

**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

© Israel Gedeón Elías Matínez Montenegro , 2018



• Algunos derechos reservados. Esta obra está bajo una Licencia Creative Commons Atribución-Chile 3.0. Sus condiciones de uso pueden ser revisadas en:  
<http://creativecommons.org/licenses/by/3.0/cl/>.

## 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 posee 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

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Antecedentes y motivación	1
1.2	Descripción del problema	2
1.3	Solución Propuesta	3
1.3.1	Características de la solución	3
1.3.2	Propósitos de la solución	4
1.4	Objetivos y alcance del proyecto	4
1.4.1	Objetivo general	4
1.4.2	Objetivos específicos	4
1.5	Metodologías y herramientas utilizadas	5
1.5.1	Metodología a usar	5
1.5.2	Herramientas de Software	6
1.5.3	Herramientas de Hardware	7
1.6	Organización del documento	8
<b>2</b>	<b>Marco teórico</b>	<b>10</b>
2.1	Marco conceptual	10
2.1.1	API	10
2.1.2	SDK	10
2.1.3	Tipos de aplicaciones móviles	11
2.1.3.1	Nativas	11
2.1.3.2	Web	12
2.1.3.3	Híbridas	12
2.1.4	WebSocket	13
2.2	Estado del arte	13
2.2.1	OpenGlove	14
2.2.2	AvatarVR	15
2.2.3	Dexmo	15
2.2.4	Manus VR	16
2.2.5	Haptx	17
2.3	Resumen	17
<b>3</b>	<b>Análisis</b>	<b>19</b>
3.1	Levantamiento de requisitos de software	19
3.1.1	Antecedentes	19
3.1.2	Requisitos	19
3.2	Prototipos	25
3.2.1	Primer prototipo: Activación de motores	25
3.2.2	Segundo prototipo: Obtención de datos desde flexores	27
3.2.3	Tercer prototipo	28
3.2.4	Cuarto prototipo	30
3.2.5	Quinto prototipo	31
3.2.6	Sexto prototipo	31
3.3	APIs	32
3.3.1	Cuarto prototipo	32
3.3.2	Quinto prototipo	33
3.3.3	Sexto prototipo	33
3.4	Resumen	34

<b>4</b>	<b>Diseño e implementación</b>	<b>35</b>
4.1	Arquitectura general . . . . .	35
4.2	Estructura . . . . .	36
4.2.1	Servicio . . . . .	36
4.2.2	APIs . . . . .	37
4.2.3	Software de configuración . . . . .	37
4.3	Comportamiento . . . . .	37
4.3.1	Obtener guantes . . . . .	38
4.3.2	Activación . . . . .	39
4.3.3	Añadir flexor a una región . . . . .	40
4.3.4	Asignar Threshold . . . . .	41
4.3.5	Iniciar IMU . . . . .	42
4.3.6	Lectura de datos provenientes de Arduino . . . . .	43
4.4	Aspectos de implementación . . . . .	44
4.4.1	Desarrollo multiplataforma en Xamarin.Forms . . . . .	44
4.4.2	Servicio . . . . .	44
4.4.3	Software de configuración . . . . .	44
4.4.4	APIs . . . . .	44
4.4.4.1	Java . . . . .	44
4.4.4.2	C# . . . . .	44
4.5	Resumen . . . . .	44
<b>5</b>	<b>Evaluación técnica</b>	<b>45</b>
5.1	Evaluación aplicaciones nativa y multiplataforma . . . . .	45
5.1.1	Prototipo 3 : Droid - Galaxy . . . . .	45
5.1.1.1	Motores . . . . .	45
5.1.1.2	Flexores . . . . .	48
5.1.2	Prototipo 4: Xamarin - Galaxy . . . . .	50
5.1.2.1	Motores . . . . .	50
5.1.2.2	Flexores . . . . .	53
5.1.3	Prototipo 3 : Droid - Nexus . . . . .	56
5.1.3.1	Motores . . . . .	56
5.1.3.2	Flexores . . . . .	56
5.1.4	Prototipo 4: Xamarin - Nexus . . . . .	56
5.1.4.1	Motores . . . . .	56
5.1.4.2	Flexores . . . . .	56
5.2	Evaluación tiempo de activación . . . . .	56
5.2.1	API C# . . . . .	56
5.2.2	API Java . . . . .	56
5.3	Evaluación tiempo de lectura de datos . . . . .	56
5.3.1	API C# . . . . .	56
5.3.2	API Java . . . . .	56
5.4	Resumen . . . . .	56
<b>6</b>	<b>Conclusiones</b>	<b>57</b>
6.1	Objetivos . . . . .	57
6.1.1	Objetivos específicos . . . . .	57
6.1.2	Objetivo general . . . . .	57
6.2	Resultados obtenidos . . . . .	57
6.2.1	Desarrollo de software . . . . .	57
6.2.2	Resultados de las pruebas . . . . .	57
6.2.2.1	Tiempo de respuesta . . . . .	57
6.2.2.2	Líneas de código . . . . .	57
6.3	Alcances y limitaciones . . . . .	57

6.4 Trabajo futuro . . . . .	57
6.5 Observaciones finales . . . . .	57
<b>Glosario</b>	<b>58</b>
<b>Referencias bibliográficas</b>	<b>60</b>



## ÍNDICE DE TABLAS

Tabla 2.1	Comparación SDK de distintos Guantes . . . . .	14
Tabla 3.1	Requisitos funcionales de software . . . . .	20
Tabla 3.2	Requisitos no funcionales de software . . . . .	25
Tabla 3.3	Primer prototipo: activación de motores . . . . .	26
Tabla 3.4	Segundo prototipo: obtención de datos desde flexores . . . . .	27
Tabla 3.5	Tercer prototipo: activación de motores y obtención de datos desde flexores . . . . .	29
Tabla 3.6	Matriz de prototipos de software vs requisitos funcionales . . . . .	34
Tabla 3.7	Matriz de prototipos de software vs requisitos no funcionales . . . . .	34
Tabla 5.1	Resumen resultado pruebas motor Droid-Galaxy . . . . .	45
Tabla 5.2	Resumen resultado pruebas flexor Droid-Galaxy . . . . .	48
Tabla 5.3	Resumen resultado pruebas motor Xamarin-Galaxy . . . . .	50
Tabla 5.4	Resumen resultado pruebas flexor Xamarin-Galaxy . . . . .	53

# ÍNDICE DE ILUSTRACIONES

Figura 1.1	Arquitectura OpenGlove . . . . .	2
Figura 2.1	Guantes OpenGlove . . . . .	14
Figura 2.2	Guantes AvatarVR y TrackBand . . . . .	15
Figura 2.3	Guantes Dexmo . . . . .	16
Figura 2.4	Guantes Manus VR . . . . .	16
Figura 2.5	Guantes Haptx . . . . .	17
Figura 3.1	Primer prototipo: activación de motores . . . . .	26
Figura 3.2	Segundo prototipo: obtención de datos desde flexores . . . . .	28
Figura 3.3	Tercer prototipo: activación de motores y obtención de datos desde flexores . . . . .	29
Figura 3.4	Cuarto prototipo: app navigation, devices, server and board configuration . . . . .	30
Figura 3.5	Quinto prototipo: actuator mapping and actuator test . . . . .	31
Figura 3.6	Sexto prototipo: flexor mapping, flexor test and IMU configuration . . . . .	32
Figura 4.1	Arquitectura de Openglove . . . . .	35
Figura 4.2	Detalle Arquitectura de Openglove . . . . .	36
Figura 4.3	Obtener guantes . . . . .	38
Figura 4.4	Activar región . . . . .	39
Figura 4.5	Agregar flexor a una región . . . . .	40
Figura 4.6	Asignar Threshold . . . . .	41
Figura 4.7	Iniciar IMU . . . . .	42
Figura 4.8	Lectura de datos provenientes de Arduino . . . . .	43
Figura 5.1	Histogramas de motores Droid-Galaxy . . . . .	46
Figura 5.2	Gráfico QQ de motores Droid-Galaxy . . . . .	47
Figura 5.3	Gráficos de cajas de motores Droid-Galaxy . . . . .	48
Figura 5.4	Histogramas de flexores Droid-Galaxy . . . . .	49
Figura 5.5	Gráfico QQ de flexores Droid-Galaxy . . . . .	49
Figura 5.6	Gráficos de cajas de flexores Droid-Galaxy . . . . .	50
Figura 5.7	Histogramas de motores Xamarin-Galaxy . . . . .	51
Figura 5.8	Gráfico QQ de motores Xamarin-Galaxy . . . . .	52
Figura 5.9	Gráficos de cajas de motores Xamarin-Galaxy . . . . .	53
Figura 5.10	Histogramas de flexores Xamarin-Galaxy . . . . .	54
Figura 5.11	Gráfico QQ de flexores Xamarin-Galaxy . . . . .	54
Figura 5.12	Gráficos de cajas de flexores Xamarin-Galaxy . . . . .	55

# CAPÍTULO 1. INTRODUCCIÓN

## 1.1 ANTECEDENTES Y MOTIVACIÓN

El tamaño del mercado de la realidad virtual (VR)<sup>1</sup> y realidad aumentada (AR)<sup>2</sup> 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)<sup>3</sup> 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 presentar una brecha cuando cuando se desea incluir retroalimentación vibrotáctil o el denominado *haptic feedback*<sup>4</sup> 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.

---

<sup>1</sup>**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).

<sup>2</sup>**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)

<sup>3</sup>**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).

<sup>4</sup>**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 4.1. El guante posee motores distribuidos en distintos lugares de la mano, los cuales pueden ser activados y desactivados según se requiera mediante APIs (Monsalve, 2015), también se tienen sensores de flexibilidad y la captura de movimientos de la mano (Cerdeña, 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<sup>5</sup>, ni iOS <sup>6</sup> 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?.

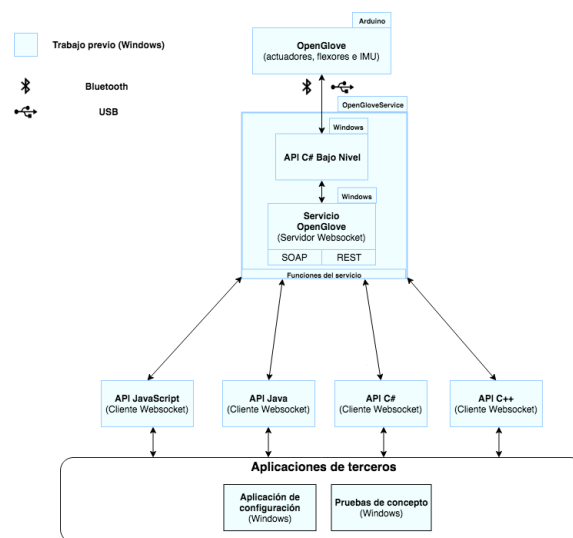


Figura 1.1: Arquitectura OpenGlove  
Fuente: Elaboración propia (2018)

<sup>5</sup>Comunidad Android: <https://developers.google.com/vr/android/>

<sup>6</sup>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<sup>7</sup> 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 <sup>8</sup>, que permiten mantener un servicio que pueda seguir funcionando mientras otras aplicaciones se estén utilizando en el dispositivo.
- Websocket <sup>9</sup>, 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<sup>10</sup>, tecnología de red inalámbrica que permite la transmisión de datos entre dispositivos.

---

<sup>7</sup>SDK: conjunto de utilidades de desarrollo para escribir aplicaciones de software, generalmente asociadas a entornos específicos (por ejemplo, el SDK de Windows) (Gartner, 2017c).

<sup>8</sup>Background service Android: <https://developer.android.com/training/run-background-service/create-service.html>

<sup>9</sup>Websocket: <http://websocket.org/aboutwebsocket.html>

<sup>10</sup>Bluetooth Android: <https://developer.android.com/guide/topics/connectivity/bluetooth.html>

- Xamarin.Forms<sup>11</sup> tecnología que permite el desarrollo de aplicaciones multiplataforma para 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.

### **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 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.

---

<sup>11</sup>Xamarin.Forms: <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/>

También permitirá la activación y control de actuadores <sup>12</sup>, flexores <sup>13</sup> e IMU (*Inertial Measurement Unit*).

3. Realizar evaluaciones de rendimiento para las APIs Java y C#.
4. Demostrar el uso del SDK en un ambiente de VR, AR o MR, utilizando OpenGlove y el visor de realidad virtual ZapBox<sup>14</sup>.

## 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.

### 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 deseable desarrollado. El prototipo final aprobado será la base con lo que se desarrollará la siguiente etapa. La segunda etapa consiste en el desarrollo

<sup>12</sup>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)

<sup>13</sup> "... es un sensor de flexión que produce una resistencia variable en función del grado al que este doblada".(Rambal, 2018)

<sup>14</sup>Visor Zapbox: <https://www.zappar.com/zapbox/>

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 siguientes 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 <sup>15</sup>.

### 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 <sup>16</sup> en conjunto con el ecosistema de plataformas para compilar <sup>17</sup> aplicaciones móviles

<sup>15</sup>Manifiesto ágil: <http://agilemanifesto.org/>

<sup>16</sup>Xamarin más VS Community para Mac 2018: <https://store.xamarin.com/>

<sup>17</sup> 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.ictca.com/cs/knowledgebase.php?action=displayarticle&id=8817>



Xamarin.Forms, el cual utiliza el lenguaje de programación C# para el desarrollo de aplicaciones nativas multiplataforma. Adicionalmente se utilizó el IDE Android Studio <sup>18</sup>, 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<sup>19</sup> para el control de versiones.
2. Texmaker <sup>20</sup>, para la escritura de la memoria.
3. RStudio<sup>21</sup>: para el análisis de datos de la evaluación del proyecto.
4. Google Drive<sup>22</sup>, para generar y compartir documentos y archivos de manera colaborativa.
5. Microsoft Project Professional y Ganttter, para el desarrollo de la carta gantt del proyecto de título.

### 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* <sup>23</sup> con las siguientes características:

1. Sistema Operativo Android Marshmallow 6.0.1.
2. CPU: 1.4Ghz Quad-Core ARM Cortex-A7.

<sup>18</sup>Android Studio: <https://developer.android.com/studio/index.html?hl=es-419>

<sup>19</sup>Github: <https://github.com/>

<sup>20</sup>Texmaker : <http://www.xm1math.net/texmaker/>

<sup>21</sup>Rstudio: <https://www.rstudio.com/>

<sup>22</sup>Google Drive: [https://www.google.com/intl/es\\_ALL/drive/](https://www.google.com/intl/es_ALL/drive/)

<sup>23</sup> <http://www.movilcelular.es/samsung-galaxy-s5-mini-duos-sm-g800h/caracteristicas/1659>

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.

## **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 comprensión 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. Primero se presenta el Marco conceptual, en el cual se establecen los conceptos relevantes del proyecto, luego en el Estado del arte se realiza una revisión del mismo.

### 2.1 MARCO CONCEPTUAL

#### 2.1.1 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). Por ejemplo, para el sistema operativo <sup>1</sup> Android, todo su conjunto de funciones está disponible mediante APIs escritas en el lenguaje Java. Estas APIs son los cimientos necesarios para crear aplicaciones de Android simplificando la reutilización de componentes del sistema, servicios centrales y modulares.

#### 2.1.2 SDK

Es un conjunto de utilidades de desarrollo para escribir aplicaciones de software, generalmente asociadas a entornos específicos (Gartner, 2017c). Ejemplos de SDKs incluyen a Windows 7 SDK, the Mac OS X SDK, iPhone SDK y Android SDK.

Los SDK suelen incluir un entorno de desarrollo integrado (IDE), que sirve como interfaz de programación central. El IDE puede incluir una ventana de programación para escribir el código fuente, un depurador para corregir errores del programa y un editor visual, que permite a los desarrolladores crear y editar la interfaz gráfica de usuario (GUI) del programa. Los IDE también incluyen un compilador, que se usa para crear aplicaciones a partir de archivos de código fuente (Techterms, 2010).

---

<sup>1</sup>En resumidas palabras, un sistema operativo (OS por sus siglas en inglés) es un software que, después de ser cargado en la computadora por un programa de arranque inicial, administra los recursos de una computadora, controlando el flujo de información hacia y desde un procesador principal, administrando recursos como, la memoria, redes, archivos, etc. (Gartner, 2018)

La mayoría de los SDK contienen código de ejemplo, que proporciona a los desarrolladores ejemplos de programas y bibliotecas. Estas muestras ayudan a los desarrolladores a aprender cómo crear programas básicos con el SDK, lo que les permite eventualmente crear aplicaciones más complejas. Los SDK también ofrecen documentación técnica, que puede incluir tutoriales y preguntas frecuentes. Algunos SDK también pueden incluir gráficos de muestra, como botones e iconos, que se pueden incorporar a las aplicaciones (Techterms, 2010). En resumen, un SDK usualmente incluye lo siguiente:

- APIs
- IDE
- Software de configuración
- Códigos de ejemplo (programas, pruebas de concepto y bibliotecas)
- Documentación (técnica, tutoriales, preguntas frecuentes)
- Materiales gráficos de muestra (iconos, botones, etc.)

### **2.1.3 Tipos de aplicaciones móviles**

Actualmente los sistemas operativos que predominan el mercado de dispositivos móviles son dos, iOS con un 14,4% y Android con un 84,8% (Statista, 2017). Cuando se habla de desarrollar aplicaciones móviles se tienen como objetivo los dispositivos que tienen iOS o Android como sistema operativo. A continuación se detallan los tres principales tipos de aplicaciones móviles: Nativas, Híbridas y Web.

#### **2.1.3.1 Nativas**

Apple y Google ofrecen a los desarrolladores de aplicaciones sus propias herramientas de desarrollo, elementos de interfaz y SDK estandarizado; Xcode y Android Studio. Esto permite a cualquier desarrollador profesional desarrollar una aplicación nativa con relativa facilidad.

Las ventajas que ofrecen las aplicaciones móviles nativas, son mayor rapidez que las aplicaciones Híbridas y Web, también una experiencia con elementos de interfaz nativos. Con aplicaciones nativas, es posible acceder a la cámara, el micrófono, almacenamiento interno, Bluetooth, la

brújula, el acelerómetro y deslizar gestos fácilmente. Todavía es posible usar las alternativas, pero es más fácil de manera nativa (MobiLoud, 2018).

Las desventajas de las aplicaciones nativas es que se requiere más de un código base, dado que las aplicaciones iOS no se ejecutarán en Android y viceversa, por lo que se tiene que trabajar con diferentes códigos para cada plataforma que se elija construir. Esto se traduce a mayores costos de mantención, puesto que se necesitan desarrolladores con conocimientos en ambas plataformas o bien equipos diferenciados, teniendo ambos que desarrollar lo mismo en ambas aplicaciones nativas (MobiLoud, 2018).

Actualmente existen alternativas a las herramientas para desarrollo nativo ofrecidas por iOS y Android. Estas son, Xamarin, React Native y Titanium, las cuales permiten desarrollar aplicaciones nativas multiplataforma (cross-platform) para iOS y Android. Gracias a estas alternativas se permite compartir código para ambas plataformas.

#### *2.1.3.2 Web*

La mayoría están desarrolladas en JavaScript, CSS y HTML5. No poseen SDK para trabajar con iOS y Android. En resumidas cuentas son aplicaciones web que son adaptadas para parecer aplicaciones nativas, pero sin acceder completamente a las funcionalidades del dispositivo y sin tener un comportamiento igual al nativo.

Hasta hace poco, las aplicaciones web carecían de la funcionalidad de las aplicaciones nativas, como la capacidad de enviar notificaciones automáticas, trabajar sin conexión y cargar en la pantalla de inicio.

Sin embargo, han habido algunas mejoras en los navegadores y aplicaciones web que ofrecen estas características. Las aplicaciones que aprovechan estas características se denominan aplicaciones web progresivas (MobiLoud, 2018). Esta alternativa permite compartir el 100% del código para ambas plataformas, ya que son aplicaciones web accedidas por Internet desarrolladas con herramientas web.

#### *2.1.3.3 Híbridas*

Se instala como una aplicación nativa, pero en realidad es una aplicación web en el interior. Las aplicaciones híbridas, como las aplicaciones web, se crean con Javascript, HTML y CSS y se ejecutan en algo llamado Webview, un navegador simplificado dentro de su aplicación. Las

ventajas que presentan estas aplicaciones es que son menos costosas ya que comparten el mismo código base. Al igual que con las aplicaciones nativas, conserva la misma capacidad para acceder a las funciones del dispositivo. Esto es gracias a soluciones como PhoneGap que actúan como un puente entre el SDK nativo y la vista web en la que se ejecuta la aplicación.

La principal desventaja de este tipo de aplicación móvil, es su rendimiento, además de que presentan diferencias en el comportamiento con el usuario de como lo son las aplicaciones nativas. alguna de las alternativas existentes para desarrollar aplicaciones híbridas son: PhoneGap/Cordova y Canvas.

#### **2.1.4 WebSocket**

(WebSocket.org, 2018a) (WebSocket.org, 2018b) (Grigorik, 2013)

## **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).

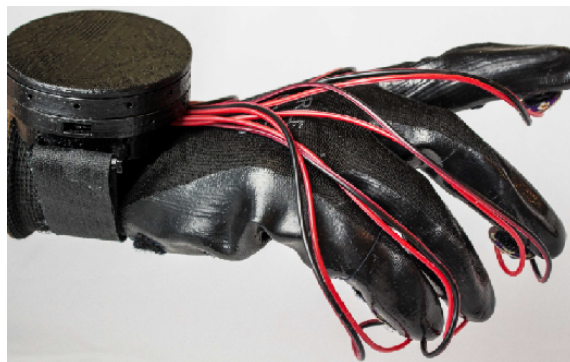


Figura 2.1: Guantes OpenGlove  
Fuente: InTeracTion (2018)



### 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. <sup>2</sup>

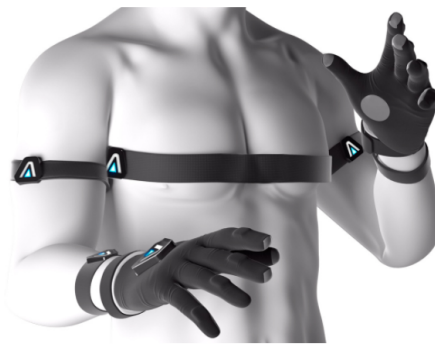


Figura 2.2: Guantes AvatarVR y TrackBand  
Fuente: Neurodigital (2018)

### 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.

---

<sup>2</sup>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 €<sup>3</sup>



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

---

<sup>3</sup>Precio de Manus VR y licencias <https://manus-vr.com/order.php>

### 2.2.5 Haptx

“HaptX presenta los primeros guantes hápticos de grado industrial del mundo. Los guantes HaptX brindan un toque realista, realimentación de fuerza y seguimiento de movimiento preciso” Haptx (2018). Anunciado el 20 de Noviembre del 2017, su principal diferencia con los demás guantes, es la tecnología textil inteligente que desarrollaron, es flexible y basado en silicona contiene una serie de actuadores neumáticos de alto desplazamiento y canales de aire microfluídicos integrados. Los actuadores proporcionan retroalimentación háptica empujando contra la piel del usuario, desplazándola de la misma manera que un objeto real cuando se toca. Una segunda capa opcional de microcanales puede agregar retroalimentación de temperatura entregando variaciones de agua fría y caliente. No se han anunciado precios, pues es una herramienta de grado industrial diseñada específicamente para empresas y organizaciones.



Figura 2.5: Guantes Haptx  
Fuente: Haptx (2018)

## 2.3 RESUMEN

OpenGlove es un proyecto Open Source<sup>4</sup> que permite la retroalimentación 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

<sup>4</sup> “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).

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.

## **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 relacionados a OpenGlove, realizados por Monsalve (2015), Meneses (2016) y Cerda (2017) se ha logrado establecer las bases de hardware y software para dar soporte a la retroalimentación vibrotáctil y el seguimiento de las manos. En el trabajo de Meneses (2016), se realizó una extensión de software a OpenGlove, desarrollando un SDK de alto nivel para la retroalimentación vibrotáctil. Por otra parte, el trabajo de Cerda (2017), se realizó una extensión de software y hardware para la captura de movimientos de la mano. En base a estos dos últimos se desprenden las diferentes funcionalidades y soporte que se debe lograr a nivel de software, para dar soporte a los actuadores, sensores de flexibilidad e IMU. Actualmente OpenGlove cuenta con un SDK para Windows, el cual incluye el software de configuración y cuatro APIs de alto nivel para los lenguajes C#, Java, C++ y JavaScript. El SDK fue desarrollado en el IDE Visual Studio.

#### **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, modificado de Meneses (2016)
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. Se debe diferenciar pines digitales y análogos de la placa Arduino	Inicio, modificado de Meneses (2016) y RF001 de Cerde (2017)
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, Meneses (2016)
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, Meneses (2016)

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. También debe permitir la posibilidad de agregar su propia imagen que represente el mapeo.	Inicio, modificado de Meneses (2016)
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, Meneses (2016)
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, Meneses (2016)
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, modificado de Meneses (2016)

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, modificado de Meneses (2016)
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, Meneses (2016)
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, modificado de Cerde (2017)
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, modificado de Cerde (2017)



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, Cerde (2017)
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, Cerde (2017)
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, Cerde (2017)
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, Cerde (2017)
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, Cerde (2017)
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, Cerde (2017)

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, Cerde (2017)
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, Cerde (2017)
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, Cerde (2017)
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, Cerde (2017)
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

La Tabla 3.2, muestra los requisitos no funcionales obtenidos mediante las reuniones con el profesor guía. De estas restricciones al SDK, se le debe tomar especial importancia al requisito no funcional RNF004, puesto este indica el umbral límite que no se debe superar. Experimentos enfocados en medir los tiempos de latencia que las personas pueden detectar son del orden de 54 ms para retroalimentación kinestésica (Batteau et al., 2004), 60 ms en estudios de percepción (Okamoto et al., 2009) y 60 ms simulaciones quirúrgicas simulaciones (Wu et al., 2015). Referencias de la literatura obtenidas del primer trabajo relacionado a OpenGlove realizado por Monsalve (2015).

Tabla 3.2: Requisitos no funcionales de software  
Fuente: Elaboración propia (2018)

ID	Síntesis del requisito	Descripción	Origen
RNF001	Sistema multiplataforma	El software a desarrollar debe ser ejecutado inicialmente en el sistema operativo Android, sin embargo, el proyecto en un futuro debe poder dar soporte al sistema operativo iOS.	Inicio
RNF002	Soportar comunicación Bluetooth	El sistema debe ser capaz de comunicarse con los dispositivos OpenGlove mediante Bluetooth Clásico.	Inicio
RNF003	Interoperabilidad	El SDK debe permitir la interoperabilidad entre los lenguajes de programación C# y Java.	Inicio
RNF004	Umbral latencia aceptada	El SDK no debe superar el umbral de 60 ms de latencia perceptible por el usuario.	Inicio

## 3.2 PROTOTIPOS

A continuación se muestran los diferentes prototipos desarrollados durante la escritura de esta memoria. Los prototipos serán detallados utilizando una tabla resumen del mismo, la que incluye, los Objetivos del prototipo, una Descripción, los requisitos funcionales y no funcionales que aborda el prototipo. Se incluyen las capturas del trabajo realizado, el 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 <sup>1</sup>, permitiendo en primera instancia la obtención de dispositivos vinculados y la conexión con alguno 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 3.3.

<sup>1</sup>Documentación oficial android: <https://developer.android.com/guide/topics/connectivity/bluetooth>

Tabla 3.3: 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. 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 001
Requisitos no funcionales	RNF002, RNF004

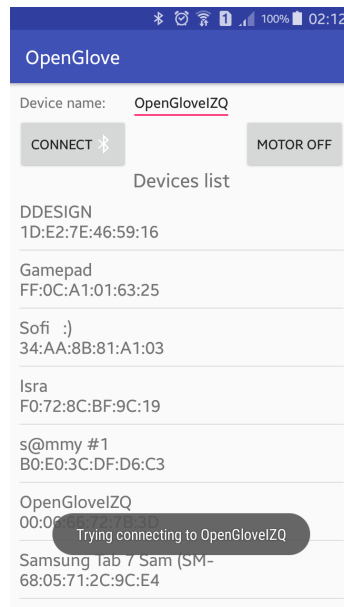


Figura 3.1: Primer prototipo: Activación de motores  
Fuente: elaboración propia (2018).

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 los dispositivos, se requiere de un hilo<sup>2</sup> encargado de ello

<sup>2</sup>Hilo o Thread: En palabras muy resumidas, un hilo es una secuencia instrucciones dentro de un programa que se puede ejecutar independientemente de otro código, permitiendo así la ejecución en paralelo de instrucciones.

(*ConnectedThread*), el cual se comunica con el hilo de la *interfaz de usuario* (UI desde ahora) mediante mensajes. El uso de hilos, permite administrar multiples conexiones con dispositivos Bluetooth de manera simultánea. En conclusión es posible realizar envío de mensajes bajo el protocolo que acepta OpenGlove desde una aplicación Android nativa en Java.

### 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 3.4 se muestra el resumen del prototipo ya mencionado. El RNF004 es cubierto parcialmente, debido a que se consulta continuamente el valor del flexor mediante funciones de bajo nivel, aumentando la latencia debido a las continuas consultas. No se presentan problemas de latencia para la activación y desactivación de motores.

Tabla 3.4: Segundo prototipo  
Fuente: elaboración propia (2018).

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 001
Requisitos no funcionales	RNF002, RNF004 (parcialmente)

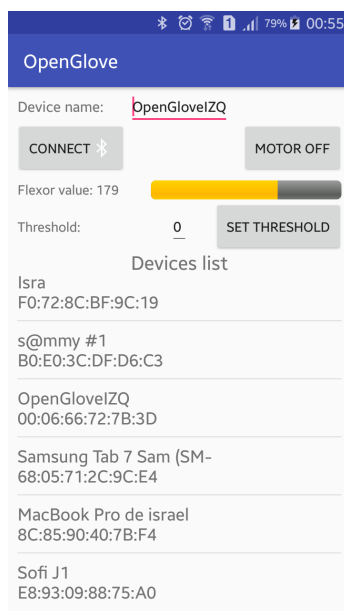


Figura 3.2: Segundo prototipo: obtención de datos desde flexores  
Fuente: elaboración propia (2018).

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.Forms, que es una tecnología de desarrollo móvil multiplataforma <sup>3</sup>, el cual permite desarrollar aplicaciones nativas para Android e iOS en C#. La tabla 3.5 muestra el resumen del tercer prototipo. El RNF004 es cubierto parcialmente, debido a que se consulta continuamente el valor del flexor al igual que el

<sup>3</sup>Traducción libre

segundo prototipo. De igual manera no se presentan problemas de latencia para la activación y desactivación de motores.

Tabla 3.5: Tercer prototipo  
Fuente: elaboración propia (2018).

<b>ID del prototipo</b>	<b>P003</b>
<b>Nombre</b>	Activación de motores y obtención de datos desde los flexores.
<b>Objetivos</b>	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.
<b>Descripción</b>	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.
<b>Requisitos funcionales</b>	RF 001
<b>Requisitos no funcionales</b>	RNF001, RNF002, RNF004 (parcialmente)

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.

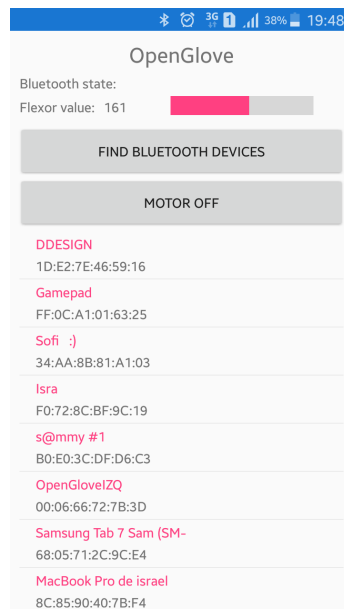


Figura 3.3: Tercer prototipo: activación de motores y obtención de datos desde flexores  
Fuente: elaboración propia (2018).

### 3.2.4 Cuarto prototipo

Este prototipo tiene como propósito cubrir los diferentes requisitos funcionales relacionados al comportamiento y ser la base de las siguientes iteraciones para dar soporte a los sensores de flexibilidad, actuadores e IMU. Se establece la navegación que posee la aplicación para hacer uso de todas las funcionalidades, por medio del diseño de un prototipo usando la herramienta AdobeXD en su versión gratuita, con el cual se diseña, prototipa y se generan las diferentes imágenes e íconos que se utilizaron en el desarrollo del cuarto, quinto y sexto prototipo. La Figura 3.4, muestra el prototipo desarrollado en Xamarin.Forms para Android y iOS, los cuales comparten la interfaz de usuario diferenciándose en la representación nativas de los elementos de cada plataforma.

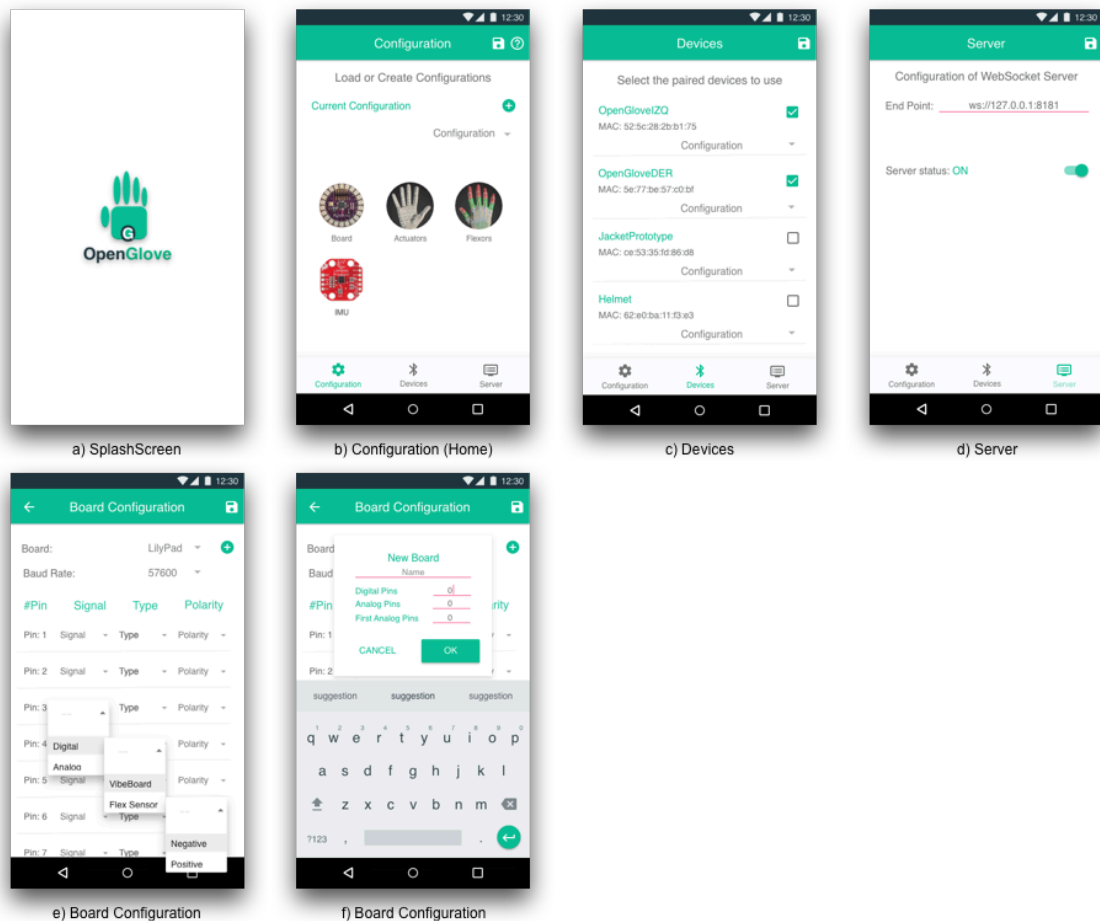


Figura 3.4: Cuarto prototipo: app navigation, devices, server and board configuration  
Fuente: elaboración propia (2018).



### 3.2.5 Quinto prototipo

Este prototipo tiene como propósito cubrir los diferentes requisitos funcionales relacionados a los actuadores.

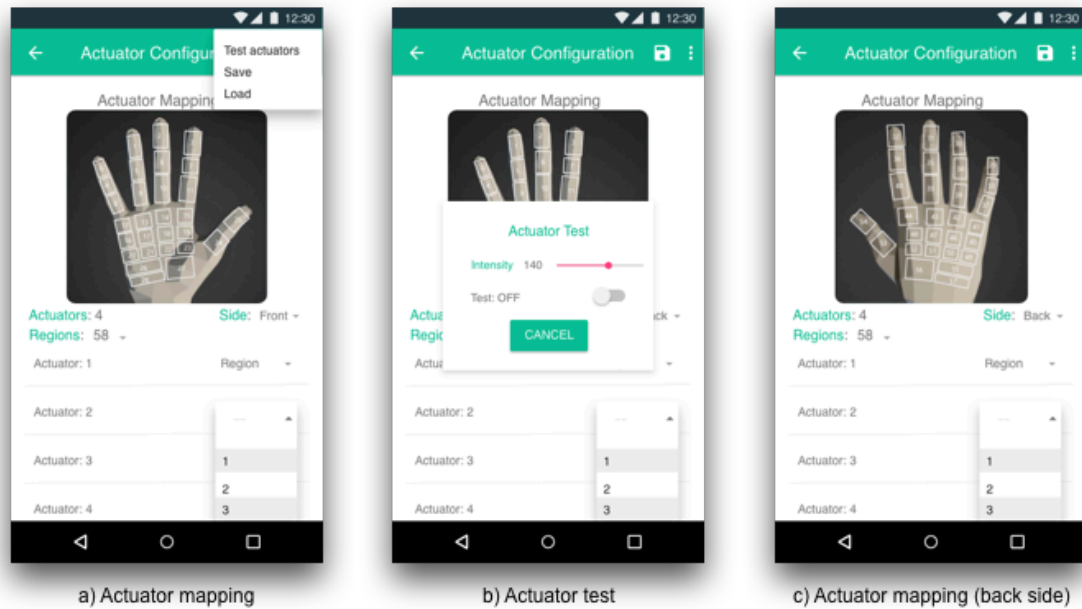


Figura 3.5: Quinto prototipo: actuator mapping and actuator test  
Fuente: elaboración propia (2018).

### 3.2.6 Sexto prototipo

Este prototipo tiene como propósito cubrir los diferentes requisitos funcionales relacionados a los flexores e IMU.

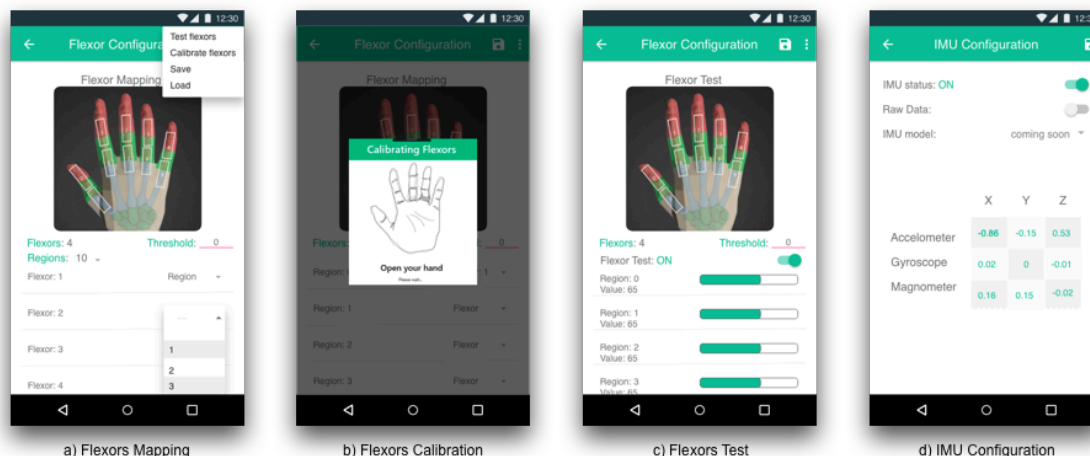


Figura 3.6: Sexto prototipo: flexor mapping, flexor test and IMU configuration  
 Fuente: elaboración propia (2018).

### 3.3 APIS

Para que los desarrolladores puedan hacer uso de las funcionalidades disponibles en el SDK, se requiere del soporte de distintos lenguajes de programación permitiendo abarcar a las más importantes plataformas tecnológicas de Realidad Virtual, Aumentada y Mixta. Por lo tanto se considera la interoperabilidad para los lenguajes C# y Java, como se especificó en el Requisito no funcional RNF003. Estas APIs fueron desarrolladas paralelamente a la aplicación de configuración pues se hace uso de ella, iniciando con la API de alto nivel de C#. Una vez que la API C# fue completada, se replicó utilizando el lenguaje Java. Las mencionadas APIs son clientes WebSockets que establecen conexión con el Servidor WebSocket provisto y administrado por la aplicación de configuración. El primer, segundo y tercer prototipo no incluyen el desarrollo de las APIs de alto nivel, porque estos prototipos abarcaron permitieron evaluar la factibilidad de distintas tecnologías y definir el trabajo a seguir en los siguientes prototipos.

El capítulo Diseño e Implementación especifica en mayor detalle los aspectos arquitecturales de la solución y del comportamiento en los cuales las APIs están involucradas.

#### 3.3.1 Cuarto prototipo

Este prototipo no incluye desarrollo de la APIs de alto nivel, pero si la implementación de distintas tecnologías y la definición del protocolo de mensajes a utilizar por las APIs de los siguientes

prototipos. En este prototipo se implementó un servidor WebSocket que recibe conexiones entrantes de clientes WebSocket, los cuales pueden ser la aplicación de configuración y una aplicación de VR por ejemplo. El servidor WebSocket puede activarse o desactivarse desde la aplicación de configuración, permitiendo además definir el punto de acceso o EndPoint del Servidor. El servidor WebSocket se comunica con la API de Bajo nivel mediante el uso de EventHandlers, los cuales permiten por una parte al Servidor recibir los mensajes provenientes de los dispositivos Bluetooth (datos de flexores e IMU) como también el enviar mensajes para la activación de actuadores, obtención de lista de dispositivos vinculados y la conexión de a los mismos. Es importante destacar que la administración de la conexión de dispositivos Bluetooth está implementada para el Sistema Operativo Android, por lo que es necesario realizar la implementación específica para iOS.

### **3.3.2 Quinto prototipo**

En este prototipo se abarcan las funcionalidades referidas a los actuadores, por tanto se aplicó el protocolo de comunicación definido en el cuarto prototipo, entre los clientes y el servidor WebSocket. Con ello se logró cubrir la inicialización, activación y mapeo de los actuadores mediante la API de alto nivel en C#. Gracias a estas funcionalidades, la aplicación de configuración permite realizar el mapeo y pruebas de activación de los actuadores de los distintos dispositivos OpenGlove conectados.

### **3.3.3 Sexto prototipo**

Este prototipo incluye todas las funcionalidades de la API C# para el uso de OpenGlove en dispositivos móviles utilizando este lenguaje de programación. Esto se logró desarrollando las funcionalidades que no fueron cubiertas en el quinto prototipo, las referidas a los sensores de flexibilidad e IMU. Las funcionalidades implementadas respecto a los sensores de flexibilidad , permiten agregar un flexor a una región para iniciar la transmisión de datos, remover un flexor de una región, calibrar los flexores, testear los flexores asignados a una región y la asignación de un umbral o Threshold (el dispositivo OpenGlove, enviará el dato de un flexor si la diferencia es mayor o igual a este valor). Respecto a las funcionalidades referidas al IMU, se permite iniciar el IMU, asignar el status del IMU, recibir datos crudos o procesados del IMU. Inicio y suspensión de lectura de datos es una funcionalidad que involucra a ambos sensores.

### 3.4 RESUMEN

Durante el desarrollo del SDK, fue necesario un desarrollo de seis prototipos, los primeros tres que permitieron la validando la factibilidad de un desarrollo nativo en Android comparado al desarrollo multiplataforma con Xamarin.Forms. Los siguientes prototipos corresponden al desarrollo de la aplicación de configuración y las APIs para lograr cumplir con los requisitos funcionales y no funcionales del SDK. La Tabla 3.6 muestra el resumen de los requisitos funcionales, en el cual se aprecia la cobertura de los mismos en cada prototipo anteriormente detallado. La Tabla 3.7, muestra de la misma forma la cobertura de los requisitos no funcionales del SDK.

Tabla 3.6: Matriz de prototipos de software vs requisitos funcionales  
Fuente: Elaboración propia (2018)

	P001	P002	P003	P004	P005	P006
RF001	X	X	X	X	X	X
RF002				X	X	X
RF003				X	X	X
RF004				X	X	X
RF005					X	X
RF006					X	X
RF007					X	X
RF008	X	X	X	X	X	X
RF009					X	X
RF010				X	X	X
RF011					X	X
RF012						X
RF013						X
RF014						X
RF015						X
RF016						X
RF017						X
RF018						X
RF019						X
RF020						X
RF021						X
RF022						X
RF023						X

Tabla 3.7: Matriz de prototipos de software vs requisitos no funcionales  
Fuente: Elaboración propia (2018)

	P001	P002	P003	P004	P005	P006
RNF001			X	X	X	X
RNF002	X	X	X	X	X	X
RNF003						X
RNF004	X	X		X	X	X

# CAPÍTULO 4. DISEÑO E IMPLEMENTACIÓN

## 4.1 ARQUITECTURA GENERAL

La Figura 4.1, muestra la arquitectura general de OpenGlove, donde es posible ver el trabajo previo realizado por Monsalve (2015), Meneses (2016) y Cerda (2017), como también el trabajo que resulta de este proyecto y el que se realizará en un futuro.

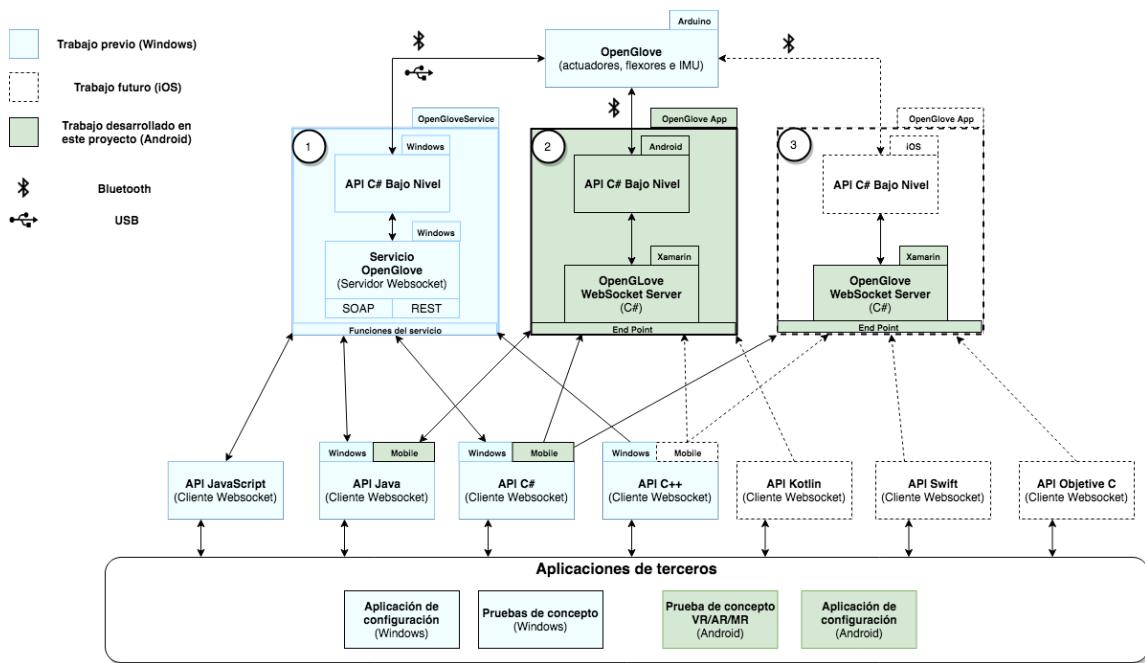


Figura 4.1: Arquitectura de OpenGlove  
Fuente: Elaboración propia (2018)

La aplicación de configuración de OpenGlove desarrollada en este proyecto, permite comunicarse con dispositivos OpenGlove a través de comunicación serial Bluetooth. Para acceder a sus funcionalidades los desarrolladores hacen uso de las APIs de alto nivel, las cuales corresponden a clientes WebSocket. Éstas pueden ser utilizadas en cualquier aplicación compatible con el lenguaje de la API a utilizar. Dado que se utiliza Xamarin.Forms para desarrollar aplicaciones nativas multiplataforma, para iOS y Android, el servidor Websocket es desarrollado utilizando bibliotecas de software de C#, para que ambas plataformas compartan el código. La API de bajo nivel, debe ser específica en cada sistema operativo, por las diferencias de comunicación presentes en cada uno. Por esta misma razón se desarrollan las APIs de alto nivel en Java y C#. La primera permite hacer uso de OpenGlove en proyectos Android y la segunda permite hacer aplicaciones para iOS y Android utilizando Xamarin. El uso de Xamarin.Forms permite reutilizar

código de la Interfaz de Usuario y desarrollar de manera específica para cada plataforma utilizando un mismo lenguaje de programación. Cabe destacar que la comunicación realizada entre las APIs y la aplicación es mediante WebSockets considerando que OpenGlove es una aplicación que transmite datos en tiempo real.

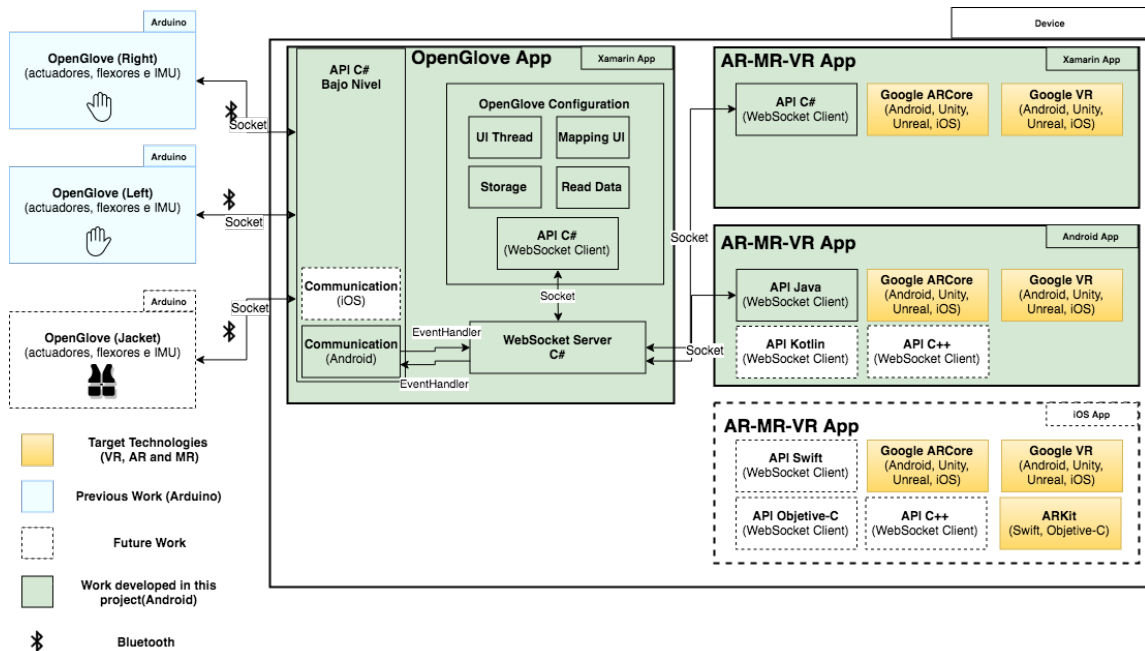


Figura 4.2: Detalle Arquitectura de OpenGlove  
Fuente: Elaboración propia (2018)

La Figura 4.2 muestra el detalle de la arquitectura considerando el objetivo de uso de las APIs de alto nivel en Java y C#. Las APIs de alto nivel mencionadas permiten el desarrollo de aplicaciones de terceros en Android nativo usando Java o con C# utilizando Xamarin. Para realizar aplicaciones de AR o VR, se puede utilizar Google ARCore y Google VR respectivamente. Ambas herramientas de Google están disponibles para Android, iOS, Unity y Unreal.

Dado el alcance de la solución, no se ha desarrollado APIs de alto nivel que tengan como objetivos aplicaciones de terceros para iOS, para lo cual requiere un dispositivo iPhone real para realizar las pruebas de Bluetooth y funcionamiento, o en su defecto un MacBook para poder visualizar la interfaz de la aplicación, pero sin hacer uso de las funcionalidades Bluetooth ni WebSocket. También es importante recalcar la necesidad de implementar parte de la API de bajo nivel en C# (Communication) específica para iOS para gestionar las conexiones de los dispositivos Bluetooth y la administración de mensajes entre la API de bajo nivel y el servidor WebSocket. Este es un trabajo que se puede realizar a futuro para dar soporte a OpenGlove en iOS.

## **4.2 ESTRUCTURA**

En esta sección se detalla la estructura de los distintos componentes de software desarrollados, los cuales son: el servicio que expone las funcionalidades, las APIs que las consumen, el software de configuración y el diagrama de clases de cada uno.

### **4.2.1 Servicio**

Desarrollando ...

### **4.2.2 APIs**

Desarrollando ...

### **4.2.3 Software de configuración**

## **4.3 COMPORTAMIENTO**

A continuación se describe el comportamiento de los diferentes componentes del sistema. Estos diagramas son basados en el trabajo realizado por Meneses (2016) y Cerda (2017), dado que se busca replicar el comportamiento del sistema en dispositivos móviles.

4.3.1 Obtener guantes

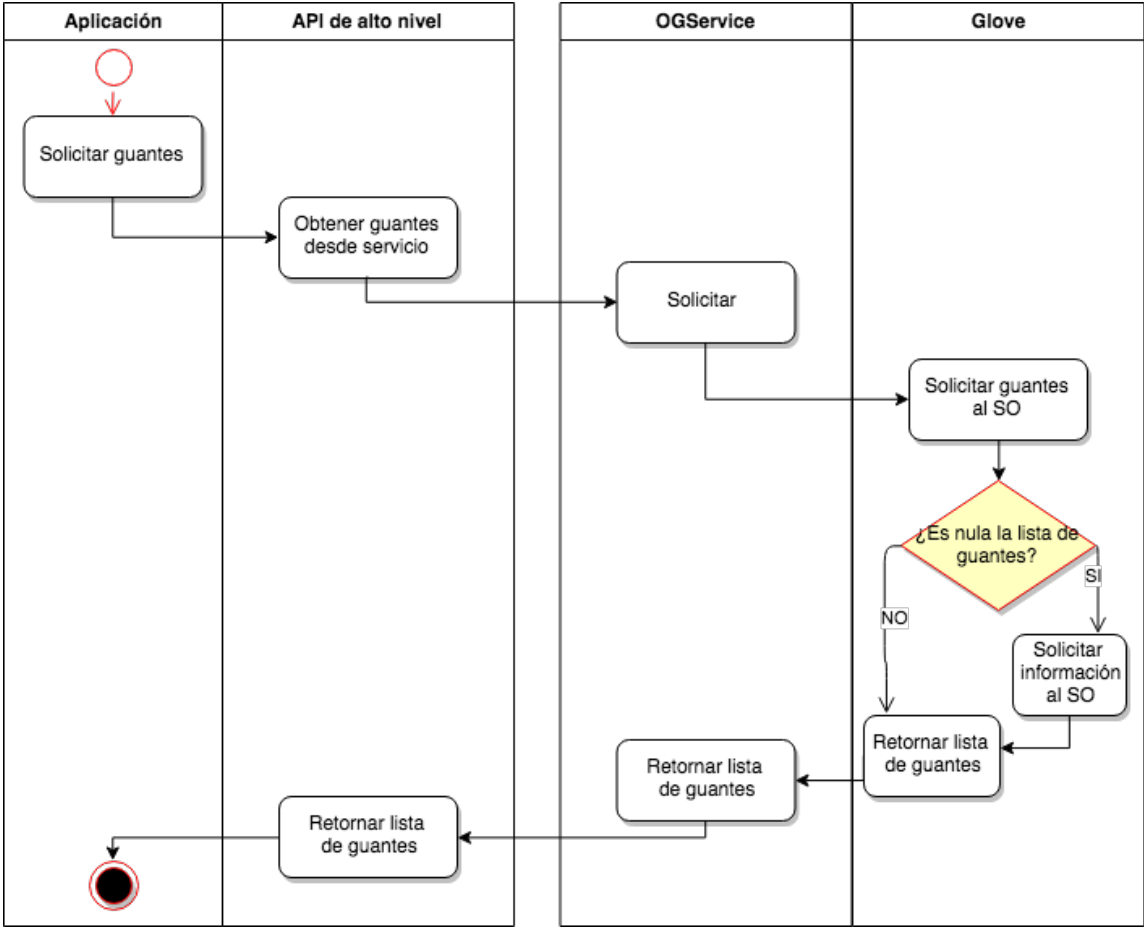


Figura 4.3: Obtener guantes  
Fuente: Elaboración propia (2018)



4.3.2 Activación

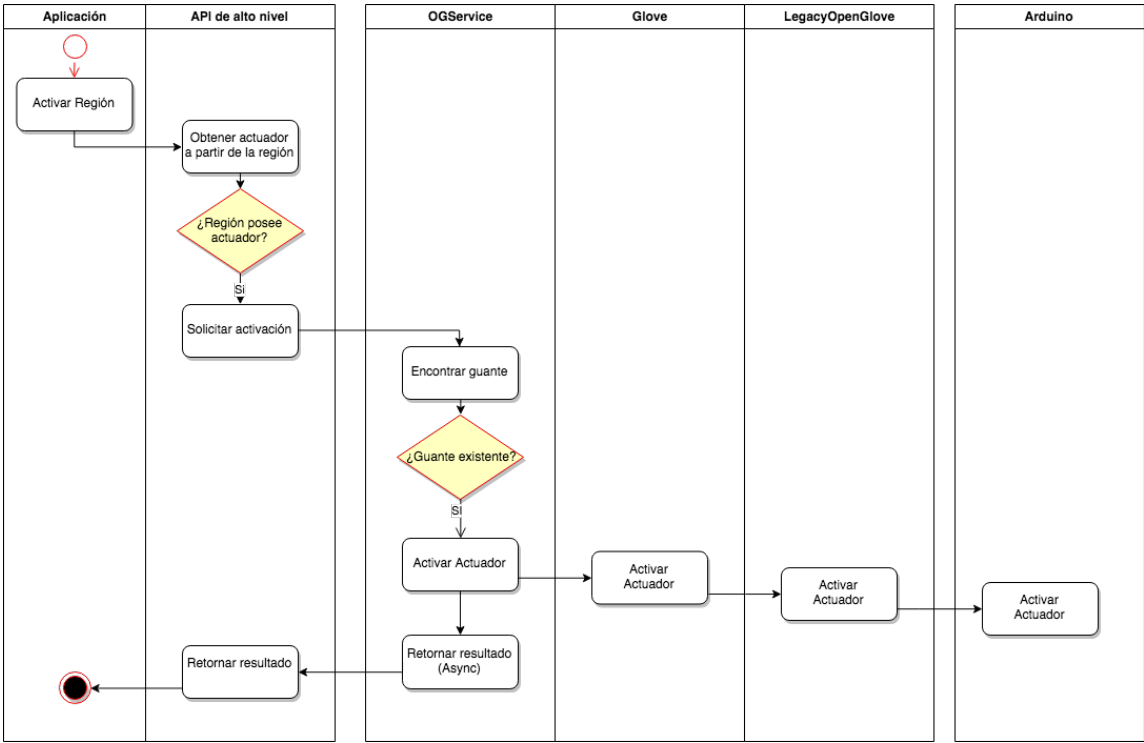


Figura 4.4: Activar región  
Fuente: Elaboración propia (2018)

4.3.3 Añadir flexor a una región

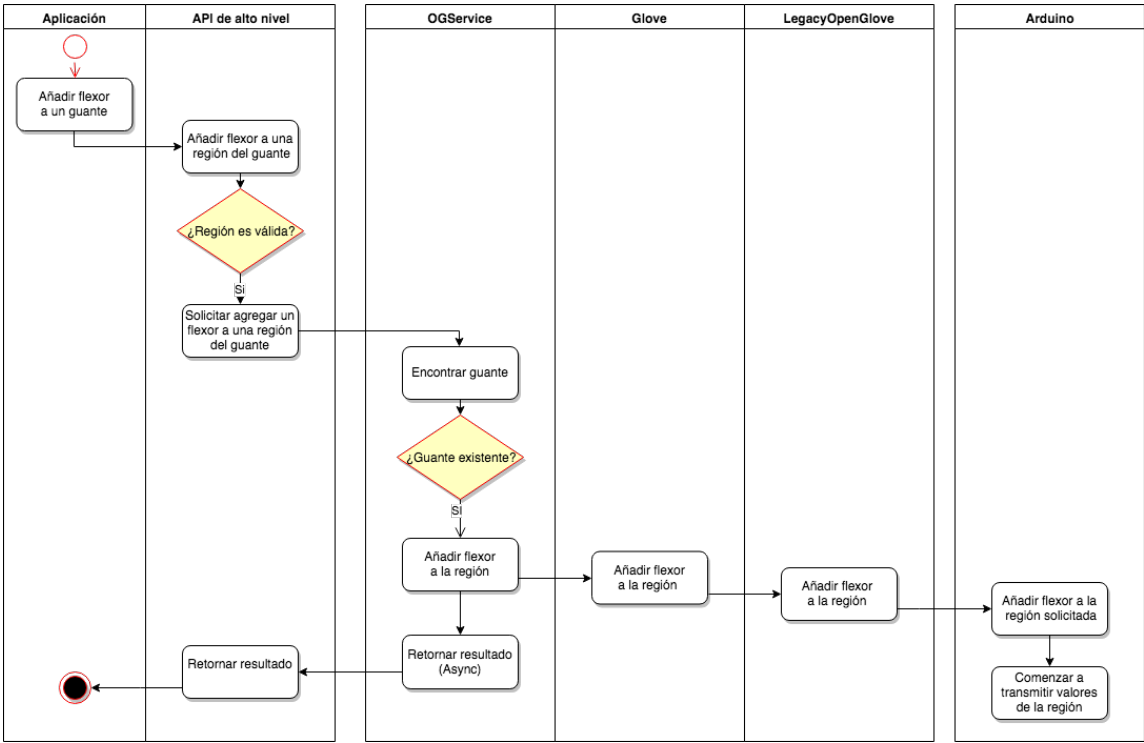


Figura 4.5: Agregar flexor a una región  
Fuente: Elaboración propia (2018)

4.3.4 Asignar Threshold

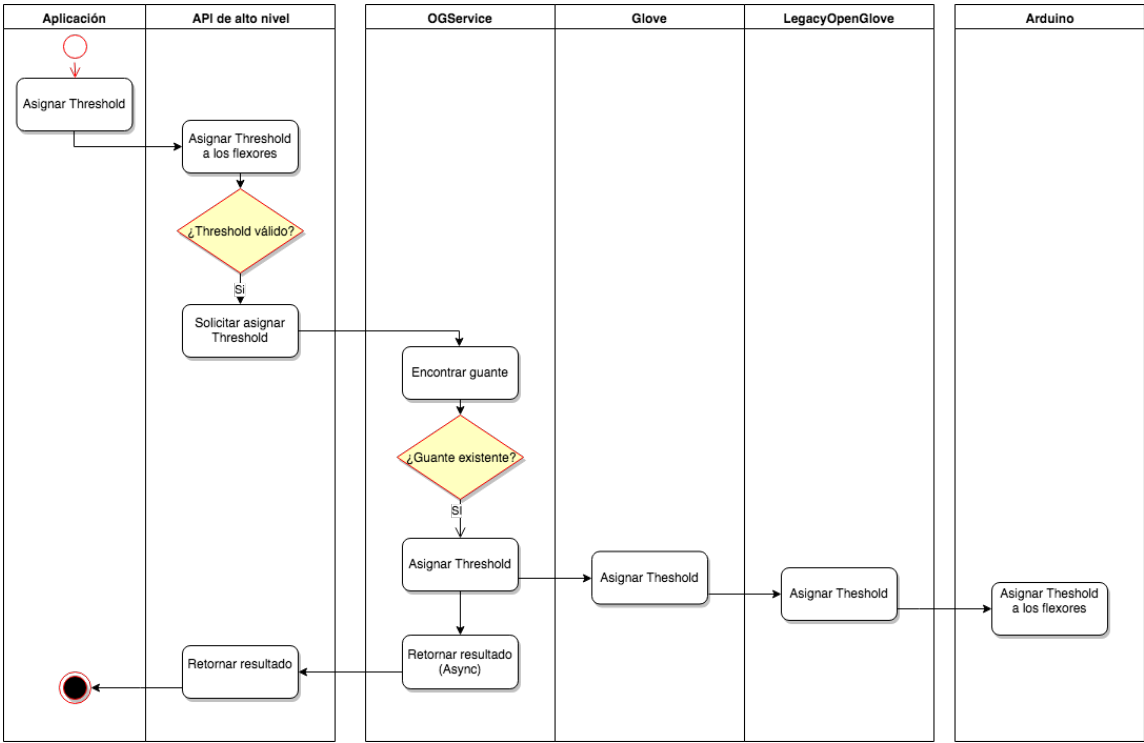


Figura 4.6: Asignar Threshold  
Fuente: Elaboración propia (2018)

4.3.5 Iniciar IMU

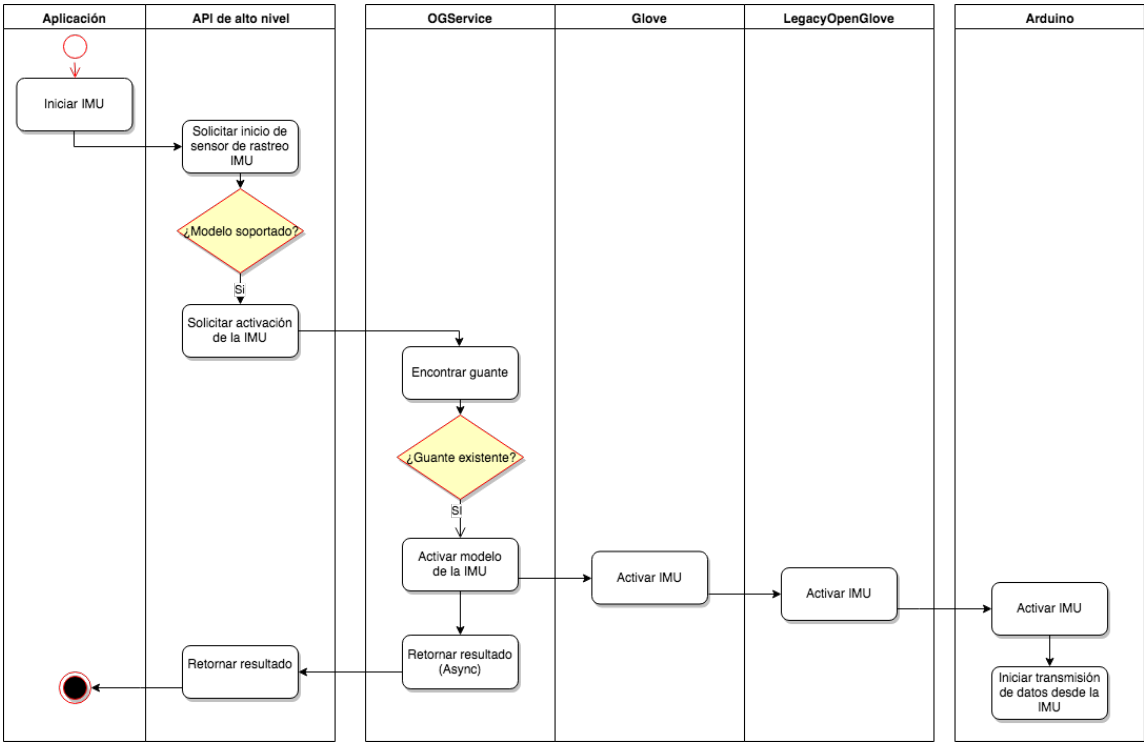


Figura 4.7: Iniciar IMU  
Fuente: Elaboración propia (2018)

#### 4.3.6 Lectura de datos provenientes de Arduino

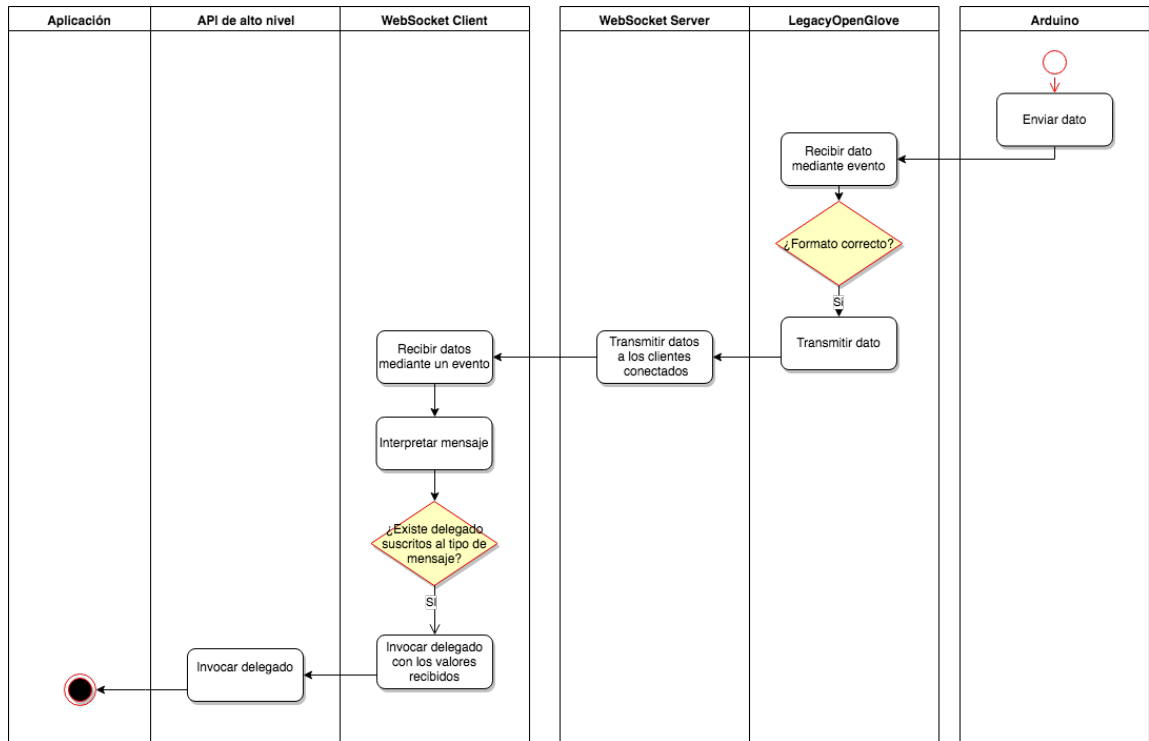


Figura 4.8: Lectura de datos provenientes de Arduino  
Fuente: Elaboración propia (2018)

## **4.4 ASPECTOS DE IMPLEMENTACIÓN**

### **4.4.1 Desarrollo multiplataforma en Xamarin.Forms**

### **4.4.2 Servicio**

### **4.4.3 Software de configuración**

### **4.4.4 APIs**

#### *4.4.4.1 Java*

#### *4.4.4.2 C#*

## **4.5 RESUMEN**

## 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. Para el análisis de latencia de los motores, se considerando el tiempo medido de activación y desactivación con una precisión de nanosegundos ( $ns$ ), transformando los resultados a microsegundos ( $\mu s$ ). Por tanto el umbral límite de 60 milisegundos ( $ms$ ) es equivalente a 60000  $\mu s$ .

#### 5.1.1 Prototipo 3 : Droid - Galaxy

##### 5.1.1.1 Motores

Tabla 5.1: Resumen resultado pruebas motor Droid-Galaxy  
Fuente: Elaboración propia (2018)

Motors	Mean	Median	Min	Max	Std. Dev.	Skewness	Kurtosis
1	245.345	170.156	135.469	681.354	141.369	1.336	3.376
2	263.026	272.031	192.084	357.083	36.334	-0.114	2.965
3	570.168	375.026	313.802	1,576.875	337.726	1.257	2.911
4	418.957	424.843	304.688	668.750	61.357	0.417	4.613
5	484.390	497.240	353.073	724.479	78.401	0.221	3.303

La Figura 5.1, muestra los histogramas de las latencias obtenidas al mandar los mensajes de activación y desactivación. Se realizaron pruebas desde uno a cinco motores.

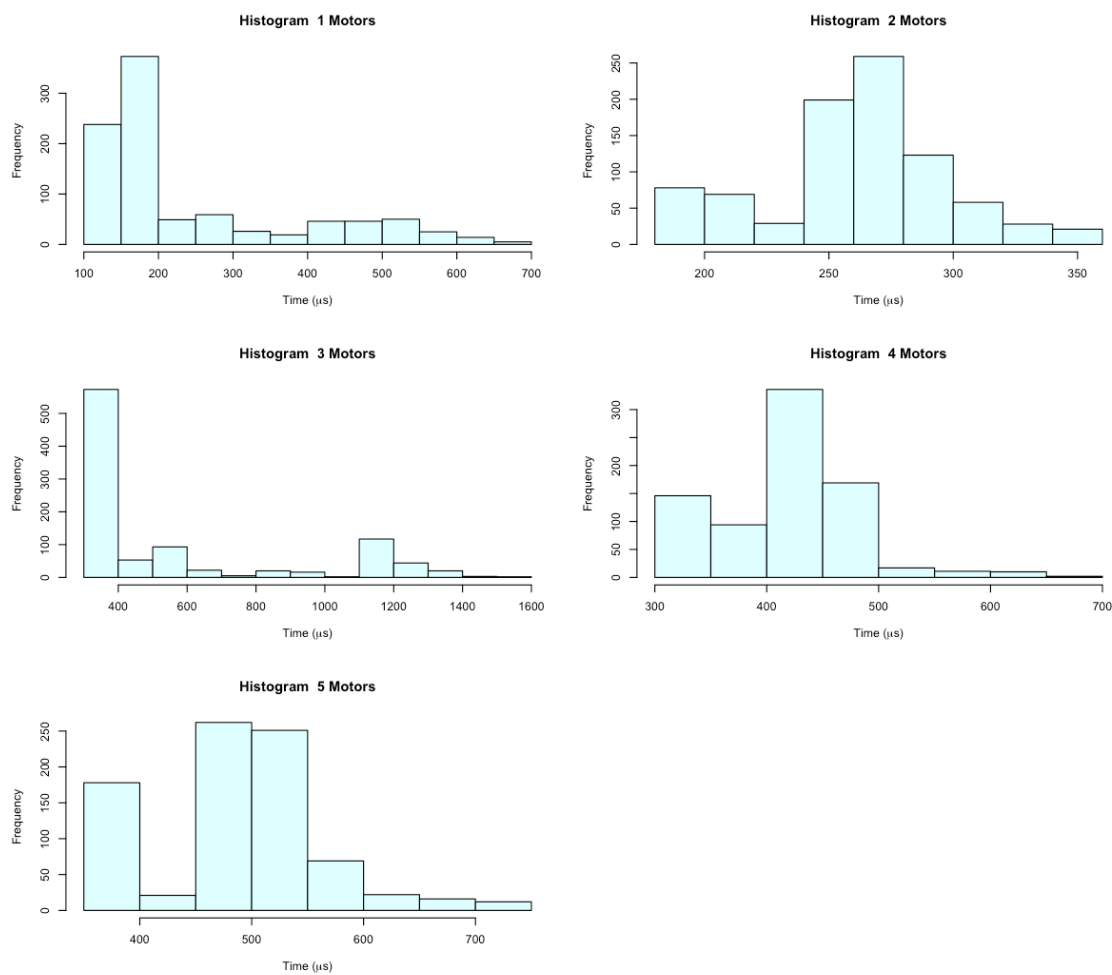


Figura 5.1: Histogramas de motores Droid-Galaxy  
Fuente: elaboración propia (2018)



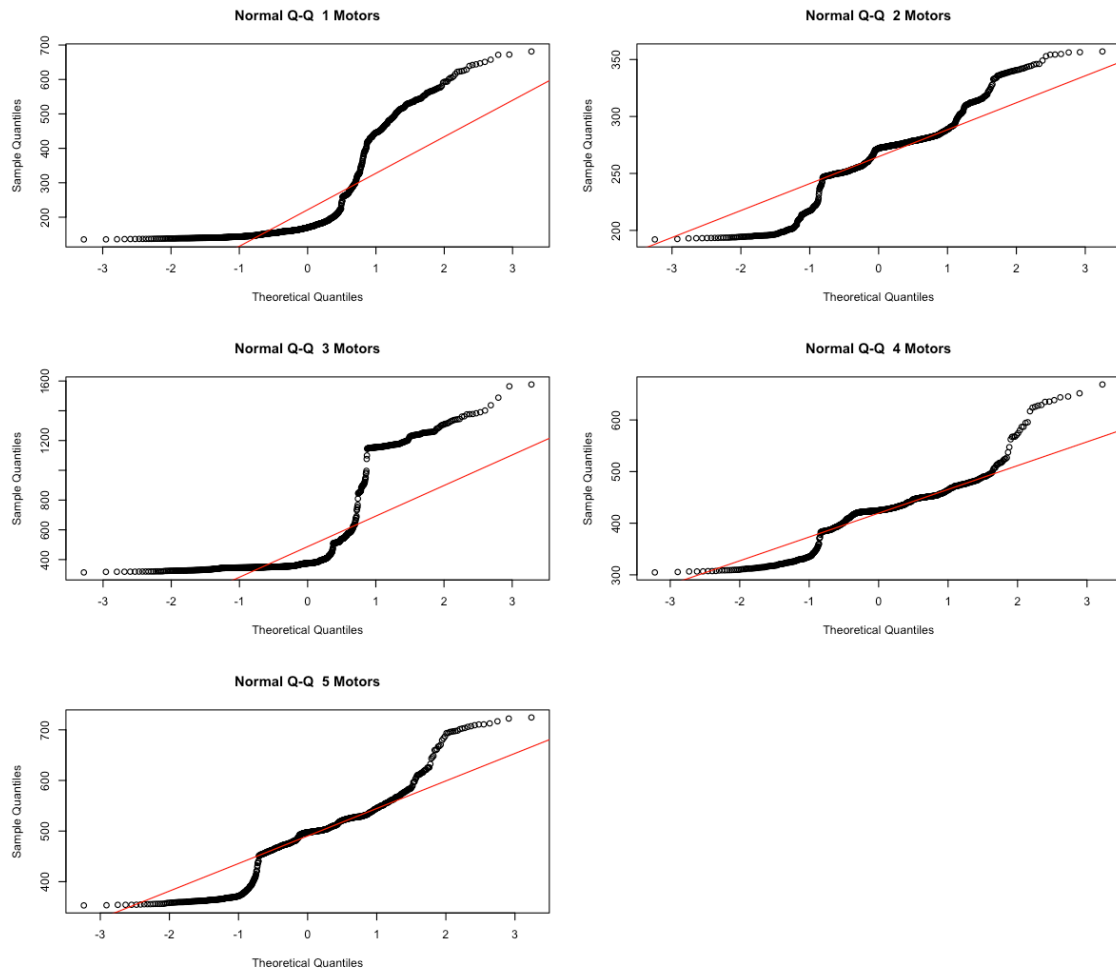


Figura 5.2: Gráficos QQ de motores Droid-Galaxy  
Fuente: elaboración propia (2018)

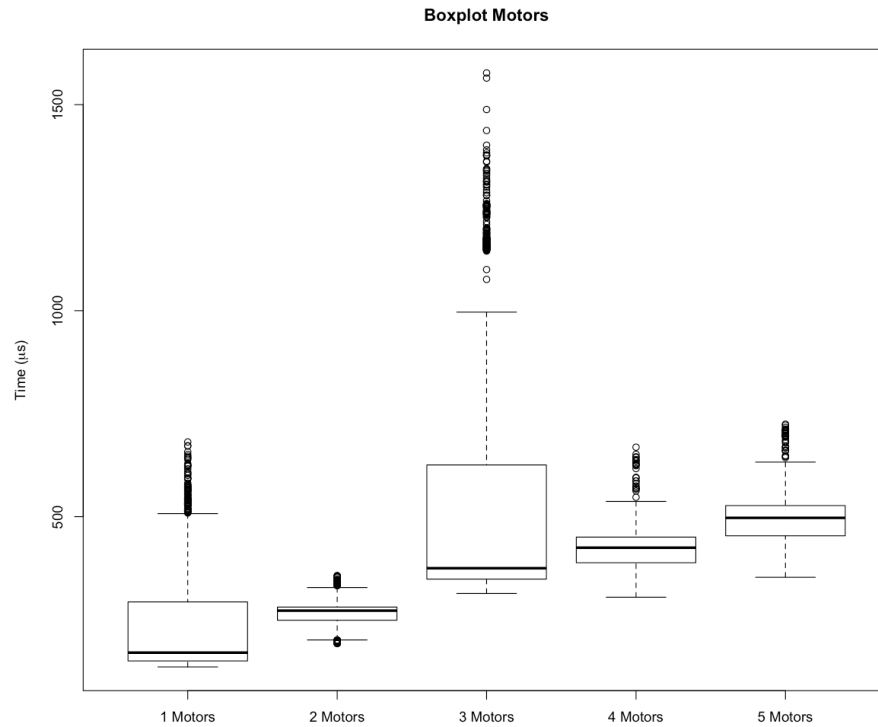


Figura 5.3: Gráficos de cajas de motores Droid-Galaxy  
Fuente: elaboración propia (2018)

#### 5.1.1.2 Flexores

Tabla 5.2: Resumen resultado pruebas flexor Droid-Galaxy  
Fuente: Elaboración propia (2018)

Motors	Mean	Median	Min	Max	Std. Dev.	Skewness	Kurtosis
1	121,879.900	108,551.400	34,473.860	252,435.200	38,318.940	0.658	2.637

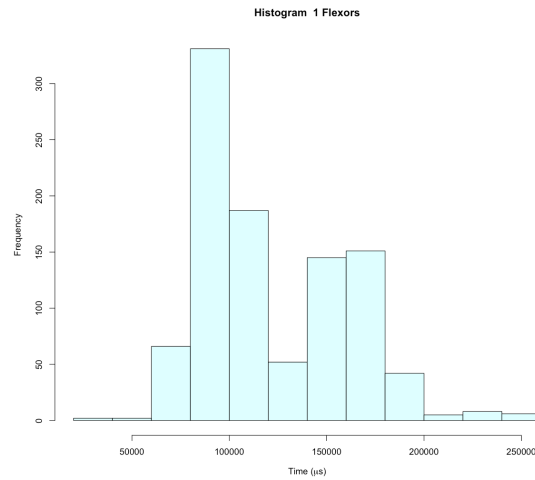


Figura 5.4: Histogramas de flexores Droid-Galaxy  
Fuente: elaboración propia (2018)

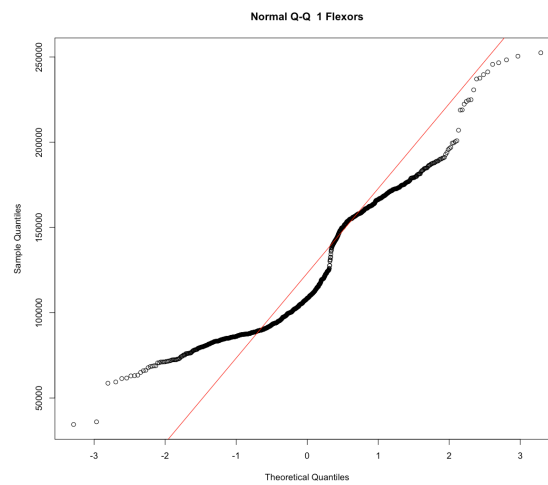


Figura 5.5: Gráficos QQ de flexores Droid-Galaxy  
Fuente: elaboración propia (2018)

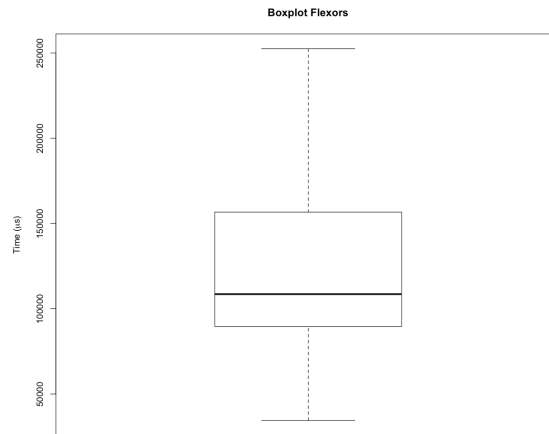


Figura 5.6: Gráficos de cajas de flexores Droid-Galaxy  
Fuente: elaboración propia (2018)

### 5.1.2 Prototipo 4: Xamarin - Galaxy

#### 5.1.2.1 Motores

Tabla 5.3: Resumen resultado pruebas motor Xamarin-Galaxy  
Fuente: Elaboración propia (2018)

Motors	Mean	Median	Min	Max	Std. Dev.	Skewness	Kurtosis
1	299.835	268.800	231.500	512	77.214	1.376	4.065
2	367.823	387.100	282	600.800	59.888	0.290	2.832
3	363.318	358.200	333	472.100	26.389	1.707	6.343
4	491.245	506.050	380.800	785.600	78.420	0.503	3.042
5	574.120	600	429.200	832.200	84.596	0.045	2.703

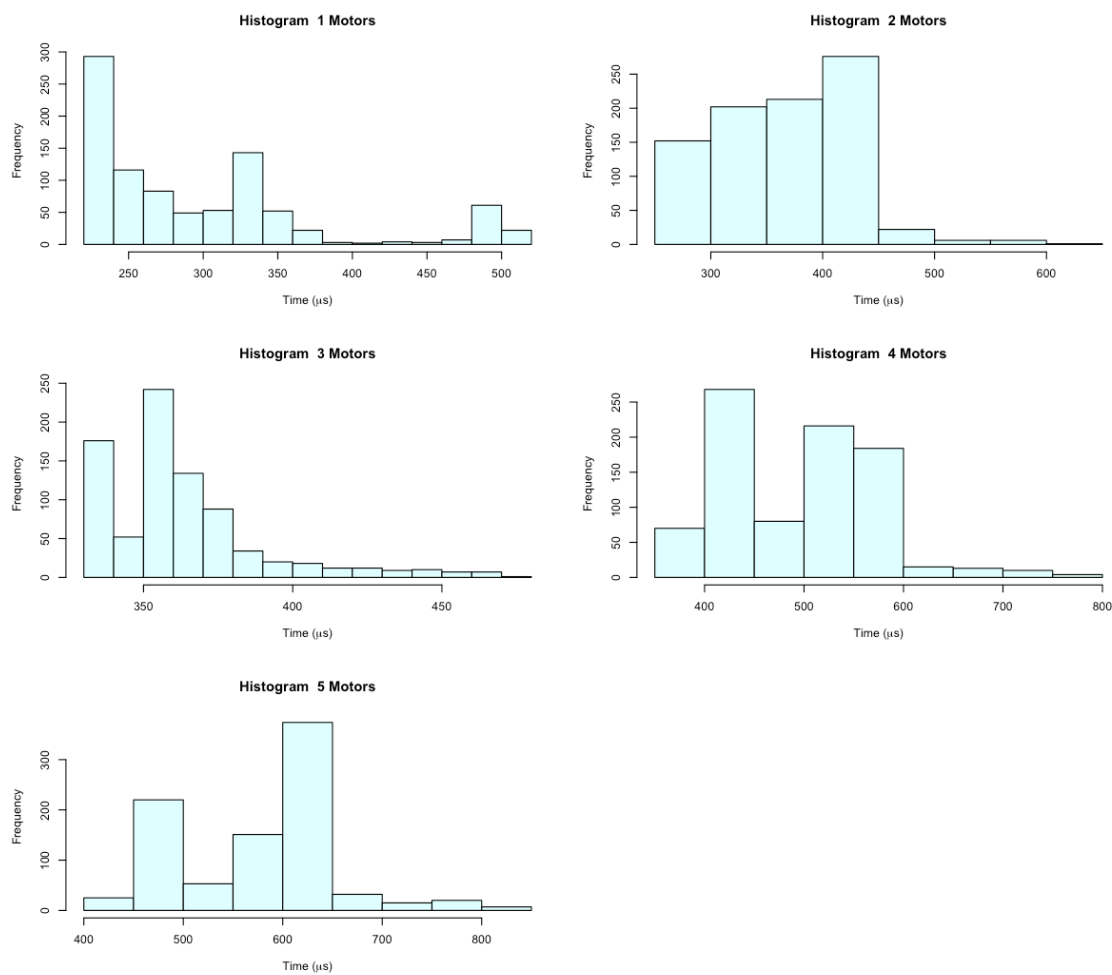


Figura 5.7: Histogramas de motores Xamarin-Galaxy  
Fuente: elaboración propia (2018)

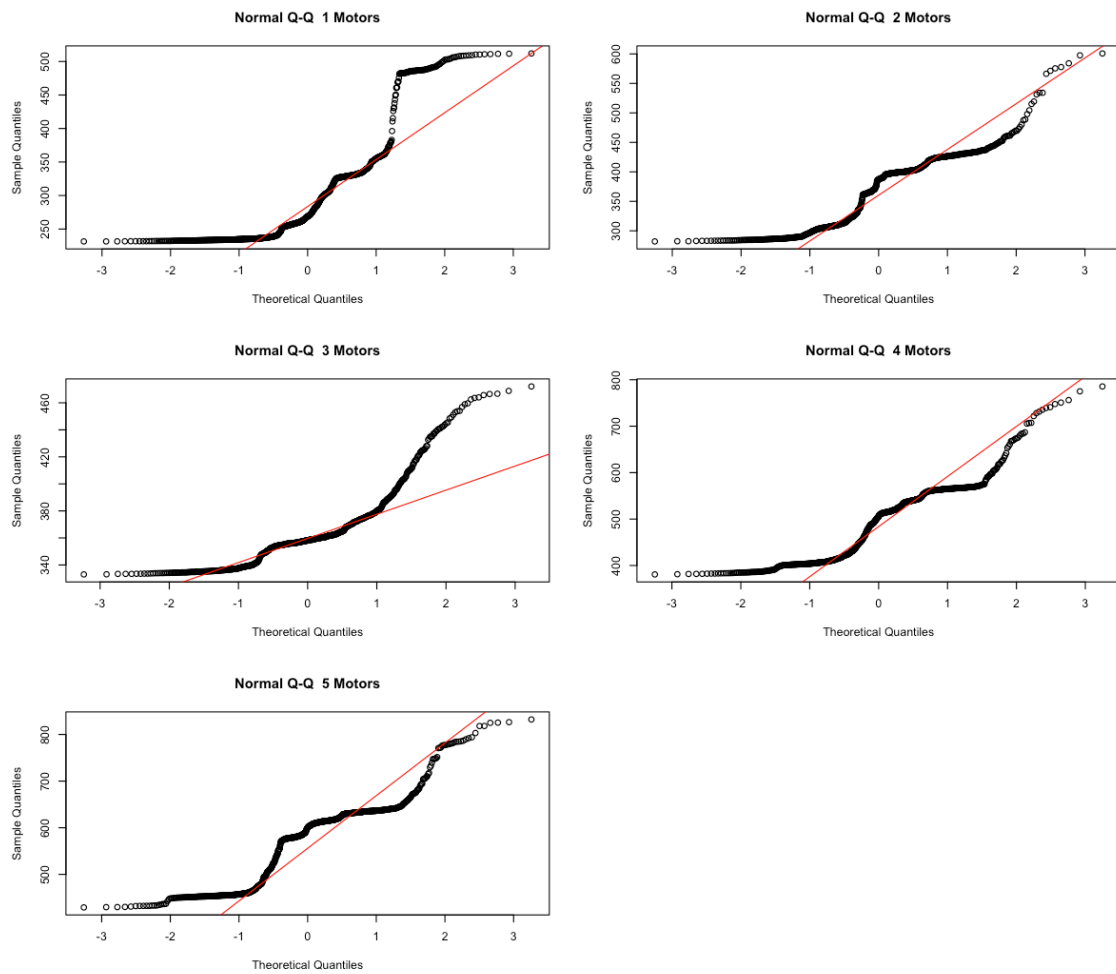


Figura 5.8: Gráficos QQ de motores Xamarin-Galaxy  
Fuente: elaboración propia (2018)

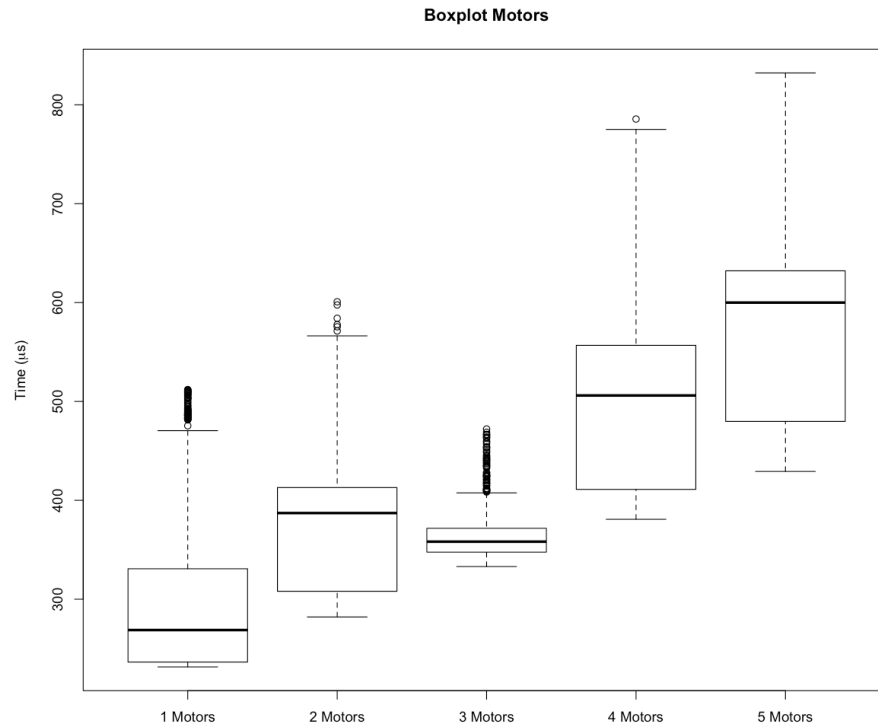


Figura 5.9: Gráficos de cajas de motores Xamarin-Galaxy  
Fuente: elaboración propia (2018)

#### 5.1.2.2 Flexores

Tabla 5.4: Resumen resultado pruebas flexor Xamarin-Galaxy  
Fuente: Elaboración propia (2018)

Motors	Mean	Median	Min	Max	Std. Dev.	Skewness	Kurtosis
1	123,483.700	109,237.600	55,364.200	253,747.900	39,493.110	0.696	2.824

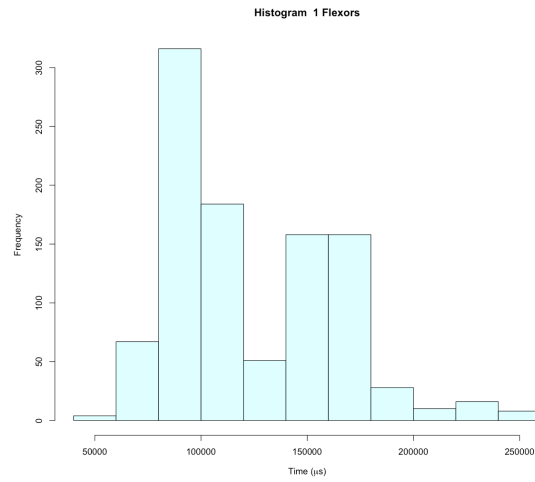


Figura 5.10: Histogramas de flexores Xamarin-Galaxy  
Fuente: elaboración propia (2018)

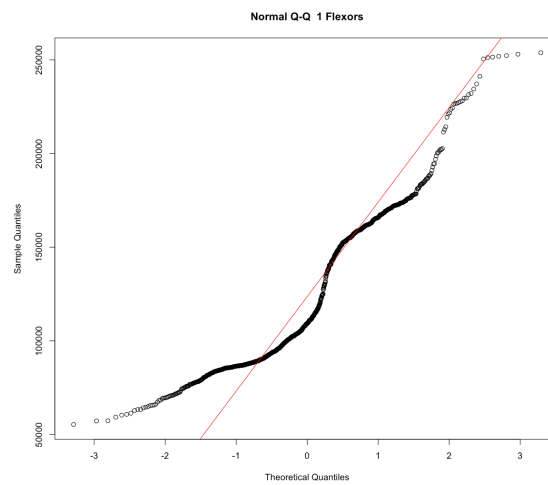


Figura 5.11: Gráficos QQ de flexores Xamarin-Galaxy  
Fuente: elaboración propia (2018)



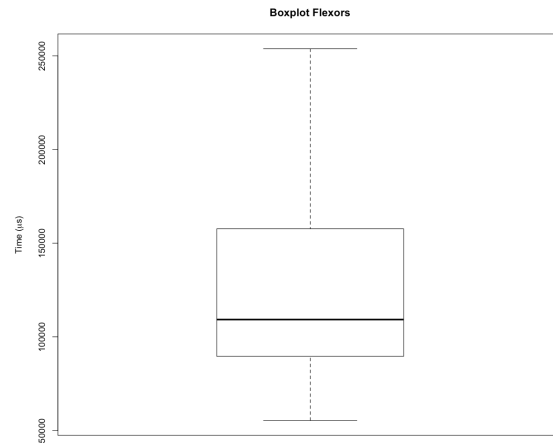


Figura 5.12: Gráficos de cajas de flexores Xamarin-Galaxy  
Fuente: elaboración propia (2018)

### **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.
- Batteau, L. M., Liu, A., Maintz, J. B. A., Bhasin, Y., & Bowyer, M. W. (2004). A study on the perception of haptics in surgical simulation. In S. Cotin, & D. Metaxas (Eds.) *Medical Simulation*, (pp. 185–192). Berlin, Heidelberg: Springer Berlin Heidelberg.
- 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.
- Gartner (2018). It glossary: Operating system. Recuperado de <https://www.gartner.com/it-glossary/os-operating-system> Revisado el 28 Junio de 2018.
- Grigorik, I. (2013). High performance browser networking: Websocket. Recuperado de <https://hpbn.co/websocket/> Revisado el 25 de Junio de 2018.
- Haptx (2018). Haptx. Recuperado de <https://manus-vr.com/order.php> el 28 de Junio de 2018.
- 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 <https://manus-vr.com/order.php> 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](https://developer.microsoft.com/en-us/windows/mixed-reality/mixed_reality) Revisado el 26 de Noviembre de 2017.
- MobiLoud (2018). Native, web or hybrid apps? what's the difference? Recuperado de <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/> Revisado el 30 de Junio de 2018.
- Monsalve, R. A. (2015). Dispositivo de retroalimentación táctil para interfaces naturales y de realidad 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.

- Okamoto, S., Konyo, M., Saga, S., & Tadokoro, S. (2009). Detectability and perceptual consequences of delayed feedback in a vibrotactile texture display. *IEEE Transactions on Haptics*, 2(2), 73–84.
- 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.
- Statista (2017). The smartphone platform war is over. Recuperado de <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/> Revisado el 29 de Junio de 2018.
- 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.
- Techterms (2010). Sdk definition. Recuperado de <https://techterms.com/definition/sdk> Revisado el 30 de Junio de 2018.
- 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.
- Websocket.org (2018a). About html5 websocket. Recuperado de <http://websocket.org/aboutwebsocket.html> Revisado el 5 de Junio de 2018.
- Websocket.org (2018b). Html5 websocket: A quantum leap in scalability for the web. Recuperado de <https://websocket.org/quantum.html> Revisado el 25 de Junio de 2018.
- Wu, B., Sim, S. H., Enquobahrie, A., & Ortiz, R. (2015). Effects of visual latency on visual-haptic experience of stiffness. In *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, (pp. 1–6).