UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y COMUNICACIONES

Tema:

"Analizador de Espectro con Hardware Libre"

Trabajo de Graduación Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Electrónica y Comunicaciones

SUBLINEA DE INVESTIGACION: Procesamiento Digital de Señales

AUTOR: Freddy Daniel Carrillo Bustos.

TUTOR: Ing. Mg. Manzano Villafuerte Víctor Santiago

Ambato – Ecuador
Julio 2015

APROBACIÓN DEL TUTOR

En mi calidad de tutor del Trabajo de Investigación sobre el tema: “Analizador de Espectro con Hardware Libre”, del señor Freddy Daniel Carrillo Bustos, estudiante de la Carrera de Ingeniería en Electrónica y Comunicaciones de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad con el numeral 7.2 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ambato Julio, 2015

EL TUTOR

_____

Ing. Mg. Santiago Manzano

AUTORÍA

El presente trabajo de investigación titulado “Analizador de Espectro con Hardware Libre”, es absolutamente original, auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor

Ambato, Julio del 2015

_____

Freddy Daniel Carrillo Bustos

C.I.: 1804487823

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación, con fines de difusión pública, además autorizo su reproducción dentro de las regulaciones de la Universidad.

Ambato julio, 2015

Freddy Daniel Carillo Bustos

CC: 1804487823

APROBACIÓN DE LA COMISIÓN CALIFICADORA

La Comisión Calificadora del presente trabajo conformada por los señores docentes Ing. Mg. Juan Pablo Pallo, e Ing. Mg. Marco Jurado, revisó y aprobó el informe Final del trabajo de graduación titulado: “Analizador de Espectro con Hardware Libre”, de acuerdo al numeral 9.1 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ing. José Vicente Morales Lozada, Mg.
PRESIDENTE DEL TRIBUNAL

Ing. Juan Pablo Pallo, Mg
DOCENTE CALIFICADOR

Ing. Marco Jurado, Mg
DOCENTE CALIFICADOR

AGRADECIMIENTO

Agradezco a mi padre y a mi madre por haberme apoyado siempre que lo he necesitado, a mi padre por haberme inculcado honestidad y ética, a mi madre por orientarme y aconsejarme.

A mi novia, compañera y amiga, por estar siempre ahí.

Y a todas aquellas personas que me han inspirado para tratar de ser mejor cada día, sin abandonar mi pasión ni mis valores.

Freddy Daniel Carrillo Bustos

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En el presente proyecto se diseñó e implementó un Analizador de Espectro con hardware libre a un bajo costo, cuyo objetivo es facilitar la realización de prácticas y complementar el aprendizaje teórico de los estudiantes y aficionados a la Electrónica.

El proyecto tiene como parte central la placa Raspberry PI que es un computador de tamaño reducido, capaz de ejecutar un sistema operativo basado en Linux; posee puertos de alto y bajo nivel que le permiten interactuar con varios tipos de hardware.

La información es recolectada a través de una tarjeta de adquisición de datos que utiliza el conversor análogo digital MCP3008, la comunicación se realiza a través del puerto GPIO de la placa Raspberry Pi utilizando la interface de puerto serial SPI.

El software está desarrollado en Python, empleando principalmente los módulos PyQtGraph y Spidev, la máxima frecuencia de muestreo alcanzada es de 20KHz y la frecuencia de operación final es 8KHz, estas limitaciones se producen por el modo de funcionamiento del Kernel Linux

ABSTRACT

In this project was designed and implemented an open hardware Spectrum Analyzer at a low cost, which aims to facilitate the practices and so complement the theorical learning of electronical students and hobbyist.

The project main part is Raspberry Pi, which is a tiny size computer, capable of running Linux based distributions; in addition it has high and low level ports that allow you to interact with various types of hardware.

The information is collected through a data acquisition card that uses the digital to analog converter MCP3008, the communication is via GPIO port of Raspberry Pi using the Serial Port Interface SPI.

The software is developed in Python, mainly using the PyQtGraph and Spidev modules, the maximum sample rate is 20KHz and the final reached frequency is 8KHz; this happens because of limitations of Linux kernel.

INTRODUCCIÓN

En el presente trabajo de investigación se realizó el diseño e implementación de una Analizador de Espectro utilizando hardware libre como una alternativa económica orientado a estudiantes de carreras afines a la Electrónica; a continuación se detalla los procedimientos seguidos en cada capítulo.

En el primer capítulo se explican las necesidades de las Universidades Ecuatorianas en el ámbito de equipamiento de laboratorios de Electrónica, también se exponen las causas para crear el equipo planteado en beneficio de los estudiantes.

En el segundo capítulo se presentan antecedentes investigativos sobre trabajos relacionados con el tema propuesto, también se detallan los siguientes temas, Analizador de Espectro y tipos, Transformada de Fourier y Hardware.

En el tercer capítulo, se presenta la modalidad de investigación y los pasos utilizados para el desarrollo del proyecto,

En el cuarto capítulo se detalla el diseño y desarrollo de un Analizador de Espectro con hardware libre, la selección de componentes, la utilización de Python con el módulo PyqtGraph y las pruebas de funcionamiento para determinar los rangos de operación

En el quinto capítulo se establecen las conclusiones y recomendaciones

CAPÍTULO I

EL PROBLEMA

1.1Tema

Analizador de Espectro con Hardware Libre.

1.2 Planteamiento del Problema

Según datos publicados por la Secretaria Nacional de Educación Superior Ciencia, Tecnología e Innovación, el presupuesto promedio de las Universidades Ecuatorianas es 50 millones de dólares [1].

En el caso de la Universidad Técnica de Ambato el presupuesto referente a los años 2011 y 2013 es 39,5 y 40,7 millones de dólares respectivamente, un porcentaje de este dinero es destinado a la adquisición de “Maquinarias y Equipos” por cada facultad. La Facultad de Ingeniería en Sistemas Electrónica e Industrial destino en el presente año 42.000,32$ para equipamiento de laboratorios [62,63].

Las carreras universitarias orientadas a la electrónica requieren laboratorios con un mínimo equipamiento para funcionar apropiadamente, a pesar de los esfuerzos realizados por Facultades y Universidades en la adquisición de equipos, estos tienen una vida útil corta debido a las largas jornadas de trabajo que deben soportar, lo cual los deja obsoletos en poco tiempo y sin muchas posibilidades de reparación.

La escasez de equipos ocasiona grandes problemas en el desarrollo de las actividades académicas de estudiantes y profesores, quienes se ven obligados a improvisar soluciones para solventar estas carencias

La Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato también presenta estos inconvenientes debido al ajustado presupuesto que tiene para la renovación o adquisición de equipos.

Los estudiantes o aficionados a la Electrónica tienen limitaciones en la adquisición de equipos por el alto costo y conseguirán solo equipos básicos que estén a su disposición económica.

Los factores mencionados anteriormente afectan directamente la calidad de la formación académica, en un área técnica cuyo complemento vital es la realización de prácticas paralelas a la instrucción teórica; las cuales permiten lograr un desarrollo integral de los estudiantes, brindando de este modo los conocimientos necesarios para desenvolverse en el campo laboral.

1.3 Delimitación

Delimitación de contenidos

Área Académica: Electrónica y Comunicaciones

Línea de Investigación: Tecnologías de Comunicación

Sublínea: Procesamiento Digital de Señales

Delimitación Espacial

La presente investigación se realizó en la ciudad de Ambato.

Delimitación Temporal

El proyecto de investigación se desarrolló en un plazo de un año a partir de la aprobación del Honorable Consejo Directivo de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

1.4 Justificación

El interés por desarrollar este proyecto radica en la necesidad de contar con equipos de bajo costo para los estudiantes y mejorar los laboratorios de la Facultad de Ingeniería en Electrónica y Comunicaciones de la Universidad Técnica de Ambato, para la realización de prácticas y así complementar los conocimientos adquiridos dentro de las aulas.

El proyecto busca incentivar el análisis y el diseño electrónico aplicado, al proveer un equipo compuesto de secciones modulares que funciona como: Analizador de Espectro. Este dispositivo es un instrumento necesario en cualquier laboratorio donde se requiere analizar señales eléctricas en el dominio del tiempo o la frecuencia.

El diseño realizado permite reparar o reutilizar de manera total o parcial el dispositivo en caso de cualquier avería, esto es útil para los estudiantes principiantes quienes en ocasiones cometen errores al realizar sus prácticas de laboratorio.

Los beneficiarios directos del proyecto son los estudiantes del área de Electrónica, que tendrán a su disposición un equipo alternativo para la medición del espectro eléctrico.

1.5 Objetivos

1.5.1 Objetivo General

- Implementar un analizador de espectro con hardware libre.

1.5.2 Objetivos Específicos

- Definir el tipo de Analizador de Espectro y su construcción
- Establecer los requerimientos del Analizador de Espectro en hardware y software.
- Desarrollar la interfaz del Analizador de Espectro por software
- Diseñar el hardware del Analizador de Espectro

CAPÍTULO II
MARCO TEÓRICO

2.1 Antecedentes Investigativos.

Al realizar una investigación bibliográfica en repositorios digitales se encontraron los siguientes temas afines a la propuesta de solución:

En la Escuela Politécnica Nacional del Ecuador, Facultad de Ingeniería Eléctrica y Electrónica en el año de 1977 el Ingeniero Jorge Doring Humeres, realizó la implementación de un "Analizador de espectro de Audiofrecuencias" utilizando un circuito más simple que el encontrado en equipos comerciales y por tanto de menor precio [2]."

En la Universidad Tecnológica de la Mixteca, en México 1996, el Ingeniero Ildefonso Hernández Martínez realizo el diseño e implementación de un "Analizador Dinámico de Señales" en base a una tarjeta de adquisición de datos y un conjunto de librerías construidas en C++ [3].

En la Escuela Politécnica Nacional del Ecuador, Facultad de Ingeniería Eléctrica y Electrónica en el año 2000 el Ingeniero Merchán Gavilánez Pedro Ángel realizó el diseño y la implementación del tema "Sistema de Adquisición de Datos para convertir a un Computador Personal en un Analizador de Espectros", desarrollado

para incorporar el análisis en el dominio del tiempo y la frecuencia en las prácticas de laboratorio de dispositivos electrónicos. [4].

En la Escuela Politécnica Nacional del Ecuador, Facultad de Ingeniería Eléctrica y Electrónica en el año 2002 el Ingeniero Vilatuña Arellano, Francisco Javier realizó el diseño e implementación de un "Analizador de Espectros Utilizando la transformada rápida de Fourier en un microprocesador DSP" mediante la implementación de un algoritmo para el cálculo de la DFT para señales de hasta 20KHz, mediante un control electrónico de los parámetros de visualización y calculo [5].

En la Universidad Complutense de Madrid, Facultad de Informática, en el año 2005 se realizó el proyecto bautizado como CocleoMatic que es un Analizador de un Espectro musical cuyos autores son: Andrés Jiménez Sánchez, Amaia de Miguel Morate y Javier Villellas Bernal. Su objetivo fue crear un Analizador de Espectro de sonidos para realizar una simulación realista del oído humano, mediante la transformada discreta de Fourier [6].

En la Universidad de San Carlos de Guatemala, Facultad de Ingeniería, Escuela Mecánica Eléctrica en el año 2006, el Ingeniero Héctor Leonel Munguía Valiente se realizó la "Implementación de un Analizador de Espectro para Frecuencias de Audio Utilizando un Procesador Digital de Señales (DSP)"[7].

En el Instituto Politécnico Nacional, Escuela Superior de Ingeniería Mecánica y Eléctrica, en México D.F. 2008, el Ingeniero en Comunicaciones y Electrónica Alejandro Soberanis Garfias realizó "Diseño y Construcción de un Analizador de Espectros Usando una Plataforma Basada en FPGA" para su uso en bajas frecuencias, con la finalidad de realizar mediciones de armónicos en instalaciones Eléctricas, con el propósito de evitar el deterioro de las mismas por malfuncionamiento o sobrecargas [8].

En la Universidad de Cuenca, Facultad de Ingeniería Electrónica, en el año 2009 los estudiantes: Arichábala Soto Javier Egidio y Zea Vintimilla, Renato Patricio, realizaron el diseño e implementación de un Sistema de Análisis de Señales en el Dominio del tiempo para ser usado en una red sísmica, a fin de determinar las componentes espectrales de la misma. Este proyecto puede ser usado para varios ámbitos que requieran un análisis en el dominio de la frecuencia [9].

2.2 Fundamentación Teórica.

2.2.1. Espectro Electromagnético.

En el vacío las ondas electromagnéticas se mueven a la misma rapidez, y difieren entre sí por la frecuencia, la clasificación de las ondas electromagnéticas por su frecuencia es el espectro electromagnético. A continuacion en la Figura 2.1 se muestra la representación del espectro electromagnetico completo.

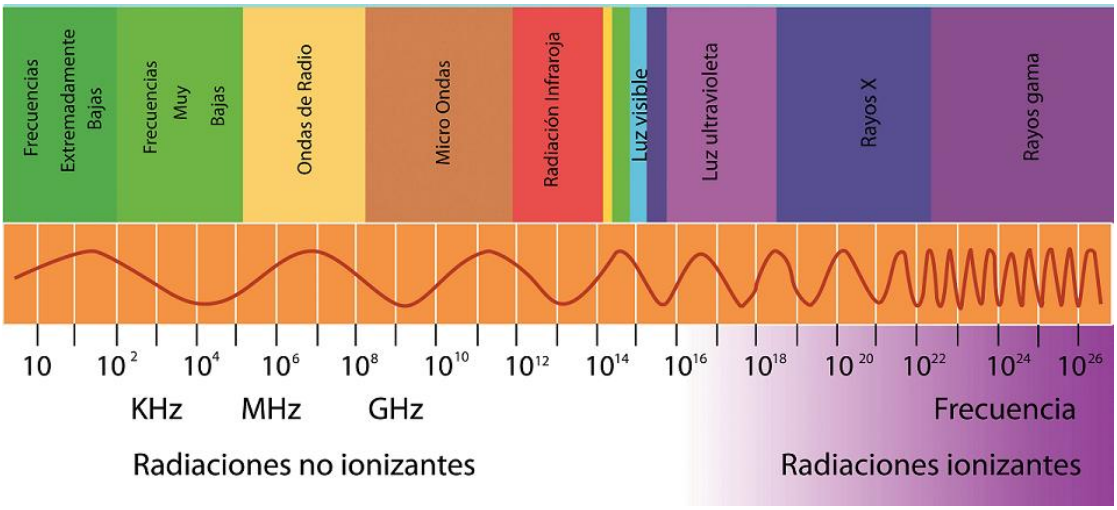


Figura 2.1. Espectro Electromagnético.

Fuente: http://www.mimagnetoterapia.com/espectro-electromagnetico

Se han detectado ondas electromagnéticas de frecuencia tan baja como 0.01 Hz, las ondas de varios miles de Hertz (KHz) se consideran ondas de radio de muy baja frecuencia; un millón de Hertz (MHz) está a la mitad del cuadrante de un radio de AM.

La banda de TV, de ondas de muy alta frecuencia (VHF) comienza en unos 50MHz y la radio de FM va desde los 88MHz a los 108MHz. Después vienen las frecuencias ultra altas (UHF), seguidas de las microondas, más allá de las cuales están las ondas infrarrojas, que con frecuencia se llaman ondas caloríficas; más adelante se encuentra la luz visible, que forma menos de la millonésima parte del 1% del espectro electromagnético medido. La luz de frecuencia mínima que podemos ver es el color rojo, las frecuencias máximas que podemos ver son de casi el doble [64].

2.2.2 Analizador de Espectro

El Analizador de Espectro es una herramienta capaz de representar componentes espectrales de una determinada señal a partir de su transformada de Fourier. Mide la magnitud de una señal de entrada en relación a su frecuencia, dentro de un rango seleccionado, como se ilustra en la figura 2.2. La representación en el dominio de la frecuencia permite visualizar las diversas componentes que difícilmente podrían ser apreciados al trabajar en el dominio del tiempo [10].

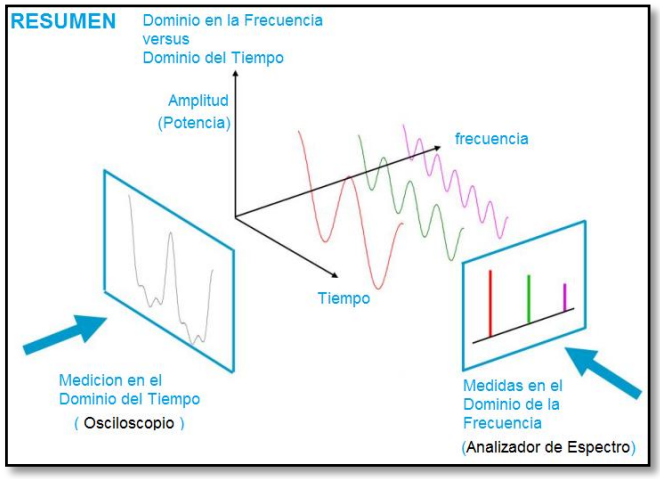


Figura 2.2. Dominio de la frecuencia y el tiempo.

Fuente: Miranda J., Sebastián J. Sierra M., Margineda J. Spectrum Analysis Back to Basics

2.2.3 Analizador de Espectro en Tiempo Real.

El Analizador de Espectro en tiempo real utiliza un divisor de potencia con múltiples salidas, que alimentan canales con filtros pasa banda, con un detector al término de cada uno, como se muestra en la figura 2.3. De esta manera cada canal realiza la detección de una frecuencia distinta, trabajando de manera simultánea, lo que resulta en un analizador de espectro en tiempo real. Tradicionalmente se empleó este diseño en el rango de frecuencias de audio, actualmente existen analizadores en tiempo real que cubren márgenes de frecuencia de DC hasta 3Ghz.

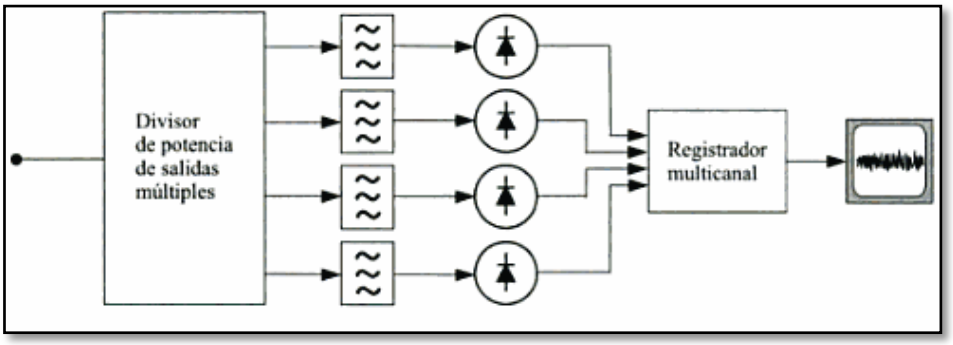


Figura 2.3. Esquema de un analizador de espectro en tiempo real

Fuente: Miranda J., Sebastián J. Sierra M., Margineda J. Spectrum Analysis Back to Basics

2.2.4 Analizador de Espectro ADS (Analizador Dinámico de Señales).

El Analizador Dinámico de Señales se muestra en la figura 2.4, es un instrumento digital que calcula numéricamente su espectro con ayuda de un microprocesador, utilizando técnicas basadas en la transformada de Fourier. Se obtiene información sobre la amplitud, la frecuencia y la fase de la señal.

Los analizadores dinámicos de espectro se utilizan generalmente a frecuencias bajas, como aplicaciones de audio (hasta unos 100Khz); dado que permiten hacer análisis en tiempo real de espectros variables con el tiempo, sin embargo,

recientemente se han hecho muy populares en ingeniería de microondas debido a que constituyen una opción competitiva para realizar medidas de ruido 1/f en transistores.

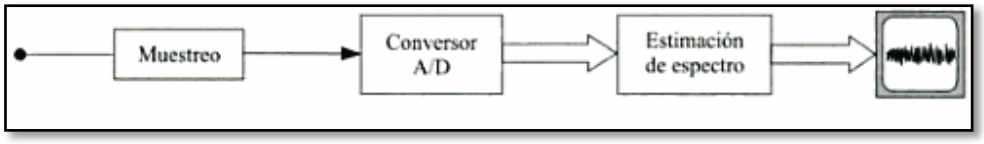


Figura 2.4. Esquema de un analizador de espectro dinámico

Fuente: Miranda J., Sebastián J. Sierra M., Margineda J. Spectrum Analysis Back to Basics

2.2.5. Analizador de Espectro de Filtro Sintonizado.

El Analizador de Espectro de filtro sintonizado dispone de un filtro pasa banda cuya frecuencia central puede ser ajustada mediante control Electrónico como se muestra en la figura 2.5, esto permite medir el espectro de la señal mediante un barrido. El control de esta frecuencia central se realiza mediante un generador de diente de sierra, el eje X representa un valor proporcional a la frecuencia.

Este dispositivo solo proporciona información sobre la magnitud del espectro y no sobre la fase, debido a ello el analizador de filtro sintonizado no permite reconstruir la señal en el dominio del tiempo. Esta clase de instrumentos es de bajo costo, pero sus prestaciones como se puede apreciar no son muy altas comparándolo con el analizador superheterodino.

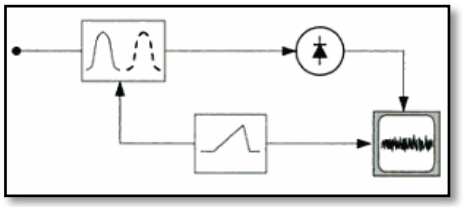


Figura 2.5. Esquema de un analizador de espectro de filtro sintonizado.

Fuente: Miranda J., Sebastián J. Sierra M., Margineda J. Spectrum Analysis Back to Basics

2.2.6 Analizador de Espectro Superheterodino.

El Analizador de Espectro Superheterodino es ampliamente utilizado en ingeniería de microondas, ya que alcanza frecuencias de hasta 300GHz, ofrece una gran resolución de frecuencia y opera en un régimen lineal con una amplitud de señal que supera los 10dB, este analizador solo proporciona información sobre la magnitud del espectro, de manera similar al Analizador de Espectro Sintonizado.

En la figura 2.6 se muestra un esquema básico del Analizador de Espectro Superheterodino. El barrido se realiza mediante el control electrónico de la frecuencia del oscilador local de la etapa de mezclado. La señal de diente de sierra se utiliza simultáneamente para el barrido horizontal y para controlar esta frecuencia. Al aumentar el voltaje de la onda el mezclador se sintoniza para frecuencias de entrada cada vez más altas, y al mismo tiempo la traza en la pantalla se desplaza de izquierda a derecha, generándose así la representación del espectro. En ausencia de señal, el trazo es esencialmente una línea recta contaminada por el ruido, se conoce como línea de base.

En la salida del mezclador se obtienen, entre otros, dos componentes de frecuencia de valores iguales a la suma y la diferencia de las dos frecuencias colocadas a la entrada del mezclador, las cuales son: la señal capturada y la señal del oscilador local. La frecuencia intermedia debe seleccionarse de entre ambas para que no se encuentre dentro de la banda que este midiendo el analizador [11].

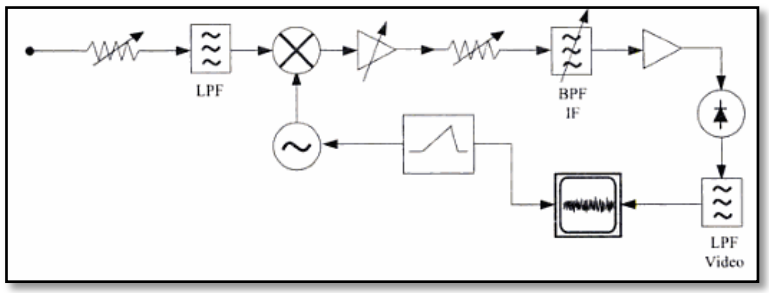


Figura 2.6. Analizador Superheterodino.

Fuente: Miranda J., Sebastián J. Sierra M., Margineda J. Spectrum Analysis Back to Basics

2.2.7 Transformada de Fourier

La integral o transformación de Fourier nos permite cambiar una función conocida $f(t)$ desde el dominio del tiempo, transformándola en una función $F(w)$ para ser apreciada en el dominio de la frecuencia angular.

La condición para que exista $F(w)$, es que la integral del valor absoluto de $f(t)$ debe ser finita, como se indica en la ecuación (1). La transformación de una función del dominio del tiempo al de la frecuencia tiene su base en la Transformada de Fourier y su inversa, que están definidas por la ecuación (2) y (3) respectivamente [12].

$$\int_{-\infty}^{+\infty} /f(t)/dt \ < \ \infty \qquad \text{Ec (1)}$$

$$S(w) = \int_{\infty}^{\infty} s(t) e^{-j2\pi ft} dt \qquad \text{Ec (2)}$$

$$s(t) = \frac{1}{2\pi} \int_{\infty}^{\infty} S(f) e^{j2\pi ft} df \qquad \text{Ec (3)}$$

Transformada de Fourier Discreta (DFT)

El cálculo de la transformada de Fourier de forma numérica mediante un computador requiere la discretización más la integración numérica. De este modo se consigue una aproximación mucho más cercana a la realidad (en términos matemáticos), de forma analítica se define la Transformada de Fourier en un ámbito digital, pero esto causa tres dificultades al momento de realizar el cálculo computacional:

- La discretización produce periodicidad en el dominio del tiempo y la frecuencia
- La integración numérica induce un error numérico de aproximación (es menos efectivo que la integración normal)
- El tiempo de duración es finito y por tanto no se cumplen las condiciones ideales, lo que ocasiona limitaciones de máxima frecuencia y resolución.

Transformada Rápida de Fourier. FFT

En 1965 Cooley y Tukey desarrollaron el algoritmo conocido como FFT (Fast Fourier Transform), el cual demostró adaptarse perfecta y eficientemente a la ejecución digital, lo cual redujo el tiempo necesario para calcular las transformadas por orden de magnitud [13].

Este algoritmo optimizado para calcular la DFT requiere extensivos cálculos y mucho tiempo de procesamiento, más aun si se tiene muchas muestras. Este algoritmo hace la suposición de que “n” es un entero múltiplo de 2, lo que permite ciertas simetrías reduciendo el número de cálculos (especialmente multiplicaciones). Al efectuar una transformada de Fourier en un medio discreto se producen efectos tales como: Efecto de Solapamiento, Fuga Espectral y Pérdida por Scalloping (irregularidades en una curva normalmente lisa) [14].

2.2.8 Hardware

Entre los requerimientos de hardware para la realización de un Analizador de Espectro tenemos: unidad de cómputo para la realización de cálculos de la FFT y presentación de los datos, hardware para la adquisición de datos e interfaz para visualización.

Unidad de Computo

La unidad seleccionada debe tener un alto poder de cómputo y la portabilidad suficiente para la realización de proyectos, incluir puertos de entrada y salida, soporte de periféricos y conexión USB. Los dispositivos que cumplen con estos requisitos están bajo la categoría de “computadores del tamaño de una tarjeta de crédito”

Raspberry PI

Es un computador de bajo costo que puede ser conectado a cualquier monitor o TV, ha sido usado en una gran cantidad de proyectos debido a su popularidad y el

soporte brindado por la comunidad de software libre. Entre los diferentes puertos que posee tenemos: HDMI, Ethernet, GPIO, Jack de audio 3.5mm y USB, a continuación en la figura 2.7 se muestran las diferentes partes de Raspberry PI [15,16].

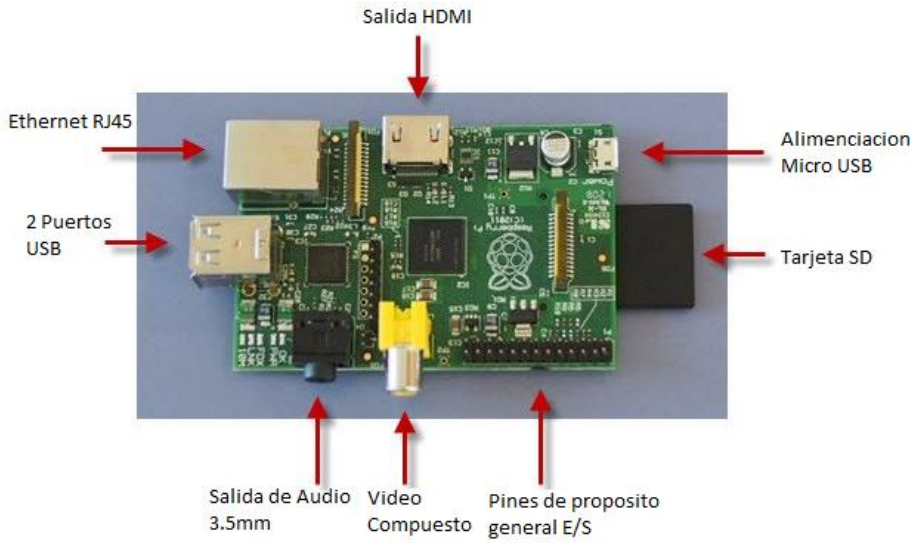


Figura 2.7. Partes de Raspberry PI

Fuente: http://www.raspberryshop.es/

BeagleBone Black

Es una plataforma de bajo costo diseñada para desarrolladores y aficionados, cuenta con el apoyo de la comunidad de software libre. Es capaz de arrancar la distribución Linux Armstrong en aproximadamente 10 segundos y empezar proyectos de desarrollo en menos de 5 minutos con solo un cable USB; consta de muchos accesorios y placas de expansión que están listas para usarse.

BeagleBone Black cuenta con almacenamiento interno de alta velocidad, y múltiples puertos de comunicación; también tiene dos opciones de alimentación, además; el sistema operativo por defecto trae precargados servicios de

conectividad y programación. En la figura 2.8 se muestra las partes de la placa [17].

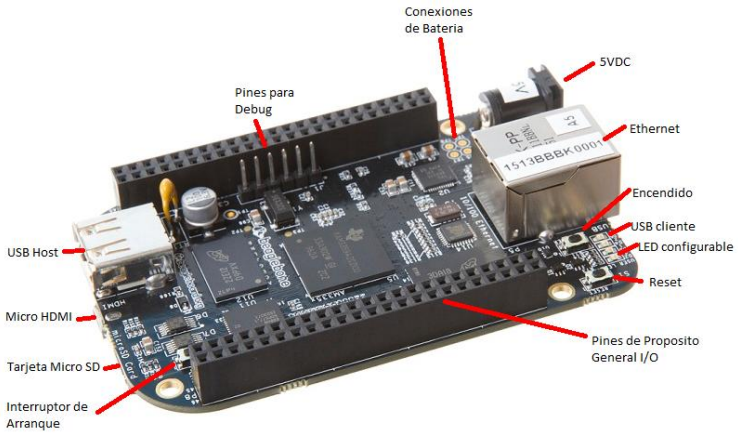


Figura 2.8. Partes de BeagleBone Black

Fuente: http://elinux.org/Beagleboard:BeagleBoneBlack

PcDuino

Es una placa de alto desempeño a un costo moderado, puede ejecutar sistemas operativos como Ubuntu y Android, posee varias interfaces de comunicación, entre estas tenemos un puerto HDMI que soporta video de alta definición. Su diseño permite que sea 100% compatible con las tarjetas de expansión de Arduino, Linux y Android. A continuación en la Figura 2.9 se muestran las partes principales de PcDuino3 [18].

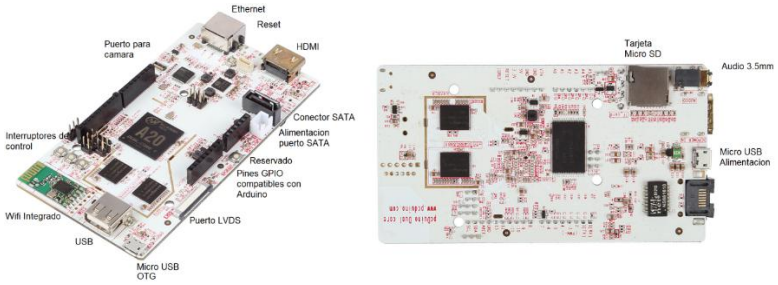


Figura 2.9. Partes de PcDuino3

Fuente: http://learn.linksprite.com/pcduino/quick-start/explaination-of-pcduino3-headers/

Banana PI

Es un dispositivo de bajo costo, pequeño y lo suficientemente flexible para el uso diario. Construido con un procesador ARM de doble núcleo y tarjeta gráfica dedicada, usa software libre y puede ser utilizado como plataforma para aplicaciones de diferentes propósitos.

Es mecánica y eléctricamente compatible con Raspberry PI, puede ejecutar los sistemas operativos Linux, Android y Firefox OS; también posee una distribución de puertos similar a Raspberry PI. En la figura 2.10 se detalla las partes principales de la placa [19,20].

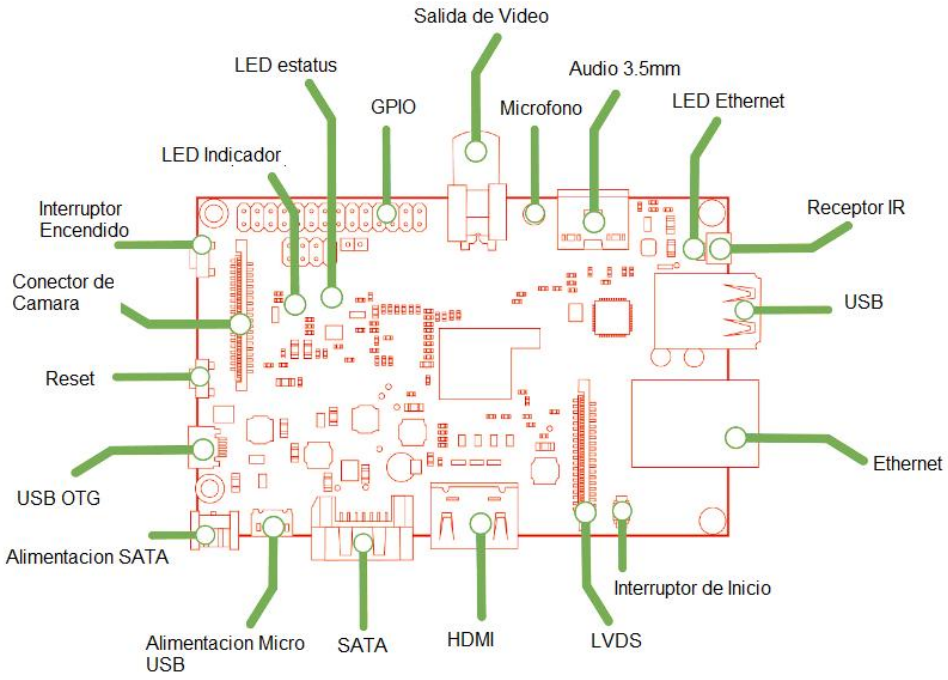


Figura 2.10 Partes Principales de Banana PI

Fuente: http://www.bananapi.org/p/product.html

HummingBoard

Una placa muy similar a Raspberry PI, brinda la posibilidad de reemplazar el procesador, fue diseñada para un mejor desempeño gráfico y máxima interconexión Ethernet; además posee una extensa variedad de puertos para desarrollo y conexión de hardware adicional, puede correr Linux y Android.

Este dispositivo cuenta con varias versiones a la venta, desde un modelo simple y económico hasta un modelo más completo y costoso. A continuación en la Figura 2.11 se muestra las principales partes del dispositivo [21].

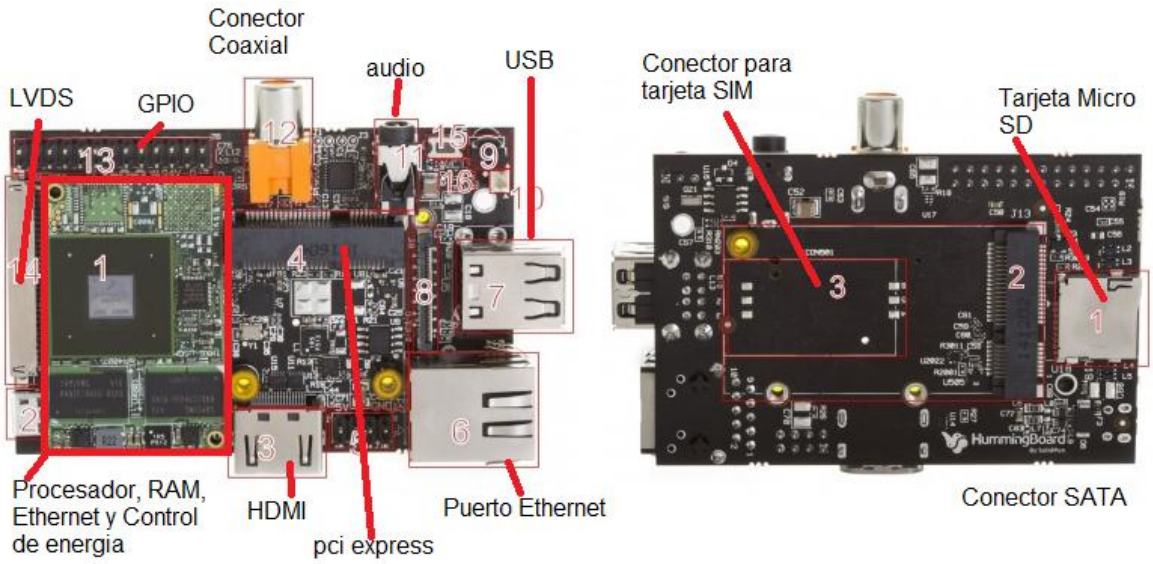


Figura 2.11. Partes Principales de HummingBoard

Fuente: http://www.solid-run.com/wiki/index.php?title=HummingBoard_Hardware

UDOO

Es una mini computadora de bajo costo, de hardware libre, puede ejecutar Android o Linux, tiene un procesador Atmega integrado que se programa como un módulo Arduino estándar y está orientada a la realización de prototipos o desarrollo de software.

En la figura 2.12 se muestra los principales componentes de UDOO; tiene más interfaces que sus competidores, es compatible con Arduino y sus tarjetas de expansión [19].

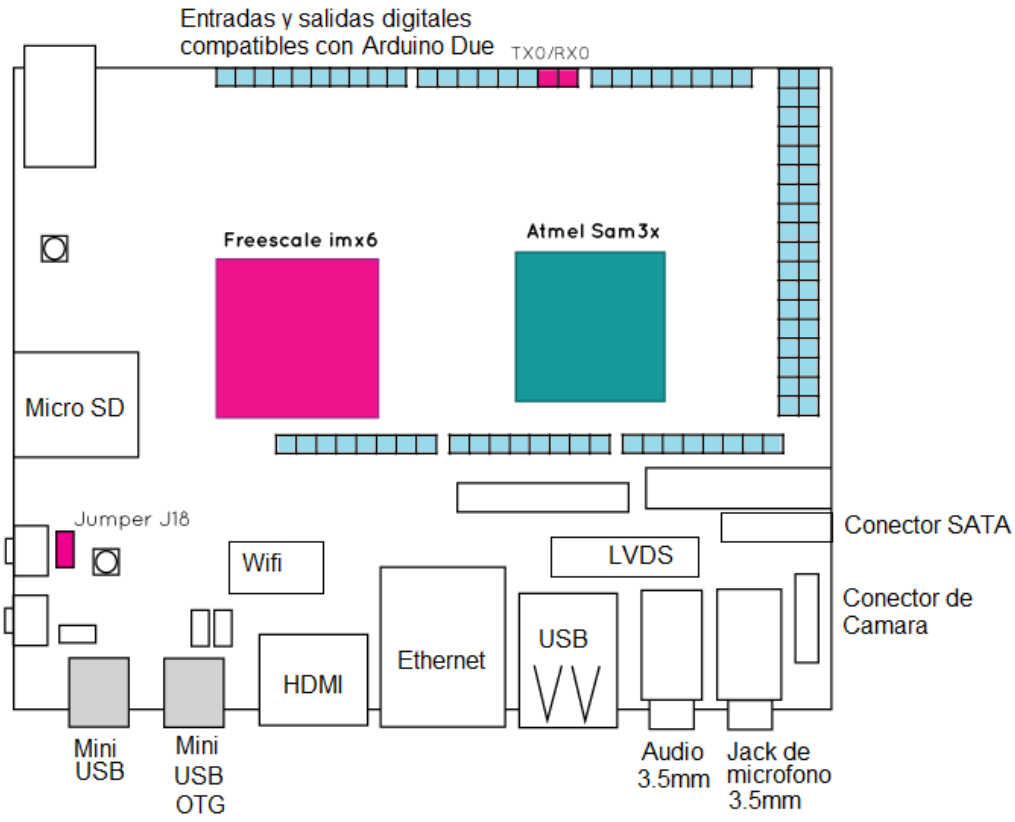


Figura 2.12. Partes Principales de UDOO

Fuente: http://www.udoo.org/features/

2.2.9 Conversor Análogo Digital

Un conversor análogo digital es un circuito que permite transformar una señal eléctrica en información digital discreta a través de las etapas de: muestreo, cuantización y codificación; que serán explicadas a continuación.

Muestreo

Es la señal en tiempo discreto obtenida al tomar muestras de una señal análoga o digital cada T segundos; de este modo el intervalo de muestreo es el tiempo T entre dos muestras y la frecuencia de muestreo es la inversa del periodo de muestreo.

Para obtener la frecuencia de muestreo debemos conocer la Frecuencia Máxima de operación, para evitar que exista aliasing (solapamiento) se debe aplicar el teorema de Nyquist.

El teorema de Nyquist enuncia lo siguiente: la frecuencia de muestreo debe ser mayor que dos veces la frecuencia máxima encontrada en la señal, para que pueda ser recuperada sin pérdidas de información.

Cuantización

Es la conversión de una señal en tiempo discreto con valores continuos a una señal en tiempo discreto con valores discretos (una señal digital). Existe un conjunto de valores finitos que puede representar cada muestra, los cuales se llaman niveles de cuantificación. La cuantificación es un proceso irreversible, ya que al discretizar un valor continuo se produce una pérdida de información.

En este proceso cada uno de los datos digitales se representa con un número de bits finito, ocasionando que la señal original y muestreada difieran. Existen dos técnicas para realizar la cuantificación que son útiles en diferentes casos [23]:

La Cuantización Uniforme: Se caracteriza porque los niveles de cuantificación tienen espacios uniformes.

La Cuantización No Uniforme: Se subdivide en dos métodos que son Ley μ y Ley A, en los cuales el espaciado entre los niveles de cuantificación responde a las ecuaciones correspondientes a cada ley [24].

Codificación

Consiste en la traducción de los valores muestreados y cuantizados al sistema binario mediante códigos pre establecidos. La señal se convierte en un tren de pulsos digitales, la codificación es un proceso reversible por el cual se puede reconstruir la señal que ha sido recibida por la etapa de Cuantización [25].

2.3 Propuesta de Solución.

La implementación de un Analizador de Espectro para los estudiantes y aficionados relacionados con la Carrera de Ingeniería Electrónica facilitará el desarrollo de prácticas, ya que dispondrán de un equipo de bajo costo y diversas prestaciones.

CAPÍTULO III

METODOLOGÍA

3.1 Modalidad de Investigación

En el presente proyecto de titulación se utilizaron las siguientes modalidades de investigación:

Investigación Documental Bibliográfica, permitió la obtención de información y profundización de conocimientos a partir de distintas fuentes como son: Repositorios Digitales, Artículos Científicos, Libros y Diversas Fuentes Bibliográficas. A través de esto se obtuvieron todos los detalles concernientes al desarrollo del proyecto, lo que permitió alcanzar los objetivos propuestos.

Investigación Aplicada o Técnica, que se encuentra enfocada en la aplicación de la información recolectada referente al problema, dando así una solución al mismo; utilizando los procedimientos e información adecuados.

3.2 Recolección de Información

Se realizó a través de la revisión de la documentación referente al tema acudiendo a las fuentes bibliográficas disponibles, en busca de adquirir los conocimientos necesarios, las cuales permitirán tener una visión más clara del proyecto y de esta

manera realizar una implementación basada en las mejores tecnologías y procedimientos existentes.

3.3 Población y Muestra

El presente proyecto no requerirá de población y muestra, ya que se realizó una investigación experimental técnica que permitirá el desarrollo de un prototipo.

3.4 Procesamiento de Datos

El procesamiento de datos consiste en utilizar la información recolectada en las pruebas de funcionamiento con el dispositivo parcialmente finalizado, para determinar los rangos de frecuencia soportados por el equipo. Este proceso permite conocer los errores en el diseño y posteriormente ejecutar las respectivas correcciones.

3.5 Desarrollo del Proyecto

- Selección del hardware y software necesarios en base al tipo de analizador escogido.
- Elaboración del programa e interfaz gráfica de usuario.
- Construcción de la estructura básica del dispositivo.
- Diseño de la etapa de adquisición de datos.
- Integración de todas las etapas del sistema.
- Realización de pruebas de rendimiento.
- Depuración del código fuente.
- Finalización del dispositivo.
- Elaboración del informe.

CAPÍTULO IV

DESARROLLO DE LA PROPUESTA

En el presente proyecto se desarrolló un Analizador de Espectro económico a través del software y hardware libre, está orientado a estudiantes de carreras relacionadas con la Electrónica; los cuales requieren estos equipos para realizar prácticas.

El proyecto utiliza software libre, por lo tanto el desarrollo está limitado al nivel en que se encuentre el software principal utilizado en el proyecto. En los años recientes las comunidades de software libre han cobrado gran fuerza, sin embargo aún carecen de suficientes recursos económicos para impulsarse.

Utilizando la información recolectada sobre los tipos de analizadores se determinó la mejor alternativa para realizar la implementación del proyecto sería de forma digital, ya que la construcción de un Analizador de Espectro analógico es obsoleta dado el actual desarrollo tecnológico.

Las partes principales de un Analizador de Espectro Digital son: Conversor Análogo Digital, Software Para El Cálculo De La Transformada Rápida De Fourier, Hardware Para Computación, Periféricos de Entrada e Interfaz Gráfica que permita visualizar los resultados e interactuar con el usuario.

El Analizador de Espectro utiliza las señales eléctricas capturadas a través del conversor análogo digital, para luego procesarlas utilizando el software instalado en la unidad de procesamiento de datos, a continuación a través de una interfaz visual el usuario modifica los parámetros necesarios para realizar las lecturas correspondientes. En la figura 4.1 se muestran las diferentes etapas del proyecto.

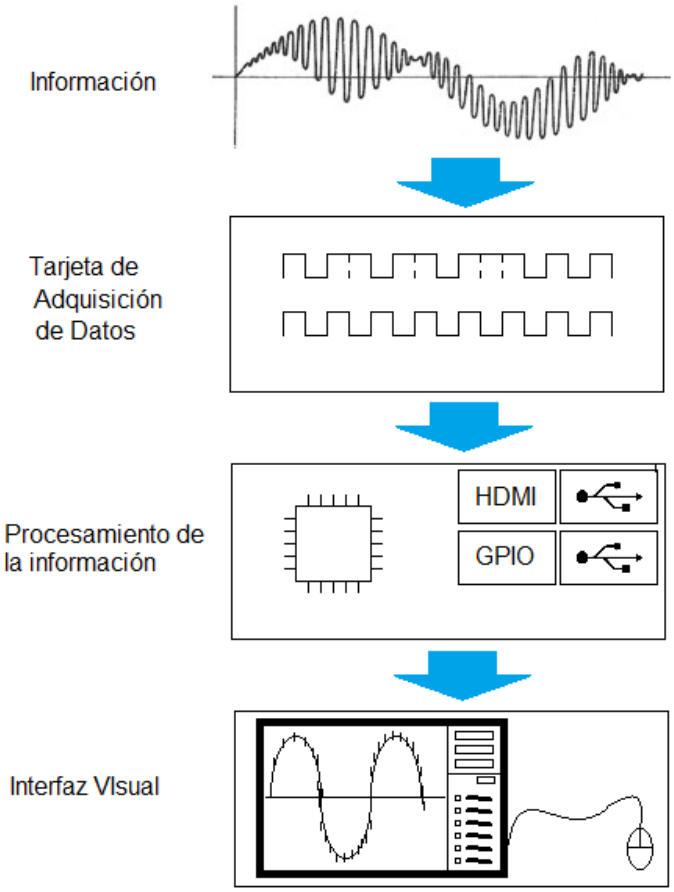


Figura 4.1 Diagrama de Funcionamiento del dispositivo

Elaborado por: Freddy Carrillo

La adquisición de datos se realizó a través de un conversor análogo digital que permite tener un flujo de datos constante y exacto, con una resolución dependiente de la cantidad de bits usados para la conversión. Su selección debe basarse en la frecuencia de operación esperada y la compatibilidad con la plataforma usada [23].

La Transformada Rápida de Fourier es utilizada ya que es un algoritmo optimizado para el cálculo de la Transformada Discreta de Fourier, que demostró adaptarse perfecta y eficientemente a la ejecución digital reduciendo el tiempo que emplea; obteniendo un resultado aproximado al real [13,14].

La unidad de cómputo es la parte central del proyecto, en la cual se realizan todos los procesos lógicos necesarios, además de la comunicación con todos los dispositivos, ingreso y control de datos.

4.1 Selección de Hardware y Software.

4.1.1 Selección de la unidad de Cómputo

Una de las partes principales del proyecto es el hardware, encargado de la recolección, procesamiento y manipulación de datos, además de la comunicación con distintas interfaces de bajo y alto nivel; que permiten la visualización y control de la información por parte del usuario.

Los dispositivos con estas características son conocidos como mini ordenadores, estos cuentan con interfaces de alto y bajo nivel. Las interfaces de bajo nivel o pines GPIO (pines de entrada y salida de propósito general) permiten realizar una comunicación con dispositivos lógicos básicos como son circuitos integrados o periféricos de bajo nivel que requieran comunicación serie o paralela. Las interfaces de alto nivel son puertos que se encuentran en todos los computadores actuales, los más utilizados son: USB, Ethernet y salidas de audio y video.

A partir de la creación de los mini ordenadores se logró impulsar diversos proyectos que requieren capacidades de cómputo mucho más elevadas y cierta portabilidad. Estas plataformas utilizan sistemas operativos de código abierto basados en Linux, los cuales son una modificación de los sistemas estándar por lo tanto no son 100% compatibles con todo el software; por ello requieren el apoyo de la comunidad de software libre para el desarrollo de drivers y actualizaciones

posteriores, incluyendo la creación de nuevas herramientas para estas plataformas.

El hardware libre ha permitido que estos dispositivos sean accesibles para todas las personas a precios módicos, el software y la información necesaria están disponibles gratuitamente en internet. A continuación en la tabla 4.1 se muestra una comparación referente a los equipos y sus características.

Tabla 4.1 Cuadro Comparativo Raspberry PI, PcDuino, BeagleBone Black

	Raspberry PI	PcDuino	BeagleBone Black
Chip	Broadcom BCM2835		TI AM3359
CPU	ARM1176JZ-F de 700MHz	1GHz ARM Cortex A8	1 GHz ARM Cortex-A8
GPU	Dual Core VideoCore IV® Multimedia Co-Processor	OpenGL ES2.0, OpenVG 1.1 Mali 400 core	PowerVR SGX530
Memoria RAM	512MB de SDRAM	1GB	512 MB DDR3
Almacenamiento	Ranura para tarjetas SD, MMC, SDIO	2GB Flash, tarjeta SD de hasta 32GB	2 GB 8-bit embebidos MMC de memoria flash, tarjeta micro SD
Interfaces de bajo nivel	26 pines GPIO	Interface de 2.54mm compatible con Arduino	2 filas de 46 pines
Interfaces de alto nivel	3.5mm Jack de Audio, HDMI, 2x USB, 1 Ethernet RJ45	2xUSB, HDMI, Ethernet RJ45, 3.5mm Jack de Audio y WiFi integrado	2x USB2.0, micro HDMI, Ethernet RJ45.
Sistemas Operativos	Linux	Linux y Android	Linux, Android y otros
Soporte de la Comunidad	Alto	Medio	Bajo

Fuente: http://www.onemansanthology.com/blog/pcduino-vs-beaglebone-black-vs-raspberry-pi/

Al realizar el análisis de la tabla comparativa sobre los equipos propuestos se muestra que tienen características similares en cuanto a niveles de voltaje, puertos de alto y bajo nivel, procesadores, sistemas operativos y memoria RAM.

Todos estos dispositivos están orientados al desarrollo y la investigación, por lo tanto son susceptibles de mejoras de software.

Las contribuciones de la comunidad de software libre brindan drivers y actualizaciones que impulsan el desarrollo de nuevos proyectos, es por esta razón que la cantidad de miembros de la comunidad es decisiva a la hora de seleccionar un dispositivo; por este motivo se toma como parte central del proyecto el equipo Raspberry PI, ya que la comunidad que lo respalda es extensa.

Al tener la unidad de cómputo Raspberry PI se tomó en cuenta sus especificaciones eléctricas y los tipos de comunicación disponibles a través del puerto GPIO, con esta información se seleccionó el hardware enfatizando la compatibilidad con la placa, sobre todo con la interfaz de bajo nivel GPIO la cual está en contacto con los circuitos principales.

4.1.2. GPIO (Pines de Entrada y Salida de Propósito General)

Son pines de propósito general de entrada y salida, Raspberry PI permite conectar periféricos y placas de expansión para acceder al CPU a través de este puerto, los cuales están controlados por el circuito integrado BCM2835, que a su vez se encarga de otras funciones como manejar los puertos de alto nivel. La distribución de pines se muestra a continuación en la figura 4.2.

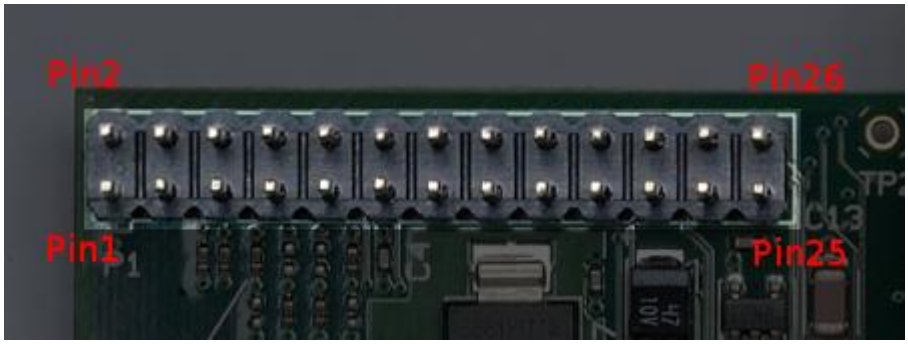


Figura 4.2. Pines GPIO de Raspberry PI

Fuente http://elinux.org/RPi_Low-level_peripherals

A la fecha existen 2 versiones principales de la placa, las cuales son: Raspberry PI B y Raspberry PI B+, se diferencian en la cantidad de pines GPIO disponibles. En la tabla 4.2 se muestra la distribución de los 26 pines y sus funciones [26].

El modelo B cuenta con 26 pines, mientras que el modelo B+ cuenta con 40 pines; los primeros 26 pines no han tenido modificaciones, los 14 pines adicionales de la segunda fase no cuentan todavía con controladores; es por esta razón que la compatibilidad solamente se da desde B hacia la B+.

Tabla 4.2. Distribución de pines de Raspberry PI, fila superior.

Pin	**Nombre Rev1, Rev2**	**Notas de Hardware**	**Función Alt 0**	**Función Alternativa**
P1-02 y P1-04	5 Voltios	Alimentación a través de entrada con múltiples fusibles	-	-
P1-06	GND	-	-	-
P1-08	GPIO14	Inicio para Alt 0	UART0_TXD	ALT5 = UART1_TXD
P1-10	GPIO15	Inicio para Alt 0	UART0_RXD	ALT5 = UART1_RXD
P1-12	GPIO18	-	PCM_CLK	ALT4= SPI1_CE0_N ALT5 = PWM0
P1-14	GND	-	-	-
P1-16	GPIO23	-	-	ALT3 = SD1_CMD ALT4= ARM_RTCK
P1-18	GPIO24	-	-	ALT3 = SD1_DAT0 ALT4 = ARM_TDO
P1-20	GND	-	-	
P1-22	GPIO25	-	-	ALT3 = SD1_DAT1 ALT4 = ARM_TCK
P1-24	GPIO08	-	SPI0_CE0_N	-
P1-26	GPIO07		SPI0_CE1_N	-

P1-01	3.3 V	50 mA máximo	-	-
P1-03	GPIO 2	1K8Ω pull up	I2C0_SDA I2C1_SDA	-
P1-05	GPIO 3	1K8 Ω pull up	I2C0_SCL I2C1_SCL	-
P1-07	GPIO 4	-	GPCLK0	ALT5 = ARM_TDI
P1-09	GND	-	-	-
P1-11	GPIO17	-	-	ALT3 = UART0_RTS ALT4 = SPI1_CE1_N ALT5 = UART1_RTS
P1-13	GPIO27	-	PCM_DOUT reservado	ALT4 = SPI1_SCLK ALT5 = GPCLK1 ALT3= SD1_DAT3 ALT4 = ARM_TMS
P1-15	GPIO22	-	-	ALT3 = SD1_CLK ALT4 = ARM_TRST
P1-17	3.3 V	50 mA máximo	-	-
P1-19	GPIO10	-	SPI0_MOSI	-
P1-21	GPIO9	-	SPI0_MISO	-
P1-23	GPIO11	-	SPI0_SCLK	-
P1-25	GND	-	-	-

Fuente: http://elinux.org/RPi_Low-level_peripherals

Pull-Up y Pull-Down Internos

El puerto GPIO incluye la opción de habilitar o deshabilitar estas resistencias, estos ajustes se realizan únicamente por software, los valores predeterminados se muestran a continuación se muestra en la tabla 4.3.

Tabla 4.3 Valores de Pull Up, Pull Down en Raspberry PI

Resistencia	Mínimo	Máximo
PULL-UP	50 K Ω	65K Ω
PULL-DOWN	50K Ω	60K Ω

Fuente: http://elinux.org/RPi_Low-level_peripherals

4.1.3 Soporte de Controladores

La Fundación Raspberry PI no incluyó el controlador de los pines GPIO en la fase inicial del sistema operativo, los controladores estándar de Linux para GPIO pueden ser usados tras ciertas modificaciones. La comunidad desarrollo los drivers para SPI e I^2C, estos vienen incluidos en el kernel desde el 14 de Octubre de 2012, soporta la comunicación de 1 hilo mediante el método de BitBanging lo que resulta en un mayor consumo del CPU.

La configuración de las funciones de los pines GPIO puede ser realizada a través de varios lenguajes como: C, C++, WiringPi, C + sysfs, C#, Ruby, Perl, Python, Scratch, RpiScratchIO, Java, Java usando la librería Pi4J, Aplicación web de Java a través de HTTP, Scripts en sysfs de Raspbian, lazarus/Free Pascal y BASIC [27].

Interface SPI

Las sigas SPI corresponden a: Interface de Puerto Serial, esta comunicación puede ser de transmisión y recepción con las opciones dúplex, half dúplex y simplex, dependiendo del tipo de comunicación se requiera el dispositivo conectado

Raspberry PI está equipado con un bus SPI con 2 selectores de chip, aunque por el momento solo uno de ellos es utilizable; el controlador maestro de SPI esta desactivado en Raspbian, en la tabla 4.4 se detallan los pines para el bus SPI.

Tabla 4.4 Especificación de Pines para el bus SPI

Función	Pin	Pin	Nombre
MOSI	P1-19		
MISO	P1-21		
SCLK	P1-23	P1-24	CE0
GND	P1-25	P1-26	CE1

Fuente: http://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md

Controladores

Wiring Pi

El desarrollador a cargo de la página "Gordon Projects" escribió en lenguaje C la mayoría de las librerías bajo el nombre "Wiring Pi" para controlar los pines GPIO al estilo de Arduino y enfatizando la compatibilidad, la información necesaria está disponible de forma gratuita, se puede controlar los pines GPIO al acceder directamente a los registros de hardware.

Librería de BCM2835

Esta es una librería de C que provee el control de los pines GPIO y otras funciones de entrada y salida en el circuito integrado, proporciona un control más directo sobre el hardware al acceder directamente a los registros del mismo.

El Circuito Integrado BCM2835 tiene 3 controladores SPI, solamente el controlador de SPI0 está disponible para su utilización desde el puerto P1. A continuación en la tabla 4.5 se muestran los modos maestros del bus SPI.

Tabla 4.5 Abreviaciones de hardware SPI

Abreviación	Nombre	Significado
SCLK	Serial Clock	Reloj del puerto serial
CE	Chip Enable	Activador de Circuito Integrado
MOSI	Master Out Slave In	Maestro: salida, Esclavo: entrada
MISO	Master In Slave Out	Maestro: entrada, Esclavo: salida
MOMI	Master Out Master In	Maestro: salida, Maestro: entrada
MIMO	Master In Master Out	Maestro: entrada, Maestro salida

http://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md

Modo Estándar o Normal

La comunicación a través de la Interfaz de Puerto Serial SPI en su modo Maestro Estándar requiere de la conexión de los pines: SCLK, MOSI, MISO; este tipo de conexión se denomina de tres cables. En la tabla 4.5 muestran las funciones de cada pin.

Modo Bidireccional

El modo bidireccional maestro usa el mismo estándar SPI, con la excepción de que se utiliza el mismo canal para la transmisión MIMO, MISO y MOSI. En la tabla 4.5 muestran las funciones de cada pin.

Modo LOSSI

El modo de interfaz serial de baja velocidad LOSSI permite la emisión de comandos a los periféricos como LCD y transferir datos de estos. Las instrucciones LOSSI y los parámetros tienen una longitud de 8 bits, se agrega un bit para indicar cuando el byte es un comando, parámetro o dato, este bit es 1 cuando se trata de datos y en 0 para comandos; este noveno bit es transmitido de manera serial. LOSSI es comúnmente usado con MIPI DBI tipo C compatible con controladores LCD. Normalmente se soportan 8 bits, pero en modo LOSSI se soportan 9 bits.

Velocidad de Transmisión.

La velocidad de reloj del bus SPI se determina en base a la frecuencia del núcleo sobre CDIV (divisor de reloj), el cual es determinado por la configuración. El parámetro CDIV debe ser múltiplo de 2, los valores resultantes se redondean al menor. El máximo valor de SCLK es 65536, a continuación en la tabla 4.6 se detallan las velocidades SPI disponibles.

Tabla 4.6 Velocidades del bus SPI

cdiv	Velocidad
2	125.0 MHz
4	62.5 MHz
8	31.2 MHz
16	15.6 MHz
32	7.8 MHz
64	3.9 MHz

Fuente: http://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md

Selector de Circuito Integrado

Se encarga de indicar al circuito integrado cuando se comenzarán a transmitir o recibir los datos, los tiempos de configuración y espera son establecidos automáticamente para los canales CS cuando se encuentra en modo DMA; el canal se activará después de transcurridos 3 ciclos del núcleo, del mismo modo será desactivado una vez transcurrido 1 ciclo del reloj del núcleo [28].

4.1.3 Selección de los Conversores Análogo Digital.

Los conversores análogo digital deben ser compatibles eléctricamente con el puerto GPIO, ya que este no soporta sobre voltajes o cortocircuitos; también debe poseer alguno de los tipos de comunicación soportados por el puerto. Al conectar hardware incompatible en alguno de estos dos sentidos se puede ocasionar un daño permanente al procesador de la placa Raspberry Pi.

En teoría la velocidad máxima de comunicación de un dispositivo SPI a través del puerto GPIO de Raspberry es 125MHz, esta velocidad en la práctica se ha logrado alcanzar solamente bajo condiciones especiales; por lo tanto en el desarrollo de un equipo real se deben realizar pruebas con distintos tipos de conversores para determinar los más adecuados.

En base a las características de la placa de desarrollo Raspberry PI se han seleccionado varios circuitos integrados para realizar la conversión análogo digital.

La transmisión serie utiliza una velocidad de comunicación mucho más alta, basado a la cantidad de bits empleados.

La comunicación SPI utiliza 3 cadenas de 8bytes, por lo tanto un conversor análogo digital de 1MSPS o 1MHz necesita una velocidad de comunicación de $1x10^6x3x8 = 24MHz$. La velocidad máxima del puerto ha sido debatida en muchos foros, con variedad de resultados, por lo tanto se han estimado valores para posibles frecuencias de funcionamiento y en consecuencia se han seleccionado los siguientes circuitos integrados especificados en la tabla 4.7.

Tabla 4.7. Circuitos Integrados Seleccionados

Nombre	**Función**	**Interfaz de entrada**	**Comunicación**	**Resolución**	**Muestras por segundo**	**Voltaje de Operación**
MCP3008	ADC	Seudo diferencial y Single Ended	SPI y serial	10 bits	$2x10^5$	2.7V hasta 5.5V
AD7366BRUZ-5	ADC	Diferencial	SPI y serial	12 bits	$1x10^6$	2.7V a 16.5V
AD9220ARSZ	ADC	diferencial y Single Ended	paralela	12 bits	$1x10^7$	2.7V a 5.25V
AD9201ARSZ	ADC	Diferencial y Single Ended	paralela	10 bits	$2x10^7$	2.7V a 5.5V
AD9283BRSZ-50	ADC	Diferencial y Single Ended	paralela	8 bits	$5x10^7$	2.7V a 3.6V
AD9283BRSZ-100	ADC	Diferencial y Single Ended	paralela	8 bits	$1x10^8$	2.7V a 3.6V
AD5541ABRMZ	DAC	SPI, serie, QSPI y Microwire	-	12 - 16 bits	$1x10^6$	2.7V a 5.5V

Fuente: [29 – 37].

Previo a la conexión y pruebas de funcionamiento con los circuitos integrados seleccionados, se debe realizar el diseño y construcción de las placas respectivas

en base a los ajustes necesarios encontrados en las hojas de información; ya que la mayoría de los circuitos integrados son de tipo SMD.

Se diseñó una placa por cada circuito integrado, ya que se realizaran pruebas para determinar cuáles son funcionales en la práctica; y a partir de esto se elaborará la placa definitiva donde se acoplará el circuito integrado más idóneo.

La interconexión con la placa Raspberry Pi requirió de un circuito de acoplamiento, por lo tanto en caso de cualquier reparación o modificación propuesta por sus usuarios deberá realizarse en dicha placa. Ciertos circuitos integrados requieren más de dos fuentes de energía, con polaridades positivas o negativas, es por este motivo que una fuente genérica de computador servirá perfectamente al propósito del proyecto debido a su estabilidad, precio, protección contra cortocircuitos, potencia máxima, cantidad de salidas y tamaño.

4.1.4 Selección del sistema operativo y lenguaje de programación

Raspberry PI cuenta con varias distribuciones oficiales disponibles gratuitamente; la alternativa para principiantes es NOOBS, que es un instalador con todas las distribuciones oficiales en un mismo compilado; el cual puede ser cargado en una tarjeta SD de 8GB o más [40].

NOOBS incluye los sistemas operativos: Raspbian (basado en Debian), PIDORA (basado en Fedora), OPENELEC (centro multimedia basado en XBMC), RASPBMC (es una instalación mínima de Debian, con XBMC), RISC OS (diseñado específicamente para procesadores ARM, no está basado en Linux ni está relacionado con Windows). El sistema operativo predeterminado es Raspbian, la instalación de los demás sistemas operativos se requiere una conexión a internet [38,39].

La distribución recomendada es Raspbian, que ha sido específicamente diseñada y optimizada para Raspberry PI, la mayoría de mejoras y actualizaciones están orientadas a esta distribución; el proceso para reemplazar la partición root en la

tarjeta SD por otra distribución Linux es algo simple. La fundación Raspberry PI recomienda el uso del lenguaje de programación Python, pero puede utilizarse cualquier lenguaje que funcione con los procesadores ARMv6, a continuación en la tabla 4.8 se muestra una comparativa de Python con otros lenguajes de programación [40].

4.8. Comparación de Python con otros lenguajes.

Lenguaje	**Comparación con Python**
C/C++/Java	Python permite escribir el mismo programa con menos líneas de código, se estima que requiere 5 veces menos líneas de código.
VB/PHP	Al escribir un programa muy largo en VB/PHP se empiezan a ver problemas debido a sus fallas de diseño.
Lisp/Scala/Haskell/Closure/Erlang	Estos lenguajes tienen buenas características y es muy útil para programadores avanzados, por lo tanto se puede aprender a largo plazo
Perl	Perl fue considerado uno de los mejores lenguajes, actualmente ha perdido popularidad. Es difícil de aprender ya que tiene formas diferentes de realizar las mismas tareas, su sintaxis no es intuitiva y no se considera un buen lenguaje para estudiantes.
Sd/sed/awk/bash	Estos programas pueden ser problemáticos al usarlos en consola. Python por otro lado es adecuado para estas tareas, pero Perl es mejor aún en este ámbito
Conclusión	El lenguaje más adecuado para la realización de proyectos, a excepción de aplicaciones móviles, es Python o Ruby, ya que permiten obtener resultados en la mitad del tiempo.

Fuente: http://reliscore.com/blog/why-every-programmer-should-learn-python-or-ruby/

El uso de Python facilita mucho el desarrollo del proyecto debido a la gran cantidad de aportes realizados en la plataforma Raspberry PI, por lo tanto se

encuentra más documentado. El control de los pines GPIO, el cálculo de la Transformada Rápida de Fourier y la construcción de una interfaz gráfica para la visualización de las señales y el control por parte del usuario se puede realizar completamente a través de Python y sus módulos.

Los módulos Python contienen definiciones y declaraciones almacenadas en un archivo con extensión ".py", pueden ser importados a otros módulos o al módulo principal. Existe una amplia variedad de módulos, que han sido desarrollados por terceros y son aplicables en diferentes propósitos [41].

4.1.5. Módulos Python

SPIDEV

Este módulo define un tipo de objeto que permite transacciones SPI que se ejecutan en el kernel Linux, el cual debe tener soporte para la interfaz y el driver SPI. La interface SPI está abierta para lectura o escritura y se debe tener permiso de súper usuario.

El módulo SPIDEV requiere en primer lugar actualizar Raspbian, a continuación se debe instalar el módulo python-dev; se debe habilitar el modulo editando el archivo blacklist.conf, agregando un # antes del comando spi-bcm2708. Al reiniciar, se crea una carpeta con el nombre python-spi; en ella se descargan los archivos necesarios y se procede a instalar. En el cuadro 4.9 se muestra el procedimiento.

Tabla 4.9 Procedimiento para la instalación de Spidev

```
sudo apt-get update
sudo apt-get upgrade
sudo reboot
sudo apt-get install python-dev
sudo nano /etc/modprobe.d/raspi-blacklist.conf
sudo reboot
mkdir python-spi
cd python-spi
wget https://raw.github.com/doceme/py-spidev/master/setup.py
wget https://raw.github.com/doceme/py-spidev/master/spidev_module.c
sudo python setup.py install
```

Fuente: http://tightdev.net/SpiDev_Doc.pdf

Utilizando el módulo Spidev se realiza la comunicación a través de los pines dedicados para la comunicación SPI, con el fin de transmitir y recibir cadenas de datos; a continuación muestra los comandos necesarios en la tabla 4.10.

Tabla 4.10. Código básico para la transmisión de bytes por SPI

Código	Descripción

import spidev
import time
spi = spidev.SpiDev()
spi.open(0, 1)
 	
try:
  while True:
  resp = spi.xfer2([0xAA])
  time.sleep(0.1)
except KeyboardInterrupt:
  spi.close()
	Importar módulo Spidev
	Importar módulo time
	Crea el objeto SPI
	Abre el puerto SPI 0 y el activa el dispositivo CS 1
	
	Inicio del ciclo
	Transferencia de 1 byte
	tiempo entre ciclos
	Interrupción del ciclo por teclado
	Cierre del puerto

Fuente: http://tightdev.net/SpiDev_Doc.pdf

Entre las herramientas incluidas en el módulo están: Reverse bits, esta opción invierte el orden de los datos de un Byte. La función “Print Bytes” permite transmitir un arreglo de bytes en un formato legible (hexadecimal), es útil para la depuración, la sintaxis correspondiente a estos comandos se encuentra en la tabla 4.11.

Tabla 4.11. Utilidades adicionales de Spidev

Código	**Descripción**

def ReverseBits(byte):
  byte = ((byte & 0xF0) >> 4) | ((byte & 0x0F) << 4)
 	
  byte = ((byte & 0xCC) >> 2) | ((byte & 0x33) << 2)
 	
  byte = ((byte & 0xAA) >> 1) | ((byte & 0x55) << 1)
 	
return byte
	
	Mover cuatro posiciones y regresar cuatro
	Mover dos posiciones y regresar dos
	Mover una posiciones y regresar una
	Devolver el valor byte

def BytesToHex(Bytes): return ''.join(["0x%02X " % x for x in Bytes]).strip()	 Devuelve en formato hexadecimal lo leído en bytes

Fuente: http://tightdev.net/SpiDev_Doc.pdf

Los comandos correspondientes a los diferentes modos de funcionamiento y configuraciones del módulo Spidev se encuentran especificados en la tabla 4.12.

Tabla 4.12. Comandos del módulo Spidev

Comando	**Descripción**
bits_per_word	Esta propiedad permite ver o configurar la cantidad de bits po palabra. El rango es de 8 - 16
close	Desconecta el objeto de la interface, no devuelve información, su sintaxis es: close()
cshigh	Esta propiedad permite ver o configurar si la línea CS es activa alta.
loop	Esta propiedad permite ver o configurar el loopback. Es usada para probar el PCB. Todo lo que se reciba será enviado como eco.
lsbfirst	Permite ver si el bit menos significativo será transmitido al inicio o final. Raspberry solamente envía el bit más significativo al inicio, se puede usar el opción de reverse bit en su lugar.
max_speed_hz	Permite ver o configurar la velocidad máxima del bus en Hz.
mode	Permite ver o configurar el modo SPI como un patrón de dos bit para la polaridad de reloj y fase [CPOL] [CPHA]. El rango va desde 0b00=0 hasta 0b11=3
open	Su sintaxis es: open(bus, dispositivo) Permite conectar el objeto con el dispositivo SPI deseado. Ejemplo: open(X,Y) abrirá /dev/spidev-X.Y
readbytes	Su sintaxis es: read(N) Devuelve un valor Lee N bytes del dispositivo SPI
threewire	Permite ver o configurar las senales compartidas SI, SO. Solo existe una línea de datos disponible en el puerto GPIO.
writebytes	Su sintaxis es: write([valores]) No devuelve algún valor Transmite bytes al dispositivo SPI
xfer	Su sintaxis es: xfer([valores]) Devuelve valores Realiza una transacción SPI. La línea CS será liberada y reactivada entre bloques. La instrucción delay especifica el tiempo en microsegundos entre bloques.
xfer2	Su sintaxis es: xfer2([valores]) Devuelve valores Realiza una transacción SPI. la línea CS se mantendrá activa entre bloques

Fuente: http://tightdev.net/SpiDev_Doc.pdf

Trazado de Señales

El trazado de curvas y la interfaz de usuario interactiva a realizarse en Python pueden ser llevadas a cabo a través de módulos elaborados específicamente para este trabajo. El módulo escogido para este propósito es PyQtGraph, a continuación se detalla una comparativa de los módulos existentes.

- Matplotlib es una biblioteca estándar para la realización del trazado en Python, ideal para la realización de trazos simples. Esta biblioteca se encuentra mucho más desarrollada por la comunidad y produce gráficos con calidad de publicación.
- PyQtGraph es superior en velocidad de cambios de trama, video e interactividad en tiempo real; además no requiere compilación y cuenta con muchas herramientas para el análisis grafico de datos escritas en código Python.
- PyQwt posee un buen conjunto de características y es lo suficientemente rápido como para el trazado en tiempo real. Actualmente ya no está mantenido por la comunidad y es difícil que funcione en distintas plataformas, por lo tanto es recomendable no usarlo hasta que vuelva a tener un desarrollador a cargo del proyecto. PyQwt carece de muchas de las características avanzadas que encontramos en PyQtGraph.
- Chaco es un proyecto muy interesante con buenos gráficos y una velocidad adecuada, el cual es desarrollado activamente; al igual que PyQwt puede ser difícil instalarlo en todas las plataformas y carece de algunas funciones avanzadas de PyQtGraph, aunque PyQtGraph también carece de ciertas opciones encontradas en chaco.
- GuiQwt está basado en PyQwt, también cuenta con muchos inconvenientes, aunque posee características similares a PyQtGraph [42].

PyQtGraph

Es una biblioteca gráfica escrita en Python y una librería GUI basada en PyQt4, PySide y Numpy. Está orientado a aplicaciones matemáticas, científicas y de ingeniería, en la figura 4.3 se muestra una interfaz creada con PyQtGraph.

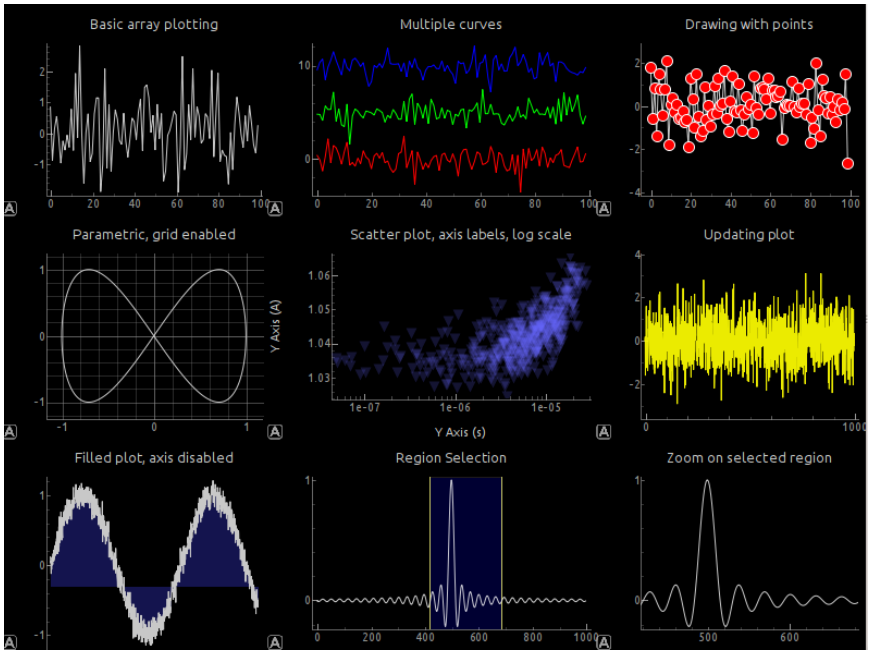


Figura 4.3. Programa: Basic Plotting Examples de PyQtGraph.

Fuente: http://www.pyqtgraph.org/

A pesar de estar escrita completamente en Python es muy rápida debido a que está basado principalmente en Numpy para el procesamiento numérico y en el framework GraphicsView de Qt para una rápida visualización.

PyqtGraph se distribuye bajo la licencia de código abierto del MIT, los requisitos mínimos para utilizar PyQtGraph son: Python 2.7, PyQt 4.8, Pyside, Numpy, Scipy, Python-OpenGl (opcional) [42].

Numpy

Este módulo es fundamental para la realización de cálculos científicos complejos, también es un eficiente contenedor de datos genéricos multidimensionales. Pueden definirse tipos de datos arbitrariamente, esto permite a numpy integrar de manera rápida y simple con muchas bases de datos [44].

PyQt4

PyQt pose muchas de las características y la funcionalidad de Qt en Python, como son: Widgets, Arreglos gráficos de Widgets, Menús, Barras de Herramientas y Muelles; posee una fácil comunicación entre componentes de la misma aplicación [45].

4.2 Elaboración del software para adquisición de datos e interfaz gráfica

La construcción de una interfaz gráfica de usuario basada en la librería PyQtGraph permite una ejecución más rápida en sistemas con pocos recursos como la placa de desarrollo Raspberry PI. La fuerte integración de esta librería con un módulo especializado en cálculos matemáticos como la Transformada Rápida de Fourier permite una representación más fluida del espectro, ya que los cuadros por segundo aumentan.

La máxima frecuencia de muestreo del dispositivo depende exclusivamente del circuito integrado que se use, por lo tanto mientras más alta sea la velocidad del conversor análogo digital que se comunique con la placa Raspberry PI más alta será la frecuencia máxima de trabajo del dispositivo. La eficiencia de las librerías destinadas al control de los pines GPIO del dispositivo es vital en este propósito.

Enfatizando la funcionalidad y flexibilidad del equipo se optó por una interfaz susceptible de modificaciones de apariencia, partiendo de los ejemplos proporcionados por el grupo encargado del desarrollo de PyQtGraph; se utilizó

ciertos fragmentos que se consideraron necesarios para el desarrollo del programa principal.

El desarrollo de la propuesta comprende la elaboración de dos programas distintos que deben trabajar en conjunto ya sea fusionándolos o intercomunicándolos. El primero es la interfaz gráfica interactiva con sus respectivos controles, que son los encargados de la representación de los datos e ingreso de parámetros de configuración; en la segunda sección se realiza la comunicación a través del puerto GPIO con los respectivos circuitos integrados, ya sea usando comunicación SPI o paralela para recibir los datos resultantes de la conversión análogo digital.

4.2.1 Elaboración de la Interfaz Gráfica

Entre los ejemplos explicativos de la librería gráfica PyQtGraph tenemos diferentes opciones de trazado de datos en planos cartesianos con múltiples configuraciones y funciones dedicadas a propósitos específicos como son: trazado de datos simple o diagramas de dispersión, configuraciones de límites, colores y rangos de visión, función de auto rango, etc.

El trazado de los datos se basa en el código perteneciente al ejemplo PanningPlot.py, consta de una ventana gráfica con un trazador que puede ser configurado en tamaño, color y desplazamiento; la ventana se actualiza continuamente para mostrar los cambios de información.

En cada ciclo se llama a una función de actualización, donde se calcula un nuevo conjunto de valores para incorporarlos a un arreglo, posteriormente se actualiza la ventana para mostrar este nuevo conjunto; al llegar a un límite definido el arreglo desplaza sus valores por lo que se obtiene el efecto de un osciloscopio.

Para demostrar el funcionamiento de este código se generó una señal de 5sen(n*0.1) con componentes aleatorias en un arreglo con un límite de 10000 datos, al final de los mismos se recorrerá el arreglo una posición a la vez; creando el efecto de un osciloscopio, mientras tanto se muestra el arreglo completo que va

creciendo a medida que transcurre el tiempo. En la figura 4.4 se observa a la derecha el arreglo con pocos datos y en la izquierda se muestra una acumulación de datos tras varios segundos activo.

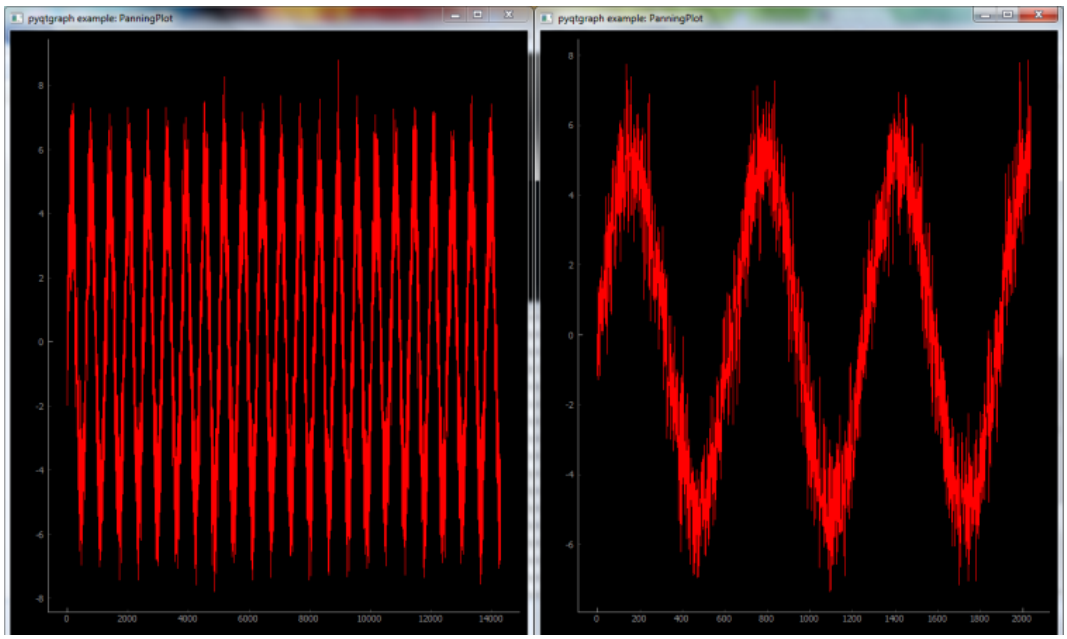


Figura 4.4 Ejecución de PanningPlot.py en tiempos diferentes

Fuente: Ejemplos demostrativos de PyQtGraph

El programa dockarea.py contiene la instrucción DockArea() que permite la inclusión de múltiples áreas de trabajo con tamaños y posición configurable, cada área puede ser dedicada a un tipo de Widget específico; estas pueden solaparse para maximizar el espacio de trabajo.

Esta instrucción permite modificar la posición, forma e incluso cerrar u ocultar secciones no deseadas una vez iniciado la ejecución del programa, en la figura 4.5 se muestra un ejemplo incorporando la instrucción DockArea(), en la cual se tiene 6 áreas de trabajo, mostrando una configuración diferente para cada una; también se aprecia el solapamiento de dos áreas.

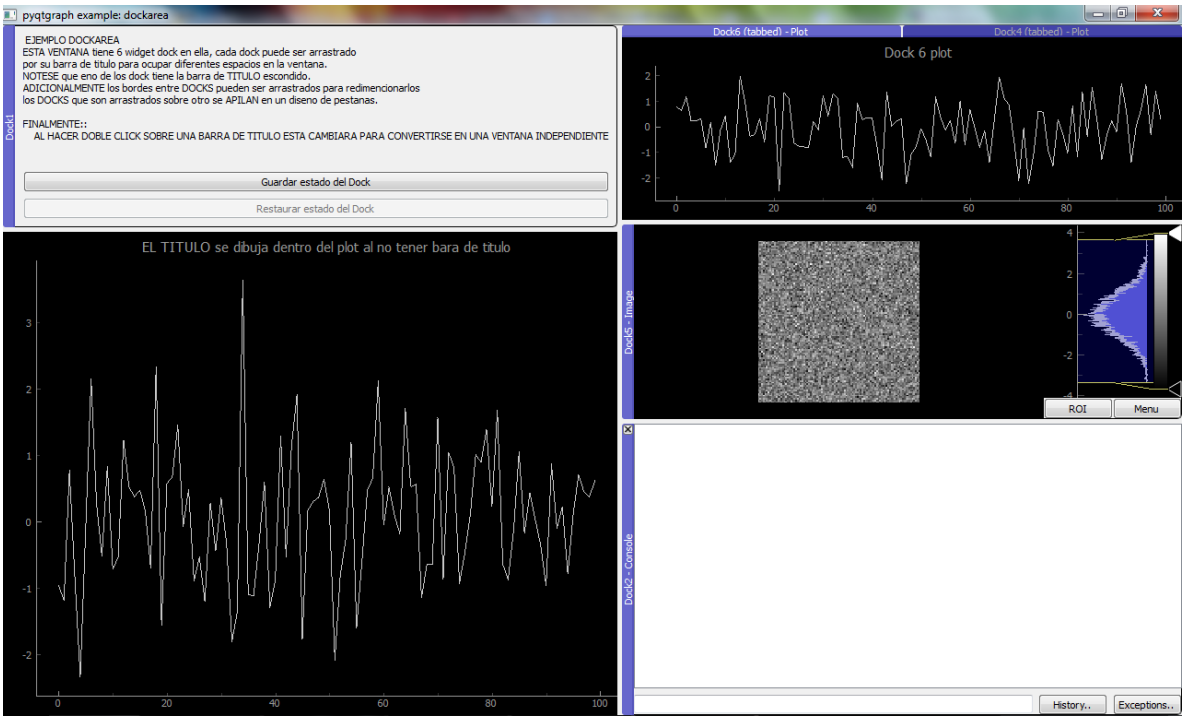


Figura 4.5 Ejecución del programa DockArea.py

Elaborado por: Freddy Carrillo

El código del programa crosshair.py contiene funciones avanzadas que permiten la obtención del valor de un punto del plano cartesiano, en base a la posición del puntero del mouse; también permite segmentar una señal, mostrando la panorámica y la recta segmentada en dos planos distintos.

En base a lo antes descrito se generó la segunda fase del programa, que ahora cuenta con las características de los dos anteriores. A continuación en la figura 4.6 se muestra el programa en ejecución.

En este programa se obtienen 3 puntos de la posición del mouse, X, Y1, Y2. Los puntos Y1 y Y2 pertenecen a las dos curvas representadas en la parte superior de la imagen, la información mostrada X corresponde al número de puntos en la posición respectiva y en Y corresponde a la amplitud de dicho punto, sin embargo existe un error de cálculo, en este ejemplo en ciertos casos se entrega resultados falsos de la posición en Y2.

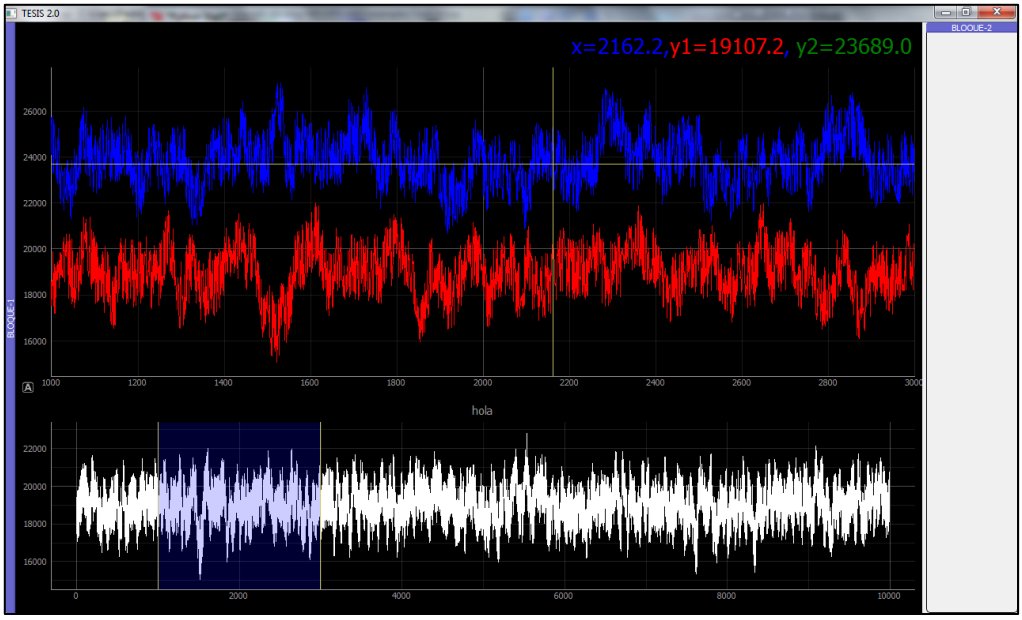


Figura 4.6 Ejecución de la segunda fase del programa

Elaborado por: Freddy Carrillo

En base al código de los programas Dockarea.py y PlotSpeedTest.py se realizó la codificó un programa con trazado dinámico de datos, que utiliza un cálculo en base al tiempo de ejecución para mostrar los cuadros por segundo que es capaz de procesar; que demuestran la velocidad de actualización del sistema bajo parámetros mínimos. En la imagen 4.7 se muestra la ejecución del programa.

Este programa genera una distribución de datos aleatoria dentro de una de las áreas de trabajo, no se incluyó controles o Widgets, con el propósito de medir el rendimiento básico del programa. La prueba de velocidad en un computador normal alcanzó un promedio de 55 cuadros por segundo; al ejecutar el mismo código sobre el sistema operativo Raspbian en la placa Raspberry PI se logró un máximo de 7 cuadros por segundo, adicionalmente se debe considerar que el trazador incluye el Widget ROI; que permite tomar muestras del área que ocupa para utilizarlas en otros trazados, pero en este ejemplo solo se lo posicionó sin establecer vínculos.

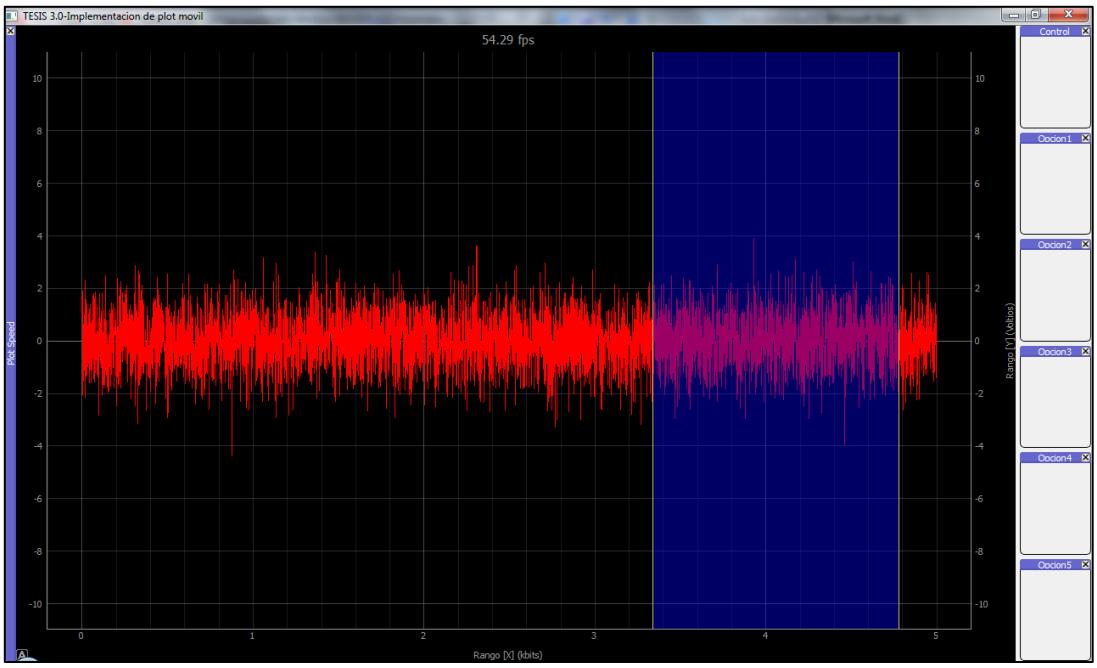


Figura 4.7 Ejecución de la tercera fase del programa

Elaborado por: Freddy Carrillo

La cantidad de cuadros por segundo que el sistema es capaz de procesar se determina mediante el siguiente algoritmo realizado en base al tiempo transcurrido entre cada ciclo completo del sistema, cabe recalcar que este proceso fue realizado solamente para determinar la funcionalidad del programa sin implementar la comunicación con circuitos externos.

El proceso para determinar los cuadros por segundo es: Medir el tiempo de ejecución, Restar el tiempo Actual menos el del ciclo Anterior. Posteriormente se actualiza el valor de la variable tiempo Anterior, para repetir el proceso en el siguiente ciclo. En la tabla 4.13 se muestran los códigos que permiten el cálculo de FPS.

Tabla 4.13 Actualización del trazado y medición del tiempo de ejecución

ahora=time()	Almacena el tiempo actual
dif=ahora-inicio	Cálculo de la diferencia de tiempo
inicio=ahora	Actualización del tiempo inicial
fps=1.0/dif	Cálculo de la frecuencia en base al tiempo
trazo1.setTitle('%0.2f fps' % fps)	Actualización de la etiqueta de FPS

Elaborado por: Freddy Carrillo

La cuarta fase se basa en el código correspondiente a los tres programas anteriores, incluyendo en esta ocasión el control y modificación de los parámetros para la generación y visualización de los datos a través de los Widgets Spinbox.

Existen configuraciones adicionales para este tipo de trazadores, entre los cuales se tiene:

Para configuraciones generales:

- Mostrar cuadrícula, efecto de antialising y cambios del color de fondo.

Para configuraciones Individuales tenemos:

- Llenado del espacio bajo la curva, se puede configurar el nivel de inicio y la intensidad del color; al superponerse con el llenado de otras curvas sus colores se combinan pero aún se puede distinguir entre ellos.
- Remarcar los puntos: se puede escoger el tipo de representación de los puntos de la curva entre puntos, círculos y cruz.

La integración de PyQtGraph con Numpy nos permite efectuar de forma inmediata el cálculo de la transformada rápida de Fourier para el conjunto de datos asociados al trazador, el control de la cantidad de datos que maneje el arreglo permite aumentar o disminuir la calidad del cálculo de la FFT, todos los efectos aplicados al trazado son también efectivos para este modo.

El trazado de datos a través de GraphicsWindow(), resulta útil debido a la cantidad de funciones integradas que posee, es por esta razón que genera más carga al

procesador para su ejecución; el uso de PlotWidget() resulta más fluido al momento de trazado y alcanza una mayor tasa de cuadros por segundo, pero carece de la función para el cálculo de la FFT; el código base es similar ya que se emplea la misma distribución de Dockarea().

En todas las fases del programa que se ha realizado se incluyó la instrucción ventana.showMaximized() para mantener la ventana maximizada por default. En la imagen 4.8, se muestra la cuarta fase del programa en modo normal (derecha) y como Analizador de espectro (izquierda).

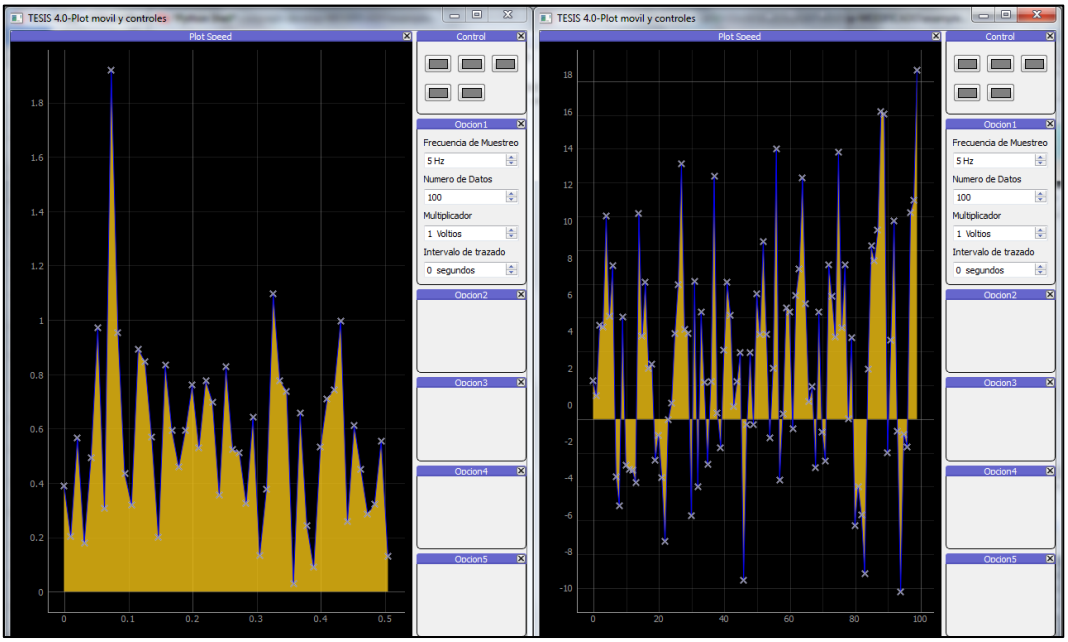


Figura 4.8 Cuarta fase: FFT y trazado de datos
Elaborado por: Freddy Carrillo

El desarrollo de la quinta fase incluye un control a través del Widget Checkbox, para la activación previa del trazador, una vez iniciado el proceso no se cargan datos en el trazador. El propósito de este diseño es liberar toda la carga posible del procesador al mantener el cálculo y adquisición de los datos separados de los procesos gráficos que consumen muchos más recursos, esto mejora significativamente desempeño.

Al activar la casilla “Inicio – Pausa” empieza el ciclo de captura o generación de datos, a continuación son almacenados en el arreglo y siguen el proceso normal de desplazamiento al llegar a su límite definido; en este punto no se realiza el trazado en la ventana gráfica, facilitando el proceso. Al desactivarse la casilla el ciclo se detiene, pero todos los datos generados se mantienen y puede ser reanudado indefinidamente. Este método es de gran utilidad para realizar análisis posteriores sobre los datos capturados, en la figura 4.9 se muestra la ejecución del código.

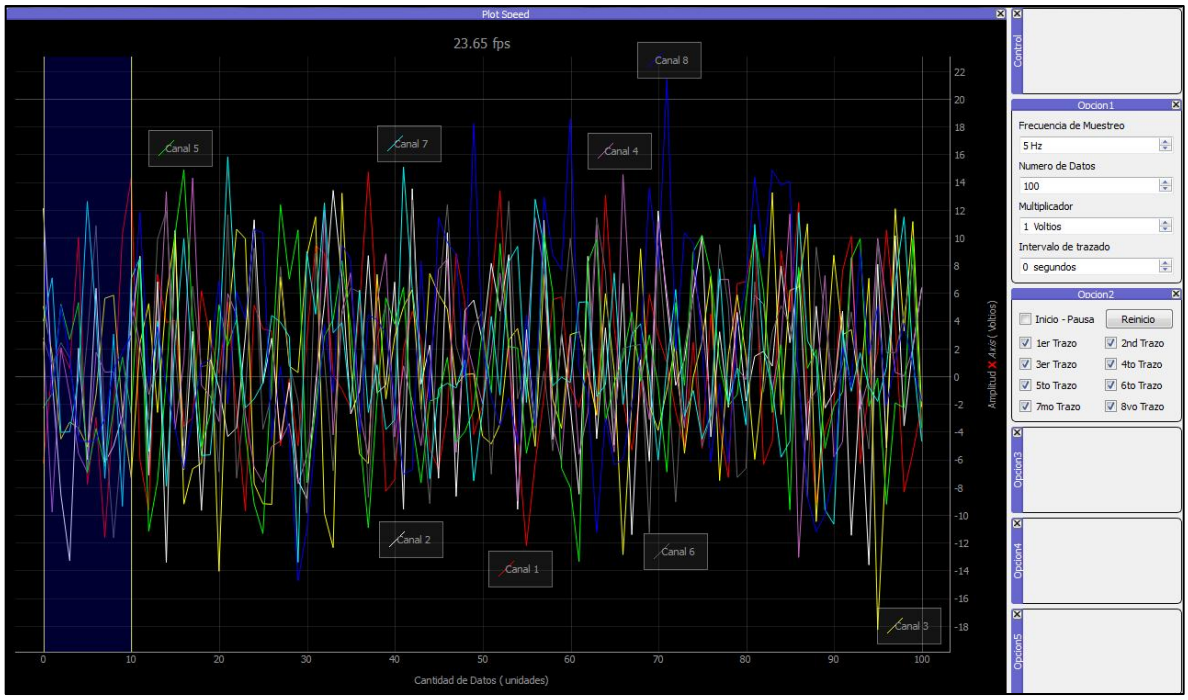


Figura 4.9 Quinta fase del programa

Elaborado por: Freddy Carrillo

El trazado de los datos se realiza solamente si se encuentra activa la casilla del canal respectivo, aquí se muestra el estado actual del arreglo y al ser desactivada se mantiene en pantalla el último dato registrado, el efecto de todas las casillas de configuración también afecta a la función del cálculo de la FFT.

Existen muchas herramientas graficas de utilidad que pueden ser añadidas al trazador para que su manejo resulte más práctico, en función del rendimiento alcanzado sobre la placa Raspberry PI se decidirá si es necesario activarlas.

La sexta fase del programa incluirá la adquisición de datos a través del circuito integrado MCP3008, para ello se utilizó el módulo SPIDEV de Python que permite la comunicación con dispositivos SPI a través de los pines dedicados. El número máximo dispositivos SPI conectados simultáneamente puede ser 2, la lógica SPI diferencia a sus esclavos a través del pines CS, en la figura 4.10 se muestra el diagrama de conexión.

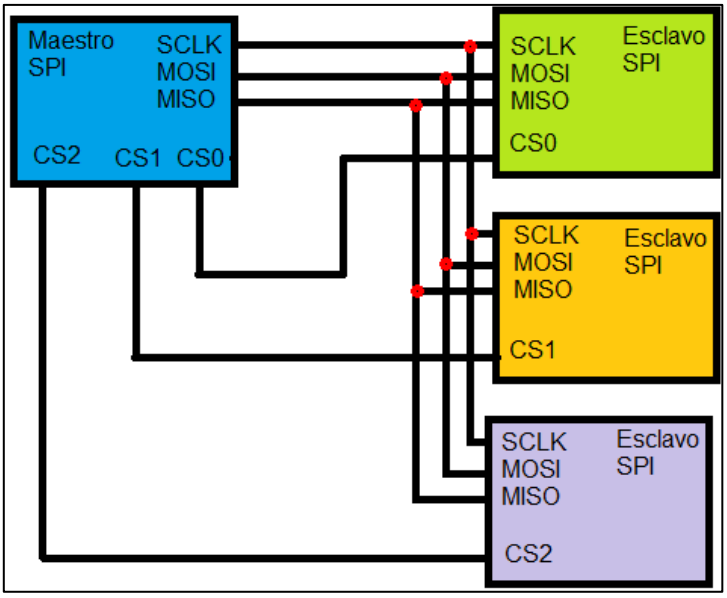


Figura 4.10 Esquema de comunicación SPI

Fuente: http://www.mct.net/faq/spi.html

4.2.2. Pruebas de Rendimiento

La primera prueba se realiza con el conversor análogo digital utilizando las instrucciones del módulo SPIDEV para realizar la comunicación a velocidad mínima, posteriormente se incluye la instrucción max_speed_hz para configurar la velocidad de trabajo del puerto. La velocidad máxima de comunicación SPI soportada por el C.I. MCP3008 según la hoja de datos del fabricante es de 3.6MHz.

La máxima tasa de muestreo es 200KSPS (muestras por segundo) a 5V, los pines GPIO de la placa Raspberry PI tienen una estricta lógica de funcionamiento de 3.3V; lo que no permite alcanzar el máximo potencial del circuito. El funcionamiento teórico máximo esperado oscilaría entre 75KHz (2,7V) y 200KHz (5V) [29].

Las pruebas realizadas manipulando la velocidad de reloj muestran que existe comunicación coherente hasta los 12MHz. Al tomar en cuenta la cantidad de bits que requiere la transmisión de la cadena y la máxima velocidad de muestreo se configuro el límite de velocidad en 9.6MHz. En la figura 4.11 se muestre el esquema de comunicación en bits requerido para realizar una comunicación con el circuito integrado MCP3008.

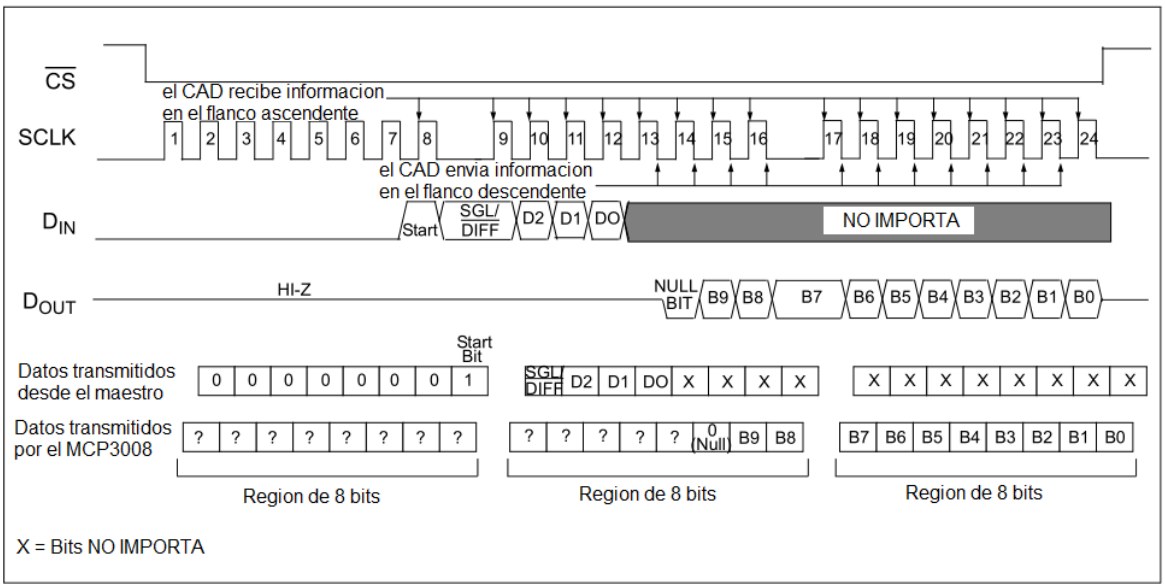


Figura 4.11 Comunicación SPI con MCP3008 usando segmentos de 8 bits

Fuente: Datasheet MCP3004/3008

La teoría de funcionamiento del puerto GPIO de Raspberry PI plantea velocidades teóricas del puerto SPI de 125Mhz como máximo, basado en la frecuencia de reloj; dada la distorsión de la señal no se logra alcanzar estos valores.

La librería SPIDEV en un programa básico desarrollado en Python para la recolección de datos de un canal a la frecuencia máxima soportada, permitió capturar un promedio de $1x10^5$ en 10 segundos, por lo tanto se realizó capturas aproximadamente cada 100 micro segundos; esto produce una frecuencia de muestreo de 10KHz.

Los resultados anteriores muestran las limitaciones de Python en la captura de datos a alta velocidad, dado que es un lenguaje de alto nivel y su ejecución requiere de distintos procesos internos. Por este motivo no es posible obtener muestras periódicas ya que el tiempo de ejecución de las instrucciones no es fijo, produciendo una variación del tiempo de cada captura, y no es viable realizar ambas tareas simultáneamente.

La adición de líneas de código para la manipulación de datos aumenta el tiempo de procesamiento en cada ciclo, debido a esto en condiciones reales sería imposible alcanzar rendimientos cercanos a los 10KHz; a esto se suma la ejecución de la interfaz gráfica, controles y trazado de curvas. La frecuencia de operación alcanzada al integrar la adquisición de datos en el programa principal difícilmente supera 1KHz.

La adquisición de datos se realizó a través un ejecutable desarrollado en C, el cual es más preciso y rápido, para eso se tomó como base el código provisto por “hallherta”, donde se configuro la velocidad máxima de 9,6Mhz [46].

El programa se modificó para que realice la captura y almacenamiento de datos a un arreglo de 200x8, para realizar capturas de 200 datos por cada uno de los 8 canales disponibles cada vez que se ejecuta. Una vez realizada la captura de un canal se transfiere el arreglo al archivo de texto correspondiente, este proceso se realiza del mismo modo para todos los canales.

Las pausas fueron retiradas para ejecutar el código a la máxima velocidad posible, de este modo el programa realiza una captura a la misma frecuencia de muestreo,

dadas las ventajas de visualización de PyQtGraph no es necesario incluir controles que permitan sintonizar el dispositivo; ya que a través del mouse se puede manipular la visualización de los datos y hacer zoom en las áreas necesarias.

4.3. Pruebas de Funcionamiento.

El proceso para determinar la velocidad máxima de muestreo que alcanza este programa se basa en la cantidad de datos que puede capturar y almacenar en un tiempo determinado, se alcanzó un promedio de 20000 datos por segundo, lo que resulta en un tiempo de 50 µs por cada medición y almacenaje registrado. A continuación se muestra en la figura 4.12 la captura de datos a 10KHz, comparada con la onda generada.

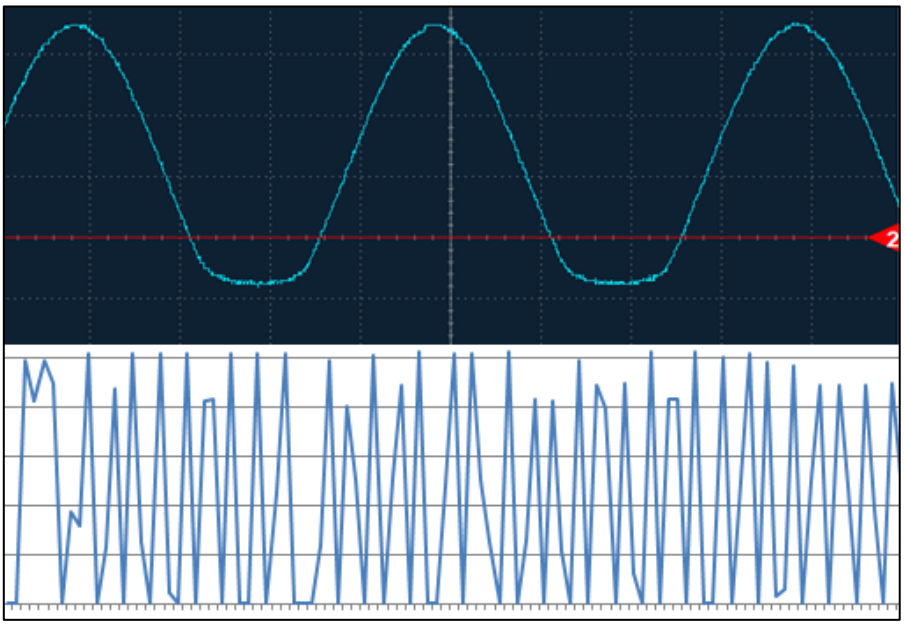


Figura 4.12 Captura de datos a 10KHz representados en EXCEL

Elaborado por: Freddy Carrillo

El teorema de Nyquist afirma que la frecuencia de muestreo debe ser al menos dos veces la frecuencia de operación, por lo tanto la frecuencia de operación del dispositivo será menor o igual a 10KHz; una señal sinusoidal a esta frecuencia no

puede ser claramente apreciada, ya que se requiere muchos más puntos para observar correctamente la señal.

A continuación en la imagen 4.13 se aprecia la captura de una señal de tipo diente de sierra a 4KHz en la parte inferior y la señal generada en la parte superior.

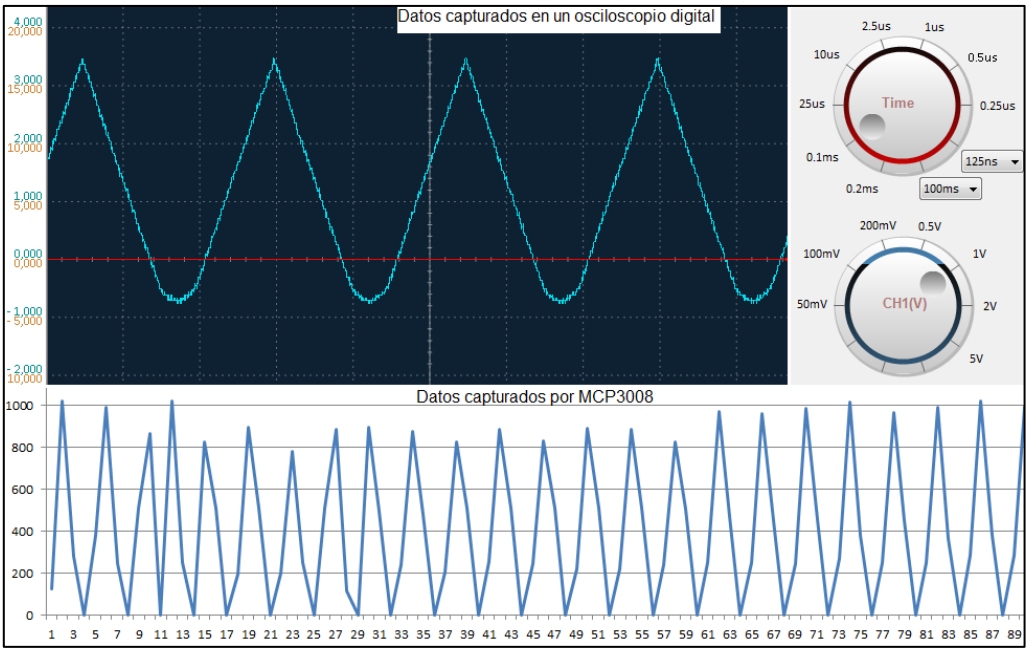


Figura 4.13 Captura de datos a 4KHz mostrados en EXCEL

Elaborado por: Freddy Carrillo

4.4 Determinar la Frecuencia Real de Funcionamiento.

El tiempo total que necesita la aplicación para realizar la captura, almacenamiento y transferencia es aproximadamente 1 segundo, utilizando la máxima velocidad posible de muestreo.

A continuación en la figura 4.14 se muestra una de las tres señales de reloj usadas para sincronizar el envió de las tres cadenas necesarias para la comunicación, ya sea de transmisión o recepción.

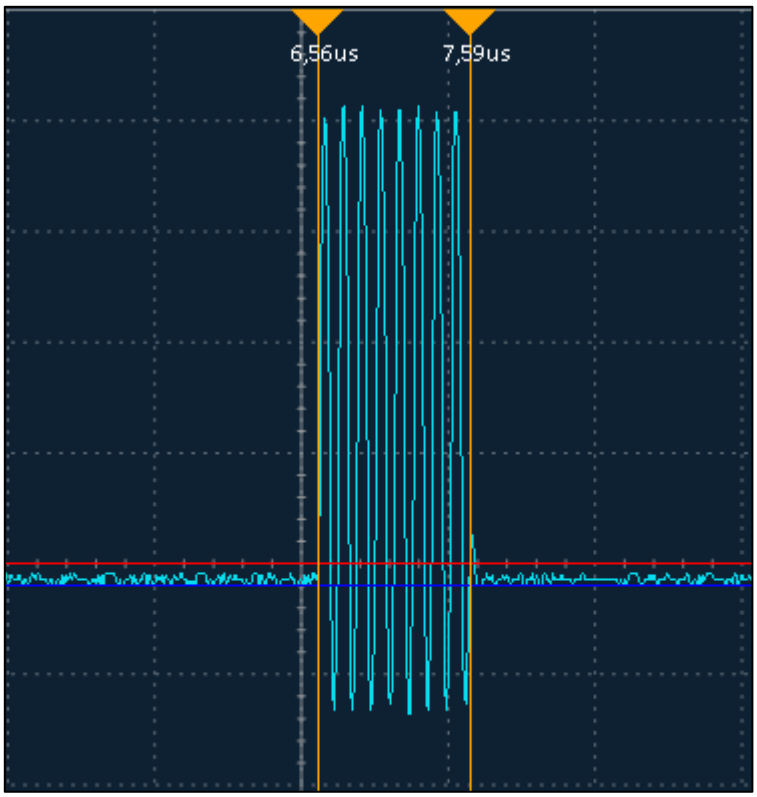


Figura 4.14 Señal de Reloj correspondiente a una cadena de 8 bits

Elaborado por: Freddy Carrillo

La velocidad de reloj generada por el pin SCLK será menor que la máxima velocidad determinada en la configuración, el método de comunicación empleado por la librería consiste en tres ráfagas de reloj conjuntamente con la información enviada o transmitida, con una frecuencia máxima de 9,6MHz tenemos que la frecuencia de la señal de reloj es:

$$t = (7{,}59 - \ 6{,}56) * 10^{\ -6} = 1.03 * 10^{-6} \qquad \text{Ec (5).}$$

Donde los valores de la ecuación 5 provienen de la figura 4.12 mostrando el intervalo de inicio y final de la señal de reloj. El tiempo de cada ciclo se obtiene al dividir el tiempo total para el número de ciclos (siempre son 8), la frecuencia de reloj es:

$$fsclk = 8/((7{,}59 - \ 6{,}56) * 10^{\ -6}) \qquad \text{Ec (6)}$$

$$fsclk = 7{,}767MHz \qquad \text{Ec (7)}$$

La frecuencia de muestreo está determinada por el intervalo en que se toman las muestras, por lo tanto es independiente de la frecuencia empleada para realizar un solo muestreo; lo importante es el tiempo entre ciclos que requiere el equipo para procesar la información, a continuación en la imagen 4.15 se observa el intervalo de tiempo entre mediciones con el programa finalizado.

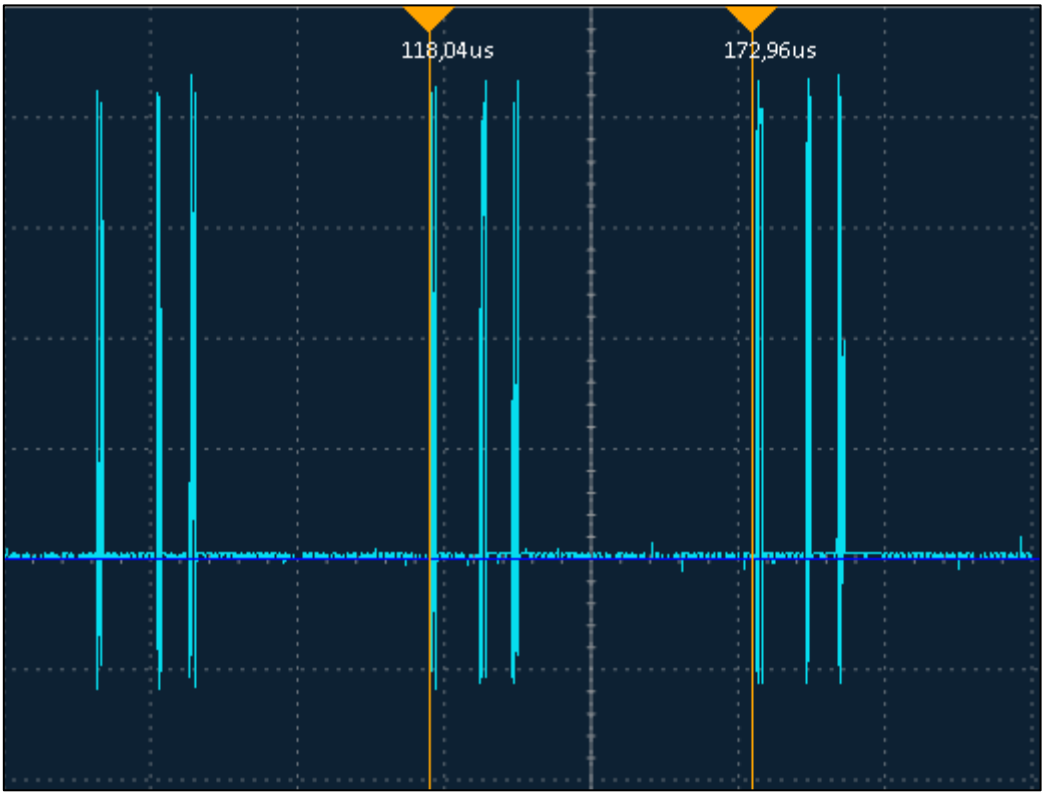


Figura 4.15 Transmisión de las señales de reloj en una comunicación normal.

Elaborado por: Freddy Carrillo

La frecuencia de muestreo real se obtiene con los datos de la Figura 4.15, donde se puede apreciar el intervalo entre muestras; a continuación se realizan los cálculos correspondientes:

$$fm = \frac{1}{(172.96 - 118{,}04) * 10^{-6}}\ [Hz] \qquad \text{Ec (8)}$$

Por lo tanto la frecuencia de operación del dispositivo será:

$$f = \frac{fm}{2}\ [Hz] \qquad \text{Ec (9)}$$

$$f = 9160{,}864786\ Hz\ \approx 9{,}160 Khz \qquad \text{Ec (10)}$$

El programa incluye controles para modificar los parámetros de amplitud y cantidad de muestras acumuladas, estos controles tienen valores predeterminados; en el caso de requerir un valor específico de amplitud se puede configurar en el selector secundario, para ingresar señales de amplitudes fuera del rango se requiere un divisor de tensión.

La resolución de la Transformada de Fourier depende directamente de la cantidad de puntos que posea el arreglo utilizado para el trazado, pero a la vez un arreglo más grande significa más consumo de recursos, por este motivo se añadió un control de la cantidad de muestras almacenadas, en la figura 4.16 se puede apreciar la comparación de un cálculo con pocas muestras (izquierda) y con varias muestras (derecha) en el cálculo de la FFT.

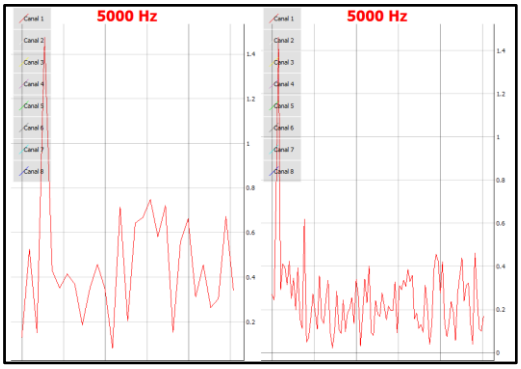


Figura 4.16 Comparación de muestras en el cálculo de la FFT.
Elaborado por: Freddy Carrillo

4.5. Diseño del Hardware

La comunicación de alta velocidad entre Raspberry Pi y el circuito integrado MCP3008 a través del bus SPI requiere una conexión directa a los pines GPIO, por lo tanto es necesario implementar una protección a la entrada del conversor análogo digital para evitar daños ante cualquier error humano que pueda ocasionarse, es por este motivo que se utilizó Buffers o Seguidores de voltaje en

base a transistores NPN dado que son fáciles de implementar y reemplazar. A continuación en la Figura 4.17 se muestra el diagrama esquemático.

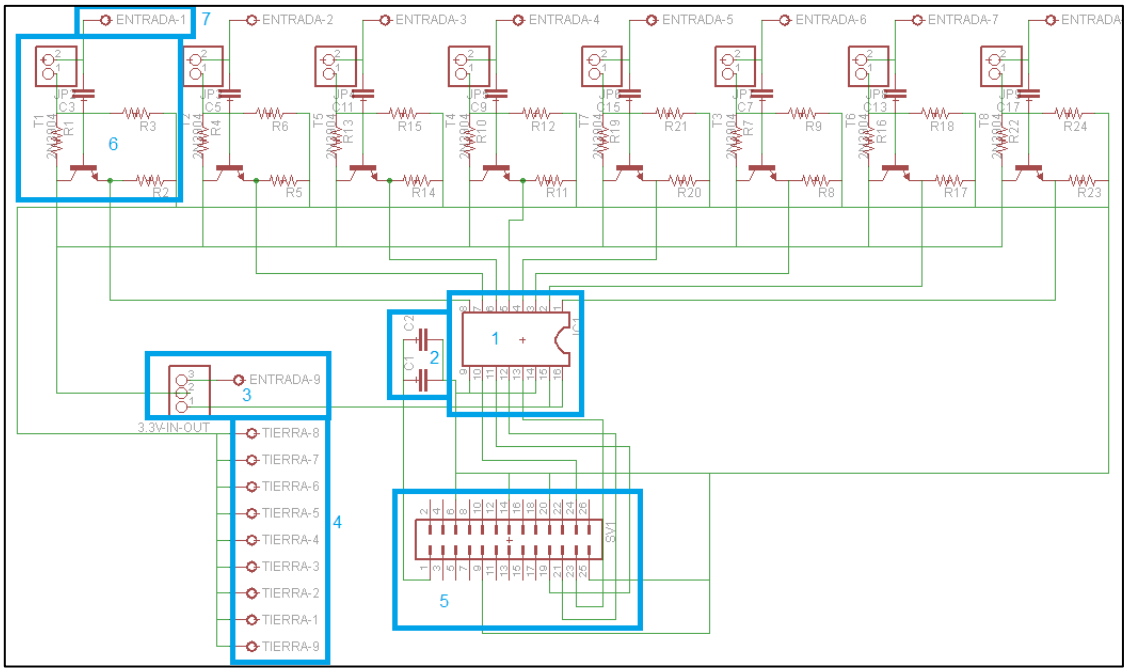


Figura 4.17 Esquema de conexión de la tarjeta para adquisición de datos

Elaborado por: Freddy Carrillo

Las diferentes partes del circuito en la figura 4.16 son:

1. Conversor Análogo Digital MCP3008.
2. Filtros para estabilización de voltaje y reducción de ruido.
3. Selector de fuente de energía para los seguidores de voltaje, esta opción se debe a las especificaciones eléctricas de Raspberry PI; puede entregar un máximo de 50mA por pin.
4. Terminales de conexión a tierra para puntas externas.
5. Pines de entrada y salida (GPIO).
6. Seguidor de voltaje con 2N3904, en esta etapa se puede cortocircuitar el filtro a la entrada; con el propósito de eliminar este efecto en caso de ser necesario.
7. Terminales de conexión para entrada de señal.

El seguidor de voltaje mostrado la figura 4.17, permite acoplar cualquier señal aplicada a los terminales de entrada a la lógica de funcionamiento de conversor análogo digital, en caso de un sobre voltaje este circuito también permite recortar la señal de entrada y de esta manera no causar daños.

La fuente suministra corriente en la medida que la consuma el circuito, por lo tanto existe un rango amplio de valores de resistencia que se pueden utilizar para la implementación del seguidor de voltaje; tras la experimentación con diferentes resistencias se llegó a la conclusión que los valores adecuados son R1,R2=100KΩ y R3=47KΩ. En la figura 4.18 se muestra el seguidor de voltaje con los valores establecidos y su respuesta (azul) en comparación con la señal original (amarillo).

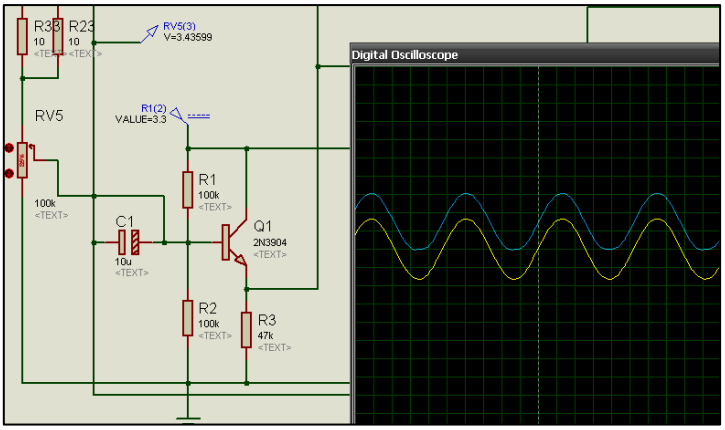


Figura 4.18 Simulación de seguidor de voltaje

Elaborado por: Freddy Carrillo

4.6. Funciones adicionales y Ajustes Finales

Como segunda opción de funcionamiento se incorporó la adquisición de datos a través de la plataforma Arduino, realizando una captura de datos usando un esquema similar al empleado en el circuito integrado MCP3008; según la página oficial de Arduino la máxima frecuencia de muestreo es 10KHz [48].

La transferencia de datos se realizó a través del puerto USB, por lo tanto dada la flexibilidad de esta comunicación solamente se habilitan los canales

seleccionados; guardando la información en una matriz de 100 datos, la activación se produce al reconocer el carácter correspondiente a cada canal. A continuación en un bucle se transfieren todos los datos desde la memoria del microcontrolador hacia el software de Python para posteriormente ser cargados en el trazador [49, 50, 51, 52, 53].

Esta función posee una menor frecuencia de muestreo, sin embargo es más exacto dado que el proceso se realiza de manera separada, también es más fácil de implementar y reemplazar, ya que solamente requiere una conexión USB. El mayor inconveniente ocurre cuando la conexión USB es interrumpida intempestivamente lo que provoca un colapso del software. A continuación en la figura 4.19 se muestra la captura de datos utilizando esta función.

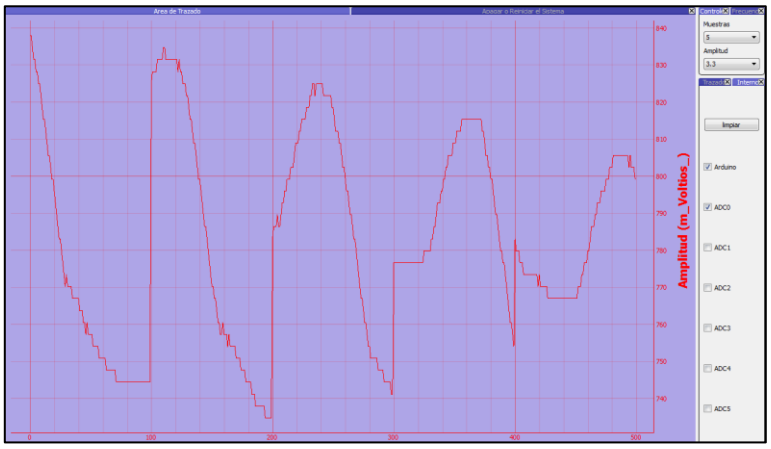


Figura 4.19 Función de captura utilizando placa Arduino

Elaborado por: Freddy Carrillo

El dispositivo finalizado trabaja con el circuito Integrado MCP3008, en dos modos de funcionamiento, el primero con la frecuencia de trabajo calculada anteriormente y el segundo utiliza las funciones de captura del módulo SPIDEV de Python, en este modo no se especifica una frecuencia de funcionamiento ya que está pensado para su uso como osciloscopio simple y la captura con el módulo SPIDEV no garantiza una frecuencia estable.

El programa antes descrito se ejecuta automáticamente al iniciar el sistema, la función donde está la comunicación con Arduino no se incluyó en el programa principal, ya que cualquier interrupción en la conexión genera un error critico; por lo tanto se realizó un segundo ejecutable donde se añade esta función.

Descripción de los controles de la interfaz

La distribución de las áreas de trabajo está diseñada para maximizar el espacio del trazado, por lo tanto independientemente de la resolución de pantalla el área de trazado siempre será más grande; algunos controles o funciones se encuentran en segundo plano ya que otras áreas de trabajo se encuentran solapándolas. En la Figura 4.20 se indican las diferentes partes de la interfaz del software desarrollado.

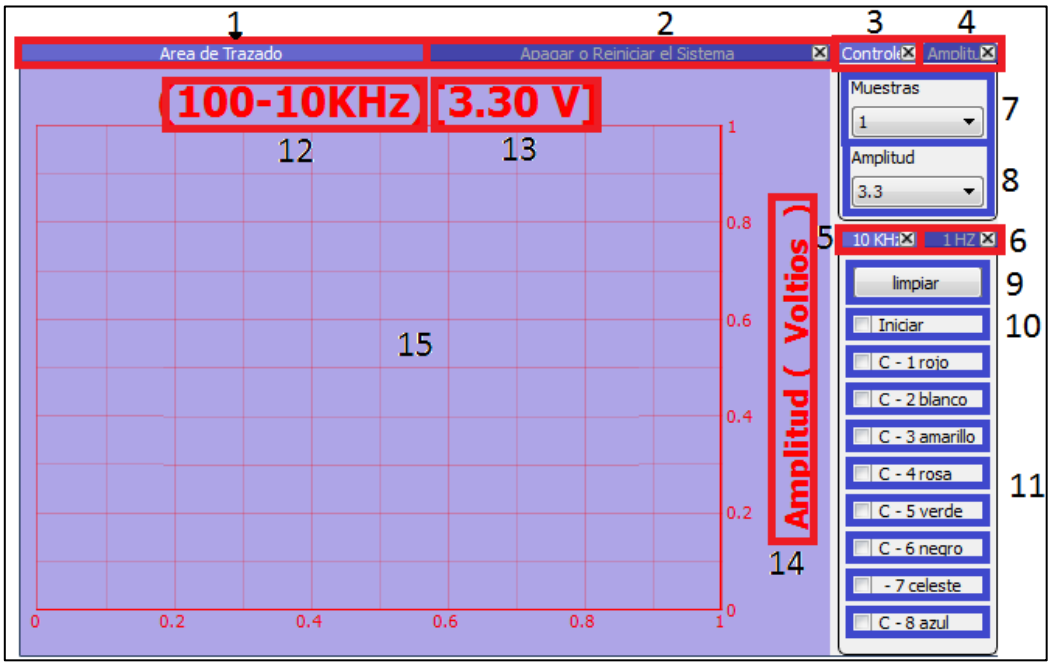


Figura 4.20 descripción de la interfaz gráfica principal
Elaborado por: Freddy Carrillo

- Selección de las distintas áreas de trabajo (1 – 6).
- Selección de la cantidad de muestras almacenadas (7)
- Selección de Valores definidos de Amplitud (8)
- Botón para limpiar datos del trazador (9)
- Casilla de activación para el ciclo de captura (10)

- Casilla de activación de los canales respectivos (11)
- Información sobre el modo de funcionamiento (12)
- Voltaje establecido (13)
- Información de la Amplitud en el plano (14)
- Trazador (15)

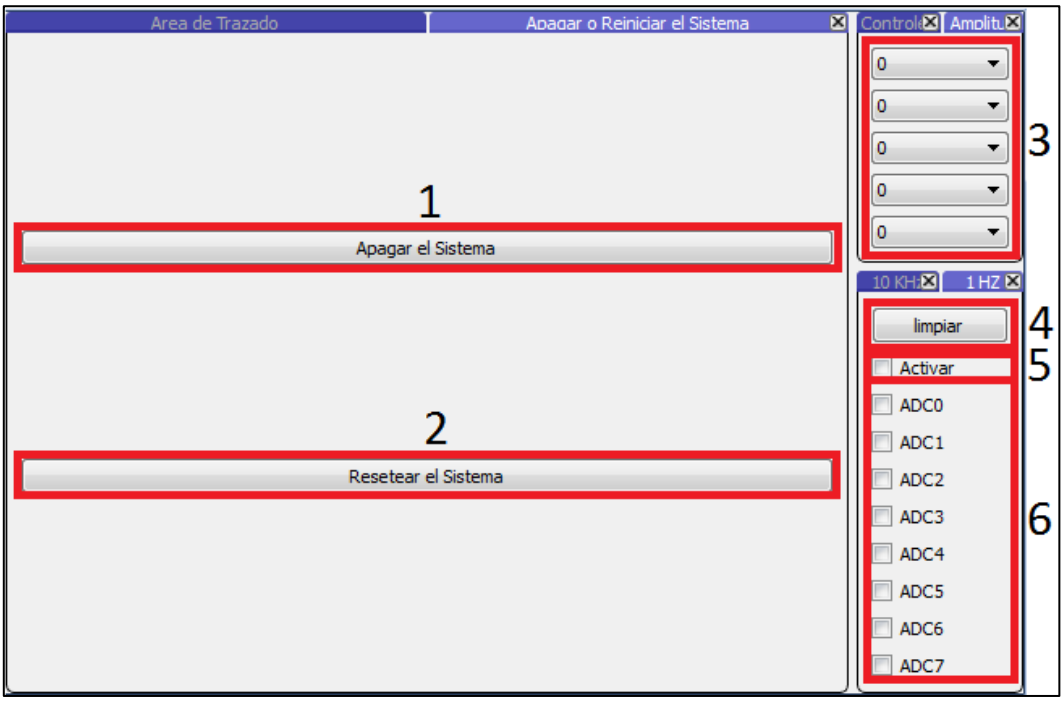


Figura 4.21 descripción de la interfaz gráfica secundaria
Elaborado por: Freddy Carrillo

En la Figura 4.21 se indican las diferentes partes en segundo plano de la interfaz del software desarrollado:

- Botón para apagado del sistema operativo (1)
- Botón para reseteo del sistema operativo (2)
- Selectores de configuración de voltaje especifico (3)
- Botón para limpiar datos del trazador (4)
- Casilla de activación para el ciclo de captura (5)
- Casilla de activación de los canales respectivos (6)

Ajustes Finales

Una vez finalizado el software y hardware correspondiente, se procedió a desactivar las funciones que consumen recursos de manera innecesaria como las consolas virtuales adicionales, desactivar el servicio SSH y deshabilitar IPv6 [54, 55].

Interfaz Visual.

La forma óptima para visualizar los datos obtenidos del Raspberry Pi, es una pantalla de 7 pulgadas que permite portabilidad y una resolución adecuada para la comodidad del usuario.

Al realizar una búsqueda de los modelos disponibles con puerto HDMI en la tienda Amazon.com se encontró dos pantallas de resolución de 800x480 y 1200x800; adicionalmente cuentan con puertos VGA y RCA, por lo tanto son completamente compatibles con Raspberry Pi. A continuación en la figura 4.22 se muestra la pantalla adquirida.

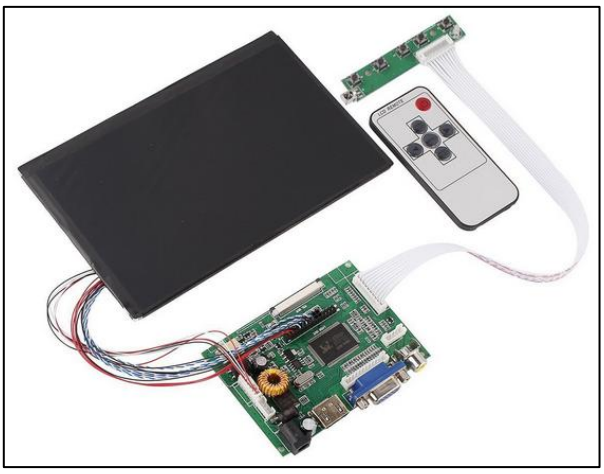


Figura 4.22 Pantalla de 7 pulgadas de 1800x800

Fuente:http://www.amazon.com/gp/product/B00JOY5PGM/ref=oh_aui_detailpage_o05_s00?ie=UTF8&psc=1

4.7 Dispositivo Finalizado

En las siguientes imágenes se muestra el dispositivo finalizado y se muestran los distintos módulos que lo integran. En la Figura 4.23 se muestra la pantalla LCD de 7 pulgadas montada en la caja y ejecutando el software desarrollado, para proteger la pantalla de cualquier daño externo se la cubrió con una placa de acrílico.

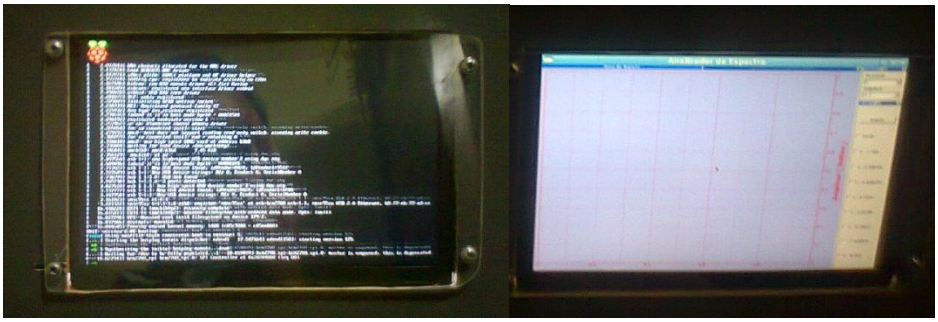


Figura 4.23 Pantalla LCD en el montaje final.
Elaborado por: Freddy Carrillo.

En la imagen 4.24 se muestra el montaje de la fuente de alimentación en la parte izquierda y el módulo que permite operar la pantalla LCD en la derecha.

Figura 4.24 Fuente de energía y modulo controlador de LCD.
Elaborado por: Freddy Carrillo.

La tarjeta de adquisición de datos desarrollada para el puerto GPIO de Raspberry Pi está asegurada y cableada hacia los conectores de cable coaxial situados en la parte exterior, la tarjeta se encuentra alimentada por la fuente principal de 3,3V; como se puede apreciar en la figura 4.25.

Figura 4.25 Tarjeta de Adquisición de Datos

Elaborado por: Freddy Carrillo.

Los filtros opcionales están colocados sobre los conectores de cable, como se muestra en la figura 4.26, cuando no se encuentran conectadas las puntas a algún circuito se filtra ruido proveniente de los conectores coaxiales.

Figura 4.26 Conectores de cable coaxial y filtros opcionales.

Elaborado por: Freddy Carrillo.

El acceso a los puertos USB y Ethernet se encuentra en la parte superior y son claramente visibles y sin obstrucciones para evitar daños al equipo, en la figura 4.27 se muestra esta parte del equipo.

Figura 4.27 Ranura para conexión de periféricos

Elaborado por: Freddy Carrillo.

El dispositivo fue diseñado para la utilización de los estudiantes, por lo tanto sus componentes deben ser fácilmente reemplazables, es por esta razón que se decidió utilizar puntas construidas en base a cable coaxial RG-59 con pinzas tipo lagarto, ya que es muy barato y ofrece cierto aislamiento.

En la imagen 4.28 se muestran varias puntas construidas de este modo, las cuales son más baratas que las puntas tradicionales de osciloscopio, que tienen un valor mínimo de 20 dólares por unidad en el mercado local.

Figura 4.28 Puntas de Conexión

Elaborado por: Freddy Carrillo.

4.8. Análisis de Costos

El proyecto ha sido desarrollado utilizando materiales económicos, pero esto sin comprometer demasiado el rendimiento del equipo, con el propósito de brindar al

estudiante un dispositivo que pueda ser utilizado a lo largo de su formación académica.

Existen Analizadores de Espectro y Osciloscopios en el mercado, pero su costo es elevado si se hace referencia a equipos profesionales, en el caso de los equipos más económicos y accesibles, estos no poseen una larga vida útil dada la fragilidad de su diseño.

Un Analizador de Espectro también puede ser utilizado como osciloscopio, pero no siempre se puede trabajar inversamente, la función de Analizador de Espectro se encuentra en su mayoría en los Osciloscopios Digitales. En la tabla 4.14 se realizó una comparación de los distintos equipos disponibles en el mercado con el dispositivo desarrollado.

Tabla 4.14 Especificaciones de los equipos disponibles en el mercado local

Modelo	Siglent SDS1102DL	Siglent SHS810	Labvolt 797	DS201 Mini 2.8”
Tipo	Estacionario	Portátil	Estacionario	Portátil
Frecuencia de muestreo	50G/s	1G/s		1M/s
Ancho de banda	100MHz	60MHz - 200MHz	40Mhz	200KHz
Análisis de Fourier	SI	SI	NO	SI
Voltaje Máximo	1000V (DC) 750(AC)	1000V (DC) 750(AC)	300V(DC) 250V(AC)	80V(AC)
Otras características	Puerto USB, Múltiples lenguajes	Puerto USB, pantalla LCD de 5,7 pulgadas	Calibración de brillo en el eje Z	LCD 2.8", Ranura para tarjeta SD
Precio Base	660	720$ hasta 860$	250$ Usado	150$ 100$(usado)
Precio Final	660$ + 60$	720$ hasta 860$ + 60$	250$ + 60$	150$

Elaborado por: Freddy Carrillo.

Entre los Osciloscopios digitales y analógicos de altas prestaciones se tiene los siguientes modelos, que se encuentran disponibles en la tienda en línea MercadoLibre.com.ec:

- Siglent SDS1102DL es un osciloscopio de mesa, resistente, eficiente, puede realizar el análisis de Fourier y tolera un amplio rango de voltaje; el costo final del equipo son 720$.
- Siglent SHS810, es un osciloscopio digital portátil, posee memoria interna, puerto USB, tolera un amplio rango de voltajes, dependiendo de la frecuencia de muestreo máxima su valor puede llegar a los 860$.
- Labvolt 797 es un osciloscopio Analógico, no posee la función de análisis de Fourier, el voltaje y su ancho de banda es menor que los equipos anteriores.

DS201 Mini 2.8”, es un osciloscopio económico de bolsillo, con una autonomía de 2 horas, consta de dos canales y una frecuencia de muestreo de 1M/s, con un ancho de banda de 200KHz. Su vida útil es corta ya que es sensible ante cortocircuitos o golpes, las posibilidades de repararlo son muy bajas.

Las prestaciones de los equipos anteriores superan ampliamente a las del proyecto, sin embargo el costo de construcción del dispositivo es un factor importante a la hora de adquirir equipos de laboratorio; por este motivo este proyecto es buena opción ante osciloscopios de altas prestaciones. A continuación en la tabla 4.15 se detalla las características del equipo desarrollado [56, 57, 58, 59, 60, 61]

Tabla 4.15 Características del equipo desarrollado.

Tipo	Estacionario
Frecuencia de Muestreo	18K/s
Ancho de Banda	7,9KHz
Análisis de Fourier	SI
Voltaje Máximo	3,3V
Otras Características	Compatibilidad con Arduino, Módulos Reemplazables
Costo de los Materiales	120$

Elaborado por: Freddy Carrillo

La facilidad de reparación, el costo de los repuestos y la durabilidad son factores decisivos, por lo cual en estos aspectos el equipo supera a las opciones disponibles en el mercado. Adicionalmente las puntas estándar de osciloscopio cuestan en promedio 20 dólares y las puntas fabricadas artesanalmente son más económicas.

Costo de Implementación del Analizador de Espectro.

El presente proyecto está elaborado bajo los conceptos de software y hardware libre, sin embargo es necesario determinar el costo total del desarrollo tomando en cuenta los equipos adicionales requeridos, los materiales extras, el pago de impuestos por la importación de ciertos equipos, así como también las horas de programación empleadas; todo esto se detalla en la tabla 4.16.

Tabla 4.16 Costos del equipo desarrollado

Ítem	Descripción	Cantidad	Valor/u	Valor total
1	Raspberry pi b	2	$ 35,00	$ 70,00
2	Pantalla LCD 7"	1	$ 75,00	$ 75,00
3	Fuente de computador de 400W	1	$ 16,00	$ 16,00
4	Cable HDMI y adaptadores		$ 25,00	$ 25,00
5	Mcp3008	3	$ 5,00	$ 15,00
6	Baquelita	5	$ 2,00	$ 10,00
7	Elementos electrónicos varios		$ 50,00	$ 50,00
8	Conectores		$ 15,00	$ 15,00
9	Chasis	1	$ 35,00	$ 35,00
10	Multímetro	1	$ 20,00	$ 20,00
11	Osciloscopio	1	$ 200,00	$ 200,00
12	Cables varios		$ 10,00	$ 10,00
13	Periféricos	2	$ 10,00	$ 20,00
14	Horas de programación	1080	$ 5,00	$ 5.400,00
15	Gastos varios		$ 100,00	$ 100,00
	TOTAL			$ 6.061,00

Elaborado por: Freddy Carrillo

4.9 Comprobación del funcionamiento utilizando MATLAB

Una vez finalizado el dispositivo es necesario comprobar su funcionamiento, para ello se utilizó un software matemático que permita calcular la Transformada Rápida de Fourier de los datos capturados a diferentes frecuencias; se tomó muestras de 300 datos para mejorar la resolución del espectro obtenido.

El cálculo de la Transformada Rápida de Fourier se realiza mediante un algoritmo optimizado para computadores, en el cual es necesario configurar ciertos parámetros como: la matriz de datos, la cantidad de datos de la matriz, la frecuencia, y el tiempo de captura. A continuación en la tabla 4.17, se muestra el código utilizado.

Tabla 4.17 Calculo del espectro en Matlab

Código	Función
X= [125,……..]	Matriz de datos de señal de 4KHz
X2= [0……..]	Matriz de datos de señal de 10KHz
X3= [1022……]	Matriz de datos de señal de 1KHz
X4= [1005……]	Matriz de datos de señal de 3KHz
Fs=18320	Frecuencia de muestreo
T=1/Fs	Tiempo de muestro
L=300	Cantidad de puntos
t=(0:L-1)*T	Tiempo de la captura
NFFT = 2^nextpow2(L); % Next power of 2 from length of y	entrega la potencia de 2 del tamaño de la matriz
Y = fft(x3,NFFT)/L;	Calcula la Transformada Rápida de Fourier para la matriz designada, en función de la cantidad de puntos
f = Fs/2*linspace(0,1,NFFT/2+1);	Crea la matriz donde se muestran el rango de frecuencias del calculo
plot(f,2*abs(Y(1:NFFT/2+1)))	Se traza en función del F(contiene frecuencias que aparecen en la FFT) y Y(cálculo de la FFT)
title('Analisis del Espectro en Matlab F=1KHz')	Título del trazado
xlabel('Frequencia (Hz)')	Etiqueta en eje X
ylabel('|Y(f)|')	Etiqueta en eje Y

Elaborado por: Freddy Carrillo

Se realizó la captura de 3 señales a distintas frecuencias, todas tienen una amplitud de 3.3V, ya que si se supera este voltaje el buffer colocado a la entrada del Conversor Análogo Digital la recortará para evitar daños al equipo.

La primera captura corresponde a una señal senoidal de 10KHz, esta frecuencia se encuentra por encima de la frecuencia de corte del dispositivo; por lo tanto el espectro no se podrá visualizar adecuadamente y solamente se observará múltiples armónicos.

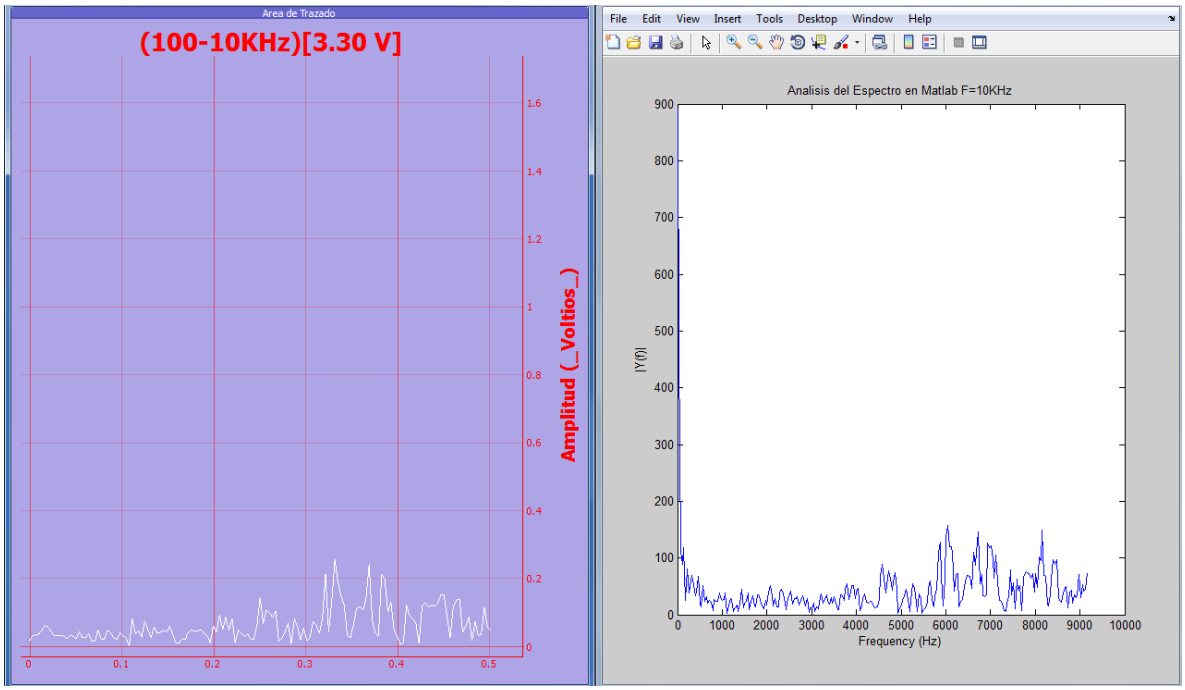


Figura 4.29. Calculo del Espectro de señal de 10KHz

Elaborado por: Freddy Carrillo

A continuación se calcula el espectro de tres señales diente de sierra con frecuencias de 1 KHz, 3Khz, 4 KHz respectivamente. Estas señales se encuentran dentro del rango de operación del dispositivo, por lo tanto su espectro es claramente visible.

Como se muestra en la Figura 4.30, el espectro calculado es similar al mostrado por el software del Analizador de Espectro, sin embargo se observa una componente de baja frecuencia que no es apreciada en el software desarrollado,

esto se debe a que el sistema posee una frecuencia mínima de funcionamiento de 100Hz; por lo tanto esta componente de baja frecuencia es discriminada.

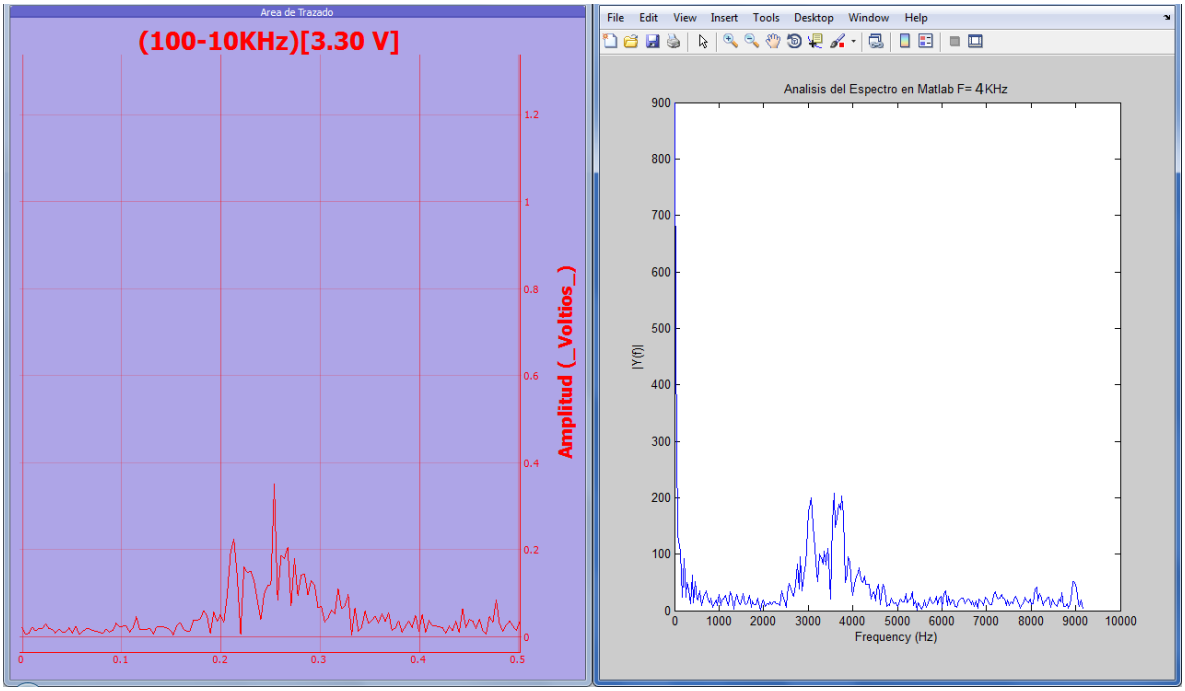


Figura 4.30. Calculo del Espectro de señal de 4KHz

Elaborado por: Freddy Carrillo

En la Figura 4.31 se aprecia un caso similar ya que el espectro calculado coincide, se muestra una componente de baja frecuencia que es discriminada por el software desarrollado.

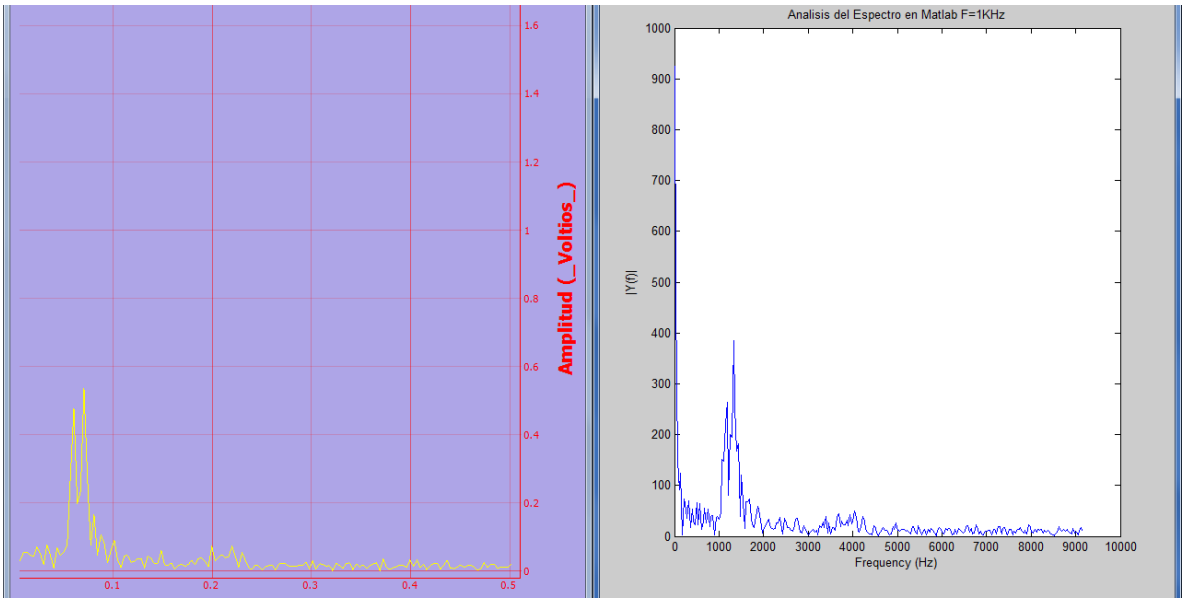


Figura 4.31. Calculo del Espectro de señal de 1KHz

Elaborado por: Freddy Carrillo

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- La tarjeta de adquisición de datos y la entrada opcional con Arduino brindan flexibilidad y facilidad de reparación.
- El C.I. MCP3008 tiene una frecuencia de muestro máxima de 200KHz, sin embargo no es alcanzable dado que el kernel Linux ejecuta múltiples procesos simultáneamente que reducen la velocidad hasta 18,32KHz.
- La frecuencia de muestreo, no está relacionada con la velocidad de escritura de la tarjeta SD, ya que los datos de conversión se almacenan en la memoria RAM y luego son transferidos al programa principal.
- Los módulos Python permiten una rápida implementación con gran cantidad de herramientas disponibles y mucha fluidez en la ejecución.

5.2 RECOMENDACIONES

- La fuente de energía y los buffers de entrada proveen cierta protección a la placa Raspberry Pi, sin embargo no se debe forzar demasiado las mismas.
- El sobrecalentamiento del procesador puede provocar daños, por lo tanto es necesario no utilizar la conexión de red mientras funcione la tarjeta de adquisición de datos, para evitar sobrecargas de corriente.
- Al desarrollar o instalar nuevos programas, es preferible utilizar otra tarjeta SD para evitar disminuir el rendimiento del equipo y con esto la frecuencia máxima de operación.
- El equipo debe ser apagado correctamente para evitar daños a los archivos del sistema operativo.

REFERENCIAS

[1] Secretaría Nacional de Educación Superior Ciencia Tecnología e Innovación. (2013) educacionsuperior.gob.ec. [Online]. http://www.educacionsuperior.gob.ec/cerca-de-1700-millones-usd-para-la-educacion-superior-en-el-2013/

[2] Jorge Doring, Analizador de Espectro de Frecuencias, Quito, Ecuador: Escuela Politécnica Nacional del Ecuador, 1977.

[3] Ildefonso Hernández, Analizador Dinámico de Señales, México, México: Universidad Tecnológica de la Mixteca, 1996

[4] Pedro Merchán, Sistema de adquisición de datos para convertir a un computador personal en un analizador de espectros, Quito, Ecuador: Escuela Politécnica Nacional del Ecuador, 2000

[5] Francisco Vilatuña, Analizador de Espectros Utilizando la transformada rápida de Fourier en un microprocesador DSP, Quito, Ecuador: Escuela Politécnica Nacional del Ecuador, 2002

[6] Andrés Jiménez, Amaia de Miguel, Javier Villelas, Analizador de un Espectro Musical, Madrid, España: Universidad Complutense de Madrid, 2005

[7] Héctor Munguia, Implementación de un Analizador de Espectro para Frecuencias de Audio Utilizando un Procesador Digital de Señales (DSP), Guatemala, Guatemala: Universidad de San Carlos, 2006.

[8] Alejandro Soberanis, Diseño y Construcción de un Analizador de Espectros Usando una Plataforma Basada en FPGA), México, México: Instituto Politécnico Nacional, 2008.

[9] Javier Arichábala, Renato Zea, Analizador Dinámico, Cuenca, Ecuador: Universidad Tecnológica de Cuenca, 2006

[10] Agilent Technologies. (2012, Octubre) IEEE. [Online]. Http://ewh.ieee.org/r5/denver/sscs/Presentations/2012_10_Agilent1.pdf

[11] José Miguel Miranda, José Luis Sebastián, Manuel Sierra, and José Margineda, Ingeniería de Microondas Técnicas Experimentales, Primera ed., Isabel Capella, Ed. Madrid, España: Pearson Educación, S.A., 2002.

[12] Martha Cecilia Guzmán Zapata, Matemáticas Especiales para Ingeniería. Medellín, Colombia: Instituto Tecnológico Metropolitano, 2008.

[13] Alan V. Oppenheim, Señales y Sistemas. Segunda Edición, Pearson Editorial

[14] L. Sevgi. (2006, Septiembre) Numerical Fourier Transforms DFT and FFT. [Online]. http://ewh.ieee.org/reg/8/news/articles/sep06/literacy_test.html

[15] What is a Raspberry PI? (2012, Octubre) IEEE. [Online]. http://www.raspberrypi.org/help/what-is-a-raspberry-pi/

[16] Raspberry PI. (2012) [Online]. http://www.raspberryshop.es/

[17] BeagleBone [Online]. http://elinux.org/Beagleboard:BeagleBoneBlack

[18] PcDuino (2013, Mayo) [Online]. http://www.linksprite.com/?page_id=812

[19] Banana PI (2013, Abril) [Online]. http://www.bananapi.org/p/product.html

[20] Banana PI es una placa compatible con Raspberry PI (2014, Mayo) [Online] http://www.cnx-software.com/2014/04/20/banana-pi-is-a-raspberry-pi-compatible-board-fitted-with-an-allwinner-a20-soc/

[21] HummingBoard (2012, Enero) [Online]. http://liliputing.com/2014/07/solidruns-hummingboard-credit-card-sized-computer-launches-for-45-and-up.html

[22] Udoo (2012, Febrero) [Online]. https://www.kickstarter.com/projects/udoo/udoo-android-linux-arduino-in-a-tiny-single-board

[23] E. Soria, M, Martínez, Tratamiento Digital de Señales. Segunda Edición, Prentice Hall

[24] Constantito Pérez, José Zamanillo, Sistemas de Telecomunicación, Universidad de Cantabria 2007

[25] Digitalización. [Online]. http://www2.udec.cl/~lsalazarv/digitalizacion.html

[26] Raspberry Pi - Model B+ (2014, Septiembre) [Online] https://www.modmypi.com/raspberry-pi-model-b-plus

[27] RPi Low-level peripherals (2014, Noviembre) [Online] http://elinux.org/RPi_Low-level_peripherals

[28] SPI (2014, Agosto) [Online] http://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md

[29] MICROCHIP MCP3008-I/P 10BIT ADC, 2.7V, 8CH, SPI, 16DIP (2014) [Online] http://be.farnell.com/webapp/wcs/stores/servlet/ProductDisplay?catalogId=15001&langId=33&urlRequestType=Base&partNumber=1627174&storeId=10154

[30] ANALOG DEVICES AD7366BRUZ-5 IC, 12BIT ADC DUAL, SMD, TSSOP24 (2014) [Online] http://be.farnell.com/webapp/wcs/stores/servlet/ProductDisplay?catalogId=15001&langId=33&urlRequestType=Base&partNumber=1498671&storeId=10154

[31] ANALOG DEVICES AD9220ARSZ IC, ADC, 12BIT, 10MSPS, 28SSOP (2014) [Online] http://be.farnell.com/webapp/wcs/stores/servlet/ProductDisplay?catalogId=15001&langId=33&urlRequestType=Base&partNumber=2305579&storeId=10154

[32] ANALOG DEVICES AD9201ARSZ IC, 10BIT ADC, SMD, 9201, SOIC28 (2014) [Online] http://be.farnell.com/webapp/wcs/stores/servlet/ProductDisplay?catalogId=15001&langId=33&urlRequestType=Base&partNumber=1079291&storeId=10154

[33] ANALOG DEVICES AD9283BRSZ-50 ADC, 8BIT, 50MSPS, SSOP-20 (2014) [Online] http://be.farnell.com/webapp/wcs/stores/servlet/ProductDisplay?catalogId=15001&langId=33&urlRequestType=Base&partNumber=2376585&storeId=10154

[34] ANALOG DEVICES AD9283BRSZ-100 ADC, 8BIT, 100MSPS, SSOP-20 (2014) [Online] http://be.farnell.com/webapp/wcs/stores/servlet/ProductDisplay?catalogId=15001&langId=33&urlRequestType=Base&partNumber=2376612&storeId=10154

[35] ANALOG DEVICES AD5541ABRMZ IC, DAC, 16/12BIT, 10MSOP (2014) [Online] http://be.farnell.com/fr-BE/analog-devices/ad5541abrmz/ic-dac-16-12bit-10msop/dp/1843653

[36] STMICROELECTRONICS USBDF01W5 FILTER, LOW PASS, 1GHZ, SOT-323-5 (2014) [Online] http://be.farnell.com/webapp/wcs/stores/servlet/ProductDisplay?catalogId=15

001&langId=33&urlRequestType=Base&partNumber=2341665&storeId=10154

[37] LINEAR TECHNOLOGY LTC1563-2CGN#PBF FILTER, LOWPASS, ACTIVE RC, SSOP16 (2014) [Online] http://be.farnell.com/webapp/wcs/stores/servlet/ProductDisplay?catalogId=15001&langId=33&urlRequestType=Base&partNumber=1417718&storeId=10154

[38] NOOBS 8GB SD Card (2014) [Online] http://swag.raspberrypi.org/products/noobs-8gb-sd-card

[39] Welcome to RISC OS (2014, Abril) [Online] https://www.riscosopen.org/wiki/documentation/show/Welcome%20to%20RISC%20OS%20Pi

[40] FAQS (2014) [Online] http://www.raspberrypi.org/help/faqs/#software

[41] Guido Van Rossum, Tutorial de Python, Argentina, 2009.

[42] PyQtGraph Scientific Graphics and GUI Library for Python (2011) [Online] http://www.pyqtgraph.org/

[43] SpiDev Documentation (2013, Diciembre) [Online] http://tightdev.net/SpiDev_Doc.pdf

[44] NumPy (2013) [Online] http://www.numpy.org/

[45] What is PyQt? (2013) [Online] http://www.riverbankcomputing.co.uk/software/pyqt/intro

[46] Interfacing an SPI ADC (MCP3008) chip to the Raspberry Pi using C++ (spidev) (2013, Julio) [Online]

http://hertaville.com/2013/07/24/interfacing-an-spi-adc-mcp3008-chip-to-the-raspberry-pi-using-c/

[47] Raspberry Pi real-time kernel [SD image available] (2014, mayo) [Online] http://www.emlid.com/raspberry-pi-real-time-kernel/

[48] analogRead (2015) [Online] http://arduino.cc/en/pmwiki.php?n=Reference/analogRead

[49] Arduino Python Communication via USB (2014, marzo) [Online] http://www.instructables.com/id/Arduino-Python-Communication-via-USB/?lang=es&ALLSTEPS

[50] Communicating between Python and Arduino with pyserial (2011, Diciembre) [Online] www.elcojacobs.com/communicating-between-python-and-arduino-with-pyserial/

[51] Raspberry Pi and Arduino Connected Over Serial GPIO (2013, Mayo) [Online] blog.oscarliang.net/raspberry-pi-and-arduino-connected-serial-gpio/

[52] Arduino + Raspberry Pi – Lectura de datos (2013, Agosto) [Online] https://geekytheory.com/arduino-raspberry-pi-lectura-de-datos/

[53] Arduino + Raspberry Pi – RaspDuino (2013, Julio) [Online] geekytheory.com/arduino-raspberry-pi-raspduino/

[54] Raspberry PI Raspbian Tunning (2012, julio) [Online] https://extremeshok.com/1081/raspberry-pi-raspbian-tuning-optimising-optimizing-for-reduced-memory-usage/

[55] Setup 2 – More tweaks to save precious RAM (2012, Octubre) [Online] https://projects.drogon.net/raspberry-pi/initial-setup2/

[56] Osciloscopio Sds Dl Siglent100 Mhz 2 Canales Incluye Puntas (2015) [Online] articulo.mercadolibre.com.ec/MEC-407069872-osciloscopio-sds-dl-siglent100-mhz-2-canales-incluye-puntas-_JM

[57] SDS1102DL 100MHz (2013) [Online] www.siglent.eu/oscilloscopes/sds-1000dl-series/sds1102dl-100mhz.html

[58] SHS810 100MHz (2013) [Online] www.siglent.eu/shs810-100mhz.html

[59] Osciloscopio 40 Mhz Doble Canal (2015) [Online] articulo.mercadolibre.com.ec/MEC-407024705-osciloscopio-40-mhz-doble-canal-_JM

[60] Dual Trace Oscilloscope, 40 MHz (2015) [Online] https://www.labvolt.com/solutions/9_telecommunications/40-797-20_dual_trace_oscilloscope

[61] Osciloscopio Digital Ds201 Mini 2.8 Armcortex Arduino (2015) [Online] articulo.mercadolibre.com.ec/MEC-406794582-osciloscopio-digital-ds201-mini-28-armcortex-arduino-_JM

[62] Partidas Presupuestarias – Presupuesto 2011 (2011) [Online] http://uta.edu.ec/v2.0/pdf/uta/presupuesto/presupuestogastos2011.pdf

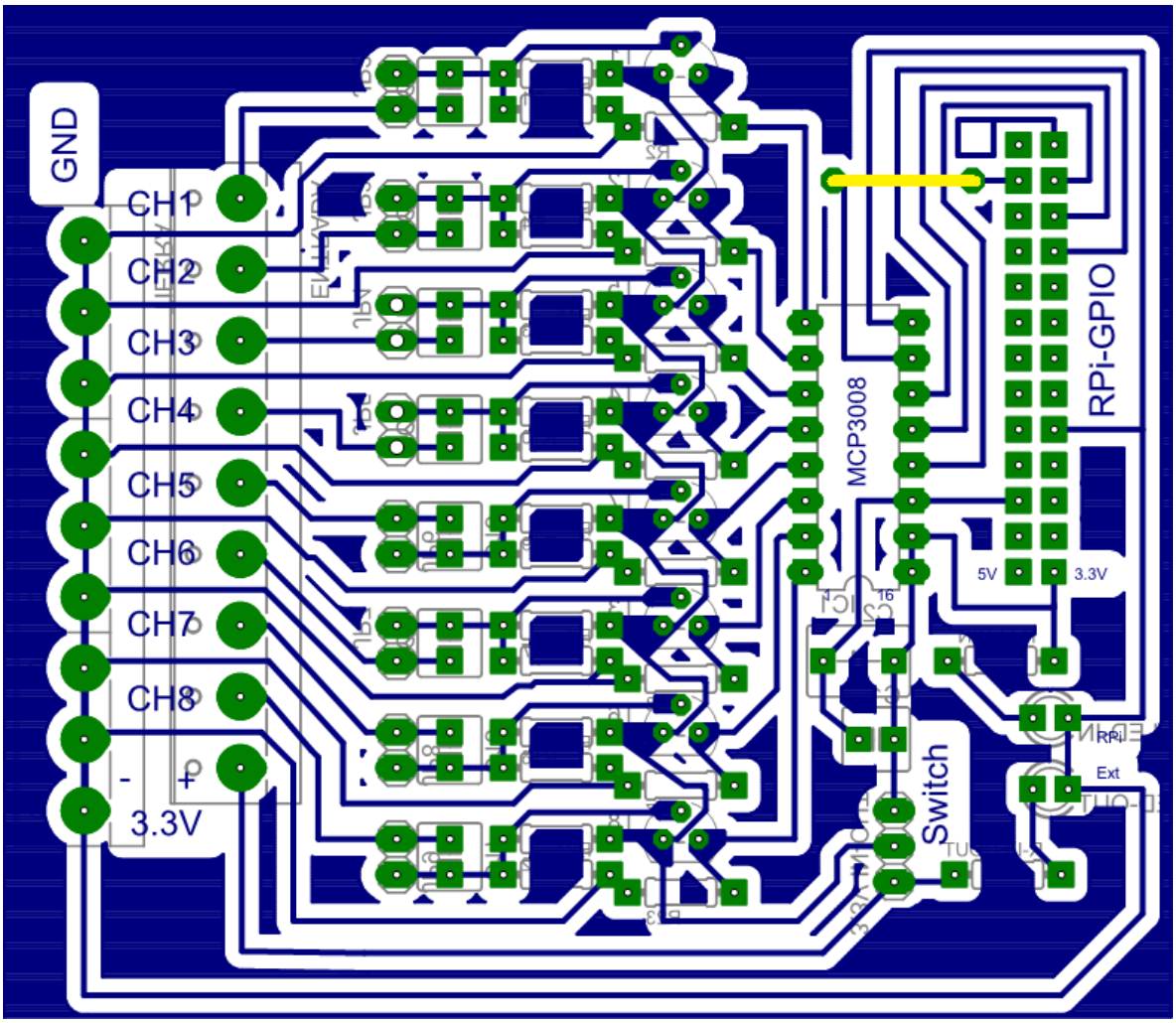
[63] Partidas Presupuestarias – Presupuesto 2013 (2013) [Online] http://www.uta.edu.ec/v2.0/phocadownload/lotaip2/g/presupuestoinicial2013.pdf

[64] Hewitt Paul G. Física Conceptual. México. Enrique Quintanar Duarte. Pearson Education México, 2004.

ANEXOS

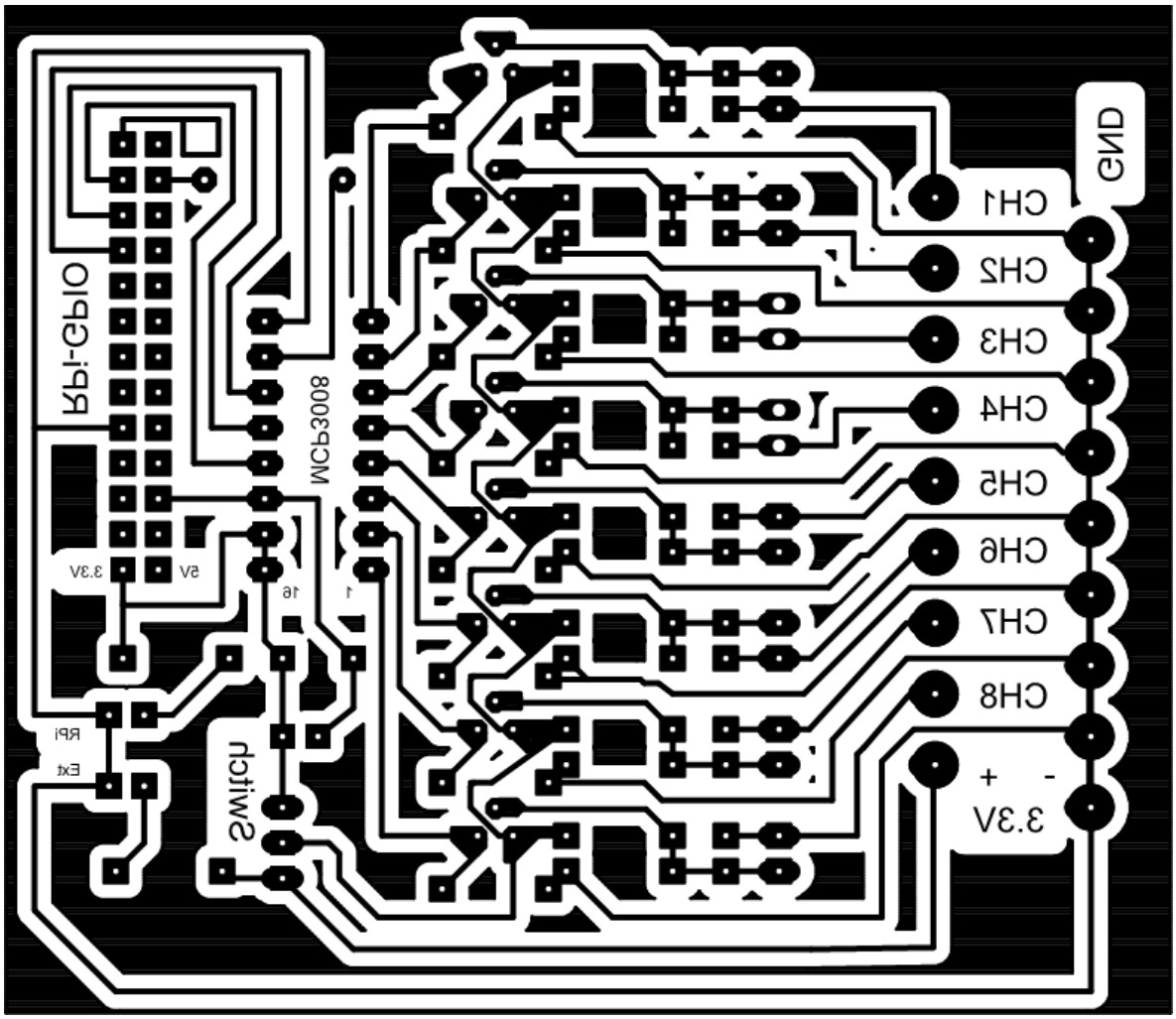
ANEXO 1

Diseño de la Placa Impresa con elementos sobrepuestos



ANEXO 2

Diseño en espejo para transferencia térmica



ANEXO 3

Codificación del programa

	Código	
1	#!/usr/bin/env python	Convierte el archivo en ejecutable
2	import initExample, os, serial	Declaración de librerías
3	import pyqtgraph as pg	
4	from pyqtgraph.Qt import QtCore, QtGui	
5	import numpy as np	
6	from pyqtgraph.dockarea import *	
7	import spidev	
8	spi = spidev.SpiDev()	Abre puerto SPI a través de SPIDEV
9	spi.open(0,0)	
10	aplicacion = QtGui.QApplication([])	Declarar aplicación
11	ventana = QtGui.QMainWindow()	Ventana principal
12	ventana.setWindowTitle('Analizador de Espectro')	Titulo
13	distribucion=DockArea()	Agregar Dock
14	ventana.setCentralWidget(distribucion)	Establecer como central
15	ventana.resize(1200,700)	Tamaño
16	bloque1=Dock("Area de Trazado",size=(1200,1000), closable=False)	Configuración de Áreas de trabajo
17	bloque2=Dock("1 HZ", size=(10,600), closable=True)	
18	bloque3=Dock("Controles", size=(10,10), closable=True)	
19	bloque4=Dock("10 KHz", size=(10,10), closable=True)	
20	bloque5=Dock("Amplitud" ,size=(10,10), closable=True)	

21	bloque7=Dock("Apagar o Reiniciar el Sistema", size=(10,10), closable=True)	
22	distribucion.addDock(bloque7,'top')	Posicionamiento de las áreas
23	distribucion.addDock(bloque1,'above',bloque7)	
24	distribucion.addDock(bloque5,'right',bloque7)	
25	distribucion.addDock(bloque3,'above',bloque5)	
26	distribucion.addDock(bloque2,'bottom',bloque5)	
27	distribucion.addDock(bloque4,'above',bloque2)	
28	pg.setConfigOption('background','aea5e7')	Color de fondo
29	pg.setConfigOption('foreground','r')	Color de cuadricula
30	trazo_s=pg.GraphicsWindow()	Ventana gráfica
31	despla=trazo_s.addPlot()	Agregar trazador a ventana gráfica
32	despla.showGrid(True,True)	Cuadricula
33	despla.showAxis('left',False)	Ocultar eje izquierdo
34	despla.setLabel('right'," Amplitud ", units='_Voltios_')	Etiqueta
35	despla_t0=despla.plot(pen=(255,0,0))	Color del trazo
36	despla_t1=despla.plot(pen=(255,255,255))	
37	despla_t2=despla.plot(pen=(255,255,0))	
38	despla_t3=despla.plot(pen=(200,100,200))	
39	despla_t4=despla.plot(pen=(0,255,0))	
40	despla_t5=despla.plot(pen=(100,100,100))	
41	despla_t6=despla.plot(pen=(0,255,255))	
42	despla_t7=despla.plot(pen=(0,0,255))	
43	datoS0=[]	Declarar arreglo
44	datoS01=[]	
45	datoS1=[]	

46	datoS2=[]	
47	datoS3=[]	
48	datoS4=[]	
49	datoS5=[]	
50	datoS6=[]	
51	datoS7=[]	
52	volta=3.30	Valores predefinidos
53	frec=1	
54	frec2=1	
55	bloque1.addWidget(trazo_s)	Agregar la ventana gráfica al bloque 1
56	def Actual_S():	Función de trazado
57	global frec2, frec, datoS0, datoS01,datoS1, datoS2, datoS3, datoS4, datoS5, datoS6, datoS7, despla_t0,despla_t1, despla_t2, despla_t3, despla_t4, despla_t5, despla_t6, despla_t7, volta	
58	if A_check0.isChecked():	Si la casilla se activa
59	resultado=open("canal_1.txt")	abrir archivo txt
60	for archivo in resultado:	Ciclo de lectura
61	datoS01.append(volta*float(archivo)/1024)	Agregar cada valor con los parámetros definidos
62	resultado.close()	Cerrar archivo
63	datoS0.append(datoS01)	Agregar el arreglo leído al arreglo principal
64	datoS01=[]	Limpiar arreglo
65	despla_t0.setData(np.hstack(datoS0))	Agregar un arreglo al trazador
66	if A_check1.isChecked():	
67	resultado=open("canal_2.txt")	
68	for archivo in resultado:	
69	datoS01.append(volta*float(archivo)/1024)	
70	resultado.close()	

```
71          datoS1.append(datoS01)
72          datoS01=[]
73          despla_t1.setData(np.hstack(datoS1))
74      if A_check2.isChecked():
75          resultado=open("canal_3.txt")
76          for archivo in resultado:
77              datoS01.append(volta*float(archivo)/1024)
78          resultado.close()
79          datoS2.append(datoS01)
80          datoS01=[]
81          despla_t2.setData(np.hstack(datoS2))
82      if A_check3.isChecked():
83          resultado=open("canal_4.txt")
84          for archivo in resultado:
85              datoS01.append(volta*float(archivo)/1024)
86          resultado.close()
87          datoS3.append(datoS01)
88          datoS01=[]
89          despla_t3.setData(np.hstack(datoS3))
90      if A_check4.isChecked():
91          resultado=open("canal_5.txt")
92          for archivo in resultado:
93              datoS01.append(volta*float(archivo)/1024)
94          resultado.close()
95          datoS4.append(datoS01)
96          datoS01=[]
97          despla_t4.setData(np.hstack(datoS4))
98      if A_check5.isChecked():
```

99	resultado=open("canal_6.txt")	
100	for archivo in resultado:	
101	datoS01.append(volta*float(archivo)/1024)	
102	resultado.close()	
103	datoS5.append(datoS01)	
104	datoS01=[]	
105	despla_t5.setData(np.hstack(datoS5))	
106	if A_check6.isChecked():	
107	resultado=open("canal_7.txt")	
108	for archivo in resultado:	
109	datoS01.append(volta*float(archivo)/1024)	
110	resultado.close()	
111	datoS6.append(datoS01)	
112	datoS01=[]	
113	despla_t6.setData(np.hstack(datoS6))	
114	if A_check7.isChecked():	
115	resultado=open("canal_8.txt")	
116	for archivo in resultado:	
117	datoS01.append(volta*float(archivo)/1024)	
118	resultado.close()	
119	datoS7.append(datoS01)	
120	datoS01=[]	
121	despla_t7.setData(np.hstack(datoS7))	
122	if len(datoS0)> frec:	Determinar límite del arreglo
123	datoS0.pop(0)	Desplazar arreglo
124	if len(datoS1) >frec:	
125	datoS1.pop(0)	
126	if len(datoS2) >frec:	

127	datoS2.pop(0)	
128	if len(datoS3) >frec:	
129	datoS3.pop(0)	
130	if len(datoS4) >frec:	
131	datoS4.pop(0)	
132	if len(datoS5) >frec:	
133	datoS5.pop(0)	
134	if len(datoS6) >frec:	
135	datoS6.pop(0)	
136	if len(datoS7) >frec:	
137	datoS7.pop(0)	
138	if frec2 != frec:	Cuando se cambie la configuración
139	datoS0=[]	limpiar arreglo
140	datoS1=[]	
141	datoS2=[]	
142	datoS3=[]	
143	datoS4=[]	
144	datoS5=[]	
145	datoS6=[]	
146	datoS7=[]	
147	frec2=frec	
148	despla.setTitle("(100-10KHz)[%0.2f V]"%(volta))	Título
149	def aperiodico():	Función del segundo trazador
150	global datos,frec2,frec, datoAr, datoS0,datoS1, datoS2, datoS3, datoS4, datoS5, datoS6, datoS7, despla_t0,despla_t1, despla_t2, despla_t3, despla_t4, despla_t5, despla_t6, despla_t7, volta	

151	if A2_check0.isChecked():	Si la casilla se activa
152	datoS0.append(leer(0))	Agregar dato al arreglo
153	despla_t0.setData(np.hstack(datoS0))	Agregar arreglo al trazador
154	if A2_check1.isChecked():	
155	datoS1.append(leer(1))	
156	despla_t1.setData(np.hstack(datoS1))	
157	if A2_check2.isChecked():	
158	datoS2.append(leer(2))	
159	despla_t2.setData(np.hstack(datoS2))	
160	if A2_check3.isChecked():	
161	datoS3.append(leer(3))	
162	despla_t3.setData(np.hstack(datoS3))	
163	if A2_check4.isChecked():	
164	datoS4.append(leer(4))	
165	despla_t4.setData(np.hstack(datoS4))	
166	if A2_check5.isChecked():	
167	datoS5.append(leer(5))	
168	despla_t5.setData(np.hstack(datoS5))	
169	if A2_check6.isChecked():	
170	datoS6.append(leer(6))	
171	despla_t6.setData(np.hstack(datoS6))	
172	if A2_check7.isChecked():	
173	datoS7.append(leer(7))	
174	despla_t7.setData(np.hstack(datoS7))	
175	despla.setTitle("Osciloscopio [%0.2f V]"%(volta))	Título del trazador
176	if len(datoS0)>= frec*100:	Determinar límite del arreglo

177	datoS0.pop(0)	Desplazar arreglo
178	if len(datoS1) >=frec*100:	
179	datoS1.pop(0)	
180	if len(datoS2) >=frec*100:	
181	datoS2.pop(0)	
182	if len(datoS3) >=frec*100:	
183	datoS3.pop(0)	
184	if len(datoS4) >=frec*100:	
185	datoS4.pop(0)	
186	if len(datoS5) >=frec*100:	
187	datoS5.pop(0)	
188	if len(datoS6) >=frec*100:	
189	datoS6.pop(0)	
190	if len(datoS7) >=frec*100:	
191	datoS7.pop(0)	
192	if frec2 != frec:	Cuando se cambie la configuración
193	datoS0=[]	Limpiar arreglo
194	datoS1=[]	
195	datoS2=[]	
196	datoS3=[]	
197	datoS4=[]	
198	datoS5=[]	
199	datoS6=[]	
200	datoS7=[]	
201	frec2=frec	Igualar referencias
202	def limpiar():	Función de Limpieza de trazador
203	datoS0=[]	Limpiar arreglo
204	datoS1=[]	

205	datoS2=[]	
206	datoS3=[]	
207	datoS4=[]	
208	datoS5=[]	
209	datoS6=[]	
210	datoS7=[]	
211	IterS=0	
212	datoS0.append(0)	Agregar valor CERO
213	datoS1.append(0)	
214	datoS2.append(0)	
215	datoS3.append(0)	
216	datoS4.append(0)	
217	datoS5.append(0)	
218	datoS6.append(0)	
219	datoS7.append(0)	
220	despla_t0.setData(np.hstack(datoS0))	Agregar al trazador
221	despla_t1.setData(np.hstack(datoS1))	
222	despla_t2.setData(np.hstack(datoS2))	
223	despla_t3.setData(np.hstack(datoS3))	
224	despla_t4.setData(np.hstack(datoS4))	
225	despla_t5.setData(np.hstack(datoS5))	
226	despla_t6.setData(np.hstack(datoS6))	
227	despla_t7.setData(np.hstack(datoS7))	
228	numer_ctrl=QtGui.QWidget()	Agregar Widget
229	N_grid_ctrl=QtGui.QGridLayout()	Cuadricula
230	numer_ctrl.setLayout(N_grid_ctrl)	Agregar cuadricula
231	A_Combo1_ctrl=QtGui.QComboBox()	Agregar "Combo Box"
232	A_Combo2_ctrl=QtGui.QComboBox()	

233	ctrl_eti_sp1=QtGui.QLabel("Muestras")	etiqueta
234	ctrl_eti_sp2=QtGui.QLabel("Amplitud")	
235	items_ctrl1=['1','2','3','4','5','6','7','8','9','10']	Arreglo de cadenas
236	items_ctrl2=['3.3','5','9','12','15','20','30','50','100','120']	
237	A_Combo1_ctrl.addItems(items_ctrl1)	Agregar arreglos al "Combo Box"
238	A_Combo2_ctrl.addItems(items_ctrl2)	
239	N_grid_ctrl.addWidget(ctrl_eti_sp1,1,0)	Agregar al Widget
240	N_grid_ctrl.addWidget(A_Combo1_ctrl,2,0)	
241	N_grid_ctrl.addWidget(ctrl_eti_sp2,3,0)	
242	N_grid_ctrl.addWidget(A_Combo2_ctrl,4,0)	
243	def f1(ind):	Función para determinar cambios en la configuración del "Combo Box"
244	global frec	
245	frec=int(A_Combo1_ctrl.currentText())	
246	def f2(ind):	Función para determinar cambios en la configuración del "Combo Box"
247	global volta	
248	volta=int(A_Combo2_ctrl.currentText())	
249	A_Combo1_ctrl.currentIndexChanged.connect(f1)	Conectar con función al detectar cambios
250	A_Combo2_ctrl.currentIndexChanged.connect(f2)	
251	bloque3.addWidget(numer_ctrl)	Agregar Widget al bloque 3
252	salida=QtGui.QWidget()	Widget
253	salida_grid=QtGui.QGridLayout()	Cuadricula
254	salida.setLayout(salida_grid)	Agregar cuadricula
255	salida_btonCA=QtGui.QPushButton('Apagar el Sistema')	Declarar botón
256	salida_btonCR=QtGui.QPushButton('Resetear el Sistema')	
257	salida_grid.addWidget(salida_btonCA)	Agregar Botón

258	salida_grid.addWidget(salida_btonCR)	
259	bloque7.addWidget(salida)	Agregar Widget al bloque 7
260	def Apagar():	Función Apagar
261	os.system('sudo poweroff')	Llamada del sistema para apagar
262	def Reset():	Función Resetear
263	os.system('sudo reboot')	Llamada del sistema para resetear
264	salida_btonCA.clicked.connect(Apagar)	Conectar con función al presionar botón
265	salida_btonCR.clicked.connect(Reset)	
266	activador2=QtGui.QWidget()	Widget
267	A2_grid=QtGui.QGridLayout()	Cuadricula
268	activador2.setLayout(A2_grid)	Agregar cuadricula
269	A2_btonCl=QtGui.QPushButton('limpiar')	Botón
270	A2_checkA=QtGui.QCheckBox('Activar')	Casilla
271	A2_checkA.setChecked(False)	Configurar desactivado
272	A2_check0=QtGui.QCheckBox('ADC0')	
273	A2_check0.setChecked(False)	
274	A2_check1=QtGui.QCheckBox('ADC1')	
275	A2_check1.setChecked(False)	
276	A2_check2=QtGui.QCheckBox('ADC2')	
277	A2_check2.setChecked(False)	
278	A2_check3=QtGui.QCheckBox('ADC3')	
279	A2_check3.setChecked(False)	
280	A2_check4=QtGui.QCheckBox('ADC4')	
281	A2_check4.setChecked(False)	
282	A2_check5=QtGui.QCheckBox('ADC5')	
283	A2_check5.setChecked(False)	
284	A2_check6=QtGui.QCheckBox('ADC6')	
285	A2_check6.setChecked(False)	

286	A2_check7=QtGui.QCheckBox('ADC7')	
287	A2_check7.setChecked(False)	
288	A2_grid.addWidget(A2_btonCl,0,0)	Posicionar elementos
289	A2_grid.addWidget(A2_checkA,1,0)	
290	A2_grid.addWidget(A2_check0,2,0)	
291	A2_grid.addWidget(A2_check1,3,0)	
292	A2_grid.addWidget(A2_check2,4,0)	
293	A2_grid.addWidget(A2_check3,5,0)	
294	A2_grid.addWidget(A2_check4,6,0)	
295	A2_grid.addWidget(A2_check5,7,0)	
296	A2_grid.addWidget(A2_check6,8,0)	
297	A2_grid.addWidget(A2_check7,9,0)	
298	A2_btonCl.clicked.connect(limpiar)	Conectar con función limpiar al activar
299	bloque2.addWidget(activador2)	Agregar Widget al Bloque 2
300	activador=QtGui.QWidget()	Widget
301	A_grid=QtGui.QGridLayout()	Cuadricula
302	activador.setLayout(A_grid)	Agregar cuadricula
303	A_btonCl=QtGui.QPushButton('limpiar')	Botón
304	A_checkA=QtGui.QCheckBox('Iniciar')	Casilla
305	A_checkA.setChecked(False)	Configurar desactivado
306	A_check0=QtGui.QCheckBox('C - 1 rojo')	
307	A_check0.setChecked(False)	
308	A_check1=QtGui.QCheckBox('C - 2 blanco')	
309	A_check1.setChecked(False)	
310	A_check2=QtGui.QCheckBox('C - 3 amarillo')	
311	A_check2.setChecked(False)	
312	A_check3=QtGui.QCheckBox('C - 4 rosa')	
313	A_check3.setChecked(False)	

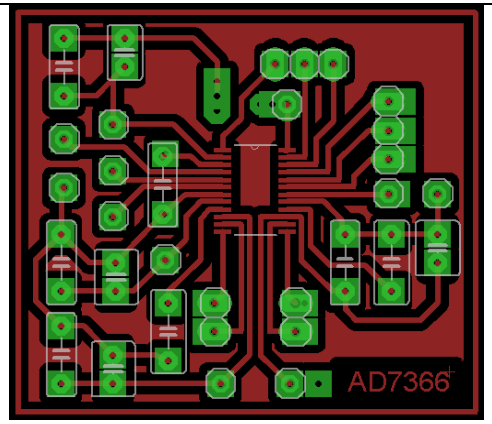
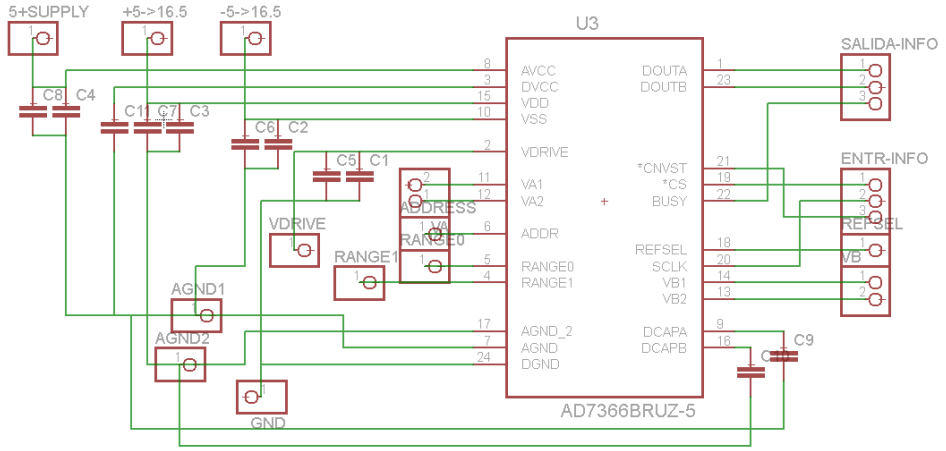
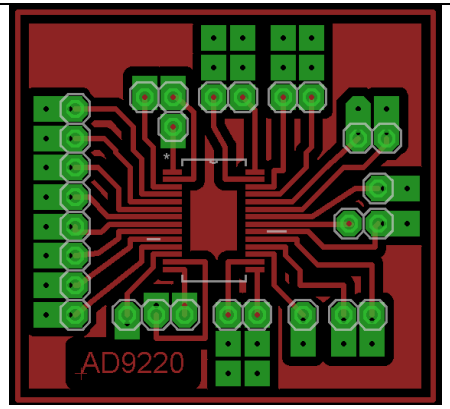
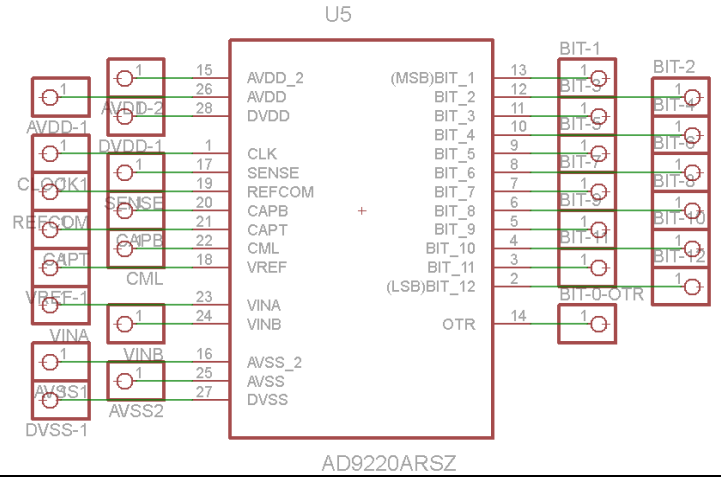
314	A_check4=QtGui.QCheckBox('C - 5 verde')	
315	A_check4.setChecked(False)	
316	A_check5=QtGui.QCheckBox('C - 6 negro')	
317	A_check5.setChecked(False)	
318	A_check6=QtGui.QCheckBox(' - 7 celeste')	
319	A_check6.setChecked(False)	
320	A_check7=QtGui.QCheckBox('C - 8 azul')	
321	A_check7.setChecked(False)	
322	A_grid.addWidget(A_btonCl,0,0)	Posicionar elementos
323	A_grid.addWidget(A_checkA,1,0)	
324	A_grid.addWidget(A_check0,2,0)	
325	A_grid.addWidget(A_check1,3,0)	
326	A_grid.addWidget(A_check2,4,0)	
327	A_grid.addWidget(A_check3,5,0)	
328	A_grid.addWidget(A_check4,6,0)	
329	A_grid.addWidget(A_check5,7,0)	
330	A_grid.addWidget(A_check6,8,0)	
331	A_grid.addWidget(A_check7,9,0)	
332	A_btonCl.clicked.connect(limpiar)	Conectar con función limpiar al activar
333	bloque4.addWidget(activador)	Agregar Widget al Bloque 4
334	numer=QtGui.QWidget()	Widget
335	N_grid=QtGui.QGridLayout()	Cuadricula
336	numer.setLayout(N_grid)	Agregar cuadricula
337	A_Combo1=QtGui.QComboBox()	Combo Box
338	A_Combo2=QtGui.QComboBox()	
339	A_Combo3=QtGui.QComboBox()	
340	A_Combo4=QtGui.QComboBox()	
341	A_Combo5=QtGui.QComboBox()	

342	items=['0','1','2','3','4','5','6','7','8','9']	Arreglo de cadenas
343	items2=['0','1']	
344	A_Combo1.addItems(items2)	Agregar Combo Box
345	A_Combo2.addItems(items)	
346	A_Combo3.addItems(items)	
347	A_Combo4.addItems(items)	
348	A_Combo5.addItems(items)	
349	N_grid.addWidget(A_Combo1,1,0)	Posicionar elementos
350	N_grid.addWidget(A_Combo2,2,0)	
351	N_grid.addWidget(A_Combo3,3,0)	
352	N_grid.addWidget(A_Combo4,4,0)	
353	N_grid.addWidget(A_Combo5,5,0)	
354	def f1(ind):	Función para determinar cambios en la configuración del "Combo Box"
355	global Frecuencia,f1,f2,f3,f4,f5,volta	
356	f1=int(A_Combo1.currentText())	Obtener el valor del combo box
357	f2=int(A_Combo2.currentText())	
358	f3=int(A_Combo3.currentText())	
359	f4=int(A_Combo4.currentText())	
360	f5=int(A_Combo5.currentText())	
361	volta=f1*100+f2*10+f3+f4*0.1+f5*0.01	Calcular voltaje
362	Frecuencia=f1	
363	A_Combo1.currentIndexChanged.connect(f1)	Conectar con función F1 al detectar cambios
364	A_Combo2.currentIndexChanged.connect(f1)	
365	A_Combo3.currentIndexChanged.connect(f1)	
366	A_Combo4.currentIndexChanged.connect(f1)	
367	A_Combo5.currentIndexChanged.connect(f1)	
368	bloque5.addWidget(numer)	Agregar numer al bloque 5

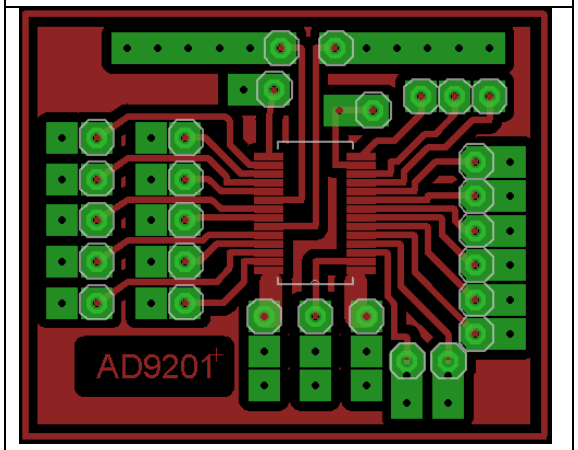
369	def leer(CH):	Función leer canal
370	global volta	
371	ADC=spi.xfer2([1,(8+CH)<<4,0])	Transferencia de cadenas
372	datoCH=((ADC[1]&3)<<8)+ADC[2]	Algoritmo para calcular el valor capturado
373	datos=float(datoCH*volta/1024)	Ajuste con parámetros establecidos
374	return datos	Devolver datos
375	def delayer():	FUNCION PRINCIPAL
376	if A_checkA.isChecked():	Si se activa la casilla principal
377	os.system('sudo /home/pi/Analizador/examples/./Salida')	Ejecutar en modo SUDO el ejecutable en C
378	A2_checkA.setChecked(False)	Desactivar casilla del segundo trazado
379	Actual_S()	conectar con función
380	if A2_checkA.isChecked():	Si se activa la casilla principal
381	aperiodico()	conectar con función
382	timer = QtCore.QTimer()	Timer de QT
383	timer.timeout.connect(delayer)	Conectar con función
384	timer.start(50)	Retardo entre Ciclos en milisegundos
385	ventana.showMaximized()	Configuración de ventana
386	if __name__ == '__main__':	Configuración de la aplicación
387	import sys	
388	if (sys.flags.interactive != 1) or not hasattr(QtCore, 'PYQT_VERSION'):	
389	QtGui.QApplication.instance().exec_()	

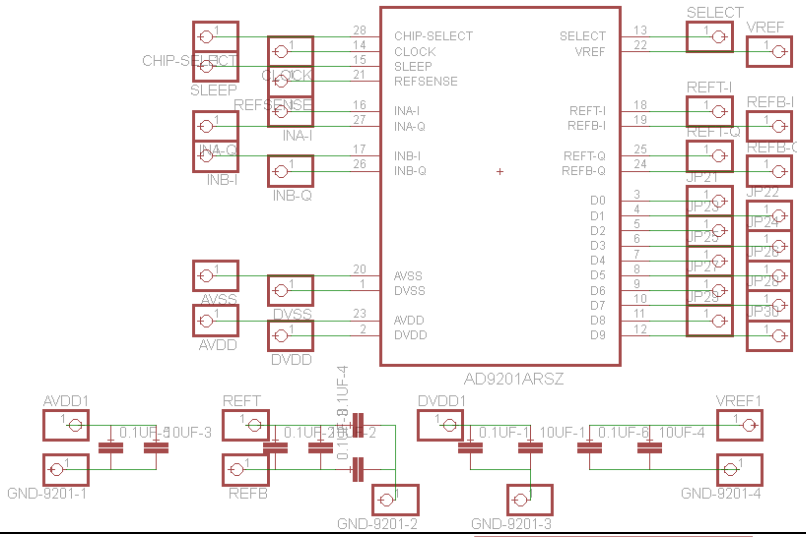
Anexo 4

Diseño de circuitos impresos para acoplar los Conversores Análogo digital que se utilizaron para pruebas de conexión y a través de esto se determinó las limitaciones de conectividad del puerto GPIO.

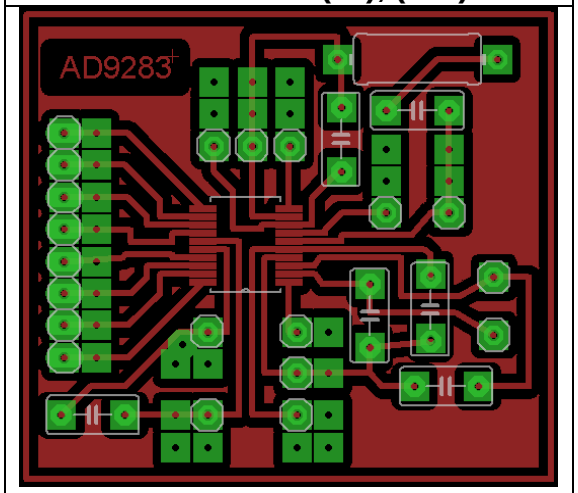
Conversor Análogo Digital	Diagrama de conexión correspondiente
AD7366BRUZ-5 	
AD9220ARSZ 	

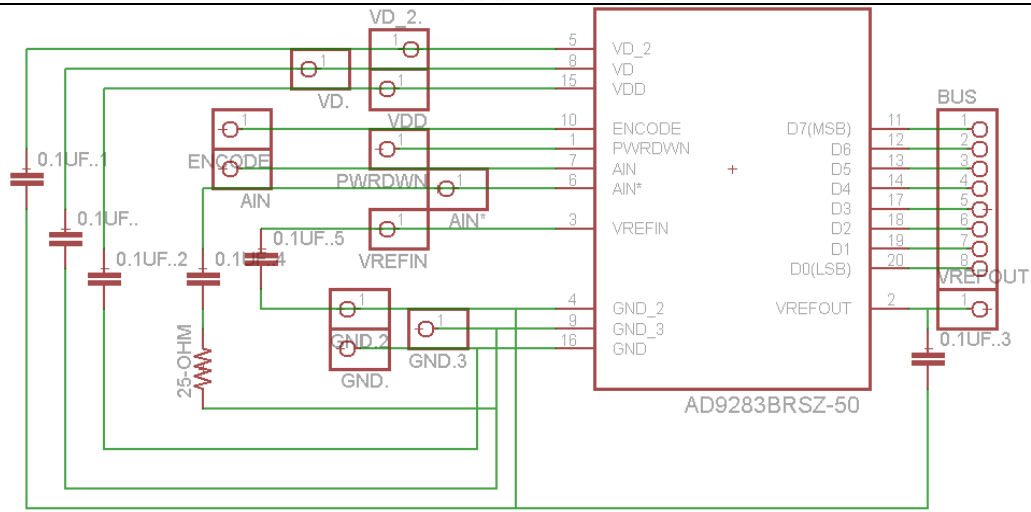
AD9201ARSZ





AD9283BRSZ (50), (100)





AD5541ABRMZ