ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DISEÑO DE UNA ARQUITECTURA DE SOFTWARE Y APLICACIONES PRÁCTICAS PARA EXPLOTAR LAS CAPACIDADES DE UN SISTEMA DE RECONOCIMIENTO DE GESTOS DEL BRAZO HUMANO

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

FABRIZIO ELOY RAMÍREZ CUTIMBO

fabrizio.ramirez@epn.edu.ec

DIRECTOR: MARCO ANTONIO SEGURA MORALES, PhD.
marco.segura@epn.edu.ec

CODIRECTOR: MARCO ENRIQUE BENALCAZAR PALACIOS, PhD.
marco.benalcazar@epn.edu.ec

Quito, noviembre 2018

DECLARACIÓN

Yo, Fabrizio Eloy Ramírez Cutimbo, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Fabrizio Eloy Ramírez Cutimbo

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Fabrizio Eloy Ramírez Cutimbo, bajo nuestra supervisión.

PhD. Marco Segura Morales	**PhD Marco Benalcázar Palacios**
DIRECTOR DEL PROYECTO	**CODIRECTOR DEL PROYECTO**

AGRADECIMIENTOS

Le agradezco al PhD Marco Segura por el apoyo incondicional brindado durante el desarrollo de este trabajo, al PhD Marco Benalcázar por la confianza y consejos, y a todo el equipo de investigación PIJ-16-13 por la ayuda y conocimiento compartido.

Fabrizio Ramírez

DEDICATORIA

A mi familia y a mí.

Fabrizio Ramírez

ÍNDICE DE CONTENIDO

FIGURAS

TABLAS

RESUMEN

Este proyecto presenta una propuesta de arquitectura de soluciones diseñada para proporcionar a un "sistema de reconocimiento de gestos" la capacidad de exponer sus funcionalidades y capacidades para ser consumidas por cualquier otra aplicación de software con diversos propósitos. En este contexto, la arquitectura propuesta proporcionará capacidades de integración e interoperabilidad al sistema de reconocimiento de gestos del brazo humano, dándole la capacidad de interactuar con otras aplicaciones. La solución completa incluye diferentes elementos, como el dispositivo Myo Armband (*wereable*), un sistema de reconocimiento de gestos del brazo humano, una interfaz de comunicación y cualquier otra aplicación práctica que consuma las funcionalidades del sistema de reconocimiento. Para demostrar la factibilidad y utilidad de la arquitectura propuesta, se han diseñado e implementado dos prototipos de software, basadas en esta arquitectura, que son un emulador de mouse y un videojuego Tetris.

Palabras Clave— Arquitectura de Software, Integración de Software, TCP Sockets, UML, EMGs, Reconocimiento de gestos, Myo Armband, Tetris, Emulador de Mouse.

ABSTRACT

The present project presents a Solutions Architecture proposal designed to provide to a “gesture recognition system” the capacity to expose its functionalities and capabilities to be consumed by any other software application with diverse purposes. In this context, the proposed architecture will provide integration and interoperability capabilities to the human-arm gesture recognition system, giving it the ability to interact with other applications. The whole solution includes different elements such as the MyoArmband device (wearable), a human-arm gesture recognition system, a communications interface, and any other practical applications which consume the functionalities of the recognition system. To demonstrate the feasibility and usefulness of the proposed architecture, two software applications have been designed and implemented, based on this architecture, which are a mouse emulator and a Tetris game.

Keywords—Software Architecture, Software Integration, TCP Sockets, UML, EMGs, Gesture Recognition, MYO Armband, Tetris, Mouse Emulator.

1 INTRODUCCIÓN

1.1 OBJETIVOS

1.1.1 Objetivo General

Diseñar una arquitectura de software destinada a explotar las capacidades de un sistema de reconocimiento de gestos del brazo humano, e implementación de aplicaciones prácticas de software utilizando *SCRUM.*

1.1.2 Objetivos Específicos

- Proponer tres prototipos de arquitectura basados en distintos métodos de comunicación para evaluarlos.
- Comparar los prototipos en función de la factibilidad y simplicidad de su implementación para seleccionar uno.
- Mejorar iterativamente la arquitectura seleccionada hasta proveer todas las funcionalidades requeridas.
- Definir un esquema de comunicación estándar entre el sistema de reconocimiento y la interfaz gráfica.
- Implementar una interfaz gráfica que permita gestionar la información consumida y producida por el sistema de reconocimiento.
- Desarrollar un módulo de gestión de perfiles para usuarios.
- Desarrollar dos aplicaciones que hagan uso del sistema de reconocimiento de gestos a través de la interfaz.
- Validar las dos aplicaciones respecto al rendimiento, funcionalidad y facilidad de uso.

1.2 PLANTEAMIENTO DEL PROBLEMA

El presente proyecto integrador se desarrolla en el marco de trabajo del proyecto de investigación PIJ-16-13 “Clasificación de señales electromiográficas del brazo humano usando técnicas de reconocimiento de patrones y Machine Learning” [1], dentro del cual se ha desarrollado un “Sistema de reconocimiento de gestos del brazo humano”. Este sistema ha utilizado como herramienta de apoyo el dispositivo Myo Armband [2] el cual es un

Wereable, es decir, un dispositivo electrónico lo suficientemente pequeño para ser vestido por una persona. Este dispositivo permite utilizar los gestos y movimientos del usuario para interactuar con computadoras, teléfonos móviles y otros dispositivos [2] sin necesidad de hacer contacto físico. El "sistema de reconocimiento" extrae las señales electromiográficas superficiales (*EMGs*) [3] de los diferentes sensores (electrodos) del dispositivo, para procesarlas y determinar qué gesto está ejecutando el usuario. El sistema de reconocimiento ha sido íntegramente implementado utilizando Matlab. Los gestos detectados actualmente por el sistema son seis: los dedos estirados (*Fingers Spread*), el puño (*Fist*), la mano apuntando hacia adentro con los dedos juntos (Wave In), la mano apuntando hacia afuera con los dedos juntos (*Wave out*), el doble chasquido entre el pulgar y el dedo medio (*Double Tap*) y finalmente la mano en estado de reposo (*No Gesture*).

La problemática que se plantea resolver es la necesidad del "sistema de reconocimiento de gestos" de exponer su funcionalidad de reconocer los gestos del brazo humano. Para ello es necesario brindarle las capacidades de integración necesarias para que pueda interactuar con otras aplicaciones. Estas aplicaciones podrán tomar ventaja del reconocimiento de gestos y otra información obtenida directamente desde el dispositivo Myo Armband.

Se propone diseñar una arquitectura de software que incluya elementos de hardware como el dispositivo Myo Armband, componentes de software como el sistema de reconocimiento de gestos, una interfaz de comunicación, y aplicaciones prácticas cualquiera. Esta arquitectura permitirá la interacción entre el sistema de reconocimiento de gestos y cualquier aplicación de software a través de una interfaz de comunicación.

Para demostrar la factibilidad y utilidad de la arquitectura se ha definido desarrollar dos aplicaciones de software basadas en dicha arquitectura. Estas emplearán tanto la funcionalidad de reconocimiento de gestos como la información espacial obtenida de los sensores del dispositivo Myo Armband. Las aplicaciones a desarrollar son: un emulador de mouse y un video juego Tetris. Estas aplicaciones se evaluarán respecto al rendimiento, funcionalidad y facilidad de uso de las mismas. A partir de los resultados obtenidos, el equipo de investigación considerará desarrollar aplicaciones para controlar e interactuar con *gadgets*, *robots*, u otros *wereables*.

1.3 ALCANCE

El presente proyecto integrador ha sido desarrollado en el marco de trabajo del proyecto de investigación PIJ-16-13 de la Escuela Politécnica Nacional. Al inicio de este proyecto, el equipo de investigación ya había desarrollado un modelo de reconocimiento de gestos y paralelamente se encontraba investigando diferentes propuestas sobre la misma temática.

Tal como se especificó en una reunión con el representante del equipo de investigación, para agilizar el proceso de desarrollo solo se utilizó el sistema de reconocimiento de gestos que ya se encontraba terminado.

La arquitectura de software ha sido diseñada para dotar de capacidades de integración a los sistemas de reconocimiento de gestos, implementados en Matlab, que el equipo de investigación pueda crear. Sin embargo, la implementación de la arquitectura se limita a utilizar el sistema de reconocimiento de gestos del brazo humano finalizado descrito en el siguiente *paper* [1].

El sistema de reconocimiento de gestos implementado en Matlab y su versión compilada, será considerada como una caja negra de la cual solo se conocen los parámetros de entrada y de salida. Esta caja negra puede tener errores, los cuales inevitablemente son heredados a la arquitectura de software y por ende a las aplicaciones.

Las aplicaciones prácticas han sido ideadas y desarrolladas considerando que el sistema puede reconocer solo seis gestos del brazo humano, los gestos se listan a continuación: *Wave In, Wave Out, Fingers Spread, Fist, Double Tap y Rest*.

Es importante mencionar que particularmente el sistema de reconocimiento de gestos empleado, requiere de elevados recursos computacionales lo cual puede reducir el rendimiento de las aplicaciones y aumenta los requisitos mínimos para la instalación del producto en computadoras de usuarios finales.

Finalmente, las aplicaciones prácticas y la arquitectura implementada serán evaluadas únicamente respecto al rendimiento, funcionalidad y facilidad de uso.

1.4 JUSTIFICACIÓN

1.4.1 Justificación Teórica

El presente trabajo plantea el diseño de una arquitectura de software que permita explotar las capacidades del sistema de reconocimiento de gestos del brazo humano. En la actualidad, como lo explica Pandit [4], las computadoras se están acoplando a diferentes

aspectos de la vida cotidiana, por lo cual es necesario estudiar nuevas perspectivas de interacción humano computador, como la que ofrece la interacción con dispositivos *Wereables*, como el dispositivo Myo Armband [2], dando paso a una nueva tendencia denominada BAN [5] (Body Area Network). Por este motivo, es necesario encontrar una forma de exponer la funcionalidad del "sistema de reconocimiento de gestos" para ser aprovechada en la creación de aplicaciones de software de múltiples propósitos.

Para la explotación de las funcionalidades del "sistema de reconocimiento de gestos", primero es necesario diseñar una arquitectura. Como lo explican Bass y Gorton [6] [7], el desarrollo de una arquitectura de software de un programa o un sistema informático, permite definir la estructura del sistema. Esta puede incluir elementos de software, hardware y las relaciones entre ellos. Además, como lo explica Garland [8] el desarrollo de una arquitectura permite identificar los problemas estructurales que involucran la organización general y estructura de control global. Dicha estructura de control global comprende: protocolos de comunicación, métodos de sincronización, acceso a datos, asignación de funcionalidad a elementos de diseño, composición de elementos de diseño, distribución física, escalabilidad, selección entre alternativas de diseño y asegura la calidad del producto final [9].

La arquitectura a desarrollar detallará la estructura del nuevo sistema. Dicha estructura será conformada por el dispositivo Myo Armband, un "sistema de reconocimiento de gestos", una interfaz de comunicación y una aplicación de software cualquiera. La interfaz debe establecer un canal de comunicación entre el "sistema de reconocimiento de gestos", que en principio es una aplicación independiente, y cualquier otra aplicación. En ese contexto, se ha determinado que la comunicación se realizará usando "Sockets de comunicación". Como lo explica Makofske [10] es una abstracción a través de la cual una aplicación puede enviar y recibir datos de forma ordenada y confiable. Se ha elegido este mecanismo de comunicación debido al buen rendimiento y fácil implementación que ofrece cuando se trabaja de forma local.

1.4.2 Justificación Práctica

La tendencia hacia el desarrollo de aplicaciones para dispositivos *wereables* va en aumento [4]. Según Huang [5], el reconocimiento de gestos tiene múltiples aplicaciones en el sector del entretenimiento, internet de las cosas, salud, robótica, etc. El diseño de esta arquitectura de software permitirá que se puedan desarrollar diferentes tipos de aplicaciones. Por ejemplo, programas utilitarios como el emulador de mouse o video juegos como Tetris. Las aplicaciones desarrolladas en base a esta arquitectura podrán ser usados por cualquier

persona que posea un dispositivo Myo Armband. También se ofrece la posibilidad que otros desarrolladores puedan utilizar la interfaz de comunicación para la creación de nuevas aplicaciones.

1.5 MARCO TEÓRICO

1.5.1 Arquitectura de Software

Definición

De acuerdo a la ANSI/IEEE Std 1471-2000 [11] la Arquitectura de Software es definida como la organización fundamental de un sistema representado en componentes, las relaciones que existen entre estos, el entorno que la define y los principios que guían su diseño y evolución. Es decir, se concibe a la arquitectura como un conjunto interrelacionado de componentes que depende de un entorno y en consecuencia puede cambiar. Len Bass [6] define la Arquitectura de Software como la estructura o estructuras de un sistema computacional, el cual comprende los “elementos” de software, las propiedades externas visibles de dichos elementos y las relaciones que existen entre estos. Además, estos autores especifican que una arquitectura de software es una abstracción que elimina los detalles de los elementos de software que no afecten a la definición de su uso o las interacciones entre ellos.

Garlan [8], añade que una arquitectura de software permite identificar los problemas estructurales que involucra la organización general y estructura de control global; determinar los protocolos de comunicación, sincronización, acceso a datos, asignación de funcionalidad a elementos de diseño, distribución física, composición de elementos de diseño, escalabilidad, selección entre alternativas de diseño y asegura la calidad del producto final.

Comunicación

Una Arquitectura de Software al ser dividida en un conjunto de componentes o elementos de software debe especificar cómo realizará la comunicación de datos y el control de la información entre estos componentes. Para ello existen varias formas de realizar dicha comunicación, por ejemplo: los componentes pueden compartir el mismo espacio de memoria y realizar llamadas a los métodos; pueden **ejecutarse en diferentes Hilos (Threads) o procesos y comunicarse a través de mecanismos sincrónicos** o los componentes pueden estar a la espera de un evento determinado [7]. Existen varias

posibilidades.

Por otra parte, si se adoptan patrones arquitectónicos, estos tienen definidas estructuras conocidas y comprobadas que facilitan la comunicación entre componentes. Cuando se trata de sistemas complejos se pueden utilizar y combinar distintos patrones de diseño y a la vez sus diferentes estructuras de comunicación.

Marketecture

También llamada "*Marchitecture*", es una abstracción coloquial de una arquitectura de software. Describe informalmente la estructura e interacciones de una arquitectura de software. Se enfoca en mostrar los componentes y las relaciones más importantes teniendo en cuenta las expectativas de los interesados. Una *Marketecture*, es una herramienta muy útil que facilita la discusión con los interesados durante el diseño, construcción y revisión. **Esta debe ser fácil de entender y explicar, sirve como el punto de partida para realizar un análisis más profundo**. [7]

Vistas Arquitectónicas

La arquitectura de software representa una descripción detallada de un producto. Es necesario entender y analizar una arquitectura desde diferentes puntos de vista. De acuerdo a Philippe Krutchen [7] existen cuatro formas para describir y entender una arquitectura:

- Vista Lógica (*Logical view*): Describe los elementos relevantes para la arquitectura y las relaciones entre ellos. La vista lógica captura la estructura de una aplicación utilizando diagramas de clase, componentes o equivalentes.
- Vista de Proceso (*Process view*): Describe los elementos de comunicación y concurrencia de una arquitectura. Detalla los mecanismos de comunicación sincrónicos y asincrónicos utilizados.
- Vista Física (*Physical view*): Describe como se mapean los principales procesos y componentes en los elementos de hardware que la arquitectura utilizará.
- Vista de Desarrollo (*Development view*): Describe la organización interna de los componentes de software. Explica cómo se implementarán o configurarán los elementos de software en el ambiente de desarrollo.

Integración

La integración es un atributo de calidad de software junto al rendimiento, escalabilidad, modificabilidad, seguridad, disponibilidad, *testability*, portabilidad y mantenibilidad. Se refiere a la facilidad con la que una aplicación puede incorporarse de manera útil en un contexto de aplicación más amplio. El valor de una aplicación o componente aumenta si su funcionalidad o datos se pueden utilizar de formas que el diseñador no anticipó originalmente. Las estrategias más comunes para proporcionar integración son:

- Integración de datos: implica almacenar los datos que una aplicación manipula, de forma que otras aplicaciones puedan acceder a estos. Por ejemplo: el uso de una base de datos relacional estándar, implementar mecanismos para extraer datos en un formato conocido como XML o un archivo de texto separado por comas.
- Interfaz de programación de aplicaciones (API): En este caso, los datos que posee la aplicación están ocultos detrás de un conjunto de funciones que facilitan el acceso externo controlado. De esta manera, las reglas comerciales y la seguridad se pueden aplicar en la implementación de la API. La única forma de acceder a los datos e integrarlos con la aplicación es mediante el API proporcionado.

La integración de datos debe ser flexible y simple. La mayoría de lenguajes de programación pueden procesar texto o acceder a bases de datos relacionales mediante SQL. Crear una API requiere más esfuerzo, pero proporciona mayor control, en términos de corrección y seguridad. La elección de una estrategia de integración no es simple y depende de lo que se desea lograr y de las restricciones que existen [7].

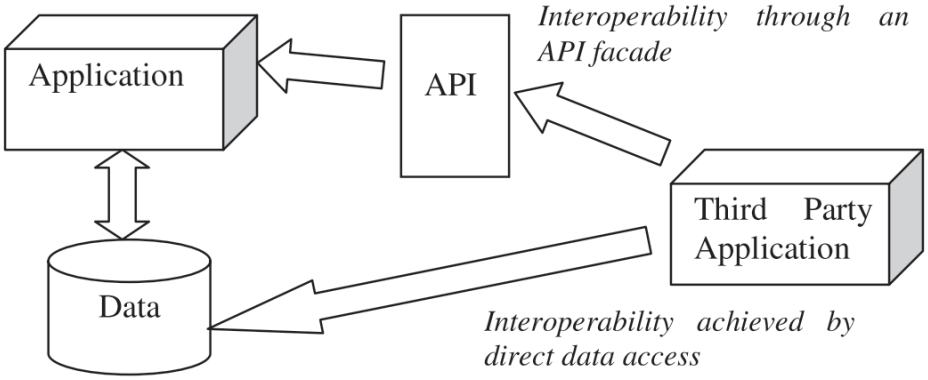

Figura 1 Estrategias de Integración de Software. Imagen tomada de [7]

Documentación

La documentación de la arquitectura permite que el equipo de un proyecto y otras personas (Interesados) puedan entender fácilmente el diseño. Es decir que permite la comunicación entre los miembros de un proyecto: arquitectos, diseñadores, programadores, testers, jefes de proyecto, clientes, etc. Además, permite analizar el diseño para evaluarlo respecto al rendimiento o las métricas estándar de cohesión y acoplamiento.

No existe un estándar para la documentación de arquitecturas, debido a que estas pueden ser complejas. La notación de diseño más utilizada en la industria son los diagramas "*Block and Arrow*"[7]. Sin embargo, es recomendable utilizar notaciones formales y ampliamente aceptadas como UML (*Unified Modeling Language*).

Dependiendo del tamaño y distribución de los equipos, la documentación puede ser mínima. En ocasiones para completar la documentación de un sistema, basta con una *marketecture*, que describa los principales componentes. Este nivel de documentación es fácil de entender, explicar, producir y no genera mayor costo. Cuando el equipo y el proyecto son de mayor escala es necesario que se contemplen los siguientes elementos:

- Interfaces de componentes.
- Restricciones de subsistemas.
- Escenarios de prueba.
- Decisiones de compra de componentes de terceros.
- Estructura del equipo y dependencias del cronograma.
- Servicios externos a ser ofrecidos por la aplicación.

También es importante considerar la longevidad de una aplicación para determinar el nivel de documentación. **Un diseño arquitectónico evoluciona a medida que el sistema/aplicación se desarrolla incrementalmente y se obtienen más conocimientos sobre el dominio del problema**. Es importante mantener la documentación actualizada [7].

1.5.2 Unified Modeling Language

El Lenguaje unificado de modelado UML es un estándar controlado por la *Object Management Group* (OMG)[1]. Es una familia de notaciones gráficas, basada en un

[1] Object Management Group (OMG) es un consorcio internacional sin fines de lucro de estándare tecnología.

metamodelo único [12], que permite especificar, visualizar y documentar modelos de sistemas de software, incluyendo su estructura y diseño [13]. UML permite modelar casi cualquier tipo de aplicación, aunque se adapta mejor a sistemas de software orientados a objetos. La OMG como proveedor de UML no especifica ninguna metodología de desarrollo en particular para el uso de UML.

La versión actual UML 2.0 posee trece tipos de diagramas estándar, divididos en tres categorías: Seis tipos de diagramas representan la estructura estática de la aplicación; tres representan tipos generales de comportamiento; y cuatro representan diferentes aspectos de las interacciones[13]:

- **Los diagramas de estructura** incluyen el diagrama de clases, el diagrama de objetos, el diagrama de componentes, el diagrama de estructura compuesta, el diagrama de paquetes y el diagrama de despliegue.
- **Los diagramas de comportamiento** incluyen el Diagrama de caso de uso (utilizado por algunas metodologías durante la recopilación de requisitos); Diagrama de actividades y Diagrama de máquina de estado.
- **Los Diagramas de Interacción**, derivados del Diagrama de Comportamiento general, incluyen el Diagrama de Secuencia, el Diagrama de Comunicación, el Diagrama de Tiempo y el Diagrama de Descripción de Interacción.

Diagrama de Componentes

Un diagrama de componentes muestra la organización y dependencias de un conjunto de componentes de software [14]. Los componentes representan piezas de software que se pueden adquirir, actualizar o mejorar independientemente tales como: ejecutables, librerías, archivos, documentos, código fuente, *scripts* y tablas [15].

Se debe utilizar este tipo de diagrama cuando se desea dividir el sistema en componentes para mostrar la interrelaciones que existen entre ellos o la configuración del diseño a bajo nivel [12]. Los diagramas de componentes se utilizan para modelar la vista estática de un sistema [14].

Diagrama de Secuencia

Los diagramas de secuencia, se utilizan para explorar la naturaleza dinámica de un producto de software. Los diagramas de secuencia muestran el flujo de mensajes entre objetos en una aplicación orientada a objetos [15]. Pero también permite ubicar libremente los

participantes del sistema, establecer enlaces para mostrar cómo se conectan los participantes y usar una numeración para mostrar la secuencia de los mensajes [12].

1.5.3 Sockets de comunicación

Un socket de comunicación es una abstracción a través de la cual una aplicación puede enviar y recibir datos, su funcionamiento es similar a permitir que una aplicación lea y escriba datos en un archivo compartido. Un socket permite que una aplicación se "conecte" a una red y se comunique con otras aplicaciones que también están conectadas a la misma red. La información escrita en el socket por una aplicación, en una computadora, puede ser leída por otra aplicación, en otra máquina diferente, y viceversa [10].

Existen diferentes tipos de Sockets dependiendo del protocolo o *stack* de protocolos al cual se encuentran asociados. Los *stacks* de protocolos más conocidos son TCP que utiliza "*Stream Sockets*" y UDP que utiliza "*Datagram Sockets*". Un *socket* TCP/IP se identifica utilizando una dirección IP y un número de puerto [10].

1.5.4 Interacción Humano Computador

La interacción Humano Computador (HCI), es una disciplina concerniente al diseño, evaluación e implementación de sistemas computacionales interactivos para los humanos usados en contextos sociales y el estudio de los principales fenómenos que los rodean [16] [17].

HCI es una materia multidisciplinaria que se basa en las siguientes disciplinas: psicología, ciencia cognitiva, ergonomía, sociología, ingeniería, diseño gráfico, ciencias de la computación, diseño de sistemas e ingeniería de software. Los principales problemas que abarca la disciplina HCI son el diseño, **usabilidad** e interacción. HCI es utilizado para el desarrollo de diversas aplicaciones en: e-learning, informática de la salud, entretenimiento y muchos otros [17].

1.5.5 Electromiografía

La electromiografía (EMG por sus siglas en inglés) es una técnica que estudia el movimiento humano, para evaluar los mecanismos que involucran la fisiología neuromuscular y diagnosticar trastornos neuromusculares [18]. **Las señales EMG son generadas por la**

contracción de los músculos esqueléticos (motores) los cuales permiten el movimiento humano. El mecanismo de funcionamiento de estos motores incorpora estrategias de control central tales como: transmisión de señales a lo largo de fibras nerviosas y uniones neuromusculares, activación eléctrica de las fibras musculares organizadas en motores elementales (las unidades motoras), activación de una cadena de eventos bioquímicos complejos y producción de fuerzas que actúan sobre los tendones de los músculos agonistas y/o antagonistas para mover los huesos [19].

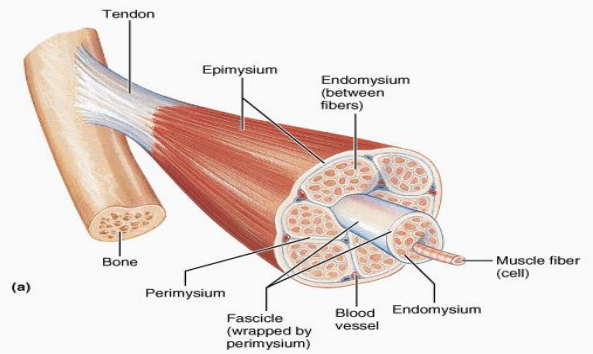

Figura 2 Anatomía de los músculos esqueléticos. El músculo se encuentra unido al hueso por medio de tendones. Imagen obtenida de [19].

Técnicas de obtención de señales EMG

Existen dos técnicas para la obtención de señales EMG, la invasiva que utiliza agujas y las no invasivas que utilizan electrodos superficiales, guantes, cámaras, etc. Ambas son instrumentos complementarios y se integran entre sí. La técnica invasiva es más adecuada y ampliamente aceptada para aplicaciones de diagnóstico e investigación fisiológica. La segunda permite la evaluación frecuente y sin dolor, tiene aplicaciones importantes en los campos de ergonomía, control de prótesis, medicina deportiva y entretenimiento [19].

Figura 3 Herramientas para la obtención de señales EMG, de izquierda a derecha: electrodos de aguja EMG, electrodos adhesivos superficiales, cámara de profundidad (Kinect), guante instrumentado.

1.5.6 Ángulos Tait-Bryan

Es una convención ampliamente utilizada de una secuencia de ángulos XYZ: *Yaw, Pitch* y *Roll*. Técnicamente son ángulos náuticos o de Cardan. Son utilizados en la industria aeroespacial y terrestre debido a que son intuitivos al describir la actitud de vehículos como barcos, aviones y automóviles. *roll*, *pitch* y *yaw* (también llamado *bank, attitude* y *heading*) se refieren a las rotaciones sobre los ejes X, Y, Z, respectivamente. El eje X se define comúnmente en la dirección hacia adelante, el eje Z hacia abajo y el eje Y hacia el lado derecho [20].

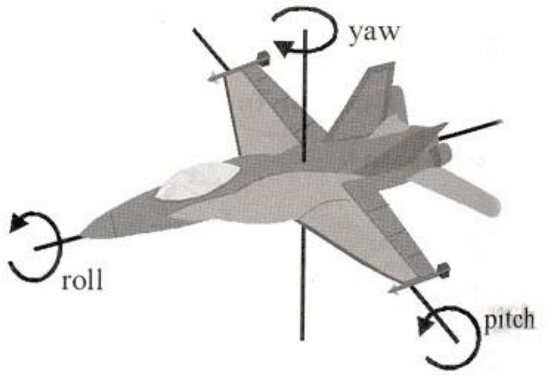

Figura 4 Ángulos Yaw, Pitch y Roll descritos sobre un avión.

1.6 MATERIALES Y HERRAMIENTAS

1.6.1 Myo Armband

Es un dispositivo comercial desarrollado por la empresa Thalmic Labs[2]. Es un *Wearable*[3] que permite utilizar los gestos y movimientos del antebrazo del usuario para interactuar con computadoras, teléfonos móviles y otros dispositivos sin necesidad de hacer contacto físico. El dispositivo Myo Armband es un brazalete compuesto por 8 electrodos que se distribuyen equidistantemente alrededor del antebrazo del usuario. Utiliza *bluetooth* como medio de transmisión de datos hacia la computadora. Posee un software propietario para la detección de gestos y varias aplicaciones prácticas disponibles en su tienda *Online*. Sin embargo, no permite interactuar directamente con otros sistemas de reconocimiento de gestos [2].

[2] Thalmic Labs Inc. Empresa de software que desarrolla tecnologías de interacción humano-computador. Ofrece el dispositivo Myo Armband. La compañía fue fundada en 2012 y tiene su sede en Kitchener, Canadá. (Página web no disponible).
[3] Wereable: dispositivo electrónico lo suficientemente pequeño para ser vestido por una persona.

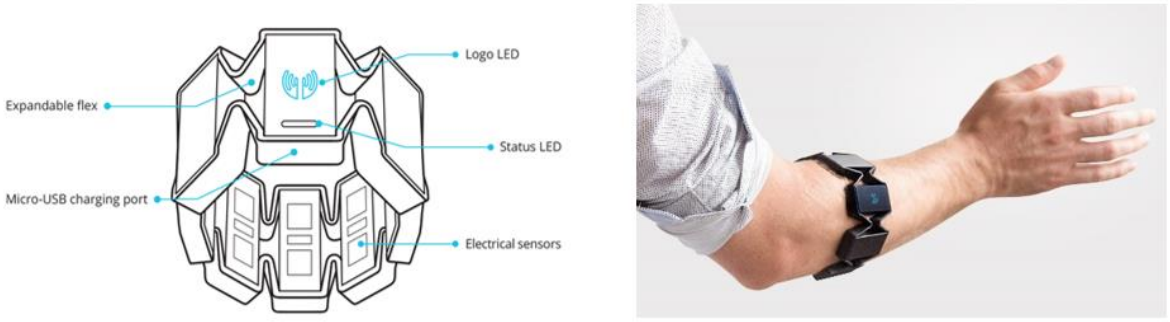

Figura 5 Izquierda: Estructura del dispositivo Myo Armband. Derecha: Dispositivo siendo correctamente vestido por un usuario [2].

1.6.2 Myo Connect

Es el Software privativo de Talmics Labs encargado de controlar el dispositivo Myo Armband. Establece conexión con el dispositivo Myo Armband vía *Bluetooth* a través de un *USB Dongle* que debe conectarse al PC. Este software es el único que puede encender, apagar, conectar y desconectar el dispositivo Myo Armband. Además, esta aplicación permite configurar el dispositivo y gestionar las aplicaciones descargadas de Myo Market [21] (tienda *online*).

1.6.3 Myo SDK

Es el *Software Development Kit* oficial creado por Talmic Labs, disponible en la página de desarrolladores de Myo Armband. Este SDK permite al desarrollador acceder a dos tipos de datos:

- Datos Espaciales *(Spatial Data)*: proporciona información acerca de la orientación y movimiento del brazo del usuario.
- Datos de Gestos *(Gestural Data)*: informa que está realizando el usuario con su mano, detecta cuando un usuario realiza un gesto.

El núcleo de Myo SDK es la librería “Libmyo”. Esta librería permite a las aplicaciones interactuar con el Myo Armband. Toda la funcionalidad de la librería es expuesta en una API desarrollada en C.

Generalmente, las aplicaciones no interactúan directamente con la librería “LibMyo” sino que utilizan un “Language Binding” que corresponde al lenguaje de programación en el cual se va a desarrollar la aplicación. Oficialmente Myo Developers ofrece el conector para el lenguaje C++. Sin embargo, también existen en internet conectores para otros lenguajes

como C# y Matlab los cuales analizaremos más adelante [22]. La **Figura 6** ilustra el *stack* de desarrollo Myo, desde una aplicación que usa el SDK hasta un dispositivo Myo físico [23].

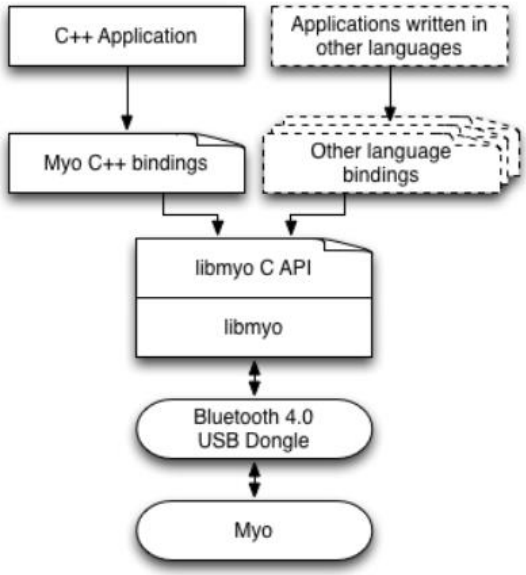

Figura 6 Stack de desarrollo de una aplicación que usa el SDK hasta el dispositivo Myo físico. Figura tomada de [23].

1.6.4 Matlab

Es una plataforma de programación enfocada en la matemática computacional que permite realizar: análisis de datos, desarrollo de algoritmos y creación de modelos. Esta herramienta es ampliamente usada por ingenieros y científicos para aplicaciones de Deep Learning, Machine Learning, **procesamiento** y comunicaciones **de señales**, procesamiento de imágenes y video, sistemas de control, finanzas, biología computacional, etc. Se utilizaron las versiones Matlab R2015a y R2017b [24].

1.6.5 Matlab Application Compiler

Matlab Compiler permite compartir programas desarrollados en Matlab como aplicaciones independientes también llamadas *Standalone Applications*. Todas las aplicaciones creadas con Matlab Compiler utilizan el Matlab Runtime, el cual es un conjunto de librerías que permite la ejecución de aplicaciones compiladas sin la necesidad de tener instalado Matlab. El Matlab Runtime puede ser empaquetado e instalado cuando se crea la aplicación [25]. Además, todas las aplicaciones creadas por el Application Compiler, pueden ser instaladas en cualquier computadora sin requerir de una licencia de Matlab.

1.6.6 Matlab Library Compiler

Extiende la funcionalidad del Matlab Compiler y permite convertir código M (Lenguaje Matlab) en librerías para C/C++, Java y .NET. Estos componentes pueden ser integrados y desplegados en aplicaciones web o de escritorio libre de Impuestos. Esta funcionalidad, al ser una extensión del Matlab Compiler también hace uso del Matlab Runtime [26].

1.6.7 Myo MEX

Es un SDK desarrollado para Matlab que permite a los usuarios obtener datos de uno o dos dispositivos Myo Armband. Es posible obtener señales EMGs de los 8 sensores electromiográficos en forma de Stream y los datos IMU (*Inertial Measurement Unit*) que incluyen: cuaternios (Orientación), tres ejes del giroscopio (Velocidad Angular) y tres ejes del acelerómetro (Aceleración Lineal). Este SDK se encuentra disponible en GitHub [27]. Se utilizó específicamente la versión 3.0.0

1.6.7.1 Visual Studio IDE – C#

Para la implementación de las aplicaciones prácticas se utilizó el entorno de desarrollo integrado [28] Visual Studio 2017 con el framework .NET 4.5, el lenguaje de programación C# y Windows Forms [29]. En conjunto, estas herramientas permiten implementar fácilmente sockets de comunicación, interfaces gráficas de usuario y aplicaciones de escritorio.

1.6.8 Myo Sharp

Es un SDK desarrollado para el lenguaje de programación C#, es compatible con el framework .NET 2 en adelante. Se encuentra disponible en GitHub [30].

1.6.9 Bunifu

Librería que facilita la implementación de interfaces gráficas atractivas y con un diseño moderno. Se utilizó esta librería para mejorar la experiencia del usuario [31].

2 METODOLOGÍA

2.1 Metodología SCRUM

Considerando que el objetivo del proyecto es diseñar e implementar una arquitectura de software y el desarrollo de dos aplicaciones prácticas que empleen dicha arquitectura, se hace necesario elegir una metodología de desarrollo de software. Se ha decido utilizar una metodología de desarrollo ágil apoyada en SCRUM [32], dado que, para este proyecto, el equipo de desarrollo y el de investigación suman un número reducido de personas que emplearán una comunicación fluida y constante. Como lo indica Schwaber y Sutherland [32], ésta es una forma de desarrollar software que define una secuencia de iteraciones (*Sprints*). Cada una de estas iteraciones genera un incremento completo de funcionalidad que se basa en los incrementos previos. Las iteraciones continúan hasta que se alcanza el objetivo del proyecto y se optimiza el producto.

2.1.1 Equipo SCRUM (SCRUM Team)

2.1.1.1 Dueño del Producto (Product Owner)

El dueño del producto es la persona responsable de gestionar la Lista del Producto (Product Backlog), es decir, se encargará de expresar claramente los elementos de la Lista del Producto y también es la única persona que puede modificarla. El dueño del producto es una única persona, representa las necesidades de los interesados [33]. Sus funciones principales son las siguientes:

- Ordenar los elementos de la lista del producto para alcanzar los objetivos del proyecto (Priorización)
- Determinar las siguientes tareas del equipo de desarrollo.
- Asegurar que la lista del producto sea visible y clara para el equipo de desarrollo.
- Asegura que el equipo de desarrollo comprenda los elementos de la lista del producto al nivel necesario.

2.1.1.2 Equipo de Desarrollo (Development Team)

El equipo de desarrollo se encarga de convertir los elementos de la Lista del Producto en incrementos de funcionalidad potencialmente desplegables al final de cada *Sprint*. El equipo de desarrollo debe ser autoorganizado y multifuncional. Como equipo deben poseer las

habilidades necesarias para generar un incremento, pero individualmente los miembros del equipo deben tener habilidades especializadas en determinadas áreas [33].

2.1.1.3 SCRUM Master

Es el responsable de asegurar que el equipo SCRUM trabaja de acuerdo a la teoría, buenas prácticas y reglas de SCRUM. Ayuda a las personas externas al equipo a entender qué interacciones con el equipo pueden ser de ayuda y cuáles no. El SCRUM Master se asegura que el dueño del producto sepa cómo ordenar correctamente la Lista de Producto para maximizar el valor, y que los elementos de la lista sean claros y concisos. Guía al equipo SCRUM a crear productos de alto valor. Ayuda al equipo SCRUM a ser autoorganizado y multifuncional. Elimina impedimentos para el progreso del proyecto. Se encarga de coordinar los eventos de SCRUM cuando se requiera o necesite [33].

2.1.2 Eventos de SCRUM (SCRUM Team)

2.1.2.1 El Sprint

Es el evento más importante de SCRUM, es un periodo de tiempo (*time-box*) en el cual se crea un incremento de producto “terminado”, utilizable y potencialmente desplegable. Un *Sprint* debe tener una duración de un mes calendario o menos. Un Sprint contiene y consiste en los siguientes eventos: Reunión de Planificación del *Sprint*, los SCRUM Diarios, el trabajo de desarrollo, la Revisión del *Sprint*, y la Retrospectiva del *Sprint* [33].

2.1.2.2 Reunión de Planificación de Sprint (Sprint Planning Meeting)

En este evento el equipo SCRUM completo planifica el trabajo a realizar durante el Sprint. La Reunión de Planificación de Sprint responde a las siguientes preguntas:

- ¿Qué puede entregarse en el Incremento resultante del Sprint que comienza?
- ¿Cómo se conseguirá hacer el trabajo necesario para entregar el Incremento?

Esta reunión toma como entrada la lista de producto y el último incremento del producto (si existe). El equipo de desarrollo selecciona los elementos de la lista de producto que considera pueden completar en el siguiente *Sprint*. Luego, el Equipo SCRUM elabora el Objetivo del Sprint (*Sprint Goal*).

Después de seleccionar los elementos de la lista de producto, el equipo de desarrollo decide como construirá esta funcionalidad y crea un plan para terminarlo, esta información se

registra en un artefacto de salida que recibe el nombre Lista de Pendientes del Sprint (*Sprint Backlog*) [33].

2.1.2.3 SCRUM Diario (Daily SCRUM)

Es una reunión diaria que dura 15 minutos para que el equipo de desarrollo sincronice sus actividades, cree un plan para las siguientes 24 horas y se formule las siguientes preguntas [33]:

- ¿Qué hice ayer que ayudó al Equipo de Desarrollo a lograr el Objetivo del *Sprint*?
- ¿Qué haré hoy para ayudar al Equipo de Desarrollo a lograr el Objetivo del *Sprint*?
- ¿Veo algún impedimento que evite que el Equipo de Desarrollo o yo logremos el Objetivo del *Sprint*?

2.1.2.4 Revisión de Sprint (Sprint Review)

Es una reunión que se realiza al final de un *Sprint* para revisar el incremento realizado y actualizar la lista de producto. En esta reunión participa todo el equipo SCRUM y los interesados para revisar que se hizo durante el *Sprint*.

En esta reunión el equipo SCRUM, explica que elementos de la Lista del Producto se han terminado y cuáles no, que se hizo bien durante el Sprint, que problemas hubo y como fueron resueltos. También se realiza una demostración del producto “Terminado” y se responde preguntas acerca del incremento.

El dueño del producto habla acerca de la Lista de Producto y su estado actual, junto al grupo completo se conversa qué hacer a continuación, de modo que la revisión del *Sprint* proporcione información de entrada para la siguiente reunión de planificación de *Sprint*

El resultado de la Revisión de *Sprint* es una Lista de Producto revisada y actualizada [33].

2.1.2.5 Retrospectiva de Sprint (Sprint Retrospective)

La Retrospectiva de *Sprint* tiene lugar después de la Revisión de *Sprint* y antes de la siguiente Reunión de Planificación de *Sprint*. Es un evento que permite al equipo SCRUM inspeccionarse a sí mismo y crear un plan de mejoras que se implementará en el siguiente *Sprint* [33].

2.1.3 Artefactos de SCRUM

2.1.3.1 Lista de Producto (Product Backlog)

Es una lista ordenada de todos los requisitos del producto. Este artefacto nunca está completo. A medida que un producto es utilizado y el interesado proporciona retroalimentación, la Lista de Producto se convierte en una lista más larga y exhaustiva.
Los elementos de la Lista de Producto de orden más alto son generalmente más claros y detallados que los de menor orden. El refinamiento (*refinement*) de la Lista de Producto es el acto de añadir detalle, estimaciones y orden a los elementos de la Lista de Producto. El contenido, disponibilidad y ordenación de la lista es responsabilidad del Dueño de Producto [33].

2.1.3.2 Lista de Pendientes del Sprint (Sprint Backlog)

Es un subconjunto de elementos de la Lista de Producto seleccionados para ser completados durante un *Sprint*. Se detalla todo el trabajo que el Equipo de Desarrollo identifica como necesario para alcanzar el objetivo del *Sprint*. También incluye un plan para entregar el incremento del producto y conseguir el objetivo del *Sprint*. Este artefacto es gestionado por el equipo de desarrollo durante el *Sprint* [33].

2.1.3.3 Incremento

Es el conjunto de elementos de la Lista de Producto que fueron completados durante el *Sprint* sumado a los incrementos de los *Sprint* anteriores. Al final de un *Sprint* el nuevo incremento debe estar “Terminado” [33].

2.1.3.4 Definición de “Terminado” (Definition of “Done”)

Para que un elemento de la Lista de Producto o un Incremento se considere como “Terminado”, este debe ser utilizable, de modo que el Dueño de Producto podría liberarlo inmediatamente. La definición puede variar significativamente en cada equipo SCRUM por ello se debe asegurar el entendimiento de la definición [33].

2.2 Implementación de la Metodología

Definición del Equipo SCRUM

Como se mencionó en la sección anterior, un equipo SCRUM está conformado por 3 roles, a continuación, se lista las personas que tomará cada rol:

- El rol de Product Owner fue desempeñado por el director del proyecto de investigación PIJ-16-13.
- El rol de SCRUM Master fue desempeñado por el director del proyecto integrador.
- El Equipo de Desarrollo fue integrado por las siguientes personas:
    - Fabrizio Ramírez quien se encargó del diseño e implementación de la arquitectura de software propuesta y las aplicaciones prácticas (Emulador de mouse y Tetris).
    - Otro miembro ocasional, integrante del proyecto de investigación. Se encargó de realizar mejoras y modificaciones al sistema de reconocimiento de gestos.

Finalmente, cabe mencionar que los **interesados** del presente proyecto, son los miembros del equipo de investigación PIJ-16-13 “Clasificación de señales Electromiográficas del brazo humano usando técnicas de reconocimiento de patrones y machine Learning” de la Escuela Politécnica Nacional. En adelante, para referirse a los **interesados**, se usará el término “Equipo de Investigación”.

Tabla 1 Tabla de Roles y responsables del equipo SCRUM.

Equipo SCRUM	
Rol SCRUM	**Responsable**
Dueño del Producto	Director del proyecto de investigación PIJ-16-13
SCRUM Master	Director del proyecto Integrador
Equipo Desarrollo	Fabrizio Ramírez

Definición de iteraciones

El presente proyecto tuvo un *Sprint* inicial (*Sprint 0*) de un mes para analizar la factibilidad del proyecto. Posteriormente, el equipo SCRUM definió el número y duración de los *Sprints*. Considerando el nivel de interacción con el equipo de investigación y que el equipo SCRUM dispondría solo de 4 horas diarias para desarrollar el producto, se determinó que se realizarían 3 iteraciones (*Sprints*) con una duración de un semestre académico, pues el

equipo estimó que sería el tiempo adecuado para entregar un incremento que genere valor al producto.

2.2.1 Sprint 0

Planificación del Sprint (Sprint Planning)

Este *Sprint* tuvo la duración de un mes. Se desarrolló una reunión de apertura para realizar la presentación del equipo de investigación y del proyecto PIJ-16-13. También se realizó la recolección de requerimientos iniciales del proyecto integrador. El equipo de investigación solicitó realizar varias pruebas de concepto respecto a las herramientas software y hardware disponibles e ideas del equipo de investigación. Además, se acordó plantear 3 posibles arquitecturas de software para ser evaluadas al final de este *Sprint*.

Lista de Producto (Product Backlog)

Durante la primera reunión con el equipo de investigación el equipo SCRUM obtuvo la lista de requerimientos iniciales, con dicha información se pudo elaborar la primera Lista del Producto (*Product Backlog*):

Tabla 2 Lista del Producto Sprint 0.

Orden (Prioridad)	**Elementos / Descripción**	**Estimación (Horas)**
1	Revisar la documentación oficial del Dispositivo Myo Armband.	6
2	Revisar la documentación producida por el equipo de investigación y la documentación técnica del sistema de reconocimiento de gestos del brazo humano.	14
3	Instalación de software propietario de Myo Armband, Matlab, Visual Studio, SDK's y configuración del sistema de reconocimiento de gestos.	6
4	Realizar una prueba de concepto utilizando el Matlab *Library Compiler*.	10
5	Realizar una prueba de concepto utilizando el Matlab *Application Compiler*.	10

6	Realizar una prueba de concepto del uso del SDK de Myo Sharp.	4
7	Realizar una prueba de concepto sobre el uso de *TCP/IP Sockets* en Matlab y el envío de datos.	10
8	Definir una trama para el envío de datos utilizando *sockets* de comunicación.	2
9	Con todas las pruebas de concepto realizadas anteriormente, plantear tres posibles arquitecturas de software.	12

Revisión del Sprint (Sprint Review)

En esta primera reunión de revisión del *Sprint*, todo el equipo de investigación se reunió para: revisar los resultados de las pruebas de concepto, revisar las tres arquitecturas propuestas y decidir cuál de las tres arquitecturas sería utilizada para el desarrollo del proyecto. A continuación, se muestra una tabla de resumen de las arquitecturas propuestas, una breve descripción, ventajas y desventajas.

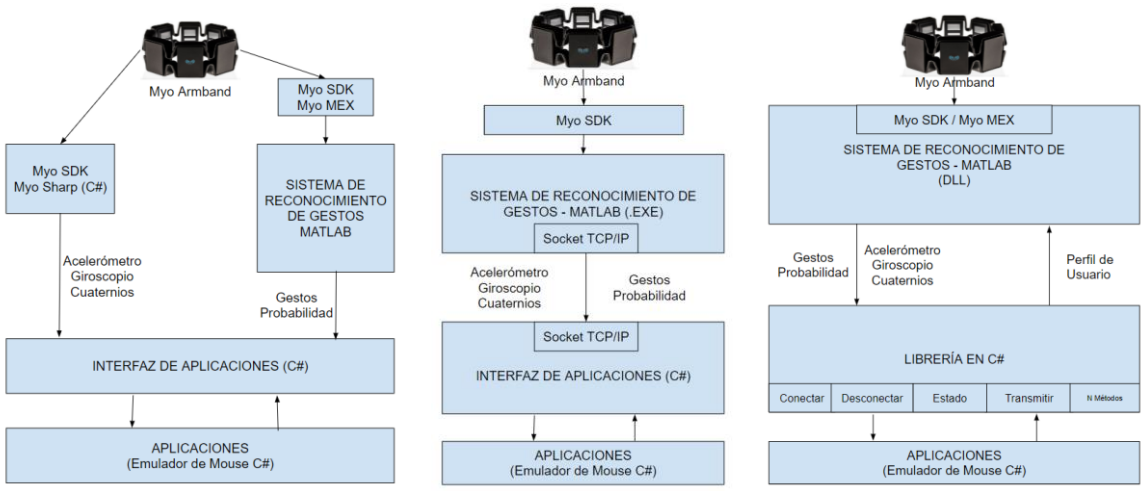

Figura 7 De izquierda a derecha, propuestas arquitectónicas 1, 2 y 3.

Tabla 3 Tabla comparativa entre las tres arquitecturas propuestas.

	Arquitectura 1	**Arquitectura 2**	**Arquitectura 3**
Descripción	El sistema de reconocimiento se compila como una librería o como un ejecutable. En paralelo	El sistema de reconocimiento de gestos es modificado para obtener datos espaciales del dispositivo Myo Armband e	El sistema de reconocimiento de gestos es modificado para obtener datos espaciales del dispositivo Myo Armband. El

	la interfaz en C# se conecta al dispositivo Myo Armband a través de Myo Sharp (SDK). Las aplicaciones prácticas reciben los datos de los gestos desde el sistema de reconocimiento y los datos espaciales directamente desde el Dispositivo Myo Armband.	implementa un Socket TCP/IP. El sistema de reconocimiento es compilado en forma de ejecutable (.exe). La aplicación en C# también implementa un Socket TCP/IP para recibir la información del sistema de reconocimiento en una trama predefinida.	sistema de reconocimiento es compilado en forma de librería para .Net(dll). La aplicación en C# referencia la librería y obtiene los datos espaciales y de gestos a través de los métodos que implementa.
Ventajas	La información espacial llega directamente del dispositivo Myo Armband y la frecuencia de extracción de datos puede ser definida por la aplicación en C#.	Implementación de Sockets TCP/IP tanto en Matlab como en C# es fácil. Se puede embeber el ejecutable en cualquier aplicación. Los sockets se instancian en Localhost lo cual reduce el *delay* en la transmisión. Todo la lógica matemática y el procesamiento se realiza en Matlab.	Se accede a las capacidades del sistema de reconocimiento como método o funciones. La integración a cualquier aplicación es más sencilla ya que se referencia como cualquier librería externa.
Desventajas	Existen 2 instancias del Objeto Myo Armband uno a través del SDK Myo Sharp y otro a través del SDK MYO MEX. Dependiendo del tipo de compilación elegida (.exe o .dll) adoptaría las ventajas y desventajas de las otras arquitecturas. Requiere instalación del Matlab Runtime en la computadora de destino.	Durante la primera ejecución de los ejecutables(.exe) se realiza una instalación en la carpeta temporal donde se copian todos los archivos necesarios para el funcionamiento del programa lo cual incrementa el tiempo de carga del sistema de reconocimiento de gestos. Requiere instalación del Matlab Runtime en la computadora de destino. No todas las funciones de Matlab son compilables.	Las funciones de Matlab compiladas tienen muchas sobrecargas, y solo pueden ser accedidas a través de la clase MWArray. Requiere instalación del Matlab Runtime en la computadora de destino. No todas las funciones de Matlab son compilables.

En esta reunión el equipo de investigación concluyó que la “Arquitectura 3” sería la óptima para el desarrollo del proyecto, pero necesitaría un mayor tiempo de análisis. Por este

motivo la “Arquitectura 2” fue seleccionada, además, es mucho más sencilla de implementar y facilitó la exploración de esta opción a mayor detalle. Luego de haber seleccionado una arquitectura la Lista de Producto (*Product Backlog*) fue actualizada.

Product Backlog (Actualizado)

Tabla 4 Lista del producto actualizada al final del Sprint 0.

Orden	Historia Usuario	Descripción	Estimación (Horas)	Estado
1	N/A	Compilar el sistema de reconocimiento de gestos del brazo humano para exponer sus funcionalidades.	74	Creado
2	N/A	Completar e implementar la arquitectura de Software seleccionada.	110	Creado
3	EUS01	Desarrollar un emulador de mouse en base a la arquitectura implementada.	110	Creado
4	N/A	Proponer y desarrollar una segunda aplicación en base a la arquitectura implementada.	110	Creado

2.2.2 Sprint 1

2.2.2.1 Planificación del Sprint (Sprint Planning)

En esta reunión el equipo SCRUM se reunió para elegir los *ítems* del *Product Backlog* que se realizaría durante el *Sprint* 1. El Objetivo de este *Sprint* (*Sprint Goal*) fue implementar la arquitectura de software seleccionada durante el *Sprint Review* anterior y desarrollar un primer prototipo del emulador de mouse.

El equipo de desarrollo definió las tareas necesarias para alcanzar el objetivo. Considerando que la implementación de la arquitectura de software son especificaciones técnicas no tiene asociada ninguna historia de usuario. A continuación, se presenta la Lista de Pendientes del *Sprint* 1 (*Sprint Backlog*).

Lista de Pendientes (Sprint Backlog)

Tabla 5 Lista de Pendientes del Sprint 1.

Orden	US	Descripción	Estimación (Horas)
	N/A	**Compilar el sistema de reconocimiento de gestos del brazo humano para exponer sus funcionalidades.**	40
1	N/A	Modificar el sistema de reconocimiento de gestos para que pueda recolectar datos espaciales del dispositivo Myo Armband.	6
2	N/A	Modificar el sistema de reconocimiento de gestos para transformar los datos espaciales en ángulos de navegación (Tait-Bryan).	14
3	N/A	Definir una trama para el envío y recepción de los datos.	4
4	N/A	Modificar el sistema de reconocimiento de gestos para implementar sockets *TCP/IP* y enviar la trama definida.	6
5	N/A	Compilar el sistema de reconocimiento utilizando el *Matlab Application Compiler.*	10
	N/A	**Implementar la arquitectura de Software seleccionada.**	40
6	N/A	Crear un módulo que gestione la comunicación con el sistema de reconocimiento utilizando *Sockets TCP/IP*.	8
7	N/A	Implementar un módulo para automatizar y parametrizar la ejecución en segundo plano de los ejecutables (archivos .exe).	8
8	N/A	Módulo para procesar los datos recibidos por el sistema de reconocimiento.	8
9	N/A	Implementar una interfaz gráfica para visualizar los datos recibidos por el sistema de reconocimiento de gestos.	16
	EUS01	**Desarrollar un emulador de mouse en base a la arquitectura implementada.**	40
10	US01	Implementar la funcionalidad para asociar el movimiento del antebrazo al desplazamiento del cursor.	30
11	US02	Implementar la funcionalidad para asociar los gestos reconocibles a acciones de mouse.	10

Historias de usuario

Historia de Usuario Épica		EUS01
Título	Desarrollar un Emulador de Mouse	
Descripción	Yo como usuario Quiero un emulador de mouse Para poder manejar el cursor utilizando el movimiento de mi antebrazo y los gestos de mi mano.	
Prioridad: Alta		Esfuerzo:40

Historia de Usuario		US01
Título	Asociar el movimiento del antebrazo al desplazamiento del cursor.	
Descripción	Yo como usuario requiero utilizar el movimiento de mi antebrazo para desplazar el cursor en la pantalla de mi computadora	
Prioridad: Alta		Esfuerzo: 30 horas
Criterios de Aceptación	Movimiento horizontal del antebrazo humano mueve horizontalmente el cursor. Movimiento vertical del antebrazo humano mueve verticalmente el cursor. La velocidad de desplazamiento del cursor es equivalente a la velocidad de movimiento del antebrazo. El movimiento del cursor en la pantalla es fluido.	

Historia de Usuario		US02
Título	Asociar los gestos reconocibles a acciones de mouse.	
Descripción	Yo como usuario requiero utilizar los gestos reconocibles por el sistema de reconocimiento de gestos para ejecutar acciones de mouse	
Prioridad: Alta		Esfuerzo: 10 horas
Criterios de Aceptación	Cada uno de los gestos reconocibles se encuentra asociado a un evento de mouse.	

2.2.2.2 Implementación

En esta sección se mostrarán los detalles relevantes de la implementación de los ítems del Sprint Backlog.

Trama para envío de datos:

Para poder realizar una comunicación exitosa entre el “sistema de reconocimiento de gestos” y cualquier otra aplicación primero es necesario definir un formato para él envió y recepción de datos, el cual debe ser estándar y escalable.

El sistema de reconocimiento de gestos utilizado, realiza un “reconocimiento" cada 0.2 segundos. Este valor es configurable, entre más cercano a 0, mayor la interacción en tiempo real, pero también mayor la cantidad de recursos computacionales requeridos. Cada vez que se realiza un reconocimiento, el modelo debe obtener información del giroscopio, acelerómetro y magnetómetro, los cuales son provistos por el dispositivo Myo Armband, procesar los datos pertinentes y organizarlos en una trama. El modelo define una función que envía la trama a la interfaz en C# a través de sockets *TCP/IP*.

Tabla 6 Estructura de la trama de datos definida para la arquitectura.

Nombre	**Significado**	**Representación**	**Rango + Unidad**
Gesto	Identificador único asignado a cada tipo de gesto identificable por el modelo.	“gest(relax,WI,WO,F,O,P)|”	0: No gesto 1: Wave in 2: Wave out 3: Fist 4: Open 5: pinch
Probabilidad	Probabilidad que asigna el modelo a cada gesto, dada una muestra.	“prob(0-1)”	0: cuando no se detectan gestos. 0.6 a 1 cuando si se detectan gestos con variación de 0.2 [0.6; 0.8; 1.0]
Acelerómetro	Coordenadas que representan la aceleración de movimiento del dispositivo Myo Armband.	“acel(xyz)[m/s^2]”	Vector de tres elementos [x, y, z]. Medidos en metros sobre segundos al cuadrado (m/s2)
Giroscopio	Coordenadas que representan la orientación en el espacio del dispositivo Myo Armband.	“gyro(xyz)[rad/s]”	[x, y, z] Vector expresado en Radianes sobre segundo

Ángulos de Tait-Bryan	Ángulos que permiten describir la orientación de un objeto en un sistema de 3 dimensiones.	“ang(yaw,pitch,roll) _(XYZo)[deg]”	Tres ángulos en grados sexagesimales. Yaw – Pitch – Roll

De acuerdo a la tabla anterior, la trama se estructura en el mismo orden en el que se presentan los elementos: primero se coloca la “representación”, luego el caracter separador “|” y finalmente el valor de la “Unidad”.

A continuación, se muestra una trama de ejemplo:

gest(relax,WI,WO,F,O,P)|1 |prob(0-1)|1 |acel(xyz)[m/s^2]|0.10352 -0.084961 0.99121 |gyro(xyz)[rad/s]|-2.4375 -0.5 0.5 |ang(yaw,pitch,roll)_(XYZo)[deg]|41 7 6

En el ejemplo podemos observar que el gesto detectado por el modelo es 1 (Fist), con un valor de 1 en la probabilidad (es el valor retornado cuando se reconoce un gesto), tres valores del acelerómetro en m/s^2 (x=0.10352, y=-0.084961, z=0.99121), tres valores del giroscopio en rad/s (x=-2.4375, y=-0.5, z=0.5), y finalmente tres valores para los ángulos de Tait-Bryan en grados (yaw=41, pitch=7, roll=6).

Desplazamiento del cursor:

Para emular el desplazamiento del cursor en la pantalla se usaron los ángulos de navegación (Tait-Bryan) enviados por el sistema de reconocimiento de gestos. Como el cursor se desplaza en una pantalla de dos dimensiones solo se utilizaron los ángulos “Yaw” para describir movimiento Horizontal y “Pitch” para el movimiento vertical.

Este primer prototipo del emulador, toma el primer valor de los ángulos enviados por el sistema de reconocimientos como la posición inicial. La posición inicial se entiende como el centro de la pantalla. Luego, a partir de este primer valor se calculan los ángulos máximos y mínimos, para ello se suma y resta un valor obtenido experimentalmente (ángulo promedio que puede describir el movimiento del antebrazo humano). Estos límites son asociados a los 0 pixeles de la pantalla (mínimo) y la resolución máxima tanto en altura como en ancho.

Finalmente, cada valor de los ángulos de navegación que se recibe ubica el cursor en un punto de la pantalla proporcional a los valores calculados anteriormente.

Figura 8 Primera aproximación para el desplazamiento del emulador de mouse. Punto de referencia central, ancho y largo de la pantalla.

Eventos de Mouse

Es necesario asociar cada gesto reconocido por el modelo a un evento de mouse. Primero se listará todos los eventos que puede realizar un mouse convencional: Un mouse básicamente puede realizar 2 acciones (sin considerar el desplazamiento y el *scroll*):

- Clic derecho
- Clic izquierdo

Entendiendo "clic" como la acción de presionar y soltar un botón del dispositivo mouse. Sin embargo, con estas acciones se pueden realizar otros eventos que también son necesarios para la implementación del emulador de mouse y considerando las limitaciones del modelo es necesario tratarlas individualmente, es decir asignarles un gesto diferente. Las acciones derivadas son las siguientes:

- Presionar un botón (*Mouse Down*)
- Soltar un botón (*Mouse Up*)
- Doble Click Derecho (el izquierdo no tiene utilidad)

Para asignar eventos de mouse a cada gesto se ha utilizado la función "Mouse_Events" [34], la cual es propia del *framework* WinForms, pues ofrece la posibilidad de realizar las cuatro acciones básicas definidas anteriormente, además permite la combinación de estas acciones para emular el resto de Eventos:

```
public const int MOUSEEVENTF_LEFTDOWN = 0x02;
public const int MOUSEEVENTF_LEFTUP = 0x04;
public const int MOUSEEVENTF_RIGHTDOWN = 0x0008;
public const int MOUSEEVENTF_RIGHTUP = 0x0010;
```

Figura 9 Definición de eventos de mouse en C#.

Actualmente el sistema de reconocimiento puede identificar 6 gestos, a cada uno de ellos se le ha asignado un evento de mouse de acuerdo a la siguiente tabla:

Tabla 7 Asociación entre Gestos reconocibles y eventos de Mouse

Gesto	Acción	Comentarios / Condiciones
0: No gesto	Ninguna	Ninguno.
1: Wave in	*Drag*	Realiza un *Mouse Down y mouse clic*.
2: Wave out	*Drop*	Realiza un *Mouse up.*
3: Fist	*Left Click*	Realiza un *Mouse down, mouse clic y mouse up* Izquierdo.
4: Open	*Right Click*	Realiza un *Mouse down, mouse clic y mouse up* Derecho.
5: pinch	*Double Left Click*	Realiza un *Mouse down y mouse up* dos veces.

Interfaz gráfica de la aplicación

El objetivo de esta interfaz gráfica, en su primera versión, es poder visualizar la información recibida desde el sistema de reconocimiento de gestos y también poder gestionar el ejecutable.

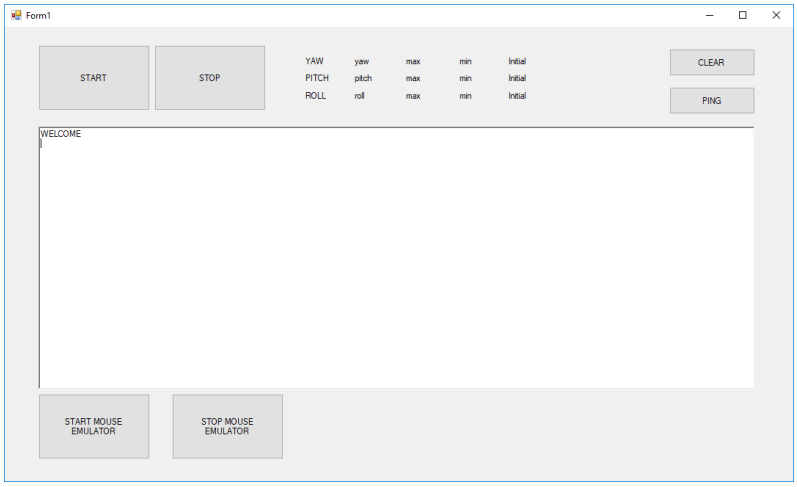

Figura 10 Primera versión de la Interfaz Gráfica que muestra los datos recibidos del ejecutable de grabación de gestos en tiempo real.

2.2.2.3 Pruebas

Al final de la fase de implementación, se realizaron pruebas de integración para verificar que la interfaz de aplicaciones en C# reciba íntegramente la trama enviada por el módulo de reconocimiento de gestos (Ejecutable).

2.2.2.4 Revisión del Sprint (Sprint Review)

En esta revisión se reunió nuevamente al equipo de investigación. Considerando que no todos los miembros del equipo tienen un *background* en ingeniería de software, se decidió utilizar una *marketecture* (ver **Figura 11**) para la presentación. En esta reunión se presentó el incremento que consta de: la implementación de la arquitectura de software y la primera versión del emulador de mouse.

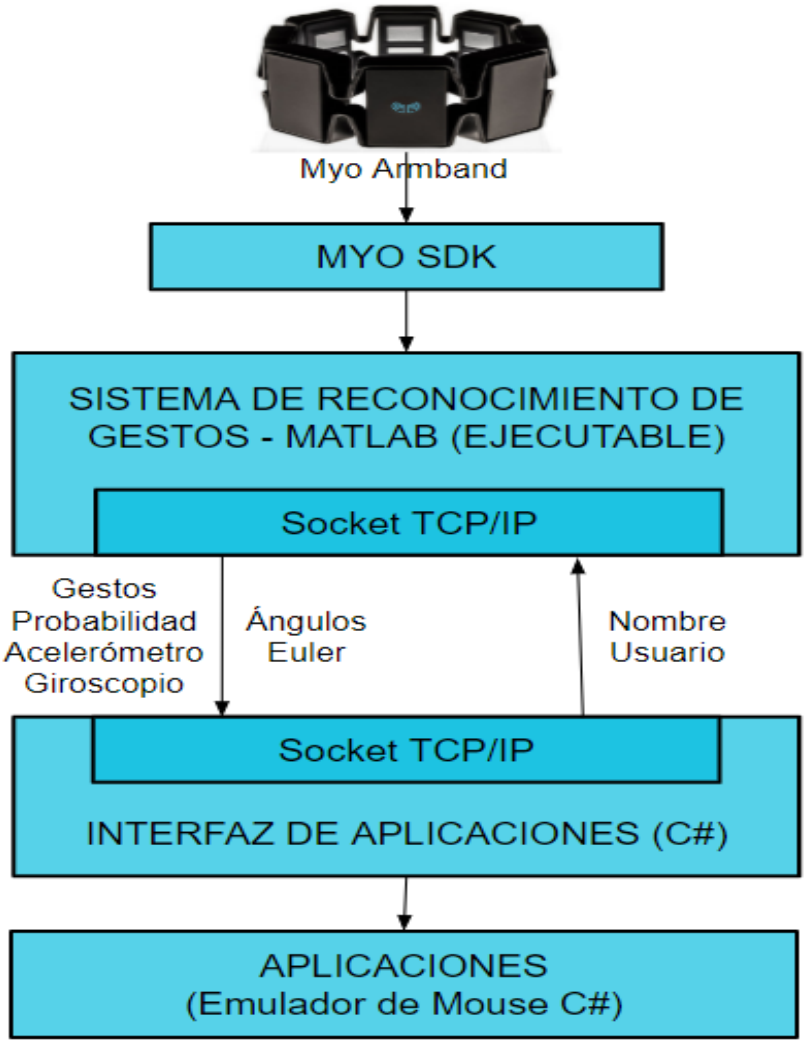

Figura 11 *Marketecture* de la Arquitectura de Software durante el *Sprint* 1.

Al final de la reunión, el equipo SCRUM realizó la actualización de la Lista del Producto (*Product Backlog*).

Tabla 8 Lista del producto actualizada al final del Sprint 1.

Orden	Historia Usuario	Descripción	Estado
1	N/A	Compilar el sistema de reconocimiento de gestos del brazo humano para exponer sus funcionalidades.	Terminado
2	N/A	Implementar la arquitectura de Software seleccionada.	Parcialmente completo
3	EUS01	Desarrollar un emulador de mouse en base a la Arquitectura implementada.	Parcialmente completo
4	EUS02	Implementar un módulo de gestión de usuarios.	Creado
5	N/A	Proponer y desarrollar una segunda aplicación en base a la arquitectura implementada	Creado

2.2.2.5 Retrospectiva de Sprint (Sprint Retrospective)

En este evento el equipo de desarrollo realizó una autoevaluación del incremento realizado en el Sprint anterior e identificó las siguientes oportunidades de mejora:

Ángulos de Tait-Bryan:

El emulador de mouse toma el primer ángulo que envía el sistema de reconocimiento como su posición inicial y central en la pantalla. Respecto a este primer ángulo, se determina la amplitud máxima y mínima tanto del ángulo *Yaw* como *Pitch*. Este método falla cuando el primer ángulo "*Yaw*" enviado por el sistema de reconocimiento está muy cerca de sus límites, pues entonces el valor máximo o mínimo desbordan el límite del ángulo y se genera un conflicto con la proporción respecto al movimiento del cursor en pantalla. Tomando en cuenta este error, el equipo SCRUM propuso investigar otro método para simular el movimiento del cursor.

Fluidez del emulador:

El sistema de reconocimiento envía datos cada 0.2 segundos. Lamentablemente este valor limita el movimiento del cursor, pues a menor frecuencia de envió, el movimiento del mouse

se ve entrecortado, es decir, visualmente el cursor salta en la pantalla. Para solucionar este problema, se puede modificar el sistema de reconocimiento para que envíe con mayor frecuencia la trama de datos.

2.2.3 Sprint 2

2.2.3.1 Planificación del Sprint (Sprint Planning)

En este evento el equipo SCRUM se reunió para elegir los ítems del *Product Backlog* (actualizado) que se realizarían durante el *Sprint* 2. El Objetivo de este *Sprint (Sprint Goal)* fue implementar un módulo de gestión de usuarios, mejorar el emulador de mouse y proponer una segunda aplicación.

El equipo de desarrollo definió las tareas necesarias para alcanzar el objetivo. Para la creación del módulo de gestión de usuarios y la mejora del emulador de mouse, sí se recolectaron Historias de Usuarios. A continuación, se presenta la Lista de Pendientes del *Sprint* 2 (*Sprint Backlog*).

Lista de pendientes del Sprint (Sprint Backlog)

Tabla 9 Lista de pendientes del Sprint 2.

Orden	US	Descripción	Estimación (Horas)
	US03	**Crear Usuario.**	**20**
1		Crear un árbol de directorios con el nombre del usuario para almacenar las señales EMGs recolectadas.	2
2		Crear una interfaz gráfica para la creación de usuarios.	4
3		Crear una interfaz gráfica que permita cambiar de usuario.	4
4		Crear una interfaz gráfica para borrar usuarios y volver a entrenar un usuario.	8
5		Crear una interfaz informativa antes de la grabación de gestos.	2
	N/A	**Módulo de grabación de gestos de usuario.**	**40**
6		Implementar la función de grabación de señales EMGs del usuario en Matlab.	14
7		Implementar la función de creación de estructuras de datos para el usuario en Matlab.	8

8		Implementar una interfaz de comunicación utilizando Sockets *TCP/IP* para el módulo de Grabación de Gestos en Matlab.	8
9		Parametrizar la función de grabación de gestos para ajustar el tiempo de grabación y el número de muestras requeridas.	8
10		Compilar el módulo de grabación de gestos.	2
	US04	**Módulo de video entrenamiento**	**30**
11		Crear una interfaz que muestre un video tutorial para la grabación de gestos del usuario.	15
12		Implementar un módulo para la comunicación con el módulo de Grabación de Gestos.	15
	US05	**Módulo de Configuración de grabación de gestos.**	**10**
13		Interfaz gráfica para editar el número de muestras requeridas y el tiempo de grabación.	8
14		Opción para habilitar la visibilidad de la Consola del ejecutable (Depuración).	2
	N/A	**Mejora Emulador de Mouse.**	**20**
15		Modificar el sistema de reconocimiento para enviar con mayor frecuencia los datos espaciales a la interfaz.	2
16		Mejorar el proceso de obtención de los ángulos Tait-Bryan.	2
17		Modificar el algoritmo para el desplazamiento del cursor.	16
	US06	**Módulo de Configuración Emulador de Mouse.**	**10**
18		Asociar los gestos reconocibles a las acciones de mouse.	4
19		Crear funcionalidad para modificar la sensibilidad del cursor.	4
20		Crear funcionalidad para restablecer la configuración por defecto.	2

Historias de usuario

Historia de Usuario		US03
Título	Crear Usuario	
Descripción	Yo como usuario Quiero crear un perfil Para poder utilizar el sistema de reconocimiento de gestos y sus aplicaciones.	

Prioridad: Media		Esfuerzo: 20
Criterios de Aceptación	Validar que no exista otro usuario con el mismo nombre. Validar que el nombre no contenga caracteres especiales no admitidos por el sistema de archivos. Se crea un directorio con el nombre del usuario ingresado. Se crea el árbol de directorios para almacenar las señales EMGs del usuario.	

Historia de Usuario		US04
Título	Módulo de video entrenamiento	
Descripción	Yo como usuario Quiero una interfaz Para grabar mis gestos	
Prioridad: Alta		Esfuerzo: 30
Criterios de Aceptación	Existe una interfaz que muestra un video tutorial. El video tutorial y la barra de carga se encuentran sincronizados. Existen *shortcuts* para controlar la interfaz de video entrenamiento.	

Historia de Usuario		US05
Título	Módulo de Configuración de Ejecutables	
Descripción	Yo como usuario Quiero una interfaz de configuración Para cambiar los parámetros del ejecutable de Grabación de Gestos	
Prioridad: Media		Esfuerzo: 10
Criterios de Aceptación	Se puede editar el tiempo de grabación de señales EMGs. Se puede editar el número de repeticiones por cada gesto. Se puede habilitar la visibilidad de la Consola del ejecutable (Depuración). Se puede restablecer la configuración por defecto.	

Historia de Usuario		US06
Título	Módulo de Configuración del emulador de mouse	
Descripción	Yo como usuario Quiero una interfaz de configuración Para cambiar los controles del emulador de mouse	

Prioridad: Media		Esfuerzo: 10
Criterios de Aceptación	Se puede cambiar la asociación entre gestos reconocibles y acciones de mouse. Se puede modificar la sensibilidad del cursor. Se puede restablecer la configuración por defecto.	

2.2.3.2 Arquitectura de Software

En la segunda iteración el diseño arquitectónico fue completado y documentado. La arquitectura seleccionada se basa principalmente en el uso de la herramienta *Matlab Application Compiler* para transformar el sistema de reconocimiento de gestos en archivos ejecutables. Utiliza *Sockets TCP/IP* para dotar de un canal de comunicación a los ejecutables. Y utiliza el sistema de archivos de Windows como método de almacenamiento de datos. A continuación, se realizará una descripción detallada del diagrama de componentes UML de la Arquitectura de Software.

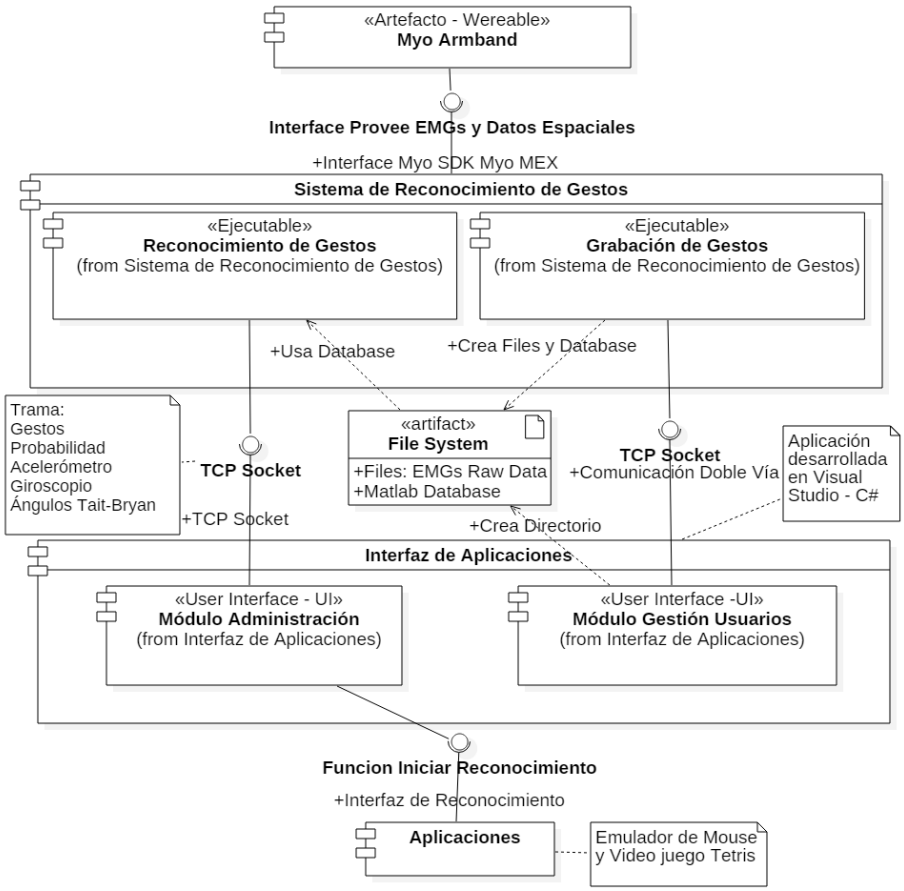

Figura 12 Diagrama de componentes UML de la arquitectura de software.

Descripción detallada del diagrama de Componentes

Myo Armband: El dispositivo Myo Armband establece conexión con el computador del usuario a través de un “Dongle USB” y el programa “Myo Connect”.

Myo SDK: El sistema de reconocimiento de gestos interactúa con el dispositivo Myo Armband a través de “MyoMex” el cual es un *kit* de desarrollo de software (SDK) para Matlab.

Sistema de reconocimiento de gestos (MATLAB)

- **Módulo de Reconocimiento de Gestos (Ejecutable):** Contiene la función de reconocimiento de gestos. Este módulo interactúa directamente con el dispositivo Myo Armband a través de Myo MEX. Para su funcionamiento necesita los datos EMGs de un usuario, los cuales extrae de un directorio compartido. Este módulo genera información de gestos y también se encarga de recolectar información espacial del dispositivo Myo Armband, sintetiza toda esta información en una trama previamente definida (ver **Tabla 6**). Finalmente, envía periódicamente estas tramas a la Interfaz de Aplicaciones a través de un socket de comunicación.

- **Módulo de Grabación de Gestos (Ejecutable):** Este módulo expone dos funciones del sistema de reconocimiento de gestos. Primero, la función de grabar las señales EMGs de un usuario y etiquetarlo de acuerdo al tipo de gesto y número de iteración realizada, esta información se almacena en un directorio determinado. Segundo, la función de crear estructuras de datos que agrupen todas las grabaciones realizadas de un mismo gesto por cada usuario, esta información es almacenada en el mismo directorio. Este módulo utiliza el SDK para realizar la conexión con Myo Armband y un *socket TCP/IP* para comunicarse con el módulo de gestión de usuarios.

Interfaz de Aplicaciones

- **Módulo Administración:** Este módulo puede ser implementado en cualquier lenguaje de programación. Se encarga de iniciar o terminar el ejecutable de reconocimiento de gestos. Implementa un *socket TCP/IP* para comunicarse con el módulo de reconocimiento de gestos, debe procesar la información recibida en la

trama. Cada vez que recibe una trama, separa los datos y los almacena en variables globales para que sean utilizados por las aplicaciones prácticas.

- **Módulo de gestión de usuarios:** Este módulo también puede ser implementado en cualquier lenguaje de programación. Se encarga de iniciar o terminar el ejecutable de grabación de gestos. Implementa un *socket TCP/IP* para comunicarse con el módulo de Grabación de Gestos. A través de este medio, solicita la ejecución de alguna de las 2 funciones que expone este módulo. Debe implementar una interfaz gráfica que guíe al usuario en el proceso de recolección de gestos, la interacción entre la interfaz gráfica y el módulo de Grabación de Gestos se detalla en un diagrama de secuencia, ver **Figura 13**.

Aplicaciones Prácticas: Aplicaciones de software de diferentes propósitos. Las aplicaciones deben ser desarrolladas en el mismo lenguaje utilizado para la implementación del módulo Interfaz de Aplicaciones. Consume directamente la información procesada por el módulo de administración, específicamente los datos de gestos reconocidos y los datos espaciales.

Sistema de Archivos: La arquitectura utiliza el sistema de archivos de Windows para almacenar información necesaria tal como los datos EMGs o los archivos ejecutables. El Módulo de Gestión de Usuarios es el encargado de crear el árbol de directorios especificado en la **Figura 14**. Y el módulo de Grabación de Gestos es el encargado de guardar los datos EMGs en este árbol de directorios.

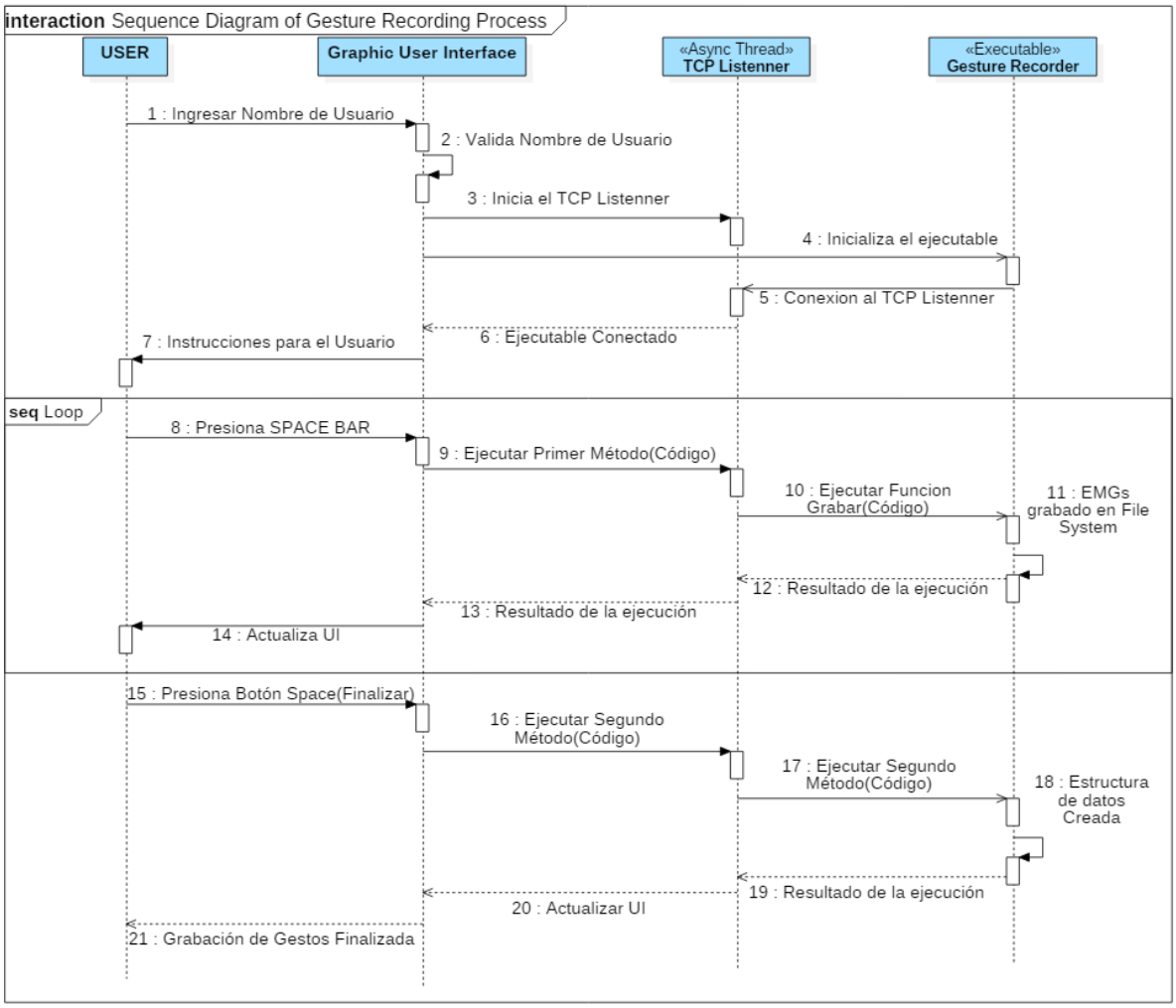

Figura 13 Diagrama de secuencia UML entre el usuario, la interfaz de video entrenamiento, *socket TCPI/IP* y el ejecutable de grabación de gestos.

2.2.3.3 Implementación de la Arquitectura

Sistema de Archivos

Para desarrollar un módulo de gestión de usuarios es necesario pensar en un método para almacenar información relevante, en el caso del sistema de reconocimiento, los datos que se deben almacenar son las señales EMGs de cada uno de los gestos del usuario. Para elegir este método se deben considerar los siguientes puntos:

- Se trata de una aplicación de escritorio *offline*, el almacenamiento debe ser local.
- La cantidad de usuarios por instalación es limitada.
- Las estructuras de datos que se pretende almacenar son archivos de Matlab (.mat).
- Los ejecutables deben tener fácil acceso de lectura y escritura a los datos.

- Los ejecutables al ser desarrollados en Matlab usan *workspaces* (*Paths*) basados en el sistema de archivos del sistema operativo.

Considerando los puntos anteriores, se decidió utilizar el sistema de archivos propio del sistema operativo Windows para poder almacenar la información pertinente. De este modo el ejecutable de reconocimiento podría leer, y el de grabación de gestos podría escribir sobre el mismo directorio. Para mantener una organización correcta, se ha decidido implementar programáticamente el siguiente árbol de directorios.

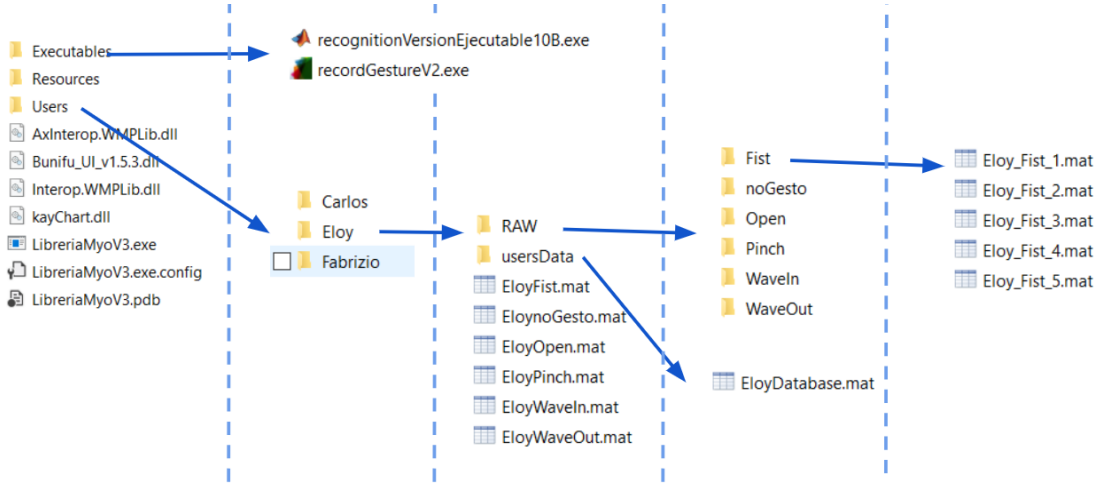

Figura 14 Árbol de directorios para el almacenamiento de datos EMGs (.mat) y Archivos Ejecutables.

Ejecutable de Grabación de Gestos

Para gestionar perfiles de usuarios es necesario exponer dos funciones del sistema de reconocimiento de gestos. Primero, la funcionalidad de grabar las señales EMGs de un usuario al realizar un gesto. Segundo, la funcionalidad de transformar las señales EMGs recolectadas en una estructura de datos (archivo .mat) para que puedan ser utilizadas por el sistema de reconocimiento de gestos.

Para la creación de este ejecutable se utilizaron *sockets TCP/IP*, para establecer comunicación con el módulo de gestión de usuarios de la interfaz de aplicaciones. El ejecutable contiene ambas funcionalidades grabación de gestos y creación de estructuras de datos. Similar a ofrecer un servicio, el ejecutable se limita a esperar la petición de la interfaz de video entrenamiento para realizar una de las dos funciones. Para que el ejecutable sepa cuál de las funcionalidades debe efectuar, es necesario definir valores de entrada. A continuación, se muestran un resumen de las entradas y códigos de comunicación del

ejecutable de Grabación de Gestos:

Tabla 10 Valores de entrada que acepta el ejecutable de Grabación de Gestos.

Función	Primer Dígito	Segundo Dígito	Tercer Dígito
Grabación de gestos	1: Indica que se ejecutará la función de grabación de gestos.	1: Wave in 2: Wave out 3: Fist 4: Open 5: Pinch 6: No gesto	1-n Indica el número de iteración con el cual se etiquetará el gesto.
Creación de estructura de datos	2: Indica que se ejecutará la función de creación de estructura de datos	1-n Indica el número de iteración realizados por cada gesto.	No aplica

Módulo de Gestión de usuarios en C#

Este módulo tiene la finalidad de iniciar e interactuar con el ejecutable de Grabación de Gestos. Ofrece al usuario una interfaz gráfica intuitiva que guiará al usuario durante todo el proceso de grabación de gestos. Para la creación de este módulo fue necesario crear distintos recursos visuales tales como imágenes y video tutoriales. En resumen, este módulo contiene las siguientes interfaces:

Nuevo Usuario: Solicita el nombre de Usuario que se utilizará como entrada en el ejecutable de Grabación de Gestos y también se utilizará para etiquetar las señales EMGs.

Figura 15 Interfaz Gráfica Nuevo Usuario

Instrucciones Iniciales: La interfaz gráfica detalla una secuencia de indicaciones que el usuario debe seguir para que pueda realizar correctamente la grabación de gestos. Incluye imágenes y videos demostrativos.

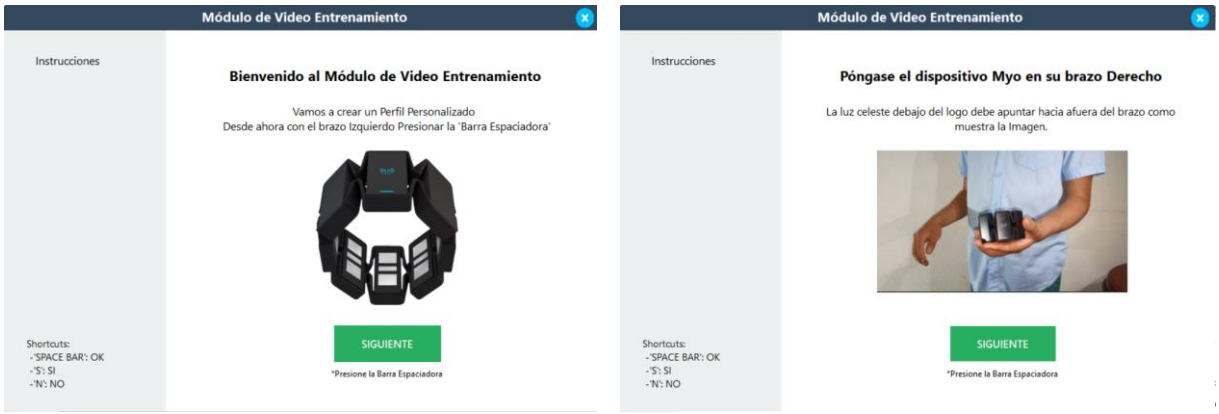

Figura 16 Instrucciones y videos tutoriales antes de iniciar con la grabación de gestos.

Video entrenamiento: Esta interfaz interactúa directamente con el ejecutable de grabación de gestos. Las funciones específicas de este módulo se listan a continuación en orden de ejecución:

Cuando este módulo es instanciado lo primero que realiza es crear un Hilo (Thread) asíncrono, al cual se le asigna la función de levantar un *TCP Listenner* en *localhost* (127.0.0.1) con puerto 14000. Este *socket TCP/IP* se encargará de recibir conexiones de clientes.

A continuación, el Hilo principal (UIThread – Interfaz gráfica) inicia el ejecutable de Grabación de Gestos que se encuentra en la carpeta “Executables”. Este programa puede ser ejecutado mostrando la consola de Windows para propósitos de *Debug* o también se puede ocultar a través del módulo de configuración. El ejecutable de Grabación de Gestos crea un *socket TCP* con *NetworkRol* “cliente” que realizará la conexión con el *TCP Listener*, que ya se encuentra funcionando en el *Thread* Asíncrono. Esto permitirá la comunicación entre ambos componentes.

La interfaz gráfica de este módulo está diseñada para guiar al usuario en el proceso de grabar sus gestos sin ayuda de terceros, es decir, la aplicación primero demostrará al usuario que gesto debe realizar:

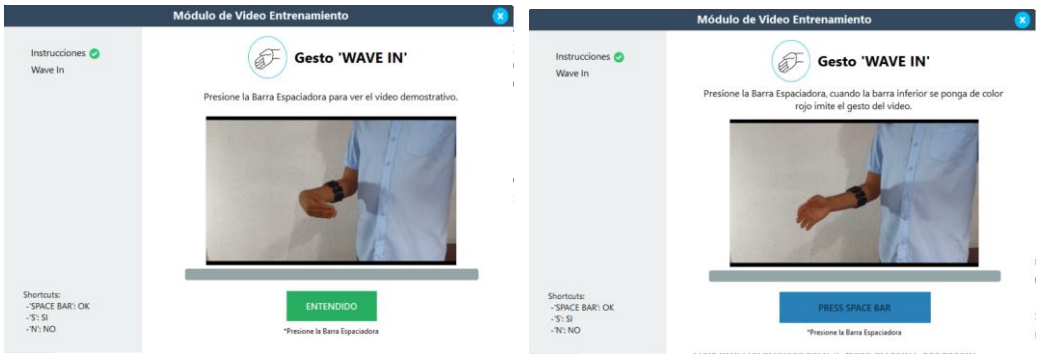

Figura 17 Interfaces gráficas del módulo de video entrenamiento.

Luego el usuario decidirá cuándo comenzar la grabación. Para ello se ha implementado una interfaz gráfica intuitiva que guiará al usuario durante este proceso. La interfaz consta de los siguientes elementos:

- En la parte superior se muestra una etiqueta que contendrá texto descriptivo respecto al gesto que el usuario debe realizar, o la acción necesaria para comenzar con la grabación (Presionar la Barra espaciadora).
- A la izquierda de la pantalla se muestra la lista de los gestos reconocibles por el sistema. Además de la imagen del gesto correspondiente, cuando se ha completado un número determinado de repeticiones por cada gesto, aparecerá un *check* al lado del nombre del gesto para indicar que se ha culminado con la grabación del mismo.

En el centro-izquierdo de la pantalla se muestran 3 elementos:

- Un reproductor multimedia donde se reproducirá un video demostrativo del gesto a realizar. La duración de los videos ha sido calculada para sincronizarse con el tiempo de grabación de gestos (Este valor es configurable).
- Una barra de progreso cuya duración es de 600 milisegundos más el tiempo seleccionado en la configuración del ejecutable (por defecto es de 2000 milisegundos). La barra de progreso inicia en color azul y a los 800 milisegundos cambia a color naranja lo cual indica que el usuario debe iniciar la realización del gesto. Todos estos valores son editables desde el módulo de configuración
- Botón “Presione Space Bar”: este elemento concentra gran parte de la funcionalidad de este módulo. Cuando se presiona, comunica al ejecutable de Grabación de Gestos, a través del *socket*, que se va a ejecutar una de las funciones que este expone. En principio, ejecutará la función de grabación hasta realizar todas las repeticiones necesarias.

Figura 18 Interfaz de Video Entrenamiento, video tutorial en reproducción.

Luego de completar con la grabación de un gesto, se mostrará un mensaje de confirmación, donde se preguntará al usuario si el gesto realizado fue correcto, o en su defecto si desea repetirlo. Esta decisión queda a criterio del usuario.

Figura 19 Mensajes de confirmación, luego de haber ejecutado la grabación de un gesto.

Cuando se haya grabado todos los gestos necesarios, el texto del botón cambia a “Finalizar”, el cual ejecutará la segunda función del ejecutable. Creará las estructuras de datos y registrará al usuario como predeterminado.

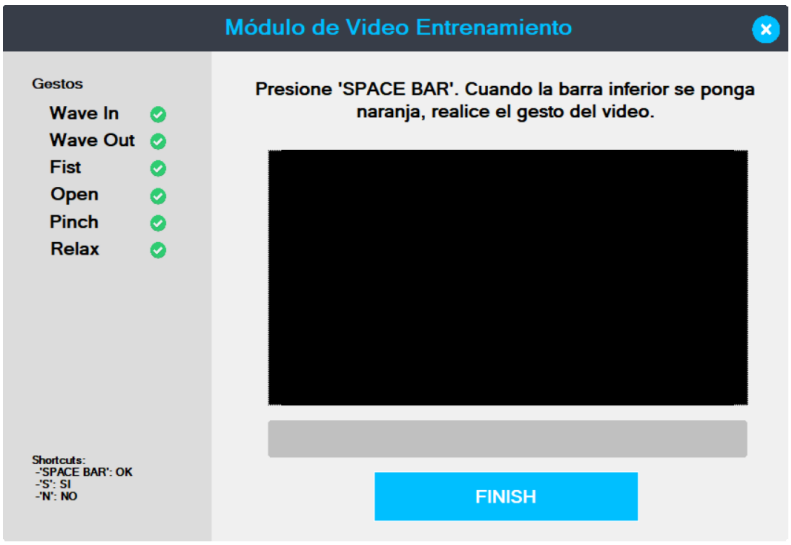

Figura 20 Interfaz gráfica luego de terminar la recolección de datos EMGs.

Módulo CRUD Usuario

Este módulo ofrece las operaciones CRUD (*Create, Read, Update, Delete*) para la gestión de usuarios. Tiene la finalidad de mostrar información del usuario actual: el nombre y la fecha de recolección de datos. Este último dato es relevante, pues según lo explicó el creador del sistema de reconocimiento de gestos, es recomendable que el usuario actualice los datos de sus gestos cada cierto tiempo (una semana).

Figura 21 Interfaz gráfica CRUD Gestión Usuarios.

User settings

Es una propiedad del Framework .NET (C#) que permite almacenar y recuperar dinámicamente la configuración de una propiedad u otro tipo de información para una aplicación. También nos permiten mantener las aplicaciones personalizadas y las preferencias en la computadora del usuario final. Esta propiedad se ha utilizado para gestionar las configuraciones del usuario, guardando las siguientes preferencias:

- El nombre del usuario actual o último usuario utilizado en la aplicación.
- Los códigos de los gestos que serán utilizados para asociarlos con las acciones del emulador de mouse.
- *Status Debug Mode* y Sensibilidad del emulador de mouse.
- Número de iteraciones y el tiempo de grabación requeridos.

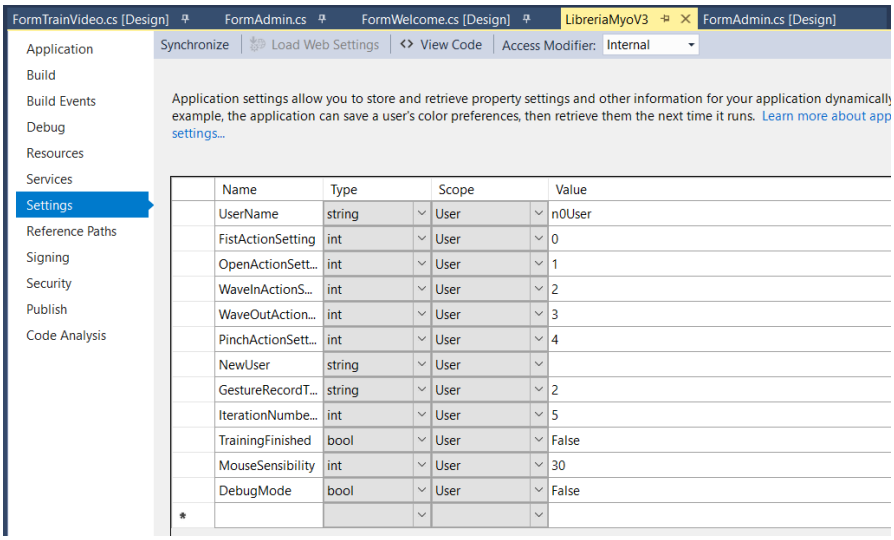

Figura 22 User Settings, valores de los módulos de configuración.

Módulo de configuración.

Esta interfaz ha sido diseñada para que el usuario pueda configurar algunas opciones respecto a los ejecutables o el emulador de mouse. Utiliza los "*User Setting*" para almacenar permanentemente la configuración del usuario. Es posible configurar los parámetros de entrada del ejecutable de grabación de gestos, la asociación entre gestos y eventos de mouse y el modo Debug que permite o no visualizar las consolas de Windows.

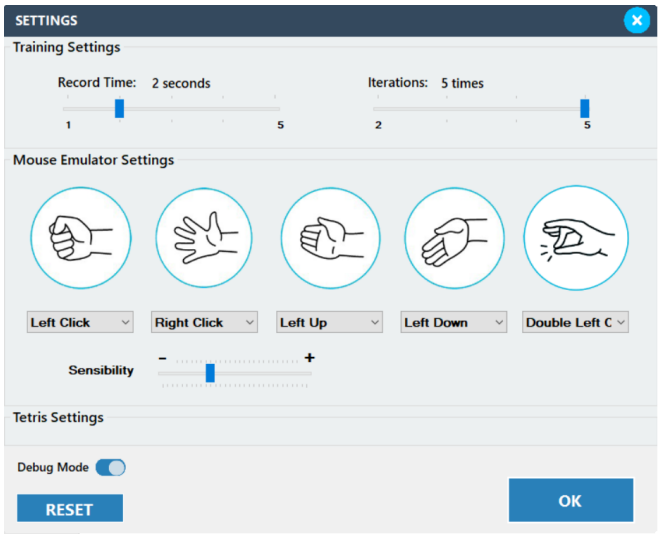

Figura 23 Interfaz gráfica del módulo de configuración de los ejecutables y emulador de mouse.

Mejoras a la primera iteración

Uno de los objetivos del segundo *Sprint* es mejorar las funcionalidades existentes. A continuación, se presenta un resumen de los cambios realizados a las aplicaciones de la primera iteración.

Módulo Principal:

La interfaz gráfica del Módulo de Administración fue mejorada con el fin de que esta sea de mayor ayuda durante las pruebas de integración y el desarrollo de las aplicaciones prácticas. Ya que permite monitorear al sistema de reconocimiento de gestos en tiempo real.

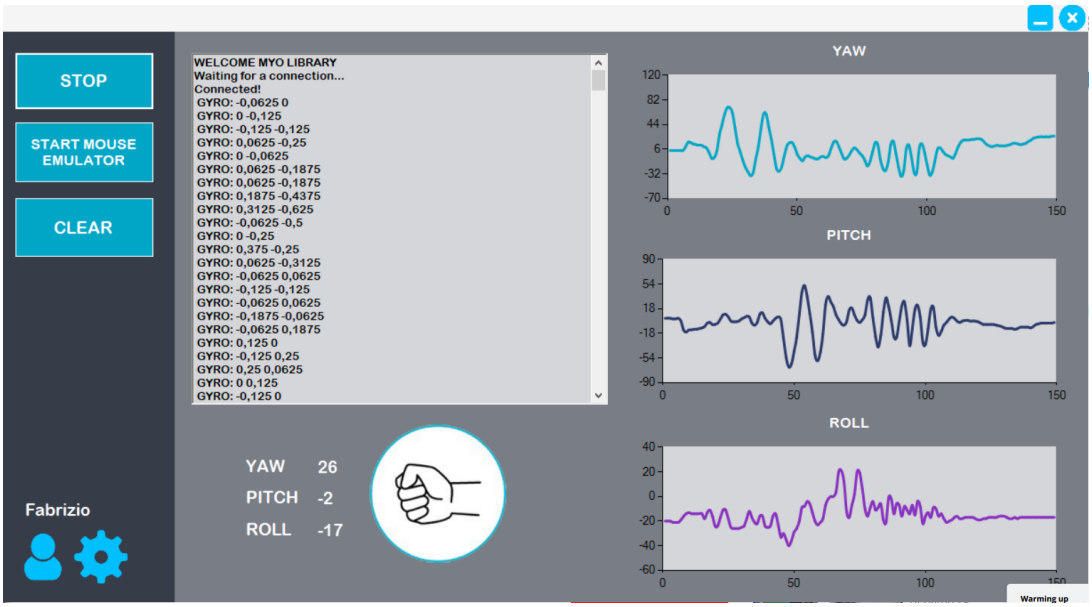

Figura 24 Interfaz gráfica del Módulo de Administración en ejecución.

Emulador de mouse:

Como se explicó en la retrospectiva del *sprint* 1, el funcionamiento del emulador de mouse no era muy confiable. Por este motivo se decidió reestructurar el funcionamiento del emulador de mouse. Para ello, nos basamos en el algoritmo utilizado por Thalmic Labs para controlar el emulador de mouse que viene incluido con el dispositivo Myo Armband. Este algoritmo está basado en la variación de los ángulos de Tait-Bryan (Deltas).

En resumen, el sistema de reconocimiento de gestos envía el valor actual de los ángulos de Tait-Bryan cada 0.2 segundos. El módulo Principal en C# se encarga de calcular la variación de cada ángulo (Yaw, Pitch) y almacenar esta información en variables globales. Se calcula el valor de desplazamiento del cursor, multiplicando las deltas por la densidad de la pantalla y un factor de sensibilidad extraído del giroscopio.

Las ventajas de este algoritmo son varias: no existe un punto inicial único, es decir, ahora el usuario puede moverse de la posición inicial sin preocuparse por exceder los límites de los ángulos. La precisión y rendimiento del emulador de mouse ha mejorado notablemente respecto a la primera iteración. Sin embargo, aún no alcanza la precisión del emulador desarrollado por Thalmic Labs. Aunque, es importante mencionar que debido a la frecuencia con la que envía los datos el módulo de reconocimiento (0.2 segundos) y el *delay* que existe debido a la gran cantidad de procesamiento que requiere el sistema de reconocimiento, no se podrá alcanzar dicha precisión.

Figura 25 Interfaz gráfica del emulador de mouse.

2.2.3.4 Pruebas

Al final de la fase de implementación se realizaron pruebas de funcionalidad en base a los criterios de aceptación de las historias de usuario. Los casos de prueba utilizados se pueden

revisar en el Anexo I. También se realizaron pruebas de integración para verificar que las aplicaciones en C# reciban exactamente la misma información que envía el módulo de reconocimiento de gestos.

2.2.3.5 Revisión del Sprint (Sprint Review)

En esta revisión se reunió nuevamente parte del equipo de investigación. Se presentó formalmente la arquitectura de software y su implementación. En esta reunión, además de mostrar el diagrama de componentes UML, se utilizó una *marketecture* para describir detalladamente la arquitectura.

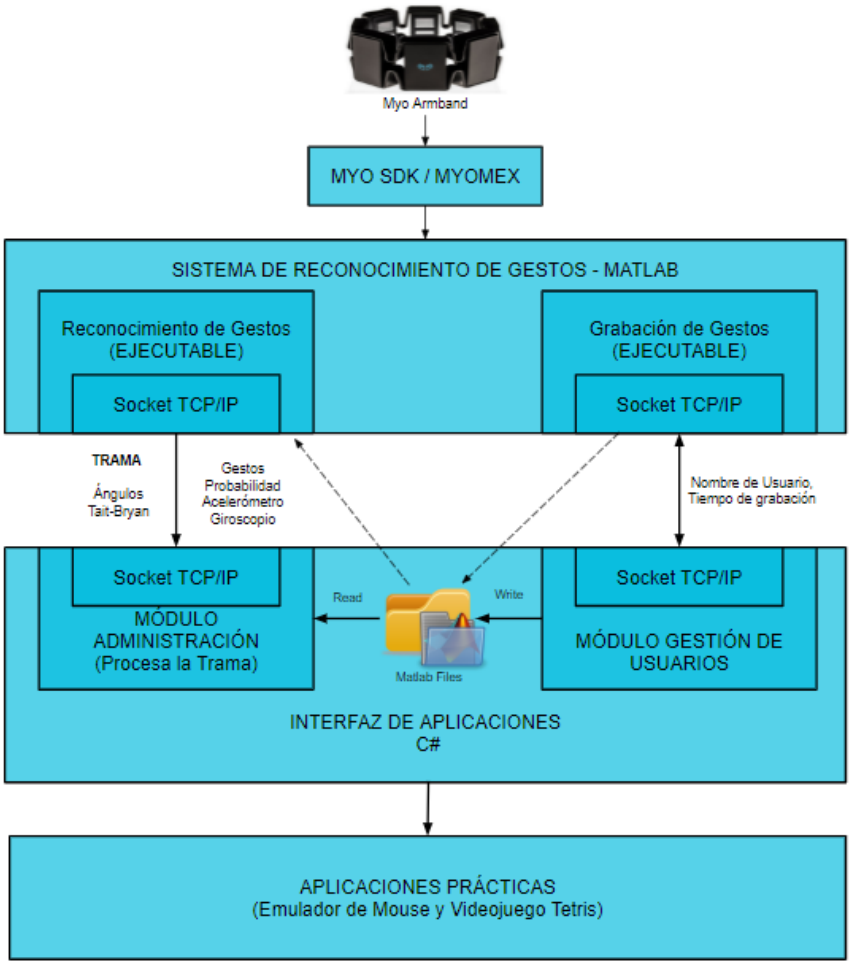

Figura 26 Marketecture de la arquitectura de software

Se conversó respecto a la segunda aplicación práctica. La propuesta realizada por el equipo SCRUM fue un videojuego Tetris debido a las siguientes razones:

- Es un videojuego muy conocido y no hay que enfocarse en explicar su funcionamiento base, sino en la integración con el sistema de reconocimiento.

- Es un videojuego sencillo y no requiere mayor esfuerzo para el desarrollo.
- Los controles del videojuego pueden emularse fácilmente utilizando los gestos reconocibles y la información espacial.

El equipo de investigación accedió al desarrollo de esta aplicación y finalmente se realizó la actualización de la Lista del Producto (*Product Backlog*).

Tabla 11 Lista del producto actualizado al final del Sprint 2.

Orden	**Historia Usuario**	**Descripción**	**Estado**
1	N/A	Compilar el sistema de reconocimiento de gestos del brazo humano para exponer sus funcionalidades.	Terminado
2	N/A	Implementar la arquitectura de Software seleccionada.	Terminado
3	EUS01	Desarrollar un emulador de mouse en base a la arquitectura implementada.	Terminado
4	EUS02	Implementar un módulo de gestión de Usuarios.	Terminado
5	N/A	Proponer y desarrollar una segunda aplicación en base a la arquitectura implementada.	Terminado
6	EUS03	Desarrollar el video juego Tetris en base a la arquitectura implementada.	Creado

2.2.3.6 Retrospectiva del Sprint (Sprint Retrospective)

En esta reunión el equipo de desarrollo discutió los retrasos registrados durante las mejoras del emulador de Mouse, se tuvo que invertir más tiempo de lo planificado para poder completar dicha tarea. El equipo de desarrollo considera que este inconveniente ocurrió por el poco conocimiento que se tenía respecto al uso de los ángulos de Tait-Bryan y el uso de los datos del giroscopio. Para solucionar este problema se acordó solicitar consejo al equipo de investigación, quienes tiene un mayor conocimiento en estos temas.

2.2.4 Sprint 3

2.2.4.1 Planificación del Sprint (Sprint Planning)

En este evento el equipo SCRUM se reunió para elegir los elementos del *Product Backlog* (actualizado) que se realizarían durante el *Sprint* 3. El Objetivo de este *Sprint (Sprint Goal)* fue implementar el video juego Tetris y realizar mejoras en los incrementos anteriores.
El equipo de desarrollo definió las tareas necesarias para alcanzar el objetivo. Para el desarrollo del video juego Tetris se recolectaron Historias de Usuarios. A continuación, se presenta la Lista de Pendientes del *Sprint* 3 (*Sprint Backlog*).

Lista de Pendientes del Sprint (Sprint Backlog)

Tabla 12 Lista de Pendientes del Sprint 3.

Orden	**US**	**Descripción**	**Estimación (Horas)**
	US07	**Desarrollar un video juego Tetris.**	**40**
1		Implementar la funcionalidad original del video juego Tetris.	10
2		Crear la funcionalidad para rotar los bloques en sentido horario.	10
3		Crear la funcionalidad para mover bloques horizontalmente.	5
4		Crear marcador de puntos.	5
5		Controlar la velocidad de caída de los bloques.	5
6		Mostrar el siguiente bloque a jugar.	5
	US08	**Asociar gestos reconocibles a los controles.**	**20**
7		Asociar cada uno de los gestos reconocibles a un control del video Juego Tetris (Iniciar/Reiniciar, Pausar/Reanuda, Rotar un bloque, soltar un bloque, intercambiar bloque).	10
8		Crear una funcionalidad para que la interfaz gráfica muestre la configuración actual de los controles asociados a los gestos.	10
	US09	**Asociar el movimiento del antebrazo al desplazamiento de bloques.**	**20**
9		Asociar los datos espaciales del dispositivo Myo Armband con el desplazamiento horizontal de los bloques.	10

10		Controlar el desplazamiento de los bloques para que no excedan los límites del campo de juego.	10
	US10	**Módulo de Configuración Controles Tetris**	**20**
11		Crear una interfaz gráfica que muestre la configuración actual de controles.	5
12		Crear una función para reestablecer por defecto los controles.	5
13		Crear una función para validar relación uno a uno entre controles y gestos reconocibles.	5
14		Crear una función para editar la asociación entre gestos reconocibles y controles del emulador de mouse.	5

Historias de Usuario

Historia de Usuario		US07
Título	Desarrollar un video juego Tetris	
Descripción	Yo como usuario Quiero un video juego Tetris Para vincular sus controles al sistema de reconocimiento de gestos	
Prioridad: Media		Esfuerzo: 40
Criterios de Aceptación	Se ha implementado la funcionalidad original del video juego Tetris. Los bloques pueden rotar en sentido horario. Los bloques solo pueden moverse horizontalmente. Se muestra un marcador de puntos que aumenta cada vez que se completa una fila. La velocidad de caída de los bloques aumenta respecto al número de puntos obtenidos. El juego muestra el siguiente bloque a jugar. Se puede intercambiar el bloque actual por el siguiente bloque. El juego puede ser controlado utilizando el teclado del computador.	

Historia de Usuario		US08
Título	Asociar gestos reconocibles a los controles.	
Descripción	Yo como usuario	

	quiero utilizar los gestos reconocibles por el sistema de reconocimiento para realizar las acciones del video juego Tetris
Prioridad: Alta	Esfuerzo: 20
Criterios de Aceptación	Cada uno de los gestos reconocibles se encuentra asociado a un control del video Juego Tetris. Al menos un gesto reconocible puede iniciar/reiniciar el Videojuego. Al menos un gesto reconocible puede pausar/reanudar el Videojuego. Al menos un gesto reconocible puede rotar un bloque. Al menos un gesto reconocible puede soltar un bloque. Al menos un gesto reconocible puede intercambiar el bloque actual por el siguiente. La interfaz gráfica muestra la configuración actual de los controles asociados a los gestos.

Historia de Usuario		US09
Título	Asociar el movimiento del antebrazo al desplazamiento de bloques.	
Descripción	Yo como usuario quiero utilizar el movimiento horizontal de mi antebrazo. para desplazar los bloques horizontalmente.	
Prioridad: Media		Esfuerzo: 20
Criterios de Aceptación	El Movimiento horizontal del antebrazo del usuario mueve horizontalmente los bloques de Tetris. Los bloques no exceden los límites del campo de juego.	

Historia de Usuario		US10
Título	Módulo de Configuración Controles Tetris	
Descripción	Yo como usuario Quiero un módulo de configuración Para cambiar la asociación entre gestos reconocibles y controles del Videojuego Tetris.	
Prioridad: Media		Esfuerzo: 20
Criterios de Aceptación	La interfaz gráfica muestra la configuración actual. Se puede asociar los gestos reconocibles a los controles del Video Juego. Se valida que cada control tiene al menos un gesto asociado.	

	La asociación entre controles y gestos asociados es uno a uno. Se puede restablecer la configuración por defecto.

2.2.4.2 Implementación

En esta última iteración se realizó el desarrollo completo del videojuego Tetris, primero se implementó la funcionalidad básica del juego. Posteriormente se realizó la integración con la arquitectura de software. Se asoció los controles del Tetris con los gestos reconocibles y se vinculó el movimiento horizontal del antebrazo del usuario (Ángulo Yaw) con el movimiento horizontal de los bloques del juego.

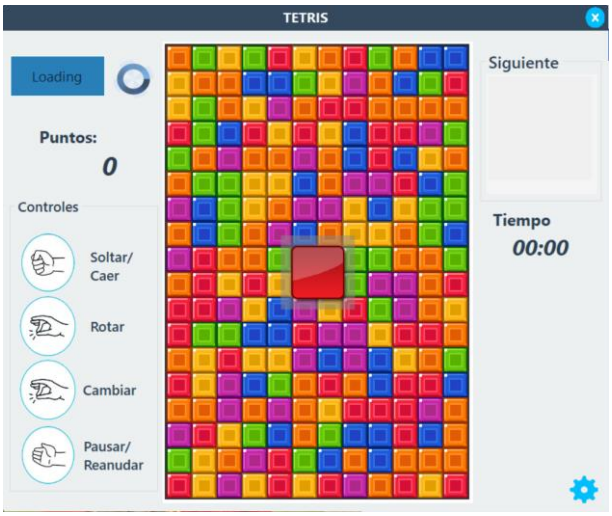

Figura 27 Interfaz gráfica del video juego Tetris

Luego de implementar y unir el videojuego con la arquitectura, se desarrolló el módulo de configuración para poder cambiar la asociación entre los gestos reconocidos con los controles del videojuego. La **Figura 28** muestra la interfaz de configuración, donde se puede definir la asociación entre cada gesto reconocible con los controles del videojuego.

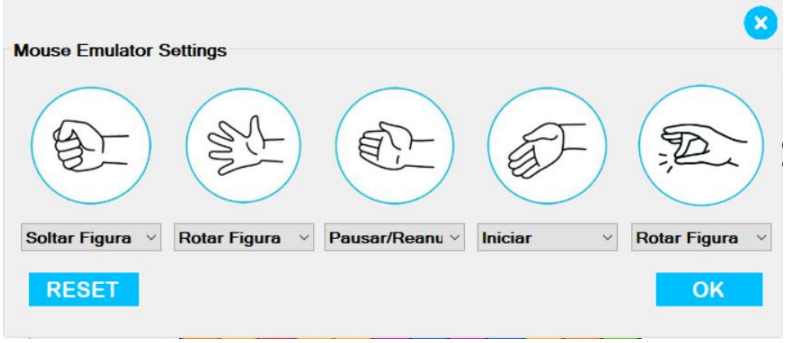

Figura 28 Módulo de configuración del video juego Tetris.

2.2.4.3 Pruebas

Al final de la fase de implementación se realizaron las pruebas funcionales sobre los criterios de aceptación de las historias de usuario planificadas. Se verificó que la aplicación cumplía con todos los requerimientos ofrecidos al equipo de investigación. Se pueden revisar los casos de prueba en el Anexo I.

2.2.4.4 Revisión del Sprint (Sprint Review)

En esta última revisión, se reunió parte del equipo de investigación. Se presentó el incremento obtenido en el *Sprint* 3. Se realizó la última actualización de la Lista del Producto (*Product Backlog*), todos los elementos de la Lista del Producto se marcaron como "terminados". Tomando esto en cuenta, se coordinaron las Pruebas de Usabilidad y junto a los interesados se formularon los casos de prueba para las Pruebas de Aceptación.

Tabla 13 Lista del Producto actualizada al final del Sprint 3

Orden	**Historia Usuario**	**Descripción**	**Estado**
1	N/A	Compilar el sistema de reconocimiento de gestos del brazo humano para exponer sus funcionalidades.	Terminado
2	N/A	Implementar la arquitectura de Software seleccionada.	Terminado
3	EUS01	Desarrollar un emulador de mouse en base a la arquitectura implementada.	Terminado
4	EUS02	Implementar un módulo de gestión de Usuarios.	Terminado
5	N/A	Proponer y desarrollar una segunda aplicación en base a la arquitectura implementada	Terminado
6	EUS03	Desarrollar el video juego Tetris en base a la arquitectura implementada	Terminado

2.3 Pruebas de Software

2.3.1 Pruebas de Facilidad de Uso

Para la evaluación de la facilidad de uso o usabilidad, se utilizó una encuesta de 15 preguntas basada en la escala de usabilidad SUS (System Usability Scale) [35] que sugiere utilizar una escala para la valoración de las preguntas, donde 1 es la peor o mínima puntuación y 5 es la mejor o máxima puntuación.

Además, se añadieron preguntas referentes a la eficiencia, los errores, la satisfacción de usuario, facilidad de aprendizaje y memorabilidad. Esta evaluación será aplicada únicamente para la gestión de usuarios que incluye el módulo de video entrenamiento, el emulador de Mouse y el videojuego Tetris. El documento completo para la evaluación de la facilidad de uso se encuentra disponible en el Anexo II. A continuación, se muestra la encuesta para la evaluación de facilidad de Uso:

Tabla 14 Cuestionario para la evaluación de facilidad de uso

N°	Preguntas					
		1	2	3	4	5
1	¿Visualmente que tan atractivo le pareció el conjunto total de interfaces (Colores, botones, texto, formato, estilo, abreviaturas, etc.)?					
2	¿El conjunto total de interfaces de usuario muestra una integridad conceptual y secuencia coherente?					
3	Tiempos de carga entre pantalla, interfaces gráficas.					
4	Tiempos de carga de las aplicaciones (Video Entrenamiento, Emulador de Mouse, Tetris).					
5	Navegabilidad entre las interfaces gráficas.					
6	Los Mensajes informativos y los video tutoriales que tanto ayudaron a comprender el objetivo del módulo de video entrenamiento.					
7	¿Cuán intuitivo considera que es el módulo de video entrenamiento?					
8	¿Qué tan sencillos de comprender son los mensajes de error o de advertencia?					
9	¿Qué tan complejas considera que son las aplicaciones?					
10	¿Qué tan fácil le resultó aprender a utilizar el Emulador de Mouse?					
11	¿Qué tan preciso considera que es el Emulador de Mouse?					
12	¿Qué tan fácil le resultó aprender a utilizar el videojuego Tetris?					
13	¿Qué tan entretenido le resultó jugar el videojuego Tetris utilizando solo su brazo y los gestos de su mano?					
14	¿Cuál es su nivel de satisfacción luego de utilizar el Emulador de Mouse?					
15	¿Cuál es su nivel de satisfacción respecto al reconocimiento de gestos?					

2.3.2 Pruebas de Aceptación de Usuario

Las pruebas de aceptación también llamadas UAT (*User Acceptance Testing*) [36], se realizan para determinar si el producto de software cumple con los requisitos de

funcionalidad y facilidad de uso. Este tipo de pruebas son diseñadas con los interesados a partir de las historias de usuarios. La proporción de historias de usuario a las pruebas de aceptación suele ser de una a muchas. Las UAT deben ser pruebas manuales y deben ser dirigidas por los interesados. Cuando las pruebas de aceptación concluyen exitosamente el producto de software se considera terminado y entregable [37]. Estas pruebas fueron realizadas al final de la tercera iteración. El documento firmado por el dueño del producto y el equipo SCRUM se encuentra en CD Anexado.

3 RESULTADO Y DISCUSIÓN

En el presente proyecto se propusieron 3 arquitecturas de software, las cuales fueron diseñadas en función de la tecnología disponible, recursos y requerimientos del proyecto de investigación PIJ-16-13 de la Escuela Politécnica Nacional. Las 3 arquitecturas propuestas fueron evaluadas en función de su factibilidad y simplicidad de implementación. Además, se realizó un análisis comparativo exponiendo sus ventajas y desventajas para facilitar la toma de decisiones.

El equipo de investigación seleccionó una de las arquitecturas propuestas, la cual fue completada y mejorada utilizando un enfoque iterativo e incremental. La arquitectura seleccionada fue descrita utilizando el Lenguaje Unificado de Modelado (UML) y es independiente de cualquier lenguaje de programación. Además, se diseñó una versión descriptiva de la arquitectura (*marketecture*) la cual fue empleada durante los *Sprint Review*.

La comunicación entre los componentes de la arquitectura se realiza a través de *sockets TCP/IP* de comunicación. También se definió una trama para el envío y recepción de datos entre el componente de reconocimiento y las aplicaciones. Esta trama es simple y escalable, y recopila los datos espaciales del dispositivo Myo Armband y los datos de gestos del sistema de reconocimiento de gestos.

La arquitectura de software contempla la necesidad de algunos sistemas de reconocimiento de gestos, de recolectar señales EMGs de los usuarios para realizar un entrenamiento del sistema. Esta necesidad fue contemplada en el módulo de gestión de usuarios y se diseñó un diagrama de secuencia UML que detalla la interacción entre el usuario, el componente de grabación de gestos y la interfaz gráfica.

Para demostrar la factibilidad y utilidad de la arquitectura de software se desarrollaron 2 prototipos de aplicaciones de software: un emulador de mouse y un videojuego Tetris, ambas aplicaciones fueron pensadas para aprovechar la funcionalidad de reconocimiento de gestos y los sensores del dispositivo Myo Armband. Se utilizó una metodología de desarrollo ágil y el marco de trabajo *SCRUM* para el desarrollo de las aplicaciones.

El módulo de Gestión de usuarios de la arquitectura de software y las aplicaciones prácticas fueron evaluadas respecto a la facilidad de uso (usabilidad). Estas pruebas fueron realizadas al final de la tercera iteración, se contó con la participación de 20 estudiantes de la Facultad de Ingeniería de Sistemas de la Escuela Politécnica Nacional. Considerando que solo se disponía de un dispositivo Myo Armband las pruebas fueron guiadas individualmente. El tiempo aproximado para el desarrollo de las pruebas por usuario fue de 20 minutos en promedio. 19 de los participantes de estas pruebas no conocían el dispositivo Myo Armband y tampoco conocían como realizar los gestos reconocibles. Las encuestas realizadas se encuentran en el CD anexado.

Como se detalló en la sección anterior, cada pregunta del cuestionario tuvo una puntuación del 1 al 5. A continuación, se muestra una tabla resumen del promedio obtenido por cada pregunta:

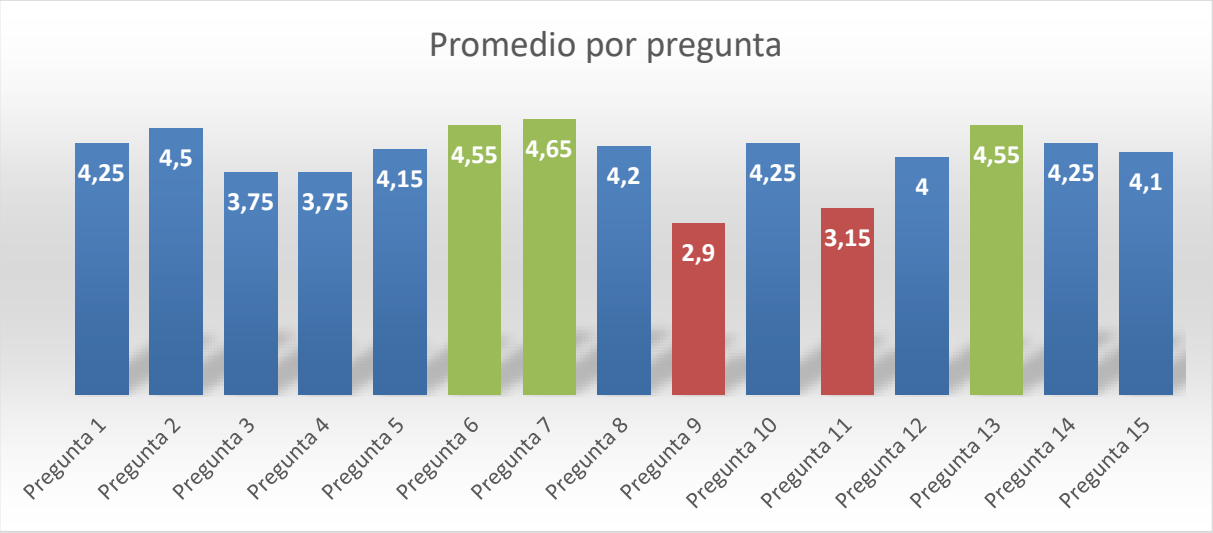

Figura 29 Resumen, promedio por cada pregunta del cuestionario de facilidad de uso.

La pregunta 1 y 2 hacen referencia al apartado visual, la integridad conceptual y secuencia coherente de las interfaces gráficas, el promedio fue bueno con una puntuación de 4.25/5 y 4.5/5 lo cual indica que los usuarios entendieron el objetivo de las aplicaciones y el módulo de video entrenamiento, y les pareció visualmente atractivo.

La pregunta 3 hacen referencia al tiempo de carga entre las interfaces gráficas, la pregunta 4 se refiere al tiempo de carga de los Aplicaciones. Ambas preguntas obtuvieron una puntuación de 3.75/5, la cual es relativamente baja. Esto se debe a que el tiempo de carga de las aplicaciones está sujeto al tiempo de carga del sistema de reconocimiento, el cual es de aproximadamente de 30 a 60 segundos. Este tiempo de carga es heredado del ejecutable de reconocimiento de gestos y varía de acuerdo al equipo utilizado. Se puede observar que este tiempo de carga excesivo de las aplicaciones, afectó la percepción general de los usuarios respecto a todo el producto.

La navegabilidad entre las interfaces gráficas obtuvo una puntuación de 4.15/5, lo cual indica que los usuarios son capaces de moverse a través de las diferentes pantallas de la aplicación fácilmente. Los íconos utilizados y los botones son descriptivos. Los mensajes de error son fácilmente comprensibles, en esta sección se obtuvo una puntuación de 4.2/5

La pregunta 6 evalúa los mensajes informativos y videos tutoriales del módulo de video entrenamiento, en esta pregunta se obtuvo una puntuación de 4.55/5, esto indica que los usuarios pudieron realizar correctamente los gestos tan solo con ver los videos tutoriales y seguir las instrucciones de la interfaz gráfica. La pregunta 7 obtuvo una calificación de 4.65/5, la más alta del cuestionario, esto indica que los usuarios consideran que el módulo de video entrenamiento es intuitivo y no requiere de la ayuda de un experto para ejecutarlo correctamente.

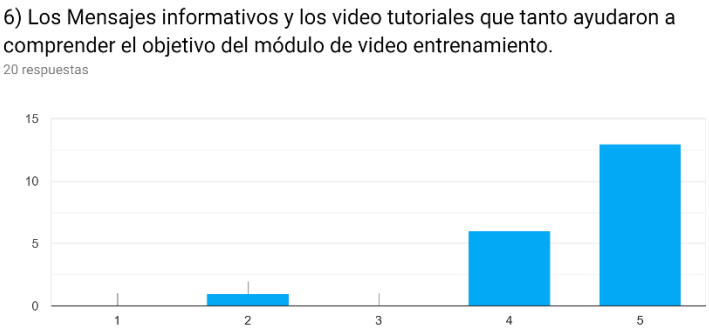

Figura 30 Gráfico de barras Pregunta 6.

Respecto al emulador de mouse se realizaron 2 preguntas. La primera respecto a la facilidad para aprender a utilizar el emulador de mouse, donde se obtuvo una puntuación de 4.25/5. A pesar de haber obtenido una puntuación aceptable, se pudo observar que muchos usuarios no entendieron la premisa de utilizar solo el antebrazo para desplazar el cursor, lo cual complicaba el uso de esta aplicación. La segunda pregunta fue dirigida a la precisión del emulador, donde se obtuvo una puntuación de 3.15/5, este apartado obtuvo la puntuación más baja respecto a las otras aplicaciones. Esto se debe a que a el sistema de reconocimiento de gestos utilizado, tiene un porcentaje de precisión del 86%, además, los usuarios no están acostumbrados a realizar los gestos reconocibles, lo cual disminuye más la precisión de reconocimiento de gestos y adicionalmente la dificultad que tenían algunos usuarios para desplazar el cursor. Sin embargo, esto no influyó negativamente a la pregunta 14 que se refiere al nivel de satisfacción al utilizar el emulador donde se obtuvo una puntuación de 4.25/5.

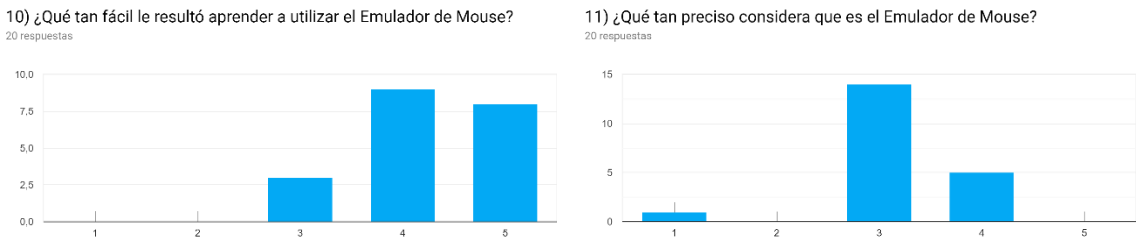

Figura 31 Gráfico de barras Pregunta 10 y 11.

Respecto al video juego Tetris, la facilidad con la cual los usuarios aprendieron a controlar el juego con su antebrazo y los gestos de su mano obtuvo una puntuación de 4/5. Pero al

tratarse de un juego, se pudo observar mayor atención por parte de los usuarios en tratar de aprender a utilizar esta aplicación, esto influyó en el nivel de satisfacción luego de probar esta aplicación, en esta pregunta se obtuvo una puntuación de 4.55/5, la segunda más alta del cuestionario.

Figura 32 Gráfico de barras Pregunta 12 y 13.

La pregunta 9 obtuvo 2.9/5 la calificación más baja, esta pregunta hace referencia al nivel de complejidad de las aplicaciones desde la perspectiva del usuario. Se pudo observar durante la sesión de pruebas que algunos usuarios dominaron fácilmente el emulador de mouse y el videojuego Tetris, mientras otros tuvieron dificultades. El porqué de esta diferencia puede responder a diferentes factores como: la correcta o incorrecta ejecución de los gestos durante el video entrenamiento, la comprensión de las instrucciones iniciales, el nivel de precisión del sistema de reconocimiento, no reconocer la diferencia entre brazo, antebrazo y mano, etc.

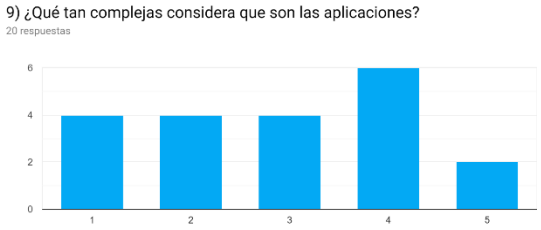

Figura 33 Gráfico de barras Pregunta 9.

Las Pruebas de Aceptación fueron ejecutadas de acuerdo a lo planificado, se pudo validar que: la Arquitectura de Software implementada, el emulador de mouse y el videojuego Tetris, cumplen con todos los requerimientos funcionales y de facilidad de uso que esperaba el equipo de investigación PIJ-16-13.

4 CONCLUSIONES Y RECOMENDACIONES

- Se propusieron 3 tipos de arquitecturas de software creadas a partir de las necesidades y restricciones del equipo de Investigación PIJ-16-13. Las arquitecturas incluyen varias herramientas tecnológicas de las cuales dispone el equipo de investigación. La creación de estas arquitecturas ofrece una perspectiva de como poder exponer la funcionalidad de reconocimiento de gestos de los modelos que ellos investigan.
- Las propuestas arquitectónicas fueron analizadas, comparadas y expuestas al equipo de investigación, quienes eligieron una. La arquitectura seleccionada básicamente utiliza el compilador de aplicaciones de Matlab para transformar el sistema de reconocimiento de gestos en un archivo ejecutable y *sockets TCP/IP* para establecer un canal de comunicación entre los componentes.
- El enfoque iterativo e incremental empleado para completar y mejorar la arquitectura de software seleccionada fue el correcto. Debido a que en cada iteración el equipo SCRUM obtenía un mayor conocimiento sobre el dominio del problema y retroalimentación del equipo de investigación, obteniendo un diseño que responde a las necesidades de los interesados.
- El uso de *Sockets* como canal de comunicación entre los componentes de la arquitectura fue correcto. Es una abstracción ampliamente conocida, de fácil implementación y con buen rendimiento al trabajar en un entorno local.
- La trama para el envío y recepción de datos es simple y escalable. Recopila tanto la información espacial disponible del dispositivo Myo Armband, así como la información de gestos producida por el Sistema de reconocimiento de gestos. Aunque no todos los datos de la trama fueron utilizados por los prototipos de software, estos están disponibles para ser utilizado por aplicaciones futuras.
- Durante el proceso de implementación de la arquitectura, se desarrolló una interfaz gráfica que permitió visualizar en tiempo real los datos enviados por el sistema de reconocimiento de gestos. Esta interfaz facilitó las pruebas de integración, depuración de errores y desarrollo de las aplicaciones prácticas.
- El uso de diagramas de componentes y diagramas de comunicación UML permite describir la arquitectura de software de forma detallada, utilizando una notación vigente y ampliamente conocida. Esto permitirá al equipo de investigación poder implementar la arquitectura en cualquier otro lenguaje de programación si así lo

requieren en el futuro.

- La arquitectura contempla la necesidad de recolectar señales EMGs de los usuarios para algunos sistemas de reconocimiento que requieren entrenamiento. La implementación de este módulo requiere de mucha sincronización entre el ejecutable de Grabación de Gestos y la interfaz gráfica de usuario.
- El módulo de Grabación de Gestos fue sometido a pruebas de usabilidad para asegurar que sea lo suficientemente intuitivo como para ser usado por cualquier persona sin la ayuda de miembros del equipo de investigación. Se pudo verificar que los datos EMGs obtenidos utilizando este módulo permitieron el correcto funcionamiento del sistema de reconocimiento de gestos.
- Los prototipos de software fueron desarrollados en base a la arquitectura de software planteada, ambas aplicaciones cumplen con todos los requisitos funcionales especificados por el equipo de investigación. Esto valida la utilidad y factibilidad de la arquitectura propuesta.
- En la primera iteración, cuando se desarrolló el prototipo del emulador de mouse, se registraron retrasos debido a que la arquitectura aún no estaba completa. En la tercera iteración, cuando la arquitectura ya estaba depurada e implementada, la integración del Tetris fue mucho más rápida de lo planificado, esto también avala la utilidad de la arquitectura de software propuesta.
- Respecto al rendimiento de las aplicaciones, se puede concluir que a pesar de que el sistema de reconocimiento de gestos es una caja negra para este proyecto, sí es un factor determinante para el rendimiento de las aplicaciones y la percepción de los usuarios.
- Las pruebas de usabilidad demostraron que las aplicaciones dependen en gran medida de la calidad de gestos recolectados y la precisión de reconocimiento del sistema. Se puede concluir que las aplicaciones de software que utilizan este tipo de sistemas de reconocimientos y el dispositivo Myo Armband no son muy fáciles de aprender a utilizar, dependen en gran medida de la familiaridad que tienen los usuarios tanto con los gestos reconocibles como con el dispositivo Myo Armband. Sin embargo, las pruebas de usabilidad demuestran que la satisfacción de los usuarios luego de utilizar los prototipos de software es muy buena.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] M. E. Benalcázar *et al.*, “Real-Time Hand Gesture Recognition Using the Myo Armband and Muscle Activity Detection,” *2017 IEEE Second Ecuador Tech. Chapters Meet.*, pp. 1–6, 2017.

[2] “Myo Gesture Control Armband | Wearable Technology by Thalmic Labs.” [Online]. Available: https://www.myo.com/. [Accessed: 23-May-2018].

[3] H. A. Romo, J. C. Realpe, P. E. Jojoa, and U. Cauca, “Surface EMG Signals Analysis and Its Applications in Hand Prosthesis Control,” *Av. en Sist. e Informática*, vol. 4, no. 1, pp. 127–136, 2007.

[4] A. Pandit, D. Dand, S. Mehta, and S. Sabesan, “A simple wearable hand gesture recognition device using iMEMS,” *SoCPaR 2009 - Soft Comput. Pattern Recognit.*, pp. 592–597, 2009.

[5] Y.-F. Huang, Y. Hua-Jui, and T. Tan-Hsu, “A STUDY OF HAND GESTURE RECOGNITION WITH WIRELESS CHANNEL MODELING BY USING WEREABLE DEVICES,” *Int. Conf. Mach. Learn. Cybern.*, 2015.

[6] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, Second. Addison Wesley, 2003.

[7] I. Gorton, *Essential software architecture*, First. Berlin: Springer, 2006.

[8] D. Garlan and M. Shaw, “An Introduction to Software Architecture An Introduction to Software Architecture,” no. January 1993, 1994.

[9] Albin.S.T, *The Art of Software Architecture : Design Methods and Techniques*. John Wiley & Sons, 2003.

[10] D. B. Makofske, M. J. Donahoo, and K. L. Calvert, *TCP / IP Sockets in C : Practical Guide for Programmers*. .

[11] IEEE-SA Standards Board, “IEEE Recommended Practice for Architectural Description of Software-Intensive Systems,” *IEEE Std*, vol. 1471–2000, pp. 1–23, 2000.

[12] M. Fowler, “UML Distilled: A Brief Guide to the Standard Object Modeling Language,” *Pearson Paravia Bruno Mondad*, p. 175, 2004.

[13] “What is UML | Unified Modeling Language.” [Online]. Available: http://www.uml.org/what-is-uml.htm. [Accessed: 23-May-2018].

[14] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, vol. 3. 1998.

[15] S. W. Ambler, *The elements of UML 2.0 style*. 2005.

[16] T. Hewett, C. Elizabeth, B. Anne, and P. Jeniffer, “Teaching and learning human-computer interaction | ACM Interactions.” [Online]. Available: http://interactions.acm.org/archive/view/march-april-2013/teaching-and-learning-human-computer-interaction. [Accessed: 22-Sep-2018].

[17] C. Ghaoui, *Encyclopedia Of Human Computer Interaction*. 2006.

[18] G. Kamen and D. A. Gabriel, *Essential of Electromyography*. 2010.

[19] G. Madhavan, *Electromyography: Physiology, Engineering and Non-Invasive Applications*, vol. 33, no. 11. 2005.

[20] M. Whitty, *Robotics, Vision and Control. Fundamental Algorithms in MATLAB*, vol. 39, no. 6. 2012.

[21] “Connect | Take Control with Myo Mouse and Keyboard.” [Online]. Available: https://www.myo.com/connect. [Accessed: 23-May-2018].

[22] ThalmicLabs, “Myo Getting Started.” [Online]. Available: https://developer.thalmic.com/docs/api_reference/platform/getting-started.html.

[23] ThalmicLabs, “Myo SDK 0.9.0: The Myo SDK.” [Online]. Available: https://developer.thalmic.com/docs/api_reference/platform/the-sdk.html. [Accessed: 25-Sep-2018].

[24] “What is MATLAB? - MATLAB & Simulink.” [Online]. Available: https://la.mathworks.com/discovery/what-is-matlab.html. [Accessed: 21-Sep-2018].

[25] “MATLAB Compiler Documentation - MathWorks America Latina.” [Online]. Available: https://la.mathworks.com/help/compiler/index.html. [Accessed: 23-May-2018].

[26] Matlab, “Compiler SDK Rumtime.” [Online]. Available: https://la.mathworks.com/help/compiler_sdk/index.html. [Accessed: 21-Sep-2018].

[27] “Myo SDK MATLAB MEX Wrapper - File Exchange - MATLAB Central.” [Online]. Available: https://la.mathworks.com/matlabcentral/fileexchange/55817-myo-sdk-matlab-mex-wrapper. [Accessed: 21-Sep-2018].

[28] “Windows Presentation Foundation | WPF y .NET | Visual Studio.” [Online]. Available: https://www.visualstudio.com/es/vs/features/wpf/. [Accessed: 23-May-2018].

[29] “Windows Forms | Microsoft Docs.” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/winforms/. [Accessed: 23-May-2018].

[30] “Myo Sharp| Github.” [Online]. Available: https://github.com/tayfuzun/MyoSharp. [Accessed: 21-Sep-2018].

[31] “Home - Bunifu Framework | Empowering software developers craft great user experiences in less time. Productivity tools for C# & VB.NET UX/UI design.” [Online]. Available: https://bunifuframework.com/. [Accessed: 21-Sep-2018].

[32] J. Sutherland and K. Schwaber, *Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, and Leave Competitors in the Dust*. 2012.

[33] K. Schwaber and J. Sutherland, “The Scrum Guide,” *Scrum.Org and ScrumInc*, no. July, p. 19, 2017.

[34] Microsoft Docs, “mouse_event function | Microsoft Docs.” [Online]. Available: https://docs.microsoft.com/en-us/windows/desktop/api/winuser/nf-winuser-mouse_event. [Accessed: 25-Sep-2018].

[35] P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I. L. McClelland, “SUS: a ‘quick and dirty’ usability scale,” *Usability Eval. Ind.*, pp. 189–194, 1996.

[36] G. J. Myers, T. Badgett, T. M. Thomas, and C. Sandler, *Second Edition The Art of Software Testing*. 2008.

[37] N. Svensson and J. Gradén, “Software testing in extreme programming,” 2009.

6 ANEXOS

6.1 Anexo I

Casos de Prueba

ID Caso Prueba	CP-01	**Prioridad**	Media	**Autor del CP**	Fabrizio Ramírez
US Asociada	US01	**Ambiente**	Test	**Ejecutor de CP**	
Objetivo	Verificar que el emulador de mouse asocia el movimiento del antebrazo del usuario al desplazamiento del curso.				
Precondiciones	El usuario debe tener correctamente puesto el dispositivo Myo Armband en su brazo derecho. El programa Myo Connect se encuentra iniciado. El dispositivo Myo Armband se encuentra conectado al programa Myo Connect vía Bluetooth. El modo Debug de la aplicación se encuentra habilitado.				

#	**Acciones**	**Resultado esperado**	**Resultado**
1	Hacer clic en el botón Emulador de mouse de la interfaz inicial.	Se muestra una pantalla de carga en la esquina inferior derecha y se observa que el ejecutable de grabación de gestos se inició.	
2	Esperar a que se cargue el emulador de mouse.	La pantalla de carga ha desaparecido se muestra una notificación indicando que el emulador de mouse se encuentra activo.	
3	Con el dispositivo Mouse ubicar el cursor en el centro de la pantalla. Soltar el mouse.	El mouse se encuentra en el centro de la pantalla.	
4	Mover el antebrazo lentamente hacia la izquierda.	El cursor se desplaza hacia la izquierda.	
5	Mover el antebrazo lentamente hacia la derecha.	El cursor se desplaza hacia la derecha.	
6	Mover el antebrazo rápidamente hacia la izquierda.	El cursor se desplaza hacia la el borde izquierdo de la pantalla.	
7	Mover el antebrazo lentamente hacia la derecha.	El cursor se desplaza hacia la el borde derecho de la pantalla.	
8	Con el dispositivo Mouse ubicar el cursor en el centro de la pantalla. Soltar el mouse.	El mouse se encuentra en el centro de la pantalla.	
9	Mover el antebrazo lentamente hacia arriba.	El cursor se desplaza hacia arriba.	
10	Mover el antebrazo lentamente hacia abajo.	El cursor se desplaza hacia abajo.	
11	Mover el antebrazo rápidamente hacia arriba.	El cursor se desplaza hacia la el borde superior de la pantalla.	
12	Mover el antebrazo lentamente hacia la abajo.	El cursor se desplaza hacia la el borde inferior de la pantalla.	
13	Mover el antebrazo libremente.	El cursor sigue la trayectoria que dibuja la mano del usuario.	

Resultado		**Evidencia**	

ID Caso Prueba	CP-02	**Prioridad**	Media	**Autor del CP**	Fabrizio Ramírez
US Asociada	US02	**Ambiente**	Test	**Ejecutor de CP**	
Objetivo	Verificar que el emulador de mouse asocia los gestos reconocibles por el sistema de reconocimiento a acciones de mouse.				
Precondiciones	El usuario debe tener correctamente puesto el dispositivo MYO Armband en su brazo derecho. El programa Myo Connect se encuentra activo. El dispositivo Myo Armband se encuentra conectado al programa Myo Connect vía Bluetooth. El modo Debug se encuentra habilitado.				

#	**Acciones**	**Resultado esperado**	**Resultado**
1	Hacer clic en el botón Emulador de Mouse de la interfaz inicial.	Se muestra una pantalla de carga en la esquina inferior derecha y se observa que el ejecutable de grabación de gestos se inició.	
2	Esperar a que se cargue el emulador de mouse.	La pantalla de carga ha desaparecido se muestra una notificación indicando que el emulador de mouse se encuentra activo.	
3	En el escritorio de Windows ubicar el cursor sobre un icono. Realizar el gesto Fist (puño).	El sistema de reconocimiento de gestos detecta el gesto Fist y se realiza la acción de mouse clic izquierdo sobre el icono.	
4	En el escritorio de Windows ubicar el cursor sobre un icono. Realizar el gesto Open (mano estirada)	El sistema de reconocimiento de gestos detecta el gesto Open y se realiza la acción de mouse clic derecho sobre el icono.	
5	En el escritorio de Windows ubicar el cursor sobre un icono. Realizar el gesto Wave In (mano junta hacia adentro)	El sistema de reconocimiento de gestos detecta el gesto Wave In y se realiza la acción “Right Mouse Down” sobre el icono.	
6	Mover el cursor a otra posición del escritorio utilizando el antebrazo.	El cursor se desplaza siguiendo la trayectoria de la mano y arrastra el icono.	
7	Realizar el gesto Wave Out (mano junta hacia afuera)	El sistema de reconocimiento de gestos detecta el gesto Wave Out y se realiza la acción “Right Mouse Up” sobre el icono.	
8	Realizar el gesto Double Tap (Doble chasquido entre el dedo medio y el pulgar) sobre un icono.	El sistema de reconocimiento de gestos detecta el gesto Double Tap y se realiza la acción “Doble clic derecho” sobre el icono.	

Resultado		**Evidencia**	

ID Caso Prueba	CP-03	**Prioridad**	Alta	**Autor del CP**	Fabrizio Ramírez
US Asociada	US03	**Ambiente**	Test	**Ejecutor de CP**	
Objetivo	Probar la creación de un nuevo usuario.				
Precondiciones	Ya existe un usuario creado				

#	**Acciones**	**Resultado esperado**	**Resultado**
1	En la interfaz de inicio dar clic en el ícono de Usuario.	Se muestra una interfaz con los datos del usuario actual.	
2	Dar clic en el botón “Nuevo Usuario”	Se muestra una interfaz que solicita ingresar el nombre del nuevo usuario	
3	Sin haber ingresado un nombre hacer clic en el botón “Continuar”.	Se muestra el mensaje de advertencia: “Ingrese un nombre”	
4	Ingresar el nombre del usuario Actual y hacer clic en el botón “Continuar”.	Se muestra el mensaje de advertencia: “El usuario ingresado ya existe”	
5	En el Textbox digitar un carácter especial.	Se muestra el mensaje de advertencia: “Ingrese solo letras”	
6	En el Textbox digitar un número.	Se muestra el mensaje de advertencia: “Ingrese solo letras”	
7	Ingresar un nombre que no exista y hacer clic en el botón “Continuar”.	En la carpeta “Users” se crea la estructura de directorios predefinida para almacenar los datos del usuario. Se inicia el módulo de video entrenamiento.	

Resultado		**Evidencia**	

ID Caso Prueba	CP-04	**Prioridad**	Alta	**Autor del CP**	Fabrizio Ramírez
US Asociada	US04	**Ambiente**	Test	**Ejecutor de CP**	
Objetivo	Probar el correcto funcionamiento del módulo de video entrenamiento.				
Precondiciones	El usuario debe tener correctamente puesto el dispositivo MYO Armband en su brazo derecho. El dispositivo Myo Armband se encuentra conectado al programa Myo Connect vía Bluetooth. Este caso de prueba se ejecuta inmediatamente después del CP-03 El usuario tiene su brazo derecho pegado a su torso, en una posición relajada apuntando hacia adelante (puede estar sentado).				

#	**Acciones**	**Resultado esperado**	**Resultado**
1	Leer y seguir las instrucciones iniciales del módulo de video entrenamiento.	El módulo de video entrenamiento presenta texto, imágenes y videos demostrativos. Se muestra la interfaz de video entrenamiento.	
2	Presionar la tecla “Space Bar”.	Se muestra un mensaje informativo indicando que se realizará la grabación del gesto “Wave In”. Se muestra un video de ejemplo de cómo realizar el gesto.	
3	Presionar la tecla “Space Bar”.	El Label superior indica: “Vamos a realizar la grabación del gesto Wave In”, el texto botón inferior cambia por “Empezar”	
4	Presionar la tecla “Space Bar”. Cuando la barra de carga	El sistema graba el gesto. Se muestra una ventana de confirmación	

	se ponga de color naranja imitar el gesto que se muestra en el video.	preguntando si realizó correctamente el gesto o desea volver a intentarlo.	
5	Presionar la tecla “N”	El Label superior indica: “Vamos a realizar la grabación del gesto Wave In”, el texto botón inferior cambia por “Empezar”	
6	Presionar la tecla “Space Bar”. Cuando la barra de carga se ponga de color naranja imitar el gesto que se muestra en el video.	El sistema graba el gesto. Se muestra una ventana de confirmación preguntando si realizó correctamente el gesto o desea volver a intentarlo.	
7	Presionar la tecla “S”	El Label superior indica: “Vamos a realizar la grabación del gesto Wave In”, el texto botón inferior cambia por “Empezar”	
8	Realizar la grabación del gesto correctamente 4 veces más.	Se muestra un mensaje informativo indicando que se finalizó con la grabación del gesto “Wave In”.	
9	Presiona la tecla “Space Bar”.	Junto a la etiqueta Wave In se muestra un icono verde.	
10	Realizar la grabación del resto de gestos (Wave Out, Fist, Open, Double Tap, Rest).	Se muestra un mensaje indicando que se realizó la grabación de todos los gestos.	
11	Dar clic en el botón OK	Se muestra una ventana de carga. Luego se muestra un mensaje indicando que se realizó la creación de un usuario correctamente.	
12	Dar clic en el botón OK	Se regresa a la interfaz de inicio. El nombre de usuario ha cambiado por el nombre del nuevo usuario	
Resultado		**Evidencia**	

ID Caso Prueba	CP-05	**Prioridad**	Baja	**Autor del CP**	Fabrizio Ramírez
US Asociada	US05	**Ambiente**	Test	**Ejecutor de CP**	
Objetivo	Probar el correcto funcionamiento del módulo de configuración de ejecutables.				
Precondiciones	El usuario debe tener correctamente puesto el dispositivo MYO Armband en su brazo derecho. El dispositivo Myo Armband se encuentra conectado al programa Myo Connect vía Bluetooth.				

#	**Acciones**	**Resultado esperado**	**Resultado**
1	En la interfaz de inicio dar clic en el icono de “Configuración”	Se abre la interfaz de configuración	
2	Modificar el tiempo de grabación y el número de iteraciones y presionar el botón Guardar.	La configuración se graba correctamente	
3	Ejecutar el módulo de grabación de gestos.	El tiempo de grabación y el número de iteraciones corresponde a los valores establecidos en el módulo de configuración	
4	En el módulo de configuración desactivar el modo Debug.	Cuando se ejecuta el módulo de grabación o el ejecutable de reconocimiento no se muestra la consola de Windows.	
5	En el módulo de configuración activar el modo	Cuando se ejecuta el módulo de grabación o el ejecutable de	

	Debug.	reconocimiento se muestra la consola de Windows.	
6	Presionar el botón “Reset”	La interfaz muestra la configuración inicial	
Resultado		**Evidencia**	

ID Caso Prueba	CP-06	**Prioridad**	Baja	**Autor del CP**	Fabrizio Ramírez
US Asociada	US06	**Ambiente**	Test	**Ejecutor de CP**	
Objetivo	Probar el correcto funcionamiento del módulo configuración del emulador de mouse.				
Precondiciones	El usuario debe tener correctamente puesto el dispositivo MYO Armband en su brazo derecho. El dispositivo Myo Armband se encuentra conectado al programa Myo Connect vía Bluetooth.				

#	**Acciones**	**Resultado esperado**	**Resultado**
1	En la interfaz de inicio dar clic en el icono de “Configuración”.	Se abre la interfaz de configuración	
2	Modificar la asociación entre gestos reconocibles y eventos de mouse. Dar clic en Grabar.	La configuración se graba correctamente	
3	Ejecutar el emulador de gestos.	El emulador de mouse se inicia correctamente y utiliza la nueva configuración.	
4	Abrir la interfaz de configuración. Presionar el botón “Reset”.	La interfaz muestra la configuración inicial del emulador de mouse.	
Resultado		**Evidencia**	

ID Caso Prueba	CP-07	**Prioridad**	Media	**Autor del CP**	Fabrizio Ramírez
US Asociada	US07	**Ambiente**	Test	**Ejecutor de CP**	
Objetivo	Probar el correcto funcionamiento del Video juego Tetris				
Precondiciones	El usuario debe tener correctamente puesto el dispositivo MYO Armband en su brazo derecho. El dispositivo Myo Armband se encuentra conectado al programa Myo Connect vía Bluetooth.				

#	**Acciones**	**Resultado esperado**	**Resultado**
1	En la interfaz de inicio dar clic en el botón “Tetris”.	Se abre la interfaz del video juego Tetris.	
2	Hacer clic en el botón “Start”	El videojuego inicia, se muestra un bloque en el centro superior de la pantalla y empieza descender.	
3	Presionar la tecla direccional Arriba	El bloque gira en sentido horario.	
4	Presionar la tecla direccional Izquierda	El bloque se desplaza hacia la izquierda	
5	Presionar la tecla direccional Derecha	El bloque se desplaza hacia la derecha	
6	Presionar la tecla “Space Bar”	El bloque cae a la parte inferior del campo de juego. Se muestra un nuevo bloque en la parte superior	
7	Jugar hasta formar una línea horizontal con los bloques.	La línea horizontal es eliminada el contador de puntos aumenta.	

8	Presionar la tecla “Shift”	El bloque actual se intercambia por el siguiente bloque.	
9	Presionar la tecla “Enter”	El juego es pausado.	
10	Presionar la tecla “Enter”	El juego se reanuda.	
Resultado		**Evidencia**	

ID Caso Prueba	CP-08	**Prioridad**	Media	**Autor del CP**	Fabrizio Ramírez
US Asociada	US08	**Ambiente**	Test	**Ejecutor de CP**	
Objetivo	Verificar que el juego Tetris asocia los gestos reconocibles a los controles del videojuego				
Precondiciones	El usuario debe tener correctamente puesto el dispositivo MYO Armband en su brazo derecho. El dispositivo Myo Armband se encuentra conectado al programa Myo Connect vía Bluetooth.				

#	Acciones	Resultado esperado	Resultado
1	En la interfaz de inicio dar clic en el botón “Tetris”.	Se abre la interfaz del video juego Tetris. Se inicia el ejecutable de reconocimiento de gestos. Se muestra una pantalla de carga.	
2	Esperar hasta que la pantalla de carga desaparezca.	El botón Start se habilita. Se verifica que exista un resumen de los controles del videojuego respecto a los gestos reconocibles.	
3	Realizar el gesto “Wave In”	El videojuego inicia, se muestra un bloque en el centro superior de la pantalla y empieza descender.	
4	Realizar el gesto “Open”	El bloque gira en sentido horario.	
5	Realizar el gesto “Fist”	El bloque cae a la parte inferior del campo de juego. Se muestra un nuevo bloque en la parte superior	
6	Realizar el gesto “Double Tap”	El bloque actual se intercambia por el siguiente bloque.	
7	Realizar el gesto “Wave Out”	El juego es pausado.	
8	Realizar el gesto “Wave Out”	El juego se reanuda.	
Resultado		**Evidencia**	

ID Caso Prueba	CP-09	**Prioridad**	Media	**Autor del CP**	Fabrizio Ramírez
US Asociada	US09	**Ambiente**	Test	**Ejecutor de CP**	
Objetivo	Verificar que el juego Tetris asocia el movimiento del antebrazo del usuario al desplazamiento de los bloques.				
Precondiciones	El usuario debe tener correctamente puesto el dispositivo MYO Armband en su brazo derecho. El dispositivo Myo Armband se encuentra conectado al programa Myo Connect vía Bluetooth.				

#	Acciones	Resultado esperado	Resultado
1	En la interfaz de administración dar clic en el botón	Se abre la interfaz del video juego Tetris. Se inicia el ejecutable	

	“Tetris”.	de reconocimiento de gestos. Se muestra una pantalla de carga	
2	Esperar hasta que la pantalla de carga desaparezca.	El botón Start se habilita.	
3	Realizar el gesto “Wave In”	El videojuego inicia, se muestra un bloque en el centro superior de la pantalla y empieza descender.	
4	Mover el antebrazo lentamente hacia la izquierda	El bloque se desplaza hacia la izquierda	
5	Mover el antebrazo lentamente hacia la derecha	El bloque se desplaza hacia la derecha	
Resultado		**Evidencia**	

ID Caso Prueba	CP-10	**Prioridad**	Media	**Autor del CP**	Fabrizio Ramírez
US Asociada	US10	**Ambiente**	Test	**Ejecutor de CP**	
Objetivo	Verificar la funcionalidad del módulo de configuración del Video Juego Tetris.				
Precondiciones	El usuario debe tener correctamente puesto el dispositivo MYO Armband en su brazo derecho. El dispositivo Myo Armband se encuentra conectado al programa Myo Connect vía Bluetooth.				

#	**Acciones**	**Resultado esperado**	**Resultado**
1	En la interfaz del video juego Tetris dar clic en el icono de “Configuración”.	Se abre la interfaz de configuración	
2	Modificar la asociación entre gestos reconocibles y Controles del Videojuego Tetris. Dar clic en Grabar.	La configuración se graba correctamente. Se muestran los nuevos controles para el videojuego.	
3	Probar/jugar el videojuego Tetris.	Los controles se asocian correctamente a la nueva configuración	
Resultado		**Evidencia**	

6.2 Anexo II

Evaluación Facilidad de Uso – Usabilidad

PROYECTO: DISEÑO DE UNA ARQUITECTURA DE SOFTWARE Y APLICACIONES PRÁCTICAS PARA EXPLOTAR LAS CAPACIDADES DE UN SISTEMA DE RECONOCIMIENTO DE GESTOS DEL BRAZO HUMANO

PRODUCTOS: MÓDULO DE VIDEO ENTRENAMIENTO – EMULADOR DE MOUSE – TETRIS

LUEGO DE UTILIZAR LOS PRODUCTOS LISTADOS ANTERIORMENTE RESPONDA LAS SIGUIENTES PREGUNTAS, SIENDO:

1- LA PEOR RESPUESTA O MENOR

5- LA MEJOR RESPUESTA O MAYOR

N°	Preguntas					
		1	2	3	4	5
1	Visualmente que tan atractivo le pareció el conjunto total de interfaces (Colores, botones, texto, formato, estilo, abreviaturas, etc.)					
2	¿El conjunto total de interfaces de usuario muestra una integridad conceptual y secuencia coherente?					
3	Tiempos de carga entre pantalla, interfaces gráficas.					
4	Tiempos de carga de las aplicaciones (Video Entrenamiento, Emulador de Mouse, Tetris).					
5	Navegabilidad entre las interfaces gráficas.					
6	Los Mensajes informativos y los video tutoriales que tanto ayudaron a comprender el objetivo del módulo de video entrenamiento.					
7	¿Cuán intuitivo considera que es el módulo de video entrenamiento?					
8	¿Qué tan sencillos de comprender son los mensajes de error o de advertencia?					
9	¿Qué tan complejas considera que son las aplicaciones?					
10	¿Qué tan fácil le resultó aprender a utilizar el Emulador de Mouse?					
11	¿Qué tan preciso considera que es el Emulador de Mouse?					
12	¿Qué tan fácil le resultó aprender a utilizar el videojuego Tetris?					
13	¿Qué tan entretenido le resultó jugar el videojuego Tetris utilizando solo su brazo y los gestos de su mano?					
14	¿Cuál es su nivel de satisfacción luego de utilizar el Emulador de Mouse?					
15	¿Cuál es su nivel de satisfacción respecto al reconocimiento de gestos?					

Comentarios / Sugerencias:

_____

_____

_____

Nombre: _____

Firma: _____