UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA DE INGENIERÍA INDUSTRIAL DE TOLEDO

TRABAJO FIN DE GRADO Nº 18-B-225203

ITANIUM: Interfaz tangible de bajo coste para el desarrollo de capacidades en niños de corta edad

Autor:
Rafael Suárez-Bustamante Pica

Director:
Francisco Moya Fernández

Septiembre de 2018

ITANIUM: Interfaz tangible de bajo coste para el desarrollo de capacidades en niños de corta edad

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA DE INGENIERÍA INDUSTRIAL DE TOLEDO

TRABAJO FIN DE GRADO Nº 18-B-225203

ITANIUM: Interfaz tangible de bajo coste para el desarrollo de capacidades en niños de corta edad

Autor:
Rafael Suárez-Bustamante Pica

Director:
Francisco Moya Fernández

Septiembre de 2018

Rafael Suárez-Bustamante Pica

Toledo – Spain

E-mail: Rafael.Suarezbustamante@alu.uclm.es
Teléfono: 685966456

© 2018 Rafael Suárez-Bustamante Pica

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Se permite la copia, distribución y/o modificación de este documento bajo los términos de la Licencia de Documentación Libre GNU, versión 1.3 o cualquier versión posterior publicada por la *Free Software Foundation*; sin secciones invariantes. Una copia de esta licencia esta incluida en el apéndice titulado «GNU Free Documentation License».

Muchos de los nombres usados por las compañías para diferenciar sus productos y servicios son reclamados como marcas registradas. Allí donde estos nombres aparezcan en este documento, y cuando el autor haya sido informado de esas marcas registradas, los nombres estarán escritos en mayúsculas o como nombres propios.

Resumen

Este Trabajo Fin de Grado consiste en el diseño y desarrollo de una plataforma interactiva de juego de bajo coste, basado en interfaces tangibles para aplicaciones lúdico-educativas, empleando sistemas empotrados basados en microprocesadores, junto a un conjunto de periféricos, aprovechando la tecnología para facilitar los mecanismos de aprendizaje.

En esta aplicación educativa, más que construir y diseñar un juguete, se intenta explotar la tecnología para facilitar los mecanismos de aprendizaje.

Actualmente se ha demostrado que los sistemas tangibles ofrecen una serie de beneficios en la educación y la creatividad. La finalidad es fomentar este sistema en aplicaciones educativas, combinando para este propósito las mejoras de los medios digitales utilizando entornos visuales, animaciones y sonidos.

Se han analizado los distintos enfoques y la tecnología aplicada en los actuales sistemas de programación para niños mediante interacción tangible, mostrando los resultados obtenidos en estudios experimentales aplicados a la interacción tangible, y cómo mejoran las habilidades computacionales de los niños.

El sistema de programación tangible diseñado en este proyecto, es un sistema electrónico con un entorno gráfico simple, que permite al niño ser capaz de diseñar secuencias y modificarlas interactuando con elementos tangibles.

Está formado por dos elementos principales. Ambos dispositivos disponen de una pantalla táctil donde se desarrolla el juego. El intercambio de información se realiza utilizando comunicaciones inalámbricas *WiFi*, por lo que son totalmente independientes un dispositivo del otro.

El empleo de un sensor de color y una unidad de medición inercial, son herramientas que ofrecen la posibilidad al niño, de interactuar con el medio que les rodea, mejorando la experiencia de juego.

La idea principal parte de que los niños no sean simplemente consumidores de contenido y que tengan la posibilidad de crearlo, que ellos mismos sean capaces de tomar decisiones durante el juego de una manera tangible e intuitiva.

Agradecimientos

A mi familia, por todo el ánimo y preocupación que han mostrado por mi.

A mis amigos, en especial a Rox", por los consejos durante el Trabajo Fin de Grado.

A todas aquellas personas que han pasado por mi vida en estos años y han influido en mí ya que todas, en mayor o menor medida, tienen un hueco en mi memoria.

Rafa

A mi padre. Que allá donde estés, puedas sentirte orgulloso de mi.
A mi hermano Pablo, por todo el esfuerzo para superar los momentos más complicados en su vida.

Índice general

Resumen III

Agradecimientos V

Índice general IX

Índice de cuadros XI

Índice de figuras XIII

Índice de listados XV

1. Introducción 1

1.1. Motivación 2

1.2. Estructura del documento 3

2. Objetivos 5

2.1. Objetivo general 5

2.2. Objetivos específicos 5

3. Antecedentes 7

3.1. Estudios experimentales sobre Interacción Tangible 7

3.2. Interacción tangible en la enseñanza de programación 12

3.3. Aplicaciones de sistemas multipantalla 21

4. Interacción Persona-Ordenador e Interfaces 25

4.1. Interación Persona-Ordenador 25

4.2. Interfaz Gráfica de Usuario GUI 27

4.3. Interfaz Tangible de Usuario TUI 28

4.3.1. Uso de TUI en la educación 29

4.4. Interfaces Sistema Multi-Pantalla 29

5. Tecnologías para el desarrollo. Hardware y Software. 31

5.1. Hardware. 31

5.1.1. Raspberry Pi 3. 31

5.1.2. Pantalla táctil capacitiva. 33

5.1.3. Pantalla táctil resistiva HDMI. 37

5.1.4. Dispositivos de entrada. Sensores. 39

5.2. Software para el desarrollo. 43

5.2.1. Sistema Operativo GNU/Linux. Raspbian. 43

5.2.2. Python. 44

5.2.3. Kivy. 45

5.2.4. Interprete para el entorno de desarrollo Python Pycharm. 46

6. Método de trabajo 49

6.1. Metodología Agile 50

6.2. Scrum como metodología de trabajo 50

6.2.1. Roles 50

6.2.2. Artefactos 50

6.2.3. Motivos de la elección del método Scrum 50

6.3. Aplicación del método de trabajo 50

6.3.1. Iteración 1: Inicio del TFG 50

6.3.2. Iteración 2 : Formación y planificación del TFG. 51

6.3.3. Iteración 3: Estudio avanzado del problema. Identificación y selección de las áreas de mejora en sistemas tangibles. 54

6.3.4. Iteración 4: Requisitos iniciales. Objetivos del TFG. 61

6.3.5. Iteración 5: Diseño de la plataforma de juego. Tecnologías para el desarrollo. 64

6.3.6. Iteración 6: Estructura de la aplicación. Programa de pruebas comunicaciones entre dispositivos. 70

6.3.7. Iteración 7: Inicio del sistema de localización entre dispositivos. . . 73

6.3.8. Iteración 8: Lectura de datos y diseño de máquina de estados de la unidad de medición inercial *MPU9250*. 73

6.3.9. Iteración 9: Diseño software de las interfaces gráficas de usuario . . 73

6.3.10. Iteración 10: Diseño software de juego en la plataforma. 73

6.3.11. Iteración 11 Sistema de actualización software. 73

7. Resultados 75

7.1. Estructura de la aplicación. Gestión de eventos. 75

7.1.1. Máquina de estados finitos dispositivo *TUIO1/TUIO2*. 76
7.1.2. Clases y métodos del módulo `Tuio1.py/Tuio2.py`. 79
7.1.3. Gestión de eventos. Pruebas de validación. 81
7.2. Comunicaciones Servidor / Cliente. 81
7.2.1. Máquina de estados finitos Servidor / Cliente. 81
7.2.2. Gestión de eventos Servidor/Cliente. 84
7.2.3. Pruebas de validación en las comunicaciones cliente-servidor. . . . 85
7.3. Sistema de localización. 86
7.3.1. Versión de pruebas 1. Localización de coordenadas con patrón de triángulo rectángulo escaleno. 86
7.3.2. Versión de pruebas 2. Localización de coordenadas con patrón lineal. 93
7.4. Diseño software de la interfaz gráfica. 96
7.4.1. Manejo de eventos de la interfaz. 96
7.4.2. Módulo para el manejo de la interfaz `InterfaceManagement`. 97
7.5. Lectura y configuración de sensores. 98
7.5.1. Unidad de medición inercial MPU-9255. 98
7.5.2. Sensor de color. 101
7.6. Diseño de las aplicaciones. 102
7.6.1. Aplicación sensor resistivo. 102
7.6.2. Aplicación sensor capacitivo. 102
7.6.3. Aplicación sistema tangible orientada a la programación. 102
7.7. Pruebas y validaciones de librerías y módulos. 102

8. Conclusiones **103**

A. GNU Free Documentation License **107**
A.0. PREAMBLE 107
A.1. APPLICABILITY AND DEFINITIONS 107
A.2. VERBATIM COPYING 108
A.3. COPYING IN QUANTITY 108
A.4. MODIFICATIONS 109
A.5. COLLECTIONS OF DOCUMENTS 110
A.6. AGGREGATION WITH INDEPENDENT WORKS 110
A.7. TRANSLATION 110
A.8. TERMINATION 110
A.9. FUTURE REVISIONS OF THIS LICENSE 111
A.10. RELICENSING 111

Referencias **113**

Índice de cuadros

7.1. Intervalos de medida de cada segmento del triángulo 90
7.2. Métodos de la clase `InterfaceManagement()` 98
7.3. Métodos de la clase `MPU-9255` 101

Índice de figuras

1.1. Modelo de interacción de interfaz TUI para este proyecto 2

3.1. Seymour Papert Turtle. (Fuente: Mindstorms: children, computers, and powerful ideas [25]) 13

3.2. Radia Perlman's Slot Machine. (Fuente: Using computer technology to provide a creative learning environment for preschool children [26]) 15

3.3. Evolución de los bloques programables. (Fuente:[13]) 15

3.4. Programación de un submarino en pantalla, usando el sistema AlgoBlocks. (Fuente: AlgoBlocks [28]) 16

3.5. Digital Construction Set. (Fuente:Tangible programming bricks [29]) 17

3.6. Lenguaje de programación tangible: Electronic Blocks. (Fuente: Tangible Programming Elements for Preschoolers [30]) 18

3.7. Caso básico de uso TanPro-Kit. (Fuente: TanPro-Kit: a Tangible Programming Tool for Children [32]) 19

3.8. Ejemplo de programación de un bucle en Quetzal. (Fuente: Designing tangible programming languages for classroom use [34]) 19

3.9. Implementación de un automóvil autocontrolado. (Fuente: Programming and Robotics with Scratch in Primary Education [35]) 20

3.10. Mpad Controller. (Fuente: PlayMore [38]) 22

3.11. Sifteo Cubes. (Fuente: Sifteo Cubes [39]) 22

3.12. Tangicons versión 3,0. (Fuente: An Educational Non-Competitive Collaborative Game [40]) 23

3.13. Super Sync Sports. (Fuente: Awwwards [43]) 23

3.14. Visualizaciones en pCubee: imagen estática, dinámica, navegación por paisaje y selección directa de objetos mediante puntero.(Fuente: pCubee: A Perspective-Corrected Handheld Cubic Display [44]) 24

4.1. Ciencias que se relacionan con el HCI. (Fuente: Interacción tangible en aplicaciones educativas [46]) 26

4.2. Modelo de interacción de las interfaces gráficas (Adaptado de Ishii [47]) . . 27

4.3. Modelo de interacción de las interfaces tangibles (Adaptado de Ishii [47]) . 28

5.1. Pinout Raspberry Pi 3B. Fuente: [50] 32

5.2. Elementos de la laca Raspberry Pi 3B. Fuente: Raspberry Pi User Guide [50] 33
5.3. Pantalla táctil capacitiva 7 pulgadas Waveshare. Fuente: Waveshare Electronics [55] 36
5.4. Componentes panel táctil capacitivo 7 pulgadas. 37
5.5. Componentes pantalla táctil capacitiva 3.5 pulgadas 38
5.6. Componentes pantalla táctil capacitiva 3.5 pulgadas 38
5.7. Pines de la unidad de medición inercial MPU9255. 42
5.8. Esquema para una aplicación con bus I2C en MPU9255. 42
5.9. Logotipo Raspbian/Debian. Fuente: Página Oficial Raspberry Pi [59]. . . . 43
5.10. Menú de instalación NOOBS. 44
5.11. Pantalla de configuración de las interfaces I2C y SPI. 44
5.12. Logo de Python. Fuente: Python [61] 45
5.13. Logo de kivy. Fuente: Kivy [64] 45
5.14. Logotipo PyCharm. Fuente: Pycharm [65] 47

6.1. Diseño del elemento tangible *inicio*. 53
6.2. Ejemplo de secuencia. Al pulsar el botón *iniciar secuencia*, se produce un desplazamiento hacia la derecha. A continuación, se genera una señal acústica. La secuencia se mantiene durante un tiempo en pausa (según el tiempo ajustado en el temporizador). Finalmente se repite la secuencia. 53
6.3. Diseño del manejo de datos en los elementos tangibles mediante registro de desplazamiento y multiplexor. 54
6.4. Diseño del dispositivo principal en la segunda propuesta de plataforma de juego. 58
6.5. Diseño del dispositivo de control en la segunda propuesta de plataforma de juego. 59
6.6. Definir una secuencia en el dispositivo principal. 59
6.7. Desarrollo de una secuencia en el dispositivo principal. 60
6.8. Modificación de la secuencia usando sensores de proximidad. 60
6.9. *Captura* del cuadro activo dentro de la secuencia usando el dispositivo de control. 60
6.10. Modificación de la secuencia usando del dispositivo de control. 61
6.11. Emplazamiento dentro de la secuencia del cuadro modificado por medio del dispositivo de control. 61
6.12. Dispositivo de juego TUIO1. 65
6.13. Conexión de la interfaz gráfica de usuario y interfaz de eventos táctiles, entre pantalla y Raspberry Pi. 66
6.14. Dispositivo de juego TUIO2. 67

6.15. Disposición de las almohadillas conductivas en dispositivo de juego TUIO2. 68
6.16. Conexión de la interfaz gráfica de usuario y interfaz de eventos táctiles, entre pantalla y Raspberry Pi en TUIO2. 68
6.17. Ejemplo de aplicación de desplazamientos angulares mediante la unidad de medición inercial. 69
6.18. Sensor de color incorporado en el dispositivo TUIO2. 70
6.19. Ejemplo de aplicación para detectar objetos color rojo mediante el sensor de color. 71
6.20. Estructura de carpetas para la aplicación. 72

7.1. Manejo de eventos en *TUIO1/TUIO2*. 76
7.2. Estructura máquina de estados finitos `Tuio1FSM/Tuio2FSM`. 77
7.3. Manejo de eventos en *TUIO2*. Pruebas comenzar juego **PAINT GAME**. . . 82
7.4. Manejo de eventos en *TUIO1*. Pruebas comenzar juego **PAINT GAME**. . . 83
7.5. Estructura máquina de estados finitos del servidor/cliente. 84
7.6. Estructura manejo de eventos Servidor/Cliente. 85
7.7. Estructura de la secuencia de las pruebas cliente-servidor. 86
7.8. Versión de pruebas no validas. Posición de los dispositivos antes de la interacción sobre la pantalla capacitiva. 87
7.9. Versión de prueba no valida. Posición de los dispositivos después de la interacción sobre la pantalla capacitiva. 87
7.10. Versión de prueba no valida. Disposición de las almohadillas conductoras en la parte inferior del *widget tangible* para generar los eventos táctiles. 88
7.11. Versión de prueba no valida. Coordenadas de los puntos **A, B** y **C** del *widget tangible* sobre la pantalla capacitiva. 88
7.12. Situación de los puntos **A, B** y **C** del *widget tangible* en *TUIO2*. 94
7.13. Situación de los puntos **A, B** y **C** del *widget tangible* en *TUIO2*, sobre la pantalla táctil capacitiva del dispositivo *TUIO1* 95
7.14. Estructura de la gestión de eventos del módulo `InterfaceManagement()`. . . . 97
7.15. Esquema de conexiones con Raspberry Pi para comunicaciones I2C. 99
7.16. Conexionado TCS34725 Bus I2C con Raspberry Pi. 102

Índice de listados

7.1. Bucle para asignar las coordenadas de los vértices (Versión de prueba no valida). 91

Capítulo 1

Introducción

Desde los comienzos del nuevo siglo, el uso de la tecnología se ha convertido en parte de nuestra vida diaria, implementándose y evolucionando en todos los ámbitos, sobre todo para simplificar y facilitar el desarrollo de todo tipo de tareas. Por ejemplo, en el ámbito educativo cada vez son menos los centros docentes que no cuentan con un sistema informático aplicado a la búsqueda de información, mediante el uso de ordenadores, o sistemas que ayudan a los propios docentes a impartir su materia, como son los ordenadores o proyectores. El uso de la tecnología en las aulas es cada vez más evidente, y evoluciona adaptándose según las necesidades y los requisitos que se exigen, sobre todo en edades tempranas, utilizando distintos juguetes educativos que permiten a los niños aprender a la vez que se divierten, que es el tema que ocupa este Trabajo Fin de Grado.

El uso de juguetes educativos a lo largo de la historia ha sabido adaptar la evolución tecnológica para su beneficio, pudiendo realizar tareas complejas de una manera fácil, por ejemplo, controlar un pequeño robot con un simple tablero de instrucciones, o poder realizar secuencias de objetos y poder visualizar los movimientos en una pantalla. Esta aplicación de la tecnología en los juguetes educativos, ofrece cada vez, un abanico más amplio de juegos, donde también los niños son protagonistas en la creación de los mismo, como los llamados *juegos de interacción tangible*.

Estos sistemas permiten a los niños escribir un programa mediante el uso de objetos físicos, sin hacer uso de un teclado, para posteriormente ver los resultados realizados por ellos mismos.

En este Trabajo Fin de Grado se presenta un juego educativo basado en interacción tangible, aplicando las nuevas tecnologías, para fomentar la capacidad de los niños a la hora de utilizar su razonamiento en el proceso creativo en el transcurso de creación de un juego.

La idea principal parte de que los niños no sean simplemente consumidores de contenido y que tengan la posibilidad de crearlo, que ellos mismos sean capaces de tomar decisiones durante el juego de una manera tangible e intuitiva.

1.1 Motivación

La interfaz ha ido evolucionando ofreciendo un amplio abanico de opciones, especializándose y diversificándose para dar lugar a nuevos modelos de interacción, como son las basadas en soportes móviles, interfaces gráficas, tangibles, de realidad aumentada, las interfaces cooperativas y colaborativas, etc. Todas ellas poseen una serie de ventajas y restricciones a la hora de adaptarse a distintos entornos y situaciones.

Este TFG pretende desarrollar una plataforma interactiva de juego introduciendo *TUIOs*[1] accionados, que son controlables por el sistema con el fin de representar cambios dinámicos de la información, combinando elementos tangibles, gráficos y nuevas tecnologías.

La Figura 1.1 muestra una representación del modelo propuesto de interacción para la plataforma interactiva de juego en este proyecto. Está formado por dos componentes tangibles principales, *TUIO1* y *TUO2*.

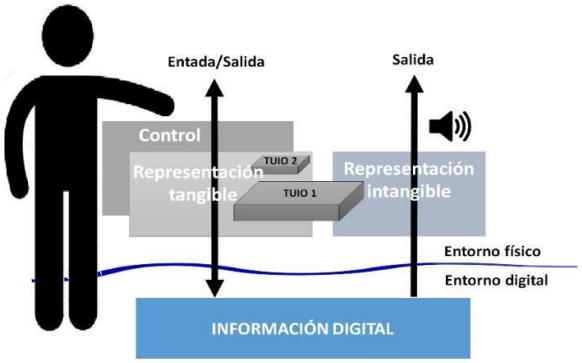

Figura 1.1: Modelo de interacción de interfaz TUI para este proyecto

El sistema de juego propuesto abarca desde el desarrollo de una secuencia, a partir de los criterios de creación del niño, hasta la resolución de pequeños problemas de lógica. El elemento *TUIO1* consta de una pantalla táctil, encargada de la representación gráfica tangible e intangible del juego. Si bien los elementos tangibles juegan el papel central en la representación y control en un *TUI*, la representación intangible es también importante, añadiendo gráficos y audio, siendo esta información dinámica proporcionada, una gran ayuda a la hora de realizar la interacción.

La existencia en tiempo real de una realimentación de la representación intangible, cuando se ha manipulado la representación tangible, es fundamental para asegurar un correcto acoplamiento perceptivo. Las representaciones intangibles que incorpora el diseño son tanto auditivas, producidas en determinados puntos del desarrollo del juego, como gráficas, mostrando en determinadas zonas de la pantalla táctil, información no modificable por el usuario.

[1] Tangible User Interface Objects

Los componentes, y los eventos que se producen en la interacción, junto con las comunicaciones con *TUIO2*, están controladas mediante un microprocesador.

El segundo elemento tangible *TUIO2*, actúa como elemento de programación tangible, agrupando funciones, características y elementos que pueden ser aplicados al juego, y que facilitan las interacciones o modificaciones a efectuar sobre el juego. Uno de los factores principales de las interfaces tangibles, es el acoplamiento de las representaciones tangibles a información digital y modelos computacionales.

Uno de los retos en el diseño de *TUI*, es como *mapear* los objetos físicos y como realizar una manipulación a la computación digital y la retroalimentación de una manera significativa y completa. Para solucionar el problema de disponer de distintos elementos tangibles para el desarrollo del juego, el dispositivo *TUIO2* hace de puente entre la interfaz física y la digital, agrupando todos los elementos de interacción en un solo dispositivo.

Al disponer de una pantalla táctil se hace más intuitivo y sencillo el uso de este elemento tangible. Además de una pantalla táctil dispone de sensores de posicionamiento y orientación. Estos componentes y los eventos producidos son tratados en un microcontrolador, el cual también es el encargado de las comunicaciones con el dispositivo *TUIO1*.

1.2 Estructura del documento

A continuación, se describe la estructura de contenidos que se ha seguido en el presente documento y que consta de ocho capítulos: **Capítulo 2: Objetivos**
Este capítulo se detalla los principales objetivos del TFG, indicando cuales son los requisitos mínimos que debe cumplir el software de la aplicación.

Capítulo 3: Antecedentes
En este capítulo se muestran los resultados sobre los principales estudios experimentales aplicados a la interacción tangible, y como motiva a niños de tempranas edades en sus habilidades computacionales, además de las ventajas en el aprendizaje mientras manipulan objetos en el escenario educativo. Se muestran algunos de los dispositivos más importantes y los diferentes enfoques que aportan en la interacción tangible aplicados al aprendizaje computacional, así como sus características y tecnología utilizada. También son mencionados los dispositivos con aplicaciones multipantalla, ya que algunas de sus características son aplicadas en el presente TFG.

Capítulo 4: Interacción Persona-Ordenador e Interfaces
La finalidad del capítulo es introducir una visión más amplia de la interacción Persona-Ordenador, describiendo sus principales características, y como las interfaces han ido evolucionando. Se analizan las ventajas y restricciones de cada una de ellas a la hora de adaptarse a distintos entornos y situaciones. Las interfaces tratadas en este capítulo son:

- Interfaz Gráfica de Usuario (GUI).

- Interfaz de Usuario Tangible (TUI), y su uso en la educación.
- Interfaz Sistema Multipantalla.
- Pantallas capacitivas. Funcionamiento *Widgets Capacitivos*.

Capítulo 5: Tecnologías para el desarrollo. Hardware y Software.
Este capítulo explica las funciones y características, de cada uno de los elementos hardware utilizados para realizar la plataforma de juego, así como el conjunto de herramientas software aplicadas para desarrollar los programas que manejan la plataforma.

Capítulo 6: Método de trabajo
Exposición y planteamiento del método de trabajo seguido para desarrollar la aplicación. Se ha aplicado la metodología de desarrollo **SCRUM**. En este capitulo son explicados cada uno de los pasos que han sido necesarios para realizar la plataforma de juego.

Capítulo 7: Resultados
En este capítulo se presentan los resultados obtenidos en el desarrollo de la aplicación, explicando la evolución por aspectos, comunicaciones, software de la interfaz gráfica, etc..

Capítulo 8: Conclusiones

Capítulo 2

Objetivos

La propuesta de este TFG se plantea para realizar una plataforma de juego educativa, basada en interacción tangible, aplicando las nuevas tecnologías para fomentar la capacidad de los niños, a la hora de utilizar su razonamiento en el proceso creativo en el transcurso de creación de un juego. La idea principal parte de que los niños no sean simplemente consumidores de contenido y que tengan la posibilidad de crearlo, que ellos mismos sean capaces de tomar decisiones durante el juego de una manera tangible e intuitiva.

2.1 Objetivo general

Diseñar y desarrollar una plataforma interactiva de juego de bajo coste, basada en interfaces tangibles y haciendo uso de las nuevas tecnologías, para promover el desarrollo de capacidades en niños de corta edad.

2.2 Objetivos específicos

- Diseñar y programar un sistema de localizacion y reconocimiento de las interfaces de usuario tangibles.
- Desarrollar un sistema eficiente de transferencia de datos entre ambas interfaces tangibles.
- Realizar una correcta sincronización en la visualización de la aplicación entre ambas pantallas, que permita una buena representación de juego.
- Ofrecer un entorno gráfico simple, que permita al niño ser capaz de diseñar secuencias y modificarlas interactuando con los elementos tangibles.
- Adaptar sensores inerciales y ópticos para una mejor experiencia de juego, donde ambos elementos tangibles interactúen entre sí, permitiendo la manipulación de la información entre los dispositivos.
- Diseñar un sistema de actualización de software de los dispositivos, a través de la conexión externa a un servidor.

Capítulo 3

Antecedentes

3.1 Estudios experimentales sobre Interacción Tangible

El pensamiento computacional *(CT)* [1], es un término calificado por Wing [1] para describir un conjunto de habilidades, pensamientos, hábitos y enfoques, que son integrales para resolver problemas complejos usando un sistema de computación, y ampliamente aplicables en la sociedad de la información. Cada vez hay más consenso que el pensamiento computacional es una habilidad fundamental que todo el mundo necesita para tener éxito en nuestra cultura compleja y tecnológica. *CT* permite a los niños mejorar la capacidad analítica que puede ser útil tanto en ciencia, tecnología, ingeniería y matemáticas, y en otras áreas profesionales, incluso en la vida cotidiana.

Exponer a los niños en edades tempranas a un pensamiento computacional, les ayuda a construir una base sólida en la mecánica de programación de computadoras [2]. Grover toma la programación, como una manera útil para la resolución de problemas, y evolucionar en un pensamiento computacional.

Las investigaciones indican que aprender a programar puede tener un efecto positivo y medible en el logro de los niños, no sólo en áreas como matemáticas y ciencias, sino también en habilidades del lenguaje, creatividad e interacción social y emocional. El hecho, es que aun la programación para niños sigue siendo difícil, y requiere de una fuerte motivación por parte del niño para tener éxito.

El entorno de programación debe motivar fuertemente al usuario, colocándolo en un contexto que sea convincente y significativo, para que no se dé por vencido. Las interfaces tangibles ofrecen un entorno apropiado para este propósito. Permite a los niños manipular códigos directamente, lo que hace que la programación sea más atractiva.

Manipular objetos con las manos en lugar de modelos mentales, puede cambiar la forma en la que un niño recuerda y recupera la información enseñada. Los niños entre 4 y 6 años aprenden a entender, representar y recordar objetos en su mente sin tener el objeto físico frente a ellos [3].

Las interfaces digitales se convierten en poderosas herramientas de aprendizaje cuando los

[1] Del ingles, Computational Thinking

niños se convierten en creadores en lugar de sólo consumidores de contenido digital [4].

Al programar, los niños exploran los conceptos fundamentales de secuenciación, reconocimiento de patrones y relaciones de causa y efecto. Los niños aprenden a contar una historia de principio a fin, realizar secuencias de números y letras, formas o colores [5].

Las interfaces de programación tangible son especialmente adecuadas para introducir a niños pequeños a la programación, desarrollando y aprovechando las estrategias de aprendizaje. Los juegos que implican la manipulación de objetos físicos que simbolizan otros objetos o ideas, permite a los niños explorar las reglas y sistemas de conceptos abstractos más amplios en diferentes ámbitos educativos [3].

Aprovechan las estrategias de aprendizaje kinestésico natural de los niños para hacer que los conceptos abstractos sean accesibles e intuitivos [6].

Mientras los niños juegan, se involucran con el mundo físico haciendo uso de sus cinco sentidos [7]. Los estudios realizados en adultos sobre el aprendizaje en entornos técnicos han demostrado que las interfaces tangibles son más intuitivas de dominar, al igual que más favorables a los comportamientos en resolución de problemas, y apropiadas para fomentar la conciencia espacial, especialmente en usuarios con baja cognición espacial [8] [9].

¿El uso de interfaces tangibles estimula el aprendizaje en los niños de corta edad?

Son diversas las investigaciones realizadas sobre los beneficios que aportan a los niños el uso interfaces tangibles. Se realiza un análisis sobre diferentes estudios aplicados en este ámbito, mostrando el potencial de estos sistemas a la hora de promover el aprendizaje sobre los lenguajes de programación tangibles.

Un estudio realizado por [10] sobre el aprendizaje de los alumnos de secundaria sobre las partes del cuerpo humano mediante una interfaz gráfica (representación en un ordenador) e interfaz tangible (partes de madera del cuerpo humano). Mientras la interfaz gráfica era más eficiente y promovía un aprendizaje más rápido de nuevos conceptos, al interfaz tangible promovía un compromiso más profundo y un enfoque prolongado explorando nuevas estructuras anatómicas.

Otro ejemplo, es el **estudio realizado por [11]**, que compara rompecabezas físicos, y otros que que se basan en interfaces gráficas. Los niños de 7 a 9 años consiguieron realizar los puzles tangibles más rápidamente y con una colaboración de compañeros más directa que los que estaban computarizados.

Un estudio llevado a cabo por Horn [12], haciendo uso de un lenguaje de programación robótico tangible / gráfico, pudo demostrar que las interfaces tangibles eran más atractivas para los niños y más útiles para fomentar la colaboración.

La interfaz tangible y la interfaz gráfica proporcionan un andamio estratificado a los niños mientras avanzan hacia un entorno de programación cada vez más auténtico [12].

En comparación con el funcionamiento de un ordenador que utiliza un teclado y un ratón, el uso de objetos físicos para interactuar es mucho más fácil, ya que consigue involucrar a los niños en el proceso [13]. Los niños pueden escribir programas montando los objetos físicos sin hacer uso pulsaciones en un teclado [12][13].

Estos hallazgos han sido repetidos en estudios similares como el de Cheng [14] y el realizado por Manches y Price [15].

Amanda Strawhacker y Marina U. Bers [16] presentan los resultados de un estudio piloto realizado en tres aulas de jardín de infancia, con tres tipos diferentes de interfaces, para enseñar a los niños como programar un robot. Las interfaces utilizadas son: interfaz gráfica de usuario *(GUI)*, interfaz de usuario tangible *(TUI)* y una interfaz en que los usuarios pueden cambiar libremente entre la entrada tangible y gráfica.

Al proporcionar a los niños la opción de cambiar libremente entre interfaces, se les ofrece una herramienta de resolución de problemas. Manches y Price [15] argumentan que cuando los niños se enfrentan a desafíos en el proceso de diseño, es importante que puedan elegir que herramientas quieren utilizar para resolver el problema, lo que ofrece una ayuda para entender mejor el problema en sí mismo.

Los conceptos básicos a la hora de hacer el estudio, haciendo uso de una interfaz tangible se definieron de la siguiente manera:

- **Secuenciación:** este concepto introduce la idea de que el orden es importante al dar una serie de instrucciones, más específicamente, observar como el orden de los comandos afecta al orden de las acciones.

- **Repetición de bucles:** Repetir *loops* permite explorar las acciones y secuencias que se repiten dentro de un programa, creando esencialmente programas dentro de otros más largos.

- **Programación creativa:** Alentar a los niños a fijarse una meta imaginativa para su propio programa. Esto ofrece a los niños explorar todos los conceptos de programación aprendidos anteriormente.

Las cuestiones tratadas en esta investigación son: cuales son los conocimientos adquiridos adecuadamente en los niños aplicando la tecnología (es decir, los lenguajes usados de programación en el contexto de la robótica) y que características y propiedades aprenden sobre ella. Los resultados obtenidos desvelan una mejora significativa entre interfaces *TUI, GUI y HUI* en tareas sobre bucles de repetición, siendo la interfaz *GUI* ligeramente más atractiva en tareas de secuenciación. El estudio muestra que los niños usan la interfaz tangible más que la gráfica, cuando se les da la oportunidad de usar ambas.

Estudio realizado por Wang y Liu [17]. El estudio presenta los resultados relacionados con la aplicación de sistemas tangibles, a la hora de asimilar conceptos computacionales

en actividades de programación. Formaron parte del estudio participantes de edades comprendidas entre los 5 y 9 años. Se les facilitó herramientas para la interacción, en este caso, una serie de módulos tangibles con la finalidad de generar un programa (secuencia) al ser colocados de una forma determinada.

El juego consiste en controlar un avatar virtual, el cual se desplaza a través de un laberinto. Este laberinto es mostrado en una interfaz gráfica, donde el niño observa los progresos conseguidos. El objetivo principal del estudio es analizar los resultados de las interacciones en el pensamiento computacional de los niños, tomando especial interés a estos cuatro conceptos:

- **Abstracción:** La capacidad de abstracción consiste en encontrar el nivel adecuado de detalle para definir y resolver un problema. [18].
- **Automatización:** Proceso por el cual se produce un ahorro de trabajo que se instruye para efectuar un conjunto de tareas repetitivas mediante una computadora. Es mucho más eficiente en comparación con el poder de procesamiento de un ser humano, por lo que las ejecuciones automatizadas de procesos realizados por máquinas simplifican el trabajo [19].
- **Descomposición y análisis de problemas:** La descomposición puede tomar la forma de desmontar un problema a lo que se cree que es su esencia fundamental [20]. Romper los problemas en partes más pequeñas para poder resolverlos y analizarlos de una manera más fácil, ayuda a la simplificación y resolución de problemas más complejos y de mayor escala.
- **Creatividad:** La creatividad es a la vez una capacidad integral de los seres humanos y una parte importante del conjunto de actividades del pensamiento computacional [21]. Se intenta desarrollar la habilidad creativa de los niños estimulando su curiosidad, especialmente la imaginación.

¿Cómo toman forma estos cuatro conceptos *CT* para los niños en la programación? Se propuso a los participantes realizar unas sesiones en las cuales debían completar unos niveles tanto de creación, como de resolución de laberintos. Los resultados obtenidos del estudio en las diferentes sesiones muestran como los niños son capaces de formar sus propios programas mediante bloques tangibles, involucrándose en la creación y resolución de los problemas planteados.

Durante las diferentes etapas se observa, como los niños necesitan el proceso de abstracción para reducir el problema. La abstracción también tiene lugar cuando los niños diseñan y crean diferentes tipos de laberinto.

En cuanto al concepto de automatización, la mayoría de los niños son capaces de indicar a sus distintos avatares las instrucciones para realizar el recorrido, tomando especial entusiasmo, las órdenes que ellos mismos indican.

En el concepto de descomposición y análisis del problema, los niños realizan un análisis cuando deciden si el avatar se comporta como se espera. Si el avatar esta atascado o *mal encaminado*, indica que su implementación de su idea de control es defectuosa o que algunos bloques están fuera de lugar.

Creatividad. Para crear un laberinto, la programación se convierte en un medio para la expresión personal y creativa de los niños. En el diseño de sus laberintos, los niños involucran sus fantasías y construyen recorridos que van más allá de los pensamientos comunes.

Se concluye que los sistemas tangibles tienen el prometedor potencial de ayudar a los niños a cultivar un cierto conocimiento del pensamiento computacional como abstracción, automatización, descomposición de problemas y análisis.

Teodosios Sapounidis y Stavros Demetriadis [22] realizaron una evaluación sobre el rendimiento de los niños con datos gráficos y tangibles en tareas de programación de un robot. El sistema propuesto está formado por dos subsistemas. El primero una interfaz GUI y el segundo una interfaz TUI.

El subsistema tangible consiste en bloques en forma de cubo que representan las estructuras de programación, tanto las simples como las avanzadas. Los usuarios conectan los diferentes cubos, para así crear la programación para el control de un robot.

El subsistema gráfico consiste en un entorno de programación, donde los participantes indican las operaciones a realizar por medio de la visualización de distintas estructuras, como; flechas, relojes, números. . . , que indican la acción a realizar por el robot.

En este estudio se forman grupos de participantes de edades comprendidas entre los 6 y los 12 años. Las tareas a realizar en la prueba consisten en:

- Realizar un programa secuencial simple de hasta seis órdenes (avanzar/retroceder, encender/apagar led, provocar un sonido) que implica el uso de varios cubos.
- Formar un programa secuencial más avanzado que el anterior, donde además de las ordenes descritas en la primera prueba, se añaden otras como girar a un lado y a otro.
- Construir un programa secuencial que implique repeticiones (lazos).

Se les pidió a los niños realizar dichas tareas, aplicando ambas interfaces. Finalmente, después del proceso de interacción, se dejó a los niños utilizar libremente la interfaz que ellos quisieran.

El tiempo usado en cada una de las pruebas realizadas a los diferentes grupos de niños objeto del análisis, muestran que los tiempos promedio utilizados para las tareas con la interfaz tangible, están por debajo de los tiempos utilizados para la gráfica.

El uso de la interfaz gráfica ofreció un porcentaje superior de error a la hora de ejecutar cada una de las tareas asignadas. La disminución del número de errores durante la progra-

mación del robot con el subsistema tangible se debe a la participación activa y a la motivación.

Por otra parte, la tarea de ejecución libre dejo los siguientes resultados: Los niños de mayor edad, que eran usuarios de ordenadores, y por lo tanto, más experimentados, exploraron más órdenes y parámetros alternativos, y crearon programas más complicados con el subsistema tangible. Se produjeron menos errores de programación y se logró una mejor depuración en el caso tangible. Por el contrario, los niños más pequeños pasaron casi el mismo tiempo en ambos casos, pero se involucraron más con los tangibles.

Se muestra una transición equilibrada de una interfaz a otra, en relación con la experiencia del usuario y el tiempo de interacción. El estudio resalta que el uso de la interfaz tangible hace más probable que se lleve a cabo un proceso completo de depuración y corrección del error, mientras que con la gráfica es más probable que se descuide. En conclusión, los resultados muestran que los sistemas tangibles aplicados en la programación, fomentan el aprendizaje y el rendimiento en tareas computacionales.

3.2 Interacción tangible en la enseñanza de programación

La educación tecnológica para los niños se está volviendo cada vez más importante. Existe un desafío continuo para que los educadores desarrollen programas que reconozcan y respondan al impacto de la revolución de la información. Toman especial importancia para los niños pequeños, al poder explorar y dar sentido al mundo que les rodea, tener experiencias agradables y positivas entre ordenadores y tecnología. Existe la preocupación por la preparación física y cognitiva de los niños pequeños para usar ordenadores, y su impacto en su desarrollo social y educativo, que ha llevado a cierto rechazo de muchos educadores al uso de ordenadores como un recurso educativo apropiado para la primera infancia [23].

La cuestión es que los niños pequeños aprenden mejor mientras manipulan y transforman activamente materiales reales [24]. Los niños deben ser capaces de desempeñar un papel activo en sus encuentros con la tecnología. Por ello el desarrollo de una interfaz tangible aborda este problema, ofreciendo una serie de beneficios en su educación y creatividad.

La finalidad es fomentar este sistema en el uso en aplicaciones para el escenario educativo, combinando para este propósito, las mejoras de los medios digitales, utilizando entornos visuales, animaciones y sonidos, que son un importante estímulo a la hora del aprendizaje aumentando su motivación en tareas educativas.

Lo que nos ofrece el mercado en estos momentos en el ámbito de juegos dedicados a la programación tangible o visual, se basa principalmente, en realizar una secuencia en la cual el niño establece previamente unas características y una serie de acciones a realizar sobre el juego, ya sea con un ordenador personal o dispositivo móvil, haciendo uso de una pantalla para el desarrollo del juego, o mediante piezas pasivas colocadas con una determinada

posición, que indican las acciones a realizar en una secuencia.

A lo largo de la historia se han ido desarrollando aplicaciones de índole educativo para enseñar el lenguaje de la programación a niños pequeños, haciendo uso de interfaces tangibles. A continuación, se describen algunos de los proyectos enfocados al aprendizaje y enseñanza de la programación tangible.

Seymour Papert Turtle [25]. Fue uno de los principales promotores de la programación para niños mediante sistemas tangibles. Papert y sus alumnos del *MIT*[2], comenzaron a investigar métodos para introducir a los niños en el mundo de la informática, para ofrecer maneras de programación de forma física. Esta investigación inicial, condujo a la creación del lenguaje *Logo*. *Logo* es un lenguaje de programación *LISP*[3]. Este lenguaje es potente pero simple, y sirve para explorar la resolución de problemas lingüísticos y matemáticos. *Logo* es relativamente libre de las reglas sintácticas que hacen que muchos lenguajes de programación sean difíciles de aprender para los niños.

Su diseño consiste en un robot con forma de tortuga, el cual dispone de un mecanismo de dibujo. Este mecanismo fue diseñado para que los niños aprendieran conceptos geométricos. El centro de este enfoque sostiene que los niños aprenden conceptos geométricos con más facilidad, si pueden modelar formas geométricas en el espacio físico con su propio cuerpo. Este enfoque se denomina geometría centrada en el cuerpo. Por ejemplo, en lugar de pedirle inmediatamente a un niño que describa como podría instruir a la tortuga que dibujase una forma cuadrada, se le pide que explore la creación de la forma moviendo su propio cuerpo en el espacio sobre el suelo. A través de la prueba y error, el niño rápidamente se da cuenta de que caminar unos pasos, girar a la derecha 90º, y repetirlo cuatro veces, consigue realizar un cuadrado. Después de este ejercicio, el niño está mejor preparado para programar la tortuga para repetir los movimientos. Estos movimientos, son transmitidas al robot con un ordenador (ver Figura 3.1.

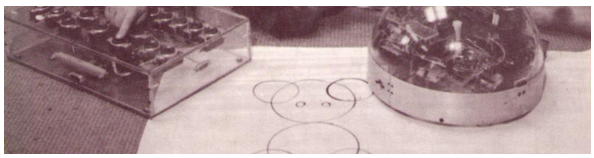

Figura 3.1: Seymour Papert Turtle. (Fuente: Mindstorms: children, computers, and powerful ideas [25])

Tortis Slot Machine [26]. Radia Perlman fue una de las pioneras en aplicar sistemas para el uso de lenguajes de programación en niños de 3 a 5 años, desarrollando la máquina Tortis. Durante años de enseñanza de *Logo*, se hizo evidente para los investigadores, que la mayoría de los niños no estaban preparados para comenzar a programar ordenadores de la manera

[2]Computer Science and Artificial Intelligence Laboratory.
[3]Familia de lenguajes de programación de tipo multiparadigma

tradicional, es decir, tecleando el código de *Logo* en una computadora usando un teclado, hasta alcanzar al menos los 10-14 años de edad.

Radia Perlamn a mediados de la década de los 70, creía que uno de los principales impedimentos para el acceso de los niños a la programación de ordenadores no era solo la sintaxis del lenguaje, sino también la interfaz de usuario. Perlman comenzó a diseñar interfaces, que permitirían que incluso los preescolares, aprendieran a programar una tortuga. Para ello, se diseñó dos nuevos dispositivos de entrada informalmente llamados *Button Box* (Panel de funciones), y *Slot Machine* (Máquina de ranuras).

El panel de funciones sirve para indicar todas las operaciones a realizar, ya sea movimientos, encender o apagar una luz, etc. Los niños pequeños podrían usarlo como control remoto para ordenar a la tortuga que acción realizar, sin tener que aprender a escribir los comandos en un teclado. Este panel fue mejorado añadiendo un sistema de memoria, donde se puede registrar una transcripción de órdenes, para más adelante ser realizadas. Con ello, el niño podría mostrar físicamente al ordenador que hacer, para posteriormente realizar la secuencia.

Los dos grandes problemas encontrados fueron que los procedimientos eran demasiado abstractos para los niños pequeños. En segundo lugar, una vez iniciado el programa, no se permitía modificar la secuencia una vez grabada, por lo que el niño debía volver a realizar el programa. La solución fue la utilización de Slot Machine. Esta máquina hace uso de tarjetas de plástico, que son insertadas dentro de ellas. La incorporación de estas tarjetas ofrece la capacidad de manipular un programa directamente, con la adición, la reorganización y la eliminación de las tarjetas de forma manual.

A principios de los años 80, cuando los ordenadores personales se hicieron más comunes en las escuelas, los investigadores decidieron cambiar el enfoque de la tortuga moviéndose por el suelo, y adaptaron una representación de la tortuga por medio de la pantalla de un ordenador. La visualización de manera virtual en una pantalla ofreció nuevas oportunidades de aprendizaje y exploración.

P-Brick [27]. Hasta principios de 1990, no fue cuando los investigadores de la informática educativa del *MIT*, volvieron a estudiar la interacción con el mundo físico. La interfaz *LEGO TC Logo*, desarrollada por Steve Ocko, fue capaz de permitir a los desarrolladores, escribir un programa *Logo* en un ordenador personal para controlar dispositivos construidos con bloques de construcción de LEGO. Estos artefactos pueden realizar tareas tales como clasificar piezas LEGO por tamaños, aunque aún estos dispositivos estaban unidos con cables a un ordenador. Martin, Sargent y Silverman, eliminaron este problema utilizando microcontroladores dentro de cada bloque, que ejecutaba el intérprete de *Logo*. P-Brick fue utilizado para controlar los primeros LEGO Mindstorms Robotics

P-Brick fue la inspiración y la base para una serie de juguetes diseñados para niños.

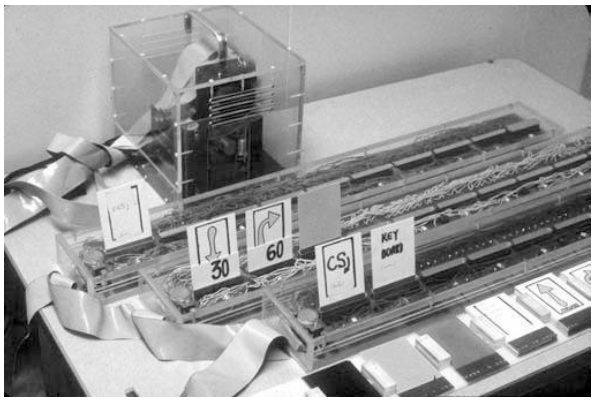

Figura 3.2: Radia Perlman's Slot Machine. (Fuente: Using computer technology to provide a creative learning environment for preschool children [26])

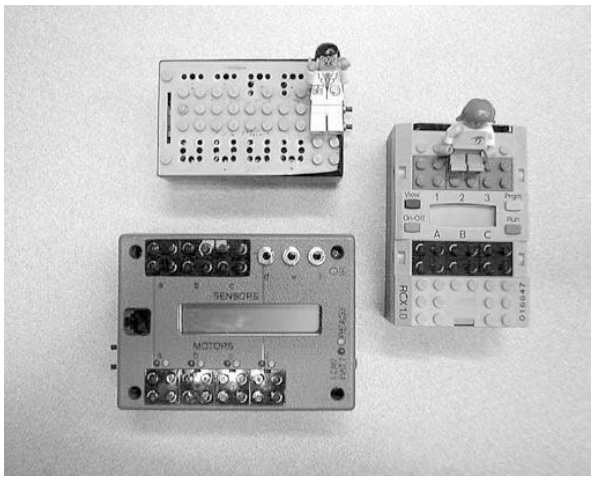

Figura 3.3: Evolución de los bloques programables. (Fuente:[13])

Dr.LegoHead (Rick Borovoy), personaje interactivo, donde las partes de su cuerpo son removibles. **MyBot** lanzado en el 2000, donde el juguete sigue la línea de componentes por bloques añadiendo nuevos sensores, como por ejemplo de inclinación, donde el juguete emite un sonido cuando es cambiado de orientación.

Neurosmith Music Blocks. Dispositivo donde el niño dispone de cinco bloques cúbicos coloreados, donde cada uno representa una nota musical, creando diferentes composiciones musicales.

AlgoBlocks [28]. El término *programación tangible*, fue ideado por Suzuki y Kato, desarrolladores del sistema AlgoBlocks (Figura 3.4). Este sistema está diseñado para la resolución colaborativa de problemas. Está formado por cubos relativamente grandes (15 cm), donde el objetivo del juego es guiar a un submarino a través de un laberinto. El lenguaje utilizado es bastante parecido a *Logo*. La programación es realizada físicamente por medio de los cubos, que definen el recorrido a seguir para resolver un laberinto. La ejecución del

programa es virtual, representando las acciones gráficamente. Los bloques disponibles permiten realizar acciones como pueden ser giros o movimientos, incluso añadir parámetros. Otros bloques están diseñados para realizar sentencias de control como son: *if-then-else* y *repeat-until*, para un mejor control lógico.

Figura 3.4: Programación de un submarino en pantalla, usando el sistema AlgoBlocks. (Fuente: AlgoBlocks [28])

Digital Construction Set [29]. En el año 2000, McNerney diseño esta plataforma para explorar lenguajes de programación tangibles que cumplieran ciertos requisitos de diseño:

- Fácil de entender.
- Apropiado para realizar el juego de forma libre.
- Bloques simples que representen el vocabulario/léxico.
- Posibilidad de realizar un gran número de programas.
- Mejorar/depurar sin necesidad de herramientas externas.

El autor diseñó un sistema de *ladrillos LEGO*, apilables para construir los programas. El lateral de cada *ladrillo* dispone de una ranura donde el niño inserta elementos como ruedas digitales, botones analógicos, además de permitir definir parámetros, permitiendo así una manipulación adicional (Figura 3.5).

El sistema para comunicar y alimentar los *ladrillos* es por contactos eléctricos (ISO 7816) en su parte superior e inferior. Pueden adaptarse en orientaciones de 0 y 180 grados. Cada bloque dispone de un microcontrolador *PIC*[4] con un intérprete *Logo*. Cada bloque es alimentado con una única batería que se sitúa en el bloque de la base que forma la pila de *ladrillos*. La comunicación entre ladrillos es mediante *Bus serie I2C*. Cada ladrillo se comunica enviando mensajes que contienen eventos o datos. Un bloque situado en la parte superior puede recibir (y responder) mensajes del bloque posterior, produciéndose un flujo de datos ascendente.

[4]Peripheral Interface Controller

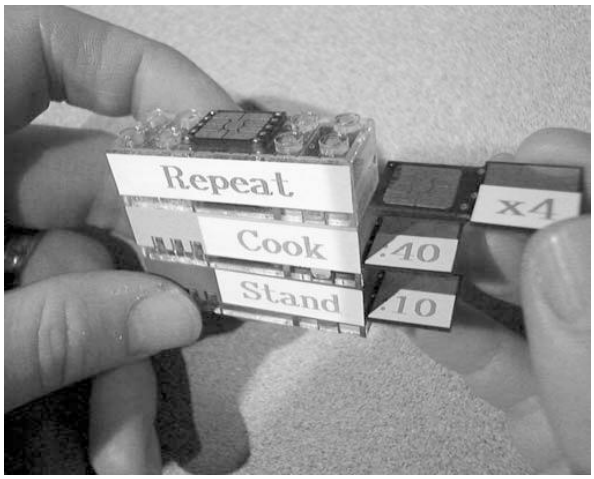

Figura 3.5: Digital Construction Set. (Fuente:Tangible programming bricks [29])

Las característica más importante del sistema *Digital Construction Set*, es que permite a los investigadores y programadores jóvenes reprogramar el comportamiento de cualquiera de los bloques, permitiendo inventar su propio vocabulario específico, adecuando cada uno de los bloques a una tarea particular. No obstante, esta interfaz de programación tangible ofrece una limitación, ya que solo permite una construcción en una sola dirección, lo que limita posibles ramificaciones laterales en el programa.

Electronic Block [30]. Este juego está formado por bloques, diseñados para que los niños puedan conectarlos unos con otros, siendo fácilmente apilables y enfocado principalmente para niños en edad preescolar, por lo que hace uso de una sintaxis fácil de programar. Dentro de cada bloque están situados diferentes componentes electrónicos para la interacción y realización del juego. Existen tres tipos de bloques electrónicos:

- Bloques de sensores: Estos bloques son capaces de detectar luz, sonido y tacto. Son bloques de conector único, es decir, tienen una entrada del bloque anterior, y una salida al bloque de abajo. Para que los niños puedan identificar este tipo de bloque más fácilmente, todos ellos son de color amarillo.
- Bloques de acción: Los bloques producen algún tipo de salida física, como puede ser encender una bombilla, emitir un sonido o reproducir una simple melodía. Existe un bloque de movimiento que emula un coche de cuatro ruedas que se mueve en línea recta. El diseño de este tipo de bloques solo permite colocar otro bloque en la parte inferior. Estos bloques están adornados con iconos explicativos para una mejor interacción en el juego.
- Bloques lógicos: Tienen un rol intermedio. Colocados entre un bloque sensor y un bloque de acción, tienen la capacidad de alterar la acción esperada. Los bloques lógicos proporcionan al niño la capacidad de producir una acción si no se recibe un estímulo particular (not), producir retardos en las acciones (delay), solo producir una acción

si las señales de entrada son recibidas simultáneamente (and), etc. Cada bloque esta diferenciado por colores e iconos identificativos.

Estos bloques están diseñados para que los niños, jugando simplemente con ellos, puedan producir acciones y comportamientos. Pueden construir una torre que parpadea cuando hablan, o se mueve con un simple toque. La adicción de bloques lógicos añaden complejidad, y dan la posibilidad de crear una gran variedad de estructuras (Figura 3.6).

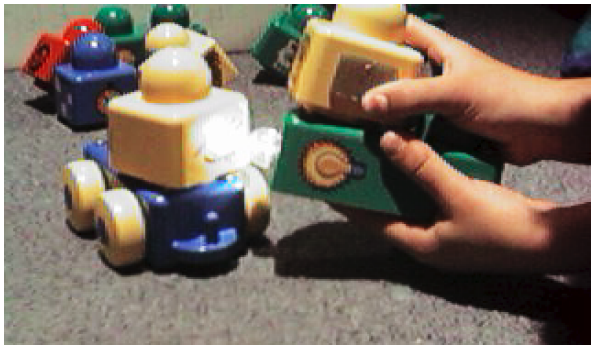

Figura 3.6: Lenguaje de programación tangible: Electronic Blocks. (Fuente: Tangible Programming Elements for Preschoolers [30])

D.Wang creador de **T-Maze** [31], posteriormente **TanPro-Kit** [32], y **E-BLOCKS** [33]. Estos sistemas de programación tangible utilizan bloques que al ser conectados, generan en tiempo real movimientos dentro de un laberinto. En su funcionamiento se dispone de distintos tipos de bloques: bloque de inicio, bloque de fin, bloque de dirección y bloque sensor. Las interacciones entre bloques son mostradas en una interfaz gráfica y en los propios bloques tangibles (realimentación), indicando si la acción llevada a cabo es correcta. Este tipo de *TUI*, no depende de la representación *intangible* puesto que la retroalimentación activa a través de la representación tangible, sirve como el canal de visualización principal. En el caso de TanPro-Kit existe dos niveles de juego: un nivel introductorio, y un nivel adicional. Este sistema se centra en ayudar a los niños a aprender programación como principiantes. Además, no requiere de requisitos especiales a la hora de ser usado.

Quetzal [34] es un lenguaje de programación diseñado para controlar LEGO Mindstorms[5]. Consiste en realizar el enclavamiento de varias piezas de plástico que representan las estructuras de control de flujo, acciones y parámetros. Las declaraciones en el lenguaje están conectadas entre sí para formar cadenas de flujo de control. Al realizar un programa simple, se comienza con la instrucción *Begin* y termina con la instrucción *End*. En la Figura 3.8 se presenta un ejemplo de programa, que inicia un motor, espera tres segundos para posteriormente detener el motor. Los usuarios pueden agregar o cambiar valores en los parámetros para ajustar tanto el tiempo de espera, como la potencia que genera el motor. Ciertas

[5]Línea de juguetes para niños enfocados a la robótica, que posee elementos básicos para la programación de forma interactiva.

Figura 3.7: Caso básico de uso TanPro-Kit. (Fuente: TanPro-Kit: a Tangible Programming Tool for Children [32])

sentencias aceptan valores de parámetros que pueden incluir constantes o lectura de sensores. Estos parámetros son representados en pequeñas fichas de plástico, que son insertadas en la cara superior de las piezas principales (declaraciones). El orden de las secuencias es un factor importante, pero la forma general de un programa no cambia su significado.

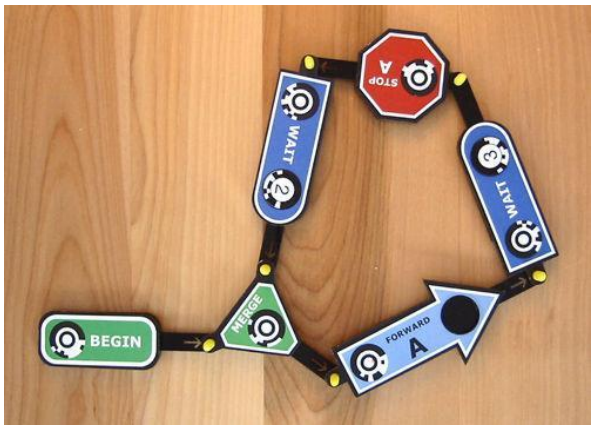

Figura 3.8: Ejemplo de programación de un bucle en Quetzal. (Fuente: Designing tangible programming languages for classroom use [34])

La implementación del programa se realiza por técnicas de procesamiento de imágenes, que convierten el programa físico en código de máquina. Cada sentencia es identificada según su posición, tamaño, o color. Las rutinas de procesamiento de imágenes utilizan un algoritmo de umbral adaptativo, que trabajan bajo diferentes condiciones de iluminación sin necesidad de calibración por parte del usuario.

Una vez procesada la imagen e identificado cada una de las sentencias a realizar, el robot LEGO Mindstorms es accionado según las instrucciones especificadas.

LEGO WeDo and Scratch [35]. Scratch [36] es un lenguaje de programación visual ba-

sado en Logo, diseñado para que los niños creen fácilmente su propio contenido interactivo (juegos, música, historias interactivas...). Además, Scratch facilita una página web donde los niños pueden subir sus propias creaciones y compartirlas, fomentando de esa manera, el trabajo en colaboración. Incluye funcionalidades, interfaz gráfica, multimedia, eliminación de errores de sintaxis, etc. Permite usar programación dirigida por eventos con objetos activos denominados *sprites*. Estos *sprites* pueden dibujarse como gráficos de tipo vectorial o mapa de bits, o pueden ser importados desde fuentes externas incluyendo webcams. Una vez creado un objeto, éste es manipulado aplicando bloques de funciones que van desde el movimiento, apariencia, sonido, y controlados mediante bloques de eventos, operadores, etc.

El conjunto de construcciones WeDo [37], permite a los usuarios la construcción y programación de modelos simples de LEGO mediante el uso de un ordenador.

La combinación de Scratch y WeDo en la creación de proyectos dedicados a la robótica, ejemplifica el paradigma del aprendizaje del construccionismo[6] [25], donde el joven estudiante es el protagonista del proceso de aprendizaje. El construccionismo afirma que la programación, depuración, el acto de construir un sistema, encontrar problemas y resolverlos, es la manera más eficaz de que un niño aprenda. Las dos plataformas (software y hardware), facilitan la introducción de la programación y la robótica en la escuela primaria (Figura 3.9).

Figura 3.9: Implementación de un automóvil autocontrolado. (Fuente: Programming and Robotics with Scratch in Primary Education [35])

Estos sistemas proporcionan al usuario un camino para avanzar en el aprendizaje hacia unos entornos de programación cada vez más complejos. Los niños pueden crear programas sencillos fácilmente, y aprender rápidamente de las interacciones físicas, preparándose para participar en el apasionante mundo de la programación de computadoras dada la interfaz correcta.

[6]Teoría de la educación basada en la teoría del aprendizaje desarrollada por el sicólogo suizo Jean Piaget [3]

3.3 Aplicaciones de sistemas multipantalla

Este enfoque de diseño introduce una nueva experiencia multi-dispositivo, donde los distintos dispositivos se complementan entre sí. Aunque hay casos en el que el usuario interactúa con los distintos elementos de manera individual, su finalidad de funcionamiento es como un grupo completo conectado, para ofrecer una experiencia más interactiva.

La consecución de un enfoque complementario implica dos tipos de relaciones entre dispositivos:

- **Colaboración:** Cada uno de los dispositivos tienen un papel distinto, trabajando juntos normalmente simultáneamente, para construir la experiencia. Por ejemplo, jugar al domino. El tablero donde son situadas cada una de las piezas es visualizada en un dispositivo (televisor, tablet, ordenador) y las fichas de cada jugador son mostradas en otro dispositivo (teléfono inteligente).

- **Control:** La experiencia principal del usuario tiene lugar con un dispositivo en particular, mientras que los otros dispositivos controlan los aspectos del juego, generalmente remoto. Por ejemplo, el uso de un teléfono inteligente que sirve como control remoto de una televisión.

Además, los dispositivos pueden tener un peso diferente en el conjunto del juego y son parte integral de la experiencia.

MPad-Controller es una aplicación multipantalla para móviles y tablets, desarrollado por la empresa PlayMore para la compañía Apple. Se trata de un conjunto de juegos donde están implicados varios dispositivos, comunicados entre sí mediante una aplicación software, para ofrecer un entorno de juego compartido entre distintas pantallas (Figura 3.10). Uno de los dispositivos actúa como pantalla principal para la visualización central del juego, mientras que los otros dispositivos realizan la función de *joystick virtual*.

Sifteo Cubes [39]. Plataforma interactiva de juego formada por varios bloques. Cada uno posee una pantalla donde el usuario los hace interactuar entre sí cuando se agitan, inclinan, giran o se colocan unos junto a otros. Los cubos tienen un tamaño de 1,5 pulgadas y se comunican entre sí a través de un protocolo de radio de 2,4 GHz. El prototipo actual soporta hasta seis cubos, y tanto la animación como la programación son efectuadas mediante lenguaje C. Este tipo de juego esta diseñado para edades a partir de 6 años.

Tangicons 3.0 [40]. Este sistema pretende presentar a los niños de entre 6 y 9 años, las herramientas necesarias para fomentar el razonamiento y el trabajo colaborativo, en los primeros pasos de la programación con objetos tangibles dentro de un juego educativo. Dependiendo de los jugadores, el juego puede actualizar su complejidad para iteraciones adicionales.

La primera versión del juego está formada por cubos de madera pasivos, donde los niños

Figura 3.10: Mpad Controller. (Fuente: PlayMore [38])

Figura 3.11: Sifteo Cubes. (Fuente: Sifteo Cubes [39])

deben colocarlos de forma correcta para elaborar una secuencia que finalmente acciona un motor. La secuencia es reconocida mediante procesamiento de imágenes, captada por una cámara digital. El manejo de esta cámara fue un reto para los niños pequeños, y la transmisión entre el ordenador y la estación de ejecución tampoco era muy confiable.

La segunda versión integró la tecnología dentro de los cubos de programación utilizando microcontroladores *ATmega8*, acelerómetros, módulos de radiofrecuencia, leds, y sonido. La estación de ejecución se encarga de alimentar a los cubos con una batería interna. En esta nueva versión se proporciona a la estación, un módulo de comunicación inalámbrico para la conexión con el ordenador. Uno de los juegos consiste en realizar parejas de cubos con el mismo color. Desafortunadamente surgió el problema de que la electrónica de su interior además de tener un alto coste es bastante frágil y no soporta la rutina diaria a la que puede ser expuesto el juego. Para solucionar estos problemas, se introdujo el uso de los cubos Sifteo.

Tangicons 3.0 busca una mezcla entre juego interactivo con el movimiento físico, es decir, los niños tienen que moverse, incluso correr entre sus tareas. Numerosos estudios muestran la relación positiva entre el movimiento y la intensidad de las habilidades intelectuales en los estudiantes. Zimmer [41] demuestra que las experiencias motrices, sirven de base para la posible creación de un instrumento cognitivo. Zahner [42] también muestra que existe una interrelación entre la actividad motora y un mayor desarrollo intelectual.

Para ejecutar uno de los juegos, los niños deben superar un laberinto donde controlan un *avatar*. Para hacer el recorrido del laberinto, se han de colocar los cubos en fila, en las posiciones correctas para ejecutar una secuencia (Figura 3.12). Después de cada ciclo de ejecución, el avatar se quedará sin energía y tiene que ser recargado. Para recargarlo, existe una estación de recarga situada en el otro extremo del recinto donde se desarrolla el juego, por lo tanto, los niños deben correr hacia esa estación junto con el cubo que representa su avatar para poder superar correctamente el juego, manteniendo su cubo contra el cargador para posteriormente regresar a la estación de programación.

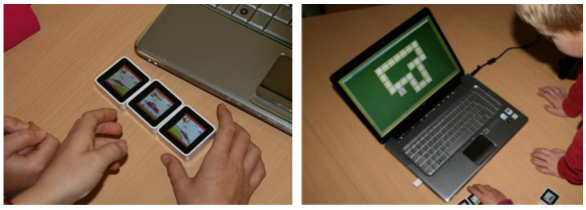

Figura 3.12: Tangicons versión 3,0. (Fuente: An Educational Non-Competitive Collaborative Game [40])

Chrome Super Sync Sports [43] es un experimento de Google que permite sincronizar un teléfono móvil con un ordenado para usarlo como un controlador de juego. Consiste en varias pruebas de carácter deportivo, que los jugadores han de superar con la pantalla táctil de su dispositivo móvil. Construido con tecnologías *HTML5, CSS3, Javascrip*t y la *API WebSockets*, Super Sync Sports permite a hasta 4 usuarios competir en juegos en una pantalla de un ordenador compartida, usando sus smartphones o tablets como controladores (ver Figura 3.13) .

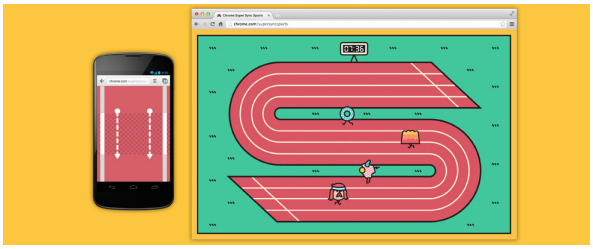

Figura 3.13: Super Sync Sports. (Fuente: Awwwards [43])

Pcubee [44]. Diseño de una pantalla cúbica que ofrece técnicas de interacción para conte-

nido 3D estático y dinámico. Cinco de las caras del cubo están formadas por pantallas LCD para representar imágenes en perspectiva (ver Figura 3.14). La simulación en tiempo real simula la interacción de tener objetos reales dentro de una caja física que el usuario puede sostener y manipular. Se puede interactuar de diferentes maneras: ver una escena estática, navegar por un paisaje, jugar con objetos haciéndoles colisionar dentro del cubo, y usar un lápiz para mover los objetos 3D alrededor.

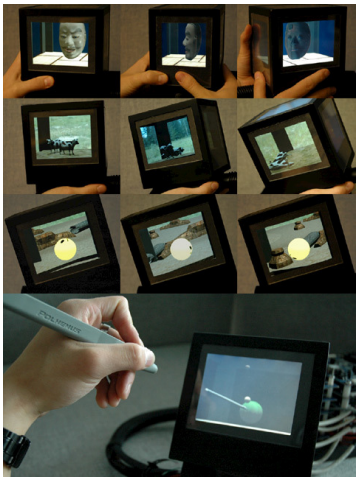

Figura 3.14: Visualizaciones en pCubee: imagen estática, dinámica, navegación por paisaje y selección directa de objetos mediante puntero.(Fuente: pCubee: A Perspective-Corrected Handheld Cubic Display [44])

Capítulo 4

Interacción Persona-Ordenador e Interfaces

A lo largo de la historia la interfaz ha ido evolucionando ofreciendo un amplio abanico de opciones, especializándose y diversificándose para dar lugar a nuevos modelos de interacción, como son las basadas en soportes móviles, interfaces gráficas, tangibles, de realidad aumentada, las interfaces cooperativas y colaborativas, etc. Todas ellas poseen una serie de ventajas y restricciones a la hora de adaptarse a distintos entornos y situaciones.

4.1 Interación Persona-Ordenador

La Interacción Persona-Ordenador (IPO o HCI: Human Computer Interaction), es la disciplina enfocada al estudio de la interacción realizada entre sistemas computacionales y las personas. El objetivo principal es el de mejorar esta interacción, consiguiendo que los sistemas computacionales sean más manejables, de manera que la interacción sea lo más intuitiva posible.

La ACM[1] define IPO como: *La disciplina encargada del diseño, evaluación e implementación de sistemas computacionales interactivos para uso humano y del estudio de los que rodea.*

El área de investigación de IPO, es aplicada en el mundo empresarial y académico, siendo una de las razones de cambio en el uso de la informática, mejorando los sistemas a medida que se localizaban nuevas necesidades de usuarios. Un ejemplo de ello son las interfaces gráficas de usuario, que revolucionaron y aumentaron las posibilidades de utilización en el uso de ordenadores. A día de hoy para el desarrollo de interfaces software, es posible hacer uso de bibliotecas y paquetes ya existentes, que facilitan la programación de los desarrolladores. El crecimiento de World Wide Web (WWW) es el resultado directo de investigaciones IP0.

La evolución de esta disciplina es debido principalmente a factores como:

- Creatividad humana: factor fundamental que determina qué necesidad se desea cubrir, y como sería del diseño de la misma. Ya en los primeros pasos dados en la ciencia de la informática, visionarios realizaron proyecciones imaginarias de cómo serían los

[1] Association for Computing Machinery.

ordenadores, y que funciones podrían ser capaces de llevar acabo.

- El estado del arte en la tecnología: a menudo actuá como límite al diseño.
- El mercado que engloba los ordenadores: está directamente en relación con el precio de los elementos, y que cantidad de usuarios utilizarán dicho producto, además del uso que harán de él.

Según Shackel [45], a principios de los años 50, ordenadores tenían como objetivo la investigación, pensados principalmente para resolver problemas matemáticos y científicos, donde la fiabilidad de los cálculos era primordial. En la década de los 60 y 70 son fabricadas las primeras macrocomputadoras, dirigidas a profesionales en el tratamiento de datos. Estas máquinas tenían unos altos costes de fabricación y poca flexibilidad, además de las dificultades a la hora de ser programadas. En los años 70, las minicomputadoras aparecen en el mercado, enfocadas a ingenieros y otros profesionales, y donde aún la programación es elevada y compleja. Sobre los años 80 aparecen los ordenadores dirigidos a todo tipo de consumidores, donde la usabilidad toma especial importancia, ya que intenta satisfacer las necesidades y requerimientos que demandan los usuarios. Durante la siguiente década, surgen nuevos dispositivos de menor tamaño, donde la usabilidad sigue siendo su principal característica, presentando nuevas dificultades a los profesionales e investigadores de la IPO.

IPO analiza conjuntamente los aspectos que dependen del ordenado, y los que son humanos. Dependiendo del punto de vista desde que se realiza el estudio, el enfoque es tratado desde los siguientes contextos:

- Contexto humano: es completado por ciencias tales como psicología, ciencias cognitivas, de comunicación, de diseño gráfico e industrial, entre otras.
- Computadoras y maquinaria: comprende los lenguajes de programación, gráficos por ordenador, sistemas operativos, y desarrollo de ambientes.

En la Figura 4.1, se muestran las disciplinas que intervienen en IPO.

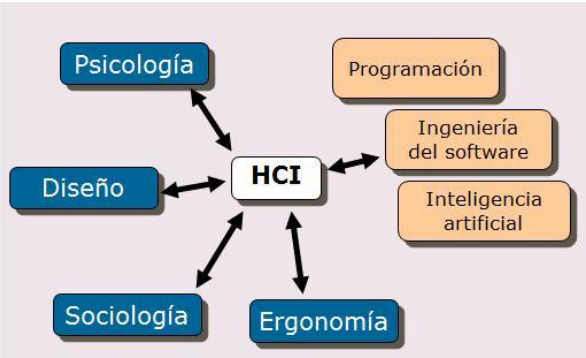

Figura 4.1: Ciencias que se relacionan con el HCI. (Fuente: Interacción tangible en aplicaciones educativas [46])

La interacción entre personas y tecnología se realiza por medio de un componente implícito: **La Interfaz**. La perfecta interfaz debe entender las necesidades de los usuarios, sus estilos de interacción natural, apoyándoles para eliminar los problemas en el uso de herramientas computacionales. Para el correcto desarrollo de una interfaz, es preciso comprender todos los factores implicados (organizativos, ergonómicos, psicológicos, y sociales), los cuales determinan la manera de actuar de los usuarios y como ellos hacen uso de los ordenadores, para conseguir desarrollar herramientas donde los sistemas diseñados satisfagan los requisitos del usuario, y así, conseguir una interacción eficiente, efectiva y segura.

4.2 Interfaz Gráfica de Usuario GUI

La interfaz gráfica de usuario *GUI*[2] hace uso de un conjunto de imágenes y objetos de forma gráfica, para mostrar la información proporcionando un entorno visual sencillo, que facilita la interacción *HCI*[3]. Algunas de estas *GUI*, son diseñadas para un uso específico, como son las pantallas táctiles, que simplifican aún más la interacción *HCI* mediante el sentido del tacto. La interfaz entre personas e información digital requiere de dos componentes fundamentales: la entrada/salida o control, y representación. Los controles permiten a los usuarios manipular la información, mientras que las representaciones externas son percibidas por medio de los sentidos. La Figura 4.2, muestra el modelo básico de una interacción con interfaz gráfica. En ella se hace uso de un elemento de entrada, como controlador remoto (ratón de ordenador), cuya información digital es procesada, para ser representada de manera intangible, usando un monitor, o produciendo un sonido determinado. Es decir, el usuario interactúa mediante un dispositivo a distancia y, en última instancia, experimenta una representación intangible de información digital (pixeles y sonido).

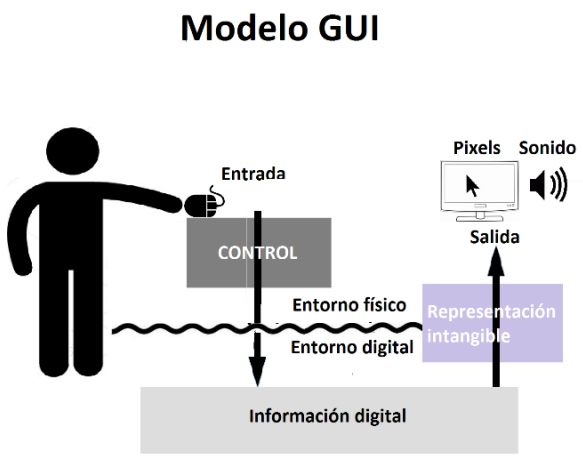

Figura 4.2: Modelo de interacción de las interfaces gráficas (Adaptado de Ishii [47])

[2]del inglés, Graphical User Interfaz
[3]Human-Computer Interaction

4.3 Interfaz Tangible de Usuario TUI

Las interfaces de usuario tangibles *TUI* (Tangible User Interface), en su área de estudio, tiene como objetivo analizar el paradigma de interacción *HCI*, no limitándolo a una simple pantalla. Hacer que la información sea comprensible y literalmente captable, es la finalidad principal de la interacción tangible, al proporcionar representación física de los datos digitales mediante objetos que los representen *TUIOs* (Tangible User Interface Objects). Estos objetos pueden ser utilizados con la manipulación natural de los usuarios, realizando un puente de unión entre el mundo digital y el mundo real.

El objetivo detrás de las interfaces tangibles, es permitir la interacción con las computadoras a través de objetos familiares, combinando la experiencia del usuario en el mundo táctil, con el poder de la tecnología [47]. Tal interacción física es básicamente de tipo unidireccional, dirigida desde el usuario al sistema, limitando los posibles patrones de interacción. En otras palabras, el sistema no tiene medios para apoyar activamente la interacción física. La Figura 4.3 ilustra la idea clave de hacer uso de la representación tangible (física y captable) como control.

Esta representación tangible ayuda a superar la barrera entre lo físico y lo digital. En contra del modelo *GUI* propuesto en la Figura 4.2, que hace uso de un ratón como elemento de control, en este caso se hace uso de un prototipo tangible, que intenta incorporar la información digital en forma física. A través de la manipulación física de las representaciones tangibles, la información digital es alterada, mostrando dichos cambios, bien mediante salida en la representación tangible, como en la salida de la representación intangible.

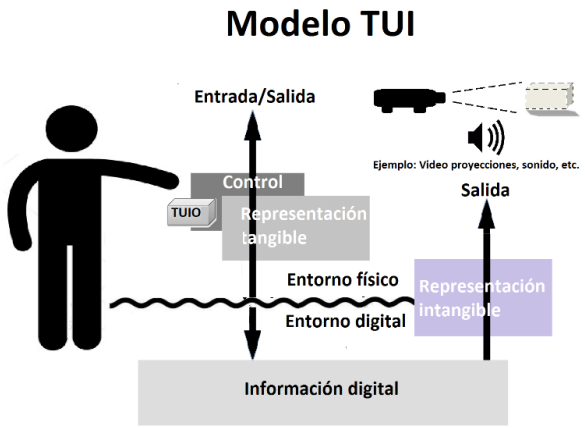

Figura 4.3: Modelo de interacción de las interfaces tangibles (Adaptado de Ishii [47])

Aunque la representación tangible permite que la realización física se acople directamente a la información digital, tiene capacidad limitada para representar el cambio de muchas propiedades materiales o físicas. A diferencia de los píxeles en la pantalla de un ordenador, es muy difícil cambiar un objeto físico en su forma, posición o propiedades (por ejemplo,

color, tamaño) en tiempo real. Para complementar esta limitación, *TUI* también utiliza representaciones tales como proyecciones de video y sonidos, para acompañar las representaciones tangibles en el mismo espacio, dando expresión dinámica de la información digital subyacente. El éxito de una *TUI* a menudo se basa en un equilibrio y fuerte acoplamiento perceptual entre las representaciones tangibles e intangibles. Tanto las representaciones tangibles e intangibles se acoplan perceptualmente para lograr una interfaz transparente, que media activamente la interacción con la información digital y borre, de manera apropiada, el límite entre lo físico y lo digital. La coincidencia de entrada y salida junto con una respuesta en tiempo real, son requisitos importantes para lograr este objetivo.

4.3.1 Uso de TUI en la educación

Los sistemas tangibles se están convirtiendo en una alternativa a la interfaz gráfica de usuario. Diferentes marcos de diseño se han aplicado al desarrollo de estos sistemas, abriendo la puerta a nuevas formas de relacionarse con el aprendizaje [48].

Un diseño tangible, ofrece libertad para poder explorar y manipular objetos físicos y observar que efectos producen sobre el mundo digital. La teoría del aprendizaje y la cognición justifica la asimilación de los conceptos teóricos donde se incluyen las prácticas de participación, construcción de modelos, y la actividad colaborativa entre otros. El aprendizaje realizado por medio de elementos tangibles tiene el potencial de que los niños pueden combinar lo conocido y familiar, en nuevas formas [15].

En el área del cálculo y sus conceptos abstractos (en los primeros años de educación de los niños), existen varios argumentos acerca de los beneficios de la manipulación de elementos tangibles. Esta manipulación de elementos físicos donde interviene la memoria visual podría guiar a la resolución de problemas. Este hecho pone de manifiesto, que los niños pequeños pueden llegan a reconocer ciertas cosas sin ser capaces de expresare a través del lenguaje, o sin ser capaces de reflexionar sobre lo que pueden llegar a conocer en un sentido explícito [49].

4.4 Interfaces Sistema Multi-Pantalla

Un sistema multi-pantalla o multi-monitor, consisten en combinar varios dispositivos de visualización con el fin de aumentar el espacio visual disponible para ejecutar una o varias tareas. Inicialmente este tipo de interfaz fue diseñada con el propósito de mostrar la misma imagen en los distintos dispositivos conectados. Este tipo de uso era aplicado principalmente en las presentaciones, donde el usuario podría disponer de un duplicado de la imagen proyectada.

De manera posterior, los fabricantes aprovecharon esta tecnología para ampliar las apli-

caciones posibles, incorporando visualizaciones independientes, aumentando el rango de visión de las aplicaciones, etc.

Son muy distintos los ámbitos de aplicación de esta tecnología, pudiéndose incorporar en los sistemas de control en la industria, sistemas de video-vigilancia, fabricación de juegos, etc. El uso de varias pantallas en la industria de los videojuegos fue introducida por Nintendo en 1980 con su serie de consolas portátiles *Game and Watch*.

Capítulo 5

Tecnologías para el desarrollo. Hardware y Software.

En este capítulo son detallados los componentes *hardware* que han sido utilizados para realizar la plataforma de juego, así como las funciones y datos técnicos de los componentes. Son detallados los pasos a seguir para la configuración, instalación y conexionado de cada uno de los dispositivos. Se exponen el conjunto de herramientas *software* empleadas para desarrollar los programas que manejan la plataforma de juego, cuya finalidad es conseguir un entorno gráfico, tangible e intuitivo para el usuario final.

5.1 Hardware.

Se conoce como *hardware*, al conjunto de elementos físicos que trabajan conjuntamente como unidad, y que componen un sistema informático.

El cometido principal de los dispositivos *hardware*, es automatizar la plataforma de juego aplicando la electrónica, para controlar distintas funciones y operaciones, trabajando todos los elementos en unidad. Los dispositivos empleados en la plataforma de juego son los siguientes:

- Ordenador de placa reducida *Raspberry Pi*. Encargado de administrar y procesar ordenes y eventos dentro del sistema, así como controlar el funcionamiento del resto de los dispositivos *hardware*.
- Pantallas táctiles. Parte fundamental de la interfaz gráfica e interfaz tangible, para poder visualizar el desarrollo del juego.
- Sensores. Para mejorar la experiencia de juego, son incorporados un sensor de color, y una unidad de medición inercial.

5.1.1 Raspberry Pi 3.

Raspberry Pi es un dispositivo *hardware*, de placa única y bajo coste, diseñado para promover la programación en edades tempranas, y empleado para diversas aplicaciones en proyectos de electrónica [50].

El proyecto *Raspberry Pi*, fue creado en 2006 en Reino Unido, por el grupo *Computer Lab* de la Universidad de Cambridge, con el objetivo de estimular la enseñanza de *Ciencias*

de la Computación en las escuelas [50].

En la actualizad este dispositivo se ha popularizado en todo el mundo, contando con una gran comunidad de usuarios, los cuales aportan tutoriales y documentación, que son de gran ayuda a la hora de emprender un proyecto con este dispositivo.

El modelo B de *Raspberry Pi 3*, utilizado para este proyecto, aparece en marzo de 2016. Incorpora un procesador diseñado específicamente para ese dispositivo, el *BCM2837 (quad-core Cortex-A53) a 1.2GHz*, que supone un cambio importante, al utilizar una arquitectura de 64bits, a diferencia de los modelos anteriores que aplican una arquitectura de 32bits. Este procesador, además, dispone de un módulo de implementación para las comunicaciones inalámbricas, el *BCM43438*. Incluye un *BCM WiFi 802.11n, Bluetooth LE y Bluetooth 4.1 Classic*. Este modelo dispone de un cabezal de 40 pines *GPIO*[1], dentro de los cuales se encuentran las conexiones para los buses de datos serie *SPI, I2C y UART*. La Figura 5.1 muestra la correspondencia con cada uno de los pines. La alimentación del dispositivo se realiza por medio de una fuente de alimentación externa (5VDC), vía Micro USB o *GPIO*.

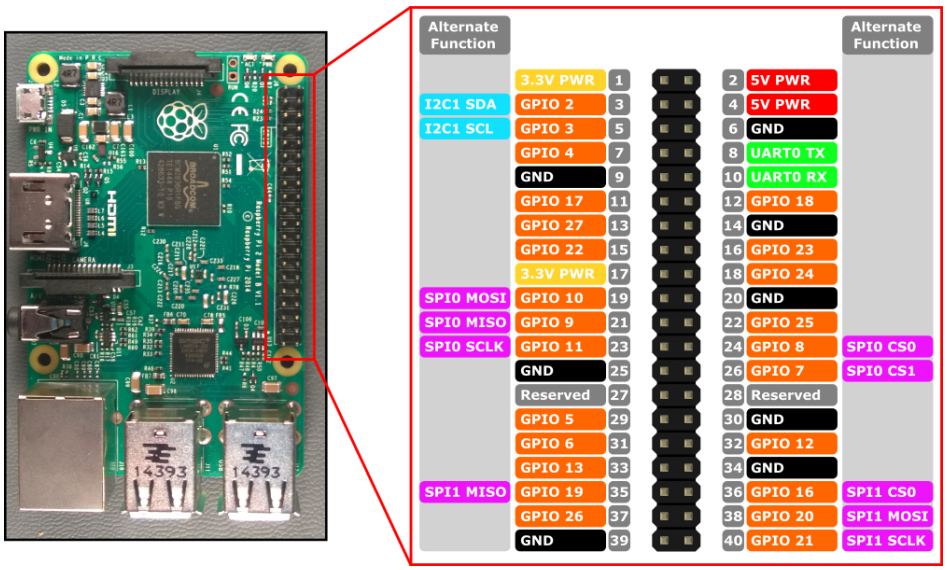

Figura 5.1: Pinout Raspberry Pi 3B. Fuente: [50]

Datos técnicos de *Raspberry Pi 3B*

A continuación, se muestran las características de la placa *Raspberry Pi 3B*:

- Procesador BCM2837 quad-core Cortex-A53 a 1.2GHz.
- VideoCore IV multimedia 400MHz .
- Memoria RAM de 1GB.
- BCM43438 WiFi 802.11n, Bluetooth LE y Bluetooth 4.1 Classic.

[1] General-Purpose Input/Output

- Controlador SMSC LAN9514 USB/Ethernet
- Cabecera 40 pines GPIO.
- Ranura MicroSD para memoria flash.
- Puerto DSI[2] dedicado a pantalla táctil de Raspberry Pi.
- Puerto CSI[3] para cámara de Raspberry Pi.
- Puerto de salida HDMI.
- Entrada de alimentación Micro USB.
- 4 puertos USB 2.0.
- Puerto Ethernet 10/100, RJ-45.
- Salida de audio compuesto, conexión tipo Jack de 3.5mm.

.

Las características han sido obtenidas de **Raspberry Pi User Guide [50]**

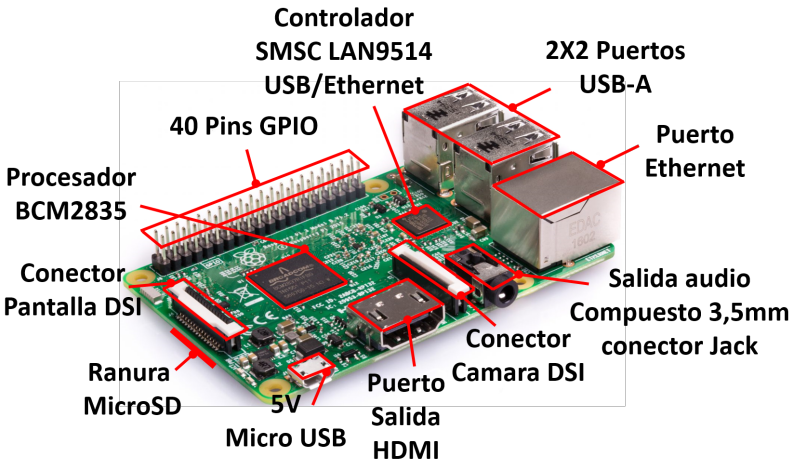

Figura 5.2: Elementos de la laca Raspberry Pi 3B. Fuente: Raspberry Pi User Guide [50]

5.1.2 Pantalla táctil capacitiva.

En la actualidad, la gran mayoría de los dispositivos comercialmente disponibles, y que hacen uso de una pantalla, usan la tecnología de contacto capacitivo. Estas tecnologías no solo ofrecen un entorno gráfico, sino que también simplifican la interacción *HCI* por el sentido del tacto.

Las pantallas capacitivas multitáctiles no han sido diseñadas para detectar objetos pasivos al ser situados sobre ellas, estas pantallas incluso disponen de filtros para rechazar de

[2]Display Serial Interface.
[3]Camera Serial Interface.

manera activa ese tipo de eventos táctiles. Los *widgets tangibles* en tales pantallas táctiles capacitivas, se han empezado a explorar relativamente hace poco más de una década[51], pero necesitan generalmente del cuerpo del usuario para proporcionar la capacitancia necesaria para la detección del tacto. La gran limitación es que, para mantener la detección, el usuario tiene que seguir tocando la superficie conductora del *widget*.

Las pantallas táctiles capacitivas, detectan la presencia de un conductor eléctrico conectado a tierra, por lo general un dedo humano, cerca de la superficie de la pantalla que es detectado mediante el uso de electrodos transparentes situados encima de la misma. Principalmente existen dos técnicas de detección: auto capacitancia y capacitancia mutua [52]. La capacitancia mutua es el método más comúnmente utilizado en pantallas de tipo multitáctil.

La configuración en la que se disponen los electrodos en una pantalla de capacitancia mutua consiste en un conjunto de filas y otro conjunto de columnas. Uno de los conjuntos actúa como transmisores (Tx) y el otro conjunto como receptores (Rx) [51].

Al aplicar una señal a uno de los electrodos Tx, la capacitancia entre ese electrodo Tx y otro electrodo Rx (que está entrecruzado) acopla la señal al electrodo Rx [53]. La señal de cada uno de los electrodos Rx es medida y mediante un controlador táctil es determinada la capacitancia entre el electrodo Tx activo y cada uno de los electrodos Rx. Al ser activado un electrodo Tx de manera simultánea (multiplexación en el tiempo), el controlador táctil es capaz de medir esta capacitancia en cada una de las intersecciones de los electrodos Tx - Rx en la pantalla. Cuando un conductor conectado a tierra como es el caso de un dedo se aproxima a una de estas intersecciones entre electrodos Tx-Rx, la capacitancia entre ambos electrodos se ve reducida cuando el campo eléctrico entre ellos es perturbado por el conductor [54].

El paso típico de un electrodo es de 5mm, al realizar un toque sobre la pantalla se verán afectadas más de una intersección. El controlador táctil mediante interpolación es capaz de determinar el centro del área tocada e informa de ello con un evento táctil.

La mayor parte de los controladores están diseñados para detectar toques con los dedos, aceptando formas elípticas del tamaño de la yema del dedo y rechazando e ignorando eventos táctiles con otras formas y tamaños que pueden reportar eventos táctiles impredecibles.

Resumiendo, para que el controlador informe de un evento táctil, la capacitancia del electrodo Tx-Rx ha de ser reducida por debajo de cierto umbral y ha de suceder a lo largo una elíptica que forma la yema de un dedo.

Funcionamiento de un *widget capacitivo*.

Se define *widget capacitivo* como un objeto tangible completo que contiene uno o más marcadores en su parte inferior y que comunican su posición a la superficie táctil de la pantalla capacitiva. Cada marcador consiste en una almohadilla redonda conductiva que es

detectada por la superficie de la pantalla.

Un marcador de *widget* para poder ser detectado como evento táctil sobre la superficie de la pantalla capacitiva, tiene que cumplir el requisito de conexión a tierra. Esto se puede conseguir mediante la capacitancia corporal de un usuario como propone Rekimoto [51]. Esto requiere que el usuario toque el *widget* y que las almohadillas del mismo sean conductivas, además de estar conectadas eléctricamente a la parte del *widget* que es tocada por el usuario. De esta forma el *widget* funciona simplemente como conductor eléctrico entre usuario y superficie táctil. Este enfoque tiene el inconveniente de tan pronto como el usuario deja de tener contacto, el controlador de la pantalla no puede detectar ninguna variación de capacitancia.

Otro enfoque que permite reemplazar al usuario como tierra eléctrica, es el uso de un cable conductor que conecta permanentemente el *widget* a un objeto relativamente conectado a tierra, como es el conector negativo de una batería.

Pantalla táctil capacitiva HDMI.

La pantalla multitáctil capacitiva utilizada en la plataforma de juego, y que es incorporada al dispositivo *TUIO1*, es el modelo *7inch HDMI LCD (B)* de la compañía *Waveshare* (ver Figura 5.3).

La conexión de la interfaz gráfica con la placa para el desarrollo es tipo *HDMI*, por su compatibilidad y facilidad de uso con *Raspberry Pi*. Los eventos generados en el panel táctil son transmitidos desde el puerto *Micro USB-B* de la propia pantalla, a el puerto *Micro USB-B* de la placa *Raspberry Pi*.

Características principales de la pantalla:

- 800x400px alta resolución.
- Control táctil capacitivo.
- Interfaz HDMI para la visualización.
- Interfaz Micro USB-B en el control táctil.
- Protocolo estándar HID para la integración en el sistema.
- Soporte para sistemas Raspbian.

Circuitos y componentes integrados principales para el funcionamiento de la pantalla (Figura 5.4):

1. **Texas Instruments TFP401APZP.** Controlador interfaz digital de pantalla DVI[4].
2. Atmel AT24MAC602. Memoria EEPROM[5].

[4]Digital Visual Interface
[5]Electrically Erasable Programmable Read-Only Memory

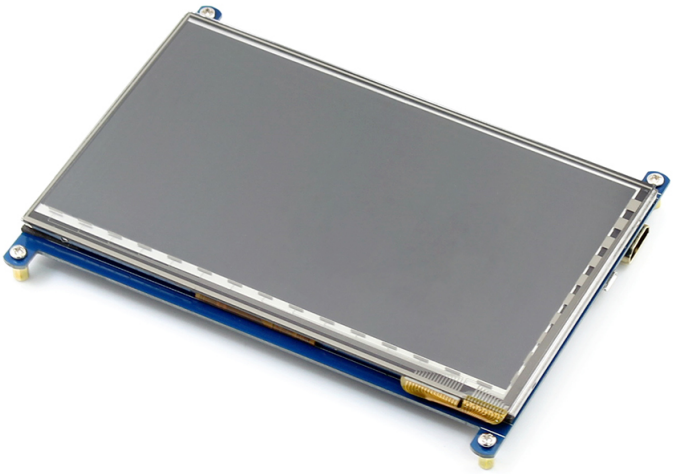

Figura 5.3: Pantalla táctil capacitiva 7 pulgadas Waveshare. Fuente: Waveshare Electronics [55]

3. Goodix GT811. Controlador interfaz panel táctil.
4. ST Microelectronics STM32F103. Procesador ARM 32-bit Cortex-M3 CPU core.
5. AMS1117. Regulador de voltaje.
6. Conector FFC[6] panel táctil .
7. Conector FCC pantalla.
8. Puerto HDMI.
9. Puerto Micro USB-B.
10. Micro interruptor de luz de fondo.

Instalación de los controladores de la pantalla en Raspbian.

La instalación de los *drivers* dentro del sistema operativo *Raspbian*, se realiza por línea de comandos, ejecutando los siguientes pasos:
Obtener la última versión de los controladores desde el repositorio Git.

```
git clone https://github.com/goodtft/LCD-show.git
```

Cambiar los permisos de lectura, escritura y ejecución de `LCD-show` y entrar al directorio.

```
chmod -R 755 LCD-show
```

```
cd LCD-show/
```

Ejecutar la instalación de los drivers, en este caso, la version B de la pantalla 7.0 pulgadas.

[6]Flexible Flat Cable

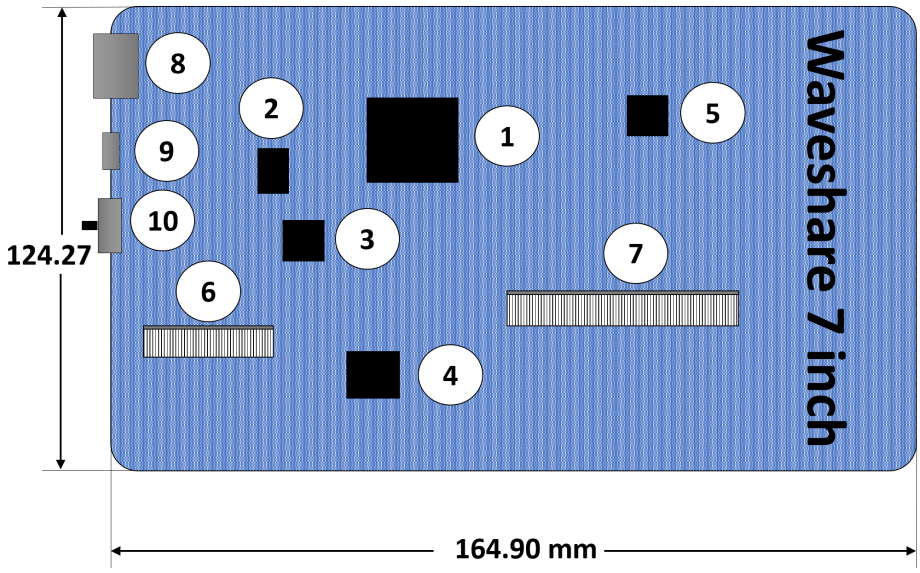

Figura 5.4: Componentes panel táctil capacitivo 7 pulgadas.

```
sudo ./LCD7B-show
```

Después de reiniciar el sistema, la pantalla puede ser utilizada en Raspbian.

Las características y proceso de configuración, han sido obtenidos de **Waveshare Electronics [55]**

5.1.3 Pantalla táctil resistiva HDMI.

Una pantalla táctil resistiva está formada por un panel de vidrio o acrílico, recubierto con dos capas eléctricamente conductoras y resistivas, con una separación entre ambas, y fabricadas con óxido de indio y estaño. Estas capas al ser presionadas producen un cambio en su resistencia. La producción de ente tipo de tecnología es más económico que el tipo capacitivo, con el inconveniente de solo detectar un evento táctil simultáneo [56].

La pantalla resistiva utilizada en la plataforma de juego, y que es incorporada al dispositivo *TUIO2*, es el modelo *3.5inch HDMI LCD* de la compañía *Kedei*(ver Figura 5.3).

La conexión de la interfaz gráfica con la placa para el desarrollo es *HDMI*, por su compatibilidad y facilidad de uso con *Raspberry Pi*. Los eventos generados en el panel táctil son transmitidos a la placa por comunicación *SPI*.

Características principales de la pantalla:

- 480x320px alta resolución.
- Control táctil resistivo.
- Interfaz HDMI para la visualización.
- Interfaz SPI en el control táctil.

- Protocolo estandar HID para la integración en el sistema.
- Soporte para sistemas Raspbian.

Figura 5.5: Componentes pantalla táctil capacitiva 3.5 pulgadas

Circuitos y componentes integrados principales, para el funcionamiento de la pantalla (Figura 5.6):

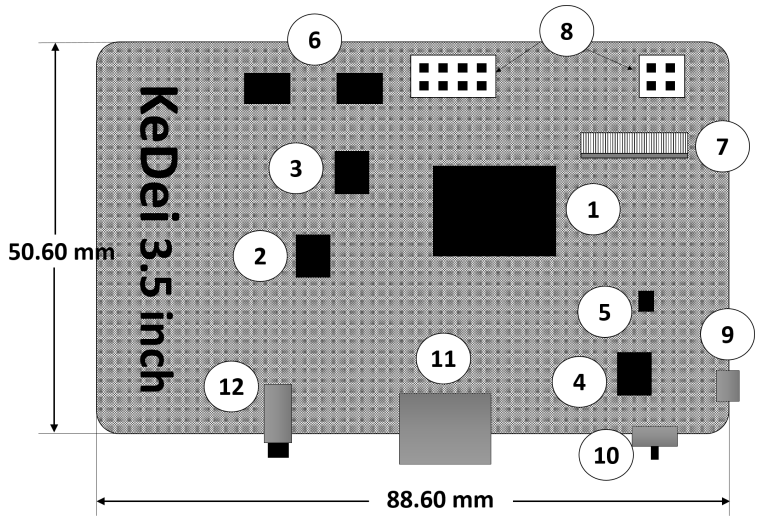

Figura 5.6: Componentes pantalla táctil capacitiva 3.5 pulgadas

1. **Realtek RTD2660H.** Controlador interfaz digital de pantalla.

2. **Winbond W25X40CLNIG.** Memoria Flash 4Mbit SPI.
3. **Cirrus Logic CS4334.** Convertido estéreo D/A 24bit 96kHz con interfaz I2S.
4. **Shenzhen Xptek Technology XPT2046.** Controlador interfaz panel táctil con interfaz I2C.
5. **PowerTech PT4103.** Controlador para leds por ancho de pulsos PWM [7].
6. **AMS1117.** Regulador de voltaje.
7. Conector FCC pantalla.
8. Conector GPIO (Alimentación y panel táctil).
9. Puerto Micro USB-B.
10. Interruptor de luz de fondo.
11. Puerto HDMI.
12. Puerto salida de audio.

Instalación de los controladores de la pantalla en Raspbian.

El primer paso para la instalación de los *drivers*, es descargar la última versión en la dirección: *http://osoyoo.com/driver/LCD_show_35hdmi.tar.gz.*
Desde el directorio de descarga, y por medio de línea de comandos, se cambian los permisos de ejecución:

```
sudo chmod 777 LCD_show_35hdmi.tar.gz
```

Descomprimir el archivo y entrar dentro de la carpeta donde se encuentran los controladores:

```
tar -xzvf LCD_show_35hdmi.tar.gz
cd LCD_show_35hdmi
```

Instalar el controlador adecuado para la resolución de la pantalla (480x320):

```
sudo ./LCD35_480*320
```

Después de reiniciar automáticamente el sistema, la pantalla puede ser utilizada en *Raspbian*.

5.1.4 Dispositivos de entrada. Sensores.

Un sensor es un dispositivo que, a partir de la energía del medio donde se mide, da una señal de salida transducible, que es función de la variable medida. Es decir, un sensor es capaz de convertir una señal mecánica, a una señal de tipo eléctrico [57].

Unidad de medición inercial MPU9255.

Definimos un sensor inercial como aquel que es usado para medir las variaciones de velocidad lineal, angular y campo magnético, producidas en una masa. Un sensor inercial por

[7] pulse width modulation

tanto dispone de tres transductores de señal: acelerómetro (medidas de velocidad lineal), giróscopo (medición de velocidad angular), y magnetómetro (medidas de la variación del campo magnético).

Habitualmente son utilizados como componentes fundamentales en sistemas de navegación donde es necesario estimar mediciones de posición, posición y velocidad de un cuerpo, en ausencia de referencias externas.

La unidad de medición inercial utilizada es *MPU9255* de la compañía *InvenSense*. Este dispositivo electrónico es la combinación de dos sensores, *MPU-6500* y *AK8963*. El sensor *MPU-6500* contiene un giroscopio de tres ejes y un acelerómetro de tres ejes, capaz de tomar mediciones de aceleraciones y velocidades angulares, de un cuerpo al que este fijado. Está complementado con un procesador digital de movimiento *DMP*[8]. *AK8963* es un sensor de medición del campo magnético gravitacional (magnetómetro) digital. La comunicación con los registros del dispositivo se realiza utilizando el bus de datos *I2C* a 400kHz.

Especificaciones generales del MPU9255:

- Voltaje de operación (VDD): 2.4V – 3.6V
- Voltaje de interfaz digital (VDDIO): 1.71V – VDD
- Rango de temperatura: -40ºC – 85ºC
- Frecuencia de operación I2C: 400kHz
- Bus auxiliar I2C maestro para leer datos de sensores externos (p. Ej. Sensor de presión).
- Corriente de operación de 3.5mA cuando los 9 ejes de detección de movimiento y el DMP están habilitados.
- Rango de voltaje VDD: 2.4 - 3.6V.
- Voltaje de referencia VDDIO para dispositivos I2C auxiliares.
- Memoria intermedia FIFO de 512 bytes que permite al procesador leer los datos en ráfagas.
- Sensor de temperatura de salida digital.
- Filtros digitales programables por el usuario para giroscopio, acelerómetro y sensor de temperatura

Características del giroscopio:

- Sensores de velocidad angular ejes X, Y y Z de salida digital con un rango de escala completa programable por el usuario de $\pm 250, \pm 500, \pm 1000, \pm 2000 \circ / seg$ y controladores integrados de 16 bits.

[8]Data Management Platform

- Filtro de paso bajo programable digitalmente.
- Corriente de funcionamiento: 3.2mA.
- Modo de reposo: $8\mu A$.

Características del acelerómetro:

- Acelerómetro de triple eje, salida digital con un rango programable de escala completa de $\pm 2g, \pm 4g, \pm 8gy \pm 16g$ y ADC integrado de 16 bits.
- Corriente de funcionamiento normal del acelerómetro: $450\mu A$.
- Corriente del modo acelerómetro de baja potencia: $8{,}4\mu A$ a 0.98Hz, $19{,}8\mu A$ a 31.25Hz.
- Modo de reposo: $8\mu A$

Características del magnetómetro:

- Sensor magnético de *efecto Hall* monolítico de silicio de 3 ejes con concentrador magnético.
- Resolución de datos de salida de 14 bit ($0{,}6\mu T/LSB$) o 16 bit ($15\mu T/LSB$).
- El rango de medición a escala completa es de $4800\mu T$.
- Corriente de funcionamiento normal del magnetómetro: $280\mu A$ a una frecuencia de repetición de 8Hz.

El sensor permite comunicarse por dos buses distintos: *I2C* y *SPI*. El dispositivo *TUIO2* emplea las comunicaciones *SPI* con la interfaz del panel táctil, por lo que para simplificar se emplea las comunicaciones *I2C*.

Descripción de las señales y pines de salida para una configuración I2C(Figura 5.7 y Figura 5.8):

1. **VCC:** Entrada voltaje positivo fuente de alimentación del circuito.
2. **GND:** Terminal negativo de la fuente de voltaje de alimentación.
3. **SCL:** Señal de reloj serie I2C.
4. **SDA:** Entrada de datos serie I2C.
5. **nCS:** Selector de chip (solo para comunicaciones SPI).
6. **AD0:** Dirección esclavo I2C.
7. **INT:** Entrada interrumpir suspensión del sistema.
8. **FSYNC:** Entrada digital de sincronización de *frames*.
9. **AUX-CL:** Señal de reloj I2C para conectar sensores externos al dispositivo.
10. **AUX-DA:** Señal de entrada de datos serie I2C, para conexiones de sensores externos.

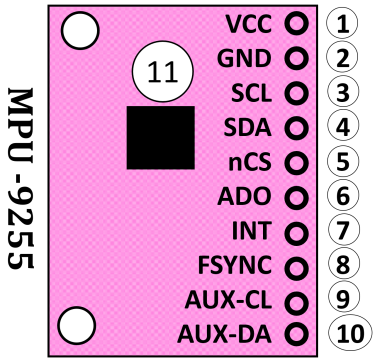

Figura 5.7: Pines de la unidad de medición inercial MPU9255.

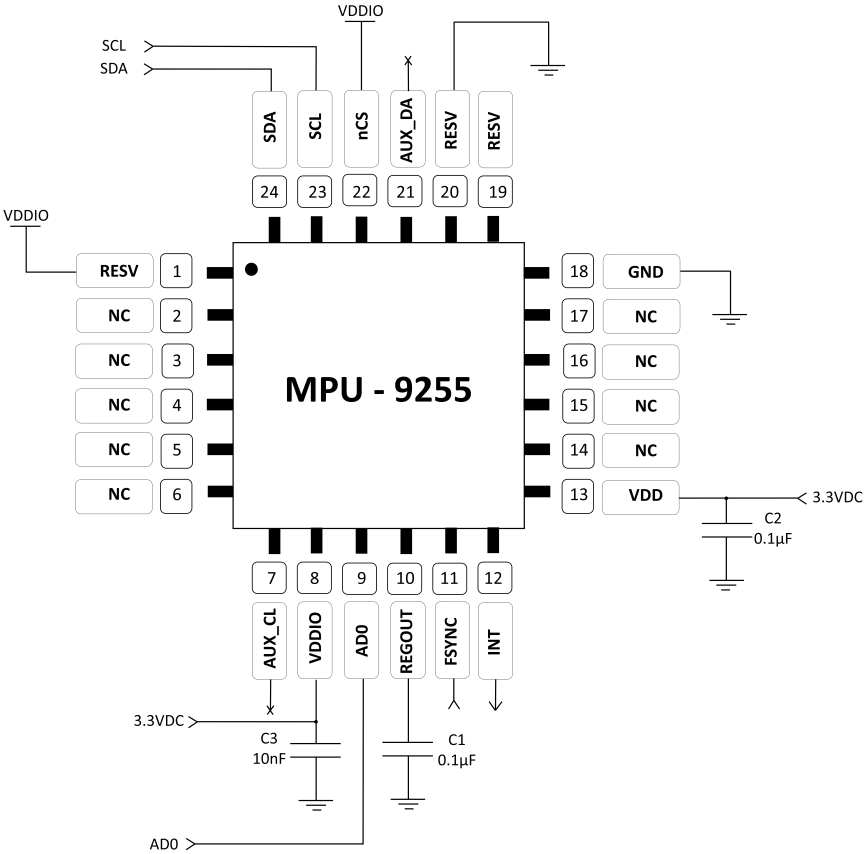

Figura 5.8: Esquema para una aplicación con bus I2C en MPU9255.

Los datos técnicos, así como la descripción de sus componentes, han sido obtenidos de la documentación facilitada por el fabricante [58].

Sensor de color.

5.2 Software para el desarrollo.

5.2.1 Sistema Operativo GNU/Linux. Raspbian.

GNU/Linux es un sistema operativo que utiliza el *Kernel Linux* como núcleo. Fue creado y difundido, a través del Proyecto *GNU* por la *Free Software Foundation*, fundada por Richard Stallman, uno de los principales percusores del Software Libre . Las distribuciones basadas en el Kernel Linux, destacan: *Ubuntu, Kubuntu, Fedora, ArchLinux, Debian*, entre otros [50]. *Raspbian* es un sistema operativo basado en la distribución *Debian*, recomendado para *Raspberry Pi* por estar optimizado para su hardware, siendo la más estable y con mayor rendimiento. Dispone de conjuntos de paquetes (por defecto incorpora más de 35000 paquetes) con fines específicos para este tipo de dispositivos [50].

Figura 5.9: Logotipo Raspbian/Debian. Fuente: Página Oficial Raspberry Pi [59].

Instalación de Raspbian

Para instalar el sistema operativo Raspbian, es necesario descargar y descomprimir en una tarjeta microSD (con formato VFAT), la versión de NOOBS 1.9 desde la página oficial de *Raspberry Pi Foundation [60]*. Al realizar el primer arranque, el menú de NOOBS (Figura 5.10 instala de manera automática *Raspbian*. Al reiniciar, el sistema entra automáticamente en modo gráfico. El siguiente paso es habilitar las interfaces *I2C* y *SPI* (ver Figura 5.11), ejecutando la aplicación de configuración de *Raspbian*, desde la línea de comandos (la aplicación de por defecto es `LXTerminal`), por medio del siguiente comando:

```
raspi-config
```

La habilitación de estas dos interfaces, permite comunicar diferentes dispositivos que utilizan este tipo de *bus de datos*. En nuestro caso, manejo de eventos táctiles, interfaz gráfica, y sensores. De manera adicional, se procede a instalar diferentes paquetes que servirán de ayuda para realizar la plataforma de juego, como son: `i2c-tools, python-smbus, python3-smbus, python-dev, python3-dev python-gpiozero`

La documentación y los datos técnicos, así como el procedimiento de instalación, ha sido obtenido de la guía *Raspberry Pi User Guide [50]*.

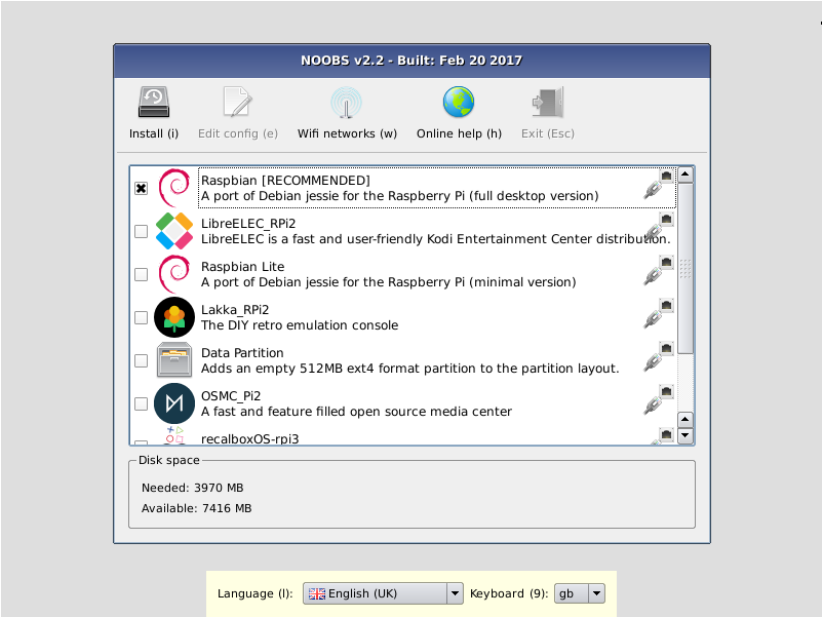

Figura 5.10: Menú de instalación NOOBS.

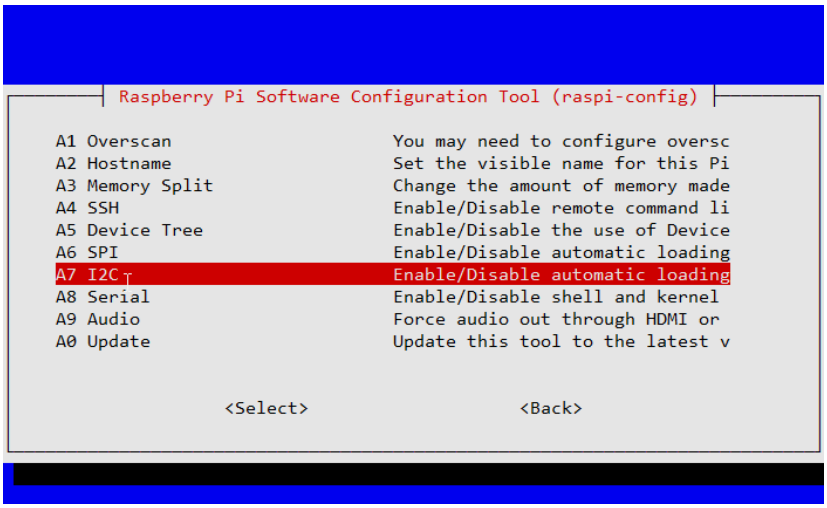

Figura 5.11: Pantalla de configuración de las interfaces I2C y SPI.

5.2.2 Python.

Python es un lenguaje de programación de alto nivel, multiplataforma, de tipado dinámico, multiparadigma, diseñado para ser ejecutado por intérprete. Los lenguajes de alto nivel están orientados a ofrecer al usuario, distintos algoritmos que puedan ser leídos, reorganizados e interpretados de forma sencilla, por ser más parecidos al lenguaje y a la lógica humana. Son independientes de la arquitectura del *hardware*, por lo que asumen mayor portabilidad [61].

La característica de tipado dinámico indica que una variable no requiere ser definida asignando su tipo de datos. Estos valores son asignados automáticamente según el valor decla-

rado. Se trata de un lenguaje de programación multiparadigma, al soportar orientación a objetos, programación imperativa y programación funcional [62] .

Puede ser interpretado en diversos *Sistemas Operativos* como *GNU/Linux, Windows, Mac OS*. El código fuente debe ser escrito de forma estandarizada según unas reglas de estilos recogidas en *Python Enhancement Proposal [63]*.

Figura 5.12: Logo de Python. Fuente: Python [61]

5.2.3 Kivy.

Kivy es una biblioteca de *Python* de código abierto y multiplataforma, la cual ofrece un conjunto de herramientas para el desarrollo de interfaces gráficas de usuario en aplicaciones donde se requiere detectar varios eventos táctiles simultáneos. *Kivy* está escrito en *Python* y *Cython*, basado en *OpenGL ES 2*, soporta varios dispositivos de entrada y tiene una extensa biblioteca de widgets. Con la misma base de código puede orientarse a *Windows, OS X, Linux, Android e iOS*. Todos los *widgets* en *Kivy* están construidos con soporte multitáctil [64].

El lenguaje **Kv (Kivy)** permite crear *widgets* de forma declarativa, y vincular las propiedades de los *Widgets* entre sí. Ofrece una estructura del lenguaje que facilita cambios ágiles en la interfaz de usuario. Ofrece una buena separación entre la lógica de la aplicación y la interfaz de usuario.

Figura 5.13: Logo de kivy. Fuente: Kivy [64]

Instalación de Kivy en Raspbian.

Instalación de las dependencias:

```
sudo apt-get update
sudo apt-get install libsdl2-dev libsdl2-image-dev libsdl2-mixer-d libsdl2-ttf-dev \
     pkg-config libgl1-mesa-dev libgles2-mesa-dev \
     python-setuptools libgstreamer1.0-dev git-core\
     gstreamer1.0-plugins-{bad,base,good,ugly} \
     gstreamer1.0-{omx,alsa} python-dev libmtdev-dev xclip xsel
```

Instalación de la última versión de *Cython*:

```
sudo pip install -U Cython==0.28.2
```

Instalar de manera global *Kivy*:

```
sudo pip install git+https://github.com/kivy/kivy.git@master
```

La documentación, datos técnicos, así como el procedimiento de instalación, ha sido obtenido de la guía *Kivy Documentation. User's Guide [64]*.

5.2.4 Interprete para el entorno de desarrollo Python Pycharm.

Python por defecto, incorpora su propio *Shell* interactivo, para escribir y ejecutar código. Sin embargo, se ha optado por la instalación de un entorno de programación alternativo: *PyCharm*. Este entorno ofrece ventajas sobre el *Shell nativo*, como por ejemplo, poder visualizar números de línea, sangrado automático, además de ejecutar partes de código y corregir los eventuales errores. *PyCharm* es un entorno de desarrollo integrado *(IDE)*[9], multiplataforma, desarrollado por la compañía *JetBrains*, para programación en lenguaje *Python* [65].

Características:

- Asistencia y análisis de codificación, con finalización de código , sintaxis y resaltado de errores.
- Navegación de proyecto y código: vistas de proyecto especializadas, vistas de estructura de archivos y saltos rápidos entre archivos, clases, métodos y usos.
- Depurador integrado de Python.
- Prueba de unidad integrada, con cobertura de código línea por línea.

[9] Integrated Development Environment

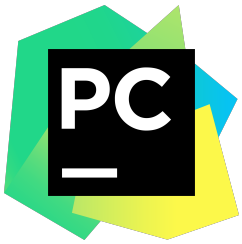

Figura 5.14: Logotipo PyCharm. Fuente: Pycharm [65]

Capítulo 6

Método de trabajo

Tras haber analizado cuales son los objetivos a desarrollar en el TFG, y realizado un análisis de las tecnologías existentes, es necesario asignar una metodología para la gestión del proyecto.

Para ello, y valorado el número de personas que están involucradas en el proyecto (director del TFG y su autor), se efectúa una investigación sobre cuales son las alternativas de metodologías para llevarlo a cabo, que sea ágil, ligera, y por supuesto, flexible a cambios para adaptar nuevos requisitos que puedan surgir.

Las metodologías Agile, surgen en el año 1990, como herramienta de gestión software que indicase unas correctas directivas para llevar a acabo los proyectos.

Las metodologías existentes hasta la época, tenían una rigurosa asignación de roles, actividades y artefactos (incluyendo el modelado y una documentación muy detallada), que, no obstante, a día de hoy sigue siendo necesaria para proyectos de gran envergadura, que necesitan una alta gestión de tiempo y recursos.

Según avanza el desarrollo de aplicaciones software, entran en escena nuevas situaciones en las cuales este tipo de métodos de gestión no encaja totalmente con estas metodologías tradicionales.

Los métodos Agile promueven una gestión de proyectos, que se basa principalmente en fomentar la constante inspección del trabajo y la adaptación de éste. Se trata de un sistema organizado que facilita el trabajo el equipo, una correcta organización, y favorece el rendimiento del tiempo empleado en el desarrollo.

Scrum fue creado, con las características de estos métodos de gestión, mediante una comunicación directa y haciendo uso de ingeniería concurrente, basándose en las ideas del «Manifiesto por el Desarrollo Ágil de Software» (detalladas en la siguiente sección del documento).

Por lo tanto, basándose en estas ideas, y teniendo en cuenta el número de personas que están involucradas en el proyecto, se ha elegido la metodología de trabajo Scrum.

6.1 Metodología Agile

6.2 Scrum como metodología de trabajo

6.2.1 Roles

6.2.2 Artefactos

6.2.3 Motivos de la elección del método Scrum

6.3 Aplicación del método de trabajo

6.3.1 Iteración 1: Inicio del TFG

Esta iteración tiene como finalidad, conocer el alcance del proyecto y los puntos a tratar en la línea de trabajo, la cual está dedicada al área de electrónica de consumo y lúdico-educativa.

Establecer una propuesta de Trabajo Fin de Grado, así como los diferentes aspectos relacionados con la organización, planificación y seguimiento del proyecto.

El presente Trabajo Fin de Grado, será dirigido y supervisado como director **Francisco Moya Fernández**.

Descripción de la línea de trabajo.

La propuesta de este TFG consiste en el diseño y prototipado de un sistema electrónico de bajo coste, basado en interfaces tangibles, para aplicaciones lúdico-educativas. Para llevar acabo la plataforma de juego, se hace uso de sistemas empotrados basados en microprocesadores, junto a un conjunto de periféricos, aprovechando la tecnología para facilitar los mecanismos de aprendizaje. La elección de esta línea de trabajo ofrece la posibilidad de integrar y desarrollar los contenidos, capacidades y habilidades adquiridas durante todo el periodo de docencia en el **Grado en Ingeniería Electrónica Industrial y Automática**.

Organización y planificación del proyecto.

La metodología aplicada a la organización y planificación del proyecto es *Scrum*, la cual divide el proceso del desarrollo del proyecto en varias iteraciones (*Sprints*). Cada iteración tiene como objetivo realizar una reunión de planificación, donde se definen y analizan las diferentes etapas que componen la evolución del proyecto expuestas como historias de usuario. Las historias de usuario contienen una descripción corta de las necesidades del proyecto. Cada una de ellas es redactada al final de cada reunión y analizadas al principio de la iteración siguiente. La duración de tiempo entre iteraciones establecido es de **dos semanas**.

Redacción, seguimiento y desarrollo del trabajo escrito.

La memoria es redactada mediante el editor de textos *TextMaker* basado en LaTeX, debido a la simplicidad para redactar documentos de forma estructurada. El trabajo escrito es alojado en un repositorio de la plataforma *github.com* para la supervisión y seguimiento del proyecto. Esto permite visualizar y realizar cambios dinámicos en la memoria por parte del autor y del

director del TFG.

Próximas historias de usuario.

Al finalizar la primera reunión son establecidas las historias de usuario para la próxima iteración.

- Estudio y primer análisis del problema.
- Elaborar una primera propuesta de diseño a partir del análisis previo del problema.

6.3.2 Iteración 2 : Formación y planificación del TFG.

Análisis y exposición de las distintas características que definen una plataforma de juego basada en interfaces tangibles. Se seleccionan y destacan aquellos aspectos que serán de utilidad, para ser aplicados a un primer diseño de sistema tangible.
Planteamiento y explicación del funcionamiento de la plataforma de juego basada en interacción tangible, tras haber realizado un primer estudio y análisis previo.

Estudio y primer análisis del problema.

Localizar y estudiar los principales sistemas basados en interacción tangible, para destacar sus principales características. Para ello, se hace uso de las diferentes bases de datos actualmente disponibles como *Scopus* o *Web Of Science*.

Características principales del modelo básico en una interfaz de usuario tangible:

- Elementos físicos de entrada/salida o control, y representaciones tangibles e intangibles.
- Información comprensible y literalmente palpable.
- Acoplamiento de las representaciones tangibles a la información digital latente, mediante la manipulación de objetos que sirven como puente de unión entre el mundo digital y el mundo real. Convertir la información digital en forma física.
- Equilibrio y fuerte acoplamiento perceptual entre representaciones tangibles e intangibles.

Propuesta de un primer dispositivo de juego a partir de las propiedades en sistemas tangibles.

La propuesta incorpora las características principales aplicadas a interfaces tangibles, como primera toma de contacto con este tipo de dispositivos. Estos sistemas, permiten a los niños escribir un programa mediante el uso de objetos físicos, sin hacer uso de un teclado, para posteriormente ver los resultados realizados por ellos mismos.

Modelo básico en las interfaces de usuario tangible.
Las propiedades más importantes en la interfaz entre personas e información digital son:

- Elementos de entrada/salida o control
- Representación tangible de la información digital como mecanismo de control físico interactivo directo.
- Representación intangible como herramienta de acoplamiento entre la realización física y la información digital.

Descripción.
El objetivo del juego es realizar una secuencia (propuesta por la propia aplicación o creada por el niño), a partir de elementos tangibles, que corresponden a unas instrucciones específicas, como pueden ser movimientos, representaciones gráficas, salidas acústicas, etc. El resultado de la secuencia es mostrado de manera gráfica en otro dispositivo.
Existen dos modos de funcionamiento dentro de la aplicación. El niño puede crear una secuencia aleatoria con los elementos tangibles para obtener una salida gráfica, o de manera inversa, proponer gráficamente al niño que realice una secuencia con los elementos tangibles.

El modelo propuesto incorpora las características básicas de las interfaces de usuario tangible, y presenta dos tipos de componentes principales:

- **Placa principal:** procesa y muestra gráficamente, las instrucciones recibidas del resto de componentes (elementos tangibles de entrada/salida). La comunicación con el resto de los elementos es ejecutada por una red inalámbrica de comunicaciones *WiFi*.
- **Elementos de entrada/salida:** corresponden a la representación intangible de la información digital. Está formado por una serie de bloques de pequeñas dimensiones y estructura cúbica. Estos cubos son conectados entre sí, para formar una secuencia. Cada elemento se comunica con el anterior, con un conector magnético *micro USB*. Cada cubo esta diferenciado por código de colores, para una interacción más intuitiva, e incluyen un indicador visual (*led RGB*), para indicar en que momento de la secuencia se encuentra. Las instrucciones son enviadas a la placa principal vía inalámbrica para ser procesadas. Cada elemento lleva a cabo únicamente una función. Existen tres funciones principales:
    - **Elemento de inicio.** Elemento principal de control imprescindible para poder crear una secuencia (ver Figura 6.1). Incorpora un microcontrolador el cual maneja un módulo de comunicaciones inalámbricas Wifi, para el intercambio de información con la placa principal. Dispone de un pulsador para iniciar la secuencia. El resto de los elementos tangibles deben ser conectados en *serie* a este módulo para establecer comunicación.

- **Elemento de eventos.** Genera un evento dentro de la secuencia (desplazamientos, temporizador, indicador luminoso, acústico, ...).
- **Elemento repetición.** Al situarse al final de una secuencia, retoma el inicio de esta (vuelve al elemento de inicio).

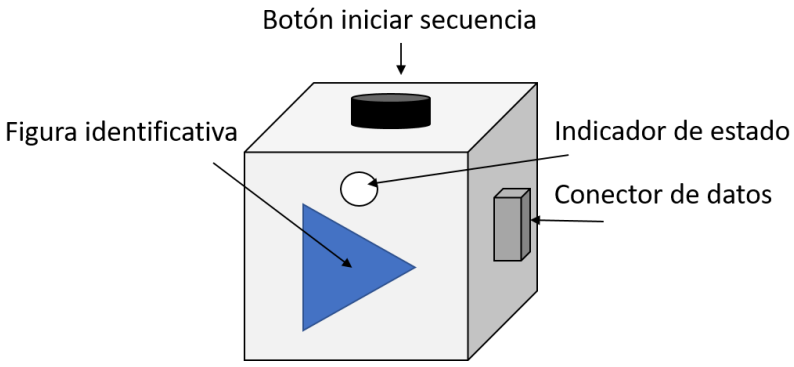

Figura 6.1: Diseño del elemento tangible *inicio*.

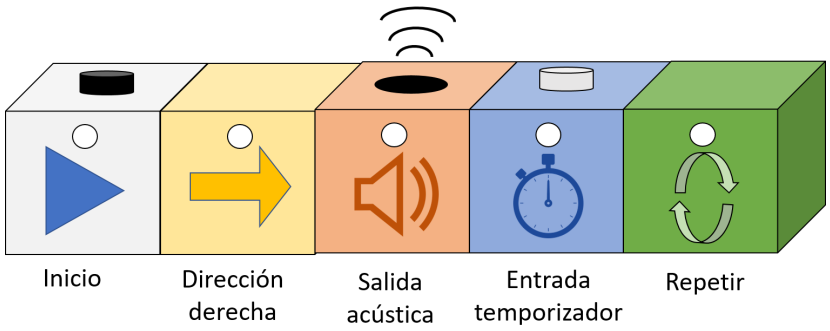

Figura 6.2: Ejemplo de secuencia. Al pulsar el botón *iniciar secuencia*, se produce un desplazamiento hacia la derecha. A continuación, se genera una señal acústica. La secuencia se mantiene durante un tiempo en pausa (según el tiempo ajustado en el temporizador). Finalmente se repite la secuencia.

Las comunicaciones entre módulos son realizadas por medio de un conector magnético *micro USB* que facilita la conexión entre los mismos. Existen 4 líneas de comunicación serie (entrada de datos, salida de datos, habilitación y señal de reloj). Incorpora un *registro de desplazamiento* y un *multiplexor* para el manejo de datos (Figura 6.3). La trama de datos consta de 8 bits. Los 3 bits más significativos, identifican el tipo de instrucción asignada al cubo. El resto de los bits son utilizados para la iluminación de un *led RGB*, que indica el estado de la secuencia.

La Figura 6.2 muestra un ejemplo de una posible secuencia.

Análisis de la primera propuesta de diseño

La primera propuesta de diseño es analizada junto con el director del proyecto, quedando

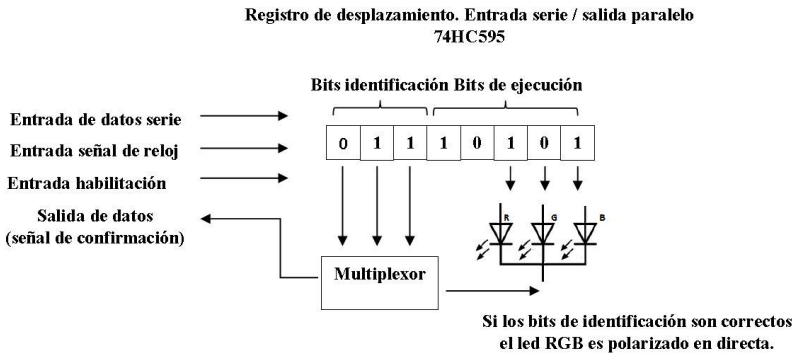

Figura 6.3: Diseño del manejo de datos en los elementos tangibles mediante registro de desplazamiento y multiplexor.

descartada. En el mercado existen dispositivos de características muy similares, no aportando nada destacable. Los conectores magnéticos no cumplen la normativa vigente para dispositivos en este tipo de aplicaciones, siendo peligroso para los niños. Se concreta conocer en profundidad el alcance del proyecto, entendiendo que hay que construir, realizando un estudio más amplio sobre los actuales sistemas enfocados a la programación tangible, tecnología y características. Identificar los puntos a mejorar en los actuales sistemas y contribuir a mejorarlos, diseñando una plataforma de juego que resuelva esos puntos. El análisis permite establecer unos requisitos de diseño.

Próximas historias de usuario.

- Estudio avanzado del problema. Revisión sistemática (recopilación de información).
- Identificar áreas de mejora en los actuales sistemas tangibles.
- Selección de las áreas de mejora para la redacción de los objetivos.
- Elaborar una segunda propuesta de diseño a partir del estudio avanzado del problema y de las áreas de mejora.

6.3.3 Iteración 3: Estudio avanzado del problema. Identificación y selección de las áreas de mejora en sistemas tangibles.

Determinar cuáles son los principales sistemas con interfaz tangible aplicados a la programación para niños, obteniendo las ventajas y desventajas, así como las principales características y tecnología utilizada en cada uno de ellos. Segunda propuesta de diseño de plataforma de juego haciendo uso del análisis desarrollado en esta iteración.

Estudio avanzado del problema.

Estudiar los diferentes enfoques que aportan los dispositivos actuales basados en interfaces tangibles. Destacar los sistemas aplicados al aprendizaje de la programación para niños. Análisis de las distintas tecnologías que utilizan. Recopilar información relevante referida

al tema tratado. Posteriormente, elaborar un capítulo dedicado a las distintas plataformas y dispositivos que emplean este tipo de interfaces, referenciando las fuentes consultadas en cada caso.

En el **Capítulo Antecedentes 3** de este documento, se muestran los resultados sobre los principales estudios experimentales aplicados a la interacción tangible, y como motiva a niños de tempranas edades en sus habilidades computacionales.

En ese capítulo, se indican los dispositivos que han sido utilizados para el aprendizaje computacional más importantes, destacando los diferentes enfoques que aportan cada uno de ellos en la interacción tangible.

Estudios experimentales sobre Interacción tangible.

A continuación, se exponen a modo de resumen, las citas más importantes destacadas por los distintos autores, en los artículos analizados:

- Exponer a los niños en edades tempranas a un pensamiento computacional, ayuda a construir una base sólida en la mecánica de programación de computadoras [2].
- El entorno de programación debe motivar fuertemente al usuario, colocándolo en un contexto que sea convincente y significativo, para que no se dé por vencido [47].
- Manipular objetos con las manos en lugar de modelos mentales, puede cambiar la forma en la que un niño recuerda y recupera la información enseñada [3].
- Las interfaces digitales se convierten en poderosas herramientas de aprendizaje cuando los niños se convierten en creadores en lugar de sólo consumidores de contenido digital [4].
- Al programar, los niños exploran los conceptos fundamentales de secuenciación, reconocimiento de patrones y relaciones de causa y efecto. Los niños aprenden a contar una historia de principio a fin, realizar secuencias de números y letras, formas o colores [5].
- Aprovechan las estrategias de aprendizaje kinestésico natural de los niños para hacer que los conceptos abstractos sean accesibles e intuitivos [6].
- Los sistemas tangibles tienen el prometedor potencial de ayudar a los niños a cultivar un cierto conocimiento del pensamiento computacional como abstracción, automatización, descomposición de problemas y análisis [17].
- Mientras los niños juegan, se involucran con el mundo físico haciendo uso de sus cinco sentidos [7].
- La interfaz tangible y la interfaz gráfica proporcionan un andamio estratificado a los niños mientras avanzan hacia un entorno de programación cada vez más auténtico [12].

- Los sistemas tangibles aplicados en la programación, fomentan el aprendizaje y el rendimiento en tareas computacionales [22].
- Los niños pequeños aprenden mejor mientras manipulan y transforman activamente materiales reales [24].
- La programación, depuración, el acto de construir un sistema, encontrar problemas y resolverlos, es la manera más eficaz de que un niño aprenda [25].
- Las experiencias motrices, sirven de base para la posible creación de un instrumento cognitivo [41].
- Existe una interrelación entre la actividad motora y un mayor desarrollo intelectual [42].

Conceptos clave aplicados a la programación tangible para niños, extraídos de los diferentes estudios:

- **Abstracción:** La capacidad de abstracción consiste en encontrar el nivel adecuado de detalle para definir y resolver un problema. [18].
- **Automatización:** Proceso por el cual se produce un ahorro de trabajo que se instruye para efectuar un conjunto de tareas repetitivas mediante una computadora. Es mucho más eficiente en comparación con el poder de procesamiento de un ser humano, por lo que las ejecuciones automatizadas de procesos realizados por máquinas simplifican el trabajo [19].
- **Descomposición y análisis de problemas:** La descomposición puede tomar la forma de desmontar un problema a lo que se cree que es su esencia fundamental [20]. Romper los problemas en partes más pequeñas para poder resolverlos y analizarlos de una manera más fácil, ayuda a la simplificación y resolución de problemas más complejos y de mayor escala.
- **Creatividad:** La creatividad es a la vez una capacidad integral de los seres humanos y una parte importante del conjunto de actividades del pensamiento computacional [21]. Se intenta desarrollar la habilidad creativa de los niños estimulando su curiosidad, especialmente la imaginación.

Identificar áreas de mejora en sistemas tangibles.

Identificar y seleccionar las áreas de mejora en dispositivos tangibles, a partir de la información recopilada. Esta información es de gran importancia para poder elaborar una lista de requisitos iniciales para la plataforma de juego. Las áreas de mejora localizadas son aplicadas en el diseño de una segunda propuesta de interfaz tangible.

Los diferentes sistemas de interacción tangible tienen como propósito general, que la información sea comprensible y literalmente captable, haciendo uso de objetos físicos que sean

manipulables de una forma natural.

La evolución de la tecnología aplicada a los modelos actuales de sistemas tangibles, van adaptándose para mejorar los diseños y conseguir de esta forma sistemas más fiables. No obstante, existen diferentes áreas de mejora localizadas sobre los diferentes sistemas de interacción tangible:

- **Uso de elementos externos para su funcionamiento.** En determinados juegos, es imprescindible el uso de un ordenador para su normal funcionamiento (como es el caso de *Scratch & WeDo*). El uso de periféricos o el uso de una cámara para el reconocimiento de imágenes.
- **Elevado número de elementos tangibles para el desarrollo del juego.** Si bien es cierto que uno de los puntos más importantes a la hora de la programación tangible para niños, es poder representar partes de código de diferentes elementos tangibles, un gran número de elementos puede suponer un problema para el niño a la hora del desarrollo del juego, pudiendo en algunos casos, confundirlo.
- **Enfoque exclusivo de un rango de edades para el juego.** En la mayoría de los casos, los juegos están enfocados a niños de un determinado rango de edades, limitando de esta manera el juego, al resultar de una dificultad elevada para niños de corta edad, o demasiado fáciles para el resto de las edades.
- **Diseño estático del juego.** Una de las limitaciones de todos los juegos que hacen uso de interfaces tangibles, es que no pueden ser actualizados, o no disponen de actualizaciones de software, por tanto, el juego no evoluciona, e imposibilita posibles futuras mejoras en los juegos.
- **Alto coste de los bloques o dispositivos de realimentación.** El uso de un gran número de elementos para la interacción tangible eleva el precio del diseño.

Segunda propuesta de plataforma de juego.

Propuesta de diseño, de un dispositivo o plataforma de juego, en base al modelo básico de interfaces tangibles, que cumpla los requisitos identificados en las distintas áreas de mejora localizadas en esta iteración.
Identificar cual es la tecnología más adecuada a utilizar en el dispositivo una vez analizada la evolución tecnológica aplicada a los modelos actuales.

Este sistema de programación tangible consta de dos dispositivos. Uno principal y otro de control. El objetivo del juego es realizar programas secuenciales, los cuales son representados en la pantalla del dispositivo principal. Estos programas pueden ser creados o modificados, usando el dispositivo de control, que actúa como elemento tangible.

La plataforma de juego se basa en los conceptos de secuenciación, repetición de bucles, y programación creativa.

- **Dispositivo principal**: indica todas las operaciones a realizar, ya sea movimientos, secuencias, eventos acústicos, etc. Dispone de una pantalla táctil donde se visualiza y establece el desarrollo del juego. La estructura alberga el sistema de control, comunicación y procesamiento de datos (*Raspberry Pi*). Incorpora sensores de proximidad situados en las cuatro caras que rodean el dispositivo. Dispone de un magnetómetro, cuya finalidad es detectar el dispositivo auxiliar en una posición concreta (encima de la pantalla o en las proximidades del dispositivo principal). La Figura 6.4 muestra el diseño del dispositivo principal.

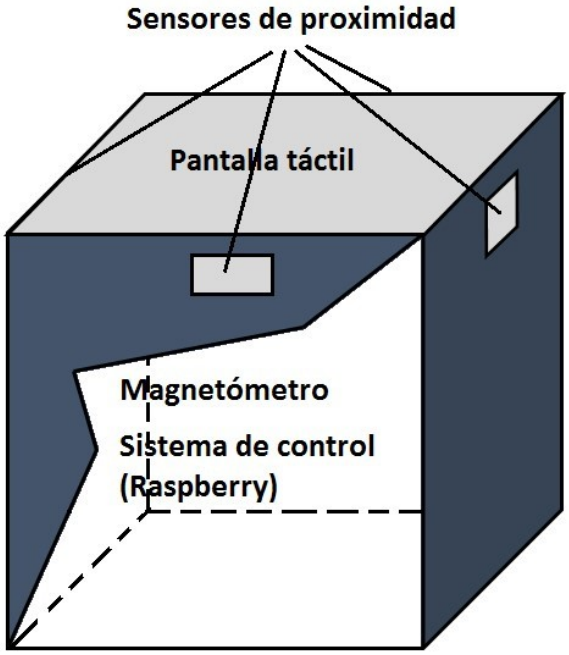

Figura 6.4: Diseño del dispositivo principal en la segunda propuesta de plataforma de juego.

- *Dispositivo auxiliar*: es diseñado con el propósito de manipular el programa, para la adicción, reorganización, y eliminación de partes de la secuencia (estructuras de control de flujo, acciones y parámetros). Estas órdenes son representadas en la pantalla táctil del propio dispositivo. La interacción ocurre cuando se sitúa el elemento de control sobre la pantalla del dispositivo principal. Incorpora un giroscopio que posibilita el control de movimiento de la secuencia en determinadas etapas del juego. La Figura 6.5 muestra el diseño del dispositivo de control.

Ejemplos de aplicaciones dentro de la plataforma de juego.
Son implementados los diferentes sensores en la plataforma de juego. Se emplean colores para simplificar la explicación.

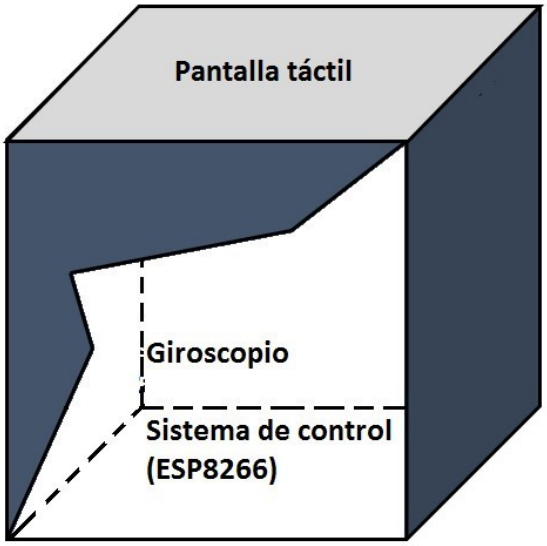

Figura 6.5: Diseño del dispositivo de control en la segunda propuesta de plataforma de juego.

Definir una secuencia. La pantalla del cubo principal se divide en nueve secciones para que la interacción sea más intuitiva. Se define el recorrido que tendrá la secuencia del juego, deslizando el dedo sobre la superficie de la pantalla (Figura 6.6).

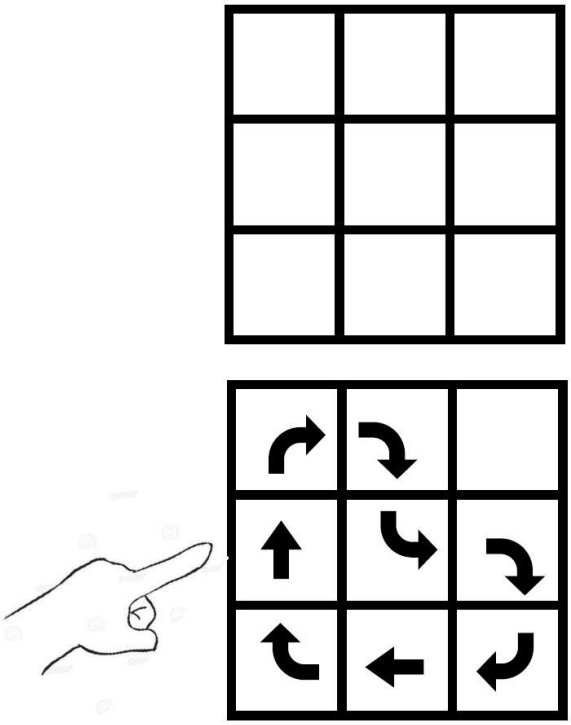

Figura 6.6: Definir una secuencia en el dispositivo principal.

El programa comienza a ejecutarse en función del recorrido marcado, desplazándose de un cuadro a otro (Figura 6.7).

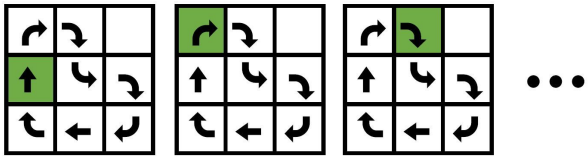

Figura 6.7: Desarrollo de una secuencia en el dispositivo principal.

Uso de sensores de proximidad. Aproximando la mano en los laterales del dispositivo principal, el cuadro es desplazado, siendo la secuencia alterada (Figura 6.8).

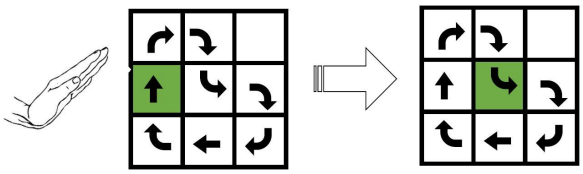

Figura 6.8: Modificación de la secuencia usando sensores de proximidad.

Interacción directa entre ambos dispositivos. Intercambio de información que permite modificar la secuencia realizada directamente con el dispositivo de control. Como ejemplo, se realiza una *captura* dentro de la secuencia. La idea es posicionar el dispositivo de control sobre el dispositivo principal, para obtener la información del cuadro activo en ese momento en el juego, eliminándolo de la secuencia (Figura 6.9).

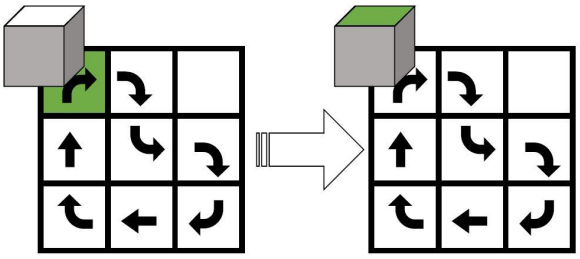

Figura 6.9: *Captura* del cuadro activo dentro de la secuencia usando el dispositivo de control.

Una vez *capturado*, se puede modificar los atributos (por ejemplo, el color), tocando sobre la pantalla (Figura 6.10), para después emplazarlo en otra localización de la secuencia (Figura 6.11).

Análisis de la segunda propuesta de diseño

La segunda propuesta de diseño es analizada junto con el director del proyecto. Se decide aceptar la propuesta al cubrir gran parte de las áreas de mejora localizadas en los actuales sistemas en la programación tangible. No obstante, se realizan algunas modificaciones, con el objetivo de eliminar posibles futuros problemas dentro del diseño e implementación.

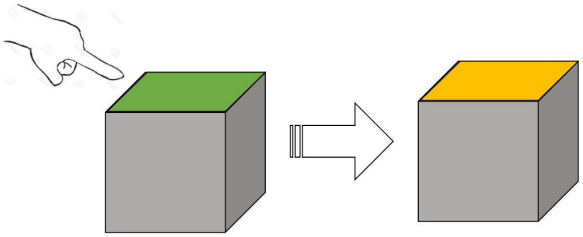

Figura 6.10: Modificación de la secuencia usando del dispositivo de control.

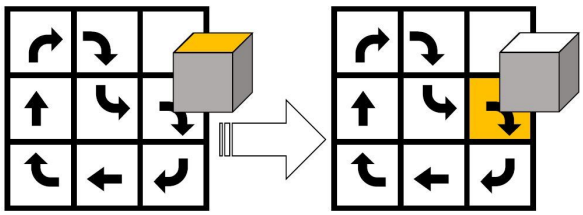

Figura 6.11: Emplazamiento dentro de la secuencia del cuadro modificado por medio del dispositivo de control.

Posibles problemas y mejoras de la plataforma de juego.

- Localización del dispositivo de control, sobre el dispositivo principal, usando un magnetómetro. Debido a que el sensor no ofrece la precisión necesaria que requiere la aplicación, para indicar la posición exacta de un dispositivo respecto a otro, se decide eliminar este sistema de localización.
- Definir un tamaño para ambos dispositivos que permita un correcto desarrollo de la interacción del juego.
- Concretar la tecnología necesaria para implementar la aplicación en los dispositivos para un correcto funcionamiento.
- Incorporar una unidad de medición inercial y sensores ópticos, que mejoren la experiencia de juego.

Próximas historias de usuario.

- Establecer requisitos iniciales.
- Identificar y redactar objetivos.

6.3.4 Iteración 4: Requisitos iniciales. Objetivos del TFG.

Establecer unos requisitos iniciales para empezar a desarrollar una plataforma de juego basada en interfaces tangibles, una vez se han analizado las posibilidades y limitaciones de las diferentes tecnologías actuales.

Los requisitos iniciales establecen las funcionalidades del dispositivo, y sirven de guía para el desarrollo del resto de la aplicación.
Los objetivos del proyecto son fijados a partir de la tecnología y requisitos iniciales obtenidos.
Los objetivos específicos podrán ser modificados durante el transcurso del resto de iteraciones, acorde con las necesidades del dispositivo.

Requisitos iniciales del dispositivo.

Eliminar el uso de elementos externos al dispositivo.
El uso de elementos externos para el funcionamiento del juego, como por ejemplo el uso de un teclado o una pantalla que el propio usuario debe disponer, en ocasiones puede acarrear un problema a la hora de utilizar el juego. Para eliminar el uso de elementos externos para el funcionamiento del sistema de juego, se diseña un prototipo, el cual prescinda de dichos elementos, integrando todo lo necesario dentro de la misma plataforma.

Reducir el elevado número de piezas para la interacción tangible.
Un sistema de interacción tangible se caracteriza principalmente en que la información sea comprensible y literalmente captable, haciendo uso de objetos físicos que sean manipulables de forma natural. En ocasiones, el número de objetos físicos para el desarrollo del juego es elevado. Si bien es cierto que uno de los puntos más importantes a la hora de la programación para niños, es poder representar partes de código en diferentes elementos tangibles, un gran número de elementos puede suponer un problema para el niño a la hora del desarrollo del juego, pudiendo en algunos casos confundirlo.
Los sistemas de programación tangible utilizan bloques que al conectarlos o posicionarlos de una manera concreta, generan en tiempo real movimientos o acciones, bien mostrando la acción realizada en una pantalla, o bien mediante acciones llevadas a cabo por algún tipo de robot articulado, etc... Cada bloque indica una acción, por ejemplo, bloque de inicio, bloque de fin, bloque de dirección, bloque sensor...Para dar solución a este problema, la idea pasa por agrupar los bloques tangibles, que son utilizados en la interacción del juego y que tienen como finalidad representar una acción, en un solo elemento tangible. Se propone el diseño de un dispositivo tangible que, por medio de un microprocesador, muestre en una pantalla estas funciones.

Ampliar el enfoque del rango de edades en el juego. Diseño dinámico del juego.
La mayoría de los diseños actuales de dispositivos dedicados a la programación tangible, están enfocados a un rango de edades determinado. Un juego puede resultar demasiado simple o demasiado complejo para algunos niños, debido a la edad de estos. Otro punto que tener en consideración, es el diseño estático en los juegos. El usuario dispone de unos recursos para ser utilizados, y aunque las combinaciones y posiciones de elementos tangibles permiten

realizar diferentes acciones de juego, no dejan de ser siempre los mismos elementos con los mismos propósitos. Estos dos puntos están relacionados con una característica común, no tienen la posibilidad de mejorar, no pueden ser actualizados, al menos que se adquiera una nueva versión del producto, lo que perjudica al usuario final.

Reducir costes de los bloques o dispositivos de realimentación.
El alto coste de los bloques o dispositivos de realimentación, al igual que el uso de un gran número de elementos empleados en la interacción tangible, eleva el precio del diseño.

Requisitos iniciales del dispositivo, a partir de las áreas de mejoras localizadas en los diferentes aplicaciones dedicadas a la programación tangible para niños:

- Dispositivo completo e independiente de elementos externos al mismo.
- Agrupar elementos tangibles en un solo dispositivo tangible.
- Disponer de actualizaciones software, para mejoras la plataforma.
- Utilización de elementos de bajo coste para el diseño.

Requisitos de diseño del juego dentro de la plataforma:

- Fácil de entender.
- Apropiado para realizar el juego de forma libre.
- Bloques simples que representen el vocabulario/léxico.
- Posibilidad de realizar un gran número de programas.
- Mejorar/depurar sin necesidad de herramientas externas.

Objetivos del TFG

Objetivo general: Diseñar y desarrollar una plataforma interactiva de juego de bajo coste, basada en interfaces tangibles, haciendo uso de las nuevas tecnologías, para promover el desarrollo de capacidades en niños de corta edad.

Objetivos específicos:

- Diseñar y programar un sistema de localización y reconocimiento de las interfaces de usuario tangibles.
- Desarrollar un sistema eficiente de transferencia de datos entre ambas interfaces tangibles.
- Realizar una correcta sincronización en la visualización de la aplicación entre ambas pantallas, que permita una buena representación de juego.
- Ofrecer un entorno gráfico simple, que permita al niño ser capaz de diseñar secuencias y modificarlas, por medio de elementos tangibles.

- Adaptar sensores inerciales y ópticos para una mejor experiencia de juego, donde ambos elementos tangibles interactúen entre sí, permitiendo la manipulación de la información entre los dispositivos.
- Diseñar un sistema de actualización de software de los dispositivos, a través de la conexión externa a un servidor.

Próximas historias de usuario.

- Tecnologías para el desarrollo.
- Diseño de la plataforma de juego, a partir de los requisitos y objetivos fijados.

6.3.5 Iteración 5: Diseño de la plataforma de juego. Tecnologías para el desarrollo.

Descripción de la plataforma de juego propuesta en el proyecto.

Este diseño pretende abordar los problemas señalados en la iteración 4 6.3.4 de este documento, introduciendo *TUIOs*[1] accionados que son controlables por el sistema, con el fin de representar cambios dinámicos de la información, combinando elementos tangibles, gráficos y nuevas tecnologías. La plataforma de juego se basa en los conceptos de secuenciación, repetición de bucles, y programación creativa.

La idea principal parte de que los niños no sean simplemente consumidores de contenido y que tengan la posibilidad de crearlo, que ellos mismos sean capaces de tomar decisiones durante el juego de una manera tangible e intuitiva. El sistema de juego abarca desde el desarrollo de una secuencia a partir de los criterios de creación del niño, hasta la resolución de pequeños problemas de lógica.

La plataforma ofrece un entorno de juego compartido entre dos dispositivos tangibles principales, *TUIO1* y *TUIO2*. Ambos dispositivos disponen de interfaz gráfica con panel táctil, para la visualización y desarrollo del juego. Los eventos durante el juego son transmitidos vía inalámbrica. La plataforma incorpora sensores para mejorar la experiencia de juego.

A continuación, son detallados los componentes y características de cada dispositivo.

Dispositivo TUIO1.

Este dispositivo integra un ordenador de placa reducida *Raspberry Pi*, que maneja todos los eventos generados durante el juego. Se encarga de la representación gráfica principal del juego por medio de una pantalla de 7 pulgadas táctil capacitiva (ver Figura 6.12). Otra de las características es que dispone de una salida de audio para eventos tipo acústicos.

Placa para el desarrollo en TUIO1. Raspberry Pi.

Esta placa dispone de un microprocesador *BCM2837* de 64 bits de cuádruple núcleo a 1,2GHz, 1GB de memoria *RAM* y *GPU* de doble núcleo Videocore IV, que permite ma-

[1] Tangible User Interface Objects

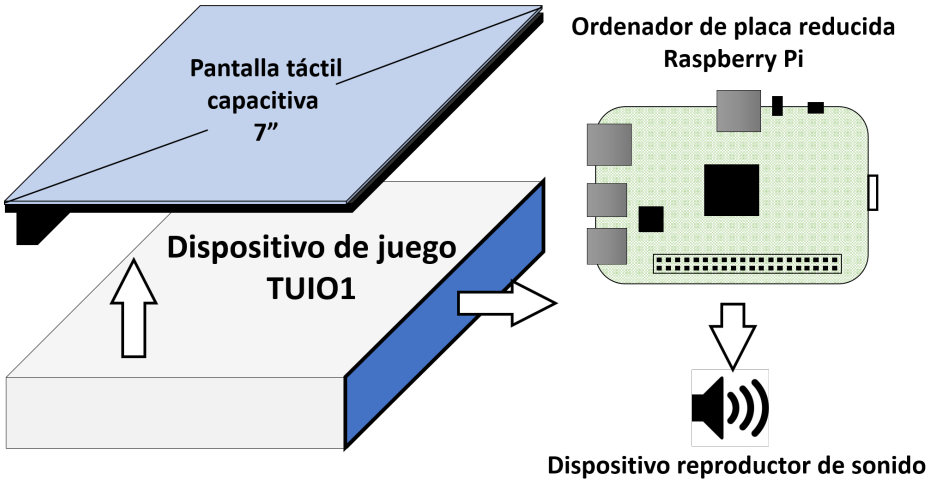

Figura 6.12: Dispositivo de juego TUIO1.

nejar y procesar eventos e imágenes según los requisitos de la aplicación. Integra el chip *BCM43143* para la comunicación *WiFi* entre dispositivos, lo que elimina la necesidad de utilizar cables. Interfaz gráfica de usuario por puerto HDMI. Incorpora 40 pines *GPIO*, los cuales disponen de un bus *I2C* [2] para la utilización de sensores dentro del sistema de juego, y un bus *SPI*[3] como alternativa al puerto HDMI para mostrar la interfaz gráfica de usuario. Su reducido tamaño, bajo consumo, además de contar con una gran comunidad de usuarios, hacen de este dispositivo el adecuado para la realización de un prototipo de plataforma de juego.

La documentación relacionada con este dispositivo, se encuentra detallada en: *Sección 5.1.1 Raspberry Pi 3B.*

Toda la documentación relacionada con el dispositivo *Raspberry Pi*, se encuentra detallada en el Capítulo 5 Tecnologías para el desarrollo. Hardware y Software, en la sección 5.1.1 Raspberry Pi 3.

Elementos para la representación intangible de la información en TUIO1.
Si bien los elementos tangibles juegan el papel central en la representación y control en una interfaz de usuario tangible, la representación intangible es también importante, siendo esta información dinámica proporcionada, una gran ayuda al realizar una interacción. La existencia en tiempo real de una realimentación de la representación intangible, cuando se ha manipulado la representación tangible, es fundamental para asegurar un correcto acoplamiento perceptivo. Las representaciones intangibles que incorpora el diseño son auditivas y gráficas. Estas representaciones se producen en determinados momentos en desarrollo del juego.

[2] I2C, «Inter-Integrated Circuit».
[3] SPI, «Serial Peripheral Interface»

Interfaz gráfica en *TUIO1*.

Pantalla táctil capacitiva de 7 pulgadas para la representación gráfica del juego. Una pantalla capacitiva detecta varias pulsaciones simultaneas sobre ella. Esta característica es fundamental para desarrollar un sistema de localización del dispositivo *TUIO2* sobre la pantalla. La conexión de la interfaz gráfica con la placa para el desarrollo es HDMI, por su compatibilidad y facilidad de uso con *Raspberry Pi*. Los eventos generados en el panel táctil son transmitidos desde el puerto *Micro USB-B*, al puerto de comunicaciones *USB-A* de la placa. La Figura 6.13 muestra las conexiones de ambas interfaces. Se considera que las dimensiones de la pantalla son las adecuadas para una correcta visualización de los contenidos del juego. Queda abierta la posibilidad del empleo de una pantalla de mayores dimensiones.

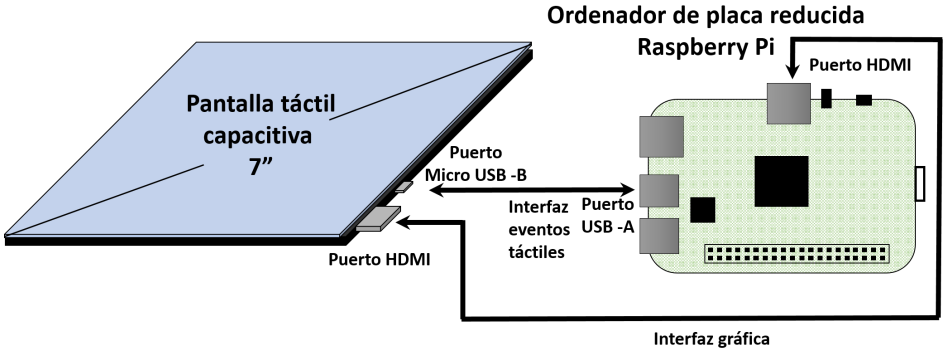

Figura 6.13: Conexión de la interfaz gráfica de usuario y interfaz de eventos táctiles, entre pantalla y Raspberry Pi.

La documentación relacionada con este dispositivo, se encuentra detallada en: *Sección 5.1.2 Pantalla táctil capacitiva HDMI.*

Dispositivo *TUIO2*.

Este dispositivo está diseñado con la finalidad de agrupar las acciones, movimientos, apariencia, eventos acústicos, parámetros, etc., que definen un lenguaje de programación tangible. Esta agrupación elimina el problema de disponer de distintos elementos tangibles para el desarrollo del juego. *TUIO2* hace de puente entre la interfaz física y la digital. La idea es mostrar de forma gráfica en una pantalla de 3,5 pulgadas, todos estos elementos de programación para crear una secuencia, y ser aplicados al juego por medio una interacción física (sistema de localización), o transmitir estas instrucciones de forma inalámbrica. El sistema de localización aplicado en la interacción física es posible gracias a unas almohadillas conductivas situadas en la parte posterior del dispositivo, y que son detectadas por la pantalla capacitiva de *TUIO1*. *TUIO2* dispone de dos sensores. Un sensor de color, y una unidad de medición inercial. Estos sensores son utilizados para mejor la experiencia de juego. La Figura 6.14 muestra una representación general del dispositivo *TUIO2*.

Placa para el desarrollo en *TUIO2*.

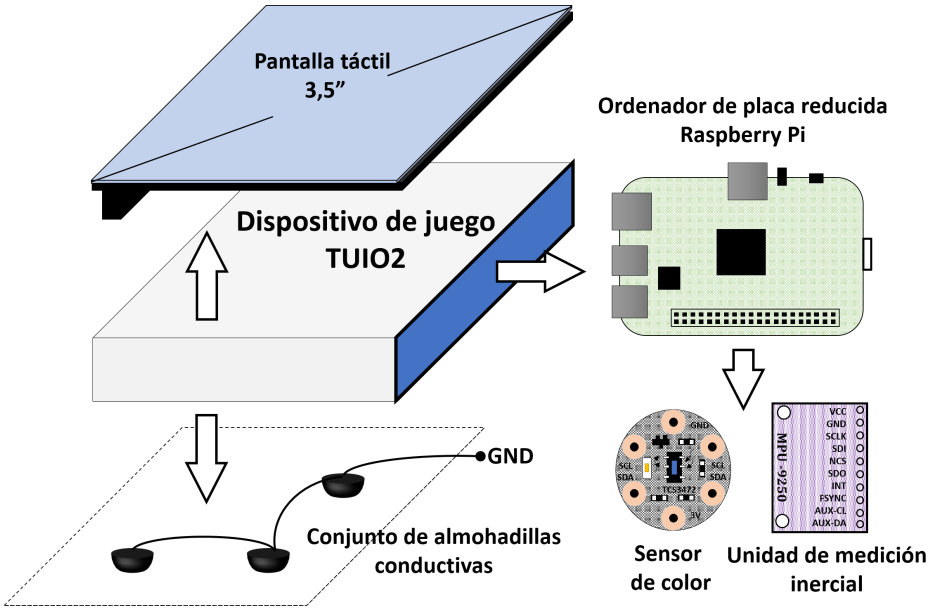

Figura 6.14: Dispositivo de juego TUIO2.

Al igual que *TUIO1*, este dispositivo integra un ordenador de placa reducida *Raspberry Pi*, que maneja todos los eventos generados durante el juego.

Sistema de localización de TUIO2 sobre TUIO1.
Un panel táctil de tipo capacitivo detecta varios eventos sobre el mismo, por lo general un dedo un humano, pero también detecta la presencia de un conductor eléctrico conectado a tierra. Estas dos características ofrecen la posibilidad de detectar la posición del dispositivo *TUIO2* sobre *TUIO1*. Por ejemplo, si se desea ordenar al juego realizar un desplazamiento hacia la derecha dentro de la secuencia, el dispositivo *TUIO2* muestra de forma gráfica esta acción (símbolo hacia la derecha), y puede realizar esta instrucción simplemente presionando sobre el símbolo, o posicionar el dispositivo sobre la pantalla de *TUIO1*. El dispositivo *TUIO2* dispone de unas almohadillas conductivas (ver Figura 6.15) que son detectadas por la pantalla de *TUIO1*. Esto es posible debido a la capacidad de detectar varios eventos táctiles de una pantalla capacitiva. La pantalla de *TUIO1* refleja dentro de la secuencia la instrucción enviada.

Interfaz gráfica en *TUIO2*.
Pantalla táctil resistiva de 3.5 pulgadas para la representación gráfica del juego. La conexión de la interfaz gráfica con la placa para el desarrollo es *HDMI*, por su compatibilidad y facilidad de uso con Raspberry Pi. Los eventos generados en el panel táctil son transmitidos a la placa por comunicación *SPI*. La Figura 6.16 muestra las conexiones de ambas interfaces. Se considera que las dimensiones de la pantalla son las adecuadas para una correcta visualización de los contenidos del juego. Queda abierta la posibilidad del empleo de una pantalla de mayores dimensiones.

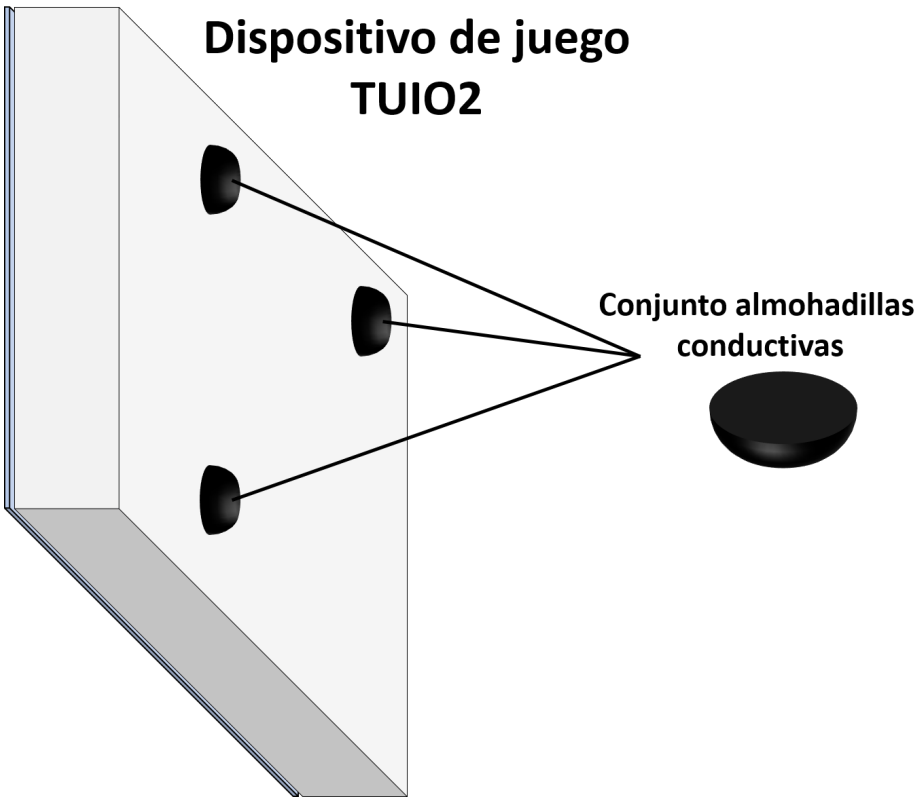

Figura 6.15: Disposición de las almohadillas conductivas en dispositivo de juego TUIO2.

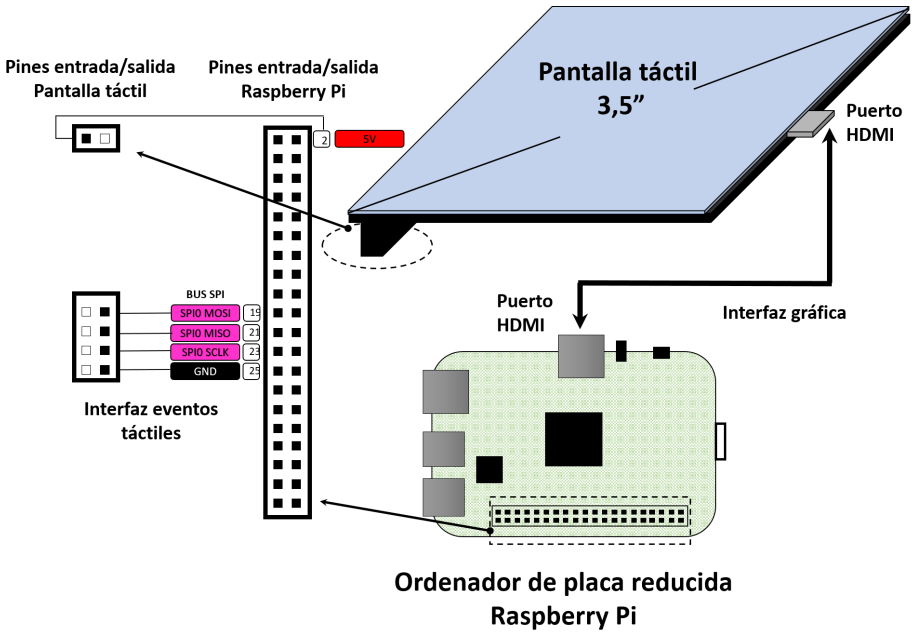

Figura 6.16: Conexión de la interfaz gráfica de usuario y interfaz de eventos táctiles, entre pantalla y Raspberry Pi en TUIO2.

La documentación relacionada con este dispositivo, se encuentra detallada en: *Sección 5.1.3 Pantalla táctil resistiva HDMI.*

Sensores implementados en el dispositivo TUIO2.

Uno de los objetivos específicos es adaptar sensores inerciales y ópticos para una mejor experiencia de juego, donde los elementos *TUIO1* y *TUIO2* interactúen entre sí, permitiendo la manipulación de la información entre los dispositivos.

Unidad de medición inercial.
La unidad de medición inercial integrada en la plataforma, es el sensor *MPU9255*. Este sensor forma parte del dispositivo *TUIO2*. Esta destinado principalmente a capturar eventos de movimiento, para generar acciones durante el transcurso del juego. Al realizar movimientos angulares sobre los ejes X,Y,Z del dispositivo *TUIO2*, se generan unos eventos que son filtrados y comunicados al dispositivo *TUIO1*, para ser ejecutados durante el transcurso del juego. Por ejemplo, girar una pieza representada en *TUIO1* (Ver Figura 6.17).

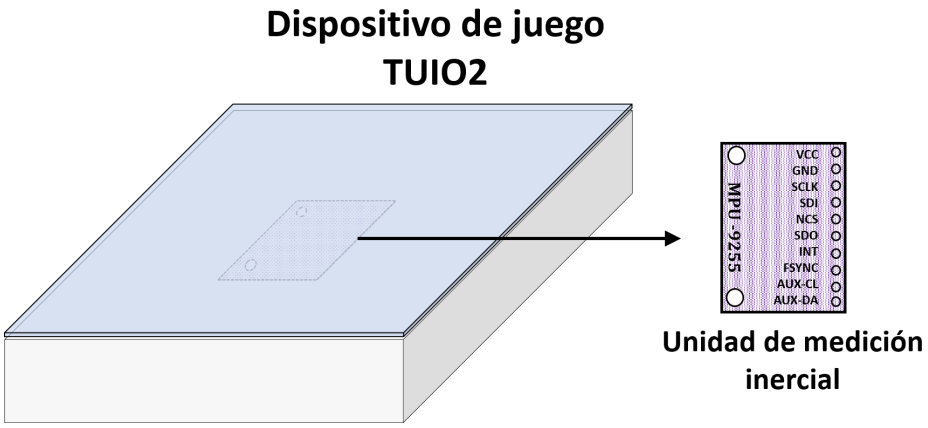

Figura 6.17: Ejemplo de aplicación de desplazamientos angulares mediante la unidad de medición inercial.

Sensor de color.
Los sensores de color detectan el color de una superficie, y entregan la información de manera digital. El dispositivo implementado es el sensor de color *TCS34725*, que proporciona un retorno digital de los valores de detección de luz roja, verde y azul (RGB). Dispone de un filtro de bloqueo *IR*[4] para una mejor detección de la luz ambiente. La integración de este dispositivo (ver Figura 6.18), hace posible una interacción directa con el medio que le rodea, lo que permite ampliar las opciones de juego dentro de la plataforma. Por ejemplo, localizar objetos de color rojo. La información de color obtenida sirve para cambiar parámetros, o generar eventos durante el transcurso del juego (ver Figura 6.19).

Sistema operativo y lenguajes de programación. Herramientas software El sistema operativo utilizado en ambos dispositivos es *Raspbian*. *Raspbian* es un sistema operativo basado en la distribución *Debian*, recomendado para *Raspberry Pi* por estar optimizado para

[4]Infrared radiation

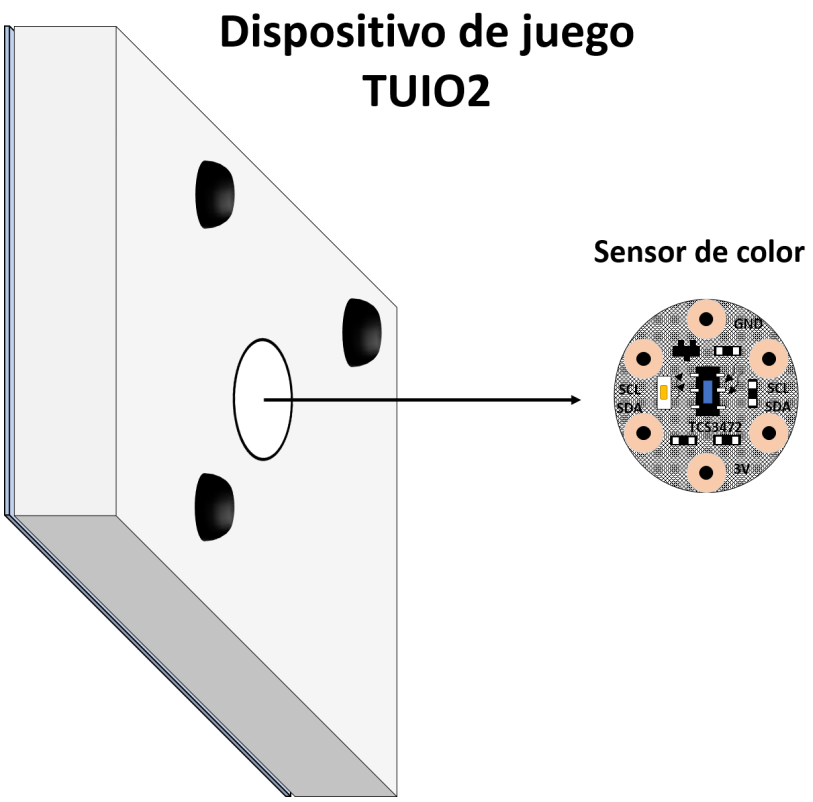

Figura 6.18: Sensor de color incorporado en el dispositivo TUIO2.

su *hardware*, siendo el más estable y con mayor rendimiento. En esta distribución del sistema operativo, son instalados los paquetes necesarios para el intérprete *Python* y *Kivy*, que serán las herramientas utilizadas para realizar la programación de toda la plataforma de juego.

La documentación relacionada con la instalación de *Raspbian*, se encuentra detallada en: *Sección 5.2.1 Sistema Operativo GNU/Linux. Raspbian.*

Próximas historias de usuario.

- Estructura de la aplicación.
- Programa de pruebas comunicaciones entre dispositivos.

6.3.6 Iteración 6: Estructura de la aplicación. Programa de pruebas comunicaciones entre dispositivos.

Organización y estructura de la aplicación.

Antes de proceder a la programación de las aplicaciones, es necesario establecer una estructura para el proyecto, almacenando los archivos en distintos directorios, dependiendo de la función que realice cada uno de ellos (ver Figura 6.20).

El directorio principal de la aplicación es nombrado `itanium`. En este directorio se encuentra el archivo `Tuio1.py`, que es el módulo principal de la aplicación. Contiene el código en

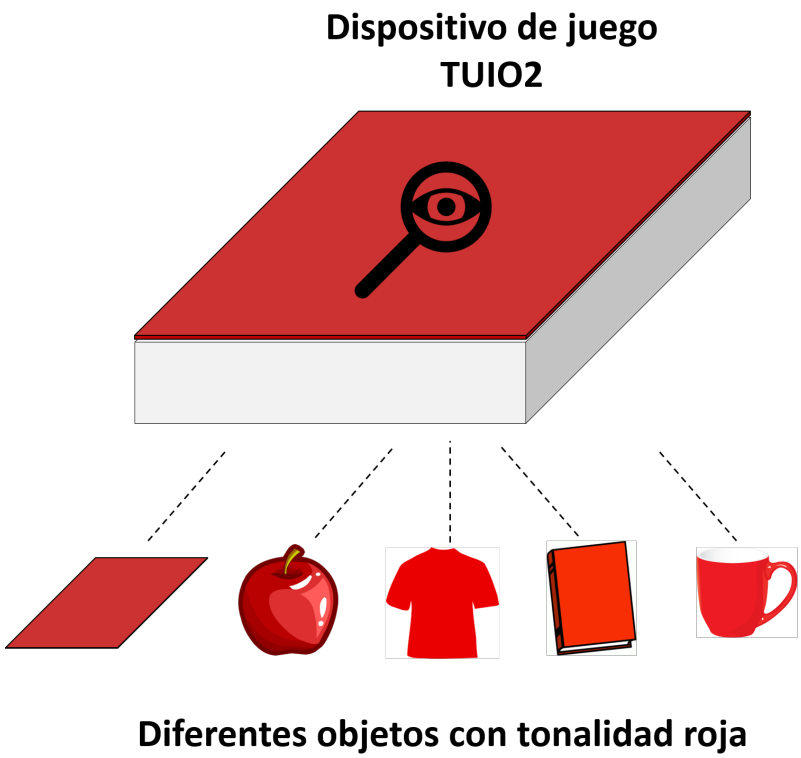

Figura 6.19: Ejemplo de aplicación para detectar objetos color rojo mediante el sensor de color.

lenguaje *Python*, donde se ejecutan los diferentes módulos, gestión de eventos, ejecución de la interfaz gráfica, etc. Dentro del directorio `itanium`, existen tres carpetas donde se organizan los diferentes archivos que dispone la aplicación:

- Carpeta `lib`, dedicada a las diferentes librerías creadas para el proyecto, como son las comunicaciones de servidor y sensores.
- Carpeta `screens`. Incluye los archivos de extensión *kv*. Estos archivos contienen el código (escritos en lenguaje *Kivy*), de los distintos elementos y objetos que componen la interfaz gráfica de la aplicación (botones, figuras, fondos, etc..).
- Carpeta `data`. En este directorio se encuentran las distintas imágenes utilizadas para la interfaz gráfica.

Librerías para servidor y cliente.

Las librerías utilizadas para crear las comunicaciones del lado servidor y cliente son:

- socket
- Theread
- struct
- Queue

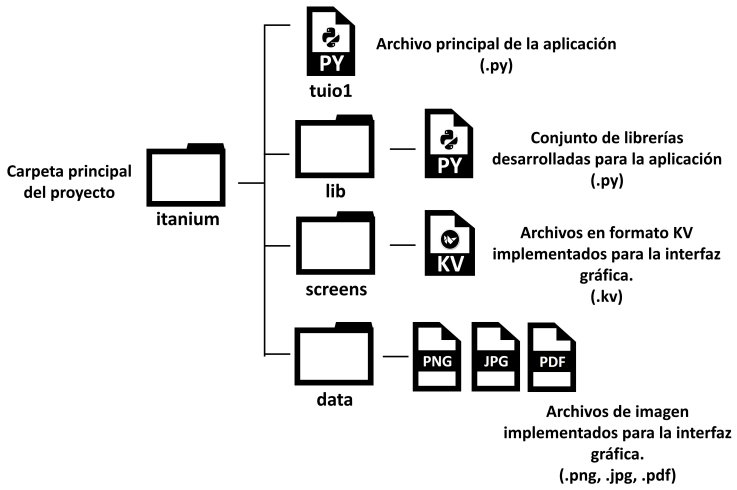

Figura 6.20: Estructura de carpetas para la aplicación.

Las comunicaciones entre los dispositivos *TUIO1* y *TUIO2*, se realizan de manera inalámbrica (*WiFi*), utilizando la librería `sockect`, con protocolo de comunicaciones *TCP/IP*. El dispositivo *TUIO1* ejecuta el lado servidor, y *TUIO2* el lado cliente. Se establece la siguiente jerarquía: **el cliente solo enviará información a petición del servidor.** Los mensajes de entrada/salida son almacenados en *colas* (librería `Queue`), de una manera estructurada (librería `struct`, para generar eventos dentro del programa. Tanto el cliente como el servidor administran los eventos mediante *hilos* (librería `Thread`), que se ejecutan de forma paralela al resto del programa.

Estructura básica servidor.

La secuencia de operaciones básicas para el servidor son las siguientes:

- Crear un canal de comunicaciones. La llamada *socket* asigna un descriptor de archivo a un canal de comunicación. Es necesario proporcionar datos sobre el dominio y tipo del conector. En este caso se emplea el protocolo *IPv4* y un conector tipo *STREAM*, que es el tipo de flujo que utiliza el protocolo *TCP*, el cual asegura que los mensajes enviados al cliente llegan en el mismo orden en el cual fueron enviados. `sock = socket(AF_INET, SOCK_STREAM)` El objeto `sock` creado, implementa toda la *API socket*.
- Configurar direcciones origen/destino. El *host* hace referencia a la dirección *IP* del servidor, y *port* al puerto de escucha para conexiones de entrada. `sock.bind((host, port))`
- Configurar el número de conexiones simultáneas, método `listen()`. Solo es necesario una conexión simultánea. `sock.listen(1)`
- Crear un *socket* esclavo para atender la conexión. `client_connection, client_address = sock.accept()` Se instancia el objeto `cliente_connection`.

- Recibir datos del cliente. `client_connection.recv(1024)`
- Enviar datos al cliente. (En este ejemplo es necesario introducir un mensaje por medio del teclado) `client_connection.send(input().encode('utf-8'))`
- Cerrar el canal de comunicación. `client_connection.close()`

Estructura básica cliente.

La secuencia de operaciones básicas para el cliente son las siguientes:

- Crear un canal de comunicaciones. Al igual que el lado servidor, la llamada *socket* asigna un descriptor de archivo a un canal de comunicación. `sock = socket(AF_INET, SOCK_STREAM)`
- Establecer dirección de destino con el método `conect()`, indicando los parámetros de dirección IP del servidor (`host`), y puerto de escucha (`port`. `sock.conect(host,port)`
- Recibir datos del servidor. `client_connection.recv(1024)`
- Enviar datos al servidor. (En este ejemplo es necesario introducir un mensaje por medio del teclado) `client_connection.send(input().encode('utf-8'))`
- Cerrar el canal de comunicación. `client_connection.close()`

Resultados programa de pruebas comunicaciones.

Los resultados obtenidos referentes a las comunicaciones del lado cliente y servidor, al igual que el manejo de eventos en ambos dispositivos, son reflejados en: *Capítulo 7 Resultados, en el apartado* **??** *Comunicaciones cliente-servidor*.

Próximas historias de usuario.

- Inicio del sistema de localización entre dispositivos.

6.3.7 Iteración 7: Inicio del sistema de localización entre dispositivos.

6.3.8 Iteración 8: Lectura de datos y diseño de máquina de estados de la unidad de medición inercial *MPU9250*.

6.3.9 Iteración 9: Diseño software de las interfaces gráficas de usuario

6.3.10 Iteración 10: Diseño software de juego en la plataforma.

6.3.11 Iteración 11 Sistema de actualización software.

Capítulo 7

Resultados

A lo largo de este capítulo, se desarrolla los resultados obtenidos según la evolución en cada uno de los aspectos de la plataforma de juego.

- Estructura de la aplicación.
- Comunicaciones cliente-servidor.
- Sistema de localización.
- Lectura, tratamiento y comunicación de datos, de los sensores.
- Gestión de eventos de la aplicación. Máquina de estados.
- Diseño software de la interfaz gráfica.
- Actualización software de la aplicación.

7.1 Estructura de la aplicación. Gestión de eventos.

El funcionamiento de los dispositivos *TUIO1* y *TUIO2* es gestionado con el uso de *Máquinas de Estado Finitos (FSM)*[1]. La estructura base para el manejo de eventos, y las máquinas de estado finitos, es la misma para ambos dispositivos. Por simplificar la descripción y lectura de este documento, se detallan de manera conjunta ambos los módulos que forman la aplicación.

La estructura para la gestión de eventos sigue un proceso secuencial de adquisición y manejo de datos, dentro de los módulos de la aplicación. La Figura 7.1 muestra el manejo de eventos de salida (`output_events`) desde los módulos de la aplicación, y que son manejados por el módulo `Tuio1.py/Tuio2.py`. Los módulos de comunicaciones (`Server/Client`) no disponen de eventos de entrada propios. Esto es debido, a que el acceso a los métodos de ambos módulos, se hace de manera directa, ejecutando el método `send_data` para el envío de datos entre dispositivos. Los datos recibidos cliente/servidor, son añadidos a sus respectivas colas de eventos.

[1] Finite State Machine.

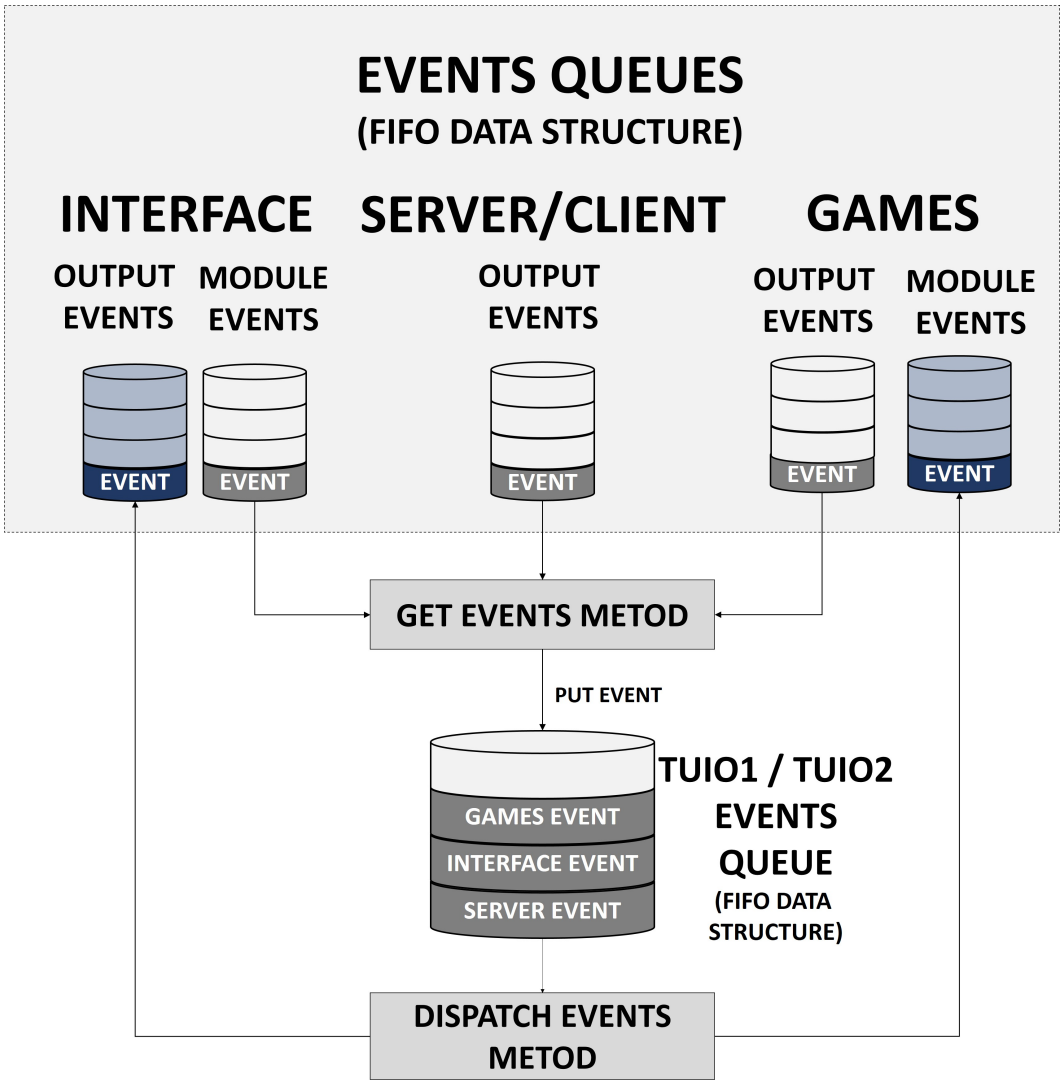

Figura 7.1: Manejo de eventos en *TUIO1/TUIO2*.

7.1.1 Máquina de estados finitos dispositivo *TUIO1/TUIO2*.

IDLE:

Estado inicial del dispositivo *TUIO1/TUIO2*. La máquina permanece en este estado hasta recibir confirmación de comunicación con el dispositivo *TUIO2/TUIO1*.

MAIN:

Estado principal de la aplicación una vez establecida la comunicación entre los dispositivos *TUIO1* y *TUIO2*.

GAME:

El dispositivo comienza a ejecutar el juego, al recibir un evento de comenzar juego, desde servidor o cliente.

EXIT:

Estado de la máquina por el cual se finaliza toda la aplicación.

En la Figura 7.2 se muestra la estructura de la máquina de estados, transiciones y métodos de los dispositivos *TUIO1/TUIO2*.

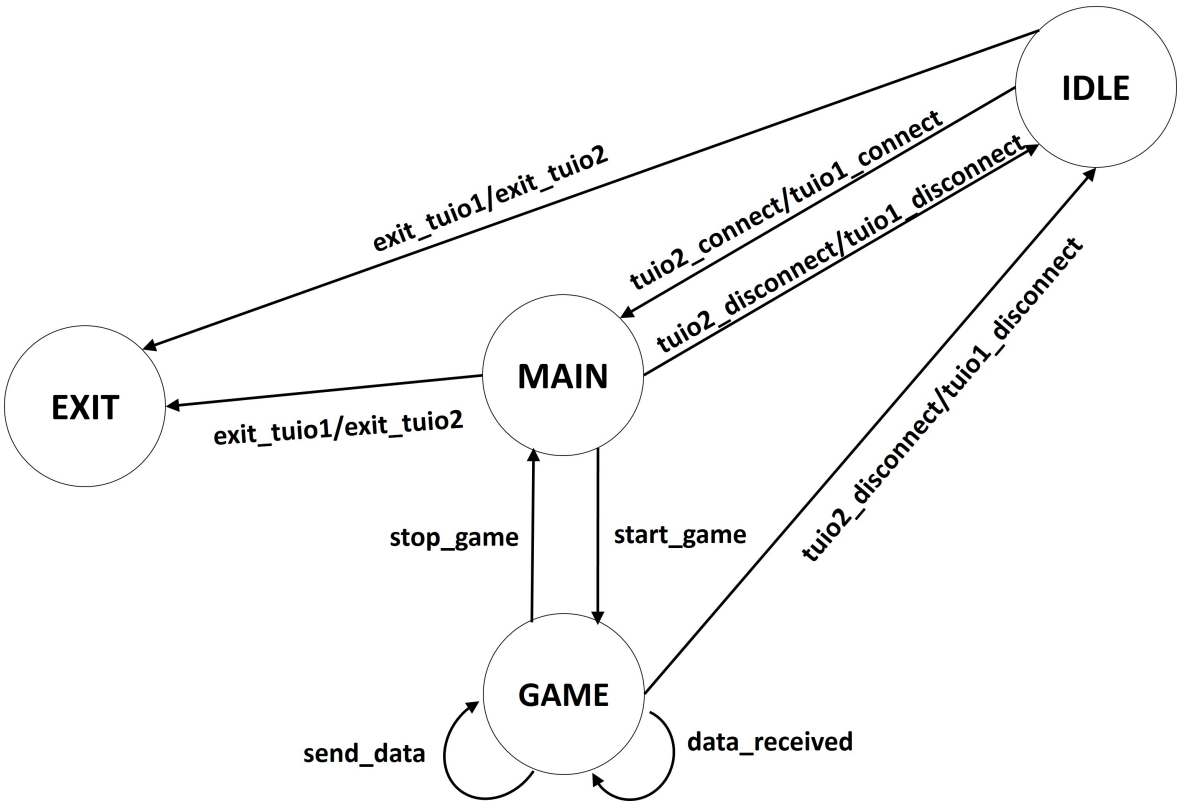

Figura 7.2: Estructura máquina de estados finitos `Tuio1FSM/Tuio2FSM`.

Módulos de control `Tuio1.py/Tuio2.py`

Estos módulos adquieren y distribuyen los eventos generados en las comunicaciones, interfaz gráfica y juegos, a la parte de la aplicación que corresponda siguiendo la estructura de eventos explicada en el punto anterior. Los eventos tienen una estructura de *tupla* de tres elementos. Dos de tipo *string*, que indican el tipo de evento y la acción a realizar dentro de la aplicación, y otra *tupla*, de cuatro valores tipo *float*, para los datos obtenidos de los sensores.

Módulo `Enum`. Librería estándar de *Python*.

Método de importación: `from enum import Enum`.

Una enumeración es un conjunto de nombres simbólicos (miembros) vinculados a valores únicos y constantes. Dentro de una enumeración, los miembros se pueden comparar por identidad, y la enumeración misma se puede repetir.

Referencia oficial (24/05/2018): https://docs.python.org/3/library/enum.html

Módulo `Clock`. Librería *Kivy*.

Método de importación: `from kivy.clock import Clock`

El objeto `Clock` permite programar una llamada a función una vez o repetidamente, en intervalos especificados. Puede obtener el tiempo transcurrido entre la programación y la llamada de la devolución de llamada mediante el argumento *dt*.

Referencia oficial (24/05/2018): https://kivy.org/docs/api-kivy.clock.html

Módulo `Queue`. Paquete `multiprocessing` de la librería estándar de *Python*.

Método de importación: `from multiprocessing import Queue`

Módulo implementado en la programación con hilos, cuando la información debe intercambiarse de forma segura entre múltiples hilos. La clase `Queue` en este módulo, implementa todas las semánticas de bloqueo requeridas. Las primeras tareas agregadas son las primeras recuperadas (datos *FIFO*).

Referencia oficial (24/05/2018): https://docs.python.org/2/library/queue.html

Módulo `Thread`. Paquete `threading` de la librería estándar de *Python*.

Método de importación: `from threading import Thread`

Este módulo proporciona primitivas de bajo nivel para trabajar con varios subprocesos (también denominados procesos o tareas ligeros): múltiples hilos de control que comparten su espacio de datos global. Para la sincronización, se proporcionan bloqueos simples (también denominados *mutexes* o *semáforos binarios*).

Referencia oficial (24/05/2018): https://docs.python.org/2/library/threading.html

Módulos propios: `Server/Client, InterfaceManagement, Games`. Paquete `lib` de la aplicación.

Método de importación: `from lib import Server/Client, InterfaceManagement, Games`

El módulo `Server/Client` proporciona los métodos para el intercambio de datos entre servidor y cliente. El módulo `InterfaceManagement` implementa las funciones y métodos para el manejo de pantallas (`ScreenManager`) dentro de la aplicación, que ofrece la interfaz gráfica de usuario. `Games` está compuesto por las clases y métodos de clase, de juegos básicos de la plataforma, donde serán añadidos los eventos propios para la ejecución de cada uno de ellos.

7.1.2 Clases y métodos del módulo `Tuio1.py/Tuio2.py`.

Clase `Tuio1State()/Tuio2State()`.

En esta clase son enumerados los distintos estados de la máquina de estados finitos de la aplicación. Hereda del módulo `Enum`.

```
IDLE = 0 # Estado inicial de la maquina de estados. Sin comunicaciones.

MAIN = 1 # Estado principal de la aplicación. Comunicaciones establecidas.

GAME = 2 # Juego activo.

EXIT = 3 # Estado final de la aplicación.
```

Métodos de clase `Tuio1FSM()/Tuio2FSM()`. Transiciones de estado.

Método`__init__()`

Establece el estado inicial del objeto al ser instanciado.

Inicialización de los atributos de clase.

- `current`. Atributo que determina el estado de la máquina.
- `server/client`. Instancia objeto `Server/Client`, para las comunicaciones con el cliente/servidor.
- `interface`. Instancia objeto `InterfaceManagement`, que controla los eventos gráficos de la aplicación.
- `games`. Instancia al objeto `Games` para los eventos gráficos, y ejecuta los métodos de los juegos de la aplicación.
- `events`. Cola de eventos de la clase `Tuio1FSM/Tuio2FSM`, donde son añadidos los diferentes eventos de los módulos `Server/Client, InterfaceManagement` y `Games`.
- Llamada al método `init_plataform()`, que inicializa las comunicaciones e interfaz gráfica de la plataforma.
- Llamada al método `thread_get_event()`, para iniciar el hilo de manejo de eventos.

Método `start_thread_get_event()`.

Llamada al método `Thread`. Inicia el hilo que contiene el método de reloj cíclico, que obtiene los eventos de los diferentes módulos que componen la aplicación.

Método `thread_get_event()`.

Este método implementa la función reloj (`clock`), que ejecuta de forma cíclica, el método `get_event()`, el cual obtiene los eventos de la aplicación.

Método `get_event()`.

Estructura de control que obtiene los eventos de los módulos servidor/cliente, interfaz y juegos. Estos eventos son añadidos a la cola `events` del propio módulo, para ser gestionados por el método `dispatch_event()`.

Método `dispatch_event()`.

Argumento de entrada: `event`.

Estructura de control para distribuir los eventos a los diferentes métodos, según la transición de estado en la máquina.

Método `init_plataform()`.

Llamada a los métodos `thread_init_server()/client(), interface.init_interface()`, que inicializan los objetos servidor e interfaz.

Método `start_game()`.

Argumento de entrada: `game`.

Método que añade a la cola de eventos del objeto `game`, para iniciar el juego según el argumento de entrada.

Método`stop_game()`.

Argumento de entrada: `game`.

Método que añade a la cola de eventos del objeto `game`, para detener el juego según el argumento de entrada.

Métodos `exit_tuio1()/exit_tuio2()`.

Transición al estado máquina `EXIT`. Finaliza las comunicaciones, interfaz gráfica y juegos. Llamada a los métodos `close_server()/client()` del objeto `server/client`, y `Clock.unschedule` para terminar la ejecución cíclica de adquisición de eventos.

Métodos `tuio2_connect()/tuio1_connect()`.

Método que indica que se han establecido las comunicaciones con el cliente/servidor. Ejecuta un cambio de pantalla (screens) en la interfaz gráfica.

Transición del estado `IDLE` al estado `MAIN`.

Llamada al método `main_state()` del objeto `interface`.

Métodos `tuio2_disconnect()/tuio1_discconnect()`.

Método para indicar que se han establecido las comunicaciones con el cliente/servidor.

Transición del estado `IDLE` al estado `MAIN`.

Llamada al método `main_state()` del objeto `interface`, que ejecuta un cambio de pantalla en la interfaz gráfica.

7.1.3 Gestión de eventos. Pruebas de validación.

Prueba para las comunicaciones y manejo de eventos entre *TUIO1* y *TUIO2*.

Ejemplo de proceso de manejo de eventos para comenzar un juego, partiendo del estado **MAIN** en ambas máquinas de estados (ver Figura 7.3 y Figura 7.4. Pasos seguidos :

Desde la interfaz gráfica de *TUIO2*, se presiona el botón de iniciar juego **PAINT GAME**. Este evento es de carácter interno del propio módulo `InterfaceMagmanement` que maneja la interfaz. El evento pasa por el gestor de eventos de la interfaz gráfica, y es incorporado a la cola de eventos externos, para ser tratado desde el gestor de eventos principal `Tuio1FSM`. El gestor de eventos principal obtiene el evento, el cual es añadido a la cola de eventos dedicada a juegos, para comunicar los datos al cliente, donde es gestionado por el gestor de eventos de *TUIO2*. El cliente recibe el evento, y lo añade a la cola de eventos del cliente. El gestor principal de eventos `Tuio2FSM`, lo distribuye, añadiendo las ordenes al gestor de eventos de la interfaz gráfica (`InterfaceManagement`), y al juego correspondiente para que sea iniciado (`PaintGame`).

Validación del módulo y manejo de eventos `Tuio1.py` y `Tuio2.py`.

La prueba efectuada sobre el manejo de eventos de los módulos principales de la aplicación, han ofrecido resultados correctos en todas las transiciones de estado y ejecución de métodos. Los módulos `Tuio1FSM` y `Tuio2FSM` son validados para su utilización dentro de la aplicación. Queda abierta la posibilidad de añadir distintos métodos y transiciones, siempre y cuando no se vea afectada la estructura principal de manejador de eventos.

7.2 Comunicaciones Servidor / Cliente.

7.2.1 Máquina de estados finitos Servidor / Cliente.

IDLE: Estado inicial del servidor a la espera de recibir un evento desde la máquina de estados del dispositivo *TUIO1/TUIO2* para iniciar el servidor/cliente. **Transiciones/Métodos.**

- `init_server/init_client`: transición al estado **LISTEN**. La ejecución de esta transición sucede cuando la máquina de estados del servidor/cliente recibe el evento `init_server/init_client` dentro de la máquina de estados `ServerFSM/ClientFSM`.
- `close_server/close_client`: transición al estado **EXIT** al recibir el evento `close_server/close_client` desde la máquina de estados `Tuio1FSM/Tuio2FSM`, para cerrar el servidor/cliente.
- `create_server/create_client`. Entrada desde la máquina de estados `Tuio1FSM/Tuio2FSM`, al iniciar el dispositivo.
- `reconnect`(perdida de conexión). Entrada de evento de reconexión al interrumpirse la conexión con el cliente/servidor.

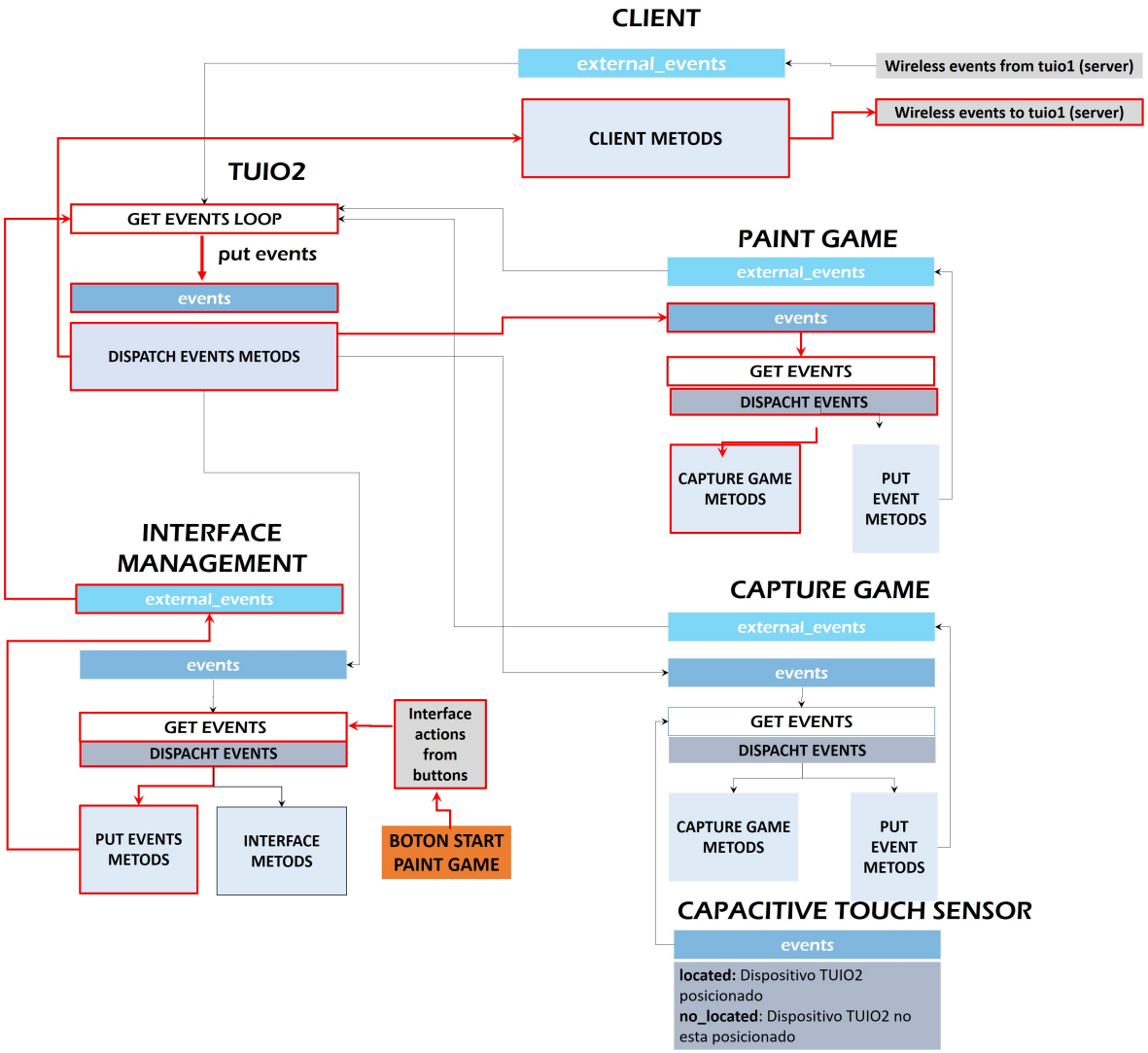

Figura 7.3: Manejo de eventos en *TUIO2*. Pruebas comenzar juego **PAINT GAME**.

- `reconnect`(tiempo de espera). Agotado tiempo de espera para conectar con *TUIO2/TUIO1*.

LISTEN: El servidor/cliente queda a la espera de conectar con (*TUIO2/TUIO1*). En el caso de no conectar con el cliente/servidor (en un tiempo definido de 10 segundos), es ejecutada la transición `reconnect`. Al conectar con *TUIO2/TUIO1* (cliente/servidor), *ServerFSM/-ClientFSM* ejecuta la transición `wait_data` hasta el estado **CONNECT**, a la espera de recibir datos desde *TUIO2/TUIO1*.

Transiciones/Métodos.

- `reconnect`: la máquina vuelve al estado **IDLE**, al agotarse el tiempo de espera para conectar con el cliente/servidor. Se crea el evento *init_client*, para volver a iniciar el servidor/cliente.
- `close_server/close_client`: transición al estado **EXIT** al recibir el evento `close_ser-`

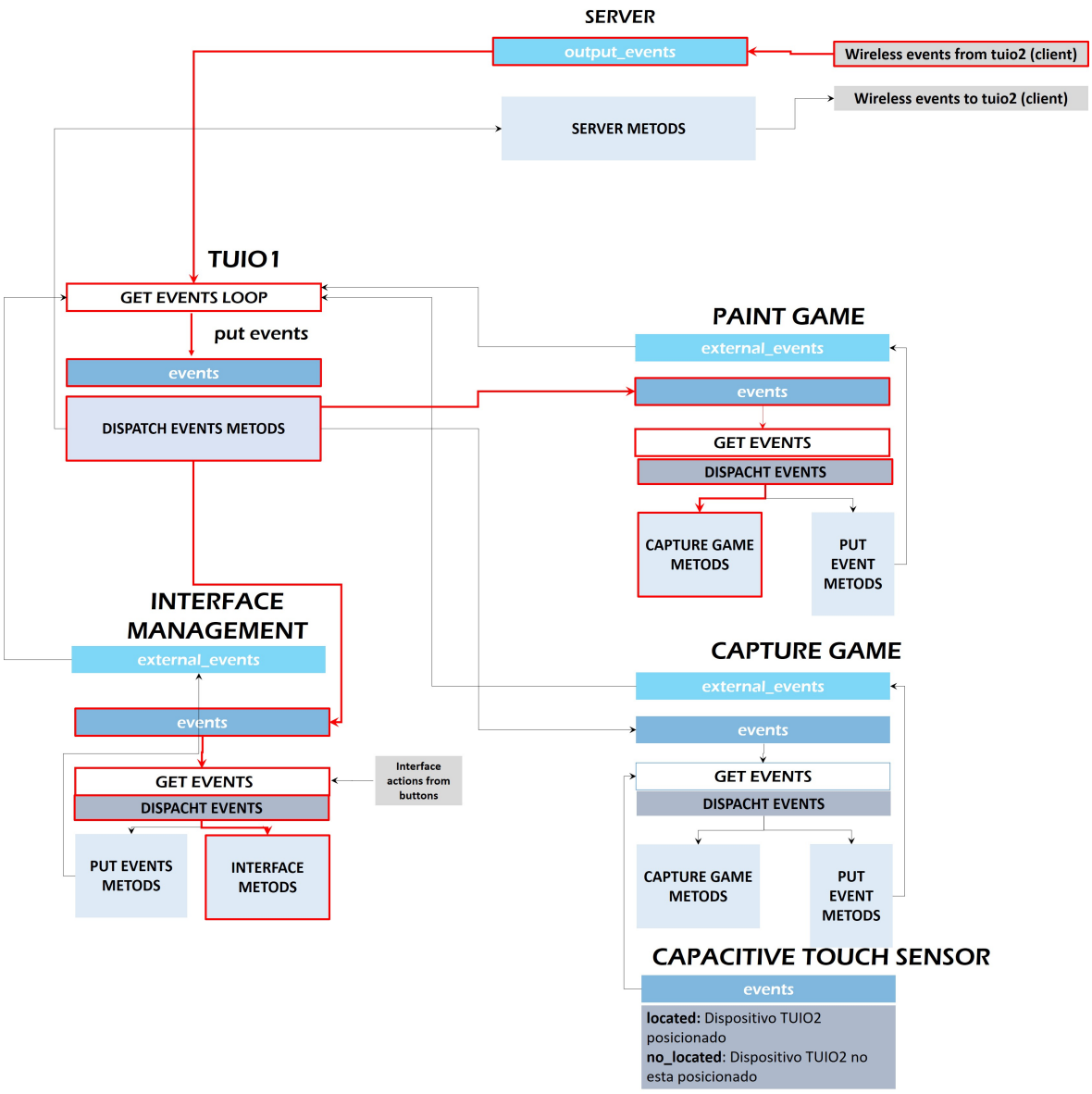

Figura 7.4: Manejo de eventos en *TUIO1*. Pruebas comenzar juego **PAINT GAME**.

`ver/close_client` desde la máquina de estados `Tuio1FSM/Tuio2FSM`, para cerrar las comunicaciones.

- `init_server/init_client`: el servidor/cliente queda a la espera de conectar con *TUIO2/TUIO1*.

CONNECT: Existe comunicación entre los dispositivos *TUIO1* y *TUIO2*. Dispone de dos entradas y una transición posible: recibir datos, enviar datos, o volver a iniciar el servidor si la conexión ha sido interrumpida respectivamente.

Transiciones/Métodos.

- `reconnect`: la máquina vuelve al estado **IDLE**, al interrumpirse la comunicación con el cliente/servidor. Se crea el evento *init_server/init_client*, para volver a iniciar el

servidor/cliente.

- `close_server/close_client`: transición al estado **EXIT** al recibir el evento `close_server/close_client` desde la máquina de estados `Tuio1FSM/Tuio2FSM`, para cerrar las comunicaciones.
- `receive_data`: es ejecutada si se reciben datos desde *TUIO2/TUIO1*. Los datos son añadidos a la cola de eventos de salida, la cual es manejada por *Tuio1FSM/Tuio1FSM* (por medio de `data_received`). Se mantiene en el estado **CONNECT**, a la espera de recibir mas datos desde *TUIO2/TUIO1*.
- `send_data`: método para enviar datos al cliente/servidor.

EXIT: Estado de la máquina por el cual se finaliza el servidor/cliente.

Transiciones/Métodos.

`close_server/close_client`: evento de entrada para finalizar las comunicaciones.

La Figura 7.5 muestra la estructura de la máquina de estados del servidor/cliente de *TUIO1/TUIO2*.

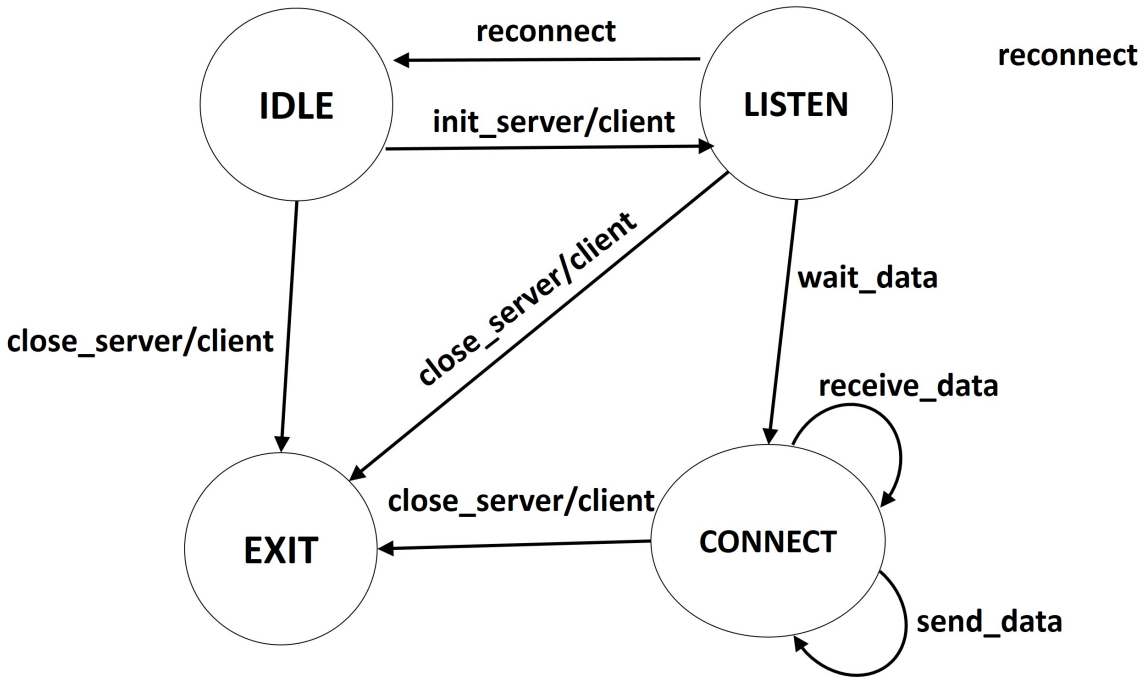

Figura 7.5: Estructura máquina de estados finitos del servidor/cliente.

7.2.2 Gestión de eventos Servidor/Cliente.

Los eventos de entrada desde servidor/cliente son añadidos a la cola `output_events`, donde son adquiridos por el gestor de eventos de `Tuio1FSM/Tuio2FSM`, con el método `send_event(event)`, donde `event` es el evento a enviar. Tanto el servidor como el cliente no dis-

ponen de eventos de salida. Para enviar eventos desde un dispositivo a otro, se utilizan los métodos `server.output_event.get()/server.output_event.get()` de clase de `ServerFSM/ClientFSM`. En la Figura 7.6, se muestra de forma gráfica, la gestión de eventos.

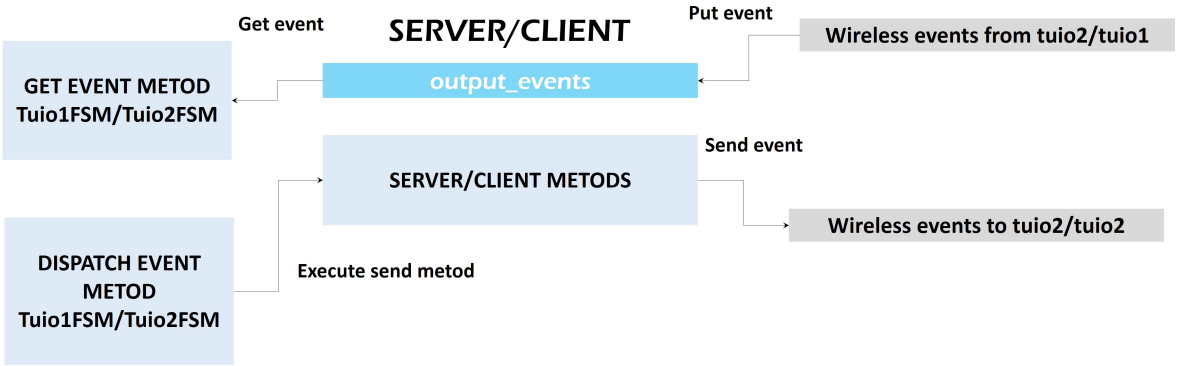

Figura 7.6: Estructura manejo de eventos Servidor/Cliente.

7.2.3 Pruebas de validación en las comunicaciones cliente-servidor.

El programa de pruebas consiste en realizar él envió/recepción de datos entre los dispositivos, donde se realizan secuencias básicas en las comunicaciones. La secuencia del programa de pruebas en las comunicaciones cliente servidor, es la siguiente:

El servidor se inicia en el *puerto 8888* y queda a la espera de la conexión del cliente. Se imprime por pantalla lado servidor: «Esperando cliente».

Una vez el cliente establece la comunicación con el servidor, se imprime por pantalla en lado servidor: «Cliente conectado». De igual manera, en el lado cliente, se imprime por pantalla: «Conectado a servidor».

Se crea un evento para indicar a ambos dispositivos que se inicie el juego (estado **GAME**). Se imprime por pantalla el mensaje: «Juego iniciado»

Cuando ambos dispositivos están en el estado **GAME**, el servidor envía una petición de datos de la unidad de medición inercial *MPU-9255*.

Este evento es gestionado en el programa principal `tuio2`, que devuelve la petición solicitada del servidor. Se imprime por pantalla: «Datos enviados al servidor.». Son mostrados los datos recibidos en el servidor una vez manejado el evento de entrada: «Datos recibidos del cliente: »

Es finalizada la conexión y la aplicación de ambos dispositivos, mostrando respectivamente: «Aplicación finalizada»

La Figura 7.7 muestra de manera gráfica, la secuencia básica expuesta para las pruebas de las comunicaciones.

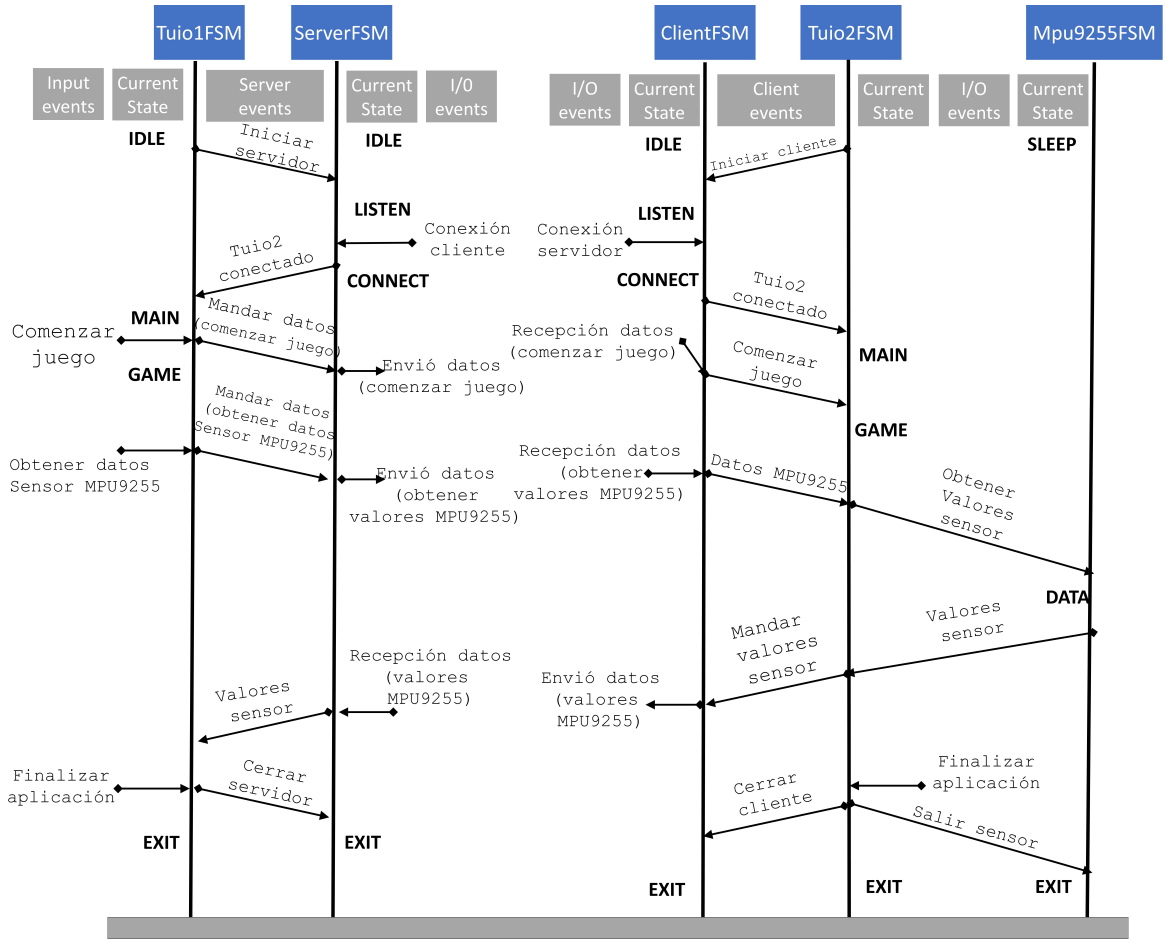

Figura 7.7: Estructura de la secuencia de las pruebas cliente-servidor.

7.3 Sistema de localización.

El sistema de localización consiste en obtener las coordenadas de posición y ángulo, del *Widget tangible TUIO2*, al ser posicionado sobre la pantalla táctil capacitiva del dispositivo *TUIO1*. Estos eventos táctiles siguen un patrón determinado, cuya información es procesada para una interacción gráfica y tangible entre dispositivos. Son dos las interacciones posibles mediante el sistema de localización: incorporar información gráfica de *TUIO2* a *TUIO1*, o *extraer* información desde *TUIO1* a *TUIO2*.

Para generar estos eventos táctiles, el dispositivo *TUIO2* dispone de tres almohadillas conductoras, que al tomar contacto con la pantalla capacitiva, se generan varios eventos táctiles simultáneos.

7.3.1 Versión de pruebas 1. Localización de coordenadas con patrón de triángulo rectángulo escaleno.

Ejemplo de aplicación del sistema de localización. Antes de posicionar *TUIO2* sobre *TUIO1*, las pantallas de ambos dispositivos muestran imágenes diferentes (ver Figura 7.8).

Al posicionar *TUIO2* sobre la pantalla de *TUIO1*, la pantalla de *TUIO2* muestra la parte de

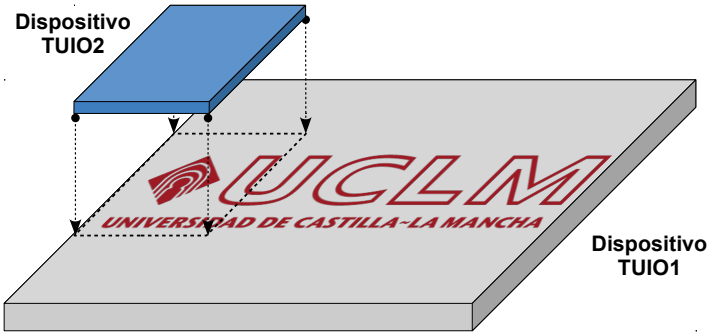

Figura 7.8: Versión de pruebas no validas. Posición de los dispositivos antes de la interacción sobre la pantalla capacitiva.

imagen correspondiente que es tapada por el dispositivo. (Figura 7.9).

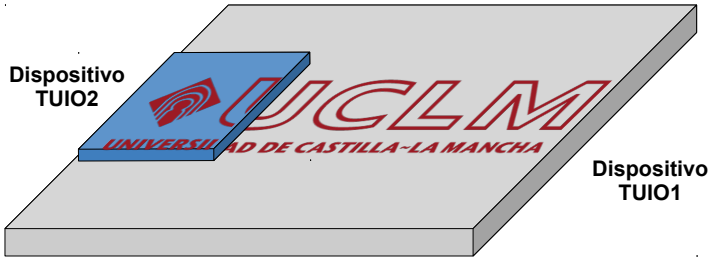

Figura 7.9: Versión de prueba no valida. Posición de los dispositivos después de la interacción sobre la pantalla capacitiva.

El procedimiento para obtener la posición, se consigue con el uso del sensor capacitivo de la propia pantalla de *TUIO1* que, al detectar una variación de la capacitancia, genera un evento táctil. El dispositivo *TUIO2* produce tres eventos táctiles sobre la pantalla capacitiva. Estos eventos son generados por medio de tres almohadillas de goma conductora, que están conectadas al borne negativo de la *RaspberryPi*. Esta conexión evita que el usuario tenga que sostener con su propia mano el dispositivo, haciendo de conexión a tierra el propio borne negativo. El diseño de los puntos de contacto es el que se muestra en la Figura 7.10. Los tres puntos de contacto son identificados con las letras **A, B** y **C**, que dan nombre a los vértices de un triángulo rectángulo escaleno que forman. Se ha elegido este tipo de diseño, ya que cada uno de los tres lados tiene una medida diferente, lo que hace que sea más fácil

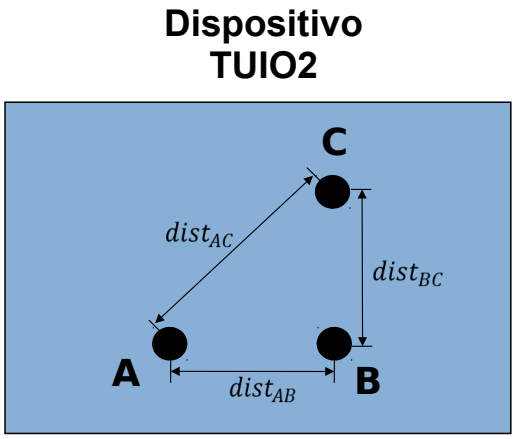

Figura 7.10: Versión de prueba no valida. Disposición de las almohadillas conductoras en la parte inferior del *widget tangible* para generar los eventos táctiles.

identificar cada segmento del triángulo, y así evitar posibles problemas cuando se localice cualquiera de sus vértices.

Cuando *TUIO2* está sobre la pantalla, los eventos táctiles generados, proporcionan la siguiente información al dispositivo *TUIO1*: posición de las coordenadas (x,y), y número de evento. Las coordenadas de la posición de cada evento táctil está expresada en *pixels*, por lo que cada evento producido será manejado en esas unidades de media (ver Figura 7.11).
Estos datos son interpretados y tratados para calcular la posición de *TUIO2* en una librería

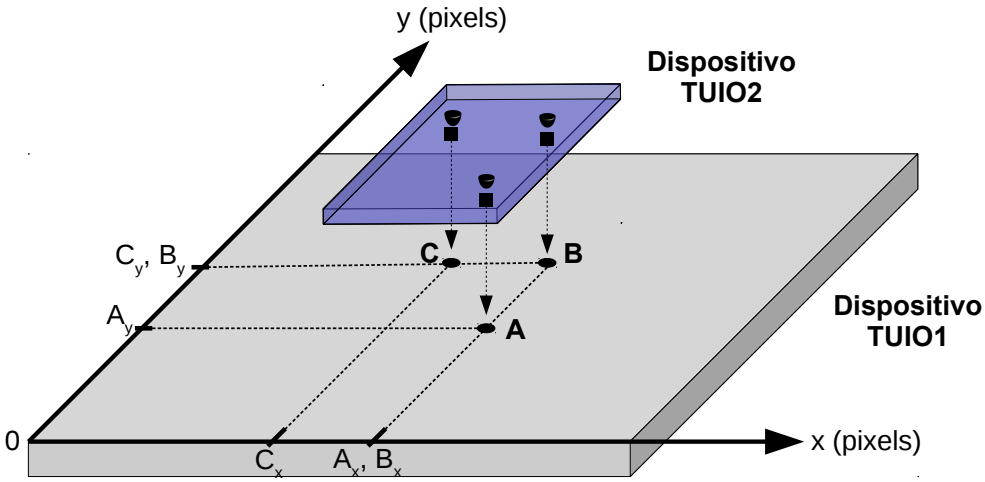

Figura 7.11: Versión de prueba no valida. Coordenadas de los puntos **A, B** y **C** del *widget tangible* sobre la pantalla capacitiva.

desarrollada en *Python*.

Librería para la localización (Versión de prueba no valida).

Todos los eventos táctiles generados sobre la pantalla capacitiva del dispositivo *TUIO1*, son manejados por la *clase Widget* de *Kivy*, con el método `on_touch_down()`.

Este método obtiene la información de posición e ID del evento táctil producido, almacenando estos datos en la lista `touch[]`.

Se crea una librería específica para el manejo de esta lista, la cual tratará y manejara los datos para obtener la información de posición de *TUIO2*.

Método `pulsacion(touch)` (Versión de prueba no valida).

El argumento de entrada, corresponde con las coordenadas y número de evento táctil, que contiene la lista de eventos (`touch`).
Estos datos son almacenados en la última posición de la lista `puls[]` aplicando el método `append()`:
`self.puls.append((t.x,t.y,t.id)).`
donde:
`t.x`: coordenada x del evento táctil.
`t.y`: coordenada y del evento táctil.
`t.id`: identificación del evento táctil (número de evento).

Cuando el número de eventos almacenados en `puls[]` es mayor que 1, se llama al método `determinar_distancias()`.

Método `determinar_distancias()` (Versión de prueba no valida).

El método `Vector()`, de la librería `kivy`, calcula la distancia entre los eventos táctiles producidos hasta el momento, que son almacenados en la lista `puls[]`. Estos eventos son recorridos desde el último elemento de la lista `puls[]`, hasta el primero. El método `determinar_lados` determina las distancias entre puntos. Si la distancia es correcta, se incrementa la variable `contador`, la cual al llegar a 2, llama al método `determinar_puntos()`.

Método `determinar_lados()` (Versión de prueba no valida).

Método para verificar si dos eventos táctiles, cuyas distancias ya han sido determinadas, corresponden con uno de los lados del triángulo en el *widget tangible*, donde *m* corresponde con la distancia entre los dos eventos táctiles, *i* el tamaño de la lista `puls[]`, y `a` el número de iteración del bucle que recorre cada uno de los elementos de la lista.

Este método es llamado cada vez que el método `determinar_distancias()` es invocado. Se comprueba si la distancia entre los dos puntos corresponde con el rango de medidas establecidas para los lados del triángulo del *widget tangible*. Los rangos de medida se muestran en la tabla 7.1.

Rango	Segmento del triángulo
(120,150)	AB
(160,150)	BC
(210,260)	AC

Cuadro 7.1: Intervalos de medida de cada segmento del triángulo

Si la medida corresponde con alguno de los rangos establecidos, dicha medida es almacenada junto con las coordenadas de ambos eventos en la lista `dist[]`. Por ejemplo, si la distancia entre los dos eventos táctiles es correcta, y corresponde al segmento **AB**, los datos son añadidos en la lista `dist[]`:

```
self.dist.append((m,'AB',toq1,toq2))
```

donde `m` corresponde a la distancia entre ambos puntos, `AB` es el nombre del segmento al que corresponde, `toq1` son las coordenadas del evento táctil con el cual se está verificando la distancia en el método `determinar_distancias()`, y `toq2` son las coordenadas del ultimo evento táctil. Si la distancia es correcta se incrementa en una unidad la variable `contador`.

Método `determinar_puntos()` (Versión de prueba no valida).

Este método es invocado dentro del método `determinar_distancias()`, cuando la variable `contador` es mayor a 1, es decir, existen dos segmentos correctos del triángulo cuyos datos han sido almacenados en la lista `dist[]`, la cual contiene en ese momento dos elementos. Los puntos del triángulo **(A,B,C)**, son calculados siguiendo la siguiente lógica:

Si los segmentos de los elementos de `dist[]` corresponden con $\overline{BC}$ y $\overline{AB}$ respectivamente, se llama al método `asignar_puntos`, donde los argumentos de entrada se establecen:

```
self.asignar_puntos(1,2,0,'B','C','A')
```

Método `asignar_puntos()` (Versión de prueba no valida).

Este método asigna a cada vértice **A, B** y **C** del triángulo, las coordenadas donde se encuentran cada uno de ellos, al igual que el nombre del vértice. Estos datos son almacenados en la lista `coordenadas[]`

Para establecer un orden a la hora de añadir valores a la lista `coordenadas[]`, es pasado como argumento, la posición a la que corresponden cada uno de los puntos.

0: corresponde a la posición de la letra A en la lista `coordenadas[]`.
1: posición de la letra B para la lista `coordenadas[]`.
2: posición de la letra C en la lista `coordenadas[]`.

La finalidad es que la lista `coordenadas` quede con la siguiente estructura, para ser manejada:

$[(x_A, y_A,' A'), (x_B, y_B,' B'), (x_C, y_C,' C')]$

En el ejemplo anterior, los elementos de `dist[]` corresponden con $\overline{BC}$ y $\overline{AB}$. El punto en común para ambos segmentos corresponde a **B**, **C** para el primer segmento, y **A** para el segundo segmento.

El método para asignar los puntos es:

`self.asignar_puntos(1,2,0,'B','C','A')`

donde los argumentos de entrada son: c = 1, posición del punto en común entre los dos segmentos. Como el punto en común es *B* que corresponde con la posición 1 de la lista `coordenadas[]`.
c1 = 2, es la posición del punto **C** para el ejemplo según el criterio establecido.
c2 = 0, corresponde a la posición del punto **A** para la lista `coordenadas[]`.
p = 'B', Nombre del vértice en común, en este ejemplo corresponde al vértice **B**.
p1 = 'C', Nombre del vértice restante del primer segmento, en este caso el primer segmento es el $\overline{BC}$, por lo tanto, el vértice corresponde al **C**.
p2 = 'A', Nombre del vértice del segundo segmento, que corresponde con el vértice A.

Los valores que contiene la lista `dist[]` para ser manejados por el método `asignar_puntos()` son: `[(0, 'NADA'), (164.0, 'BC', (440.00000000000006, 346.0, 'mouse2'), (440.0000000000000 182.0, 'mouse1')), (134.03357788255903, 'AB', (306.0, 179.0, 'mouse3'), (440.00000000000006, 182.0, 'mouse1'))]`

La lista es recorrida mediante el siguiente bucle (ver Listado 7.1):

```
for a in r
        for b in range(2):
                if self.dist[1][a+2][2] == self.dist[2][b+2][2]:
                        self.coordenadas[c] = (self.dist[1][a+2][0],self.dist[1][a+2][1],p)
                        if a == 0:
                                self.coordenadas[c1] = (self.dist[1][3][0],self.dist[1][3][1],p1)
                        else:
                                self.coordenadas[c1] = (self.dist[1][2][0],self.dist[1][2][1],p1)
                        if b == 0:
                                self.coordenadas[c2] = (self.dist[2][3][0],self.dist[2][3][1],p2)
                        else:
                                self.coordenadas[c2] = (self.dist[2][2][0],self.dist[2][2][1],p2)
                        break
```

Listado 7.1: Bucle para asignar las coordenadas de los vértices (Versión de prueba no valida).

Donde `a` es la posición para el primer elemento de la lista `dist[]`, y `b` la posición del

segundo elemento de la lista.

Para poder analizar mejor este ejemplo, separamos los dos elementos de la lista `dist[]`:

```
Elemento 1 de dist[]:

(164.0, 'BC', (440.00000000000006, 346.0, 'mouse2'), (440.00000000000006, 182.0, 'mouse1'))

Elemento 2 de dist[]:

(134.03357788255903, 'AB', (306.0, 179.0, 'mouse3'), (440.00000000000006, 182.0, 'mouse1'))
```

La *posición 0* corresponde a la distancia entre segmentos, la *posición 1* es el nombre del segmento, y las dos restantes posiciones son las coordenadas de los puntos del segmento junto con la *id* del evento táctil.

`id: 'mouse1'` es el elemento en común para los dos segmentos, en este caso $\overline{BC}$ y $\overline{AB}$, donde el punto en común es el punto *B*. Por lo tanto:

```
self.coordenadas[c] = (self.dist[1][a+2][0],self.dist[1][a+2][1],p)
```

Las coordenadas del punto **B** (coordenada x y coordenada y) son almacenadas en la lista `coordenadas[]`, al igual que el nombre del vértice común (argumento `p = 'B'`), en la posición `c`, que como argumento de entrada se indicó como 1 (punto **B**).

La posición de `'mouse1'` para el primer elemento de `dist[]`, es para un valor de `a` igual a 1 (ya que al valor de `a`, se le suma 2 posiciones, por ser los dos primeras, la distancia entre segmentos, y el nombre del segmento), es decir, la posición 3 del elemento 1 de `dist[]`.

Para el segundo elemento, la posición del vértice *B* corresponde a un valor de `b = 1` como en el caso del primer segmento. Una vez asignada las coordenadas del punto en común, el nombre del vértice, y almacenados dichos datos en la lista `coordenadas[]` en la posición 1, son determinadas las coordenadas de los dos puntos restantes del triángulo (**C** y **A**): Con los tres vértices del triángulo, se calcula el área del triángulo mediante la llamada al método `calculo_area`, para verificar que el triángulo es correcto y corresponde con las posiciones del *widget tangible* y no responde a un error de eventos sobre la pantalla táctil.

Método `calculo_area()` (Versión de prueba no valida). El método `calculo_area`, retorna el valor del área, la cual, se debe encontrar en el intervalo (9545,16000). Si el área no es correcta, se realiza una llamada al método `inicializar_a_0`, para restablecer los valores a sus condiciones iniciales.

Se aplica la *regla de Sarrus* (determinante), para el cálculo del área. El argumento de es-

trada es la lista `coordenadas[]` que contiene las coordenadas de los vértices del triángulo:

```
  det = abs((c[0][0]*c[1][1])+(c[0][1]*c[2][0])+(c[2][1]*c[1][0])
-((c[1][1]*c[2][0])+(c[0][1]*c[1][0])+(c[0][0]*c[2][1])))*0.5.
```

Validación de la librería para el sistema de localización.

Las pruebas realizadas siguiendo el patrón de un triángulo equilátero escaleno, arrojan fallos significativos durante el proceso del cálculo de las coordenadas y ángulo de posición, del dispositivo *TUIO2*, destacando los siguientes problemas:

- Determinación incorrecta de los vértices en distintas etapas de la localización.
- La adquisición de datos queda en estado de espera, cuando son detectados eventos táctiles involuntarios sobre el sensor capacitivo.
- Medidas erróneas captadas por el sensor capacitivo, debidas a la poca estabilidad de los puntos de contacto de las almohadillas conductoras, retornando valores inexactos de posición.
- Poca organización del código y del tratamiento de datos.

Todos estos resultados obligan a determinar un nuevo patrón de localización, al igual que simplificar los procedimientos de almacenamiento y procesado de datos.

7.3.2 Versión de pruebas 2. Localización de coordenadas con patrón lineal.

Esta versión de pruebas del sistema de localización tiene como objetivo optimizar los procesos de adquisición y tratamiento de datos, aplicando diferentes patrones para determinar la posición y ángulo del dispositivo *TUIO2*.

La situación de las almohadillas conductoras ha sido modificada, aplicando un patrón de tres puntos **(0, A, B)** situados en paralelo a la parte inferior del dispositivo, tomando como puntos de referencia, la esquina posterior izquierda **(0)**, y posterior derecha**(B)**. El tercer punto queda situado a una distancia de **(0-A) >(A-B)** (ver Figura 7.12). Tomando como referencia un vector, el punto B determinaría la dirección de dicho vector.

El procedimiento de calculo obtiene las coordenadas del punto 0, que corresponde con el origen de coordenadas de las representaciones gráficas en *TUIO2*. La adquisición de la posición y ángulo del dispositivo tiene la siguiente finalidad:

- Punto de referencia 0: trasladar la representación gráfica del dispositivo *TUO2*, acorde con la representación gráfica de *TUIO1* al ser situado sobre el sensor capacitivo.
- Ángulo de inclinación: obtener una correcta representación gráfica en *TUIO2*, al ser posicionado con un ángulo de inclinación con respecto al sistema de coordenadas de la representación gráfica de *TUIO1*.

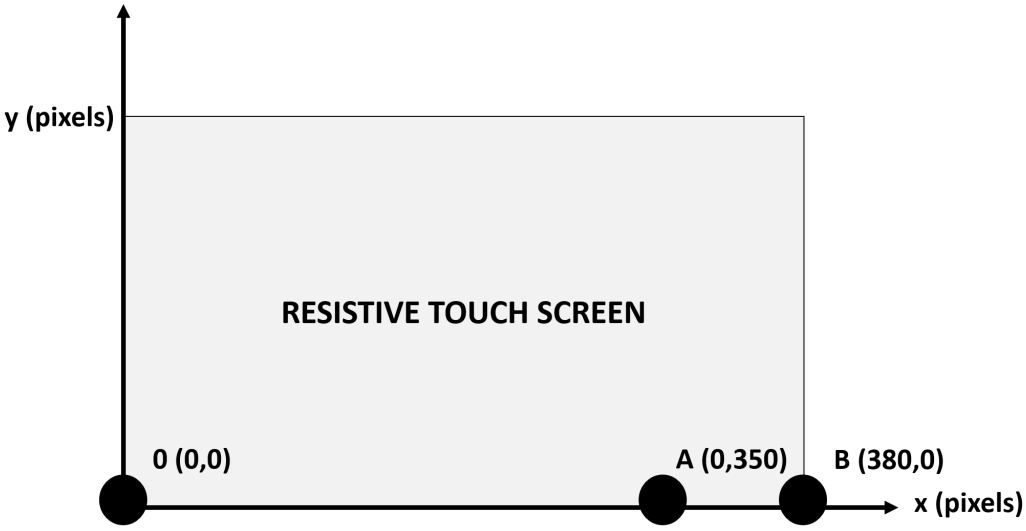

Figura 7.12: Situación de los puntos **A, B** y **C** del *widget tangible* en *TUIO2*.

Estos datos, junto con un escalado correcto de los elementos gráficos en *TUIO2*, permite obtener de manera correcta, los elementos situados bajo el dispositivo al ser detectado por el sensor capacitivo (Figura 7.13).

Librería para la localización.

El módulo desarrollado para la localización se divide en dos clases. Clase para la obtención de eventos táctiles: `TouchInput`. Clase para el cálculo del ángulo y punto de referencia. `TouchPositions`.

Clase `TouchInput`.

Clase que hereda de la clase `Widget` del módulo `kivy.uix.widget`. Esta clase obtiene los eventos táctiles producidos sobre la pantalla de manera automática. Se crea el objeto `touch_positions` de la Clase `TouchPosition`, para realizar los cálculos a partir de los eventos producidos.

Métodos de clase.

- `on_touch_down()`: el atributo `touch` indica la posición y número de toque del evento táctil. El evento es registrado en el *diccionario* `touchs`, del objeto `touchs_positions`.
- `on_touch_up()`: obtiene los datos de posición y número de evento táctil, al ser desacoplado. El número de evento es utilizado para ser eliminado del diccionario de eventos táctiles `touchs`.

Clase `TouchPositions`.

Esta clase hereda de la clase base `object`, y contiene los métodos para el calculo de las posiciones de los puntos de contacto de los eventos táctiles.

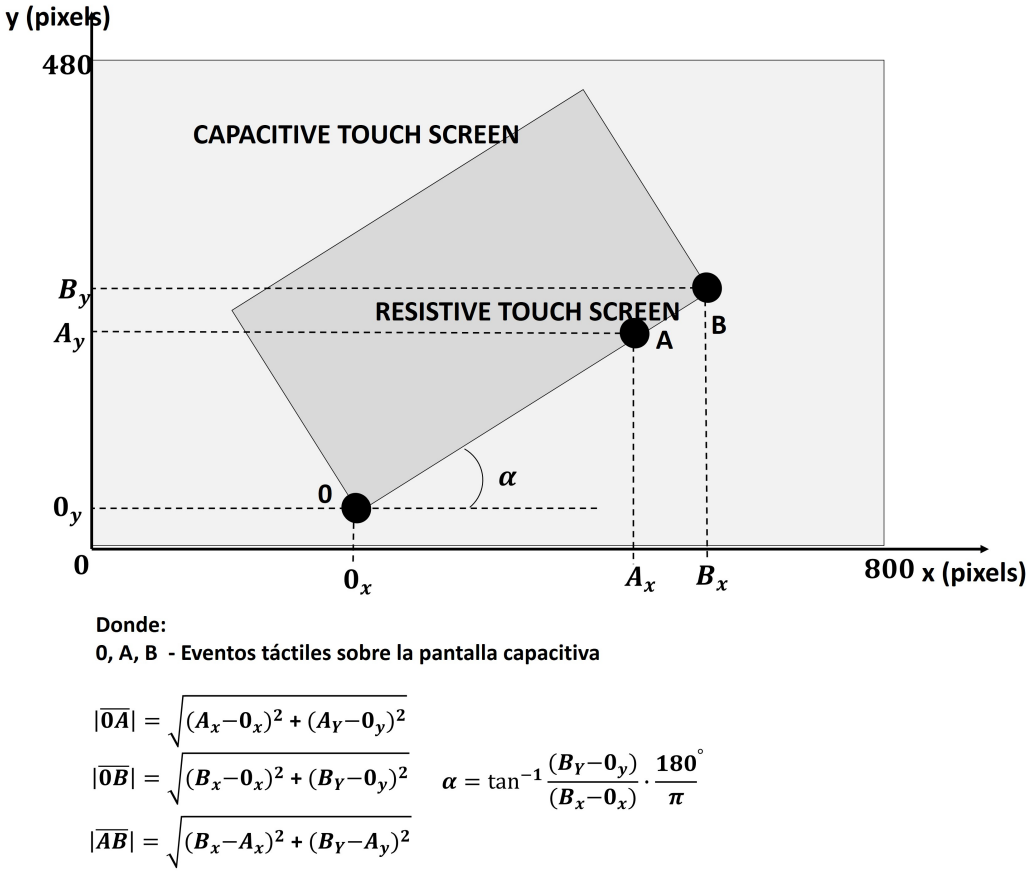

Figura 7.13: Situación de los puntos **A, B** y **C** del *widget tangible* en *TUIO2*, sobre la pantalla táctil capacitiva del dispositivo *TUIO1*

Métodos de clase.

- `add_touch()`. Añade los eventos táctiles en el diccionario `touch`, tomando como clave el número de evento táctil.
- `remove_touch()`: Elimina los eventos táctiles que han sido desacoplados, utilizando la clave del evento táctil sobre el diccionario de datos `touch`.
- `calculate_distance()`: Calcula la distancia entre dos eventos táctiles, utilizando el método `Vector` de la clase `Vector` del módulo `kivy.vector`. Llamada al método `calculate_segments` para verificar si la distancia corresponde con el patrón predefinido. La distancia es guardada de manera temporal en la variable local `dist`.
- `calculate_segments()`: Determina si la distancia corresponde con los márgenes del patrón de localización. Si el segmento es correcto, se almacenan las coordenadas de los puntos del segmento en un diccionario de datos, con las tres claves posibles `(0A)`, `(0B)`, `(AB)`.
- `calculate_points()`: Obtiene la correspondencia de los puntos **0, A, B**, con las coordenadas de los eventos táctiles. Las coordenadas son almacenadas en tres variables

locales, para ser usadas como argumento de entrada en el método `calculate_angle`.

- `calculate_angle()`: Las coordenadas del punto de referencia **0** y las coordenadas el punto **A/B**, se calcula el ángulo del sistema de coordenadas de *TUIO2*, con respecto al sistema de coordenadas de *TUIO1*, con la función `atan2` de la librería `math`.

Los datos de las coordenadas del punto de referencia **0** de *TUIO2*, y el ángulo, son almacenados en la cola de eventos `events`.

Validación de la librería para el sistema de localización.

Las pruebas de localización realizadas con un patrón lineal arrojan ciertas mejorías sobre el patrón triangular aplicado en la primera versión de pruebas, destacando los siguientes aspectos:

- Determinación correcta de los puntos de coordenadas **A,B,C** correspondientes al patrón lineal.

- Mejora en el cálculo del ángulo del sistema de coordenadas de *TUIO2* con respecto a *TUIO1*.

Las variaciones en el ángulo se deben principalmente a errores del sensor capacitivo, con los puntos de contacto de las almohadillas conductoras.

7.4 Diseño software de la interfaz gráfica.

7.4.1 Manejo de eventos de la interfaz.

Todos los eventos relacionados con cambios producidos en las representaciones gráficas en la plataforma de juego, son gestionados con los métodos de la clase `InterfaceManagement`, y el objeto de clase `ScreenManager`.

`InterfaceManagement` maneja los eventos de entrada desde el módulo principal `Tuio1FSM/Tuio2FSM`, y los eventos táctiles procedentes desde la propia interfaz gráfica de usuario.

`ScreenManager` gestiona los eventos de entrada desde los módulos dedicados a juegos, principalmente para realizar cambios de pantalla (`Screen`).

Instancia del objeto `screenmanager`.

El objeto `screenmanager` es instanciado de manera global, para facilitar el manejo de eventos gráficos por el resto de los módulos de la aplicación, ya que los juegos disponen de sus propias pantallas (`Screens`), y sus propios objetos gráficos, que son añadidos a las mismas.

La estructura del manejo y gestión de eventos sigue una distribución secuencial, como se muestra en la Figura 7.14).

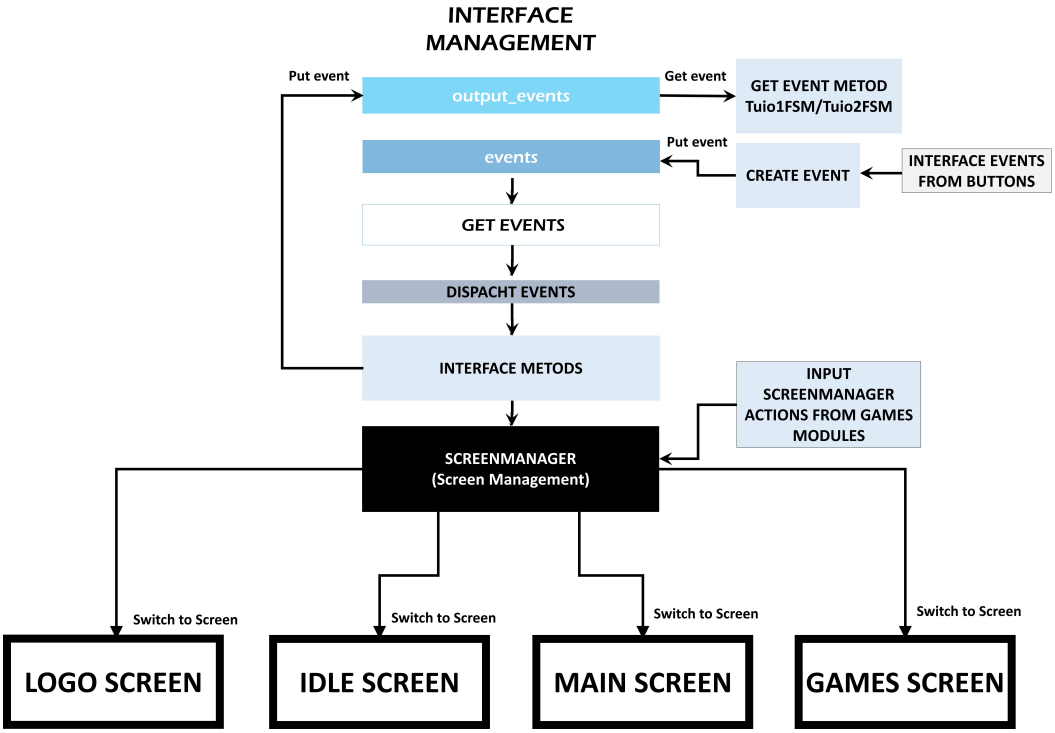

Figura 7.14: Estructura de la gestión de eventos del módulo `InterfaceManagement()`.

7.4.2 Módulo para el manejo de la interfaz `InterfaceManagement`.

Módulos `ScreenManager` y `Screen`.

`ScreenManager` y `Screen`, son módulos del paquete `uix` de la librería `Kivy`, que disponen de los métodos de clase básicos para el diseño, y manejo de eventos gráficos generados durante el transcurso de la aplicación.

`ScreenManager` es un *widget* dedicado a administrar múltiples pantallas en la aplicación. Muestra solo una pantalla (`Screen`) a la vez, y usa la librería `TransitionBase` para cambiar de una pantalla a otra.

`Screen` es un objeto destinado a ser utilizado con `ScreenManager`, donde son incorporados los diferentes elementos que componen la interfaz gráfica de usuario.

Funcionamiento básico de `ScreenManager`.

Importar módulos: `from kivy.uix.screenmanager import ScreenManager , Screen`

Crear el objeto `sm` (*ScreenManager*): `sm = ScreenManager()`

Crear una pantalla: `screen = Screen(name='screen_1')`

Añadir una pantalla al administrador: `sm.add_widget(screen)`

Cambiar a la pantalla `'screen_1'`: `sm.current = 'screen_1'`

Clases de la interfaz gráfica.

Clase `Widget()`. Clase base requerida para crear *widgets*.

Método de importación del módulo: `from kivy.uix.widget import Widget` //

Clase `Button()`. `Button` es una clase del módulo `Label`, que dispone de acciones asociadas, que son ejecutadas al ser presionado el botón, o cuando se deja de presionar. //

Método de importación del módulo: `from kivy.uix.button import Button`

Pantallas (`Screens`), manejadas por los gestores de eventos `Tuio1FSM/Tuio2FSM`.

- **Pantalla Logo. `LogoScreen`**. Pantalla de bienvenida, que ejecuta una pequeña animación del nombre de la plataforma, al iniciar la aplicación.
- **Pantalla Idle. `IdleScreen`.** Pantalla que muestra una imagen (fondo de pantalla), indicando que la conexión no ha sido establecida. Se mantiene en ese estado hasta ejecutar el método de cambio de pantalla desde el gestor de eventos principal.
- **Pantalla Main. `MainScreen`.** Pantalla principal de la aplicación, una vez son establecidas las comunicaciones entre ambos dispositivos.

Métodos de clase para el manejo de la interfaz, con el módulo `InterfazManagement` 7.2:

Método	Descripción
`init_interface()`	Inicio de la interfaz gráfica. Cambio a la pantalla `'logo_screen'` (pantalla de bienvenida).
`create_event()`	Añade a la cola de eventos con el método `put()`, el argumento de entrada al método.
`get_event()`	Obtiene el evento de la cola de eventos, con el método `get()`.
`dipatch_event()`	Según el tipo de evento, se ejecutan los métodos de clase.
`main_state()`	Ejecuta un cambio a la pantalla `'main_screen'` con `ScreenManager`.
`idle_state()`	Ejecuta un cambio a la pantalla `'idle_screen'` con `ScreenManager`.
`close_interface()`	Termina todos los procesos de la interfaz y de la aplicación, con el método textttstop() de la clase `App`.

Cuadro 7.2: Métodos de la clase `InterfaceManagement()`

7.5 Lectura y configuración de sensores.

7.5.1 Unidad de medición inercial MPU-9255.

La unidad de medición inercial integrada en la plataforma, es el sensor *MPU9255*. Este sensor forma parte del dispositivo *TUIO2*. Esta destinado principalmente a capturar eventos

de movimiento, para generar acciones durante el transcurso del juego. Al realizar movimientos angulares sobre los ejes X,Y,Z del dispositivo *TUIO2*, se generan unos eventos que son filtrados y comunicados al dispositivo *TUIO1*, para ser ejecutados durante el transcurso del juego. Por ejemplo, girar una pieza representada en *TUIO1*

Conexionado Bus I2C MPU-9255.

La conexión con la placa para el desarrollo *Raspberry Pi* para comunicaciones *I2C*, se realiza mediante 4 líneas (ver Figura 5.8).

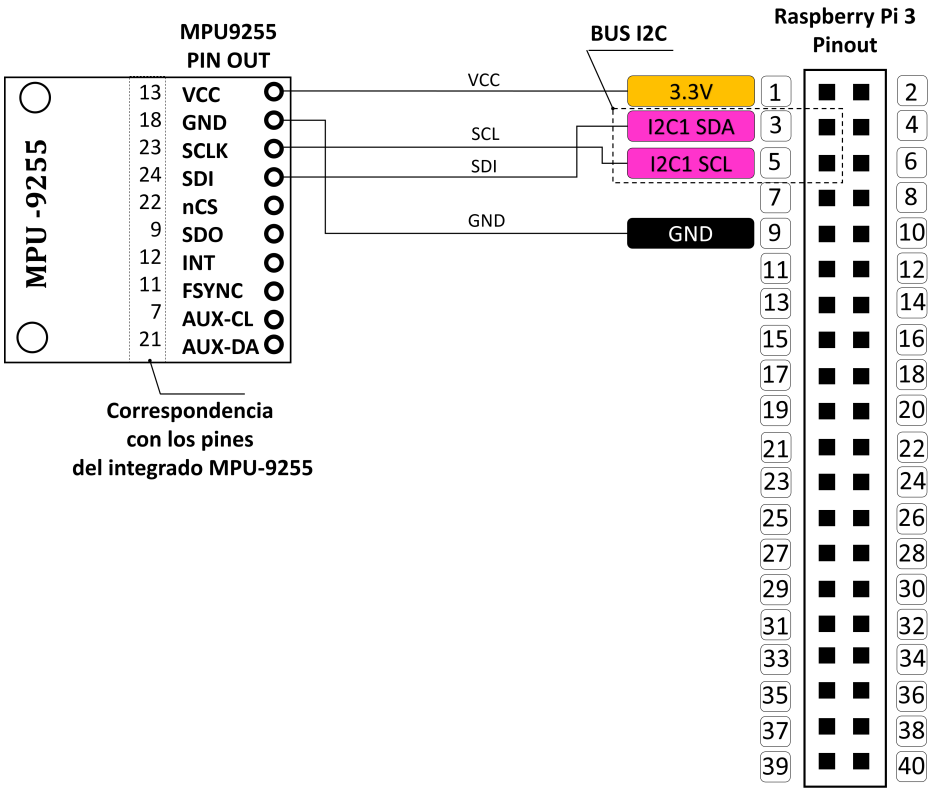

Figura 7.15: Esquema de conexiones con Raspberry Pi para comunicaciones I2C.

Configuración y lectura, de las posiciones de memoria MPU-9255.

Para establecer comunicaciones mediante *Bus I2C*, entre *Raspberry Pi* y el *sensor MPU-9255*, es necesario importar la librería: `smbus`

Las comunicaciones con el *Bus I2C*, son ejecutadas instanciando el objeto de de clase: `bus = smbus.SMBus(1)`

Para leer datos de los registros de memoria, se utiliza el método: `bus.read_byte_data()`. Los argumentos de entrada del método son la dirección de memoria del dispositivo, y la dirección del registro que se desea leer.

Para la escritura de datos, se emplea el método: `bus.write_byte_data()`, que toma como

argumentos de entrada la dirección del dispositivo, la dirección a la que se desea acceder, y el valor a escribir en la dirección de memoria.

La dirección por defecto del sensor *MPU-9255* es: `0x68`

El sensor magnetómetro dispone de una dirección propia para las comunicaciones *I2C*: `0x0C`

Es necesario definir la escala de lectura de cada uno de los sensores. Para ello hay que acceder a la zona de memoria, donde están ubicadas estas configuraciones.

Las direcciones de memoria son:

- Configuración del giroscopio para un fondo de escala de +-250dps: dirección `0x1B` establecer a `0x00`.

- Configuración del acelerómetro para un fondo de escala de +- 2g: dirección `0x1C` establecer a `0x00`.

- Configuración del magnetómetro para un fondo de escala de 14 *bits* dirección `0x0A` establecer a `0x00`.

Tanto el giroscopio, acelerómetro, y magnetómetro, guardan las lecturas de los valores en el mapa de registro del dispositivo. Las direcciones de memoria son las siguientes:

- Rango direcciones de memoria datos de lectura del giroscopio: `0x43 - 0x48`

- Rango direcciones de memoria datos de lectura del acelerómetro: `0x3B - 0x40`

- Rango direcciones de memoria datos de lectura del magnetómetro: `0x03 - 0x08`

Importante a tener en cuenta: las comunicaciones *I2C* transfieren las lecturas *byte* a *byte*, por lo que el orden de lectura arroja una interpretación erronea. Es necesario invertir el orden de la lectura de *bytes*.

Métodos de la clase `MPU-9255`.

Métodos de clase 7.3.

Método	Descripción
`configure_MPU9255()`	Configuración de la escala del acelerómetro y giroscópio.
`sleep_offMPU()`	Establece en activo el sensor MPU-9255, para regresar datos de lectura de las magnitudes.
`configure_AK8963()`	Configuración de la escala de lectura del magnetómetro.
`sleep_offAK()`	Establece en activo el sensor AK8963, para regresar datos de lectura de las magnitudes.
`read_accel()`	Retorno de datos de la aceleración.
`read_gyro()`	Retorno de lectura de datos de los desplazamientos angulares del sensor.
`read_magnet()`	Retorno de valores obtenidos por el magnetómetro AK8963.

Cuadro 7.3: Métodos de la clase `MPU-9255`

7.5.2 Sensor de color.

Los sensores de color detectan el color de una superficie, y entregan la información de manera digital. El dispositivo implementado es el sensor de color *TCS34725*, que proporciona un retorno digital de los valores de detección de luz roja, verde y azul (RGB).

La integración de este dispositivo hace posible una interacción directa con el medio que le rodea, lo que permite ampliar las opciones de juego dentro de la plataforma.

Conexionado Bus I2C TCS34725

La conexión con la placa para el desarrollo *Raspberry Pi* para comunicaciones *I2C*, se realiza mediante 4 líneas (ver Figura 7.16)

Configuración y lectura, de las posiciones de memoria TCS3472.

Para establecer comunicaciones mediante *Bus I2C*, entre *Raspberry Pi* y el *sensor TCS3472*, es necesario importar la librería: `smbus`

Las comunicaciones con el *Bus I2C*, son ejecutadas instanciando el objeto de de clase: `bus2 = smbus.SMBus(1)`

Para leer datos de los registros de memoria, se utiliza el método: `bus2.read_byte_data()`. Los argumentos de entrada del método son la dirección de memoria del dispositivo, y la dirección del registro que se desea leer.

Para la escritura de datos, se emplea el método: `bus2.write_byte_data()`, que toma como argumentos de entrada la dirección del dispositivo, la dirección a la que se desea acceder, y el valor a escribir en la dirección de memoria.

La dirección por defecto del sensor *TCS3472* es: `0x29`

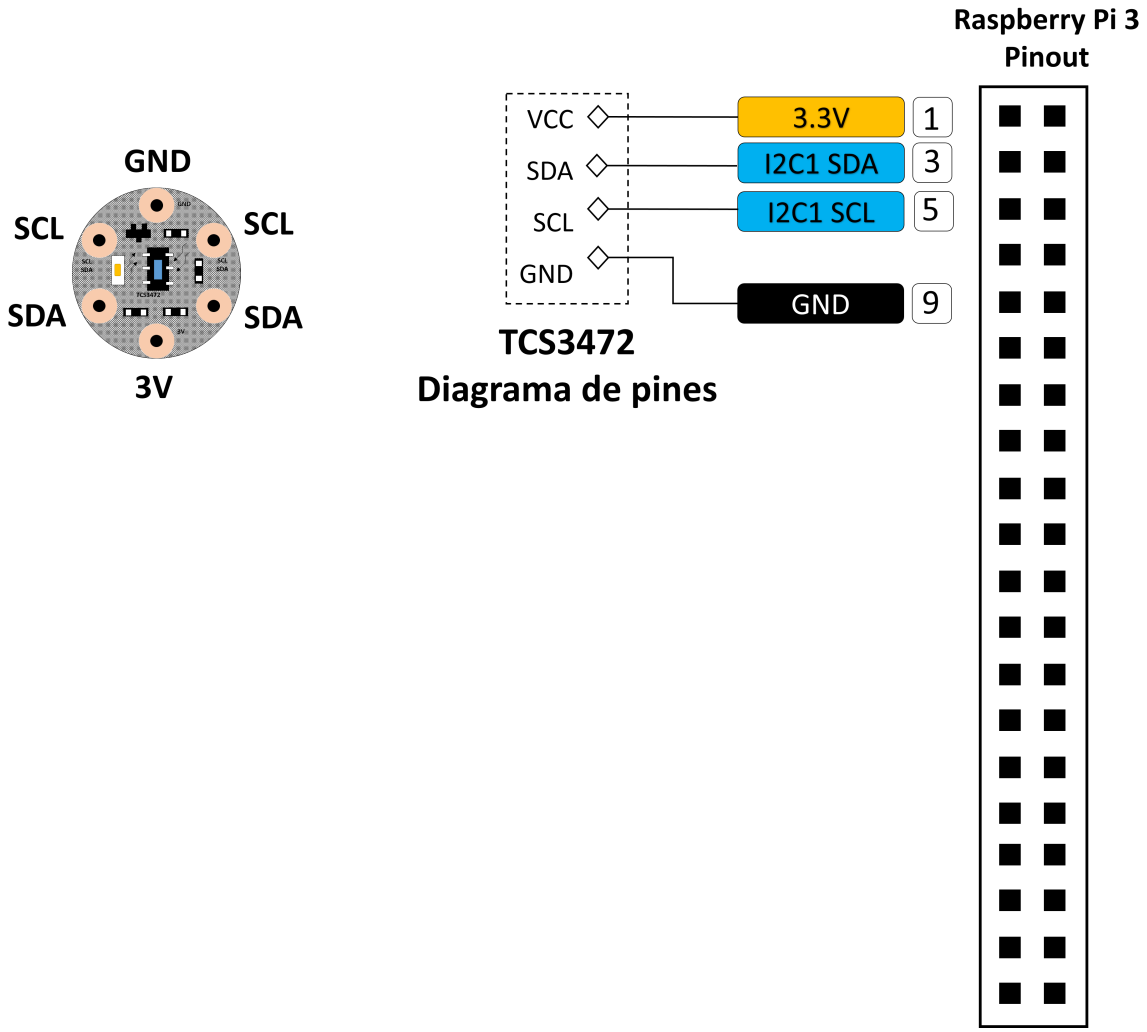

Figura 7.16: Conexionado TCS34725 Bus I2C con Raspberry Pi.

7.6 Diseño de las aplicaciones.

7.6.1 Aplicación sensor resistivo.

Máquinas de estados finitos sensor resistivo.

7.6.2 Aplicación sensor capacitivo.

Máquinas de estados finitos sensor capacitivo.

7.6.3 Aplicación sistema tangible orientada a la programación.

Máquinas de estados finitos sistema tangible.

7.7 Pruebas y validaciones de librerías y módulos.

Capítulo 8

Conclusiones

ANEXOS

Anexo A

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

5. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

6. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

7. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

8. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

9. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

10. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

Referencias

[1] J. M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

[2] S. Grover. *Expanding the Technology Curriculum to Include Foundational Elements of Computer Science for K-8*. http://www.thesmartbean.com/magazine/home-schooling/ expanding-the-technology-curriculum-foundational-elements-of-computer-science-for-k-8, 2009.

[3] J. Piaget. *The language and thought of the child*. Humanities Press, third edition, 1959.

[4] M. Resnick, F. Martin, R. Berg, R. Borovoy, V. Colella, and K. Kramer. Digital manipulatives: New toys to think. In M. E. Atwood, C. Karat, A. Lund, J. Coutaz, and J. Karat, editors, *SIGCHI conference on human factors in computing systems*. ACM Press, 1998.

[5] E. R. Kazakoff, A. A. Sullivan, and M. U. Bers. The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal*, 41(4):245–255, 2013.

[6] D. Xu. Tangible user interface for children—an overview. In *Sixth Conference in the Department of Computing*. University of Central Lancashire, 2005.

[7] L. Vygotsky. Interaction between learning and development. In M. Gauvain and M. Cole, editors, *Mind and Society*, pages 79–91. Harvard University Press, 1978.

[8] J. Quarles, S. Lampotang, I. Fischler, P. Fishwick, and B. Lok. Tangible user interfaces compensate for low spatial cognition. In *IEEE symposium on 3D user interfaces*. IEEE, 2008.

[9] J. M. Kim and M. L. Maher. Comparison of designers using a tangible user interface and a graphical user interface and the impact on spatial cognition. Master's thesis, University of Sydney, Australia, 2006. Centre of Design Computing and Cognition.

[10] B. Ploderer. Tangible user interfaces: Potentials inherent in tangible user interfaces for simplified handling of computer applications. Master's thesis, University of Applied Sciences FH JOANNEUM, Austria, 2005.

[11] L. Xie, A. N. Antle, and N. Motamedi. Are tangibles more fun? comparing children's enjoyment and engagement using physical, graphical, and tangible user interfaces. In B. C. Surrey, editor, *Second International Conference on Tangible and Embedded Interaction*. School of Interactive Arts and Technology, 2008.

[12] M. S. Horn, J. R. Crouser, and M. U. Bers. Tangible interaction and learning: the case for a hybridapproach. *Personal and Ubiquitous Computing*, 16(4):379–389, 2012.

[13] T. S. McNerney. From turtles to tangible programming bricks: explorations in physical language design. *Pers Ubiquit Comput*, 8(3):326–337, 2004.

[14] L. K. Cheng, C. S. Der, M. S. Sidhu, and R. Omar. Gui versus tui: Engagement for children with no prior computing experience. *Electronic Journal of Computer Science and Information Technology*, 3(1):31–39, 2011.

[15] A. Manches and S. Price. Designing learning representations around physical manipulation: Hands and objects. In *10th international conference on interaction design and children*. ACM Press, 1998.

[16] A. Strawhacker and M. U. Bers. I want my robot to look for food": Comparing kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces. *International Journal of Technology and Design Education*, 25(3):293–319, 2015.

[17] D. Wang, T. Wang, and Z. Liu. A tangible programming tool for children to cultivate computational thinking. *The Scientific World Journal*, page 10, 2015. art. no. 428080.

[18] J. L'Heureux, D. Boisvert, K. Sanghera, and R. Cohen. It problem solving: an implementation of computational thinking in information technology. In *Proceedings of the 13th Annual Conference on Information Technology Education (SIGITE '12)*, pages 183–188. ACM, 2012.

[19] M. Guzdial. Education: paving the way for computational thinking. *Communications of the ACM*, 51(8):25–27, 2008.

[20] I. Lee, F. Martin, and J. Denner. Computational thinking for youth in practice. *ACM Inroads*, 2(1):32–37, 2011.

[21] P. Curzon, J. Peckham, A. Settle, E. Roberts, and H. Taylor. Computational thinking(ct): on weaving it in. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '09)*, pages 201–202. ACM, 2009.

[22] T. Sapounidis, S. Demetriadis, and I. Stamelos. Evaluating children performance with graphical and tangible robot programming tools. *Personal and Ubiquitous Computing*, 19(1):225–237, 2015.

[23] J. Bailey and H. Weippert. Educational computing challenges for early childhood educators. *Australian Journal of Early Childhood*, 16(3):28–33, 1994.

[24] J. J. Beaty. *Skills for preschool teachers*. Charles E. Merrill Publishing Company, first edition, 1984.

[25] S. Papert. *Mindstorms: children, computers, and powerful ideas*. Basic Books, second edition, 1999.

[26] R. Perlman. *Using computer technology to provide a creative learning environment for preschool children*. MIT Artificial Intelligence Laboratory Publications, 1976.

[27] F. Martin and Resnick M. *LEGO/Logo and electronic bricks: creating a scienceland for children*. Advanced educational technologies for mathematics and science, first edition, 1993.

[28] H. Suzuki anda H. Kato. Algoblock: a tangible programming language, a tool for collaborative learning. In *Proceedings of the 4th European Logo conference (Eurologo'93)*, pages 297–303. Ibaraki University, 1993.

[29] T. McNerney. Tangible programming bricks: an ap- proach to making programming accessible to everyone. Master's thesis, University of Cambridge, Massachusetts, 2000. Computer Science and Artificial Intelligence Laboratory.

[30] P. Wyeth and H. Purchase. Tangible programming elements for young children. In *Extended abstracts from the proceedings of the CHI 2002 conference on human factors in computing systems*, pages 774–775. ACM Press, 2002.

[31] D. Wang, C. Zhang, and H. A. Wang. T-maze: a tangible programming tool for children. In *Proceedings of the 10th International Conference on Interaction Design and Children*. ACM, 2011.

[32] D. Wang, Y. Zhang, and T. Wang. Tanpro-kit: a tangible programming tool for children. In *Proceedings of the 12th International Conference on Interaction Design and Children*, pages 344–347. ACM, 2013.

[33] D. Wang, Y. Zhang, and S. Chen. *E-Block: A Tangible Programming Tool with Graphical Blocks*. Mathematical Problems in Engineering, first edition, 2013.

[34] M. S. Horn and R. J. K. Jacob. Designing tangible programming languages for classroom use. In *Conference: Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, page 159. ACM Press, 2007.

[35] J. C. Olabe, X. Basogain, M. A. Olabe, I. Maíz, and C. Castaño. *Education in a technological worl: communicating current and emerging research and technological efforts*. A. Méndez-Vilas, first edition, 2011.

[36] Grupo Lifelong Kindergarten. *Scratch-MIT*. https://scratch.mit.edu/, 2017.

[37] LEGO-Education. *WeDo*. http://www.legoeducation.us, 2017.

[38] Varios Autores. *MPad series*. http://www.playmore.com.hk/company/achievements, 2012.

[39] D. Merrill, E. Sun, and J. Kalanithi. Sifteo cubes. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2012.

[40] F. Scharf, T. Winkler, C. Hahn, C. Wolters, and M. Herczeg. Tangicons 3.0: an educational non-competitive collaborative game. In *IDC '12 Proceedings of the 11th International Conference on Interaction Design and Children*, pages 144–151. ACM, 2012.

[41] R. Zimmer. *Motorik und Persönlichkeitsentwicklung bei Kindern*. Hofmann Verlag GmbH, first edition, 1996.

[42] L. Zahner, U. Pühse, C. Stüssi, J. Schmid, and Dössegger A. *Aktive Kindheit – gesund durchs Leben. Handbuch für Fachpersonen*. Magglingen: BASPO, first edition, 2004.

[43] Awwwards Team. *Chrome Super Sync Sports*. http://www.awwwards.com/site-of-the-month-may-2013-chrome-super-sync-sports.html, 2007.

[44] I. Stavness, B. Lam, and S. Fels. pcubee: A perspective-corrected handheld cubic display. In *28th Annual CHI Conference on Human Factors in Computing Systems*, volume 2, pages 1381–1390. ACM, 2010.

[45] B. Shackel. Human–computer interaction: whence and whither. *American Society for Information Science*, 48(11):970–986, 1997.

[46] V. Artola. Interacción tangible en aplicaciones educativas. diseño e implementación de un prototipo basado en este paradigma de interacción orientado al aprendizaje. Master's thesis, Universidad Nacional de La Plata, Buenos Aires. Argentina, 2013.

[47] H. Ishii. Tangible user interfaces. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2006.

[48] P. Marshall, S.Price, and Y. Rogers. Conceptualising tangibles to support learning. In *Proceedings of the Interaction Design and Children (IDC'03)*. ACM Press, 2003.

[49] C. O'Malley, A. Manches, and S. Benford. The role of physical representations in solving number problems: A comparison of young children's use of physic .al and virtual materials. *Computers and Education*, 54(3):622–640, 2009.

[50] E.Upton and G.Halfacree. *Raspberry Pi User Guide*. WILEY, fourth edition, 2016.

[51] J.Rekimoto and J.SmartSkin. An infrastructure for freehand manipulation on interactive surfaces. In *CHI '02*, pages 113–120. ACM, 2002.

[52] G.Barret and R.Omote. Projected-capacitive touch technology. In *Information Display*, volume 26, pages 16–21. ACM, 2100.

[53] Silicon Labs. *How to Minimize Touchscreen Electromagnetic Interference*. http://pages.silabs.com/LP-minimizing-touchscreen-emi.html, 2011.

[54] T.G. Zimmerman, J.R. Smith, J.A. Paradiso, D. Allpor, and N. Gershenfeld. Applying electric field sensing to human-computer interfaces. In *Proc. CHI '95*, pages 280–287. ACM, 1995.

[55] Waveshare Electronics. *Capacitive display LCD 7inch HDMI(B)*. https://www.waveshare.com, 2015.

[56] R.Downs. Using resistive touch screens for human/maquine interface. *Analog Applications Journal*, 2005.

[57] R. Payás. *Sensores y acondicionadores de señal*. Marcombo, fourth edition, 2003.

[58] InvenSense. *MPU-9250 Nine-Axis (Gyro + Accelerometer + Compass) MEMS Motion-Tracking Device*. https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/, 2015.

[59] Raspberry Pi Foundation. *Raspberry Pi - Teach, Learn, and Make with Raspberry Pi*. https://www.raspberrypi.org, 2018.

[60] Raspberry Pi Foundation. *Installing operating system images*. https://www.raspberrypi.org/downloads/noobs/, 2018.

[61] The Python Software Foundation. *Python Web*. https://www.python.org, 2018.

[62] E.Bahit. *Curso: Python para Principiantes*. http://www.safecreative.org/work/1207302042960-curso-python-para-principiantes, 2012.

[63] David Goodger. *Index of Python Enhancement Proposals*. https://www.python.org/dev/peps/, 2018.

[64] D. Phillips. *Creating Apps in Kivy*. Oreilly, first edition, 2014.

[65] JetBrains. *PyCharm*. https://www.jetbrains.com/pycharm/, 2018.

Este documento fue editado y tipografiado con LaTeX empleando
una versión modificada de la clase **esi-tfg** (versión 0.20180726)
que se puede encontrar en:
https://bitbucket.org/arco_group/esi-tfg

[respeta esta atribución al autor]