



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

“APLICACIONES CON MINICOMPUTADORES RASPBERRY PI PROVISTO DE MÓDULO GPS Y ACELERÓMETRO PARA CONTROL DE VELOCIDAD Y POSICIONAMIENTO”

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO ELÉCTRICO ESPECIALIZACIÓN AUTOMATIZACIÓN

INDUSTRIAL

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Fernando Rainer Chávez Castrillón

Christian Gonzalo Yuquilema León

GUAYAQUIL – ECUADOR

AÑO 2013

AGRADECIMIENTOS

Principalmente a Dios por haberme dado fuerzas y por guiarme en los momentos más difíciles.

A mis padres por haberme apoyado en todo momento y ser los pilares fundamentales para conseguir este logro tan importante.

CHRISTIAN YUQUILEMA LEON

En primer lugar deseo agradecer a Dios, a mis padres por haber apoyado y guiado durante toda mi vida y a mi esposa porque ha sido mi eterna compañera.

Deseo igualmente realizar un agradecimiento muy especial al Ing. Carlos Valdivieso por ser mentor para el desarrollo de esta tesina.

FERNANDO CHÁVEZ CASTRILLÓN

DEDICATORIA

A Dios por ser nuestro creador, amparo y fortaleza, cuando más lo necesitamos, y por hacer palpable su amor a través de cada uno de los que nos rodea.

CHRISTIAN YUQUILEMA LEON

A mis padres por haber estado junto a mí en todo momento, a mi esposa que me ha compartido conmigo los momentos más felices de mi vida y a mis hijas que son el motivo para superarme día a día.

FERNANDO CHÁVEZ CASTRILLÓN

TRIBUNAL DE SUSTENTACIÓN



Ing. Carlos Valdivieso A.

PROFESOR DEL SEMINARIO DE GRADUACIÓN



Ing. Hugo Villavicencio V.

PROFESOR DELEGADO POR LA UNIDAD ACADÉMICA

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesina de Grado, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de exámenes y títulos profesionales de la ESPOL)



Yuquilema León Christian Gonzalo



Chávez Castrillón Fernando Rainer

RESUMEN

El presente trabajo de investigación pretende implementar las técnicas aprendidas en nuestra vida académica para realizar el control de velocidad y posicionamiento de un móvil mediante el uso del minicomputador moderno Raspberry Pi, el cual está provisto de un módulo GPS y acelerómetro para medir las variables antes señaladas.

Puntal fundamental para el desarrollo de esta tesina es la programación del minicomputador Raspberry Pi, que mediante la utilización del lenguaje de programación llamado Python, se ha logrado recibir y procesar los datos enviados desde el módulo GPS y acelerómetro.

La adquisición de datos desde el Sistema de Posicionamiento Global, hacia la mini computadora Raspberry Pi se lo ha realizado a través del puerto GPIO, el cual es un puerto de entrada y salida de propósitos generales. Este puerto puede generar señales digitales, señales PWM, comunicación serial SPI, RS232, entre otras. La adaptación del módulo GPS hacia el minicomputador se lo ha implementado mediante una tarjeta puente. Cabe mencionar que los datos enviados por el módulo GPS cumplen con el protocolo NMEA 0183.

Finalmente el proyecto desarrollado ha sido probado con éxito tanto con un simulador de GPS FURUNO GP-32, así como también de forma real mediante la instalación de este dispositivo en un vehículo motorizado.

ÍNDICE GENERAL

AGRADECIMIENTOS	I
DEDICATORIA.....	II
TRIBUNAL DE SUSTENTACIÓN	III
DECLARACIÓN EXPRESA	IV
RESUMEN	V
ÍNDICE GENERAL.....	VII
ÍNDICE DE FIGURAS	X
ÍNDICE DE TABLAS	XIII
CAPÍTULO 1	1
1 DESCRIPCIÓN GENERAL DEL PROYECTO	1
1.1 Antecedentes	1
1.2 Motivación.....	4
1.3 Identificación del problema.....	6
1.4 Objetivos Principales.....	6
1.5. Limitaciones	7

CAPÍTULO 2	8
2 FUNDAMENTO TEÓRICO.....	9
2.1. Minicomputador Raspberry – Pi.....	10
2.1.1 Configuración Del Raspberry Pi para Recepción de Data desde el módulo GPS.....	11
2.1.1 Instrucción en Python para programación del GPIO y UART.....	15
2.2 Protocolo de Comunicaciones NMEA 0183.	18
2.2.1 Interface Eléctrica del Protocolo.....	18
2.2.2 Formato de la Estructura del Protocolo.....	19
2.3 Descripción de los Sistemas de Posicionamiento Satelital.	22
2.4 Descripción del Firmware del Módulo GPS CSC3 Vincotech.....	24
2.4.1 Mensajes de Inicialización del GPS.	24
2.4.2 Sentencias propietarias del Módulo GPS.....	25
2.4.3 Configuración del Puerto Serial del Módulo GPS.....	26
2.4.4 Selección de las Sentencias que enviará el Módulo GPS.....	26
2.5 Descripción general de un acelerómetro.....	28
CAPÍTULO 3	31
3 DISEÑO E IMPLEMENTACIÓN DEL PROYECTO.....	32
3.1 Conexión de Componentes.....	32

3.2 Configuración del Raspberry para conexión con Módulo GPS y Simulador.....	35
3.3 Instalación de Software para lectura de data del GPS.....	38
3.4 Instalación del Software GPS Daemon (gpsd).....	42
3.5 Adquisición de datos desde el módulo GPS utilizando Python	44
CAPÍTULO 4	53
4 PRUEBAS Y SIMULACIONES.....	54
4.1 Procedimientos de Encendido e Inicialización.	54
4.2 Pruebas realizadas con el Simulador GPS.	56
4.3 Pruebas realizadas con el módulo GPS en un vehículo.	62
CONCLUSIONES	70
RECOMENDACIONES	73
BIBLIOGRAFÍA	74

ÍNDICE DE FIGURAS

Figura No.2.1: Tarjeta Minicomputador Raspberry Pi Modelo B.	11
Figura No. 2.2: Diagrama en Bloques de la Tarjeta Minicomputador Raspberry Pi Modelo B.....	11
Figura No. 2.3: Distribución de los Pines del Puerto GPIO del Raspberry Pi Modelo B.....	13
Figura No. 2.4: Ubicación del archivo cmdline.txt.	15
Figura No. 3.1: Determinación del Funcionamiento del Módulo mediante un Osciloscopio.....	33
Figura No 3.2. Diagrama Básico de Interconexión.....	33
Figura No 3.3. Interconexión Física de Componentes.	34
Figura No 3.4. Pruebas de Recepción con Vidrio de 5mm de Espesor.	35
Figura No 3.5 . Edición del archivo cmdline.txt	36
Figura No 3.6. Edición del archivo Initab.txt.....	37
Figura No 3.7 . Resultado del Comando ls en Consola	38
Figura No 3.8. Resultado del Comando “cat” en el Terminal	38
Figura No 3.9. Configuración del Software Minicom	39

Figura No 3.10: Datos visualizado en el Software Minicom.	40
Figura No 3.11: Datos visualizado en el Software Cutecom.	41
Figura No.3.12: Comandos para ejecución del Software GPSD	43
Figura No. 3.13: Visualización de datos en el Software GPSD	44
Figura No 3.14. Algoritmo para la Implementación del Software.	45
Figura No 3.15. Display que Visualiza Ingreso de Data GPGGA	52
Figura No 3.16. Display que Visualiza Ingreso de Data GPRMC.....	53
Figura No. 4.1: Diagrama en Bloques de Conexión del Simulador de GPS....	56
Figura No. 4.2: Interconexión Física del Simulador de GPS.	57
Figura No. 4.3: Pantalla para Selección del Modo de Simulación del GPS. ..	58
Figura No. 4.4 Pantalla para Cambio de Valores de Rumbo, Velocidad y Posición.	58
Figura No. 4.5 Configuración del Software Cutecom para Simulación.....	59
Figura No. 4.6 Visualización de Data Recibida en Software Cutecom.....	60
Figura No. 4.7 Visualización de datos Entregados por el Simulador y Recibidos en Software Python (Ejemplo No.1).	61

Figura No. 4.8 Visualización de datos Entregados por el Simulador y Recibidos en Software Python (Ejemplo No.2).	61
Figura No. 4.9 Conexión del Raspberry-Pi en el vehículo.....	62
Figura No. 4.10 Instalación de antena en el interior del vehículo.....	63
Figura No. 4.11 Instalación de display en el interior del vehículo.....	63
Figura No. 4.12 Adquisición de data del módulo GPS mediante el Software Cutecom.....	64
Figura No. 4.13 Adquisición de datos del módulo GPS mediante el Software GPSD.....	65
Figura No. 4.14 Adquisición de datos del módulo GPS mediante el Software GPSD – Auto Detenido.	66
Figura No. 4.15 Adquisición de datos del módulo GPS mediante el Software GPSD – Auto en Movimiento.	68
Figura No. 4.16 Adquisición de datos del módulo GPS mediante el Software Desarrollado en Python – Auto en Movimiento.	69
Figura No. 4.17 Proyecto Implementado para control de posición y velocidad de un móvil mediante el uso de un Micocomputador Raspberry Pi.	70

ÍNDICE DE TABLAS

Tabla No. 1: Mensajes de Entrada NMEA al Módulo GPS.....	25
Tabla No. 2: Configuración del Puerto Serial.....	26
Tabla No. 3: Formato de la Sentencia de Control.....	27
Tabla No. 4: Valor asignado a cada Mensaje de Control.....	27

INTRODUCCIÓN

Este proyecto se basa en el control de velocidad y posicionamiento de un móvil mediante la utilización de un módulo GPS, el mismo que se encuentra conectado a una minicomputadora Raspberry Pi, para realizar la adquisición, procesamiento y presentación de los datos antes indicados mediante el uso del lenguaje de programación Python.

Para la ejecución de la presente tesina nos hemos planteado el desarrollo de 04 capítulos, los mismos que contienen tanto el sustento teórico como práctico del proyecto implementado. En el primer capítulo se determinan los antecedentes, objetivos, motivación y limitaciones planteadas del presente estudio.

En el segundo capítulo encontraremos el fundamento teórico que permitirá la implementación del control de posición y velocidad. Principalmente se ha enfocado el conocimiento detallado para el manejo de los puertos GPIO y UART, así como también el conocimiento especializado para la configuración del módulo GPS, el mismo que se interconectará con el minicomputador Raspberry PI. En el tercer capítulo se expone el desarrollo e implementación del proyecto en mención, encontrándose en este capítulo los diagramas en bloques, descripción del algoritmo y la descripción del software requerido para la implementación.

Finalmente en el cuarto y último capítulo, se detallan las pruebas y simulaciones realizadas para demostrar que se ha alcanzado el objetivo final del proyecto,

emitiéndose posteriormente las conclusiones sobre los resultados obtenidos y recomendaciones finales del proyecto.

CAPÍTULO 1

1 DESCRIPCIÓN GENERAL DEL PROYECTO

1.1 Antecedentes

El apasionante mundo de la electrónica e informática se ha desarrollado a pasos extraordinarios durante los últimos años, alcanzándose productos de alta integración de componentes, conjugándose el hardware y software para realizar aplicaciones que antes eran alcanzadas mediante la adquisición de costos componentes que usualmente poseían propiedad intelectual de sus fabricantes.

Sin embargo el desarrollo de los códigos abiertos en Linux y la colaboración de miles de desarrolladores desinteresados en el mundo, han permitido que ahora existan aplicaciones extraordinarias que estén al alcance de todos, permitiendo esto, que cualquier aplicación deseada pueda ser implementada de una manera fácil y sobre todo menos costosa.

El reto que nos hemos propuesto para la ejecución del presente proyecto es realizar el control de posición y velocidad de un vehículo mediante la utilización del minicomputador Raspberry Pi, el mismo que interactúa con un módulo GPS para obtener las variables deseadas.

Cabe mencionar que este minicomputador es una tarjeta de bajo costo que fue desarrollada por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de las ciencias de la computación en las escuelas [1]. La idea original era crear un ordenador suficientemente barato como para que los estudiantes pudieran aprender informática con él.

Esta minicomputadora soporta un sistema operativo desarrollado en Linux, lo que facilita la programación y soporte durante el desarrollo, especialmente por la existencia en la web de aplicaciones gratuitas que son de gran utilidad para ejecutar proyectos a un costo relativamente bajo.

En el año 2006 se inició el diseño del minicomputador Raspberry Pi, los cuales se basaron en el microcontrolador Atmel ATmega644. Sus esquemas y el diseño original del circuito impreso estaban disponibles para su descarga pública.

El primer prototipo del Raspberry Pi basado en ARM se instaló en un módulo del mismo tamaño que una memoria USB y tenía un puerto USB en un extremo y un puerto HDMI en el otro.

En agosto del 2011, se fabricaron cincuenta placas Alpha, que tenían las mismas características que el modelo B, pero eran un poco más grandes para poder depurar problemas que existían con sus interfaces.

En diciembre de 2011, 25 placas Beta del modelo B fueron ensambladas y probadas de un total de 100 placas vacías. El diagrama de componentes de las placas finales sería el mismo que el de las placas Beta.

Durante las pruebas a estas placas se encontró un error de diseño en los pines que suministraban alimentación a la CPU, este error fue depurado en su versión final.

Finalmente se realizó una demostración de la placa beta arrancando Linux, reproduciendo un tráiler de una película a 1080p y ejecutando el programa benchmark Rightware Samurai OpenGL ES para la elaboración de gráficos en 2D y 3D.

La principal potencialidad de este minicomputador es su capacidad de comunicación con periféricos a través de sus diferentes puertos y en especial por el puerto de propósitos generales GPIO, que permite realizar aplicaciones para el control y manejo de dispositivos.

Muchos fabricantes han desarrollado en este corto tiempo hardware que puede ser conectado directamente a este minicomputador mediante una tarjeta interfaz como por ejemplo sensores médicos, módulos bluetooth, GPS, GPRS, entre otros, pudiendo ampliar el espectro de aplicaciones si se lo integra con un microcontrolador, el cual permitiría manejar sensores analógicos.

En el caso puntual de este proyecto, se ha considerado la integración de un módulo GPS, que proporciona la información de datos de posición y velocidad. Para facilitar la integración de este módulo ha sido necesaria la incorporación de un puente que facilita la interconexión del módulo GPS con el minicomputador.

Finalmente una vez que se realice la adaptación del hardware, será necesario realizar la adquisición de datos de posición y velocidad y su posterior procesamiento y análisis, para lo cual se ha seleccionado el lenguaje de programación Python, mediante el cual se ha elaborado la respectiva interface hombre máquina, que demostrará el cumplimiento del objetivo planteado.

1.2 Motivación

La principal motivación de este proyecto fue aprender el manejo del minicomputador Raspberry Pi, principalmente por la potencialidad que tiene éste componente, para realizar la comunicación con otros dispositivos a través del puerto de propósitos generales GPIO y mediante el cual se fundamenta la implementación del control de velocidad y posición.

De igual manera el utilizar este dispositivo implicó el aprendizaje de un nuevo lenguaje de programación Python, el cual facilita la interacción con este minicomputador.

Finalmente, luego de realizar este proyecto se abre la expectativa de ampliar los conocimientos sobre las potencialidades del Raspberry Pi, ya que este dispositivo permite además manejar aplicaciones como:

1-Servidor de archivos y de web, ya que se lo puede utilizar para almacenar películas, música, fotos, programas, entre otros.

2-Servidor Streaming

Colocando una webcam, podemos emitir streaming mediante vlc, ffmpeg o flumotion a un servidor remoto o local.

3-Media center

Se puede instalar XBMC y conectarlo a la televisión mediante el HDMI y ver películas en full HD a 1080p.

4-Domótica

Se puede implementar el control automatizados de hogares a un costo relativamente bajo.

5- Aplicaciones con Arduino

Se puede controlar y programar dispositivos robóticos Arduino.

6-PC de escritorio

Se puede agregar una pantalla, un teclado y un ratón y puede trabajar como cualquier ordenador con Linux.

1.3 Identificación del problema

Mediante este proyecto se desea alcanzar el conocimiento necesario para realizar el manejo del puerto de entrada y salida de propósitos generales del minicomputador Raspberry Pi, a fin de realizar el control de posición y velocidad de un móvil, utilizando para el efecto una interface hombre máquina para la presentación de los datos en mención.

Estos datos se obtendrán del módulo GPS y acelerómetro, lo que implica un conocimiento suficiente de este hardware para manejar sus entradas y salidas de datos.

De igual manera se requiere analizar y entender el protocolo de comunicaciones NMEA 0183 que emplea el módulo GPS que se interconectará con el minicomputador.

Finalmente, una vez que se logre controlar y obtener la data desde el GPS, se desarrollará una HMI utilizando el lenguaje de programación Python para presentar los datos de posición y velocidad de un móvil, concretándose de esta manera el objetivo de esta tesina.

1.4 Objetivos Principales

- Realizar el control de posición y velocidad de un móvil, mediante la adquisición de datos desde el módulo GPS y a través del puerto de propósitos generales GPI del minicomputador Raspberry Pi.

- Conocer el funcionamiento del módulo GPS y su protocolo de comunicación a fin de interpretar la información que entrega mencionado módulo para la determinación de la posición y velocidad del móvil en tiempo real.
- Aprender el manejo de la herramienta Cutecom en modo local, el cual permitirá demostrar que la comunicación entre el módulo GPS y el minicomputador se encuentra correctamente establecida.
- Desarrollar capacidades para implementar interfaces Hombre-máquina, mediante la utilización del lenguaje Python, el mismo que permitirá realizar la visualización de los datos que ingresan en tiempo real a nuestro minicomputador, a través del puerto GPIO.

1.5. Limitaciones

Al ser los minicomputadoras Raspberry Pi, placas desarrolladas prácticamente en este último año, existe poca información relacionada con el manejo de su puerto GPIO, debiéndose tomar en cuenta que este proyecto requiere del almacenamiento de datos en tiempo real y se desconoce si esté minicomputador tendrá la capacidad de manejar y procesar la cantidad de datos que ingresarán a través de mencionado puerto.

Se debe recordar además que el principal enfoque de este minicomputador fue el de incentivar el aprendizaje de la programación, razón por la cual este tipo de aplicaciones que requieren procesamiento rápido de señales puedan afectar al rendimiento global de este dispositivo.

Por otro lado es necesario indicar que manejar señales que provienen de dispositivos GPS, los cuales receptan las señales emitidas por los satélites requieren prácticamente que el módulo GPS se encuentren con acceso visual hacia los satélites para poder recibir y entregar información de posición y velocidad del blanco, en caso de no cumplir con esta condición no se podrá realizar ningún tipo de simulación.

CAPÍTULO 2

2 FUNDAMENTO TEÓRICO

En este capítulo realizaremos un estudio sobre el funcionamiento y las características del Minicomputador Raspberry Pi, siendo necesario un conocimiento general sobre su arquitectura (hardware) y de su forma de programación (software).

De igual forma, se analizará el funcionamiento de los Sistemas de Posicionamiento Global (GPS) y en especial del módulo GPS que se conectará al minicomputador para el procesamiento de la data que contiene información de posición y velocidad.

Por otra parte, se analizará el protocolo de comunicaciones en formato NMEA, por cuanto el GPS entrega mediante éste protocolo los datos de navegación y se expondrán los conocimientos básicos de programación en Python para realizar el ambiente gráfico para la presentación de la data obtenida.

Finalmente se analizará el principio de funcionamiento de los acelerómetros, considerándose que este fundamento teórico será suficiente para el desarrollo de la presente tesina y poder cumplir con el objetivo planteado.

2.1. Minicomputador Raspberry – Pi

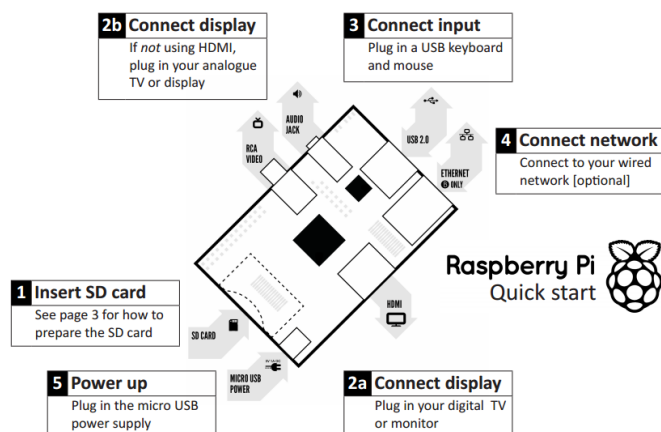
Raspberry-Pi es un minicomputador que posee un procesador central (CPU) ARM1176JZF-S a 700 MHz y una memoria de 512 MB de RAM. Posee 02 puertos USB, 02 salidas de video (RCA y HDMI), salida de audio, conectividad de red mediante el conector RJ-45 y ranura para la instalación de un dispositivo de almacenamiento del tipo SD.

Es una tarjeta de bajo consumo aproximadamente 1 Amp y requiere de un voltaje de alimentación de 5 voltios. Soporta lenguajes operativos tipo Linux. Esto implica que tiene la capacidad de realizar procesamiento gráfico, permitiendo la comunicación con varios tipos de hardware. Se recomienda la instalación del sistema operativo(SO) llamado Raspbian.

En la figura No. 1 se puede apreciar la arquitectura del minicomputador Raspberry Pi, mientras que en la figura No. 2 se muestra la ubicación esquemática de sus principales componentes, donde se puede apreciar las capacidades antes descritas.



Figura No.2.1: Tarjeta Minicomputador Raspberry Pi Modelo B.



1

Figura No. 2.2: Diagrama en Bloques de la Tarjeta Minicomputador Raspberry Pi Modelo B. [3]

2.1.1 Configuración Del Raspberry Pi para Recepción de Data desde el módulo GPS.

Como se indicó anteriormente, para poder acceder a todos los servicios que presenta el Raspberry se requiere la instalación de un sistema operativo Linux, que para el caso de esta aplicación se ha seleccionado el SO Raspbian.

Por otra parte el lenguaje de programación Python que se ha seleccionado para programar el Raspberry nos permitirá manejar el puerto de propósitos generales, el mismo que está compuesto de 26 pines que pueden ser configurados como entradas o salidas.

Para el caso de esta aplicación se conectará a éstos pines la tarjeta interface del módulo GPS, el mismo que nos enviará la información de posición y velocidad.

El puerto GPIO se encuentra ubicado en la parte superior izquierda del integrado, sus 26 pines se encuentran distribuidos en dos columnas de 13 pines cada una. Cada pin tiene su propósito específico el mismo que puede ser visualizado en la figura 2.3.

Como se puede visualizar cada pin tiene su función dentro del puerto. Es muy importante recordar que no se debe conectar ningún dispositivo en los pines que se encuentran etiquetados "Do not Connect". El conectar algún dispositivo a estos pines, podría ocasionar que el minicomputador resulte afectado.

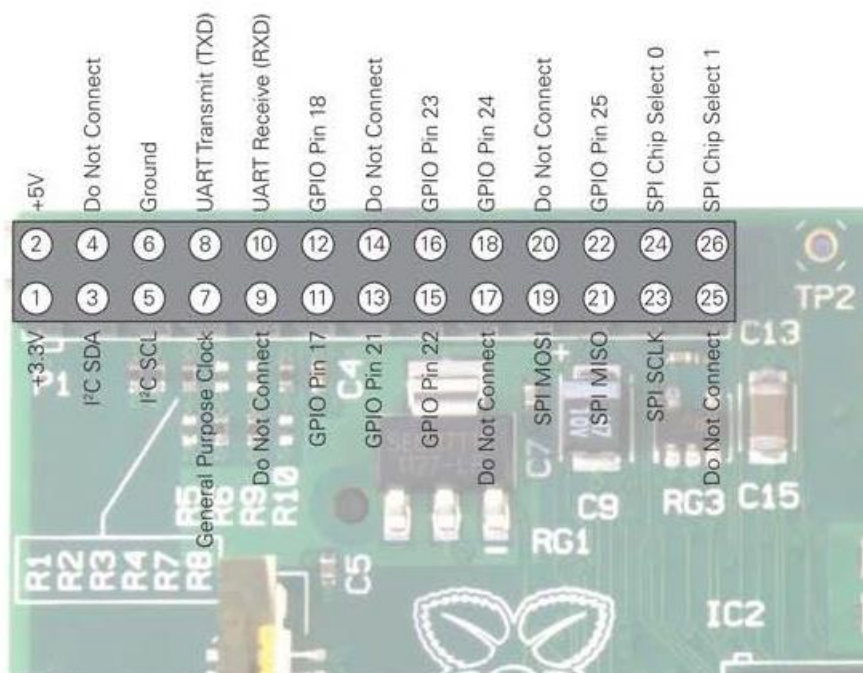


Figura No. 2.3: Distribución de los Pines del Puerto GPIO del Raspberry Pi Modelo B. [1]

Aunque el puerto GPIO provee una alimentación de 5 Vdc, el Raspberry Pi internamente se encuentra trabajando con un voltaje lógico de 3.3 Vdc. Esto es fundamental al momento de conectar algún dispositivo, ya que se podría necesitar de un circuito regulador, en caso de que el dispositivo a conectar no pueda operar con un voltaje de 3.3 voltios.

Los pines que son utilizados de propósitos generales son los pines físicos 11, 12, 13, 15, 16, 18 y 22. El pin 7 provee una señal de clock para propósitos generales o puede ser configurado como un pin de entrada o salida. Estos pines deben recibir señales de 3.3 Vdc, para representar los estados lógicos de 0 y 1.

De igual forma dentro de los pines del puerto GPIO, pin 8 y 10, existe un TX/RX asincrónico UART, el mismo que consiste en un puerto serial, donde el pin 8 es utilizado para transmisión de data y el pin 10 para la recepción. Este puerto serial es configurado mediante la modificación del archivo *cmdline.txt*.

Para el caso del Raspberry, se debe configurar el puerto UART con una velocidad adecuada para poder comunicarse con el módulo GPS GSC3 de la empresa Vicontech. Más adelante se hará una explicación detalla acerca de la configuración de esta tarjeta, sin embargo podemos mencionar que la misma se comunicará con el Raspberry-Pi a una velocidad de 4800 baudios.

Esta velocidad deberá ser seteada en el archivo *cmdline.txt* para la correcta comunicación con dicho módulo. A continuación la instrucción que debe contener mencionado archivo:

```
dwc_otg.lpm_enable=0 console=ttyAMA0, 4800 kgdboc=ttyAMA0, 4800  
console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait.
```

En la Figura No. 2.4, se presenta la ubicación física del archive *cmdline.txt* que debe ser modificado.

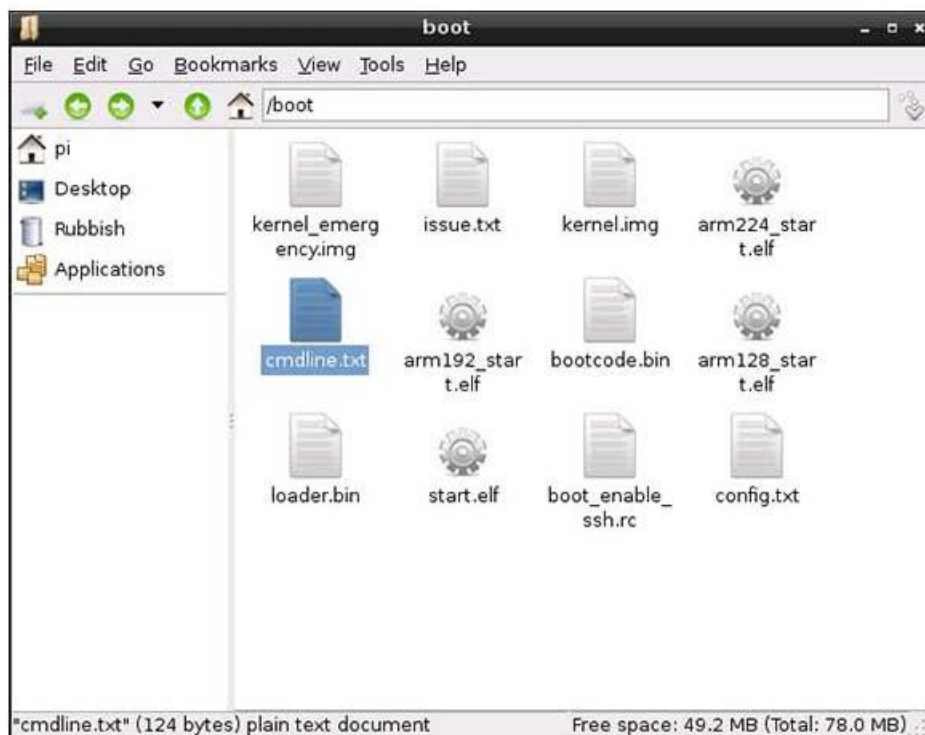


Figura No. 2.4: Ubicación del archivo cmdline.txt. [1].

2.1.1 Instrucción en Python para programación del GPIO y UART.

Para facilitar el manejo de los puertos, es necesario instalar la librería correspondiente, para direccionar adecuadamente cada uno de los puertos que conforman el GPIO.

Debido a que la programación del Raspberry Pi tiene una arquitectura abierta, existen varias librerías en Python que podrían ser instaladas, sin embargo se recomienda la librería `raspberry-gpi-pythonlibrary`. Para descargar de la web e instalar mencionada librería se debe ejecutar las siguientes sentencias:

```
http://raspberrypi-gpio-python.googlecode.com/files/RPi.GPIO-0.3.1a.tar.gz
```

Para extraer el contenido del archivo escribir:

```
tar xvzf RPi.GPIO-0.3.1a.tar.gz
```

Para grabar en un nuevo directorio escribir:

```
Cd RPi.gpio-0.3.1a
```

Para instalar la librería escribir:

```
Sudo python setup.py
```

Es importante mencionar que a pesar de que la librería está instalada, esta debe ser nombrada dentro del programa en la parte superior mediante la sentencia:

```
Import RPi.GPIO as GPIO.
```

Para la configuración de los puertos de entrada o de salida se la realizada de la siguiente forma:

Puerto de Entrada: *GPIO.input*, en nuestro caso configuraremos el pin 10 como entrada de la siguiente manera:

```
GPIO.setup (10, GPIO.IN)
```

Puerto de Salida: *GPIO.output*, en nuestro caso configuraremos el pin 8 como salida de la siguiente manera:

```
GPIO.setup(8, GPIO.OUT)
```


Para trabajar con el puerto serial se debe realizar el mismo procedimiento que se explicó anteriormente, es decir que se debe instalar la librería Pyserial, los pasos son similares que cuando se desea instalar la librería para el manejo de los pines del puerto GPIO.

Para descargar de la web la librería escribir:

```
http://sourceforge.net/projects/pyserial/files/latest/download?source=files
```

Para extraer el contenido del archivo escribir:

```
tar xvzf pyserial-2.6.tar.gz
```

Para grabar en un nuevo directorio escribir:

```
cd pyserial-2.6
```

Para instalar la librería escribir:

```
Sudo python setup.py install
```

De igual manera se debe importar este módulo al programa mediante la instrucción:

```
import serial
```

Para establecer una comunicación standard con un baudrate de 4800, 8 bits, sin paridad y 01 bit de parada es suficiente con escribir las siguientes instrucciones:

```
ser=serial.Serial ()  
ser.baudrate=4800  
ser.port=0  
ser
```

Para almacenar la data que es leída en el puerto del UART se puede utilizar la sentencia:

```
ser.open()  
GPS = ser.readline()
```

2.2 Protocolo de Comunicaciones NMEA 0183.

La asociación “National Electronics Association (NMEA)”, es una asociación sin fines de lucro de los fabricantes, distribuidores, institutos de educación y otros interesados en equipos periféricos relacionados con el área marítima. El protocolo NMEA 0183 se lo define como una interface eléctrica y protocolo de datos para la comunicación efectiva entre los diferentes equipos del área marítima

2.2.1 Interface Eléctrica del Protocolo.

Este protocolo está diseñado tanto para los equipos que transmiten como los que reciben data y emplea una comunicación asincrónica mediante la utilización de una interface serial. Esta interface debe ser configurada con los siguientes parámetros: Baudrate, Número de bits, Bit de Parada, Paridad y Handshake.

NMEA 0183 permite que existan una sola fuente de transmisión (Talker) y varias fuentes de recepción (listeners) en un mismo circuito. La conexión recomendada es a través de conductores que tenga protección

electromagnética especialmente conductores de par trenzado con malla, la misma que debe estar aterrizada solamente del lado del Talker.

No se requieren conectores especiales sin embargo es recomendable que el talker cumpla con el standard RS-422 para la transmisión de datos, el mismo que es un sistema diferencial que utiliza dos líneas. Este es un formato más inmune al ruido y debe ser aterrizado. De igual forma se recomienda que al realizar la interconexión de dispositivos que cumplan con este protocolo, se instale en el receptor opto acopladores para establecer una protección al circuito.

2.2.2 Formato de la Estructura del Protocolo.

Toda la data es transmitida en forma de línea de caracteres (sentencias), que son mostradas con caracteres ASCII. Cada sentencia posee al inicio el siguiente símbolo "\$" y finaliza con <CR> (Return) y <LF> (Fin de Línea).

El formato del protocolo se lo puede resumir de la siguiente forma:

```
$tsss,d1,d2,..... <CR><LF>
```

Las dos primeras letras de la línea anterior después de \$, se denominan el identificador del talker. Los tres caracteres siguientes (sss) son el identificador de los datos que contienen esta línea de caracteres. Los datos d1, d2, etc. representan la información o datos que el talker quiere enviar al resto de usuarios. Se puede añadir si se desea checksum para comprobar la integridad de la data. Por ejemplo tenemos la siguiente instrucción.

\$HCHDM,238,M<CR><LF>

En donde “HC” corresponde al talker que corresponde a compás magnético, los caracteres “HDM” especifica que es un rumbo magnético, el “238” corresponde al valor del rumbo y finalmente la letra “M” indica que es un rumbo magnético.

Una línea de caracteres puede contener hasta 80 caracteres incluido los símbolos del inicio y fin. Si no existe data para un campo específico, el contenido del campo es omitido, pero delimitado por comas, que también son enviadas, sin separación entre ellas. El checksum consiste en “*” y 02 dígitos hexadecimales que representan el OR exclusivo incluidos el “\$” y “*”.

El identificador del Talker que nos interesa para la aplicación que vamos a desarrollar es el siguiente.

GP: Global Position System (GPS)

Por otra parte el identificador de las sentencias que se van a presentar en esta aplicación son las siguientes [4]:

GGA Global Positioning System Fix Data. Time, Position and fix related data for a GPS receiver

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

```

$--GGA,hhmmss.ss,llll.ll,a,yyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh
1) Time (UTC)
2) Latitude
3) N or S (North or South)
4) Longitude
5) E or W (East or West)
6) GPS Quality Indicator,
0 - fix not available,
1 - GPS fix,
2 - Differential GPS fix

```

- 7) Number of satellites in view, 00 - 12
- 8) Horizontal Dilution of precision
- 9) Antenna Altitude above/below mean-sea-level (geoid)
- 10) Units of antenna altitude, meters
- 11) Geoidal separation, the difference between the WGS-84 earth ellipsoid and mean-sea-level (geoid), "-" means mean-sea-level below ellipsoid
- 12) Units of geoidal separation, meters
- 13) Age of differential GPS data, time in seconds since last SC104 type 1 or 9 update, null field when DGPS is not used
- 14) Differential reference station ID, 0000-1023
- 15) Checksum

GLL Geographic Position – Latitude/Longitude

```

      1      2 3      4 5      6 7
      |      | |      | |      | |
$--GLL,llll.ll,a,yyyy.yy,a,hmmss.ss,A*hh

```

- 1) Latitude
- 2) N or S (North or South)
- 3) Longitude
- 4) E or W (East or West)
- 5) Time (UTC)
- 6) Status A - Data Valid, V - Data Invalid
- 7) Checksum

VTG Track Made Good and Ground Speed

```

      1      2 3      4 5      6 7      8 9
      |      | |      | |      | |      | |
$--VTG,x.x,T,x.x,M,x.x,N,x.x,K*hh

```

- 1) Track Degrees
- 2) T = True
- 3) Track Degrees
- 4) M = Magnetic
- 5) Speed Knots
- 6) N = Knots
- 7) Speed Kilometers Per Hour
- 8) K = KilometresPer Hour
- 9) Checksum

RMC Recommended Minimum Navigation Information

```

      1      2 3      4 5      6 7      8      9      10      11 12
      |      | |      | |      | |      |      |      |      | |
$--RMC,hmmss.ss,A,llll.ll,a,yyyy.yy,a,x.x,x.x,xxxx,x.x,a*hh

```

- 1) Time (UTC)
- 2) Status, V = Navigation receiver warning
- 3) Latitude
- 4) N or S
- 5) Longitude
- 6) E or W
- 7) Speed over ground, knots
- 8) Track made good, degrees true
- 9) Date, ddmmyy
- 10) Magnetic Variation, degrees
- 11) E or W
- 12) Checksum

GSA GPS DOP and active satellites

```

      1 2 3      14 15 16      17 18
      | | |      | | |      | |
$--GSA,a,a,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x*hh

```

- 1) Selection mode
- 2) Mode

- 3) ID of 1st satellite used for fix
- 4) ID of 2nd satellite used for fix
- ...
- 14) ID of 12th satellite used for fix
- 15) PDOP in meters
- 16) HDOP in meters
- 17) VDOP in meters
- 18) Checksum

GSV Satellites in view

```

      1 2 3 4 5 6 7      n
      | | | | | | |      |
$--GSV,x,x,x,x,x,x,x,...*hh
1) total number of messages
2) message number
3) satellites in view
4) satellite number
5) elevation in degrees
6) azimuth in degrees to true
7) SNR in dB
more satellite infos like 4)-7)
n) Checksum

```

2.3 Descripción de los Sistemas de Posicionamiento Satelital.

El SPG o GPS (Global Positioning System: sistema de posicionamiento global) o NAVSTAR-GPS es un sistema global de navegación por satélite (GNSS) que permite determinar en todo el mundo la posición de un objeto, una persona o un vehículo con una precisión hasta de centímetros (si se utiliza GPS diferencial), aunque lo habitual son unos pocos metros de precisión. El sistema fue desarrollado, instalado y actualmente operado por el Departamento de Defensa de los Estados Unidos.

El GPS funciona mediante una red de 24 satélites en órbita sobre el planeta tierra, a 20.200 km, con trayectorias sincronizadas para cubrir toda la superficie de la Tierra. Cuando se desea determinar la posición, el receptor que se utiliza para ello localiza automáticamente como mínimo tres satélites de la red, de los que recibe unas señales indicando la identificación y la hora del reloj de cada uno de ellos.

Con base en estas señales, el aparato sincroniza el reloj del GPS y calcula el tiempo que tardan en llegar las señales al equipo, de tal modo que mide la distancia al satélite mediante triangulación, la cual se basa en determinar la distancia de cada satélite respecto al punto de medición. Conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los tres satélites.

Conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtiene la posición absoluta o las coordenadas reales del punto de medición. También se consigue una exactitud extrema en el reloj del GPS, similar a la de los relojes atómicos que llevan a bordo cada uno de los satélites.

Mediante la triangulación, cada satélite indica que el receptor se encuentra en un punto en la superficie de la esfera, con centro en el propio satélite y de radio equivalente a la distancia total hasta el receptor.

Si se obtiene la información de dos satélites queda determinada una circunferencia que resulta cuando se intersecan las dos esferas en algún punto de la cual se encuentra el receptor.

Mientras que si se dispone la información de un cuarto satélite, se elimina el inconveniente de la falta de sincronización entre los relojes de los receptores GPS y los relojes de los satélites. Y es en este momento cuando el receptor GPS puede determinar una posición 3D exacta (latitud, longitud y altitud).

2.4 Descripción del Firmware del Módulo GPS CSC3 Vincotech.

El módulo GPS CSC3 de Vincotech soporta una interface serial bidireccional. Esta es implementada utilizando la interface del procesador del GPS, utilizando una comunicación full dúplex del tipo UART.

La configuración por defecto de este módulo es una comunicación por puerto serial con los siguientes parámetros: 4800 baudios, 8 bits, No Paridad y 1 Bit de parada, no flujo de control.

El módulo en mención soporta 08 sentencias en formato NMEA 0183 relacionadas con la información que entrega el GPS, las mismas que fueron explicadas en el punto anterior y son las siguientes:

- a. \$GPGGA (default: ON)
- b. \$GPVTG (default: OFF)
- c. \$GPRMC (default: ON)
- d. \$GPGSA (default: ON)
- e. \$GPGSV (default: ON, 0.2Hz)
- f. \$GPGLL (default: OFF)

2.4.1 Mensajes de Inicialización del GPS.

Después del encendido del módulo GPS o reset del módulo, este dispositivo enviará el primer mensaje de inicialización en el cual se podrá apreciar las siguientes sentencias:

```
$PSRFTXT,Version: GSW3.5.0_3.5.00.00-C25P2.01 *03
```

```
$PSRFTXT,TOW: 0*25
```

```
$PSRFTXT,WK: 1517*67
```

```
$PSRFTXT,POS: 6378137 0 0*2A
```


\$PSRFTXT,CLK: 96250*25

\$PSRFTXT,CHNL: 12*73

\$PSRFTXT,Baud rate: 4800*65

Estos mensajes tiene información de la versión del firmware del GPS, tiempo del GPS, posición entre otros. Esta información realmente no debe ser utilizada durante la implementación del programa por cuanto es únicamente informativa.

2.4.2 Sentencias propietarias del Módulo GPS.

El módulo GPS permite la habilitación de mensajes de entrada en formato NMEA. Por defecto en el puerto 0 es configurado como modo NMEA. Los mensajes pueden ser enviados mediante la utilización de un terminal de programa. Los siguientes mensajes de entrada son soportadas:

Message	MID⁽¹⁾	Description
Set serial port	100	Set Port 0 parameters and protocol
Reset Configuration	101	Initialize various start up behaviors
Query/rate control	103	Query standard NMEA message and/or set out-put rate
Development data On/Off	105	Development Data messages On/Off
Select Datum	106	Selection of datum to be used for coordinate transforming

(1) Message Identification (MID)

Tabla No. 1: Mensajes de Entrada NMEA al Módulo GPS. [5]

2.4.3 Configuración del Puerto Serial del Módulo GPS.

El comando que es utilizado para configurar los parámetros de baud rate, data bits, stop bits y paridad es el siguiente:

```
$PSRF100,0,9600,8,1,0*0C
```

Name	Example	Description
Message ID	\$PSRF100	PSRF100 protocolheader
Protocol	0	0 SiRFbinary / 1 NMEA
Baud	9600	4800, 9600, 19200, 38400, 57600, 115200
DataBits	8	8, 7 ⁽¹⁾
StopBits	1	0, 1
Parity	0	0 none / 1 odd / 2 even
Checksum	*0C	End of messagetermination

(1) SiRF protocol is only valid for 8 data bits, 1 stop bit and no parity

Tabla No. 2: Configuración del Puerto Serial [5]

2.4.4 Selección de las Sentencias que enviará el Módulo GPS.

Se requiere de comando especiales para habilitar o deshabilitar las diferentes sentencias que son enviadas por la salida serial del GPS. Como se mencionó anteriormente este módulo GPS en especial tiene la capacidad de enviar los mensajes GGA, GLL, GSA, GSV, RMC y VTG. Se puede de igual forma habilitar o no el envío del checksum en caso de ser necesario.

A continuación las instrucciones que son necesarias para habilitar el checksum y habilitar el envío de los datos de velocidad que por defecto se encuentra deshabilitada:

1. Mensaje GGA con checksum habilitado

- \$PSRF103,00,01,00,01*25

2. Mensaje VTG para salida a 1 Hz con checksum habilitado

- \$PSRF103,05,00,01,01*20

La interpretación de estos mensajes se puede visualizar en las siguientes tablas:

Name	Example	Units	Description
Message ID	\$PSRF103		PSRF103 protocolheader
Msg	00		Ver Tabla 4.
Mode	01		0=SetRate, 1=Query
Rate	00	seconds	Output rate 0 off Max 255
CksumEnable	01		0 Disable Checksum 1 EnableChecksum
Checksum	*25		End of message termination

Tabla No. 3: Formato de la Sentencia de Control [5]

Value	Description
0	GGA
1	GLL
2	GSA
3	GSV
4	RMC
5	VTG
6	MSS (If internal beacon is supported)
7	Notdefined
8	ZDA (if 1PPS output is supported)
9	Notdefined

Tabla No. 4: Valor asignado a cada Mensaje de Control [5]

2.5 Descripción general de un acelerómetro

Se denomina acelerómetro a cualquier instrumento destinado a medir aceleraciones. La aceleración se define como la razón entre el cambio de velocidad y el intervalo de tiempo en el cual esto ocurre. Es decir mide que tan rápidos son los cambios de velocidad de un objeto.

El funcionamiento estándar de los acelerómetros consiste en que el sensor envía señales PWM, en donde varía la amplitud del ancho de pulso según la aceleración del sensor en la dirección de las coordenadas cartesianas x, y.

Es por eso que se puede decir que:

- Una aceleración grande significa que la velocidad cambia rápidamente
- Una aceleración pequeña significa que la velocidad cambia lentamente.
- Una aceleración cero significa que la velocidad no cambia

Por convención se ha determinado que si un móvil está disminuyendo su rapidez, entonces su aceleración va en el sentido contrario al movimiento. En cambio si éste aumenta su rapidez, la aceleración tiene el mismo sentido que el propio objeto.

Este principio es también conocido como La segunda ley de Newton la cual nos dice:

“Que la fuerza neta aplicada sobre un cuerpo es proporcional a la aceleración que adquiere dicho cuerpo”. Una forma más rigurosa y clara, sería su expresión

matemática, la cual relaciona tres magnitudes como son fuerza, masa y aceleración:

$$F(t) = m \cdot a(t) \quad (2.1)$$

Gracias a esta relación podemos determinar la ecuación diferencial de funcionamiento del acelerómetro:

$$F(t) = m \cdot a(t) = m \cdot x''(t) + b \cdot x'(t) + k \cdot x(t) \quad (2.2)$$

Dónde:

- k: valor del término de la constante recuperadora del muelle o placa.
- m: valor del término de masa.
- b: valor del coeficiente de amortiguamiento de viscosidad.
- x: desplazamiento en un eje del sensor.

Definimos como la frecuencia de resonancia la relación existente entre las constantes:

$$w = \sqrt{\frac{k}{m}} \quad (2.3)$$

Ahora definimos el factor de calidad Q como el parámetro que sirve para medir la calidad de un sistema resonante, entendiendo por sistema resonante aquel en el que para una frecuencia determinada (frecuencia de resonancia) responde de una manera mayor que al resto de frecuencias. En la ecuación diferencial el factor de calidad viene determinado por las tres variables (m, b y k) y será:

$$Q = \sqrt{\frac{k.m}{b}} \quad (2.4)$$

El factor de calidad también puede entenderse como la relación a dimensional entre la energía que un sistema almacena y la que el sistema consume.

A partir de las expresiones anteriores rescribimos de nuevo la ecuación diferencial y obtenemos la siguiente expresión:

$$a(t) = \ddot{x}(t) + \frac{w}{Q} \dot{x}(t) + w^2 x(t) \quad (2.5)$$

Donde:

- Q: es factor de calidad del sistema.
- Wn: es la frecuencia de resonancia del sistema.

Si trabajamos en el dominio de la frecuencia y utilizamos la transformada de

Laplace obtenemos que la expresión del acelerómetro sea:

$$A(s) = s^2 \cdot x(s) + \frac{w}{Q} s \cdot x(s) + w^2 x(s) \quad (2.6)$$

Podemos calcular la función de transferencia del sistema H(s). Esta función de transferencia, es en realidad la expresión del cálculo de la sensibilidad del sensor (en el dominio de la transformada), es decir, la respuesta en unidades de desplazamiento respecto la entrada de aceleración.

Cuyo valor es:

$$H(s) = \frac{X(s)}{A(s)} \quad (2.7)$$

Analizando las expresiones anteriores en el dominio transformando y despejando las variables de las ecuaciones obtenemos la siguiente expresión:

$$H(s) = \frac{X(s)}{A(s)} = \frac{1}{s^2 - \frac{W}{Q}s + W^2} \quad (2.8)$$

CAPÍTULO 3

3 DISEÑO E IMPLEMENTACIÓN DEL PROYECTO.

En el presente capítulo se detallará el proceso que se ha llevado a cabo para realizar la implementación de este proyecto, iniciando con el proceso de conexión de componentes, la configuración del minicomputador Raspberry Pi para que pueda reconocer al Módulo GPS y al simulador FURUNO GPS/WA A5; así como también, se detalla el procedimiento requerido para realizar la instalación del Software GPSD para la lectura de los datos provenientes del módulo GPS y finalmente se presenta el desarrollo del software para adquisición de datos desde el módulo GPS o simulador, el mismo que fue implementado en Python.

3.1 Identificación de Componentes.

Para realizar la identificación y conexión de los componentes, fue necesario identificar los pines de entrada y salida del módulo GPS, para lo cual con la

ayuda de un osciloscopio, se determinó que el dispositivo se encontraba trabajando correctamente. En la figura 3.1, se puede apreciar la prueba realizada.



Figura No. 3.1: Determinación del Funcionamiento del Módulo mediante un Osciloscopio.

Los pulsos que se pueden visualizar en la Figura 3.1, corresponden a la data que está saliendo del dispositivo GPS. El tipo de transmisión de datos es serial, por lo que los pines del UART (Rx – Tx) se deberán conectar a los pines del Raspberry Pi, además de la alimentación respectiva que necesita el mencionado módulo.

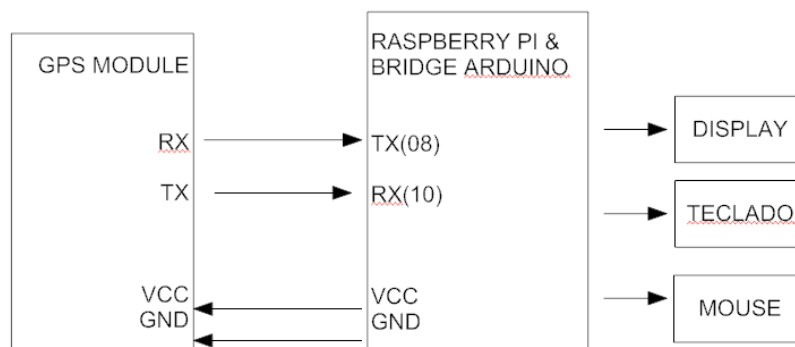


Figura No 3.2. Diagrama Básico de Interconexión.

En la figura 3.3 se puede visualizar la interconexión física de estos módulos:

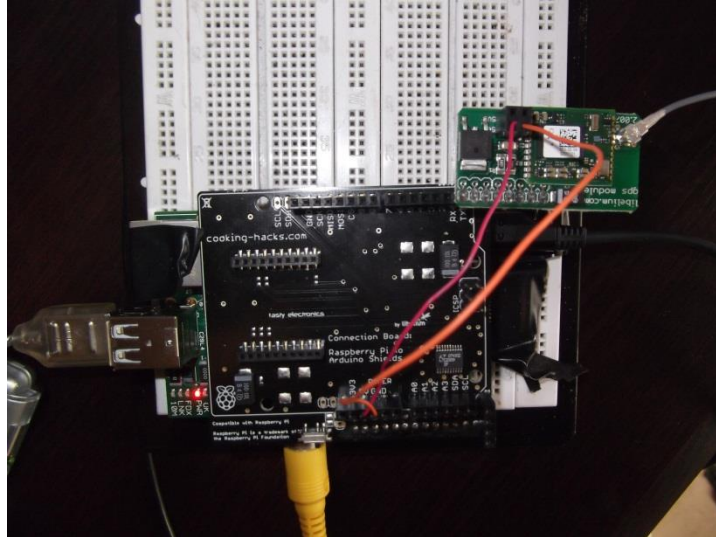


Figura No 3.3. Interconexión Física de Componentes.

Se debe tomar la precaución, que para recibir la información de los satélites, se requiere que la antena GPS se encuentre al exterior del edificio, sin embargo es factible colocar la antena en la cercanía de una ventana, a pesar que el vidrio puede degradar levemente las señales recibidas por los GPS, después de un cierto tiempo se consigue recibir señal desde los satélites. En la figura 3.4 se puede apreciar las pruebas de recepción realizadas con un pyrex de 5mm de grosor, con el cual se obtuvieron resultados exitosos.

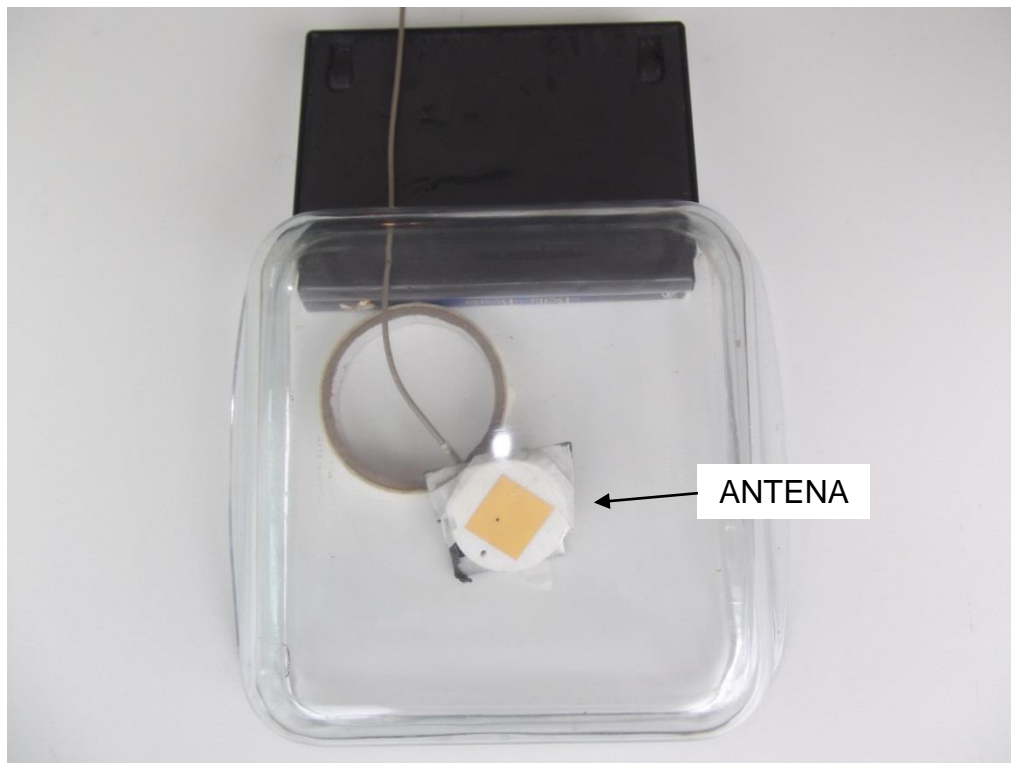


Figura No 3.4. Pruebas de Recepción con Vidrio de 5mm de Espesor.

Se necesita al menos 03 señales de satélites con niveles de señal ruido sobre 18 para que el módulo pueda calcular los valores de posición y velocidad de un blanco.

3.2 Configuración del Raspberry para conexión con Módulo GPS y Simulador.

Uno de los retos más importantes de este proyecto, es lograr la comunicación con el módulo GPS, para lo cual es necesario configurar los puertos UART, para la utilización del módulo GPS y del puerto USB para la utilización de una comunicación RS232 para la adaptación del simulador que es utilizado en el presente proyecto.

Una vez que el dispositivo que se encuentra conectado, este debe ser seteado vía comandos en el terminal de Linux. Las direcciones físicas de los dispositivos en mención son:

1.- MODULO GPS: `//dev/ttyAMA0`

2.- SIMULADOR GPS:`//dev/ttyUSB0`

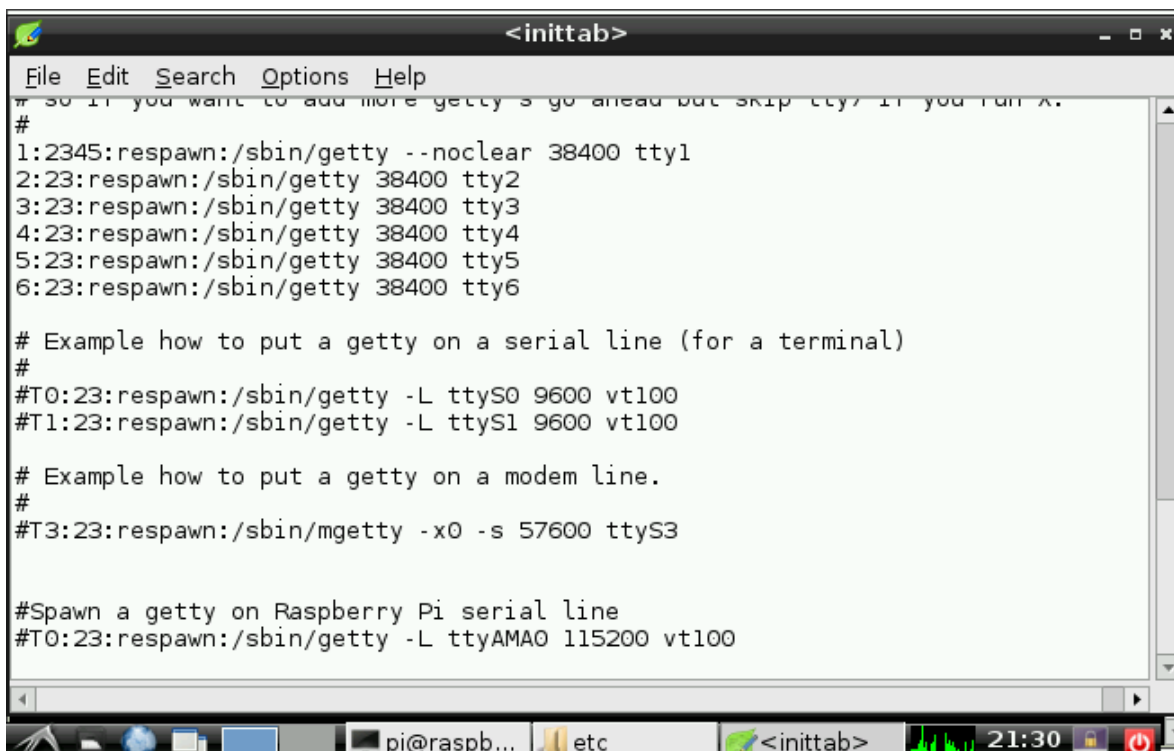
Para poder establecer esas direcciones se debe modificar el archivo `/boot/cmdline.txt`, debiéndose visualizar la siguiente instrucción en el mencionado archivo.



```
*<cmdline.txt>
File Edit Search Options Help
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4....
. . . elevator=deadline rootwait
```

Figura No 3.5 . Edición del archivo `cmdline.txt`

Mientras tanto en el archivo `/etc/inittab`, se debe comentar todo lo relacionado con el puerto serial. Debiéndose visualizarlo de la siguiente manera:



```

File Edit Search Options Help
# so if you want to add more getty's go ahead but skip tty7 if you run X.
#
1:2345:respawn:/sbin/getty --noclear 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6

# Example how to put a getty on a serial line (for a terminal)
#
#T0:23:respawn:/sbin/getty -L ttyS0 9600 vt100
#T1:23:respawn:/sbin/getty -L ttyS1 9600 vt100

# Example how to put a getty on a modem line.
#
#T3:23:respawn:/sbin/mgetty -x0 -s 57600 ttyS3

#Spawn a getty on Raspberry Pi serial line
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

```

Figura No 3.6. Edición del archivo Initab.txt

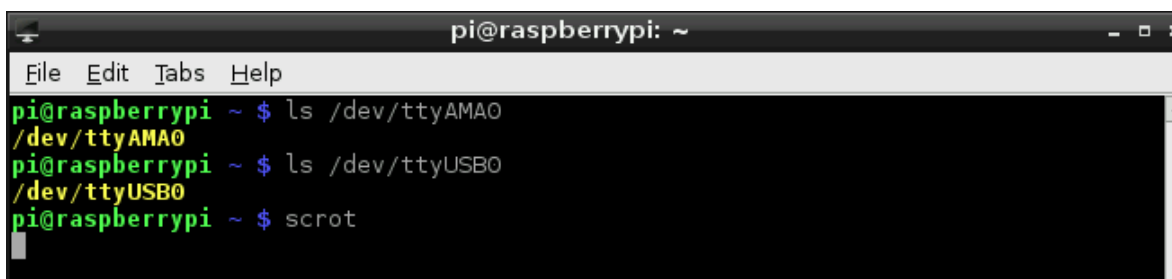
Es necesario mencionar que estos archivos son protegidos, razón por la cual se debe dar los privilegios de escritura, vía comando:

```
sudo chmod 666 <ruta y nombre del archivo>
```

Posteriormente se puede visualizar si los puertos están disponibles y reconocidos mediante la instrucción:

```
ls /dev/ttyUSB, para el caso del simulador y ls /dev/tty AMA0 para el módulo GPS.
```

En la figura 3.7, se puede visualizar el resultado de estas instrucciones:



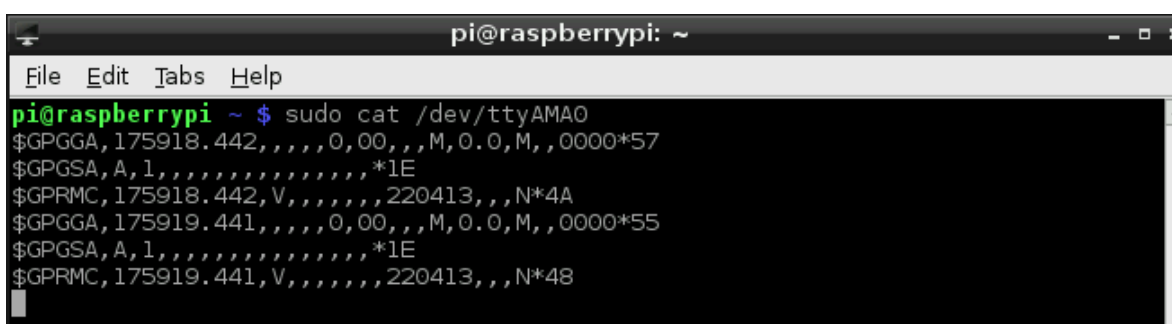
```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi ~ $ ls /dev/ttyAMA0  
/dev/ttyAMA0  
pi@raspberrypi ~ $ ls /dev/ttyUSB0  
/dev/ttyUSB0  
pi@raspberrypi ~ $ scrot
```

Figura No 3.7 . Resultado del Comando ls en Consola

Para ver rápidamente lo que está ingresando desde el GPS conectado se puede ingresar el siguiente comando, dependiendo del puerto en el que se encuentre el dispositivo:

```
sudo cat /dev/ttyAMA0 o sudo cat /dev/ttyUSB0
```

A continuación el resultado del comando “cat” en el terminal.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi ~ $ sudo cat /dev/ttyAMA0  
$GPGGA,175918.442,,,,,0,00,,,M,0.0,M,,0000*57  
$GPGSA,A,1,,,,,,,,,,,,,*1E  
$GPRMC,175918.442,V,,,,,220413,,N*4A  
$GPGGA,175919.441,,,,,0,00,,,M,0.0,M,,0000*55  
$GPGSA,A,1,,,,,,,,,,,,,*1E  
$GPRMC,175919.441,V,,,,,220413,,N*48
```

Figura No 3.8. Resultado del Comando “cat” en el Terminal

3.3 Instalación de Software para lectura de data del GPS.

Como se indicó en el capítulo anterior, el módulo GPS entrega la data basada en el protocolo NMEA 183.0, para lo cual es necesario la instalación del

software que nos permita determinar si estamos recibiendo o no la data desde mencionado módulo.

En primer lugar se realizó la instalación del programa MINICOM, utilizando la siguiente instrucción desde el terminal:

```
Sudo apt-get install minicom.
```

Posteriormente se configura el puerto que se desea utilizar mediante la instrucción:

```
minicom -s
```

Se obtiene la siguiente pantalla desde la cual, se configura el puerto, velocidad y otras características como paridad, número de bits, bit de parada y control de flujo de los datos. A continuación se presenta la pantalla donde se configura lo antes indicado:



```
pi@raspberrypi: ~
File Edit Tabs Help

+-----+
| A - Serial Device      : /dev/ttyAMA0 |
| B - Lockfile Location  : /var/lock    |
| C - Callin Program     :              |
| D - Callout Program    :              |
| E - Bps/Par/Bits       : 9600 8N1    |
| F - Hardware Flow Control : Yes       |
| G - Software Flow Control : Yes       |
+-----+
Change which setting? █
+-----+
| Screen and keyboard |
| Save setup as dfl   |
| Save setup as..    |
| Exit                |
| Exit from Minicom  |
+-----+
```

Figura No 3.9. Configuración del Software Minicom

Una vez que se ejecuta correctamente esta configuración de acuerdo al puerto que vamos a utilizar, sea el USB o el UART, seleccionamos “Exit” y se empieza a recibir la data desde el dispositivos seleccionado. A continuación la pantalla que representa lo antes indicado:

```

pi@raspberrypi: ~
File Edit Tabs Help
Welcome to minicom 2.6.1
OPTIONS: I18n
Compiled on Apr 28 2012, 19:24:31.
Port /dev/ttyAMA0
Press CTRL-A Z for help on special keys
95,S,08000.2981,W,1,09,1.1,28.2,M,12.4,M,,0000*69
$GPGSA,A,3,06,20,14,03,11,32,01,16,19,,,1.9,1.1,1.6*30
$GPRMC,202155.000,A,0211.1895,S,08000.2981,W,0.04,72.18,220413,,,A*53
$GPGGA,202156.000,0211.1892,S,08000.2981,W,1,09,1.1,27.6,M,12.4,

```

Figura No 3.10: Datos visualizado en el Software Minicom.

Como se puede apreciar en la Figura 3.10, ya tenemos data que es presentada en el software de visualización, sin embargo esta data debe ser manipulada para que pueda ser correctamente visualizada, entendible y podamos obtener los datos de posición y velocidad, objetivo del presente trabajo.

De igual forma se puede instalar otro software mucho más amigable que el anteriormente explicado, el mismo que se denomina “Cutecom”. Este software se lo puede conseguir mediante la instrucción:

```
sudo apt-get install cutecom.
```

La interfaz gráfica de este software se la visualiza en la figura 3.11:

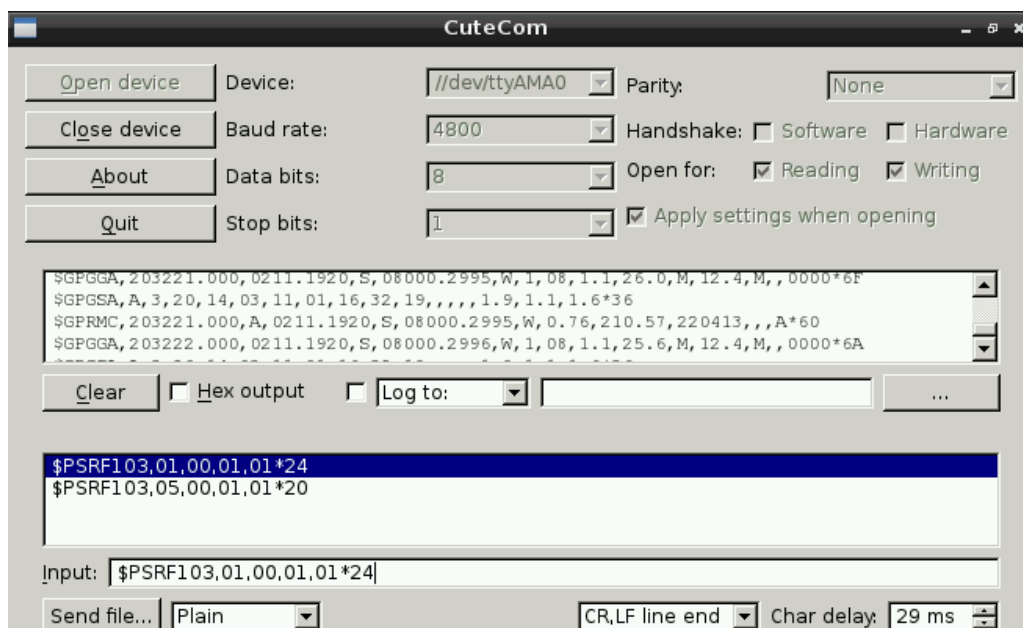


Figura No 3.11: Datos visualizado en el Software Cutecom. Como se puede ver en este software existe una presentación similar de la data que con el software Minicom, sin embargo es factible introducir comandos que pueden ser interpretados por el módulo GPS o simulador.

Para el efecto posee una celda denominada input, en el cual se introduce la instrucción que se desea enviar hacia el dispositivo. Cabe mencionar que para manejar el envío de la data con otra velocidad es necesario introducir la sentencia adecuada. A continuación ejemplos de las instrucciones para el cambio de velocidad:

4800/8/N/1: \$PSRF100,1,4800,8,1,0*0E

9600/8/N/1: \$PSRF100,1,9600,8,1,0*0D

19200/8/N/1: \$PSRF100,1,19200,8,1,0*38

Es necesario mencionar que se debe también configurar mediante el terminal la velocidad a la cual trabajará el puerto, para eso aplicamos la siguiente instrucción:

```
stty speed 9600 </dev/ttyAMA0.
```

De la misma forma en la cual se configura la velocidad de transmisión es también factible realizar la activación de una serie de sentencias que envían información dedicada desde el GPS. Estas instrucción también se las activan desde el cuadro input del programa Cutecom.

Para nuestro proyecto vamos a aplicar los siguientes 2 comandos, los cuales son los principales para que nuestro GPS pueda tener una lectura exacta de ubicación y velocidad.

```
GLL
```

```
$PSRF103,01,00,01,01*24
```

```
VTG
```

```
$PSRF103,05,00,01,01*20
```

3.4 Instalación del Software GPS Daemon (gpsd).

El software GPS Daemon (gpsd), es el software mediante el cual vamos a poder entender la data que está enviando el GPS. Este software envía la información ordenada de tal manera que la interpretación de la información del blanco se lo hace de una manera sencilla.

Este programa, como la mayoría de los desarrollados en Linux se encuentra de manera gratuita y se lo instala mediante la siguiente instrucción:

```
Sudo apt-get install gpsd-clients python-gps.
```

Con estas instrucciones estamos habilitando tanto el software GPSD y también la librería GPS de python, la misma que la utilizaremos más adelante.

Dependiendo del puerto en el que se encuentre nuestro dispositivo GPS, debemos habilitarlo para la lectura mediante el siguiente comando:

```
Sudo gpsd /dev/ttyAMA0 -F /var/run/gpsd.sock
```

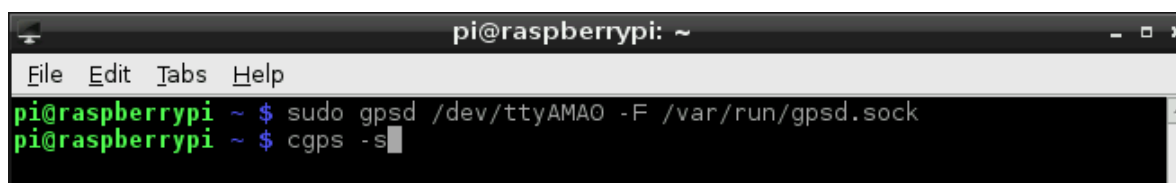
A screenshot of a terminal window titled 'pi@raspberrypi: ~'. The window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows two lines of command execution: the first line is 'pi@raspberrypi ~ \$ sudo gpsd /dev/ttyAMA0 -F /var/run/gpsd.sock' and the second line is 'pi@raspberrypi ~ \$ cgps -s' with a cursor at the end.

Figura No.3.12: Comandos para ejecución del Software GPSD

Una vez que se tiene instalado el mencionado software, lo ejecutamos mediante el comando:

```
cgps -s
```

Como se puede visualizar en la figura 3.13, se despliega la información que entrega de manera ordenada el GPS. En esta figura se puede visualizar la información de posición y velocidad del blanco, objetivo del presente proyecto:

```

pi@raspberrypi: ~
File Edit Tabs Help

Time:      2013-04-22T23:49:47.000Z
Latitude:  2.186493 S
Longitude: 80.005018 W
Altitude:  16.9 m
Speed:     0.5 kph
Heading:   280.7 deg (true)
Climb:     42.0 m/min
Status:    3D FIX (11 secs)
Longitude Err: +/- 12 m
Latitude Err: +/- 8 m
Altitude Err: +/- 30 m
Course Err: n/a
Speed Err: +/- 89 kph
Time offset: 0.643
Grid Square: EI97xt

PRN:  Elev:  Azim:  SNR:  Used:
11   66   074   24   Y
13   65   258   26   Y
23   58   195   32   Y
1    58   131   32   Y
7    27   348   27   Y
20   26   174   20   Y
19   23   021   28   Y
17   19   232   21   Y
32   17   148   28   Y
28   06   297   00   N
3    04   030   00   N
8    03   333   00   N

```

Figura No. 3.13: Visualización de datos en el Software GPSD

Cabe mencionar que este programa puede ser utilizado únicamente cuando se tiene recepción de datos reales de GPS, por lo que para la utilización de simuladores se debe realizar una programa en Python, que permita interpretar la data simulada, la misma que es muy similar a la data real. La única diferencia es que no existe información de los satélites que entregan información, razón por la cual el software GPSD identifica que la data recibida no es válida.

3.5 Adquisición de datos desde el módulo GPS utilizando Python

A pesar que ya se ha logrado controlar la posición y velocidad de un blanco mediante el programa “GPSD”, se ha considerado pertinente desarrollar el código requerido para poder monitorear la data del GPS, mediante la utilización de un programa desarrollado en Python, el mismo que servirá para futuras implementaciones. Para realizar el desarrollo del software nos guiaremos del siguiente algoritmo, el mismo que será la base para la implementación.

Descripción del Algoritmo

1. Inicialización de los puertos
2. Definición de puertos a utilizar.
3. Ingreso en un lazo infinito
4. Captura de los datos entregados por el módulo GPS.
5. Si el dato recibido corresponde a una sentencia GGA o RMC, procesa la información y los muestra en la pantalla



Figura No 3.14. Algoritmo para la Implementación del Software.

Cabe mencionar que en Python, al igual que en otros lenguajes es necesario utilizar librerías que contienen comandos especiales que permiten el desarrollo

del software de una manera más sencilla. Para el caso del manejo del puerto UART del Raspberry Pi es necesario instalar y utilizar la librería serial. La instalación de esta librería fue explicada en el capítulo anterior.

De igual forma se requiere la instalación de la librería respectiva para la elaboración de la interfaz gráfica. La librería más utilizada es la denominada Tkinter y la forma de instalarla es mediante código en el terminal mediante la siguiente instrucción:

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

Una vez que se instala esa librería se debe llamar a la misma en el inicio del programa, para que se habiliten todas sus funciones.

El programa desarrollado en Python, se compone de dos módulos internos, el primero que corresponde a la adquisición de datos y procesamiento de los mismos y el segundo que corresponde a la interfaz gráfica (HMI).

A continuación se presenta el código que realiza la adquisición de datos y se configura la velocidad de transmisión de datos:

```
#PROGRAMA PARA ADQUISICION DE DATOS DESDE GPS
```

```
import serial
usbport = '/dev/ttyUSB0'
Ser=serial.Serial(usbport, 4800, timeout=1)

while 1:
    Ser.read()
    GPS = Ser.readline()
    str1 = GPS
    str2= "GPGGA"
    m=str1.find(str2)
```

```

str3= "GPRMC"
if m==0:

    lista = []
    pos_inicial = 0
    try:
        while True:
            pos_inicial = GPS.index(',',pos_inicial+1)
            lista.append(pos_inicial)

    except ValueError:
        a = 1

    LAT_GRADOS = GPS[(lista[1])+1:(lista[1])+3]
    LAT_MINUTOS = GPS[(lista[1])+3:(lista[2])]
    LAT_ORIENTACION = GPS[(lista[2])+1:(lista[3])]

    LON_GRADOS = GPS[(lista[3])+1:(lista[3])+4]
    LON_MINUTOS = GPS[(lista[3])+4:(lista[4])]
    LON_ORIENTACION = GPS[(lista[4])+1:(lista[5])]

    LATG = LAT_GRADOS
    LATM = LAT_MINUTOS
    DIRL = LAT_ORIENTACION

    LONG = LON_GRADOS
    LONM = LON_MINUTOS
    DIRLO = LON_ORIENTACION

    print (" LATITUD = " + LATG + "° " + LATM + "' " +
DIRL)
    print ("LONGITUD = " + LONG + "° " + LONM + "' " +
DIRLO)

m=str1.find(str3)
if m==0:
    lista = []
    pos_inicial = 0
    try:
        while True:
            pos_inicial = GPS.index(',',pos_inicial+1)
            lista.append(pos_inicial)

    except ValueError:
        a = 1

    VEL = GPS[(lista[6])+1:lista[7]]
    print ("VELOCIDAD =" + VEL + " Nudos")

```

Finalmente, el otro código que realiza la presentación de la HMI, como ya se indicó utiliza la función Tkinter para poder elaborar los diferentes cuadros de diálogo. A continuación el código completo, incluyendo el módulo de adquisición de datos e interfaz gráfica:

```
#PROGRAMA PARA ADQUISICION DE DATOS DESDE GPS Y PRESENTACIÓN

import threading
import serial
import time
from time import sleep
usbport = '/dev/ttyAMA0'
Ser=serial.Serial(usbport, 4800, timeout=1)

from Tkinter import *

usbport = '/dev/ttyAMA0'
Ser=serial.Serial(usbport, 4800, timeout=1)

serialdata = []
velocidad_data = [0]
latitud_data = [0]
longitud_data = [0]
data = True
global GPS
class SensorThread(threading.Thread):

    def run(self):
        try:

            while 1:

                Ser.read()
                GPS = Ser.readline()
                veloc = GPS
                serialdata.append(veloc)
                sleep(1.25)
                print veloc

                global VEL
                global LATITUD
                global LONGITUD
                global GPS1

                str1 = GPS
                str2= "GPGGA"
                m=str1.find(str2)
```



```

str3= "GPRMC"
if m==0:
    # print GPS

    lista = []
    pos_inicial = 0
    try:
        while True:
            pos_inicial =
GPS.index(',',',pos_inicial+1)
            lista.append(pos_inicial)

        except ValueError:
            a = 1

            LAT_GRADOS =
GPS[(lista[1])+1:(lista[1])+3]
            LAT_MINUTOS =
GPS[(lista[1])+3:(lista[2])]
            LAT_ORIENTACION =
GPS[(lista[2])+1:(lista[3])]

            LON_GRADOS =
GPS[(lista[3])+1:(lista[3])+4]
            LON_MINUTOS =
GPS[(lista[3])+4:(lista[4])]
            LON_ORIENTACION =
GPS[(lista[4])+1:(lista[5])]

            LATG = LAT_GRADOS
            LATM = LAT_MINUTOS
            DIRL = LAT_ORIENTACION

            LONG = LON_GRADOS
            LONM = LON_MINUTOS
            DIRLO = LON_ORIENTACION

            GPS1 = GPS

            LATITUD = LATG + "Â° " + LATM + "' " +
DIRL
            LONGITUD = LONG + "Â° " + LONM + "' " +
DIRLO

            Ser.flushOutput()

            print (" LATITUD = " + LATG + "Â° " +
LATM + "' " + DIRL)
            print ("LONGITUD = " + LONG + "Â° " +
LONM + "' " + DIRLO)

            latitud_data.append(LATITUD)
            longitud_data.append(LONGITUD)

```

```

        m=str1.find(str3)
        if m==0:
            lista = []
            pos_inicial = 0
            try:
                while True:
                    pos_inicial =
GPS.index(',',',pos_inicial+1)
                    lista.append(pos_inicial)

            except ValueError:
                a = 1
                Ser.flushInput()
                GPS1 = GPS
                VEL = GPS[(lista[6])+1:lista[7]]
                print ("VELOCIDAD =" + VEL + " Nudos")
                VEL = VEL + " Nudos"
                velocidad_data.append(VEL)

    except KeyboardInterrupt:
        exit()

```

```

class Gui(object):
    def __init__(self):
        self.root = Tk()
        self.root.title("Microcontroladores Avanzados")
        self.lbl = Label(self.root, text="SISTEMA DE CONTROL DE
POSICION Y VELOCIDAD
")
        self.Velocidad = Label(self.root, text="Velocidad = ")
        self.Velocidad2 = Label(self.root, text= "" )
        self.Data = Label (self.root, text = "Data de Ingreso =
")
        self.Data_in = Label (self.root, text = "" )

        self.Latitud = Label(self.root, text="Latitud = ")
        self.Latitud2 = Label(self.root, text= "" )
        self.Longitud = Label(self.root, text="Longitud = ")
        self.Longitud2 = Label(self.root, text= "")

        self.autor1= Label(self.root, text="AUTORES: Fernando
Chávez Castrillón")
        self.autor2= Label(self.root, text="Gonzalo Yuquilema")

        self.updateGUI()
        self.readSensor()

    def run(self):

        self.root.geometry("550x300+50+20")

```

```

self.lbl.pack (padx=15,pady=15)
self.Velocidad2.place (x=100,y=50)
self.Velocidad2.after (150, self.updateGUI)

self.Longitud2.place (x=100,y=110)
self.Latitud.place (x=10,y=80)
self.Latitud2.place (x=100,y=80)
self.Longitud.place (x=10,y=110)

self.autor1.place (x=10,y=220)
self.autor2.place (x=78,y=235)

self.Data.place (x=10, y=140)
self.Data_in.place (x=10, y=155)

self.root.mainloop()

def updateGUI(self):
    self.root.update()
    self.Velocidad2.after (150, self.updateGUI)

def readSensor(self):
    self.Velocidad2["text"] = velocidad_data[-1]
    self.Latitud2["text"] = latitud_data[-1]
    self.Longitud2["text"] = longitud_data[-1]

    self.Data_in["text"] = serialdata[-1]
    self.root.update()
    self.root.after (527, self.readSensor)

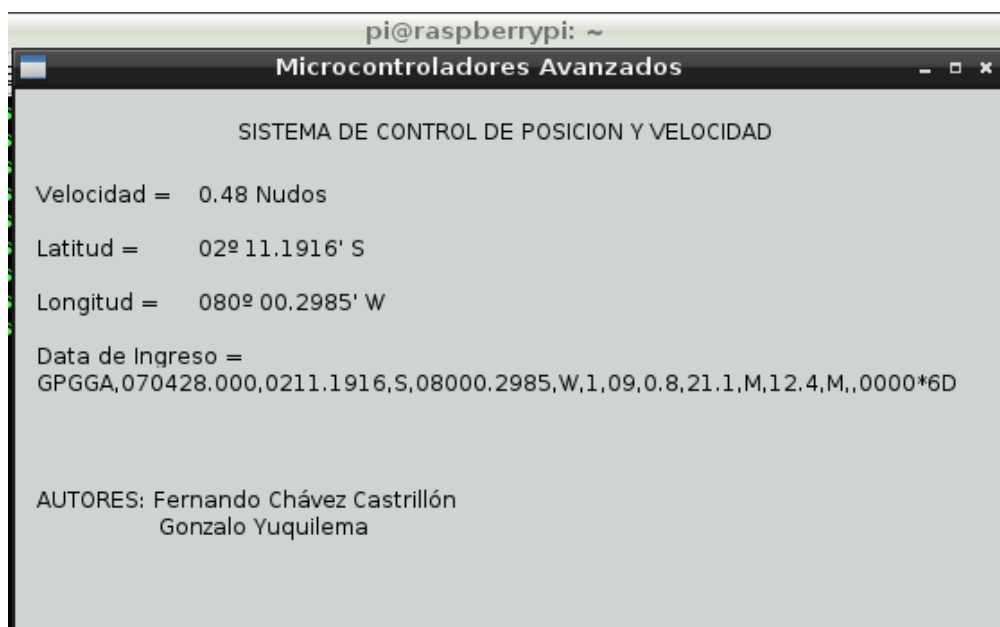
if __name__ == "__main__":
    SensorThread().start()
    Gui().run()

```

Una vez que se ejecuta el software, teniendo presente que el módulo GPS se encuentra entregando data válida al Raspberry Pi, se podrán visualizar los valores de velocidad y posición (Latitud y Longitud).

Debido a que el GPS entrega una serie de sentencias las mismas que son visualizadas una a la vez, se ha elaborado el código que permite verificar el código que se está analizando, manteniendo el valor anterior, hasta que éste cambie.

La pantalla donde se visualiza el ingreso de la data GPGGA, de la cual se ha tomado la información de posición se puede visualizar en la figura 3.15:



```
pi@raspberrypi: ~  
Microcontroladores Avanzados  
SISTEMA DE CONTROL DE POSICION Y VELOCIDAD  
Velocidad = 0.48 Nudos  
Latitud = 02° 11.1916' S  
Longitud = 080° 00.2985' W  
Data de Ingreso =  
GPGGA,070428.000,0211.1916,S,08000.2985,W,1,09,0.8,21.1,M,12.4,M,,0000*6D  
  
AUTORES: Fernando Chávez Castrillón  
Gonzalo Yuquilema
```

Figura No 3.15. Display que Visualiza Ingreso de Data GPGGA

Por otra parte, la pantalla donde se visualiza el ingreso de la data GPRMC, de la cual se ha tomado la información de posición se puede visualizar figura 3.16.

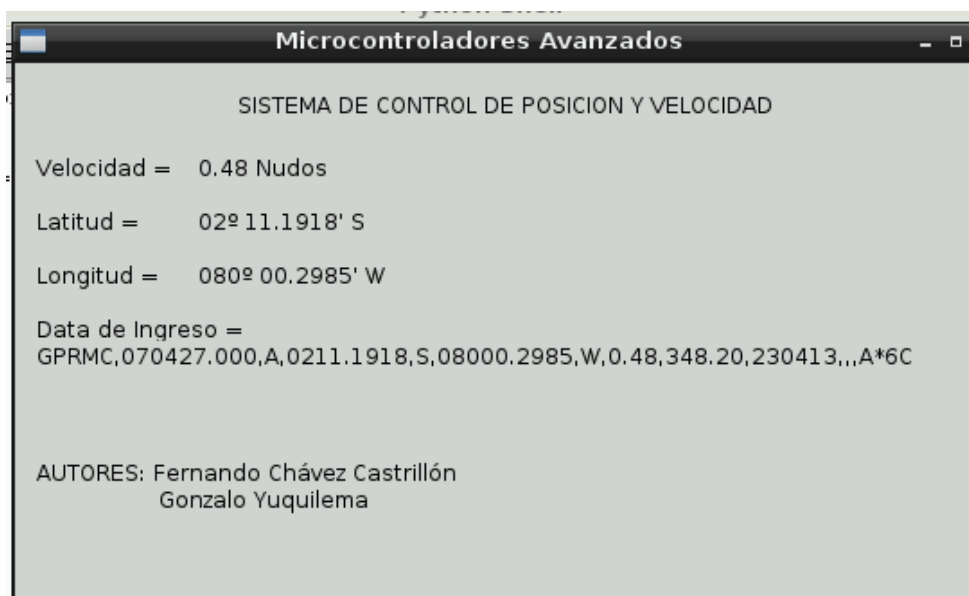


Figura No 3.16. Display que Visualiza Ingreso de Data GPRMC

Cabe mencionar que existen otras sentencias, de las cuales se puede tomar la información de posición y velocidad, como por ejemplo GPGLL y GPVTG, las mismas que debido a limitaciones del módulo GPS, no fue factible configurarlo desde Python y únicamente se lo consiguió habilitar desde el programa Cutecom, en el cual se puede colocar un delay a la transmisión de caracteres de 30 mseg.

CAPÍTULO 4

4 PRUEBAS Y SIMULACIONES

El presente capítulo presentarán los procedimientos de encendido e inicialización de los módulos y software utilizados para realizar el control de posición y velocidad. De igual forma se presentan las pruebas realizadas al software desarrollado en Python mediante la utilización del simulador de GPS. Finalmente se exponen las pruebas realizadas al módulo GPS, que fue adaptado al minicomputador Raspberry Pi e instalado en un vehículo para cumplir con el objetivo de este proyecto.

4.1 Procedimientos de Encendido e Inicialización.

Para llevar a cabo la adquisición de datos GPS utilizando el simulador o directamente desde el módulo GPS se deben realizar el siguiente procedimiento:

1. Se debe verificar las alimentaciones necesarias para realizar el proceso de encendido de los equipos. Para el caso del simulador, el voltaje requerido es de 115 V ACC, mientras que el Raspberry Pi requiere una alimentación de 5 Vdc. Para las pruebas que se realizarán en el auto se utilizará la alimentación desde el encendedor de cigarrillos.
2. Para el caso del simulador no se requiere la instalación de una antena, sin embargo para la prueba con el módulo GPS, se requiere que la antena se encuentre en un lugar donde reciba las señales entregadas por los satélites. Si la antena va a ser colocada al interior del auto, ésta debe estar ubicada en las cercanías del parabrisas.
3. Una vez que se realice la conexión del simulador o del módulo GPS, se debe verificar que el software Cutecom se encuentra recibiendo correctamente la información, para lo cual se deberá tener precaución de la velocidad de transmisión de datos, así como la dirección del dispositivo.
4. Luego de haber probado la comunicación con el dispositivo GPS, se procede a ejecutar en el caso del módulo GPS, el software de monitoreo GPSD, el mismo que permite visualizar en tiempo real la información recibida por los satélites. También se puede realizar las pruebas con el software desarrollado en Python.
5. En el caso del Simulador, se requiere la utilización del software desarrollado en Python, tomando la precaución de primero ejecutar el programa y luego el simulador para evitar que el buffer del simulador se sature de datos que son enviados mediante la secuencia FIFO.

4.2 Pruebas realizadas con el Simulador GPS.

Siguiendo las instrucciones antes indicadas y luego de realizar la conexión adecuada del simulador al mini computador al Raspberry procedemos a realizar la simulación de la data para verificar que efectivamente estamos recibiendo la información y ésta es procesada por nuestro software.

Se debe recordar que para la prueba con el simulador no se puede ejecutar el software GPST, por cuanto éste necesita información de los satélites para verificar si la data envía es válida o no.

La conexión del Simulador, hacia el Raspberry se lo va a realizar mediante el puerto RS-232, razón por la cual se debe verificar los pines de este simulador, los mismos que irán a un convertidos RS232 a USB. A continuación una figura de la conexión del simulador.

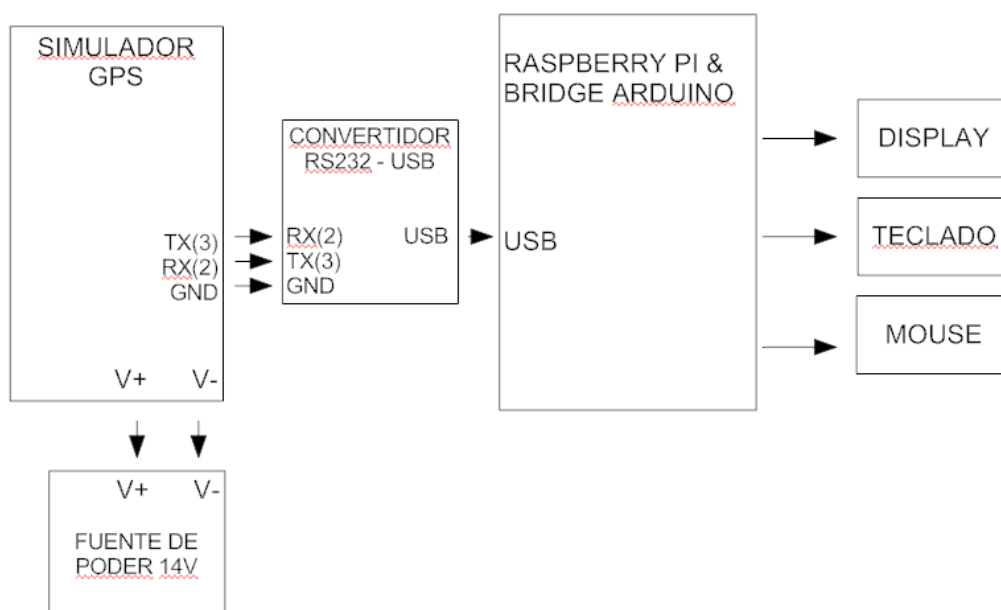


Figura No. 4.1: Diagrama en Bloques de Conexión del Simulador de GPS

Este diagrama en bloques ha sido implementado y en la figura 4.2 se puede visualizar la conexión de cada uno de sus componentes.

Cabe mencionar que este simulador de GPS, puede también funcionar como un GPS normal, por lo que se debe seleccionar en el menú respectivo la función de simulación para que el mismo entregue la data requerida.

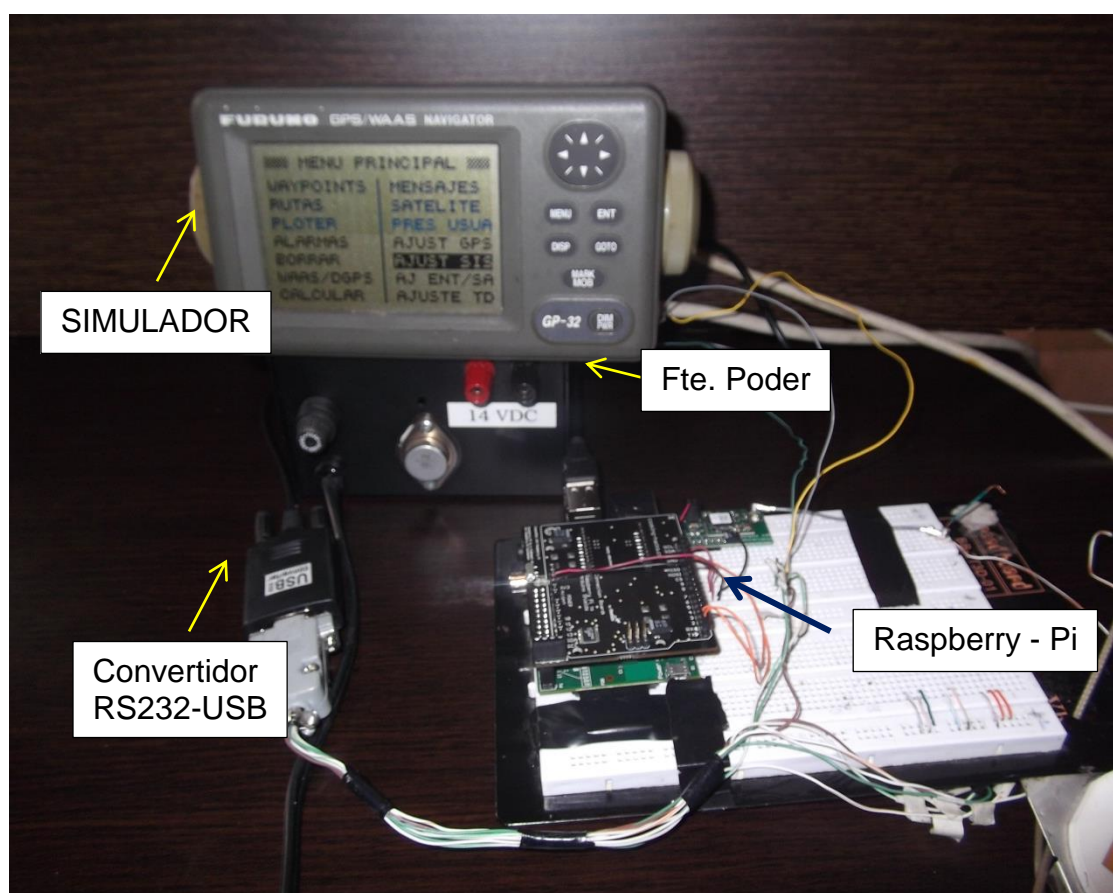


Figura No. 4.2: Interconexión Física del Simulador de GPS.

Una vez que se accede a la función simulación se puede variar los valores de Posición, Rumbo y Velocidad. En las figuras 4.3 y 4.4 se puede apreciar la

secuencia de selección del modo simulación, así como también la pantalla para selección de posición, rumbo y velocidad.



Figura No. 4.3: Pantalla para Selección del Modo de Simulación del GPS.



Figura No. 4.4 Pantalla para Cambio de Valores de Rumbo, Velocidad y Posición.

Una vez que se ha realizado la interconexión física del GPS y se ha seleccionado correctamente la función de simulación y se han introducido los

valores que deseamos monitorear, ejecutamos el programa Cutecom para verificar si se está recibiendo correctamente la data enviada desde el simulador.

Se debe tener precaución en seleccionar el puerto respectivo, para lo cual en este caso particular se encuentra asignado en la ruta `//dev/ttyUSB0`. La selección de la velocidad de transmisión y configuración de la data recibida, se la hace en los respectivos cuadros de selección. En la figura 4.5 se puede visualizar lo antes indicado:

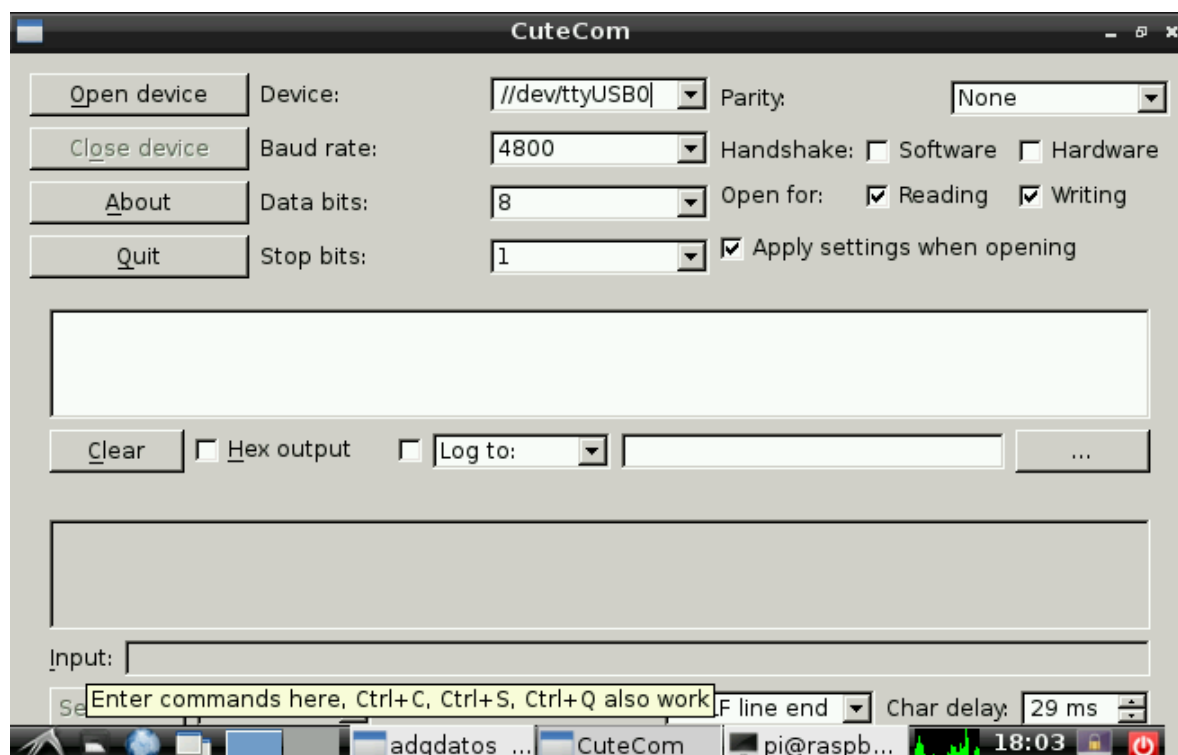


Figura No. 4.5 Configuración del Software Cutecom para Simulación.

Una vez que se realiza los pasos antes indicados se procede a realizar la apertura del puerto, mediante la activación "Open device". Una vez presionado este botón se empieza a recibir la data. En caso que no se reciba la data

verificar si no está intercambiado el cable de Rx y Tx o si se encuentra desconectada la tierra.

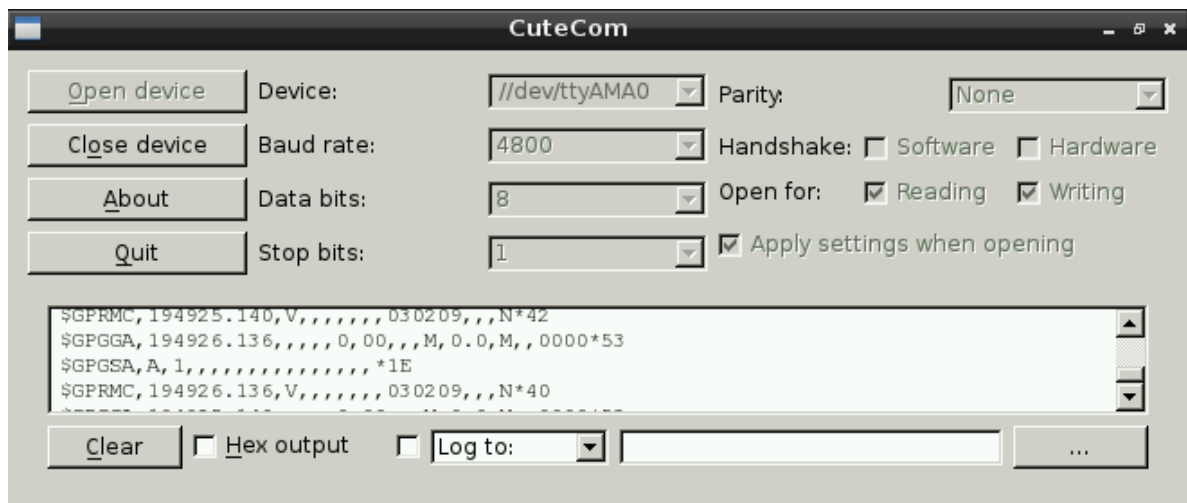


Figura No. 4.6 Visualización de Data Recibida en Software Cutecom

Luego de verificar que estamos recibiendo la data correctamente, procedemos a ejecutar el software desarrollado en Python para visualizar los datos de posición y velocidad.

En las figuras 4.7 y 4.8 se puede visualizar que la data que es enviada por el simulador se ve reflejada en el software de adquisición de datos.

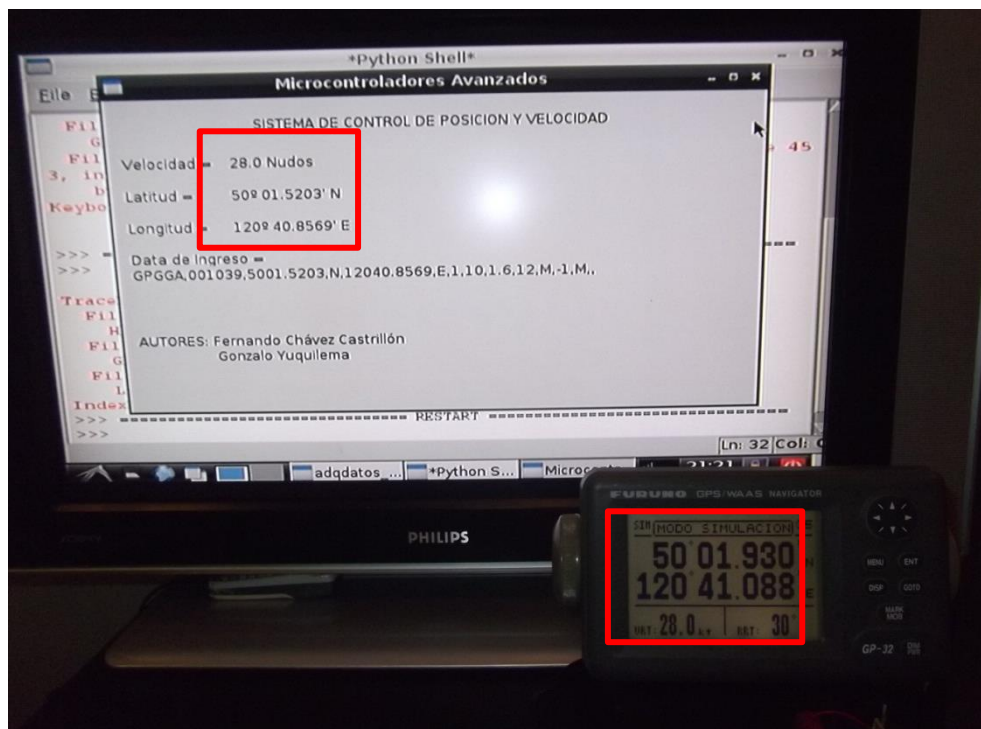


Figura No. 4.7 Visualización de datos Entregados por el Simulador y Recibidos en Software Python (Ejemplo No.1).

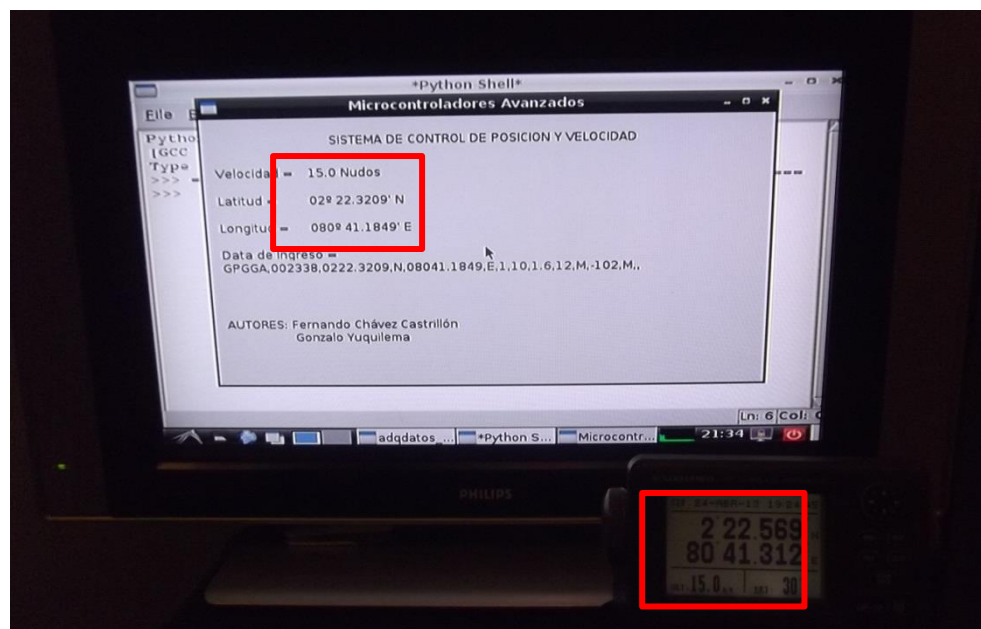


Figura No. 4.8 Visualización de datos Entregados por el Simulador y Recibidos en Software Python (Ejemplo No.2).

4.3 Pruebas realizadas con el módulo GPS en un vehículo.

Una vez que se ha realizado las pruebas con el simulador, procedemos a instalar el dispositivo en un vehículo para monitorear la velocidad y posición del auto en tiempo real. El poder de alimentación para el Raspberry Pi, se lo toma del encendedor de cigarrillos, teniendo la precaución de conectar el dispositivo una vez que el auto se encuentra arrancado, para evitar que existan picos de corriente que pueden dañar al dispositivo. En las figuras 4.9, 4.10 y 4.11, se puede ver la instalación del dispositivo en el automóvil.

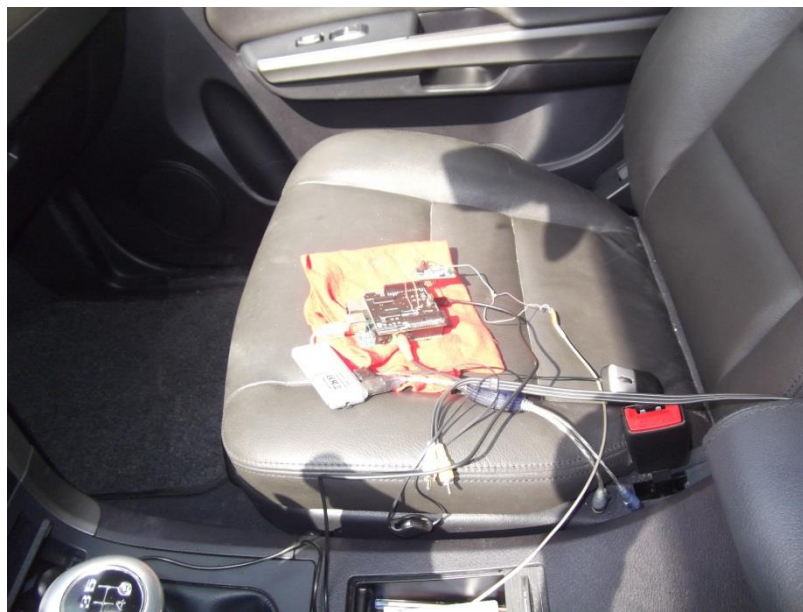


Figura No. 4.9 Conexión del Raspberry-Pi en el vehículo.



Figura No. 4.10 Instalación de antena en el interior del vehículo.



Figura No. 4.11 Instalación de display en el interior del vehículo.

Como se puede visualizar la antena es colocada detrás del parabrisas, el mismo que tiene la transparencia necesaria para captar las señales de los satélites.

Posteriormente ejecutamos el programa Cutecom para visualizar si ha sido reconocido el módulo GPS. Se debe tener precaución de seleccionar bien el puerto mediante el cual se recibirá la data. Mencionado puerto es //dev/ttyAMA0. A continuación una gráfica de lo antes indicado:



Figura No. 4.12 Adquisición de data del módulo GPS mediante el Software Cutecom.

Como se puede visualizar, el dispositivo se encuentra conectado y enviado información al Raspberry PI, sin embargo no está enviando datos de posición y velocidad del blanco, esto es debido a que el dispositivo toma alrededor de 10 minutos para procesar la información de los satélites y entregar la información requerida.

Una vez que estamos seguros que estamos conectados al módulo GPS, verificamos el estado de los satélites mediante el software GPST, el cual lo ejecutamos desde el terminal y podemos determinar, que existen satélites

detectados, sin embargo todavía no son utilizados por el módulo para entregar los valores de posición y velocidad. A continuación una gráfica de lo antes mencionado.

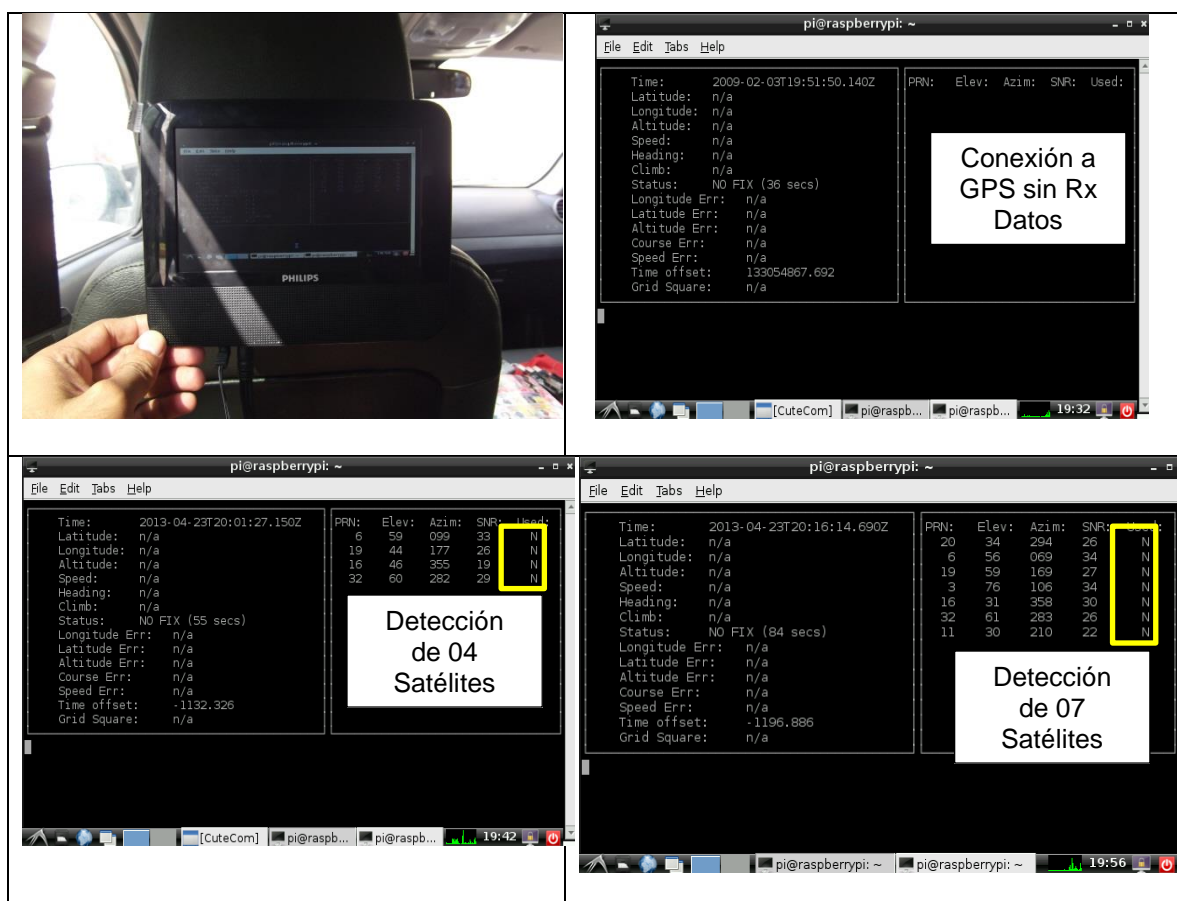


Figura No. 4.13 Adquisición de datos del módulo GPS mediante el Software GPSD.

Después de transcurrido cerca de 10 minutos las señales recibidas de los satélites son procesadas por el módulo GPS para entregar valores ciertos de posición y velocidad, objetivos de este proyecto, sin embargo también se recibe información complementaria que puede ser utilizada en otro tipo de proyecto. A continuación una gráfica de la obtención de la data real desde el módulo GPS, una vez que el vehículo se encuentra detenido.

```

pi@raspberrypi: ~
File Edit Tabs Help

Time:      2013-04-23T20:50:54.000Z
Latitude:  2.186500 S
Longitude: 80.005152 W
Altitude:  9.5 m
Speed:     0.1 kph
Heading:   102.2 deg (true)
Climb:     -7.4 m/min
Status:    3D FIX (319 secs)
Longitude Err: +/- 10 m
Latitude Err: +/- 8 m
Altitude Err: +/- 27 m
Course Err: n/a
Speed Err: +/- 78 kph
Time offset: -1196.645
Grid Square: EI97xt

PRN:  Elev:  Azim:  SNR:  Used:
 32   52    229    20    Y
  6   56    055    26    Y
 19   63    165    18    Y
 31   11    063    21    Y
  3   79    076    24    Y
 20   36    262    19    Y
 23   26    336    31    Y
 11   26    196    25    Y
  1   14    211    14    Y
 16   28    003    30    Y

```

Figura No. 4.14 Adquisición de datos del módulo GPS mediante el Software GPSTool – Auto Detenido.

Como se puede visualizar en la figura anterior los datos entregados por los satélites son procesados correctamente y existe presentación de valores de posición, velocidad y otros datos. Se puede visualizar que el auto está detenido por cuanto la velocidad indicada es 0.1 Kph.

Finalmente, lo que queda pendiente es poner el vehículo en movimiento para determinar que efectivamente se ha cumplido con el objetivo del proyecto que consistía en realizar el control de posición y velocidad de móvil. Se presentarán varias fotografías donde se visualiza el velocímetro del vehículo y se realiza una comparación con el valor entregado por el software GPSTool.

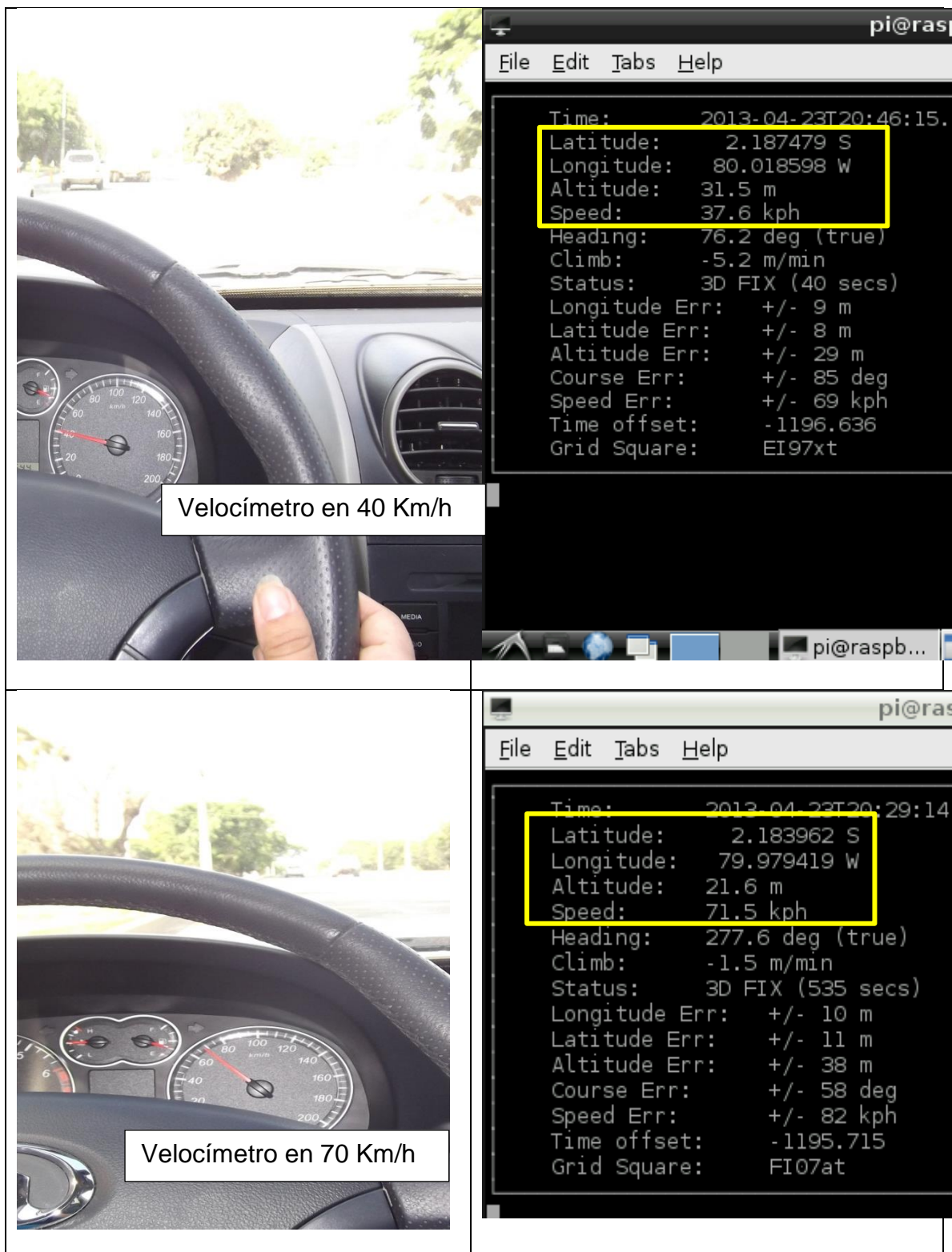
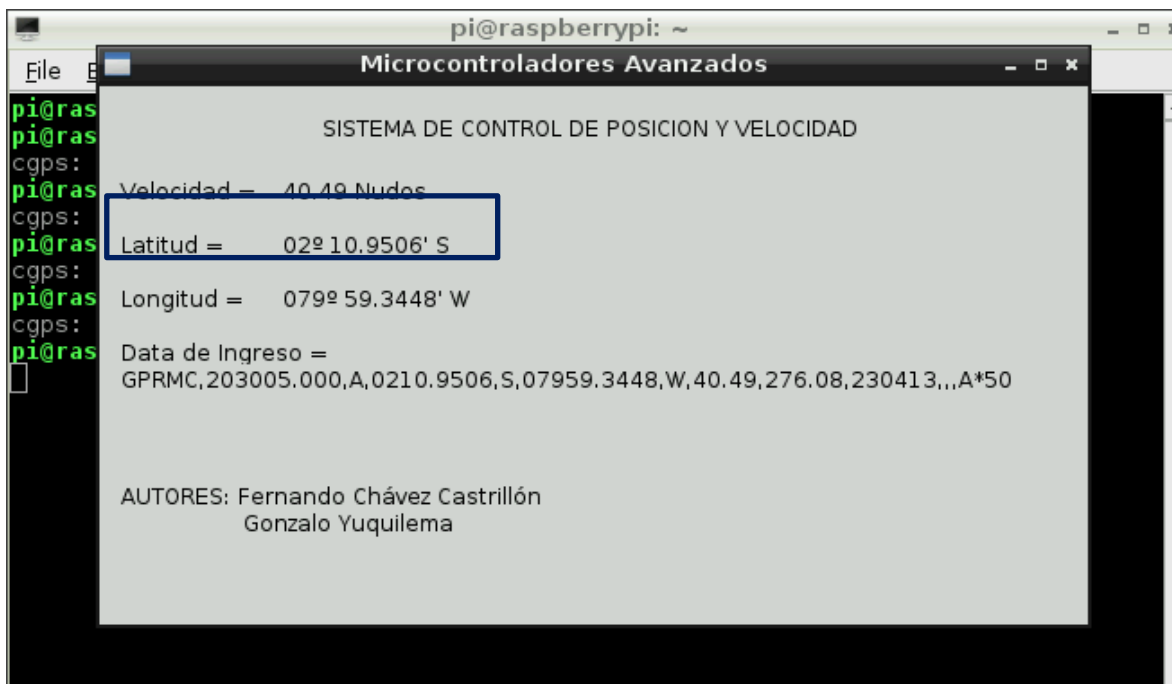




Figura No. 4.15 Adquisición de datos del módulo GPS mediante el Software GPSD – Auto en Movimiento.

De igual manera se hicieron pruebas reales con el software desarrollado en Python, obteniendo resultados similares a los presentados anteriormente. Cabe resaltar que para realizar comparaciones con velocímetro es necesario mantener la velocidad al menos un minuto para que el módulo pueda procesar correctamente la información enviada por los satélites.



```
pi@raspberrypi: ~  
File Edit Shell  
Microcontroladores Avanzados  
SISTEMA DE CONTROL DE POSICION Y VELOCIDAD  
Velocidad = 40.49 Nudos  
Latitud = 02° 10.9506' S  
Longitud = 079° 59.3448' W  
Data de Ingreso =  
GPRMC,203005.000,A,0210.9506,S,07959.3448,W,40.49,276.08,230413,,A*50  
  
AUTORES: Fernando Chávez Castrillón  
Gonzalo Yuquilema
```



Figura No. 4.16 Adquisición de datos del módulo GPS mediante el Software Desarrollado en Python – Auto en Movimiento.

Finalmente se presenta la fotografía del proyecto que ha sido implementado, una vez que ha superado con éxito las pruebas realizadas de control de posición y velocidad de un móvil.

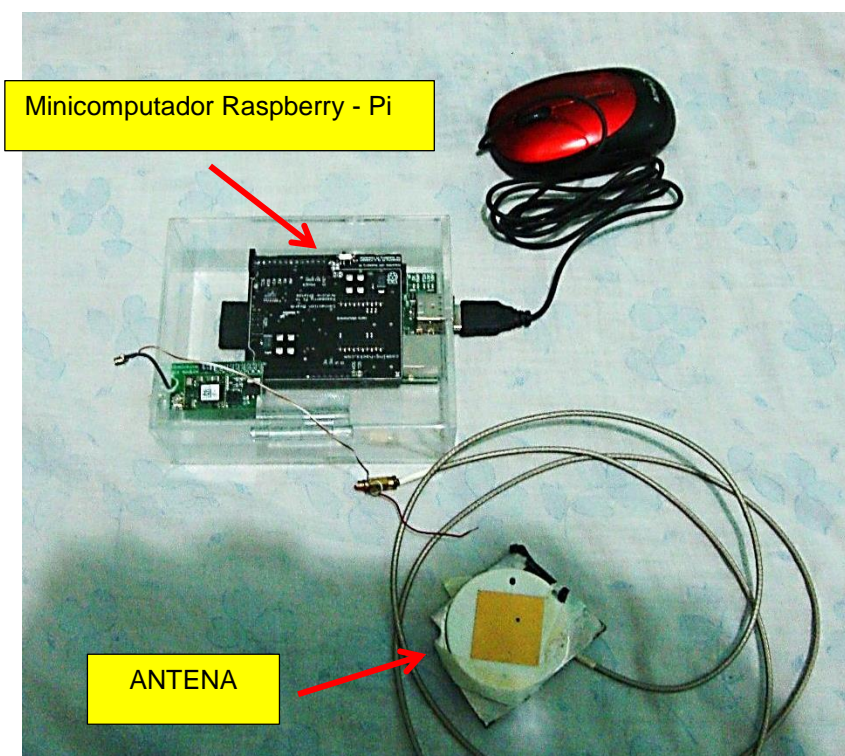


Figura No. 4.17 Proyecto Implementado para control de posición y velocidad de un móvil mediante el uso de un Micocomputador Raspberry Pi

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1. El minicomputador Raspberry Pi, utilizado para realizar la adquisición de datos en tiempo real, por su gran versatilidad ha facilitado la instalación del módulo GPS CSC3, del software especializado GPS Daemon y del programa desarrollado en el lenguaje de programación Python, herramientas utilizadas para realizar el control de la velocidad y posición de un vehículo en movimiento, objetivo principal de esta tesina.
2. La exactitud de los datos de posición y velocidad que son calculados por el módulo GPS CSC3, depende del número de señales válidas que son recibidas desde los satélites, siendo necesario al menos tres satélites para obtener una posición confiable del blanco, por lo tanto mientras más satélites se encuentren disponibles, más exactos serán los valores entregados por mencionado módulo.
3. A pesar que el módulo GPS CSC3 se encuentre recibiendo información correcta desde los satélites, es necesario que el móvil se mantenga a

velocidad constante, a fin de que este dispositivo pueda calcular de manera exacta la velocidad de desplazamiento, en base a las señales que son recibidas desde los satélites.

4. Dentro de la etapa de desarrollo del proyecto, fue muy importante la utilización del simulador de GPS, el cual permitió validar el funcionamiento del software de adquisición de datos y del procesamiento de la información, por cuanto se pudo controlar y verificar la data enviada, asegurando de esta forma el éxito en la integración del minicomputador Raspberry Pi con el módulo GPS CSC3.
5. Debido a que el minicomputador Raspberry Pi fue desarrollado para ser utilizado con el Sistema Operativo Linux y al existir programadores que promueven mediante aportes gratuitos el uso de este sistema operativo, abre las puertas para que en un futuro cercano mediante el uso de este dispositivo se encuentren soluciones a bajo costo tanto para la industria como para el uso diario de las personas.

RECOMENDACIONES

1. Se debe tener precaución al momento de interconectar dispositivos a través de los puertos UART y GPIO del Minicomputador Raspberry, por cuanto estos puertos están diseñado para recibir voltajes de 3.3 voltios, en caso de conectar voltajes de mayor valor pueden afectar o dañar al minicomputador.
2. Se recomienda que la instalación del módulo electrónico se lo realice en una ubicación, la cual permita que la antena del GPS tenga una vista al cielo, lo que facilitará la triangulación. Además, se recomienda que su instalación no se realice en un lugar cerrado por metales, debido a que un sitio así produciría una jaula de Faraday, lo que provocaría el no funcionamiento del módulo GPS.
3. Debido a que el Minicomputador Raspeberry Pi está concebido para ejecutar tareas puntuales, se recomienda no recargar a este dispositivo con aplicaciones que se ejecuten al mismo tiempo, ya que esto puede elevar la temperatura del procesador y debido a que el Raspberry no cuenta con un dispositivo disipador de calor, ésto puede dañar de manera permanente e irreversible al minicomputador.
4. Previo a la conexión de cualquier tipo de hardware se debe verificar el datasheet de cada uno de estos componentes, para evitar malas conexiones que pueden dañar a los componentes, así como también se debe asegurar que las tierras de cada uno de estos se encuentren interconectadas para evitar diferenciales de potencial que pueden de igual forma afectar a los mismos.

BIBLIOGRAFÍA

- [1] Eben UPTON y Gareth HALFACREE, Raspberry Pi – User Guide, Editorial Wiley, Inglaterra, 2012.
- [2] Wikipedia.com, Raspberry Pi, http://es.wikipedia.org/wiki/Raspberry_Pi, Fecha de Consulta: 16/abril/2013.
- [3] Fundación Raspberry Pi Raspberry Pi, Quick Start Guide, <http://www.raspberrypi.org/wp-content/uploads/2012/12/quick-start-guide-v1.1.pdf>, Fecha de consulta: 19/abril/2013.
- [4] National Marine Electronics Association (NMEA), Protocolo NMEA 0183, <http://www.nmea.org/>, 2001, Fecha de consulta: 22/abril/2013.
- [5] VINCOTECH, GPS Firmware GSC3-Bases Product- User Manual, VICONTECH, 2009.
- [6] Simon MONK, Programming the Raspberry Pi – Getting Started With Python, Editorial Mc Graw Hill, New York, 2013
- [7] Karl WRIGHT, Robert CRUSE y Paul KINGETT, The Raspberry Pi – Educational Manual, Inglaterra, 2012.
- [8] FURUNO Electric Co. LTD., Operator´s Manual “GPS NAVIGATOR Model GP-37/G-32, FURUNO, Nishinomiya City – Japón, 2010.