ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

“DISEÑO DE UN SISTEMA DE CONTROL DE MOVIMIENTOS DE AGARRE PARA UNA PRÓTESIS DE MANO ROBÓTICA CON BRAZALETE MYO”

INFORME DE MATERIA INTEGRADORA

Previo a la obtención del Título de:

INGENIERA EN ELECTRICIDAD ESPECIALIZACIÓN ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL

RUTTY ALEXANDRA CEDEÑO ARANA

GUAYAQUIL – ECUADOR

AÑO: 2017

AGRADECIMIENTOS

Mis más sinceros agradecimientos a la MSc. Lisbeth Mena quien propuso y financió en gran medida el desarrollo del presente proyecto.

DEDICATORIA

El presente proyecto lo dedico de manera especial a mi padre quien fue el principal cimiento e inspiración para la construcción de mi vida profesional, pues me atrevo a confesar que gracias a su ejemplo, el apoyo y las enseñanzas que me ha brindado a lo largo de mi vida, me he convertido en la persona que soy hoy en día.

A mi madre y a mis dos hermanos que son las personas que me han ofrecido amor y calidez de la familia a la cual amo.

TRIBUNAL DE EVALUACIÓN

MSc. Ricardo Cajo	**MSc. Lisbeth Mena**
PROFESOR EVALUADOR	PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me corresponde exclusivamente; y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

.....

Rutty Cedeño Arana

RESUMEN

El proyecto surgió con el fin de desarrollar un sistema de control de movimientos básicos de agarre para una prótesis de mano de bajo costo, para permitirle realizar dichos movimientos a un joven quien no posee dedos en su mano derecha debido a una malformación congénita, considerando que los movimientos de su muñeca son completos. Bajo estas condiciones se inicia el desarrollo del primer prototipo de prótesis de mano haciendo uso del brazalete Myo, el cual posee ocho sensores mioeléctricos no invasivos que se posicionan alrededor del antebrazo a manera de brazalete, además Myo posee un giroscopio y acelerómetro los cuales permite la obtener datos de los tres diferentes ejes correspondiente (x, y, z), y a su vez envía datos de posición. Esto abre un sinnúmero de posibilidades a desarrolladores de todas partes del mundo.

En el presente trabajo se optó por diseñar un control de movimientos para una prótesis de mano, el cual no exige mayor esfuerzo muscular del usuario, usándolo solo para indicar el inicio y fin de lectura de gestos, en otras palabras movimientos simples del antebrazo son traducidos en movimientos de dedo en el prototipo de prótesis de mano.

.

ÍNDICE GENERAL

CAPÍTULO 1

1. DELIMITACIÓN DEL PROBLEMA

1.1 Planteamiento del problema.

Existen diversas causas por las que una persona puede poseer una discapacidad física, que van desde malformaciones congénitas hasta aquellas que son causadas por enfermedades degenerativas o accidentes, lo que por ende hace que todos estemos expuestos a la posibilidad de adquirir una discapacidad física.

En ciertos casos es posible que las personas que poseen esta condición tengan que enfrentarse a barreras sociales, además de que se presentan problemas para desarrollar actividades cotidianas que una persona sin discapacidad física realiza con toda normalidad, sin mencionar que puede llegar a ser factor limitante a la hora de conseguir un trabajo.

En la actualidad con los avances tecnológicos que se han venido llevando a cabo ya existen soluciones para determinado tipo de discapacidades físicas, como lo son las prótesis robóticas, que permiten a las personas poder ejercer control sobre el movimiento de partes del cuerpo que antes no poseían.

A pesar de ser un área que se encuentra en constante mejora y desarrollo, la principal problemática que se presenta a la hora de adquirir una prótesis robótica, son los costos inaccesibles para un considerable porcentaje de la población, no solo del país, sino del mundo, por lo que dicho porcentaje se encuentra lejos de satisfacer las necesidades que una persona con discapacidad física posee.

Lo ideal sería buscar formas en las que una prótesis de mano robótica este dentro de los estándares de calidad y funcionalidad básicas necesarias, pero a su vez a costos considerablemente menores a los comúnmente encontrados en el mercado, a manera de que sea accesible para personas con diversas condiciones socio económicas de la población, logrando así satisfacer sus necesidades.

1.2 Objetivo general

- Controlar movimientos de agarre para una prótesis robótica de mano, mediante el uso del brazalete Myo y microcontrolador Arduino Nano.

1.3 Objetivos específicos.

- Establecer interfaz de comunicación directa entre el brazalete Myo y el microcontrolador Arduino Nano.
- Interpretar las señales adquiridas del brazalete Myo para la generación de algoritmos de control.
- Generar patrones de movimiento en el prototipo de prótesis de mano, dependientes de las señales adquiridas del brazalete Myo, enfocados en un portador con movimientos de muñeca completos.

1.4 Justificación

El proyecto surgió como el desarrollo de una prótesis de bajo costo, para permitir realizar movimientos básicos de agarre, bajo el caso especifico de una persona que posee una malformación de nacimiento en una de sus manos, tomando en cuenta que la condición de dicha persona se diagnostica en agenesia metacarpiana que hace referencia a la ausencia parcial o total de un órgano o de un tejido del organismo, en este caso ausencia de dedos, sin embargo los movimientos de la muñeca son completos.

Debido a que existen movimientos de muñeca y presencia de actividad muscular en el antebrazo se torno condición primordial en el desarrollo de este proyecto, se propuso hacer uso de un brazalete comercial llamado Myo, el cual brinda libre acceso a los datos que obtiene tanto de sus sensores mioeléctricos como de su unidad de medición inercial.

El proyecto demanda el desarrollo de algoritmos de control, mediante programación en la plataforma de Arduino, que permitan la adquisición y almacenamiento de los datos censados por el brazalete para después traducirlos en movimientos de dedos de una prótesis impresa, haciendo control de cinco tendones por medio de un sistema de servomotores.

Una vez se logre la comunicación con el brazalete Myo, independiente de un computador, los proyectos que se pueden llevar a cabo se vuelven infinitos, desde el control de una prótesis en base a patrones de movimientos, hasta el desarrollo de un algoritmo de control que responda a las señales adquiridas en respuesta a la actividad muscular de una persona en específico. Permitiendo así la elaboración de prótesis que cumplan con satisfacer necesidades específicas de cada persona que lo necesite, yendo más allá de mal formaciones congénitas y poder elaborar prótesis como sustitución competa de una mano o un brazo.

1.5 Alcance del proyecto

Se realiza el análisis y estudio de funcionamiento del brazalete Myo, para tener en consideración información relevante para el proyecto, como el tiempo de carga y el tiempo de autonomía del brazalete, todo esto con el fin de asegurar confiabilidad tanto en comunicación como en el funcionamiento conjunto con el sistema de control de movimiento para el prototipo de prótesis de mano.

Teniendo en cuenta que si bien la comunicación entre el brazalete Myo y la tarjeta controladora Arduino Nano es posible usando como medio una computadora, para efectos del proyecto esto no es práctico, por lo que se procederá a establecer una interfaz de comunicación directa entre el brazalete Myo y la tarjeta controladora Arduino Nano mediante el uso de un bluetooth 4.0 de bajo consumo de energía, de modo que este sea capaz de receptar todas las señales captadas por el brazalete Myo y transmitirlas directamente al microcontrolador.

Ya que en la actualidad son solo 5 los movimientos o poses preestablecidas que es capaz de reconocer el brazalete Myo, se realizará las investigaciones necesarias para poder realizar una correcta interpretación de las señales captadas por el brazalete Myo y poder hacer uso de las mismas para el reconocimiento de nuevos gestos. Los datos a los que se tendrá acceso son:

- Giroscopio (x, y, z).
- Acelerómetro (x, y, z).

- Orientación (x, y, z, w).
- Sensores mioeléctricos (8).

Se analizan los datos obtenidos luego de realizar diversas pruebas de movimiento con el brazalete Myo, aislando los datos censados provenientes de movimientos de dedos, y en su lugar hacer énfasis en los datos censados como consecuencia de movimientos completos de muñeca y antebrazo.

Luego del análisis y de depuración de los datos obtenidos en las diversas pruebas, se procede a establecer patrones de movimientos dirigidos a personas que presentan ausencia de dedos, para que estos puedan ser traducidos a movimientos específicos en el prototipo de prótesis de mano. Para ello se generará un algoritmo de control, sobre la plataforma de Arduino que utilice los datos obtenidos y permita el manejo de un sistema que servo motores que serán los encargados de dar movilidad a la prótesis**.**

CAPÍTULO 2

2. ESTADO DEL ARTE

2.1 Antecedentes

A mediados del año 2010, estudiantes de la Universidad de Bogotá, Colombia, utilizaron el microcontrolador de Microchip dspic30f6014A y el entorno de desarrollo LabView para crear un prototipo de prótesis de mano constituida por tres dedos y la palma de la mano, que permitía realizar cinco diferentes tipos de agarres. Estos agarres eran posibles gracias a que cada dedo poseía tres articulaciones conformadas por micromotores. Los resultados en aquel entonces dieron ideas para que a futuro se logre establecer el movimiento de los dedos en base a la actividad muscular del antebrazo. [1]

A finales del año 2011, Richard Van As, luego de perder sus dedos mientras trabajaba, se pone en contacto con Ivan Owen quien ya había desarrollado prótesis mecánicas en conjunto con la empresa Makerbot logran desarrollar una prótesis de mano totalmente mecánica a partir de impresiones en 3D. Esta prótesis basada en el ángulo de giro de la muñeca permite realizar el agarre de objetos de una manera muy cómoda. Lo mejor de todo es que el diseño fue subido a la red para que todas las personas que posean una impresora 3D en sus hogares puedan fabricarlo. [2]

En el 2012 Thalmic Labs desarrolla, el brazalete Myo el cual da posibilidades infinitas a desarrolladores, dándoles acceso a los SDK y datos en bruto para dar paso a la adaptación de inimaginables usos del brazalete. En 2005, Johnny Matheny fue diagnosticado con cáncer en el antebrazo izquierdo y en 2008, los médicos tuvieron que amputarle el brazo más allá de su codo. En el 2015 se dio a conocer que en el Laboratorio de Física Aplicada Johns Hopkins, se estuvo llevando a cabo el proyecto de fabricar una prótesis de brazo unida directamente al esqueleto de Johnny, esta para su movilidad utiliza dos brazaletes Myo posicionadas en la parte superior del brazo para detectar la actividad muscular y movimientos. [3]

2.2 Marco teórico

2.2.1 Flexy Hand2

Esta prótesis de mano originalmente mecánica, es la tentativa a usar para realizar las primeras pruebas del sistema de control de movimientos de agarre del proyecto, todos los archivos correspondientes se los puede descargar de forma gratuita, lista para ajustar medidas e imprimirla con ayuda de una impresora 3D. [4] La prótesis impresa se puede apreciar en la figura 2.1

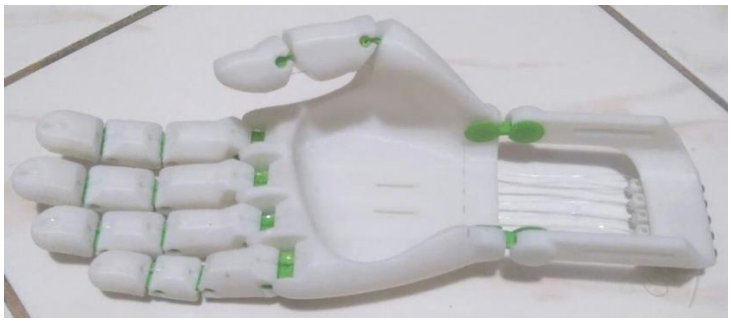

Figura 2.1: Prótesis impresa Flexy Hand2.

La prótesis está diseñada para personas que no poseen dedos, a modo de que con movimientos de muñeca hacia arriba o hacia abajo, los dedos de la prótesis realicen un movimiento conjunto de extensión o flexión respectivamente. Esto es posible gracias a la simulación de tendones para lo cual se utiliza un cable que une cada representación de falanges de la prótesis, este debe pasar dos veces por dedo, en cada punta de los dedos existe dos orificios por donde entrara el cable, pero este saldrá por la muñeca, por un mismo orificio. Lo orificios por donde pasa el cable son internos, a manera de que este no permanezca a la vista.

2.2.2 Brazalete Myo

El brazalete Myo es un dispositivo portátil capaz de conectarse a otros diversos dispositivos a través de su propio adaptador bluetooth USB. El brazalete fue diseñado con el fin de, a partir de movimientos y gestos

con la mano, permitir el control en teléfonos, computadoras, juegos, entre otros.

Posee sensores mioeléctricos de acero inoxidable de grado médico, IMU de nueve ejes de alta sensibilidad que contiene giroscopio de tres ejes, acelerómetro de tres ejes y magnetómetro de tres ejes. Brinda facilidad de obtención de datos digitalizados mediante la conexión a un ordenador Windows o Mac, a través de su adaptador Bluetooth USB incluido, además de dispositivo iOS o Android con una conexión Bluetooth 4.0 de bajo consumo de energía. [5]

En cuanto a sus dimensiones, es un brazalete ampliable de 19 a 34 cm, con un peso de 93 gr. Y un espesor de aproximadamente 1,14 cm. Lo que permite que sea un dispositivo portátil, fácil de llevar sin ejercer esfuerzo alguno. El brazalete Myo y sus componentes externos se muestran en la figura 2.2.

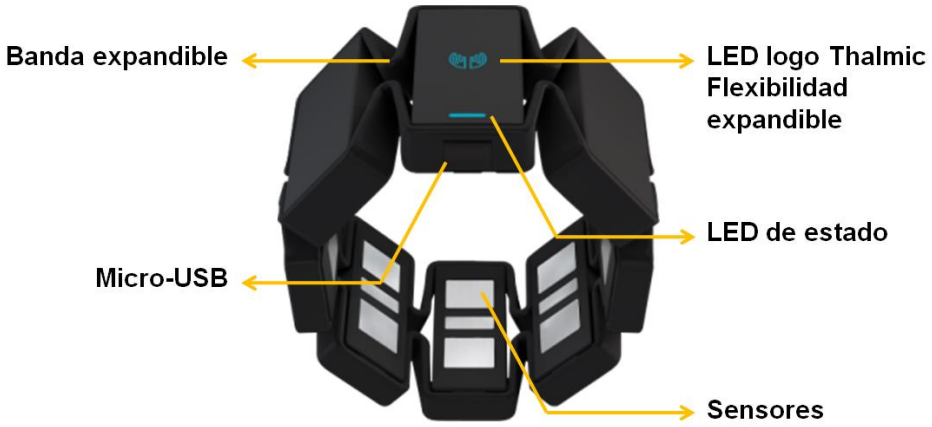

Figura 2.2: Componentes externos del brazalete Myo. [6]

Myo posee dos indicadores LED que dan información del estado de la Batería del brazalete como se muestra en la tabla 1. [7]

LED LOGO THALMIC	LED ESTADO	ESTADO DE BATERÍA
Apagado	**Apagado**	Batería descargada
Constante azul	**Destello naranja**	Batería baja

Apagado	**Constante verde**	Se ha completado la carga
Apagado	**Pulso naranja**	Cargando

Por medio de los indicadores LED también se puede conocer el estado de conexión de brazalete como se muestra en la tabla 2.

LED LOGO THALMIC	LED ESTADO	ESTADO DE CONEXIÓN
Constante azul	**Constante azul**	Bluetooth conectado y se ha relazado sincronización
Constante azul	**Pulso azul**	Bluetooth desconectado y se ha relazado sincronización
Pulso azul	**Apagado**	Bluetooth desconectado y sin sincronizar (Modo reposo)
Apagado	**Apagado**	Modo suspensión (Mover MYO para activarlo)
Destello azul	**Contante azul**	Modo calentando (MYO vibrará al terminar)

Tabla 2: Estado de conexión del brazalete Myo

2.2.3 Módulo Bluetooth HM-11

Puesto que la comunicación y obtención de datos del brazalete Myo se dan mediante la conexión a un ordenador, a través de su adaptador Bluetooth USB, quita la posibilidad de desarrollar un proyecto portátil, es por esto que se tiene la necesidad de cambiar la ruta de transmisión de datos, para realizar una conexión directa con otras plataformas como una placa Arduino. [8] Esto se puede realizar con la ayuda un módulo Bluetooth de bajo consumo de energía, como se muestra en la figura: 2.3.

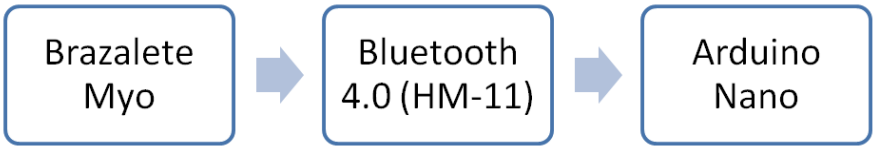

Figura 2.3: Conexión entre brazalete Myo y Arduino.

Es un módulo Bluetooth 4.0 de bajo consumo de energía que consta con un chip integrado CC2541. Es pequeño y fácil de usar, con el firmware pre programado del fabricante. [9] La idea es cargar sobre este módulo el firmware que nos permita una comunicación directa con el brazalete MYO para tener a disposición los datos de los sensores mioeléctricos, así como de su giroscopio y acelerómetro. La disposición de pines del módulo HM-11 se muestra en la figura 2.4, y se detallan en la tabla 3.

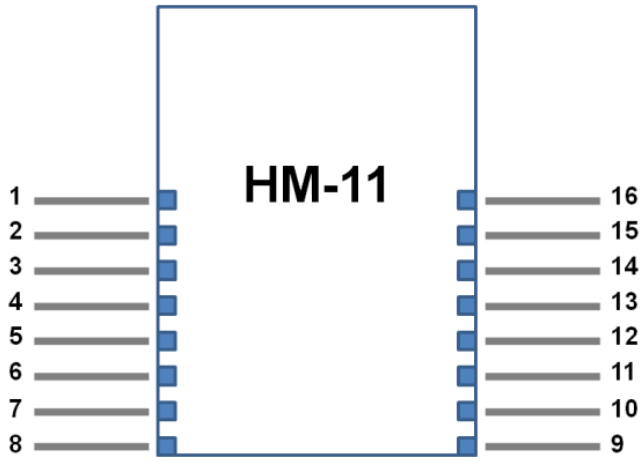

Figura 2.4: Disposición de pines Módulo HM-11.

NÚMERO	NOMBRE	NÚMERO	NOMBRE
1	UART_RTS	**9**	3.3 V
2	UART_TX	**10**	NC
3	UART_CTS	**11**	RESETB
4	UART_RX	**12**	GND

5	NC	**13**	PIO3
6	NC	**14**	PIO2
7	NV	**15**	PIO1 / LED
8	NV	**16**	PIO0 / KEY

Tabla 3: Nombre de pines Módulo HM-11

2.2.4 Placa Arduino Nano

Arduino es una plataforma electrónica, de código abierto que consta tanto de software como hardware. La razón por la que se utiliza la placa Arduino Nano, es debido a su conveniente tamaño y versatilidad. La placa posee un puerto tipo mini B-USB y cuenta con 8 entradas analógicas y 14 puertos digitales de entrada/salida, además trabaja con un voltaje de operación a nivel lógico de 5 V y un rango de voltaje de alimentación de 7 a 12 V teniendo un consumo de corriente DC por pin de entrada/salida de 40 mA. En cuanto a su disposición de pines, estos se muestran en la imagen 2.5 y se detallan en la tabla 2.4.

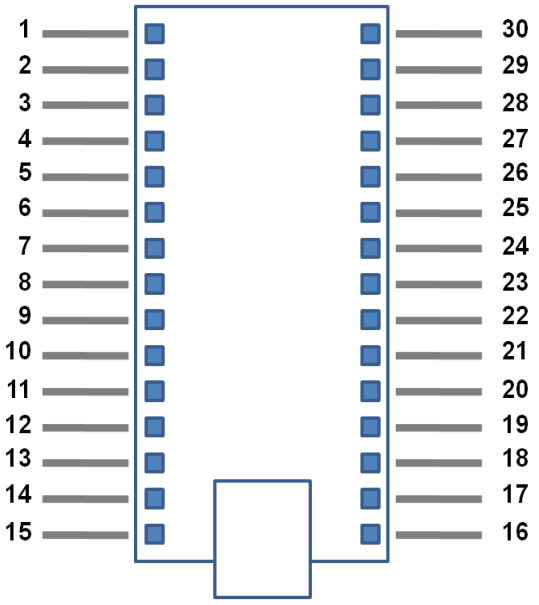

Figura 2.5: Disposición de pines Arduino NANO.

NÚMERO	NOMBRE	NÚMERO	NOMBRE
1	D1 / TXD	16	D13
2	D0 / RXD	17	3.3 V
3	RESET	18	AREF
4	GND	19	ADC0
5	D2	20	ADC1
6	D3 / PWM	21	ADC2
7	D4	22	ADC3
8	D5 / PWM	23	ADC4 / SDA
9	D6 / PWM	24	ADC5 / SCL
10	D7	25	ADC6
11	D8	26	ADC7
12	D9 / PWM	27	5 V
13	D10 / PWM / SS	28	RESET
14	D11 / PWM / MOSI	29	GND
15	D12 / MISO	30	Vin

Tabla 4: Nombre de pines Arduino nano.

2.2.5 Actuadores

De manera que se busca darle variedad a los movimientos de agarre de la prótesis se necesita hacer un control individual sobre el movimiento de flexión y extensión de cada dedo, esto se puede llevar a cabo acoplando cada tendón de la prótesis a los ejes de un servo motor de engranajes metálicos, como lo es el modelo MG90S.

2.2.6 Fuente de alimentación

En cuanto al tema de suministro de energía del proyecto, aún se encuentra en estado de prueba, pero la opción más destacada son un conjunto de cuatro baterías de LiPo recargables con un voltaje nominal de 3.7V cada una y capaz de entregar 1200mA.Las baterías se muestran en la figura 2.6. [10]

Las baterías son ligeras con un peso aproximado de 20g y con las siguientes dimensiones:

- Altura: 48mm
- Ancho: 30mm
- Grosor: 7mm

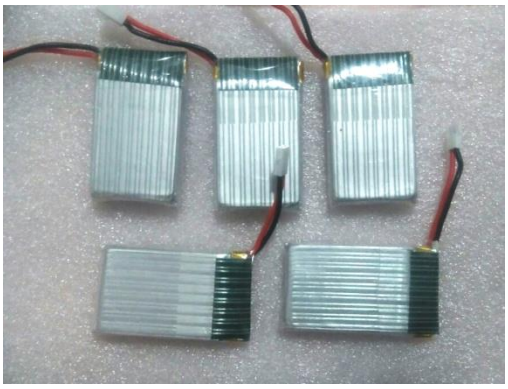

Figura 2.6: Baterías LiPo.

El estado del nivel de carga de las baterías se debe monitorear con la frecuencia debida del caso, ya que estas baterías si bien traen un cargado incluido, el cual se muestra en la figura 2.7. Existes límites de voltaje de carga y descarga que se deben tener en cuenta ya que caso contrario se pueden echar a perder las baterías.

A continuación se muestra los niveles críticos del estado de carga:

- Voltaje límite de carga: 4.2 V
- Voltaje nominal: 3.7 V

- Voltaje límite de descarga: 3.2 V

Figura 2.7: Cargador de baterías de LiPo.

CAPÍTULO 3

3. METODOLOGÍA DEL TRABAJO

La metodología que se utilizará para realizar el control de movimientos de agarre para una prótesis robótica de mano, mediante el uso del brazalete Myo, está basada en la programación de una placa Arduino Nano, debido a características físicas que posee, considerando que la tarjeta de control de movimientos de la prótesis debe cumplir con dimensiones que brinden comodidad al usuario, además de poseer un lenguaje de programación sencillo, abriendo así la posibilidad a futuras mejoras sobre el algoritmo de control.

3.1 Interfaz de comunicación entre Myo y Arduino

El brazalete Myo cuenta con reconocimiento de gestos preestablecidos, seguimiento de movimientos y medición de la actividad muscular, toda esta información, es transmitida de manera inalámbrica a través comunicación bluetooth, lo que presenta una ventaja al no depender de cableado. Para poder realizar la transmisión y recepción de datos de manera directa entre el brazalete Myo y la placa Arduino Nano, sin tener de por medio de un computador o teléfono inteligente, [11] es por ello que se optará por hace uso de un módulo bluetooth 4.0 de bajo consumo de energía, específicamente el módulo HM-11 debido a su pequeñas dimensiones.

3.1.1 Cargar Firmware Myo en módulo bluetooth

Debido a que la mayoría de los modelos bluetooth de bajo consumo de energía poseen un conjunto de características muy limitado, ninguno de estos es capaz de, bajo su estado de fábrica, conectarse directamente con el brazalete Myo. Este inconveniente se resuelve cargando un firmware personalizado para el Módulo bluetooth HM-11.

Para cargar el Firmware en el HM-11 Se hizo uso de una placa Arduino Uno en conjunto con el programa CCLoader, los pasos que se realizaron se detallan a continuación:

1. Para poder cargar el firmware en el bluetooth, así como para empezar a hacer uso del mismo para la comunicación directa entre el brazalete Myo y la placa Arduino, Es necesario soldar cables a los pines del HM-11 mostrados en la figura 3.1.

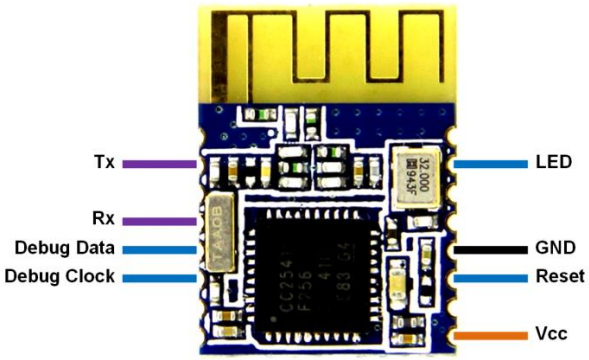

Figura 3.1: Pines a utilizar del módulo HM-11

2. Se debe realizar la descarga de la carpeta CCLoader-master, la cual se encuentra disponible en Githbug. [12]
3. Se debe cargar a la placa Arduino el programa CCLoader.ino que se halla dentro de la carpeta CCLoader-master, que descargamos con anterioridad.
4. Realizar las conexiones entre la placa Arduino y el HM-11, haciendo uso de los pines declarados dentro del código que hemos cargado a la placa Arduino, como se muestra en la figura 3.2.

   Es decir, para este caso Los pines 3.3V, GND, 6, 5 y 4 de la placa Arduino deben ser conectados a los pines Vcc, GND, 5, 6 y 11 del HM-11 respectivamente.

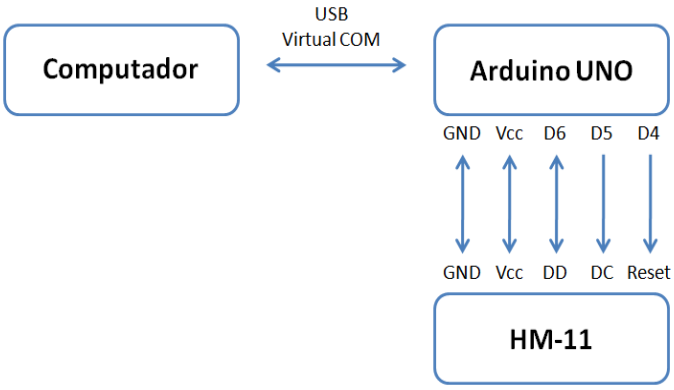

Figura 3.2: Conexión de Hardware

5. Abrir Símbolo de sistema y escribir cmd como se muestra en la figura 3.3.

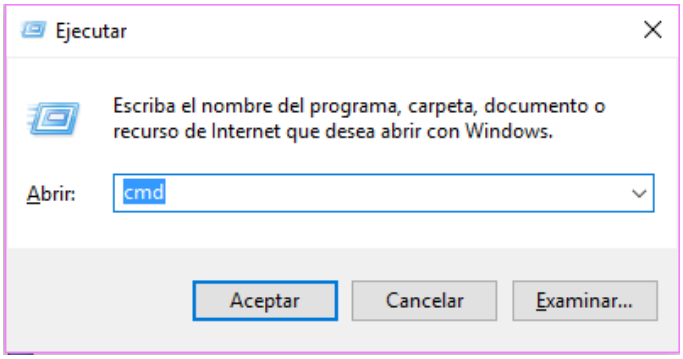

Figura 3.3: Abrir Símbolo de sistema

6. Escribir el comando cd + ubicación de CcLoader.exe, luego el programa nos indica que debemos ingresar lo siguiente:

   Dirección del archivo CcLoader.exe - # de puerto COM donde se encuentra conectada la tarjeta Arduino UNO - Dirección del archivo .Bin – 1 en caso de utilizar un Arduino Leonardo ó 0 en caso de utilizar cualquier otra placa Arduino. Se muestra un ejemplo en la figura 3.4.

```
C:\Users\equipo1>cd C:\Users\equipo1\Desktop\CCLoader-master\Windows\

C:\Users\equipo1\Desktop\CCLoader-master\Windows>CCLoader.exe
Copyright (c) 2013 RedBearLab.com
CCLoader.exe version 0.5
Invalid parameters.
Usage: CCLoader.exe <com number> <bin file> <device>
Example: CCLoader.exe 2 abc.bin 0
 <device>: 0 -- Default (e.g. UNO)
           1 -- Leonardo
```

Figura 3.4: Ejecutar CCLoader

7. Una vez finalizada la carga el HM-11 está listo para poder conectarse con el brazalete Myo. Sin embargo para la primera conexión del HM-11 con el brazalete Myo, en caso de que el indicador LED de conexión bluetooth no se encienda, se recomienda desconectar la alimentación del HM-11, conectar el brazalete Myo con su adaptador USB a la computadora hasta que se encienda el indicador LED, luego desconectarlo e inmediatamente energizar el módulo HM-11.

8. Finalmente para empezar con la transmisión y recepción de datos con Arduino Nano se deben realizar las conexiones que se indican en la figura 3.5.

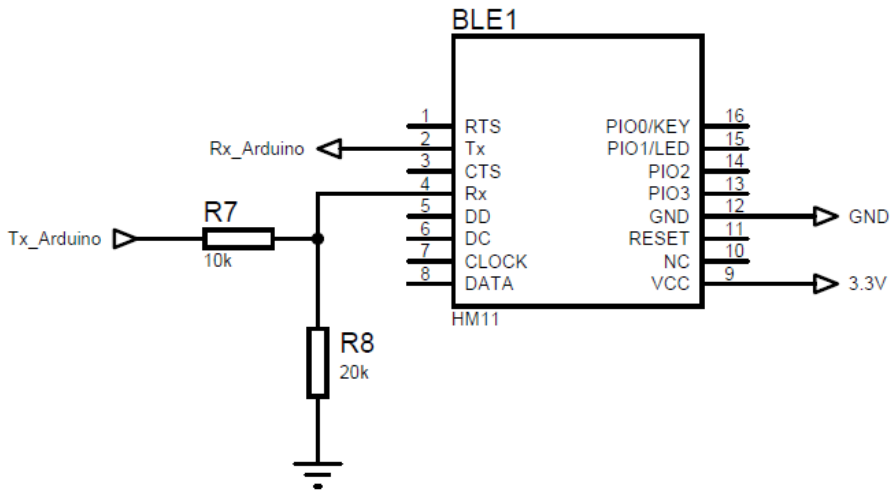

Figura 3.5: Esquema de conexión HM-11

3.2 Alimentación y estado de batería

Tanto la etapa de control como de la etapa de potencia, serán alimentaos por Baterías LiPo de una celda, que poseen un voltaje nominal de 3.7V, debido a su pequeño tamaño además de ser capaces de entregar hasta 1,2A.

Para la alimentación de la etapa de control, se usarán dos baterías LiPo conectadas en serie, directamente conectado al pin de entrada e voltaje Vin de la placa Arduino Nano, ya que este tiene un rango nominal de entre 7V a 12V, por lo que conectar las barrerías directamente no presenta inconveniente. En cuanto a la alimentación de del módulo bluetooth se realizara por medio de la salida de voltaje regulado de 3.3V que proporciona la placa Arduino Nano. De la misma forma que en la etapa de control, la etapa de potencia también tendrá como suministro de energía a dos baterías LiPo conectadas en serie con la variante de que se añade una pequeña etapa de regulación de voltaje como se muestra en la figura 3.6. Con la finalidad de entregar a los servomotores un voltaje fijo de 5V.

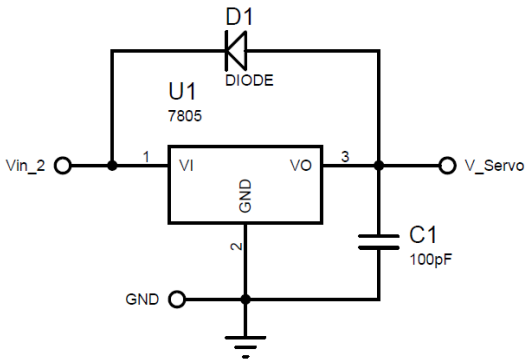

Figura 3.6: Regulador para alimentación de servomotor

Ya que el planteamiento del proyecto es tener control sobre una prótesis completamente portátil, uno de los factores a tener en consideración es el estado de la batería de todos los sistemas involucrados, sobre todo de las baterías LiPo que tienen voltajes límites de funcionamiento tanto en estado de carga como descarga. En cuanto a la carga de las baterías LiPo, si bien también tiene un límite de voltaje de carga, que a medida que se sobrepasa los 4.2V incrementa el riesgo de incendiarse la batería, para este proyecto no presenta

inconvenientes, ya que no se implementara ningún circuito de sistema de carga, se mantendrá el método de carga, mediante el cargador que viene de fábrica con las baterías. Por otro lado en el estado de descarga de las baterías, su voltaje no puede estar por debajo de los 3.2V, pasado este rango la batería puede llegar a tener daños irreversibles en la celda, acabando así con su vida útil.

Durante las pruebas se evidencio el caso de que en un par de baterías conectadas en serie, una de las baterías puede llegar a bajar su voltaje a niveles críticos cuando la otra puede aún estar en niveles de voltaje nominales, es por ello que se torno necesario el saber el estado actual de cada batería LiPo de manera individual, para evitar en lo posible de que ninguna de estas quede totalmente descargada.

El monitoreo del estado del par de baterías utilizadas tanto para la alimentación del sistema de control para el sistema de fuerza, se realiza siguiendo el esquema de le la Figura 3.7 y Figura 3.8 para el sistema de control y fuerza respectivamente, Donde A0, A1, A2 y A3 son entradas analógicas de la placa Arduino Nano.

En las entradas A0 y A2 se marca el voltaje de las Baterías V2 y V4 respectivamente, para obtener el voltaje de las baterías V1 y V3 bastará con seguir la ecuación 3.1, y 3.2 respectivamente.

$$V1 = 2(A1) - V2 \qquad (3.1)$$

$$V3 = 2(A3) - V4 \qquad (3.2)$$

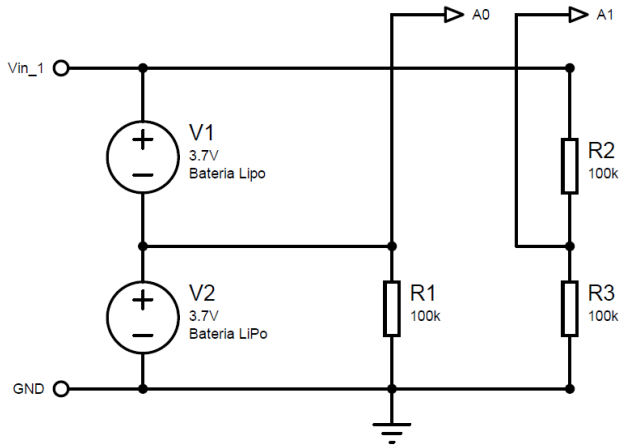

Figura 3.7: Estado de baterías en placa Arduino

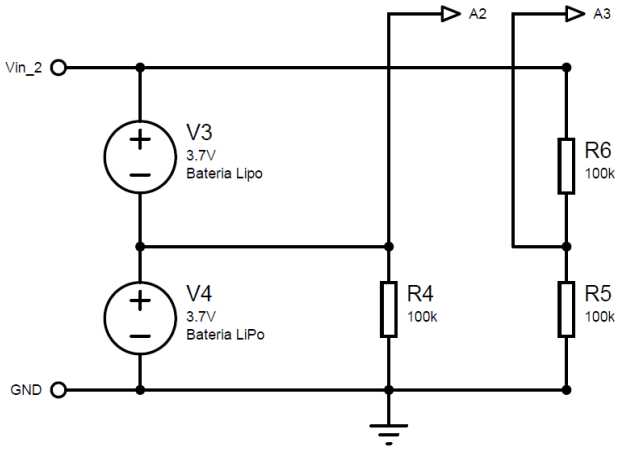

Figura 3.8: Estado de baterías en servomotores

3.3 Visualización y almacenamiento de datos EMG

Los sensores de EMG (electromiografía), del brazalete Myo hacen referencia a sus 8 sensores mioeléctricos, quienes son los encargados de captar la actividad muscular del antebrazo, la cual podemos visualizar en tiempo real mediante gráficas, haciendo uso de su conexión con Arduino, mediante comunicación serial USB, con un sencillo programa realizado en Matlab-Simulink, que se muestra en la figura 3.9. Aunque cabe recalcar que para el presente proyecto, los movimientos de control de la prótesis no se llevaran a cabo en base a la actividad muscular del usuario, debido que no es posible realizar pruebas y toma

de datos de la actividad muscular específica del usuario para quien está destinada la prótesis.

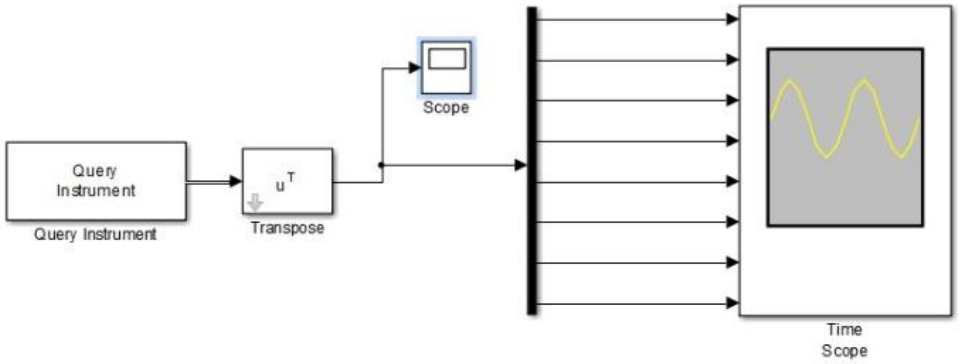

Figura 3.9: Visualización datos EMG

En cuanto al almacenamiento de los datos EMG receptados en la placa Arduino, estos pueden ser almacenados en documentos de Excel mediante un pequeño código en Matlab como se muestra en la figura 3.10. Con almacenamiento de cinco datos por segundo.

```
1 -    dataEMG(:,1)=EMG.time;
2 -    dataEMG(:,2)=EMG.signals.values(:,1);
3 -    dataEMG(:,3)=EMG.signals.values(:,2);
4 -    dataEMG(:,4)=EMG.signals.values(:,3);
5 -    dataEMG(:,5)=EMG.signals.values(:,4);
6 -    dataEMG(:,6)=EMG.signals.values(:,5);
7 -    dataEMG(:,7)=EMG.signals.values(:,6);
8 -    dataEMG(:,8)=EMG.signals.values(:,7);
9 -    dataEMG(:,9)=EMG.signals.values(:,8);
10 -   xlswrite('EMG_Datos.xlsx',dataEMG,1);
```

Figura 3.10: Almacenamiento de datos EMG

En La figura 3.11 se muestra las gráficas en respuesta a la actividad muscular en una de las pruebas realizadas con Myo y Arduino.

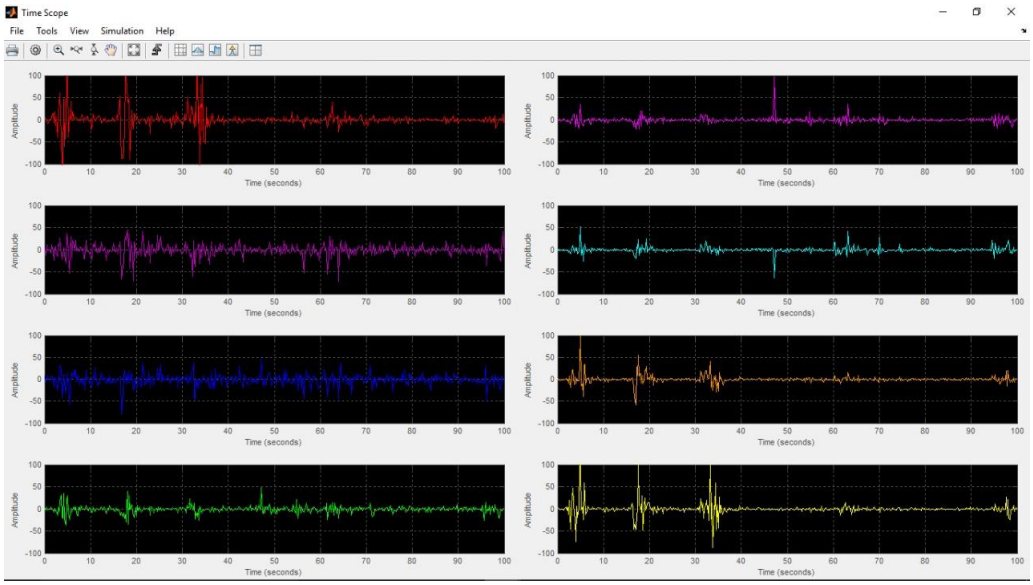

Figura 3.11: Graficas de datos EMG

3.4 Visualización y almacenamiento de datos IMU

De los sensores de IMU (Unidad de medida Inercial) que posee el brazalete Myo podemos obtener datos de acelerómetro y giroscopio, que son de vital importancia para la generación de patrones de movimiento en la prótesis. Haciendo uso de su conexión con Arduino, mediante comunicación serial USB con un computador podemos visualizar en tiempo real las gráficas de variación de los parámetros IMU que está recibiendo Arduino, con un sencillo programa realizado en Matlab-Simulink, que se muestra en la figura 3.12.

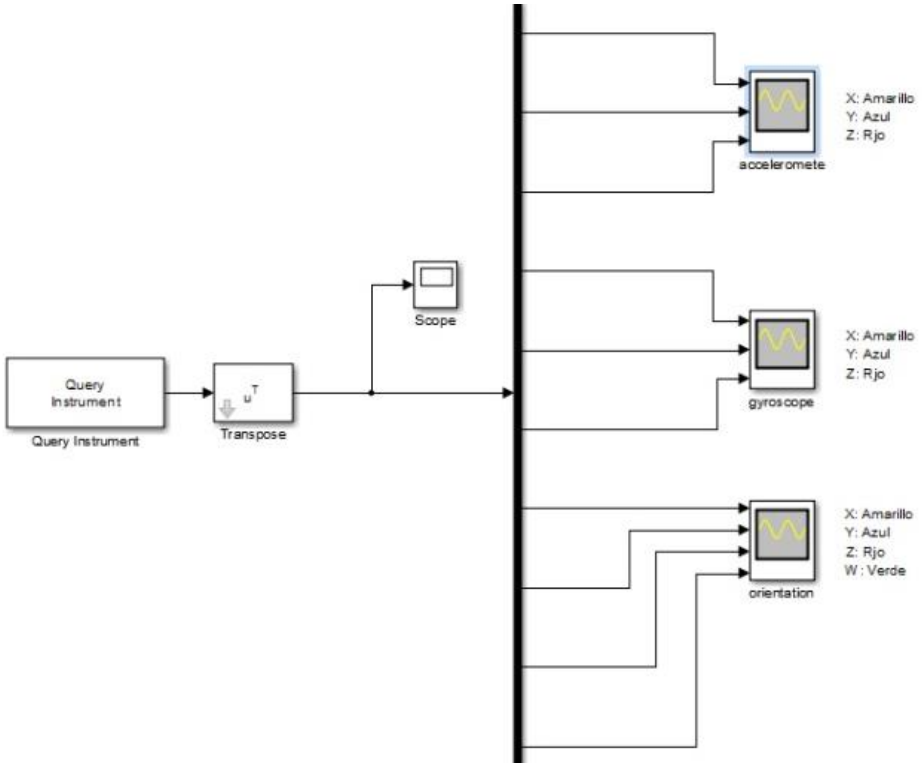

Figura 3.12: Visualización datos IMU

En cuanto al almacenamiento de los datos IMU receptados en la placa Arduino, estos pueden ser almacenados en documentos de Excel mediante un pequeño código en Matlab como se muestra en la figura 3.13. Con almacenamiento de 5 datos por segundo.

```
1 -    dataIMU(:,1)=IMU.time;
2 -    dataIMU(:,2)=IMU.signals.values(:,1);
3 -    dataIMU(:,3)=IMU.signals.values(:,2);
4 -    dataIMU(:,4)=IMU.signals.values(:,3);
5 -    dataIMU(:,5)=IMU.signals.values(:,4);
6 -    dataIMU(:,6)=IMU.signals.values(:,5);
7 -    dataIMU(:,7)=IMU.signals.values(:,6);
8 -    dataIMU(:,8)=IMU.signals.values(:,7);
9 -    dataIMU(:,9)=IMU.signals.values(:,8);
10 -   dataIMU(:,10)=IMU.signals.values(:,9);
11 -   dataIMU(:,11)=IMU.signals.values(:,10);
12 -   xlswrite('IMU_Datos.xlsx',dataIMU,1);
```

Figura 3.13: Almacenamiento de datos IMU

3.5 Ensamblaje de la prótesis impresa Flexy Hand 2

La prótesis con la que se llevan a cabo las pruebas del proyecto es Flexy Hand 2, En la figura 3.14 se puede observar cómo se ensamblan cada pieza de la prótesis, mientras que en la tabla 5. Se detallan cada una de las piezas.

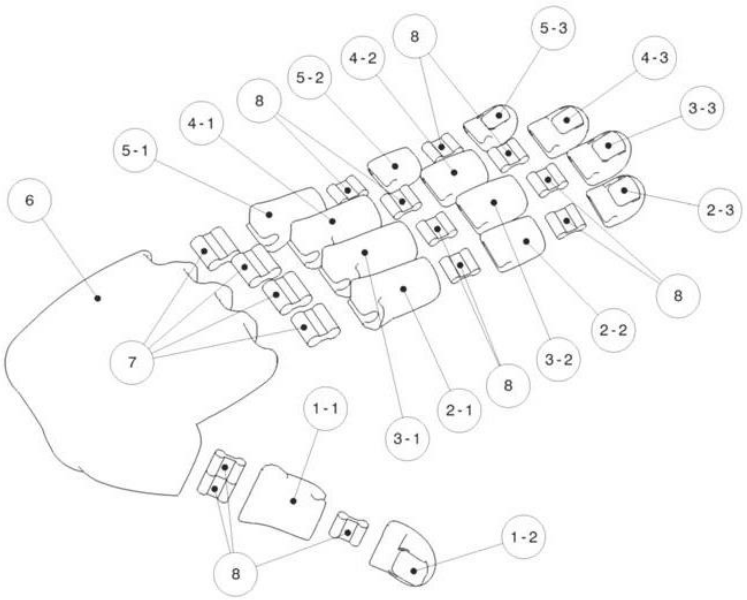

Figura 3.14: Partes de la prótesis Flexy Hand 2 [13]

Ítem	Descripción	Unidades
1-1	Falange proximal pulgar	1
1-2	Falange distal pulgar	1
2-1	Falange proximal índice	1
2-2	Falange medio índice	1
2-3	Falange distal índice	1
3-1	Falange proximal dedo medio	1
3-2	Falange medio dedo medio	1
3-3	Falange distal dedo medio	1
4-1	Falange proximal anular	1
4-2	Falange medio anular	1
4-3	Falange distal anular	1
5-1	Falange proximal meñique	1

5-2	Falange medio meñique	1
5-3	Falange distal meñique	1
6	Cuerpo de la mano	4
7	Palmar digital	4
8	Interfalángica proximal y distal	11

Tabla 5: Descripción de las partes de la prótesis Flexy Hand 2

La prótesis fue obtenía a través de impresión 3D, con material de plástico rígido para las la palma, agarradera de muñeca y falanges (distales, medias y proximal), y en plástico flexible fueron impresas todas las articulaciones o uniones de la prótesis, permitiendo así la movilidad de dedos. En la figura 3.15 se muestra el estado de reposo y contracción de uno de los dedos de la prótesis.

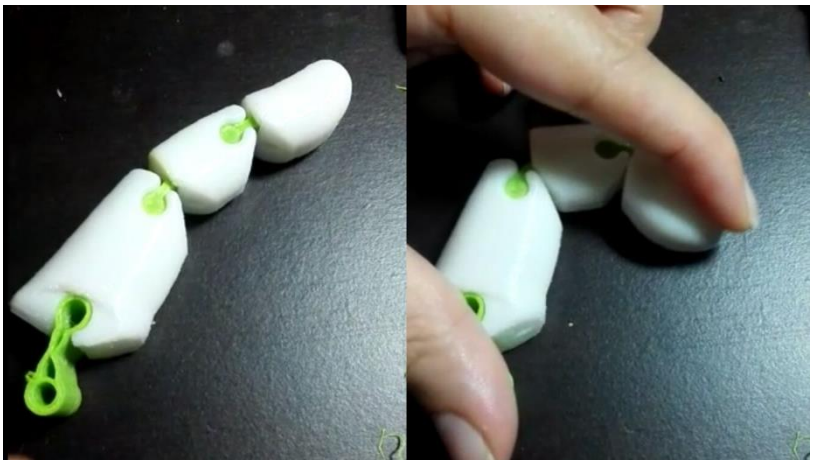

Figura 3.15: reposo y contracción de dedo

3.6 Control de actuadores

Debido a que en Arduino la librería servo hace uso de temporizadores que a su vez necesitan las librerías que permiten la interpretación de datos del brazalete Myo, se crean conflictos en el movimiento de los servos, durante las primeras pruebas los servo motores presentaban vibraciones aleatorias no deseadas, por lo que se opta por no hacer uso de la librería servo, y en su lugar sustituirla por líneas de código que permitan controlar el movimiento de los 5 servomotores por variación de ancho de pulso como se muestra en la figura 3.16.

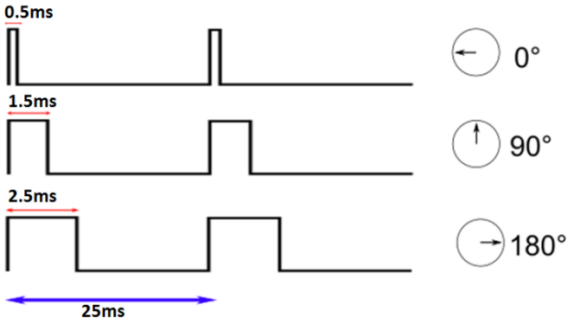

Figura 3.16: Control de posición de servo motor

3.7 Estructura general del software

La estructura general del software que se muestra en la figura 3.18 está compuesta por dos librerías, MyoBridge [14] que permite básicamente la comunicación bidireccional con el brazalete Myo y MyoIMUGestureController [15] quien se encarga de la lectura y reconocimiento gestos, además del programa principal donde se establecen los parámetros y condiciones para la generación de movimientos de la prótesis.

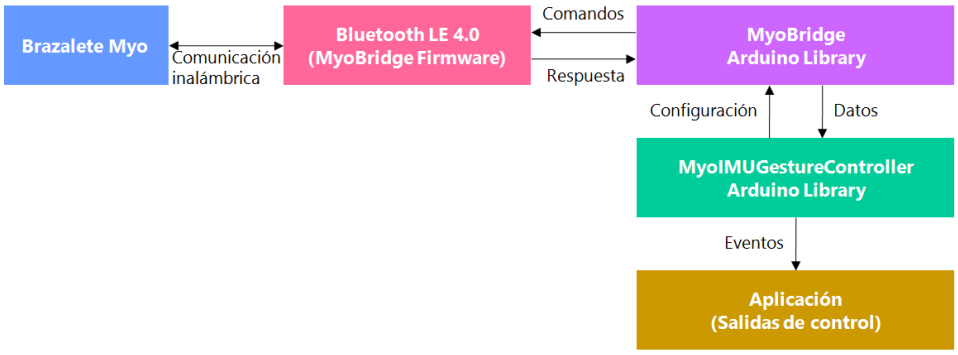

Figura 3.17: Diagrama de bloques de la estructura general del software

3.8 Diagrama de flujo

El diagrama de flujo del programa principal se muestra en la figura 3.18, done se explica la secuencia lógica de pasos realizados por la tarjeta controladora Arduino Nano.

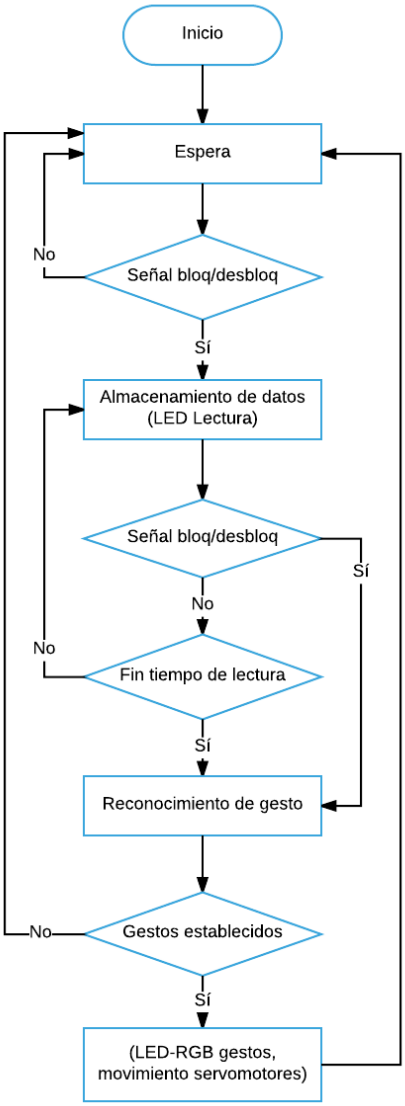

Figura 3.18: Diagrama de flujo de programa principal

CAPÍTULO 4

4. RESULTADOS

4.1 Lectura y almacenamiento de datos IMU y EMG

Una vez realizada la comunicación directa entre Arduino y el brazalete Myo, se procedió a realizar las primeras pruebas de lectura de los sensores IMU y EMG. De las cuales se obtuvieron los siguientes resultados:

De los datos EMG se obtuvo la información de cada uno de los 8 sensores mioeléctricos, que se visualizan en la figura 4.1.

	A	B	C	D	E	F	G	H	I	J
1	Tiempo	emg_1	emg_2	emg_3	emg_4	emg_5	emg_6	emg_7	emg_8	
2	0	-3	28	-24	30	10	1	-19	1	
3	0,2	-3	28	-24	30	10	1	-19	1	
4	0,4	-3	28	-24	30	10	1	-19	1	
5	0,6	-3	28	-24	30	10	1	-19	1	
6	0,8	-3	28	-24	30	10	1	-19	1	
7	1	-3	28	-24	30	10	1	-19	1	
8	1,2	-3	28	-24	30	10	1	-19	1	
9	1,4	-3	28	-24	30	10	1	-19	1	
10	1,6	-3	28	-24	30	10	1	-19	1	
11	1,8	-3	28	-24	30	10	1	-19	1	
12	2	-3	28	-24	30	10	1	-19	1	
13	2,2	-3	28	-24	30	10	1	-19	1	
14	2,4	-3	28	-24	30	10	1	-19	1	
15	2,6	-3	28	-24	30	10	1	-19	1	
16	2,8	-3	28	-24	30	10	1	-19	1	
17	3	-3	28	-24	30	10	1	-19	1	
18	3,2	-3	28	-24	30	10	1	-19	1	
19	3,4	-3	28	-24	30	10	1	-19	1	
20	3,6	-3	28	-24	30	10	1	-19	1	
21	3,8	-3	28	-24	30	10	1	-19	1	

Figura 4.1: Datos EMG

De los datos IMU se obtuvo la información del acelerómetro en los ejes x, y, z. Así también se pueden visualizar los datos obtenidos de giroscopio en los ejes x, y, z. Y por último tenemos acceso a los datos de posición representada en los cuaterniones x, y, z, w. como se muestra en la figura 4.2.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Tiempo	acel(x)	acel(y)	acel(z)	girosc(x)	girosc(y)	girosc(z)	x	y	z	w	
2	0	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
3	0,2	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
4	0,4	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
5	0,6	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
6	0,8	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
7	1	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
8	1,2	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
9	1,4	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
10	1,6	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
11	1,8	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
12	2	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
13	2,2	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
14	2,4	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
15	2,6	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
16	2,8	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
17	3	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
18	3,2	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
19	3,4	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
20	3,6	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	
21	3,8	6	1896	-314	1899	-307	-143	-1	0	-1	-1912	

Figura 4.2: Datos IMU

4.2 Reconocimiento de gestos

El reconocimiento de gestos se lleva a cabo durante el estado de lectura, el cual comienza con una señal de actividad muscular, o un movimiento de fuerza, esto se traduce en movimiento bruscos de la muñeca o bien una simple contracción muscular, el lapso de lectura si bien esta temporizado, puede terminar o bloquearse mediante la misma señal de actividad muscular que se realizó inicialmente. Para su visualización, se hace uso de un LED indicador de estado, como se muestra en la figura 4.3, el cual permanecerá encendido únicamente en el lapso de adquisición de datos para el reconocimiento de gestos.

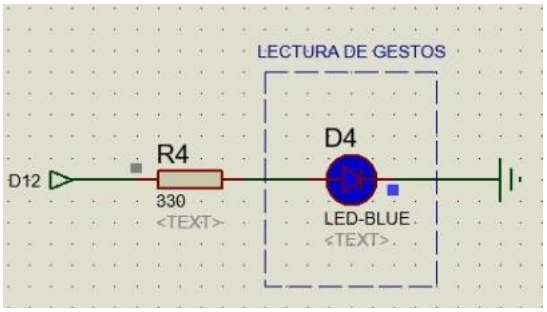

Figura 4.3: Indicador estado de lectura

Una vez finalizado el tiempo de adquisición de datos, estos son evaluados y si el gesto realizado durante el estado de lectura es reconocido, como se muestran en la figura 4.4. Se proceden a realizar las debidas instrucciones correspondientes a cada gesto definido.

Figura 4.4: Detección de gesto mostrada en monitor serie de Proteus

Si bien el terminal virtual de Proteus o el mismo monitor serial de Arduino, son herramientas útiles en las simulaciones para conocer el gesto interpretado por el sistema, únicamente podemos usarlos durante las simulaciones en un ordenador, por lo que otra forma de saber si el gesto que fue o no reconocido por el sistema es a través de indicadores LED, y teniendo presente el número reducido de pines de salida digital que pose Arduino Nano, se muestra el gesto actual haciendo uso de un LED RGB.

El LED RGB muestra un color verde como se muestra en la figura 4.5, cuando es detectado que el brazo ha realizado un movimiento hacia arriba.

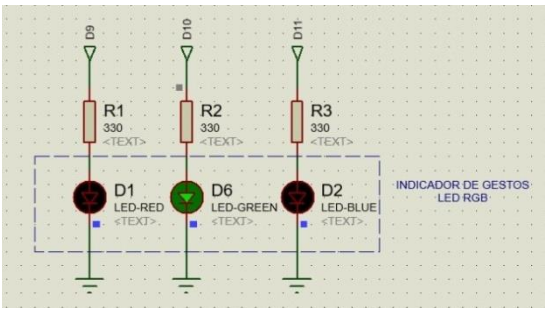

Figura 4.5: indicador de gesto en verde

El LED RGB muestra un color amarillo como se muestra en la figura 4.6, cuando es detectado que el brazo ha realizado un movimiento hacia abajo.

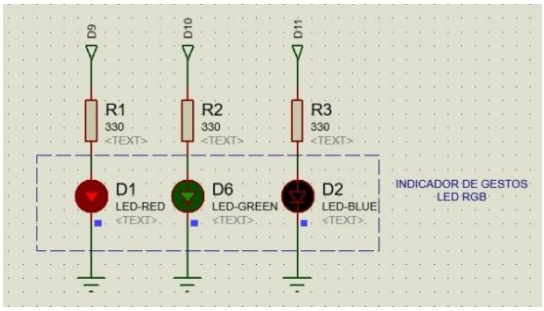

Figura 4.6: indicador de gesto en amarillo

El LED RGB muestra un color naranja como se muestra en la figura 4.7, cuando es detectado que el brazo ha realizado un movimiento hacia la izquierda.

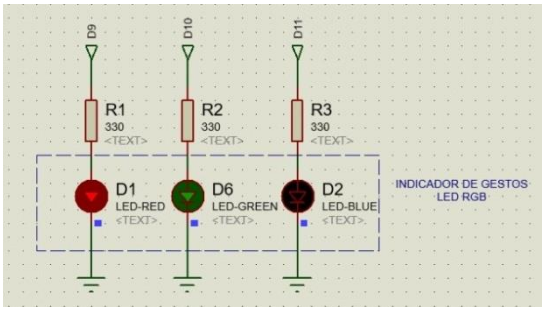

Figura 4.7: indicador de gesto en naranja

El LED RGB muestra un color rojo como se muestra en la figura 4.8, cuando es detectado que el brazo ha realizado un movimiento hacia la derecha.

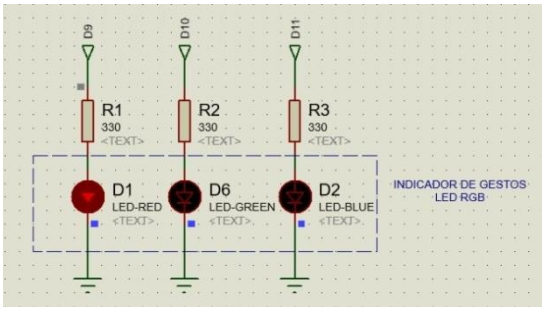

Figura 4.8: indicador de gesto en rojo

El LED RGB muestra un color azul como se muestra en la figura 4.9, cuando es detectado que el brazo ha realizado un movimiento de rotación de muñeca en sentido horario.

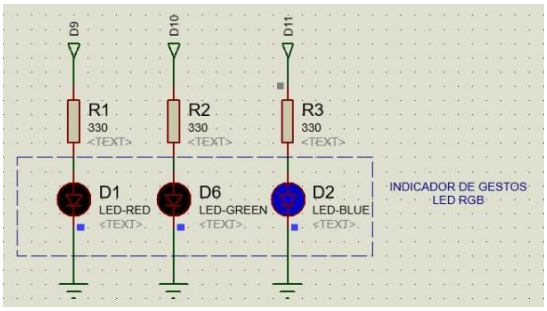

Figura 4.9: indicador de gesto en azul

El LED RGB muestra un color violeta como se muestra en la figura 4.10, cuando es detectado que el brazo ha realizado un movimiento de rotación de muñeca en sentido anti horario.

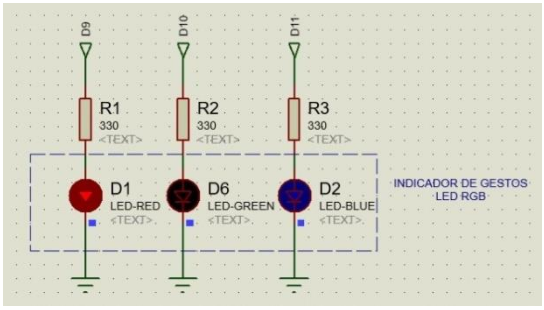

Figura 4.10: indicador de gesto en violeta

4.3 Alarma de batería baja

El estado de alarma de batería baja, está representado por un LED rojo, el cual se enciende en el momento en que una o más de las baterías de alimentación bajan del nivel mínimo de voltaje establecido, como se muestra en la figura 4.11.

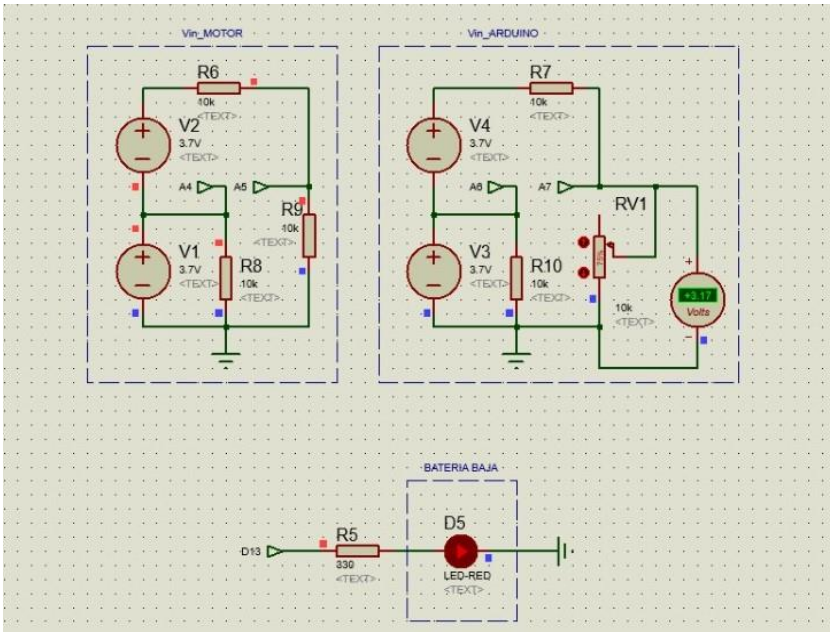

Figura 4.10: Alarma batería baja

Mientras que en la figura 4.12 se observa que la alerta permanece apagada si todas las baterías de alimentación cumplen con la condición de poseer niveles de voltaje superiores al mínimo permitido.

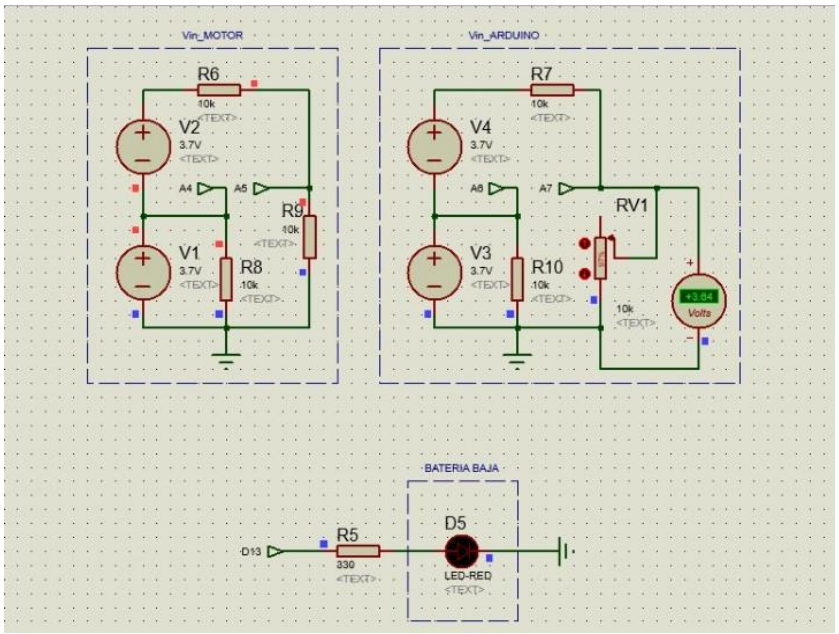

Figura 4.12: Niveles de batería aceptables

4.4 Primeras pruebas - movimientos de señas

Para las primeras validaciones de código haciendo uso de la prótesis impresa, se realizaron movimientos de flexión y extensión completa de dedos específicos para la simulación de señas, en la tabla 6 se detallan los primeros movimientos realizados con la prótesis.

MOVIMIENTO	DESCRIPCIÓN
Seña	Número 1, Señalar con dedo índice
Seña	Número 2, extender únicamente índice y medio
Seña	Número 3, flexionar únicamente pulgar e índice
Seña	Número 4, flexionar únicamente pulgar
Puño	Contraer todos los dedos

Tabla 6: Movimientos de señas

En la figura 4.13 se muestra una de las señas realizadas durante las primeras pruebas, correspondiente al número 3 como resultado de contraer únicamente pulgar e índice.

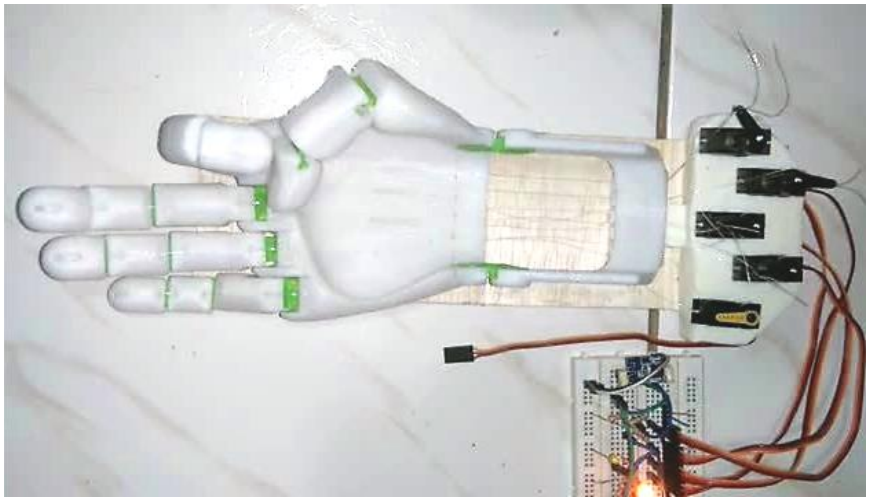

Figura 4.13: Movimiento de señas con prótesis impresa

4.5 Movimientos de agarre

Actualmente están establecidos como respuesta a los gestos censados, cuatro movimientos de señas y cinco movimientos de agarre como se muestra en la tabla 7.

MOVIMIENTO	DESCRIPCIÓN
Agarre	Dar la mano
Agarre	Agarre cilíndrico
Agarre	Sujetar jarro
Agarre	Sujetar mouse (Clic derecho , Clic izquierdo)
Seña	Número 1, Señalar con dedo índice
Seña	Número 2, extender únicamente índice y medio
Seña	Número 3, flexionar únicamente pulgar e índice
Seña	Abrir la mano, extender todos los dedos

Tabla 7: Lista de movimientos

A continuación se describirán en detalle cada uno de los movimientos de agarre establecidos:

4.5.1 Dar la mano

Esta postura se muestra se logra cuando se flexionan los dedos meñique, anular y medio, con el fin de poder dar saludos con apretón de manos.

Figura 4.14: Movimiento para dar la mano

4.5.2 Agarre cilíndrico

Esta postura se muestra en la figura 4.15 y se logra cuando todos los dedos a excepción de pulgar se flexionan, con el fin de sujetar objetos cilíndricos.

Figura 4.15: Movimiento de agarre cilíndrico

4.5.3 Sujetar Jarro

Esta postura se muestra en la figura 4.16 y se logra flexionando los dedos meñique y anular, de modo que los dedos medio e índice se coloquen en la oreja de una taza o jarro, para luego flexionarse y poder sujetarla.

Figura 4.16: Movimiento para sujetar jarro

4.5.4 Sujetar mouse

Esta postura se muestra en la figura 4.17 y se logra flexionando el dedo meñique y medianamente el anular y el pulgar, de modo que estos dedos en conjunto con el apoyo de la palma permitan controlar el deslizamiento de un mouse sobre una superficie plana, además, una vez estemos en esta postura se desbloquean dos movimientos mas, dar clic derecho con el dedo medio y clic izquierdo con dedo índice.

Figura 4.17: Movimiento para sujetar mouse

4.5.5 Sujetar con dedos índice y medio

Esta postura se muestra en la figura 4.18 y se logra cuando el dedo índice y medio se flexionan para sujetas objetos de pequeñas dimensiones, como monederos, pelotas de tenis, pelotas de ping pong, etc. Y se adapta a objetos diversos tamaños ya que en el dedo índice se colocó un sensor de fuerza resistivo.

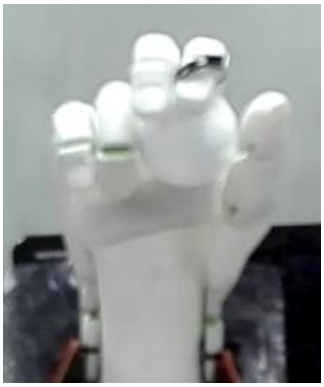

Figura 4.18: Sujetando pelota con índice y medio

4.6 Placa de circuitos impresos

Para demostración del funcionamiento del sistema de control de movimientos, sobre un prototipo de prótesis de mano impresa en 3D se realizó el diseño de una PCB o placa de circuitos impresos, la cual se puede observar en vista frontal en la figura 4.19 y su vista posterior en la figura 4.20. La placa posee las dimensiones de 7.3 cm x 5.1 cm.

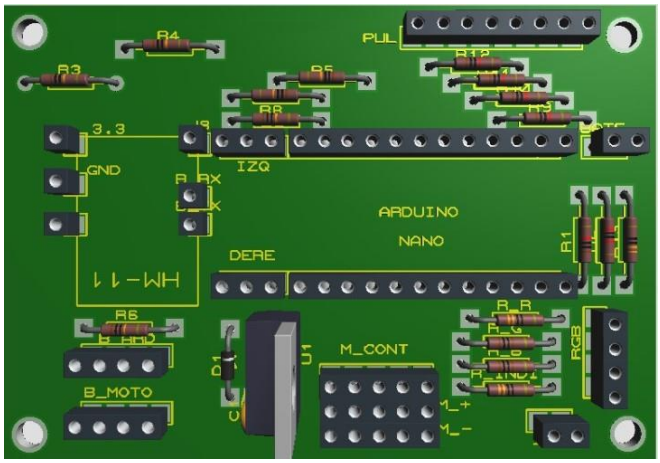

Figura 4.19: Vista frontal PCB

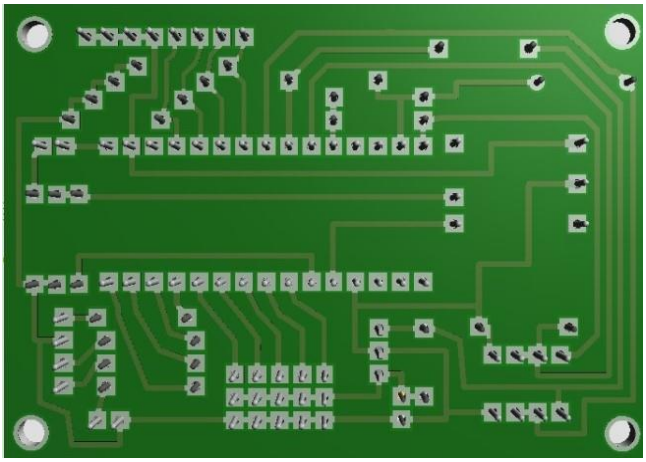

Figura 4.20: Vista posterior PCB

En el diseño de la PCB incluye conectores para sensores de fuerza que den sensibilidad de tacto a la prótesis, pero estos no fueron incluidos en la implementación debido a que durante el lapso de tiempo de desarrollo del proyecto integrador no fue posible adquirir dichos sensores.

4.7 Análisis Económico

En esta sección se realizará un análisis económico que permitirá conocer la factibilidad económica del desarrollo de un sistema de control de movimientos de agarre para una prótesis robótica de mano. En la tala 8 se muestra la lista de precios de los componentes electrónicos utilizados en el desarrollo del presente proyecto.

PART.	DESCRIPCIÓN	CANT	C/UNITARIO	TOTAL
1	Myo	1	$199.00	$199.00
2	Arduino NANO	1	$12.00	$12.00
3	Batería LiPo 1S	4	$5.00	$20.00
4	MG90S Servomotor	5	$7.50	$37.50
5	HM-11 Bluetooth	1	$8.00	$8.00
6	LM7805	1	$1.50	$1.50
7	Resistencia	16	$0.20	$3.20
8	Capacitor	1	$0.20	$0.20
			TOTAL:	$280.00

Tabla 8: Lista de precios de componentes electrónicos

Del análisis económico realizado sobre los componentes empleados, se obtuvieron los datos estadísticos mostrados en la figura 4.21 done se puede observar que sobre el costo total, el brazalete Myo representa un 71% de la inversión que se necesita para llevar a cabo la implementación netamente electrónica del primer prototipo del proyecto.

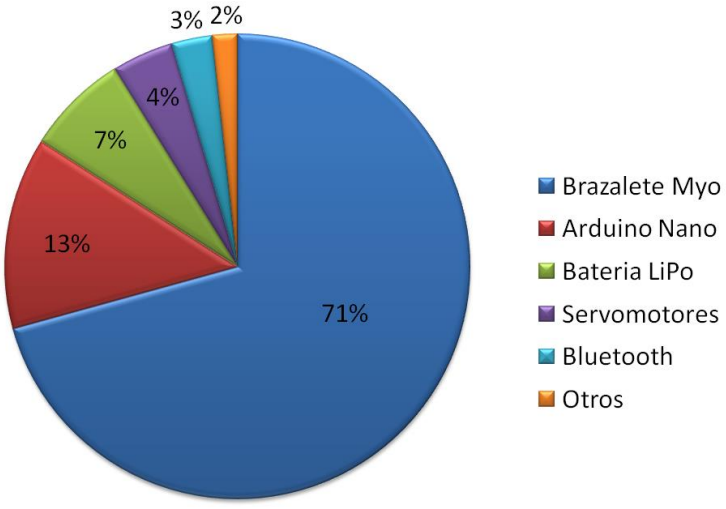

Figura 4.21: Porcentaje de costos

CONCLUSIONES Y RECOMENDACIONES

El control de la prótesis de mano fue realizado por medio de comunicación bluetooth 4.0 entre el Brazalete Myo y la tarjeta Arduino Nano, con esto se eliminó el uso de cables para la captación de las señales provenientes de los sensores del brazalete Myo, facilitando así el movimiento y la manipulación de la prótesis.

La implementación de la prótesis de mano robótica que utiliza un controlador que funciona sobre software libre permitió tener acceso a una gran cantidad de información en la red sin la necesidad de realizar inversiones económicas que elevaran el presupuesto inicial de la prótesis.

Los patrones de lectura que debe leer el brazalete Myo son generados a partir de movimientos del antebrazo que se logran realizar con total naturalidad, lo cual representa una gran ventaja ya que facilita el manejo de la prótesis.

El uso de tecnología de impresión en 3D puede facilitar en gran manera la fabricación de prótesis personalizadas, ya que, en su mayoría no todas las personas poseen los mismos patrones de deformación en sus extremidades.

El prototipo es capaz de realizar nueve movimientos, de los cuales cuatro son de realización de señas y cinco son movimientos de agarre.

Si bien la prótesis impresa permitió validar los movimientos de agarre definidos en el sistema de control, esta presento severos inconvenientes debido al material que se utilizó en su impresión, el plástico rígido genera gran deslizamiento en la gran mayoría de las texturas que la prótesis simula sujetar.

Se recomienda revestir la prótesis de un material que disminuya el deslizamiento sobre su superficie, en caso de que esta sea impresa en plástico rígido.

Si se añaden sensores de fuerza resistivos, a los demás dedos de la prótesis, esta se podrá adaptar con mayor precisión a las diversas dimensiones de objetos que se desee agarrar.

Para futuros avances sobre el proyecto se recomienda tener en consideración que los servomotores utilizados para dar movimiento a la prótesis, no poseen un torque

lo suficientemente elevado como para ejercer la fuerza suficiente sobre los tendones de la misma y levantar objetos de considerable peso.

BIBLIOGRAFÍA

[1] Cesar Augusto, Mariela Muñoz, Oscar Vivas, Carlos Cavira, “Diseño y construcción de la prótesis robotica de mano UC-1,” Proyecto de investigacion, Universidad del Cauca, Popayan, Colombia, 2010.

[2] Anon, (2017). [online] Disponible en: https://www.imparablesdocumental.com/richard-van-as-e-ivan-owen/ https://www.unocero.com/2013/05/28/altruista-protesis-de-mano-a-partir-de-impresora-3d/

[3] Thalmic Labs, (2015, noviembre). Meet the Man With a Myo-Controlled Robotic Arm. [online] The Lab. Disponible en: http://developerblog.myo.com/meet-the-man-with-a-myo-controlled-robotic-arm/

[4] "Flexy-Hand2", Thingiverse.com, 2014. [Online]. Disponible en: https://www.thingiverse.com/thing:380665.

[5] "Myo Gesture Control Armband", Myo.com, 2014. [Online]. Disponible en: https://www.myo.com/techspecs.

[6] Thalmic Labs, Myo Black. 2014.

[7] "Myo SDK 0.9.0: Myo SDK Manual", Developer.thalmic.com, 2014. [Online]. Disponible en: https://developer.thalmic.com/docs/api_reference/platform/index.html

[8] P. Bernhardt, "Myo Bluetooth Protocol Released", Thalmic Labs, 2015. [Online]. Disponible en: http://developerblog.myo.com/myo-bluetooth-spec-released/.

[9] "Introduction to Bluetooth Low Energy", Learn.adafruit.com, 2015. [Online]. Disponible en: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/.

[10] R. FAQ and I. Batteries, "Introduction to Lipo Batteries", iCharger NZ, 2017. [Online]. Disponible en: http://www.icharger.co.nz/buying/resources-faq/introduction-to-lipo-batteries/.

[11] "Featured Developer: Connecting Myo to an Arduino Board", Thalmic Labs, 2016. [Online]. Disponible en: http://developerblog.myo.com/connecting-myo-to-arduino/.

[12] "Burn BLE Mini (CC2540) firmware using Blend Micro board.", GitHub, 2015. [Online]. Available: https://github.com/RedBearLab/CCLoader.

[13] Flexy-Hand 2 Parts Diagram.. 2014.

[14] "A high-level Arduino library and custom firmware for the HM-11 (CC2541 SoC) to enable direct Myo Armband and Arduino communication", GitHub, 2015. [Online]. Disponible en: https://github.com/vroland/MyoBridge.

[15] "MyoIMUGestureController", GitHub, 2015. [Online]. Disponible en: https://github.com/vroland/MyoIMUGestureController.

[16] Setiawan, Cahyo, “Tangan Robot Bionik dengan Kendali Otot Berbasis EMG”, Computer Engineering, Electronic Engineering Polytechnic Institute of Surabaya, Surabaya, Indonesia, 2015.

[17] Francesco V. G. Tenore, Ander Ramos, Amir Fahmy, Soumyadipta Acharya, Ralph Etienne-Cummings and Nitish V. Thakor, “Decoding of Individuated Finger Movements Using Surface Electromyography,”

ANEXOS

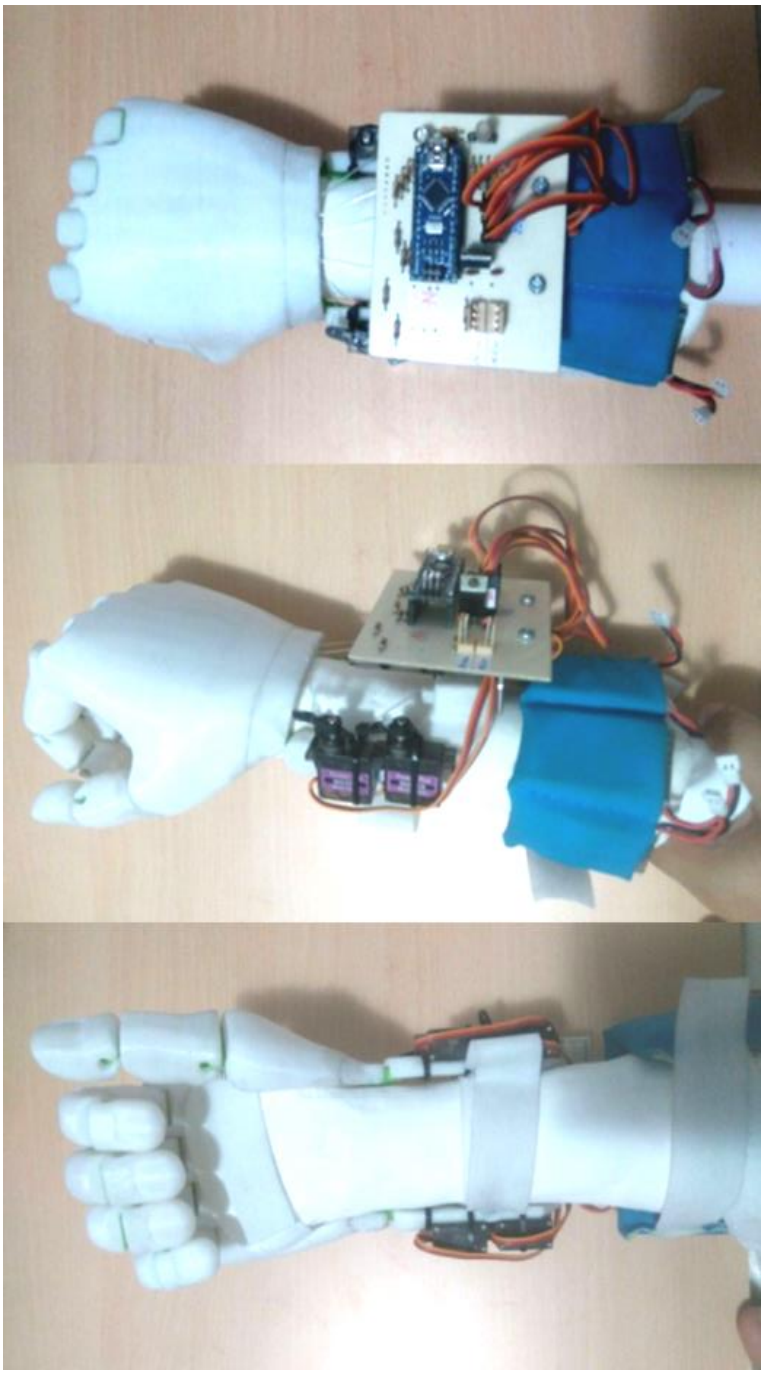

Figura Anexo 1: Prototipo de prótesis de mano

RGB- COLOR	MOVIMIENTO	MENÚ 1	MENÚ 2
	ARRIBA	Señalar con 1 dedo	Saludo
	ABAJO	Abrir mano	Abrir mano
	IZQUIERDA	Alzar 2 dedos	Agarre cilíndrico
	DERECHA	Alzar 3 dedos	Sujetar jarro
	ROTAR CSR	Agarre dedo índice y medio	Sujetar mouse
	ROTAR SR	Ir a Menú 2	Ir a Menú 1

Figura Anexo 2: Patrones de movimiento e indicadores de gestos

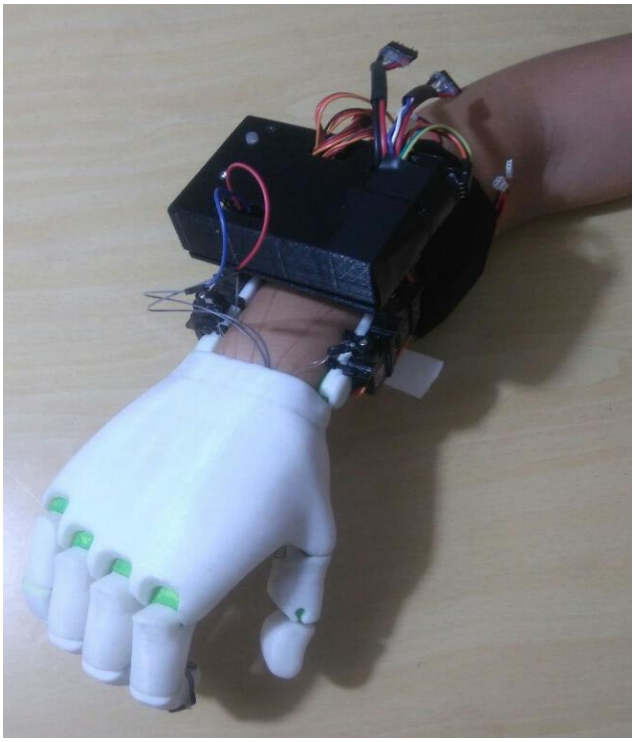

Figura Anexo 3: Prototipo de prótesis de mano

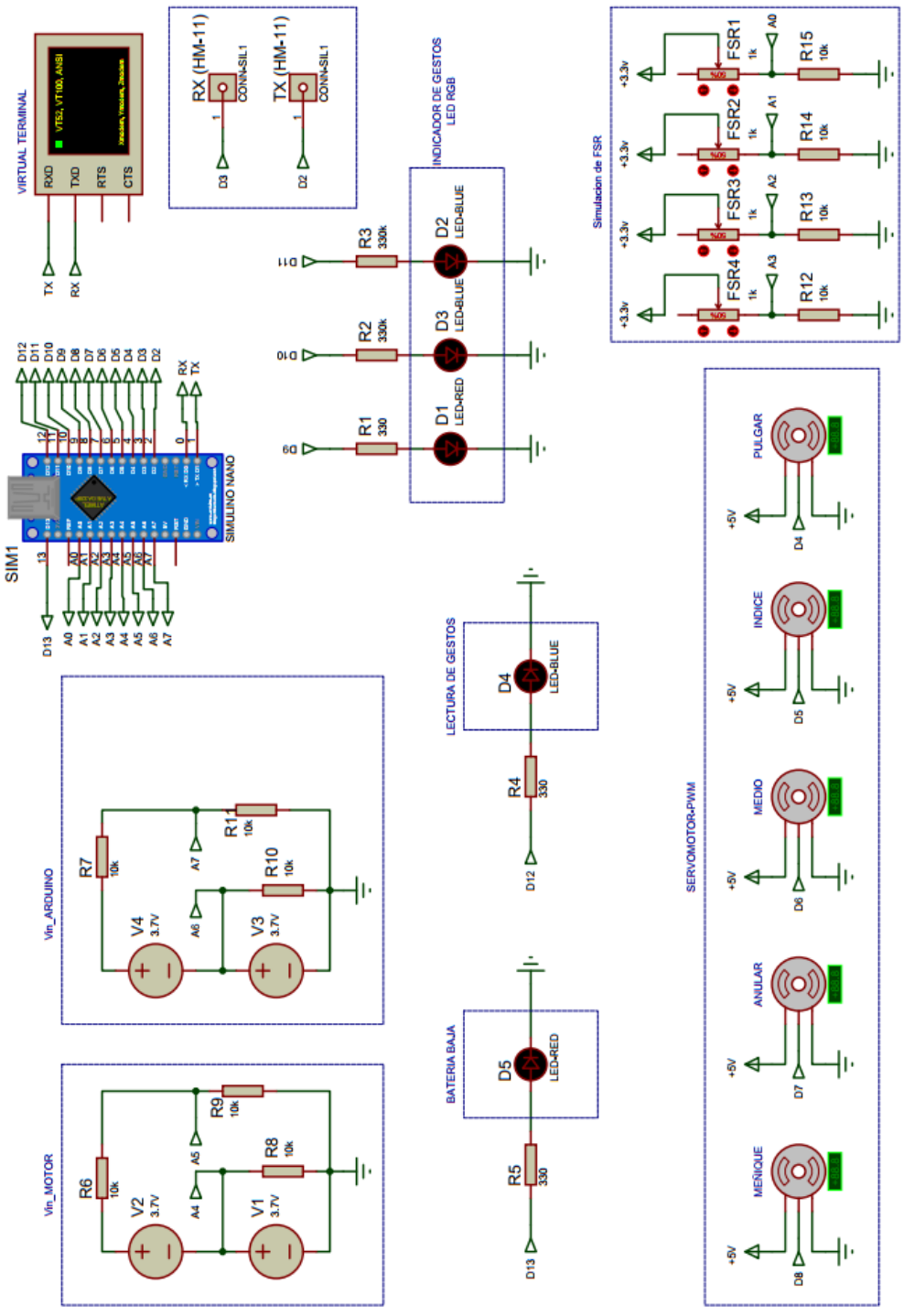

Figura Anexo 4: Diagrama de conexiones

Anexo 5: Código del programa de la tarjeta controladora

```
#include <MyoBridge.h>
#include <SoftwareSerial.h>
#include <MyoIMUGestureController.h>
//SoftwareSerial connection to MyoBridge
SoftwareSerial bridgeSerial(2,3);   ///////(2,3) (Rx arduino,Tx arduino)
//initialize MyoBridge object with software serial connection
MyoBridge bridge(bridgeSerial);
int R_Pose = 9;       // Salida R LED-RGB indicador de pose en el PIN 9
int G_Pose = 10;      // Salida G LED-RGB indicador de pose en el PIN 10
int B_Pose = 11;      // Salida B LED-RGB indicador de pose en el PIN 11
int Led_Lectura = 12;  // Salida pin de estado de lectura
int Alerta = 13;       // Salida pin de estado de lectura
String GESTO;
int Servo1 = 4;       // PUGAR
int Servo2 = 5;       // INDICE
int Servo3 = 6;       // MEDIO
int Servo4 = 7;       // ANULAR
int Servo5 = 8;       // MEÑIQUE
float pausa;
int anguloCerrar =179;    // Angulo max índice, medio, anular, meñique
int anguloAbrir = 5;     // Angulo min índice, medio, anular, meñique
int anguloAbrir1 = 175;   // Angulo min PUGAR
int anguloCerrar1 = 0;  // Angulo max PUGAR
bool MovAgarre = false;
```

```
bool MovRaton = false;
//  Estado de cada dedo // True= abierto // False= Cerrado
bool d1  = true; //true= dedo1 abierto, false= dedo1 cerrado
bool d2 = true;
bool d3 = true;
bool d4 = true;
bool d5 = true;
///////////////  Orden de Cerrar dedos
bool cerrar1;
bool cerrar2;
bool cerrar3;
bool cerrar4;
bool cerrar5;
//////////////   VARIABLES DE LECTOR DE BAT
float BateriaBaja= 3.6; ///Valor mínimo de volt
int LecturaCero = A4;
int LecturaUno = A5;
int LecturaDos = A6;
int LecturaTres = A7;
int ValCero;
int ValUno;
int ValDos;
int ValTres;
float bateriaUno =0;
float bateriaDos=0;
```

```
float bateriaTres =0;
float bateriaCuatro =0;
//////////////////////////////////// FSR
int FSR1 = A0;
int FsrLectura1;
int fuerzaMAX = 210;


//a function to inform us about the current state and the progess of the connection to the Myo.
void printConnectionStatus(MyoConnectionStatus status) {
  //print the status constant as string
Serial.println(status);
}
//_____________________________________________________________________________
///Esta función es llamada cuando un bloqueo / desbloqueo que sucede
void updateLockOutput(bool locked) {
  digitalWrite(Led_Lectura, !locked);
}
//-----------------------------------------------------------------
void updateControls(GestureType gesture) {
  GESTO = gestureToString(gesture);
  //Serial.println(gestureToString(gesture));
  bridge.vibrate(1);
  if (gesture == ARM_UP) {   //Verde
      analogWrite(R_Pose,0);
```

```
      analogWrite(G_Pose,210);
      analogWrite(B_Pose,0);
   Arriba();
  }
  else if  (gesture == ARM_DOWN ) {  //Amarillo
      analogWrite(R_Pose,250);
      analogWrite(G_Pose,100);
      analogWrite(B_Pose,0);
    abrirMAno();
  }
  else if (gesture == ARM_LEFT ) {  //Naranja
analogWrite(R_Pose,250);
      analogWrite(G_Pose,30);
      analogWrite(B_Pose,0);
    Izquierda();
}
  else if (gesture == ARM_RIGHT ) {  //Rojo
      analogWrite(R_Pose,250);
      analogWrite(G_Pose,0);
      analogWrite(B_Pose,0);
    Derecha();
  }
  else if (gesture == ARM_CIRCLE_CW ) {  //Turque
     // analogWrite(R_Pose,0);
     // analogWrite(G_Pose,150);
```

```
    // analogWrite(B_Pose,150);
  }
  else if (gesture == ARM_CIRCLE_CCW  ) { //Morado
    //  analogWrite(R_Pose,200);
    // analogWrite(G_Pose,0);
    // analogWrite(B_Pose,160);
  }
  else if (gesture == ARM_ROTATE_CW  ) {  //Azul
MovAgarre = !MovAgarre;
   if (MovAgarre){
     analogWrite(R_Pose,0);    //turque
analogWrite(G_Pose,150);
     analogWrite(B_Pose,150);
     }
   else{
     analogWrite(R_Pose,0);
     analogWrite(G_Pose,0);
     analogWrite(B_Pose,250);
     }
  }
  else if (gesture == ARM_ROTATE_CCW ) {  //Rosa
     analogWrite(R_Pose,250);
     analogWrite(G_Pose,0);
     analogWrite(B_Pose,30);
   RotarR();
```

```
  }
  else{
  }
 AlertaBat();
}
///_____________________________________________
void Arriba( )
  {
    if (MovAgarre)
      {saludar();}
    else
      {apuntar();}
  }
////////////////////////////////
void RotarR()
  {
   if (MovAgarre)
{Raton();}
    else
      {Pelota();}
  }
////////////////////////////////
void   Derecha()
{
   if (MovAgarre&&MovRaton)
```

```
      {ClickDerecho();}
    else if (MovAgarre && !MovRaton)
      {Taza();}
    else
      {DosDedos();}
  }
////////////////////////////////
void   Izquierda()
  {
   if (MovAgarre && MovRaton)
      {ClickIzquierdo();}
    else if (MovAgarre && !MovRaton)
      {botella();}
else
      {TresDedos();}
  }
////////////////////////////////
 void Taza()
 {
  cerrar1=false; cerrar2=false; cerrar3=false; cerrar4=false; cerrar5=true;
if(d2 && d3 && d4){}
      else{abrir(anguloAbrir);}
      if(!d1) { Dedo1(anguloAbrir1); }
      if(!d5){}
      else { cerrar(anguloCerrar);}
```

```
    d1=true; d2=true; d3=true; d4=true; d5= false;
delay(3000);
    cerrar1=false; cerrar2=true; cerrar3=true; cerrar4=true; cerrar5=false;
    cerrar(80); d4=false;
    cerrar(120);
    d2=false; d3=false;
    MovRaton = false;
  }
 ////////////////////////////
 void ClickDerecho()
   {
   cerrar1=false; cerrar2=false; cerrar3=true; cerrar4=false; cerrar5=false;
   //delay(250);
   cerrar(100);  d3=false;
   cerrar1=true; cerrar2=true; cerrar3=false; cerrar4=true; cerrar5=true;
   //delay(1000);
   abrir(anguloAbrir);  d3=true;
  }
 ////////////////////////////
 void ClickIzquierdo()
   {
   cerrar1=false; cerrar2=true; cerrar3=false; cerrar4=false; cerrar5=false;
   //delay(250);
   cerrar(130);  d2=false;
   cerrar1=true; cerrar2=false; cerrar3=true; cerrar4=true; cerrar5=true;
```

```
//delay(1000);
   abrir(anguloAbrir);  d2=true;
  }
////////////////////////////////
void Raton()
{
  abrirMAno();
  cerrar1=false; cerrar2=false; cerrar3=false; cerrar4=false; cerrar5=true;
     cerrar(150);  d5=false;
  cerrar4=true;
     cerrar(100);  d4=false;
  cerrar1=true;
     Dedo1(120);  d1=false;
  MovRaton = true;
  }
////////////////////////////////
void saludar()
{
  abrirMAno();
    delay(1000); //////
  cerrar1=false; cerrar2=false; cerrar3=true; cerrar4=true; cerrar5=true;
     cerrar(90);  d3=false; d4=false;
   cerrar(115);  d5=false;
  delay(4000);     ///// tiempo que demora el saludo
  abrirMAno();
```

```
  MovRaton = false;
  }
////////////////////////////////
void cerrarMano( )
{
  cerrar1=true; cerrar2=true; cerrar3=true; cerrar4=true; cerrar5=true;
      if(d1 && cerrar1) { Dedo1(anguloCerrar1); }
      if(!d2 && !d3 && !d4 && !d5){}
      else { cerrar(anguloCerrar);}
   d1=false; d2=false; d3=false; d4=false; d5= false;
   MovRaton = false;
}
////////////////////////////////
void abrirMAno()
{
  cerrar1=false; cerrar2=false; cerrar3=false; cerrar4=false; cerrar5=false;
if(d2 && d3 && d4 && d5){}
      else { abrir(anguloAbrir);}
if(!d1 && !cerrar1) { Dedo1(anguloAbrir1); }
d1=true; d2=true; d3=true; d4=true; d5= true;
MovRaton = false;
}
////////////////////////////////
void apuntar()
{
```

```
  cerrar1=true; cerrar2=false; cerrar3=true; cerrar4=true; cerrar5=true;
      if(!d2){abrir(anguloAbrir);}
      if(d1) { Dedo1(anguloCerrar1); }
      if(!d3 && !d4 && !d5){}
      else { cerrar(anguloCerrar);}
d1=false; d2=true; d3=false; d4=false; d5= false;
MovRaton = false;
}
////////////////////////////////
void DosDedos()
{
  cerrar1=true; cerrar2=false; cerrar3=false; cerrar4=true; cerrar5=true;
if(d2 && d3){}
      else{abrir(anguloAbrir);}
if(d1) { Dedo1(anguloCerrar1); }
      if(!d4 && !d5){}
      else { cerrar(anguloCerrar);}
    d1=false; d2=true; d3=true; d4=false; d5= false;
    MovRaton = false;
}
////////////////////////////////
void TresDedos()
{
  cerrar1=true; cerrar2=true; cerrar3=false; cerrar4=false; cerrar5=false;
if(d3 && d3 && d5){}
```

```
      else{abrir(anguloAbrir);}
if(d1) { Dedo1(anguloCerrar1); }
      if(!d2){}
      else { cerrar(anguloCerrar);}
    d1=false; d2=false; d3=true; d4=true; d5= true;
    MovRaton = false;
}
////////////////////////////////////////////////// ----------> pelota
void Pelota()
    {
     cerrar1=false; cerrar2=false; cerrar3=false; cerrar4=true; cerrar5=true;
if(d2 && d3){}
      else{abrir(anguloAbrir);}
      if(!d1) { Dedo1(anguloAbrir1); }
      if(!d4 && !d5){}
else { cerrar(anguloCerrar);}
     d4=false; d5= false;
delay(3000);
     d1=true; d2=true; d3=true;
cerrar2=true; cerrar3=true;  ///
     ServoFSR();
     d2=false; d3=false;
     MovRaton = false;
    }
```

```
////////////////////////////////////////////////// ----------> BOTELLA
void botella()
   {
      cerrar1=true; cerrar2=true; cerrar3=true; cerrar4=true; cerrar5=true;
    ServoFSR();
    d1=false; d2=false; d3=false; d4=false; d5= false;
    MovRaton = false;
   }
//////////////////////////////////      ---------->Control grado a grado aumento FSR
void ServoFSR()
 {
  int var = 0;
   while(var < 180){
Ungrado(var);
      FsrLectura1 = analogRead(FSR1);
var++;
      if (FsrLectura1 > fuerzaMAX)
       { break;}
   }
}
//////////////////////////////
void Ungrado(int angulo)
 {
   for(int i=0; i<2; i++)
{
```

```
      pausa = angulo*2000.0/180.0 + 500;
          if (d2 && cerrar2){ digitalWrite(Servo2, HIGH); }
          if (d3 && cerrar3){ digitalWrite(Servo3, HIGH); }
          if (d4 && cerrar4){ digitalWrite(Servo4, HIGH); }
          if (d5 && cerrar5){ digitalWrite(Servo5, HIGH); }
      delayMicroseconds(pausa);
          if (d2 && cerrar2){ digitalWrite(Servo2, LOW); }
          if (d3 && cerrar3){ digitalWrite(Servo3, LOW); }
          if (d4 && cerrar4){ digitalWrite(Servo4, LOW); }
if (d5 && cerrar5){ digitalWrite(Servo5, LOW); }
      delayMicroseconds(25000-pausa);
    }


  }
//////////////////////////////
void cerrar(int angulo)
{
    for(int i=0; i<50; i++)
    {
      pinMode(Servo1,OUTPUT);
      pausa = angulo*2000.0/180.0 + 500;
          if (d2 && cerrar2){ digitalWrite(Servo2, HIGH); }
          if (d3 && cerrar3){ digitalWrite(Servo3, HIGH); }
          if (d4 && cerrar4){ digitalWrite(Servo4, HIGH); }
          if (d5 && cerrar5){ digitalWrite(Servo5, HIGH); }
```

```
    delayMicroseconds(pausa);
        if (d2 && cerrar2){ digitalWrite(Servo2, LOW); }
        if (d3 && cerrar3){ digitalWrite(Servo3, LOW); }
        if (d4 && cerrar4){ digitalWrite(Servo4, LOW); }
        if (d5 && cerrar5){ digitalWrite(Servo5, LOW); }
    delayMicroseconds(25000-pausa);
   }
  }
//////////////////////////////
void abrir(int angulo)
  {
   for(int i=0; i<40; i++)
   {
    pinMode(Servo1,OUTPUT);
    pausa = angulo*2000.0/180.0 + 500;
        if (!d2 && !cerrar2){ digitalWrite(Servo2, HIGH); }
        if (!d3 && !cerrar3){ digitalWrite(Servo3, HIGH); }
        if (!d4 && !cerrar4){ digitalWrite(Servo4, HIGH); }
        if (!d5 && !cerrar5){ digitalWrite(Servo5, HIGH); }
    delayMicroseconds(pausa);
        if (!d2 && !cerrar2){ digitalWrite(Servo2, LOW); }
        if (!d3 && !cerrar3){ digitalWrite(Servo3, LOW); }
        if (!d4 && !cerrar4){ digitalWrite(Servo4, LOW); }
        if (!d5 && !cerrar5){ digitalWrite(Servo5, LOW); }
    delayMicroseconds(25000-pausa);
```

```
    }
  }
  ////////////////////////////////
void Dedo1(int angulo)
  {
    for(int i=0; i<30; i++)
    {
      pinMode(Servo1,OUTPUT);
      pausa = angulo*2000.0/180.0 + 500;
      digitalWrite(Servo1, HIGH);
      delayMicroseconds(pausa);
      digitalWrite(Servo1, LOW);
      delayMicroseconds(25000-pausa);
}
  }
////////////////////////////////////////////////
void AlertaBat(){

   ValCero = analogRead(LecturaCero);
   ValUno = analogRead(LecturaUno);
   ValDos = analogRead(LecturaDos);
   ValTres = analogRead(LecturaTres);
  // Obtenemos el voltaje

   bateriaUno =(0.0049 * ValCero);
```

```
   bateriaDos = (0.0098 * ValUno) - bateriaUno;
   bateriaTres = (0.0049 * ValDos);
   bateriaCuatro = (0.0098 * ValTres) - bateriaTres;
 //Serial.println(bateriaUno);
 //Serial.println(bateriaDos);
 //Serial.println(bateriaTres);
 //Serial.println(bateriaCuatro);

  if ((bateriaUno<BateriaBaja)||( bateriaDos< BateriaBaja)||( bateriaTres<
BateriaBaja)||(bateriaCuatro < BateriaBaja)){
    digitalWrite(Alerta, HIGH);
    }
   else{
    digitalWrite(Alerta, LOW);
}
 }
////////////////////////////////////////////
//-----------------------------------------------------------------
void pin_setup() {

pinMode(Servo1,OUTPUT);
pinMode(Servo2,OUTPUT);
pinMode(Servo3,OUTPUT);
pinMode(Servo4,OUTPUT);
pinMode(Servo5,OUTPUT);
```

```
pinMode(R_Pose, OUTPUT);
pinMode(G_Pose, OUTPUT);
pinMode(B_Pose, OUTPUT);
pinMode(Led_Lectura, OUTPUT);
pinMode(Alerta, OUTPUT);
}
//------------------------------------------------------------------
void setup() {
pin_setup();  // LLama a definir pines de salida
//initialize both serial connections
  Serial.begin(115200);
  bridgeSerial.begin(115200);
  //wait until MyoBridge has found Myo and is connected. Make sure Myo is not connected to anything else and not in standby!
  Serial.println(F("Searching for Myo..."));
  //initiate the connection with the status callback function
  bridge.begin(printConnectionStatus);
  Serial.println(F("connected!"));
  MyoIMUGestureController::begin(bridge, updateControls, updateLockOutput);
}
//------------------------------------------------------------------
void loop() {
   // Actualizar la conexión a MyoBridge
  bridge.update();
}
```