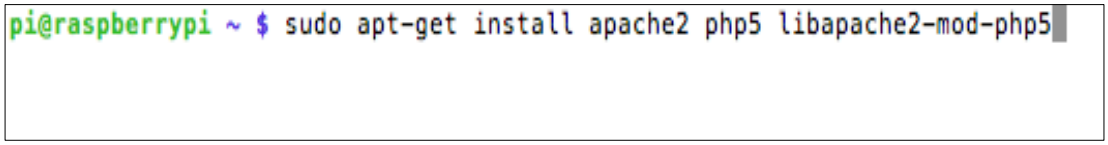
Para cerrar la ejecución de *motion, basta con apagar la* Raspberry Pi desde el terminal utilizando la línea *sudo poweroff*, la cual finaliza todas las ejecuciones que se están realizando en la tarjeta embebida.

Para que la Raspberry pi reciba las ordenes de la aplicación móvil utilizando como medio de transporte una señal Wi-Fi y a la vez envía la orden al Arduino mega, el

cual es el encargado de controlar los movimientos de la plataforma robótica es necesario implementar un software que permita la comunicación entre el robot y el dispositivo móvil, para eso se instala apache dentro de Raspbian para volver la tarjeta embebida como servidor.

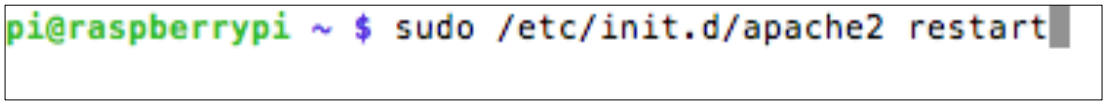
Es necesario ejecutar en el terminal las líneas que se muestran en la Figura86, adicionalmente se instala un interprete de lenguaje php para escribir archivos con esta forma de programación.

Figura86. Instalación de apache y php en Raspberry Pi.



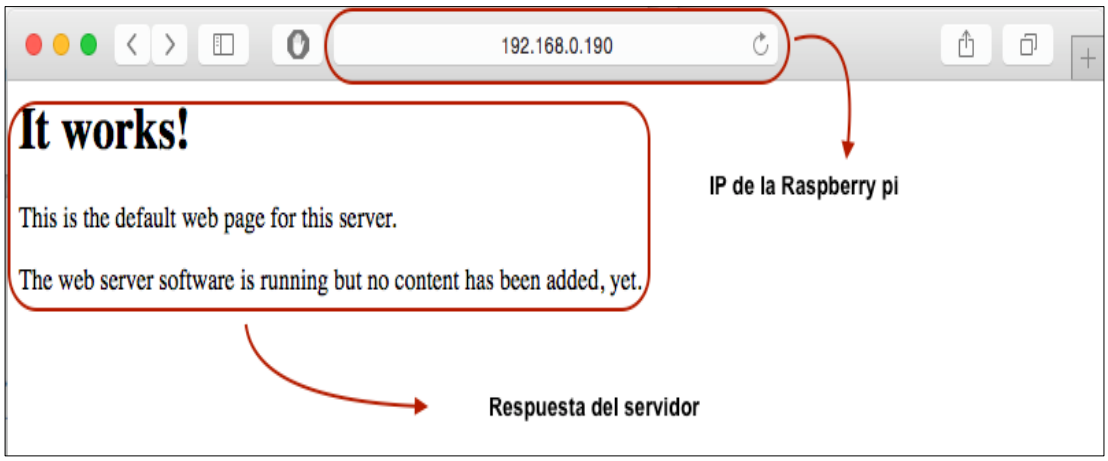
El modulo libapache2-mod-php proporciona soporte para que el servidor apache funcione con ficheros php [73]. Luego de instalar los paquetes de software necesarios para hacer funcionar la tarjeta embebida con apache es necesario reiniciarlo como se muestra en la Figura87. Para comprobar su funcionamiento es necesario observar en una computadora la respuesta de un navegador al escribir la dirección IP de la Raspberry Pi como se muestra en la Figura88.

Figura87. Comando para reiniciar el servidor.



[73] DEBIAN. Package: libapache2-mod-php5 (5.6.12+dfsg-1 and others). [online]. Disponible en: < https://packages.debian.org/sid/libapache2-mod-php5>

Figura88. Respuesta del servidor apache.



Por defecto los archivos del servidor que se necesitan crear se guardan en la ruta */var/www/*, desde allí por medio de un navegador se accede a todos los ficheros agregando al buscado la dirección IP, la ruta, el nombre y la extensión de este. En la Figura89 se observa un ejemplo de lo mencionado.

Figura89. Cargar un archivo PHP en el servidor.

Es esta la manera para recibir la orden de la aplicación móvil, que al mismo tiempo le enviara un dato al Arduino mega; este procesara el dato y ejecutara la acción correspondiente. Para comunicar la Raspberry pi con microcontrolador Atmega2560 es necesario ejecutar un *script* de Python, el cual será procesado a través de un archivo PHP.

En Python es necesario la inclusión de una librería para realizar comunicación serial con el Arduino, esta se descarga utilizando el terminal de Raspbian, para realizar esto se ejecuta el comando observado en la Figura90.

Figura90. Instalación de la librería serial de Python

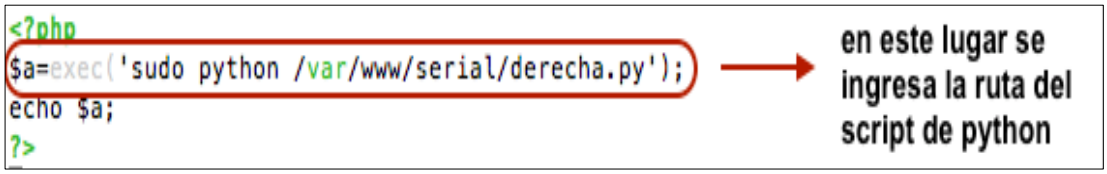
```
pi@raspberrypi /var/www $ sudo apt-get install python-serial
```

Esta librería permite ajustar la velocidad de transmisión de la comunicación y el dispositivo que se desea conectar, además se puede enviar y recibir datos[74] entre otras funciones. Un *script* de Python utilizado en el control de la plataforma robótica se observa en la Figura91, aquí se observa que antes de enviar el dato de la Raspberry pi es necesario esperar un tiempo de 2 segundos para que el Arduino mega lo recibe sin problemas y este pueda ejecutar la acción correspondiente.

Figura91. Código en Python para establecer una comunicación con el Arduino.

Para cargar el archivo de Python es necesario que el servidor cargue un archivo PHP el cual tiene en su código fuente la ruta del *script* que se desea ejecutar como lo muestra la Figura92.

Figura92. Código php para la ejución de un *script* en python.

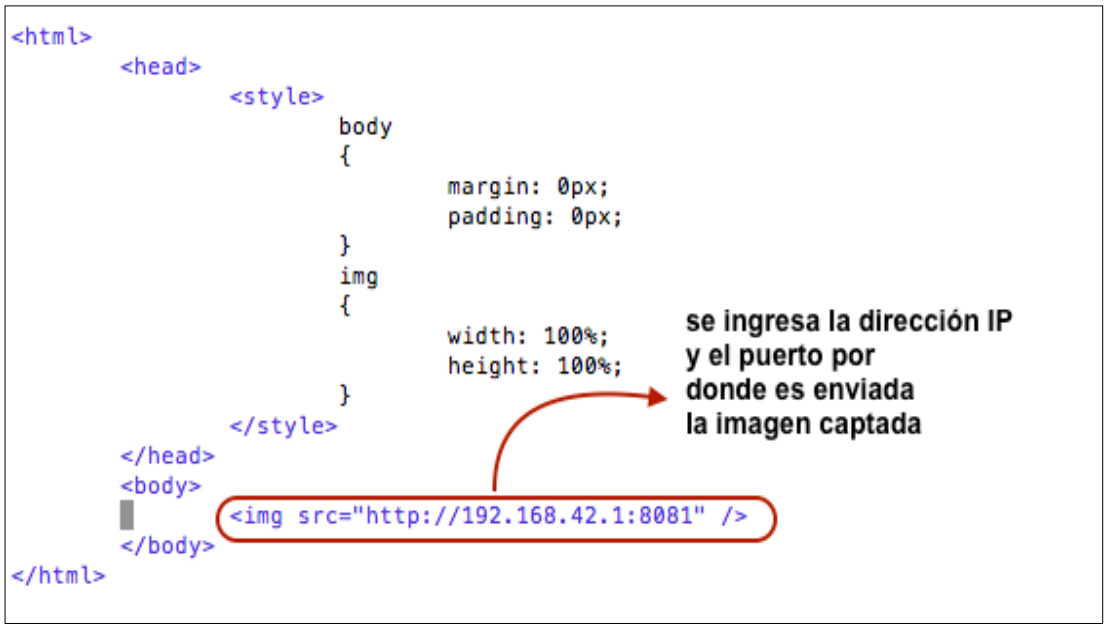


[74] MONK, Simon. Programming the Raspberry Pi. Getting Started with Python. 1 ed. McGraw-Hill. 170 p. ISBN 978-0071807838.

,

Para garantizar que el video de la cámara se pueda observar en cualquier dispositivo móvil es necesario cargar las imágenes que son captadas en un archivo html, el cual se aprecia en la Figura93; es importante decirle a este archivo la dirección IP y el puerto donde se visualiza normalmente en un navegador de la computadora.

Figura93. HTML para observar el video de la cámara de Raspberry pi.




```
<html>
        <head>
                <style>
                        body
                        {
                                margin: 0px;
                                padding: 0px;
                        }
                        img
                        {
                                width: 100%;
                                height: 100%;
                        }
                </style>
        </head>
        <body>
                <img src="http://192.168.42.1:8081" />
        </body>
</html>
```

4.2 CAPA DE CONTROL

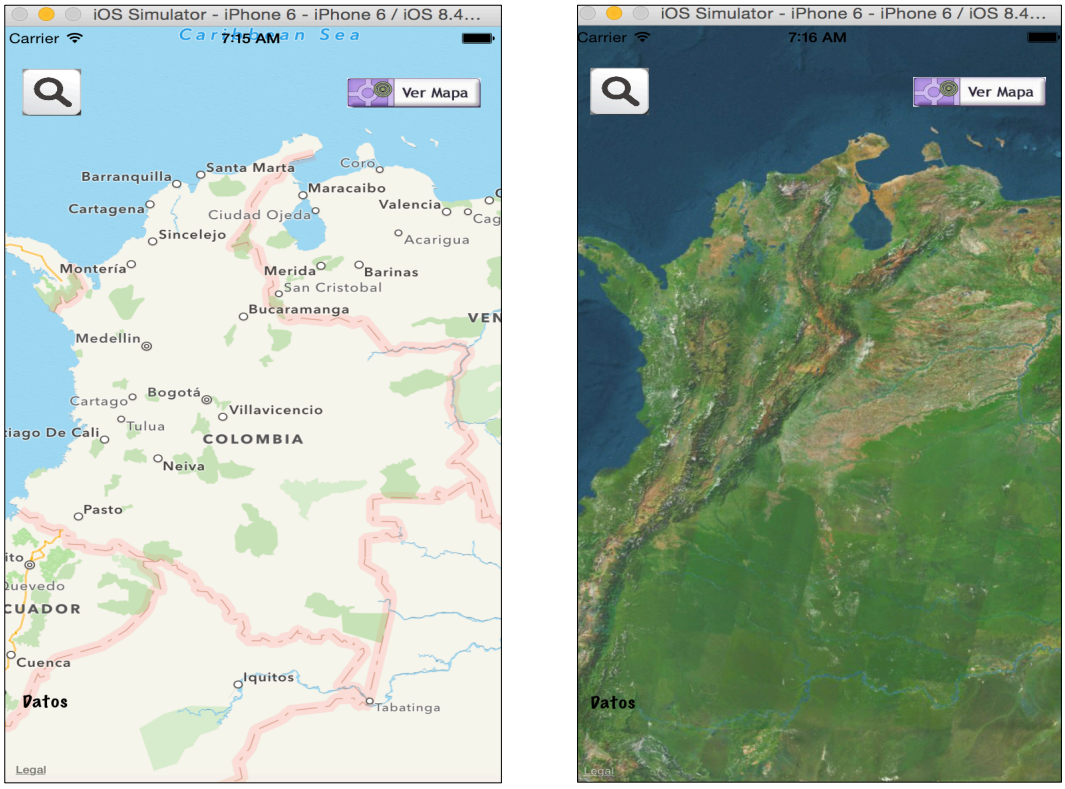
La capa de control se desarrolló en los sistemas operativos Android y IOS, para IOS (el cual se enfatizará más para este libro) específicamente para IPhone. La aplicación se desarrolló con el fin de tener una interfaz gráfica amigable con el usuario. Esta aplicación cuenta con dos tipos de conectividad, la primera por vía internet local, donde el usuario debe ingresar la dirección IP especifica a la red que este conectado. La segunda conectividad es punto a punto, donde la Raspberry Pi crea su propia red con una IP fija.

Al tener la conectividad ya realizada, la aplicación muestra otro tipo de vista donde se puede visualizar la grabación de la cámara en tiempo real, los botones para el control de la plataforma robótica, y un botón donde se puede ver la ubicación del usuario en el mapa que facilita Apple en el desarrollo de la aplicación.

4.2.1 Mapas

Se utiliza el sistema de posicionamiento global para ubicar al usuario que se encuentra controlando la plataforma robótica, la mejor forma para visualizar este tipo de información es por medio de la implementación de los mapas. *MapKit* es una herramienta basada en API de Google Maps y Google Earth y proporciona toda la información necesaria para el uso de los mapas en su *Framework*[76]. El elemento utilizado en la aplicación es la clase *MkMapView* el cuál permite la visualización del mapa con información satelital. La información del mapa puede ser presentada en forma *standard* o satelital como se puede observar en la Figura94.

Figura94. Representación de mapa.

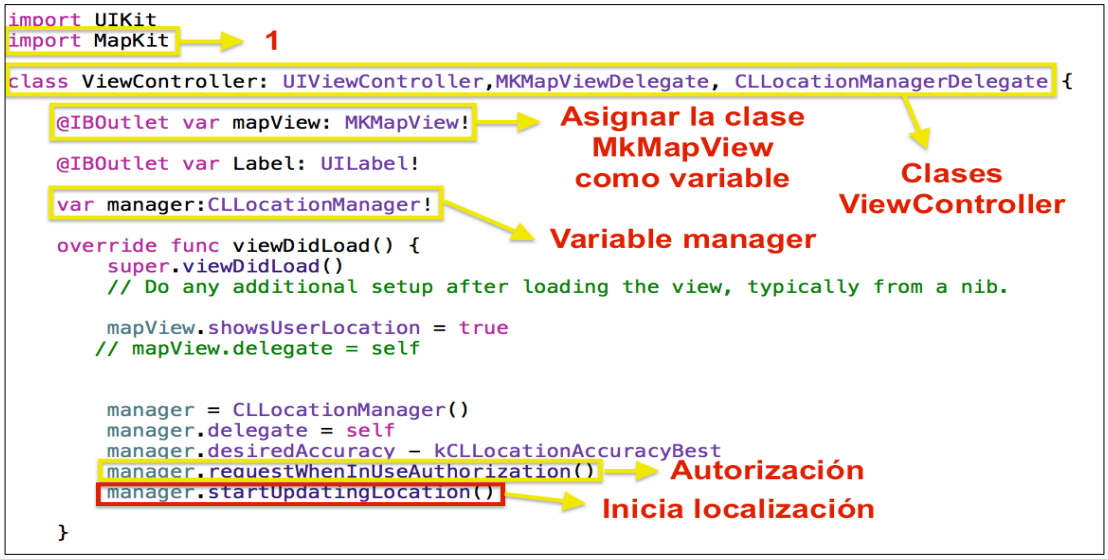


El mapa Tiene la función multi touch el cuál permite acercar el mapa haciendo zoom como normalmente se haría en un dispositivo Apple; lo mismo para realizar el desplazamiento.

[76] PÁEZ. Op. cit., p. 73.

En la Figura95 se observa el código (tomado de la siguiente página [77]) implementado para la localización en el mapa. En el número 1 muestra la importación de la librería *MapKi*t que es necesaria para mostrar el mapa; en la clase *ViewController* es necesario agregar *MkMapViewDelegate* que se encarga de hacer las delegaciones respectivas con los mapas. Además, también se debe agregar *CLLocationManagerDelegate* el cuál permite utilizar las delegaciones necesarias para mostrar las coordenadas de ubicación, como lo son latitud, longitud, velocidad, curso, país, entre otros.

Figura95. Código mapas.



En la parte de autorización es donde al momento de ejecutar la aplicación se lanza un mensaje diciendo si autoriza o permite activar la localización en el dispositivo móvil. En la Figura96 se muestra los dos tipos de mapas que se pueden utilizar para visualizar el punto de coordenada. En la Figura97 se observa la región, lo cuál permite encontrar la ubicación de la persona. Posteriormente el *userLocation* permite localizar las coordenadas correspondientes con un zoom (2000, 2000), estos números indican que tan cerca se verá el punto de localización.

[77]RAYWENDERLICH. MapKit Tutorial: Getting Started. [online]. Disponible en: <http://www.raywenderlich.com/90971/introduction-mapkit-swift-tutorial>

Figura96. Tipo de mapa.

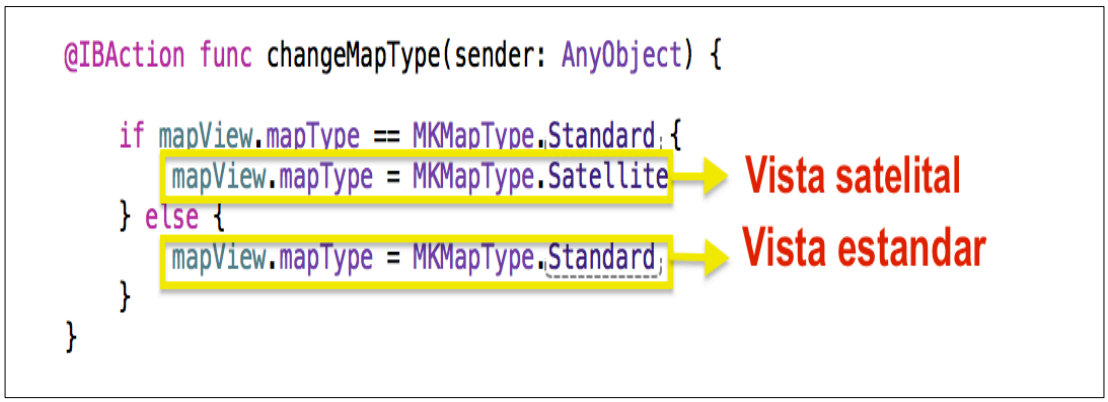
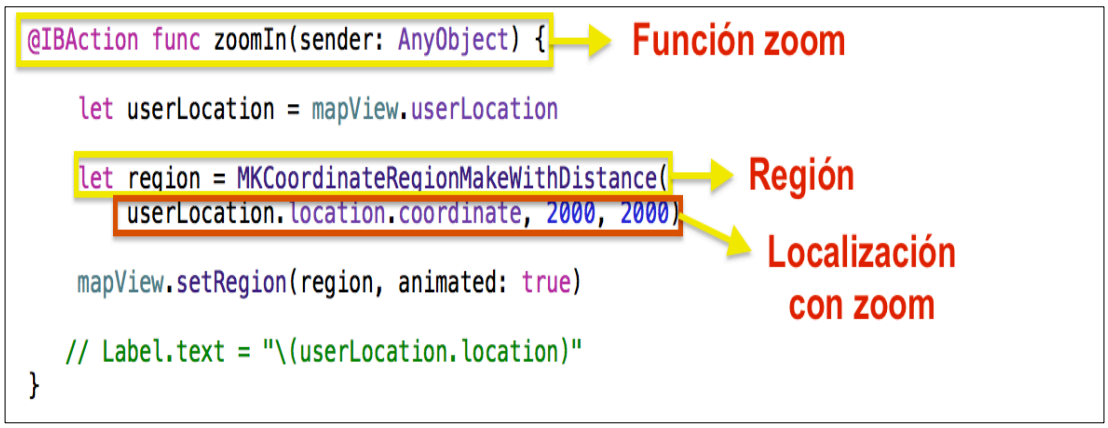


Figura97. Ubicación mapa.



4.2.2 Varias vistas

Cuando se usa la clase *Single View Application* sólo se puede manejar la aplicación como una sola vista, para vincular más vistas (llamadas *View Controller*) es necesario añadir a la pantalla de trabajo gráfica o *Main.storyboard* una herramienta llamada *View Controller*, así como se muestra en la Figura98. Se arrastra al entorno de trabajo así como se ha hecho con las herramientas anteriores. Ahora el nuevo entorno queda como se observa en la Figura99.

Figura98. Nuevo View Controller.

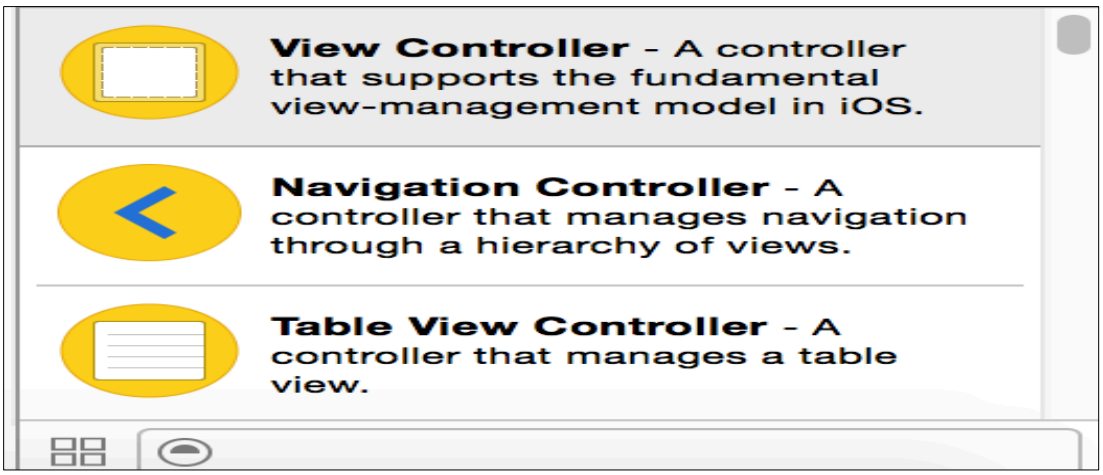
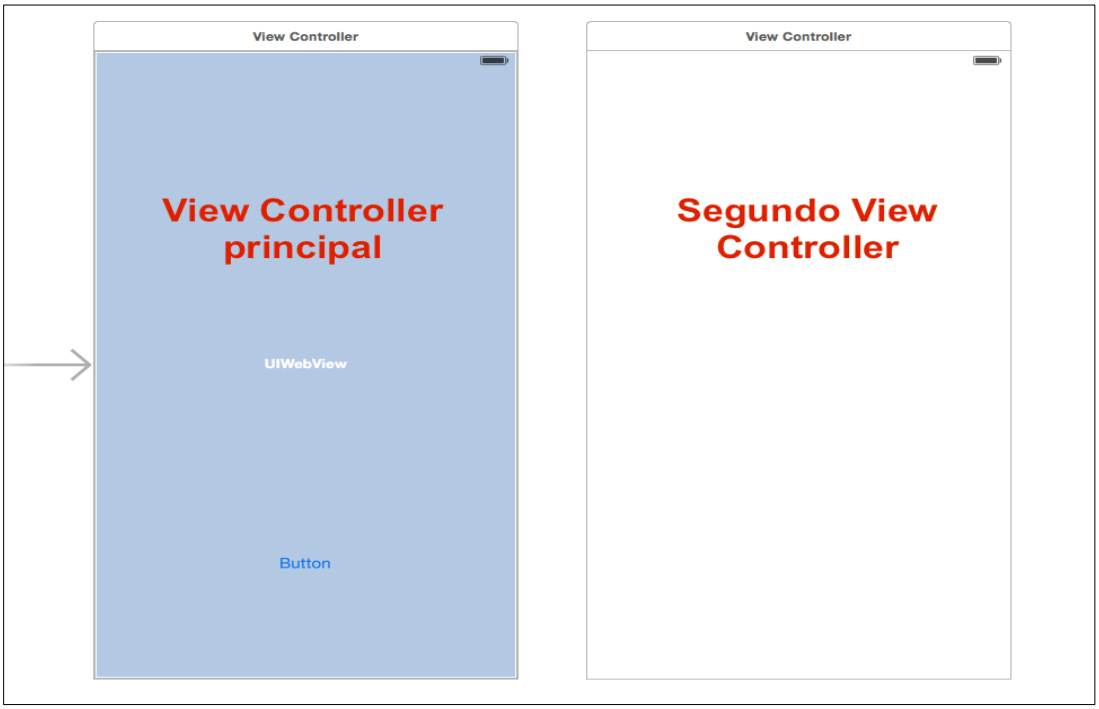


Figura99. Segundo View Controller.



Para poder utilizar la nueva vista, es necesario crear una *carpeta.swift* como la del *ViewController.swift,* En la Figura100 se observa el procedimiento que se debe seguir. Una vez asignado, se le ingresa el mismo nombre al nuevo *View*

Controller, esto es importante para que ambos queden enlazados. En la Figura101 se pude ver la asignación del nombre al nuevo *View Controller.*

Figura100. Asignar nombre al View Controller.

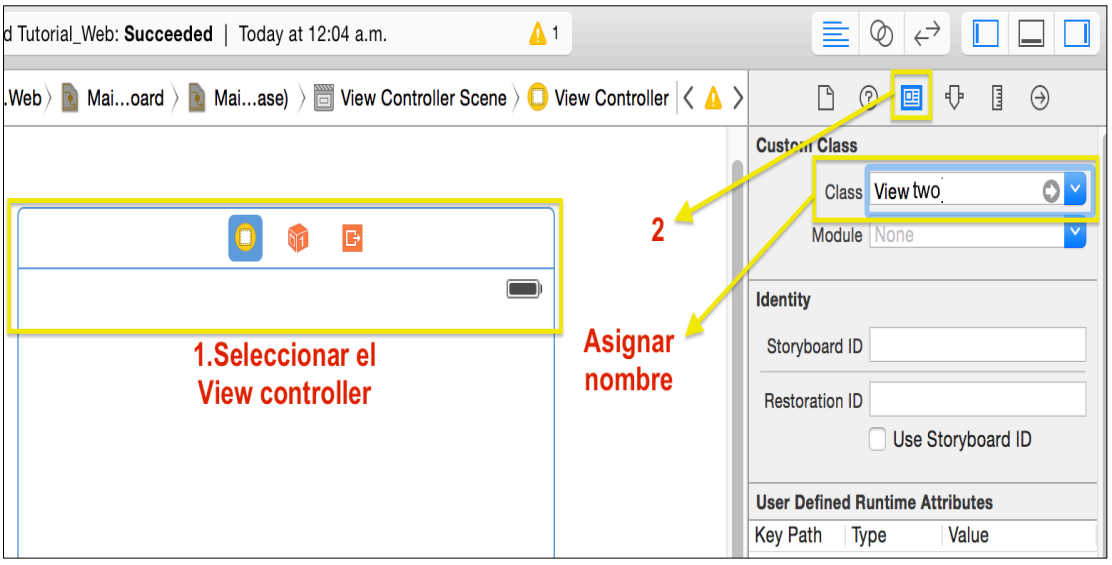
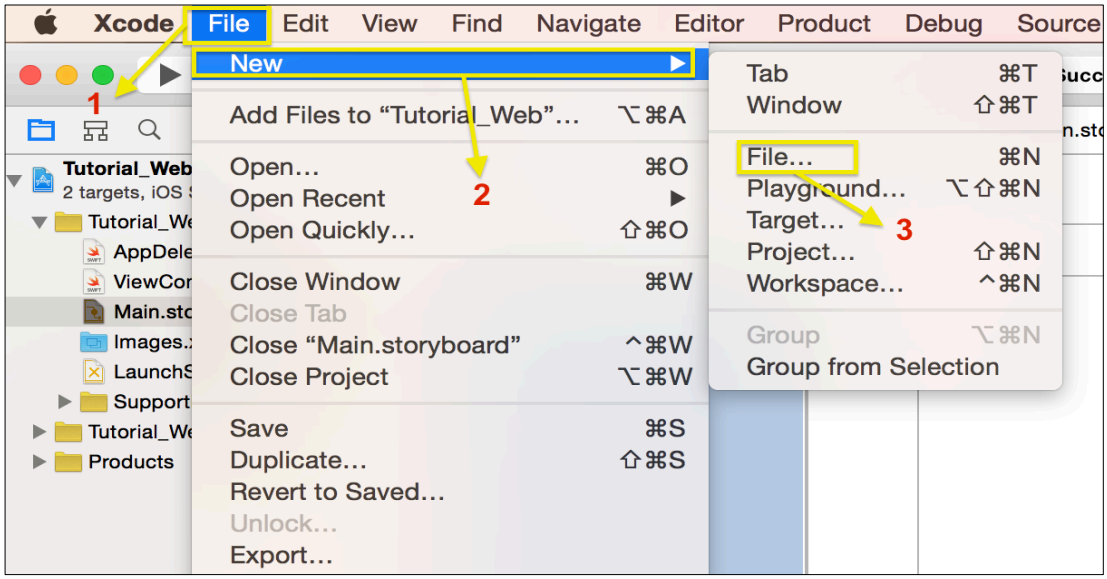


Figura101. Viewtwo.swift



Luego aparece la ventana como se ve en la Figura102. Se selecciona *Swift File* y se le asigna el nombre Viewtwo. En la parte superior izquierda, se elije

Viewtwo.swfit, aparecerá el entorno para programar como se muestra en la Figura103.

Una vez relazado esto, se importa la librería *UIKit,* la cual permite la inicialización de la interfaz gráfica, adicionalmente, se llama la clase *Viewtwo* para empezar a escribir el programa dentro de este (ver Figura104).

Figura102. Swift File.

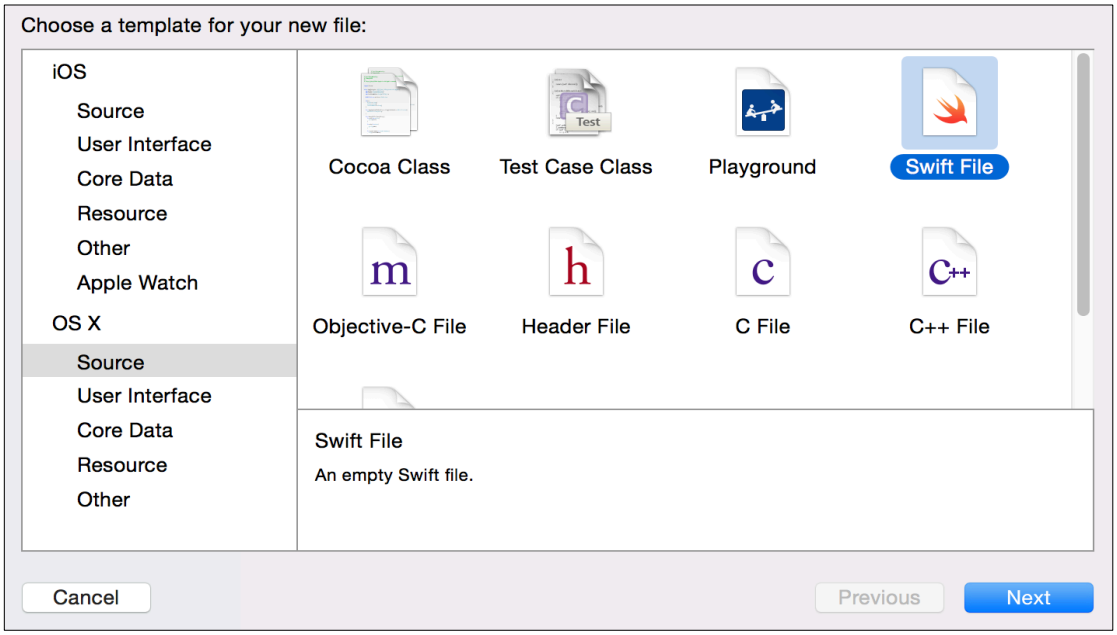


Figura103. Entorno para programar segundo *View Controller*.

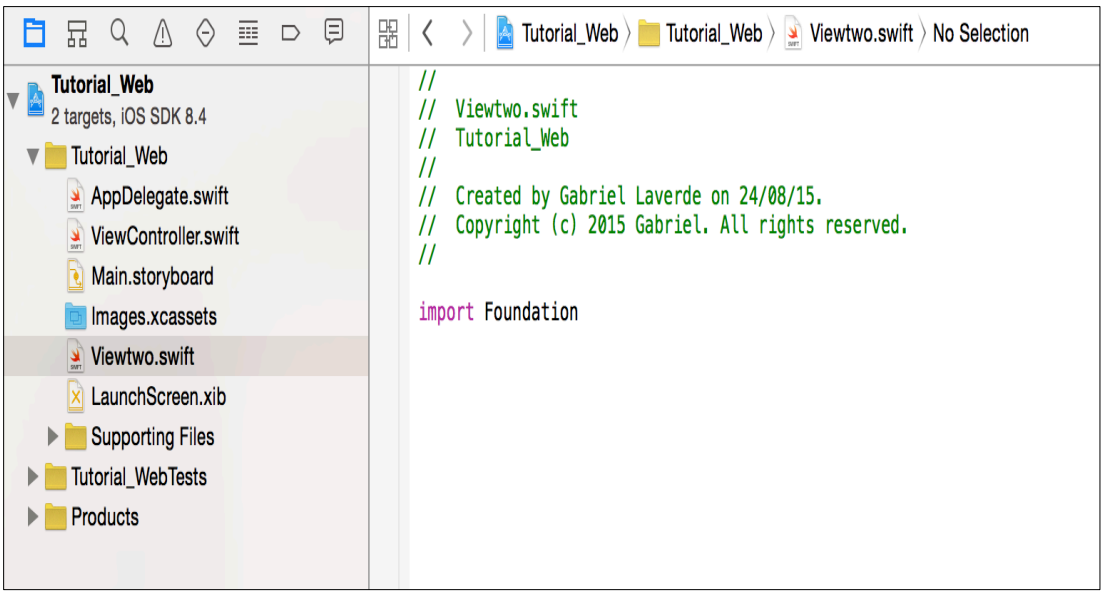
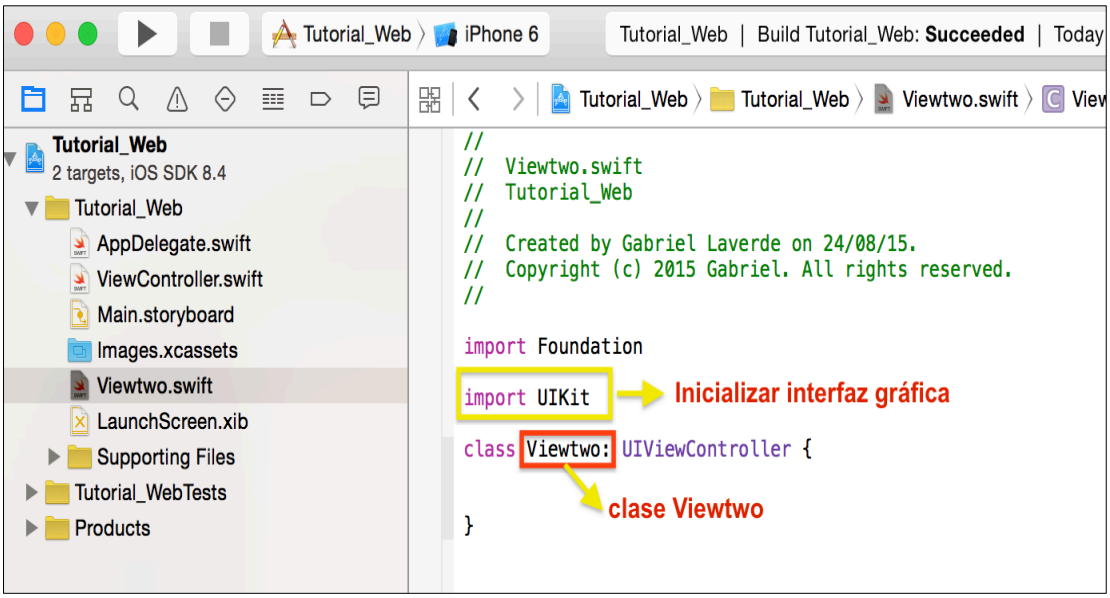


Figura104. Importando librería y clase a Viewtwo.



Ahora la aplicación posee dos vistas diferentes las cuáles se pueden programar a gusto. Se aclara que los elementos que se asignen a cada controlador de vista, debe ser enlazado con la respectiva carpeta.swift, ya que *Xcode* no permite la combinación de un controlador de vista con una carpeta.swift de otro controlador.

En la Figura105 se observa la manera de mezclar controladores de vista con carpetas.swift, pero solo para mostrar textos, o realizar acciones de botones, más no para enlazar herramientas de un *View Controller* a carpetas.swift diferentes.

Figura105. Mezclar controladores de vista.

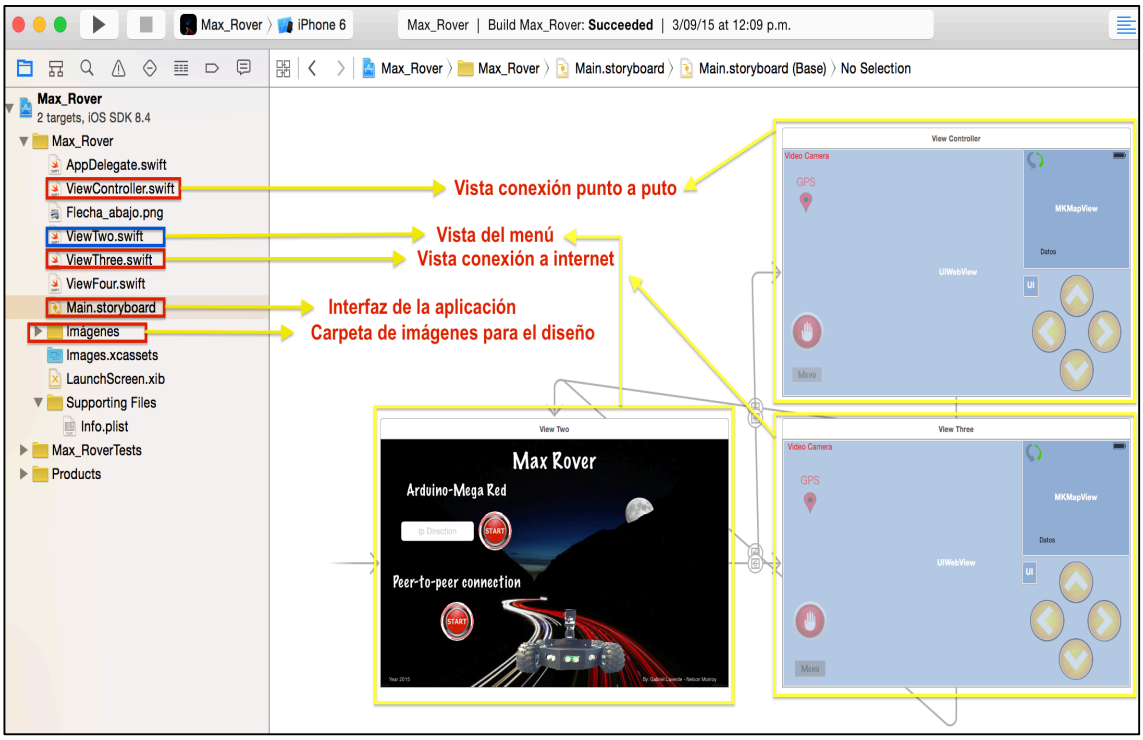
```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {

    if (segue.identifier == "ButtonIp") {

        var DestViewController = segue.destinationViewController as! ViewThree;

        DestViewController.https = FieldIP.text

    }


}
```

La función *prepareForSegue* es la que permite la conexión entre *View Controller*, a continuación se pregunta por el nombre de un botón (para este ejemplo), si se presiona realiza la acción la cuál tiene como propósito asignarle una variable al destino a dónde se quiera ir, en este caso al *ViewThree*(*ViewThree* porque se asignó otra vista más al proyecto de la aplicación) y se realiza la ejecución que sería mostrar un dato en un label, para este ejemplo llamado *https*. El dato que se muestra en el *https* es un dato ingresado por el usuario en un *Text Field* que se encuentra ubicado en el ViewTwo.

4.2.3 Aplicación final

Las partes que se utilizó para crear la aplicación se pueden ver en la Figura106. La vista principal tiene el nombre de *"View Two",* esta vista contiene las opciones para la conexión punto a punto y la conexión via internet local. La vista llamada *"View Controller",* es la enlazada a la conexión punto a punto, esta vista ofrece la visualización del video de la cámara de la Raspberry Pi, la posición del usuario en el mapa satelital y los botones para controlar la plataforma robótica. La vista con nombre *"View Three*", es la de conexión via internet local, esta vista ofrece las mismas opciones que la vista mencionada anteriormente.

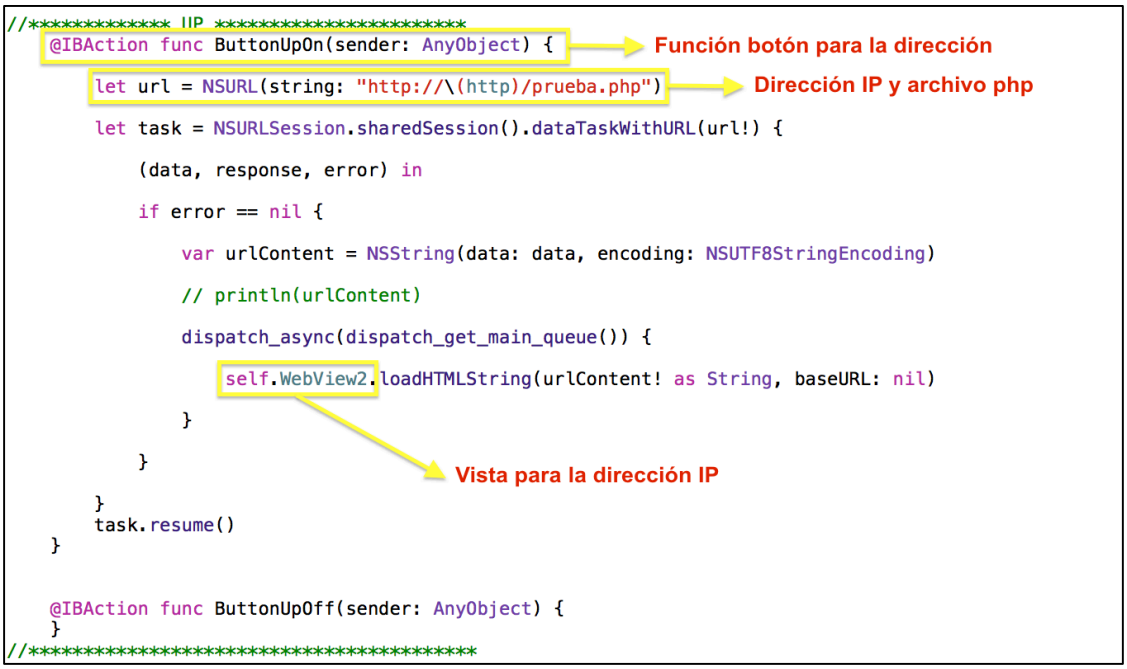
Figura106. Partes que conforman la aplicación.



El código que se implementa para contolar la plataforma robótica por medio de los botones de dirección en la aplicación se puede ver en la Figura107. Esta función se encarga de realizar la acción que se encuentra dentro de los corchetes; por consiguiente se observa la parte que dice “let url”, esta indica la dirección IP por la cual se va a enviar la acción del botón, que para este caso es “*Touch Up Inside*” que ejecuta la acción después de soltar el botón, de seguido se escribe el nombre del archivo php lo cual, este archivo llama un código en python que se encuentra en la Raspberry Pi y contiene el comando que se envía al Arduino Mega por medio del cable USB y así, lograr ejecutar la acción de control que se ordenó por medio del botón de la plaicación. Para que el comando de la dirección IP funcione, se debe mostrar en un navegador web, para este caso se visualiza en “*WebView2*”, esta parte es muy importante ya que si la dirección IP no se le asigna un destino para visualizar la página, simplemente el envio no se realiza y no carga ninguna dirección.

Las otras acciones de control fucioanan de la misma manera, solo que el nombre que contine el php cambia, como por ejemplo arriba.php, derecha.php, abajo.php entre otros.

Figura107. Código del botón para la dirección.



Para visulizar la cámara se hace el mismo procedimeinto que para los botones, solo que en este caso ya no se carga a un archivo php sino a un html, esto se debe a que el sistema operativo IOS no tiene la capacidad de visualizar video por medio de un archivo php, entonces se ve la necesidad de hacerlo por medio de html. El video se visualiza en un navegador web diferente a la de los botones, llamado "*WebView*", esto es necesario ya que este navegador sí va mostrar una imagen que va ser el video captado por la cámara de la Raspberry Pi, mientras que en la de los botones no se visualizaba nada, solo se ejecutaba una acción. Este codigó para la visualización de la cámara se puede ver en la Figura108.

Figura108. Código para visualizar el video captado por la Cámara de la Raspberry Pi.

```
    DireccionIP.text = http

    let url = NSURL(string: "http://\(http)/cam.html")

    let task = NSURLSession.sharedSession().dataTaskWithURL(url!) {
        (data, response, error) in

        if error == nil {

            var urlContent = NSString(data: data, encoding: NSUTF8StringEncoding)

            // println(urlContent)

            dispatch_async(dispatch_get_main_queue()) {

                self.WebView.loadHTMLString(urlContent! as String, baseURL: nil)

            }

        }

    }

    task.resume()

}
```

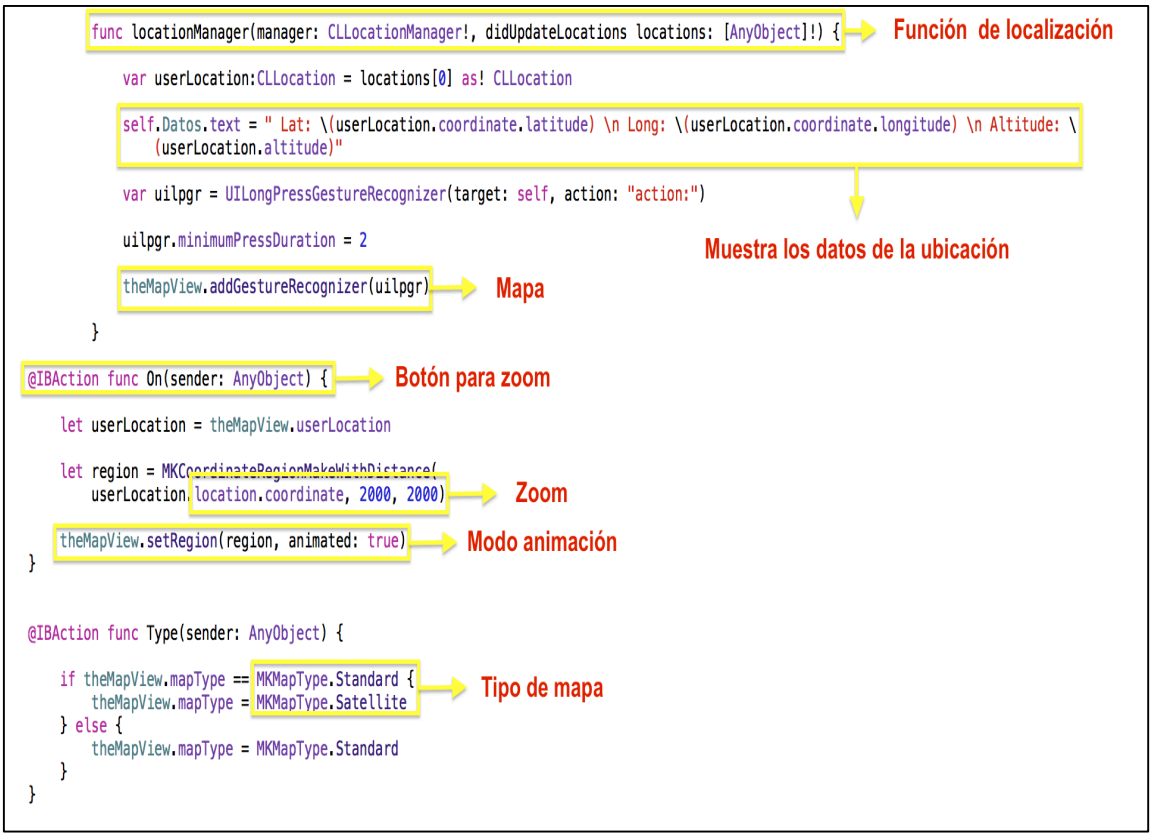
Dirección de la cámara en html

Vista para visualizar el video

Ahora solo hace falta visualizar la localización del usuario en un mapa satelital de la aplicación. En la Figura109 se puede observar el código implementado. Primero se declara una función la cual permite obtener todos los datos necesarios para obtener la ubicación, esta función es "locationManager". Para visualizar los datos de latitud, longitud y altitud, es necesario usar los comandos "*userLocation.coordinate.latitud*", "*userLocation.coordinate.longitude*", "*userLocation.altitude*", *userLocation* es una varible creada por el usuario. Los datos anteriores se muestran en un *Label.*

Para ver el punto de ubicación en el mapa, es necesrio utilizar "*theMapView*", que fue el nombre que se le asigno a la vista *MapKit.* Posteriormente es necesario acercar la vista del punto ya que no se ve exactamente en que parte del mapa se encuentra ubicado, para esto se hace uso de un botón que permita realizar un zoom considerable. Los números 2000, 2000 es lo que se va acercar al punto, entre menor sea el número, mas cerca se verá. Si se quiere ver en forma animada durante el acercameinto, soo basta con usar el comando "*animated: true*" como muestra la Figura109. Por último se muestra otra función de botón, lo cual permite al usuario escoger el mapa en el que quiera ver su ubicación, ya sea modo satelital o estandar.

Figura109. Código para la localización del usuario.

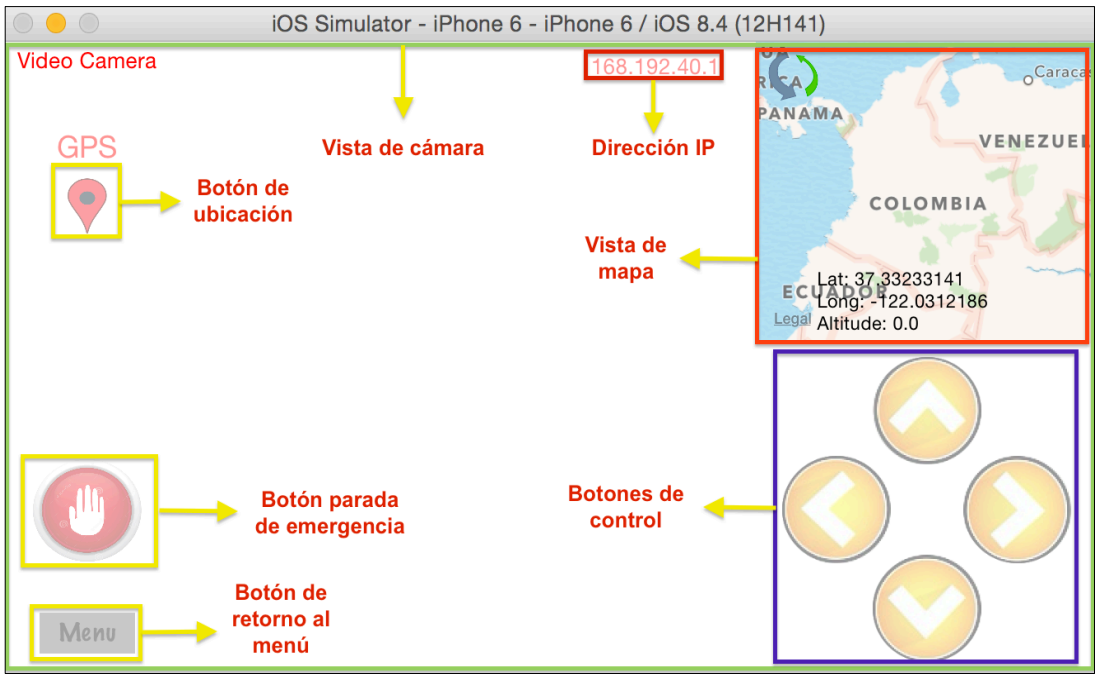


La aplicación final se basa en la compilación de varias vistas. Como se explicó anteriormente esta consta de dos partes principales, la primera tiene que ver con una conexión vía internet local, la cuál se realiza por medio de una IP que el usuario debe ingresar por medio del teclado en la aplicación, esta dirección es la IP de la red a la que se encuentra conectado el dispositivo móvil y la Raspberry Pi. En la Figura110 se muestra la interfaz gráfica de la aplicación que controla la plataforma robótica.

Figura110. Diseño de aplicación.

La vista que se abre al momento de establecer la conexión vía internet local (ver Figura111), tiene varias herramientas, la cuál ofrece la visualización de la cámara en tiempo real, los botones para controlar la plataforma robótica, botón para mostrar la posición del usuario en el mapa, botón para parada de emergencia, visualización de la IP a la que se esta conectado, botón de regreso al menú principal.

Figura111. Vista de control.



El simulador que ofrece *Xcode*, no siempre permite hacer la localización, provocando un cierre inesperado de la aplicación, para esto se debe probar directamente desde el dispositivo móvil como se muestra en la Figura112. En este momento el dispositivo móvil se encuentra localizado en la Universidad Pontificia Bolivariana seccional Bucaramanga, el botón que está dentro de la vista del mapa, ubicado en la esquina superior izquierda permite hacer el cambio de modo estándar a satelital y viceversa (ver Figura113).

Figura112. Ubicación del dispositivo móvil.

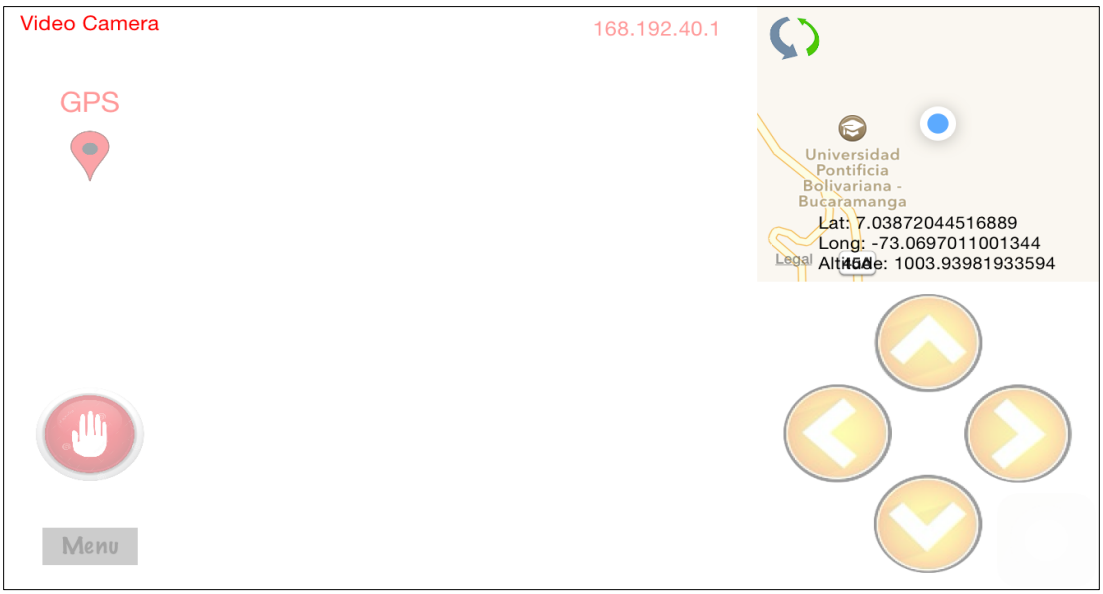


Figura113. Ubicación modo satelital.

Para la conexión punto a punto ya se establece una dirección IP fija creada por la Raspberry Pi, al momento de enlazar se abre una vista exactamente igual a la vista anterior de la conexión vía internet local, la dirección IP fija es “192.168.42.1”.

La plataforma robótica es controlada por medio de los 4 botones. Estos botones están configurados como *Touch Up Inside,* lo cuál consiste en ejecutar la acción cuando se deja de presionar el botón. La acción que se genera se debe a la programación que se implementó en Python. El comando que envía la acción del botón a la Raspberry Pi, se hace por medio de una dirección IP, por lo tanto, esa dirección IP carga un archivo PHP en la Raspberry Pi la cuál contiene un código escrito en Python, este archivo en Python es el encargado de comunicar las Raspberry con el Arduino e indicarle el tipo de acción que debe realizar la plataforma robótica.

En el entorno de Arduino se encuentra una pestaña donde está la programación para el control de los motores, como se puede observar en la Figura114. Para este caso se muestra cuando los motores van hacia delante; este código es el que se ejecuta en el Arduino Mega cuando el botón que ilustra la flecha hacia arriba de la aplicación se le da un toque, ya que el botón envía a la dirección IP la acción que se acaba de realizar y así el PHP se encarga de direccionar al código en python donde está el llamado de acción para comunicar las Raspberry Pi al Arduino Mega por medio del cable USB y así lograr el desplazamiento de la plataforma robótica hacia adelante.

Figura114. Dirección de los motores en el Arduino Mega.

ROVER10 § | GPS § | control | motores | ultrasonicos

```
void motoresadelante()
{
//motor 1 direccion
 pinMode(motor1a,OUTPUT);
 digitalWrite(motor1a,HIGH);
 pinMode(motor1b,OUTPUT);
 digitalWrite(motor1b,LOW);
//motor 2 direccion
 pinMode(motor2b,OUTPUT);
 digitalWrite(motor2b,HIGH);
 pinMode(motor2a,OUTPUT);
digitalWrite(motor2a,LOW);
//motor 3 direccion
 pinMode(motor3b,OUTPUT);
 digitalWrite(motor3b,HIGH);
 pinMode(motor3a,OUTPUT);
 digitalWrite(motor3a,LOW);
//motor 4 direccion
 pinMode(motor4b,OUTPUT);
 digitalWrite(motor4b,LOW);
 pinMode(motor4a,OUTPUT);
 digitalWrite(motor4a,HIGH);
}
```

Las otras direcciones de la plataforma como atrás, derecha, izquierda, se realizan de la misma manera como la de “motoresadelante” que se muestra en Figura114, solo se debe tener en cuenta que pines de los motores se le asignan *LOW* o *HIGH* dependiendo de la dirección que se le quiera dar al motor en la programacion de Arduino, se aclara que para cada acción de botón se ejecuta un comando diferente en python.

Además en la aplicción se muestra un botón de fondo rojo con una mano blanca (ver Figura113), esto indica una parada de emergencia para la plataforma robótica ya que es necesario detener los motores en cualquier situación de riesgo, este comando funciona exactamente igual al comando de “motoresadelante” como se mencionó anteriormente.

5 CONCLUSIONES

Teniendo en cuenta los objetivos propuestos en este proyecto, se concluye satisfactoreamente el diseño del controlador de velocidad de los motores de la plataforma robotica, teniendo resultados muy similares a los esperados, además se pudo desarrollar que se maneja esta plataforma por medio de dispostivos moviles utilizando sistemas operativos como IOS de la empresa Apple y Android de la empresa Google.

Se logró capturar el video por la cámara de la Raspberry Pi, este video se visualizó en la aplicación móvil diseñada para los sistemas operativos. La señal de video se visualiza tanto para la conexión punto a punto como por via internet. Por medio del video permite observar el recorrido de la plataforma robótica sin tenerla a la vista.

Con este proyecto se busco fortalecer el desarrollo de aplicaciones para dispositivos móviles, ya que es un área que está creciendo considerablemente día día, y más usuarios portan un *Smartphone* en sus bolsillos.

Exponencialmente crece en el mercado las aplicaciones en los dispositivos móviles, la cual pueden estar sujetas para el desarrollo en las ramas de ingeniería como la robótica, las telecomunicaciones, Bioingeniería entre otros. Hasta el momento en la Universidad Pontificia Bolivariana Seccional Bucaramanga solo se ha desarrollado un sistema robótico basado en aplicaciones para dispositivos móviles.

Se logró comunicar satisfactoriamente el sistema de computación embebido (Raspberry Pi) con el Arduino Mega por medio del cable USB, lo cual el primero se encargaba de la recepción de todos los datos, enlace, monitoreo y el otro de aplicar las acciones de control como el control PID y la acción de los sensores de ultrasonidos.

6 RECOMENDACIONES Y TRABAJOS FUTUROS

Para trabajos futuros se plantea implementar un tratamiento de imágenes en la tarjeta de computación embebida para reconocer sucesos tales como la presencia humana, obstáculos, animales u otros objetos, adicionalmente desarrollar un control multivariable en el microcontrolador para obtener velocidad y torque constante.

Desarrollar la aplicación de control en otros sistemas operativos como Windows Phone y un servidor alojado en la web, posteriormente implementar un sistema de comunicación inalámbrica de mayor alcance entre el sistema robótico y el dispositivo móvil por medio de la conexión punto a punto.

7 BIBLIOGRAFIA

APPLE. IOS Developer Library – Prerelease. Agosto 24, 2015.

APR. Tipos de variables en PHP. Declaración y asignación. Sentencia echo: insertar texto en el HTML (CU00816B). Septiembre 1, 2015. [online]. Disponible en: <http://www.aprenderaprogramar.com/index.php?option=com_content&id=544:tipos-de-variables-en-php-declaracion-y-asignacion-sentencia-echo-insertar-texto-en-el-html-cu00816b&Itemid=193>

BYTEMARK HOSTING. Welcome to Raspbian. Agosto 30, 2015. [online]. Disponible en : < http://raspbian.org/FrontPage >

CONTRINEX. Sensores de ultrasonidos. Agosto 30, 3015. [online]. Disponible en: < http://www.contrinex.com/file-download/myPDFFiles/products/ultra/documentation/es/sensores_ultra-sonicos.pdf >

Dani Korpi, Lauri Anttila, Mikko Valkama, Risto Wichman Taneli Riihonen, Ville Syrja¨la¨. Full-Duplex Transceiver System Calculations: Analysis of ADC and Linearity Challenges. Junio 5, 2013. [online]. Disponible en < http://www.cs.tut.fi/~valkama/full-duplex/DKorpi_TWireless_submit.pdf >

DEBIAN. Package: libapache2-mod-php5 (5.6.12+dfsg-1 and others). Agosto 28, 2015. [online]. Disponible en: < https://packages.debian.org/sid/libapache2-mod-php5>

PÁEZ ARDILA. Diego Ricardo, Diseño e implementación de una plataforma hardware y software para el control de un robot móvil utilizando el iPad, Trabajo de grado para optar a título de Ingeniero Electrónico, Bucaramanga Colombia. Universidad Pontificia Bolivariana Seccional Bucaramanga, 2012, p. 28.

DYNAMO. 4WD2 Plataforma robótica 4x4 configurable. Septiembre 5, 2015. [online]. Disponible en:

<http://www.dynamoelectronics.com/index.php?page=shop.product_details&flypage=dynamo.tpl&product_id=1066&category_id=63&option=com_virtuemart&Itemid=58>

EARTH POINT. Convert Coordinates - Calculate a position in a variety of formats. Septiembre 2, 2015. [online]. Disponible en: < http://www.earthpoint.us/Convert.aspx>

GPS.GOV. Control Segment Elements. Marzo 27, 2015. [online]. Disponible en: < http://www.gps.gov/systems/gps/control/ >
GPS.GOV. Space Segment. Agosto 17, 2015. [online]. Disponible en: < http://www.gps.gov/systems/gps/space/ >

IEEE SPECTRUM. Create a “Wheel of excuse” With BASIC and the New Raspberry Pi. Octubre 24, 2014. [online]. Disponible en: <http://spectrum.ieee.org/geek-life/hands-on/create-a-wheel-of-excuses-with-basic-and-the-new-raspberry-pi>

INSTITUCIÓN NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA. Sistema de Posicionamiento Global (GPS). Agosto 20, 2015. [online]. Disponible en: < http://www.inegi.org.mx/geo/contenidos/geodesia/gps.aspx?dv=c1 >

Javier Smaldone. Introducción a Secure Shell. Enero 20, 2004. [online]. Disponible en: < http://es.tldp.org/Tutoriales/doc-ssh-intro/introduccion_ssh-0.2.pdf>
KODI. Raspberry Pi. Agosto 18, 2015. [online]. Disponible en: <http://kodi.wiki/view/Raspberry_Pi>

KONGSBERTG. Norwegian University of Technology and Science purchases REMUS 100 Mayo 12, 2012. [online]. Disponible en: <http://www.km.kongsberg.com/ks/web/nokbg0238.nsf/AllWeb/B1A85BBB450D17A6C1257A00002CA7BD?OpenDocument>

GUERRERO PABON, Hugo. Diseño de un robot explorador para tuberias de alcantarillado con camara integrada. Proyecto de trabajo de grado para optar por titulo de Ingeniero Mecatrónico. Pamplona Colombia. Universidad de Pamplona 2007. 5.

CHÁVEZ GONZÁLEZ ,Manuel Alberto. Prototipo de Robot Móvil Teleoperado. Tesis que para obtener el grado de Maestría en Tecnología Avanzada. Querétaro. Instituto Politécnico Nacional. 2012. I.

MATHWORKS. Hardware Support. [online]. Disponible en : <http://www.mathworks.com/hardware-support/arduino-matlab.html>

MONK, Simon. Programming the Raspberry Pi. Getting Started with Python. 1 ed. McGraw-Hill. 170 p. ISBN 978-0071807838.
Motion. Motion - Web Home. Julio 30, 2015. [online]. Disponible en: < http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome>

OMNIVISION. 5-megapixel ¼" Image Sensor with 1.4 Omni BSI Technology Offering HD Video. Enero, 2010. [online]. Disponible en: < http://www.ovt.com/uploads/parts/OV5647.pdf>
PIDORA 2014. Raspberry Pi Fedora 2014. [online]. Disponible en: < http://pidora.ca/pidora/releases/20/release-announcement.txt >

RASPBERRYCONNECT. Raspberry Pi NoIR Infrared camera module. Noviembre 2, 2013. [online]. Disponible en: <http://www.raspberryconnect.com/hardware-add-ons/item/148-raspberry-pi-noir-infrared-camera-module>

RASPBERRY PI FOUNDATION. APT. Septiembre 1, 2015. [online]. Disponible en : < https://www.raspberrypi.org/documentation/linux/software/apt.md >

RASPBERRY PI FOUNDATION. PI NOIR CAMERA. Septiembre 1, 2015. [online]. Disponible en: < https://www.raspberrypi.org/products/pi-noir-camera/>

RASPBERRY PI FOUNDATION. Python. Septiembre 1, 2015. [online]. Disponible en: <https://www.raspberrypi.org/documentation/usage/python/README.md>

RASPBERRY PI FOUNDATION. PYTHON PICAMERA. Septiembre 2, 2015. [online]. Disponible en: < https://www.raspberrypi.org/documentation/usage/camera/python/README.md >

RASPBERRY PI FOUNDATION. RASPICAM COMMAND. Septiembre 1, 2015. [online]. Disponible en <https://www.raspberrypi.org/documentation/usage/camera/raspicam/README.md >

RASPBERRY PI FOUNDATION. Remote Access. Septiembre 3, 2015. [online]. Disponible en: < https://www.raspberrypi.org/documentation/remote-access/README.md >

RASPBERRY PI FOUNDATION. what is the camera board?. Septiembre 1, 2015. [online]. Disponible en: < https://www.raspberrypi.org/help/faqs/#camera>

SPARKFUN. SparkFun Venus GPS with SMA Connector. Septiembre 4, 2015. [online]. Disponible en: < https://www.sparkfun.com/products/11058>

THE APACHE SOFTWARE FOUNDATION. About Apache. Agosto 25, 2015. [online]. Disponible en: <http://www.apache.org/history/timeline.html>

THE APACHE SOFTWARE FOUNDATION. Módulos de MultiProcesamiento (MPMs). Agosto 29, 2015. [online]. Disponible en: < http://httpd.apache.org/docs/2.0/mpm.html >

THE PHP GROUP. ¿Qué es php?. Septiembre 2, 2015. [online]. Disponible en: <https://secure.php.net/manual/es/intro-whatis.php>

THE PHP GROUP. ¿Qué pude hacer php?. Septiembre 1, 2015. [online]. Disponible en: <https://secure.php.net/manual/es/intro-whatcando.php>

THE PHP GROUP. Su primera página con php. Septiembre 1, 2015. [online]. Disponible en: <https://secure.php.net/manual/es/tutorial.firstpage.php>

UBUNTUUPDATES.ORG. Package "libjpeg62". Agosto 20, 2015. [online]. Disponible en: < http://www.ubuntuupdates.org/package/core/vivid/main/base/libjpeg62>

XATAKAON. NAT (Network Address Translation): Qué es y cómo funciona. Agosto 23, 2015. [online]. Disponible en: < http://www.xatakaon.com/tecnologia-de-redes/nat-network-address-translation-que-es-y-como-funciona>