UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Diseño de un osciloscopio accesible para estudiantes

TESIS

Que para obtener el título de

Ingeniero Mecánico

P R E S E N T A

Rodrigo Aragón Mureddu

DIRECTOR DE TESIS

M.I. Ulises Martín Peñuelas Rivas

Ciudad Universitaria, Cd. Mx., 2024

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado DISEÑO DE UN OSCILOSCOPIO ACCESIBLE PARA ESTUDIANTES que presenté para obtener el titulo de INGENIERO MECÁNICO es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

RODRIGO ARAGON MUREDDU
Número de cuenta: 314590835

AGRADECIMIENTOS

Le quiero agradecer a mi madre y a mi padre por darme sustento durante toda mi formación profesional, la cual ha encumbrado en este pequeño pico llamado ”final de carrera”.

Le agradezco al profesor Yukihiro Minami, por permitirme tantas horas de conversación y asistencia sin las cuales no habría sido posible el proceso creativo de este proyecto.

Le agradezco al profesor Ulises Peñuelas Rivas, sin el cual la presentación de este documento no hubiera sido posible, así como tantos comentarios para el mejoramiento y avance de este proyecto.

Le agradezco a la profesora Yoloxóchil Jiménez, por obligarme a ver que la electricidad y la electrónica eran la rama de la ingeniería a la que debía dedicar mi tiempo.

Le agradezco a tantos amigos que durante toda mi formación han estado abiertos a escucharme e intercambiar ideas. En concreto, Mario Garrido y Daniel Vela, amigos entusiastas de la ingeniería con los cuales tuve intercambios fructíferos para el desarrollo de este proyecto en concreto.

Y le agradezco a Gabriela Zamudio por haber sido apoyo incondicional durante buena parte del desarrollo de este proyecto así como gran ayuda en las últimas etapas de revisión del documento aquí presente.

Este trabajo de tesis fue realizado gracias al apoyo del programa UNAM-DGAPA-PAPIIT IT102121 ”Diseño de dispositivos mecatrónicos para apoyo en emergencias”.

DEDICATORIA

Este proyecto lo dedico a la Facultad de Ingeniería, lugar cultural y académico donde he aprendido en igual magnitud estando tanto dentro como fuera de aulas. Que sea ésta una de muchas contribuciones más.

TABLA DE CONTENIDO

1 Introduccion **11**
- 1.1 Importancia de los laboratorios . . . 11
- 1.2 Laboratorios de electrónica . . . 12
- 1.3 Objetivo . . . 12
- 1.4 Resumen . . . 12
  - 1.4.1 Antecedentes . . . 12
  - 1.4.2 Diseño . . . 13
  - 1.4.3 Conclusiones y trabajo futuro . . . 13
  - 1.4.4 Apéndices . . . 13

2 Antecedentes **15**
- 2.1 Componentes de un osciloscopio . . . 15
- 2.2 Clasificación de osciloscopios . . . 16
- 2.3 Resumen . . . 17

3 Diseño **19**
- 3.1 Análisis del problema . . . 19
- 3.2 Primera aproximación abstracta . . . 21
  - 3.2.1 Circuito de acondicionamiento . . . 21
  - 3.2.2 ADC lector . . . 21
  - 3.2.3 Programa intérprete . . . 21
  - 3.2.4 Programa graficador . . . 22
- 3.3 Búsqueda de soluciones . . . 22
  - 3.3.1 Circuito de acondicionamiento . . . 22
  - 3.3.2 Tarjeta de desarollo . . . 23
  - 3.3.3 Lenguaje de programación . . . 24
- 3.4 Diseño final . . . 25
  - 3.4.1 Esquema general . . . 25
  - 3.4.2 Dispositivo de medición . . . 25
  - 3.4.3 Dispositivo de procesamiento . . . 29
  - 3.4.4 Mensajero UDP, buzón UDP . . . 31
  - 3.4.5 Graficación . . . 32
- 3.5 Pruebas . . . 33
- 3.6 Resultados . . . 34
- 3.7 Resumen . . . 37

4 Conclusiones y trabajo futuro **39**
- 4.1 Conclusiones . . . 39
- 4.2 Trabajo futuro . . . 39

A Apéndices **41**
A.1 Con respecto al ruido . . . 41
A.2 Material adicional . . . 41
A.2.1 Material didáctico adicional . . . 41
A.2.2 Manual de usuario . . . 42
A.2.3 Usuario avanzado y proceso de diseño . . . 42
A.3 Demostraciones de comportamiento y funcionamiento . . . 42
A.3.1 Demostración de transformación de circuito de acondicionamiento . . . 42
A.3.2 Cálculo del valor de los resistores . . . 47
A.3.3 Diseño de la comunicación serial . . . 47
A.3.4 Código implementado en la tarjeta de desarrollo (ESP32) . . . 50
A.3.5 Decodificador del cifrado de bytes . . . 51
A.3.6 Mensajero entre aplicaciones UDP o RAM . . . 54
A.3.7 Graficador . . . 54

B Ejemplo de implementación **59**
B.1 Determinación del resistor RN . . . 59
B.2 Código Mensajero-Buzón UDP . . . 60
B.3 Código del graficador . . . 62

Referencias **69**

RESUMEN

En este documento se detallará el proceso de diseño y uso de un osciloscopio, una herramienta dedicada a la medición de señales eléctrico-electrónicas de moderada o alta velocidad. El proceso de diseño dará detalle a todas las decisiones tomadas durante el proceso mientras que, al explicar el uso, se dará especial enfoque al uso de la última implementación realizada al momento de escritura de este documento.

Este osciloscopio se diseñó con la accesibilidad, economía y facilidad de uso en mente, así, manteniendo siempre como foco de atención a los estudiantes de la Facultad de Ingeniería y, por extensión, a los estudiantes de ingeniería de otras escuelas en el ámbito nacional.

Si bien el diseño del osciloscopio podrá seguir cambiando con el paso del tiempo, así como con la implementación de correcciones y programación posterior, el esquema general de su funcionamiento no cambiará de manera sustancial. Éste quedará como un testimonio de su funcionamiento a la fecha, abril 2024.

1
INTRODUCCION

1.1 Importancia de los laboratorios

En la enseñanza de la ingeniería es común tener dos tipos de clases, las teóricas y las prácticas. Las prácticas, por los requerimientos tan específicos, y a veces riesgos que suelen tener los experimentos a realizar, se tiene que realizar en un laboratorio donde se provean los materiales necesarios para desarrollarlas de forma efectiva y didáctica.

Este tipo de educación ha probado ser la única alternativa realmente viable para complementar la educación abstracta que se imparte en el aula. Así, los estudiantes pueden observar con sus propios ojos los fenómenos descritos en el salón, además de aprender a usar las herramientas, que muchas veces son versiones simplificadas de las que se usan hoy día en la industria, les dan una segunda oportunidad de anclar ese conocimiento adquirido en el salón por medio de una experiencia práctica real, no ideal ni abstracta.

Durante la pandemia de COVID-19, se volvió patente la importancia de estos laboratorios, desafortunadamente, por su ausencia. Esto llevó a muchos profesores de las diferentes facultades a pensar en diferentes alternativas didácticas para suplir los entonces inaccesibles laboratorios. Entre las alternativas más recurridas estaba el uso de videos, fotografías y simuladores para suplir el conocimiento adquirido en el aula. El uso de estas tecnologías demostró ser insuficiente y, a ojos de los estudiantes, insoportablemente tedioso.

Otras de las propuestas atacaron esta circunstancia con un ángulo distinto. En primera instancia, se consideró que era imperativa la interacción del estudiante con el experimento. Segundo, se consideró que los estudiantes debían idealmente desarrollar su propio ecosistema de trabajo individual en casa. Esto llevó a varios profesores a diseñar experimentos que no dependieran del herramental presente en los laboratorios, sino el que fuera fácil y económico de conseguir para los estudiantes. Así surgió el concepto del ”laboratorio en casa”.

Por último, como complemento a este laboratorio en casa, también se decidió que las herramientas digitales podrían actuar como un canal de comunicación entre el estudiante y el laboratorio, en caso de requerirse material inaccesible para los estudiantes. Así se proponía que los estudiantes, por medio de una computadora o un teléfono celular, pudieran interactuar con un experimento, haciendo uso de cámaras y sensores conectados a una plataforma digital con la cual el estudiante pudiera interactuar. Así surgió el concepto de ”laboratorio remoto” o ”laboratorio desde casa”.

Tras el retorno a las clases presenciales, estos últimos dos tipos de laboratorios accesibles

para los estudiantes desde su hogar, parecieron ser herramientas suficientemente valiosas por mérito propio, aún acceso a los laboratorios de la Facultad. Por medio de estos laboratorios, se podrían seguir realizando prácticas más sencillas que las realizadas en las instalaciones físicas y también abrir un horizonte de pequeñas tareas-prácticas para realizar desde casa, ofreciendo así más herramientas didácticas a profesores y estudiantes.

1.2 Laboratorios de electrónica

En el ámbito de la docencia de la electrónica, existen el material de modelado y el herramental de precisión. Al momento de diseñar prácticas y demás actividades en laboratorios de enseñanza, generalmente se espera que los estudiantes proporcionen el material de modelado, tales como resistores, condensadores, inductores y tarjetas de desarrollo, mientras que la institución pone los instrumentos de precisión, entre ellos osciloscopios y generadores de señales, los cuales generalmente resultan inaccesibles por su precio.

Durante la contingencia de COVID-19. Durante las clases en línea, muchas de las prácticas de laboratorio se tuvieron que llevar a cabo en casa, algunos por medio de simulación, otros por medio del armado, pruebas y verificación del funcionamiento de circuitos electrónicos sencillos. El no tener acceso a un osciloscopio durante las clases de la pandemia, así como la falta de realización de prácticas presenciales en los laboratorios, fueron las principales causas circunstanciales de este proyecto. El osciloscopio es una herramienta bastante útil para la enseñanza de electrónica por la posibilidad de permitir la visualización de señales eléctricas, a diferencia de mostrar su valor en un momento dado como se haría con un multímetro.

1.3 Objetivo

Mencionada la importancia de los laboratorios, este proyecto intenta contribuir como -una herramienta más- de los laboratorios que los estudiantes podrán construir a partir de insumos sencillos y accesibles, para realizar prácticas y trabajos personales desde casa.

Si bien la principal audiencia para esta tesis supone ser el estudiantado de la Facultad de Ingeniería, ya que las herramientas no están limitadas únicamente a estos experimentos, se considera que esta propuesta también podría ser útil para la impartición de talleres o uso particular dentro del mundo de los entusiastas de la electrónica.

1.4 Resumen

Dada la importancia de la experiencia práctica de los estudiantes, en el afán de diversificar los recursos didácticos para los profesores, éste se suma como uno más de las iniciativas en los cuales los profesores podrán apoyarse. Además, habiendo vivido un contingente como lo fue de COVID-19, este proyecto también será una herramienta didáctica alternativa accesibles desde casa, formando parte del herramental disponible en caso de acontecimientos similares en el futuro.

1.4.1 Antecedentes

Se realizará un análisis de los diferentes componentes necesarios para el funcionamiento de un osciloscopio digital. Igualmente, se explorarán las diferentes alternativas que existen en el mercado y cómo se compran esas alternativas del mercado con la propuesta de este documento.

1.4.2 Diseño

Durante este capítulo se propondrá la estructura de cada una de las etapas de manipulación de información en un osciloscopio digital. Se realizará posteriormente un análisis de cada transformación de esta información de manera superficial (dejando el análisis exhaustivo para el documento de anexos). En este análisis se realizará un análisis detallado del flujo de la información para cada fase del diseño; acondicionamiento, digitalización, interpretación y graficación de las señales leídas. Se hará una revisión de las diferentes herramientas que se consideraron para la creación del osciloscopio; circuitos de acondicionamiento, tarjetas de desarrollo y lenguajes de programación para la implementación. Finalmente, se realizará una demostración de las diferentes transformaciones, así como una visita de la interfaz de usuario (GUI) propuesta.

1.4.3 Conclusiones y trabajo futuro

Se hará una revisión de las solicitudes que se establecieron durante las primeras fases de diseño y el planteamiento del problema a resolver para posteriormente analizar la satisfacción de estos lineamientos por cumplir. Se propondrán diferentes maneras de dar soporte a este osciloscopio para la posteridad cercana y media; un repositorio público para actualizaciones y solución de problemas; un manual por medio del cual se dará a conocer el proceso del diseño orientado a las aquellas personas que busquen reproducir y modificar el osciloscopio para satisfacer sus necesidades específicas.

1.4.4 Apéndices

Se hará mención del material didáctico adicional propuesto, así como esclarecer aquellas instancias en las que se haya citado un apéndice. Finalmente, se hará una mención del documento de anexos y de la información que se podrá obtener de este documento.

En el apéndice A se verá el material adicional que se utilizó durante la fase de diseño y una demostración detallada de cada una de las transformaciones que se realizaron en el flujo de información.

En el apéndice B se verá una implementación concreta de este osciloscopio. Se determinarán los valores de los resistores necesarios para el acondicionamiento de la señal, partiendo de R_N, el aprovechamiento del protocolo de monitor serial y UDP y por último un ejemplo de la graficación.

2
ANTECEDENTES

2.1 Componentes de un osciloscopio

Dentro de los osciloscopios existen muchos tipos. Si se hace un análisis en el tiempo se puede ver que es una tecnología que se ha digitalizado, así como muchas otras. Así, el osciloscopio digital es el que se usa hoy día con prevalencia.

Merece la pena recordar que un osciloscopio es una herramienta para medir voltajes eléctricos a través del tiempo a una velocidad muy elevada.

Actualmente, los osciloscopios digitales se fabrican a partir de una serie de componentes bastante estandarizados. Si se hiciera un seguimiento de la información, desde la punta de medición hasta la pantalla, se verían varios componentes interactuando en un proceso de varias etapas.

- La punta de medición se carga eléctricamente al momento de hacer contacto con el componente eléctrico cuyo voltaje se quiere medir. Siempre se incluyen circuitos de acondicionamiento de señales, con objeto de que la señal de entrada se adecue al rango de voltaje aceptado por el osciloscopio.
- Existe un dispositivo llamado ADC, de las siglas en inglés de convertidor analógico a digital, que lee ese voltaje con respecto a otro voltaje referencia. Esto convierte el voltaje en información digital manejable por un circuito digital o un ordenador.
- Esa información pasa a un microprocesador, el equivalente a una computadora muy pequeña, que hace los cálculos que se requieren para determinar la diferencia de potencial entre las dos puntas del dispositivo.
- La información resultante de ese microprocesador se envía a una pantalla para dibujarse a un ritmo normalmente mayor a 24 cuadros por segundo, para mantener la ilusión de movimiento de la señal medida.

El dispositivo propuesto en esta tesis seguirá el mismo formato aquí analizado, sin importar qué decisiones se vayan a tomar posteriormente, deberán atañerse a este flujo de información. A pesar de que los osciloscopios más antiguos, normalmente llamados osciloscopios analógicos, no seguían exactamente este flujo, tenían uno parecido. Uno de los pasos que no existía era el paso a través del convertidor analógico a digital, ya que toda la información se manipulaba de forma analógica y continua. Estos métodos, aunque resultan muy interesantes, están más allá del enfoque de esta tesis.

Un esquema bastante general del dispositivo planeado será el que se muestra en la figura 2.1.

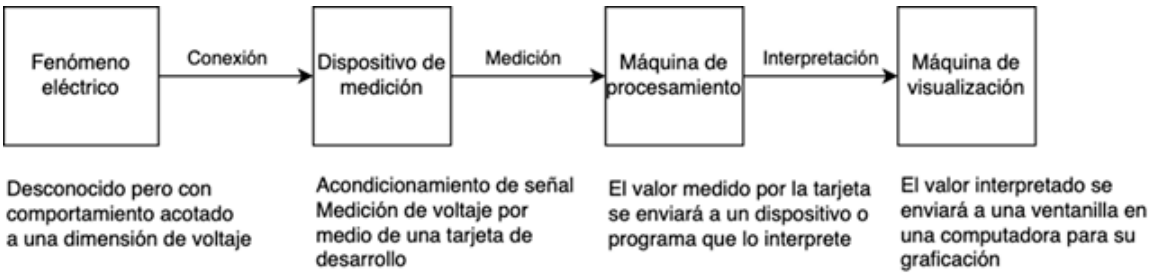


Figura 2.1: Flujo de información

2.2 Clasificación de osciloscopios

Se pueden considerar tres tipos de osciloscopio, así como tres factores de comparación importantes: calidad, precio y accesibilidad, es decir, facilidad de uso.

Los osciloscopios que pueden denominarse profesionales y que se venden para su uso en los laboratorios de enseñanza, tienen una altísima calidad y facilidad de uso, pero un precio más allá del accesible para un estudiante promedio.

Los osciloscopios llamados de bolsillo, generalmente se encuentran a un precio mucho más alcanzable, son muy fáciles de usar por su reducido número de capacidades, en comparación a uno profesional, y son capaces de cubrir cualquier actividad de la vida diaria, e incluso para muchas de las actividades de los laboratorios de enseñanza.

Los osciloscopios ”para entusiastas”, son diseños creados a partir de componentes de hardware. Son más limitados que uno de bolsillo y también son más difíciles de usar ya que involucran el paso del armado. El único rubro en el que superan con creces a los otros dos es en el rubro del precio.

Así que como ejemplo, se hará la comparación de los siguientes osciloscopios:

1. Profesional: Siglent Technologies SDS1104X-E, similar con los que se cuenta en los laboratorios de enseñanza
2. De bolsillo: Eujgoov - Osciloscopio de Mano DSO TC3
3. Para entusiastas: Medidor de señales con Arduino (por Meetechniek).

Tabla 2.1: Comparación de osciloscopios

Nombre	Ancho de banda	Canales	Voltaje de operación	Precio
Siglent	100MHz	4	1000 V	$8800
Eujgoov	500kHz	1	400 V	$800
Meetcechniek	9.6 kHz*	1	5 V*	*

*El medidor de señales diseñado por Meetechniek es únicamente un esquema básico para la realizar mediciones a altas velocidades con la tarjeta Arduino Nano. Los resultados son

los máximos que puede ofrecer el procesador de la Arduino Nano (ATM328p). Los valores de precio, voltaje y ancho de banda dependen del fabricante y del procesador utilizado.

El osciloscopio que aquí se pretende diseñar, será una alternativa especial por medio de la cual se intentará obtener la mayor cantidad de beneficios a partir de un osciloscopio de los denominados ”para entusiastas”. Ésta es la manera económicamente más factible de contar con un osciloscopio y, como se dijo desde un principio, esto es prioritario al momento de buscar poner este dispositivo en una situación económicamente alcanzable para los estudiantes de la Facultad de Ingeniería.

Aunque este osciloscopio no pueda competir con los digitales o los analógicos que ya se tienen en los laboratorios de clase, el nicho que intenta llenar es el de las prácticas de casa o la sustitución de equipo, en caso de que esos osciloscopios de mayor calidad llegaran a faltar por alguna contingencia.

2.3 Resumen

El osciloscopio que aquí se va a desarrollar, será una alternativa especial por medio de la cual se intentará obtener la mayor cantidad de beneficios a partir de un osciloscopio de los llamados ”para entusiastas”. Ésta es la manera económicamente más accesible de obtener un osciloscopio y, como se dijo desde un principio, esto es prioritario al momento de buscar poner la herramienta en un sitio económicamente alcanzable para los estudiantes de la Facultad.

3
DISEÑO

3.1 Análisis del problema

Para el diseño del osciloscopio conviene considerar una serie de requerimientos mínimos. La expectativa será alcanzar o, si es posible, superar estas especificaciones.

Requerimientos de diseño:

- Velocidad de muestreo de 1 kHz, o superior
- El menor número de partes, con objeto de mantener el precio accesible
- Procurar utilizar material que los estudiantes ya posean para los laboratorios de enseñanza
- Apoyarse en dispositivos electrónicos en los que los estudiantes ya tengan acceso, tales como teléfonos inteligentes y computadoras personales
- Diseñar un dispositivo fácil de armar y difícil de descomponer por descuido o falta de capacitación
- Hacer del proyecto lo más abierto y fácil de modificar posible, para que se pueda adaptar mejor a las necesidades de los profesores y alumnos que lo vayan a utilizar.

Una decisión que se tomó desde que se abordó este proyecto fue la aplicación de una metodología de diseño. Esto daría la posibilidad de alcanzar una de las mejores soluciones para la propuesta del osciloscopio accesible.

El diagrama del funcionamiento de un osciloscopio propuesto en capítulo de Antecedentes es útil porque permite visualizar estas fases por las que tiene que transitar la información que se quiere obtener.

Hay ciertas partes del diseño en las que se puede y otras en las que no se puede tener agencia, por ejemplo, no se puede modificar el comportamiento del fenómeno eléctrico, mientras que sí se puede modificar el circuito de acondicionamiento usado para medirlo.

En la figura 3.1 se muestra el mismo flujo de información que se había visto previamente, ahora con comentarios sobre el diseño realizado parte por parte.

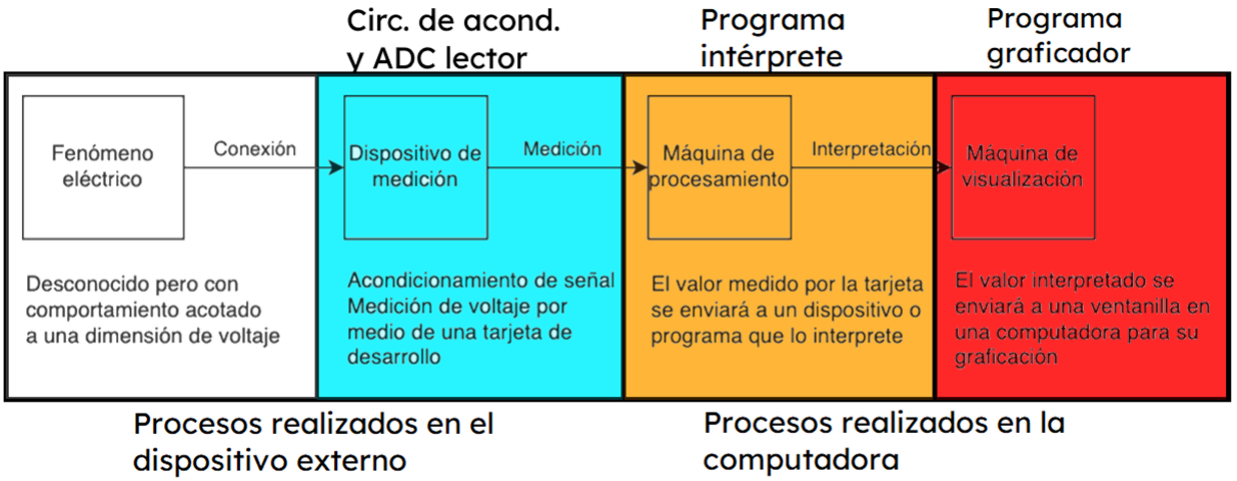


Figura 3.1: Flujo de la información anotado

En la tabla 3.1 se presenta un listado exhaustivo de qué componente del osciloscopio realizará cada proceso.

Tabla 3.1: Flujo de la información

Paso	Dispositivo
Conexión	Circuito de acondicionamiento
Acondicionamiento	Circuito de acondicionamiento
Medición de voltaje	ADC lector
Procesamiento numérico	Programa intérprete
Interpretación	Programa intérprete
Visualización	Programa graficador

Puede parecer redundante agregar la tabla 3.2, sin embargo, resulta práctico ya que no son tan evidentes las funciones de los componentes de forma ya integrada. A pesar de que el funcionamiento ya está descrito en esta tabla, falta agregar los canales de comunicación entre el ADC lector, el programa intérprete y el programa graficador. Merece la pena hacerlo únicamente para dar plena claridad.

Tabla 3.2: Flujo de la información con pasos exhaustivos

Paso	Dispositivo
Fenómeno eléctrico externo	Dispositivo externo
Conexión	Circuito de acondicionamiento
Acondicionamiento	Circuito de acondicionamiento
Medición de voltaje	ADC lector
Envío de información a intérprete	–
Procesamiento numérico	Programa intérprete
Interpretación	Programa intérprete
Envío de información a graficador	–
Visualización	Programa graficador

Los dos apartados señalados con dos guiones son abstractos, sin embargo, se tendrán que

implementar durante la fase de diseño concreto.

3.2 Primera aproximación abstracta

De acuerdo con el análisis del problema, merece la pena dar una primera propuesta sobre la cual construir. Esto permite descartar opciones de diseño que podían haber parecido viables al momento, sin embargo, ya basándose en los pasos del flujo de información, resultan más convincentes las decisiones de aquí en adelante.

3.2.1 Circuito de acondicionamiento

El voltaje leído por cualquier dispositivo tiene que estar acondicionado para no resultar dañino para el propio dispositivo y sus usuarios. El circuito de acondicionamiento de un osciloscopio necesita atenuar las señales de gran magnitud e idealmente amplificar las de pequeña. Existen dos maneras de conectar el circuito de acondicionamiento al fenómeno que se quiere caracterizar, por medio de un sistema acoplado, es decir, compartiendo tierra física, o desacoplado, sin compartirla. Este circuito tiene como requisito hacer llegar el voltaje acondicionado al ADC lector.

3.2.2 ADC lector

El ADC lector será el dispositivo por medio del cual se -medirá- el voltaje. Es importante recordar que los ADC funcionan generalmente con un voltaje de referencia. Esto quiere decir que, además de conocer las variables introducidas en el circuito de acondicionamiento, también será necesario conocer este voltaje de referencia. El ADC es un dispositivo que por sí solo únicamente realiza lecturas de un voltaje analógico y los escribe en pines binarios digitales. Conviene comentar que se requiere de un dispositivo adicional para comunicar esta información desde el ADC hacia el programa intérprete.

Entra aquí la propuesta de la tarjeta de desarrollo.

Muchas tarjetas de desarrollo incluyen un ADC interno, además de un procesador con capacidades de comunicación serial con otros dispositivos y, a veces, conectividad por medio de Bluetooth o Wifi. El uso de una tarjeta de desarrollo da la oportunidad de hacer las lecturas de voltaje y comunicar los valores leídos al programa intérprete con un solo dispositivo externo. Existen centenares de diferentes fabricantes de tarjetas, esto da la oportunidad de obtener una suficiente para las especificaciones planteadas al inicio, manteniendo un presupuesto bajo.

3.2.3 Programa intérprete

Una vez que la información es leída por el ADC y enviada al programa intérprete, la información es únicamente un número entero entre el 0 y el valor 2^{n-1}, donde n es la resolución en bits del ADC. Necesita interpretación posterior. También hay que recordar que los voltajes leídos por el ADC ya habían sido modificados por el circuito de acondicionamiento. Estos datos, además, pueden ser enviados en diferentes formas, como se había dicho previamente; comunicación serial, Bluetooth y Wifi, son solo unas de las alternativas.

Para mantener la simplicidad y compatibilidad al máximo entre tarjetas de desarrollo, se pensó que lo ideal sería utilizar como implementación inicial el protocolo llamado comunicación serial.

Es por esto por lo que se propone un programa específico para dar significado cuantitativo a los valores recibidos en este paso del flujo de información.

Para diseñar este programa, se necesitará saber qué atenuación se le está realizando al voltaje con el circuito de acondicionamiento, además de conocer la resolución y el voltaje de

referencia del ADC.

Este programa también deberá garantizar la conectividad con el programa graficador. De aquí nace la idea del uso de dos canales de comunicación entre programas, uso de memoria compartida y mensajes TCP/UDP.

En las primeras iteraciones de este proyecto, se intentó tanto TCP como UDP. Ambas alternativas proponen una interesante idea, ya que incluso permitiría la interconectividad entre el intérprete y el graficador, sin estar corriendo ambos procesos en la misma computadora. Posteriores iteraciones, no referidas en este documento, trabajan con memoria compartida por la mayor velocidad que provee.

3.2.4 Programa graficador

Por último, éste es el programa con el cual el usuario tendrá más interacción. En realidad, lo ideal sería que el usuario pudiera conectar el dispositivo sin configurar nada, y abrir directamente este programa para visualizar las señales leídas.

Este programa deberá extraer la información enviada desde el programa intérprete. Algo importante a mantener en mente, es que esta información recibida desde el intérprete ya será un valor inmediatamente utilizable. Esto quiere decir que cada par de valores recibidos será un voltaje y un tiempo, en concreto, el tiempo en el que se recibió ese valor de voltaje.

Recordando los objetivos del proyecto, es imperativo mantener la compatibilidad con la mayor cantidad de dispositivos posibles. La decisión del lenguaje de programación, así como el uso de memoria compartida para comunicar ambos programas, estuvieron altamente influidos por este factor.

3.3 Búsqueda de soluciones

A partir de la aproximación anterior, se pueden encontrar múltiples alternativas para obtener los resultados deseados.

3.3.1 Circuito de acondicionamiento

El diseño de este circuito es libre, en ese sentido, resulta muy complicado proponer tres alternativas para éste. Parecería más adecuado proponer familias de diseños y componentes a usar.

Se contemplaron múltiples familias de circuitos para el acondicionamiento, entre ellas, éstas destacaron:

a) Circuitos con amplificadores operacionales

b) Circuitos RC desacoplados

c) Divisor de voltaje con dos ADC

Cada una de las alternativas representa beneficios y detrimentos explorados brevemente:

a) Amplificadores operacionales

Esta familia de circuitos contempla el uso de un componente muy común en el diseño de electrónica analógica, el amplificador operacional. Éste es el tipo de circuito que normalmente se utiliza en el acondicionamiento de señales a nivel industrial, ya que el uso de estos componentes permite realizar interesantes operaciones con las señales trabajadas, tales

como multiplicaciones, divisiones, inversiones, desplazamientos etc. Se realizaron pruebas con estos componentes, en concreto, con amplificadores operacionales de bajo voltaje. Esta familia de circuitos fue descartada ya que no se encontró ningún circuito integrado con el cual se pudiera trabajar de la forma requerida sin necesitar de un voltaje de alimentación dos veces más grande que el suministrado por la mayoría de las tarjetas de desarrollo. No se encontró ninguna que operara correctamente con 5 V y pudiera realizar la operación de inversión.

b) Circuitos RC desacoplados

Los circuitos RC desacoplados dan una propuesta interesante en cuanto a la seguridad del dispositivo. La propuesta constaba de trabajar con tierras desacopladas entre el dispositivo del que se realiza la lectura y del dispositivo medidor. Esto generalmente es una buena idea al momento de leer fluctuaciones en corriente alterna, ya que se ignora el componente de directa una vez transcurrido suficiente tiempo. Esta familia de circuitos fue descartada por dos razones, diseñar un circuito de esta forma conllevaba mayor dificultad para realizar las mediciones de corriente directa, lo que quitaba la posibilidad de utilizar el dispositivo como multímetro de voltaje, dejándolo únicamente como dispositivo medidor de fluctuaciones de voltaje.

Adicionalmente, ya que estos circuitos involucraban condensadores de desacoplo, el comportamiento no se podía garantizar de forma cuantitativa. A pesar de que los valores observados y verificados en simulación eran cualitativamente correctos, cuantitativamente eran mucho más difíciles de describir.

c) Divisor de voltaje con dos ADC

Esta es la familia de circuitos con la que se diseñó el proyecto. Esta familia de voltajes funciona de forma acoplada. Una medición consta de dos lecturas de voltaje, una del voltaje de la señal y otra de la tierra sobre la cual se tiene referido ese voltaje. Esto quiere decir que se tienen que realizar dos mediciones para obtener una única lectura de voltaje, sin embargo, como lo dice el nombre de la familia de circuitos, únicamente se necesitaba de un divisor de voltaje por cada ADC. Por consiguiente, fácilmente se podían realizar las mediciones utilizando únicamente resistores.

A pesar de traer ligeras complicaciones con ciertas tarjetas de desarrollo, en particular aquellas que únicamente pueden realizar mediciones analógicas en un solo pin, presentaba una gran ventaja: únicamente se requerirían resistores, que además de baratos, llevan a un modelado matemático lineal constante con muy poca variación.

3.3.2 Tarjeta de desarollo

Entre las tarjetas de desarrollo, existen varias que destacan por su precio y facilidad de uso. Un requisito adicional que se tuvo que contemplar fue la facilidad de obtención de la tarjeta. Ya que este documento tiene como enfoque la Facultad de Ingeniería y, en general, el estudiantado mexicano, se procuró que todas las tarjetas fueran fáciles de conseguir en las tiendas minoristas de electrónica más abundantes, o por compra en línea.

A nivel nacional, Steren es la tienda más común en artículos de electrónica de modelado; es por esto por lo que se procuró que todos los componentes se pudieran obtener en una de sus sucursales. Minoristas más pequeños en la Ciudad de México también fueron tomados en cuenta, tales como SandoRobotics y 330 Ohms.

Amazon México, AG Electrónica y Mouser México fueron las tiendas digitales se pueden contemplar para compra en línea.
Las tres tarjetas candidatas para el diseño de este circuito fueron:

a) Arduino Nano

b) ESP8266

c) ESP32

A pesar de que las tres tarjetas se pueden utilizar, el proceso de diseño terminó orientado mayormente al desarrollo de la ESP32. Como se vio previamente, el diseño del circuito de acondicionamiento depende de dos ADC. En la práctica, esto quiere decir que deben existir dos pines en la tarjeta en la cual se puedan realizar mediciones analógicas de forma secuencial: primero se lee el valor en un pin y luego en el otro. Este comportamiento se puede obtener en cualquiera de las tres tarjetas, sin embargo, ya que la ESP32 cuenta con dos ADC, se consideró que se podrían obtener mayores velocidades de lectura.

3.3.3 Lenguaje de programación

El lenguaje de programación usado para las aplicaciones intérprete y graficador, fue otro tema donde existieron varias opciones con las cuales se realizaron pruebas.

Se consideró que se podrían utilizar los siguientes lenguajes:

a) C++

b) C#

c) Java

Dos factores fueron los principales al momento de tomar la decisión de qué lenguaje de programación se usaría; compatibilidad entre plataformas y habilidad del autor de esta propuesta en el lenguaje.

C++ es interesante por la velocidad de ejecución del código, manteniendo compatibilidad entre plataformas, esto, con una consideración: en el flujo de información existe un punto en el que se tiene que leer el puerto serial. El puerto serial, denominado COM en Windows, TTY en Linux y Mac, es un puerto manejado por el sistema operativo, esto quiere decir que el proceso de compilado tiene que ser diferente, según el sistema operativo en el que se esté trabajando. Existen archivos de biblioteca que se pueden utilizar para programar sobre una capa de abstracción que resuelve estas incompatibilidades entre sistemas operativos, sin embargo, no se consiguió que funcionaran de la forma deseada.

C# presenta una serie de ventajas por la diversidad de archivos de biblioteca que incluye por defecto. Esto hace sumamente fácil el desarrollo de aplicaciones, sin tener que buscar completar demasiadas dependencias externas. El problema por el cual se descartó C# fue la incompatibilidad entre las interfaces gráficas, en particular el de Windows Forms con el sistema operativo MacOS. Al momento de escribir este documento, ya existe un archivo de biblioteca llamado Maui, el cual propone resolver estos problemas en las aplicaciones gráficas; sin embargo, al momento del diseño de este proyecto, era algo que no existía.

Java, por su parte, a pesar de ser más lento que C++ y generalmente también que C#, ofrece la ventaja de correr sobre una capa de compatibilidad ya con muchos años de antigüedad, la JVM. Esto quiere decir que, sin importar qué archivo de biblioteca se use, existe suficiente garantía de que el código escrito en Java en una computadora pueda correr en otra sin importar su sistema operativo. El factor más decisivo al momento de elegir Java fue encontrar que el Arduino IDE también estaba programado en Java. El Arduino IDE, y también este proyecto, dependen de un archivo de biblioteca llamado jSerialComm, que facilita muchísimo el manejo del puerto serial.

A pesar de que se eligió Java como el lenguaje para crear esta implementación, no quiere decir que algún lector de este documento no pueda utilizar el lenguaje de su preferencia. Por la naturaleza de este proyecto, al estar todo dividido en partes, uno podría fácilmente reemplazar la programación en cualquiera de dos programas y volver a implementarlo a su gusto, sin necesidad de reprogramar o rediseñar el osciloscopio.

3.4 Diseño final

En esta sección se abordarán con más lujo de detalle los puntos identificados en la primera aproximación abstracta. Sin embargo, la elaboración de este diseño llevó muchas opciones fallidas, desarrollos de álgebra y más de mil líneas de código; es por esto que mucho de este contenido se proporcionará para su consulta en los documentos adjuntos, no como parte del cuerpo del documento.

3.4.1 Esquema general

Retomando el esquema elaborado durante la primera aproximación abstracta, se pueden verificar qué puntos se concretaron durante la búsqueda de soluciones, tal como se muestra en la tabla 3.3.

Tabla 3.3: Flujo de la información implementado

Paso	Dispositivo
Fenómeno eléctrico externo	Dispositivo externo
Conexión	Divisor de voltaje
Acondicionamiento	Divisor de voltaje
Medición de voltaje	ADC de ESP32
Envío de información a intérprete	ESP32
Procesamiento numérico	Intérprete (Java + jSerialComm)
Interpretación	Intérprete (Java)
Envío de información a graficador	Intérprete (Java)
Visualización	Graficador (Java)

3.4.2 Dispositivo de medición

Merece la pena realizar el análisis del dispositivo de medición como uno solo, conformado por dos partes, que se describen a continuación.

3.4.2.1 Circuito de acondicionamiento

El circuito que se diseñó fue el mostrado en la figura 3.2.

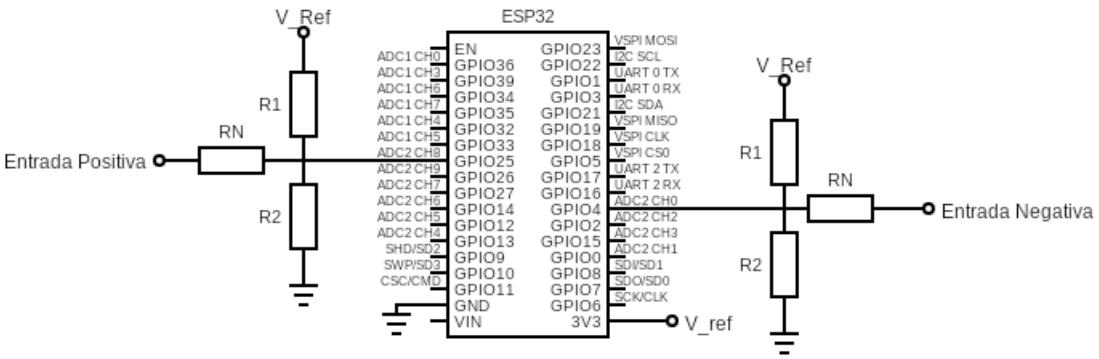


Figura 3.2: Diagrama del dispositivo de medición

Dadas las limitaciones de las diversas herramientas de ilustración y simulación, se muestra en la siguiente tabla las diferentes formas en las que se podrán encontrar escritos diferentes nombres de los resistores y de los voltajes que se encontrarán más adelante.

Nombre completo	Notación de guión	Notación de subíndice
Voltaje de tierra	V_tierra ó V_gnd	V_{tierra} ó V_{GND}
Voltaje de referencia	V_referencia ó V_ref	$V_{Referencia}$ ó V_{Ref}
Resistor de referencia	R1 ó R_1	R_1
Resistor de tierra	R2 ó R_2	R_2
Resistor de entrada (También llamada terminal)	RN ó R_N	R_N
Voltaje de entrada (positiva o negativa)	V_ T (V_TPos ý V_ TNeg)	V_T ($V_{T_{Pos}}$ ý $V_{T_{Neg}}$)

En la figura anterior, se muestra una tarjeta de desarrollo genérica y los pines que están en uso para conectar el arreglo de resistores, que son arbitrarios. En una implementación real, lo único importante a considerar es que esos pines tengan acceso a la lectura del ADC.

Habrá dos terminales al momento de realizar cualquier medición de voltaje, ambas se denotarán como V_T ó V_T , una de estas dos terminales se considerará la terminal de referencia y la otra la terminal de voltaje con respecto a esa referencia.

Lo primero que se tiene que calcular, es el voltaje resultante en los pines de medición ante una aplicación de voltaje en la entrada. Se puede ver que los tres resistores, (llamados R1, R2 y RN, convergen en un mismo nodo para ambos arreglos. Este nodo se le llamará N. Se deberá calcular el voltaje de N en función del voltaje introducido en las entradas.Es fácil ver que este voltaje será exactamente el voltaje presente en R2.

Ya que el arreglo de resistores es idéntico para ambas lecturas, positiva y negativa, la función para describir el voltaje en el nodo N será igual también.

Esta función es la siguiente:

$$V_N(V_T) = V_T \frac{R_1 R_2}{R_1 R_2 + R_N R_1 + R_N R_2} + V_{Ref} \frac{R_N R_2}{R_1 R_2 + R_N R_1 + R_N R_2}$$

Con álgebra sencilla se puede describir el valor de V_T en función de V_N:

$$V_T(V_N) = \frac{V_N(R_1 R_2 + R_N R_1 + R_N R_2) - V_{Ref}(R_N R_2)}{R_1 R_2}$$

Esto da una manera fácil de calcular el voltaje en la terminal T cuando se lea el valor de voltaje en el nodo N.

Como nota, para que el circuito sea seguro, se tiene que mantener el voltaje en N dentro del intervalo de operación recomendado. Generalmente los valores límite son el voltaje de referencia, (V_{Ref}), el superior, y el voltaje de tierra, (V_{GND}) el inferior.

La desventaja que se presenta al únicamente utilizar resistores en el diseño de este circuito es que no resulta posible hacer nada aparte de atenuar la señal de entrada. Sin embargo, por lo confiables que son, su comportamiento lineal fácil de predecir y, sobre todo, su precio, el resistor terminó siendo el único componente indispensable para esta implementación.

En resumen, el diseño del dispositivo de medición consiste en un circuito armado a partir de resistores, por medio de los cuales se acondiciona la señal de entrada, además de una tarjeta de desarrollo que puede ser Arduino Uno, ESP-32, Arduino Nano, Raspberry Pi, entre otros, por medio de la cual se realizará la medición del voltaje ya acondicionado.

3.4.2.2 ADC y envío de información

La manera más sencilla de programar una tarjeta de desarrollo, como lo muestra su amplio uso en aulas, es por medio del Arduino IDE.

Se han de proponer dos pines: uno llamado A para la lectura positiva del ADC; otro llamado B para la medición del voltaje negativo o de tierra del dispositivo externo. Se realiza entonces una resta de la lectura de ambos voltajes. Posteriormente se traduce el valor obtenido a un cifrado diseñado para optimizar velocidad y se envía a la computadora en la que se realizarán los cálculos posteriores.

El pseudocódigo que representa ese proceso es el siguiente:

```
int lectura_ADC_A
int lectura_ADC_B
int pin_ADC_A
int pin_ADC_B
int resta_ADC_AB

inicializa(){
    //De este valor dependerá la velocidad de envío
    configurar_tasa_baudios_monitor_serial(máximo_valor_estable)
}

ciclo(){

    //Se leen los valores de las terminal positiva (A)
    //y la terminal negativa (B)
    lectura_ADC_A = leer(pin_ADC_A)
    lectura_ADC_B = leer(pin_ADC_B)

    //Al valor de la terminal positiva se le ha de restar
    //valor de la terminal negativa
    resta_ADC_AB = lectura_ADC_A - lectura_ADC_B

    //Se convierte la resta al cifrado propuesto
    cifrar(resta_ADC_AB)

    //Se manda el valor a la computadora
    mandar_serial(resta_ADC_AB)
}
```

Es únicamente un pseudocódigo; sin embargo, el cambio a código de Arduino o de C es una traducción casi directa. En el caso del cifrado, para maximizar la velocidad y la estabilidad del envío de información de la tarjeta de desarrollo hacia la computadora, se decidió enviar el número resultado de la resta RESTA_ADC_AB en el formato de dos bytes. En el caso de la tarjeta que se utilizó como referencia, la ESP32, la resolución máxima es de 12 bits. Dado que se tienen que mandar los paquetes de información en bytes completos, se tendrán que enviar necesariamente dos paquetes de 8 bits. Los 4 bits sobrantes se usarán como indicadores del signo del resultado de la resta, positivo o negativo, y el orden de los dos bytes al momento de leerlos. Como se leerá un flujo constante de bytes en la computadora, se requerirá una manera de identificar el inicio y el final de cada paquete de dos bytes.

Si se enviaran los números sin esta codificación, sería terriblemente ineficiente por la impresión de caracteres en vez de valores directamente numéricos. Además, se enviarían diferentes cantidades de bytes para los números de un dígito, de 0 a 9, de dos, de 10 a 99,

de tres, de 100 a 999, etc. Esto es sin considerar que también habría que enviar el carácter adicional "-" para conocer el signo del número.

Ya que es una resta de dos números de 12 bits, el número que se obtendrá de la resta será un número entre -4095 (0 - (4095)) y 4095 (4095 - (0)). El formato de número entero normal en Arduino y en C es un número de 32 bits, sin embargo, para esta implementación un número de 16 bits es suficiente.

En el siguiente recuadro se explicará el cifrado propuesto y utilizado para enviar los valores leídos por la tarjeta de desarrollo por medio del puerto serial. Ya que este cifrado se diseñó con una resolución de 12 bits con signo (13 bits) en mente, puede utilizarse para cualquier resolución menor.

```
Considerando que ya se cuenta con el resultado de la resta de los valores leídos por
↪ los ADC de la tarjeta de desarrollo, se usará este valor "RESTA_ADC_AB" (un
↪ número de 16 bits) como base para la decodificación. El primer byte estará
↪ representado con A y el segundo con B.

RESTA_ADC_AB:     AAAA_AAAA BBBB_BBBB

Se espera que los cuatro bits más significativos sean 0 ya que el número está entre 0
↪ y 4095 en valor absoluto. De ser el primer número un 1 eso significará un
↪ desbordamiento durante la resta. Quiere decir que el resultado de la resta fue
↪ negativo (valor coincidente con el "complemento a uno" del número).

Visto en binario
Número positivo:  0000_MMMM MMMM_MMMM
Número negativo:  1111_NNNN NNNN_NNNN

Con esta consideración, se pueden ignorar los primeros cuatro bits de este punto en
↪ adelante siempre recordando si el número es positivo o negativo.

Se dividen los 12 bits restantes de una manera más conveniente. En este caso, eso
↪ será en dos paquetes de 6 bits. Esto se hace una vez se conozca el signo.

AAAA BBBB_BBBB -> AAA_ABB BBB_BBB

Se crearán dos bytes en los cuales se guardarán estos 12 bits de información. Estos
↪ dos bytes se representarán con X y con Y y se les realizarán ciertas
↪ preparaciones previas al guardado de los valores de bytes A y B.

Primer byte:    XXXX_XXXX
Segundo byte:   YYYY_YYYY

Se pone un 1 al inicio del primer byte y un 0 al inicio del segundo para diferenciar
↪ cuál es el inicio del paquete de dos bytes y cuál es el final. Durante la lectura
↪ se recibirá un flujo constante de bytes por lo que se necesitará una manera de
↪ distinguirlos.

Primer byte:    1XXX_XXXX (inicio)
Segundo byte:   0YYY_YYYY (final)
Primer byte:    1XXX_XXXX (inicio de otro paquete)

Se cambia el segundo bit a 1 si el número es negativo y a 0 si el número es positivo.
↪ Esto debe hacerse para los dos bytes.

Primer byte (pos.): 10XX_XXXX       Primer byte (neg.):  11XX_XXXX
```

```
Segundo byte(pos.): 00YY_YYYY       Segundo byte(neg.):  01YY_YYYY

Por último, en los 6 espacios residuales de los dos bytes se agrega el valor de los
↪  12 bits de resta de los valores de voltaje leídos.

Primer byte (pos.): 10AA_AABB       Primer byte (neg.):  11AA_AABB
Segundo byte(pos.): 00BB_BBBB       Segundo byte(neg.):  01BB_BBBB
```

Así, se enviará el primer byte, luego el segundo byte, uno tras otro, un flujo continuo de paquetes de dos bytes con tal de optimizar la velocidad del puerto serial. Esto representa una gran ganancia de eficiencia sobre enviar los valores en forma de caracteres, puesto que un solo carácter se representa en un byte completo.

Entonces, ya que se sabe qué valores se enviarán y qué representarán, merece la pena seguir a la lectura e interpretación de ese flujo de bytes. Esto significa descifrar los paquetes de dos bytes, convertirlos en un número entero y, por último, calcular el voltaje que representan.

La implementación de este programa de cifrado y comunicación se detalla en el apéndice A.2.3.

3.4.3 Dispositivo de procesamiento

A partir de este momento, todo el desarrollo del proyecto se realizará por medio de programación y pseudocódigo.

Para este punto, ya se ha enviado del dispositivo físico un número que contiene la información del voltaje leído; el siguiente paso consiste en crear una máquina que pueda interpretarlo con suficiente velocidad y obtenga valores congruentes con el comportamiento esperado. Esta máquina lógica será un programa en la computadora. Decodificador traductor, o intérprete traductor, parecen ser los términos más sugerentes, ya que se traducirán los valores enviados desde la tarjeta de desarrollo, los cuales están cifrados, en valores de interés: la diferencia de voltaje entre las terminales positiva y negativa.

Entonces, el intérprete traductor es un programa que toma el flujo de valores provenientes de un puerto serial, los descifrará y los interpretará como un voltaje. Cabe la pena mencionar que todavía no se graficarán estos valores, únicamente se obtendrán los valores del voltaje recibido desde el puerto serial. La graficación se tratará en la sección siguiente.

El primer paso es descifrar los números que llegan al monitor serial. Se asume un ejemplo análogo al siguiente:

```
0010_1110
1001_1011
0011_1100
1001_1001
```

Solo de ver el primer dígito de cada byte se puede inmediatamente inferir qué byte inicia cada paquete y cuál lo termina.

```
0010_1110   //fin de número (se descarta a falta de inicio)
1001_1011   //inicio de número
0011_1100   //fin de número
1001_1001   //inicio del siguiente número
```

El paso siguiente es leer el segundo bit del paquete de dos bytes. Si ese número es 0 el número será positivo, si es 1, negativo.

```
1_0_011011   //inicio de número positivo
0_0_111100   //fin de número positivo
```

En este ejemplo, el número entre los guiones bajos es 0, esto quiere decir que el número es positivo.

Para concluir con el cifrado, los seis últimos bits concatenados serán el valor del número.

```
10_011011
00_111100

011011 111100 -> 0110_1111_1100 -> 1788
```

Este número se tiene que interpretar como la proporción que representa en el rango de la resolución del ADC. Como la tarjeta objetivo es la ESP32 con resolución de 12 bits y el voltaje de referencia es el de la propia ESP32, se trabajará en el rango de -4095 (0 - (4095)) y 4095 (4095 - (0)) y 3.3 V. Este valor obtenido es la diferencia de los valores obtenidos por los ADC_A y ADC_B, al que se llamará deltaADC.

Siguiendo con el ejemplo previo, se necesita normalizar el valor obtenido con anterioridad a los valores entre -3.3 V y 3.3 V.

```
DeltaVRN = VRef * deltaADC /(resolucion * 2);
```

Donde VRef es el voltaje de referencia de la tarjeta de desarrollo y la resolución es la resolución de su ADC. (No es necesario hacer la consideración del bit del signo ya que no contribuye con la dimensión del voltaje, solo su signo). DeltaVRN será la diferencia de voltaje -real- entre los nodos RN de la terminal positiva y la terminal negativa. Este voltaje es el voltaje entre los nodos acondicionados, no el voltaje de las terminales de medición.

Posteriormente, se necesita obtener el valor de la diferencia entre las terminales de medición, deltaVT, a partir de la función inversa mencionada durante el planteamiento del circuito de acondicionamiento. Esta función se demuestra en el apéndice apéndice A.2.1.

$$\Delta(V_{TAB}) = (V_{RN_A} - V_{RN_A})(\frac{R_1R_2 + R_NR_1 + R_NR_2}{R_1R_2})$$

```
DeltaVT = DeltaVRN *(R1 * R2 + RN * R1 + RN * R2)/(R1 * R2);
DeltaVT = 21.82576006043632
```

En este punto amerita mencionarse que ya no se está trabajando con el voltaje del nodo RN, sino con la diferencia de voltajes entre los dos nodos RN: el de la terminal positiva y el de la negativa.

Es importantísimo que no queden dudas en este punto; al realizar la resta de los voltajes en el código de la tarjeta de desarrollo, ya no se podrá conocer el voltaje de las entradas leídas con respecto a la tarjeta, únicamente se conocerá la diferencia de los voltajes entre las dos entradas. A cambio de perder esta información, se obtiene con mucha rapidez la diferencia de voltaje entre las dos puntas de medición, tal y como lo haría un osciloscopio de mercado.

Al momento de realizar esta última operación, se incurre en un problema: el programa

traductor no tiene una manera directa de saber el valor de los resistores ni el voltaje de referencia de la tarjeta. Es por ello que esos valores se tienen que suministrar manualmente al programa.

Dado el rango de voltajes que se utilizan en los laboratorios de enseñanza, se consideró que tener un intervalo de voltajes de -50 a +50 V era suficiente para ser seguro en casi cualquier escenario, sin sacrificar demasiado la calidad de la señal leída por atenuación.

En el apéndice A.2.1.1 se demuestra el cálculo de los resistores necesarios. Para este intervalo de voltajes los resistores resultantes son los siguientes:

```
R1 = 217.8
R2 = 233.2
RN = 3300.0
```

En la tarjeta ESP32, el valor del voltaje de referencia es de 3.3 V, por lo que es un valor conocido a priori.

Por último, el valor de V_{RN} es el valor recién interpretado a partir de los valores leídos del ADC. Así, sustituyendo:

```
V_T = 21.8257 V
```

Una manera de convencerse de que este número no se obtuvo por alguna casualidad, es calcular la proporción V_T / V_Max y comparar esa proporción con el número obtenido en la división Delta_ADC / $2^{resolucion}$.

```
21.8257 / 50.0 = 0.4365
1788 / 4096 = 0.4365
```

3.4.4 Mensajero UDP, buzón UDP

Para el diseño de esta herramienta, merece la pena imaginar que ya se encuentre programada la aplicación de graficación. Esa aplicación de graficación necesita recibir los valores de voltaje por medio del protocolo UDP. El protocolo UDP es parecido a una antena de radio; no importa si están llegando ondas al radio o no, uno puede tener su radio escuchando una frecuencia sin recibir nada. Al igual que la radio, no importa si no hay ningún canal UDP escuchando, las señales se pueden emitir al puerto con la esperanza de que lleguen a algún destino.

Siguiendo esta analogía, la aplicación de graficación se comporta como una radio de casa esperando las señales que llegan de fuera. Entonces, la aplicación graficación tendrá una especie de buzón al que llegarán los valores de voltaje.

Para que esos valores lleguen a la aplicación de graficación, habrá que transmitirlos desde la aplicación traductor o intérprete. Una vez que un valor de voltaje se haya interpretado, ese valor se mandará a un socket en la computadora. La aplicación de graficación estará escuchando ese socket, tomará el valor e inmediatamente lo graficará.

Así, se podrá establecer un protocolo de comunicación entre la aplicación de traducción con la aplicación de graficación.

A su vez, si se decidiera en algún momento crear una aplicación que no interprete valores provenientes de la aplicación intérprete sino valores generados de forma ideal por medio de la propia computadora, al transmitir los valores a ese socket, se podría igualmente leer, interpretar y graficar esos valores ideales.

Esto resultaría útil, porque sin necesitar de un generador de señales, se podrían simular funciones sencillas como la sinusoidal, la diente de sierra, el escalón, etc. y manipularlas obteniendo ejemplos ideales de estas señales.

3.4.5 Graficación

El diseño de la aplicación de graficación se basa en la premisa de ya contar con una lista de valores de voltaje y tiempo provenientes desde la aplicación intérprete. Esos voltajes son la diferencia de los valores de las terminales positiva y negativa del osciloscopio. En esta fase, lo único restante es graficar en algún dispositivo estos valores de voltaje.

Por diseño se decidió que la aplicación de graficación debía ser completamente independiente de la aplicación de traducción. Esto ofrece ciertas ventajas como poder generar señales simulando una entrada de valores que en realidad no existe. De esta manera uno podría simular señales sinusoidales, diente de sierra, escalones, entre otras, sin necesitar un generador de señales externo y medirlas usando otro dispositivo de medición.

Dicho de otra forma, se decidió que el graficador pudiera graficar señales reales medidas con la tarjeta de desarrollo, así como señales simuladas provenientes de la propia computadora.

Para lograr esto, se decidió que todo el tráfico de datos fuera a través de puertos de red usando el protocolo UDP.

3.4.5.1 Ventana Principal

Los osciloscopios cuentan con un diseño un tanto universal en la pantalla, como se puede observar en la figura 3.3.

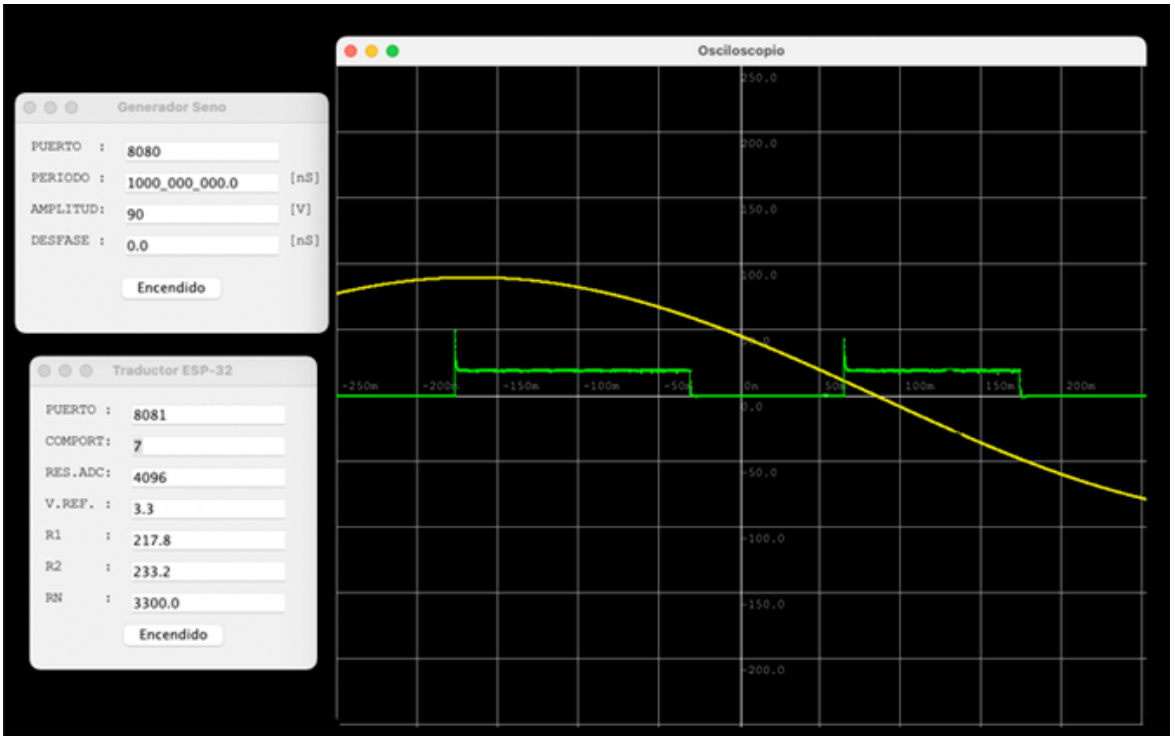


Figura 3.3: Ejemplo de visualización

No se requiere demasiado análisis para identificar que se necesitan dos valores para poder graficar un voltaje a través del tiempo; un voltaje y un tiempo.

Dado que la implementación presentada trabaja con el protocolo UDP, existen dos diferentes momentos en los que se podría medir el tiempo para graficar: el momento de generar la señal que se va a enviar por UDP y el momento en el que llega esa señal al buzón de UDP de la aplicación de graficación. Tras experimentar con las dos opciones, se optó por siempre usar el tiempo de generación de la señal. Esto agrega un valor a enviar por medio del mensajero UDP; se enviará el tiempo en que se interpretó el voltaje y el voltaje mismo.

El diseño del código para la graficación, así como las matemáticas involucradas, son un tema fuera del enfoque de este documento por lo que se incluyen en el capítulo apéndices B.

3.4.5.2 Controles de usuario

En todos los osciloscopios existen diferentes formas en las que uno puede manipular la representación de la señal que se está visualizando. Al momento del diseño de este programa, se consideró que los fundamentales son:

- Escala en el eje Y (voltaje)
- Escala en el eje X (tiempo)
- Desfase en el eje Y (individual para cada canal)
- Desfase en el eje X (compartido por todos los canales).

Generalmente estos son los únicos que se consideran primordiales. Sin embargo, como este proyecto está dirigido al uso principalmente de estudiantes, se decidió agregar también un modo de visualización de figuras de Lissajous. Además, dada la naturaleza discreta de la medición de los voltajes, también se agregó la posibilidad de ver las señales únicamente como puntos o como puntos conectados por líneas.

También se agregó la posibilidad de pausar la imagen mostrada en cualquier momento.

Todas estas funciones se programaron para funcionar con entrada de usuario por medio de teclado, pero también se agregó la posibilidad de modificar directamente los valores de escala y desfase por medio de una pequeña interfaz gráfica, haciendo uso del cursor y el teclado. Idealmente, un usuario acostumbrado al uso de este osciloscopio debería de poder moverse con naturalidad únicamente usando el teclado, sin embargo, para nuevos usuarios puede resultar más interactivo el uso de la interfaz gráfica.

3.5 Pruebas

En la tabla 3.4 se presentan los componentes, funciones y detalles considerados en la realización de las pruebas de funcionamiento del osciloscopio diseñado.

Tabla 3.4: Tabla de pruebas

Componente	Función	Detalles
Circuito de resistores propuesto	Atenuación y conexión al ADC	Intervalo de seguridad de -50 a +50 V
Tarjeta ESP32	Lectura y cifrado	Tasa de muestreo de ±12.5 kHz con resolución de 12 bits
Traductor implementado en Java	Interpretación de los valores cifrados y envío en UDP	Velocidad máxima de envío sujeta únicamente a la velocidad de la computadora de 90 kHz estables en pruebas
Graficador implementado en Java	Graficación de voltajes recibidos por UDP	Gráfica y manipulación sencilla de la señal enviada. Visualización de 2 canales y modo de figuras de Lissajous

Con suficiente optimización, se logró una tasa de muestreo de la tarjeta ESP32 de entre 12.5 kHz y 13 kHz, según el fabricante. Esto es más de diez veces la frecuencia de muestreo que se había propuesto en un principio.

La calidad de las mediciones está en un rango de ±10% con respecto al valor real. Existen tres factores principales que desvían este valor de la medición exacta:

1. Los resistores utilizados nunca podrán ser exactamente de los valores que se calcularon en el modelado. Se tiene que sacrificar ligeramente la calidad de las lecturas, utilizando resistores con valores comerciales. Se podrían utilizar resistores de alta precisión con los valores solicitados, pero éstos violarían el principio de ser baratos y fáciles de conseguir.

2. Las mediciones de los dos ADC de la ESP32 no es exacta todo el tiempo; de hecho, está sujeta a mucho más ruido del que sería conveniente. Existen múltiples opciones para atenuar este ruido; Espressif, que es la empresa que fabrica el microcontrolador de la ESP32, recomienda el uso de un filtro pasa-bajas con un condensador y muestreo múltiple. Se implementaron ambas propuestas de solución durante la fase de pruebas de este osciloscopio. Se decidió dejar de lado el filtro pasa-bajas por la poca o nula diferencia que ofrecía con respecto al circuito sin filtro. El uso de muestreo múltiple para obtener un valor promedio de las lecturas reduce el ruido considerablemente, sin embargo, también reduce la tasa de muestreo de manera proporcional.

3. Existe un componente de ruido que nace de la propia tarjeta ESP32 en sus terminales de 3.3 V, denominada 3V3, y su terminal de tierra o GND. Este ruido normalmente no sería suficiente como para crear una verdadera desviación en las lecturas del ADC en circunstancias normales, sin embargo, dado que con el circuito de acondicionamiento se está realizando una atenuación enorme, el valor del ruido de la tarjeta ESP32 se vuelve significativo.

3.6 Resultados

La demostración de cada una de las funciones inversas se encuentra en los respectivos apéndices A.2.1, A.2.2, A.2.3, A.2.4. Sin embargo, valdría la pena realizar una tabla para resumir estas funciones, que se proporcionan en la tabla 3.5.

Tabla 3.5: Tabla de funciones

Componente	Función	Origen de información	Paso
Circuito	Atenuación y conexión al ADC	Conexión con la fuente de voltaje	Conexión
ESP32	Lectura y cifrado	Medición de voltaje del ADC	Medición
Decodificador	Interpretación de los valores cifrados	Comunicación serial del ESP32	Interpretación
Graficador	Graficación del voltaje recibido	Valores enviados por el decodificador	Visualización

En las fotografías de las figuras 5, 6 y 7 se pueden visualizar, respectivamente, el material necesario para armar el circuito, el dispositivo ensamblado y la demostración de funcionamiento comparado con el trazo obtenido en un osciloscopio comercial.

Figura 3.4: Material necesario para el dispositivo de medición

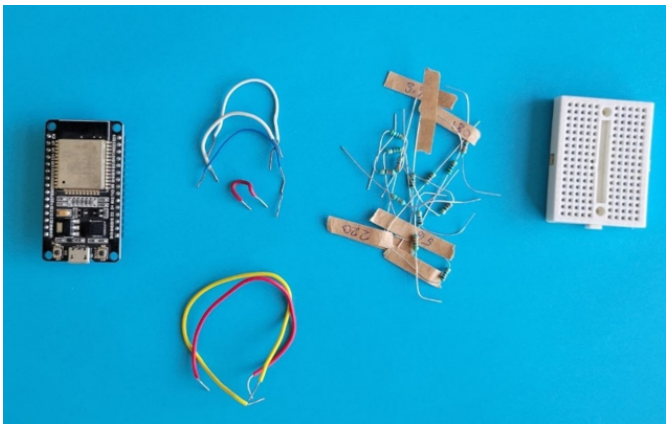


Figura 3.5: Dispositivo de medición ensamblado

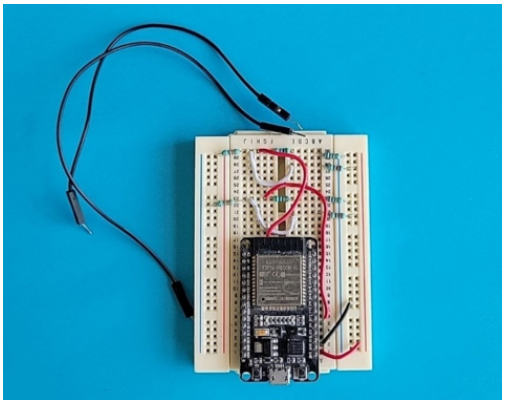
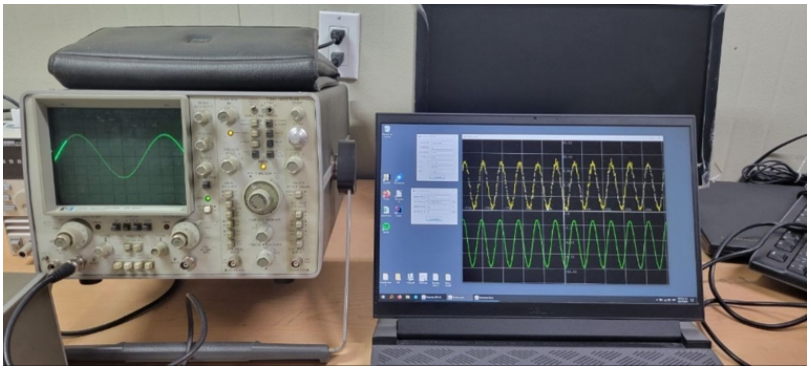


Figura 3.6: Demostración de uso del osciloscopio



Hasta este momento, no tiene nombre el osciloscopio que se desarrolló para esta tesis, por lo que se le bautizará como Osciloscopio de Casa, Modelo 1 $\rightarrow$ OCM1, con tal de darle bombo y platillo.

3.6.0.1 Especificaciones y precio

1. El ancho de banda del osciloscopio OCM1 es de 12.5 kHz en la implementación con ESP32, ya que se aprovecha el paralelismo de los dos núcleos y los dos ADC. Se puede aumentar, según la documentación oficial de Espressif, hasta aproximadamente 83.3 kHz. Para ello, se debe utilizar la interfaz I2C, DMA y el procesador de bajo consumo ULP. Para lograr estas velocidades, se requiere de más conocimiento por parte del usuario, en concreto, configuraciones que no se pueden lograr por medio del Arduino IDE.

2. El número de canales que uno puede utilizar con el OCM1 es completamente dependiente del número de ESP32 que se encuentren conectadas a la computadora. De haber una sola ESP32, será un solo canal, de haber dos conectadas, serán dos canales. El número de canales que se contempla en esta propuesta como ideal para el usuario es de dos, sin embargo, por la naturaleza de código abierto, cualquier usuario que llegara a necesitar más podría fácilmente modificar esto.

3. El precio de ambas propuestas de osciloscopio es difícil de definir, ya que dependerá mucho de si el usuario ya cuenta con una tarjeta ESP32 o Arduino Nano desde antes de comenzar a trabajar. En el caso del OCM1, se requiere de una tarjeta ESP32, 8 resistores, una tarjeta para prototipos, comúnmente llamada protoboard pequeña y alambre, en caso de usar dos canales en la misma tarjeta protoboard, se recomienda usar una grande.

Cotizando los precios en la tienda SandoRobotics para el osciloscopio con dos canales:

ESP32x2	$160x2	=$320
4 tipos de resistores	$5 x 4	=$20
Tarjeta protoboard	$35	=$35
Suma		=$375

Pensando en un estudiante que ya tenga una ESP32, por ejemplo, para otros proyectos de clase:

ESP32x2	$160	=$160
4 tipos de resistores	$5 x 4	=$20
Tarjeta protoboard	$35	=$35
Suma		=$215

Se puede ver que el precio estará alrededor de los $200 en caso de ya contar con un poco de material previo, por ejemplo, una tarjeta protoboard. A su vez, por la naturaleza de esta propuesta, es fácil compartir material con compañeros, con excepción de la tarjeta ESP32, potencialmente reduciendo más el precio del prototipo.

3.7 Resumen

Éste es indudablemente el capítulo más largo del documento por lo que el resumen se hará en partes.

La primera etapa del diseño constó en investigar y entender cómo funciona un osciloscopio digital. A partir de esta investigación, se propuso el esquema de flujo de información mostrada en la figura 1.

La segunda etapa constó en dar cabalidad a este esquema inicial. Se propuso un diseño muy general y abstracto, propuesta que se hizo con el propósito de poder analizar en el papel los canales de comunicación necesarios, así como las transformaciones a realizar en la información obtenida durante el proceso. Por medio de este análisis, se encontró que habría dos herramientas necesarias adicionales a las identificadas durante el análisis preliminar del diseño del osciloscopio.

La tercera etapa fue el análisis de las diferentes alternativas para la implementación concreta del osciloscopio. Ya que el diseño de este proyecto se realizó de manera segmentada, hubo varias opciones a elegir para cada uno de estos segmentos de diseño. Por medio de este proceso de selección, se obtuvieron las herramientas con las que se haría la implementación final.

La cuarta etapa constó en dar una implementación concreta a la primera aproximación abstracta, haciendo uso de las herramientas seleccionadas.

1. Se propuso un esquema general y posteriormente se diseñaron cada una de las partes, en el mismo orden en que la información transita desde el fenómeno eléctrico hasta la ventana de visualización.

2. Se diseñó un dispositivo de medición compuesto de un circuito de acondicionamiento, cuya demostración se muestra en el apéndice A.2.1 y una tarjeta de desarrollo, diseño detallado en A.2.2, por medio de la cual se realizarían las mediciones y la comunicación serial con la computadora donde se graficarían los valores. Igualmente se detallará el protocolo de cifrado.

3. Se diseñó un programa por medio del cual se pudieran obtener estos valores desde el puerto serial e interpretar como valores de voltaje. El uso de este programa se detallará en el apéndice A.2.3.

4. Se diseñó un mecanismo de intercomunicación entre la aplicación de procesamiento e interpretación con aplicaciones exteriores, detallada en el apéndice, facilitando así el diseño de la aplicación graficadora como un programa independiente, además de abrir la posibilidad de programar generadores de señales simuladas y no necesariamente leídas desde el puerto serial.

5. Se diseñó una aplicación de interfaz gráfica, proceso detallado en el apéndice, por medio de la cual se podría realizar la visualización de las señales obtenidas y procesadas en la aplicación de procesamiento e interpretación. Este programa facilitaría al usuario todas las manipulaciones comunes, durante la visualización de señales eléctricas, tal como lo hacen los osciloscopios comerciales.

4

CONCLUSIONES Y TRABAJO FUTURO

4.1 Conclusiones

Este proyecto se diseñó con el objetivo de ser una herramienta de medición de voltaje para el uso de los estudiantes. A pesar de construirlo a partir de piezas bastante sencillas, los resultados obtenidos son suficientemente satisfactorios como para considerar que se cumplió el objetivo del desarrollo de un osciloscopio económico para uso en casa.

4.2 Trabajo futuro

Se les conmina a todos los usuarios de este proyecto a que le realicen las modificaciones que consideren pertinentes. Algunos ejemplos sencillos serían:

- Agregar diodos zener como medida de seguridad adicional
- Cambio de resistores a valores con menor/mayor atenuación
- Uso de condensadores como filtros

A los usuarios más avanzados con conocimiento de programación se les invita a proponer soluciones tales como:

- Ajuste de los voltajes leídos con algún referente de calibración
- Uso de las capacidades de la ESP32 como la DMA y la comunicación I2C para obtener mayores frecuencias de muestreo
- Desarrollo de la función de trigger o disparo, en la aplicación de visualización
- Desarrollo de algoritmos de filtrado

Por último, la adición que por limitantes personales no se pudo desarrollar, pero sería igualmente de enorme utilidad para la comunidad estudiantil, es crear una aplicación para teléfono móvil para la parte de interpretación y visualización.

El autor de este trabajo tiene la esperanza de que haya sido únicamente el primer ladrillo de un proyecto que preste más y más utilidad a los estudiantes a través del tiempo. Aquí solo se

intentó demostrar el potencial de las herramientas accesibles y reutilizables a las cuales los estudiantes ya tienen acceso.

A
APÉNDICES

A.1 Con respecto al ruido

Se contempló desde el inicio el uso de diferentes resistores para dar mayor o menor atenuación. En caso de tener una cantidad excesiva de ruido, el cambio por resistores con menor atenuación dará lecturas más fieles a los valores reales de voltaje. Esto, sin embargo, será sacrificando la seguridad que dan los resistores de mayor magnitud a los grandes voltajes y picos de tensión.

A.2 Material adicional

Como material didáctico, se encontró que el osciloscopio es suficientemente bueno como para realizar tareas en casa, así como para la reposición de algunas prácticas de laboratorio cuando no se tuviera acceso a los osciloscopios de los laboratorios de la Facultad. Sin embargo, a pesar de se puede aprovechar como herramienta para realizar mediciones del mundo real, también se consideró que podría cumplir una labor como material únicamente didáctico.

A.2.1 Material didáctico adicional

Como se mencionó durante la toma de la decisión del uso del protocolo UDP para la transferencia de datos; no importa si el origen de las señales graficadas es un dispositivo real, como las mediciones provenientes la ESP32, o si se generan en una aplicación que únicamente simule esos voltajes. Para el protocolo UDP, solo es necesario ver el valor de voltaje y el tiempo en que se leyó.

Por esto se decidió implementar y presentar también un generador de señales sinusoidales. La realidad es que no se está leyendo ninguna función de voltaje, únicamente se está generando el valor de manera digital y enviando por medio del protocolo UDP al graficador.

Esta aplicación, a pesar de solo poder utilizarse como una referencia, sí cumple con una labor didáctica: ofrece la oportunidad de visualizar una función sinusoidal, así como de manipularla en un ambiente seguro completamente virtual.

El programa generador, tal y como se presenta aquí, simula únicamente señales sinusoidales. Sin embargo, fácilmente se podría cambiar la simulación de esa sinusoidal por cualquier otro tipo de señal, diente de sierra, escalón, impulso, o cualquiera que el usuario pudiera pensar en implementar.

Esta funcionalidad sirve especialmente al momento de presentar a los estudiantes conceptos como la amplitud y el periodo de señales, así como temas que normalmente son difíciles de simular sin el material adecuado, como el concepto de ángulo de desfase.

A.2.2 Manual de usuario

En conjunto con este documento de tesis, se presenta un manual en el cual se explica el armado, la programación de la tarjeta ESP32, así como diversos ejemplos del uso del osciloscopio propuesto.

El manual se encuentra como una página completamente pública por vía digital en el siguiente sitio red: https://marroja.github.io/book/index.html.

Se espera que este manual sea suficiente para cualquier usuario de este osciloscopio, a pesar de que cuente con un conocimiento muy básico de electrónica.

Se espera que los estudiantes de la Facultad de Ingeniería puedan consultar con sus profesores cualquier tipo de dudas, sin embargo, considerando que esta herramienta fue creada para su uso público, se procuró ser lo más claro y explícito al momento de realizar alguna acción durante su uso.

A.2.3 Usuario avanzado y proceso de diseño

Se tuvo especial cuidado al momento de documentar cada uno de los pasos efectuados durante el proceso de diseño.

Conviene hacer notar que fue muy diferente lo que se tomó en cuenta durante el proceso de diseño respecto a la ingeniería de implementación. Durante el proceso de diseño, se trabajó con temas abstractos y se contemplaron diferentes formas de obtener una medición cualquiera. Durante la ingeniería de implementación, se trabajó con ejemplos concretos. En esta última fase se obvió la teoría, de no haber sido válida, ni siquiera hubiera tenido sentido intentar implementarla.

Esta última fase se centró en demostrar cómo se realizaron las implementaciones concretas, dando así un ejemplo a los usuarios que por curiosidad o necesidad tengan que realizar modificaciones en su implementación del osciloscopio.

A.3 Demostraciones de comportamiento y funcionamiento

En las siguientes subsecciones se hará una revisión suficientemente exhaustiva como para que no quede duda del funcionamiento y de los valores obtenidos dado el diseño propuesto del osciloscopio. Sin embargo, todos los pasos realizados para alcanzar estos diseños finales se encuentran documentados en el apéndice B. Se hará referencia a la subsección correspondiente.

A.3.1 Demostración de transformación de circuito de acondicionamiento

Se parte del arreglo de resistores propuesto en el capítulo de diseño. Es importante recordar que se estará trabajando con el divisor de voltaje con tres resistores; R1, R2 y RN. Por convención, el nodo donde se conectan esos tres resistores se llamará N. El principal valor de interés será el voltaje presente en el nodo N en función del voltaje aplicado al resistor RN. El voltaje en el nodo N se denotará como V_N o VN y el voltaje de excitación en la terminal será V_T (VT). Por último, el voltaje de referencia de la tarjeta de desarrollo con referencia al cual se realizarán todas las mediciones del ADC, se denotará como V_Ref, VRef. Véase la figura A.1.

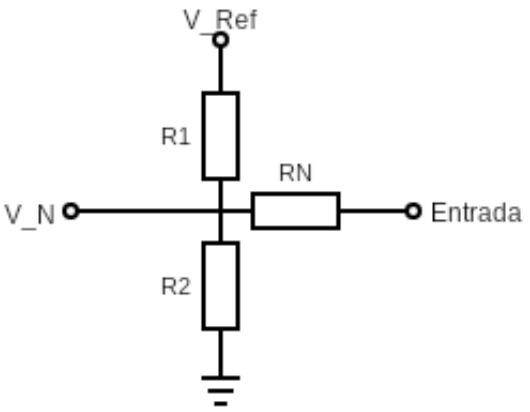


Figura A.1: Representación del circuito de acondicionamiento

El comportamiento que se esperaría obtener de este arreglo de resistores es el de una gran atenuación de la señal introducida en V_T, haciendo que se desvíe el voltaje V_N de un valor 'centrado' en el voltaje medio entre V_Ref y el voltaje de la tierra de la tarjeta de desarrollo. Primera condición:

$$V_N(0) = \frac{1}{2}V_{Ref}$$

El segundo paso involucra determinar los voltajes máximos y mínimos que deberá alcanzar el voltaje V_N en el rango de operación planeado para el dispositivo.

Por limitaciones de la gran mayoría de las tarjetas de desarrollo, es imperativo que el voltaje en el nodo N permanezca siempre por encima del voltaje de la tierra de la tarjeta para evitar la avería de la tarjeta.

El voltaje mínimo, entonces, debe de ser igual al voltaje de la tierra de la tarjeta en ese caso extremo de operación. El voltaje mínimo de excitación se denotará con V_Min (V_{Min}). El voltaje de la tierra será V_Tierra (V_{Tierra}) o V_GND(V_{GND}), que siempre se considerará igual a cero. Segunda condición:

$$V_N((V_{Min}) = V_{Tierra} = 0$$

El tercer límite que se puede establecer es que el voltaje máximo de excitación resulte en un voltaje igual al del V_Ref, ya que de superarse el voltaje V_Ref en V_N, cualquier medición llevaría a información equivocada por haberse alcanzado el voltaje de saturación del ADC. Tercera condición:

$$V_N(V_{Max}) = V_{Ref}$$

Entonces, esto da un sistema de tres ecuaciones por satisfacer para conseguir el comportamiento que se desea. (véase la figura A.2):

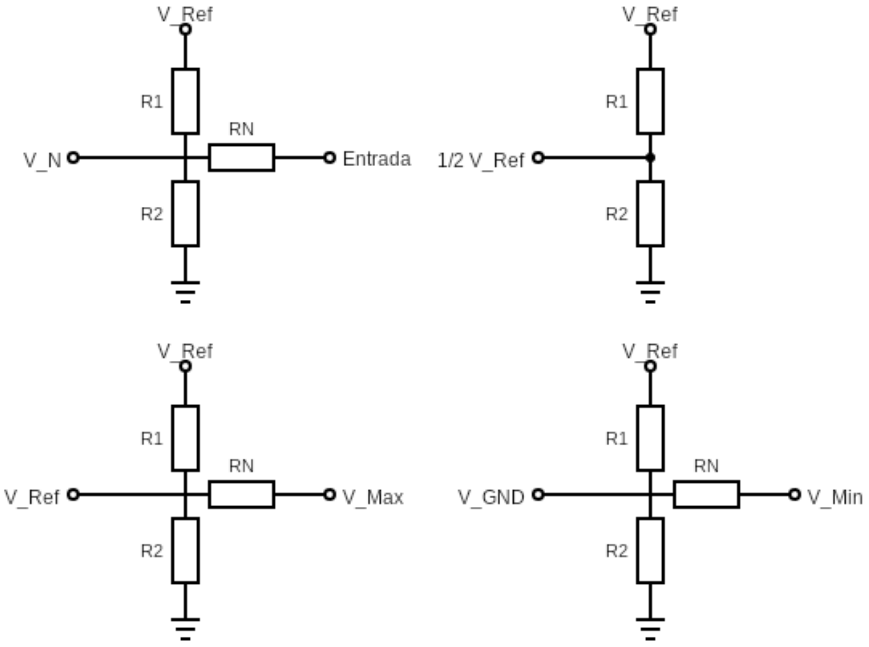


Figura A.2: Circuitos con fuentes independientes

Merece la pena agregar una última solicitud del comportamiento del circuito. Por conveniencia del usuario, el voltaje máximo de operación conviene que tenga una magnitud igual al voltaje mínimo.

$$V_{Max} = -V_{Min}$$

Primera condición (voltaje de excitación 0):

$$\frac{1}{2}V_{Ref} = V_{Ref}\frac{R_2||R_N}{R_1 + R_2||R_N}$$

De aquí no es difícil identificar la siguiente equivalencia:

$$\frac{R_2||R_N}{R_1 + R_2||R_N} = \frac{1}{2} \tag{A.1}$$

$$R_1 + R_2||R_N = 2(R_2||R_N)$$

Esto dará una manera de sustituir el resistor R_1, reduciendo incógnitas.

$$R_1 = R_2||R_N \tag{A.2}$$

Haciendo uso de fuentes independientes se obtiene la expresión:

$$V_N(V_T) = V_{Ref}\frac{R_2||R_N}{R_1 + R_2||R_N} + V_T\frac{R_1||R_2}{R_N + R_1||R_2}$$

Aplicando la segunda condición, (voltaje de excitación mínimo) se obtiene:

$$V_N(V_{Min}) = V_{Tierra}$$

$$V_N(V_{Min}) = 0$$

$$V_{Ref}\frac{R_2||R_N}{R_1+R_2||R_N}+V_{Min}\frac{R_1||R_2}{R_N+R_1||R_2}=0$$

Luego de sustituir las ecuaciones 1 y 2 en la expresión anterior, se obtiene:

$$V_{Ref}\frac{1}{2}+V_{Min}\frac{(R_2||R_N)||R_2}{R_N+(R_2||R_N)||R_2}=0$$

$$-V_{Ref}\frac{1}{2}=V_{Min}\frac{(R_2||R_N)||R_2}{R_N+(R_2||R_N)||R_2}$$

$$-V_{Ref}\frac{1}{2}=-V_{Max}\frac{(R_2||R_N)||R_2}{R_N+(R_2||R_N)||R_2}$$

$$V_{Ref}\frac{1}{2}=V_{Max}*\frac{\frac{R_2}{2}||R_N}{R_N+\frac{R_2}{2}||R_N}$$

$$V_{Max}=V_{Ref}\frac{1}{2}\frac{N+\frac{R_2}{2}||R_N}{\frac{R_2}{2}||R_N}$$

$$2\frac{V_{Max}}{V_{Ref}}=\frac{N+\frac{R_2}{2}||R_N}{\frac{R_2}{2}||R_N}$$

Dado que:

$$\frac{R_2}{2}||R_N=\frac{1}{\frac{2}{R_2}+\frac{1}{R_N}}$$

Entonces:

$$\frac{R_2}{2}||R_N=\frac{1}{\frac{2R_N+R_2}{R_NR_2}}$$

$$\frac{R_2}{2}||R_N=\frac{R_NR_2}{2R_N+R_2} \quad \textbf{(A.3)}$$

$$2\frac{V_{Max}}{V_{Ref}}=\frac{\frac{R_N(2R_N+R_2)}{2R_N+R_2}+\frac{R_NR_2}{2R_N+R_2}}{\frac{R_NR_2}{2R_N+R_2}}$$

Si al numerador y al denominador del miembro derecho de la ecuación anterior se multiplica por $(2R_N+R_2)$, se obtiene que:

$$\frac{R_N(2R_NR_2)+R_NR_2}{R_NR_2}=\frac{(2R_N+R_2)+R_2}{R_2}$$

$$\frac{R_N(2R_NR_2)+R_NR_2}{R_NR_2}=\frac{(2R_N+2R_2)}{R_2}$$

Por consiguiente:

$$2\frac{V_{Max}}{V_{Ref}}=\frac{2R_N+2R_2}{R_2}$$

$$\frac{V_{Max}}{V_{Ref}}=\frac{R_N+R_2}{R_2}$$

$$V_{Ref} = V_{Max}\frac{R_2}{R_N + R_2}$$

$$V_{Ref}(R_N + R_2) = V_{Max}R_2$$

$$R_N V_{Ref} + R_2 V_{Ref} = R_2 V_{Max}$$

$$R_N V_{Ref} = R_2 V_{Max} - R_2 V_{Ref}$$

$$R_N V_{Ref} = R_2(V_{Max} - V_{Ref})$$

$$R_2 = R_N\frac{V_{Ref}}{(V_{Max} - V_{Ref})} \tag{A.4}$$

No es necesario realizar más cálculos con el tercer requisito, ya que el cuarto requisito (simetría entre V_Max y V_Min) fue suficiente para determinar el resto de las variables.
El siguiente paso es obtener la función inversa de la obtenida a de fuentes independientes:

$$V_N(V_T) = V_{Ref}\frac{R_2||R_N}{R_1 + R_2||R_N} + V_T\frac{R_1||R_2}{R_N + R_1||R_2}$$

La función inversa satisfará:

$$V_N = V_{Ref}\frac{R_2||R_N}{R_1 + R_2||R_N} + V_T(V_N)\frac{R_1||R_2}{R_N + R_1||R_2}$$

Obteniendo la función inversa:
Se aplica la ecuación A.1:

$$V_N = V_{Ref}\frac{1}{2} + V_T(V_N)\frac{R_1||R_2}{R_N + R_1||R_2}$$

Como el valor de interés es la diferencia de voltajes entre ambas terminales del osciloscopio, se trabajará con la resta de estos voltajes. El nodo donde se anticipa una lectura positiva será V_{N_A} y el nodo donde se esperará la referencia será V_{N_B}, resultando en la siguiente igualdad:

$$V_{N_A} - V_{N_B} = V_{Ref}\frac{1}{2} + V_T(V_{N_A})\frac{R_1||R_2}{R_N + R_1||R_2} - (V_{Ref}\frac{1}{2} + V_T(V_{N_B})\frac{R_1||R_2}{R_N + R_1||R_2})$$

Simplificando los términos constantes y factorizando el factor lineal resultante, se obtiene que:

$$V_{N_A} - V_{N_B} = (V_T(V_{N_A}) - V_T(V_{N_B}))\frac{R_1||R_2}{R_N + R_1||R_2}$$

Desarrollando la expresión de los resistores paralelos se obtiene:

$$V_{N_A} - V_{N_B} = (V_T(V_{N_A}) - V_T(V_{N_B}))\frac{\frac{R_1R_2}{R_1+R_2}}{R_N + \frac{R_1R_2}{R_1+R_2}}$$

$$V_{N_A} - V_{N_B} = (V_T(V_{N_A}) - V_T(V_{N_B}))\frac{\frac{R_1R_2}{R_1+R_2}}{\frac{R_N(R_1+R_2)}{R_1+R_2} + \frac{R_1R_2}{R_1+R_2}}$$

$$V_{N_A} - V_{N_B} = (V_T(V_{N_A}) - V_T(V_{N_B}))\frac{R_1R_2}{R_N(R_1+R_2) + R_1R_2}$$

$$V_{N_A} - V_{N_B} = (V_T(V_{N_A}) - V_T(V_{N_B}))\frac{R_1R_2}{R_N(R_1+R_2) + R_1R_2}$$

$$V_{N_A} - V_{N_B} = (V_T(V_{N_A}) - V_T(V_{N_B}))\frac{R_1 R_2}{R_N R_1 + R_N R_2 + R_1 R_2}$$

Se puede ver que el comportamiento es lineal con un factor constante dependiente únicamente de los resistores R_N, R_1 y R_2.

De este punto en adelante, se ignorará el voltaje de cada terminal y se trabajarán como una diferencia de voltajes.

$$\Delta(V_N) = \Delta(V_T)\frac{R_1 R_2}{R_N R_1 + R_N R_2 + R_1 R_2}$$

A partir de esta igualdad se puede fácilmente encontrar el voltaje de las terminales de conexión a partir del voltaje de los nodos de medición de los ADC.

$$\Delta(V_T) = \Delta(V_N)\frac{R_N R_1 + R_N R_2 + R_1 R_2}{R_1 R_2} \tag{A.5}$$

Ésta será la función inversa.

A.3.2 Cálculo del valor de los resistores

En este punto se tiene que determinar una de las variables, ya sea R2 o RN a partir de consideraciones externas hasta las proporcionadas hasta el momento.

Los valores de V_Max y V_Min se determinarán de acuerdo con el intervalo de uso que se deseé tener en el osciloscopio. Para uso general, un voltaje máximo y mínimo de +50 V y -50 V es uno adecuado para la medición de casi cualquier fenómeno de laboratorio escolar, por lo que se optó por ese valor.

Considerando el valor de 50 V en magnitud, se demuestra en la sección B.1 Determinación del resistor RN, que 3300 Ω es un valor adecuado para el resistor RN. La principal consideración es el límite de corriente que puede disipar la tarjeta de desarrollo.

A partir de estos valores, se obtiene el valor de los resistores R1 y R2. Usando las ecuaciones obtenidas en A.2.1: Con base en las ecuación A.4:

$$R_2 = R_N\frac{V_Ref}{V_{Max} - V_{Ref}}$$

Se puede determinar R2:

$$R_2 = 3300\frac{3.3}{50 - 3.3}$$
$$R_2 = 233.19\Omega$$

Y con base en la ecuación A.2:

$$R_1 = 233.19||3300 = \frac{233.19 * 3300}{233.19 + 3300}$$
$$R_1 = 217.8\Omega$$

A.3.3 Diseño de la comunicación serial

Ya que se usará el protocolo de comunicación serial, se aplicará el método descrito en el capítulo de diseño. En esta sección de detallará con uso de código en el Arduino IDE.

Se realizarán las transformaciones de dos números como ejemplo, uno positivo y uno negativo. Recordando, la lectura del ADC de la tarjeta ESP32 es de 12 bits, sin embargo, ya que se estarán realizando dos lecturas de forma simultánea para efectuar una resta del voltaje del pin

positivo y el voltaje del pin de referencia, se obtienen dos números de 12 bits, lo que dará un resultado entre -4095 y 4095.

Se realizará este procedimiento con pseudocódigo. El código implementado se mostrará en el siguiente apéndice; A.2.4.

```
 Se leen los dos valores de los pines, se denominarán valA y valB la lectura positiva
 ↪  y negativa de manera respectiva.

Los valores de ejemplo a leer serán los siguientes;
Ejemplo 1: valA = 1000     valB = 100
Ejemplo 2: valA = 0        valB = 4095
```

```
Ejemplo 1

difAB = valA - valB
difAB = 1000 - 100

difAB vale entonces 900
este valor estará representado en binario
↪  de 16 bits como:

0000_0011_1000_0100

Este número no presenta ningún problema,
↪  ya que su representación es
↪  directamente en binario.
Se pueden omitir los cuatro dígitos más
↪  significativos recordando el signo
↪  del número, positivo.




Los dígitos de interés son, entonces:
0011_1000_0100
```

```
Ejemplo 2

difAB = valA - valB
difAB = 0 - 4095

difAB vale entonces -4095
este valor estará representado en binario
↪  de 16 bits como:

1111_0000_0000_0001

Este número parece un tanto extraño ya
↪  que, al ser un número negativo, se
↪  representará como complemento a uno.
↪  Aquí es claramente visible que cuatro
↪  bits más significativos siempre serán
↪  cuatro unos cuando el número sea
↪  negativo.
Se pueden omitir los cuatro dígitos mas
↪  significativos, recordando el signo
↪  del número, negativo.

Los dígitos de interés son, entonces:
0000_0000_0001
```

Cada doce bits deberán transportarse por medio de dos bytes de información a través del puerto serial. Para hacerlo, se usará esta convención:

```
Se enviarán dos bytes en el siguiente orden:

PQXXXXXX RSXXXXXX

Donde P será el bit que indique el inicio (0) de un paquete y R el indicador de final
↪  de paquete (1). Q y S deberán ser ambos iguales y señalizarán el signo del número
↪  (positivo con 0 o negativo con 1) y por último los valores marcados con XXXXXX
↪  serán los seis dígitos más significativo del valor difAB en el primer paquete y
↪  los seis menos significativos en el segundo paquete.
```

Continuando así con los ejemplos:

```
Continuando con el ejemplo 1:

0011_1000_0100

Reacomodando
001110_000100

Se representará en los dos bytes:

00_    Inicio de paquete
00_001110
10_    Final de paquete
10_000100

Estos bytes se escribirán en el puerto
↪  serial.
```

```
Continuando con el ejemplo 2:

0000_0000_0001

Reacomodando
000000_000001

Se representará en los dos bytes:

01_    Inicio de paquete
01_000000
11_    Final de paquete
11_000001

Estos bytes se escribirán en el puerto
↪  serial.
```

El código que implementa los algoritmos aquí propuestos se presenta y explica en la siguiente subsección.

A.3.4 Código implementado en la tarjeta de desarrollo (ESP32)

```
//El número de pin en que se esperará leer los valores de voltaje
#define pin_ADC_A A0
#define pin_ADC_B A1

int lectura_ADC_A = 0b0;
int lectura_ADC_B = 0b0;
int resta_ADC = 0b0;

//Estas serán las variables que se enviarán por medio del serial
byte num_codificado_primero = 0b0;
byte num_codificado_segundo = 0b0;

void setup() {
  Serial.begin(921600);
  pinMode(pin_ADC_A, INPUT);
  pinMode(pin_ADC_B, INPUT);
}

void loop() {
  ciclo:
    //Se inicializan los valores de nuestros bytes
    //Se pueden poner el etiquetado del primero byte de una vez
    num_codificado_primero = 0b10000000;
    num_codificado_segundo = 0b00000000;

    lectura_ADC_A = analogRead(pin_ADC_A);
    lectura_ADC_B = analogRead(pin_ADC_B);
    resta_ADC = lectura_ADC_A - lectura_ADC_B;

    //Recordando que resta_ADC es un número de 16 bits
    //Se trabaja con dos bytes separados
    //Por esto se tiene que recorrer el entero 8 bits
    //Así se tiene acceso a la cola de desbordamiento 1111 (ó 0000)
    //Aquí se suma el segundo bit, donde va el signo
    //Por eso se aplica una máscara lógica & 0b0100

    num_codificado_primero += ((resta_ADC >> 8) & 0b01000000);
    num_codificado_segundo += ((resta_ADC >> 8) & 0b01000000);
    //Aplicando otra máscara lógica se suman los 6 bits
    //Se tienen que recorrer 6 lugares para sumar al primer byte
    //  ssss_xxxx_xxxx_xxxx  ->   ssss_xxxx_xxxx_xxxx
    //            isxx_xxxx  ->     is_xxxx_xx

    num_codificado_primero += ((resta_ADC >> 6) & 0b00111111);
    //Se tiene que recorrer 0 lugares para sumar al primer byte
    //  ssss_xxxx_xxxx_xxxx  ->   ssss_xxxx_xxyy_yyyy
    //            isxx_xxxx  ->             isxx_xxxx
    num_codificado_segundo += (resta_ADC & 0b00111111);

    Serial.write(num_codificado_primero);
    Serial.write(num_codificado_segundo);

  goto ciclo;
  //Resulta más veloz usar este goto que esperar el "loop()"
}
```

A.3.5 Decodificador del cifrado de bytes

El proceso de decodificación consiste en regresar del valor codificado en los dos bytes a un valor entero. Posteriormente, se le aplicará la función inversa obtenida durante la fase del diseño del circuito de acondicionamiento. Ese valor será ya un valor de voltaje, el cual se enviará al graficador.

El análisis se realiza dígito por dígito, sin embargo, dado que los valores leídos serán enteros, se pueden hacer pruebas utilizando los signos de mayor y menor, en vez de realizar pruebas bit por bit.

Continuando con los ejemplos previos:

```
 Ejemplo 1

Se leen en el puerto serial los
↪  siguientes dos bytes:

00001110
10000100

O bien:

Se pueden llamar estos bytes
B1 = 00_001110
B2 = 10_000100

Se verifica que el orden de los bytes sea
↪  correcto, es decir, que B1 sea menor
↪  que 10_000_000 y que B2 sea mayor o
↪  igual que 10_000_000:

Usando la notación de C para números
↪  binarios 0b1,1,1## y 0x1,1,1## para
↪  números hexadecimales:

B1 <  0b10_000_000
B2 >= 0b10_000_000

Se crea un número en el cual se agrega el
↪  valor de estos bytes.

Num = 0b0;

Se verifica el valor del signo, para lo
↪  cual se realizan seis corrimientos a
↪  la derecha de B1 y, posteriormente,
↪  la operación de conjunción lógica (Y)
↪  con el número 00_000_001:

Signo = (B1 >> 6) & 0b1;
En este caso, Signo valdrá 0.

Si el signo es 0, no sucede nada.
```

```
 Ejemplo 2

Se leen en el puerto serial los
↪  siguientes dos bytes:

01000000
11000001

O bien:

Se pueden llamar estos bytes
B1 = 01_000000
B2 = 11_000001

Se verifica que el orden de los bytes sea
↪  correcto,  es decir, que B1 sea menor
↪  que 10_000_000 y que B2 sea mayor o
↪  igual que 10_000_000:

Usando la notación de C para números
↪  binarios 0b1,1,1## y 0x1,1,1## para
↪  números hexadecimales:

B1 <  0b10_000_000
B2 >= 0b10_000_000

Se crea un número en el cual se agrega el
↪  valor de estos bytes.

Num = 0b0;

Se verifica el valor del signo, para lo
↪  cual se realizan seis corrimientos a
↪  la derecha de B1 y, posteriormente,
↪  la operación de conjunción lógica (Y)
↪  con el número 00_000_001:

Signo = (B1 >> 6) & 0b1;
En este caso, Signo valdrá 1.
Si el signo es 1, se tiene que llenar el
↪  número de representación
↪  completamente con unos, como el
↪  complemento a 1.
```

```
Por último, se agregan los 12 bits de B1
↪ y B2 en las 12 casillas menos
↪ significativas del número de
↪ representación.

Num &= (B1 & 0b00111111) << 6
Num += (B2 & 0b00111111)

El número resultante es:
00000000_00000000_00000011_10000100
Que es la representación en binario de
↪ 900.
```

```
Num --;

Esto dará el número hexadecimal:
0xFFFF_FFFF, el cual en binario son 32
↪ unos dada la representación usual de
↪ números enteros de 32 bits.

Por último, se agregan los 12 bits de B1
↪ y B2 en las 12 casillas menos
↪ significativas del número de
↪ representación.

Num &= (B1 & 0b00111111) << 6
Num += (B2 & 0b00111111)

El número resultante es:
11111111_11111111_11110000_00000001
Que es la representación en binario de
↪ -4095
```

Un código mucho más optimizado para Java queda de la siguiente forma:

```java
    //Considerando los valores ya leídos
    int numCodificadoPrimero = primerByteLeido();
    int numCodificadoSegundo = segundoByteLeido();

    //Se extrae el signo del primer paquete de bytes
    int decSigno = (numCodificadoPrimero >> 6) & 0x0000_0001;
    //Este valor podrá ser 1 ó 0
    //Se prepara un entero de 32 bits para que éste sea el número ya interpretado
    int nuevoNum = -decSigno;
    //En caso de signo ser 0, nuevoNum será 0  (0x0000_0000)
    //En caso de signo ser 1, nuevoNum será -1 (0xFFFF_FFFF)

    //Se asegura que los 12 bits menos significativos sean 0 por medio de una máscara
    nuevoNum = nuevoNum & 0xFFFF_F000;
    //Se agregan los 6 bits menos significativos del primer byte
    nuevoNum += (numCodificadoPrimero & 0b00_111111) << 6;
    //Se agregan los 6 bits menos significativos del segundo byte
    nuevoNum += (num_codificado_segundo & 0b00_111111);

    //Se comprueba el correcto funcionamiento este método de decodificación
    System.out.println(Integer.toBinaryString(nuevoNum));
    System.out.println(nuevoNum);
```

Por último, es necesario interpretar el valor nuevoNum como un voltaje y no como un número entre -4095 y 4095. Para ello, se usa la función inversa demostrada previamente, ecuación A.5:

$$\Delta(V_T) = \Delta(V_N)\frac{R_N R_1 + R_N R_2 + R_1 R_2}{R_1 R_2}$$

El valor que representa nuevoNum es el valor de la diferencia de los valores leídos por los ADC, ADC_A y ADC_B; por lo que no representa un voltaje en sí. Es necesario obtener el valor $\Delta(V_{R_N})$ a partir de esta información.

$$ADC = \frac{V(ADC)}{V_{Ref}} ResolADC \tag{A.6}$$

La variable ResolADC siempre será el número máximo representable por el número de bits con los que cuenta el ADC.

En el caso de la ESP32, V_Ref es 3.3V y ResolADC es (2^{12} - 1); 4095. Merece la pena mencionar que este valor es el máximo en magnitud legible por el ADC, no la máxima diferencia de voltajes legibles por por esta implementación del osciloscopio. Dado que se realiza una resta, el máximo valor legible por el osciloscopio es de 4095 - 0 y el menor valor legible es 0 - 4095. Esto hace que todos los valores entre -4095 y 4095 sean válidos valores como diferencia de valores entre los ADC. Esto da un intervalo de lecturas aparente de 8090 posibles lecturas (2^{13} - 2).

Es necesario obtener el valor V(ADC) mostrado en la ecuación anterior a partir del valor leído en el ADC.

$$V(ADC) = \frac{ADCV_{Ref}}{ResolADC}$$

Sustituyendo:

$$V(ADC) = \frac{ADC * 3.3}{4095} \tag{A.7}$$

Es necesario caracterizar entonces la ecuación 5 a partir de la 7:

$$\Delta(V_N) = \frac{ADC_A V_{Ref}}{ResolADC} - \frac{ADC_B V_{Ref}}{ResolADC}$$

$$\Delta(V_N) = \frac{(ADC_A - ADC_B)V_{Ref}}{ResolADC}$$

$$\Delta(V_N) = \frac{(ADC_A - ADC_B) * 3.3}{4095}$$

Usando esta nueva ecuación y los valores de R1, R2 y RN, se puede obtener una función caracterizada completamente en la ecuación 5:

$$\Delta(V_T) = \frac{(ADC_A - ADC_B)V_{Ref}}{ResolADC} \frac{R_N R_1 + R_N R_2 + R_1 R_2}{R_1 R_2}$$

$$\Delta(V_T) = \frac{(ADC_A - ADC_B) * 3.3}{4095} \frac{3300 * 217.8 + 3300 * 233.2 + 217.8 * 233.2}{217.8 * 233.2}$$

Como nota muy importante; el valor Δ(V_T) es únicamente una diferencia de voltajes, no voltajes únicos. Es por esto que podría parecer que el dispositivo es capaz de medir voltajes más grandes de los que se establecieron durante la fase de diseño.
Los valores constantes definidos se pueden calcular en una nueva constante K tal que.

$$\Delta(V_T) = K(ADC_A - ADC_B)$$

$$K = \frac{3.3}{4095} \frac{3300 * 217.8 + 3300 * 233.2 + 217.8 * 233.2}{217.8 * 233.2}$$

$$K = 0.02435$$

Esto da la caracterización completa del sistema:

$$\Delta(V_T) = 0.02435(ADC_A - ADC_B)$$

Merece la pena realziar observaciones en los comportamientos extremos:

ADC_A	ADC_B	$ADC_A - ADC_B$	$\Delta(V_T)$
0	0	0	0
4095	0	4095	99.71325
2048	0	2048	49.8688
0	2048	-2048	-49.8688
0	4095	-4095	-99.71325
4095	4095	0	0

Se puede observar que existen dos mediciones diferentes con las que se obtiene el valor de diseño de 50 V. Esto no es un error, sino que quiere decir que tanto el ADC_A como el ADC_B están obteniendo los valores límite que pueden reportar, 50 V y -50 V. Es posible obtener un resultado de 100 V si las dos mediciones individuales de ADC_A y ADC_B son, respectivamente, 50 V y -50 V. Si el fenómeno medido se referencia a la tierra del dispositivo de medición, el valor del ADC_B rondará el valor 2048 con ligeras desviaciones por la manufactura de los resistores. El ADC_B será 0 únicamente cuando se esté referenciando el voltaje medido con un valor de -50 V, que es el límite inferior de operación.

A.3.6 Mensajero entre aplicaciones UDP o RAM

El valor determinado por la aplicación de descifrado será únicamente un valor de voltaje obtenido en un instante. Será necesario graficar este valor de voltaje en una interfaz de usuario diseñada para la graficación de señales eléctricas.
Se necesita de dos valores para graficar en cualquier plano cartesiano, y un osciloscopio no es la excepción: X y Y. En el caso del osciloscopio, se acostumbra utilizar el valor del tiempo como la X y la Y como el voltaje leído. Esto quiere decir que se deberá obtener el tiempo de cada valor de voltaje leído. Posteriormente, esa información deberá llegar desde la aplicación que manipuló el lector serial para obtener las lecturas de la tarjeta de desarrollo a una aplicación donde se realice esta graficación. Se optó por usar un puerto UDP para realizar la comunicación entre ambos programas: Decodificador y Graficador.
Implementaciones posteriores podrían cambiar este método de comunicación por alternativas tales como mapas de memoria, implementados como MappedByteBuffer en Java.
El código de la implementación de esta funcionalidad se encuentra en el documento de anexos en la sección B2 Código Mensajero UDP.

A.3.7 Graficador

La aplicación de graficación es aquella en la que se visualizarán todas las señales y, por tanto,se espera que sea la más utilizada y manipulada por el usuario. La aplicación está diseñada de forma tal que se pueda manipular con facilidad aprovechando muchas de las ventajas que ofrecen los osciloscopios digitales modernos y la interfaz tan familiar con la que cuentan las computadoras personales.

La aplicación se asemeja mucho visualmente a la pantalla de graficación con la que cuentan los osciloscopios de cualquier laboratorio, véase la figura 3.3 en el capítulo 3 en la sección 3.4.5. Diseño Final. Por medio de esta ventana, se podrán visualizar las señales leídas y manipular los valores graficados. Una lista no exhaustiva de estas propiedades es:

- Ampliación del eje X (zoom en el tiempo)
- Ampliación del eje Y (zoom de la amplitud)
- Pausa de la graficación
- Gradilla de referencia de voltaje y tiempo
- Graficación de dos canales para comparación
- Desplazamiento eje X o el Y.

La implementación de todas estas funcionalidades, así como el funcionamiento en general, se puede encontrar en el documento de anexos en la sección B.3 Ejemplo código graficador.

Algo importante a tener en mente sobre el graficador es que es únicamente una propuesta de aplicación y no una implementación que deba usarse de forma obligatoria. Puede sustituirse esta aplicación por otro graficador.

La implementación de este graficador utiliza dos arreglos de mil valores representando mil duplas formadas por un valor de voltaje representado con un número de valor flotante doble (double) y un entero de 64 bits (long) que se sobrescriben para siempre mantener las lecturas más recientes vigentes. Cada dupla representa una medición: un voltaje, correspondiente a la lectura del ADC, y un tiempo en nanosegundos, que es el tiempo en el que se recibió en el puerto serial).

Por convención, el tiempo, eje X, se mide de izquierda a derecha, lo que significa que las lecturas más viejas se deslizarán a la izquierda hasta quedar fuera del área graficada por la aplicación. Los valores de voltaje generalmente se grafican por encima o por debajo de una línea de referencia, la cual normalmente está calibrada al voltaje de tierra.

Por último, se puede graficar una línea vertical en el centro o en alguno de los lados de la graficación para usarse como referencia al momento de medir valores en el eje Y, voltaje.

Una gráfica ilustrativa de este comportamiento es la siguiente: que se muestra en la figura A.3.

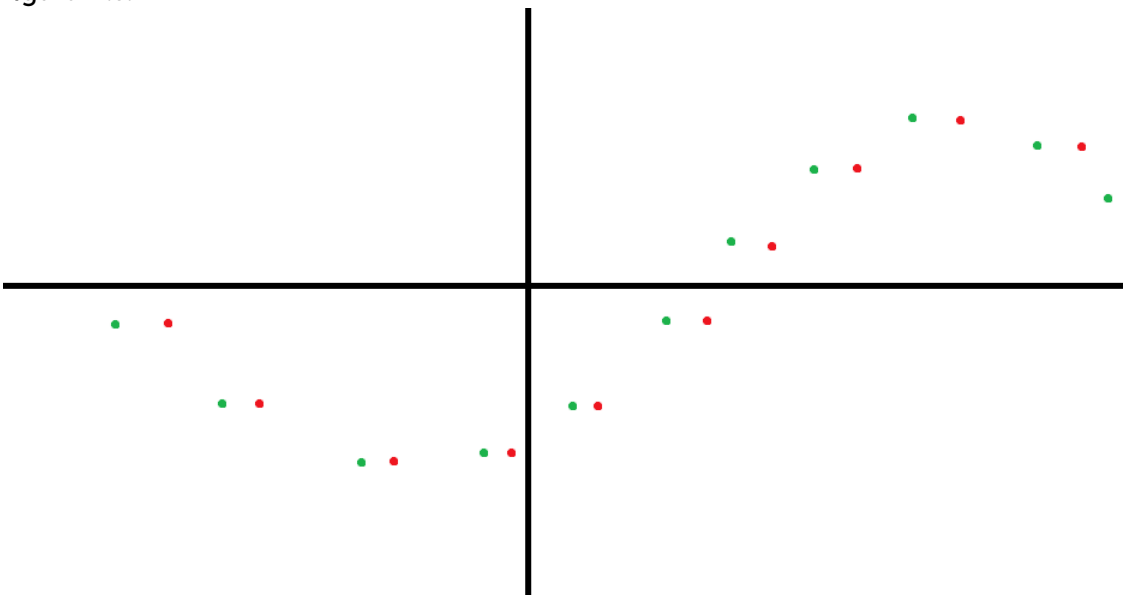


Figura A.3: Ejemplo de gráfica ilustrativa del funcionamiento del osciloscopio

Los puntos dibujados en verde son la gráfica de los valores en un tiempo dado y los puntos dibujados en rojo son los mismos valores unos pocos microsegundos después.

Se puede observar en el extremo derecho que existe un nuevo punto verde, lo que quiere decir que una nueva lectura fue agregada al arreglo de valores. De continuar agregando valores al arreglo, se perdería la visibilidad de los puntos trazados en el extremo izquierdo.

Una funcionalidad que se suele agregar en los osciloscopios modernos, es conectar los puntos graficados con segmentos de recta para hacer más visibles las señales eléctricas y no solo las lecturas individuales en el tiempo. Realizando esta conexión, se obtiene una ilustración como la que se puede observar en la figura A.4.

Figura A.4: Gráfica en la que los puntos trazados están unidos por segmentos de recta

Se puede observar que de acuerdo con el número de lecturas realizadas, se obtendrán mejores o peores representaciones de la misma señal a través del tiempo.

Es indispensable que se borre constantemente el fondo de la gráfica, para mantener una única gráfica dibujada a la vez.

Por último, se implementaron las transformaciones necesarias para realizar la ampliación y traslación de las medidas obtenidas con referencia a los ejes X e Y.

La última funcionalidad agregada en el osciloscopio fue el modo de graficación de Lissajous, en el cual se realiza el trazo de una señal con respecto a la otra. Para ello, en el eje X se representa el valor del voltaje de uno de los canales de lectura mientras que en el eje Y se representa el del otro canal. Con este modo de graficación, es más fácil apreciar la relación entre la fase y la amplitud de las señales en cada canal.

Para el correcto funcionamiento de este método, fue necesario encontrar una equivalencia de los tiempos de las lecturas de ambos arreglos.

Aquí se realizará una pequeña muestra de esto:
Arreglo 1:

V	1.0	1.1	1.2	1.3	1.2	1.1	1.0	0.9	0.8	0.7
T	131	142	151	160	169	180	189	201	212	221

Arreglo 2:

V	1.2	1.1	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3
T	130	143	152	160	171	180	191	202	209	220

Si bien existen algunos valores que coinciden en el tiempo en el que fueron leídos, hay otros valores que no coinciden. Es importante recordar que el tiempo se asigna en el momento en el que se reciben los valores del puerto serial, no en intervalos fijos. Es por esto que se tiene que realizar una equivalencia entre las medidas del arreglo de un canal con el arreglo del otro.

En la gráfica de la figura A.5 se muestra un ejemplo como se efectúa el proceso anterior; las lecturas del segundo canal se muestran en azul y las lecturas del primer canal en rojo, las cuales se toman como referencia de tiempo. Se puede ver una línea gris que conecta la lectura roja a un punto púrpura que coincide con el voltaje derivado de la interpolación lineal. Son los valores de voltaje de los puntos púrpuras los que se utilizan para realizar las gráficas

de Lissajous. Los valores obtenidos no serán exactamente los valores de la señal, pero para frecuencias bajas, la aproximación será suficientemente fiel para uso demostrativo.

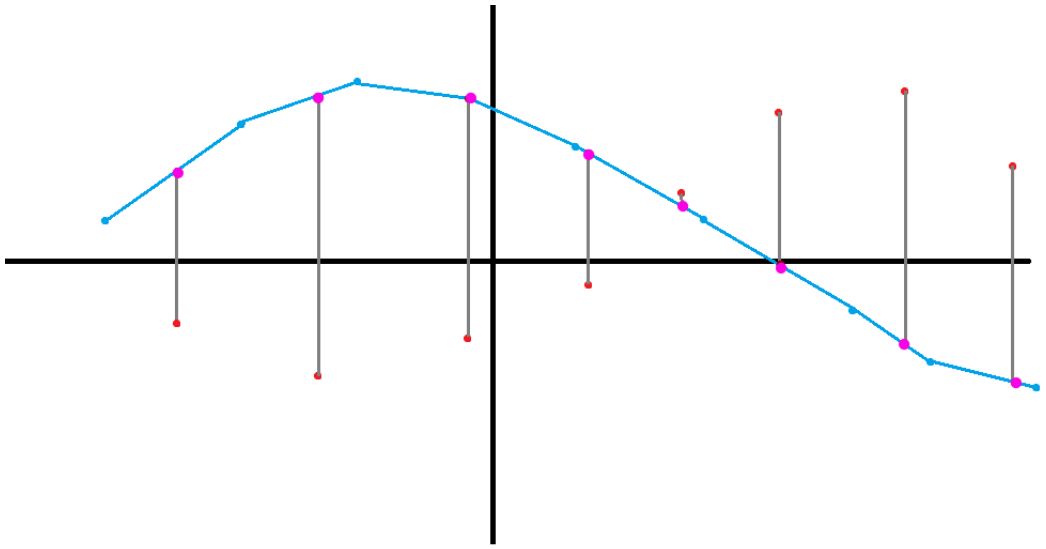


Figura A.5: Gráfica explicativa del proceso de interpolación lineal

B

EJEMPLO DE IMPLEMENTACIÓN

B.1 Determinación del resistor RN

La decisión que se consideró más prudente fue determinar el resistor N a partir del voltaje máximo que se aplicará en la terminal T. Esto es porque, a pesar de que se tenga una buena idea de cuánto es el voltaje máximo que se le va a aplicar a la tarjeta de desarrollo, es indispensable agregar un parámetro de seguridad adicional. En especial, con las mediciones en el mundo real, es muy común que el componente de tierra no sea el mismo para la computadora donde se ejecute el osciloscopio y, por extensión, en la ESP32 utilizada, y el del fenómeno que se está intentando medir.

Asumiendo que la computadora funciona por medio de una batería, como es el caso de las computadoras portátiles, no existe ninguna garantía que la tierra de la batería sea la misma que la tierra de un generador se señales conectado a la corriente de un enchufe de pared. En realidad, dado que están desacopladas ambas fuentes de voltaje, existe garantía que de que la tierra no sea la misma.

Lo más probable es que la diferencia de voltaje entre las tierras de la batería y de la red eléctrica no sea demasiada, sin embargo, sí puede ser de unos cinco o diez voltios en condiciones no tan anómalas como apagones, generadores de emergencia, baterías de coche, etc. Es por eso que se recomienda darle un parámetro de seguridad al diseño de al menos 20%, por lo que se tendrá una sobre-atenuación que posteriormente tendrá que contemplarse.

Es indispensable garantizar la seguridad de la ESP32, eso quiere decir, el voltaje presente en el nodo N no puede ser mayor a 3.3 V ni menor a 0 V.

Se correrá riesgo de exceder los 3.3 V cuando V_T sea máximo, y por debajo de 0 V cuando V_T sea mínimo. Se puede plantear un circuito de ejemplo para determinar, entonces, la resistencia de conexión del resistor N.

Ya que R1 y R2 serán mucho más pequeños que el resistor N, se pueden considerar ambos de algún valor arbitrario. Se usarán 330 Ω, ya que además, estos resistores serán muy semejantes entre ellos y no representará un cambio sustancial en la disipación de energía en el resistor RN.

Por este medio, se obtendrá un valor propuesto para el resistor N el cual se podrá multiplicar por algún factor de seguridad y, así, posteriormente cambiar el valor de R1 y R2 siguiendo la formulación que se realizó en la demostración del valor de los resistores, donde R1 y R2 dependen de RN.

Ya que el voltaje de V_T, o voltaje de la terminal, es mucho más grande que V_Ref, voltaje de referencia de ADC, se puede considerar V_Ref como un voltaje nulo. Entonces, el voltaje en N queda caracterizado como el análisis realizado para fuentes linealmente independientes cuando V_Ref se apaga:

$$V_N = V_T \frac{R_1||R_2}{R_N + R_1||R_2}$$

Ahora se necesitará un valor propuesto de V_T. Esta primera propuesta tendrá que ser el valor de voltaje máximo que se anticipa encontrar en los escenarios de lectura. Como ejemplo y únicamente para ser congruente con los valores propuestos en la parte del armado, se usará un valor de 50 V. Esto quiere decir que el osciloscopio será capaz de medir valores de voltaje de +50 V y -50 V con respecto a su tierra de manera segura y bastante exacta.
Se sustituyen los valores de los 330 Ω:

$$3.3 = 50 \frac{330||330}{R_N + 330||330}$$

$$3.3 = 50 \frac{165}{R_N + 165}$$

De donde se obtiene el valor de R_N:

$$3.3(R_N + 165) = 50 * 165$$

$$R_N = \frac{50 * 165}{3.3} - 165$$

$$R_N = 2335\Omega$$

Considerando ahora valores comerciales de resistores, es fácil ver que un resistor de 3300 Ω cumple con una tolerancia adicional de aproximadamente 30%. Así que el valor N se considerará de 3300 Ω:

$$R_N = 3300\Omega$$

B.2 Código Mensajero-Buzón UDP

Se necesitará de dos fragmentos diferentes para garantizar la funcionalidad del mensajero; uno será el código que envíe los valores por medio de un socket de UDP, el cual actuará en el programa de traducción o descifrado y el otro actuará en el programa de graficación.

La implementación de este código deberá permitir el envío de un valor long y un valor double, los cuales corresponderán al tiempo en el que se envía la lectura y el voltaje que representa dicha lectura.

La implementación en Java del código que envía los valores se muestra a continuación.

```
import java.io.IOException;
import java.net.*;
import java.nio.ByteBuffer;

public class MensajeroUDP {

    DatagramSocket socketUDP;
    byte[] buffer = new byte[16];
    InetAddress direccion;
    DatagramPacket paquete;
```

```
    int puerto;

    MensajeroUDP(String host, int puerto){
        try {
            socketUDP = new DatagramSocket();
            socketUDP.setSoTimeout(0);
            direccion = InetAddress.getByName(host);
            this.puerto = puerto;
            System.out.println("Puerto en "+puerto+" generado");
        } catch (SocketException | UnknownHostException e) {
            System.err.println("Error generando el mensajero");
        }
    }

    public int enviaPaquete(double voltaje, long momentoNano){

        byte[] bytesVoltaje = doubleToBytes(voltaje);
        byte[] bytesLong = longToBytes(momentoNano);

        System.arraycopy(bytesLong, 0, buffer, 0, 8);
        System.arraycopy(bytesVoltaje, 0, buffer, 8, 8);

        paquete = new DatagramPacket(buffer, buffer.length, direccion, puerto);
        try {
            socketUDP.send(paquete);
        } catch (IOException e) {
            System.err.println("Error mandando paquete");
        }

        return 1;
    }

    public void cambiaPuerto(int puerto){
        this.puerto = puerto;
    }

//...
}
```

En el código, se define una clase llamada BuzónUDP, que recibirá los valores enviados por el "mensajero". La funcionalidad de este buzón será recibir los dos valores enviados por el mensajero; el long y el double, e interpretarlos.

El buzón queda implementado con el siguiente código:

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;

public class BuzonUDP {
    //Este clase existe únicamente para simplificar la creación del socket
    // del puerto.

    byte[] buffer = new byte[16];
    DatagramSocket socketUDP;
    DatagramPacket paquete;
```

```
    BuzonUDP(int puerto){
        try {
            socketUDP = new DatagramSocket(puerto);

        } catch (SocketException e) {
            throw new RuntimeException(e);
        }
    }

    public byte[] recibePaquete(){
        try {
            paquete = new DatagramPacket(buffer, buffer.length);
            socketUDP.receive(paquete);
        } catch (IOException e) {
            System.out.println("Fallo al recibir paquete");
            return new byte[16];
        }
        return paquete.getData();
    }
}
```

B.3 Código del graficador

La implementación de este graficador podría ser un proyecto paralelo por sí solo, es por ello que únicamente se presentará una breve muestra de la implementación y no la implementación final.

Se decidió que el mejor lenguaje de programación para implementar este graficador sería Java. A pesar de haberse discutido varias otras formas de implementar el mismo programa, Java ofrece por medio de la JVM, compatibilidad con los sistemas operativos en los que se anticipa el uso de este programa, Windows, Linux y Mac, sin importar si la arquitectura es x86, x86_64 o alguna arquitectura de ARM, arquitectura cada vez más común.

Soluciones semejantes por medio de C y C++ son muy dependientes del compilador que se use en el momento, por lo que quedaron descartados. También se contempló implementar el graficador como una aplicación de navegador (“webapp”) para así obviar igualmente los problemas de compatibilidad, sin embargo, ya que ésta será una aplicación gráficamente demandante, por la cantidad de puntos que se espera graficar y dependiente de la velocidad de procesamiento de las lecturas recibidas, se dejó de lado también. El esquema mostrado en la figura B.1 representa la arquitectura de la implementación paso a paso por realizar. Nótese que se contempla el uso de dos hilos de ejecución para buzones UDP (BUDP1 y BUDP2) esta implementación facilitaría el uso y manipulaión de dos canales en el osciloscopio.

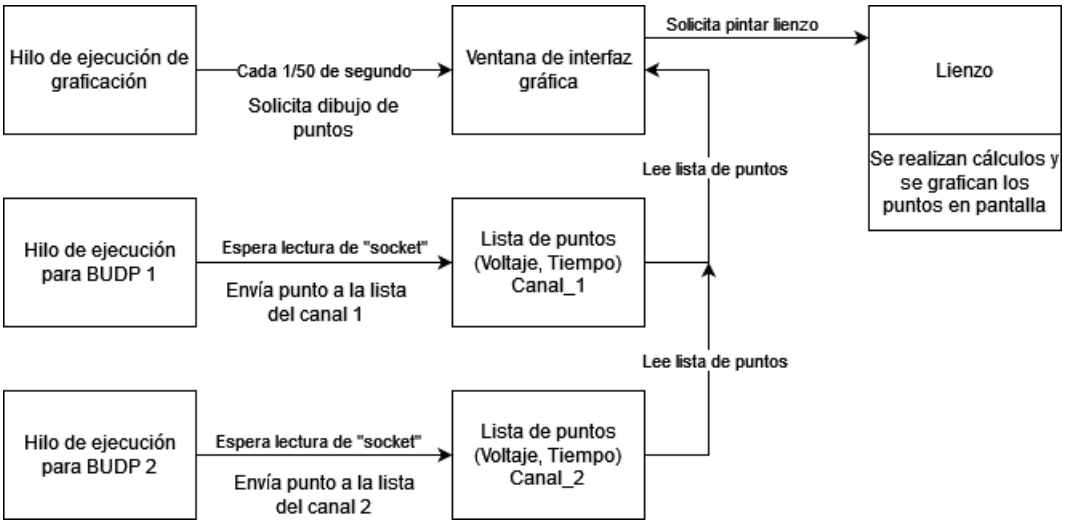


Figura B.1: Esquema de la implementación paso a paso del graficador

Existen muchas funcionalidades que se están obviando como el repintado del fondo al momento de cambiar el tamaño de la pantalla, tener la ventana principal a la espera de entradas de botones, etc.

Para ilustrar este ejemplo, se creará un pequeñísimo graficador; la mínima aplicación posible de este tipo. Se programará para un único canal, sin embargo, el ejercicio para agregar un canal adicional será trivial a partir de este.

Se parte de la funcionalidad del buzón UDP.

Aprovechando la clase programada en la sección anterior, se tendrá que conseguir que los valores leídos desde el UDP estos valores leídos serán un long (tiempo de lectura) y un double (voltaje interpretado). En el siguiente código se muestra la copia de los valores recibidos en el buzón para ser interpretados como double y como long.

El código de ejecución de este hilo es el siguiente:

```
@Override
public void run() {
    while (true) {
        //Se reciben 16 bytes en la lectura del puerto UDP
        paqueteRecibido = budp.recibePaquete();
        byte[] bytesLong = new byte[8];
        byte[] bytesDouble = new byte[8];

        System.arraycopy(paqueteRecibido, 0, bytesLong, 0, 8);
        System.arraycopy(paqueteRecibido, 8, bytesDouble, 0, 8);

        //Se realiza la conversión de los primeros 8 bytes en el Long (momentoNanos)
        //Y la conversión de los otros 8 bytes en Double (el voltaje)

        System.out.println(bytesToLong(bytesLong));
        System.out.println(bytesToDouble(bytesDouble));

        //Agregar los puntos a la ventana en que se graficarán
        //VP.Agregar(bytesToLong(bytesLong),bytesToDouble(bytesDouble))
    }
}
```

En la última línea de código se encuentra comentada la línea en la que se deberían agregar los valores leídos a la ventana de graficación.

Por último, se tendrá que implementar el código de la ventana en la que se graficarán los puntos.

```
 import javax.swing.*;

public class VentanaPrincipal extends JFrame implements Runnable{

    Lienzo lienzo;
    double[] valores;
    long[] tiempos;
    final int tamañoListaPuntos = 1000;
    int indicePuntoNuevo = 0;

    long tiempoUltimoDibujo = 0;
    final int ANCHO = 600, ALTO = 600;

    VentanaPrincipal() {
    this.setLayout(null);
    this.setSize(ANCHO, ALTO);
    this.setResizable(false);
    this.setVisible(true);
    this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

    lienzo = new Lienzo();
    this.add(lienzo);

    valores = new double[tamañoListaPuntos];
    tiempos = new long[tamañoListaPuntos];
    }

    @Override
    public void run() {
        //Solicitaremos al lienzo que actualice cosntantemente
        //Los puntos en la pantalla
        lienzo.pintaPuntos(tiempos, valores);
    }
}



//Tendremos un lienzo, aquí se pintarán los puntos
class Lienzo extends JPanel{
    Lienzo(){
        pintaFondo();
    }
    void pintaFondo(){
        //Pintaremos el fondo solo cuando sea necesario
    }
    void pintaPuntos(long[] tiempos, double[] valores){
        //Procuraremos pintar los puntos 50 veces por segundo
    }
}
```

Ahora, se desarrolla el código de manera más detallada. Se comenzará con el código para agregar los puntos a los arreglos de tiempo y de voltaje a graficar.

```
public void agregaValor(long tiempo, double valor){
    tiempos[indicePuntoNuevo] = tiempo;
    valores[indicePuntoNuevo] = valor;

    indicePuntoNuevo++;
    indicePuntoNuevo = indicePuntoNuevo % tamañoListaPuntos;
}
```

El método que se mantendrá corriendo constantemente será el método run, por medio del cual se solicitará cada 20 milisegundos una actualización de la imagen.

```
 @Override
public void run() {
    //Cada 20 milisegundos solicitaremos que se vuelva a dibujar
    if (tiempoUltimoDibujo + 20 > System.currentTimeMillis())
        return; //Sino, nos salimos del método

    //actualizamos el tiempo
    tiempoUltimoDibujo = System.currentTimeMillis();

    //Copiamos los valores a un arreglo donde no se estará escribiendo
    //Esto evitará escritura simultanea durante el dibujo y la recepción del buzón
    long[] copiaTiempos = new long[tamañoListaPuntos];
    double[] copiaValores = new double[tamañoListaPuntos];

    System.arraycopy(tiempos, 0, copiaTiempos, 0, tamañoListaPuntos);
    System.arraycopy(valores, 0, copiaValores, 0, tamañoListaPuntos);

    lienzo.pintaPuntos(copiaTiempos, copiaValores);
    lienzo.paintComponents(lienzo.getGraphics());
}
```

El lienzo queda como se muestra en el siguiente cuadro:

```
 class Lienzo extends JPanel{

    BufferedImage biPuntos;
    BufferedImage biFondo;
    final int ANCHO = 600, ALTO = 600;
    Lienzo(){
        setSize(ANCHO,ALTO);
        //Pintamos el fondo solo cuando inicia el programa
        pintaFondo();
    }
    void pintaFondo(){
        biFondo = new BufferedImage(this.getWidth(), this.getHeight(),
                                    BufferedImage.TYPE_INT_RGB);
        Graphics2D g = (Graphics2D) biFondo.getGraphics();

        g.setColor(Color.BLACK);
        g.fillRect(0, 0, ANCHO, ALTO);

        g.setColor(Color.WHITE);
        g.drawLine(0, ALTO/2, ANCHO, ALTO/2);
```

```
        g.drawLine(ANCHO/2, 0, ANCHO/2, ALTO);
    }
    void pintaPuntos(long[] tiempos, double[] valores){

        //Creamos una nueva imagen vacía de las dimensiones del lienzo
        biPuntos = new BufferedImage(this.getWidth(), this.getHeight(),
                                            BufferedImage.TYPE_INT_ARGB);

        //Agregamos el fondo a la nueva imagen que vamos a pintar
        Graphics2D g = (Graphics2D) biPuntos.getGraphics();
        g.drawImage(biFondo, 0, 0, null);

        g.setColor(Color.GREEN);
        g.drawString(String.valueOf(System.currentTimeMillis()), 30, 30);

        //Como solo queremos obtener un comportamiento sencillo
        //No tenemos que pensar demasiado en en qué dimensiones se mostrarán
        //En los ejes X y Y, solamente queremos visualizar una señal
        long tiempoAhorita = System.nanoTime();
        long intervaloLadoALado = 100_000;

        for(int i = 0; i < tiempos.length; i++){
            g.fillRect((int) (ANCHO - ((tiempoAhorita - tiempos[i])
            / intervaloLadoALado)), (int) (ALTO/2 + valores[i]), 2,2);
        }
    }

    @Override
    public void paintComponents(Graphics g) {
        g.drawImage(biPuntos, 0, 0, null);
        g.dispose();
    }
}
```

Por último, así queda el método ”main”, a partir del cual se inicia el programa:

```
 public class Main {
    public static void main(String[] args) {
        VentanaPrincipal vp = new VentanaPrincipal();
        HiloUDP hiloUDP = new HiloUDP(8080, vp);

        hiloUDP.start();
        while(true){
            vp.run();
        }
    }
}
```

Con el uso de un generador de funciones con la selección del modo seno, y empleando el mensajero UDP, se obtiene el trazo mostrado en la figura B.2.

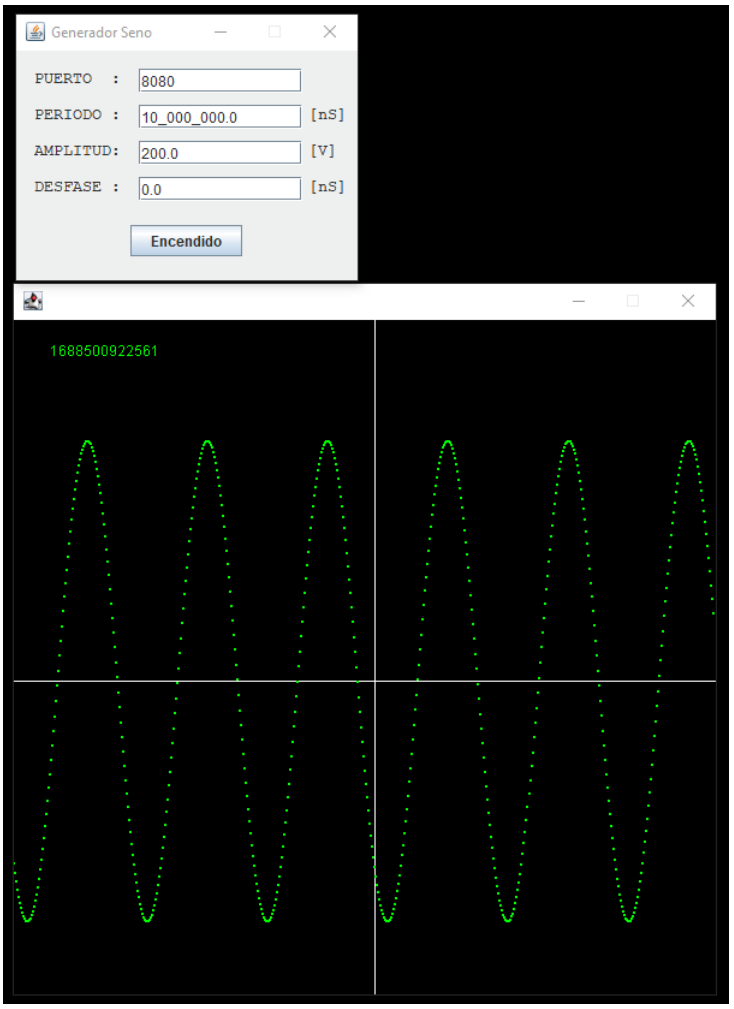


Figura B.2: Trazo de una señal sinusoidal

Es importante recordar que esta programación ha sido únicamente ilustrativa y, aunque es funcional, carece de muchas de las funcionalidades implementadas en el código final. El código de la implementación completa puede encontrarse en el repositorio: https://github.com/Marroja/graficador-multicanal-ram

REFERENCIAS

1. Krick, E. V. (2005). *Introducción a la Ingeniería y al diseño en ingeniería.* 3ª edición. Limusa. México.
2. O. Barberá & P. Valdés. (1996). *El trabajo práctico en la enseñanza de las ciencias: una revisión.* Enseñanza de las ciencias, 14 (3), pp. 365–379. Recuperado de: https://ddd.uab.cat/pub/edlc/02124521v14n3/02124521v14n3p365.pdf [2023].
3. A. P. Galván–Cardoso & E. Siado–Ramos. (2021). *Educación Tradicional: Un modelo de enseñanza centrada en el estudiante.* Cienciamatria: Revista Interdisciplinaria de Humanidades, Educación, Ciencia y Tecnología, VII (12), pp. 962–975. 2021. Recuperado de: https://doi.org/10.35381/cm.v7i12.457 [2023].
4. Minami–Koyama, Y. y Arenas–González, A. (2017). *Diseño de prácticas de laboratorio para fortalecer el aprendizaje de conceptos matemáticos en ciencias básicas.* Protocolo del proyecto UNAM–DGAPA–PAPIME PE111218.
5. Minami–Koyama, Y. y Gámez–Leal, R. (2020) *Creación de material didáctico y dispositivos para la implementación de prácticas experimentales a distancia en la División de Ciencias Básicas.* Protocolo del proyecto UNAM–DGAPA–PAPIME PE109021.
6. Minami–Koyama, Y. y Serrano–Miranda, H. G. (2020) *Diseño de dispositivos mecatrónicos para apoyo en emergencias.* Protocolo del proyecto UNAM–DGAPA–PAPIIT IT102121.
7. Aragón–Mureddu, R. y Minami–Koyama, Y. (2023). *Accessible oscilloscope for experimental learning of engineering students.* Proceedings of 16th annual International Conference of Education, Research and Innovation. España. pp. 6285 – 6292.