UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Desarrollo y Validación de una Plataforma de Hardware y Software para el Programador Inalámbrico de un Neuroestimulador del Nervio Vago - Fase II

Trabajo de graduación presentado por Miguel Alfonso Alvarez Sierra
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2021

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

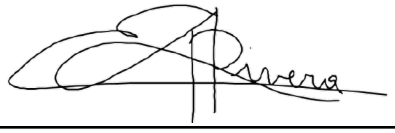
Desarrollo y Validación de una Plataforma de Hardware y Software para el Programador Inalámbrico de un Neuroestimulador del Nervio Vago - Fase II

Trabajo de graduación presentado por Miguel Alfonso Alvarez Sierra
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2021

Vo.Bo.:



(f) _____

Dr. Luis Alberto Rivera Estrada

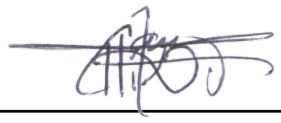
Tribunal Examinador:

(f) _____

Dr. Luis Alberto Rivera Estrada

(f) _____

MAEB. Pablo Daniel Mazariegos de la Cerda



(f) _____

MSc. Carlos Alberto Esquit Hernández

Fecha de aprobación: Guatemala, 11 de enero de 2021.

Prefacio

El motivo de este trabajo de graduación es continuar con lo realizado en la fase anterior del proyecto de Neuroestimulación del Nervio Vago (VNS) de la UVG.

Desde pequeño nunca tuve una meta fija para mí, nunca he sabido en qué quiero convertirme. Lo único que siempre he querido es ser un héroe. Y, en este mundo, una de las mejores opciones que tengo para ser un héroe, es dedicar mis esfuerzos a la salud, con enfoque en nuevas tecnologías. Cuando me enteré que había una línea de investigación de Biomédica en la universidad, no dudé un segundo en unirme. El proyecto que me fue asignado me dio un poco de miedo, pues no es lo que esperaba y no es uno de mis fuertes en la electrónica. Pero, después realizar una pequeña investigación sobre lo que era la Estimulación del Nervio Vago, entendí que era lo que yo necesitaba hacer.

Con el sistema VNS se puede ayudar a muchas personas que sufren de epilepsia resistente a fármacos. La epilepsia resistente a fármacos suele ser muy fuerte, dañando la vida diaria de las personas que la sufren, haciendo que no puedan vivir de manera plena. Además, con los años, he ganado mucho interés en la salud mental. Con nuevas investigaciones a nivel mundial, se ha descubierto que el sistema VNS también puede ser utilizado para tratar la depresión. Esto fue lo que me impulsó a dar mi mayor esfuerzo en este proyecto.

Quiero agradecer a mis padres y a mi hermana por darme todo el apoyo necesario para poder seguir adelante con mis estudios. Cuando me sentía mal porque algo no iba de acuerdo con mis planes, siempre estuvieron ahí para alegrarme y ayudarme de la manera en la que les fuera posible. No pasa un día en el que no me ría si estoy con ellos. También quiero agradecer a mis amigos de la carrera, con los que fuimos saliendo adelante año con año. Algunas veces sentíamos que las cosas no iban bien pero siempre encontrábamos la manera de seguir todos juntos. Por último quiero agradecer a mi compañero de proyecto, Gustavo Ordoñez, por trabajar conmigo en un proyecto que, en el futuro, ayudará a muchas personas en Guatemala a vivir de mejor manera. Y a mi asesor, el Dr. Luis Alberto Rivera, por dedicar tiempo a mí y a Gustavo todas las semanas, para sacar este proyecto adelante y lograr los objetivos de la mejor manera posible.

Índice

Lista de figuras

Lista de cuadros

Resumen

El propósito de este trabajo de graduación era diseñar un prototipo de módulo de programación inalámbrica para poder enviar parámetros a un módulo de estimulación (generador de pulsos, módulo a implantar); esto como parte del proyecto del Neuroestimulador del Nervio Vago, trabajado en UVG. Este prototipo buscaba eliminar las tres mayores limitaciones de la Fase I: tamaño del módulo de comunicación inalámbrica, tipo de conexión al software de programación y el uso de comunicación de únicamente una vía.

El prototipo de módulo de programación diseñado tiene tres componentes principales: software de programación, varilla programadora y módulos de comunicación/programación inalámbrica. El software de programación se utiliza para seleccionar los parámetros de estimulación de forma fácil y así poder cargarlos a la varilla programadora. La varilla programadora se encarga de enviar de forma inalámbrica, utilizando los módulos de comunicación inalámbrica, los parámetros de estimulación al generador de pulsos a implantar.

Como controlador de la varilla programadora se utilizó el módulo WiFi `NodeMCU ESP8266`. Este controlador permitió la conexión al software de programación de manera inalámbrica, por medio de WiFi, y también por medio de cable USB, en caso que la red WiFi no esté disponible o esté fallando. Para la programación inalámbrica de parámetros de estimulación, se utilizaron los módulos por radiofrecuencia (RF) `nRF24L01+`. Estos módulos funcionan como transceptores, lo que significa que pueden transmitir y recibir datos, permitiendo la comunicación de dos vías para verificación de recepción. Estos módulos tienen la capacidad de comunicarse a través de un tejido carnoso, que se simula ser la piel humana. Además, estos módulos RF son un 45 % más pequeños que los utilizados en la Fase I, ayudando a reducir el tamaño total del sistema VNS.

Por último, se diseñó un software de programación amigable y fácil de usar para poder seleccionar y cargar los parámetros de estimulación a la varilla programadora. Este software de programación se diseñó como una aplicación para que puede ser utilizado de una forma más universal, habiendo comprobado su uso en computadoras con Windows 7 o una versión más reciente.

Abstract

The purpose of this project was to design a prototype for a wireless programming module that is capable of sending stimulation parameters to the pulse generator of the Vagus Nerve Stimulation (VNS) System. This module is part of the VNS System being developed at Universidad del Valle de Guatemala. This prototype was designed to suppress the three biggest limitations of Phase I: size of the wireless module, connection type between programming wand and programming software and one-way programming.

The prototype designed consists of three major parts: programming software, programming wand and wireless programming modules. The programming software is used to select the stimulation parameters and then load them to the programming wand. The programming wand, using the wireless programming modules, sends the stimulation parameters to the pulse generator of the stimulation module.

For the controller of the programming wand, the WiFi module `NodeMCU EPS8266` was selected. This module enabled wireless communication between programming software and programming wand, via WiFi. Communication using a USB cable is also supported, if the WiFi network is failing or not available. The radio frequency (RF) modules called `nRF24L01+` were used for the wireless programming. These modules are transceivers, meaning they can work both as transmitters and receivers. These modules are 45 % smaller than the RF modules used in Phase I and they are capable of working through chicken meat, that simulates human skin.

Lastly, an easy to use and friendly programming software was created. This software was developed as an app, so that it can be used in any computer with Windows 7 or more recent versions.

CAPÍTULO 1

Introducción

La epilepsia es una de las enfermedades neurológicas más comunes en el mundo. Esta enfermedad produce convulsiones impredecibles, con la posibilidad de causar otros problemas de salud, y afecta a personas de todas las edades. Las convulsiones causadas por epilepsia pueden estar relacionadas a una lesión cerebral o una tendencia familiar, pero la mayoría de veces, no se conoce la causa. Durante una convulsión, se pierde el control de cuerpo, causando movimientos involuntarios de los músculos.

Existen cuatro tratamientos farmacológicos esenciales para tratar con la epilepsia. En América Latina, la disponibilidad de estos fármacos es escasa y en Guatemala ninguno de estos fármacos está disponible. Con poca disponibilidad de tratamientos farmacológicos y un porcentaje alto (30-40 %) de personas que sufren epilepsia resistente a fármacos, un tipo de epilepsia que no mejora con el uso de fármacos, surge la necesidad de una solución no farmacológica. La solución más utilizada actualmente es la Estimulación del Nervio Vago.

La Estimulación del Nervio Vago consiste en estimular el nervio con pulsos eléctricos para que el cerebro, que es donde se generan las convulsiones, responda de la manera esperada. Para lograr esto, se utiliza un dispositivo electrónico que se implanta por medio de cirugía en el área pectoral. Este dispositivo es el encargado de estimular el nervio vago con los pulsos eléctricos. Para definir todos los parámetros de estimulación (tiempo de estimulación, frecuencia, ancho de pulso, etc.) se utiliza un programador junto con una varilla programadora. El programador carga los parámetros a la varilla y ésta envía los parámetros de forma inalámbrica al generador de pulsos implantado.

El estimulador del nervio vago ha beneficiado a muchas personas en el mundo pues, aunque no respondan de manera positiva a los fármacos, tienen una alternativa para no sufrir las consecuencias de la epilepsia. Sin embargo, en Guatemala, no es un tratamiento accesible para todos, ya que cuesta alrededor de 20,000 USD. Debido a esto, surge la necesidad de una alternativa que sea accesible para la mayoría de guatemaltecos afectados. La alternativa es el proyecto de VNS de la universidad.

El proyecto de la Universidad del Valle de Guatemala está divido en dos módulos prin-

cipales: el módulo generador de pulsos y el módulo de programación inalámbrica. En este documento se presentará la validación y el desarrollo de una mejor alternativa del módulo de programación inalámbrica.

Para conseguir una mejor alternativa de módulo de programación inalámbrica, se diseñó una varilla programadora capaz de enviar los parámetros de forma inalámbrica a una simulación del controlador a implantar. Estos parámetros fueron enviados de forma eficiente y segura, puesto que siempre que se necesitaba enviarlos, se lograron enviar de forma rápida y sin pérdida de datos. La comunicación inalámbrica se logró a través de un tejido carnoso, que sirve para simular la piel humana. Esto se realizó debido a que es necesario enviar los parámetros de forma inalámbrica al módulo implantado debajo de la piel. Por último, se presenta una aplicación (interfaz gráfica) fácil de usar, que sirve como el programador del sistema VNS. Desde esta interfaz se pueden cargar los parámetros de estimulación a la varilla programadora, ya sea de forma inalámbrica por Wifi o con cable USB.

CAPÍTULO 2

Antecedentes

2.1. Cyberonics/LivaNova

Cyberonics es una empresa de tecnología médica la cual es experta en neuromodulación. La empresa se dedica al diseño, desarrollo y ventas de dispositivos médicos implantables que proveen terapia de neuroestimulación, como el sistema de estimulación del nervio vago (VNS por sus siglas en inglés) para el tratamiento de la epilepsia. El sistema de terapia VNS usa un dispositivo que se implanta debajo de la piel en el área pectoral que estimula el nervio vago con pulsos eléctricos y un módulo programador que envía los parámetros de estimulación al generador de pulsos de manera inalámbrica [1].

En 2015, Cyberonics y Sorin decidieron unirse para convertirse en una sola empresa, LivaNova. Estudios de LivaNova demuestran que la estimulación del nervio vago es el método alternativo, en caso de que las personas no respondan de la manera correcta al tratamiento con fármacos, más efectivo para tratar la epilepsia [2].

El dispositivo que se implanta debajo de la piel es un generador de señales implantable que entrega un patrón de estimulación preciso al nervio vago izquierdo. Uno de los generadores de pulsos más comunes es el modelo 102 de Cyberonics. Tiene un estuche de titanio de 6.9 mm $\times$ 52.2 mm $\times$ 51.6 mm y solo pesa 25 gramos. Tiene una batería que dura aproximadamente 6 años (la duración de la batería depende de los parámetros de estimulación utilizados) [3]. Para conectar el generador de pulsos 102 a el nervio vago se utiliza la sonda 302 de Cyberonics. Es un cable eléctrico bipolar que sirve para transmitir los pulsos de estimulación desde el generador hasta el nervio vago izquierdo. Los pulsos se transmiten a través de un clavija conectada al generador y se conecta a las hélices donde se conectan los electrodos [4].

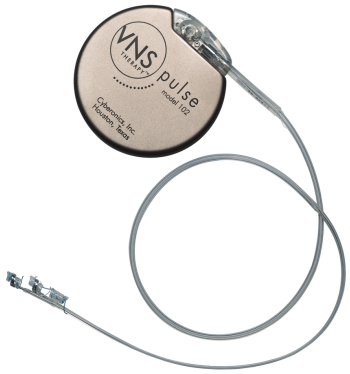


Figura 1: Generador de pulsos 102 con sonda 302 de Cyberonics [5].

Además del generador de pulsos se cuenta con software de programación, en donde se modifican y almacenan los parámetros de estimulación. También sirve para recuperar datos de telemetría. Estos parámetros se cargan en una varilla programadora para luego ser enviados de manera inalámbrica al generador de pulsos [6].

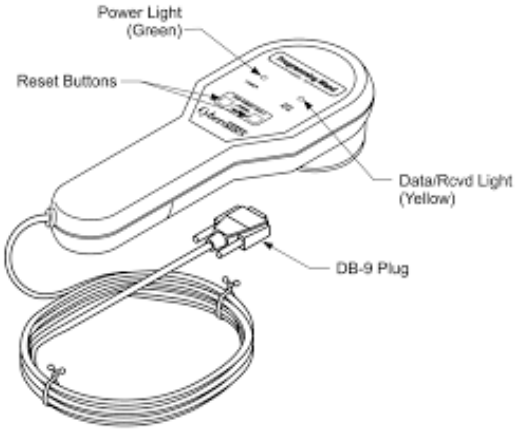


Figura 2: Varilla programadora utilizado en los VNS de Cyberonics [7].

Los Sistemas de Terapia VNS de Cyberonics incluyen la varilla programadora junto con un software, utilizado para interrogar al dispositivo y programar los parámetros de estimulación desde una tablet personal hasta el generador de pulsos por medio de señales electromagnéticas. Los parámetros de estimulación incluyen ancho de pulso, corriente de salida, frecuencia de la señal y la duración de los intervalos de la estimulación [8].

2.2. Fase I - 2019

Como parte de la línea de investigación de Biomédica de la universidad, está el proyecto de un Neuroestimulador del Nervio Vago. El proyecto está divido en dos módulos principales: módulo de estimulación (generador de pulsos) y módulo de programación (varilla programadora). Este trabajo es continuación del trabajo de graduación de Andrés Girón, 2018-2019,

quien trabajó en el módulo de programación.

En la Fase I, el prototipo final desarrollado fue un prototipo funcional de varilla programadora, capaz de enviar los parámetros al módulo de estimulación de forma inalámbrica. Para más información sobre el desarrollo del módulo de estimulación, referirse a [9].

En este prototipo final del módulo de programación inalámbrica, se utilizó el PIC16F1789, tanto para el controlador de la varilla programadora como para el controlador del módulo de estimulación. Para la comunicación inalámbrica entre varilla y módulo de estimulación, se utilizó el módulo RF SPSGRFC, que se comunica con los controladores por medio de SPI. Se logró evitar ruido de señales externas, gracias a que se estableció una frecuencia central limpia, utilizada en muchas aplicaciones médicas. Para esta fase, también se diseñó una interfaz gráfica fácil de usar para poder cargar los parámetros de estimulación deseados a la varilla programadora [10].

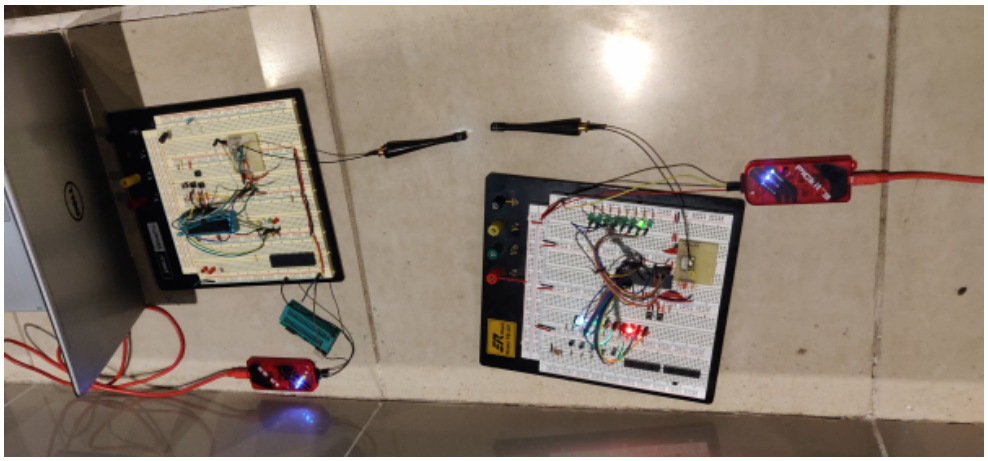


Figura 3: Prototipo final de la Fase I - Andrés Girón [10].

La mayor limitación que tiene el prototipo final de la Fase I es el tamaño del módulo RF. El módulo SPSGRFC es pequeño y sí se pudo incluir en la placa del módulo de estimulación. Sin embargo, para el correcto funcionamiento del módulo RF, se requiere de una antena. Esta antena es demasiado grande para ser implantada en el cuerpo, no cumple con los requerimientos médicos del modelo 102 de Cyberonics.

A diferencia de las varillas programadoras de nivel comercial, el prototipo final de la Fase I no es capaz de comunicarse con la interfaz gráfica (programador) de manera inalámbrica. Las de nivel comercial se pueden comunicar de forma inalámbrica con el programador, lo cual es mucho más cómodo. Además, la forma de comunicación conectada (con cable) entre la interfaz gráfica y el controlador de la varilla programadora es con un Pickit 3, algo no utilizado a nivel comercial.

CAPÍTULO 3

Justificación

Según *The Epilepsy Foundation of Greater Chicago*, la terapia de estimulación del nervio vago cuesta alrededor de 20,000 USD, incluyendo la implantación y la cirugía [11]. Esto es un precio demasiado elevado para muchas personas, sobre todo en un país en vías de desarrollo como Guatemala. Es por esto que, diseñar, y posteriormente implementar, un sistema estimulador del nervio vago, prueba ser muy útil para la población que necesita tratamiento anti-epiléptico en Guatemala.

Existen cuatro medicamentos esenciales para el tratamiento de la epilepsia. En los países de América Latina, estos medicamentos solo se encuentran en algunos países, y Guatemala no es uno de estos países [12]. En los países en los que se encuentran disponibles estos medicamentos, solo es para servicios especializados, por lo que son de difícil acceso para la persona promedio.

Además, alrededor de un 30-40 % de los casos complejos de epilepsia no responden al tratamiento con fármacos [13]. Entonces, surge la necesidad de una solución, una alternativa al tratamiento con medicamentos. El diseño y desarrollo nacional de un tratamiento de estimulador de nervio vago para mitigar la epilepsia, como el que se propone en este trabajo, podría funcionar como una solución al problema.

En Guatemala, se estima que dos de cada 100 habitantes padecen de epilepsia. HUMANA es un asociación en Guatemala que se encarga de procesos y tratamientos médicos, como el de la epilepsia. HUMANA cuenta con varios testimonios en los que se demuestra el funcionamiento del estimulador de nervio vago. Poder tratar esta enfermedad de una forma eficiente es muy importante para mejorar e incluso salvar la vida de muchos guatemaltecos [14].

Los modelos comerciales del sistema VNS cuentan con un programador y una varilla programadora para cargar los parámetros de estimulación al generador de pulsos implantado, de forma inalámbrica. Es necesario diseñar un módulo de programación inalámbrica seguro (sin errores en el envío de datos) y que sea capaz de comunicarse a través de un tejido carnoso, ya que el generador está implantado debajo de la piel.

En la fase anterior del proyecto, no se pretendía diseñar y crear un sistema VNS que fuera implantable. El proyecto tuvo sus limitaciones. En el módulo de programación inalámbrica existieron limitaciones en los módulos RF, pues estos necesitaban de antenas demasiado grandes para comunicarse, lo cual no se puede implantar en el cuerpo. Además, debido a que se utilizó un microcontrolador PIC, la conexión con la PC (interfaz gráfica) no fue la conexión más estándar. Por último, en los modelos comerciales, la varilla programadora se puede conectar con la interfaz gráfica sin necesidad de estar conectada.

Este trabajo busca corregir y superar las limitaciones de la fase anterior del proyecto. Se busca llevar más lejos lo logrado anteriormente, para poder acercarse más al objetivo final del proyecto.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Mejorar y validar la plataforma de hardware y software del programador inalámbrico desarrollado en la fase 1 del proyecto de neuroestimulación del nervio vago.

4.2. Objetivos específicos

- Diseñar e implementar una varilla programadora que envíe los parámetros de estimulación, de forma inalámbrica, al controlador a implantar.
- Implementar de forma eficiente y segura la comunicación inalámbrica entre el controlador a implantar y la varilla programadora.
- Crear una interfaz gráfica amigable y fácil de usar para definir los parámetros de estimulación de la varilla programadora.
- Diseñar el módulo de programación inalámbrica para que sea capaz de lograr la comunicación a través de una membrana que simule un tejido carnoso.

CAPÍTULO 5

Alcance

En este trabajo de graduación, se buscaba diseñar y crear un prototipo de módulo de comunicación inalámbrica del sistema VNS, que logre eliminar las limitaciones de la Fase I y llevar el proyecto más adelante. Las limitaciones que se buscaba eliminar en esta fase son:

- Tamaño del módulo receptor de información
- Tipo de conexión entre la PC y la varilla programadora

En este trabajo se redujo el tamaño del módulo receptor de parámetros para que el módulo de estimulación pueda acercarse más a un modelo comercial. Además, se mejoró la conexión entre programador (PC, interfaz gráfica) y la varilla programadora, ya sea utilizando un cable USB o por medio de comunicación por WiFi.

Como se mencionó, el programador (la interfaz gráfica) se comunica con la varilla programadora por WiFi para que la carga de parámetros sea mucho más cómoda. En caso de fallos de comunicación, también se puede conectar por medio de un cable USB. La interfaz gráfica busca si se puede conectar con el dispositivo de manera inalámbrica y, si no es posible, le pide al usuario conectar el cable USB de la varilla. Esta interfaz gráfica es fácil de usar; con solo presionar algunos botones se puede cargar los parámetros de estimulación a la varilla. La interfaz gráfica fue exportada como aplicación, para poder ser utilizada de una manera más universal.

En este trabajo no se diseñó un estuche para la varilla programadora, como la que se puede ver en al Figura 2 y no se diseñó la interfaz gráfica para ser utilizada como una aplicación de teléfono inteligente.

CAPÍTULO 6

Marco teórico

6.1. Epilepsia

La epilepsia es un trastorno neurológico (cerebral) crónico que causa que una persona sufra de trastornos convulsivos (convulsiones de manera repetida). Estas convulsiones se deben a una actividad cerebral anormal y descontrolada, provocando pérdida de atención o de la conciencia o que la persona presente comportamientos anormales. La epilepsia es causada cuando los cambios en el tejido cerebral hacen que el cerebro se encuentre excitable. Gracias a esto, el cerebro no es capaz de enviar las señales de manera normal, produciendo una convulsión [15].

Una convulsión es una descarga excesiva y corta de la actividad cerebral eléctrica, que causa que cambie el cómo siente, piensa o comporta una persona. No todas las convulsiones son causadas por epilepsia. Durante una convulsión epiléptica, el balance normal entre la estimulación e inhibición de la actividad cerebral se rompe. Aunque el término correcto para cuando se produce la actividad anormal es convulsión, muchas personas se refieren a éstas como ataques o episodios [16].

Hay varios tipos de epilepsia, que afectan a las personas de diferentes maneras. Por lo general, las convulsiones se clasifican como convulsiones focales (*partial-onset/focal-onset seizure*) o convulsiones generalizadas (*generalized onset seizures*). Una comparación ilustrativa entre las convulsiones generalizadas y las focales se puede observar en la Figura 4.

Las convulsiones focales empiezan en solo un lado del cerebro, y pueden llegar a ser muy serias y afectar la habilidad para responder al entorno. No son fáciles de identificar o sentir; hay personas que no se dan cuenta de que están sufriendo una convulsión en ese momento. Existen dos tipos principales de convulsiones focales: convulsión parcial simple y convulsión parcial compleja. Las convulsiones parciales simples son sutiles. La persona no es capaz de controlarla, pero sabe que algo está pasando. Por otro lado, las convulsiones parciales complejas perjudican la conciencia o perder ésta completamente. Las convulsiones focales pueden llegar a esparcirse por todo el cerebro, causando una convulsión tonicoclónica

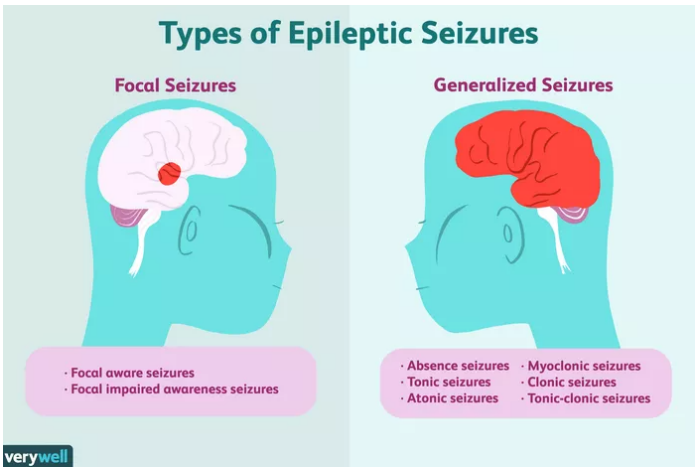


Figura 4: Convulsiones focales vs convulsiones generalizadas [17].

(*convulsive seizure*), requiriendo atención médica inmediata [18].

Las convulsiones generalizadas afectan ambos lados del cerebro. Se dividen en dos grupos principales: convulsiones de ausencia y convulsiones tonicoclónicas. Las convulsiones de ausencia, llamadas también pequeño mal, pueden causar que la persona mantenga la mirada fija en un punto lejano por varios segundos. Esto puede estar acompañado de un mascado de labios o de parpadeo rápido [19].

Las convulsiones tonicoclónicas, también conocidas como gran mal, involucran todo el cuerpo. El término tónica sirve para indicar que los músculos se ponen rígidos y el término clónica para indicar periodos en que partes del cuerpo tiemblan o se sacuden. Estas convulsiones que todo el cuerpo se sacuda y tiemble, a veces de manera violenta, involucrando también una pérdida de la conciencia. Durante una convulsiones tonicoclónica, la persona puede gritar o caer al piso, lo cual es muy peligroso pues puede sufrir un golpe fuerte en la cabeza [20].

6.1.1. Diagnóstico

Para poder diagnosticar la epilepsia es muy importante el historial clínico del paciente: contexto en el que ocurrió la convulsión, supervisión de signos, detalles de la convulsión como la fenomenología, la respuesta de la persona afectada y el estado postictal. Dependiendo de cómo sean los resultados del historial, la evaluación varía. Estos exámenes neurológicos evalúan los signos focales que pueden implicar o ubicar la patología cerebral. Para diagnosticar la epilepsia se puede usar un electroencefalograma, una tomografía computarizada o una imagen por resonancia magnética.

Un electroencefalograma (EEG) es una grabación de la actividad eléctrica cerebral. Sirve para detectar actividad anormal, como picos u ondas focales, consistentes con las convulsiones focales, u ondas pico (*spike-and-wave*) bilaterales difusas, consistentes con la epilepsia generalizada. Un EEG rutinario debe ser realizado durante tres estados diferentes de conciencia: despierto, somnoliento y dormido. Las anomalías epilépticas varían mucho entre estos tres estados de conciencia y por eso es necesario observar la actividad eléctrica cerebral en

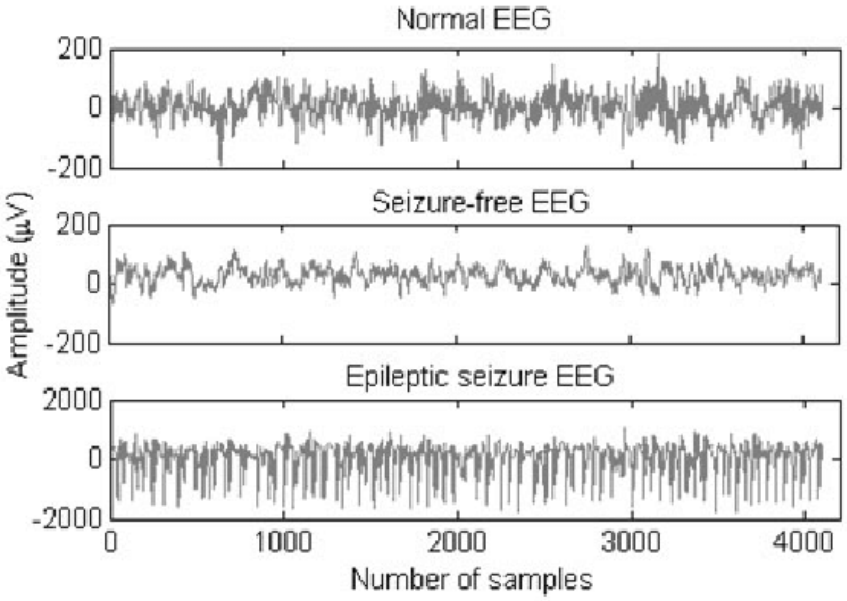


Figura 5: Comparación de la actividad eléctrica cerebral utilizando un EEG [22].

los tres.

Procedimientos activos como la hiperventilación se realizan durante un EEG para incrementar el rendimiento de la actividad epiléptica. La hiperventilación puede ayudar a causar una convulsión de ausencia. Utilizar un EEG por mucho tiempo (algunos días) puede ayudar a concluir si la convulsión fue un evento causado por epilepsia o una convulsión "común".

Las neuroimágenes, como las tomografías computarizadas (CT) y las imágenes por resonancia magnética (MRI), son muy importantes para la evaluación de personas que sufren convulsiones. Estas técnicas son especialmente sensibles a las lesiones estructurales del sistema nervioso central por lo que son muy útiles.

Una MRI tiene mayores posibilidades de mostrar anomalías en un paciente con convulsiones focales que un EEG. Una MRI también es mejor que una CT para la detección de malformaciones de la corteza cerebral y esclerosis del hipocampo por lo que se usa para evaluaciones más avanzadas [21].

6.1.2. Tratamiento

El tratamiento más común para tratar la epilepsia es el farmacológico. La mayoría de las personas que sufren epilepsia pueden utilizar fármacos anti-epilépticos (AEDs, *Anti-epileptic drugs*) como forma de tratamiento. Es la forma de tratamiento más común, ayudando a un 70-80 % de las personas a curarse completamente o a lograr que las convulsiones desaparezcan con el tiempo.

Los tipos más comunes de AEDs incluyen:

- Valproato de sodio

- Carbamazepina
- Lamotrigina
- Topiramato

De la población que queda libre de convulsiones epilépticas, un 80 % necesitará solo de una fármaco para control y un 10-15 % necesitará una combinación de dos agentes (fármaco). La AED a tomar depende del tipo de convulsiones del paciente, su edad y otros factores; se debe consultar al médico. La meta de esta terapia con fármacos es lograr eliminar/controlar las convulsiones con un solo agente tomado una o dos veces al día, sin efectos secundarios. Si no se puede controlar de buena manera, se puede incrementar la dosis o combinar con otro fármaco. Es necesario observar de manera cuidadosa al paciente pues se pueden presentar tumores o defectos metabólicos al incrementar la dosis o combinar agentes [23].

Epilepsia refractaria farmacológica/Epilepsia intratable

Los fármacos anti-epilépticos son fundamentales para el tratamiento de la epilepsia. Sin embargo, existen pacientes que no logran controlar las convulsiones con los primeros dos fármacos, incluyendo combinaciones de éstos dentro de 1-2 años de empezar el tratamiento. Los pacientes que no logran controlar las convulsiones epilépticas de esta manera tienen muy pocas probabilidades de lograr la remisión. Estos pacientes, un 30-40 %, que siguen sufriendo convulsiones después de muchas pruebas con AEDs, se consideran pacientes con epilepsia refractaria farmacológica o epilepsia intratable [24].

La epilepsia refractaria se da cuando las crisis epilépticas se dan de manera muy frecuente, afectando la vida diaria de los pacientes y su salud mental y física. También es posible que los tratamientos farmacológicos no sean capaces de controlar las crisis o los efectos secundarios de estos no permiten a la persona vivir de manera plena [25].

La epilepsia intratable o epilepsia refractaria farmacológica, es decir, epilepsia resistente a fármacos, es cuando no se pueden controlar las convulsiones con las AEDs. Aún cuando la mayor parte de la población con epilepsia puede ser tratada con fármacos, un porcentaje de los afectados son resistentes a las drogas. La Liga Internacional Contra la Epilepsia (ILAE) requiere "el fallo de lograr la libertad de convulsiones con dos fármacos anti-epilépticos toleradas, seleccionados y utilizadas de manera apropiada" [26].

Persistencia de crisis	Índice de intratabilidad
Fármaco no de primera línea, sin importar su dosis	0
Fármaco de primera línea con subdosificación	1
Fármaco de primera línea con dosis adecuada	2
Fármaco de primera línea con rango sérico adecuado	3
Fármaco de primera línea con dosis máxima tolerable	4
2 o más fármacos de primera línea a dosis máxima tolerables	5
2 o más fármacos de primera línea a dosis máxima tolerable y droga de segunda línea	6

Cuadro 1: Escala de Schmidt [25].

Existen escalas, como la Escala de Schmidt 1, que sirven para catalogar a la epilepsia de dos distintas maneras: una epilepsia no tratada de manera suficiente (0-3) o refractariedad, es decir, epilepsia intratable (4-6), dependiendo de la persistencia de las crisis epilépticas basada en los fármacos utilizados [25].

Una de las opciones más comunes para tratar personas con epilepsia resistente a fármacos es con una cirugía de epilepsia. Suele ser efectiva pero es la opción que más riesgo presenta, dado a que esta es a cráneo abierto, lo que podría causar dolor, complicaciones e impactos psicológicos. Aún cuando es una de las opciones más comunes, muy pocos afectados son elegibles para la cirugía. El foco de las convulsiones tiene que estar en un lugar del cerebro en el que se pueda remover sin impacto a las funciones como lenguaje y fuerza, y en la mayoría de pacientes, el foco de convulsiones no se puede remover.

Otras de las opciones más populares son los dispositivos de neuromodulación o neuroestimulación: estimulación del nervio vago (VNS), neuroestimulación responsiva (RNS) y estimulación profunda cerebral (DBS) [27].

6.2. El Nervio Vago

Existen 12 nervios craneales. Estos nervios vienen en pares y ayudan a conectar el cerebro con otras áreas del cuerpo como la cabeza, el cuello y el torso. A los nervios craneales que envían información sensorial como detalles sobre lo que se huele, mira, siente y escucha, al cerebro, se les conoce por tener funciones sensoriales. A los nervios craneales que controlan el movimiento de varios músculos y las funciones de ciertas glándulas, se les conoce por sus funciones motoras. Algunos nervios, como el nervio vago, tienen ambas funciones, motoras y sensoriales [28].

El nervio vago, también conocido como nervio craneal X/décimo nervio craneal, es un nervio mixto, pues contiene fibras aferentes (sensoriales) y fibras eferentes (motoras). Esto quiere decir que es responsable de llevar las señales motoras a los órganos que controla y de llevar la información sensorial de estos órganos al Sistema Nervioso Central [29]. Es el nervio más largo y complejo; su nombre en inglés *vagus* indica que es un nervio errante que va desde el cerebro hasta parte del colon [30].

Las funciones sensoriales del nervio vago incluyen:

- Proveer información de sensaciones somáticas para la piel detrás de la oreja, la parte externa del canal auditivo y ciertas partes de la garganta
- Proveer información de sensaciones viscerales para la laringe, el esófago, pulmones, traquea, corazón y la mayor parte del tracto digestivo.
- Jugar un pequeño rol en la sensación de sabor cerca de la raíz de la lengua

Las funciones motoras del nervio vago incluyen:

- Estimular músculos en la faringe, laringe y el palate suave (la parte carnosa en el "techo"de la boca)

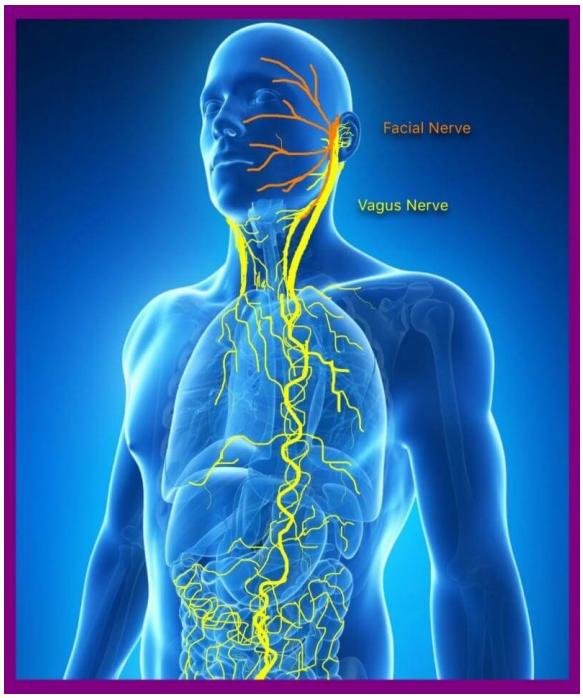


Figura 6: El nervio vago [31].

- Estimular músculos en el corazón, ayudando a disminuir el ritmo cardiaco al descansar
- Estimular contracciones involuntarias en el tracto digestivo, lo que ayuda a la comida a moverse en el tracto.

Si el nervio vago se llegara a dañar, puede existir una gran variedad de síntomas debido a la longitud del nervio. Algunos síntomas por el daño del nervio vago pueden ser: dificultad para hablar, dolor en los oídos, ritmo cardiaco inusual, nausea y vómito [32].

Pasando por varias partes del cuerpo, el nervio vago es el encargado de proveer el control primario para el sistema nervioso parasimpático, la contraparte de descanso del sistema nervioso simpático. Cuando el cuerpo no se encuentra bajo estrés, el nervio vago es el que envía los comandos para hacer que el ritmo cardiaco y respiratorio bajen y la digestión sea estimulada [33].

6.3. Estimulación del Nervio Vago (VNS)

La Estimulación del Nervio Vago (*Vagus Nerve Stimulation*, VNS) es un tipo de neuromodulación diseñado para cambiar cómo las células cerebrales trabajan, estimulando ciertas áreas del cerebro con impulsos eléctricos [34]. Fue el primer tratamiento no farmacológico desarrollado, probado y aprobado y el más utilizado para la epilepsia resistente a fármacos. Este tratamiento tiene puede tener tres efectos temporales anti-epilépticos: abortivo agudo, profiláctico agudo y profiláctico progresivo crónico.

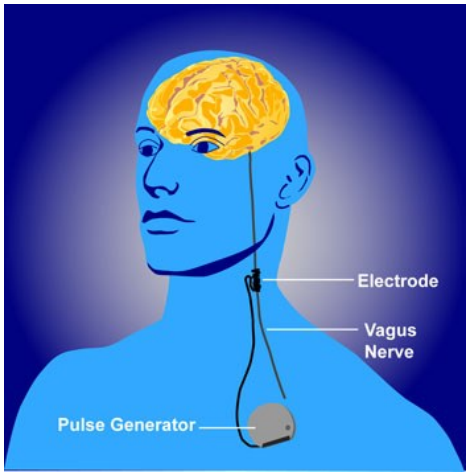


Figura 7: Estimulación del nervio vago [37].

El efecto abortivo agudo es cuando una convulsión en progreso se logra atenuar gracias a la estimulación. El profiláctico agudo logra que se tenga una menor probabilidad de sufrir convulsiones con solo unos minutos de estimulación. Por último, el efecto profiláctico progresivo crónico es cuando la frecuencia de ocurrencia de las convulsiones reduce después de semanas o meses de uso del dispositivo [35].

El uso más común de VNS involucra un sistema compuesto por dispositivo generador de pulsos eléctricos programable y su programador. El generador de pulsos se implanta de forma subcutánea en el área alta izquierda del pecho y un cable con electrodos se conecta del generador al nervio vago para estimularlo con los pulsos eléctricos.

El programador se utiliza para programar los parámetros de estimulación del generador mediante una varilla programadora. Los parámetros de estimulación son: corriente de estimulación (mA) que indica la intensidad el estímulo eléctrico, ancho de pulso (μs) que indica la duración del pulso eléctrico, frecuencia del pulso/estimulación (Hz) y el ciclo de trabajo ON/OFF (s ó min) que indica el tiempo de encendido y apagado de la estimulación. Los parámetros o ajustes iniciales por lo general se modifican más adelante para optimizar la eficacia y la tolerancia. [36]

	Visita 1	Visita 2	Visita 3	Visita 4
Corriente de estimulación (mA)	0.25	0.5	0.75	1.0
Frecuencia de estimulación (Hz)	20/30	20/30	20/30	20/30
Ancho de pulso (μs)	250/500	250/500	250/500	250/500
Tiempo de estimulación (s)	30	30	30	30
Tiempo apagado de estimulación (min)	5	5	5	5
Corriente - Modo Imán (mA)	0.5	0.75	1.0	1.25
Tiempo de estimulación - Modo Imán (s)	60	60	60	60
Ancho de pulso - Modo Imán (μs)	500	500	500	500

Cuadro 2: Parámetros programados en 4 visitas al médico [24].

Se recomienda que la estimulación comience 2 semanas después de la implantación del sistema VNS, para que la región en la cual fue implantada el dispositivo generador ten-

ga suficiente tiempo para sanar. En el Cuadro 2 se puede observar como la corriente de estimulación va aumentando en cada visita. Esto es debido a que mientras más alta sea la estimulación, o sea mayor corriente, la estimulación es más efectiva. Sin embargo, hay límites que se deben respetar para no afectar la salud del paciente [24].

6.3.1. Partes del Sistema VNS

El sistema de VNS contiene cuatro partes principales: generador (con cable), programador (tablet/computadora y software), varilla e imán.

Generador de pulsos

El generador con su respectivo cable con electrodos, es la única parte interna del sistema VNS. Este se implanta debajo de la piel en el área pectoral. El generador se encarga de enviar pulsos eléctricos al nervio vago, a través del cable que se "enrolla" en éste. En realidad, los electrodos del cable son los que se enrollan/colocan en el nervio vago, como se observa en la Figura 7. A esto es lo que se le conoce como la estimulación. El generador estimula el nervio con pulsos eléctricos de manera intermitente (30 segundos encendido y 5 minutos apagado, por ejemplo) a toda hora y todos los días.

El procedimiento de implantación se debe llevar a cabo por un neurocirujano y lleva alrededor de 45-90 minutos, mientras el paciente está bajo anestesia general. Requiere de dos pequeñas incisiones: la primera en el lado izquierdo del pecho para implantar el generador de pulsos y la segunda se hace en la parte baja del cuello para poder colocar los cables que se enrollan en el nervio vago.

El dispositivo implantado tiene una batería que puede durar desde uno hasta 15 años. Cuando la batería se agota, se remplaza el estimulador con un proceso menos invasivo que la primera cirugía [38].

Programador

El programador es una de las partes externas del sistema VNS. Este programador por lo general es una tablet con el software programador pre-instalado. El programador se conecta a la varilla programadora de manera inalámbrica.

El software programador es un software desde el cual se puede observar el estado de conexión con la varilla, el ID del paciente, la versión de software, parámetros de estimulación actuales y otras cosas. El software sirve para poder interrogar al generador con la varilla y programar nuevos parámetros [39].

Varilla programadora

La varilla programadora (*programming wand*) es otra de las partes externas del sistema VNS. Como fue mencionado anteriormente, esta varilla se comunica con el programador

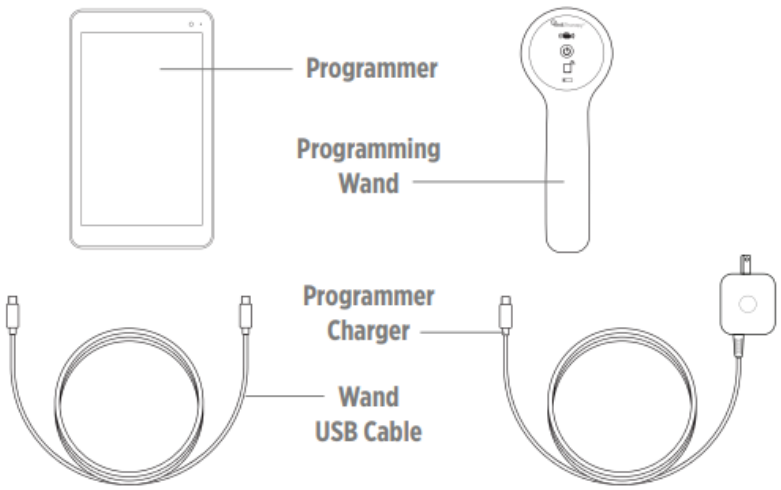


Figura 8: Programador y varilla programadora [39].

de manera inalámbrica, pero si esta comunicación es inestable o llegara a fallar, se puede conectar al programador con un cable USB.

Para interrogar al generador y para poder programar nuevos parámetros, la varilla se debe mantener a no más de una pulgada del pecho para mantener una buena comunicación con el generador de pulsos. La varilla utiliza baterías AA alcalinas o de litio para funcionar; éstas son recargables.

La varilla y el programador trabajan juntos para poder interrogar al generador para poder ajustar los parámetros de estimulación, conocer si el generador y el cable con los electrodos están funcionando de la manera esperada, ver el historial del dispositivo y exportar los reportes de las sesiones de conexión [39].

Imán

La última parte externa del sistema VNS es el Imán. Es un imán especialmente diseñado para el sistema VNS, el cual sirve para activar el generador cuando sea necesario. Es una opción para que el paciente active el sistema en caso de una convulsión.

El imán se utiliza para:

- Intentar reducir la intensidad de una convulsión o incluso llegar a abortarla.
- Deshabilitar la estimulación de manera temporal.
- Verificar que el generador se encuentre estimulando.
- Reiniciar el generador (esto se hace en conjunto con el programador y la varilla).

Este imán tiene dos estilos: reloj y "buscapersonas"[39].

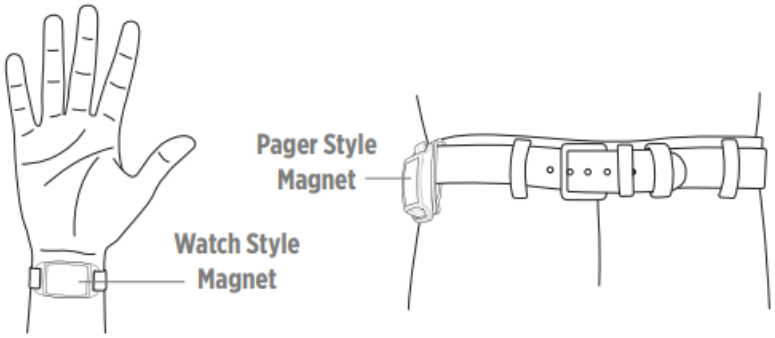


Figura 9: Estilos disponibles para el imán [39].

6.3.2. Modos de funcionamiento

El sistema VNS tiene dos modos de funcionamiento principales. Para los dispositivos más avanzados, existe un tercer modo de funcionamiento que se conoce como Autostim.

Modo Normal

En el Modo Normal, el generador inicia los intervalos de estimulación en los presets ON y OFF varias veces durante el día. Los parámetros de estimulación en este modo decididos y cargados por el médico interesado [39].

Parámetros	Valor	Unidades
Corriente de estimulación	0.25	mA
Frecuencia de estimulación	30	Hz
Ancho de pulso	500	μs
Tiempo de estimulación	30	s
Tiempo apagado de estimulación	5	min

Cuadro 3: Parámetros de estimulación usuales en el Modo Normal [24].

Modo Imán

El Modo Imán es un modo en el que la estimulación se da bajo demanda (*on-demand*, activada por el usuario) que intenta abortar o reducir la intensidad de una convulsión en proceso o a punto de iniciar.

El paciente o el médico de interés inician el Modo Imán pasando el imán sobre el generador implantado [39].

Generalmente, la corriente de estimulación es 0.25 mA mayor a la del Modo Normal (0.25 a 0.5 mA). La frecuencia de estimulación en el Modo Imán es la misma que la que se usa en el Modo Normal que, por lo general, es 30 Hz al inicio. No hay parámetro de tiempo apagado pues se regresa al Modo Normal después de un ciclo completo de apagado (5 minutos) [24].

Parámetros	Valor	Unidades
Corriente de estimulación	0.5	mA
Frecuencia de estimulación	30	Hz
Ancho de pulso	500	μs
Tiempo de estimulación	60	s
Tiempo apagado de estimulación	-	min

Cuadro 4: Parámetros de estimulación usuales en el Modo Imán [24].

Modo Autostim

El Modo Autostim (estimulación automática), también conocido como *Detect & Respond* (detectar y responder), es un modo de operación adicional en algunos modelos. Este modo se encarga de estimular el nervio vago cuando el generador detecta un rápido incremento en el ritmo cardiaco, $\geq 20\,\%$ de incremento, el cual podría estar asociado con una convulsión a punto de iniciar. Este modo funciona junto con el Modo Normal y se puede inhibir utilizando el Imán [39].

Figura 10: Icono de LivaNova para el modo Autostim/Detect & Respond [40].

6.3.3. Seguridad en las MRI

Es completamente seguro realizarse una resonancia magnética (MRI) con el dispositivo VNS implantado, siempre y cuando se sigan las instrucciones y los protocolos de seguridad de la manera correcta.

Si la persona llegara a tener una convulsión durante la MRI, el personal médico detendrá la resonancia de inmediato y aplicará los protocolos de primeros auxilios. Es por esto, que es importante comunicarle al personal médico que el paciente es propenso a convulsiones antes de iniciar la resonancia [41]

6.3.4. Sistema VNS como ayuda para la depresión

La depresión es una enfermedad mental seria que afecta negativamente a cómo una persona se siente, cómo piensa y cómo actúa. Es una de las enfermedades que más contribuye a la discapacidad global, con una relación muy fuerte con el suicidio [42].

Uno de los mayores problemas es que no todas las personas con depresión responden de manera positiva a los tratamientos antidepresivos. Para solucionar esto, los científicos y médicos empezaron a utilizar VNS para tratar con la depresión. Los científicos y médicos aún no entienden de manera completa cómo es que los impulsos eléctricos enviados al cerebro logran aliviar los síntomas de depresión. Sin embargo, los resultados por lo general son positivos y se seguirá investigando sobre este tema en el futuro [43].

6.4. Comunicaciones de datos

Las comunicaciones de datos son el intercambio de datos entre dos dispositivos a través de un medio de transmisión. La efectividad de las comunicaciones de datos depende de cuatro características fundamentales:

- Entrega (*Delivery*): el sistema debe entregar la data al destino correcto. Los datos solo deben ser recibida por el dispositivo al que se debía entregar.
- Precisión (*Accuracy*): no se debe alterar la data en la transmisión.
- Tiempo (*Timiliness*): la data se debe entregar en el tiempo establecido.
- *Jitter*: variación en el tiempo de llegada de los paquetes.

Un sistema de comunicación de datos tiene cinco componentes: mensaje, emisor, receptor, medio de transmisión y protocolo. El mensaje es la información a comunicar, el emisor es el dispositivo que envía los datos y el receptor es el que se encarga de recibirlos, el medio de transmisión es un método físico o inalámbrico por el cual el mensaje viaje y el protocolo es un set de reglas que gobierna la comunicación.

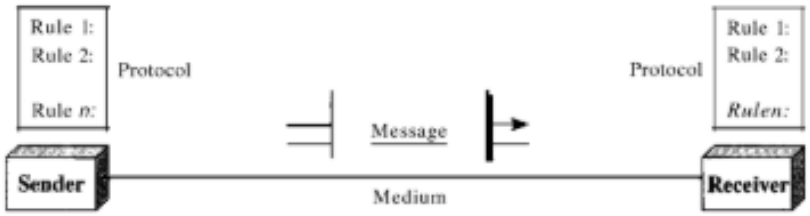


Figura 11: Comunicación de datos [44].

Existen tres formas en las que el flujo de datos entre dos dispositivos se presenta: *simplex*, *half-duplex* y *full-duplex*. *Simplex* significa que el flujo de datos es unidireccional, solo viaja

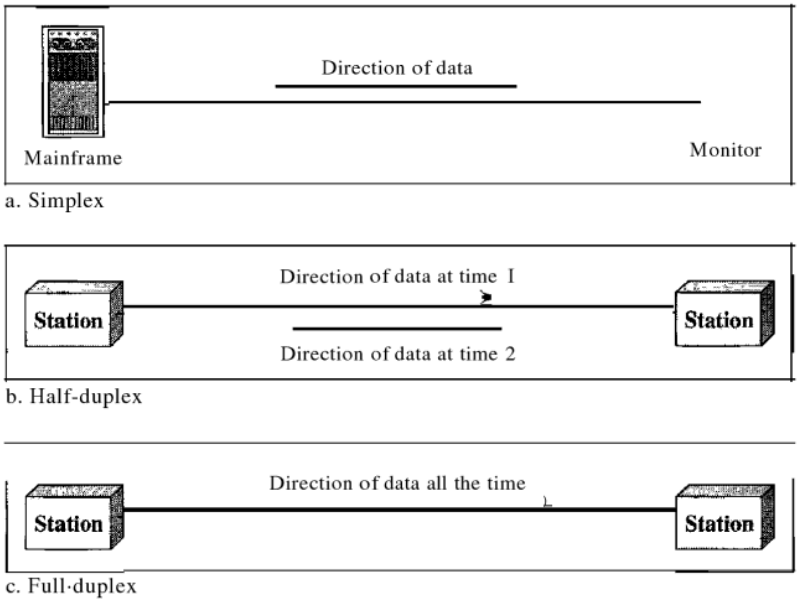


Figura 12: Formas de flujo de datos [44].

en una dirección del emisor al receptor. *Half-duplex* significa que ambos dispositivos pueden emitir y recibir los datos, pero no al mismo tiempo. *Full-duplex* es el método en el que ambos dispositivos son capaces de recibir y emitir información al mismo tiempo [44].

6.5. Protocolos de comunicación serial

Existen dos categorías principales de protocolos/interfaces de comunicación: paralela y serial. Las interfaces paralelas son capaces de transmitir varios bits de datos al mismo tiempo, utilizando 8, 16 o más cables, llamados buses de datos. Es una transmisión de 0s y 1s en grandes oleadas. Por otro lado, las interfaces seriales utilizan solo un cable, es decir, solo pueden transmitir un bit de datos a la vez. Se divide en dos categorías: serial asíncrona y serial síncrona [45].

6.5.1. Comunicación serial asíncrona

En este tipo de comunicación serial no se requiere una señal de reloj externa para transferir los datos. No requiere de muchos cables ni de pines I/O, pero necesita de un mayor

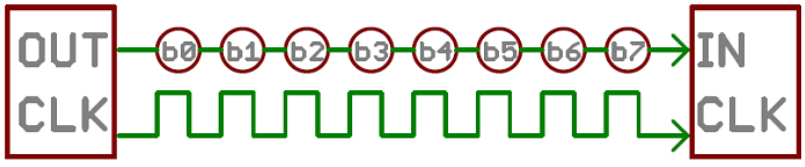


Figura 13: Ejemplo de comunicación serial [45].

esfuerzo para controlar de manera segura el envío y la recepción de los datos. Para el control y seguridad del envío y la recepción de datos, existen cuatro mecanismos/reglas que hacen que la comunicación sea robusta y se de sin errores: bits de datos, bits de sincronización, bits de paridad y tasa de baudios.

Generalmente, los datos se transmiten en paquetes (*packets*) o marcos (*frames*) de bits. Los paquetes se forman de tres partes principales: el bloque de datos, que es la parte que contiene la información que se quiere transmitir, los bits de sincronización (inicio y parada) y los bits de paridad.

Los bits de sincronización son dos o tres bits (para el (los) bit(s) de parada se pueden usar uno o dos bits), que sirven para indiciar el inicio (*start*) y el final/parada (*stop*) del paquete de datos.

Los bits de paridad son una forma de revisión de errores de bajo nivel. Se suman todos los bits del bloque de datos, si el resultado de la suma es impar (número impar de 1s), se coloca un 1 en el bit de paridad. Si el resultado de la suma es par (número par de 0s), se coloca un 0. Actualmente, no se utiliza comúnmente pues esto hace que la velocidad de transmisión disminuya; se utiliza en medios de transmisión en los que hay mucho ruido.

La tasa de baudios (*Baud Rate*) especifica el número de cambios por segundo que se dan en la línea de transmisión de datos. Aunque por lo general, la tasa de baudios se usa para referirse a la cantidad de bits por segundo que se transmiten. Con la tasa de baudios se determina el periodo al que el dispositivo receptor debe muestrear los datos. Ambos dispositivos deben estar configurados a la misma tasa de baudios [45].

UART

El bloque UART (*Universal Asynchronous Receiver/Transmitter* en inglés) es el responsable de la implementación de comunicación serial asíncrona en los dispositivos electrónicos. Funciona como un intermediario entre las interfaces paralelas y las seriales.

La R y la T de la acrónimo indican que los UARTs son los que se encargan de recibir y transmitir los datos seriales. Para la transmisión, UART crea el paquete de datos, incluyendo sus bits de sincronización (y paridad si se utilizan) y lo envía a la línea de transmisión por el pin TX. Para recibir datos, muestrea la línea de recepción, a la tasa de baudios, en el pin RX. Debe encontrar los bits de sincronización para poder escoger de manera correcta el bloque de datos [45].

6.5.2. Comunicación serial sincrónica

La comunicación serial síncrona requiere de una línea separada para la señal de reloj. La señal de reloj es una señal oscilante que le dice al receptor cuando muestrear los datos en la línea. Cuando el receptor detecta el flanco (de subida o de bajada) en la señal de reloj, empieza a leer los datos en la línea [46]. Esta señal puede ser o no ser periódica y sirve para proveer la información de sincronización (*timing*) [47].

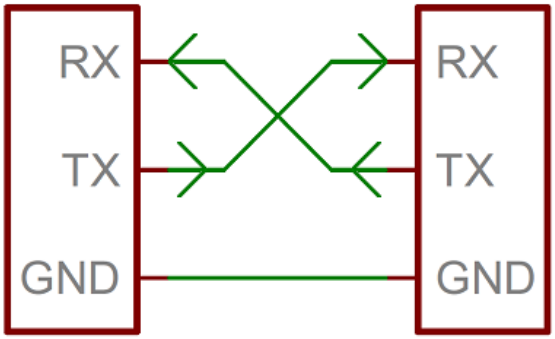


Figura 14: Conexión entre dos dispositivos con UART [45].

SPI

SPI (*Serial Peripheral Interface*) es una interfaz serial que requiere de cuatro cables: dos cables/líneas para datos y dos cables/líneas para control. Las dos líneas de datos son MOSI (*Master Out, Slave In*) y MISO (*Master In, Slave OUT*) y las dos líneas de control son SCK (para la señal de reloj) y SS (*Slave Select*) que sirve para indicar el dispositivo (esclavo) conectado al bus al que se le enviará información [47].

En esta interfaz serial existe un maestro y uno o más esclavos. El maestro es el que se encarga de generar la señal de reloj, así que necesita saber cuándo regresará información y cuántos datos serán, para generar los pulsos de reloj necesarios. Si solo es un esclavo, no es necesario usar la línea SS, disminuyendo los cables necesarios a tres [46].

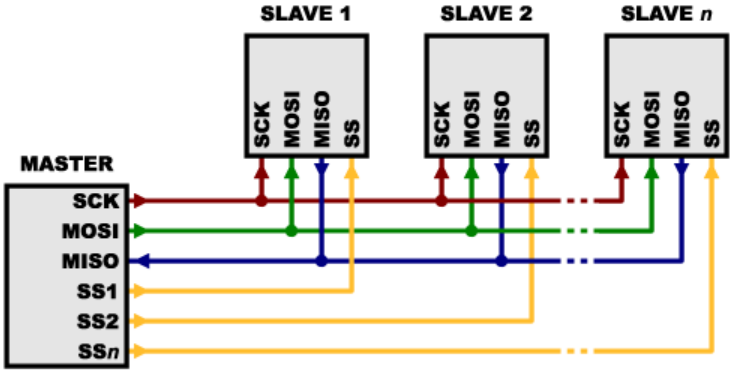


Figura 15: Ejemplo de comunicación SPI con varios esclavos [46].

I^{2}C

I^{2}C (*Inter-Integrated Circuit* en inglés) es un protocolo de comunicación serial síncrona desarrollado por Philips en los 70s. Tiene un diseño de dos cables, una línea utilizada para la transferencia de datos (SDA) y otra línea utilizada para la señal de reloj (SCL).

A diferencia de SPI, I2C no tiene selección de esclavo/chip (SS/CS), por lo que todos los dispositivos conectados al maestro deben leer el bus de datos de manera constante, revisando

si los datos van dirigidos a ellos. La transferencia de datos debe ser de 8 bits con 1 bit de reconocimiento (*acknowledge*). Luego de haber activado la transmisión, se pueden transmitir una cantidad ilimitada de bytes de datos hasta que la transmisión se desactive [47].

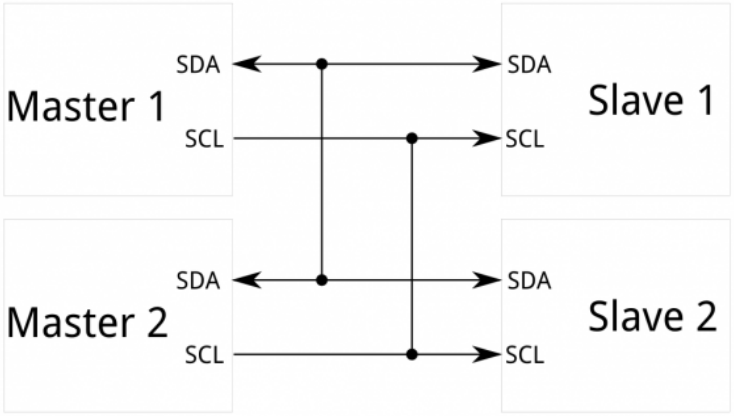


Figura 16: Ejemplo de comunicación serial con I2C [48].

6.6. Comunicación inalámbrica

La comunicación inalámbrica es un tipo de comunicación en la que no existe un medio de propagación físico como cables. Los dispositivos operan utilizando radiación electromagnética para enviar la información desde el transmisor hasta el receptor. Los únicos dispositivos físicos que hay en al comunicación inalámbrica son el receptor y el emisor y las antenas, utilizadas para lograr la comunicación inalámbrica.

Existen sistemas con antenas de apertura y con antenas con cable. Las antenas de apertura primarias se energizan por medio de la fuente y se utilizan para poder transmitir y recibir señales de manera independiente. Las antenas de apertura secundarias necesitan de otra antena como fuente para poder funcionar. Para transmitir la información, se activa una antena que genera campos electromagnéticos, propagando la señal a través de los campos de radiación [49].

Al referirse a este método de comunicación, por lo general se refiere a radiofrecuencia (RF).

6.6.1. Radiofrecuencia

Una señal de radiofrecuencia (RF) se refiere a una señal inalámbrica electromagnética utilizada como una forma de comunicación. Las ondas de radio son una forma de radiación electromagnética con frecuencias que van desde los 3 kHz hasta los 300 GHz. Estas ondas se propagan a velocidad de la luz y no necesitan un medio, como el aire, para viajar. La comunicación por radiofrecuencia se utiliza en muchas aplicaciones como transmisión televisiva, sistemas de radar, redes celulares, control remoto, entre otras.

Muchos dispositivos electrónicos utilizan radiofrecuencias debajo de 1 GHz, aún cuando la comunicación inalámbrica es asociada generalmente con 2.4 GHz. La frecuencia de 900 MHz es un rango de frecuencias ISM (*industrial, scientific and medical*) utilizado por teléfonos inalámbricos y *walkie-talkies*. Las transmisiones de radio AM (535 kHz - 1.7 MHz) y FM (87 - 108 MHz) también pertenecen al grupo sub 1GHz [50].

WiFi

Fidelidad inalámbrica (WiFi, *Wireless Fidelity*) es una tecnología de transmisión de datos inalámbricos, por medio de ondas de radio, utilizada principalmente para internet. Se utilizan dos frecuencias: 2.4 GHz y 5 GHz, basándose en el estándar IEEE 802.11.

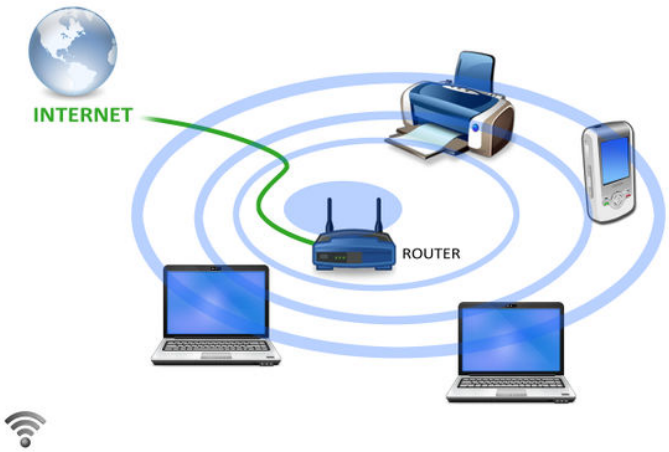


Figura 17: Conexión de dispositivos a internet vía WiFi [51].

WiFi tiene dos componentes principales: una adaptador inalámbrico en el dispositivo que se quiere conectar y un router (punto de acceso). El adaptador inalámbrico es el encargado de traducir los datos en forma de radiofrecuencias y enviarlos a través de una antena al router. El router recibe la señal y la decodifica para enviarla por medios físicos, como Ethernet, a Internet [52].

Uno de los protocolos más utilizados a nivel mundial es TCP/IP. TCP/IP es un grupo de protocolos de red, que sirven para poder transferir datos entre di positivos electrónicos (con capacidad de red) e internet. TCP viene de Protocolo de Control de Transmisión (*Transmission Control Protocol*) y permite conectarse a otro dispositivo para poder intercambiar datos de una forma fiable. IP, por sus siglas Protocolo de Internet (*Internet Protocol*) y sirve para llevar los datos a otros dispositivos en la red. Entonces TCP/IP permite un intercambio de datos fiable dentro de una red, desde el envío hasta la recepción.

CAPÍTULO 7

Prototipo 1

Para el Prototipo 1 se utilizó Arduino Uno (y Arduino Mega) para el controlador de la varilla programadora y también para el controlador que simula ser el controlador del generador de pulsos.

Como primera prueba, se conectó de manera física, un Arduino Uno a la PC con un cable USB. Utilizando comunicación serial, se enviaron números al Arduino Uno desde la terminal de Spyder. Luego, se creó una pequeña interfaz gráfica capaz de enviar comandos, de encendido y apagado de un led, al Arduino Uno. Uniendo estás dos pruebas, se diseñó una interfaz gráfica que envía números (frecuencias) al Arduino Uno, que hace resonar un buzzer a la frecuencia recibida. En la Figura 18 se puede observar un diagrama que muestra el funcionamiento de la Prueba 1.

Para la segunda prueba, se utilizaron dos Arduino Mega en simulación, pues para esta prueba no se tenían los componentes de radiofrecuencia que se necesitaban en físico. Como software de simulación se usó Proteus. Proteus Design Suite es un software de automatización de diseño electrónico, desarrollado por *Labcenter Electronics Ltd.* Sirve para diseñar esquemas electrónicos, programar el software, simular circuitos y diseñar placas impresas. Para este prototipo, Proteus se utilizó para diseñar el esquemático y simular los códigos creados.

En esta prueba el Arduino Mega de la varilla programadora recibe comandos/parámetros de estimulación desde Putty (y posteriormente desde Spyder) para luego enviarlos por medio de los módulos RF de Proteus al otro Arduino Mega. Este Arduino que simula ser el controlador del generador de pulsos, recibe los parámetros y varía las características de un pin PWM de acuerdo a estos. Para observar los cambios en el pin, se utilizó el osciloscopio virtual de Proteus. Además se enviaron los parámetros a la terminal virtual de Proteus para verificar la recepción. En la Figura 19 se muestra un diagrama que ilustra el funcionamiento de esta prueba.

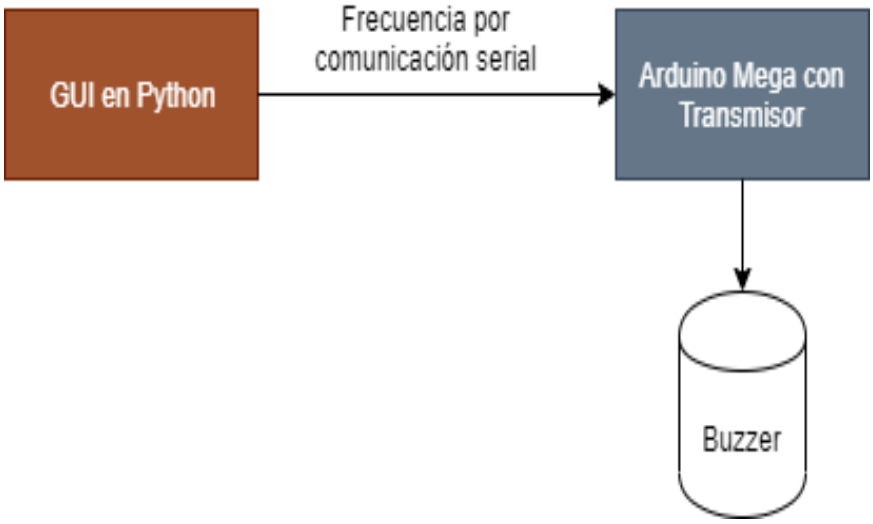


Figura 18: Diagrama que muestra el funcionamiento de la Prueba 1.

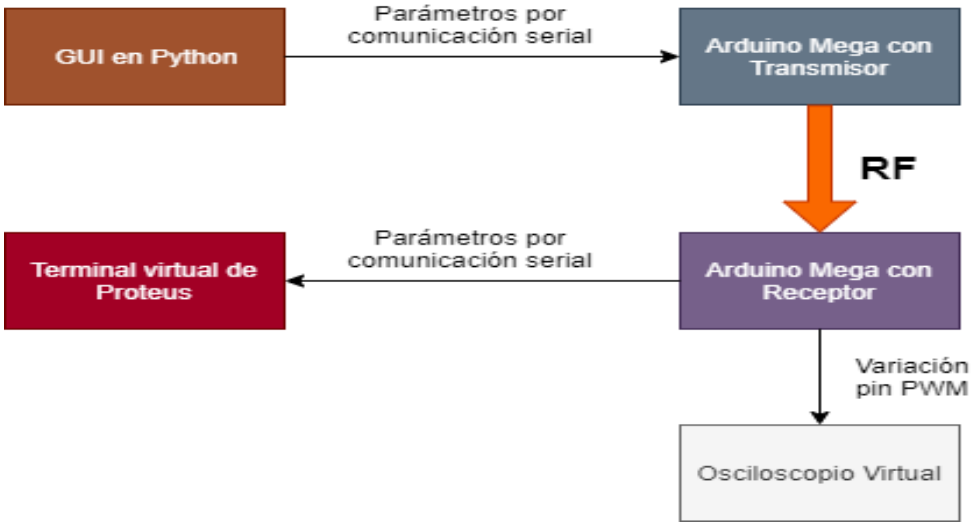


Figura 19: Diagrama que muestra el funcionamiento de la Prueba 2.

Por último, luego de conseguir los módulos RF físicos, se procedió a plasmar todo lo logrado en simulación en los Arduino Uno físicos. La idea es la misma que en la Figura 19, pero en lugar de la terminal virtual de Proteus se utiliza el Monitor Serial de Arduino y no se muestra la variación de la señal PWM en un osciloscopio. A continuación se presentan todas las pruebas de manera detallada.

7.1. Prueba 1

7.1.1. Envío de números a Arduino desde la terminal de Spyder

Para garantizar una buena comunicación entre Arduino y la computadora (utilizando Python), se envían números al Arduino Uno desde la terminal de Spyder. Estos números definen la frecuencia a la que debe resonar un buzzer conectado a éste. Probando con diferentes números, se pudo escuchar cómo cambiaba el sonido que reproducía el buzzer. En la Figura 20, se muestra el envío de los números desde la terminal de Spyder, mostrando el número que se envió para garantizar el envío correcto (visto en la terminal como "Frecuencia: número enviado").

```
In [27]: runfile('C:/Users/Mike/Desktop/Int2Com.py', wdir='C:/Users/Mike/Desktop')
Introduzca una frecuencia o 's' para salir del programa:

300
Frecuencia: 300
Introduzca la frecuencia:

1200
Frecuencia: 1200
Introduzca la frecuencia:

1500
Frecuencia: 1500
Introduzca la frecuencia:
```

Figura 20: Números enviados desde la terminal de Spyder.

7.1.2. Envío de comandos desde una interfaz gráfica hecha en Python

Luego de lograr enviar los números desde la terminal, se diseñó una interfaz gráfica simple en Python para el envío de comandos a Arduino. Esta interfaz preliminar se creó utilizando Tkinter y se puede observar en la Figura 21. En la interfaz se tienen dos botones: ON y OFF. Estos botones sirven para encender (ON) y apagar (OFF) un LED conectado a el Arduino Uno. Esta parte se realizó para probar la comunicación entre el controlador y la interfaz gráfica.

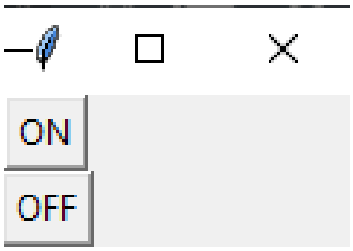


Figura 21: Interfaz para encender y apagar un LED.

Se puede observar que el botón de ON enciende el LED (Figura 22), mientras que el botón de OFF lo apaga (Figura 23).

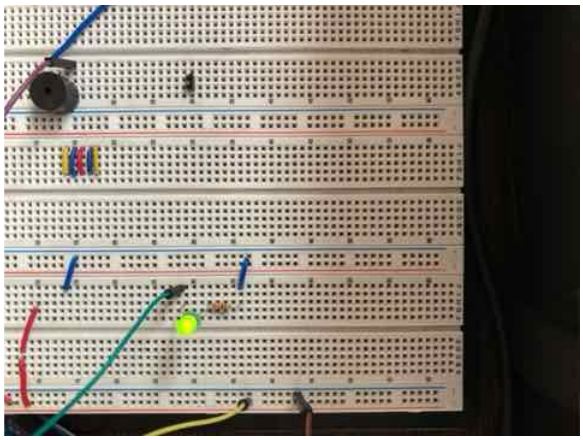


Figura 22: LED encendido por medio de la interfaz.

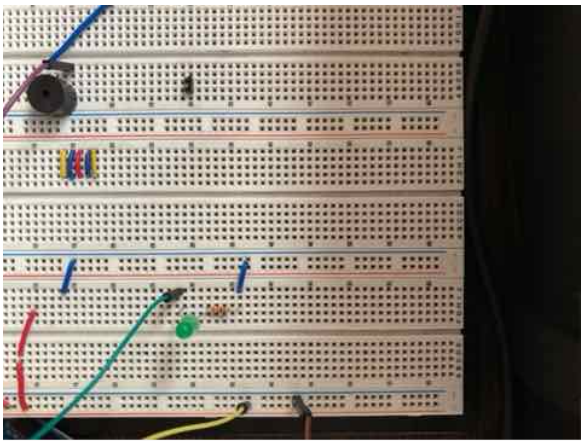


Figura 23: LED apagado por medio de la interfaz.

7.1.3. Interfaz para enviar frecuencias a un buzzer

Se diseñó otra interfaz gráfica simple capaz de enviar una frecuencia al buzzer conectado al Arduino Uno. La frecuencia se introduce en la caja de texto (*textbox*) con el teclado de la computadora y al presionar el botón que lee SEND, se envía el número al controlador por medio de comunicación serial. Al recibir este número, se hace resonar al buzzer a la frecuencia indicada. El botón de STOP sirve para apagar al buzzer (no suena nada) y el botón QUIT se usa para salir de la interfaz y cerrar el puerto serial. La nueva interfaz gráfica preliminar diseñada se puede observar en la Figura 24.

Figura 24: Interfaz gráfica para enviar frecuencias al Arduino Uno.

7.2. Prueba 2

En esta prueba se simula la comunicación inalámbrica en Proteus. Se utilizaron dos Arduino Mega, debido a que estos tienen varios puertos seriales, a diferencia del Arduino Uno que solo tiene un puerto serial. Es necesario tener una mayor cantidad de puertos seriales, pues se necesita un puerto serial para comunicar el controlador de la varilla con Python (para recibir los parámetros) y otro para conectar el módulo transmisor RF (en Proteus éste se conecta por medio de comunicación serial). Del otro lado, el controlador del módulo de estimulación se debe conectar por medio de comunicación serial al módulo receptor RF y a la terminal virtual de Proteus, que sirve para corroborar la recepción correcta de los parámetros. En la Figura 25 se muestra el esquemático de Proteus diseñado para la simulación de comunicación inalámbrica.

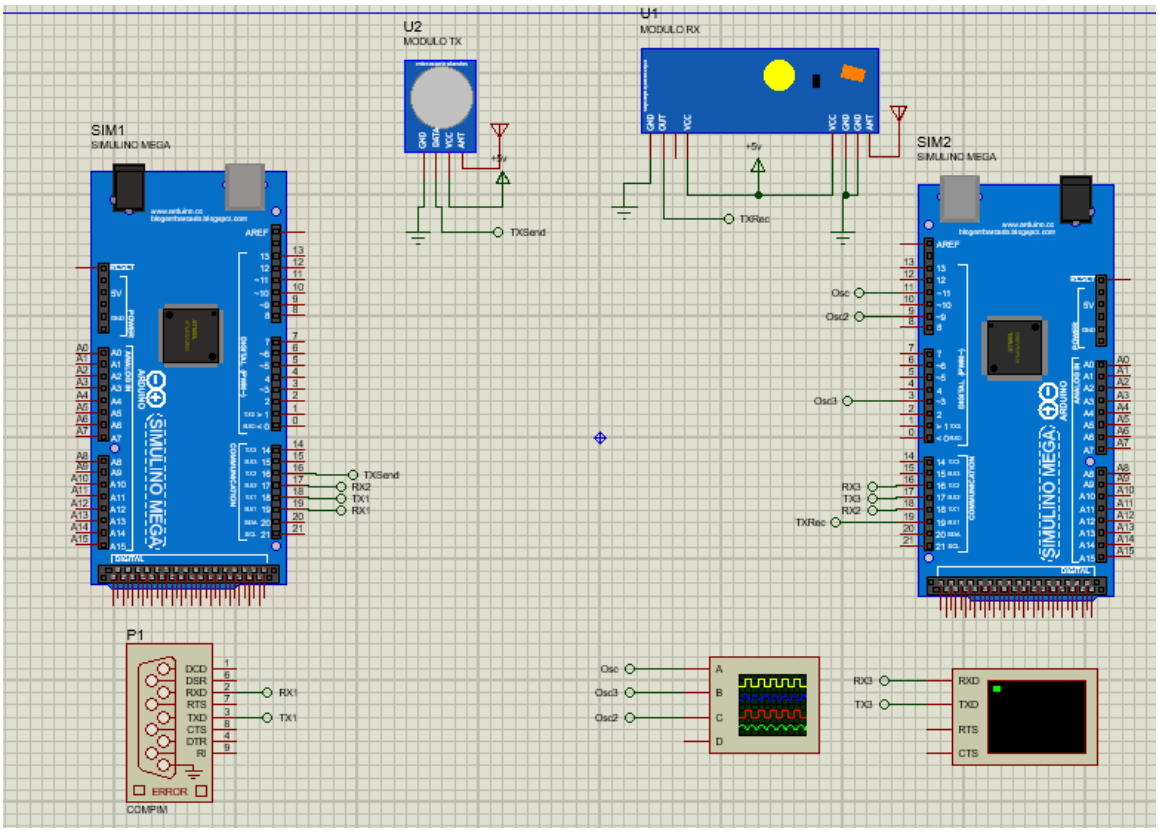


Figura 25: Simulación de comunicación inalámbrica entre Arduinos en Proteus.

7.2.1. Limitaciones del hardware - Arduino Mega/Uno

Una de las limitaciones más grandes de Arduino para este proyecto, es que no se puede variar la corriente de salida de los pines. Es decir, no se puede variar la corriente de estimulación. Este parámetro no será incluido en ninguna de las pruebas realizadas en este proyecto, ya que esto es parte del módulo de estimulación.

Frecuencia de la señal

No es posible cargar un valor de frecuencia arbitrario a los pines PWM de Arduino. Sin embargo, existen opciones predeterminadas de frecuencia que se pueden utilizar en los pines. Para poder usar los valores predeterminados de frecuencia, es necesario modificar los registros de los temporizadores/contadores de Arduino.

Para la señal que se muestra en el osciloscopio virtual más adelante (Figuras 27 y 28), se utilizó un pin PWM asociado al `TIMER2` (temporizador 2) del Arduino Mega. Las frecuencias a las que puede trabajar este temporizador son: 30.64, 122.50, 245.10, 490.20 (predeterminada), 980.93, 3921.16 y 31372.55 Hz.

Tiempo de estimulación y tiempo apagado de estimulación

Los tiempos de estimulación y de apagado de estimulación suceden en ciclos. Luego de terminar el tiempo de estimulación, se deshabilita la estimulación por la cantidad de tiempo definida por el tiempo de apagado. Al terminar este tiempo, se vuelve a habilitar la estimulación por el tiempo definido de estimulación y así en adelante.

Para poder realizar ciclos de tiempo en Arduino, se utilizan los temporizadores. Utilizar retrasos (*delays*) hace que el programa sea mucho más lento. Estos retrasos paran completamente el programa mientras se espera a que termine el tiempo definido, lo cual no es útil para la aplicación requerida, pues se requiere poder variar parámetros en tiempo real. Para esto, se deben utilizar los temporizadores, para que el tiempo de estimulación y de apagado se realicen siempre en ciclos.

Sin embargo, no se pueden utilizar los temporizadores al mismo tiempo para definir la frecuencia y para definir los tiempos de estimulación y de apagado. Al modificar el registro del temporizador para variar la frecuencia, no se puede definir el tiempo deseado. Por otro lado, al modificar el registro del temporizador para definir los tiempos, no se puede escoger la frecuencia deseada. No se pueden realizar variaciones a la frecuencia y a los tiempos en el mismo momento.

Ancho de pulso

Para variar el ancho de pulso de la señal, se utiliza la función `analogWrite()`. Esta función escribe al pin un número entre 0 y 255 (8 bits), que indica de cuánto debe ser el ciclo de trabajo de la señal. Por ejemplo, escribir un valor de 255, indicará un ciclo de trabajo

del 100 %, la señal siempre estará en alto. Si se escribe un valor de 64, el ciclo de trabajo será del 25 %, un cuarto del tiempo estará en alto y tres cuarto es en bajo.

A diferencia de los modelos comerciales, no se puede decidir el ancho de pulso como 500 μs. Para poder acercarse a uno de los valores usados de forma comercial, se debe escoger una frecuencia y un ciclo de trabajo que logren el ancho de pulso deseado.

Resumiendo

Debido a las limitaciones de hardware de Arduino mencionadas, las pruebas de recepción de parámetros se realizaron por separado. Los cinco parámetros son enviados desde el módulo transmisor pero solo uno o dos actúan sobre la señal PWM. Por ejemplo, para probar la variación de la frecuencia, solo se toma en cuenta el parámetro de frecuencia recibido y en el osciloscopio digital solo se muestran cambios en la frecuencia y así con los demás parámetros. Los parámetros utilizados se tratan de acercar a los mencionados en los Cuadros 2, 3 y 4, pero no son exactamente los mismos debido a las limitaciones mencionadas anteriormente.

Las limitaciones del hardware son revisadas y superadas por el proyecto del módulo de estimulación, trabajado por Gustavo Ordoñez.

7.2.2. Variación de parámetros desde Putty

Utilizando los módulos RF de Proteus, se logró la comunicación inalámbrica entre los dos Arduino Mega. El Arduino Mega que simula ser el controlador de la varilla programadora, recibe comandos desde una terminal serial (utilizando Putty) y se los envía al otro Arduino Mega por medio de los módulos RF. En la Figura 25 se puede observar el esquemático diseñado para la comunicación inalámbrica en Proteus y en la Figura 26 se muestra la terminal serial de Putty, en la que se envían 1s y 0s como comandos.

Figura 26: Envío de comandos desde Putty.

Los 1s y 0s enviados desde el controlador de la varilla programadora, los recibe el controlador del módulo de estimulación y los interpreta como comandos para cambiar las propiedades de un pin PWM. Esta señal PWM se puede observar en el osciloscopio digital de Proteus, como en la Figura 27

Para las pruebas realizadas con Putty, solo se muestran las variaciones en la frecuencia. Más adelante, cuando se muestren los resultados al utilizar una interfaz gráfica hecha en Python, se varían todos los parámetros de estimulación como el ancho de pulso y los tiempos de estimulación y apagado.

Variación de la frecuencia

Para la variación de la frecuencia se utilizaron dos frecuencias: 490.20 Hz, que es la frecuencia por defecto de los pines PWM, y 30.64 Hz, que es la que más se acerca a los parámetros mostrados en el Cuadro 2.

En la Figura 27 y en la Figura 28 se muestran dos señales: una azul y una roja. La señal roja es la que se modifica dependiendo del parámetro de frecuencia recibido. La señal azul es una señal de referencia que tiene una frecuencia establecida de 490.20 Hz. Según se modifiquen los parámetros, la señal roja cambia respecto de la azul, pudiendo observar cómo cambia la frecuencia al enviar parámetros distintos. En la Figura 27, se enviaron los comandos para una señal de 490.20 Hz. De esta manera, se puede observar la misma frecuencia en ambas señales. En estas pruebas, las señales tienen un ciclo de trabajo del 25 %.

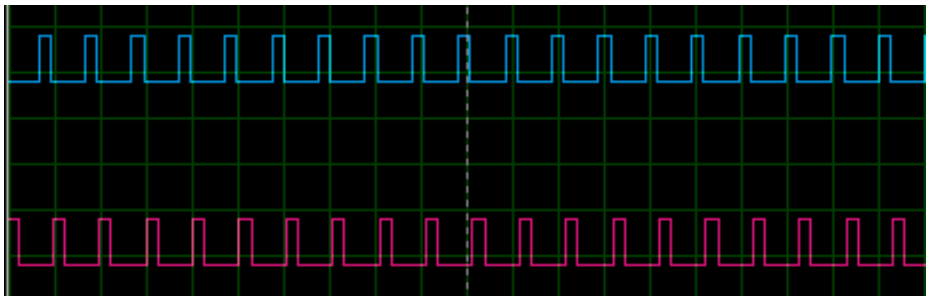


Figura 27: Frecuencia de 490.20 Hz | Escala horizontal: 1ms/div, Escala vertical: 5V/div.

Luego de esto, se envió el parámetro para una frecuencia de 30.64 Hz. En la Figura 28 se puede observar como la frecuencia de la señal roja disminuye considerablemente respecto de la frecuencia de la señal de referencia (azul), comprobando que el envío y la recepción de parámetros fueron realizados de manera correcta.

7.2.3. Simulación de comunicación con Python

Con la comunicación inalámbrica exitosa entre los dos Arduino Mega, utilizando los módulos RF y enviando los comandos desde la terminal serial Putty, se procedió a enviar los parámetros desde la terminal de Spyder, que es el entorno de Python donde se creó la

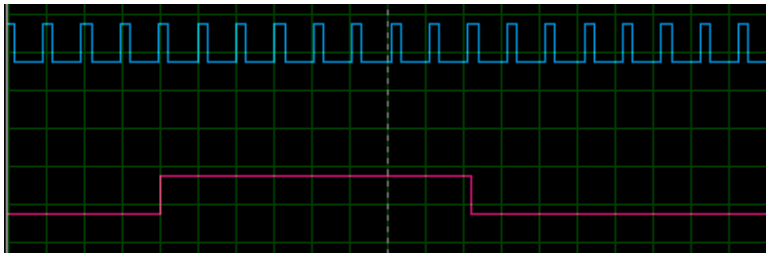


Figura 28: Frecuencia de 30.64 Hz | Escala horizontal: 1ms/div, Escala vertical: 5V/div.

interfaz gráfica.

Se creó un programa en Python para enviar los parámetros, por medio de comunicación serial, al Arduino de la varilla programadora. Estos parámetros son: modo de operación, ancho de pulso, frecuencia, tiempo de estimulación y tiempo apagado (*sleep*). Los cinco parámetros son enviados al módulo receptor y éste muestra en la terminal virtual de Proteus los parámetros recibidos. Sin embargo, al igual que en la parte realizada con Putty, en esta parte solo se muestra variación de la frecuencia. En la siguiente parte de la prueba, ya que se tenga una interfaz gráfica simple preliminar, se procederá a realizar la variación de todos los parámetros.

El programa de Python creado le pregunta al usuario, en la terminal de Spyder, la opción que quiere cargar de cada parámetro. Para seleccionar la opción deseada, solo se debe escribir un número. Como primera prueba se escogió la configuración 1 de todos los parámetros y como segunda prueba se escogió la configuración 2 de éstos. En la Figura 29 y en la Figura 33, se puede observar de manera completa la selección de parámetros en Python y el resultado en la frecuencia de la señal en el osciloscopio de Proteus, más los parámetros recibidos en la terminal virtual. Para estas pruebas, se utilizó un ciclo de trabajo del 25 % en las señales, para que los cambios se pudieran observar de mejor manera.

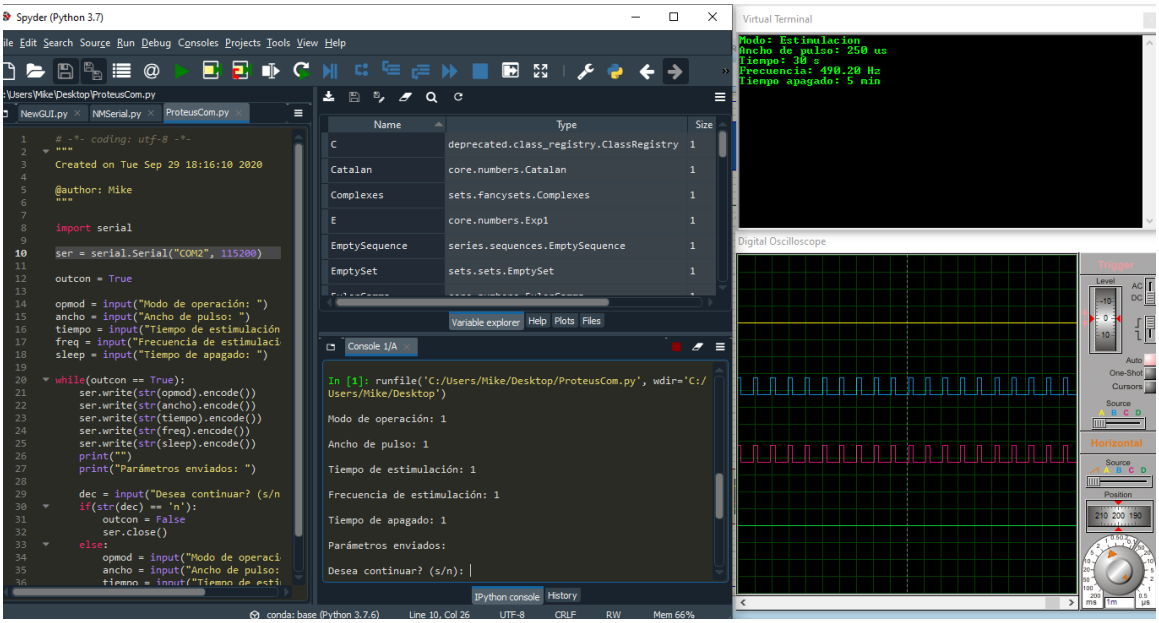


Figura 29: Programación de parámetros y resultado en la señal | Frecuencia de 490.20 Hz.

En las Figuras 30, 31 y 32 se puede observa de mejor manera las diferentes partes de la Figura 29. Se muestra la terminal de Spyder recibiendo las entradas del usuario, la terminal virtual de Proteus mostrando los parámetros recibidos y la señal afectada en el osciloscopio. En el osciloscopio se muestra una señal roja/rosada y una azul. La roja es la señal afectada directamente por los parámetros y la azul sirve como una referencia, para poder observar las variaciones en la frecuencia de la roja con respecto de la azul. Esta señal tiene una frecuencia fija de 490.20 Hz.

La primera configuración de parámetros es: modo de estimulación, ancho de pulso de 250 μs, tiempo de estimulación de 30 segundos con tiempo apagado 5 minutos y una frecuencia de 490.20 Hz.

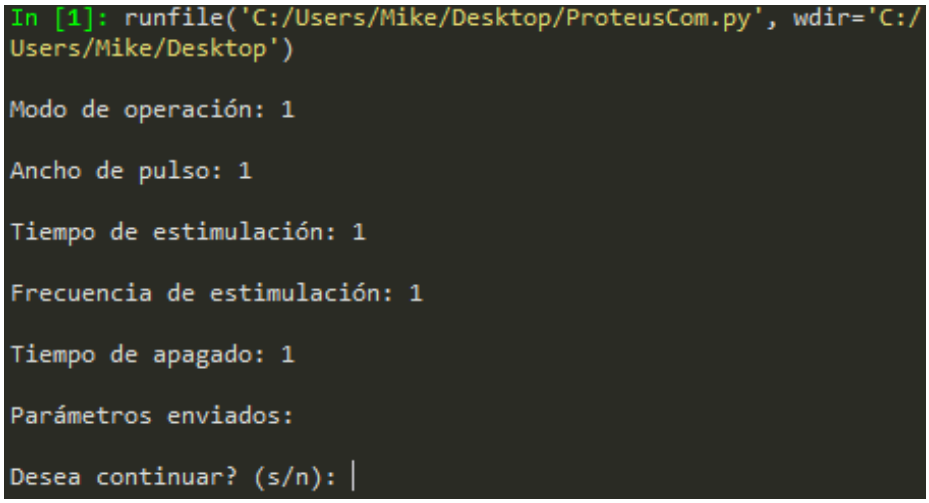


Figura 30: Selección de parámetros en terminal de Spyder, Configuración 1.

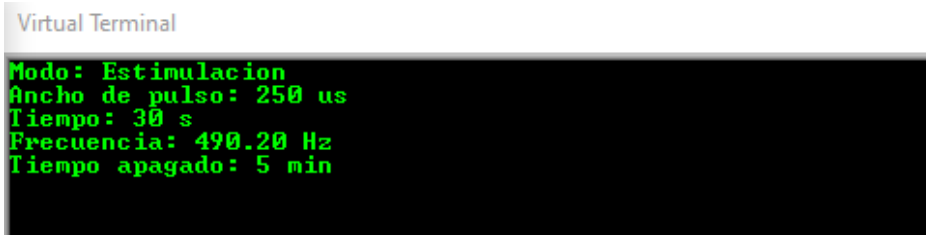


Figura 31: Muestra al usuario de selecciones en terminal de Proteus, Configuración 1.

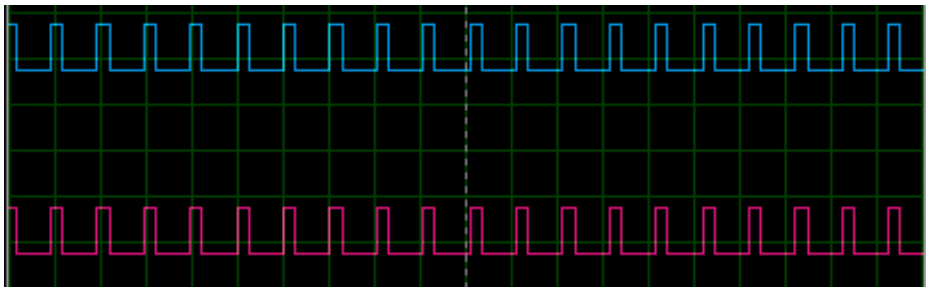


Figura 32: Resultado en la señal, Configuración 1.

Para observar cambios, se probó la segunda configuración de todos los parámetros. Como se mencionó, la Figura 33 muestra de forma completa el sistema. La segunda configuración de parámetros es: modo de programación, ancho de pulso de 500 μs, tiempo de estimulación de 60 segundos sin tiempo de apagado y una frecuencia de 30.64 Hz.

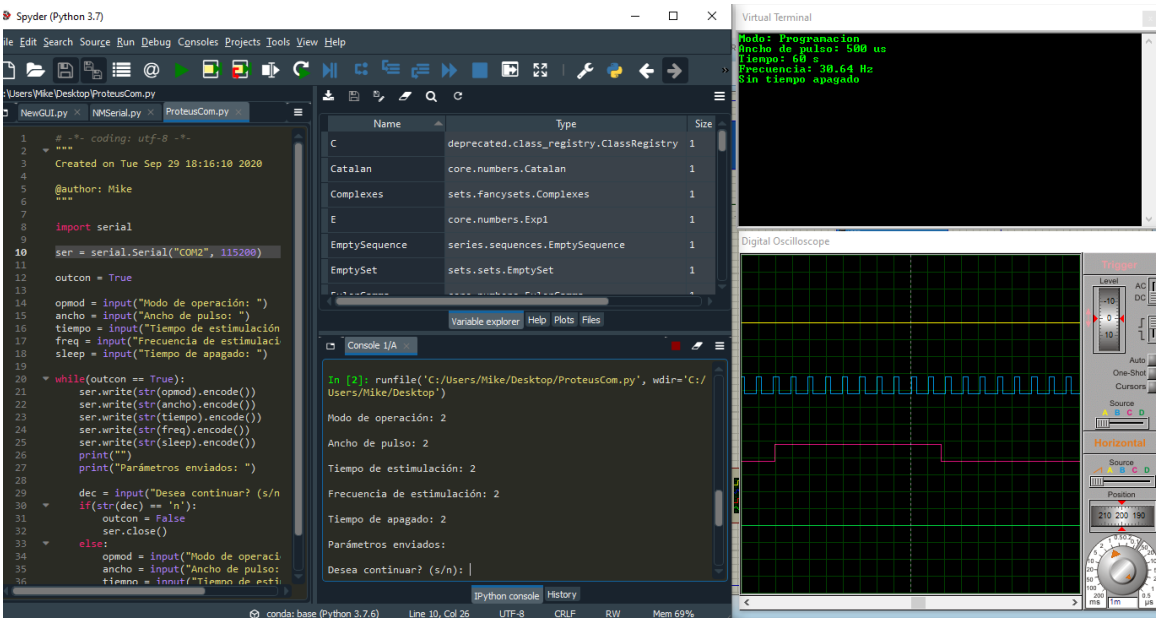


Figura 33: Programación de parámetros y resultado en la señal | Frecuencia de 30.64 Hz

En las Figuras 34, 35 y 36 se puede observar de mejor manera lo mostrado en la Figura 33.

```
In [2]: runfile('C:/Users/Mike/Desktop/ProteusCom.py', wdir='C:/
Users/Mike/Desktop')

Modo de operación: 2

Ancho de pulso: 2

Tiempo de estimulación: 2

Frecuencia de estimulación: 2

Tiempo de apagado: 2

Parámetros enviados:

Desea continuar? (s/n):
```

Figura 34: Selección de parámetros en terminal de Spyder, Configuración 2.

Figura 35: Muestra al usuario de selecciones en terminal de Proteus, Configuración 2.

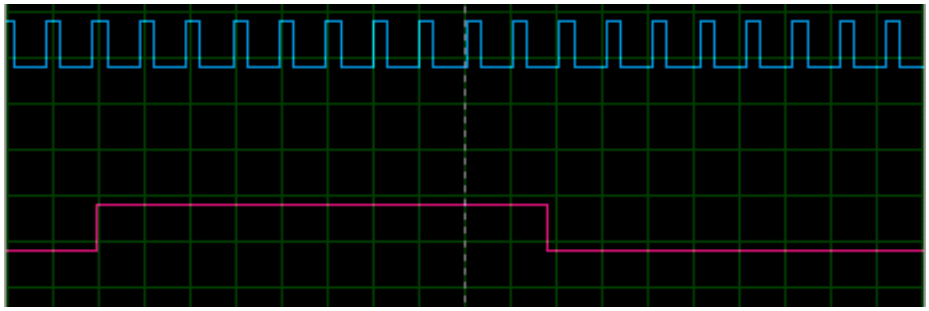


Figura 36: Resultado en la señal, Configuración 2.

Con esta prueba, se pudo comprobar que los parámetros sí se envían de manera correcta desde Python hasta el Arduino Mega conectado por comunicación serial. Luego se envían de la forma correcta por medio de los módulos RF, haciendo que la frecuencia de la señal varíe.

7.2.4. Simulación de comunicación con Python utilizando GUI

Sabiendo que se podían enviar los parámetros al controlador de la varilla programadora desde Python, se procedió a hacer lo mismo pero esta vez desde una interfaz gráfica (GUI). Es importante poder cargar los parámetros a la varilla desde una interfaz gráfica, pues el personal médico que programe los parámetros probablemente no tenga un conocimiento amplio de programación, por lo que programar desde una terminal sería muy difícil.

La nueva interfaz gráfica diseñada (diseño funcional preliminar) se puede observar en la Figura 37 y tiene 2 botones y 5 listas desplegables (*dropdown list/box*). Uno de los botones sirve para enviar los parámetros por medio de comunicación serial y lee "ENVIAR". El otro botón lee "SALIR" y sirve para salir de la interfaz gráfica, cerrando el puerto serial para poder conectarse de nuevo sin problemas al iniciar el programa.

Las 5 listas desplegables, una para cada parámetro de estimulación, contienen las diferentes opciones disponibles para cada parámetro. Al presionar la lista, se despliegan todas las opciones y al seleccionar una, la lista se vuelve a plegar, mostrando solo la opción seleccionada. Una vez se hayan decidido todos los parámetros, se presiona ENVIAR para enviarlos al controlador de la varilla programadora por medio de comunicación serial.

Ya teniendo la interfaz gráfica preliminar lista, se procedió a realizar las mismas pruebas que en la sección anterior: comprobar que la comunicación entre Python y Proteus, esta

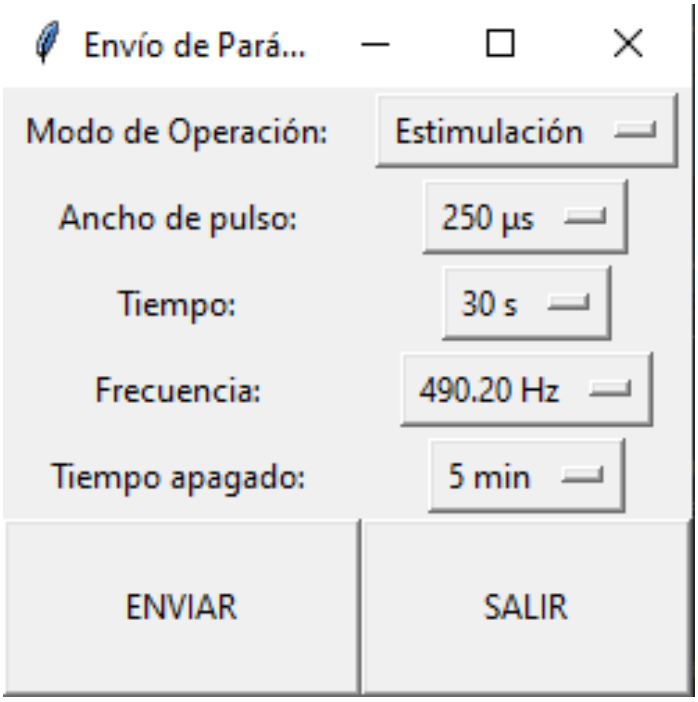


Figura 37: Interfaz gráfica utilizada para enviar los parámetros por medio de comunicación serial.

vez utilizando la GUI, fuera exitosa y que la programación inalámbrica de los parámetros también fuera exitosa. Para estas pruebas sí se variaron todos los parámetros. En las Figuras 38 y 39, se puede observar el envío y recepción correctos de parámetros desde la GUI. En estas figuras, se muestra la variación de la frecuencia, de 490.20 a 30.64 Hz, en el osciloscopio y la correcta recepción de todos los parámetros en la terminal virtual de Proteus.

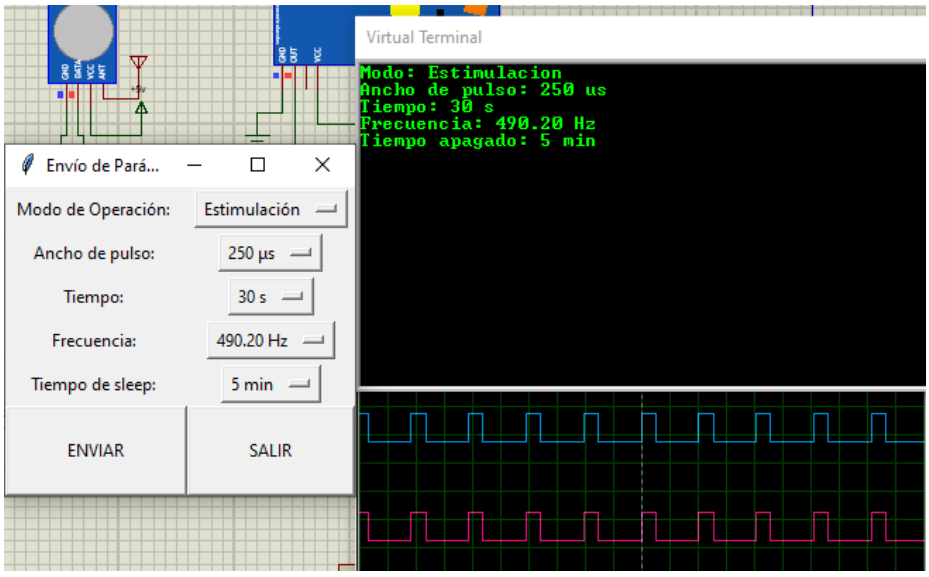


Figura 38: Resultado de la señal con los parámetros establecidos desde la GUI, Configuración 1.

En la figura anterior, los parámetros seleccionados fueron: modo de estimulación, ancho

de pulso del 250 μs, tiempo de estimulación 30 s, tiempo apagado 5 min y frecuencia de 490.20 Hz. Al igual que en pruebas anteriores, la señal azul se utiliza como referencia para observar los cambios en la señal roja respecto de la azul.

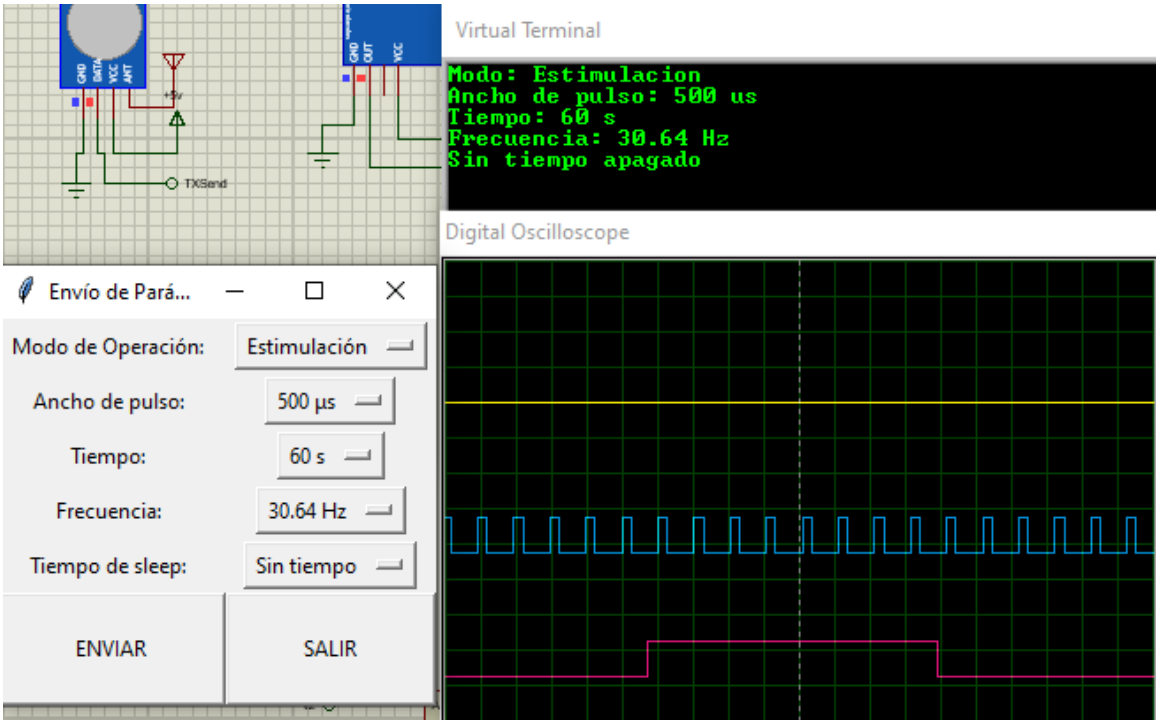


Figura 39: Resultado de la señal con los parámetros establecidos desde la GUI, Configuración 2.

En la Figura 39, se muestra la señal afectada por la segunda configuración de frecuencia (30.64 Hz), y se puede observar cómo la frecuencia de la señal roja disminuye considerablemente respecto de la frecuencia de la señal azul.

Confirmando que los parámetros fueron bien recibidos cuando se envían desde la GUI, se probó variar los demás parámetros. En las siguientes figuras solo se muestra el resultado de la señal en el osciloscopio, para tener un mejor enfoque en la señal, que es la parte importante.

Variación de los Modos de Operación

Siguiendo lo realizado en la Fase I, se utilizan tres modos de operación: estimulación, programación y reposo. En este prototipo, solo se utilizan para demostración del envío correcto de parámetros y no se implementan modos de funcionamiento como tal.

Variación del ancho de pulso

En los Cuadros 3 y 4 se muestran dos configuraciones de ancho de pulso utilizadas comúnmente: 250 μs y 500 μs. Debido a las limitaciones del hardware mencionadas anteriormente,

para poder configurar un ancho de pulso cercano a los valores utilizados de forma comercial, se debe variar la frecuencia y usar la función `analogWrite()`.

Para lograr el ancho de pulso de 250 μs, se utilizó la frecuencia de 3921.16 Hz con un ciclo de trabajo del 98 % (`analogWrite(250)`). En la Figura 40 se puede observar la señal resultante. En esta figura, el osciloscopio tiene una escala horizontal de 50 μs/div. Esto quiere decir que 5 divisiones (cuadrados) horizontales son 250 μs. Como se puede observar en la figura, el pulso dura 5 divisiones, indicando que duró la cantidad indicada de tiempo.

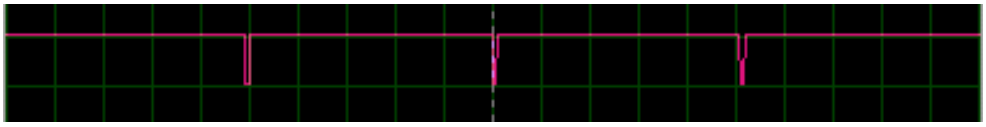


Figura 40: Señal con ancho de pulso igual de 250 μs.

En la Figura 41, se varió el ancho de pulso a 500 μs. Para esto se utilizó la frecuencia de 980.39 Hz con un ciclo de trabajo del 51 %. En la figura, el osciloscopio tiene una escala horizontal de 0.5 ms/div (500 μs/div), indicando que el pulso solo debería durar un cuadrado y eso es lo que se puede observar.

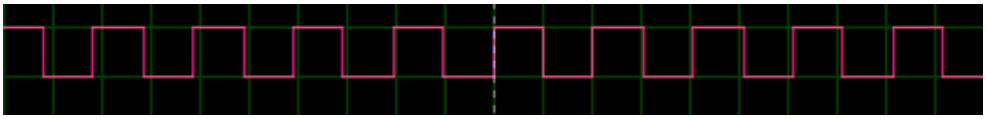


Figura 41: Señal con ancho de pulso igual de 500 μs.

Variación de los tiempos de estimulación y apagado

Para la variación de los tiempos de estimulación y apagado de la señal, se utilizó el `TIMER1` de Arduino. Utilizando este temporizador, se activa y se apaga una señal cada medio segundo. Es decir, un cuarto de segundo en alto y un cuarto de segundo en bajo. Se utiliza medio segundo porque, utilizar los tiempos reales (30/60 s de estimulación y 5 min de apagado), hace que la simulación se sobrecargue y no funcione. En la Figura 42, esta señal es la de color azul y, como referencia, se muestra la señal roja, una señal cuadrada con periodo de un cuarto de segundo.

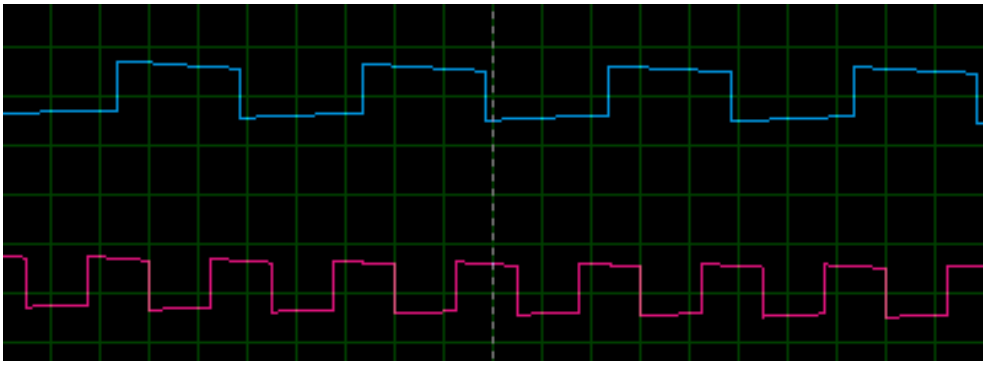


Figura 42: Señal con los tiempos de estimulación y apagado | Escala horizontal: 0.25s/div.

En la Figura 42, las señales no son cuadradas perfectas. Esto se debe a que los temporizadores de Arduino no son muy buenos para generar señales controladas por los temporizadores. Sin embargo, se observa que sí funciona el envío de parámetros desde la GUI hecha en Python.

7.3. Prueba 3

Para este prueba, ya se contaba con los módulos RF físicos (Figura 43), así que se procedió a realizar las pruebas anteriores de manera física. Las partes de esta prueba número 3 difieren, en algunos aspectos, de las realizadas en la prueba 2. Esto es debido a que para la prueba 2 se utilizaron dos Arduino Mega y para esta prueba se usaron dos Arduino Uno.

7.3.1. Comunicación inalámbrica física con Módulos RF de 433 MHz

Para familiarizarse con los módulos RF físicos, se utilizaron estos conectados a dos Arduino Uno (controlador de la varilla programadora y controlador del módulo de estimulación) para enviar números enteros. Estos módulos trabajan a 433 MHz y se pueden utilizar para enviar información de un controlador a otro sin muchas pérdidas en la información y con una velocidad estable.

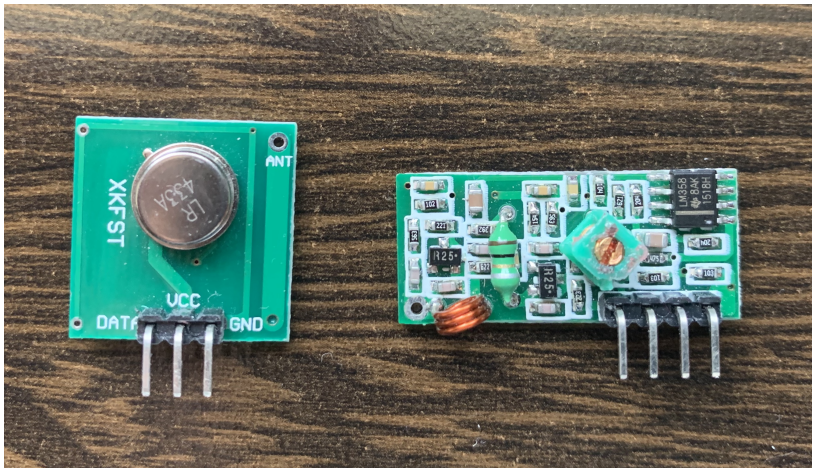


Figura 43: Módulo transmisor y módulo receptor de 433 MHz.

En el IDE de Arduino, se puede utilizar la librería `RadioHead` para controlar de una manera fácil los módulos RF sin la necesidad de usar el puerto serial, debido a que estos estarán ocupados conectados con las computadoras (una que contiene la interfaz gráfica y otra que se usa para monitorear los valores recibidos en el monitor serial de Arduino). En la Figura 44 se puede observar la forma de conectar el transmisor al Arduino Uno. El pin de datos puede ser cambiado en programación de manera fácil.

Por el otro lado, en la Figura 45 se muestra la forma de conectar el módulo receptor al Arduino Uno. Los dos pines de en medio están conectados entre sí, así que se puede utilizar

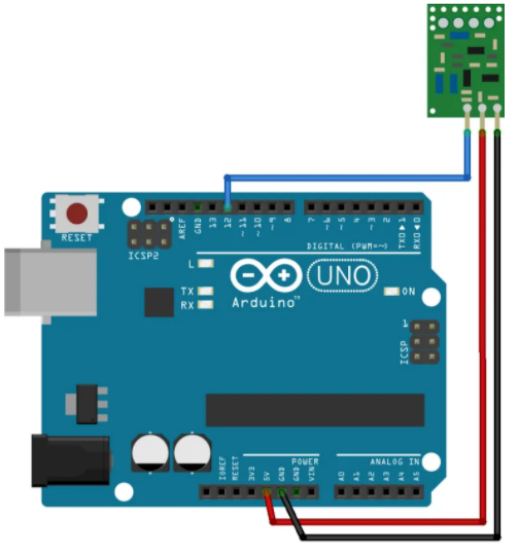


Figura 44: Esquemático de conexión del transmisor [53].

cualquiera para recibir los datos. Al igual que con el módulo transmisor, el pin al que está conectado en el Arduino puede ser cambiado de manera fácil en programación.

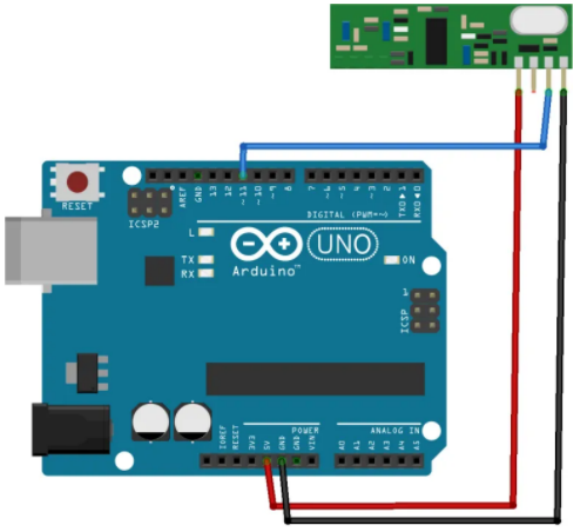


Figura 45: Esquemático de conexión del receptor [53].

Siguiendo los esquemáticos, se procedió a conectar todo de forma física. El resultado se puede observar en la Figura 46. De lado izquierdo (protoboard blanco) está conectado el transmisor, y del lado derecho (protoboard rojo) está conectado el receptor.

Como primera parte de esta prueba, se probó la comunicación inalámbrica entre Arduinos. Para comprobar que la comunicación fuera correcta, se envío un número entero (500 en este caso) desde el transmisor hasta el receptor. Para poder enviar el número desde el transmisor, se utilizó el código que se muestra en la Figura 47, usando la librería `RadioHead`.

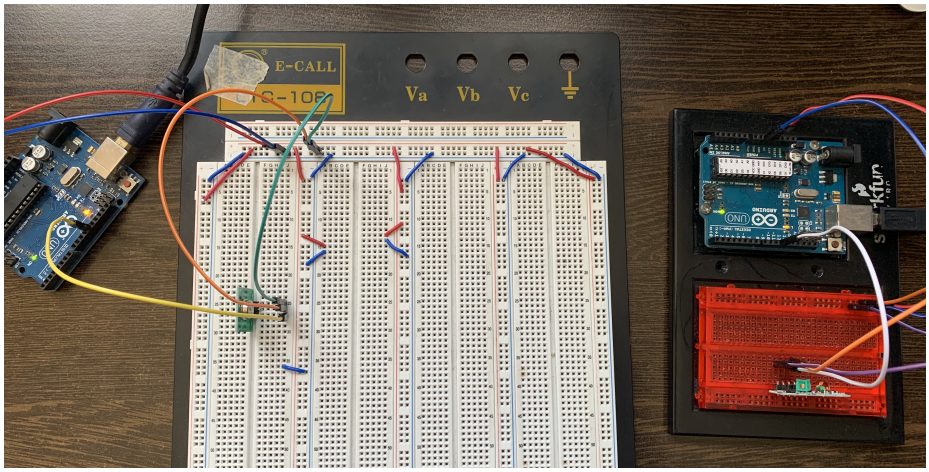


Figura 46: Conexiones de los módulos transmisor y receptor en físico.

Como prueba, el número se envía de forma al repetida al receptor cada 500 ms, para que sea posible leerlo en el Monitor Serial de Arduino

```
int val = 500;
String sstr = String(val);

static char *msg = sstr.c_str();

//const char *msg = "Hola";
rf_driver.send((uint8_t *)msg, strlen(msg));
rf_driver.waitPacketSent();
delay(500);
```

Figura 47: Código para envío de un número a módulo receptor.

Este número se recibe en el Arduino Uno conectado al módulo receptor y se imprime en el monitor serial de Arduino como “Message Received: 500”. El monitor serial imprimiendo los números se puede observar en la Figura 48.

7.3.2. Envío de parámetros utilizando los Módulos RF

Habiendo comprobado la conexión entre los módulos RF físicos, se procedió a unir todas las pruebas realizadas anteriormente. Se conectó la PC al Arduino Uno de la varilla programadora por medio cable USB (serial) y desde la interfaz gráfica de la Figura 37 se le cargaron los parámetros. Los parámetros se envían desde el transmisor hasta el receptor y se imprimen en el monitor serial de Arduino para comprobar que sean los que se enviaron.

Figura 48: Impresión de número recibido en el monitor serial de Arduino.

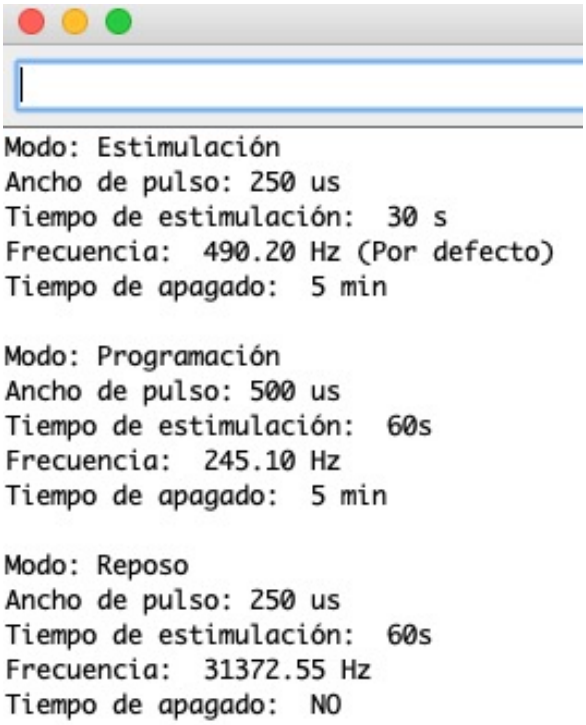


Figura 49: Parámetros recibidos mostrados en monitor serial.

En la Figura 49 se muestran tres diferentes envíos de parámetros. Los parámetros fueron recibidos e interpretados sin problemas. Para estas pruebas, no se pudo acceder a un osciloscopio físico, por lo que no se pudieron verificar las señales resultantes. Sin embargo, el cambio de la señal de acuerdo a los parámetros ya fue validado con la simulación en la sección anterior.

7.4. Mejoras a la interfaz gráfica/programador

Con el Prototipo 1 terminado y realizado con éxito, se procedió a realizar dos mejoras importantes a la interfaz gráfica que funciona como el programador de la varilla programadora: búsqueda automática del puerto serial en el que se conectó el Arduino Uno y exportación de la interfaz gráfica (un archivo de Python) a una aplicación (un archivo .EXE).

Como primer punto a mejorar, se estableció la búsqueda automática del puerto al que se conecta el Arduino Uno. Es decir, al conectar el Arduino, el programa se encarga de encontrar el puerto USB (serial) en el que está conectado. Esto es muy necesario debido a que, probablemente, las personas que manejen el sistema VNS no tengan los conocimientos más amplios de programación. Pedirle a alguien con pocos conocimientos de programación que cambie el puerto del Arduino en el código, complicaría mucho el uso. Además, incluso para una persona con conocimientos de programación sería demasiado tedioso tener que ingresar al código de la GUI cada vez que se conecte la varilla programadora, para poder cambiar el número de puerto en el que se encuentra conectada.

```
def findArduino(portsFound): # Funcion para buscar al Arduino en los puertos
    commPort = 'None' # Condicion para errores
    numConnection = len(portsFound) # Numero de puertos activos

    for i in range(0, numConnection): #Buscar en todos los puertos activos
        port = foundPorts[i] # Tomar el puerto que coincide con el contador
        strPort = str(port) # Convertir su nombre a string

        if 'Arduino' in strPort: # Se busca la palabra Arduino en el string
            splitPort = strPort.split(' ') #Se separa el string en los espacios
            commPort = (splitPort[0]) # Se guarda el puerto donde se encuentra
            # el Arduino
            # Arduino aparece como COMX - Arduino Uno (COM3), lo que nos interesa
            # es el COMX por eso tomamos el espacio 0 del string divido, lo que
            # corresponde a COMX

    return commPort # Se retornoa el puerto del Arduino

#*****************************************************************************
# CONEXION CON ARDUINO
#*****************************************************************************
foundPorts = get_ports() # Se obtienen los puertos disponibles
connectPort = findArduino(foundPorts) # Se busca el puerto del Arduino

if connectPort != 'None': # Si se encontro el puerto
    ser = serial.Serial(connectPort, 115200) # Nos conectamos a este
    print('Connected to' + connectPort)
else:
    print('Error de conexion, revisar')
```

Figura 50: Código para búsqueda automática del puerto serial de Arduino.

En la Figura 50, se muestra el código implementado para la búsqueda automática del puerto USB. Se lee la lista de puertos desde Python. El puerto del Arduino en esta lista aparece como "COMX - Arduino Uno (COMX)", donde X es el número del puerto en el que se encuentra conectado. Entonces, para automatizar el proceso, se busca la palabra "Arduino" en el listado de puertos. Al encontrarlo, se separa la cadena resultante en pedazos, separados por los espacios en la cadena. El primer pedazo contiene "COMX" que es lo que se necesita para inicializar el puerto serial, así que este pedazo es el que se utiliza para conectar con el Arduino.

La otra mejora necesaria es la de crear un aplicación a base de la interfaz gráfica. No todas las computadoras tienen Python, para poder utilizarlo y abrir el programa que implementa la GUI y correrlo. Aun si todas las computadoras lo tuvieran, no todos los médicos encargados sabrían cómo abrirlo y correr el programa desde la terminal o desde una aplicación como Spyder. Por esto, se decidió crear una aplicación por aparte (.EXE) que se puede abrir de forma más universal sin la necesidad de tener instalado Python. Debido a que la interfaz se creó en Python 3.7, la aplicación solo puede ser usada en máquinas que tengan sistema operativo Windows 7 o una versión más reciente. Esta aplicación corre la interfaz gráfica creada y se puede utilizar sin problemas para enviar los parámetros al Arduino conectado en el puerto USB (serial).

Figura 51: Icono de la aplicación.

En la Figura 51 se puede observar el icono de la aplicación. Al abrir la aplicación, se observa la interfaz gráfica con la que se ha estado trabajando de la sección de Prueba 2. Se puede cerrar presionando el botón CERRAR o presionando la X, ambas opciones cierran el programa de manera completa, así que no hay problemas para conectar el puerto serial al abrir la aplicación de nuevo.

CAPÍTULO 8

Prototipo 2

Para el Prototipo 2 se utilizó el módulo WiFi ESP8266 como controlador de la varilla programadora. Este módulo se utilizó para poder conectar la varilla programadora a la computadora (interfaz gráfica) de dos maneras: con cable USB y de forma inalámbrica. Para conectar el controlador de la varilla programadora de forma directa a la computadora con un cable USB, se utilizó la placa de desarrollo `NodeMcu`. El `NodeMcu` es un kit de desarrollo basado en el módulo ESP8266, que explota las capacidades del módulo WiFi, integrando protocolos como I2C y SPI, más GPIO, pines con PWM, etc. todo en una placa.

Para poder conectar el controlador de la varilla a la computadora de forma inalámbrica, se utilizó WiFi. A través de WiFi, se enviaron los parámetros a programar desde la interfaz gráfica hasta la varilla programadora.

Al igual que en el prototipo anterior, se utilizaron los módulos RF de 433 MHz. Un Arduino Uno siguió siendo el controlador que simula ser el módulo de estimulación. Como primer prueba, se realizó lo logrado en el Prototipo 1 pero, esta vez, utilizando el ESP8266 (`NodeMcu`), verificando el funcionamiento correcto de la comunicación de parámetros desde la computadora al controlador de la varilla, por medio de USB. Después de comprobar que sí se enviaron los parámetros del `NodeMcu` al Arduino Uno por medio de RF, se procedió a conectar la varilla con la interfaz por medio de WiFi.

Por último, se implementó una interfaz gráfica capaz de comunicarse con el `NodeMcu` por medio de WiFi y comunicación serial. Si no hay problemas de red, los parámetros se envían por medio de WiFi. Si no se puede conectar a la red, se conecta el cable USB y se envían los parámetros por medio de comunicación serial. En la Figura 52, se muestra un diagrama que muestra el funcionamiento de este prototipo.

A continuación se presentan las pruebas realizadas de este prototipo con mayor detalle.

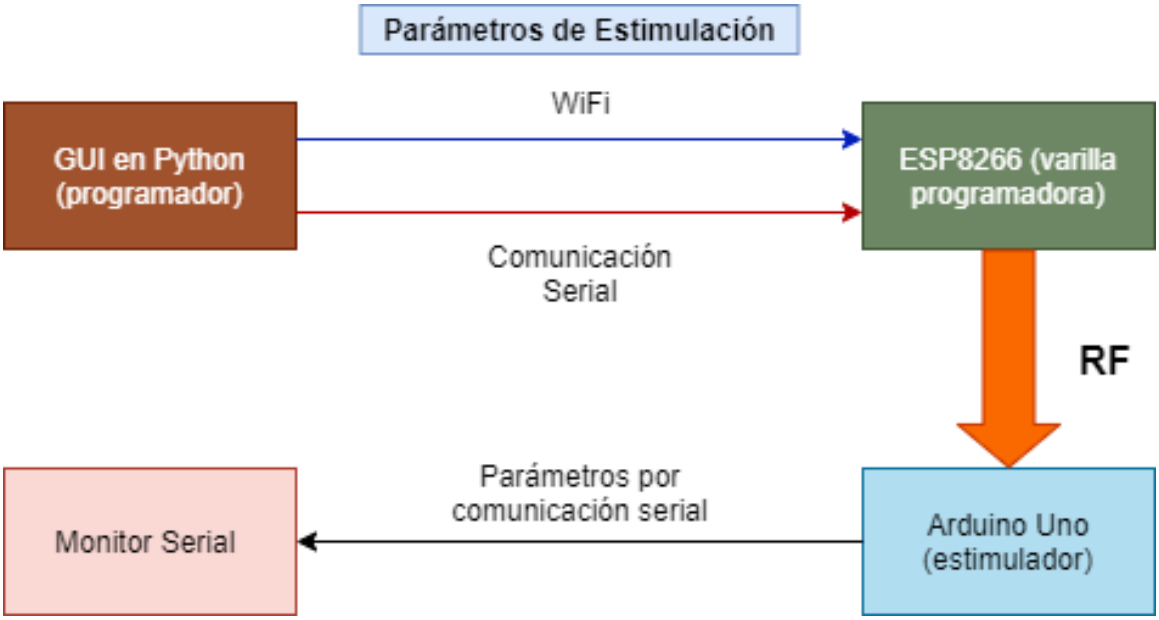


Figura 52: Diagrama de funcionamiento del Prototipo 2.

8.1. Replicación de lo logrado en el Prototipo 1

Para empezar a replicar en el NodeMcu lo que se logró en el Prototipo 1 con los módulos RF de 433 MHz, se envío un número del módulo transmisor al módulo receptor. Esta vez utilizando la librería `RCSwitch`, debido a que la librería `RadioHead` no tiene soporte para ESP8266.

Se conectó el módulo transmisor al `NodeMcu` en el pin 4 de propósito general y el módulo receptor se conectó al pin 2 (un pin con capacidad de interrupciones) del Arduino Uno. En la Figura 53 se puede observar las conexiones en físico de los módulos.

En la Figura 53, abajo se encuentra el NodeMcu conectado al módulo transmisor y arriba se observa el Arduino Uno conectado al módulo receptor.

Para comprobar que la comunicación fuera correcta, se enviaron dos números distintos: 5393 y 5397 (números utilizados en los ejemplos de la librería). Estos números se fueron enviando de manera intercalada, con retrasos (*delays*) de 1 segundo entre cada envío. En la Figura 54 se muestra el monitor serial que despliega los valores recibidos en el módulo receptor. Sin embargo, como se puede observar en la figura, los números fueron desplegados 4 veces cada uno. Esto quiere decir que el transmisor está enviando un número cuatro veces, lo cual es un error para lo que se quiere lograr.

Para solucionar esto, en el módulo receptor solo se toma en cuenta la primera vez que se recibe cada parámetro. Luego de recibir el parámetro la primera vez, se omiten/desechan las otras tres recepciones. Luego de recibir todos los parámetros, se muestran en el Monitor Serial de Arduino para poder verificar la recepción.

Como comprobación, se enviaron los 5 números entre 5393 y 5397, incluyendo ambos. Estos números solo se usan como prueba y no tienen ninguna relación con los parámetros

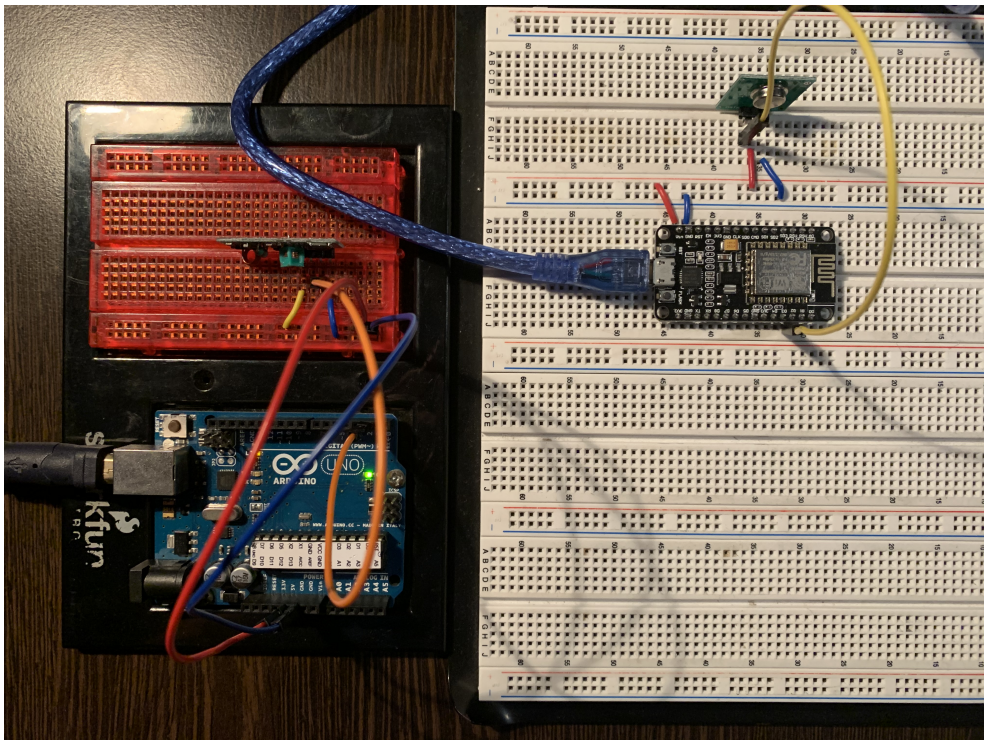


Figura 53: Conexión física de los módulos RF usando NodeMcu.

de estimulación. Como se puede observar en la Figura 55, los 5 números se muestran una única vez en el Monitor Serial de Arduino, a diferencia de lo que se observa en la Figura 54.

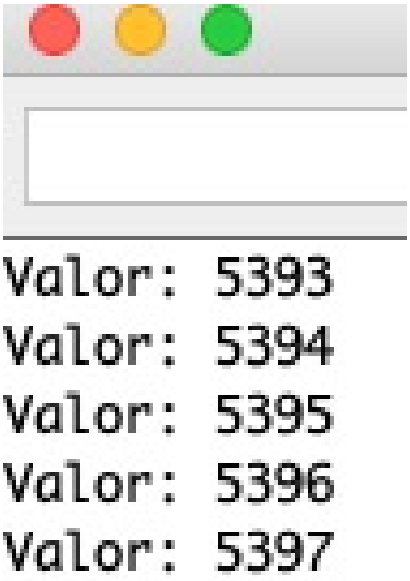


Figura 55: Mejora en la recepción de números utilizando RCSwitch.

Habiendo arreglado la recepción de parámetros, se procedió a comunicar la computadora, mediante la interfaz gráfica hecha en Python, y el módulo ESP8266 por medio del protocolo

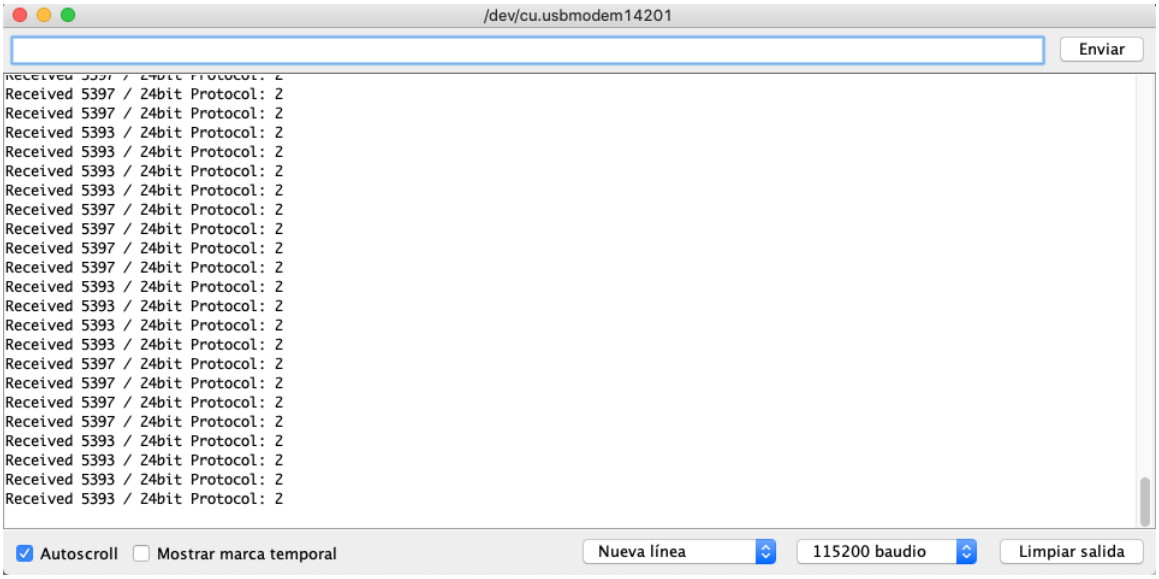


Figura 54: Recepción de números utilizando RCSwitch.

de comunicación serial, usando un cable USB. Para la interfaz gráfica de Python, se utilizó el mismo programa, únicamente cambiando el indicador de la búsqueda automática del puerto serial. En lugar de buscar la palabra "Arduino", se busca la palabra "Silicon", pues el puerto se lee como "Silicon Labs CP210x".

Para el programa del módulo ESP8266, se utilizó la base del programa de Arduino del Prototipo 1. Para recibir los parámetros se utilizó el mismo código. Lo que se cambió fue la librería utilizada, cambiando de `RadioHead` a `RCSwitch`.

Utilizando este programa y el de la interfaz gráfica en Python, cambiando solo el puerto serial, se logró enviar los parámetros de la computadora al controlador de la varilla y que éstos se enviaran, por medio de RF, al controlador del módulo de estimulación. La interfaz gráfica utilizada se puede observar en la Figura 37 y los parámetros recibidos en el Arduino Uno con el módulo receptor en la Figura 56, para una corrida con: modo de programación, 250 μs de ancho de pulso, 60 segundos de tiempo de estimulación, 30.64 Hz y sin tiempo de apagado.

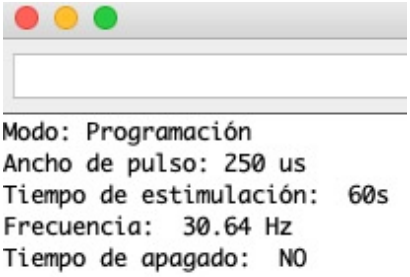


Figura 56: Parámetros recibidos conectando GUI y ESP8266 por medio de comunicación serial.

8.2. Envío de parámetros por medio de WiFi

Después de replicar el funcionamiento del Prototipo 1 con el módulo ESP8266, se procedió a realizar la comunicación entre Python (la interfaz gráfica) y el módulo por medio de WiFi. Para esto, se utilizó el conjunto de protocolos TCP/IP con la arquitectura cliente-servidor. En este tipo de arquitectura, un dispositivo electrónico llamado cliente, realiza peticiones a otro dispositivo electrónico llamado servidor, que da respuesta a las peticiones del cliente.

Se utilizó la librería `WiFiClient` de Arduino, para poder utilizar el ESP8266 como servidor. El servidor busca conectarse a la red y, una vez lo logra, muestra en el Monitor Serial que la conexión fue exitosa y la dirección IP del módulo. Luego busca si hay un cliente conectado al servidor. Si no lo hay, busca a un cliente que esté disponible, un cliente que esté conectado al servidor. De esta manera, la computadora se conecta como cliente al módulo ESP8266 funcionando como servidor, y le envía los parámetros de estimulación al servidor.

Para poder utilizar la computadora como cliente en Python, se utilizó la librería `Socket`. Se indica el puerto y la dirección IP del servidor y con esto se conecta el programa. Con la conexión exitosa, se pueden enviar los parámetros al servidor.

Se utilizó una nueva interfaz gráfica simple preliminar para enviar los parámetros al módulo de la varilla programadora por medio de WiFi. La Figura 57 muestra esta interfaz gráfica preliminar. Se seleccionan los parámetros en las listas desplegables y al presionar "Enviar", éstos se envían por WiFi al módulo ESP8266.

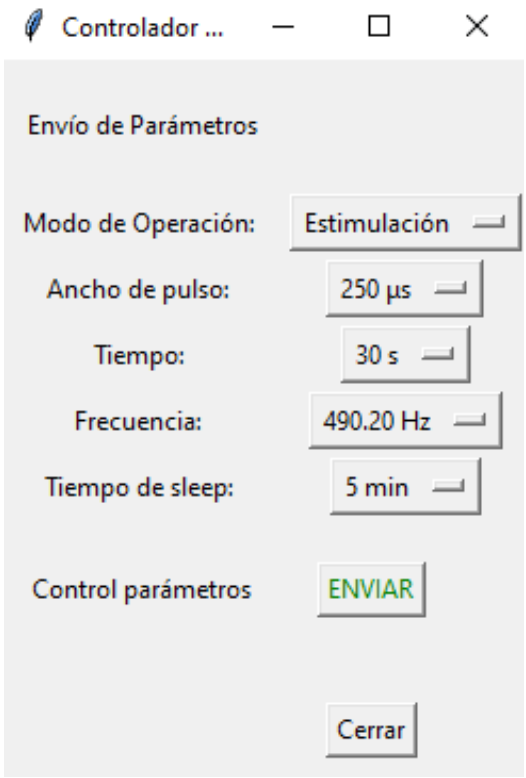


Figura 57: Confirmación de conexión a la red del módulo ESP8266.

Luego de recibir los parámetros, el módulo envía por medio de RF, los parámetros al módulo receptor. Al recibirlos, el Arduino Uno muestra los parámetros en el Monitor Serial para verificación. En la Figura 58 se muestran la información recibida, con los parámetros de la Figura 57 enviados desde el módulo transmisor.

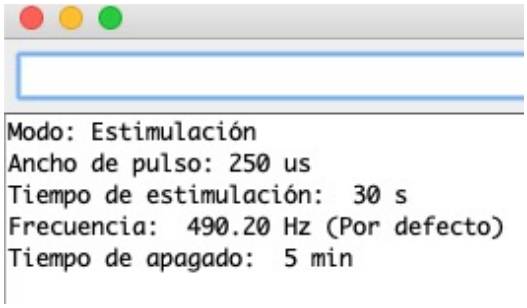


Figura 58: Muestra de parámetros recibidos por medio de WiFi en monitor serial de Arduino

8.3. Combinación de envío de parámetros por medio de WiFi y/o comunicación serial

Luego de lograr la comunicación por medio de protocolo serial y por medio de WiFi, entre la PC (interfaz gráfica) y el módulo ESP8266 por separado, se procedió a combinar los dos tipos/protocolos de comunicación en un mismo programa.

Primero, se implementaron ambos tipos de comunicación en el módulo ESP8266. Al iniciarse, el módulo busca conectarse a la red WiFi. Si lo logra, muestra en el Monitor Serial de Arduino que se conectó de manera correcta y la IP del módulo, como puede observar en la Figura 59. Luego de lograr conectarse a la red, se prepara para recibir parámetros por WiFi. Es importante darle bastante tiempo (10-15 segundos) al módulo para que se pueda conectar a la red, no lo hace de manera inmediata.

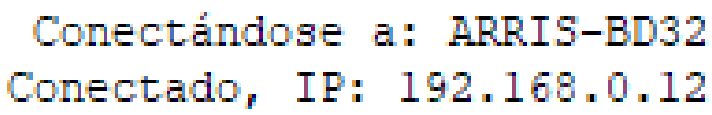


Figura 59: Confirmación de conexión a la red del módulo ESP8266.

Si no logra conectarse a la red, lo indica en el Monitor Serial, como se muestra en la Figura 62 y se prepara para recibir datos por medio de comunicación serial.

Segundo, se diseñó una nueva interfaz gráfica mejorada y final, que sirve para enviar parámetros por medio de WiFi y por medio del cable USB (comunicación serial) de una forma ordenada y fácil. Al igual que en el Prototipo 1 y el resto de pruebas de este prototipo, la interfaz gráfica se exportó como una aplicación, para poder usarla de una forma más universal.

Al abrir la aplicación de la interfaz gráfica, se muestran dos botones: uno para la conexión

por WiFi y otro para la conexión por cable USB. Esta ventana de inicio se puede ver en la Figura 60. Cuando se presiona el botón “Conectar por medio de WiFi”, la aplicación intenta conectarse con el módulo ESP8266. Si la conexión no es exitosa, se muestra una ventana de error como la de la Figura 61.

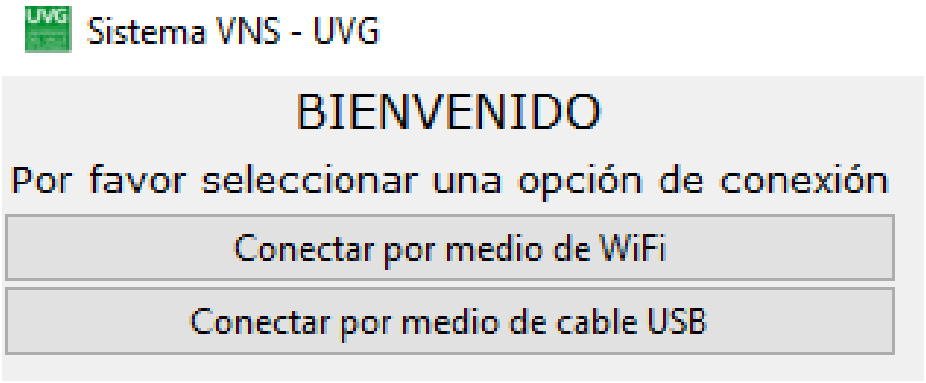


Figura 60: Ventana de inicio de la interfaz gráfica.

La ventana de error (Figura 61), con título “ERROR: SIN WIFI”, muestra el texto “No se pudo conectar con ESP8266 por medio de WiFi. Conectar cable USB”. Esta ventana tiene dos botones. Uno dice “Por favor, seleccionar la opción para conexión por cable USB”, y lleva al usuario a la ventana de inicio para que pueda seleccionar la opción de conexión por cable USB. El otro botón lee “Salir de la aplicación”,y permite al usuario cerrar completamente la aplicación si así lo desea. En el Monitor Serial de Arduino, se puede observar el estado de conexión a la red del ESP8266. Cuando falla, se muestra el mensaje de la Figura 62. En esta figura, ARRIS-BD32 es el nombre de la red con la que se trabajó a la hora de hacer las pruebas.

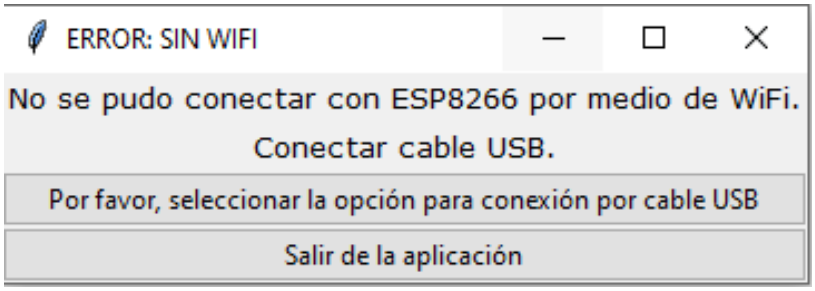


Figura 61: Ventana de error: No se pudo conectar por medio de WiFi.

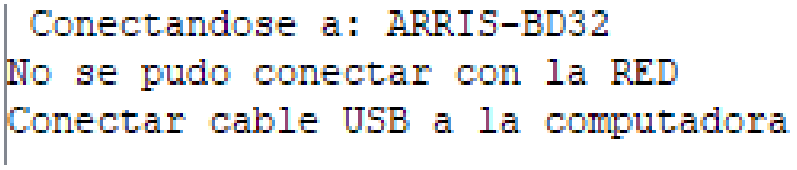


Figura 62: Estado de conexión del ESP8266: Sin éxito.

Al regresar a la ventana de inicio de la Figura 60, se puede presionar el segundo botón, “Conectar por medio de cable USB” para conectar la aplicación con el módulo ESP8266 por

medio de cable USB y así poder enviar los parámetros por comunicación serial. Al presionar el botón, se intenta comunicar con el módulo y si la comunicación fue exitosa, se muestra la ventana de la Figura 63. Esta ventana tiene un botón para ir a la ventana de Selección y Envío de Parámetros como la que se muestra en la Figura 65. El botón "ENVIAR", manda los parámetros seleccionados al módulo por comunicación serial.

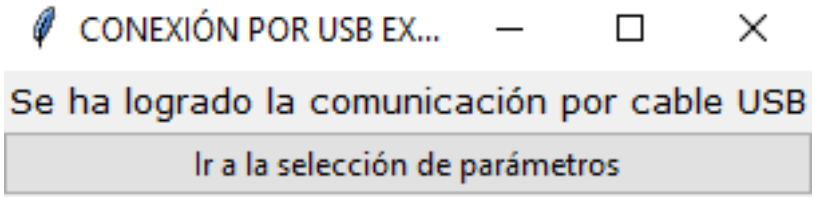


Figura 63: Ventana: Conexión por cable USB exitosa.

Por otro lado, si la conexión al módulo ESP8266 por medio de WiFi fue exitosa, se muestra la ventana de la Figura 64. Esta ventana muestra le mensaje "Se ha logrado la comunicación por WiFi" y tiene un botón que lee "Ir a la selección de parámetros". Este botón lleva al usuario a la ventana en la que se pueden seleccionar los parámetros de estimulación y enviarlos al controlador de la varilla por medio de WiFi. Esta ventana se muestra en la Figura 65, que es idéntica a la ventana mostrada para programar los parámetros por medio de comunicación serial.

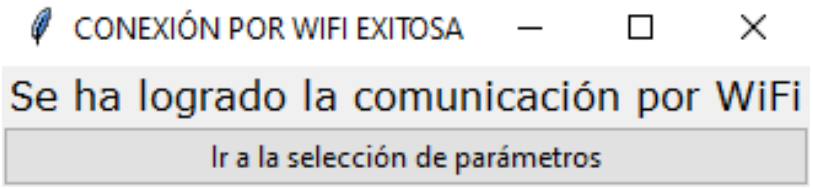


Figura 64: Ventana: Conexión por WiFi exitosa.

En la Figura 65, se muestran los parámetros que se desean enviar. Cada parámetro tiene una lista desplegable que, al presionarla, muestra las diferentes opciones que se pueden programar a cada parámetro. Luego de haber decidido los parámetros a enviar, se debe presionar el botón "ENVIAR" para que lleguen al controlador de la varilla programadora. Las opciones de programación de parámetros son las mismas que las usadas en el Prototipo 1.

Si se presiona el botón de "CERRAR APP", la aplicación se cierra de manera completa, cerrando todos los puertos y sockets para poder conectarse sin problemas al abrir la aplicación de nuevo.

Luego de que el módulo ESP8266 recibe los parámetros, ya sea por WiFi o por comunicación serial, los envía por medio del módulo transmisor RF al módulo receptor RF. Al igual que en las pruebas anteriores, el Arduino Uno recibe los parámetros y los interpreta para poder modificarlos. Los parámetros resultantes se muestran el Monitor Serial de Arduino para verificar que se hayan recibido de la manera correcta. En la Figura 66, se muestran los parámetros recibidos para dos corridas. La primera corrida con modo estimulación, ancho de pulso de 250 μs, tiempo de estimulación de 30 s, 490.20 Hz de frecuencia y un tiempo de apagado de 5 min. En la segunda corrida se utilizó el modo de estimulación, 500 μ de

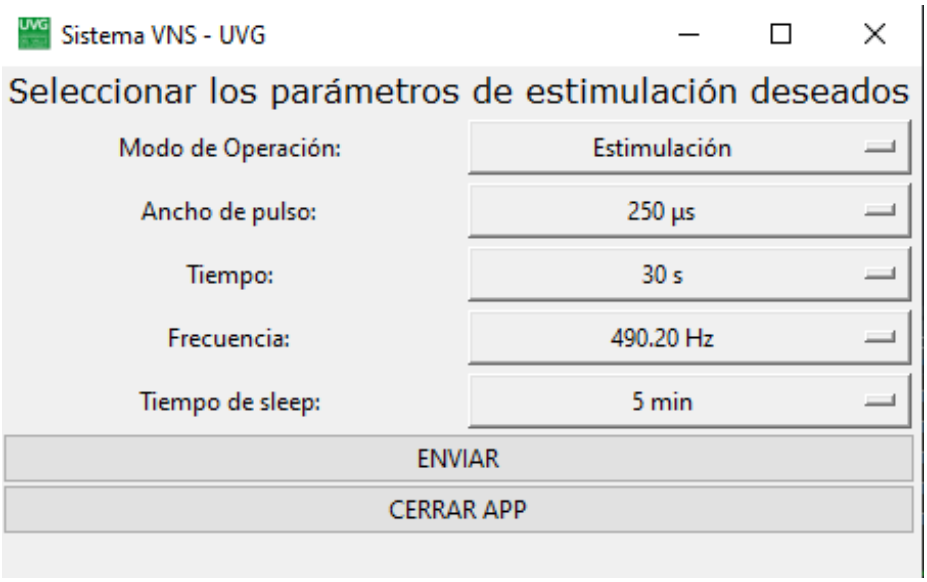


Figura 65: Ventana de selección y envío de parámetros.

ancho de pulso, 60 segundos de tiempo activado, 31372.55 Hz de frecuencia y sin tiempo de apagado.

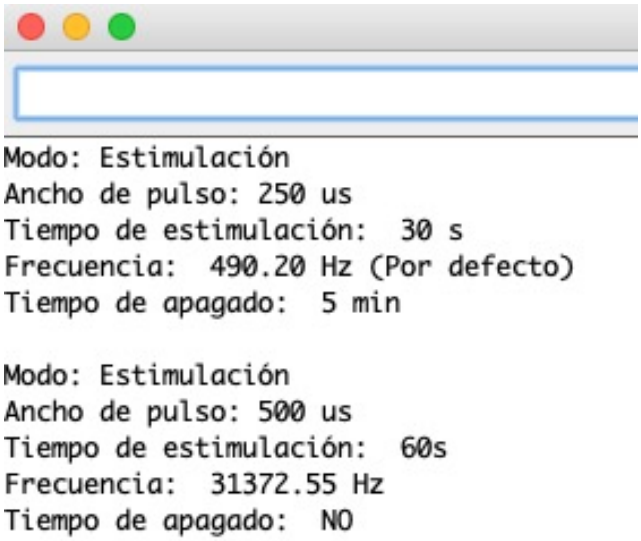


Figura 66: Recepción correcta de parámetros.

8.4. Comunicación entre Módulos RF por medio de un tejido carnoso

Es muy importante que los módulos RF se logren comunicar entre ellos a través de un tejido carnoso. El módulo de estimulación con el receptor RF estará implantado dentro del

cuerpo de la persona, por lo que es necesario poder programar los parámetros a través de la piel humana.

Para probar que los módulos RF se lograran comunicar a través de un tejido carnoso, que simula ser la piel humana, se utilizó carne de pollo. El módulo receptor se enrolló en carne de pollo, añadiendo plástico para evitar dañar el componente, y el módulo transmisor se colocó encima de la carne de pollo para poder comunicar los parámetros de estimulación. En la Figura 67 se muestra el módulo receptor dentro de la carne de pollo, y en la Figura 68 se muestra al módulo transmisor colocado sobre la carne de pollo, listo para comunicar los parámetros de estimulación al módulo receptor que está enrollado dentro de la carne de pollo.

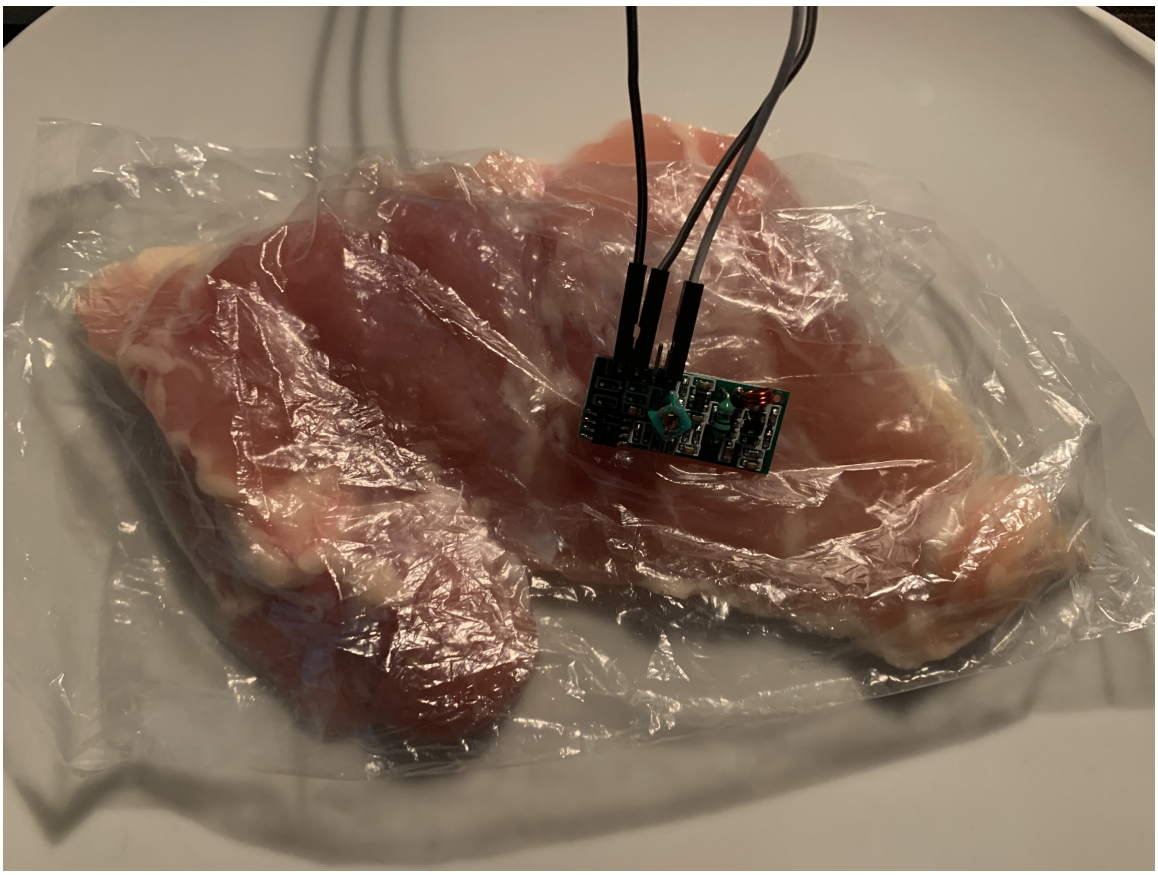


Figura 67: Módulo receptor colocado dentro de carne de pollo.

De esta manera, se comprueba que los módulos RF utilizados para este proyecto, son capaces de comunicarse entre sí a través de un tejido carnoso. Por lo tanto, estos componentes pueden ser utilizados en el módulo de estimulación, que se podrá implantar dentro del cuerpo humano en fases posteriores del proyecto. En el capítulo de Anexos se muestra una Figura en la que se observa el sistema completo utilizando la carne de pollo. Además, en el mismo capítulo, se encuentra el link a un video en el que se muestra el comportamiento completo del sistema. En la Figura 69, se muestra la recepción correcta de parámetros en el módulo receptor cuando se programan los parámetros a través de un tejido carnoso.

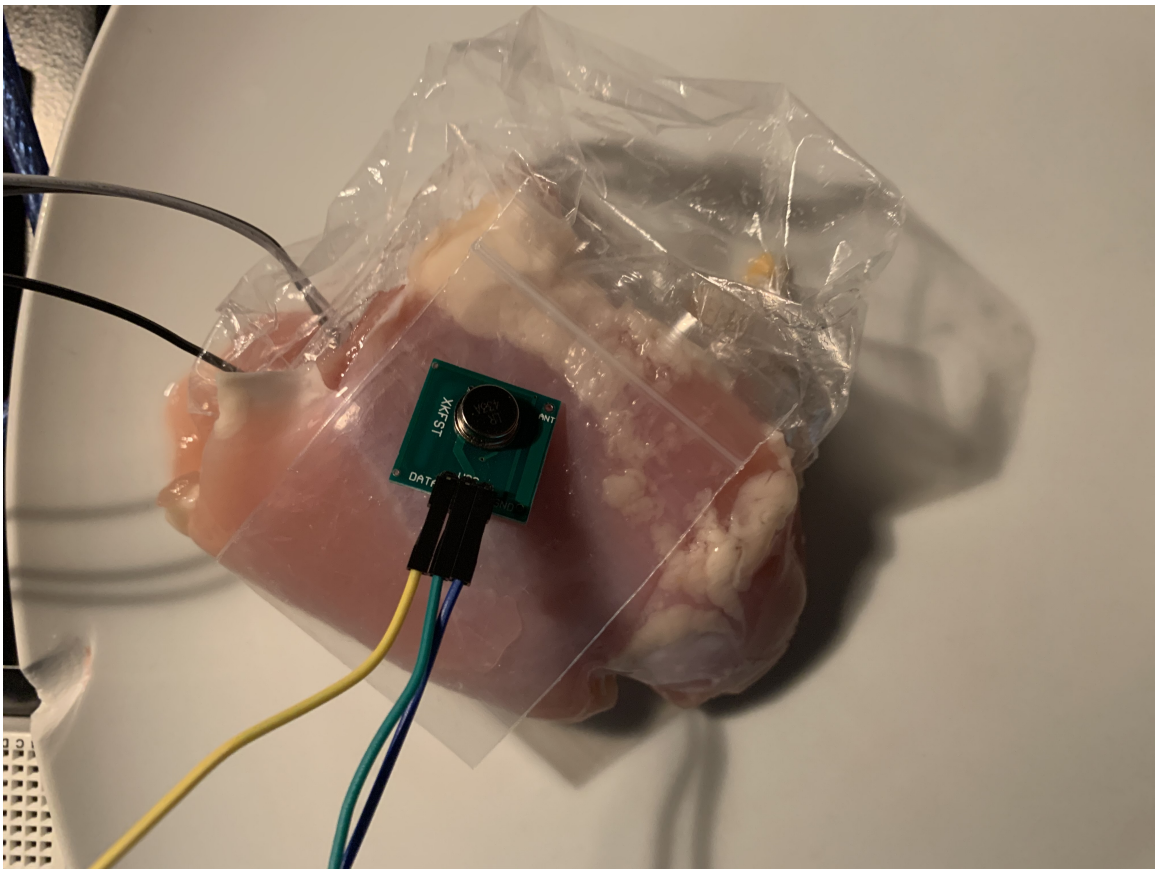


Figura 68: Módulo transmisor listo para comunicar parámetros a través de la carne de pollo.

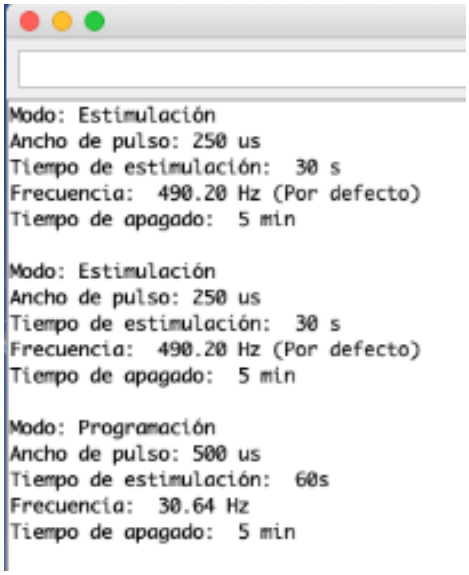


Figura 69: Parámetros recibidos cuando los módulos RF se comunican a través de un tejido carnoso.

8.5. Uso de una pantalla LCD en la varilla programadora

El módulo `NodeMcu`, al conectarse a la red, se le asigna una dirección IP dinámica. Esta dirección puede variar cada determinado tiempo, dependiendo de cambios en la red. Debido a esto, si esta dirección IP no se conoce, es posible que no se pueda conectar el software de programación a la varilla programadora.

Para solucionar este problema, se utiliza una pantalla LCD. Esta pantalla se conecta el controlador de la varilla programadora. Cuando el módulo WiFi se conecta a la red, se muestra la dirección IP asignada en la pantalla LCD. Cuando se selecciona la opción para conectarse por WiFi en el software de programación, se muestra una pantalla en la que se puede ingresar la dirección IP mostrada en la pantalla LCD, Figura 70. Si es la correcta, se va a la selección de parámetros.

Si la varilla programadora no se puede conectar a la red, se muestra un mensaje en la pantalla LCD que indica la falla en la conexión. Esto sirve para que el usuario pueda seleccionar la opción de conexión por USB al ingresar al software de programación.

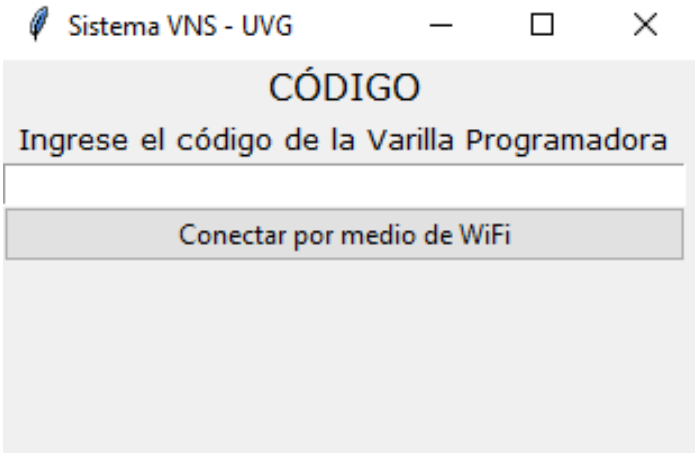


Figura 70: Pantalla para ingresar la dirección IP en el software de programación.

CAPÍTULO 9

Prototipo 3

El Prototipo 3 consistió en realizar lo logrado en el Prototipo 2, pero utilizando módulos RF diferentes. En lugar de utilizar los módulos de 433 MHz, se utilizaron los módulos `nRF24L01+`. Estos módulos, a diferencia de los utilizados en la Fase I del proyecto, no necesitan de una antena externa para transmitir los datos, ya que cuentan con una antena integrada en la placa. Una de las ventajas principales de los módulos `nRF24L01+` es su funcionalidad como transceptores. Los módulos pueden transmitir y recibir datos, a diferencia de los módulos de 433 MHz que solo funcionan como transmisor o como receptor.

Los módulos `nRF24L01+` trabajan a una frecuencia de 2.4 GHz, la cual es una frecuencia ISM (*Industrial, Scientific and Medical*) global, lo que indica que se puede usar de forma libre para motivos médicos en todo el mundo, algo que es muy útil para una aplicación como el Sistema VNS. Los módulos RF se comunican con el controlador de la varilla programadora y el controlador del módulo de estimulación por medio de SPI, por lo que el `NodeMcu` resulta ser una buena elección de controlador, ya que tiene pines diseñados específicamente para el uso de SPI.

Para este prototipo, como ya fue mencionado, se utilizó el `NodeMcu` como controlador de la varilla programadora. Se usó el mismo software de programación (GUI en Python) que en el Prototipo 2, para la selección y carga de parámetros de estimulación a la varilla. La varilla, al recibir los parámetros, los envía por SPI al módulo `nRF24L01+` emisor, que los transmite por medio de radiofrecuencia al módulo `nRF24L01+` receptor. Un `Arduino Uno`, que simula ser el controlador del módulo de estimulación, es el encargado de leer los parámetros recibidos por el `nRF24L01+`. En lugar de mostrar los parámetros recibidos en el monitor serial de Arduino, como se hizo en el Prototipo 2, el módulo receptor envía (utilizando su funcionalidad de transmisor) un mensaje de confirmación de recepción a la varilla programadora. La varilla, al recibir la confirmación, envía el mensaje al Software de Programación, quien lo muestra como una ventana de alerta. Esto para notificar al usuario del éxito en la transmisión, sin la necesidad de tener el monitor serial de Arduino abierto en la PC. En la Figura 71 se puede

observar un diagrama que explica el funcionamiento de este prototipo. Los cuadros de color azul representan los componentes de la varilla programadora y los de color rojo representan los componentes del módulo de estimulación, mientras que la línea corinta representa el envío de parámetros y la línea amarilla el mensaje de confirmación de recepción.

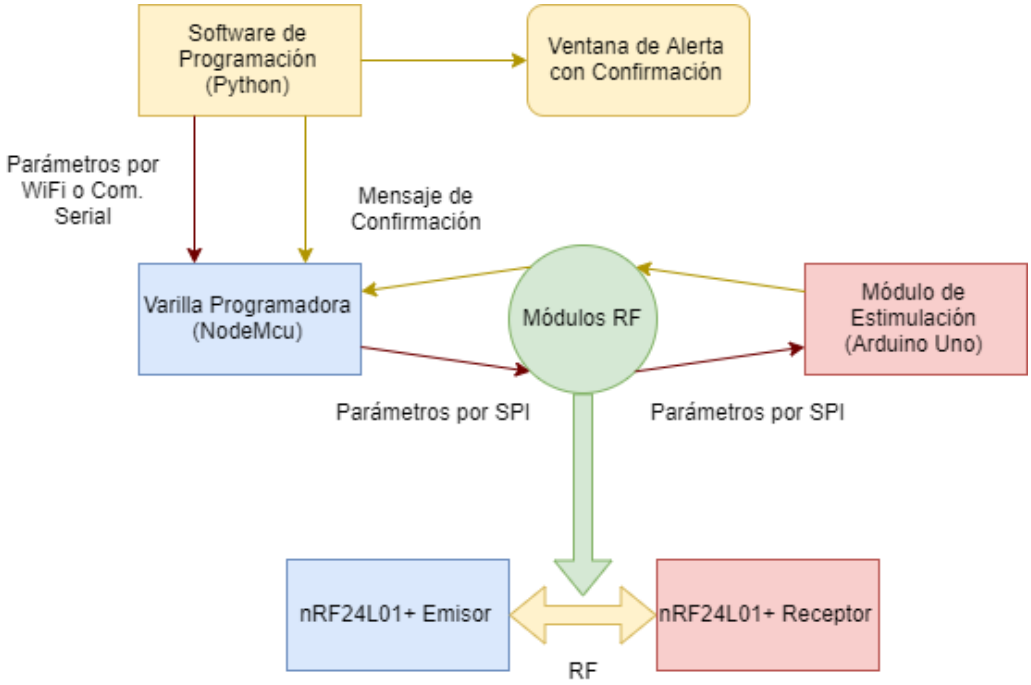


Figura 71: Diagrama de funcionamiento del Prototipo 3.

En este capítulo se muestran las pruebas realizadas para lograr la comunicación inalámbrica exitosa con los módulos `nRF24L01+`. En la Figura 72 se muestra la configuración en físico utilizada.

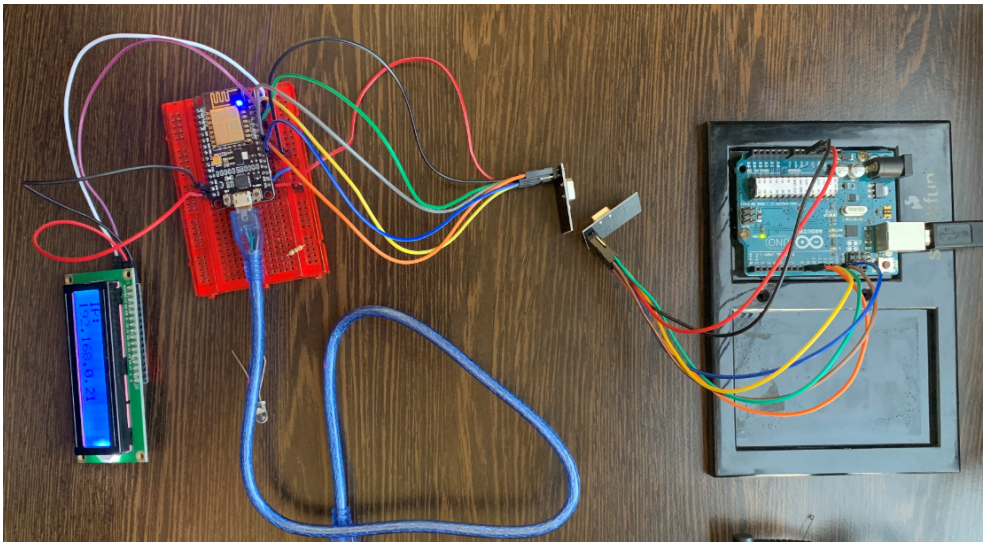


Figura 72: Configuración en físico del Módulo de Programación.

9.1. Prueba de comunicación entre Módulos RF con mensaje de verificación

Para comprobar la comunicación inalámbrica entro los dos módulos RF, se decidió realizar una prueba en la que se envía un *array* de números. Se envía un array, desde el `NodeMcu`, con tres números: 55, 43 y 22, los cuales fueron números elegidos al azar y su valor no tenía relación con el proyecto. El módulo RF receptor recibe el array y despliega los números recibidos en el monitor serial de Arduino. Luego de esto, se envía un mensaje de verificación al módulo emisor de la varilla programadora. Al recibirlo, se despliega un mensaje de éxito en el monitor serial de Arduino en la PC a la que está conectada la varilla programadora. En la Figura 73 se puede observar el monitor serial del módulo receptor, en el cual se muestran los números recibidos, y en la Figura 74 se muestra el monitor serial del módulo emisor, donde se puede observar el mensaje de comunicación exitosa.

```
Primer valor: 55
Segundo valor: 43
Tercer valor: 22

Primer valor: 55
Segundo valor: 43
Tercer valor: 22

Primer valor: 55
Segundo valor: 43
Tercer valor: 22
```

Figura 73: Software de programación.

```
Mensaje recibido: Comunicación Exitosa
Mensaje recibido: Comunicación Exitosa
Mensaje recibido: Comunicación Exitosa
```

Figura 74: Mensaje de éxito mostrado en el monitor serial.

9.2. Unión entre el software de programación y los nuevos Módulos RF

Después de haber comprobado el funcionamiento de los módulos `nRF24L01+` con el `NodeMcu` y el `Arduino Uno`, se procedió a modificar los códigos para trabajar con el software de programación.

En los códigos de la Varilla Programadora y el Módulo de Estimulación solo fue necesario cambiar la sección de código que envía y recibe los parámetros por RF. A ambos códigos se les agregó su contra parte, recibir para la Varilla y enviar para el Módulo de Estimulación, para que la comunicación de dos vías fuera posible.

Con los códigos adaptados, se pudo usar el mismo Software de Programación para comunicar Varilla Programadora con Módulo de Estimulación. En la Figura 75 se muestra el Software de Programación en la pantalla de selección de parámetros del lado izquierdo y el mensaje de confirmación del lado derecho. En la Figura 76 se muestran los parámetros recibidos en el Módulo de Estimulación.

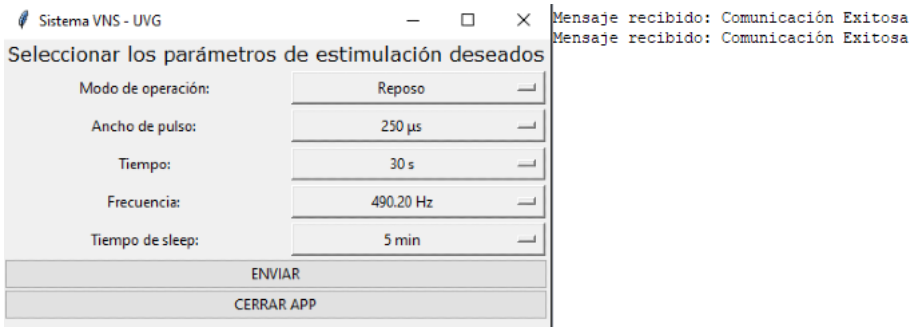


Figura 75: Software de programación y confirmación de recepción exitosa.

```
Modo de Operación: Estimulación
Ancho de Pulso: 250 us
Tiempo de Estimulación: 30 s
Frequencia: 490.20 Hz (Por defecto)
Tiempo de apagado: 5 min

Modo de Operación: Reposo
Ancho de Pulso: 50%
Tiempo de Estimulación: 30 s
Frequencia: 3921.16 Hz
Tiempo de apagado: 5 min
```

Figura 76: Parámetros recibidos en el Módulo de Estimulación.

9.3. Mensajes de confirmación en software de programación

Debido a que, al momento que uno de los médicos encargados use el Sistema VNS, no se contará con el IDE de Arduino para observar las confirmaciones en el monitor serial, es ne-

cesario que el software de programación muestre las confirmaciones. Para esto, se agregaron ventanas de alerta que muestran el estado de la comunicación por RF.

Después presionar el botón "ENVIAR" en la página de selección de parámetros (tanto para comunicación por WiFi como por cable USB), se muestra una ventana de alerta con dos posibles textos. Si la comunicación entre Varilla Programadora y Módulo de Estimulación fue exitosa, se muestra el mensaje de la Figura 77. En esta ventana se muestra un mensaje que confirma que el Módulo de Estimulación ha recibido los parámetros de estimulación de forma correcta.

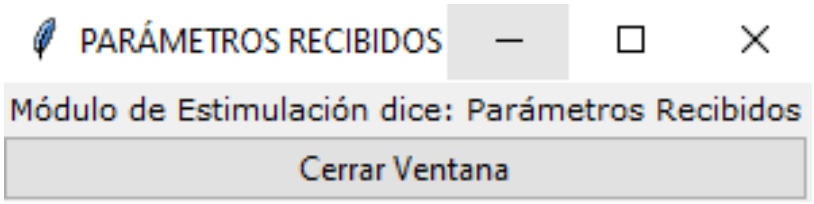


Figura 77: Ventana con confirmación de recepción de parámetros.

En caso de que haya un error, se muestra la ventana de la Figura 78, en la que se muestra un mensaje que indica que el Módulo de Estimulación no recibió los parámetros de estimulación. Se debe revisar la conexión del `Arduino Uno` con el `nRF24L01+`.

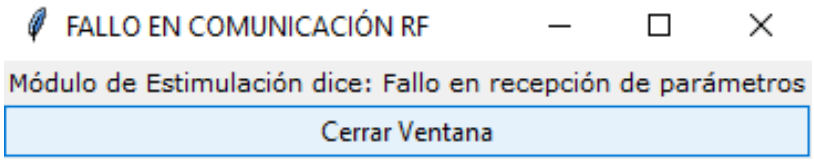


Figura 78: Ventana que indica fallo en la recepción de parámetros.

Ambas ventanas cuentan con un botón de "Cerrar ventana" que regresa al usuario a la selección de parámetros. Con estos mensajes mostrados en el Software de Programación, se eliminó la necesidad del monitor serial de Arduino para observar la confirmación de recepción de parámetros.

9.4. Programación inalámbrica segura

Para garantizar que la programación inalámbrica fuera segura, se decidió realizar 10 envíos consecutivos de distintos parámetros, tanto por WiFi como por cable USB. Si los 10 envíos recibían la confirmación de recepción y los 10 sets de parámetros eran recibidos de manera correcta (esto observado los parámetros en el monitor serial del Módulo de Estimulación), se confirmaba que la programación inalámbrica era segura. Los parámetros a utilizar en cada envío se decidieron antes de iniciar la programación. Los parámetros seleccionados se pueden observar en los Cuadros 5, 6 y 7.

Parámetro	Valor 1	Valor 2	Valor 3	Valor 4
Modo de operación	Estimulación	Estimulación	Reposo	Reposo
Ancho de pulso	250 μs	250 μs	250 μs	500 μs
Frecuencia	490.20 Hz	245.10 Hz	245.10 Hz	245.10 Hz
Tiempo de estimulación	30 s	30 s	60 s	60 s
Tiempo de apagado	5 min	5 min	5 min	NO

Cuadro 5: Parámetros a enviar 1 a 4.

Parámetro	Valor 5	Valor 6	Valor 7	Valor 8
Modo de operación	Estimulación	Programación	Programación	Reposo
Ancho de pulso	500 μs	250 μs	250 μs	500 μs
Frecuencia	31372.55 Hz	31372.55 Hz	30.64 Hz	122.50 Hz
Tiempo de estimulación	60 s	30 s	60 s	60 s
Tiempo de apagado	NO	5 min	5 min	NO

Cuadro 6: Parámetros a enviar 5 a 8.

Parámetro	Valor 9	Valor 10
Modo de operación	Reposo	Estimulación
Ancho de pulso	500 μs	250 μs
Frecuencia	3921.16 Hz	490.20 Hz
Tiempo de estimulación	60 s	30 s
Tiempo de apagado	NO	5 min

Cuadro 7: Parámetros a enviar 9 y 10.

9.4.1. Comunicación por WiFi

En primer lugar, se decidió comprobar si la programación inalámbrica por WiFi era segura. En las Figuras 79 y 80 se muestran las 10 recepciones en el monitor serial.

```
Recepción No.1
Modo de Operación: Estimulación
Ancho de Pulso: 250 us
Tiempo de Estimulación: 30 s
Frequencia: 490.20 Hz (Por defecto)
Tiempo de apagado: 5 min

Recepción No.2
Modo de Operación: Estimulación
Ancho de Pulso: 250 us
Tiempo de Estimulación: 30 s
Frequencia: 245.10 Hz
Tiempo de apagado: 5 min
```

Figura 79: Recepciones 1 y 2 por WiFi.

```
Recepción No.3
Modo de Operación: Reposo
Ancho de Pulso: 250 us
Tiempo de Estimulación: 60 s
Frequencia: 245.10 Hz
Tiempo de apagado: 5 min

Recepción No.4
Modo de Operación: Reposo
Ancho de Pulso: 500 us
Tiempo de Estimulación: 60 s
Frequencia: 245.10 Hz
Tiempo de apagado: NO

Recepción No.5
Modo de Operación: Estimulación
Ancho de Pulso: 500 us
Tiempo de Estimulación: 60 s
Frequencia: 31372.55 Hz
Tiempo de apagado: NO

Recepción No.6
Modo de Operación: Programación
Ancho de Pulso: 250 us
Tiempo de Estimulación: 30 s
Frequencia: 31372.55 Hz
Tiempo de apagado: 5 min

Recepción No.7
Modo de Operación: Programación
Ancho de Pulso: 250 us
Tiempo de Estimulación: 60 s
Frequencia: 30.64
Tiempo de apagado: 5 min

Recepción No.8
Modo de Operación: Reposo
Ancho de Pulso: 500 us
Tiempo de Estimulación: 60 s
Frequencia: 122.50 Hz
Tiempo de apagado: NO

Recepción No.9
Modo de Operación: Reposo
Ancho de Pulso: 500 us
Tiempo de Estimulación: 60 s
Frequencia: 3921.16 Hz
Tiempo de apagado: NO

Recepción No.10
Modo de Operación: Estimulación
Ancho de Pulso: 250 us
Tiempo de Estimulación: 30 s
Frequencia: 490.20 Hz (Por defecto)
Tiempo de apagado: 5 min
```

Figura 80: Recepciones 3 a 10 por WiFi.

Como se puede observar, las recepciones fueron correctas. Esto verificó que la programación al utilizar WiFi era la deseada.

9.4.2. Comunicación por cable USB

Luego de verificar que la comunicación utilizando WiFi era segura, se hizo lo mismo con la comunicación por cable USB. Las Figuras 81 y 82 muestran las 10 recepciones obtenidas. 3

```
Por cable USB
Recepción No.1
Modo de Operación: Estimulación
Ancho de Pulso: 250 us
Tiempo de Estimulación: 30 s
Frequencia: 490.20 Hz (Por defecto)
Tiempo de apagado: 5 min

Por cable USB
Recepción No.2
Modo de Operación: Estimulación
Ancho de Pulso: 250 us
Tiempo de Estimulación: 30 s
Frequencia: 245.10 Hz
Tiempo de apagado: 5 min
```

Figura 81: Recepciones 1 y 2 por cable USB.

```
Por cable USB
Recepción No.3
Modo de Operación: Reposo
Ancho de Pulso: 250 us
Tiempo de Estimulación: 60 s
Frequencia: 245.10 Hz
Tiempo de apagado: 5 min

Por cable USB
Recepción No.4
Modo de Operación: Reposo
Ancho de Pulso: 500 us
Tiempo de Estimulación: 60 s
Frequencia: 245.10 Hz
Tiempo de apagado: NO

Por cable USB
Recepción No.5
Modo de Operación: Estimulación
Ancho de Pulso: 500 us
Tiempo de Estimulación: 60 s
Frequencia: 31372.55 Hz
Tiempo de apagado: NO

Por cable USB
Recepción No.6
Modo de Operación: Programación
Ancho de Pulso: 250 us
Tiempo de Estimulación: 30 s
Frequencia: 31372.55 Hz
Tiempo de apagado: 5 min
```

```
Por cable USB
Recepción No.7
Modo de Operación: Programación
Ancho de Pulso: 250 us
Tiempo de Estimulación: 60 s
Frequencia: 30.64
Tiempo de apagado: 5 min

Por cable USB
Recepción No.8
Modo de Operación: Reposo
Ancho de Pulso: 500 us
Tiempo de Estimulación: 60 s
Frequencia: 122.50 Hz
Tiempo de apagado: NO

Por cable USB
Recepción No.9
Modo de Operación: Reposo
Ancho de Pulso: 500 us
Tiempo de Estimulación: 60 s
Frequencia: 3921.16 Hz
Tiempo de apagado: NO

Por cable USB
Recepción No.10
Modo de Operación: Estimulación
Ancho de Pulso: 250 us
Tiempo de Estimulación: 30 s
Frequencia: 490.20 Hz (Por defecto)
Tiempo de apagado: 5 min
```

Figura 82: Recepciones 3 a 10 por cable USB.

Con esto se comprobó que la recepción de parámetros era segura tanto al usar WiFi o comunicación serial (por cable USB).

9.5. Programación inalámbrica eficiente

9.5.1. Bajo consumo

En el Capítulo 10 se puede observar la configuración seleccionada para reducir el consumo de corriente del módulo `nRF24L01+` al mínimo. Con la configuración seleccionada, se reduce el consumo de corriente a 7.0 mA en la transmisión y 12.6 mA en la recepción. El módulo RF, al no estar transmitiendo o recibiendo datos, entra en un modo inactivo conocido como Standby-I, en el cual solo se consumen 26 μA. De esta forma, el consumo de corriente de el módulo se hace lo más eficiente posible.

Sin embargo, existe un gran problema con el consumo de corriente de este módulo. El módulo RF puede entrar en un modo inactivo llamado Power Down, el cual solo consume 900 nA. El problema es que en este modo no se puede recibir datos. Entonces, no hay una manera en la cual el módulo se pueda "despertar" de este modo PowerDown sin una señal externa.

Explicando un poco más a profundidad esto, la explicación se enfoca en que el módulo de estimulación estará implantado en el cuerpo. Debido a que el módulo estará implantado, no existe una forma en la que se pueda enviar una señal externa para despertar al módulo RF en cualquier momento. Si el controlador del generador de pulsos mandara esta señal al módulo, solo podría hacerlo cada cierto tiempo y la programación de parámetros sería muy complicada porque solo existirían ventanas de tiempo para la programación, las cuales no se podrían conocer de manera sencilla.

Por esta razón, el módulo receptor siempre tiene que estar en el modo de RX, siempre debe estar listo para recibir parámetros de estimulación. El problema de esto es que el consumo, aún cuando fue optimizado, sigue siendo bastante alto. Una batería de 6000 mAh solo duraría alrededor de 476.2 horas o 20 días si el módulo siempre consume 12.6 mA.

Es necesario buscar una manera de optimizar este consumo mucho más o buscar un nuevo módulo RF que permita recibir datos en el modo de consumo más bajo, como el Power Down. Una posible solución se encuentra en el capítulo 11 - Trabajo a Futuro.

Cambio de parámetro

Debido a este problema mencionado, se elimina el parámetro de "Modo de Operación" utilizado en la Fase I. Este parámetro servía para indicar el modo de operación al módulo de estimulación. Por ejemplo, si el valor era "Estimulación" el módulo estimulaba con pulsos eléctricos. Si era "Programación" el módulo estaba listo para recibir los parámetros por RF. Sin embargo, como fue mencionado, es necesario mantener al módulo siempre en recepción, es decir en modo de Programación. Por esta razón, se eliminó este parámetro.

Para iniciar a parecerse a los modelos comerciales, se añadió el parámetro de "Corriente de Estimulación" con valores de 0.25, 0.5 y 0.75 mA, valores comunes en los modelos comerciales.

9.5.2. Tiempo de programación

La programación de parámetros debe ser eficiente, debe ocurrir muy rápido. Una programación rápida podría significar la diferencia entre abortar una convulsión en camino o dejar que esta ocurra, afectando al paciente.

Se realizaron 10 mediciones del tiempo de programación, las cuales se muestran en el Cuadro 8. El promedio de estas 10 mediciones resultó ser de 11.350 ms, un tiempo bastante bajo. Con esto se concluye que la velocidad de programación es rápida y eficiente.

Medición	Tiempo en μs	Tiempo en ms
1	9831	9.831
2	10658	10.658
3	13365	13.365
4	13925	13.295
5	11468	11.468
6	9785	9.785
7	13366	13.366
8	10496	10.496
9	10970	10.970
10	10261	10.261
Promedio	**11349.5**	**11.350**

Cuadro 8: Mediciones de tiempo de programación realizadas.

9.6. Nuevas funcionalidades del software de programación

Para hacer que el software de programación fuera más amigable, se agregaron algunas funcionalidades para ayudar al usuario a guiarse de una mejor manera.

9.6.1. Indicación de estado de conexión a la red del software de programación

Al seleccionar la opción para conectarse con la Varilla Programadora por medio de WiFi, se muestra la pantalla en la que se ingresa el código (dirección IP) de la Varilla. En esta pantalla ahora se muestra un mensaje que indica el estado de conexión a la red del Software de Programación. Si el mensaje es verde, Figura 83, quiere decir que la conexión a la red es exitosa, sí se puede utilizar WiFi para comunicarse con la Varilla Programadora. Si el mensaje es de color rojo, como el de la Figura 84, quiere decir que el Software de Programación no se pudo conectar a la red WiFi y, por lo tanto, no se puede comunicar con la Varilla Programadora por WiFi.

En esta página, debajo del estado de la conexión a la red, se muestra un mensaje que lee “Si el color es rojo, por favor seleccionar opción para conectar por cable USB”, indicando al usuario lo que debe realizar en caso de una conexión fallida. Debajo de este texto se encuentra un botón para regresar al Menú Principal, y así poder entablar la comunicación con la Varilla Programadora por medio de cable USB.

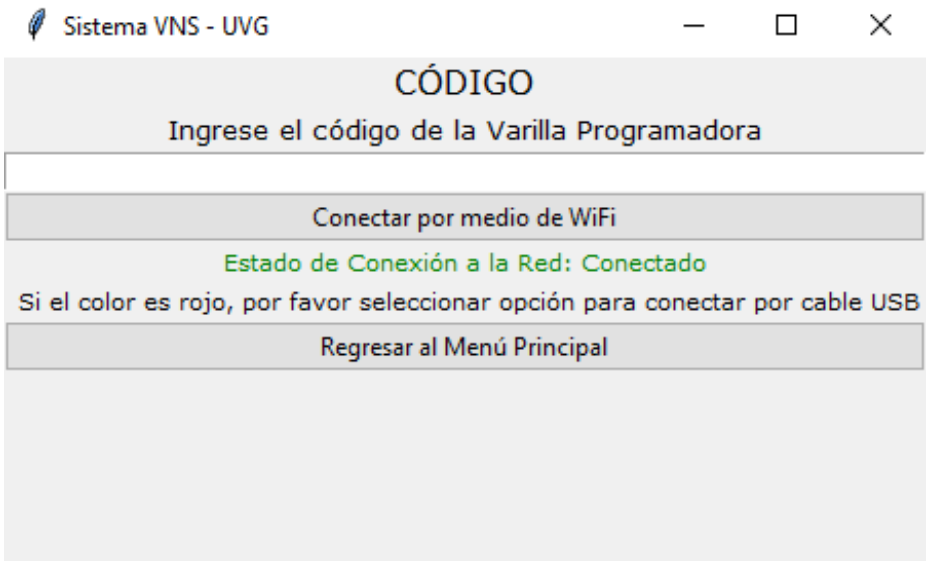


Figura 83: Indicación de conexión a la red exitosa.

Figura 84: Indicación de conexión a la red fallida.

9.6.2. Mensaje de conexión por WiFi fallida

En caso que no se pueda entablar la conexión con la Varilla Programadora luego de ingresar el código, una ventana de alerta salta y le indica al usuario que no se pudo realizar la conexión con la Varilla por medio de WiFi. En esta ventana, que se puede observar en la Figura 85, se muestran botones con tres diferentes opciones: regresar al menú principal, intentar ingresar el código de nuevo o salir de la aplicación.

El botón “Regresar al Menú Principal” sirve para regresar al menú principal y así poder

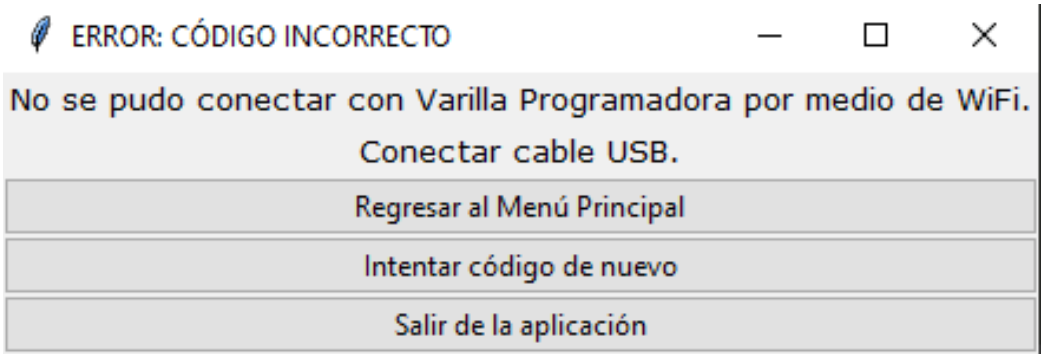


Figura 85: Ventana que indica error en el código de la varilla programadora.

intentar la comunicación por medio de cable USB. A veces se puede pasar por alto que el Software de Programación (la PC) no se pudo conectar a la red y por lo tanto es importante mostrar una alerta de error que permita regresar directamente al menú principal.

El botón "Intentar código de nuevo" permite al usuario ingresar de nuevo el código, ya que es fácil cometer un error mientras éste se ingresa. Y el botón "Salir de la aplicación" sirve como una manera directa para cerrar la aplicación completamente.

9.6.3. Mensaje de error en conexión por cable USB

En caso de no detectar la Varilla Programadora conectado en el momento en el que se intenta establecer la conexión, se muestra el mensaje de error de la Figura 86.

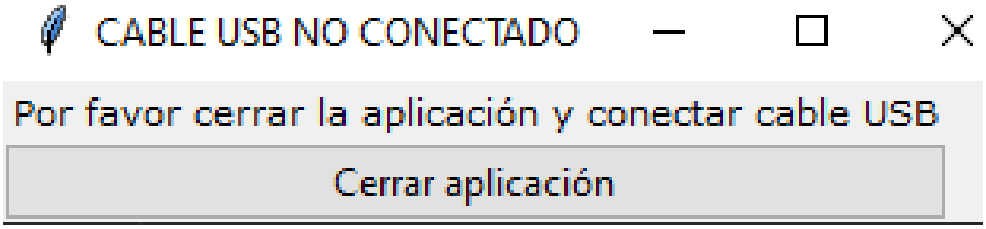


Figura 86: Ventana que indica varilla programadora no conectada.

El mensaje de error indica al usuario que se debe cerrar la aplicación y conectar el cable USB para que la Varilla Programadora sea detectada de forma correcta. En la ventana de error se muestra un botón que permite cerrar la aplicación directamente. Luego de esto se debe conectar

9.7. Comparación de tamaño entre Fase I y Fase II

Las dimensiones del módulo utilizado en la Fase I, el módulo `SPSGRFC-868`, se pueden observar en la Figura 87. El módulo RF tiene dimensiones 13.5 × 11 mm, resultando en un área de 148.50 mm^2. A este módulo se le debe conectar externa para la transmisión de

datos. Esta antena tiene dimensiones aproximadas (ya que la antena no es un rectángulo) de 7.8 × 84 mm, con un área de 655.20 mm^2. Sumando las dos áreas, se obtiene un área total de 803.70 mm^2. Las dimensiones de la antena se pueden observar en la Figura 88.

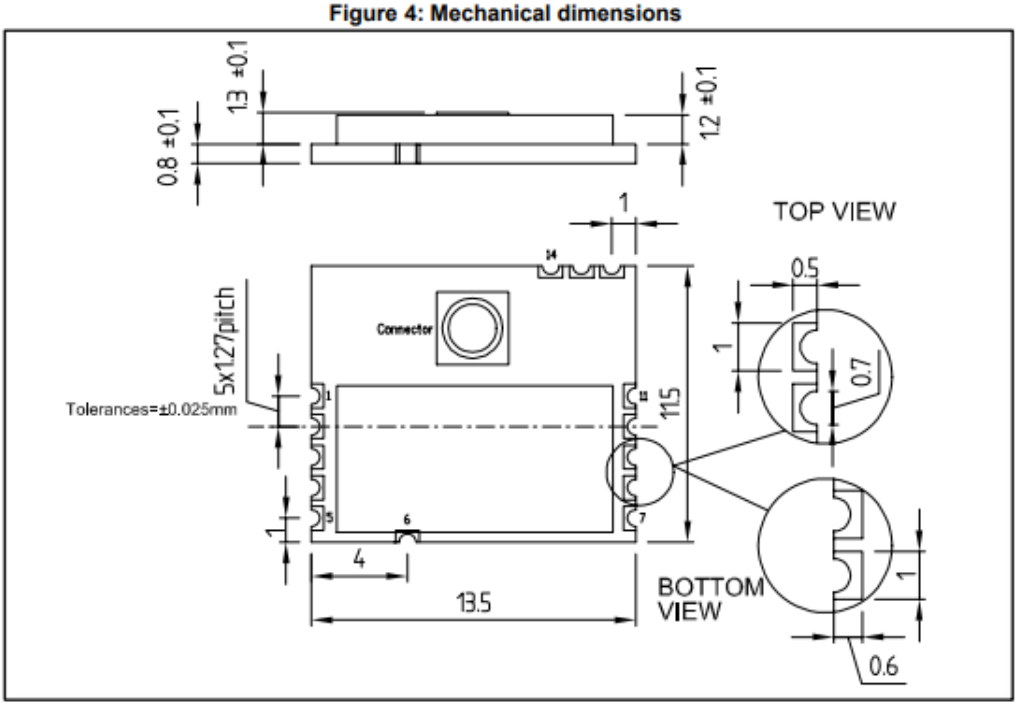


Figura 87: Dimensiones del módulo RF utilizado en Fase I [54].

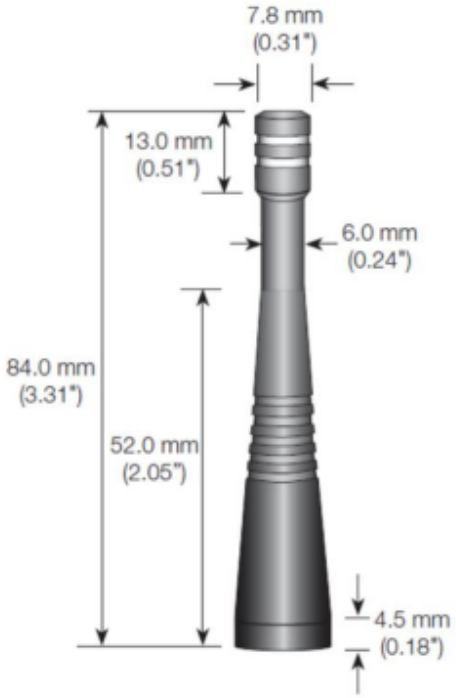


Figura 88: Dimensiones de la antena utilizada con el módulo RF utilizado en Fase I [54].

El módulo RF utilizado en la Fase II, el `nRF24L01+`, tiene dimensiones 29.05 × 15.24 mm, resultando en un área de 442.72 mm^2. Estas dimensiones se pueden observar en la Figura 89. Aunque el módulo RF de la Fase I es más pequeño que el utilizado en la Fase II, el módulo de la Fase I requiere de una antena externa, lo que aumenta la dimensión total del módulo de comunicación inalámbrica. La antena ocuparía demasiado espacio en una placa de 2500 mm^2. Tomando en cuenta las áreas totales, se logró reducir el tamaño del módulo RF en un 45 %.

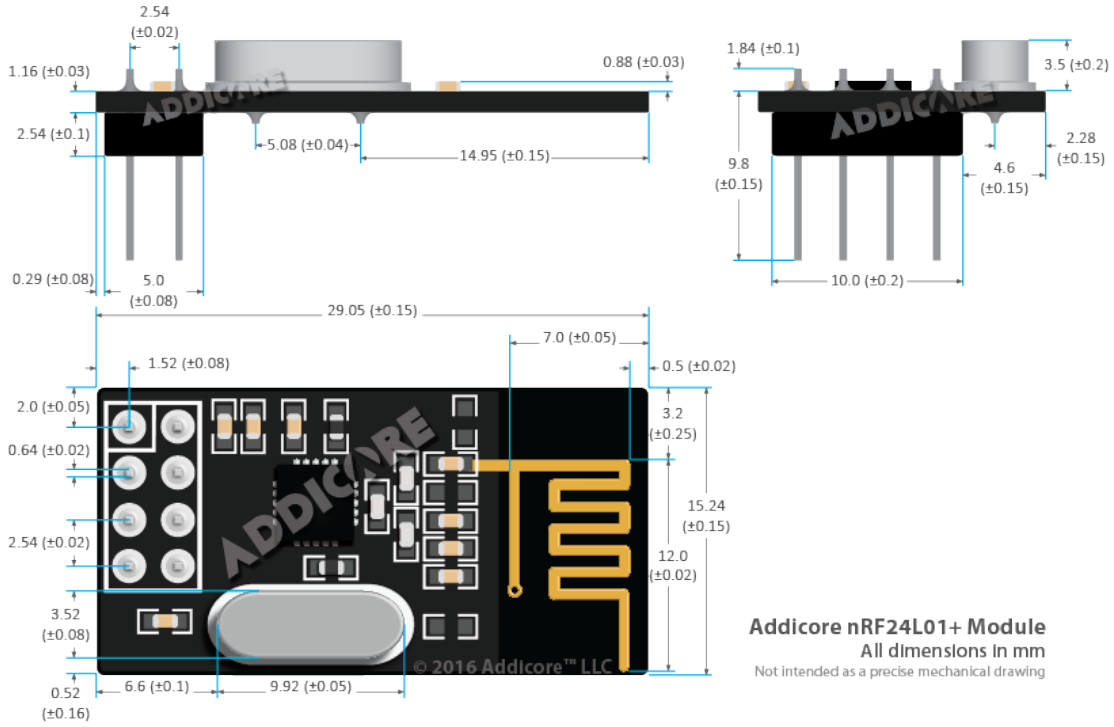


Figura 89: Dimensiones del módulo RF utilizado en Fase II [54]

9.8. Comparación de precio entre este proyecto y los modelos comerciales

Como ya fue mencionado, el Sistema VNS comercial puede costar hasta 10,000 USD sin tomar en cuenta la operación. Este es un precio demasiado elevado para familias comunes en Guatemala. En este proyecto era necesario disminuir el precio de manera considerable.

El precio de los componentes utilizados en el Prototipo 3 del módulo de programación se puede observar en el Cuadro 9. El precio total del módulo de programación, incluyendo el controlador utilizado en el módulo de estimulación, el ATSAMD21, es de GTQ 112.80 o USD 14.40. Este precio total es menos del 1 % del precio del Sistema VNS comercial.

Componente	Cantidad	Precio GTQ	Precio USD
NodeMCU ESP8266	1	70.00	9.00
nRF24L01+	2	18.80	2.40
Software de Programación	1	0.00	0.00
ATSAMD21	1	24.00	3.00
Total	**5**	**112.80**	**14.40**

Cuadro 9: Precio de los componentes utilizados en el módulo de programación.

Aunque este precio no es final, es una buena base hacia una alternativa de bajo costo.

CAPÍTULO 10

Configuración del Módulo nRF24L01+ y protocolo de transmisión

En este capítulo se habla sobre la configuración del módulo `nRF24L01+` utilizada para el Prototipo 3. La configuración utilizada se enfoca en reducir el consumo de corriente del módulo al menor valor posible. Para configurar el módulo y utilizarlo, se utilizó la librería `RF24.h` que optimiza el uso del módulo RF. La documentación se puede leer en [55].

En este capítulo también se menciona el protocolo de comunicación para la transmisión de parámetros desde el software de programación hasta la recepción en el módulo de estimulación.

10.1. Tasa de bits

El módulo RF tiene tres configuraciones de tasa de bits (*bitrate*), las cuales se pueden observar en el Cuadro 10. La corriente consumida en el modo de recepción depende de la tasa de bits. Mientras más baja sea esta tasa, más bajo es el consumo de corriente. Por esta razón, se selecciona la tasa de 250 kbps, ya que solo consuma 12.6 mA y no se requiere una velocidad de procesamiento alta.

Recepción (RX)	Corriente consumida
2 Mbps	13.5 mA
1 Mbps	13.1 mA
250 kbps	12.6 mA

Cuadro 10: Corriente consumida en el modo de recepción.

10.2. Frecuencia de operación

El `nRF24L01+` se puede utilizar en el rango de frecuencias de 2400 MHz a 2525 MHz. Se seleccionó una frecuencia de 2510 MHz, ya que ésta se encuentra arriba de las frecuencias de uso cotidiano como WiFi (2.4 GHz), Bluetooth (2.485 GHz) y algunos microondas que trabajan a 2.45 GHz.

Para seleccionar la frecuencia utilizando la librería ya mencionada, se debe seleccionar uno de los 125 canales disponibles. Ya que se está utilizando una tasa de bits de 250 kbps, cada canal ocupa 1 MHz. Por lo que se seleccionó el canal 110, que corresponde a la frecuencia de 2510 MHz. En el Cuadro 11 se muestra un descripción de los rangos de frecuencia a los que puede trabajar el módulo RF.

Canal	Frecuencia (MHz)	Descripción
0-82	2400-2482	Legal pero ruidoso
83-99	2483-2499	Ilegal
101-119	2501-2519	Legal y limpio
120-125	2520-2525	Usados en defensa

Cuadro 11: Frecuencias del módulo y sus descripciones.

10.3. Potencia de transmisión

Este módulo tiene cuatro configuraciones distintas de potencia de transmisión (potencia del amplificador de potencia, PA) y se pueden observar en el Cuadro 12. Para minimizar el consumo de corriente se seleccionó la opción de -18 dBm, "PA Level: MIN" en la librería. Esta selección de potencia de transmisión reduce la distancia a la que se pueden transmitir los datos al mínimo, pero en esta aplicación no se requiere una distancia grande, por lo que no hay ningún problema con seleccionar esta opción.

Transmisión	Corriente consumida
0 dBm	11.3 mA
-6 dBm	9.0 mA
-12 dBm	7.5 mA
-18 dBm	7.0 mA

Cuadro 12: Consumo de corriente en el modo de transmisión.

10.4. Modulación

Este módulo RF utiliza modulación por desplazamiento de frecuencia gausiana (*Gaussian Frequency Shift Keying*, GFSK). Es un tipo de modulación muy utilizado en aplicaciones de bajo costo, bajo consumo y baja tasa de datos. Este tipo de modulación es ideal para este proyecto.

10.5. Protocolo de transmisión

El protocolo de transmisión utilizado para el módulo de programación está conformado de cinco pasos:

1. Selección de parámetros en software de programación
2. Carga de parámetros a la varilla programadora
3. Bus de datos
4. Transmisión por radiofrecuencia
5. Mensaje de confirmación

El primer paso consiste en la selección de parámetros de estimulación en el software de programación. Al seleccionar los parámetros utilizando las listas desplegables, se asigna a una variable un número que depende de la opción seleccionada. Por ejemplo, si se selecciona la opción de "Corriente de estimulación: 0.5 mA", se asigna el número 2 a la variable llamada corriente.

Luego de asignar a las cinco variables de los parámetros un número correspondiente, estas variables se cargan a la varilla programadora por medio de WiFi o por comunicación serial. En el tercer paso, los números de los parámetros se leen en la varilla programadora y se convierten a entero para mejor manejo. Luego de la conversión, se almacenan en un bus de datos de 6 casillas, un parámetro por cada casilla.

Al terminar de almacenar los valores en el bus de datos, la última casilla del bus de datos se coloca en 1 y se envía por radiofrecuencia al módulo receptor. Este 1 sirve para que, cuando el módulo de estimulación lo lea, sepa que la recepción fue correcta.

Cuando el módulo de estimulación confirma que ha recibido los parámetros de forma correcta, envía de vuelta un 1 para confirmar la recepción. La varilla programadora, al recibir este 1, informa al software de programación que la programación de parámetros fue realizada de forma correcta.

CAPÍTULO 11

Trabajo a futuro

Este capítulo habla sobre el trabajo que se debe realizar a futuro para lograr que el módulo de programación funcione de la mejor manera, y está dividido en dos secciones. La primera sección habla sobre la incorporación del módulo de programación presentado en el Prototipo 3 con el controlador ATSAMD21, el módulo de estimulación utilizado por Gustavo Ordóñez. La segunda sección menciona una solución posible al consumo de corriente excesivo.

11.1. Unión de los dos módulos principales

El Prototipo 3 del módulo de programación utiliza `Arduino Uno` como simulación del controlador del módulo de estimulación. En la siguiente fase de este proyecto, es necesario utilizar el mismo controlador que se usó en el módulo de estimulación desarrollado por Gustavo Ordóñez.

Esto debe ser realizado para poder hacer pruebas más concretas con los prototipos finales y verificar que todo funcione de la manera esperada.

11.2. Solución posible al problema de consumo de corriente

Como fue mencionado anteriormente, el problema del consumo de corriente radica en que el módulo RF receptor siempre tiene que estar listo para recibir datos, lo que mantiene un consumo constante de 12.6 mA.

Una solución a este problema es el uso del imán. El Sistema VNS comercial tiene un

imán que se utiliza para activar un modo bajo demanda, un modo que es activado por el usuario. Este imán se podría utilizar para activar el modo de recepción del módulo RF. El módulo RF puede ser despertado por una señal externa. Si el controlador del módulo de estimulación, que se activa al recibir la señal del imán, envía una señal externa al módulo RF, este podría entrar y salir del modo de recepción cuando se requiera. Esto lograría disminuir el consumo de corriente a los 900 nA del modo Power Down cuando no sea necesario programar parámetros.

CAPÍTULO 12

Conclusiones

Con el trabajo realizado anteriormente, se concluyó lo siguiente:

- Se llevó acabo un prototipo funcional de varilla programadora capaz de programar los parámetros al módulo de estimulación de forma inalámbrica.
- Esta varilla programadora se puede comunicar con el software de programación (interfaz gráfica) de forma inalámbrica, por WiFi, o por cable USB, por medio de comunicación serial.
- Se implementó una comunicación inalámbrica segura, no existen errores en la recepción de parámetros. Además, se envía un mensaje desde el módulo receptor que confirma la recepción de parámetros.
- Se implementó una comunicación inalámbrica eficiente, al reducir el consumo de corriente del módulo en sus modos de transmisión y recepción, y al lograr una comunicación rápida de 11.35 milisegundos en promedio.
- Se diseñó un software de programación (interfaz gráfica) simple y amigable para seleccionar y cargar los parámetros de estimulación a la varilla programadora. Es un software fácil de usar para que el personal médico no tenga problemas al momento de su uso.
- Este software de programación se exportó como una aplicación para un uso más universal. Se puede usar en computadoras con sistema operativo Windows 7 o una versión más reciente, sin la necesidad de tener Python instalado.
- Se utilizaron módulos de comunicación inalámbrica (por radiofrecuencia, RF) un 45 % más pequeños que los usados en la Fase I.
- Estos módulos RF se pueden comunicar a través de un tejido carnoso, carne de pollo, que simula ser la piel humana.

CAPÍTULO 13

Recomendaciones

Para la siguiente fase del proyecto, se recomienda agregar un módulo de energía independiente para la varilla programadora. En esta fase, aunque el software de programación y la varilla programadora se pueden comunicar de forma inalámbrica, es necesario mantener la varilla conectada a una fuente de energía (como la computadora que tiene el software de programación) para poder utilizarla. Agregar un módulo de energía independiente haría mucho mejor el uso de la varilla. Además, se recomienda diseñar un estuche similar al de los modelos comerciales, esto para hacer más cómodo el uso de la varilla, sin tener que preocuparse por la pantalla LCD o el módulo RF.

Si se desea cambiar de controlador de varilla programadora, se recomienda utilizar un controlador que tenga funcionalidad de conexión inalámbrica integrada, así como el NodeMCU ESP8266 tiene funcionalidad WiFi integrada. Es mucho más fácil usar un controlador de este tipo, que integrar un módulo de conexión inalámbrica extra. Además, de esta manera, se reduce el costo de la varilla programadora.

También, si se desea cambiar de módulos de programación inalámbrica (RF), se recomienda buscar módulos que soporten 3.3 VDC y 5 VDC en sus pines I/O. Así, no es necesario incluir un regulador de voltaje al módulo de estimulación o al módulo de programación, lo cual solo aumentaría el tamaño y el costo. Se recomienda que los módulos de programación inalámbrica a utilizar usen una frecuencia diferente a las de uso cotidiano (WiFi, Bluetoth, algunos microondas, etc.) para evitar ruido en la transmisión.

Por último, se recomienda hacer la prueba del tejido carnoso con una carne más parecida a la de los humanos. La carne de cerdo es la que utilizan los estudiantes de medicina para aprender a suturar, por lo cual es una buena opción.

CAPÍTULO 14

Bibliografía

[1] Forbes. (2012). Cyberonics, dirección: https://www.forbes.com/companies/cyberonics/37c73e1e3293 (visitado 10-04-2020).

[2] LivaNova. (). About Us, dirección: https://www.livanova.com/en-US/Home/About-Us.aspx (visitado 10-04-2020).

[3] Cyberonics y V. Therapy. (2007). Implanted Components, dirección: http://dynamic.cyberonics.com/depression/hcp/ForSurgeons/implanted.aspx (visitado 10-04-2020).

[4] M. Expo. (2007). Sonda de neuroestimulación nervio vago, dirección: https://www.medicalexpo.es/prod/cyberonics/product-84639-544481.html (visitado 10-04-2020).

[5] D. Lulic, A. Ahmadian, A. Baaj, S. Benbadis y F. Vale, "Vagus nerve stimulation", *Neurosurgical focus*, vol. 27, E5, oct. de 2009. DOI: 10.3171/2009.6.FOCUS09126.

[6] M. Expo. (2007). Unidad de programación para la estimulación del nervio vago, dirección: https://www.medicalexpo.es/prod/cyberonics/product-84639-544485.html (visitado 10-04-2020).

[7] V. Therapy, *Physician's Manual NeuroCybernetic Prothesis Programming Wand Model 201*, English, ver. Worldwide Version, LivaNova, 2019, 22 págs.

[8] C. E. Donovan, *Out of the Black Hole: A Patient's Guide to Vagus Nerve Stimulation and Depression*. Wellness Publishers, L.L.C, 2006.

[9] R. J. Cacacho, "Diseño e implementación de un dispositivo implantable para el tratamiento de la epilepsia por medio de la estimulación del nervio vago", Universidad del Valle de Guatemala, Guatemala, Guatemala, diciembre de 2019.

[10] R. A. Girón, "Diseño e implementación de plataforma de hardware y de software que permita la comunicación remota de un neuroestimulador del nervio vago luego de su implantación", Universidad del Valle de Guatemala, Guatemala, Guatemala, diciembre de 2019.

[11] E. Foundation. (2016). Vagus Nerve Stimulation, dirección: https://epilepsychicago.org/what-is-epilepsy/treatment/vagus-nerve-stimulation/ (visitado 12-04-2020).

[12] C. Acevedo, C. Miranda, M. Campos, R. Caraballo y A. Carpio, "Informe sore la epilepsia en Latinoamérica", Organización Panamericana de la Salud, inf. téc., 2008.

[13] E. López. (2019). OPS/OMS Guatemala - Más de la mitad de personas con epilepsia no reciben ningún tipo de atención en América Latina y el Caribe, dirección: https://www.paho.org/gut/index.php?option=com_content&view=article&id=1188:mas-de-la-mitad-de-las-personas-con-epilepsia-no-reciben-ningun-tipo-de-atencion-en-america-latina-y-el-caribe&Itemid=441 (visitado 12-04-2020).

[14] Humana. (2020). HUMANA Centro de Epilepsia y Neurología Funcional, dirección: https://humanagt.org/index.html (visitado 12-04-2020).

[15] MedlinePlus. (2020). Epilepsia, dirección: https://medlineplus.gov/spanish/ency/article/000694.htm (visitado 14-04-2020).

[16] O. Devinsky, *Epilepsy: A Patient and Family Guide*. Demos Medical Publishing, 2007.

[17] H. Moawad. (2019). What is Epilepsy?, dirección: https://www.verywellhealth.com/epilepsy-overview-4155857 (visitado 14-04-2020).

[18] Vimpat. (2019). Understanding Epilepsy, dirección: https://www.vimpat.com/understanding-epilepsy (visitado 16-04-2020).

[19] HealthyChildrenOrg. (2020). Seizures and Epilepsy in Children, dirección: https://www.healthychildren.org/English/health-issues/conditions/seizures/Pages/Seizures-and-Epilepsy-in-Children.aspx (visitado 16-04-2020).

[20] CDC. (2018). Tipos de convulsiones, dirección: https://www.cdc.gov/epilepsy/spanish/basicos/convulsiones.html (visitado 16-04-2020).

[21] C. E. Stasftrom y L. Carmant, "Seizures and epilepsy: an overview for neuroscientists", *Cold Spring Harbor perspectives in medicine*, vol. 5,6, jun. de 2015. DOI: 10.1101/cshperspect.a022426.

[22] R. Ebrahimpour, K. Babakhani, S. Arani y S. Masoudnia, "Epileptic seizure detection using a neural network ensemble method and wavelet transform", *Neural Network World*, vol. 22, págs. 291-310, ene. de 2012. DOI: 10.14311/NNW.2012.22.017.

[23] J. S. Duncan, "Modern treatment strategies for patients with epilepsy: a review", *Journal of the Royal Society of Medicine*, vol. 84,3, págs. 159-162, mar. de 1991.

[24] T. Yamamoto, "Vagus Nerve Stimulation Therapy: Indications, Programming, and Outcomes", *Neurologio medico-chirurgica*, vol. 55, págs. 407-415, mayo de 2015. DOI: 10.2176/nmc.ra.2014-0405.

[25] J. C. Moog y J. W. C. Ochoa, "¿Qué es la epilepsia refractaria?", *Iatreia*, pág-163, 2003.

[26] S. Herman, "Intractable epilepsy: relapsing, remitting, or progressive?", *Epilepsy Curr*, vol. 10, n.º 6, págs. 146-148, nov. de 2010.

[27] M. Wood. (2019). New treatment options for drug-resistant epilepsy, dirección: https://www.uchicagomedicine.org/forefront/neurosciences-articles/new-treatment-options-for-people-with-drug-resistant-epilepsy#:~:text=What%5C%20do%5C%20we%5C%20do?,deep%5C%20brain%5C%20stimulation%5C%20(DBS). (visitado 20-04-2020).

[28] J. Seladi-Schulman. (2018). Vagus Nerve: Anatomy and Function, Diagram, Stimulation, Conditions, dirección: https://www.healthline.com/human-body-maps/vagus-nerve (visitado 22-04-2020).

[29] C. Pradas. (2018). Sistema nervioso simpático y parasimpático: diferencias y funciones, dirección: https://www.psicologia-online.com/sistema-nervioso-simpatico-y-parasimpatico-diferencias-y-funciones-3916.html (visitado 09-09-2020).

[30] M. Byrne. (2020). Vagus nerve, dirección: https://www.kenhub.com/en/library/anatomy/the-vagus-nerve (visitado 09-09-2020).

[31] C. Bergland. (2019). Respiratory vagus nerve stimulation (rVNS) counteracts fight-or-flight stress., dirección: https://breathwork-science.org/2019/08/29/respiratory-vagus-nerve-stimulation-rvns-counteracts-fight-or-flight-stress/ (visitado 07-04-2020).

[32] J. Seladi-Schulman. (2018). Vagus Nerve Overview, dirección: https://www.healthline.com/human-body-maps/vagus-nerve (visitado 07-04-2020).

[33] K. Gould. (2019). The Vagus Nerve: Your Body's Communication Superhighway, dirección: https://www.livescience.com/vagus-nerve.html (visitado 09-09-2020).

[34] P. Shafer y P. Dean. (2018). Vagus Nerve Stimulation (VNS), dirección: https://www.epilepsy.com/learn/treating-seizures-and-epilepsy/devices/vagus-nerve-stimulation-vns (visitado 07-04-2020).

[35] J. Robertson, C. Roux y K. Wiggins, *Vagus Nerve Stimulation*. CRC Press, 2002.

[36] R. Howland, "Vagus Nerve Stimulation", *Current Behavioral Neuroscience Reports*, vol. 1, págs. 64-73, jun. de 2014. DOI: 10.1007/s40473-014-0010-5.

[37] N. Khodaparast, S. Hays, A. Sloan y D. Hulsey. (2013). Researchers Find Early Success in New Treatment for Stroke Recovery, dirección: https://neurosciencenews.com/neurology-limb-recovery-stroke-vns-462/ (visitado 10-04-2020).

[38] AANS. (2020). Vagus Nerve Stimulation, dirección: https://www.aans.org/en/Patients/Neurosurgical-Conditions-and-Treatments/Vagus-Nerve-Stimulation (visitado 18-04-2020).

[39] V. Therapy, *Sentiva Technical Guide*, Guía técnica para el uso del Sistema de VNS, LivaNova, 2019.

[40] LivaNova. (2020). How VNS Therapy works, dirección: https://vnstherapy.com/learn-more/how-it-works (visitado 09-09-2020).

[41] ——, (2020). MRI is safe with VNS Therapy, dirección: https://vnstherapy.com/learn-more/mri-safety (visitado 09-09-2020).

[42] J. O'Reardon, P. Cristancho y A. Peshek, "Vagus Nerve Stimulation (VNS) and Treatment of Depression: To the Brainstem and Beyond", *Psychiatry (Edgmont)*, vol. 3, págs. 54-63, mayo de 2006.

[43] H. Lv, Y.-h. Zhao, D.-y. Wang y H. Chen, "Vagus Nerve Stimulation for Depression: A Systematic Review", *Front Pyschol*, vol. 10, pág. 64, ene. de 2019. DOI: 10.3389/fpsyg.2019.00064.

[44] B. Forouzan y S. Chung Fegan, *Data Communications and Networking*. Huga Media, 2007.

[45] jimblom. (2012). Serial Communication, dirección: https://learn.sparkfun.com/tutorials/serial-communication/all (visitado 02-09-2020).

[46] M. Grusin. (2013). Serial Peripheral Interface, dirección: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi (visitado 15-09-2020).

[47] D. Mattern, "Synchronous Serial Communication: A PC Link to a SPI/Microwire Device", *Circuit Cellar*, vol. 113, dic. de 1999.

[48] SFUptownMaker. (2013). I2C, dirección: https://learn.sparkfun.com/tutorials/i2c (visitado 15-09-2020).

[49] D. Tse y P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[50] Mouser. (). RF Wireless Technology, dirección: https://www.mouser.com/applications/rf-wireless-technology/ (visitado 20-08-2020).

[51] T. I. D. E. Wiki. (2011). WiFi (802.11 technology), dirección: http://wikid.io.tudelft.nl/WikID/index.php/Wi-Fi_(802.11_technology) (visitado 15-09-2020).

[52] C. González. (2020). Qué es el WiFi y cómo funciona para conectar todo a Internet, dirección: https://www.adslzone.net/reportajes/tecnologia/que-es-wifi-como-funciona/ (visitado 15-09-2020).

[53] R. Pendergast, R. Santos y S. Santos. (2019). RF 433MHz Transmitter/Receiver Module With Arduino, dirección: https://randomnerdtutorials.com/rf-433mhz-transmitter-receiver-module-with-arduino/ (visitado 03-08-2020).

[54] ST, *Sub-1 GHz (433, 868 or 915 MHz) programmable transceiver module*, English, ver. Version 3, ST Elecronics, 2017, 42 págs.

[55] TMRh20. (2020). Optimized high speed nRF24L01+ driver class documentation, dirección: tmrh20.github.io/RF24/index.html (visitado 02-01-2021).

CAPÍTULO 15

Anexos

15.1. Código

Para acceder al código que implementa el módulo Programador del proyecto Estimulador del Nervio Vago de la UVG, acceder al repositorio de GitHub en el siguiente link: `https://github.com/larivera-UVG/Estimulador-Nervio-Vago/tree/master/Programador`.

15.2. Video demostrativo

Para observar un video en el que se muestra el funcionamiento del sistema completo, consultar el siguiente link: `https://youtu.be/A6lljqlvt8c`.

CAPÍTULO 16

Glosario

Arduino: Plataforma electrónica open-soruce basada en hardware y software fácil de usar. 32, 36, 37, 45, 46, 48, 54, 55, 57–60

Arduino Mega: La placa de Arduino de 8 bits con 54 pines digitales, 16 entradas analógicas y 4 puertos seriales. Basada en el microcontrolador ATmega2560. 31, 35–38, 46

Arduino Uno: La placa de Arduino más fácil de utilizar para principiantes. Basada en el microcontrolador ATmega328p. 31–34, 46, 48, 50, 53, 54, 60

ESP8266: Chip de bajo costo Wi-Fi con un stack TCP/IP completo y un microcontrolador fabricado por Espressif. 53, 56–60

fibras aferentes: Transforman la información que recogen los sentidos y las transforman en impulsos nerviosos. 17

fibras eferentes: Encargadas de propagar los impulsos eléctricos que están destinados a activar o desactivar ciertas glándulas y grupos musculares. 17

GUI: *Graphical User Interface*, Interfaz Gráfica de Usuario. Permite al usuario interactuar con dispositivos electrónicos por medio de iconos gráficos. 42–44, 46, 50, 51

punto de acceso: Dispositivo o un área con conectividad inalámbrica a internet a través de WiFi. 29

Putty: Es un cliente SSH y Telnet que permite conectarse a servidores remotos por medio de un inicio de sesión en ellos, permitiendo ejecutar comandos. Tiene soporte para conexiones de puerto serie local. 31, 37–39

Python: Lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código, lenguaje de programación multiparadigma, que soporta orientación a objetos, programación imperativa y programación funcional. 33, 38, 39, 42, 46, 51, 55

remisión: Atenuación o desaparición completa en el paciente de los signos y síntomas de su enfermedad, ya sea como consecuencia del tratamiento o de forma espontánea. 16

Sistema Nervioso Central: Conformado por el cerebro y la médula espinal, se desempeña como el centro de procesamiento principal para todo el sistema nervioso y controlan todas las funciones del cuerpo. 17

sistema nervioso parasimpático: Encargado de regresar a los órganos al estado natural luego de haber estado en estado de alerta. Reduce el volumen de los pulmones, disminuye la frecuencia cardiaca, relaja las músculos y estimula el proceso digestivo. 18

sistema nervioso simpático: Encargado de poner al cuerpo en un estado de alerta fisiológica. Dilata las pupilas, acelera la frecuencia cardiaca, mantiene el tono muscular, inhibe el sistema digestivo para concentrar esfuerzos en tareas de ataque y huida y encarga a la glándula suprarrenal liberar adrenalina para todo el torrente sanguíneo. 18

Spyder: Entorno científico escrito en Python, para Python y diseñado por y para científicos, ingenieros y analistas de datos. 31, 33, 38–40, 51

Tkinter: Es el paquete estándar para interfaz gráfica de usuario de Python. 33