UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE CUENCA

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

SISTEMA LICORNIO TECH: UNA HERRAMIENTA TECNOLÓGICA PARA EL SOPORTE A LA COMUNICACIÓN BASADO EN LICORNIO Y PICTOGRAMAS PARA NIÑOS CON IMOC

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica

AUTORES: WALTER MARCELO ARIAS NIEVES
BORIS FERNANDO MOROCHO YARI

TUTOR: ING. ANGEL FERNANDO SOTO SARANGO, Mg. T

Cuenca – Ecuador
2024

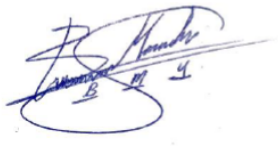
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Walter Marcelo Arias Nieves con documento de identificación N° 0106085897 y Boris Fernando Morocho Yari con documento de identificación N° 0150107613; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 18 de marzo del 2024

Atentamente,



Walter Marcelo Arias Nieves

0106085897

Boris Fernando Morocho Yari

0150107613

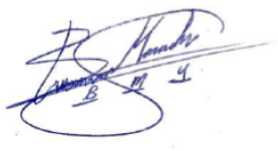
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Walter Marcelo Arias Nieves con documento de identificación N° 0106085897 y Boris Fernando Morocho Yari con documento de identificación N° 0150107613, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: "Sistema licornio TECH: una herramienta tecnológica para el soporte a la comunicación basado en licornio y pictogramas para niños con IMOC", el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 18 de marzo del 2024

Atentamente,



Walter Marcelo Arias Nieves

0106085897

Boris Fernando Morocho Yari

0150107613

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Angel Fernando Soto Sarango con documento de identificación N° 1103455802, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: SISTEMA LICORNIO TECH: UNA HERRAMIENTA TECNOLÓGICA PARA EL SOPORTE A LA COMUNICACIÓN BASADO EN LICORNIO Y PICTOGRAMAS PARA NIÑOS CON IMOC, realizado por Walter Marcelo Arias Nieves con documento de identificación N° 0106085897 y Boris Fernando Morocho Yari con documento de identificación N° 0150107613, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 18 de marzo del 2024

Atentamente,

Ing. Angel Fernando Soto Sarango, Mgst.

1103455802

Índice general

Índice de figuras

Índice de tablas

Resumen

En el presente proyecto desarrollamos una herramienta tecnológica para el soporte a la comunicación y la interacción de niños con Insuficiencia Motora de Origen Cerebral (IMOC). Para ello, se diseñó y construyó un dispositivo electrónico capaz de capturar datos de movimientos de la cabeza, los cuales son enviados mediante bluetooth a una aplicación móvil desarrollada en Android Studio. La aplicación móvil, por su parte, presenta una interfaz intuitiva y amigable basada en pictogramas del sistema ARASAAC. Estos pictogramas se organizan en categorías y subcategorías para facilitar la navegación y la selección por parte del usuario.

La aplicación procesa los datos de movimiento recibidos del dispositivo electrónico para controlar un cursor virtual en la pantalla del dispositivo móvil. De esta manera, los niños con IMOC pueden utilizar los movimientos de la cabeza para seleccionar y comunicarse a través de los pictogramas disponibles en la interfaz. Este enfoque ofrece una alternativa accesible y efectiva para aquellos niños con dificultades motoras que limitan su capacidad para comunicarse de manera convencional.

Finalmente, se realizaron pruebas exhaustivas para asegurar que el dispositivo se ejecute de manera precisa, con el fin de identificar y resolver cualquier problema, como retrasos en la respuesta o falta de precisión para seleccionar los pictogramas. Cabe aclarar que se han tomado consideraciones de las características de interacción que tienen los niños con IMOC para diseñar el sistema, sin embargo, en este proyecto solo se llegó a desarrollar la tecnología y a evaluar que se pueda seleccionar pictogramas mediante movimientos de la cabeza.

Palabras clave: IMOC; ARASAAC; Android Studio; Comunicación; Pictogramas.

Abstract

In this project we developed a technological tool to support the communication and interaction of children with Motor Impairment of Cerebral Origin (IMOC). For this purpose, we designed and built an electronic device capable of capturing head movement data, which are sent via Bluetooth to a mobile application developed in Android Studio. The mobile application, in turn, presents an intuitive and user-friendly interface based on pictograms from the ARASAAC system. These pictograms are organized into categories and subcategories to facilitate navigation and selection by the user.

The application processes motion data received from the electronic device to control a virtual cursor on the mobile device's screen. In this way, children with IMOC can use head movements to select and communicate through the pictograms available on the interface. This approach offers an accessible and effective alternative for children with motor difficulties that limit their ability to communicate conventionally.

Finally, extensive testing was performed to ensure that the device performs accurately and to identify and resolve any problems, such as delays in response or lack of accuracy in selecting pictograms. It should be noted that the interaction characteristics of children with IMOC have been taken into consideration when designing the system, however, in this project, we only went as far as developing the technology and evaluating the ability to select pictograms through head movements.

Keywords: IMOC; ARASAAC; Android Studio; Communication; Pictograms.

Antecedentes

Los niños con Insuficiencia Motora de Origen Cerebral (IMOC) enfrentan grandes desafíos en la comunicación debido a la parálisis en sus extremidades. Esto puede dificultar el movimiento y la función muscular como es el habla, la escritura y gestos, haciendo que dependan de dispositivos especiales para comunicarse, lo que a su vez puede influir significativamente en la vida de quien la padece, así como la de sus familias. En Ecuador los niños con discapacidad de grado IV es del 30.67 por ciento del total de niños registrados en el registro nacional de discapacidad (26.062 niños) con rango de edades entre 4 y 12 años con discapacidad [1].

La interacción humano-computadora es un proceso mediante el cual un individuo se relaciona con una computadora o con dispositivos que incorporan dichas computadoras [2]. Con el paso del tiempo, se han producido notables avances en la tecnología de interacción humano-computadora, lo cual, ha contribuido a desarrollar una mejor calidad de vida y la habilidad de comunicación de personas con IMOC. Estos avances continúan evolucionando y mejorando, lo que permite a las personas con IMOC acceder a un mundo digital de maneras que antes eran impensables.

En la actualidad, existen diversas tecnologías utilizadas para la interacción humano-computadora, algunas de ellas por ejemplo: pantallas táctiles, ratones, teclados, comunicadores de voz, dispositivos de asistencia, seguimiento de movimientos, entre otras. Sin embargo, gran parte de estas tecnologías no están diseñadas para personas con IMOC, lo que genera grandes desafíos en el uso de estas tecnologías, desde dificultades de accesibilidad y limitaciones motoras de comunicación, con costos de dispositivos muy elevados. Abordar estos problemas requiere un enfoque en el diseño de tecnologías de bajo costos que sean accesibles, un avance constante en la investigación y desarrollo de tecnologías de asistencia.

Por otra parte, existe un dispositivo que comúnmente se le llama sistema licornio, este es un dispositivo situado a modo de corona en la cabeza de los niños, que al mover tratara de apuntar hacia una imagen u objeto que requiera en ese momento y le permita interactuar con los demás. Normalmente este tipo de dispositivo es mecánico, dado que cuenta con un mecanismo de fijación y una barra en forma de curva con la que se puede realizar diversas tareas. Esto hace que sea muy difícil para los niños con IMOC realizar el control de diferentes elementos mediante la cabeza. Por tal motivo, en este proyecto se busca solventar este problema del sistema licornio tradicional, para esto se diseñará un dispositivo electrónico, el cual podrá ser utilizado para dirigir el cursor de una computadora a través de los movimientos de la cabeza. Además, se diseñará un interfaz de aplicación el cual contendrá pictogramas con los que los niños podrán interactuar con sus cuidadores o familias.

Justificación

El uso de tecnologías de interacción humano-computadora en personas con IMOC es importante porque les proporciona medios efectivos de comunicación, promueve su independencia, facilita el acceso a la educación, y contribuye a la inclusión social, lo cual posibilita la mejora de sus condiciones de vida. Por tal motivo, en diversas regiones del mundo se han desarrollado dispositivos de asistencia para personas con discapacidades, tales como:

Dispositivo de Visión por Computadora

Este dispositivo de visión por computadora posibilita la detección del habla, el reconocimiento de imágenes, el monitoreo de la atención, el análisis de expresiones faciales y el reconocimiento de emociones. La identificación de rostros desempeña un papel clave en la comprensión del contenido audiovisual [3].

Dispositivo de Teclado Virtual

Este teclado virtual puede ajustarse automáticamente para minimizar la distancia que se debe recorrer visualmente y el tiempo necesario para hacer selecciones, brindando una mayor asistencia a los usuarios con discapacidad motora y del habla grave (SSMI) sin necesidad de la intervención de un asistente [4].

Software Interactivo

El Software Interactivo utiliza un Sistema de Electromiografía Superficial que captura las señales eléctricas correspondientes a tres movimientos esenciales de la mano, habilitando de esta manera la gestión del cursor y la exploración en la

aplicación informática [5].

Sin embargo, estas tecnologías pueden ser difíciles de encontrar en el mercado convencional, lo que limita el acceso a dichas tecnologías. Además, de que estas tecnologías también pueden ser muy costosas y requiere grandes recursos económicos, lo que imposibilita que las familias con escasos recursos lo adquieran. Por tal motivo en este trabajo se busca diseñar una herramienta tecnológica para el soporte a la comunicación basado en licornio y pictogramas para niños con IMOC, la cual sea barata y de fácil acceso. Para lograr esto, se utilizará dispositivos electrónicos baratos, para así no incurrir en costos altos para que más niños puedan beneficiarse de esta tecnología. Además, de hacer que su facilidad de uso sea más accesible tanto para los niños como para sus cuidadores, lo que puede acelerar el proceso de aprendizaje y aumentar la eficacia de la comunicación. Esto, a su vez, puede permitir a los niños expresarse y relacionarse de manera más efectiva, fortaleciendo sus habilidades de comunicación y su interacción social, y posiblemente contribuyendo a un mayor desarrollo cognitivo y emocional en el proceso.

Objetivos

Objetivo General

- Diseñar e implementar una herramienta tecnológica para el soporte a la comunicación basado en licornio y pictogramas para niños con IMOC.

Objetivos específicos:

- Realizar un estado del arte de las herramientas tecnológicas disponibles para la interacción humano-computadora direccionada a niños con limitaciones motoras.
- Diseñar e implementar una interfaz electrónica para el control del cursor de una computadora basado en los movimientos de la cabeza.
- Diseñar e implementar un software basado en pictogramas de ARASAAC que abarca a licornio para la interacción con la interfaz electrónica.
- Realizar una evaluación preliminar sobre la capacidad del sistema para poder seleccionar pictogramas basados en movimientos de la cabeza.

Introducción

La Insuficiencia Motora de Origen Cerebral (IMOC) es una condición médica que afecta el movimiento y la función muscular debido a daños o anomalías en el cerebro en etapas tempranas de desarrollo. Para algunos niños, esto se convierte en un desafío debido a las limitaciones motoras que dificultan o impiden el uso de métodos convencionales de expresión verbal o escrita. Dentro de este marco, la tecnología de asistencia surge como un instrumento prometedor que ayuda a fomentar la inclusión social de estos niños, ofreciendo soluciones adaptadas a sus necesidades específicas.

El objetivo principal de este proyecto es diseñar e implementar una herramienta tecnológica que permita a los niños con IMOC comunicarse de manera más eficiente y autónoma, utilizando movimientos de la cabeza como medio de interacción. Con el fin de alcanzar este objetivo, se diseña un dispositivo electrónico capaz de recopilar datos de movimiento de la cabeza, junto con un software de aplicación desarrollado en Android Studio, que presenta interfaces interactivas con pictogramas de ARASAAC, un sistema de símbolos pictográficos utilizado en la comunicación aumentativa y alternativa (CAA).

A lo largo de este proyecto, se detallarán los procesos de diseño, implementación y efectividad del sistema propuesto. Además, se presentarán los resultados obtenidos de las pruebas realizadas en los niños para validar la efectividad del dispositivo.

Por lo tanto, con el desarrollo de esta herramienta tecnológica, se busca que los niños con IMOC puedan establecer una comunicación de una forma más sencilla y apropiada, ya sea en su entorno familiar, educativo o social.

Capítulo 1

Estado del Arte

En el capítulo actual se da a conocer los conocimientos teóricos fundamentales necesarios para el desarrollo del dispositivo, tanto en hardware y software, lo cual mediante una breve explicación, análisis comparativos de los diferentes software y hardware que se presentan en el mercado, los que hemos elegido para nuestro proyecto basado en las ventajas que más adelante se detallan, que garantice que el proyecto llegue a una culminación en el tiempo establecido, ayudando a mejorar la accesibilidad y la comunicación para las personas con insuficiencia motora.

1.1. Sistema Licornio

Es el diseño y la construcción de un comunicador electrónico para niños con parálisis cerebral, su función es brindar a los niños con déficit en el habla y la escritura, la posibilidad de comunicarse mediante movimientos de la cabeza, y así disminuir el grado de aislamiento que esta patología puede generar [6]. Los aspectos requeridos para el desarrollo del comunicador se basaron en la búsqueda de información. No sólo sirve como una nueva herramienta para la comunicación del usuario, si no que ayuda a la concentración.

1.2. Pictogramas de Arasaac

Son una variedad de representaciones visuales, que incluyen símbolos en blanco, negro y a color, con un estilo simple y uniforme, Las técnicas y estrategias que amplían y sustituyen la comunicación en personas con impedimentos, conocidas como Sistemas Aumentativos y Alternativos de Comunicación (SAAC) abarcan un amplio vocabulario destinado a favorecer la comunicación de personas con distintos tipos de discapacidades [7]. Estos sistemas se han convertido en un mecanismo esencial para permitirles interactuar y tener un impacto en su entorno inmediato, al mismo tiempo que participan en el mundo que los rodea.

La cualidad de los pictogramas que ha sido objeto de mayor atención en este contexto es el nivel de referencialidad de una imagen, es decir, la correspondencia visual entre la imagen misma y el objeto al que se refiere.

1.2.1. Sistemas de Símbolos

Los conjuntos de símbolos empleados en la Comunicación Aumentativa y Alternativa CAA se clasifican en gestuales y gráficos, ambos con una escala que abarca desde opciones simples adecuadas para personas con diversas limitaciones cognitivas y lingüísticas, hasta sistemas más complejos que permiten un nivel avanzado de lenguaje utilizando signos manuales o gráficos [8].

En el caso de los símbolos gestuales, esta escala va desde gestos y mímica de uso cotidiano hasta la utilización de signos manuales, siguiendo típicamente el orden del lenguaje hablado, lo que se conoce como lenguaje con signos o bimodal.

1.2.2. Procesamiento de Datos

Este procedimiento busca hacer accesibles las funciones de control, supervisión, recopilación de datos, análisis estadístico, diagnóstico de fallas y otras actividades relacionadas a un público más amplio. Esto no se limita únicamente a los operadores de las máquinas y del proceso, sino que también involucra al personal que participe en la producción. Esta apertura posibilita que todos puedan observar, comprender, analizar y tomar decisiones pertinentes con el fin de corregir y mejorar las operaciones

[9].

1.3. Principios de los Sistemas de Interacción Humano-Máquina

Usabilidad: Para este método tenemos que tomar en consideración los siguientes parámetros: facilidad de uso (La interfaz debe ser fácil de aprender y utilizar), eficiencia (Los usuarios deben poder realizar tareas de forma rápida y precisa). Por último, satisfacción del usuario (Los usuarios deben sentirse satisfechos con la experiencia de uso).

Accesibilidad: En este caso los usuarios deben tener planteados los siguientes puntos: Diseño Inclusivo (Asegurar que la interfaz sea accesible para personas con diferentes habilidades y discapacidades) y cumplimiento de estándares (seguir pautas y estándares de accesibilidad para garantizar un uso equitativo).

Feedback e Iteración: Para este caso el creador debe tener en cuenta dos aspectos los cuales son: Retroalimentación Inmediata (Proporcionar respuestas claras y rápidas a las acciones del usuario) y la Iteración del Diseño (Mejorar continuamente la interfaz a través de pruebas y ajustes basados en la retroalimentación del usuario).

Adaptabilidad: El creador de programa debe dar las distintas facilidades, en este caso hay q tomar en cuenta dos aspectos: Personalización (Permitir que los usuarios personalicen la interfaz según sus 5 preferencias y necesidades) y la contextualización (Adaptar la interfaz a diferentes contextos y situaciones de uso).

Diseño Centrado en el Usuario: Por último, en esta parte se debe tomar en cuenta algunos aspectos: Investigación del Usuario (Entender las necesidades, expectativas y comportamientos del usuario) y un prototipado Iterativo (Crear versiones tempranas de la interfaz para obtener retroalimentación temprana de los usuarios).

1.4. Sistemas Operativos

Un sistema operativo es una pieza fundamental de software que funciona como un intermediario entre el hardware de un dispositivo y las aplicaciones o programas que se ejecutan en él. Es la capa esencial que administra los recursos del sistema, ofrece servicios a los programas de aplicación y facilita la comunicación del usuario con el hardware.

En la Tabla 1.1, explicaremos las caracteristicas de los principales sistemas operativos.

Tabla 1.1: Comparación de las distintivas características de los sistemas operativos. [10]

Sistemas Operativos	Características
Windows	Ofrece capacidad de realizar múltiples tareas simultáneamente. presenta una interfaz gráfica para el usuario. Es ampliamente utilizado debido a su compatibilidad con una variedad de dispositivos.
MacOS	Facilita la ejecución de varias tareas al mismo tiempo Está construido sobre la base del sistema Unix. Se destaca por su alto nivel de seguridad y capacidad para admitir múltiples usuarios.
Linux	Multiusuario. Sistema basado en Unix. Nivel de seguridad muy alto.
Android	Opera sobre el núcleo de Linux como su base. Conocido por su estabilidad alto nivel de seguridad Posee una variedad de aplicaciones que están diseñadas para este sistema operativo.
iOS	Exclusivamente para los dispositivos iPhone. Destacada por su estabilidad y diseño atractivo Rendimiento óptimo en aplicaciones Además cuentan con una tienda de aplicaciones exclusiva.

En la Tabla 1.2, explicacion de ventajas y desventajas de los principales sistemas operativos.

Como podemos observar existen varias diferencias entre los sistemas operativos en los cuales podemos programar y básicamente depende del programar elegir el adecuado, en nuestro proyecto utilizamos el sistema Android con el cual consta la mayoría de las personas hasta la última actualización en enero de 2022, Android es el sistema operativo utilizado con mayor frecuencia en tabletas, en términos de cuota de mercado. Android es utilizado por una variedad de fabricantes de tabletas, lo que contribuye a su predominio en el mercado.

Tabla 1.2: Cuadro comparativo de ventajas y desventajas de los sistemas operativos [10]

	Ventajas y Desventajas
Windows	**Ventajas:** Abundante información disponible para resolver problemas. **Desventajas:** Vulnerabilidad a virus, troyanos y gusanos que estan diseñados para atacarlo. Retrasos significativos en la disponibilidad de parches de seguridad.
MacOS	**Ventajas:** Inmunidad ante virus . alta estabilidad y su diseño altamente elogiado. Respaldado por la marca. **Desventajas:** Limitado a computadoras de la marca Apple. Costo elevado de las aplicaciones en comparación con otros sistemas operativos.
Linux	**Ventajas:** Inmunidad ante virus. Actualización rápida de parches de seguridad. Alta estabilidad y seguridad. Gratis y de distribución libre. **Desventajas:** Limitaciones en el soporte para tarjetas gráficas avanzadas y juegos. Diversidad de versiones que pueden causar confusión al usuario.
Android	**Ventajas:** Amplio soporte por parte de varias compañías de teléfonos móviles. Variedad extensa de aplicaciones, muchas de ellas gratuitas. Inmunidad ante virus. **Desventajas:** Riesgos de seguridad al instalar aplicaciones desde fuentes externas a la tienda oficial. Posibilidad de encontrar aplicaciones no deseadas con troyanos o malware.
iOS	**Ventajas:** Actualizaciones regulares para mejorar la seguridad. Inmunidad ante virus. Entorno optimizado para un rendimiento óptimo del dispositivo. **Desventajas:** Entorno optimizado para un rendimiento óptimo del dispositivo. Posibles problemas de rendimiento de la batería. Costo elevado de la mayoría de las aplicaciones.

1.5. Herramientas de desarrollo para apps móviles

Para el desarrollo de apps móviles se hizo la búsqueda de los lenguajes de programación más frecuentes que tengan mayor facilidad de programar. Existen diversos entornos de desarrollo y programas que pudríamos utilizar para programar aplicaciones móviles.

En la Tabla 1.3, presentamos algunos de los entornos y herramientas más populares para la creación de las diferentes aplicaciones móviles.

Como podemos observar en la Tabla 1.3 es importante destacar que la elección del lenguaje y framework dependerá de varios factores, como la preferencia del desarrollador, la naturaleza del proyecto, el rendimiento deseado y los requisitos

Tabla 1.3: Cuadro comparativo de plataformas de desarrollo [Autor]

Herramienta	Plataformas soportadas	Lenguajes	Características
Android Studio	Android	Kotlin, Java	Plataforma de desarrollo oficial para Android.
Xcode	iOS, macOS	Swift, Objective-C	Plataforma de desarrollo oficial para iOS y macOS.
Visual Studio (Xamarin)	iOS, Android, Windows	C++	Desarrollo multiplataforma con Xamarin.
Flutter	iOS, Android, Web	Dart	Desarrollo multiplataforma, framework de interfaz de usuario.
React Native	iOS, Android	JavaScript (React)	Desarrollo multiplataforma basado en React.
App Inventor	Android	Bloques visuales	Plataforma visual, fácil de aprender, creada por MIT.

específicos de la plataforma. Además, las tecnologías evolucionan, por lo que es fundamental estar al tanto de las tendencias y cambios en el panorama del desarrollo móvil. Por esta razón y estudiando los diferentes lenguajes de programación optamos por Android Studio la cual el 80 por ciento de las aplicaciones son creadas por esta aplicación.

Para un mejor estudio presentamos las diferentes ventajas y desventajas de las plataformas que ayudan al desarrollo, como se muestra en la Tabla 1.4.

La elección del entorno de desarrollo depende de los objetivos, habilidades y las necesidades específicas del proyecto ya que cada una tiene sus propias fortalezas y debilidades, en cuanto a todo el estudio realizado de los diferentes plataformas de desarrollo para aplicaciones móviles la mejor opción fue Android Studio.

A continuación, en la Tabla 1.5 presentamos las razones por las cuales se eligio trabajar con Android Studio.

Lenguajes de programación utilizados en Android Studio

Java y Kotlin son dos idiomas de programación que disfrutan de una gran popularidad para el creación de aplicaciones, especialmente en el ámbito de las aplicaciones Android. A continuación, proporcionaremos una breve descripción de cada uno, ya que serán los lenguajes que utilizaremos.

Java: Es un idioma de codificación utilizado en el desarrollo de aplicaciones móviles, siendo Android un caso relevante de su utilización. Destacado por su velocidad, facilidad de uso y versatilidad, Java es una opción ideal tanto para aplicaciones móviles como para software personalizado. Ejemplos de su aplicación incluyen

Tabla 1.4: Ventajas y desventajas de los diferentes plataformas de desarrollo para aplicaciones móviles [Autor]

Herramienta	
Android Studio	**Ventajas** Oficialmente respaldado por Google para el desarrollo de aplicaciones Android. Amplia comunidad de desarrolladores y recursos de soporte. Integración con otras herramientas y servicios de Google. **Desventajas** Enfoque principalmente en el desarrollo para Android, no compatible con iOS.
Xcode	**Ventajas** Entorno de desarrollo oficial para iOS y macOS. Herramientas robustas para pruebas y depuración. Integración con servicios de Apple. **Desventajas** Limitado al desarrollo para plataformas de Apple (iOS y macOS). Requiere hardware de Apple para ejecutarse.
Visual Studio (Xamarin)	**Ventajas** Desarrollo multiplataforma para iOS, Android y Windows. Utiliza el lenguaje C++. Integración con el ecosistema .NET. **Desventajas** Algunas características pueden requerir licencias de pago. La adopción de nuevas actualizaciones puede llevar tiempo.
Flutter	**Ventajas** Desarrollo multiplataforma. Interfaz de usuario atractiva y personalizable. Hot Reload para desarrollo rápido **Desventajas** Menor cantidad de bibliotecas y módulos comparado con otras opciones. La comunidad puede ser más pequeña en comparación con tecnologías más establecidas.
React Native	**Ventajas** Desarrollo multiplataforma basado en JavaScript. Uso de componentes nativos para un rendimiento óptimo. Gran comunidad y amplia adopción. **Desventajas** Posible necesidad de módulos nativos para ciertas funcionalidades. Problemas de rendimiento en aplicaciones complejas.
App Inventor	**Ventajas** Plataforma visual, fácil de aprender y usar. Enfoque en la simplicidad para principiantes y educación. **Desventajas** Restringido en cuanto a la complejidad y funciones avanzadas. Inapropiado para proyectos profesionales o de gran complejidad.

Twitter, Netflix y Uber, entre otros [11].

Kotlin: Kotlin es considerado dentro de los principales lenguajes de programación utilizados en dispositivos móviles Android debido a su código intuitivo, sencillo y eficaz. A pesar de que fue inicialmente diseñado para complementar a Java, en los últimos años, Google lo ha destacado como su elección prioritaria para la creación de aplicaciones en la plataforma Android. [11].

Tabla 1.5: Descripciones generales de Android Studio [Autor]

Características	Descripción
IDE Oficial	Android Studio, respaldado oficialmente por Google, es el ambiente de desarrollo favorito para la creación de aplicaciones destinadas a Android.
Soporte para Kotlin	Ofrece un fuerte soporte para Kotlin, el lenguaje de programación oficial para Android.
Emulador Integrado	Ofrece una integración directa con Firebase, facilitando el desarrollo de aplicaciones con servicios en la nube.
Compatibilidad con Jetpack	Integra nativamente el conjunto de bibliotecas Jetpack, mejorando la eficiencia y calidad del desarrollo.
Análisis de Rendimiento	Proporciona herramientas avanzadas para el análisis de rendimiento y detección de problemas.
Totalmente Gratuito	Android Studio, al ser gratuito y de libre acceso al código fuente, está al alcance de los desarrolladores, lo que lo hace accesible para su uso y desarrollo.

1.6. Arduino

Es una plataforma de electrónica de código abierto que comprende tanto hardware como software. Es ampliamente empleado para la creación de prototipos y proyectos electrónicos interactivos. Esta plataforma ofrece una manera sencilla y adaptable para que aficionados, estudiantes y profesionales desarrollen y experimenten con dispositivos electrónicos.

En la Figura 1.1, se observa algunas características y diseños físicos de las placas Arduino más importantes.

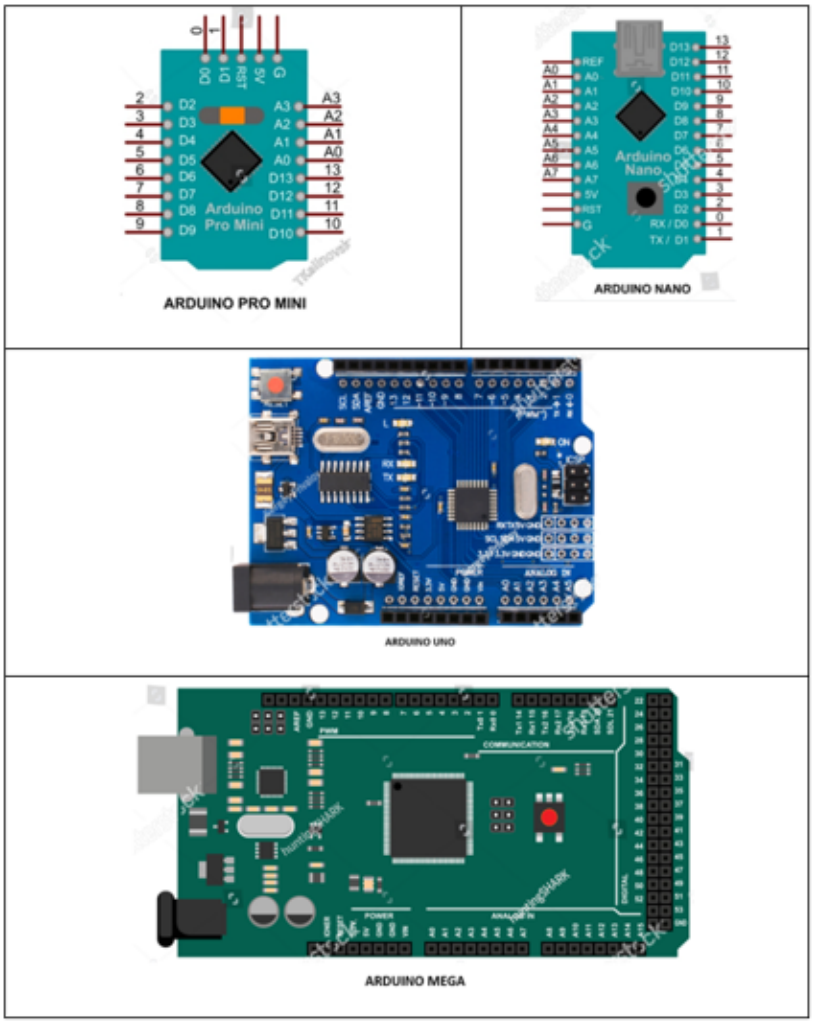


Figura 1.1: Diferentes tipos de Arduino [12]

A continuación, en la Tabla 1.6, hacemos una comparación de las diferentes características técnias de las placas Arduino para así elegir la mas adecuada para las necesidad del proyecto.

Teniendo encuenta los diferentes aspectos en la Tabla 1.6, expuesta anteriormente Arduino Pro Mini es una excelente elección cuando el espacio es un factor crítico y se necesita una solución compacta y versátil. Su tamaño reducido no impide su capacidad para manejar una variedad de proyectos, especialmente aquellos que se centran en la portabilidad y la integración en espacios limitados.

Tabla 1.6: Características técnicas de las placas Arduino [13]

Caraterísticas	Arduino Uno	Arduino Nano	Arduino Mega	Arduino Pro Mini
Conectividad USB	Sí	Sí	Sí	Requiere adaptador externo
Pines E/S Digitales	14	14	54	14
Pines de Entrada Analógica	6	8	16	6
Memoria Flash	32KB	32KB	256KB	16KB
Memoria SRAM	2KB	2KB	8KB	1KB
Velocidad del Procesador	16MHz	16MHz	16MHz	8MHz
Alimentación	5VDC	5VDC	5VDC	3.3V o 5VDC
Corriente Máxima (por pin)	20mA	40mA	40mA	40mA
Corriente Máxima (3.3V pin)	50mA	50mA	50mA	150mA
Comunicación Serial	Sí (UART)	Sí (UART)	Sí (UART)	Sí (UART)
I2C y SPI	Sí	Sí	Sí	Sí
PWM Pines	6	6	14	6
Precio Relativo	Moderado	Moderado	Mayor	Moderado
Dimensiones Típicas	68.6 x 53.4 mm	18.5 x 43.2 mm	101.6 x 53.4 mm	Variable

1.7. Módulo Bluetooth

En el contexto de Arduino, un módulo Bluetooth hace referencia a un dispositivo que emplea la tecnología Bluetooth para posibilitar la comunicación inalámbrica entre un Arduino y otros dispositivos, como teléfonos móviles, tabletas o computadoras. Estos módulos Bluetooth simplifican la creación de proyectos que demandan la transmisión de datos de forma inalámbrica.

Un módulo Bluetooth típicamente consta de un hardware y un conjunto de comandos AT (comandos de atención) que permiten la configuración y la comunicación con el módulo a través de un microcontrolador, como el que se encuentra en una placa Arduino. Algunos de los módulos Bluetooth comunes utilizados con Arduino incluyen el HC-05 y el HC-06.

Aspectos que se deben considerar al utilizar un módulo Bluetooth con Arduino.

Conexión Física: Comúnmente, la conexión del módulo Bluetooth a la placa Arduino implica conectar los pines de alimentación y tierra, así como los pines RX (receptor) y TX (transmisor) del módulo Bluetooth a los pines correspondientes de la placa Arduino.

Configuración del Módulo Bluetooth: Se utiliza los comandos AT (comandos de atención) para configurar el modo de funcionamiento. Esto se puede hacer mediante un adaptador serie conectado a la placa Arduino y a través del monitor serie en el IDE

de Arduino.

Programación: El IDE del Arduino permite la comunicación con el módulo Bluetooth. Se puede utilizar la biblioteca SoftwareSerial para crear una comunicación serial virtual en pines determinados de la placa Arduino.

Comunicación Inalámbrica: El módulo Bluetooth posibilita la transmisión y recepción de datos de manera inalámbrica entre el Arduino y otros dispositivos Bluetooth.

1.8. Módulo MPU6050

El módulo MPU6050 es un sensor de seis ejes que integra un un chip que integra tanto un acelerómetro como un giroscopio en una sola unidad. Este tipo de sensor es frecuentemente empleado en proyectos electrónicos, especialmente en aplicaciones que requieren medición de movimiento, orientación y control de dispositivos.

En la Tabla 1.7, se expone algunos aspectos clave sobre el módulo MPU6050.

Tabla 1.7: Descripción del módulo MPU6050 [Autor]

Características	Descripción
Integración de Acelerómetro y Giroscopio	Combina ambas funciones en un solo chip, Lo que simplifica el diseño y ahorra espacio.
Comunicación Digital	Utiliza un bus I2C para la comunicación con el microcontrolador, Lo que facilita la integración con una amplia variedad de plataformas.
Precisión y Resolución	Ofrece una buena precisión y resolución para muchas aplicaciones, Lo que lo hace adecuado para tareas que requieren mediciones precisas de movimiento.
Fácil Implementación	Proporciona una interfaz simple y fácil de implementar, Esta característica lo convierte en una elección muy popular para proyectos de bricolaje y prototipos.
Bajo Consumo de Energía	Diseñado para operar con un bajo consumo de energía, Lo que lo hace adecuado para dispositivos alimentados por batería.
Amplia Aplicación	Adecuado para una variedad de aplicaciones, como sistemas de navegación inercial, Monitoreo de actividad física, control de movimiento en robots, entre otros.

Como observamos en la Tabla 1.7, es crucial tener en cuenta que, si bien el MPU6050 es una opción ampliamente elegida, existen otros sensores de movimiento disponibles en el mercado, cada uno con sus propias características y beneficios. La selección del sensor adecuado dependerá de los requisitos particulares del proyecto.

Ejemplos de Aplicación: El MPU6050 se emplea en una diversidad de aplicaciones, como sistemas de estabilización para drones, robots, juguetes y dispositivos de realidad virtual. Puede proporcionar información sobre la inclinación, la orientación

y los movimientos en tiempo real. Para utilizar el módulo MPU6050 con Arduino, primero debes conectarlo físicamente a la placa Arduino. Por lo general, el módulo tiene pines etiquetados como VCC (alimentación), GND (tierra), SDA (línea de datos I2C), y SCL (línea de reloj I2C).

Capítulo 2

Diseño del Dispositivo Electrónico y Software de Aplicación

En el presente capítulo vamos a describir el proceso de diseño de un módulo tipo mouse. De igual manera, se detallará el diseño del software de aplicación, enfocándose en la creación de una interfaz de usuario intuitiva.

Esta aplicación estará diseñada para que se comunique con el dispositivo electrónico y así poder controlar el cursor en la pantalla de la aplicación mediante movimientos de la cabeza, utilizando las herramientas descritas en el capítulo anterior. Adicionalmente, se dará una breve explicación de la arquitectura del sistema propuesto.

2.1. Diseño del Dispositivo Electrónico

A continuación, se detalla el diseño del dispositivo electrónico.

2.1.1. Conexión del Circuito

En la Figura 2.1, se muestra el esquema de conexión de los diferentes elementos electrónicos.

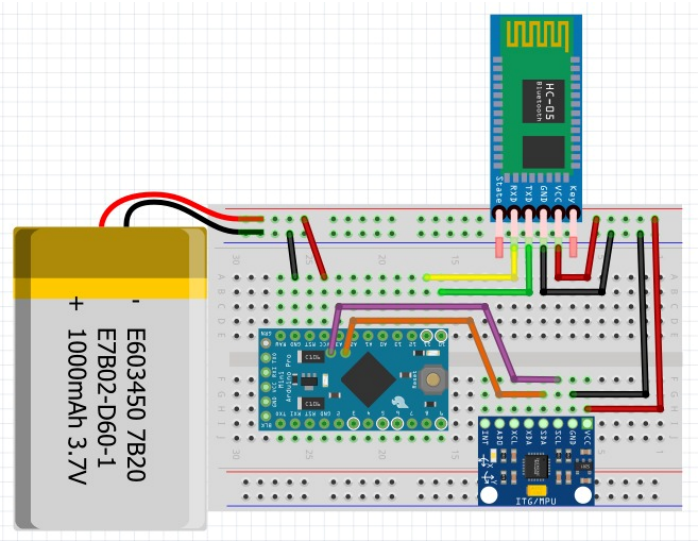


Figura 2.1: Esquema de conexión de los dispositivos [Autor]

Para conectar un módulo Bluetooth con un Arduino Pro Mini y un MPU (Unidad de Medición Inercial) como el MPU6050, seguimos los siguientes pasos básicos teniendo en cuenta los materiales a utilizar y conexiones de los pines. Materiales necesarios para la conexión:

- Arduino Mini Pro
- Módulo Bluetooth (por ejemplo, HC-05 o HC-06)
- MPU6050 (módulo de sensor de movimiento)
- Cables de conexión
- Fuente de alimentación para el Arduino Pro Mini

En la Figura 2.2, se puede observar la conexión entre un dispositivo Bluetooth, un MPU (Unidad de Procesamiento de Movimiento) y un Arduino Pro Mini, esto implica realizar la comunicación del dispositivo Bluetooth y el Arduino, y la integración del MPU para la detección y procesamiento de movimientos.

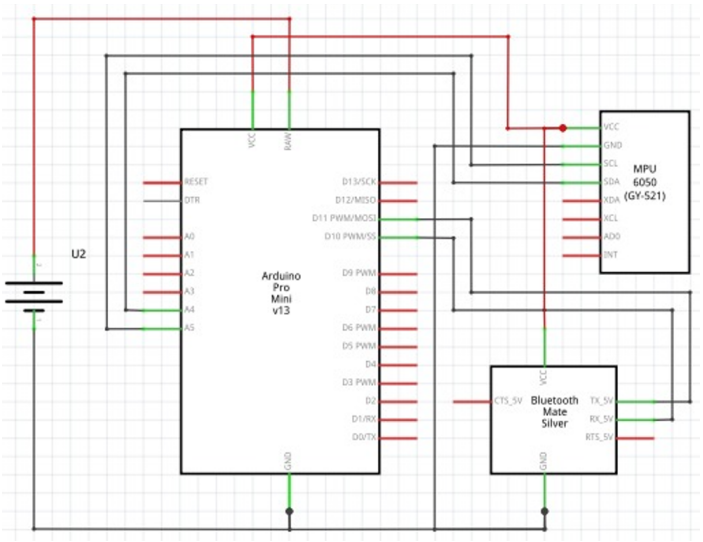


Figura 2.2: Circuito eléctrico del dispositivo [Autor]

2.1.2. Diseño de la Placa en Proteus

El software Proteus nos permite diseñar, simular y verificar el funcionamiento de circuitos electrónicos antes de implementarlos en hardware real. Este programa consta de dos softwares: Isis y Ares. A continuación, daremos una breve explicación de los diferentes softwares.

Diseño del circuito electrónico en Isis

Utilizamos las herramientas de diseño Isis para el circuito esquemático, ya que es una parte del software Proteus que se utiliza para simular y diseñar circuitos electrónicos. En la Figura 2.3, se observa el diseño del circuito electrónico en Isis.

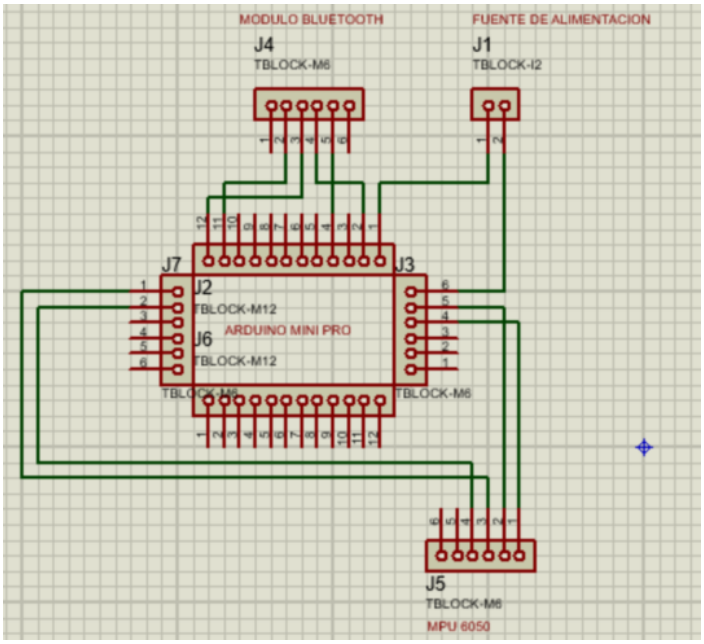


Figura 2.3: Diseño del circuito electrónico en Isis [Autor]

Distribución de componentes y diseño de pistas en Ares

En el editor de PCB, se coloca los componentes en la posición deseada. Ya que podemos utilizar la opción de "Auto Route" para que Proteus enlace automáticamente los componentes con las pistas o utiliza las herramientas de diseño para trazar las pistas que conectan los componentes. Asegurándonos de que las pistas no se crucen y de que haya suficiente espacio entre ellas. Tal y como se ve en la Figura 2.4.

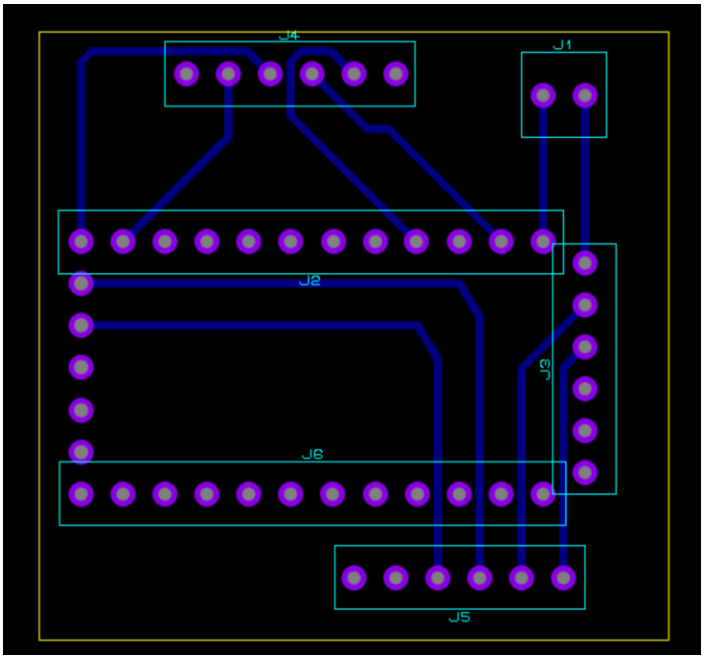


Figura 2.4: Distribución de componentes y diseño de pistas en Ares [Autor]

Huellas y pistas para la generación del archivo de salida

Una vez que hayamos diseñado la placa PCB, generamos el archivo de salida. Para imprimirlo y quemarlo en la placa como se puede observar a continuación en la Figura 2.5.

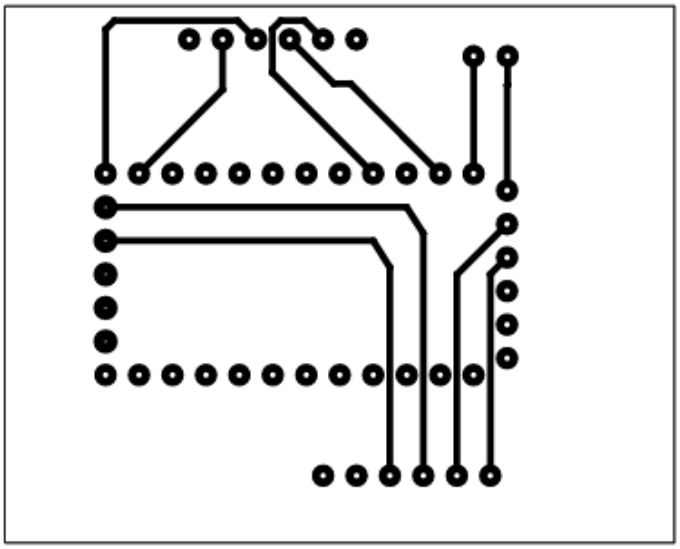


Figura 2.5: Huellas y pistas para la generación del archivo de salida [Autor]

2.1.3. Desarrollo del Software del dispositivo electrónico

A continuación, se presenta el desarrollo del software necesario para controlar el dispositivo electrónico mediante el movimiento de la cabeza. Se detallan las diferentes etapas de programación del Arduino Pro Mini, como es la lectura de los datos del módulo MPU hasta la comunicación inalámbrica a través del módulo Bluetooth como se presenta en la Figura 2.6.

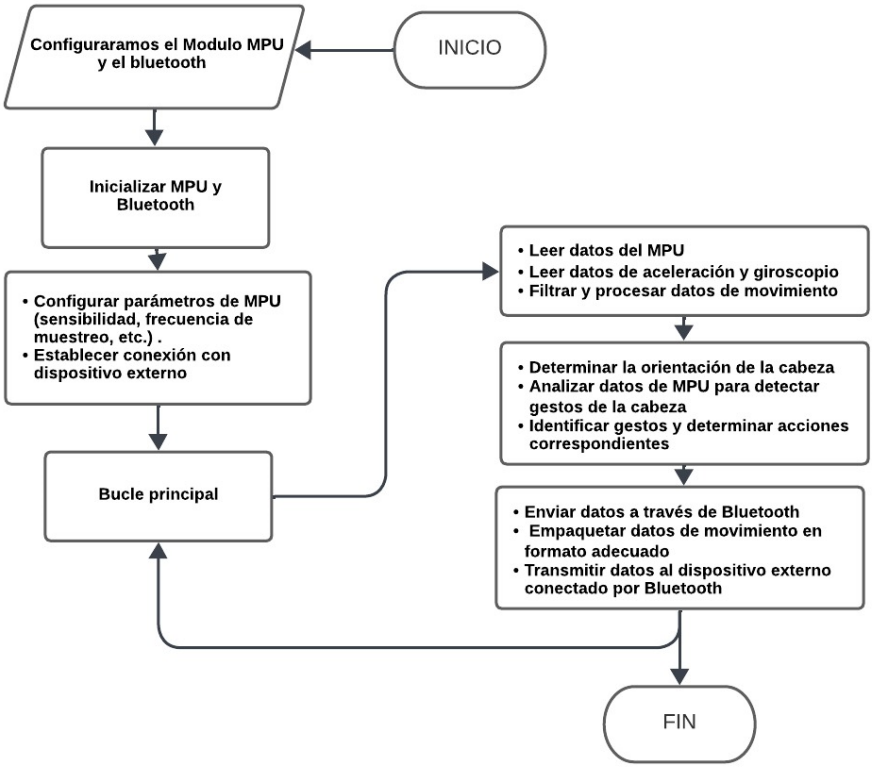


Figura 2.6: Diagrama de flujo del software en Arduino [Autor]

Descripcion del desarrollo del Software

- Programamos el Arduino Pro Mini para leer los datos del módulo MPU y procesarlos para determinar la orientación de la cabeza del usuario.
- Desarrollamos un algoritmo de filtrado y suavizado para mejorar la precisión de captura de datos de movimiento obtenidos del MPU.
- Desarrollamos un código para establecer la comunicación Bluetooth y enviar los datos de movimiento a otros dispositivos conectados.

2.1.4. Diseño de la carcasa del dispositivo

Utilizamos la herramienta de modelado en 3D de Fusión 360 para crear la forma general de la carcasa de acuerdo con las dimensiones y especificaciones de todos los componentes para que abarquen en la carcasa. En la Figura 2.7 se observa las medidas que posee la carcasa.

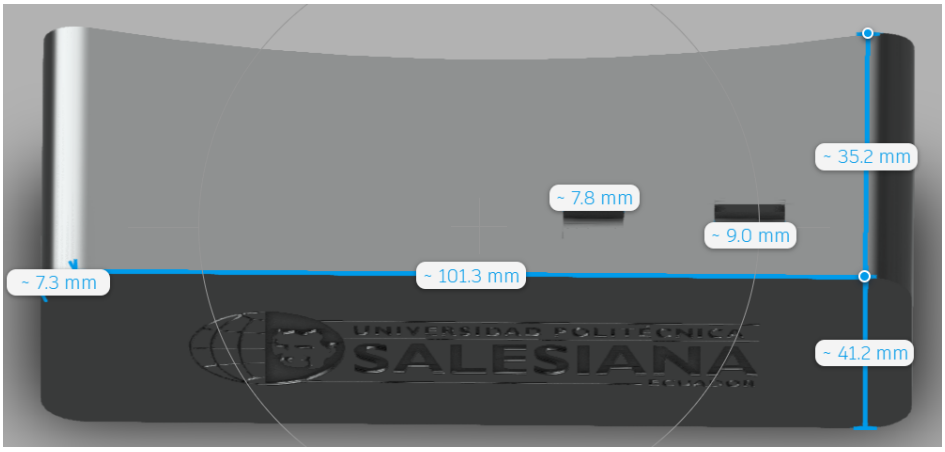


Figura 2.7: Medidas de la carcasa del dispositivo [Autor]

Ahora, agregamos detalles técnicos como orificios de montaje, ranuras para ventilación, aberturas para cables y conectores. Así, asegurándonos de que estos detalles sean precisos y estén ubicados correctamente para garantizar un montaje adecuado de los componentes internos. En la Figura 2.8 se observa este proceso.

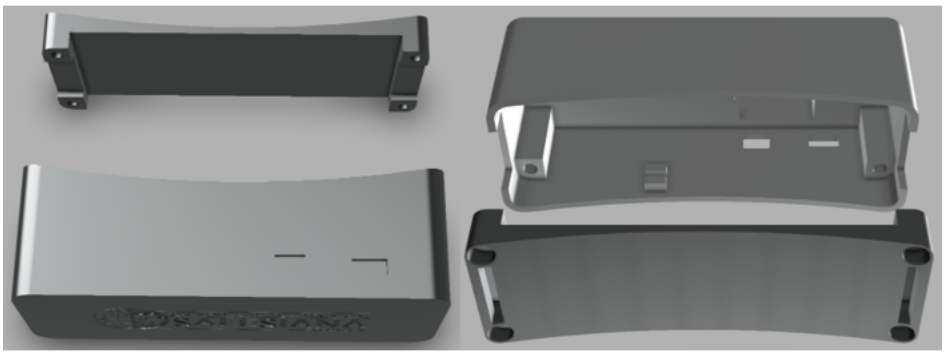


Figura 2.8: Detalles técnicos de la carcasa del dispositivo [Autor]

Finalmente, utilizamos las herramientas de personalización de Fusión 360 para agregar detalles estéticos personalizados, como grabados o logotipos. Esto experimentamos con diferentes texturas y acabados para mejorar la apariencia visual de la carcasa. En la Figura 2.9 se muestra como queda personalizada la carcasa.

Figura 2.9: Personalización de la carcasa del dispositivo [Autor]

2.2. Diseño de la Aplicación

En está sección realizaremos el diseño de la aplicación. Con este diseño se busca presentar una disposición ordenada de pictogramas de ARASACC en diferentes categorías, estos pictogramas serán representados como una opción seleccionable para el usuario en donde ellos puedan interactuar con sus tutores o familiares. Adicionalmente, en esta sección se detallará las funcionalidades que poseerá esta aplicación.

2.2.1. Interfaz de usuario

La aplicación se compone principalmente de tres interfaces: Interfaz de Inicio de Sesión, Interfaz de Categorías de Pictogramas e Interfaz de Selección de

Pictogramas, cada una de ellas diseñada para cumplir con funciones específicas. A continuación, se detalla cada una de ellas:

a. Interfaz de Inicio de sesión

El objetivo de diseñar esta interfaz es garantizar la privacidad del usuario y no permitir el acceso sin autorización a la aplicación. Para esto se necesita que el usuario ingrese sus credenciales (nombre de usuario y contraseña) para autenticarse e ingresar al contenido personalizado que contiene la aplicación.

En la Figura 2.10, observamos como quedaría diseñada la Interfaz de Inicio de Sesión. Aquí se puede ver los diferentes campos que contendrá esta vista.

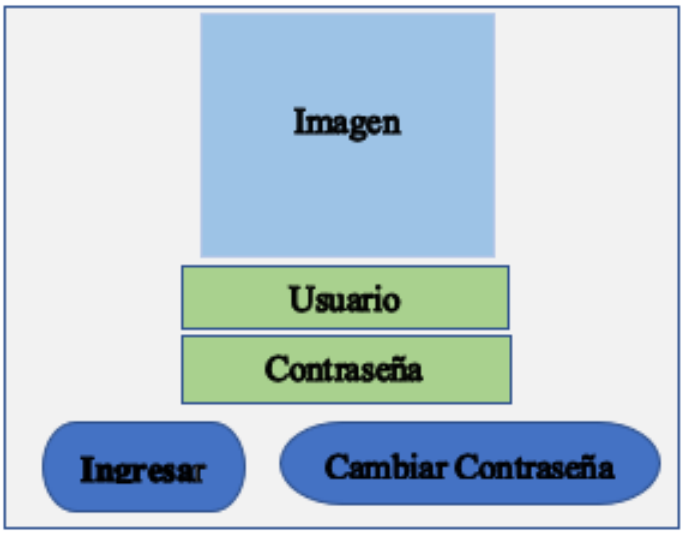


Figura 2.10: Diseño de la Interfaz de Inicio de Sesión [Autor]

b. Interfaz de Categorías de Pictogramas

En el diseño de esta interfaz, el usuario podrá ver 6 categorías que contendrán pictogramas de ARASACC, cada categoría representada respectivamente con una temática específica como son: Emociones, Comidas y Bebidas, Enfermedades, Partes del Cuerpo, Números, Prendas de Vestir. Además, en esta vista se incorporarán botones que nos permitirán realizar la conexión Bluetooth con el módulo mouse diseñado previamente en el apartado 2.1., esto con el objetivo de que exista una comunicación entre ellos y poder tomar los datos que llegan del dispositivo electrónico e interpretarlos como movimientos del mouse en la pantalla para poder seleccionar

una categoría especifica. En la Figura 2.11, se puede ver la disposición de los campos que contendrá la Interfaz de Categorías de Pictogramas.

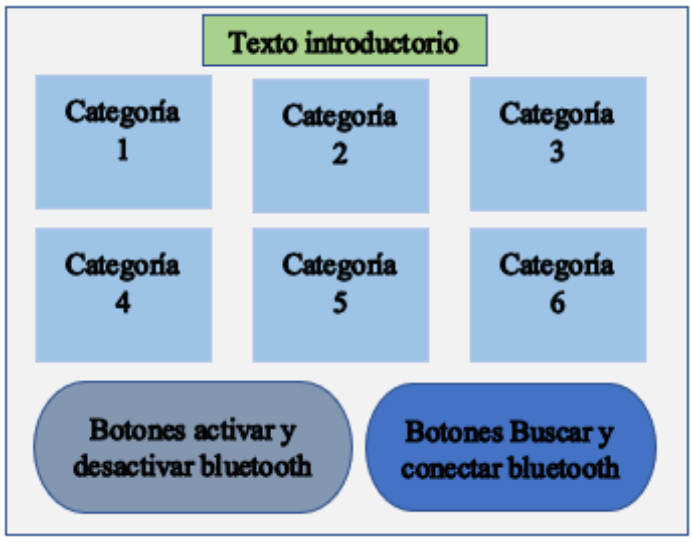


Figura 2.11: Diseño de la Interfaz de Categorías de Pictogramas [Autor]

c. Interfaz de Selección de Pictogramas

Para el diseño de la Interfaz de Selección de Pictogramas, se tomó en consideración incorporar una lista de 16 pictogramas de ARASACC, relacionados con cada categoría. Esto con el objetivo de que a la hora que el usuario quiera elegir un pictograma no tenga una lista muy extensa, haciendo que este pierda el interés o simplemente no se decida cual elegir, además, esto ayudará a que haya un mejor control para el tutor o familiar que este guiando al usuario.

Por otra parte, una vez que el usuario encuentre el pictograma deseado y lo seleccione este pictograma aumentará de tamaño, esto permitirá mejorar la visibilidad de la imagen seleccionada. Adicionalmente, en el momento que la imagen seleccionada se hace más grande aparecerá una barra de sonido en la parte inferior de la imagen, en este sonido se escuchará el nombre de la imagen haciendo que el usuario tenga una interacción más intuitiva a la hora que seleccione el pictograma.

En la Figura 2.12, se puede ver la disposición de los campos que contendrá la Interfaz de Selección de Pictogramas.

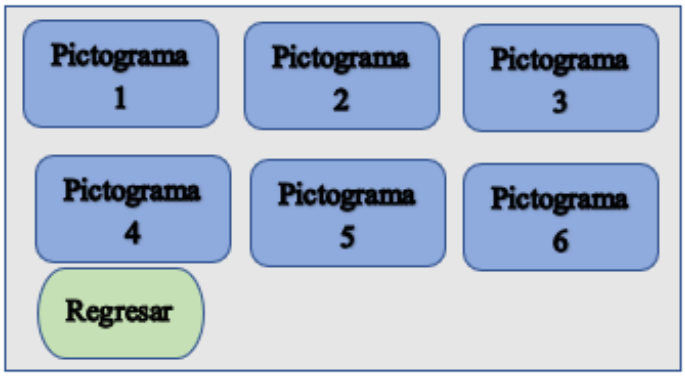


Figura 2.12: Diseño de la Interfaz de Selección de Pictogramas [Autor]

2.2.2. Funcionalidades de la Aplicación

La principal función de esta aplicación es permitir al usuario seleccionar pictogramas mediante el seguimiento de movimientos de la cabeza. Este proceso implica que el usuario pueda desplazar el cursor virtual sobre los distintos pictogramas de ARASACC en la pantalla y seleccionar uno de ellos. Por tal motivo, la aplicación ha sido diseñada para ofrecer una navegación eficiente y cómoda entre las diferentes interfaces vistas anteriormente en la sección 2.2.1.

Así mismo, una característica clave para la selección del pictograma el usuario debe posicionarse con el cursor durante un tiempo determinado en el pictograma que desea y entonces este se seleccionará automáticamente. Esta funcionalidad permite a la aplicación adaptarse a las necesidades específicas de cada usuario, garantizando una experiencia personalizada y cómoda.

Por otra parte, la aplicación facilita el proceso de emparejamiento y desconexión del dispositivo mediante Bluetooth. Esta conexión inalámbrica se establece de manera segura y eficiente, permitiendo que el dispositivo móvil y el dispositivo electrónico de seguimiento de cabeza se comuniquen sin complicaciones. De manera que, la capacidad de conectar y desconectar fácilmente el dispositivo contribuye a una experiencia de usuario más fácil y eficaz.

En resumen, estas funcionalidades se combinan para ofrecer una experiencia de usuario cómoda y efectiva. Por esta razón, la aplicación proporciona no solo una forma efectiva de selección de pictogramas, sino también una navegación fácil, una conectividad segura y una retroalimentación visual clara. Estas características trabajan

en conjunto para optimizar la usabilidad y asegurar que la aplicación sea accesible, eficiente y adecuada para una amplia variedad de usuarios.

2.3. Arquitectura del sistema

En el desarrollo de este proyecto, nos basamos en una arquitectura que está compuesta del siguiente proceso, en donde el usuario mediante un dispositivo electrónico que captura datos de los movimientos de la cabeza interactúa con una aplicación que contiene pictogramas con los cuales el usuario puede seleccionar uno de ellos y comunicarse con la persona encargada de cuidar de él. A continuación se detalla este proceso:

- El usuario interactúa con la interfaz de usuario utilizando el dispositivo electrónico. Este dispositivo captura datos de los movimientos realizados mediante la cabeza y los envía mediante bluetooth a la aplicación.
- En la aplicación se recibe los datos provenientes del dispositivo electrónico y los interpreta para convertirlos en movimientos del mouse en la pantalla de la aplicación, mediante los movimientos del mouse el usuario puede seleccionar los pictogramas que se encuentran definidos en la aplicación según sean sus necesidades.
- Una persona encargada observa la interacción que tenga el usuario con la aplicación para posteriormente evaluar los resultados que se presentan en la comunicación del usuario mediante el sistema desarrollado.

En la Figura 2.13, se puede ver el funcionamiento de nuestra propuesta de arquitectura del sistema.

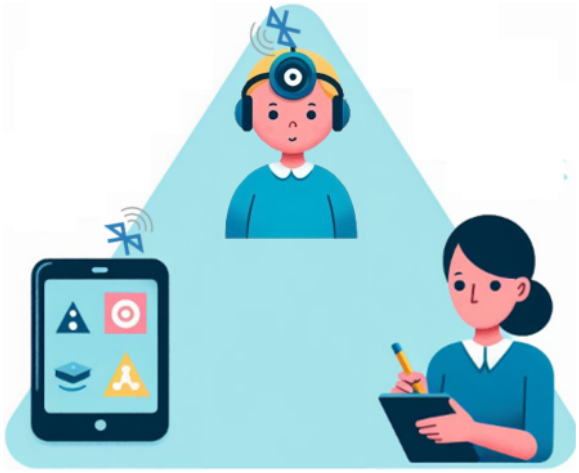


Figura 2.13: Arquitectura del sistema propuesto [Autor]

Con la implementación de esta arquitectura se pretende realizar una comunicación eficiente entre el usuario y las personas que lo asisten ya sean sus tutores o familiares.

Capítulo 3

Implementación

En este capítulo, se describe el proceso de implementación del dispositivo tipo mouse para la captura de datos de los movimientos de la cabeza, así como la aplicación de tablet que contiene pictogramas de Arasac divididos en categorías. La implementación de este proyecto involucró la integración de hardware y software para lograr un sistema funcional y accesible.

Comenzando con la configuración del hardware, se explica cómo se conectaron y configuraron el Arduino Pro Mini, el MPU-5060 y el módulo Bluetooth para capturar y transmitir los datos de movimiento. Luego, se aborda el desarrollo del software, describiendo el código Arduino utilizado para procesar los datos del sensor y la aplicación de la tablet desarrollada para interactuar con el dispositivo.

3.1. Implementación del Módulo Mouse

En esta sección se presenta en detalle la implementación del hardware utilizado para crear el dispositivo tipo mouse que detecte movimientos de la cabeza. La combinación de componentes electrónicos como el Arduino Pro Mini, el MPU-5060 y el módulo Bluetooth constituyen la columna vertebral de nuestro sistema.

Se exploran también los desafíos y consideraciones específicas que surgieron durante la implementación del hardware. Esto incluye la calibración del MPU-5060 para garantizar mediciones precisas de los movimientos de la cabeza, así como la selección adecuada de componentes adicionales para optimizar el rendimiento

general del dispositivo.

3.1.1. Desarrollo del Módulo Mouse

Como se explico en el Capitulo 2, comenzando con la configuración física, se detalla cómo se conectaron cada uno de los componentes para garantizar una interacción fluida y precisa entre ellos. Desde la conexión de los pines del MPU-5060 al Arduino hasta la configuración del módulo Bluetooth para la comunicación inalámbrica.

Visualización de los componentes y la placa

La Figura 3.1, presenta la distribucion de los diferentes componetes en la placa con sus respectivos orificios, como se puede observar en la Figura del lado izquierdo.

En la Figura del lado derecho se observa la conexión de los componentes (Arduino Pro Mini, Módulo Bluetooth, MPU6050), con la batería de 7.5 V para su alimentación.

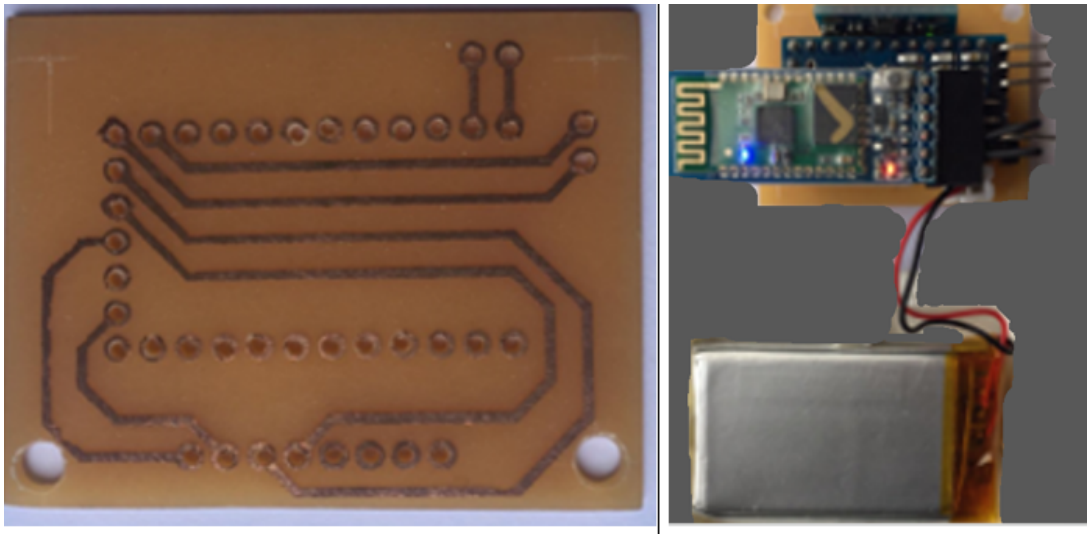


Figura 3.1: Visualización de los componentes y la placa [Autor]

Implementación de los componentes en la placa

La Figura 3.2, presenta los componentes (Arduino Pro Mini, Módulo Bluetooth, MPU6050), ya distribuidos en los distintos lugares de la placa diseñada por el software

Proteus.

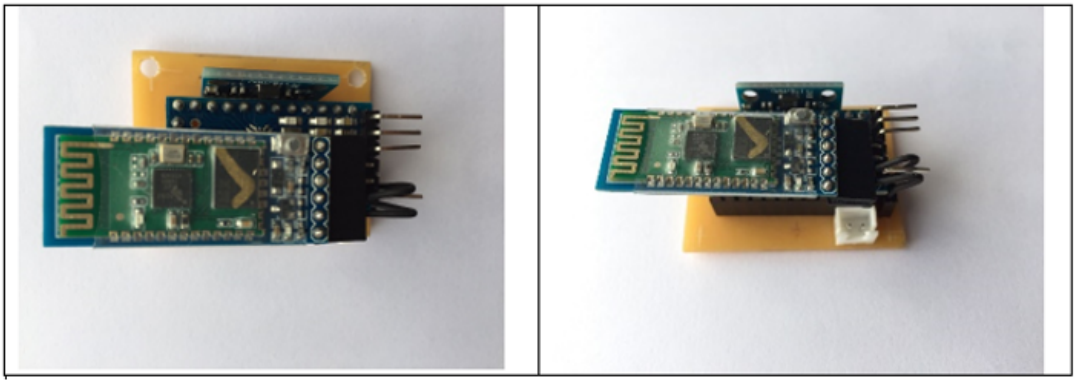


Figura 3.2: Visualización de los componentes en la placa [Autor]

Implementación de los componentes en la carcasa

En la Figura 3.3, se presenta detalladamente como quedan distribuidos los componentes electrónicos en la carcasa la cual fue diseñada por el software Fusión 360.

Figura 3.3: Visualización de los componentes en la carcasa [Autor]

3.2. Implementación de la Aplicación en Android Studio

En esta sección, nos centraremos en la implementación de la aplicación utilizando el lenguaje de programación Kotlin en Android Studio. Aquí, se proporcionarán detalles sobre la implementación de las funciones que contiene esta aplicación.

3.2.1. Desarrollo de la interfaz de usuario

Como se explicó en el Capítulo 2 (Sección 2.2.1), la aplicación se compone principalmente de tres interfaces, las cuales son: Interfaz de Inicio de Sesión, Interfaz de Categorías de Pictogramas e Interfaz de Selección de Pictogramas. A continuación, se explicará el desarrollo de cada vista que contiene la aplicación:

Implementación de la Interfaz de Inicio de Sesión (Log-In)

La Figura 3.4, muestra la implementación de esta interfaz, con las funciones de Nombre de Usuario, Contraseña y un botón para ingresar. Estos campos fueron implementados de manera que el usuario pueda acceder de manera segura a la aplicación.

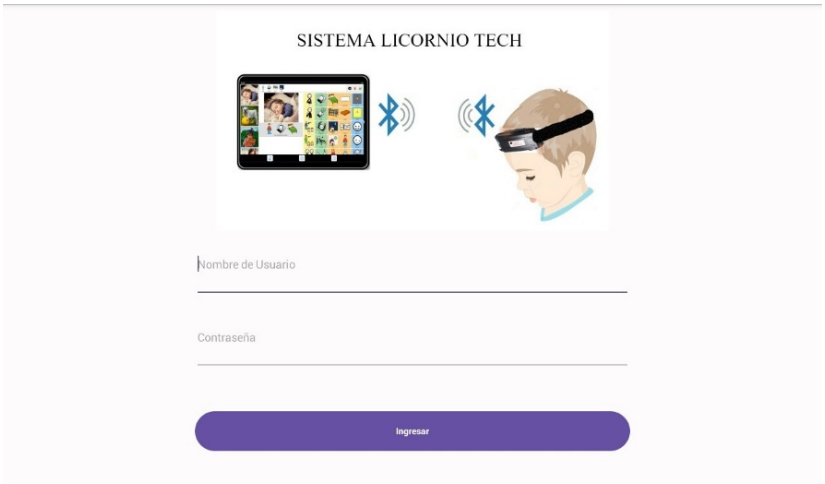


Figura 3.4: Implementación de la Interfaz de Inicio de Sesión [Autor]

Implementación de la Interfaz de Categorías de Pictogramas

En la Figura 3.5, se observa las categorías de pictogramas de ARASACC definidas en esta interfaz en las cuales se implementó seis categorías, cada categoría contiene dieciséis pictogramas relacionados con esa categoría. Además, se muestra los botones los cuales fueron implementados con el objetivo de realizar la comunicación con el dispositivo mouse mediante bluetooth.

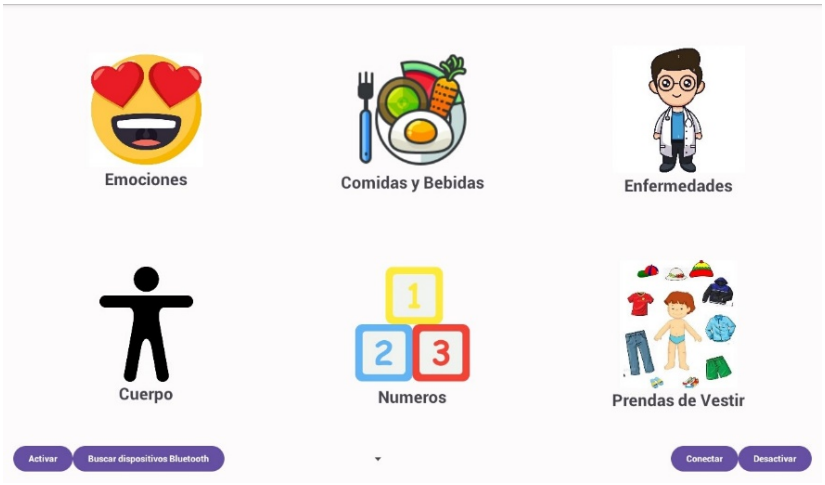


Figura 3.5: Implementación de la Interfaz de Categorías de Pictogramas [Autor]

Implementación de la Interfaz de Selección de Pictogramas

En la Figura 3.6, se pueden ver los diferentes pictogramas de ARASACC correspondientes a la categoría de Emociones. En total fueron colocados dieciséis pictogramas en cada categoría y adicionalmente se colocó un botón para regresar a la Interfaz de Categorías de Pictogramas descrita anteriormente.

Figura 3.6: Pictogramas de la categoría emociones [Autor]

En la Figura 3.7, se muestran ocho de los dieciseis pictogramas de ARASACC correspondientes a la categoría de Comidas y Bebidas. En total fueron colocados dieciséis pictogramas en esta categoría.

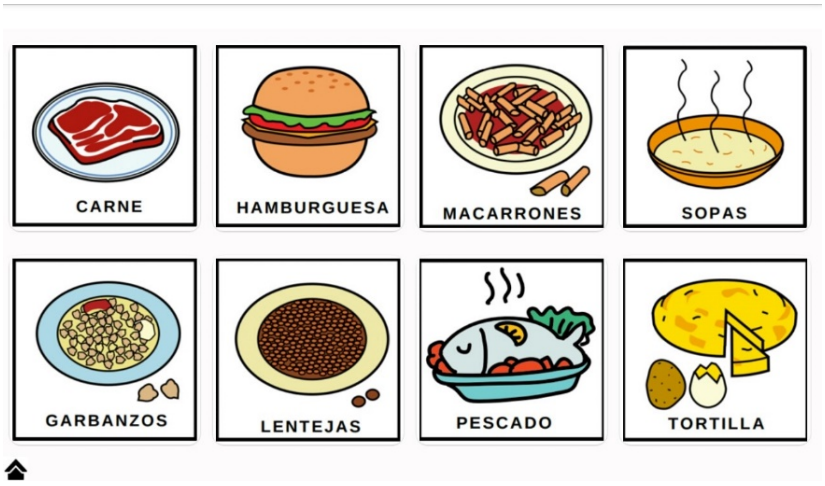


Figura 3.7: Pictogramas de la categoría comidas y bebidas [Autor]

En la Figura 3.8, se muestran ocho de los dieciseis pictogramas de ARASACC correspondientes a la categoría de Enfermedades. En total fueron colocados dieciséis pictogramas en esta categoría.

Figura 3.8: Pictogramas de la categoría enfermedades [Autor]

En la Figura 3.9, se muestran ocho de los dieciseis pictogramas de ARASACC correspondientes a la categoría de Cuerpo. En total fueron colocados dieciséis pictogramas en esta categoría.

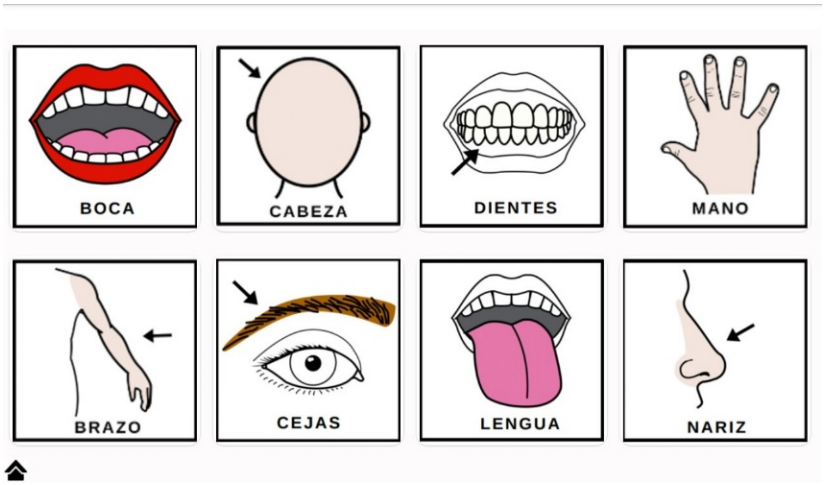


Figura 3.9: Pictogramas de la categoría cuerpo [Autor]

En la Figura 3.10, se muestran ocho de los dieciseis pictogramas de ARASACC correspondientes a la categoría de Números. En total fueron colocados dieciséis pictogramas en esta categoría.

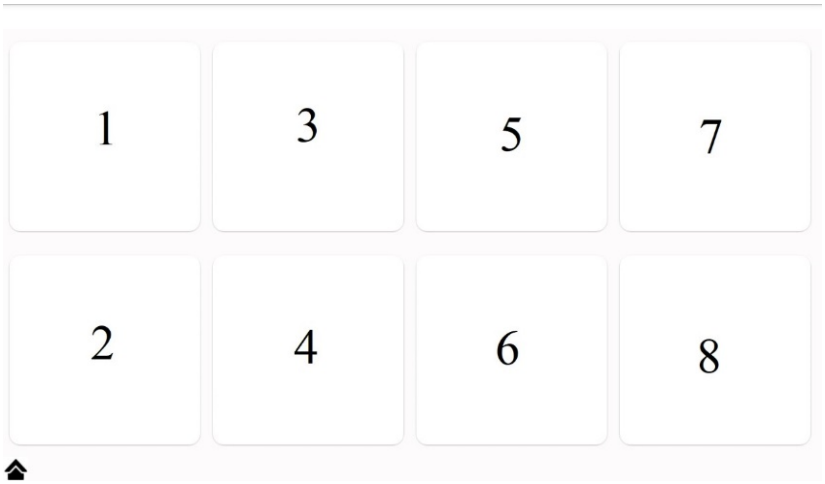


Figura 3.10: Pictogramas de la categoría números [Autor]

En la Figura 3.11, se muestran ocho de los dieciseis pictogramas de ARASACC correspondientes a la categoría de Prendas de Vestir. En total fueron colocados dieciséis pictogramas en esta categoría.

Figura 3.11: Pictogramas de la categoría prendas de vestir [Autor]

Finalmente, en la Figura 3.12, se muestra la función de seleccionar un pictograma la cual se realza haciendo más grande la visualización del pictograma y además contiene un sonido relacionado con dicho pictograma. Cabe aclarar que todos los pictogramas realizan esta acción, pero como ejemplo, se muestra solo en un pictograma para demostrar su funcionalidad.

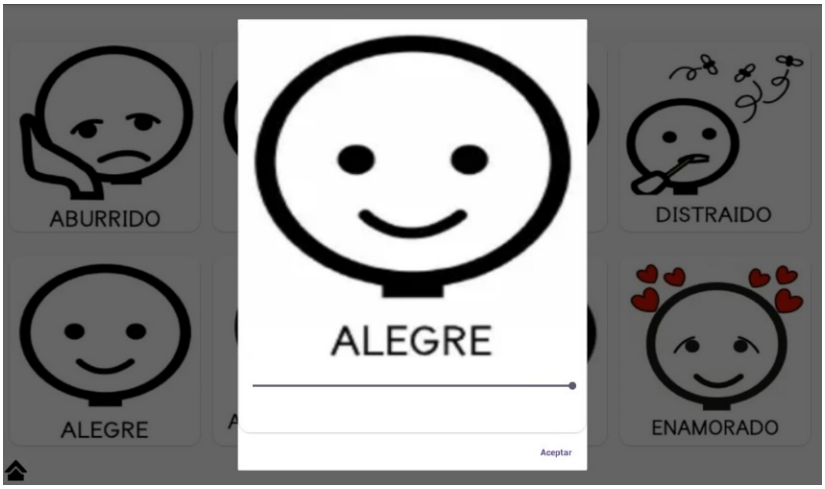


Figura 3.12: Pictograma seleccionado [Autor]

Capítulo 4

Resultados, Conclusiones y Recomendaciones

En este capítulo, se presentan los resultados obtenidos tras realizar las pruebas del dispositivo tipo mouse en niños con IMOC del Instituto de Parálisis Cerebral (IPCA), así como en niños sin ninguna capacidad diferente. Estos resultados reflejan el rendimiento, la usabilidad y la eficacia general del sistema, así como los logros alcanzados en relación con los objetivos establecidos para el proyecto. Por último, presentamos las conclusiones y recomendaciones.

4.1. Resultados Obtenidos de las Pruebas

Estas pruebas se realizaron en niños y jóvenes del Instituto de Parálisis Cerebral del Azuay (IPCA), entre edades de 12 a 17 años, los cuales poseen un grado de discapacidad de nivel IV con IMOC. Estas pruebas fueron realizadas con asesoramiento de profesores de dicha institución. Además, para tener una mejor percepción del funcionamiento del dispositivo, las pruebas también fueron realizadas en niños entre edades de 4 a 8 años, que no poseen ninguna discapacidad motora.

En la Tabla 4.1, se presentan los datos tomados de las personas con las que se realizaron las pruebas como son: Estudiante, Edad, Grado de discapacidad, Nivel Académico y Género.

Tabla 4.1: Datos de los estudiantes con los que se realizaron las pruebas del dispositivo mouse[Autor]

Estudiante	Edad	Grado de discapacidad	Nivel Académico	Género
Alumno 1	17 Años	IV	Funcional Social 3	Masculino
Alumno 2	12 Años	IV	Funcional Social 3	Masculino
Alumno 3	17 Años	IV	Funcional Social 3	Femenino
Alumno 4	10 Años	IV	Funcional Social 1	Femenino
Alumno 5	8 Años	I	Cuarto de Básica	Masculino
Alumno 6	4 Años	I	Inicial 2	Masculino

Nota: Cabe resaltar que el grado de discapacidad I es de 0 %.

Pruebas del dispositivo mouse en el instituto IPCA

Para verificar el funcionamiento del dispositivo tipo mouse y de aplicación con pictogramas, estos fueron probados en diferentes estudiantes del instituto IPCA. En estas pruebas el objetivo fue que los estudiantes pudieran seleccionar los pictogramas de ARASAAC que contiene la aplicación mediante el dispositivo tipo mouse colocado en la cabeza.

En el caso del Alumno 1, se pudo observar que al principio le costo un poco mantener la concentración para poder seleccionar un pictograma especifico, ya que era la primera vez que utilizaba el dispositivo y movía la cabeza de un lado al otro. Pero finalmente, con la ayuda de su profesora quien le guio para que se mantenga concentrado pudo seleccionar los pictogramas de la aplicación con el dispositivo mouse. En la Figura 4.1, se observa como el Alumno 1 realiza las pruebas del dispositivo mouse y la aplicación con pictogramas.

Figura 4.1: Pruebas con el Alumno 1, selección de pictogramas [Autor]

En el caso del Alumno 2, tuvo una mejor respuesta a la hora de interactuar con la aplicación y el dispositivo, ya que previamente se le explico como tenia que

utilizar el dispositivo para que seleccione los pictogramas, con esto el estudiante pudo seleccionar con mayor facilidad el pictograma que deseaba. En la Figura 4.2, se observa como realiza las pruebas el Alumno 2.

Figura 4.2: Pruebas con el Alumno 2, selección de pictogramas [Autor]

En el caso del Alumno 3, se realizaron pruebas con respecto a las emociones que presentaba en esos momentos. Se le preguntaba ¿cómo se sentía? Y ella escogía una emoción y la seleccionaba. Se puede decir que las pruebas con la estudiante fueron buenas. En la Figura 4.3, se observa como se realiza las pruebas con el Alumno 3.

Figura 4.3: Pruebas con el Alumno 3, selección de pictogramas [Autor]

Finalmente, las pruebas realizadas con el Alumno 4 tuvieron una mejor respuesta, ya que luego de haber realizado las pruebas con los otros estudiantes se

le pudo explicar de mejor manera como tenia que utilizar el dispositivo tipo mouse con la aplicación. Esto ayudo a que se le haga más fácil interactuar y seleccionar los pictogramas. En la Figura 4.4, se muestra las pruebas realizadas con el Alumno 4.

Figura 4.4: Pruebas con el Alumno 4, selección de pictogramas [Autor]

En la Tabla 4.2, se muestra los porcentajes de efectividad que se obtuvieron al realizar las pruebas del dispositivo tipo mouse con la aplicación que contiene pictogramas de ARASAAC, en los estudiantes de la institución IPCA.

Tabla 4.2: Porcentajes de efectividad de las pruebas realizadas en estudiantes de Instituto IPCA [Autor]

Estudiante	Número de Intentos Realizados	Porcentaje de Efectividad
Alumno 1	10	50 %
Alumno 2	10	70 %
Alumno 3	10	60 %
Alumno 4	10	80 %

Según la Tabla 4.2, la efectividad de la utilización del dispositivo fue del 65 %, esto es en base a las primeras pruebas de utilización del dispostivo, pero seguramente con la utilización diaria del mismo el porcentaje seguro alcanzará un mejor rendimiento.

Pruebas del dispositivo mouse en niños sin ninguna capacidad diferente

Los resultados obtenidos de realizar las pruebas del dispositivo en el Alumno 5 fueron muy satisfactorios, ya que podía seleccionar los pictogramas de la aplicación con mucha facilidad, demostrando que el dispositivo funcionaba correctamente. En la Figura 4.5, se puede observar como el niño interactúa con el dispositivo mouse y la aplicación.

Figura 4.5: Pruebas con el Alumno 5, selección de pictogramas [Autor]

En las pruebas del Alumno 6 se observó una alta precisión en la captura de datos de movimiento por parte del dispositivo mouse. También, se pudo observar que la interfaz de usuario basada en pictogramas de ARASAAC resultó altamente intuitiva y fácil de usar para los niños. En la Figura 4.6, se observa las pruebas que se realizaron.

Figura 4.6: Pruebas con el Alumno 6, selección de pictogramas [Autor]

En la Tabla 4.3, se muestra los porcentajes de efectividad que se obtuvieron al realizar las pruebas del dispositivo tipo mouse con la aplicación que contiene pictogramas de ARASAAC, en niños sin ninguna capacidad diferente.

Tabla 4.3: Porcentajes de efectividad de las pruebas realizadas del dispositivo en niños sin ninguna capacidad diferente [Autor]

Estudiante	Número de Intentos Realizados	Porcentaje de Efectividad
Alumno 5	10	100 %
Alumno 6	10	100 %

En general, tanto los niños con IMOC como los niños sin ninguna capacidad diferente mostraron una alta aceptación y facilidad de uso del dispositivo. La interfaz de pictogramas de ARASAAC resultó intuitiva y fácil de entender para ambos grupos de niños, lo que facilitó la interacción con el sistema.

4.2. Conclusiones

Este proyecto resalta la importancia de considerar todos los aspectos del sistema: hardware, software, interacción usuario-dispositivo, y experiencia general del usuario.

Para el hardware se trabajó con componentes como el MPU-5060, modulo bluetooth y el Arduino Pro Mini. Esto implica no solo la capacidad de conectarlos físicamente, sino también de comprender cómo interactúan y cómo optimizar su rendimiento.

En cuanto a la evaluación de la efectividad del dispositivo parece estar teniendo un impacto positivo en el desempeño de los niños, dado que el porcentaje de efectividad total es del 76 %, asi logrando pasar de un proceso manual a uno automático, donde muchas veces quedaba a la percepción de la educadora o acompañante.

Dado que el dispositivo está destinado a mejorar la comunicación de los niños con IMOC, y siendo un prototipo se debe seguir con las pruebas para entender mejor su impacto y realizar algun ajuste necesario.

Se ha logrado que la estructura de la interfaz de usuario sea intuitiva y fácil de usar. Esto implica no solo la disposición de los pictogramas y las categorías, sino también la implementación de controles de movimiento de cabeza que sean precisos y cómodos de usar durante períodos prolongados.

4.3. Recomendaciones

Sería relevante investigar más a fondo qué factores están contribuyendo a la variación en los porcentajes de efectividad. Esto podría incluir factores como la frecuencia de uso del dispositivo, la comprensión del material, la motivación del estudiante, entre otros.

Pruebas Exhaustivas: Previo al lanzamiento del producto definitivo, es fundamental llevar a cabo pruebas rigurosas en cada fase del proceso de desarrollo. Esto implica la realización de pruebas exhaustivas en el hardware, software, usabilidad e integración para asegurar el correcto funcionamiento del dispositivo en

diversas circunstancias.

Documentación Detallada: Tener una documentación minuciosa de todos los aspectos del proyecto simplifica el seguimiento de modificaciones, la solución de problemas,asi como la participacion de los miembros del equipo. Esta documentación debe contener diagramas, esquemas, flujos de trabajo y cualquier otro elemento pertinente para el proyecto.

Consideraciones de Accesibilidad: Si el dispositivo está destinado a usuarios con necesidades especiales, es fundamental garantizar que sea accesible para todos. Esto puede implicar la incorporación de características de accesibilidad, como opciones de personalización, soporte para diferentes idiomas y adaptaciones para diferentes niveles de habilidad.

Seguridad y Privacidad: Cuando se trata de manejar información confidencial, es esencial aplicar medidas de seguridad efectivas para salvaguardar los datos del usuario y asegurar su privacidad. Esto implica la utilización de técnicas como el cifrado de la información, la autenticación segura y el cumplimiento de las normativas de privacidad correspondientes.

Colaboración Interdisciplinaria: Fomentar la colaboración de equipos para la colaboración multidisciplinaria es fundamental para el éxito de proyecto. La combinación de habilidades en ingeniería, diseño, usabilidad y otras áreas asegura un enfoque integral y bien informado.

Al seguir estas recomendaciones se prepara para abordar proyectos similares en el futuro y para ofrecer soluciones que sean tanto técnicamente sólidas como satisfactorias para los usuarios.

Glosario

ARASAAC Sistema de Comunicación Aumentativa y Alternativa de Aragón – Augmentative and Alternative Communication System of Aragon.

CAA Comunicación Aumentativa y Alternativa – Augmentative and Alternative Communication.

IDE Entorno de Desarrollo Integrado – Integrated Development Environment.

IMOC Insuficiencia Motora de Origen Cerebral – Motor Insufficiency of Cerebral Origin.

PCB Placa de Circuito Impreso – Printed Circuit Board.

SAAC Sistema Aumentativo y Alternativo de Comunicación – Augmentative and Alternative Communication System.

SSMI Insuficiencia Motora el Habla Grave – Severe speech motor insufficiency.

Referencias

[1] C. N. para la Igualdad de Discapacidades-CONADIS, *Estadísticas de Discapacidad*, ene. de 2022. dirección: https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/.

[2] T. K. Pérez, G. F. Sosa, F. A. Ferrari y J. I. Gialonardo, «Comunicador para personas con parálisis cerebral severa,» 2023.

[3] S. Ren, «Computer Vision for Facial Analysis Using Human-Computer Interaction Models,» *Journal of Interconnection Networks*, vol. 22, 2022, Cited by: 0. DOI: 10.1142/S0219265921440059. dirección: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85120775157&doi=10.1142%2fS0219265921440059&partnerID=40&md5=ff684c62384c445ab1d581b167e4353f.

[4] J. Dv, P. Jain, A. Mukhopadhyay, K. P. S. Saluja y P. Biswas, «Eye gaze controlled adaptive virtual keyboard for users with SSMI,» *Technology and Disability*, vol. 33, págs. 319-338, 4 2021, Cited by: 1. DOI: 10.3233/TAD-200292. dirección: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85120380650&doi=10.3233%2fTAD-200292&partnerID=40&md5=e5e92b0a2224ccaefa8fa11b408b33f1.

[5] A. G. Restrepo y T. H. Torres, «PictBoard-Diseño y construcción de un prototipo de un tablero comunicador electrónico para niños con parálisis cerebral,» 2008.

[6] A. Guerrero, R. Tatiana y H. Torres, *PictBoard - Diseño y construcción de un prototipo de un tablero comunicador electrónico para niños con parálisis cerebral*, 2008. dirección: https://repository.eia.edu.co/handle/11190/5473%20https://repository.eia.edu.co/entities/publication/b594bb22-7c1b-4fbe-8b73-3e1f719685fd.

[7] U. de Castilla-La Mancha España Cabello, «Revista de Investigación en Logopedia,» *Revista de Investigación en Logopedia*, vol. 5, págs. 60-70, 1 2015, ISSN: 2174-5218. dirección: http://revistalogopedia.uclm.es.

[8] R. Cerino, D. Pinto, S. Vergara, R. Cerino, D. Pinto y S. Vergara, «Representación pictográfica del lenguaje Toki-Pona para su uso en sistemas aumentativos y alternativos de comunicación,» *Computación y Sistemas*, vol. 27, págs. 569-580, 2 2023, ISSN: 1405-5546. DOI: 10.13053/CYS-27-2-4418. dirección: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-55462023000200569&lng=es&nrm=iso&tlng=es%20http://www.scielo.org.mx/scielo.php?script=sci_abstract&pid=S1405-55462023000200569&lng=es&nrm=iso&tlng=es.

[9] F. R. R. Jiménez, «Las interfaces humano-máquina (HMI) y su importancia en el control de procesos industriales,» 2012, ISSN: 2070-0458. dirección: http://redicces.org.sv/jspui/handle/10972/1733.

[10] *Cuadro comparativo de los principales sistemas operativos - TeExplico Docente*. dirección: https://teexplicodocente.com/2021/10/19/cuadro-comparativo-de-los-principales-sistemas-operativos/.

[11] *Lenguajes de programación para móvil*. dirección: https://immune.institute/blog/lenguajes-de-programacion-para-movil/.

[12] *8.458 imágenes, fotos de stock, objetos en 3D y vectores sobre Arduino | Shutterstock*. dirección: https://www.shutterstock.com/es/search/arduino.

[13] *Arduino - Home*. dirección: https://www.arduino.cc/.

ANEXO A: MANUAL DE USUARIO

Paso 1: Descarga e Instalación de la aplicación.

Para iniciar con la descarga de la aplicación dirígete al enlace proporcionado:
`https://drive.google.com/file/d/1XrWE_h3Vdm96aeXJl1ZTx9thnTbJwAgW/view?usp=sharing`
Luego de haber entrado en el enlace toca "Descargar" para iniciar la descarga.

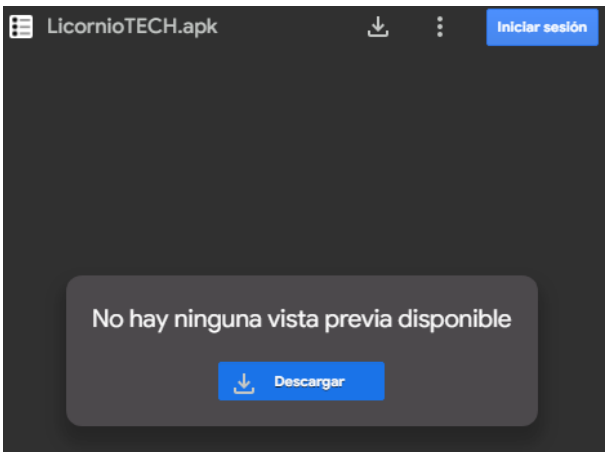


Figura A. 1 Proceso de descarga

Una vez que se allá descargado la aplicación en tu dispositivo, dar clic en el archivo APK descargado para empezar con la instalación. Si es la primera vez que instalas una aplicación fuera de Google Play Store, es posible que debas habilitar la opción de "Fuentes Desconocidas".

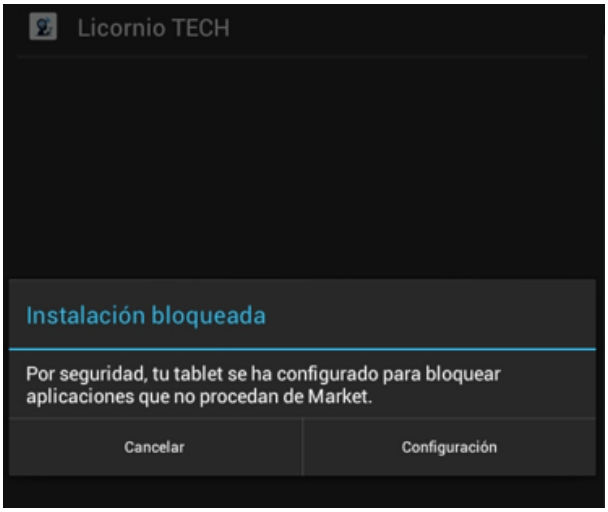


Figura A. 2 Habilitar fuentes desconocidas

Ahora, Selecciona "Instalar" cuando se te solicite.

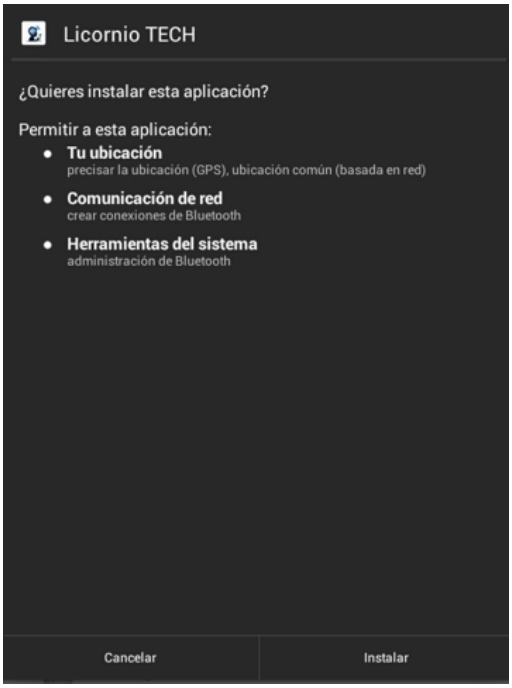


Figura A. 3 Proceso de instalar

Espera a que se complete la instalación. Puede que necesites conceder permisos adicionales durante este proceso. Una vez finalizada la instalación, verás la opción de "Abrir". Toca en ella para iniciar la aplicación.

Figura A. 4 Proceso de abrir

Paso 2: Emparejamiento del Dispositivo Mouse con la aplicación.

Para empezar con el emparejamiento, asegúrate que el dispositivo mouse este cargado y encendido. Para encender utiliza el switch deslizante incorporado en el dispositivo.

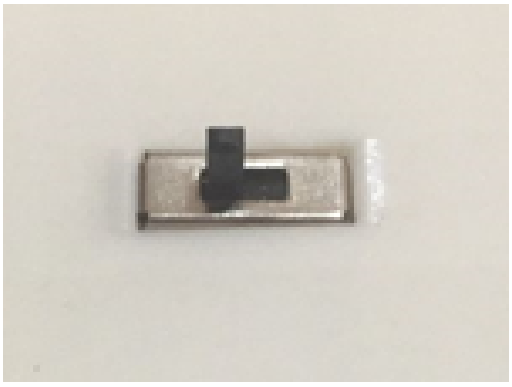


Figura A. 5 Switch de encendido del dispositico mouse

Ahora, ve a tu dispositivo móvil (en este caso una tableta en la que previamente te descargaste la App), entra en **Ajustes>Conexiones Inalámbricas y redes**, y busca "**Bluetooth**".

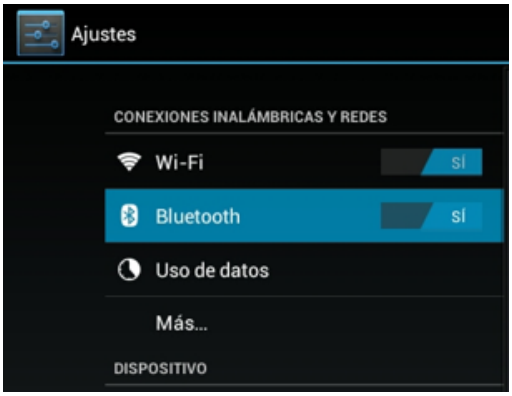


Figura A. 6 Ajustes de la tableta

Luego da clic en la opción de "**Buscar Dispositivos**", espera unos segundos y te aparecerá los dispositivos cercanos a los que te puedes conectar elige el que tenga el nombre de "**LicornioTECH**". Te pedirá permisos para realizar el emparejamiento dar en "**Sí**".

Finalmente, el dispositivo se abra sincronizado y podrás utilizarlo con la aplicación.

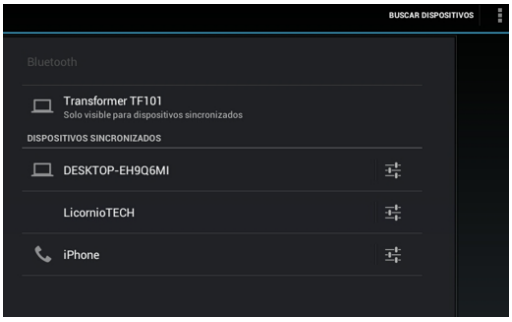


Figura A. 7 Sincronización de dispositivos

Paso 3: Descarga e Instalación de la aplicación.

Inicia sesión en la aplicación con el nombre de usuario definido como "**ProyectoLicornio**" y la contraseña "**tesis**". Una vez iniciado sesión, aparecerá la interfaz de Categorías de pictogramas. Aquí podrás conectarte con el dispositivo mouse.

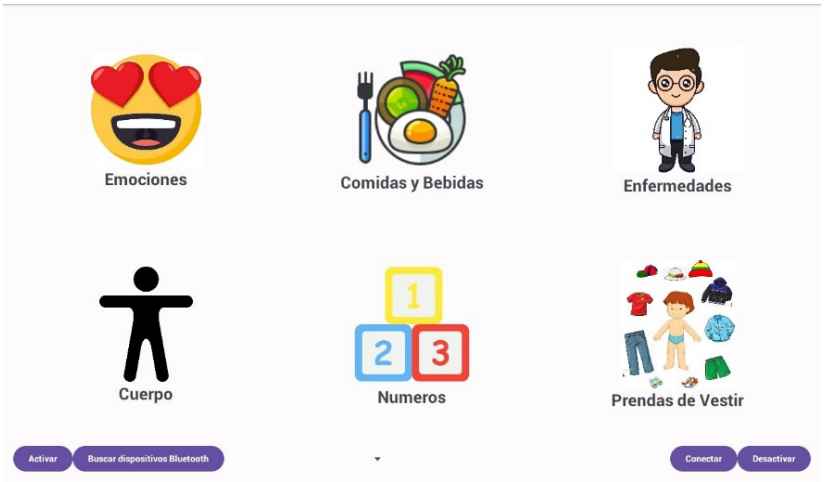


Figura A. 8 Interfaz de usuario

Para conectarse con el dispositivo mouse darle clic en el botón "**Activar**". Te solicitará permiso para activar el Bluetooth darle en "**si**".

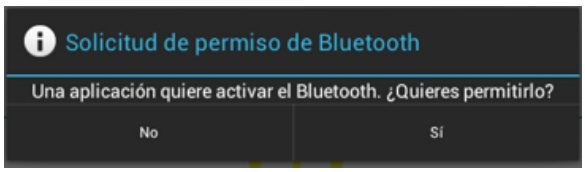


Figura A. 9 Permisos para activar Bluetooth

Una vez activado el Bluetooth, darle clic en el botón "**Buscar dispositivos Bluetooth**"

te aparecerá una lista con los dispositivos emparejados. Elige el nombre del dispositivo en este caso es "**LicornioTECH**".

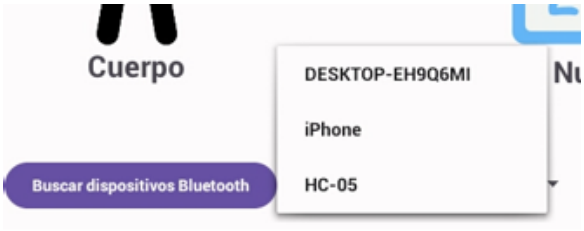


Figura A. 10 Botón buscar dispositivos emparejados

Cuando hayas realizado todos los pasos anteriores darle clic en el botón "**Conectar**" espera unos segundos y aparecerá un círculo de color rojo en la pantalla el cual puedes moverlo utilizando el dispositivo mouse. Tambien, una vez que deseas salir del la aplicación darle en "**Desactivar**".

Figura A. 11 Botones Conectar y desactivar con el dispositivo mouse

Si deseas Seleccionar un elemento de la pantalla posiciónate en dicho elemento durante 3 segundos y este se seleccionará automáticamente.

ANEXO B: PROGRAMA EN ARDUINO PARA ENVIAR DATOS DE MOVIMIENTO

Lectura de la aceleración en los ejes X e Y, aplicación de filtro promedio a las lecturas para suavizarlas, y luego envíarlas a través de Bluetooth.

```
1 //LIBRERIAS UTILIZADAS
2 #include "I2Cdev.h"
3 #include "MPU6050.h"
4 #include "Wire.h"
5 #include <SoftwareSerial.h>
6 //DECLARACION DE VARIABLES GLOBALES
7 ////TXD 10; RXD 11
8 SoftwareSerial blue(10,11);
9 MPU6050 mpu;
10 int16_t ax, ay, az, gx, gy, gz;
11 const int numReadings=10;
12 int readings[numReadings];
13 int index=0;
14 int total=0;
15 int averageX=0;
16 const int numReadings2=10;
17 int readings2[numReadings2];
18 int index2=0;
19 int total2=0;
20 int averageY=0;
21 //FUNCION  setup()
22 void setup(){
23   Serial.begin(9600);
24   Wire.begin();         //Iniciando I2C
25   mpu.initialize();    //Iniciando el sensor
26   blue.begin(9600);
27
28   //LLENA LOS ARREGLOS DE LECTURAS CON CEROS
29   for (int thisReading=0; thisReading < numReadings; thisReading ++)
30   readings[thisReading]=0;
31   for (int thisReading2=0; thisReading2 < numReadings2; thisReading2 ++)
32   readings2[thisReading2]=0;
33   //VERIFICA LA CONEXION CON EL SENSOR USANDO mpu.testConnection()
34   if(!mpu.testConnection()){
35     while (1);
36   }
37 }
38 //FUNCION loop()
39 void loop(){
40   //OBTIENE LAS LECTURAS DE ACELERACION DEL SENSOR CON mpu.getMotion6()//
```

```
41  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
42  ax= -(ax/100);
43  ay= (ay/100);
44
45  //APLICA FILTRO PROMEDIO A LAS LECTURAS DE LOS EJES X e Y
46  ////////Filtro X///////
47  total=total-readings[index];
48  readings[index]=-ax;
49  total=total+readings[index];
50  index=index+1;
51
52  if(index >= numReadings)
53  index=0;
54  averageX=total/numReadings;
55
56  ////////Filtro Y//////////
57  total2=total2-readings2[index2];
58  readings2[index2]=ay;
59  total2=total2+readings2[index2];
60  index2=index2+1;
61
62  if(index2 >= numReadings2)
63  index2=0;
64  averageY=total2/numReadings2;
65
66  //IMPRIMIR LAS LECTURAS PROMEDIADAS POR EL MONITOR SERIAL
67  Serial.print("Eje␣X:␣"); Serial.print("\t");
68  Serial.print(averageX);  Serial.print("\t");
69  Serial.print("␣--␣");    Serial.print("\t");
70  Serial.print("Eje␣Y:␣"); Serial.print("\t");
71  Serial.println(averageY);
72  //ENVIA LAS LECTURAS PROMEDIADAS AL DISPOSITIVO BLUETOOTH MEDIANTE LA COMUNICACION SERIAL
73  blue.print(averageY);
74  blue.print(",");
75  blue.print(averageX);
76 }
```

ANEXO C: CÓDIGO DE LA APLICACÓN EN ANDORID STUDIO

Código del MainActivity.kt:

```
package ec.repositoriocompartido.colorsapplication
//Importar librerias
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.LinearLayout
import android.Manifest
import android.bluetooth.BluetoothAdapter
import android.bluetooth.BluetoothDevice
import android.bluetooth.BluetoothSocket
import android.content.ContentValues.TAG
import android.content.pm.PackageManager
import android.graphics.Rect
import android.os.Handler
import android.os.Looper
import android.util.Log
import android.view.View
import android.view.ViewTreeObserver
import android.widget.*
import androidx.constraintlayout.widget.ConstraintLayout
import androidx.core.app.ActivityCompat
import java.io.IOException
import java.io.InputStream
import java.util.*

class MainActivity : AppCompatActivity() {
    // Declaracion de variables y objetos
    private var canClick = true // Variable de control de clics
    private var activityOpened = false
    private val clickHandler = Handler(Looper.getMainLooper())
    private var lastCollidedLayout: LinearLayout? = null
    private val collisionStartTimeMap =
    mutableMapOf<LinearLayout, Long>()
    private val clickedLayoutSet = mutableSetOf<LinearLayout>()

    private lateinit var opcion1: LinearLayout
    private lateinit var opcion2: LinearLayout
    private lateinit var opcion3: LinearLayout
    private lateinit var opcion4: LinearLayout
    private lateinit var opcion5: LinearLayout
    private lateinit var opcion6: LinearLayout
    // Inicializacion de objetos y variables de Bluetooth
```

```
44    val REQUEST_ENABLE_BT = 1
45    lateinit var mBtAdapter: BluetoothAdapter
46    var mAddressDevices: ArrayAdapter<String>? = null
47    var mNameDevices: ArrayAdapter<String>? = null
48    ////////////////////////////////////////////////
49    private lateinit var circleView: ImageView
50    private lateinit var cursorHandler: Handler
51    private lateinit var inputStream: InputStream
52    private var m_bluetoothSocket: BluetoothSocket? = null
53    ////////////////////////////////////////////////
54    companion object {
55        var m_myUUID: UUID =
56        UUID.fromString("00001101-0000-1000-8000-00805F9B34FB")
57        var m_isConnected: Boolean = false
58        lateinit var m_address: String
59    }
60    override fun onCreate(savedInstanceState: Bundle?) {
61        super.onCreate(savedInstanceState)
62        setContentView(R.layout.activity_main)
63
64        circleView = findViewById(R.id.circleView)
65        mAddressDevices =
66        ArrayAdapter(this, android.R.layout.simple_list_item_1)
67        mNameDevices =
68        ArrayAdapter(this, android.R.layout.simple_list_item_1)
69
70        ////////////////////Hacer imagen visible
71        val imagenVisible: ImageView =
72        findViewById(R.id.circleView)
73        // Referencias a elementos de la interfaz de usuario
74        val idBtnBuscarBT= findViewById<Button>(R.id.idBtnBuscarBT)
75        val idSpinDisp= findViewById<Spinner>(R.id.idSpinDisp)
76        val idBtnConect= findViewById<Button>(R.id.idBtnConect)
77        val idBtnActivar= findViewById<Button>(R.id.idBtnActivar)
78        val idBtnDesact= findViewById<Button>(R.id.idBtnDesact)
79        // Configuracion de listeners para las opciones
80        opcion1= findViewById(R.id.op1)
81        opcion2= findViewById(R.id.op2)
82        opcion3= findViewById(R.id.op3)
83        opcion4= findViewById(R.id.op4)
84        opcion5= findViewById(R.id.op5)
85        opcion6= findViewById(R.id.op6)
86
87        opcion1.setOnClickListener {
88            navegar(CatalogoActivity::class.java,"op1")
89        }
90        opcion2.setOnClickListener {
```

```
91              navegar(CatalogoActivity::class.java,"op2")
92          }
93          opcion3.setOnClickListener {
94              navegar(CatalogoActivity::class.java,"op3")
95          }
96          opcion4.setOnClickListener {
97              navegar(CatalogoActivity::class.java,"op4")
98          }
99          opcion5.setOnClickListener {
100             navegar(CatalogoActivity::class.java,"op5")
101         }
102         opcion6.setOnClickListener {
103             navegar(CatalogoActivity::class.java,"op6")
104         }
105
106         // Inicializacion de cursorHandler
107         cursorHandler = Handler(Handler.Callback {
108             try {
109                 val datos = it.obj as String
110                 processBluetoothData(datos)
111             } catch (e: Exception) {
112                 Log.e("BluetoothData",
113                 "Error␣al␣procesar␣datos:␣${e.message}")
114             }
115             true
116         })
117         //-----------------------------------------------------
118         //-----------------------------------------------------
119         val someActivityResultLauncher =
120         registerForActivityResult(StartActivityForResult()
121         ) { result ->
122             if (result.resultCode == REQUEST_ENABLE_BT) {
123                 Log.i("MainActivity", "ACTIVIDAD␣REGISTRADA")
124             }
125         }
126         //Inicializacion del bluetooth adapter
127         mBtAdapter = BluetoothAdapter.getDefaultAdapter()
128
129         //Checar si esta encendido o apagado
130         if (mBtAdapter == null) {
131             Toast.makeText(this,
132             "Bluetooth␣no␣esta␣disponible␣en␣este␣dipositivo", Toast.LENGTH_LONG).show()
133         } else {
134             Toast.makeText(this,
135             "Bluetooth␣esta␣disponible␣en␣este␣dispositivo", Toast.LENGTH_LONG).show()
136         }
137         //-----------------------------------------------------
```

```
138         //-------------------------------------------------
139
140         // Configuracion de botones para
141         encender y buscar dispositivos Bluetooth
142         idBtnActivar.setOnClickListener {
143             if (mBtAdapter.isEnabled) {
144                 //Si ya esta activado
145                 Toast.makeText(this,
146                 "Bluetooth ya se encuentra activado", Toast.LENGTH_LONG).show()
147             } else {
148                 //Encender Bluetooth
149                 val enableBtIntent =
150                 Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE)
151                 if (ActivityCompat.checkSelfPermission(
152                         this,
153                         Manifest.permission.BLUETOOTH_CONNECT
154                     ) != PackageManager.PERMISSION_GRANTED
155                 ) {
156                     Log.i("MainActivity",
157                     "ActivityCompat#requestPermissions")
158                 }
159                 someActivityResultLauncher.launch(enableBtIntent)
160             }
161
162
163         }
164
165         idBtnBuscarBT.setOnClickListener {
166             if (mBtAdapter.isEnabled) {
167
168                 val pairedDevices: Set<BluetoothDevice>? =
169                 mBtAdapter?.bondedDevices
170                 mAddressDevices!!.clear()
171                 mNameDevices!!.clear()
172
173                 pairedDevices?.forEach { device ->
174                     val deviceName = device.name
175                     val deviceHardwareAddress =
176                     device.address // MAC address
177                     mAddressDevices!!.add(deviceHardwareAddress)
178                 }
179                 //ACTUALIZAR LOS DISPOSITIVOS
180                 idSpinDisp.setAdapter(mNameDevices)
181             } else {
182                 val noDevices
183                 = "Ningun dispositivo pudo ser emparejado"
184                 mAddressDevices!!.add(noDevices)
```

```
185                 mNameDevices!!.add(noDevices)
186                 Toast.makeText(this,
187                 "Primero␣vincule␣un␣dispositivo␣bluetooth",
188                 Toast.LENGTH_LONG).show()
189             }
190         }
191         idBtnConect.setOnClickListener {
192             try {
193                 if (m_bluetoothSocket ==
194                 null || !m_isConnected) {
195                     val IntValSpin =
196                     idSpinDisp.selectedItemPosition
197                     m_address =
198                     mAddressDevices!!.getItem(IntValSpin).
199                     toString()
200                     Toast.makeText
201                     (this,m_address,Toast.LENGTH_LONG).show()
202                     mBtAdapter?.cancelDiscovery()
203                     val device: BluetoothDevice = mBtAdapter.getRemoteDevice(m_address)
204                     m_bluetoothSocket =
205                     device.
206                     createInsecureRfcommSocketToServiceRecord(m_myUUID)
207                     m_bluetoothSocket!!.connect()
208
209                 }
210
211                 Toast.makeText(this,
212                 "CONEXION␣EXITOSA",Toast.LENGTH_LONG).show()
213                 Log.i("MainActivity", "CONEXION␣EXITOSA")
214 ////////////////////////////////////////////////////
215                 DatosCompartidos.bluetoothActivado.value =
216                 true /////activar visibilidad en catalogo
217                 // Cambia la visibilidad de la imagen
218                 if (imagenVisible.visibility == View.VISIBLE) {
219                     imagenVisible.visibility = View.GONE
220                 } else {
221                     imagenVisible.visibility = View.VISIBLE
222                 }
223                 ////////////////////////////////////
224                 inputStream =
225                 m_bluetoothSocket?.inputStream ?: throw IOException("Input␣stream␣is␣null.")
226                 beginListenForData()
227
228
229                 ////////////////////////////////////
230
231             } catch (e: IOException) {
```

```
232                 //connectSuccess = false
233                 e.printStackTrace()
234                 Toast.makeText(this,"ERROR␣DE␣CONEXION",Toast.LENGTH_LONG).show()
235                 Log.i("MainActivity", "ERROR␣DE␣CONEXION")
236             }
237         }
238 
239         //Boton apagar bluetooth
240         idBtnDesact.setOnClickListener {
241             if (!mBtAdapter.isEnabled) {
242                 //Si ya esta desactivado
243                 Toast.makeText(this,
244                 "Bluetooth␣ya␣se␣encuentra␣desactivado", Toast.LENGTH_LONG).show()
245             } else {
246                 //Encender Bluetooth
247                 mBtAdapter.disable()
248                 Toast.makeText(this, "Se␣ha␣desactivado␣el␣bluetooth", Toast.LENGTH_LONG).show()
249             }
250         }
251         // Observador para cambios en la posicion de circleView
252     circleView.viewTreeObserver.addOnPreDrawListener(object : ViewTreeObserver.OnPreDrawListener {
253             override fun onPreDraw(): Boolean {
254                 checkCollisionAndClick()
255                 return true
256             }
257         })
258     }
259     // Metodo para iniciar la escucha de datos de Bluetooth
260     private fun beginListenForData() {
261         Thread {
262             val buffer = ByteArray(1024)
263             var bytes: Int
264 
265             while (true) {
266                 try {
267                     bytes = inputStream.read(buffer)
268                     val readMessage = String(buffer, 0, bytes)
269                     cursorHandler.obtainMessage(0, bytes, -1, readMessage).sendToTarget()
270                     //Log.d("BluetoothData", "Datos recibidos: $readMessage")
271                 } catch (e: IOException) {
272                     Log.e(TAG, "Error␣reading␣from␣Bluetooth␣input␣stream:␣${e.message}")
273                     break
274                 }
275             }
276         }.start()
277     }
278     // Procesamiento de datos de Bluetooth recibidos
```

```
279     fun processBluetoothData(datos: String) {
280         val parts = datos.split(",")
281         if (parts.size == 2) {
282             val ax = parts[0].toInt()
283             val ay = parts[1].toInt()
284 
285             val factorDeSensibilidad = 30
286             val nuevaPosicionX = ax * factorDeSensibilidad
287             val nuevaPosicionY = ay * factorDeSensibilidad
288         DatosCompartidos.dato1.postValue(nuevaPosicionX)
289         //Datos para pasar datos de bluetooth a otra actividad
290             DatosCompartidos.dato2.postValue(nuevaPosicionY)
291             runOnUiThread {
292                 moverCirculo(nuevaPosicionX, nuevaPosicionY)
293             }
294             //Log.d("BluetoothData", "Datos recibidos: ${parts.size}")
295         }
296     }
297     // Metodo para mover el circulo en la interfaz de usuario
298     private fun moverCirculo(nuevaPosicionX: Int, nuevaPosicionY: Int) {
299         // Actualiza las coordenadas de la vista del circulo
300         val layoutParams = circleView.layoutParams as ConstraintLayout.LayoutParams
301         layoutParams.leftMargin = nuevaPosicionX
302         layoutParams.topMargin = nuevaPosicionY
303         circleView.layoutParams = layoutParams
304 
305     }
306     // Metodo para verificar si dos vistas se estan superponiendo
307     private fun isViewOverlapping(firstView: View, secondView: View): Boolean {
308         val rectFirstView = Rect()
309         firstView.getGlobalVisibleRect(rectFirstView)
310 
311         val rectSecondView = Rect()
312         secondView.getGlobalVisibleRect(rectSecondView)
313 
314         return rectFirstView.intersect(rectSecondView)
315     }
316     // Metodo para verificar colisiones y esperar clics
317     private fun checkCollisionAndClick() {
318         val currentTime = System.currentTimeMillis()
319         // Verificar colisiones
320         for (layout in listOf(opcion1, opcion2, opcion3, opcion4, opcion5, opcion6)) {
321             if (isViewOverlapping(circleView, layout)) {
322                 // Verificar si la colision esta registrada
323                 if (lastCollidedLayout == layout) {
324                     val collisionStartTime = collisionStartTimeMap[layout] ?: 0
325                     if (currentTime - collisionStartTime >= 3000 && layout !in clickedLayoutSet) {
```

```
                        waitForClick(layout)
                    }
                } else {
                    // Iniciar el tiempo de espera para la nueva colision
                    lastCollidedLayout = layout
                    collisionStartTimeMap[layout] = currentTime
                }
                return
            }
        }
        // Si el CircleView no esta en colision con ningun LinearLayout, reiniciar el tiempo
        lastCollidedLayout = null
    }
    // Metodo para esperar un clic despues de una colision
    private fun waitForClick(linearLayout: LinearLayout) {
        // Despues de la espera de 3 segundos, realiza el clic en el LinearLayout
        clickHandler.post {
            activityOpened = true
            canClick = false
            linearLayout.performClick()
            clickedLayoutSet.add(linearLayout)
        }
    }
    // Reinicia las variables cuando regresas a la actividad principal
    override fun onResume() {
        super.onResume()
        activityOpened = false
        canClick = true
        lastCollidedLayout = null
        collisionStartTimeMap.clear()
        clickedLayoutSet.clear()
    }
    // Metodo para navegar a la actividad de catalogo
    fun navegar(clase:Class<*>,texto:String){
        val intent= Intent(this,clase)
        intent.putExtra("texto",texto)
        startActivity(intent)
    }
}
```

Código del CatalogoActivity.kt:

```
import android.app.Activity
import android.content.Context
import android.content.Intent
import android.graphics.Rect
```

```
import android.media.MediaPlayer
import android.os.Build
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler
import android.view.LayoutInflater
import android.view.ViewTreeObserver
import android.widget.ImageView
import android.widget.SeekBar
import android.widget.TextView
import androidx.appcompat.app.AlertDialog
import androidx.constraintlayout.widget.ConstraintLayout
import androidx.lifecycle.Observer
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import android.view.View
import java.util.Timer

class CatalogoActivity : AppCompatActivity() , ItemClickListener{

    // Mapa para almacenar el tiempo de inicio de espera por tipo de colision
    private val collisionStartTimeMap: MutableMap<String, Long> = mutableMapOf()
    private var currentCollisionType: String? = null
    ////////////////////////////////////////
    private lateinit var itemsList: List<Item>
    private var mediaPlayer: MediaPlayer? = null
    // Tiempo de espera por colision en milisegundos
    private var collisionClickDelay: Long = 3000
    private var collisionTimer: Timer? = null

    ////////////////////////////////////////////////
    private lateinit var circleView: ImageView
    private var waitingForClick = false

    // Declara variables para almacenar los datos que llegan del MainActivity
    private var dato1: Int = 0
    private var dato2: Int = 0
    ////////////////////////////////////////////
    private var canClick = true
    private var activityOpened = false
    ////////////////////////////////
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_catalogo)

        circleView = findViewById(R.id.circleView)
        ////////////////Hacer imagen visible
```

```
52          // Verifica si Bluetooth esta activado (segun la variable en DatosCompartidos)
53          if (DatosCompartidos.bluetoothActivado.value == true) {
54              circleView.visibility = View.VISIBLE
55          } else {
56              circleView.visibility = View.GONE
57          }
58
59          //METODO PARA REGRESAR AL MENU PRINCIPAL
60
61          val regreso= findViewById<ImageView>(R.id.regresar)
62
63          regreso.setOnClickListener {
64              val intent= Intent(this,MainActivity::class.java)
65              startActivity(intent)
66              finish() // Cierra la actividad actual
67          }
68          //FIN METODOS PARA EL REGRESO
69          //Datos que vienen del MainActivity para mover cursor
70          // Observar los LiveData
71          DatosCompartidos.dato1.observe(this, Observer { nuevoDato1 ->
72              dato1 = nuevoDato1 // Actualiza la variable dato1 con el nuevoDato1
73              moverCirculo(dato1, dato2)
74          })
75          DatosCompartidos.dato2.observe(this, Observer { nuevoDato2 ->
76              dato2 = nuevoDato2 // Actualiza la variable dato2 con el nuevoDato2
77              moverCirculo(dato1, dato2)
78          })
79          //Fin variables para mover cursor
80          //AQUI SE RECUPERA LA OPCION QUE VIENE DEL MAINACTIVITY Y
81          // DEPENDIENDO DE ESO SE CREA EL ARREGLO CON EL NOMBRE DE LAS IMAGENES
82          val textoEnviado:String=intent.extras?.getString("texto").orEmpty()
83          // Opcion por defecto si no coincide con ninguna de las anteriores
84          println("Opcion␣no␣valida"+textoEnviado)
85
86          when (textoEnviado) {
87              "op1" -> {
88                  itemsList=listOf(
89                      Item( "op11"),
90                      Item( "op12"),
91                      Item( "op13"),
92                      Item( "op14"),
93                      Item( "op15"),
94                      Item( "op16"),
95                      Item( "op17"),
96                      Item( "op18"),
97                      Item( "op19"),
98                      Item( "op110"),
```

```
99                  Item( "op111"),
100                 Item( "op112"),
101                 Item( "op113"),
102                 Item( "op114"),
103                 Item( "op115"),
104                 Item( "op116"),
105             )
106
107         }
108         "op2" -> {
109             itemsList=listOf(
110                 Item( "op21"),
111                 Item( "op22"),
112                 Item( "op23"),
113                 Item( "op24"),
114                 Item( "op25"),
115                 Item( "op26"),
116                 Item( "op27"),
117                 Item( "op28"),
118                 Item( "op29"),
119                 Item( "op210"),
120                 Item( "op211"),
121                 Item( "op212"),
122                 Item( "op213"),
123                 Item( "op214"),
124                 Item( "op215"),
125                 Item( "op216"),
126             )
127
128
129         }
130         "op3" -> {
131             itemsList=listOf(
132                 Item( "op31"),
133                 Item( "op32"),
134                 Item( "op33"),
135                 Item( "op34"),
136                 Item( "op35"),
137                 Item( "op36"),
138                 Item( "op37"),
139                 Item( "op38"),
140                 Item( "op39"),
141                 Item( "op310"),
142                 Item( "op311"),
143                 Item( "op312"),
144                 Item( "op313"),
145                 Item( "op314"),
```

```
146                     Item( "op315"),
147                     Item( "op316"),
148                 )
149 
150             }
151             "op4" -> {
152                 itemsList=listOf(
153                     Item( "op41"),
154                     Item( "op42"),
155                     Item( "op43"),
156                     Item( "op44"),
157                     Item( "op45"),
158                     Item( "op46"),
159                     Item( "op47"),
160                     Item( "op48"),
161                     Item( "op49"),
162                     Item( "op410"),
163                     Item( "op411"),
164                     Item( "op412"),
165                     Item( "op413"),
166                     Item( "op414"),
167                     Item( "op415"),
168                     Item( "op416"),
169                 )
170             }
171             "op5" -> {
172                 itemsList=listOf(
173                     Item( "op51"),
174                     Item( "op52"),
175                     Item( "op53"),
176                     Item( "op54"),
177                     Item( "op55"),
178                     Item( "op56"),
179                     Item( "op57"),
180                     Item( "op58"),
181                     Item( "op59"),
182                     Item( "op510"),
183                     Item( "op511"),
184                     Item( "op512"),
185                     Item( "op513"),
186                     Item( "op514"),
187                     Item( "op515"),
188                     Item( "op516"),
189                 )
190             }
191             "op6" -> {
192                 itemsList=listOf(
```

```
                Item( "op61"),
                Item( "op62"),
                Item( "op63"),
                Item( "op64"),
                Item( "op65"),
                Item( "op66"),
                Item( "op67"),
                Item( "op68"),
                Item( "op69"),
                Item( "op610"),
                Item( "op611"),
                Item( "op612"),
                Item( "op613"),
                Item( "op614"),
                Item( "op615"),
                Item( "op616"),
            )

        }
        else -> {
            itemsList=listOf(
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
                Item( "love"),
            )

        }
    }
    val rvOpciones = findViewById<RecyclerView>(R.id.rvOpciones)
    val categoriasAdapter = ItemsAdapter(itemsList, this@CatalogoActivity)
    val gridLayoutManager =
        GridLayoutManager(this, 2,
            GridLayoutManager.HORIZONTAL, false)
    rvOpciones.layoutManager = gridLayoutManager
```

```
240         rvOpciones.adapter = categoriasAdapter
241         ////////////////////////////////////////////
242         circleView.viewTreeObserver.addOnPreDrawListener(object :
243             ViewTreeObserver.OnPreDrawListener {
244             override fun onPreDraw(): Boolean {
245                 if (canClick && !activityOpened) {
246                     val layoutManager = (rvOpciones.layoutManager as GridLayoutManager)
247                     val firstVisiblePosition = layoutManager.findFirstVisibleItemPosition()
248                     val lastVisiblePosition = layoutManager.findLastVisibleItemPosition()
249 
250                     var foundCollision = false
251 
252                     for (i in firstVisiblePosition..lastVisiblePosition) {
253                         val view = layoutManager.findViewByPosition(i)
254                         if (view != null && isViewOverlapping(circleView, view)) {
255                             val itemType = getItemType(textoEnviado, i)
256                             foundCollision = true
257                             handleCollision(itemType)
258                             break
259                         }
260                     }
261 
262                     if (!foundCollision) {
263                         // Si no hay colision, reiniciar el temporizador
264                         resetCollisionTimer()
265                     }
266                 }
267                 return true
268             }
269         })
270         ////////////////////////////////////////////////
271 
272     }
273 
274     override fun onItemClick(item: Item) {
275         if (!activityOpened) {
276             if (!isFinishing && !isActivityDestroyed(this)) {
277                 mostrarPopup(this, item)
278                 activityOpened = true
279                 canClick = false
280             }
281         }
282 
283     }
284     private fun mostrarPopup(context: Context, item: Item) {
285 
286             val inflater = LayoutInflater.from(context)
```

```
287             val popupView = inflater.inflate(R.layout.popuop_layoutt, null)
288
289             val popupImageView: ImageView = popupView.findViewById(R.id.popupImageView)
290             val popupSeekBar: SeekBar = popupView.findViewById(R.id.seekBar)
291
292             // Obtiene la ruta de la imagen y el sonido desde el objeto Item
293             val rutaImagen = item.imagen
294             val rutaSonido = item.imagen
295
296             // Carga la imagen en el ImageView
297             cargarImagenDesdeRuta(context, rutaImagen, popupImageView)
298
299             // Configura el SeekBar para el sonido
300             configurarSeekBar(context, rutaSonido, popupSeekBar)
301
302             // Construye y muestra el cuadro de dialogo
303             val alertDialogBuilder = AlertDialog.Builder(context)
304                 .setView(popupView)
305                 .setPositiveButton("Aceptar") { dialog, _ ->
306                     detenerReproduccion()
307                     dialog.dismiss()
308
309                     // Restablece las variables al cerrar la imagen
310                     activityOpened = false
311                     canClick = true
312                 }
313
314             val alertDialog = alertDialogBuilder.create()
315             alertDialog.show()
316     }
317
318     private fun cargarImagenDesdeRuta(context: Context, rutaImagen: String, imageView: ImageView) {
319         val resourceId =
320             context.resources.getIdentifier(rutaImagen, "drawable", context.packageName)
321         imageView.setImageResource(resourceId)
322     }
323
324     private fun configurarSeekBar(context: Context, rutaSonido: String, seekBar: SeekBar) {
325         val idSonido = obtenerIdSonido(context, rutaSonido)
326         val duracion = obtenerDuracionSonido(context, idSonido)
327
328         seekBar.max = duracion
329
330         mediaPlayer = MediaPlayer.create(context, idSonido)
331
332         seekBar.setOnSeekBarChangeListener(object : SeekBar.OnSeekBarChangeListener {
333             override fun onProgressChanged(seekBar: SeekBar?, progress: Int, fromUser: Boolean) {
```

```
334                 if (fromUser) {
335                     mediaPlayer?.seekTo(progress)
336                 }
337             }
338 
339             override fun onStartTrackingTouch(seekBar: SeekBar?) {
340                 // No es necesario implementar nada aqui
341             }
342 
343             override fun onStopTrackingTouch(seekBar: SeekBar?) {
344                 // No es necesario implementar nada aqui
345             }
346         })
347 
348         mediaPlayer?.start()
349 
350         // Inicializa el hilo para actualizar la posicion del SeekBar
351         actualizarSeekBar(seekBar)
352     }
353     private fun obtenerIdSonido(context: Context, rutaSonido: String): Int {
354         return context.resources.getIdentifier(rutaSonido, "raw", context.packageName)
355     }
356     private fun obtenerDuracionSonido(context: Context, idSonido: Int): Int {
357         val mediaPlayer = MediaPlayer.create(context, idSonido)
358         val duracion = mediaPlayer?.duration ?: 0
359         mediaPlayer?.release()
360         return duracion
361     }
362     private fun detenerReproduccion() {
363         mediaPlayer?.release()
364         mediaPlayer = null
365         collisionTimer?.cancel()
366     }
367 
368     private fun actualizarSeekBar(seekBar: SeekBar) {
369         val handler = Handler()
370 
371         // Este Runnable se ejecutara cada 100 milisegundos
372         val runnable = object : Runnable {
373             override fun run() {
374                 // Verifica si el MediaPlayer esta inicializado y esta reproduciendo
375                 if (mediaPlayer != null && mediaPlayer!!.isPlaying) {
376                     val currentPosition = mediaPlayer!!.currentPosition
377                     seekBar.progress = currentPosition
378                     // Programa la proxima actualizacion
379                     handler.postDelayed(this, 100)
380                 } else {
```

```
381                     // Detiene la actualizacion cuando la reproduccion se detiene
382                     handler.removeCallbacks(this)
383                 }
384             }
385         }
386         handler.post(runnable)  // Inicia el proceso de actualizacion
387     }
388     ////////////////////////////////////////////////////////////7
389     private fun moverCirculo(nuevaPosicionX: Int, nuevaPosicionY: Int) {
390         // Actualiza las coordenadas de la vista del circulo
391         val layoutParams = circleView.layoutParams as ConstraintLayout.LayoutParams
392         layoutParams.leftMargin = nuevaPosicionX
393         layoutParams.topMargin = nuevaPosicionY
394         circleView.layoutParams = layoutParams
395     }
396     ////////////////////////////////////////////////////////////////
397     private fun isViewOverlapping(firstView: View, secondView: View): Boolean {
398         val rectFirstView = Rect()
399         firstView.getGlobalVisibleRect(rectFirstView)
400 
401         val rectSecondView = Rect()
402         secondView.getGlobalVisibleRect(rectSecondView)
403 
404         return rectFirstView.intersect(rectSecondView)
405     }
406 
407     private fun isActivityDestroyed(activity: Activity): Boolean {
408         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1) {
409             return activity.isDestroyed
410         } else {
411             return activity.window == null || activity.window.decorView == null
412         }
413     }
414     private fun getItemType(textoEnviado: String, position: Int): String {
415         // Logica para determinar el tipo de item en funcion del textoEnviado y la posicion
416         // Puedes ajustar esta logica segun sea necesario
417         return when (textoEnviado) {
418             "op1" -> "op1_$position"
419             "op2" -> "op2_$position"
420             "op3" -> "op3_$position"
421             "op4" -> "op4_$position"
422             "op5" -> "op5_$position"
423             "op6" -> "op6_$position"
424             else -> "default_$position"
425         }
426     }
427     private fun handleCollision(itemType: String) {
```

```
428         val currentTimeMillis = System.currentTimeMillis()
429 
430         if (itemType != currentCollisionType) {
431             // Ha cambiado de colision, reiniciar el temporizador
432             resetCollisionTimer()
433         }
434 
435         if (collisionStartTimeMap.containsKey(itemType)) {
436             val startTime = collisionStartTimeMap[itemType] ?: 0
437 
438             if (currentTimeMillis - startTime >= collisionClickDelay) {
439                 // Se ha pasado el tiempo de espera, ejecutar el clic
440                 waitingForClick = false
441                 onItemClick(itemsList[getPositionFromItemType(itemType)])
442                 // Limpia el registro de tiempo para este tipo de colision
443                 collisionStartTimeMap.remove(itemType)
444                 // Actualiza el tipo de colision actual
445                 currentCollisionType = itemType
446             }
447         } else {
448             // Primer encuentro con este tipo de item, almacenar el tiempo de inicio
449             collisionStartTimeMap[itemType] = currentTimeMillis
450             waitingForClick = true
451             // Actualiza el tipo de colision actual
452             currentCollisionType = itemType
453         }
454     }
455     private fun resetCollisionTimer() {
456         collisionStartTimeMap.clear()
457         currentCollisionType = null
458     }
459     private fun getPositionFromItemType(itemType: String): Int {
460         // Extraer la posicion del itemType (puedes ajustar la logica segun sea necesario)
461         val parts = itemType.split("_")
462         return parts.last().toIntOrNull() ?: 0
463     }
464 
465 }
```

Código del LoginActivity.kt:

```
1 import android.os.Bundle
2 import android.content.Intent
3 import android.util.Log
4 import android.widget.Toast
5 import android.widget.Button
```

```
6 import android.widget.EditText
7 import androidx.appcompat.app.AppCompatActivity
8
9 class LoginActivity : AppCompatActivity(){
10
11     // Credenciales fijas para demostracion
12     private val usuarioCorrecto = "ProyectoLicornio"
13     private val contraseaCorrecta = "tesis"
14
15     override fun onCreate(savedInstanceState: Bundle?) {
16         super.onCreate(savedInstanceState)
17         setContentView(R.layout.activity_login)
18
19         // Referencias a elementos de la interfaz de usuario
20         val idTextUsername = findViewById<EditText>(R.id.idTextUsername)
21         val idTextPassword = findViewById<EditText>(R.id.idTextPassword)
22         val idBtnLogin = findViewById<Button>(R.id.idBtnLogin)
23
24         // Configuracion del boton de inicio de sesion
25         idBtnLogin.setOnClickListener {
26             val username = idTextUsername.text.toString()
27             val password = idTextPassword.text.toString()
28
29             // Verificar si los campos de usuario y contrasena no estan vacios
30             if (username.isNotEmpty() && password.isNotEmpty()) {
31                 // Verificar credenciales
32                 if (validarCredenciales(username, password)) {
33                     // Credenciales validas, abre la MainActivity
34                     val intent= Intent(this,MainActivity::class.java)
35                     startActivity(intent)
36                     Log.d("MiApp", "Iniciando␣MainActivity")
37                     finish() // Cierra la actividad actual
38                 } else {
39                     showToast("Credenciales␣incorrectas")
40                 }
41             } else {
42                 showToast("Por␣favor,␣completa␣todos␣los␣campos")
43             }
44         }
45     }
46
47     // Metodo para verificar las credenciales
48     private fun validarCredenciales(username: String, password: String): Boolean {
49         return username == usuarioCorrecto && password == contrasenaCorrecta
50     }
51
52     // Metodo para mostrar un Toast con un mensaje
```

```
53     private fun showToast(message: String) {
54         Toast.makeText(this, message, Toast.LENGTH_SHORT).show()
55     }
56 }
```