DESARROLLO DE UNA BIBLIOTECA QUE PERMITA LA PROGRAMACIÓN DE APLICACIONES MÓVILES QUE UTILICEN INTERFACES FÍSICAS ARTESANALES

JULIÁN ANDRÉS RODRÍGUEZ ALMANZA

UNIVERSIDAD MILITAR NUEVA GRANADA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA EN MULTIMEDIA

CAJICÁ

2020

DESARROLLO DE UN *TOOLKIT* QUE PERMITA LA PROGRAMACIÓN DE APLICACIONES MÓVILES QUE UTILICEN INFERFACES FÍSICAS ARTESANALES

JULIÁN ANDRÉS RODRÍGUEZ ALMANZA

Desarrollo tecnológico para optar por el título de:
INGENIERA EN MULTIMEDIA

DIRIGIDO POR:
ING. LAURA JULIANA CORTÉS RICO, MSc.

UNIVERSIDAD MILITAR NUEVA GRANADA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA EN MULTIMEDIA

CAJICÁ

2020

AGRADECIMIENTOS

En primer lugar, quisiera agradecer a mis padres que me han ayudado y apoyado durante toda mi vida. A mi directora de proyecto Laura Cortes por su apoyo, dedicación y orientación en todo el desarrollo del proyecto.

Contenido

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE CÓDIGOS

RESUMEN

Las interfaces tangibles son aquellas que tienen componentes tanto digitales como físicos, entendiendo físicos como objetos cotidianos, manipulables, con los que se interactúa, principal pero no únicamente, a través de las manos. En este trabajo se realizó un desarrollo tecnológico en torno a la construcción de este tipo de interfaces, haciendo uso de dispositivos móviles con sistema operativo Android. Este desarrollo se realizó en el marco del proyecto de investigación INV-ING-2981 “Diálogos de saberes en la creación de interfaces físicas artesanales para la interacción con dispositivos móviles”, financiado por la Universidad Militar Nueva Granada, convocatoria interna Vigencia 2019. El desarrollo es la biblioteca SensorMov que integra una serie de funcionalidades que buscan reducir el tiempo de desarrollo de aplicaciones que requieran el acceso a los diferentes sensores que se encuentran en el dispositivo móvil. En particular, cuando los sensores del móvil se utilizan para detectar ciertas acciones o identificadores sobre objetos físicos que hacen parte de la interacción.

En este documento se detalla la investigación y proceso realizados para la creación de esta biblioteca. Así mismo, sobre cada funcionalidad se muestran ejemplos de implementación, junto con videos. Se anexan los manuales de usuario e instalación.

INTRODUCCIÓN

Las interfaces tangibles de usuario (TUI, por su sigla en inglés) son un concepto propuesto por Hiroshi Ishii, que corresponde a interfaces de usuario como aquellos objetos físicos que permiten al usuario modificar o manipular información en el mundo digital [8]. Uno de los referentes más famosos de TUI es *The Reactable Experience,* un instrumento musical electrónico que permite experimentar con el sonido, y cambiar su estructura, agregando, moviendo, rotando o quitando objetos de la superficie *Reactable[15]*. Los objetos son cubos que, además, se pueden conectar entre sí para combinar efectos, sintetizadores, modificar volumen, etc. Todos los efectos tienen una representación visual, convirtiendo la música en algo visual y tangible [15].

Este tipo de interfaces se han fortalecido en el área de interacción humano computador (HCI) desde los años 90's por el Instituto Tecnológico de Massachusetts – MIT [16]. En Colombia, ha sido poco el desarrollo que se ha dado a este tema y se puede observar en algunos trabajos como ApTUI -Framework para el diseño participativo de interacciones tangibles [17]. Proyecto donde se obtuvo un *framework* que propone formas para diseñar interacciones tangibles, a través de una investigación sobre procesos de diseño participativo e involucraba caladoras, diseñadores, científicos sociales e ingenieros de áreas diversas [17]. Dado que este concepto acerca la computación a los usuarios a través de objetos físicos, que pueden ser de uso cotidiano y que pueden contribuir con procesos de apropiación de las tecnologías de la información y de las comunicaciones (TIC), se desarrolla una biblioteca para un sistema operativo abierto, como Android, que pueda aumentar y diversificar la cantidad y calidad de aplicaciones móviles que usen TUI. Este desarrollo se realizó en el marco del proyecto de investigación INV-ING-2981 "Diálogos de saberes en la creación de interfaces físicas artesanales para la

interacción con dispositivos móviles", financiado por la Universidad Militar Nueva Granada, convocatoria interna, vigencia 2019. En este sentido, los requisitos funcionales y, especialmente los *no* funcionales, surgieron de los diálogos que se dieron entre estudiantes de Ingeniería en Multimedia, diseñadores de interacciones, una antropóloga y artesanas textiles de la región Bogotá Sabana en Colombia. Uno de los más significativos es que la biblioteca sea en español, considerando que se desea propiciar, localmente, el desarrollo de interfaces tangibles que aprovechen el extendido uso de dispositivos móviles, y que, a la vez, reconozcan las capacidades locales, como en el caso de las artesanías textiles.

La biblioteca *SensorMov* está desarrollada en lenguaje nativo, facilitando la comunicación con el sistema operativo. Esto permite que se pueda combinar con otras funcionalidades, como las de comunicación, manejo de bases de datos, gestión de archivos, entre otras. Así mismo, está exportada en un formato que permitiría incluirla, con algunos ajustes, en otras plataformas de desarrollo, como *Processing for Android*.

Este documento se divide en las siguientes secciones: Introducción, en la que se explican de forma general algunos conceptos y se da un contexto al proyecto, antecedentes, donde se incluyen algunos de referentes que se desarrollaron con el objetivo de facilitar tareas a los desarrollados de este tipo de tecnologías, marco conceptual, que presenta los conceptos de los dispositivos móviles y las interfaces tangibles. Metodología, donde se detalla la forma en la que fue desarrollado el proyecto. Biblioteca SensorMov, que contiene el diseño de la biblioteca, sus clases, atributos, métodos y ejemplos. Finalmente, conclusiones y trabajo futuro.

PLANTEAMIENTO DEL PROBLEMA

En el proyecto de investigación "Diálogo de saberes en la creación de interfaces físicas artesanales para la interacción con dispositivos móviles" Código INV-ING-

2981 en uno de sus objetivos buscó crear aplicaciones móviles que permitieran interactuar con los dispositivos móviles, a través de objetos físicos artesanales, es decir, hechos manualmente.

El proyecto de investigación tuvo una duración de un año, por tanto, se tuvo poco tiempo para poder desarrollar la aplicación que interactúe con los objetos tangibles. La creación de interfaces físicas requiere un gran desarrollo para la comunicación entre los objetos que permiten la interacción con la aplicación, esto se puede ver reflejado en trabajos como *Phidgets* donde uno de sus objetivos es asegurarse de la correcta comunicación de los sensores con un computador para permitir al usuario centrarse en cómo usar los datos obtenidos [1]. Dentro del proyecto se realizaron talleres con los artesanos buscando crear nuevas interfaces tangibles, la parte importante de los talleres era la experiencia e interacción entre los miembros del equipo y los artesanos por lo que se debía encontrar una manera de facilitarles el trabajo técnico y de desarrollo de dichas interfaces tangibles.

Así, se planteó como pregunta motivadora de este desarrollo tecnológico: ¿Cómo reducir los tiempos de desarrollo de quienes requieren crear interfaces tangibles que hagan uso de dispositivos móviles, permitiéndoles concentrarse más en la experiencia de usuario que en el desarrollo técnico?

JUSTIFICACIÓN

El presente desarrollo se enfoca en el desarrollo de una biblioteca que permita facilitar el desarrollo de interfaces físicas artesanales dentro del proyecto “Diálogo de saberes en la creación de interfaces físicas artesanales para la interacción con dispositivos móviles”. Debido a que la creación de bibliotecas, *frameworks* o *toolkits* facilita herramientas de alto nivel, que permiten a potenciales desarrolladores de diferentes tipos de aplicaciones mejorar el tiempo que les toma realizar su aplicación o producto, lo que puede también hacer que se reduzcan algunos costos al momento del desarrollo. En el caso de interfaces tangibles, realizarlas requiere un

alto conocimiento técnico para la comunicación entre la aplicación digital y los objetos físicos con los cual va a interactuar.

Una de las bibliotecas más famosas que podemos tomar de ejemplo para el sistema operativo Android es *ARToolkit,* que ayuda en el desarrollo de aplicaciones que hacen uso de la realidad aumentada, como el seguimiento de los objetos en el escenario físico [2]. Entre sus ventajas está que es rápido y bastante preciso, además de su fácil acceso hace que sea utilizado por muchos investigadores que trabajan en esta área alrededor del mundo demostrando su gran utilidad [2].

Se buscó utilizar los dispositivos móviles porque son de uso cotidiano para la mayoría de las personas, permitiendo que fuera un punto de convergencia entre los miembros del equipo del proyecto y las artesanas que participaron el proyecto, que era uno de los puntos claves encontrar puntos clave dentro del proyecto. Además, la interacción con un celular, al ser con nuestras manos, se constituyó como una herramienta de interacción que se pudo acercar más al quehacer artesanal. Así mismo, se buscó generar espacios de reflexión sobre el mundo artesanal y el digital, como mundos no tan diferentes.

DELIMITACIÓN

La biblioteca que permite la programación de diversas aplicaciones móviles, para que interactúen con interfaces físicas artesanales, se desarrolló en 12 meses, trabajando desde mayo a mayo del año 2020. La biblioteca va dirigido al proyecto de investigación “Diálogo de saberes en la creación de interfaces físicas artesanales para la interacción con dispositivos móviles” Código INV-ING-2981. Éste se desarrolló con los artesanos textiles de la región sabana centro (municipios de Cogua, Cajicá, Zipaquirá y Bogotá.).

OBJETIVOS

OBJETIVO GENERAL

Desarrollar una biblioteca que permita la programación de aplicaciones móviles que utilicen interfaces físicas artesanales.

OBJETIVOS ESPECÍFICOS

- Definir la arquitectura de la biblioteca que utilice los recursos de un dispositivo móvil.
- Especificar los requisitos funcionales y no funcionales de la biblioteca.
- Programar la biblioteca para la creación de las aplicaciones que interactúen con las interfaces físicas.
- Evaluar el cumplimiento de los requisitos funcionales de la biblioteca.
- Evaluar el correcto funcionamiento y usabilidad de los manuales de la biblioteca.

ANTECEDENTES

En esta sección se presentan cinco antecedentes. Productos similares: *Phidgets[1]*, *CapWidgets[3] y TUIO[4]*, artículos: *SPINE: A TUI Toolkit and Physical Computing Hybri*d[5], *Evolving TUIs with smart objects for multi-context interaction[6]* y el Proyecto *Composing Cube[7]*.

Los *Phidgets* son bloques que facilitan la detección y control de variables físicas como la temperatura, humedad, presión, etc.; usando un computador o un dispositivo móvil. Los *Phidgets* se conectan al computador por puerto USB (Ver figura 1); para su uso se requiere la *API Phidgets* que está disponible en diferentes lenguajes de programación como C#, C++, Python, Java, Visual Basic, JavaScript, etc. [1]. Sus librerías tienen soporte con los siguientes sistemas operativos: Windows, Linux, OS X, iOS, Android y *Phidget* SBC [1].

Su mayor ventaja es su fácil comunicación con el computador, se encarga de que los elementos electrónicos funcionen como deberían esto permite concentrarse en la programación y detalles del producto que se quiera desarrollar [1].

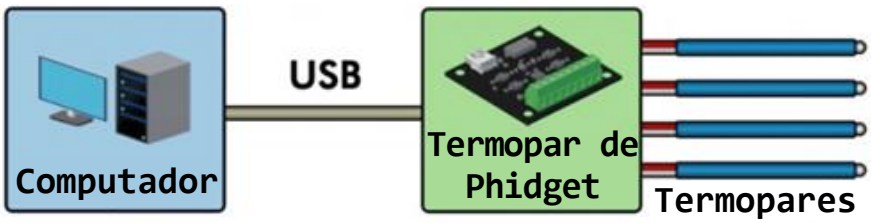

Figura 1 - Ejemplo para conectar un Phidget traducido [1]

Para el uso de los *phidgtes* lo primero que se hace es abrir un canal para permitir el seguimiento del *phidget* que se esté utilizando, luego definir parámetros básicos para indicar a qué *phidget* se debe conectar; después, abrir el canal para hacer coincidir el canal físico del *phidget* con el del software que se esté desarrollando. Antes de empezar con la programación del producto, es necesario verificar que el *phidget* se adjuntó al programa, después se usa un *loop* donde se recibe la

información captada por el *phidget* y se realiza la programación buscando el objetivo del producto. Por último, se debe cerrar el canal del *phidget* [1 (Ver figura 2)].

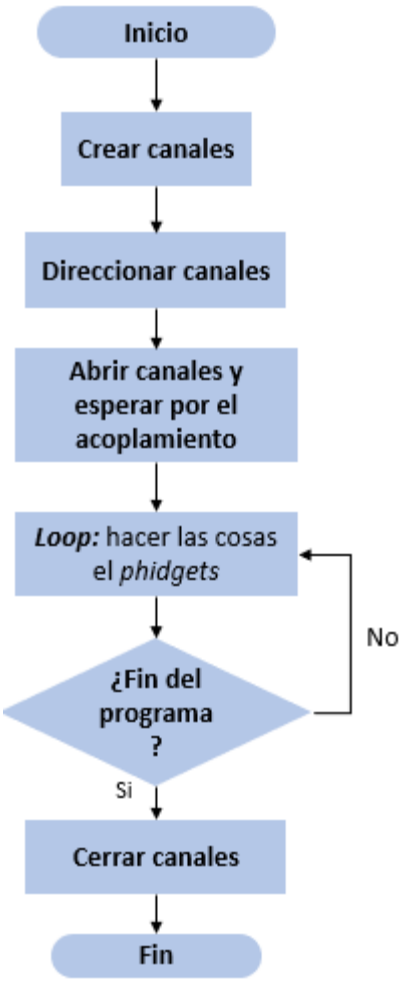

Figura 2 - Diagrama de un programa con Phidgets traducido [1].

Los *CapWidgets* son controles tangibles para pantallas táctiles. Se crean puntos de contacto artificiales en lugares arbitrarios en el parte inferior de un *CapWidget*. Estos puntos son detectables por una pantalla táctil de un dispositivo móvil cuando entra en contacto con los *CapWidgets* (Ver figura 3). Esta técnica permite la creación de una gran cantidad de controles físicos para pantallas táctiles [3].

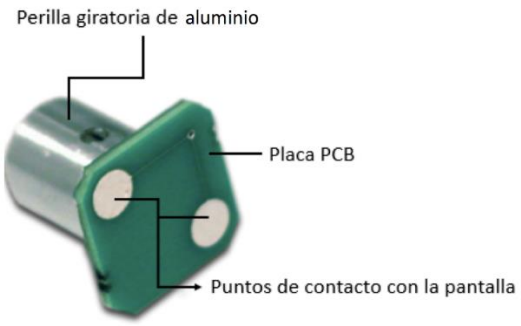

Figura 3 - Imagen de un tangible de CapWidgets modificada [3]

La PCB crea puntos de contacto artificiales al transmitir el potencial de tierra del usuario a la superficie de la pantalla táctil [3].

TUIO es un *framework* que tiene un protocolo y una API para superficies multitáctiles tangibles. TUIO permite la transmisión de eventos táctiles y estados de objetos tangibles. Este protocolo codifica los datos y los envía a cualquier aplicación que sea capaz de decodificar el protocolo, permite el rápido desarrollo de interfaces multitáctiles tangibles basadas en tablas. TUIO se basa en *Open Sound Control*, un estándar para entornos interactivos [4].

En el artículo *SPINE: A TUI Toolkit and Physical Computing Hybri*d, se presenta un conjunto de herramientas orientado al desarrollo de pines universales que permitan una conexión entre tarjetas de desarrollo y una variedad de sensores, sé hizo utilizando una mezcla de técnicas entre la computación física y las interfaces tangibles de usuarios (TUI), el objetivo es facilitar la creación de interfaces musicales por parte de artistas o compositores en lugar programadores [5].

Evolving TUIs with smart objects for multi-context interaction es un artículo, se creó para extender el concepto de interfaces naturales y tangibles a entornos compuestos por muchos sistemas interactivos. Los usuarios pueden realizar actividades individuales o en grupos utilizando diferentes sistemas (como mesas o

muros interactivos) e interactuar con ellos a través de objetos inteligentes tangibles con sensores, almacenamiento, procesamiento y con comunicación inalámbrica. [6]

Composing Cubes (Ver figura 4) es un proyecto donde la posición en *x y* de cada cubo era rastreado y cada cubo representa diferentes instrumentos, dependiendo de la posición en *y* dentro del tablero cambia el volumen y cada posición en *x* representa ciertas características como el eco o la reverberación del sonido [6], se hizo mediante *ARToolkit*, que es una librería que permite calcular la posición y orientación de objetos físicos en tiempo real, esta librería se encuentra en el HITLab de la universidad de Washington [7]

Figura 4-Composing Cubes [7]

MARCO CONCEPTUAL

En esta sección se describirán algunos conceptos relevantes para el proyecto y que se tuvieron en cuenta en cada momento de este desarrollo, se divide en dos grandes secciones: interfaces tangibles de usuario y dispositivos móviles e interacción con tangibles.

INTEFACES TANGIBLES DE USUARIO

Hiroshi Ishii propone el termino interfaces de usuario tangibles (TUI)y establece que su objetivo es aprovechar habilidades de interacción háptica [8]. Define el objetivo de las TUI como dar formas físicas a la información digital. Las TUI permiten manipular lo digital con nuestras manos. Las TUI sirven como una interfaz de propósito especial para una aplicación [8].

CLASIFICACIÓN

De acuerdo con Ishii y Ulmer, las TUI se clasifican en cuatro categorías [9]:

- Espacial: Sistemas que interpretan la posición y orientación de los *tokens* en el espacio físico.
- Constructivos: Sistemas de construcción en bloques o elementos modulares que pueden ser acoplados.
- Relacionales: Sistemas donde hay relaciones entre diferentes *tokens* con información digital.
- Asociativos: es parecido al relacional donde los tokens se asocian con información digital, pero en este los tokens no tienen relación unos con otros.

DISPOTIVOS MÓVILES E INTERACCIÓN CON TANGIBLES

La evolución de los dispositivos móviles ha llegado a un punto que permite crear diferentes interacciones, ya que, la mayoría de los dispositivos con tienen sensores que miden el movimiento, la orientación y diversas condiciones ambientales. Estos sensores son capaces de proporcionar datos sin procesar con alta precisión, y son

útiles si desea monitorear el movimiento del dispositivo o revisar su posición, o si desea monitorear los cambios ambientales cerca de un dispositivo.

A continuación, se muestra una tabla que cuenta con todos los sensores que vienen en un dispositivo *Android*, dando su nombre, tipo, una breve descripción y sus usos comunes.

Tabla 1 Sensores de los dispositivos Android [10]. La tipología puede ser i) Hardware, que refiere a aquellos que son componentes físicos integrados en el móvil y que obtienen los datos directamente de la captura, o ii) Software, también llamados sintéticos o virtuales, son datos que se derivan de la combinación de sensores de hardware, y no son componentes físicos.

Sensor	Tipo	Descripción	Usos comunes
TYPE_ACCELEROMETER	Hardware	Mide la fuerza de aceleración en m/s^2 que se aplica a un dispositivo en los tres ejes físicos (*x*, *y*, y *z*), incluida la fuerza de la gravedad.	Detección de movimiento (sacudir, inclinación, etc.).
TYPE_AMBIENT_TEMPERATURE	Hardware	Mide la temperatura ambiente de la habitación en grados Celsius ($°C$).	Monitorear la temperatura del aire.
TYPE_GRAVITY	Software o Hardware	Mide la fuerza de gravedad en m/s^2 que se aplica a un dispositivo en los tres ejes físicos (*x*, *y*, y *z*).	Detección de movimiento (sacudir, inclinación, etc.).
TYPE_GYROSCOPE	Hardware	Mide la velocidad de rotación de un dispositivo en $rad\ /\ s$ alrededor de cada uno de los tres ejes físicos (*x*, *y*, y *z*).	Detección de rotación.
TYPE_LIGHT	Hardware	Mide el nivel de luz ambiente (iluminación) en lx.	Controlar el brillo de la pantalla.
TYPE_LINEAR_ACCELERATION	Software o Hardware	Mide la fuerza de aceleración en m/s^2 que se aplica a un dispositivo en los tres ejes físicos (*x*, *y*, y *z*), excluyendo la fuerza de la gravedad.	Monitoreo de la aceleración a lo largo de un solo eje.
TYPE_MAGNETIC_FIELD	Hardware	Mide el campo geomagnético ambiental para los tres ejes físicos (*x*, *y*, y *z*) en μT.	Crear una brújula.
TYPE_ORIENTATION	Software	Mide los grados de rotación que realiza un dispositivo alrededor de los tres ejes físicos (*x*, *y*, y *z*). Funciona a partir del nivel API-3, puede obtener la matriz de inclinación y la matriz de rotación para un dispositivo utilizando el sensor de gravedad y el sensor de campo geomagnético junto con el	Determinación de la orientación del dispositivo.

		método `getRotationMatrix()`.	
TYPE_PRESSURE	Hardware	Mide la presión del aire ambiente en hPa o $mBar$.	Monitoreo de cambios de presión de aire.
TYPE_PROXIMITY	Hardware	Mide la proximidad de un objeto en cm con respecto a la pantalla de visualización de un dispositivo. Este sensor se usa generalmente para determinar si un auricular se está sosteniendo cerca del oído de una persona.	Posición del teléfono durante una llamada.
TYPE_RELATIVE_HUMIDITY	Hardware	Mide la humedad ambiental relativa en porcentaje (%).	Monitoreo del punto de rocío, humedad absoluta y relativa.
TYPE_ROTATION_VECTOR	Software o Hardware	Mide la orientación de un dispositivo proporcionando los tres elementos del vector de rotación del dispositivo.	Detección de movimiento y detección de rotación.
TYPE_TEMPERATURE	Hardware	Mide la temperatura del dispositivo en grados Celsius ($°C$). La implementación de este sensor varía según el dispositivo. Este sensor se reemplazó con el sensor TYPE_AMBIENT_TEMPERATURE en el nivel 14 de API	Monitoreo de temperaturas.

El uso de estos sensores se puede ver en diversos artículos, por ejemplo, el magnetómetro o brújula se usa en *MagiThings: Gestural Interaction with Mobile Devices Based on Using Embedded Compass (Magnetic Field)* Sensor, donde se reconocen diferentes gestos realizados con imanes alrededor del celular detectando los cambios en el campo magnético del entorno, las aplicaciones creadas permiten dibujar en el aire, cambiar una canción, desbloquear el celular, etc [11].

En *MobiLimb: Augmenting Mobile Devices with a Robotic Limb* desarrollaron un tangible que sirve como accesorio para la pantalla táctil, en su variedad de aplicaciones se puede ver la idea de crear “mensajes de texto físicos”, el brazo podía acariciar o dar pequeños toques a un usuario, también podía escribir por ti en

un papel o mostrar gestos como para mascotas virtuales, o levantarse para indicar que había un mensaje nuevo [12].

Otro que usa la pantalla táctil es *Interactiles: 3D Printed Tactile Interfaces to Enhance Mobile Touchscreen Accessibility*, busca crear una interfaz tangible para ayudar a usuarios que tengan dificultades para usar los dispositivos móviles, esta interfaz es impresa en 3D, y permite un teclado numérico, un *scroll* y un botón para poder controlar el dispositivo móvil [13].

Una forma inesperada de usar los micrófonos nace en *Sound of Tapping user interface technology with medium identification*, detectando golpes en una mesa alrededor del celular permite encontrar la posición donde se dio el golpe, ya que el sonido llegará de forma distinta a los micrófonos con los que contaba el dispositivo, en sus aplicaciones estaba subir y bajar en una página web dando golpes en una mesa, moverse, hacer zoom y rotar dentro de un mapa [14].

METODOLOGÍA

Se utilizó una metodología ágil, basada en SCRUM, pero adaptada a las particularidades de este proyecto, en donde se puede observar un proceso divido en cinco grandes etapas: Investigación, definición de tareas, trabajo personal, retroalimenta ción y por último la entrega o finalización del proyecto (figura 5).

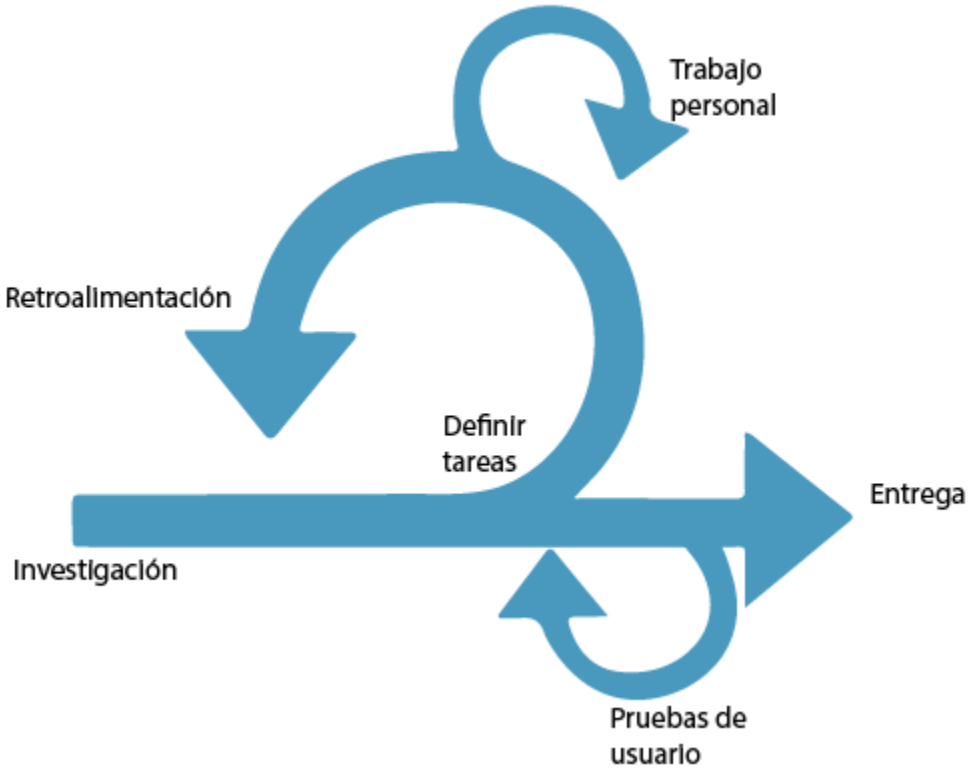

Figura 5 - Metodología de trabajo

En la primera etapa de investigación se buscaron ejemplos de las posibles interfaces que se podrían llegar a realizar para así encontrar que debía incluir la biblioteca, en donde se podría desarrollar y como se exporta.

A partir de esta investigación se definen tareas que dependiendo de su complejidad tomaban más o menos tiempo. La siguiente etapa de trabajo personal es el desarrollo de las tareas asignadas que una vez finalizadas deben ser reportadas para una retroalimentación.

En la retroalimentación se hace una reunión donde se analiza el resultado y adversidades encontradas en cada una de las tareas, para luego investigar o definir las siguientes tareas dependiendo del análisis realizado.

Este proceso se hace continuamente para ir avanzando y obteniendo resultados hasta llegar a la meta deseada.

Al final las pruebas de usuario se realizaron con ayuda de la Ingeniera en Multimedia Yuleisy Rincón. Se le pidió que realizara una pequeña aplicación, se le brindó el archivo de la biblioteca y el manual se usuario para que con base a ello pudiera dar una retroalimentación respecto al funcionamiento de estos elementos.

BIBLIOTECA SENSORMOV

La biblioteca SensorMov está desarrollada en lenguaje nativo, facilitando la comunicación con el sistema operativo. Esto permite que se pueda combinar con otras funcionalidades, como las de comunicación, manejo de bases de datos, gestión de archivos, entre otras. Así mismo, está exportada en un formato que permitiría incluirla, con algunos ajustes, en otras plataformas de desarrollo, como *Processing for Android*.

¿QUÉ ES SENSORMOV?

Es una biblioteca de código abierto, desarrollada en lenguaje nativo, para dispositivos móviles con sistema operativo Android. Esta biblioteca reúne un conjunto de funcionalidades orientado a extender el uso de varios de los sensores del móvil (de allí su nombre), a través de objetos físicos denominados “tangibles”.

A continuación, se presenta un glosario de términos utilizados con frecuencia en este manual y en la biblioteca.

- **Datos crudos**: Se refiere a datos provenientes de los sensores, sin ningún tipo de preprocesamiento. Por ejemplo, la posición x,y de un toque sobre la pantalla táctil del móvil.
- ***MainActivity***: Es la actividad principal por defecto de Android. Todos los ejemplos presentados en esta biblioteca son realizados sobre el *MainActivity*, por lo que acá se refiere a una actividad ejemplo que hace uso de alguna funcionalidad de la biblioteca.
- **SensorMov:** Es el nombre de la biblioteca.
- **Tangibles:** Objeto con componentes tanto físicos como digitales. En este sentido, es un objeto físico que sirve como control para modificar el estado digital del sistema, o uno que representa físicamente la información digital.

¿PARA QUÉ SIRVE SENSORMOV?

SensorMov busca facilitar, a los desarrolladores, la creación de aplicaciones que utilicen los sensores que se encuentran embebidos en el dispositivo móvil. En particular, para extender su funcionalidad más allá de los límites físicos del dispositivo, a través de tangibles.

Con facilitar nos referimos a reducir el tiempo de desarrollo, proporcionando funcionalidades ya establecidas, pero facilitando su modificación y permitiendo la inclusión de otras funcionalidades del sistema operativo.

DISEÑO

Esta biblioteca se diseñó siguiendo una lógica orientada a objetos, dado que el lenguaje nativo de Android está basado en Java. Debido al limitado espacio de este documento se recomiendo ver el diagrama de clases que se presenta en el Anexo 2. Igualmente se presentará continuación (Ver figura 6):

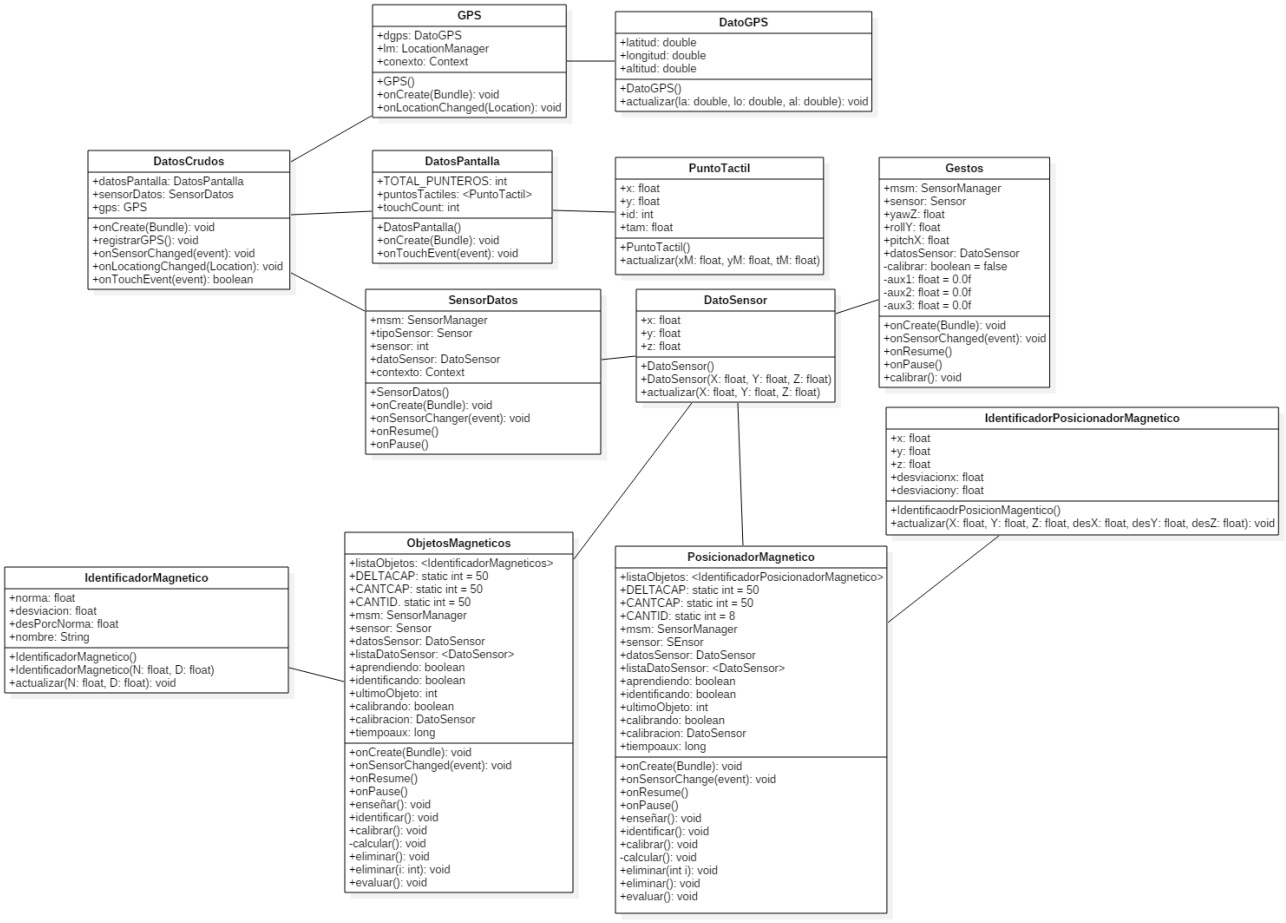

Figura 6 – Diagrama de Clases

Sensormov funciona por herencia, esto quiere decir que las clases principales heredan de la actividad *main* brindada al inicio de la creación de un proyecto en *AndroidStudio* algunos métodos como lo son el *OnCreate* y el *context* de dicha actividad para poder acceder a los sensores que se utilizarán.

Cada una de clases se explican detalladamente más adelante.

REQUISITOS

REQUISITOS FUNCIONALES

1. La biblioteca debe permitir capturar los datos del acelerómetro por cada eje.
2. La biblioteca debe permitir capturar los datos del giroscopio por cada eje.
3. La biblioteca debe permitir capturar los datos del magnetómetro por cada eje.
4. La biblioteca debe permitir capturar los datos del vector de rotación por cada eje.

5. La biblioteca debe permitir capturar provenientes de la pantalla: posición horizontal, posición vertical, presión, y área, según el orden en el que se activó el *touch*.
6. La biblioteca debe permitir la identificación de objetos, de acuerdo con su campo magnético.
7. La biblioteca debe permitir el posicionamiento de un objeto, en los tres ejes, según su campo magnético.
8. La biblioteca debe proveer el ángulo en grados en el que se encuentra el dispositivo móvil, en los tres ejes.

REQUISITOS NO FUNCIONALES

1. La biblioteca debe estar construida en el lenguaje nativo de Android.
2. La biblioteca debe estar orientada al uso de los sensores del móvil para interacción con objetos físicos, tangibles.
3. La biblioteca debe funcionar en Android desde la versión 18 – Jelly Bean 4.3
4. La biblioteca debe quedar en un repositorio, inicialmente privado, pero posteriormente con la posibilidad de hacerlo libre, bajo la licencia *Creative Commons*.
5. La biblioteca debe estar en español.
6. El código fuente de la biblioteca debe estar comentado y los comentarios deben estar en español.
7. La biblioteca debe incluir ejemplos para cada funcionalidad.
8. Se debe informar al usuario de la biblioteca, sobre la ocurrencia de eventos o acciones, a través de bitácoras (LOG) informativas.

RESULTADOS

A continuación, se explican las clases que se encuentran dentro de la biblioteca, su objetivo, atributos y métodos. Sobre cada una de las clases se incluyeron mensajes informativos a través de bitácoras (LOG) informativas o de ocurrencia de errores, que le permiten saber la ocurrencia de eventos, acciones, o incluso errores, cuando ocurra alguna excepción.

Formato de las bitácoras: (Clase: función, mensaje o valores).

CLASES

DatosCrudos

DatosCrudos es la clase principal y permite utilizar las otras tres que se encargan de suministrar los datos crudos provenientes de los sensores (SensorDatos, Datos Pantalla y GPS). La Tabla 2 presenta los atributos y métodos de esta clase.

Tabla 2 – Atributos y métodos DatosCrudos

Atributos	
+DatosPantalla datosPantalla	Objeto DatosPantalla para poder acceder a las funciones de esta clase.
+SensorDatos sensorDatos:	Objeto SensorDatos para poder acceder a las funciones de esta clase.
+GPS gps:	Objeto GPS para poder acceder a las funciones de esta clase.
Métodos	
`+onCreate (Bundle):void`	Método que permite la contiene todas las inicializaciones que se utilizaran en el MainActivity para las funciones de esta clase.
`-registrarGPS(): void`	Método que activa el GPS.
`-registrarSensor():void`	Método que activa el uso del sensor elegido (acelerómetro, giroscopio, magnetómetro, vector de rotación).
`+onSensorChanged (event): void`	Método del SensorEventListener que permite el registro del sensor.

+onLocationChanged (Location): void	Método del LocationListener que permite el registro de los datos brindados por dicho sensor.
+onTouchEvent(event): Boolean	Método de AndroidStudio que permite el registro de los datos brindados por la pantalla.

SensorDatos

SensorDatos es la clase que se encarga de obtener los valores brindados por los sensores que se van a utilizar (acelerómetro, giroscopio, magnetómetro, vector de rotación). La Tabla 3 presenta los atributos y métodos de esta clase.

Tabla 3 – Atributos y métodos SensorDatos

Atributos:	
+SensorManager msm	Objeto de la clase abstracta que permite el uso del Sensor
+Sensor tipoSensor	Objeto que representa el sensor.
+int sensor	Variable utilizada para escoger el sensor que se va a utilizar.
+DatoSensor datosSensor	Objeto tipo DatosEsnor para almacenar variables.
-Context contexto	Representa un identificador para obtener datos del entorno.
Métodos:	
SensorDatos()	Constructor del objeto.
+onCreate (Bundle): void	Método que permite la contiene todas las inicializaciones que se utilizaran en el MainActivity para las funciones de esta clase.

+onSensorChanged (event): void	Método del SensorEventListener que permite el registro del sensor.
+onResume: void	Método del SensorEventListener que permite el registro del sensor cuando la aplicación esta activa.
+onPause: void	Método del SensorEventListener que pausa el registro del sensor cuando la aplicación esta inactiva.

DatoSensor

Clase creada para el manejo de los 3 valores provenientes del sensor en un solo objeto. La Tabla 4 presenta los atributos y métodos de esta clase.

Tabla 4 – Atributos y métodos DatoSensor

Atributos:	
+float x	Variable que almacena los valores del eje X del sensor.
+float y	Variable que almacena los valores del eje Y del sensor.
+float z	Variable que almacena los valores del eje Z del sensor.
Métodos:	
DatoSensor ()	Constructor del objeto que inicializa las variables en 0.
DatoSensor (float X, float Y, floatZ)	Constructor del objeto que inicializa las variables con los valores recibidos.
+actualizar (float X, float Y, float Z) : void	Actualiza las variables del objeto con los valores recibidos.

DatosPantalla

Clase que se encarga de obtener los valores brindados por la pantalla. La Tabla 5 presenta los atributos y métodos de esta clase.

Tabla 5 – Atributos y métodos DatosPantalla

Atributos:	
`+int TOTAL_PUNTEROS`	Cantidad máxima de puntos activos al mismo tiempo (10).
`+<PuntoTactil > puntosTactiles.`	Lista para almacenar los objetos PuntoTactil que se van creando.
`+int touchCount`	Cantidad de puntos activos.
Métodos	
DatosPantalla()	Constructor
+onCreate(Bundle): void	Método que permite la contiene todas las inicializaciones que se utilizaran en el MainActivity para las funciones de esta clase.
+onTouchEvent(event): boolean	Método de AndroidStudio que permite el registro de los datos brindados por la pantalla.

PuntoTactil

Clase creada para el manejo de los valores provenientes de la pantalla en un solo objeto. La Tabla 6 presenta los atributos y métodos de esta clase.

Tabla 6 – Atributos y métodos PuntoTactil

Atributos:	
`+float x`	Posición en el eje x.
`+float y`	Posición en el eje y.
`+int id`	Orden en el que se active el evento.
Métodos:	

`PuntoTactil()`	Constructor del objeto que inicializa la variable id en -1.
`+actualizar` (float xl, float yl, int i, float y): void.	Actualiza las variables del objeto con los valores recibidos.

DatoGPS

Clase creada para el manejo de los 3 valores provenientes del gps en un solo objeto. La Tabla 7 presenta los atributos y métodos de esta clase.

Tabla 7 – Atributos y métodos DatoGPS

Atributos:	
`+double latitud`	Variable para la latitud.
`+double longitud`	Variable para la longitud.
`+double altitude`	Variable para la altitud.
Métodos:	
`DatoGps()`	Constructor del objeto que inicializa las variables en 0.
`+actualizar(double la, double lo, double al): void`	Actualiza las variables del objeto con los valores recibidos.

GPS

Clase que se encarga de obtener los valores brindados por el gps. La Tabla 8 presenta los atributos y métodos de esta clase.

Tabla 8 – Atributos y métodos GPS

Atributos:	
`+DatoGps dgps`	Objeto tipo Datogps para almacenar las variables.

+LocationManager lm	Variable para acceder a los servicios de localización del dispositivo.
-Context context contexto	Representa un identificador para obtener datos del entorno.
Métodos	
GPS()	Constructor del objeto.
+onCreate (Bundle): void	Método que permite la contiene todas las inicializaciones que se utilizaran en el MainActivity para las funciones de esta clase
+onLocationChanged (Location): void	Método del LocationListener que permite el registro de los datos brindados por dicho sensor.

Gestos

Esta clase permite obtener el ángulo en grados en el que se encuentra el celular por cada uno de los tres ejes utilizando solamente el sensor vector de rotación. La Tabla 9 presenta los atributos y métodos de esta clase.

Tabla 9 – Atributos y métodos Gestos

Atributos:	
+ SensorManager msm	Objeto de la clase abstracta que permite el uso del Sensor.
+ Sensor sensor	Objeto que representa el sensor.
+ float yawZ	Ángulo el eje Z.
+ float rollY	Ángulo el eje Y.
+ float pitchX	Ángulo el eje X.
+DatoSensor datosSensor	Objeto tipo DatoSensor para almacenar.
- boolean calibrar	Variable para activar la calibración.

- float aux1	Variable auxiliar para la calibración eje X.
- float aux2	Variable auxiliar para la calibración eje Y.
- float aux3	Variable auxiliar para la calibración eje Z.
Métodos:	
`+onCreate (Bundle): void`	Método que permite la contiene todas las inicializaciones que se utilizaran en el MainActivity para las funciones de esta clase.
`+onSensorChanged (event): void`	Método del SensorEventListener que permite el registro del sensor.
`+onResume: void`	Método del SensorEventListener que permite el registro del sensor cuando la aplicación esta activa.
`+onPause: void`	Método del SensorEventListener que pausa el registro del sensor cuando la aplicación esta inactiva.
`+calibrar ()`	Método que activa la calibración.

ObjetosMagneticos

Clase que permite identificar distintos imanes utilizando solamente la norma del vector proveniente de cada imán. La Tabla 10 presenta los atributos y métodos de esta clase.

Tabla 10 – Atributos y métodos ObjetosMagneticos

Atributos:	
`+<IdentificadorMagnetico> listaObjetos`	Lista de objetos IdentificadorMagnetico.

-static int DELTACAP=50	Milisegundos entre cada muestra tomada.
-static int CANTCAP=50	Cantidad de datos almacenados al enseñar.
-static int CANTID=50	Cantidad de datos almacenados al identificar.
-Sensormanager msm	Objeto de la clase abstracta que permite el uso del Sensor.
-Sensor sensor	Objeto que representa el sensor.
-DatoSensor datosSensor	Objeto tipo DatoSensor para almacenar.
+<DatoSensor> listaDatoSensor	Lista de objetos tipo DatoSensor para calcular norma y desviación.
-boolean aprendiendo	Variable para activar el aprendizaje.
-boolean identificando	Variable para activar la identificación.
+int ultimoObjeto	Variable que indica la ubicación en el arreglo del objeto identificando.
-boolean calibrando	Variable para activar la calibración.
-DatoSensor calibración	Objeto tipo DatoSensor para almacenar datos que se utilizan en la calibración.
-long tiempoaux	auxiliar para el control del muestro.
Métodos:	
+onCreate (Bundle): void	Método que permite la contiene todas las inicializaciones que se utilizaran en el MainActivity para las funciones de esta clase.
+onSensorChanged (event): void	Método del SensorEventListener que permite el registro del sensor.

+onResume: void	Método del SensorEventListener que permite el registro del sensor cuando la aplicación esta activa.
+onPause: void	Método del SensorEventListener que pausa el registro del sensor cuando la aplicación esta inactiva.
+enseñar(): void	Método para enseñar un nuevo imán a la aplicación.
+identificar(): void	Método para identificar un imán en la aplicación.
+calibrar(): void	Método para activar la calibración la aplicación.
-calcular(): void	Método para calcular las normas y desviaciones de los imanes.
+eliminar(): void	Método para eliminar el último elemento de la lista de IdentificadorMagnetico.
+eliminar(int i):	Método para el elemento deseado de la lista de IdentificadorMagnetico.
+evaluar(): void	Método para evaluar usando las normas y desviaciones de los imanes para escoger que elemento de la lista.

IdentificadorMagnetico

Clase creada para el manejo de las variables usadas para la identificación en ObjetosMagneticos en un solo objeto. La Tabla 11 presenta los atributos y métodos de esta clase.

Tabla 11 – Atributos y métodos IdentificadorMagnetico

Atributos:	
+float norma	Norma del vector obtenido de las muestras.

+float desviacion	Desviación del vector obtenido de las muestras.
+float desPorcNorma	Desviación porcentual del vector obtenido de las muestras.
+String:	Nombre el imán.
Métodos:	
+IdentificadorMagnecito()	Constructor del objeto que inicializa los atributos en 0.
+IdentificadorMagnecito(float N, float D)	Constructor del objeto que inicializa los atributos con los parámetros recibidos.
+actualizar(float N, float D): void	Actualiza las variables del objeto con los valores recibidos.

PosicionadorMagnetico

Clase que permite identificar la posición de un imán. Para esta clase el método de identificación se utiliza un promedio de los datos obtenidos por cada eje, estos promedios cuentan con su desviación estándar que es utilizada como el criterio a evaluar. La Tabla 12 presenta los atributos y métodos de esta clase. Se escogió basado en la figura 7 ya que cada posición es perfectamente diferenciable, viendo claramente las cuatro nubes de puntos.

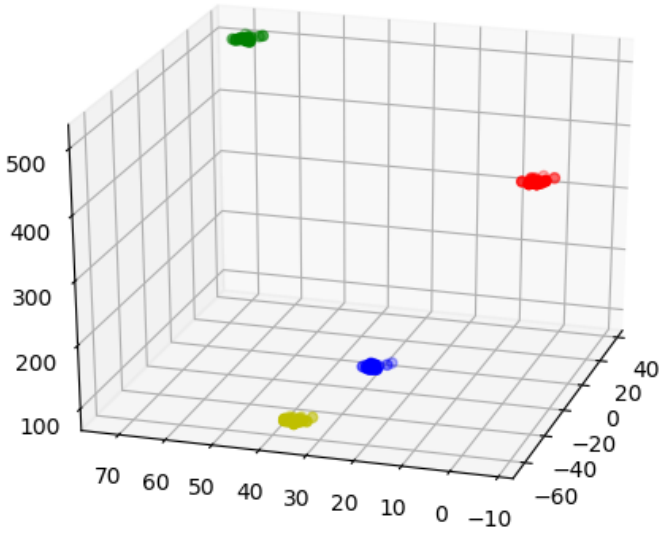

Figura 7 – Datos de cuatro posiciones de un imán.

Tabla 12 – Atributos y métodos PosicionadorMagnetico

Atributos:	
`+<IdentificadorPosicionadorMagnetico> listaObjetos`	Lista de objetos IdentificadorPosicionadorMagnetico.
`-static int DELTACAP=50`	Milisegundos entre cada muestra tomada.
`-static int CANTCAP=50`	Cantidad de datos almacenados al enseñar.
`-static int CANTID=50`	Cantidad de datos almacenados al identificar.
`-static int CANTDESV=8`	Cantidad de desviaciones para la identificación.
`-Sensormanager msm`	Objeto de la clase abstracta que permite el uso del Sensor.
`-Sensor sensor`	Objeto que representa el sensor.

-DatoSensor datosSensor	Objeto tipo DatoSensor para almacenar.
+<DatoSensor> listaDatoSensor	Lista de objetos tipo DatoSensor para calcular promedio y desviación.
-boolean aprendiendo	Variable para activar el aprendizaje.
-boolean identificando	Variable para activar la identificación.
+int ultimoObjeto	Variable que indica la ubicación en el arreglo del objeto identificando.
-boolean calibrando	Variable para activar la calibración.
-DatoSensor calibración	Objeto tipo DatoSensor para almacenar datos que se utilizan en la calibración.
-long tiempoaux Variable	Auxiliar para el control del muestro.
Métodos:	
+onCreate (Bundle): void	Método que permite la contiene todas las inicializaciones que se utilizaran en el MainActivity para las funciones de esta clase.
+onSensorChanged (event):	Método del SensorEventListener que permite el registro del sensor.
+onResume: void	Método del SensorEventListener que permite el registro del sensor cuando la aplicación esta activa.
+onPause: void	Método del SensorEventListener que pausa el registro del sensor cuando la aplicación esta inactiva.
+enseñar(): void	Método para enseñar un nuevo imán a la aplicación.

+identificar(): void.	Método para identificar un imán en la aplicación.
+calibrar(): void	Método para activar la calibración la aplicación.
-calcular(): void	Método para calcular los promedios de las muestras y desviaciones de los imanes.
+eliminar(): void	Método para eliminar el último elemento de la lista de IdentificadorPosicionMagnetico.
+eliminar(int i):	Método para el elemento deseado de la lista de IdentificadorPosicionMagnetico.
+evaluar(): void	Método para evaluar usando el valor de cada eje y desviaciones de los imanes para escoger que elemento de la lista.

IdentificadorPosicionadorMagnetico

Clase creada para el manejo de las variables usadas para la identificación en PosicionadorMagnetico en un solo objeto. La Tabla 13 presenta los atributos y métodos de esta clase.

Tabla 13 – Atributos y métodos IdentificadorPosicionadorMagnetico

Atributos:	
+float x	Variable del valor obtenido en el eje X.
+float y	Variable del valor obtenido en el eje Y.
+float z	Variable del valor obtenido en el eje Z.
+float desviacionx	Variable de la desviación obtenida en el eje X.

+float desviaciony	Variable de la desviación obtenida en el eje Y.
+float desviacionz	Variable de la desviación obtenida en el eje Z.
Métodos:	**Métodos:**
+IdentificadorPosicionMagnecito()	Constructor del objeto.
+actualizar(float X, float Y, float Z, float desX, float desY, float desZ): void	Actualiza las variables del objeto con los valores recibidos.

EJEMPLOS

Esta subsección presenta algunos ejemplos sencillos de cómo utilizar las diferentes funcionalidades para cada una de las clases. El Código 1 presenta el ejemplo de DatosCrudos para GPS.

```
Package com.umng.tangibles;

import android.content.Context;
import android.location.LocationManager;
import android.os.Build;
import android.os.Bundle;
import android.widget.EditText;
//Importar la biblioteca x
import com.umng.sensormov.DatosCrudos;


import androidx.annotation.RequiresApi;
public class MainActivity extends DatosCrudos {
    EditText cuadroTexto;
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //Creamos el objeto que se usa para el gps
        LocationManager lm=(LocationManager)getSystemService(Context.LOCATION_SERVICE);
        setContentView(R.layout.activity_main);
        cuadroTexto = findViewById(R.id.cuadroTextoid);
        String mes = "";
        //Llamamos el objeto que se usa para el gps
        mes = "Latitud: " + gps.dgps.latitud;
        cuadroTexto.setText(mes);
```

```
    }
}
```

Código 1 Código fuente de la MainActivity ejemplo de la funcionalidad de GPS utilizando Datos Crudos

Más información Aquí, para la versión digital de este manual, o en el QR, para la versión impresa.

DatosCrudos

Este ejemplo despliega en un campo de texto el valor de latitud brindada por el GPS. **Nota:** Recuerda revisar si la aplicación creada si tiene los permisos necesarios para acceder a la ubicación del dispositivo

Este ejemplo despliega en un campo de texto el valor del campo magnético en el eje x. El Código 2 presenta el ejemplo de DatosCrudos para SensorDatos.

```
package com.umng.tangibles;
import android.hardware.SensorEventListener;
import android.os.Build;
import android.os.Bundle;
import android.widget.EditText;

//Importar las bibliotecas necesarias
import android.hardware.SensorEvent;
import com.umng.sensormov.DatosCrudos;

import androidx.annotation.RequiresApi;
public class MainActivity extends DatosCrudos implements SensorEventListener {
    EditText cuadroTexto;
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        sensorDatos.sensor=2; //Iniciamos el sensor magnetometro o brujula
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        cuadroTexto = findViewById(R.id.cuadroTextoid);
    }
    @Override
   //Método del SensorEventListener
    public void onSensorChanged(SensorEvent event) {
        super.onSensorChanged(event);
        String mes = "";
        //Llamamos el objeto que se usa para el sensor
        mes = "X: " + sensorDatos.datosSensor.x;
```

```
        cuadroTexto.setText(mes);
    }
```

Código 2 Código fuente de la MainActivity ejemplo de la funcionalidad SensorDatos utilizando Datos Crudos

Más información Aquí, para la versión digital de este manual, o en el QR, para la versión impresa.

Este ejemplo despliega en un campo de texto los valores de la posición x,y al momento de realizar un touch en la pantalla. El Código 3 presenta el ejemplo de DatosCrudos para DatosPantalla.

```
package com.umng.tangibles;
import android.os.Build;
import android.os.Bundle;
import android.widget.EditText;
import android.util.Log;
import androidx.annotation.RequiresApi;

//Importar las bibliotecas necesarias
import com.umng.sensormov.DatosCrudos;
import android.view.MotionEvent;
public class MainActivity extends DatosCrudos {
    EditText cuadroTexto;
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        cuadroTexto = findViewById(R.id.cuadroTextoid);
    }
    @Override
    //Método de MotionEvent
    public boolean onTouchEvent(MotionEvent event) {
        super.onTouchEvent(event);
        Log.i("Main", "Action "+ event.getActionMasked()+": "+event.getActionIndex());
        String mes="";
    //Utilizamos esta parte para detectar los touch, la posición xy y el orden en el que
están
        for(int t=0;t<datosPantalla.TOTAL_PUNTEROS;t++)
            if(datosPantalla.puntosTactiles[t].id!=-1)
                mes=mes+"                "+datosPantalla.puntosTactiles[t].id+      ":
x="+datosPantalla.puntosTactiles[t].x+",  y="+datosPantalla.puntosTactiles[t].y+",  tam:
"+datosPantalla.puntosTactiles[t].tam+"\n";
        cuadroTexto.setText(mes);
        return true;
```

```
    }
}
```

Código 3 Código fuente de la MainActivity ejemplo de la funcionalidad datosPantanlla utilizando Datos Crudos.

Más información Aquí, para la versión digital de este manual, o en el QR, para la versión impresa.

Datos Pantalla

Este ejemplo despliega en un campo de texto los valores de la posición x,y al momento de realizar un touch en la pantalla. El Código 4 presenta el ejemplo de DatosPantalla

```
package com.umng.tangibles;
import android.os.Build;
import android.os.Bundle;
import android.widget.EditText;
import android.util.Log;
import androidx.annotation.RequiresApi;
//Importar las bibliotecas necesarias
import android.view.MotionEvent;
import com.umng.sensormov.DatosPantalla;


public class MainActivity extends DatosPantalla {
    EditText cuadroTexto;
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        toques = findViewById(R.id.cuadroTextoid);
    }
    @Override
//Método de MotionEvent
    public boolean onTouchEvent(MotionEvent event) {
        super.onTouchEvent(event);
        Log.i("Main", "Action "+ event.getActionMasked()+": "+event.getActionIndex());
        String mes="";
//Utilizamos esta parte para detectar los touch, la posición xy y el orden en el que
están
        for(int t=0;t<TOTAL_PUNTEROS;t++)
            if(puntosTactiles[t].id!=-1)
```

```
                mes=mes+"       "+puntosTactiles[t].id+   ":   x="+puntosTactiles[t].x+",
y="+puntosTactiles[t].y+", tam: "+puntosTactiles[t].tam+"\n";
        cuadroTexto.setText(mes);
        return true;
    }
}
```

Código 4 Código fuente de la MainActivity ejemplo de la funcionalidad datosPantanlla

Más información Aquí, para la versión digital de este manual, o en el QR, para la versión impresa.

Sensor Datos

Este ejemplo despliega en un campo de texto el valor del campo magnético en el eje x. El Código 5 presenta el ejemplo de SensorDatos.

```
package com.umng.tangibles;
import android.hardware.SensorEventListener;
import android.os.Build;
import android.os.Bundle;
import android.widget.EditText;
//Importar las bibliotecas necesarias
import android.hardware.SensorEvent;
import com.umng.sensormov.SensorDatos;

import androidx.annotation.RequiresApi;
public class MainActivity extends SensorDatos implements SensorEventListener {
    EditText cuadroTexto;
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        sensor=2; // Iniciamos el sensor magnetometro o brujula
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        cuadroTexto = findViewById(R.id.cuadroTextoid);
    }
    @Override
//Método del SensorEventListener
    public void onSensorChanged(SensorEvent event) {
        super.onSensorChanged(event);
        String mes = "";
//Llamamos el objeto que se usa para el sensor
        mes = "X: " + datosSensor.x;
        cuadroTexto.setText(mes);
```

```
    }
}
```

Código 5 Código fuente de la MainActivity ejemplo de la funcionalidad SensorDatos.

Más información Aquí, para la versión digital de este manual, o en el QR, para la versión impresa.

GPS

Este ejemplo despliega en un campo de texto el valor de latitud brindada por el GPS.

El Código 6 presenta el ejemplo de GPS.

Nota: Recuerda revisar si la aplicación creada si tiene los permisos necesarios para acceder a la ubicación del dispositivo

```
package com.umng.tangibles;

import android.content.Context;
import android.location.LocationManager;
import android.os.Build;
import android.os.Bundle;
import android.widget.EditText;
import androidx.annotation.RequiresApi;
//Importar la biblioteca
import com.umng.sensormov.GPS;

public class MainActivity extends GPS {
    EditText cuadroTexto;
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
//Creamos el objeto que se usa para el gps
        LocationManager lm=(LocationManager)getSystemService(Context.LOCATION_SERVICE);
        setContentView(R.layout.activity_main);
        cuadroTexto = findViewById(R.id.cuadroTextoid);
        String mes = "";
//Llamamos el objeto que se usa para el gps
        mes = "Latitud: " + dgps.latitud;
        cuadroTexto.setText(mes);
    }
}
```

Código 6 Código fuente de la MainActivity ejemplo de la funcionalidad de GPS

Más información Aquí, para la versión digital de este manual, o en el QR, para la versión impresa.

Gestos

A continuación, se muestra una aplicación que permite obtener el valor del ángulo en grados en el eje z del celular. El Código 7 presenta el ejemplo de Gestos.

```
package com.umng.tangibles;
import android.os.Build;
import android.os.Bundle;
import android.widget.EditText;
import androidx.annotation.RequiresApi;
//Importar la biblioteca
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import com.umng.sensormov.Gestos;
public class MainActivity extends Gestos implements SensorEventListener {
    EditText cuadroTexto;
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        cuadroTexto = findViewById(R.id.cuadroTextoid);
        calibrar();
    }
    @Override
//Método del SensorEventListener
    public void onSensorChanged(SensorEvent event) {
        super.onSensorChanged(event);
        String mes ="";
//Llamamos el objeto que tiene el ángulo en el eje z
        mes ="yawZ: "+ yawZ;
        cuadroTexto.setText(mes);
    }
```

Código 7 Código fuente de la MainActivity ejemplo de la funcionalidad de GPS

Más información Aquí, para la versión digital de este manual, o en el QR, para la versión impresa.

Objetos Magnéticos

A continuación, se muestra una aplicación que permite obtener enseñarle a reconocer imanes y mostrar una imagen de acuerdo con cada uno de ellos. El Código 8 presenta el ejemplo de ObjetosMangeticos.

```
package com.umng.tangibles;
import static android.view.View.*;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
//Importar la biblioteca
import com.umng.sensormov.ObjetosMagneticos;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;

public class MainActivity extends ObjetosMagneticos implements SensorEventListener {
//imágenes que se utilzian en el ejemplo
    ImageView jim;
    ImageView tom;
    ImageView hercules;
    Button enseñarBoton;
    Button calibrarBoton;
    Button identificarBoton;
    Button elimnarBoton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        jim = findViewById(R.id.jimid);
        tom = findViewById(R.id.tomid);
        hercules = findViewById(R.id.herculesid);
        enseñarBoton= (Button)findViewById(R.id.enseñarid);
        calibrarBoton= (Button) findViewById(R.id.calibrarid);
        identificarBoton=(Button)findViewById(R.id.indentificarid);
        elimnarBoton=(Button)findViewById(R.id.eliminarid);
//funciones de los botones
        enseñarBoton.setOnClickListener(new OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
//llamar método para enseñar
                enseñar();
            }
        });
        calibrarBoton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
//llamar método para calibrar
                calibrar();
            }
        });
        identificarBoton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
//llamar método para identificar
                identificar();
            }
        });
        elimnarBoton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
//llamar método para eliminar
                eliminar();
            }
        });
    }
    @Override
//Método del SensorEventListener
    public void onSensorChanged(SensorEvent event) {
        super.onSensorChanged(event);
        Log.i("MAIN: ","Ultimo Objeto"+ultimoObjeto);
//Validar el valor retornado por la biblioteca
        if (ultimoObjeto==0){
            jim.setVisibility(VISIBLE);
            tom.setVisibility(INVISIBLE);
            hercules.setVisibility(INVISIBLE);
        } else {
            if (ultimoObjeto==1){
                jim.setVisibility(INVISIBLE);
                tom.setVisibility(VISIBLE);
                hercules.setVisibility(INVISIBLE);
            } else {
                if (ultimoObjeto==2) {
                    jim.setVisibility(INVISIBLE);
                    tom.setVisibility(INVISIBLE);
                    hercules.setVisibility(VISIBLE);
                }
            }
        }
    }
}
```

Código 8 Código fuente de la MainActivity ejemplo de la funcionalidad de Objetos Magneticos

Más información Aquí, para la versión digital de este manual, o en el QR, para la versión impresa.

Posicionador Magnético

A continuación, se muestra una aplicación que permite obtener enseñarle a reconocer la posición de unos imanes y mostrarla en un cuadro de texto. El Código 9 presenta el ejemplo de PosicionadorMagnetico.

```
package com.umng.tangibles;
import static android.view.View.*;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
//Importar la biblioteca
import com.umng.sensormov.PosicionadorMagnetico;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;


public class MainActivity extends PosicionadorMagnetico implements SensorEventListener
{

    Button enseñarBoton;
    Button calibrarBoton;
    Button identificarBoton;
    TextView texto;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        enseñarBoton= (Button)findViewById(R.id.enseñarid);
        calibrarBoton= (Button) findViewById(R.id.calibrarid);
        identificarBoton=(Button)findViewById(R.id.identificarid);
        texto=(TextView)findViewById(R.id.textid);
//funciones de los botones
        enseñarBoton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                enseñar();
//llamar método para enseñar
            }
```

```
        });
        calibrarBoton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
//llamar método para calibrar
                calibrar();
            }
        });
        identificarBoton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
//llamar método para identificar
                identificar();
            }
        });
    }
    @Override
//Método del SensorEventListener
    public void onSensorChanged(SensorEvent event) {
        super.onSensorChanged(event);
//Validar el valor retornado por la biblioteca
       Log.i("MAIN: ","Ultimo Objeto"+ultimoObjeto);
        if (ultimoObjeto==1){
            texto.setText("posicion 1");
        } else {
            if (ultimoObjeto==2){
                texto.setText("posicion 2");
            } else {
                if (ultimoObjeto==3) {
                    texto.setText("posicion 3");
                }else{
                    texto.setText("no identificada");
                }
            }
        }
    }
}
```

Código 9 Código fuente de la MainActivity ejemplo de la funcionalidad de PosicionadorMagnetico

Más información Aquí, para la versión digital de este manual, o en el QR, para la versión impresa.

Para más información ver Anexos 1-4.

PRUEBAS DE USUARIO

Las pruebas de usuario se ejecutaron con ayuda de la Ingeniera en Multimedia Yuleisy Rincón egresada de la Universidad Militar Nueva Granada.

Se le pidió que realizara un “Hola mundo” en *AndroidStudio* para que descargara el programa y los archivos pertinentes para poder exportar aplicaciones, esto con el propósito de que se familiarizara con el entorno de desarrollo puesto que ella no tenía experiencia previa en este medio, además, para ahorrar tiempo ya que el programa puede ser pesado y demorado al descargar.

Después se le pido que hiciera una pequeña aplicación, se le brindó el archivo de la biblioteca y el manual se usuario para que con base a ello pudiera dar una retroalimentación respecto al funcionamiento de estos elementos.

La aplicación desarrollada consistió en mostrar el circulo de colores al girar el celular (ver Anexo 5), haciendo uso del sensor vector de rotación, esta prueba se realizó en menos de una semana, y los comentarios recibidos se dividen en dos secciones: Respecto al manual de usuario y sobre la biblioteca en general.

1. Manual de Usuario

Es bastante claro y conciso, explica el objetivo de la biblioteca y sus posibles usos de una manera fácil de entender. El proceso de instalación es sencillo y el manual muestra claramente paso a paso de una manera gráfica.

Sería bueno tener un poco más de descripción en el uso de algunas clases y en qué casos usarlas, por ejemplo, la diferencia entre usar la clase Gestos en lugar de DatoSensor.

Los ejemplos son bastante útiles y en general se entienden claramente, sin embargo, se usan los mismos nombres de variables, por ejemplo,“toques”; cuando

se están usando datos referentes a ángulos y no a toques en la pantalla, lo cual puede llegar a ser confuso.

2. Uso de la biblioteca

Cualquier persona familiarizada con el lenguaje de programación de Java y con la ayuda de la documentación, puede usar fácilmente la biblioteca. Es bastante práctica y simplifica mucho el uso de los sensores de los dispositivos móviles, ya que facilita el tener que manejar los datos crudos de los sensores.

Después de recibir esto la única mejora que se realizo fueron el cambio de nombres de variables a unas más pertinentes en cada uno de los ejemplos.

CONCLUSIONES

SensorMov se desarrolló como una versión inicial de una biblioteca orientada a la creación de interfaces físicas artesanales. Se cumplió con los requisitos funcionales y no funcionales planteados para esta biblioteca, y que provienen de los diálogos de saberes que ocurrieron entre artesanos textiles y estudiantes de Ingeniería en Multimedia, en el marco del proyecto “Diálogos de saberes en la creación de interfaces físicas artesanales para la interacción con dispositivos móviles”. La biblioteca también se encuentra en GitHub como posibilidad para dejarla como código abierto y uso libre para aumentar sus funcionalidades con ayuda de la comunidad de desarrolladores. Se espera también probar la utilidad de los manuales de instalación y de usuario, probando con diferentes desarrolladores, para así mismo mejorar la comunicación a través de estos documentos, teniendo en cuenta que esta evaluación no se alcanzó a realizar al momento de entregar este documento. Adicionalmente, se contempla la refactorización de todo el código para mejorar el rendimiento de la biblioteca.

Durante el desarrollo de Sensormov se descubrió que, aunque su objetivo principal es ser utilizada en el desarrollo de interfaces físicas artesanales, su potencial va más allá y se puede usar en otras interfaces tangibles. Esto hace que sea posible usarla en otros ámbitos de la universidad como semilleros de investigación, ej. Techcraft que se centra en interacciones tangibles, haciendo que la biblioteca pueda ser de utilidad para los estudiantes. Así mismo, al estar en un repositorio que será público, se espera que sea de utilidad en el desarrollo de aplicaciones móviles que hagan uso de interfaces tangibles, que promuevan un uso más extendido los teléfonos inteligentes y su potencial diversificación.

Desde mi experiencia en este proyecto de desarrollo tecnológico, una de las partes más difíciles fue definir las posibles interacciones que se podrían realizar con la biblioteca, y cómo configurar los datos de bajo nivel que provenían de los sensores para priorizar y realizar desarrollos escalonados. Así mismo, determinar cómo usar

los datos para convertirlos en funciones de alto nivel orientadas a crear interacciones tangibles con el móvil. Además, el desarrollo me permitió iniciar el aprendizaje de una nueva herramienta, *AndroidStudio,* ya que nunca había trabajado en ella.

A nivel personal el desarrollo de este proyecto me permitió profundizar y comprender un poco más diversas áreas de la Ingeniería en Multimedia, en especial las enfocadas en programación orientada a objetos y desarrollo móvil. No solamente practicar los conocimientos adquiridos, sino entendiendo su real importancia en casos muchos prácticos y cercanos a un mundo profesional. Además, dentro del trabajo en el proyecto “Diálogos de saberes en la creación de interfaces físicas artesanales para la interacción con dispositivos móviles”, fue una experiencia inolvidable que me brindo momentos con todos los miembros del proyecto, los artesanos con los que trabajamos en el proyecto y con los que visitamos. Esto me hacía salir de mi zona de confort haciendo que mejorara y adquiriera habilidades en campos más allá de lo profesional y académico.

TRABAJO FUTURO

El proyecto se encuentra en GitHub de manera privada porque se encuentra en proceso de registro. Se buscará dejarlo como software libre bajo una licencia Creative Commons, que permita que otros miembros de la comunidad de desarrolladores la complementen o modifiquen, de acuerdo con sus necesidades específicas. Así mismo se su busca ampliar funcionalidades de la biblioteca como: reconocer gestos mediante cambios en el campo magnético, reconocer gestos mediante el acelerómetro, giroscopio y la pantalla del celular. Además, es necesario realizar pruebas de la biblioteca con desarrolladores externos al desarrollo del proyecto para verificar y mejorar tanto los manuales de usuario e instalación como la biblioteca misma. También se quiere implementar la biblioteca en el desarrollo de nuevas interfaces físicas artesanales. Específicamente en la aplicación “Parchando Diálogos”, creada en el marco del proyecto “Diálogos de saberes en la creación de interfaces físicas artesanales para la interacción con dispositivos móviles”. Por último, lograr adaptar la biblioteca para que pueda ser usada en otros ambientes de desarrollo más allá de AndroidStudio, como pueden ser *processing* y *Unity*.

ANEXOS

ANEXO 1. Archivo de la biblioteca

Se adjunta el archivo sensormov-release.aar que puede ser importando a un proyecto de AndroidStudio

ANEXO 2. Diagrama de Clases

Se adjunta el archivo pdf llamado “Diagrama de clases”.

ANEXO 3. Manual de Usuario

Se incluye dentro de los archivos un pdf uno con el nombre “Manual de usuario” el cual contiene el manual de usuario desarrollado.

ANEXO 4. Manual de Instalación

Se incluye dentro de los archivos un pdf uno con el nombre “Manual de instalación” el cual contiene el manual de instalación desarrollado.

ANEXO 5. Videos de los ejemplos de prueba

Se incluye dentro de los archivos un archivo pdf con el nombre “Enlaces Videos” con los enlaces a los videos de ejemplo y de la prueba de usuario.

ANEXO 6. Enlace del proyecto en GitHub

Se incluye dentro de los archivos un archivo pdf

REFERENCIAS

[1] Phidgets.com. (2019). *Phidget Programming Basics - Phidgets Support*. [online] Disponible en: https://www.phidgets.com/docs/Phidget_Programming_

[2] Hitl.washington.edu. (2019). Human Interface Technology Lab - Shared Space Download Page.[online]Disponible en: http://www.hitl.washington.edu/research/shared_space/download/

[3] Kratz, S., Westermann, T., Rohs, M., & Essl, G. (2011). CapWidgets: Tangile Widgets Versus Multi-touch Controls on Mobile Devices. En CHI '11 Extended Abstracts on Human Factors in Computing Systems (pp. 1351–1356). New York, NY, USA: ACM. https://hci.uni-hannover.de/papers/kratz2011capwidgets.pdf

[4] Tuio.org. (2019). TUIO. [online] Disponible en: https://www.tuio.org/

[5] Hadjakos, Aristotelis & Waloschek, Simon. (2014). SPINE: A TUI Toolkit and Physical Computing Hybrid. 10.13140/2.1.2869.6646.

[6] Baraldi, S., Benini, L., Cafini, O., Del Bimbo, A., Farella, E., Gelmini, G., Landucci, L., Pieracci, A. and Torpei, N. (2008). Evolving TUIs with smart objects for multi-context interaction. 10.1145/1358628.1358790

[7] Hitl.washington.edu. (2019). *Human Interface Technology Lab - Shared Space* online] Disponible en: http://www.hitl.washington.edu/research/shared_space/download/

[8] Ishii, H. (2006). *Tangible User Interfaces* [online] Disponible en: http://www.cs.tufts.edu/~jacob/workshop/papers/ishii.pdf

[9] Locher, P. (2006). *Tangible User Interfaces – Classification* [online] Disponible en: https://diuf.unifr.ch/people/lalanned/Seminar/Seminar0506/TUI_classification.pdf

[10] GOOGLE DEVELOPER, «Descripción General de los Sensores,» 27 12 2019. [online]. Disponible en: https://developer.android.com/guide/topics/sensors/sensors_overview.

[11] Ketabdar, H., Haji-Abolhassani, A., Roshandel, M. (2012). MagiThings: Gestural Interaction with Mobile Devices Based on Using Embedded Compass (Magnetic Field) Sensor [online] Disponible en: https://www.researchgate.net/publication/264975598_MagiThings_Gestural_Interaction_with_Mobile_Devices_Based_on_Using_Embedded_Compass_Magnetic_Field_Sensor

[12] Teyssier, M., Bailly, G., Pelachaud, C., Lecolinet, E. (2018). MobiLimb: Augmenting Mobile Devices with a Robotic Limb [online] Disponible en: https://marcteyssier.com/content/publications/uist18-mobilimb-teyssier.pdf

[13] Zhang, X., Tran, T., Sun, Y., Culhane, I., Jain, S., Fogarty, J. Mankoff, J. (2018). Interactile: 3D Printed Tactile Interfaces to Enhance Mobile Touchscreen Accessibility. [online] Disponible en: https://homes.cs.washington.edu/~jfogarty/publications/assets2018-interactiles.pdf

[14] Song, SJ., Nam, H. (2018). Sound of Tapping user interface technology with medium identification, [online] Disponible en: https://www.sciencedirect.com/science/article/pii/S0141938217300549

[15] Locher, P. (2006). Tangible User Interfaces.[online] Disponible en: https://diuf.unifr.ch/people/lalanned/Seminar/Seminar0506/TUI_classification.pdf

[16] Ishii, H. (2008). The tangible user interface and its evolution [online] Disponible en: https://dl.acm.org/doi/fullHtml/10.1145/1349026.1349034