UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
Departamento de Informática

Desarrollo de un SDK para dar soporte a OpenGlove en dispositivos móviles

Israel Gedeón Elías Matínez Montenegro

Profesor guía: Profesor Roberto González Ibañez

Trabajo de titulación presentado en conformidad a los requisitos para obtener el título de Ingeniero Civil en Informática

Santiago – Chile
2018

RESUMEN

OpenGlove es un dispositivo diseñado en el Departamento de Ingeniería Informática por el grupo de investigación InTeracTion perteneciente a la Universidad de Santiago de Chile. El guante permite la retroalimentación vibrotáctil o *haptic feedback* cuando se interactúa con objetos en ambientes de realidad virtual (VR), aumentada (AR) o mixta (MR). Esta retroalimentación es generada a través de la vibración de motores distribuidos en distintas partes del guante, distribución que depende de los requerimientos del usuario o la aplicación. El guante también sensores de flexibilidad para la captura del movimiento de los dedos y una unidad de medición inercial o IMU(Inertial Measurement Unit) para capturar la orientación de la mano.

Si bien es posible tener una inmersión en estos ambientes, esta es parcial, dejando de lado el sentido del tacto. El estado actual de OpenGlove no permite el uso del mismo en comunidades de desarrollo de VR, AR y MR en entornos móviles como Android e iOS. Esto limita la portabilidad y desacople del sistema operativo Windows. Por ello lo que se propone es dar soporte a OpenGlove en dispositivos móviles enriqueciendo la interacción en los entornos ya mencionados. Esto se realizó mediante el desarrollo de un SDK para dispositivos móviles, el cual permite la conexión por Bluetooth de varias instancias de OpenGlove, como también la configuración de los mismos mediante la carga de perfiles utilizando una aplicación de configuración. Esta aplicación expone un servicio, el cual es utilizado mediante las APIs en Java y C#. Se realizaron evaluaciones de rendimiento y pruebas de concepto utilizando el sistema propuesto.

Palabras Claves: Haptic Feedback; Virtual Reality (VR); Augmented Reality (AR); Mixed Reality (MR); OpenGlove; SDK; API

Dedicado a...

AGRADECIMIENTOS

Agradezco a

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

CAPÍTULO 1. INTRODUCCIÓN

1.1 ANTECEDENTES Y MOTIVACIÓN

El tamaño del mercado de la realidad virtual (VR) [1] y realidad aumentada (AR)[2] registra 6.1 billones de dólares para el año 2016 y se estima que para el 2017 ascienda a 11.4 billones (Statista, 2016). En este contexto también aparece la denominada realidad mixta (MR) [3] la cual es un punto intermedio entre VR y AR. Actualmente es bastante popular el uso de VR, AR y MR en dispositivos móviles, pero estos carecen del soporte de otros sentidos como lo es el tacto lo cual genera una disrupción cuando se interactúa con los objetos virtuales. En este contexto se puede presenta una brecha cuando cuando se desea incluir retroalimentación vibrotáctil o el denominado *haptic feedback*[4] a aplicaciones en los entornos ya mencionados. Adicionalmente se incluye la captura de movimiento, para su representación en los ambientes ya mencionados.

En primer lugar, el tamaño del mercado entre el 2017 y 2020 para VR y AR espera más de un 1200% de aumento hablando de billones de dólares. Esto es una interesante oportunidad de negocio, como también el aprovechar las distintas comunidades de desarrollo de VR, AR y MR para masificar el uso de OpenGlove en proyectos de tales ambientes.

En segundo lugar, la disrupción que se genera cuando se interactúa con objetos virtuales y no se obtiene una respuesta similar a la experiencia real, crea un punto de quiebre entre lo real y lo virtual. Esto implica una inmersión parcial en los ambientes de VR, AR y MR.

En tercer lugar se tiene el alto costo asociados a la adquisición de los guantes. Es posible constatar precios de guantes que van desde los 300€ hasta los 1300€ y licencias desde los 2500€ hasta los 13300€. Esto se traduce costos más altos de desarrollo como también para los usuarios finales. Además genera un mercado con un alcance más limitado.

En base a los argumentos desarrollados, se evidencia una brecha en la inclusión de Haptic Feedback y la captura de movimiento de las manos en proyectos de VR, AR y MR en dispositivos móviles, la cual actualmente no es cubierta de manera global.

El estado actual de OpenGlove no permite el uso del mismo en comunidades de desarrollo de VR, AR y MR en entornos móviles como Android e iOS. Esto limita la portabilidad y desacople del sistema operativo Windows, reduciendo el alcance que puede tener en las ya mencionadas comunidades de desarrollo.

[1] **Virtual Reality (VR):** "La realidad virtual (VR) proporciona un entorno 3D generado por computadora que rodea al usuario y responde a las acciones de esa persona de forma natural ..." Gartner (2017d).

[2] **Augmented Reality (AR):** "es el uso de información en tiempo real en forma de texto, gráficos, audio y otras mejoras virtuales integradas con objetos del mundo real ... "" (Gartner, 2017a)

[3] **Mixed reality (MR):** "es el resultado de mezclar el mundo físico con el mundo digital, incluyendo la interacción con objetos virtuales representados en el real ..." (Microsoft, 2017).

[4] **Haptic Feedback:** Haptics es una tecnología táctil o de retroalimentación de fuerza que aprovecha el sentido del tacto de una persona al aplicar vibraciones y / o movimiento a la punta del dedo del usuario ..." (Gartner, 2017b)

1.2 DESCRIPCIÓN DEL PROBLEMA

Actualmente el proyecto OpenGlove posee la arquitectura que se aprecia en la Figura 1.1. El guante posee motores distribuidos en distintos lugares de la mano, los cuales pueden ser activados y desactivados según se requiera mediante APIs (Monsalve, 2016), también se tienen sensores de flexibilidad y la captura de movimientos de la mano (Cerda, 2017) . Existe un servicio que utiliza SOAP y REST para exponer los servicios mediante Bluetooth en Windows. Luego las APIs de alto nivel (Meneses, 2016) para lenguajes de programación como C#, C++, Java, y JavaScript , permiten la abstracción de la complejidad en el uso de instancias y configuración de OpenGlove. El estado actual de OpenGlove no permite el uso del mismo en comunidades de desarrollo de realidad virtual móvil en Android VR[5], ni iOS [6] sin depender del sistema que permite la configuración y la ejecución del servicio en Windows. Lo cual dificulta la integración de OpenGlove para soluciones en dispositivos móviles y la independencia de servicios alojados en otro sistema operativo, para poder desarrollar aplicaciones de VR, AR o MR para dispositivos móviles.

El problema se puede resumir en la siguiente pregunta ¿Cómo facilitar a la comunidad de desarrolladores de VR/AR/MR a integrar OpenGlove en entornos móviles y realizar una fácil configuración para los dispositivos?.

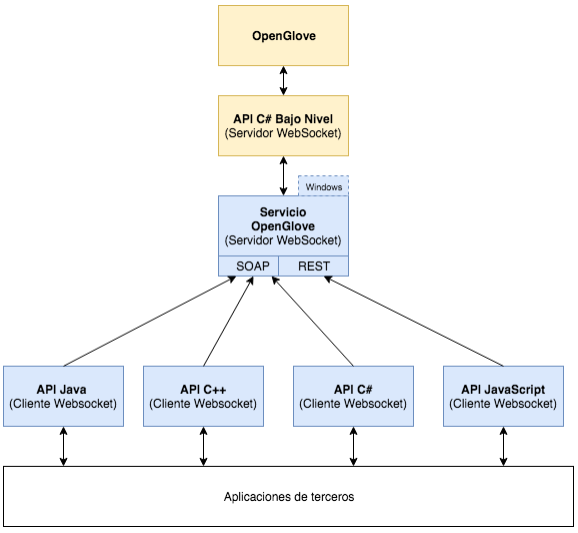

Figura 1.1: Arquitectura OpenGlove
Fuente: Elaboración propia (2018)

[5]Comunidad Android: `https://developers.google.com/vr/android/`
[6]Comunidad iOS: `https//developers.google.com/vr/ios/`

1.3 SOLUCIÓN PROPUESTA

A continuación se describen las características y el propósito de la solución propuesto para el problema planteado.

1.3.1 Características de la solución

Para resolver el problema, se propuso una solución para Android, la cual corresponde a un SDK que incluye las herramientas necesarias para el desarrollo de aplicaciones de terceros relacionadas a VR/AR/MR. El SDK incluye la documentación, APIs y software de configuración. Las APIs permiten activar y desactivar los distintos motores y sensores distribuidos en el guante como también el agregar, quitar y listar dispositivos Bluetooth. El software de configuración permite levantar un servicio Bluetooth en segundo plano, que permita agregar, activar, desactivar y listar dispositivos conectados. También se pueden guardar los perfiles de configuración de los guantes en el dispositivo. Dicha aplicación de configuración soporta la conexión de múltiples dispositivos, para lograr utilizar uno o más OpenGlove desde dispositivos móviles.

Cabe destacar que el SDK no implica un costo para los desarrolladores, ni para los usuarios del guante.

Las tecnologías que permiten implementar las características anteriormente mencionadas, son las siguientes:

- Servicios en segundo plano en Android [7], que permiten mantener un servicio que pueda seguir funcionando mientras otras aplicaciones se estén utilizando en el dispositivo.
- Websocket [8], tecnología que proporciona un canal de comunicación bidireccional y full-duplex (información enviada simultáneamente) sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor.
- Bluetooth[9], tecnología de red inalámbrica que permite la transmisión de datos entre dispositivos.
- Xamarin.Forms[10] tecnología que permite el desarrollo de aplicaciones multiplataforma para

[7] Background service Android: `https://developer.android.com/training/run-background-service/create-service.html`
[8] Websocket: `http://websocket.org/aboutwebsocket.html`
[9] Bluetooth Android: `https://developer.android.com/guide/topics/connectivity/bluetooth.html`
[10] Xamarin.Forms: `https://docs.microsoft.com/en-us/xamarin/xamarin-forms/`

iOS y Android. En este proyecto, permite compartir la interfaz de usuario en ambos sistemas operativos de manera nativa y realizar las implementaciones específicas para cada uno.

La Figura 1.2 muestra la ubicación del proyecto considerando la arquitectura de OpenGlove. Es importante considerar que las APIs y servicios desarrollados considera Android como alcance.

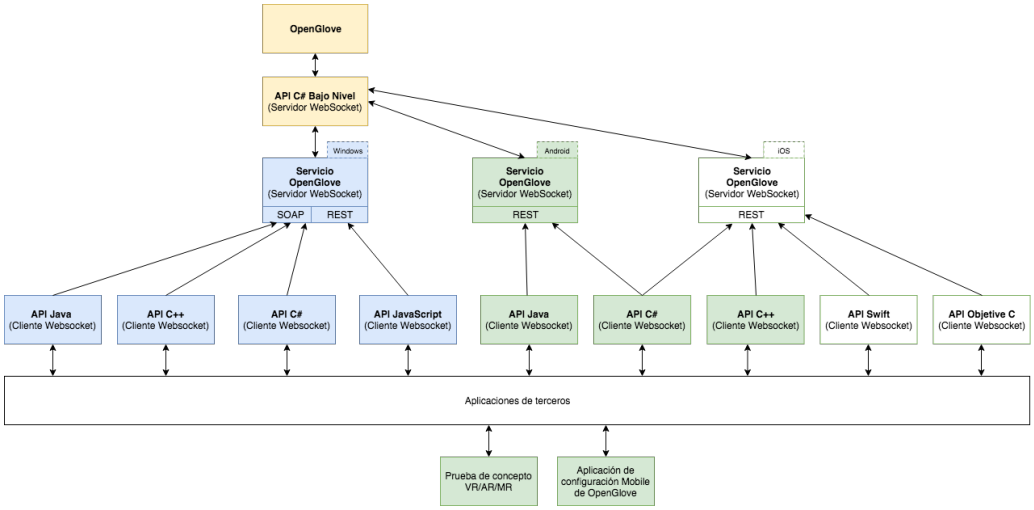

Figura 1.2: Arquitectura OpenGlove
Fuente: Elaboración propia (2018)

1.3.2 Propósitos de la solución

El propósito del proyecto es poner al alcance de distintas comunidades las herramientas para el desarrollo y uso de dispositivos de respuesta vibrotáctil y seguimiento de las manos en ambientes de realidad virtual, mixta y aumentada, junto con promover el uso de OpenGlove en la comunidad Android, los entusiastas del DIY (Do It Yourself) o “házlo tú mismo” y fabricantes que quieran hacer uso de estas tecnologías para sus proyectos.

1.4 OBJETIVOS Y ALCANCE DEL PROYECTO

1.4.1 Objetivo general

El objetivo general del proyecto es desarrollar un SDK que permita dar soporte a OpenGlove en dispositivos móviles.

1.4.2 Objetivos específicos

En esta sección se listan los resultados parciales definidos para lograr el objetivo general del proyecto de titulación.

1. Desarrollar la aplicación de configuración de OpenGlove en Android. Permitiendo la creación, visualización, actualización y eliminación de los perfiles de configuración.
2. Desarrollar y documentar APIs en Java y C# que permitan la creación de servicios Bluetooth en segundo plano en Android para la conexión, desconexión, activación y listado de guantes OpenGlove. También permitirá la activación y control de actuadores [11], flexores [12] e IMU (*Inertial Measurement Unit*).
3. Demostrar el uso del SDK en un ambiente de VR, AR o MR, utilizando OpenGlove y el visor de realidad virtual ZapBox[13].

1.5 METODOLOGÍAS Y HERRAMIENTAS UTILIZADAS

En esta sección se presenta la metodología utilizada a lo largo del proyecto, las herramientas y el ambiente de desarrollo del SDK que permitirá el soporte de OpenGlove en dispositivos móviles.

[11] Definición actuadores: Un actuador es un dispositivo inherentemente mecánico cuya función es proporcionar fuerza para mover o "actuar" otro dispositivo mecánico. Dependiendo de el origen de la fuerza el actuador se denomina "neumático", "hidráulico" o "eléctrico" (Aie, 2017)

[12] "... es un sensor de flexión que produce una resistencia variable en función del grado al que este doblada".(Rambal, 2018)

[13] Visor Zapbox: `https://www.zappar.com/zapbox/`

1.5.1 Metodología a usar

Para alcanzar los objetivos planteados en este proyecto, también es necesario elegir una metodología adecuada para el proyecto. En base a esto y a la incertidumbre que presenta el proyecto, es necesario que se elija una metodología que reduzca los riesgos asociados, permitiendo entregas de software funcional de manera rápida y que puedan ser validados por el cliente (profesor guía). En este contexto y dado que el proyecto de título debe ser realizado dentro del semestre considerando un trabajo de 612 horas cronológicas y basado en la experiencia y el desarrollo de proyectos de similares características, la metodología planteada a continuación es la adecuada para el presente proyecto de ingeniería propuesto.

La metodología corresponde a la utilizada por el grupo de investigación y desarrollo InTeracTion. Puede separarse en dos etapas principales. La primera etapa consiste en el desarrollo de prototipos funcionales, con la finalidad de obtener retroalimentación del cliente y así cubrir los requerimientos del cliente de manera iterativa. Esta etapa tiene como referencia la metodología RAD (Rapid Application Development) (Martin, 1991), la cual está orientada a grupos pequeños, enfocada en el producto y no en la documentación generada, necesitando entregas rápidas para obtener retroalimentación del producto desechable desarrollado. El prototipo final aprobado será la base con lo que se desarrollará la siguiente etapa. La segunda etapa consiste en el desarrollo de las funcionalidades requeridas para el producto, utilizando como base el prototipo funcional de la primera etapa. El proyecto se desarrolla de manera iterativa e incremental, para su gestión se establece como mínimo una reunión semanal en la cual se tratan temas como el estado de avance, los compromisos asumidos para la semana, los compromisos siguientes y los problemas ocurridos. En estas reuniones es posible tomar decisiones sobre los problemas ocurridos y cambiar de estrategia según sea necesario, esto permite reducir los riesgos ocasionados por la incertidumbre del proyecto.

En resumen la metodología a utilizar contempla las siguiente características:

- Primera etapa de desarrollo mediante prototipos basada en RAD.
- Segunda etapa de desarrollo iterativo del prototipo maduro y aceptado.
- Como mínimo reuniones una vez a la semana en todas las etapas.

Para la gestión del proyecto se utilizan los siguientes recursos, que permiten tener un seguimiento del trabajo a realizar, :

- Kanban físico y compromisos semanales.
- Kanban simple de tres columnas o estados por hacer (TO DO), haciendo (DOING) y hecho (DONE).

Para que el proyecto llegue a buen término, se debe elegir la metodología adecuada al proyecto dado el contexto del mismo. Dada las características y la interacción que se desea establecer con el profesor guía como cliente, se adoptó la metodología utilizada por InTeracTion. Además, esta metodología presenta ventajas por sobre las tradicionales respecto de la adaptabilidad, colaboración con el cliente y entregas funcionales iterativas que pueden ser evaluadas cada cierto tiempo, mostrando los avances y favoreciendo el producto funcionando por sobre la documentación exhaustiva [14].

1.5.2 Herramientas de Software

En esta sección se listan las herramientas de Software que se utilizaron para el desarrollo del proyecto de título. La aplicación fue desarrollada utilizando el IDE Visual Studio Community 2018 para Mac [15] en conjunto con el ecosistema de plataformas para compilar [16] aplicaciones móviles Xamarin.Forms, el cual utiliza el lenguaje de programación C# para el desarrollo de aplicaciones nativas multiplataforma. Adicionalmente se utilizó el IDE Android Studio [17], el cual corresponde al IDE oficial para el sistema operativo Android, el cual integra distintas herramientas de desarrollo, como el SDK de Android, el editor de texto, el soporte para control de versiones, compilador, etc. A continuación se listan otras herramientas de software para el apoyo para el desarrollo del proyecto.

1. Github[18] para el control de versiones.
2. Texmaker [19], para la escritura de la memoria.
3. RStudio[20]: para el análisis de datos de la evaluación del proyecto.
4. Google Drive[21], para generar y compartir documentos y archivos de manera colaborativa.
5. Microsoft Project Professional y Gantter, para el desarrollo de la carta gantt del proyecto de título.

[14] Manifiesto ágil: `http://agilemanifesto.org/`
[15] Xamarin más VS Community para Mac 2018: `https://store.xamarin.com/`
[16] En resumidas palabras un compilador es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación.Este proceso de traducción se conoce como compilación. `http://www.ictea.com/cs/knowledgebase.php?action=displayarticle&id=8817`
[17] Android Studio: `https://developer.android.com/studio/index.html?hl=es-419`
[18] Github: `https://github.com/`
[19] Texmaker : `http://www.xm1math.net/texmaker/`
[20] Rstudio: `https://www.rstudio.com/`
[21] Google Drive: `https://www.google.com/intl/es_ALL/drive/`

1.5.3 Herramientas de Hardware

El ambiente de desarrollo en el cual se desarrolló SDK fue un Macbook Pro con las siguientes características:

1. Sistema operativo SO macOS Sierra versión 10.13.3.
2. Procesador Intel Core i5, 3,1 GHz .
3. Memoria RAM 8GB 2133 MHz LPDDR3.
4. 512 GB disco duro SSD.
5. Gráficos Intel Iris Plus Graphics 650 1536 MB.

Las pruebas de la aplicación se realizarán utilizando Android Virtual Device (AVD) mediante Android Studio y un smartphone *Samsung Galaxy S5 mini* [22] con las siguientes características:

1. Sistema Operativo Android Marshmallow 6.0.1.
2. CPU: 1.4Ghz Quad-Core ARM Cortex-A7.
3. GPU: ARM Mali-400 MP4 450Mhz.
4. Memoria RAM 1,5GB LPDDR2.
5. Almacenamiento interno 16GB (12GB accesible al usuario).
6. Bluetooth Versión 4.0 con A2DP.
7. Sensores: Acelerómetro, Proximidad, Brújula , Luz ambiental, Giroscopio, Biométrico (huellas digitales).

El prototipo de dispositivo OpenGlove disponible en InTeracTion fue utilizado para realizar las pruebas y el desarrollo del SDK. Se usaron el IMU, motores y sensores de flexibilidad disponibles en el prototipo.

1. Sensores de flexibilidad de 2,2”, SparkFun.
2. Sensor de rastreo IMU, SparkFun.
3. Actuadores (En específico motores de vibración).
4. Bluetooth mate silver, Sparkfun.

[22] http://www.movilcelular.es/samsung-galaxy-s5-mini-duos-sm-g800h/caracteristicas/1659

1.6 ORGANIZACIÓN DEL DOCUMENTO

El presente documento posee seis capítulos que abarcan la totalidad de este proyecto de desarrollo en sus distintas fases. El Capítulo 1, Introducción, incluye los antecedentes y motivación del proyecto, presentando el problema y solución propuesta junto con los objetivos necesarios para concretarla. El Capítulo 2, Marco teórico, introduce los conceptos relevantes que permitirán una mejor compresión del problema junto a su solución. También se incluye el estado del arte, el cual detalla las soluciones alternativas actuales del problema planteado, detallando y comparándolas entre ellas. En el Capítulo 3, Análisis, se realiza un análisis sobre el desarrollo del SDK analizando los componentes de hardware y software disponibles, generando prototipos hasta que se genere el producto esperado. Luego en el Capítulo 4, Diseño e implementación, se diseña la solución a nivel de mockups o maquetas, también a nivel arquitectural, comportamiento y de protocolos de comunicación. Posteriormente se implementan para lograr la solución diseñada. El Capítulo 5, Evaluación técnica, se realiza una evaluación del software desarrollado para establecer cuáles son los tiempos de respuesta (latencia) presentes y la comparativa con la versión de escritorio. Además se considera otro factor importante respecto a la solución , esto es, la eficiencia energética de la misma. Por último, en el Capítulo 6, Conclusiones, se detallan los objetivos cumplidos, también los resultados obtenidos con la solución, los alcances y limitaciones de la misma, posibles mejoras para trabajos futuros y observaciones finales pertinentes.

CAPÍTULO 2. MARCO TEÓRICO

Este capítulo permite establecer las bases teóricas necesarias para una mejor comprensión del presente documento. El capítulo se divide en tres secciones, la primera correspondiente al Marco conceptual, en el cual se establecen los conceptos bases pertinentes al proyecto, luego en el Estado del arte se realiza una revisión del mismo.

2.1 MARCO CONCEPTUAL

2.1.1 API

2.1.2 SDK

2.1.3 Haptics

2.1.4 Tipos de aplicaciones móviles

2.1.4.1 Nativas

2.1.4.2 Híbridas

2.1.4.3 Web

2.2 ESTADO DEL ARTE

En esta sección se busca presentar los antecedentes y el estado actual sobre los dispositivos de retroalimentación vibrotáctil disponibles en el mercado, las cuales suelen venir con herramientas para el desarrollo de aplicaciones de terceros. Una de las herramientas que se le suelen facilitar a los desarrolladores es el SDK (*Software Development Kit*), el cual es un conjunto de herramientas utilizadas para desarrollar aplicaciones proporcionadas por proveedores de hardware y software. Los SDK suelen estar compuestos por interfaces de programación de aplicaciones APIs (*Application Program Interface*), código de muestra, documentación, entre otros elementos (Techopedia, 2017). En la Tabla 2.1 muestra una comparativa de distintas características de los SDKs del mercado referente a dispositivos de *haptic feedback*. Es importante

señalar que el SDK no posee soporte multiplataforma, tampoco tiene una integración directa con Unity, sería necesario hacer un plugin para ello.

Tabla 2.1: Comparación SDK de distintos Guantes
Fuente: Elaboración propia (2018)

	Lenguajes soportados	SO soportado	Información Bidireccional	Integración SDK con Unity
OpenGlove	C#, Java, JavaScript	Windows	SI	NO (Requiere plugin)
GloveOne	C++ y C#	Windows, MAC, Linux, Android e iOS	SI	SI
AvatarVR	C++ y C#	Windows, MAC, Linux, Android e iOS	SI	SI
Dexmo	No anunciado	No anunciado	SI	Si

2.2.1 OpenGlove

Es un proyecto que consiste en un dispositivo que entrega *haptics feedback*. Fue pensado para dispositivos de realidad virtual e interfaces naturales. El proyecto partió en el 2014 con el propósito de facilitar la construcción y flexibilizar uno de los recursos claves para la inmersión en ambientes de realidad virtual (InTeracTion, 2018). OpenGlove se ha pensado para que pueda ser utilizado en conjunto con otros dispositivos, como Oculus Rift, Kinect y Leap Motion. Por otra parte, los prototipos de OpenGlove, utilizan motores configurables con diferentes niveles de potencia, lo que permite la respuesta vibrotáctil en distintas áreas de la mano. Esto puede ser utilizado para representar la sensación de tocar objetos en entornos virtuales, como también, recibir retroalimentación que represente impacto, el cual sería útil en un juego de boxeo por ejemplo. Actualmente, se soporta el uso de actuadores, sensores de flexibilidad y Unidad de Medición Inercial (IMU por sus siglas en inglés).

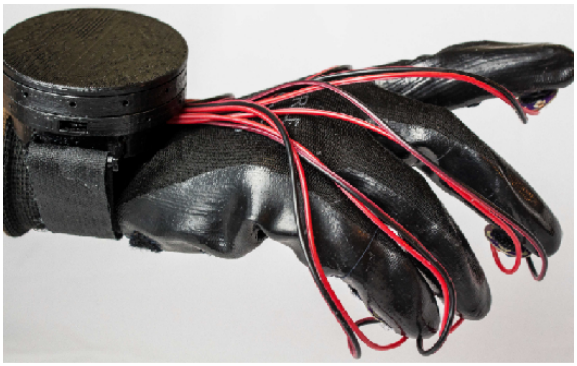

Figura 2.1: Guantes OpenGlove
Fuente: InTeracTion (2018)

2.2.2 AvatarVR

AvatarVR es otro guante diseñado por Neurodigital e incluye todas las funcionalidades presentes en GloveOne, junto a unas capacidades adicionales que ofrecen más funcionalidades. Además de los guantes se incluye un accesorio llamado TrackBand, que permite la captura de movimiento de la parte superior del cuerpo, basada en una configuración minimalista de los sensores para en brazos y manos. Además sensores para el seguimiento de dedos mediante sensores 6x 9-AXIS IMUs. El guante y las trackbands poseen un costo desde 1.100 € . Las licencias son de dos tipos profesional a 3.300 € y premium a 13.300 € para acceder a la documentación, SDK, soporte, entre otros servicios. [1]

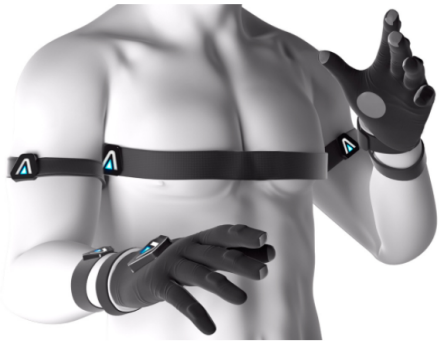

Figura 2.2: Guantes AvatarVR y TrackBand
Fuente: Neurodigital (2018)

2.2.3 Dexmo

Dexmo es un exoesqueleto que permite *haptic feedback* en ambientes de realidad virtual. Esto lo logra mediante la capacidad de retroalimentación de fuerza, lo que permite al usuario sentir el tamaño y la forma de cualquier objeto digital, lo que mejora enormemente la inmersión. La rigidez variable se logra mediante un control preciso del motor. Con esta característica, cada objeto virtual puede tener su propia rigidez.

Por protección de propiedad intelectual, su SDK sólo es accesible a clientes que hayan comprado Dexmo DK1.

[1] Precio AvatarVR y licencias: `https://www.neurodigital.es/avatarvr/`

Figura 2.3: Guantes Dexmo
Fuente: DextaRobotics (2018)

2.2.4 Manus VR

Manus VR es un guante que permite *haptic feedback*, seguimiento de dedos y manos para ambientes de realidad virtual. Sus especificaciones comerciales establecen que, es lavable, permite el seguimiento de los brazos, que también incluye un IMU para medir la orientación de la mano. Manus VR puede ser adquirido en dos versiones, la de desarrollador con un precio de 1.990 € y una versión profesional con un precio de 4.990 € [2]

Figura 2.4: Guantes Manus VR
Fuente: ManusVR (2018)

[2]Precio de Manus VR y licencias `https://manus-vr.com/order.php`

2.2.5 Haptx

3

2.2.6 Sense glove

4

2.2.7 Virtual Reality Smart Glove

5

2.3 RESUMEN

OpenGlove es un proyecto Open Source[6] que permite la retroalimenación vibrotáctil y comunicación bidireccional entre el guante y las aplicaciones mediante el uso del protocolo WebSocket. Como se ha podido ver, existen diversas alternativas en el mercado de dispositivos hápticos, siendo OpenGlove la alternativa Open Source que no requiere de costos altos de licencias ni del guante, el cual es desarrollado según las necesidades específicas requeridas. Para lograr que la comunidad de desarrolladores de VR/AR/MR pueda integrar OpenGlove en entornos móviles y realizar una fácil configuración de ellos, se desarrollará un SDK que incluye las APIs de alto nivel necesarias para el desarrollo en las plataformas móviles, la aplicación de configuración y la documentación de uso de las APIs en pruebas de concepto. Es importante señalar la importancia de las pruebas de rendimiento sobre la solución a desarrollar. esto toma relevancia cuando se busca disminuir la latencia de los dispositivos conectados. Esto se debe considerar para brindar una excelente experiencia en entornos virtuales sin que se presenten retardos perceptibles.

[3] Haptx `https://haptx.com/`

[4] Sense Glove: `https://www.senseglove.com/`

[5] VR Smart Glove: `http://www.designpartners.com/projects/virtual-reality-smart-glove/`

[6] “Open source o código abierto es el término empleado al software distribuido bajo una licencia que permite al usuario acceso al código fuente. Este tipo de licencia posibilita el estudio y la modificación del software con total libertad. Además, su redistribución está permitida siempre y cuando esta posibilidad vaya en concordancia con los términos de licencia bajo la que se adquiere el software” (Ticportal, 2018).

CAPÍTULO 3. ANÁLISIS

En este capítulo se realiza un levantamiento de requisitos funcionales y no funcionales del SDK a desarrollar, considerando el estado actual del proyecto. Para luego realizar los diferentes prototipos de software basados en la metodología RAD, hasta obtener un producto final que cumpla con todos los requisitos establecidos en un comienzo.

3.1 LEVANTAMIENTO DE REQUISITOS DE SOFTWARE

3.1.1 Antecedentes

En los trabajos anteriores se ha logrado establecer las bases de hardware y software para

3.1.2 Requisitos

Al realizar un análisis de los antecedentes y el problema planteado en el Capítulo 1, se capturan los siguientes requisitos funcionales del software.

Tabla 3.1: Requisitos funcionales de software
Fuente: Elaboración propia (2018)

ID	Síntesis del Requisito	Descripción	Origen
RF001	El sistema debe permitir al usuario ver los dispositivos OpenGlove emparejados	El sistema debe ofrecer una recopilación de los dispositivos OpenGlove emparejados con el dispositivo móvil, su estado de conexión actual y la dirección MAC del mismo.	Inicio

RF002	El sistema debe permitir al usuario definir la configuración de hardware de cada guante	Cada placa LilyPad posee pines programables, en los cuales el usuario puede conectar actuadores, sensores de flexibilidad e IMU para crear su propio OpenGlove. Es necesario que se establezca esta configuración en el sistema para cada guante, ya que de ella depende la activación de actuadores y la lectura de datos de los flexores e IMU.	Inicio
RF003	El sistema debe permitir al usuario guardar una configuración de hardware	Al crear un nuevo perfil de hardware para un guante, el sistema debe contar con un mecanismo para la persistencia de esta configuración.	Inicio
RF004	El sistema debe permitir al usuario abrir una configuración de hardware previamente almacenada por el sistema	Una vez guardada una configuración, esta debe ser reconocible por el sistema para su uso posterior en otro guante.	Inicio
RF005	El sistema debe permitir al usuario definir la configuración de actuadores de cada guante	Dependiente de la configuración de hardware, la configuración de actuadores es una representación de la distribución física de los actuadores LilyPad en una mano virtual. Esta representación permite establecer mapeos región-actuador usables en una API de alto nivel. Se debe ofrecer al usuario una solución gráfica que permita ordenar la posición de los actuadores en una representación de la mano.	Inicio
RF006	El sistema debe permitir al usuario guardar una configuración de actuadores	Al crear un nuevo perfil de actuadores para un guante, el sistema debe contar con un mecanismo para la persistencia de esta configuración.	Inicio

RF007	El sistema debe permitir al usuario abrir una configuración de actuadores previamente almacenada por el sistema	Una vez guardada una configuración de actuadores, esta debe ser reconocible por el sistema para su uso posterior en otro guante.	Inicio
RF008	El sistema debe permitir al usuario establecer conexión con un dispositivo OpenGlove emparejado	Una vez emparejado un guante OpenGlove, el sistema debe permitir que el usuario inicie la conexión Bluetooth. No es necesario que el usuario especifique la dirección MAC del guante.	Inicio
RF009	El sistema debe permitir al usuario activar una región de un guante con intensidad a voluntad	El sistema debe exponer al usuario una sección que le permita activar una región de un guante con la intensidad (entre 0 y 255) que él desee para probar el hardware. Esta región esta predefinida y debe ser independiente de la configuración de hardware (actuadores) presente en el guante. Esta función debe estar disponible para uno o varios actuadores en un guante.	Inicio
RF010	El sistema debe permitir al usuario operar con distintas implementaciones de OpenGlove	Al momento de crear un nuevo perfil de hardware, el sistema debe proveer un mecanismo para que el usuario genere su propia placa, lo que se traduce en un nombre y una cantidad de pines para poder usarla en su configuración.	Inicio
RF011	El sistema actual debe permitir al usuario guardar y cargar una configuración de hardware incluyendo los flexores	El SDK debe ser capaz de guardar y cargar una configuración de hardware, compuesta por actuadores y/o sensores de flexibilidad.	Inicio
RF012	El sistema debe permitir al usuario crear una configuración de los sensores de flexibilidad y del sensor de rastreo IMU	El software debe ser capaz de dar soporte para la creación de nuevas configuraciones de los sensores de flexibilidad y el IMU.	Inicio

RF013	El sistema debe permitir al usuario seleccionar un sensor de flexibilidad y relacionarlo a una región del guante	La configuración correspondiente a los sensores de flexibilidad, debe ser capaz de seleccionar una región del guante y relacionarlo con un sensor de flexibilidad.	Inicio
RF014	El sistema debe permitir al usuario eliminar un sensor de flexibilidad de una región del guante	La configuración de los sensores de flexibilidad debe ser capaz de eliminar un flexor de una región del guante, de tal manera que la región quede libre y el flexor pueda asignarse a una nueva región.	Inicio
RF015	El sistema debe enviar los datos provenientes de los sensores de flexibilidad automáticamente	Cuando un sensor de flexibilidad es asignado a una región del guante, el sistema debe ser capaz de transmitir los datos de dicho sensor de manera automática, especificando el tipo de dato, región y el valor leído.	Inicio
RF016	El sistema debe parar de enviar los datos provenientes de los sensores de flexibilidad automáticamente	Cuando un sensor de flexibilidad es eliminado de una región del guante, el sistema debe ser capaz de parar la transmisión de datos de dicho flexor automáticamente.	Inicio
RF017	El sistema debe permitir al usuario definir un threshold al momento de enviar datos de los sensores de flexibilidad	El guante enviará el dato de un flexor, solo si este dato posee una diferencia mayor o igual al valor definido como threshold, con respecto al último valor enviado por dicho flexor.	Inicio
RF018	El sistema debe permitir al usuario la opción de calibrar los sensores de flexibilidad	Los datos provenientes de los sensores de flexibilidad deben ser calibrados con respecto al máximo y mínimo valor leído de la articulación de un dedo.	Inicio
RF019	El sistema debe permitir al usuario probar los sensores de flexibilidad	Luego de asignar uno o más sensores de flexibilidad a una región del guante, se debe habilitar un botón que permita probar si la configuración es correcta, visualizando el valor entregado por cada flexor en dicha región.	Inicio

RF020	El usuario debe permitir al usuario poder obtener datos desde un sensor de rastreo IMU	La configuración correspondiente al sensor de rastreo IMU, debe ser capaz de activar o desactivar el envío de datos de ésta.	Inicio
RF021	El sistema debe permitir al usuario poder obtener datos crudos o procesados desde el sensor de rastreo IMU	La configuración del sensor de rastreo IMU, debe ser capaz de definir si los datos enviados por ésta son procesados o no.	Inicio
RF022	El sistema debe permitir al usuario poder probar el sensor de rastreo IMU	Luego de activar el envío de datos del sensor de rastreo IMU, se debe activar un botón que active la visualización de todos los datos entregados por el sensor.	Inicio
RF023	El sistema debe permitir al usuario calibrar el sensor de rastreo IMU	Los datos provenientes del IMU deben ser calibrados bajo una posición de referencia, para poder determinar la posición y orientación de la mano.	Nuevo

3.2 PROTOTIPOS

En la presente sección, se muestran los diferentes prototipos desarrollados durante la escritura de esta memoria. Los prototipos serán detallados utilizando una tabla resumen del mismo, la cual incluye, los Objetivos del prototipo, una Descripción, los requisitos funcionales y no funcionales que aborda el prototipo. Se incluyen las capturas del trabajo realizado, análisis y conclusión del avance .

3.2.1 Primer prototipo: Activación de motores

Este primer prototipo tiene como principal objetivo establecer las bases necesarias para conectar de OpenGlove con un dispositivo Android y establecer una comunicación básica. Para ello se utiliza la documentación oficial de Android sobre las conexiones bluetooth [1], permitiendo en primera instancia la obtención de dispositivos vinculados y la conexión con alguno

[1]Documentación oficial android: `https://developer.android.com/guide/topics/connectivity/bluetooth`

de los mismos. No es necesaria la modificación de código cargado en la placa arduino, puesto que los protocolos de comunicación (mensajes) ya han sido establecidos con anterioridad. Por tanto se procede a hacer uso de la API en Java de bajo nivel desarrollada por Monsalve (2015). El prototipo se resume en la Tabla **??**.

Tabla 3.2: Primer prototipo
Fuente: elaboración propia (2018).

ID del prototipo	**P001**
Nombre	Activación de motores.
Objetivos	Verificar la correcta activación de motores en una aplicación de Android nativo, utilizando las APIs de bajo nivel disponibles.
Descripción	El primer prototipo desarrollado hace uso de la API de bajo nivel de Java desarrollada por Monsalve (2015), sumándose modificaciones realizadas para establecer la conexión en Android. De esta manera, se obtiene un prototipo capaz de establecer una conexión bluetooth con dispositivos previamente vinculados, permitiendo activar y desactivar un motor de manera remota.
Requisitos funcionales	RF 000
Requisitos no funcionales	RNF 000

Para lograr el objetivo propuesto, luego obtener los dispositivos vinculados y de la conexión con el guante mediante las APIs de bluetooth de Android, se procedió a enviar mensajes bajo el protocolo establecido en el desarrollo de Monsalve (2015). Dicho de una manera más detallada se utilizó la clase *Message Generator* de la API de bajo nivel y la implementación nativa en Android para la escritura serial por medio de bluetooth. La Figura 3.1 consiste en el primer prototipo que muestra el listado de los dispositivos vinculados, la posibilidad de la conexión con el dispositivo deseado, permitiendo finalmente activar y desactivar el motor seleccionado para las pruebas. Para administrar la conexión entre el dispositivos, se requiere de un hilo encargado de ello (*ConnectedThread*), el cual se comunica con el hilo de la *interfaz de usuario* (UI desde ahora) mediante mensajes. En conclusión es posible realizar envio de mensajes bajo el protocolo que acepta OpenGlove desde una aplicación Android nativa en Java.

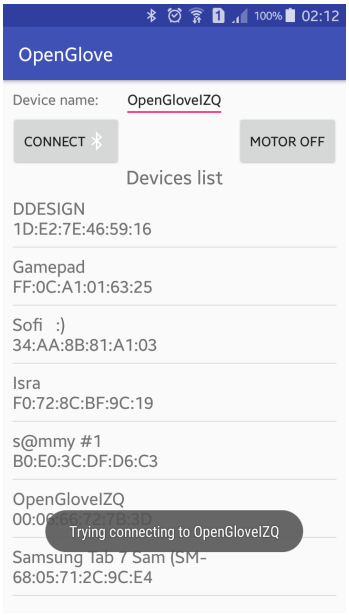

Figura 3.1: Primer prototipo

3.2.2 Segundo prototipo: Obtención de datos desde flexores

En el segundo prototipo se mantiene lo desarrollado previamente, añadiendo en esta iteración el soporte para los flexores. En este caso es necesaria la lectura desde el dispositivo OpenGlove. En la Tabla 3.3 se muestra el resumen del prototipo ya mencionado.

Tabla 3.3: Segundo prototipo

ID del prototipo	**P002**
Nombre	Obtención de datos desde los flexores.
Objetivos	Verificar la correcta obtención de datos desde los flexores en la aplicación de Android nativo, utilizando las APIs de bajo nivel disponibles.
Descripción	El segundo prototipo desarrollado agrega los métodos disponibles de los flexores de la API de bajo nivel C# hecha por Cerda (2017). De esta manera, se obtiene un prototipo capaz de obtener los datos del flexor.
Requisitos funcionales	RF 000
Requisitos no funcionales	RNF 000

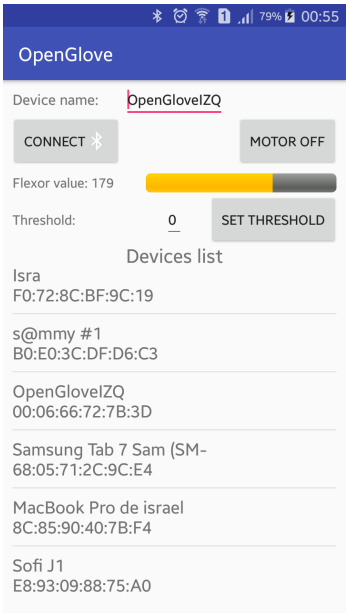

Figura 3.2: Segundo prototipo

Para realizar la captura de datos del flexor, se obtiene valor actual del pin al cual el está conectado, esto se logra con el desarrollo de la función analogRead(pin) en Java basada en la API C# de bajo nivel Cerda (2017). El hilo encargado de la administrar conexión (*ConnectedThread*), tiene la responsabilidad de leer los mensajes desde OpenGlove y actualiza la UI enviando como un mensaje ente hilos el valor obtenido del flexor. Además se agregan las demás funciones de generación de mensajes relacionadas a los flexores en la API C# ya mencionada. En la figura 3.2 se puede ver el estado actual del flexor el cual varia en un rango de entre 60 a 300 y 170 el valor medio del flexor sin aplicar fuerza.

3.2.3 Tercer prototipo

En el segundo prototipo fue posible la activación del motor y obtener la información del flexor, permitiendo así comprobar la factibilidad de un desarrollo nativo en Android. En este tercer prototipo, se buscó dar soporte multiplataforma al proyecto, considerando la importancia de mantener umbrales de latencia, se optó por Xamarin. En concreto Xamarin.Form, que es una tecnología de desarrollo móvil multiplataforma [2], el cual permite desarrollar aplicaciones nativas para Android e iOS en C#. La tabla 3.4 muestra el resumen del tercer prototipo.

[2]Traducción libre

Tabla 3.4: Tercer prototipo

ID del prototipo	**P003**
Nombre	Activación de motores y obtención de datos desde los flexores.
Objetivos	Verificar la correcta activación de motores y la obtención de datos desde los flexores en la aplicación de Android nativo con Xamarin.Forms, utilizando las APIs C# de bajo nivel disponibles.
Descripción	El tercer prototipo desarrollado agrega las mismas funcionalidades que en el segundo prototipo. De esta manera, se obtiene un prototipo capaz de obtener los datos del flexor.
Requisitos funcionales	RF 000
Requisitos no funcionales	RNF 000

La figura 3.3 muestra el prototipo hecho en Xamarin.Forms, el cual es similar al segundo prototipo, con la diferencia en la forma de conectarse a un dispositivo bluetooth, el cual difiere en la forma de conectarse, este prototipo requiere presionar el dispositivo y aceptar el mensaje que explica el intento de conexión.

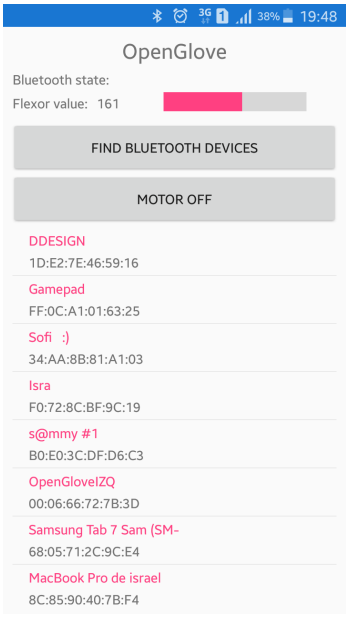

Figura 3.3: Tercer prototipo

3.2.4 Cuarto prototipo

3.3 APIS

3.3.1 Primer prototipo?

3.3.2 Segundo prototipo

3.3.3 Tercer prototipo

3.3.4 Cuarto prototipo

3.4 RESUMEN

CAPÍTULO 4. DISEÑO E IMPLEMENTACIÓN

4.1 ARQUITECTURA GENERAL

4.2 ESTRUCTURA

4.2.1 Servicios

4.2.2 APIs

4.2.3 Software de configuración

4.3 COMPORTAMIENTO

4.3.1 Obtener guantes

4.3.2 Activación

4.3.3 Añadir flexor a una región

4.3.4 Asignar Threshold

4.3.5 Asignar IMU

4.3.6 Lectura de datos proveniente de Arduino

4.4 ASPECTOS DE IMPLEMENTACIÓN

4.4.1 Desarrollo multiplataforma en Xamarin.Forms

4.4.2 Servicio

CAPÍTULO 5. EVALUACIÓN TÉCNICA

5.1 EVALUACIÓN APLICACIONES NATIVA Y MULTIPLATAFORMA

A continuación se realizará una evaluación de los prototipos 3 y 4 ya expuestos en la Sección 3.2, los cuales poseen las mismas funcionalidades pero desarrollados con distintas herramientas, el primero utilizando el IDE Android Studio para desarrollar con el SDK nativo de android (desde ahora Droid) y el segundo prototipo utilizando el IDE VisualStudio Community en un proyecto Xamarin.Forms (desde ahora Xamarin). Ambos prototipos fueron modificados para realizar la siguiente evaluación que consta de 1000 muestras y su almacenamiento en la memoria interna para la cantidad de motores y flexores del hardware disponible. Las pruebas fueron realizadas en el dispositivo Samsung Galaxy S5 mini y Nexus 5 ya señalados en el Capítulo 1.

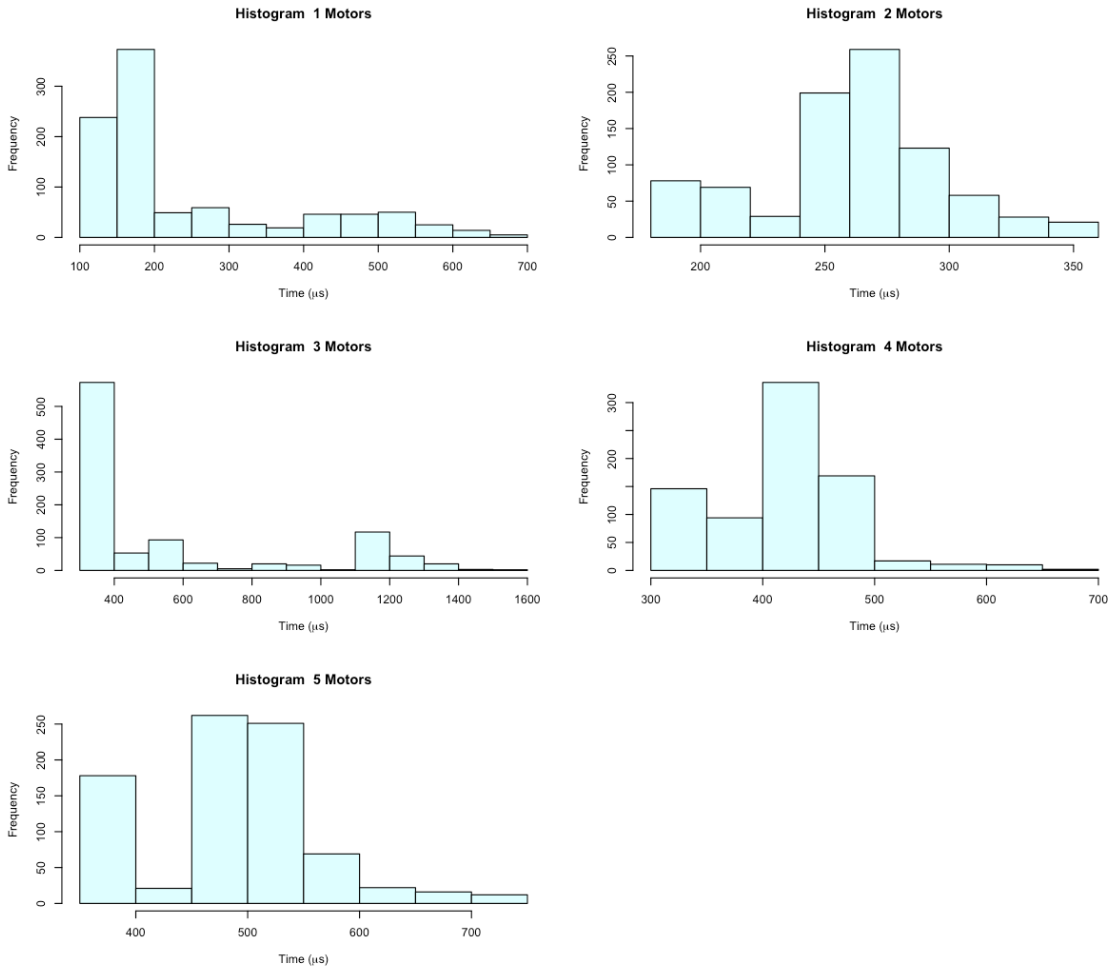

Figura 5.1: Histogramas de motores Droid-Galaxy
Fuente: elaboración propia (2018)

5.1.1 Prototipo 3 : Droid - Galaxy

5.1.1.1 Motores

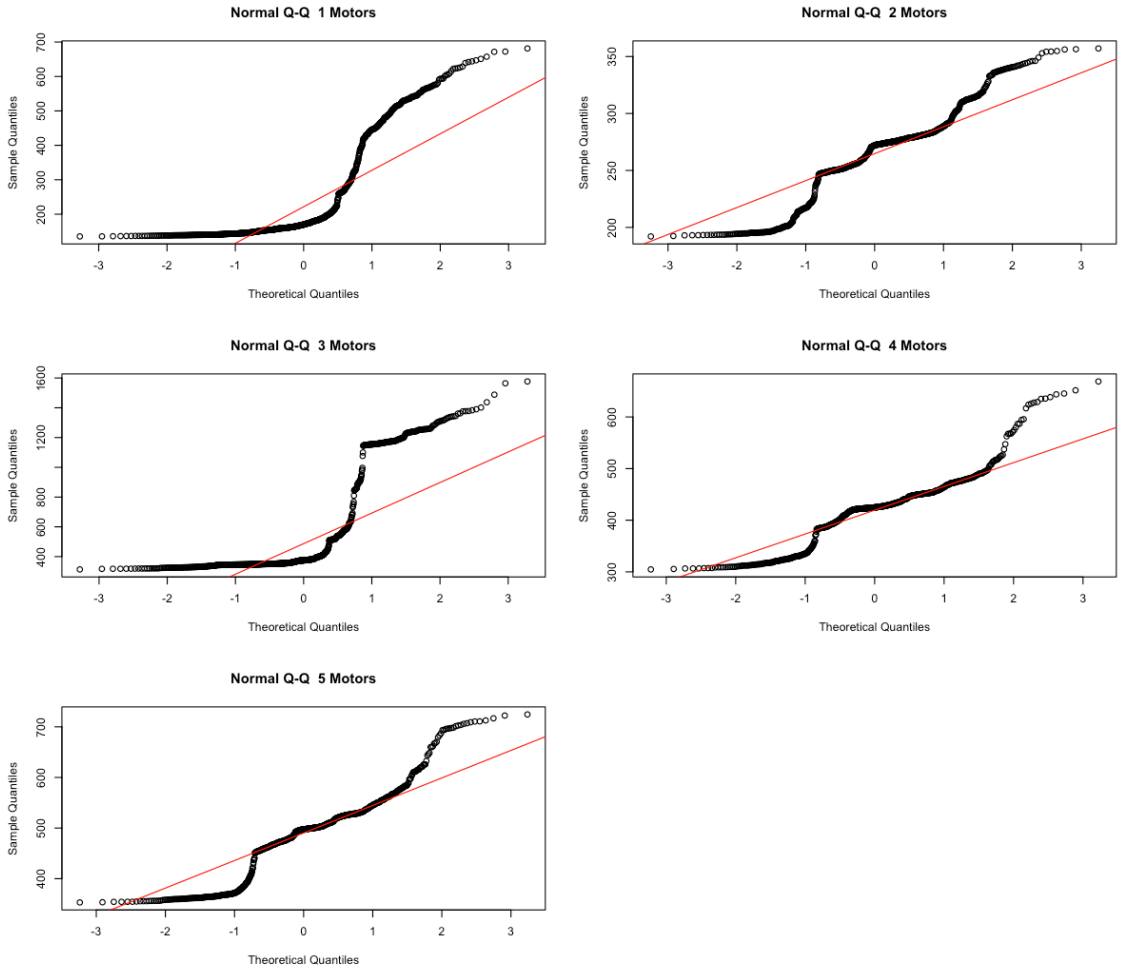

Figura 5.2: Gráficos QQ de motores Droid-Galaxy
Fuente: elaboración propia (2018)

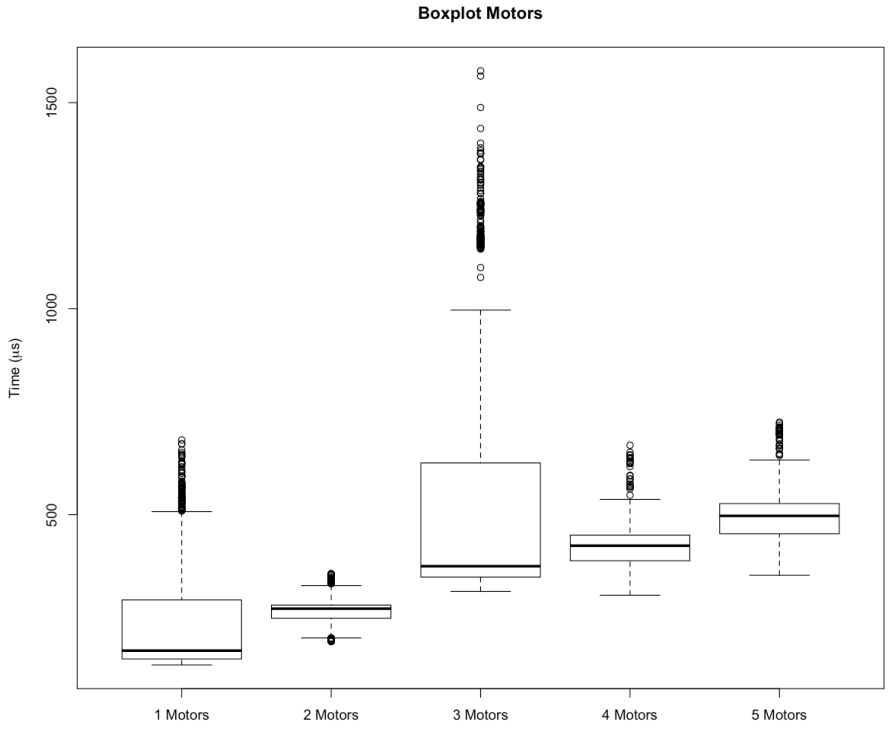

Figura 5.3: Gráficos de cajas de motores Droid-Galaxy
Fuente: elaboración propia (2018)

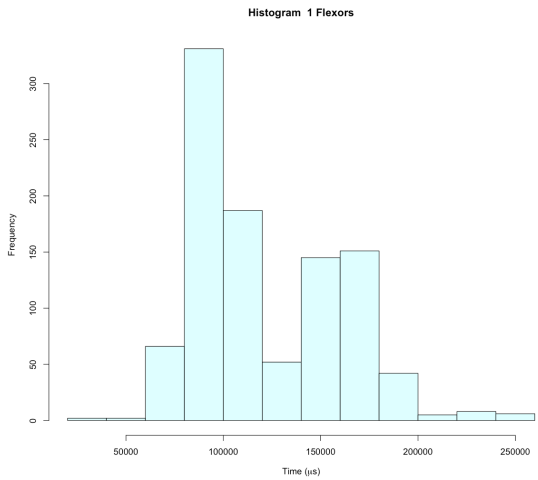

Figura 5.4: Histogramas de flexores Droid-Galaxy
Fuente: elaboración propia (2018)

5.1.1.2 Flexores

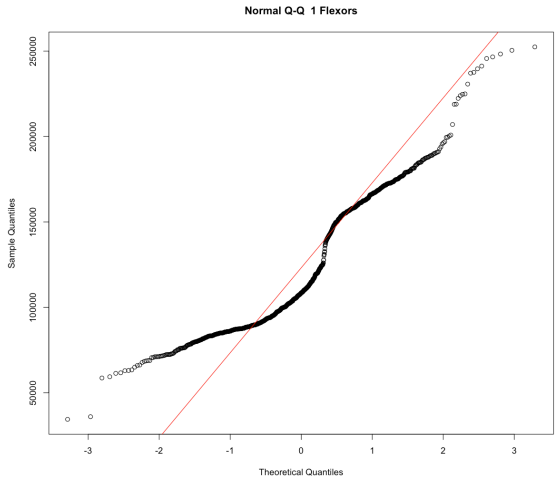

Figura 5.5: Gráficos QQ de flexores Droid-Galaxy
Fuente: elaboración propia (2018)

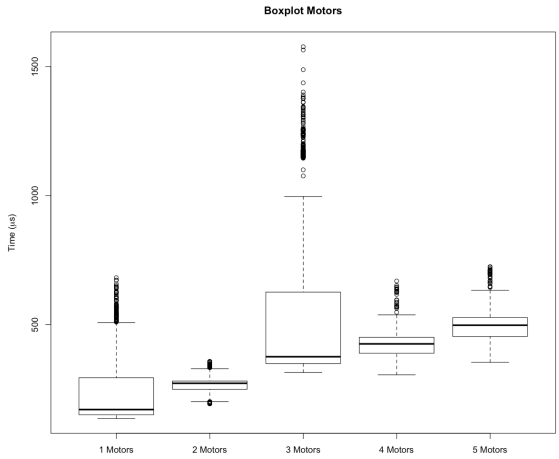

Figura 5.6: Gráficos de cajas de flexores Droid-Galaxy
Fuente: elaboración propia (2018)

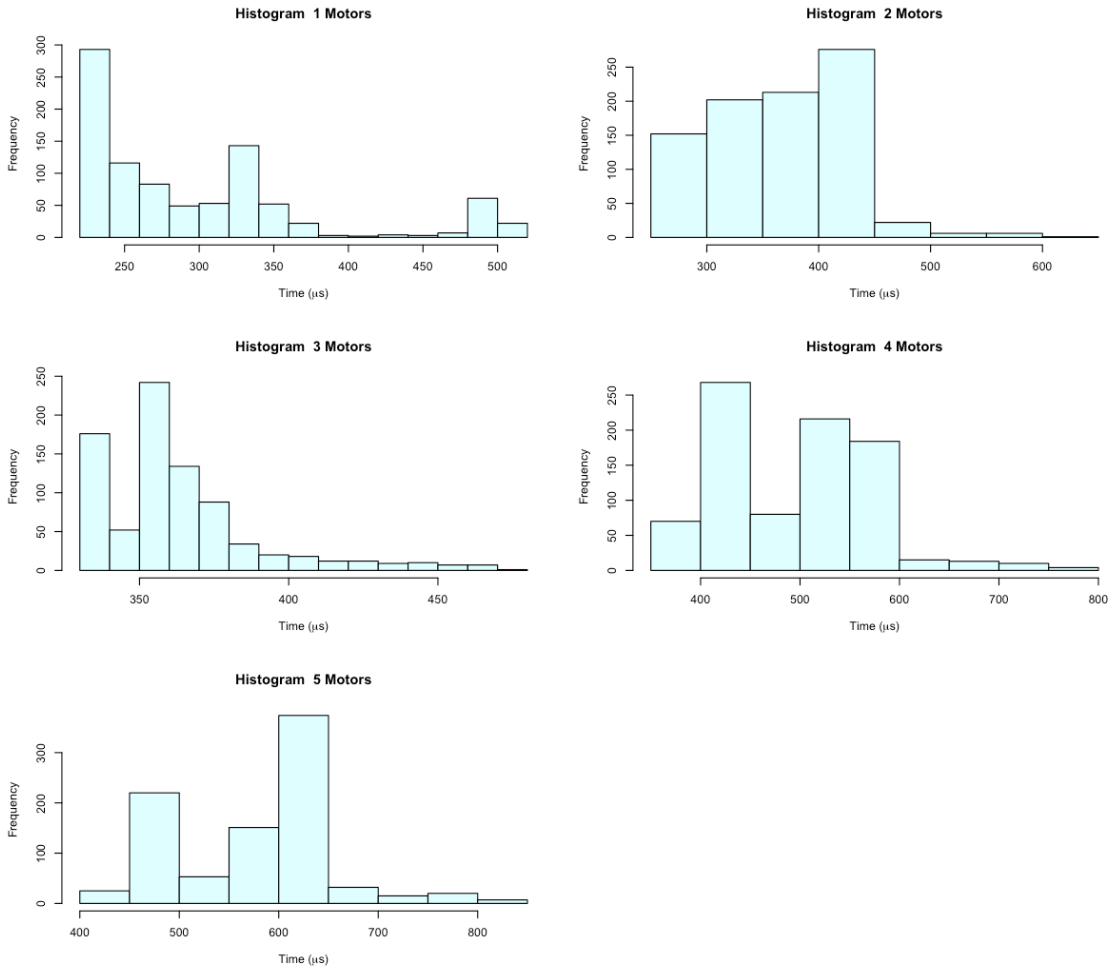

Figura 5.7: Histogramas de motores Xamarin-Galaxy
Fuente: elaboración propia (2018)

5.1.2 Prototipo 4: Xamarin - Galaxy

5.1.2.1 Motores

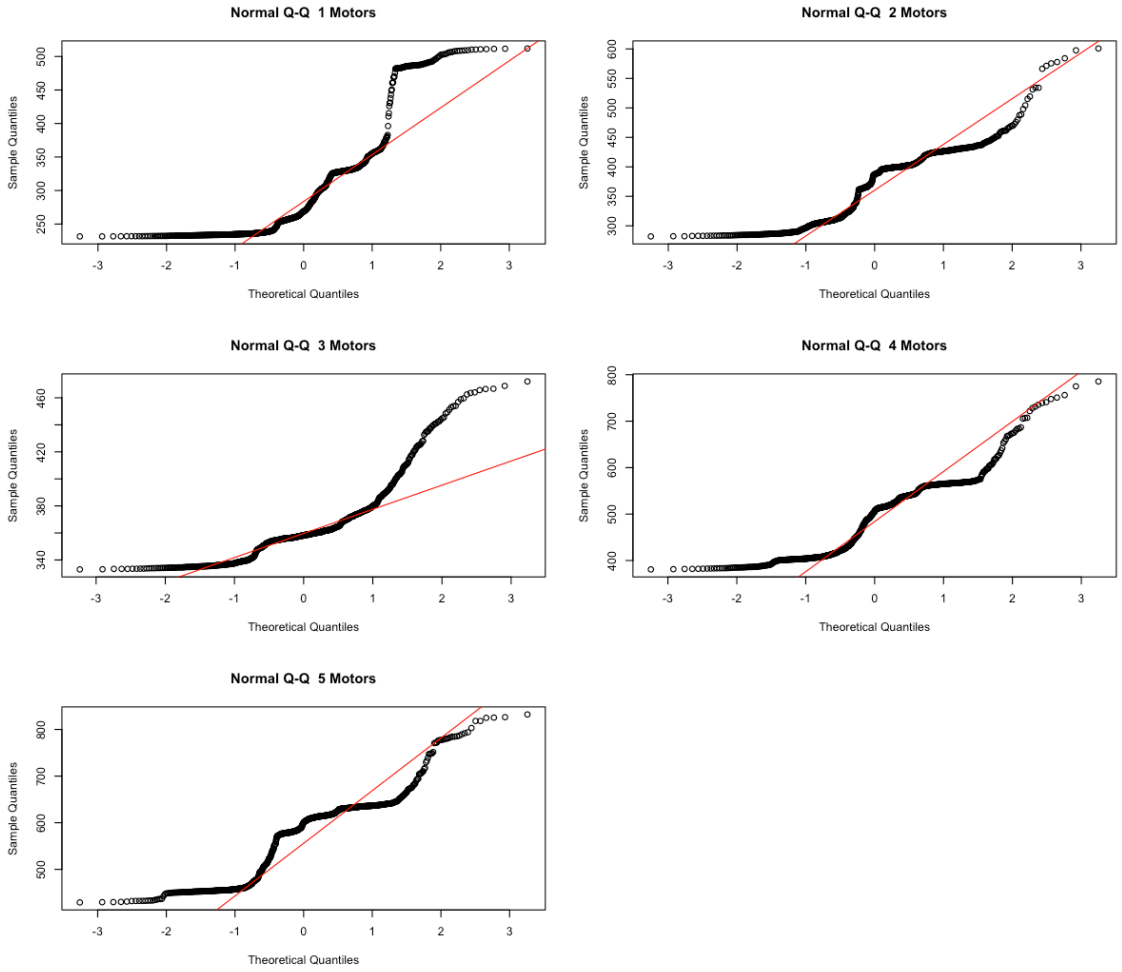

Figura 5.8: Gráficos QQ de motores Xamarin-Galaxy
Fuente: elaboración propia (2018)

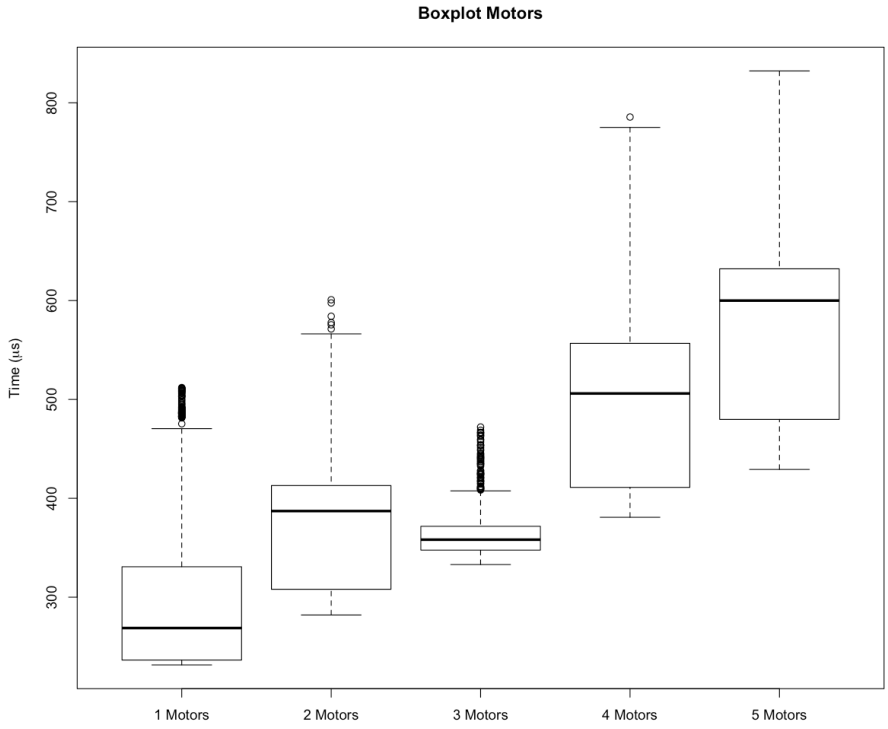

Figura 5.9: Gráficos de cajas de motores Xamarin-Galaxy
Fuente: elaboración propia (2018)

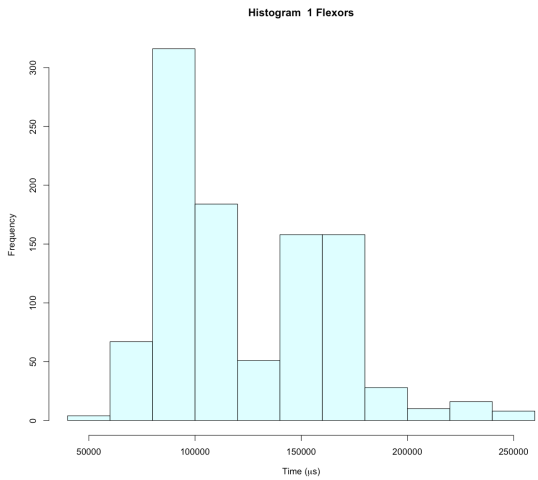

Figura 5.10: Histogramas de flexores Xamarin-Galaxy
Fuente: elaboración propia (2018)

5.1.2.2 Flexores

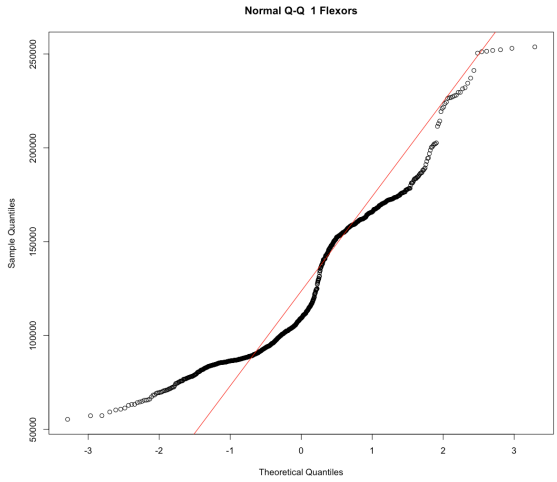

Figura 5.11: Gráficos QQ de flexores Xamarin-Galaxy
Fuente: elaboración propia (2018)

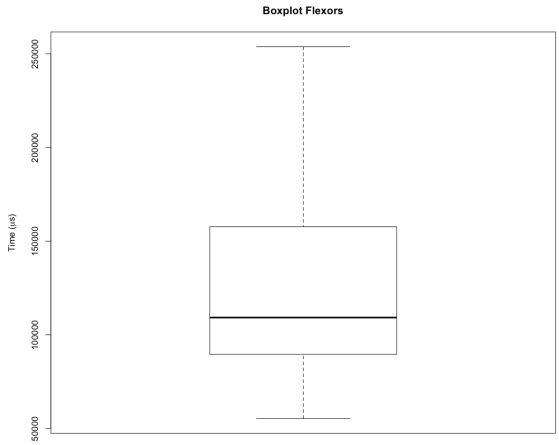

Figura 5.12: Gráficos de cajas de flexores Xamarin-Galaxy
Fuente: elaboración propia (2018)

5.1.3 Prototipo 3 : Droid - Nexus

5.1.3.1 Motores

5.1.3.2 Flexores

5.1.4 Prototipo 4: Xamarin - Nexus

5.1.4.1 Motores

5.1.4.2 Flexores

5.2 EVALUACIÓN TIEMPO DE ACTIVACIÓN

5.2.1 API C#

5.2.2 API Java

5.3 EVALUACIÓN TIEMPO DE LECTURA DE DATOS

5.3.1 API C#

5.3.2 API Java

5.4 RESUMEN

CAPÍTULO 6. CONCLUSIONES

6.1 OBJETIVOS

6.1.1 Objetivos específicos

6.1.2 Objetivo general

6.2 RESULTADOS OBTENIDOS

6.2.1 Desarrollo de software

6.2.2 Resultados de las pruebas

6.2.2.1 Tiempo de respuesta

6.2.2.2 Líneas de código

6.3 ALCANCES Y LIMITACIONES

6.4 TRABAJO FUTURO

6.5 OBSERVACIONES FINALES

GLOSARIO

- **Actuadores:** Un actuador es un dispositivo inherentemente mecánico cuya función es proporcionar fuerza para mover o "actuar" otro dispositivo mecánico. La fuerza que provoca el actuador proviene de tres fuentes posibles: Presión neumática, presión hidráulica, y fuerza motriz eléctrica (motor eléctrico o solenoide). Dependiendo de el origen de la fuerza el actuador se denomina "neumático", "hidráulico" o "eléctrico". (Aie, 2017)
- **API:** *Application Program Interface* por sus siglas en inglés es código que actúa como interfaz para la programación de aplicaciones, permitiendo por ejemplo, que dos aplicaciones se comuniquen entre si, como el acceder a funcionalidades sin la necesidad de conocer la complejidad del código implementado(Rouse, 2017).
- **Augmented Reality (AR):** La realidad aumentada (AR) es el uso de información en tiempo real en forma de texto, gráficos, audio y otras mejoras virtuales integradas con objetos del mundo real. Es este elemento del "mundo real" lo que diferencia a AR de la realidad virtual. AR integra y agrega valor a la interacción del usuario con el mundo real, frente a una simulación. (Gartner, 2017a).
- **Haptic Feedback:** Haptics es una tecnología táctil o de retroalimentación de fuerza que aprovecha el sentido del tacto de una persona al aplicar vibraciones y / o movimiento a la punta del dedo del usuario. Esta estimulación puede ayudar a la tecnología en el desarrollo de objetos virtuales en la pantalla del dispositivo. En su sentido más amplio, hápticos puede ser cualquier sistema que incorpore elementos táctiles y vibre a través de un sentido del tacto (Gartner, 2017b).
- **IMU (Inertial Measurement Unit):** Los sensores inerciales, también llamados IMU (Unidad de medición inercial), son dispositivos electrónicos de medición que permiten estimar la orientación de un cuerpo de las fuerzas inerciales que el cuerpo experimenta. Su principio de funcionamiento se basa en la medición de las fuerzas de aceleración y velocidad angular ejercidas independientemente en masas pequeñas ubicadas en el interior (Technaid, 2018).
- **OpenGlove:** es un guante desarrollado por la Universidad de Santiago de Chile, por el grupo de investigación y desarrollo Interaction, el cual provee *haptic feedback* o retroalimentación táctil en ambientes virtuales, como también la captura de movimientos de la mano (InTeracTion, 2018).
- **Mixed reality (MR):** La realidad mixta es el resultado de mezclar el mundo físico con el mundo digital. La realidad mixta es la siguiente evolución en la interacción entre el hombre, la computadora y el entorno, y abre posibilidades que antes estaban restringidas a nuestra imaginación. Es posible gracias a los avances en visión artificial, potencia de procesamiento gráfico, tecnología de visualización y sistemas de entrada. El término realidad mixta fue presentado originalmente en un artículo de 1994 por Paul Milgram y Fumio Kishino, "Una taxonomía de visualizaciones de realidad mixta". Su trabajo introdujo el concepto del continuum de virtualidad y se centró en cómo se aplica la categorización de la taxonomía a las exhibiciones. Desde entonces, la aplicación de la realidad mixta va más allá de las pantallas, pero también incluye la información ambiental, el sonido espacial y la ubicación. (Microsoft, 2017).
- **SDK:** conjunto de utilidades de desarrollo para escribir aplicaciones de software, generalmente asociadas a entornos específicos (por ejemplo, el SDK de Windows) (Gartner, 2017c).
- **UX:** La "experiencia de usuario" abarca todos los aspectos de la interacción del usuario final con la empresa, sus servicios y sus productos (Norman & Nielsen, 2017).
- **Virtual Reality (VR):** La realidad virtual (VR) proporciona un entorno 3D generado por computadora que rodea al usuario y responde a las acciones de esa persona de forma

natural, generalmente a través de pantallas inmersivas montadas en la cabeza y el seguimiento de la cabeza. También se pueden usar guantes que proporcionen seguimiento de las manos y retroalimentación háptica (sensible al tacto). Los sistemas basados en sala brindan una experiencia 3D para múltiples participantes; sin embargo, son más limitados en sus capacidades de interacción. (Gartner, 2017d).

REFERENCIAS BIBLIOGRÁFICAS

Aie (2017). Actuadores. Recuperado de http://www.aie.cl/files/file/comites/ca/abc/actuadores.pdf Revisado el 23 de Octubre de 2017.

Cerda, R. A. (2017). Extensión de openglove a nivel de hardware y software para la captura del movimiento de la mano.

DextaRobotics (2018). Dexmo. Recuperado de http://www.dextarobotics.com/ el 10 de Marzo de 2018.

Gartner (2017a). It glossary: Augmented reality (ar). Recuperado de https://www.gartner.com/it-glossary/augmented-reality-ar/ Revisado el 19 de Octubre de 2017.

Gartner (2017b). It glossary: Haptics. Recuperado de https://www.gartner.com/it-glossary/haptics Revisado el 19 de Octubre de 2017.

Gartner (2017c). It glossary: Sdk. Recuperado de https://www.gartner.com/it-glossary/?s=SDK Revisado el 19 de Octubre de 2017.

Gartner (2017d). It glossary: Virtual reality (vr). Recuperado de https://www.gartner.com/it-glossary/vr-virtual-reality/ Revisado el 19 de Octubre de 2017.

InTeracTion (2018). Openglove haptics easy to develop. Recuperado de http://www.openglove.org/ Revisado el 10 de Marzo de 2018.

ManusVR (2018). Manus vr. Recuperado de http://www.dextarobotics.com/ el 10 de Marzo de 2018.

Martin, J. (1991). *Rapid Application Development*. Indianapolis, IN, USA: Macmillan Publishing Co., Inc.

Meneses, S. A. (2016). Openglove sdk: Apis de alto nivel para c#, c++ java y javascript.

Microsoft (2017). Mixed reality. https://developer.microsoft.com/en-us/windows/mixed-reality/mixed_reality Revisado el 26 de Noviembre de 2017.

Monsalve, R. A. (2016). Dispositivo de retroalimentación táctil para interfaces naturaltes y de ralidad virtual.

Neurodigital (2018). Avatarvr. Recuperado de https://www.neurodigital.es/avatarvr/ el 10 de Marzo de 2018.

Norman, D., & Nielsen, J. (2017). The definition of user experience (ux). Recuperado de https://www.nngroup.com/articles/definition-user-experience/ Revisado el 21 de Octubre de 2017.

Rambal (2018). Sensor flex. Recuperado de http://rambal.com/presion-peso-nivel-liquido/250-sensor-flex.html Revisado el 23 de Abril de 2018.

Rouse, M. (2017). application program interface (api). Recuperado de http://searchmicroservices.techtarget.com/definition/application-program-interface-API Revisado el 21 de Octubre de 2017.

Statista (2016). Virtual reality (vr) - statistics and facts. Recuperado de https://www.statista.com/topics/2532/virtual-reality-vr/ Revisado el 20 de Octubre de 2017.

Technaid (2018). Sensor de rastreo imu. Recuperado de http://www.technaid.com/support/research/imu-working-principles/ Revisado el 22 de Abril de 2018.

Techopedia (2017). Software development kit (sdk). Recuperado de https://www.techopedia.com/definition/3878/software-development-kit-sdk Revisado el 19 de Octubre de 2017.

Ticportal (2018). Open source (código abierto). Recuperado de https://www.ticportal.es/glosario-tic/open-source-codigo-abierto Revisado el 28 de Abril de 2018.