Universidad Central de Venezuela
Facultad de Ciencias

Escuela de Física

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE FÍSICA

DESARROLLO DE UN SISTEMA PROTOTIPO DE ESTIMULACIÓN Y ADQUISICIÓN DE DATOS PARA PRÁCTICAS ELECTROFISIOLÓGICAS UTILIZANDO HARDWARE Y SOFTWARE LIBRE

Trabajo Especial de Grado presentado por
Gabriel Barreto
ante la Facultad de Ciencias de la
Ilustre Universidad Central de Venezuela
como requisito parcial para optar al título
de: **Licenciado en Física**
Con la tutoría de: Dr. Esteban Álvarez
M.Sc. Christian Calderón

Octubre-2015

Caracas-Venezuela

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE FÍSICA

DESARROLLO DE UN SISTEMA PROTOTIPO DE ESTIMULACIÓN Y ADQUISICIÓN DE DATOS PARA PRÁCTICAS ELECTROFISIOLÓGICAS UTILIZANDO HARDWARE Y SOFTWARE LIBRE

Trabajo Especial de Grado presentado por
Gabriel Barreto
ante la Facultad de Ciencias de la
Ilustre Universidad Central de Venezuela
como requisito parcial para optar al título
de: **Licenciado en Física**
Con la tutoría de: Dr. Esteban Álvarez
M.Sc. Christian Calderón

Octubre-2015

Caracas-Venezuela

27 de octubre de 2015

Consejo de la Escuela de Física
Facultad de Ciencias
Universidad Central de Venezuela

Estimados miembros del Consejo de Escuela

Reciban un cordial saludo. Conforme a lo establecido en el artículo 13 de la "Normativa de Trabajo Especial de Grado de la Licenciatura en Física de la Facultad de Ciencias de la UCV" les remitimos tres ejemplares de la monografía de TEG del estudiante **Gabriel Barreto**, CI: **16.714.502**, titulado **DESARROLLO DE UN SISTEMA PROTOTIPO DE ESTIMULACIÓN Y ADQUISICIÓN DE DATOS PARA PRÁCTICAS ELECTROFISIOLÓGICAS UTILIZANDO HARDWARE Y SOFTWARE LIBRE**, los cuales hemos revisado y consideramos listos para la evaluación por parte de un jurado.

Agradeciendo la consideración que sirvan prestar a la presente, nos despedimos atentamente,

Dr. Esteban Álvarez
9.525.744
Tutor
UCV

M.Sc. Christian Calderón
13.694.654
Tutor
UCV

"At some point, everything's gonna go south on you and you're going to say, this is it. This is how I end. Now you can either accept that, or you can get to work. That's all it is. You just begin. You do the math. You solve one problem and you solve the next one, and then the next. And If you solve enough problems, you get to come home."

- Mark, Watney [1]

"En algún punto, todo te va a salir mal y vas a decir, aquí es. Aquí es donde yo muero. Ahora, puedes aceptar eso o puedes ponerte a trabajar. Eso es todo. Sólo empieza. Haz los cálculos. Resuelve un problema y luego resuelves el siguiente, y el siguiente. Y si resuelves suficientes problemas, entonces puedes ir a casa."

- Mark, Watney [1]

Agradecimientos

Quisiera agradecer especialmente a mis padres por toda la paciencia y apoyo emocional y económico que me han regalado, aguantando todos los cambios de rumbo que he realizado en mi vida, en búsqueda de una vocación verdadera.

A mis hermanos Gilbert, Guilmer y Gabriela, lo que más le agradezco es no graduarse antes que yo. Gracias por ser mis amigos, mis confidentes y mis compañeros. Ahora si se pueden graduar. Igualmente a mi familia cercana: Margarita, Carlucho, Ingrid, Geri, Gerardo, Majo, Blas (casi), Maria Angélica, Pedro (padre e hijo) y Olga. Su cariño y apoyo incondicional no pasa desapercibido.

A *Los Líderes*, el grupo de amigos con los que celebro cada victoria, cada derrota, cada sinsabor de la vida y cada evento importante. Angel, Rainer, Joile , Gabo U, Maria Gabriela, Luis Miguel, Omar, Eddy Dize, Daniel, Ropsy, Giovanny, Theo, Mixo, Mery, Yana, Gaba Tu, Gaby Ro, Lorena, Jherma, Maria T, Ricardo, Roberto y José. Ustedes generan suficiente felicidad para aguantar la realidad que vivimos y tener una sonrisa siempre para la vida y para los demás.

A María Soledad Cobos Rubio, que me regaló todo el apoyo, las herramientas y la libertad para determinar la carrera que quería ejercer en mi vida. Gracias por todo el apoyo incondicional y el cariño. Gracias a Eric Vandernoot y su charla de Astrofísica que cambiaría mi vida.

Gracias a mis compañeros de carrera, Fátima, Dinibel, Enrique y Claudia, que siempre me dieron energía cuando a mi se me acababa la pila. Gracias por todo el apoyo y compañerismo.

Gracias a Ariana Ascanio, que en la fase final de este proyecto me ha brindado su apoyo y paciencia. Gracias por creer en mí y hacerme ver cuánto valía mi trabajo, cuando mi ánimo desaparecía. Gracias por levantarme en los momentos en que la autocrítica o la duda me paralizaron.

Gracias al profesor José Antonio López (JAL) por el apoyo desde que solicité que se abriera la mención Biofísica en la escuela. Los contactos para el tema de tesis actual se deben a él. Gracias también por el apoyo en la parte final de este proyecto, su presión y oportunidades permitieron cerrar este documento y esta etapa.

Gracias a mi tutor por parte de física el Dr. Esteban Álvarez y toda la paciencia y apoyo que tuvo con este proyecto. Gracias por la libertad que me ofreció para trabajar de manera independiente y con las herramientas que quisiera. Gracias también por la comunicación sincera y abierta, que permitió en todo momento mantener un trabajo amistoso.

Gracias a mi tutor por parte de biología el M.Sc. Christian Calderón y toda su disposición a hacerse cargo de este proyecto, en el cual usted tuvo que caer como paracaidista. Su perfeccionismo y críticas constructivas son la base que permite crear mejores profesionales científicos.

Gracias a ti, que has leído hasta este punto buscando tu nombre, pero no lo encontraste.

Gracias por estar pendiente de mí y mi tesis y por eso esperas tu debido reconocimiento en estas líneas. Tienes el permiso de reclamar tu puesto y en persona tendremos que ver cómo lo arreglamos.

RESUMEN

En el presente trabajo se propone el desarrollo de un Prototipo de Estimulador Eléctrico para experimentación fisiológica, utilizando hardware y software libre, con el propósito de atender y dar solución al problema de obsolescencia de los equipos de uso frecuente en las prácticas del Laboratorio de Docencia en el área de fisiología de la Escuela de Biología – UCV. Por esta razón se decidió diseñar un prototipo de bajo costo y con elementos que se encuentran en el mercado nacional que pudiese reemplazar los equipos dañados u obsoletos que son necesarios en el laboratorio. Para alcanzar ese objetivo se utilizaron herramientas Open Source (software y hardware libre). El instrumento a reemplazar en el Laboratorio es el Estimulador Eléctrico que es capaz de generar pulsos y trenes de pulsos de frecuencia y amplitud variable, específicamente el equipo "Ortec 4710 Dual Channel Stimulator". Por sus extensivos años de uso, falta de diagramas electrónicos y manual de usuario, se decidió crear un nuevo equipo que emule al instrumento mencionado. El prototipo se realizó utilizando el microcontrolador Arduino UNO R3, controlado por una interfaz gráfica (GUI) realizada con el programa multiplataforma Qt en su versión 5.5, también se realizó el circuito para adecuar la señal tanto de estimulación del músculo gastrocnemio en sapos *bufo marinus* como de recepción de la señal repuesta del mismo. Se aprovechó la interfaz gráfica para guardar los datos en los registros de la computadora y graficar en tiempo real la respuesta fisiológica muscular del sapo mencionado. Se obtuvo un equipo capaz de generar pulsos de una duración mínima de 100us y máxima de 11s, capaz de suministrar una corriente máxima de 84 mA, y una potencia máxima de 290 mW, con 4 escalas para modificar la amplitud del pulso (10V, 1V, 100mV y 10 mV) con un nivel de ruido de 6mV y una tasa de muestreo de 5814 muestras por segundo. La Interfaz se puede utilizar tanto en Windows como en Linux.

Índice general

Índice de figuras

Índice de cuadros

Introducción

La Bioelectricidad ha sido motivo de fascinación para el ser humano desde su descubrimiento a finales del siglo XVIII por Luigi Galvani. Desde entonces el estudio de los nervios, la respuesta muscular y el funcionamiento del cerebro ha sido de particular interés en diversas ramas del conocimiento científico. Para el estudio fisiológico del funcionamiento de los músculos y los nervios motores, los científicos han experimentado aplicando estímulos eléctricos de origen no biológico a los nervios para poder controlar y medir los efectos que estos causan. Sin embargo, la aplicación de estimuladores eléctricos en los nervios no se limita al campo de la investigación científica, actualmente en el campo de la medicina, los estimuladores eléctricos se utilizan en pacientes en el área operatoria para poder localizar nervios específicos y poder suministrar de manera local anestesia a zonas extensas del cuerpo sin necesidad de dormir al paciente. En nuestro trabajo se desarrolló un estimulador eléctrico, diseñando e implementando una Interfaz Gráfica de Usuario (GUI por sus siglas en inglés Graphic User Interface) y un microcontrolador, basándonos en modelos anteriores de estimuladores comerciales existentes (Ortec 7410 Dual Channel Stimmulator), pensando en su aplicación inmediata en el área científica.

Para la elaboración del prototipo se tuvo en cuenta que el material a utilizar en lo posible debería soportarse en hardware y software libre con el propósito que en un futuro el proyecto pueda ser ampliado y multiplicado sin necesidad de pagar regalías a terceros que acarreen mayores gastos. Por esta razón se utilizó el programa Qt v.5.5 para el desarrollo del GUI y el microcontrolador Atmel 328p integrado en la placa Arduino UNO R3 para desarrollar la parte de hardware. Se hizo énfasis en adquirir suministros y componentes electrónicos presentes en el mercado local, de forma tal de lograr la reposición y puesta a punto del dispositivo a diseñar.

Este proyecto está motivado por las necesidades en las prácticas de electro-fisiología de la Escuela de Biología de la UCV, donde la escasez y costo de equipos estimuladores eléctricos limita el número de estudiantes que pueden simultáneamente realizar las prácticas que requieren este instrumento. El proyecto pretende servir como una colaboración interescuela para solventar los problemas propios de nuestra alma mater.

El siguiente documento tendrá 5 capítulos, en el primer capítulo se iniciará proporcionando información sobre los antecedentes del problema, donde se expondrá a cabalidad el problema y se definirán los objetivos que persigue este trabajo. Seguidamente tendremos una descripción de los aspectos teóricos más relevantes necesarios para entender el trabajo

en el capítulo 2. En el capítulo 3 describiremos la metodología empleada para abordar el problema y poder diseñar y construir el instrumento necesario, para luego en el capítulo 4 exponer los resultados obtenidos en cada paso de la construcción del instrumento. Finalmente tendremos las conclusiones y consideraciones finales en el capítulo 5.

Capítulo 1

Planteamiento del Problema, Objetivos, Alcance y Justificación

El presente capítulo empezará abordando el problema a tratar, se definieron los objetivos del trabajo realizado además del alcance y la justificación del mismo.

1.1. Planteamiento del Problema

La Escuela de Biología de la UCV desde hace unos años ha empezado a recurrir a otras escuelas dentro de la misma universidad, con el fin de solicitar ayuda para resolver los problemas y carencias que posee actualmente. En ese sentido, la Escuela de Física ha respondido promoviendo la cooperación inter-escuela. Entre la ayuda que hace falta, la Escuela de Biología informó de un déficit importante de algunos equipos electrónicos utilizado en los Laboratorios de Docencia (LDB - Laboratorio de Docencia de Biología UCV) y decidió plantear a la Escuela de Física la posibilidad de su construcción. Un equipo relevante empleado en el LDB y que presenta problemas, es el del Estimulador Eléctrico "Ortec 4710 Dual Channel Stimulator", el cual es utilizado en las prácticas fisiológicas donde intervienen nervios o músculos. En un inicio, el LDB contaba con un número suficiente de Estimuladores Eléctricos, pero estos se han dañado con el uso a través de los años. La escuela de Biología ha hecho otros intentos de solucionar este problema contratando técnicos que intentaron poner en funcionamiento los equipos dañados, pero sin éxito. Al ser un equipo antiguo, los diagramas eléctricos y características del equipo no se encuentran en el laboratorio y tampoco se ha encontrado en la base de datos que proporciona Internet, lo cual ha contribuido a no disponer de la información necesaria para arreglar los equipos que se han deteriorado.

Actualmente existen en funcionamiento unicamente 2 Estimuladores Eléctricos y semestralmente el LDB reciben aproximadamente a 48 alumnos que necesitan hacer uso de este instrumento. Cada semestre se abren cerca de 6 secciones, donde cada sección tiene aproximadamente 8 alumnos. Las prácticas se organizan en 2 secciones por día. El

escaso número de Estimuladores Eléctricos obliga a los alumnos a trabajar en grupos y de igual manera es necesario tomar turnos para realizar las medidas. Esto repercute en el tiempo total en el que se realizan las prácticas y también compromete el uso del material biológico, ya que éste, al ser un animal al que se le practica una disección, perece a las pocas horas.

El Laboratorio de Instrumentación Científica de la Escuela de Física propuso el proyecto que se desarrolló, donde cada computadora que tenga el LDB se puede convertir en un instrumento capaz de estimular, con las mismas características que el antiguo estimulador, e incluso se puede utilizar todos los recursos del computador para tener también un instrumento que permita recoger y visualizar datos.

1.2. Objetivos

1.2.1. Objetivo General

Diseño y construcción de un dispositivo prototipo estimulador de tejido neuro-muscular y registro de la señal respuesta, controlado por una PC haciendo uso de software y hardware libre, para dotar los Laboratorios Docentes de Fisiología de la Escuela de Biología de la Facultad de Ciencias de la UCV.(LDB)

1.2.2. Objetivos Específicos

- Hacer una revisión bibliográfica de los aspectos básicos relacionados con el tejido neuromuscular.

- Determinar los valores adecuados de voltaje y corriente de las señales biológicas involucradas en los experimentos del LDB.

- Diseñar un generador de pulsos (ondas cuadradas), a partir de la selección de componentes electrónicos presentes en el mercado local y accesibles según el presupuesto del LDB, que permita el diseño de un instrumento con características similares al estimulador Ortec 4710 Dual Channel Stimulator

- Implementar un sistema de adquisición de datos a partir de la selección de componentes electrónicos presentes en el mercado local y accesibles según el presupuesto del LDB, para visualizar y guardar la respuesta ante el estímulo al tejido neuromuscular.

- Implementar una interfaz gráfica que permita el control y adquisición de los datos para procesamiento y análisis de la respuesta neuromuscular.

1.3. Alcance

El presente proyecto tiene el potencial de llegar a ser un producto comercial. La idea es que se pueda reproducir a un bajo costo y con componentes presentes en el mercado local; para incrementar las potencialidades de los computadores presentes en el laboratorio, pasando a ser instrumentos de control y medida. El trabajo actual, se desarrolló hasta su etapa de prototipo, donde se presenta una GUI con la cual se puede controlar los tiempos de estimulación y además guardar los datos. De igual, manera se presentó un circuito para acondicionar la señal de estimulación y también de la recepción de la respuesta mecánica del músculo estimulado.

1.4. Justificación

La creciente inflación, el escaso acceso a divisas, y el limitado presupuesto universitario otorgado por el estado a las universidades públicas, son factores que afectan directamente la capacidad de las universidades para reponer los equipos dañados u obsoletos. Es por esto que es necesario realizar un proyecto, aprovechando los recursos que se poseen y utilizando las ventajas del Software y Hardware Libre, para poder suplir las necesidades presentes en nuestra alma mater. Específicamente, en nuestro proyecto nos enfocamos en la necesidad inmediata del desarrollo de un prototipo de Estimulador Eléctrico para el LDB.

Capítulo 2

Marco Teórico

En este capítulo se exponen los conceptos teóricos que se utilizarán posteriormente para el desarrollo del instrumento, para ello empezaremos estudiando el objeto al cual se le aplicarán los estímulos eléctricos, el tejido neuro-muscular, y finalmente hablaremos del hardware y sofware

2.1. Tejido neuromuscular: De la motoneurona hasta el músculo

Los seres vivos poseen diferentes mecanismos para desplazarse en su entorno, desde los más pequeños seres unicelulares hasta los organismos multicelulares complejos. En los organismos pluricelulares complejos el mecanismo evolutivo que han desarrollado estos seres es la creación de un tejido especializado para contraerse denominado músculo. Los músculos se reparten en el organismo y se pueden diferenciar de acuerdo a su función y su morfología. De acuerdo a su función se puede clasificar como:[2]

- el músculo cardíaco, especializado para las funciones del corazón, bombeando sangre al sistema circulatorio de manera involuntaria y rítmica.
- el músculo liso, encontrado generalmente en órganos internos, vasos sanguíneos, y aparato reproductor y excretor. También de contracción involuntaria.
- finalmente el músculo esquelético, de contracción voluntaria, el cual tiene como principales funciones el desarrollo de tensión y el acortamiento y es el responsable del movimiento del esqueleto, el globo ocular y el sistema bucal.

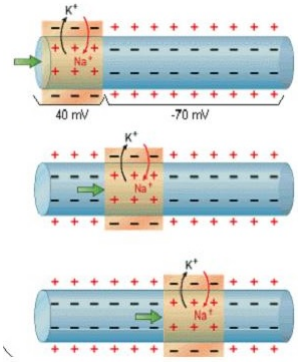
El presente trabajo se centrará sobre el músculo esquelético, ya que éste es el objetivo a estimular. Una de las características del músculo esquelético es que su contracción es controlada y esto viene dado por el sistema nervioso. El sistema nervioso es una red de tejido cuya unidad básica es la *neurona* y es el encargado de recoger y procesar información, para generar respuestas coordinadas y poder controlar comportamientos complejos

que puedan efectuar una adecuada, oportuna y eficaz interacción con el medio ambiente. Las neuronas transmiten información utilizando una combinación de señales eléctricas y químicas. Las propiedades de las señales eléctricas permiten a las neuronas transportar información rápidamente y con exactitud para coordinar las acciones de muchas partes del cuerpo de un animal.

2.1.1. La motoneurona

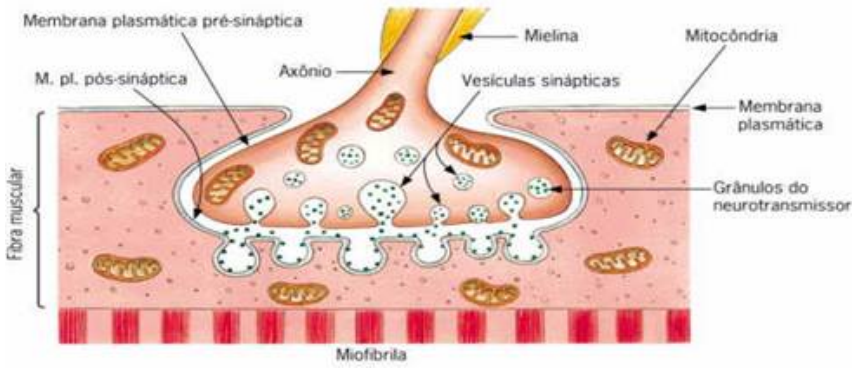
Las neuronas han desarrollado propiedades especializadas que les permiten recibir información, procesarla y transmitirla a otras células. Aunque las neuronas varían ampliamente en su forma y tamaño, cada neurona tiene típicamente un *soma*, o cuerpo celular, que es el responsable del mantenimiento metabólico de la célula y del que emanan diversas finas prolongaciones. Hay dos tipos de prolongaciones principales: *dendritas* y *axones*. Las *dendritas* se extienden desde el cuerpo celular y sirven de superficie receptora para conducir señales de otras neuronas hacie el cuerpo celular. Los *axones* son sistemas especializados que conducen señales lejos del cuerpo celular. Las *motoneuronas* son neuronas que estan especializadas para estimular el sistema motor del individuo, generalmente poseen dendritas ramificadas y axones muy largos que se extienden desde el soma hasta el músculo. Los haces de axones, que recorren el cuerpo de los animales a través de los tejidos, son lo que se denominan nervios. Las motoneuronas reciben a través de las dentritas señales de otras neuronas que son sumadas y ésto determinará si la neurona iniciará su propia respuesta. La respuesta de una motoneurona es una señal eléctrica que se propagará por el axón hasta llegar al músculo. De manera natural, la concentración desigual de iones entre el ambiente intra-celular y extra-celular establece una diferencia de carga y un potencial denominado *potencial de membrana*. El movimiento de estos iones entre los ambientes intra y extra celular produce una modificación del potencial de membrana que es conocido como una despolarización de la membrana. Una despolarización local estimula a que las zonas contiguas también se despolaricen y así sucesivamente. La célula luego tiene mecanismos para devolver los iones a su lugar original, intra o extra membrana, y así volver a restablecer el potencial de membrana existente antes de la despolarización. La propagación por el axón de esta despolarización es lo que denominamos *impulso nervioso*.

Entre el nervio y el músculo no hay ninguna unión física, por lo tanto es necesario utilizar un mensajero químico que transmita la información desde el nervio, hasta el músculo. La sustancias químicas que llevan información fuera de las células nerviosas son llamadas neurotransmisores y para transmitir la información desde el nervio hasta el músculo se utiliza el neurotransmisor *acetilcolina* (Ach). El extremo del axón que se comunica con el músculo tiene forma de bulbo y es llamado *botón sináptico*. En este botón, se producen vesículas llenas de acetilcolina que cuando reciben el impulso nervioso liberan su contenido al espacio entre el nervio y el músculo que es llamado *espacio sináptico*. El músculo posee receptores de Ach y cuando este neurotransmisor se une con su receptor producen la liberación de iones que estimulan la contracción del músculo.



Tomado de [3]

Figura 2.1: Impulso nervioso: Despolarización de la membrana por movimiento de iones y luego su propagación

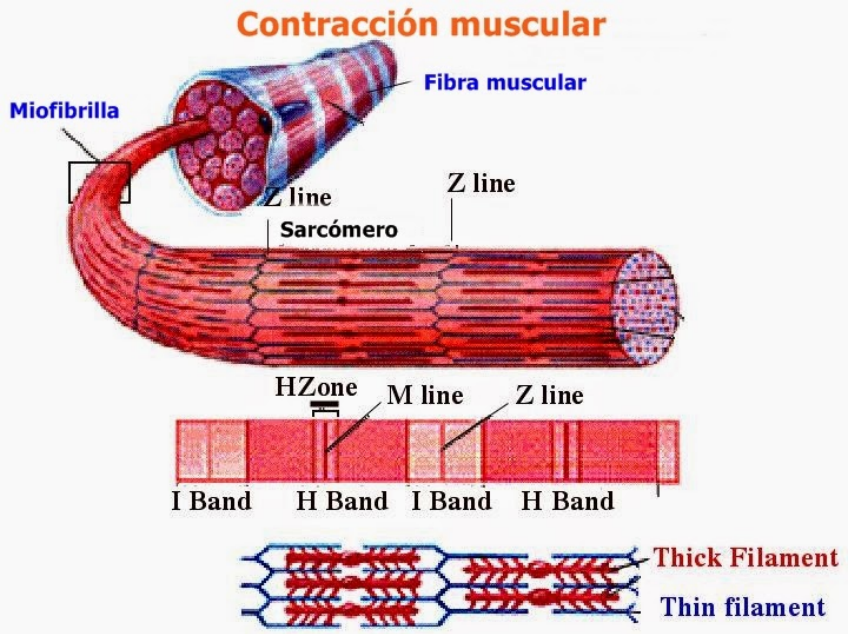


Tomado de [4]

Figura 2.2: Placa motora: unión de la motoneurona con el músculo

2.1.2. El músculo esquelético

El músculo esquelético puede mover partes de un ser humano o de u animal porque cada extremo del músculo está unido a un hueso o a alguna otra estructura, y cuando el músculo se acorta, cambia la relación espacial física entre los puntos de anclaje. Normalmente, los músculos se hallan anclados en cada extremo por una resistente banda de tejido conectivo llamada tendón. Los músculos están compuestos por unas células largas cilíndricas y multinucleadas llamadas *fibras musculares* que estan dispuestas en paralelo, una al lado de la otra. Cada fibra muscular está constituida a su vez, por numerosas subunidades paralelas denominadas *miofibrillas*, las cuales consisten en una serie de unidades repetidas logitudinalmente llamadas *sarcómeros*[5]. El sarcómero es la unidad funcional del músculo esquelético. En la Figura 2.3 podemos apreciar gráficamente lo descrito en palabras

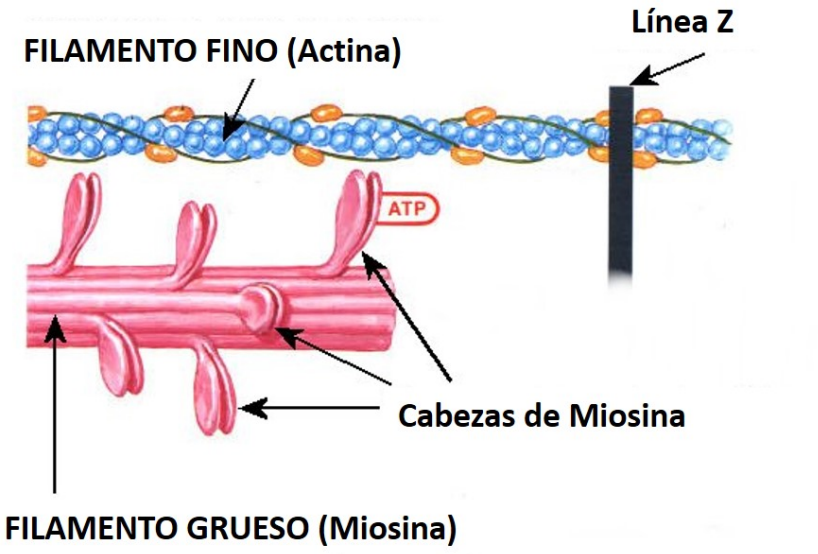


Tomado de [6]

Figura 2.3: Composición del músculo esquelético, desde la fibra muscular hasta el sarcómero

Cada sarcómero esta limitado a cada extremo por la *Línea Z*. Extendiéndose en ambas direcciones desde la línea Z, hay numerosos *filamentos delgados* que están constituidos principalmente por una proteína denominada *actina*. Estos filamentos delgados se intercalan con filamentos gruesos constituidos por la proteína *miosina*. La contracción muscular ocurre por el deslizamiento de los filamentos finos sobre los filamentos gruesos en todos los sarcómeros de la miofibrilla. Los filamentos gruesos halan a los filamentos finos hacia el centro del sarcómero y como los filamentos finos están unidos a la Línea Z, los sarcómeros se acortan. La interacción entre los filamentos finos y los filamentos grueso se le denomina *puente cruzado*. Estos son los encargados de producir el movimiento de deslizamiento entre las fibras.

La señal que proviene de la neurona se transmite al músculo en forma del neurotransmisor acetilcolina. Cuando la acetilcolina se une a su receptor en el músculo, promueve que dentro de éste se liberen iones de Calcio. El Calcio estimula la unión entre miosina y actina. Una vez que se unen la miosina y la actina ocurren cambios conformacionales en el puente cruzado (unión de actina y miosina) que producen un movimiento de la actina (filamento fino) sobre la miosina (filamento grueso). Esto se traduce en un acortamiento del sarcómero, que se traduce en un acortamiento de la fibra muscular. A nivel macroscópico para poder medir el acortamiento podemos utilizar 2 variables, la distancia y la tensión. Cuando el músculo se contrae, la distancia entre sus extremos disminuye y puede ser medida, y también el músculo produce una fuerza que puede ser medida mediante un transductor de tensión.

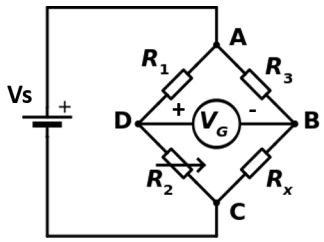


Tomado de [7]

Figura 2.4: Filamentos finos y filamentos gruesos

2.2. Transductor de tensión

El transductor de tensión también conocido como *celda de carga*, es un instrumento capaz de convertir una fuerza en una señal eléctrica. El dispositivo se basa en usar galgas extensiométricas como resistencias en un puente de Wheatstone. Las galgas extensiométricas son materiales piezorresistivos, es decir son materiales que cambian su resistencia eléctrica cuando se somete a esfuerzos o deformaciones. El puente de Wheatstone es una configuración de 4 resistencias formando un circuito cerrado. Esta configuración se utiliza para conocer indirectamente el valor desconocido de alguna de las resistencias que componen el puente. En la figura 2.5 se puede observar la disposición del puente donde R1, R2 y R3 son conocidas y Rx desconocida.



Tomado de [8]

Figura 2.5: Puente de Wheatstone

En este caso podemos calcular Rx

$$Rx = \frac{R2}{R1} * R3 \tag{2.1}$$

$$Vg = (\frac{R2}{R2 + R1} - \frac{Rx}{Rx + R1})Vs \tag{2.2}$$

Es importante mencionar que la relación entre Vg y Rx no es lineal como se puede apreciar en la ecuación 2.2. Por lo tanto es necesario realizar una curva de calibración para asociar una fuerza determinada a un valor de voltaje.

2.3. Open Source: Hardware y Software libre

Por razones económicas el proyecto siempre se pensó en realizarlo utilizando Software y Hardware que fueran Open Source, conocido en el idioma español como Software y Hardware Libre.

2.3.1. Open Source

El Open Source es el término utilizado para hardware o software donde el usuario puede acceder a los planos o al código fuente de los equipos de manera abierta y gratis, de manera que el usuario puede usar, copiar y redistribuir el código fuente sin necesidad de pagar regalías o licencias. Como el proyecto esta pensado para ser utilizado, desarrollado y multiplicado para el área académica de una universidad pública con presupuesto limitado, usar tecnología de código abierto, permite disminuir costos y tener más libertad para utilizar y mejorar el producto para ajustarse a las especificaciones requeridas por los usuarios. Para el proyecto se eligió el Programa Qt para desarrollar la interfaz gráfica y Arduino como placa para utilizar el microcontrolador Atmel 328p.

2.3.2. Qt

Qt es una infraestructura digital o Framework multi-plataforma para el desarrollo de Interfaces Gráficas de Usuario (GUI, por sus siglas en inglés, Graphic User Interface). Esta desarrollado como sofware libre y de código abierto a través de Qt Project, un proyecto donde participa la comunidad, desarrolladores de Nokia, Digia y otras empresas. La libertad de poder crear programas multiplataforma fue uno de los primeros atractivos del programa, ya que permite desarrollar programas para Windows o para Linux, mantiendo la apariencia del sistema operativo nativo, sin necesidad de cambiar el código fuente. Además la gran comunidad de usuarios de Qt es importante, ya que la abundancia de información y tutoriales disponibles para aprender Qt es gigante de manera que el desarrollo puede realizarse de manera más rápida que con otros programas.[9] [10]

2.3.3. Arduino

Arduino es una compañía que produce software y hardware open source, diseña y produce kits para desarrollar artículos digitales y objetos interactivos que pueden sensar y controlar el mundo físico a través de un microcontrolador.

los kits Arduino esta basado en placas que poseen microcontroladores Atmel. Los sistemas Arduino vienen con una serie de pines de salida y entrada que pueden ser conectadas con varias tablas de expansión conocidas como "shields" y con otros circuitos. Las tablas Arduino utilizan comunicación serial, incluyendo puertos USB en algunos modelos, para descargar los programas desde un computador. Los programas para controlar el microcontrolador se realizan en un Ambiente de Desarrollo Integrado (IDE por sus siglas en ingles Integrated Development Enviroment) producido por la misma compañía, donde se puede desarrollar en los lenguajes C, C++ y Java.

Tomado de [11]

Figura 2.6: Arduino UNO

El Arduino UNO R3 posee las características del microcoontrolador ATMEGA 328p:

- Voltaje de Operación: 5V
- Una memoria Flash de 32 Kbytes para guardar el programa o código
- 32 pines de los cuales 14 son pines de entrada y de salida donde 6 de ellos son capaces de dar una señal PWM (pulsos de ancho modulado por sus siglas en inglés) y 6 pines son entradas analógicas.
- Reloj de 16 MHz
- 2 contadores internos de 8 bits y uno de 16 bits con pre-escalador propio

- Conversor A/D de 10 bits
- El voltaje aceptado para las entradas analógicas es de 0 a 5V
- 20mA máximos de corriente DC entregada por los puertos de salida

El arduino posee una serie de funciones para comunicarse y manipular los puertos. Por ejemplo, para escribir en los puertos se utiliza la función *digitalWrite(pin,status)* donde se puede manipular los puertos sin necesidad de usar el registro de los diferentes puertos (PORTA, PORTB, PORTC, etc). Para realizar la conversión A/D se utiliza la función *analogRead(pin)*. La frecuencia de la conversión A/D viene dada por la siguiente fórmula

$$\text{Frecuencia conversor A/D} = \text{Frecuencia Reloj} \times \frac{1}{\text{Preescalador}} \times \frac{1}{\text{Ciclos de reloj por conversión}} \tag{2.3}$$

Donde el preescalador puede tener ciertos valores determinados (128, 64, 32, 16, 8, 4, 2, 1) y los ciclos necesarios para la conversión se pueden encontrar en el datasheet [12]

2.4. Ortec 4710 Dual Channel Stimmulator

El equipo Ortec 4710 Dual Channel Stimmulator es un equipo de estimulación por medio de pulsos cuadrados donde el usuario tiene una amplia variedad de parámetros para poder controlar la frecuencia de los pulsos y su amplitud. Se dedicará una sección completa a explicar el funcionamiento de dicho equipo, ya que sus características y funciones se buscarán emular en la interfaz gráfica.

Figura 2.7: Equipo Ortec

El Ortec 4710 puede generar pulsos con un período y un ancho determinado y también puede generar trenes de pulso, para ello el Ortec dispone de un switch para hacer cambio entre las funciones. En la figura 2.8 se puede observar dicho switch. Para generar el pulso o los trenes de pulso el usuario debe configurar varios parámetros que son el Período Maestro, Delay, la Duración del Pulso, Duración del Tren, Período del Tren y la Amplitud del

Pulso. Todos los parámetros son controlados por medio de 2 perillas dispuestas verticalmente, donde la perilla inferior es un factor de multiplicación para los valores dispuestos en la perilla superior. En la figura 2.9 se puede observar las perillas mencionadas. Todos los parámetros están en unidades de milisegundos, excepto la Amplitud del Pulso, que sus perillas tienen unidades de Voltio.

Figura 2.8: Switch para cambiar entre el modo de pulso simple o tren de pulsos

Figura 2.9: Perillas para ajustar parámetros del Ortec 4710

El **Período Maestro** se refiere al tiempo en que el usuario desea que la estimulación se repita. Al transcurrir el tiempo del período maestro los otros parámetros como Delay, Duración del Pulso, Duración del Tren y Período del Tren se repiten con los mismos valores. El Ortec tiene en uno de sus puertos de salida, una señal de sincronización que consiste en un pequeño pulso sincronizado con el Período Maestro. El usuario puede definir un retraso entre la señal de sincronización del Período Maestro y el inicio del pulso o de los trenes de pulso utilizando el parámetro **Delay**. Si el valor de Delay es cero (0) milisegundos, entonces el pulso o el tren de pulso comenzará sincronizadamente el Período Maestro. La **Duración del Pulso** es el tiempo en el cual se requiere que el pulso permanezca en el valor DC seleccionado. En el modo de pulso simple, este tiempo se refiere a la duración del único pulso, y en el modo de tren de pulsos, será la duración de cada uno de los pequeños pulsos del tren. La **Duración del Tren** se refiere al tiempo durante

el cual se desea que se repita el tren de pulsos. Una vez transcurrido este tiempo no se producirán más pulsos hasta el próximo Período Maestro. En el modo de pulso simple este parámetro no altera en ningún aspecto la estimulación. Finalmente el **Período del Tren** se refiere al tiempo en que se desea que se repita el pulso durante la Duración del Tren. En las Figuras 2.10 y 2.11 se pueden observar ejemplos típicos de estimulación de pulso simple y estimulación de tren de pulsos señalando los parámetros mencionados. La **Amplitud del Pulso** se refiere al valor de voltaje DC que se desea que tenga el pulso.

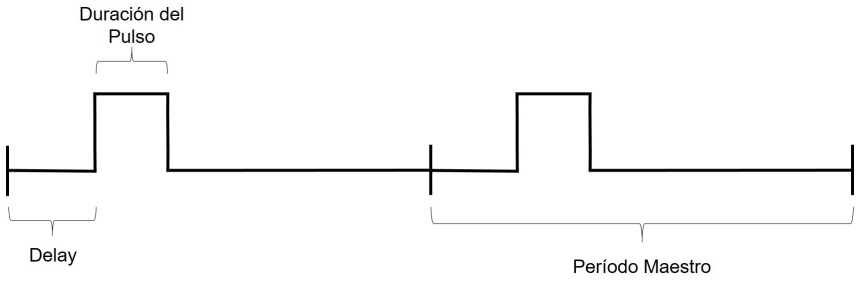


Figura 2.10: Ejemplo de una estimulación típica en modo de pulsos simples

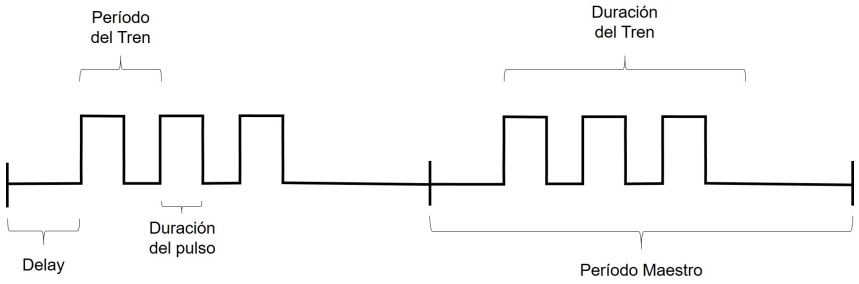


Figura 2.11: Ejemplo de una estimulación típica en modo de trenes de pulso

Figura 2.12: Switch 3 estados para selección del modo de estimulación

Finalmente después de enceder el equipo, los usuarios tienen un switch de 3 posiciones para seleccionar el modo de realizar la estimulación. el detalle del switch se puede ver en la figura 2.12. En la posición central el equipo esta en *Standby*, es decir el equipo está encendido pero no se realiza ningún pulso. En la posición superior el equipo pasa a estar en *On* (encendido), en esta posición el equipo empieza a estimular y se repite el Período

Maestro de manera infinita. Para detener la estimulación es necesario regresar al estado de *Standby*. En la posición inferior *Single Cycle* (Ciclo Único) el equipo realiza un(1) Período Maestro de estimulación y luego el equipo queda como si estuviera de nuevo en la posición *Standby*. Para volver a estimular, es necesario llevar el switch de nuevo a la posición central *Standby* para luego hacer una nueva selección.

El equipo funciona aún cuando los usuarios colocan parámetros que pudieran llegar a ser incoherentes o límites. A continuación se numeran algunos casos límites y el funcionamiento del equipo.

- **Período Maestro = 0 :** El equipo se mantiene en standby en 0V y no emite pulsos.
- **Delay > Período Maestro :** El equipo se mantiene en 0V y no emite pulsos.
- **Modo: Pulso Simple, Duración Pulso > Período Maestro, Delay = 0 :** El equipo se mantiene constantemente en un valor DC fijo establecido por la Amplitud del Pulso
- **Modo: Pulso Simple, (Delay + Duración Pulso) > Período Maestro :** El equipo modifica la duración del pulso para respetar el Período Maestro de manera que la nueva duración del pulso vendrá dada por *Período Maestro - Delay*
- **Modo: Tren de Pulso, (Delay + Duración de Tren) > Período Maestro :** El equipo modifica la duración del tren para respetar el Período Maestro de manera que la nueva duración del tren vendrá dada por *Período Maestro - Delay*
- **Modo: Tren de Pulso, Duración del Tren > Período Maestro, Delay = 0 :** El equipo modifica la duración del tren para respetar el Período Maestro de manera que *Duración del Tren = Período Maestro*
- **Modo: Tren de Pulso, Período del Tren > Duración del Tren :** El equipo funciona como si estuviera en modo pulso simple y se produce un único pulso, además si *Duración Pulso > Duración del Tren*, el pulso tendrá un tiempo igual a Duración del Tren
- **Modo: Tren de Pulso, Duración de Pulso > Período del Tren :** El equipo funciona como si estuviera en modo simple donde el pulso tendrá como tiempo Duración del Tren

Capítulo 3

Marco Metodológico

Luego de haber definido los conceptos básicos para el desarrollo del instrumento, se establecerán los pasos para lograr alcanzar los objetivos planteados anteriormente (Capítulo 1), estos pasos se dividieron en etapas.

3.0.1. Etapa 1: Estimulación

En la primera etapa se desarrolló la base principal que sirve como esqueleto y soporte para el resto del trabajo. Esta etapa consta de 3 partes fundamentales que son: la interfaz gráfica, el microcontrolador y la comunicación serial.

Interfaz Gráfica

Se da inicio con una fase de aprendizaje y manejo de la IDE Qt v.5.5 y el desarrollo de GUI, para lograr diseñar una interfaz que sea visualmente parecida al estimulador eléctrico "Ortec 4710 Dual Channel Stimmulator". Esta etapa necesita un aprendizaje del lenguaje C++ y un entendimiento sobre la programación orientada a objetos. La interfaz gráfica permitirá a los futuros usuarios seleccionar los parámetros de estimulación a utilizar y también podrán observar gráficamente los datos que proceden de sus experimentos; de una manera sencilla y amigable que no requiere de conocimientos en programación para su uso. Esta interfaz facilita a los usuarios experimentados del Ortec 4710 su manipulación sin necesidad de un entrenamiento previo y brinda una primera interacción agradable a los que por primera vez harán uso del mismo. Se presume que los estudiantes al aprender a utilizar el equipo Ortec en seguida son capaces de utilizar el equipo virtual y vice-versa.

Microcontrolador

Una vez que la interfaz gráfica está creada, es necesario ahora aprender sobre la placa Arduino UNO R3. El Arduino viene con un Ambiente de Desarrollo Integrado (en inglés IDE) que esta basado en el lenguaje de programación C, pero tiene funciones propias de

Arduino, de manera que es necesario aprender a utilizar estas funciones. El microcontrolador es el encargado de realizar las tareas de envío de la señal de estimulación, convertir la señal analógica del transductor de tensión en una señal digital y también enviar los datos recibidos a la computadora vía puerto serial.

Comunicación Serial

Después que la interfaz y el microcontrol han sido implementados y estudiados, viene el paso importante de poder comunicar ambos dispositivos via comunicación serial. Para ello es necesario aprender las funciones de comunicación que vienen incluidas en el programa de desarrollo Qt v.5.5 y las presentes en Arduino. Dentro del Arduino encontramos las instrucciones de la librería *Serial*, la cual posee varias funciones destinadas a este fin. Qt v.5.5 también trae librerias propias especializadas en comunicación serial llamadas *QSerialPort* y *QSerialInfo*. Para evaluar el desempeño de la comunicación se utilizó el envío de secuencias controladas de números hexadecimales verificando su recepción. Esto permitió buscar el mecanismo adecuado para el envío y recepción de datos sin pérdida de información.

Una vez realizadas estas 3 bases fundamentales se puede abordar completamente el problema del estimulador. Utilizando la interfaz gráfica el usuario debe ajustar los parámetros de estimulación deseados. Éstos parámetros deben ser enviados al Arduino por comunicación serial, para que éste ejecute la estimulación ajustada a los parámetros seleccionados por el usuario. La estimulación se realiza colocando un uno(1) lógico (5V) o un cero(0) lógico (0V) en uno de los pines del Arduino para generar la señal cuadrada. La estimulación necesariamente debe mantener los parámetros fijados por el usuario en la interfaz gráfica.

Al tener la estimulación a punto, se diseñará un circuito que suministrará los niveles de voltaje y corriente necesarios para realizar las prácticas fisiológicas del LDB. Dicho circuito debe elevar el voltaje para tener un valor máximo de 10 V y tener un seleccionador de escala de 10V, 1V, 100mV y 10mV para que el usuario pueda colocar el valor de voltaje deseado. El usuario dispondrá de un control analógico no virtual para colocar un valor dentro del rango de cada escala.

3.0.2. Etapa 2: Adquisición de datos

Para poder adquirir la señales respuesta del músculo, se utiliza un transductor de tensión. Se requiere adecuar la señal del transductor de tensión a los valores que necesita el Arduino para la conversión A/D (entre 0 y 5V); para ello se diseñó un circuito que la condiciona. Luego de esta etapa se realizó la conversión de datos por el Arduino, tratando de que sea lo más rápido posible. Estos datos deben ser enviados por comunicación serial a la computadora.

En la computadora, se debe recibir estos datos y es necesario construir la interfaz

que sea capaz de dibujarlos en tiempo real y guardarlos en el registro de memoria de la computadora. Para ello, será necesario documentarse sobre las herramientas gráficas de Qt v.5.5 y también conocer las funciones que permiten generar archivos.

Con el objetivo de asegurarnos que la adquisición de datos es correcta, se usó de un generador de ondas para obtener una señal senosoidal de frecuencia conocida. Esto permitió ajustar las herramientas de dibujos en tiempo real, estimar la velocidad de muestreo y además detectar la presencia o no de alguna discontinuidad en la señal.

3.0.3. Etapa 3: Integración de estimulación, adquisición y transmisión

Despúes de obtener las señales de estimulación y la adquisición por separado, se integraron los 3 procesos importantes que esta ejecutando el microcontrolador: la estimulación, la conversión A/D para adquirir los datos y la transmisión de los datos a la computadora. Para ello, se utilizó la estimulación más rápida lograda con nuestro dispositivo, mientras se convierte una onda senosoidal de frecuencia conocida, y se observa su gráfica en tiempo real dentro la GUI. Así se consiguió optimizar el programa para lograr estas funciones simultáneamente.

3.0.4. Etapa 4: Evaluación del desempeño

Señal de Estimulación

Una vez obtenido el dispositivo funcional, se realizó una serie de evaluaciones para ponerlo a punto. Primero se evaluaron los casos límite mencionados en el capítulo 2, sección 2.4. Posteriormente se evaluaron los tiempos estrictos asociados a la señal de estimulación mediante la observación a través de un osciloscopio marca Tektronix (TDS 1002 Doble Canal, Almacenamiento digital 60MHz, 1 G muestra por segundo).

Adquisición de datos

Para adquirir los datos se configuró el Arduino para la recepción de datos por uno de sus puertos analógicos. Se hicieron pruebas en la frecuencia de muestreo, amplitud del registro usando un generador de ondas marca Goldstar FG8002. Los datos fueron grabados en un archivo de texto plano (.txt) para su posterior visualización y análisis siguiendo un formato que facilita su graficación bajo cualquier paquete comercial.

Capítulo 4

Resultados

Se empezará mostrando los elementos de la interfaz gráfica en función de su operatividad, para luego mostrar el desenlace de cada uno de los elementos señalados durante el Marco Metodológico.

4.1. Interfaz gráfica: Similitud con el equipo Original

Como es evidente en las imagenes 4.1 y 4.2 los controles en la GUI tienen una distibución espacial que se observa en el panel frontal del Ortec 4710.

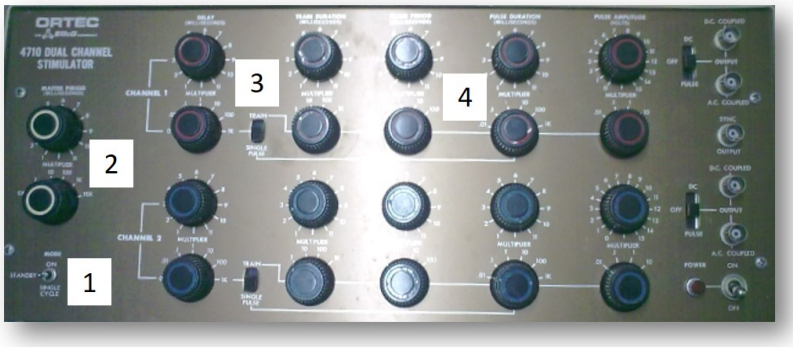


Figura 4.1: Ortec 4710 Dual Stimmulator

En las figuras 4.1 y 4.2 se señala con números los detalles de diseño generales que se mantuvieron en el desarrollo de la Interfaz Gráfica los cuales son:

1. Interruptor de inicio con 3 posiciones, mencionado en la sección 2.4: Standby, On y Single Cycle;

2. Dos diales de control para controlar cada parámetro, como ya se describió;

3. Interruptor para seleccionar el tipo de señal de estimulación; pulso simple o tren;

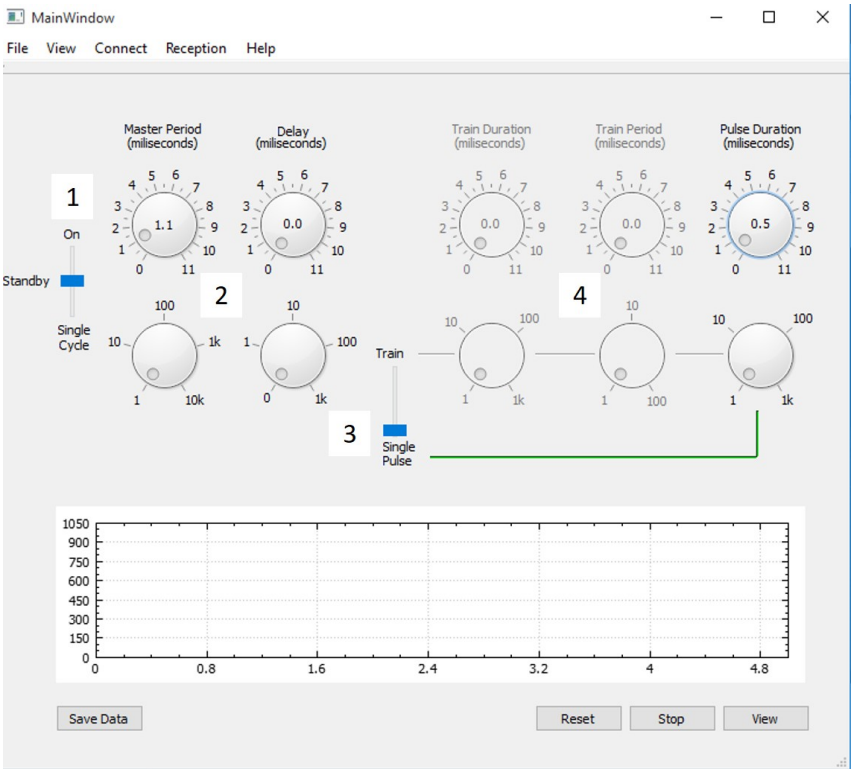


Figura 4.2: Interfaz Gráfica desarrollada mostrada bajo el sistema operativo Windows 10

4. Finalmente se observan los controles para los parámetros de la señal mencionados anteriormente. El único parámetro que no se manejará digitalmente es la Amplitud del Pulso. Este parámetro se controla en la etapa analógica del sistema.

4.2. Estimulador: manejo de tiempos estrictos y niveles adecuados de corriente

En las prácticas de fisiología el tiempo mínimo que se utiliza para estimular es del orden de milisegundos. Por lo tanto fue necesario construir un dispositivo capaz de manejar esos tiempos. En una primera aproximación se pensó en utilizar el ordenador para ello bajo la plataforma Windows, sin embargo, por problemas con la jerarquización de los procesos se dificulta lograr el control de estos tiempos. Por otro lado, además por ser Windows un sistema privativo se dificulta el acceso a la información de sus librerías, lo que genera limitaciones a la hora del desarrollo de este proyecto. Al hacer pruebas con el sistema Windows 7, la estimulación mínima lograda era del orden de 16ms con una incertidumbre muy grande ya que en algunos casos la estimulación mínima obtenida podía llegar a 47ms. En cambio, en el sistema operativo Linux, es posible manejar estos tiempos estrictos. Sin embargo, para mantener compatibilidad entre plataformas, se decidió utilizar un microcontrolador para lograr una independecia de estas tareas en relación al sistema

operativo.

Por razones ya expuestas en el Capítulo 2, se decidió utilizar el Arduino como microcontrolador. En la IDE del Arduino se pueden encontrar herramientas para desarrollar *delays* del orden de milisegundos. Los *delays* dentro del programa de un microcontrolador, significa que durante el tiempo que se estipuló el microcontrolador no va a poder realizar ninguna otra actividad. Como además de estimular, se desea recibir datos a través del conversor A/D y también enviar los mismos al computador utilizando comunicación serial, realmente lo que se necesita son *Interrupciones* que puedan configurarse para ocurrir en tiempos estrictos definidos por los usuarios. Las *Interrupciones* son rutinas que suspenden el funcionamiento de la rutina principal del programa y ejecutan una pequeña tarea antes de regresar al programa principal. Las *Interrupciones* pueden ser originadas por eventos externos, como variaciones en pines específicos, o por eventos internos como por ejemplo terminar la conversión A/D.

Se utilizó la librería creada por la comunidad Arduino llamada TimerOne [13]. Esta librería posee funciones que crean interrupciones en el tiempo deseado en el orden de microsegundos. La librería utiliza el contador interno del microcontrolador de 16-bits que genera una interrupción al desbordarse. El programa hace los cálculos necesarios para que el desbordamiento y la interrupción se genere en el tiempo solicitado por el usuario en microsegundos, específicamente el límite inferior es de 1us y el límite superior es de 11s. Esta lógica se utiliza para generar la señal cuadrada solicitada por el usuario utilizando las interrupciones para poner un uno lógico o un cero lógico en el pin de salida.

Luego fue necesario acondicionar la señal para poder estimular con los valores de voltaje y corriente adecuados. Se creó un divisor de voltaje para seleccionar entre varias escalas con valores máximos de 10V, 1V, 100 mV y 10mV para que el usuario pueda colocar el valor de voltaje deseado con cierta facilidad. Para seleccionar los niveles de voltaje adecuado se tomaron en consideración trabajos hechos anteriormente sobre la resistividad del tejido muscular y nervioso específicamente en sapos [14] [15]. En la figura 4.3 se puede observar el circuito diseñado para estimular con los niveles de voltaje y corriente adecuados.

Al colocar los parámetros de estimulación se pudo corroborar que el equipo es capaz de respetar los valores de estimulación seleccionados. Se realizaron pruebas colocando diferentes valores en los parámetros de estimulación y se midieron los tiempos con el Osciloscopio Tektronix TDS 1002. En la figura 4.4 se puede observar el comportamiento del equipo construido al exigirle la estimulación mínima que se puede alcanzar con él: 100us. En la figura 4.5 se puede observar un tren de pulso con las siguientes características: Duración del pulso: 1ms, Período del Tren: 2ms, Duración del Tren: 5.5 ms y Período Maestro: 7,7 ms

Para corroborar que el circuito funciona de manera similar al equipo Ortec, se decidió hacer una comparación directa. Para las pruebas, en el equipo se puso como parámetro una duración de pulso superior al período maestro. De esta manera el equipo funcionó como si fuera un voltaje DC continuo como se explicó en la sección 2.4. Se puso a funcionar ambos

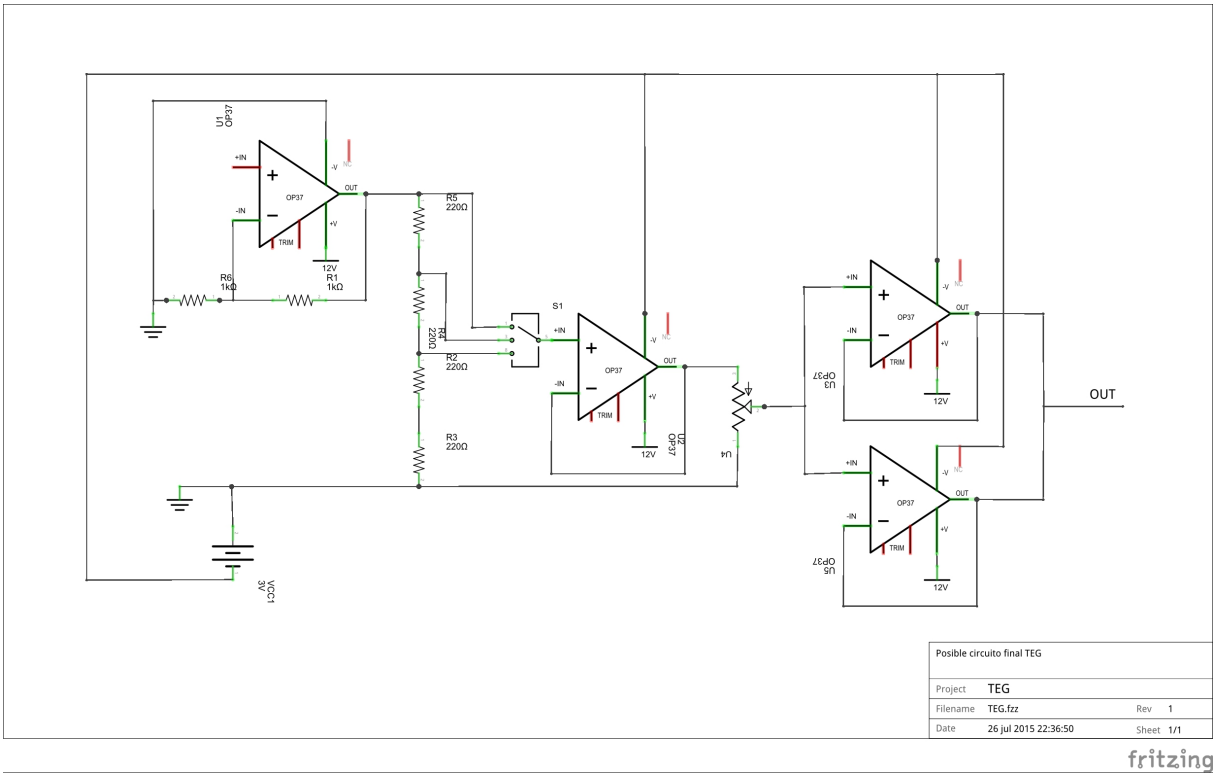


Figura 4.3: Esquema del circuito para estimular

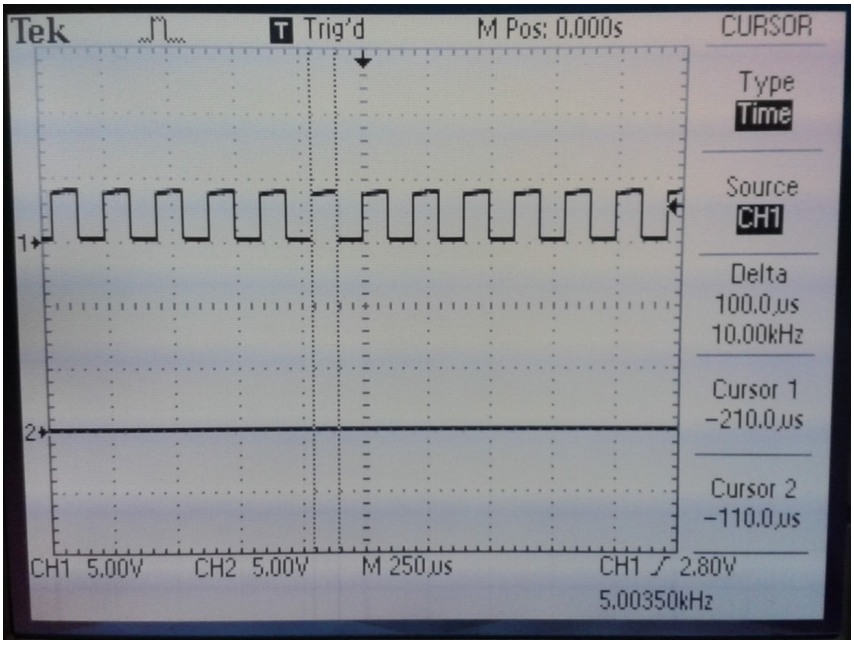


Figura 4.4: Estimulación mínima posible por el equipo diseñado: 100us

equipos con un valor de 10V, el valor máximo, para poder obtener la máxima potencia que pueden entregar ambos equipos. En las figuras 4.6 y 4.7 se colocan los resultados

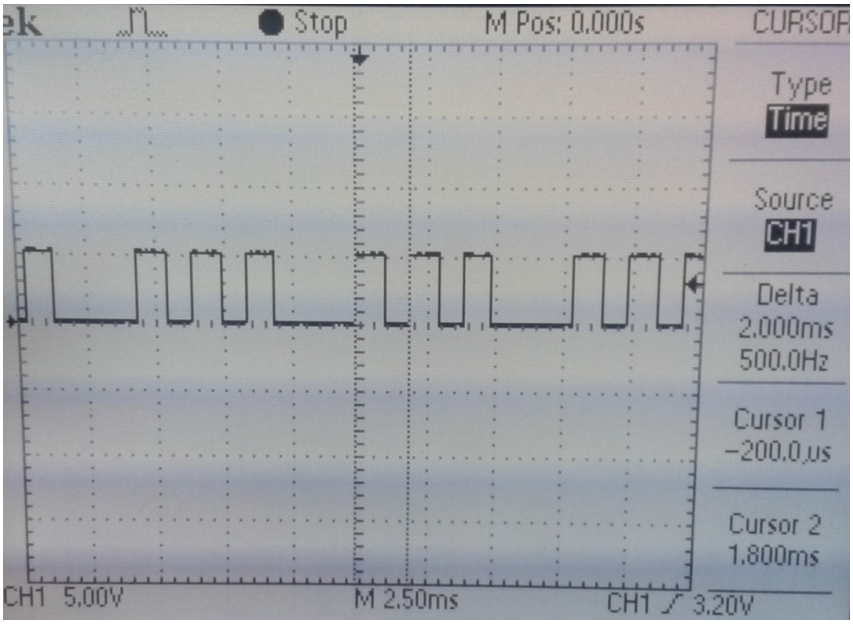


Figura 4.5: Ejemplo de Tren de Pulso

experimentales de dichas pruebas. Es importante resaltar que la potencia máxima en ambos equipos se alcanza en valores cercanos a los 100 Ω. De acuerdo a experimentos realizados anteriormente [14] [15] la resistividad promedio observada en la materia blanca (axones) del cerebro de una vaca y un conejo es de 580 Ω - cm. Haciendo uso de este resultado se puede sugerir que la distancia entre los electrodos de estimulación debería ser menor a 1 cm para aprovechar la mayor potencia que pueden entregar ambos equipos.

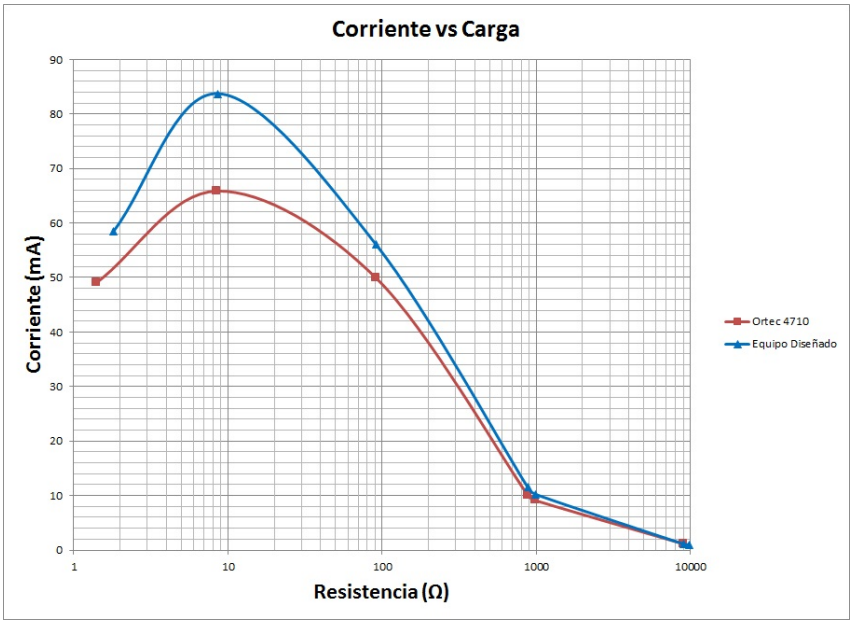


Figura 4.6: Comparación de la corriente entregada por el equipo Ortec y el equipo diseñado

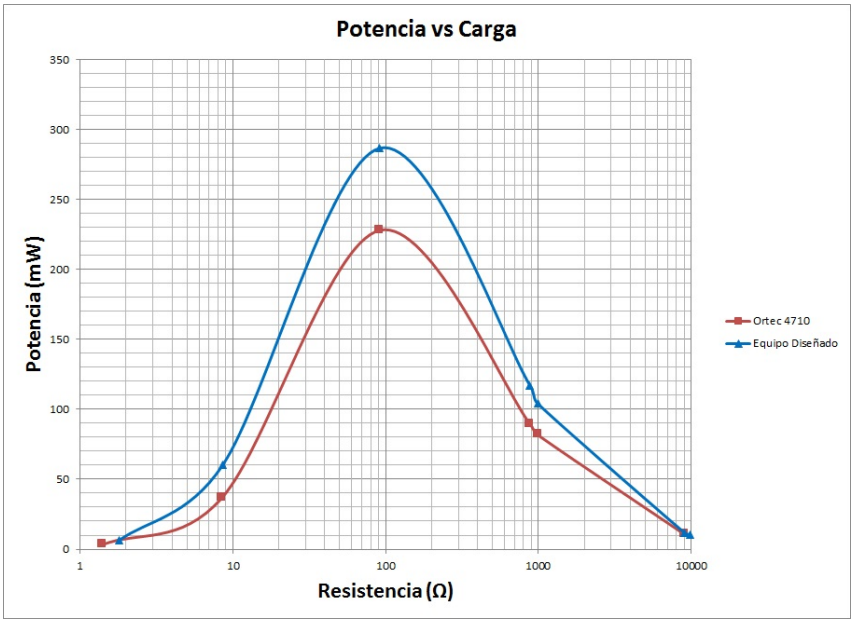


Figura 4.7: Comparación entre la potencia entregada por el equipo Ortec y el equipo diseñado

4.3. Conversión Análoga-Digital, graficación en tiempo real y almacenamiento de datos

El microcontrolador Atmel 328p, presente en la placa Arduino UNO R3 tiene un conversor A/D de 10 bits y de 0 a 5V, por lo que es necesario adecuar la señal de entrada a ese intervalo.Por otro lado, el transductor mecano-eléctrico se utiliza para monitorear la respuesta de tensión del músculo y convertirla en una señal eléctrica. Dicho transductor viene acoplado para un pre-amplificador que produce niveles de voltaje acotados. El de 4 oz. tiene una respuesta máxima de 418mV al halar y -428mV al empujar. En el caso del de 16 oz la respuesta máxima es de 512 mV al halar y -408 mV al empujar. En las prácticas de fisiología solamente se requiere utilizar el sensor en un sentido. De esta manera sabemos que la respuesta del transductor es positiva y tiene un valor máximo de 512 mV en el caso de uno de los transductores. Por esta razón se diseñó un pequeño amplificador que permite aumentar la parte positiva de la señal que proviene del pre-amplificador a los niveles de voltaje que necesita el microcontrolador.

Para convertir los datos de manera rápida se manipularon directamente los registros asociados a la conversión A/D en vez de utilizar las funciones propias de Arduino. Esto se debe a que las funciones propias de Arduino tardan más tiempo en ejecutarse, ya que realizan chequeos de seguridad para asegurarse que los usuarios no configuren las cosas de manera errónea [16].

Una vez que los datos se han digitalizado, son enviados al computador utilizando el puerto USB como puerto serial. El programa Qt viene con la capacidad de realizar dibujos

y gráficos utilizando Open GL, el cual es un software dedicado a desarrollar imágenes 2D y 3D. Sin embargo, se utilizó la aplicación QCustomPlot, la cual es una aplicación en C++ para Qt que permite hacer gráficos y visualización de datos de manera sencilla [17]. Finalmente estos datos pueden ser almacenados en archivos de extensión *.txt* para que el usuario pueda manipularlos.

En las figuras 4.8 y 4.9 se puede observar la transformación de una señal senosoidal de prueba de 0.5 kHz de frecuencia. Variaciones en frecuencia y amplitud pueden ser observadas en tiempo real. Otro aspecto importante a resaltar en la figura 4.8 es el número de puntos en un período de la señal. Al tener aproximadamente 10 puntos en un ciclo de una señal de 500 Hz, podemos asumir que la velocidad de muestreo es de aproximadamente 5.000 muestras por segundo. Esto concuerda con el valor obtenido experimentalmente por medio de los Osciloscopio Textronik de 5.814 muestras por segundo. El procedimiento para determinar este resultado se explica en la sección 4.4

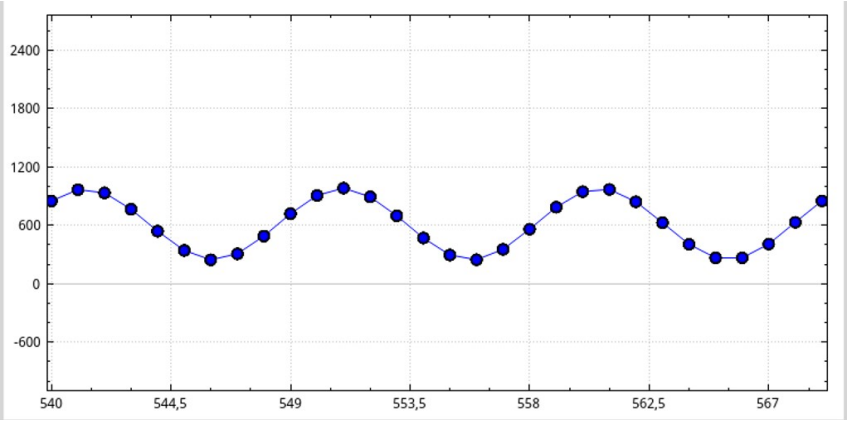


Figura 4.8: Señal senosoidal de 0.5 kHz mostrada en tiempo real

4.4. Integración de las etapas de: estimulación, conversión A/D y envio de datos

Uno de los retos que se abordó en el presente trabajo fue lograr que el equipo lograra digitalizar, enviar los datos a la PC y producir la señal de estimulación, de manera simultánea. Para ello, siguiendo los antecedentes de proyectos relacionados con el desarrollo de un osciloscopios por medio del Arduino [18], se decidió hacer un buffer con la mayor cantidad de datos que se puede almacenar en un *arreglo* con en el Arduino UNO R3: 244 datos[19]. Sin embargo, mientras se transfiere un bloque de datos a la PC, la conversión A/D se detiene y se pierden datos. En la Figura 4.10 se puede observar la falta de continuidad en la digitalización de una onda senosoidal al momento de mandar los datos a la computadora.

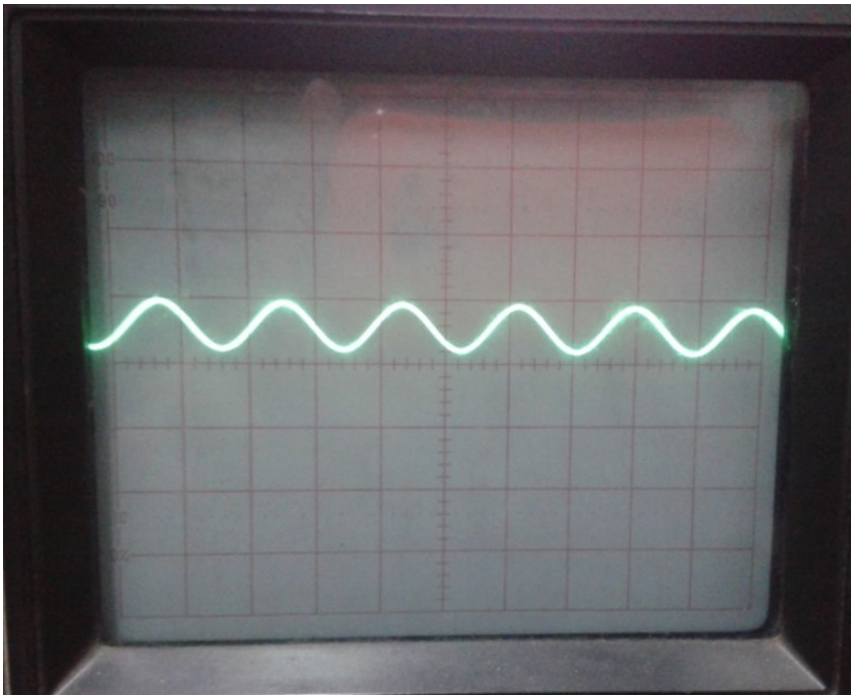


Figura 4.9: Señal senosoidal de 0.5 kHz de prueba para ser mostrada en tiempo real en la GUI

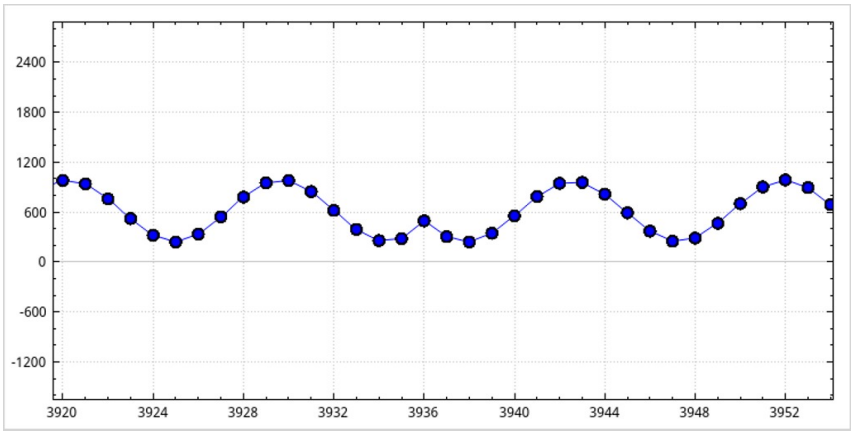


Figura 4.10: Interrupción en la señal adquirida en el momento en que se envían los datos via comunicación serial

Por esta razón se decidió que para poder tener una visualización en tiempo real continua, era necesario convertir un dato a la vez y mandarlo por puerto serial a la computadora. Luego se buscó la mayor velocidad a la cual se pueden realizar las 3 tareas simultáneamente. Para ello, se sometió el equipo a varias pruebas donde manteníamos la frecuencia más alta de estimulación y variabamos la velocidad del conversor de A/D y la velocidad de transmisión de datos. Para variar la velocidad del conversor A/D utilizamos distintos valores del *Preescalador* , como se explicó en la sección 2.3.3. La velocidad

de transmisión serial se varió utilizando los siguientes valores: 9.600 Bps, 57.600 Bps y 115.200 Bps. Además de variar estos valores, para aumentar la velocidad de estimulación se evitó el uso las funciones propias de la Arduino IDE (digitalWrite(pin)). Se utilizaron funciones que escriben directamente en los puertos asociados a los pines para una mayor velocidad [20].

Para analizar cómo se desarrollaban los procesos, se utilizó otro puerto del Arduino. Se modificó el programa del Arduino para que antes que el programa mandara los datos vía comunicación serial, se elevara el voltaje en ese puerto, y al terminar de mandar los datos bajara el voltaje. De esta manera la señal tenía las siguientes características: cuando el Arduino estaba mandando datos, el puerto estaba elevado (5V), y cuando terminaba el envío de datos y empezaba la conversión A/D de datos el puerto estaba bajo (0V). En la figura 4.11 se puede observar en el Canal 1 del osciloscopio (superior) la señal de estimulación, y en el Canal 2 (inferior) se tiene la señal que acabamos de describir. En este caso se tiene un valor de preescalador de 64, velocidad de transmisión de 115.200 Bps. Se puede observar que los procesos no se interrumpen y por eso se logra la integración de todos los procesos. En la figura 4.12 se puede observar cuando los procesos se interrumpen causando fallas en la estimulación. En este caso la velocidad de transmisión es de 57.600 y el preescalador tiene un valor de 128. Es importante resaltar la falta de regularidad en los tiempos de envío de datos. El envío de datos se realiza mediante librerías de Arduino y cuando las interrupciones de la estimulación o de la conversión A/C se solapan con la transmisión de datos, empiezan a ocurrir errores y los tiempos de transmisión de datos se vuelven irregulares. En la tabla 4.1 se tiene un registro de las pruebas realizadas para encontrar el óptimo de las 3 funciones, determinándose que la prueba 3 posee los valores de mayor velocidad sin errores. Para determinar la frecuencia de muestreo se midió en el osciloscopio el tiempo entre el final de una transformación A/D y la siguiente. Al conocer que se transforma un dato en este tiempo, tenemos una relación directa de la frecuencia.

Prueba	Preescalador	Bps (Baudios por segundo)	Muestras por segundo	Errores en la estimulación
1	128	57.600	2.941	Si
2	128	115.200	4.000	No
3	64	115.200	5.814	No
4	32	115.200	5.814	Si

Cuadro 4.1: Búsqueda experimental de las velocidades para optimizar las 3 funciones simultaneamente del instrumento.

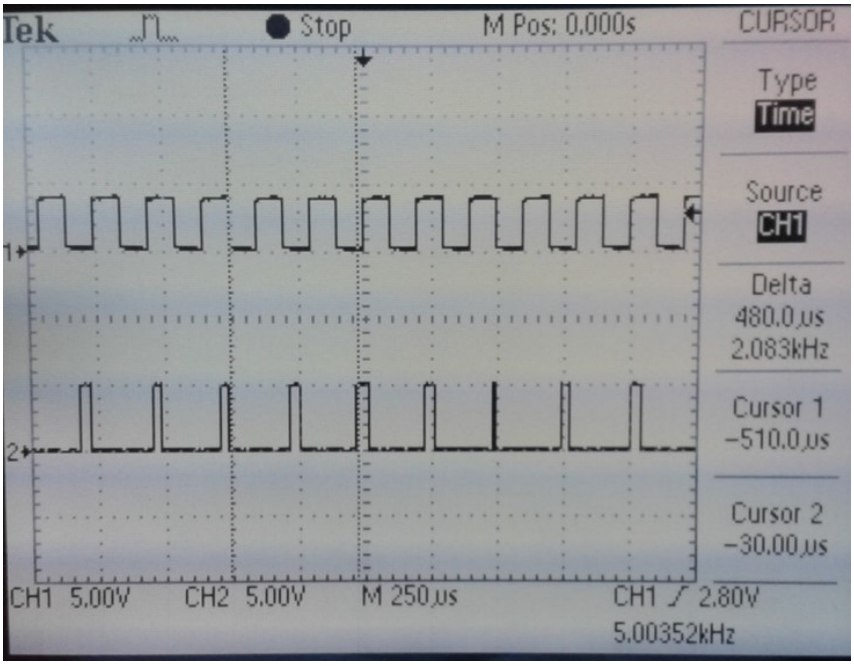


Figura 4.11: Visualización simultánea de la estimulación y el envío de datos en el Arduino. Los procesos no se interrumpen.

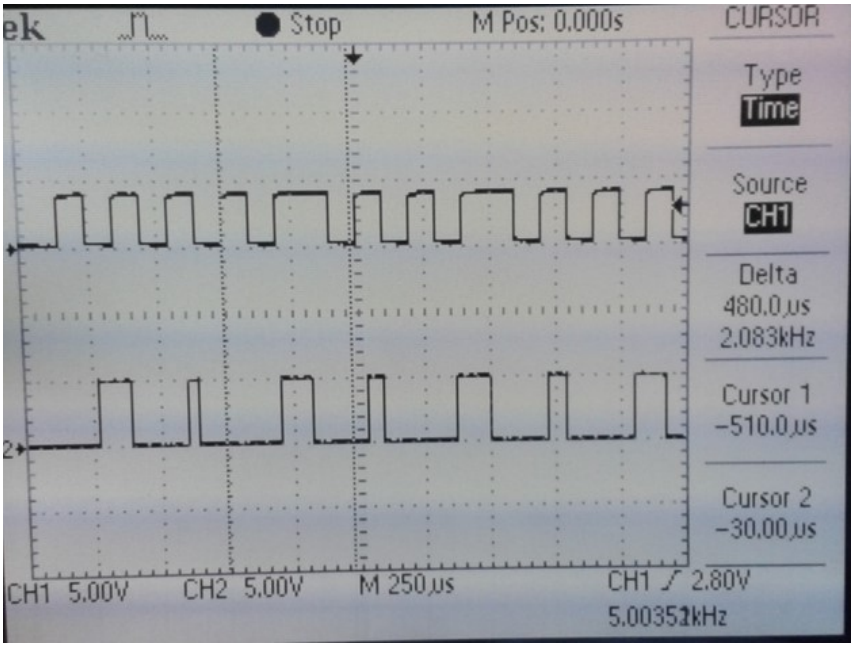


Figura 4.12: Visualización simultánea de la estimulación y el envío de datos via comunicación serial en el Arduino. Los procesos se interrumpen

Capítulo 5

Conclusiones y consideraciones finales

En el quinto y último capítulo encontraremos las conclusiones y consideraciones finales para cerrar todas las ideas expuestas en el desarrollo del trabajo.

5.1. Conclusiones

- Se desarrolló un equipo con una GUI que emula la apariencia del panel de control del Ortec 4710, capaz de controlar los parámetros de tiempo de la señal de estimulación.
- Se acondicionó la señal de estimulación para tener valores de voltaje y corriente cumpliendo con las especificaciones del equipo Ortec 4710.
- Se evaluó y se demostró la capacidad del equipo de estimular bajo los parámetros establecidos por el usuario haciendo uso de la GUI.
- Se evaluó y se comprobó la posibilidad de recibir datos, visualizarlos en tiempo real y guardarlos en un archivo de texto plano capaz de visualizarse en paquetes de graficación comercial.
- Se determinó el intervalo de operatividad donde las funciones del equipo diseñado no presentan errores: velocidad de transmisión mínima de 115.200 Bps, preescalador de la conversión A/D con un valor máximo de 128 y mínimo de 64, mientras se estimula con la velocidad máxima de 100us como ancho del pulso.

5.2. Consideraciones finales

- La interfaz gráfica abre la posibilidad a muchas aplicaciones como la implementación del procesamiento de datos o como una herramienta para mejorar la actividad

docente. Por ejemplo, la aplicación pudiera ser capaz de mostrar un gráfico de la estimulación que se esta aplicando, permitiendo a los estudiantes visualizar la señal de estimulación antes de aplicarla.

- El conversor del microcontrolador Atmel presente en la placa Arduino UNO R3 de 10-bits, no permite ver la señal respuesta de la conducción nerviosa y esta limitado a la respuesta mecánica del músculo en las prácticas de neuro-fisiología. Se recomienda utilizar una placa Arduino con un conversor de mayor velocidad o utilizar un conversor A/D externo.

Apéndice A

Diagrama de flujo del programa Arduino

En este apéndice se puede observar un diagrama de flujo del funcionamiento del programa de Arduino

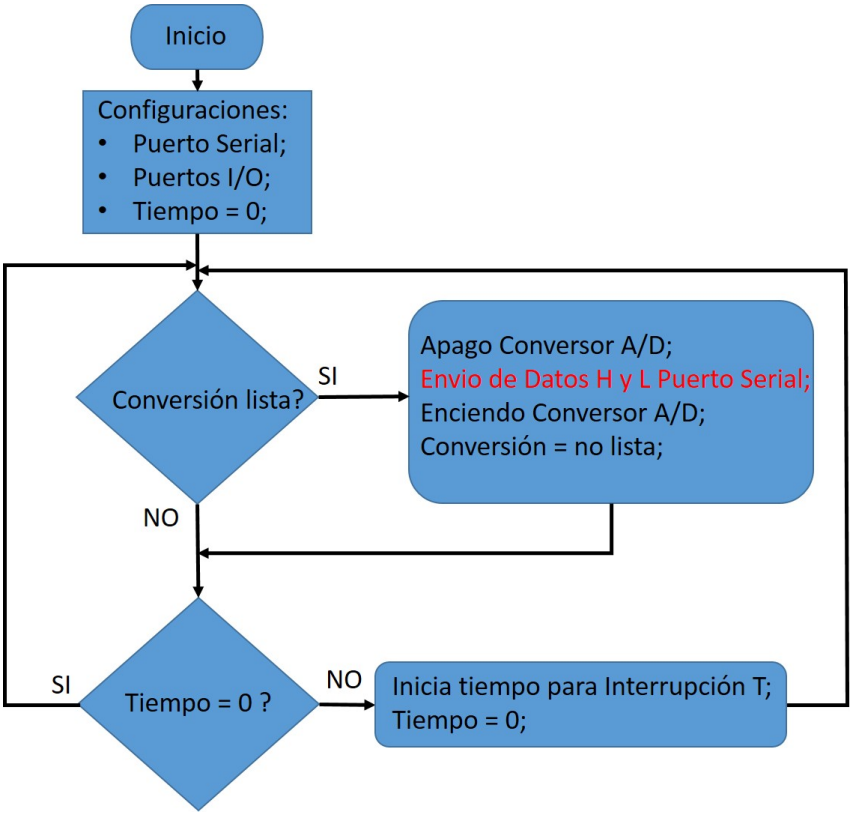


Figura A.1

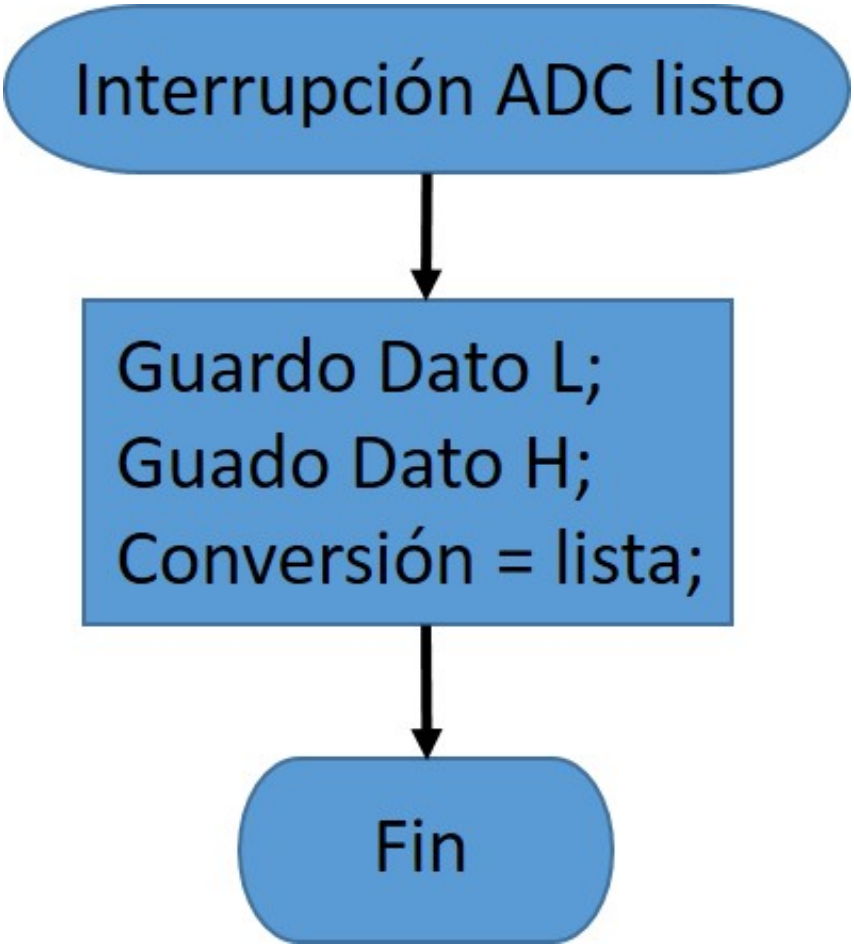

Interrupción ADC listo
Guardo Dato L;
Guado Dato H;
Conversión = lista;
Fin

Figura A.2

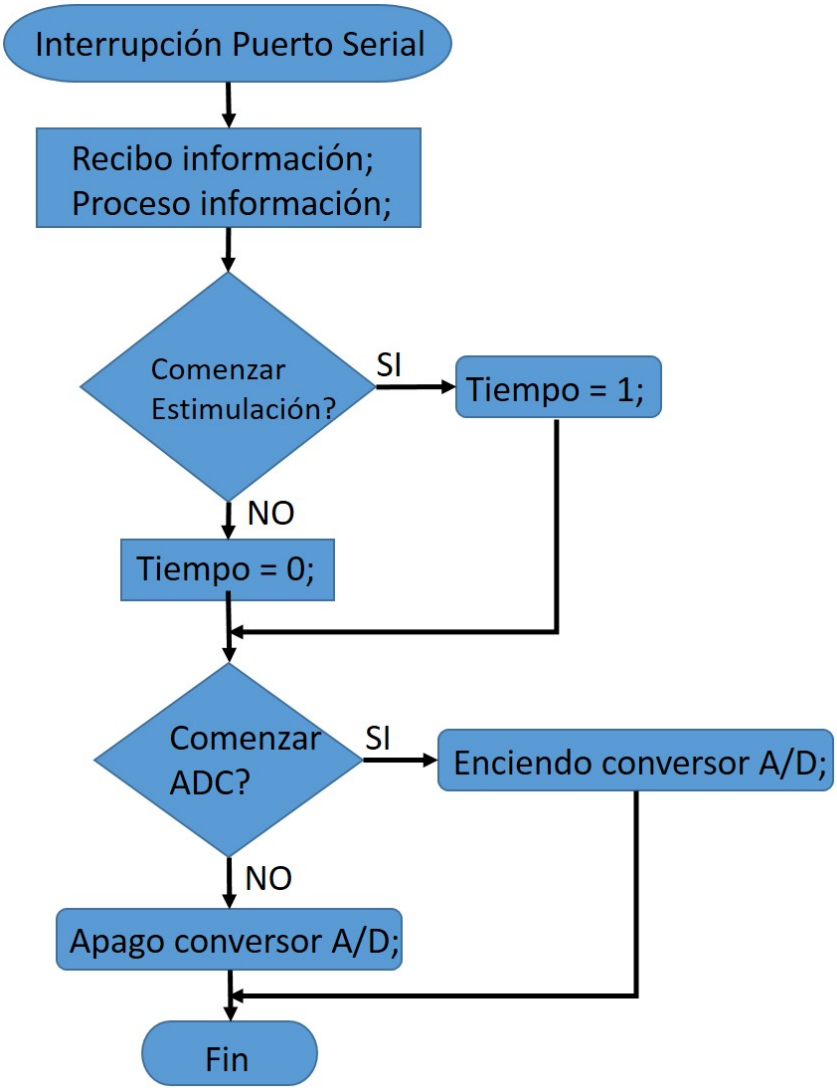

Interrupción Puerto Serial
Recibo información;
Proceso información;
Comenzar Estimulación?
SI
Tiempo = 1;
NO
Tiempo = 0;
Comenzar ADC?
SI
Enciendo conversor A/D;
NO
Apago conversor A/D;
Fin

Figura A.3

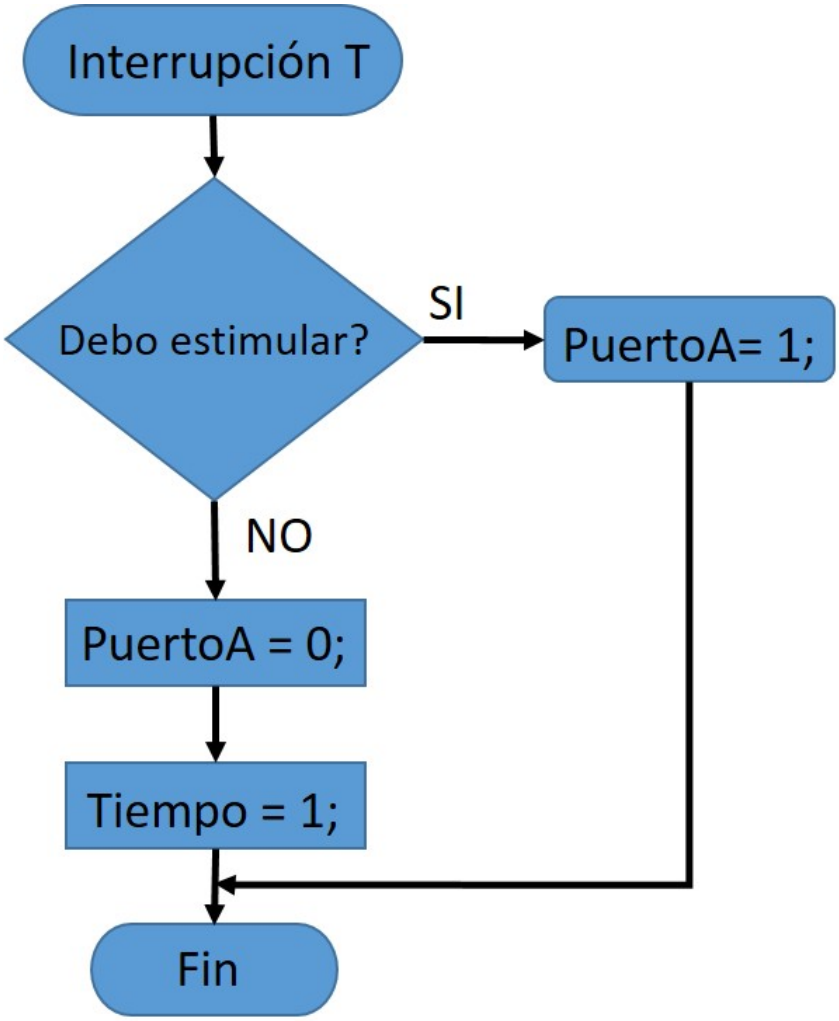

Interrupción T
Debo estimular?
SI
PuertoA= 1;
NO
PuertoA = 0;
Tiempo = 1;
Fin

Figura A.4

Apéndice B

Código Fuente Arduino

Colocamos el código fuente del programa Arduino diseñado, para que pueda ser utilizado siguiente la licencia y filosofía Open Source.

Estimulador2.ino

```
//-------------------------------------------------------------------------------------
//Includes
//-------------------------------------------------------------------------------------
#include <TimerOne.h>
#include "Variables.h"

//-------------------------------------------------------------------------------------
//Variables Globales
//-------------------------------------------------------------------------------------

Variables v;
Variablestotales vt;
const int outPin = 9;
const int sincroPin = 8;
int bitestimulacion = 7;
int bitsincronizacion = 4;
volatile int tiempo = 0;
volatile int counter = 0;
volatile int extrat = 0;
volatile int locura =0;
volatile int duty = 0;
volatile int rep = 0;
volatile int veceshigh = 0;
volatile int veceslow = 0;
volatile int cuentatotal = 0;
```

```
volatile int final = 0;
volatile int parte = 0;
volatile int finalrapido = 0;
volatile int batisenal = 0;
volatile unsigned int datoL = 0;
volatile unsigned int datoH = 0;
volatile int datolisto = 0;
volatile int counterADC = 0;
volatile uint8_t bufferADC[CAPACIDAD_BUFFER];
volatile int numerobloque = 0;
volatile uint8_t bufferficticio[CAPACIDAD_BUFFER];
volatile int contadorcillo1 = 0;
volatile int contadorcillo2 = 0;
volatile int contadorcillo3 = 0;
volatile int crescendo = 1;

//--------------------------------------------------------------------------------
//Rutinas Principales
//--------------------------------------------------------------------------------

void setup()
{
ADCinicio();
Serial.begin(115200);
//Serial.begin(57600);
/*DDRB |= B00111111;*/
DDRD |= B11111100;
/*pinMode(outPin, OUTPUT);
pinMode(sincroPin, OUTPUT);*/
Timer1.initialize(1000);
pinMode(3, OUTPUT);
}

void loop()
{
if (datolisto == 1)
{
/*Serial.write((uint8_t *)bufferficticio, 2);
Serial.write((uint8_t *)bufferficticio + 2, 2);
datolisto = 0;*/
```

```
cbi(ADCSRA, ADEN);
PORTD |= B01000000;
Serial.write((uint8_t *)bufferADC + (CAPACIDAD_BLOQUE * numerobloque), CAPACIDAD_BLOQU
PORTD &= ~(B01000000);
sbi(ADCSRA, ADEN);
sbi(ADCSRA, ADSC);
numerobloque++;
if(numerobloque == NUMERO_BLOQUES)
{
numerobloque = 0;
}
datolisto = 0;

/*datoH = datoH << 8;
unsigned int mensaje = datoH | datoL;
String ari = String(mensaje, HEX);
while (ari.length() < 3)
{
ari = "0" + ari;
}
Serial.print(ari);*/

datolisto = 0;
}

if (tiempo != 0)
{
switch(tiempo)
{
case 1:
Timer1.detachInterrupt();
Timer1.attachInterrupt(tdelay, vt.delaytotal);
tiempo = 0;
break;
case 2:
Timer1.detachInterrupt();
if (v.modoest == 's'|| v.modoest == 'p')
{
Timer1.attachInterrupt(durap, vt.pulsedtotal);
}
else if (v.modoest == 'u' || v.modoest == 't')
```

```
{
if (final == 1)
{
Timer1.attachInterrupt(duratfinal, vt.trestante);
}
else if (parte == 1)
{
Timer1.attachInterrupt(durat1, vt.pulsedtotal);
}
else
{
Timer1.attachInterrupt(durat2, vt.tlow);
}
}
tiempo = 0;
break;
case 3:
Timer1.detachInterrupt();
Timer1.attachInterrupt(textra, vt.tiempoextra);
tiempo = 0;
break;
}
}
}
```

Calculos.cpp

```
#include "Variables.h"
#include <TimerOne.h>

unsigned long procesameste(char a[2], char b)
{
long dial1;
String parapasar;
for (int i = 0; i < 2; i++)
{
const char ch = a[i];
parapasar += ch;
}
char *parapasar2 = new char [2];
strcpy (parapasar2, parapasar.c_str());        /*esto cambia de string a c-string*/
```

```
dial1 = strtol(parapasar2, NULL, 16);

int prueba = b - '0';     /*esto pasa de char a int*/
prueba = prueba - 1;

unsigned long dial2 = 1;
if (prueba != -1)
{
for (int i = 0; i < prueba; i++)
{
dial2 = dial2 * 10;
}
}
unsigned long total;

total =dial1 * dial2;
delete [] parapasar2;
return total;
}

//int calcularduty()
//{
//  /*int prescale[] = {0,1,8,64,256,1024};
//  int prescaleValue = prescale[Timer1.clockSelectBits];
//  long precision = (F_CPU / 128000) * prescaleValue;
//  long period = precision * ICR1 / 100;
//  int hola = map(vt.pulsedtotal, 0, period, 0,1024);*/
//  int hola = map(vt.pulsedtotal, 0, vt.trainptotal, 0, 1024);
//  return hola;
//}

void sacarcuentas()
{
if (vt.masttotal == 0)
{
v.modoest = 'a';
}
else if(vt.delaytotal > vt.masttotal || vt.pulsedtotal == 0)
{
vt.delaytotal = vt.masttotal;
```

```
locura = 1;
}
else
{
if (v.modoest == 'p' || v.modoest == 's')
{
if((vt.delaytotal + vt.pulsedtotal) >= vt.masttotal)
{
vt.pulsedtotal = vt.masttotal - vt.delaytotal;
extrat = 0;
locura = 0;
}
else
{
vt.tiempoextra = vt.masttotal - vt.delaytotal - vt.pulsedtotal;
extrat = 1;
locura = 0;
}
}
else if (v.modoest == 't' || v.modoest == 'u')
{
if(vt.traindtotal == 0 || vt.trainptotal == 0)
{
vt.delaytotal = vt.masttotal;
locura = 1;
}
else
{
if (vt.trainptotal > vt.traindtotal)
{
vt.trainptotal = vt.traindtotal;
}

if ((vt.delaytotal + vt.traindtotal) > vt.masttotal)
{
vt.traindtotal = vt.masttotal - vt.delaytotal;
if (vt.pulsedtotal >= vt.trainptotal)
{
vt.pulsedtotal = vt.traindtotal;
if (v.modoest == 't')
{
```

```
v.modoest = 'p';
}
else
{
v.modoest = 's';
}
sacarcuentas();
}
else
{
calcularpwm();
extrat = 0;
locura = 0;
}
}
else
{
vt.tiempoextra = vt.masttotal - vt.delaytotal - vt.traindtotal;
if (vt.pulsedtotal >= vt.trainptotal)
{
vt.pulsedtotal = vt.traindtotal;
if (v.modoest == 't')
{
v.modoest = 'p';
}
else
{
v.modoest = 's';
}
sacarcuentas();
}
else
{
calcularpwm();
extrat = 1;
locura = 0;
}
}
}
}
}
```

```
}


void calcularpwm()
{
rep = vt.traindtotal / vt.trainptotal;
vt.residuo = vt.traindtotal - (vt.trainptotal * rep);
vt.tlow = vt.trainptotal - vt.pulsedtotal;
if(vt.residuo == 0)
{
veceshigh = rep;
veceslow = rep;
cuentatotal = veceshigh + veceslow;
vt.trestante = vt.tlow;
}
if(vt.residuo > vt.pulsedtotal)
{
veceshigh = rep + 1;
veceslow = rep +1;
cuentatotal = veceshigh + veceslow;
vt.trestante = vt.residuo - vt.pulsedtotal;
}
else
{
veceshigh = rep + 1;
veceslow = rep;
cuentatotal = veceshigh + veceslow;
vt.trestante = vt.residuo;
}
}

void pulsito()
{
/*PORTB |= 1;
PORTB &= (~1);*/
PORTD |= B00010000;
PORTD &= ~(B00010000);
/*digitalWrite(sincroPin, HIGH);
digitalWrite(sincroPin, LOW);*/
}
```

```
void estadoestimular()
{
if (v.modorecep == 'e')
{
sbi(ADCSRA, ADEN);
sbi(ADCSRA, ADSC);
//datolisto = 1;
}
else
{
datolisto = 0;
cbi(ADCSRA, ADEN);
}
if (v.modoest == 's'|| v.modoest == 'u')
{
batisenal = 1;
finalrapido = 0;
}
else
{
batisenal = 0;
finalrapido = 0;
}

if (v.modoest == 'a')
{
tiempo = 0;
Timer1.detachInterrupt();
/*PORTB = B00;*/
PORTD &= ~(B11111100);

/*digitalWrite(outPin, LOW);*/
}
else if (locura == 1)
{
tiempo = 1;
/*PORTB = B01;
PORTB = B00;*/
PORTD &= ~(B11111100);
PORTD |= B00010000;
PORTD &= ~(B00010000);
```

```
/*digitalWrite(outPin, LOW);
pulsito();*/
}
else
{
if (v.modoest == 's' || v.modoest == 'p')
{
if (vt.delaytotal == 0)
{
tiempo = 2;
/*PORTB &= ~(B00111111)
PORTB |= B00010011;
PORTB &= ~(B00000001);*/
PORTD &= ~(B11111100);
PORTD |= B10010000;
PORTD &= ~(B00010000);
/*digitalWrite(outPin, HIGH);
pulsito();*/
}
else
{
tiempo = 1;
/*PORTB = B01;
PORTB = B00;*/
PORTD &= ~(B11111100);
PORTD |= B00010000;
PORTD &= ~(B00010000);
/*digitalWrite(outPin, LOW);
pulsito();*/
}
}
else if (v.modoest == 't' || v.modoest == 'u')
{
if (vt.delaytotal == 0)
{
tiempo = 2;
/*PORTB = B11;
PORTB = B10;*/
PORTD &= ~(B11111100);
PORTD |= B10010000;
PORTD &= ~(B00010000);
```

```
/*digitalWrite(outPin, HIGH);
pulsito();*/
counter = cuentatotal;
parte = 1;
}
else
{
tiempo = 1;
/*PORTB = B01;
PORTB = B00;*/
PORTD &= ~(B11111100);
PORTD |= B00010000;
PORTD &= ~(B00010000);
/*digitalWrite(outPin, LOW);
pulsito();*/
}
}
}
}
```

ConfigADC.cpp

```
#include "Variables.h"

void ADCinicio(void)
{
//---------------------------------------------------------------------
// ADMUX settings
//---------------------------------------------------------------------
// These bits select the voltage reference for the ADC. If these bits
// are changed during a conversion, the change will not go in effect
// until this conversion is complete (ADIF in ADCSRA is set). The
// internal voltage reference options may not be used if an external
// reference voltage is being applied to the AREF pin.
// REFS1 REFS0 Voltage reference
// 0 0 AREF, Internal Vref turned off
// 0 1 AVCC with external capacitor at AREF pin
// 1 0 Reserved
// 1 1 Internal 1.1V Voltage Reference with external
// capacitor at AREF pin
cbi(ADMUX,REFS1);
```

```
sbi(ADMUX,REFS0);
// The ADLAR bit affects the presentation of the ADC conversion result
// in the ADC Data Register. Write one to ADLAR to left adjust the
// result. Otherwise, the result is right adjusted. Changing the ADLAR
// bit will affect the ADC Data Register immediately, regardless of any
// ongoing conversions.
cbi(ADMUX,ADLAR);
// The value of these bits selects which analog inputs are connected to
// the ADC. If these bits are changed during a conversion, the change
// will not go in effect until this conversion is complete (ADIF in
// ADCSRA is set).
ADMUX |= 0;

//------------------------------------------------------------------
// ADCSRA settings
//------------------------------------------------------------------
// Writing this bit to one enables the ADC. By writing it to zero, the
// ADC is turned off. Turning the ADC off while a conversion is in
// progress, will terminate this conversion.
cbi(ADCSRA,ADEN);
// In Single Conversion mode, write this bit to one to start each
// conversion. In Free Running mode, write this bit to one to start the
// first conversion. The first conversion after ADSC has been written
// after the ADC has been enabled, or if ADSC is written at the same
// time as the ADC is enabled, will take 25 ADC clock cycles instead of
// the normal 13. This first conversion performs initialization of the
// ADC. ADSC will read as one as long as a conversion is in progress.
// When the conversion is complete, it returns to zero. Writing zero to
// this bit has no effect.
cbi(ADCSRA,ADSC);
// When this bit is written to one, Auto Triggering of the ADC is
// enabled. The ADC will start a conversion on a positive edge of the
// selected trigger signal. The trigger source is selected by setting
// the ADC Trigger Select bits, ADTS in ADCSRB.
sbi(ADCSRA,ADATE);
// When this bit is written to one and the I-bit in SREG is set, the
// ADC Conversion Complete Interrupt is activated.
sbi(ADCSRA,ADIE);
// These bits determine the division factor between the system clock
// frequency and the input clock to the ADC.
// ADPS2 ADPS1 ADPS0 Division Factor
```

```
// 0 0 0 2
// 0 0 1 2
// 0 1 0 4
// 0 1 1 8
// 1 0 0 16
// 1 0 1 32
// 1 1 0 64
// 1 1 1 128
sbi(ADCSRA,ADPS2);
sbi(ADCSRA,ADPS1);
cbi(ADCSRA,ADPS0);

//-----------------------------------------------------------------------
// ADCSRB settings
//-----------------------------------------------------------------------
// When this bit is written logic one and the ADC is switched off
// (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative
// input to the Analog Comparator. When this bit is written logic zero,
// AIN1 is applied to the negative input of the Analog Comparator.
cbi(ADCSRB,ACME);
// If ADATE in ADCSRA is written to one, the value of these bits
// selects which source will trigger an ADC conversion. If ADATE is
// cleared, the ADTS2:0 settings will have no effect. A conversion will
// be triggered by the rising edge of the selected Interrupt Flag. Note
// that switching from a trigger source that is cleared to a trigger
// source that is set, will generate a positive edge on the trigger
// signal. If ADEN in ADCSRA is set, this will start a conversion.
// Switching to Free Running mode (ADTS[2:0]=0) will not cause a
// trigger event, even if the ADC Interrupt Flag is set.
// ADTS2 ADTS1 ADTS0 Trigger source
// 0 0 0 Free Running mode
// 0 0 1 Analog Comparator
// 0 1 0 External Interrupt Request 0
// 0 1 1 Timer/Counter0 Compare Match A
// 1 0 0 Timer/Counter0 Overflow
// 1 0 1 Timer/Counter1 Compare Match B
// 1 1 0 Timer/Counter1 Overflow
// 1 1 1 Timer/Counter1 Capture Event
cbi(ADCSRB,ADTS2);
cbi(ADCSRB,ADTS1);
cbi(ADCSRB,ADTS0);
```

```
//-----------------------------------------------------------------------
// DIDR0 settings
//-----------------------------------------------------------------------
// When this bit is written logic one, the digital input buffer on the
// corresponding ADC pin is disabled. The corresponding PIN Register
// bit will always read as zero when this bit is set. When an analog
// signal is applied to the ADC5..0 pin and the digital input from this
// pin is not needed, this bit should be written logic one to reduce
// power consumption in the digital input buffer.
// Note that ADC pins ADC7 and ADC6 do not have digital input buffers,
// and therefore do not require Digital Input Disable bits.
sbi(DIDR0,ADC5D);
sbi(DIDR0,ADC4D);
sbi(DIDR0,ADC3D);
sbi(DIDR0,ADC2D);
sbi(DIDR0,ADC1D);
sbi(DIDR0,ADC0D);

}
```

ISR.cpp

```
#include "Variables.h"

ISR(ADC_vect)
{
/*datoL =ADCL;
datoH = ADCH;
datolisto = 1;*/

/*if(crescendo ==1)
{
bufferficticio[contadorcillo1] = contadorcillo2;
contadorcillo1++;
contadorcillo2++;
bufferficticio[contadorcillo1] = contadorcillo3;
contadorcillo1++;
if(contadorcillo2 == 255)
{
contadorcillo2 = 0;
contadorcillo3++;
```

```
if (contadorcillo3 == 4)
{
crescendo = 0;
contadorcillo3 = 0;
}
}
}
else
{
bufferficticio[contadorcillo1] = (254 - contadorcillo2);
contadorcillo1++;
contadorcillo2++;
bufferficticio[contadorcillo1] = (3 - contadorcillo3);
contadorcillo1++;
if (contadorcillo2 == 255)
{
contadorcillo2 = 0;
contadorcillo3++;
if (contadorcillo3 == 4)
{
crescendo = 1;
contadorcillo3 = 0;
}
}
}
if (contadorcillo1 == (CAPACIDAD_BLOQUE *(numerobloque + 1)))
{
datolisto = 1;
}
if (contadorcillo1 >= (CAPACIDAD_BUFFER - 1))
{
contadorcillo1 = 0;
}*/


bufferADC[counterADC]= ADCL;
++counterADC;
bufferADC[counterADC] = ADCH;
++counterADC;
if (counterADC == (CAPACIDAD_BLOQUE *(numerobloque + 1)))
{
```

```
datolisto = 1;
}
if (counterADC >= (CAPACIDAD_BUFFER - 1))
{
counterADC = 0;
}


}

void serialEvent()
{
if(Serial.available() == 17)
{
v.modoest = Serial.read();
v.modorecep = Serial.read();
v.mastnumero[0] = Serial.read();
v.mastnumero[1] = Serial.read();
v.multmastp = Serial.read();
v.delaynumero[0] = Serial.read();
v.delaynumero[1] = Serial.read();
v.multdelay = Serial.read();
v.traindnumero[0] = Serial.read();
v.traindnumero[1] = Serial.read();
v.multtraind = Serial.read();
v.trainpnumero[0] = Serial.read();
v.trainpnumero[1] = Serial.read();
v.multtrainp = Serial.read();
v.pulsednumero[0] = Serial.read();
v.pulsednumero[1] = Serial.read();
v.multpulsed = Serial.read();

vt.masttotal = procesameste(v.mastnumero , v.multmastp);
vt.delaytotal = procesameste(v.delaynumero, v.multdelay);
vt.traindtotal = procesameste(v.traindnumero, v.multtraind);
vt.trainptotal = procesameste (v.trainpnumero, v.multtrainp);
vt.pulsedtotal = procesameste (v.pulsednumero, v.multpulsed);

sacarcuentas();
estadoestimular();
```

```
/*char hola[50];
ultoa(vt.masttotal, hola, 10);
Serial.write(hola);*/
}

}
```

InterrupTime.cpp

```
#include "Variables.h"

void tdelay()
{
if(locura == 1)
{
if (finalrapido == 2 && batisenal == 1)
{
tiempo = 0;
}
else
{
tiempo = 1;
finalrapido ++;
pulsito();
}
}
else
{
/*PORTB = B10;*/
PORTD |= B10000000;
/*digitalWrite(outPin, HIGH);*/
tiempo = 2;
if (v.modoest == 't' || v.modoest == 'u')
{
counter = cuentatotal;
parte = 1;
}
}
}

void durap()
```

```
{
if(vt.delaytotal == 0 && extrat == 0)
{
if(finalrapido == 2 && batisenal == 1)
{
/*PORTB = B00;*/
PORTD &= ~(B11111100);
/*digitalWrite(outPin, LOW);*/
tiempo = 0;
}
else
{
finalrapido ++;
tiempo = 2;
pulsito();
}
}
else
{
/*digitalWrite(outPin, LOW);*/
if (extrat == 0)
{
if (finalrapido == 2 && batisenal == 1)
{
/*PORTB = B00;*/
PORTD &= ~(B11111100);
tiempo = 0;
}
else
{
PORTD ^= B10001000;
PORTD &= ~(B00001000);
/*PORTB = B01;
PORTB = B00;*/
finalrapido ++;
tiempo = 1;
/*pulsito();*/
}
}
else
{
```

```
PORTD &= ~(B11111100);
/*PORTB = B00;*/
tiempo = 3;
}
}
}

void durat1()
{
PORTD &= ~(B11111100);
/*PORTB = B00;*/
/*digitalWrite(outPin, LOW);*/
tiempo = 2;
parte = 2;
counter = counter -1;
if (counter == 1)
{
final = 1;
}
}

void durat2()
{
PORTD |= B10000000;
/*PORTB = B10;*/
/*digitalWrite(outPin, HIGH);*/
tiempo = 2;
parte = 1;
counter = counter -1;
if (counter == 1)
{
final = 1;
}
}

void duratfinal()
{
if (finalrapido == 2 && batisenal == 1)
{
PORTD &= ~(B11111100);
/*PORTB = B00;*/
```

```
/*digitalWrite(outPin, LOW);*/
tiempo = 0;
}
else if(vt.delaytotal == 0 && extrat == 0)
{
PORTD |= B10001000;
PORTD &= ~(B00001000);
/*PORTB = B11;
PORTB = B10;*/
/*digitalWrite(outPin, HIGH);
pulsito();*/
tiempo = 2;
finalrapido ++;
counter = cuentatotal;
parte = 1;
final = 0;
}
else if (extrat == 0)
{
if ((PORTD >> 7) == B1)
{
char jj[] = "1";
Serial.print(jj);
PORTD ^= B10001000;
}
else
{
char jj[] = "2";
Serial.print(jj);
PORTD |= B00001000;
}
PORTD &= ~(B11111100);
/*PORTB = B01;
PORTB = B00;*/
/*digitalWrite(outPin, LOW);
pulsito();*/
finalrapido ++;
tiempo = 1;
final = 0;
}
else
```

```
{
PORTD &= ~(B11111100);
/*PORTB = B00;*/
/*digitalWrite(outPin, LOW);*/
tiempo = 3;
final = 0;
}
}

void textra()
{
if (finalrapido == 2 && batisenal == 1)
{
tiempo = 0;
}
else if(vt.delaytotal == 0)
{
finalrapido ++;
PORTD |= B10001000;
PORTD &= ~(B00001000);
/*PORTB = B11;
PORTB = B10;*/
/*digitalWrite(outPin, HIGH);
pulsito();*/
tiempo = 2;
if (v.modoest == 't')
{
counter = cuentatotal;
parte = 1;
}
}
else
{
finalrapido ++;
tiempo = 1;
pulsito();
}
}
```

Variables.h

```
#ifndef VARIABLES
#define VARIABLES
//-------------------------------------------------------------------------
//Includes
//-------------------------------------------------------------------------
#include <Arduino.h>

//-------------------------------------------------------------------------
//Defines and Typedefs
//-------------------------------------------------------------------------

#define CAPACIDAD_BUFFER 20
#define CAPACIDAD_BLOQUE 2
#define NUMERO_BLOQUES 10

// Defines for setting and clearing register bits
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

//-------------------------------------------------------------------------
//Prototipo de Funciones
//-------------------------------------------------------------------------

void ADCinicio(void);
void tdelay();
void pulsito();
void durap();
void durat1();
void durat2();
void duratfinal();
void textra();
unsigned long procesameste(char a[2], char b);
void sacarcuentas();
void calcularpwm();
void estadoestimular();

//-------------------------------------------------------------------------
```

```
//Declaracion de Variables Globales
//--------------------------------------------------------------------------------

extern volatile unsigned int datoL;
extern volatile unsigned int datoH;
extern volatile int datolisto;
extern volatile int locura;
extern volatile int finalrapido;
extern volatile int batisenal;
extern volatile int tiempo;
extern volatile int counter;
extern volatile int parte;
extern volatile int cuentatotal;
extern volatile int extrat;
extern volatile int final;
extern volatile int veceslow;
extern volatile int veceshigh;
extern volatile int rep;
extern volatile int duty;
extern volatile int counterADC;
extern volatile uint8_t bufferADC[CAPACIDAD_BUFFER];
extern volatile int numerobloque;
extern volatile uint8_t bufferficticio[CAPACIDAD_BUFFER];
extern volatile int contadorcillo1;
extern volatile int contadorcillo2;
extern volatile int contadorcillo3;
extern volatile int crescendo;


struct Variables
{
char modoest;
char modorecep;
char mastnumero[2];
char delaynumero[2];
char traindnumero[2];
char trainpnumero[2];
char pulsednumero[2];
char multmastp;
char multdelay;
char multtraind;
```

```
char multtrainp;
char multpulsed;
};

struct Variablestotales
{
unsigned long masttotal;
unsigned long delaytotal;
unsigned long traindtotal;
unsigned long trainptotal;
unsigned long pulsedtotal;
unsigned long tiempoextra;
unsigned long residuo;
unsigned long tlow;
unsigned long trestante;
};

extern Variables v;
extern Variablestotales vt;

#endif
```

Bibliografía

[1] Ridley Scott. (Director). The martian. 20th Century Fox, 2015.

[2] Cardinali Dvorkin, Mario A Dvorkin, Daniel P Cardinali, et al. *Best &Taylor. Bases Fisiológicas de la Práctica Médica*. Ed. Médica Panamericana, 12a. edición edition, 1998.

[3] Veganimal. Veganimal - tu sitio de biología! - transmisión del impulso nervioso. `https://veganimal.wikispaces.com/Transmisi%C3%B3n+del+impulso+nervioso`, 2015. [Online; accessed 11-October-2015].

[4] Apuntes Neuroanatomía-UFRO. Tono muscular. `http://www.med.ufro.cl/Recursos/neuroanatomia/archivos/13_sistematizacion_archivos/Page387.htm`, 2015. [Online; accessed 11-October-2015].

[5] David J. Randall, Warren W Burggren, Kathleen French, and Roger Eckert. *Eckert animal physiology mechanisms and adaptations*. New York W.H. Freeman and Co, ed. 4 edition, 2002. Includes bibliographical references and index.

[6] Temas de estudio para bioquímica y fisiología. Fisiología del proceso contráctil de músculo esquelético - temas de estudio para bioquímica y fisiología. `http://www.bioquimicayfisiologia.com/2014/03/fisiologia-del-proceso-contractil-del-musculo-esqueletico.html`, 2015. [Online; accessed 11-October-2015].

[7] Só Biologia. contração muscular - só biologia. `http://www.sobiologia.com.br/conteudos/FisiologiaAnimal/sustentacao7.php`, 2015. [Online; accessed 11-October-2015].

[8] Wikipedia. Wheatstone bridge — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Wheatstone_bridge`, 2015. [Online; accessed 07-October-2015].

[9] Wikipedia. Qt (software) — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Qt_(software)`, 2015. [Online; accessed 07-October-2015].

[10] The Qt Company. Qt - home. http://www.qt.io/, 2015. [Online; accessed 07-October-2015].

[11] Imprimalia 3D. Se desata la guerra con una escisión dentro de arduino - impresoras 3d -imprimalia 3d. http://www.imprimalia3d.com/noticias/2015/02/11/004289/se-desata-guerra-una-escisi-n-dentro-arduino, 2015. [Online; accessed 07-October-2015].

[12] Atmel. 8-bit microcontroller with 4/8/16/32k bytes in-system programmable flash. http://www.atmel.com/Images/doc8161.pdf, 2015.

[13] Paul Stoffregen. Paul stoffregen/timerone - github. https://github.com/PaulStoffregen/TimerOne/, 2015. [Online; accessed 10-October-2015].

[14] LA Geddes and LE Baker. The specific resistance of biological material—a compendium of data for the biomedical engineer and physiologist. *Medical and biological engineering*, 5(3):271–293, 1967.

[15] Otto Schanne, Hiroshi Kawata, Bärbel Schäfer, and Marc Lavallée. A study on the electrical resistance of the frog sartorius muscle. *The Journal of general physiology*, 49(5):897–912, 1966.

[16] Caffeinomane. Girino - fast arduino oscilloscope - all. http://www.instructables.com/id/Girino-Fast-Arduino-Oscilloscope/?ALLSTEPS, 2015. [Online; accessed 11-October-2015].

[17] Emanuel Eichhammer. Qt plotting widget qcustomplot - introduction. http://www.qcustomplot.com/index.php/introduction, 2015. [Online; accessed 11-October-2015].

[18] Freddy Alferink. Arduino analog measurements::electronic measurements. http://meettechniek.info/embedded/arduino-analog.html, 2015. [Online; accessed 10-October-2015].

[19] Smilen Dimitrov. Arduino playground - corruptarrayvariablesandmemory. http://playground.arduino.cc/Main/CorruptArrayVariablesAndMemory#Max_size_of_array, 2015. [Online; accessed 11-October-2015].

[20] Bill Porter. Ready, set, oscillate! the fastest way to change arduino pins - the mind of bill porter. http://www.billporter.info/2010/08/18/ready-set-oscillate-the-fastest-way-to-change-arduino-pins/, 2015. [Online; accessed 11-October-2015].