Grado en Ingeniería de
Tecnologías de Telecomunicación

Trabajo Fin de Grado

"Desarrollo de prototipo de prótesis biónica 3D mioeléctrica con conexión al sistema operativo Android"

Pablo Bueno Torrijos
Cuenca, Septiembre de 2017

Campus Universitario | 16071 Cuenca | Telf.: (+34) 969 179 100 (+34) 902 204 100 | Fax.: (+34) 969 179 119
politecnicacuenca.uclm.es | politecnica.cu@uclm.es

“Los límites solo se encuentran en tu propia mente”

AGRADECIMIENTOS

Principalmente, agradecer a mi familia y mi pareja, el apoyo y la confianza que han depositado en mí.

En segundo lugar quiero dar las gracias a todos mis amigos que han estado a pie del cañón conmigo en todo momento. Además, a aquellas personas que estén leyendo o hayan leído este trabajo.

Especialmente quiero agradecer a mi director de trabajo fin de grado, mi tutor, César Sánchez, por su dedicación, apoyo y ayudarme a tener claras las ideas sobre este trabajo.

Por último, y no menos importante, resaltar también a mis compañeros de carrera, por todas las vivencias y enseñanzas adquiridas juntos, al igual que su apoyo.

RESUMEN

El objetivo inicial del proyecto es el diseño de prótesis de una mano a partir de tecnologías accesibles y de menor coste de fabricación e implantación. Esta prótesis se adquiere mediante la impresión 3D, una aplicación Arduino y una aplicación Android. Todo ello sin ánimo de lucro para uso personal y docente.

El funcionamiento del sistema se basa en primer lugar en la obtención de señales mioeléctricas para poder establecer el movimiento de la prótesis. Tras esto podemos comunicarnos a través del teléfono móvil con Arduino, el cual interpretará los comandos enviados y recibidos. Todo ello gracias a la comunicación mediante un socket entre el teléfono móvil y el módulo bluetooth que se comunica con Arduino a través de un puerto serie.

El electromiograma es una prueba para estudiar el funcionamiento del sistema nervioso que consiste en registrar mediante electrodos las corrientes eléctricas que se forman en los nervios y músculos al producirse contracciones.

Los equipos de registro de electromiografía que se encuentran en entornos clínicos tienen un coste muy elevado, para esto se propone un modelo en el que se emplean materiales de bajo costo y accesibles.

Para la obtención de estas señales se ha empleado una **placa E-HEALTH**, que se trata de un circuito electrónico que amplifica y filtra la señal. A continuación, tenemos la **placa Arduino mega 2560** que por medio de la aplicación integrada recoge la señal ya tratada para poder controlar el sentido de giro de cada uno de los motores que componen la prótesis.

Cuenta con una pantalla LCD, un giroscopio para realizar el movimiento de muñeca y un módulo bluetooth con el que se establece un sistema para la comunicación inalámbrica entre la aplicación Arduino y una aplicación Android.

Esta aplicación Android consigue un monitoreo de los pulsos eléctricos obtenidos de los músculos para su correcta calibración y su correcto posicionamiento de los sensores. También se puede dotar de movimiento a los motores de la prótesis.

ABSTRACT

The main aim of this project is the design of a hand prosthesis on the basis of accessible technologies of fewer cost of manufacturing and implementation. This prosthesis is developed through 3D printing, an Arduino application and an Android application. All this is done with non-profit purposes but for personal and teaching use.

The system working is based, firstly, on the collection of myoelectric signals in order to establish the movement of the prosthesis. After that, it is possible to communicate through a mobile phone using Arduino, which will interpret the sent and received commands. This will be done thanks to the communication through a socket between the mobile phone and the bluetooth module which communicates with Arduino though a serial port.

The electromyogram is a test used to study the functioning of the nervous system which consists of registering through electrodes the electrical currents formed in nerves and muscles when contractions occur. The electromyogram registering equipment usually found in clinical surroundings is very expensive, so to solve this drawback it is proposed a model in which materials used are affordable and with a fewer cost.

For the obtention of this signals, it is used an **E-HEALTH board** which consists of an electrical circuit that amplifies and filters the signal. After this, the **Arduino mega 2560 board** collects the signal though an integrated application in order to control the sense of rotation of each of the engines that comprise the prosthesis.

It counts with a LCD screen, a gyroscope to make the hand movement and a bluetooth module to establish a wireless system of communication between the Arduino application and the Android application. This Android application obtains a monitoring of the electric pulse got from muscles for the proper calibration and the proper positioning of the sensors. Movement can also be endowed in the prosthesis engines.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS 15
ACRÓNIMOS 17
I-MEMORIA 19
CAPÍTULO 1 21
INTRODUCIÓN 21
1.1. MOTIVACIÓN 21
1.2. OBJETIVOS 21
CAPÍTULO 2 22
IMPRESIÓN 3D 22
2.1. ¿QUÉ ES LA IMPRESIÓN 3D? 22
2.2. HISTORIA DE LA IMPRESIÓN 3D 23
2.3. SOFTWARE IMPRESIÓN 3D – CURA 24
2.4. MATERIALES IMPRESIÓN 3D 28
2.5. PROBLEMAS A LA HORA DE LA IMPRESIÓN 3D 30
2.6. COMUNIDADES DE IMPRESORES 3D 34
2.7. DISEÑO DE LA PRÓTESIS 35
CAPÍTULO 3 40
SEÑALES MIOELÉCTRICAS 40
3.1. TEJIDO MUSCULAR 40
3.2. ELECTROMIOGRAMA 41
3.3. OBTENCIÓN DEL EMG CON E-HEALTH 43
CAPÍTULO 4 49
ARDUINO 49
3.1. ¿QUÉ ES ARDUINO? 49
3.2. HARDWARE Y SOFTWARE 50
3.3. MODELOS DE ARDUINO 52
3.4. SENSORES UTILIZADOS 55
3.5. APLICACIÓN ARDUINO 60
CAPÍTULO 5 66
ANDROID 66
5.1. ¿QUÉ ES ANDROID? 66
5.2. ARQUITECTURA 68
5.3. DESARROLLO APLICACIÓN ANDROID 70
CAPÍTULO 6 80
DISEÑO FINAL 80

6.1. ENSAMBLADO DEL PROTOTIPO 80
6.2. ESQUEMA DE FUNCIONAMIENTO 87
CAPÍTULO 7 88
CONCLUSIÓN Y FUTURAS LÍNEAS DE ACTUACIÓN 88
7.1. CONCLUSIONES 88
7.2. FUTURAS LÍNEAS DE ACTUACIÓN 89
REFERENCIAS BIBLIOGRÁFICAS 90
II- PLANOS 93
1 PLANOS 95
2 ESQUEMAS ELÉCTRICOS 100
III- PLIEGO DE CONDICIONES 103
1. CARACTERÍSTICAS TÉCNICAS DE LOS EQUIPOS UTILIZADOS 105
REQUISITOS HARDWARE 105
REQUISITOS SOFTWARE 108
IV- PRESUPUESTO 110
PRESUPUESTO 112

ÍNDICE DE FIGURAS

Figura 1 - Impresión 3D. 22
Figura 2 - Fused Deposition Modelling (FDM). 23
Figura 3 - Interfaz de Cura. 25
Figura 4 - Configuración Cura. 26
Figura 5 - Materiales Impresión 3D. 28
Figura 6 - Levantamiento de la base (Warping). 30
Figura 7 - Almhadillado (Pillowing). 31
Figura 8 - Encordado (Strining). 32
Figura 9 - Voladizos (Overhang). 33
Figura 10 - Subextrusión (Under extursion). 34
Figura 11 - Huesos de la mano. 36
Figura 12 - Diseño palma de la mano. 36
Figura 13 - Movimiento dedo pulgar. Imagen detalle sobre la prótesis realizada. 37
Figura 14 - Movimiendo dedos índice y corazón. Imagen detalle sobre la prótesis realizada. 38
Figura 15 - Movimiento dedos anular y meñique. Imagen detalle sobre la prótesis realizada. 38
Figura 16 - Grado de movimiento de los metacarpianos en la palma de la mano. 39
Figura 17 - Giro muñeca. Imagen detalle sobre la prótesis realizada. 39
Figura 18 - Anatomía músculos. 41
Figura 19 - EMG intramuscular. 42
Figura 20 - EMG superficial. 42
Figura 21 - E-HEALTH. 44
Figura 22 - Placa e-Health. 45
Figura 23 - Esquema EMG e-Health. 46
Figura 24 - Posiciones electrodos. 47
Figura 25 - Primer Arduino. 49
Figura 26 - IDE Software Arduino. 51
Figura 27 - Arduino Uno. 52
Figura 28 - Arduino Mega 2560. 53
Figura 29 - Arduino Nano. 54
Figura 30 - Electrodos EMG. 55
Figura 31 - Módulo bluetooth HC-06. 56
Figura 32 - LCD HD44780. 57
Figura 33 - MPU 6050. 58
Figura 34 - Servo Futaba S3003. 59
Figura 35 - Máquina de estados. 62
Figura 36 - Android. 66
Figura 37 - Arquitectura Android. 68
Figura 38 - Graph View. 71
Figura 39 - Ciclo de vida de una actividad. 72
Figura 40 - Fragment control aplicación Android. 77
Figura 41 - Fragent giroscopio aplicación Android. 78
Figura 42 - Flechas posición de fragment giroscopio 78

Figura 44 - Ensamblado palma. 80
Figura 45 - Errores ensamblado palma de la mano. 81
Figura 46 - Ensamblado dedos. 82
Figura 47 - Ensamblado dedo pulgar. 82
Figura 48 - Ensamblado muñeca. 83
Figura 49 - Ensamblado servos. 84
Figura 50 - Ensamblaje, orificios tensores palma mano y muñeca. 85
Figura 51 - Ensamblaje mano, tensores. 85
Figura 52 - Ensamblado final. 86

ACRÓNIMOS

Símbolo	Nombre completo
EMG	Electromiograma
ECG	Electrocardiograma
3D	Three Dimension
3DP	Three Dimensional Printing
FDM	Fused Deposition Modelling
STL	Standart Triangle Language
ABS	Acrilonitrilo Butadieno Estireno
PLA	Ácido poliactico
HIPS	Poliestireno de alto impacto
PET	Tereftalato de polietireno
GND	Ground
DC	Direct Current
V	Volt
SENIAM	Surface EMG for the Non-Invasive Assessmen of Muscles
USB	Universal Serial Bus
IDE	Integrated Development
PWM	Modulación por anchura de pulso
EEPRON	Electrically Erasable Programmable Read-Only Memory
SRAM	Static Random Access Memory
I2C	Iner-Integrated Circuit
SCL	Serial Clock
SDA	Serial Data
XML	eXtensible Markup Language
SQL	Structured Query Language
OpenGL	Open Graphics Library
FSR	Fisio Runners

Cuadro 1 - Tabla de Acrónimos

PARTE 1

I- MEMORIA

CAPÍTULO 1

INTRODUCIÓN

1.1. MOTIVACIÓN

Debido al elevado coste de las prótesis, este proyecto propone generar un modelo más económico y mostrar que con un hardware accesible se pueden resolver problemas de la vida cotidiana. Este prototipo puede ser un ejemplo para su desarrollo y perfección, ya que la impresión 3D está en plena expansión y la medicina es uno de los campos donde hemos visto su llegada.

Las prótesis existentes, aunque avanzan en ellas a buen ritmo, son dispositivos increíblemente costosos, no siendo un capricho para las personas que padecen algún tipo de malformación o amputación de los miembros superiores. Es prácticamente imposible usarlas en niños, ya que éstos se van desarrollando con la edad y ello hace que tengan que cambiar de prótesis a menudo.

Es por esto que surgió mi motivación, por crear dicho prototipo de esta prótesis basado en Arduino, donde se ha intentado facilitar las limitaciones de las personas discapacitadas de manera sencilla y atractiva.

1.2. OBJETIVOS

El objetivo de este trabajo es desarrollar un prototipo de prótesis de una mano mediante la impresión 3D, con la placa e-Health de la mano de Arduino y sus correspondientes sensores y servos.

La finalidad es realizar un electromiograma (EMG) superficial continuo a tiempo real y poder trasladar estos pulsos eléctricos al movimiento mecánico de la mano. Para ello se empleará una placa Arduino, que actuará como tarjeta de adquisición de datos y la placa e-Health, la cual se encargará de amplificar y filtrar la señal obtenida.

CAPÍTULO 2

IMPRESIÓN 3D

2.1. ¿QUÉ ES LA IMPRESIÓN 3D?

Es la técnica de realizar réplicas de diseños en 3D, creando piezas o maquetas volumétricas a partir de un diseño hecho por ordenador, descargado de internet o recogido a partir de un escáner 3D. Surgen con la idea de convertir archivos de 2D en prototipos reales o 3D [1].

La impresión 3D se refiere a los procesos en los que secuencialmente se acumula material en una cama o plataforma por diferentes métodos de fabricación, tales como:

- **Sinterización láser**, donde un suministrador va depositando finas capas de polvo de diferentes metales (acero, aluminio, titanio...) y a continuación un láser funde cada capa con la anterior.

- **Estereolitografía**, donde una resina fotosensible es curada con haces de luz ultravioleta, solidificándola.

- **Compactación**, con una masa de polvo que se compacta por estratos.

- **Adición**, o de inyección de polímeros, en las que el propio material se añade por capas.

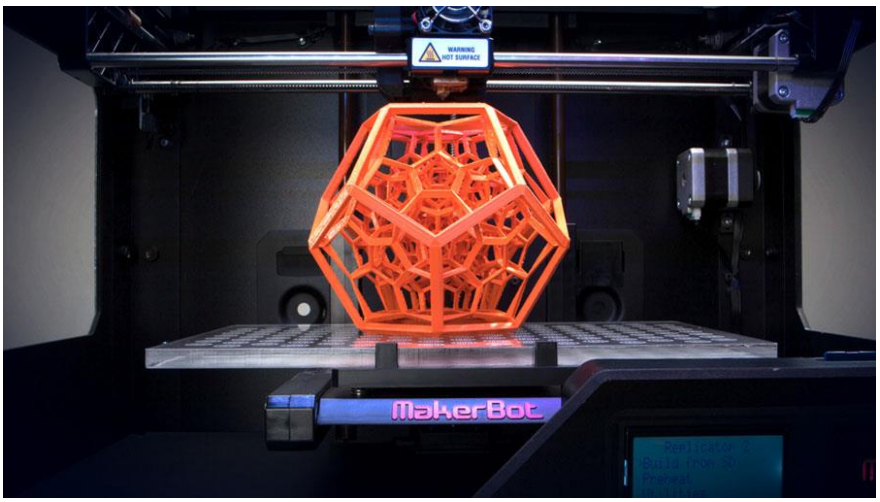


Figura 1 - Impresión 3D.

2.2. HISTORIA DE LA IMPRESIÓN 3D

Las impresoras 3D tienen un largo recorrido en la historia, pero su inicio se centra sobre el año 1970. Como en los inicios de cualquier tecnología, las máquinas empezaron teniendo un gran volumen y muchas limitaciones, con el paso delos años ha habido una gran optimización, haciéndolas más reducidas, con menor coste y con mayor fiabilidad.

Concretamente la primera impresora 3D aparece en 1984 con la invención de "stereolithography" [1], una tecnología que consiste en crear modelos 3D, donde los usuarios podían realizar diseños antes de la fabricación.

En 1987 la invención del Sinterizado Laser Selectivo dio un cambio de vuelta, lo que consistía en fundir un polvo metálico sobre el sustrato creando un objeto [1].

Entre los años 1987 y 1992 aparece ya la tecnología Fused Deposition Modelling (FDM) [1], cuyo creador fue Scott Crump (1988), que intentó realzar un diseño comerciable. Es la usada hoy en día y consiste en utilizar cabezales o boquillas a altas temperaturas y un material fundente para crear los objetos tridimensionales.

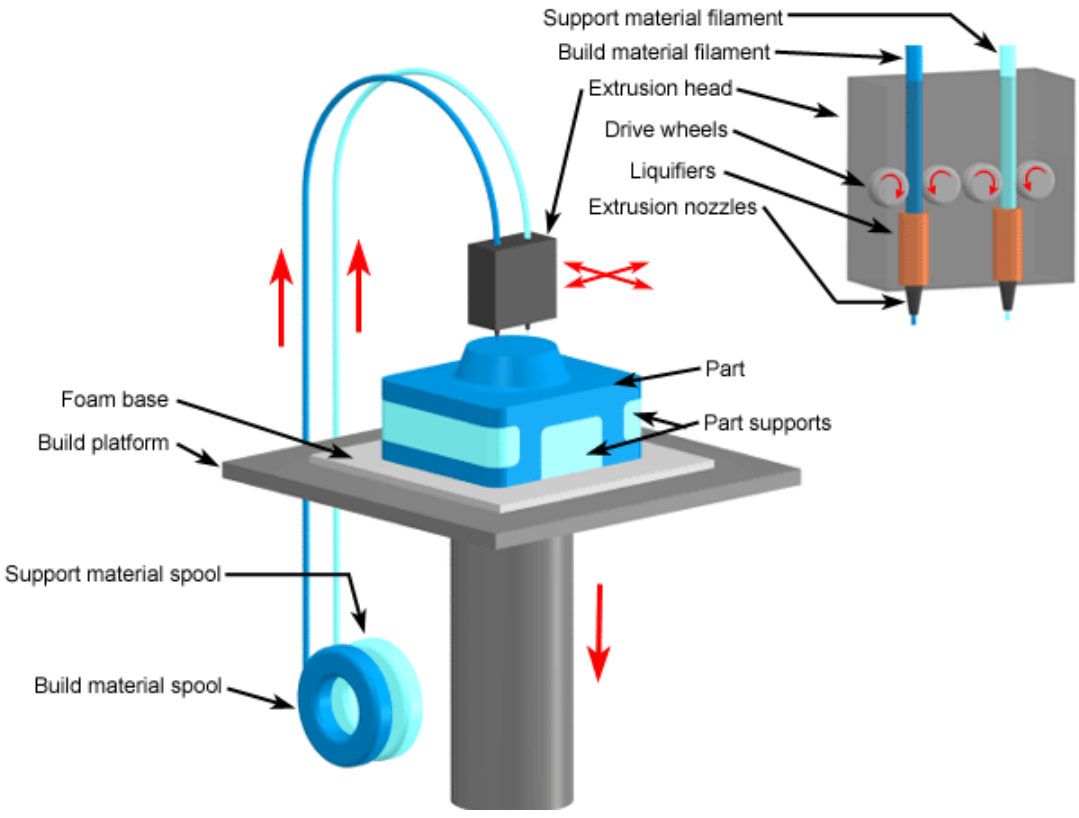


Figura 2 - Fused Deposition Modelling (FDM).

Posteriormente, se patentará la “impresión tridimensional”, Three Dimensional Printing (3DP). Esta patente fue desarrollada por “Z Corp” en el Instituto Tecnológico de Massachusetts, hoy en día parte de “3D Systems” [1].

Entre 1992 y 1996 esta empresa desarrollaría una primera impresora 3D (Z402) basándose en el modelo estándar de hoy en día, el de inyección.

En 1996 se vende la primera impresora auto-replicante basada en el proyecto RepRap, destacando este proyecto, el cual fue el impulsor de las comunidades de impresores y estableció unas normas de la impresión 3D, haciendo que se desarrollaran las licencias “*creative commons”* y código abierto.

Las ideas de este proyecto se pueden encontrar en el documento de los creadores, D.Holland, G. O’Donnell y G. Bennett, los cuales publicaron el libro: “Open Design and the Reprap Project” [2].

2.3. SOFTWARE IMPRESIÓN 3D – CURA

Cura es un software que perteneciente a Ultimaker, nos va a permitir convertir los archivos en formato Standard Triangle Language (STL) que contienen nuestro diseño 3D en piezas físicas en un solo entorno de trabajo. Este software lo podemos obtener en la página de Ultimaker [3]. También cuenta con gran cantidad de información y manuales de uso.

Es un programa "*Open Source*" [4], desarrollado por Ultimaker para la comunicación con la impresora y el laminado de objetos 3D, por lo que desde este programa se pueden realizar todos los pasos necesarios para pasar de un modelo 3D a un objeto real.

Originalmente, Cura se llamaba SkeinPyPy, cuyo nombre proviene de la combinación de Skeinforge con PyPy. Skeinforge es una sofware que realiza el corte de las piezas 3D en capas (Slicer) generando el código para que nuestras impresoras lo comprendan.

Cura ofrece opciones de configuración simples y una interfaz gráfica más intuitiva y sencilla a la hora de generar el G-Code de impresión.

El G-Code, es un lenguaje de programación con el que se expresan las órdenes que debe ejecutar la impresora 3D. En realidad el G-Code, o código o lenguaje G, es un estándar muy difundido para programar máquinas de control numérico, como tornos o fresadoras, no sólo impresoras 3D.

Como desventajas frente a *Skeinforge* está que al tener menos opciones configurables pierdes algunas propiedades de *Skeinforge*. Como ventajas, la pantalla

que te muestra las piezas en 3D y el resultado del laminado te da una idea bastante clara de lo que va a producir y de cómo lo va a hacer.

Finalmente, es interesante comentar que Cura tiene como objetivo no sólo actuar como programa de laminado sino también como anfitrión (*host*) de la impresión, por lo que su objetivo es ser un *todo en uno*, un programa que además de generar el *g-code*, lo envíe a la impresora.

A la hora de configurar nuestro programa hay que tener en cuenta que cada pieza tendrá un tipo de configuración que se verá afectada por los siguientes parámetros:

- *Material utilizado*: Hay diferentes tipos de materiales que tienen distinta composición, densidad, temperatura de extrusión, etc.

- *Tipo de impresora*: En el mercado hay diferentes modelos, por lo que variarán las dimensiones de impresión.

- *Estructura del modelo*: Cada modelo tendrá una estructura geométrica que podrá imprimirse en función de las limitaciones de cada impresora.

Este programa muestra un área de impresión que es una representación tridimensional del volumen de impresión de la impresora. Este es el espacio con el que contamos para imprimir y no podremos exceder sus límites, ya que quedarían fuera de los límites de impresión reales de nuestra máquina.

En esta área nos encontraremos las herramientas necesarias para cargar las figuras que queramos imprimir, modificarlas y visualizar las diferentes capas.

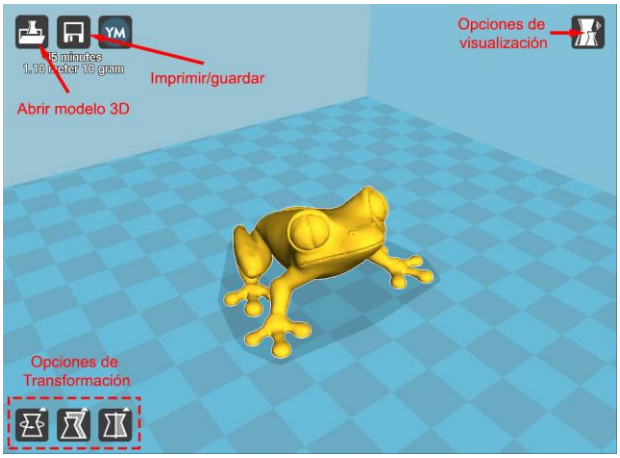


Figura 3 - Interfaz de Cura.

Las opciones de transformación nos permiten rotar, escalar y función espejo de la figura inicial, se puede espejar la figura en cualquiera de los 3 ejes.

La configuración de laminado es la más importante. Aquí es donde se genera el G-Code determinado para que nuestra impresora pueda realizar la impresión correctamente.

Para configurar la forma en la que nuestra maquina va a realizar el laminado, tenemos una serie de pestañas en la parte izquierda de la pantalla. En estas pestañas vamos a encontrar las opciones más comunes a la hora de laminar. En la siguiente figura se mostrará el tipo de configuración.

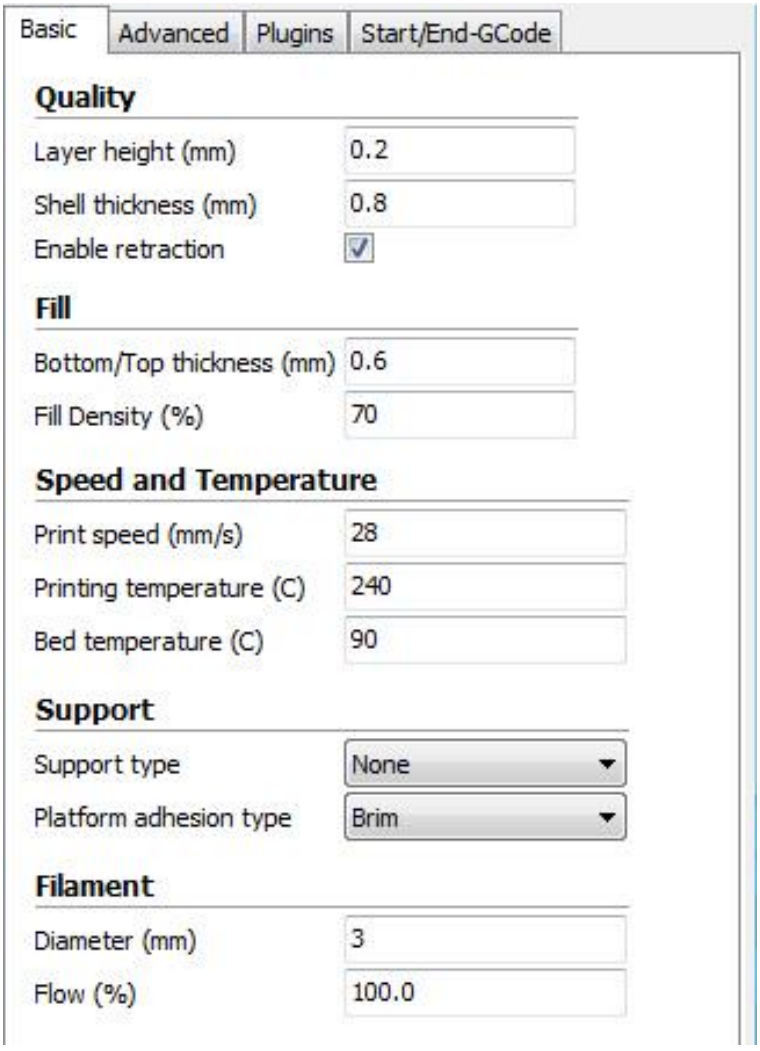


Figura 4 - Configuración Cura.

Quality (calidad)

Layer height (atura de capa): Este parámetro indica la altura de capa a la que se va a realizar la impresión.

Shell thickness (grosor del borde): Determina la anchura del borde del objeto.

Enable retraction (habilitar retracción): Esta opción hace que en los desplazamientos el extrusor retraiga un poco el plástico para que no gotee, evitando así pequeños defectos en la impresión.

Fill (relleno)

Bottom/Top thickness: Con este parámetro indicaremos qué grosor tendrán las capas superior e inferior.

Fill Density (densidad de relleno): Este valor indica el relleno que va a tener la figura.

Speed and Temperature (velocidad y temperatura)

Print speed: En este parámetro vamos a fijar la velocidad de impresión.

Printing Temperature: Fija la temperatura a la que se va a imprimir.

Bed Temperature: Fija la temperatura de la cama caliente.

Support (soporte)

Para muchas de las impresiones debemos de usar elementos que aseguren una correcta impresión, como pueden ser los elementos de soporte o de mejora de la adherencia.

Support type (tipo de soporte): Esta opción creará soportes donde sea necesario.

Platform adhesion type (plataforma de adhesión): Con esta opción podemos crear una plataforma en la base que mejore la adhesión de la pieza.

Filament (filamento)

Diameter (Diámetro de filamento): Establece el diámetro del filamento que estemos usando.

Flow (Multiplicador del flujo de filamento): Este parámetro modifica la cantidad de filamento que extruye la impresora.

También existe una configuración avanzada donde nos permite configurar los valores de nuestra impresora que usaremos para una impresión más personalizada y para dar un mejor acabado.

2.4. MATERIALES IMPRESIÓN 3D

Figura 5 - Materiales Impresión 3D.

Lo más importante en estos materiales, además de su composición, es el diámetro. Esto afectará directamente a la calibración de la propia impresora, dando lugar a los errores de impresión 3D, deformaciones y otros errores como atascos del cabezal, parón en mitad de la impresión y multitud de problemas en la impresión. Por tanto, se realiza una serie de mediciones a lo largo de las muestras de filamento, obteniendo la media, desviación típica y tolerancias finales.

Además del filamento en sí, también es importante la bobina en la que esté enrollado y como lo esté, así como el empaquetado y sellado de los mismos.

Los materiales más comunes [5] que se encuentran en el mercado son:

ABS

"Acrilonitrilo Butadieno Estireno" es uno de los termoplásticos más usados en la impresión 3D. No es biodegradable, pero es muy tenaz, duro y rígido, con resistencia química y la abrasión, pero que sufre con la exposición a rayos UV. Es soluble en acetona y su densidad es de 1,05 g/cm^3. Requiere una temperatura de cabezal de unos 240ºC y de bandeja de 110ºC. Un buen ejemplo son las piezas de LEGO, fabricadas con ABS.

PLA

"Ácido poliláctico" es otro de los filamentos más comunes en la impresión 3D. Es biodegradable y normalmente se obtiene de almidón de maíz. La textura de las piezas no queda tan suave como con el ABS, pero sí más brillantes. Su densidad es de entre 1,2 y 1,4 g/cm^3. La temperatura necesaria para su impresión es de unos 210ºC con la cama a unos 60ºC.

HIPS

"Poliestireno de alto impacto" es un material muy parecido al ABS y que requiere los mismos perfiles de temperaturas. Suele usarse en combinación con el ABS para hacer piezas con espacios huecos. Al igual que el ABS soporta mal la luz UV y su densidad es de 1,04 g/cm^3.

PET

"Tereftalato de polietileno" es uno de los materiales más usados para las botellas y otro tipo de envases. Su principal propiedad es su capacidad de cristalización, generando piezas transparentes con efectos sorprendentes. Es muy fuerte y resistente a los impactos. Su densidad cristalina es de 1,45 g/cm^3.

Ninjaflex

Se trata de un revolucionario elastómero termoplástico (TPE) que permite crear piezas con una flexibilidad sorprendente. Las piezas resultantes pueden deformarse ampliamente. La temperatura es muy parecida a la del PLA, con el cabezal a 215ºC y la bandeja a 40ºC.

Nylon

Este material es quizás uno de los materiales más complejos para la impresión 3D. Su principal problema es la falta de adhesión de la pieza a la bandeja, que causa muchos fallos además de un warping (deformación) muy difícil de controlar. Además, suele coger fácilmente humedad, por lo que previamente a la impresión 3D deberemos secarlo. A cambio de todas estas dificultades, el nylon es un material muy resistente, poco viscoso, muy resistente a la temperatura.

2.5. PROBLEMAS A LA HORA DE LA IMPRESIÓN 3D

A la hora de imprimir estos materiales comentados anteriormente, se va a tener muchos problemas debido a las características del material a imprimir, de la configuración de impresión y de la calibración de la impresora. Algunos de estos errores y posibles soluciones son [6]:

Levantamiento de la base (Warping)

Figura 6 - Levantamiento de la base (Warping).

Este error es debido a que las esquinas de la base de la pieza se separan de la plataforma y deforman el objeto.

El Warping se produce con el enfriamiento y a su vez la contracción del material. A medida que transcurre la impresión el plástico que es extruido, se contrae y empieza a tirar del material que lo rodea. Con el tiempo las fuerzas internas se hacen tan grandes que la base de la pieza que está en impresión, la pieza se dobla hacia arriba y se despega de la plataforma.

La mejor manera de corregir este problema es utilizando una plataforma de impresión caliente (o a veces llamada cama caliente). Conforme transcurre la impresión, la plataforma mantiene la pieza templada lo justo para que el plástico deje de estar totalmente sólido.

También es importante que la plataforma esté lo mejor nivelada posible. El plástico tiene que estar aplastado sobre la plataforma para asegurarnos de que se adhiere correctamente. Además de conseguir que la impresión no se deforme o despegue conseguiremos una capa inferior uniforme. Lo que buscamos son unas líneas uniformes que se toquen entre sí. Si las líneas muestran signos de rebose significa que la plataforma está demasiado pegada al cabezal y tendremos que calibrar de nuevo. Si las líneas no se adhieren y quedan desordenadas quiere decir que la plataforma está demasiado baja.

Almhadillado (Pillowing)

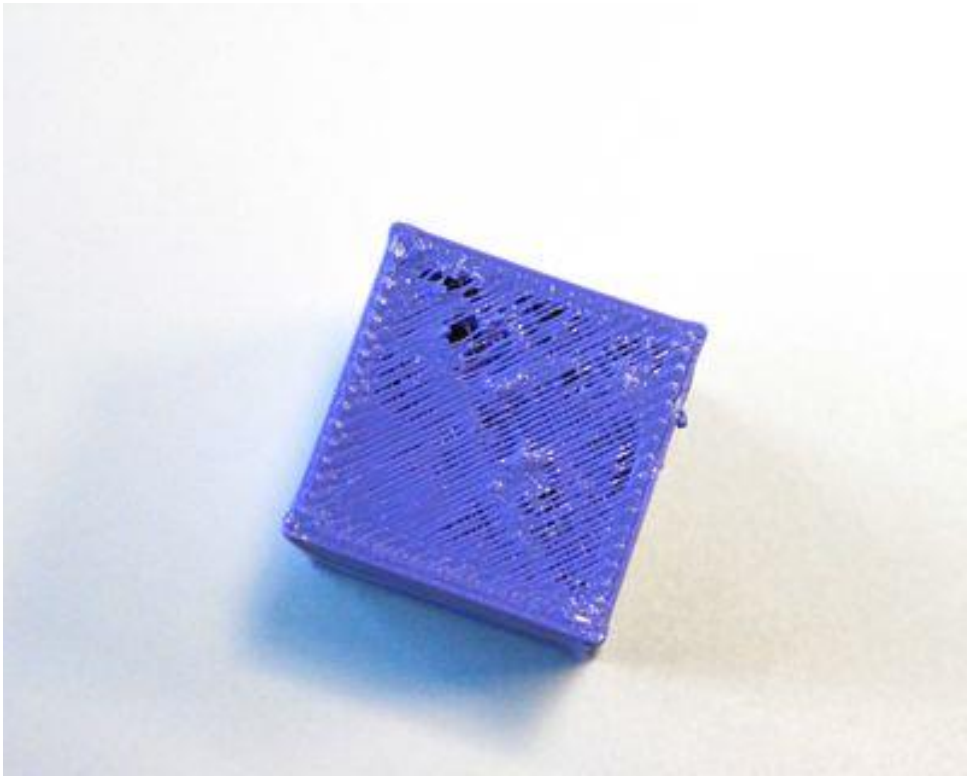


Figura 7 - Almhadillado (Pillowing).

Puede ocurrir que al cerrar la parte final de una pieza aparezcan protuberancias o huecos, siendo este efecto más común usando PLA. Lo más importante para solucionar el problema es fijarse en la refrigeración y en el grosor de la capa final.

Sin la refrigeración adecuada, las hebras de plástico tienden a rizarse y elevarse de la superficie de impresión haciendo más difícil que las capas posteriores se depositen correctamente.

Hay que tener en cuenta varios aspectos para solucionar este problema, uno es la refrigeración como hemos dicho anteriormente, otro es el grosor de las capas finales, aumentándolo también se corregiría.

Encordado (Strining)

Figura 8 - Encordado (Strining).

El encordado consiste en hilos horizontales que se generan cuando imprimimos dos piezas a la vez moviéndose de una pieza a otra capa por capa o cuando el objeto que queremos imprimir tiene capas en las que se divide en partes distanciadas e independientes.

La solución principal para resolver la aparición del encordado es la retracción. Cuando la retracción está activada, permitimos a la impresora absorber algo de filamento antes de mover el cabezal de impresión a través de un espacio abierto donde no tenga que imprimir. Retrayendo el filamento ayudamos a prevenir el goteo de plástico a través de la boquilla durante el desplazamiento en vacío del cabezal.

Antes de nada lo que debemos hacer es asegurarnos de que la retracción está habilitada. En Cura, esta configuración se encuentra en la pestaña "Básica" en forma de casilla de verificación que dice “Enable retraction” (habilitar retracción).

Voladizos (Overhang)

Figura 9 - Voladizos (Overhang).

Las impresoras 3D de deposición no pueden depositar material en el aire. Esta limitación genera piezas con aspectos indeseados y marañas de hilos.

Cada capa se imprime sobre la anterior, sirviendo ésta de apoyo para la siguiente. Hay ocasiones en los que debido a la forma del objeto a imprimir no tenemos material debajo y el material queda colgando formando una “U” en vez de quedar como una línea recta. Se imprimen parcialmente en el aire y tienden a hundirse ligeramente hacia abajo. Hay muchas variables que afectan a lo bien o mal que vayamos a imprimir: temperatura, velocidad de impresión, ángulo y longitud del voladizo, el material, la refrigeración, etc.

El software Cura nos ofrece la opción de “soportes”, creando pequeños soportes en estas zonas donde la pieza se queda en el aire. Estos soportes son fáciles de quitar después de haber acabado la impresión.

Subextrusión (Under extursion)

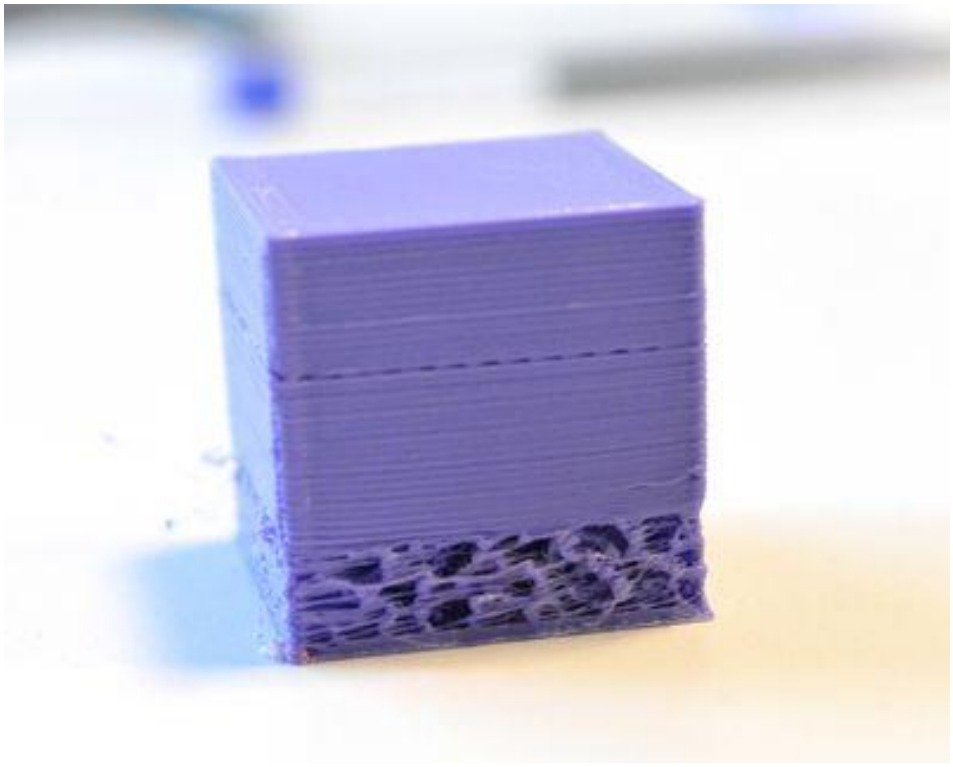


Figura 10 - Subextrusión (Under extursion).

Se llama *subextrusión* al error que ocurre cuando el extrusor de la impresora no puede suministrar la cantidad de plástico que debería. Los síntomas de este problema pueden ser tres, pueden faltar capas, las capas pueden ser muy finas o puede que tengan puntos y agujeros en ellas al azar.

2.6. COMUNIDADES DE IMPRESORES 3D

Básicamente son repositorios online donde se puede intercambiar modelos 3D para la impresión. Estos lugares son un encuentro entre usuarios, diseñadores, impresores, fabricantes, donde se pueden obtener estos modelos para diferentes usos.

A día de hoy, un gran número de procesos aditivos y comunidades de impresores están disponibles para cualquiera, y debido a las ideas de código abierto este tipo de diseños y modelos se pueden obtener bajo unas licencias de uso y comercialización sin ánimo de lucro.

Existen varios tipos de licencias libres, que conceden permisos a las personas que lo descargan, ya que se trata de una plataforma para aprender y compartir. Para ello existe la licencia Common Creative[7] con distintas opciones que mantienen la autoría del diseñador, pero totalmente abierta a los demás usuarios.

A continuación, se enumeran algunos de los ejemplos más relevantes de este tipo de comunidades:

- 3D printing Industry [8]
- 3D printer hub [9]
- Thinginverse [10]
- Solidforum [11]

2.7. DISEÑO DE LA PRÓTESIS

El diseño de la mano viene dado por un repositorio online de modelos 3D, como se ha mencionado en el apartado anterior. Concretamente, este archivo se ha obtenido de la plataforma Thinginverse [10], cuyo autor es la comunidad InMoov [12].

Esta comunidad fue iniciada para crear un robot construido con piezas de plástico impresas mediante cualquier impresora 3D. El proyecto es una plataforma de desarrollo y de aprendizaje de la robótica, donde los derechos están bajo licencia Common Creative [7].

Debido al uso de componentes y tecnologías abiertas, muchos desarrolladores han realizado modificaciones en el robot Inmoov para mejorar sus funciones.

En este proyecto se ha aprovechado la oportunidad de esta licencia de desarrollo y aprendizaje para poder adaptar una parte de este robot, su mano, para satisfacer otras necesidades como en este caso, el uso para prótesis humanas.

La mano del ser humano está constituida por los huesos carpianos, un cúmulo de 8 huesos. Seguidamente están los huesos metacarpianos, uno por cada dedo de la mano, después de cada metacarpiano están las falanges proximales, que finalmente se suman a ellas las falanges medias y distales.

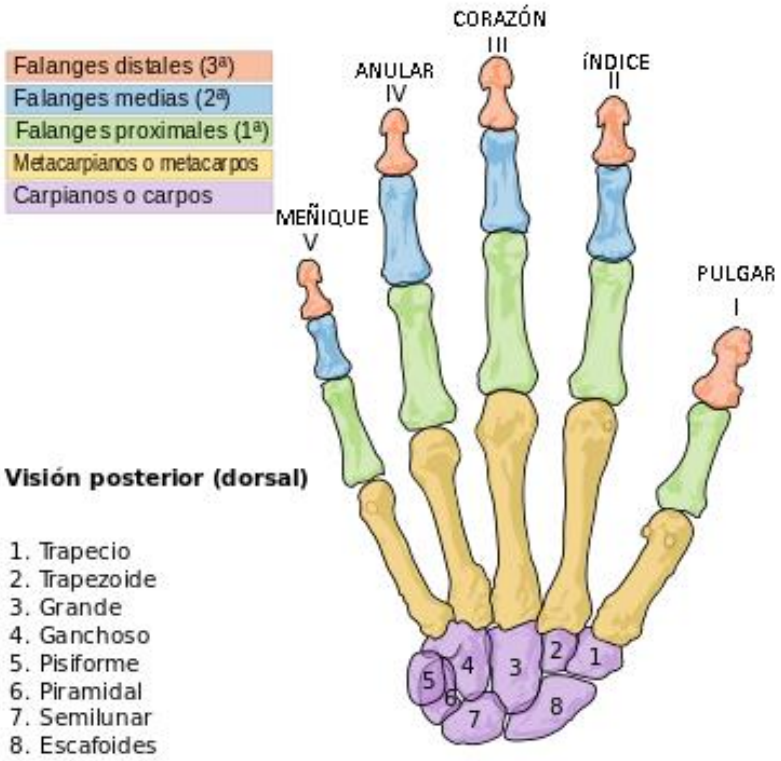


Figura 11 - Huesos de la mano.

En diseño de la mano que se muestra en la figura 12 se puede observar que está inspirada en una mano humana intentando imitar todos sus movimientos naturales.

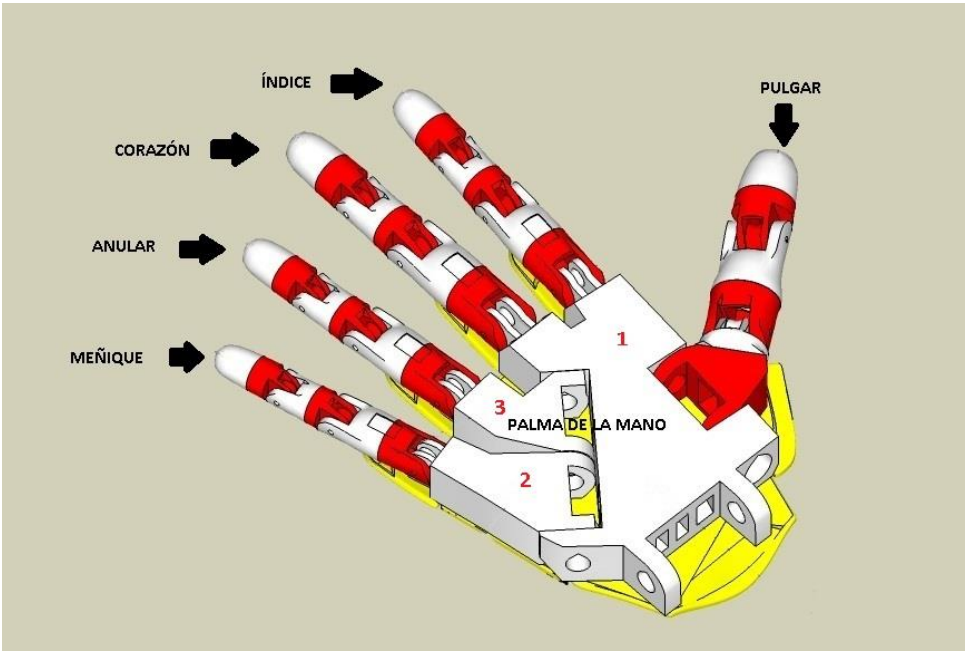


Figura 12 - Diseño palma de la mano.

La palma de la mano está constituida por 3 partes. La número 1 hace referencia a los huesos carpianos y los metacarpianos del dedo índice y corazón, haciendo de soporte también para los Metacarpianos del dedo meñique (figura 12, 2) y del dedo anular (figura 12, 3). También hace de soporte para los metacarpianos del dedo pulgar su falange proximal y distal.

Para el movimiento de la prótesis se tiene en cuenta las articulaciones en:

- **Dedo pulgar:** Movimiento de metacarpianos, falanges proximales, medias y distales.

Figura 13 - Movimiento dedo pulgar. Imagen detalle sobre la prótesis realizada.

- **Dedos índice y corazón**: Movimiento de falanges proximales, medias y distales.

Figura 14 - Movimiendo dedos índice y corazón. Imagen detalle sobre la prótesis realizada.

- **Dedos anular y meñique:** Movimiento de metacarpianos, falanges proximales, medias y distales.

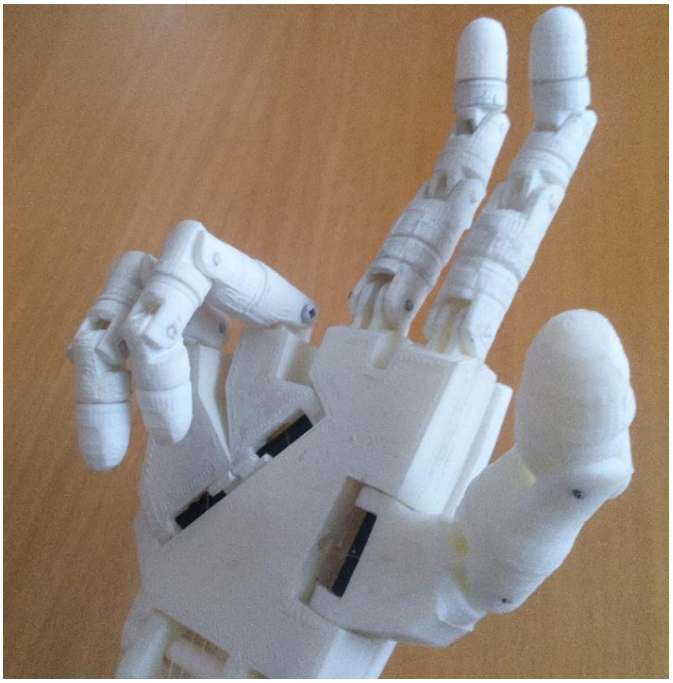


Figura 15 - Movimiento dedos anular y meñique. Imagen detalle sobre la prótesis realizada.

En cuanto al movimiento en la palma de la mano de los metacarpianos del dedo meñique y anular, como muestra la figura 16, el dedo meñique tiene un movimiento de 20 grados y el dedo anular 15 grados respectivamente.

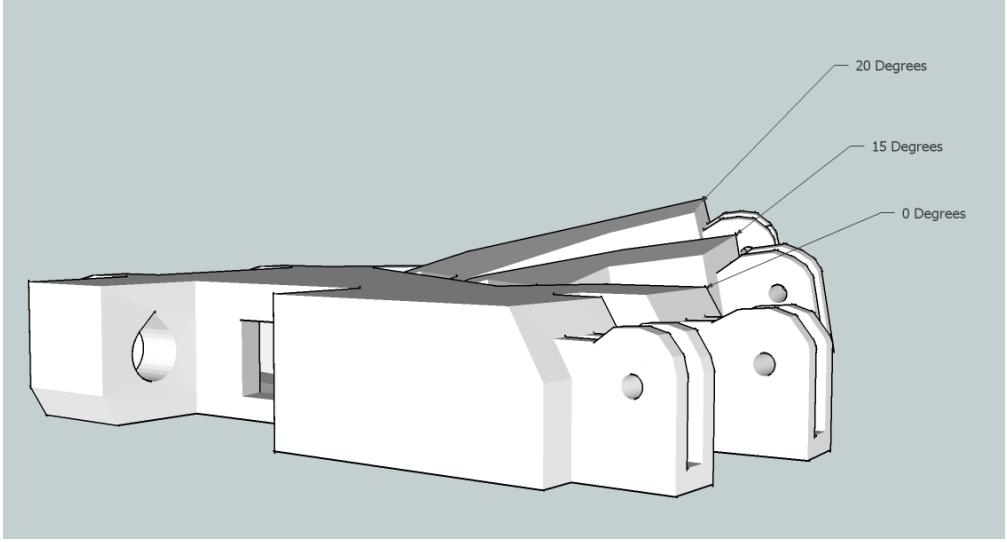


Figura 16 - Grado de movimiento de los metacarpianos en la palma de la mano.

Finalmente, el movimiento de muñeca tiene un giro de 90 grados para cada sentido como muestra la figura 17.

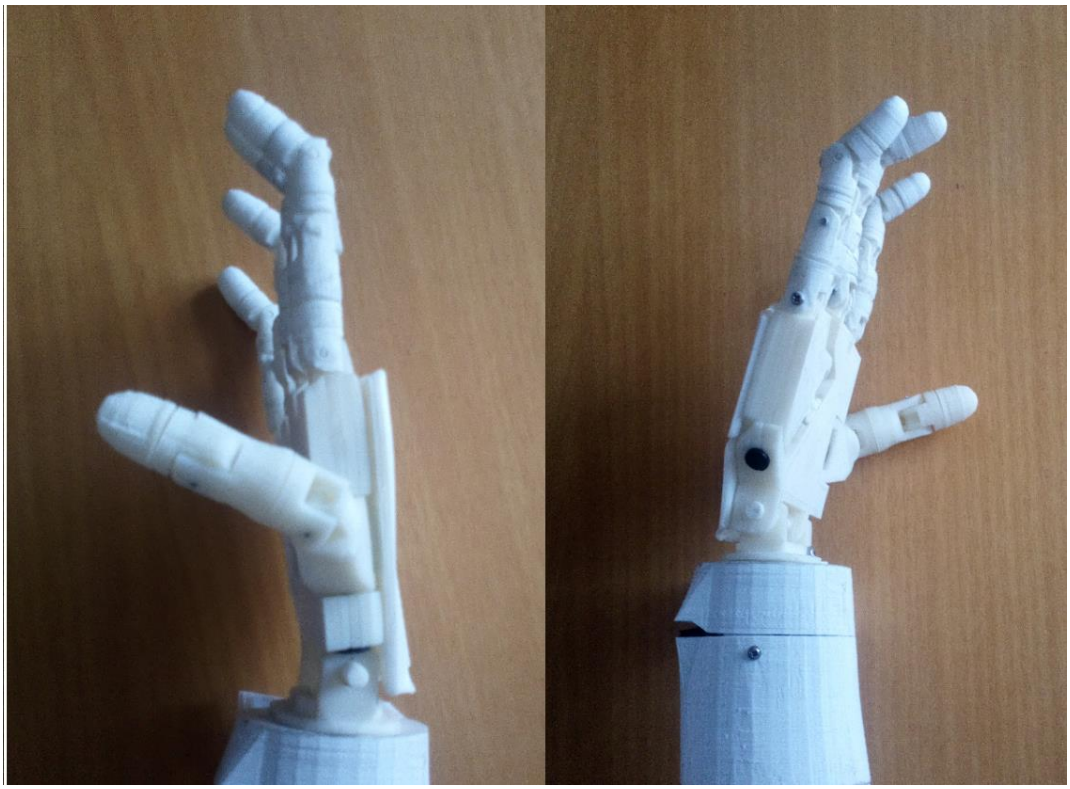


Figura 17 - Giro muñeca. Imagen detalle sobre la prótesis realizada.

CAPÍTULO 3

SEÑALES MIOELÉCTRICAS

3.1. TEJIDO MUSCULAR

Se define músculo como un tejido blando compuesto por fibras que se puede contraer y relajar mediante impulsos nerviosos generando movimientos [13]. Otras de sus funciones son brindar estabilidad articular, mantener la postura, transformar la energía mecánica en química, aportar calor, estimular los vasos sanguíneos e informar sobre el estado fisiológico del cuerpo.

Existen varias clasificaciones del tejido muscular dependiendo de su localización, forma o naturaleza.

Según su localización: Cutáneos y profundos.

Según su forma: Cuadrados y circulares.

Según su naturaleza: Estriados, lisos y cardíacos.

- Estriado o esquelético: Son los únicos que están bajo control voluntario, están adheridos a los huesos.

- Lisos: Están bajo control involuntario, de apariencia estriada, se encuentran en las paredes de los órganos viscerales huecos (excepto del corazón). Presentan una conductividad y una excitabilidad mayor que el resto de los músculos.

- Cardíacos: También están bajo control involuntario, tienen apariencia estriada, localizado en las paredes del corazón.

El cuerpo humano tiene unos 650 músculos, siendo el 80% esqueléticos y el 20% restante cardíacos y lisos. Nos centraremos en desarrollar los músculos estriados o esqueléticos, los cuales están formados por fibras, que a su vez están formadas por miofibrillas, y éstas se constituyen de miofilamentos. Los miofilamentos son pequeños hilos compuestos por proteínas de miosina y actina. La actina se entrelaza con la miosina, dando lugar a una especie de puentes que servirán para tirar de los filamentos y contraer el músculo.

Según la interacción entre la miosina y la actina, los músculos se contraerán y se relajarán. Cuando el músculo se encuentra en reposo relativo, es decir las fibras musculares no están totalmente en reposos, no hay interacción entre ambas proteínas.

Esto se debe a la acción de la troponina y la tropomiosina, otras dos proteínas que bloquean la interacción entre la actina y la miosina.

Los impulsos nerviosos procedentes del cerebro liberan iones de calcio, lo cual cambia la configuración y desplaza a la troponina y la tropomiosina. Esto permite que los miofilamentos de actina formen puentes con los de miosina y se desplacen sobre ellos, acortando así la longitud de las fibras musculares, lo que hacer contraer el músculo. Información obtenida [14].

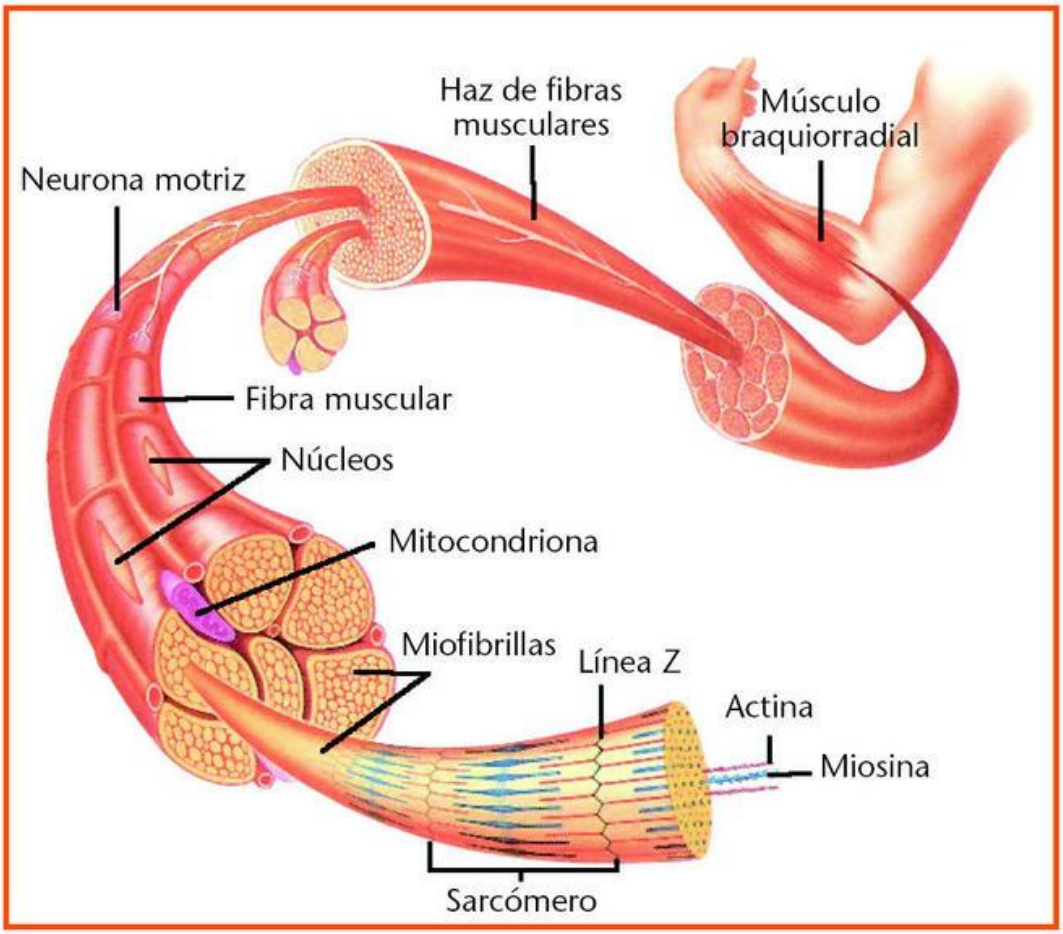


Figura 18 - Anatomía músculos.

3.2. ELECTROMIOGRAMA

La señal de electromiografía [15] tiene la misma base que el ECG. Se trata de obtener, mediante electrodos, la información del potencial de acción que activa una unidad motora. Cuando se quiere flexionar un músculo, las neuronas motoras activan un conjunto de fibras musculares. Para una mayor fuerza, o bien se activan más fibras musculares (reclutamiento espacial) o se incrementa la cantidad de disparos de potenciales por unidad de tiempo (reclutamiento temporal).

En la obtención del EMG se utiliza un electrodo de referencia situado lejos de los paquetes musculares (codo, rodilla) y dos o más electrodos sensores. Estos pueden colocarse de forma intramuscular (técnica invasiva mediante la colocación de los electrodos en el interior del músculo) o superficial (con electrodos adheridos a la piel).

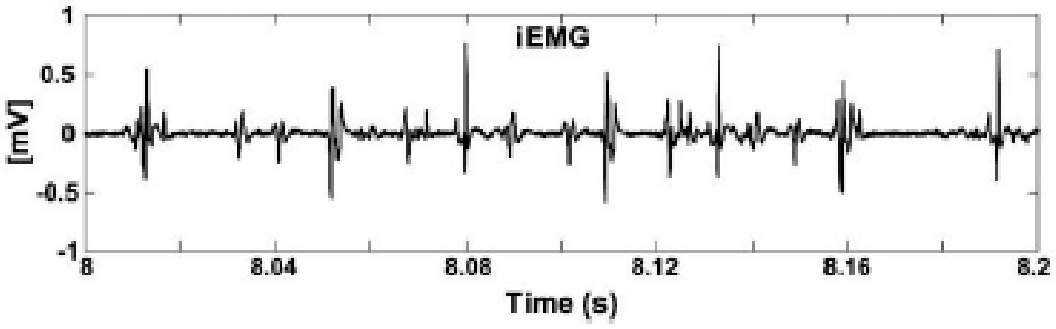


Figura 19 - EMG intramuscular.

Estas señales suelen tener componentes frecuenciales altas (varios KHz), por lo que hay que tener cuidado en su filtrado.

Por otro lado en los EMG superficiales, la señal no suele tener mucha precisión al recogerse señal de muchas unidades motoras, las cuales se superponen, siendo muy complejo analizar el reclutamiento espacial y/o temporal. No obstante, se puede utilizar fácilmente para detectar el inicio del impulso y el nivel de éste. Por ello, la señal suele ser filtrada y rectificada para obtener una integración de todas las activaciones de unidades motoras durante el movimiento.

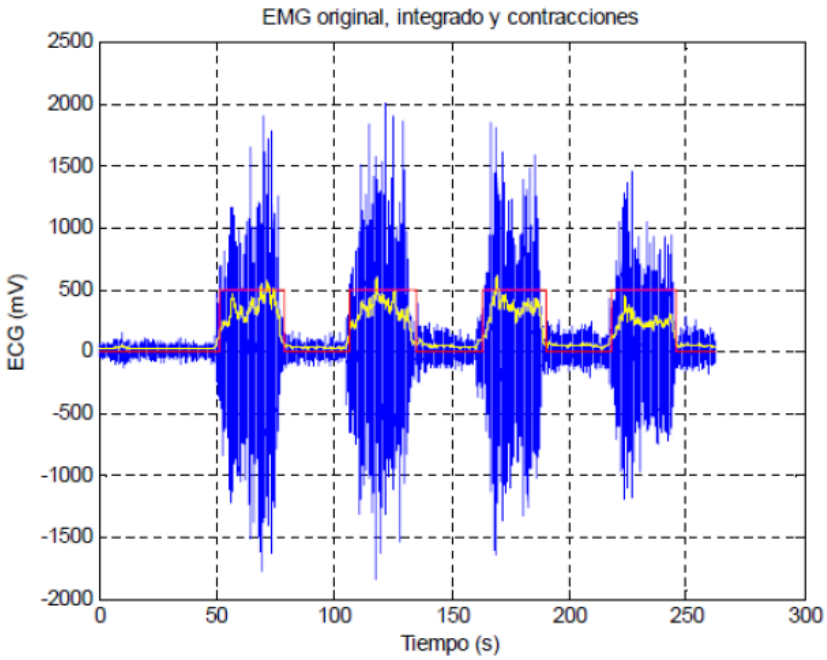


Figura 20 - EMG superficial.

Las principales fuentes de ruido son las de la alimentación o movimientos indeseados del electrodo, aunque estos suelen ser de baja frecuencia y pueden ser fácilmente filtrados en el acondicionamiento de señal.

3.3. OBTENCIÓN DEL EMG CON E-HEALTH

EMG superficial

La electromiografía es una técnica para medir la actividad de las corrientes eléctricas de los músculos esqueléticos. Gracias a esta técnica los médicos pueden diagnosticar enfermedades neuromusculares y desórdenes en el control motor.

Para realizar un EMG se necesita un Electromiógrafo. Los dispositivos más simples constan de dos electrodos: un emisor y un ground (GND), que miden la conductividad de la parte externa del músculo. Sin embargo, se suelen utilizar tres electrodos para eliminar ruido: dos emisores y un GND. Posteriormente, la señal de los electrodos se procesa, primero se amplifica y después le aplica un filtro de paso alto para eliminar interferencias.

La amplitud de las señales EMG varía desde los μV hasta un bajo rango de mV (menor de 10mV). La amplitud, y las propiedades de las señales EMG, tanto en el dominio del tiempo como en la frecuencia, dependen de factores tales como:

- El tiempo y la intensidad de la contracción muscular.
- La distancia entre el electrodo y la zona de actividad muscular.
- Las propiedades de la piel (por ejemplo el espesor de la piel y tejido adiposo)
- Las propiedades del electrodo y el amplificador y la calidad del contacto entre la piel y el electrodo.

Los aspectos más importantes relacionados con la adquisición y el análisis de señales EMG de superficie fueron tratados recientemente en un consenso multinacional llamado Surface EMG for the Non-Invasive Assessment of Muscles (SENIAM), donde se discute desde la construcción del electrodo hasta su ubicación.

La calidad de la señal EMG medida es usualmente descrita por la relación entre la señal EMG medida y las contribuciones de ruido indeseadas por el ambiente. La meta es maximizar la amplitud de la señal mientras se minimiza el ruido.

E-HEALTH

Figura 21 - E-HEALTH.

Utilizando como base Arduino, han surgido numerosas extensiones que permiten ampliar las funcionalidades del mismo. Una de ellas es el Shield e-Health, pensado para la adquisición de señales biomédicas.

El e-Health Sensor Platform V2.0 [16], además de para la medida del ECG y el EMG, presenta conectores para otros seis sensores de señales biológicas. En total, las ocho señales biológicas tratadas son:

- Posición del cuerpo
- Electrocardiograma (ECG)
- Pulso y oxímetro
- Presión Sanguínea
- Temperatura
- Flujo Respiratorio
- Respuesta galvánica de la piel
- Glucosímetro
- Electromiograma

La placa contiene toda la electrónica de acondicionamiento de la señal mientras que los sensores propiamente dichos han de ser adquiridos aparte. La alimentación de la extensión se realiza mediante una señal de Direct Current (DC) de 12V / 2A que toma directamente de la placa del Arduino a la que se conecta. Dependiendo del uso que se le dé y del PC al que esté conectado, si la alimentación del Arduino es a través del

Universal Serial Bus (USB) puede no ser suficiente, necesitándose una fuente de alimentación externa. En el caso del sensor EMG y el PC utilizado no ha sido necesaria ninguna alimentación adicional.

Los sensores de ECG y EMG son físicamente muy parecidos, diferenciándose en que son conectados a distintos terminales y ha de seleccionarse mediante un puente cuál de los dos es utilizado para medir.

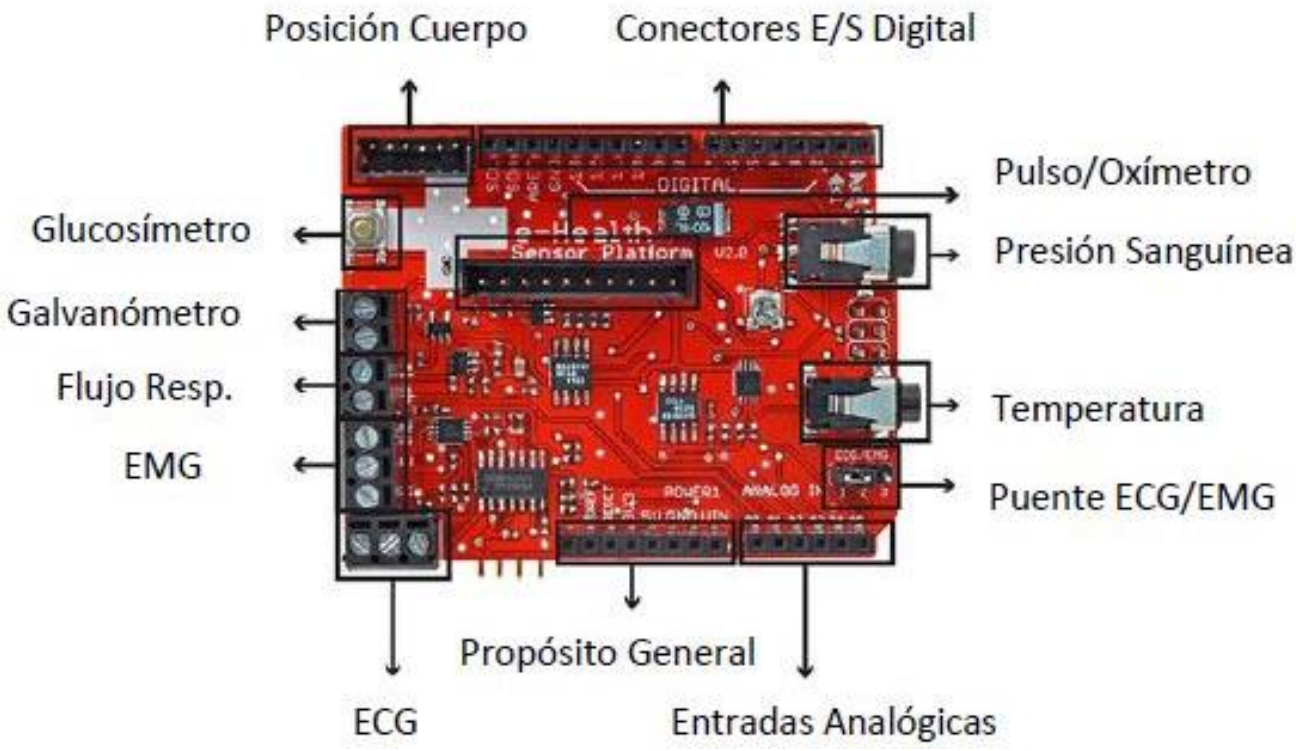


Figura 22 - Placa e-Health.

HARDWARE

El sistema de acondicionamiento para el sensor deL EMG consta de cuatro etapas, una amplificación diferencial, un rectificado, un filtrado activo y una etapa de amplificación regulable.

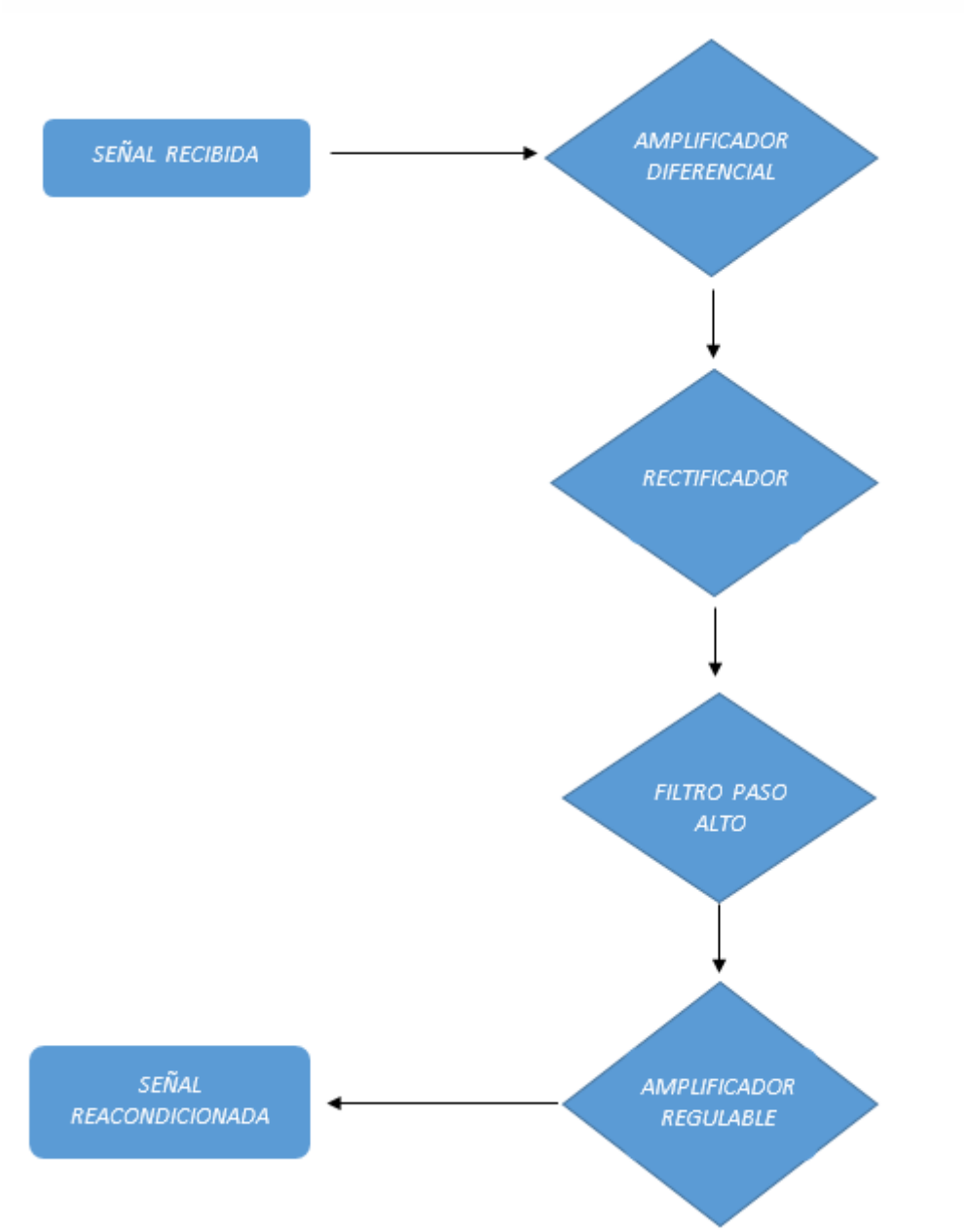


Figura 23 - Esquema EMG e-Health.

La primera etapa hace uso del AD8221 de Analog Device para amplificar la señal diferencial MID-END. La ganancia es regulable mediante R1, según la relación:

$$G = 1 + \frac{49{,}4\ K}{R_1} \approx 207$$

La siguiente etapa realiza el rectificado de la señal. Se tiene en primer lugar un filtrado paso alto, que elimina señales de baja frecuencia. Después el rectificador propiamente dicho. En función de que la señal de entrada sea positiva o negativa, conducirá un diodo u otro, atacando a la segunda etapa por la entrada positiva o negativa y, por tanto, invirtiéndose o no dicha señal, con el efecto final de tener la señal rectificada a la salida.

La tercera etapa es un inversor, con comportamiento paso bajo y ganancia unidad. Su frecuencia de corte es de 2Hz, que elimina todas las componentes de alta frecuencia que aparecen del rectificador. De esta forma se obtiene la envolvente de la señal, que agrupa la información de todas las fibras participantes en el movimiento muscular.

Finalmente, se tiene un amplificador inversor, cuya ganancia se puede regular con el potenciómetro RV1.

Los electrodos han de posicionarse en función del músculo que se desee analizar. En los experimentos realizados en el presente proyecto se ha analizado el bíceps del brazo, con lo cual, los electrodos han sido posicionados en las posiciones indicadas en la siguiente figura.

Figura 24 - Posiciones electrodos.

Una vez analizada la electrónica de acondicionamiento de señal, los pasos necesarios para la conexión física del sistema son los siguientes:

- Conexión de la extensión e-Health al Arduino, conexión del Arduino al PC.

- Conexión de los electrodos de medida y preparación de un latiguillo para la selección de la bioseñal a capturar a la extensión e-Health. Hay que tener en cuenta que, además, se tendrá que colocar el puente incluido en el e-Health en la posición correspondiente a la medida deseada.

SOFTWARE

La programación del Arduino para la captura del EMG es muy sencilla gracias a la librería suministrada por el propio e-Health. Es importante tener en cuenta que estas librerías solo se pueden utilizar con el Arduino Integrated Development Environment (IDE). Software Arduino 1.0.1, por lo que la programación con la última versión del IDE de Arduino no funciona correctamente.

El sketch es muy sencillo. El bloque de declaración incluye la librería de e-Health [16].

```
#include <eHealth.h>
```

El bloque **setup()** iniciará la velocidad del puerto serie.

```
void setup() {
Serial.begin(115200);
}
```

El bloque **loop()** capturará continuamente la señal deseada, en función del nivel encontrado en el pin digital que se quiera configurar para la entrada de datos. Cada dato capturado es enviado por el puerto serie.

```
void loop() {
float ECG =eHealth.getECG();
Serial.println("");
}
```

Los datos así capturados pueden ser mostrados en el terminal del puerto serie y ser fácilmente trasladados a un fichero de texto para su posterior tratamiento.

CAPÍTULO 4

ARDUINO

3.1. ¿QUÉ ES ARDUINO?

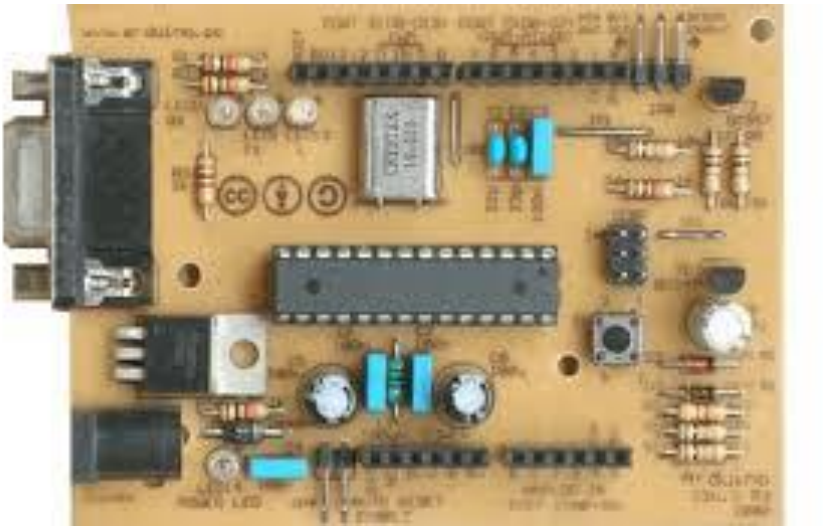


Figura 25 - Primer Arduino.

Arduino [17] es un proyecto de *hardware* y *software* de código abierto que se creó en el Instituto Ivrea de Italia en 2005. Basta con decir que se trata de un proyecto de fuente libre (open-source) consistente en un sistema *hardware* y un entorno de desarrollo para su programación llamado Arduino Integrated Development Environment (IDE). Esta pequeña placa tiene una larga historia que se puede consultar en numerosos libros y textos, por lo que no es objetivo de este proyecto narrarlo.

Es una plataforma de prototipos electrónica de código abierto basada en hardware y software flexibles y fáciles de usar. Está pensado e inspirado en artistas, diseñadores, y estudiantes de computación o robótica y para cualquier interesado en crear objetos o entornos interactivo, o simplemente por hobby.

Arduino consta de una placa principal de componentes electrónicos, donde se encuentran conectados los controladores principales que gestionan los demás complementos y circuitos ensamblados en la misma.

Además, requiere de un lenguaje de programación para poder ser utilizado y, como su nombre lo dice, programado y configurarlo a nuestra necesidad, por lo que se puede decir que Arduino es una herramienta "completa" en cuanto a las herramientas principales nos referimos, ya que sólo debemos instalar y configurar con el lenguaje de programación de esta placa los componentes eléctricos que queramos.

Arduino también simplifica el proceso de trabajo con micro controladores, ya que está fabricada de tal manera que viene “pre ensamblada” y lista con los controladores necesarios para poder operar con ella una vez que la saquemos de su caja, ofreciendo una ventaja muy grande para profesores, estudiantes y aficionados interesados en el desarrollo de tecnologías.

3.2. HARDWARE Y SOFTWARE

El hardware [17] consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados en las plataformas Arduino son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez.

Como hemos comentado en el apartado anterior Arduino es hardware de “Open source”, que consiste en una placa con un microcontrolador programable a alto nivel. Cuenta con una serie de puertos de entradas y de salidas, utiliza microcontroladores AVR de 8 bits y los más usados son el Atmega168, Atmega328, Atmega1280, y Atmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños AVR. Gracias al IDE de Arduino los programas se compilan y luego se ejecutan sin cambios en cualquiera de las plataformas.

Arduino cuenta con una conexión USB a un ordenador, por donde se comunicarán las instrucciones que viajarán a otro tipo de hardware conectado, como pueden ser leds, motores, shields que se apilan sobre Arduino, módulos bluetooth, gps, etc entre otros.

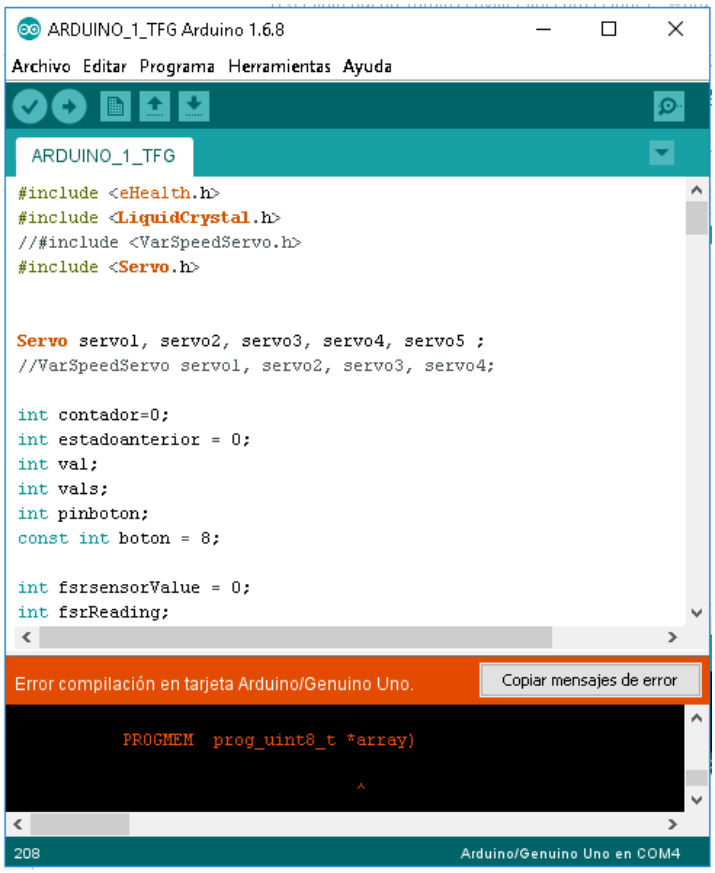


Figura 26 - IDE Software Arduino.

El sistema está formado por:

- Un circuito impreso con microcontrolador.
- Un software para programarlo.
- Una amplia comunidad online con la cual interactuar para recibir y dar soporte.

El lenguaje de programación utilizado para Arduino está basado en Wiring, que a su vez está basado en Java, mientras que el IDE está basado en Processing y puede ser descargado gratuitamente en [18].

Respecto al IDE de Arduino, que se denomina Sketch, está compuesto de tres partes claramente diferenciadas:

- Declaración de librerías y variables
- Setup()
- Loop()

En la declaración de librerías se pueden incluir, tanto ofrecidas por el propio IDE, como creadas por el propio programador o por terceros.

El bloque **setup()** incluye todas aquellas instrucciones que se ejecutarán una sola vez cuando el programa sea cargado o cuando se pulse el botón de *reset*. Suele contener la inicialización del sistema y de variables.

El bloque **loop()** forma un bucle que se repetirá indefinidamente una vez ejecutado el **setup()**. Contiene la programación propiamente dicha del Arduino.

3.3. MODELOS DE ARDUINO

No hay un único Arduino, sino muchos modelos [17], y la placa más conocida es la de Arduino Uno. El programa se implementará haciendo uso del entorno de programación propio de Arduino y se transferirá empleando un cable USB. Si bien en el caso de la placa USB no es preciso utilizar una fuente de alimentación externa, ya que el propio cable USB la proporciona. Para la realización de algunos de los experimentos prácticos sí que será necesario disponer de una fuente de alimentación externa ya que la alimentación proporcionada por el USB puede no ser suficiente. El voltaje de la fuente puede estar entre 6 y 25 Voltios.

Arduino Uno

Figura 27 - Arduino Uno.

Es la versión de Arduino más utilizada y revisada y de la que se puede encontrar más información y ayuda para el diseño de aplicaciones. Está basado en el microcontrolador ATmega328P y consta de catorce entradas/salidas digitales (seis de ellas con posibilidad de utilizar modulación en anchura de pulso o PWM), seis entradas analógicas (no tiene salidas analógicas), un cristal de cuarzo de 16 MHz, un conector USB, un conector de alimentación (7/12V), un ICSP (In Circuit Serial Programming, que sirve para programar el BootLoader del Microcontrolador) y un botón de *reset* como características básicas.

Arduino Mega 2560

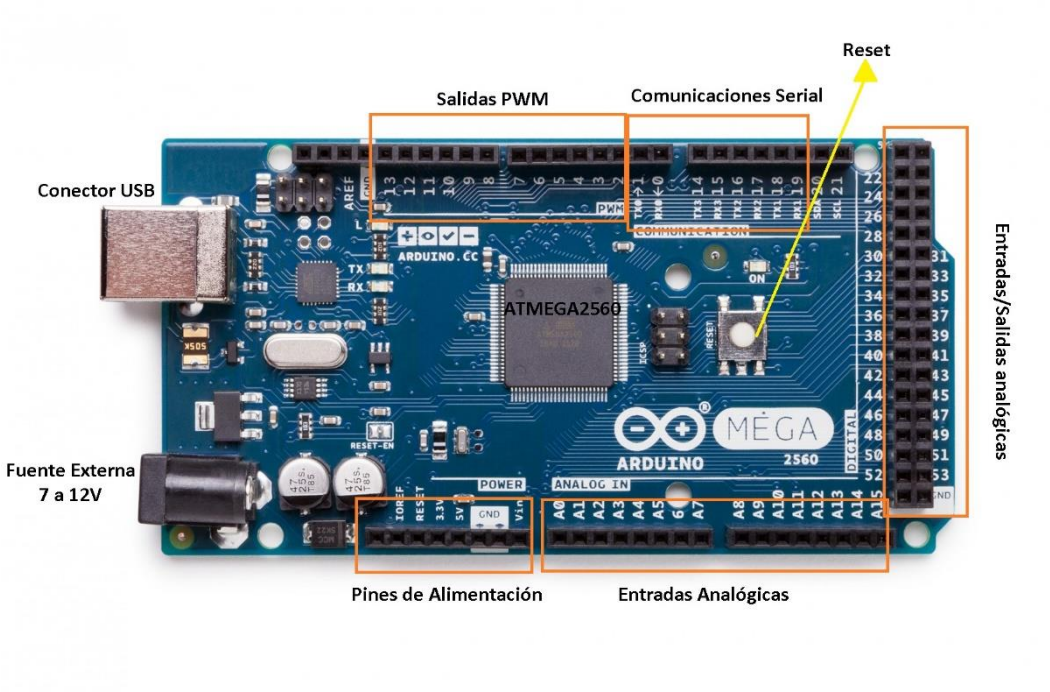


Figura 28 - Arduino Mega 2560.

Su nombre proviene del microcontrolador que lo maneja, un ATmega2560. Este chip trabaja a 16Mhz y con un voltaje de 5v. Sus capacidades son superiores al ATmega320 del Arduino UNO. Este microcontrolador de 8 bits trabaja conjuntamente con una Static Random Access Memory (SRAM) de 8KB, 4KB de Electrically Erasable Programmable Read-Only Memory (EEPROM) y 256KB de flash (8KB para el bootloader). En cuanto a características electrónicas es bastante similar al UNO. Pero como se puede apreciar, el número de pines es superior: 54 pines digitales (15 de ellos PWM) y 16 pines analógicos. Esta placa es idónea para la necesidad de más pines y potencia de la que aporta UNO.

Arduino Nano

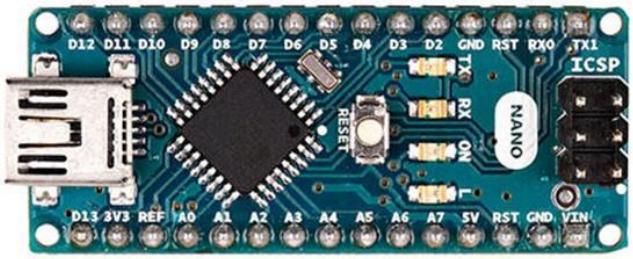


Figura 29 - Arduino Nano.

Empezaron incorporando un ATMega328 como el de otras placas vistas anteriormente, pero se sustituyó por un ATmega168 a 16Mhz. Sus dimensiones son aún muy reducidas 18,5×43.2mm. Su reducido tamaño no le quita la posibilidad de ser una placa completa, pero sí que necesita de un cable mini-USB y no posee conector de alimentación externa. Especialmente pensado para aplicaciones de reducido costo y de pequeño tamaño. A nivel eléctrico se comporta como un Arduino UNO, con 14 pines digitales (6 PWM) y 8 analógicos. Pero sus capacidades y consumo son menores, 16KB de flash (2 reservados al cargador de arranque), 1KB de SRAM y 512 bytes de EEPROM.

En este proyecto se ha optado por la utilización de un solo Arduino, el Arduino Mega 2560. Se elige este modelo porque el uso de numerosas salidas y entradas de datos hacen que Arduino Uno y Nano acaben sin recursos para la demanda que se presenta con los diversos sensores y procesado de datos.

3.4. SENSORES UTILIZADOS

Electrodos de polímeros conductivos

Figura 30 - Electrodos EMG.

Son hechos de un material que es conductivo y adhesivo a su vez. No son buenos para mediciones de bajo ruido por su alta resistividad. Sin embargo, cuando el nivel de la señal es alto y se minimiza la interferencia restringiendo al sujeto de movimiento, estos electrodos ofrecen una buena y económica solución. Nos permiten captar la señal eléctrica generada por los músculos cuando éstos responden a un estímulo o realizan un movimiento.

Módulo HC-06 Bluetooth

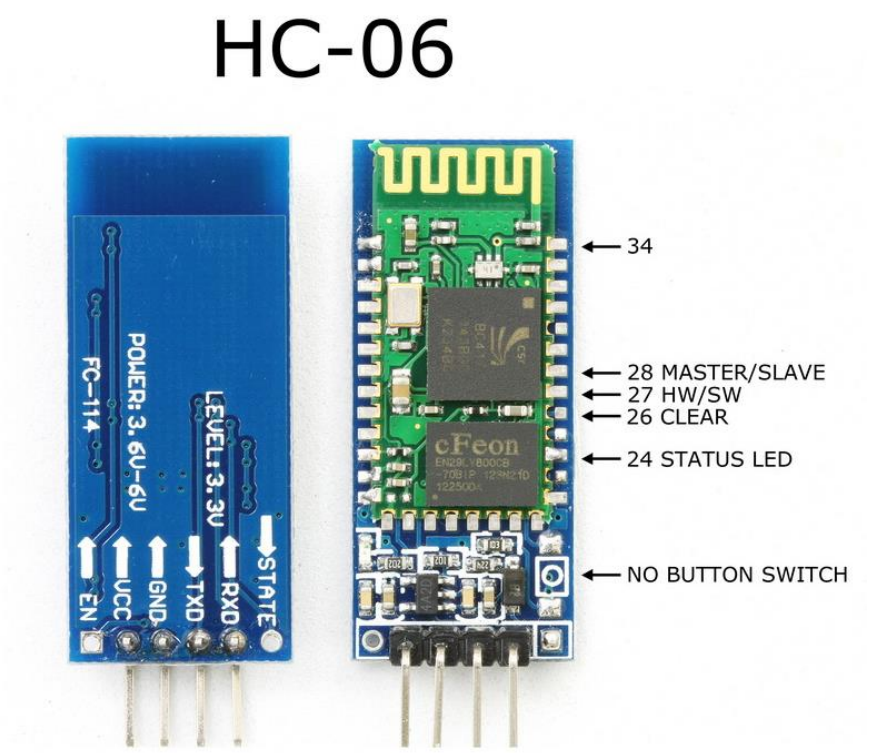


Figura 31 - Módulo bluetooth HC-06.

Este sensor será el encargado de la transmisión de datos por ondas electromagnéticas entre Arduino y el dispositivo móvil. Trabaja a la frecuencia de 2.4 GHz, y tiene un alcance de 5 a 10 metros.

Para que el software de Arduino pueda interactuar con este módulo es necesario incluir la librería serial para transmitir y recibir datos por los puertos seriales del Arduino al igual que definir por donde se van a enviar los datos.

```
#include <SoftwareSerial.h>
```

Se definen los pines Rx y Tx (recepción y transmisión) del Arduino conectados al Bluetooth

```
SoftwareSerial BT(0, 1);
```

En el **setup()** se inicial el puerto serie del bluetooth y el puerto serie.

```
void setup()
{
  BT.begin(9600);
```

```
 Serial.begin(9600);
 }
void loop()
{
  if(BT.available())     // Si llega un dato por el puerto BT se envía
al monitor serial
  {
    Serial.write(BT.read());
  }

  if(Serial.available())  // Si llega un dato por el monitor serial se
envía al puerto BT
     BT.write(Serial.read());
```

Pantalla LCD HD44780

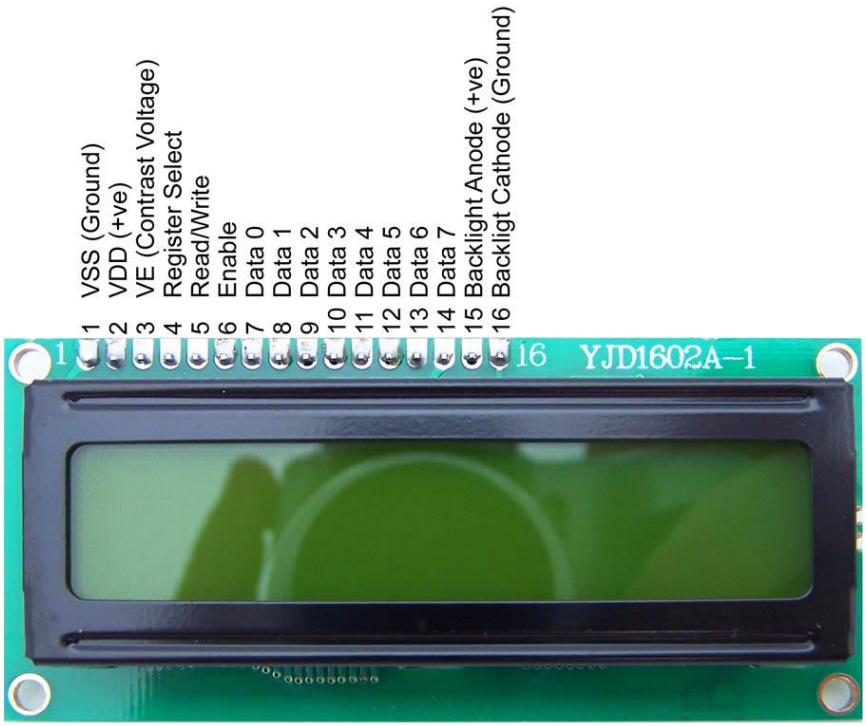


Figura 32 - LCD HD44780.

Esta pantalla permite mostrar diferentes mensajes que nos avisan del estado en el que nos encontramos, muestra un valor o cadena de caracteres o puede mostrar la respuesta de otro sensor utilizado.

Giroscopio MPU6050

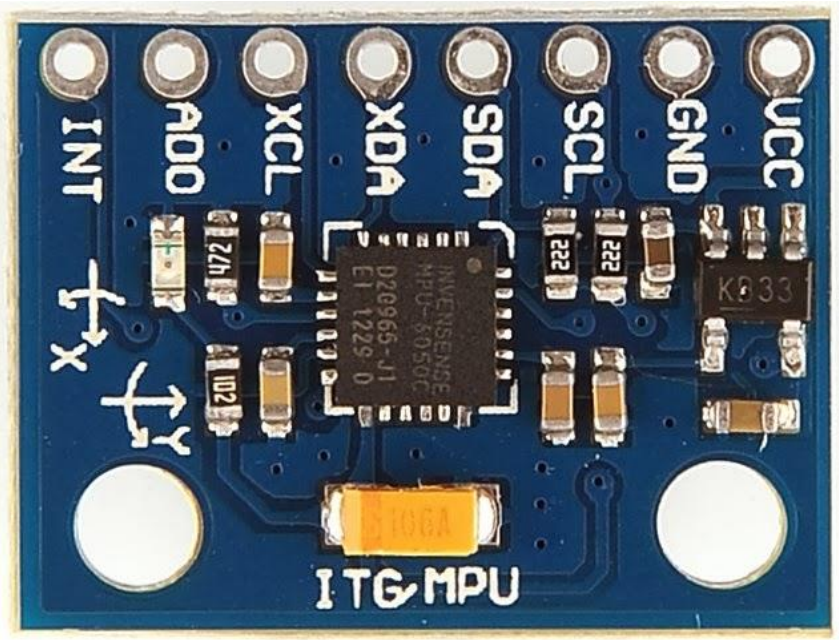


Figura 33 - MPU 6050.

El sensor InvenSense MPU-6050 contiene un acelerómetro MEMS y un giroscopio MEMS en un solo chip. Es muy preciso, ya que contiene hardware de conversión de 16 bits analógico a digital para cada canal. Por lo tanto, captura los canales “x”, “y” y “z” al mismo tiempo. El sensor utiliza el protocolo de comunicación I2C para interconectar con Arduino.

```
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Servo.h"

MPU6050 mpu;

int16_t ax, ay, az;
int16_t gx, gy, gz;

Servo myservo;

int val;
int prevVal;
```

En el **setup()** se inicializa el MPU5060 y se comprueba su conexión.

```
void setup()
{
    Wire.begin();
    Serial.begin(38400);

    Serial.println("Initialize MPU");
    mpu.initialize();
    Serial.println(mpu.testConnection() ? "Connected" : "Connection
failed");
    myservo.attach(9);
}
```

En el **loop()** se obtiene la posición del sensor y se convierte el rango de variación de movimiento en el sentido de giro del servo con la función **map()**.

```
void loop()
{
    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    val = map(ay, -17000, 17000, 0, 179);
    if (val != prevVal)
    {
        myservo.write(val);
        prevVal = val;
    }
    delay(50);
}
```

Servo Futaba S3003

Figura 34 - Servo Futaba S3003.

El servomotor Futaba S3003 consta de tres terminales de las cuales sobresale un cable. El cable de color rojo, representa la alimentación de entrada para el funcionamiento del motor (4.5-6 voltios). El negro, es tierra o GND. Y el blanco, representa la señal de control.

Este servo tiene un rango de movimiento de 180º y su función es la de dotar de movimiento a los dedos de la mano y al giro de muñeca.

3.5. APLICACIÓN ARDUINO

En esta aplicación se han tenido que instalar librerías externas para un correcto funcionamiento de la aplicación.

Las librerías son trozos de código hechas por terceros que usamos en nuestro sketch facilitando la interconexión de sensores, pantallas, módulos electrónicos, etc.

En este proyecto las librerías externas usadas son:

- eHealth.h

Esta librería [16] permite ejecutar el método getEMG(). Esta función obtiene los valores del EMG y devuelve un valor analógico en voltios (leído 0 - 1023 por el ADC) para representar la forma de onda EMG.

- VarSpeedServo.h

Con esta librería [19] se modifica la velocidad de movimiento de un servo motor. Permite variar la velocidad y girar el servo motor a los grados especificados. Para ello se utiliza la función:

servo.write(ángulo, velocidad, true);

Donde el ángulo de giro es el ángulo al que se desea girar el servomotor (en unos de 0º a 180º y en otros de 0º a 360º) y la velocidad del giro puede variar desde 0 (velocidad mínima) hasta 100 (velocidad máxima), que variará dependiendo de las características del mismo.

- I2Cdev.h

La biblioteca de dispositivos Inter-Integrated Circuit (I2C) es una colección de clases uniformes y bien documentadas para proporcionar interfaces sencillas e intuitivas a los dispositivos I2C [20].

I2C [21] es un protocolo síncrono. I2C usa solo 2 cables, uno para el reloj serial clock (SCL) y otro para el dato serial data (SDA). Esto significa que el maestro y el esclavo envían datos por el mismo cable, el cual es controlado por el maestro, que crea la señal de reloj. I2C no utiliza selección de esclavo, sino direccionamiento.

- MPU6050.h

Esta librería [22] añade información a los datos que llegan del MPU para obtener las lecturas de los ángulos en nuestro código final.

Asimismo, se implementa una serie de métodos públicos para las operaciones estrictamente necesarias:

initDmp(offsetXgyro, offsetYgyro, offsetZgyro, offsetXacel, offsetYacel, offsetZacel) : inicializa el bus I2C para la conexión con el dispositivo, los valores de offset del giroscopio y el acelerómetro, además del propio DMP.

readMPU(): comprueba si el dispositivo ha enviado valores de ángulos y, en su caso, acualiza las variables correspondientes.

getAngles(): devuelve los últimos valores para los ángulos leído desde el dispositivo.

Las salidas y las entradas que se necesitarán son declaradas en los siguientes pines:

E/S Digitales Arduino	
PIN 0 TXD	RXD (HC-06)
PIN 1 RXD	TXD (HC-06)
PIN 2	D7 (LCD)
PIN 3	D6 (LCD)
PIN 4	D5 (LCD)
PIN 5	D4 (LCD)
PIN 11	E (LCD)
PIN 12	RS (LCD)
PIN 7	Pulsador
A4	SDA (MPU6050)
A5	SCL (MPU6050)
PIN 6	INT (MPU6050)
PIN 40	Servo 1
PIN 41	Servo2
PIN 42	Servo 3
PIN 43	Servo 4
PIN 44	Servo 5
PIN 45	LED 1
PIN 46	LED 2

El siguiente parámetro a configurar es el baudratio (velocidad de recibir datos) en **setup().** Por defecto viene a 9600bps (bits por segundo) pero se usara la tasa de 115200bps para obtener correctamente los valores de EMG y no tener problemas de comunicación. Los pines de entrada y de salida también se declaran aquí.

Finalmente, en el **loop()** se realizarán las acciones que realiza la aplicación.

La aplicación está basada en una máquina de estados que consiste en una abstracción computacional que describe el comportamiento de un sistema mediante un número determinado de estados y un número determinado de transiciones entre dichos estados.

Para entrar a un estado diferente nos encontramos con un pulsador, que al ser accionado se produce el cambio de estado. En cada estado se encuentra una función que esta instanciada, así tendremos un código más ordenado y más fácil de leer.

Las transiciones de un estado a otro se generan en respuesta a eventos de entrada externos e internos, que a su vez estas transiciones pueden generar otros eventos de salida. Esta dependencia de las acciones (respuesta) del sistema a los eventos de entrada hace que las máquinas de estado sean una herramienta adecuada para la Programación Conducida por Eventos (Event Driven Programming), cual es el caso de la mayoría de los sistemas embebidos basados en microcontroladores o microprocesadores.

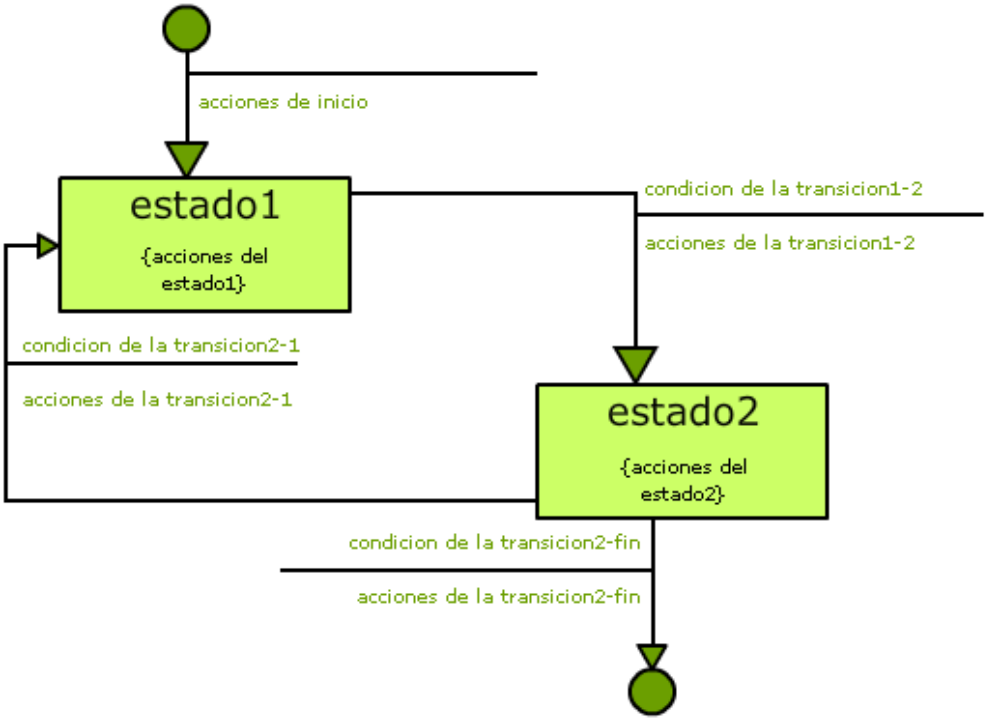


Figura 35 - Máquina de estados.

Este programa ejecuta principalmente un contador, el cual empezará desde 0 y se suma una posición cada vez que el pulsador sea accionado. En el siguiente fragmento de código se muestra este contador:

```
  if (pinboton!=estadoanterior)
{
  if(pinboton==HIGH)
  {
    contador++;
  }
}
  if(contador==4)
{
  contador=0;
}
```

Después del contador nos encontramos con la estructura de control *Switch case* que nos permitirá ejecutar un fragmento de código u otro en función del valor de una variable de nuestra elección.

Dentro de cada estado se ejecutará la función específica para cada movimiento de la mano:

```
switch (contador)
{
 case 0: CERRARMANO();

 break;

 case 1: COGERPULGARINDICE();

 break;

 case 2: CONMOVMUÑECA();

 break;

 case 3: CONEXION_ANROID();

 break;
```

- Función **CERRARMANO( ):**

Esta función hace una lectura continua de EMG convirtiendo los datos de entrada a valores de movimiento de los servos.

Una vez medida la actividad muscular que detectan los sensores, devuelve un número entero entre 0 y 1024. Lo normal es recibir valores entre 80 y 400, que varían en función de la persona.

Después con la función **map()** se convierte el número entero recibido al sentido de giro del servo. Contra mayor sea la contracción del musculo, más movimiento realizará el servo y por tanto mayor movimiento del dedo.

En este estado se realizan acciones de la vida cotidiana como por ejemplo coger o sujetar un objeto.

- Función **COGERPULGARINDICE() :**

Esta función tiene la misma metodología de actuación que la anterior, **CERRARMANO( ),** sólo que en este caso únicamente se realiza el movimiento de los dedos pulgar e índice.

En este estado se consigue coger objetos más pequeños que no es necesario utilizar todos los dedos, por ejemplo desenroscar un tapón de una botella o coger una goma de borrar.

- Función **MOVMUÑECA():**

En esta función se realizará un cierre completo de los dedos y a su vez se establecerá una comunicación con el gisroscopio MPU6050.

Con el giroscopio se obtendrán los ángulos de inclinación del brazo que accionará el sentido de giro del servomotor situado en la muñeca para poder girarla.

Un ejemplo de un caso práctico de esta función es cuando se coge una botella de agua y se quiere derramar su agua en un vaso, al intentar verter el agua todo el movimiento de inclinación se ha de hacer con el antebrazo, siendo insuficiente

inclinación y dando lugar a error. Acompañando el giro de antebrazo con un giro de muñeca solucionamos este problema.

- Función **CONEXION_ANROID():**

La finalidad de esta función es enviar o recibir datos por medio de una conexión serial, siendo una comunicación inalámbrica a través del módulo bluetooth HC06.

Primero, con la librería “SofwareSerial” añadida, inicializamos los pines de transmisión (TX) y recepción (RX) de datos. En el **setup()** definimos el baudratio con el que va a actuar nuestro bluetooth.

Seguidamente en la parte más importante, el **loop()** se ejecutará la función que está constituida por tres condiciones:

- Si la conexión recibe un valor “AZ”, el Arduino realizará un control remoto recibirá valores correspondientes entre 0-180, que corresponderán a los grados del sentido de giro de los dedos de la mano. Con esto se consigue cerrar o abrir manualmente la mano a la velocidad que se desee.

- Si la conexión recibe un valor “AY”, el Arduino realizará un envío de datos procedente del sensor giroscopio MPU 6050 informando de la posición constante del brazo.

- Si la conexión recibe un valor “AX”, el Arduino realizará un envío de datos procedentes de la realización del EMG. Enviará los valores que corresponden con los pulsos eléctricos presentándolos en una gráfica a lo largo del tiempo. Con esto se podrán colocar los electrodos en una posición correcta donde los pulsos eléctricos sean tomados con mayor claridad.

CAPÍTULO 5

ANDROID

Figura 36 - Android.

5.1. ¿QUÉ ES ANDROID?

Android es un sistema operativo inicialmente diseñado para celulares [23], al igual que sus competidores iOS, Windows Phone, Blackberry,Symbian, entre otros.

Actualmente, el sistema operativo Android se encuentra en la mayor parte de dispositivos móviles (85%), ya sean celulares o tabletas. Este sistema está basado en Linux, siendo libre, multiplataforma y gratuito.

Consta con unas características haciéndole poseedor de las siguientes cualidades:

- **Plataforma realmente abierta**: Es una plataforma de desarrollo libre basada en Linux y de código abierto

- **Adaptable a cualquier tipo de hardware***:* Android no ha sido diseñado exclusivamente para su uso en teléfonos y tabletas Hoy en día podemos encontrar una gran variedad de sistemas empotrados que se basan en este sistema operativo.

- **Portabilidad asegurada***:* Las aplicaciones finales son desarrolladas en Java, lo que nos asegura que podrán ser ejecutadas en cualquier tipo de CPU, tanto presente como futuro. Esto se consigue gracias al concepto de máquina virtual.

- **Arquitectura basada en componentes inspirados en Internet:** Por ejemplo, el diseño de la interfaz de usuario se hace en eXtensible Markup Language (XML), lo que permite que una misma aplicación se ejecute en un reloj de pantalla reducida.

- **Filosofía de dispositivo siempre conectado a Internet:** Muchas aplicaciones solo funcionan si disponemos de una conexión permanente a internet, por ejemplo: comunicaciones interpersonales o navegación con mapas.

- **Gran cantidad de servicios incorporados:** Por ejemplo, localización basada en GPS como bases de datos con Structured Query Language (SQL) reconocimiento y síntesis de voz, navegador, multimedia, etc

- **Aceptable nivel de seguridad:** Los programas se encuentran aislados unos de otros gracias al concepto de ejecución dentro de una caja, que hereda de Linux. Además, cada aplicación dispone de una serie de permisos que limitan su rango de actuación. Desde la versión 6.0 el usuario puede conceder o retirar permisos a las aplicaciones en cualquier momento.

- **Optimizado para baja potencia y poca memoria:** En el diseño de Android se ha tenido en cuenta el hardware específico de los dispositivos móviles. Por ejemplo, Android utiliza la máquina virtual ART (o Dalvik en versiones antiguas). Se trata de una implementación de Google de la máquina virtual Java optimizada para dispositivos móviles

- **Alta calidad de gráficos y sonido:** Gráficos vectoriales suavizados, animaciones, gráficos en 3D basados en Open Graphics Library (OpenGL). Incorpora los codecs estándares más comunes de audio y video, incluyendo H.264 (AVC), MP3 AAC, etc.

5.2. ARQUITECTURA

La arquitectura del sistema Android está formado por cuatro capas como muestra la figura 32. Esto permite que cada capa puede hacer uso de los servicios que prestan las demás. Así, los desarrollos son mucho más sencillos de implementar si el trabajo se ha distribuido previamente en diferentes capas lógicas [23].

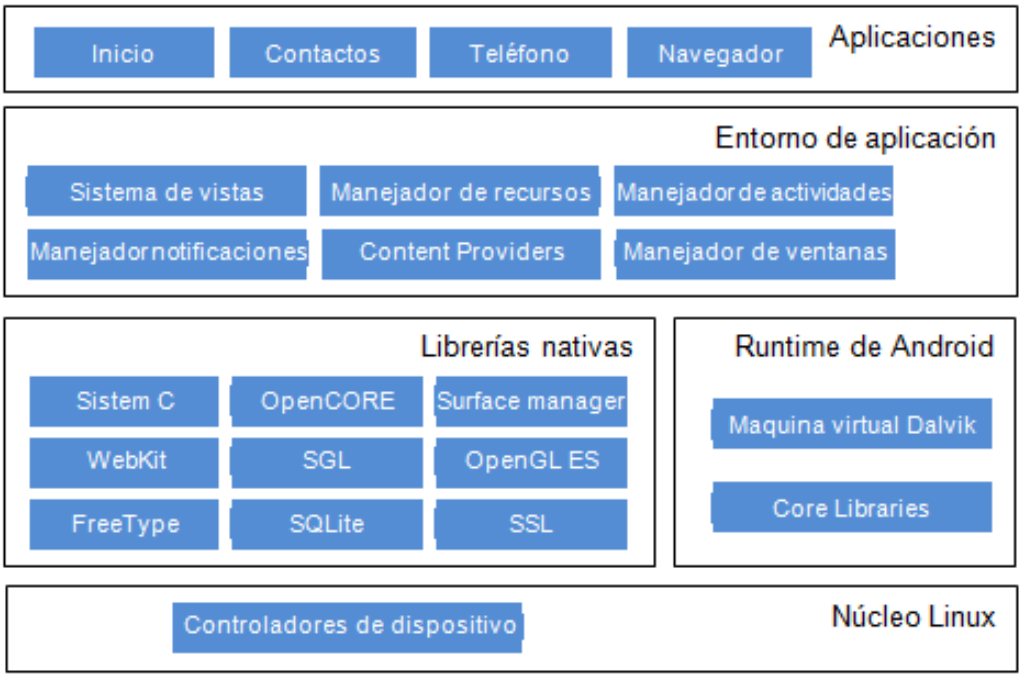


Figura 37 - Arquitectura Android.

Núcleo Linux

El núcleo de Android está formado por el sistema operativo Linux. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de drivers para dispositivos.

Actúa como capa de atracción entre el hardware y el resto de la pila. Por lo tanto, es la única dependiente del hardware.

RunTime de Android (tiempo de ejecución)

Está basado en el concepto de máquina virtual utilizado en java. Debido a las limitaciones de los dispositivos Google creó una maquina nueva para reemplazar a Dalvik.

Entre las características de la maquina Dalvik que facilita la optimización de recursos, sigue un modelo de ejecución de ficheros, consiguiendo que cada aplicación corra su propio proceso de Linux con su propia instancia de la máquina virtual.

A partir de Android 5.0 se reemplaza Dalvik por ART, reduciéndose tiempo de ejecución del código hasta un 33%.

Librerías Nativas

Este sistema operativo incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android. Están compiladas en código nativo del procesador. Estas aportan mejoras en los programas embebidos, dándoles códigos de programa menos extensos, aumentando su eficiencia.

Entre las más importantes se puede encontrar las siguientes:

System C librrary: Derivación de la librería BSD de C.

Media Framework: Librería basada en OpenCORE[24] de PAcketVideo.

Webkit: Soporta el navegador web utilizado en Android y en la vista.

SQLite: Potente y ligero motor de bases de datos.

Entorno de aplicación

Proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones (sensores, localización, servicios, barra de notificaciones, etc).

Esta capa ha sido diseñada para simplificar la reutilización de componentes.

Aplicaciones

Este nivel está formado por el conjunto de aplicaciones instaladas en una máquina Android. Todas las aplicaciones han de correr en la máquina virtual Dalvik para mayor seguridad.

Normalmente, las aplicaciones Android están desarrolladas en java, y para poder desarrollar aplicaciones para el sistema podemos utilizar el Android SDK [23].

5.3. DESARROLLO APLICACIÓN ANDROID

INTRODUCCIÓN

Esta aplicación conecta inalámbricamente a dicha prótesis con un módulo bluetooth para poder dotarla de movimientos y para visualizar la respuesta de los sensores que están incorporados.

Para el desarrollo de aplicaciones se utiliza una arquitectura multicapa, en la que se debe diferenciar una capa de diseño y otra capa lógica. En la interfaz de usuario (capa de diseño), los elementos de la aplicación interactúan con las clases necesarias (capa lógica) para mostrar un tipo de interfaz o efectuar la lógica necesaria.

INTERFAZ DE LA APLICACIÓN

La vista es un objeto que se puede dibujar y se utiliza como un elemento en el diseño de la interfaz de usuario.

Para construir la interfaz de usuario se emplean diferentes objetos, todos ellos descendientes de la clase View. Dentro de esta clase tenemos los botones o etiquetas y los llamados ViewGroup, que es una extensión a View siendo éstos la base de los layouts.

Para la capa de diseño tenemos los elementos de tipo vista mencionados anteriormente, que para combinar varios de éstos tenemos los objetos de tipo layout. Un layout es un contenedor de una o más vistas y controla su comportamiento y posición. Hay que destacar que un layout puede contener otro layout y que es descendiente de la clase *View*.

Los layout más importantes son:

- **LinearLayout**: Dispone los elementos en una fila o en un una columna.
- **TableLayout:** Distribuye los elementos de forma tabular.
- **RelativeLayout:** Dispone los elementos con relación a otro o al padre.
- **AbsoluteLayout:** Posiciona los elementos de forma absoluta.
- **FrameLayout:** Permite el cambio dinámico de los elementos que contiene.

Un GridView es una AdapterView capaz de organizar datos en forma de cuadrícula para mejorar la accesibilidad del usuario. A diferencia de un ListView, este contenedor permite scrolling horizontal y vertical en sus interacciones.

Los elementos que más se utilizan en la aplicación son:

- **TextView** *:* Son denominados “etiquetas de texto”, ya que se utilizan como medio de salida, es decir para mostrar un determinado texto al usuario. Además, permite realizar modificaciones en cuanto a la estética del texto como cursiva, negrita, tamaño, etc.

- **ButtonView** : Es la representación de un botón corriente con un texto. En el caso de implementar una imagen asociada a dicho botón, debe definirse ImageButton. La forma más habitual de interactuar con un botón es pulsarlo para desencadenar un evento. Para que nuestra aplicación realice una determinada acción cuando sea pulsado el botón, debemos implementar un 45 manejador para el evento OnClick. Se presentan varias alternativas a la hora de definir botones, una de ellas es que permite desplegar varias opciones a partir del objeto MenuItem.

- **SeekBar**: Este widget permite al usuario seleccionar un valor dentro de un rango simplemente desplazando un pequeño botón o “thumb” a lo largo de una barra o bien pulsando directamente en cualquier punto de la misma. Los valores seleccionables van desde cero hasta el que definamos mediante el atributo android:max que por defecto es 100. Es imprescindible definir en el código XML las propiedades *paddingRight* y *paddingLeft*para que el thumb no se oculte parcialmente cuando se ubique en un extremo la barra

- **GraphView:** Es un archivo de diseño XML que puede crear gráficos a partir de datos. Para usar este elemento es necesario añadir la librería GraphView donde se añadirán las clases necesarias para poder implementar este tipo de gráfico.

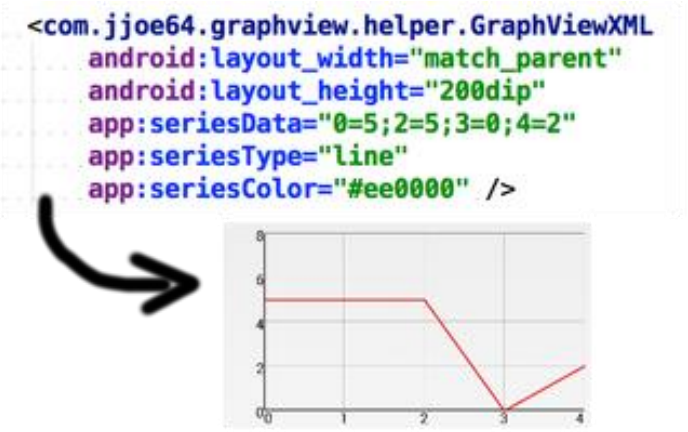


Figura 38 - Graph View.

ACTIVIDADES DE LA APLICACIÓN

Lo primero a tener en cuenta en Android es el ciclo de vida de una actividad.

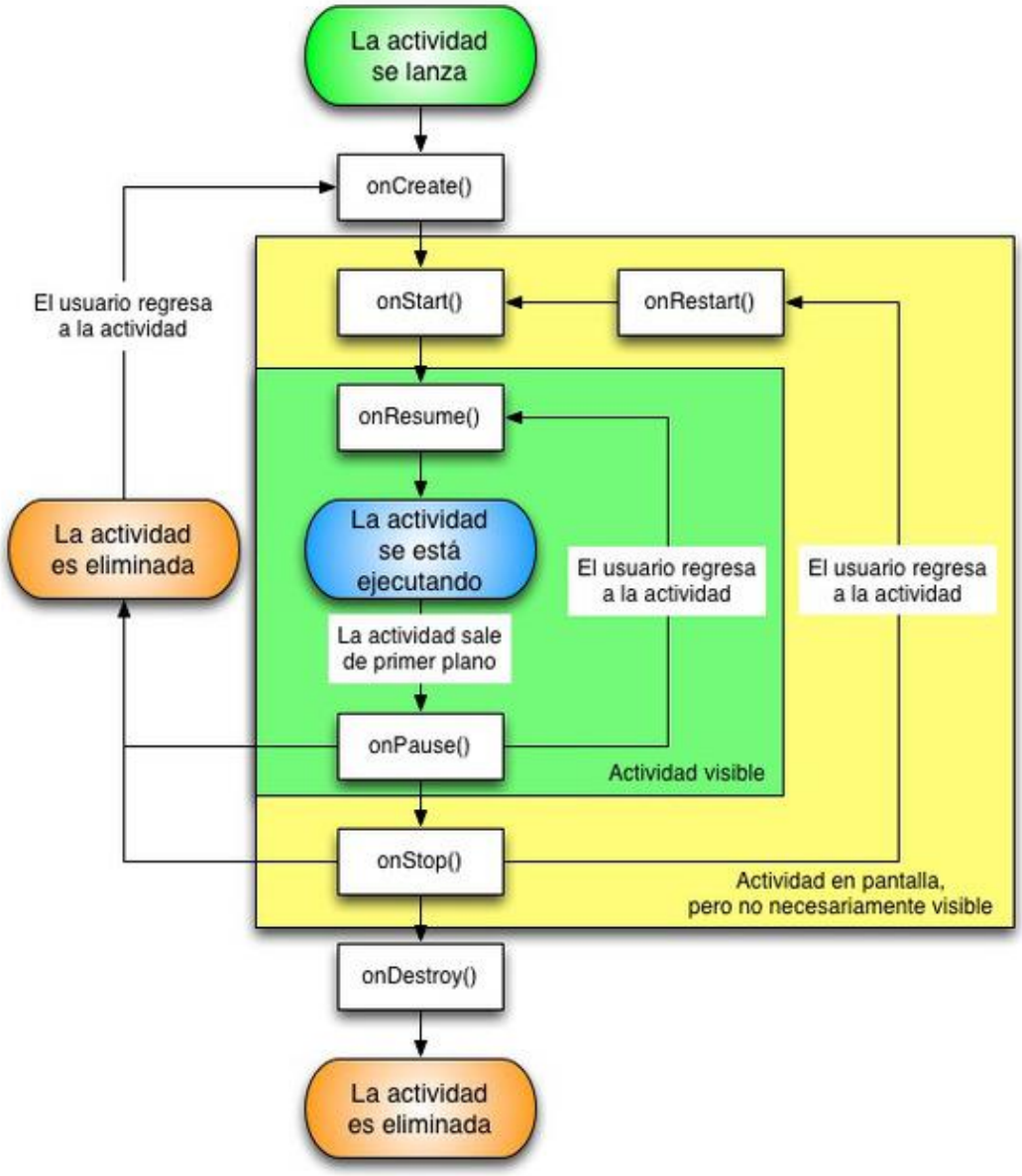


Figura 39 - Ciclo de vida de una actividad.

El ciclo de vida de una aplicación Android es bastante diferente al ciclo de vida de una aplicación en otros S.O., como Windows. La mayor diferencia es que, en Android el ciclo de vida es controlado principalmente por el sistema, en lugar de ser controlado directamente por el usuario.

Son las actividades las que realmente controlan el ciclo de vida de las aplicaciones, dado que el usuario no cambia de aplicación, sino de actividad. El sistema mantiene una pila con las actividades previamente visualizadas, de forma que el usuario puede regresar a la actividad anterior pulsando la tecla “retorno”.

El ciclo de vida de una aplicación Android es bastante diferente al ciclo de vida de una aplicación en otros S.O., como Windows. La mayor diferencia es que, en Android el ciclo de vida es controlado principalmente por el sistema, en lugar de ser controlado directamente por el usuario.

Una actividad en Android puede estar en uno de estos cuatro estados:

- **Activa** (*Running*): La actividad está encima de la pila, lo que quiere decir que es visible y tiene el foco.

- **Visible** (*Paused*): La actividad es visible pero no tiene el foco. Se alcanza este estado cuando pasa a activa otra actividad con alguna parte transparente o que no ocupa toda la pantalla. Cuando una actividad está tapada por completo, pasa a estar parada.

- **Parada** (*Stopped*): Cuando la actividad no es visible. El programador debe guardar el estado de la interfaz de usuario, preferencias, etc.

- **Destruida** (*Destroyed*): Cuando la actividad termina al invocarse el método *finish()*, o es matada por el sistema.

Cada vez que una actividad cambia de estado se van a generar eventos que podrán ser capturados por ciertos métodos de la actividad. A continuación se muestra un esquema que ilustra los métodos que capturan estos eventos.

- **onCreate(Bundle)**: Se llama en la creación de la actividad. Se utiliza para realizar todo tipo de inicializaciones, como la creación de la interfaz de usuario o la inicialización de estructuras de datos. Puede recibir información de estado dela actividad (en una instancia de la clase Bundle), por si se reanuda desde una actividad que ha sido destruida y vuelta a crear.

- **onStart()**: Nos indica que la actividad está a punto de ser mostrada al usuario.

- **onResume()**: Se llama cuando la actividad va a comenzar a interactuar con el usuario. Es un buen lugar para lanzar las animaciones y la música.

- **onPause()**: Indica que la actividad está a punto de ser lanzada a segundo plano, normalmente porque otra actividad es lanzada. Es el lugar adecuado para detener animaciones, música o almacenar los datos que estaban en edición.

- **onStop()**: La actividad ya no va a ser visible para el usuario. Ojo si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método.

- **onRestart()**: Indica que la actividad va a volver a ser representada después de haber pasado por *onStop()*.

- **onDestroy()**: Se llama antes de que la actividad sea totalmente destruida. Por ejemplo, cuando el usuario pulsa el botón de volver o cuando se llama al método *finish()*. Ojo si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método.

La estructura básica de la aplicación está basada en 3 actividades:

- Inicio
- BuscarDispositivos
- ActividadPrincipal

Inicio

Esta es la actividad principal que se mostrará tras arrancar la aplicación y cargará nuestro layout donde muestra una pantalla de inicio que dura 3 segundos. Seguidamente, lo que realiza esta actividad es comprobar si el dispositivo que queremos utilizar dispone de bluetooth, sino se dispone saltará una alerta informando de ello y la aplicación se cerrará. Si el dispositivo dispone de bluetooth pero esta deshabilitado, dará la opción de habilitarlo para poder seguir el ciclo de la actividad.

```
/Comprobamos que el dispositivo tiene bluetooth
myBluetooth = BluetoothAdapter.getDefaultAdapter();

if(myBluetooth == null)
{
    //Mostramos un mensaje, indicando al usuario que no tiene conexión
bluetooth disponible
    Toast.makeText(getApplicationContext(), "Bluetooth no disponible",
Toast.LENGTH_LONG).show();

    //finalizamos la aplicación
    finish();
}
else if(!myBluetooth.isEnabled())
{
        //Preguntamos al usuario si desea encender el bluetooth
        Intent turnBTon = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(turnBTon,1);
}
```

Otra cosa que se ha tenido en cuenta es la de deshabilitar el bluetooth cuando se sale de la aplicación para el ahorro de batería evitándolo dejar conectado.

```
public void onDestroy() {
    super.onDestroy();
    bluetooth.disable();
    finish();
}
```

Al final de la pantalla está el botón el cual al pulsarlo pasaremos a BuscarDispositivos a través de un intent.

```
// Pasar a la siguiente activity.
Intent i = new Intent(Inicio.this, BuscarDispositivos.class);
//cambio de activity.
startActivity(i);
```

BuscarDispositivos

Esta función se encargará de descubrir o buscar todos los dispositivos bluetooth que están vinculados a nuestro dispositivo.

Cuando el dispositivo está vinculado, se obtiene el nombre y la dirección MAC y es presentado en una lista. De lo contrario, al no encontrar dispositivos, la aplicación volverá a reiniciarse.

```
dispVinculados = myBluetooth.getBondedDevices();
ArrayList list = new ArrayList();

if (dispVinculados.size()>0)
{
    for(BluetoothDevice bt : dispVinculados)
    {
        list.add(bt.getName() + "\n" + bt.getAddress()); //Obtenemos
los nombres y direcciones MAC de los disp. Vinculados.
    }
}
else
{
    Toast.makeText(getApplicationContext(), "No se han encontrado
dispositivos vinculados", Toast.LENGTH_LONG).show();
Intent intent2 = new Intent();
intent2.setClass(BuscarDispositivos.this, Inicio.class);
startActivity(intent2);

}

//Presenta los disp. Vinculados en una lista.
final ArrayAdapter adapter = new
ArrayAdapter(this,android.R.layout.simple_list_item_1, list);
listaDispositivos.setAdapter(adapter);
listaDispositivos.setOnItemClickListener(myListClickListener)
```

Tras seleccionar el dispositivo correcto cambiaremos a la siguiente activity.

```
public void onItemClick (AdapterView<?> av, View v, int arg2, long
arg3)
{
    // Obtener en la vista los últimos 17 caracteres de la dirección
MAC
    String info = ((TextView) v).getText().toString();
    String address = info.substring(info.length() - 17);

    // intent para iniciar la siguiente activity.
    Intent i = new Intent(BuscarDispositivos.this,
ActividadPrincipal.class);

    //cambio de activity.
    i.putExtra(EXTRA_ADDRESS, address); //Este parámetro será pasado a
la siguiente acticidad.
    startActivity(i);
}
```

Actividad principal

Esta función establece la lógica de funcionamiento entre los fragments establecidos y ejecuta diferentes vistas y funciones establecidas.

En primer lugar, un fragment [23], está formado por la unión de varias vistas para crear un bloque funcional de la interfaz de usuario. Podemos combinar uno o varios fragments dentro de una actividad, según el tamaño de la pantalla disponible. Para cambiar de vista dentro de la actividad principal basta con hacer *Scroll* de izquierda a derecha y viceversa.

Dentro de la actividad principal definimos el menú superior que mostrará el nombre de la vista en la que nos encontramos y los fragments que se usarán. En el siguiente fragmento de código se mostrará la forma de definir los fragments en la actividad principal:

```
public class PagerAdapter extends FragmentPagerAdapter {

    public PagerAdapter(FragmentManager fm) {
        super(fm);
    }

    public Fragment getItem(int arg0) {
        switch (arg0) {
            case 0:
                return new Fragment_Control();
            case 1:
                return new Fragment_Giroscopio();
            case 2:
                return new Fragment_Monitoreo();
```

```
            default:
                return null;
        }
    }
```

Donde el título que mostrará en el menú será:

```
public int getCount() {
    return 3;
}

public CharSequence getPageTitle(int position) {
    String titulo = null;
    switch (position) {
        case 0:
            titulo = "Control";
            break;
        case 1:
            titulo = "Giroscopio MPU6050";
            break;
        case 2:
            titulo = "Monitoreo";
            break;
    }
    return titulo;
}
```

En el fragment control se encontrará el botón de ON (figura 40) que enviará al Arduino el string “AZ” que permitirá asociar el fragment con la función de Arduino correspondiente. Mediante el widget **SeekBar** se realizará el envío de los datos para el movimiento del cierre de la mano.

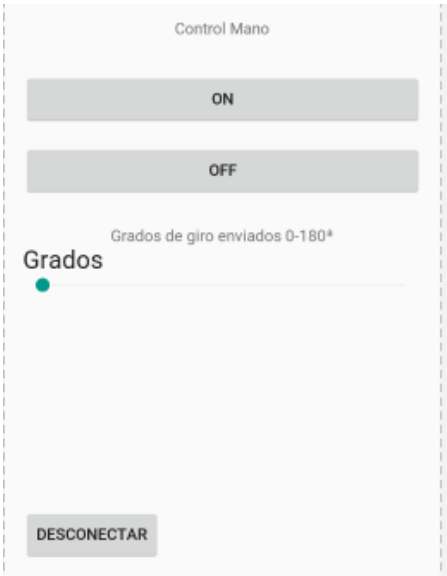


Figura 40 - Fragment control aplicación Android.

El fragment giroscopio (figura 41) recibirá valores procedentes del MPU6050 y dependiendo del movimiento del brazo mostrará una fecha (figura 42) indicando su movimiento.

Figura 41 - Fragent giroscopio aplicación Android.

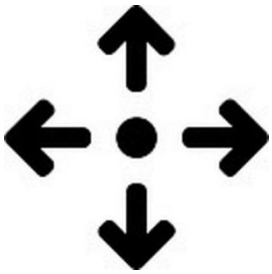
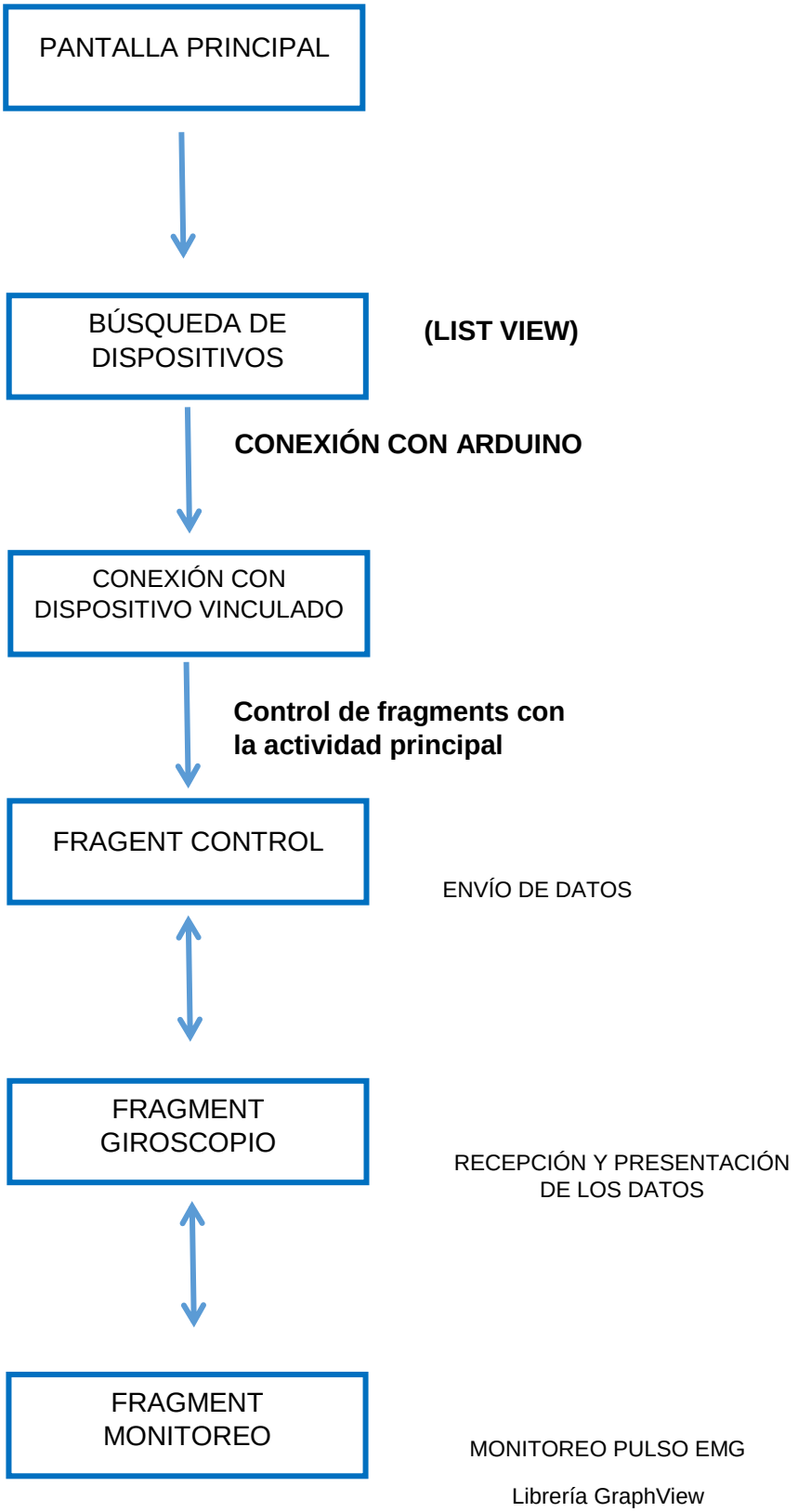


Figura 42 - Flechas posición de fragment giroscopio

En el fragment de monitorización (figura 43), la aplicación Android recibirá los datos procedentes que se han obtenido al realizar el EMG y se verán reflejados en una gráfica a lo largo del tiempo.

Figura 43 - Fragment monitorización aplicación Android.

FUNCIONAMIENTO DE LA APLICACIÓN



CAPÍTULO 6

DISEÑO FINAL

6.1. ENSAMBLADO DEL PROTOTIPO

En este apartado se mostrarán los esquemas de montaje del prototipo y el resultado final del montaje.

Cuando obtenemos con Cura [3] el G-Code correcto con los parámetros adecuados de configuración para cada pieza, se realiza la impresión con nuestra impresora 3D. Cuanto más precisa sea esta configuración, mayor será el tiempo de impresión.

Una vez obtenidas todas las piezas, comenzamos el ensamblado:

En primer lugar tenemos la palma de la mano que está constituida por tres piezas principalmente (figura 44). Para su unión se utiliza un pasador que une la palma completa.

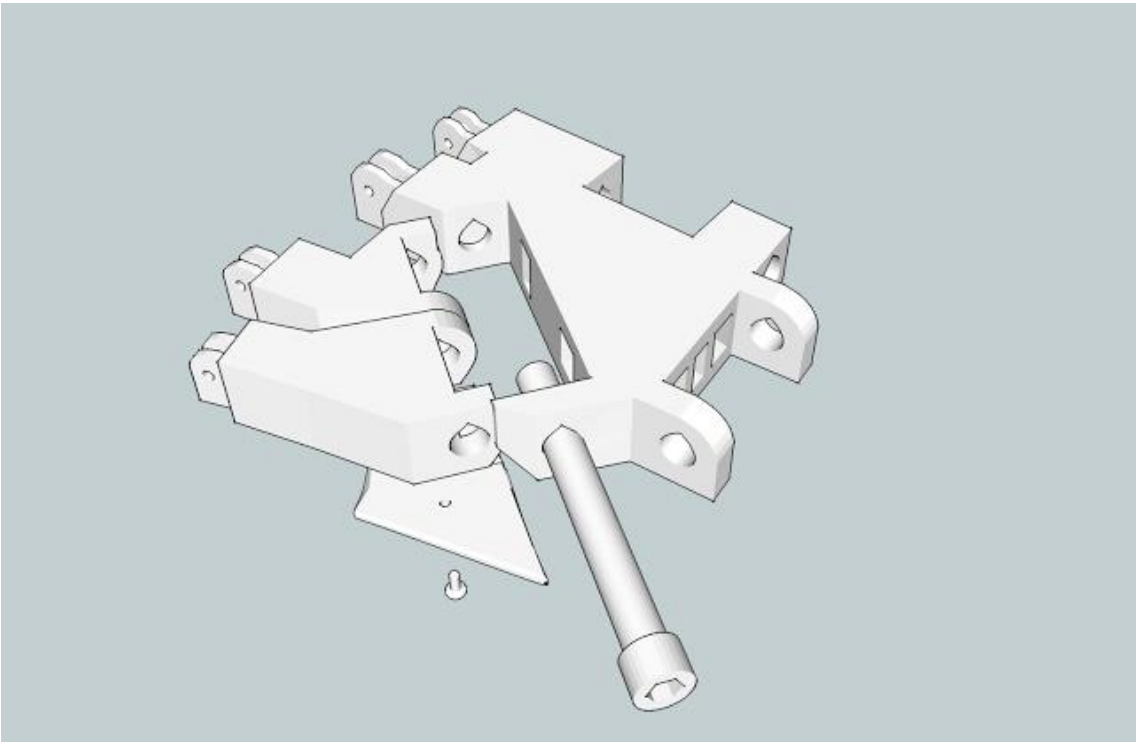


Figura 44 - Ensamblado palma.

Nos encontramos con el problema a la hora de insertar el pasador para la unión, ya que el error de las medidas de impresión varía según la boquilla utilizada para fundir el material en la impresora 3D.

Para corregirlo basta con pasar una broca para pulir los orificios como muestra la figura 45.

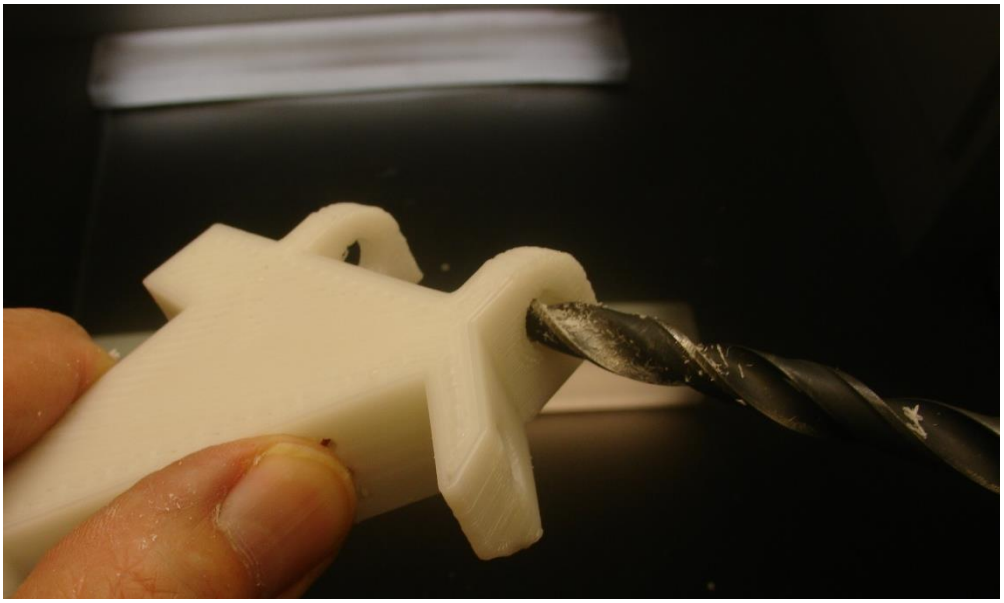


Figura 45 - Errores ensamblado palma de la mano.

En el ensamblado de los dedos, como muestra la figura 46, para el caso del dedo corazón y la figura 47 para el dedo pulgar, cada falange viene en dos piezas. Esto es debido a que si fuera de una sola pieza dificultaría la impresión.

Para unir cada falange se usa un adhesivo de fijado y tornillos haciendo de “bisagra” en cada articulación para su correcto movimiento. El dedo es formado por la unión de todas sus falanges.

Los dedos pulgar, índice y corazón son unidos directamente a la pieza central de la palma de la mano y los dedos y anular y meñique se unen a las piezas unidas anteriormente a la palma.

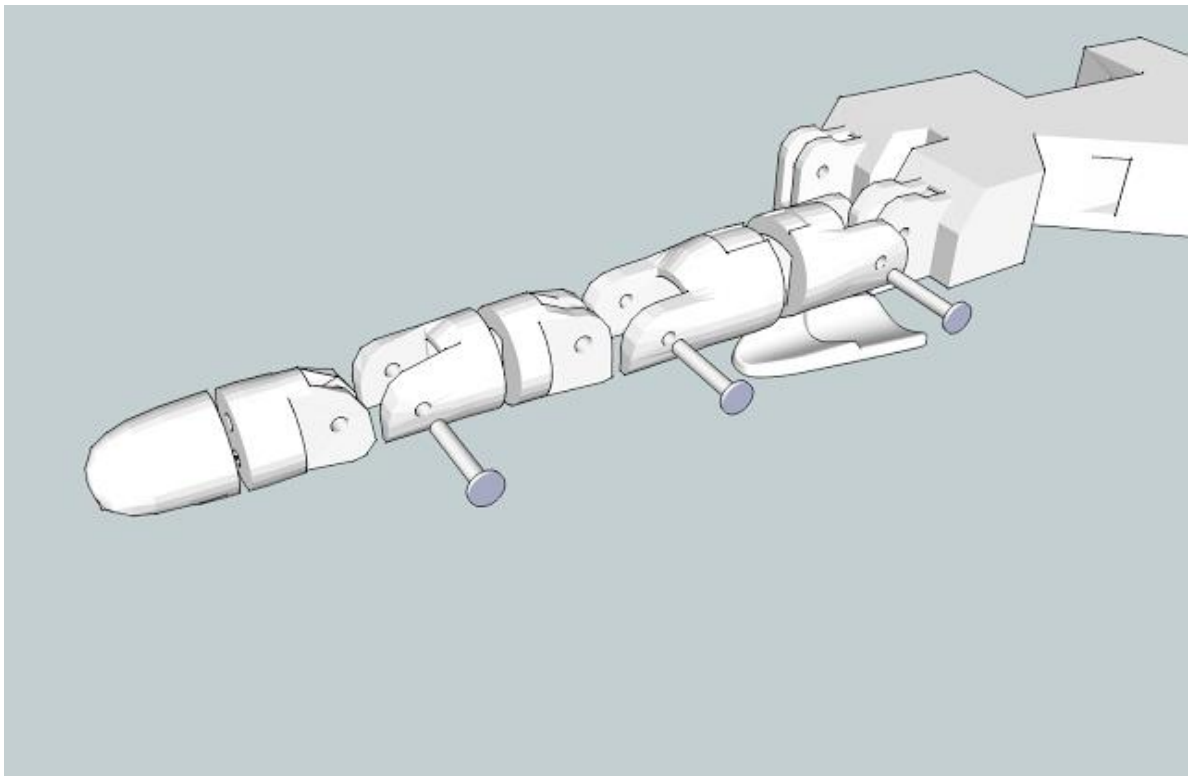


Figura 46 - Ensamblado dedos.

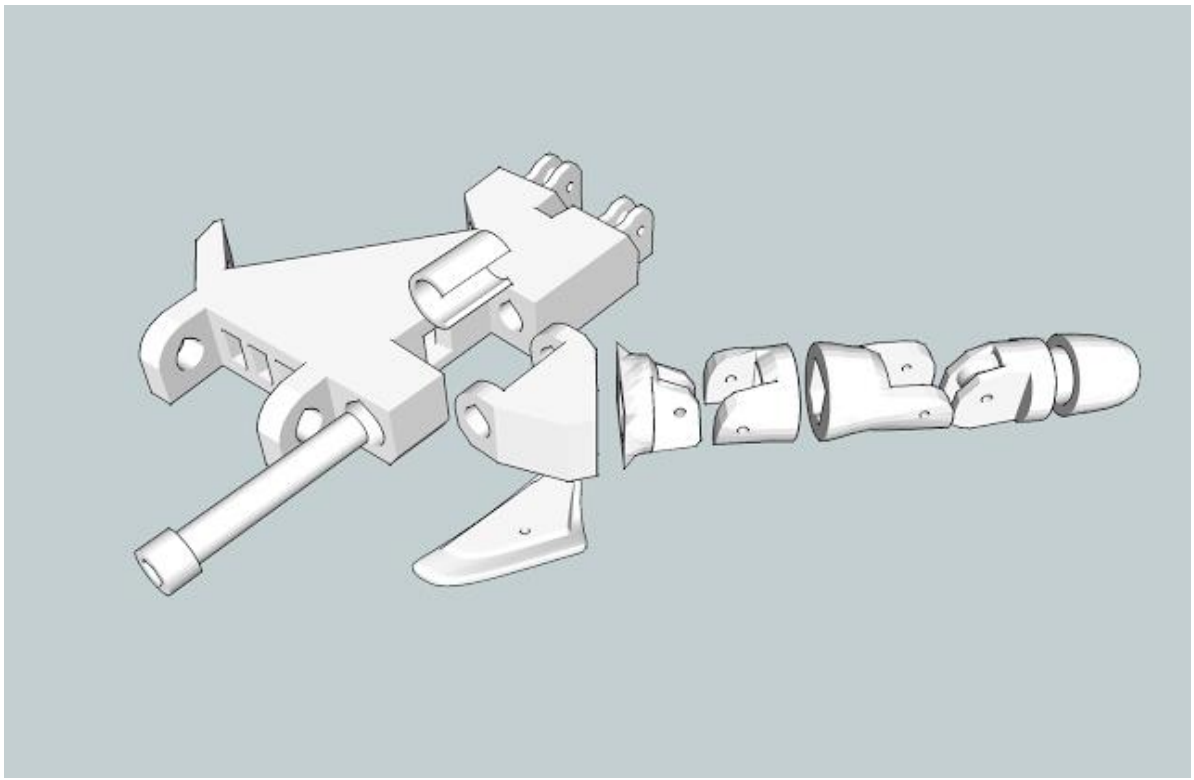


Figura 47 - Ensamblado dedo pulgar.

Los errores encontrados en esta parte del ensamblado son:

- A la hora de unir las falanges se tiene que usar un adhesivo especial para plásticos y extra fuerte para poder soportar grandes fuerzas.

- Pulido de los orificios de los dedos para que se produzca el movimiento “bisagra” correcto.

El ensamblado de la muñeca (figura 48), está constituido por varias piezas. La primera haciendo de soporte a la palma de la mano, que está unida directamente a los engranajes acoplados al servomotor. También cuenta con otras dos piezas de soporte para el servomotor y los engranajes.

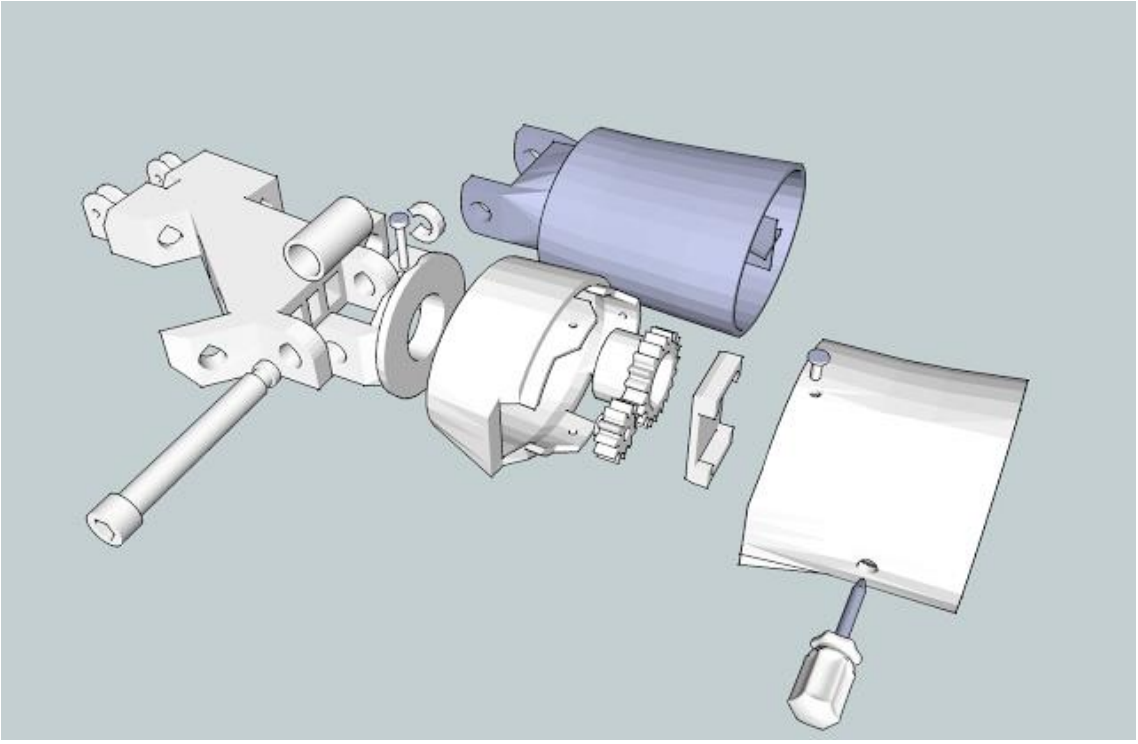


Figura 48 - Ensamblado muñeca.

Lo problemas encontrados al ensamblar la muñeca son:

- Correcta impresión de engranajes, de no ser así hay que lijarlos hasta que tengan las medidas adecuadas.

- Fijación correcta de piezas con tornillos para que no tener holguras.

En la última parte del ensamblado se colocan los servomotores en antebrazo. Estos van a ser los causantes del movimiento de los dedos y del giro de muñeca. En esta parte basta con atornillar los servomotores en las posiciones mostradas en la figura 49.

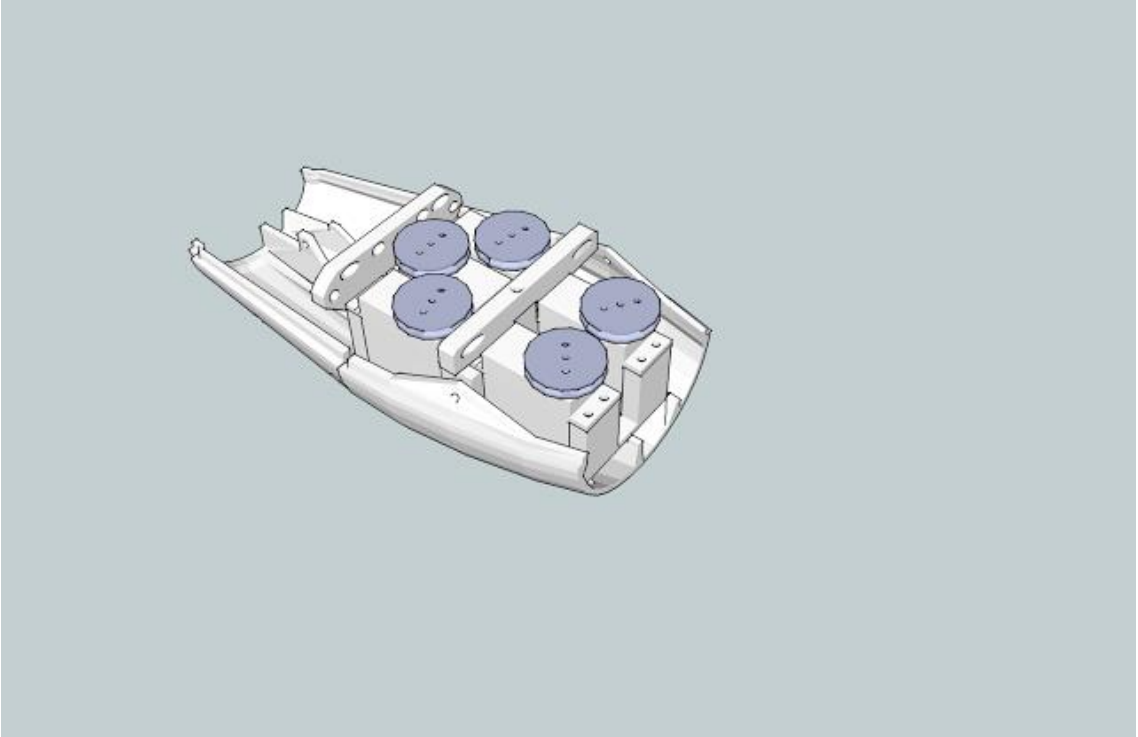


Figura 49 - Ensamblado servos.

Para que se produzca el movimiento de los dedos se han de insertar unos tensores desde los servomotores hasta los dedos. Estos tensores pasarán por los orificios que tiene cada pieza como se muestra en la figura 50.

Cada servomotor va asignado a un dedo y cuenta con dos tensores para el movimiento del dedo, flexionando y estirándolo, dependiendo del sentido del giro del servomotor (figura 51).

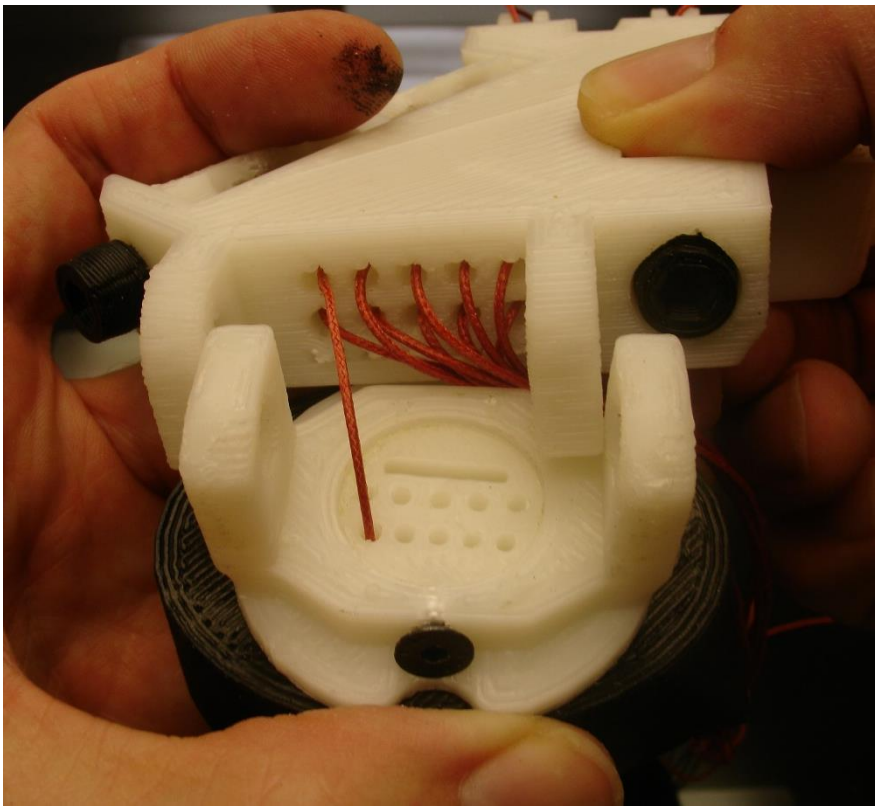


Figura 50 - Ensamblaje, orificios tensores palma mano y muñeca.

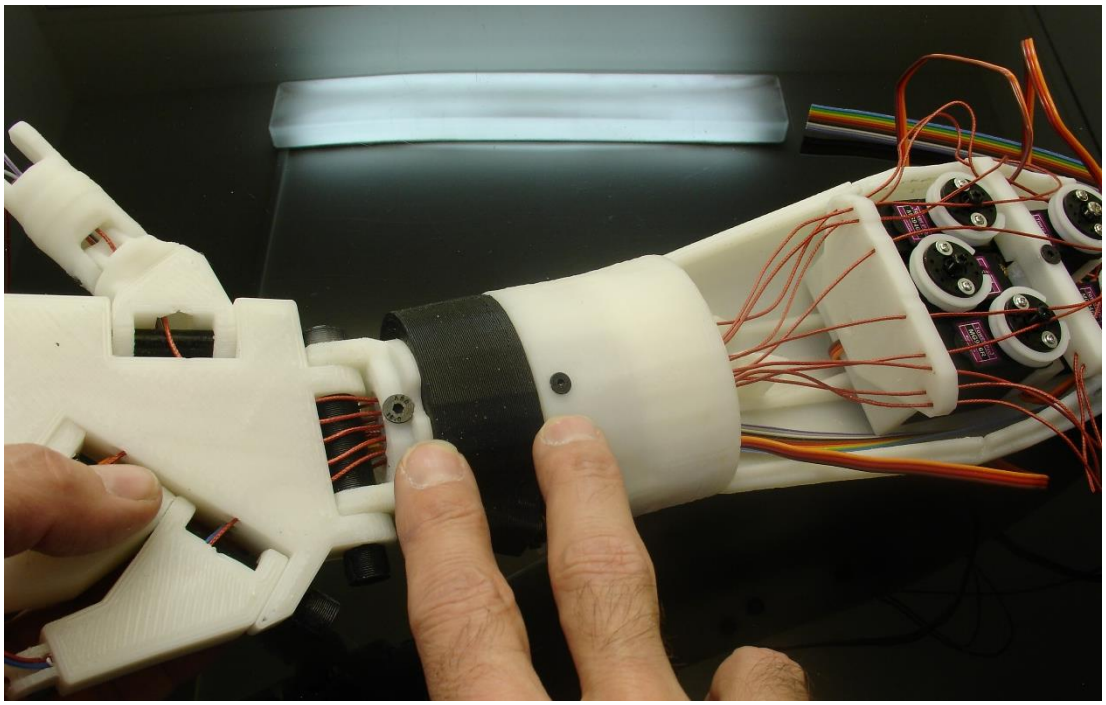


Figura 51 - Ensamblaje mano, tensores.

Figura 52 - Ensamblado final.

6.2. ESQUEMA DE FUNCIONAMIENTO

ELECTRODOS

E-HEALTH

CONEXIÓN SERIAL
USB PUERTO COM

ARDUINO

APLICACIÓN ANDROID

PRÓTESIS 3D

SERVOS

LCD

GIROSCOPIO MPU

CAPÍTULO 7

CONCLUSIÓN Y FUTURAS LÍNEAS DE ACTUACIÓN

7.1. CONCLUSIONES

Se ha realizado la impresión 3D de un prototipo de una mano mioeléctrica capaz de comunicarse con Android. Para ello se ha utilizado una placa de adquisición de datos, giroscopio, módulo bluetooth, un Arduino para procesar toda la información obtenida y actuar en forma de movimiento a través de los servomotores. Este modelo puede ser útil para propósitos docentes de la Escuela Politécnica de Cuenca.

Una vez obtenidos los objetivos deseados se pueden obtener las siguientes conclusiones:

- El potencial que tiene la impresión 3D y la capacidad de obtener cualquier tipo de modelos que se puedan dibujar en 3D.

- Para obtener modelos con buena calidad hay que tener un gran conocimiento sobre los materiales, impresoras 3D y software de impresión.

- Gracias a los conocimientos adquiridos en la asignatura de equipos audiovisuales en electromedicina y a los adquiridos durante el transcurso de esta carrera, me ha sido posible afrontar los problemas a la hora de la adquisición de las señales EMG.

- El potencial que ofrece trabajar con Arduino a bajos costos, siendo multi-plataforma, ofreciendo un entorno de programación sencillo y flexible para usuarios avanzados, con software ampliable y de código abierto.

- Arduino ofrece la posibilidad de usar infinidad de sensores y periféricos para cualquier tipo de desarrollo.

- También, los conocimientos adquiridos en la asignatura de software multimedia me han servido de base para el desarrollo de la aplicación Android, pero he tenido dificultades en el tema de comunicaciones a través de bluetooth.

- El gran potencial del que dispone el sistema operativo Android y la capacidad de comunicarse u obtener datos de otros dispositivos u elementos.

- Gran cantidad de información tanto de Android como de Arduino en Internet.

El conjunto de las conclusiones hace reflexionar sobre la infinidad de soluciones que se pueden dar a los problemas y dificultades de la vida cotidiana, siendo un ejemplo presente este prototipo. Obteniendo materiales de bajo coste al alcance de todos los usuarios y los conocimientos para su desarrollo todo es posible.

7.2. FUTURAS LÍNEAS DE ACTUACIÓN

En un futuro este proyecto se podría ampliar y desarrollar de las diferentes maneras:

- Obtención de servomotores de menor tamaño y mayor potencia.
- Un diseño del modelo más reducido y adaptable al brazo.
- Incorporación de sensores Fisio Runners (FSR) que capten a tiempo real la presión para poder adaptar el movimiento de los servos. De esta manera se podrían parar los servomotores a una determinada presión cuando se quieran coger objetos más sensibles.
- Diseño de un shield para Arduino, en el cual ya estuvieran las conexiones realizadas reduciendo el tamaño de la electrónica.

Implementar un servidor web al que envíe Arduino la toma de datos de un EMG, especificando cada usuario el grado de discapacidad y los músculos en los que se toma el EMG. Todo esto para facilitar información a otros usuarios de la mejor colocación de los electrodos y la obtención óptima de la señal.

REFERENCIAS BIBLIOGRÁFICAS

[1] Sergio Gómez González, **"Impresión 3D"**, [2016].

[2] reprap.org , "Proyecto RepRap", http://www.reprap.org/wiki/RepRap/es, [Visitado Julio 2017]

[3] ultimaker.com , software Cura, https://ultimaker.com/en/products/cura-software [Visitado Julio 2017]

[4] Código abierto, https://opensource.org/, [Visitado Julio 2017]

[5] Materiales impresión 3D, https://www.mediatrends.es/a/40090/mejores-materiales-imprimir-3d/, [Visitado Julio 2017]

[6] Errores impresión 3D, https://www.simplify3d.com/support/print-quality-troubleshooting/, [Visitado Julio 2017]

[7] Comunidad Commons Creative, https://creativecommons.org/

[8] https://3dprintingindustry.com/, [Visitado Julio 2017]

[9] https://www.3dhubs.com/, [Visitado Julio 2017]

[10] https://www.thingiverse.com/, [Visitado Julio 2017]

[11] http://www.soliforum.com/, [Visitado Julio 2017]

[12] Comunidad InMoov, https://inmoov.fr/, [Visitado Julio 2017]

[13] Músculos, https://definicion.de/musculos/, [Visitado Julio 2017]

[14] Anatomía relacionada con EMG, https://robologs.net/2016/02/11/emg-con-arduino-y-e-health-sensor-platform-parte-i-leer-los-electrodos/, [Visitado Julio 2017]

[15] Electromiografía, https://es.wikipedia.org/wiki/Electromiograf%C3%ADa, [Visitado Julio 2017]

[16] Cooking Hacks, https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical, [Visitado Julio 2017]

[17] Paolo Aliverti, "***Manual de Arduino"***, [2016].

[18] Arduino, https://www.arduino.cc/, [Visitado Julio 2017]

[19] Librería VarSpeedServo, https://github.com/netlabtoolkit/VarSpeedServo, [Visitado Julio 2017]

[20] Librería I2C, https://github.com/jrowberg/i2cdevlib, [Visitado Julio 2017]

[21] Protocolo I2C, https://www.i2cdevlib.com/, [Visitado Julio 2017]

[22] Librería MPU 6050, https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050, [Visitado Julio 2017]

[23] Jesús Tomás Girones, **"El Gran Libro Del Android"**, [2017}

[24] OpenCore, https://opencores.org/, [Visitado Julio 2017]

PARTE 2

II- PLANOS

1 PLANOS

En esta parte se muestra el diseño de las aplicaciones desarrolladas y los diagramas de flujo de las funciones firmware utilizado en el código del sistema.

Leyenda

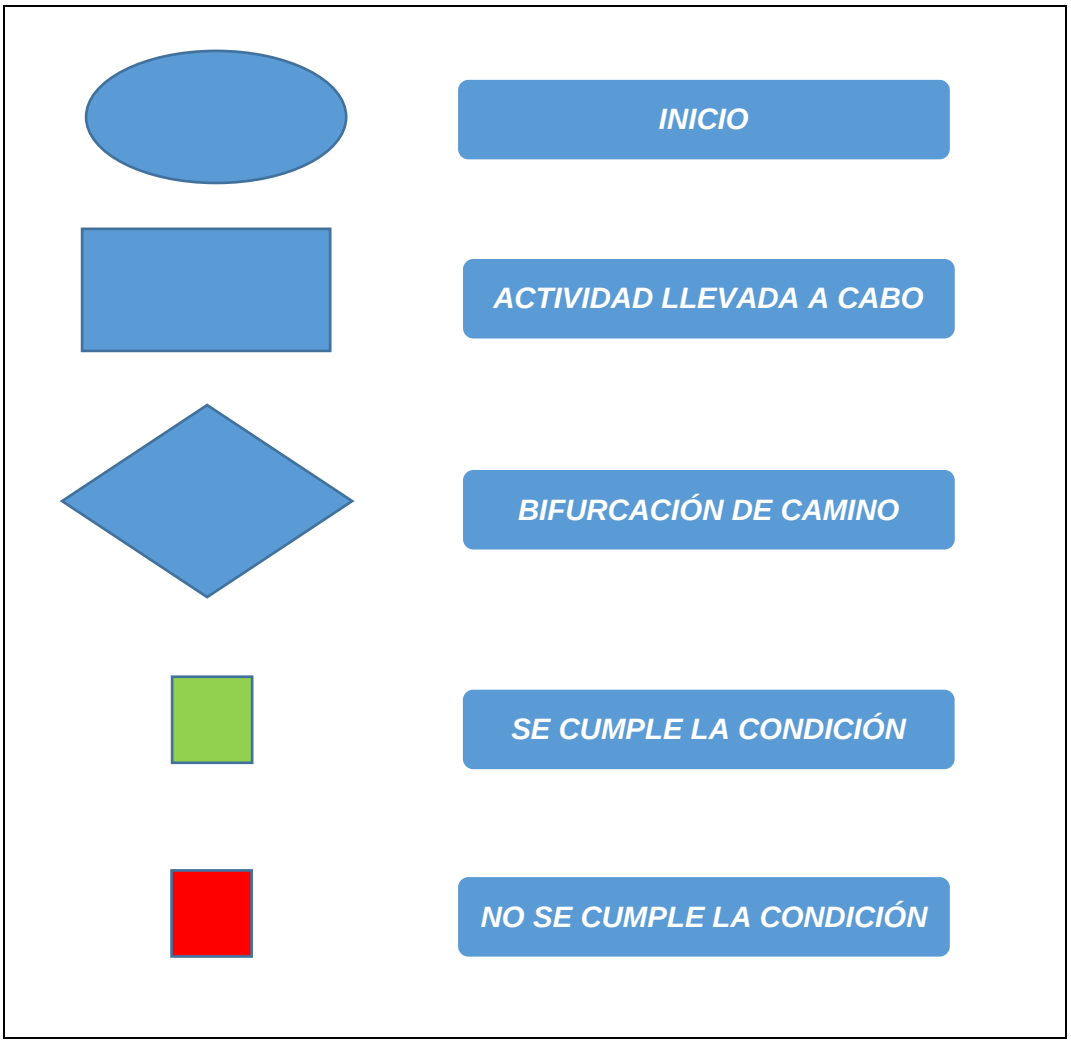


Diagrama de flujo de la aplicación Arduino.

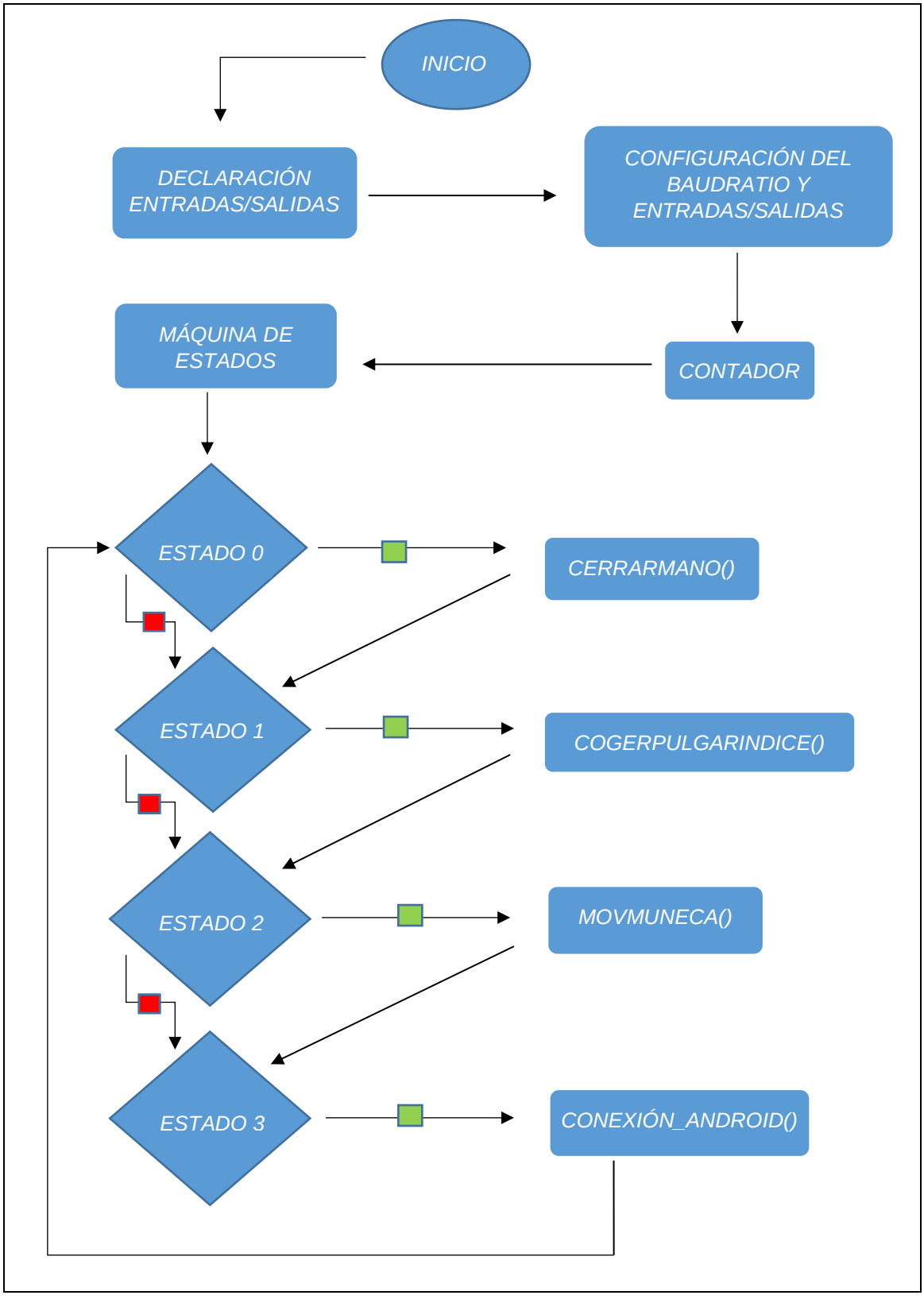


Diagrama de flujo de la aplicación Android, actividad Inicio.

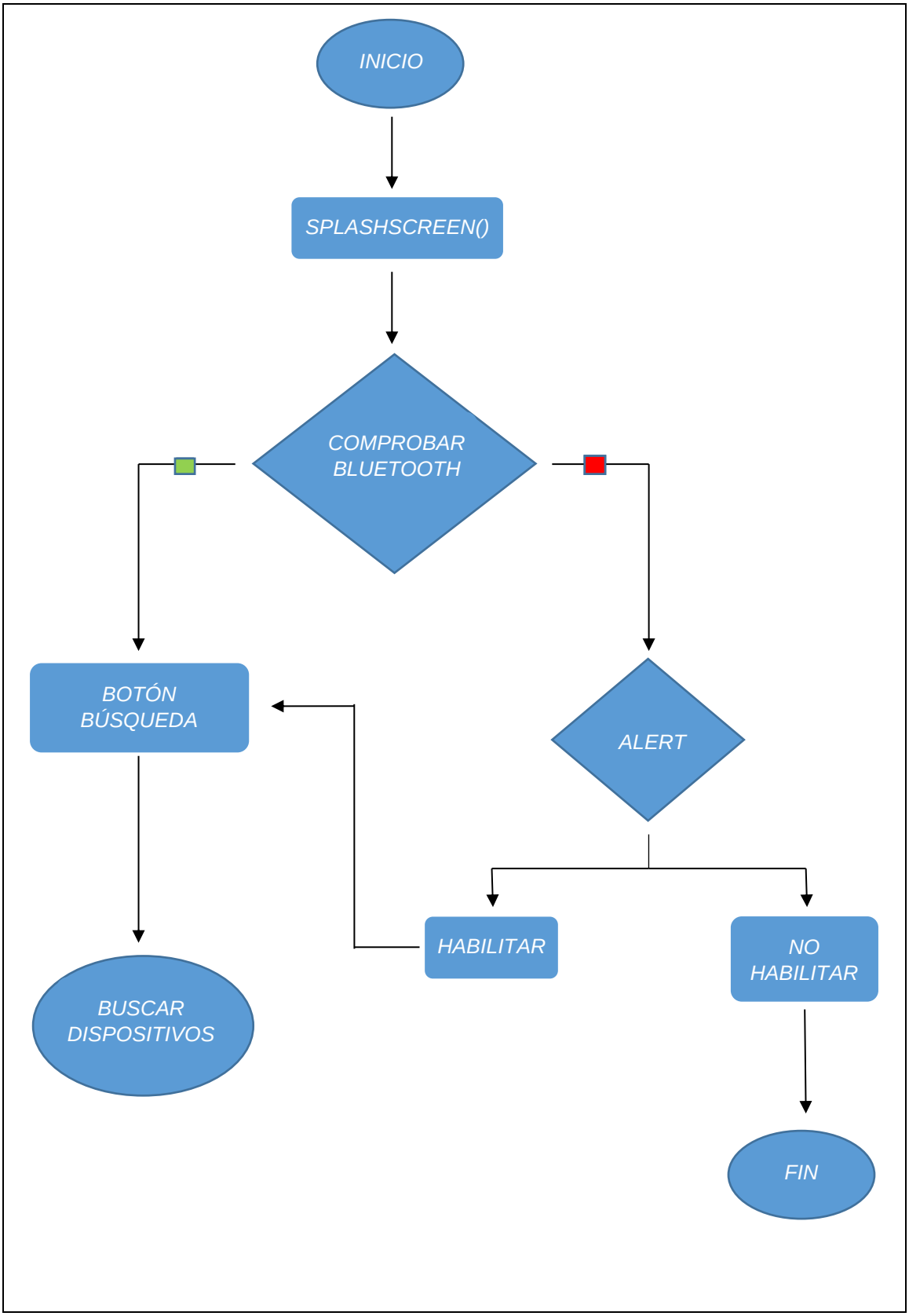


Diagrama de flujo de la aplicación Android, actividad BuscarDispositivos.

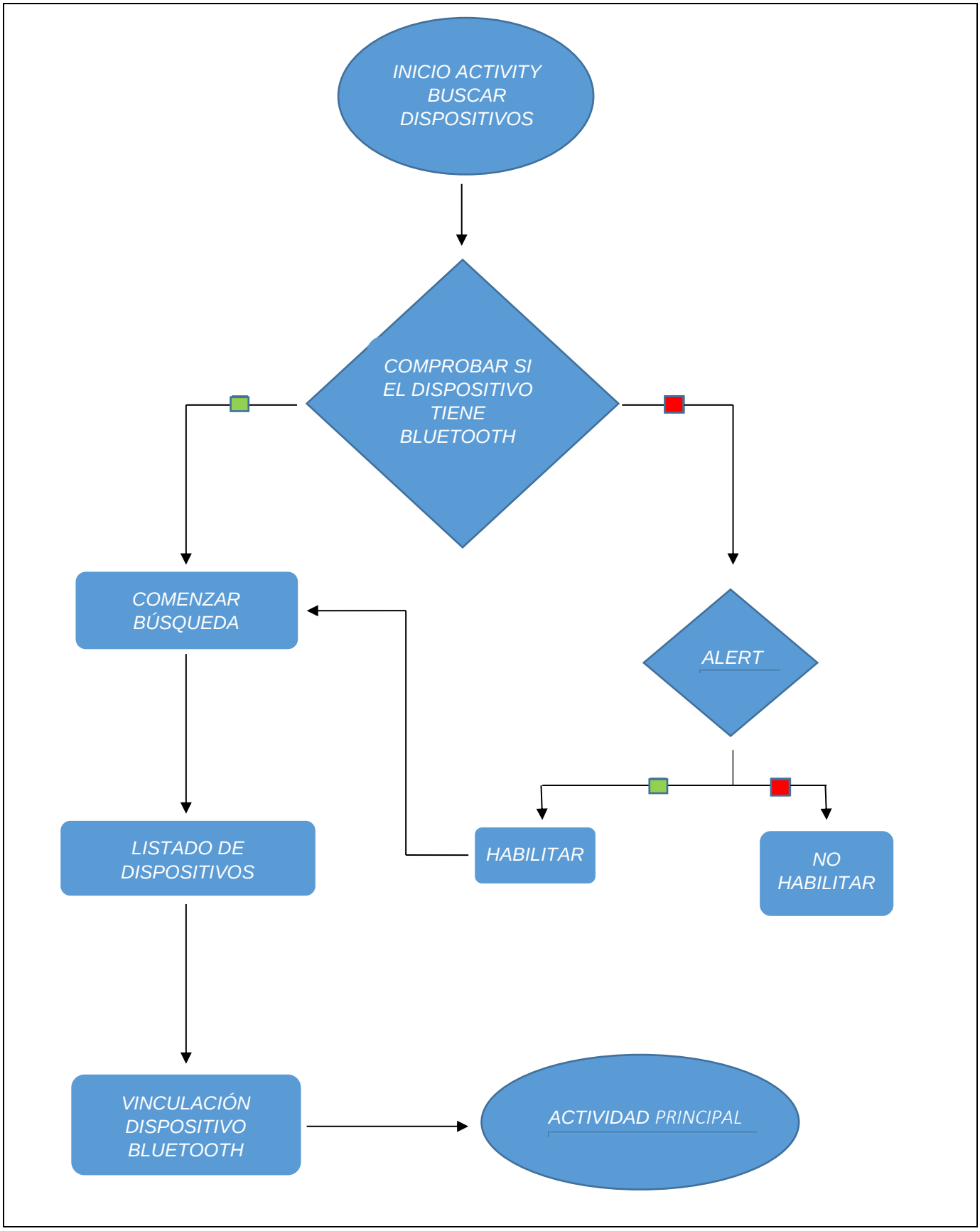
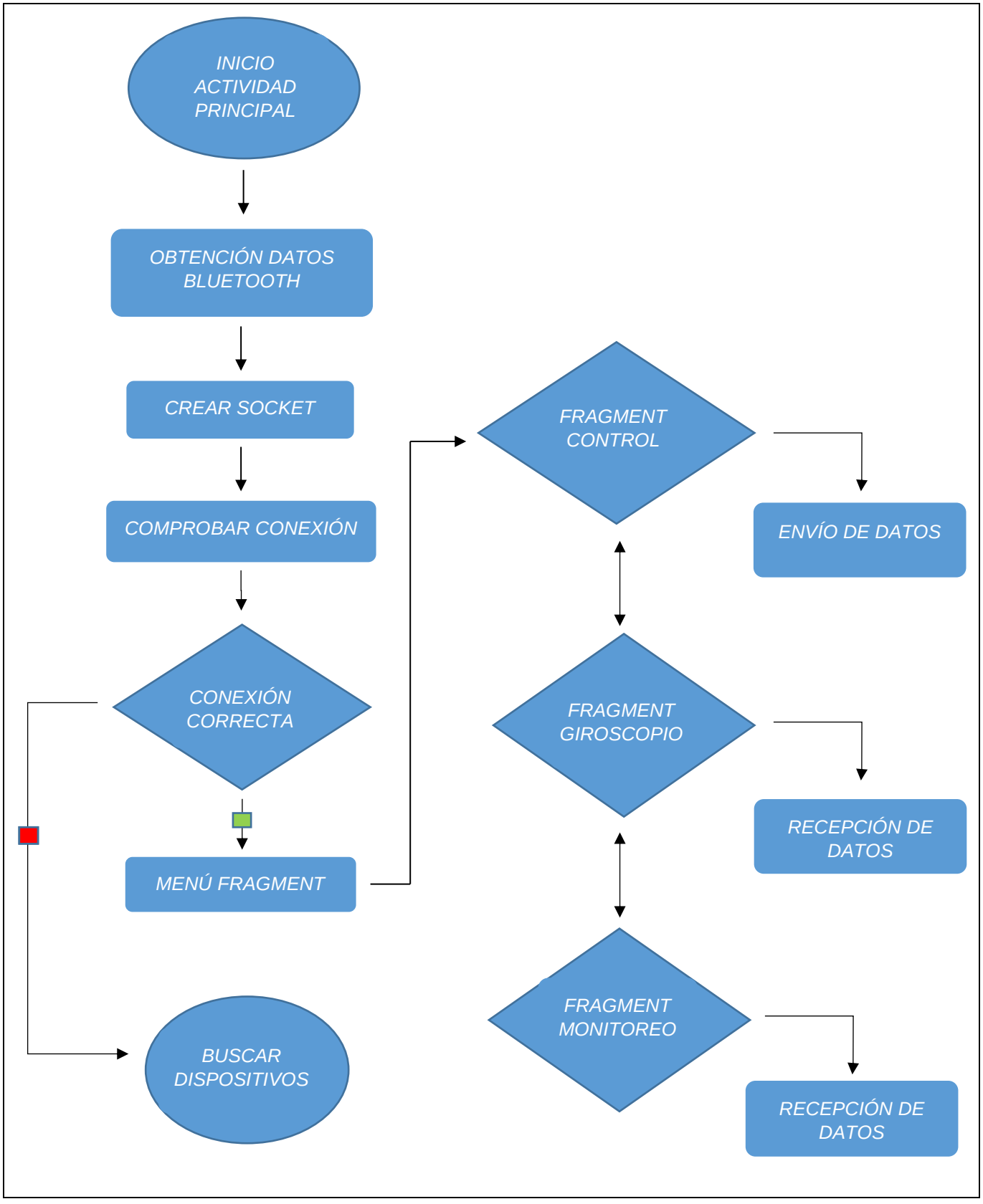
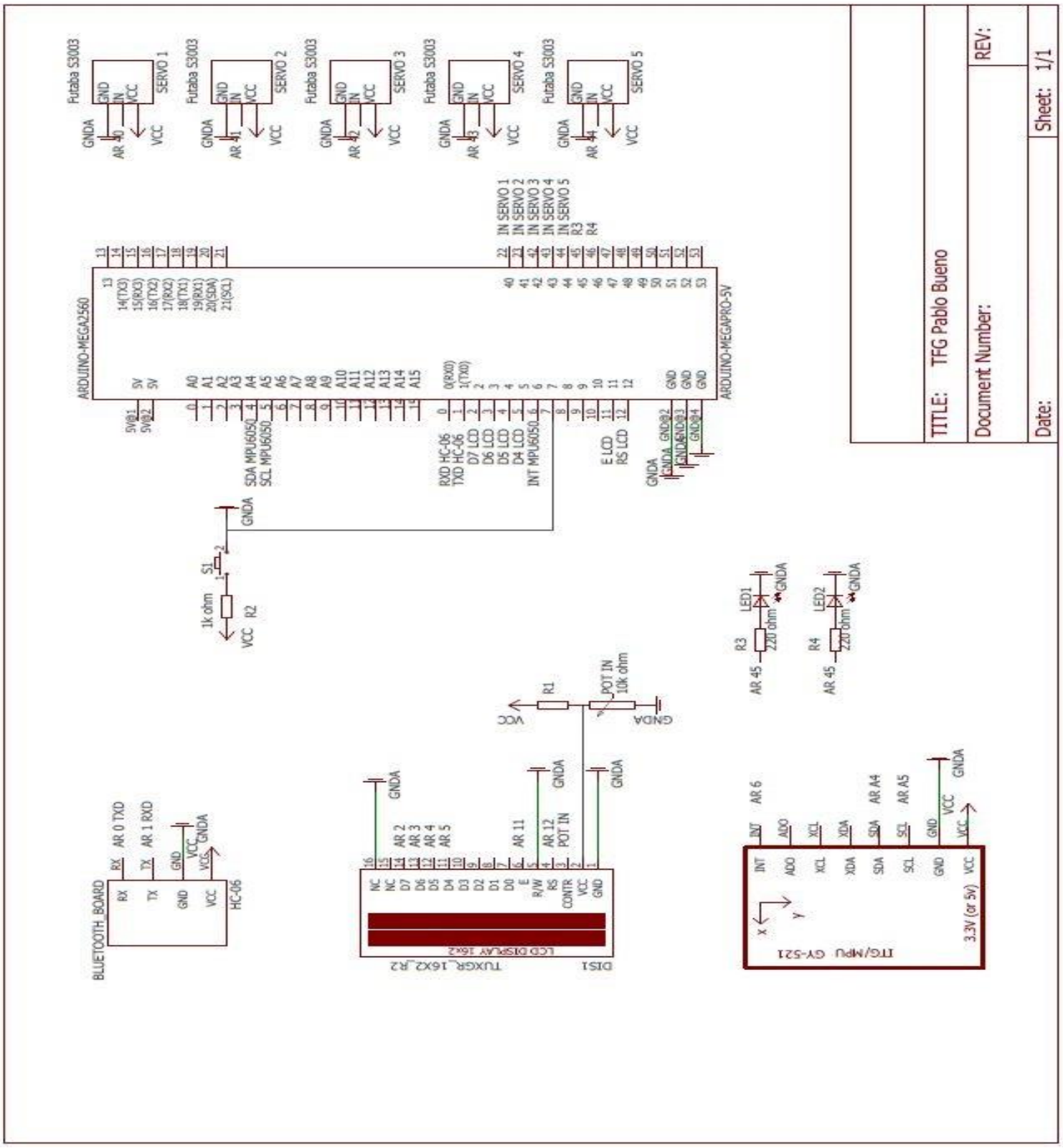


Diagrama de flujo de la aplicación Android, actividad principal.

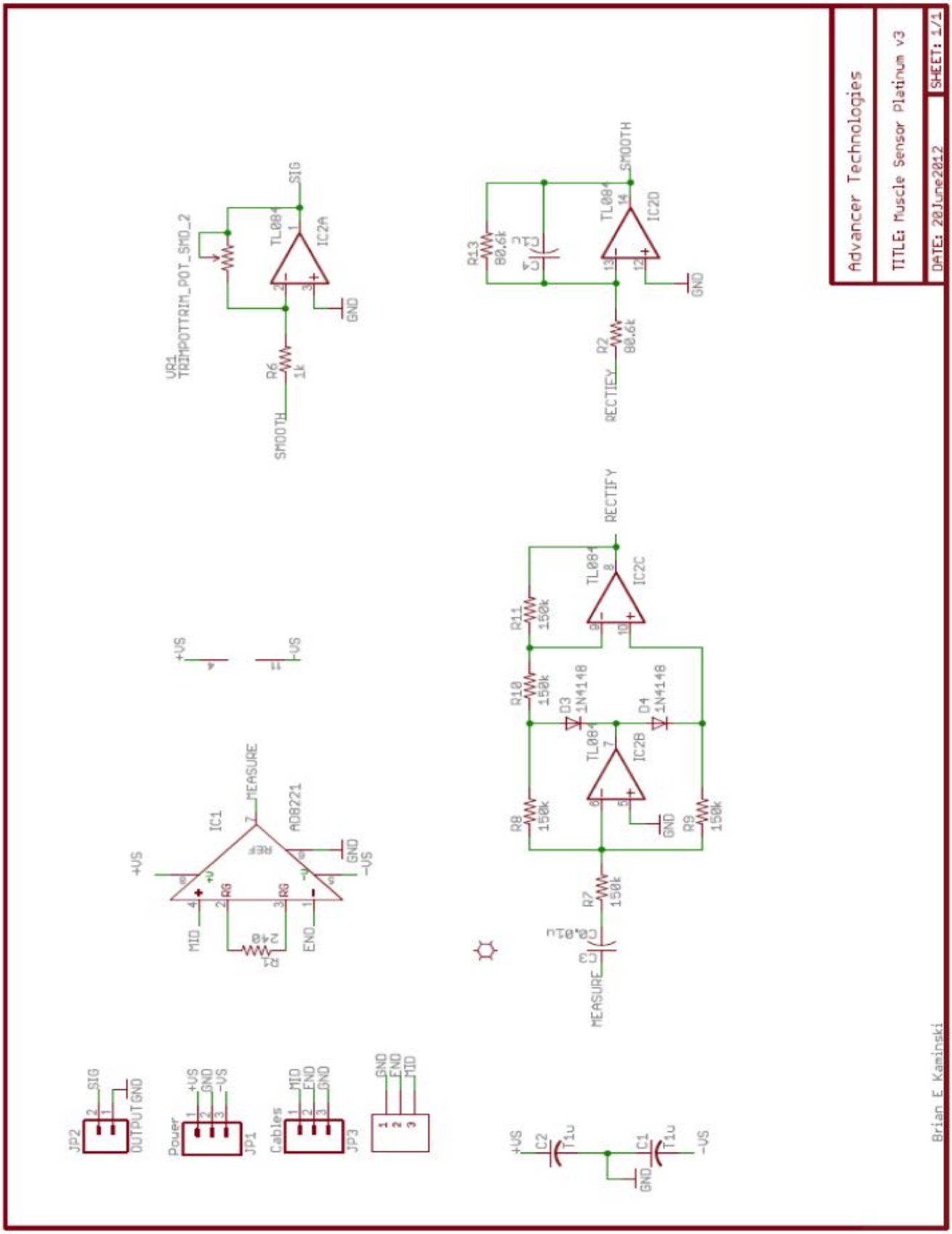


2 ESQUEMAS ELÉCTRICOS

Esquema eléctrico del montaje



Esquema circuido de acondicionamiento del EMG



PARTE 3

III- PLIEGO DE CONDICIONES

1. CARACTERÍSTICAS TÉCNICAS DE LOS EQUIPOS UTILIZADOS

En el presente apartado se van a plantear las especificaciones del hardware utilizado además de las características del software utilizado, dividiéndolo en dos grandes bloques: hardware y software.

REQUISITOS HARDWARE

Impresora 3D Prusa i3

- Fabricante: bq
- Tecnología: FDM
- Materiales: PLA, HIPS - Poliestireno de alto impacto
- Tamaño máximo de impresión: 215x210x180 mm
- Extrusor: 1
- Tipo de extrusor: bq Witbox
- Tamaño de filamento: 1,75 mm
- Diámetro de boquilla: 0,40 mm
- Espesor de capa: 60 - 300 micras
- Base calefactable
- Tipos de archivo: .stl, gcode
- Conectividad: USB, SD Card
- Firmware: Marlin
- S.O. compatibles: Windows, Mac, Linux
- Open source
- Peso: 9,00 kg
- Tamaño de la impresora: 460x370x510 mm
- Alimentación y consumo: 220 AC 12 DC 100W

Arduino Mega6050 R3

- Microcontrolador: ATmega6050
- Voltaje de funcionamiento: 5V
- Voltaje de entrada (recomendado): 7-12V
- Voltaje de entrada (limites): 6-20V
- 54 entradas/salidas digitales (14 de ellas con salida PWM)
- 16 entradas/salidas analógicas
- Corriente máxima de salida: 40 mA
- Corriente máxima de salida para pin 3,3V: 50 mA
- Memoria Flash 256K
- SRAM: 8 KB
- EEPROM: 4 KB
- Frecuencia de funcionamiento: 16 MHz

Placa E-HEALTH

- Arquitectura: Compatible con Arduino.
- Memoria RAM: 2K.
- Microprocesador: Atmega328.
- Memoria flash: 32K.
- Tomas UART: 1.
- Sensor Electromiograma (EMG).

Electrodos de pólimeros conductivos

- Ganancia ajustable.
- Factor de Forma Pequeña.
- Completo integrado.
- Impedancia ACZ 220 ohmios.
- Voltaje de compensación de CC (antes de la simulación de desfibrilación) 0,2mV
- SDR (potencial restante después de la simulación de desfibrilación) 11 mV.

Módulo bluetooth HC-06

- Compatible con el protocolo Bluetooth V2.0.
- Voltaje de alimentación: 3.3VDC – 6VDC.
- Voltaje de operación: 3.3VDC.
- Baud rate ajustable: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.
- Tamaño: 4.4 cm x 1.6 cm x 0.7 cm
- Corriente de operación: < 40 mA
- Corriente modo sleep: < 1mA

Pantalla LCD HD44780:

- Pantalla de 2 líneas x 16 caracteres.
- Fuente de alimentación: 5V DC.
- Tamaño: 80mm x 36mm x 10mm.
- Área de visualización Tamaño: 64,5 mm x 15 mm.

Acelerómetro y Giroscopio MPU6050:

Rangos de escala y el valor máximo raw:

Rango De Escala Completa Giroscopio (grados por segundo)	Sensibilidad del Giroscopio (grados por segundo)	Rango De Escala Completa Acelerómetro (gramo)	Sensibilidad del Acelerómetro (LSB/gramo)
±250	131	±2	16.384
±500	65.5	±4	8.192
±1000	32.8	±8	4.096
±2000	16.4	±16	2.048

Servos Futaba S3003

- Tamaño: 40mm * 20mm * 44mm.
- Peso : 38g.
- Movimiento max: 180º.
- Torsión : 3,2 kg - cm a 4,8 V / 4,1 kg- cm a 6.0V.
- Velocidad: 0.23 seg / 60 º 4.8V - 19 seg / 60º 6.0V.

Ordenador portátil

- Marca: Toshiba satellite pro c50-a-1HQ
- Procesador: Intel Core i5-3230M.
- Memoria Ram: 8GB.
- Disco Duro: 500GB.
- Tarjeta Gráfica: GT740M.

Dispositivo móvil

- Marca: Sony Xperia L1
- Procesador MediaTek MT6737T 1.4GHz
- 2GB RAM
- 16GB, microSD
- OS: Android 7.0

REQUISITOS SOFTWARE

Arduino IDE 1.0.1

IDE el cual se ha utilizado para el desarrollo y programación de los programas de Arduino. Este software el gratuito y de código abierto y se puede descargar desde la página oficial de Arduino. Además el software de Arduino nos permite programar desde cualquier plataforma ya bien sea Windows, Linux o Mac.

Android Studio

Android Studio es un entorno de desarrollo integrado para la plataforma Android. Basado en IntelliJ IDEA que es un entorno o ambiente de desarrollo para programas, que posee potentes herramientas de edición de código.

Cura 3D

Software de impresión 3D que nos permite convertir los archivos STL y generar el G-Code de impresión de los modelos.

EAGLE

Herramienta para el diseño de placas de circuito impresas. Es sencillo para trabajar y de una gran versatilidad. Disponible para Windows, Linux y Mac OS. Al igual que los anteriores se trata de un software gratuito el cual tiene unas limitaciones, pero para el presente proyecto con la versión gratuito ha sido suficiente.

PARTE 4

IV- PRESUPUESTO

PRESUPUESTO

Para determinar el presupuesto total del proyecto se incluyen los costes de hardware y materiales utilizados ya que en lo referente a software se ha elegido la utilización de software libre.

Los costes de hardware, software y equipamiento se calcula mediante la siguiente formula de amortización:

$$C = \frac{Meses\ de\ uso\ de\ equipo}{Periodo\ de\ amortización} * Coste\ Equipo * \%\ Uso$$

Siendo:

- Meses de uso de equipo: El número de meses de uso del equipo durante el desarrollo del proyecto.
- Periodo de amortización: Vida útil del equipo (Software y hardware 36 meses).
- Coste del equipo: Valor del equipo de euros.
- % USO : Porcentaje de uso con valores de 0 a 1 referido a la utilización del equipo especificado en relación al tiempo en el proyecto.

Hardware

Hardware	Meses uso	% Uso	Amortización (meses)	Precio (€)	Total (€)
Ordenador	6	100	36	545	90.8
Arduino mega 2560	6	100	36	30.34	5
E-HEALTH	6	100	36	240	40
Sensor EMG	6	100	36	48	8
Bluetooth HC-06	6	100	36	4.93	0.82
LCD HD44780	6	100	36	2.95	0.5
MPU6050	6	100	36	2.81	0.47
Servos Futaba S3003	6	100	36	43	7.16
Dispositivo móvil	6	100	36	178	29.6
Subtotal hardware					**182.35**

Proyecto

	Precio/hora (€/hora)	Horas	Total (€)
Honorarios	60	360	21600
Horas impresión 3D	15	61	915
Subtotal proyecto			22515

Presupuesto final

	Total
Hardware	182.35
Proyecto	22,515
Subtotal	**22697.35**
I.V.A (21%)	4,766.44
TOTAL	**27463.79**